Vision Soil Analyzer

Product design of a vision based soil analyzer

Jelle Spijker

Contents

II	Realization

III	Verification

IV	Addenda

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 refereed 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 an 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

2.1 Global Input-Proces-Output

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer 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, consectetuer 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 consectetuer.

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

4. Manuals

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

II
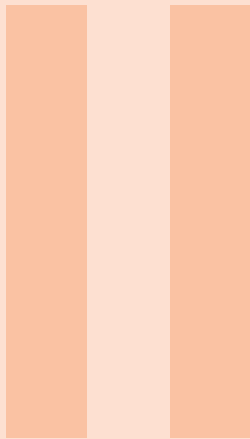
Realization

7. Technical Realization

8. Vision realization

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

8.1 Image acquisition

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

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

Acquisition strategies

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

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

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 given at the University of Applied Sciences Rotterdam will work on this challenge in the RDM Campus. This minor starts in September 2015. Their product will serve as input for the second prototype iteration. The assignment given to them is described in appendix A.

Acquisition

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

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

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

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:

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

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.

8.3 Feature extraction

In order to tell something about the individual particles, they first have to be identified and separated from the background. If there are only bright field images in the image vector, the segmentation operations are performed on these otherwise the back lit images are used.

Segmentation

The images are segmented by calculating a threshold value. This value is determined by using the Otsu threshold. Xu et al. 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 using the algorithm 8.3. Let $\mathbf{H}$ be a histogram representing the count of pixel values in the intensity matrix $\mathbf{I}$

8.3.1 CIE La*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

Algorithm 1 Otsu's threshold

1: **function** GETTHRESHOLDLEVEL(**H**)
2:     $\sigma_r = 0$
3:     $\sigma_l = 0$
4:     $\bar{l} = 0$
5:     $\bar{r} = 0$
6:     $T_p = 0$
7:     $T_c = \frac{1}{n \times m} \sum\limits_{i=1}^{n \times m} \mathbf{H}_i \times i$
8:     **while** $T_p! = T_c$ **do**
9:         $\sigma = 1$
10:     **end while**
11:     **return** δ
12: **end function**
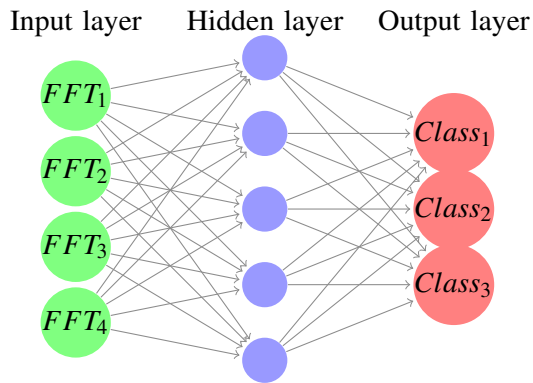
This is an example of examples.

Verification

9. Presenting Information

9.1 Table

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

Table 9.1: Table caption

9.2 Figure
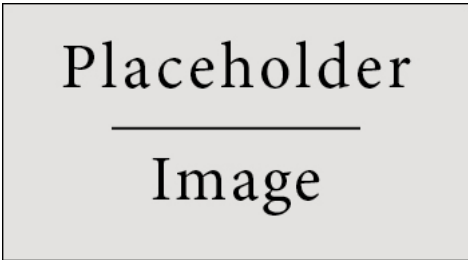
Figure 9.1: Figure caption

IV

Addenda

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).
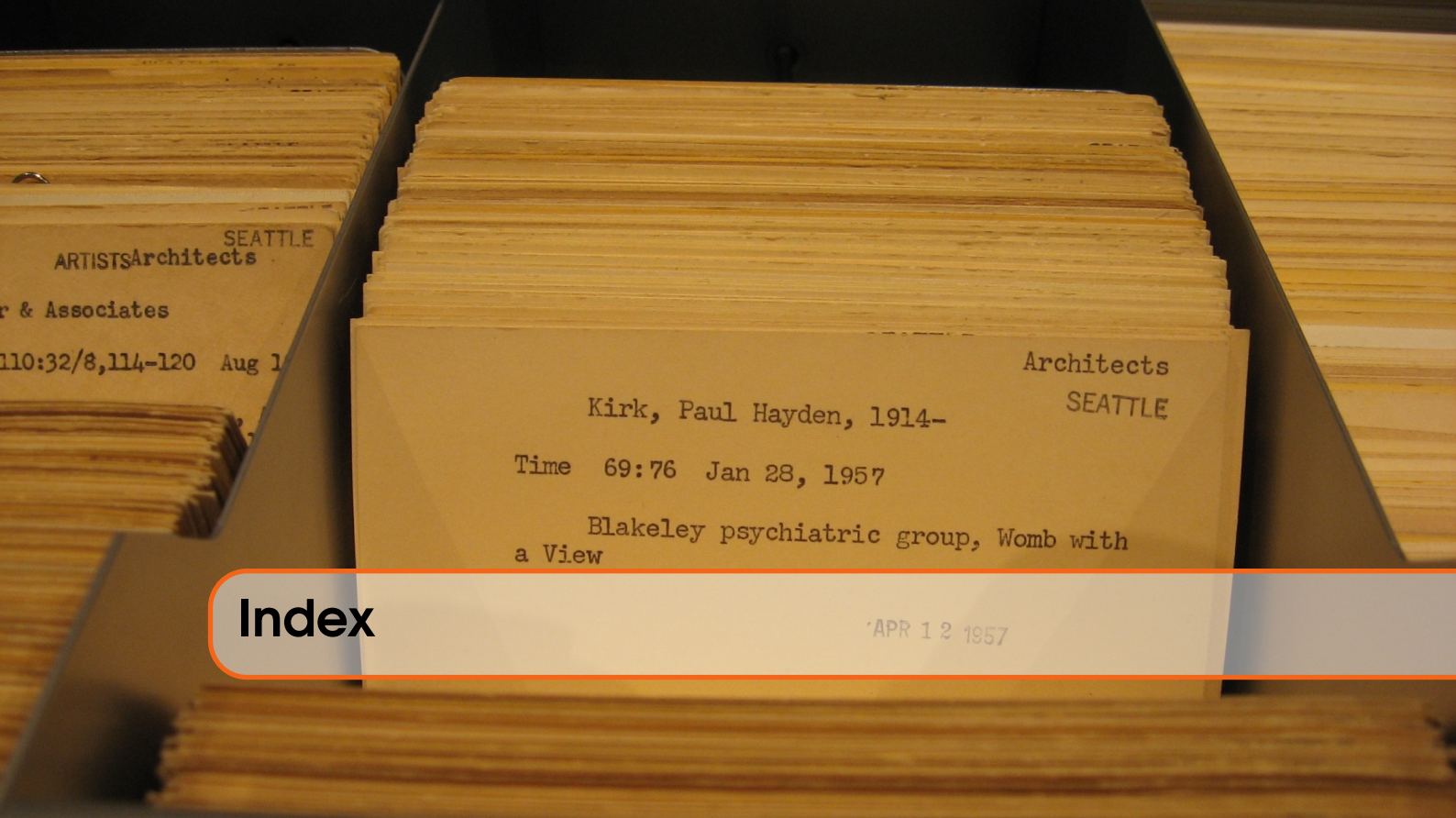
Index

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

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 optimzation
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>
```

```cpp
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
          elite bastard*/
42    float EndError = 0.001f;     /**< acceptable error between
          last itteration*/
43    bool Revolution = true;
44
45    /*!
46   * \brief GA Standard constructor
47   */
48    GA();
49
50    /*!
51     * \brief GA Construction with a Neural Network
          initializers
52     * \param nnfunction the Neural Network prediction
          function which results will
53     * be optimized
54     * \param inputneurons the number of input neurons in the
          Neural Network don't
55     * count the bias
56     * \param hiddenneurons the number of hidden neurons in
          the Neural Network
57     * don't count the bias
58     * \param outputneurons the number of output neurons in
          the Neural Network
59     */
60    GA(NNfunctionType nnfunction, uint32_t inputneurons,
        uint32_t hiddenneurons,
61      uint32_t outputneurons);
62
63    /*!
64     * \brief GA standard de constructor
65     */
66    ~GA();
67
68    /*!
69     * \brief Evolve Darwin would be proud!!! This function
          creates a population
70     * and itterates
71     * through the generation till the maximum number off
          itterations has been
72     * reached of the
73     * 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
              battle to the death
157     * commences
158     * The fittest population members have the best chance of
              survival. Death is
159     * instigated
160     * with a random uniform distibution. The elite members
              don't partake in this
161     * desctruction
162     * The ELITISME rate indicate how many top tier members
              survive this
163     * catastrophic event.
164     * \param inputValues a InputLearnVector_t with a
              reference to the inputvalues
165     * \param totalFitness a reference to the total population
              fitness
166     * \return
167     */
168    bool SurvivalOfTheFittest(Population_t &pop, float &
          totalFitness);
169
170    /*!
171     * \brief PopMemberSort a private function where the
              members are sorted
172     * according to
173     * there fitness ranking
174     * \param i left hand population member
175     * \param j right hand population member
176     * \return true if the left member is closser to the goal
              as the right member.
177     */
178    static bool PopMemberSort(PopMember_t i, PopMember_t j) {
179      return (i.Fitness < j.Fitness);
180    }
181
182    /*!
183     * \brief Conversion of the value of type T to Genome_t
184     * \details Usage: Use <tt>ConvertToGenome<Type>(type,
              range)</tt>
185     * \param value The current value wich should be converted
               to a Genome_t
186     * \param range the range in which the value should fall,
              this is to have a
187     * Genome_t
188     * which utilizes the complete range 0000...n till 1111...
              n
189     */
190    template <typename T>
191    inline Genome_t ConvertToGenome(T value, std::pair<T, T>
          range) {
192      uint32_t intVal = static_cast<uint32_t>(
193          (UINT32_MAX * (range.first + value)) / (range.second
                  - range.first));
194      Genome_t retVal(intVal);
195      return retVal;
```

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

```
 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 "GA.h"
 9
10 namespace SoilMath {
11 GA::GA() {}
12
13 GA::GA(NNfunctionType nnfunction, uint32_t inputneurons,
       uint32_t hiddenneurons,
14         uint32_t outputneurons) {
15   this->NNfuction = nnfunction;
16   this->inputneurons = inputneurons;
17   this->hiddenneurons = hiddenneurons;
18   this->outputneurons = outputneurons;
19 }
20
21 GA::~GA() {}
22
23 void GA::Evolve(const InputLearnVector_t &inputValues,
       Weight_t &weights,
24                 MinMaxWeight_t rangeweights,
                       OutputLearnVector_t &goal,
25                 uint32_t maxGenerations, uint32_t popSize) {
26   minOptim = goal[0].OutputNeurons.size();
27   minOptim = -minOptim;
28   maxOptim = 2 * goal[0].OutputNeurons.size();
```

```cpp
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                                              rangeweights.
                                                  second);
59
60    for (uint32_t i = 0; i < popSize; i++) {
61      PopMember_t I;
62      for (uint32_t j = 0; j < weights.size(); j++) {
63        I.weights.push_back(dis(gen));
64        I.weightsGen.push_back(
65            ConvertToGenome<float>(I.weights[j], rangeweights)
                );
66      }
67      pop.push_back(I);
68    }
69    return pop;
70  }
71
72  void GA::CrossOver(Population_t &pop) {
73    Population_t newPop; // create a new population
74    PopMember_t newPopMembers[2];
75    SplitGenome_t Split[2];
76
77    for (uint32_t i = 0; i < pop.size(); i += 2) {
78
79      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,
                 CROSSOVER));
83        Split[0].second = std::bitset<GENE_MAX - CROSSOVER>(
84            pop[i].weightsGen[j].to_string().substr(CROSSOVER,
85                                               GENE_MAX -

                                                 CROSSOVER
                                                 ));
86
87        // Split B
88        Split[1].first = std::bitset<CROSSOVER>(
89            pop[i + 1].weightsGen[j].to_string().substr(0,
                 CROSSOVER));
90        Split[1].second = std::bitset<GENE_MAX - CROSSOVER>(
91            pop[i + 1].weightsGen[j].to_string().substr(
                 CROSSOVER,
92                                               GENE_MAX
                                                   -
                                                 CROSSOVER
                                                 ));
93
94        // Mate A and B to AB and BA
95        newPopMembers[0].weightsGen.push_back(
96            Genome_t(Split[0].first.to_string() + Split[1].
                 second.to_string()));
97        newPopMembers[1].weightsGen.push_back(
98            Genome_t(Split[1].first.to_string() + Split[0].
                 second.to_string()));
99      }
100     newPop.push_back(newPopMembers[0]);
101     newPop.push_back(newPopMembers[1]);
102     newPopMembers[0].weightsGen.clear();
103     newPopMembers[1].weightsGen.clear();
104   }
105
106   // Allow the top tiers population partners to mate again
107   uint32_t halfN = pop.size() / 2;
108   for (uint32_t i = 0; i < halfN; i++) {
109     for (uint32_t j = 0; j < pop[i].weights.size(); j++) {
110       Split[0].first = std::bitset<CROSSOVER>(
111           pop[i].weightsGen[j].to_string().substr(0,
                CROSSOVER));
112       Split[0].second = std::bitset<GENE_MAX - CROSSOVER>(
113           pop[i].weightsGen[j].to_string().substr(CROSSOVER,
114                                             GENE_MAX -

                                               CROSSOVER
                                               ));
115
116       Split[1].first = std::bitset<CROSSOVER>(
117           pop[i + 2].weightsGen[j].to_string().substr(0,
                CROSSOVER));
118       Split[1].second = std::bitset<GENE_MAX - CROSSOVER>(
119           pop[i + 2].weightsGen[j].to_string().substr(
```

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

```
                 * hiddenneurons),
162                     P.weights.end());
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        // See issue #85
168        bool allGood = true;
169        float fitness = 0.0;
170        for (uint32_t k = 0; k < results.OutputNeurons.size();
              k++) {
171           bool resultSign = std::signbit(results.OutputNeurons
              [k]);
172           bool goalSign = std::signbit(goal[j].OutputNeurons[k
              ]);
173           fitness += results.OutputNeurons[k] / goal[j].
              OutputNeurons[k];
174           if (resultSign != goalSign) {
175              allGood = false;
176           }
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;
```

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

C.0.2　Fast Fourier Transform Class

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

#pragma once

#include <vector>
#include <complex>
#include <cmath>
#include <valarray>
#include <array>
#include <deque>
#include <queue>
#include <iterator>
#include <algorithm>
#include <stdint.h>
#include <opencv2/core.hpp>
#include "SoilMathTypes.h"
#include "MathException.h"

namespace SoilMath {
/*!
 * \brief Fast Fourier Transform class
 * \details Use this class to transform a black and white
     blob presented as a
 * cv::Mat with values 0 or 1 to a vector of complex values
     representing the Fourier
 * Descriptors.
 */
class FFT {
public:
  /*!
 * \brief Standard constructor
 */
  FFT();

  /*!
  * \brief Standard deconstructor
  */
  ~FFT();

  /*!
   * \brief Transforming the img to the frequency domain and
        returning the
   * Fourier Descriptors
   * \param img contour in the form of a cv::Mat type
       CV_8UC1. Which should
   * 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} | 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      * \brief Neighbors a private function returning the
            neighboring pixels which
83      * belong to a contour
84      * \param O uchar pointer to the data
85      * \param pixel current counter
86      * \param columns total number of columns
87      * \param rows total number of rows
88      * \return
89      */
```

```cpp
 90    iContour_t Neighbors(uchar *O, int pixel, uint32_t columns
          , uint32_t rows);

 92    /*!
 93     * \brief fft a private function calculating the Fast
             Fourier Transform
 94     * let \f$ m \f$ be an integer and let \f$ N=2^m \f$ also
 95     * \f$ CA=[x_0,\ldots,x_{N-1}] \f$ is an \f$ N \f$
             dimensional complex vector
 96     * let \f$ \omega=\exp({-2\pi i\over N}) \f$
 97     * then \f$ c_k={\frac{1}{N}}\sum_{j=0}^{j=N-1}CA_j\omega
            ^{jk} \f$
 98     * \param CA a \f$ CA=[x_0,\ldots,x_{N-1}] \f$ is an \f$ N
             \f$ dimensional
 99     * complex vector
100     */
101    void fft(ComplexArray_t &CA);

103    /*!
104     * \brief ifft
105     * \param CA
106     */
107    void ifft(ComplexArray_t &CA);
108  };
109  }
```

```cpp
 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   */

 8  #include "FFT.h"

10  namespace SoilMath {
11  FFT::FFT() {}

13  FFT::~FFT() {}

15  ComplexVect_t FFT::GetDescriptors(const cv::Mat &img) {
16    if (!fftDescriptors.empty()) {
17      return fftDescriptors;
18    }

20    complexcontour = Contour2Complex(img, img.cols / 2, img.
         rows / 2);

22    // Supplement the vector of complex numbers so that N = 2^
         m
23    uint32_t N = complexcontour.size();
24    double logN = log(static_cast<double>(N)) / log(2.0);
25    if (floor(logN) != logN) {
26      // 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 *O, 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 (O[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 *O = 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 (O[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(
            EXCEPTION_NO_CONTOUR_FOUND,
84                                        EXCEPTION_NO_CONTOUR_FOUND_NR
                                          );
85      } // Exception handling
86
87      uint32_t prev = -1;
88
89      // Extend path on queue
90      for (uint32_t i = sCont.front().front(); i < pEnd;) {
91        iContour_t nBors =
92            Neighbors(O, i, img.cols, img.rows); // find
                neighboring pixels
93        std::deque<uint32_t> cQ = sCont.front(); // store first
            queue;
94        sCont.erase(sCont.begin());               // erase first
            queue from beginning
95        if (cQ.size() > 1) {
96          prev = cQ.size() - 2;
97        } else {
98          prev = 0;
99        }
100       // Loop through each neighbor
101       for (uint32_t j = 0; j < nBors.size(); j++) {
102         if (nBors[j] != cQ[prev]) // No backtracking
103         {
104           if (nBors[j] == cQ.front() && cQ.size() > 8) {
105             i = pEnd;
106           } // Back at first node
107           if (std::find(eList.begin(), eList.end(), nBors[j])
                  ==
108               eList.end()) // Check if this current route is
                     extended elsewhere
109           {
110             std::deque<uint32_t> nQ = cQ;
111             nQ.push_back(nBors[j]); // Add the neighbor to the
                    queue
112             sCont.push_front(nQ);   // add the sequence to the
                    front of the queue
113           }
114         }
115       }
116       if (nBors.size() > 2) {
117         eList.push_back(i);
118       } // if there are multiple choices put current node in
            extension List
119       if (i != pEnd) {
120         i = sCont.front().back();
121       } // If it isn't the end set i to the last node of the
            first queue
122       if (sCont.size() == 0) {
```

