



Vision Soil Analyzer

Product design of a vision based soil analyzer

Jelle Spijker

Copyright © 2015 Jelle Spijker
PUBLISHED BY ROYAL IHC

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

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

First printing, September 2015



Foreword

I can honestly say, that I have exceeded my own expectations. Before I started this minor, I had no knowledge of electrical devices, a limited know-how in programming languages, let alone any noteworthy C++ skills and to my shame I have never ever worked on a Linux computer.

And what a change it has been; Now I'm reluctant to start up my windows computer because I'm more at home with my linux terminal, customizing and hacking a kernel, been there done that. Programming complex neural networks and Fast Fourier Transformation in C++ with pointers, euuhh whats new. Word, Excel who needs them, when you got L^AT_EX and Matlab. I can honestly say I have transcended to a new level of nerdiness.

But the best part is, that I saw people in my professional working sphere, whom seemed genuinely interested in this product. So much interested, that, although our main products are big boats and machinelike contraptions straight out of a horror flic, they are willing to give me a shot to further develop this product. Allowing me to creating a device by my own machinations and thus making sure that I have one item checked from my bucket list. For this opportunity I'm thankful. But I suspect my girlfriend doesn't share that gratitude.

Annacarina stick in there, share me just a bit longer with my muse Vision Soil Analyzer.

I still love you more then her!

Jelle Spijker



Summary

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 Analyzer 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 color, texture and structure.

This product documentation describes the design and realization of a vision based soil analyzer. It does so by describing its functional design, which lies at the heart of the product and explains that the main function of the device is:

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

From here the requirements with regard to function and production are extrapolated. The functional requirements quantize the needed performance on color, texture and structure and give conditions how to test this performance. While the technical requirements tell how the product is made. Giving constraints on dimension, programming language and electrical devices used.

An important aspect of a vision based soil analyzer is its interaction with the user. This is illustrated with the user interface and the manuals describing the interaction protocols from an end-user and administrator perspective.

The technical design dissects the structure into subsystems, such as microcontroller, light environment and sample plate. These and more are necessary to perform the main function. These subsystems have a high level of interaction with each other which are set out in the architecture.

The vision based soil analyzer rests heavily on vision algorithms, these follow a certain established lines, which are set out in the acquisition, enhancement, segmentation, feature extraction and classification steps.

All of the above serve as the basis for the realization of a vision based soil analyzer.

In order to design and build a prototype te working environment is described. This is a dual booted Windows / Linux laptop, running QT designer, Matlab and Siemens NX. The project itself is hosted on Github and Upverter.

The technical aspects are described for the disciplines: software, mechanical and electronic engineering. Because the main focus lies on the vision aspects of the device, the previous determine route is set out in detail, describing the acquisition steps and various strategies which can be pursued. The enhancement from intensity to segmentation steps are described, such as blurring and adaptive contrast stretch. The follow up algorithm, transform an intensity picture to individual blobs. This is done by finding an optimal threshold between pixels which belong to a particle and background.

Further transformation are performed to ensure each particle is identified. The shape of these are classified by describing the contour in the frequency domain, and feeding the complex numbers, obtained with a Fast Fourier Transform, in to a neural network. The resulting classification is one of angularity. Roundness is categorized with the Hu moments. The size of the particle is determined and used to describe the soil sample as a particle size distribution. All obtained information is presented to the user, through a multitude of means using the graphical user interface and a report generator.

Finally the design is verified against the previous determined requirements and a conclusion is drawn. Throwing a fare-sight into the project to come.



Contents

1	Introduction	11
2	Functional design	15
2.1	Global input-process-output	15
2.2	Specifications	16
3	User interface	19
3.1	Global work flow	19
3.2	Graphical User Interface	20
4	Manuals	21
4.1	User manual	21
4.1.1	Analyzing a soil sample	22
4.1.2	Generate report	24
4.2	Administrator manual	25
4.2.1	Maintenance and upgrade process	25
4.2.2	Teaching the Neural Network	25
5	Technical design	27
5.1	Hierarchical structure	27
5.2	Architecture	28
5.3	Detailed Input-Process-Output schematics	28

6	Vision design	29
----------	----------------------	-----------

II

Realization

7	Development Environment	33
7.0.1	Code naming conventions	33
7.0.2	Software development environment	34
7.0.3	Modeling development environment	36
7.0.4	Electronic development environment	36
8	Technical Realization	37
8.1	Run Environment	37
8.2	Electrical design	38
8.2.1	Led driver	38
8.2.2	Light level meter	39
8.2.3	Global position unit	39
8.3	Case design	40
8.4	Program structure	40
9	Vision realization	43
9.1	Image acquisition	43
9.2	Image enhancement	44
9.3	Feature extraction	45
9.3.1	Shape features	45
9.3.2	CIE La*b* extraction	49
9.3.3	Fast Fourier Descriptors	50
9.3.4	Particle Size Distribution	51
9.4	Classification	54
9.4.1	Sphericity using Hu moments	54
9.4.2	Angularity using a Neural Network	55
9.4.3	Genetic Algorithm	56

III

Verification

10	Comparing against specifications	61
11	Conclusion	65

IV

Addenda

Bibliography	69
Index	71

A	Graphical User Interface	73
B	Example Soil Report	77
C	HAN minor Machine design: Student assessment	83
D	HAN Electrical and electronic engineering: Student assessment	85
E	Assembly drawing	87
F	Development Environment setup	89
G	Run Environment setup	93
H	SoilMath Library	97
I	Hardware Library	151
J	Vision Library	207
K	Analyzer Library	263
L	QOpenCVQT Library	293
M	QParticleDisplay Library	297
N	QParticleSelector Library	303
O	QReportGenerator Library	309
P	Vision Soil Analyzer Program	323
Q	Reference manual	363



1. Introduction

This project finds its roots in the minor Embedded Vision Design taught at the university of applied sciences HAN, hereafter named EVD. During this minor an embedded device was developed which analyses soil samples using a microscope. This Vision Soil Analyzer or VSA for short, analyzes samples using the optical properties. It gives an user information on color, texture and structure.

This device is developed in collaboration with Royal IHC and MTI Holland. Royal IHC is one of Holland major shipyard companies and specializes in dredging and offshore. MTI Holland BV is IHC knowledge center. They're worldwide leading center of expertise in the area of dredging, mining and deep-sea mining processes, this information is translated 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 its 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 environment 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 rewritten in C++.

The design and realization of this second prototype is set out in this report, the goal of this document is to:

Describe the design specifications for a vision based oil analyzer, in such away that it can be reproduced and improved, within a period of 10 weeks.

The project encompasses multiple disciplines (mechanical, electrical and software). The output of each of these disciplines are described, but the focus lies at the vision based algorithms and software that fulfills the main function; To analyses a soil sample using its optical properties. In performance of this main function it will give a user information regarding color, structure and texture.

The color of a sample is presented to a user in the CIE Lab color-models. These color model show correlation between soil properties, such as color and organic carbon, related to the fertility of the soil. Conversion between different color-models are CPU intensive, because each pixel will be transformed using multiple algorithms. It's therefore paramount that calculations are done with a minimum of machine instructions and with acceptable errors.

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

Information about the structure of the soil is extrapolated from the individual particles shapes. These shapes are described in the frequency domain, using a Fast Fourier Transform and are 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 document follows the following structure: In part I the design is laid out. In this part the following chapters describe basic design; Chapter 2 explores the function the device needs to fulfill and which specifications it has to have. While chapter 3 illustrate the user interaction and interface. This serves as input for chapter 4, where the user interaction is described in the form of a manual. The technical design is illustrated in chapter 5 and the vision design is outlined in chapter 6.

Part II tells how the design is transformed in to a working prototype and how it can be reproduced. This is achieved by first describing the development environment in 7. In this chapter the setup for software, modeling and electronics engineering are described in detail. Armed with this setup the technical realization is described in chapter 8. Lastly the vision realization is recounted, this is the focal point of this documentation and can be found in chapter 9.

Verification of the design takes place in part III. In chapter 10 is prototype verified against the previous determined specifications. Lastly the conclusion is set out in chapter 11. All addenda chapters, such as bibliography, index and appendices are to be found in part IV.



Design

2	Functional design	15
2.1	Global input-process-output	
2.2	Specifications	
3	User interface	19
3.1	Global work flow	
3.2	Graphical User Interface	
4	Manuals	21
4.1	User manual	
4.2	Administrator manual	
5	Technical design	27
5.1	Hierarchical structure	
5.2	Architecture	
5.3	Detailed Input-Process-Output schematics	
6	Vision design	29



2. Functional design

A functional design lays at the heart of a product. It is an abstract representation of a device and it illustrates its main function. In this chapter the workings of a vision based soil analyzer is laid out. It explains which role a vision based soil analyzer needs to fulfill in order to satisfy a user generated need. This main functionality and its output is visualized in an Input-Process-Output (IPO) diagram. This diagram aids in deciding the specification and setting up a user interface. These in turn dictate the interaction with the outside world.

2.1 Global input-process-output

The main function of a vision based soil analyzer is evident from its name. The user can expect a device which performs an analysis of a soil sample. It does so by capturing and digitizing reflected light of the individual soil particles. This function is illustrated below in an Input-Process-Output (IPO) diagram, see figure 2.1. This is a so called black box approach. It shows an input, an output and a process, where the inner workings are not yet known and relevant.

Technical system

Prototype of an intelligent soil microscope

Main function

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



Figure 2.1: Main Input-Process-Output diagram

2.2 Specifications

With the global input-process-output in mind the functional specifications can be written. This is done by identifying requirements that lie at the heart of its main functionality. These are specification that define a product. It is important to note that there are two types of requirements: functional and technical requirements; Each requirement can either be constant or a variable. The constant requirements are the baseline. If the product doesn't fulfill these, it can't be called a soil analyzer. Whilst variable requirement determine how well a product performs.

Functional requirements

Functional requirements describe the purpose of the product.

ID	Description	Type
F1	Quantify color	
F1.1	Determine the color in a RGB color model, from all visually (by human eye) discernible particles	Const.
F1.2	Chromatic a^* values must lie within 3σ	Const.
F1.3	Chromatic b^* values must lie within 3σ	Const.
F2	Quantify texture	
F2.1	The result of an analyzed sample should fall within a probability of at least $P = 0.95\%$ when compared against the result of the same sample, but obtained using the established sieve method. These results are to be compared by Welch's t-test	Const.
F2.2	PSD bins should have the same range as the fractions used in the sieving method	Const.
F3	Quantify structure	
F3.1	Roundness should be assigned in three categories	Const.
F3.2	Angularity should be assigned in six categories	Const.
F3.3	Predicted values should have at least a linear regression value of $R \geq 0.9$ when compared to expertly classified particles	Const.
F4	General specifications	
F4.1	Analyze particle with sizes within the range $200\mu m \leq P_{size} \leq 2mm$	Const.
F4.2	No more than 2% of the extracted blobs may be connected particles	Const.
F4.3	Analyzing a sample should take no longer than 1min (rearranging of sample between shot disregarded)	Const.
F5	Interaction	
F5.1	Show individual particles	Const.
F5.2	Show PSD graph with particle size in logarithmic scale	Const.
F5.3	Show Angularity in histogram	Const.
F5.4	Show Roundness in histogram	Const.
F5.5	Show probability distribution function in the histogram	
F5.6	Information can be shown on a screen	Const.
F5.7	Exporting to pdf file	Const.

Table 2.1: Functional requirements

Technical requirements

Technical requirements describe the functionality of the device with regards to its peripherals and its technical environment. They're described in such a way that they are either true or false.

ID	Description	Type
T1	Software environment	
T1.1	The software should run on an Linux device	Const.
T1.2	The software should be written in C++	Const.
T1.3	The software should be written as OOP and be reusable	Const.
T1.4	The software should be written with revision control	Const.
T1.5	Easily portable to Windows environment	Const.
T1.6	Easily portable to Android environment	Const.
T2	Hardware environment	
T2.1	Should run on an ARMv7 or higher device	Const.
T2.2	Should run on a x86 or x64 device	Const.
T2.3	At least 1GHz processing power	Const.
T2.4	At least 128MB memory	Const.
T2.5	At least 2GB storage	Const.
T3	Peripherals	
T3.1	USB connection	Const.
T3.2	Ethernet LAN and/or WAN connection	Const.
T3.3	GPS unit	Optional
T3.4	Light controller	Const.
T4	General specifications	
T4.1	Sample file size should not exceed 10mb	Const.
T4.2	Guard the maximum size of particles to 2mm	Const.
T5	Prototype specifications	
T5.1	Dimensions should not exceed 400[mm] × 200[mm] × 200[mm]	Const.
T5.2	The total weight may not exceed 5[kg]	Const.

Table 2.2: Technical requirements



3. User interface

The User Interface is responsible for the interaction with a user. It does so by accepting user input, such as a soil sample and presents the user with human readable information. In order to guarantee accurate result, a certain work flow has to be followed. This work flow is worked in detail in the manuals, which are depicted in section 4. In the follow sections the global work flow is illustrated as well as the graphical and hardware user interface.

3.1 Global work flow

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

The user takes a snapshot, rearranges the sample on the sample plate and takes an other snapshot. This is repeated for an multitude of times, until enough particles are analyzed to give accurate statistical results.

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

3.2 Graphical User Interface

Most information is conveyed through the graphical user interface or GUI for short. The complete GUI encompasses different windows and dialogs, all these can be found in appendix A. The main window consist of the following parts:

1. **Shape selector** - This widget allows a user to see and change the shape category of the focused particle.
2. **Particle browser** - This widget allows a user to browse through the individual particles in the soil sample.
3. **FFT Graph** - A graph showing the absolute value of the Fast Fourier Transform for the particle edge.
4. **Sphericity histogram** - A histogram which shows the sphericity of the sample with a probability function and mean value.
5. **Particle Size Distribution** - A cumulative function of the particle sizes on a logarithmic scale.
6. **Angularity histogram** - A histogram which shows the angularity of the sample with a probability function and mean value.
7. **Toolbar** - A buttonbar for the most common actions: *New Sample*, *Save Sample*, *Load Sample* ...

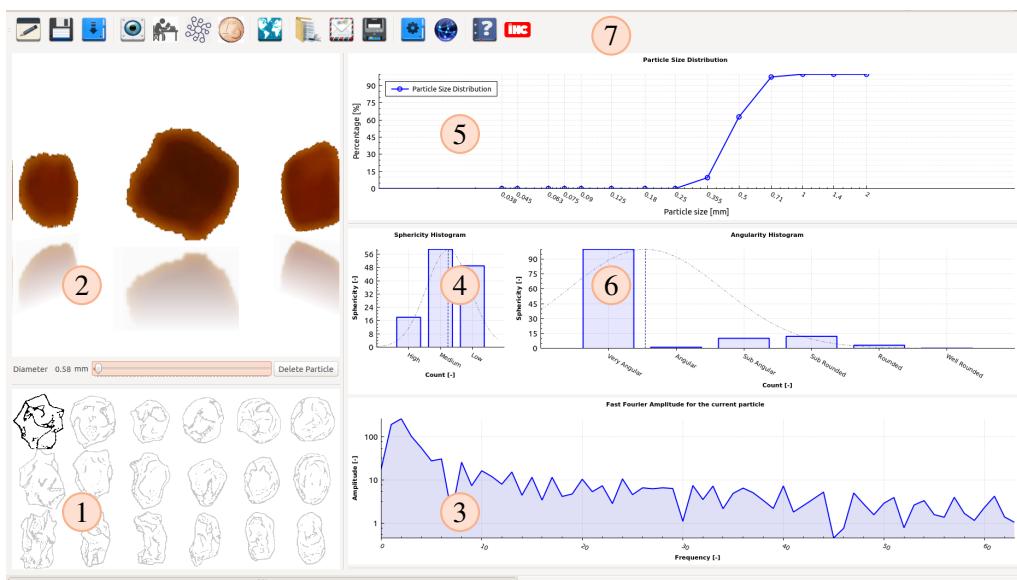


Figure 3.1: Main Graphical User Interface

4. Manuals

Interaction between user and the device is illustrated with the following basic manuals. These manuals differentiate between end-user and administrator. The end-user is the normal operator of the machine, the one who wants to know the properties of a specific soil sample, whilst the administrator has to maintain the device.

4.1 User manual

When a user wants to analyses a soil sample he first has to make sure, the device is connected to a power outlet with an adapter that provides a DC voltage of 5V with at least 2A. Any generic HDMI monitor which support 720p (HD-ready) can serve is its main user interface. It is optional that the device is hooked to a network using the UTP connection or directly to a x86 or x64 computer (running windows 7 or higher and Linux kernel 2.19 or higher) through USB connector. When communication with third parties is needed the network or gateway computer needs to be connected to a WAN¹. The enable learning button (figure 4.1) has to be grayed out for normal operation.



Figure 4.1: Learning button

¹Wide Area Network

4.1.1 Analyzing a soil sample

Most of the user interactions in the steps below have to be initiated through the toolbar, figure 4.2.



Figure 4.2: The toolbar

Step 1: Filling the sample plate

The user places a dried soil sample on the disc, as shown in figure 4.3, he make sure it is evenly placed. Making use of the complete area of the disc.



Figure 4.3: Evenly spaced soil sample

Step 2: Placing the sample plate under the microscope

The disc is placed in the slot under the microscope. The slot only allows particle to be placed under the microscope with a height of 2[mm] or smaller. This step is shown in figure 4.4



Figure 4.4: Placing the sample under the microscope

Step 3: Creating a new sample

When a user presses the "New Sample" icon (figure: 4.2 button 1) on the toolbar the microscope takes a shot of the sample. When the sample needs to be rearranged for additional shots, a dialog will appear prompting a user to shake the disc. This is done by repeating step 1 and 2. Step 1 till 3 has to be performed for the amount of times specified by the administrator. This is usually 10 times.

Step 4: browsing through the sample

The segmented particles are shown in the particle browser, a user move through these individual particles by clicking on the left or right particle, or by moving the slider below the display. Below the particle browser is the suggested shape category depicted. If a user deems the particle wrongly categorized, he simply selects the new category. These

new categories are used to further improve this classification process and serves a new learning datasets for the neural network.

Step 5: Deleting doubles

It is possible when using the current prototype, that it sees multiple connected particle as one. Because these connected particle distort the statistical analysis, a user has to delete these manually. He does so by clicking the "delete particle" button.

Step 6: Comparing the sample against a know PSD

When pressing the right mouse button on the PSD graph a popup menu appears. Which allows csv² files to be loaded as comparison.

Step 7: Saving or emailing the sample

Pressing the save button (figure 4.2, button 2) allows a sample to be saved on local storage device. Depending on the size of the individual particles and the total amount, the file size varies between 1 and 10 mb. The internal storage is 2gb of allocated space for storage. It is recommended to save the sample on an external device using the send email button (figure 4.2, button 10). This button sends the opened sample to a previously selected email address of choice.

4.1.2 Generate report

If the user wants a standard human readable report of the analyzed soil sample he has the option to select the generate report button (figure 4.2, button 9). This button opens a new window (figure 4.5) with the generated report. An example report is shown in appendix B. The device has to have working Internet connection in order to download a map of the location.

When the report generator is open the user can fill in additional information, regarding the origin of the sample. When all is according to satisfaction the report can be saved locally, send to an email address or a network printer.

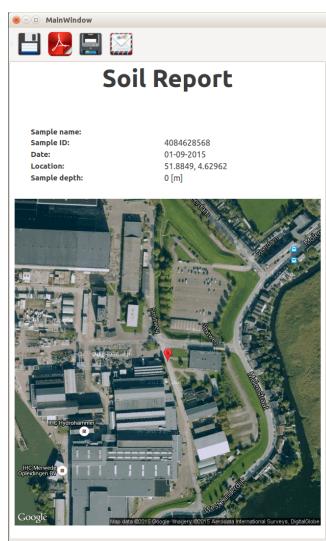


Figure 4.5: Report generator

²Comma Separated Values

4.2 Administrator manual

The VSA runs on an embedded Linux operating system, an administrator is expected to have prior knowledge of this operating system, such as working on the command prompt. Administration is performed through a dedicated network connection. It is expected that the ip address of the VSA is known.

4.2.1 Maintenance and upgrade process

Connect to a WAN is needed in order to upgrade the system. An administrator has the option to login through a console using the following credentials:

Username: ubuntu

Password: temppwd

Upgrading the OS can be performed with the usual steps:

```
~$ sudo apt-get update  
~$ sudo apt-get upgrade
```

Upgrading the software can be performed by:

```
~$ cd git/VisionSoilAnalyzer  
~/git/VisionSoilAnalyzer$ git pull  
Enter the following credentials:  
username: VSA_Admin@VSA.com  
password: VSAisThabomb  
~/git/VisionSoilAnalyzer$ TAG=$(git for-each-ref refs/tags --sort=-taggerdate --format=%(refname)' --count=1)  
~/git/VisionSoilAnalyzer$ git checkout tags/$TAG  
~/git/VisionSoilAnalyzer$ cd Linuxscripts  
~/git/VisionSoilAnalyzer/Linuxscripts$ ./config.sh  
~/git/VisionSoilAnalyzer/Linuxscripts$ ./make.sh  
~/git/VisionSoilAnalyzer/Linuxscripts$ ./install.sh
```

Afterwards you can logout.

4.2.2 Teaching the Neural Network

The Neural Network can learn from previous analyzed samples, this is done on the VSA itself. By pushing the Neural Network button, a new dialog window appears, as shown in figure A which can be found in appendix A. In this dialog the user can select a set of previous analyzed samples which serve as a learning data set.

When the button "Open Settings" is pressed the windows depicted in figure 4.6 is shown. In this window the user can select the preferred network configuration.

Setting up the neural net

The user configures the neural net by first specifying the number of input neurons, these describe are the amount Fast Fourier Descriptors used in describing the shape of particle, a number between 8 and 20 is usually more then enough. Here after the hidden neurons are to be specified. the maximum amount is 200. Finally the output neurons are to be specified. At the current time these are set in 18 categories and can't be changed.

Setting up the learning mode

The Genetic Algorithm allows the user to specify the learning variables. This is done by determining the maximum number of generations, or iterations. Secondly the population size has to be specified as well as the mutation rate. Thirdly the number of elite population member are to be set. These are population members that are guaranteed to be part of the next generation. Fourthly the acceptable end error has to be specified. As well as the minimum and maximum allowed weight. Lastly the user has the option to select the revolution mode. This mode can be useful when the error gets stuck on a certain level. It starts a fresh by killing the elite and mutate the rest of the population and ushers in a new era.

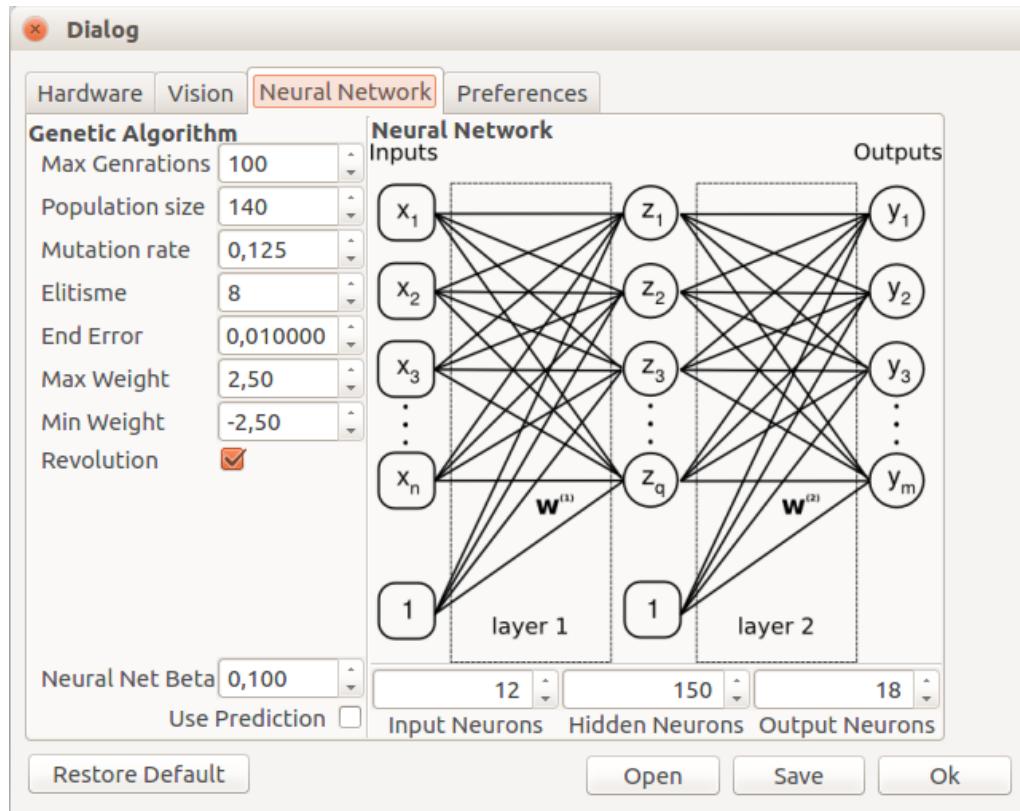


Figure 4.6: Settings Neural Network Interface

5. Technical design

The technical design describes the sub-systems and their interaction. These are set out in a hierarchical structure, identifying the individual sub-systems. Where their underlying interactions are described in the architecture. The picture which is illustrated with the hierarchical structure and the architecture serves as the basis for a detailed IPO.

5.1 Hierarchical structure

The system can be divided in the subsystems depicted in figure 5.1. These sub-systems have a high level of interaction and are in a whole responsible for the fulfillment of the VSA main function.

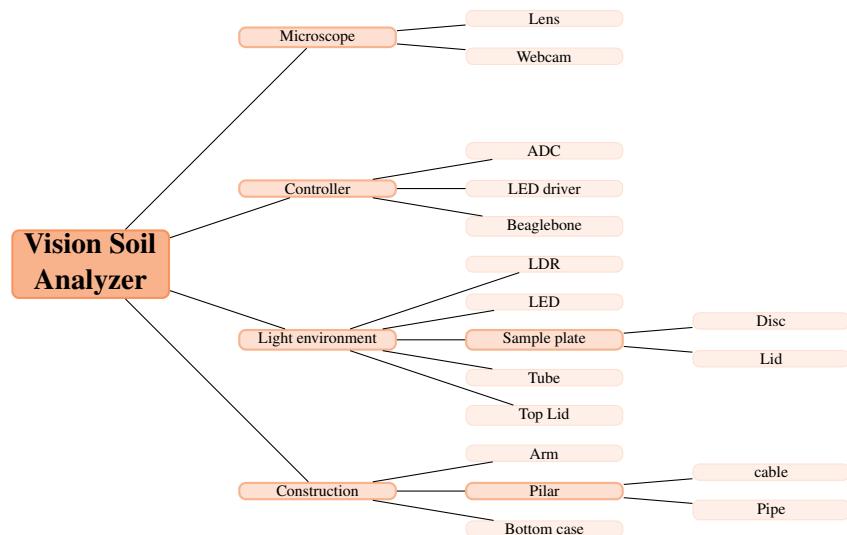


Figure 5.1: Hierarchical diagram of the vision soil analyzer

5.2 Architecture

The architecture diagram below depicts the cohesion between the different electronically systems. It is obvious that the microcontroller plays a pivotal role in these relationships.

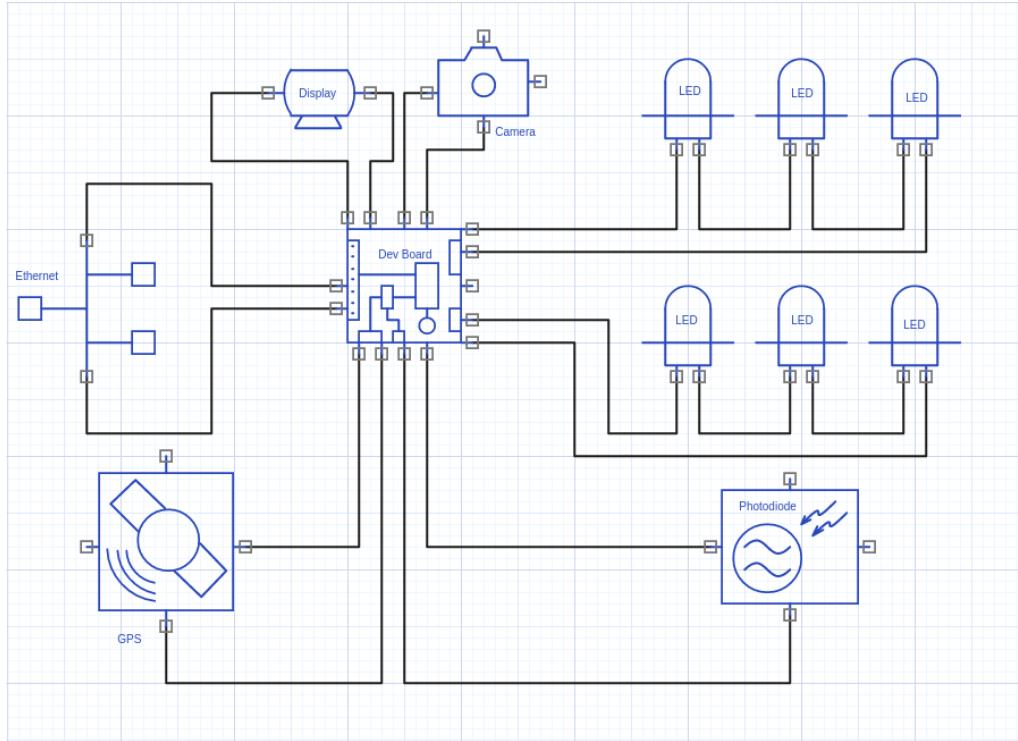


Figure 5.2: System design

5.3 Detailed Input-Process-Output schematics

The main detailed IPO is illustrated below:

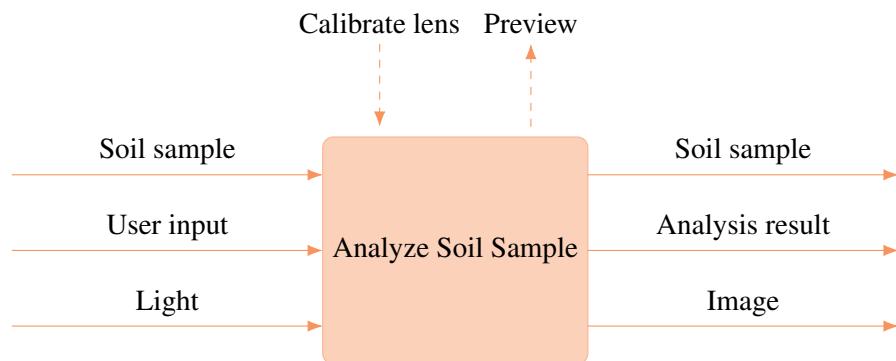


Figure 5.3: Main detailed Input-Process-Output diagram



6. Vision design

The main focus of a vision based soil analyzer lies on the vision algorithms. These lie at the heart of the device. Are to be described in detail in this document. The design and connectedness between these algorithms is described below and depicted in the flow diagram (figure 6.1).

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

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

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

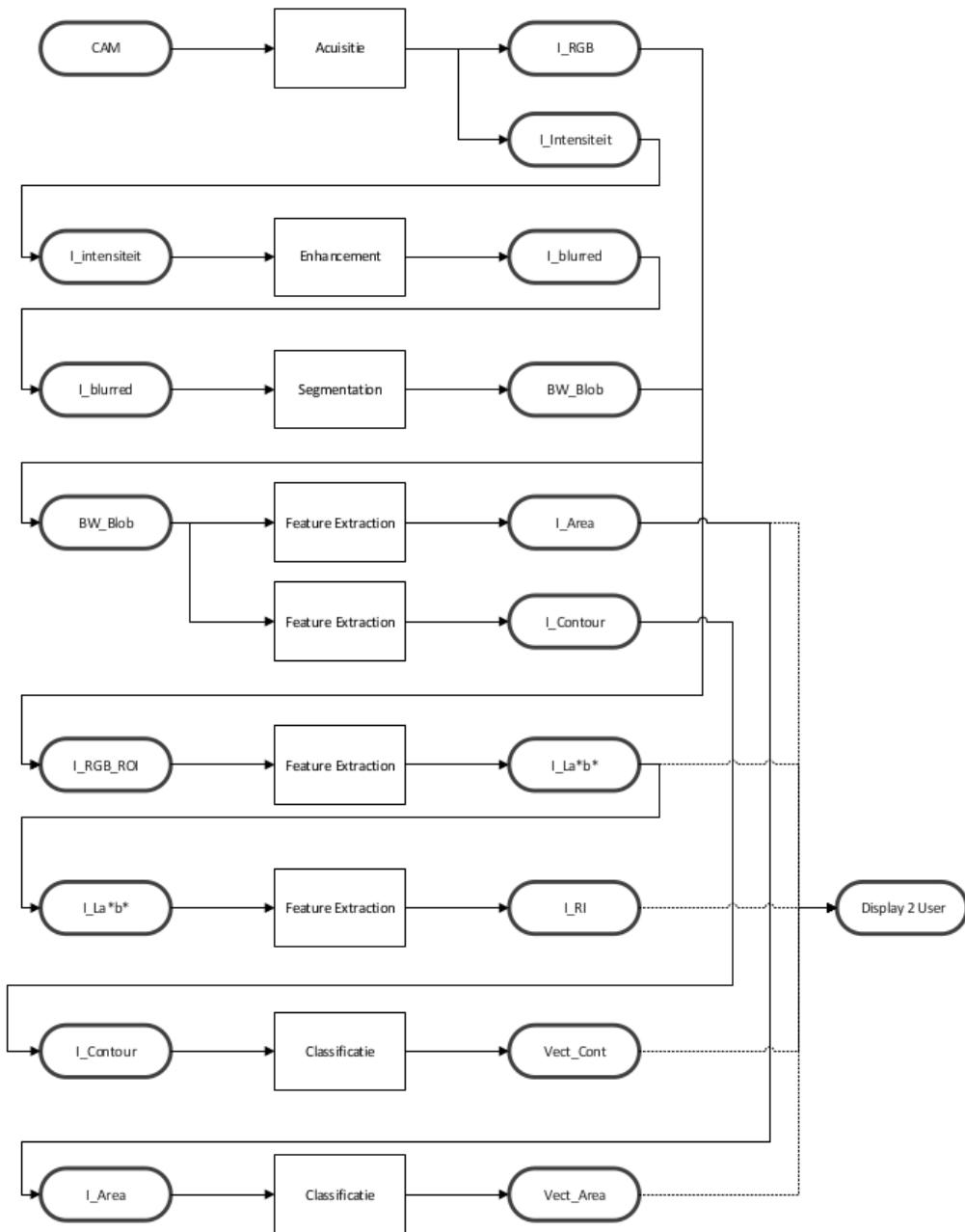
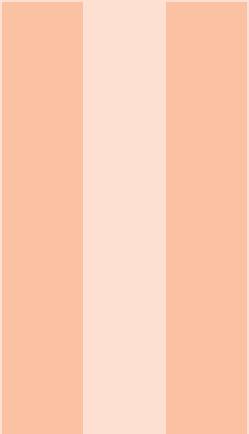


Figure 6.1: Vision flow



Realization

7	Development Environment	33
8	Technical Realization	37
8.1	Run Environment	
8.2	Electrical design	
8.3	Case design	
8.4	Program structure	
9	Vision realization	43
9.1	Image acquisition	
9.2	Image enhancement	
9.3	Feature extraction	
9.4	Classification	



7. Development Environment

The project is developed using three major disciplines; These are mechanical, electrical and software engineering. Each of these disciplines require their own setup and tools. All three are described in their own section below. As indicated in previous chapters the current focus lies on the software development stage, electrical en mechanical systems are described, but with less detail.

At the basis of all three development environments lies the hardware. The specifications given below, describe the current development computer. It is guaranteed that the project can be recreated with a similar computer.

- Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
- 8gb memory
- Nvidia 820M
- SSD 128 gb
- HDD 500 gb
- Dual boot Ubuntu 15.04 / Windows 10

7.0.1 Code naming conventions

The release are named after anime or manga characters in alphabetical follow up. The first release is dubbed "Gaara" from the anime "Naruto". Gaara is a ninja who can control sand. The following alpha en beta release have preceded version 1.0(Gaara):

- Akira** version 0.9.0
- Bunpuku** version 0.9.5
- Chouji** version 0.6.6
- Dot PYXIS** version 0.9.7
- Enryuu** version 0.9.8
- Father** version 0.9.9



Figure 7.1: Gaara of the sand

7.0.2 Software development environment

The software for the VSA runs on an embedded Linux device. This environment is described in chapter 8.1. It is highly recommended that the software development environment mirrors this configuration. From the kernel and the operating system towards the package and libraries.

When development takes place in a Linux environment, a tight integration with the prototype is ensured. Eliminating the need to setup the environmental settings and script for multiple operating systems. This also adds the option to debug the software on the development computer.

Programming language

The software for the VSA is written in C++. This language was chosen because of its efficiency and high level of abstraction. Most image processing algorithms look at each individual pixel. Since the image obtained with the VSA microscope are in the order of 10×10^6 [Pixels] a lot of machine instructions can be eliminated by programming in C++. One of its main strength is, that it allows for direct memory access without type checking and error checking. As Bjarne Stroustrup, the creator of C++, puts it [5] :

C++ is a general-purpose programming language providing a direct and efficient model of hardware combined with facilities for defining lightweight abstractions.

Integrated Development Environment

Development is performed on a desktop computer running Linux 3.19.0-18-generic #. Ubuntu 15.04. The preferred Integrated Development Environment, or IDE is QT Creator Community edition. This is open-source IDE and available for Linux /

Windows / Mac. Version control is handled using the services of Github the main project page is VisionSoilAnalyzer - project page (<http://peer23peer.github.io/VisionSoilAnalyzer/Webpage/index.html>). Access to the Github page requires collaboration privileges.

The basic list of installed packages is given below. The complete list of packages and installation steps are depicted in appendix F.

- Environment
 - Kernel Linux 3.19.0-18-generic
 - Ubuntu 15.04
- IDE-tools
 - Clang 3.6 compiler
 - C++ GNU compiler
 - QT Creator
 - Valgrind
 - Doxygen
 - Git
 - Cmake
- Libraries
 - OpenCV 3.0 beta
 - CUDA 7.0 SDK
 - ZLib
 - Boost 1.58
 - Video4Linux
 - GStreamer



It is a fact that computers and their environment evolve. New settings, packages and development changes are described in the project wiki. Which is actively maintained during the complete development phase. This wiki can be found at <https://github.com/peer23peer/VisionSoilAnalyzer/wiki>

Object Orientated

The software for the VSA is object orientated and written in such a way that external parties can work on section of the code while remaining unaware of the complete picture. This is achieved by writing classes, or so called shared libraries. These are individual projects, which are compiled individually and will be called from the main program during runtime. These classes can be reused with other projects.

Readability

It is common practice to document the routines and functions, explaining the code to third parties and improving the overall readability. These comments are scattered through out the source code and can be extracted with Doxygen into software references documentation. The resulting reference manual can be found in appendix Q.

Directory structure

When cloning the git the folder structure is automatically applied. This is not the case for the build folder. This folder hold the compiled source code and from here the program is executed. Since it is important that links between project are maintained, the directory structure as given in appendix F has to be obeyed.

Testing and benchmarking

Testing is done using the QT unit test framework results are verified against known results. Which are calculated via Matlab, Mathematica or Python. Benchmarks are done using the QT unit test framework and will test multiple solutions. Solutions that are deemed obsolete by the benchmark results will not be removed but be renamed with a _ in front of the function name _FunctionName. **Valgrind** is used to determine memory leakages and function profiles. These profiles will be the guide which determine the priority of functions to be optimized.

7.0.3 Modeling development environment

The modeling of the casing is performed with Siemens NX 10 running on a Windows 10 operating system. The models are placed in the folder **3Dmodel\<Release name>**. The hierarchical design set out in section 5.1, this has to be followed as much as possible. That means multiple assemblies and the most basic component are single parts. Each part has to be correctly named and the materials are to be specified.

Production drawings or models are to be placed **3Dmodel\<Release name>\Production** files . 2D production files are to be made according to the mono-system and saved in PDF file formats. 3D models necessary for CNC-machining or 3D-printing are to be saved as STL files. 2D production files for laser cutters are to be provided in Autocad DXF (2007) format.

7.0.4 Electronic development environment

Design of the electronic systems are to be made on Upverter.com. Upverter can be described as Github for electronic designs. the current schematic is to be found at <https://upverter.com/UniversityofAppliedSciencesHAN/9f177d2bd16397c8/Gaara/>. Simulation of the different electronic parts are to be performed in Matlab Simulink in the SimElectronics environment.



8. Technical Realization

In the subsequent chapter the various designs are worked out. These are the running environment, electrical design, The casing prototype and the program structure. Although all designs disciplines are described the focus lies on the vision and software routines.

8.1 Run Environment

Although the software is build to run on any Linux enable device, it is intended to be run on embedded ARM device. The choice for the basic run environment is made for the Beaglebone black or BBB for short. It is a low-cost community supported development platform made for prototype developers and hobbyists. It has the following specifications:

Processor AM335x 1GHz ARM®

RAM 512[MB] DDR3

flash storage 4[GB] 8-bit eMMC on-board flash storage

GPU 3D graphics accelerator

floating point accelerator NEON

PRU 2x 32-bit microcontroller (Programmable Real-time Units)

OS Ubuntu 14.04 with kernel V4.1.x

The following packages have to be installed:

- IDE-tools
 - Clang 3.6 compiler
 - C++ GNU compiler
 - QT Creator
 - Valgrind
 - Doxygen
 - Git
 - Cmake
- Libraries
 - OpenCV 3.0 beta
 - CUDA 7.0 SDK

- ZLib
 - Boost 1.58
 - Video4Linux
 - GStreamer

The complete installation steps and how to compile the custom kernel are set out in appendix G

8.2 Electrical design

The schematic depicted below shows the how the electronic parts are to be connected to the pin-out on the Beaglebone black. It consist of two LED drivers, a light level meter and a global position unit. All the datasheet can be found at [/Electronics/Prototype Gaara \(V1.0\)/Datasheets](#).

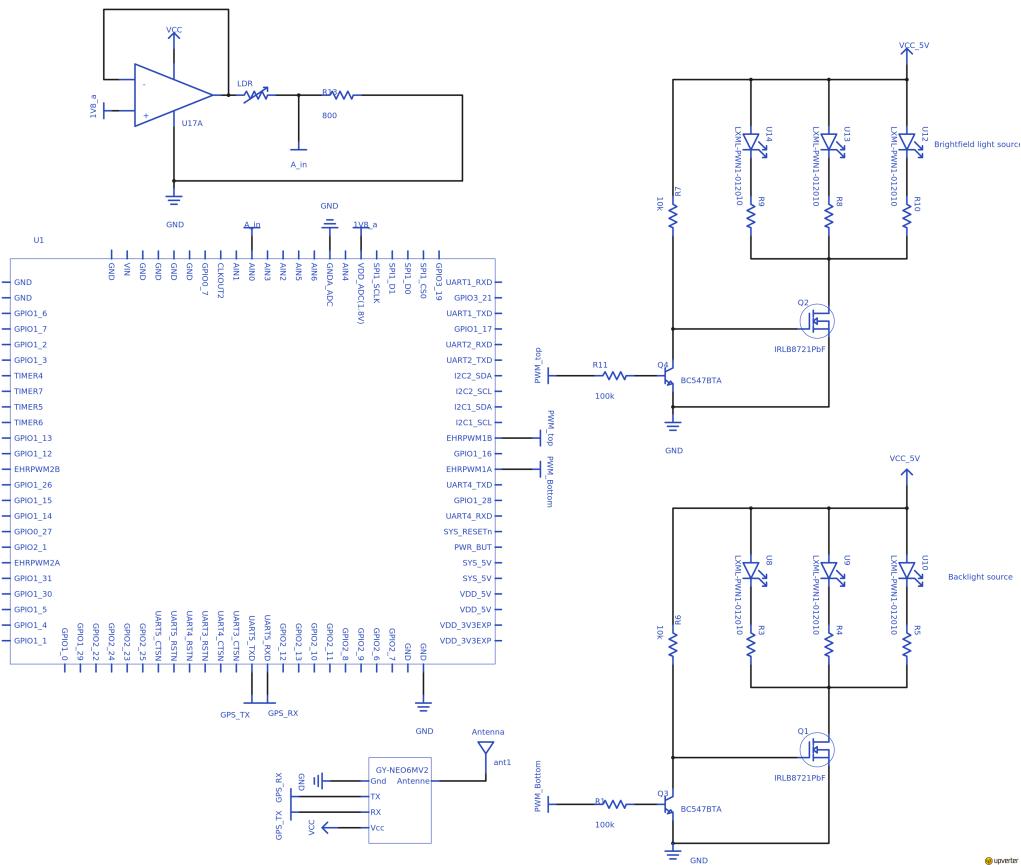


Figure 8.1: Schematic Design

8.2.1 Led driver

Each led driver consists of three Philips LUXEON Rebel white leds. These leds are chosen because they emit a cool white 10.000K color profile and generate 180 lumen each over a 160 degree cone. Each of these leds have a forward current of 450mA, which is quite high. Since the maximum allowed current to flow through a BBB GPIO¹ pin cannot exceed 10mA a MOSFET IRLB8721PbF is used. Although this type of MOSFET allows

¹General Purpose Input Output

the flow of current around $V_{GS(th,max)} = 2.35[V]$ The current doesn't exceed 1A. As shown in figure 8.2. The setup is further expended by using BC547 NPN transistor, where the base is connected to the GPIO and the current flows from the 5[V] power rail.

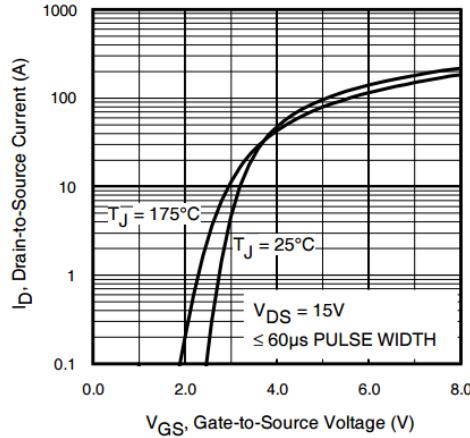


Figure 8.2: Typical Transfer Characteristic (Source: IRLB8721PbF datasheet)

The complete driver is controlled using a PWM² pin on the Beaglebone Black.

8.2.2 Light level meter

The light level is measured in the light environmental room. This is done by using the 12-bit ADC chip of the Beaglebone black which uses 1.8[V]. It works on the principle of a simple voltage divider circuit. Because hooking the LDR³ directly on the 1.8[V] power rail of the Beaglebone the circuit will draw current and acts as a variable load, which will in turn effect the voltage level and thus effect the measurement. This problem is solved by building a small voltage follower circuit using an LM358P op-amp. This op-amp uses the 1.8[V] power-rail as reference and draws it current from the 5[V] power rail.

8.2.3 Global position unit

The global position unit has yet to be implemented.

²Pulse Width Modulation

³Light Dependant Resistor

8.3 Case design

The Case is design in Siemens NX 10 and consists of 21 separate parts and assemblies. Figure 8.3 shows the main assembly drawing. Numbering the individual parts. The casing parts 6 ,11, 12 ,20 are designed to be printed using a 3D-printer

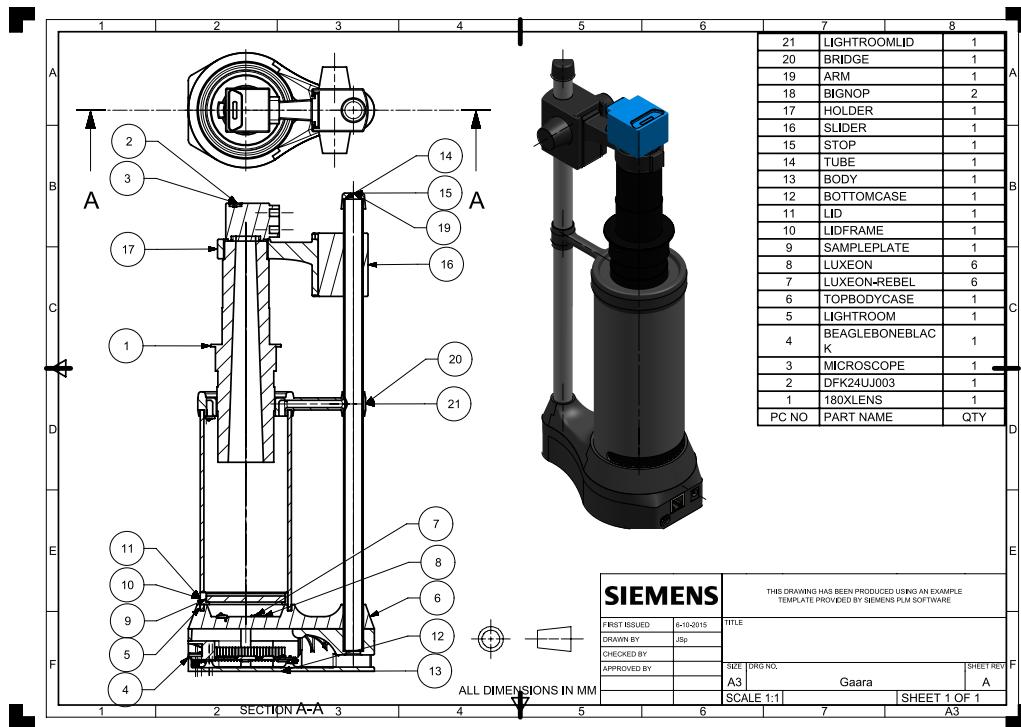


Figure 8.3: Prototype assembly

8.4 Program structure

The program is designed such that the code can be reused. This is done with an object orientated approach. The main interaction with the user is through the graphical user interface or GUI for short, which is described in detail in chapter 3.2. All the windows used by this GUI can be found in the appendix A at page 73.

The GUI is loaded from the command prompt with the following syntax:

```
#./VSA
```

The GUI is load through the **VSAMainwindow** class. This class is the main executable. figure 8.4 depicts the collaboration diagram for this class in its libraries. The structure depicted in the following paragraphs is a simplified rendition of the program structure. The complete reference file including documentation for all the namespaces and classes including inheritance and collaboration diagrams can be found in appendix Q at 363.

VSAMainwindow

This class is responsible for setting up the environment and providing user interaction. It start by loading the user setting in to memory this class is called **SoilSettings**. It then starts up communication with the microscope, taking in to a account error handling. It

handles user interruptions, menu and toolbar actions and signals from other classes, such as progress updates.

namespace SoilMath

This library is used extensively by the other libraries and consists of the following classes: FFT (Fast Fourier Transform), GA (Genetic Algorithms), Stats (Statistics), PSD (particle size distribution), NN (Neural Network), Sort (quick sort routine) and common math operations, type and error handling.

namespace SoilHardware

This library performs the hardware communication, it basically consist of two sections; Those are camera communication with the class Microscope and Beaglebone black specific classes, GPIO (General Pin Input and Output), ADC (Analogue Digital Convertor), eQEP (enhanced Quadrature Encoder Pulse) and PWM (Pulse Width Modulation). Although the Hardware class compiles on a x64 architecture the BeagleBone specific classes don't work there.

namespace SoilVision

This library performs all vision related operations and is explained in detail in section 9. It consist of the following classes: Conversion (color conversion), Enhance (Image enhancement), MorphologicalFilter, Segment and VisionDebug (Enables easy vision debugging operations).

namespace SoilAnalyzer

The SoilAnalyzer combines all the above classes in such away that the particles are segmented from the background and analyzed. It consist of the following classes: Analyzer (the actual analyzer engine), Sample (The storage container for the complete soilsample), Particle (A storage container for a single particle) and the SoilSettings (A storage class that allow easy setting movement between classes).

Serialization properties

Many of the above mentioned classes are setup in such away that the object can be (de-)serialized to a file. This allows storage and transport of the soil sample and the user settings.

GUI helper classes

The following QWidgets and QDialog classes are there to present user readable information. They consist of QOpenCVQT (convert the OpenCV image storage to QT image storage), QParticleDisplay (Particle browser), QParticleSelector (Shape selector for an individual particle), QReportGenerator (PDF report generator of the analyzed sample), DialogNN (Learning center of the Neural Network) and DialogSettings (Dialog window where the user settings can be adjusted).

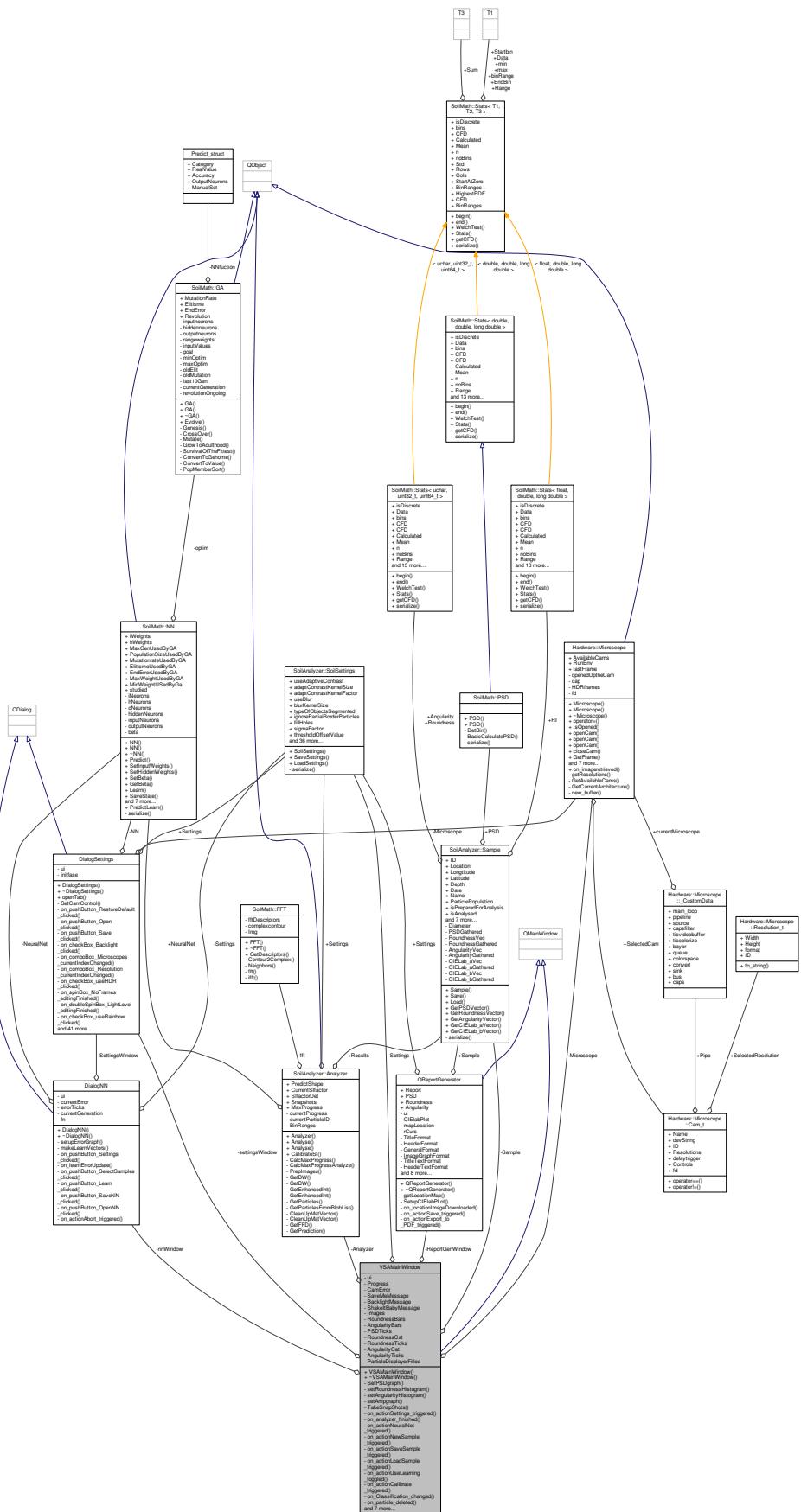


Figure 8.4: Collaboration diagram of the main program



9. 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.

9.1 Image acquisition

A thorough review of the current literature [4] identified three properties that can be used in vision based analyzing. These properties are structure (shape), color and texture (size). When looking closely at sand sample, you notice a multitude of shapes, colors and sizes, each particle is unique and differs from its neighbor. This diversity brings it own challenges. The shape of a particle determines how it will rest on the sample plate. The color and the translucency of the particle, determines how easily it can be segmented or identified from the background. Whilst the size determines the needed focus depth of the microscope.

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

Acquisition strategies

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

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

$$z\sqrt{\frac{p \times (1 - P)}{n}} \leq \alpha \rightarrow n \geq \frac{-p \times (P - 1) \times z^2}{\alpha^2} \quad (9.1)$$

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



The process of rearranging the particles, will be automated in the future. Student of the minor Machine design and major electrical engineering both taught at the university of applied sciences HAN, can choice to work on the assignment. The project are to be two weeks in length and the output will serve as input for the second prototype. These project will be executed under the auspice of MTI Holland and the author. The assignments are described in appendix C and D.

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 [8] for a more in-depth description. The choice for back lighting can be made because translucent particle are harder to segment in a bright field light. The trade off is extra processing time.

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

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



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

9.2 Image enhancement

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

Intensity image

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

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

$$\mathbf{I} = 0.2126 \times \mathbf{R} + 0.7152 \times \mathbf{G} + 0.0722 \times \mathbf{B} \quad (9.2)$$

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 9.2 and 9.2, leave the border pixels unaffected in their calculations. This offset is determined by half of the biggest kernel size. These pixels are discarded for the next step. The enhanced intensity matrix is used for particle segmentation, see section 9.3. Whilst the intensity matrix of the bright field image is used for the conversion to the CIE La*b* colorspace, as explained in section 9.3.2.

9.3 Feature extraction

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

9.3.1 Shape features

One of the main features that are of interest are those that describe shape, be it the contour or area of a particle. The feature are extracted using the algorithms below.

Segmentation

The images are segmented by calculating a threshold value. This value is determined by using the Otsu threshold. Xu et al. [7] describe that the Otsu threshold is equal to the average of the mean levels of two classes partitioned by this threshold. This threshold value can be iteratively determined.

Let \vec{h} be a vector of dimension 256 which represent a count of values in the enhanced

intensity matrix $\mathbf{I} \subset \mathbb{Z}^n \rightarrow \{0, 255\}$ with dimensions $m \times n$

$$\frac{1}{t_o} \sum_{i=1}^{t_o} \vec{h}_i = t_o - \frac{1}{256-t_o} \sum_{i=t_o}^{256} \vec{h}_i \quad (9.3)$$

In order to get more control over the segmentation process, the normal Otsu's method, as shown above is altered. A user now has the option to choose whether bright or dark object are segmented and how much the intensity values may deviation from the mean value. The mean value obtained from equation 9.3 is modified with a scaling factor and the standard deviation, as shown in equation 9.4 and 9.5.

Let $t_o \subset \mathbb{Z}^n \rightarrow \{0 \leq t \leq 255\}$ be the threshold value obtained with the iteration algorithm used to solve equation 9.3, α be the a multiplication factor given by the user and let $\vec{h} \subset \mathbb{Z}^n$ be a vector of dimension 256 which represent a count of values in the enhanced intensity matrix $\mathbf{I} \subset \mathbb{Z}^n \rightarrow \{0, 255\}$ with dimensions $m \times n$

If dark objects are to be obtained

$$t = \frac{1}{t_o} \mu + \frac{1}{2} \alpha \sigma \text{ where } \sigma = \sqrt{\frac{1}{t_o} \sum_{i=1}^t (\vec{h}_i - \mu)^2}, \text{ and } \mu = \frac{1}{t_o} \sum_{i=1}^t \vec{h}_i \quad (9.4)$$

else

$$t = \frac{1}{t_o} \mu - \frac{1}{2} \alpha \sigma \text{ where } \sigma = \sqrt{\frac{1}{256-t_o} \sum_{i=t}^{256} (\vec{h}_i - \mu)^2}, \text{ and } \mu = \frac{1}{256-t_o} \sum_{i=t}^{256} \vec{h}_i \quad (9.5)$$

Binary Image

The binary image is calculated by using the previous obtained threshold value as illustrated in equation 9.6.

Let $\mathbf{B} \subset \mathbb{Z}^n \rightarrow \{0, 1\}$ and $\mathbf{I} \subset \mathbb{Z}^n \rightarrow \{0, 255\}$ both with dimensions $m \times n$ and let $t \subset \mathbb{Z}^n \rightarrow \{0 \leq t \leq 255\}$

$$\mathbf{B} = \begin{bmatrix} \mathbf{I} \\ t \end{bmatrix} \quad (9.6)$$

Labeled blobs

If the threshold value was correctly ascertained then the binary image will consist of zeros and ones. Particles are represented by islands of connected elements with a designated value of one in an ocean of zeros. The individual particles, which are dubbed "blobs" will be identified with a two-pass connected-component labeling algorithm.

This algorithm passes each element in a binary image in a consecutive manner. When the current element belongs to a particle, it will check if previously processed neighboring pixels belong to an earlier la-

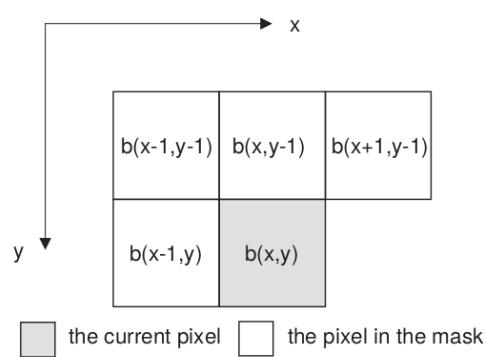


Figure 9.1: Neighboring elements Source: [2]

beled blob. If this is not the case it will assign a new label value to the current element. If it finds that one of the neighbors belong to one or more blobs, it assigns the lowest value and writes the other value to a queue. To store the connected labels.

In order to determine the lowest value of the connected component, a graphs matrix is generated, see figure 9.2. Each branch on these trees are followed till the lowest value is ascertained. All the leafs on the tree are then set to this value. These values are placed in a Look-Up-Table. With the second loop through the previously labeled image each value is replaced by looking up the lowest value in LUT.

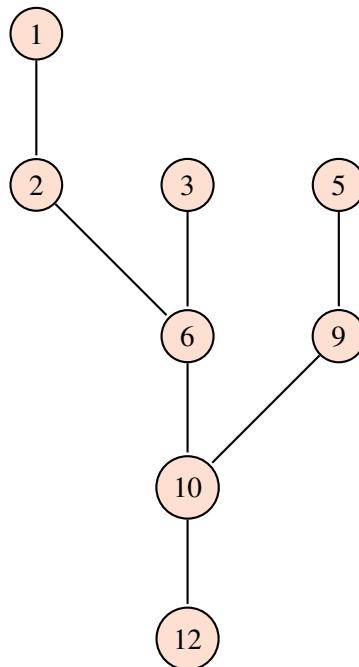


Figure 9.2: Connected Queue



A Look-Up-Table (LUT) is an array where a new value can be obtained with a simple array indexing operation. This is a very effective operation since looking up a value in memory cost less machine instructions then perform computation on each matrix element. A LUT consist of 256 elements for a unsigned byte, while a image matrix has roughly 5 million elements.

The numbering of the labels are made consecutive using an adapted quick sorting algorithm. These numbers are used to identify the individual particles in a sample. For each unique particle a type is created. This type is stored in vector and represent the sample.

For each particle a region of interest or ROI is obtained looking for the minimum and maximum value of the labeled blob within the labeled image. This ROI, is used to extract the same blob from the bright field RGB image.

Hu moments

The Hu moments are determined for the individual blobs. A Hu moment is a certain particular weighted average, or moment, of pixels in an image and can be defined as follows for a discrete image.

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

$$M_{i,j} = \sum_m \sum_n m^i n^j B_{m,n} \quad (9.7)$$

The first order or raw moment gives the total area of the particle, while the second order gives the centroid. Which is used when describing complex contour. This will be explained later on. The second order can be used to determine the orientation of the particle. By constructing a covariance matrix the rotation can be extracted from the angle of the eigenvector associated with the largest eigenvalue.

$$\Theta = \frac{1}{2} \arctan \left(\frac{2(M_{11}/M_{00} - \bar{x}\bar{y})}{(M_{20}/M_{00} - \bar{x}^2) - (M_{02}/M_{00} - \bar{y}^2)} \right) \quad (9.8)$$

Particle rotation

When the above orientation deviates from a horizontal or vertical axis, the particle is rotated. This is done by expanding the matrix in all four directions. Padding the borders. And applying the rotation matrix. When old pixels don't completely fall within the new grid. The new pixel value is obtained with linear interpolation.



The orientation of a particle is relevant because it allows for easy determination of the smallest diameter. The VSA uses a equivalent diameter to calculate the Particle Size Distribution. Normally this equivalent diameter is calculated with the assumption that the particle is round. This gives a distortion when comparing against particles that are sieved. The mesh size of the sieve allows oval particle to pass by their smallest cross-section.

Particle edge

Using the binary image from the individual particle an edge is obtained. This is done by applying the morphological operation of erosion. With this algorithm the blobs are eroded using the following principle

$$\text{Let } \mathbf{B} \subset \mathbb{Z}^n \rightarrow \{0, 1\} \text{ and } \mathbf{K} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{B} \ominus \mathbf{K} = \bigcap_{k \in \mathbf{K}} \mathbf{B}_{-b} \quad (9.9)$$

The previous obtained ROI is used to extract the edges from the individual particles. This edge is used in the Fast Fourier Transformation. In order to obtained a continuous function from the edge, a depth-first searching algorithm is applied.

Depth-first search

This starts at the top-left edge pixels and looks at its neighboring pixels. If it finds more then two neighboring pixels it stores the additional values in a queue and it moves to the first pixel new pixel. Here it performs the above decision process again. If it doesn't it only finds one neighboring pixel it know it's at a dead end. The algorithm will backtrack and start at the first branch in the queue. Subsequently storing the previously walked path, so it doesn't traverses again down this branch. This process is repeated until it find the

starting pixel. Each individual branch that is processed in this way is stored in a vector of coordinates.

Complex contour

These coordinate are stored as complex numbers where the real part is the row and the imaginary part is the column. Both are taken with respect to the center of gravity, which is obtained with the first order of the Hu moments, see earlier paragraph. The shortest vector of connected coordinates that form a loop is determined to represent the edge of the particle.



By choosing the shortest path of neighboring edge pixels as a complex contour. The roughness of the edge is represented slightly less rough then the pixel suggest that is. This effect is negated, because of the discrete nature of the digital representative of a particle compared with continuous real curve of actual particle.

9.3.2 CIE La^*b^* extraction

The conversion from the RGB color model to the CIE La^*b^* model is done for each individual particle. According to Spijker [4] its easier to ascertain a correlation between the amount of organic carbon (OC) in a soil samples when the data is presented on the chromatic a^* and b^* axis then in the RGB space. The L value represent the luminosity value of a pixel while the a^* values indicate the color on a chromatic axis between green (negative) and magenta (positive)

The relation diagram below show that there has to be a conversion from RGB to CIE XYZ first in order to traverse to CIE La^*b^* . The bright field RGB images serve as a starting point. From this image the blobs are extracted, using the previously obtained ROI or region of interests. The calculations are only performed on pixels that belong to a particle. Background pixels are ignored. After these conversions, the mean a^* and b^* values are calculated for each particle.

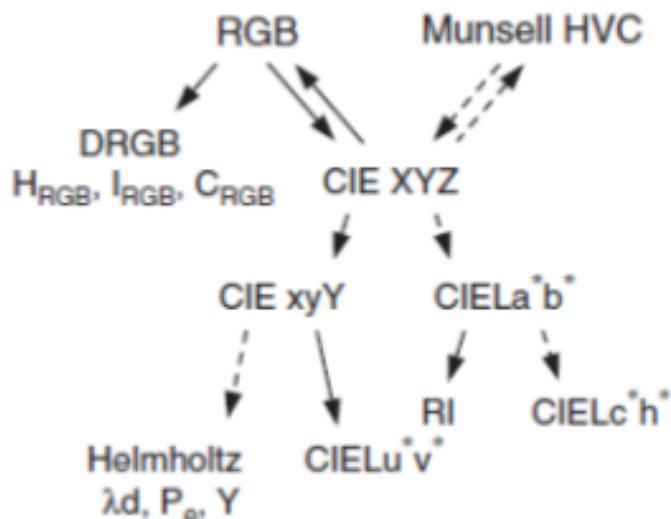


Figure 9.3: Conversion steps to different color models (Source: Viscarra Rossel, Fouad, and Walter [6])

Let $\mathbf{XYZ} \subset \mathbb{R}^n \rightarrow \{0 \leq \mathbf{XYZ}_{m,n} \leq 1\}$ and $\mathbf{RGB} \subset \mathbb{Z}^n \rightarrow \{1 \leq \mathbf{RGB}_{m,n} \leq 256\}$ both with dimension $m \times n$. The following transformation takes place for each pixel that belongs to a particle.

$$\begin{bmatrix} \mathbf{XYZ}_{i,j,1} \\ \mathbf{XYZ}_{i,j,2} \\ \mathbf{XYZ}_{i,j,3} \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119194 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{RGB}_{i,j,1} \\ \mathbf{RGB}_{i,j,2} \\ \mathbf{RGB}_{i,j,3} \end{bmatrix} \quad (9.10)$$

After the conversion to the CIE XYZ color model the particle pixels are converted to the CIE La*b* color model. This is done with the equations depicted below.

Here the following definitions stand $\mathbf{LAB} \subset \mathbb{R}^n \rightarrow \{-128 \leq \mathbf{LAB}_{m,n} \leq 128\}$ and $\mathbf{XYZ} \subset \mathbb{R}^n \rightarrow \{0 \leq \mathbf{XYZ}_{m,n} \leq 1\}$. These transformations are performed for each pixel that belongs to a particle.

$$\begin{aligned} \mathbf{LAB}_{i,j,1} &= \begin{cases} 116\left(\frac{\mathbf{XYZ}_{i,j,1}}{100}\right)^{\frac{1}{3}}, & \frac{\mathbf{XYZ}_{i,j,1}}{100} > 0.008856 \\ 903.3\left(\frac{\mathbf{XYZ}_{i,j,1}}{100}\right)^{\frac{1}{3}}, & \frac{\mathbf{XYZ}_{i,j,1}}{100} \geq 0.008856 \end{cases} \\ \mathbf{LAB}_{i,j,2} &= \left[\left(\frac{\mathbf{XYZ}_{i,j,1}}{95.047} \right)^{\frac{1}{3}} - \left(\frac{\mathbf{XYZ}_{i,j,2}}{100} \right)^{\frac{1}{3}} \right] \\ \mathbf{LAB}_{i,j,3} &= \left[\left(\frac{\mathbf{XYZ}_{i,j,2}}{100} \right)^{\frac{1}{3}} - \left(\frac{\mathbf{XYZ}_{i,j,3}}{108.883} \right)^{\frac{1}{3}} \right] \end{aligned} \quad (9.11)$$

9.3.3 Fast Fourier Descriptors

Fourier descriptor provide a way to describe a shape of a two-dimensional object by taking the Fourier transform of the boundary. Every row and column in the sparse matrix that belong to the edge of a particle, is mapped to a complex number $c + ir$ with its relation to an earlier obtained center of gravity. The process of obtaining the complex vector of coordinates is explained in detail in section 9.3.1.

The Fast Fourier Transform is obtained, with an divide and conquer technique and is based upon the *Cooley-Tukey algorithm*. According to Canale and Chapra [1] the idea behind these algorithms is that a DFT (Discrete Fourier Transform) of length N is decomposed, or "decimated" into successively smaller DFTs. This process is illustrated in figure 9.4 and 9.5 for a $m = 8$ DFT. The decimation takes place in the frequency domain and is sliced between even and odd steps. The only constraint that is placed on the computations is that $N = 2^m$ where m is the number of complex coordinates describing the contour. This complex contour first has to be validated with this constraint. This can be tested with equation 9.12, if it holds false, then the complex vector is appended with $0 + i0$.

This condition can be tested with the equation below

$$\left\lfloor \frac{\log_{10} m}{\log_{10} 2.0} \right\rfloor = \frac{\log_{10} m}{\log_{10} 2.0} \quad (9.12)$$

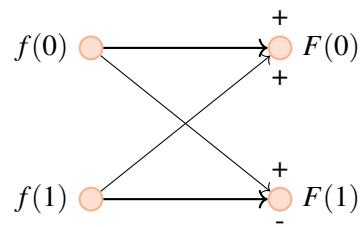
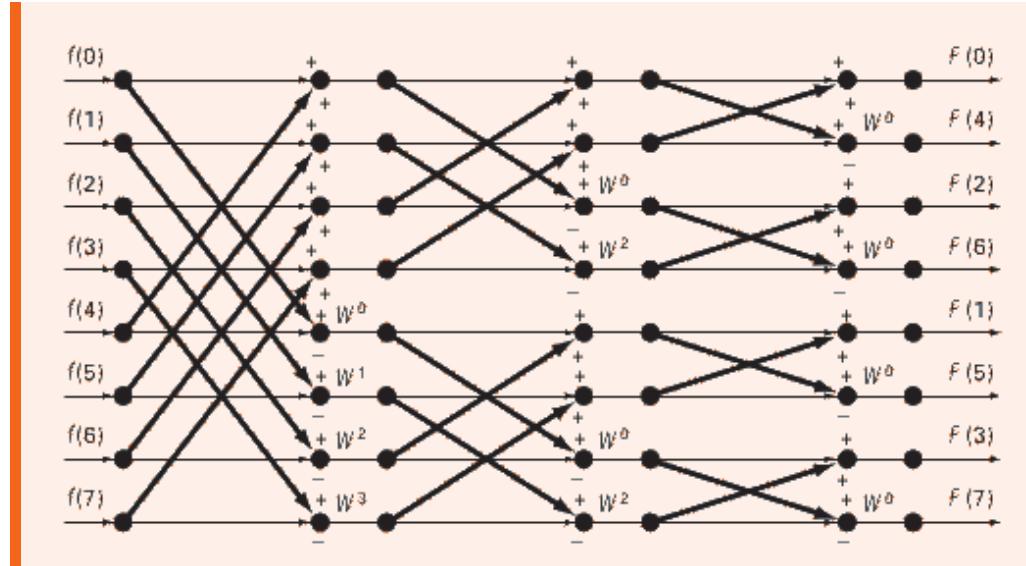


Figure 9.4: Fundamental computation of a Fast Fourier Transform

The flow diagram depicted in figure 9.5 is implemented as algorithm 9.3.1. When this

Figure 9.5: Decimation of a $m = 8$ DFT (Source: Canale and Chapra)

pseudo code is translated to a C++ object the code in appendix H at page 108 is the result.

Let \vec{c} be a complex vector of which size is assumed to be an integral of power 2
Algorithm 9.3.1: $\text{FFT}(\vec{c})$

```

 $N \leftarrow \text{size}(\vec{c})$ 
if  $N \leq 1$ 
  then return (0)
 $i \leftarrow 0$ 
 $\vec{e} \leftarrow \text{null}$ 
 $\vec{o} \leftarrow \text{null}$ 
for each  $c \in \vec{c}$ 
  do  $\begin{cases} \text{if } \text{mod } \frac{i}{2} = 0 \\ \text{then } \vec{e}_{\lfloor i/2 \rfloor} \leftarrow \vec{c}_i \\ \text{else } \vec{o}_{\lfloor i/2 \rfloor} \leftarrow \vec{c}_i \\ i \leftarrow i + 1 \end{cases}$ 
 $\text{FFT}(\vec{e})$ 
 $\text{FFT}(\vec{o})$ 
for  $k \leftarrow 0$  to  $\frac{N}{2}$ 
  do  $\begin{cases} \vec{t} \leftarrow \text{STD::POLAR}(1, \frac{-2\pi k}{N}) \cdot \vec{o}_k \\ \vec{c}_k = \vec{d}_k + \vec{t} \\ \vec{c}_{k+N/2} = \vec{d}_{k+N/2} - \vec{t} \end{cases}$ 

```

9.3.4 Particle Size Distribution

A common test to measure a particle size distribution is the sieve analysis method. In this test the PSD is obtained by allowing a soil sample to pass through a stack of sieves. These are placed in a consecutive order. Where the biggest mesh size is placed on top and the smallest below. The stack of sieves is placed in a mechanical shaker and shakes for

approximately 10[min]. This results in a segregation of the soil sample in ranges, where the demarcation is between the last mesh size that allowed a particle to pass and the first one that retained it.

Because the sieve analysis method is a commonly accepted test, the visual measured PSD has to mimic its characteristics traits. Early test suggested an offset when a soil sample was tested against a known sample. The sieved samples seemed to favor lower ranged bins when compared with the visual analyzed sample. This effect seemed to increase when the particles became less round. This implies a correlation between sphericity and the equivalent diameter, used in the particle size distribution. This theory is described in theorem 9.3.1 but needs additional proofing and or testing.

Obtaining the diameter of a particle

The size of each individual particle can be obtained by counting the pixels that belong to that particle. Determining the equivalent diameter with $\frac{4}{\pi} \sqrt{A}$ and applying a conversion from pixel to millimeter, which was obtained when the initial images where acquired, see section 9.1 paragraph acquisition. As stated above and shown in theorem 9.3.1 It still needs to be corrected for it's sphericity. This conversion factor was obtained previously when the Hu moments where calculated. For now this factor is treated as a linear correction. The complete equation is shown below.

Let $\mathbf{B} \subset \mathbb{Z}^n \rightarrow \{0, 1\}$ with dimensions $m \times n$ be a matrix with a single connected particle, where pixels belonging to that particle are assigned the value 1 and background pixels 0. And let $\alpha \subset \mathbb{R}^n \rightarrow \{0, 1\}$ as a conversion factor between pixel and millimeter. Let $\sigma \subset \mathbb{R}^n \rightarrow \{0, 1\}$ be defined as the sphericity obtained using equation ??, see section 9.4.1

$$\frac{2\alpha\sigma}{\pi} \sqrt{\sum_{i=1}^m \sum_{j=1}^n \mathbf{B}_{m,n}} \quad (9.13)$$

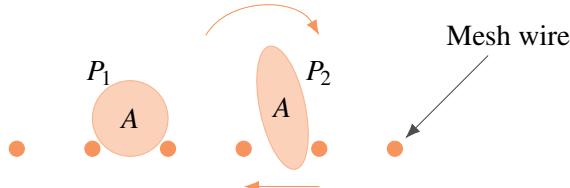


Figure 9.6: Sieving motion

Theorem 9.3.1 — Correlation between sphericity and equivalent particle diameter (obtained by sieving). The equivalent diameter of a arbitrary area A needs to be decreased with a factor when the ratio between the two axes describing that particle decreases.

Proof. The sieve mesh size can be perceived as a cross-section of a particle. A particle that passes a certain mesh size must have a smaller cross-section then the opening in the mesh itself. When cross-sections of particles are assumed as being elliptical in shape. It consist of two axis that describe it a and b . If both axes are equal it is a circle.

The orientation of a particle is relevant when it passing through the openings in a

sieve. The shaking motion, introduced by the sieve analysis method, allows elliptical shaped particle to stand upright when passing the cross-section of a sieve. Once the tip of an elliptical particle is caught in the mesh cross-section, the momentum of that same particle and the constrained of the mesh wires allows it to orient itself upright. Thus passing the mesh by its smallest cross-section. This is illustrated in figure 9.3.4.

Since both particles P_1 and P_2 have the same area but P_2 is allowed to pass to a lower sieve or bin. It can be stated that when the sphericity decreases, lower valued bins will be favored in statistical calculations.

Because the largest ellipse that fits within a square is the inscribed circle, as proven in theorem 9.3.2. An equivalent diameter will need to be corrected with a factor. This factor can be determined with the roundness value obtained by the Hu moment ascertained in section 9.3.1. ■

Theorem 9.3.2 — The largest ellipse that fits within a square is the inscribed circle. The largest ellipse that fits within a square is the inscribed circle.

Proof. It's clear that the largest ellipse touches all four sides of a square. If this isn't the case the ellipse has to be scaled in the direction where there is a gap between ellipse and square side. This will increase the area of the ellipse as well.

Its axes must lie on the square's diagonals, when the largest ellipse touches the sides. Because of the symmetric properties, . The coordinate system can in which the square is positioned can be chosen, such the corners lie on $(0, \pm 1)$ and $(\pm 1, 0)$; The ellipse will then have the equation $ax^2 + by^2 = 1$ for some a and b .

The relation between a and b can be found because the line $y = 1 - x$ must be tangent to the ellipse. Therefore the discriminant of the equation $ax^2 + b(1 - x)^2 = 1$ must be 0. This can be reworked to $b = \frac{a}{a-1}$

The area of the ellipse is $\frac{\pi}{4ab}$. In order to maximize this area $ab = \frac{a^2}{a-1}$ must be minimized. This is done by solving $\frac{d}{da} \left(\frac{a^2}{a-1} \right) = 0$ where $a = 2 \vee a = 0$. Therefore $b = \frac{2}{2-1} = 2$ and thus $a = b$. Which describes a circle. ■

Calculating the statistics

When each particle equivalent diameter is obtained with equation 9.13. Various statistical algorithms are performed. These are written in a generic template class, which allows the class to work on many different data types without being rewritten for any one.

Because a particle size distribution doesn't follow an evenly spaced bin range. The PSD class is introduced this class inherits from the statistic class. This class uses the following bin range steps:

$[0, 0.038, 0.045, 0.063, 0.075, 0.09, 0.125, 0.18, 0.25, 0.355, 0.5, 0.71, 1, 1.4, 2]$. It gives information with regard to the number of particles, the mean, minimum and maximum particle size and the standard deviation.



The statistical class, lies at the heart of many functions. For instances determining the threshold in the Otsu algorithm (section 9.3) or when applying the adaptive contrast stretch (section 9.2). But also to calculate the mean and standard deviation of the color properties (RGB, Intensity, CIE La*b*). Since this class is called many times, sometimes as much as 200×10^6 times, a great deal of effort was put in optimizing this function, while still maintaining maximum re-usability. The statistical class can be found in appendix H, page 123. Detailed information with regard to the inheritance and collaboration is given in the reference manual, in appendix Q.

9.4 Classification

The shape of a particle has a correlation with multiple properties, such as tool wear, permutation and erosion. This shape is categorized by trained humans, by comparing the particle with sixteen different classes, which are sorted according sphericity and angularity. This is illustrated in figure 9.7.

The sphericity and angularity category are both derived using vastly different techniques. These are explained in the subsection below.

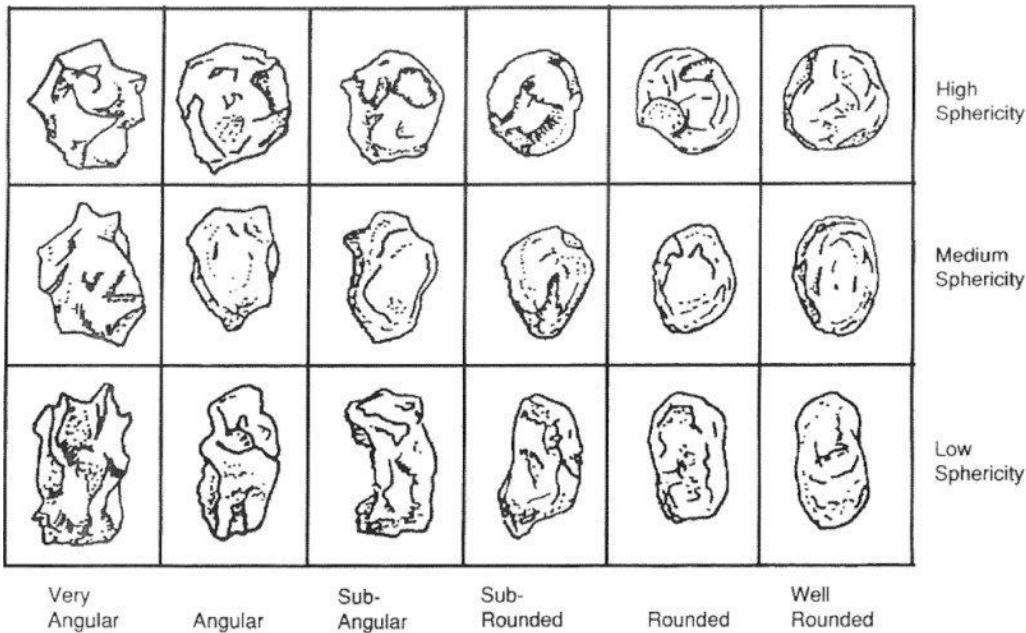


Figure 9.7: Sand shape categories (Source: Miller and Henderson [3])

9.4.1 Sphericity using Hu moments

The sphericity of a particle is the factor with which it deviates from a perfect circle. This factor can be obtained using the previous obtained Hu moments. This ensure that the shape is invariant with regard to the orientation of the particle. The equation 9.14 calculate the sphericity as a ratio and is also used as a correction factor in the PSD calculations. This can be classified using equation 9.15.

The Hu moments $M_{i,j}$ can be obtained using equation 9.7. Let σ be defined as $\sigma \subset \mathbb{R}^n \rightarrow \{0, 1\}$ and σ_{class} be defined as $\sigma_{class} \subset \mathbb{Z}^n \rightarrow \{0, 2\}$

$$\left. \begin{array}{l} \mu'_{20} = \frac{M_{20}}{M_{00}} - \left(\frac{M_{10}}{M_{00}} \right)^2 \\ \mu'_{02} = \frac{M_{02}}{M_{00}} - \left(\frac{M_{01}}{M_{00}} \right)^2 \\ \mu'_{11} = \frac{M_{11}}{M_{00}} - \frac{M_{10} M_{01}}{M_{00} M_{00}} \end{array} \right\} \sigma = \sqrt{1 - \frac{2\sqrt{\Delta\mu_2 + 4\mu'_{11}^2}}{\sqrt{\Delta\mu_2 + 4\mu'_{11}^2} + \mu'_{02} + \mu'_{20}}} \quad (9.14)$$

$$\sigma_{class} = \lfloor 2\sigma \rfloor \quad (9.15)$$

9.4.2 Angularity using a Neural Network

The angularity is categorized using a neural network or NN for short. These constructs are based upon the inner workings of a brain and can be approached as black boxes where the inner architecture is build up from interlinked neurons. Which sum up input and fires output once a threshold is obtained. They have the ability to learn and they consists of an input layer, a hidden layer and an output layer. Each layer in turn consists of individual neurons.

Neurons

The workings of a single neurons is depicted in figure 9.8 and can be described as follows; A neuron receives its input from all the neurons in a previous layer. Each of these inputs is modified with a weight ω_n . These values are summed up. When the sum of these values reach a certain threshold The neuron fires a 1. This threshold is modeled as a sigmoid value in order to have a continuous function which is needed when the neural net has to be taught.

Both $g(k) \subset \mathbb{R}^n \rightarrow \{0, 1\}$ and $k \subset \mathbb{R}^n \rightarrow \{0, 1\}$ are real numbered values. $g(k)$ is the sigmoid function and k determines the steepness of the threshold.

$$g(k) = \frac{1}{1 + e^{-k}} \quad (9.16)$$

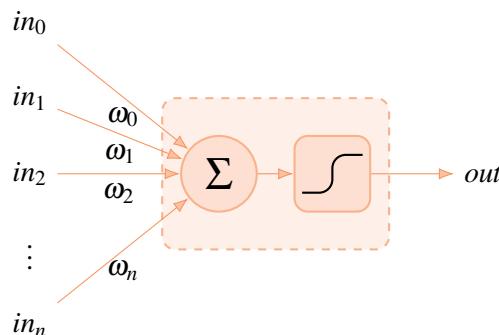


Figure 9.8: A single neuron

Programming approach

Programming of a neural network can be approached in two different ways. Object oriented, where the neurons and layers are classes or imperative oriented where the state of the program changes. The latter approach is chosen for this project. Although the code will be less reusable it will have less overhead in machine instruction which are a result of (de-)construction of the objects.

Neural Network architecture

The architecture of the Neural Network is setup according to figure 9.9. It consists of a layer of input, hidden, and output neurons. The input neurons are fed the complex Fourier Descriptors obtained with algorithm 9.3.1. The absolute values are determined from these complex number, because the magnitude of chance is the only aspect that is of interest when the angularity is to be described.

Spijker [4] describes that nine till twelve descriptor are usually enough to describe a particle contour. Any higher and the discrete nature of the pixels will start to influence

the Fourier descriptors. The number of input neurons need to resemble the amount of descriptors used. These input neurons fire feed the next hidden layers of neurons and are multiplied by a weight, which is to be determined when the network is being taught.

Each hidden neuron sums up its input and transforms it with the threshold function. These values are fired to the next layer of output neurons, which performs the same summation and transformation. The output neuron with the highest value is selected as the most probable category. There are as many output neurons as categories, which is eighteen for the Vision Soil Analyzer.

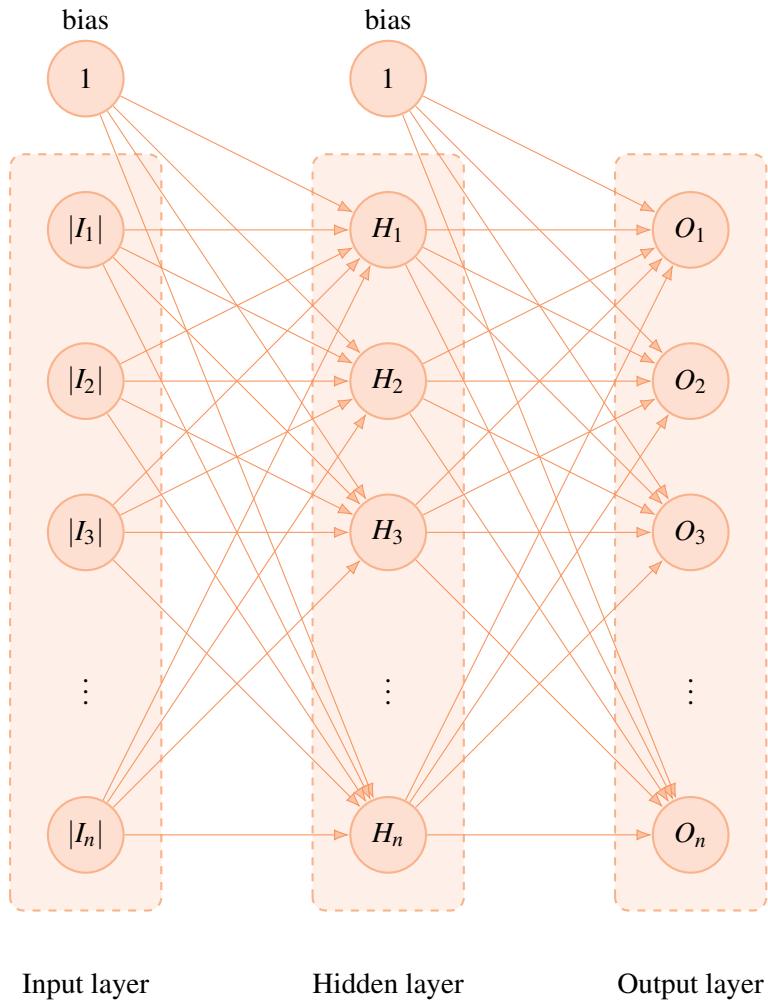


Figure 9.9: A neural network

Teaching the neural net

In order to determine the weights of the neural network it has to be taught. The network is optimized by feeding it input calculating the output and comparing it against the expected result. There are numerous algorithms and approaches to teach a neural net, the current approach is using a genetic algorithm.

9.4.3 Genetic Algorithm

Genetic algorithm are described as random search approaches, which have the ability to convergence to a global optimum. These algorithms are based on the theory of evolution,

where each sequentiality population is better adapted to its environment. The flow of the algorithm is shown in figure 9.10. This flow deviates slightly from the standard algorithm and introduces a new concept named "revolution".

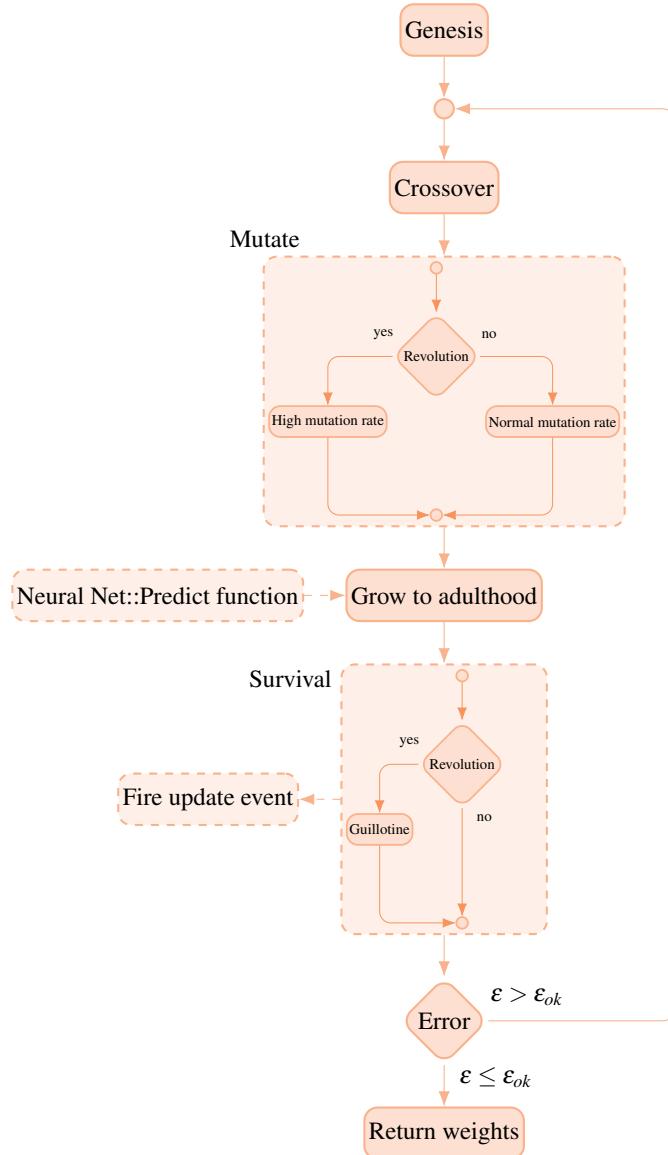
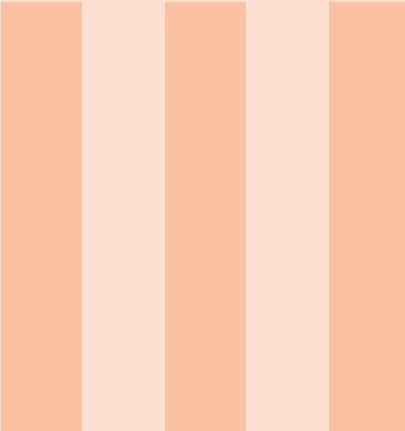


Figure 9.10: Genetic algorithm used to solve the weights of the Neural Network

Viva la revolution!

When a genetic algorithm get stuck on what is to be presumed to be local optimum, and the error isn't changed over a given number of generations, it is time to usher in a new era, one where the surviving ruling class or elite population members are led to the guillotine and the general population undergoes great chances. *Viva la revolution*¹ This spreads out the previous homogeneous population and allows it to move away from the local optimum.

¹As far as the knowledge of the author extends, the concept of moving away of a local minimum, by use of a revolution is a new concept. And because the author has been writing for 14 consecutive hours, he is willing to make some pun and call it a "revolution" in the field of Genetic Algorithms.



Verification

10	Comparing against specifications	61
11	Conclusion	65



10. Comparing against specifications

Functional requirements

Functional requirements describe the purpose of the product.

ID	Description	Achieved
F1	Quantify color	
F1.1	Determine the color in a RGB color model, from all visually (by human eye) discernible particles	Not yet tested
F1.2	Chromatic a* values must lie within 3σ	Not yet tested
F1.3	Chromatic b* values must lie within 3σ	Not yet tested
F2	Quantify texture	
F2.1	The result of an analyzed sample should fall within a probability of at least $P = 0.95\%$ when compared against the result of the same sample, but obtained using the established sieve method. These results are to be compared by Welch's t-test	Not yet tested
F2.2	PSD bins should have the same range as the fractions used in the sieving method	Yes
F3	Quantify structure	
F3.1	Roundness should be assigned in three categories	Yes
F3.2	Angularity should be assigned in six categories	Yes
F3.3	Predicted values should have at least a linear regression value of $R \geq 0.9$ when compared to expertly classified particles	Not yet tested
F4	General specifications	
F4.1	Analyze particle with sizes within the range $200\mu m \leq P_{size} \leq 2mm$	Yes
F4.2	No more than 2% of the extracted blobs may be connected particles	Not yet tested

ID	Description	Achieved
F4.3	Analyzing a sample should take no longer then 1min (ranging of sample between shot disregarded)	Yes
F5	Interaction	
F5.1	Show individual particles	Yes
F5.2	Show PSD graph with particle size in logarithmic scale	Yes
F5.3	Show Angularity in histogram	Yes
F5.4	Show Roundness in histogram	Yes
F5.5	Show probability distribution function in the histogram	Yes
F5.6	Information can be shown on a screen	Yes
F5.7	Exporting to pdf file	Yes

Table 10.1: The functional requirements compared against the product

Technical requirements

Technical requirements describe the functionality of the device with regards to its peripherals and its technical environment. They're described in such a way that they are either true or false.

