

The background of the entire page is a high-magnification, close-up photograph of soil. It shows a dense field of small, irregularly shaped particles in various shades of brown, tan, and dark grey. The lighting creates highlights and shadows, giving the particles a three-dimensional appearance.

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
----------	--	-----------

7.0.1	Electrical design	23
7.0.2	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	SI equivalent diameter	28
-------	----------------------------------	----

8.4	Classification	28
------------	-----------------------	-----------

8.4.1	Roundness using Hu-moments	28
-------	--------------------------------------	----

8.4.2	Angularity using a Neural Network	28
-------	---	----

III	Verification
------------	---------------------

9	Presenting Information	31
----------	---	-----------

9.1	Table	31
-----	-------	----

9.2	Figure	31
-----	--------	----

IV	Addenda
-----------	----------------

Bibliography	35
-------------------------------	-----------

Books	35
--------------	-----------

Reports	35
----------------	-----------

Articles	35
-----------------	-----------

Index	37
------------------------	-----------

A	RDM campus: Student project	39
----------	--	-----------

B	Literature review	41
----------	------------------------------------	-----------

C	SoilMath Library	43
----------	-----------------------------------	-----------

C.0.1	Genetic Algorithm Class	43
-------	-----------------------------------	----

C.0.2	Fast Fourier Transform Class	54
-------	--	----

C.0.3	Neural Network Class	61
-------	--------------------------------	----

C.0.4	Statistical Class	69
C.0.5	General project files	85
D	Hardware Library	97
D.0.1	Microscope Class	97
D.0.2	Beaglebone Black Class	107
D.0.3	GPIO Class	111
D.0.4	PWM Class	118
D.0.5	ADC Class	123
D.0.6	EC12P Class	128
D.0.7	eQep Class	132
D.0.8	SoilCape Class	139
D.0.9	USB Class	140
D.0.10	General project files	142
E	Vision Library	149
E.0.1	Image processing Class	149
E.0.2	Conversion Class	157
E.0.3	Enhance Class	164
E.0.4	Morphological filter Class	172
E.0.5	Segment Class	177
E.0.6	General project files	198
F	Analyzer Library	205
F.0.1	Analyzer Class	205
F.0.2	Sample Class	216
F.0.3	Particle Class	221
F.0.4	Settings Class	227
F.0.5	General project files	232
G	QOpenCVQT Library	235
H	QParticleDisplay Library	239
I	QParticleSelector Library	245
J	QReportGenerator Library	251
K	Vision Soil Analyzer Program	265
K.0.1	General project files	265
K.0.2	Main window Class	269
K.0.3	Dialog window Class	285
K.0.4	Dialog Neural Network Class	300



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 describes 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 in to a full fledge 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 describe 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

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt

ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

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

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.

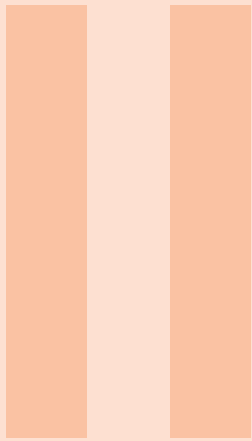
5.3.1 Led driver

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

7.0.1 Electrical design

7.0.2 Design


A close-up, high-magnification image of sand particles. The particles are irregular in shape, ranging from small, rounded grains to larger, more angular fragments. They exhibit a variety of colors, including light beige, off-white, and pale yellow, with some darker, more translucent areas. The texture is granular and uneven.

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.

 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 ($0.2[mm] \leq P_{size} \leq 2[mm]$ 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.

- R** 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 A 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 where 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.

- R** 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

In order to tell something about the individual particles, they first have to be identified and separated from the background. This is done with the enhanced intensity matrix. Which is taken from the bright field matrix or if available the back lit intensity matrix.

Segmentation

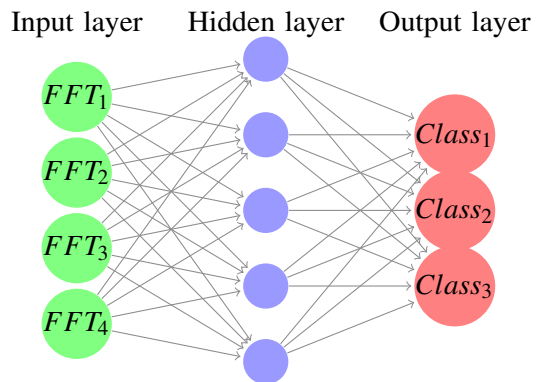
The images are segmented by calculating a threshold value. This value is determined by using the Otsu threshold. Xu et al. [2] describe that the Otsu threshold is equal to the average of the mean levels of two classes partitioned by this threshold. This threshold can be iteratively determined.

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

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

8.3.1 CIE $L^*a^*b^*$ extraction**8.3.2 Fast Fourier Descriptors****8.3.3 SI equivalent diameter****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



This is an example of examples.



Verification

9	Presenting Information	31
9.1	Table	
9.2	Figure	



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

IV

Addenda

	Bibliography	35
	Books	
	Reports	
	Articles	
	Index	37
A	RDM campus: Student project	39
B	Literature review	41
C	SoilMath Library	43
D	Hardware Library	97
E	Vision Library	149
F	Analyzer Library	205
G	QOpenCVQT Library	235
H	QParticleDisplay Library	239
I	QParticleSelector Library	245
J	QReportGenerator Library	251
K	Vision Soil Analyzer Program	265



Bibliography

Books

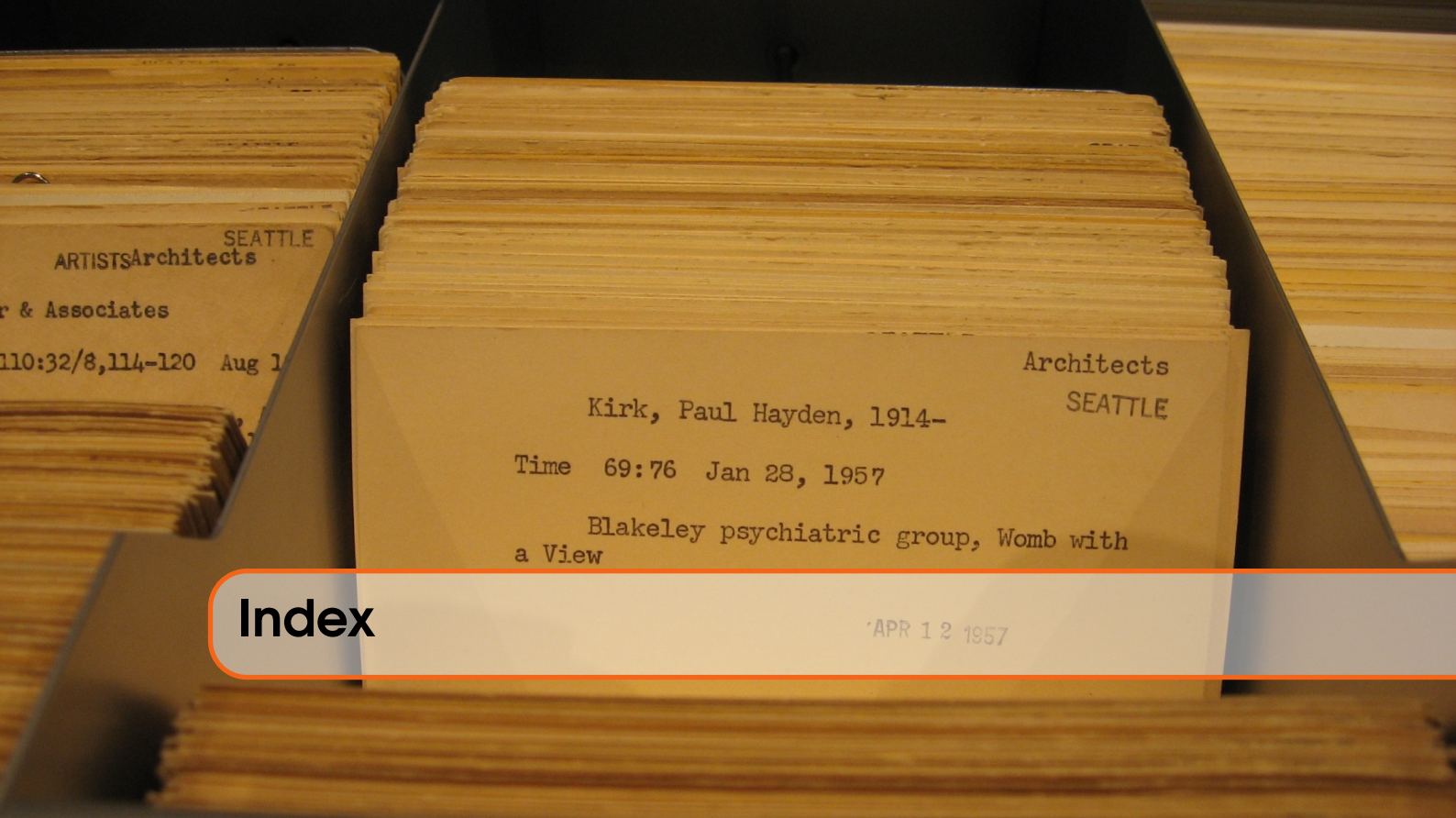
- [3] ir. P.A.C. Ypma. *Course Notes EVD2*. University of applied sciences, Sept. 2, 2014. 71 pages (cited on page 26).

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).



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. 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. It's 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 an 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.



B. Literature review

Literatuurstudie

Optische kenmerken van grond gebruikt bij computer vision

Opdrachtgever: Royal IHC

Opleverdatum: 3 november 2014

Jelle Spijker

Datum 2 november 2014

Revisie 20141102

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) expliciet kenbaar gemaakt in het geciteerde tekstdeel (cursief gedrukt).



C. SoilMath Library

C.0.1 Genetic Algorithm Class

```
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  //! Genetic Algorithmes used for optimization problems
9  /*!
10   * Use this class for optimization problems. It's currently
   optimized for
11   * Neural Network optimization
12   */
13 #pragma once
14
15 #include <bitset>
16 #include <random>
17 #include <string>
18 #include <algorithm>
19 #include <chrono>
20 #include <math.h>
21 #include <list>
22
23 // #include "NN.h"
24 #include "SoilMathTypes.h"
25 #include "MathException.h"
26
27 #include <QtCore/QObject>
28 #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

```

```

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

```

```

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

```

```

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   * destruction
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 closser 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 wich 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 }

```

```

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 "GA.h"
11
12 namespace SoilMath {
13 GA::GA() {}
14
15 GA::GA(NNfunctionType nnfunction, uint32_t inputneurons,
16       uint32_t hiddenneurons,
17       uint32_t outputneurons) {
18     this->NNfuction = nnfunction;
19     this->inputneurons = inputneurons;
20     this->hiddenneurons = hiddenneurons;
21     this->outputneurons = outputneurons;
22 }
23
24 GA::~~GA() {}
25
26 void GA::Evolve(const InputLearnVector_t &inputValues,
27                Weight_t &weights,
28                MinMaxWeight_t rangeweights,
29                OutputLearnVector_t &goal,
30                uint32_t maxGenerations, uint32_t popSize) {
31     minOptim = goal[0].OutputNeurons.size();
32     minOptim = -minOptim;
33     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
    popSize) {
51     if (popSize < 1)
52         return Population_t();
53
54     Population_t pop;
55     unsigned seed = std::chrono::system_clock::now().
        time_since_epoch().count();
56     std::default_random_engine gen(seed);
57     std::uniform_real_distribution<float> dis(rangeweights.
        first,
58
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
        rangeweights.
        second);

    for (uint32_t i = 0; i < popSize; i++) {
        PopMember_t I;
        for (uint32_t j = 0; j < weights.size(); j++) {
            I.weights.push_back(dis(gen));
            I.weightsGen.push_back(
                ConvertToGenome<float>(I.weights[j], rangeweights)
            );
        }
        pop.push_back(I);
    }
    return pop;
}

void GA::CrossOver(Population_t &pop) {
    Population_t newPop; // create a new population
    PopMember_t newPopMembers[2];
    SplitGenome_t Split[2];

    for (uint32_t i = 0; i < pop.size(); i += 2) {
        for (uint32_t j = 0; j < pop[i].weights.size(); j++) {

```

```

80     // Split A
81     Split[0].first = std::bitset<CROSSOVER>(
82         pop[i].weightsGen[j].to_string().substr(0,
83             CROSSOVER));
84     Split[0].second = std::bitset<GENE_MAX - CROSSOVER>(
85         pop[i].weightsGen[j].to_string().substr(CROSSOVER,
86             GENE_MAX -
87
88                 CROSSOVER
89             ));
90
91     // Split B
92     Split[1].first = std::bitset<CROSSOVER>(
93         pop[i + 1].weightsGen[j].to_string().substr(0,
94             CROSSOVER));
95     Split[1].second = std::bitset<GENE_MAX - CROSSOVER>(
96         pop[i + 1].weightsGen[j].to_string().substr(
97             CROSSOVER,
98
99                 GENE_MAX
100                 -
101                 CROSSOVER
102             ));
103
104     // Mate A and B to AB and BA
105     newPopMembers[0].weightsGen.push_back(
106         Genome_t(Split[0].first.to_string() + Split[1].
107             second.to_string()));
108     newPopMembers[1].weightsGen.push_back(
109         Genome_t(Split[1].first.to_string() + Split[0].
110             second.to_string()));
111 }
112 newPop.push_back(newPopMembers[0]);
113 newPop.push_back(newPopMembers[1]);
114 newPopMembers[0].weightsGen.clear();
115 newPopMembers[1].weightsGen.clear();
116 }
117
118 // Allow the top tiers population partners to mate again
119 uint32_t halfN = pop.size() / 2;
120 for (uint32_t i = 0; i < halfN; i++) {
121     for (uint32_t j = 0; j < pop[i].weights.size(); j++) {
122         Split[0].first = std::bitset<CROSSOVER>(
123             pop[i].weightsGen[j].to_string().substr(0,
124                 CROSSOVER));
125         Split[0].second = std::bitset<GENE_MAX - CROSSOVER>(
126             pop[i].weightsGen[j].to_string().substr(CROSSOVER,
127                 GENE_MAX -
128
129                     CROSSOVER
130                 ));
131
132         Split[1].first = std::bitset<CROSSOVER>(
133             pop[i + 2].weightsGen[j].to_string().substr(0,
134                 CROSSOVER));
135         Split[1].second = std::bitset<GENE_MAX - CROSSOVER>(
136             pop[i + 2].weightsGen[j].to_string().substr(

```

```

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

```

```

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

```

```

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         }
239         else if (revolutionOngoing) {
240             qDebug() << "Peace has been restort";
241             Elitisme = oldElit;
242             MutationRate = oldMutation;
243             revolutionOngoing = false;
244         }
245         last10Gen.pop_front();
246         last10Gen.push_back(learnError);
247     } else {
248         last10Gen.push_back(learnError);
249     }
250     currentGeneration++;
251     emit learnErrorUpdate(static_cast<double>(learnError));
252     if (learnError < EndError) {
253         retVal = true;
254     }
255     return retVal;
256 }
257 }

```

C.0.2 Fast Fourier Transform Class

```

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 <vector>
11 #include <complex>
12 #include <cmath>
13 #include <valarray>
14 #include <array>
15 #include <deque>
16 #include <queue>
17 #include <iterator>
18 #include <algorithm>
19 #include <stdint.h>
20 #include <opencv2/core.hpp>
21 #include "SoilMathTypes.h"
22 #include "MathException.h"
23
24 namespace SoilMath {
25     /*!
26     * \brief Fast Fourier Transform class
27     * \details Use this class to transform a black and white
   blob presented as a
28     * cv::Mat with values 0 or 1 to a vector of complex values
   representing the Fourier
29     * Descriptors.
30     */
31     class FFT {
32     public:
33         /*!
34         * \brief Standard constructor
35         */
36         FFT();
37
38         /*!
39         * \brief Standard deconstructor
40         */
41         ~FFT();
42
43         /*!
44         * \brief Transforming the img to the frequency domain and
   returning the
45         * Fourier Descriptors
46         * \param img contour in the form of a cv::Mat type
   CV_8UC1. Which should
47         * 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
      contour in the
50     * frequency domain, expressed as Fourier Descriptors
51     */
52     ComplexVect_t GetDescriptors(const cv::Mat &img);
53
54 private:
55     ComplexVect_t
56         fftDescriptors; /**< Vector with complex values which
      represent the
57                             descriptors*/
58     ComplexVect_t
59         complexcontour; /**< Vector with complex values which
      represent the
60                             contour*/
61     cv::Mat Img;          /**< Img which will be analysed*/
62
63     /*!
64     * \brief Contour2Complex a private function which
      translates a continous
65     * contour image
66     * to a vector of complex values. The contour is found
      using a depth first
67     * search with
68     * extension list. The alghorithm is based upon <a
69     * href="http://ocw.mit.edu/courses/electrical-engineering
      -and-computer-science/6-034-artificial-intelligence-
      fall-2010/lecture-videos/lecture-4-search-depth-first-
      hill-climbing-beam/">MIT
70     * opencourseware
71     * 6-034-artificial-intelligence lecture 4</a>
72     * \param img contour in the form of a cv::Mat type
      CV_8UC1. Which should
73     * consist of a continous contour. \f$ \{ img \in \mathbb{Z} \mid 0 \leq img \leq
74     * 1 \} \f$
75     * \param centerCol centre of the contour X value
76     * \param centerRow centre of the contour Y value
77     * \return a vector with complex values, represing the
      contour as a function
78     */
79     ComplexVect_t Contour2Complex(const cv::Mat &img, float
      centerCol,
80                                     float centerRow);
81
82     /*!
83     * \brief Neighbors a private function returning the
      neighboring pixels which
84     * belong to a contour
85     * \param 0 uchar pointer to the data
86     * \param pixel current counter
87     * \param columns total number of columns
88     * \param rows total number of rows
89     * \return
90     */

```

```

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

```

```

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 "FFT.h"
11
12 namespace SoilMath {
13     FFT::FFT() {}
14     FFT::~~FFT() {}
15     ComplexVect_t FFT::GetDescriptors(const cv::Mat &img) {
16         if (!fftDescriptors.empty()) {
17             return fftDescriptors;
18         }
19
20         complexcontour = Contour2Complex(img, img.cols / 2, img.
21             rows / 2);
22
23         // Supplement the vector of complex numbers so that N = 2^
24             m
25         uint32_t N = complexcontour.size();
26         double logN = log(static_cast<double>(N)) / log(2.0);
27         if (floor(logN) != logN) {
28             // Get the next power of 2

```

```

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.
    size());
38 fft(ca);
39 fftDescriptors.assign(std::begin(ca), std::end(ca));
40 return fftDescriptors;
41 }
42
43 iContour_t FFT::Neighbors(uchar *0, int pixel, uint32_t
    columns,
44                          uint32_t rows) {
45     long int LUT_nBore[8] = {-columns + 1, -columns, -columns
        - 1, -1,
46                             columns - 1, columns, 1 +
                                columns, 1};
47     iContour_t neighbors;
48     uint32_t pEnd = rows * columns;
49     uint32_t count = 0;
50     for (uint32_t i = 0; i < 8; i++) {
51         count = pixel + LUT_nBore[i];
52         while (count >= pEnd && i < 8) {
53             count = pixel + LUT_nBore[++i];
54         }
55         if (i >= 8) {
56             break;
57         }
58         if (0[count] == 1)
59             neighbors.push_back(count);
60     }
61     return neighbors;
62 }
63
64 ComplexVect_t FFT::Contour2Complex(const cv::Mat &img, float
    centerCol,
65                                    float centerRow) {
66     uchar *0 = img.data;
67     uint32_t pEnd = img.cols * img.rows;
68
69     std::deque<std::deque<uint32_t>> sCont;
70     std::deque<uint32_t> eList;
71
72     // Initialize the queue
73     for (uint32_t i = 0; i < pEnd; i++) {
74         if (0[i] == 1) {
75             std::deque<uint32_t> tmpQ;
76             tmpQ.push_back(i);
77             sCont.push_back(tmpQ);

```

```

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                 ==
114                 eList.end()) // Check if this current route is
115                             extended elsewhere
116             {
117                 std::deque<uint32_t> nQ = cQ;
118                 nQ.push_back(nBors[j]); // Add the neighbor to the
119                 queue
120                 sCont.push_front(nQ); // add the sequence to the
121                 front of the queue
122             }
123         }
124     }
125     if (nBors.size() > 2) {
126         eList.push_back(i);
127     } // if there are multiple choices put current node in
128     extension List
129     if (i != pEnd) {
130         i = sCont.front().back();
131     } // If it isn't the end set i to the last node of the
132     first queue
133     if (sCont.size() == 0) {

```

```

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

```


174 }
175 }

C.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     public:
39         /*!
40         * \brief NN constructor for the Neural Net
41         * \param inputneurons number of input neurons
42         * \param hiddenneurons number of hidden neurons
43         * \param outputneurons number of output neurons
44         */
45         NN(uint32_t inputneurons, uint32_t hiddenneurons, uint32_t
46             outputneurons);
47
48         /*!
49         * \brief NN constructor for the Neural Net
50         */
51         NN();
52     };
53 }

```

```

51     * \brief ~NN virtual destructor 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 describing the contour in the
60     *           frequency domein serve as
61     *           input. The absolute value of these im. number because I
62     *           'm not interested
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     /*!
99     * \brief SetBeta a function to set the beta value
100    * \param value a floating value usually between 0.5 and
101      1.5
102    */
103 void SetBeta(float value) { beta = value; }
104 float GetBeta() { return beta; }
105
106     /*!
107     * \brief Learn the learning function
108     * \param input a vector of vectors with complex input
109       values
110     * \param cat a vector of vectors with the know output
111       values
112     * \param noOfDescriptorsUsed the total number of
113       descriptors which should be
114     * used
115     */
116 void Learn(InputLearnVector_t input, OutputLearnVector_t
117     cat,
118     uint32_t noOfDescriptorsUsed);
119
120     /*!
121     * \brief SaveState Serialize and save the values of the
122       Neural Net to disk
123     * \details Save the Neural Net in XML valued text file to
124       disk so that a
125     * object can
126     * be reconstructed on a latter stadia.
127     * \param filename a string indicating the file location
128       and name
129     */
130 void SaveState(std::string filename);
131
132     /*!
133     * \brief LoadState Loads the previous saved Neural Net
134       from disk
135     * \param filename a string indicating the file location
136       and name
137     */
138 void LoadState(std::string filename);
139
140 Weight_t iWeights; /**< a vector of real valued floating
141     point input weights*/
142 Weight_t hWeights; /**< a vector of real valued floating
143     point hidden weight*/
144
145 uint32_t MaxGenUsedByGA = 200;
146 uint32_t PopulationSizeUsedByGA = 30;
147 float MutationrateUsedByGA = 0.075f;
148 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 signals:
154     void learnErrorUpdate(double newError);
155
156 private:
157     GA *optim = nullptr;
158     std::vector<float> iNeurons; /**< a vector of input values
159                                   , the bias is
160                                   included, the bias is
161                                   included and
162                                   is the first value*/
163     std::vector<float> hNeurons; /**< a vector of hidden values, the bias is
164                                   included and
165                                   is the first value*/
166     std::vector<float> oNeurons; /**< a vector of output
167                                   values*/
168     uint32_t hiddenNeurons = 50; /**< number of hidden neurons
169                                   minus bias*/
170     uint32_t inputNeurons = 20; /**< number of input neurons
171                                   minus bias*/
172     uint32_t outputNeurons = 18; /**< number of output neurons
173                                   */
174     float beta; /**< the beta value, this indicates the
175                   steepness of the sigmoid
176                   function*/
177
178     friend class boost::serialization::access; /**< a private
179                                                   friend class so the
180                                                   serialization
181                                                   can
182                                                   access
183                                                   all
184                                                   the needed
185                                                   functions
186                                                   */
187
188     /*!
189     * \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
   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 "NN.h"
9  using namespace std;
10
11  namespace SoilMath {
12  NN::NN() { beta = 0.666; }
13
14  NN::NN(uint32_t inputneurons, uint32_t hiddenneurons,
   uint32_t outputneurons) {
15      // Set the number of neurons in the network
16      inputNeurons = inputneurons;
17      hiddenNeurons = hiddenneurons;
18      outputNeurons = outputneurons;
19      // Reserve the vector space
20      iNeurons.reserve(inputNeurons + 1); // input neurons +
   bias
21      hNeurons.reserve(hiddenNeurons + 1); // hidden neurons +
   bias
22      oNeurons.reserve(outputNeurons); // output neurons
23
24      beta = 0.666;
25  }

```