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

```
174  }
175  }
```

C.0.3 Neural Network 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 <fstream>
14
15  #include <boost/archive/xml_iarchive.hpp>
16  #include <boost/archive/xml_oarchive.hpp>
17  #include <boost/serialization/vector.hpp>
18  #include <boost/serialization/version.hpp>
19
20  #include "GA.h"
21  #include "MathException.h"
22  #include "SoilMathTypes.h"
23  #include "FFT.h"
24
25  #include <QtCore/QObject>
26
27  namespace SoilMath {
28  /*!
29   * \brief The Neural Network class
30   * \details This class is used to make prediction on large
         data set. Using self
31   * learning algoritmes
32   */
33  class NN : public QObject {
34    Q_OBJECT
35
36  public:
37    /*!
38   * \brief NN constructor for the Neural Net
39   * \param inputneurons number of input neurons
40   * \param hiddenneurons number of hidden neurons
41   * \param outputneurons number of output neurons
42   */
43    NN(uint32_t inputneurons, uint32_t hiddenneurons, uint32_t
         outputneurons);
44
45    /*!
46     * \brief NN constructor for the Neural Net
47     */
48    NN();
49
50    /*!
```

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

```
 93      * \param value the real valued vector with the values
 94      */
 95     void SetHiddenWeights(Weight_t value) { hWeights = value;
          }
 96
 97     /*!
 98      * \brief SetBeta a function to set the beta value
 99      * \param value a floating value ussualy between 0.5 and
          1.5
100      */
101     void SetBeta(float value) { beta = value; }
102     float GetBeta() { return beta; }
103
104     /*!
105      * \brief Learn the learning function
106      * \param input a vector of vectors with complex input
          values
107      * \param cat a vector of vectors with the know output
          values
108      * \param noOfDescriptorsUsed the total number of
          descriptos which should be
109      * used
110      */
111     void Learn(InputLearnVector_t input, OutputLearnVector_t
        cat,
112                 uint32_t noOfDescriptorsUsed);
113
114     /*!
115      * \brief SaveState Serialize and save the values of the
          Neural Net to disk
116      * \details Save the Neural Net in XML valued text file to
           disk so that a
117      * object can
118      * be reconstructed on a latter stadia.
119      * \param filename a string indicating the file location
          and name
120      */
121     void SaveState(std::string filename);
122
123     /*!
124      * \brief LoadState Loads the previouse saved Neural Net
          from disk
125      * \param filename a string indicating the file location
          and name
126      */
127     void LoadState(std::string filename);
128
129     Weight_t iWeights; /**< a vector of real valued floating
          point input weights*/
130     Weight_t hWeights; /**< a vector of real valued floating
          point hidden weight*/
131
132     uint32_t MaxGenUsedByGA = 200;
133     uint32_t PopulationSizeUsedByGA = 30;
134     float MutationrateUsedByGA = 0.075f;
135     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
               results of a
151                    learning curve*/
152
153  signals:
154    void learnErrorUpdate(double newError);
155
156  private:
157    GA *optim = nullptr;
158    std::vector<float> iNeurons; /**< a vector of input values
         , the bias is
159                                    included, the bias is
                                        included and
160                                    is the first value*/
161    std::vector<float>
162        hNeurons; /**< a vector of hidden values, the bias is
               included and
163                    is the first value*/
164    std::vector<float> oNeurons; /**< a vector of output
         values*/
165
166    uint32_t hiddenNeurons = 50; /**< number of hidden neurons
           minus bias*/
167    uint32_t inputNeurons = 20;  /**< number of input neurons
         minus bias*/
168    uint32_t outputNeurons = 18; /**< number of output neurons
         */
169    float beta; /**< the beta value, this indicates the
         steepness of the sigmoid
170                    function*/
171
172    friend class boost::serialization::access; /**< a private
         friend class so the
173                                                serialization
                                                    can
                                                    access
                                                    all
174                                                the needed
                                                    functions
                                                    */
175    /*!
176     * \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  }
```

```cpp
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(
          EXCEPTION_SIZE_OF_INPUT_NEURONS,
59                                      EXCEPTION_SIZE_OF_INPUT_NEURONS_NR
                                         );
60    }
61    if (!studied) {
62      throw Exception::MathException(
          EXCEPTION_NEURAL_NET_NOT_STUDIED,
63                                      EXCEPTION_NEURAL_NET_NOT_STUDIED_NR
                                         );
64    }
65
66    iNeurons.clear();
67    hNeurons.clear();
68    oNeurons.clear();
69
70    // Set the bias in the input and hidden vector to 1 (real
        number)
71    iNeurons.push_back(1.0f);
72    hNeurons.push_back(1.0f);
73
```

```
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] *
          beta)));
95    }
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
              be calculated
106   }
107
108   retVal.OutputNeurons = oNeurons;
109   retVal.ManualSet = false;
110   return retVal;
111 }
112
113 void NN::Learn(InputLearnVector_t input, OutputLearnVector_t
        cat,
114                 uint32_t noOfDescriptorsUsed __attribute__((
                    unused))) {
115   if (optim == nullptr) {
116     optim = new SoilMath::GA(PredictLearn, inputNeurons,
            hiddenNeurons, outputNeurons);
117   }
118   connect(optim, SIGNAL(learnErrorUpdate(double)), this,
        SIGNAL(learnErrorUpdate(double)));
119
120   optim->Elitisme = ElitismeUsedByGA;
121   optim->EndError = EndErrorUsedByGA;
122   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);
133
134    this->iWeights = Weight_t(
135        weight.begin(), weight.begin() + ((inputNeurons + 1) *
136            hiddenNeurons));
136    this->hWeights = Weight_t(
137        weight.begin() + ((inputNeurons + 1) * hiddenNeurons),
138            weight.end());
138    studied = true;
139 }
140
141 void NN::SetInputNeurons(uint32_t value) {
142    if (value != inputNeurons) {
143        inputNeurons = value;
144        iNeurons.clear();
145        iNeurons.reserve(inputNeurons + 1);
146        studied = false;
147    }
148 }
149
150 void NN::SetHiddenNeurons(uint32_t value) {
151    if (value != hiddenNeurons) {
152        hiddenNeurons = value;
153        hNeurons.clear();
154        hNeurons.reserve(hiddenNeurons + 1);
155        studied = false;
156    }
157 }
158
159 void NN::SetOutputNeurons(uint32_t value) {
160    if (value != outputNeurons) {
161        outputNeurons = value;
162        oNeurons.clear();
163        oNeurons.reserve(outputNeurons);
164        studied = false;
165    }
166 }
167 }
```

C.0.4 Statistical Class

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

#pragma once
#define MAX_UINT8_VALUE 256
#define VECTOR_CALC 1

#include <stdint.h>
#include <utility>
#include <vector>
#include <cstdlib>
#include <cmath>
#include <limits>
#include <typeinfo>
#include <string>

#include <fstream>

#include <boost/archive/binary_iarchive.hpp>
#include <boost/archive/binary_oarchive.hpp>
#include <boost/serialization/version.hpp>
#include <boost/math/distributions/students_t.hpp>

#include "MathException.h"
#include "SoilMathTypes.h"
#include "CommonOperations.h"

namespace SoilMath {

/*!
 * \brief Stats class
 * \details Usage Stats<type1, type2, type3>Stats() type 1,
     2 and 3 shoudl be of
 * the same value and concecuative in size
 */
template <typename T1, typename T2, typename T3> class Stats
    {
public:
  bool isDiscrete = true; /**< indicates if the data is
      discrete or real*/

  T1 *Data = nullptr;        /**< Pointer the data*/
  uint32_t *bins = nullptr; /**< the histogram*/
  double *CFD = nullptr;    /**< the CFD*/
  bool Calculated = false;  /**< indication if the data has
      been calculated*/
  float Mean = 0.0;         /**< the mean value of the data
      */
```

```cpp
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 /
                           statComp.n +
91                           this->Std * this->Std / this->n);
92      //
```

```
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
            hypothesis, the two means
100     // are the same
101     if (q < alpha / 2)
102       rejected = false;
103     else
104       rejected = true;
105     return rejected;
106   }
107
108   /*!
109    * \brief Stats Constructor
110    * \param rhs Right hand side
111    */
112   Stats(const Stats &rhs)
113       : bins{new uint32_t[rhs.noBins]{0}}, CFD{new double[
            rhs.noBins]{}},
114         BinRanges{new double[rhs.noBins]{}} {
115     this->binRange = rhs.binRange;
116     this->Calculated = rhs.Calculated;
117     this->Cols = rhs.Cols;
118     this->EndBin = rhs.EndBin;
119     this->isDiscrete = rhs.isDiscrete;
120     this->max = rhs.max;
121     this->Mean = rhs.Mean;
122     this->min = rhs.min;
123     this->n = rhs.n;
124     this->noBins = rhs.noBins;
125     this->n_end = rhs.n_end;
126     this->Range = rhs.Range;
127     this->Rows = rhs.Rows;
128     this->Startbin = rhs.Startbin;
129     this->Std = rhs.Std;
130     this->Sum = rhs.Sum;
131     std::copy(rhs.bins, rhs.bins + rhs.noBins, this->bins);
132     std::copy(rhs.CFD, rhs.CFD + rhs.noBins, this->CFD);
133     std::copy(rhs.BinRanges, rhs.BinRanges + rhs.noBins,
            this->BinRanges);
134     this->Data = rhs.Data;
135     this->StartAtZero = rhs.StartAtZero;
136     this->HighestPDF = rhs.HighestPDF;
137   }
138
139   /*!
140    * \brief operator = Assigmnet operator
141    * \param rhs right hand side
142    * \return returns the right hand side
143    */
144   Stats &operator=(Stats const &rhs) {
145     if (&rhs != this) {
```

```cpp
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        std::copy(rhs.CFD, rhs.CFD + rhs.noBins, this->CFD);
183        std::copy(rhs.BinRanges, rhs.BinRanges + rhs.noBins,
              this->BinRanges);
184        this->StartAtZero = rhs.StartAtZero;
185        this->HighestPDF = rhs.HighestPDF;
186      }
187      return *this;
188    }
189
190    /*!
191     * \brief Stats Constructor
192     * \param noBins number of bins with which to build the
              histogram
193     * \param startBin starting value of the first bin
194     * \param endBin end value of the second bin
195     */
196    Stats(int noBins = 256, T1 startBin = 0, T1 endBin = 255)
          {
197      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(
               double) ||
208            typeid(T1) == typeid(long double)) {
209          isDiscrete = false;
210          binRange = static_cast<T1>((EndBin - Startbin) /
                 noBins);
211        } else {
212          isDiscrete = true;
213          binRange = static_cast<T1>(round((EndBin - Startbin) /
                 noBins));
214        }
215      }
216
217      /*!
218       * \brief Stats constructor
219       * \param data Pointer to the data
220       * \param rows Number of rows
221       * \param cols Number of Columns
222       * \param noBins Number of bins
223       * \param startBin Value of the start bin
224       * \param startatzero bool indicating if the bins should
                 be shifted from zero
225       */
226      Stats(T1 *data, uint16_t rows, uint16_t cols, int noBins =
               256,
227            T1 startBin = 0, bool startatzero = true) {
228        min = std::numeric_limits<T1>::max();
229        max = std::numeric_limits<T1>::min();
230        Range = max - min;
231
232        Startbin = startBin;
233        EndBin = startBin + noBins;
234        StartAtZero = startatzero;
235
236        if (typeid(T1) == typeid(float) || typeid(T1) == typeid(
               double) ||
237            typeid(T1) == typeid(long double)) {
238          isDiscrete = false;
239        } else {
240          isDiscrete = true;
241        }
242
243        Data = data;
244        Rows = rows;
245        Cols = cols;
246        bins = new uint32_t[noBins]{0};
247        CFD = new double[noBins]{};
```

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

```
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(
            double) ||
313          typeid(T1) == typeid(long double)) {
314        isDiscrete = false;
315        throw Exception::MathException(
              EXCEPTION_TYPE_NOT_SUPPORTED,
316                                       EXCEPTION_TYPE_NOT_SUPPORTED_NR
                                          );
317      } else {
318        isDiscrete = true;
319      }
320
321      bins = new uint32_t[noBins]{0};
322      CFD = new double[noBins]{};
323      BinRanges = new double[noBins]{};
324      while (i-- > 0) {
325        bins[i] = binData[i];
326        n += binData[i];
327      }
328      BinCalculations(startC, endC);
329    }
330
331    ~Stats() {
332      Data == nullptr;
333      if (bins != nullptr) {
334        delete[] bins;
335        bins = nullptr;
336      }
337      if (CFD != nullptr) {
338        delete[] CFD;
339        CFD = nullptr;
340      }
341      if (BinRanges != nullptr) {
342        delete[] BinRanges;
343        BinRanges = nullptr;
344      }
345    }
346
347    /*!
348     * \brief BasicCalculateFloat execute the basic float data
            calculations
349     */
350    void BasicCalculateFloat() {
```

```
351      float sum_dev = 0.0;
352      n = Rows * Cols;
353      for (uint32_t i = 0; i < n; i++) {
354        if (Data[i] > max) {
355          max = Data[i];
356        }
357        if (Data[i] < min) {
358          min = Data[i];
359        }
360        Sum += Data[i];
361      }
362      binRange = (max - min) / noBins;
363      uint32_t index = 0;
364      Mean = Sum / (float)n;
365      Range = max - min;
366
367      if (StartAtZero) {
368        for (uint32_t i = 0; i < n; i++) {
369          index = static_cast<uint32_t>(Data[i] / binRange);
370          if (index == noBins) {
371            index -= 1;
372          }
373          bins[index]++;
374          sum_dev += pow((Data[i] - Mean), 2);
375        }
376      } else {
377        for (uint32_t i = 0; i < n; i++) {
378          index = static_cast<uint32_t>((Data[i] - min) /
                 binRange);
379          if (index == noBins) {
380            index -= 1;
381          }
382          bins[index]++;
383          sum_dev += pow((Data[i] - Mean), 2);
384        }
385      }
386      Std = sqrt((float)(sum_dev / n));
387      getCFD();
388      Calculated = true;
389    }
390
391    /*!
392     * \brief BasicCalculateFloat execute the basic float data
           calculations with a
393     * mask
394     * \param mask uchar mask type 0 don't calculate, 1
           calculate
395     */
396    void BasicCalculateFloat(uchar *mask) {
397      float sum_dev = 0.0;
398      n = Rows * Cols;
399      uint32_t nmask = 0;
400      for (uint32_t i = 0; i < n; i++) {
401        if (mask[i] != 0) {
402          if (Data[i] > max) {
403            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) /
                       binRange);
431            if (index == noBins) {
432              index -= 1;
433            }
434            bins[index]++;
435            sum_dev += pow((Data[i] - Mean), 2);
436          }
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
             calculations
446     */
447    void BasicCalculate() {
448      double sum_dev = 0.0;
449      n = Rows * Cols;
450      for (uint32_t i = 0; i < n; i++) {
451        if (Data[i] > max) {
452          max = Data[i];
453        }
454        if (Data[i] < min) {
455          min = Data[i];
456        }
457        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) /
             static_cast<float>(noBins)));
514      Mean = Sum / (float)nmask;
515      Range = max - min;
516
517      uint32_t index;
518      if (StartAtZero) {
519        i = 0;
520        std::for_each(Data, Data + n, [&](T1 &d) {
521          if (mask[i++] != 0) {
522            index = static_cast<uint32_t>(d / binRange);
523            if (index == noBins) {
524              index -= 1;
525            }
526            bins[index]++;
527            sum_dev += pow((d - Mean), 2);
528          }
529        });
530      } else {
531        i = 0;
532        std::for_each(Data, Data + n, [&](T1 &d) {
533          if (mask[i++] != 0) {
534            index = static_cast<uint32_t>((d - min) / binRange
               );
535            if (index == noBins) {
536              index -= 1;
537            }
538            bins[index]++;
539            sum_dev += pow((d - Mean), 2);
540          }
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
            histogram
550     * \param startC start counter
551     * \param endC end counter
552     */
553    void BinCalculations(uint16_t startC, uint16_t endC
          __attribute__((unused))) {
554      float sum_dev = 0.0;
555      // Get the Sum
556      uint32_t i = 0;
557      for_each(begin(), end(), [&](uint32_t &b) { Sum += b * (
            startC + i++); });
558
559      // Get Mean
560      Mean = Sum / (float)n;
561
```

```cpp
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(
                             Std, 2))));
616        yAxis.push_back(yVal);
617        if (yVal > HighestPDF) {
618          HighestPDF = yVal;
619        }
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<
                double>(n)) * 100.;
633      for (uint32_t i = 1; i < noBins; i++) {
634        sumBin[i] = (sumBin[i - 1] + bins[i]);
635        CFD[i] = (static_cast<double>(sumBin[i]) / static_cast
                  <double>(n)) * 100.;
636        if (CFD[i] > HighestPDF) {
637          HighestPDF = CFD[i];
638        }
639      }
640      delete[] sumBin;
641    }
642
643    friend class boost::serialization::access; /**<
              Serialization class*/
644
645    /*!
646     * \brief serialize the object
647     * \param ar argument
648     * \param version
649     */
650    template <class Archive>
651    void serialize(Archive &ar, const unsigned int version) {
652      if (version == 0) {
653        ar &isDiscrete;
654        ar &n;
655        ar &noBins;
656        for (size_t dc = 0; dc < noBins; dc++) {
657          ar &bins[dc];
658        }
659        for (size_t dc = 0; dc < noBins; dc++) {
660          ar &CFD[dc];
661        }
662        for (size_t dc = 0; dc < noBins; dc++) {
663          ar &BinRanges[dc];
664        }
665        ar &Calculated;
```