ID	Description	Achieved
T1	Software environment	
T1.1	The software should run on an Linux device	Yes
T1.2	The software should be written in C++	Yes
T1.3	The software should be written as OOP and be reusable	Yes
T1.4	The software should be written with revision control	Yes
T1.5	Easily portable to Windows environment	Yes
T1.6	Easily portable to Android environment	Unknown
T2	Hardware environment	
T2.1	Should run on an ARMv7 or higher device	Yes
T2.2	Should run on a x86 or x64 device	Yes
T2.3	At least 1GHz processing power	Yes
T2.4	At least 128MB memory	Yes
T2.5	At least 2GB storage	Yes
T3	Peripherals	
T3.1	USB connection	Yes
T3.2	Ethernet LAN and/or WAN connection	Yes
T3.3	GPS unit	No
T3.4	Light controller	No
T4	General specifications	
T4.1	Sample file size should not exceed 10mb	Yes
T4.2	Guard the maximum size of particles to 2mm	Yes
T5	Prototype specifications	
T5.1	Dimensions should not exceed 400[mm] × 200[mm] × 200[mm]	Yes
T5.2	The total weight may not exceed 5[kg]	Yes

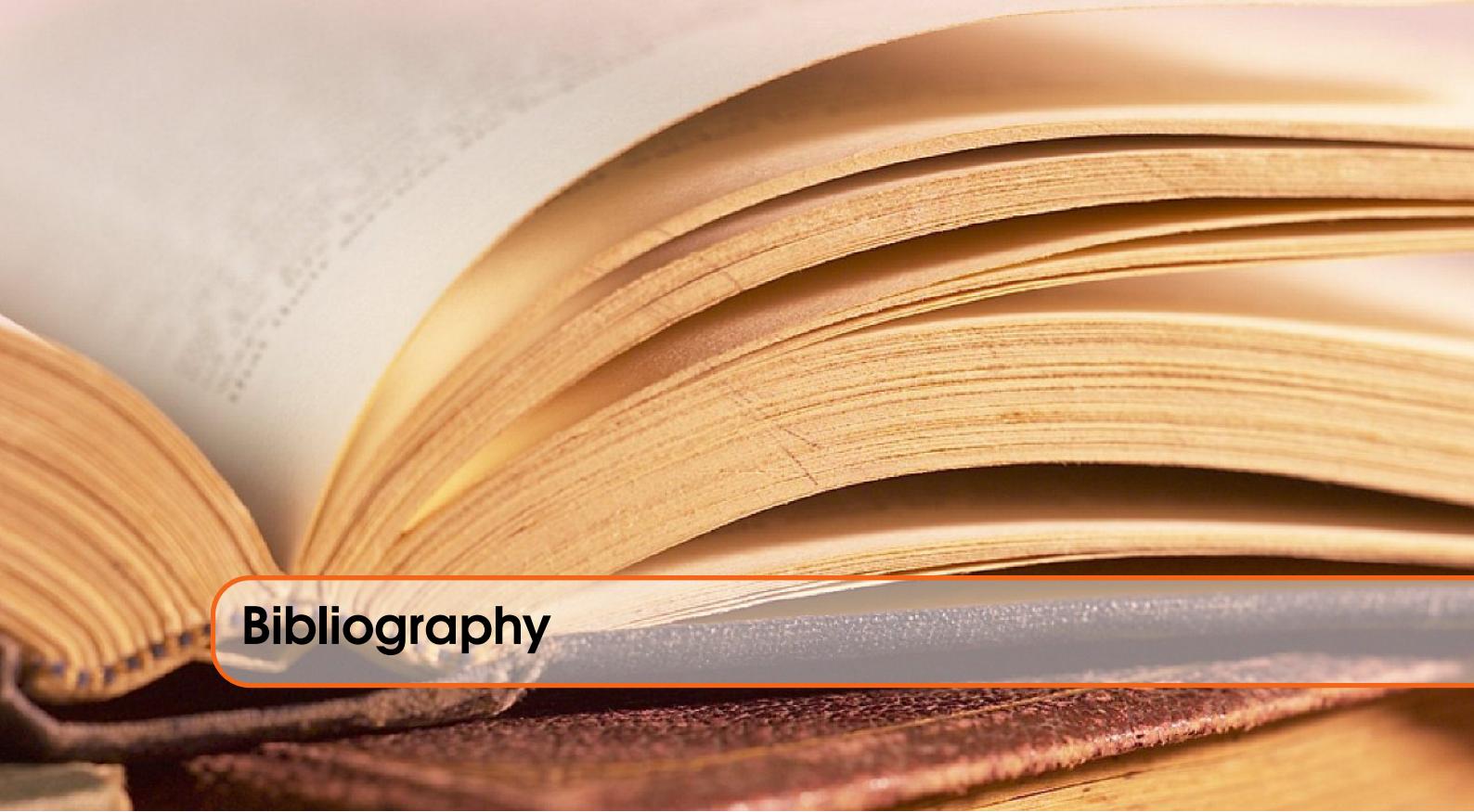
Table 10.2: Comparing technical requirements against the product



11. Conclusion

Addenda

	Bibliography	69
	Index	71
A	Graphical User Interface	73
B	Example Soil Report	77
C	HAN minor Machine design: Student assessment	83
D	HAN Electrical and electronic engineering: Student assessment	85
E	Assembly drawing	87
F	Development Environment setup	89
G	Run Environment setup	93
H	SoilMath Library	97
I	Hardware Library	151
J	Vision Library	207
K	Analyzer Library	263
L	QOpenCVQT Library	293
M	QParticleDisplay Library	297
N	QParticleSelector Library	303
O	QReportGenerator Library	309
P	Vision Soil Analyzer Program	323
Q	Reference manual	363



Bibliography

Books

- [1] R. Canale and S. Chapra. *Numerical Methods for Engineers*. McGraw-Hill Education, 2014. ISBN: 9780073397924. URL: <https://books.google.nl/books?id=avsmQEACAAJ> (cited on pages 50, 51).
- [5] *The C++ Programming Language, 4th Edition*. 4 edition. Upper Saddle River, NJ: Addison-Wesley Professional, May 19, 2013. 1368 pages. ISBN: 978-0-321-56384-2 (cited on page 34).
- [8] ir. P.A.C. Ypma. *Course Notes EVD2*. University of applied sciences, Sept. 2, 2014. 71 pages (cited on page 44).

Reports

- [4] Jelle Spijker. *Optische kenmerken van grond gebruikt bij computer vision*. Literatuurstudie. HAN University of applied sciences, 2014 (cited on pages 43, 49, 55).

Articles

- [2] Lifeng He et al. “Fast connected-component labeling”. In: *Pattern Recognition* 42.9 (Sept. 2009), pages 1977–1987. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2008.10.013. URL: <http://www.sciencedirect.com/science/article/pii/S0031320308004573> (visited on 09/01/2015) (cited on page 46).
- [3] NA Miller and JJ Henderson. “Quantifying sand particle shape complexity using a dynamic, digital imaging technique”. In: *Agronomy journal* 102.5 (2010), pages 1407–1414 (cited on page 54).

- [6] R. A. Viscarra Rossel, Y. Fouad, and C. Walter. “Using a digital camera to measure soil organic carbon and iron contents”. In: *Biosystems Engineering* 100.2 (June 2008), pages 149–159. ISSN: 1537-5110. DOI: 10.1016/j.biosystemseng.2008.02.007. URL: <http://www.sciencedirect.com/science/article/pii/S1537511008000597> (visited on 09/14/2014) (cited on page 49).
- [7] 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 45).

Index

- Acquisition, 43, 44
Acquisition strategies, 43
Adaptive contrast stretch, 45
Angularity, 54, 55
Architecture, 28

Back lighting, 44
Binary Image, 46
Blob, 46
Blur, 45
Bright field illumination, 44
BW, 46

CIE La*b*, 49
CIE XYZ, 49
Classification, 54
Complex contour, 48, 49
Connected component, 47
credentials, 25

Depth-first search, 48
Dijkstra shortest path, 29
Discrete Fourier Transform, 50

Edge, 48
Ellipse, 53
Enhancement, 44
Equivalent diameter, 48, 52

Fast Fourier Descriptors, 50
Feature extraction, 45

Features, 45
FFT, 50
Functional design, 15
Functional requirement, 16

Genetic Algorithm, 56
Graph matrix, 47
GUI, 20

Hierarchical structure, 27
Hu moments, 47, 52, 54

Imperative oriented programming, 55
Input-Process-Output, 15, 28
Input-process-output, 15
Intensity image, 44
IPO, 15, 28

Labeled blobs, 46
Labeled image, 47
Look-Up-Table, 47

Main function, 15
Manual:Administrator manual, 25
Manual:Maintenance und upgrade, 25
Manual:Teaching a neural Network, 25
Manual:User manual, 21
Maximum, 53
Mean, 53
Mechanical shaker, 51
Minimum, 53

Minimum sample size, 43
Morphological operation, 48
Morphological operation:Erosion, 48

Neural Network, 25, 55
Neural Network Architecture, 55
Neurons, 55

Object orientated programming, 40
Object oriented programming, 35, 55
Otsu's method, 45, 46, 53

Particle edge, 48
Particle Size Distibution, 51
Particle Size Distribution, 54
Pseudo code, 51

Region of Index, 47
requirement, 16
Revolution, 57
RGB, 44
Rotation, 48
Roundness, 54

Segmentation, 45
Shape, 45
SI-conversion factor, 44
Sieve analysis method, 51
Specifications, 16
Sphericity, 54
Standard deviation, 53
Statistics, 53
Structure, 43

Technical design, 27
Technical requirement, 17
Technical system, 15
Texture, 43
Threshold, 45, 46, 53
Tool wear, 54

User interface, 19

A. Graphical User Interface

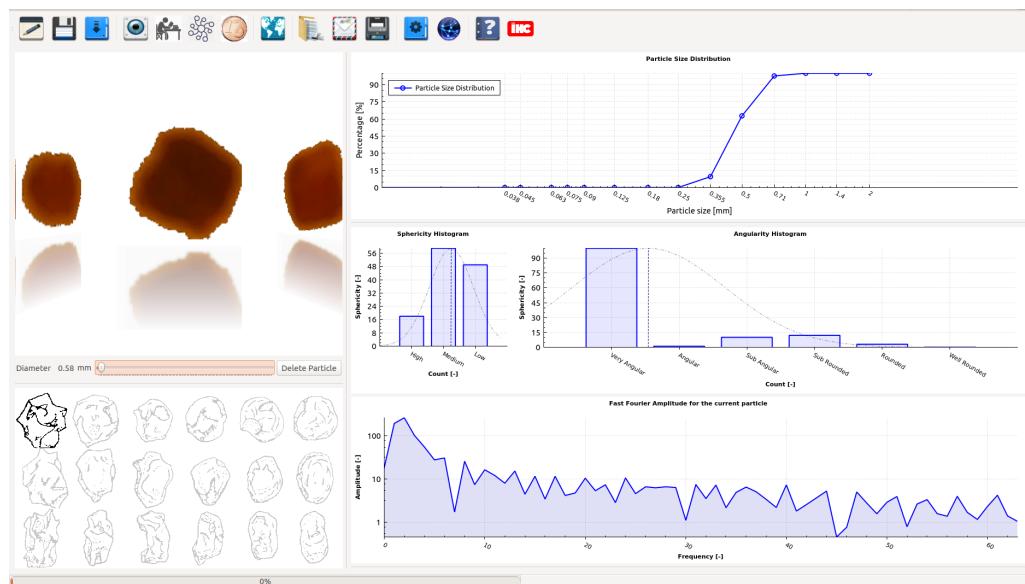


Figure A.1: Main Graphical User Interface

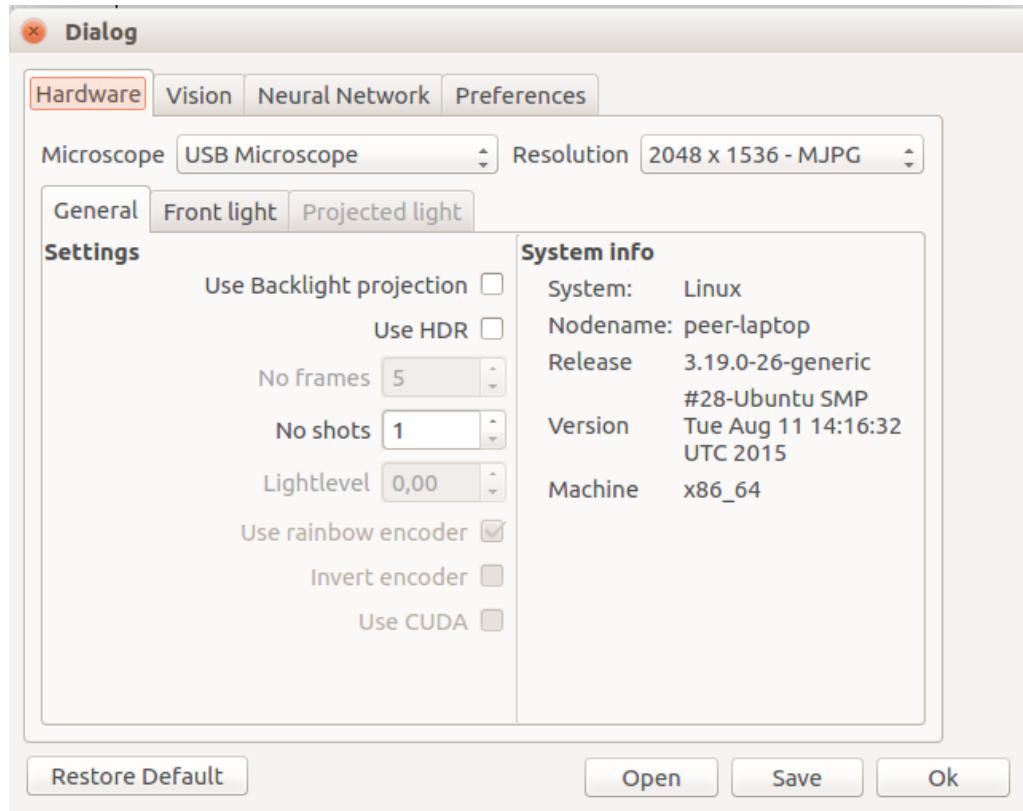


Figure A.2: Settings Hardware Interface

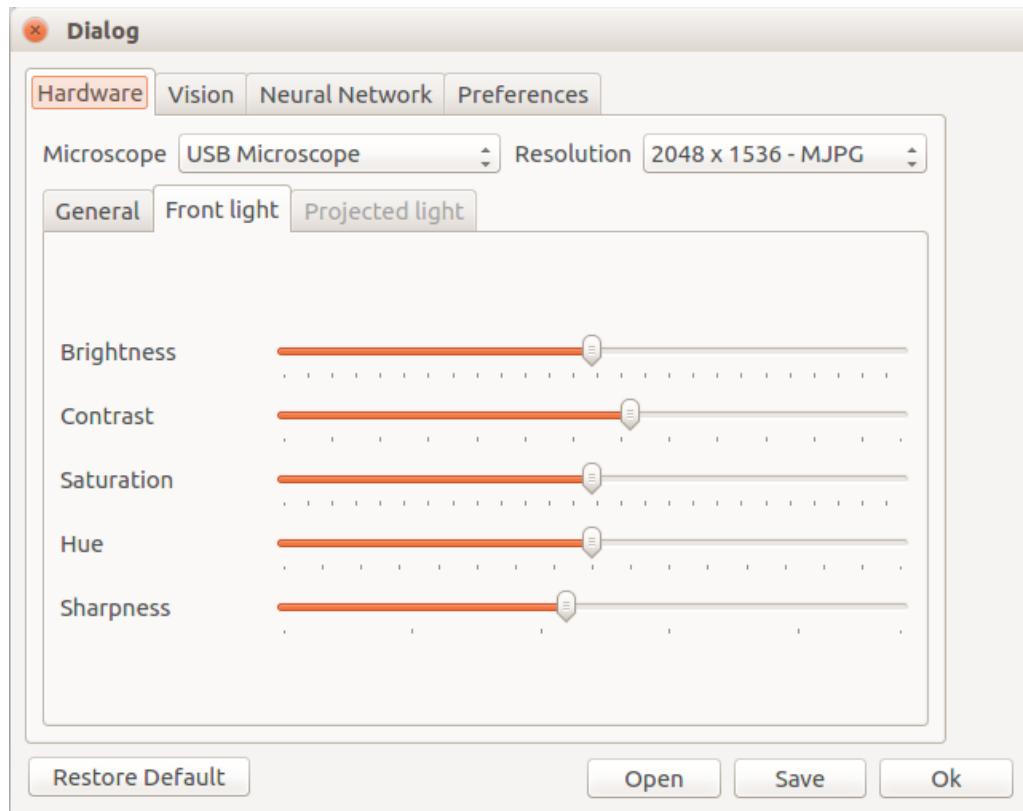


Figure A.3: Settings Hardware Cam Interface

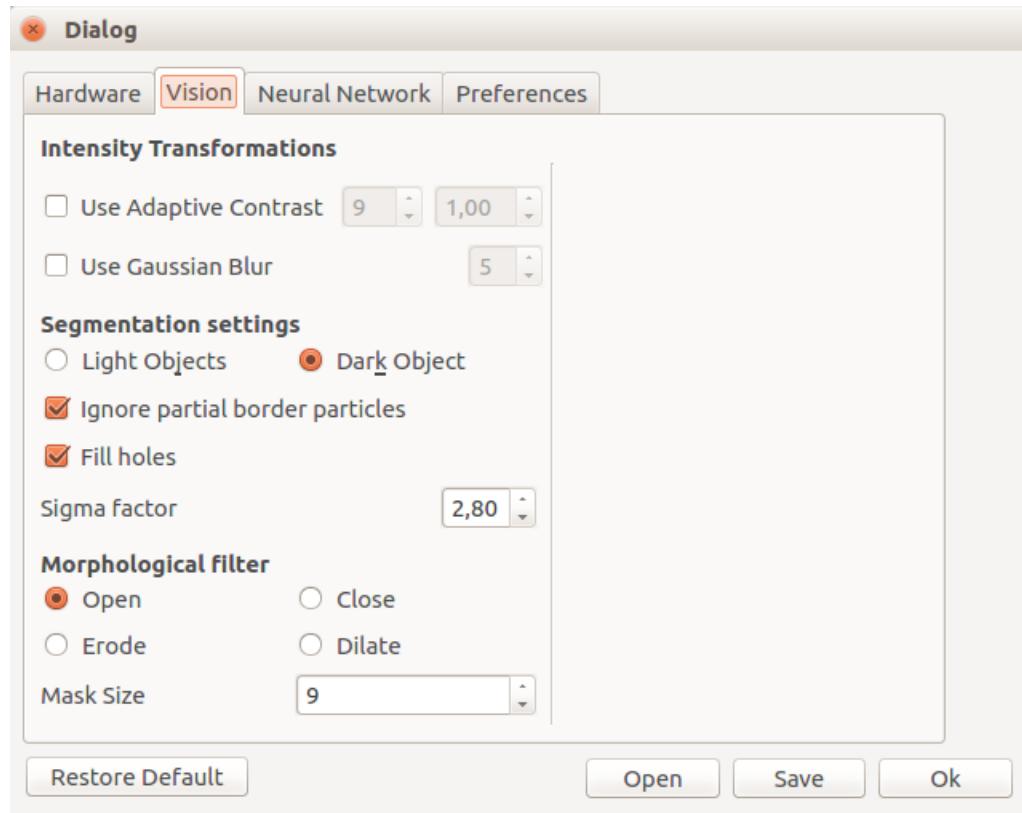


Figure A.4: Settings Vision Interface

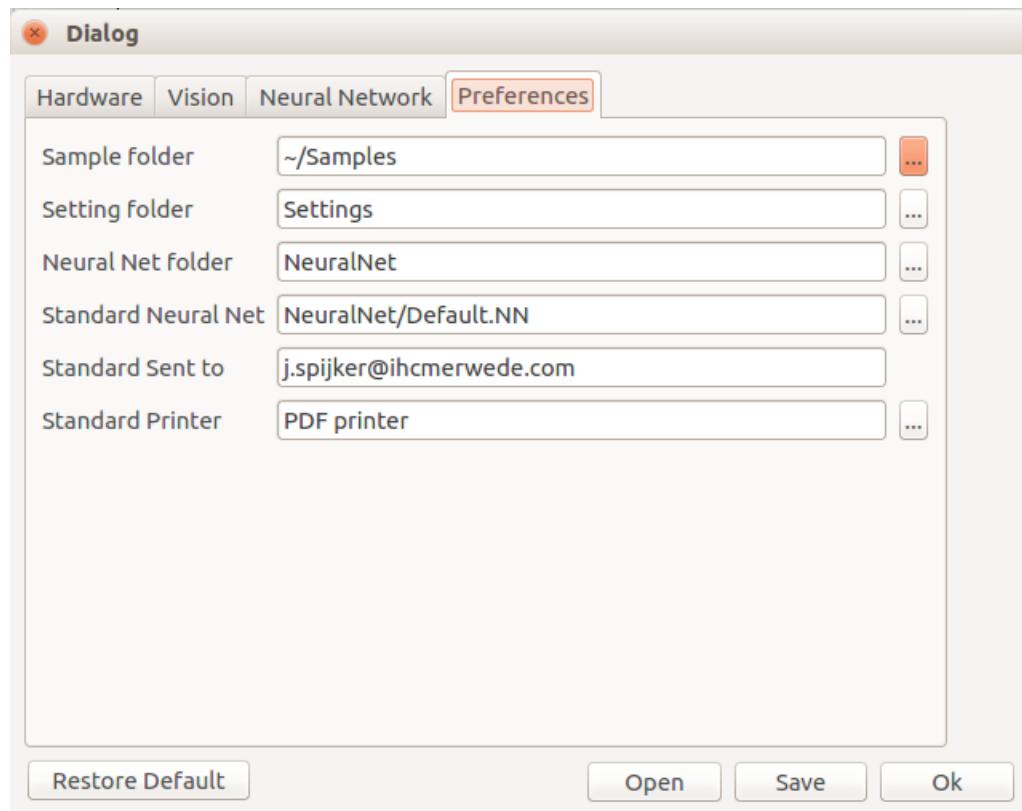


Figure A.5: Settings Preference Interface

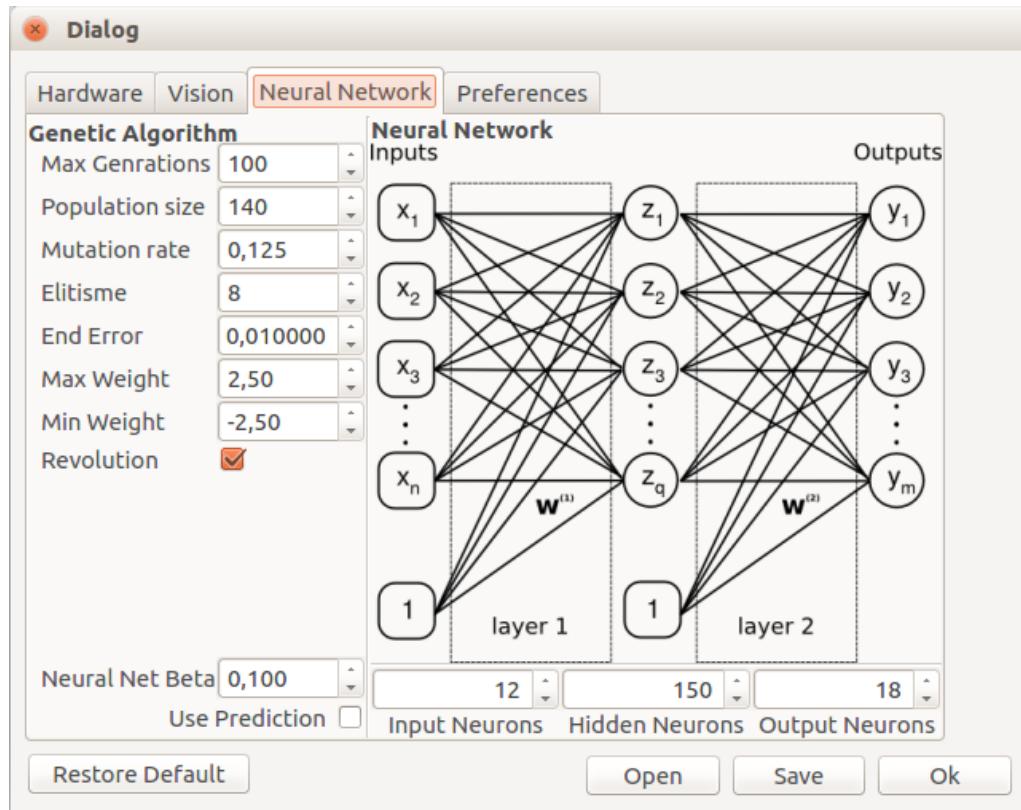


Figure A.6: Settings Neural Network Interface

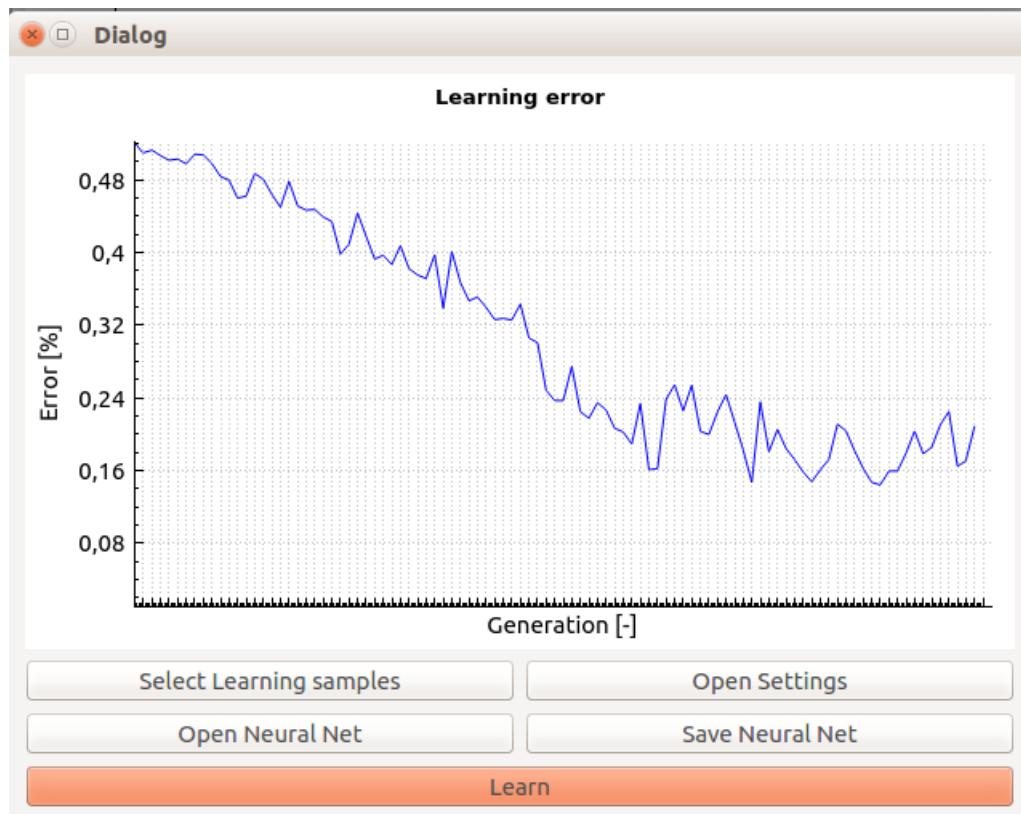
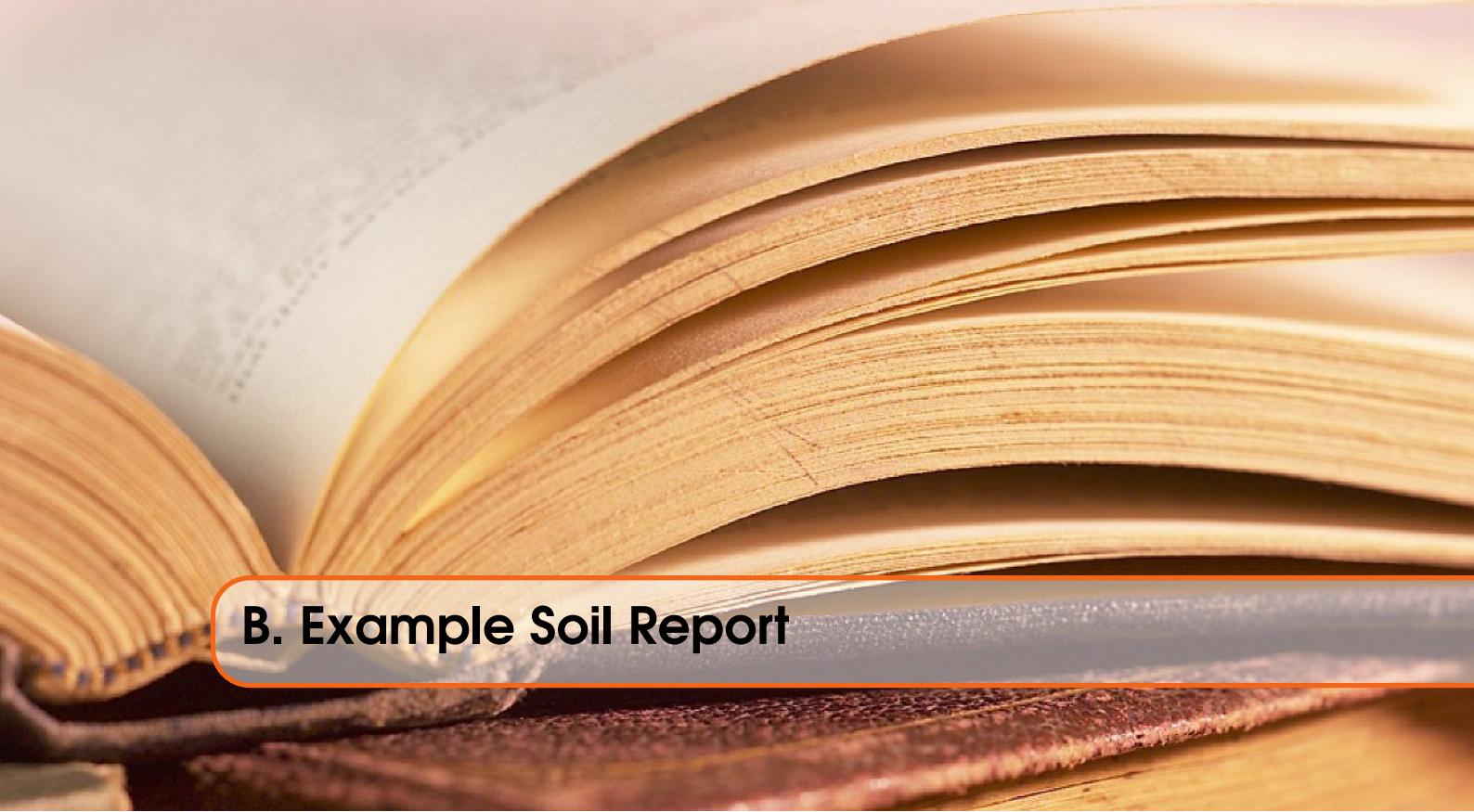


Figure A.7: Neural Network Learning Interface



B. Example Soil Report

Soil Report

Sample name:

Sample ID:

4084628568

Date:

01-09-2015

Location:

51.8849, 4.62962

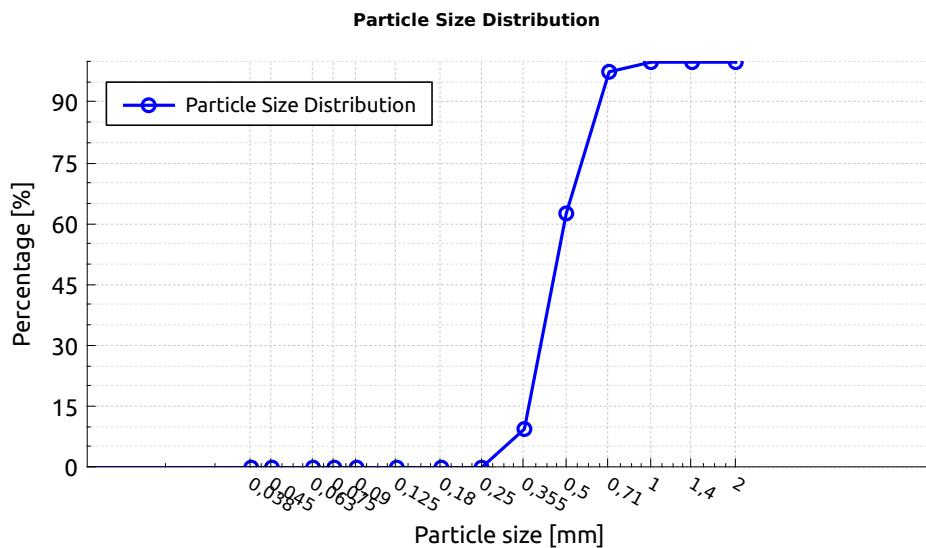
Sample depth:

0 [m]



Particle Size Distribution

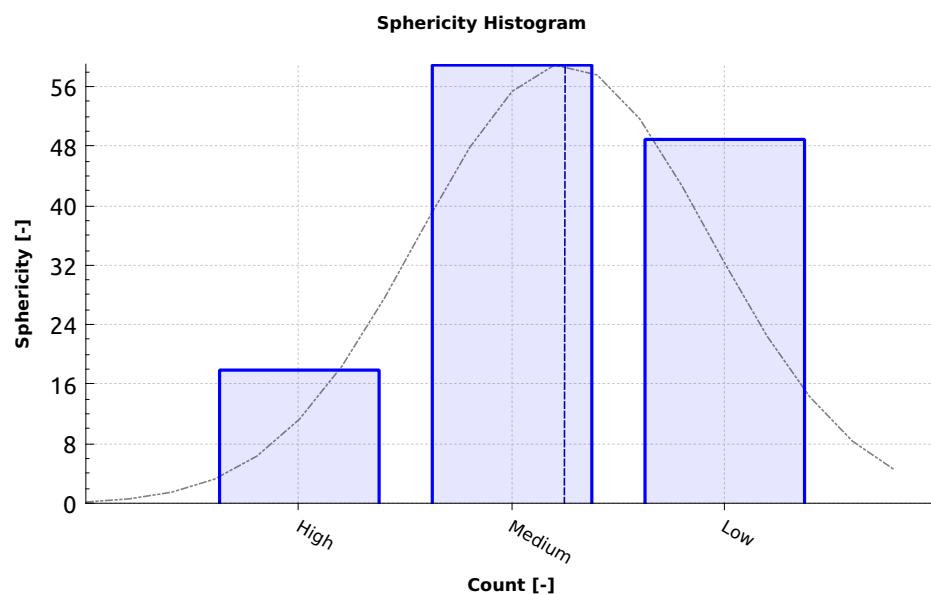
No of particles:	126
Mean:	0.673049
Minimum:	0.407876
Maximum:	1.14742
Range:	0.739541
Standard deviation:	0.142802



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

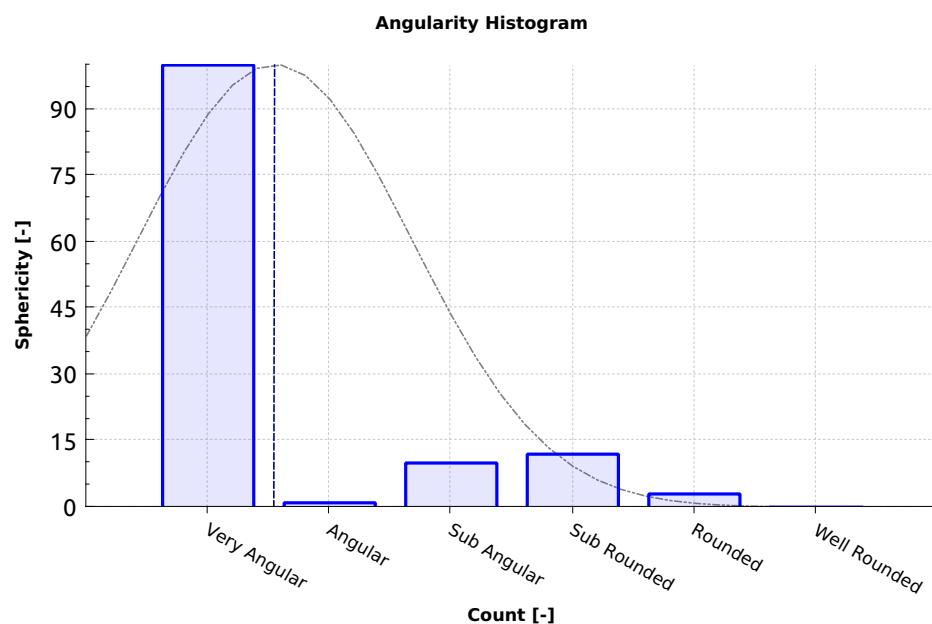
Sphericity Classification

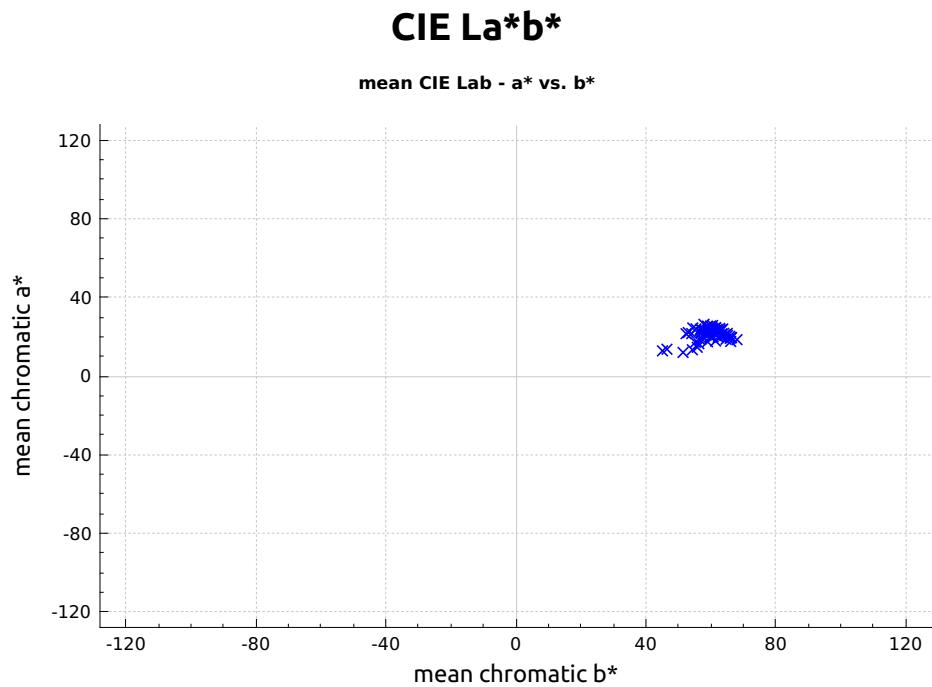
No of particles:	126
Mean:	2.24603
Minimum:	1
Maximum:	3
Range:	2
Standard deviation:	0.686451

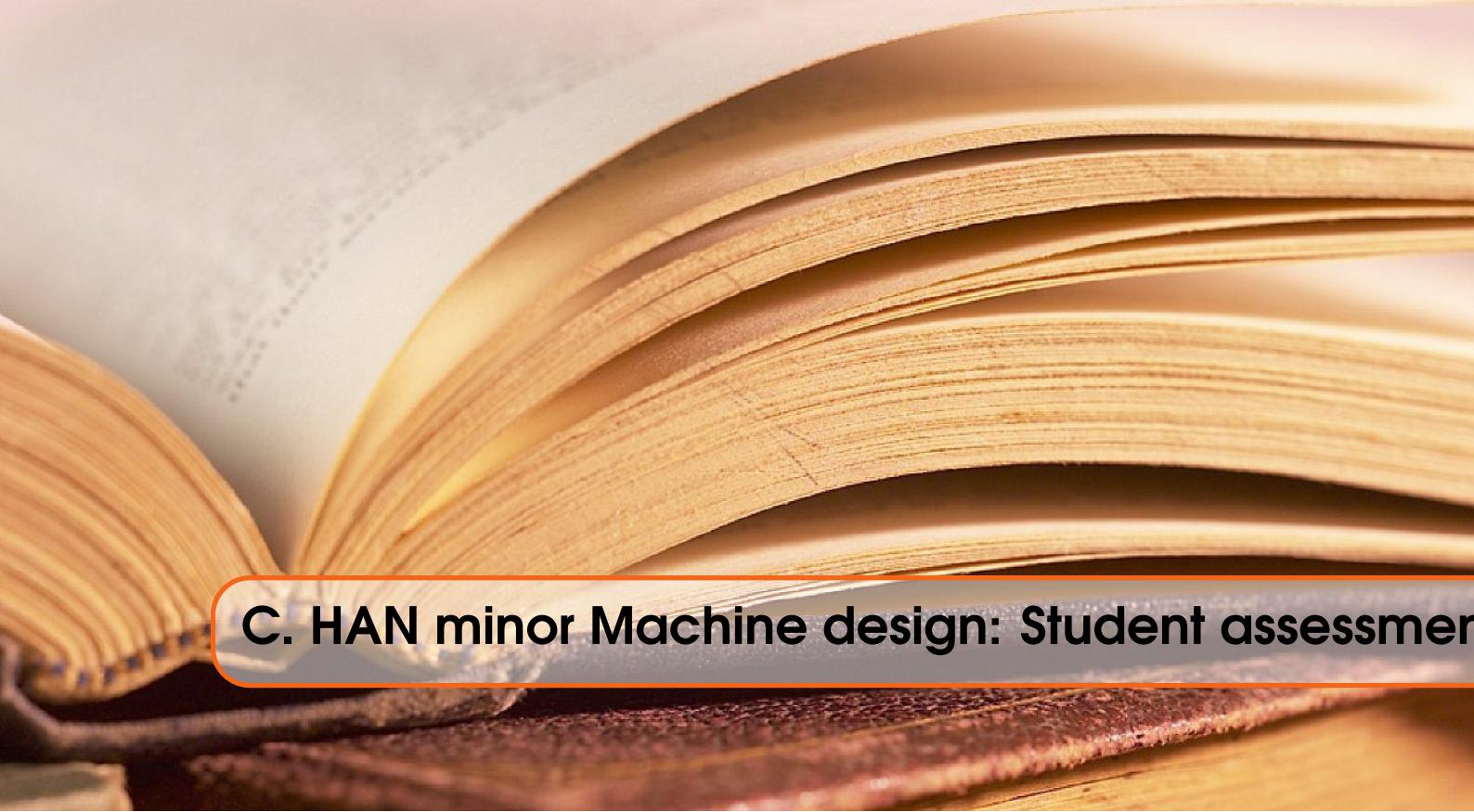


Angularity Classification

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







C. HAN minor Machine design: Student assessment

Subject: RDM Student project
Author: Jelle Spijker

Introduction

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

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

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

Problem description

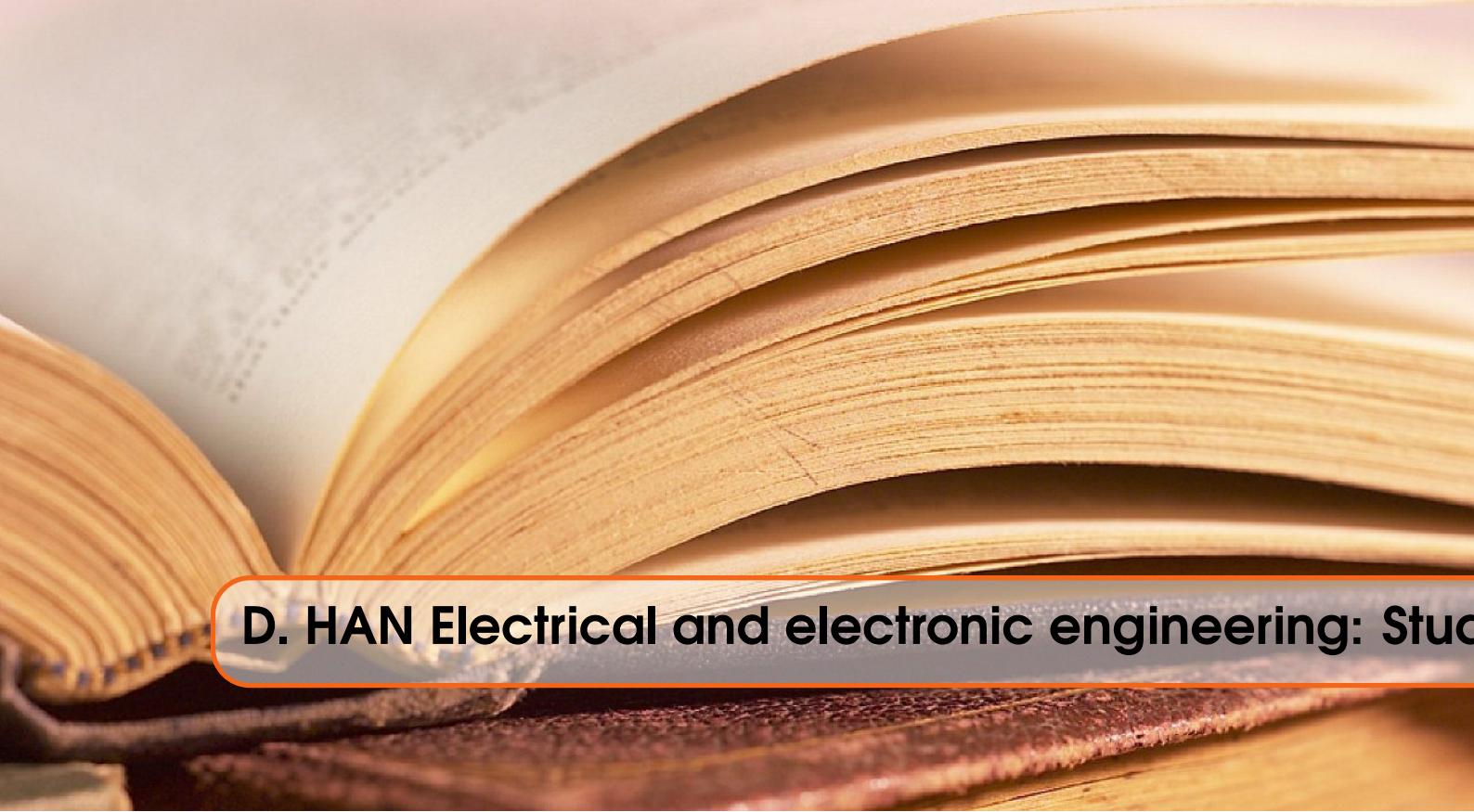
Due to the transformation from 3D particles to a discrete 2D image certain data is lost. This degradation of data introduces errors in the statistical data. One of the forms of degradations is the overlap of bigger particle onto smaller particles. These particles are identified as a particle with at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

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

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

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

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



D. HAN Electrical and electronic engineering: Stud

Subject: RDM Student project
Author: Jelle Spijker

Introduction

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

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

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

Problem description

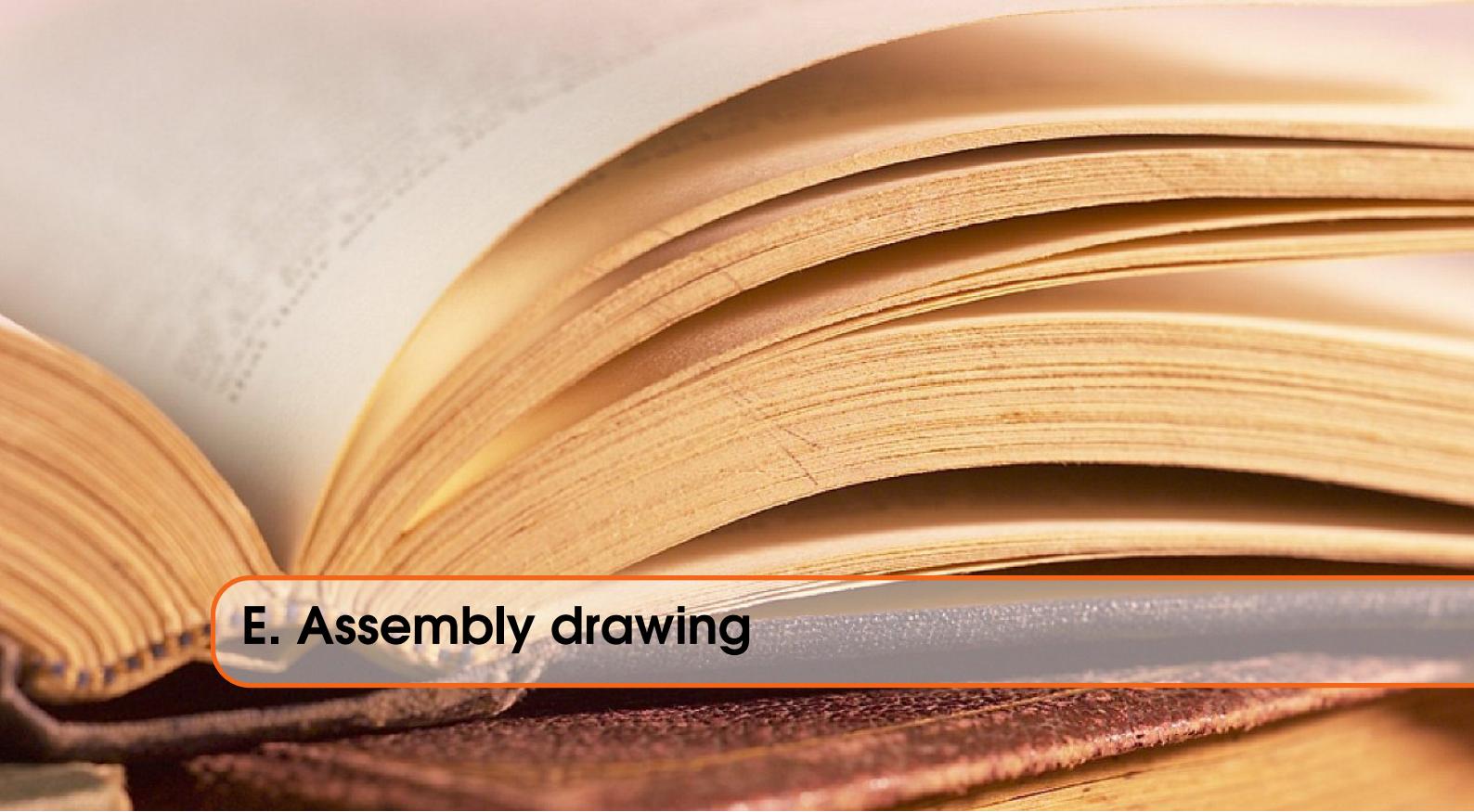
Due to the transformation from 3D particles to a discrete 2D image certain data is lost. This degradation of data introduces errors in the statistical data. One of the forms of degradations is the overlap of bigger particle onto smaller particles. These particles are identified as a particle with at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

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

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

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

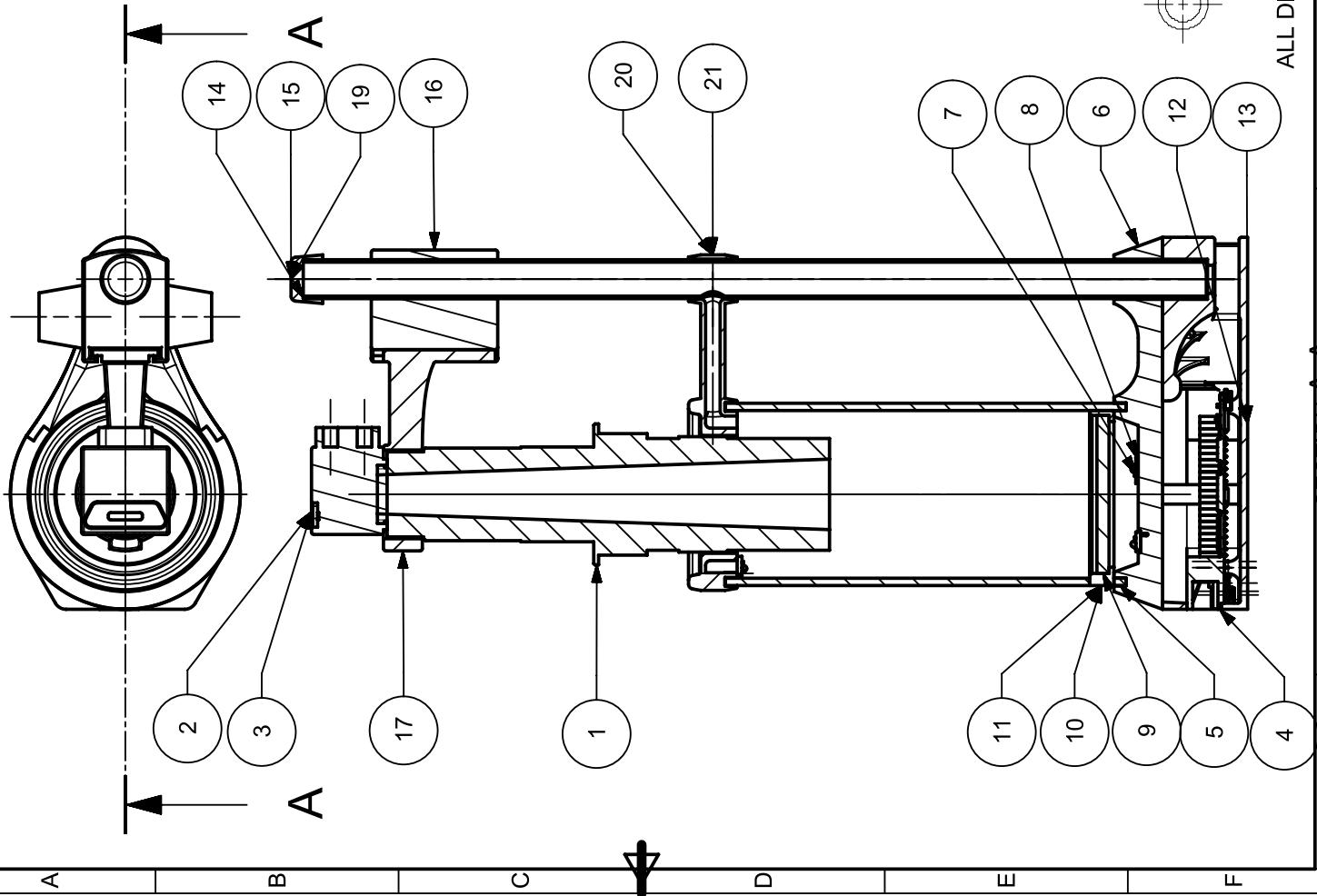
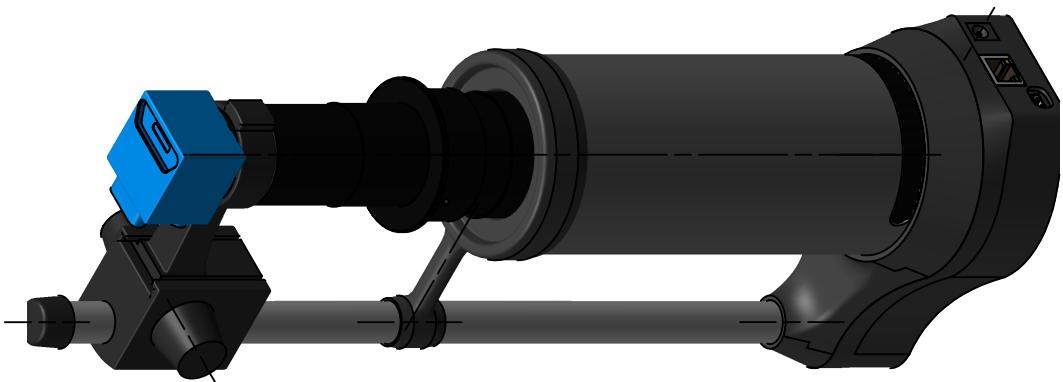
The prototype is to be CE compliant and should be build according to technical specifications. It should be described in a Technical Dossier, containing all necessary documents such as: technical drawings (according to mono system), bill of materials, calculation, analysis and design reports.



E. Assembly drawing

Chapter E. Assembly drawing

A	21	LIGHTROOMLID	1
	20	BRIDGE	1
	19	ARM	1
	18	BIGNOP	2
	17	HOLDER	1
B	16	SLIDER	1
	15	STOP	1
	14	TUBE	1
	13	BODY	1
	12	BOTTOMCASE	1
	11	LID	1
C	10	LIDFRAME	1
	9	SAMPLEPLATE	1
	8	LUXEON	6
	7	LUXEON-REBEL	6
	6	TOPBODYCASE	1
D	5	LIGHTROOM	1
	4	BEAGLEBONEBLAC	1
	3	MICROSCOPE	1
	2	DFK24UJ003	1
	1	180XLENS	1
		PC NO	PART NAME
		QTY	



SIEMENS

THIS DRAWING HAS BEEN PRODUCED USING AN EXAMPLE
TEMPLATE PROVIDED BY SIEMENS PLM SOFTWARE

FIRST ISSUED	6-10-2015	TITLE
DRAWN BY	JSp	
CHECKED BY		
APPROVED BY		
SIZE	DRG NO.	
A3		

SHEET REV
A
SHEET 1 OF 1
A3



F. Development Environment setup

Below is a list of used libraries during and their installation instructions:

Common packages

```
sudo apt-get install build-essential cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev libv4l-dev v4l-utils libqt5multimediawidgets5 clang libboost-all-dev cheese cmake-qt-gui qt-sdk libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev libtbb-dev libqt4-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils unzip libopencv-dev build-essential cmake git libgtk2.0-dev pkg-config python-dev python-numpy libdc1394-22 libdc1394-22-dev libjpeg-dev libpng12-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libxine2-dev libtiff5-dev libgstreamer0.10-dev libpython3-all-dev libpython-all-dev libbz2-dev valgrind python3-numpy
```

Nvidia driver (820M)

```
sudo apt-get purge nvidia*
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
sudo apt-get install nvidia-355 nvidia-settings
sudo nvidia-xconfig
sudo apt-get install bumblebee bbswitch-dkms primus
sudo systemctl enable bumblebeed
sudo echo "i915" » /etc/modules-load.d/modules.conf && sudo echo "bbswitch" » /etc/modules-load.d/modules.conf
sudo ln -s /usr/lib/nvidia-current /usr/lib/nvidia-355
sudo ln -s /usr/lib32/nvidia-current /usr/lib32/nvidia-355
sudo nano /etc/bumblebee/bumblebee.conf
```

```
change the parameter 'Driver=' > 'Driver=nvidia
change the parameter 'KernelDriver=nvidia-current' > 'KernelDriver=nvidia-355
restart the computer
```

CUDA

```
wget http://developer.download.nvidia.com/compute/cuda/7_0/Prod/local_installers/rpmdeb/cuda-
repo-ubuntu1410-7-0-local_7.0-28_amd64.deb
sudo dpkg -i cuda-repo-ubuntu1410-7-0-local_7.0-28_amd64.deb
sudo apt-get update
sudo apt-get install cuda
export PATH=/usr/local/cuda-7.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-7.0/lib64:$LD_LIBRARY_PATH
```

Qt and Qt Creator

```
wget http://download.qt.io/official_releases/online_installers/qt-unified-linux-x64-online.run
sudo chmod +x qt-unified-linux-x64-online.run
./qt-unified-linux-x64-online.run
```

OpenCV 3.0 beta

```
cd ~
git clone https://github.com/Itseez/opencv.git
cd opencv
mkdir release
cd release
```

Enter the following command for an NVIDIA CUDA enabled environment:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local
-D BUILD_CUDA_STUBS=ON -D BUILD_DOCS=OFF -D BUILD_JPEG=ON -D
BUILD_PNG=ON -D BUILD_TESTS=OFF -D BUILD_WITH_DEBUG_INFO=OFF
-D CUDA_FAST_MATH=ON -D ENABLE_FAST_MATH=ON -D WITH_CUBLAS=ON
WITH_OPENGL=ON WITH_QT=ON ..
```

Or the command below for a computer that has no NVIDIA CUDA capabilities:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local
-D WITH_CUDA=OFF -D BUILD_DOCS=OFF -D BUILD_JPEG=ON -D BUILD_PNG=ON
-D BUILD_TESTS=OFF -D BUILD_WITH_DEBUG_INFO=OFF -D ENABLE_FAST_MATH=ON
-D WITH_OPENGL=ON -D WITH_QT=ON ..
make -jnumber of processors
sudo make install
sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
```

ZLib

```
wget http://zlib.net/zlib-1.2.8.tar.gz
tar xf zlib-1.2.8.tar.gz
cd zlib-1.2.8
./configure
make -jnumber of processors
```

```
sudo make install
```

Folder structure

```
/  
  src  
    pictureflow-qt  
    qcustomplot  
    QOpenCVQT  
    QParticleDisplay  
    QReportGenerator  
    SoilAnalyzer  
    SoilHardware  
    SoilMath  
    SoilVision  
    Tests  
      ComparisionPictures  
      Microscope_Test  
      SoilMath_Test  
      Soil_Test  
      Vision_Test  
    tiscamera  
    VSA  
      Icons  
      Images  
      NeuralNet  
      Settings  
      SoilSamples  
      TestedSamples  
  build  
    debug  
      pictureflow-qt  
      qcustomplot  
      QOpenCVQT  
      QParticleDisplay  
      QReportGenerator  
      SoilAnalyzer  
      SoilHardware  
      SoilMath  
      SoilVision  
    install  
      pictureflow-qt  
      qcustomplot  
      QOpenCVQT  
      QParticleDisplay  
      QReportGenerator  
      SoilAnalyzer  
      SoilHardware  
      SoilMath  
      SoilVision
```

```
/  
  build  
    release  
      pictureflow-qt  
      qcustomplot  
      QOpenCVQT  
      QParticleDisplay  
      QReportGenerator  
      SoilAnalyzer  
      SoilHardware  
      SoilMath  
      SoilVision  
    Tests  
      ComparisionPictures  
      Microscope_Test  
      SoilMath_Test  
      Soil_Test  
      Vision_Test  
  doxygen  
    html  
    latex
```



G. Run Environment setup

Building the Kernel

First setup the cross-compile environment on the development PC.

Linaro

```
wget -c https://releases.linaro.org/14.09/components/toolchain/binaries/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz
tar xf gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux.tar.xz
export CC='pwd'/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux/bin/arm-linux-gnueabihf-
```

Bootloader: U-Boot

```
git clone git://git.denx.de/u-boot.git
cd u-boot/
git checkout v2015.10-rc2 -b tmp
```

Apply the Beaglebone patch against the cloned boot-loader, make sure you are in the ~/u-boot directory

```
git revert --no-edit 0a9e34056fcf86fb64e70bd281875eb7bbdbabde
wget -c https://rcn-ee.com/repos/git/u-boot-patches/v2015.10-rc2/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
```

OS ubuntu 14.04 Download the Ubuntu 14.04 OS

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/ubuntu-14.04.2-minimal-armhf-2015-06-09.tar.xz tar xf ubuntu-14.04.2-minimal-armhf-2015-06-09.tar.xz
```

Setup the SD card

```
export DISK=/dev/mmcblk0
sudo dd if=/dev/zero of=${DISK} bs=1M count=10
sudo dd if=./u-boot/MLO of=${DISK} count=1 seek=1 bs=128k
sudo dd if=./u-boot/u-boot.img of=${DISK} count=2 seek=1 bs=384k
sudo sfdisk -in-order -Linux -unit M ${DISK} <-__EOF__
1,,0x83,*
__EOF__
sudo mkfs.ext4 /dev/mmcblk0p1 -L rootfs
```

remember that initial user and password are **ubuntu** and **temppwd**

Setup pinmuxing

Enable Beaglebone overlays for kernel 4.1.x by cloning Robert C. Nelson bb.org-overlays do this on the BBB. Check if the kernel has the CONFIG_BONE_CAPEMGR=y option

```
zcat /proc/config.gz | grep CONFIG_BONE_CAPEMGR
```

Update the kernel, not needed on a fresh build.

```
cd /opt/scripts/tools
git pull
sudo ./update_kernel.sh -lts -bone-channel
```

check if the DTC version is atleast Version: DTC 1.4.1-g2341721b

```
dtc --version
```

Install the overlays

```
git clone https://github.com/RobertCNelson/bb.org-overlays.git
cd bb.org-overlays
sudo ./dtc-overlay.sh
sudo ./install.sh
```

Install the universal IO device tree for easy PWM and GPIO acces

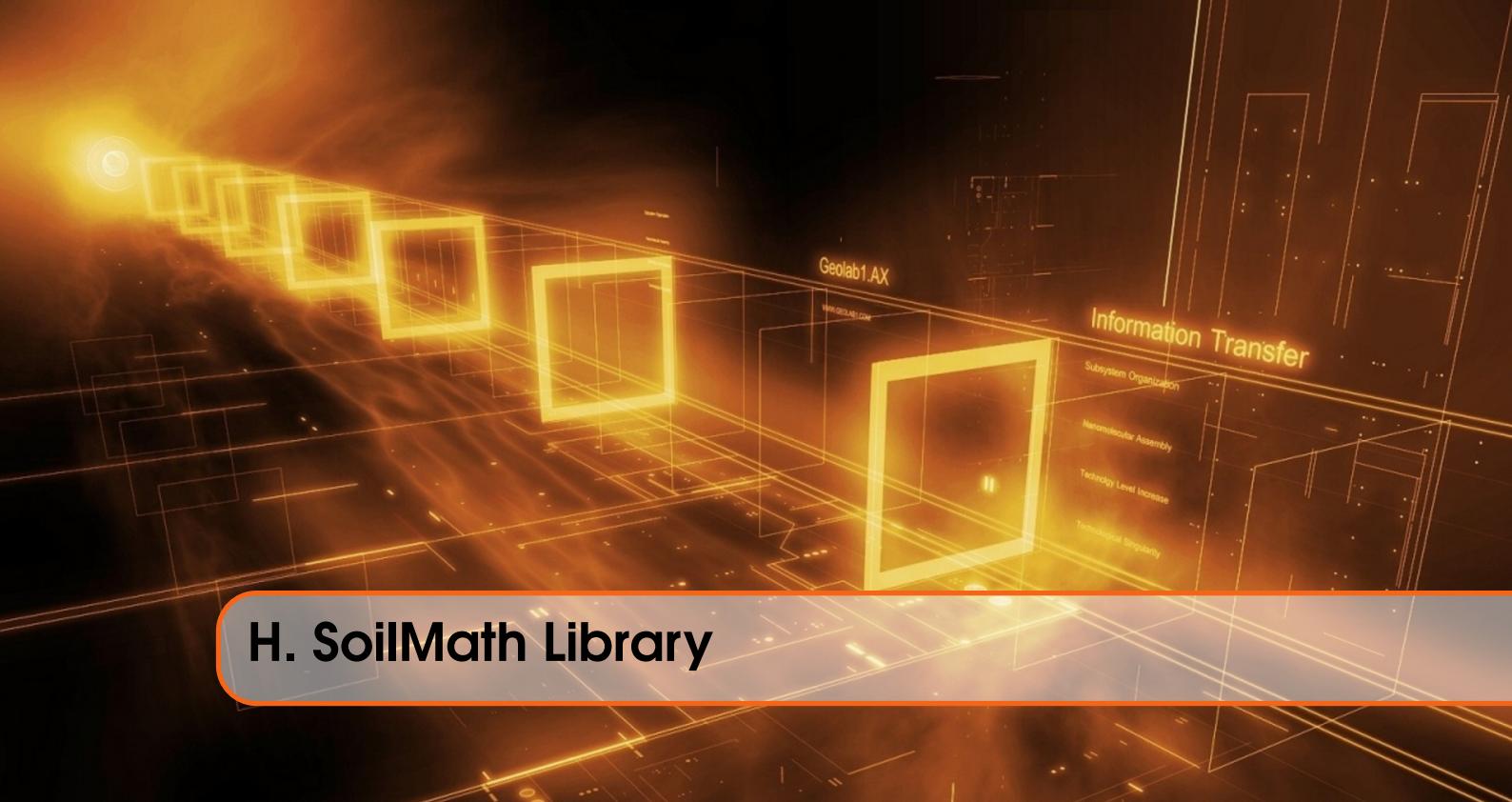
```
git clone https://github.com/cdsteinkuehler/beaglebone-universal-io.git
sudo sh -c "echo 'cape-universalm' > /sys/devices/platform/bone_capemgr/slots"
sudo cp config-pin /bin/
```

Setting up the software

The following software should be installed on the BBB

OpenCV

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local
-D WITH_CUDA=OFF -D WITH_CUFFT=OFF -D WITH_CUBLAS=OFF -D WITH_NVCUVID=OFF
-D WITH_OPENCL=OFF -D WITH_OPENCLAMDFFT=OFF -D WITH_OPENCLAMDBLAS=OFF
-D BUILD_opencv_apps=OFF -D BUILD_DOCS=OFF -D BUILD_PERF_TESTS=OFF
-D BUILD_TESTS=OFF -D ENABLE_NEON=on ..
```

H. SoilMath Library

Genetic Algorithm Class

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

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

```

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

```

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

```

155 	/*!
156 	* \brief SurvivalOfTheFittest a private function where a
157 	battle to the death
158 	* commences
159 	* The fittest population members have the best chance of
160 	survival. Death is
161 	* instigated
162 	* with a random uniform distribution. The elite members
163 	don't partake in this
164 	* destruction
165 	* The ELITISME rate indicate how many top tier members
166 	survive this
167 	* catastrophic event.
168 	* \param inputValues a InputLearnVector_t with a
169 	reference to the inputvalues
170 	* \param totalFitness a reference to the total population
171 	fitness
172 	* \return
173 	*\/
174 	bool SurvivalOfTheFittest(Population_t &pop, float &
175 	totalFitness);
176
177 	/*!
178 	* \brief PopMemberSort a private function where the
179 	members are sorted
180 	* according to
181 	* there fitness ranking
182 	* \param i left hand population member
183 	* \param j right hand population member
184 	* \return true if the left member is closer to the goal
185 	as the right member.
186 	*\/
187 	static bool PopMemberSort(PopMember_t i, PopMember_t j) {
188 	return (i.Fitness < j.Fitness);
189 }
190
191 	/*!
192 	* \brief Conversion of the value of type T to Genome_t
193 	* \details Usage: Use <tt>ConvertToGenome<Type>(type,
194 	range)</tt>
195 	* \param value The current value which should be converted
196 	to a Genome_t
197 	* \param range the range in which the value should fall,
198 	this is to have a
199 	* Genome_t
200 	* which utilizes the complete range 0000...n till 1111...
201 	n
202 	*\/
203 	template <typename T>
204 	inline Genome_t ConvertToGenome(T value, std::pair<T, T>
205 	range) {
206 	uint32_t intVal = static_cast<uint32_t>(
207 	(UINT32_MAX * (range.first + value)) / (range.second
208 	- range.first));
209 	Genome_t retVal(intVal);
210 	return retVal;
211 }
```

```

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



---


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

```

```

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

```

```

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

```



```

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

```

```

205         i = Elitisme;
206     }
207     std::uniform_real_distribution<float> dis(0, maxFitness)
208         ;
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) / (
221     maxOptim - minOptim));
221
222 // Viva la Revolution
223 if (currentGeneration > 9) {
224     double avg = 0;
225     for_each(last10Gen.begin(), last10Gen.end(), [&] (double
226         &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 restart";
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 }

```

Fast Fourier Transform Class

```

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

```

```

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

```

```

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

```

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
10 #include "FFT.h"
11
12
13 namespace SoilMath {
14 FFT::FFT() {}
15 FFT::~FFT() {}
16
17 ComplexVect_t FFT::GetDescriptors(const cv::Mat &img) {
18     if (!fftDescriptors.empty()) {
19         return fftDescriptors;
20     }
21
22     complexcontour = Contour2Complex(img, img.cols / 2, img.
23                                     rows / 2);
24
25     // Supplement the vector of complex numbers so that N = 2^
26     // m
27     uint32_t N = complexcontour.size();
28     double logN = log(static_cast<double>(N)) / log(2.0);
29     if (floor(logN) != logN) {
30         // Get the next power of 2
31     }
32
33     fftDescriptors = ComplexVect_t();
34     fftDescriptors.push_back(complexcontour);
35
36     for (int i = 1; i < N; i *= 2) {
37         complexcontour = Contour2Complex(img, img.cols / i, img.
38                                         rows / i);
39         fftDescriptors.push_back(complexcontour);
40     }
41
42     return fftDescriptors;
43 }
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

```

27     double nextLogN = floor(logN + 1.0);
28     N = static_cast<uint32_t>(pow(2, nextLogN));
29
30     uint32_t i = complexcontour.size();
31     // Append the vector with zeros
32     while (i++ < N) {
33         complexcontour.push_back(Complex_t(0.0, 0.0));
34     }
35 }
36
37 ComplexArray_t ca(complexcontour.data(), complexcontour.
38     size());
39 fft(ca);
40 fftDescriptors.assign(std::begin(ca), std::end(ca));
41 return fftDescriptors;
42
43 iContour_t FFT::Neighbors(uchar *0, int pixel, uint32_t
44     columns,
45     uint32_t rows) {
46     long int LUT_nBore[8] = {-columns + 1, -columns, -columns
47     - 1, -1,
48     columns - 1, columns, 1 +
49     columns, 1};
50     iContour_t neighbors;
51     uint32_t pEnd = rows * columns;
52     uint32_t count = 0;
53     for (uint32_t i = 0; i < 8; i++) {
54         count = pixel + LUT_nBore[i];
55         while (count >= pEnd && i < 8) {
56             count = pixel + LUT_nBore[++i];
57         }
58         if (i >= 8) {
59             break;
60         }
61         if (0[count] == 1)
62             neighbors.push_back(count);
63     }
64     return neighbors;
65 }
66
67 ComplexVect_t FFT::Contour2Complex(const cv::Mat &img, float
68     centerCol,
69     float centerRow) {
70     uchar *0 = img.data;
71     uint32_t pEnd = img.cols * img.rows;
72
73     std::deque<std::deque<uint32_t>> sCont;
74     std::deque<uint32_t> eList;
75
76     // Initialize the queue
77     for (uint32_t i = 0; i < pEnd; i++) {
78         if (0[i] == 1) {
79             std::deque<uint32_t> tmpQ;
80             tmpQ.push_back(i);
81             sCont.push_back(tmpQ);
82         }
83     }
84
85     while (!sCont.empty()) {
86         std::deque<uint32_t> &tmpQ = sCont.front();
87         uint32_t i = tmpQ.back();
88         sCont.pop_front();
89
90         for (int j = 0; j < 8; j++) {
91             int n = i + LUT_nBore[j];
92             if (n < 0 || n >= pEnd)
93                 continue;
94             if (0[n] == 1) {
95                 0[n] = 0;
96                 eList.push_back(n);
97             }
98         }
99     }
100
101    std::swap(sCont, eList);
102
103    ComplexVect_t result;
104    result.reserve(eList.size());
105    for (uint32_t i : eList)
106        result.push_back(Complex_t((centerCol - img.cols / 2) +
107            (i % img.cols) * 2.0 / img.cols, (centerRow - img.rows / 2) +
108            (i / img.cols) * 2.0 / img.rows));
109
110    return result;
111 }
```

```

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

```

```

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

```

174 }

175 }

Neural Network Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11
12 #include <stdint.h>
13 #include <vector>
14 #include <string>
15 #include <fstream>
16 #include <boost/archive/xml_iarchive.hpp>
17 #include <boost/archive/xml_oarchive.hpp>
18 #include <boost/serialization/vector.hpp>
19 #include <boost/serialization/version.hpp>
20
21 #include "GA.h"
22 #include "MathException.h"
23 #include "SoilMathTypes.h"
24 #include "FFT.h"
25
26 #include <QtCore/QObject>
27
28 namespace SoilMath {
29 /*!
30  * \brief The Neural Network class
31  * \details This class is used to make prediction on large
32  * data set. Using self
33  * learning algoritmes
34  */
35 class NN : public QObject {
36     Q_OBJECT
37
38 public:
39 /*!
40  * \brief NN constructor for the Neural Net
41  * \param inputneurons number of input neurons
42  * \param hiddenneurons number of hidden neurons
43  * \param outputneurons number of output neurons
44  */
45     NN(uint32_t inputneurons, uint32_t hiddenneurons, uint32_t
46         outputneurons);
47
48     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
58     * the input which are
59     * the complex values descpriing the contour in the
60     * frequency domein serve as
61     * input. The absolute value of these im. number because I
62     * 'm not interrested
63     * in the orrientation of the particle but more in the
64     * degree of variations.
65     * \param input vector of complex input values, these're
66     * the Fourier
67     * descriptors
68     * \return a real valued vector of the output neurons
69     */
70     Predict_t Predict(ComplexVect_t input);
71
72     /*!
73     * \brief PredictLearn a static function used in learning
74     * of the weights
75     * \details It starts a new Neural Network object and
76     * passes all the
77     * paramaters in to this newly created object. After this
78     * the predict function
79     * is called and the value is returned. This work around
80     * was needed to pass
81     * the neural network to the Genetic Algorithm class.
82     * \param input a complex vector of input values
83     * \param inputweights the input weights
84     * \param hiddenweights the hidden weights
85     * \param inputneurons the input neurons
86     * \param hiddenneurons the hidden neurons
87     * \param outputneurons the output neurons
88     * \return
89     */
90     static Predict_t PredictLearn(ComplexVect_t input,
91         Weight_t inputweights,
92                     Weight_t hiddenweights,
93                     uint32_t inputneurons,
94                     uint32_t hiddenneurons,
95                     uint32_t outputneurons);
96
97     /*!
98     * \brief SetInputWeights a function to set the input
99     * weights
100    * \param value the real valued vector with the values
101    */
102    void SetInputWeights(Weight_t value) { iWeights = value; }
103
104    /*!
105     * \brief SetHiddenWeights a function to set the hidden
106     * weights

```

```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

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

Statistical Class

```

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

```

```

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

```

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

```

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

```

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

```

```

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

```

```

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

```

```

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



---


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

```

```

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

```

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

General project files

```

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

```

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


---


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


---


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

```

22     }
23 }
24
25 inline uint16_t Max(uint16_t a, uint16_t b) { return (a > b)
26     ? a : b; }
27
28 inline uint16_t Max(uint16_t a, uint16_t b, uint16_t c,
29     uint16_t d) {
30     return (Max(a, b) > Max(c, d)) ? Max(a, b) : Max(c, d);
31 }
32
33 inline uint16_t Min(uint16_t a, uint16_t b) { return (a < b)
34     ? a : b; }
35
36
37 static inline double quick_pow10(int n) {
38     static double pow10[19] = {1, 10, 100, 1000, 10000,
39     100000, 1000000, 10000000,
40     100000000, 1000000000, 10000000000,
41     100000000000, 1000000000000,
42     10000000000000, 100000000000000,
43     1000000000000000, 10000000000000000};
44
45     return pow10[(n >= 0) ? n : -n];
46 }
47
48 // Source:
49 // http://martin.ankerl.com/2012/01/25/optimized-
50 //   approximative-pow-in-c-and-cpp/
51
52 static inline double fastPow(double a, double b) {
53     union {
54         double d;
55         int x[2];
56     } u = {a};
57     u.x[1] = (int)(b * (u.x[1] - 1072632447) + 1072632447);
58     u.x[0] = 0;
59     return u.d;
60 }
61
62
63 static inline double quick_pow2(int n) {
64     static double pow2[256] = {
65         0, 1, 4, 9, 16, 25, 36, 49,
66         64, 81,
67         100, 121, 144, 169, 196, 225, 256, 289,
68         324, 361,
69         400, 441, 484, 529, 576, 625, 676, 729,
70         784, 841,
71         900, 961, 1024, 1089, 1156, 1225, 1296, 1369,
72         1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025,
73         2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809,
74         2904, 3025, 3144, 3261, 3384, 3509, 3636, 3761,
75         3884, 4009, 4136, 4261, 4384, 4509, 4636, 4761,
76         4884, 5009, 5136, 5261, 5384, 5509, 5636, 5761,
77         5884, 6009, 6136, 6261, 6384, 6509, 6636, 6761,
78         6884, 7009, 7136, 7261, 7384, 7509, 7636, 7761,
79         7884, 8009, 8136, 8261, 8384, 8509, 8636, 8761,
80         8884, 9009, 9136, 9261, 9384, 9509, 9636, 9761,
81         9884, 10000, 10136, 10261, 10384, 10509, 10636, 10761,
82         10884, 11009, 11136, 11261, 11384, 11509, 11636, 11761,
83         11884, 12009, 12136, 12261, 12384, 12509, 12636, 12761,
84         12884, 13009, 13136, 13261, 13384, 13509, 13636, 13761,
85         13884, 14009, 14136, 14261, 14384, 14509, 14636, 14761,
86         14884, 15009, 15136, 15261, 15384, 15509, 15636, 15761,
87         15884, 16009, 16136, 16261, 16384, 16509, 16636, 16761,
88         16884, 17009, 17136, 17261, 17384, 17509, 17636, 17761,
89         17884, 18009, 18136, 18261, 18384, 18509, 18636, 18761,
90         18884, 19009, 19136, 19261, 19384, 19509, 19636, 19761,
91         19884, 20009, 20136, 20261, 20384, 20509, 20636, 20761,
92         20884, 21009, 21136, 21261, 21384, 21509, 21636, 21761,
93         21884, 22009, 22136, 22261, 22384, 22509, 22636, 22761,
94         22884, 23009, 23136, 23261, 23384, 23509, 23636, 23761,
95         23884, 24009, 24136, 24261, 24384, 24509, 24636, 24761,
96         24884, 25009, 25136, 25261, 25384, 25509, 25636, 25761,
97         25884, 26009, 26136, 26261, 26384, 26509, 26636, 26761,
98         26884, 27009, 27136, 27261, 27384, 27509, 27636, 27761,
99         27884, 28009, 28136, 28261, 28384, 28509, 28636, 28761,
100        28884, 29009, 29136, 29261, 29384, 29509, 29636, 29761,
101        29884, 29999, 30136, 30261, 30384, 30509, 30636, 30761,
102        30884, 30999, 31136, 31261, 31384, 31509, 31636, 31761,
103        31884, 31999, 32136, 32261, 32384, 32509, 32636, 32761,
104        32884, 32999, 33136, 33261, 33384, 33509, 33636, 33761,
105        33884, 33999, 34136, 34261, 34384, 34509, 34636, 34761,
106        34884, 34999, 35136, 35261, 35384, 35509, 35636, 35761,
107        35884, 35999, 36136, 36261, 36384, 36509, 36636, 36761,
108        36884, 36999, 37136, 37261, 37384, 37509, 37636, 37761,
109        37884, 37999, 38136, 38261, 38384, 38509, 38636, 38761,
110        38884, 38999, 39136, 39261, 39384, 39509, 39636, 39761,
111        39884, 39999, 40136, 40261, 40384, 40509, 40636, 40761,
112        40884, 40999, 41136, 41261, 41384, 41509, 41636, 41761,
113        41884, 41999, 42136, 42261, 42384, 42509, 42636, 42761,
114        42884, 42999, 43136, 43261, 43384, 43509, 43636, 43761,
115        43884, 43999, 44136, 44261, 44384, 44509, 44636, 44761,
116        44884, 44999, 45136, 45261, 45384, 45509, 45636, 45761,
117        45884, 45999, 46136, 46261, 46384, 46509, 46636, 46761,
118        46884, 46999, 47136, 47261, 47384, 47509, 47636, 47761,
119        47884, 47999, 48136, 48261, 48384, 48509, 48636, 48761,
120        48884, 48999, 49136, 49261, 49384, 49509, 49636, 49761,
121        49884, 49999, 50136, 50261, 50384, 50509, 50636, 50761,
122        50884, 50999, 51136, 51261, 51384, 51509, 51636, 51761,
123        51884, 51999, 52136, 52261, 52384, 52509, 52636, 52761,
124        52884, 52999, 53136, 53261, 53384, 53509, 53636, 53761,
125        53884, 53999, 54136, 54261, 54384, 54509, 54636, 54761,
126        54884, 54999, 55136, 55261, 55384, 55509, 55636, 55761,
127        55884, 55999, 56136, 56261, 56384, 56509, 56636, 56761,
128        56884, 56999, 57136, 57261, 57384, 57509, 57636, 57761,
129        57884, 57999, 58136, 58261, 58384, 58509, 58636, 58761,
130        58884, 58999, 59136, 59261, 59384, 59509, 59636, 59761,
131        59884, 59999, 60136, 60261, 60384, 60509, 60636, 60761,
132        60884, 60999, 61136, 61261, 61384, 61509, 61636, 61761,
133        61884, 61999, 62136, 62261, 62384, 62509, 62636, 62761,
134        62884, 62999, 63136, 63261, 63384, 63509, 63636, 63761,
135        63884, 63999, 64136, 64261, 64384, 64509, 64636, 64761,
136        64884, 64999, 65136, 65261, 65384, 65509, 65636, 65761,
137        65884, 65999, 66136, 66261, 66384, 66509, 66636, 66761,
138        66884, 66999, 67136, 67261, 67384, 67509, 67636, 67761,
139        67884, 67999, 68136, 68261, 68384, 68509, 68636, 68761,
140        68884, 68999, 69136, 69261, 69384, 69509, 69636, 69761,
141        69884, 69999, 70136, 70261, 70384, 70509, 70636, 70761,
142        70884, 70999, 71136, 71261, 71384, 71509, 71636, 71761,
143        71884, 71999, 72136, 72261, 72384, 72509, 72636, 72761,
144        72884, 72999, 73136, 73261, 73384, 73509, 73636, 73761,
145        73884, 73999, 74136, 74261, 74384, 74509, 74636, 74761,
146        74884, 74999, 75136, 75261, 75384, 75509, 75636, 75761,
147        75884, 75999, 76136, 76261, 76384, 76509, 76636, 76761,
148        76884, 76999, 77136, 77261, 77384, 77509, 77636, 77761,
149        77884, 77999, 78136, 78261, 78384, 78509, 78636, 78761,
150        78884, 78999, 79136, 79261, 79384, 79509, 79636, 79761,
151        79884, 79999, 80136, 80261, 80384, 80509, 80636, 80761,
152        80884, 80999, 81136, 81261, 81384, 81509, 81636, 81761,
153        81884, 81999, 82136, 82261, 82384, 82509, 82636, 82761,
154        82884, 82999, 83136, 83261, 83384, 83509, 83636, 83761,
155        83884, 83999, 84136, 84261, 84384, 84509, 84636, 84761,
156        84884, 84999, 85136, 85261, 85384, 85509, 85636, 85761,
157        85884, 85999, 86136, 86261, 86384, 86509, 86636, 86761,
158        86884, 86999, 87136, 87261, 87384, 87509, 87636, 87761,
159        87884, 87999, 88136, 88261, 88384, 88509, 88636, 88761,
160        88884, 88999, 89136, 89261, 89384, 89509, 89636, 89761,
161        89884, 89999, 90136, 90261, 90384, 90509, 90636, 90761,
162        90884, 90999, 91136, 91261, 91384, 91509, 91636, 91761,
163        91884, 91999, 92136, 92261, 92384, 92509, 92636, 92761,
164        92884, 92999, 93136, 93261, 93384, 93509, 93636, 93761,
165        93884, 93999, 94136, 94261, 94384, 94509, 94636, 94761,
166        94884, 94999, 95136, 95261, 95384, 95509, 95636, 95761,
167        95884, 95999, 96136, 96261, 96384, 96509, 96636, 96761,
168        96884, 96999, 97136, 97261, 97384, 97509, 97636, 97761,
169        97884, 97999, 98136, 98261, 98384, 98509, 98636, 98761,
170        98884, 98999, 99136, 99261, 99384, 99509, 99636, 99761,
171        99884, 99999, 100136, 100261, 100384, 100509, 100636, 100761,
172        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
173        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
174        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
175        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
176        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
177        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
178        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
179        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
180        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
181        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
182        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
183        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
184        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
185        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
186        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
187        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
188        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
189        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
190        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
191        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
192        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
193        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
194        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
195        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
196        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
197        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
198        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
199        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
200        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
201        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
202        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
203        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
204        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
205        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
206        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
207        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
208        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
209        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
210        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
211        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
212        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
213        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
214        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
215        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
216        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
217        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
218        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
219        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
220        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
221        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
222        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
223        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
224        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
225        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
226        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
227        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
228        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
229        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
230        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
231        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
232        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
233        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
234        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
235        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
236        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
237        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
238        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
239        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
240        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
241        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
242        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
243        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
244        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
245        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
246        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
247        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
248        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
249        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
250        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
251        100884, 100999, 100136, 100261, 100384, 100509, 100636, 100761,
2
```

```
64      900,    961,    1024,    1089,    1156,    1225,    1296,    1369,
65      1444,    1521,
66      1600,    1681,    1764,    1849,    1936,    2025,    2116,    2209,
67      2304,    2401,
68      2500,    2601,    2704,    2809,    2916,    3025,    3136,    3249,
69      3364,    3481,
70      3600,    3721,    3844,    3969,    4096,    4225,    4356,    4489,
71      4624,    4761,
72      4900,    5041,    5184,    5329,    5476,    5625,    5776,    5929,
73      6084,    6241,
74      6400,    6561,    6724,    6889,    7056,    7225,    7396,    7569,
75      7744,    7921,
76      8100,    8281,    8464,    8649,    8836,    9025,    9216,    9409,
77      9604,    9801,
78      10000,   10201,   10404,   10609,   10816,   11025,   11236,
79      11449,   11664,   11881,
80      12100,   12321,   12544,   12769,   12996,   13225,   13456,
81      13689,   13924,   14161,
82      14400,   14641,   14884,   15129,   15376,   15625,   15876,
83      16129,   16384,   16641,
84      16900,   17161,   17424,   17689,   17956,   18225,   18496,
85      18769,   19044,   19321,
86      19600,   19881,   20164,   20449,   20736,   21025,   21316,
87      21609,   21904,   22201,
88      22500,   22801,   23104,   23409,   23716,   24025,   24336,
89      24649,   24964,   25281,
90      25600,   25921,   26244,   26569,   26896,   27225,   27556,
91      27889,   28224,   28561,
92      28900,   29241,   29584,   29929,   30276,   30625,   30976,
93      31329,   31684,   32041,
94      32400,   32761,   33124,   33489,   33856,   34225,   34596,
95      34969,   35344,   35721,
96      36100,   36481,   36864,   37249,   37636,   38025,   38416,
97      38809,   39204,   39601,
```

86 40000, 40401, 40804, 41209, 41616, 42025, 42436,
87 42849, 43264, 43681,
88 44100, 44521, 44944, 45369, 45796, 46225, 46656,
89 47089, 47524, 47961,
90 48400, 48841, 49284, 49729, 50176, 50625, 51076,
91 51529, 51984, 52441,
92 52900, 53361, 53824, 54289, 54756, 55225, 55696,
93 56169, 56644, 57121,
94 57600, 58081, 58564, 59049, 59536, 60025, 60516,
95 61009, 61504, 62001,
96 62500, 63001, 63504, 64009, 64516, 65025};
97 **return** pow2[(n >= 0) ? n : -n];

88 }

89

90 **static inline long** float2intRound(**double** d) {

91 d += 6755399441055744.0;

92 **return** **reinterpret_cast**<**int** &>(d);

93 }

94

95 **/*!**

96 * \brief calcVolume according to ISO 9276-6

97 * \param A

```

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



---


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

```

```

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

```

```

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

```

```
9 // http://stackoverflow.com/questions/16125574/how-to-
10 // serialize-opencv-mat-with-boost-xml-archive
11 #pragma once
12 #include <boost/archive/binary_iarchive.hpp>
13 #include <boost/archive/binary_oarchive.hpp>
14 #include <boost/serialization/access.hpp>
15 #include <opencv/cv.h>
16 #include <opencv2/core.hpp>
17
18 namespace boost {
19 namespace serialization {
20 /*!
21 * \brief serialize Serialize the openCV mat to disk
22 */
23 template <class Archive>
24 inline void serialize(Archive &ar, cv::Mat &m, const
25 unsigned int version __attribute__((unused))) {
26 int cols = m.cols;
27 int rows = m.rows;
28 int elemSize = m.elemSize();
29 int elemType = m.type();
30
31 ar &cols;
32 ar &rows;
33 ar &elemSize;
34 ar &elemType; // element type.
35
36 if (m.type() != elemType || m.rows != rows || m.cols !=
37 cols) {
38 m = cv::Mat(rows, cols, elemType, cv::Scalar(0));
39 }
40
41 size_t dataSize = cols * rows * elemSize;
42
43 for (size_t dc = 0; dc < dataSize; dc++) {
44 ar &m.data[dc];
45 }
46 }
47
48 /* Copyright (C) Jelle Spijker - All Rights Reserved
49 * Unauthorized copying of this file, via any medium is
50 * strictly prohibited
51 * and only allowed with the written consent of the author (Jelle Spijker)
52 * This software is proprietary and confidential
53 * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
54 */
55
56 // Source:
57 // http://stackoverflow.com/questions/16125574/how-to-
58 // serialize-opencv-mat-with-boost-xml-archive
59 #pragma once
60
```

```

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

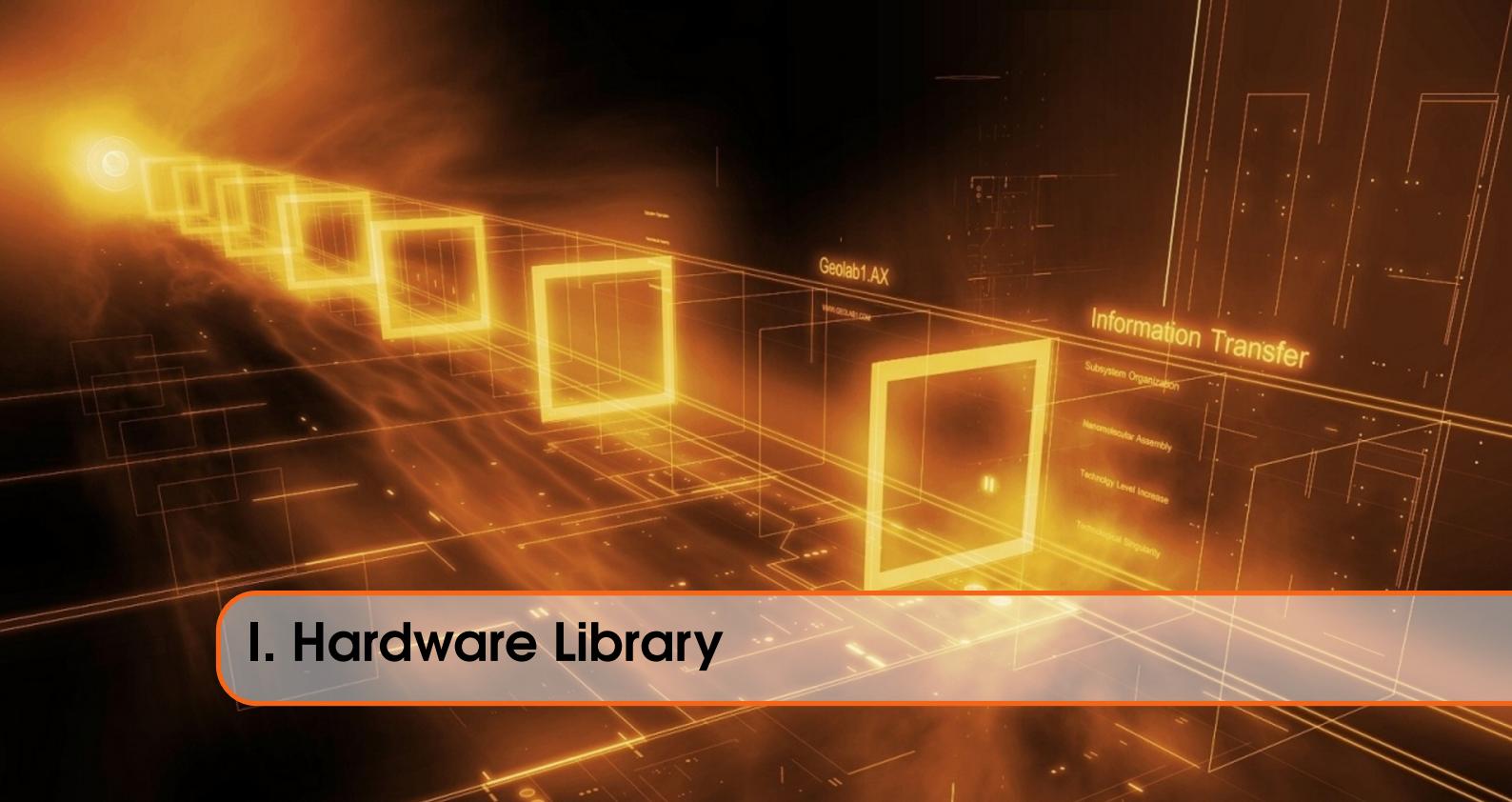
/*
Copyright (C) Jelle Spijker - All Rights Reserved
* Unauthorized copying of this file, via any medium is
strictly prohibited
* and only allowed with the written consent of the author (
Jelle Spijker)
* This software is proprietary and confidential
* Written by Jelle Spijker <spijkerv.jelle@gmail.com>, 2015
*/
7
8 #define EXCEPTION_MATH "Math Exception!"
9 #define EXCEPTION_MATH_NR 0
10 #define EXCEPTION_NO_CONTOUR_FOUND
11     "No continuous contour found, or less than 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 {

```



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

```
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 - 1);
97 }
98 };
99 }
```



I. Hardware Library

Microscope Class

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

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

```

86     uint32_t ID = V4L2_CID_BASE;
87     bool operator==(Control_t &rhs) {
88         if (this->name.compare(rhs.name) == 0) {
89             return true;
90         } else {
91             return false;
92         }
93     }
94     bool operator!=(Control_t &rhs) {
95         if (this->name.compare(rhs.name) != 0) {
96             return true;
97         } else {
98             return false;
99         }
100    }
101 };
102
103 typedef std::vector<Control_t> Controls_t;
104
105 typedef struct _CustomData {
106     GMainLoop *main_loop;
107     GstElement *pipeline;
108     GstElement *source;
109     GstElement *capsfilter;
110     GstElement *tisvideobuffer;
111     GstElement *tiscolorize;
112     GstElement *bayer;
113     GstElement *queue;
114     GstElement *colorspace;
115     GstElement *convert;
116     GstElement *sink;
117     GstBus *bus;
118     GstCaps *caps;
119     Hardware::Microscope *currentMicroscope;
120 } CustomData;
121
122 struct Cam_t {
123     std::string Name;
124     std::string devString;
125     uint32_t ID;
126     std::vector<Resolution_t> Resolutions;
127     uint32_t delaytrigger = 1;
128     Resolution_t *SelectedResolution = nullptr;
129     Controls_t Controls;
130     CustomData Pipe;
131     int fd;
132     bool operator==(Cam_t const &rhs) {
133         if (this->ID == rhs.ID || this->Name == rhs.Name) {
134             return true;
135         } else {
136             return false;
137         }
138     }
139     bool operator!=(Cam_t const &rhs) {
140         if (this->ID != rhs.ID && this->Name != rhs.Name) {
141             return true;

```

```
142         } else {
143             return false;
144         }
145     }
146 };
147
148 std::vector<Cam_t> AvailableCams;
149 Cam_t *SelectedCam = nullptr;
150 Arch RunEnv;
151
152 Microscope();
153 Microscope(const Microscope &rhs);
154
155 ~Microscope();
156
157 Microscope operator=(Microscope const &rhs);
158
159     bool IsOpened();
160     bool openCam(Cam_t *cam);
161     bool openCam(int &cam);
162     bool openCam(std::string &cam);
163
164     bool closeCam(Cam_t *cam);
165
166     void GetFrame(cv::Mat &dst);
167     void GetGstreamFrame(cv::Mat &dst);
168     void GetHDRFrame(cv::Mat &dst, uint32_t noframes = 3);
169
170     Control_t *GetControl(const std::string name);
171     void SetControl(Control_t *control);
172
173     Cam_t *FindCam(std::string cam);
174     Cam_t *FindCam(int cam);
175     cv::Mat lastFrame;
176
177     void SendImageRetrieved();
178
179 public slots:
180     void on_imageretrieved();
181
182 signals:
183     void imageretrieved();
184
185 private:
186     static void new_buffer(GstElement *sink, CustomData *data)
187         ;
188     void getResolutions(Cam_t &currentCam, int FormatType);
189     bool openedUpTheCam = false;
190     cv::VideoCapture *cap = nullptr;
191
192     std::vector<cv::Mat> HDRframes;
193
194     std::vector<Cam_t> GetAvailableCams();
195     Arch GetCurrentArchitecture();
196     int fd;
197 }
```

```
197 }



---


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

40 Microscope::Arch Microscope::GetCurrentArchitecture() {
41     struct utsname unameData;
42     Arch retVal;
43     uname(&unameData);
44     std::string archString = static_cast<std::string>(
45         unameData.machine);
46     if (archString.find("armv7l") != string::npos) {
47         retVal = Arch::ARM;
48     } else {
49         retVal = Arch::X64;
50     }
51     return retVal;
52 }
```