```

26
27 NN::~~NN()
28 {
29     if (optim != nullptr) {
30         delete optim;
31     }
32 }
33
34 void NN::LoadState(string filename) {
35     std::ifstream ifs(filename.c_str());
36     boost::archive::xml_iarchive ia(ifs);
37     ia >> boost::serialization::make_nvp("NeuralNet", *this);
38 }
39
40 void NN::SaveState(string filename) {
41     std::ofstream ofs(filename.c_str());
42     boost::archive::xml_oarchive oa(ofs);
43     oa << boost::serialization::make_nvp("NeuralNet", *this);
44 }
45
46 Predict_t NN::PredictLearn(ComplexVect_t input, Weight_t
    inputweights,
47                             Weight_t hiddenweights, uint32_t
    inputneurons,
48                             uint32_t hiddenneurons, uint32_t
    outputneurons) {
49     NN neural(inputneurons, hiddenneurons, outputneurons);
50     neural.studied = true;
51     neural.SetInputWeights(inputweights);
52     neural.SetHiddenWeights(hiddenweights);
53     return neural.Predict(input);
54 }
55
56 Predict_t NN::Predict(ComplexVect_t input) {
57     if (input.size() != inputNeurons) {
58         throw Exception::MathException(
59             EXCEPTION_SIZE_OF_INPUT_NEURONS,
60             EXCEPTION_SIZE_OF_INPUT_NEURONS_NR
61         );
62     }
63     if (!studied) {
64         throw Exception::MathException(
65             EXCEPTION_NEURAL_NET_NOT_STUDIED,
66             EXCEPTION_NEURAL_NET_NOT_STUDIED_NR
67         );
68     }
69
70     iNeurons.clear();
71     hNeurons.clear();
72     oNeurons.clear();
73
74     // Set the bias in the input and hidden vector to 1 (real
75     number)
76     iNeurons.push_back(1.0f);
77     hNeurons.push_back(1.0f);
78
79 }

```

```

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 for (uint32_t i = 0; i < oNeurons.size(); i++) {
98     wCount = i;
99     for (uint32_t j = 0; j < hNeurons.size(); j++) {
100         oNeurons[i] += hNeurons[j] * hWeights[wCount];
101         wCount += oNeurons.size();
102     }
103     oNeurons[i] =
104         (2 / (1.0f + pow(2.71828f, (-oNeurons[i] * beta))))
105         - 1; // Shift plus scale so the learning function can
106             // be calculated
107 }
108 retVal.OutputNeurons = oNeurons;
109 retVal.ManualSet = false;
110 return retVal;
111 }
112
113 void NN::Learn(InputLearnVector_t input, OutputLearnVector_t
114     cat,
115     uint32_t noOfDescriptorsUsed __attribute__((
116         unused))) {
117     if (optim == nullptr) {
118         optim = new SoilMath::GA(PredictLearn, inputNeurons,
119             hiddenNeurons, outputNeurons);
120     }
121     connect(optim, SIGNAL(learnErrorUpdate(double)), this,
122         SIGNAL(learnErrorUpdate(double)));
123
124     optim->Elitisme = ElitismeUsedByGA;
125     optim->EndError = EndErrorUsedByGA;
126     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 }
170 }
171 }

```

C.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 (
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 MAX_UINT8_VALUE 256
12 #define VECTOR_CALC 1
13
14 #include <stdint.h>
15 #include <utility>
16 #include <vector>
17 #include <cstdlib>
18 #include <cmath>
19 #include <limits>
20 #include <typeinfo>
21 #include <string>
22
23 #include <fstream>
24
25 #include <boost/archive/binary_iarchive.hpp>
26 #include <boost/archive/binary_oarchive.hpp>
27 #include <boost/serialization/version.hpp>
28 #include <boost/math/distributions/students_t.hpp>
29
30 #include "MathException.h"
31 #include "SoilMathTypes.h"
32 #include "CommonOperations.h"
33
34 namespace SoilMath {
35
36     /*!
37     * \brief Stats class
38     * \details Usage Stats<type1, type2, type3>Stats() type 1,
39     * 2 and 3 should be of
40     * the same value and consecutive in size
41     */
42     template <typename T1, typename T2, typename T3> class Stats
43     {
44     public:
45         bool isDiscrete = true; /**< indicates if the data is
46             discrete or real*/
47
48         T1 *Data = nullptr; /**< Pointer the data*/
49         uint32_t *bins = nullptr; /**< the histogram*/
50         double *CFD = nullptr; /**< the CFD*/
51         bool Calculated = false; /**< indication if the data has
52             been calculated*/
53         float Mean = 0.0; /**< the mean value of the data
54             */
55     };
56 
```

```

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
    values*/
58     uint16_t Rows = 0;        /**< number of rows from the
    data matrix*/
59     uint16_t Cols = 0;        /**< number of cols from the
    data matrix*/
60     bool StartAtZero = true;  /**< indication of the minimum
    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
    the first bin*/
66     uint32_t *end() { return &bins[noBins]; } /**< pointer to
    the last + 1 bin*/
67
68     /*!
69     * \brief WelchTest Compare the sample using the Welch's
    Test
70     * \details (source:
71     * 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)
72     * \param statComp Statiscs Results of which it should be
    tested against
73     * \return
74     */
75     bool WelchTest(SoilMath::Stats<T1, T2, T3> &statComp) {
76         double alpha = 0.05;
77         // Degrees of freedom:
78         double v = statComp.Std * statComp.Std / statComp.n +
79             this->Std * this->Std / this->n;
80         v *= v;
81         double t1 = statComp.Std * statComp.Std / statComp.n;
82         t1 *= t1;
83         t1 /= (statComp.n - 1);
84         double t2 = this->Std * this->Std / this->n;
85         t2 *= t2;
86         t2 /= (this->n - 1);
87         v /= (t1 + t2);
88         // t-statistic:
89         double t_stat = (statComp.Mean - this->Mean) /
90             sqrt(statComp.Std * statComp.Std /
91                 statComp.n +
92                 this->Std * this->Std / this->n);
93         //

```

```

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 /*!
110 * \brief Stats Constructor
111 * \param rhs Right hand side
112 */
113 Stats(const Stats &rhs)
114     : bins{new uint32_t[rhs.noBins]{0}}, CFD{new double[
115         rhs.noBins]{}},
116       BinRanges{new double[rhs.noBins]{} } {
117     this->binRange = rhs.binRange;
118     this->Calculated = rhs.Calculated;
119     this->Cols = rhs.Cols;
120     this->EndBin = rhs.EndBin;
121     this->isDiscrete = rhs.isDiscrete;
122     this->max = rhs.max;
123     this->Mean = rhs.Mean;
124     this->min = rhs.min;
125     this->n = rhs.n;
126     this->noBins = rhs.noBins;
127     this->n_end = rhs.n_end;
128     this->Range = rhs.Range;
129     this->Rows = rhs.Rows;
130     this->Startbin = rhs.Startbin;
131     this->Std = rhs.Std;
132     this->Sum = rhs.Sum;
133     std::copy(rhs.bins, rhs.bins + rhs.noBins, this->bins);
134     std::copy(rhs.CFD, rhs.CFD + rhs.noBins, this->CFD);
135     std::copy(rhs.BinRanges, rhs.BinRanges + rhs.noBins,
136         this->BinRanges);
137     this->Data = rhs.Data;
138     this->StartAtZero = rhs.StartAtZero;
139     this->HighestPDF = rhs.HighestPDF;
140 }
141
142 /*!
143 * \brief operator = Assignmet operator
144 * \param rhs right hand side
145 * \return returns the right hand side
146 */
147 Stats &operator=(Stats const &rhs) {
148     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 return *this;
191 }
192
193 /*!
194  * \brief Stats Constructor
195  * \param noBins number of bins with which to build the
196  *         histogram
197  * \param startBin starting value of the first bin
198  * \param endBin end value of the second bin
199  */
200 Stats(int noBins = 256, T1 startBin = 0, T1 endBin = 255)
201 {
202     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 /*!
221  * \brief Stats constructor
222  * \param data Pointer to the data
223  * \param rows Number of rows
224  * \param cols Number of Columns
225  * \param noBins Number of bins
226  * \param startBin Value of the start bin
227  * \param startatzero bool indicating if the bins should
228  *       be shifted from zero
229 */
230 Stats(T1 *data, uint16_t rows, uint16_t cols, int noBins =
231     256,
232     T1 startBin = 0, bool startatzero = true) {
233     min = std::numeric_limits<T1>::max();
234     max = std::numeric_limits<T1>::min();
235     Range = max - min;
236
237     Startbin = startBin;
238     EndBin = startBin + noBins;
239     StartAtZero = startatzero;
240
241     if (typeid(T1) == typeid(float) || typeid(T1) == typeid(
242         double) ||
243         typeid(T1) == typeid(long double)) {
244         isDiscrete = false;
245     } else {
246         isDiscrete = true;
247     }
248
249     Data = data;
250     Rows = rows;
251     Cols = cols;
252     bins = new uint32_t[noBins]{0};
253     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     } else {
321         isDiscrete = true;
322     }
323
324     bins = new uint32_t[noBins]{0};
325     CFD = new double[noBins]{};
326     BinRanges = new double[noBins]{};
327     while (i-- > 0) {
328         bins[i] = binData[i];
329         n += binData[i];
330     }
331     BinCalculations(startC, endC);
332 }
333
334 ~Stats() {
335     Data == nullptr;
336     if (bins != nullptr) {
337         delete[] bins;
338         bins = nullptr;
339     }
340     if (CFD != nullptr) {
341         delete[] CFD;
342         CFD = nullptr;
343     }
344     if (BinRanges != nullptr) {
345         delete[] BinRanges;
346         BinRanges = nullptr;
347     }
348 }
349
350 /*!
351  * \brief BasicCalculateFloat execute the basic float data
352         calculations
353  */
354 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 /*!
393  * \brief BasicCalculateFloat execute the basic float data
394  *        calculations with a
395  * mask
396  * \param mask uchar mask type 0 don't calculate, 1
397  *        calculate
398  */
399 void BasicCalculateFloat(uchar *mask) {
400     float sum_dev = 0.0;
401     n = Rows * Cols;
402     uint32_t nmask = 0;
403     for (uint32_t i = 0; i < n; i++) {
404         if (mask[i] != 0) {
405             if (Data[i] > max) {
406                 max = Data[i];

```

```

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) /
        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
         calculations with
493  * mask
494  * \param mask uchar mask type 0 don't calculate, 1
         calculate
495  */
496 void BasicCalculate(uchar *mask) {
497     double sum_dev = 0.0;
498     n = Rows * Cols;
499     uint32_t nmask = 0;
500     uint32_t i = 0;
501     std::for_each(Data, Data + n, [&](T1 &d) {
502         if (mask[i++] != 0) {
503             if (d > max) {
504                 max = d;
505             }
506             if (d < min) {
507                 min = d;
508             }
509             Sum += d;
510             nmask++;

```

```

511     }
512   });
513   binRange = static_cast<T1>(ceil((max - min) /
514     static_cast<float>(noBins)));
515   Mean = Sum / (float)nmask;
516   Range = max - min;
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 excute the cacluations 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   }

```

```

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::
        vector<double> &yAxis,
602                     double Step, double start = 0, double
                        stop = 7) {
603     uint32_t resolution;
604     resolution = static_cast<uint32_t>(((stop - start) /
        Step) + 0.5);
605
606     xAxis.push_back(start);
607     double yVal0 = (1 / (Std * 2.506628274631)) *
608         exp(-(pow((start - Mean), 2) / (2 * pow(
            Std, 2))));
609     yAxis.push_back(yVal0);
610     HighestPDF = yVal0;
611     for (uint32_t i = 1; i < resolution; i++) {
612         double xVal = xAxis[xAxis.size() - 1] + Step;
613         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
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 "Stats.h"
13 #include <boost/serialization/base_object.hpp>
14
15 namespace SoilMath {
16 class PSD : public SoilMath::Stats<double, double, long
17     double> {
18 private:
19     uint32_t DetBin(float value) {
20         uint32_t i = noBins - 1;
21         while (i > 0) {
22             if (value > BinRanges[i]) {
23                 return i;
24             }
25             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
74         BasicCalculatePSD();
75     }
76 };
77 }

```

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

C.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 -
50         lboost_iostreams
51
52 #Zlib
53 LIBS += -L/usr/local/lib -lz
54 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
   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  /*! \brief Collection of the public SoilMath headers
9  * Commonpractice is to include this header when you want to
   add Soilmath
10 * routines
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
   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  #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)
    ? a : b; }
26
27 inline uint16_t Max(uint16_t a, uint16_t b, uint16_t c,
    uint16_t d) {
28     return (Max(a, b) > Max(c, d)) ? Max(a, b) : Max(c, d);
29 }
30
31 inline uint16_t Min(uint16_t a, uint16_t b) { return (a < b)
    ? a : b; }
32
33 inline uint16_t Min(uint16_t a, uint16_t b, uint16_t c,
    uint16_t d) {
34     return (Min(a, b) > Min(c, d)) ? Min(a, b) : Min(c, d);
35 }
36
37 static inline double quick_pow10(int n) {
38     static double pow10[19] = {1, 10, 100, 1000, 10000,
        100000, 1000000, 10000000,
39                                     100000000, 1000000000,
        10000000000, 100000000000,
40                                     1000000000000, 10000000000000,
        100000000000000,
41                                     1000000000000000,
        10000000000000000,
42                                     100000000000000000,
        1000000000000000000};
43     return pow10[(n >= 0) ? n : -n];
44 }
45
46
47 // Source:
48 // http://martin.ankerl.com/2012/01/25/optimized-approximative-pow-in-c-and-cpp/
49 static inline double fastPow(double a, double b) {
50     union {
51         double d;
52         int x[2];
53     } u = {a};
54     u.x[1] = (int)(b * (u.x[1] - 1072632447) + 1072632447);
55     u.x[0] = 0;
56     return u.d;
57 }
58
59 static inline double quick_pow2(int n) {
60     static double pow2[256] = {
61         0,      1,      4,      9,      16,      25,      36,      49,
        64,      81,
62         100,    121,    144,    169,    196,    225,    256,    289,
        324,    361,
63         400,    441,    484,    529,    576,    625,    676,    729,
        784,    841,
64         900,    961,    1024,    1089,    1156,    1225,    1296,    1369,

```

```

        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 /*!
112  * \brief calcDiameter according to ISO 9276-6
113  * \param A
114  * \return
115  */
116 static inline float calcDiameter(float A) {
117     //return sqrt((4 * A) / M_PI);
118     return 1.1283791670955 * sqrt(A);
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 <spijker.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
46         precise to outcome is*/
47     float Accuracy = 1.; /**< the accuracy of the category*/
48     std::vector<float> OutputNeurons; /**< the output Neurons
    */
49     bool ManualSet = true;
50 } Predict_t;              /**< The prediction
    results*/
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
    unsigned int version __attribute__((unused))) {
25     int cols = m.cols;
26     int rows = m.rows;
27     int elemSize = m.elemSize();
28     int elemType = m.type();
29
30     ar &cols;
31     ar &rows;
32     ar &elemSize;
33     ar &elemType; // element type.
34
35     if (m.type() != elemType || m.rows != rows || m.cols !=
        cols) {
36         m = cv::Mat(rows, cols, elemType, cv::Scalar(0));
37     }
38
39     size_t dataSize = cols * rows * elemSize;
40
41     for (size_t dc = 0; dc < dataSize; dc++) {
42         ar &m.data[dc];
43     }
44 }
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
10 // Source:
11 // http://stackoverflow.com/questions/16125574/how-to-
12 //     serialize-opencv-mat-with-boost-xml-archive
13 #pragma once
14
15 #include <boost/archive/binary_iarchive.hpp>
16 #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
    unsigned int version __attribute__((unused))) {
26     ar &P.Accuracy;
27     ar &P.Category;
28     ar &P.OutputNeurons;
29     ar &P.RealValue;
30 }
31 }
32 }

```

```

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 #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:

```

```

30     MathException(std::string m = EXCEPTION_MATH, int n =
        EXCEPTION_MATH_NR)
31         : msg(m), nr(n){};
32     ~MathException() _GLIBCXX_USE_NOEXCEPT{};
33     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
        msg.c_str(); };
34     const int *id() const _GLIBCXX_USE_NOEXCEPT { return &nr;
        }
35
36 private:
37     std::string msg;
38     int nr;
39 };
40 }
41 }

```

```

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 #include <stdint.h>
10
11 namespace SoilMath {
12     /*!
13     * \brief The Sort template class
14     */
15     class Sort {
16     public:
17         Sort() {}
18         ~Sort() {}
19
20         /*!
21         * \brief QuickSort a static sort a Type T array with i
                values
22         * \details Usage: QuickSort<type>(*type , i)
23         * \param arr an array of Type T
24         * \param i the number of elements
25         */
26         template <typename T> static void QuickSort(T *arr, int i)
                {
27             if (i < 2)
28                 return;
29
30             T p = arr[i / 2];
31             T *l = arr;
32             T *r = arr + i - 1;
33             while (l <= r) {
34                 if (*l < p) {
35                     l++;
36                 } else if (*r > p) {

```

```

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             l++;
90             r--;

```

```
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 }
```

D. Hardware Library

D.0.1 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>
```

```

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 (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
10 #include "Microscope.h"
11
12 namespace Hardware {
13
14     Microscope::Microscope() {
15         RunEnv = GetCurrentArchitecture();
16         AvailableCams = GetAvailableCams();
17         for_each(AvailableCams.begin(), AvailableCams.end(), [](
18             Cam_t &C) {
19             C.SelectedResolution = &C.Resolutions[C.Resolutions.size
20                 () - 1];
21         });
22     }
23
24     Microscope::Microscope(const Microscope &rhs) {
25         std::copy(rhs.AvailableCams.begin(), rhs.AvailableCams.end
26             (),
27             this->AvailableCams.begin());
28         this->RunEnv = rhs.RunEnv;
29         this->SelectedCam = rhs.SelectedCam;
30         this->cap = rhs.cap;
31         this->fd = rhs.fd;
32         this->HDRframes = rhs.HDRframes;
33     }
34
35     Microscope::~Microscope() { delete cap; }
36
37     Microscope::Arch Microscope::GetCurrentArchitecture() {
38         struct utsname unameData;
39         Arch retVal;
40         uname(&unameData);
41         std::string archString = static_cast<std::string>(
42             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
81             // Get controls
82             memset(&queryctrl, 0, sizeof(queryctrl));
83             memset(&controlctrl, 0, sizeof(controlctrl));
84             for (queryctrl.id = V4L2_CID_BASE; queryctrl.id <
85                 V4L2_CID_LASTP1;
86                 queryctrl.id++) {
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.
            default_value;
88         currentControl.step = queryctrl.step;
89         controlctrl.id = queryctrl.id;
90         if (ioctl(currentCam.fd, VIDIOC_G_CTRL, &
            controlctrl) == 0) {
91             currentControl.current_value = controlctrl.
                value;
92         }
93         currentCam.Controls.push_back(currentControl);
94     }
95     } else {
96         if (errno == EINVAL)
97             continue;
98         throw Exception::MicroscopeException(
99             EXCEPTION_QUERY,
100                                     EXCEPTION_QUERY_NR
101                                     );
102     }
103 }
104
105 // Get image formats
106 struct v4l2_format format;
107 memset(&format, 0, sizeof(format));
108
109 uint32_t width[5] = {640, 800, 1280, 1600, 2048};
110 uint32_t height[6] = {480, 600, 960, 1200, 1536};
111
112 uint32_t ResolutionID = 0;
113
114 // YUYV
115 for (uint32_t i = 0; i < 5; i++) {
116     format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
117     format.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;
118     format.fmt.pix.width = width[i];
119     format.fmt.pix.height = height[i];
120     int ret = ioctl(currentCam.fd, VIDIOC_S_FMT, &format);
121     if (ret != -1 && format.fmt.pix.height == height[i]
122         &&
123         format.fmt.pix.width == width[i]) {
124         Resolution_t res;
125         res.Width = format.fmt.pix.width;
126         res.Height = height[i];
127         res.ID = ResolutionID++;
128         res.format = PixelFormat::YUYV;
129         currentCam.Resolutions.push_back(res);
130     }
131 }
132
133 // MJPEG
134 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::MJPG;
146             currentCam.Resolutions.push_back(res);
147         }
148     }
149     close(currentCam.fd);
150     retVal.push_back(currentCam);
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.
        begin();
185         it != SelectedCam->Controls.end(); ++it) {
186         SetControl(&*it);
187     }
188     return true;
189 }
190 }
191 return false;
192 }
193
194 bool Microscope::openCam(std::string &cam) { return openCam(
    FindCam(cam)); }
195
196 bool Microscope::openCam(int &cam) { return openCam(FindCam(
    cam)); }
197
198 Microscope::Cam_t *Microscope::FindCam(int cam) {
199     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
200         if (cam == AvailableCams[i].ID) {
201             return &AvailableCams[i];
202         }
203     }
204     return nullptr;
205 }
206
207 Microscope::Cam_t *Microscope::FindCam(string cam) {
208     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
209         if (cam.compare(AvailableCams[i].Name) == 0) {
210             return &AvailableCams[i];
211         }
212     }
213     return nullptr;
214 }
215
216 bool Microscope::closeCam(Cam_t *cam) {
217     if (cap != nullptr) {
218         if (cap->isOpened()) {
219             cap->release();
220         }
221         delete cap;
222         cap = nullptr;
223     }
224 }
225
226 void Microscope::GetFrame(cv::Mat &dst) {
227     openCam(SelectedCam);
228     sleep(SelectedCam->delaytrigger);
229     if (RunEnv == Arch::ARM) {
230         for (uint32_t i = 0; i < 2; i++) {
231             if (!cap->grab()) {
232                 throw Exception::CouldNotGrabImageException();
233             }
234             sleep(SelectedCam->delaytrigger);
235         }

```

```

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     return nullptr;
291 }

```



```

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     } else if (queryctrl.flags & V4L2_CTRL_FLAG_DISABLED) {
312         close(SelectedCam->fd);
313         throw Exception::MicroscopeException(
314             EXCEPTION_CTRL_NOT_FOUND,
315             EXCEPTION_CTRL_NOT_FOUND_NR);
316     } else {
317         memset(&controlctrl, 0, sizeof(controlctrl));
318         controlctrl.id = control->ID;
319         controlctrl.value = control->current_value;
320
321         if (ioctl(SelectedCam->fd, VIDIOC_S_CTRL, &controlctrl)
322             == -1) {
323             // Fails on auto white balance
324             // throw Exception::MicroscopeException(
325                 EXCEPTION_CTRL_VALUE,
326                 EXCEPTION_CTRL_VALUE_NR);
327         }
328     }
329     close(SelectedCam->fd);
330 }
331 }
332 }
333 }