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

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
        strictly prohibited
3   * and only allowed with the written consent of the author (
        Jelle Spijker)
4   * This software is proprietary and confidential
5   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
6   */
7
8  #pragma once
9
10  #include "Stats.h"
11  #include <boost/serialization/base_object.hpp>
12
13  namespace SoilMath {
14  class PSD : public SoilMath::Stats<double, double, long
      double> {
15  private:
16    uint32_t DetBin(float value) {
17      uint32_t i = noBins - 1;
18      while (i > 0) {
19        if (value > BinRanges[i]) {
20          return i;
21        }
22        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>>(*
                  this);
58        }
59      }
60
61  public:
62      PSD() : SoilMath::Stats<double, double, long double>() {}
63
64      PSD(double *data, uint32_t nodata, double *binranges,
          uint32_t nobins,
65          uint32_t endbin)
66          : SoilMath::Stats<double, double, long double>(nobins,
              0, endbin) {
67        std::copy(binranges, binranges + nobins, BinRanges);
68        Data = data;
69        Rows = nodata;
70        Cols = 1;
71
72        BasicCalculatePSD();
73      }
74  };
75  }
```

```
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 -
        lboost_iostreams
50
51 #Zlib
52 LIBS += -L/usr/local/lib -lz
53 INCLUDEPATH += /usr/local/include
```

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

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

```cpp
 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,
         uint32_t noNeurons) {
105    std::vector<float> retVal(noNeurons, -1);
106    retVal[value - 1] = 1;
107    return retVal;
108  }
109
110  /*!
111   * \brief calcDiameter according to ISO 9276-6
112   * \param A
113   * \return
114   */
115  static inline float calcDiameter(float A) {
116    //return sqrt((4 * A) / M_PI);
117    return 1.1283791670955 * sqrt(A);
118  }
119  }
```

```cpp
 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  #define GENE_MAX 32  /**< maximum number of genes*/
10  #define CROSSOVER 16 /**< crossover location*/
11
12  #include <stdint.h>
13  #include <bitset>
14  #include <vector>
15  #include <complex>
16  #include <valarray>
17  #include <array>
18
19  typedef unsigned char uchar;   /**< unsigned char*/
20  typedef unsigned short ushort; /**< unsigned short*/
21  typedef unsigned int uint32_t;
22
23  typedef std::complex<double> Complex_t;       /**< complex
         vector of doubles*/
24  typedef std::vector<Complex_t> ComplexVect_t; /**< vector of
         Complex_t*/
25  typedef std::valarray<Complex_t> ComplexArray_t; /**<
         valarray of Complex_t*/
26  typedef std::vector<uint32_t> iContour_t;       /**< vector
         of uint32_t*/
27  typedef std::bitset<GENE_MAX> Genome_t; /**< Bitset
```

```cpp
                 repressenting 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*/
```

```
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
        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>
```

```cpp
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  }
```

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

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

D. Hardware Library

D.0.6 Microscope 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 Microscope
9  Interaction with the USB 5 MP microscope
10 */
11
12 #pragma once
13
14 #include <stdint.h>
15 #include <vector>
16 #include <string>
17 #include <utility>
18 #include <algorithm>
19
20 #include <sys/stat.h>
21 #include <sys/utsname.h>
22 #include <sys/ioctl.h>
23 #include <fstream>
24 #include <fcntl.h>
25
26 #include <linux/videodev2.h>
27 #include <linux/v4l2-controls.h>
28 #include <linux/v4l2-common.h>
29
```

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

```cpp
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
         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 "Microscope.h"
 9
10   namespace Hardware {
11
12   Microscope::Microscope() {
13     RunEnv = GetCurrentArchitecture();
14     AvailableCams = GetAvailableCams();
15     for_each(AvailableCams.begin(), AvailableCams.end(), [](
         Cam_t &C) {
16       C.SelectedResolution = &C.Resolutions[C.Resolutions.size
           () - 1];
17     });
18   }
19
20   Microscope::Microscope(const Microscope &rhs) {
21     std::copy(rhs.AvailableCams.begin(), rhs.AvailableCams.end
         (),
22               this->AvailableCams.begin());
23     this->RunEnv = rhs.RunEnv;
24     this->SelectedCam = rhs.SelectedCam;
25     this->cap = rhs.cap;
26     this->fd = rhs.fd;
27     this->HDRframes = rhs.HDRframes;
28   }
29
30   Microscope::~Microscope() { delete cap; }
31
32   Microscope::Arch Microscope::GetCurrentArchitecture() {
33     struct utsname unameData;
34     Arch retVal;
35     uname(&unameData);
36     std::string archString = static_cast<std::string>(
         unameData.machine);
```

```cpp
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    const string path_ss = "/sys/class/video4linux";
47    const string path_ss_dev = "/dev/video";
48    std::vector<Cam_t> retVal;
49    struct v4l2_queryctrl queryctrl;
50    struct v4l2_control controlctrl;
51
52    // Check if there're videodevices installed
53    // Itterate through the cams
54    for (boost::filesystem::directory_iterator itr(path_ss);
55         itr != boost::filesystem::directory_iterator(); ++itr
             ) {
56      string videoln = itr->path().string();
57      videoln.append("/name");
58      if (boost::filesystem::exists(videoln)) {
59        Cam_t currentCam;
60        std::ifstream camName;
61        camName.open(videoln);
62        std::getline(camName, currentCam.Name);
63        camName.close();
64        currentCam.ID =
65            std::atoi(itr->path().string().substr(28, std::
                string::npos).c_str());
66
67        // Open Cam
68        currentCam.devString = path_ss_dev + std::to_string(
            currentCam.ID);
69        if ((currentCam.fd = open(currentCam.devString.c_str()
            , O_RDWR)) == -1) {
70          throw Exception::MicroscopeException(
              EXCEPTION_NOCAMS,
71                                                EXCEPTION_NOCAMS_NR
                                                  );
72        }
73
74        // Get controls
75        memset(&queryctrl, 0, sizeof(queryctrl));
76        memset(&controlctrl, 0, sizeof(controlctrl));
77        for (queryctrl.id = V4L2_CID_BASE; queryctrl.id <
            V4L2_CID_LASTP1;
78             queryctrl.id++) {
79
80          if (ioctl(currentCam.fd, VIDIOC_QUERYCTRL, &
              queryctrl) == 0) {
81            if (!(queryctrl.flags & V4L2_CTRL_FLAG_DISABLED))
                {
82              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(
                EXCEPTION_QUERY,
99                                                EXCEPTION_QUERY_NR
                                                  );
100         }
101       }
102
103       // Get image formats
104       struct v4l2_format format;
105       memset(&format, 0, sizeof(format));
106
107       uint32_t width[5] = {640, 800, 1280, 1600, 2048};
108       uint32_t height[6] = {480, 600, 960, 1200, 1536};
109
110       uint32_t ResolutionID = 0;
111
112       // YUYV
113       for (uint32_t i = 0; i < 5; i++) {
114         format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
115         format.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;
116         format.fmt.pix.width = width[i];
117         format.fmt.pix.height = height[i];
118         int ret = ioctl(currentCam.fd, VIDIOC_S_FMT, &format
                  );
119         if (ret != -1 && format.fmt.pix.height == height[i]
                  &&
120             format.fmt.pix.width == width[i]) {
121           Resolution_t res;
122           res.Width = format.fmt.pix.width;
123           res.Height = height[i];
124           res.ID = ResolutionID++;
125           res.format = PixelFormat::YUYV;
126           currentCam.Resolutions.push_back(res);
127         }
128       }
129
130       // MJPEG
131       for (uint32_t i = 0; i < 5; i++) {
```

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

```cpp
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    // create the brightness steps
248    Control_t *brightness = GetControl("Brightness");
249    Control_t *contrast = GetControl("Contrast");
250
251    uint32_t brightnessStep =
252        (brightness->maximum - brightness->minimum) / noframes
             ;
253    int8_t currentBrightness = brightness->current_value;
254    int8_t currentContrast = contrast->current_value;
255    contrast->current_value = contrast->maximum;
256
257    cv::Mat currentImg;
258    // take the shots at different brightness levels
259    for (uint32_t i = 1; i <= noframes; i++) {
260      brightness->current_value = brightness->minimum + (i *
           brightnessStep);
261      GetFrame(currentImg);
262      HDRframes.push_back(currentImg);
263    }
264
265    // Set the brightness and back to the previous used level
266    brightness->current_value = currentBrightness;
267    contrast->current_value = currentContrast;
268
269    // Perform the exposure fusion
270    cv::Mat fusion;
271    cv::Ptr<cv::MergeMertens> merge_mertens = cv::
           createMergeMertens();
272    merge_mertens->process(HDRframes, fusion);
273    fusion *= 255;
274    fusion.convertTo(dst, CV_8UC1);
275  }
276
277  Microscope::Control_t *Microscope::GetControl(const string
         name) {
278    for (Controls_t::iterator it = SelectedCam->Controls.begin
           ();
279          it != SelectedCam->Controls.end(); ++it) {
280      if (name.compare(it->name) == 0) {
281        return &*it;
282      }
283    }
284    return nullptr;
285  }
```

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

D.0.7 Beaglebone Black Class

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

/*! \class BBB
The core BeagleBone Black class used for all hardware
   related classes.
Consisting of universal used method, functions and variables
   . File operations,
polling and threading
*/

#pragma once

#define SLOTS


    \
  "/sys/devices/platform/bone_capemgr/slots" /*!< Beaglebone
      capemanager slots file*/

#include <fstream>
#include <sstream>
#include <string>
#include <sys/stat.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/epoll.h>
#include <fcntl.h>
#include <regex>
#include <stdexcept>

#include "GPIOReadException.h"
#include "FailedToCreateGPIOPollingThreadException.h"
#include "ValueOutOfBoundsException.h"

using namespace std;

namespace Hardware {
typedef int (*CallbackType)(
    int); /*!< CallbackType used to pass a function to a
        thread*/

class BBB {
public:
  int debounceTime; /*!< debounce time for a button in
      milliseconds*/

  BBB();
  ~BBB();
```

```
46
47  protected:
48    bool threadRunning;              /*!< used to stop the
          thread*/
49    pthread_t thread;                /*!< The thread*/
50    CallbackType callbackFunction;   /*!< the callbakcfunction*/
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: " +
          path).c_str());
31    }
32    string input;
33    getline(fs, input);
34    fs.close();
35    return input;
36  }
37
38  /*! Writes a value to a file
39  \param path a constant string pointing towards the file
40  \param value a constant string which should be written in
      the file
41  */
42  void BBB::Write(const string &path, const string &value) {
43    ofstream fs;
44    fs.open(path.c_str());
45    if (!fs.is_open()) {
46      throw Exception::GPIOReadException(("Can't open: " +
          path).c_str());
47    }
48    fs << value;
49    fs.close();
50  }
51
52  /*! Checks if a directory exist
53  \returns true if the directory exists and false if not
54  */
55  bool BBB::DirectoryExist(const string &path) {
56    struct stat st;
57    if (stat((char *)path.c_str(), &st) != 0) {
58      return false;
59    }
60    return true;
61  }
62
63  /*! Checks if a cape is loaded in the file /sys/devices/
      bone_capemgr.9/slots
64  \param shield a const search string which is a (part) of the
        shield name
65  \return true if the search string is found otherwise false
66  */
67  bool BBB::CapeLoaded(const string &shield) {
68    bool shieldFound = false;
69
70    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.0.8 GPIO Class

```cpp
/* Copyright (C) Jelle Spijker - All Rights Reserved
 * Unauthorized copying of this file, via any medium is
     strictly prohibited
 * and only allowed with the written consent of the author (
     Jelle Spijker)
 * This software is proprietary and confidential
 * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
 * This code is based upon:
 * Derek Molloy, "Exploring BeagleBone: Tools and Techniques
     for Building
 * with Embedded Linux", Wiley, 2014, ISBN:9781118935125.
 * See: www.exploringbeaglebone.com
 */

#pragma once
#include "BBB.h"

#define EXPORT_PIN "/sys/class/gpio/export"
#define UNEXPORT_PIN "/sys/class/gpio/unexport"
#define GPIOS "/sys/class/gpio/gpio"
#define DIRECTION "/direction"
#define VALUE "/value"
#define EDGE "/edge"

using namespace std;

namespace Hardware {
class GPIO : public BBB {
public:
  enum Direction { Input, Output };
  enum Value { Low = 0, High = 1 };
  enum Edge { None, Rising, Falling, Both };

  int number; // Number of the pin

  int WaitForEdge();
  int WaitForEdge(CallbackType callback);
  void WaitForEdgeCancel() { this->threadRunning = false; }

  Value GetValue();
  void SetValue(Value value);

  Direction GetDirection();
  void SetDirection(Direction direction);

  Edge GetEdge();
  void SetEdge(Edge edge);

  GPIO(int number);
  ~GPIO();

private:
  string gpiopath;
  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
         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 "GPIO.h"
9
10 namespace Hardware {
11 GPIO::GPIO(int number) {
12
13   this->number = number;
14   gpiopath = GPIOS + NumberToString<int>(number);
15
16   if (!isExported(number, direction, edge)) {
17     ExportPin(number);
18     direction = ReadsDirection(gpiopath);
19     edge = ReadsEdge(gpiopath);
20   }
21   usleep(250000);
22 }
23
24 GPIO::~GPIO() { UnexportPin(number); }
25
26 int GPIO::WaitForEdge(CallbackType callback) {
27   threadRunning = true;
28   callbackFunction = callback;
29   if (pthread_create(&this->thread, NULL, &threadedPollGPIO,
30                      static_cast<void *>(this))) {
31     threadRunning = false;
32     throw Exception::
          FailedToCreateGPIOPollingThreadException();
```

```cpp
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::
          FailedToCreateGPIOPollingThreadException(
46        "GPIO: Failed to create epollfd!");
47    }
48    if ((fd = open((gpiopath + VALUE).c_str(), O_RDONLY |
        O_NONBLOCK)) == -1) {
49      throw Exception::GPIOReadException();
50    }
51
52    // read operation | edge triggered | urgent data
53    ev.events = EPOLLIN | EPOLLET | EPOLLPRI;
54    ev.data.fd = fd;
55
56    if (epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev) == -1) {
57      throw Exception::
          FailedToCreateGPIOPollingThreadException(
58        "GPIO: Failed to add control interface!");
59    }
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  void GPIO::SetValue(GPIO::Value value) { WritesValue(
      gpiopath, value); }
76
77  GPIO::Direction GPIO::GetDirection() { return direction; }
78  void GPIO::SetDirection(Direction direction) {
79    this->direction = direction;
80    WritesDirection(gpiopath, direction);
81  }
82
83  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)),
        Direction &dir,
90                         Edge &edge) {
91     // Checks if directory exist and therefore is exported
92     if (!DirectoryExist(gpiopath)) {
93       return false;
94     }
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:
```

```cpp
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   if (Read(gpiopath + DIRECTION) == "in") {
207     return Input;
208   } else {
209     return Output;
210   }
211 }
212
213 void GPIO::WritesDirection(const string &gpiopath, Direction
        direction) {
214   switch (direction) {
215   case Hardware::GPIO::Input:
216     Write((gpiopath + DIRECTION), "in");
217     break;
218   case Hardware::GPIO::Output:
219     Write((gpiopath + DIRECTION), "out");
220     break;
221   }
222 }
223
224 GPIO::Edge GPIO::ReadsEdge(const string &gpiopath) {
225   string reader = Read(gpiopath + EDGE);
226   if (reader == "none") {
227     return None;
228   } else if (reader == "rising") {
229     return Rising;
230   } else if (reader == "falling") {
231     return Falling;
232   } else {
233     return Both;
234   }
235 }
236
237 void GPIO::WritesEdge(const string &gpiopath, Edge edge) {
238   switch (edge) {
239   case Hardware::GPIO::None:
240     Write((gpiopath + EDGE), "none");
241     break;
242   case Hardware::GPIO::Rising:
243     Write((gpiopath + EDGE), "rising");
244     break;
245   case Hardware::GPIO::Falling:
246     Write((gpiopath + EDGE), "falling");
247     break;
248   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      Write(gpiopath + VALUE, NumberToString<int>(value));
264  }
265
266  void *threadedPollGPIO(void *value) {
267      GPIO *gpio = static_cast<GPIO *>(value);
268      while (gpio->threadRunning) {
269        gpio->callbackFunction(gpio->WaitForEdge());
270        usleep(gpio->debounceTime * 1000);
271      }
272      return 0;
273  }
274  }
```