```

47 std::vector<Microscope::Cam_t> Microscope::GetAvailableCams
48     () {
49         const string path_ss = "/sys/class/video4linux";
50         const string path_ss_dev = "/dev/video";
51         std::vector<Cam_t> retVal;
52         struct v4l2_queryctrl queryctrl;
53         struct v4l2_control controlctrl;
54
55         // Check if there're videodevices installed
56         // Iterate through the cams
57         for (boost::filesystem::directory_iterator itr(path_ss);
58             itr != boost::filesystem::directory_iterator(); ++itr
59             ) {
60             string videoLn = itr->path().string();
61             videoLn.append("/name");
62             if (boost::filesystem::exists(videoLn)) {
63                 Cam_t currentCam;
64                 std::ifstream camName;
65                 camName.open(videoLn);
66                 std::getline(camName, currentCam.Name);
67                 camName.close();
68                 currentCam.ID =
69                     std::stoi(itr->path().string().substr(28, std::
70                         string::npos).c_str());
71
72                 // Open Cam
73                 currentCam.devString = path_ss_dev + std::to_string(
74                     currentCam.ID);
75                 if ((currentCam.fd = open(currentCam.devString.c_str()
76                     , O_RDWR)) == -1) {
77                     throw Exception::MicroscopeException(
78                         EXCEPTION_NOAMS,
79                         EXCEPTION_NOAMS_NR
80                     );
81
82                 // Get controls
83                 memset(&queryctrl, 0, sizeof(queryctrl));
84                 memset(&controlctrl, 0, sizeof(controlctrl));
85                 for (queryctrl.id = V4L2_CID_BASE; queryctrl.id <
86                     V4L2_CID_LASTP1;
87                     queryctrl.id++) {
88
89                     if (ioctl(currentCam.fd, VIDIOC_QUERYCTRL, &
90                         queryctrl) == 0) {
91                         if (!(queryctrl.flags & V4L2_CTRL_FLAG_DISABLED))
92                         {
93                             Control_t currentControl;
94                             currentControl.ID = queryctrl.id;
95                             currentControl.name = (char *)queryctrl.name;
96                             currentControl.minimum = queryctrl.minimum;
97                             currentControl.maximum = queryctrl.maximum;
98                             currentControl.default_value = queryctrl.
99                                 default_value;
100                             currentControl.step = queryctrl.step;
101                             controlctrl.id = queryctrl.id;
102                         }
103                     }
104                 }
105             }
106         }
107     }
108 }
```

```

92         if (ioctl(currentCam.fd, VIDIOC_G_CTRL, &
93             controlctrl) == 0) {
94             currentControl.current_value = controlctrl.
95                 value;
96         }
97     } else {
98         if (errno == EINVAL)
99             continue;
100        throw Exception::MicroscopeException(
101            EXCEPTION_QUERY,
102            EXCEPTION_QUERY_NR
103        );
104    }
105    getResolutions(currentCam, V4L2_PIX_FMT_YUYV);
106    getResolutions(currentCam, V4L2_PIX_FMT_MJPEG);
107    getResolutions(currentCam, V4L2_PIX_FMT_GREY);
108    close(currentCam.fd);
109    retVal.push_back(currentCam);
110 }
111 }
112
113 for (uint32_t i = 0; i < retVal.size(); i++) {
114     if (retVal[i].Resolutions.size() == 0) {
115         retVal.erase(retVal.begin() + i);
116         i--;
117     }
118 }
119
120 return retVal;
121 }
122
123 void Microscope::getResolutions(Cam_t &currentCam, int
124     FormatType) {
125     // Get image formats
126     struct v4l2_format format;
127     memset(&format, 0, sizeof(format));
128     uint32_t width[10] = {640, 800, 1280, 1280, 1920,
129                         1600, 2048, 2560, 3840, 3872};
130     uint32_t height[10] = {480, 600, 720, 960, 1080,
131                         1200, 1536, 1440, 2160, 2764};
132
133     uint32_t ResolutionID = 0;
134
135     for (uint32_t i = 0; i < 10; i++) {
136         format.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
137         format.fmt.pix.pixelformat = FormatType;
138         format.fmt.pix.width = width[i];
139         format.fmt.pix.height = height[i];
140         int ret = ioctl(currentCam.fd, VIDIOC_S_FMT, &format);
141         if (ret != -1 && format.fmt.pix.height == height[i] &&
142             format.fmt.pix.width == width[i]) {

```

```

143     Resolution_t res;
144     res.Width = format.fmt.pix.width;
145     res.Height = height[i];
146     res.ID = ResolutionID++;
147     switch (FormatType) {
148     case V4L2_PIX_FMT_YUYV:
149         res.format = PixelFormat::YUYV;
150         break;
151     case V4L2_PIX_FMT_MJPEG:
152         res.format = PixelFormat::MJPEG;
153         break;
154     case V4L2_PIX_FMT_GREY:
155         res.format = PixelFormat::GREY;
156         break;
157     default:
158         break;
159     }
160     currentCam.Resolutions.push_back(res);
161 }
162 }
163 }
164
165 bool Microscope::IsOpened() { return openedUptheCam; }
166
167 bool Microscope::openCam(Cam_t *cam) {
168     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
169         if (AvailableCams[i] == *cam) {
170             closeCam(SelectedCam);
171             SelectedCam = cam;
172             for (Controls_t::iterator it = SelectedCam->Controls.
173                 begin();
174                 it != SelectedCam->Controls.end(); ++it) {
175                 SetControl(&*it);
176             }
177             SelectedCam->Pipe.currentMicroscope = this;
178             gst_init(NULL, NULL);
179
180             SelectedCam->Pipe.pipeline = gst_pipeline_new("SoilCam
181             ");
182             if (!SelectedCam->Pipe.pipeline) {
183                 throw Exception::MicroscopeException(
184                     EXCEPTION_GSTREAM_INIT_EXCEPTION,
185                     EXCEPTION_GSTREAM_INIT_EXCEPTION_NR);
186             }
187             SelectedCam->Pipe.source = gst_element_factory_make("v4l2src", "source");
188             SelectedCam->Pipe.capsfilter =
189                 gst_element_factory_make("capsfilter", "filter");
190             SelectedCam->Pipe.colorspace =
191                 gst_element_factory_make("ffmpegcolorspace", "colorspace");
192             SelectedCam->Pipe.convert =
193                 gst_element_factory_make("capsfilter", "convert");

```

```

194     SelectedCam->Pipe.sink = gst_element_factory_make(
195         "appsink", "output");
196     if (!SelectedCam->Pipe.source || !SelectedCam->Pipe.
197         capsfilter ||
198         !SelectedCam->Pipe.colorspace || !SelectedCam->
199         Pipe.sink ||
200         !SelectedCam->Pipe.convert) {
201         throw Exception::MicroscopeException(
202             EXCEPTION_GSTREAM_ELEM_EXCEPTION,
203             EXCEPTION_GSTREAM_ELEM_EXCEPTION_NR);
204     }
205
206     if (SelectedCam->Name.compare("DFK 24UJ003") == 0) {
207         SelectedCam->Pipe.tisvideobuffer =
208             gst_element_factory_make(
209                 "tisvideobufferfilter", "tisvideobufferfilter");
210         SelectedCam->Pipe.tiscolorize =
211             gst_element_factory_make("tiscolorize", "tiscolorize");
212         SelectedCam->Pipe.queue = gst_element_factory_make(
213             "queue", "queue");
214         SelectedCam->Pipe.bayer =
215             gst_element_factory_make("bayer2rgb", "bayer");
216         if (!SelectedCam->Pipe.tisvideobuffer ||
217             !SelectedCam->Pipe.tiscolorize || !SelectedCam->
218             Pipe.queue ||
219             !SelectedCam->Pipe.bayer) {
220             throw Exception::MicroscopeException(
221                 EXCEPTION_GSTREAM_ELEM_EXCEPTION,
222                 EXCEPTION_GSTREAM_ELEM_EXCEPTION_NR);
223         }
224         g_object_set(SelectedCam->Pipe.source, "device",
225             SelectedCam->devString.c_str());
226
227         switch (SelectedCam->SelectedResolution->format) {
228             case PixelFormat::MJPEG:
229                 SelectedCam->Pipe.caps = gst_caps_new_simple(
230                     "video/x-raw-rgb", "width", G_TYPE_INT,
231                     SelectedCam->SelectedResolution->Width, "height"
232                     , G_TYPE_INT,
233                     SelectedCam->SelectedResolution->Height, NULL);
234             case PixelFormat::GREY:
235                 SelectedCam->Pipe.caps = gst_caps_new_simple(
236                     "video/x-raw-gray", "width", G_TYPE_INT,
237                     SelectedCam->SelectedResolution->Width, "height"
238                     , G_TYPE_INT,
239                     SelectedCam->SelectedResolution->Height, NULL);
240             break;
241             case PixelFormat::YUYV:
242                 SelectedCam->Pipe.caps = gst_caps_new_simple(
243                     "video/x-raw-gray", "format", G_TYPE_STRING, "((
244                         fourcc)UYVY",

```

```

239         "width", G_TYPE_INT, SelectedCam->
240             SelectedResolution->Width,
241         "height", G_TYPE_INT, SelectedCam->
242             SelectedResolution->Height,
243             NULL);
244     default:
245         break;
246 }
247 g_object_set(SelectedCam->Pipe.capsfilter, "caps",
248             SelectedCam->Pipe.caps,
249             NULL);
250 gst_caps_unref(SelectedCam->Pipe.caps);
251 SelectedCam->Pipe.caps = gst_caps_new_simple(
252     "video/x-raw-rgb", "width", G_TYPE_INT,
253     SelectedCam->SelectedResolution->Width, "height",
254     G_TYPE_INT,
255     SelectedCam->SelectedResolution->Height, NULL);
256 g_object_set(SelectedCam->Pipe.convert, "caps",
257             SelectedCam->Pipe.caps,
258             NULL);
259
260     SelectedCam->Pipe.bus = gst_element_get_bus(
261         SelectedCam->Pipe.pipeline);
262     g_object_set(SelectedCam->Pipe.sink, "emit-signals",
263             TRUE, NULL);
264     g_signal_connect(SelectedCam->Pipe.sink, "new-buffer",
265                     G_CALLBACK(new_buffer), &SelectedCam
266                     ->Pipe);
267
268     if (SelectedCam->Name.compare("DFK 24UJ003") == 0) {
269         gst_bin_add_many(GST_BIN(SelectedCam->Pipe.pipeline)
270             ,
271             SelectedCam->Pipe.source,
272             SelectedCam->Pipe.capsfilter,
273             SelectedCam->Pipe.tisvideobuffer,
274             SelectedCam->Pipe.tiscolorize,
275             SelectedCam->Pipe.queue,
276             SelectedCam->Pipe.bayer,
277             SelectedCam->Pipe.sink, NULL);
278         gst_element_link_many(
279             SelectedCam->Pipe.source, SelectedCam->Pipe.
280             capsfilter,
281             SelectedCam->Pipe.tisvideobuffer, SelectedCam->
282             Pipe.tiscolorize,
283             SelectedCam->Pipe.queue, SelectedCam->Pipe.bayer
284             ,
285             SelectedCam->Pipe.sink, NULL);
286     } else {
287         gst_bin_add_many(
288             GST_BIN(SelectedCam->Pipe.pipeline), SelectedCam
289             ->Pipe.source,
290             SelectedCam->Pipe.capsfilter, SelectedCam->Pipe.
291             colorspace,
292             SelectedCam->Pipe.convert, SelectedCam->Pipe.
293             sink, NULL);
294         gst_element_link_many(

```

```

277         SelectedCam->Pipe.source, SelectedCam->Pipe.
278             capsfilter,
279             SelectedCam->Pipe.colorspace, SelectedCam->Pipe.
280                 convert,
281                 SelectedCam->Pipe.sink, NULL);
282     }
283     openedUptheCam = true;
284     return true;
285 }
286 openedUptheCam = false;
287 return false;
288 }
289 bool Microscope::openCam(std::string &cam) { return openCam(
290     FindCam(cam)); }
291 bool Microscope::openCam(int &cam) { return openCam(FindCam(
292     cam)); }
293 Microscope::Cam_t *Microscope::FindCam(int cam) {
294     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
295         if (cam == AvailableCams[i].ID) {
296             return &AvailableCams[i];
297         }
298     }
299     return nullptr;
300 }
301
302 Microscope::Cam_t *Microscope::FindCam(string cam) {
303     for (uint32_t i = 0; i < AvailableCams.size(); i++) {
304         if (cam.compare(AvailableCams[i].Name) == 0) {
305             return &AvailableCams[i];
306         }
307     }
308     return nullptr;
309 }
310
311 bool Microscope::closeCam(Cam_t *cam) {
312     if (openedUptheCam) {
313         gst_element_set_state(cam->Pipe.pipeline, GST_STATE_NULL
314             );
315         gst_object_unref(GST_OBJECT(cam->Pipe.pipeline));
316         openedUptheCam = false;
317     }
318 }
319 void Microscope::GetFrame(cv::Mat &dst) {
320     if (!IsOpened()) {
321         openCam(SelectedCam);
322     }
323     QEventLoop loop;
324     loop.connect(this, SIGNAL(imageretrieved()), SLOT(quit()))
325         ;
326     gst_element_set_state(SelectedCam->Pipe.pipeline,
327             GST_STATE_PLAYING);

```

```

326     loop.exec();
327     dst = lastFrame;
328     closeCam(SelectedCam);
329 }
330
331 void Microscope::on_imageretrieved() { return; }
332
333 void Microscope::GetHDRFrame(cv::Mat &dst, uint32_t noframes
334 ) {
335     // create the brightness steps
336     Control_t *brightness = GetControl("Brightness");
337     Control_t *contrast = GetControl("Contrast");
338
339     uint32_t brightnessStep =
340         (brightness->maximum - brightness->minimum) / noframes
341         ;
342     int8_t currentBrightness = brightness->current_value;
343     int8_t currentContrast = contrast->current_value;
344     contrast->current_value = contrast->maximum;
345
346     cv::Mat currentImg;
347     // take the shots at different brightness levels
348     for (uint32_t i = 1; i <= noframes; i++) {
349         brightness->current_value = brightness->minimum + (i *
350             brightnessStep);
351         GetFrame(currentImg);
352         HDRframes.push_back(currentImg);
353     }
354
355     // Set the brightness and back to the previous used level
356     brightness->current_value = currentBrightness;
357     contrast->current_value = currentContrast;
358
359     // Perform the exposure fusion
360     cv::Mat fusion;
361     cv::Ptr<cv::MergeMertens> merge_mertens = cv::
362         createMergeMertens();
363     merge_mertens->process(HDRframes, fusion);
364     fusion *= 255;
365     fusion.convertTo(dst, CV_8UC1);
366 }
367
368 Microscope::Control_t *Microscope::GetControl(const string
369     name) {
370     for (Controls_t::iterator it = SelectedCam->Controls.begin
371         ());
372         it != SelectedCam->Controls.end(); ++it) {
373         if (name.compare(it->name) == 0) {
374             return &*it;
375         }
376     }
377     return nullptr;
378 }
379
380 void Microscope::SetControl(Control_t *control) {

```

```

375     if ((SelectedCam->fd = open(SelectedCam->devString.c_str()
376         , O_RDWR)) == -1) {
377         throw Exception::MicroscopeException(EXCEPTION_NOCAMS ,
378                                         EXCEPTION_NOCAMS_NR);
379     }
380
381     struct v4l2_queryctrl queryctrl;
382     struct v4l2_control controlctrl;
383
384     memset(&queryctrl, 0, sizeof(queryctrl));
385     queryctrl.id = control->ID;
386     if (ioctl(SelectedCam->fd, VIDIOC_QUERYCTRL, &queryctrl)
387         == -1) {
388         if (errno != EINVAL) {
389             close(SelectedCam->fd);
390             throw Exception::MicroscopeException(EXCEPTION_QUERY ,
391                                         EXCEPTION_QUERY_NR);
392         } else {
393             close(SelectedCam->fd);
394             throw Exception::MicroscopeException(
395                 EXCEPTION_CTRL_NOT_FOUND ,
396                                         EXCEPTION_CTRL_NOT_FOUND_NR
397                                         );
398         }
399     } else if (queryctrl.flags & V4L2_CTRL_FLAG_DISABLED) {
400         close(SelectedCam->fd);
401         throw Exception::MicroscopeException(
402             EXCEPTION_CTRL_NOT_FOUND ,
403                                         EXCEPTION_CTRL_NOT_FOUND_NR
404                                         );
405     } else {
406         memset(&controlctrl, 0, sizeof(controlctrl));
407         controlctrl.id = control->ID;
408         controlctrl.value = control->current_value;
409
410         if (ioctl(SelectedCam->fd, VIDIOC_S_CTRL, &controlctrl)
411             == -1) {
412             // Fails on auto white balance
413             // throw Exception::MicroscopeException(
414             //     EXCEPTION_CTRL_VALUE ,
415             //     EXCEPTION_CTRL_VALUE_NR);
416         }
417     }
418     close(SelectedCam->fd);
419 }
420
421 void Microscope::SendImageRetrieved() { emit imageretrieved
422     (); }
423
424 void Microscope::new_buffer(GstElement *sink, CustomData *
425     data) {
426     GstBuffer *buffer;
427     g_signal_emit_by_name(sink, "pull-buffer", &buffer);
428     if (buffer) {
429         cv::Mat bufferMat(

```

```
418     data->currentMicroscope->SelectedCam->
419         SelectedResolution->Height,
420     data->currentMicroscope->SelectedCam->
421         SelectedResolution->Width,
422     CV_8UC4, (uchar *)buffer->data);
423     std::vector<cv::Mat> chans;
424     cv::split(bufferMat, chans);
425     chans.erase(chans.begin() + 4);
426     cv::merge(chans, data->currentMicroscope->lastFrame);
427     cv::namedWindow("test");
428     cv::imshow("test", data->currentMicroscope->lastFrame);
429     cv::waitKey(0);
430     data->currentMicroscope->SendImageRetrieved();
431     //      gst_element_set_state(data->currentMicroscope->
432     //      SelectedCam->Pipe.pipeline,
433     //      GST_STATE_PAUSED);
434 }
```

Beaglebone Black Class

```

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

```

```

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



---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
10 #include "BBB.h"
11
12 namespace Hardware {
13     /*! Constructor*/
14     BBB::BBB() {
15         threadRunning = false;
16         callbackFunction = NULL;
17         debounceTime = 0;
18         thread = (pthread_t)NULL;
19     }
20
21     /*! De-constructor*/
22 }
```

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

```
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 }
```

GPIO Class

```

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

```

```

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

```

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

```

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

```

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

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

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

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

PWM Class

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

```

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



---


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

```

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

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

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

ADC Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 /*! \class ADC
11  Interaction with the beaglebone analogue pins
12 */
13
14 #pragma once
15
16 #include "BBB.h"
17 #include "ADCReadException.h"
18
19 #define ADC0_PATH
20
21
22 #define ADC1_PATH
23
24
25 #define ADC2_PATH
26
27
28 #define ADC3_PATH
29
30
31 #define ADC4_PATH
32
33
34 #define ADC5_PATH

```

```

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

```

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



---


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

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

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

EC12P Class

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

```

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



---


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

```

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

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

eQep Class

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

```

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



---


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

```

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

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

```
109 // Check that we opened the file correctly
110 if (fp == NULL) {
111     // Error, break out
112     std::cerr << "[eQEP " << this->path << "] Unable to open
113         mode for write"
114         << std::endl;
115     return;
116 }
117 // 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::
131             FailedToCreateGPIOPollingThreadException();
132     }
133     return 0;
134 }
135
136 // Get the position of the hardware
137 int32_t eQEP::get_position(bool _poll) {
138     // Position temporary variable
139     int32_t position;
140     char dummy;
141     struct pollfd ufd;
142
143     // Do we want to poll?
144     if (_poll) {
145         // Open a connection to the attribute file.
146         if ((ufd.fd = open((this->path + "/position").c_str(),
147                           O_RDWR)) < 0) {
148             // Error, break out
149             std::cerr << "[eQEP " << this->path
150                 << "] unable to open position for polling"
151                 << std::endl;
152         }
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
161         occurred whilst polling"
162         << std::endl;
163         close(ufd.fd);
164     }
165 }
166
167 // Read the position
168 FILE *fp = fopen((this->path + "/position").c_str(), "r");
169
170 // Check that we opened the file correctly
171 if (fp == NULL) {
172     // Error, break out
173     std::cerr << "[eQEP " << this->path << "] Unable to open
174         position for read"
175         << std::endl;
176         close(ufd.fd);
177     return 0;
178 }
179
180 // Write the desired value to the file
181 fscanf(fp, "%d", &position);
182
183 // Commit changes
184 fclose(fp);
185
186 // If we were polling, close the polling file
187 if (_poll) {
188     close(ufd.fd);
189 }
190
191 // Return the position
192 return position;
193
194 // Get the period of the eQEP hardware
195 uint64_t eQEP::get_period() {
196     // Open the file representing the position
197     FILE *fp = fopen((this->path + "/period").c_str(), "r");
198
199     // Check that we opened the file correctly
200     if (fp == NULL) {
201         // Error, break out
202         std::cerr << "[eQEP " << this->path << "] Unable to open
203             period for read"
204             << std::endl;
205     }
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     // Return the period
214     return period;
215 }
216 }
217
218 // Get the mode of the eQEP hardware
219 eQEP::eQEP_Mode eQEP::get_mode() {
220     // Open the file representing the position
221     FILE *fp = fopen((this->path + "/mode").c_str(), "r");
222
223     // Check that we opened the file correctly
224     if (fp == NULL) {
225         // Error, break out
226         std::cerr << "[eQEP " << this->path << "] Unable to open
227             mode for read"
228             << std::endl;
229         return eQEP::eQEP_Mode_Error;
230     }
231
232     // Write the desired value to the file
233     eQEP::eQEP_Mode mode;
234     fscanf(fp, "%u", (unsigned int *)&mode);
235
236     // Commit changes
237     fclose(fp);
238
239     // Return the mode
240     return mode;
241 }
242
243 void *threadedPolleqep(void *value) {
244     eQEP *eqep = static_cast<eQEP *>(value);
245     while (eqep->threadRunning) {
246         eqep->callbackFunction(eqep->get_position(true));
247         usleep(eqep->debounceTime * 1000);
248     }
249 }
250 }
```

SoilCape Class

```

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

```

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

USB Class

```

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

```

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


General project files

```
1 #-----
2 #
3 # Project created by QtCreator 2015-06-06T10:49:23
4 #
5 #-----
6 QT      += core gui concurrent
7 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
8
9 TARGET = SoilHardware
10 TEMPLATE = lib
11 VERSION = 0.9.2
12
13 DEFINES += SOILHARDWARE_LIBRARY
14 QMAKE_CXXFLAGS += -std=c++11 -pthread
15 unix:!macx: QMAKE_RPATHDIR += $$PWD/../../../../build/install/
16
17 SOURCES += \
18     USB.cpp \
19     SoilCape.cpp \
20     PWM.cpp \
21     Microscope.cpp \
22     GPIO.cpp \
23     eqep.cpp \
24     EC12P.cpp \
25     BBB.cpp \
26     ADC.cpp
27
28 HEADERS += \
29     ValueOutOfBoundsException.h \
30     USB.h \
31     SoilCape.h \
32     PWM.h \
33     MicroscopeNotFoundException.h \
34     Microscope.h \
35     Hardware.h \
36     GPIOReadException.h \
37     GPIO.h \
38     FailedToCreateThreadException.h \
39     FailedToCreateGPIOPollingThreadException.h \
40     eqep.h \
41     EC12P.h \
42     CouldNotGrabImageException.h \
43     BBB.h \
44     ADCReadException.h \
45     ADC.h
46
47 #opencv
48 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui - \
49         opencv_photo -lopencv_imgcodecs -lopencv_videoio
50 INCLUDEPATH += /usr/local/include/opencv
51 INCLUDEPATH += /usr/local/include
52 #boost
53 DEFINES += BOOST_ALL_DYN_LINK
```

```

54 INCLUDEPATH += /usr/include/boost
55 LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_filesystem -
      lboost_system
56
57 unix {
58     target.path = $PWD/../../build/install
59     INSTALLS += target
60 }
61
62 #Gstreamer
63 INCLUDEPATH += /usr/include/gstreamer-0.10
64 INCLUDEPATH += /usr/include/glib-2.0/
65 INCLUDEPATH += /usr/lib/x86_64-linux-gnu/glib-2.0/include/
66 INCLUDEPATH += /usr/include/libxml2/
67 LIBS += 'pkg-config --cflags --libs gstreamer-0.10'

```

```

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

```

```

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

```

```
13 using namespace std;
14
15 namespace Hardware {
16 namespace Exception {
17 class ValueOutOfBoundsException : public std::exception {
18 public:
19     ValueOutOfBoundsException(string m = "Value out of bounds!
20     ") : msg(m){};
20     ~ValueOutOfBoundsException() _GLIBCXX_USE_NOEXCEPT{};
21     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
22         msg.c_str(); };
23
24     private:
25         string msg;
26     };
27 }
28
29
30 /* Copyright (C) Jelle Spijker - All Rights Reserved
31  * Unauthorized copying of this file, via any medium is
32  * strictly prohibited
33  * and only allowed with the written consent of the author (
34  * Jelle Spijker)
35  * This software is proprietary and confidential
36  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
37 */
38
39 #pragma once
40 #include <exception>
41 #include <string>
42
43 using namespace std;
44
45 namespace Hardware {
46 namespace Exception {
47 class ADCReadException : public std::exception {
48 public:
49     ADCReadException(string m = "Can't read ADC data!") : msg(
50         m){};
51     ~ADCReadException() _GLIBCXX_USE_NOEXCEPT{};
52     const char *what() const _GLIBCXX_USE_NOEXCEPT { return
53         msg.c_str(); };
54
55     private:
56         string msg;
57     };
58 }
59
60
61 /* Copyright (C) Jelle Spijker - All Rights Reserved
62  * Unauthorized copying of this file, via any medium is
63  * strictly prohibited
64  * and only allowed with the written consent of the author (
65  * Jelle Spijker)
66  * This software is proprietary and confidential
67  * 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
18   : exception {
19 public:
20   FailedToCreateGPIOPollingThreadException(
21     string m = "Failed to create GPIO polling thread!")
22   : msg(m){};
23   ~FailedToCreateGPIOPollingThreadException()
24     _GLIBCXX_USE_NOEXCEPT{};
25   const char *what() const _GLIBCXX_USE_NOEXCEPT { return
26     msg.c_str(); };
27
28 }
29 }

```

```

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

```



```
42
43     private:
44         string msg;
45         int nr;
46     };
47 }
48 }



---


1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (
5  * Jelle Spijker)
6  * This software is proprietary and confidential
7  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8  */
9
10 #pragma once
11 #include <exception>
12 #include <string>
13
14 using namespace std;
15
16 namespace Hardware {
17     namespace Exception {
18         class CouldNotGrabImageException : public std::exception {
19             public:
20                 CouldNotGrabImageException(string m = "Unable to grab the
21                     next image!")
22                 : msg(m){};
23             ~CouldNotGrabImageException() _GLIBCXX_USE_NOEXCEPT{};
24             const char *what() const _GLIBCXX_USE_NOEXCEPT { return
25                 msg.c_str(); };
26         };
27     }
28 }



---


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




J. Vision Library

Image processing Class

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

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

```

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

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

```

```

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

```

```

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

```

```

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

```

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

```

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

```

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

Conversion Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
8   */
9
10 #pragma once
11 #include "ImageProcessing.h"
12 #include "ConversionNotSupportedException.h"
13
14 namespace Vision {
15 class Conversion : public ImageProcessing {
16 public:
17   /*! Enumerator which indicates the colorspace used*/
18   enum ColorSpace {
19     CIE_lab,    /*!< CIE La*b* colorspace */
20     CIE_XYZ,   /*!< CIE XYZ colorspace */
21     RI,        /*!< Redness Index colorspace */
22     RGB,       /*!< RGB colorspace */
23     Intensity, /*!< Grayscale colorspace */
24     None       /*!< none */
25   };
26   ColorSpace OriginalColorSpace; /*!< The original
27   colorspace*/
28   ColorSpace ProcessedColorSpace; /*!< The destination
29   colorspace*/
30
31 Conversion();
32 Conversion(const Mat &src);
33 Conversion(const Conversion &rhs);
34
35 ~Conversion();
36
37 Conversion &operator=(Conversion rhs);
38
39 void Convert(ColorSpace convertFrom, ColorSpace convertTo,
40               bool chain = false);
41 void Convert(const Mat &src, Mat &dst, ColorSpace
42             convertFrom,
43             ColorSpace convertTo, bool chain = false);
44
45 private:
46   /*!< Conversion matrix used in the conversion between RGB
47   and CIE XYZ*/
48   float XYZmat[3][3] = {{0.412453, 0.357580, 0.180423},
49                         {0.212671, 0.715160, 0.072169},
50                         {0.019334, 0.119194, 0.950227}};
51
52   float whitePoint[3] = {
53     0.9504, 1.0000, 1.0889}; /*!< Natural whitepoint in
54     XYZ colorspace D65

```

```

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



---


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

```

```

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

```

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

```

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

```

```

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

```

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

Enhance Class

```

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

```

```

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

```

```

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

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

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

```

```

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

```

```

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

```

```

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

```

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

```
274 /*! Stretches the image using a histogram
275 \param chain use the results from the previous operation
276     default value = false;
277 */
278 void Enhance::HistogramEqualization(bool chain) {
279     // Exception handling
280     EMPTY_CHECK(OriginalImg);
281     CV_Assert(OriginalImg.depth() != sizeof(uchar));
282
283     // Make the pointers to the Data
284     uchar *O;
285     CHAIN_PROCESS(chain, 0, uchar);
286     uchar *P = ProcessedImg.data;
287
288     // Calculate the statics of the whole image
289     ucharStat_t imgStats(0, OriginalImg.rows, OriginalImg.cols
290                         );
291     float sFact;
292     if (imgStats.min != imgStats.max) {
293         sFact = 255.0f / (imgStats.max - imgStats.min);
294     } else {
295         sFact = 1.0f;
296     }
297
298     uint32_t i = 256;
299     uchar LUT_changeValue[256];
300     while (i-- > 0) {
301         LUT_changeValue[i] = (uchar)((float)(i)*sFact) + 0.5f;
302     }
303
304     O = OriginalImg.data;
305
306     i = OriginalImg.cols * OriginalImg.rows + 1;
307     while (i-- > 0) {
308         *P++ = LUT_changeValue[*O++ - imgStats.min];
309     }
310 }
```

Morphological filter Class

```

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

```

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

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

```

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

```

```

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

```

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

Segment Class

```

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

```

```

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

```

```

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

```

```

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



---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
3  * strictly prohibited
4  * and only allowed with the written consent of the author (Jelle Spijker)
5  * This software is proprietary and confidential
6  * Written by Jelle Spijker <spijker.jelle@gmail.com>, 2015
7  */
8 /*! \class Segment
9 \brief Segmentation algorithms
10 With this class, various segmentation routines can be
11 applied to a greyscale or
12 black and white source image.
13 */
14
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;

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

```

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

```

```

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

```

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

```

```

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

```

```

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

```

```

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

```

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

```

```

697     uint32_t nData = nCols * nRows;
698
699     uint16_t currentlbl = 0;
700     vector<uint16_t> zeroVector;
701     zeroVector.push_back(currentlbl);
702     adj.push_back(zeroVector);
703
704     // Determine which borderpixels should be handled
705     // differently
706     uchar *nRow = new uchar[nData]{};
707     for (uint32_t i = nCols; i < nData; i += nCols) {
708         nRow[i] = 1;
709         nRow[i - 1] = 2;
710     }
711
712     // Set the first pixel
713     if (O[0] == 0) {
714         P[0] = 0;
715     } else if (O[0] == 1) {
716         P[0] = 1;
717     } else {
718         throw Exception::PixelValueOutOfBoundException();
719     }
720
721     // Walk through the toprow and determine if it's a new
722     // blob or it's connected
723     // with previously determine blob
724     for (uint32_t i = 1; i < nCols; i++) {
725         if (O[i] == 0) {
726             P[i] = 0;
727         } else if (O[i] == 1) {
728             // If West is zero assume this is a new blob
729             if (P[i - 1] == 0) {
730                 P[i] = ++currentlbl;
731                 vector<uint16_t> cVector;
732                 cVector.push_back(currentlbl);
733                 adj.push_back(cVector);
734             } else { // set as previous blob
735                 P[i] = P[i - 1];
736             }
737         } else { // Value of of bounds
738             throw Exception::PixelValueOutOfBoundException();
739         }
740     }
741
742     // walk through each pixel and determine if it's a new
743     // blob or it's connected
744     // with previously determine blob
745     for (uint32_t i = OriginalImg.cols; i < nData; i++) {
746         if (O[i] == 0) { // Original pixel = 0
747             P[i] = 0;
748         } else if (O[i] == 1) {
749             // Get an array of Neighboring Pixels
750             uint16_t *nPixels = new uint16_t[noConn[nRow[i]]];
751             if (nRow[i] != 1) {
752                 nPixels[0] = P[i - 1];
753             }
754             for (int j = 1; j < noConn[nRow[i]]; j++) {
755                 nPixels[j] = P[i + j];
756             }
757             for (int j = 0; j < noConn[nRow[i]]; j++) {
758                 P[i + j] = nPixels[j];
759             }
760             delete[] nPixels;
761         }
762     }
763
764     // Set the first pixel
765     if (O[0] == 0) {
766         P[0] = 0;
767     } else if (O[0] == 1) {
768         P[0] = 1;
769     } else {
770         throw Exception::PixelValueOutOfBoundException();
771     }
772
773     // Set the first pixel
774     if (O[0] == 0) {
775         P[0] = 0;
776     } else if (O[0] == 1) {
777         P[0] = 1;
778     } else {
779         throw Exception::PixelValueOutOfBoundException();
780     }
781
782     // Set the first pixel
783     if (O[0] == 0) {
784         P[0] = 0;
785     } else if (O[0] == 1) {
786         P[0] = 1;
787     } else {
788         throw Exception::PixelValueOutOfBoundException();
789     }
790
791     // Set the first pixel
792     if (O[0] == 0) {
793         P[0] = 0;
794     } else if (O[0] == 1) {
795         P[0] = 1;
796     } else {
797         throw Exception::PixelValueOutOfBoundException();
798     }
799
800     // Set the first pixel
801     if (O[0] == 0) {
802         P[0] = 0;
803     } else if (O[0] == 1) {
804         P[0] = 1;
805     } else {
806         throw Exception::PixelValueOutOfBoundException();
807     }
808
809     // Set the first pixel
810     if (O[0] == 0) {
811         P[0] = 0;
812     } else if (O[0] == 1) {
813         P[0] = 1;
814     } else {
815         throw Exception::PixelValueOutOfBoundException();
816     }
817
818     // Set the first pixel
819     if (O[0] == 0) {
820         P[0] = 0;
821     } else if (O[0] == 1) {
822         P[0] = 1;
823     } else {
824         throw Exception::PixelValueOutOfBoundException();
825     }
826
827     // Set the first pixel
828     if (O[0] == 0) {
829         P[0] = 0;
830     } else if (O[0] == 1) {
831         P[0] = 1;
832     } else {
833         throw Exception::PixelValueOutOfBoundException();
834     }
835
836     // Set the first pixel
837     if (O[0] == 0) {
838         P[0] = 0;
839     } else if (O[0] == 1) {
840         P[0] = 1;
841     } else {
842         throw Exception::PixelValueOutOfBoundException();
843     }
844
845     // Set the first pixel
846     if (O[0] == 0) {
847         P[0] = 0;
848     } else if (O[0] == 1) {
849         P[0] = 1;
850     } else {
851         throw Exception::PixelValueOutOfBoundException();
852     }
853
854     // Set the first pixel
855     if (O[0] == 0) {
856         P[0] = 0;
857     } else if (O[0] == 1) {
858         P[0] = 1;
859     } else {
860         throw Exception::PixelValueOutOfBoundException();
861     }
862
863     // Set the first pixel
864     if (O[0] == 0) {
865         P[0] = 0;
866     } else if (O[0] == 1) {
867         P[0] = 1;
868     } else {
869         throw Exception::PixelValueOutOfBoundException();
870     }
871
872     // Set the first pixel
873     if (O[0] == 0) {
874         P[0] = 0;
875     } else if (O[0] == 1) {
876         P[0] = 1;
877     } else {
878         throw Exception::PixelValueOutOfBoundException();
879     }
880
881     // Set the first pixel
882     if (O[0] == 0) {
883         P[0] = 0;
884     } else if (O[0] == 1) {
885         P[0] = 1;
886     } else {
887         throw Exception::PixelValueOutOfBoundException();
888     }
889
890     // Set the first pixel
891     if (O[0] == 0) {
892         P[0] = 0;
893     } else if (O[0] == 1) {
894         P[0] = 1;
895     } else {
896         throw Exception::PixelValueOutOfBoundException();
897     }
898
899     // Set the first pixel
900     if (O[0] == 0) {
901         P[0] = 0;
902     } else if (O[0] == 1) {
903         P[0] = 1;
904     } else {
905         throw Exception::PixelValueOutOfBoundException();
906     }
907
908     // Set the first pixel
909     if (O[0] == 0) {
910         P[0] = 0;
911     } else if (O[0] == 1) {
912         P[0] = 1;
913     } else {
914         throw Exception::PixelValueOutOfBoundException();
915     }
916
917     // Set the first pixel
918     if (O[0] == 0) {
919         P[0] = 0;
920     } else if (O[0] == 1) {
921         P[0] = 1;
922     } else {
923         throw Exception::PixelValueOutOfBoundException();
924     }
925
926     // Set the first pixel
927     if (O[0] == 0) {
928         P[0] = 0;
929     } else if (O[0] == 1) {
930         P[0] = 1;
931     } else {
932         throw Exception::PixelValueOutOfBoundException();
933     }
934
935     // Set the first pixel
936     if (O[0] == 0) {
937         P[0] = 0;
938     } else if (O[0] == 1) {
939         P[0] = 1;
940     } else {
941         throw Exception::PixelValueOutOfBoundException();
942     }
943
944     // Set the first pixel
945     if (O[0] == 0) {
946         P[0] = 0;
947     } else if (O[0] == 1) {
948         P[0] = 1;
949     } else {
950         throw Exception::PixelValueOutOfBoundException();
951     }
952
953     // Set the first pixel
954     if (O[0] == 0) {
955         P[0] = 0;
956     } else if (O[0] == 1) {
957         P[0] = 1;
958     } else {
959         throw Exception::PixelValueOutOfBoundException();
960     }
961
962     // Set the first pixel
963     if (O[0] == 0) {
964         P[0] = 0;
965     } else if (O[0] == 1) {
966         P[0] = 1;
967     } else {
968         throw Exception::PixelValueOutOfBoundException();
969     }
970
971     // Set the first pixel
972     if (O[0] == 0) {
973         P[0] = 0;
974     } else if (O[0] == 1) {
975         P[0] = 1;
976     } else {
977         throw Exception::PixelValueOutOfBoundException();
978     }
979
980     // Set the first pixel
981     if (O[0] == 0) {
982         P[0] = 0;
983     } else if (O[0] == 1) {
984         P[0] = 1;
985     } else {
986         throw Exception::PixelValueOutOfBoundException();
987     }
988
989     // Set the first pixel
990     if (O[0] == 0) {
991         P[0] = 0;
992     } else if (O[0] == 1) {
993         P[0] = 1;
994     } else {
995         throw Exception::PixelValueOutOfBoundException();
996     }
997
998     // Set the first pixel
999     if (O[0] == 0) {
1000        P[0] = 0;
1001    } else if (O[0] == 1) {
1002        P[0] = 1;
1003    } else {
1004        throw Exception::PixelValueOutOfBoundException();
1005    }
1006
1007     // Set the first pixel
1008     if (O[0] == 0) {
1009        P[0] = 0;
1010    } else if (O[0] == 1) {
1011        P[0] = 1;
1012    } else {
1013        throw Exception::PixelValueOutOfBoundException();
1014    }
1015
1016     // Set the first pixel
1017     if (O[0] == 0) {
1018        P[0] = 0;
1019    } else if (O[0] == 1) {
1020        P[0] = 1;
1021    } else {
1022        throw Exception::PixelValueOutOfBoundException();
1023    }
1024
1025     // Set the first pixel
1026     if (O[0] == 0) {
1027        P[0] = 0;
1028    } else if (O[0] == 1) {
1029        P[0] = 1;
1030    } else {
1031        throw Exception::PixelValueOutOfBoundException();
1032    }
1033
1034     // Set the first pixel
1035     if (O[0] == 0) {
1036        P[0] = 0;
1037    } else if (O[0] == 1) {
1038        P[0] = 1;
1039    } else {
1040        throw Exception::PixelValueOutOfBoundException();
1041    }
1042
1043     // Set the first pixel
1044     if (O[0] == 0) {
1045        P[0] = 0;
1046    } else if (O[0] == 1) {
1047        P[0] = 1;
1048    } else {
1049        throw Exception::PixelValueOutOfBoundException();
1050    }
1051
1052     // Set the first pixel
1053     if (O[0] == 0) {
1054        P[0] = 0;
1055    } else if (O[0] == 1) {
1056        P[0] = 1;
1057    } else {
1058        throw Exception::PixelValueOutOfBoundException();
1059    }
1060
1061     // Set the first pixel
1062     if (O[0] == 0) {
1063        P[0] = 0;
1064    } else if (O[0] == 1) {
1065        P[0] = 1;
1066    } else {
1067        throw Exception::PixelValueOutOfBoundException();
1068    }
1069
1070     // Set the first pixel
1071     if (O[0] == 0) {
1072        P[0] = 0;
1073    } else if (O[0] == 1) {
1074        P[0] = 1;
1075    } else {
1076        throw Exception::PixelValueOutOfBoundException();
1077    }
1078
1079     // Set the first pixel
1080     if (O[0] == 0) {
1081        P[0] = 0;
1082    } else if (O[0] == 1) {
1083        P[0] = 1;
1084    } else {
1085        throw Exception::PixelValueOutOfBoundException();
1086    }
1087
1088     // Set the first pixel
1089     if (O[0] == 0) {
1090        P[0] = 0;
1091    } else if (O[0] == 1) {
1092        P[0] = 1;
1093    } else {
1094        throw Exception::PixelValueOutOfBoundException();
1095    }
1096
1097     // Set the first pixel
1098     if (O[0] == 0) {
1099        P[0] = 0;
1100    } else if (O[0] == 1) {
1101        P[0] = 1;
1102    } else {
1103        throw Exception::PixelValueOutOfBoundException();
1104    }
1105
1106     // Set the first pixel
1107     if (O[0] == 0) {
1108        P[0] = 0;
1109    } else if (O[0] == 1) {
1110        P[0] = 1;
1111    } else {
1112        throw Exception::PixelValueOutOfBoundException();
1113    }
1114
1115     // Set the first pixel
1116     if (O[0] == 0) {
1117        P[0] = 0;
1118    } else if (O[0] == 1) {
1119        P[0] = 1;
1120    } else {
1121        throw Exception::PixelValueOutOfBoundException();
1122    }
1123
1124     // Set the first pixel
1125     if (O[0] == 0) {
1126        P[0] = 0;
1127    } else if (O[0] == 1) {
1128        P[0] = 1;
1129    } else {
1130        throw Exception::PixelValueOutOfBoundException();
1131    }
1132
1133     // Set the first pixel
1134     if (O[0] == 0) {
1135        P[0] = 0;
1136    } else if (O[0] == 1) {
1137        P[0] = 1;
1138    } else {
1139        throw Exception::PixelValueOutOfBoundException();
1140    }
1141
1142     // Set the first pixel
1143     if (O[0] == 0) {
1144        P[0] = 0;
1145    } else if (O[0] == 1) {
1146        P[0] = 1;
1147    } else {
1148        throw Exception::PixelValueOutOfBoundException();
1149    }
1150
1151     // Set the first pixel
1152     if (O[0] == 0) {
1153        P[0] = 0;
1154    } else if (O[0] == 1) {
1155        P[0] = 1;
1156    } else {
1157        throw Exception::PixelValueOutOfBoundException();
1158    }
1159
1160     // Set the first pixel
1161     if (O[0] == 0) {
1162        P[0] = 0;
1163    } else if (O[0] == 1) {
1164        P[0] = 1;
1165    } else {
1166        throw Exception::PixelValueOutOfBoundException();
1167    }
1168
1169     // Set the first pixel
1170     if (O[0] == 0) {
1171        P[0] = 0;
1172    } else if (O[0] == 1) {
1173        P[0] = 1;
1174    } else {
1175        throw Exception::PixelValueOutOfBoundException();
1176    }
1177
1178     // Set the first pixel
1179     if (O[0] == 0) {
1180        P[0] = 0;
1181    } else if (O[0] == 1) {
1182        P[0] = 1;
1183    } else {
1184        throw Exception::PixelValueOutOfBoundException();
1185    }
1186
1187     // Set the first pixel
1188     if (O[0] == 0) {
1189        P[0] = 0;
1190    } else if (O[0] == 1) {
1191        P[0] = 1;
1192    } else {
1193        throw Exception::PixelValueOutOfBoundException();
1194    }
1195
1196     // Set the first pixel
1197     if (O[0] == 0) {
1198        P[0] = 0;
1199    } else if (O[0] == 1) {
1200        P[0] = 1;
1201    } else {
1202        throw Exception::PixelValueOutOfBoundException();
1203    }
1204
1205     // Set the first pixel
1206     if (O[0] == 0) {
1207        P[0] = 0;
1208    } else if (O[0] == 1) {
1209        P[0] = 1;
1210    } else {
1211        throw Exception::PixelValueOutOfBoundException();
1212    }
1213
1214     // Set the first pixel
1215     if (O[0] == 0) {
1216        P[0] = 0;
1217    } else if (O[0] == 1) {
1218        P[0] = 1;
1219    } else {
1220        throw Exception::PixelValueOutOfBoundException();
1221    }
1222
1223     // Set the first pixel
1224     if (O[0] == 0) {
1225        P[0] = 0;
1226    } else if (O[0] == 1) {
1227        P[0] = 1;
1228    } else {
1229        throw Exception::PixelValueOutOfBoundException();
1230    }
1231
1232     // Set the first pixel
1233     if (O[0] == 0) {
1234        P[0] = 0;
1235    } else if (O[0] == 1) {
1236        P[0] = 1;
1237    } else {
1238        throw Exception::PixelValueOutOfBoundException();
1239    }
1240
1241     // Set the first pixel
1242     if (O[0] == 0) {
1243        P[0] = 0;
1244    } else if (O[0] == 1) {
1245        P[0] = 1;
1246    } else {
1247        throw Exception::PixelValueOutOfBoundException();
1248    }
1249
1250     // Set the first pixel
1251     if (O[0] == 0) {
1252        P[0] = 0;
1253    } else if (O[0] == 1) {
1254        P[0] = 1;
1255    } else {
1256        throw Exception::PixelValueOutOfBoundException();
1257    }
1258
1259     // Set the first pixel
1260     if (O[0] == 0) {
1261        P[0] = 0;
1262    } else if (O[0] == 1) {
1263        P[0] = 1;
1264    } else {
1265        throw Exception::PixelValueOutOfBoundException();
1266    }
1267
1268     // Set the first pixel
1269     if (O[0] == 0) {
1270        P[0] = 0;
1271    } else if (O[0] == 1) {
1272        P[0] = 1;
1273    } else {
1274        throw Exception::PixelValueOutOfBoundException();
1275    }
1276
1277     // Set the first pixel
1278     if (O[0] == 0) {
1279        P[0] = 0;
1280    } else if (O[0] == 1) {
1281        P[0] = 1;
1282    } else {
1283        throw Exception::PixelValueOutOfBoundException();
1284    }
1285
1286     // Set the first pixel
1287     if (O[0] == 0) {
1288        P[0] = 0;
1289    } else if (O[0] == 1) {
1290        P[0] = 1;
1291    } else {
1292        throw Exception::PixelValueOutOfBoundException();
1293    }
1294
1295     // Set the first pixel
1296     if (O[0] == 0) {
1297        P[0] = 0;
1298    } else if (O[0] == 1) {
1299        P[0] = 1;
1300    } else {
1301        throw Exception::PixelValueOutOfBoundException();
1302    }
1303
1304     // Set the first pixel
1305     if (O[0] == 0) {
1306        P[0] = 0;
1307    } else if (O[0] == 1) {
1308        P[0] = 1;
1309    } else {
1310        throw Exception::PixelValueOutOfBoundException();
1311    }
1312
1313     // Set the first pixel
1314     if (O[0] == 0) {
1315        P[0] = 0;
1316    } else if (O[0] == 1) {
1317        P[0] = 1;
1318    } else {
1319        throw Exception::PixelValueOutOfBoundException();
1320    }
1321
1322     // Set the first pixel
1323     if (O[0] == 0) {
1324        P[0] = 0;
1325    } else if (O[0] == 1) {
1326        P[0] = 1;
1327    } else {
1328        throw Exception::PixelValueOutOfBoundException();
1329    }
1330
1331     // Set the first pixel
1332     if (O[0] == 0) {
1333        P[0] = 0;
1334    } else if (O[0] == 1) {
1335        P[0] = 1;
1336    } else {
1337        throw Exception::PixelValueOutOfBoundException();
1338    }
1339
1340     // Set the first pixel
1341     if (O[0] == 0) {
1342        P[0] = 0;
1343    } else if (O[0] == 1) {
1344        P[0] = 1;
1345    } else {
1346        throw Exception::PixelValueOutOfBoundException();
1347    }
1348
1349     // Set the first pixel
1350     if (O[0] == 0) {
1351        P[0] = 0;
1352    } else if (O[0] == 1) {
1353        P[0] = 1;
1354    } else {
1355        throw Exception::PixelValueOutOfBoundException();
1356    }
1357
1358     // Set the first pixel
1359     if (O[0] == 0) {
1360        P[0] = 0;
1361    } else if (O[0] == 1) {
1362        P[0] = 1;
1363    } else {
1364        throw Exception::PixelValueOutOfBoundException();
1365    }
1366
1367     // Set the first pixel
1368     if (O[0] == 0) {
1369        P[0] = 0;
1370    } else if (O[0] == 1) {
1371        P[0] = 1;
1372    } else {
1373        throw Exception::PixelValueOutOfBoundException();
1374    }
1375
1376     // Set the first pixel
1377     if (O[0] == 0) {
1378        P[0] = 0;
1379    } else if (O[0] == 1) {
1380        P[0] = 1;
1381    } else {
1382        throw Exception::PixelValueOutOfBoundException();
1383    }
1384
1385     // Set the first pixel
1386     if (O[0] == 0) {
1387        P[0] = 0;
1388    } else if (O[0] == 1) {
1389        P[0] = 1;
1390    } else {
1391        throw Exception::PixelValueOutOfBoundException();
1392    }
1393
1394     // Set the first pixel
1395     if (O[0] == 0) {
1396        P[0] = 0;
1397    } else if (O[0] == 1) {
1398        P[0] = 1;
1399    } else {
1400        throw Exception::PixelValueOutOfBoundException();
1401    }
1402
1403     // Set the first pixel
1404     if (O[0] == 0) {
1405        P[0] = 0;
1406    } else if (O[0] == 1) {
1407        P[0] = 1;
1408    } else {
1409        throw Exception::PixelValueOutOfBoundException();
1410    }
1411
1412     // Set the first pixel
1413     if (O[0] == 0) {
1414        P[0] = 0;
1415    } else if (O[0] == 1) {
1416        P[0] = 1;
1417    } else {
1418        throw Exception::PixelValueOutOfBoundException();
1419    }
1420
1421     // Set the first pixel
1422     if (O[0] == 0) {
1423        P[0] = 0;
1424    } else if (O[0] == 1) {
1425        P[0] = 1;
1426    } else {
1427        throw Exception::PixelValueOutOfBoundException();
1428    }
1429
1430     // Set the first pixel
1431     if (O[0] == 0) {
1432        P[0] = 0;
1433    } else if (O[0] == 1) {
1434        P[0] = 1;
1435    } else {
1436        throw Exception::PixelValueOutOfBoundException();
1437    }
1438
1439     // Set the first pixel
1440     if (O[0] == 0) {
1441        P[0] = 0;
1442    } else if (O[0] == 1) {
1443        P[0] = 1;
1444    } else {
1445        throw Exception::PixelValueOutOfBoundException();
1446    }
1447
1448     // Set the first pixel
1449     if (O[0] == 0) {
1450        P[0] = 0;
1451    } else if (O[0] == 1) {
1452        P[0] = 1;
1453    } else {
1454        throw Exception::PixelValueOutOfBoundException();
1455    }
1456
1457     // Set the first pixel
1458     if (O[0] == 0) {
1459        P[0] = 0;
1460    } else if (O[0] == 1) {
1461        P[0] = 1;
1462    } else {
1463        throw Exception::PixelValueOutOfBoundException();
1464    }
1465
1466     // Set the first pixel
1467     if (O[0] == 0) {
1468        P[0] = 0;
1469    } else if (O[0] == 1) {
1470        P[0] = 1;
1471    } else {
1472        throw Exception::PixelValueOutOfBoundException();
1473    }
1474
1475     // Set the first pixel
1476     if (O[0] == 0) {
1477        P[0] = 0;
1478    } else if (O[0] == 1) {
1479        P[0] = 1;
1480    } else {
1481        throw Exception::PixelValueOutOfBoundException();
1482    }
1483
1484     // Set the first pixel
1485     if (O[0] == 0) {
1486        P[0] = 0;
1487    } else if (O[0] == 1) {
1488        P[0] = 1;
1489    } else {
1490        throw Exception::PixelValueOutOfBoundException();
1491    }
1492
1493     // Set the first pixel
1494     if (O[0] == 0) {
1495        P[0] = 0;
1496    } else if (O[0] == 1) {
1497        P[0] = 1;
1498    } else {
1499        throw Exception::PixelValueOutOfBoundException();
1500    }
1501
1502     // Set the first pixel
1503     if (O[0] == 0) {
1504        P[0] = 0;
1505    } else if (O[0] == 1) {
1506        P[0] = 1;
1507    } else {
1508        throw Exception::PixelValueOutOfBoundException();
1509    }
1510
1511     // Set the first pixel
1512     if (O[0] == 0) {
1513        P[0] = 0;
1514    } else if (O[0] == 1) {
1515        P[0] = 1;
1516    } else {
1517        throw Exception::PixelValueOutOfBoundException();
1518    }
1519
1520     // Set the first pixel
1521     if (O[0] == 0) {
1522        P[0] = 0;
1523    } else if (O[0] == 1) {
1524        P[0] = 1;
1525    } else {
1526        throw Exception::PixelValueOutOfBoundException();
1527    }
1528
1529     // Set the first pixel
1530     if (O[0] == 0) {
1531        P[0] = 0;
1532    } else if (O[0] == 1) {
1533        P[0] = 1;
1534    } else {
1535        throw Exception::PixelValueOutOfBoundException();
1536    }
1537
1538     // Set the first pixel
1539     if (O[0] == 0) {
1540        P[0] = 0;
1541    } else if (O[0] == 1) {
1542        P[0] = 1;
1543    } else {
1544        throw Exception::PixelValueOutOfBoundException();
1545    }
1546
1547     // Set the first pixel
1548     if (O[0] == 0) {
1549        P[0] = 0;
1550    } else if (O[0] == 1) {
1551        P[0] = 1;
1552    } else {
1553        throw Exception::PixelValueOutOfBoundException();
1554    }
1555
1556     // Set the first pixel
1557     if (O[0] == 0) {
1558        P[0] = 0;
1559    } else if (O[0] == 1) {
1560        P[0] = 1;
1561    } else {
1562        throw Exception::PixelValueOutOfBoundException();
1563    }
1564
1565     // Set the first pixel
1566     if (O[0] == 0) {
1567        P[0] = 0;
1568    } else if (O[0] == 1) {
1569        P[0] = 1;
1570    } else {
1571        throw Exception::PixelValueOutOfBoundException();
1572    }
1573
1574     // Set the first pixel
1575     if (O[0] == 0) {
1576        P[0] = 0;
1577    } else if (O[0] == 1) {
1578        P[0] = 1;
1579    } else {
1580        throw Exception::PixelValueOutOfBoundException();
1581    }
1582
1583     // Set the first pixel
1584     if (O[0] == 0) {
1585        P[0] = 0;
1586    } else if (O[0] == 1) {
1587        P[0] = 1;
1588    } else {
1589        throw Exception::PixelValueOutOfBoundException();
1590    }
1591
1592     // Set the first pixel
1593     if (O[0] == 0) {
1594        P[0] = 0;
1595    } else if (O[0] == 1) {
1596        P[0] = 1;
1597    } else {
1598        throw Exception::PixelValueOutOfBoundException();
1599    }
1600
1601     // Set the first pixel
1602     if (O[0] == 0) {
1603        P[0] = 0;
1604    } else if (O[0] == 1) {
1605        P[0] = 1;
1606    } else {
1607        throw Exception::PixelValueOutOfBoundException();
1608    }
1609
1610     // Set the first pixel
1611     if (O[0] == 0) {
1612        P[0] = 0;
1613    } else if (O[0] == 1) {
1614        P[0] = 1;
1615    } else {
1616        throw Exception::PixelValueOutOfBoundException();
1617    }
1618
1619     // Set the first pixel
1620     if (O[0] == 0) {
1621        P[0] = 0;
1622    } else if (O[0] == 1) {
1623        P[0] = 1;
1624    } else {
1625        throw Exception::PixelValueOutOfBoundException();
1626    }
1627
1628     // Set the first pixel
1629     if (O[0] == 0) {
1630        P[0] = 0;
1631    } else if (O[0] == 1) {
1632        P[0] = 1;
1633    } else {
1634        throw Exception::PixelValueOutOfBoundException();
1635    }
1636
1637     // Set the first pixel
1638     if (O[0] == 0) {
1639        P[0] = 0;
1640    } else if (O[0] == 1) {
1641        P[0] = 1;
1642    } else {
1643        throw Exception::PixelValueOutOfBoundException();
1644    }
1645
1646     // Set the first pixel
1647     if (O[0] == 0) {
1648        P[0] = 0;
1649    } else if (O[0] == 1) {
1650        P[0] = 1;
1651    } else {
1652        throw Exception::PixelValueOutOfBoundException();
1653    }
1654
1655     // Set the first pixel
1656     if (O[0] == 0) {
1657        P[0] = 0;
1658    } else if (O[0] == 1) {
1659        P[0] = 1;
1660    } else {
1661        throw Exception::PixelValueOutOfBoundException();
1662    }
1663
1664     // Set the first pixel
1665     if (O[0] == 0) {
1666        P[0] = 0;
1667    } else if (O[0] == 1) {
1668        P[0] = 1;
1669    } else {
1670        throw Exception::PixelValueOutOfBoundException();
1671   
```

```

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

```

```

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

```

```
846                               uint16_t noElem, uint16_t &
847                               maxlabel) {
848     SoilMath::Sort::QuickSort<uint16_t>(valueArr, keyArr,
849                                         noElem);
850     uint16_t count = 0;
851     for (uint32_t i = 1; i < noElem; i++) {
852         if (valueArr[i] != valueArr[i - 1]) {
853             count++;
854         }
855         valueArr[i] = count;
856     }
857     SoilMath::Sort::QuickSort<uint16_t>(keyArr, valueArr,
858                                         noElem);
859     delete[] keyArr;
860     maxlabel = count;
861 }
```

General project files

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

```

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


---


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


---


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


---


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

```

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

```
18 namespace Vision {
19 namespace Exception {
20 class ConversionNotSupportedException : public std::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
27         msg.c_str(); };
28
29 private:
30     string msg;
31 }
32 }
```

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

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

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

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



K. Analyzer Library

Analyzer Class

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

```

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

```

```

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



---


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

```

```
30  /*!
31   * \brief Analyzer::PrepImages
32   */
33  void Analyzer::PrepImages() {
34      if (Snapshots == nullptr || Snapshots->size() == 0) {
35          throw Exception::SoilAnalyzerException(
36              EXCEPTION_NO_SNAPSHOTS,
37              EXCEPTION_NO_SNAPSHOTS_NR
38          );
39      }
40
41      std::vector<cv::Mat> intensityVector;
42      GetEnhancedInt(Snapshots, intensityVector);
43
44      std::vector<cv::Mat> BWVector;
45      GetBW(intensityVector, BWVector);
46      CleanUpMatVector(intensityVector);
47
48      GetParticles(BWVector, Snapshots, Results->
49          ParticlePopulation);
50
51      CleanUpMatVector(BWVector);
52      CleanUpMatVector(Snapshots);
53
54      Results->isPreparedForAnalysis = true;
55  }
56
57  void Analyzer::Analyse(Images_t *snapshots, Sample *results,
58                         SoilSettings *settings) {
59      Snapshots = snapshots;
60      Results = results;
61      Settings = settings;
62      Analyse();
63  }
64
65  /*!
66   * \brief Analyzer::Analyse
67   */
68  void Analyzer::Analyse() {
69      CalcMaxProgress();
70      if (!Results->isPreparedForAnalysis && !Results->
71          IsLoadedFromDisk) {
72          PrepImages();
73      }
74
75      Results->Angularity =
76          ucharStat_t(Results->GetAngularityVector()->data(),
77                      Results->GetAngularityVector()->size(), 1,
78                      7, 0, true);
79      emit on_progressUpdate(currentProgress++);
80
81      Results->Roundness =
```

```

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

```

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

```

```

183     Vision::Enhance IntAdaptContrast(
184         IntBlur.ProcessedImg(
185             cv::Rect(HBK, HBK, IntBlur.
186                 ProcessedImg.cols - BK,
187                     IntBlur.ProcessedImg.rows -
188                         BK)).clone());
188     IntAdaptContrast.AdaptiveContrastStretch(
189         Settings->adaptContrastKernelSize,
190         Settings->adaptContrastKernelFactor);
191     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.
195             cols - AK,
196                 IntAdaptContrast.ProcessedImg.rows - AK));
196             ;
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.
203         ProcessedImg.clone());
204     IntAdaptContrast.AdaptiveContrastStretch(
205         Settings->adaptContrastKernelSize, Settings->
206             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.
210             cols - AK,
211                 IntAdaptContrast.ProcessedImg.rows - AK));
211     } else {
212         intensity = IntConvertor.ProcessedImg;
213     }
214     SHOW_DEBUG_IMG(intensity, uchar, 255, "Enhanced Int",
215         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

```

```

231  /*!
232  * \brief Analyzer::GetBW
233  * \param img
234  * \param BW
235  */
236 void Analyzer::GetBW(cv::Mat &img, cv::Mat &BW) {
237     Vision::Segment SegBL(img.clone());
238     SegBL.sigma = Settings->sigmaFactor;
239     SegBL.thresholdOffset = Settings->thresholdOffsetValue;
240     SegBL.ConvertToBW(Settings->typeOfObjectsSegmented);
241     emit on_progressUpdate(currentProgress++);
242     SHOW_DEBUG_IMG(SegBL.ProcessedImg, uchar, 255, "Segment",
243                     true);
244     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"
262                         , true);
262     } else {
263         BWborder = BWholes;
264     }
265
266     if (Settings->morphFilterType != Vision::
267         MorphologicalFilter::NONE) {
268         Vision::MorphologicalFilter Morph(BWborder.clone());
269         cv::Mat kernel = cv::Mat::zeros(Settings->filterMaskSize
270                                         ,
271                                         Settings->filterMaskSize
272                                         , CV_8UC1);
270         uint32_t hMaskSize = Settings->filterMaskSize / 2;
271         cv::circle(kernel, cv::Point(hMaskSize, hMaskSize),
272                    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"
292         , true);
293 } else {
294     BW = BWholes;
295 }
296 }
297 */
298 * \brief Analyzer::GetParticles
299 * \param BW
300 * \param snapshots
301 * \param partPopulation
302 */
303 void Analyzer::GetParticles(std::vector<Mat> &BW, Images_t *
304     snapshots,
305     Particle::ParticleVector_t &
306     partPopulation) {
307     for (uint32_t i = 0; i < snapshots->size(); i++) {
308         Vision::Segment prepBW(BW[i]);
309         prepBW.GetBlobList();
310         emit on_progressUpdate(currentProgress++);
311         GetParticlesFromBlobList(prepBW.BlobList, &(snapshots->
312             at(i)),
313             partPopulation);
314         emit on_progressUpdate(currentProgress++);
315     }
316 }
317 */
318 * \brief Analyzer::GetParticlesFromBlobList
319 * \param bloblist
320 * \param snapshot
321 * \param edge
322 * \param partPopulation
323 */
324 void Analyzer::GetParticlesFromBlobList(
325     Vision::Segment::BlobList_t &bloblist, Image_t *snapshot
326     ,
327     Particle::ParticleVector_t &partPopulation) {
328     for_each(bloblist.begin(), bloblist.end(), [&](Vision::
329         Segment::Blob_t &B) {
330         Particle part;
331         part.ID = currentParticleID++;
332         part.PixelArea = B.Area;
333         Vision::Segment::getOriententated(B.Img, B.Centroid, B.
334             Theta,

```

```

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

```

```

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

```

Sample Class

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

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

Particle Class

```

1  /* Copyright (C) Jelle Spijker - All Rights Reserved
2   * Unauthorized copying of this file, via any medium is
3   * strictly prohibited
4   * and only allowed with the written consent of the author (
5   * Jelle Spijker)
6   * This software is proprietary and confidential
7   * Written by Jelle Spijker <spijkert.jelle@gmail.com>, 2015
8   */
9  #pragma once
10
11 #include <opencv2/core.hpp>
12 #include <stdint.h>
13 #include <vector>
14 #include "SoilMath.h"
15 #include <fstream>
16 #include <boost/archive/binary_iarchive.hpp>
17 #include <boost/archive/binary_oarchive.hpp>
18 #include <boost/serialization/string.hpp>
19 #include <boost/serialization/version.hpp>
20 #include <boost/serialization/vector.hpp>
21 #include <boost/iostreams/filter/zlib.hpp>
22 #include <boost/iostreams/filtering_streambuf.hpp>
23 #include "zlib.h"
24 #include "soilanalyzerexception.h"
25 #include "lab_t_archive.h"
26 #include "soilanalyzertypes.h"
27 #include "Vision.h"
28
29 namespace SoilAnalyzer {
30 class Particle {
31 public:
32     typedef std::vector<Particle>
33     ParticleVector_t; /*!< a vector consisting of
34     individual particles*/
35     typedef std::vector<double> PSDVector_t; /*!< a vector
36     used in the PSD*/
37     typedef std::vector<uint8_t>
38     ClassVector_t; /*!< a vector used in the
39     classification histogram*/
40     typedef std::vector<float> floatVector_t;
41     typedef std::vector<double> doubleVector_t;
42
43     Particle();
44
45     uint32_t ID; /*!< The particle ID*/
46
47     cv::Mat BW; /*!< The binary image of the particle*/
48     cv::Mat Edge; /*!< The binary edge image of the particle*/
49     cv::Mat RGB; /*!< The RGB image of the particle*/
50
51     Point_t Centroid = {0, 0};
52     std::vector<Complex_t> FFDescriptors; /*!< The Fast
53     Fourier Descriptors

```

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

```

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


---


1 /* Copyright (C) Jelle Spijker - All Rights Reserved
2  * Unauthorized copying of this file, via any medium is
 * 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 <spijkerv.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                 );
26         out.push(ofs);
27     }

```

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

```

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

```

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

```

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

```

```

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

```

```

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

```

```

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

```

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

General project files

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

```

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

```

```

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

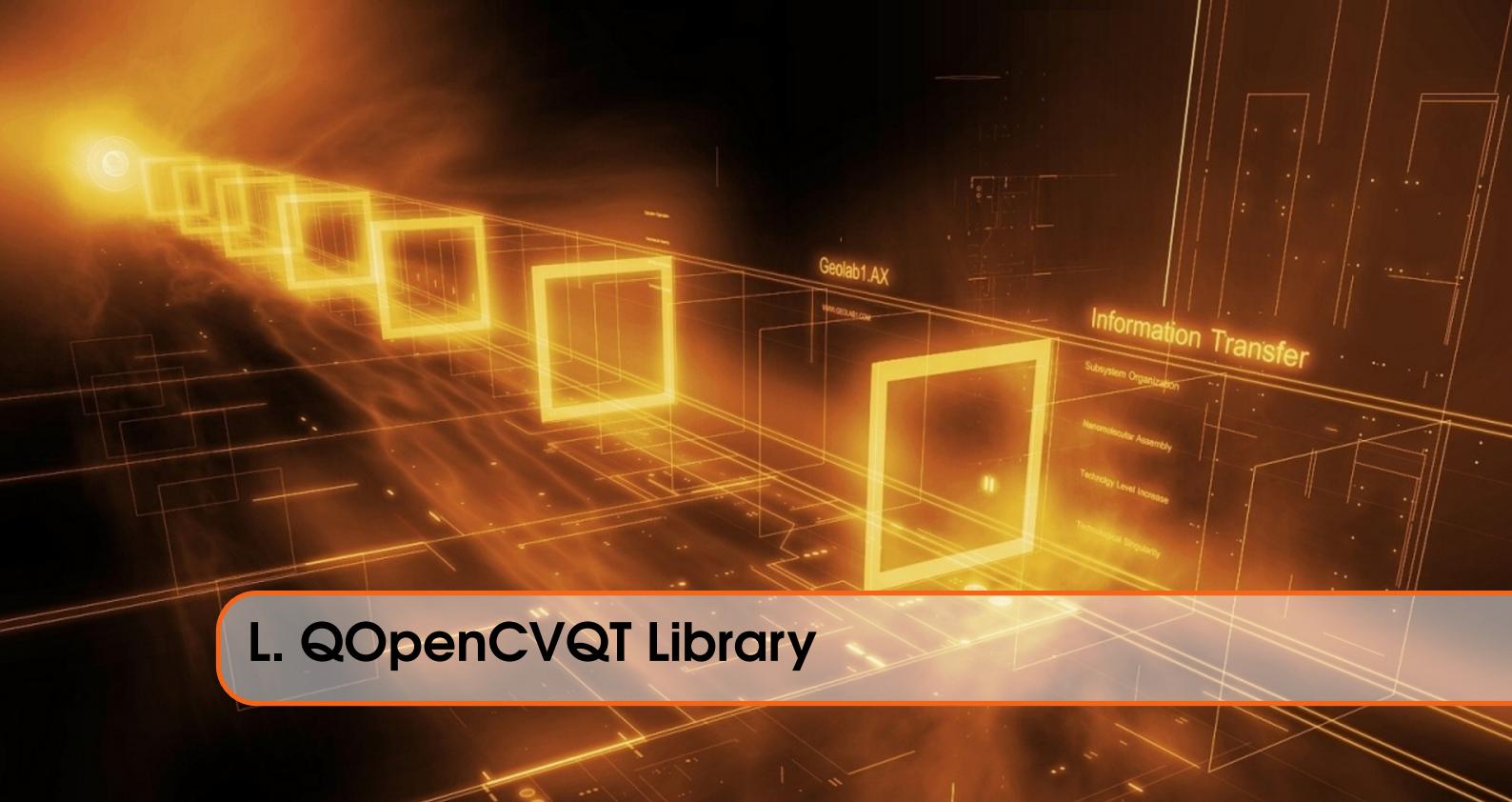
```

```

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

```

```
11 #define EXCEPTION_NO_IMAGES_PRESENT "No images to analyse
12 #define EXCEPTION_NO_IMAGES_PRESENT_NR 2
13
14 #pragma once
15 #include <exception>
16 #include <string>
17
18 namespace SoilAnalyzer {
19     namespace Exception {
20         class SoilAnalyzerException : public std::exception {
21     public:
22         SoilAnalyzerException(std::string m =
23             EXCEPTION_PARTICLE_NOT_ANALYZED,
24             int n =
25                 EXCEPTION_PARTICLE_NOT_ANALYZED_NR
26             ) : msg(m), nr(n) { }
27         ~SoilAnalyzerException() _GLIBCXX_USE_NOEXCEPT {}
28         const char *what() const _GLIBCXX_USE_NOEXCEPT {
29             return msg.c_str(); }
30         const int *id() const _GLIBCXX_USE_NOEXCEPT { return &
31             nr; }
32     private:
33         std::string msg;
34         int nr;
35     };
36 }
37
38 #ifndef SOILANALYZERTYPES
39 #define SOILANALYZERTYPES
40
41 namespace SoilAnalyzer {
42     struct Point_t {
43         double x;
44         double y;
45     };
46
47     struct Lab_t {
48         float L;
49         float a;
50         float b;
51     };
52 }
53 #endif // SOILANALYZERTYPES
```



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

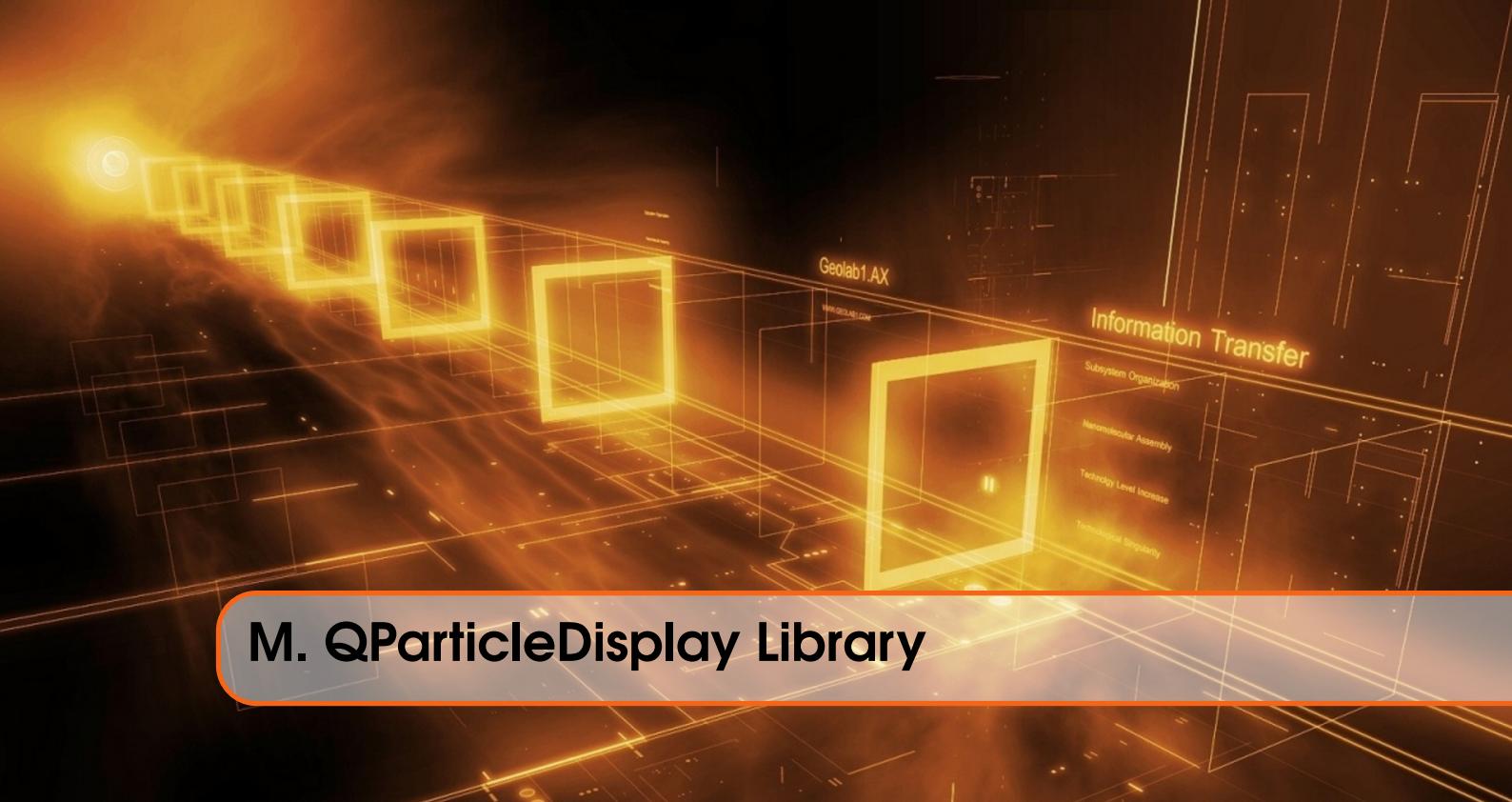
```

```

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

```

```
7
8 #include "qopencvqt.h"
9
10
11 QOpenCVQT::QOpenCVQT()
12 {
13 }
```



M. QParticleDisplay Library

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

```

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

```

```

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

```

```

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


---


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

```

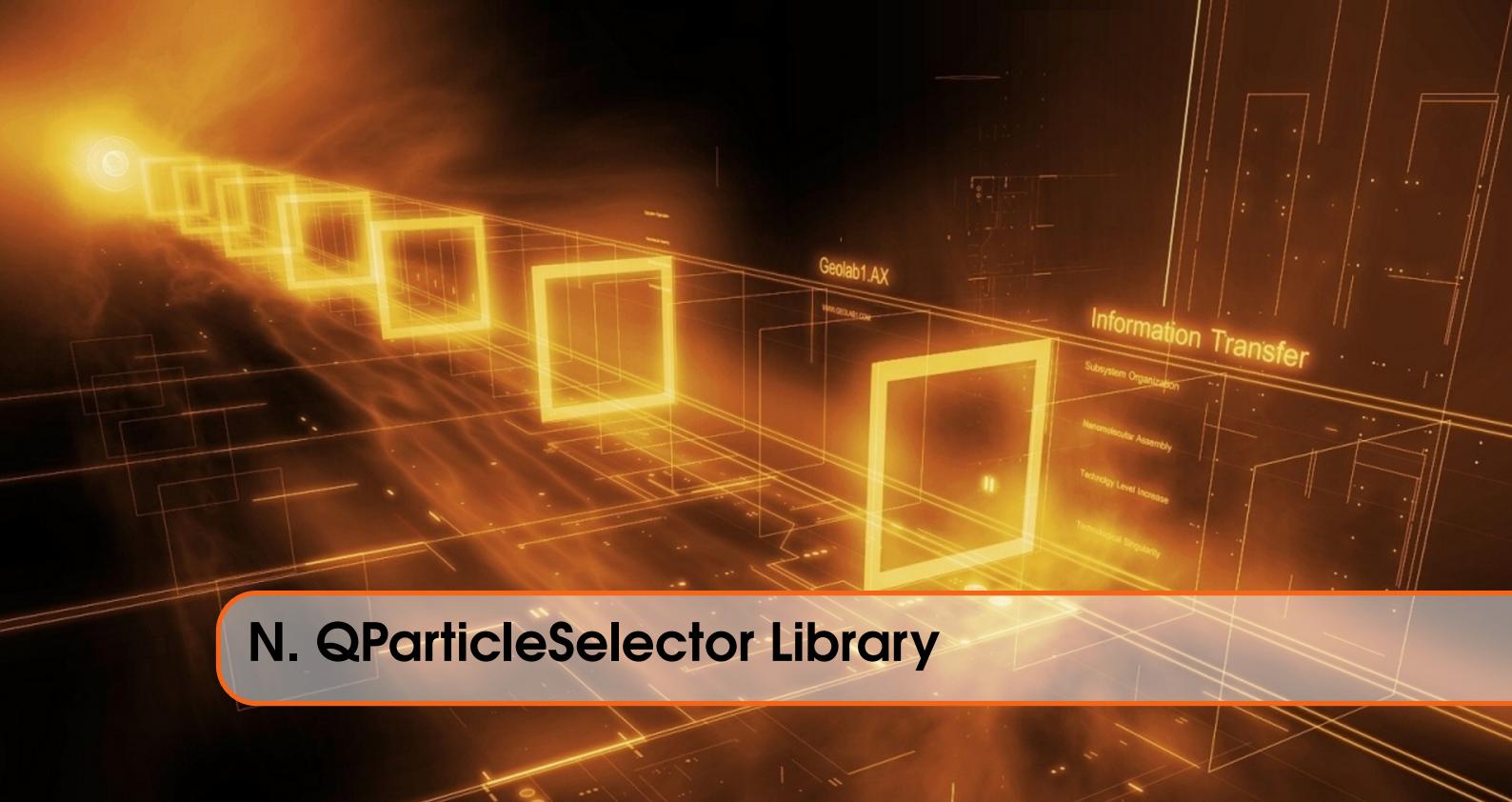
```
31     ui->widget->setCenterIndex(newValue);
32     ui->horizontalSlider->setValue(newValue);
33 }
34
35 void QParticleDisplay::SetSample(SoilAnalyzer::Sample *
36     sample) {
37     this->Sample = sample;
38     images.clear();
39     ui->widget->clear();
40     ui->horizontalSlider->setMaximum(this->Sample->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

```

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

```

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



N. QParticleSelector Library

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

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


---


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

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


---


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

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

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

```

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

O. QReportGenerator Library

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

```

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

```

```

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

```

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

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

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

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

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

```

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

```

```
180 rCurs.insertText(QString::number(Sample->PSD.bins[13]),  
181     GtxtFormat);  
182 rCurs.movePosition(QTextCursor::NextCell);  
183 rCurs.insertText("1", GFieldtxtFormat);  
184 rCurs.movePosition(QTextCursor::NextCell);  
185 rCurs.insertText(QString::number(Sample->PSD.CFD[12]),  
186     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]),  
191     GtxtFormat);  
192 rCurs.movePosition(QTextCursor::NextCell);  
193 rCurs.insertText(QString::number(Sample->PSD.bins[11]),  
194     GtxtFormat);  
195 rCurs.movePosition(QTextCursor::NextCell);  
196 rCurs.insertText(QString::number(Sample->PSD.CFD[10]),  
197     GtxtFormat);  
198 rCurs.movePosition(QTextCursor::NextCell);  
199 rCurs.insertText(QString::number(Sample->PSD.bins[10]),  
200     GtxtFormat);  
201 rCurs.movePosition(QTextCursor::NextCell);  
202 rCurs.insertText(QString::number(Sample->PSD.CFD[9]),  
203     GtxtFormat);  
204 rCurs.movePosition(QTextCursor::NextCell);  
205 rCurs.insertText(QString::number(Sample->PSD.bins[9]),  
206     GtxtFormat);  
207 rCurs.movePosition(QTextCursor::NextCell);  
208 rCurs.insertText(QString::number(Sample->PSD.CFD[8]),  
209     GtxtFormat);  
210 rCurs.movePosition(QTextCursor::NextCell);  
211 rCurs.insertText(QString::number(Sample->PSD.bins[8]),  
212     GtxtFormat);  
213 rCurs.movePosition(QTextCursor::NextCell);  
214 rCurs.insertText(QString::number(Sample->PSD.CFD[7]),  
215     GtxtFormat);  
216 rCurs.movePosition(QTextCursor::NextCell);  
217 rCurs.insertText(QString::number(Sample->PSD.bins[7]),  
218     GtxtFormat);  
219 rCurs.movePosition(QTextCursor::NextCell);  
220 rCurs.insertText(QString::number(Sample->PSD.CFD[6]),  
221     GtxtFormat);  
222 rCurs.movePosition(QTextCursor::NextCell);
```

```

222     rCurs.insertText(QString::number(Sample->PSD.bins[6]),
223         GtxtFormat);
224     rCurs.movePosition(QTextCursor::NextCell);
225     rCurs.insertText("0.09", GFieldtxtFormat);
226     rCurs.movePosition(QTextCursor::NextCell);
227     rCurs.insertText(QString::number(Sample->PSD.CFD[5]),
228         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]),
233         GtxtFormat);
234     rCurs.movePosition(QTextCursor::NextCell);
235     rCurs.insertText(QString::number(Sample->PSD.bins[4]),
236         GtxtFormat);
237     rCurs.movePosition(QTextCursor::NextCell);
238     rCurs.insertText(QString::number(Sample->PSD.CFD[3]),
239         GtxtFormat);
240     rCurs.movePosition(QTextCursor::NextCell);
241     rCurs.insertText(QString::number(Sample->PSD.bins[3]),
242         GtxtFormat);
243     rCurs.movePosition(QTextCursor::NextCell);
244     rCurs.insertText(QString::number(Sample->PSD.CFD[2]),
245         GtxtFormat);
246     rCurs.movePosition(QTextCursor::NextCell);
247     rCurs.insertText(QString::number(Sample->PSD.bins[2]),
248         GtxtFormat);
249     rCurs.movePosition(QTextCursor::NextCell);
250     rCurs.insertText(QString::number(Sample->PSD.CFD[1]),
251         GtxtFormat);
252     rCurs.movePosition(QTextCursor::NextCell);
253     rCurs.insertText(QString::number(Sample->PSD.bins[1]),
254         GtxtFormat);
255     rCurs.movePosition(QTextCursor::NextCell);
256     rCurs.insertText(QString::number(Sample->PSD.CFD[0]),
257         GtxtFormat);
258     rCurs.movePosition(QTextCursor::NextCell);
259     rCurs.insertText(QString::number(Sample->PSD.bins[0]),
260         GtxtFormat);
261 // Setup the Textdata for the Roundness
262     rCurs.insertBlock(HeaderFormat, HeaderTextFormat);
263     rCurs.insertText("Sphericity Classification");
264

```

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

```

310     rCurs.insertText("No of particles:", GFieldtxtFormat);
311     rCurs.movePosition(QTextCursor::NextCell);
312     rCurs.insertText(QString::number(Sample->Angularity.n),
313                     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),
318                     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),
323                     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),
328                     GtxtFormat);
328     rCurs.movePosition(QTextCursor::NextCell);
329
330     rCurs.insertText("Range: ", GFieldtxtFormat);
331     rCurs.movePosition(QTextCursor::NextCell);
332     rCurs.insertText(QString::number(Sample->Angularity.Range),
333                     GtxtFormat);
333     rCurs.movePosition(QTextCursor::NextCell);
334
335     rCurs.insertText("Standard deviation: ", GFieldtxtFormat);
336     rCurs.movePosition(QTextCursor::NextCell);
337     rCurs.insertText(QString::number(Sample->Angularity.Std),
338                     GtxtFormat);
338     rCurs.movePosition(QTextCursor::NextBlock);
339
340 // Setup the Roundness Graph
341 rCurs.insertBlock(ImageGraphFormat);
342 rCurs.insertText(QString(QChar::ObjectReplacementCharacter),
343                  QCPDocumentObject::generatePlotFormat(
344                  Angularity, 600, 400));
345
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 SetupCIELabPPlot();
351 rCurs.insertBlock(ImageGraphFormat);
352 rCurs.insertText(QString(QChar::ObjectReplacementCharacter),
353                  QCPDocumentObject::generatePlotFormat(
354                  CIELabPlot, 600, 400));
355 }
```

```
356
357 void QReportGenerator::getLocationMap(double &latitude,
358     double &longitude) {
359     QNetworkAccessManager *manager = new QNetworkAccessManager
360     ;
361     connect(manager, SIGNAL(finished(QNetworkReply *)), this,
362             SLOT(on_locationImageDownloaded(QNetworkReply *)))
363     ;
364     QString locationURL("http://maps.googleapis.com/maps/api/
365         staticmap?center=");
366     locationURL.append(QString::number(latitude));
367     locationURL.append(",");
368     locationURL.append(QString::number(longitude));
369     locationURL.append("&zoom=17&size=600x750&maptype=hybrid&&
370         format=png&visual_"
371         "refresh=true&markers=size:mid%7Ccolor
372             :0xff0000%7Clabel:S%
373             "7C");
374     locationURL.append(QString::number(latitude));
375     locationURL.append(",");
376     locationURL.append(QString::number(longitude));
377     qDebug() << locationURL;
378     QUrl googleStaticMapUrl(locationURL);
379     manager->get(QNetworkRequest(googleStaticMapUrl));
380 }
381
382 void QReportGenerator::on_locationImageDownloaded(
383     QNetworkReply *reply) {
384     if (mapLocation == nullptr) {
385         mapLocation = new QImage;
386     }
387     mapLocation->loadFromData(reply->readAll());
388
389     if (mapLocation->isNull()) {
390         mapLocation->load("Maps/SampleLocation.png");
391     }
392
393     QTextBlock location = Report->findBlockByNumber(15);
394     QTextCursor insertMap(location);
395     insertMap.setBlockFormat(ImageGraphFormat);
396     insertMap.insertImage(*mapLocation);
397     insertMap.insertBlock();
398     insertMap.insertHtml("<br>");
399 }
400
401 void QReportGenerator::~QReportGenerator()
402 {
403     delete CIELabPlot;
404     delete mapLocation;
405     delete ui;
406 }
407
408 void QReportGenerator::on_actionSave_triggered() {
409     QString fn = QFileDialog::getSaveFileName(
410         this, tr("Save Report"), QString::fromStdString(
411             Settings->SampleFolder),
```

```

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

```

```
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 }
```



P. Vision Soil Analyzer Program

General project files

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

```
31
32 FORMS     += vsemainwindow.ui \
33     dialogsettings.ui \
34     dialognn.ui
35
36 #opencv
37 LIBS += -L/usr/local/lib -lopencv_core -lopencv_highgui -
38         opencv_imgcodecs
39 INCLUDEPATH += /usr/local/include/opencv
40 INCLUDEPATH += /usr/local/include
41
42 #boost
43 DEFINES += BOOST_ALL_DYN_LINK
44 INCLUDEPATH += /usr/include/boost
45 LIBS += -L/usr/lib/x86_64-linux-gnu/ -lboost_filesystem -
46         lboost_serialization -lboost_system -lboost_iostreams
47
48 #Gstreamer
49 INCLUDEPATH += /usr/include/gstreamer-0.10
50 INCLUDEPATH += /usr/include/glib-2.0/
51 INCLUDEPATH += /usr/lib/x86_64-linux-gnu/glib-2.0/include/
52 INCLUDEPATH += /usr/include/libxml2/
53 LIBS += `pkg-config --cflags --libs gstreamer-0.10`
54
55 #SoilMath lib
56 unix:!macx: LIBS += -L$$PWD/../../build/install/ -lSoilMath
57 INCLUDEPATH += $$PWD/../SoilMath
58 DEPENDPATH += $$PWD/../SoilMath
59
60 #SoilHardware lib
61 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
62         lSoilHardware
63 INCLUDEPATH += $$PWD/../SoilHardware
64 DEPENDPATH += $$PWD/../SoilHardware
65
66 #SoilVision lib
67 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
68         lSoilVision
69 INCLUDEPATH += $$PWD/../SoilVision
70 DEPENDPATH += $$PWD/../SoilVision
71
72 #QCustomplot lib
73 DEFINES += QCUSTOMPLOT_USE_LIBRARY
74 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
75         lqcustomplot
76 INCLUDEPATH += $$PWD/../qcustomplot
77 DEPENDPATH += $$PWD/../qcustomplot
78
79 #QParticleSelector
80 unix:!macx: LIBS += -L$$PWD/../../build/install/ -
81         lQParticleSelector
82 INCLUDEPATH += $$PWD/../QParticleSelector
83 DEPENDPATH += $$PWD/../QParticleSelector
84
85 #QParticleDisplay
```

```

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

```

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

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();
```

```
55     void on_actionAutomatic_Shape_Pediction_triggered(bool
56         checked);
57
58     void on_reset_graph(QMouseEvent * e);
59
60     void on_actionReport_Generator_triggered();
61
62     void on_particleChanged(int newPart);
63
64     void on_PSD_contextMenuRequest(QPoint point);
65
66     void on_compare_against();
67
68     void on_restore_PSD();
69
70     private:
71     Ui::VSAMainWindow *ui;
72     DialogSettings *settingsWindow = nullptr;
73     DialogNN *nnWindow = nullptr;
74     QProgressBar *Progress;
75     QErrorMessage *CamError = nullptr;
76     QMessageBox *SaveMeMessage = nullptr;
77     QMessageBox *BacklightMessage = nullptr;
78     QMessageBox *ShakeItBabyMessage = nullptr;
79     QReportGenerator *ReportGenWindow = nullptr;
80
81     SoilAnalyzer::SoilSettings *Settings = nullptr;
82     Hardware::Microscope *Microscope = nullptr;
83     SoilAnalyzer::Sample *Sample = nullptr;
84     SoilAnalyzer::Analyzer *Analyzer = nullptr;
85     SoilAnalyzer::Analyzer::Images_t *Images = nullptr;
86     QCPBars *RoundnessBars = nullptr;
87     QCPBars *AngularityBars = nullptr;
88     std::vector<double> PSDTicks = {0.0, 0.038, 0.045, 0.063,
89                                     0.075,
90                                     0.09, 0.125, 0.18, 0.25,
91                                     0.355,
92                                     0.5, 0.71, 1.0, 1.4,
93                                     2.0};
94     QVector<QString> RoundnessCat = {"High", "Medium", "Low"};
95     std::vector<double> RoundnessTicks = {1, 2, 3};
96     QVector<QString> AngularityCat = {"Very Angular", "Angular
97                                         ", "Sub Angular",
98                                         "Sub Rounded", "Rounded
99                                         ", "Well Rounded"};
100    std::vector<double> AngularityTicks = {1, 2, 3, 4, 5, 6};
101
102    bool ParticleDisplayerFilled = false;
103
104    void SetPSDgraph();
105    void setRoundnessHistogram();
106    void setAngularityHistogram();
107    void setAmpgraph();
108    void TakeSnapShots();
109
110};
```

```

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

```

```
48         tr("No cams found! Connect the cam and set the  
49             default"));  
50     settingsWindow = new DialogSettings(this, Settings  
51             , Microscope);  
52     }  
53 }  
54  
55 // Setup the sample  
56 Sample = new SoilAnalyzer::Sample;  
57 Images = new SoilAnalyzer::Analyzer::Images_t;  
58 Analyzer = new SoilAnalyzer::Analyzer(Images, Sample,  
59             Settings);  
60  
61 // Setup the setting Window  
62 if (settingsWindow == nullptr) {  
63     settingsWindow =  
64         new DialogSettings(this, Settings, Microscope, &  
65             Analyzer->NeuralNet);  
66 }  
67  
68 // Setup the NN window  
69 if (nnWindow == nullptr) {  
70     nnWindow =  
71         new DialogNN(this, &Analyzer->NeuralNet, Settings,  
72             settingsWindow);  
73 }  
74  
75 // Setup the progressbar and connect it to the Analyzer  
76 Progress = new QProgressBar(ui->statusBar);  
77 Progress->setMaximum(Analyzer->MaxProgress);  
78 Progress->setValue(0);  
79 Progress->setAlignment(Qt::AlignLeft);  
80 Progress->setMinimumSize(750, 19);  
81 ui->statusBar->addWidget(Progress);  
82 connect(Analyzer, SIGNAL(on_progressUpdate(int)), Progress  
83             ,  
84                 SLOT(setValue(int)));  
85 connect(Analyzer, SIGNAL(on_progressUpdate(int)), Progress  
86             ,  
87                 SLOT(setMaximum(int)));  
88 connect(Analyzer, SIGNAL(on_AnalysisFinished()), this,  
89                 SLOT(on_analyzer_finished()));  
90  
91 // Setup the plot linestyles;  
92 QPen pdfPen;  
93 pdfPen.setColor(QColor("gray"));  
94 pdfPen.setStyle(Qt::DashDotDotLine);  
95 pdfPen.setWidthF(1);  
96  
97 QPen meanPen;  
98 meanPen.setColor(QColor("darkBlue"));  
99 meanPen.setStyle(Qt::DashLine);  
100 meanPen.setWidthF(1);  
101  
102 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->
109     Qplot_PSD->yAxis);
110 ui->Qplot_PSD->graph(0)
111     ->setScatterStyle(QCPScatterStyle(QCPScatterStyle:::
112         ssCircle, 8));
113 ui->Qplot_PSD->graph(0)->setPen(binPen);
114 ui->Qplot_PSD->graph(0)->setName("Particle Size
115     Distribution");
116 ui->Qplot_PSD->graph(0)->addToLegend();
117
118 ui->Qplot_PSD->xAxis->setLabel("Particle size [mm]");
119 ui->Qplot_PSD->xAxis->setRange(0.01, 10);
120 ui->Qplot_PSD->xAxis->setAutoTicks(false);
121 ui->Qplot_PSD->xAxis->setTickVector(QVector<double>:::
122         fromStdVector(PSDTicks));
123 ui->Qplot_PSD->xAxis->setTickLabelRotation(30);
124 ui->Qplot_PSD->xAxis->setTickLabelFont(QFont("sans", 8,
125         QFont::Normal));
126 ui->Qplot_PSD->xAxis->setScaleType(QCPAxis::stLogarithmic)
127         ;
128
129 QFont legendfont;
130 legendfont.setPointSize(10);
131 ui->Qplot_PSD->legend->setFont(legendfont);
132 ui->Qplot_PSD->legend->setSelectedFont(legendfont);
133 ui->Qplot_PSD->legend->setVisible(true);
134 ui->Qplot_PSD->axisRect()->insetLayout()->
135         setInsetAlignment(
136             0, Qt::AlignTop | Qt::AlignLeft);
137
138 ui->Qplot_PSD->yAxis->setLabel("Percentage [%]");
139 ui->Qplot_PSD->yAxis->setRange(0, 100);
140 ui->Qplot_PSD->setInteractions(QCP::iRangeDrag | QCP::
141         iRangeZoom);
142 ui->Qplot_PSD->yAxis->grid()->setSubGridVisible(true);
143
144 connect(ui->Qplot_PSD, SIGNAL(mouseDoubleClick(QMouseEvent
145         *)), this,
146         SLOT(on_reset_graph(QMouseEvent *)));
147 ui->Qplot_PSD->setContextMenuPolicy(Qt::CustomContextMenu)
148         ;
149 connect(ui->Qplot_PSD, SIGNAL(customContextMenuRequested(
150         QPoint)), this,
151         SLOT(on_PSD_contextMenuRequest(QPoint)));

```

```

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

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

```

472     }
473 }
474
475 void VSAMainWindow::on_actionLoadSample_triggered() {
476     if (Sample->ChangesSinceLastSave) {
477         if (SaveMeMessage->exec() == QMessageBox::Abort) {
478             return;
479         }
480     }
481
482     QString fn = QFileDialog::getOpenFileName(
483         this, tr("Open Sample"), QString::fromStdString(
484             Settings->SampleFolder),
485         tr("Sample (*.VSA)"));
486     if (!fn.isEmpty()) {
487         if (!fn.contains(tr(".VSA")))
488             fn.append(tr(".VSA"));
489
490         delete Sample;
491         Sample = nullptr;
492         delete Images;
493         Images = nullptr;
494         Sample = new SoilAnalyzer::Sample;
495         Images = new SoilAnalyzer::Analyzer::Images_t;
496         try {
497             Sample->Load(fn.toStdString());
498         } catch (boost::archive::archive_exception &e) {
499             // qDebug() << *e.what();
500         }
501         ParticleDisplayerFilled = false;
502         Sample->Angularity.Data = Sample->GetAngularityVector()
503             ->data();
504         Sample->Roundness.Data = Sample->GetRoundnessVector()->
505             data();
506         Sample->PSD.Data = Sample->GetPSDVector()->data();
507         Analyzer->Results = Sample;
508         on_analyzer_finished();
509         ui->widget_ParticleSelector->setDisabled(
510             false,
511             ui->widget_ParticleDisplay->SelectedParticle->
512                 Classification.Category);
513     }
514 }
515
516 void VSAMainWindow::on_actionUseLearning_toggled(bool arg1)
517 {
518     Analyzer->PredictShape = !arg1;
519 }
520
521 void VSAMainWindow::on_actionCalibrate_triggered() {
522     cv::Mat calib;
523     Microscope->GetFrame(calib);
524     Analyzer->CalibrateSI(16.25, calib);
525 }
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727

```

```

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

```

```

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

```

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

Dialog window Class

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

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

```

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

```

```

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

```

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