```

D.0.2 Beaglebone Black Class

```

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  /*! \class BBB
9  The core BeagleBone Black class used for all hardware
   related classes.
10 Consisting of universal used method, functions and variables
   . File operations,
11 polling and threading
12 */
13
14 #pragma once
15
16 #define SLOTS
   \
17  "/sys/devices/platform/bone_capemgr/slots" /*!< Beaglebone
   capemanager slots file*/
18
19 #include <fstream>
20 #include <sstream>
21 #include <string>
22 #include <sys/stat.h>
23 #include <pthread.h>
24 #include <unistd.h>
25 #include <sys/epoll.h>
26 #include <fcntl.h>
27 #include <regex>
28 #include <stdexcept>
29
30 #include "GPIOReadException.h"
31 #include "FailedToCreateGPIOPollingThreadException.h"
32 #include "ValueOutOfBoundsException.h"
33
34 using namespace std;
35
36 namespace Hardware {
37 typedef int (*CallbackType)(
38     int); /*!< CallbackType used to pass a function to a
   thread*/
39
40 class BBB {
41 public:
42     int debounceTime; /*!< debounce time for a button in
   milliseconds*/
43
44     BBB();
45     ~BBB();

```

```

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

```

```

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 "BBB.h"
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
39 /*! Writes a value to a file
40 \param path a constant string pointing towards the file
41 \param value a constant string which should be written in
42     the file
43 */
44 void BBB::Write(const string &path, const string &value) {
45     ofstream fs;
46     fs.open(path.c_str());
47     if (!fs.is_open()) {
48         throw Exception::GPIOReadException(("Can't open: " +
49             path).c_str());
50     }
51     fs << value;
52     fs.close();
53 }
54
55 /*! Checks if a directory exist
56 \returns true if the directory exists and false if not
57 */
58 bool BBB::DirectoryExist(const string &path) {
59     struct stat st;
60     if (stat((char *)path.c_str(), &st) != 0) {
61         return false;
62     }
63     return true;
64 }
65
66 /*! Checks if a cape is loaded in the file /sys/devices/
67     bone_capemgr.9/slots
68 \param shield a const search string which is a (part) of the
69     shield name
70 \return true if the search string is found otherwise false
71 */
72 bool BBB::CapeLoaded(const string &shield) {
73     bool shieldFound = false;
74     ifstream fs;

```

```
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 }
```

D.0.3 GPIO Class

```

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  * This code is based upon:
7  * Derek Molloy, "Exploring BeagleBone: Tools and Techniques
   for Building
8  * with Embedded Linux", Wiley, 2014, ISBN:9781118935125.
9  * See: www.exploringbeaglebone.com
10 */
11
12 #pragma once
13 #include "BBB.h"
14
15 #define EXPORT_PIN "/sys/class/gpio/export"
16 #define UNEXPORT_PIN "/sys/class/gpio/unexport"
17 #define GPIOS "/sys/class/gpio/gpio"
18 #define DIRECTION "/direction"
19 #define VALUE "/value"
20 #define EDGE "/edge"
21
22 using namespace std;
23
24 namespace Hardware {
25 class GPIO : public BBB {
26 public:
27     enum Direction { Input, Output };
28     enum Value { Low = 0, High = 1 };
29     enum Edge { None, Rising, Falling, Both };
30
31     int number; // Number of the pin
32
33     int WaitForEdge();
34     int WaitForEdge(CallbackType callback);
35     void WaitForEdgeCancel() { this->threadRunning = false; }
36
37     Value GetValue();
38     void SetValue(Value value);
39
40     Direction GetDirection();
41     void SetDirection(Direction direction);
42
43     Edge GetEdge();
44     void SetEdge(Edge edge);
45
46     GPIO(int number);
47     ~GPIO();
48
49 private:
50     string gpiorpath;
51     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
        direction);
61
62     Edge ReadsEdge(const string &gpiopath);
63     void WritesEdge(const string &gpiopath, Edge edge);
64
65     Value ReadsValue(const string &gpiopath);
66     void WritesValue(const string &gpiopath, Value value);
67 };
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     this->number = number;
15     gpiopath = GPIOS + NumberToString<int>(number);
16
17     if (!isExported(number, direction, edge)) {
18         ExportPin(number);
19         direction = ReadsDirection(gpiopath);
20         edge = ReadsEdge(gpiopath);
21     }
22     usleep(250000);
23 }
24
25 GPIO::~~GPIO() { UnexportPin(number); }
26
27 int GPIO::WaitForEdge(CallbackType callback) {
28     threadRunning = true;
29     callbackFunction = callback;
30     if (pthread_create(&this->thread, NULL, &threadedPollGPIO,
31         static_cast<void *>(this))) {
32         threadRunning = false;
33         throw Exception::
            FailedToCreateGIOPollingThreadException();

```

```

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             FailedToCreateGIOPollingThreadException(
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     // read operation | edge triggered | urgent data
54     ev.events = EPOLLIN | EPOLLET | EPOLLPRI;
55     ev.data.fd = fd;
56     if (epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev) == -1) {
57         throw Exception::
58             FailedToCreateGIOPollingThreadException(
59                 "GPIO: Failed to add control interface!");
60     }
61     while (count <= 1) {
62         i = epoll_wait(epollfd, &ev, 1, -1);
63         if (i == -1) {
64             close(fd);
65             return -1;
66         } else {
67             count++;
68         }
69     }
70     close(fd);
71     return 0;
72 }
73
74 GPIO::Value GPIO::GetValue() { return ReadsValue(gpiopath);
75     }
76
77 void GPIO::SetValue(GPIO::Value value) { WritesValue(
78     gpiopath, value); }
79
80 GPIO::Direction GPIO::GetDirection() { return direction; }
81 void GPIO::SetDirection(Direction direction) {
82     this->direction = direction;
83     WritesDirection(gpiopath, direction);
84 }
85
86 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     return 0;
274 }
```

D.0.4 PWM Class

```

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  #include "BBB.h"
10 #include <dirent.h>
11
12 #define OCP_PATH "/sys/class/pwm/"
13 #define PWM_CAPE "Override Board Name,00A0,Override Manuf,
   cape-universaln"
14
15 namespace Hardware {
16 class PWM : public BBB {
17 public:
18     enum Pin // Four possible PWM pins
19     { P8_13,
20       P8_19,
21       P9_14,
22       P9_16 };
23     enum Run // Signal generating
24     { On = 1,
25       Off = 0 };
26     enum Polarity // Inverse duty polarity
27     { Normal = 1,
28       Inverted = 0 };
29
30     Pin pin; // Current pin
31
32     uint8_t GetPixelValue() { return pixelvalue; }
33     void SetPixelValue(uint8_t value);
34
35     float GetIntensity() { return intensity; };
36     void SetIntensity(float value);
37
38     int GetPeriod() { return period; };
39     void SetPeriod(int value);
40
41     int GetDuty() { return duty; };
42     void SetDuty(int value);
43     void SetIntensity();
44
45     Run GetRun() { return run; };
46     void SetRun(Run value);
47
48     Polarity GetPolarity() { return polarity; };
49     void SetPolarity(Polarity value);
50
51     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 polaity
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
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 "PWM.h"
11
12 namespace Hardware {
13     /// <summary>
14     /// Constructeur
15     /// </summary>
16     /// <param name="pin">Pin</param>
17     PWM::PWM(Pin pin) {
18         this->pin = pin;
19
20         // Check if PWM cape is loaded, if not load it
21         if (!CapeLoaded(PWM_CAPE)) {
22             Write(SLOTS, PWM_CAPE);
23         }
24
25         // Init the pin
26         switch (pin) {
27             case Hardware::PWM::P8_13:
28                 system("config-pin P8.13 pwm");
29                 basepath = OCP_PATH;
30                 basepath.append("pwmchip4/pwm1");
31                 break;
32             case Hardware::PWM::P8_19:
33                 system("config-pin P8.19 pwm");
34                 basepath = OCP_PATH;
35                 basepath.append("pwmchip4/pwm0");
36                 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 }
87
88 /// <summary>
89 /// Set the intensity level as percentage
90 /// </summary>

```

```

89  /// <param name="value">floating value multiplication factor
    </param>
90  void PWM::SetIntensity(float value) {
91      if (polarity == Normal) {
92          SetDuty(static_cast<int>((value * duty) + 0.5));
93      } else {
94          SetPeriod(static_cast<int>((value * period) + 0.5));
95      }
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 }
```

D.0.5 ADC Class

```

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  /*! \class ADC
9  Interaction with the beaglebone analogue pins
10 */
11
12 #pragma once
13
14 #include "BBB.h"
15 #include "ADCReadException.h"
16
17 #define ADC0_PATH
18     \
19     "/sys/bus/iio/devices/iio:device0/in_voltage0_raw" /*!<
20     path to analogue pin \
21                                     0*/
22 #define ADC1_PATH
23     \
24     "/sys/bus/iio/devices/iio:device0/in_voltage1_raw" /*!<
25     path to analogue pin \
26                                     1*/
27 #define ADC2_PATH
28     \
29     "/sys/bus/iio/devices/iio:device0/in_voltage2_raw" /*!<
30     path to analogue pin \
31                                     2*/
32 #define ADC3_PATH
33     \
34     "/sys/bus/iio/devices/iio:device0/in_voltage3_raw" /*!<
35     path to analogue pin \
36                                     3*/
37 #define ADC4_PATH
38     \
39     "/sys/bus/iio/devices/iio:device0/in_voltage4_raw" /*!<
40     path to analogue pin \
41                                     4*/
42 #define ADC5_PATH
43     \
44     "/sys/bus/iio/devices/iio:device0/in_voltage5_raw" /*!<
45     path to analogue pin \
46                                     5*/

```

```

35 #define ADC6_PATH
    \
36  "/sys/bus/iio/devices/iio:device0/in_voltage6_raw" /*!<
    path to analogue pin \
37                                          6*/
38 #define ADC7_PATH
    \
39  "/sys/bus/iio/devices/iio:device0/in_voltage7_raw" /*!<
    path to analogue pin \
40                                          7*/
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(ADCPin 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 =
    false; }
73
74 private:
75  string adcpath; /*!< Path to analogue write file*/
76  float Intensity; /*!< Current intensity expressed as
    percentage*/
77  int MinIntensity; /*!< Voltage level which represent 0
    percentage*/
78  int MaxIntensity; /*!< Voltage level which represent 100
    percentage*/
79

```

```

80     friend void *threadedPollADC(void *value); /*!< friend
        polling function*/
81 };
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
        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 "ADC.h"
9
10 namespace Hardware {
11     /*! Constructor
12     \param pin and ADCPin type indicating which analogue pin to
        use
13     */
14     ADC::ADC(ADCPin pin) {
15         this->Pin = pin;
16         switch (pin) {
17             case Hardware::ADC::ADC0:
18                 adcpath = ADC0_PATH;
19                 break;
20             case Hardware::ADC::ADC1:
21                 adcpath = ADC1_PATH;
22                 break;
23             case Hardware::ADC::ADC2:
24                 adcpath = ADC2_PATH;
25                 break;
26             case Hardware::ADC::ADC3:
27                 adcpath = ADC3_PATH;
28                 break;
29             case Hardware::ADC::ADC4:
30                 adcpath = ADC4_PATH;
31                 break;
32             case Hardware::ADC::ADC5:
33                 adcpath = ADC5_PATH;
34                 break;
35             case Hardware::ADC::ADC6:
36                 adcpath = ADC6_PATH;
37                 break;
38             case Hardware::ADC::ADC7:
39                 adcpath = ADC7_PATH;
40                 break;
41         }
42
43         MinIntensity = 0;
44         MaxIntensity = 4096;
45     }
46

```

```

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(adcpath));
55      Intensity = (float)(retVal - MinIntensity) /
56                  (4096 - (MinIntensity + (4096 - MaxIntensity))
57                  );
57      return retVal;
58  }
59
60  /*! Set the current voltage at the pin as the minimum
61  voltage*/
62  void ADC::SetMinIntensity() {
63      MinIntensity = StringToNumber<int>(Read(adcpath));
64  }
65  void ADC::SetMaxIntensity() {
66      MaxIntensity = StringToNumber<int>(Read(adcpath));
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
87  /*! Polling of the analogue pin
88  \return the current value
89  */
90  int ADC::WaitForValueChange() {
91      int fd, i, epollfd, count = 0;
92      struct epoll_event ev;
93      epollfd = epoll_create(1);
94      if (epollfd == -1) {
95          throw Exception::
96              FailedToCreateGPIOPollingThreadException(
97                  "GPIO: Failed to create epollfd!");
98      }
99      if ((fd = open(adcpath.c_str(), O_RDONLY | O_NONBLOCK)) ==
100          -1) {

```

```
197     throw Exception::ADCReadException();
198 }
199 ev.events = EPOLLIN;
200 ev.data.fd = fd;
201
202 if (epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev) == -1) {
203     throw Exception::
204         FailedToCreateGPIOPollingThreadException(
205             "ADC: Failed to add control interface!");
206 }
207
208 while (count <= 1) {
209     i = epoll_wait(epollfd, &ev, 1, -1);
210     if (i == -1) {
211         close(fd);
212         return -1;
213     } else {
214         count++;
215     }
216 }
217 close(fd);
218 return StringToNumber<int>(Read(adcpath));
219 }
220
221 /*! friendly function to start the threading*/
222 void *threadedPollADC(void *value) {
223     ADC *adc = static_cast<ADC *>(value);
224     while (adc->threadRunning) {
225         adc->callbackFunction(adc->WaitForValueChange());
226         usleep(200000);
227     }
228 }
```

D.0.6 EC12P Class

```

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 <spiijker.jelle@gmail.com>, 2015
6  */
7
8  /*! \class EC12P
9  Interaction with the sparksfun RGB encoder
10 */
11
12 #pragma once
13
14 #include "eqep.h"
15 #include "GPIO.h"
16 #include "FailedToCreateThreadException.h"
17
18 #include <pthread.h>
19
20 using namespace std;
21
22 namespace Hardware {
23 class EC12P {
24 public:
25     EC12P();
26     ~EC12P();
27
28     /*! Enumerator indicating the color of the encoder shaft*/
29     enum Color {
30         Red,          /*!< Red*/
31         Pink,         /*!< Pink*/
32         Blue,         /*!< Blue*/
33         SkyBlue,     /*!< SkyBlue*/
34         Green,       /*!< Green*/
35         Yellow,      /*!< Yellow*/
36         White,       /*!< White*/
37         None         /*!< Off*/
38     };
39
40     void SetPixelColor(Color value);
41     Color GetPixelColor() { return PixelColor; };
42
43     void RainbowLoop(int sleeperperiod);
44     void StopRainbowLoop() { threadRunning = false; };
45
46     eQEP Rotary{eQEP2, eQEP::eQEP_Mode_Absolute}; /*!< The
   encoder*/
47     GPIO Button{68}; /*!< The
   pushbutton*/
48
49 private:
50     Color PixelColor; /*!< Current shaft color*/

```

```

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 sleeperperiod;   /*!< 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         threadRunning = false;
29     }
30
31     /*! De-constructor*/
32     EC12P::~EC12P() {}
33
34     /*! Set the shaft color
35     \param value as Color enumerator
36     */
37     void EC12P::SetPixelColor(Color value) {
38         switch (value) {
39             case Hardware::EC12P::Red:
40                 R.SetValue(GPIO::High);
41                 B.SetValue(GPIO::Low);
42                 G.SetValue(GPIO::Low);
43                 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 sleeperperiod) {
83      this->sleepperiod = sleeperperiod;
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 }
```

D.0.7 eQep Class

```

1  /*
2  * TI eQEP driver interface API
3  *
4  * Copyright (C) 2013 Nathaniel R. Lewis - http://
    nathanielrlewis.com/
5  *
6  * This program is free software; you can redistribute it and
    /or modify
7  * it under the terms of the GNU General Public License as
    published by
8  * the Free Software Foundation; either version 2 of the
    License, or
9  * (at your option) any later version.
10 *
11 * This program is distributed in the hope that it will be
    useful,
12 * but WITHOUT ANY WARRANTY; without even the implied
    warranty of
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
    the
14 * GNU General Public License for more details.
15 *
16 * You should have received a copy of the GNU General Public
    License
17 * along with this program; if not, write to the Free
    Software
18 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
19 *
20 *
21 * This code is changed by Jelle Spijker (C) 2014.
22 * Introducing polling with threading.
23 *
24 */
25
26 #pragma once
27
28 #include <iostream>
29 #include <stdint.h>
30 #include <string>
31 #include "BBB.h"
32
33 #define eQEP0 "/sys/devices/ocp.3/48300000.epwmss/48300180.
    eqep"
34 #define eQEP1 "/sys/devices/ocp.3/48302000.epwmss/48302180.
    eqep"
35 #define eQEP2 "/sys/devices/ocp.3/48304000.epwmss/48304180.
    eqep"
36
37 namespace Hardware {
38 // Class which defines an interface to my eQEP driver
39 class eQEP : public BBB {
40 // Base path for the eQEP unit
41     std::string path;
42

```

```

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
88 void *threadedPolleqep(void *value);
89

```

```

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

```

```

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

```

```

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
            position for write"
73             << std::endl;
74         return;
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
            period for write"
93             << std::endl;
94         return;
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_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
        mode for write"
113         << std::endl;
114     return;
115 }
116
117 // Write the desired value to the file
118 fprintf(fp, "%u\n", _mode);
119
120 // Commit changes
121 fclose(fp);
122 }
123
124 int eQEP::WaitForPositionChange(CallbackType callback) {
125     threadRunning = true;
126     callbackFunction = callback;
127     if (pthread_create(&this->thread, NULL, &threadedPolleqep,
128         static_cast<void *>(this))) {
129         threadRunning = false;
130         throw Exception::
            FailedToCreateGPIOPollingThreadException();
131     }
132
133     return 0;
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             return 0;
153         }
154     }
155
156     // Dummy read
157     read(ufd.fd, &dummy, 1);
158
159     // Poll the port
160     ufd.events = (short)EPOLLET;
161     if (poll(&ufd, 1, -1) < 0) {
162         // Error, break out
163         std::cerr << "[eQEP " << this->path << "] Error

```

```

        occurred whilst polling"
161         << std::endl;
162         close(ufd.fd);
163         return 0;
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
        position for read"
174         << std::endl;
175     close(ufd.fd);
176     return 0;
177 }
178
179 // Write the desired value to the file
180 fscanf(fp, "%d", &position);
181
182 // Commit changes
183 fclose(fp);
184
185 // If we were polling, close the polling file
186 if (_poll) {
187     close(ufd.fd);
188 }
189
190 // Return the position
191 return position;
192 }
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
            period for read"
203             << std::endl;
204         return 0;
205     }
206
207     // Write the desired value to the file
208     uint64_t period = 0;
209     fscanf(fp, "%llu", &period);
210
211     // Commit changes
212     fclose(fp);
213

```



```
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
                mode for read"
                << std::endl;
227         return eQEP::eQEP_Mode_Error;
228     }
229
230     // Write the desired value to the file
231     eQEP::eQEP_Mode mode;
232     fscanf(fp, "%u", (unsigned int *)&mode);
233
234     // Commit changes
235     fclose(fp);
236
237     // Return the mode
238     return mode;
239 }
240
241
242 void *threadedPolleqep(void *value) {
243     eQEP *eqep = static_cast<eQEP *>(value);
244     while (eqep->threadRunning) {
245         eqep->callbackFunction(eqep->get_position(true));
246         usleep(eqep->debounceTime * 1000);
247     }
248     return 0;
249 }
250 }
```

D.0.8 SoilCape Class

```

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 "EC12P.h"
11 #include "GPIO.h"
12 #include "PWM.h"
13 #include "ADC.h"
14
15 namespace Hardware {
16 class SoilCape {
17 public:
18     EC12P RGBEncoder;
19     PWM MicroscopeLEDs{PWM::P9_14};
20     ADC MicroscopeLDR{ADC::ADC0};
21
22     SoilCape();
23     ~SoilCape();
24 };
25 }

```

```

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 "SoilCape.h"
9
10 namespace Hardware {
11 SoilCape::SoilCape() {}
12
13 SoilCape::~~SoilCape() {}
14 }

```

D.0.9 USB Class

```

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 <stdio.h>
11 #include <unistd.h>
12 #include <fcntl.h>
13 #include <errno.h>
14 #include <sys/ioctl.h>
15
16 #include <linux/usbdevice_fs.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
   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 "USB.h"
9
10 namespace Hardware {
11 USB::USB() {}
12
13 USB::~~USB() {}
14
15 void USB::ResetUSB() {
16     int fd, rc;
17
18     fd = open("/dev/bus/usb/001/002", O_WRONLY);
19     rc = ioctl(fd, USBDEVFS_RESET, 0);
20     if (rc < 0) {
21         throw - 1;
22     }
23     close(fd);
24 }

```

25 }

D.0.10 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         opencv_photo -lopencv_imgcodecs -lopencv_videoio
49 INCLUDEPATH += /usr/local/include/opencv
50 INCLUDEPATH += /usr/local/include
51
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 <spijker.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 <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 ValueOutOfBoundsException : public std::exception {
20         public:

```

```

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

```

```

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  #include <exception>
10 #include <string>
11
12 using namespace std;
13
14 namespace Hardware {
15     namespace Exception {
16         class ADCReadException : public std::exception {
17         public:
18             ADCReadException(string m = "Can't read ADC data!") : msg(
                m){};
19             ~ADCReadException() _GLIBCXX_USE_NOEXCEPT{};
20             const char *what() const _GLIBCXX_USE_NOEXCEPT { return
                msg.c_str(); };
21
22         private:
23             string msg;
24         };
25     }
26 }

```

```

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 <exception>
11 #include <string>

```

```

12
13 using namespace std;
14
15 namespace Hardware {
16 namespace Exception {
17 class FailedToCreateGPIOPollingThreadException : public std
    ::exception {
18 public:
19     FailedToCreateGPIOPollingThreadException(
20         string m = "Failed to create GPIO polling thread!")
21         : msg(m){};
22     ~FailedToCreateGPIOPollingThreadException()
23         _GLIBCXX_USE_NOEXCEPT{};
24     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
25         msg.c_str(); };
26 private:
27     string msg;
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
10 #pragma once
11
12 #include <exception>
13 #include <string>
14
15 using namespace std;
16
17 namespace Hardware {
18 namespace Exception {
19 class FailedToCreateThreadException : public std::exception
20 {
21 public:
22     FailedToCreateThreadException(string m = "Couldn't create
23         the thread!")
24         : msg(m){};
25     ~FailedToCreateThreadException() _GLIBCXX_USE_NOEXCEPT{};
26     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
27         msg.c_str(); };
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   */
9  #define EXCEPTION_OPENCAM "Exception could not open cam!"
10 #define EXCEPTION_OPENCAM_NR 0
11 #define EXCEPTION_NOCAMS "Exception no cam available!"
12 #define EXCEPTION_NOCAMS_NR 1
13 #define EXCEPTION_QUERY "Exception could not query device!"
14 #define EXCEPTION_QUERY_NR 3
15 #define EXCEPTION_FORMAT_RESOLUTION "Exception No supported
16   formats and resolutions!"
17 #define EXCEPTION_FORMAT_RESOLUTION_NR 4
18 #define EXCEPTION_CTRL_NOT_FOUND "Control not found!"
19 #define EXCEPTION_CTRL_NOT_FOUND_NR 5
20 #define EXCEPTION_CTRL_VALUE "Control value not set!"
21 #define EXCEPTION_CTRL_VALUE_NR 5
22
23 #pragma once
24 #include <exception>
25 #include <string>
26
27 using namespace std;
28
29 namespace Hardware {
30 namespace Exception {
31 class MicroscopeException : public std::exception {
32 public:
33     MicroscopeException(string m = EXCEPTION_OPENCAM,
34                         int n = EXCEPTION_OPENCAM_NR) : msg{m
35                             }, nr{n} { }
36     ~MicroscopeException() _GLIBCXX_USE_NOEXCEPT {}
37     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
38         msg.c_str(); }
39     const int *id() const _GLIBCXX_USE_NOEXCEPT { return &nr;
40     }
41 private:
42     string msg;
43     int nr;
44 };
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

```

```

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 private:
25     string msg;
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  */
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 private:
27     string msg;
28 };
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 #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 };
30 }
31 }
```



E. Vision Library

E.0.1 Image processing Class