D.0.9 PWM Class

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

#pragma once
#include "BBB.h"
#include <dirent.h>

#define OCP_PATH "/sys/class/pwm/"
#define PWM_CAPE "Override Board Name,00A0,Override Manuf,
    cape-universaln"

namespace Hardware {
class PWM : public BBB {
public:
  enum Pin // Four possible PWM pins
  { P8_13,
    P8_19,
    P9_14,
    P9_16 };
  enum Run // Signal generating
  { On = 1,
    Off = 0 };
  enum Polarity // Inverse duty polarity
  { Normal = 1,
    Inverted = 0 };

  Pin pin; // Current pin

  uint8_t GetPixelValue() { return pixelvalue; }
  void SetPixelValue(uint8_t value);

  float GetIntensity() { return intensity; };
  void SetIntensity(float value);

  int GetPeriod() { return period; };
  void SetPeriod(int value);

  int GetDuty() { return duty; };
  void SetDuty(int value);
  void SetIntensity();

  Run GetRun() { return run; };
  void SetRun(Run value);

  Polarity GetPolarity() { return polarity; };
  void SetPolarity(Polarity value);

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

```cpp
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    polarity = static_cast<Polarity>(StringToNumber<int>(Read(
        polaritypath)));
60
61    // calculate the current intensity
62    calcIntensity();
63  }
64
65  PWM::~PWM() {}
66
67  /// <summary>
68  /// Calculate the current intensity
69  /// </summary>
70  void PWM::calcIntensity() {
71    if (polarity == Normal) {
72      if (duty == 0) {
73        intensity = 0.0f;
74      } else {
75        intensity = (float)period / (float)duty;
76      }
77    } else {
78      if (period == 0) {
79        intensity = 0.0f;
80      } else {
81        intensity = (float)duty / (float)period;
82      }
83    }
84  }
85
86  /// <summary>
87  /// Set the intensity level as percentage
88  /// </summary>
```

```
89   /// <param name="value">floating value multipication 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.0.10 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    "/sys/bus/iio/devices/iio:device0/in_voltage0_raw" /*!<
        path to analogue pin \
19                                                       0*/
20  #define ADC1_PATH

      \
21    "/sys/bus/iio/devices/iio:device0/in_voltage1_raw" /*!<
        path to analogue pin \
22                                                       1*/
23  #define ADC2_PATH

      \
24    "/sys/bus/iio/devices/iio:device0/in_voltage2_raw" /*!<
        path to analogue pin \
25                                                       2*/
26  #define ADC3_PATH

      \
27    "/sys/bus/iio/devices/iio:device0/in_voltage3_raw" /*!<
        path to analogue pin \
28                                                       3*/
29  #define ADC4_PATH

      \
30    "/sys/bus/iio/devices/iio:device0/in_voltage4_raw" /*!<
        path to analogue pin \
31                                                       4*/
32  #define ADC5_PATH

      \
33    "/sys/bus/iio/devices/iio:device0/in_voltage5_raw" /*!<
        path to analogue pin \
34                                                       5*/
```

```cpp
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
```

```cpp
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
         voltage*/
61  void ADC::SetMinIntensity() {
62    MinIntensity = StringToNumber<int>(Read(adcpath));
63  }
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
         polling indicates a
71  change CallbackType
72  \return 0
73  */
74  int ADC::WaitForValueChange(CallbackType callback) {
75    threadRunning = true;
76    callbackFunction = callback;
77    if (pthread_create(&thread, NULL, &threadedPollADC,
78                       static_cast<void *>(this))) {
79      threadRunning = false;
80      throw Exception::
           FailedToCreateGPIOPollingThreadException();
81    }
82    return 0;
83  }
84
85  /*! Polling of the analogue pin
86  \return the current value
87  */
88  int ADC::WaitForValueChange() {
89    int fd, i, epollfd, count = 0;
90    struct epoll_event ev;
91    epollfd = epoll_create(1);
92    if (epollfd == -1) {
93      throw Exception::
           FailedToCreateGPIOPollingThreadException(
94        "GPIO: Failed to create epollfd!");
95    }
96    if ((fd = open(adcpath.c_str(), O_RDONLY | O_NONBLOCK)) ==
         -1) {
```

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

D.0.11 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 <spijker.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 sleepperiod);
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 sleepperiod;    /*!< Sleep period*/
59    friend void *colorLoop(void *value);
60  };
61  void *colorLoop(void *value);
62  }
```

```
1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
        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 "EC12P.h"
9
10 namespace Hardware {
11 /*! Constructor*/
12 EC12P::EC12P() {
13   // Init Rotary button
14   Button.SetDirection(GPIO::Input);
15   Button.SetEdge(GPIO::Rising);
16
17   // Init Encoder
18   Rotary.set_period(100000000L);
19
20   // Init Encoder color
21   R.SetDirection(GPIO::Output);
22   B.SetDirection(GPIO::Output);
23   G.SetDirection(GPIO::Output);
24   SetPixelColor(None);
25
26   threadRunning = false;
27 }
28
29 /*! De-constructor*/
30 EC12P::~EC12P() {}
31
32 /*! Set the shaft color
33 \param value as Color enumerator
34 */
35 void EC12P::SetPixelColor(Color value) {
36   switch (value) {
37   case Hardware::EC12P::Red:
38     R.SetValue(GPIO::High);
39     B.SetValue(GPIO::Low);
40     G.SetValue(GPIO::Low);
41     break;
```

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

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

D.0.12 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
```

```cpp
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
         poll the pin, whereas
63    // passing false reads the immediate value
64    int32_t get_position(bool _poll = true);
65
66    // Thread of the poll
67    int WaitForPositionChange(CallbackType callback);
68    void WaitForPositionChangeCancel() { this->threadRunning =
         false; }
69
70    // Set the polling period
71    void set_period(long long unsigned int period);
72
73    // Get the polling period of the encoder
74    uint64_t get_period();
75
76    // Set the mode of the eQEP hardware
77    void set_mode(eQEP_Mode mode);
78
79    // Get the mode of the eQEP hardware
80    eQEP_Mode get_mode();
81
82  private:
83    friend void *threadedPolleqep(void *value);
84  };
85
86  void *threadedPolleqep(void *value);
87  }
```

```cpp
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  * This file is modified by Jelle Spijker 2014
21  * Added polling and threading capabilties
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 <cstdio>
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
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::eQEP_Mode _mode) {
106     // Open the file representing the position
107     FILE *fp = fopen((this->path + "/mode").c_str(), "w");
108
```

```cpp
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(),
            O_RDWR)) < 0) {
147        // Error, break out
148        std::cerr << "[eQEP " << this->path
149                   << "] unable to open position for polling"
                      << std::endl;
150        return 0;
151      }
152
153      // Dummy read
154      read(ufd.fd, &dummy, 1);
155
156      // Poll the port
157      ufd.events = (short)EPOLLET;
158      if (poll(&ufd, 1, -1) < 0) {
159        // Error, break out
160          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
```

```cpp
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"
227                << std::endl;
228      return eQEP::eQEP_Mode_Error;
229    }
230
231    // Write the desired value to the file
232    eQEP::eQEP_Mode mode;
233    fscanf(fp, "%u", (unsigned int *)&mode);
234
235    // Commit changes
236    fclose(fp);
237
238    // Return the mode
239    return mode;
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.0.13 SoilCape Class

```cpp
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 }
```

```cpp
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.0.14 USB Class

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

#pragma once

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>

#include <linux/usbdevice_fs.h>

namespace Hardware {
class USB {
public:
  USB();
  ~USB();
  void ResetUSB();
};
}
```

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

#include "USB.h"

namespace Hardware {
USB::USB() {}

USB::~USB() {}

void USB::ResetUSB() {
  int fd, rc;

  fd = open("/dev/bus/usb/001/002", O_WRONLY);
  rc = ioctl(fd, USBDEVFS_RESET, 0);
  if (rc < 0) {
    throw - 1;
  }
  close(fd);
}
```

```
25  }
```

D.0.15 General project files

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

```cpp
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 }
```

```cpp
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 }
```

```cpp
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()
        _GLIBCXX_USE_NOEXCEPT{};
23    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
        msg.c_str(); };
24
25  private:
26    string msg;
27  };
28  }
29  }
```

```
 1  /* Copyright (C) Jelle Spijker - All Rights Reserved
 2   * Unauthorized copying of this file, via any medium is
        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 FailedToCreateThreadException : public std::exception
        {
18  public:
19    FailedToCreateThreadException(string m = "Couldn't create
          the thread!")
20        : msg(m){};
21    ~FailedToCreateThreadException() _GLIBCXX_USE_NOEXCEPT{};
22    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
        msg.c_str(); };
23
24  private:
25    string msg;
26  };
27  }
28  }
```

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

```cpp
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
```

```cpp
 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
          next image!")
19        : msg(m){};
20    ~CouldNotGrabImageException() _GLIBCXX_USE_NOEXCEPT{};
21    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
          msg.c_str(); };
22
23  private:
24    string msg;
25  };
26  }
27  }
```

```cpp
 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 GPIOReadException : public std::exception {
18  public:
19    GPIOReadException(string m = "Can't read GPIO data!") :
          msg(m){};
20    ~GPIOReadException() _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
10  #include <exception>
11  #include <string>
12
13  using namespace std;
14
15  namespace Hardware {
16  namespace Exception {
17  class GPIOReadException : public std::exception {
18  public:
19    GPIOReadException(string m = "Can't read GPIO data!") :
         msg(m){};
20    ~GPIOReadException() _GLIBCXX_USE_NOEXCEPT{};
21    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
         msg.c_str(); };
22
23  private:
24    string msg;
25  };
26  }
27  }
```

E. Vision Library

E.0.16 Image processing Class

```c
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   if (chain) {                                                 \
16     TempImg = ProcessedImg.clone();                            \
17     O = (type *)TempImg.data;                                  \
18   } else {                                                     \
19     O = (type *)OriginalImg.data;                              \
20   }
```

```cpp
21  /*! MACRO which trows an EmtpyImageException if the matrix
        is empty*/
22  #define EMPTY_CHECK(img)
                                                                      \
23    if (img.empty()) {

          \
24      throw Exception::EmtpyImageException();
                                                        \
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 "PixelValueOutOfBoundException.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 getOriented(Mat &BW, cv::Point_<double> &
          centroid,
67                                    double &theta, double &
                                        eccentricty);
```

```cpp
68    static void RotateImg(Mat &src, Mat &dst, double &theta,
         cv::Point_<double> &Centroid, Rect &ROI);
69
70    double currentProg = 0.;
71    double ProgStep = 0.;
72
73    static std::vector<Mat> extractChannel(const Mat &src);
74
75    /*! Copy a matrix to a new matrix with a LUT mask
76    \param src the source image
77    \param *LUT type T with a LUT to filter out unwanted pixel
         values
78    \param cvType an in where you can pas CV_UC8C1 etc.
79    \return The new matrix
80    */
81    template <typename T1, typename T2>
82    static Mat CopyMat(const Mat &src, T1 *LUT, int cvType) {
83      Mat dst(src.size(), cvType);
84      uint32_t nData = src.rows * src.cols * dst.step[1];
85      if (cvType == 0 || cvType == 8 || cvType == 16 || cvType
           == 24) {
86        for (uint32_t i = 0; i < nData; i += dst.step[1]) {
87          dst.data[i] =
88              static_cast<uint8_t>(LUT[*(T2 *)(src.data + (i *
                 src.step[1]))]);
89        }
90      } else if (cvType == 1 || cvType == 9 || cvType == 17 ||
           cvType == 25) {
91        for (uint32_t i = 0; i < nData; i += src.step[1]) {
92          dst.data[i] =
93              static_cast<int8_t>(LUT[*(T2 *)(src.data + (i *
                 src.step[1]))]);
94        }
95      } else if (cvType == 2 || cvType == 10 || cvType == 18
           || cvType == 26) {
96        for (uint32_t i = 0; i < nData; i += src.step[1]) {
97          dst.data[i] =
98              static_cast<uint16_t>(LUT[*(T2 *)(src.data + (i
                 * src.step[1]))]);
99        }
100     } else if (cvType == 3 || cvType == 11 || cvType == 19
           || cvType == 27) {
101       for (uint32_t i = 0; i < nData; i += src.step[1]) {
102         dst.data[i] =
103             static_cast<int16_t>(LUT[*(T2 *)(src.data + (i *
                 src.step[1]))]);
104       }
105     } else if (cvType == 4 || cvType == 12 || cvType == 20
           || cvType == 28) {
106       for (uint32_t i = 0; i < nData; i += src.step[1]) {
107         dst.data[i] =
108             static_cast<int32_t>(LUT[*(T2 *)(src.data + (i *
                 src.step[1]))]);
109       }
110     }
111     return dst;
```

```
112    }
113
114    /*! Copy a matrix to a new matrix with a mask
115    \param src the source image
116    \param *LUT type T with a LUT to filter out unwanted pixel
              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 {
141                          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 &
                 sItem) {
169              std::for_each(sItem.begin<T1>(), sItem.end<T1>(),
                   [&](const T1 &s) {
170                if (s > MatMax) {
171                  MatMax = s;
172                } else if (s < MatMin) {
173                  MatMin = s;
174                }
175              });
176            });
177
178            int Range = MatMax - MatMin;
179            if (Range < 1)
180              Range = maxVal;
181
182            // Convert the values
183            for_each(exSrc.begin(), exSrc.end(), [&](const cv::
                 Mat &sItem) {
184              Mat dItem(img.size(), cvBaseType);
185              std::transform(sItem.begin<T1>(), sItem.end<T1>(),
                   dItem.begin<T1>(),
186                            [&](const T1 &s) -> T1 {
187                              return (T1)round(((s - MatMin) *
                                maxVal) / Range);
188                            });
189              exDst.push_back(dItem);
190            });
191
192            merge(exDst, tempImg);
193          } else {
194            tempImg = img;
195          }
196          cv::namedWindow(windowName, cv::WINDOW_NORMAL);
197          cv::imshow(windowName, tempImg);
198          cv::waitKey(0);
199          cv::destroyWindow(windowName);
200        };
201      };
202    };
203  }
```

```
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 &
                                          eccentricty) {
50    cv::Moments Mu = cv::moments(BW, true);
51
52    centroid.x = Mu.m10 / Mu.m00;
```

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

```
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::
       slot_type &subscriber) {
131    return prog_sig.connect(subscriber);
132 }
133 }
```

E.0.17 Conversion Class

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

#pragma once
#include "ImageProcessing.h"
#include "ConversionNotSupportedException.h"

namespace Vision {
class Conversion : public ImageProcessing {
public:
  /*! Enumerator which indicates the colorspace used*/
  enum ColorSpace {
    CIE_lab,   /*!< CIE La*b* colorspace */
    CIE_XYZ,   /*!< CIE XYZ colorspace */
    RI,        /*!< Redness Index colorspace */
    RGB,       /*!< RGB colorspace */
    Intensity, /*!< Grayscale colorspace */
    None       /*!< none */
  };
  ColorSpace OriginalColorSpace;  /*!< The original
      colorspace*/
  ColorSpace ProcessedColorSpace; /*!< The destination
      colorspace*/

  Conversion();
  Conversion(const Mat &src);
  Conversion(const Conversion &rhs);

  ~Conversion();

  Conversion &operator=(Conversion rhs);

  void Convert(ColorSpace convertFrom, ColorSpace convertTo,
               bool chain = false);
  void Convert(const Mat &src, Mat &dst, ColorSpace
      convertFrom,
               ColorSpace convertTo, bool chain = false);

private:
  /*!< Conversion matrix used in the conversion between RGB
       and CIE XYZ*/
  float XYZmat[3][3] = {{0.412453, 0.357580, 0.180423},
                        {0.212671, 0.715160, 0.072169},
                        {0.019334, 0.119194, 0.950227}};

  float whitePoint[3] = {
      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-constructor of the class*/
38 Conversion::~Conversion() {}
39
40 /*! Assignment operator*/
```

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

```
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 *O;
104     CHAIN_PROCESS(chain, O, uchar);
105
106     prog_sig(currentProg, "RGB 2 Intensity conversion");
107     RGB2Intensity(O, 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 *O;
115     CHAIN_PROCESS(chain, O, uchar);
116
117     prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
118     RGB2XYZ(O, 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 *O;
126     CHAIN_PROCESS(chain, O, uchar);
127
128     prog_sig(currentProg, "RGB 2 CIE XYZ conversion");
129     RGB2XYZ(O, 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 *O;
138     CHAIN_PROCESS(chain, O, uchar);
139
```

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

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

E.0.18 Enhance Class

```cpp
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 ENHANCE_VERSION 1
10
11 #include "ImageProcessing.h"
12 #include "../SoilMath/SoilMath.h"
13
14 using namespace std;
15 using namespace SoilMath;
16
17 namespace Vision {
18 class Enhance : public ImageProcessing {
19 private:
20   void CalculateSumOfNeighboringPixels(uchar *O, int i, int
        hKsize, int nCols,
21                                         uint32_t &sum);
22   float CalculateStdOfNeighboringPixels(uchar *O, int i, int
        hKsize, int nCols,
23                                         int noNeighboursPix,
                                            float mean);
24
25 public:
26   /*! Enumerator indicating the requested enhancement
        operation*/
27   enum EnhanceOperation {
28     _AdaptiveContrastStretch, /*!< custom adaptive contrast
          stretch operation*/
29     _Blur,                    /*!< Blur operation*/
30     _HistogramEqualization    /*!< Histogram equalization*/
31   };
32
33   Enhance();
34   Enhance(const Mat &src);
35   Enhance(const Mat &src, Mat &dst, uint8_t kernelsize = 9,
        float factor = 1.0,
36           EnhanceOperation operation = _Blur);
37   Enhance(const Enhance &rhs);
38
39   ~Enhance();
40
41   Enhance &operator=(Enhance rhs);
42
43   void AdaptiveContrastStretch(uint8_t kernelsize, float
        factor,
44                                bool chain = false);
```