```
93 // Setup the Vision tab
94 ui->cb_fillHoles_3->setChecked(Settings->fillHoles);
95 ui->cb_ignoreBorder_3->setChecked(Settings->
96     ignorePartialBorderParticles);
97 ui->cb_useBlur_3->setChecked(Settings->useBlur);
98 if (!Settings->useBlur) {
99     ui->sb_blurMask_3->setEnabled(false);
100 }
101 ui->cb_use_adaptContrast_3->setChecked(Settings->
102     useAdaptiveContrast);
103 if (!Settings->useAdaptiveContrast) {
104     ui->sb_adaptContrastFactor_3->setEnabled(false);
105     ui->sb_adaptContrKernel_3->setEnabled(false);
106 }
107 switch (Settings->typeOfObjectsSegmented) {
108     case Vision::Segment::Bright:
109         ui->rb_useDark_3->setChecked(false);
110         ui->rb_useLight_3->setChecked(true);
111         break;
112     case Vision::Segment::Dark:
113         ui->rb_useDark_3->setChecked(true);
114         ui->rb_useLight_3->setChecked(false);
115         break;
116     switch (Settings->morphFilterType) {
117         case Vision::MorphologicalFilter::CLOSE:
118             ui->rb_useClose_3->setChecked(true);
119             ui->rb_useDilate_3->setChecked(false);
120             ui->rb_useErode_3->setChecked(false);
121             ui->rb_useOpen_3->setChecked(false);
122             break;
123         case Vision::MorphologicalFilter::OPEN:
124             ui->rb_useClose_3->setChecked(false);
125             ui->rb_useDilate_3->setChecked(false);
126             ui->rb_useErode_3->setChecked(false);
127             ui->rb_useOpen_3->setChecked(true);
128             break;
129         case Vision::MorphologicalFilter::ERODE:
130             ui->rb_useClose_3->setChecked(false);
131             ui->rb_useDilate_3->setChecked(false);
132             ui->rb_useErode_3->setChecked(true);
133             ui->rb_useOpen_3->setChecked(false);
134             break;
135         case Vision::MorphologicalFilter::DILATE:
136             ui->rb_useClose_3->setChecked(false);
137             ui->rb_useDilate_3->setChecked(true);
138             ui->rb_useErode_3->setChecked(false);
139             ui->rb_useOpen_3->setChecked(false);
140             break;
141     }
142     ui->sb_adaptContrastFactor_3->setValue(Settings->
143         adaptContrastKernelFactor);
144     ui->sb_adaptContrKernel_3->setValue(Settings->
145         adaptContrastKernelSize);
146     ui->sb_blurMask_3->setValue(Settings->blurKernelSize);
```

```

145     ui->sb_morphMask_3->setValue(Settings->filterMaskSize);
146     ui->sb_sigmaFactor_3->setValue(Settings->sigmaFactor);
147
148     // Setup the neural Network tab
149     NN = nn;
150     QPixmap NNpix("Images/feedforwardnetwork2.png");
151     ui->label_NNimage->setPixmap(NNpix);
152     ui->label_NNimage->setScaledContents(true);
153
154     ui->spinBox_InputNeurons->setValue(NN->GetInputNeurons());
155     ui->spinBox_HiddenNeurons->setValue(NN->GetHiddenNeurons()
156         );
156     ui->spinBox_OutputNeurons->setValue(NN->GetOutputNeurons()
157         );
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         ;
161     ui->doubleSpinBox_MutationRate->setValue(NN->
162         MutationrateUsedByGA);
162     ui->doubleSpinBox_Beta->setValue(NN->GetBeta());
163     ui->doubleSpinBox_maxWeight->setValue(NN->
164         MaxWeightUsedByGA);
164     ui->doubleSpinBox_MinWeight->setValue(NN->
165         MinWeightUsedByGA);
165     ui->checkBox_PredictShape->setChecked(Settings->
166         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 {
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(
203             "Settings (*.ini)"));
204     if (!fn.isEmpty()) {
205         if (!fn.contains(tr(".ini"))) {
206             fn.append(tr(".ini"));
207         }
208         Settings->LoadSettings(fn.toStdString());
209     }
210 }
211
212 void DialogSettings::on_pushButton_Save_clicked() {
213     QString fn = QFileDialog::getSaveFileName(
214         this, tr("Save Settings"), QDir::homePath(), tr(
215             "Settings (*.ini)"));
216     if (!fn.isEmpty()) {
217         if (!fn.contains(tr(".ini"))) {
218             fn.append(tr(".ini"));
219         }
220     }
221 }
222 void DialogSettings::on_checkBox_Backlight_clicked(bool
223     checked) {
224     ui->tabWidget_Hardware->setTabEnabled(2, checked);
225     Settings->useBacklightProjection = checked;
226 }
227
228 void DialogSettings::
229     on_comboBox_Microscopes_currentIndexChanged(
230         const QString &arg1) {
231
232     if (!initfase) {
233         std::string selectedCam = arg1.toStdString();
234         Microscope->openCam(selectedCam);
235         Settings->defaultWebcam = selectedCam;
236
237         ui->comboBox_Resolution->clear();
238         QStringList Resolutions;
239         for (uint32_t i = 0; i < Microscope->SelectedCam->
240             Resolutions.size(); i++) {
241             Resolutions
242                 << Microscope->SelectedCam->Resolutions[i].
243                     to_string().c_str();
244         }
245         ui->comboBox_Resolution->addItems(Resolutions);
246         ui->comboBox_Resolution->setcurrentIndex(
```

```

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

```

```
281     Sharpness->setMaximum(selectedCam->Controls[i].maximum
282     );
283 }
284 }
285
286 void DialogSettings::on_spinBox_NoFrames_editingFinished() {
287     Settings->HDRframes = ui->spinBox_NoFrames->value();
288 }
289
290 void DialogSettings::
291     on_doubleSpinBox_LightLevel_editingFinished() {
292     Settings->lightLevel =
293         static_cast<float>(ui->doubleSpinBox_LightLevel->value
294         ());
295 }
296
297 void DialogSettings::on_checkBox_useRainbow_clicked(bool
298     checked) {
299     Settings->enableRainbow = checked;
300 }
301
302 void DialogSettings::on_checkBox_InvertEncoder_clicked(bool
303     checked) {
304     Settings->encInv = checked;
305 }
306
307 void DialogSettings::
308     on_horizontalSlider_BrightFront_valueChanged(int value) {
309     if (!initfase) {
310         Settings->Brightness_front = value;
311     }
312 }
313
314 void DialogSettings::
315     on_horizontalSlider_ContrastFront_valueChanged(int value)
316     {
317     if (!initfase) {
318         Settings->Contrast_front = value;
319     }
320 }
321
322 void DialogSettings::
323     on_horizontalSlider_SaturationFront_valueChanged(
324     int value) {
325     if (!initfase) {
326         Settings->Saturation_front = value;
327     }
328 }
```

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

```
372     ui->sb_adaptContrastFactor_3->setDisabled(!checked);
373     ui->sb_adaptContrKernel_3->setDisabled(!checked);
374 }
375
376 void DialogSettings::on_cb_useBlur_3_clicked(bool checked) {
377     Settings->useBlur = checked;
378     ui->sb_blurMask_3->setDisabled(!checked);
379 }
380
381 void DialogSettings::on_rb_useDark_3_toggled(bool checked) {
382     if (checked) {
383         Settings->typeOfObjectsSegmented = Vision::Segment::Dark
384             ;
385     } else {
386         Settings->typeOfObjectsSegmented = Vision::Segment::
387             Bright;
388     }
389 }
390
391 void DialogSettings::on_cb_ignoreBorder_3_clicked(bool
392     checked) {
393     Settings->ignorePartialBorderParticles = checked;
394 }
395
396
397 void DialogSettings::on_sb_sigmaFactor_3_editingFinished() {
398     Settings->sigmaFactor = ui->sb_sigmaFactor_3->value();
399 }
400
401 void DialogSettings::on_rb_useOpen_3_clicked(bool checked) {
402     Settings->morphFilterType = Vision::MorphologicalFilter::
403         OPEN;
404 }
405
406 void DialogSettings::on_rb_useClose_3_clicked(bool checked)
407 {
408     Settings->morphFilterType = Vision::MorphologicalFilter::
409         CLOSE;
410 }
411
412
413 void DialogSettings::on_rb_useErode_3_clicked(bool checked)
414 {
415     Settings->morphFilterType = Vision::MorphologicalFilter::
416         ERODE;
417 }
418
419 void DialogSettings::on_rb_useDilate_3_clicked(bool checked)
420 {
421     Settings->morphFilterType = Vision::MorphologicalFilter::
422         DILATE;
423 }
424
```

```
417 void DialogSettings::on_sb_morphMask_3_editingFinished() {
418     Settings->filterMaskSize = ui->sb_morphMask_3->value();
419 }
420
421 void DialogSettings::on_spinBox_MaxGen_editingFinished() {
422     NN->MaxGenUsedByGA = ui->spinBox_MaxGen->value();
423 }
424
425 void DialogSettings::on_spinBox_PopSize_editingFinished() {
426     NN->PopulationSizeUsedByGA = ui->spinBox_PopSize->value();
427 }
428
429 void DialogSettings::
430     on_doubleSpinBox_MutationRate_editingFinished() {
431     NN->MutationrateUsedByGA = ui->doubleSpinBox_MutationRate
432         ->value();
433 }
434
435 void DialogSettings::on_spinBox_Elitisme_editingFinished() {
436     NN->ElitismeUsedByGA = ui->spinBox_Elitisme->value();
437 }
438
439 void DialogSettings::
440     on_doubleSpinBox_endError_editingFinished() {
441     NN->EndErrorUsedByGA = ui->doubleSpinBox_endError->value()
442         ;
443 }
444
445 void DialogSettings::
446     on_doubleSpinBox_maxWeight_editingFinished() {
447     NN->MaxWeightUsedByGA = ui->doubleSpinBox_maxWeight->value()
448         ;
449 }
450
451 void DialogSettings::
452     on_doubleSpinBox_MinWeight_editingFinished() {
453     NN->MinWeightUsedByGA = ui->doubleSpinBox_MinWeight->value()
454         ;
455 }
456
457 void DialogSettings::
458     on_spinBox_InputNeurons_editingFinished() {
459     NN->SetInputNeurons(ui->spinBox_InputNeurons->value());
460 }
```

```
461 void DialogSettings::  
462     on_spinBox_OutputNeurons_editingFinished() {  
463     NN->SetOutputNeurons(ui->spinBox_OutputNeurons->value());  
464 }  
465 void DialogSettings::  
466     on_pushButton_selectSampleFolder_clicked() {  
467     QString fn = QFileDialog::getExistingDirectory(  
468         this, tr("Select the Sample Directory"),  
469         QString::fromStdString(Settings->SampleFolder),  
470         QFileDialog::ShowDirsOnly | QFileDialog::  
471             DontResolveSymlinks);  
472     if (!fn.isEmpty()) {  
473         ui->lineEdit_Samplefolder->setText(fn);  
474         Settings->SampleFolder = fn.toStdString();  
475     }  
476 }  
477 void DialogSettings::  
478     on_pushButton_SelectSettingFolder_clicked() {  
479     QString fn = QFileDialog::getExistingDirectory(  
480         this, tr("Select the Setting Directory"),  
481         QString::fromStdString(Settings->SettingsFolder),  
482         QFileDialog::ShowDirsOnly | QFileDialog::  
483             DontResolveSymlinks);  
484     if (!fn.isEmpty()) {  
485         ui->lineEdit_SettingFolder->setText(fn);  
486         Settings->SettingsFolder = fn.toStdString();  
487     }  
488 }  
489 void DialogSettings::on_pushButton_SelectNNFolder_clicked()  
490 {  
491     QString fn = QFileDialog::getExistingDirectory(  
492         this, tr("Select the NeuralNet Directory"),  
493         QString::fromStdString(Settings->NNFolder),  
494         QFileDialog::ShowDirsOnly | QFileDialog::  
495             DontResolveSymlinks);  
496     if (!fn.isEmpty()) {  
497         ui->lineEdit_NeuralNetFolder->setText(fn);  
498         Settings->NNFolder = fn.toStdString();  
499     }  
500 }  
501 void DialogSettings::on_pushButton_SelectNN_clicked() {  
502     QString fn =  
503         QFileDialog::getOpenFileName(this, tr("Select the  
504             standard Neural Net"),  
505                                         QDir::homePath(), tr("NeuralNet (*.NN)"));  
506     if (!fn.isEmpty()) {  
507         if (!fn.contains(tr(".NN"))) {  
508             fn.append(tr(".NN"));  
509         }  
510         Settings->NNlocation = fn.toStdString();  
511         ui->lineEdit__NeuralNet->setText(fn);  
512     }  
513 }
```

```
508     }
509 }
510
511 void DialogSettings::on_spinBox_NoShots_editingFinished() {
512     Settings->StandardNumberOfShots = ui->spinBox_NoShots->
513         value();
514
515 void DialogSettings::on_checkBox_PredictShape_clicked(bool
516     checked) {
517     Settings->PredictTheShape = checked;
518 }
519 void DialogSettings::on_checkBox_revolt_clicked(bool checked
520     )
521 {
522     Settings->Revolution = checked;
523 }
```

Dialog Neural Network Class

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

```

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


---


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

```

```

49
50 void DialogNN::on_learnErrorUpdate(double newError) {
51     ui->widget_NNError->graph(0)->addData(currentGeneration,
52         newError);
53     currentGeneration += 1;
54     ui->widget_NNError->yAxis->rescale();
55     //ui->widget_NNError->yAxis->setRange(0, 20);
56     ui->widget_NNError->replot();
57 }
58
59 void DialogNN::setupErrorGraph() {
60     errorTicks.clear();
61     for (uint32_t i = 0; i < NeuralNet->MaxGenUsedByGA; i++) {
62         errorTicks.push_back(i);
63     }
64     ui->widget_NNError->xAxis->setRange(0, NeuralNet->
65         MaxGenUsedByGA);
66     QVector<double> endErrorValue(2, NeuralNet->
67         EndErrorUsedByGA);
68     QVector<double> endErrorKey(2, 0);
69     endErrorKey[1] = NeuralNet->MaxGenUsedByGA;
70     ui->widget_NNError->graph(1)->setData(endErrorKey,
71         endErrorValue);
72     ui->widget_NNError->xAxis->setAutoTicks(false);
73     ui->widget_NNError->xAxis->setTickVector(errorTicks);
74     ui->widget_NNError->xAxis->setTickLabels(false);
75     //ui->widget_NNError->yAxis->setScaleType(QCPAxis::
76         stLogarithmic);
77     ui->widget_NNError->replot();
78 }
79
80 void DialogNN::on_pushButton_SelectSamples_clicked() {
81     fn = QFileDialog::getOpenFileNames(
82         this, tr("Open Samples"), QString::fromStdString(
83             Settings->SampleFolder),
84         tr("Samples (*.VSA)"));
85     for_each(fn.begin(), fn.end(), [](QString &f) {
86         if (!f.contains(tr(".VSA")))) {
87             f.append(tr(".VSA"));
88         }
89     });
90 }
91
92 void DialogNN::on_pushButton_Learn_clicked() {
93     if (fn.size() < 1) {
94         return;
95     }
96     InputLearnVector_t InputVec;
97     OutputLearnVector_t OutputVec;
98     makeLearnVectors(InputVec, OutputVec);
99     NeuralNet->Learn(InputVec, OutputVec, NeuralNet->
100         GetInputNeurons());
101     setupErrorGraph();
102 }
103
104 void DialogNN::makeLearnVectors(InputLearnVector_t &input,
105     OutputLearnVector_t &output) {
106     for (int i = 0; i < fn.size(); i++) {
107         QString file = fn[i];
108         InputVec = readInputVector(file);
109         OutputVec = readOutputVector(file);
110         NeuralNet->addData(InputVec, OutputVec);
111     }
112 }
113
114 void DialogNN::on_pushButton_Load_clicked() {
115     if (fn.size() < 1) {
116         return;
117     }
118     InputLearnVector_t InputVec;
119     OutputLearnVector_t OutputVec;
120     makeLearnVectors(InputVec, OutputVec);
121     NeuralNet->Load(InputVec, OutputVec, NeuralNet->
122         GetInputNeurons());
123 }
124
125 void DialogNN::on_pushButton_Save_clicked() {
126     if (fn.size() < 1) {
127         return;
128     }
129     QString file = QFileDialog::getSaveFileName(
130         this, tr("Save Model"), QString::fromStdString(
131             Settings->ModelFolder),
132         tr("Model (*.VSA)"));
133     if (!file.isEmpty()) {
134         NeuralNet->Save(file);
135     }
136 }
137
138 void DialogNN::on_pushButton_Exit_clicked() {
139     QApplication::quit();
140 }
141
142 void DialogNN::on_pushButton_Import_clicked() {
143     if (fn.size() < 1) {
144         return;
145     }
146     InputLearnVector_t InputVec;
147     OutputLearnVector_t OutputVec;
148     makeLearnVectors(InputVec, OutputVec);
149     NeuralNet->Import(InputVec, OutputVec, NeuralNet->
150         GetInputNeurons());
151 }
152
153 void DialogNN::on_pushButton_Export_clicked() {
154     if (fn.size() < 1) {
155         return;
156     }
157     InputLearnVector_t InputVec;
158     OutputLearnVector_t OutputVec;
159     makeLearnVectors(InputVec, OutputVec);
160     NeuralNet->Export(InputVec, OutputVec, NeuralNet->
161         GetInputNeurons());
162 }
163
164 void DialogNN::on_pushButton_Statistics_clicked() {
165     if (fn.size() < 1) {
166         return;
167     }
168     InputLearnVector_t InputVec;
169     OutputLearnVector_t OutputVec;
170     makeLearnVectors(InputVec, OutputVec);
171     NeuralNet->Statistics(InputVec, OutputVec, NeuralNet->
172         GetInputNeurons());
173 }
174
175 void DialogNN::on_pushButton_Plot_clicked() {
176     if (fn.size() < 1) {
177         return;
178     }
179     InputLearnVector_t InputVec;
180     OutputLearnVector_t OutputVec;
181     makeLearnVectors(InputVec, OutputVec);
182     NeuralNet->Plot(InputVec, OutputVec, NeuralNet->
183         GetInputNeurons());
184 }
185
186 void DialogNN::on_pushButton_Exit2_clicked() {
187     QApplication::quit();
188 }
189
190 void DialogNN::on_pushButton_Import2_clicked() {
191     if (fn.size() < 1) {
192         return;
193     }
194     InputLearnVector_t InputVec;
195     OutputLearnVector_t OutputVec;
196     makeLearnVectors(InputVec, OutputVec);
197     NeuralNet->Import2(InputVec, OutputVec, NeuralNet->
198         GetInputNeurons());
199 }
200
201 void DialogNN::on_pushButton_Export2_clicked() {
202     if (fn.size() < 1) {
203         return;
204     }
205     InputLearnVector_t InputVec;
206     OutputLearnVector_t OutputVec;
207     makeLearnVectors(InputVec, OutputVec);
208     NeuralNet->Export2(InputVec, OutputVec, NeuralNet->
209         GetInputNeurons());
210 }
211
212 void DialogNN::on_pushButton_Statistics2_clicked() {
213     if (fn.size() < 1) {
214         return;
215     }
216     InputLearnVector_t InputVec;
217     OutputLearnVector_t OutputVec;
218     makeLearnVectors(InputVec, OutputVec);
219     NeuralNet->Statistics2(InputVec, OutputVec, NeuralNet->
220         GetInputNeurons());
221 }
222
223 void DialogNN::on_pushButton_Plot2_clicked() {
224     if (fn.size() < 1) {
225         return;
226     }
227     InputLearnVector_t InputVec;
228     OutputLearnVector_t OutputVec;
229     makeLearnVectors(InputVec, OutputVec);
230     NeuralNet->Plot2(InputVec, OutputVec, NeuralNet->
231         GetInputNeurons());
232 }
233
234 void DialogNN::on_pushButton_Exit3_clicked() {
235     QApplication::quit();
236 }
237
238 void DialogNN::on_pushButton_Import3_clicked() {
239     if (fn.size() < 1) {
240         return;
241     }
242     InputLearnVector_t InputVec;
243     OutputLearnVector_t OutputVec;
244     makeLearnVectors(InputVec, OutputVec);
245     NeuralNet->Import3(InputVec, OutputVec, NeuralNet->
246         GetInputNeurons());
247 }
248
249 void DialogNN::on_pushButton_Export3_clicked() {
250     if (fn.size() < 1) {
251         return;
252     }
253     InputLearnVector_t InputVec;
254     OutputLearnVector_t OutputVec;
255     makeLearnVectors(InputVec, OutputVec);
256     NeuralNet->Export3(InputVec, OutputVec, NeuralNet->
257         GetInputNeurons());
258 }
259
260 void DialogNN::on_pushButton_Statistics3_clicked() {
261     if (fn.size() < 1) {
262         return;
263     }
264     InputLearnVector_t InputVec;
265     OutputLearnVector_t OutputVec;
266     makeLearnVectors(InputVec, OutputVec);
267     NeuralNet->Statistics3(InputVec, OutputVec, NeuralNet->
268         GetInputNeurons());
269 }
270
271 void DialogNN::on_pushButton_Plot3_clicked() {
272     if (fn.size() < 1) {
273         return;
274     }
275     InputLearnVector_t InputVec;
276     OutputLearnVector_t OutputVec;
277     makeLearnVectors(InputVec, OutputVec);
278     NeuralNet->Plot3(InputVec, OutputVec, NeuralNet->
279         GetInputNeurons());
280 }
281
282 void DialogNN::on_pushButton_Exit4_clicked() {
283     QApplication::quit();
284 }
285
286 void DialogNN::on_pushButton_Import4_clicked() {
287     if (fn.size() < 1) {
288         return;
289     }
290     InputLearnVector_t InputVec;
291     OutputLearnVector_t OutputVec;
292     makeLearnVectors(InputVec, OutputVec);
293     NeuralNet->Import4(InputVec, OutputVec, NeuralNet->
294         GetInputNeurons());
295 }
296
297 void DialogNN::on_pushButton_Export4_clicked() {
298     if (fn.size() < 1) {
299         return;
300     }
301     InputLearnVector_t InputVec;
302     OutputLearnVector_t OutputVec;
303     makeLearnVectors(InputVec, OutputVec);
304     NeuralNet->Export4(InputVec, OutputVec, NeuralNet->
305         GetInputNeurons());
306 }
307
308 void DialogNN::on_pushButton_Statistics4_clicked() {
309     if (fn.size() < 1) {
310         return;
311     }
312     InputLearnVector_t InputVec;
313     OutputLearnVector_t OutputVec;
314     makeLearnVectors(InputVec, OutputVec);
315     NeuralNet->Statistics4(InputVec, OutputVec, NeuralNet->
316         GetInputNeurons());
317 }
318
319 void DialogNN::on_pushButton_Plot4_clicked() {
320     if (fn.size() < 1) {
321         return;
322     }
323     InputLearnVector_t InputVec;
324     OutputLearnVector_t OutputVec;
325     makeLearnVectors(InputVec, OutputVec);
326     NeuralNet->Plot4(InputVec, OutputVec, NeuralNet->
327         GetInputNeurons());
328 }
329
330 void DialogNN::on_pushButton_Exit5_clicked() {
331     QApplication::quit();
332 }
333
334 void DialogNN::on_pushButton_Import5_clicked() {
335     if (fn.size() < 1) {
336         return;
337     }
338     InputLearnVector_t InputVec;
339     OutputLearnVector_t OutputVec;
340     makeLearnVectors(InputVec, OutputVec);
341     NeuralNet->Import5(InputVec, OutputVec, NeuralNet->
342         GetInputNeurons());
343 }
344
345 void DialogNN::on_pushButton_Export5_clicked() {
346     if (fn.size() < 1) {
347         return;
348     }
349     InputLearnVector_t InputVec;
350     OutputLearnVector_t OutputVec;
351     makeLearnVectors(InputVec, OutputVec);
352     NeuralNet->Export5(InputVec, OutputVec, NeuralNet->
353         GetInputNeurons());
354 }
355
356 void DialogNN::on_pushButton_Statistics5_clicked() {
357     if (fn.size() < 1) {
358         return;
359     }
360     InputLearnVector_t InputVec;
361     OutputLearnVector_t OutputVec;
362     makeLearnVectors(InputVec, OutputVec);
363     NeuralNet->Statistics5(InputVec, OutputVec, NeuralNet->
364         GetInputNeurons());
365 }
366
367 void DialogNN::on_pushButton_Plot5_clicked() {
368     if (fn.size() < 1) {
369         return;
370     }
371     InputLearnVector_t InputVec;
372     OutputLearnVector_t OutputVec;
373     makeLearnVectors(InputVec, OutputVec);
374     NeuralNet->Plot5(InputVec, OutputVec, NeuralNet->
375         GetInputNeurons());
376 }
377
378 void DialogNN::on_pushButton_Exit6_clicked() {
379     QApplication::quit();
380 }
381
382 void DialogNN::on_pushButton_Import6_clicked() {
383     if (fn.size() < 1) {
384         return;
385     }
386     InputLearnVector_t InputVec;
387     OutputLearnVector_t OutputVec;
388     makeLearnVectors(InputVec, OutputVec);
389     NeuralNet->Import6(InputVec, OutputVec, NeuralNet->
390         GetInputNeurons());
391 }
392
393 void DialogNN::on_pushButton_Export6_clicked() {
394     if (fn.size() < 1) {
395         return;
396     }
397     InputLearnVector_t InputVec;
398     OutputLearnVector_t OutputVec;
399     makeLearnVectors(InputVec, OutputVec);
400     NeuralNet->Export6(InputVec, OutputVec, NeuralNet->
401         GetInputNeurons());
402 }
403
404 void DialogNN::on_pushButton_Statistics6_clicked() {
405     if (fn.size() < 1) {
406         return;
407     }
408     InputLearnVector_t InputVec;
409     OutputLearnVector_t OutputVec;
410     makeLearnVectors(InputVec, OutputVec);
411     NeuralNet->Statistics6(InputVec, OutputVec, NeuralNet->
412         GetInputNeurons());
413 }
414
415 void DialogNN::on_pushButton_Plot6_clicked() {
416     if (fn.size() < 1) {
417         return;
418     }
419     InputLearnVector_t InputVec;
420     OutputLearnVector_t OutputVec;
421     makeLearnVectors(InputVec, OutputVec);
422     NeuralNet->Plot6(InputVec, OutputVec, NeuralNet->
423         GetInputNeurons());
424 }
425
426 void DialogNN::on_pushButton_Exit7_clicked() {
427     QApplication::quit();
428 }
429
430 void DialogNN::on_pushButton_Import7_clicked() {
431     if (fn.size() < 1) {
432         return;
433     }
434     InputLearnVector_t InputVec;
435     OutputLearnVector_t OutputVec;
436     makeLearnVectors(InputVec, OutputVec);
437     NeuralNet->Import7(InputVec, OutputVec, NeuralNet->
438         GetInputNeurons());
439 }
440
441 void DialogNN::on_pushButton_Export7_clicked() {
442     if (fn.size() < 1) {
443         return;
444     }
445     InputLearnVector_t InputVec;
446     OutputLearnVector_t OutputVec;
447     makeLearnVectors(InputVec, OutputVec);
448     NeuralNet->Export7(InputVec, OutputVec, NeuralNet->
449         GetInputNeurons());
450 }
451
452 void DialogNN::on_pushButton_Statistics7_clicked() {
453     if (fn.size() < 1) {
454         return;
455     }
456     InputLearnVector_t InputVec;
457     OutputLearnVector_t OutputVec;
458     makeLearnVectors(InputVec, OutputVec);
459     NeuralNet->Statistics7(InputVec, OutputVec, NeuralNet->
460         GetInputNeurons());
461 }
462
463 void DialogNN::on_pushButton_Plot7_clicked() {
464     if (fn.size() < 1) {
465         return;
466     }
467     InputLearnVector_t InputVec;
468     OutputLearnVector_t OutputVec;
469     makeLearnVectors(InputVec, OutputVec);
470     NeuralNet->Plot7(InputVec, OutputVec, NeuralNet->
471         GetInputNeurons());
472 }
473
474 void DialogNN::on_pushButton_Exit8_clicked() {
475     QApplication::quit();
476 }
477
478 void DialogNN::on_pushButton_Import8_clicked() {
479     if (fn.size() < 1) {
480         return;
481     }
482     InputLearnVector_t InputVec;
483     OutputLearnVector_t OutputVec;
484     makeLearnVectors(InputVec, OutputVec);
485     NeuralNet->Import8(InputVec, OutputVec, NeuralNet->
486         GetInputNeurons());
487 }
488
489 void DialogNN::on_pushButton_Export8_clicked() {
490     if (fn.size() < 1) {
491         return;
492     }
493     InputLearnVector_t InputVec;
494     OutputLearnVector_t OutputVec;
495     makeLearnVectors(InputVec, OutputVec);
496     NeuralNet->Export8(InputVec, OutputVec, NeuralNet->
497         GetInputNeurons());
498 }
499
500 void DialogNN::on_pushButton_Statistics8_clicked() {
501     if (fn.size() < 1) {
502         return;
503     }
504     InputLearnVector_t InputVec;
505     OutputLearnVector_t OutputVec;
506     makeLearnVectors(InputVec, OutputVec);
507     NeuralNet->Statistics8(InputVec, OutputVec, NeuralNet->
508         GetInputNeurons());
509 }
510
511 void DialogNN::on_pushButton_Plot8_clicked() {
512     if (fn.size() < 1) {
513         return;
514     }
515     InputLearnVector_t InputVec;
516     OutputLearnVector_t OutputVec;
517     makeLearnVectors(InputVec, OutputVec);
518     NeuralNet->Plot8(InputVec, OutputVec, NeuralNet->
519         GetInputNeurons());
520 }
521
522 void DialogNN::on_pushButton_Exit9_clicked() {
523     QApplication::quit();
524 }
525
526 void DialogNN::on_pushButton_Import9_clicked() {
527     if (fn.size() < 1) {
528         return;
529     }
530     InputLearnVector_t InputVec;
531     OutputLearnVector_t OutputVec;
532     makeLearnVectors(InputVec, OutputVec);
533     NeuralNet->Import9(InputVec, OutputVec, NeuralNet->
534         GetInputNeurons());
535 }
536
537 void DialogNN::on_pushButton_Export9_clicked() {
538     if (fn.size() < 1) {
539         return;
540     }
541     InputLearnVector_t InputVec;
542     OutputLearnVector_t OutputVec;
543     makeLearnVectors(InputVec, OutputVec);
544     NeuralNet->Export9(InputVec, OutputVec, NeuralNet->
545         GetInputNeurons());
546 }
547
548 void DialogNN::on_pushButton_Statistics9_clicked() {
549     if (fn.size() < 1) {
550         return;
551     }
552     InputLearnVector_t InputVec;
553     OutputLearnVector_t OutputVec;
554     makeLearnVectors(InputVec, OutputVec);
555     NeuralNet->Statistics9(InputVec, OutputVec, NeuralNet->
556         GetInputNeurons());
557 }
558
559 void DialogNN::on_pushButton_Plot9_clicked() {
560     if (fn.size() < 1) {
561         return;
562     }
563     InputLearnVector_t InputVec;
564     OutputLearnVector_t OutputVec;
565     makeLearnVectors(InputVec, OutputVec);
566     NeuralNet->Plot9(InputVec, OutputVec, NeuralNet->
567         GetInputNeurons());
568 }
569
570 void DialogNN::on_pushButton_Exit10_clicked() {
571     QApplication::quit();
572 }
573
574 void DialogNN::on_pushButton_Import10_clicked() {
575     if (fn.size() < 1) {
576         return;
577     }
578     InputLearnVector_t InputVec;
579     OutputLearnVector_t OutputVec;
580     makeLearnVectors(InputVec, OutputVec);
581     NeuralNet->Import10(InputVec, OutputVec, NeuralNet->
582         GetInputNeurons());
583 }
584
585 void DialogNN::on_pushButton_Export10_clicked() {
586     if (fn.size() < 1) {
587         return;
588     }
589     InputLearnVector_t InputVec;
590     OutputLearnVector_t OutputVec;
591     makeLearnVectors(InputVec, OutputVec);
592     NeuralNet->Export10(InputVec, OutputVec, NeuralNet->
593         GetInputNeurons());
594 }
595
596 void DialogNN::on_pushButton_Statistics10_clicked() {
597     if (fn.size() < 1) {
598         return;
599     }
600     InputLearnVector_t InputVec;
601     OutputLearnVector_t OutputVec;
602     makeLearnVectors(InputVec, OutputVec);
603     NeuralNet->Statistics10(InputVec, OutputVec, NeuralNet->
604         GetInputNeurons());
605 }
606
607 void DialogNN::on_pushButton_Plot10_clicked() {
608     if (fn.size() < 1) {
609         return;
610     }
611     InputLearnVector_t InputVec;
612     OutputLearnVector_t OutputVec;
613     makeLearnVectors(InputVec, OutputVec);
614     NeuralNet->Plot10(InputVec, OutputVec, NeuralNet->
615         GetInputNeurons());
616 }
617
618 void DialogNN::on_pushButton_Exit11_clicked() {
619     QApplication::quit();
620 }
621
622 void DialogNN::on_pushButton_Import11_clicked() {
623     if (fn.size() < 1) {
624         return;
625     }
626     InputLearnVector_t InputVec;
627     OutputLearnVector_t OutputVec;
628     makeLearnVectors(InputVec, OutputVec);
629     NeuralNet->Import11(InputVec, OutputVec, NeuralNet->
630         GetInputNeurons());
631 }
632
633 void DialogNN::on_pushButton_Export11_clicked() {
634     if (fn.size() < 1) {
635         return;
636     }
637     InputLearnVector_t InputVec;
638     OutputLearnVector_t OutputVec;
639     makeLearnVectors(InputVec, OutputVec);
640     NeuralNet->Export11(InputVec, OutputVec, NeuralNet->
641         GetInputNeurons());
642 }
643
644 void DialogNN::on_pushButton_Statistics11_clicked() {
645     if (fn.size() < 1) {
646         return;
647     }
648     InputLearnVector_t InputVec;
649     OutputLearnVector_t OutputVec;
650     makeLearnVectors(InputVec, OutputVec);
651     NeuralNet->Statistics11(InputVec, OutputVec, NeuralNet->
652         GetInputNeurons());
653 }
654
655 void DialogNN::on_pushButton_Plot11_clicked() {
656     if (fn.size() < 1) {
657         return;
658     }
659     InputLearnVector_t InputVec;
660     OutputLearnVector_t OutputVec;
661     makeLearnVectors(InputVec, OutputVec);
662     NeuralNet->Plot11(InputVec, OutputVec, NeuralNet->
663         GetInputNeurons());
664 }
665
666 void DialogNN::on_pushButton_Exit12_clicked() {
667     QApplication::quit();
668 }
669
670 void DialogNN::on_pushButton_Import12_clicked() {
671     if (fn.size() < 1) {
672         return;
673     }
674     InputLearnVector_t InputVec;
675     OutputLearnVector_t OutputVec;
676     makeLearnVectors(InputVec, OutputVec);
677     NeuralNet->Import12(InputVec, OutputVec, NeuralNet->
678         GetInputNeurons());
679 }
680
681 void DialogNN::on_pushButton_Export12_clicked() {
682     if (fn.size() < 1) {
683         return;
684     }
685     InputLearnVector_t InputVec;
686     OutputLearnVector_t OutputVec;
687     makeLearnVectors(InputVec, OutputVec);
688     NeuralNet->Export12(InputVec, OutputVec, NeuralNet->
689         GetInputNeurons());
690 }
691
692 void DialogNN::on_pushButton_Statistics12_clicked() {
693     if (fn.size() < 1) {
694         return;
695     }
696     InputLearnVector_t InputVec;
697     OutputLearnVector_t OutputVec;
698     makeLearnVectors(InputVec, OutputVec);
699     NeuralNet->Statistics12(InputVec, OutputVec, NeuralNet->
700         GetInputNeurons());
701 }
702
703 void DialogNN::on_pushButton_Plot12_clicked() {
704     if (fn.size() < 1) {
705         return;
706     }
707     InputLearnVector_t InputVec;
708     OutputLearnVector_t OutputVec;
709     makeLearnVectors(InputVec, OutputVec);
710     NeuralNet->Plot12(InputVec, OutputVec, NeuralNet->
711         GetInputNeurons());
712 }
713
714 void DialogNN::on_pushButton_Exit13_clicked() {
715     QApplication::quit();
716 }
717
718 void DialogNN::on_pushButton_Import13_clicked() {
719     if (fn.size() < 1) {
720         return;
721     }
722     InputLearnVector_t InputVec;
723     OutputLearnVector_t OutputVec;
724     makeLearnVectors(InputVec, OutputVec);
725     NeuralNet->Import13(InputVec, OutputVec, NeuralNet->
726         GetInputNeurons());
727 }
728
729 void DialogNN::on_pushButton_Export13_clicked() {
730     if (fn.size() < 1) {
731         return;
732     }
733     InputLearnVector_t InputVec;
734     OutputLearnVector_t OutputVec;
735     makeLearnVectors(InputVec, OutputVec);
736     NeuralNet->Export13(InputVec, OutputVec, NeuralNet->
737         GetInputNeurons());
738 }
739
740 void DialogNN::on_pushButton_Statistics13_clicked() {
741     if (fn.size() < 1) {
742         return;
743     }
744     InputLearnVector_t InputVec;
745     OutputLearnVector_t OutputVec;
746     makeLearnVectors(InputVec, OutputVec);
747     NeuralNet->Statistics13(InputVec, OutputVec, NeuralNet->
748         GetInputNeurons());
749 }
750
751 void DialogNN::on_pushButton_Plot13_clicked() {
752     if (fn.size() < 1) {
753         return;
754     }
755     InputLearnVector_t InputVec;
756     OutputLearnVector_t OutputVec;
757     makeLearnVectors(InputVec, OutputVec);
758     NeuralNet->Plot13(InputVec, OutputVec, NeuralNet->
759         GetInputNeurons());
760 }
761
762 void DialogNN::on_pushButton_Exit14_clicked() {
763     QApplication::quit();
764 }
765
766 void DialogNN::on_pushButton_Import14_clicked() {
767     if (fn.size() < 1) {
768         return;
769     }
770     InputLearnVector_t InputVec;
771     OutputLearnVector_t OutputVec;
772     makeLearnVectors(InputVec, OutputVec);
773     NeuralNet->Import14(InputVec, OutputVec, NeuralNet->
774         GetInputNeurons());
775 }
776
777 void DialogNN::on_pushButton_Export14_clicked() {
778     if (fn.size() < 1) {
779         return;
780     }
781     InputLearnVector_t InputVec;
782     OutputLearnVector_t OutputVec;
783     makeLearnVectors(InputVec, OutputVec);
784     NeuralNet->Export14(InputVec, OutputVec, NeuralNet->
785         GetInputNeurons());
786 }
787
788 void DialogNN::on_pushButton_Statistics14_clicked() {
789     if (fn.size() < 1) {
790         return;
791     }
792     InputLearnVector_t InputVec;
793     OutputLearnVector_t OutputVec;
794     makeLearnVectors(InputVec, OutputVec);
795     NeuralNet->Statistics14(InputVec, OutputVec, NeuralNet->
796         GetInputNeurons());
797 }
798
799 void DialogNN::on_pushButton_Plot14_clicked() {
800     if (fn.size() < 1) {
801         return;
802     }
803     InputLearnVector_t InputVec;
804     OutputLearnVector_t OutputVec;
805     makeLearnVectors(InputVec, OutputVec);
806     NeuralNet->Plot14(InputVec, OutputVec, NeuralNet->
807         GetInputNeurons());
808 }
809
810 void DialogNN::on_pushButton_Exit15_clicked() {
811     QApplication::quit();
812 }
813
814 void DialogNN::on_pushButton_Import15_clicked() {
815     if (fn.size() < 1) {
816         return;
817     }
818     InputLearnVector_t InputVec;
819     OutputLearnVector_t OutputVec;
820     makeLearnVectors(InputVec, OutputVec);
821     NeuralNet->Import15(InputVec, OutputVec, NeuralNet->
822         GetInputNeurons());
823 }
824
825 void DialogNN::on_pushButton_Export15_clicked() {
826     if (fn.size() < 1) {
827         return;
828     }
829     InputLearnVector_t InputVec;
830     OutputLearnVector_t OutputVec;
831     makeLearnVectors(InputVec, OutputVec);
832     NeuralNet->Export15(InputVec, OutputVec, NeuralNet->
833         GetInputNeurons());
834 }
835
836 void DialogNN::on_pushButton_Statistics15_clicked() {
837     if (fn.size() < 1) {
838         return;
839     }
840     InputLearnVector_t InputVec;
841     OutputLearnVector_t OutputVec;
842     makeLearnVectors(InputVec, OutputVec);
843     NeuralNet->Statistics15(InputVec, OutputVec, NeuralNet->
844         GetInputNeurons());
845 }
846
847 void DialogNN::on_pushButton_Plot15_clicked() {
848     if (fn.size() < 1) {
849         return;
850     }
851     InputLearnVector_t InputVec;
852     OutputLearnVector_t OutputVec;
853     makeLearnVectors(InputVec, OutputVec);
854     NeuralNet->Plot15(InputVec, OutputVec, NeuralNet->
855         GetInputNeurons());
856 }
857
858 void DialogNN::on_pushButton_Exit16_clicked() {
859     QApplication::quit();
860 }
861
862 void DialogNN::on_pushButton_Import16_clicked() {
863     if (fn.size() < 1) {
864         return;
865     }
866     InputLearnVector_t InputVec;
867     OutputLearnVector_t OutputVec;
868     makeLearnVectors(InputVec, OutputVec);
869     NeuralNet->Import16(InputVec, OutputVec, NeuralNet->
870         GetInputNeurons());
871 }
872
873 void DialogNN::on_pushButton_Export16_clicked() {
874     if (fn.size() < 1) {
875         return;
876     }
877     InputLearnVector_t InputVec;
878     OutputLearnVector_t OutputVec;
879     makeLearnVectors(InputVec, OutputVec);
880     NeuralNet->Export16(InputVec, OutputVec, NeuralNet->
881         GetInputNeurons());
882 }
883
884 void DialogNN::on_pushButton_Statistics16_clicked() {
885     if (fn.size() < 1) {
886         return;
887     }
888     InputLearnVector_t InputVec;
889     OutputLearnVector_t OutputVec;
890     makeLearnVectors(InputVec, OutputVec);
891     NeuralNet->Statistics16(InputVec, OutputVec, NeuralNet->
892         GetInputNeurons());
893 }
894
895 void DialogNN::on_pushButton_Plot16_clicked() {
896     if (fn.size() < 1) {
897         return;
898     }
899     InputLearnVector_t InputVec;
900     OutputLearnVector_t OutputVec;
901     makeLearnVectors(InputVec, OutputVec);
902     NeuralNet->Plot16(InputVec, OutputVec, NeuralNet->
903         GetInputNeurons());
904 }
905
906 void DialogNN::on_pushButton_Exit17_clicked() {
907     QApplication::quit();
908 }
909
910 void DialogNN::on_pushButton_Import17_clicked() {
911     if (fn.size() < 1) {
912         return;
913     }
914     InputLearnVector_t InputVec;
915     OutputLearnVector_t OutputVec;
916     makeLearnVectors(InputVec, OutputVec);
917     NeuralNet->Import17(InputVec, OutputVec, NeuralNet->
918         GetInputNeurons());
919 }
920
921 void DialogNN::on_pushButton_Export17_clicked() {
922     if (fn.size() < 1) {
923         return;
924     }
925     InputLearnVector_t InputVec;
926     OutputLearnVector_t OutputVec;
927     makeLearnVectors(InputVec, OutputVec);
928     NeuralNet->Export17(InputVec, OutputVec, NeuralNet->
929         GetInputNeurons());
930 }
931
932 void DialogNN::on_pushButton_Statistics17_clicked() {
933     if (fn.size() < 1) {
934         return;
935     }
936     InputLearnVector_t InputVec;
937     OutputLearnVector_t OutputVec;
938     makeLearnVectors(InputVec, OutputVec);
939     NeuralNet->Statistics17(InputVec, OutputVec, NeuralNet->
940         GetInputNeurons());
941 }
942
943 void DialogNN::on_pushButton_Plot17_clicked() {
944     if (fn.size() < 1) {
945         return;
946     }
947     InputLearnVector_t InputVec;
948     OutputLearnVector_t OutputVec;
949     makeLearnVectors(InputVec, OutputVec);
950     NeuralNet->Plot17(InputVec, OutputVec, NeuralNet->
951         GetInputNeurons());
952 }
953
954 void DialogNN::on_pushButton_Exit18_clicked() {
955     QApplication::quit();
956 }
957
958 void DialogNN::on_pushButton_Import18_clicked() {
959     if (fn.size() < 1) {
960         return;
961     }
962     InputLearnVector_t InputVec;
963     OutputLearnVector_t OutputVec;
964     makeLearnVectors(InputVec, OutputVec);
965     NeuralNet->Import18(InputVec, OutputVec, NeuralNet->
966         GetInputNeurons());
967 }
968
969 void DialogNN::on_pushButton_Export18_clicked() {
970     if (fn.size() < 1) {
971         return;
972     }
973     InputLearnVector_t InputVec;
974     OutputLearnVector_t OutputVec;
975     makeLearnVectors(InputVec, OutputVec);
976     NeuralNet->Export18(InputVec, OutputVec, NeuralNet->
977         GetInputNeurons());
978 }
979
980 void DialogNN::on_pushButton_Statistics18_clicked() {
981     if (fn.size() < 1) {
982         return;
983     }
984     InputLearnVector_t InputVec;
985     OutputLearnVector_t OutputVec;
986     makeLearnVectors(InputVec, OutputVec);
987     NeuralNet->Statistics18(InputVec, OutputVec, NeuralNet->
988         GetInputNeurons());
989 }
990
991 void DialogNN::on_pushButton_Plot18_clicked() {
992     if (fn.size() < 1) {
993         return;
994     }
995     InputLearnVector_t InputVec;
996     OutputLearnVector_t OutputVec;
997     makeLearnVectors(InputVec, OutputVec);
998     NeuralNet->Plot18(InputVec, OutputVec, NeuralNet->
1000         GetInputNeurons());
1001 }
1002
1003 void DialogNN::on_pushButton_Exit19_clicked() {
1004     QApplication::quit();
1005 }
1006
1007 void DialogNN::on_pushButton_Import19_clicked() {
1008     if (fn.size() < 1) {
1009         return;
1010     }
1011     InputLearnVector_t InputVec;
1012     OutputLearnVector_t OutputVec;
1013     makeLearnVectors(InputVec, OutputVec);
1014     NeuralNet->Import19(InputVec, OutputVec, NeuralNet->
1015         GetInputNeurons());
1016 }
1017
1018 void DialogNN::on_pushButton_Export19_clicked() {
1019     if (fn.size() < 1) {
1020         return;
1021     }
1022     InputLearnVector_t InputVec;
1023     OutputLearnVector_t OutputVec;
1024     makeLearnVectors(InputVec, OutputVec);
1025     NeuralNet->Export19(InputVec, OutputVec, NeuralNet->
1026         GetInputNeurons());
1027 }
1028
1029 void DialogNN::on_pushButton_Statistics19_clicked() {
1030     if (fn.size() < 1) {
1031         return;
1032     }
1033     InputLearnVector_t InputVec;
1034     OutputLearnVector_t OutputVec;
1035     makeLearnVectors(InputVec, OutputVec);
1036     NeuralNet->Statistics19(InputVec, OutputVec, NeuralNet->
1037         GetInputNeurons());
1038 }
1039
1040 void DialogNN::on_pushButton_Plot19_clicked() {
1041     if (fn.size() < 1) {
1042         return;
1043     }
1044     InputLearnVector_t InputVec;
1045     OutputLearnVector_t OutputVec;
1046     makeLearnVectors(InputVec, OutputVec);
1047     NeuralNet->Plot19(InputVec, OutputVec, NeuralNet->
1048         GetInputNeurons());
1049 }
1050
1051 void DialogNN::on_pushButton_Exit20_clicked() {
1052     QApplication::quit();
1053 }
1054
1055 void DialogNN::on_pushButton_Import20_clicked() {
1056     if (fn.size() < 1) {
1057         return;
1058     }
1059     InputLearnVector_t InputVec;
1060     OutputLearnVector_t OutputVec;
1061     makeLearnVectors(InputVec, OutputVec);
1062     NeuralNet->Import20(InputVec, OutputVec, NeuralNet->
1063         GetInputNeurons());
1064 }
1065
1066 void DialogNN::on_pushButton_Export20_clicked() {
1067     if (fn.size() < 1) {
1068         return;
1069     }
1070     InputLearnVector_t InputVec;
1071     OutputLearnVector_t OutputVec;
1072     makeLearnVectors(InputVec, OutputVec);
1073     NeuralNet->Export20(InputVec, OutputVec, NeuralNet->
1074         GetInputNeurons());
1075 }
1076
1077 void DialogNN::on_pushButton_Statistics20_clicked() {
1078     if (fn.size() < 1) {
1079         return;
1080     }
1081     InputLearnVector_t InputVec;
1082     OutputLearnVector_t OutputVec;
1083     makeLearnVectors(InputVec, OutputVec);
1084     NeuralNet->Statistics20(InputVec, OutputVec, NeuralNet->
1085         GetInputNeurons());
1086 }
1087
1088 void DialogNN::on_pushButton_Plot20_clicked() {
1089     if (fn.size() < 1) {
1090         return;
1091     }
1092     InputLearnVector_t InputVec;
1093     OutputLearnVector_t OutputVec;
1094     makeLearn
```

```

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

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



Q. Reference manual

Vision Soil Analyzer

1.0.0

Generated by Doxygen 1.8.9.1

Mon Oct 5 2015 21:06:41

Contents

1 Namespace Index	355
1.1 Namespace List	355
2 Hierarchical Index	355
2.1 Class Hierarchy	355
3 Class Index	357
3.1 Class List	357
4 File Index	359
4.1 File List	360
5 Namespace Documentation	362
5.1 boost Namespace Reference	362
5.2 boost::serialization Namespace Reference	362
5.2.1 Function Documentation	362
5.3 Hardware Namespace Reference	363
5.3.1 Typedef Documentation	363
5.3.2 Function Documentation	363
5.4 Hardware::Exception Namespace Reference	364
5.5 SoilAnalyzer Namespace Reference	364
5.6 SoilAnalyzer::Exception Namespace Reference	365
5.7 SoilMath Namespace Reference	365
5.7.1 Detailed Description	365
5.7.2 Function Documentation	366
5.8 SoilMath::Exception Namespace Reference	369
5.9 Ui Namespace Reference	370
5.10 Vision Namespace Reference	370
5.11 Vision::Exception Namespace Reference	370
6 Class Documentation	370
6.1 Hardware::Microscope::__CustomData Struct Reference	370
6.1.1 Detailed Description	372
6.1.2 Member Data Documentation	372
6.2 Hardware::ADC Class Reference	373
6.2.1 Detailed Description	375
6.2.2 Member Enumeration Documentation	375
6.2.3 Constructor & Destructor Documentation	375
6.2.4 Member Function Documentation	376
6.2.5 Friends And Related Function Documentation	378
6.2.6 Member Data Documentation	378
6.3 ADC Class Reference	378
6.3.1 Detailed Description	379
6.4 Hardware::Exception::ADCReadException Class Reference	379
6.4.1 Detailed Description	380
6.4.2 Constructor & Destructor Documentation	380
6.4.3 Member Function Documentation	380

6.4.4	Member Data Documentation	380
6.5	SoilAnalyzer::Analyzer Class Reference	381
6.5.1	Detailed Description	383
6.5.2	Member Typedef Documentation	383
6.5.3	Constructor & Destructor Documentation	384
6.5.4	Member Function Documentation	384
6.5.5	Member Data Documentation	391
6.6	Hardware::BBB Class Reference	392
6.6.1	Detailed Description	394
6.6.2	Constructor & Destructor Documentation	394
6.6.3	Member Function Documentation	394
6.6.4	Member Data Documentation	397
6.7	BBB Class Reference	398
6.7.1	Detailed Description	398
6.8	Vision::Segment::Blob Struct Reference	398
6.8.1	Detailed Description	399
6.8.2	Constructor & Destructor Documentation	399
6.8.3	Member Data Documentation	399
6.9	Hardware::Microscope::Cam_t Struct Reference	399
6.9.1	Detailed Description	401
6.9.2	Member Function Documentation	401
6.9.3	Member Data Documentation	401
6.10	Vision::Exception::ChannelMismatchException Class Reference	402
6.10.1	Detailed Description	403
6.10.2	Constructor & Destructor Documentation	403
6.10.3	Member Function Documentation	403
6.10.4	Member Data Documentation	403
6.11	ChannelMismatchException Class Reference	403
6.11.1	Detailed Description	403
6.12	Hardware::Microscope::Control_t Struct Reference	404
6.12.1	Detailed Description	404
6.12.2	Member Function Documentation	404
6.12.3	Member Data Documentation	404
6.13	Vision::Conversion Class Reference	405
6.13.1	Detailed Description	408
6.13.2	Member Enumeration Documentation	408
6.13.3	Constructor & Destructor Documentation	408
6.13.4	Member Function Documentation	409
6.13.5	Member Data Documentation	413
6.14	Conversion Class Reference	414
6.14.1	Detailed Description	414
6.15	Vision::Exception::ConversionNotSupportedException Class Reference	414
6.15.1	Detailed Description	415
6.15.2	Constructor & Destructor Documentation	416
6.15.3	Member Function Documentation	416
6.15.4	Member Data Documentation	416

6.16	ConversionNotSupportedException Class Reference	416
6.16.1	Detailed Description	416
6.17	Hardware::Exception::CouldNotGrabImageException Class Reference	416
6.17.1	Detailed Description	417
6.17.2	Constructor & Destructor Documentation	418
6.17.3	Member Function Documentation	418
6.17.4	Member Data Documentation	418
6.18	DialogNN Class Reference	418
6.18.1	Detailed Description	421
6.18.2	Constructor & Destructor Documentation	421
6.18.3	Member Function Documentation	421
6.18.4	Member Data Documentation	424
6.19	DialogSettings Class Reference	425
6.19.1	Detailed Description	428
6.19.2	Constructor & Destructor Documentation	429
6.19.3	Member Function Documentation	429
6.19.4	Member Data Documentation	435
6.20	Hardware::EC12P Class Reference	436
6.20.1	Detailed Description	438
6.20.2	Member Enumeration Documentation	438
6.20.3	Constructor & Destructor Documentation	438
6.20.4	Member Function Documentation	439
6.20.5	Friends And Related Function Documentation	439
6.20.6	Member Data Documentation	440
6.21	EC12P Class Reference	441
6.21.1	Detailed Description	441
6.22	EmtpyImageException Class Reference	441
6.22.1	Detailed Description	441
6.23	Vision::Exception::EmtpyImageException Class Reference	441
6.23.1	Detailed Description	442
6.23.2	Constructor & Destructor Documentation	443
6.23.3	Member Function Documentation	443
6.23.4	Member Data Documentation	443
6.24	Vision::Enhance Class Reference	443
6.24.1	Detailed Description	446
6.24.2	Member Enumeration Documentation	446
6.24.3	Constructor & Destructor Documentation	446
6.24.4	Member Function Documentation	446
6.25	Enhance Class Reference	448
6.25.1	Detailed Description	448
6.26	Hardware::eQEP Class Reference	448
6.26.1	Detailed Description	451
6.26.2	Member Enumeration Documentation	451
6.26.3	Constructor & Destructor Documentation	451
6.26.4	Member Function Documentation	451
6.26.5	Friends And Related Function Documentation	453

6.26.6 Member Data Documentation	453
6.27 Hardware::Exception::FailedToCreateGPIOPollingThreadException Class Reference	453
6.27.1 Detailed Description	455
6.27.2 Constructor & Destructor Documentation	455
6.27.3 Member Function Documentation	455
6.27.4 Member Data Documentation	455
6.28 Hardware::Exception::FailedToCreateThreadException Class Reference	455
6.28.1 Detailed Description	456
6.28.2 Constructor & Destructor Documentation	456
6.28.3 Member Function Documentation	456
6.28.4 Member Data Documentation	456
6.29 SoilMath::FFT Class Reference	457
6.29.1 Detailed Description	458
6.29.2 Constructor & Destructor Documentation	458
6.29.3 Member Function Documentation	458
6.29.4 Member Data Documentation	461
6.30 SoilMath::GA Class Reference	462
6.30.1 Detailed Description	464
6.30.2 Constructor & Destructor Documentation	465
6.30.3 Member Function Documentation	465
6.30.4 Member Data Documentation	469
6.31 Hardware::GPIO Class Reference	470
6.31.1 Detailed Description	473
6.31.2 Member Enumeration Documentation	473
6.31.3 Constructor & Destructor Documentation	474
6.31.4 Member Function Documentation	474
6.31.5 Friends And Related Function Documentation	480
6.31.6 Member Data Documentation	480
6.32 Hardware::Exception::GPIOReadException Class Reference	480
6.32.1 Detailed Description	481
6.32.2 Constructor & Destructor Documentation	482
6.32.3 Member Function Documentation	482
6.32.4 Member Data Documentation	482
6.33 SoilAnalyzer::Analyzer::Image_t Struct Reference	482
6.33.1 Detailed Description	482
6.33.2 Member Data Documentation	482
6.34 ImageProcessing Class Reference	483
6.34.1 Detailed Description	483
6.35 Vision::ImageProcessing Class Reference	483
6.35.1 Detailed Description	485
6.35.2 Member Typedef Documentation	485
6.35.3 Constructor & Destructor Documentation	485
6.35.4 Member Function Documentation	486
6.35.5 Member Data Documentation	488
6.36 SoilAnalyzer::Lab_t Struct Reference	489
6.36.1 Detailed Description	489

6.36.2 Member Data Documentation	489
6.37 SoilMath::Exception::MathException Class Reference	489
6.37.1 Detailed Description	491
6.37.2 Constructor & Destructor Documentation	491
6.37.3 Member Function Documentation	491
6.37.4 Member Data Documentation	491
6.38 Hardware::Microscope Class Reference	491
6.38.1 Detailed Description	494
6.38.2 Member Typedef Documentation	494
6.38.3 Member Enumeration Documentation	495
6.38.4 Constructor & Destructor Documentation	495
6.38.5 Member Function Documentation	496
6.38.6 Member Data Documentation	502
6.39 Microscope Class Reference	503
6.39.1 Detailed Description	503
6.40 Hardware::Exception::MicroscopeException Class Reference	503
6.40.1 Detailed Description	505
6.40.2 Constructor & Destructor Documentation	505
6.40.3 Member Function Documentation	505
6.40.4 Member Data Documentation	505
6.41 Vision::MorphologicalFilter Class Reference	505
6.41.1 Detailed Description	508
6.41.2 Member Enumeration Documentation	508
6.41.3 Constructor & Destructor Documentation	508
6.41.4 Member Function Documentation	509
6.42 SoilMath::NN Class Reference	511
6.42.1 Detailed Description	515
6.42.2 Constructor & Destructor Documentation	515
6.42.3 Member Function Documentation	515
6.42.4 Friends And Related Function Documentation	521
6.42.5 Member Data Documentation	521
6.43 SoilAnalyzer::Particle Class Reference	523
6.43.1 Detailed Description	525
6.43.2 Member Typedef Documentation	525
6.43.3 Constructor & Destructor Documentation	526
6.43.4 Member Function Documentation	526
6.43.5 Friends And Related Function Documentation	529
6.43.6 Member Data Documentation	529
6.44 Vision::Exception::PixelValueOutOfBoundsException Reference	531
6.44.1 Detailed Description	532
6.44.2 Constructor & Destructor Documentation	532
6.44.3 Member Function Documentation	532
6.44.4 Member Data Documentation	532
6.45 PixelValueOutOfBoundsException Class Reference	533
6.45.1 Detailed Description	533
6.46 SoilAnalyzer::Point_t Struct Reference	533

6.46.1	Detailed Description	533
6.46.2	Member Data Documentation	533
6.47	PopMemberStruct Struct Reference	534
6.47.1	Detailed Description	534
6.47.2	Member Data Documentation	534
6.48	Predict_struct Struct Reference	535
6.48.1	Detailed Description	535
6.48.2	Member Data Documentation	535
6.49	SoilMath::PSD Class Reference	536
6.49.1	Detailed Description	539
6.49.2	Constructor & Destructor Documentation	539
6.49.3	Member Function Documentation	539
6.49.4	Friends And Related Function Documentation	540
6.50	Hardware::PWM Class Reference	540
6.50.1	Detailed Description	543
6.50.2	Member Enumeration Documentation	543
6.50.3	Constructor & Destructor Documentation	544
6.50.4	Member Function Documentation	544
6.50.5	Member Data Documentation	548
6.51	QOpenCVQT Class Reference	549
6.51.1	Detailed Description	550
6.51.2	Constructor & Destructor Documentation	550
6.51.3	Member Function Documentation	550
6.52	QParticleDisplay Class Reference	550
6.52.1	Detailed Description	552
6.52.2	Constructor & Destructor Documentation	552
6.52.3	Member Function Documentation	552
6.52.4	Member Data Documentation	556
6.53	QParticleSelector Class Reference	556
6.53.1	Detailed Description	559
6.53.2	Constructor & Destructor Documentation	559
6.53.3	Member Function Documentation	559
6.53.4	Member Data Documentation	561
6.54	QReportGenerator Class Reference	562
6.54.1	Detailed Description	564
6.54.2	Constructor & Destructor Documentation	564
6.54.3	Member Function Documentation	565
6.54.4	Member Data Documentation	566
6.55	Vision::Segment::Rect Struct Reference	568
6.55.1	Detailed Description	568
6.55.2	Constructor & Destructor Documentation	569
6.55.3	Member Data Documentation	569
6.56	Hardware::Microscope::Resolution_t Struct Reference	569
6.56.1	Detailed Description	570
6.56.2	Member Function Documentation	570
6.56.3	Member Data Documentation	570

6.57	SoilAnalyzer::Sample Class Reference	570
6.57.1	Detailed Description	572
6.57.2	Constructor & Destructor Documentation	572
6.57.3	Member Function Documentation	573
6.57.4	Friends And Related Function Documentation	575
6.57.5	Member Data Documentation	575
6.58	Segment Class Reference	578
6.58.1	Detailed Description	578
6.59	Vision::Segment Class Reference	578
6.59.1	Detailed Description	581
6.59.2	Member Typedef Documentation	581
6.59.3	Member Enumeration Documentation	582
6.59.4	Constructor & Destructor Documentation	582
6.59.5	Member Function Documentation	583
6.59.6	Member Data Documentation	589
6.60	SoilAnalyzer::Exception::SoilAnalyzerException Class Reference	590
6.60.1	Detailed Description	591
6.60.2	Constructor & Destructor Documentation	591
6.60.3	Member Function Documentation	591
6.60.4	Member Data Documentation	592
6.61	Hardware::SoilCape Class Reference	592
6.61.1	Detailed Description	593
6.61.2	Constructor & Destructor Documentation	594
6.61.3	Member Data Documentation	594
6.62	SoilAnalyzer::SoilSettings Class Reference	594
6.62.1	Detailed Description	596
6.62.2	Constructor & Destructor Documentation	596
6.62.3	Member Function Documentation	596
6.62.4	Friends And Related Function Documentation	597
6.62.5	Member Data Documentation	597
6.63	SoilMath::Sort Class Reference	602
6.63.1	Detailed Description	603
6.63.2	Constructor & Destructor Documentation	603
6.63.3	Member Function Documentation	603
6.64	SoilMath::Stats< T1, T2, T3 > Class Template Reference	603
6.64.1	Detailed Description	606
6.64.2	Constructor & Destructor Documentation	606
6.64.3	Member Function Documentation	606
6.64.4	Friends And Related Function Documentation	608
6.64.5	Member Data Documentation	608
6.65	Hardware::USB Class Reference	610
6.65.1	Detailed Description	611
6.65.2	Constructor & Destructor Documentation	611
6.65.3	Member Function Documentation	611
6.66	Hardware::Exception::ValueOutOfBoundsException Class Reference	611
6.66.1	Detailed Description	612

6.66.2 Constructor & Destructor Documentation	613
6.66.3 Member Function Documentation	613
6.66.4 Member Data Documentation	613
6.67 VSAMainWindow Class Reference	613
6.67.1 Detailed Description	616
6.67.2 Constructor & Destructor Documentation	616
6.67.3 Member Function Documentation	617
6.67.4 Member Data Documentation	624
6.68 Vision::Exception::WrongKernelSizeException Class Reference	626
6.68.1 Detailed Description	627
6.68.2 Constructor & Destructor Documentation	628
6.68.3 Member Function Documentation	628
6.68.4 Member Data Documentation	628
6.69 WrongKernelSizeException Class Reference	628
6.69.1 Detailed Description	628

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	362
boost::serialization	362
Hardware	363
Hardware::Exception	364
SoilAnalyzer	364
SoilAnalyzer::Exception	365
SoilMath Genetic Algorithmes used for optimization problems	365
SoilMath::Exception	369
Ui	370
Vision	370
Vision::Exception	370

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Hardware::Microscope::_CustomData	370
ADC	378
Hardware::BBB	392
Hardware::ADC	373
Hardware::eQEP	448

Hardware::GPIO	470
Hardware::PWM	540
BBB	398
Vision::Segment::Blob	398
Hardware::Microscope::Cam_t	399
ChannelMismatchException	403
Hardware::Microscope::Control_t	404
Conversion	414
ConversionNotSupportedException	416
Hardware::EC12P	436
EC12P	441
EmtpyImageException	441
Enhance	448
exception	
Hardware::Exception::ADCReadException	379
Hardware::Exception::CouldNotGrabImageException	416
Hardware::Exception::FailedToCreateGPIOPollingThreadException	453
Hardware::Exception::FailedToCreateThreadException	455
Hardware::Exception::GPIOReadException	480
Hardware::Exception::MicroscopeException	503
Hardware::Exception::ValueOutOfBoundsException	611
SoilAnalyzer::Exception::SoilAnalyzerException	590
SoilMath::Exception::MathException	489
Vision::Exception::ChannelMismatchException	402
Vision::Exception::ConversionNotSupportedException	414
Vision::Exception::EmtpyImageException	441
Vision::Exception::PixelValueOutOfBoundsException	531
Vision::Exception::WrongKernelSizeException	626
SoilMath::FFT	457
SoilAnalyzer::Analyzer::Image_t	482
ImageProcessing	483
Vision::ImageProcessing	483
Vision::Conversion	405
Vision::Enhance	443
Vision::MorphologicalFilter	505
Vision::Segment	578
SoilAnalyzer::Lab_t	489
Microscope	503

SoilAnalyzer::Particle	523
PixelValueOutOfBoundException	533
SoilAnalyzer::Point_t	533
PopMemberStruct	534
Predict_struct	535
QDialog	
DialogNN	418
DialogSettings	425
QMainWindow	
QReportGenerator	562
VSAMainWindow	613
QObject	
Hardware::Microscope	491
SoilAnalyzer::Analyzer	381
SoilMath::GA	462
SoilMath::NN	511
QOpenCVQT	549
QWidget	
QParticleDisplay	550
QParticleSelector	556
Vision::Segment::Rect	568
Hardware::Microscope::Resolution_t	569
SoilAnalyzer::Sample	570
Segment	578
Hardware::SoilCape	592
SoilAnalyzer::SoilSettings	594
SoilMath::Sort	602
SoilMath::Stats< T1, T2, T3 >	603
SoilMath::Stats< double, double, long double >	603
SoilMath::PSD	536
SoilMath::Stats< float, double, long double >	603
SoilMath::Stats< uchar, uint32_t, uint64_t >	603
Hardware::USB	610
WrongKernelSizeException	628

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Hardware::Microscope::_CustomData	370
--	-----

Hardware::ADC	373
ADC	378
Hardware::Exception::ADCReadException	379
SoilAnalyzer::Analyzer	381
Hardware::BBB	392
BBB	398
Vision::Segment::Blob	398
Hardware::Microscope::Cam_t	399
Vision::Exception::ChannelMismatchException	402
ChannelMismatchException	403
Hardware::Microscope::Control_t	404
Vision::Conversion	405
Conversion	414
Vision::Exception::ConversionNotSupportedException	414
ConversionNotSupportedException	416
Hardware::Exception::CouldNotGrabImageException	416
DialogNN	418
DialogSettings	425
Hardware::EC12P	436
EC12P	441
EmtpyImageException	441
Vision::Exception::EmtpyImageException	441
Vision::Enhance	443
Enhance	448
Hardware::eQEP	448
Hardware::Exception::FailedToCreateGPIOPollingThreadException	453
Hardware::Exception::FailedToCreateThreadException	455
SoilMath::FFT Fast Fourier Transform class	457
SoilMath::GA	462
Hardware::GPIO	470
Hardware::Exception::GPIOReadException	480
SoilAnalyzer::Analyzer::Image_t	482
ImageProcessing Core class of all the image classes Core class of all the image classes with a few commonly shared functions and variables	483
Vision::ImageProcessing	483
SoilAnalyzer::Lab_t	489
SoilMath::Exception::MathException	489

Hardware::Microscope	491
Microscope	503
Hardware::Exception::MicroscopeException	503
Vision::MorphologicalFilter	505
SoilMath::NN	
The Neural Network class	511
SoilAnalyzer::Particle	523
Vision::Exception::PixelValueOutOfBoundsException	531
PixelValueOutOfBoundsException	533
SoilAnalyzer::Point_t	533
PopMemberStruct	534
Predict_struct	535
SoilMath::PSD	536
Hardware::PWM	540
QOpenCVQT	549
QParticleDisplay	550
QParticleSelector	556
QReportGenerator	562
Vision::Segment::Rect	568
Hardware::Microscope::Resolution_t	569
SoilAnalyzer::Sample	570
Segment	
Segmentation algorithms With this class, various segmentation routines can be applied to a greyscale or black and white source image	578
Vision::Segment	578
SoilAnalyzer::Exception::SoilAnalyzerException	590
Hardware::SoilCape	592
SoilAnalyzer::SoilSettings	
The SoilSettings class	594
SoilMath::Sort	
The Sort template class	602
SoilMath::Stats< T1, T2, T3 >	
Stats class	603
Hardware::USB	610
Hardware::Exception::ValueOutOfBoundsException	611
VSAMainWindow	613
Vision::Exception::WrongKernelSizeException	626
WrongKernelSizeException	628

4.1 File List

Here is a list of all files with brief descriptions:

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QOpenCVQT/qopencvqt.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QOpenCVQT/qopencvqt.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleDisplay/qparticledisplay.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleDisplay/qparticledisplay.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleSelector/qparticleselector.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleSelector/qparticleselector.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QReportGenerator/qreportgenerator.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QReportGenerator/qreportgenerator.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/lab_t_archive.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/particle.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/particle.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/sample.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/sample.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzer.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzerexception.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzertypes.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilsettings.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilsettings.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADC.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADC.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADCReadException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/CouldNotGrabImageException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/eqep.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/eqep.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/FailedToCreateGPIOPollingThreadException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/FailedToCreateThreadException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIO.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIO.h	??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIOReadException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Hardware.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/MicroscopeNotFoundException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ValueOutOfBoundsException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/CommonOperations.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Mat_archive.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/MathException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/predict_t_archive.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/psd.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMath.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMathTypes.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Sort.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Stats.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ChannelMismatchException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ConversionNotSupportedException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/EmptyImageException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/MorphologicalFilter.cpp	??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ MorphologicalFilter.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ PixelValueOutOfBoundException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ Segment.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ Segment.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ Vision.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ VisionDebug.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ WrongKernelSizeException.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ dialognn.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ dialognn.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ dialogsettings.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ dialogsettings.h	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ main.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ vsemainwindow.cpp	??
/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/ vsemainwindow.h	??

5 Namespace Documentation

5.1 boost Namespace Reference

Namespaces

- [serialization](#)

5.2 boost::serialization Namespace Reference

Functions

- template<class Archive >
void [serialize](#) (Archive &ar, cv::Mat &m, const unsigned int version __attribute__((unused)))
 serialize Serialize the openCV mat to disk
- template<class Archive >
void [serialize](#) (Archive &ar, [Predict_t](#) &P, const unsigned int version __attribute__((unused)))
 serialize Serialize the openCV mat to disk
- template<class Archive >
void [serialize](#) (Archive &ar, [SoilAnalyzer::Lab_t](#) &P, const unsigned int version __attribute__((unused)))
 serialize Serialize the openCV mat to disk

5.2.1 Function Documentation

5.2.1.1 template<class Archive > void boost::serialization::serialize (Archive & ar, [SoilAnalyzer::Lab_t](#) & P, const unsigned int version __attribute__((unused))) [inline]

serialize Serialize the openCV mat to disk

Definition at line 21 of file [lab_t_archive.h](#).

References [SoilAnalyzer::Lab_t::a](#), [SoilAnalyzer::Lab_t::b](#), and [SoilAnalyzer::Lab_t::L](#).

5.2.1.2 template<class Archive > void boost::serialization::serialize (Archive & ar, cv::Mat & m, const unsigned int version __attribute__((unused))) [inline]

serialize Serialize the openCV mat to disk

Definition at line 24 of file [Mat_archive.h](#).

5.2.1.3 `template<class Archive > void boost::serialization::serialize (Archive & ar, Predict_t & P, const unsigned int version __attribute__(unused)) [inline]`

serialize Serialize the openCV mat to disk

Definition at line 25 of file [predict_t_archive.h](#).

References [Predict_struct::Accuracy](#), [Predict_struct::Category](#), [Predict_struct::OutputNeurons](#), and [Predict_struct::RealValue](#).

5.3 Hardware Namespace Reference

Namespaces

- [Exception](#)

Classes

- class [ADC](#)
- class [BBB](#)
- class [EC12P](#)
- class [eQEP](#)
- class [GPIO](#)
- class [Microscope](#)
- class [PWM](#)
- class [SoilCape](#)
- class [USB](#)

Typedefs

- `typedef int(* CallbackType) (int)`

Functions

- `void * threadedPollADC (void *value)`
- `void * colorLoop (void *value)`
- `void * threadedPolleqep (void *value)`
- `void * threadedPollGPIO (void *value)`

5.3.1 Typedef Documentation

5.3.1.1 `typedef int(* Hardware::CallbackType) (int)`

CallbackType used to pass a function to a thread

Definition at line 37 of file [BBB.h](#).

5.3.2 Function Documentation

5.3.2.1 `void * Hardware::colorLoop (void * value)`

The thread function that runs trough all the colors

Definition at line 91 of file [EC12P.cpp](#).

References [Hardware::EC12P::SetPixelColor\(\)](#), [Hardware::EC12P::sleepperiod](#), and [Hardware::EC12P::threadRunning](#).

Here is the call graph for this function:



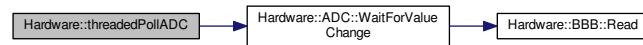
5.3.2.2 void * Hardware::threadedPollADC (void * value)

friendly function to start the threading

Definition at line 121 of file [ADC.cpp](#).

References [Hardware::BBB::callbackFunction](#), [Hardware::BBB::threadRunning](#), and [Hardware::ADC::WaitForValueChange\(\)](#).

Here is the call graph for this function:

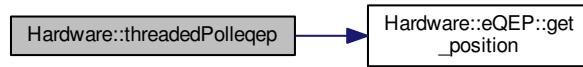


5.3.2.3 void * Hardware::threadedPolleqep (void * value)

Definition at line 242 of file [eQEP.cpp](#).

References [Hardware::BBB::callbackFunction](#), [Hardware::BBB::debounceTime](#), [Hardware::eQEP::get_position\(\)](#), and [Hardware::BBB::threadRunning](#).

Here is the call graph for this function:



5.3.2.4 void * Hardware::threadedPollGPIO (void * value)

Definition at line 266 of file [GPIO.cpp](#).

References [Hardware::BBB::callbackFunction](#), [Hardware::BBB::debounceTime](#), [Hardware::BBB::threadRunning](#), and [Hardware::GPIO::WaitForEdge\(\)](#).

Here is the call graph for this function:



5.4 Hardware::Exception Namespace Reference

Classes

- class [ADCReadException](#)
- class [CouldNotGrabImageException](#)
- class [FailedToCreateGPIOPollingThreadException](#)
- class [FailedToCreateThreadException](#)
- class [GPIOReadException](#)
- class [MicroscopeException](#)
- class [ValueOutOfBoundsException](#)

5.5 SoilAnalyzer Namespace Reference

Namespaces

- [Exception](#)

Classes

- class [Analyzer](#)
- struct [Lab_t](#)
- class [Particle](#)
- struct [Point_t](#)
- class [Sample](#)
- class [SoilSettings](#)

The [SoilSettings](#) class.

5.6 SoilAnalyzer::Exception Namespace Reference

Classes

- class [SoilAnalyzerException](#)

5.7 SoilMath Namespace Reference

Genetic Algorithms used for optimization problems.

Namespaces

- [Exception](#)

Classes

- class [FFT](#)
Fast Fourier Transform class.
- class [GA](#)
- class [NN](#)
The Neural Network class.
- class [PSD](#)
- class [Sort](#)
The [Sort](#) template class.
- class [Stats](#)
[Stats](#) class.

Functions

- `uint16_t MinNotZero (uint16_t a, uint16_t b)`
- `uint16_t Max (uint16_t a, uint16_t b)`
- `uint16_t Max (uint16_t a, uint16_t b, uint16_t c, uint16_t d)`
- `uint16_t Min (uint16_t a, uint16_t b)`
- `uint16_t Min (uint16_t a, uint16_t b, uint16_t c, uint16_t d)`
- static double [quick_pow10](#) (int n)
- static double [fastPow](#) (double a, double b)
- static double [quick_pow2](#) (int n)
- static long [float2intRound](#) (double d)
- static float [calcVolume](#) (float A)
[calcVolume](#) according to ISO 9276-6
- static std::vector< float > [makeOutput](#) (uint8_t value, uint32_t noNeurons)
- static float [calcDiameter](#) (float A)
[calcDiameter](#) according to ISO 9276-6

5.7.1 Detailed Description

Genetic Algorithms used for optimization problems.

Use this class for optimization problems. It's currently optimized for Neural Network optimization

5.7.2 Function Documentation

5.7.2.1 static float SoilMath::calcDiameter(float A) [inline], [static]

calcDiameter according to ISO 9276-6

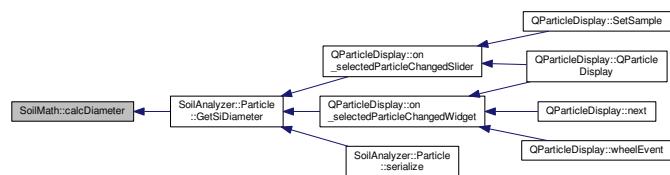
Parameters

A

Returns

Definition at line 115 of file [CommonOperations.h](#).Referenced by [SoilAnalyzer::Particle::GetSiDiameter\(\)](#).

Here is the caller graph for this function:



5.7.2.2 static float SoilMath::calcVolume (float A) [inline], [static]

calcVolume according to ISO 9276-6

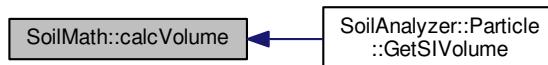
Parameters

A

Returns

Definition at line 100 of file [CommonOperations.h](#).Referenced by [SoilAnalyzer::Particle::GetSIVolume\(\)](#).

Here is the caller graph for this function:



5.7.2.3 static double SoilMath::fastPow (double a, double b) [inline], [static]

Definition at line 49 of file [CommonOperations.h](#).Referenced by [Vision::Enhance::CalculateStdOfNeighboringPixels\(\)](#).

Here is the caller graph for this function:



5.7.2.4 static long SoilMath::float2intRound (double d) [inline], [static]

Definition at line 90 of file [CommonOperations.h](#).

5.7.2.5 static std::vector<float> SoilMath::makeOutput (uint8_t value, uint32_t noNeurons) [inline], [static]

Definition at line 104 of file [CommonOperations.h](#).

Referenced by [DialogNN::makeLearnVectors\(\)](#).

Here is the caller graph for this function:



5.7.2.6 uint16_t SoilMath::Max (uint16_t a, uint16_t b) [inline]

Definition at line 25 of file [CommonOperations.h](#).

Referenced by [Max\(\)](#).

Here is the caller graph for this function:



5.7.2.7 uint16_t SoilMath::Max (uint16_t a, uint16_t b, uint16_t c, uint16_t d) [inline]

Definition at line 27 of file [CommonOperations.h](#).

References [Max\(\)](#).

Here is the call graph for this function:



5.7.2.8 `uint16_t SoilMath::Min (uint16_t a, uint16_t b)` [inline]

Definition at line 31 of file [CommonOperations.h](#).

Referenced by [Min\(\)](#).

Here is the caller graph for this function:



5.7.2.9 `uint16_t SoilMath::Min (uint16_t a, uint16_t b, uint16_t c, uint16_t d)` [inline]

Definition at line 33 of file [CommonOperations.h](#).

References [Min\(\)](#).

Here is the call graph for this function:



5.7.2.10 `uint16_t SoilMath::MinNotZero (uint16_t a, uint16_t b)` [inline]

Definition at line 17 of file [CommonOperations.h](#).

5.7.2.11 `static double SoilMath::quick_pow10 (int n)` [inline], [static]

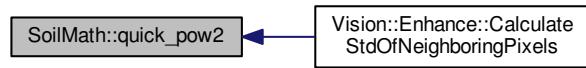
Definition at line 37 of file [CommonOperations.h](#).

5.7.2.12 `static double SoilMath::quick_pow2 (int n)` [inline], [static]

Definition at line 59 of file [CommonOperations.h](#).

Referenced by [Vision::Enhance::CalculateStdOfNeighboringPixels\(\)](#).

Here is the caller graph for this function:



5.8 SoilMath::Exception Namespace Reference

Classes

- class [MathException](#)

5.9 Ui Namespace Reference

5.10 Vision Namespace Reference

Namespaces

- [Exception](#)

Classes

- class [Conversion](#)
- class [Enhance](#)
- class [ImageProcessing](#)
- class [MorphologicalFilter](#)
- class [Segment](#)

5.11 Vision::Exception Namespace Reference

Classes

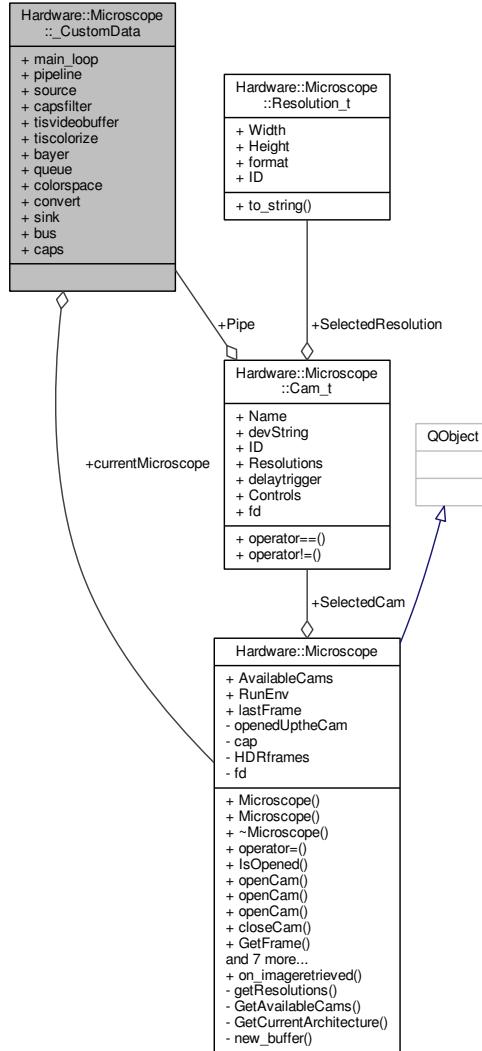
- class [ChannelMismatchException](#)
- class [ConversionNotSupportedException](#)
- class [EmptyImageException](#)
- class [PixelValueOutOfBoundsException](#)
- class [WrongKernelSizeException](#)

6 Class Documentation

6.1 Hardware::Microscope::_CustomData Struct Reference

```
#include <Microscope.h>
```

Collaboration diagram for Hardware::Microscope::_CustomData:



Public Attributes

- `GMainLoop * main_loop`
- `GstElement * pipeline`
- `GstElement * source`
- `GstElement * capsfilter`
- `GstElement * tisvideobuffer`
- `GstElement * tiscolorize`
- `GstElement * bayer`
- `GstElement * queue`
- `GstElement * colorspace`
- `GstElement * convert`
- `GstElement * sink`
- `GstBus * bus`
- `GstCaps * caps`
- `Hardware::Microscope * currentMicroscope`

6.1.1 Detailed Description

Definition at line 105 of file [Microscope.h](#).

6.1.2 Member Data Documentation

6.1.2.1 GstElement* Hardware::Microscope::_CustomData::bayer

Definition at line 112 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.2 GstBus* Hardware::Microscope::_CustomData::bus

Definition at line 117 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.3 GstCaps* Hardware::Microscope::_CustomData::caps

Definition at line 118 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.4 GstElement* Hardware::Microscope::_CustomData::capsfilter

Definition at line 109 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.5 GstElement* Hardware::Microscope::_CustomData::colorspace

Definition at line 114 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.6 GstElement* Hardware::Microscope::_CustomData::convert

Definition at line 115 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.7 Hardware::Microscope* Hardware::Microscope::_CustomData::currentMicroscope

Definition at line 119 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::new_buffer\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

6.1.2.8 GMainLoop* Hardware::Microscope::_CustomData::main_loop

Definition at line 106 of file [Microscope.h](#).

6.1.2.9 GstElement* Hardware::Microscope::_CustomData::pipeline

Definition at line 107 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::closeCam\(\)](#), [Hardware::Microscope::GetFrame\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

6.1.2.10 GstElement* Hardware::Microscope::_CustomData::queue

Definition at line 113 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.11 GstElement* Hardware::Microscope::_CustomData::sink

Definition at line 116 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.12 GstElement* Hardware::Microscope::_CustomData::source

Definition at line 108 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.13 GstElement* Hardware::Microscope::_CustomData::tiscolorize

Definition at line 111 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

6.1.2.14 GstElement* Hardware::Microscope::_CustomData::tisvideobuffer

Definition at line 110 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::openCam\(\)](#).

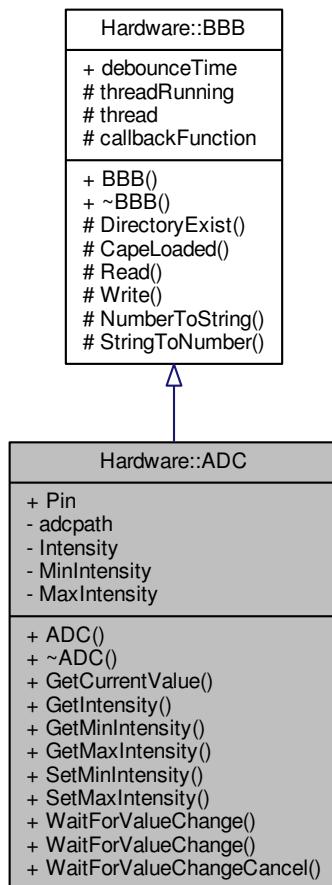
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h](#)

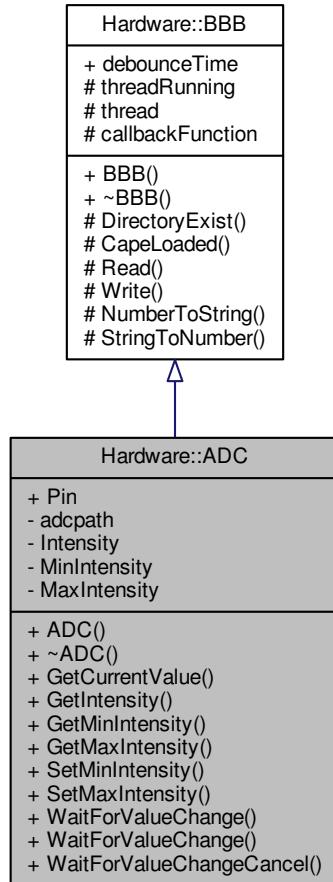
6.2 Hardware::ADC Class Reference

```
#include <ADC.h>
```

Inheritance diagram for Hardware::ADC:



Collaboration diagram for Hardware::ADC:



Public Types

- enum **ADCPin** {
 ADC0, **ADC1**, **ADC2**, **ADC3**,
ADC4, **ADC5**, **ADC6**, **ADC7** }

Public Member Functions

- ADC (ADCPin pin)**
- ~ADC ()**
- int GetCurrentValue ()**
- float GetIntensity ()**
- int GetMinIntensity ()**
- int GetMaxIntensity ()**
- void SetMinIntensity ()**
- void SetMaxIntensity ()**
- int WaitForValueChange ()**
- int WaitForValueChange (CallbackType callback)**
- void WaitForValueChangeCancel ()**

Public Attributes

- [ADCPin Pin](#)

Private Attributes

- string [adcpath](#)
- float [Intensity](#)
- int [MinIntensity](#)
- int [MaxIntensity](#)

Friends

- void * [threadedPollADC](#) (void *value)

Additional Inherited Members**6.2.1 Detailed Description**

Definition at line 51 of file [ADC.h](#).

6.2.2 Member Enumeration Documentation**6.2.2.1 enum Hardware::ADC::ADCPin**

Enumerator to indicate the analogue pin

Enumerator

- ADC0** AIN0 pin
- ADC1** AIN1 pin
- ADC2** AIN2 pin
- ADC3** AIN3 pin
- ADC4** AIN4 pin
- ADC5** AIN5 pin
- ADC6** AIN6 pin
- ADC7** AIN7 pin

Definition at line 54 of file [ADC.h](#).

6.2.3 Constructor & Destructor Documentation**6.2.3.1 ADC::ADC (ADCPin pin)**

Constructor

Parameters

<i>pin</i>	and ADCPin type indicating which analogue pin to use
------------	--

Definition at line 14 of file [ADC.cpp](#).

References [ADC0](#), [ADC0_PATH](#), [ADC1](#), [ADC1_PATH](#), [ADC2](#), [ADC2_PATH](#), [ADC3](#), [ADC3_PATH](#), [ADC4](#), [ADC4_PATH](#), [ADC5](#), [ADC5_PATH](#), [ADC6](#), [ADC6_PATH](#), [ADC7](#), [ADC7_PATH](#), [adcpath](#), [MaxIntensity](#), [MinIntensity](#), and [Pin](#).

6.2.3.2 ADC::~ADC ()

De-constructor

Definition at line 48 of file [ADC.cpp](#).

6.2.4 Member Function Documentation

6.2.4.1 int ADC::GetCurrentValue()

Reads the current voltage in the pin

Returns

an integer between 0 and 4096

Definition at line 53 of file [ADC.cpp](#).

References [adcpath](#), [Intensity](#), [MaxIntensity](#), [MinIntensity](#), and [Hardware::BBB::Read\(\)](#).

Here is the call graph for this function:



6.2.4.2 float Hardware::ADC::GetIntensity() [inline]

Definition at line 71 of file [ADC.h](#).

6.2.4.3 int Hardware::ADC::GetMaxIntensity() [inline]

Definition at line 73 of file [ADC.h](#).

6.2.4.4 int Hardware::ADC::GetMinIntensity() [inline]

Definition at line 72 of file [ADC.h](#).

6.2.4.5 void ADC::SetMaxIntensity()

Definition at line 65 of file [ADC.cpp](#).

References [adcpath](#), [MaxIntensity](#), and [Hardware::BBB::Read\(\)](#).

Here is the call graph for this function:



6.2.4.6 void ADC::SetMinIntensity()

Set the current voltage at the pin as the minimum voltage

Definition at line 61 of file [ADC.cpp](#).

References [adcpath](#), [MinIntensity](#), and [Hardware::BBB::Read\(\)](#).

Here is the call graph for this function:



6.2.4.7 int ADC::WaitForValueChange ()

Polling of the analogue pin

Returns

the current value

Definition at line 88 of file [ADC.cpp](#).

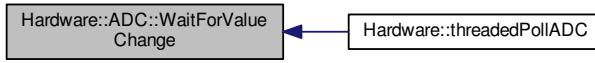
References [adcpPath](#), and [Hardware::BBB::Read\(\)](#).

Referenced by [Hardware::threadedPollADC\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.4.8 int ADC::WaitForValueChange (CallbackType *callback*)

Threading enabled polling of the analogue pin

Parameters

<i>callback</i>	the function which should be called when polling indicates a change
-----------------	---

Returns

0

Definition at line 74 of file [ADC.cpp](#).

References [Hardware::BBB::callbackFunction](#), [Hardware::BBB::thread](#), [threadedPollADC](#), and [Hardware::BBB::threadRunning](#).

6.2.4.9 void Hardware::ADC::WaitForValueChangeCancel () [inline]

Definition at line 80 of file [ADC.h](#).

6.2.5 Friends And Related Function Documentation

6.2.5.1 `void* threadedPollADC (void * value) [friend]`

friend polling function

friendly function to start the threading

Definition at line 121 of file [ADC.cpp](#).

Referenced by [WaitForValueChange\(\)](#).

6.2.6 Member Data Documentation

6.2.6.1 `string Hardware::ADC::adcpath [private]`

Path to analogue write file

Definition at line 83 of file [ADC.h](#).

Referenced by [ADC\(\)](#), [GetCurrentValue\(\)](#), [SetMaxIntensity\(\)](#), [SetMinIntensity\(\)](#), and [WaitForValueChange\(\)](#).

6.2.6.2 `float Hardware::ADC::Intensity [private]`

Current intensity expressed as percentage

Definition at line 84 of file [ADC.h](#).

Referenced by [GetCurrentValue\(\)](#).

6.2.6.3 `int Hardware::ADC::MaxIntensity [private]`

Voltage level which represent 100 percentage

Definition at line 86 of file [ADC.h](#).

Referenced by [ADC\(\)](#), [GetCurrentValue\(\)](#), and [SetMaxIntensity\(\)](#).

6.2.6.4 `int Hardware::ADC::MinIntensity [private]`

Voltage level which represent 0 percentage

Definition at line 85 of file [ADC.h](#).

Referenced by [ADC\(\)](#), [GetCurrentValue\(\)](#), and [SetMinIntensity\(\)](#).

6.2.6.5 `ADCPin Hardware::ADC::Pin`

current pin

Definition at line 65 of file [ADC.h](#).

Referenced by [ADC\(\)](#).

The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/[ADC.h](#)
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/[ADC.cpp](#)

6.3 ADC Class Reference

```
#include <ADC.h>
```

Collaboration diagram for ADC:



6.3.1 Detailed Description

Interaction with the beaglebone analogue pins

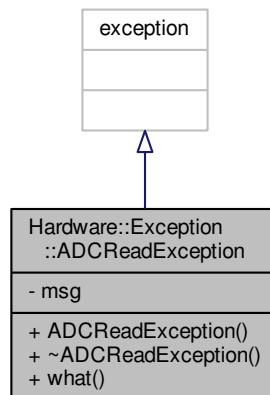
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADC.h](#)

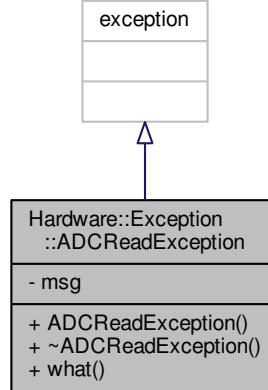
6.4 Hardware::Exception::ADCReadException Class Reference

```
#include <ADCReadException.h>
```

Inheritance diagram for Hardware::Exception::ADCReadException:



Collaboration diagram for Hardware::Exception::ADCReadException:



Public Member Functions

- `ADCReadException` (string m="Can't read **ADC** data!")
- `~ADCReadException` () `_GLIBCXX_USE_NOEXCEPT`
- `const char * what` () `const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- `string msg`

6.4.1 Detailed Description

Definition at line 16 of file [ADCReadException.h](#).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `Hardware::Exception::ADCReadException (string m = "Can't read ADC data!")` [inline]

Definition at line 18 of file [ADCReadException.h](#).

6.4.2.2 `Hardware::Exception::ADCReadException::~ADCReadException ()` [inline]

Definition at line 19 of file [ADCReadException.h](#).

6.4.3 Member Function Documentation

6.4.3.1 `const char* Hardware::Exception::ADCReadException::what () const` [inline]

Definition at line 20 of file [ADCReadException.h](#).

6.4.4 Member Data Documentation

6.4.4.1 `string Hardware::Exception::ADCReadException::msg` [private]

Definition at line 20 of file [ADCReadException.h](#).

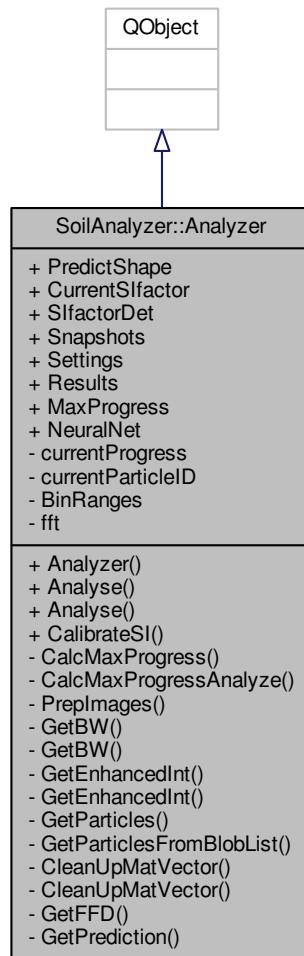
The documentation for this class was generated from the following file:

- `/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADCReadException.h`

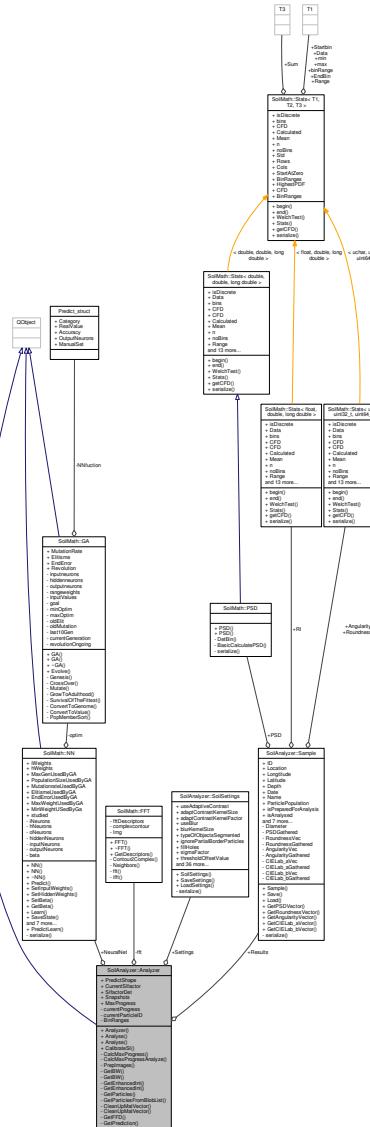
6.5 SoilAnalyzer::Analyzer Class Reference

```
#include <analyzer.h>
```

Inheritance diagram for SoilAnalyzer::Analyzer:



Collaboration diagram for SoilAnalyzer::Analyzer::



Classes

- struct `Image_t`

Public Types

- `typedef std::vector< Image_t > Images_t`

Signals

- void **on_progressUpdate** (int value)
 - void **on_maxProgressUpdate** (int value)
 - void **on_AnalysisFinished** ()

Public Member Functions

- `Analyzer (Images_t *snapshots, Sample *results, SoilSettings *settings)`
`Analyzer::Analyzer.`
- `void Analyse ()`
`Analyzer::Analyse.`
- `void Analyse (Images_t *snapshots, Sample *results, SoilSettings *settings)`
- `float CalibrateSI (float SI, cv::Mat &img)`

Public Attributes

- `bool PredictShape = true`
- `float CurrentSIfactor = 0.0111915`
- `bool SIfactorDet = false`
- `Images_t * Snapshots = nullptr`
- `SoilSettings * Settings = nullptr`
- `Sample * Results`
- `uint32_t MaxProgress = STARTING_ESTIMATE_PROGRESS`
- `SoilMath::NN NeuralNet`

Private Member Functions

- `void CalcMaxProgress ()`
`Analyzer::CalcMaxProgress.`
- `void CalcMaxProgressAnalyze ()`
- `void Preplimages ()`
`Analyzer::Preplimages.`
- `void GetBW (std::vector< cv::Mat > &images, std::vector< cv::Mat > &BWvector)`
`Analyzer::GetBW.`
- `void GetBW (cv::Mat &img, cv::Mat &BW)`
`Analyzer::GetBW.`
- `void GetEnhancedInt (Images_t *snapshots, std::vector< cv::Mat > &intensityVector)`
- `void GetEnhancedInt (cv::Mat &img, cv::Mat &intensity)`
- `void GetParticles (std::vector< cv::Mat > &BW, Images_t *snapshots, Particle::ParticleVector_t &partPopulation)`
`Analyzer::GetParticles.`
- `void GetParticlesFromBlobList (Vision::Segment::BlobList_t &bloblist, Image_t *snapshot, Particle::ParticleVector_t &partPopulation)`
`Analyzer::GetParticlesFromBlobList.`
- `void CleanUpMatVector (std::vector< cv::Mat > &mv)`
- `void CleanUpMatVector (Images_t *mv)`
`Analyzer::CleanUpMatVector.`
- `void GetFFD (Particle::ParticleVector_t &particalPopulation)`
`Analyzer::GetFFD.`
- `void GetPrediction (Particle::ParticleVector_t &particlePopulation)`
`Analyzer::GetPrediction.`

Private Attributes

- `uint32_t currentProgress = 0`
- `uint32_t currentParticleID = 0`
- `double BinRanges [15]`
- `SoilMath::FFT fft`

6.5.1 Detailed Description

Definition at line 32 of file [analyzer.h](#).