```
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  /*! Current class version*/
10 #define IMAGEPROCESSING_VERSION 1
11
12 /*! MACRO which sets the original pointer to the original
   image or a clone of
13  * the earlier processed image */
14 #define CHAIN_PROCESS(chain, O, type)
15
16     if (chain) {
17         \
18         TempImg = ProcessedImg.clone();
19         \
20         O = (type *)TempImg.data;
21     } else {
22         \
23         O = (type *)OriginalImg.data;
24     }
25 }
```

```

21  /*! MACRO which throws an EmptyImageException if the matrix
    is empty*/
22  #define EMPTY_CHECK(img)
23      if (img.empty()) {
24          \
          throw Exception::EmptyImageException();
25      }
26
27  #include <opencv2/core.hpp>
28  #include <opencv2/highgui.hpp>
29  #include <opencv2/imgproc.hpp>
30
31  #include <stdint.h>
32  #include <cmath>
33  #include <vector>
34  #include <string>
35
36  #include <boost/signals2.hpp>
37  #include <boost/bind.hpp>
38
39  #include "EmptyImageException.h"
40  #include "WrongKernelSizeException.h"
41  #include "ChannelMismatchException.h"
42  #include "PixelValueOutOfBoundsException.h"
43  #include "VisionDebug.h"
44
45  using namespace cv;
46
47  namespace Vision {
48  class ImageProcessing {
49  public:
50      typedef boost::signals2::signal<void(float, std::string)>
          Progress_t;
51      boost::signals2::connection
52      connect_Progress(const Progress_t::slot_type &subscriber);
53
54  protected:
55      uchar *GetNRow(int nData, int hKsize, int nCols, uint32_t
          totalRows);
56      Mat TempImg;
57
58      Progress_t prog_sig;
59
60  public:
61      ImageProcessing();
62      ~ImageProcessing();
63      Mat OriginalImg;
64      Mat ProcessedImg;
65
66      static void getOrientented(Mat &BW, cv::Point_<double> &
          centroid,
67          double &theta, double &
          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 == 3 || cvType == 11 || cvType == 19
109             || cvType == 27) {
110             for (uint32_t i = 0; i < nData; i += src.step[1]) {
111                 dst.data[i] =
112                     static_cast<int16_t>(LUT[*](T2 *)(src.data + (i *
113                         src.step[1]))));
114             }
115         } else if (cvType == 4 || cvType == 12 || cvType == 20
116             || cvType == 28) {
117             for (uint32_t i = 0; i < nData; i += src.step[1]) {
118                 dst.data[i] =
119                     static_cast<int32_t>(LUT[*](T2 *)(src.data + (i *
120                         src.step[1]))));
121             }
122         }
123         return dst;
124     }

```

```

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
           values
117     \param cvType an in where you can pas CV_UC8C1 etc.
118     \return The new matrix
119     */
120     template <typename T1>
121     static Mat CopyMat(const Mat &src, const Mat &mask, int
           cvType) {
122         if (src.size() != mask.size) {
123             throw Exception::WrongKernelSizeException(
124                 "Mask not the same size as src Exception!");
125         }
126         if (mask.channels() != 1) {
127             throw Exception::WrongKernelSizeException(
128                 "Mask has more then 1 channel Exception!");
129         }
130         Mat dst(src.size(), cvType);
131
132         vector<Mat> exSrc = Vision::ImageProcessing::
           extractChannel(src);
133         vector<Mat> exDst;
134
135         int cvBaseType = cvType % 8;
136         for_each(exSrc.begin(), exSrc.end(), [&](const Mat &
           sItem) {
137             Mat dItem(src.size(), cvBaseType);
138             std::transform(sItem.begin<T1>(), sItem.end<T1>(),
           mask.begin<T1>(),
139                           dItem.begin<T1>(),
140                           [](const T1 &s, const T1 &m) -> T1 {
           return s * m; });
141             exDst.push_back(dItem);
142         });
143
144         merge(exDst, dst);
145
146         return dst;
147     }
148
149     static cv::Mat WhiteBackground(const cv::Mat &src) {
150         cv::Mat dst;
151         cv::floodFill(src, dst, cv::Point(0, 0), cv::Scalar(255,
           255, 255));
152         return dst;
153     }
154
155     template <typename T1>
156     static void ShowDebugImg(cv::Mat img, T1 maxVal, std::
           string windowName,
157                             bool scale = true) {
158         if (img.rows > 0 && img.cols > 0) {
159             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             });
179
180         int Range = MatMax - MatMin;
181         if (Range < 1)
182             Range = maxVal;
183
184         // Convert the values
185         for_each(exSrc.begin(), exSrc.end(), [&](const cv::
186             Mat &sItem) {
187             Mat dItem(img.size(), cvBaseType);
188             std::transform(sItem.begin<T1>(), sItem.end<T1>(),
189                 dItem.begin<T1>(),
190                 [&](const T1 &s) -> T1 {
191                     return (T1)round(((s - MatMin) *
192                         maxVal) / Range);
193                 });
194             exDst.push_back(dItem);
195         });
196
197         merge(exDst, tempImg);
198     } else {
199         tempImg = img;
200     }
201 }
202 }
203 }

```

```

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  /*! \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 uchars 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,
31                                uint32_t totalRows) {
32     // Create LUT to determine when there is an new row
33     uchar *nRow = new uchar[nData + 1]{};
34     // int i = 0;
35     int shift = nCols - hKsize - 1;
36     for (uint32_t i = 0; i < totalRows; i++) {
37         nRow[(i * nCols) + shift] = 1;
38     }
39     return nRow;
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
52     centroid.x = Mu.m10 / Mu.m00;

```



```
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 }
```

E.0.2 Conversion Class

```

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  #include "ImageProcessing.h"
10 #include "ConversionNotSupportedException.h"
11
12 namespace Vision {
13 class Conversion : public ImageProcessing {
14 public:
15     /*! Enumerator which indicates the colorspace used*/
16     enum ColorSpace {
17         CIE_lab,    /*!< CIE La*b* colorspace */
18         CIE_XYZ,    /*!< CIE XYZ colorspace */
19         RI,         /*!< Redness Index colorspace */
20         RGB,        /*!< RGB colorspace */
21         Intensity,  /*!< Grayscale colorspace */
22         None        /*!< none */
23     };
24     ColorSpace OriginalColorSpace; /*!< The original
   colorspace*/
25     ColorSpace ProcessedColorSpace; /*!< The destination
   colorspace*/
26
27     Conversion();
28     Conversion(const Mat &src);
29     Conversion(const Conversion &rhs);
30
31     ~Conversion();
32
33     Conversion &operator=(Conversion rhs);
34
35     void Convert(ColorSpace convertFrom, ColorSpace convertTo,
36                 bool chain = false);
37     void Convert(const Mat &src, Mat &dst, ColorSpace
38                 convertFrom,
39                 ColorSpace convertTo, bool chain = false);
40 private:
41     /*!< Conversion matrix used in the conversion between RGB
   and CIE XYZ*/
42     float XYZmat[3][3] = {{0.412453, 0.357580, 0.180423},
43                           {0.212671, 0.715160, 0.072169},
44                           {0.019334, 0.119194, 0.950227}};
45
46     float whitePoint[3] = {
47         0.9504, 1.0000, 1.0889}; /*!< Natural whitepoint in
   XYZ colorspace D65

```

```

48                                     according to Matlab */
49 // float whitePoint[3] = { 0.9642, 1.0000, 0.8251 }; /*!<
    Natural whitepoint
50 // in XYZ colorspace D50 according to Matlab */
51
52 void Lab2RI(float *O, float *P, int nData);
53 void RGB2XYZ(uchar *O, float *P, int nData);
54 void XYZ2Lab(float *O, float *P, int nData);
55 void RGB2Intensity(uchar *O, 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
    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 Conversion
9  class which converts a cv::Mat image from one colorspace to
    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-structor 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
    Intensity
100 {
101     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
102     uchar *P = ProcessedImg.data;
103     uchar *0;
104     CHAIN_PROCESS(chain, 0, uchar);
105
106     prog_sig(currentProg, "RGB 2 Intensity conversion");
107     RGB2Intensity(0, P, nData);
108     currentProg += ProgStep;
109     prog_sig(currentProg, "RGB 2 Intensity conversion
    Finished");
110 } else if (convertFrom == RGB && convertTo == CIE_XYZ) //
    RGB 2 XYZ
111 {
112     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
113     float *P = (float *)ProcessedImg.data;
114     uchar *0;
115     CHAIN_PROCESS(chain, 0, uchar);
116
117     prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
118     RGB2XYZ(0, P, nData);
119     currentProg += ProgStep;
120     prog_sig(currentProg, "RGB 2 CIE XYZ conversion Finished
    ");
121 } else if (convertFrom == RGB && convertTo == CIE_lab) //
    RGB 2 Lab
122 {
123     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
124     float *P = (float *)ProcessedImg.data;
125     uchar *0;
126     CHAIN_PROCESS(chain, 0, uchar);
127
128     prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
129     RGB2XYZ(0, P, nData);
130     currentProg += ProgStep;
131     prog_sig(currentProg, "RGB 2 CIE XYZ conversion Finished
    ");
132     Convert(CIE_XYZ, CIE_lab, true);
133 } else if (convertFrom == RGB && convertTo == RI) // RGB 2
    RI
134 {
135     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
136     float *P = (float *)ProcessedImg.data;
137     uchar *0;
138     CHAIN_PROCESS(chain, 0, uchar);
139

```

```

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 {
148     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
149     float *P = (float *)ProcessedImg.data;
150     float *0;
151     CHAIN_PROCESS(chain, 0, float);
152
153     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion");
154     XYZ2Lab(0, P, nData);
155     currentProg += ProgStep;
156     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion
157                Finished");
157 } else if (convertFrom == CIE_XYZ && convertTo == RI) //
158     XYZ 2 RI
159 {
159     ProcessedImg.create(OriginalImg.size(), CV_32FC3);
160     float *P = (float *)ProcessedImg.data;
161     float *0;
162     CHAIN_PROCESS(chain, 0, float);
163
164     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion");
165     XYZ2Lab(0, P, nData);
166     currentProg += ProgStep;
167     prog_sig(currentProg, "CIE XYZ 2 CIE La*b* conversion
168                Finished");
168     Convert(CIE_lab, RI, true);
169 } else if (convertFrom == CIE_lab && convertTo == RI) //
170     Lab 2 RI
171 {
171     ProcessedImg.create(OriginalImg.size(), CV_32FC1);
172     float *P = (float *)ProcessedImg.data;
173     float *0;
174     CHAIN_PROCESS(chain, 0, float);
175
176     prog_sig(currentProg, "CIE La*b* 2 Redness Index
177                conversion");
177     Lab2RI(0, P, nData * 3);
178     currentProg += ProgStep;
179     prog_sig(currentProg, "CIE La*b* 2 Redness Index
180                conversion Finsihed");
180 } else {
181     throw Exception::ConversionNotSupportedException();
182 }
183 }
184
185 /*! Conversion from RGB to Intensity
186 \param 0 a uchar pointer to the source image
187 \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     uint32_t i;
192     int j;
193     i = 0;
194     j = 0;
195     while (j < nData) {
196         P[j++] = (*(O + i + 2) * 0.2126 + *(O + i + 1) * 0.7152
            +
197                 *(O + i) * 0.0722); // Grey value
198         i += 3;
199     }
200 }
201
202 /*! Conversion from RGB to CIE XYZ
203 \param O a uchar pointer to the source image
204 \param P a uchar pointer to the destination image
205 \param nData an int indicating the total number of pixels
206 */
207 void Conversion::RGB2XYZ(uchar *O, float *P, int nData) {
208     uint32_t endData = nData * OriginalImg.step.buf[1];
209     float R, G, B;
210     for (uint32_t i = 0; i < endData; i += OriginalImg.step.
        buf[1]) {
211         R = static_cast<float>(*(O + i + 2) / 255.0f);
212         B = static_cast<float>(*(O + i + 1) / 255.0f);
213         G = static_cast<float>(*(O + i) / 255.0f);
214         P[i] = (XYZmat[0][0] * R) + (XYZmat[0][1] * B) + (XYZmat
            [0][2] * G); // X
215         P[i + 1] = (XYZmat[1][0] * R) + (XYZmat[1][1] * B) + (
            XYZmat[1][2] * G); // Y
216         P[i + 2] = (XYZmat[2][0] * R) + (XYZmat[2][1] * B) + (
            XYZmat[2][2] * G); // Z
217     }
218 }
219
220 /*! Conversion from CIE XYZ to CIE La*b*
221 \param O a uchar pointer to the source image
222 \param P a uchar pointer to the destination image
223 \param nData an int indicating the total number of pixels
224 */
225 void Conversion::XYZ2Lab(float *O, float *P, int nData) {
226     uint32_t endData = nData * 3;
227     float yy0, xx0, zz0;
228     for (size_t i = 0; i < endData; i += 3) {
229         xx0 = *(O + i) / whitePoint[0];
230         yy0 = *(O + i + 1) / whitePoint[1];
231         zz0 = *(O + i + 2) / whitePoint[2];
232
233         if (yy0 > 0.008856) {
234             P[i] = (116 * pow(yy0, 0.333f)) - 16; // L
235         } else {
236             P[i] = 903.3 * yy0; // L
237         }

```

```
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, 0.333333333333f);
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 }
```

```

45     void AdaptiveContrastStretch(const Mat &src, Mat &dst,
46                                   uint8_t kernelsize,
47                                   float factor);
48     void Blur(uint8_t kernelsize, bool chain = false);
49     void Blur(const Mat &src, Mat &dst, uint8_t kernelsize);
50
51     void HistogramEqualization(bool chain = false);
52     void HistogramEqualization(const Mat &src, Mat &dst);
53 };
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 <spijker.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
62 Enhance &Enhance::operator=(Enhance rhs) {
63     if (&rhs != this) {
64         this->OriginalImg = rhs.OriginalImg;
65         this->ProcessedImg = rhs.ProcessedImg;
66         this->TempImg = rhs.ProcessedImg;
67     }
68     return *this;
69 }
70
71 /*! Calculate the standard deviation of the neighboring
72     pixels
73 \param 0 uchar pointer to the current pixel of the original
74     image
75 \param i current counter
76 \param hKsize half the kernelsize
77 \param nCols total number of columns
78 \param noNeighboursPix total number of neighboring pixels
79 \param mean mean value of the neighboring pixels
80 \return standard deviation
81 */
82 float Enhance::CalculateStdOfNeighboringPixels(uchar *0, int
83     i, int hKsize,
84                                     int nCols,
85                                     int
86                                     noNeighboursPix
87                                     ,
88                                     float mean) {
89     uint32_t sum_dev = 0.0;
90     float Std = 0.0;
91     sum_dev = 0.0;
92     Std = 0.0;
93     for (int j = -hKsize; j < hKsize; j++) {

```

```

88     for (int k = -hKsize; k < hKsize; k++) {
89         // sum_dev += pow((O[i + j * nCols + k] - mean), 2);
90         sum_dev += SoilMath::quick_pow2((O[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
99 /*! Calculate the sum of the neighboring pixels
100 \param O uchar pointer to the current pixel of the original
101       image
102 \param i current counter
103 \param hKsize half the kernelsize
104 \param nCols total number of columns
105 \param sum Total sum of the neighboringpixels
106 */
107 void Enhance::CalculateSumOfNeighboringPixels(uchar *O, int
108     i, int hKsize,
109                                     int nCols,
110                                     uint32_t &
111                                     sum) {
112     for (int j = -hKsize; j < hKsize; j++) {
113         for (int k = -hKsize; k < hKsize; k++) {
114             sum += O[i + j * nCols + k];
115         }
116     }
117 }
118
119 /*! Homebrew AdaptiveContrastStretch function which
120     calculate the mean and
121     standard deviation from the neighboring pixels if the
122     current pixel is higher
123     then the mean the value is incremented with an given factor
124     multiplied with the
125     standard deviation, and decreased if it's lower then the
126     mean.
127 \param src cv::Mat source image
128 \param dst cv::Mat destination image
129 \param kernelsize an uchar which represent the kernelsize
130                 should be an uneven
131                 number higher than two
132 \param factor float which indicates the amount the effect
133                 should take place
134 standard value is 1.0 only used in the adaptive contrast
135 stretch enhancement
136 */
137 void Enhance::AdaptiveContrastStretch(const Mat &src, Mat &
138     dst,
139                                     uchar kernelsize,
140                                     float factor) {
141     OriginalImg = src;
142     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     // Exception handling
155     EMPTY_CHECK(OriginalImg);
156     if (kernelsize < 3 || (kernelsize % 2) == 0) {
157         throw Exception::WrongKernelSizeException();
158     }
159     CV_Assert(OriginalImg.depth() != sizeof(uchar));
160
161     // Make the pointers to the Data
162     uchar *O;
163     CHAIN_PROCESS(chain, O, uchar);
164     uchar *P = ProcessedImg.data;
165
166     int i = 0;
167     int hKsize = kernelsize / 2;
168     int nCols = OriginalImg.cols;
169     int pStart = (hKsize * nCols) + hKsize + 1;
170
171     int nData = OriginalImg.rows * OriginalImg.cols;
172     int pEnd = nData - pStart;
173     uint32_t noNeighboursPix = kernelsize * kernelsize;
174     uint32_t sum;
175     float mean = 0.0;
176
177     uchar *nRow = GetNRow(nData, hKsize, nCols, OriginalImg.
178         rows);
179
180     i = pStart;
181     while (i++ < pEnd) {
182         // Checks if pixel isn't a border pixel and progresses
183             to the new row
184         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 mean = (float)(sum / noNeighboursPix);
184 float Std = CalculateStdOfNeighboringPixels(0, i, hKsize
    , nCols,
185                                             noNeighboursPix
    , mean);
186
187 // Stretch
188
189 if (O[i] > mean) {
190     // int addValue = O[i] + (int)(round(factor * Std));
191     int addValue = O[i] + static_cast<int>(round(factor *
        Std));
192     if (addValue < 255) {
193         P[i] = addValue;
194     } else {
195         P[i] = 255;
196     }
197 } else if (O[i] < mean) {
198     // int subValue = O[i] - (int)(round(factor * Std));
199     int subValue = O[i] - static_cast<int>(round(factor *
        Std));
200     if (subValue > 0) {
201         P[i] = subValue;
202     } else {
203         P[i] = 0;
204     }
205 } else {
206     P[i] = O[i];
207 }
208 }
209
210 // Stretch the image with an normal histogram equalization
211 HistogramEqualization(true);
212
213 delete[] nRow;
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
    should be an uneven
220 number higher than two
221 */
222 void Enhance::Blur(const Mat &src, Mat &dst, uchar
    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, 0, 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
273         P[i] = (uchar)(round(sum / noNeighboursPix));
274     }
275     delete[] nRow;
276 }

```

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

E.0.4 Morphological filter Class

```

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  #define MORPHOLOGICALFILTER_VERSION 1
10
11 #include "ImageProcessing.h"
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
        = FilterType::NONE);
21     MorphologicalFilter(const MorphologicalFilter &rhs);
22
23     ~MorphologicalFilter();
24
25     MorphologicalFilter &operator=(MorphologicalFilter &rhs);
26
27     void Dilation(const Mat &mask, bool chain = false);
28     void Erosion(const Mat &mask, bool chain = false);
29
30     void Close(const Mat &mask, bool chain = false);
31     void Open(const Mat &mask, bool chain = false);
32
33 private:
34     void Filter(const Mat &mask, bool chain, uchar startVal,
        uchar newVal,
35                uchar switchVal);
36 };
37 }

```

```

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 "MorphologicalFilter.h"
9
10 namespace Vision {

```

```

11 MorphologicalFilter::MorphologicalFilter() {}
12
13 MorphologicalFilter::MorphologicalFilter(FilterType
    filtertype) {
14     switch (filtertype) {
15     case FilterType::OPEN:
16         Open(OriginalImg);
17         break;
18     case FilterType::CLOSE:
19         Close(OriginalImg);
20         break;
21     case FilterType::ERODE:
22         Erosion(OriginalImg);
23         break;
24     case FilterType::DILATE:
25         Dilation(OriginalImg);
26         break;
27     case FilterType::NONE:
28         break;
29     }
30 }
31
32 MorphologicalFilter::MorphologicalFilter(const Mat &src,
33                                         FilterType
                                         filtertype) {
34     OriginalImg = src;
35     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
36     switch (filtertype) {
37     case FilterType::OPEN:
38         Open(OriginalImg);
39         break;
40     case FilterType::CLOSE:
41         Close(OriginalImg);
42         break;
43     case FilterType::ERODE:
44         Erosion(OriginalImg);
45         break;
46     case FilterType::DILATE:
47         Dilation(OriginalImg);
48         break;
49     case FilterType::NONE:
50         break;
51     }
52 }
53
54 MorphologicalFilter::MorphologicalFilter(const
    MorphologicalFilter &rhs) {
55     this->OriginalImg = rhs.OriginalImg;
56     this->ProcessedImg = rhs.ProcessedImg;
57     this->TempImg = rhs.ProcessedImg;
58 }
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
77 void MorphologicalFilter::Close(const Mat &mask, bool chain)
78 {
79     Dilation(mask, chain);
80     Erosion(mask, true);
81 }
82
83 void MorphologicalFilter::Dilation(const Mat &mask, bool
84     chain) {
85     Filter(mask, chain, 0, 1, 1);
86 }
87
88 void MorphologicalFilter::Erosion(const Mat &mask, bool
89     chain) {
90     Filter(mask, chain, 1, 0, 0);
91 }
92
93 void MorphologicalFilter::Filter(const Mat &mask, bool chain
94     , uchar startVal,
95     uchar newVal, uchar
96     switchVal) {
97
98     // Exception handling
99     CV_Assert(OriginalImg.depth() != sizeof(uchar));
100     EMPTY_CHECK(OriginalImg);
101     if (mask.cols % 2 == 0 || mask.cols < 3) {
102         throw Exception::WrongKernelSizeException("Wrong
103             Kernelsize columns!");
104     }
105     if (mask.rows % 2 == 0 || mask.rows < 3) {
106         throw Exception::WrongKernelSizeException("Wrong
107             Kernelsize rows!");
108     }
109
110     uint32_t hKsizeCol = (mask.cols / 2);
111     uint32_t hKsizeRow = (mask.rows / 2);
112
113     // make Pointers
114     Mat workOrigImg(ProcessedImg.rows + mask.rows,
115         ProcessedImg.cols + mask.cols,
116         CV_8UC1);
117     workOrigImg.setTo(0);
118     if (chain) {
119         ProcessedImg.copyTo(workOrigImg(

```