```
45   void AdaptiveContrastStretch(const Mat &src, Mat &dst,
        uint8_t kernelsize,
46                                  float factor);
47
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
        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 Enhance
9  class which enhances a greyscale cv::Mat image
10 */
11 #include "Enhance.h"
12
13 namespace Vision {
14 /*! Constructor*/
15 Enhance::Enhance() {}
16
17 /*! Constructor
18 \param src cv::Mat source image
19 */
20 Enhance::Enhance(const Mat &src) {
21   OriginalImg = src;
22   ProcessedImg.create(OriginalImg.size(), CV_8UC1);
23 }
24
25 Enhance::Enhance(const Enhance &rhs) {
26   this->OriginalImg = rhs.OriginalImg;
27   this->ProcessedImg = rhs.OriginalImg;
28   this->TempImg = rhs.TempImg;
29 }
30
31 /*! Constructor
32 \param src cv::Mat source image
33 \param dst cv::Mat destination image
34 \param kernelsize an uchar which represent the kernelsize
        should be an uneven
35 number higher than two
36 \param factor float which indicates the amount the effect
        should take place
37 standard value is 1.0 only used in the adaptive contrast
        stretch enhancement
38 \param operation enumerator EnhanceOperation which
        enhancement should be
```

```
39  performed
40  */
41  Enhance::Enhance(const Mat &src, Mat &dst, uchar kernelsize,
        float factor,
42                    EnhanceOperation operation) {
43    OriginalImg = src;
44    ProcessedImg.create(OriginalImg.size(), CV_8UC1);
45    switch (operation) {
46    case Vision::Enhance::_AdaptiveContrastStretch:
47      AdaptiveContrastStretch(kernelsize, factor);
48      break;
49    case Vision::Enhance::_Blur:
50      Blur(kernelsize);
51      break;
52    case Vision::Enhance::_HistogramEqualization:
53      HistogramEqualization();
54      break;
55    }
56    dst = ProcessedImg;
57  }
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
        pixels
72  \param O uchar pointer to the current pixel of the original
        image
73  \param i current counter
74  \param hKsize half the kernelsize
75  \param nCols total number of columns
76  \param noNeighboursPix total number of neighboring pixels
77  \param mean mean value of the neighboring pixels
78  \return standard deviation
79  */
80  float Enhance::CalculateStdOfNeighboringPixels(uchar *O, int
        i, int hKsize,
81                                                  int nCols,
                                                     int
                                                     noNeighboursPix
                                                     ,
82                                                  float mean) {
83    uint32_t sum_dev = 0.0;
84    float Std = 0.0;
85    sum_dev = 0.0;
86    Std = 0.0;
87    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]
             - mean));
 91      }
 92    }
 93    // Std = sqrt(sum_dev / noNeighboursPix);
 94    Std = SoilMath::fastPow((static_cast<double>(sum_dev) /
         noNeighboursPix), 2);
 95    return Std;
 96  }
 97
 98  /*! Calculate the sum of the neighboring pixels
 99  \param O uchar pointer to the current pixel of the original
       image
100  \param i current counter
101  \param hKsize half the kernelsize
102  \param nCols total number of columns
103  \param sum Total sum of the neighboringpixels
104  */
105  void Enhance::CalculateSumOfNeighboringPixels(uchar *O, int
       i, int hKsize,
106                                               int nCols,
                                                  uint32_t &
                                                  sum) {
107    for (int j = -hKsize; j < hKsize; j++) {
108      for (int k = -hKsize; k < hKsize; k++) {
109        sum += O[i + j * nCols + k];
110      }
111    }
112  }
113
114  /*! Homebrew AdaptiveContrastStretch function which
       calculate the mean and
115  standard deviation from the neighboring pixels if the
       current pixel is higher
116  then the mean the value is incremented with an given factor
       multiplied with the
117  standard deviation, and decreased if it's lower then the
       mean.
118  \param src cv::Mat source image
119  \param dst cv::Mat destination image
120  \param kernelsize an uchar which represent the kernelsize
       should be an uneven
121  number higher than two
122  \param factor float which indicates the amount the effect
       should take place
123  standard value is 1.0 only used in the adaptive contrast
       stretch enhancement
124  */
125  void Enhance::AdaptiveContrastStretch(const Mat &src, Mat &
       dst,
126                                        uchar kernelsize,
                                          float factor) {
127    OriginalImg = src;
128    ProcessedImg.create(OriginalImg.size(), CV_8UC1);
```

```
129    AdaptiveContrastStretch(kernelsize, factor);
130    dst = ProcessedImg;
131  }
132
133  /*! Homebrew AdaptiveContrastStretch function which
          calculate the mean and
134  standard deviation from the neighboring pixels if the
          current pixel is higher
135  then the mean the value is incremented with an given factor
          multiplied with the
136  standard deviation, and decreased if it's lower then the
          mean.
137  \param kernelsize an uchar which represent the kernelsize
          should be an uneven
138  number higher than two
139  \param factor float which indicates the amount the effect
          should take place
140  standard value is 1.0 only used in the adaptive contrast
          stretch enhancement
141  \param chain use the results from the previous operation
          default value = false;
142  */
143  void Enhance::AdaptiveContrastStretch(uchar kernelsize,
          float factor,
144                                        bool chain) {
145    // Exception handling
146    EMPTY_CHECK(OriginalImg);
147    if (kernelsize < 3 || (kernelsize % 2) == 0) {
148      throw Exception::WrongKernelSizeException();
149    }
150    CV_Assert(OriginalImg.depth() != sizeof(uchar));
151
152    // Make the pointers to the Data
153    uchar *O;
154    CHAIN_PROCESS(chain, O, uchar);
155    uchar *P = ProcessedImg.data;
156
157    int i = 0;
158    int hKsize = kernelsize / 2;
159    int nCols = OriginalImg.cols;
160    int pStart = (hKsize * nCols) + hKsize + 1;
161
162    int nData = OriginalImg.rows * OriginalImg.cols;
163    int pEnd = nData - pStart;
164    uint32_t noNeighboursPix = kernelsize * kernelsize;
165    uint32_t sum;
166    float mean = 0.0;
167
168    uchar *nRow = GetNRow(nData, hKsize, nCols, OriginalImg.
          rows);
169
170    i = pStart;
171    while (i++ < pEnd) {
172      // Checks if pixel isn't a border pixel and progresses
            to the new row
173      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(O, i, hKsize, nCols, sum
            );
183      mean = (float)(sum / noNeighboursPix);
184      float Std = CalculateStdOfNeighboringPixels(O, 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
          should be an uneven
231 number higher than two
232 \param chain use the results from the previous operation
          default value = false;
233 */
234 void Enhance::Blur(uchar kernelsize, bool chain) {
235    // Exception handling
236    EMPTY_CHECK(OriginalImg);
237    if (kernelsize < 3 || (kernelsize % 2) == 0) {
238       throw Exception::WrongKernelSizeException();
239    }
240    CV_Assert(OriginalImg.depth() != sizeof(uchar));
241
242    // Make the pointers to the Data
243    uchar *O;
244    CHAIN_PROCESS(chain, O, uchar);
245    uchar *P = ProcessedImg.data;
246
247    int nData = OriginalImg.rows * OriginalImg.cols;
248    int hKsize = kernelsize / 2;
249    int nCols = OriginalImg.cols;
250    int pStart = (hKsize * nCols) + hKsize + 1;
251    int pEnd = nData - pStart;
252    int noNeighboursPix = kernelsize * kernelsize;
253    uint32_t sum;
254
255    int i;
256    uchar *nRow = GetNRow(nData, hKsize, nCols, OriginalImg.
          rows);
257    i = pStart;
258    while (i++ < pEnd) {
259       // Checks if pixel isn't a border pixel and progresses
             to the new row
260       if (nRow[i] == 1) {
261          i += kernelsize;
262       }
263
264       // Calculate the sum of the kernel
265       sum = 0;
266       CalculateSumOfNeighboringPixels(O, i, hKsize, nCols, sum
             );
267
268       P[i] = (uchar)(round(sum / noNeighboursPix));
269    }
270
271    delete[] nRow;
272 }
273
```

```
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 statics of the whole image
288    ucharStat_t imgStats(O, 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.0.19 Morphological filter Class

```cpp
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 }
```

```cpp
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) {
```

```cpp
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    Erosion(mask, chain);
73    Dilation(mask, true);
74  }
75
76  void MorphologicalFilter::Close(const Mat &mask, bool chain)
        {
77    Dilation(mask, chain);
78    Erosion(mask, true);
79  }
80
81  void MorphologicalFilter::Dilation(const Mat &mask, bool
      chain) {
82    Filter(mask, chain, 0, 1, 1);
83  }
84
85  void MorphologicalFilter::Erosion(const Mat &mask, bool
      chain) {
86    Filter(mask, chain, 1, 0, 0);
87  }
88
89  void MorphologicalFilter::Filter(const Mat &mask, bool chain
      , uchar startVal,
90                                   uchar newVal, uchar
                                      switchVal) {
91    // Exception handling
92    CV_Assert(OriginalImg.depth() != sizeof(uchar));
93    EMPTY_CHECK(OriginalImg);
94    if (mask.cols % 2 == 0 || mask.cols < 3) {
95      throw Exception::WrongKernelSizeException("Wrong
          Kernelsize columns!");
96    }
97    if (mask.rows % 2 == 0 || mask.rows < 3) {
98      throw Exception::WrongKernelSizeException("Wrong
          Kernelsize rows!");
99    }
100
101   uint32_t hKsizeCol = (mask.cols / 2);
102   uint32_t hKsizeRow = (mask.rows / 2);
103
104   // make Pointers
105   Mat workOrigImg(ProcessedImg.rows + mask.rows,
          ProcessedImg.cols + mask.cols,
106                  CV_8UC1);
107   workOrigImg.setTo(0);
108   if (chain) {
109     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.0.20 Segment Class

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

#pragma once

#include <vector>
#include <queue>
#include <string>
#include <stdint.h>
#include <iostream>
#include <algorithm>
#include <utility>

#include <boost/range/adaptor/reversed.hpp>

#include "opencv2/imgproc/imgproc.hpp"

#include "ImageProcessing.h"
#include "MorphologicalFilter.h"
#include "../SoilMath/SoilMath.h"

namespace Vision {
class Segment : public ImageProcessing {
public:
  /*! Coordinates for the region of interest*/
  typedef struct Rect {
    uint16_t leftX;  /*!< Left X coordinate*/
    uint16_t leftY;  /*!< Left Y coordinate*/
    uint16_t rightX; /*!< Right X coordinate*/
    uint16_t rightY; /*!< Right Y coordinate*/
    Rect(uint16_t lx, uint16_t ly, uint16_t rx, uint16_t ry)
        : leftX(lx), leftY(ly), rightX(rx), rightY(ry){}
  } Rect_t;

  typedef std::vector<Vision::Segment::Rect_t> RectList_t;

  /*! Individual blob*/
  typedef struct Blob {
    uint16_t Label; /*!< ID of the blob*/
    cv::Mat Img;    /*!< BW image of the blob all the pixel
         belonging to the blob
                     are set to 1 others are 0*/
    cv::Rect ROI;   /*!< Coordinates for the blob in the
         original picture as a
                     cv::Rect*/
    uint32_t Area; /*!< Calculated stats of the blob*/
    cv::Point_<double> Centroid;
    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
                        pixels, feeding it in
79                     an Neural Net */
80      GraphMinCut /*!< Segmentation using a graph function and
           the minimum cut */
81    };
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>> &
                                    adj, uint32_t nCols,
137                              uint32_t nRows, Connected conn);
138      void InvertAdjacencyList(std::vector<std::vector<uint16_t
            >> &adj,
139                                  std::vector<std::vector<uint16_t
                                        >> &adjInv);
140  };
141  }
```

```cpp
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  Segment
9  \brief Segmentation algorithms
10 With this class, various segmentation routines can be
       applied to a greyscale or
11 black and white source image.
12 */
13 #include "Segment.h"
14
15 namespace Vision {
16 //! Constructor of the Segmentation class
17 Segment::Segment() {}
18
19 //! Constructor of the Segmentation class
20 Segment::Segment(const Mat &src) {
21   OriginalImg = src;
22   ProcessedImg.create(OriginalImg.size(), CV_8UC1);
23   LabelledImg.create(OriginalImg.size(), CV_16UC1);
24 }
25
26 Segment::Segment(const Segment &rhs) {
27   this->BlobList = rhs.BlobList;
28   this->LabelledImg = rhs.LabelledImg;
29   this->MaxLabel = rhs.MaxLabel;
30   this->noOfFilteredBlobs = rhs.noOfFilteredBlobs;
31   this->OriginalImg = rhs.OriginalImg;
32   this->OriginalImgStats = rhs.OriginalImgStats;
33   this->ProcessedImg = rhs.ProcessedImg;
34   this->TempImg = rhs.TempImg;
35   this->ThresholdLevel = rhs.ThresholdLevel;
36 }
37
38 //! De-constructor
39 Segment::~Segment() {}
40
41 Segment &Segment::operator=(Segment &rhs) {
42   if (&rhs != this) {
43     this->BlobList = rhs.BlobList;
```

```cpp
44        this->LabelledImg = rhs.LabelledImg;
45        this->MaxLabel = rhs.MaxLabel;
46        this->noOfFilteredBlobs = rhs.noOfFilteredBlobs;
47        this->OriginalImg = rhs.OriginalImg;
48        this->OriginalImgStats = rhs.OriginalImgStats;
49        this->ProcessedImg = rhs.ProcessedImg;
50        this->TempImg = rhs.TempImg;
51        this->ThresholdLevel = rhs.ThresholdLevel;
52    }
53    return *this;
54 }
55
56 void Segment::LoadOriginalImg(const Mat &src) {
57    OriginalImg = src;
58    ProcessedImg.create(OriginalImg.size(), CV_8UC1);
59    LabelledImg.create(OriginalImg.size(), CV_16UC1);
60 }
61
62 /*! Determine the threshold level by iteration, between two
       distribution,
63 presumably back- and foreground. It works towards the
       average of the two
64 averages and finally sets the threshold with two time the
       standard deviation
65 from the mean of the set object
66 \param TypeObject is an enumerator indicating if the bright
       or the dark pixels
67 are the object and should be set to one
68 \return The threshold level as an uint8_t   */
69 uint8_t Segment::GetThresholdLevel(TypeOfObjects TypeObject)
       {
70    // Exception handling
71    EMPTY_CHECK(OriginalImg);
72    CV_Assert(OriginalImg.depth() != sizeof(uchar));
73
74    // Calculate the statistics of the whole picture
75    ucharStat_t OriginalImgStats(OriginalImg.data, OriginalImg
       .rows,
76                                 OriginalImg.cols);
77
78    // Sets the initial threshold with the mean of the total
       picture
79    pair<uchar, uchar> T;
80    T.first = (uchar)(OriginalImgStats.Mean + 0.5);
81    T.second = 0;
82
83    uchar Rstd = 0;
84    uchar Lstd = 0;
85    uchar Rmean = 0;
86    uchar Lmean = 0;
87
88    // Iterate till optimum Threshold is found between back- &
         foreground
89    while (T.first != T.second) {
90      // Gets an array of the left part of the histogram
91      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) +
              0.5);
113
114     Rmean = (uchar)(sRight.Mean + 0.5);
115     Lmean = (uchar)(sLeft.Mean + 0.5);
116     Rstd = (uchar)(sRight.Std + 0.5);
117     Lstd = (uchar)(sLeft.Std + 0.5);
118     delete[] Left;
119     delete[] Right;
120   }
121
122   // Assumes the pixel value of the sought object lies
          between 2 sigma
123   int val = 0;
124   switch (TypeObject) {
125   case Bright:
126     val = Rmean - (sigma * Rstd) - thresholdOffset;
127     if (val < 0) {
128       val = 0;
129     } else if (val > 255) {
130       val = 255;
131     }
132     T.first = (uchar)val;
133     break;
134   case Dark:
135     val = Lmean + (sigma * Lstd) + thresholdOffset;
136     if (val < 0) {
137       val = 0;
138     } else if (val > 255) {
139       val = 255;
140     }
141     T.first = (uchar)val;
142     break;
143   }
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 tiping 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
```