6.5.2 Member Typedef Documentation

6.5.2.1 `typedef std::vector<Image_t> SoilAnalyzer::Analyzer::Images_t`

Definition at line 45 of file [analyzer.h](#).

6.5.3 Constructor & Destructor Documentation

6.5.3.1 `SoilAnalyzer::Analyzer (Images_t * snapshots, Sample * results, SoilSettings * settings = nullptr)`

Analyzer::Analyzer.

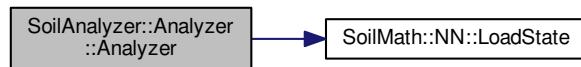
Parameters

<i>snapshots</i>	
<i>results</i>	
<i>settings</i>	

Definition at line 18 of file [analyzer.cpp](#).

References `SoilMath::NN::LoadState()`, `NeuralNet`, `SoilAnalyzer::SoilSettings::NNlocation`, `Results`, `Settings`, and `Snapshots`.

Here is the call graph for this function:



6.5.4 Member Function Documentation

6.5.4.1 void SoilAnalyzer::Analyzer::Analyse ()

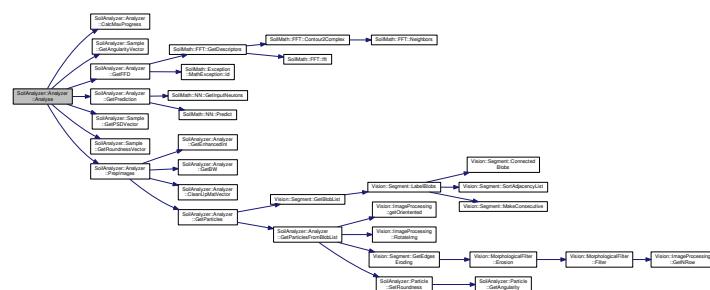
Analyzer::Analyse.

Definition at line 65 of file [analyzer.cpp](#).

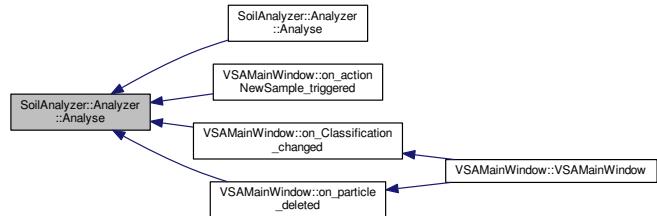
References SoilAnalyzer::Sample::Angularity, BinRanges, CalcMaxProgress(), currentProgress, SoilAnalyzer::Sample::GetAngularityVector(), GetFFD(), GetPrediction(), SoilAnalyzer::Sample::GetPSDVector(), SoilAnalyzer::Sample::GetRoundnessVector(), SoilAnalyzer::Sample::IsLoadedFromDisk, SoilAnalyzer::Sample::isPreparedForAnalysis, on_AnalysisFinished(), on_progressUpdate(), SoilAnalyzer::Sample::ParticlePopulation, PredictShape, SoilAnalyzer::SoilSettings::PredictTheShape, PreplImages(), SoilAnalyzer::Sample::PSD, Results, SoilAnalyzer::Sample::Roundness, and Settings.

Referenced by [Analyse\(\)](#), [VSAMainWindow::on_actionNewSample_triggered\(\)](#), [VSAMainWindow::on_Classification_changed\(\)](#), and [VSA←MainWindow::on_particle_deleted\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

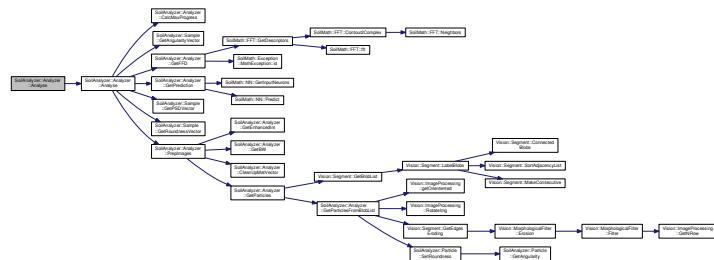


6.5.4.2 void SoilAnalyzer::Analyzer::Analyse (Images_t * snapshots, Sample * results, SoilSettings * settings)

Definition at line 54 of file [analyzer.cpp](#).

References [Analyse\(\)](#), [Results](#), [Settings](#), and [Snapshots](#).

Here is the call graph for this function:



6.5.4.3 void SoilAnalyzer::Analyzer::CalcMaxProgress() [private]

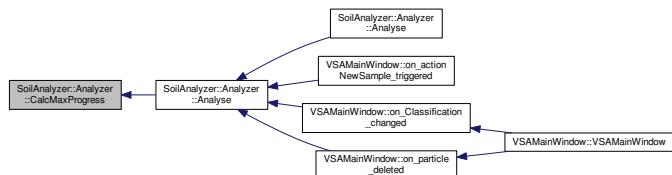
Analyzer::CalcMaxProgress.

Definition at line 112 of file [analyzer.cpp](#).

References `SoilAnalyzer::SoilSettings::fillHoles`, `SoilAnalyzer::SoilSettings::ignorePartialBorderParticles`, `MaxProgress`, `SoilAnalyzer::SoilSettings::morphFilterType`, `Vision::MorphologicalFilter::NONE`, `on_maxProgressUpdate()`, `Settings`, `Snapshots`, `SoilAnalyzer::SoilSettings::useAdaptiveContrast`, and `SoilAnalyzer::SoilSettings::useBlur`.

Referenced by [Analyse\(\)](#).

Here is the caller graph for this function:



6.5.4.4 void SoilAnalyzer::Analyzer::CalcMaxProgressAnalyze () [private]

Definition at line 136 of file [analyzer.cpp](#).

References MaxProgress, on_maxProgressUpdate(), SoilAnalyzer::Sample::ParticlePopulation, Results, and STARTING_ESTIMATE_PROGRESS.

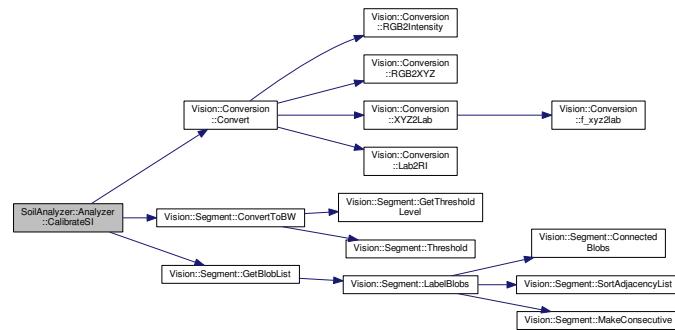
6.5.4.5 float SoilAnalyzer::Analyzer::CalibrateSI (float *SI*, cv::Mat & *img*)

Definition at line 388 of file [analyzer.cpp](#).

References [Vision::Segment::BlobList](#), [Vision::Conversion::Convert\(\)](#), [Vision::Segment::ConvertToBW\(\)](#), [CurrentSIfactor](#), [Vision::Segment::Dark](#), [Vision::Segment::GetBlobList\(\)](#), [Vision::Conversion::Intensity](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::Conversion::RGB](#).

Referenced by [VSAMainWindow::on_actionCalibrate_triggered\(\)](#).

Here is the call graph for this function:



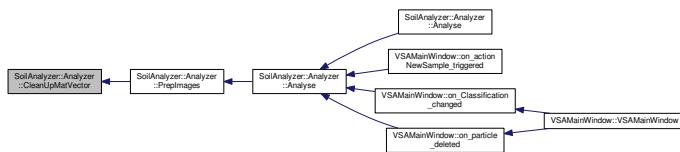
Here is the caller graph for this function:



6.5.4.6 void SoilAnalyzer::Analyzer::CleanUpMatVector (std::vector< cv::Mat > & *mv*) [private]

Referenced by [PreplImages\(\)](#).

Here is the caller graph for this function:



6.5.4.7 void SoilAnalyzer::Analyzer::CleanUpMatVector (Images_t * *mv*) [private]

[Analyzer::CleanUpMatVector](#).

Parameters

<i>mv</i>

Definition at line 101 of file [analyzer.cpp](#).

6.5.4.8 void SoilAnalyzer::Analyzer::GetBW (std::vector< cv::Mat > & *images*, std::vector< cv::Mat > & *BWvector*) [private]

[Analyzer::GetBW](#).

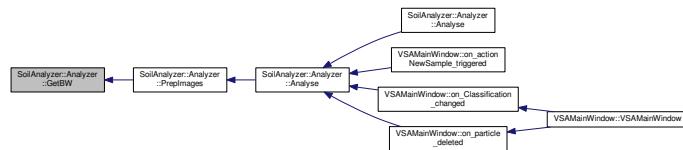
Parameters

<i>images</i>	
<i>BWvector</i>	

Definition at line 222 of file [analyzer.cpp](#).

Referenced by [Preplimages\(\)](#).

Here is the caller graph for this function:



6.5.4.9 void [SoilAnalyzer::Analyzer::GetBW \(cv::Mat & img, cv::Mat & BW \)](#) [private]

[Analyzer::GetBW](#).

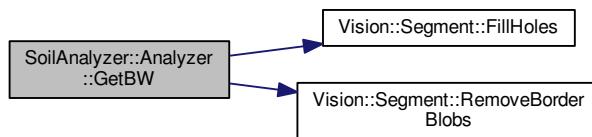
Parameters

<i>img</i>	
<i>BW</i>	

Definition at line 236 of file [analyzer.cpp](#).

References [Vision::MorphologicalFilter::CLOSE](#), [currentProgress](#), [Vision::MorphologicalFilter::DILATE](#), [Vision::MorphologicalFilter::EROD](#)←
[E](#), [SoilAnalyzer::SoilSettings::fillHoles](#), [Vision::Segment::FillHoles\(\)](#), [SoilAnalyzer::SoilSettings::filterMaskSize](#), [SoilAnalyzer::SoilSettings::ignorePartialBorderParticles](#), [SoilAnalyzer::SoilSettings::morphFilterType](#), [Vision::MorphologicalFilter::NONE](#), [on_progressUpdate\(\)](#), [Vision::MorphologicalFilter::OPEN](#), [Vision::ImageProcessing::ProcessedImg](#), [Vision::Segment::RemoveBorderBlobs\(\)](#), [Settings](#), [SHOW_DEBUG_IMG](#), [Vision::Segment::sigma](#), [SoilAnalyzer::SoilSettings::sigmaFactor](#), [SoilAnalyzer::SoilSettings::thresholdOffsetValue](#), and [SoilAnalyzer::SoilSettings::typeOfObjectsSegmented](#).

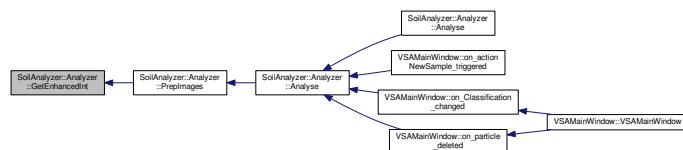
Here is the call graph for this function:



6.5.4.10 void [SoilAnalyzer::Analyzer::GetEnhancedInt \(Images_t * snapshots, std::vector< cv::Mat > & intensityVector \)](#) [private]

Referenced by [Preplimages\(\)](#).

Here is the caller graph for this function:



- 6.5.4.11 void SoilAnalyzer::Analyzer::GetEnhancedInt (cv::Mat & img, cv::Mat & intensity) [private]
- 6.5.4.12 void SoilAnalyzer::Analyzer::GetFFD (Particle::ParticleVector_t & particlePopulation) [private]

Analyzer::GetFFD.

Parameters

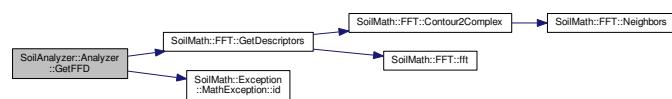
particalPopulation

Definition at line 350 of file [analyzer.cpp](#).

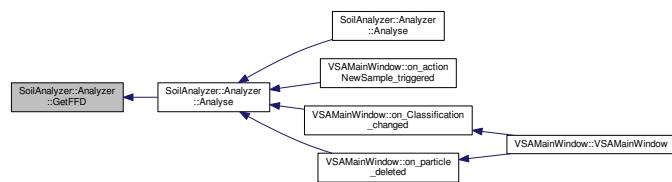
References `currentProgress`, `EXCEPTION_NO_CONTOUR_FOUND_NR`, `fft`, `SoilMath::FFT::GetDescriptors()`, `SoilMath::Exception::MathException::id()`, and `on_progressUpdate()`.

Referenced by [Analyse\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.13 void SoilAnalyzer::Analyzer::GetParticles (std::vector< cv::Mat > & BW, Images_t * snapshots, Particle::ParticleVector_t & partPopulation) [private]

Analyzer::GetParticles.

Parameters

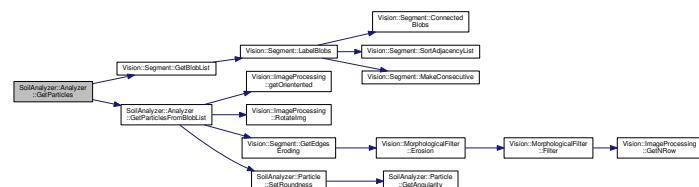
BW	
$snapshots$	
$partPopulation$	

Definition at line 303 of file [analyzer.cpp](#).

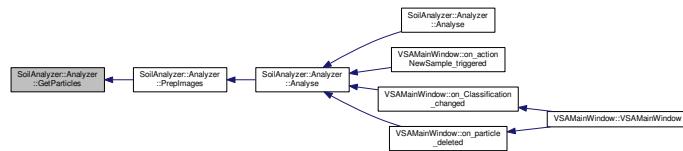
References `Vision::Segment::BlobList`, `currentProgress`, `Vision::Segment::GetBlobList()`, `GetParticlesFromBlobList()`, and `on_progressUpdate()`.

Referenced by [PreImages\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.14 void SoilAnalyzer::Analyzer::GetParticlesFromBlobList (Vision::Segment::BlobList_t & *bloblist*, Image_t * *snapshot*, Particle::ParticleVector_t & *partPopulation*) [private]

[Analyzer::GetParticlesFromBlobList](#).

Parameters

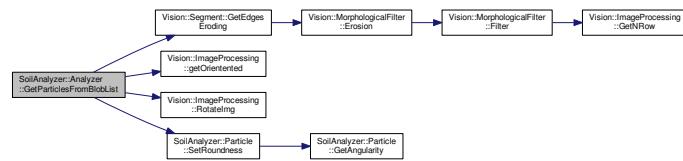
<i>bloblist</i>	
<i>snapshot</i>	
<i>edge</i>	
<i>partPopulation</i>	

Definition at line 322 of file [analyzer.cpp](#).

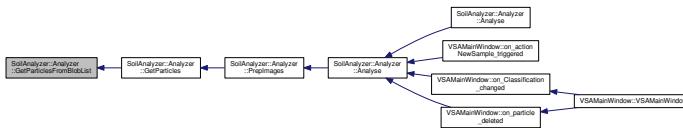
References [SoilAnalyzer::Particle::BW](#), [currentParticleID](#), [SoilAnalyzer::Particle::Eccentricity](#), [SoilAnalyzer::Particle::Edge](#), [SoilAnalyzer::Analyzer::Image_t::FrontLight](#), [Vision::Segment::GetEdgesEroding\(\)](#), [Vision::ImageProcessing::getOriented\(\)](#), [SoilAnalyzer::Particle::ID](#), [SoilAnalyzer::Particle::isPreparedForAnalysis](#), [SoilAnalyzer::Particle::PixelArea](#), [Vision::ImageProcessing::ProcessedImg](#), [SoilAnalyzer::Particle::RGB](#), [Vision::ImageProcessing::RotateImg\(\)](#), [SoilAnalyzer::Particle::SetRoundness\(\)](#), [SoilAnalyzer::Analyzer::Image_t::SIPixelFactor](#), and [SoilAnalyzer::Particle::SIPixelFactor](#).

Referenced by [GetParticles\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.15 void SoilAnalyzer::Analyzer::GetPrediction (Particle::ParticleVector_t & *particlePopulation*) [private]

[Analyzer::GetPrediction](#).

Parameters

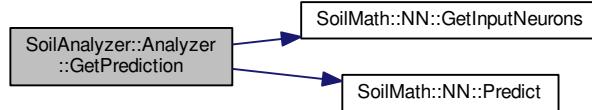
<i>particlePopulation</i>	
---------------------------	--

Definition at line 373 of file [analyzer.cpp](#).

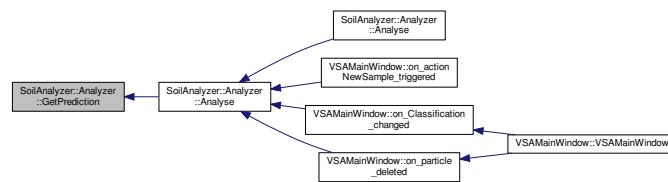
References [SoilMath::NN::GetInputNeurons\(\)](#), [NeuralNet](#), and [SoilMath::NN::Predict\(\)](#).

Referenced by [Analyse\(\)](#).

Here is the call graph for this function:



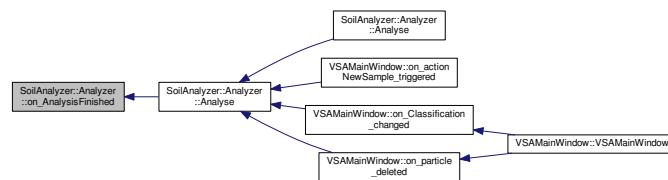
Here is the caller graph for this function:



6.5.4.16 void SoilAnalyzer::Analyzer::on_AnalysisFinished() [signal]

Referenced by [Analyse\(\)](#).

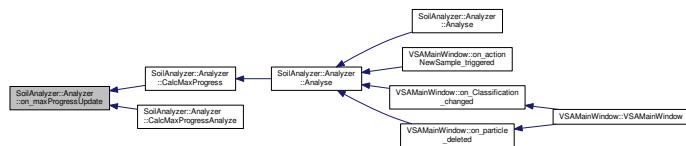
Here is the caller graph for this function:



6.5.4.17 void SoilAnalyzer::Analyzer::on_maxProgressUpdate(int value) [signal]

Referenced by [CalcMaxProgress\(\)](#), and [CalcMaxProgressAnalyze\(\)](#).

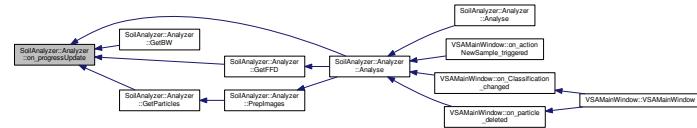
Here is the caller graph for this function:



6.5.4.18 void SoilAnalyzer::Analyzer::on_progressUpdate(int value) [signal]

Referenced by [Analyse\(\)](#), [GetBW\(\)](#), [GetFFD\(\)](#), and [GetParticles\(\)](#).

Here is the caller graph for this function:



6.5.4.19 void SoilAnalyzer::Analyzer::PreplImages() [private]

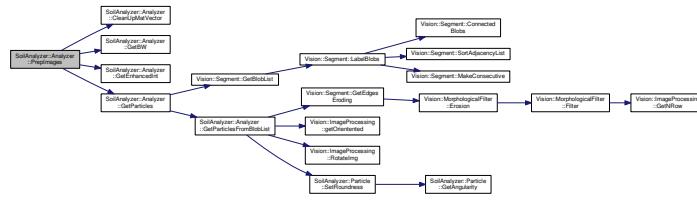
Analyzer::PreplImages.

Definition at line 33 of file [analyzer.cpp](#).

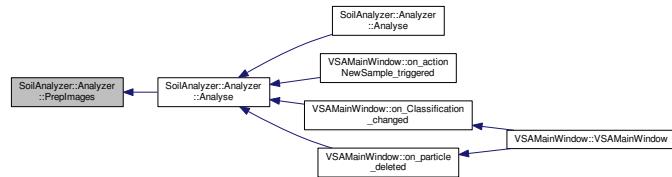
References [CleanUpMatVector\(\)](#), [EXCEPTION_NO_SNAPSHOTS](#), [EXCEPTION_NO_SNAPSHOTS_NR](#), [GetBW\(\)](#), [GetEnhancedInt\(\)](#), [GetParticles\(\)](#), [SoilAnalyzer::Sample::isPreparedForAnalysis](#), [SoilAnalyzer::Sample::ParticlePopulation](#), [Results](#), and [Snapshots](#).

Referenced by [Analyse\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.5 Member Data Documentation

6.5.5.1 double SoilAnalyzer::Analyzer::BinRanges[15] [private]

Initial value:

```
{0.0, 0.038, 0.045, 0.063, 0.075, 0.09, 0.125, 0.18, 0.25, 0.355, 0.5, 0.71, 1.0, 1.4, 2.0}
```

Definition at line 69 of file [analyzer.h](#).

Referenced by [Analyse\(\)](#).

6.5.5.2 uint32_t SoilAnalyzer::Analyzer::currentParticleID = 0 [private]

Definition at line 68 of file [analyzer.h](#).

Referenced by [GetParticlesFromBlobList\(\)](#).

6.5.5.3 `uint32_t SoilAnalyzer::Analyzer::currentProgress = 0` [private]

Definition at line 67 of file [analyzer.h](#).

Referenced by [Analyse\(\)](#), [GetBW\(\)](#), [GetFFD\(\)](#), and [GetParticles\(\)](#).

6.5.5.4 `float SoilAnalyzer::Analyzer::CurrentSlfactor = 0.0111915`

Definition at line 37 of file [analyzer.h](#).

Referenced by [CalibrateSI\(\)](#), and [VSAMainWindow::TakeSnapShots\(\)](#).

6.5.5.5 `SoilMath::FFT SoilAnalyzer::Analyzer::fft` [private]

Definition at line 72 of file [analyzer.h](#).

Referenced by [GetFFD\(\)](#).

6.5.5.6 `uint32_t SoilAnalyzer::Analyzer::MaxProgress = STARTING_ESTIMATE_PROGRESS`

Definition at line 57 of file [analyzer.h](#).

Referenced by [CalcMaxProgress\(\)](#), [CalcMaxProgressAnalyze\(\)](#), and [VSAMainWindow::VSAMainWindow\(\)](#).

6.5.5.7 `SoilMath::NN SoilAnalyzer::Analyzer::NeuralNet`

Definition at line 59 of file [analyzer.h](#).

Referenced by [Analyzer\(\)](#), [GetPrediction\(\)](#), [VSAMainWindow::on_actionNeuralNet_triggered\(\)](#), and [VSAMainWindow::VSAMainWindow\(\)](#).

6.5.5.8 `bool SoilAnalyzer::Analyzer::PredictShape = true`

Definition at line 36 of file [analyzer.h](#).

Referenced by [Analyse\(\)](#), and [VSAMainWindow::on_actionUseLearning_toggled\(\)](#).

6.5.5.9 `Sample* SoilAnalyzer::Analyzer::Results`

Definition at line 49 of file [analyzer.h](#).

Referenced by [Analyse\(\)](#), [Analyzer\(\)](#), [CalcMaxProgressAnalyze\(\)](#), [VSAMainWindow::on_actionLoadSample_triggered\(\)](#), and [PreplImages\(\)](#).

6.5.5.10 `SoilSettings* SoilAnalyzer::Analyzer::Settings = nullptr`

Definition at line 47 of file [analyzer.h](#).

Referenced by [Analyse\(\)](#), [Analyzer\(\)](#), [CalcMaxProgress\(\)](#), and [GetBW\(\)](#).

6.5.5.11 `bool SoilAnalyzer::Analyzer::SlfactorDet = false`

Definition at line 38 of file [analyzer.h](#).

Referenced by [VSAMainWindow::TakeSnapShots\(\)](#).

6.5.5.12 `Images_t* SoilAnalyzer::Analyzer::S snapshots = nullptr`

Definition at line 46 of file [analyzer.h](#).

Referenced by [Analyse\(\)](#), [Analyzer\(\)](#), [CalcMaxProgress\(\)](#), and [PreplImages\(\)](#).

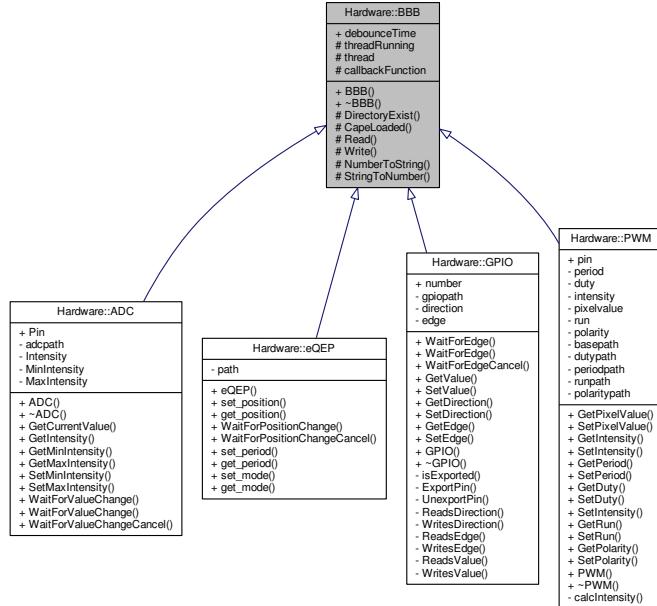
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.cpp](#)

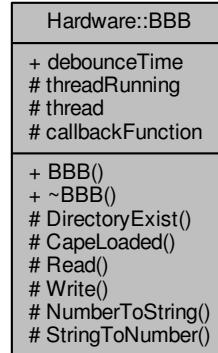
6.6 Hardware::BBB Class Reference

```
#include <BBB.h>
```

Inheritance diagram for Hardware::BBB:



Collaboration diagram for Hardware::BBB:



Public Member Functions

- **BBB ()**
- **~BBB ()**

Public Attributes

- int **debounceTime**

Protected Member Functions

- bool [DirectoryExist](#) (const string &path)
- bool [CapeLoaded](#) (const string &shield)
- string [Read](#) (const string &path)
- void [Write](#) (const string &path, const string &value)
- template<typename T >
string [NumberToString](#) (T Number)
- template<typename T >
T [StringToNumber](#) (string Text)

Protected Attributes

- bool [threadRunning](#)
- pthread_t [thread](#)
- [CallbackType](#) [callbackFunction](#)

6.6.1 Detailed Description

Definition at line 40 of file [BBB.h](#).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 BBB::BBB()

Constructor

Definition at line 12 of file [BBB.cpp](#).

References [callbackFunction](#), [debounceTime](#), [thread](#), and [threadRunning](#).

6.6.2.2 BBB::~BBB()

De-constructor

Definition at line 20 of file [BBB.cpp](#).

6.6.3 Member Function Documentation

6.6.3.1 bool BBB::CapeLoaded(const string & shield) [protected]

Checks if a cape is loaded in the file /sys/devices/bone_capemgr.9/slots

Parameters

<i>shield</i>	a const search string which is a (part) of the shield name
---------------	--

Returns

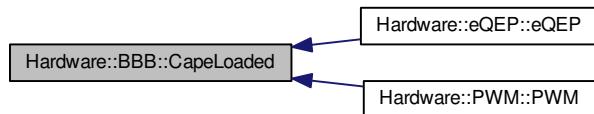
true if the search string is found otherwise false

Definition at line 67 of file [BBB.cpp](#).

References [SLOTS](#).

Referenced by [Hardware::eQEP::eQEP\(\)](#), and [Hardware::PWM::PWM\(\)](#).

Here is the caller graph for this function:



6.6.3.2 bool BBB::DirectoryExist (const string & path) [protected]

Checks if a directory exist

Returns

true if the directory exists and false if not

Definition at line 55 of file [BBB.cpp](#).

Referenced by [Hardware::GPIO::isExported\(\)](#).

Here is the caller graph for this function:



6.6.3.3 template<typename T> string Hardware::BBB::NumberToString (T Number) [inline], [protected]

Converts a number to a string

Parameters

Number	as typename
--------	-------------

Returns

the number as a string

Definition at line 62 of file [BBB.h](#).

6.6.3.4 string BBB::Read (const string & path) [protected]

Reads the first line from a file

Parameters

path	constant string pointing towards the file
------	---

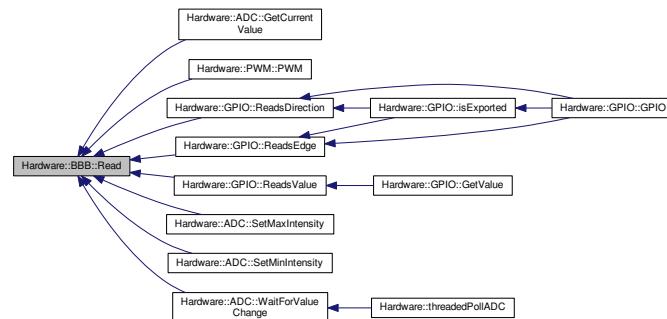
Returns

this first line

Definition at line 26 of file [BBB.cpp](#).

Referenced by [Hardware::ADC::GetCurrentValue\(\)](#), [Hardware::PWM::PWM\(\)](#), [Hardware::GPIO::ReadsDirection\(\)](#), [Hardware::GPIO::ReadsEdge\(\)](#), [Hardware::GPIO::ReadsValue\(\)](#), [Hardware::ADC::SetMaxIntensity\(\)](#), [Hardware::ADC::SetMinIntensity\(\)](#), and [Hardware::ADC::WaitForValueChange\(\)](#).

Here is the caller graph for this function:



6.6.3.5 `template<typename T> T Hardware::BBB::StringToNumber(string Text) [inline], [protected]`

Converts a string to a number

Parameters

Text	the string that needs to be converted
------	---------------------------------------

Returns

the number as typename

Definition at line 72 of file [BBB.h](#).

6.6.3.6 void BBB::Write (const string & path, const string & value) [protected]

Writes a value to a file

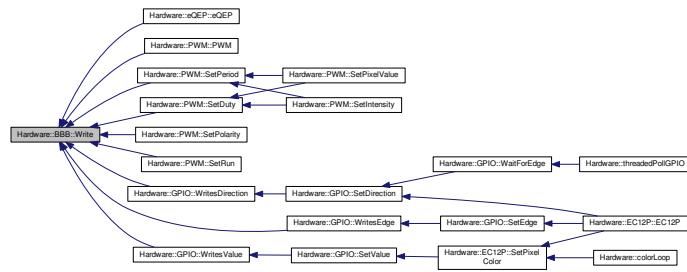
Parameters

<i>path</i>	a constant string pointing towards the file
<i>value</i>	a constant string which should be written in the file

Definition at line 42 of file [BBB.cpp](#).

Referenced by [Hardware::eQEP::eQEP\(\)](#), [Hardware::PWM::PWM\(\)](#), [Hardware::PWM::SetDuty\(\)](#), [Hardware::PWM::SetPeriod\(\)](#), [Hardware::PWM::SetPolarity\(\)](#), [Hardware::PWM::SetRun\(\)](#), [Hardware::GPIO::WritesDirection\(\)](#), [Hardware::GPIO::WritesEdge\(\)](#), and [Hardware::GPIO::WritesValue\(\)](#).

Here is the caller graph for this function:



6.6.4 Member Data Documentation

6.6.4.1 CallbackType Hardware::BBB::callbackFunction [protected]

the callbackfunction

Definition at line 50 of file [BBB.h](#).

Referenced by [BBB\(\)](#), [Hardware::threadedPollADC\(\)](#), [Hardware::threadedPolleqep\(\)](#), [Hardware::threadedPollGPIO\(\)](#), [Hardware::GPIO::WaitForEdge\(\)](#), [Hardware::eQEP::WaitForPositionChange\(\)](#), and [Hardware::ADC::WaitForValueChange\(\)](#).

6.6.4.2 int Hardware::BBB::debounceTime

debounce time for a button in milliseconds

Definition at line 42 of file [BBB.h](#).

Referenced by [BBB\(\)](#), [Hardware::threadedPolleqep\(\)](#), and [Hardware::threadedPollGPIO\(\)](#).

6.6.4.3 pthread_t Hardware::BBB::thread [protected]

The thread

Definition at line 49 of file [BBB.h](#).

Referenced by [BBB\(\)](#), [Hardware::GPIO::WaitForEdge\(\)](#), [Hardware::eQEP::WaitForPositionChange\(\)](#), and [Hardware::ADC::WaitForValueChange\(\)](#).

6.6.4.4 bool Hardware::BBB::threadRunning [protected]

used to stop the thread

Definition at line 48 of file [BBB.h](#).

Referenced by [BBB\(\)](#), [Hardware::threadedPollADC\(\)](#), [Hardware::threadedPolleqep\(\)](#), [Hardware::threadedPollGPIO\(\)](#), [Hardware::GPIO::WaitForEdge\(\)](#), [Hardware::eQEP::WaitForPositionChange\(\)](#), [Hardware::eQEP::WaitForPositionChangeCancel\(\)](#), and [Hardware::ADC::WaitForValueChange\(\)](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.cpp](#)

6.7 BBB Class Reference

```
#include <BBB.h>
```

Collaboration diagram for BBB:



6.7.1 Detailed Description

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

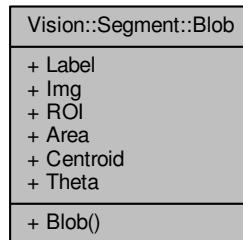
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.h](#)

6.8 Vision::Segment::Blob Struct Reference

```
#include <Segment.h>
```

Collaboration diagram for Vision::Segment::Blob:



Public Member Functions

- [Blob \(uint16_t label, uint32_t area\)](#)

Public Attributes

- `uint16_t Label`
- `cv::Mat Img`
- `cv::Rect ROI`
- `uint32_t Area`
- `cv::Point_< double > Centroid`
- `double Theta`

6.8.1 Detailed Description

Individual blob

Definition at line 42 of file [Segment.h](#).

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `Vision::Segment::Blob(uint16_t label, uint32_t area)` [inline]

Definition at line 51 of file [Segment.h](#).

6.8.3 Member Data Documentation

6.8.3.1 `uint32_t Vision::Segment::Blob::Area`

Calculated stats of the blob

Definition at line 48 of file [Segment.h](#).

6.8.3.2 `cv::Point_<double> Vision::Segment::Blob::Centroid`

Definition at line 49 of file [Segment.h](#).

6.8.3.3 `cv::Mat Vision::Segment::Blob::Img`

BW image of the blob all the pixel belonging to the blob are set to 1 others are 0

Definition at line 44 of file [Segment.h](#).

6.8.3.4 `uint16_t Vision::Segment::Blob::Label`

ID of the blob

Definition at line 43 of file [Segment.h](#).

6.8.3.5 `cv::Rect Vision::Segment::Blob::ROI`

Coordinates for the blob in the original picture as a `cv::Rect`

Definition at line 46 of file [Segment.h](#).

6.8.3.6 `double Vision::Segment::Blob::Theta`

Definition at line 50 of file [Segment.h](#).

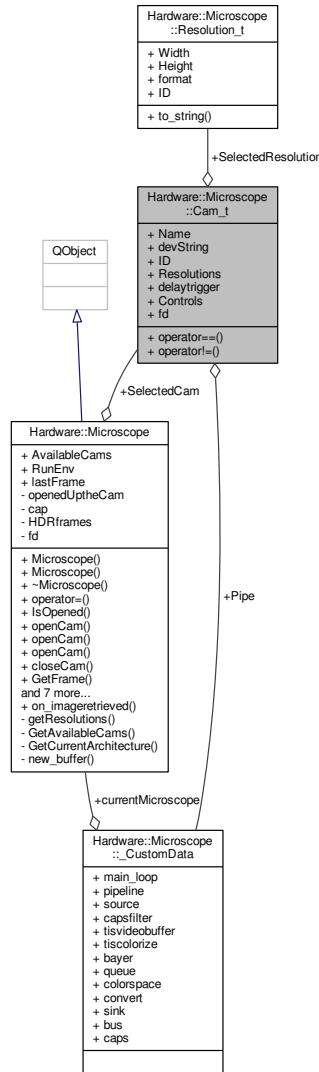
The documentation for this struct was generated from the following file:

- `/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.h`

6.9 Hardware::Microscope::Cam_t Struct Reference

```
#include <Microscope.h>
```

Collaboration diagram for Hardware::Microscope::Cam_t:



Public Member Functions

- `bool operator== (Cam_t const &rhs)`
- `bool operator!= (Cam_t const &rhs)`

Public Attributes

- `std::string Name`
- `std::string devString`
- `uint32_t ID`
- `std::vector< Resolution_t > Resolutions`
- `uint32_t delaytrigger = 1`
- `Resolution_t * SelectedResolution = nullptr`
- `Controls_t Controls`
- `CustomData Pipe`
- `int fd`

6.9.1 Detailed Description

Definition at line 122 of file [Microscope.h](#).

6.9.2 Member Function Documentation

6.9.2.1 `bool Hardware::Microscope::Cam_t::operator!= (Cam_t const & rhs) [inline]`

Definition at line 139 of file [Microscope.h](#).

References [ID](#), and [Name](#).

6.9.2.2 `bool Hardware::Microscope::Cam_t::operator== (Cam_t const & rhs) [inline]`

Definition at line 132 of file [Microscope.h](#).

References [ID](#), and [Name](#).

6.9.3 Member Data Documentation

6.9.3.1 `Controls_t Hardware::Microscope::Cam_t::Controls`

Definition at line 129 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [Hardware::Microscope::GetControl\(\)](#), [Hardware::Microscope::openCam\(\)](#), and [DialogSettings::SetCamControl\(\)](#).

6.9.3.2 `uint32_t Hardware::Microscope::Cam_t::delaytrigger = 1`

Definition at line 127 of file [Microscope.h](#).

6.9.3.3 `std::string Hardware::Microscope::Cam_t::devString`

Definition at line 124 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [Hardware::Microscope::openCam\(\)](#), and [Hardware::Microscope::SetControl\(\)](#).

6.9.3.4 `int Hardware::Microscope::Cam_t::fd`

Definition at line 131 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [Hardware::Microscope::getResolutions\(\)](#), and [Hardware::Microscope::SetControl\(\)](#).

6.9.3.5 `uint32_t Hardware::Microscope::Cam_t::ID`

Definition at line 125 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [operator!=\(\)](#), and [operator==\(\)](#).

6.9.3.6 `std::string Hardware::Microscope::Cam_t::Name`

Definition at line 123 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [Hardware::Microscope::openCam\(\)](#), [operator!=\(\)](#), and [operator==\(\)](#).

6.9.3.7 `CustomData Hardware::Microscope::Cam_t::Pipe`

Definition at line 130 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::closeCam\(\)](#), [Hardware::Microscope::GetFrame\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

6.9.3.8 `std::vector<Resolution_t> Hardware::Microscope::Cam_t::Resolutions`

Definition at line 126 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::getResolutions\(\)](#).

6.9.3.9 `Resolution_t* Hardware::Microscope::Cam_t::SelectedResolution = nullptr`

Definition at line 128 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::new_buffer\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

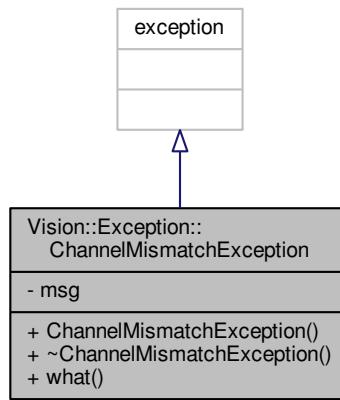
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h](#)

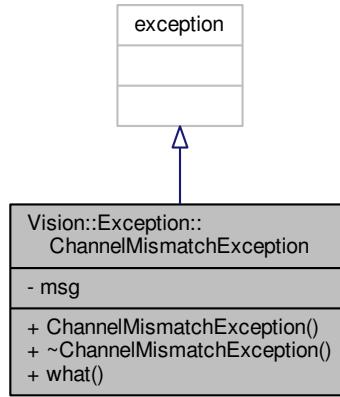
6.10 Vision::Exception::ChannelMismatchException Class Reference

#include <ChannelMismatchException.h>

Inheritance diagram for Vision::Exception::ChannelMismatchException:



Collaboration diagram for Vision::Exception::ChannelMismatchException:



Public Member Functions

- `ChannelMismatchException` (string m="Extracted channel out of bounds exception!")
- `~ChannelMismatchException` () `_GLIBCXX_USE_NOEXCEPT`
- `const char * what` () `const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- string `msg`

6.10.1 Detailed Description

Definition at line 21 of file [ChannelMismatchException.h](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `Vision::Exception::ChannelMismatchException::ChannelMismatchException (string m = "Extracted channel out of bounds exception!") [inline]`

Definition at line 23 of file [ChannelMismatchException.h](#).

6.10.2.2 `Vision::Exception::ChannelMismatchException::~ChannelMismatchException () [inline]`

Definition at line 26 of file [ChannelMismatchException.h](#).

6.10.3 Member Function Documentation

6.10.3.1 `const char* Vision::Exception::ChannelMismatchException::what () const [inline]`

Definition at line 27 of file [ChannelMismatchException.h](#).

6.10.4 Member Data Documentation

6.10.4.1 `string Vision::Exception::ChannelMismatchException::msg [private]`

Definition at line 27 of file [ChannelMismatchException.h](#).

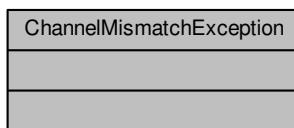
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ChannelMismatchException.h](#)

6.11 ChannelMismatchException Class Reference

`#include <ChannelMismatchException.h>`

Collaboration diagram for ChannelMismatchException:

**6.11.1 Detailed Description**

Exception class which is thrown when Extracted channel out of bounds exception

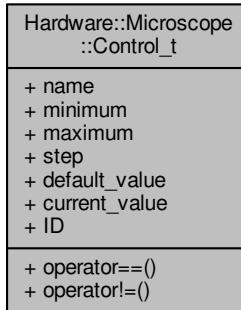
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ChannelMismatchException.h](#)

6.12 Hardware::Microscope::Control_t Struct Reference

#include <Microscope.h>

Collaboration diagram for Hardware::Microscope::Control_t:



Public Member Functions

- bool [operator== \(Control_t &rhs\)](#)
- bool [operator!= \(Control_t &rhs\)](#)

Public Attributes

- std::string [name](#)
- int [minimum](#)
- int [maximum](#)
- int [step](#)
- int [default_value](#)
- int [current_value](#)
- uint32_t [ID = V4L2_CID_BASE](#)

6.12.1 Detailed Description

Definition at line 79 of file [Microscope.h](#).

6.12.2 Member Function Documentation

6.12.2.1 bool Hardware::Microscope::Control_t::operator!= (Control_t & rhs) [inline]

Definition at line 94 of file [Microscope.h](#).

References [name](#).

6.12.2.2 bool Hardware::Microscope::Control_t::operator== (Control_t & rhs) [inline]

Definition at line 87 of file [Microscope.h](#).

References [name](#).

6.12.3 Member Data Documentation

6.12.3.1 int Hardware::Microscope::Control_t::current_value

Definition at line 85 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [Hardware::Microscope::GetHDRFrame\(\)](#), and [Hardware::Microscope::SetControl\(\)](#).

6.12.3.2 int Hardware::Microscope::Control_t::default_value

Definition at line 84 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#).

6.12.3.3 uint32_t Hardware::Microscope::Control_t::ID = V4L2_CID_BASE

Definition at line 86 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), and [Hardware::Microscope::SetControl\(\)](#).

6.12.3.4 int Hardware::Microscope::Control_t::maximum

Definition at line 82 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), and [Hardware::Microscope::GetHDRFrame\(\)](#).

6.12.3.5 int Hardware::Microscope::Control_t::minimum

Definition at line 81 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), and [Hardware::Microscope::GetHDRFrame\(\)](#).

6.12.3.6 std::string Hardware::Microscope::Control_t::name

Definition at line 80 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#), [operator!=\(\)](#), and [operator==\(\(\)\)](#).

6.12.3.7 int Hardware::Microscope::Control_t::step

Definition at line 83 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::GetAvailableCams\(\)](#).

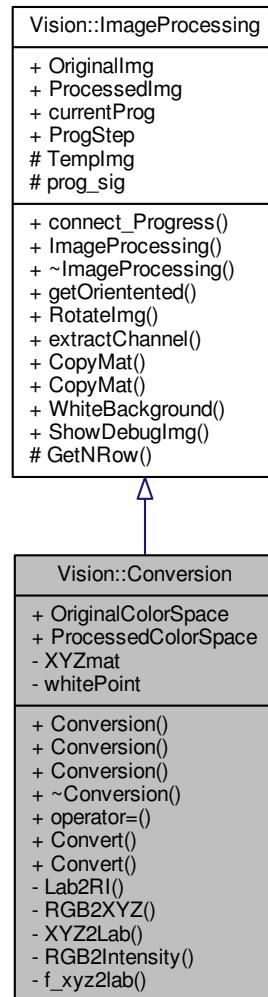
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/[Microscope.h](#)

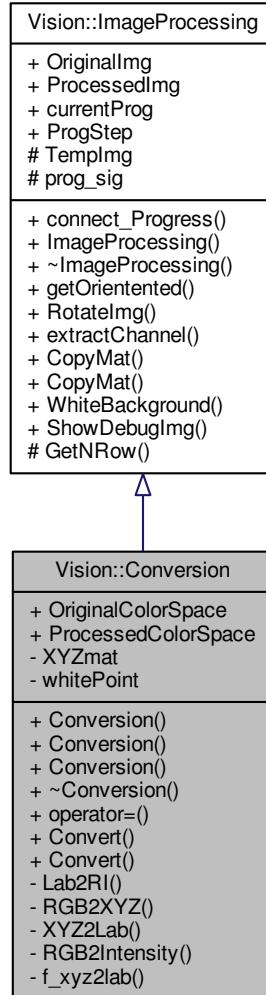
6.13 Vision::Conversion Class Reference

```
#include <Conversion.h>
```

Inheritance diagram for Vision::Conversion:



Collaboration diagram for Vision::Conversion:



Public Types

- enum `ColorSpace` {
 `CIE_lab`, `CIE_XYZ`, `RI`, `RGB`,
`Intensity`, `None` }

Public Member Functions

- `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)`

Public Attributes

- [ColorSpace OriginalColorSpace](#)
- [ColorSpace ProcessedColorSpace](#)

Private Member Functions

- void [Lab2RI](#) (float *O, float *P, int nData)
- void [RGB2XYZ](#) (uchar *O, float *P, int nData)
- void [XYZ2Lab](#) (float *O, float *P, int nData)
- void [RGB2Intensity](#) (uchar *O, uchar *P, int nData)
- float [f_xyz2lab](#) (float t)

Private Attributes

- float [XYZmat](#) [3][3]
- float [whitePoint](#) [3]

Additional Inherited Members

6.13.1 Detailed Description

Definition at line 13 of file [Conversion.h](#).

6.13.2 Member Enumeration Documentation

6.13.2.1 enum Vision::Conversion::ColorSpace

Enumerator which indicates the colorspace used

Enumerator

- CIE_lab** CIE La*b* colorspace
- CIE_XYZ** CIE XYZ colorspace
- RI** Redness Index colorspace
- RGB** RGB colorspace
- Intensity** Grayscale colorspace
- None** none

Definition at line 16 of file [Conversion.h](#).

6.13.3 Constructor & Destructor Documentation

6.13.3.1 Conversion::Conversion ()

Constructor of the class

Definition at line 14 of file [Conversion.cpp](#).

References [None](#), [OriginalColorSpace](#), and [ProcessedColorSpace](#).

6.13.3.2 Conversion::Conversion (const Mat & src)

Constructor of the class

Parameters

<i>src</i>	a cv::Mat object which is the source image
------------	--

Definition at line 22 of file [Conversion.cpp](#).

References [None](#), [OriginalColorSpace](#), [Vision::ImageProcessing::OriginalImg](#), and [ProcessedColorSpace](#).

6.13.3.3 Conversion::Conversion (const Conversion & rhs)

Copy constructor

Definition at line 29 of file [Conversion.cpp](#).

References [OriginalColorSpace](#), [Vision::ImageProcessing::OriginalImg](#), [ProcessedColorSpace](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TemplImg](#).

6.13.3.4 Conversion::~Conversion ()

De-constructor of the class

Definition at line 38 of file [Conversion.cpp](#).

6.13.4 Member Function Documentation

6.13.4.1 void Conversion::Convert (ColorSpace convertFrom, ColorSpace convertTo, bool chain = false)

Convert the source image from one colorspace to a destination colorspace possibilities are:

- RGB 2 Intensity
- RGB 2 XYZ
- RGB 2 Lab
- RGB 2 Redness Index
- XYZ 2 Lab
- XYZ 2 Redness Index
- Lab 2 Redness Index

Parameters

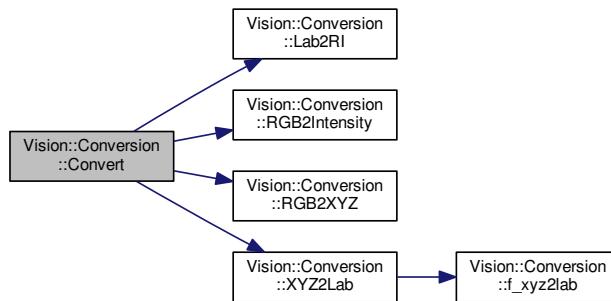
<i>convertFrom</i>	the starting colorspace
<i>convertTo</i>	the destination colorspace
<i>chain</i>	use the results from the previous operation default value = false;

Definition at line 86 of file [Conversion.cpp](#).

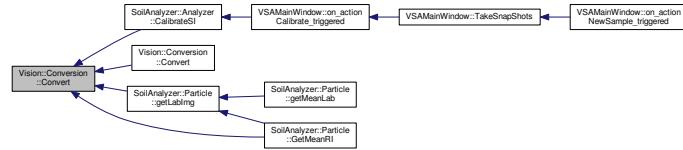
References [CHAIN_PROCESS](#), [CIE_lab](#), [CIE_XYZ](#), [Vision::ImageProcessing::currentProg](#), [EMPTY_CHECK](#), [Intensity](#), [Lab2RI\(\)](#), [OriginalColorSpace](#), [Vision::ImageProcessing::OriginalImg](#), [ProcessedColorSpace](#), [Vision::ImageProcessing::ProcessedImg](#), [Vision::ImageProcessing::prog_sig](#), [Vision::ImageProcessing::ProgStep](#), [RGB](#), [RGB2Intensity\(\)](#), [RGB2XYZ\(\)](#), [RI](#), and [XYZ2Lab\(\)](#).

Referenced by [SoilAnalyzer::Analyzer::CalibrateSI\(\)](#), [Convert\(\)](#), [SoilAnalyzer::Particle::getLabImg\(\)](#), and [SoilAnalyzer::Particle::GetMeanRI\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

Convert the source image from one colorspace to a destination colorspace

- RGB 2 Intensity
- RGB 2 XYZ
- RGB 2 Lab
- RGB 2 Redness Index
- XYZ 2 Lab
- XYZ 2 Redness Index
- Lab 2 Redness Index

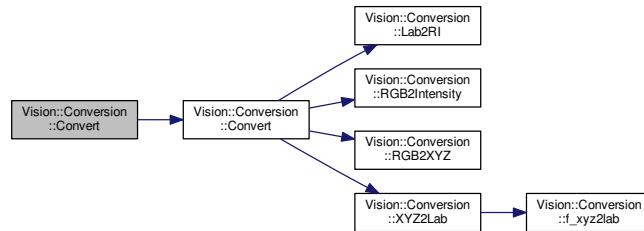
Parameters

<i>src</i>	a cv::Mat object which is the source image
<i>dst</i>	a cv::Mat object which is the destination image
<i>convertFrom</i>	the starting colorspace
<i>convertTo</i>	the destination colorspace
<i>chain</i>	use the results from the previous operation default value = false;

Definition at line 66 of file [Conversion.cpp](#).

References [Convert\(\)](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

Here is the call graph for this function:

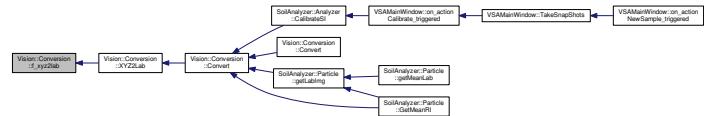


6.13.4.3 float Conversion::f_xyz2lab (float t) [inline], [private]

Definition at line 244 of file [Conversion.cpp](#).

Referenced by [XYZ2Lab\(\)](#).

Here is the caller graph for this function:



6.13.4.4 void Conversion::Lab2RI (float * O, float * P, int nData) [private]

Conversion from CIE La*b* to Redness Index

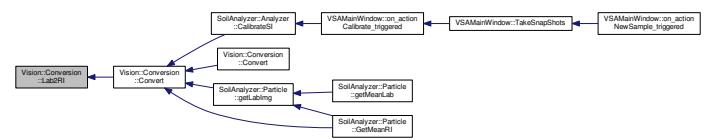
Parameters

O	a uchar pointer to the source image
P	a uchar pointer to the destination image
nData	an int indicating the total number of pixels

Definition at line 256 of file [Conversion.cpp](#).

Referenced by [Convert\(\)](#).

Here is the caller graph for this function:



6.13.4.5 Conversion & Conversion::operator= (Conversion rhs)

Assignment operator

Definition at line 41 of file [Conversion.cpp](#).

References [OriginalColorSpace](#), [Vision::ImageProcessing::OriginalImg](#), [ProcessedColorSpace](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TempImg](#).

6.13.4.6 void Conversion::RGB2Intensity (uchar * O, uchar * P, int nData) [private]

Conversion from RGB to Intensity

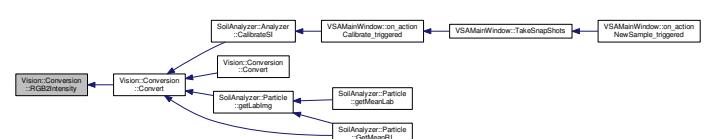
Parameters

O	a uchar pointer to the source image
P	a uchar pointer to the destination image
nData	an int indicating the total number of pixels

Definition at line 190 of file [Conversion.cpp](#).

Referenced by [Convert\(\)](#).

Here is the caller graph for this function:



6.13.4.7 void Conversion::RGB2XYZ(uchar * *O*, float * *P*, int *nData*) [private]

Conversion from RGB to CIE XYZ

Parameters

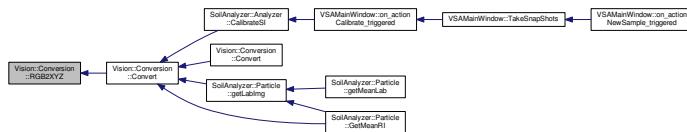
<i>O</i>	a uchar pointer to the source image
<i>P</i>	a uchar pointer to the destination image
<i>nData</i>	an int indicating the total number of pixels

Definition at line 207 of file [Conversion.cpp](#).

References [Vision::ImageProcessing::OriginalImg](#), and [XYZmat](#).

Referenced by [Convert\(\)](#).

Here is the caller graph for this function:



6.13.4.8 void Conversion::XYZ2Lab (float * *O*, float * *P*, int *nData*) [private]

[Conversion](#) from CIE XYZ to CIE La*b*

Parameters

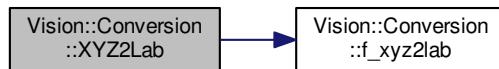
<i>O</i>	a uchar pointer to the source image
<i>P</i>	a uchar pointer to the destination image
<i>nData</i>	an int indicating the total number of pixels

Definition at line 225 of file [Conversion.cpp](#).

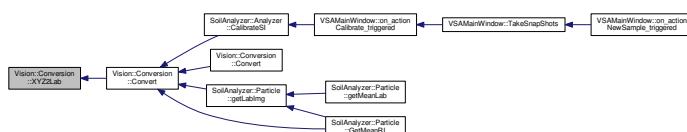
References [f_xyz2lab\(\)](#), and [whitePoint](#).

Referenced by [Convert\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.5 Member Data Documentation

6.13.5.1 ColorSpace Vision::Conversion::OriginalColorSpace

The original colorspace

Definition at line 24 of file [Conversion.h](#).

Referenced by [Conversion\(\)](#), [Convert\(\)](#), and [operator=\(\)](#).

6.13.5.2 ColorSpace Vision::Conversion::ProcessedColorSpace

The destination colorspace

Definition at line 25 of file [Conversion.h](#).

Referenced by [Conversion\(\)](#), [Convert\(\)](#), and [operator=\(\(\)\)](#).

6.13.5.3 float Vision::Conversion::whitePoint[3] [private]

Initial value:

```
= {
    0.9504, 1.0000, 1.0889}
```

Natural whitepoint in XYZ colorspace D65 according to Matlab

Definition at line 46 of file [Conversion.h](#).

Referenced by [XYZ2Lab\(\)](#).

6.13.5.4 float Vision::Conversion::XYZmat[3][3] [private]

Initial value:

```
= {{0.412453, 0.357580, 0.180423},
    {0.212671, 0.715160, 0.072169},
    {0.019334, 0.119194, 0.950227}}
```

< [Conversion](#) matrix used in the conversion between RGB and CIE XYZ

Definition at line 42 of file [Conversion.h](#).

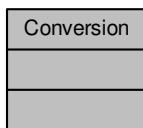
Referenced by [RGB2XYZ\(\)](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.cpp](#)

6.14 Conversion Class Reference

Collaboration diagram for Conversion:



6.14.1 Detailed Description

class which converts a cv::Mat image from one colorspace to the next colorspace

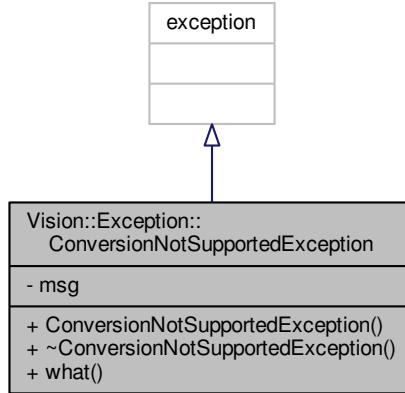
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.cpp](#)

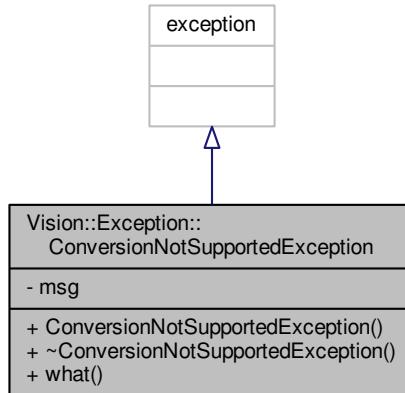
6.15 Vision::Exception::ConversionNotSupportedException Class Reference

```
#include <ConversionNotSupportedException.h>
```

Inheritance diagram for Vision::Exception::ConversionNotSupportedException:



Collaboration diagram for Vision::Exception::ConversionNotSupportedException:



Public Member Functions

- `ConversionNotSupportedException` (string m="Requested conversion is not supported!")
- `~ConversionNotSupportedException ()` `_GLIBCXX_USE_NOEXCEPT`
- `const char * what ()` `const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- string `msg`

6.15.1 Detailed Description

Definition at line 20 of file [ConversionNotSupportedException.h](#).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 `Vision::Exception::ConversionNotSupportedException::ConversionNotSupportedException (string m = "Requested conversion is not supported!") [inline]`

Definition at line 22 of file [ConversionNotSupportedException.h](#).

6.15.2.2 `Vision::Exception::ConversionNotSupportedException::~ConversionNotSupportedException () [inline]`

Definition at line 25 of file [ConversionNotSupportedException.h](#).

6.15.3 Member Function Documentation

6.15.3.1 `const char* Vision::Exception::ConversionNotSupportedException::what () const [inline]`

Definition at line 26 of file [ConversionNotSupportedException.h](#).

6.15.4 Member Data Documentation

6.15.4.1 `string Vision::Exception::ConversionNotSupportedException::msg [private]`

Definition at line 26 of file [ConversionNotSupportedException.h](#).

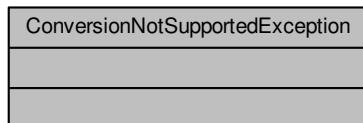
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ConversionNotSupportedException.h](#)

6.16 ConversionNotSupportedException Class Reference

`#include <ConversionNotSupportedException.h>`

Collaboration diagram for `ConversionNotSupportedException`:



6.16.1 Detailed Description

Exception class which is thrown when an illegal conversion is requested.

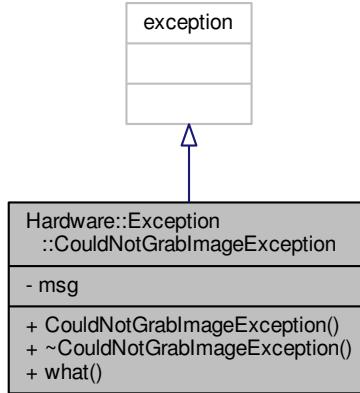
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ConversionNotSupportedException.h](#)

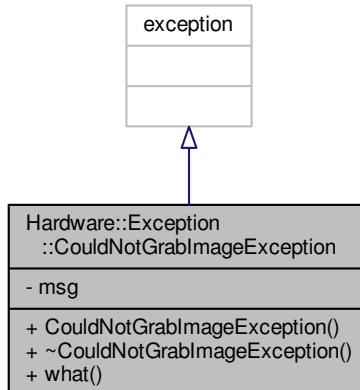
6.17 Hardware::Exception::CouldNotGrabImageException Class Reference

`#include <CouldNotGrabImageException.h>`

Inheritance diagram for Hardware::Exception::CouldNotGrabImageException:



Collaboration diagram for Hardware::Exception::CouldNotGrabImageException:



Public Member Functions

- `CouldNotGrabImageException` (string m="Unable to grab the next image!")
- `~CouldNotGrabImageException` () `__GLIBCXX__USE_NOEXCEPT`
- const char * `what` () const `__GLIBCXX__USE_NOEXCEPT`

Private Attributes

- string `msg`

6.17.1 Detailed Description

Definition at line 16 of file [CouldNotGrabImageException.h](#).

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `Hardware::Exception::CouldNotGrabImageException::CouldNotGrabImageException(string m = "Unable to grab the next image!")` [inline]

Definition at line 18 of file [CouldNotGrabImageException.h](#).

6.17.2.2 `Hardware::Exception::CouldNotGrabImageException::~CouldNotGrabImageException()` [inline]

Definition at line 20 of file [CouldNotGrabImageException.h](#).

6.17.3 Member Function Documentation

6.17.3.1 `const char* Hardware::Exception::CouldNotGrabImageException::what() const` [inline]

Definition at line 21 of file [CouldNotGrabImageException.h](#).

6.17.4 Member Data Documentation

6.17.4.1 `string Hardware::Exception::CouldNotGrabImageException::msg` [private]

Definition at line 21 of file [CouldNotGrabImageException.h](#).

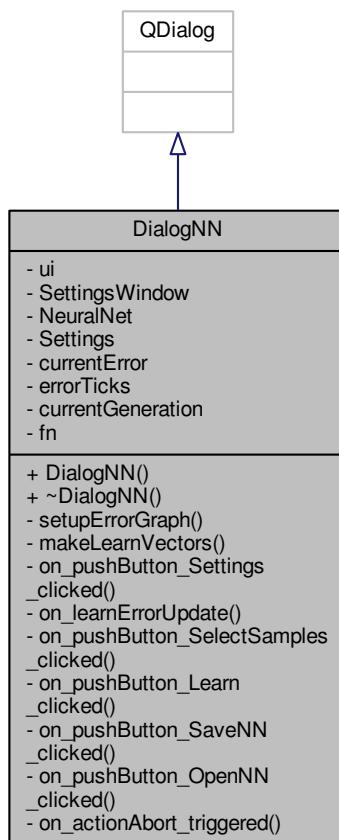
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/CouldNotGrabImageException.h](#)

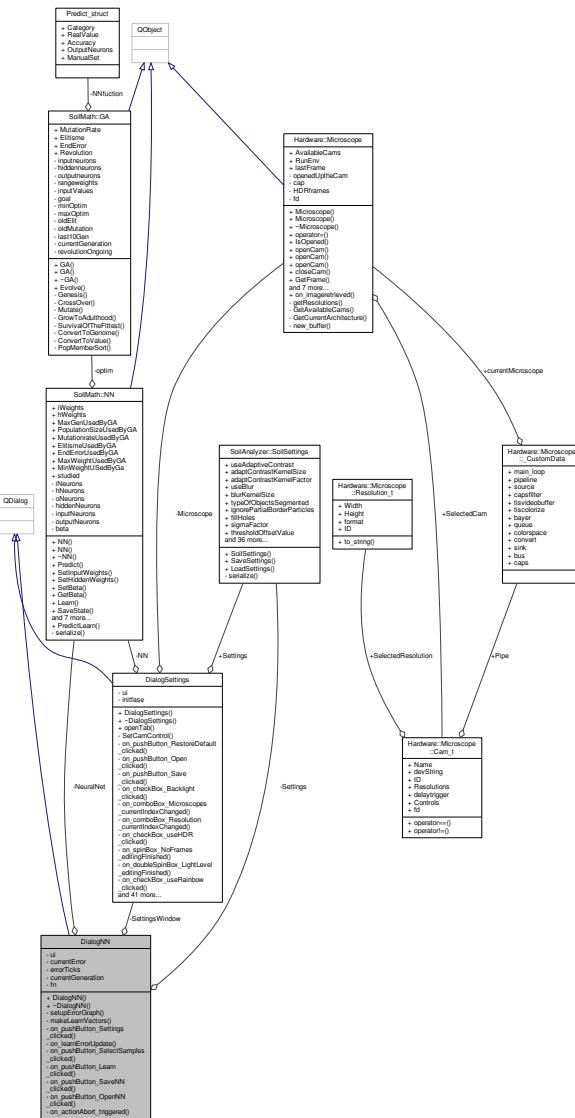
6.18 DialogNN Class Reference

```
#include <dialognn.h>
```

Inheritance diagram for DialogNN:



Collaboration diagram for DialogNN:



Public Member Functions

- **DialogNN** (QWidget *parent=0, SoilMath::NN *neuralnet=nullptr, SoilAnalyzer::SoilSettings *settings=nullptr, DialogSettings *setting←Window=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 Member Functions

- void `setupErrorGraph ()`
- void `makeLearnVectors (InputLearnVector_t &input, OutputLearnVector_t &output)`

Private Attributes

- `Ui::DialogNN * ui`
- `DialogSettings * SettingsWindow = nullptr`
- `SoilMath::NN * NeuralNet = nullptr`
- `SoilAnalyzer::SoilSettings * Settings = nullptr`
- `QVector< double > currentError`
- `QVector< double > errorTicks`
- `double currentGeneration = 0`
- `QStringList fn`

6.18.1 Detailed Description

Definition at line 15 of file `dialognn.h`.

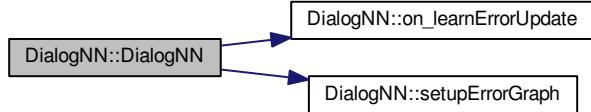
6.18.2 Constructor & Destructor Documentation

6.18.2.1 `DialogNN::DialogNN (QWidget * parent = 0, SoilMath::NN * neuralnet = nullptr, SoilAnalyzer::SoilSettings * settings = nullptr, DialogSettings * settingWindow = nullptr) [explicit]`

Definition at line 4 of file `dialognn.cpp`.

References `NeuralNet`, `on_learnErrorUpdate()`, `Settings`, `SettingsWindow`, `setupErrorGraph()`, and `ui`.

Here is the call graph for this function:

6.18.2.2 `DialogNN::~DialogNN ()`

Definition at line 42 of file `dialognn.cpp`.

References `ui`.

6.18.3 Member Function Documentation

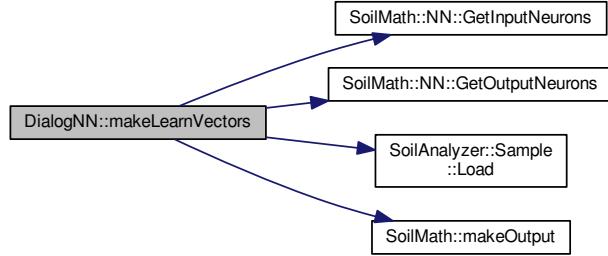
6.18.3.1 `void DialogNN::makeLearnVectors (InputLearnVector_t & input, OutputLearnVector_t & output) [private]`

Definition at line 97 of file `dialognn.cpp`.

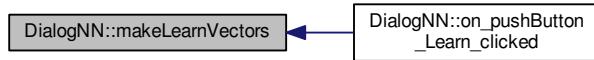
References `fn`, `SoilMath::NN::GetInputNeurons()`, `SoilMath::NN::GetOutputNeurons()`, `SoilAnalyzer::Sample::Load()`, `SoilMath::makeOutput()`, `NeuralNet`, `Predict_struct::OutputNeurons`, and `SoilAnalyzer::Sample::ParticlePopulation`.

Referenced by `on_pushButton_Learn_clicked()`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.3.2 void DialogNN::on_abort_triggered() [private], [slot]

Definition at line 147 of file [dialognn.cpp](#).

References [SoilMath::NN::EndErrorUsedByGA](#), [NeuralNet](#), and [ui](#).

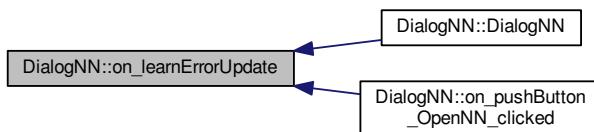
6.18.3.3 void DialogNN::on_learnErrorUpdate(double newError) [private], [slot]

Definition at line 50 of file [dialognn.cpp](#).

References [currentGeneration](#), and [ui](#).

Referenced by [DialogNN\(\)](#), and [on_pushButton_OpenNN_clicked\(\)](#).

Here is the caller graph for this function:

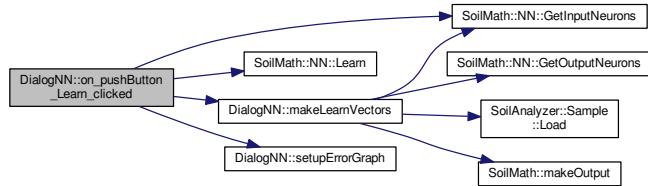


6.18.3.4 void DialogNN::on_pushButton_Learn_clicked() [private], [slot]

Definition at line 86 of file [dialognn.cpp](#).

References [fn](#), [SoilMath::NN::GetInputNeurons\(\)](#), [SoilMath::NN::Learn\(\)](#), [makeLearnVectors\(\)](#), [NeuralNet](#), and [setupErrorGraph\(\)](#).

Here is the call graph for this function:

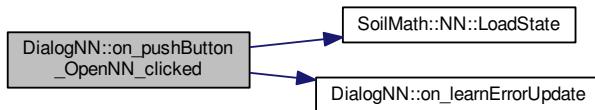


6.18.3.5 void DialogNN::on_pushButton_OpenNN_clicked() [private], [slot]

Definition at line 130 of file [dialognn.cpp](#).

References [fn](#), [SoilMath::NN::LoadState\(\)](#), [NeuralNet](#), [on_learnErrorUpdate\(\)](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), and [Settings](#).

Here is the call graph for this function:



6.18.3.6 void DialogNN::on_pushButton_SaveNN_clicked() [private], [slot]

Definition at line 118 of file [dialognn.cpp](#).

References [fn](#), [NeuralNet](#), [SoilAnalyzer::SoilSettings::NNFolder](#), [SoilMath::NN::SaveState\(\)](#), and [Settings](#).

Here is the call graph for this function:



6.18.3.7 void DialogNN::on_pushButton_SelectSamples_clicked() [private], [slot]

Definition at line 75 of file [dialognn.cpp](#).

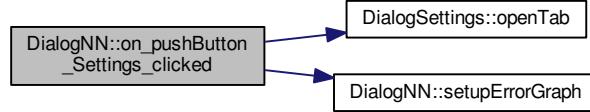
References [fn](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), and [Settings](#).

6.18.3.8 void DialogNN::on_pushButton_Settings_clicked() [private], [slot]

Definition at line 44 of file [dialognn.cpp](#).

References [DialogSettings::openTab\(\)](#), [SettingsWindow](#), and [setupErrorGraph\(\)](#).

Here is the call graph for this function:



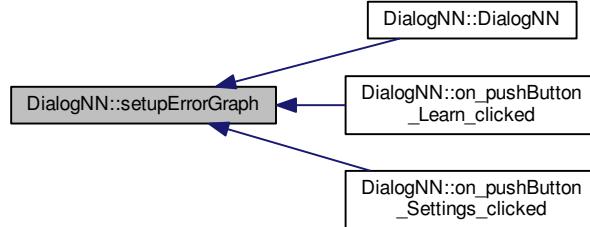
6.18.3.9 void DialogNN::setupErrorGraph () [private]

Definition at line 58 of file [dialognn.cpp](#).

References [SoilMath::NN::EndErrorUsedByGA](#), [errorTicks](#), [SoilMath::NN::MaxGenUsedByGA](#), [NeuralNet](#), and [ui](#).

Referenced by [DialogNN\(\)](#), [on_pushButton_Learn_clicked\(\)](#), and [on_pushButton_Settings_clicked\(\)](#).

Here is the caller graph for this function:



6.18.4 Member Data Documentation

6.18.4.1 QVector<double> DialogNN::currentError [private]

Definition at line 48 of file [dialognn.h](#).

6.18.4.2 double DialogNN::currentGeneration = 0 [private]

Definition at line 50 of file [dialognn.h](#).

Referenced by [on_learnErrorUpdate\(\)](#).

6.18.4.3 QVector<double> DialogNN::errorTicks [private]

Definition at line 49 of file [dialognn.h](#).

Referenced by [setupErrorGraph\(\)](#).

6.18.4.4 QStringList DialogNN::fn [private]

Definition at line 51 of file [dialognn.h](#).

Referenced by [makeLearnVectors\(\)](#), [on_pushButton_Learn_clicked\(\)](#), [on_pushButton_OpenNN_clicked\(\)](#), [on_pushButton_SaveNN_clicked\(\)](#), and [on_pushButton_SelectSamples_clicked\(\)](#).

6.18.4.5 SoilMath::NN* DialogNN::NeuralNet = nullptr [private]

Definition at line 42 of file [dialognn.h](#).

Referenced by [DialogNN\(\)](#), [makeLearnVectors\(\)](#), [on_actionAbort_triggered\(\)](#), [on_pushButton_Learn_clicked\(\)](#), [on_pushButton_OpenNN_clicked\(\)](#), [on_pushButton_SaveNN_clicked\(\)](#), and [setupErrorGraph\(\)](#).

6.18.4.6 **SoilAnalyzer::SoilSettings*** DialogNN::Settings = `nullptr` [private]

Definition at line 43 of file [dialognn.h](#).

Referenced by [DialogNN\(\)](#), [on_pushButton_OpenNN_clicked\(\)](#), [on_pushButton_SaveNN_clicked\(\)](#), and [on_pushButton_SelectSamples_clicked\(\)](#).

6.18.4.7 **DialogSettings* DialogNN::SettingsWindow = `nullptr`** [private]

Definition at line 41 of file [dialognn.h](#).

Referenced by [DialogNN\(\)](#), and [on_pushButton_Settings_clicked\(\)](#).

6.18.4.8 **Ui::DialogNN* DialogNN::ui** [private]

Definition at line 40 of file [dialognn.h](#).

Referenced by [DialogNN\(\)](#), [on_actionAbort_triggered\(\)](#), [on_learnErrorUpdate\(\)](#), [setupErrorGraph\(\)](#), and [~DialogNN\(\)](#).

The documentation for this class was generated from the following files:

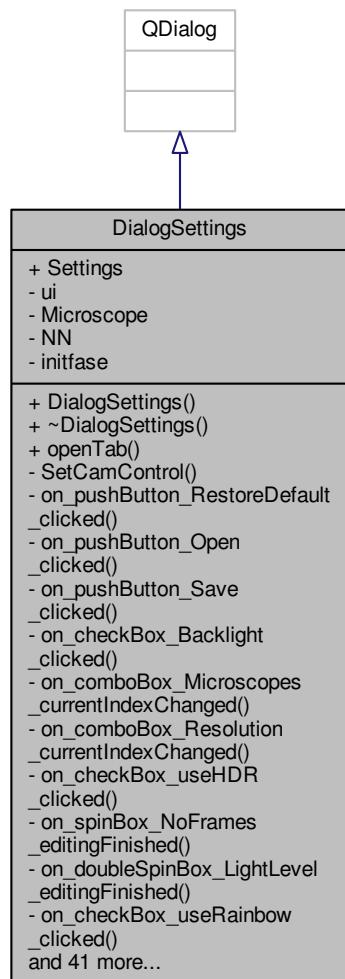
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialognn.h](#)

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialognn.cpp](#)

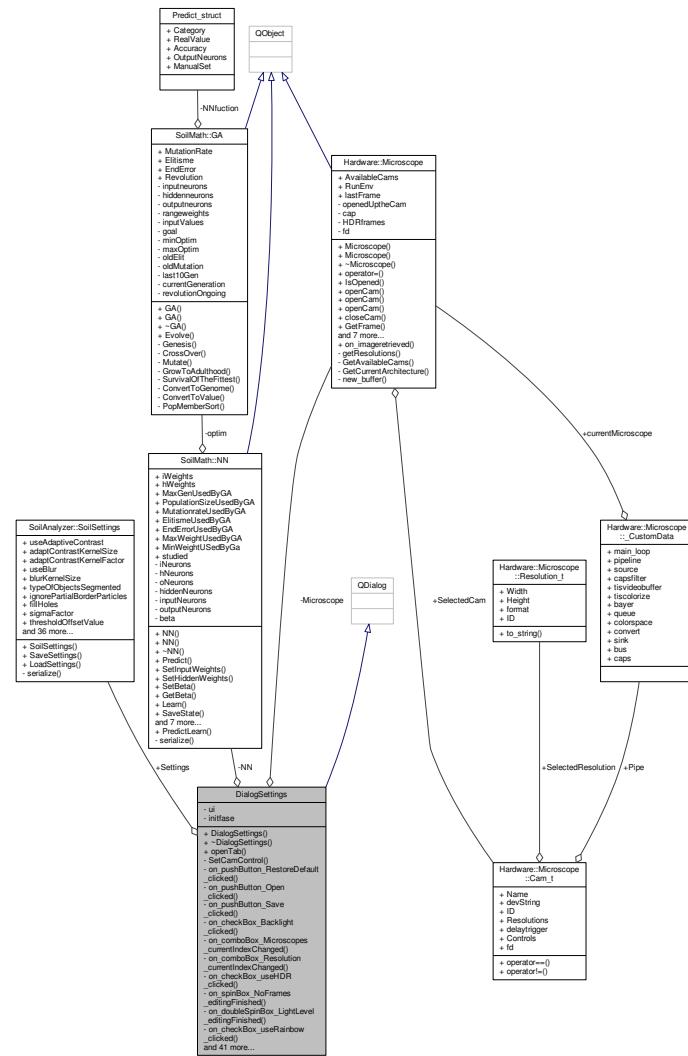
6.19 DialogSettings Class Reference

```
#include <dialogsettings.h>
```

Inheritance diagram for DialogSettings:



Collaboration diagram for DialogSettings:



Public Member Functions

- `DialogSettings` (QWidget *parent=0, SoilAnalyzer::SoilSettings *settings=nullptr, Hardware::Microscope *microscope=nullptr, SoilMath::NN *nn=nullptr, bool openNN=false)
 - `~DialogSettings` ()
 - void `openTab` (int newValue)

Public Attributes

- `SoilAnalyzer::SoilSettings * Settings = nullptr`

Private Slots

- void **on_pushButton_RestoreDefault_clicked** ()
 - void **on_pushButton_Open_clicked** ()
 - void **on_pushButton_Save_clicked** ()
 - void **on_checkBox_Backlight_clicked** (bool checked)
 - void **on_comboBox_Microscopes_currentIndexChanged** (const QString &arg1)

- void `on_comboBox_Resolution_currentIndexChanged` (int index)
- void `on_checkBox_useHDR_clicked` (bool checked)
- void `on_spinBox_NoFrames_editingFinished` ()
- void `on_doubleSpinBox_LightLevel_editingFinished` ()
- void `on_checkBox_useRainbow_clicked` (bool checked)
- void `on_checkBox_InvertEncoder_clicked` (bool checked)
- void `on_checkBox_useCUDA_clicked` (bool checked)
- void `on_horizontalSlider_BrightFront_valueChanged` (int value)
- void `on_horizontalSlider_ContrastFront_valueChanged` (int value)
- void `on_horizontalSlider_SaturationFront_valueChanged` (int value)
- void `on_horizontalSlider_HueFront_valueChanged` (int value)
- void `on_horizontalSlider_SharpnessFront_valueChanged` (int value)
- void `on_horizontalSlider_BrightProj_valueChanged` (int value)
- void `on_horizontalSlider_ContrastProj_valueChanged` (int value)
- void `on_horizontalSlider_SaturationProj_valueChanged` (int value)
- void `on_horizontalSlider_HueProj_valueChanged` (int value)
- void `on_horizontalSlider_SharpnessProj_valueChanged` (int value)
- void `on_cb_use_adaptContrast_3_clicked` (bool checked)
- void `on_cb_useBlur_3_clicked` (bool checked)
- void `on_rb_useDark_3_toggled` (bool checked)
- void `on_cb_ignoreBorder_3_clicked` (bool checked)
- void `on_cb_fillHoles_3_clicked` (bool checked)
- void `on_sb_sigmaFactor_3_editingFinished` ()
- void `on_rb_useOpen_3_clicked` (bool checked)
- void `on_rb_useClose_3_clicked` (bool checked)
- void `on_rb_useErode_3_clicked` (bool checked)
- void `on_rb_useDilate_3_clicked` (bool checked)
- void `on_sb_morphMask_3_editingFinished` ()
- void `on_spinBox_MaxGen_editingFinished` ()
- void `on_spinBox_PopSize_editingFinished` ()
- void `on_doubleSpinBox_MutationRate_editingFinished` ()
- void `on_spinBox_Elitisme_editingFinished` ()
- void `on_doubleSpinBox_endError_editingFinished` ()
- void `on_doubleSpinBox_maxWeight_editingFinished` ()
- void `on_doubleSpinBox_MinWeight_editingFinished` ()
- void `on_doubleSpinBox_Beta_editingFinished` ()
- void `on_spinBox_InputNeurons_editingFinished` ()
- void `on_spinBox_HiddenNeurons_editingFinished` ()
- void `on_spinBox_OutputNeurons_editingFinished` ()
- void `on_pushButton_selectSampleFolder_clicked` ()
- void `on_pushButton_SelectSettingFolder_clicked` ()
- void `on_pushButton_SelectNNFolder_clicked` ()
- void `on_pushButton_SelectNN_clicked` ()
- void `on_spinBox_NoShots_editingFinished` ()
- void `on_checkBox_PredictShape_clicked` (bool checked)
- void `on_checkBox_revolt_clicked` (bool checked)

Private Member Functions

- void `SetCamControl (Hardware::Microscope::Cam_t *selectedCam, QSlider *Brightness, QSlider *Contrast, QSlider *Saturation, QSlider *Hue, QSlider *Sharpness)`

Private Attributes

- `Ui::DialogSettings * ui`
- `Hardware::Microscope * Microscope`
- `SoilMath::NN * NN`
- `bool initfase = true`

6.19.1 Detailed Description

Definition at line 16 of file `dialogsettings.h`.

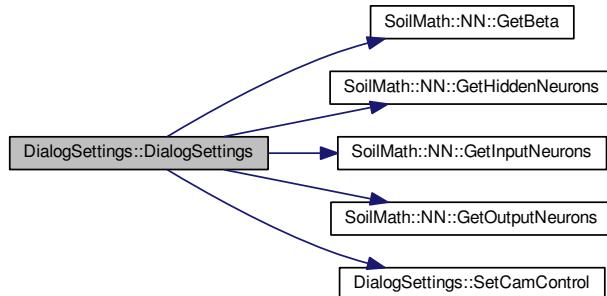
6.19.2 Constructor & Destructor Documentation

6.19.2.1 `DialogSettings::DialogSettings (QWidget * parent = 0, SoilAnalyzer::SoilSettings * settings = nullptr, Hardware::Microscope * microscope = nullptr, SoilMath::NN * nn = nullptr, bool openNN = false) [explicit]`

Definition at line 5 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::adaptContrastKernelFactor](#), [SoilAnalyzer::SoilSettings::adaptContrastKernelSize](#), [SoilAnalyzer::SoilSettings::blurKernelSize](#), [Vision::Segment::Bright](#), [SoilAnalyzer::SoilSettings::Brightness_front](#), [SoilAnalyzer::SoilSettings::Brightness_proj](#), [Vision::MorphologicalFilter::CLOSE](#), [SoilAnalyzer::SoilSettings::Contrast_front](#), [SoilAnalyzer::SoilSettings::Contrast_proj](#), [Vision::Segment::Dark](#), [Vision::MorphologicalFilter::DILATE](#), [SoilMath::NN::ElitismeUsedByGA](#), [SoilAnalyzer::SoilSettings::enableRainbow](#), [SoilAnalyzer::SoilSettings::enclnv](#), [SoilMath::NN::EndErrorUsedByGA](#), [Vision::MorphologicalFilter::ERODE](#), [SoilAnalyzer::SoilSettings::fillHoles](#), [SoilAnalyzer::SoilSettings::filterMaskSize](#), [SoilMath::NN::GetBeta](#), [SoilMath::NN::GetHiddenNeurons](#), [SoilMath::NN::GetInputNeurons](#), [SoilMath::NN::GetOutputNeurons](#), [SoilAnalyzer::SoilSettings::HDRframes](#), [SoilAnalyzer::SoilSettings::Hue_front](#), [SoilAnalyzer::SoilSettings::Hue_proj](#), [SoilAnalyzer::SoilSettings::ignorePartialBorderParticles](#), [initfase](#), [SoilMath::NN::MaxGenUsedByGA](#), [SoilMath::NN::MaxWeightUsedByGA](#), [SoilMath::NN::MinWeightUsedByGA](#), [SoilAnalyzer::SoilSettings::morphFilterType](#), [SoilMath::NN::MutationrateUsedByGA](#), [SoilAnalyzer::SoilSettings::NNFolder](#), [SoilAnalyzer::SoilSettings::NNlocation](#), [Vision::MorphologicalFilter::OPEN](#), [SoilMath::NN::PopulationSizeUsedByGA](#), [SoilAnalyzer::SoilSettings::PredictTheShape](#), [SoilAnalyzer::SoilSettings::Revolution](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), [SoilAnalyzer::SoilSettings::Saturation_front](#), [SoilAnalyzer::SoilSettings::Saturation_proj](#), [SetCamControl\(\)](#), [Settings](#), [SoilAnalyzer::SoilSettings::SettingsFolder](#), [SoilAnalyzer::SoilSettings::Sharpness_front](#), [SoilAnalyzer::SoilSettings::Sharpness_proj](#), [SoilAnalyzer::SoilSettings::sigmaFactor](#), [SoilAnalyzer::SoilSettings::StandardNumberOfShots](#), [SoilAnalyzer::SoilSettings::StandardPrinter](#), [SoilAnalyzer::SoilSettings::StandardSentTo](#), [SoilAnalyzer::SoilSettings::typeOfObjectsSegmented](#), [ui](#), [SoilAnalyzer::SoilSettings::useAdaptiveContrast](#), [SoilAnalyzer::SoilSettings::useBacklightProjection](#), [SoilAnalyzer::SoilSettings::useBlur](#), [SoilAnalyzer::SoilSettings::useCUDA](#), [SoilAnalyzer::SoilSettings::useHDR](#), and [Hardware::Microscope::X64](#).

Here is the call graph for this function:

6.19.2.2 `DialogSettings::~DialogSettings ()`

Definition at line 188 of file [dialogsettings.cpp](#).

References [ui](#).

6.19.3 Member Function Documentation

6.19.3.1 `void DialogSettings::on_cb_fillHoles_3_clicked (bool checked) [private], [slot]`

Definition at line 393 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::fillHoles](#), and [Settings](#).

6.19.3.2 `void DialogSettings::on_cb_ignoreBorder_3_clicked (bool checked) [private], [slot]`

Definition at line 389 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::ignorePartialBorderParticles](#), and [Settings](#).

6.19.3.3 `void DialogSettings::on_cb_use_adaptContrast_3_clicked (bool checked) [private], [slot]`

Definition at line 370 of file [dialogsettings.cpp](#).

References [Settings](#), [ui](#), and [SoilAnalyzer::SoilSettings::useAdaptiveContrast](#).

6.19.3.4 void DialogSettings::on_cb_useBlur_3_clicked (bool *checked*) [private], [slot]

Definition at line 376 of file [dialogsettings.cpp](#).

References [Settings](#), [ui](#), and [SoilAnalyzer::SoilSettings::useBlur](#).

6.19.3.5 void DialogSettings::on_checkBox_Backlight_clicked (bool *checked*) [private], [slot]

Definition at line 222 of file [dialogsettings.cpp](#).

References [Settings](#), [ui](#), and [SoilAnalyzer::SoilSettings::useBacklightProjection](#).

6.19.3.6 void DialogSettings::on_checkBox_InvertEncoder_clicked (bool *checked*) [private], [slot]

Definition at line 299 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::enclInv](#), and [Settings](#).

6.19.3.7 void DialogSettings::on_checkBox_PredictShape_clicked (bool *checked*) [private], [slot]

Definition at line 515 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::PredictTheShape](#), and [Settings](#).

6.19.3.8 void DialogSettings::on_checkBox_revolt_clicked (bool *checked*) [private], [slot]

Definition at line 519 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Revolution](#), and [Settings](#).

6.19.3.9 void DialogSettings::on_checkBox_useCUDA_clicked (bool *checked*) [private], [slot]

Definition at line 303 of file [dialogsettings.cpp](#).

References [Settings](#), and [SoilAnalyzer::SoilSettings::useCUDA](#).

6.19.3.10 void DialogSettings::on_checkBox_useHDR_clicked (bool *checked*) [private], [slot]

Definition at line 256 of file [dialogsettings.cpp](#).

References [Settings](#), [ui](#), and [SoilAnalyzer::SoilSettings::useHDR](#).

6.19.3.11 void DialogSettings::on_checkBox_useRainbow_clicked (bool *checked*) [private], [slot]

Definition at line 295 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::enableRainbow](#), and [Settings](#).

6.19.3.12 void DialogSettings::on_comboBox_Microscopes_currentIndexChanged (const QString & *arg1*) [private], [slot]

Definition at line 227 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::defaultWebcam](#), [initfase](#), [Settings](#), and [ui](#).

6.19.3.13 void DialogSettings::on_comboBox_Resolution_currentIndexChanged (int *index*) [private], [slot]

Definition at line 247 of file [dialogsettings.cpp](#).

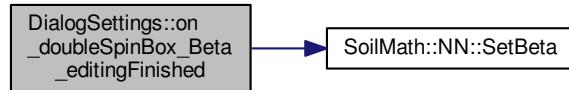
References [initfase](#), [SoilAnalyzer::SoilSettings::selectedResolution](#), and [Settings](#).

6.19.3.14 void DialogSettings::on_doubleSpinBox_Beta_editingFinished () [private], [slot]

Definition at line 449 of file [dialogsettings.cpp](#).

References [SoilMath::NN::SetBeta\(\)](#), and [ui](#).

Here is the call graph for this function:



6.19.3.15 void DialogSettings::on_doubleSpinBox_endError_editingFinished() [private], [slot]

Definition at line 437 of file [dialogsettings.cpp](#).

References [SoilMath::NN::EndErrorUsedByGA](#), and [ui](#).

6.19.3.16 void DialogSettings::on_doubleSpinBox_LightLevel_editingFinished() [private], [slot]

Definition at line 290 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::lightLevel](#), [Settings](#), and [ui](#).

6.19.3.17 void DialogSettings::on_doubleSpinBox_maxWeight_editingFinished() [private], [slot]

Definition at line 441 of file [dialogsettings.cpp](#).

References [SoilMath::NN::MaxWeightUsedByGA](#), and [ui](#).

6.19.3.18 void DialogSettings::on_doubleSpinBox_MinWeight_editingFinished() [private], [slot]

Definition at line 445 of file [dialogsettings.cpp](#).

References [SoilMath::NN::MinWeightUsedByGA](#), and [ui](#).

6.19.3.19 void DialogSettings::on_doubleSpinBox_MutationRate_editingFinished() [private], [slot]

Definition at line 429 of file [dialogsettings.cpp](#).

References [SoilMath::NN::MutationrateUsedByGA](#), and [ui](#).

6.19.3.20 void DialogSettings::on_horizontalSlider_BrightFront_valueChanged(int value) [private], [slot]

Definition at line 307 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Brightness_front](#), [initfase](#), and [Settings](#).

6.19.3.21 void DialogSettings::on_horizontalSlider_BrightProj_valueChanged(int value) [private], [slot]

Definition at line 339 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Brightness_proj](#), [initfase](#), and [Settings](#).

6.19.3.22 void DialogSettings::on_horizontalSlider_ContrastFront_valueChanged(int value) [private], [slot]

Definition at line 313 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Contrast_front](#), [initfase](#), and [Settings](#).

6.19.3.23 void DialogSettings::on_horizontalSlider_ContrastProj_valueChanged(int value) [private], [slot]

Definition at line 345 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Contrast_proj](#), [initfase](#), and [Settings](#).

6.19.3.24 void DialogSettings::on_horizontalSlider_HueFront_valueChanged(int value) [private], [slot]

Definition at line 326 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Hue_front](#), [initfase](#), and [Settings](#).

6.19.3.25 void DialogSettings::on_horizontalSlider_HueProj_valueChanged (int value) [private], [slot]

Definition at line 358 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::Hue_proj](#), [initfase](#), and [Settings](#).

6.19.3.26 void DialogSettings::on_horizontalSlider_SaturationFront_valueChanged (int value) [private], [slot]

Definition at line 319 of file [dialogsettings.cpp](#).

References [initfase](#), [SoilAnalyzer::SoilSettings::Saturation_front](#), and [Settings](#).

6.19.3.27 void DialogSettings::on_horizontalSlider_SaturationProj_valueChanged (int value) [private], [slot]

Definition at line 351 of file [dialogsettings.cpp](#).

References [initfase](#), [SoilAnalyzer::SoilSettings::Saturation_proj](#), and [Settings](#).

6.19.3.28 void DialogSettings::on_horizontalSlider_SharpnessFront_valueChanged (int value) [private], [slot]

Definition at line 332 of file [dialogsettings.cpp](#).

References [initfase](#), [Settings](#), and [SoilAnalyzer::SoilSettings::Sharpness_front](#).

6.19.3.29 void DialogSettings::on_horizontalSlider_SharpnessProj_valueChanged (int value) [private], [slot]

Definition at line 364 of file [dialogsettings.cpp](#).

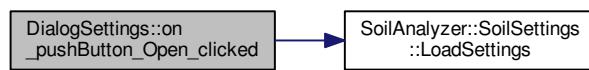
References [initfase](#), [Settings](#), and [SoilAnalyzer::SoilSettings::Sharpness_proj](#).

6.19.3.30 void DialogSettings::on_pushButton_Open_clicked () [private], [slot]

Definition at line 200 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::LoadSettings\(\)](#), and [Settings](#).

Here is the call graph for this function:

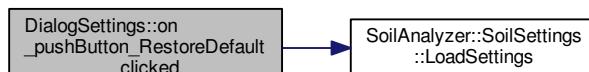


6.19.3.31 void DialogSettings::on_pushButton_RestoreDefault_clicked () [private], [slot]

Definition at line 196 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::LoadSettings\(\)](#), and [Settings](#).

Here is the call graph for this function:

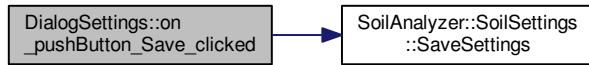


6.19.3.32 void DialogSettings::on_pushButton_Save_clicked () [private], [slot]

Definition at line 211 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::SaveSettings\(\)](#), and [Settings](#).

Here is the call graph for this function:



6.19.3.33 void DialogSettings::on_pushButton_SelectNN_clicked() [private], [slot]

Definition at line 498 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::NNlocation](#), [Settings](#), and [ui](#).

6.19.3.34 void DialogSettings::on_pushButton_SelectNNFolder_clicked() [private], [slot]

Definition at line 487 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::NNFolder](#), [Settings](#), and [ui](#).

6.19.3.35 void DialogSettings::on_pushButton_selectSampleFolder_clicked() [private], [slot]

Definition at line 465 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::SampleFolder](#), [Settings](#), and [ui](#).

6.19.3.36 void DialogSettings::on_pushButton_SelectSettingFolder_clicked() [private], [slot]

Definition at line 476 of file [dialogsettings.cpp](#).

References [Settings](#), [SoilAnalyzer::SoilSettings::SettingsFolder](#), and [ui](#).

6.19.3.37 void DialogSettings::on_rb_useClose_3_clicked(bool checked) [private], [slot]

Definition at line 405 of file [dialogsettings.cpp](#).

References [Vision::MorphologicalFilter::CLOSE](#), [SoilAnalyzer::SoilSettings::morphFilterType](#), and [Settings](#).

6.19.3.38 void DialogSettings::on_rb_useDark_3_toggled(bool checked) [private], [slot]

Definition at line 381 of file [dialogsettings.cpp](#).

References [Vision::Segment::Bright](#), [Vision::Segment::Dark](#), [Settings](#), and [SoilAnalyzer::SoilSettings::typeOfObjectsSegmented](#).

6.19.3.39 void DialogSettings::on_rb_useDilate_3_clicked(bool checked) [private], [slot]

Definition at line 413 of file [dialogsettings.cpp](#).

References [Vision::MorphologicalFilter::DILATE](#), [SoilAnalyzer::SoilSettings::morphFilterType](#), and [Settings](#).