```

110         cv::Rect(hKsizeCol, hKsizeRow, ProcessedImg.cols,
                ProcessedImg.rows)));
111     // workOrigImg(cv::Rect(hKsizeCol, hKsizeRow,
                ProcessedImg.cols,
112     // ProcessedImg.rows)) = ProcessedImg.clone();
113 } else {
114     OriginalImg.copyTo(workOrigImg(
115         cv::Rect(hKsizeCol, hKsizeRow, ProcessedImg.cols,
                ProcessedImg.rows)));
116     // workOrigImg(cv::Rect(hKsizeCol, hKsizeRow,
                ProcessedImg.cols,
117     // ProcessedImg.rows)) = OriginalImg.clone();
118 }
119 uchar *O = workOrigImg.data;
120
121 Mat workProcImg(ProcessedImg.rows + mask.rows,
                ProcessedImg.cols + mask.cols,
122                 CV_8UC1);
123 uchar *P = workProcImg.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) +
                hKsizeRow;
129 uint32_t nWEnd = nWData - hKsizeCol - hKsizeRow *
                workProcImg.cols - 1;
130 uchar *nRow = GetNRow(nWData, hKsizeCol, workProcImg.cols,
                workProcImg.rows);
131 int MaskPixel = 0, OPixel = 0;
132
133 workProcImg.setTo(0);
134 if (startVal != 0) {
135     workProcImg(cv::Rect(hKsizeCol, hKsizeRow, ProcessedImg.
                cols,
136                         ProcessedImg.rows)).setTo(startVal)
                ;
137 }
138 SHOW_DEBUG_IMG(workOrigImg, uchar, 255, "workOrigImg
                Filter!", false);
139 SHOW_DEBUG_IMG(mask, uchar, 255, "Filter mask", true);
140
141 for (uint32_t i = nWStart; i < nWEnd; i++) {
142     // Checks if pixel isn't a border pixel and progresses
                to the new row
143     if (nRow[i] == 1) {
144         i += mask.cols;
145     }
146     for (int r = 0; r < mask.rows; r++) {
147         for (int c = 0; c < mask.cols; c++) {
148             MaskPixel = c + r * mask.cols;
149             OPixel = i - hKsizeCol + c + (r - hKsizeRow) *
                workProcImg.cols;
150             if (mask.data[MaskPixel] == 1 && O[OPixel] ==
                switchVal) {
151                 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
    Filter!", true);
160 ProcessedImg = workProcImg(Rect(hKsizeCol, hKsizeRow,
    ProcessedImg.cols,
161                               ProcessedImg.rows)).clone
    ();
162 SHOW_DEBUG_IMG(ProcessedImg, uchar, 255, "Processed Image
    Filter!", true);
163 }
164 }
```

E.0.5 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 "opencv2/imgproc/imgproc.hpp"
23
24 #include "ImageProcessing.h"
25 #include "MorphologicalFilter.h"
26 #include "../SoilMath/SoilMath.h"
27
28 namespace Vision {
29     class Segment : public ImageProcessing {
30     public:
31         /*! Coordinates for the region of interest*/
32         typedef struct Rect {
33             uint16_t leftX; /*!< Left X coordinate*/
34             uint16_t leftY; /*!< Left Y coordinate*/
35             uint16_t rightX; /*!< Right X coordinate*/
36             uint16_t rightY; /*!< Right Y coordinate*/
37             Rect(uint16_t lx, uint16_t ly, uint16_t rx, uint16_t ry)
38                 : leftX(lx), leftY(ly), rightX(rx), rightY(ry){}
39         } Rect_t;
40
41         typedef std::vector<Vision::Segment::Rect_t> RectList_t;
42
43         /*! Individual blob*/
44         typedef struct Blob {
45             uint16_t Label; /*!< ID of the blob*/
46             cv::Mat Img; /*!< BW image of the blob all the pixel
47                 belonging to the blob
48                 are set to 1 others are 0*/
49             cv::Rect ROI; /*!< Coordinates for the blob in the
50                 original picture as a
51                 cv::Rect*/
52             uint32_t Area; /*!< Calculated stats of the blob*/
53             cv::Point_<double> Centroid;
54             double Theta;

```



```

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

```

```

91     uint8_t ThresholdLevel = 0;    /*!< Current calculated
        threshold level*/
92
93     float sigma = 2;
94     uint32_t thresholdOffset = 4;
95
96     Segment();
97     Segment(const Mat &src);
98     Segment(const Segment &rhs);
99
100    ~Segment();
101
102    Segment &operator=(Segment &rhs);
103
104    void LoadOriginalImg(const Mat &src);
105
106    void ConvertToBW(TypeOfObjects Typeobjects);
107    void ConvertToBW(const Mat &src, Mat &dst, TypeOfObjects
        Typeobjects);
108
109    void GetEdges(bool chain = false, Connected conn = Eight);
110    void GetEdges(const Mat &src, Mat &dst, bool chain = false
        ,
111                Connected conn = Eight);
112
113    void GetEdgesEroding(bool chain = false);
114
115    void GetBlobList(bool chain = false, Connected conn =
        Eight);
116
117    void Threshold(uchar t, TypeOfObjects Typeobjects);
118
119    void LabelBlobs(bool chain = false, uint16_t minBlobArea =
        25,
120                Connected conn = Eight);
121
122    void RemoveBorderBlobs(uint32_t border = 1, bool chain =
        false);
123
124    void FillHoles(bool chain = false);
125
126 private:
127     uint8_t GetThresholdLevel(TypeOfObjects TypeObject);
128     void SetBorder(uchar *P, uchar setValue);
129     void FloodFill(uchar *O, uchar *P, uint16_t x, uint16_t y,
        uchar fillValue,
130                uchar OldValue);
131     void MakeConsecutive(uint16_t *valueArr, uint32_t noElem,
        uint16_t &maxlabel);
132     void MakeConsecutive(uint16_t *valueArr, uint16_t *keyArr,
        uint16_t noElem,
133                uint16_t &maxlabel);
134     void SortAdjacencyList(std::vector<std::vector<uint16_t>>
        &adj);
135     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 (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
10 /*! \class Segment
11 \brief Segmentation algorithms
12 With this class, various segmentation routines can be
13 applied to a greyscale or
14 black and white source image.
15 */
16 #include "Segment.h"
17
18 namespace Vision {
19     //! Constructor of the Segmentation class
20     Segment::Segment() {}
21
22     //! Constructor of the Segmentation class
23     Segment::Segment(const Mat &src) {
24         OriginalImg = src;
25         ProcessedImg.create(OriginalImg.size(), CV_8UC1);
26         LabelledImg.create(OriginalImg.size(), CV_16UC1);
27     }
28
29     Segment::Segment(const Segment &rhs) {
30         this->BlobList = rhs.BlobList;
31         this->LabelledImg = rhs.LabelledImg;
32         this->MaxLabel = rhs.MaxLabel;
33         this->noOfFilteredBlobs = rhs.noOfFilteredBlobs;
34         this->OriginalImg = rhs.OriginalImg;
35         this->OriginalImgStats = rhs.OriginalImgStats;
36         this->ProcessedImg = rhs.ProcessedImg;
37         this->TempImg = rhs.TempImg;
38         this->ThresholdLevel = rhs.ThresholdLevel;
39     }
40
41     //! De-structor
42     Segment::~Segment() {}
43
44     Segment &Segment::operator=(Segment &rhs) {
45         if (&rhs != this) {
46             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
63    distribution,
64    presumably back- and foreground. It works towards the
65    average of the two
66    averages and finally sets the threshold with two time the
67    standard deviation
68    from the mean of the set object
69    \param TypeObject is an enumerator indicating if the bright
70    or the dark pixels
71    are the object and should be set to one
72    \return The threshold level as an uint8_t */
73 uint8_t Segment::GetThresholdLevel(TypeOfObjects TypeObject)
74 {
75     // Exception handling
76     EMPTY_CHECK(OriginalImg);
77     CV_Assert(OriginalImg.depth() != sizeof(uchar));
78
79     // Calculate the statistics of the whole picture
80     ucharStat_t OriginalImgStats(OriginalImg.data, OriginalImg
81         .rows,
82         OriginalImg.cols);
83
84     // Sets the initial threshold with the mean of the total
85     picture
86     pair<uchar, uchar> T;
87     T.first = (uchar)(OriginalImgStats.Mean + 0.5);
88     T.second = 0;
89
90     uchar Rstd = 0;
91     uchar Lstd = 0;
92     uchar Rmean = 0;
93     uchar Lmean = 0;
94
95     // Iterate till optimum Threshold is found between back- &
96     foreground
97     while (T.first != T.second) {
98         // Gets an array of the left part of the histogram
99         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    // Assumes the pixel value of the sought object lies
123    // between 2 sigma
124    int val = 0;
125    switch (TypeObject) {
126    case Bright:
127        val = Rmean - (sigma * Rstd) - thresholdOffset;
128        if (val < 0) {
129            val = 0;
130        } else if (val > 255) {
131            val = 255;
132        }
133        T.first = (uchar)val;
134        break;
135    case Dark:
136        val = Lmean + (sigma * Lstd) + thresholdOffset;
137        if (val < 0) {
138            val = 0;
139        } else if (val > 255) {
140            val = 255;
141        }
142        T.first = (uchar)val;
143        break;
144    }
145    return T.first;

```

```

146 }
147
148 /*! Convert a greyscale image to a BW using an automatic
      Threshold
149 \param src is the source image as a cv::Mat
150 \param dst destination image as a cv::Mat
151 \param TypeObject is an enumerator indicating if the bright
      or the dark pixels
152 are the object and should be set to one */
153 void Segment::ConvertToBW(const Mat &src, Mat &dst,
      TypeOfObjects Typeobjects) {
154     OriginalImg = src;
155     ProcessedImg.create(OriginalImg.size(), CV_8UC1);
156     LabelledImg.create(OriginalImg.size(), CV_16UC1);
157     ConvertToBW(Typeobjects);
158     dst = ProcessedImg;
159 }
160
161 /*! Convert a greyscale image to a BW using an automatic
      Threshold
162 \param TypeObject is an enumerator indicating if the bright
      or the dark pixels
163 are the object and should be set to one */
164 void Segment::ConvertToBW(TypeOfObjects Typeobjects) {
165     // Determine the threshold
166     uchar T = GetThresholdLevel(Typeobjects);
167
168     // Threshold the picture
169     Threshold(T, Typeobjects);
170 }
171
172 /*! Convert a greyscale image to a BW
173 \param t uchar set the value which is the tipping point
174 \param TypeObject is an enumerator indicating if the bright
      or the dark pixels
175 are the object and should be set to one */
176 void Segment::Threshold(uchar t, TypeOfObjects Typeobjects)
    {
177     // Exception handling
178     EMPTY_CHECK(OriginalImg);
179     CV_Assert(OriginalImg.depth() != sizeof(uchar) ||
180             OriginalImg.depth() != sizeof(uint16_t));
181
182     // Create LUT
183     uchar LUT_newValue[256]{0};
184     if (Typeobjects == Bright) {
185         for (uint32_t i = t; i < 256; i++) {
186             LUT_newValue[i] = 1;
187         }
188     } else {
189         for (uint32_t i = 0; i <= t; i++) {
190             LUT_newValue[i] = 1;
191         }
192     }
193
194     // 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
205 /*! Set all the border pixels to a set value
206 \param *P uchar pointer to the Mat.data
207 \param setValue uchar the value which is written to the
208         border pixels
209 */
210 void Segment::SetBorder(uchar *P, uchar setValue) {
211     // Exception handling
212     EMPTY_CHECK(OriginalImg);
213     CV_Assert(OriginalImg.depth() != sizeof(uchar) ||
214             OriginalImg.depth() != sizeof(uint16_t));
215
216     uint32_t nData = OriginalImg.cols * OriginalImg.rows;
217
218     // Set borderPixels to 2
219     uint32_t i = 0;
220     uint32_t pEnd = OriginalImg.cols + 1;
221
222     // Set the top row to value 2
223     while (i < pEnd) {
224         P[i++] = setValue;
225     }
226
227     // Set the bottom row to value 2
228     i = nData + 1;
229     pEnd = nData - OriginalImg.cols;
230     while (i-- > pEnd) {
231         P[i] = setValue;
232     }
233
234     // Sets the first and the last Column to 2
235     i = 1;
236     pEnd = OriginalImg.rows;
237     while (i < pEnd) {
238         P[(i * OriginalImg.cols) - 1] = setValue;
239         P[(i++ * OriginalImg.cols)] = setValue;
240     }
241 }
242
243 /*! Remove the blobs that are connected to the border
244 \param conn set the pixel connection eight or four
245 \param chain use the results from the previous operation
246         default value = false;
247 */
248 void Segment::RemoveBorderBlobs(uint32_t border, bool chain)
249 {
250     CV_Assert(OriginalImg.depth() != sizeof(uchar));

```



```

294     }
295     SHOW_DEBUG_IMG(ProcessedImg, uchar, 255,
296         "Processed Image RemoverBorderBlobs before
           LUT!", true);
297
298     // Change values 2 -> 0
299     uchar LUT_newValue[3]{0, 1, 0};
300     P = ProcessedImg.data;
301     uint32_t nData = rows * cols;
302     for (uint32_t i = 0; i < nData; i++) {
303         P[i] = LUT_newValue[P[i]];
304     }
305
306     SHOW_DEBUG_IMG(ProcessedImg, uchar, 255,
307         "Processed Image RemoverBorderBlobs!", true
           );
308 }
309
310 /*! Label all the individual blobs in a BW source image. The
           result are written
311 to the labelledImg as an ushort
312 \param conn set the pixel connection eight or four
313 \param chain use the results from the previous operation
           default value = false;
314 \param minBlobArea minimum area when an artifact is
           considered a blob
315 */
316 void Segment::LabelBlobs(bool chain, uint16_t minBlobArea,
           Connected conn) {
317     // Exception handling
318     CV_Assert(OriginalImg.depth() != sizeof(uchar));
319     EMPTY_CHECK(OriginalImg);
320
321     // make the Pointers to the data
322     uchar *O;
323     if (chain) {
324         TempImg = ProcessedImg.clone();
325         ProcessedImg = cv::Mat(OriginalImg.rows, OriginalImg.
           cols, CV_16UC1);
326         O = (uchar *)TempImg.data;
327     } else {
328         O = (uchar *)OriginalImg.data;
329     }
330     uint16_t *P = (uint16_t *)LabelledImg.data;
331
332     uint32_t nCols = OriginalImg.cols;
333     uint32_t nRows = OriginalImg.rows;
334     uint32_t nData = nCols * nRows;
335
336     vector<vector<uint16_t>> CLdownstream;
337
338     ConnectedBlobs(O, P, CLdownstream, nCols, nRows,
339         conn); // First loop through the image
340     SortAdjacencyList(
341         CLdownstream); // Sort all the adjacencylists and make
           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);
396     }

```

```

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, O, 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++;

```

```

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
467             // Sort the neighbors for easier checking
468             SoilMath::Sort::QuickSort<uchar>(nPixels, 8);
469
470             if (nPixels[0] == 0) {
471                 P[i] = 1;
472             } else {
473                 P[i] = 0;
474             }
475             delete[] nPixels;
476         } else {
477             throw Exception::PixelValueOutOfBoundsException();
478         }
479         i++;
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
    zero
513 for (uint32_t i = 0; i < nData; i++) {
514     if (O[i] != eroder.ProcessedImg.data[i]) {
515         P[i] = 1;
516     } else {
517         P[i] = 0;
518     }
519 }
520
521 // ProcessedImg = OriginalImg.clone() - eroder.
    ProcessedImg.clone();
522
523 SHOW_DEBUG_IMG(eroder.ProcessedImg, uchar, 255, "Eroded
    img Processed Image!",
524               true);
525 SHOW_DEBUG_IMG(ProcessedImg, uchar, 255, "GetEdgesEroding
    Processed Image!",
526               true);
527 }
528
529 /*! Create a BlobList subtracting each individual blob out
    of a Labelled image.
530 If the labelled image is empty build a new one with a BW
    image.
531 \param conn set the pixel connection eight or four
532 \param chain use the results from the previous operation
    default value = false;
533 */
534 void Segment::GetBlobList(bool chain, Connected conn) {
535     // Exception handling
536     CV_Assert(OriginalImg.depth() != sizeof(uchar));
537     EMPTY_CHECK(OriginalImg);
538
539     // If there isn't a labelledImg make one
540     if (MaxLabel < 1) {
541         LabelBlobs(chain, 5, conn);
542     }
543
544     // Make an empty BlobList
545     uint32_t nCols = OriginalImg.cols;
546     uint32_t nRows = OriginalImg.rows;
547     uint32_t nData = nCols * nRows;
548     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
599 // Loop through the BlobList and finalize it
600 uint8_t *LUT_filter = new uint8_t[MaxLabel + 1]{};
601 for (uint32_t i = 1; i <= MaxLabel; i++) {
602     LUT_filter[i] = 1;
603     BlobList[i].ROI.y = rectList[i].leftY;
604     BlobList[i].ROI.x = rectList[i].leftX;

```

```

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

```

```

651     }
652 }
653
654 /*!
655  * \brief Segment::SortAdjacencyList Sort the the sub
        vectors
656  * \param adj std::vector<std::vector<uint16_t>> &adj
657  */
658 void Segment::SortAdjacencyList(std::vector<std::vector<
        uint16_t>> &adj) {
659     uint32_t j = 0;
660     for_each(adj.begin(), adj.end(), [&](std::vector<uint16_t>
        &L) {
661         std::sort(L.begin(), L.end());
662         std::vector<uint16_t>::iterator it;
663         it = std::unique(L.begin(), L.end());
664         L.resize(std::distance(L.begin(), it));
665         if (L.size() > 1) {
666             for (std::vector<uint16_t>::iterator iter = L.begin();
        iter != L.end();
667                 ++iter) {
668                 if (*iter == j) {
669                     L.erase(iter);
670                     break;
671                 }
672             }
673         }
674         j++;
675     });
676 }
677
678 /*!
679  * \brief Segment::ConnectedBlobs Connect all the blobs and
        created the
680  * adjacency list
681  * \param O
682  * \param P
683  * \param adj
684  * \param nCols
685  * \param nRows
686  * \param conn
687  */
688 void Segment::ConnectedBlobs(uchar *O, uint16_t *P,
689                             std::vector<std::vector<
        uint16_t>> &adj,
690                             uint32_t nCols, uint32_t nRows,
        Connected conn) {
691     // Determine the size of the array for beginning and
        endrow and middle of a
692     // row
693     uint32_t noConn[3] = {static_cast<uint32_t>(conn),
694                          (static_cast<uint32_t>(conn) / 2),
695                          (static_cast<uint32_t>(conn) / 2) +
        1};
696     uint32_t lastConn[3] = {noConn[0] - 1, noConn[1] - 1,
        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 (O[0] == 0) {
714         P[0] = 0;
715     } else if (O[0] == 1) {
716         P[0] = 1;
717     } else {
718         throw Exception::PixelValueOutOfBounds();
719     }
720
721     // Walk through the toprow and determine if it's a new
722     // blob or it's connected
723     for (uint32_t i = 1; i < nCols; i++) {
724         if (O[i] == 0) {
725             P[i] = 0;
726         } else if (O[i] == 1) {
727             // If West is zero assume this is a new blob
728             if (P[i - 1] == 0) {
729                 P[i] = ++currentlbl;
730                 vector<uint16_t> cVector;
731                 cVector.push_back(currentlbl);
732                 adj.push_back(cVector);
733             } else { // set as previous blob
734                 P[i] = P[i - 1];
735             }
736         } else { // Value of of bounds
737             throw Exception::PixelValueOutOfBounds();
738         }
739     }
740
741     // walk through each pixel and determine if it's a new
742     // blob or it's connected
743     for (uint32_t i = OriginalImg.cols; i < nData; i++) {
744         if (O[i] == 0) { // Original pixel = 0
745             P[i] = 0;
746         } else if (O[i] == 1) {
747             // Get an array of Neighboring Pixels
748             uint16_t *nPixels = new uint16_t[nConn[nRow[i]]];
749             if (nRow[i] != 1) {
750                 nPixels[0] = P[i - 1];

```

```

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
778         /* If previous blobs belong to different connected
779            components set the
780            * current processed value to the lowest value and
781            remember that the
782            * other values should be the lowest value*/
783         if (P[i] != nPixels[lastConn[nRow[i]]]) {
784             for (int j = lastConn[nRow[i]]; j >= 0; --j) {
785                 if (nPixels[j] <= P[i]) {
786                     break;
787                 } else {
788                     adj[nPixels[j]].push_back(P[i]);
789                 }
790             }
791         }
792     }
793     delete[] nPixels;
794 } else {
795     throw Exception::PixelValueOutOfBoundsException();
796 }
797 delete[] nRow;
798 }
799
800 /*!
801 * \brief Segment::InvertAdjacencyList invert the
802        adjacencylist for upstream
803 * (unused)
804 * \param adj

```

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

E.0.6 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 += $$PWD/../../../build/install/
17
18 SOURCES += \
19     Segment.cpp \
20     MorphologicalFilter.cpp \
21     ImageProcessing.cpp \
22     Enhance.cpp \
23     Conversion.cpp
24
25 HEADERS += \
26     WrongKernelSizeException.h \
27     VisionDebug.h \
28     Vision.h \
29     Segment.h \
30     PixelValueOutOfBoundsException.h \
31     MorphologicalFilter.h \
32     ImageProcessing.h \
33     Enhance.h \
34     EmptyImageException.h \
35     ConversionNotSupportedException.h \
36     Conversion.h \
37     ChannelMismatchException.h
38
39 unix {
40     target.path = $PWD/../../../build/install
41     INSTALLS += target
42 }
43
44 #opencv
45 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui -
46         opencv_imgproc
47 INCLUDEPATH += /usr/local/include/opencv
48 INCLUDEPATH += /usr/local/include
49
50 #boost
51 DEFINES += BOOST_ALL_DYN_LINK
52 INCLUDEPATH += /usr/include/boost
53
54 unix:!macx: LIBS += -L$$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
   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  /*! Collection header of all the basic Vision headers*/
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
   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  // Debugging helper macros
10 #ifndef DEBUG
11 // #define DEBUG
12 #endif
13
14 #ifdef DEBUG
15 #include <limits>
16 #include <opencv2/highgui/highgui.hpp>
17 #include <vector>
18 #include "ImageProcessing.h"
19 #ifndef SHOW_DEBUG_IMG
20 #define SHOW_DEBUG_IMG(img, T1, maxVal, windowName, scale)
   \
21     Vision::ImageProcessing::ShowDebugImg<T1>(img, maxVal,
   windowName, scale)
22 #endif // !SHOW_DEBUG_IMG
23 #else
24 #ifndef SHOW_DEBUG_IMG
25 #define SHOW_DEBUG_IMG(img, T1, maxVal, windowName, scale)
26 #endif // !SHOW_DEBUG_IMG
27 #endif

```

```

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  /*! \class ChannelMismatchException
9  Exception class which is thrown when Extracted channel out
    of bounds exception
10 */
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 }

```

```

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  /*! \class ConversionNotSupportedException
9  Exception class which is thrown when an illegal conversion
    is requested.
10 */
11 #pragma once
12
13 #include <exception>
14 #include <string>
15
16 using namespace std;
17

```

```

18 namespace Vision {
19 namespace Exception {
20 class ConversionNotSupportedException : public std::
    exception {
21 public:
22     ConversionNotSupportedException(
23         string m = "Requested conversion is not supported!")
24         : msg(m){};
25     ~ConversionNotSupportedException() _GLIBCXX_USE_NOEXCEPT
        {};
26     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
        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   */
9
10 /*! \class EmptyImageException
11  * Exception class which is thrown when operations are about to
12  * start on a empty
13  * image.
14  */
15
16 #pragma once
17
18 #include <exception>
19 #include <string>
20
21 using namespace std;
22
23 namespace Vision {
24 namespace Exception {
25 class EmptyImageException : public std::exception {
26 public:
27     EmptyImageException(string m = "Empty Image!") : msg(m){};
28     ~EmptyImageException() _GLIBCXX_USE_NOEXCEPT{};
29     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
30     msg.c_str(); };
31
32 private:
33     string msg;
34 };
35 }
36 }

```

```

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  /*! \class PixelValueOutOfBoundsException
9  Exception class which is thrown when an unexpected pixel
   value has to be
10 computed
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 PixelValueOutOfBoundsException : public std::exception
   {
22 public:
23     PixelValueOutOfBoundsException(string m = "Current pixel
   value out of bounds!")
24         : msg(m){};
25     ~PixelValueOutOfBoundsException() _GLIBCXX_USE_NOEXCEPT{};
26     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
   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
   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 WrongKernelSizeException
9  Exception class which is thrown when a wrong kernel size is
   requested
10 */
11 #pragma once
12
13 #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
28 private:
29     string msg;
30 };
31 }
```



F. Analyzer Library

F.0.1 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 <spijker.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 "soilalyzerexception.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);
88     void GetParticlesFromBlobList(Vision::Segment::BlobList_t
89         &bloblist,
90         Image_t *snapshot,
91         Particle::ParticleVector_t &
92             partPopulation);
93     void CleanUpMatVector(std::vector<cv::Mat> &mv);
94     void CleanUpMatVector(Images_t *mv);
95     void GetFFD(Particle::ParticleVector_t &particalPopulation
96         );
97     void GetPrediction(Particle::ParticleVector_t &
98         particlePopulation);
99 };
100 }