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

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

```
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        // itterate 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
               ]][0]);
362          if (route[nodesVisited] == route[nodesVisited + 1])
               {
363            route.pop_back();
364            lastNodeReached = true;
365          }
366        }
367        // Set all values to the lowest value
368        for (uint32_t k = 0; k < route.size(); k++) {
369          *route[k] = minVal;
370        }
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
          final value
379    for_each(P, P + nData, [&](uint16_t &V) { V = valueArr[V];
          });
380    delete[] valueArr;
381  }
382
383  /*! Create a BW image with only edges from a BW image
384  \param src source image as a const cv::Mat
385  \param dst destination image as a cv::Mat
386  \param conn set the pixel connection eight or four
387  \param chain use the results from the previous operation
        default value = false;
388  */
389  void Segment::GetEdges(const Mat &src, Mat &dst, bool chain,
        Connected conn) {
390    OriginalImg = src;
391    GetEdges(chain, conn);
```

```
392    dst = ProcessedImg;
393  }
394
395  /*! Create a BW image with only edges from a BW image
396  \param conn set the pixel connection eight or four
397  \param chain use the results from the previous operation
        default value = false;
398  */
399  void Segment::GetEdges(bool chain, Connected conn) {
400    // Exception handling
401    CV_Assert(OriginalImg.depth() != sizeof(uchar));
402    EMPTY_CHECK(OriginalImg);
403
404    // make Pointers
405    uchar *O;
406    CHAIN_PROCESS(chain, O, uchar);
407    uchar *P = ProcessedImg.data;
408
409    uint32_t nCols = OriginalImg.cols;
410    uint32_t nRows = OriginalImg.rows;
411    uint32_t nData = nCols * nRows;
412    uint32_t pEnd = nData + 1;
413    uint32_t i = 0;
414
415    // Loop through the image and set each pixel which has a
          zero neighbor set it
416    // to two.
417    if (conn == Four) {
418      // Loop through the picture
419      while (i < pEnd) {
420        // If current value = zero processed value = zero
421        if (O[i] == 0) {
422          P[i] = 0;
423        }
424        // If current value = 1 check North West, South and
              East and act
425        // accordingly
426        else if (O[i] == 1) {
427          uchar *nPixels = new uchar[4];
428          nPixels[0] = O[i - 1];
429          nPixels[1] = O[i - nCols];
430          nPixels[2] = O[i + 1];
431          nPixels[3] = O[i + nCols];
432
433          // Sort the neighbors for easier checking
434          SoilMath::Sort::QuickSort<uchar>(nPixels, 4);
435          if (nPixels[0] == 0) {
436            P[i] = 1;
437          } else {
438            P[i] = 0;
439          }
440          delete[] nPixels;
441        } else {
442          throw Exception::PixelValueOutOfBoundException();
443        }
444        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
                  East and act
454          // accordingly
455          else if (O[i] == 1) {
456            uchar *nPixels = new uchar[8];
457            nPixels[0] = O[i - 1];
458            nPixels[1] = O[i - nCols];
459            nPixels[2] = O[i - nCols - 1];
460            nPixels[3] = O[i - nCols + 1];
461            nPixels[4] = O[i + 1];
462            nPixels[5] = O[i + nCols + 1];
463            nPixels[6] = O[i + nCols];
464            nPixels[7] = O[i + nCols - 1];
465
466            // Sort the neighbors for easier checking
467            SoilMath::Sort::QuickSort<uchar>(nPixels, 8);
468
469            if (nPixels[0] == 0) {
470              P[i] = 1;
471            } else {
472              P[i] = 0;
473            }
474            delete[] nPixels;
475          } else {
476            throw Exception::PixelValueOutOfBoundException();
477          }
478          i++;
479        }
480      }
481  }
482
483  void Segment::GetEdgesEroding(bool chain) {
484    // Exception handling
485    CV_Assert(OriginalImg.depth() != sizeof(uchar));
486    EMPTY_CHECK(OriginalImg);
487
488    // make Pointers
489    uchar *O;
490    CHAIN_PROCESS(chain, O, uchar);
491    uchar *P = ProcessedImg.data;
492
493    uint32_t nCols = OriginalImg.cols;
494    uint32_t nRows = OriginalImg.rows;
495    uint32_t nData = nCols * nRows;
496
497    // Setup the erosion
498    MorphologicalFilter eroder;
499    if (chain) {
```

```
500      eroder.OriginalImg = TempImg;
501    } else {
502      eroder.OriginalImg = OriginalImg;
503    }
504    // Setup the processed image of the eroder
505    eroder.ProcessedImg.create(OriginalImg.size(), CV_8UC1);
506    eroder.ProcessedImg.setTo(0);
507    // Setup the mask
508    Mat mask(3, 3, CV_8UC1, 1);
509    // Erode the image
510    eroder.Erosion(mask, false);
511
512    // Loop through the image and set the not eroded pixels to
           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,
           LabelledImg.cols,
552                                    LabelledImg.rows, MaxLabel + 1, 0,
                                       MaxLabel);
553
554    BlobList.reserve(LabelStats.EndBin);
555    rectList.reserve(LabelStats.EndBin);
556
557    BlobList.push_back(Blob_t(0, 0));
558    rectList.push_back(Rect_t(0, 0, 0, 0));
559
560    for (uint32_t i = 1; i < LabelStats.EndBin; i++) {
561      BlobList.push_back(Blob_t(i, LabelStats.bins[i]));
562      rectList.push_back(Rect_t(nCols, nRows, 0, 0));
563    }
564
565    // make Pointers
566    uint16_t *L = (uint16_t *)LabelledImg.data;
567
568    uint32_t currentX, currentY;
569    // uint16_t leftX, leftY, rightX, rightY;
570    // Loop through the labeled image and extract the Blobs
571    for (uint32_t i = 0; i < nData; i++) {
572      if (L[i] != 0) {
573        /* Determine the current x and y value of the current
             blob and
574        checks if it is min/max */
575        currentY = i / nCols;
576        currentX = i % nCols;
577
578        // Min value
579        if (currentX < rectList[L[i]].leftX) {
580          rectList[L[i]].leftX = currentX;
581        }
582        if (currentY < rectList[L[i]].leftY) {
583          rectList[L[i]].leftY = currentY;
584        }
585
586        // Max value
587        if (currentX > rectList[L[i]].rightX) {
588          rectList[L[i]].rightX = currentX;
589        }
590        if (currentY > rectList[L[i]].rightY) {
591          rectList[L[i]].rightY = currentY;
592        }
593      }
594    }
595
596    // Loop through the BlobList and finalize it
597    uint8_t *LUT_filter = new uint8_t[MaxLabel + 1]{};
598    for (uint32_t i = 1; i <= MaxLabel; i++) {
599      LUT_filter[i] = 1;
600      BlobList[i].ROI.y = rectList[i].leftY;
601      BlobList[i].ROI.x = rectList[i].leftX;
```

```cpp
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 *O;
622      CHAIN_PROCESS(chain, O, 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
           differently
705    uchar *nRow = new uchar[nData]{};
706    for (uint32_t i = nCols; i < nData; i += nCols) {
707      nRow[i] = 1;
708      nRow[i - 1] = 2;
709    }
710
711    // Set the first pixel
712    if (O[0] == 0) {
713      P[0] = 0;
714    } else if (O[0] == 1) {
715      P[0] = 1;
716    } else {
717      throw Exception::PixelValueOutOfBoundException();
718    }
719
720    // Walk through the toprow and determine if it's a new
           blob or it's connected
721    // with previously determine blob
722    for (uint32_t i = 1; i < nCols; i++) {
723      if (O[i] == 0) {
724        P[i] = 0;
725      } else if (O[i] == 1) {
726        // If West is zero assume this is a new blob
727        if (P[i - 1] == 0) {
728          P[i] = ++currentlbl;
729          vector<uint16_t> cVector;
730          cVector.push_back(currentlbl);
731          adj.push_back(cVector);
732        } else { // set as previous blob
733          P[i] = P[i - 1];
734        }
735      } else { // Value of of bounds
736        throw Exception::PixelValueOutOfBoundException();
737      }
738    }
739
740    // walk through each pixel and determine if it's a new
           blob or it's connected
741    // with previously determine blob
742    for (uint32_t i = OriginalImg.cols; i < nData; i++) {
743      if (O[i] == 0) { // Original pixel = 0
744        P[i] = 0;
745      } else if (O[i] == 1) {
746        // Get an array of Neighboring Pixels
747        uint16_t *nPixels = new uint16_t[noConn[nRow[i]]];
748        if (nRow[i] != 1) {
749          nPixels[0] = P[i - 1];
```

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

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

```
                                              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.0.21 General project files

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

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

```
1   /* Copyright (C) Jelle Spijker - All Rights Reserved
2    * Unauthorized copying of this file, via any medium is
         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   // Debuging 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        : msg(m){};
26    ~ChannelMismatchException() _GLIBCXX_USE_NOEXCEPT{};
27    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
         msg.c_str(); };
28
29  private:
30    string msg;
31  };
32  }
33  }
```

```
 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
```

```cpp
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  }
```

```cpp
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 EmtpyImageException
9   Exception class which is thrown when operations are about to
        start on a empty
10  image.
11  */
12
13  #pragma once
14
15  #include <exception>
16  #include <string>
17
18  using namespace std;
19
20  namespace Vision {
21  namespace Exception {
22  class EmtpyImageException : public std::exception {
23  public:
24    EmtpyImageException(string m = "Empty Image!") : msg(m){};
25    ~EmtpyImageException() _GLIBCXX_USE_NOEXCEPT{};
26    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
        msg.c_str(); };
27
28  private:
29    string msg;
30  };
31  }
32  }
```

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

/*! \class PixelValueOutOfBoundException
Exception class which is thrown when an unexpected pixel
    value has to be
computed
*/
#pragma once

#include <exception>
#include <string>

using namespace std;

namespace Vision {
namespace Exception {
class PixelValueOutOfBoundException : public std::exception
    {
public:
  PixelValueOutOfBoundException(string m = "Current pixel
      value out of bounds!")
      : msg(m){};
  ~PixelValueOutOfBoundException() _GLIBCXX_USE_NOEXCEPT{};
  const char *what() const _GLIBCXX_USE_NOEXCEPT { return
      msg.c_str(); };

private:
  string msg;
};
}
}
```

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

/*! \class WrongKernelSizeException
Exception class which is thrown when a wrong kernelsize is
    requested
*/
#pragma once

#include <exception>
```

```cpp
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
          dimensions!") : msg(m){};
23    ~WrongKernelSizeException() _GLIBCXX_USE_NOEXCEPT{};
24    const char *what() const _GLIBCXX_USE_NOEXCEPT { return
          msg.c_str(); };
25
26  private:
27    string msg;
28  };
29  }
30  }
```

F. Analyzer Library

F.0.22 Analyzer Class

```cpp
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 STARTING_ESTIMATE_PROGRESS 300
10 #ifndef DEBUG
11 //#define DEBUG
12 #endif
13
14 #include <opencv2/core.hpp>
15 #include <opencv2/imgproc.hpp>
16 #include <vector>
17 #include <cmath>
18
19 #include "sample.h"
20 #include "soilsettings.h"
21 #include "soilanalyzerexception.h"
22
23 #include "SoilMath.h"
24
25 #include <QtCore/QObject>
26 #include <QThread>
27 #include <QtConcurrent>
28
29 #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);
52
53     void Analyse();
54     void Analyse(Images_t *snapshots, Sample *results,
55         SoilSettings *settings);
55     float CalibrateSI(float SI, cv::Mat &img);
56
57     uint32_t MaxProgress = STARTING_ESTIMATE_PROGRESS; /*!< */
58
59     SoilMath::NN NeuralNet; /*!< */
60
61   signals:
62     void on_progressUpdate(int value);     /*!< */
63     void on_maxProgressUpdate(int value); /*!< */
64     void on_AnalysisFinished();            /*!< */
65
66   private:
67     uint32_t currentProgress = 0;    /*!< */
68     uint32_t currentParticleID = 0; /*!< */
69     double BinRanges[15]{0.0,   0.038, 0.045, 0.063, 0.075,
70         0.09, 0.125, 0.18,
70                           0.25, 0.355, 0.5,   0.71,  1.0,   1.4,
71                           2.0};
71
72     SoilMath::FFT fft; /*!< */
73
74     void CalcMaxProgress();
75     void CalcMaxProgressAnalyze();
76     void PrepImages();
77     void GetBW(std::vector<cv::Mat> &images, std::vector<cv::
78         Mat> &BWvector);
78     void GetBW(cv::Mat &img, cv::Mat &BW);
79
80     void GetEnhancedInt(Images_t *snapshots,
```

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

```
 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 "analyzer.h"
 9
10 namespace SoilAnalyzer {
11
12 /*!
13 *\brief Analyzer::Analyzer
14 *\param snapshots
15 *\param results
16 *\param settings
17 */
18 Analyzer::Analyzer(Images_t *snapshots, Sample *results,
19                    SoilSettings *settings = nullptr) {
20   this->Snapshots = snapshots;
21   this->Results = results;
22   if (settings == nullptr) {
23     Settings = new SoilSettings;
24   } else {
25     this->Settings = settings;
26   }
27   NeuralNet.LoadState(Settings->NNlocation);
28 }
29
```

```cpp
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,
                                                      EXCEPTION_NO_SNAPSHOTS_NR
                                                        );
37    }
38
39    std::vector<cv::Mat> intensityVector;
40    GetEnhancedInt(Snapshots, intensityVector);
41
42    std::vector<cv::Mat> BWVector;
43    GetBW(intensityVector, BWVector);
44    CleanUpMatVector(intensityVector);
45
46    GetParticles(BWVector, Snapshots, Results->
          ParticlePopulation);
47
48    CleanUpMatVector(BWVector);
49    CleanUpMatVector(Snapshots);
50
51    Results->isPreparedForAnalysis = true;
52  }
53
54  void Analyzer::Analyse(Images_t *snapshots, Sample *results,
55                         SoilSettings *settings) {
56    Snapshots = snapshots;
57    Results = results;
58    Settings = settings;
59    Analyse();
60  }
61
62  /*!
63   * \brief Analyzer::Analyse
64   */
65  void Analyzer::Analyse() {
66    CalcMaxProgress();
67    if (!Results->isPreparedForAnalysis && !Results->
          IsLoadedFromDisk) {
68      PrepImages();
69    }
70    GetFFD(Results->ParticlePopulation);
71    if (PredictShape && Settings->PredictTheShape) {
72      GetPrediction(Results->ParticlePopulation);
73    }
74
75    Results->Angularity =
76        ucharStat_t(Results->GetAngularityVector()->data(),
77                    Results->GetAngularityVector()->size(), 1,
                          7, 0, true);
78    emit on_progressUpdate(currentProgress++);
79
80    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(),
                         BinRanges, 15, 14);
87   emit on_progressUpdate(currentProgress++);
88
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   mv.clear();
95 }
96
97 /*!
98  * \brief Analyzer::CleanUpMatVector
99  * \param mv
100  */
101 void Analyzer::CleanUpMatVector(Images_t *mv) {
102   for_each(mv->begin(), mv->end(), [](Image_t &I) {
103     I.BackLight.release();
104     I.FrontLight.release();
105   });
106   mv->clear();
107 }
108
109 /*!
110  * \brief Analyzer::CalcMaxProgress
111  */
112 void Analyzer::CalcMaxProgress() {
113   // Static processing steps
114   MaxProgress += Snapshots->size() * 5;
115
116   // Optional processing steps
117   if (Settings->useBlur) {
118     MaxProgress += Snapshots->size();
119   }
120   if (Settings->useAdaptiveContrast) {
121     MaxProgress += Snapshots->size();
122   }
123   if (Settings->fillHoles) {
124     MaxProgress += Snapshots->size();
125   }
126   if (Settings->ignorePartialBorderParticles) {
127     MaxProgress += Snapshots->size();
128   }
129   if (Settings->morphFilterType != Vision::
        MorphologicalFilter::NONE) {
130     MaxProgress += Snapshots->size();
131   }
132
```

```cpp
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> &
                                   intensityVector) {
150    if (Settings->useBacklightProjection) {
151      for_each(snapshots->begin(), snapshots->end(), [&](
           Image_t &I) {
152        cv::Mat intensity;
153        GetEnhancedInt(I.BackLight, intensity);
154        intensityVector.push_back(intensity);
155      });
156    } else {
157      for_each(snapshots->begin(), snapshots->end(), [&](
            Image_t &I) {
158        cv::Mat intensity;
159        GetEnhancedInt(I.FrontLight, intensity);
160        intensityVector.push_back(intensity);
161      });
162    }
163  }
164
165  /*!
166   * \brief Analyzer::GetEnhancedInt
167   * \param img
168   * \param intensity
169   */
170  void Analyzer::GetEnhancedInt(Mat &img, Mat &intensity) {
171    Vision::Conversion IntConvertor(img.clone());
172    IntConvertor.Convert(Vision::Conversion::RGB, Vision::
          Conversion::Intensity);
173    emit on_progressUpdate(currentProgress++);
174    SHOW_DEBUG_IMG(IntConvertor.ProcessedImg, uchar, 255, "RGB
          2 Int", false);
175
176    if (Settings->useBlur) {
177      Vision::Enhance IntBlur(IntConvertor.ProcessedImg.clone
            ());
178      IntBlur.Blur(Settings->blurKernelSize);
179      emit on_progressUpdate(currentProgress++);
180      uint32_t HBK = Settings->blurKernelSize / 2;
181      uint32_t BK = Settings->blurKernelSize - 1;
182      if (Settings->useAdaptiveContrast) {
```

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