6.19.3.40 void DialogSettings::on_rb_useErode_3_clicked(bool checked) [private], [slot]

Definition at line 409 of file [dialogsettings.cpp](#).

References [Vision::MorphologicalFilter::ERODE](#), [SoilAnalyzer::SoilSettings::morphFilterType](#), and [Settings](#).

6.19.3.41 void DialogSettings::on_rb_useOpen_3_clicked(bool checked) [private], [slot]

Definition at line 401 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::morphFilterType](#), [Vision::MorphologicalFilter::OPEN](#), and [Settings](#).

6.19.3.42 void DialogSettings::on_sb_morphMask_3_editingFinished() [private], [slot]

Definition at line 417 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::filterMaskSize](#), [Settings](#), and [ui](#).

6.19.3.43 void DialogSettings::on_sb_sigmaFactor_3_editingFinished() [private], [slot]

Definition at line 397 of file [dialogsettings.cpp](#).

References [Settings](#), [SoilAnalyzer::SoilSettings::sigmaFactor](#), and [ui](#).

6.19.3.44 void DialogSettings::on_spinBox_Elitisme_editingFinished () [private], [slot]

Definition at line 433 of file [dialogsettings.cpp](#).

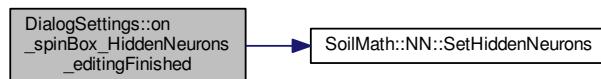
References [SoilMath::NN::ElitismeUsedByGA](#), and [ui](#).

6.19.3.45 void DialogSettings::on_spinBox_HiddenNeurons_editingFinished () [private], [slot]

Definition at line 457 of file [dialogsettings.cpp](#).

References [SoilMath::NN::SetHiddenNeurons\(\)](#), and [ui](#).

Here is the call graph for this function:

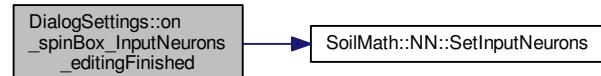


6.19.3.46 void DialogSettings::on_spinBox_InputNeurons_editingFinished () [private], [slot]

Definition at line 453 of file [dialogsettings.cpp](#).

References [SoilMath::NN::SetInputNeurons\(\)](#), and [ui](#).

Here is the call graph for this function:



6.19.3.47 void DialogSettings::on_spinBox_MaxGen_editingFinished () [private], [slot]

Definition at line 421 of file [dialogsettings.cpp](#).

References [SoilMath::NN::MaxGenUsedByGA](#), and [ui](#).

6.19.3.48 void DialogSettings::on_spinBox_NoFrames_editingFinished () [private], [slot]

Definition at line 286 of file [dialogsettings.cpp](#).

References [SoilAnalyzer::SoilSettings::HDRframes](#), [Settings](#), and [ui](#).

6.19.3.49 void DialogSettings::on_spinBox_NoShots_editingFinished () [private], [slot]

Definition at line 511 of file [dialogsettings.cpp](#).

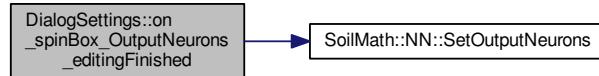
References [Settings](#), [SoilAnalyzer::SoilSettings::StandardNumberOfShots](#), and [ui](#).

6.19.3.50 void DialogSettings::on_spinBox_OutputNeurons_editingFinished () [private], [slot]

Definition at line 461 of file [dialogsettings.cpp](#).

References [SoilMath::NN::SetOutputNeurons\(\)](#), and [ui](#).

Here is the call graph for this function:



6.19.3.51 void DialogSettings::on_spinBox_PopSize_editingFinished () [private], [slot]

Definition at line 425 of file [dialogsettings.cpp](#).

References [SoilMath::NN::PopulationSizeUsedByGA](#), and [ui](#).

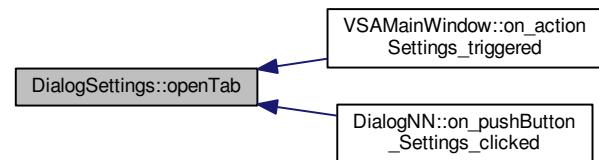
6.19.3.52 void DialogSettings::openTab (int newValue)

Definition at line 190 of file [dialogsettings.cpp](#).

References [ui](#).

Referenced by [VSAMainWindow::on_actionSettings_triggered\(\)](#), and [DialogNN::on_pushButton_Settings_clicked\(\)](#).

Here is the caller graph for this function:



6.19.3.53 void DialogSettings::SetCamControl (Hardware::Microscope::Cam_t * selectedCam, QSlider * Brightness, QSlider * Contrast, QSlider * Saturation, QSlider * Hue, QSlider * Sharpness) [private]

Definition at line 262 of file [dialogsettings.cpp](#).

References [Hardware::Microscope::Cam_t::Controls](#).

Referenced by [DialogSettings\(\)](#).

Here is the caller graph for this function:



6.19.4 Member Data Documentation

6.19.4.1 bool DialogSettings::initfase = true [private]

Definition at line 136 of file [dialogsettings.h](#).

Referenced by [DialogSettings\(\)](#), [on_comboBox_Microscopes_currentIndexChanged\(\)](#), [on_comboBox_Resolution_currentIndexChanged\(\)](#), [on_horizontalSlider_BrightFront_valueChanged\(\)](#), [on_horizontalSlider_BrightProj_valueChanged\(\)](#), [on_horizontalSlider_ContrastFront_valueChanged\(\)](#), [on_horizontalSlider_ContrastProj_valueChanged\(\)](#), [on_horizontalSlider_Saturation_valueChanged\(\)](#), [on_horizontalSlider_Hue_valueChanged\(\)](#), [on_horizontalSlider_Sharpness_valueChanged\(\)](#), [on_horizontalSlider_Brightness_valueChanged\(\)](#), and [on_horizontalSlider_Contrast_valueChanged\(\)](#).

Changed(), on_horizontalSlider_ContrastProj_valueChanged(), on_horizontalSlider_HueFront_valueChanged(), on_horizontalSlider_HueProj_valueChanged(), on_horizontalSlider_SaturationFront_valueChanged(), on_horizontalSlider_SaturationProj_valueChanged(), on_horizontalSlider_SharpnessFront_valueChanged(), and on_horizontalSlider_SharpnessProj_valueChanged().

6.19.4.2 Hardware::Microscope* DialogSettings::Microscope [private]

Definition at line 134 of file [dialogsettings.h](#).

6.19.4.3 SoilMath::NN* DialogSettings::NN [private]

Definition at line 135 of file [dialogsettings.h](#).

6.19.4.4 SoilAnalyzer::SoilSettings* DialogSettings::Settings = nullptr

Definition at line 20 of file [dialogsettings.h](#).

Referenced by [DialogSettings\(\)](#), [on_cb_fillHoles_3_clicked\(\)](#), [on_cb_ignoreBorder_3_clicked\(\)](#), [on_cb_use_adaptContrast_3_clicked\(\)](#), [on_cb_useBlur_3_clicked\(\)](#), [on_checkBox_Backlight_clicked\(\)](#), [on_checkBox_InvertEncoder_clicked\(\)](#), [on_checkBox_PredictShape_clicked\(\)](#), [on_checkBox_revolt_clicked\(\)](#), [on_checkBox_useCUDA_clicked\(\)](#), [on_checkBox_useHDR_clicked\(\)](#), [on_checkBox_useRainbow_clicked\(\)](#), [on_comboBox_Microscopes_currentIndexChanged\(\)](#), [on_doubleSpinBox_Resolution_currentIndexChanged\(\)](#), [on_doubleSpinBox_LightLevel_editingFinished\(\)](#), [on_horizontalSlider_BrightFront_valueChanged\(\)](#), [on_horizontalSlider_BrightProj_valueChanged\(\)](#), [on_horizontalSlider_ContrastFront_valueChanged\(\)](#), [on_horizontalSlider_ContrastProj_valueChanged\(\)](#), [on_horizontalSlider_HueFront_valueChanged\(\)](#), [on_horizontalSlider_HueProj_valueChanged\(\)](#), [on_horizontalSlider_SaturationFront_valueChanged\(\)](#), [on_horizontalSlider_SaturationProj_valueChanged\(\)](#), [on_horizontalSlider_SharpnessFront_valueChanged\(\)](#), [on_horizontalSlider_SharpnessProj_valueChanged\(\)](#), [on_pushButton_Open_clicked\(\)](#), [on_pushButton_RestoreDefault_clicked\(\)](#), [on_pushButton_Save_clicked\(\)](#), [on_pushButton_SelectNN_clicked\(\)](#), [on_pushButton_SelectNNFolder_clicked\(\)](#), [on_pushButton_selectSampleFolder_clicked\(\)](#), [on_pushButton_SelectSettingFolder_clicked\(\)](#), [on_rb_useClose_3_clicked\(\)](#), [on_rb_useDark_3_toggled\(\)](#), [on_rb_useDilate_3_clicked\(\)](#), [on_rb_useErode_3_clicked\(\)](#), [on_rb_useOpen_3_clicked\(\)](#), [on_sb_morphMask_3_editingFinished\(\)](#), [on_sb_sigmaFactor_3_editingFinished\(\)](#), [on_spinBox_NoFrames_editingFinished\(\)](#), and [on_spinBox_NoShots_editingFinished\(\)](#).

6.19.4.5 Ui::DialogSettings* DialogSettings::ui [private]

Definition at line 133 of file [dialogsettings.h](#).

Referenced by [DialogSettings\(\)](#), [on_cb_use_adaptContrast_3_clicked\(\)](#), [on_cb_useBlur_3_clicked\(\)](#), [on_checkBox_Backlight_clicked\(\)](#), [on_checkBox_useHDR_clicked\(\)](#), [on_comboBox_Microscopes_currentIndexChanged\(\)](#), [on_doubleSpinBox_Beta_editingFinished\(\)](#), [on_doubleSpinBox_endError_editingFinished\(\)](#), [on_doubleSpinBox_LightLevel_editingFinished\(\)](#), [on_doubleSpinBox_maxWeight_editingFinished\(\)](#), [on_doubleSpinBox_MinWeight_editingFinished\(\)](#), [on_doubleSpinBox_MutationRate_editingFinished\(\)](#), [on_pushButton_SelectNN_clicked\(\)](#), [on_pushButton_SelectNNFolder_clicked\(\)](#), [on_pushButton_selectSampleFolder_clicked\(\)](#), [on_pushButton_SelectSettingFolder_clicked\(\)](#), [on_sb_morphMask_3_editingFinished\(\)](#), [on_sb_sigmaFactor_3_editingFinished\(\)](#), [on_spinBox_Elitisme_editingFinished\(\)](#), [on_spinBox_HiddenNeurons_editingFinished\(\)](#), [on_spinBox_InputNeurons_editingFinished\(\)](#), [on_spinBox_MaxGen_editingFinished\(\)](#), [on_spinBox_NoFrames_editingFinished\(\)](#), [on_spinBox_NoShots_editingFinished\(\)](#), [on_spinBox_OutputNeurons_editingFinished\(\)](#), [on_spinBox_PopSize_editingFinished\(\)](#), [openTab\(\)](#), and [~DialogSettings\(\)](#).

The documentation for this class was generated from the following files:

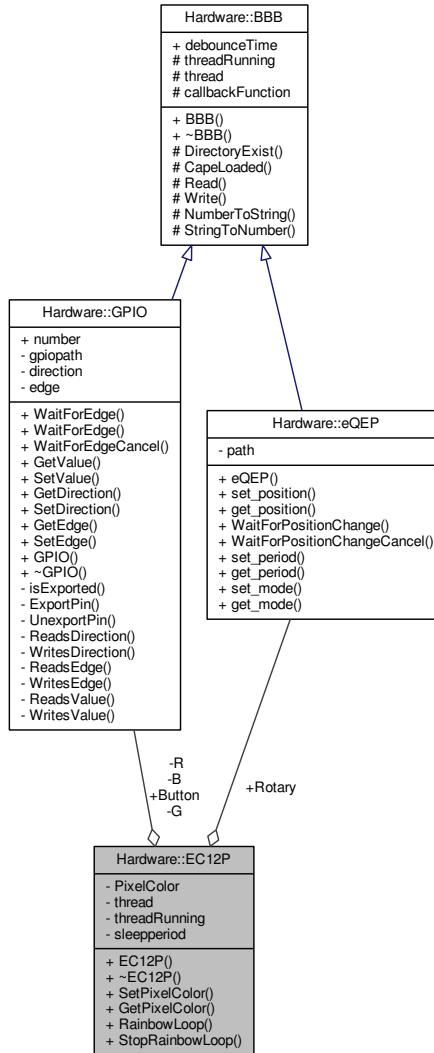
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialogsettings.h](#)

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialogsettings.cpp](#)

6.20 Hardware::EC12P Class Reference

```
#include <EC12P.h>
```

Collaboration diagram for Hardware::EC12P:



Public Types

- enum **Color** {
 Red, Pink, Blue, SkyBlue,
 Green, Yellow, White, None
 }

Public Member Functions

- EC12P ()**
- ~EC12P ()**
- void **SetPixelColor (Color value)**
- Color GetPixelColor ()**
- void **RainbowLoop (int sleepPeriod)**
- void **StopRainbowLoop ()**

Public Attributes

- `eQEP Rotary {eQEP2, eQEP::eQEP_Mode_Absolute}`
- `GPIO Button {68}`

Private Attributes

- `Color PixelColor`
- `GPIO R {31}`
- `GPIO B {48}`
- `GPIO G {51}`
- `pthread_t thread`
- `bool threadRunning`
- `int sleepperiod`

Friends

- `void * colorLoop (void *value)`

6.20.1 Detailed Description

Definition at line 23 of file [EC12P.h](#).

6.20.2 Member Enumeration Documentation

6.20.2.1 enum Hardware::EC12P::Color

Enumerator indicating the color of the encoder shaft

Enumerator

- Red** Red
- Pink** Pink
- Blue** Blue
- SkyBlue** SkyBlue
- Green** Green
- Yellow** Yellow
- White** White
- None** Off

Definition at line 29 of file [EC12P.h](#).

6.20.3 Constructor & Destructor Documentation

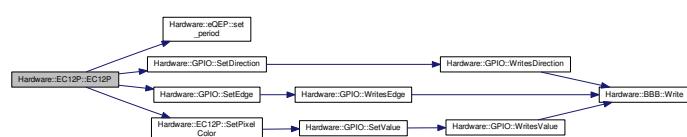
6.20.3.1 EC12P::EC12P()

Constructor

Definition at line 12 of file [EC12P.cpp](#).

References [B](#), [Button](#), [G](#), [Hardware::GPIO::Input](#), [None](#), [Hardware::GPIO::Output](#), [R](#), [Hardware::GPIO::Rising](#), [Rotary](#), [Hardware::eQEP::set<=period\(\)](#), [Hardware::GPIO::SetDirection\(\)](#), [Hardware::GPIO::SetEdge\(\)](#), [SetPixelColor\(\)](#), and [threadRunning](#).

Here is the call graph for this function:



6.20.3.2 `EC12P::~EC12P()`

De-constructor

Definition at line 30 of file [EC12P.cpp](#).

6.20.4 Member Function Documentation

6.20.4.1 `Color Hardware::EC12P::GetPixelColor() [inline]`

Definition at line 41 of file [EC12P.h](#).

6.20.4.2 `void EC12P::RainbowLoop(int sleeperperiod)`

Loops through all the colors except of as a thread

Definition at line 82 of file [EC12P.cpp](#).

References [colorLoop](#), [sleepperiod](#), [thread](#), and [threadRunning](#).

6.20.4.3 `void EC12P::SetPixelColor(Color value)`

Set the shaft color

Parameters

<code>value</code>	as Color enumerator
--------------------	---------------------

Definition at line 35 of file [EC12P.cpp](#).

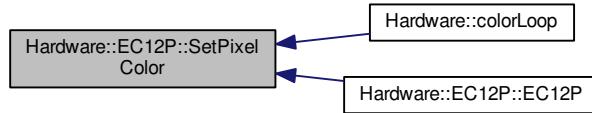
References [B](#), [Blue](#), [G](#), [Green](#), [Hardware::GPIO::High](#), [Hardware::GPIO::Low](#), [None](#), [Pink](#), [PixelColor](#), [R](#), [Red](#), [Hardware::GPIO::SetValue\(\)](#), [SkyBlue](#), [White](#), and [Yellow](#).

Referenced by [Hardware::colorLoop\(\)](#), and [EC12P\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.20.4.4 `void Hardware::EC12P::StopRainbowLoop() [inline]`

Definition at line 44 of file [EC12P.h](#).

6.20.5 Friends And Related Function Documentation

6.20.5.1 `void* colorLoop(void * value) [friend]`

The thread function that runs trough all the colors

Definition at line 91 of file [EC12P.cpp](#).

Referenced by [RainbowLoop\(\)](#).

6.20.6 Member Data Documentation

6.20.6.1 GPIO Hardware::EC12P::B {48} [private]

Blue LED

Definition at line 53 of file [EC12P.h](#).

Referenced by [EC12P\(\)](#), and [SetPixelColor\(\)](#).

6.20.6.2 GPIO Hardware::EC12P::Button {68}

The pushbutton

Definition at line 47 of file [EC12P.h](#).

Referenced by [EC12P\(\)](#).

6.20.6.3 GPIO Hardware::EC12P::G {51} [private]

Green LED

Definition at line 54 of file [EC12P.h](#).

Referenced by [EC12P\(\)](#), and [SetPixelColor\(\)](#).

6.20.6.4 Color Hardware::EC12P::PixelColor [private]

Current shaft color

Definition at line 50 of file [EC12P.h](#).

Referenced by [SetPixelColor\(\)](#).

6.20.6.5 GPIO Hardware::EC12P::R {31} [private]

Red LED

Definition at line 52 of file [EC12P.h](#).

Referenced by [EC12P\(\)](#), and [SetPixelColor\(\)](#).

6.20.6.6 eQEP Hardware::EC12P::Rotary {eQEP2, eQEP::eQEP_Mode_Absolute}

The encoder

Definition at line 46 of file [EC12P.h](#).

Referenced by [EC12P\(\)](#).

6.20.6.7 int Hardware::EC12P::sleepperiod [private]

Sleep period

Definition at line 58 of file [EC12P.h](#).

Referenced by [Hardware::colorLoop\(\)](#), and [RainbowLoop\(\)](#).

6.20.6.8 pthread_t Hardware::EC12P::thread [private]

the thread

Definition at line 56 of file [EC12P.h](#).

Referenced by [RainbowLoop\(\)](#).

6.20.6.9 bool Hardware::EC12P::threadRunning [private]

Bool used to stop the thread

Definition at line 57 of file [EC12P.h](#).

Referenced by [Hardware::colorLoop\(\)](#), [EC12P\(\)](#), and [RainbowLoop\(\)](#).

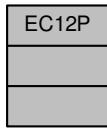
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.cpp](#)

6.21 EC12P Class Reference

```
#include <EC12P.h>
```

Collaboration diagram for EC12P:



6.21.1 Detailed Description

Interaction with the sparkfun RGB encoder

The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12Ph](#)

6.22 EmtpyImageException Class Reference

```
#include <EmptyImageException.h>
```

Collaboration diagram for EmtpyImageException:



6.22.1 Detailed Description

Exception class which is thrown when operations are about to start on a empty image.

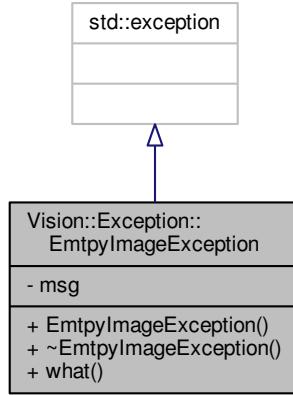
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/EmptyImageException.h](#)

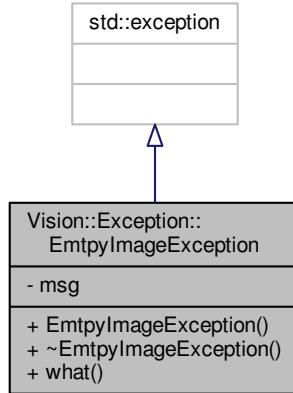
6.23 Vision::Exception::EmtpyImageException Class Reference

```
#include <EmptyImageException.h>
```

Inheritance diagram for Vision::Exception::EmtpyImageException:



Collaboration diagram for Vision::Exception::EmtpyImageException:



Public Member Functions

- `EmtpyImageException` (string m="Empty Image!")
- `~EmtpyImageException` () `_GLIBCXX_USE_NOEXCEPT`
- const char * `what` () const `_GLIBCXX_USE_NOEXCEPT`

Private Attributes

- string `msg`

6.23.1 Detailed Description

Definition at line 22 of file [EmptyImageException.h](#).

6.23.2 Constructor & Destructor Documentation

6.23.2.1 `Vision::Exception::EmptyImageException::EmptyImageException(string m = "Empty Image!")` [inline]

Definition at line 24 of file [EmptyImageException.h](#).

6.23.2.2 `Vision::Exception::EmptyImageException::~EmptyImageException()` [inline]

Definition at line 25 of file [EmptyImageException.h](#).

6.23.3 Member Function Documentation

6.23.3.1 `const char* Vision::Exception::EmptyImageException::what() const` [inline]

Definition at line 26 of file [EmptyImageException.h](#).

6.23.4 Member Data Documentation

6.23.4.1 `string Vision::Exception::EmptyImageException::msg` [private]

Definition at line 26 of file [EmptyImageException.h](#).

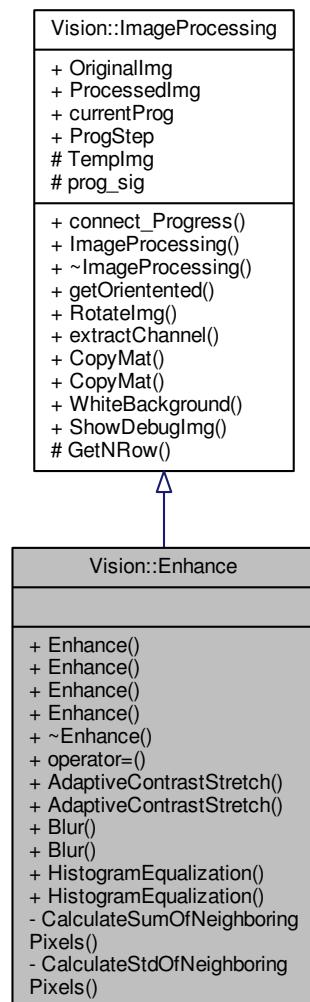
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/EmptyImageException.h](#)

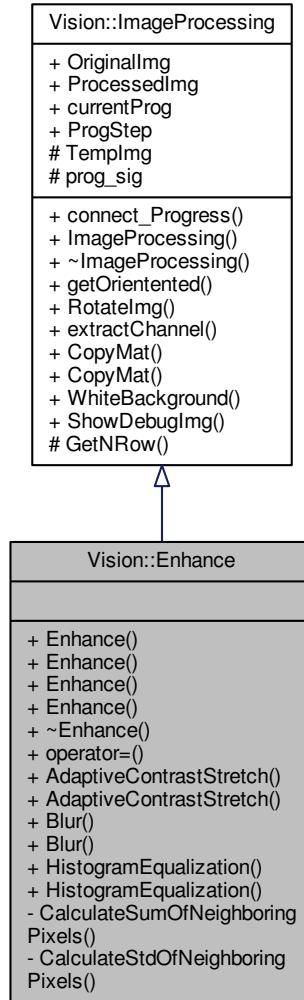
6.24 Vision::Enhance Class Reference

```
#include <Enhance.h>
```

Inheritance diagram for Vision::Enhance:



Collaboration diagram for Vision::Enhance:



Public Types

- enum `EnhanceOperation` { `_AdaptiveContrastStretch`, `_Blur`, `_HistogramEqualization` }

Public Member Functions

- `Enhance ()`
- `Enhance (const Mat &src)`
- `Enhance (const Mat &src, Mat &dst, uint8_t kernelsize=9, float factor=1.0, EnhanceOperation operation=_Blur)`
- `Enhance (const Enhance &rhs)`
- `~Enhance ()`
- `Enhance & operator= (Enhance rhs)`
- `void AdaptiveContrastStretch (uint8_t kernelsize, float factor, bool chain=false)`
- `void AdaptiveContrastStretch (const Mat &src, Mat &dst, uint8_t kernelsize, float factor)`
- `void Blur (uint8_t kernelsize, bool chain=false)`
- `void Blur (const Mat &src, Mat &dst, uint8_t kernelsize)`

-
- void [HistogramEqualization](#) (bool chain=false)
 - void [HistogramEqualization](#) (const Mat &src, Mat &dst)

Private Member Functions

- void [CalculateSumOfNeighboringPixels](#) (uchar *O, int i, int hKsize, int nCols, [uint32_t](#) &sum)
- float [CalculateStdOfNeighboringPixels](#) (uchar *O, int i, int hKsize, int nCols, int noNeighboursPix, float mean)

Additional Inherited Members

6.24.1 Detailed Description

Definition at line 18 of file [Enhance.h](#).

6.24.2 Member Enumeration Documentation

6.24.2.1 enum Vision::Enhance::EnhanceOperation

Enumerator indicating the requested enhancement operation

Enumerator

- [_AdaptiveContrastStretch](#) custom adaptive contrast stretch operation
- [_Blur](#) Blur operation
- [_HistogramEqualization](#) Histogram equalization

Definition at line 27 of file [Enhance.h](#).

6.24.3 Constructor & Destructor Documentation

6.24.3.1 Enhance::Enhance ()

Constructor

Definition at line 15 of file [Enhance.cpp](#).

6.24.3.2 Enhance::Enhance (const Mat & src)

Constructor

Parameters

<i>src</i>	cv::Mat source image
------------	----------------------

Definition at line 20 of file [Enhance.cpp](#).

References [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

6.24.3.3 Vision::Enhance::Enhance (const Mat & src, Mat & dst, [uint8_t](#) kernelsize = 9, float factor = 1.0, [EnhanceOperation](#) operation = [_Blur](#))

6.24.3.4 Enhance::Enhance (const Enhance & rhs)

Definition at line 25 of file [Enhance.cpp](#).

References [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TemplImg](#).

6.24.3.5 Enhance::~Enhance ()

Dec-constructor

Definition at line 60 of file [Enhance.cpp](#).

6.24.4 Member Function Documentation

6.24.4.1 void Vision::Enhance::AdaptiveContrastStretch ([uint8_t](#) kernelsize, float factor, bool chain = false)

6.24.4.2 void Vision::Enhance::AdaptiveContrastStretch (const Mat & src, Mat & dst, [uint8_t](#) kernelsize, float factor)

6.24.4.3 void Vision::Enhance::Blur (`uint8_t kernelsize, bool chain = false`)
 6.24.4.4 void Vision::Enhance::Blur (`const Mat & src, Mat & dst, uint8_t kernelsize`)
 6.24.4.5 float Enhance::CalculateStdOfNeighboringPixels (`uchar * O, int i, int hKsize, int nCols, int noNeighboursPix, float mean`) [private]

Calculate the standard deviation of the neighboring pixels

Parameters

<i>O</i>	uchar pointer to the current pixel of the original image
<i>i</i>	current counter
<i>hKsize</i>	half the kernelsize
<i>nCols</i>	total number of columns
<i>noNeighboursPix</i>	total number of neighboring pixels
<i>mean</i>	mean value of the neighboring pixels

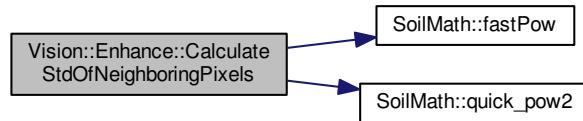
Returns

standard deviation

Definition at line 80 of file [Enhance.cpp](#).

References [SoilMath::fastPow\(\)](#), and [SoilMath::quick_pow2\(\)](#).

Here is the call graph for this function:



6.24.4.6 void Enhance::CalculateSumOfNeighboringPixels (`uchar * O, int i, int hKsize, int nCols, uint32_t & sum`) [private]

Calculate the sum of the neighboring pixels

Parameters

<i>O</i>	uchar pointer to the current pixel of the original image
<i>i</i>	current counter
<i>hKsize</i>	half the kernelsize
<i>nCols</i>	total number of columns
<i>sum</i>	Total sum of the neighboringpixels

Definition at line 105 of file [Enhance.cpp](#).

6.24.4.7 void Enhance::HistogramEqualization (`bool chain = false`)

Stretches the image using a histogram

Parameters

<i>chain</i>	use the results from the previous operation default value = false;
--------------	--

Definition at line 277 of file [Enhance.cpp](#).

References [CHAIN_PROCESS](#), [EMPTY_CHECK](#), [SoilMath::Stats< T1, T2, T3 >::max](#), [SoilMath::Stats< T1, T2, T3 >::min](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

6.24.4.8 void Vision::Enhance::HistogramEqualization (`const Mat & src, Mat & dst`)

6.24.4.9 Enhance & Enhance::operator= (`Enhance rhs`)

Definition at line 62 of file [Enhance.cpp](#).

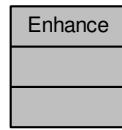
References [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TemplImg](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.cpp](#)

6.25 Enhance Class Reference

Collaboration diagram for Enhance:



6.25.1 Detailed Description

class which enhances a greyscale cv::Mat image

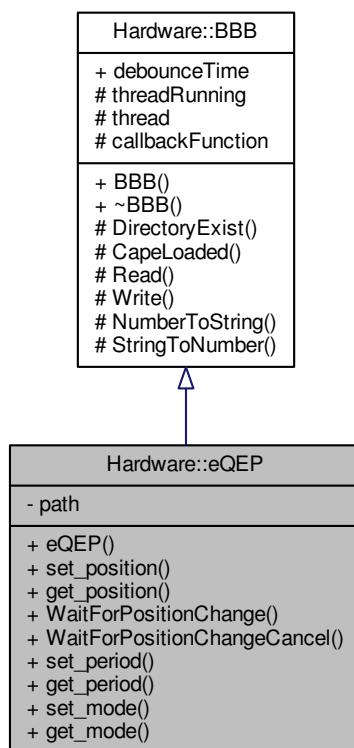
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.cpp](#)

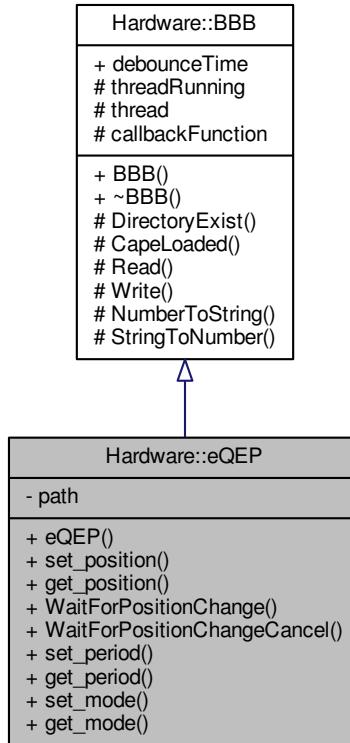
6.26 Hardware::eQEP Class Reference

```
#include <eqep.h>
```

Inheritance diagram for Hardware::eQEP:



Collaboration diagram for Hardware::eQEP:



Public Types

- enum `eQEP_Mode` { `eQEP_Mode_Absolute` = 0, `eQEP_Mode_Relative` = 1, `eQEP_Mode_Error` = 2 }

Public Member Functions

- `eQEP (std::string _path, eQEP_Mode _mode)`
- `void set_position (int32_t position)`
- `int32_t get_position (bool _poll=true)`
- `int WaitForPositionChange (CallbackType callback)`
- `void WaitForPositionChangeCancel ()`
- `void set_period (long long unsigned int period)`
- `uint64_t get_period ()`
- `void set_mode (eQEP_Mode mode)`
- `eQEP_Mode get_mode ()`

Private Attributes

- `std::string path`

Friends

- `void * threadedPolleqep (void *value)`

Additional Inherited Members

6.26.1 Detailed Description

Definition at line 39 of file [eqep.h](#).

6.26.2 Member Enumeration Documentation

6.26.2.1 enum Hardware::eQEP::eQEP_Mode

Enumerator

eQEP_Mode_Absolute

eQEP_Mode_Relative

eQEP_Mode_Error

Definition at line 45 of file [eqep.h](#).

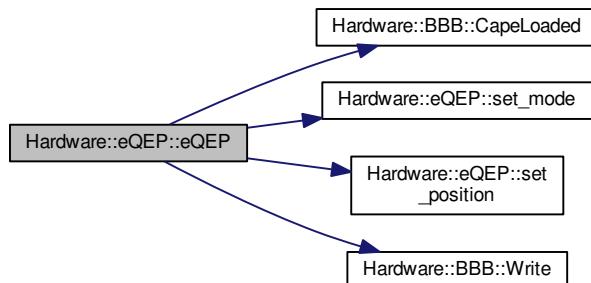
6.26.3 Constructor & Destructor Documentation

6.26.3.1 Hardware::eQEP::eQEP (std::string _path, eQEP::eQEP_Mode _mode)

Definition at line 42 of file [eqep.cpp](#).

References [Hardware::BBB::CapeLoaded\(\)](#), [eQEP0](#), [eQEP1](#), [eQEP2](#), [set_mode\(\)](#), [set_position\(\)](#), [SLOTS](#), and [Hardware::BBB::Write\(\)](#).

Here is the call graph for this function:



6.26.4 Member Function Documentation

6.26.4.1 eQEP::eQEP_Mode Hardware::eQEP::get_mode ()

Definition at line 219 of file [eqep.cpp](#).

References [eQEP_Mode_Error](#), and [path](#).

6.26.4.2 uint64_t Hardware::eQEP::get_period ()

Definition at line 195 of file [eqep.cpp](#).

References [path](#).

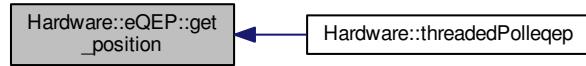
6.26.4.3 int32_t Hardware::eQEP::get_position (bool _poll = true)

Definition at line 137 of file [eqep.cpp](#).

References [path](#).

Referenced by [Hardware::threadedPolleqep\(\)](#).

Here is the caller graph for this function:



6.26.4.4 void Hardware::eQEP::set_mode (eQEP::eQEP_Mode _mode)

Definition at line 105 of file [eqep.cpp](#).

References [path](#).

Referenced by [eQEP\(\)](#).

Here is the caller graph for this function:



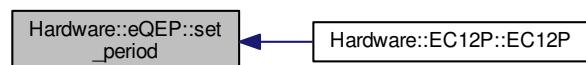
6.26.4.5 void Hardware::eQEP::set_period (long long unsigned int *period*)

Definition at line 85 of file [eqep.cpp](#).

References [path](#).

Referenced by [Hardware::EC12P::EC12P\(\)](#).

Here is the caller graph for this function:



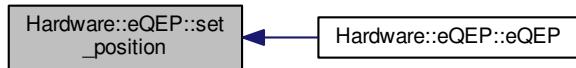
6.26.4.6 void Hardware::eQEP::set_position (int32_t *position*)

Definition at line 65 of file [eqep.cpp](#).

References [path](#).

Referenced by [eQEP\(\)](#).

Here is the caller graph for this function:



6.26.4.7 int Hardware::eQEP::WaitForPositionChange (CallbackType *callback*)

Definition at line 124 of file [eqep.cpp](#).

References [Hardware::BBB::callbackFunction](#), [Hardware::BBB::thread](#), [threadedPolleqep](#), and [Hardware::BBB::threadRunning](#).

6.26.4.8 void Hardware::eQEP::WaitForPositionChangeCancel () [inline]

Definition at line 68 of file [eqep.h](#).

References [Hardware::BBB::threadRunning](#).

6.26.5 Friends And Related Function Documentation

6.26.5.1 void* threadedPolleqep (void * *value*) [friend]

Definition at line 242 of file [eqep.cpp](#).

Referenced by [WaitForPositionChange\(\)](#).

6.26.6 Member Data Documentation

6.26.6.1 std::string Hardware::eQEP::path [private]

Definition at line 41 of file [eqep.h](#).

Referenced by [get_mode\(\)](#), [get_period\(\)](#), [get_position\(\)](#), [set_mode\(\)](#), [set_period\(\)](#), and [set_position\(\)](#).

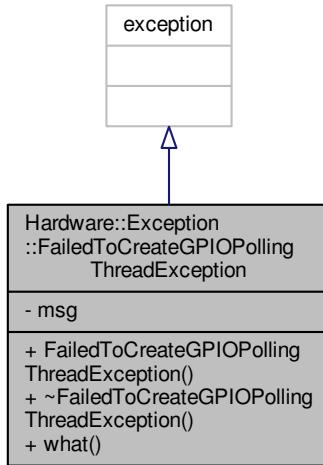
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/eqep.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/eqep.cpp](#)

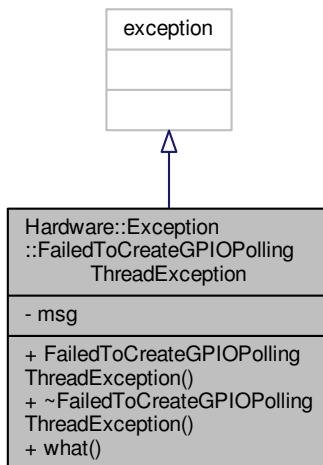
6.27 Hardware::Exception::FailedToCreateGPIOPollingThreadException Class Reference

```
#include <FailedToCreateGPIOPollingThreadException.h>
```

Inheritance diagram for Hardware::Exception::FailedToCreateGPIOPollingThreadException:



Collaboration diagram for Hardware::Exception::FailedToCreateGPIOPollingThreadException:



Public Member Functions

- `FailedToCreateGPIOPollingThreadException` (string m="Failed to create GPIO polling thread!")
- `~FailedToCreateGPIOPollingThreadException` () `_GLIBCXX_USE_NOEXCEPT`
- `const char * what` () `const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- `string msg`

6.27.1 Detailed Description

Definition at line 17 of file [FailedToCreateGPIOPollingThreadException.h](#).

6.27.2 Constructor & Destructor Documentation

6.27.2.1 `Hardware::Exception::FailedToCreateGPIOPollingThreadException::FailedToCreateGPIOPollingThreadException (string m = "Failed to create GPIO polling thread!") [inline]`

Definition at line 19 of file [FailedToCreateGPIOPollingThreadException.h](#).

6.27.2.2 `Hardware::Exception::FailedToCreateGPIOPollingThreadException::~FailedToCreateGPIOPollingThreadException () [inline]`

Definition at line 22 of file [FailedToCreateGPIOPollingThreadException.h](#).

6.27.3 Member Function Documentation

6.27.3.1 `const char* Hardware::Exception::FailedToCreateGPIOPollingThreadException::what () const [inline]`

Definition at line 23 of file [FailedToCreateGPIOPollingThreadException.h](#).

6.27.4 Member Data Documentation

6.27.4.1 `string Hardware::Exception::FailedToCreateGPIOPollingThreadException::msg [private]`

Definition at line 23 of file [FailedToCreateGPIOPollingThreadException.h](#).

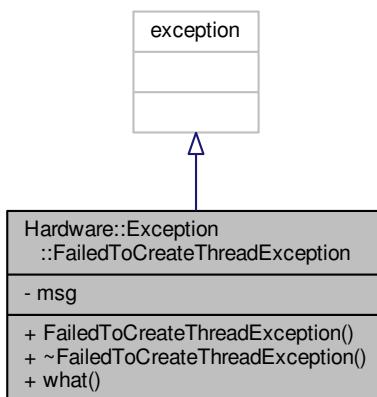
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/FailedToCreateGPIOPollingThreadException.h](#)

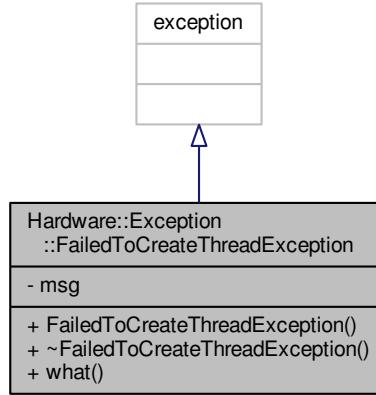
6.28 Hardware::Exception::FailedToCreateThreadException Class Reference

```
#include <FailedToCreateThreadException.h>
```

Inheritance diagram for Hardware::Exception::FailedToCreateThreadException:



Collaboration diagram for Hardware::Exception::FailedToCreateThreadException:



Public Member Functions

- `FailedToCreateThreadException` (string *m*="Couldn't create the thread!")
- `~FailedToCreateThreadException` () `_GLIBCXX_USE_NOEXCEPT`
- const char * `what` () const `_GLIBCXX_USE_NOEXCEPT`

Private Attributes

- string `msg`

6.28.1 Detailed Description

Definition at line 17 of file [FailedToCreateThreadException.h](#).

6.28.2 Constructor & Destructor Documentation

6.28.2.1 `Hardware::Exception::FailedToCreateThreadException::FailedToCreateThreadException` (string *m* = "Couldn't create the thread!") `[inline]`

Definition at line 19 of file [FailedToCreateThreadException.h](#).

6.28.2.2 `Hardware::Exception::FailedToCreateThreadException::~FailedToCreateThreadException` () `[inline]`

Definition at line 21 of file [FailedToCreateThreadException.h](#).

6.28.3 Member Function Documentation

6.28.3.1 `const char* Hardware::Exception::FailedToCreateThreadException::what` () const `[inline]`

Definition at line 22 of file [FailedToCreateThreadException.h](#).

6.28.4 Member Data Documentation

6.28.4.1 `string Hardware::Exception::FailedToCreateThreadException::msg` `[private]`

Definition at line 22 of file [FailedToCreateThreadException.h](#).

The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/[FailedToCreateThreadException.h](#)

6.29 SoilMath::FFT Class Reference

Fast Fourier Transform class.

```
#include <FFT.h>
```

Collaboration diagram for SoilMath::FFT:

SoilMath::FFT	
- fftDescriptors	
- complexcontour	
- Img	
+ FFT()	
+ ~FFT()	
+ GetDescriptors()	
- Contour2Complex()	
- Neighbors()	
- fft()	
- ifft()	

Public Member Functions

- [FFT \(\)](#)
Standard constructor.
- [~FFT \(\)](#)
Standard deconstructor.
- [ComplexVect_t GetDescriptors \(const cv::Mat &img\)](#)
Transforming the img to the frequency domain and returning the Fourier Descriptors.

Private Member Functions

- [ComplexVect_t Contour2Complex \(const cv::Mat &img, float centerCol, float centerRow\)](#)
Contour2Complex a private function which translates a continous contour image to a vector of complex values. The contour is found using a depth first search with extension list. The alghorithm is based upon [MIT opencourseware 6-034-artificial-intelligence lecture 4](#)
- [iContour_t Neighbors \(uchar *O, int pixel, uint32_t columns, uint32_t rows\)](#)
Neighbors a private function returning the neighboring pixels which belong to a contour.
- [void fft \(ComplexArray_t &CA\)](#)
fft a private function calculating the Fast Fourier Transform let m be an integer and let N = 2^m also CA = [x₀, ..., x_{N-1}] is an N dimensional complex vector let $\omega = \exp\left(\frac{-2\pi i}{N}\right)$ then $c_k = \frac{1}{N} \sum_{j=0}^{j=N-1} CA_j \omega^{jk}$
- [void ifft \(ComplexArray_t &CA\)](#)
ifft

Private Attributes

- [ComplexVect_t fftDescriptors](#)
- [ComplexVect_t complexcontour](#)
- [cv::Mat Img](#)

6.29.1 Detailed Description

Fast Fourier Transform class.

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.

Definition at line 31 of file [FFT.h](#).

6.29.2 Constructor & Destructor Documentation

6.29.2.1 SoilMath::FFT::FFT()

Standard constructor.

Definition at line 11 of file [FFT.cpp](#).

6.29.2.2 SoilMath::FFT::~FFT()

Standard deconstructor.

Definition at line 13 of file [FFT.cpp](#).

6.29.3 Member Function Documentation

6.29.3.1 ComplexVect_t SoilMath::FFT::Contour2Complex (const cv::Mat & img, float centerCol, float centerRow) [private]

Contour2Complex a private function which translates a continous contour image to a vector of complex values. The contour is found using a depth first search with extension list. The algorithim is based upon [MIT opencourseware 6-034-artificial-intelligence lecture 4](#)

Parameters

<i>img</i>	contour in the form of a cv::Mat type CV_8UC1. Which should consist of a continous contour. $\{img \in \mathbb{Z} 0 \leq img \leq 1\}$
<i>centerCol</i>	centre of the contour X value
<i>centerRow</i>	centre of the contour Y value

Returns

a vector with complex values, represing the contour as a function

Definition at line 64 of file [FFT.cpp](#).

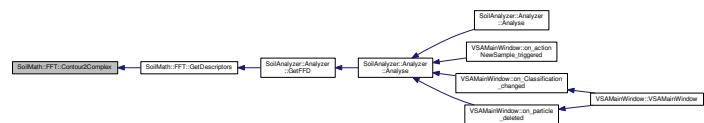
References [EXCEPTION_NO_CONTOUR_FOUND](#), [EXCEPTION_NO_CONTOUR_FOUND_NR](#), and [Neighbors\(\)](#).

Referenced by [GetDescriptors\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.29.3.2 void SoilMath::FFT::fft (ComplexArray_t & CA) [private]

fft a private function calculating the Fast Fourier Transform let m be an integer and let $N = 2^m$ also $CA = [x_0, \dots, x_{N-1}]$ is an N dimensional complex vector let $\omega = \exp\left(\frac{-2\pi i}{N}\right)$ then $c_k = \frac{1}{N} \sum_{j=0}^{j=N-1} CA_j \omega^{jk}$

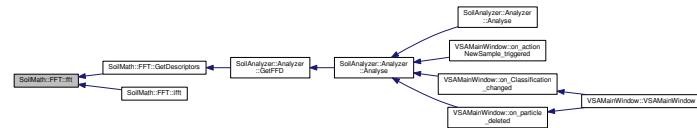
Parameters

CA | a $CA = [x_0, \dots, x_{N-1}]$ is an N dimensional complex vector

Definition at line 149 of file FFT.cpp.

Referenced by [GetDescriptors\(\)](#), and [ifft\(\)](#).

Here is the caller graph for this function:



6.29.3.3 ComplexVect_t SoilMath::FFT::GetDescriptors (const cv::Mat & *img*)

Transforming the img to the frequency domain and returning the Fourier Descriptors.

Parameters

img contour in the form of a cv::Mat type CV_8UC1. Which should consist of a continuous contour. { $img \in \mathbb{Z} | 0 \leq img \leq 1 \}$

Returns

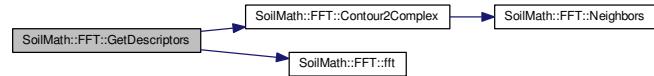
a vector with complex values, representing the contour in the frequency domain, expressed as Fourier Descriptors

Definition at line 15 of file [FFT.cpp](#).

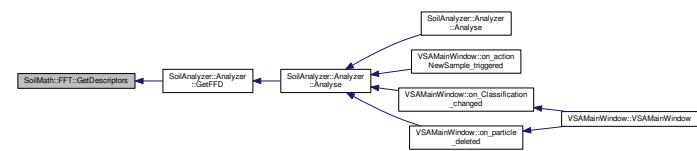
References [complexcontour](#), [Contour2Complex\(\)](#), [fft\(\)](#), and [fftDescriptors](#).

Referenced by [SoilAnalyzer::Analyzer::GetFFD\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.29.3.4 void SoilMath::FFT::ifft (ComplexArray_t & CA) [private]

ifft

Parameters

CA

Definition at line 169 of file [FFT.cpp](#).

References [fft\(\)](#).

Here is the call graph for this function:



6.29.3.5 `iContour_t SoilMath::FFT::Neighbors (uchar * O, int pixel, uint32_t columns, uint32_t rows)` [private]

Neighbors a private function returning the neighboring pixels which belong to a contour.

Parameters

<i>O</i>	uchar pointer to the data
<i>pixel</i>	current counter
<i>columns</i>	total number of columns
<i>rows</i>	total number of rows

Returns

Definition at line 43 of file [FFT.cpp](#).

Referenced by [Contour2Complex\(\)](#).

Here is the caller graph for this function:



6.29.4 Member Data Documentation

6.29.4.1 `ComplexVect_t SoilMath::FFT::complexcontour` [private]

Vector with complex values which represent the contour

Definition at line 59 of file [FFT.h](#).

Referenced by [GetDescriptors\(\)](#).

6.29.4.2 `ComplexVect_t SoilMath::FFT::fftDescriptors` [private]

Vector with complex values which represent the descriptors

Definition at line 56 of file [FFT.h](#).

Referenced by [GetDescriptors\(\)](#).

6.29.4.3 `cv::Mat SoilMath::FFT::img` [private]

Img which will be analysed

Definition at line 61 of file [FFT.h](#).

The documentation for this class was generated from the following files:

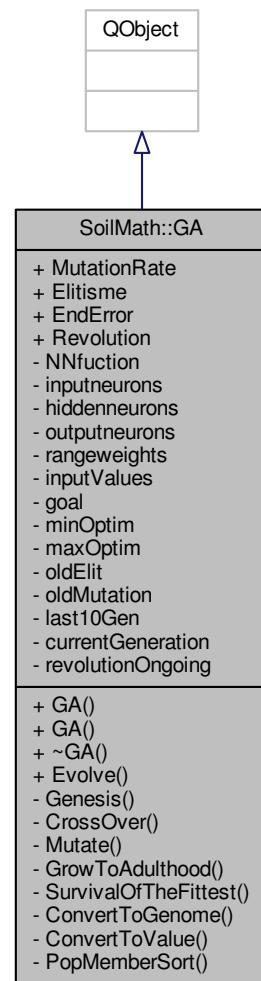
• [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.h](#)

• [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.cpp](#)

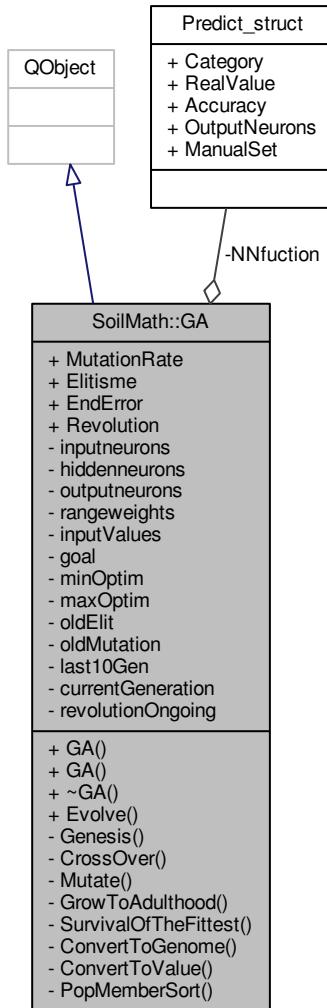
6.30 SoilMath::GA Class Reference

#include <GA.h>

Inheritance diagram for SoilMath::GA:



Collaboration diagram for SoilMath::GA:



Signals

- void `learnErrorUpdate` (double newError)

Public Member Functions

- `GA ()`
`GA` Standard constructor.
- `GA (NNfunctionType nnfunction, uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)`
`GA` Construction with a Neural Network initializers.
- `~GA ()`
`GA` standard de constructor.
- void `Evolve` (const `InputLearnVector_t` &inputValues, `Weight_t` &weights, `MinMaxWeight_t` rangeweights, `OutputLearnVector_t` &goal, `uint32_t` maxGenerations=200, `uint32_t` popSize=30)

Evolve Darwin would be proud!!! This function creates a population and iterates through the generation till the maximum number of iterations has been reached of the error is acceptable.

Public Attributes

- float `MutationRate` = 0.075f
- `uint32_t` `Elitisme` = 4
- float `EndError` = 0.001f
- bool `Revolution` = true

Private Member Functions

- `Population_t` `Genesis` (const `Weight_t` &weights, `uint32_t` popSize)

Genesis private function which is the spark of life, using a random seed.
- void `CrossOver` (`Population_t` &pop)

CrossOver a private function where the partners mate with each other The values or `PopMember_t` are expressed as bits or are cut at the point Crossover the population members are paired with the nearest neighbor and new members are created pairing the `Genome_t` of each other at the Crossover point. Afterwards all the top tiers partners are allowed to mate again.
- void `Mutate` (`Population_t` &pop)

Mutate a private function where individual bits from the `Genome_t` are mutated at a random uniform distribution event defined by the `MUTATIONRATE`.
- void `GrowToAdulthood` (`Population_t` &pop, float &totalFitness)

GrowToAdulthood a private function where the new population members serve as the input for the Neural Network prediction function. The results are weight against the goal and this weight determine the fitness of the population member.
- bool `SurvivalOfTheFittest` (`Population_t` &pop, float &totalFitness)

SurvivalOfTheFittest a private function where a battle to the death commences The fittest population members have the best chance of survival. Death is instigated with a random uniform distribution. The elite members don't partake in this destruction The `ELITISME` rate indicate how many top tier members survive this catastrophic event.
- template<typename T>
 `Genome_t` `ConvertToGenome` (T value, std::pair< T, T > range)

Conversion of the value of type T to `Genome_t`.
- template<typename T>
 T `ConvertToValue` (`Genome_t` gen, std::pair< T, T > range)

Conversion of the `Genome` to a value.

Static Private Member Functions

- static bool `PopMemberSort` (`PopMember_t` i, `PopMember_t` j)

PopMemberSort a private function where the members are sorted according to their fitness ranking.

Private Attributes

- `NNfunctionType` `NNfuction`
- `uint32_t` `inputneurons`
- `uint32_t` `hiddenneurons`
- `uint32_t` `outputneurons`
- `MinMaxWeight_t` `rangeweights`
- `InputLearnVector_t` `inputValues`
- `OutputLearnVector_t` `goal`
- float `minOptim` = 0
- float `maxOptim` = 0
- `uint32_t` `oldElit` = 0
- float `oldMutation` = 0.
- std::list< double > `last10Gen`
- `uint32_t` `currentGeneration` = 0
- bool `revolutionOngoing` = false

6.30.1 Detailed Description

Definition at line 36 of file `GA.h`.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 SoilMath::GA::GA ()

[GA](#) Standard constructor.

Definition at line 11 of file [GA.cpp](#).

6.30.2.2 SoilMath::GA::GA ([NNfunctionType](#) *nnfunction*, [uint32_t](#) *inputneurons*, [uint32_t](#) *hiddenneurons*, [uint32_t](#) *outputneurons*)

[GA](#) Construction with a Neural Network initializers.

Parameters

<i>nnfunction</i>	the Neural Network prediction function which results will be optimized
<i>inputneurons</i>	the number of input neurons in the Neural Network don't count the bias
<i>hiddenneurons</i>	the number of hidden neurons in the Neural Network don't count the bias
<i>outputneurons</i>	the number of output neurons in the Neural Network

Definition at line 13 of file [GA.cpp](#).

References [hiddenneurons](#), [inputneurons](#), [NNfunction](#), and [outputneurons](#).

6.30.2.3 SoilMath::GA::~GA ()

[GA](#) standard de constructor.

Definition at line 21 of file [GA.cpp](#).

6.30.3 Member Function Documentation

6.30.3.1 [template<typename T > Genome_t](#) [SoilMath::GA::ConvertToGenome](#) (*T* *value*, [std::pair](#)<*T*, *T*> *range*) [inline], [private]

[Conversion](#) of the value of type *T* to *Genome_t*.

Usage: Use [ConvertToGenome<Type>\(type, range\)](#)

Parameters

<i>value</i>	The current value which should be converted to a <i>Genome_t</i>
<i>range</i>	the range in which the value should fall, this is to have a <i>Genome_t</i> which utilizes the complete range 0000...n till 1111...n

Definition at line 191 of file [GA.h](#).

6.30.3.2 [template<typename T > T](#) [SoilMath::GA::ConvertToValue](#) (*Genome_t* *gen*, [std::pair](#)<*T*, *T*> *range*) [inline], [private]

[Conversion](#) of the *Genome* to a value.

Usage: use [ConvertToValue<Type>\(genome, range\)](#)

Parameters

<i>gen</i>	is the <i>Genome</i> which is to be converted
<i>range</i>	is the range in which the value should fall

Definition at line 205 of file [GA.h](#).

6.30.3.3 [void](#) [SoilMath::GA::CrossOver](#) (*Population_t* & *pop*) [private]

[CrossOver](#) a private function where the partners mate with each other. The values or *PopMember_t* are expressed as bits or are cut at the point [CROSSOVER](#) the population members are paired with the nearest neighbor and new members are created pairing the *Genome_t* of each other at the [CROSSOVER](#) point. Afterwards all the top tiers partners are allowed to mate again.

Parameters

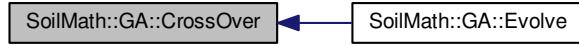
<i>pop</i>	reference to the population
------------	-----------------------------

Definition at line 72 of file [GA.cpp](#).

References [CROSSOVER](#), [GENE_MAX](#), and [PopMemberStruct::weightsGen](#).

Referenced by [Evolve\(\)](#).

Here is the caller graph for this function:



6.30.3.4 `void SoilMath::GA::Evolve (const InputLearnVector_t & inputValues, Weight_t & weights, MinMaxWeight_t rangeweights, OutputLearnVector_t & goal, uint32_t maxGenerations = 200, uint32_t popSize = 30)`

Evolve Darwin would be proud!!! This function creates a population and iterates through the generation till the maximum number off iterations has been reached of the error is acceptable.

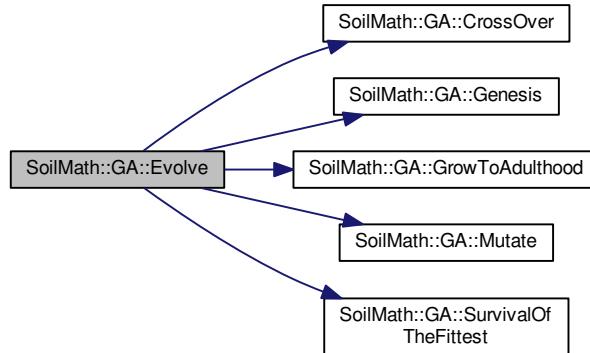
Parameters

<i>inputValues</i>	complex vector with a reference to the inputvalues
<i>weights</i>	reference to the vector of weights which will be optimized
<i>rangeweights</i>	reference to the range of weights, currently it doesn't support individual ranges this is because of the crossing
<i>goal</i>	target value towards the Neural Network prediction function will be optimized
<i>maxGenerations</i>	maximum number of iterations default value is 200
<i>popSize</i>	maximum number of population, this should be an even number

Definition at line 23 of file `GA.cpp`.

References [CrossOver\(\)](#), [Elitisme](#), [Genesis\(\)](#), [goal](#), [GrowToAdulthood\(\)](#), [inputValues](#), [maxOptim](#), [minOptim](#), [Mutate\(\)](#), [MutationRate](#), [oldElit](#), [oldMutation](#), [rangeweights](#), and [SurvivalOfTheFittest\(\)](#).

Here is the call graph for this function:



6.30.3.5 `Population_t SoilMath::GA::Genesis (const Weight_t & weights, uint32_t popSize) [private]`

Genesis private function which is the spark of live, using a random seed.

Parameters

<i>weights</i>	a reference to the used <code>Weight_t</code> vector
<i>rangeweights</i>	pointer to the range of weights, currently it doesn't support individual ranges
<i>popSize</i>	maximum number of population, this should be an even number

Returns

Definition at line 50 of file [GA.cpp](#).

References [rangeweights](#), [PopMemberStruct::weights](#), and [PopMemberStruct::weightsGen](#).

Referenced by [Evolve\(\)](#).

Here is the caller graph for this function:



6.30.3.6 void SoilMath::GA::GrowToAdulthood (Population_t & pop, float & totalFitness) [private]

`GrowToAdulthood` a private function where the new population members serve as the the input for the Neural Network prediction function. The results are weight against the goal and this weight determine the fitness of the population member.

Parameters

<code>pop</code>	reference to the population
<code>inputValues</code>	a <code>InputLearnVector_t</code> with a reference to the inputvalues
<code>rangeweights</code>	pointer to the range of weights, currently it doesn't support indivudal ranges
<code>goal</code>	a <code>Predict_t</code> type with the expected value
<code>totalFitness</code>	a reference to the total population fitness

Definition at line 152 of file [GA.cpp](#).

References [goal](#), [hiddeneurons](#), [inputneurons](#), [inputValues](#), [NNfuction](#), [Predict_struct::OutputNeurons](#), [outputneurons](#), and [rangeweights](#).

Referenced by [Evolve\(\)](#).

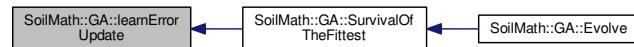
Here is the caller graph for this function:



6.30.3.7 void SoilMath::GA::learnErrorUpdate (double newError) [signal]

Referenced by [SurvivalOfTheFittest\(\)](#).

Here is the caller graph for this function:



6.30.3.8 void SoilMath::GA::Mutate (Population_t & pop) [private]

`Mutate` a private function where individual bits from the `Genome_t` are mutated at a random uniform distribution event defined by the `MUTATI-ONRATE`.

Parameters

<i>pop</i>	reference to the population
------------	-----------------------------

Definition at line 135 of file [GA.cpp](#).

References [GENE_MAX](#), and [MutationRate](#).

Referenced by [Evolve\(\)](#).

Here is the caller graph for this function:



6.30.3.9 static bool SoilMath::GA::PopMemberSort(PopMember_t i, PopMember_t j) [inline], [static], [private]

PopMemberSort a private function where the members are sorted according to there fitness ranking.

Parameters

<i>i</i>	left hand population member
<i>j</i>	right hand population member

Returns

true if the left member is closer to the goal as the right member.

Definition at line 178 of file [GA.h](#).

References [PopMemberStruct::Fitness](#).

6.30.3.10 bool SoilMath::GA::SurvivalOfTheFittest(Population_t & pop, float & totalFitness) [private]

SurvivalOfTheFittest a private function where a battle to the death commences The fittest population members have the best chance of survival. Death is instigated with a random uniform distribution. The elite members don't partake in this destruction The ELITISME rate indicate how many top tier members survive this catastrophic event.

Parameters

<i>inputValues</i>	a InputLearnVector_t with a reference to the inputvalues
<i>totalFitness</i>	a reference to the total population fitness

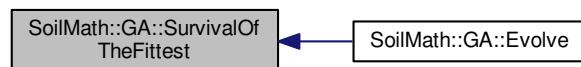
Returns

Definition at line 189 of file [GA.cpp](#).

References [currentGeneration](#), [Elitisme](#), [EndError](#), [PopMemberStruct::Fitness](#), [last10Gen](#), [learnErrorUpdate\(\)](#), [maxOptim](#), [minOptim](#), [MutationRate](#), [oldElit](#), [oldMutation](#), and [revolutionOngoing](#).

Referenced by [Evolve\(\)](#).

Here is the caller graph for this function:



6.30.4 Member Data Documentation

6.30.4.1 `uint32_t` `SoilMath::GA::currentGeneration = 0` [private]

Definition at line 105 of file [GA.h](#).

Referenced by [SurvivalOfTheFittest\(\)](#).

6.30.4.2 `uint32_t` `SoilMath::GA::Elitisme = 4`

total number of the elite bastard

Definition at line 41 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [SurvivalOfTheFittest\(\)](#).

6.30.4.3 `float` `SoilMath::GA::EndError = 0.001f`

acceptable error between last iteration

Definition at line 42 of file [GA.h](#).

Referenced by [SurvivalOfTheFittest\(\)](#).

6.30.4.4 `OutputLearnVector_t` `SoilMath::GA::goal` [private]

Definition at line 98 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.5 `uint32_t` `SoilMath::GA::hiddenneurons` [private]

the total number of hidden neurons

Definition at line 93 of file [GA.h](#).

Referenced by [GA\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.6 `uint32_t` `SoilMath::GA::inputneurons` [private]

the total number of input neurons

Definition at line 92 of file [GA.h](#).

Referenced by [GA\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.7 `InputLearnVector_t` `SoilMath::GA::inputValues` [private]

Definition at line 97 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.8 `std::list<double>` `SoilMath::GA::last10Gen` [private]

Definition at line 104 of file [GA.h](#).

Referenced by [SurvivalOfTheFittest\(\)](#).

6.30.4.9 `float` `SoilMath::GA::maxOptim = 0` [private]

Definition at line 101 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [SurvivalOfTheFittest\(\)](#).

6.30.4.10 `float` `SoilMath::GA::minOptim = 0` [private]

Definition at line 100 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [SurvivalOfTheFittest\(\)](#).

6.30.4.11 `float` `SoilMath::GA::MutationRate = 0.075f`

mutation rate

Definition at line 40 of file [GA.h](#).

Referenced by [Evolve\(\)](#), [Mutate\(\)](#), and [SurvivalOfTheFittest\(\)](#).

6.30.4.12 NNfunctionType SoilMath::GA::NNfunction [private]

The Neural Net work function

Definition at line 91 of file [GA.h](#).

Referenced by [GA\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.13 uint32_t SoilMath::GA::oldElit = 0 [private]

Definition at line 102 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [SurvivalOfTheFittest\(\)](#).

6.30.4.14 float SoilMath::GA::oldMutation = 0. [private]

Definition at line 103 of file [GA.h](#).

Referenced by [Evolve\(\)](#), and [SurvivalOfTheFittest\(\)](#).

6.30.4.15 uint32_t SoilMath::GA::outputneurons [private]

the total number of output neurons

Definition at line 94 of file [GA.h](#).

Referenced by [GA\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.16 MinMaxWeight_t SoilMath::GA::rangeweights [private]

Definition at line 96 of file [GA.h](#).

Referenced by [Evolve\(\)](#), [Genesis\(\)](#), and [GrowToAdulthood\(\)](#).

6.30.4.17 bool SoilMath::GA::Revolution = true

Definition at line 43 of file [GA.h](#).

6.30.4.18 bool SoilMath::GA::revolutionOngoing = false [private]

Definition at line 106 of file [GA.h](#).

Referenced by [SurvivalOfTheFittest\(\)](#).

The documentation for this class was generated from the following files:

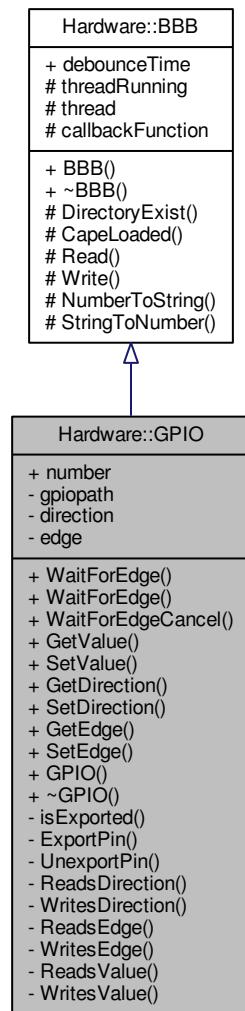
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.h](#)

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.cpp](#)

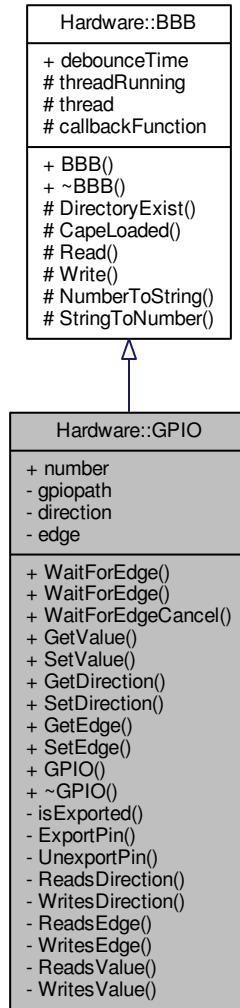
6.31 Hardware::GPIO Class Reference

```
#include <GPIO.h>
```

Inheritance diagram for Hardware::GPIO:



Collaboration diagram for Hardware::GPIO:



Public Types

- enum `Direction` { `Input`, `Output` }
- enum `Value` { `Low` = 0, `High` = 1 }
- enum `Edge` { `None`, `Rising`, `Falling`, `Both` }

Public Member Functions

- int `WaitForEdge ()`
- int `WaitForEdge (CallbackType callback)`
- void `WaitForEdgeCancel ()`
- `Value GetValue ()`
- void `SetValue (Value value)`
- `Direction GetDirection ()`
- void `SetDirection (Direction direction)`
- `Edge GetEdge ()`

- void `SetEdge (Edge edge)`
- `GPIO (int number)`
- `~GPIO ()`

Public Attributes

- int `number`

Private Member Functions

- bool `isExported (int number, Direction &dir, Edge &edge)`
- bool `ExportPin (int number)`
- bool `UnexportPin (int number)`
- `Direction ReadsDirection (const string &gpiopath)`
- void `WritesDirection (const string &gpiopath, Direction direction)`
- `Edge ReadsEdge (const string &gpiopath)`
- void `WritesEdge (const string &gpiopath, Edge edge)`
- `Value ReadsValue (const string &gpiopath)`
- void `WritesValue (const string &gpiopath, Value value)`

Private Attributes

- string `gpiopath`
- `Direction direction`
- `Edge edge`

Friends

- void * `threadedPollGPIO (void *value)`

Additional Inherited Members

6.31.1 Detailed Description

Definition at line 25 of file `GPIO.h`.

6.31.2 Member Enumeration Documentation

6.31.2.1 enum `Hardware::GPIO::Direction`

Enumerator

- Input*
Output

Definition at line 27 of file `GPIO.h`.

6.31.2.2 enum `Hardware::GPIO::Edge`

Enumerator

- None*
Rising
Falling
Both

Definition at line 29 of file `GPIO.h`.

6.31.2.3 enum `Hardware::GPIO::Value`

Enumerator

- Low*
High

Definition at line 28 of file `GPIO.h`.

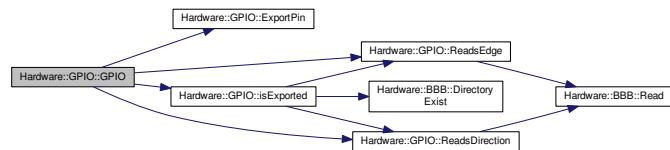
6.31.3 Constructor & Destructor Documentation

6.31.3.1 Hardware::GPIO (int *number*)

Definition at line 11 of file [GPIO.cpp](#).

References [direction](#), [edge](#), [ExportPin\(\)](#), [gpiopath](#), [GPIOs](#), [isExported\(\)](#), [number](#), [ReadsDirection\(\)](#), and [ReadsEdge\(\)](#).

Here is the call graph for this function:



6.31.3.2 Hardware::GPIO::~GPIO ()

Definition at line 24 of file [GPIO.cpp](#).

References [number](#), and [UnexportPin\(\)](#).

Here is the call graph for this function:



6.31.4 Member Function Documentation

6.31.4.1 bool Hardware::GPIO::ExportPin (int *number*) [private]

Definition at line 102 of file [GPIO.cpp](#).

Referenced by [GPIO\(\)](#).

Here is the caller graph for this function:



6.31.4.2 GPIO::Direction Hardware::GPIO::GetDirection ()

Definition at line 77 of file [GPIO.cpp](#).

References [direction](#).

6.31.4.3 GPIO::Edge Hardware::GPIO::GetEdge ()

Definition at line 83 of file [GPIO.cpp](#).

References [edge](#).

6.31.4.4 **GPIO::Value** `Hardware::GPIO::GetValue()`

Definition at line 74 of file [GPIO.cpp](#).

References [gpiopath](#), and [ReadsValue\(\)](#).

Here is the call graph for this function:

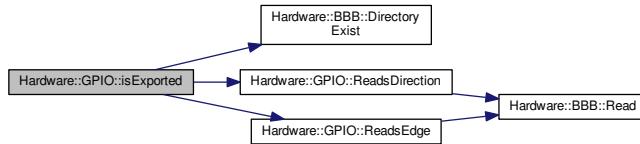
6.31.4.5 **bool Hardware::GPIO::isExported(int number, Direction & dir, Edge & edge)** [private]

Definition at line 89 of file [GPIO.cpp](#).

References [Hardware::BBB::DirectoryExist\(\)](#), [gpiopath](#), [ReadsDirection\(\)](#), and [ReadsEdge\(\)](#).

Referenced by [GPIO\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.31.4.6 **GPIO::Direction** `Hardware::GPIO::ReadsDirection(const string & gpiopath)` [private]

Definition at line 205 of file [GPIO.cpp](#).

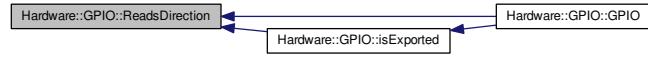
References [DIRECTION](#), [Input](#), [Output](#), and [Hardware::BBB::Read\(\)](#).

Referenced by [GPIO\(\)](#), and [isExported\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.4.7 GPIO::Edge Hardware::GPIO::ReadsEdge (const string & gpiopath) [private]

Definition at line 224 of file [GPIO.cpp](#).

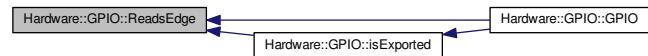
References [Both](#), [EDGE](#), [Falling](#), [None](#), [Hardware::BBB::Read\(\)](#), and [Rising](#).

Referenced by [GPIO\(\)](#), and [isExported\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.4.8 GPIO::Value Hardware::GPIO::ReadsValue (const string & gpiopath) [private]

Definition at line 256 of file [GPIO.cpp](#).

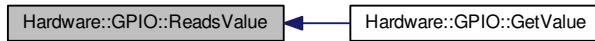
References [Hardware::BBB::Read\(\)](#), and [VALUE](#).

Referenced by [GetValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



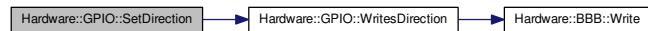
6.31.4.9 void Hardware::GPIO::SetDirection (*Direction direction*)

Definition at line 78 of file [GPIO.cpp](#).

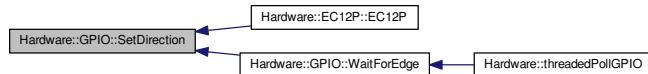
References [direction](#), [gpiopath](#), and [WritesDirection\(\)](#).

Referenced by [Hardware::EC12P::EC12P\(\)](#), and [WaitForEdge\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.31.4.10 void Hardware::GPIO::SetEdge (*Edge edge*)

Definition at line 84 of file [GPIO.cpp](#).

References [edge](#), [gpiopath](#), and [WritesEdge\(\)](#).

Referenced by [Hardware::EC12P::EC12P\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.31.4.11 void Hardware::GPIO::SetValue (*GPIO::Value value*)

Definition at line 75 of file [GPIO.cpp](#).

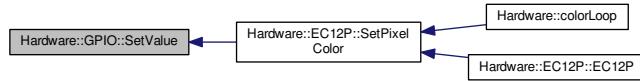
References [gpiopath](#), and [WritesValue\(\)](#).

Referenced by [Hardware::EC12P::SetPixelColor\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.4.12 bool Hardware::GPIO::UnexportPin (int *number*) [private]

Definition at line 201 of file [GPIO.cpp](#).

Referenced by [~GPIO\(\)](#).

Here is the caller graph for this function:



6.31.4.13 int Hardware::GPIO::WaitForEdge ()

Definition at line 37 of file [GPIO.cpp](#).

References [direction](#), [gpiopath](#), [Input](#), [Output](#), [SetDirection\(\)](#), and [VALUE](#).

Referenced by [Hardware::threadedPollGPIO\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.4.14 int Hardware::GPIO::WaitForEdge (CallbackType *callback*)

Definition at line 26 of file [GPIO.cpp](#).

References [Hardware::BBB::callbackFunction](#), [Hardware::BBB::thread](#), [threadedPollGPIO](#), and [Hardware::BBB::threadRunning](#).

6.31.4.15 `void Hardware::GPIO::WaitForEdgeCancel() [inline]`

Definition at line 35 of file [GPIO.h](#).

6.31.4.16 `void Hardware::GPIO::WritesDirection(const string & gpiopath, Direction direction) [private]`

Definition at line 213 of file [GPIO.cpp](#).

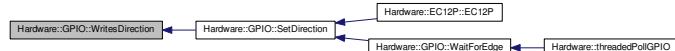
References [DIRECTION](#), [Input](#), [Output](#), and [Hardware::BBB::Write\(\)](#).

Referenced by [SetDirection\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.4.17 `void Hardware::GPIO::WritesEdge(const string & gpiopath, Edge edge) [private]`

Definition at line 237 of file [GPIO.cpp](#).

References [Both](#), [EDGE](#), [Falling](#), [None](#), [Rising](#), and [Hardware::BBB::Write\(\)](#).

Referenced by [SetEdge\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



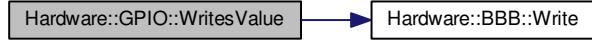
6.31.4.18 `void Hardware::GPIO::WritesValue(const string & gpiopath, Value value) [private]`

Definition at line 262 of file [GPIO.cpp](#).

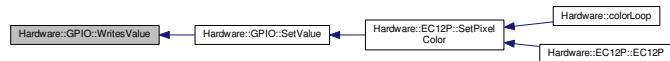
References [VALUE](#), and [Hardware::BBB::Write\(\)](#).

Referenced by [SetValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.5 Friends And Related Function Documentation

6.31.5.1 `void* threadedPolIGPIO (void * value) [friend]`

Definition at line 266 of file [GPIO.cpp](#).

Referenced by [WaitForEdge\(\)](#).

6.31.6 Member Data Documentation

6.31.6.1 `Direction Hardware::GPIO::direction [private]`

Definition at line 51 of file [GPIO.h](#).

Referenced by [GetDirection\(\)](#), [GPIO\(\)](#), [SetDirection\(\)](#), and [WaitForEdge\(\)](#).

6.31.6.2 `Edge Hardware::GPIO::edge [private]`

Definition at line 52 of file [GPIO.h](#).

Referenced by [GetEdge\(\)](#), [GPIO\(\)](#), and [SetEdge\(\)](#).

6.31.6.3 `string Hardware::GPIO::gpiopath [private]`

Definition at line 50 of file [GPIO.h](#).

Referenced by [GetValue\(\)](#), [GPIO\(\)](#), [isExported\(\)](#), [SetDirection\(\)](#), [SetEdge\(\)](#), [SetValue\(\)](#), and [WaitForEdge\(\)](#).

6.31.6.4 `int Hardware::GPIO::number`

Definition at line 31 of file [GPIO.h](#).

Referenced by [GPIO\(\)](#), and [~GPIO\(\)](#).

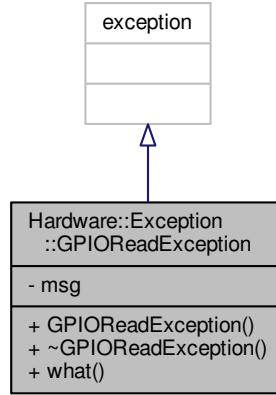
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIO.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIO.cpp](#)

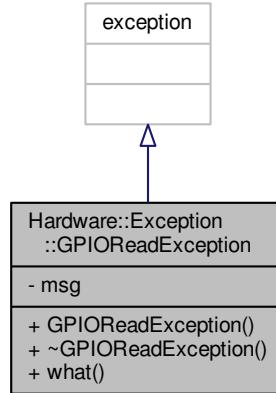
6.32 Hardware::Exception::GPIOReadException Class Reference

```
#include <GPIOReadException.h>
```

Inheritance diagram for Hardware::Exception::GPIOReadException:



Collaboration diagram for Hardware::Exception::GPIOReadException:



Public Member Functions

- `GPIOReadException` (string m="Can't read **GPIO** data!")
- `~GPIOReadException ()` _GLIBCXX_USE_NOEXCEPT
- `const char * what ()` const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string `msg`

6.32.1 Detailed Description

Definition at line 17 of file [GPIOReadException.h](#).

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `Hardware::Exception::GPIOReadException::GPIOReadException (string m = "Can't read GPIO data!") [inline]`

Definition at line 19 of file [GPIOReadException.h](#).

6.32.2.2 `Hardware::Exception::GPIOReadException::~GPIOReadException () [inline]`

Definition at line 20 of file [GPIOReadException.h](#).

6.32.3 Member Function Documentation

6.32.3.1 `const char* Hardware::Exception::GPIOReadException::what () const [inline]`

Definition at line 21 of file [GPIOReadException.h](#).

6.32.4 Member Data Documentation

6.32.4.1 `string Hardware::Exception::GPIOReadException::msg [private]`

Definition at line 21 of file [GPIOReadException.h](#).

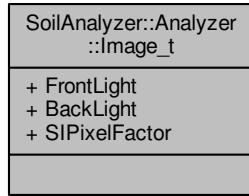
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/[GPIOReadException.h](#)

6.33 SoilAnalyzer::Analyzer::Image_t Struct Reference

#include <analyzer.h>

Collaboration diagram for SoilAnalyzer::Analyzer::Image_t:



Public Attributes

- `cv::Mat FrontLight`
- `cv::Mat BackLight`
- `float SIPixelFactor = 0.0111915`

6.33.1 Detailed Description

Definition at line 39 of file [analyzer.h](#).

6.33.2 Member Data Documentation

6.33.2.1 `cv::Mat SoilAnalyzer::Analyzer::Image_t::BackLight`

Definition at line 41 of file [analyzer.h](#).

Referenced by [VSAMainWindow::TakeSnapShots\(\)](#).

6.33.2.2 cv::Mat SoilAnalyzer::Analyzer::Image_t::FrontLight

Definition at line 40 of file [analyzer.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [VSAMainWindow::TakeSnapShots\(\)](#).

6.33.2.3 float SoilAnalyzer::Analyzer::Image_t::SIPixelFactor = 0.0111915

Definition at line 42 of file [analyzer.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [VSAMainWindow::TakeSnapShots\(\)](#).

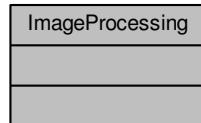
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.h](#)

6.34 ImageProcessing Class Reference

Core class of all the image classes Core class of all the image classes with a few commonly shared functions and variables.

Collaboration diagram for ImageProcessing:



6.34.1 Detailed Description

Core class of all the image classes Core class of all the image classes with a few commonly shared functions and variables.

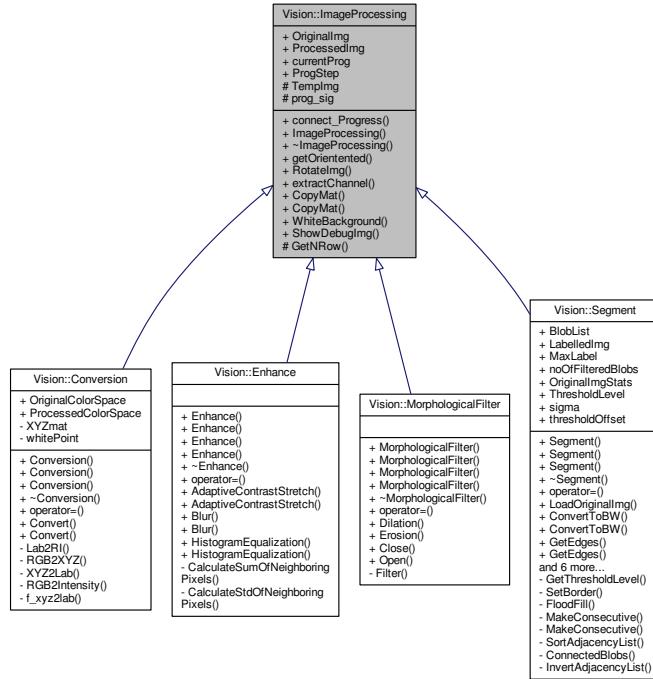
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.cpp](#)

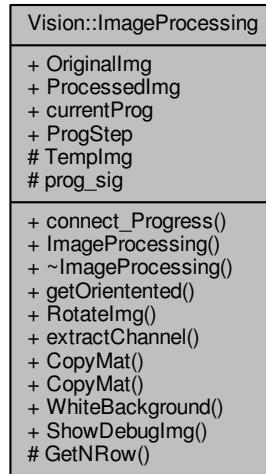
6.35 Vision::ImageProcessing Class Reference

```
#include <ImageProcessing.h>
```

Inheritance diagram for Vision::ImageProcessing:



Collaboration diagram for Vision::ImageProcessing:



Public Types

- `typedef boost::signals2::signal< void(float, std::string)> Progress_t`

Public Member Functions

- boost::signals2::connection [connect_Progress](#) (const Progress_t::slot_type &subscriber)
- [ImageProcessing](#) ()
- [~ImageProcessing](#) ()

Static Public Member Functions

- static void [getOriented](#) (Mat &BW, cv::Point_< double > ¢roid, double &theta, double &eccentricity)
- static void [RotateImg](#) (Mat &src, Mat &dst, double &theta, cv::Point_< double > &Centroid, Rect &ROI)
- static std::vector< Mat > [extractChannel](#) (const Mat &src)
- template<typename T1 , typename T2 >
static Mat [CopyMat](#) (const Mat &src, T1 *LUT, int cvType)
- template<typename T1 >
static Mat [CopyMat](#) (const Mat &src, const Mat &mask, int cvType)
- static cv::Mat [WhiteBackground](#) (const cv::Mat &src)
- template<typename T1 >
static void [ShowDebugImg](#) (cv::Mat img, T1 maxVal, std::string windowName, bool scale=true)

Public Attributes

- Mat [OriginalImg](#)
- Mat [ProcessedImg](#)
- double [currentProg](#) = 0.
- double [ProgStep](#) = 0.

Protected Member Functions

- uchar * [GetNRow](#) (int nData, int hKsize, int nCols, uint32_t totalRows)

Protected Attributes

- Mat [TemplImg](#)
- Progress_t [prog_sig](#)

6.35.1 Detailed Description

Definition at line 48 of file [ImageProcessing.h](#).

6.35.2 Member Typedef Documentation

6.35.2.1 [typedef boost::signals2::signal<void\(float, std::string\)> Vision::ImageProcessing::Progress_t](#)

Definition at line 50 of file [ImageProcessing.h](#).

6.35.3 Constructor & Destructor Documentation

6.35.3.1 [ImageProcessing::ImageProcessing \(\)](#)

Constructor of the core class

Definition at line 17 of file [ImageProcessing.cpp](#).

6.35.3.2 [ImageProcessing::~ImageProcessing \(\)](#)

De-constructor of the core class

Definition at line 20 of file [ImageProcessing.cpp](#).

6.35.4 Member Function Documentation

6.35.4.1 boost::signals2::connection ImageProcessing::connect_Progress (const Progress_t::slot_type & subscriber)

Definition at line 130 of file [ImageProcessing.cpp](#).

References [prog_sig](#).

6.35.4.2 template<typename T1 , typename T2 > static Mat Vision::ImageProcessing::CopyMat (const Mat & src, T1 * LUT, int cvType) [inline], [static]

Copy a matrix to a new matrix with a LUT mask

Parameters

<i>src</i>	the source image
<i>*LUT</i>	type T with a LUT to filter out unwanted pixel values
<i>cvType</i>	an in where you can pas CV_UC8C1 etc.

Returns

The new matrix

Definition at line 82 of file [ImageProcessing.h](#).

6.35.4.3 template<typename T1 > static Mat Vision::ImageProcessing::CopyMat (const Mat & src, const Mat & mask, int cvType) [inline], [static]

Copy a matrix to a new matrix with a mask

Parameters

<i>src</i>	the source image
<i>*LUT</i>	type T with a LUT to filter out unwanted pixel values
<i>cvType</i>	an in where you can pas CV_UC8C1 etc.

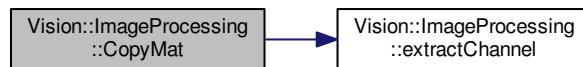
Returns

The new matrix

Definition at line 121 of file [ImageProcessing.h](#).

References [extractChannel\(\)](#).

Here is the call graph for this function:

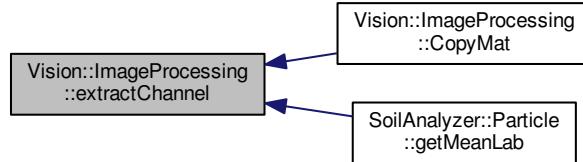


6.35.4.4 std::vector< Mat > ImageProcessing::extractChannel (const Mat & src) [static]

Definition at line 42 of file [ImageProcessing.cpp](#).

Referenced by [CopyMat\(\)](#), and [SoilAnalyzer::Particle::getMeanLab\(\)](#).

Here is the caller graph for this function:



6.35.4.5 uchar * ImageProcessing::GetNRow (int *nData*, int *hKsize*, int *nCols*, uint32_t *totalRows*) [protected]

Create a LUT indicating which iteration variable i is the end of an row.

Parameters

<i>nData</i>	an int indicating total pixels
<i>hKsize</i>	int half the size of the kernel, if any. which acts as an offset from the border pixels
<i>nCols</i>	int number of columns in a row

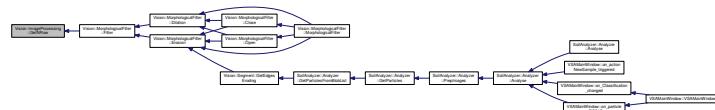
Returns

array of uchar where a zero is a middle column and a 1 indicates an end of an row minus the offset from half the kernel size

Definition at line 30 of file [ImageProcessing.cpp](#).

Referenced by [Vision::MorphologicalFilter::Filter\(\)](#).

Here is the caller graph for this function:



6.35.4.6 void ImageProcessing::getOriented (Mat & *BW*, cv::Point_< double > & *centroid*, double & *theta*, double & *eccentricity*) [static]

Definition at line 48 of file [ImageProcessing.cpp](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#).

Here is the caller graph for this function:

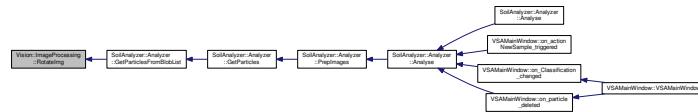


6.35.4.7 void ImageProcessing::RotateImage (Mat & src, Mat & dst, double & theta, cv::Point< double > & Centroid, Rect & ROI) [static]

Definition at line 70 of file [ImageProcessing.cpp](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#).

Here is the caller graph for this function:



6.35.4.8 `template<typename T1> static void Vision::ImageProcessing::ShowDebugImg (cv::Mat img, T1 maxVal, std::string windowName, bool scale = true) [inline], [static]`

Definition at line 156 of file [ImageProcessing.h](#).

6.35.4.9 `static cv::Mat Vision::ImageProcessing::WhiteBackground (const cv::Mat & src) [inline], [static]`

Definition at line 149 of file [ImageProcessing.h](#).

6.35.5 Member Data Documentation

6.35.5.1 `double Vision::ImageProcessing::currentProg = 0.`

Definition at line 70 of file [ImageProcessing.h](#).

Referenced by [Vision::Conversion::Convert\(\)](#).

6.35.5.2 `Mat Vision::ImageProcessing::OriginalImg`

Definition at line 63 of file [ImageProcessing.h](#).

Referenced by [Vision::Segment::ConnectedBlobs\(\)](#), [Vision::Conversion::Conversion\(\)](#), [Vision::Conversion::Convert\(\)](#), [Vision::Segment::ConvertToBW\(\)](#), [Vision::Enhance::Enhance\(\)](#), [Vision::Segment::FillHoles\(\)](#), [Vision::MorphologicalFilter::Filter\(\)](#), [Vision::Segment::GetBlobList\(\)](#), [Vision::Segment::GetEdges\(\)](#), [Vision::Segment::GetEdgesEroding\(\)](#), [Vision::Segment::GetThresholdLevel\(\)](#), [Vision::Enhance::HistogramEqualization\(\)](#), [Vision::Segment::LabelBlobs\(\)](#), [Vision::Segment::LoadOriginalImg\(\)](#), [Vision::MorphologicalFilter::MorphologicalFilter\(\)](#), [Vision::MorphologicalFilter::operator=\(\)](#), [Vision::Conversion::operator=\(\)](#), [Vision::Enhance::operator=\(\)](#), [Vision::Segment::operator=\(\)](#), [Vision::Segment::RemoveBorderBlobs\(\)](#), [Vision::Conversion::RGB2XYZ\(\)](#), [Vision::Segment::Segment\(\)](#), [Vision::Segment::SetBorder\(\)](#), and [Vision::Segment::Threshold\(\)](#).

6.35.5.3 `Mat Vision::ImageProcessing::ProcessedImg`

Definition at line 64 of file [ImageProcessing.h](#).

Referenced by [SoilAnalyzer::Analyzer::CalibrateSI\(\)](#), [Vision::Conversion::Conversion\(\)](#), [Vision::Conversion::Convert\(\)](#), [Vision::Segment::ConvertToBW\(\)](#), [Vision::Enhance::Enhance\(\)](#), [Vision::Segment::FillHoles\(\)](#), [Vision::MorphologicalFilter::Filter\(\)](#), [SoilAnalyzer::Analyzer::GetBW\(\)](#), [Vision::Segment::GetEdges\(\)](#), [Vision::Segment::GetEdgesEroding\(\)](#), [SoilAnalyzer::Particle::getLabImg\(\)](#), [SoilAnalyzer::Particle::GetMeanRI\(\)](#), [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), [Vision::Enhance::HistogramEqualization\(\)](#), [Vision::Segment::LabelBlobs\(\)](#), [Vision::Segment::LoadOriginalImg\(\)](#), [Vision::MorphologicalFilter::MorphologicalFilter\(\)](#), [Vision::MorphologicalFilter::operator=\(\)](#), [Vision::Conversion::operator=\(\)](#), [Vision::Enhance::operator=\(\)](#), [Vision::Segment::operator=\(\)](#), [Vision::Segment::RemoveBorderBlobs\(\)](#), [Vision::Segment::Segment\(\)](#), and [Vision::Segment::Threshold\(\)](#).

6.35.5.4 `Progress_t Vision::ImageProcessing::prog_sig [protected]`

Definition at line 58 of file [ImageProcessing.h](#).

Referenced by [connect_Progress\(\)](#), and [Vision::Conversion::Convert\(\)](#).

6.35.5.5 `double Vision::ImageProcessing::ProgStep = 0.`

Definition at line 71 of file [ImageProcessing.h](#).

Referenced by [Vision::Conversion::Convert\(\)](#).

6.35.5.6 `Mat Vision::ImageProcessing::TemplImg [protected]`

Definition at line 56 of file [ImageProcessing.h](#).

Referenced by [Vision::Conversion::Conversion\(\)](#), [Vision::Enhance::Enhance\(\)](#), [Vision::Segment::FillHoles\(\)](#), [Vision::Segment::GetEdgesEroding\(\)](#), [Vision::Segment::LabelBlobs\(\)](#), [Vision::MorphologicalFilter::MorphologicalFilter\(\)](#), [Vision::MorphologicalFilter::operator=\(\)](#), [Vision::Conversion::operator=\(\)](#), [Vision::Enhance::operator=\(\)](#), [Vision::Segment::operator=\(\)](#), [Vision::Segment::RemoveBorderBlobs\(\)](#), and [Vision::Segment::Segment\(\)](#).

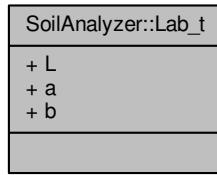
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.cpp](#)

6.36 SoilAnalyzer::Lab_t Struct Reference

#include <soilanalyzertypes.h>

Collaboration diagram for SoilAnalyzer::Lab_t:



Public Attributes

- float [L](#)
- float [a](#)
- float [b](#)

6.36.1 Detailed Description

Definition at line 10 of file [soilanalyzertypes.h](#).

6.36.2 Member Data Documentation

6.36.2.1 float SoilAnalyzer::Lab_t::a

Definition at line 12 of file [soilanalyzertypes.h](#).

Referenced by [SoilAnalyzer::Particle::getMeanLab\(\)](#), [boost::serialization::serialize\(\)](#), and [SoilAnalyzer::Particle::serialize\(\)](#).

6.36.2.2 float SoilAnalyzer::Lab_t::b

Definition at line 13 of file [soilanalyzertypes.h](#).

Referenced by [SoilAnalyzer::Particle::getMeanLab\(\)](#), [boost::serialization::serialize\(\)](#), and [SoilAnalyzer::Particle::serialize\(\)](#).

6.36.2.3 float SoilAnalyzer::Lab_t::L

Definition at line 11 of file [soilanalyzertypes.h](#).

Referenced by [SoilAnalyzer::Particle::getMeanLab\(\)](#), [boost::serialization::serialize\(\)](#), and [SoilAnalyzer::Particle::serialize\(\)](#).

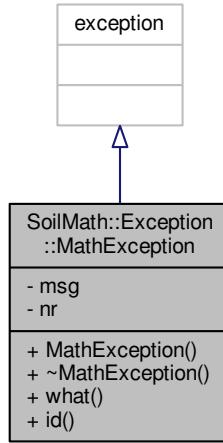
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzertypes.h](#)

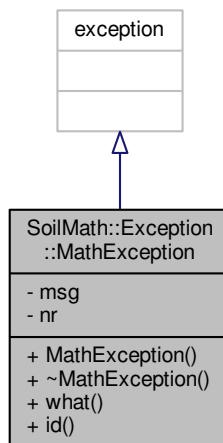
6.37 SoilMath::Exception::MathException Class Reference

#include <MathException.h>

Inheritance diagram for SoilMath::Exception::MathException:



Collaboration diagram for SoilMath::Exception::MathException:



Public Member Functions

- `MathException (std::string m=EXCEPTION_MATH, int n=EXCEPTION_MATH_NR)`
- `~MathException () _GLIBCXX_USE_NOEXCEPT`
- `const char * what () const _GLIBCXX_USE_NOEXCEPT`
- `const int * id () const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- `std::string msg`
- `int nr`

6.37.1 Detailed Description

Definition at line 28 of file [MathException.h](#).

6.37.2 Constructor & Destructor Documentation

6.37.2.1 `SoilMath::Exception::MathException::MathException (std::string m = EXCEPTION_MATH, int n = EXCEPTION_MATH_NR) [inline]`

Definition at line 30 of file [MathException.h](#).

6.37.2.2 `SoilMath::Exception::MathException::~MathException () [inline]`

Definition at line 32 of file [MathException.h](#).

6.37.3 Member Function Documentation

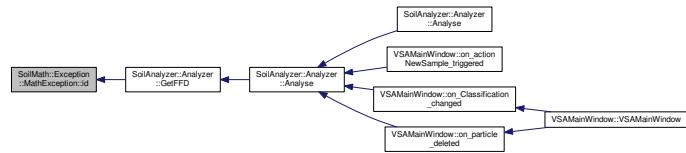
6.37.3.1 `const int* SoilMath::Exception::MathException::id () const [inline]`

Definition at line 34 of file [MathException.h](#).

References [nr](#).

Referenced by [SoilAnalyzer::Analyzer::GetFFD\(\)](#).

Here is the caller graph for this function:

6.37.3.2 `const char* SoilMath::Exception::MathException::what () const [inline]`

Definition at line 33 of file [MathException.h](#).

References [msg](#).

6.37.4 Member Data Documentation

6.37.4.1 `std::string SoilMath::Exception::MathException::msg [private]`

Definition at line 37 of file [MathException.h](#).

Referenced by [what\(\)](#).

6.37.4.2 `int SoilMath::Exception::MathException::nr [private]`

Definition at line 38 of file [MathException.h](#).

Referenced by [id\(\)](#).