```

```

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 "analyzer.h"
11
12 namespace SoilAnalyzer {
13
14     /*!
15     * \brief Analyzer::Analyzer
16     * \param snapshots
17     * \param results
18     * \param settings
19     */
20     Analyzer::Analyzer(Images_t *snapshots, Sample *results,
21         SoilSettings *settings = nullptr) {
22         this->Snapshots = snapshots;
23         this->Results = results;
24         if (settings == nullptr) {
25             Settings = new SoilSettings;
26         } else {
27             this->Settings = settings;
28         }
29         NeuralNet.LoadState(Settings->NNlocation);
30     }
31 }

```

```

30  /*!
31   * \brief Analyzer::PrepImages
32   */
33  void Analyzer::PrepImages() {
34      if (Snapshots == nullptr || Snapshots->size() == 0) {
35          throw Exception::SoilAnalyzerException(
36              EXCEPTION_NO_SNAPSHOTS,
37              EXCEPTION_NO_SNAPSHOTS_NR
38          );
39      }
40      std::vector<cv::Mat> intensityVector;
41      GetEnhancedInt(Snapshots, intensityVector);
42      std::vector<cv::Mat> BWVector;
43      GetBW(intensityVector, BWVector);
44      CleanUpMatVector(intensityVector);
45      GetParticles(BWVector, Snapshots, Results->
46          ParticlePopulation);
47      CleanUpMatVector(BWVector);
48      CleanUpMatVector(Snapshots);
49      Results->isPreparedForAnalysis = true;
50  }
51
52  void Analyzer::Analyse(Images_t *snapshots, Sample *results,
53      SoilSettings *settings) {
54      Snapshots = snapshots;
55      Results = results;
56      Settings = settings;
57      Analyse();
58  }
59
60  /*!
61   * \brief Analyzer::Analyse
62   */
63  void Analyzer::Analyse() {
64      CalcMaxProgress();
65      if (!Results->isPreparedForAnalysis && !Results->
66          IsLoadedFromDisk) {
67          PrepImages();
68      }
69      GetFFD(Results->ParticlePopulation);
70      if (PredictShape && Settings->PredictTheShape) {
71          GetPrediction(Results->ParticlePopulation);
72      }
73      Results->Angularity =
74          ucharStat_t(Results->GetAngularityVector()->data(),
75              Results->GetAngularityVector()->size(), 1,
76              7, 0, true);
77      emit on_progressUpdate(currentProgress++);
78      Results->Roundness =

```

```

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 /*!
99  * \brief Analyzer::CleanUpMatVector
100  * \param mv
101  */
102 void Analyzer::CleanUpMatVector(Images_t *mv) {
103     for_each(mv->begin(), mv->end(), [](Image_t &I) {
104         I.BackLight.release();
105         I.FrontLight.release();
106     });
107     mv->clear();
108 }
109
110 /*!
111  * \brief Analyzer::CalcMaxProgress
112  */
113 void Analyzer::CalcMaxProgress() {
114     // Static processing steps
115     MaxProgress += Snapshots->size() * 5;
116
117     // Optional processing steps
118     if (Settings->useBlur) {
119         MaxProgress += Snapshots->size();
120     }
121     if (Settings->useAdaptiveContrast) {
122         MaxProgress += Snapshots->size();
123     }
124     if (Settings->fillHoles) {
125         MaxProgress += Snapshots->size();
126     }
127     if (Settings->ignorePartialBorderParticles) {
128         MaxProgress += Snapshots->size();
129     }
130     if (Settings->morphFilterType != Vision::
131         MorphologicalFilter::NONE) {
132         MaxProgress += Snapshots->size();
133     }
134 }

```



```

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
168 /*!
169  * \brief Analyzer::GetEnhancedInt
170  * \param img
171  * \param intensity
172  */
173 void Analyzer::GetEnhancedInt(Mat &img, Mat &intensity) {
174     Vision::Conversion IntConvertor(img.clone());
175     IntConvertor.Convert(Vision::Conversion::RGB, Vision::
176         Conversion::Intensity);
177     emit on_progressUpdate(currentProgress++);
178     SHOW_DEBUG_IMG(IntConvertor.ProcessedImg, uchar, 255, "RGB
179         2 Int", false);
180
181     if (Settings->useBlur) {
182         Vision::Enhance IntBlur(IntConvertor.ProcessedImg.clone
183             ());
184         IntBlur.Blur(Settings->blurKernelSize);
185         emit on_progressUpdate(currentProgress++);
186         uint32_t HBK = Settings->blurKernelSize / 2;
187         uint32_t BK = Settings->blurKernelSize - 1;
188         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 /*!
226 * \brief Analyzer::GetBW
227 * \param images
228 * \param BWvector
229 */
230 void Analyzer::GetBW(std::vector<cv::Mat> &images,
231     std::vector<cv::Mat> &BWvector) {
232     for_each(images.begin(), images.end(), [&](cv::Mat &I) {
233         cv::Mat BW;
234         GetBW(I, BW);
235         BWvector.push_back(BW);
236     });
237 }
238 }
239
240

```

```

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);
264      } else {
265          BWborder = BWholes;
266      }
267
268      if (Settings->morphFilterType != Vision::
269          MorphologicalFilter::NONE) {
270          Vision::MorphologicalFilter Morph(BWborder.clone());
271          cv::Mat kernel = cv::Mat::zeros(Settings->filterMaskSize
272          ,
273          Settings->filterMaskSize
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);

```

```

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 /*!
319 * \brief Analyzer::GetParticlesFromBlobList
320 * \param bloblist
321 * \param snapshot
322 * \param edge
323 * \param partPopulation
324 */
325 void Analyzer::GetParticlesFromBlobList(
326     Vision::Segment::BlobList_t &bloblist, Image_t *snapshot
327     ,
328     Particle::ParticleVector_t &partPopulation) {
329     for_each(bloblist.begin(), bloblist.end(), [&](Vision::
330         Segment::Blob_t &B) {
331         Particle part;
332         part.ID = currentParticleID++;
333         part.PixelArea = B.Area;
334         Vision::Segment::getOrientented(B.Img, B.Centroid, B.
335             Theta,

```

```

330         part.Eccentricity);
331     cv::Mat RGB = Vision::Segment::CopyMat<uchar>(snapshot->
        FrontLight(B.ROI),
332                                                     B.Img,
                                                     CV_8UC3
                                                     ).clone
                                                     ();

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

```

```

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

```

F.0.2 Sample Class

```

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 "stdint.h"
11 #include <vector>
12 #include <string>
13 #include "Stats.h"
14 #include "psd.h"
15 #include "particle.h"
16 #include <fstream>
17 #include <boost/archive/binary_iarchive.hpp>
18 #include <boost/archive/binary_oarchive.hpp>
19 #include <boost/serialization/string.hpp>
20 #include <boost/serialization/version.hpp>
21 #include <boost/serialization/vector.hpp>
22 #include <boost/iostreams/filter/zlib.hpp>
23 #include <boost/iostreams/filtering_streambuf.hpp>
24 #include "zlib.h"
25 #include "soilanalyzertypes.h"
26
27 namespace SoilAnalyzer {
28 class Sample {
29 public:
30     Sample();
31
32     uint32_t ID; /*!< The sample ID*/
33     std::string Location; /*!< The Location where the sample
   was taken*/
34     double Longitude = 4.6296182999999947;
35     double Latitude = 51.8849149;
36     double Depth = 0;
37     std::string Date = "01-09-2015";
38     std::string Name; /*!< The sample name identifier*/
39
40     Particle::ParticleVector_t
41         ParticlePopulation; /*!< the individual particles of
   the sample*/
42
43     SoilMath::PSD PSD; /*!< The Particle Size Distribution*/
44     ucharStat_t Roundness;
45     ucharStat_t Angularity;
46     floatStat_t RI; /*!< The statistical Redness Index data*/
47
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 private:
72     Particle::PSDVector_t Diameter; /*!< The PSD raw data*/
73     bool PSDGathered = false; /*!< is the raw data
74                                 gathered*/
75     Particle::ClassVector_t RoundnessVec;
76     bool RoundnessGathered = false;
77     Particle::ClassVector_t AngularityVec;
78     bool AngularityGathered = false;
79     Particle::doubleVector_t CIELab_aVec;
80     bool CIELab_aGathered = false;
81     Particle::doubleVector_t CIELab_bVec;
82     bool CIELab_bGathered = false;
83
84 friend class boost::serialization::access;
85 template <class Archive>
86 void serialize(Archive &ar, const unsigned int version) {
87     ar &ID;
88     ar &Location;
89     ar &Name;
90     ar &ParticlePopulation;
91     ar &Diameter;
92     ar &RoundnessVec;
93     ar &AngularityVec;
94     ar &PSD;
95     ar &Roundness;
96     ar &Angularity;
97     ar &RI;
98     ar &isPreparedForAnalysis;
99     ar &isAnalysed;
100    ar &ChangesSinceLastSave;
101    ar &ParticleChangedStatePSD;
102    ar &ParticleChangedStateClass;
103    ar &ParticleChangedStateAngularity;
104    ar &ParticleChangedStateRoundness;
105    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 = 4.6296182999999947;
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 #include "sample.h"
11 #include "particle.h"
12
13 namespace SoilAnalyzer {
14 namespace io = boost::iostreams;
15
16 /*!
17  * \brief Sample::Sample
18  */
19 Sample::Sample() {}
20
21 /*!
22  * \brief Sample::Save
23  * \param filename
24  */
25 void Sample::Save(const std::string &filename) {

```

```

24     std::ofstream ofs(filename.c_str(), std::ios::out | std::
        ios::binary);
25     {
26         io::filtering_streambuf<io::output> out;
27
28         out.push(io::zlib_compressor(io::zlib::best_compression)
            );
29         out.push(ofs);
30         {
31             boost::archive::binary_oarchive oa(out);
32             oa << boost::serialization::make_nvp("Sample", *this);
33         }
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::
        ios::binary);
44     {
45         io::filtering_streambuf<io::input> in;
46
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.
            end(),
65             [&](Particle &P) { Diameter.push_back(P.
                GetSiDiameter()); });
66         PSDGathered = true;
67         ParticleChangedStatePSD = false;
68     }
69     return &Diameter;
70 }
71
72 Particle::ClassVector_t *Sample::GetAngularityVector() {
73     if (!AngularityGathered || ParticleChangedStateAngularity)
        {

```

```

74     AngularVec.clear();
75     for_each(ParticlePopulation.begin(), ParticlePopulation.
76         end(),
77         [&](Particle &P) { AngularVec.push_back(P.
78             GetAngularity()); });
79     AngularGathered = 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     return &RoundnessVec;
94 }
95
96 Particle::doubleVector_t *Sample::GetCIELab_aVector() {
97     if (!CIELab_aGathered || ColorChange) {
98         CIELab_aVec.clear();
99         for_each(ParticlePopulation.begin(), ParticlePopulation.
100             end(),
101             [&](Particle &P) { CIELab_aVec.push_back(P.
102                 getMeanLab().a); });
103         CIELab_aGathered = true;
104     }
105     return &CIELab_aVec;
106 }
107
108 Particle::doubleVector_t *Sample::GetCIELab_bVector() {
109     if (!CIELab_bGathered || ColorChange) {
110         CIELab_bVec.clear();
111         for_each(ParticlePopulation.begin(), ParticlePopulation.
112             end(),
113             [&](Particle &P) { CIELab_bVec.push_back(P.
114                 getMeanLab().b); });
115         CIELab_bGathered = true;
116     }
117     return &CIELab_bVec;
118 }
119 }

```

F.0.3 Particle Class

```

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  #pragma once
8
9  #include <opencv2/core.hpp>
10 #include <stdint.h>
11 #include <vector>
12 #include "SoilMath.h"
13 #include <fstream>
14 #include <boost/archive/binary_iarchive.hpp>
15 #include <boost/archive/binary_oarchive.hpp>
16 #include <boost/serialization/string.hpp>
17 #include <boost/serialization/version.hpp>
18 #include <boost/serialization/vector.hpp>
19 #include <boost/iostreams/filter/zlib.hpp>
20 #include <boost/iostreams/filtering_streambuf.hpp>
21 #include "zlib.h"
22 #include "soilanalyzereception.h"
23 #include "lab_t_archive.h"
24 #include "soilanalyzertypes.h"
25 #include "Vision.h"
26
27 namespace SoilAnalyzer {
28 class Particle {
29 public:
30     typedef std::vector<Particle>
31         ParticleVector_t; /*!< a vector consisting of
   individual particles*/
32     typedef std::vector<double> PSDVector_t; /*!< a vector
   used in the PSD*/
33     typedef std::vector<uint8_t>
34         ClassVector_t; /*!< a vector used in the
   classification histogram*/
35     typedef std::vector<float> floatVector_t;
36     typedef std::vector<double> doubleVector_t;
37
38     Particle();
39
40     uint32_t ID; /*!< The particle ID*/
41
42     cv::Mat BW; /*!< The binary image of the particle*/
43     cv::Mat Edge; /*!< The binary edge image of the particle*/
44     cv::Mat RGB; /*!< The RGB image of the particle*/
45
46     Point_t Centroid = {0, 0};
47     std::vector<Complex_t> FFDescriptors; /*!< The Fast
   Fourier Descriptors

```

```

48                                     describing the
49                                     contour in the
                                         Frequency domain
                                         */
50 Predict_t Classification;           /*!< The
    classification prediction*/
51 double SIPixelFactor = 0.0111915; /*!< The conversion
    factor from pixel to SI*/
52 uint32_t PixelArea = 0;             /*!< The total area of
    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
    ready for analysis*/
68 bool isAnalysed = false;           /*!< is the particle
    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 <spijker.jelle@gmail.com>, 2015
8  */
9
10 #include "particle.h"
11
12 namespace SoilAnalyzer {
13 namespace io = boost::iostreams;
14
15 Particle::Particle() {}
16
17 /*!
18 * \brief Particle::Save
19 * \param filename
20 */
21 void Particle::Save(const std::string &filename) {
22     std::ofstream ofs(filename.c_str(), std::ios::out | std::
23         ios::binary);
24     {
25         io::filtering_streambuf<io::output> out;
26
27         out.push(io::zlib_compressor(io::zlib::best_compression)
28             );
29         out.push(ofs);
30     }

```

```

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::
40         ios::binary);
41     {
42         io::filtering_streambuf<io::input> in;
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);
132     }

```

```
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::
            Conversion::RI);
131         floatStat_t RIstat((float *)convertor.ProcessedImg.data,
            LAB.rows, LAB.cols,
132                               (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::
        Conversion::CIE_lab);
141     LAB = convertor.ProcessedImg.clone();
142 }
143 }
```

F.0.4 Settings Class

```

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 <string>
11 #include <fstream>
12 #include <boost/archive/xml_iarchive.hpp>
13 #include <boost/archive/xml_oarchive.hpp>
14 #include <boost/serialization/version.hpp>
15 #include "../SoilVision/Vision.h"
16
17 namespace SoilAnalyzer {
18  /*!
19   * \brief The SoilSettings class
20   * \details A class with which the used settings can easily
      transfered to setup
21   * the Sample class in one go. This class is also used in
      the GUI. and has a
22   * possibility to saved to disk as a serialized object
23   */
24  class SoilSettings {
25  public:
26      SoilSettings();
27
28      /*!
29       * \brief SaveSettings a function to save the settings to
      disk
30       * \param filename a string with the filename
31       */
32      void SaveSettings(std::string filename);
33
34      /*!
35       * \brief LoadSettings a function to load the settings
      from disk
36       * \param filename a string with the filename
37       */
38      void LoadSettings(std::string filename);
39
40      bool useAdaptiveContrast =
41          false; /**< Should adaptive contrast stretch be used
      default is true*/
42      uint32_t adaptContrastKernelSize =
43          9; /**< The size of the adaptive contrast kernelsize*/
44      float adaptContrastKernelFactor = 1.; /**< the factor with
      which to multiply
45
      the effect of the
      adaptive

```

```

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

```

```

79  int Saturation_front = 64; /*!< cam saturation setting
    front light*/
80  int Saturation_proj = 0;    /*!< cam saturation setting
    projected light*/
81  int Hue_front = 0;          /*!< cam hue setting front
    light*/
82  int Hue_proj = -40;         /*!< cam hue setting projected
    light*/
83  int Gamma_front = 100;      /*!< cam gamma setting front
    light*/
84  int Gamma_proj = 200;       /*!< cam gamma setting
    projected light*/
85  int PowerLineFrequency_front =
86      1; /*!< cam powerline freq setting front light*/
87  int PowerLineFrequency_proj =
88      1; /*!< cam powerline freq setting
    projected light*/
89  int Sharpness_front = 12; /*!< cam sharpness setting front
    light*/
90  int Sharpness_proj = 25; /*!< cam sharpness setting
    projected light*/
91  int BackLightCompensation_front =
92      1; /*!< cam backlight compensation setting front light
    */
93  int BackLightCompensation_proj =
94      1; /*!< cam backlight compensation setting projected
    light*/
95  std::string NNlocation = "NeuralNet/Default.NN";
96  bool useCUDA = false; /*!< CUDA enabled*/
97  int selectedResolution = 0;
98  std::string SampleFolder = "~/Samples";
99  std::string SettingsFolder = "Settings";
100 std::string NNFolder = "NeuralNet";
101 std::string StandardSentTo = "j.spijker@ihcmerwede.com";
102 std::string StandardPrinter = "PDF printer";
103 uint32_t StandardNumberOfShots = 10;
104 bool PredictTheShape = true;
105 bool Revolution = true;
106 private:
107     friend class boost::serialization::access;
108     template <class Archive>
109     void serialize(Archive &ar, const unsigned int version) {
110         if (version >= 0) {
111             ar &BOOST_SERIALIZATION_NVP(useAdaptiveContrast);
112             ar &BOOST_SERIALIZATION_NVP(adaptContrastKernelFactor)
113                 ;
114             ar &BOOST_SERIALIZATION_NVP(adaptContrastKernelSize);
115             ar &BOOST_SERIALIZATION_NVP(useBlur);
116             ar &BOOST_SERIALIZATION_NVP(blurKernelSize);
117             ar &BOOST_SERIALIZATION_NVP(typeOfObjectsSegmented);
118             ar &BOOST_SERIALIZATION_NVP(ignorePartialBorderParticles);
119             ar &BOOST_SERIALIZATION_NVP(fillHoles);
120             ar &BOOST_SERIALIZATION_NVP(sigmaFactor);
121             ar &BOOST_SERIALIZATION_NVP(morphFilterType);
122             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(
        BackLightCompensation_front);
145     ar &BOOST_SERIALIZATION_NVP(BackLightCompensation_proj
        );
146     ar &BOOST_SERIALIZATION_NVP(NNlocation);
147     ar &BOOST_SERIALIZATION_NVP(useCUDA);
148     ar &BOOST_SERIALIZATION_NVP(selectedResolution);
149     ar &BOOST_SERIALIZATION_NVP(SampleFolder);
150     ar &BOOST_SERIALIZATION_NVP(SettingsFolder);
151     ar &BOOST_SERIALIZATION_NVP(NNFolder);
152     ar &BOOST_SERIALIZATION_NVP(StandardSentTo);
153     ar &BOOST_SERIALIZATION_NVP(StandardPrinter);
154     ar &BOOST_SERIALIZATION_NVP(StandardNumberOfShots);
155     ar &BOOST_SERIALIZATION_NVP(PredictTheShape);
156     ar &BOOST_SERIALIZATION_NVP(Revolution);
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
   * 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 "soilsettings.h"
9
10 namespace SoilAnalyzer {
11     SoilSettings::SoilSettings() {}

```

```
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 }
```

F.0.5 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 -
46         lboost_iostreams
47
48 #Zlib
49 LIBS += -L/usr/local/lib -lz
50 INCLUDEPATH += /usr/local/include
51
52 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
53 INCLUDEPATH += $$PWD/../../SoilMath
54 DEPENDPATH += $$PWD/../../SoilMath

```

```

54
55 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
    lSoilVision
56 INCLUDEPATH += $$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
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 "soilanalyzertypes.h"
16
17 namespace boost {
18 namespace serialization {
19 /*!
20  * \brief serialize Serialize the openCV mat to disk
21  */
22 template <class Archive>
23 inline void serialize(Archive &ar, SoilAnalyzer::Lab_t &P,
24     const unsigned int version __attribute__((unused))) {
25     ar &P.L;
26     ar &P.a;
27     ar &P.b;
28 }
29 }
30 }

```

```

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 #define EXCEPTION_PARTICLE_NOT_ANALYZED "Particle not
    analyzed Exception!"
11 #define EXCEPTION_PARTICLE_NOT_ANALYZED_NR 0
12 #define EXCEPTION_NO_SNAPSHOTS "No snapshots Exception!"
13 #define EXCEPTION_NO_SNAPSHOTS_NR 1

```

```

11 #define EXCEPTION_NO_IMAGES_PRESENT "No images to analyse
    exception!"
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 =
                EXCEPTION_PARTICLE_NOT_ANALYZED,
23                                 int n =
                EXCEPTION_PARTICLE_NOT_ANALYZED_NR
                ) : msg(m), nr(n) { }
24             ~SoilAnalyzerException() _GLIBCXX_USE_NOEXCEPT {}
25             const char *what() const _GLIBCXX_USE_NOEXCEPT {
                return msg.c_str(); }
26             const int *id() const _GLIBCXX_USE_NOEXCEPT { return &
                nr; }
27
28         private:
29             std::string msg;
30             int nr;
31     };
32 }
33 }

```

```

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

```

G. 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_<
            uchar>(255,255,255));
23         return dst;
24     }
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,
                destRGB.rows,
33                         destRGB.step, QImage::Format_RGB888);
34         } else {
35             dest = QImage((uchar *)src.data, src.cols, src.rows,
                src.step,
36                         QImage::Format_RGB888);
37             dest = dest.rgbSwapped();
38         }
39         return dest;
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  }
```

H. 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 <spijker.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 #include "soilalyzer.h"
18
19 namespace Ui {
20     class QParticleDisplay;
21 }
22
23 class QParticleDisplay : public QWidget
24 {
25     Q_OBJECT
26
27 public:
28     explicit QParticleDisplay(QWidget *parent = 0);
29     ~QParticleDisplay();
30     void SetSample(SoilAnalyzer::Sample *sample);
31     SoilAnalyzer::Particle *SelectedParticle;
32     void wheelEvent( QWheelEvent * event );
33     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 *
        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 <spijker.jelle@gmail.com>, 2015
8   */
9
10 #include "qparticledisplay.h"
11 #include "ui_qparticledisplay.h"
12
13 QParticleDisplay::QParticleDisplay(QWidget *parent)
14 : QWidget(parent), ui(new Ui::QParticleDisplay) {
15     ui->setupUi(this);
16     ui->widget->setBackgroundColor(QColor("white"));
17     ui->widget->setSlideSize(QSize(230, 230));
18     connect(ui->widget, SIGNAL(centerIndexChanged(int)), this,
19         SLOT(on_selectedParticleChangedWidget(int)));
20     connect(ui->horizontalSlider, SIGNAL(valueChanged(int)),
21         this,
22         SLOT(on_selectedParticleChangedSlider(int)));
23 }
24
25 QParticleDisplay::~QParticleDisplay() {
26     for (uint32_t i = 0; i < ui->widget->slideCount(); i++) {
27         ui->widget->removeSlide(0);
28     }
29     delete ui->widget;
30     delete ui;
31 }
32
33 void QParticleDisplay::setSelectedParticle(int newValue) {

```