```cpp
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",
         true);
243
244    cv::Mat BWholes;
245    if (Settings->fillHoles) {
246      Vision::Segment Fillholes(SegBL.ProcessedImg);
247      Fillholes.FillHoles();
248      BWholes = Fillholes.ProcessedImg;
249      emit on_progressUpdate(currentProgress++);
250      SHOW_DEBUG_IMG(BWholes, uchar, 255, "Fillholes", true);
251    } else {
252      BWholes = SegBL.ProcessedImg;
253    }
254
255    cv::Mat BWborder;
256    if (Settings->ignorePartialBorderParticles) {
257      Vision::Segment RemoveBB(BWholes.clone());
258      RemoveBB.RemoveBorderBlobs();
259      BWborder = RemoveBB.ProcessedImg;
260      emit on_progressUpdate(currentProgress++);
261      SHOW_DEBUG_IMG(BWborder, uchar, 255, "RemoveBorderBlobs"
           , true);
262    } else {
263      BWborder = BWholes;
264    }
265
266    if (Settings->morphFilterType != Vision::
         MorphologicalFilter::NONE) {
267      Vision::MorphologicalFilter Morph(BWborder.clone());
268      cv::Mat kernel = cv::Mat::zeros(Settings->filterMaskSize
           ,
269                                       Settings->filterMaskSize
                                           , CV_8UC1);
270      uint32_t hMaskSize = Settings->filterMaskSize / 2;
271      cv::circle(kernel, cv::Point(hMaskSize, hMaskSize),
           hMaskSize + 1, 1, -1);
272      switch (Settings->morphFilterType) {
273      case Vision::MorphologicalFilter::CLOSE:
274        Morph.Close(kernel);
275        break;
276      case Vision::MorphologicalFilter::OPEN:
277        Morph.Open(kernel);
278        break;
279      case Vision::MorphologicalFilter::DILATE:
280        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"
             , true);
292    } else {
293      BW = BWholes;
294    }
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 *
         snapshots,
304                              Particle::ParticleVector_t &
                                  partPopulation) {
305    for (uint32_t i = 0; i < snapshots->size(); i++) {
306      Vision::Segment prepBW(BW[i]);
307      prepBW.GetBlobList();
308      emit on_progressUpdate(currentProgress++);
309      GetParticlesFromBlobList(prepBW.BlobList, &(snapshots->
           at(i)),
310                               partPopulation);
311      emit on_progressUpdate(currentProgress++);
312    }
313  }
314
315  /*!
316   * \brief Analyzer::GetParticlesFromBlobList
317   * \param bloblist
318   * \param snapshot
319   * \param edge
320   * \param partPopulation
321   */
322  void Analyzer::GetParticlesFromBlobList(
323      Vision::Segment::BlobList_t &bloblist, Image_t *snapshot
           ,
324      Particle::ParticleVector_t &partPopulation) {
325    for_each(bloblist.begin(), bloblist.end(), [&](Vision::
         Segment::Blob_t &B) {
326      Particle part;
327      part.ID = currentParticleID++;
328      part.PixelArea = B.Area;
329      Vision::Segment::getOrientented(B.Img, B.Centroid, B.
           Theta,
```

```
330                                                   part.Eccentricty);
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.FFDescriptors = 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.0.23 Sample Class

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

#pragma once

#include "stdint.h"
#include <vector>
#include <string>
#include "Stats.h"
#include "psd.h"
#include "particle.h"
#include <fstream>
#include <boost/archive/binary_iarchive.hpp>
#include <boost/archive/binary_oarchive.hpp>
#include <boost/serialization/string.hpp>
#include <boost/serialization/version.hpp>
#include <boost/serialization/vector.hpp>
#include <boost/iostreams/filter/zlib.hpp>
#include <boost/iostreams/filtering_streambuf.hpp>
#include "zlib.h"
#include "soilanalyzertypes.h"

namespace SoilAnalyzer {
class Sample {
public:
  Sample();

  uint32_t ID;         /*!< The sample ID*/
  std::string Location; /*!< The Location where the sample
      was taken*/
  double Longtitude = 4.629618299999947;
  double Latitude = 51.8849149;
  double Depth = 0;
  std::string Date = "01-09-2015";
  std::string Name; /*!< The sample name identifier*/

  Particle::ParticleVector_t
      ParticlePopulation; /*!< the individual particles of
          the sample*/

  SoilMath::PSD PSD; /*!< The Particle Size Distribution*/
  ucharStat_t Roundness;
  ucharStat_t Angularity;
  floatStat_t RI; /*!< The statistical Redness Index data*/

  void Save(const std::string &filename);
  void Load(const std::string &filename);

```

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

```cpp
105         ar &RoundnessGathered;
106         ar &AngularityGathered;
107         ar &IsLoadedFromDisk;
108         if (version > 0) {
109             ar &Longtitude;
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             Longtitude = 4.629618299999947;
123             Date = "01-10-2015";
124             Depth = 0;
125             CIELab_aVec = Particle::doubleVector_t();
126             CIELab_aGathered = false;
127             CIELab_bVec = Particle::doubleVector_t();
128             CIELab_bGathered = false;
129             ColorChange = false;
130         }
131     }
132 };
133 }
134 BOOST_CLASS_VERSION(SoilAnalyzer::Sample, 1)
```

```cpp
 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 "sample.h"
 9 #include "particle.h"
10
11 namespace SoilAnalyzer {
12 namespace io = boost::iostreams;
13
14 /*!
15  * \brief Sample::Sample
16  */
17 Sample::Sample() {}
18
19 /*!
20  * \brief Sample::Save
21  * \param filename
22  */
23 void Sample::Save(const std::string &filename) {
```

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

F.0.24 Particle Class

```cpp
/* Copyright (C) Jelle Spijker - All Rights Reserved
 * Unauthorized copying of this file, via any medium is
     strictly prohibited
 * and only allowed with the written consent of the author (
     Jelle Spijker)
 * This software is proprietary and confidential
 * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
 */
#pragma once

#include <opencv2/core.hpp>
#include <stdint.h>
#include <vector>
#include "SoilMath.h"
#include <fstream>
#include <boost/archive/binary_iarchive.hpp>
#include <boost/archive/binary_oarchive.hpp>
#include <boost/serialization/string.hpp>
#include <boost/serialization/version.hpp>
#include <boost/serialization/vector.hpp>
#include <boost/iostreams/filter/zlib.hpp>
#include <boost/iostreams/filtering_streambuf.hpp>
#include "zlib.h"
#include "soilanalyzerexception.h"
#include "lab_t_archive.h"
#include "soilanalyzertypes.h"
#include "Vision.h"

namespace SoilAnalyzer {
class Particle {
public:
  typedef std::vector<Particle>
      ParticleVector_t; /*!< a vector consisting of
          individual particles*/
  typedef std::vector<double> PSDVector_t; /*!< a vector
      used in the PSD*/
  typedef std::vector<uint8_t>
      ClassVector_t; /*!< a vector used in the
          classification histogram*/
  typedef std::vector<float> floatVector_t;
  typedef std::vector<double> doubleVector_t;

  Particle();

  uint32_t ID; /*!< The particle ID*/

  cv::Mat BW;   /*!< The binary image of the particle*/
  cv::Mat Edge; /*!< The binary edge image of the particle*/
  cv::Mat RGB;  /*!< The RGB image of the particle*/

  Point_t Centroid = {0, 0};
  std::vector<Complex_t> FFDescriptors; /*!< The Fast
      Fourier Descriptors
```

```
48                                                      describing the
                                                           contour in the
49                                                      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 Eccentricty = 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 &Eccentricty;
102        } else {
103            isSmall = false;
104            SIDiameter = GetSiDiameter();
105            Centroid.x = 0;
106            Centroid.y = 0;
107            Eccentricty = 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)
```

```cpp
 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 "particle.h"
 9
10 namespace SoilAnalyzer {
11 namespace io = boost::iostreams;
12
13 Particle::Particle() {}
14
15 /*!
16  * \brief Particle::Save
17  * \param filename
18  */
19 void Particle::Save(const std::string &filename) {
20   std::ofstream ofs(filename.c_str(), std::ios::out | std::
        ios::binary);
21   {
22     io::filtering_streambuf<io::output> out;
23
24     out.push(io::zlib_compressor(io::zlib::best_compression)
          );
25     out.push(ofs);
26     {
```

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

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

```
124     }
125     if (meanRI == 0) {
126       if (LAB.empty()) {
127         getLabImg();
128       }
129       Vision::Conversion convertor(LAB);
130       convertor.Convert(Vision::Conversion::CIE_lab, Vision::
              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.0.25 Settings Class

```cpp
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
```

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

```cpp
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       ar &BOOST_SERIALIZATION_NVP(adaptContrastKernelSize);
114       ar &BOOST_SERIALIZATION_NVP(useBlur);
115       ar &BOOST_SERIALIZATION_NVP(blurKernelSize);
116       ar &BOOST_SERIALIZATION_NVP(typeOfObjectsSegmented);
117       ar &BOOST_SERIALIZATION_NVP(
              ignorePartialBorderParticles);
118       ar &BOOST_SERIALIZATION_NVP(fillHoles);
119       ar &BOOST_SERIALIZATION_NVP(sigmaFactor);
120       ar &BOOST_SERIALIZATION_NVP(morphFilterType);
121       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.0.26 General project files

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

```
54
55  unix:!macx: LIBS += -L$$PWD/../../build/install/ -
        lSoilVision
56  INCLUDEPATH += $$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
        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 <boost/archive/binary_iarchive.hpp>
11  #include <boost/archive/binary_oarchive.hpp>
12  #include <boost/serialization/access.hpp>
13  #include "soilanalyzertypes.h"
14
15  namespace boost {
16  namespace serialization {
17  /*!
18   * \brief serialize Serialize the openCV mat to disk
19   */
20  template <class Archive>
21  inline void serialize(Archive &ar, SoilAnalyzer::Lab_t &P,
        const unsigned int version __attribute__((unused))) {
22    ar &P.L;
23    ar &P.a;
24    ar &P.b;
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   #define EXCEPTION_PARTICLE_NOT_ANALYZED "Particle not
        analyzed Exception!"
8   #define EXCEPTION_PARTICLE_NOT_ANALYZED_NR 0
9   #define EXCEPTION_NO_SNAPSHOTS "No snapshots Exception!"
10  #define EXCEPTION_NO_SNAPSHOTS_NR 1
```

```cpp
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  }
```

```cpp
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
```

```
 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
         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 <QWidget>
10  #include <QImage>
11  #include <qopencvqt.h>
12  #include <QColor>
13  #include <QWheelEvent>
14
15  #include "soilanalyzer.h"
16
17  namespace Ui {
18    class QParticleDisplay;
19  }
20
21  class QParticleDisplay : public QWidget
22  {
23    Q_OBJECT
24
25  public:
26    explicit QParticleDisplay(QWidget *parent = 0);
27    ~QParticleDisplay();
28    void SetSample(SoilAnalyzer::Sample *sample);
29    SoilAnalyzer::Particle *SelectedParticle;
30    void wheelEvent( QWheelEvent * event );
31    void next();
```

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

```
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 "qparticledisplay.h"
9   #include "ui_qparticledisplay.h"
10
11  QParticleDisplay::QParticleDisplay(QWidget *parent)
12      : QWidget(parent), ui(new Ui::QParticleDisplay) {
13    ui->setupUi(this);
14    ui->widget->setBackgroundColor(QColor("white"));
15    ui->widget->setSlideSize(QSize(230, 230));
16    connect(ui->widget, SIGNAL(centerIndexChanged(int)), this,
17            SLOT(on_selectedParticleChangedWidget(int)));
18    connect(ui->horizontalSlider, SIGNAL(valueChanged(int)),
         this,
19            SLOT(on_selectedParticleChangedSlider(int)));
20  }
21
22  QParticleDisplay::~QParticleDisplay() {
23    for (uint32_t i = 0; i < ui->widget->slideCount(); i++) {
24      ui->widget->removeSlide(0);
25    }
26    delete ui->widget;
27    delete ui;
28  }
29
30  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->
       ParticlePopulation.begin() +
98                                   ui->widget->centerIndex()
                                       );
99   ui->widget->removeSlide(ui->widget->centerIndex());
100  ui->horizontalSlider->setMaximum(this->Sample->
       ParticlePopulation.size() - 1);
101  Sample->ParticleChangedStatePSD = true;
102  Sample->ParticleChangedStateAngularity = true;
103  Sample->ParticleChangedStateRoundness = true;
104  Sample->ChangesSinceLastSave = true;
105  Sample->ColorChange = true;
106  SelectedParticle = &Sample->ParticlePopulation[ui->widget
       ->centerIndex()];
107  emit particleDeleted();
108 }
109
110 void QParticleDisplay::on_selectedParticleChangedWidget(int
     value) {
111   if (!dontDoIt) {
112     dontDoIt = true;
113     ui->horizontalSlider->setValue(value);
114     SelectedParticle = &Sample->ParticlePopulation[ui->
          widget->centerIndex()];
115     QString volume;
116     volume.sprintf("%+06.2f", SelectedParticle->
          GetSiDiameter());
117     ui->label_Volume->setText(volume);
118     emit particleChanged(value);
119     emit shapeClassificationChanged(SelectedParticle->
          Classification.Category);
120     dontDoIt = false;
121   }
122 }
123
124 void QParticleDisplay::on_selectedParticleChangedSlider(int
     value) {
125   if (!dontDoIt) {
126     dontDoIt = true;
127     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  }
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 multimediawidgets
11
12  @
13  CONFIG(release, debug|release):DEFINES += QT_NO_DEBUG_OUTPUT
14  @
15
16  unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../build/install/
17
18  TARGET = QReportGenerator
19  TEMPLATE = lib
20  CONFIG += shared
21  VERSION = 0.1.00
22
23  SOURCES += \
24      qreportgenerator.cpp \
25      ../qcustomplot/examples/text-document-integration/
          qcpdocumentobject.cpp
26
27  HEADERS  += \
28      qreportgenerator.h \
29      ../qcustomplot/examples/text-document-integration/
          qcpdocumentobject.h
```

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

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

```cpp
14
15  #include "soilanalyzer.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;
```

```cpp
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
```

```cpp
1  #include "qreportgenerator.h"
2  #include "ui_qreportgenerator.h"
3
4  QReportGenerator::QReportGenerator(QWidget *parent,
5                                     SoilAnalyzer::Sample *
                                          sample,
6                                     SoilAnalyzer::
                                          SoilSettings *settings
                                          ,
7                                     QCustomPlot *psd,
                                          QCustomPlot *roundness
                                          ,
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),
          GtxtFormat);
96    rCurs.movePosition(QTextCursor::NextCell);
97
98    rCurs.insertText("Location:", GFieldtxtFormat);
99    rCurs.movePosition(QTextCursor::NextCell);
100   rCurs.insertText(QString::number(Sample->Latitude),
          GtxtFormat);
101   rCurs.insertText(", ", GtxtFormat);
102   rCurs.insertText(QString::number(Sample->Longtitude),
          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),
          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->Longtitude);
114
115   // Setup the QCustomplot handler
116   QCPDocumentObject *plotObjectHandler = new
          QCPDocumentObject(this);
117   ui->textEdit->document()->documentLayout()->
          registerHandler(
118    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,
          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),
          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),
          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),
           GtxtFormat);
140    rCurs.movePosition(QTextCursor::NextCell);
141
142    rCurs.insertText("Maximum: ", GFieldtxtFormat);
143    rCurs.movePosition(QTextCursor::NextCell);
144    rCurs.insertText(QString::number(Sample->PSD.max),
           GtxtFormat);
145    rCurs.movePosition(QTextCursor::NextCell);
146
147    rCurs.insertText("Range: ", GFieldtxtFormat);
148    rCurs.movePosition(QTextCursor::NextCell);
149    rCurs.insertText(QString::number(Sample->PSD.Range),
           GtxtFormat);
150    rCurs.movePosition(QTextCursor::NextCell);
151
152    rCurs.insertText("Standard deviation: ", GFieldtxtFormat);
153    rCurs.movePosition(QTextCursor::NextCell);
154    rCurs.insertText(QString::number(Sample->PSD.Std),
           GtxtFormat);
155    rCurs.movePosition(QTextCursor::NextBlock);
156
157    // Setup the PSD
158    rCurs.insertBlock(ImageGraphFormat);
159    rCurs.insertText(QString(QChar::ObjectReplacementCharacter
           ),
160                    QCPDocumentObject::generatePlotFormat(PSD
                        , 600, 350));
161
162    rCurs.insertBlock(ImageGraphFormat);
163    QTextTable *PSDdata = rCurs.insertTable(16, 3,
           GeneralTextTableFormat);
164    rCurs.insertText("Mesh Size [mm]", GFieldtxtFormat);
165    rCurs.movePosition(QTextCursor::NextCell);
166    rCurs.insertText("Cummulatief [%]", GFieldtxtFormat);
167    rCurs.movePosition(QTextCursor::NextCell);
168    rCurs.insertText("Retained [-]", GFieldtxtFormat);
169    rCurs.movePosition(QTextCursor::NextCell);
170    rCurs.insertText("2", GFieldtxtFormat);
171    rCurs.movePosition(QTextCursor::NextCell);
172    rCurs.insertText(QString::number(Sample->PSD.CFD[14]),
           GtxtFormat);
173    rCurs.movePosition(QTextCursor::NextCell);
174    rCurs.insertText(QString::number(Sample->PSD.bins[14]),
           GtxtFormat);
175    rCurs.movePosition(QTextCursor::NextCell);
176    rCurs.insertText("1.4", GFieldtxtFormat);
177    rCurs.movePosition(QTextCursor::NextCell);
178    rCurs.insertText(QString::number(Sample->PSD.CFD[13]),
           GtxtFormat);
179    rCurs.movePosition(QTextCursor::NextCell);
180    rCurs.insertText(QString::number(Sample->PSD.bins[13]),
```

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

```
357   void QReportGenerator::getLocationMap(double &latitude,
        double &longtitude) {
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(longtitude));
365     locationURL.append("&zoom=17&size=600x750&maptype=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(longtitude));
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)"));
```

```cpp
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),
419        tr("Report (*.pdf)"));
420    if (!fn.isEmpty()) {
421      if (!fn.contains(tr(".pdf"))) {
422        fn.append(tr(".pdf"));
423      }
424      QPrinter printer;
425      printer.setOutputFormat(QPrinter::PdfFormat);
426      printer.setOutputFileName(fn);
427      Report->print(&printer);
428    }
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    CIElabPlot->graph(0)
450        ->setScatterStyle(QCPScatterStyle(QCPScatterStyle::
             ssCircle, 8));
451    CIElabPlot->graph(0)->setPen(binPen);
452    CIElabPlot->graph(0)->setName("a* vs. b*");
453    CIElabPlot->graph(0)->setData(*Sample->GetCIELab_bVector()
         , *Sample->GetCIELab_aVector());
454    CIElabPlot->graph(0)->setScatterStyle(QCPScatterStyle::
         ssCross);
455    CIElabPlot->graph(0)->setLineStyle(QCPGraph::lsNone);
```

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

K. Vision Soil Analyzer Program

K.0.27 General project files