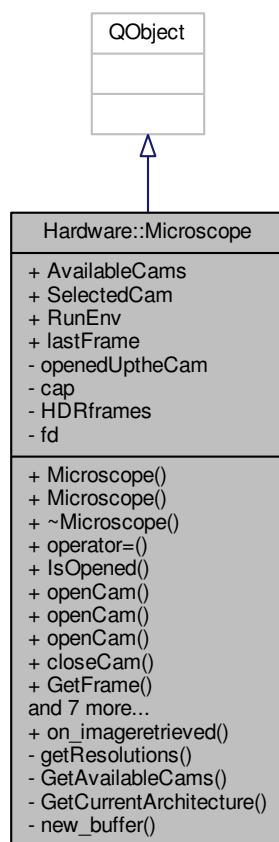
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/MathException.h

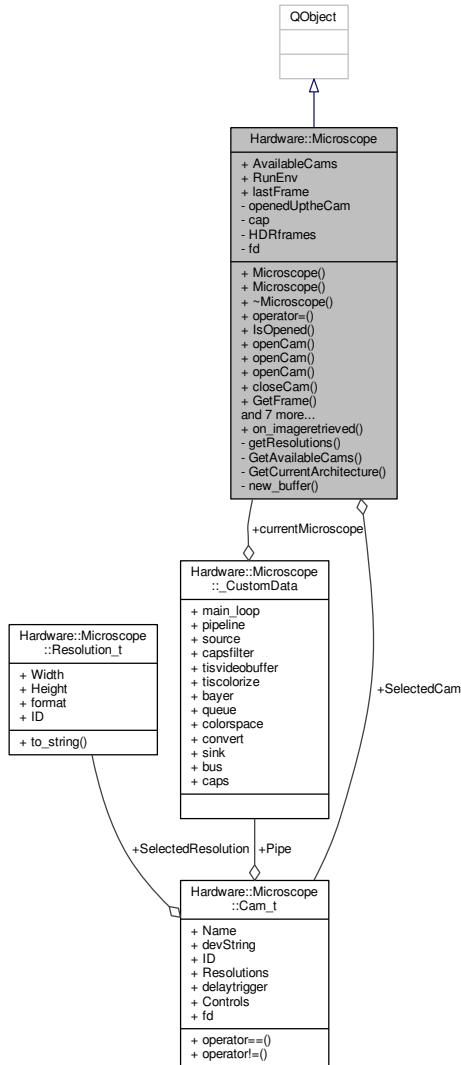
6.38 Hardware::Microscope Class Reference

```
#include <Microscope.h>
```

Inheritance diagram for Hardware::Microscope:



Collaboration diagram for Hardware::Microscope:



Classes

- struct `_CustomData`
- struct `Cam_t`
- struct `Control_t`
- struct `Resolution_t`

Public Types

- enum `Arch` { `ARM`, `X64` }
- enum `PixelFormat` { `YUYV`, `MJPG`, `GREY` }
- typedef std::vector<`Control_t`> `Controls_t`
- typedef struct `Hardware::Microscope::_CustomData` `CustomData`

Public Slots

- void `on_imageretrieved ()`

Signals

- void `imageretrieved ()`

Public Member Functions

- `Microscope ()`
- `Microscope (const Microscope &rhs)`
- `~Microscope ()`
- `Microscope operator= (Microscope const &rhs)`
- `bool IsOpened ()`
- `bool openCam (Cam_t *cam)`
- `bool openCam (int &cam)`
- `bool openCam (std::string &cam)`
- `bool closeCam (Cam_t *cam)`
- `void GetFrame (cv::Mat &dst)`
- `void GetGstreamFrame (cv::Mat &dst)`
- `void GetHDRFrame (cv::Mat &dst, uint32_t noframes=3)`
- `Control_t * GetControl (const std::string name)`
- `void SetControl (Control_t *control)`
- `Cam_t * FindCam (std::string cam)`
- `Cam_t * FindCam (int cam)`
- `void SendImageRetrieved ()`

Public Attributes

- `std::vector< Cam_t > AvailableCams`
- `Cam_t * SelectedCam = nullptr`
- `Arch RunEnv`
- `cv::Mat lastFrame`

Private Member Functions

- `void getResolutions (Cam_t ¤tCam, int FormatType)`
- `std::vector< Cam_t > GetAvailableCams ()`
- `Arch GetCurrentArchitecture ()`

Static Private Member Functions

- `static void new_buffer (GstElement *sink, CustomData *data)`

Private Attributes

- `bool openedUptheCam = false`
- `cv::VideoCapture * cap = nullptr`
- `std::vector< cv::Mat > HDRframes`
- `int fd`

6.38.1 Detailed Description

Definition at line 49 of file `Microscope.h`.

6.38.2 Member Typedef Documentation

6.38.2.1 `typedef std::vector<Control_t> Hardware::Microscope::Controls_t`

Definition at line 103 of file `Microscope.h`.

6.38.2.2 `typedef struct Hardware::Microscope::_CustomData Hardware::Microscope::CustomData`

6.38.3 Member Enumeration Documentation

6.38.3.1 `enum Hardware::Microscope::Arch`

Enumerator

ARM

X64

Definition at line 53 of file [Microscope.h](#).

6.38.3.2 `enum Hardware::Microscope::PixelFormat`

Enumerator

YUYV

MJPG

GREY

Definition at line 55 of file [Microscope.h](#).

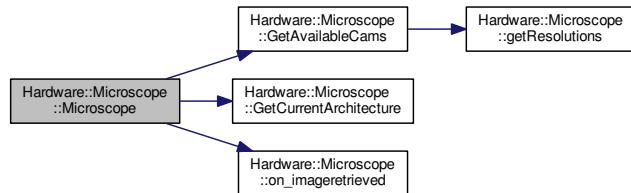
6.38.4 Constructor & Destructor Documentation

6.38.4.1 `Microscope::Microscope()`

Definition at line 12 of file [Microscope.cpp](#).

References [AvailableCams](#), [GetAvailableCams\(\)](#), [GetCurrentArchitecture\(\)](#), [imageretrieved\(\)](#), [on_imageretrieved\(\)](#), and [RunEnv](#).

Here is the call graph for this function:

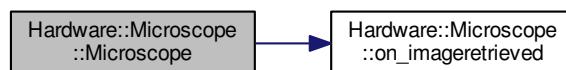


6.38.4.2 `Microscope::Microscope(const Microscope & rhs)`

Definition at line 21 of file [Microscope.cpp](#).

References [AvailableCams](#), [cap](#), [fd](#), [HDRframes](#), [imageretrieved\(\)](#), [on_imageretrieved\(\)](#), [RunEnv](#), and [SelectedCam](#).

Here is the call graph for this function:



6.38.4.3 Microscope::~Microscope()

Definition at line 32 of file [Microscope.cpp](#).

References [cap](#).

6.38.5 Member Function Documentation

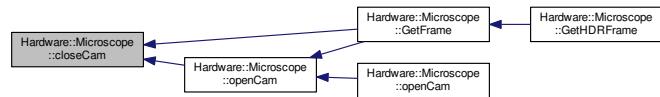
6.38.5.1 bool Microscope::closeCam(Cam_t * cam)

Definition at line 311 of file [Microscope.cpp](#).

References [openedUpTheCam](#), [Hardware::Microscope::Cam_t::Pipe](#), and [Hardware::Microscope::_CustomData::pipeline](#).

Referenced by [GetFrame\(\)](#), and [openCam\(\)](#).

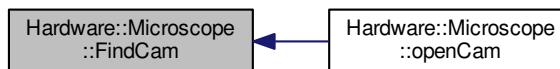
Here is the caller graph for this function:



6.38.5.2 Cam_t* Hardware::Microscope::FindCam(std::string cam)

Referenced by [openCam\(\)](#).

Here is the caller graph for this function:



6.38.5.3 Microscope::Cam_t * Microscope::FindCam(int cam)

Definition at line 293 of file [Microscope.cpp](#).

References [AvailableCams](#).

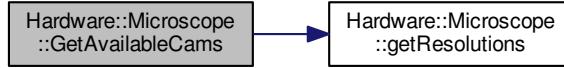
6.38.5.4 std::vector< Microscope::Cam_t > Microscope::GetAvailableCams() [private]

Definition at line 47 of file [Microscope.cpp](#).

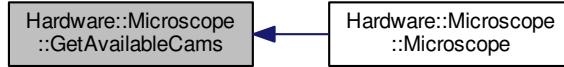
References [Hardware::Microscope::Cam_t::Controls](#), [Hardware::Microscope::Control_t::current_value](#), [Hardware::Microscope::Control_t::default_value](#), [Hardware::Microscope::Cam_t::devString](#), [EXCEPTION_NOCAMS](#), [EXCEPTION_NOCAMS_NR](#), [EXCEPTION_QUERY](#), [EXCEPTION_QUERY_NR](#), [Hardware::Microscope::Cam_t::fd](#), [getResolutions\(\)](#), [Hardware::Microscope::Control_t::ID](#), [Hardware::Microscope::Cam_t::ID](#), [Hardware::Microscope::Control_t::maximum](#), [Hardware::Microscope::Control_t::minimum](#), [Hardware::Microscope::Control_t::name](#), [Hardware::Microscope::Cam_t::Name](#), and [Hardware::Microscope::Control_t::step](#).

Referenced by [Microscope\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.38.5.5 Microscope::Control_t * Microscope::GetControl (const std::string name)

Definition at line 364 of file [Microscope.cpp](#).

References [Hardware::Microscope::Cam_t::Controls](#), and [SelectedCam](#).

Referenced by [GetHDRFrame\(\)](#).

Here is the caller graph for this function:

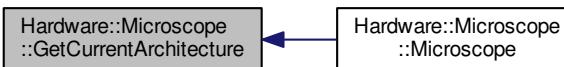


6.38.5.6 Microscope::Arch Microscope::GetCurrentArchitecture () [private]

Definition at line 34 of file [Microscope.cpp](#).

Referenced by [Microscope\(\)](#).

Here is the caller graph for this function:



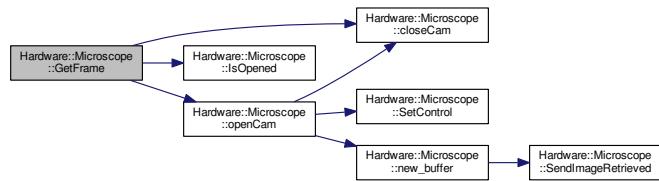
6.38.5.7 void Microscope::GetFrame (cv::Mat & dst)

Definition at line 319 of file [Microscope.cpp](#).

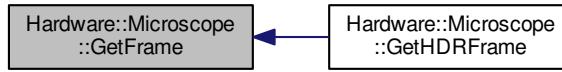
References [closeCam\(\)](#), [imageretrieved\(\)](#), [IsOpened\(\)](#), [lastFrame](#), [openCam\(\)](#), [Hardware::Microscope::Cam_t::Pipe](#), [Hardware::Microscope::CustomData::pipeline](#), and [SelectedCam](#).

Referenced by [GetHDRFrame\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



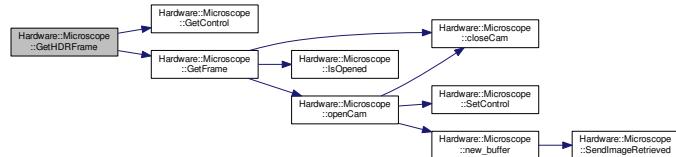
6.38.5.8 void Hardware::Microscope::GetGstreamFrame (cv::Mat & dst)

6.38.5.9 void Microscope::GetHDRFrame (cv::Mat & dst, uint32_t noframes = 3)

Definition at line 333 of file [Microscope.cpp](#).

References [Hardware::Microscope::Control_t::current_value](#), [GetControl\(\)](#), [GetFrame\(\)](#), [HDRframes](#), [Hardware::Microscope::Control_t::maximum](#), and [Hardware::Microscope::Control_t::minimum](#).

Here is the call graph for this function:



6.38.5.10 void Microscope::getResolutions (Cam_t & currentCam, int FormatType) [private]

Definition at line 123 of file [Microscope.cpp](#).

References [Hardware::Microscope::Cam_t::fd](#), [Hardware::Microscope::Resolution_t::format](#), [Hardware::Microscope::Resolution_t::Height](#), [Hardware::Microscope::Resolution_t::ID](#), [Hardware::Microscope::Cam_t::Resolutions](#), and [Hardware::Microscope::Resolution_t::Width](#).

Referenced by [GetAvailableCams\(\)](#).

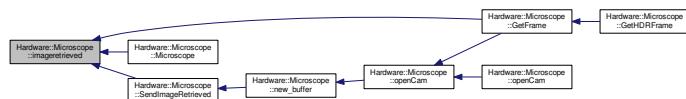
Here is the caller graph for this function:



6.38.5.11 void Hardware::Microscope::imageretrieved() [signal]

Referenced by [GetFrame\(\)](#), [Microscope\(\)](#), and [SendImageRetrieved\(\)](#).

Here is the caller graph for this function:



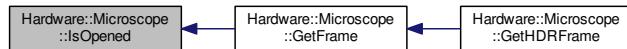
6.38.5.12 bool Microscope::IsOpened()

Definition at line 165 of file [Microscope.cpp](#).

References [openedUptheCam](#).

Referenced by [GetFrame\(\)](#).

Here is the caller graph for this function:



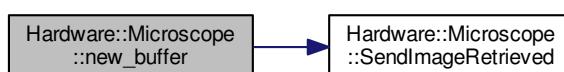
6.38.5.13 void Microscope::new_buffer(GstElement * sink, CustomData * data) [static], [private]

Definition at line 413 of file [Microscope.cpp](#).

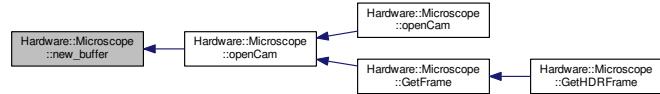
References [Hardware::Microscope::_CustomData::currentMicroscope](#), [Hardware::Microscope::Resolution_t::Height](#), [lastFrame](#), [SelectedCam](#), [Hardware::Microscope::Cam_t::SelectedResolution](#), [SendImageRetrieved\(\)](#), and [Hardware::Microscope::Resolution_t::Width](#).

Referenced by [openCam\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

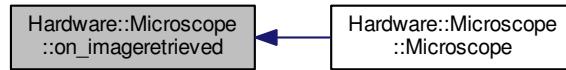


6.38.5.14 void Microscope::on_imageretrieved () [slot]

Definition at line 331 of file [Microscope.cpp](#).

Referenced by [Microscope\(\)](#).

Here is the caller graph for this function:



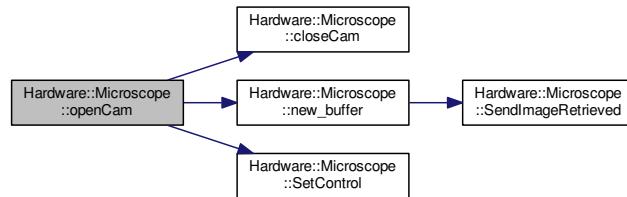
6.38.5.15 bool Microscope::openCam (Cam_t * cam)

Definition at line 167 of file [Microscope.cpp](#).

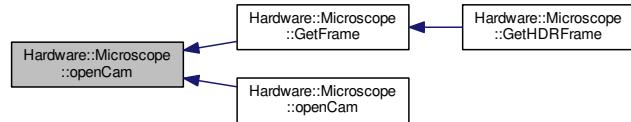
References [AvailableCams](#), [Hardware::Microscope::_CustomData::bayer](#), [Hardware::Microscope::_CustomData::bus](#), [Hardware::Microscope::_CustomData::caps](#), [Hardware::Microscope::_CustomData::capsfilter](#), [closeCam\(\)](#), [Hardware::Microscope::_CustomData::colorspace](#), [Hardware::Microscope::Cam_t::Controls](#), [Hardware::Microscope::_CustomData::convert](#), [Hardware::Microscope::_CustomData::current](#), [Microscope](#), [Hardware::Microscope::Cam_t::devString](#), [EXCEPTION_GSTREAM_ELEM_EXCEPTION](#), [EXCEPTION_GSTREAM_ELEM](#), [EXCEPTION_NR](#), [EXCEPTION_GSTREAM_INIT_EXCEPTION](#), [EXCEPTION_GSTREAM_INIT_EXCEPTION_NR](#), [Hardware::Microscope::_Resolution_t::format](#), [Hardware::Microscope::Resolution_t::Height](#), [Hardware::Microscope::Cam_t::Name](#), [new_buffer\(\)](#), [openedUpTheCam](#), [Hardware::Microscope::Cam_t::Pipe](#), [Hardware::Microscope::_CustomData::pipeline](#), [Hardware::Microscope::_CustomData::queue](#), [Selected](#), [Cam](#), [Hardware::Microscope::Cam_t::SelectedResolution](#), [SetControl\(\)](#), [Hardware::Microscope::_CustomData::sink](#), [Hardware::Microscope::_CustomData::source](#), [Hardware::Microscope::_CustomData::tiscolorize](#), [Hardware::Microscope::_CustomData::tisvideobuffer](#), and [Hardware::Microscope::Resolution_t::Width](#).

Referenced by [GetFrame\(\)](#), and [openCam\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

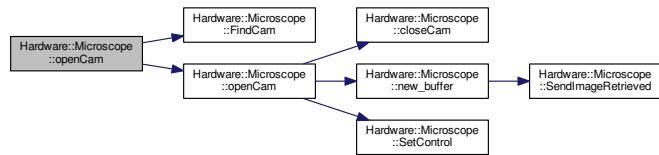


6.38.5.16 bool Microscope::openCam (int & cam)

Definition at line 291 of file [Microscope.cpp](#).

References [FindCam\(\)](#), and [openCam\(\)](#).

Here is the call graph for this function:

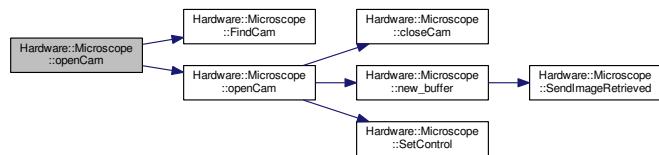


6.38.5.17 bool Microscope::openCam (std::string & cam)

Definition at line 289 of file [Microscope.cpp](#).

References [FindCam\(\)](#), and [openCam\(\)](#).

Here is the call graph for this function:



6.38.5.18 Microscope Hardware::Microscope::operator= (Microscope const & rhs)

6.38.5.19 void Microscope::SendImageRetrieved ()

Definition at line 411 of file [Microscope.cpp](#).

References [imageretrieved\(\)](#).

Referenced by [new_buffer\(\)](#).

Here is the caller graph for this function:



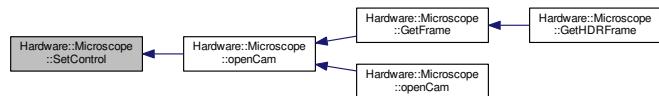
6.38.5.20 void Microscope::SetControl (Control_t * control)

Definition at line 374 of file [Microscope.cpp](#).

References [Hardware::Microscope::Control_t::current_value](#), [Hardware::Microscope::Cam_t::devString](#), [EXCEPTION_CTRL_NOT_FOUND](#), [EXCEPTION_CTRL_NOT_FOUND_NR](#), [EXCEPTION_NOAMS](#), [EXCEPTION_NOAMS_NR](#), [EXCEPTION_QUERY](#), [EXCEPTION_QUERY_NR](#), [Hardware::Microscope::Cam_t::fd](#), [Hardware::Microscope::Control_t::ID](#), and [SelectedCam](#).

Referenced by [openCam\(\)](#).

Here is the caller graph for this function:



6.38.6 Member Data Documentation

6.38.6.1 std::vector<Cam_t> Hardware::Microscope::AvailableCams

Definition at line 148 of file [Microscope.h](#).

Referenced by [FindCam\(\)](#), [Microscope\(\)](#), and [openCam\(\)](#).

6.38.6.2 cv::VideoCapture* Hardware::Microscope::cap = nullptr [private]

Definition at line 189 of file [Microscope.h](#).

Referenced by [Microscope\(\)](#), and [~Microscope\(\)](#).

6.38.6.3 int Hardware::Microscope::fd [private]

Definition at line 195 of file [Microscope.h](#).

Referenced by [Microscope\(\)](#).

6.38.6.4 std::vector<cv::Mat> Hardware::Microscope::HDRframes [private]

Definition at line 191 of file [Microscope.h](#).

Referenced by [GetHDRFrame\(\)](#), and [Microscope\(\)](#).

6.38.6.5 cv::Mat Hardware::Microscope::lastFrame

Definition at line 175 of file [Microscope.h](#).

Referenced by [GetFrame\(\)](#), and [new_buffer\(\)](#).

6.38.6.6 bool Hardware::Microscope::openedUptheCam = false [private]

Definition at line 188 of file [Microscope.h](#).

Referenced by [closeCam\(\)](#), [IsOpened\(\)](#), and [openCam\(\)](#).

6.38.6.7 Arch Hardware::Microscope::RunEnv

Definition at line 150 of file [Microscope.h](#).

Referenced by [Microscope\(\)](#).

6.38.6.8 Cam_t* Hardware::Microscope::SelectedCam = nullptr

Definition at line 149 of file [Microscope.h](#).

Referenced by [GetControl\(\)](#), [GetFrame\(\)](#), [Microscope\(\)](#), [new_buffer\(\)](#), [openCam\(\)](#), and [SetControl\(\)](#).

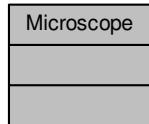
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.cpp](#)

6.39 Microscope Class Reference

```
#include <Microscope.h>
```

Collaboration diagram for Microscope:



6.39.1 Detailed Description

Interaction with the microscope

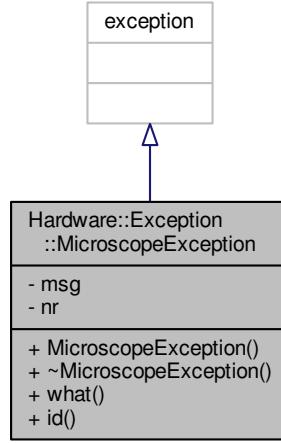
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h](#)

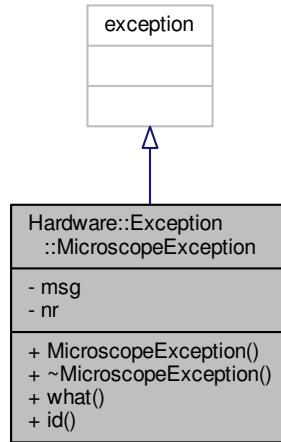
6.40 Hardware::Exception::MicroscopeException Class Reference

```
#include <MicroscopeNotFoundException.h>
```

Inheritance diagram for Hardware::Exception::MicroscopeException:



Collaboration diagram for Hardware::Exception::MicroscopeException:



Public Member Functions

- `MicroscopeException (string m=EXCEPTION_OPENCAM, int n=EXCEPTION_OPENCAM_NR)`
- `~MicroscopeException () _GLIBCXX_USE_NOEXCEPT`
- `const char * what () const _GLIBCXX_USE_NOEXCEPT`
- `const int * id () const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- string `msg`
- int `nr`

6.40.1 Detailed Description

Definition at line 35 of file [MicroscopeNotFoundException.h](#).

6.40.2 Constructor & Destructor Documentation

6.40.2.1 `Hardware::Exception::MicroscopeException::MicroscopeException (string m = EXCEPTION_OPENCAM, int n = EXCEPTION_OPENCAM_NR) [inline]`

Definition at line 37 of file [MicroscopeNotFoundException.h](#).

6.40.2.2 `Hardware::Exception::MicroscopeException::~MicroscopeException () [inline]`

Definition at line 39 of file [MicroscopeNotFoundException.h](#).

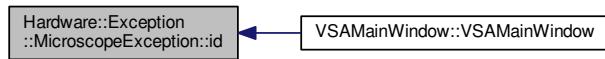
6.40.3 Member Function Documentation

6.40.3.1 `const int* Hardware::Exception::MicroscopeException::id () const [inline]`

Definition at line 41 of file [MicroscopeNotFoundException.h](#).

Referenced by [VSAMainWindow::VSAMainWindow\(\)](#).

Here is the caller graph for this function:



6.40.3.2 `const char* Hardware::Exception::MicroscopeException::what () const [inline]`

Definition at line 40 of file [MicroscopeNotFoundException.h](#).

6.40.4 Member Data Documentation

6.40.4.1 `string Hardware::Exception::MicroscopeException::msg [private]`

Definition at line 44 of file [MicroscopeNotFoundException.h](#).

6.40.4.2 `int Hardware::Exception::MicroscopeException::nr [private]`

Definition at line 45 of file [MicroscopeNotFoundException.h](#).

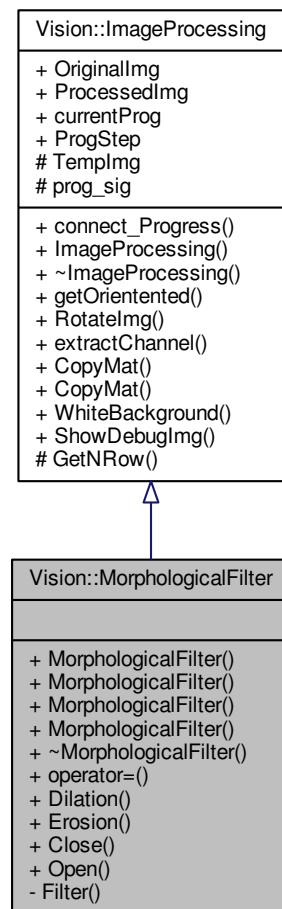
The documentation for this class was generated from the following file:

- /home/peer2peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/[MicroscopeNotFoundException.h](#)

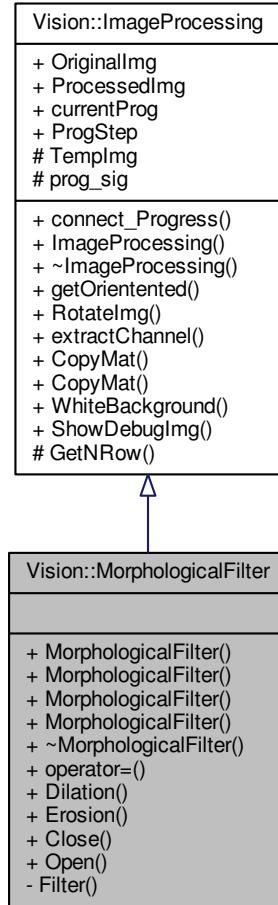
6.41 Vision::MorphologicalFilter Class Reference

```
#include <MorphologicalFilter.h>
```

Inheritance diagram for Vision::MorphologicalFilter:



Collaboration diagram for Vision::MorphologicalFilter:



Public Types

- enum **FilterType** {
 OPEN, **CLOSE**, **ERODE**, **DILATE**, **NONE**
}

Public Member Functions

- MorphologicalFilter ()**
- MorphologicalFilter (FilterType filtertype)**
- MorphologicalFilter (const Mat &src, FilterType filtertype=FilterType::NONE)**
- MorphologicalFilter (const MorphologicalFilter &rhs)**
- ~MorphologicalFilter ()**
- MorphologicalFilter & operator= (MorphologicalFilter &rhs)**
- void Dilation (const Mat &mask, bool chain=false)**
- void Erosion (const Mat &mask, bool chain=false)**
- void Close (const Mat &mask, bool chain=false)**
- void Open (const Mat &mask, bool chain=false)**

Private Member Functions

- void [Filter](#) (const Mat &mask, bool chain, uchar startVal, uchar newVal, uchar switchVal)

Additional Inherited Members

6.41.1 Detailed Description

Definition at line 14 of file [MorphologicalFilter.h](#).

6.41.2 Member Enumeration Documentation

6.41.2.1 enum Vision::MorphologicalFilter::FilterType

Enumerator

OPEN
CLOSE
ERODE
DILATE
NONE

Definition at line 16 of file [MorphologicalFilter.h](#).

6.41.3 Constructor & Destructor Documentation

6.41.3.1 Vision::MorphologicalFilter::MorphologicalFilter()

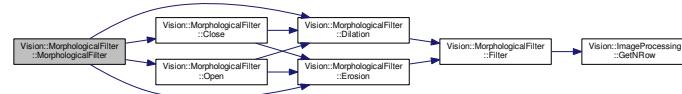
Definition at line 11 of file [MorphologicalFilter.cpp](#).

6.41.3.2 Vision::MorphologicalFilter::MorphologicalFilter(FilterType *filtertype*)

Definition at line 13 of file [MorphologicalFilter.cpp](#).

References [Close\(\)](#), [Dilation\(\)](#), [Erosion\(\)](#), [Open\(\)](#), and [Vision::ImageProcessing::OriginalImg](#).

Here is the call graph for this function:

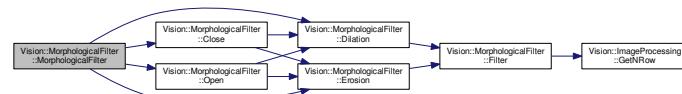


6.41.3.3 Vision::MorphologicalFilter::MorphologicalFilter(const Mat & src, FilterType *filtertype* = FilterType::NONE)

Definition at line 32 of file [MorphologicalFilter.cpp](#).

References [Close\(\)](#), [Dilation\(\)](#), [Erosion\(\)](#), [Open\(\)](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

Here is the call graph for this function:



6.41.3.4 Vision::MorphologicalFilter::MorphologicalFilter(const MorphologicalFilter & rhs)

Definition at line 54 of file [MorphologicalFilter.cpp](#).

References [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TemplImg](#).

6.41.3.5 `Vision::MorphologicalFilter::~MorphologicalFilter()`

Definition at line 60 of file [MorphologicalFilter.cpp](#).

6.41.4 Member Function Documentation

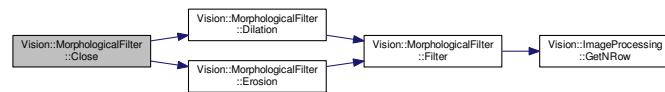
6.41.4.1 `void Vision::MorphologicalFilter::Close (const Mat & mask, bool chain = false)`

Definition at line 76 of file [MorphologicalFilter.cpp](#).

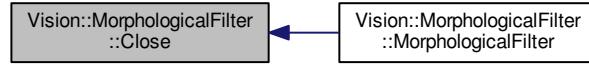
References [Dilation\(\)](#), and [Erosion\(\)](#).

Referenced by [MorphologicalFilter\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.41.4.2 `void Vision::MorphologicalFilter::Dilation (const Mat & mask, bool chain = false)`

Definition at line 81 of file [MorphologicalFilter.cpp](#).

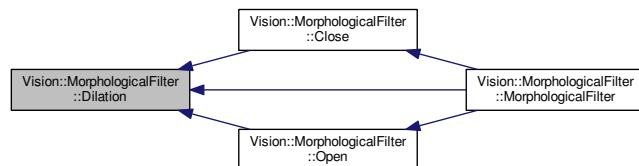
References [Filter\(\)](#).

Referenced by [Close\(\)](#), [MorphologicalFilter\(\)](#), and [Open\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.41.4.3 void Vision::MorphologicalFilter::Erosion (const Mat & mask, bool chain = false)

Definition at line 85 of file [MorphologicalFilter.cpp](#).

References [Filter\(\)](#).

Referenced by [Close\(\)](#), [Vision::Segment::GetEdgesEroding\(\)](#), [MorphologicalFilter\(\)](#), and [Open\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



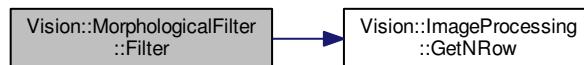
6.41.4.4 void Vision::MorphologicalFilter::Filter (const Mat & mask, bool chain, uchar startVal, uchar newVal, uchar switchVal) [private]

Definition at line 89 of file [MorphologicalFilter.cpp](#).

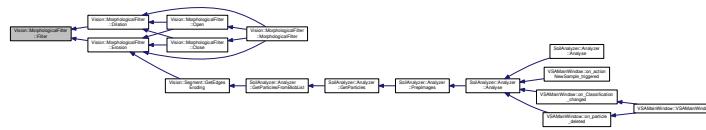
References [EMPTY_CHECK](#), [Vision::ImageProcessing::GetNRow\(\)](#), [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), and [SHOW_DEBUG_IMG](#).

Referenced by [Dilation\(\)](#), and [Erosion\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



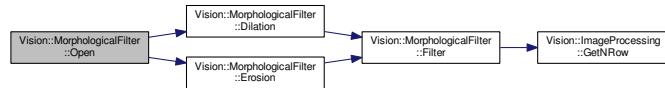
6.41.4.5 void Vision::MorphologicalFilter::Open (const Mat & mask, bool chain = false)

Definition at line 71 of file [MorphologicalFilter.cpp](#).

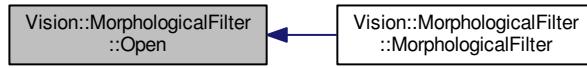
References [Dilation\(\)](#), and [Erosion\(\)](#).

Referenced by [MorphologicalFilter\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.41.4.6 MorphologicalFilter & Vision::MorphologicalFilter::operator= (MorphologicalFilter & rhs)

Definition at line 62 of file [MorphologicalFilter.cpp](#).

References [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TempImg](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/MorphologicalFilter.h](#)

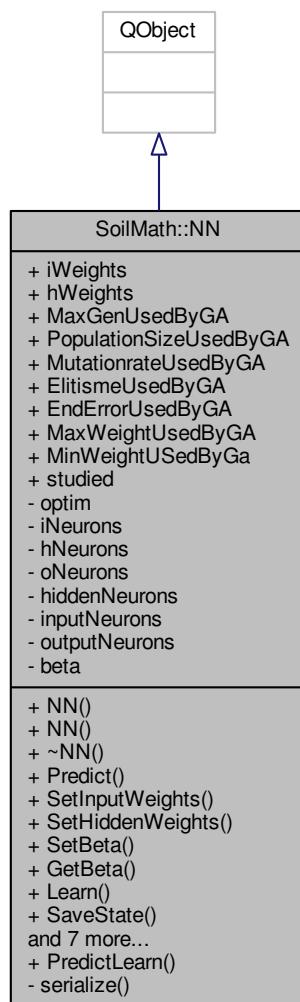
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/MorphologicalFilter.cpp](#)

6.42 SoilMath::NN Class Reference

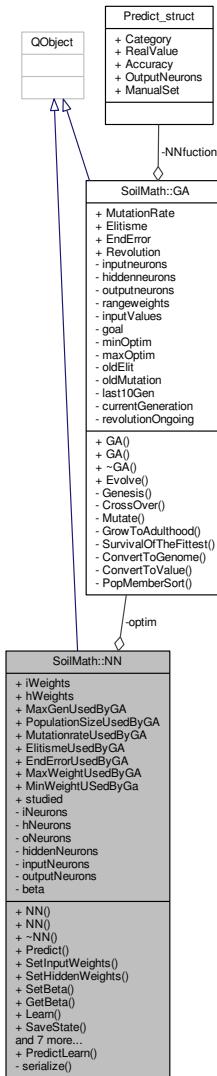
The Neural Network class.

```
#include <NN.h>
```

Inheritance diagram for SoilMath::NN:



Collaboration diagram for SoilMath::NN:



Signals

- void `learnErrorUpdate` (double newError)

Public Member Functions

- `NN (uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)`
`NN constructor for the Neural Net.`
- `NN ()`
`NN constructor for the Neural Net.`
- `virtual ~NN ()`
`~NN virtual deconstructor for the Neural Net`
- `Predict_t Predict (ComplexVect_t input)`
`Predict The prediction function.`
- `void SetInputWeights (Weight_t value)`

-
- `SetInputWeights` a function to set the input weights.
 - `void SetHiddenWeights (Weight_t value)`
 - SetHiddenWeights a function to set the hidden weights.*
 - `void SetBeta (float value)`
 - SetBeta a function to set the beta value.*
 - `float GetBeta ()`
 - `void Learn (InputLearnVector_t input, OutputLearnVector_t cat, uint32_t noOfDescriptorsUsed)`
 - Learn the learning function.*
 - `void SaveState (std::string filename)`
 - SaveState Serialize and save the values of the Neural Net to disk.*
 - `void LoadState (std::string filename)`
 - LoadState Loads the previous saved Neural Net from disk.*
 - `uint32_t GetInputNeurons ()`
 - `void SetInputNeurons (uint32_t value)`
 - `uint32_t GetHiddenNeurons ()`
 - `void SetHiddenNeurons (uint32_t value)`
 - `uint32_t GetOutputNeurons ()`
 - `void SetOutputNeurons (uint32_t value)`

Static Public Member Functions

- `static Predict_t PredictLearn (ComplexVect_t input, Weight_t inputweights, Weight_t hiddenweights, uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)`
 - PredictLearn a static function used in learning of the weights.*

Public Attributes

- `Weight_t iWeights`
- `Weight_t hWeights`
- `uint32_t MaxGenUsedByGA = 200`
- `uint32_t PopulationSizeUsedByGA = 30`
- `float MutationrateUsedByGA = 0.075f`
- `uint32_t ElitismeUsedByGA = 4`
- `float EndErrorUsedByGA = 0.001`
- `float MaxWeightUsedByGA = 50`
- `float MinWeightUsedByGA = -50`
- `bool studied`

Private Member Functions

- `template<class Archive >`
`void serialize (Archive &ar, const unsigned int version)`
 - serialization function*

Private Attributes

- `GA * optim = nullptr`
- `std::vector< float > iNeurons`
- `std::vector< float > hNeurons`
- `std::vector< float > oNeurons`
- `uint32_t hiddenNeurons = 50`
- `uint32_t inputNeurons = 20`
- `uint32_t outputNeurons = 18`
- `float beta`

Friends

- `class boost::serialization::access`

6.42.1 Detailed Description

The Neural Network class.

This class is used to make prediction on large data set. Using self learning algoritmes

Definition at line 33 of file [NN.h](#).

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `SoilMath::NN::NN (uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)`

[NN](#) constructor for the Neural Net.

Parameters

<code>inputneurons</code>	number of input neurons
<code>hiddenneurons</code>	number of hidden neurons
<code>outputneurons</code>	number of output neurons

Definition at line 14 of file [NN.cpp](#).

6.42.2.2 `SoilMath::NN::NN ()`

[NN](#) constructor for the Neural Net.

Definition at line 12 of file [NN.cpp](#).

6.42.2.3 `SoilMath::NN::~NN () [virtual]`

`~NN` virtual deconstructor for the Neural Net

Definition at line 27 of file [NN.cpp](#).

6.42.3 Member Function Documentation

6.42.3.1 `float SoilMath::NN::GetBeta () [inline]`

Definition at line 102 of file [NN.h](#).

References [beta](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

Here is the caller graph for this function:

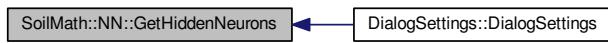
6.42.3.2 `uint32_t SoilMath::NN::GetHiddenNeurons () [inline]`

Definition at line 143 of file [NN.h](#).

References [hiddenNeurons](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

Here is the caller graph for this function:



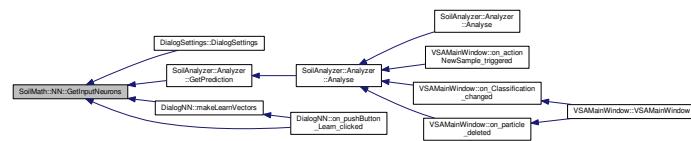
6.42.3.3 `uint32_t SoilMath::NN::GetInputNeurons() [inline]`

Definition at line 140 of file `NN.h`.

References `inputNeurons`.

Referenced by `DialogSettings::DialogSettings()`, `SoilAnalyzer::Analyzer::GetPrediction()`, `DialogNN::makeLearnVectors()`, and `DialogNN::on_pushButton_Learn_clicked()`.

Here is the caller graph for this function:



6.42.3.4 `uint32_t SoilMath::NN::GetOutputNeurons() [inline]`

Definition at line 146 of file `NN.h`.

References `outputNeurons`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogNN::makeLearnVectors()`.

Here is the caller graph for this function:



6.42.3.5 `void SoilMath::NN::Learn(InputLearnVector_t input, OutputLearnVector_t cat, uint32_t noOfDescriptorsUsed)`

Learn the learning function.

Parameters

<code>input</code>	a vector of vectors with complex input values
<code>cat</code>	a vector of vectors with the know output values
<code>noOfDescriptorsUsed</code>	the total number of descriptors which should be used

Definition at line 113 of file `NN.cpp`.

Referenced by `DialogNN::on_pushButton_Learn_clicked()`.

Here is the caller graph for this function:



6.42.3.6 `void SoilMath::NN::learnErrorUpdate(double newError) [signal]`

6.42.3.7 void SoilMath::NN::LoadState (std::string *filename*)

LoadState Loads the previous saved Neural Net from disk.

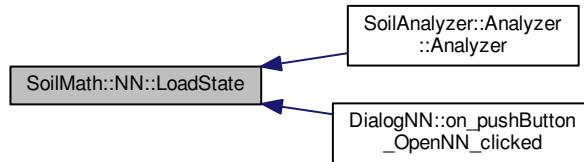
Parameters

<i>filename</i>	a string indicating the file location and name
-----------------	--

Definition at line 34 of file [NN.cpp](#).

Referenced by [SoilAnalyzer::Analyzer::Analyzer\(\)](#), and [DialogNN::on_pushButton_OpenNN_clicked\(\)](#).

Here is the caller graph for this function:

6.42.3.8 Predict_t SoilMath::NN::Predict (ComplexVect_t *input*)

Predict The prediction function.

In this function the neural net is setup and the input which are the complex values describing the contour in the frequency domain serve as input. The absolute value of these im. number because I'm not interested in the orientation of the particle but more in the degree of variations.

Parameters

<i>input</i>	vector of complex input values, these're the Fourier descriptors
--------------	--

Returns

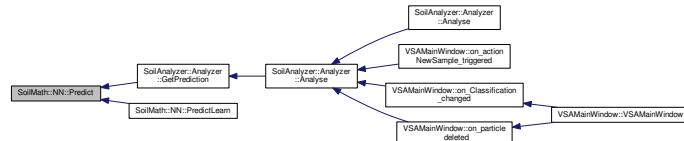
a real valued vector of the output neurons

Definition at line 56 of file [NN.cpp](#).

References [EXCEPTION_NEURAL_NET_NOT_STUDIED](#), [EXCEPTION_NEURAL_NET_NOT_STUDIED_NR](#), [EXCEPTION_SIZE_OF_INPUT_NEURONS](#), [EXCEPTION_SIZE_OF_INPUT_NEURONS_NR](#), [Predict_struct::ManualSet](#), and [Predict_struct::OutputNeurons](#).

Referenced by [SoilAnalyzer::Analyzer::GetPrediction\(\)](#), and [PredictLearn\(\)](#).

Here is the caller graph for this function:

6.42.3.9 Predict_t SoilMath::NN::PredictLearn (ComplexVect_t *input*, Weight_t *inputweights*, Weight_t *hiddenweights*, uint32_t *inputneurons*, uint32_t *hiddenneurons*, uint32_t *outputneurons*) [static]

PredictLearn a static function used in learning of the weights.

It starts a new Neural Network object and passes all the parameters in to this newly created object. After this the predict function is called and the value is returned. This work around was needed to pass the neural network to the Genetic Algorithm class.

Parameters

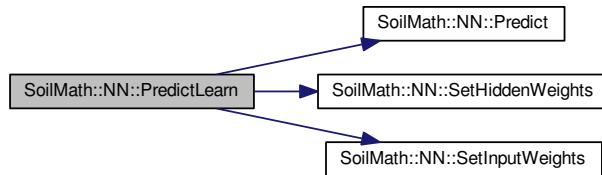
<i>input</i>	a complex vector of input values
<i>inputweights</i>	the input weights
<i>hiddenweights</i>	the hidden weights
<i>inputneurons</i>	the input neurons
<i>hiddenneurons</i>	the hidden neurons
<i>outputneurons</i>	the output neurons

Returns

Definition at line 46 of file [NN.cpp](#).

References [Predict\(\)](#), [SetHiddenWeights\(\)](#), [SetInputWeights\(\)](#), and [studied](#).

Here is the call graph for this function:



6.42.3.10 `void SoilMath::NN::SaveState (std::string filename)`

SaveState Serialize and save the values of the Neural Net to disk.

Save the Neural Net in XML valued text file to disk so that a object can be reconstructed on a latter stadia.

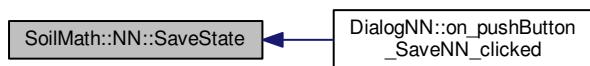
Parameters

<i>filename</i>	a string indicating the file location and name
-----------------	--

Definition at line 40 of file [NN.cpp](#).

Referenced by [DialogNN::on_pushButton_SaveNN_clicked\(\)](#).

Here is the caller graph for this function:



6.42.3.11 `template<class Archive> void SoilMath::NN::serialize (Archive & ar, const unsigned int version) [inline], [private]`

serialization function

Parameters

<i>ar</i>	the object
<i>version</i>	the version of the class

Definition at line 181 of file [NN.h](#).

6.42.3.12 `void SoilMath::NN::SetBeta (float value) [inline]`

SetBeta a function to set the beta value.

Parameters

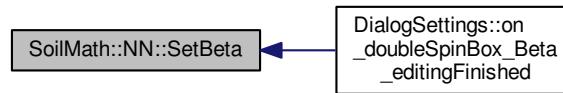
value	a floating value usually between 0.5 and 1.5
-------	--

Definition at line 101 of file [NN.h](#).

References [beta](#).

Referenced by [DialogSettings::on_doubleSpinBox_Beta_editingFinished\(\)](#).

Here is the caller graph for this function:

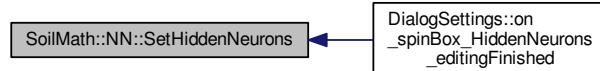


6.42.3.13 void SoilMath::NN::SetHiddenNeurons ([uint32_t](#) value)

Definition at line 150 of file [NN.cpp](#).

Referenced by [DialogSettings::on_spinBox_HiddenNeurons_editingFinished\(\)](#).

Here is the caller graph for this function:



6.42.3.14 void SoilMath::NN::SetHiddenWeights ([Weight_t](#) value) [inline]

SetHiddenWeights a function to set the hidden weights.

Parameters

value	the real valued vector with the values
-------	--

Definition at line 95 of file [NN.h](#).

References [hWeights](#).

Referenced by [PredictLearn\(\)](#).

Here is the caller graph for this function:

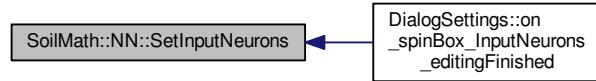


6.42.3.15 void SoilMath::NN::SetInputNeurons ([uint32_t](#) value)

Definition at line 141 of file [NN.cpp](#).

Referenced by [DialogSettings::on_spinBox_InputNeurons_editingFinished\(\)](#).

Here is the caller graph for this function:



6.42.3.16 void SoilMath::NN::SetInputWeights (Weight_t value) [inline]

SetInputWeights a function to set the input weights.

Parameters

<code>value</code>	the real valued vector with the values
--------------------	--

Definition at line 89 of file [NN.h](#).

References [iWeights](#).

Referenced by [PredictLearn\(\)](#).

Here is the caller graph for this function:

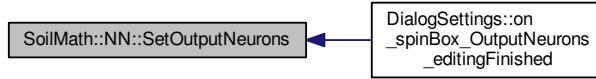


6.42.3.17 void SoilMath::NN::SetOutputNeurons (uint32_t value)

Definition at line 159 of file [NN.cpp](#).

Referenced by [DialogSettings::on_spinBox_OutputNeurons_editingFinished\(\)](#).

Here is the caller graph for this function:



6.42.4 Friends And Related Function Documentation

6.42.4.1 friend class boost::serialization::access [friend]

a private friend class so the serialization can access all the needed functions

Definition at line 172 of file [NN.h](#).

6.42.5 Member Data Documentation

6.42.5.1 float SoilMath::NN::beta [private]

the beta value, this indicates the steepness of the sigmoid function

Definition at line 169 of file [NN.h](#).

Referenced by [GetBeta\(\)](#), and [SetBeta\(\)](#).

6.42.5.2 `uint32_t` [SoilMath::NN::ElitismeUsedByGA](#) = 4

Definition at line 135 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_spinBox_Elitisme_editingFinished\(\)](#).

6.42.5.3 `float` [SoilMath::NN::EndErrorUsedByGA](#) = 0.001

Definition at line 136 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [DialogNN::on_actionAbort_triggered\(\)](#), [DialogSettings::on_doubleSpinBox_endError_editingFinished\(\)](#), and [DialogNN::setupErrorGraph\(\)](#).

6.42.5.4 `uint32_t` [SoilMath::NN::hiddenNeurons](#) = 50 [private]

number of hidden neurons minus bias

Definition at line 166 of file [NN.h](#).

Referenced by [GetHiddenNeurons\(\)](#).

6.42.5.5 `std::vector<float>` [SoilMath::NN::hNeurons](#) [private]

a vector of hidden values, the bias is included and is the first value

Definition at line 162 of file [NN.h](#).

6.42.5.6 `Weight_t` [SoilMath::NN::hWeights](#)

a vector of real valued floating point hidden weight

Definition at line 130 of file [NN.h](#).

Referenced by [SetHiddenWeights\(\)](#).

6.42.5.7 `std::vector<float>` [SoilMath::NN::iNeurons](#) [private]

a vector of input values, the bias is included, the bias is included and is the first value

Definition at line 158 of file [NN.h](#).

6.42.5.8 `uint32_t` [SoilMath::NN::inputNeurons](#) = 20 [private]

number of input neurons minus bias

Definition at line 167 of file [NN.h](#).

Referenced by [GetInputNeurons\(\)](#).

6.42.5.9 `Weight_t` [SoilMath::NN::iWeights](#)

a vector of real valued floating point input weights

Definition at line 129 of file [NN.h](#).

Referenced by [SetInputWeights\(\)](#).

6.42.5.10 `uint32_t` [SoilMath::NN::MaxGenUsedByGA](#) = 200

Definition at line 132 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [DialogSettings::on_spinBox_MaxGen_editingFinished\(\)](#), and [DialogNN::setupErrorGraph\(\)](#).

6.42.5.11 `float` [SoilMath::NN::MaxWeightUsedByGA](#) = 50

Definition at line 137 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_doubleSpinBox_maxWeight_editingFinished\(\)](#).

6.42.5.12 `float` [SoilMath::NN::MinWeightUSedByGa](#) = -50

Definition at line 138 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_doubleSpinBox_MinWeight_editingFinished\(\)](#).

6.42.5.13 float SoilMath::NN::MutationrateUsedByGA = 0.075f

Definition at line 134 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_doubleSpinBox_MutationRate_editingFinished\(\)](#).

6.42.5.14 std::vector<float> SoilMath::NN::oNeurons [private]

a vector of output values

Definition at line 164 of file [NN.h](#).

6.42.5.15 GA* SoilMath::NN::optim = nullptr [private]

Definition at line 157 of file [NN.h](#).

6.42.5.16 uint32_t SoilMath::NN::outputNeurons = 18 [private]

number of output neurons

Definition at line 168 of file [NN.h](#).

Referenced by [GetOutputNeurons\(\)](#).

6.42.5.17 uint32_t SoilMath::NN::PopulationSizeUsedByGA = 30

Definition at line 133 of file [NN.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_spinBox_PopSize_editingFinished\(\)](#).

6.42.5.18 bool SoilMath::NN::studied

Initial value:

=
 false

a value indicating if the weights are a results of a learning curve

Definition at line 149 of file [NN.h](#).

Referenced by [PredictLearn\(\)](#).

The documentation for this class was generated from the following files:

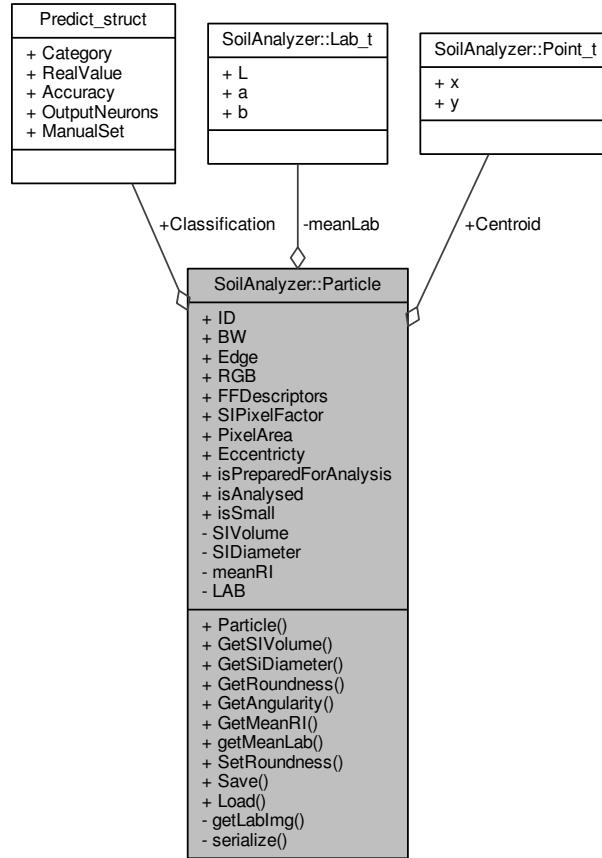
• /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.h

• /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.cpp

6.43 SoilAnalyzer::Particle Class Reference

```
#include <particle.h>
```

Collaboration diagram for SoilAnalyzer::Particle:



Public Types

- `typedef std::vector< Particle > ParticleVector_t`
- `typedef std::vector< double > PSDVector_t`
- `typedef std::vector< uint8_t > ClassVector_t`
- `typedef std::vector< float > floatVector_t`
- `typedef std::vector< double > doubleVector_t`

Public Member Functions

- `Particle()`
- `float GetSIVolume()`
 - Particle::GetSIVolume.*
- `float GetSiDiameter()`
- `uint8_t GetRoundness()`
- `uint8_t GetAngularity()`
- `float GetMeanRI()`
- `Lab_t getMeanLab()`
- `void SetRoundness()`
- `void Save (const std::string &filename)`
 - Particle::Save.*
- `void Load (const std::string &filename)`
 - Particle::Load.*

Public Attributes

- `uint32_t` `ID`
- `cv::Mat` `BW`
- `cv::Mat` `Edge`
- `cv::Mat` `RGB`
- `Point_t` `Centroid` = {0, 0}
- `std::vector< Complex_t >` `FFDescriptors`
- `Predict_t` `Classification`
- `double` `SIPixelFactor` = 0.0111915
- `uint32_t` `PixelArea` = 0
- `double` `Eccentricity` = 1
- `bool` `isPreparedForAnalysis` = false
- `bool` `isAnalysed` = false
- `bool` `isSmall` = false

Private Member Functions

- `void` `getLabelImg ()`
- `template<class Archive >`
 `void` `serialize` (`Archive &ar, const unsigned int` `version`)

Private Attributes

- `float` `SIVolume` = 0.
- `float` `SIDiameter` = 0.
- `float` `meanRI` = 0
- `Lab_t` `meanLab` {0,0,0}
- `cv::Mat` `LAB`

Friends

- `class` `boost::serialization::access`

6.43.1 Detailed Description

Definition at line 28 of file `particle.h`.

6.43.2 Member Typedef Documentation**6.43.2.1 `typedef std::vector<uint8_t>` `SoilAnalyzer::Particle::ClassVector_t`**

a vector used in the classification histogram

Definition at line 34 of file `particle.h`.

6.43.2.2 `typedef std::vector<double>` `SoilAnalyzer::Particle::doubleVector_t`

Definition at line 36 of file `particle.h`.

6.43.2.3 `typedef std::vector<float>` `SoilAnalyzer::Particle::floatVector_t`

Definition at line 35 of file `particle.h`.

6.43.2.4 `typedef std::vector<Particle>` `SoilAnalyzer::Particle::ParticleVector_t`

a vector consisting of individual particles

Definition at line 31 of file `particle.h`.

6.43.2.5 `typedef std::vector<double>` `SoilAnalyzer::Particle::PSDVector_t`

a vector used in the PSD

Definition at line 32 of file `particle.h`.

6.43.3 Constructor & Destructor Documentation

6.43.3.1 SoilAnalyzer::Particle::Particle()

Definition at line 13 of file [particle.cpp](#).

6.43.4 Member Function Documentation

6.43.4.1 uint8_t SoilAnalyzer::Particle::GetAngularity()

Definition at line 79 of file [particle.cpp](#).

References [Predict_struct::Category](#), and [Classification](#).

Referenced by [SetRoundness\(\)](#).

Here is the caller graph for this function:



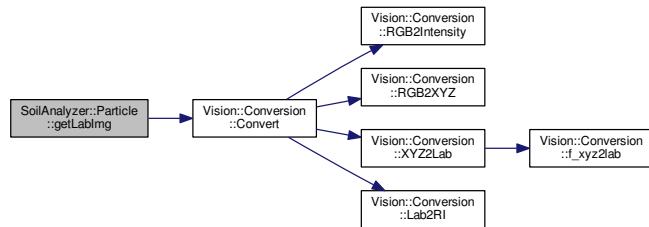
6.43.4.2 void SoilAnalyzer::Particle::getLabImg() [private]

Definition at line 138 of file [particle.cpp](#).

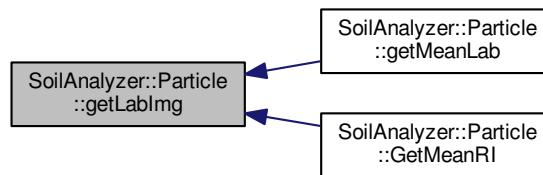
References [Vision::Conversion::CIE_lab](#), [Vision::Conversion::Convert\(\)](#), [LAB](#), [Vision::ImageProcessing::ProcessedImg](#), [Vision::Conversion::RGB](#), and [RGB](#).

Referenced by [getMeanLab\(\)](#), and [GetMeanRI\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

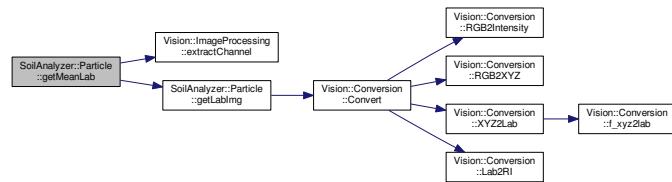


6.43.4.3 `Lab_t SoilAnalyzer::Particle::getMeanLab()`

Definition at line 96 of file [particle.cpp](#).

References [SoilAnalyzer::Lab_t::a](#), [SoilAnalyzer::Lab_t::b](#), [BW](#), [EXCEPTION_NO_IMAGES_PRESENT](#), [EXCEPTION_NO_IMAGES_PRESENT_NR](#), [Vision::ImageProcessing::extractChannel\(\)](#), [getLabImg\(\)](#), [SoilAnalyzer::Lab_t::L](#), [LAB](#), [SoilMath::Stats< T1, T2, T3 >::Mean](#), [meanLab](#), and [RGB](#).

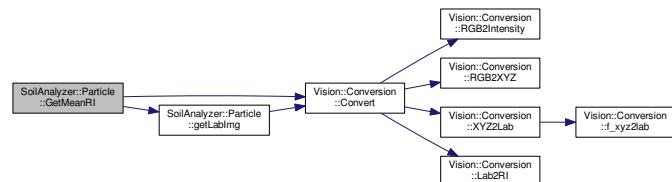
Here is the call graph for this function:

6.43.4.4 `float SoilAnalyzer::Particle::GetMeanRI()`

Definition at line 120 of file [particle.cpp](#).

References [BW](#), [Vision::Conversion::CIE_lab](#), [Vision::Conversion::Convert\(\)](#), [EXCEPTION_NO_IMAGES_PRESENT](#), [EXCEPTION_NO_IMAGES_PRESENT_NR](#), [getLabImg\(\)](#), [LAB](#), [SoilMath::Stats< T1, T2, T3 >::Mean](#), [meanRI](#), [Vision::ImageProcessing::ProcessedImg](#), [RGB](#), and [Vision::Conversion::RI](#).

Here is the call graph for this function:

6.43.4.5 `uint8_t SoilAnalyzer::Particle::GetRoundness()`

Definition at line 84 of file [particle.cpp](#).

References [Predict_struct::Category](#), and [Classification](#).

6.43.4.6 `float SoilAnalyzer::Particle::GetSiDiameter()`

Definition at line 68 of file [particle.cpp](#).

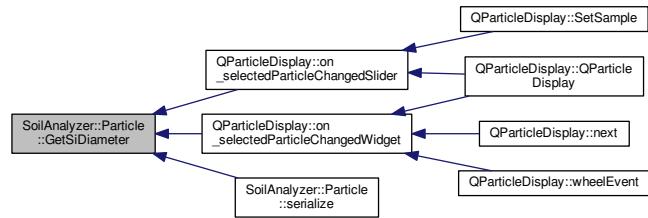
References [SoilMath::calcDiameter\(\)](#), [Eccentricity](#), [EXCEPTION_PARTICLE_NOT_ANALYZED](#), [EXCEPTION_PARTICLE_NOT_ANALYZED_NR](#), [PixelArea](#), [SiDiameter](#), and [SIPixelFactor](#).

Referenced by [QParticleDisplay::on_selectedParticleChangedSlider\(\)](#), [QParticleDisplay::on_selectedParticleChangedWidget\(\)](#), and [serialize\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.4.7 float SoilAnalyzer::Particle::GetSIVolume ()

Particle::GetSIVolume.

Returns

Definition at line 57 of file [particle.cpp](#).

References [SoilMath::calcVolume\(\)](#), [Eccentricity](#), [EXCEPTION_PARTICLE_NOT_ANALYZED](#), [EXCEPTION_PARTICLE_NOT_ANALYZED_←NR](#), [PixelArea](#), [SIPixelFactor](#), and [SIVolume](#).

Here is the call graph for this function:



6.43.4.8 void SoilAnalyzer::Particle::Load (const std::string & filename)

Particle::Load.

Parameters

filename

Definition at line 38 of file [particle.cpp](#).

6.43.4.9 void SoilAnalyzer::Particle::Save (const std::string & filename)

Particle::Save.

Parameters

filename

Definition at line 19 of file [particle.cpp](#).

6.43.4.10 template<class Archive> void SoilAnalyzer::Particle::serialize (Archive & ar, const unsigned int version) [inline], [private]

Definition at line 83 of file [particle.h](#).

References [SoilAnalyzer::Lab_t::a](#), [SoilAnalyzer::Lab_t::b](#), [BW](#), [Classification](#), [Eccentricity](#), [Edge](#), [FFDescriptors](#), [GetSiDiameter\(\)](#), [ID](#), [is←Analyzed](#), [isPreparedForAnalysis](#), [isSmall](#), [SoilAnalyzer::Lab_t::L](#), [meanLab](#), [meanRI](#), [PixelArea](#), [RGB](#), [SIDiameter](#), [SIPixelFactor](#), [SIVolume](#), [SoilAnalyzer::Point_t::x](#), and [SoilAnalyzer::Point_t::y](#).

Here is the call graph for this function:



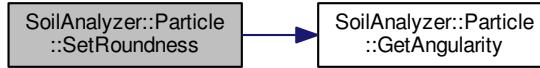
6.43.4.11 void SoilAnalyzer::Particle::SetRoundness ()

Definition at line 89 of file [particle.cpp](#).

References [Predict_struct::Category](#), [Classification](#), [Eccentricity](#), [GetAngularity\(\)](#), and [Predict_struct::ManualSet](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.5 Friends And Related Function Documentation

6.43.5.1 friend class boost::serialization::access [friend]

Definition at line 81 of file [particle.h](#).

6.43.6 Member Data Documentation

6.43.6.1 cv::Mat SoilAnalyzer::Particle::BW

The binary image of the particle

Definition at line 42 of file [particle.h](#).

Referenced by [QParticleDisplay::ConvertParticleToQImage\(\)](#), [getMeanLab\(\)](#), [GetMeanRI\(\)](#), [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [serialize\(\)](#).

6.43.6.2 Point_t SoilAnalyzer::Particle::Centroid = {0, 0}

Definition at line 46 of file [particle.h](#).

6.43.6.3 Predict_t SoilAnalyzer::Particle::Classification

The classification prediction

Definition at line 50 of file [particle.h](#).

Referenced by [GetAngularity\(\)](#), [GetRoundness\(\)](#), [QParticleDisplay::on_selectedParticleChangedSlider\(\)](#), [QParticleDisplay::on_selectedParticleChangedWidget\(\)](#), [serialize\(\)](#), and [SetRoundness\(\)](#).

6.43.6.4 double SoilAnalyzer::Particle::Eccentricity = 1

Definition at line 53 of file [particle.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), [GetSiDiameter\(\)](#), [GetSIVolume\(\)](#), [serialize\(\)](#), and [SetRoundness\(\)](#).

6.43.6.5 cv::Mat SoilAnalyzer::Particle::Edge

The binary edge image of the particle

Definition at line 43 of file [particle.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [serialize\(\)](#).

6.43.6.6 std::vector<Complex_t> SoilAnalyzer::Particle::FFDescriptors

The Fast Fourier Descriptors describing the contour in the Frequency domain

Definition at line 47 of file [particle.h](#).

Referenced by [serialize\(\)](#).

6.43.6.7 uint32_t SoilAnalyzer::Particle::ID

The particle ID

Definition at line 40 of file [particle.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [serialize\(\)](#).

6.43.6.8 bool SoilAnalyzer::Particle::isAnalysed = false

is the particle analyzed

Definition at line 68 of file [particle.h](#).

Referenced by [serialize\(\)](#).

6.43.6.9 bool SoilAnalyzer::Particle::isPreparedForAnalysis = false

is the particle ready for analysis

Definition at line 67 of file [particle.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [serialize\(\)](#).

6.43.6.10 bool SoilAnalyzer::Particle::isSmall = false

Definition at line 69 of file [particle.h](#).

Referenced by [serialize\(\)](#).

6.43.6.11 cv::Mat SoilAnalyzer::Particle::LAB [private]

Definition at line 77 of file [particle.h](#).

Referenced by [getLabImg\(\)](#), [getMeanLab\(\)](#), and [GetMeanRI\(\)](#).

6.43.6.12 Lab_t SoilAnalyzer::Particle::meanLab {0,0,0} [private]

Definition at line 76 of file [particle.h](#).

Referenced by [getMeanLab\(\)](#), and [serialize\(\)](#).

6.43.6.13 float SoilAnalyzer::Particle::meanRI = 0 [private]

Definition at line 75 of file [particle.h](#).

Referenced by [GetMeanRI\(\)](#), and [serialize\(\)](#).

6.43.6.14 uint32_t SoilAnalyzer::Particle::PixelArea = 0

The total area of the binary image

Definition at line 52 of file [particle.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), [GetSiDiameter\(\)](#), [GetSIVolume\(\)](#), and [serialize\(\)](#).

6.43.6.15 cv::Mat SoilAnalyzer::Particle::RGB

The RGB image of the particle

Definition at line 44 of file [particle.h](#).

Referenced by [QParticleDisplay::ConvertParticleToQImage\(\)](#), [getLabImg\(\)](#), [getMeanLab\(\)](#), [GetMeanRI\(\)](#), [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), and [serialize\(\)](#).

6.43.6.16 float SoilAnalyzer::Particle::SiDiameter = 0. [private]

Definition at line 73 of file [particle.h](#).

Referenced by [GetSiDiameter\(\)](#), and [serialize\(\)](#).

6.43.6.17 double SoilAnalyzer::Particle::SIPixelFactor = 0.0111915

The conversion factor from pixel to SI

Definition at line 51 of file [particle.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#), [GetSiDiameter\(\)](#), [GetSIVolume\(\)](#), and [serialize\(\)](#).

6.43.6.18 float SoilAnalyzer::Particle::SIVolume = 0. [private]

The corresponding SI volume

Definition at line 72 of file [particle.h](#).

Referenced by [GetSIVolume\(\)](#), and [serialize\(\)](#).

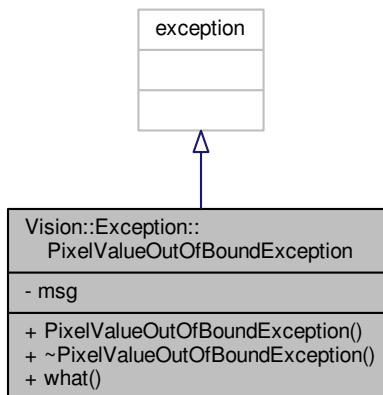
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/particle.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/particle.cpp](#)

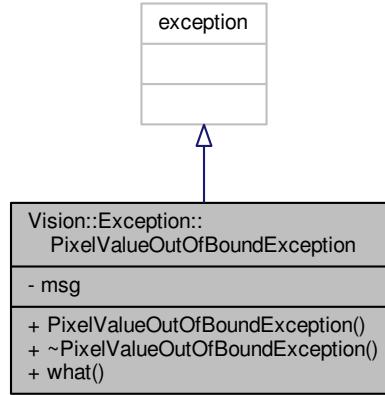
6.44 Vision::Exception::PixelValueOutOfBoundsException Class Reference

```
#include <PixelValueOutOfBoundsException.h>
```

Inheritance diagram for Vision::Exception::PixelValueOutOfBoundsException:



Collaboration diagram for Vision::Exception::PixelValueOutOfBoundsException:



Public Member Functions

- `PixelValueOutOfBoundsException` (string m="Current pixel value out of bounds!")
- `~PixelValueOutOfBoundsException ()` _GLIBCXX_USE_NOEXCEPT
- `const char * what ()` const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string `msg`

6.44.1 Detailed Description

Definition at line 21 of file [PixelValueOutOfBoundsException.h](#).

6.44.2 Constructor & Destructor Documentation

6.44.2.1 `Vision::Exception::PixelValueOutOfBoundsException::PixelValueOutOfBoundsException (string m = "Current pixel value out of bounds!")` [inline]

Definition at line 23 of file [PixelValueOutOfBoundsException.h](#).

6.44.2.2 `Vision::Exception::PixelValueOutOfBoundsException::~PixelValueOutOfBoundsException ()` [inline]

Definition at line 25 of file [PixelValueOutOfBoundsException.h](#).

6.44.3 Member Function Documentation

6.44.3.1 `const char* Vision::Exception::PixelValueOutOfBoundsException::what () const` [inline]

Definition at line 26 of file [PixelValueOutOfBoundsException.h](#).

6.44.4 Member Data Documentation

6.44.4.1 `string Vision::Exception::PixelValueOutOfBoundsException::msg` [private]

Definition at line 26 of file [PixelValueOutOfBoundsException.h](#).

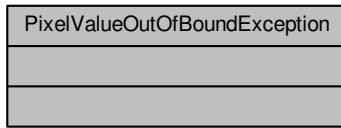
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/PixelValueOutOfBoundsException.h](#)

6.45 PixelValueOutOfBoundsException Class Reference

```
#include <PixelValueOutOfBoundsException.h>
```

Collaboration diagram for PixelValueOutOfBoundsException:



6.45.1 Detailed Description

Exception class which is thrown when an unexpected pixel value has to be computed

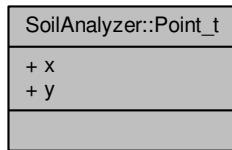
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/PixelValueOutOfBoundsException.h](#)

6.46 SoilAnalyzer::Point_t Struct Reference

```
#include <soilanalyzertypes.h>
```

Collaboration diagram for SoilAnalyzer::Point_t:



Public Attributes

- `double x`
- `double y`

6.46.1 Detailed Description

Definition at line 5 of file [soilanalyzertypes.h](#).

6.46.2 Member Data Documentation

6.46.2.1 `double SoilAnalyzer::Point_t::x`

Definition at line 6 of file [soilanalyzertypes.h](#).

Referenced by [SoilAnalyzer::Particle::serialize\(\)](#).

6.46.2.2 double SoilAnalyzer::Point_t::

Definition at line 7 of file [soilanalyzertypes.h](#).

Referenced by [SoilAnalyzer::Particle::serialize\(\)](#).

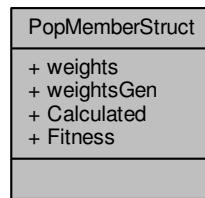
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzertypes.h](#)

6.47 PopMemberStruct Struct Reference

```
#include <SoilMathTypes.h>
```

Collaboration diagram for PopMemberStruct:



Public Attributes

- [Weight_t weights](#)
- [GenVect_t weightsGen](#)
- float [Calculated](#) = 0.0
- float [Fitness](#) = 0.0

6.47.1 Detailed Description

Definition at line 33 of file [SoilMathTypes.h](#).

6.47.2 Member Data Documentation

6.47.2.1 float PopMemberStruct::Calculated = 0.0

the calculated value

Definition at line 36 of file [SoilMathTypes.h](#).

6.47.2.2 float PopMemberStruct::Fitness = 0.0

the fitness of the population member

Definition at line 37 of file [SoilMathTypes.h](#).

Referenced by [SoilMath::GA::PopMemberSort\(\)](#), and [SoilMath::GA::SurvivalOfTheFittest\(\)](#).

6.47.2.3 Weight_t PopMemberStruct::weights

the weights the core of a population member

Definition at line 34 of file [SoilMathTypes.h](#).

Referenced by [SoilMath::GA::Genesis\(\)](#).

6.47.2.4 **GenVect_t PopMemberStruct::weightsGen**

the weights as genomes

Definition at line 35 of file [SoilMathTypes.h](#).

Referenced by [SoilMath::GA::CrossOver\(\)](#), and [SoilMath::GA::Genesis\(\)](#).

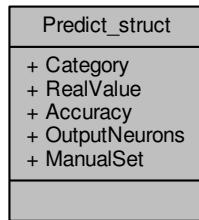
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMathTypes.h](#)

6.48 **Predict_struct Struct Reference**

```
#include <SoilMathTypes.h>
```

Collaboration diagram for Predict_struct:



Public Attributes

- `uint8_t Category = 1`
- `float RealValue = 1.`
- `float Accuracy = 1.`
- `std::vector<float> OutputNeurons`
- `bool ManualSet = true`

6.48.1 Detailed Description

Definition at line 43 of file [SoilMathTypes.h](#).

6.48.2 Member Data Documentation

6.48.2.1 `float Predict_struct::Accuracy = 1.`

the accuracy of the category

Definition at line 47 of file [SoilMathTypes.h](#).

Referenced by [boost::serialization::serialize\(\)](#).

6.48.2.2 `uint8_t Predict_struct::Category = 1`

the category number

Definition at line 44 of file [SoilMathTypes.h](#).

Referenced by [SoilAnalyzer::Particle::GetAngularity\(\)](#), [SoilAnalyzer::Particle::GetRoundness\(\)](#), [QParticleDisplay::on_selectedParticleChangedSlider\(\)](#), [QParticleDisplay::on_selectedParticleChangedWidget\(\)](#), [boost::serialization::serialize\(\)](#), and [SoilAnalyzer::Particle::SetRoundness\(\)](#).

6.48.2.3 `bool Predict_struct::ManualSet = true`

Definition at line 49 of file [SoilMathTypes.h](#).

Referenced by [SoilMath::NN::Predict\(\)](#), and [SoilAnalyzer::Particle::SetRoundness\(\)](#).

6.48.2.4 `std::vector<float> Predict_struct::OutputNeurons`

the output Neurons

Definition at line 48 of file [SoilMathTypes.h](#).

Referenced by [SoilMath::GA::GrowToAdulthood\(\)](#), [DialogNN::makeLearnVectors\(\)](#), [SoilMath::NN::Predict\(\)](#), and [boost::serialization::serialize\(\)](#).

6.48.2.5 `float Predict_struct::RealValue = 1.`

category number as float in order to estimate how precise to outcome is

Definition at line 45 of file [SoilMathTypes.h](#).

Referenced by [boost::serialization::serialize\(\)](#).

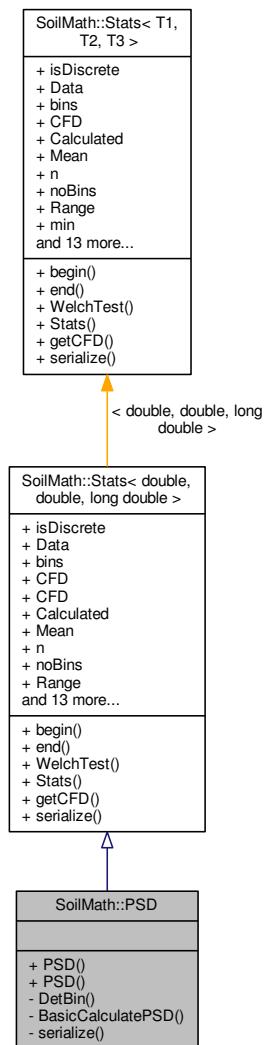
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMathTypes.h](#)

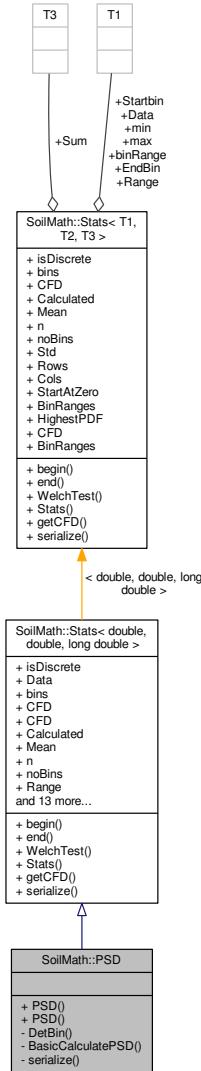
6.49 SoilMath::PSD Class Reference

```
#include <psd.h>
```

Inheritance diagram for SoilMath::PSD:



Collaboration diagram for SoilMath::PSD:



Public Member Functions

- `PSD ()`
- `PSD (double *data, uint32_t nodata, double *binranges, uint32_t nobins, uint32_t endbin)`

Private Member Functions

- `uint32_t DetBin (float value)`
- `void BasicCalculatePSD ()`
- `template<class Archive> void serialize (Archive &ar, const unsigned int version)`

Friends

- `class boost::serialization::access`

Additional Inherited Members

6.49.1 Detailed Description

Definition at line 14 of file [psd.h](#).

6.49.2 Constructor & Destructor Documentation

6.49.2.1 `SoilMath::PSD::PSD()` [inline]

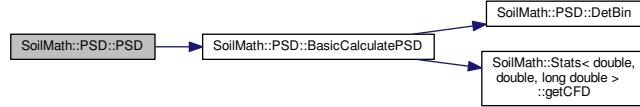
Definition at line 62 of file [psd.h](#).

6.49.2.2 `SoilMath::PSD::PSD(double * data, uint32_t nodata, double * binranges, uint32_t nobins, uint32_t endbin)` [inline]

Definition at line 64 of file [psd.h](#).

References [BasicCalculatePSD\(\)](#), [SoilMath::Stats< double, double, long double >::BinRanges](#), [SoilMath::Stats< double, double, long double >::Cols](#), [SoilMath::Stats< double, double, long double >::Data](#), and [SoilMath::Stats< double, double, long double >::Rows](#).

Here is the call graph for this function:



6.49.3 Member Function Documentation

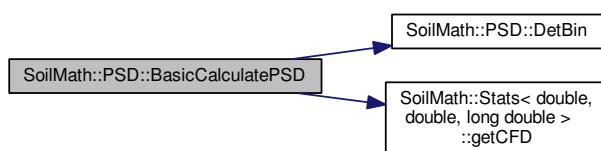
6.49.3.1 `void SoilMath::PSD::BasicCalculatePSD()` [inline], [private]

Definition at line 27 of file [psd.h](#).

References [SoilMath::Stats< double, double, long double >::bins](#), [SoilMath::Stats< double, double, long double >::Calculated](#), [SoilMath::Stats< double, double, long double >::Cols](#), [SoilMath::Stats< double, double, long double >::Data](#), [SoilMath::Stats< double, double, long double >::getCFD\(\)](#), [SoilMath::Stats< double, double, long double >::max](#), [SoilMath::Stats< double, double, long double >::Mean](#), [SoilMath::Stats< double, double, long double >::min](#), [SoilMath::Stats< double, double, long double >::n](#), [SoilMath::Stats< double, double, long double >::Range](#), [SoilMath::Stats< double, double, long double >::Rows](#), [SoilMath::Stats< double, double, long double >::Std](#), and [SoilMath::Stats< double, double, long double >::Sum](#).

Referenced by [PSD\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



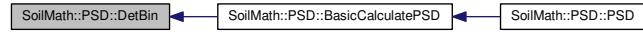
6.49.3.2 `uint32_t SoilMath::PSD::DetBin(float value)` [inline], [private]

Definition at line 16 of file [psd.h](#).

References [SoilMath::Stats< double, double, long double >::BinRanges](#), and [SoilMath::Stats< double, double, long double >::noBins](#).

Referenced by [BasicCalculatePSD\(\)](#).

Here is the caller graph for this function:



6.49.3.3 `template<class Archive> void SoilMath::PSD::serialize(Archive & ar, const unsigned int version)` [inline], [private]

Definition at line 54 of file [psd.h](#).

6.49.4 Friends And Related Function Documentation

6.49.4.1 `friend class boost::serialization::access` [friend]

Definition at line 51 of file [psd.h](#).

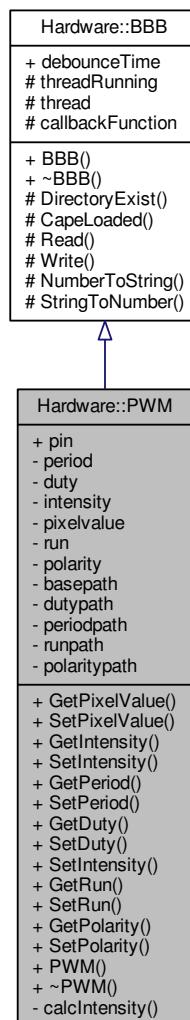
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/psd.h](#)

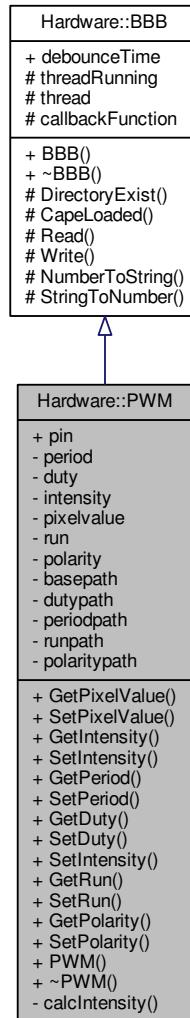
6.50 Hardware::PWM Class Reference

```
#include <PWM.h>
```

Inheritance diagram for Hardware::PWM:



Collaboration diagram for Hardware::PWM:



Public Types

- enum `Pin` { `P8_13`, `P8_19`, `P9_14`, `P9_16` }
- enum `Run` { `On` = 1, `Off` = 0 }
- enum `Polarity` { `Normal` = 1, `Inverted` = 0 }

Public Member Functions

- uint8_t `GetPixelValue` ()
- void `SetPixelValue` (uint8_t value)
Set the output as a corresponding uint8_t value
- float `GetIntensity` ()
- void `SetIntensity` (float value)
Set the intensity level as percentage
- int `GetPeriod` ()
- void `SetPeriod` (int value)

- Set the period of the signal*
 - int [GetDuty](#) ()
 - void [SetDuty](#) (int value)
- Set the duty of the signal*
 - void [SetIntensity](#) ()
 - [Run GetRun](#) ()
 - void [SetRun](#) ([Run](#) value)
- Run the signal*
 - [Polarity GetPolarity](#) ()
 - void [SetPolarity](#) ([Polarity](#) value)
- Set the polarity*
 - [PWM](#) ([Pin](#) pin)
- Constructeur*
 - [~PWM](#) ()

Public Attributes

- [Pin](#) pin

Private Member Functions

- void [calcIntensity](#) ()
Calculate the current intensity

Private Attributes

- int [period](#)
- int [duty](#)
- float [intensity](#)
- uint8_t [pixelvalue](#)
- [Run](#) run
- [Polarity](#) polarity
- string [basepath](#)
- string [dutypath](#)
- string [periodpath](#)
- string [runpath](#)
- string [polaritypath](#)

Additional Inherited Members

6.50.1 Detailed Description

Definition at line 16 of file [PWM.h](#).

6.50.2 Member Enumeration Documentation

6.50.2.1 enum [Hardware::PWM::Pin](#)

Enumerator

- [P8_13](#)**
- [P8_19](#)**
- [P9_14](#)**
- [P9_16](#)**

Definition at line 18 of file [PWM.h](#).

6.50.2.2 enum Hardware::PWM::Polarity

Enumerator

Normal

Inverted

Definition at line 26 of file [PWM.h](#).

6.50.2.3 enum Hardware::PWM::Run

Enumerator

On

Off

Definition at line 23 of file [PWM.h](#).

6.50.3 Constructor & Destructor Documentation

6.50.3.1 Hardware::PWM::PWM (Pin pin)

Constructeur

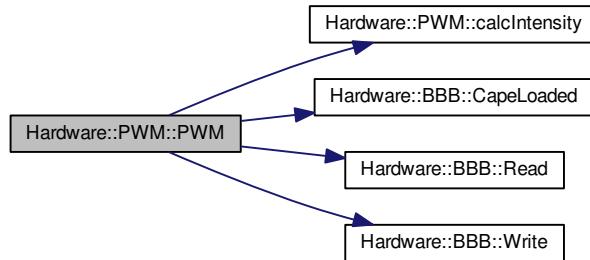
Parameters

<i>pin</i>	Pin
------------	-----

Definition at line 15 of file [PWM.cpp](#).

References [basepath](#), [calcIntensity\(\)](#), [Hardware::BBB::CapeLoaded\(\)](#), [duty](#), [dutypath](#), [OCP_PATH](#), [P8_13](#), [P8_19](#), [P9_14](#), [P9_16](#), [period](#), [periodpath](#), [pin](#), [polarity](#), [polaritypath](#), [PWM_CAPE](#), [Hardware::BBB::Read\(\)](#), [run](#), [runpath](#), [SLOTS](#), and [Hardware::BBB::Write\(\)](#).

Here is the call graph for this function:



6.50.3.2 Hardware::PWM::~PWM ()

Definition at line 65 of file [PWM.cpp](#).

6.50.4 Member Function Documentation

6.50.4.1 void Hardware::PWM::calcIntensity () [private]

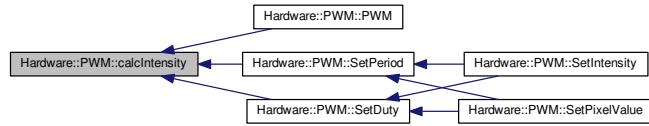
Calculate the current intensity

Definition at line 70 of file [PWM.cpp](#).

References [duty](#), [intensity](#), [Normal](#), [period](#), and [polarity](#).

Referenced by [PWM\(\)](#), [SetDuty\(\)](#), and [SetPeriod\(\)](#).

Here is the caller graph for this function:



6.50.4.2 int Hardware::PWM::GetDuty() [inline]

Definition at line 41 of file [PWM.h](#).

References [duty](#).

6.50.4.3 float Hardware::PWM::GetIntensity() [inline]

Definition at line 35 of file [PWM.h](#).

References [intensity](#).

6.50.4.4 int Hardware::PWM::GetPeriod() [inline]

Definition at line 38 of file [PWM.h](#).

References [period](#).

6.50.4.5 uint8_t Hardware::PWM::GetPixelValue() [inline]

Definition at line 32 of file [PWM.h](#).

References [pixelvalue](#).

6.50.4.6 Polarity Hardware::PWM::GetPolarity() [inline]

Definition at line 48 of file [PWM.h](#).

References [polarity](#).

6.50.4.7 Run Hardware::PWM::GetRun() [inline]

Definition at line 45 of file [PWM.h](#).

References [run](#).

6.50.4.8 void Hardware::PWM::SetDuty(int value)

Set the duty of the signal

Parameters

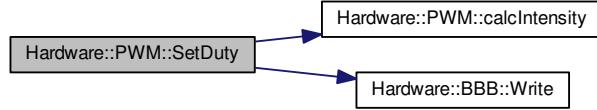
<code>value</code>	<code>duty</code> : int
--------------------	-------------------------

Definition at line 126 of file [PWM.cpp](#).

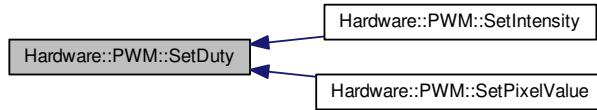
References [calclintensity\(\)](#), [duty](#), [dutypath](#), and [Hardware::BBB::Write\(\)](#).

Referenced by [SetIntensity\(\)](#), and [SetPixelValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.50.4.9 void Hardware::PWM::SetIntensity (float value)

Set the intensity level as percentage

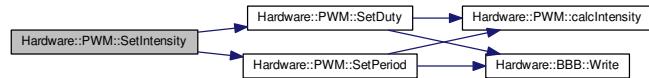
Parameters

<i>value</i>	floating value multiplication factor
--------------	--------------------------------------

Definition at line 90 of file [PWM.cpp](#).

References [duty](#), [Normal](#), [period](#), [polarity](#), [SetDuty\(\)](#), and [SetPeriod\(\)](#).

Here is the call graph for this function:



6.50.4.10 void Hardware::PWM::SetIntensity ()

6.50.4.11 void Hardware::PWM::SetPeriod (int value)

Set the period of the signal

Parameters

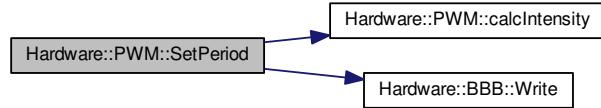
<i>value</i>	period : int
--------------	--------------

Definition at line 114 of file [PWM.cpp](#).

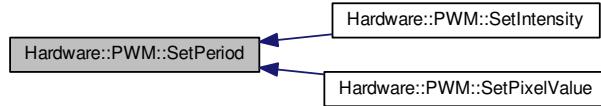
References [calIntensity\(\)](#), [period](#), [periodpath](#), and [Hardware::BBB::Write\(\)](#).

Referenced by [SetIntensity\(\)](#), and [SetPixelValue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.50.4.12 void Hardware::PWM::SetPixelValue (uint8_t value)

Set the output as a corresponding uint8_t value

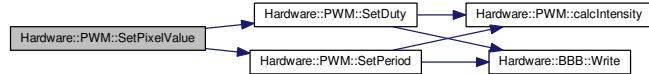
Parameters

value	pixel value 0-255
-------	-------------------

Definition at line 102 of file [PWM.cpp](#).

References [period](#), [pixelvalue](#), [SetDuty\(\)](#), and [SetPeriod\(\)](#).

Here is the call graph for this function:



6.50.4.13 void Hardware::PWM::SetPolarity (Polarity value)

Set the polarity

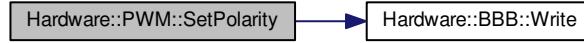
Parameters

value	Normal or Inverted signal
-------	---------------------------

Definition at line 149 of file [PWM.cpp](#).

References [polarity](#), [runpath](#), and [Hardware::BBB::Write\(\)](#).

Here is the call graph for this function:



6.50.4.14 void Hardware::PWM::SetRun (Run value)

Run the signal

Parameters

<code>value</code>	On or Off
--------------------	-----------

Definition at line 138 of file [PWM.cpp](#).

References [run](#), [runpath](#), and [Hardware::BBB::Write\(\)](#).

Here is the call graph for this function:



6.50.5 Member Data Documentation

6.50.5.1 string Hardware::PWM::basepath [private]

Definition at line 62 of file [PWM.h](#).

Referenced by [PWM\(\)](#).

6.50.5.2 int Hardware::PWM::duty [private]

Definition at line 56 of file [PWM.h](#).

Referenced by [calcIntensity\(\)](#), [GetDuty\(\)](#), [PWM\(\)](#), [SetDuty\(\)](#), and [SetIntensity\(\)](#).

6.50.5.3 string Hardware::PWM::dutypath [private]

Definition at line 63 of file [PWM.h](#).

Referenced by [PWM\(\)](#), and [SetDuty\(\)](#).

6.50.5.4 float Hardware::PWM::intensity [private]

Definition at line 57 of file [PWM.h](#).

Referenced by [calcIntensity\(\)](#), and [GetIntensity\(\)](#).

6.50.5.5 int Hardware::PWM::period [private]

Definition at line 55 of file [PWM.h](#).

Referenced by [calcIntensity\(\)](#), [GetPeriod\(\)](#), [PWM\(\)](#), [SetIntensity\(\)](#), [SetPeriod\(\)](#), and [SetPixelValue\(\)](#).

6.50.5.6 string Hardware::PWM::periodpath [private]

Definition at line 64 of file [PWM.h](#).

Referenced by [PWM\(\)](#), and [SetPeriod\(\)](#).

6.50.5.7 Pin Hardware::PWM::pin

Definition at line 30 of file [PWM.h](#).

Referenced by [PWM\(\)](#).

6.50.5.8 uint8_t Hardware::PWM::pixelvalue [private]

Definition at line 58 of file [PWM.h](#).

Referenced by [GetPixelValue\(\)](#), and [SetPixelValue\(\)](#).

6.50.5.9 Polarity Hardware::PWM::polarity [private]

Definition at line 60 of file [PWM.h](#).

Referenced by [calcIntensity\(\)](#), [GetPolarity\(\)](#), [PWM\(\)](#), [SetIntensity\(\)](#), and [SetPolarity\(\)](#).

6.50.5.10 string Hardware::PWM::polaritypath [private]

Definition at line 66 of file [PWM.h](#).

Referenced by [PWM\(\)](#).

6.50.5.11 Run Hardware::PWM::run [private]

Definition at line 59 of file [PWM.h](#).

Referenced by [GetRun\(\)](#), [PWM\(\)](#), and [SetRun\(\)](#).

6.50.5.12 string Hardware::PWM::runpath [private]

Definition at line 65 of file [PWM.h](#).

Referenced by [PWM\(\)](#), [SetPolarity\(\)](#), and [SetRun\(\)](#).

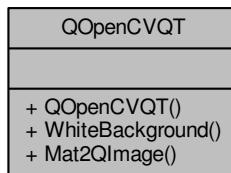
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.cpp](#)

6.51 QOpenCVQT Class Reference

```
#include <qopencvqt.h>
```

Collaboration diagram for QOpenCVQT:



Public Member Functions

- [QOpenCVQT \(\)](#)

Static Public Member Functions

- static cv::Mat [WhiteBackground](#) (const cv::Mat &src)
- static QImage [Mat2QImage](#) (const cv::Mat &src)

6.51.1 Detailed Description

Definition at line 16 of file [qopencvqt.h](#).

6.51.2 Constructor & Destructor Documentation

6.51.2.1 QOpenCVQT::QOpenCVQT()

Definition at line 11 of file [qopencvqt.cpp](#).

6.51.3 Member Function Documentation

6.51.3.1 static QImage QOpenCVQT::Mat2QImage(const cv::Mat & src) [inline], [static]

Definition at line 26 of file [qopencvqt.h](#).

6.51.3.2 static cv::Mat QOpenCVQT::WhiteBackground(const cv::Mat & src) [inline], [static]

Definition at line 20 of file [qopencvqt.h](#).

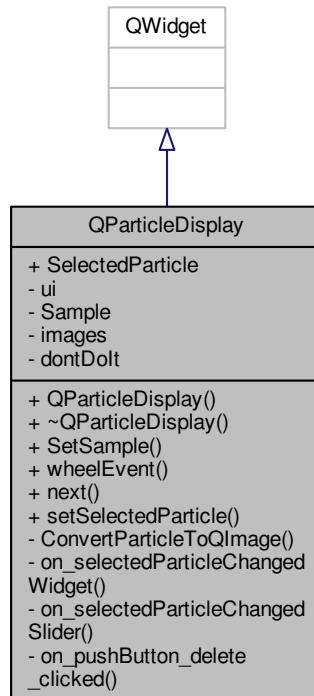
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QOpenCVQT/qopencvqt.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QOpenCVQT/qopencvqt.cpp](#)

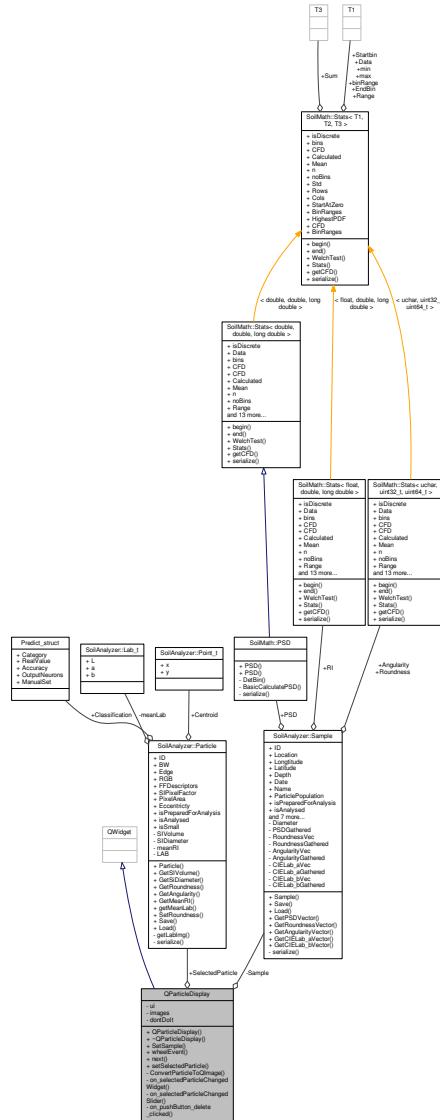
6.52 QParticleDisplay Class Reference

```
#include <qparticledisplay.h>
```

Inheritance diagram for QParticleDisplay:



Collaboration diagram for QParticleDisplay:



Public Slots

- void **setSelectedParticle** (int newValue)

Signals

- void **particleChanged** (int newValue)
 - void **shapeClassificationChanged** (int newValue)
 - void **particleDeleted** ()

Public Member Functions

- `QParticleDisplay (QWidget *parent=0)`
 - `~QParticleDisplay ()`
 - `void SetSample (SoilAnalyzer::Sample *sample)`
 - `void wheelEvent (QWheelEvent *event)`

- void [next \(\)](#)

Public Attributes

- [SoilAnalyzer::Particle * SelectedParticle](#)

Private Slots

- void [on_selectedParticleChangedWidget \(int value\)](#)
- void [on_selectedParticleChangedSlider \(int value\)](#)
- void [on_pushButton_delete_clicked \(\)](#)

Private Member Functions

- [QImage ConvertParticleToQImage \(SoilAnalyzer::Particle *particle\)](#)

Private Attributes

- [Ui::QParticleDisplay * ui](#)
- [SoilAnalyzer::Sample * Sample](#)
- [QVector< QImage > images](#)
- [bool dontDolt = false](#)

6.52.1 Detailed Description

Definition at line 21 of file [qparticledisplay.h](#).

6.52.2 Constructor & Destructor Documentation

6.52.2.1 QParticleDisplay::QParticleDisplay (QWidget * *parent* = 0) [explicit]

Definition at line 11 of file [qparticledisplay.cpp](#).

References [on_selectedParticleChangedSlider\(\)](#), [on_selectedParticleChangedWidget\(\)](#), and [ui](#).

Here is the call graph for this function:



6.52.2.2 QParticleDisplay::~QParticleDisplay ()

Definition at line 22 of file [qparticledisplay.cpp](#).

References [ui](#).

6.52.3 Member Function Documentation

6.52.3.1 QImage QParticleDisplay::ConvertParticleToQImage (SoilAnalyzer::Particle * *particle*) [private]

Definition at line 50 of file [qparticledisplay.cpp](#).

References [SoilAnalyzer::Particle::BW](#), and [SoilAnalyzer::Particle::RGB](#).

Referenced by [SetSample\(\)](#).

Here is the caller graph for this function:



6.52.3.2 void QParticleDisplay::next()

Definition at line 150 of file [qparticledisplay.cpp](#).

References [on_selectedParticleChangedWidget\(\)](#), and [ui](#).

Here is the call graph for this function:



6.52.3.3 void QParticleDisplay::on_pushButton_delete_clicked() [private], [slot]

Definition at line 96 of file [qparticledisplay.cpp](#).