```

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

```

```

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     );
101     ui->widget->removeSlide(ui->widget->centerIndex());
102     ui->horizontalSlider->setMaximum(this->Sample->
103         ParticlePopulation.size() - 1);
104     Sample->ParticleChangedStatePSD = true;
105     Sample->ParticleChangedStateAngularity = true;
106     Sample->ParticleChangedStateRoundness = true;
107     Sample->ChangesSinceLastSave = true;
108     Sample->ColorChange = true;
109     SelectedParticle = &Sample->ParticlePopulation[ui->widget
110         ->centerIndex()];
111     emit particleDeleted();
112 }
113
114 void QParticleDisplay::on_selectedParticleChangedWidget(int
115     value) {
116     if (!dontDoIt) {
117         dontDoIt = true;
118         ui->horizontalSlider->setValue(value);
119         SelectedParticle = &Sample->ParticlePopulation[ui->
120             widget->centerIndex()];
121         QString volume;
122         volume.sprintf("%+06.2f", SelectedParticle->
123             GetSiDiameter());
124         ui->label_Volume->setText(volume);
125         emit particleChanged(value);
126         emit shapeClassificationChanged(SelectedParticle->
127             Classification.Category);
128         dontDoIt = false;
129     }
130 }
131
132 void QParticleDisplay::on_selectedParticleChangedSlider(int
133     value) {
134     if (!dontDoIt) {
135         dontDoIt = true;
136         ui->widget->setCenterIndex(value);

```

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

I. 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 }
38

```

```
39 void QParticleSelector::setDisabled(bool value, int
    currentClass) {
40     for (auto b : btns) {
41         b->setDisabled(value);
42     }
43     if (currentClass > 18 || currentClass < 1) {
44         btns[0]->setChecked(true);
45     } else {
46         btns[currentClass - 1]->setChecked(true);
47     }
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 }
```



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



J. 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 multimediasupport
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/ -
46     lqcustomplot
47 INCLUDEPATH += $$PWD/../../qcustomplot
48 DEPENDPATH += $$PWD/../../qcustomplot
49
50 unix:!macx: LIBS += -L$$PWD/../../../build/install/ -
51     lSoilAnalyzer
52 INCLUDEPATH += $$PWD/../../SoilAnalyzer
53 DEPENDPATH += $$PWD/../../SoilAnalyzer
54
55 unix:!macx: LIBS += -L$$PWD/../../../build/install/ -
56     lSoilVision
57 INCLUDEPATH += $$PWD/../../SoilVision
58 DEPENDPATH += $$PWD/../../SoilVision
59
60 RESOURCES += \
61     qreportresources.qrc \
62     ../VSA/vsa_resources.qrc
63
64 #maps
65 Mapstarget.path += ${OUT_PWD}/Maps
66 Mapstarget.files += ${PWD}/Maps/*
67 INSTALLS += Mapstarget
68 bMapstarget.path += ${PWD}/../../../build/install/Maps
69 bMapstarget.files += ${PWD}/Maps/*
70 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 "soilalyzer.h"
16 #include "SoilMath.h"
17
18 #include <qcustomplot.h>
19 #include "../qcustomplot/examples/text-document-integration/
    qcpdocumentobject.h"
20
21 namespace Ui {
22     class QReportGenerator;
23 }
24
25 class QReportGenerator : public QMainWindow
26 {
27     Q_OBJECT
28
29 public:
30     QTextDocument *Report = nullptr;
31     SoilAnalyzer::Sample *Sample = nullptr;
32     SoilAnalyzer::SoilSettings *Settings = nullptr;
33     QCustomPlot *PSD = nullptr;
34     QCustomPlot *Roundness = nullptr;
35     QCustomPlot *Angularity = nullptr;
36
37     explicit QReportGenerator(QWidget *parent = 0,
        SoilAnalyzer::Sample *sample = nullptr, SoilAnalyzer::
        SoilSettings *settings = nullptr, QCustomPlot *psd =
        nullptr, QCustomPlot *roundness = nullptr, QCustomPlot
        *angularity = nullptr);
38     ~QReportGenerator();
39
40 private slots:
41     void on_locationImageDownloaded(QNetworkReply *reply);
42
43     void on_actionSave_triggered();
44
45     void on_actionExport_to_PDF_triggered();
46
47 private:
48     Ui::QReportGenerator *ui;
49     QCustomPlot *CIElabPlot = nullptr;
50
51     void getLocationMap(double &latitude, double &longtitude);
52     void SetupCIElabPlot();
53
54     QImage *mapLocation = nullptr;
55
56     QTextCursor rCurs;
57
58     // Layout formats
59     QTextBlockFormat TitleFormat;
60     QTextBlockFormat HeaderFormat;
61     QTextBlockFormat GeneralFormat;
62     QTextBlockFormat ImageGraphFormat;
63
64     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 *
5                                         sample,
6                                     SoilAnalyzer::
6                                         SoilSettings *settings
6                                         ,
7                                     QCustomPlot *psd,
7                                         QCustomPlot *roundness
7                                         ,
8                                     QCustomPlot *angularity)
9      : QMainWindow(parent), ui(new Ui::QReportGenerator) {
10     ui->setupUi(this);
11     if (settings == nullptr) {
12         settings = new SoilAnalyzer::SoilSettings;
13     }
14     this->Settings = settings;
15     if (sample == nullptr) {
16         sample = new SoilAnalyzer::Sample;
17     }
18     this->Sample = sample;
19
20     if (psd == nullptr) {
21         psd = new QCustomPlot;
22     }
23     this->PSD = psd;
24
25     if (roundness == nullptr) {
26         roundness = new QCustomPlot;
27     }
28     this->Roundness = roundness;
29
30     if (angularity == nullptr) {
31         angularity = new QCustomPlot;
32     }
33     this->Angularity = angularity;
34
35     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::
    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::
    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,
    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),
    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);
96     rCurs.movePosition(QTextCursor::NextCell);
97
98     rCurs.insertText("Location:", GFieldtxtFormat);
99     rCurs.movePosition(QTextCursor::NextCell);
100    rCurs.insertText(QString::number(Sample->Latitude),
101                    GtxtFormat);
101    rCurs.insertText(" ", GtxtFormat);
102    rCurs.insertText(QString::number(Sample->Longitude),
103                    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);
108    rCurs.insertText(" [m]", GtxtFormat);
109    rCurs.movePosition(QTextCursor::NextBlock);
110    rCurs.insertBlock();
111
112    // Insert the Google map
113    getLocationMap(Sample->Latitude, Sample->Longitude);
114
115    // Setup the QCustomplot handler
116    QCPDocumentObject *plotObjectHandler = new
117        QCPDocumentObject(this);
117    ui->textEdit->document()->documentLayout()->
118        registerHandler(
119        QCPDocumentObject::PlotTextFormat, plotObjectHandler);
119
120    // Setup the Textdata for the PSD
121    rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
122    rCurs.insertText("Particle Size Distribution");
123
124    rCurs.insertBlock(ImageGraphFormat);
125    QTextTable *PSDdescr = rCurs.insertTable(6, 2,
126        GeneralTextTableFormat);
126    rCurs = PSDdescr->cellAt(0, 0).firstCursorPosition();
127    rCurs.insertText("No of particles:", GFieldtxtFormat);
128    rCurs.movePosition(QTextCursor::NextCell);
129    rCurs.insertText(QString::number(Sample->PSD.n),
130                    GtxtFormat);
130    rCurs.movePosition(QTextCursor::NextCell);
131
132    rCurs.insertText("Mean: ", GFieldtxtFormat);
133    rCurs.movePosition(QTextCursor::NextCell);
134    rCurs.insertText(QString::number(Sample->PSD.Mean),
135                    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);
141     rCurs.movePosition(QTextCursor::NextCell);
142
143     rCurs.insertText("Maximum: ", GFieldtxtFormat);
144     rCurs.movePosition(QTextCursor::NextCell);
145     rCurs.insertText(QString::number(Sample->PSD.max),
146                     GtxtFormat);
147     rCurs.movePosition(QTextCursor::NextCell);
148
149     rCurs.insertText("Range: ", GFieldtxtFormat);
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);
156     rCurs.movePosition(QTextCursor::NextBlock);
157
158     // Setup the PSD
159     rCurs.insertBlock(ImageGraphFormat);
160     rCurs.insertText(QString(QChar::ObjectReplacementCharacter
161                             ),
162                     QCPDocumentObject::generatePlotFormat(PSD
163                             , 600, 350));
164
165     rCurs.insertBlock(ImageGraphFormat);
166     QTextTable *PSDdata = rCurs.insertTable(16, 3,
167                     GeneralTextTableFormat);
168     rCurs.insertText("Mesh Size [mm]", GFieldtxtFormat);
169     rCurs.movePosition(QTextCursor::NextCell);
170     rCurs.insertText("Cumulative [%]", GFieldtxtFormat);
171     rCurs.movePosition(QTextCursor::NextCell);
172     rCurs.insertText("Retained [-]", GFieldtxtFormat);
173     rCurs.movePosition(QTextCursor::NextCell);
174     rCurs.insertText("2", GFieldtxtFormat);
175     rCurs.movePosition(QTextCursor::NextCell);
176     rCurs.insertText(QString::number(Sample->PSD.CFD[14]),
177                     GtxtFormat);
178     rCurs.movePosition(QTextCursor::NextCell);
179     rCurs.insertText(QString::number(Sample->PSD.bins[14]),
180                     GtxtFormat);
181     rCurs.movePosition(QTextCursor::NextCell);
182     rCurs.insertText("1.4", GFieldtxtFormat);
183     rCurs.movePosition(QTextCursor::NextCell);
184     rCurs.insertText(QString::number(Sample->PSD.CFD[13]),
185                     GtxtFormat);
186     rCurs.movePosition(QTextCursor::NextCell);
187     rCurs.insertText(QString::number(Sample->PSD.bins[13]),
188                     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]),
        GtxtFormat);
185     rCurs.movePosition(QTextCursor::NextCell);
186     rCurs.insertText(QString::number(Sample->PSD.bins[12]),
        GtxtFormat);
187     rCurs.movePosition(QTextCursor::NextCell);
188     rCurs.insertText("0.71", GFieldtxtFormat);
189     rCurs.movePosition(QTextCursor::NextCell);
190     rCurs.insertText(QString::number(Sample->PSD.CFD[11]),
        GtxtFormat);
191     rCurs.movePosition(QTextCursor::NextCell);
192     rCurs.insertText(QString::number(Sample->PSD.bins[11]),
        GtxtFormat);
193     rCurs.movePosition(QTextCursor::NextCell);
194     rCurs.insertText("0.5", GFieldtxtFormat);
195     rCurs.movePosition(QTextCursor::NextCell);
196     rCurs.insertText(QString::number(Sample->PSD.CFD[10]),
        GtxtFormat);
197     rCurs.movePosition(QTextCursor::NextCell);
198     rCurs.insertText(QString::number(Sample->PSD.bins[10]),
        GtxtFormat);
199     rCurs.movePosition(QTextCursor::NextCell);
200     rCurs.insertText("0.355", GFieldtxtFormat);
201     rCurs.movePosition(QTextCursor::NextCell);
202     rCurs.insertText(QString::number(Sample->PSD.CFD[9]),
        GtxtFormat);
203     rCurs.movePosition(QTextCursor::NextCell);
204     rCurs.insertText(QString::number(Sample->PSD.bins[9]),
        GtxtFormat);
205     rCurs.movePosition(QTextCursor::NextCell);
206     rCurs.insertText("0.25", GFieldtxtFormat);
207     rCurs.movePosition(QTextCursor::NextCell);
208     rCurs.insertText(QString::number(Sample->PSD.CFD[8]),
        GtxtFormat);
209     rCurs.movePosition(QTextCursor::NextCell);
210     rCurs.insertText(QString::number(Sample->PSD.bins[8]),
        GtxtFormat);
211     rCurs.movePosition(QTextCursor::NextCell);
212     rCurs.insertText("0.18", GFieldtxtFormat);
213     rCurs.movePosition(QTextCursor::NextCell);
214     rCurs.insertText(QString::number(Sample->PSD.CFD[7]),
        GtxtFormat);
215     rCurs.movePosition(QTextCursor::NextCell);
216     rCurs.insertText(QString::number(Sample->PSD.bins[7]),
        GtxtFormat);
217     rCurs.movePosition(QTextCursor::NextCell);
218     rCurs.insertText("0.125", GFieldtxtFormat);
219     rCurs.movePosition(QTextCursor::NextCell);
220     rCurs.insertText(QString::number(Sample->PSD.CFD[6]),
        GtxtFormat);
221     rCurs.movePosition(QTextCursor::NextCell);
222     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,
        GeneralTextTableFormat);
267   rCurs = Rounddescr->cellAt(0, 0).firstCursorPosition();
268   rCurs.insertText("No of particles:", GFieldtxtFormat);
269   rCurs.movePosition(QTextCursor::NextCell);
270   rCurs.insertText(QString::number(Sample->Roundness.n),
        GtxtFormat);
271   rCurs.movePosition(QTextCursor::NextCell);
272
273   rCurs.insertText("Mean: ", GFieldtxtFormat);
274   rCurs.movePosition(QTextCursor::NextCell);
275   rCurs.insertText(QString::number(Sample->Roundness.Mean),
        GtxtFormat);
276   rCurs.movePosition(QTextCursor::NextCell);
277
278   rCurs.insertText("Minimum: ", GFieldtxtFormat);
279   rCurs.movePosition(QTextCursor::NextCell);
280   rCurs.insertText(QString::number(Sample->Roundness.min),
        GtxtFormat);
281   rCurs.movePosition(QTextCursor::NextCell);
282
283   rCurs.insertText("Maximum: ", GFieldtxtFormat);
284   rCurs.movePosition(QTextCursor::NextCell);
285   rCurs.insertText(QString::number(Sample->Roundness.max),
        GtxtFormat);
286   rCurs.movePosition(QTextCursor::NextCell);
287
288   rCurs.insertText("Range: ", GFieldtxtFormat);
289   rCurs.movePosition(QTextCursor::NextCell);
290   rCurs.insertText(QString::number(Sample->Roundness.Range),
        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),
        GtxtFormat);
296   rCurs.movePosition(QTextCursor::NextBlock);
297
298   // Setup the Roundness Graph
299   rCurs.insertBlock(ImageGraphFormat);
300   rCurs.insertText(QString(QChar::ObjectReplacementCharacter
        ),
301                   QCPDocumentObject::generatePlotFormat(
        Roundness, 600, 400));
302
303   // Setup the Textdata for the Roundness
304   rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
305   rCurs.insertText("Angularity Classification");
306
307   rCurs.insertBlock(ImageGraphFormat);
308   QTextTable *Angularitydescr = rCurs.insertTable(6, 2,
        GeneralTextTableFormat);
309   rCurs = Angularitydescr->cellAt(0, 0).firstCursorPosition
        ();
310   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);
329     rCurs.movePosition(QTextCursor::NextCell);
330     rCurs.insertText("Range: ", GFieldtxtFormat);
331     rCurs.movePosition(QTextCursor::NextCell);
332     rCurs.insertText(QString::number(Sample->Angularity.Range)
333         , GtxtFormat);
334     rCurs.movePosition(QTextCursor::NextCell);
335     rCurs.insertText("Standard deviation: ", GFieldtxtFormat);
336     rCurs.movePosition(QTextCursor::NextCell);
337     rCurs.insertText(QString::number(Sample->Angularity.Std),
338         GtxtFormat);
339     rCurs.movePosition(QTextCursor::NextBlock);
340     // Setup the Roundness Graph
341     rCurs.insertBlock(ImageGraphFormat);
342     rCurs.insertText(QString(QChar::ObjectReplacementCharacter
343         ),
344         QCPDocumentObject::generatePlotFormat(
345             Angularity, 600, 400));
346     // Setup the CIE La*b* graph
347     // Setup the Textdata for the Roundness
348     rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
349     rCurs.insertText("CIE La*b*");
350     SetupCIElabPlot();
351     rCurs.insertBlock(ImageGraphFormat);
352     rCurs.insertText(QString(QChar::ObjectReplacementCharacter
353         ),
354         QCPDocumentObject::generatePlotFormat(
355             CIElabPlot, 600, 400));
356 }

```

```

357 void QReportGenerator::getLocationMap(double &latitude,
    double &longitude) {
358     QNetworkAccessManager *manager = new QNetworkAccessManager
        ;
359     connect(manager, SIGNAL(finished(QNetworkReply *)), this,
360         SLOT(on_locationImageDownloaded(QNetworkReply *)))
        ;
361     QString locationURL("http://maps.googleapis.com/maps/api/
        staticmap?center=");
362     locationURL.append(QString::number(latitude));
363     locationURL.append(",");
364     locationURL.append(QString::number(longitude));
365     locationURL.append("&zoom=17&size=600x750&mapttype=hybrid&&
        format=png&visual_"
366         "refresh=true&markers=size:mid%7Ccolor
        :0xff0000%7Clabel:S%"
367         "7C");
368     locationURL.append(QString::number(latitude));
369     locationURL.append(",");
370     locationURL.append(QString::number(longitude));
371     qDebug() << locationURL;
372     QUrl googleStaticMapUrl(locationURL);
373     manager->get(QNetworkRequest(googleStaticMapUrl));
374 }
375
376 void QReportGenerator::on_locationImageDownloaded(
    QNetworkReply *reply) {
377     if (mapLocation == nullptr) {
378         mapLocation = new QImage;
379     }
380     mapLocation->loadFromData(reply->readAll());
381
382     if (mapLocation->isNull()) {
383         mapLocation->load("Maps/SampleLocation.png");
384     }
385
386     QTextBlock location = Report->findBlockByNumber(15);
387     QTextCursor insertMap(location);
388     insertMap.setBlockFormat(ImageGraphFormat);
389     insertMap.insertImage(*mapLocation);
390     insertMap.insertBlock();
391     insertMap.insertHtml("<br>");
392 }
393
394 QReportGenerator::~QReportGenerator()
395 {
396     delete CIElabPlot;
397     delete mapLocation;
398     delete ui;
399 }
400
401 void QReportGenerator::on_actionSave_triggered() {
402     QString fn = QFileDialog::getSaveFileName(
403         this, tr("Save Report"), QString::fromStdString(
            Settings->SampleFolder),
404         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         QPainter printer;
426         printer.setOutputFormat(QPrinter::PdfFormat);
427         printer.setOutputFileName(fn);
428         Report->print(&printer);
429     }
430
431 void QReportGenerator::SetupCIElabPlot() {
432     if (CIElabPlot == nullptr) {
433         CIElabPlot = new QCustomPlot();
434     }
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 }
```



K. Vision Soil Analyzer Program

K.0.1 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 multimediacore
13  TARGET = VSA
14  TEMPLATE = app
15  VERSION = 0.9.7
16
17  unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../build/install/
18
19  @
20  CONFIG(release, debug|release):DEFINES += QT_NO_DEBUG_OUTPUT
21  @
22
23  SOURCES += main.cpp \
24      vsamainwindow.cpp \
25      dialogsettings.cpp \
26      dialognn.cpp
27
28  HEADERS += vsamainwindow.h \
29      dialogsettings.h \
30      dialognn.h
```



```

31
32 FORMS      += vsmainwindow.ui \
33     dialogsettings.ui \
34     dialognn.ui
35
36 #opencv
37 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui -
38     opencv_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 += $$PWD/../../QReportGenerator
91
92 #NeuralNetFiles
93 NNtarget.path += ${OUT_PWD}/NeuralNet
94 NNtarget.files += ${PWD}/NeuralNet/*.NN
95 INSTALLS += NNtarget
96 bNNtarget.path += ${PWD}/../../build/install/NeuralNet
97 bNNtarget.files += ${PWD}/NeuralNet/*.NN
98 INSTALLS += bNNtarget
99
100 #SettingFiles
101 INItarget.path += ${OUT_PWD}/Settings
102 INItarget.files += ${PWD}/Settings/*.ini
103 INSTALLS += INItarget
104 bINItarget.path += $$PWD/../../build/install/Settings
105 bINItarget.files += $$PWD/Settings/*.ini
106 INSTALLS += bINItarget
107
108 #SoilSamples
109 IMGtarget.path += ${OUT_PWD}/SoilSamples
110 IMGtarget.files += ${PWD}/SoilSamples/*.VSA
111 INSTALLS += IMGtarget
112 bIMGtarget.path += ${PWD}/../../build/install/SoilSamples
113 bIMGtarget.files += ${PWD}/SoilSamples/*.VSA
114 INSTALLS += bIMGtarget
115
116 #Images
117 Imgtarget.path += ${OUT_PWD}/Images
118 Imgtarget.files += ${PWD}/Images/*
119 INSTALLS += Imgtarget
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 += Imgtarget
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 }
```

K.0.2 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 "soilalyzer.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

```

```

55     void on_actionAutomatic_Shape_Pediction_triggered(bool
        checked);
56
57     void on_reset_graph(QMouseEvent * e);
58
59     void on_actionReport_Generator_triggered();
60
61     void on_particleChanged(int newPart);
62
63     void on_PSD_contextMenuRequest(QPoint point);
64
65     void on_compare_against();
66
67     void on_restore_PSD();
68
69 private:
70     Ui::VSAMainWindow *ui;
71     DialogSettings *settingsWindow = nullptr;
72     DialogNN *nnWindow = nullptr;
73     QProgressBar *Progress;
74     QErrorMessage *CamError = nullptr;
75     QMessageBox *SaveMeMessage = nullptr;
76     QMessageBox *BacklightMessage = nullptr;
77     QMessageBox *ShakeItBabyMessage = nullptr;
78     QReportGenerator *ReportGenWindow = nullptr;
79
80     SoilAnalyzer::SoilSettings *Settings = nullptr;
81     Hardware::Microscope *Microscope = nullptr;
82     SoilAnalyzer::Sample *Sample = nullptr;
83     SoilAnalyzer::Analyzer *Analyzer = nullptr;
84     SoilAnalyzer::Analyzer::Images_t *Images = nullptr;
85     QCPBars *RoundnessBars = nullptr;
86     QCPBars *AngularityBars = nullptr;
87     std::vector<double> PSDTicks = {0.0, 0.038, 0.045, 0.063,
        0.075,
88                                     0.09, 0.125, 0.18, 0.25,
        0.355,
89                                     0.5, 0.71, 1.0, 1.4,
        2.0};
90     QVector<QString> RoundnessCat = {"High", "Medium", "Low"};
91     std::vector<double> RoundnessTicks = {1, 2, 3};
92     QVector<QString> AngularityCat = {"Very Angular", "Angular
        ", "Sub Angular",
93                                     "Sub Rounded", "Rounded
        ", "Well Rounded"};
94     std::vector<double> AngularityTicks = {1, 2, 3, 4, 5, 6};
95
96     bool ParticleDisplayerFilled = false;
97
98     void SetPSDgraph();
99     void setRoundnessHistogram();
100    void setAngularityHistogram();
101    void setAmpgraph();
102    void TakeSnapShots();
103 };
104

```

```

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
        sample?"));
16     SaveMeMessage->addButton(QMessageBox::Abort);
17     SaveMeMessage->addButton(QMessageBox::Close);
18
19     BacklightMessage = new QMessageBox(this);
20     BacklightMessage->setText("Turn off Frontlight! Turn on
        Backlight!");
21     ShakeItBabyMessage = new QMessageBox(this);
22
23     // Load the Microscope
24     Microscope = new Hardware::Microscope;
25     try {
26         Microscope->FindCam(Settings->defaultWebcam)->
            SelectedResolution =
27             &Microscope->FindCam(Settings->defaultWebcam)
28                 ->Resolutions[Settings->selectedResolution];
29     } catch (exception &e) {
30         Microscope->FindCam(0)->SelectedResolution =
31             &Microscope->FindCam(0)->Resolutions[Settings->
                selectedResolution];
32     }
33     try {
34         if (!Microscope->openCam(Settings->defaultWebcam)) {
35             int defaultCam = 0;
36             Microscope->openCam(defaultCam);
37             Settings->defaultWebcam = Microscope->SelectedCam->
                Name;
38         }
39     } catch (Hardware::Exception::MicroscopeException &e) {
40         if (*e.id() == EXCEPTION_OPENCAM_NR) {
41             try {
42                 int defaultCam = 0;
43                 Microscope->openCam(defaultCam);
44                 Settings->defaultWebcam = Microscope->SelectedCam->
                    Name;
45             } catch (Hardware::Exception::MicroscopeException &e)
46             {
47                 if (*e.id() == EXCEPTION_NOCAMS_NR) {
48                     CamError->showMessage(

```