```
1   #-------------------------------------------------
2   #
3   # Project created by QtCreator 2015-08-07T16:50:24
4   #
5   #
6   #-------------------------------------------------
7
8   QT       += core gui concurrent
9   QMAKE_CXXFLAGS += -std=c++11
10
11  greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport
        multimedia multimediawidgets
12
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    += vsamainwindow.ui \
33      dialogsettings.ui \
34      dialognn.ui
35
36  #opencv
37  LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui -
        lopencv_imgcodecs
38  INCLUDEPATH += /usr/local/include/opencv
39  INCLUDEPATH += /usr/local/include
40
41  #boost
42  DEFINES += BOOST_ALL_DYN_LINK
43  INCLUDEPATH += /usr/include/boost
44  LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_filesystem -
        lboost_serialization -lboost_system -lboost_iostreams
45
46  #SoilMath lib
47  unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
48  INCLUDEPATH += $$PWD/../SoilMath
49  DEPENDPATH += $$PWD/../SoilMath
50
51  #SoilHardware lib
52  unix:!macx: LIBS += -L$$PWD/../../build/install/ -
        lSoilHardware
53  INCLUDEPATH += $$PWD/../SoilHardware
54  DEPENDPATH += $$PWD/../SoilHardware
55
56  #SoilVision lib
57  unix:!macx: LIBS += -L$$PWD/../../build/install/ -
        lSoilVision
58  INCLUDEPATH += $$PWD/../SoilVision
59  DEPENDPATH += $$PWD/../SoilVision
60
61  #QCustomplot lib
62  DEFINES += QCUSTOMPLOT_USE_LIBRARY
63  unix:!macx: LIBS += -L$$PWD/../../build/install/ -
        lqcustomplot
64  INCLUDEPATH += $$PWD/../qcustomplot
65  DEPENDPATH += $$PWD/../qcustomplot
66
67  #QParticleSelector
68  unix:!macx: LIBS += -L$$PWD/../../build/install/ -
        lQParticleSelector
69  INCLUDEPATH += $$PWD/../QParticleSelector
70  DEPENDPATH += $$PWD/../QParticleSelector
71
72  #QParticleDisplay
73  unix:!macx: LIBS += -L$$PWD/../../build/install/ -
        lQParticleDisplay
74  INCLUDEPATH += $$PWD/../QParticleDisplay
75  DEPENDPATH += $$PWD/../QParticleDisplay
76
77  #QOpenCVQT
78  unix:!macx: LIBS += -L$$PWD/../../build/install/ -lQOpenCVQT
79  INCLUDEPATH += $$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.0.28 Main window Class

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

```cpp
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          if (*e.id() == EXCEPTION_NOCAMS_NR) {
47            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 progresbar 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
```

```cpp
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,
                                    ui->QPlot_Roudness->yAxis2);
150
151    ui->QPlot_Roudness->addGraph(ui->QPlot_Roudness->xAxis,
                                    ui->QPlot_Roudness->yAxis2);
152
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    ParticleDisplayerFilled = 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    ui ->QPlot_Angularity ->yAxis ->setRange (
321        0, static_cast < double >( Sample ->Angularity.
             HighestFrequency ()));
322    AngularityBars ->setData ( AngularityTicks , stdValues );
323
324    // Setup the Prediction Density Function
325    std :: vector < double > stdPDFkey , stdPDFvalues ;
326    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       if (*e.id() == EXCEPTION_NO_SNAPSHOTS_NR) {
378         CamError->showMessage(
379             "No images acquired! Check you microscope settings
                    ");
380         return;
381       }
382     }
383     Sample->ChangesSinceLastSave = true;
384     if (Sample->ParticlePopulation.size() > 0) {
385       ui->widget_ParticleSelector->setDisabled(
386           false,
387           ui->widget_ParticleDisplay->SelectedParticle->
                  Classification.Category);
388     }
389   }
390
391   void VSAMainWindow::TakeSnapShots() {
392     Analyzer->SIfactorDet = true; // remeber to remove
393     if (!Analyzer->SIfactorDet) {
394       QMessageBox *DetSIFactor = new QMessageBox(this);
395       DetSIFactor->setText("Put calibration Disc under the
                  microscope");
396       DetSIFactor->exec();
397       on_actionCalibrate_triggered();
398       DetSIFactor->setText("Place sample under the microscope"
                  );
399       DetSIFactor->exec();
400     }
401     if (Settings->useBacklightProjection && !Settings->useHDR)
            {
402       for (uint32_t i = 0; i < Settings->StandardNumberOfShots
                ; i++) {
403         SoilAnalyzer::Analyzer::Image_t newShot;
404         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
405         Microscope->GetFrame(newShot.FrontLight);
406         BacklightMessage->exec();
407         Microscope->GetFrame(newShot.BackLight);
408         Images->push_back(newShot);
409         QString ShakeMsg = "Shake it baby! ";
410         int number = Settings->StandardNumberOfShots - i;
411         ShakeMsg.append(QString::number(number));
412         ShakeMsg.append(" to go!");
413         ShakeItBabyMessage->setText(ShakeMsg);
414         ShakeItBabyMessage->exec();
415       }
416     } else if (Settings->useBacklightProjection && Settings->
            useHDR) {
417       for (uint32_t i = 0; i < Settings->StandardNumberOfShots
                ; i++) {
418         SoilAnalyzer::Analyzer::Image_t newShot;
419         newShot.SIPixelFactor = Analyzer->CurrentSIfactor;
420         Microscope->GetHDRFrame(newShot.FrontLight, Settings->
                  HDRframes);
421         BacklightMessage->exec();
```

```cpp
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)"));
485     if (!fn.isEmpty()) {
486       if (!fn.contains(tr(".VSA"))) {
487         fn.append(tr(".VSA"));
488       }
489       delete Sample;
490       Sample = nullptr;
491       delete Images;
492       Images = nullptr;
493       Sample = new SoilAnalyzer::Sample;
494       Images = new SoilAnalyzer::Analyzer::Images_t;
495       try {
496         Sample->Load(fn.toStdString());
497       } catch (boost::archive::archive_exception &e) {
498         // qDebug() << *e.what();
499       }
500       ParticleDisplayerFilled = false;
501       Sample->Angularity.Data = Sample->GetAngularityVector()
502           ->data();
502       Sample->Roundness.Data = Sample->GetRoundnessVector()->
503           data();
503       Sample->PSD.Data = Sample->GetPSDVector()->data();
504       Analyzer->Results = Sample;
505       on_analyzer_finished();
506       ui->widget_ParticleSelector->setDisabled(
507           false,
508           ui->widget_ParticleDisplay->SelectedParticle->
509               Classification.Category);
509     }
510   }
511
512   void VSAMainWindow::on_actionUseLearning_toggled(bool arg1)
          {
513     Analyzer->PredictShape = !arg1;
514   }
515
516   void VSAMainWindow::on_actionCalibrate_triggered() {
517     cv::Mat calib;
518     Microscope->GetFrame(calib);
519     Analyzer->CalibrateSI(16.25, calib);
520   }
521
522   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
            ->SampleFolder),
573        tr("Comma Seperated Value (*.csv)"));
574    if (!fn.isEmpty()) {
575      if (!fn.contains(tr(".csv"))) {
576        fn.append(tr(".csv"));
577      }
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(&
              conversionSucces);
605          qDebug() << cellValues[i + 3];
606          if (conversionSucces) {
607            for (uint32_t j = 0; j < 15; j++) {
608              if (binValue == PSDTicks[j]) {
609                compValues[j] = cellValues[i + 3].toDouble();
610              }
611            }
612          }
613        }
614        ui->Qplot_PSD->addGraph(ui->Qplot_PSD->xAxis, ui->
            Qplot_PSD->yAxis);
615        ui->Qplot_PSD->graph(1)->setData(PSDTicks, compValues)
            ;
616        QPen compPen;
617        compPen.setColor(QColor("darkBlue"));
618        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.0.29 Dialog window Class

```cpp
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);
25   ~DialogSettings();
26
27   void openTab(int newValue);
28 private slots:
29
30   void on_pushButton_RestoreDefault_clicked();
31
32   void on_pushButton_Open_clicked();
33
34   void on_pushButton_Save_clicked();
35
36   void on_checkBox_Backlight_clicked(bool checked);
37
38   void on_comboBox_Microscopes_currentIndexChanged(const
39       QString &arg1);
39
40   void on_comboBox_Resolution_currentIndexChanged(int index)
41       ;
41
42   void on_checkBox_useHDR_clicked(bool checked);
43
44   void on_spinBox_NoFrames_editingFinished();
45
46   void on_doubleSpinBox_LightLevel_editingFinished();
47
48   void on_checkBox_useRainbow_clicked(bool checked);
49
```

```
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
```

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

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

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

```cpp
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    Settings->LoadSettings("Settings/Default.ini");
198  }
199
200  void DialogSettings::on_pushButton_Open_clicked() {
201    QString fn = QFileDialog::getOpenFileName(
202        this, tr("Open Settings"), QDir::homePath(), tr("
            Settings (*.ini)"));
203    if (!fn.isEmpty()) {
204      if (!fn.contains(tr(".ini"))) {
205        fn.append(tr(".ini"));
206      }
207      Settings->LoadSettings(fn.toStdString());
208    }
209  }
210
211  void DialogSettings::on_pushButton_Save_clicked() {
212    QString fn = QFileDialog::getSaveFileName(
213        this, tr("Save Settings"), QDir::homePath(), tr("
            Settings (*.ini)"));
214    if (!fn.isEmpty()) {
215      if (!fn.contains(tr(".ini"))) {
216        fn.append(tr(".ini"));
217      }
218      Settings->SaveSettings(fn.toStdString());
219    }
220  }
221
222  void DialogSettings::on_checkBox_Backlight_clicked(bool
        checked) {
223    ui->tabWidget_Hardware->setTabEnabled(2, checked);
224    Settings->useBacklightProjection = checked;
225  }
226
227  void DialogSettings::
        on_comboBox_Microscopes_currentIndexChanged(
228      const QString &arg1) {
229
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->
          Resolutions.size(); i++) {
238        Resolutions
239            << Microscope->SelectedCam->Resolutions[i].
                to_string().c_str();
240      }
241      ui->comboBox_Resolution->addItems(Resolutions);
242      ui->comboBox_Resolution->setCurrentIndex(
243          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,
                                            QSlider *Contrast,
263                                       QSlider *Saturation,
                                            QSlider *Hue,
264                                       QSlider *Sharpness) {
265     for (uint32_t i = 0; i < selectedCam->Controls.size(); i
          ++) {
266       if (selectedCam->Controls[i].name.compare("Brightness")
            == 0) {
267         Brightness->setMinimum(selectedCam->Controls[i].
              minimum);
268         Brightness->setMaximum(selectedCam->Controls[i].
              maximum);
269       } else if (selectedCam->Controls[i].name.compare("
            Contrast") == 0) {
270         Contrast->setMinimum(selectedCam->Controls[i].minimum)
              ;
271         Contrast->setMaximum(selectedCam->Controls[i].maximum)
              ;
272       } else if (selectedCam->Controls[i].name.compare("
            Saturation") == 0) {
273         Saturation->setMinimum(selectedCam->Controls[i].
              minimum);
274         Saturation->setMaximum(selectedCam->Controls[i].
              maximum);
275       } else if (selectedCam->Controls[i].name.compare("Hue")
             == 0) {
276         Hue->setMinimum(selectedCam->Controls[i].minimum);
277         Hue->setMaximum(selectedCam->Controls[i].maximum);
278       } else if (selectedCam->Controls[i].name.compare("
            Sharpness") == 0) {
279         Sharpness->setMinimum(selectedCam->Controls[i].minimum
              );
280         Sharpness->setMaximum(selectedCam->Controls[i].maximum
              );
```

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

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

```
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     } else {
384       Settings->typeOfObjectsSegmented = Vision::Segment::
            Bright;
385     }
386   }
387
388   void DialogSettings::on_cb_ignoreBorder_3_clicked(bool
        checked) {
389     Settings->ignorePartialBorderParticles = checked;
390   }
391
392   void DialogSettings::on_cb_fillHoles_3_clicked(bool checked)
          {
393     Settings->fillHoles = checked;
394   }
395
396   void DialogSettings::on_sb_sigmaFactor_3_editingFinished() {
397     Settings->sigmaFactor = ui->sb_sigmaFactor_3->value();
398   }
399
400   void DialogSettings::on_rb_useOpen_3_clicked(bool checked) {
401     Settings->morphFilterType = Vision::MorphologicalFilter::
          OPEN;
402   }
403
404   void DialogSettings::on_rb_useClose_3_clicked(bool checked)
        {
405     Settings->morphFilterType = Vision::MorphologicalFilter::
          CLOSE;
406   }
407
408   void DialogSettings::on_rb_useErode_3_clicked(bool checked)
          {
409     Settings->morphFilterType = Vision::MorphologicalFilter::
          ERODE;
410   }
411
412   void DialogSettings::on_rb_useDilate_3_clicked(bool checked)
          {
413     Settings->morphFilterType = Vision::MorphologicalFilter::
          DILATE;
414   }
415
416   void DialogSettings::on_sb_morphMask_3_editingFinished() {
417     Settings->filterMaskSize = ui->sb_morphMask_3->value();
418   }
```

```
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::
         on_doubleSpinBox_MutationRate_editingFinished() {
429    NN->MutationrateUsedByGA = ui->doubleSpinBox_MutationRate
         ->value();
430  }
431
432  void DialogSettings::on_spinBox_Elitisme_editingFinished() {
433    NN->ElitismeUsedByGA = ui->spinBox_Elitisme->value();
434  }
435
436  void DialogSettings::
         on_doubleSpinBox_endError_editingFinished() {
437    NN->EndErrorUsedByGA = ui->doubleSpinBox_endError->value()
         ;
438  }
439
440  void DialogSettings::
         on_doubleSpinBox_maxWeight_editingFinished() {
441    NN->MaxWeightUsedByGA = ui->doubleSpinBox_maxWeight->value
         ();
442  }
443
444  void DialogSettings::
         on_doubleSpinBox_MinWeight_editingFinished() {
445    NN->MinWeightUSedByGa = ui->doubleSpinBox_MinWeight->value
         ();
446  }
447
448  void DialogSettings::on_doubleSpinBox_Beta_editingFinished()
         {
449    NN->SetBeta(ui->doubleSpinBox_Beta->value());
450  }
451
452  void DialogSettings::on_spinBox_InputNeurons_editingFinished
         () {
453    NN->SetInputNeurons(ui->spinBox_InputNeurons->value());
454  }
455
456  void DialogSettings::
         on_spinBox_HiddenNeurons_editingFinished() {
457    NN->SetHiddenNeurons(ui->spinBox_HiddenNeurons->value());
458  }
459
460  void DialogSettings::
         on_spinBox_OutputNeurons_editingFinished() {
461    NN->SetOutputNeurons(ui->spinBox_OutputNeurons->value());
462  }
```

```
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.0.30 Dialog Neural Network Class

```cpp
#ifndef DIALOGNN_H
#define DIALOGNN_H

#include <QDialog>
#include "SoilMath.h"
#include "soilanalyzer.h"
#include "dialogsettings.h"
#include <qcustomplot.h>
#include <QDebug>

namespace Ui {
  class DialogNN;
}

class DialogNN : public QDialog
{
  Q_OBJECT

public:
  explicit DialogNN(QWidget *parent = 0, SoilMath::NN *
      neuralnet = nullptr, SoilAnalyzer::SoilSettings *
      settings = nullptr, DialogSettings *settingWindow =
      nullptr);
  ~DialogNN();

private slots:

  void on_pushButton_Settings_clicked();

  void on_learnErrorUpdate(double newError);

  void on_pushButton_SelectSamples_clicked();

  void on_pushButton_Learn_clicked();

  void on_pushButton_SaveNN_clicked();

  void on_pushButton_OpenNN_clicked();

  void on_actionAbort_triggered();

private:
  Ui::DialogNN *ui;
  DialogSettings *SettingsWindow = nullptr;
  SoilMath::NN *NeuralNet = nullptr;
  SoilAnalyzer::SoilSettings *Settings = nullptr;

  void setupErrorGraph();
  void makeLearnVectors(InputLearnVector_t &input,
      OutputLearnVector_t &output);

  QVector<double> currentError;
  QVector<double> errorTicks;
  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->
          widget_NNError);
30    widget_NNErrorTitle->setText("Learning error");
31    widget_NNErrorTitle->setFont(QFont("sans", 10, QFont::Bold
          ));
32    ui->widget_NNError->plotLayout()->insertRow(0);
33    ui->widget_NNError->plotLayout()->addElement(0, 0,
          widget_NNErrorTitle);
34
35    setupErrorGraph();
36
37    // Connect the NN learn error
38    connect(NeuralNet, SIGNAL(learnErrorUpdate(double)), this,
39            SLOT(on_learnErrorUpdate(double)));
40  }
41
42  DialogNN::~DialogNN() { delete ui; }
43
44  void DialogNN::on_pushButton_Settings_clicked() {
45    SettingsWindow->openTab(2);
46    SettingsWindow->show();
47    setupErrorGraph();
48  }
```

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

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

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