```

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

```

```

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

```



```

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

```

```

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

```

```

232     ui->QPlot_Amp->yAxis->setScaleType(QCPAxis::stLogarithmic)
        ;
233     ui->QPlot_Amp->graph()->setPen(binPen);
234     ui->QPlot_Amp->graph()->setLineStyle(QCPGraph::lsLine);
235     ui->QPlot_Amp->graph()->setBrush(QBrush(QColor
        (50,50,200,40)));
236
237     // Connect the Particle display and Selector
238     connect(ui->widget_ParticleSelector, SIGNAL(valueChanged(
        int)), this,
239             SLOT(on_Classification_changed(int)));
240     connect(ui->widget_ParticleDisplay, SIGNAL(
        shapeClassificationChanged(int)),
241             ui->widget_ParticleSelector, SLOT(setValue(int)));
242     connect(ui->widget_ParticleDisplay, SIGNAL(particleDeleted
        ()), this,
243             SLOT(on_particle_deleted()));
244     connect(ui->widget_ParticleDisplay, SIGNAL(particleChanged
        (int)), this,
245             SLOT(on_particleChanged(int)));
246
247     // Setup the bar
248     ui->actionUseLearning->setChecked(Settings->
        PredictTheShape);
249
250     // Setup the widgets
251     ui->widget_ParticleSelector->setDisabled(true);
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
        .size() > 0) {
277         ui->widget_ParticleDisplay->SetSample(Sample);
278     }
279     SetPSDgraph();

```

```

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

```

```

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

```

```

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

```

```

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->
    useHDR) {
432     for (uint32_t i = 0; i < Settings->StandardNumberOfShots
        ; i++) {
433         SoilAnalyzer::Analyzer::Image_t newShot;
434         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
435         Microscope->GetHDRFrame(newShot.FrontLight, Settings->
            HDRframes);
436         Images->push_back(newShot);
437         QString ShakeMsg = "Shake it baby! ";
438         int number = Settings->StandardNumberOfShots - i - 1;
439         ShakeMsg.append(QString::number(number));
440         ShakeMsg.append(" to go!");
441         ShakeItBabyMessage->setText(ShakeMsg);
442         ShakeItBabyMessage->exec();
443     }
444 } else if (!Settings->useBacklightProjection && !Settings
    ->useHDR) {
445     for (uint32_t i = 0; i < Settings->StandardNumberOfShots
        ; i++) {
446         SoilAnalyzer::Analyzer::Image_t newShot;
447         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
448         Microscope->GetFrame(newShot.FrontLight);
449         Images->push_back(newShot);
450         QString ShakeMsg = "Shake it baby! ";
451         int number = Settings->StandardNumberOfShots - i - 1;
452         ShakeMsg.append(QString::number(number));
453         ShakeMsg.append(" to go!");
454         ShakeItBabyMessage->setText(ShakeMsg);
455         ShakeItBabyMessage->exec();
456     }
457 }
458 }
459
460 void VSAMainWindow::on_actionSaveSample_triggered() {
461     QString fn = QFileDialog::getSaveFileName(
462         this, tr("Save Sample"), QString::fromStdString(
            Settings->SampleFolder),
463         tr("Sample (*.VSA)"));
464     if (!fn.isEmpty()) {
465         if (!fn.contains(tr(".VSA"))) {
466             fn.append(tr(".VSA"));
467         }
468         Sample->IsLoadedFromDisk = true;
469         Sample->ChangesSinceLastSave = false;
470         Sample->Save(fn.toStdString());
471         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 void VSAMainWindow::on_actionCalibrate_triggered() {
522     cv::Mat calib;
523     Microscope->GetFrame(calib);
524     Analyzer->CalibrateSI(16.25, calib);
525 }
526
527 void VSAMainWindow::on_Classification_changed(int newValue)

```



```

    {
523     uint8_t *Cat =
524         &ui->widget_ParticleDisplay->SelectedParticle->
            Classification.Category;
525     if ((*Cat - 1) % 6 != (newValue - 1) % 6) {
526         Sample->ParticleChangedStateAngularity = true;
527     }
528     if ((*Cat - 1) / 6 != (newValue - 1) / 6) {
529         Sample->ParticleChangedStateRoundness = true;
530     }
531     ui->widget_ParticleDisplay->SelectedParticle->
        Classification.Category =
532         newValue;
533     ui->widget_ParticleDisplay->SelectedParticle->
        Classification.ManualSet = true;
534     Sample->ChangesSinceLastSave = true;
535     Analyzer->Analyse();
536     ui->widget_ParticleDisplay->next();
537 }
538
539 void VSAMainWindow::on_particle_deleted() { Analyzer->
    Analyse(); }
540
541 void VSAMainWindow::
    on_actionAutomatic_Shape_Pediction_triggered(bool checked
    ) {
542     Settings->PredictTheShape = checked;
543 }
544
545 void VSAMainWindow::on_reset_graph(QMouseEvent *e) {
546     ui->Qplot_PSD->xAxis->setRange(0, 10);
547     ui->Qplot_PSD->yAxis->setRange(0, 100);
548     ui->Qplot_PSD->setInteractions(QCP::iRangeDrag | QCP::
        iRangeZoom);
549     ui->Qplot_PSD->replot();
550 }
551
552 void VSAMainWindow::on_actionReport_Generator_triggered() {
553     if (ReportGenWindow == nullptr) {
554         ReportGenWindow =
555             new QReportGenerator(this, Sample, Settings, ui->
                Qplot_PSD,
556                                     ui->QPlot_Roudness, ui->
                    QPlot_Angularity);
557     }
558     ReportGenWindow->show();
559 }
560
561 void VSAMainWindow::on_PSD_contextMenuRequest(QPoint point)
    {
562     QMenu *menu = new QMenu(this);
563     menu->setAttribute(Qt::WA_DeleteOnClose);
564
565     menu->addAction("Compare against...", this, SLOT(
        on_compare_against()));
566     menu->addAction("Restore", this, SLOT(on_restore_PSD()));

```

```

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 Seperated 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
        Size Distribution");
622         ui->Qplot_PSD->graph(1)->addToLegend();
623         ui->Qplot_PSD->replot();
624     }
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 }
```

K.0.3 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 private slots:
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);
53

```

```
50 void on_checkBox_InvertEncoder_clicked(bool checked);
51
52 void on_checkBox_useCUDA_clicked(bool checked);
53
54 void on_horizontalSlider_BrightFront_valueChanged(int
    value);
55
56 void on_horizontalSlider_ContrastFront_valueChanged(int
    value);
57
58 void on_horizontalSlider_SaturationFront_valueChanged(int
    value);
59
60 void on_horizontalSlider_HueFront_valueChanged(int value);
61
62 void on_horizontalSlider_SharpnessFront_valueChanged(int
    value);
63
64 void on_horizontalSlider_BrightProj_valueChanged(int value
    );
65
66 void on_horizontalSlider_ContrastProj_valueChanged(int
    value);
67
68 void on_horizontalSlider_SaturationProj_valueChanged(int
    value);
69
70 void on_horizontalSlider_HueProj_valueChanged(int value);
71
72 void on_horizontalSlider_SharpnessProj_valueChanged(int
    value);
73
74 void on_cb_use_adaptContrast_3_clicked(bool checked);
75
76 void on_cb_useBlur_3_clicked(bool checked);
77
78 void on_rb_useDark_3_toggled(bool checked);
79
80 void on_cb_ignoreBorder_3_clicked(bool checked);
81
82 void on_cb_fillHoles_3_clicked(bool checked);
83
84 void on_sb_sigmaFactor_3_editingFinished();
85
86 void on_rb_useOpen_3_clicked(bool checked);
87
88 void on_rb_useClose_3_clicked(bool checked);
89
90 void on_rb_useErode_3_clicked(bool checked);
91
92 void on_rb_useDilate_3_clicked(bool checked);
93
94 void on_sb_morphMask_3_editingFinished();
95
96 void on_spinBox_MaxGen_editingFinished();
97
```

```

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

```

```

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

```



```

94  ui->cb_fillHoles_3->setChecked(Settings->fillHoles);
95  ui->cb_ignoreBorder_3->setChecked(Settings->
    ignorePartialBorderParticles);
96  ui->cb_useBlur_3->setChecked(Settings->useBlur);
97  if (!Settings->useBlur) {
98      ui->sb_blurMask_3->setEnabled(false);
99  }
100 ui->cb_use_adaptContrast_3->setChecked(Settings->
    useAdaptiveContrast);
101 if (!Settings->useAdaptiveContrast) {
102     ui->sb_adaptContrastFactor_3->setEnabled(false);
103     ui->sb_adaptContrKernel_3->setEnabled(false);
104 }
105 switch (Settings->typeOfObjectsSegmented) {
106 case Vision::Segment::Bright:
107     ui->rb_useDark_3->setChecked(false);
108     ui->rb_useLight_3->setChecked(true);
109     break;
110 case Vision::Segment::Dark:
111     ui->rb_useDark_3->setChecked(true);
112     ui->rb_useLight_3->setChecked(false);
113     break;
114 }
115 switch (Settings->morphFilterType) {
116 case Vision::MorphologicalFilter::CLOSE:
117     ui->rb_useClose_3->setChecked(true);
118     ui->rb_useDilate_3->setChecked(false);
119     ui->rb_useErode_3->setChecked(false);
120     ui->rb_useOpen_3->setChecked(false);
121     break;
122 case Vision::MorphologicalFilter::OPEN:
123     ui->rb_useClose_3->setChecked(false);
124     ui->rb_useDilate_3->setChecked(false);
125     ui->rb_useErode_3->setChecked(false);
126     ui->rb_useOpen_3->setChecked(true);
127     break;
128 case Vision::MorphologicalFilter::ERODE:
129     ui->rb_useClose_3->setChecked(false);
130     ui->rb_useDilate_3->setChecked(false);
131     ui->rb_useErode_3->setChecked(true);
132     ui->rb_useOpen_3->setChecked(false);
133     break;
134 case Vision::MorphologicalFilter::DILATE:
135     ui->rb_useClose_3->setChecked(false);
136     ui->rb_useDilate_3->setChecked(true);
137     ui->rb_useErode_3->setChecked(false);
138     ui->rb_useOpen_3->setChecked(false);
139     break;
140 }
141
142 ui->sb_adaptContrastFactor_3->setValue(Settings->
    adaptContrastKernelFactor);
143 ui->sb_adaptContrKernel_3->setValue(Settings->
    adaptContrastKernelSize);
144 ui->sb_blurMask_3->setValue(Settings->blurKernelSize);
145 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 ui->spinBox_OutputNeurons->setValue(NN->GetOutputNeurons()
    );
157 ui->spinBox_Elitisme->setValue(NN->ElitismeUsedByGA);
158 ui->spinBox_MaxGen->setValue(NN->MaxGenUsedByGA);
159 ui->spinBox_PopSize->setValue(NN->PopulationSizeUsedByGA);
160 ui->doubleSpinBox_endError->setValue(NN->EndErrorUsedByGA)
    ;
161 ui->doubleSpinBox_MutationRate->setValue(NN->
    MutationrateUsedByGA);
162 ui->doubleSpinBox_Beta->setValue(NN->GetBeta());
163 ui->doubleSpinBox_maxWeight->setValue(NN->
    MaxWeightUsedByGA);
164 ui->doubleSpinBox_MinWeight->setValue(NN->
    MinWeightUsedByGa);
165 ui->checkBox_PredictShape->setChecked(Settings->
    PredictTheShape);
166 ui->checkBox_revolt->setChecked(Settings->Revolution);
167
168 // Setup the preference tab
169 ui->lineEdit_NeuralNetFolder->setText(
170     QString::fromStdString(Settings->NNFolder));
171 ui->lineEdit_Printer->setText(
172     QString::fromStdString(Settings->StandardPrinter));
173 ui->lineEdit_Samplefolder->setText(
174     QString::fromStdString(Settings->SampleFolder));
175 ui->lineEdit_SendTo->setText(
176     (QString::fromStdString(Settings->StandardSentTo)));
177 ui->lineEdit_SettingFolder->setText(
178     QString::fromStdString(Settings->SettingsFolder));
179 ui->lineEdit__NeuralNet->setText(
180     QString::fromStdString(Settings->NNlocation));
181
182 if (openNN) {
183     ui->tabWidget->setCurrentIndex(3);
184 }
185 initfase = false;
186 }
187
188 DialogSettings::~DialogSettings() { delete ui; }
189
190 void DialogSettings::openTab(int newValue) {
191     if (newValue > ui->tabWidget->count()) {
192         ui->tabWidget->setCurrentIndex(newValue);
193     }
194 }

```

```

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

```

```

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

```

```
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::on_checkBox_useCUDA_clicked(bool
307     checked) {
308     Settings->useCUDA = checked;
309 }
310
311 void DialogSettings::
312     on_horizontalSlider_BrightFront_valueChanged(int value) {
313     if (!initfase) {
314         Settings->Brightness_front = value;
315     }
316 }
317
318 void DialogSettings::
319     on_horizontalSlider_ContrastFront_valueChanged(int value)
320     {
321     if (!initfase) {
322         Settings->Contrast_front = value;
323     }
324 }
325
326 void DialogSettings::
327     on_horizontalSlider_SaturationFront_valueChanged(
328     int value) {
329     if (!initfase) {
330         Settings->Saturation_front = value;
331     }
332 }
333
334 void DialogSettings::
335     on_horizontalSlider_HueFront_valueChanged(int value) {
336     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
339 void DialogSettings::
340     on_horizontalSlider_BrightProj_valueChanged(int value) {
341     if (!initfase) {
342         Settings->Brightness_proj = value;
343     }
344 }
345
346 void DialogSettings::
347     on_horizontalSlider_ContrastProj_valueChanged(int value)
348     {
349     if (!initfase) {
350         Settings->Contrast_proj = value;
351     }
352 }
353
354 void DialogSettings::
355     on_horizontalSlider_SaturationProj_valueChanged(
356         int value) {
357     if (!initfase) {
358         Settings->Saturation_proj = value;
359     }
360 }
361
362 void DialogSettings::
363     on_horizontalSlider_HueProj_valueChanged(int value) {
364     if (!initfase) {
365         Settings->Hue_proj = value;
366     }
367 }
368
369 void DialogSettings::
370     on_horizontalSlider_SharpnessProj_valueChanged(int value)
371     {
372     if (!initfase) {
373         Settings->Sharpness_proj = value;
374     }
375 }
376
377 void DialogSettings::on_cb_use_adaptContrast_3_clicked(bool
378     checked) {
379     Settings->useAdaptiveContrast = checked;
380     ui->sb_adaptContrastFactor_3->setDisabled(!checked);
381     ui->sb_adaptContrKernel_3->setDisabled(!checked);
382 }
```

```
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 void DialogSettings::on_cb_fillHoles_3_clicked(bool checked)
396 {
397     Settings->fillHoles = checked;
398 }
399
400 void DialogSettings::on_sb_sigmaFactor_3_editingFinished() {
401     Settings->sigmaFactor = ui->sb_sigmaFactor_3->value();
402 }
403
404 void DialogSettings::on_rb_useOpen_3_clicked(bool checked) {
405     Settings->morphFilterType = Vision::MorphologicalFilter::
406     OPEN;
407 }
408
409 void DialogSettings::on_rb_useClose_3_clicked(bool checked)
410 {
411     Settings->morphFilterType = Vision::MorphologicalFilter::
412     CLOSE;
413 }
414
415 void DialogSettings::on_rb_useErode_3_clicked(bool checked)
416 {
417     Settings->morphFilterType = Vision::MorphologicalFilter::
418     ERODE;
419 }
420
421 void DialogSettings::on_rb_useDilate_3_clicked(bool checked)
422 {
423     Settings->morphFilterType = Vision::MorphologicalFilter::
424     DILATE;
425 }
426
427 void DialogSettings::on_sb_morphMask_3_editingFinished() {
428     Settings->filterMaskSize = ui->sb_morphMask_3->value();
429 }
```

```

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::on_spinBox_Elitisme_editingFinished() {
435     NN->ElitismeUsedByGA = ui->spinBox_Elitisme->value();
436 }
437
438 void DialogSettings::
439     on_doubleSpinBox_endError_editingFinished() {
440     NN->EndErrorUsedByGA = ui->doubleSpinBox_endError->value()
441         ;
442 }
443
444 void DialogSettings::
445     on_doubleSpinBox_maxWeight_editingFinished() {
446     NN->MaxWeightUsedByGA = ui->doubleSpinBox_maxWeight->value
447         ();
448 }
449
450 void DialogSettings::
451     on_doubleSpinBox_MinWeight_editingFinished() {
452     NN->MinWeightUsedByGa = ui->doubleSpinBox_MinWeight->value
453         ();
454 }
455
456 void DialogSettings::on_doubleSpinBox_Beta_editingFinished()
457     {
458     NN->SetBeta(ui->doubleSpinBox_Beta->value());
459 }
460
461 void DialogSettings::on_spinBox_InputNeurons_editingFinished
462     () {
463     NN->SetInputNeurons(ui->spinBox_InputNeurons->value());
464 }
465
466 void DialogSettings::
467     on_spinBox_HiddenNeurons_editingFinished() {
468     NN->SetHiddenNeurons(ui->spinBox_HiddenNeurons->value());
469 }
470
471 void DialogSettings::
472     on_spinBox_OutputNeurons_editingFinished() {
473     NN->SetOutputNeurons(ui->spinBox_OutputNeurons->value());
474 }

```



```

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     QString fn = QFileDialog::getExistingDirectory(
488         this, tr("Select the NeuralNet Directory"),
489         QString::fromStdString(Settings->NNFolder),
490         QFileDialog::ShowDirsOnly | QFileDialog::
            DontResolveSymlinks);
491     if (!fn.isEmpty()) {
492         ui->lineEdit_NeuralNetFolder->setText(fn);
493         Settings->NNFolder = fn.toStdString();
494     }
495 }
496
497 void DialogSettings::on_pushButton_SelectNN_clicked() {
498     QString fn =
499         QFileDialog::getOpenFileName(this, tr("Select the
            standard Neural Net"),
500                                     QDir::homePath(), tr("
            NeuralNet (*.NN)"));
501     if (!fn.isEmpty()) {
502         if (!fn.contains(tr(".NN"))) {
503             fn.append(tr(".NN"));
504         }
505         Settings->NNlocation = fn.toStdString();
506         ui->lineEdit__NeuralNet->setText(fn);
507     }
508 }
509
510 void DialogSettings::on_spinBox_NoShots_editingFinished() {

```

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

K.0.4 Dialog Neural Network Class

```

1  #ifndef DIALOGNN_H
2  #define DIALOGNN_H
3
4  #include <QDialog>
5  #include "SoilMath.h"
6  #include "soilalyzer.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 *
        neuralnet = nullptr, SoilAnalyzer::SoilSettings *
        settings = nullptr, DialogSettings *settingWindow =
        nullptr);
21     ~DialogNN();
22
23 private slots:
24
25     void on_pushButton_Settings_clicked();
26
27     void on_learnErrorUpdate(double newError);
28
29     void on_pushButton_SelectSamples_clicked();
30
31     void on_pushButton_Learn_clicked();
32
33     void on_pushButton_SaveNN_clicked();
34
35     void on_pushButton_OpenNN_clicked();
36
37     void on_actionAbort_triggered();
38
39 private:
40     Ui::DialogNN *ui;
41     DialogSettings *SettingsWindow = nullptr;
42     SoilMath::NN *NeuralNet = nullptr;
43     SoilAnalyzer::SoilSettings *Settings = nullptr;
44
45     void setupErrorGraph();
46     void makeLearnVectors(InputLearnVector_t &input,
        OutputLearnVector_t &output);
47
48     QVector<double> currentError;
49     QVector<double> errorTicks;
50     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         ));
34     ui->widget_NNError->plotLayout()->insertRow(0);
35     ui->widget_NNError->plotLayout()->addElement(0, 0,
36         widget_NNErrorTitle);
37
38     setupErrorGraph();
39
40     // Connect the NN learn error
41     connect(NeuralNet, SIGNAL(learnErrorUpdate(double)), this,
42         SLOT(on_learnErrorUpdate(double)));
43 }
44
45 DialogNN::~DialogNN() { delete ui; }
46
47 void DialogNN::on_pushButton_Settings_clicked() {
48     SettingsWindow->openTab(2);
49     SettingsWindow->show();
50     setupErrorGraph();
51 }

```

```

49
50 void DialogNN::on_learnErrorUpdate(double newError) {
51     ui->widget_NNError->graph(0)->addData(currentGeneration,
52         newError);
53     currentGeneration += 1;
54     ui->widget_NNError->yAxis->rescale();
55     //ui->widget_NNError->yAxis->setRange(0, 20);
56     ui->widget_NNError->replot();
57 }
58 void DialogNN::setErrorGraph() {
59     errorTicks.clear();
60     for (uint32_t i = 0; i < NeuralNet->MaxGenUsedByGA; i++) {
61         errorTicks.push_back(i);
62     }
63     ui->widget_NNError->xAxis->setRange(0, NeuralNet->
64         MaxGenUsedByGA);
65     QVector<double> endErrorValue(2, NeuralNet->
66         EndErrorUsedByGA);
67     QVector<double> endErrorKey(2, 0);
68     endErrorKey[1] = NeuralNet->MaxGenUsedByGA;
69     ui->widget_NNError->graph(1)->setData(endErrorKey,
70         endErrorValue);
71     ui->widget_NNError->xAxis->setAutoTicks(false);
72     ui->widget_NNError->xAxis->setTickVector(errorTicks);
73     ui->widget_NNError->xAxis->setTickLabels(false);
74     //ui->widget_NNError->yAxis->setScaleType(QCPAxis::
75         stLogarithmic);
76     ui->widget_NNError->replot();
77 }
78 void DialogNN::on_pushButton_SelectSamples_clicked() {
79     fn = QFileDialog::getOpenFileNames(
80         this, tr("Open Samples"), QString::fromStdString(
81             Settings->SampleFolder),
82         tr("Samples (*.VSA)"));
83     for_each(fn.begin(), fn.end(), [](QString &f) {
84         if (!f.contains(tr(".VSA"))) {
85             f.append(tr(".VSA"));
86         }
87     });
88 }
89 void DialogNN::on_pushButton_Learn_clicked() {
90     if (fn.size() < 1) {
91         return;
92     }
93     InputLearnVector_t InputVec;
94     OutputLearnVector_t OutputVec;
95     makeLearnVectors(InputVec, OutputVec);
96     NeuralNet->Learn(InputVec, OutputVec, NeuralNet->
97         GetInputNeurons());
98     setErrorGraph();
99 }
100 void DialogNN::makeLearnVectors(InputLearnVector_t &input,

```

```

98                                     OutputLearnVector_t &output)
99                                     {
100     for (uint32_t i = 0; i < fn.size(); i++) {
101         SoilAnalyzer::Sample sample;
102         sample.Load(fn[i].toString());
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 void DialogNN::on_pushButton_SaveNN_clicked() {
125     QString fn = QFileDialog::getSaveFileName(
126         this, tr("Save NeuralNet"), QString::fromStdString(
127             Settings->NNFolder),
128         tr("NeuralNet (*.NN)"));
129     if (!fn.isEmpty()) {
130         if (!fn.contains(tr(".NN"))) {
131             fn.append(tr(".NN"));
132         }
133         NeuralNet->SaveState(fn.toString());
134     }
135 }
136
137 void DialogNN::on_pushButton_OpenNN_clicked() {
138     QString fn = QFileDialog::getOpenFileName(
139         this, tr("Open NeuralNet"),
140         QString::fromStdString(Settings->SampleFolder), tr(
141             "NeuralNet (*.NN)"));
142     if (!fn.isEmpty()) {
143         if (!fn.contains(tr(".NN"))) {
144             fn.append(tr(".NN"));
145         }
146         if (NeuralNet != nullptr) {
147             delete NeuralNet;
148         }
149         NeuralNet->LoadState(fn.toString());
150         connect(NeuralNet, SIGNAL(learnErrorUpdate(double)),
151             this,
152             SLOT(on_learnErrorUpdate(double)));
153     }
154 }

```

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