References [SoilAnalyzer::Sample::ChangesSinceLastSave](#), [SoilAnalyzer::Sample::ColorChange](#), [SoilAnalyzer::Sample::ParticleChanged](#)←
[StateAngularity](#), [SoilAnalyzer::Sample::ParticleChangedStatePSD](#), [SoilAnalyzer::Sample::ParticleChangedStateRoundness](#), [particleDeleted\(\)](#),
[SoilAnalyzer::Sample::ParticlePopulation](#), [Sample](#), [SelectedParticle](#), and [ui](#).

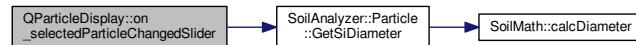
6.52.3.4 void QParticleDisplay::on_selectedParticleChangedSlider(int value) [private], [slot]

Definition at line 124 of file [qparticledisplay.cpp](#).

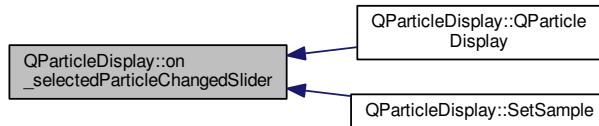
References [Predict_struct::Category](#), [SoilAnalyzer::Particle::Classification](#), [dontDolt](#), [SoilAnalyzer::Particle::GetSiDiameter\(\)](#), [particleChanged\(\)](#),
[SoilAnalyzer::Sample::ParticlePopulation](#), [Sample](#), [SelectedParticle](#), [shapeClassificationChanged\(\)](#), and [ui](#).

Referenced by [QParticleDisplay\(\)](#), and [SetSample\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



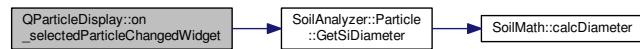
6.52.3.5 void QParticleDisplay::on_selectedParticleChangedWidget (int value) [private], [slot]

Definition at line 110 of file [qparticledisplay.cpp](#).

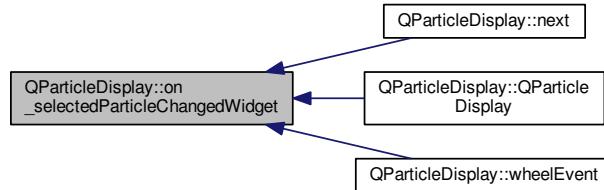
References [Predict_struct::Category](#), [SoilAnalyzer::Particle::Classification](#), [dontDolt](#), [SoilAnalyzer::Particle::GetSiDiameter\(\)](#), [particleChanged\(\)](#), [SoilAnalyzer::Sample::ParticlePopulation](#), [Sample](#), [SelectedParticle](#), [shapeClassificationChanged\(\)](#), and [ui](#).

Referenced by [next\(\)](#), [QParticleDisplay\(\)](#), and [wheelEvent\(\)](#).

Here is the call graph for this function:



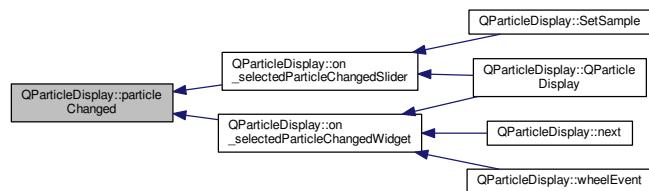
Here is the caller graph for this function:



6.52.3.6 void QParticleDisplay::particleChanged (int newValue) [signal]

Referenced by [on_selectedParticleChangedSlider\(\)](#), and [on_selectedParticleChangedWidget\(\)](#).

Here is the caller graph for this function:



6.52.3.7 void QParticleDisplay::particleDeleted () [signal]

Referenced by [on_pushButton_delete_clicked\(\)](#).

Here is the caller graph for this function:



6.52.3.8 void QParticleDisplay::SetSample (SoilAnalyzer::Sample * *sample*)

Definition at line 35 of file [qparticledisplay.cpp](#).

References [ConvertParticleToQImage\(\)](#), [images](#), [on_selectedParticleChangedSlider\(\)](#), [SoilAnalyzer::Sample::ParticlePopulation](#), [Sample](#), [SelectedParticle](#), and [ui](#).

Here is the call graph for this function:



6.52.3.9 void QParticleDisplay::setSelectedParticle (int *newValue*) [slot]

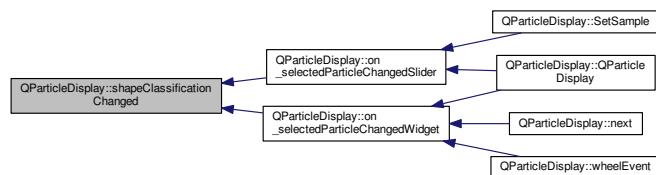
Definition at line 30 of file [qparticledisplay.cpp](#).

References [ui](#).

6.52.3.10 void QParticleDisplay::shapeClassificationChanged (int *newValue*) [signal]

Referenced by [on_selectedParticleChangedSlider\(\)](#), and [on_selectedParticleChangedWidget\(\)](#).

Here is the caller graph for this function:



6.52.3.11 void QParticleDisplay::wheelEvent (QWheelEvent * *event*)

Definition at line 138 of file [qparticledisplay.cpp](#).

References [on_selectedParticleChangedWidget\(\)](#), and [ui](#).

Here is the call graph for this function:



6.52.4 Member Data Documentation

6.52.4.1 `bool QParticleDisplay::dontDolt = false` [private]

Definition at line 51 of file [qparticledisplay.h](#).

Referenced by [on_selectedParticleChangedSlider\(\)](#), and [on_selectedParticleChangedWidget\(\)](#).

6.52.4.2 `QVector<QImage> QParticleDisplay::images` [private]

Definition at line 49 of file [qparticledisplay.h](#).

Referenced by [SetSample\(\)](#).

6.52.4.3 `SoilAnalyzer::Sample* QParticleDisplay::Sample` [private]

Definition at line 48 of file [qparticledisplay.h](#).

Referenced by [on_pushButton_delete_clicked\(\)](#), [on_selectedParticleChangedSlider\(\)](#), [on_selectedParticleChangedWidget\(\)](#), and [SetSample\(\)](#).

6.52.4.4 `SoilAnalyzer::Particle* QParticleDisplay::SelectedParticle`

Definition at line 29 of file [qparticledisplay.h](#).

Referenced by [on_pushButton_delete_clicked\(\)](#), [on_selectedParticleChangedSlider\(\)](#), [on_selectedParticleChangedWidget\(\)](#), and [SetSample\(\)](#).

6.52.4.5 `Ui::QParticleDisplay* QParticleDisplay::ui` [private]

Definition at line 47 of file [qparticledisplay.h](#).

Referenced by [next\(\)](#), [on_pushButton_delete_clicked\(\)](#), [on_selectedParticleChangedSlider\(\)](#), [on_selectedParticleChangedWidget\(\)](#), [QParticleDisplay\(\)](#), [SetSample\(\)](#), [setSelectedParticle\(\)](#), [wheelEvent\(\)](#), and [~QParticleDisplay\(\)](#).

The documentation for this class was generated from the following files:

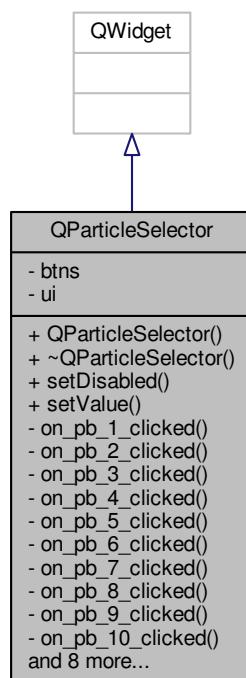
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleDisplay/qparticledisplay.h](#)

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleDisplay/qparticledisplay.cpp](#)

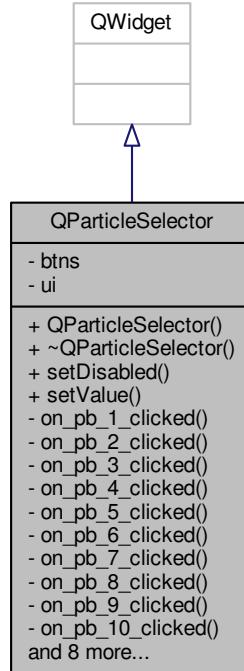
6.53 QParticleSelector Class Reference

```
#include <qparticleselector.h>
```

Inheritance diagram for QParticleSelector:



Collaboration diagram for QParticleSelector:



Public Slots

- void `setValue` (int newValue)

Signals

- void `valueChanged` (int newValue)

Public Member Functions

- `QParticleSelector` (QWidget *parent=0)
- `~QParticleSelector` ()
- void `setDisabled` (bool value, int currentClass=1)

Private Slots

- void `on_pb_1_clicked` (bool checked)
- void `on_pb_2_clicked` (bool checked)
- void `on_pb_3_clicked` (bool checked)
- void `on_pb_4_clicked` (bool checked)
- void `on_pb_5_clicked` (bool checked)
- void `on_pb_6_clicked` (bool checked)
- void `on_pb_7_clicked` (bool checked)
- void `on_pb_8_clicked` (bool checked)
- void `on_pb_9_clicked` (bool checked)
- void `on_pb_10_clicked` (bool checked)
- void `on_pb_11_clicked` (bool checked)

- void [on_pb_12_clicked \(bool checked\)](#)
- void [on_pb_13_clicked \(bool checked\)](#)
- void [on_pb_14_clicked \(bool checked\)](#)
- void [on_pb_15_clicked \(bool checked\)](#)
- void [on_pb_16_clicked \(bool checked\)](#)
- void [on_pb_17_clicked \(bool checked\)](#)
- void [on_pb_18_clicked \(bool checked\)](#)

Private Attributes

- QVector< QPushButton * > [btms](#)
- Ui::QParticleSelector * [ui](#)

6.53.1 Detailed Description

Definition at line 11 of file [qparticleselector.h](#).

6.53.2 Constructor & Destructor Documentation

6.53.2.1 QParticleSelector::QParticleSelector (QWidget * *parent* = 0) [explicit]

Definition at line 4 of file [qparticleselector.cpp](#).

References [btms](#), and [ui](#).

6.53.2.2 QParticleSelector::~QParticleSelector ()

Definition at line 27 of file [qparticleselector.cpp](#).

References [btms](#), and [ui](#).

6.53.3 Member Function Documentation

6.53.3.1 void QParticleSelector::on_pb_10_clicked (bool *checked*) [private], [slot]

Definition at line 104 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.2 void QParticleSelector::on_pb_11_clicked (bool *checked*) [private], [slot]

Definition at line 110 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.3 void QParticleSelector::on_pb_12_clicked (bool *checked*) [private], [slot]

Definition at line 116 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.4 void QParticleSelector::on_pb_13_clicked (bool *checked*) [private], [slot]

Definition at line 122 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.5 void QParticleSelector::on_pb_14_clicked (bool *checked*) [private], [slot]

Definition at line 128 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.6 void QParticleSelector::on_pb_15_clicked (bool *checked*) [private], [slot]

Definition at line 134 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.7 void QParticleSelector::on_pb_16_clicked (bool *checked*) [private], [slot]

Definition at line 140 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.8 void QParticleSelector::on_pb_17_clicked (bool *checked*) [private], [slot]

Definition at line 146 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.9 void QParticleSelector::on_pb_18_clicked (bool *checked*) [private], [slot]

Definition at line 152 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.10 void QParticleSelector::on_pb_1_clicked (bool *checked*) [private], [slot]

Definition at line 50 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.11 void QParticleSelector::on_pb_2_clicked (bool *checked*) [private], [slot]

Definition at line 56 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.12 void QParticleSelector::on_pb_3_clicked (bool *checked*) [private], [slot]

Definition at line 62 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.13 void QParticleSelector::on_pb_4_clicked (bool *checked*) [private], [slot]

Definition at line 68 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.14 void QParticleSelector::on_pb_5_clicked (bool *checked*) [private], [slot]

Definition at line 74 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.15 void QParticleSelector::on_pb_6_clicked (bool *checked*) [private], [slot]

Definition at line 80 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.16 void QParticleSelector::on_pb_7_clicked (bool *checked*) [private], [slot]

Definition at line 86 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.17 void QParticleSelector::on_pb_8_clicked (bool *checked*) [private], [slot]

Definition at line 92 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.18 void QParticleSelector::on_pb_9_clicked (bool *checked*) [private], [slot]

Definition at line 98 of file [qparticleselector.cpp](#).

References [valueChanged\(\)](#).

6.53.3.19 void QParticleSelector::setDisabled (bool *value*, int *currentClass* = 1)

Definition at line 39 of file [qparticleselector.cpp](#).

References [btns](#).

6.53.3.20 `void QParticleSelector::setValue (int newValue) [slot]`

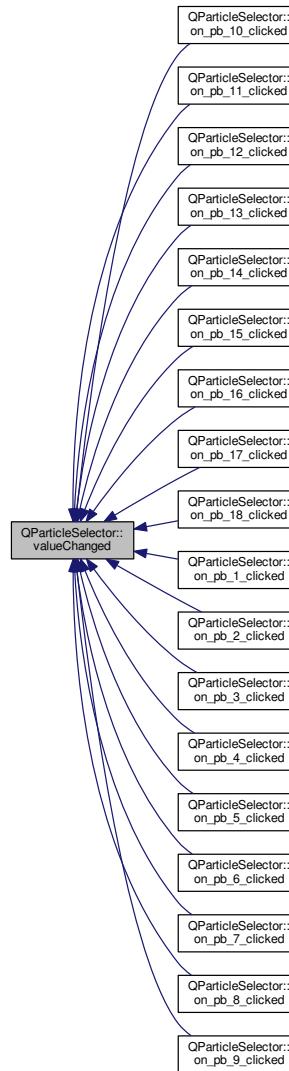
Definition at line 35 of file `qparticleselector.cpp`.

References `btns`.

6.53.3.21 `void QParticleSelector::valueChanged (int newValue) [signal]`

Referenced by `on_pb_10_clicked()`, `on_pb_11_clicked()`, `on_pb_12_clicked()`, `on_pb_13_clicked()`, `on_pb_14_clicked()`, `on_pb_15_clicked()`, `on_pb_16_clicked()`, `on_pb_17_clicked()`, `on_pb_18_clicked()`, `on_pb_1_clicked()`, `on_pb_2_clicked()`, `on_pb_3_clicked()`, `on_pb_4_clicked()`, `on_pb_5_clicked()`, `on_pb_6_clicked()`, `on_pb_7_clicked()`, `on_pb_8_clicked()`, and `on_pb_9_clicked()`.

Here is the caller graph for this function:



6.53.4 Member Data Documentation

6.53.4.1 `QVector<QPushButton *> QParticleSelector::btns [private]`

Definition at line 65 of file `qparticleselector.h`.

Referenced by `QParticleSelector()`, `setDisabled()`, `setValue()`, and `~QParticleSelector()`.

6.53.4.2 Ui::QParticleSelector* QParticleSelector::ui [private]

Definition at line 66 of file [qparticleselector.h](#).

Referenced by [QParticleSelector\(\)](#), and [~QParticleSelector\(\)](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleSelector/qparticleselector.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleSelector/qparticleselector.cpp](#)

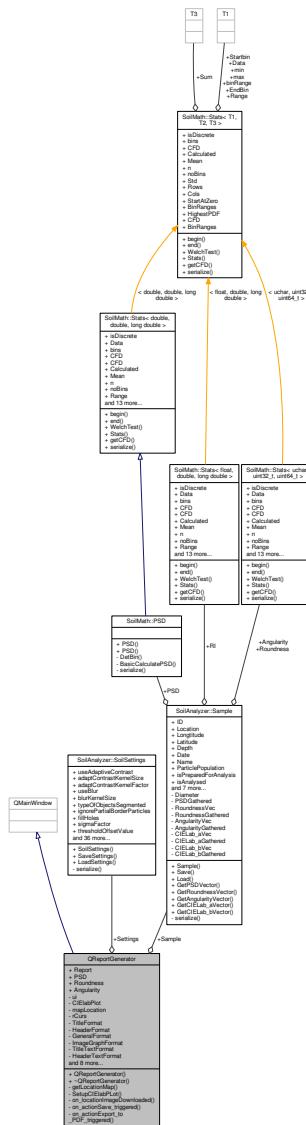
6.54 QReportGenerator Class Reference

```
#include <qreportgenerator.h>
```

Inheritance diagram for QReportGenerator:



Collaboration diagram for QReportGenerator::



Public Member Functions

- `QReportGenerator` (QWidget *parent=0, SoilAnalyzer::Sample *sample=nullptr, SoilAnalyzer::SoilSettings *settings=nullptr, QCustomPlot *psd=nullptr, QCustomPlot *roundness=nullptr, QCustomPlot *angularity=nullptr)
 - `~QReportGenerator` ()

Public Attributes

- `QTextDocument * Report` = `nullptr`
 - `SoilAnalyzer::Sample * Sample` = `nullptr`
 - `SoilAnalyzer::SoilSettings * Settings` = `nullptr`
 - `QCustomPlot * PSD` = `nullptr`
 - `QCustomPlot * Roundness` = `nullptr`
 - `QCustomPlot * Angularity` = `nullptr`

Private Slots

- void [on_locationImageDownloaded](#) (QNetworkReply *reply)
- void [on_actionSave_triggered](#) ()
- void [on_actionExport_to_PDF_triggered](#) ()

Private Member Functions

- void [getLocationMap](#) (double &latitude, double &longitude)
- void [SetupCIElabPLOT](#) ()

Private Attributes

- Ui::QReportGenerator * [ui](#)
- QCustomPlot * [CIElabPlot](#) = nullptr
- QImage * [mapLocation](#) = nullptr
- QTextCursor [rCurs](#)
- QTextBlockFormat [TitleFormat](#)
- QTextBlockFormat [HeaderFormat](#)
- QTextBlockFormat [GeneralFormat](#)
- QTextBlockFormat [ImageGraphFormat](#)
- QTextCharFormat [TitleTextFormat](#)
- QTextCharFormat [HeaderTextFormat](#)
- QTextCharFormat [GtxtFormat](#)
- QTextCharFormat [GFieldtxtFormat](#)
- QTextListFormat [GeneralSampleList](#)
- QTextTableFormat [GeneralTextTableFormat](#)
- QFont [TitleFont](#)
- QFont [HeaderFont](#)
- QFont [GeneralFont](#)
- QFont [FieldFont](#)

6.54.1 Detailed Description

Definition at line 25 of file [qreportgenerator.h](#).

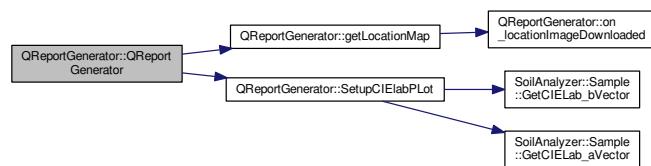
6.54.2 Constructor & Destructor Documentation

6.54.2.1 QReportGenerator::QReportGenerator (QWidget * parent = 0, SoilAnalyzer::Sample * [sample](#) = nullptr, SoilAnalyzer::SoilSettings * [settings](#) = nullptr, QCustomPlot * [psd](#) = nullptr, QCustomPlot * [roundness](#) = nullptr, QCustomPlot * [angularity](#) = nullptr)
[explicit]

Definition at line 4 of file [qreportgenerator.cpp](#).

References [Angularity](#), [SoilAnalyzer::Sample::Angularity](#), [SoilMath::Stats< T1, T2, T3 >::bins](#), [SoilMath::Stats< T1, T2, T3 >::CFD](#), [CIElabPlot](#), [SoilAnalyzer::Sample::Date](#), [SoilAnalyzer::Sample::Depth](#), [FieldFont](#), [GeneralFont](#), [GeneralFormat](#), [GeneralSampleList](#), [GeneralTextTableFormat](#), [getLocationMap\(\)](#), [GFieldtxtFormat](#), [GtxtFormat](#), [HeaderFont](#), [HeaderFormat](#), [HeaderTextFormat](#), [SoilAnalyzer::Sample::ID](#), [ImageGraphFormat](#), [SoilAnalyzer::Sample::Latitude](#), [SoilAnalyzer::Sample::Longitude](#), [SoilMath::Stats< T1, T2, T3 >::max](#), [SoilMath::Stats< T1, T2, T3 >::Mean](#), [SoilMath::Stats< T1, T2, T3 >::min](#), [SoilMath::Stats< T1, T2, T3 >::n](#), [SoilAnalyzer::Sample::Name](#), [SoilAnalyzer::Sample::PSD](#), [SoilMath::Stats< T1, T2, T3 >::Range](#), [rCurs](#), [Report](#), [Roundness](#), [SoilAnalyzer::Sample::Roundness](#), [Sample](#), [Settings](#), [SetupCIElabPLOT\(\)](#), [SoilMath::Stats< T1, T2, T3 >::Std](#), [TitleFont](#), [TitleFormat](#), [TitleTextFormat](#), and [ui](#).

Here is the call graph for this function:



6.54.2.2 `QReportGenerator::~QReportGenerator()`

Definition at line 394 of file [qreportgenerator.cpp](#).

References [CIElabPlot](#), [mapLocation](#), and [ui](#).

6.54.3 Member Function Documentation

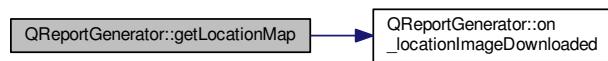
6.54.3.1 `void QReportGenerator::getLocationMap(double & latitude, double & longitude) [private]`

Definition at line 357 of file [qreportgenerator.cpp](#).

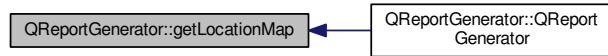
References [on_locationImageDownloaded\(\)](#).

Referenced by [QReportGenerator\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.54.3.2 `void QReportGenerator::on_actionExport_to_PDF_triggered() [private], [slot]`

Definition at line 416 of file [qreportgenerator.cpp](#).

References [Report](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), and [Settings](#).

6.54.3.3 `void QReportGenerator::on_actionSave_triggered() [private], [slot]`

Definition at line 401 of file [qreportgenerator.cpp](#).

References [Report](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), and [Settings](#).

6.54.3.4 `void QReportGenerator::on_locationImageDownloaded(QNetworkReply * reply) [private], [slot]`

Definition at line 376 of file [qreportgenerator.cpp](#).

References [ImageGraphFormat](#), [mapLocation](#), and [Report](#).

Referenced by [getLocationMap\(\)](#).

Here is the caller graph for this function:

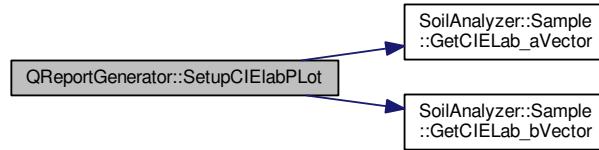
6.54.3.5 `void QReportGenerator::SetupCIElabPPlot() [private]`

Definition at line 431 of file [qreportgenerator.cpp](#).

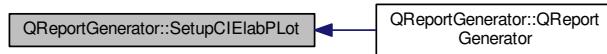
References [CIELabPlot](#), [SoilAnalyzer::Sample::GetCIELab_aVector\(\)](#), [SoilAnalyzer::Sample::GetCIELab_bVector\(\)](#), and [Sample](#).

Referenced by [QReportGenerator\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.54.4 Member Data Documentation

6.54.4.1 QCustomPlot* QReportGenerator::Angularity = nullptr

Definition at line 35 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.2 QCustomPlot* QReportGenerator::CIELabPlot = nullptr [private]

Definition at line 49 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#), [SetupCIELabPPlot\(\)](#), and [~QReportGenerator\(\)](#).

6.54.4.3 QFont QReportGenerator::FieldFont [private]

Definition at line 76 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.4 QFont QReportGenerator::GeneralFont [private]

Definition at line 75 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.5 QTextBlockFormat QReportGenerator::GeneralFormat [private]

Definition at line 61 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.6 QTextListFormat QReportGenerator::GeneralSampleList [private]

Definition at line 69 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.7 QTextTableFormat QReportGenerator::GeneralTextTableFormat [private]

Definition at line 70 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.8 `QTextCharFormat QReportGenerator::GFieldtxtFormat` [private]

Definition at line 67 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.9 `QTextCharFormat QReportGenerator::GtxtFormat` [private]

Definition at line 66 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.10 `QFont QReportGenerator::HeaderFont` [private]

Definition at line 74 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.11 `QTextBlockFormat QReportGenerator::HeaderFormat` [private]

Definition at line 60 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.12 `QTextCharFormat QReportGenerator::HeaderTextFormat` [private]

Definition at line 65 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.13 `QTextBlockFormat QReportGenerator::ImageGraphFormat` [private]

Definition at line 62 of file [qreportgenerator.h](#).

Referenced by [on_locationImageDownloaded\(\)](#), and [QReportGenerator\(\)](#).

6.54.4.14 `QImage* QReportGenerator::mapLocation = nullptr` [private]

Definition at line 54 of file [qreportgenerator.h](#).

Referenced by [on_locationImageDownloaded\(\)](#), and [~QReportGenerator\(\)](#).

6.54.4.15 `QCustomPlot* QReportGenerator::PSD = nullptr`

Definition at line 33 of file [qreportgenerator.h](#).

6.54.4.16 `QTextCursor QReportGenerator::rCurs` [private]

Definition at line 56 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.17 `QTextDocument* QReportGenerator::Report = nullptr`

Definition at line 30 of file [qreportgenerator.h](#).

Referenced by [on_actionExport_to_PDF_triggered\(\)](#), [on_actionSave_triggered\(\)](#), [on_locationImageDownloaded\(\)](#), and [QReportGenerator\(\)](#).

6.54.4.18 `QCustomPlot* QReportGenerator::Roundness = nullptr`

Definition at line 34 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.19 `SoilAnalyzer::Sample* QReportGenerator::Sample = nullptr`

Definition at line 31 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#), and [SetupCIElabPlot\(\)](#).

6.54.4.20 `SoilAnalyzer::SoilSettings* QReportGenerator::Settings = nullptr`

Definition at line 32 of file [qreportgenerator.h](#).

Referenced by [on_actionExport_to_PDF_triggered\(\)](#), [on_actionSave_triggered\(\)](#), and [QReportGenerator\(\)](#).

6.54.4.21 QFont QReportGenerator::TitleFont [private]

Definition at line 73 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.22 QTextBlockFormat QReportGenerator::TitleFormat [private]

Definition at line 59 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.23 QTextCharFormat QReportGenerator::TitleTextFormat [private]

Definition at line 64 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#).

6.54.4.24 Ui::QReportGenerator* QReportGenerator::ui [private]

Definition at line 48 of file [qreportgenerator.h](#).

Referenced by [QReportGenerator\(\)](#), and [~QReportGenerator\(\)](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QReportGenerator/qreportgenerator.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QReportGenerator/qreportgenerator.cpp](#)

6.55 Vision::Segment::Rect Struct Reference

#include <Segment.h>

Collaboration diagram for Vision::Segment::Rect:

Vision::Segment::Rect
+ leftX + leftY + rightX + rightY
+ Rect()

Public Member Functions

- [Rect](#) (uint16_t lx, uint16_t ly, uint16_t rx, uint16_t ry)

Public Attributes

- uint16_t [leftX](#)
- uint16_t [leftY](#)
- uint16_t [rightX](#)
- uint16_t [rightY](#)

6.55.1 Detailed Description

Coordinates for the region of interest

Definition at line 30 of file [Segment.h](#).

6.55.2 Constructor & Destructor Documentation

6.55.2.1 `Vision::Segment::Rect::Rect(uint16_t lx, uint16_t ly, uint16_t rx, uint16_t ry)` [inline]Definition at line 35 of file [Segment.h](#).

6.55.3 Member Data Documentation

6.55.3.1 `uint16_t Vision::Segment::Rect::leftX`

Left X coordinate

Definition at line 31 of file [Segment.h](#).6.55.3.2 `uint16_t Vision::Segment::Rect::leftY`

Left Y coordinate

Definition at line 32 of file [Segment.h](#).6.55.3.3 `uint16_t Vision::Segment::Rect::rightX`

Right X coordinate

Definition at line 33 of file [Segment.h](#).6.55.3.4 `uint16_t Vision::Segment::Rect::rightY`

Right Y coordinate

Definition at line 34 of file [Segment.h](#).

The documentation for this struct was generated from the following file:

- `/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.h`

6.56 Hardware::Microscope::Resolution_t Struct Reference

`#include <Microscope.h>`

Collaboration diagram for Hardware::Microscope::Resolution_t:

Hardware::Microscope ::Resolution_t
+ Width
+ Height
+ format
+ ID
+ to_string()

Public Member Functions

- `std::string to_string()`

Public Attributes

- `uint16_t Width = 2048`
- `uint16_t Height = 1536`
- `PixelFormat format = PixelFormat::MJPG`
- `uint32_t ID`

6.56.1 Detailed Description

Definition at line 57 of file [Microscope.h](#).

6.56.2 Member Function Documentation

6.56.2.1 `std::string Hardware::Microscope::Resolution_t::to_string() [inline]`

Definition at line 61 of file [Microscope.h](#).

6.56.3 Member Data Documentation

6.56.3.1 `PixelFormat Hardware::Microscope::Resolution_t::format = PixelFormat::MJPG`

Definition at line 60 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::getResolutions\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

6.56.3.2 `uint16_t Hardware::Microscope::Resolution_t::Height = 1536`

Definition at line 59 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::getResolutions\(\)](#), [Hardware::Microscope::new_buffer\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

6.56.3.3 `uint32_t Hardware::Microscope::Resolution_t::ID`

Definition at line 76 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::getResolutions\(\)](#).

6.56.3.4 `uint16_t Hardware::Microscope::Resolution_t::Width = 2048`

Definition at line 58 of file [Microscope.h](#).

Referenced by [Hardware::Microscope::getResolutions\(\)](#), [Hardware::Microscope::new_buffer\(\)](#), and [Hardware::Microscope::openCam\(\)](#).

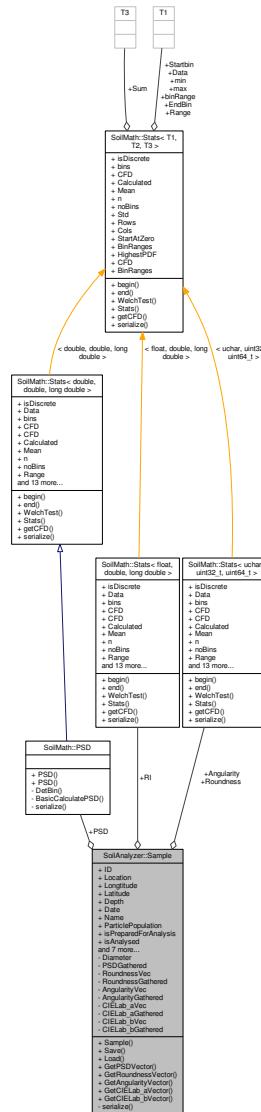
The documentation for this struct was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h](#)

6.57 SoilAnalyzer::Sample Class Reference

```
#include <sample.h>
```

Collaboration diagram for SoilAnalyzer::Sample::



Public Member Functions

- **Sample ()**
Sample::Sample.
 - void **Save** (const std::string &filename)
Sample::Save.
 - void **Load** (const std::string &filename)
Sample::Load.
 - Particle::PSDVector_t * **GetPSDVector ()**
Sample::GetPSDVector.
 - Particle::ClassVector_t * **GetRoundnessVector ()**
 - Particle::ClassVector_t * **GetAngularityVector ()**
 - Particle::doubleVector_t * **GetCIELab_aVector ()**
 - Particle::doubleVector_t * **GetCIELab_bVector ()**

Public Attributes

- `uint32_t ID`
- `std::string Location`
- `double Longitude = 4.62961829999947`
- `double Latitude = 51.8849149`
- `double Depth = 0`
- `std::string Date = "01-09-2015"`
- `std::string Name`
- `Particle::ParticleVector_t ParticlePopulation`
- `SoilMath::PSD PSD`
- `ucharStat_t Roundness`
- `ucharStat_t Angularity`
- `floatStat_t RI`
- `bool isPreparedForAnalysis`
- `bool isAnalysed = false`
- `bool ChangesSinceLastSave = false`
- `bool ParticleChangedStatePSD = false`
- `bool ParticleChangedStateClass = false`
- `bool ParticleChangedStateRoundness = false`
- `bool ParticleChangedStateAngularity = false`
- `bool ColorChange = false`
- `bool IsLoadedFromDisk = false`

Private Member Functions

- `template<class Archive> void serialize(Archive &ar, const unsigned int version)`

Private Attributes

- `Particle::PSDVector_t Diameter`
- `bool PSDGathered = false`
- `Particle::ClassVector_t RoundnessVec`
- `bool RoundnessGathered = false`
- `Particle::ClassVector_t AngularityVec`
- `bool AngularityGathered = false`
- `Particle::doubleVector_t CIELab_aVec`
- `bool CIELab_aGathered = false`
- `Particle::doubleVector_t CIELab_bVec`
- `bool CIELab_bGathered = false`

Friends

- `class boost::serialization::access`

6.57.1 Detailed Description

Definition at line 28 of file [sample.h](#).

6.57.2 Constructor & Destructor Documentation

6.57.2.1 SoilAnalyzer::Sample::Sample()

`Sample::Sample.`

Definition at line 17 of file [sample.cpp](#).

6.57.3 Member Function Documentation

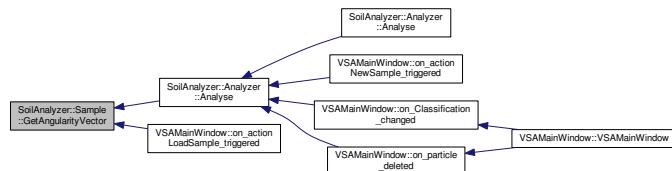
6.57.3.1 Particle::ClassVector_t * SoilAnalyzer::Sample::GetAngularityVector ()

Definition at line 72 of file [sample.cpp](#).

References [AngularityGathered](#), [AngularityVec](#), [ParticleChangedStateAngularity](#), and [ParticlePopulation](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), and [VSAMainWindow::on_actionLoadSample_triggered\(\)](#).

Here is the caller graph for this function:



6.57.3.2 Particle::doubleVector_t * SoilAnalyzer::Sample::GetCIELab_aVector ()

Definition at line 94 of file [sample.cpp](#).

References [CIELab_aGathered](#), [CIELab_aVec](#), [ColorChange](#), and [ParticlePopulation](#).

Referenced by [QReportGenerator::SetupCIELabPPlot\(\)](#).

Here is the caller graph for this function:



6.57.3.3 Particle::doubleVector_t * SoilAnalyzer::Sample::GetCIELab_bVector ()

Definition at line 104 of file [sample.cpp](#).

References [CIELab_bGathered](#), [CIELab_bVec](#), [ColorChange](#), and [ParticlePopulation](#).

Referenced by [QReportGenerator::SetupCIELabPPlot\(\)](#).

Here is the caller graph for this function:



6.57.3.4 Particle::PSDVector_t * SoilAnalyzer::Sample::GetPSDVector ()

[Sample::GetPSDVector](#).

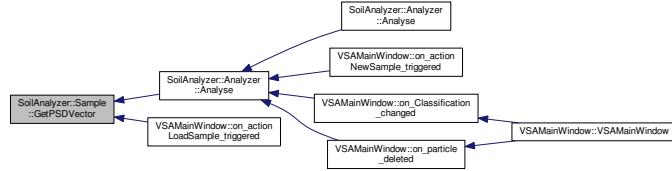
Returns

Definition at line 61 of file [sample.cpp](#).

References [Diameter](#), [ParticleChangedStatePSD](#), [ParticlePopulation](#), and [PSDGathered](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), and [VSAMainWindow::on_actionLoadSample_triggered\(\)](#).

Here is the caller graph for this function:



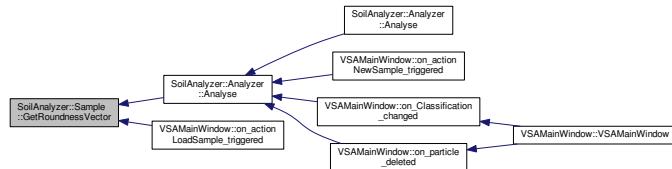
6.57.3.5 Particle::ClassVector_t * SoilAnalyzer::Sample::GetRoundnessVector ()

Definition at line 83 of file [sample.cpp](#).

References `ParticleChangedStateRoundness`, `ParticlePopulation`, `RoundnessGathered`, and `RoundnessVec`.

Referenced by `SoilAnalyzer::Analyzer::Analyse()`, and `VSAMainWindow::on_actionLoadSample_triggered()`.

Here is the caller graph for this function:



6.57.3.6 void SoilAnalyzer::Sample::Load (const std::string & filename)

Sample::Load.

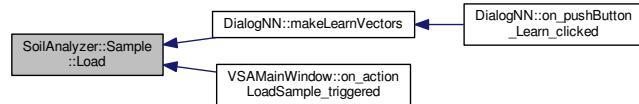
Parameters

<i>filename</i>	
-----------------	--

Definition at line 42 of file [sample.cpp](#).

Referenced by `DialogNN::makeLearnVectors()`, and `VSAMainWindow::on_actionLoadSample_triggered()`

Here is the caller graph for this function:



6.57.3.7 void SoilAnalyzer::Sample::Save (const std::string & filename)

Sample::Save.

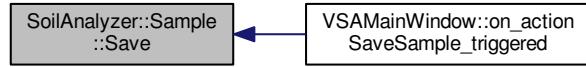
Parameters

<i>filename</i>	
-----------------	--

Definition at line 23 of file [sample.cpp](#).

Referenced by `VSAMainWindow::on_actionSaveSample_triggered()`.

Here is the caller graph for this function:



6.57.3.8 template<class Archive> void SoilAnalyzer::Sample::serialize (Archive & ar, const unsigned int version) [inline], [private]

Definition at line 85 of file [sample.h](#).

References [Angularity](#), [AngularityGathered](#), [AngularityVec](#), [ChangesSinceLastSave](#), [CIELab_aGathered](#), [CIELab_aVec](#), [CIELab_bGathered](#), [CIELab_bVec](#), [ColorChange](#), [Date](#), [Depth](#), [Diameter](#), [ID](#), [isAnalysed](#), [IsLoadedFromDisk](#), [isPreparedForAnalysis](#), [Latitude](#), [Location](#), [Longitude](#), [Name](#), [ParticleChangedStateAngularity](#), [ParticleChangedStateClass](#), [ParticleChangedStatePSD](#), [ParticleChangedStateRoundness](#), [ParticlePopulation](#), [PSD](#), [PSDGathered](#), [RI](#), [Roundness](#), [RoundnessGathered](#), and [RoundnessVec](#).

6.57.4 Friends And Related Function Documentation

6.57.4.1 friend class boost::serialization::access [friend]

Definition at line 83 of file [sample.h](#).

6.57.5 Member Data Documentation

6.57.5.1 ucharStat_t SoilAnalyzer::Sample::Angularity

Definition at line 45 of file [sample.h](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), [VSAMainWindow::on_actionLoadSample_triggered\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), [serialize\(\)](#), and [VSAMainWindow::setAngularityHistogram\(\)](#).

6.57.5.2 bool SoilAnalyzer::Sample::AngularityGathered = false [private]

Definition at line 77 of file [sample.h](#).

Referenced by [GetAngularityVector\(\)](#), and [serialize\(\)](#).

6.57.5.3 Particle::ClassVector_t SoilAnalyzer::Sample::AngularityVec [private]

Definition at line 76 of file [sample.h](#).

Referenced by [GetAngularityVector\(\)](#), and [serialize\(\)](#).

6.57.5.4 bool SoilAnalyzer::Sample::ChangesSinceLastSave = false

Definition at line 62 of file [sample.h](#).

Referenced by [VSAMainWindow::on_actionLoadSample_triggered\(\)](#), [VSAMainWindow::on_actionNewSample_triggered\(\)](#), [VSAMainWindow::on_actionSaveSample_triggered\(\)](#), [VSAMainWindow::on_Classification_changed\(\)](#), [QParticleDisplay::on_pushButton_delete_clicked\(\)](#), and [serialize\(\)](#).

6.57.5.5 bool SoilAnalyzer::Sample::CIELab_aGathered = false [private]

Definition at line 79 of file [sample.h](#).

Referenced by [GetCIELab_aVector\(\)](#), and [serialize\(\)](#).

6.57.5.6 Particle::doubleVector_t SoilAnalyzer::Sample::CIELab_aVec [private]

Definition at line 78 of file [sample.h](#).

Referenced by [GetCIELab_aVector\(\)](#), and [serialize\(\)](#).

6.57.5.7 `bool SoilAnalyzer::Sample::CIELab_bGathered = false` [private]

Definition at line 81 of file `sample.h`.

Referenced by `GetCIELab_bVector()`, and `serialize()`.

6.57.5.8 `Particle::doubleVector_t SoilAnalyzer::Sample::CIELab_bVec` [private]

Definition at line 80 of file `sample.h`.

Referenced by `GetCIELab_bVector()`, and `serialize()`.

6.57.5.9 `bool SoilAnalyzer::Sample::ColorChange = false`

Definition at line 67 of file `sample.h`.

Referenced by `GetCIELab_aVector()`, `GetCIELab_bVector()`, `QParticleDisplay::on_pushButton_delete_clicked()`, and `serialize()`.

6.57.5.10 `std::string SoilAnalyzer::Sample::Date = "01-09-2015"`

Definition at line 37 of file `sample.h`.

Referenced by `QReportGenerator::QReportGenerator()`, and `serialize()`.

6.57.5.11 `double SoilAnalyzer::Sample::Depth = 0`

Definition at line 36 of file `sample.h`.

Referenced by `QReportGenerator::QReportGenerator()`, and `serialize()`.

6.57.5.12 `Particle::PSDVector_t SoilAnalyzer::Sample::Diameter` [private]

The PSD raw data

Definition at line 72 of file `sample.h`.

Referenced by `GetPSDVector()`, and `serialize()`.

6.57.5.13 `uint32_t SoilAnalyzer::Sample::ID`

The sample ID

Definition at line 32 of file `sample.h`.

Referenced by `QReportGenerator::QReportGenerator()`, and `serialize()`.

6.57.5.14 `bool SoilAnalyzer::Sample::isAnalysed = false`

is the sample analyzed

Definition at line 60 of file `sample.h`.

Referenced by `serialize()`.

6.57.5.15 `bool SoilAnalyzer::Sample::IsLoadedFromDisk = false`

Definition at line 69 of file `sample.h`.

Referenced by `SoilAnalyzer::Analyzer::Analyse()`, `VSAMainWindow::on_actionSaveSample_triggered()`, and `serialize()`.

6.57.5.16 `bool SoilAnalyzer::Sample::isPreparedForAnalysis`

Initial value:

=
 false

is the sample ready for analysis, are all the particles extracted

Definition at line 57 of file `sample.h`.

Referenced by `SoilAnalyzer::Analyzer::Analyse()`, `SoilAnalyzer::Analyzer::PrepImages()`, and `serialize()`.

6.57.5.17 `double SoilAnalyzer::Sample::Latitude = 51.8849149`

Definition at line 35 of file `sample.h`.

Referenced by `QReportGenerator::QReportGenerator()`, and `serialize()`.

6.57.5.18 std::string SoilAnalyzer::Sample::Location

The Location where the sample was taken

Definition at line 33 of file [sample.h](#).

Referenced by [serialize\(\)](#).

6.57.5.19 double SoilAnalyzer::Sample::Longitude = 4.629618299999947

Definition at line 34 of file [sample.h](#).

Referenced by [QReportGenerator::QReportGenerator\(\)](#), and [serialize\(\)](#).

6.57.5.20 std::string SoilAnalyzer::Sample::Name

The sample name identifier

Definition at line 38 of file [sample.h](#).

Referenced by [QReportGenerator::QReportGenerator\(\)](#), and [serialize\(\)](#).

6.57.5.21 bool SoilAnalyzer::Sample::ParticleChangedStateAngularity = false

Definition at line 66 of file [sample.h](#).

Referenced by [GetAngularityVector\(\)](#), [VSAMainWindow::on_Classification_changed\(\)](#), [QParticleDisplay::on_pushButton_delete_clicked\(\)](#), and [serialize\(\)](#).

6.57.5.22 bool SoilAnalyzer::Sample::ParticleChangedStateClass = false

Definition at line 64 of file [sample.h](#).

Referenced by [serialize\(\)](#).

6.57.5.23 bool SoilAnalyzer::Sample::ParticleChangedStatePSD = false

Definition at line 63 of file [sample.h](#).

Referenced by [GetPSDVector\(\)](#), [QParticleDisplay::on_pushButton_delete_clicked\(\)](#), and [serialize\(\)](#).

6.57.5.24 bool SoilAnalyzer::Sample::ParticleChangedStateRoundness = false

Definition at line 65 of file [sample.h](#).

Referenced by [GetRoundnessVector\(\)](#), [VSAMainWindow::on_Classification_changed\(\)](#), [QParticleDisplay::on_pushButton_delete_clicked\(\)](#), and [serialize\(\)](#).

6.57.5.25 Particle::ParticleVector_t SoilAnalyzer::Sample::ParticlePopulation

the individual particles of the sample

Definition at line 41 of file [sample.h](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), [SoilAnalyzer::Analyzer::CalcMaxProgressAnalyse\(\)](#), [GetAngularityVector\(\)](#), [GetCIELab_aVector\(\)](#), [GetCIELab_bVector\(\)](#), [GetPSDVector\(\)](#), [GetRoundnessVector\(\)](#), [DialogNN::makeLearnVectors\(\)](#), [VSAMainWindow::on_action>NewSample_triggered\(\)](#), [VSAMainWindow::on_analyzer_finished\(\)](#), [QParticleDisplay::on_pushButton_delete_clicked\(\)](#), [QParticleDisplay::onSelectedParticleChangedSlider\(\)](#), [QParticleDisplay::onSelectedParticleChangedWidget\(\)](#), [SoilAnalyzer::Analyzer::PreImages\(\)](#), [serialize\(\)](#), and [QParticleDisplay::SetSample\(\)](#).

6.57.5.26 SoilMath::PSD SoilAnalyzer::Sample::PSD

The [Particle](#) Size Distribution

Definition at line 43 of file [sample.h](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), [VSAMainWindow::on_actionLoadSample_triggered\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), [serialize\(\)](#), and [VSAMainWindow::SetPSDgraph\(\)](#).

6.57.5.27 bool SoilAnalyzer::Sample::PSDGathered = false [private]

is the raw data gathered

Definition at line 73 of file [sample.h](#).

Referenced by [GetPSDVector\(\)](#), and [serialize\(\)](#).

6.57.5.28 floatStat_t SoilAnalyzer::Sample::RI

The statistical Redness Index data

Definition at line 46 of file [sample.h](#).

Referenced by [serialize\(\)](#).

6.57.5.29 ucharStat_t SoilAnalyzer::Sample::Roundness

Definition at line 44 of file [sample.h](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), [VSAMainWindow::on_actionLoadSample_triggered\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), [serialize\(\)](#), and [VSAMainWindow::setRoundnessHistogram\(\)](#).

6.57.5.30 bool SoilAnalyzer::Sample::RoundnessGathered = false [private]

Definition at line 75 of file [sample.h](#).

Referenced by [GetRoundnessVector\(\)](#), and [serialize\(\)](#).

6.57.5.31 Particle::ClassVector_t SoilAnalyzer::Sample::RoundnessVec [private]

Definition at line 74 of file [sample.h](#).

Referenced by [GetRoundnessVector\(\)](#), and [serialize\(\)](#).

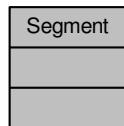
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/sample.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/sample.cpp](#)

6.58 Segment Class Reference

Segmentation algorithms With this class, various segmentation routines can be applied to a greyscale or black and white source image.

Collaboration diagram for Segment:



6.58.1 Detailed Description

Segmentation algorithms With this class, various segmentation routines can be applied to a greyscale or black and white source image.

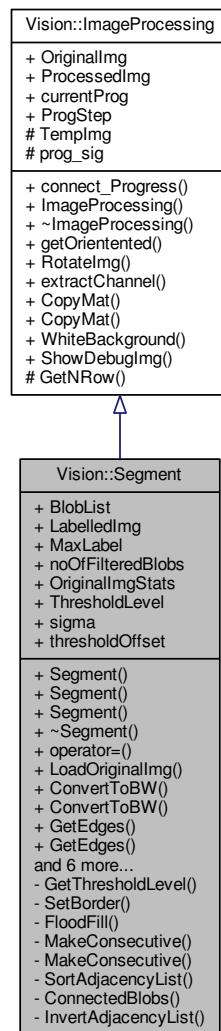
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.cpp](#)

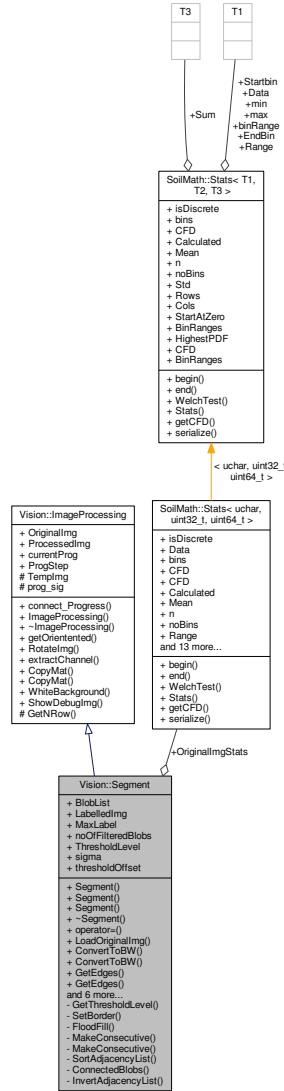
6.59 Vision::Segment Class Reference

```
#include <Segment.h>
```

Inheritance diagram for Vision::Segment:



Collaboration diagram for Vision::Segment:



Classes

- struct Blob
 - struct Rect

Public Types

- enum **TypeOfObjects** { Bright, Dark }
 - enum **Connected** { Four, Eight }
 - enum **SegmentationType** { Normal, LabNeuralNet, GraphMinCut }
 - **typedef struct Vision::Segment::Rect Rect_t**
 - **typedef std::vector< Vision::Segment::Rect_t > RectList_t**
 - **typedef struct Vision::Segment::Blob Blob_t**
 - **typedef std::vector< Blob_t > BlobList_t**

Public Member Functions

- **Segment ()**
Constructor of the Segmentation class.
- **Segment (const Mat &src)**
Constructor of the Segmentation class.
- **Segment (const Segment &rhs)**
- **~Segment ()**
De-constructor.
- **Segment & operator= (Segment &rhs)**
- **void LoadOriginalImg (const Mat &src)**
- **void ConvertToBW (TypeOfObjects Typeobjects)**
- **void ConvertToBW (const Mat &src, Mat &dst, TypeOfObjects Typeobjects)**
- **void GetEdges (bool chain=false, Connected conn=Eight)**
- **void GetEdges (const Mat &src, Mat &dst, bool chain=false, Connected conn=Eight)**
- **void GetEdgesEroding (bool chain=false)**
- **void GetBlobList (bool chain=false, Connected conn=Eight)**
- **void Threshold (uchar t, TypeOfObjects Typeobjects)**
- **void LabelBlobs (bool chain=false, uint16_t minBlobArea=25, Connected conn=Eight)**
- **void RemoveBorderBlobs (uint32_t border=1, bool chain=false)**
- **void FillHoles (bool chain=false)**

Public Attributes

- **BlobList_t BlobList**
- **cv::Mat LabelledImg**
- **uint16_t MaxLabel = 0**
- **uint16_t noOfFilteredBlobs**
- **ucharStat_t OriginalImgStats**
- **uint8_t ThresholdLevel = 0**
- **float sigma = 2**
- **uint32_t thresholdOffset = 4**

Private Member Functions

- **uint8_t GetThresholdLevel (TypeOfObjects TypeObject)**
- **void SetBorder (uchar *P, uchar setValue)**
- **void FloodFill (uchar *O, uchar *P, uint16_t x, uint16_t y, uchar fillValue, uchar OldValue)**
- **void MakeConsecutive (uint16_t *valueArr, uint32_t noElem, uint16_t &maxlabel)**
Segment::MakeConsecutive make the valueArr consecutive numbers.
- **void MakeConsecutive (uint16_t *valueArr, uint16_t *keyArr, uint16_t noElem, uint16_t &maxlabel)**
Segment::MakeConsecutive probably a fault in this function. Don't use.
- **void SortAdjacencyList (std::vector< std::vector< uint16_t >> &adj)**
Segment::SortAdjacencyList Sort the the sub vectors.
- **void ConnectedBlobs (uchar *O, uint16_t *P, std::vector< std::vector< uint16_t >> &adj, uint32_t nCols, uint32_t nRows, Connected conn)**
Segment::ConnectedBlobs Connect all the blobs and created the adjacency list.
- **void InvertAdjacencyList (std::vector< std::vector< uint16_t >> &adj, std::vector< std::vector< uint16_t >> &adjInv)**
Segment::InvertAdjacencyList invert the adjecencylist for upstream (unused)

Additional Inherited Members

6.59.1 Detailed Description

Definition at line 27 of file [Segment.h](#).

6.59.2 Member Typedef Documentation

6.59.2.1 **typedef struct Vision::Segment::Blob Vision::Segment::Blob_t**

Individual blob

6.59.2.2 `typedef std::vector<Blob_t> Vision::Segment::BlobList_t`

Definition at line 54 of file [Segment.h](#).

6.59.2.3 `typedef struct Vision::Segment::Rect Vision::Segment::Rect_t`

Coordinates for the region of interest

6.59.2.4 `typedef std::vector<Vision::Segment::Rect_t> Vision::Segment::RectList_t`

Definition at line 39 of file [Segment.h](#).

6.59.3 Member Enumeration Documentation

6.59.3.1 `enum Vision::Segment::Connected`

Enumerator to indicate how the pixel correlate between each other in a blob

Enumerator

Four Enum Four connected, relation between Center, North, East, South and West

Eight Enum Eight connected, relation between Center, North, NorthEast, East, SouthEast, South, SouthWest, West and NorthWest

Definition at line 65 of file [Segment.h](#).

6.59.3.2 `enum Vision::Segment::SegmentationType`

Enumerator

Normal Segmentation looking at the intensity of an individual pixel

LabNeuralNet Segmentation looking at the chromatic a* and b* of the processed pixel and it's surrounding pixels, feeding it in an Neural Net

GraphMinCut Segmentation using a graph function and the minimum cut

Definition at line 75 of file [Segment.h](#).

6.59.3.3 `enum Vision::Segment::TypeOfObjects`

Enumerator to indicate what kind of object to extract

Enumerator

Bright Enum value Bright object

Dark Enum value Dark object.

Definition at line 58 of file [Segment.h](#).

6.59.4 Constructor & Destructor Documentation

6.59.4.1 `Segment::Segment()`

Constructor of the Segmentation class.

Definition at line 17 of file [Segment.cpp](#).

6.59.4.2 `Segment::Segment(const Mat & src)`

Constructor of the Segmentation class.

Definition at line 20 of file [Segment.cpp](#).

References [LabelledImg](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

6.59.4.3 `Segment::Segment(const Segment & rhs)`

Definition at line 26 of file [Segment.cpp](#).

References [BlobList](#), [LabelledImg](#), [MaxLabel](#), [noOfFilteredBlobs](#), [Vision::ImageProcessing::OriginalImg](#), [OriginalImgStats](#), [Vision::ImageProcessing::ProcessedImg](#), [Vision::ImageProcessing::TemplImg](#), and [ThresholdLevel](#).

6.59.4.4 Segment::~Segment()

De-constructor.

Definition at line 39 of file [Segment.cpp](#).

6.59.5 Member Function Documentation

6.59.5.1 void Segment::ConnectedBlobs (uchar * *O*, uint16_t * *P*, std::vector< std::vector< uint16_t >> & *adj*, uint32_t *nCols*, uint32_t *nRows*, Connected *conn*) [private]

Segment::ConnectedBlobs Connect all the blobs and created the adjacency list.

Parameters

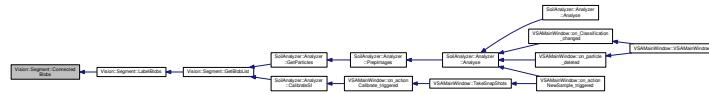
<i>O</i>	
<i>P</i>	
<i>adj</i>	
<i>nCols</i>	
<i>nRows</i>	
<i>conn</i>	

Definition at line 688 of file [Segment.cpp](#).

References [Four](#), and [Vision::ImageProcessing::OriginalImg](#).

Referenced by [LabelBlobs\(\)](#).

Here is the caller graph for this function:

6.59.5.2 void Segment::ConvertToBW (TypeOfObjects *Typeobjects*)

Convert a greyscale image to a BW using an automatic Threshold

Parameters

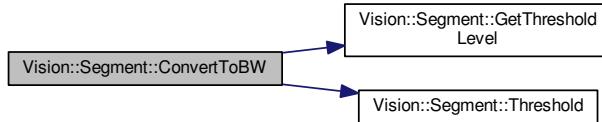
TypeObject is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Definition at line 164 of file [Segment.cpp](#).

References [GetThresholdLevel\(\)](#), and [Threshold\(\)](#).

Referenced by [SoilAnalyzer::Analyzer::CalibrateSI\(\)](#), and [ConvertToBW\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.3 void Segment::ConvertToBW (const Mat & src, Mat & dst, TypeOfObjects Typeobjects)

Convert a greyscale image to a BW using an automatic Threshold

Parameters

<i>src</i>	is the source image as a cv::Mat
<i>dst</i>	destination image as a cv::Mat
<i>TypeObject</i>	is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Definition at line 153 of file [Segment.cpp](#).

References [ConvertToBW\(\)](#), [LabelledImg](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

Here is the call graph for this function:



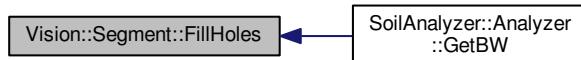
6.59.5.4 void Segment::FillHoles (bool chain = false)

Definition at line 615 of file [Segment.cpp](#).

References [CHAIN_PROCESS](#), [EMPTY_CHECK](#), [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), and [Vision::ImageProcessing::TemplImg](#).

Referenced by [SoilAnalyzer::Analyzer::GetBW\(\)](#).

Here is the caller graph for this function:



6.59.5.5 void Vision::Segment::FloodFill (uchar * O, uchar * P, uint16_t x, uint16_t y, uchar fillValue, uchar OldValue) [private]

6.59.5.6 void Segment::GetBlobList (bool chain = false, Connected conn = Eight)

Create a BlobList subtracting each individual blob out of a Labelled image. If the labelled image is empty build a new one with a BW image.

Parameters

<i>conn</i>	set the pixel connection eight or four
<i>chain</i>	use the results from the previous operation default value = false;

Definition at line 534 of file [Segment.cpp](#).

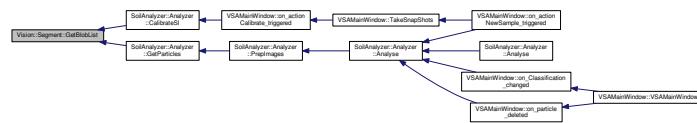
References [SoilMath::Stats< T1, T2, T3 >::bins](#), [BlobList](#), [EMPTY_CHECK](#), [SoilMath::Stats< T1, T2, T3 >::EndBin](#), [LabelBlobs\(\)](#), [LabelledImg](#), [MaxLabel](#), and [Vision::ImageProcessing::OriginalImg](#).

Referenced by [SoilAnalyzer::Analyzer::CalibrateSI\(\)](#), and [SoilAnalyzer::Analyzer::GetParticles\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.7 void Segment::GetEdges (bool chain = false, Connected conn = Eight)

Create a BW image with only edges from a BW image

Parameters

<i>conn</i>	set the pixel connection eight or four
<i>chain</i>	use the results from the previous operation default value = false;

Definition at line 399 of file [Segment.cpp](#).

References [CHAIN_PROCESS](#), [EMPTY_CHECK](#), [Four](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

Referenced by [GetEdges\(\)](#).

Here is the caller graph for this function:



6.59.5.8 void Segment::GetEdges (const Mat & src, Mat & dst, bool chain = false, Connected conn = Eight)

Create a BW image with only edges from a BW image

Parameters

<i>src</i>	source image as a const cv::Mat
<i>dst</i>	destination image as a cv::Mat
<i>conn</i>	set the pixel connection eight or four
<i>chain</i>	use the results from the previous operation default value = false;

Definition at line 389 of file [Segment.cpp](#).

References [GetEdges\(\)](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

Here is the call graph for this function:



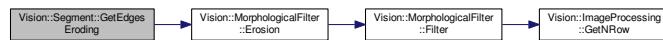
6.59.5.9 void Segment::GetEdgesEroding (bool chain = false)

Definition at line 483 of file [Segment.cpp](#).

References [CHAIN_PROCESS](#), [EMPTY_CHECK](#), [Vision::MorphologicalFilter::Erosion\(\)](#), [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), [SHOW_DEBUG_IMG](#), and [Vision::ImageProcessing::TemplImg](#).

Referenced by [SoilAnalyzer::Analyzer::GetParticlesFromBlobList\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.10 uint8_t Segment::GetThresholdLevel (TypeOfObjects TypeObject) [private]

Determine the threshold level by iteration, between two distribution, presumably back- and foreground. It works towards the average of the two averages and finally sets the threshold with two time the standard deviation from the mean of the set object

Parameters

`TypeObject` is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Returns

The threshold level as an `uint8_t`

Definition at line 69 of file [Segment.cpp](#).

References [SoilMath::Stats< T1, T2, T3 >::bins](#), [Bright](#), [Dark](#), [EMPTY_CHECK](#), [SoilMath::Stats< T1, T2, T3 >::Mean](#), [Vision::ImageProcessing::OriginalImg](#), [OriginalImgStats](#), [sigma](#), [SoilMath::Stats< T1, T2, T3 >::Std](#), and [thresholdOffset](#).

Referenced by [ConvertToBW\(\)](#).

Here is the caller graph for this function:



6.59.5.11 void Segment::InvertAdjacencyList (std::vector< std::vector< uint16_t > > & adj, std::vector< std::vector< uint16_t > > & adjInv) [private]

`Segment::InvertAdjacencyList` invert the adjecencylist for upstream (unused)

Parameters

<i>adj</i>	
<i>adjInv</i>	

Definition at line 801 of file [Segment.cpp](#).

6.59.5.12 void Segment::LabelBlobs (bool *chain* = false, uint16_t *minBlobArea* = 25, Connected *conn* = Eight)

Label all the individual blobs in a BW source image. The result are written to the labelledImg as an ushort

Parameters

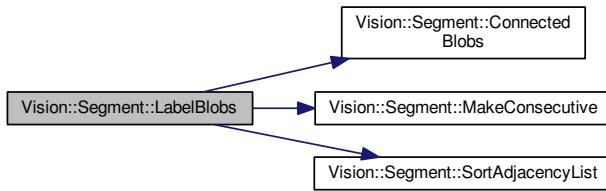
<i>conn</i>	set the pixel connection eight or four
<i>chain</i>	use the results from the previous operation default value = false;
<i>minBlobArea</i>	minimum area when an artifact is considered a blob

Definition at line 316 of file [Segment.cpp](#).

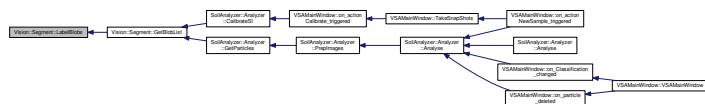
References [ConnectedBlobs\(\)](#), [EMPTY_CHECK](#), [LabelledImg](#), [MakeConsecutive\(\)](#), [MaxLabel](#), [Vision::ImageProcessing::OriginalImg](#), [Vision::ImageProcessing::ProcessedImg](#), [SortAdjacencyList\(\)](#), and [Vision::ImageProcessing::TempImg](#).

Referenced by [GetBlobList\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.13 void Segment::LoadOriginalImg (const Mat & *src*)

Definition at line 56 of file [Segment.cpp](#).

References [LabelledImg](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

6.59.5.14 void Segment::MakeConsecutive (uint16_t * *valueArr*, uint32_t *noElem*, uint16_t & *maxLabel*) [private]

[Segment::MakeConsecutive](#) make the *valueArr* consequative numbers.

Parameters

<i>valueArr</i>	
<i>noElem</i>	
<i>maxLabel</i>	

Definition at line 819 of file [Segment.cpp](#).

Referenced by [LabelBlobs\(\)](#).

Here is the caller graph for this function:



6.59.5.15 void Segment::MakeConsecutive (uint16_t * valueArr, uint16_t * keyArr, uint16_t noElem, uint16_t & maxlabel) [private]

[Segment::MakeConsecutive](#) probably a fault in this function. Don't use.

Parameters

<code>valueArr</code>
<code>keyArr</code>
<code>noElem</code>
<code>maxlabel</code>

Definition at line 845 of file [Segment.cpp](#).

6.59.5.16 Segment & Segment::operator= (Segment & rhs)

Definition at line 41 of file Segment.cpp.
References BlobList, LabelledImg, MaxLabel, noOfFilteredBlobs, Vision::ImageProcessing::OriginalImg, OriginalImgStats, Vision::Image<_>.

Processing..ProcessedImage, Vision..ImageProcessing..Templimg, and ThresholdLevel

6.59.5.17 void Segment::RemoveBorderBlobs (uint32_

Remove the

Parameters	
<i>conn</i>	set the pixel connection eight or four
<i>chain</i>	use the results from the previous operation default value = false;

Definition at line 245 of file Segment.cpp

References CHAIN_PROCESS, EMPTY_CHECK, Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, SHOW_DEBUG_IMG, and Vision::ImageProcessing::TempImg.

Referenced by [SoilAnalyzer::Analyzer::GetBW\(\)](#).

Here is the caller graph for this function:



6.59.5.18 void Segment::SetBorder (uchar * *P*, uchar *setValue*) [private]

Set all the border pixels to a set value

Parameters

<i>*P</i>	uchar pointer to the Mat.data
<i>setValue</i>	uchar the value which is written to the border pixels

Definition at line 208 of file [Segment.cpp](#).

References `EMPTY_CHECK`, and `Vision::ImageProcessing::OriginalImg`.

6.59.5.19 void Segment::SortAdjacencyList (std::vector< std::vector< uint16_t >> & adj) [private]

Segment::SortAdjacencyList Sort the the sub vectors.

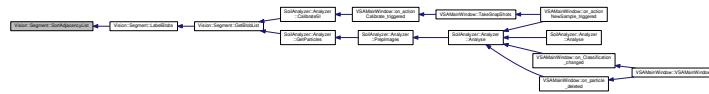
Parameters

<i>adj</i>	std::vector<std::vector<uint16_t>> &adj
------------	---

Definition at line 658 of file [Segment.cpp](#).

Referenced by [LabelBlobs\(\)](#).

Here is the caller graph for this function:

6.59.5.20 void Segment::Threshold (uchar *t*, TypeOfObjects *Typeobjects*)

Convert a greyscale image to a BW

Parameters

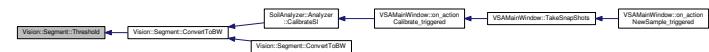
<i>t</i>	uchar set the value which is the tipping point
<i>TypeObject</i>	is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Definition at line 176 of file [Segment.cpp](#).

References [Bright](#), [EMPTY_CHECK](#), [Vision::ImageProcessing::OriginalImg](#), and [Vision::ImageProcessing::ProcessedImg](#).

Referenced by [ConvertToBW\(\)](#).

Here is the caller graph for this function:



6.59.6 Member Data Documentation

6.59.6.1 BlobList_t Vision::Segment::BlobList

vector with all the individual blobs

Definition at line 55 of file [Segment.h](#).

Referenced by [SoilAnalyzer::Analyzer::CalibrateSI\(\)](#), [GetBlobList\(\)](#), [SoilAnalyzer::Analyzer::GetParticles\(\)](#), [operator=\(\)](#), and [Segment\(\)](#).

6.59.6.2 cv::Mat Vision::Segment::LabelledImg

Image with each individual blob labeled with a individual number

Definition at line 83 of file [Segment.h](#).

Referenced by [ConvertToBW\(\)](#), [GetBlobList\(\)](#), [LabelBlobs\(\)](#), [LoadOriginalImg\(\)](#), [operator=\(\)](#), and [Segment\(\)](#).

6.59.6.3 uint16_t Vision::Segment::MaxLabel = 0

Maximum labels found in the labelled image

Definition at line 85 of file [Segment.h](#).

Referenced by [GetBlobList\(\)](#), [LabelBlobs\(\)](#), [operator=\(\)](#), and [Segment\(\)](#).

6.59.6.4 uint16_t Vision::Segment::noOfFilteredBlobs

Initial value:

=
0

Total numbers of blobs that where filtered beacuse the where smaller than the minBlobArea

Definition at line 86 of file [Segment.h](#).

Referenced by [operator=\(\)](#), and [Segment\(\)](#).

6.59.6.5 ucharStat_t Vision::Segment::OriginalImgStats

Statistical data from the original image

Definition at line 90 of file [Segment.h](#).

Referenced by [GetThresholdLevel\(\)](#), [operator=\(\)](#), and [Segment\(\)](#).

6.59.6.6 float Vision::Segment::sigma = 2

Definition at line 93 of file [Segment.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetBW\(\)](#), and [GetThresholdLevel\(\)](#).

6.59.6.7 uint8_t Vision::Segment::ThresholdLevel = 0

Current calculated threshold level

Definition at line 91 of file [Segment.h](#).

Referenced by [operator=\(\)](#), and [Segment\(\)](#).

6.59.6.8 uint32_t Vision::Segment::thresholdOffset = 4

Definition at line 94 of file [Segment.h](#).

Referenced by [GetThresholdLevel\(\)](#).

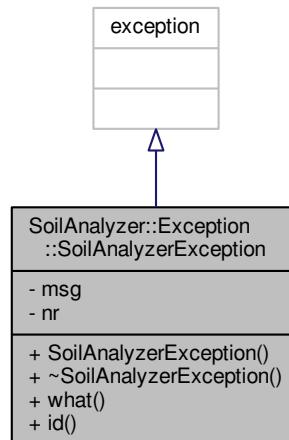
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.cpp](#)

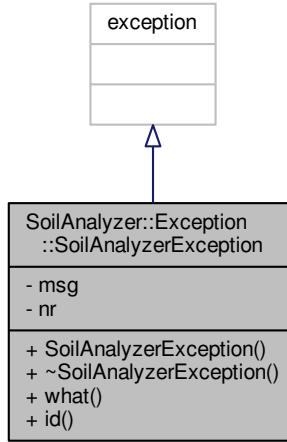
6.60 SoilAnalyzer::Exception::SoilAnalyzerException Class Reference

```
#include <soilanalyzerexception.h>
```

Inheritance diagram for SoilAnalyzer::Exception::SoilAnalyzerException:



Collaboration diagram for SoilAnalyzer::Exception::SoilAnalyzerException:



Public Member Functions

- `SoilAnalyzerException (std::string m=EXCEPTION_PARTICLE_NOT_ANALYZED, int n=EXCEPTION_PARTICLE_NOT_ANALYZED_NR)`
- `~SoilAnalyzerException () _GLIBCXX_USE_NOEXCEPT`
- `const char * what () const _GLIBCXX_USE_NOEXCEPT`
- `const int * id () const _GLIBCXX_USE_NOEXCEPT`

Private Attributes

- `std::string msg`
- `int nr`

6.60.1 Detailed Description

Definition at line 20 of file [soilanalyzerexception.h](#).

6.60.2 Constructor & Destructor Documentation

6.60.2.1 `SoilAnalyzer::Exception::SoilAnalyzerException::SoilAnalyzerException (std::string m = EXCEPTION_PARTICLE_NOT_ANALYZED, int n = EXCEPTION_PARTICLE_NOT_ANALYZED_NR) [inline]`

Definition at line 22 of file [soilanalyzerexception.h](#).

6.60.2.2 `SoilAnalyzer::Exception::SoilAnalyzerException::~SoilAnalyzerException () [inline]`

Definition at line 24 of file [soilanalyzerexception.h](#).

6.60.3 Member Function Documentation

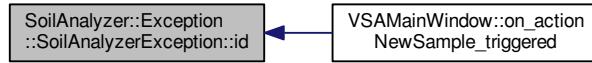
6.60.3.1 `const int* SoilAnalyzer::Exception::SoilAnalyzerException::id () const [inline]`

Definition at line 26 of file [soilanalyzerexception.h](#).

References [nr](#).

Referenced by [VSAMainWindow::on_actionNewSample_triggered\(\)](#).

Here is the caller graph for this function:



6.60.3.2 `const char* SoilAnalyzer::Exception::SoilAnalyzerException::what() const` [inline]

Definition at line 25 of file [soilanalyzerexception.h](#).

References [msg](#).

6.60.4 Member Data Documentation

6.60.4.1 `std::string SoilAnalyzer::Exception::SoilAnalyzerException::msg` [private]

Definition at line 29 of file [soilanalyzerexception.h](#).

Referenced by [what\(\)](#).

6.60.4.2 `int SoilAnalyzer::Exception::SoilAnalyzerException::nr` [private]

Definition at line 30 of file [soilanalyzerexception.h](#).

Referenced by [id\(\)](#).

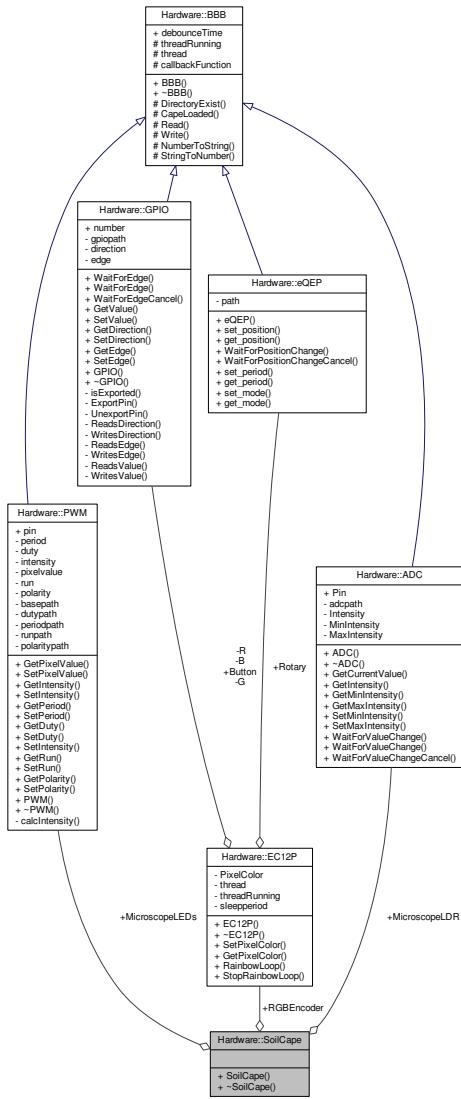
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzerexception.h](#)

6.61 Hardware::SoilCape Class Reference

```
#include <SoilCape.h>
```

Collaboration diagram for Hardware::SoilCape:



Public Member Functions

- [SoilCape \(\)](#)
- [~SoilCape \(\)](#)

Public Attributes

- [EC12P RGBEncoder](#)
- [PWM MicroscopeLEDs \(PWM::P9_14\)](#)
- [ADC MicroscopeLDR \(ADC::ADC0\)](#)

6.61.1 Detailed Description

Definition at line 16 of file [SoilCape.h](#).

6.61.2 Constructor & Destructor Documentation

6.61.2.1 Hardware::SoilCape::SoilCape ()

Definition at line 11 of file [SoilCape.cpp](#).

6.61.2.2 Hardware::SoilCape::~SoilCape ()

Definition at line 13 of file [SoilCape.cpp](#).

6.61.3 Member Data Documentation

6.61.3.1 ADC Hardware::SoilCape::MicroscopeLDR {ADC::ADC0}

Definition at line 20 of file [SoilCape.h](#).

6.61.3.2 PWM Hardware::SoilCape::MicroscopeLEDs {PWM::P9_14}

Definition at line 19 of file [SoilCape.h](#).

6.61.3.3 EC12P Hardware::SoilCape::RGBEncoder

Definition at line 18 of file [SoilCape.h](#).

The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.cpp](#)

6.62 SoilAnalyzer::SoilSettings Class Reference

The [SoilSettings](#) class.

```
#include <soilsettings.h>
```

Collaboration diagram for SoilAnalyzer::SoilSettings:

SoilAnalyzer::SoilSettings	
+ useAdaptiveContrast	
+ adaptContrastKernelSize	
+ adaptContrastKernelFactor	
+ useBlur	
+ blurKernelSize	
+ typeOfObjectsSegmented	
+ ignorePartialBorderParticles	
+ fillHoles	
+ sigmaFactor	
+ thresholdOffsetValue	
and 36 more...	
+ SoilSettings()	
+ SaveSettings()	
+ LoadSettings()	
- serialize()	

Public Member Functions

- [SoilSettings \(\)](#)
- void [SaveSettings](#) (std::string filename)

SaveSettings a function to save the settings to disk.

- void **LoadSettings** (std::string filename)
LoadSettings a function to load the settings from disk.

Public Attributes

- bool **useAdaptiveContrast**
- uint32_t **adaptContrastKernelSize**
- float **adaptContrastKernelFactor** = 1.
- bool **useBlur** = false
- uint32_t **blurKernelSize** = 5
- Vision::Segment::TypeOfObjects **typeOfObjectsSegmented**
- bool **ignorePartialBorderParticles**
- bool **fillHoles** = true
- float **sigmaFactor** = 2
- int **thresholdOffsetValue** = 0
- Vision::MorphologicalFilter::FilterType **morphFilterType**
- uint32_t **filterMaskSize** = 5
- uint32_t **HDRframes**
- float **lightLevel** = 0.5
- bool **enclnv** = false
- bool **enableRainbow**
- bool **useBacklightProjection** = true
- bool **useHDR** = false
- std::string **defaultWebcam** = "USB Microscope"
- int **Brightness_front** = 0
- int **Brightness_proj** = -10
- int **Contrast_front** = 36
- int **Contrast_proj** = 36
- int **Saturation_front** = 64
- int **Saturation_proj** = 0
- int **Hue_front** = 0
- int **Hue_proj** = -40
- int **Gamma_front** = 100
- int **Gamma_proj** = 200
- int **PowerLineFrequency_front**
- int **PowerLineFrequency_proj**
- int **Sharpness_front** = 12
- int **Sharpness_proj** = 25
- int **BackLightCompensation_front**
- int **BackLightCompensation_proj**
- std::string **NNlocation** = "NeuralNet/Default.NN"
- bool **useCUDA** = false
- int **selectedResolution** = 0
- std::string **SampleFolder** = "~/Samples"
- std::string **SettingsFolder** = "Settings"
- std::string **NNFolder** = "NeuralNet"
- std::string **StandardSentTo** = "j.spijker@ihcmerwede.com"
- std::string **StandardPrinter** = "PDF printer"
- uint32_t **StandardNumberOfShots** = 10
- bool **PredictTheShape** = true
- bool **Revolution** = true

Private Member Functions

- template<class Archive >
void **serialize** (Archive &ar, const unsigned int version)

Friends

- class **boost::serialization::access**

6.62.1 Detailed Description

The `SoilSettings` class.

A class with which the used settings can easily be transferred to setup the `Sample` class in one go. This class is also used in the GUI and has a possibility to be saved to disk as a serialized object

Definition at line 24 of file `soilsettings.h`.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 `SoilAnalyzer::SoilSettings::SoilSettings()`

Definition at line 11 of file `soilsettings.cpp`.

6.62.3 Member Function Documentation

6.62.3.1 `void SoilAnalyzer::SoilSettings::LoadSettings(std::string filename)`

LoadSettings a function to load the settings from disk.

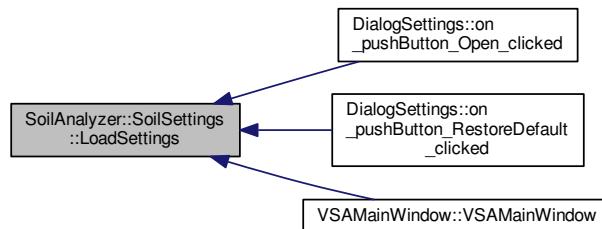
Parameters

<code>filename</code>	a string with the filename
-----------------------	----------------------------

Definition at line 13 of file `soilsettings.cpp`.

Referenced by `DialogSettings::on_pushButton_Open_clicked()`, `DialogSettings::on_pushButton_RestoreDefault_clicked()`, and `VSAMainWindow::VSAMainWindow()`.

Here is the caller graph for this function:



6.62.3.2 `void SoilAnalyzer::SoilSettings::SaveSettings(std::string filename)`

SaveSettings a function to save the settings to disk.

Parameters

<code>filename</code>	a string with the filename
-----------------------	----------------------------

Definition at line 19 of file `soilsettings.cpp`.

Referenced by `DialogSettings::on_pushButton_Save_clicked()`.

Here is the caller graph for this function:



6.62.3.3 `template<class Archive> void SoilAnalyzer::SoilSettings::serialize (Archive & ar, const unsigned int version)` [inline], [private]

Definition at line 109 of file [soilsettings.h](#).

6.62.4 Friends And Related Function Documentation

6.62.4.1 `friend class boost::serialization::access` [friend]

Definition at line 107 of file [soilsettings.h](#).

6.62.5 Member Data Documentation

6.62.5.1 `float SoilAnalyzer::SoilSettings::adaptContrastKernelFactor = 1.`

the factor with which to multiply the effect of the adaptive contrast stretch

Definition at line 44 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

6.62.5.2 `uint32_t SoilAnalyzer::SoilSettings::adaptContrastKernelSize`

Initial value:

= 9

The size of the adaptive contrast kernelsize

Definition at line 42 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

6.62.5.3 `int SoilAnalyzer::SoilSettings::BackLightCompensation_front`

Initial value:

= 1

cam backlight compensation setting front light

Definition at line 91 of file [soilsettings.h](#).

6.62.5.4 `int SoilAnalyzer::SoilSettings::BackLightCompensation_proj`

Initial value:

= 1

cam backlight compensation setting projected light

Definition at line 93 of file [soilsettings.h](#).

6.62.5.5 `uint32_t SoilAnalyzer::SoilSettings::blurKernelSize = 5`

the median blurkernel

Definition at line 49 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

6.62.5.6 `int SoilAnalyzer::SoilSettings::Brightness_front = 0`

cam brightness setting front light

Definition at line 75 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_horizontalSlider_BrightFront_valueChanged\(\)](#).

6.62.5.7 `int SoilAnalyzer::SoilSettings::Brightness_proj = -10`

cam brightness setting projected light

Definition at line 76 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_horizontalSlider_BrightProj_valueChanged()`.

6.62.5.8 int `SoilAnalyzer::SoilSettings::Contrast_front` = 36

cam contrast setting front light

Definition at line 77 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_horizontalSlider_ContrastFront_valueChanged()`.

6.62.5.9 int `SoilAnalyzer::SoilSettings::Contrast_proj` = 36

cam contrast setting projected light

Definition at line 78 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_horizontalSlider_ContrastProj_valueChanged()`.

6.62.5.10 std::string `SoilAnalyzer::SoilSettings::defaultWebcam` = "USB Microscope"

The defaultWebcam string

Definition at line 74 of file `soilsettings.h`.

Referenced by `DialogSettings::on_comboBox_Microscopes_currentIndexChanged()`, and `VSAMainWindow::VSAMainWindow()`.

6.62.5.11 bool `SoilAnalyzer::SoilSettings::enableRainbow`

Initial value:

```
=      true
```

run a rainbow loop on the RGB encoder during analysis

Definition at line 70 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_checkBox_useRainbow_clicked()`.

6.62.5.12 bool `SoilAnalyzer::SoilSettings::enclInv` = false

invert the values gained form the encoder

Definition at line 69 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_checkBox_InvertEncoder_clicked()`.

6.62.5.13 bool `SoilAnalyzer::SoilSettings::fillHoles` = true

should the holes be filled

Definition at line 55 of file `soilsettings.h`.

Referenced by `SoilAnalyzer::Analyzer::CalcMaxProgress()`, `DialogSettings::DialogSettings()`, `SoilAnalyzer::Analyzer::GetBW()`, and `DialogSettings::on_cb_fillHoles_3_clicked()`.

6.62.5.14 uint32_t `SoilAnalyzer::SoilSettings::filterMaskSize` = 5

the filter mask

Definition at line 64 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, `SoilAnalyzer::Analyzer::GetBW()`, and `DialogSettings::on_sb_morphMask_3_editingFinished()`.

6.62.5.15 int `SoilAnalyzer::SoilSettings::Gamma_front` = 100

cam gamma setting front light

Definition at line 83 of file `soilsettings.h`.

6.62.5.16 int `SoilAnalyzer::SoilSettings::Gamma_proj` = 200

cam gamma setting projected light

Definition at line 84 of file `soilsettings.h`.

6.62.5.17 `uint32_t` `SoilAnalyzer::SoilSettings::HDRframes`

Initial value:

= 5

The number of frames which should be used for the HDR image

Definition at line 66 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, `DialogSettings::on_spinBox_NoFrames_editingFinished()`, and `VSAMainWindow::TakeSnapshots()`.

6.62.5.18 `int` `SoilAnalyzer::SoilSettings::Hue_front = 0`

cam hue setting front light

Definition at line 81 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_horizontalSlider_HueFront_valueChanged()`.

6.62.5.19 `int` `SoilAnalyzer::SoilSettings::Hue_proj = -40`

cam hue setting projected light

Definition at line 82 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, and `DialogSettings::on_horizontalSlider_HueProj_valueChanged()`.

6.62.5.20 `bool` `SoilAnalyzer::SoilSettings::ignorePartialBorderParticles`

Initial value:

= `true`

Indication of partial border particles should be used

Definition at line 53 of file `soilsettings.h`.

Referenced by `SoilAnalyzer::Analyzer::CalcMaxProgress()`, `DialogSettings::DialogSettings()`, `SoilAnalyzer::Analyzer::GetBW()`, and `DialogSettings::on_cb_ignoreBorder_3_clicked()`.

6.62.5.21 `float` `SoilAnalyzer::SoilSettings::lightLevel = 0.5`

The light level of the environmental case

Definition at line 68 of file `soilsettings.h`.

Referenced by `DialogSettings::on_doubleSpinBox_LightLevel_editingFinished()`.

6.62.5.22 `Vision::MorphologicalFilter::FilterType` `SoilAnalyzer::SoilSettings::morphFilterType`

Initial value:

= `Vision::MorphologicalFilter::OPEN`

Indicating which type of morphological filter should be used

Definition at line 60 of file `soilsettings.h`.

Referenced by `SoilAnalyzer::Analyzer::CalcMaxProgress()`, `DialogSettings::DialogSettings()`, `SoilAnalyzer::Analyzer::GetBW()`, `DialogSettings::on_rb_useClose_3_clicked()`, `DialogSettings::on_rb_useDilate_3_clicked()`, `DialogSettings::on_rb_useErode_3_clicked()`, and `DialogSettings::on_rb_useOpen_3_clicked()`.

6.62.5.23 `std::string` `SoilAnalyzer::SoilSettings::NNFolder = "NeuralNet"`

Definition at line 100 of file `soilsettings.h`.

Referenced by `DialogSettings::DialogSettings()`, `DialogNN::on_pushButton_SaveNN_clicked()`, and `DialogSettings::on_pushButton_SelectNNFolder_clicked()`.

6.62.5.24 `std::string` `SoilAnalyzer::SoilSettings::NNlocation = "NeuralNet/Default.NN"`

Definition at line 95 of file `soilsettings.h`.

Referenced by `SoilAnalyzer::Analyzer::Analyzer()`, `DialogSettings::DialogSettings()`, and `DialogSettings::on_pushButton_SelectNN_clicked()`.

6.62.5.25 int SoilAnalyzer::SoilSettings::PowerLineFrequency_front

Initial value:

= 1

cam powerline freq setting front light

Definition at line 85 of file [soilsettings.h](#).

6.62.5.26 int SoilAnalyzer::SoilSettings::PowerLineFrequency_proj

Initial value:

= 1

cam powerline freq setting projected light

Definition at line 87 of file [soilsettings.h](#).

6.62.5.27 bool SoilAnalyzer::SoilSettings::PredictTheShape = true

Definition at line 104 of file [soilsettings.h](#).

Referenced by [SoilAnalyzer::Analyzer::Analyse\(\)](#), [DialogSettings::DialogSettings\(\)](#), [VSAMainWindow::on_actionAutomatic_Shape_Prediction_triggered\(\)](#), [DialogSettings::on_checkBox_PredictShape_clicked\(\)](#), and [VSAMainWindow::VSAMainWindow\(\)](#).

6.62.5.28 bool SoilAnalyzer::SoilSettings::Revolution = true

Definition at line 105 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_checkBox_revolt_clicked\(\)](#).

6.62.5.29 std::string SoilAnalyzer::SoilSettings::SampleFolder = " ~/Samples"

Definition at line 98 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [QReportGenerator::on_actionExport_to_PDF_triggered\(\)](#), [VSAMainWindow::on_actionLoadSample_triggered\(\)](#), [QReportGenerator::on_actionSave_triggered\(\)](#), [VSAMainWindow::on_actionSaveSample_triggered\(\)](#), [VSAMainWindow::on_compare_against\(\)](#), [DialogNN::on_pushButton_OpenNN_clicked\(\)](#), [DialogSettings::on_pushButton_selectSampleFolder_clicked\(\)](#), and [DialogNN::on_pushButton_SelectSamples_clicked\(\)](#).

6.62.5.30 int SoilAnalyzer::SoilSettings::Saturation_front = 64

cam saturation setting front light

Definition at line 79 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_horizontalSlider_SaturationFront_valueChanged\(\)](#).

6.62.5.31 int SoilAnalyzer::SoilSettings::Saturation_proj = 0

cam saturation setting projected light

Definition at line 80 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_horizontalSlider_SaturationProj_valueChanged\(\)](#).

6.62.5.32 int SoilAnalyzer::SoilSettings::selectedResolution = 0

Definition at line 97 of file [soilsettings.h](#).

Referenced by [DialogSettings::on_comboBox_Resolution_currentIndexChanged\(\)](#), and [VSAMainWindow::VSAMainWindow\(\)](#).

6.62.5.33 std::string SoilAnalyzer::SoilSettings::SettingsFolder = "Settings"

Definition at line 99 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_pushButton_SelectSettingFolder_clicked\(\)](#).

6.62.5.34 int SoilAnalyzer::SoilSettings::Sharpness_front = 12

cam sharpness setting front light

Definition at line 89 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_horizontalSlider_SharpnessFront_valueChanged\(\)](#).

6.62.5.35 int SoilAnalyzer::SoilSettings::Sharpness_proj = 25

cam sharpness setting projected light

Definition at line 90 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_horizontalSlider_SharpnessProj_valueChanged\(\)](#).

6.62.5.36 float SoilAnalyzer::SoilSettings::sigmaFactor = 2

The sigma factor or the bandwidth indicating which pixel intensity values count belong to an object

Definition at line 56 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [SoilAnalyzer::Analyzer::GetBW\(\)](#), and [DialogSettings::on_sb_sigmaFactor_3_editingFinished\(\)](#).

6.62.5.37 uint32_t SoilAnalyzer::SoilSettings::StandardNumberOfShots = 10

Definition at line 103 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [DialogSettings::on_spinBox_NoShots_editingFinished\(\)](#), and [VSAMainWindow::TakeSnapShots\(\)](#).

6.62.5.38 std::string SoilAnalyzer::SoilSettings::StandardPrinter = "PDF printer"

Definition at line 102 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

6.62.5.39 std::string SoilAnalyzer::SoilSettings::StandardSentTo = "j.spijker@ihcmerwede.com"

Definition at line 101 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#).

6.62.5.40 int SoilAnalyzer::SoilSettings::thresholdOffsetValue = 0

an tweaking offset value

Definition at line 58 of file [soilsettings.h](#).

Referenced by [SoilAnalyzer::Analyzer::GetBW\(\)](#).

6.62.5.41 Vision::Segment::TypeOfObjects SoilAnalyzer::SoilSettings::typeOfObjectsSegmented

Initial value:

= [Vision::Segment::Dark](#)

Which type of object should be segmented

Definition at line 51 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [SoilAnalyzer::Analyzer::GetBW\(\)](#), and [DialogSettings::on_rb_useDark_3_toggled\(\)](#).

6.62.5.42 bool SoilAnalyzer::SoilSettings::useAdaptiveContrast

Initial value:

= [false](#)

Should adaptive contrast stretch be used default is true

Definition at line 40 of file [soilsettings.h](#).

Referenced by [SoilAnalyzer::Analyzer::CalcMaxProgress\(\)](#), [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_cb_use_adaptContrast_3_clicked\(\)](#).

6.62.5.43 bool SoilAnalyzer::SoilSettings::useBacklightProjection = true

use Projection

Definition at line 72 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [DialogSettings::on_checkBox_Backlight_clicked\(\)](#), and [VSAMainWindow::TakeSnapShots\(\)](#).

6.62.5.44 bool SoilAnalyzer::SoilSettings::useBlur = false

Should the mediaan blur be used during analysis

Definition at line 48 of file [soilsettings.h](#).

Referenced by [SoilAnalyzer::Analyzer::CalcMaxProgress\(\)](#), [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_cb_useBlur_3_clicked\(\)](#).

6.62.5.45 bool SoilAnalyzer::SoilSettings::useCUDA = false

CUDA enabled

Definition at line 96 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), and [DialogSettings::on_checkBox_useCUDA_clicked\(\)](#).

6.62.5.46 bool SoilAnalyzer::SoilSettings::useHDR = false

use HDR

Definition at line 73 of file [soilsettings.h](#).

Referenced by [DialogSettings::DialogSettings\(\)](#), [DialogSettings::on_checkBox_useHDR_clicked\(\)](#), and [VSAMainWindow::TakeSnapShots\(\)](#).

The documentation for this class was generated from the following files:

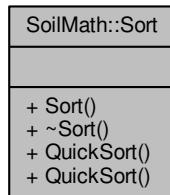
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilsettings.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilsettings.cpp](#)

6.63 SoilMath::Sort Class Reference

The [Sort](#) template class.

```
#include <Sort.h>
```

Collaboration diagram for SoilMath::Sort:



Public Member Functions

- [Sort \(\)](#)
- [~Sort \(\)](#)

Static Public Member Functions

- template<typename T >
static void [QuickSort](#) (T *arr, int i)

QuickSort a static sort a Type T array with i values.

- template<typename T >
static void [QuickSort](#) (T *arr, T *key, int i)

QuickSort a static sort a Type T array with i values where the key are also changed accordingly.

6.63.1 Detailed Description

The [Sort](#) template class.

Definition at line [15](#) of file [Sort.h](#).

6.63.2 Constructor & Destructor Documentation

6.63.2.1 [SoilMath::Sort::Sort\(\)](#) [inline]

Definition at line [17](#) of file [Sort.h](#).

6.63.2.2 [SoilMath::Sort::~Sort\(\)](#) [inline]

Definition at line [18](#) of file [Sort.h](#).

6.63.3 Member Function Documentation

6.63.3.1 [template<typename T> static void SoilMath::Sort::QuickSort\(T * arr, int i \)](#) [inline], [static]

QuickSort a static sort a Type T array with i values.

Usage: `QuickSort<type>(*type, i)`

Parameters

<i>arr</i>	an array of Type T
<i>i</i>	the number of elements

Definition at line [26](#) of file [Sort.h](#).

6.63.3.2 [template<typename T> static void SoilMath::Sort::QuickSort\(T * arr, T * key, int i \)](#) [inline], [static]

QuickSort a static sort a Type T array with i values where the key are also changed accordingly.

Usage: `QuickSort<type>(*type, *type, i)`

Parameters

<i>arr</i>	an array of Type T
<i>key</i>	an array of 0..i-1 representing the index
<i>i</i>	the number of elements

Definition at line [58](#) of file [Sort.h](#).

The documentation for this class was generated from the following file:

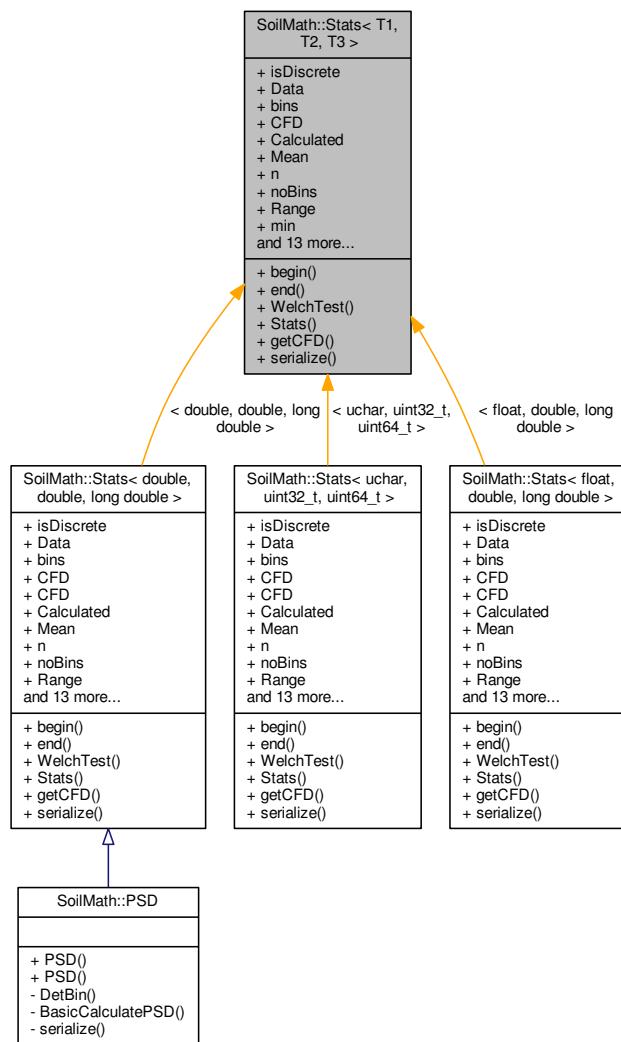
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Sort.h](#)

6.64 [SoilMath::Stats< T1, T2, T3 >](#) Class Template Reference

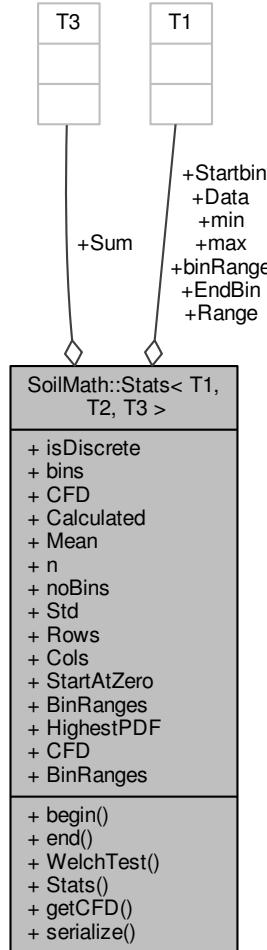
[Stats](#) class.

```
#include <Stats.h>
```

Inheritance diagram for SoilMath::Stats< T1, T2, T3 >:



Collaboration diagram for SoilMath::Stats< T1, T2, T3 >:



Public Member Functions

- `uint32_t * begin ()`
- `uint32_t * end ()`
- `bool WelchTest (SoilMath::Stats< T1, T2, T3 > &statComp)`
 - WelchTest Compare the sample using the Welch's Test.*
- `Stats (const Stats &rhs)`
 - Stats Constructor.*
- `void getCFD ()`
 - getCFD get the CFD matrix;*
- template<class Archive >
`void serialize (Archive &ar, const unsigned int version)`
 - serialize the object*

Public Attributes

- `bool isDiscrete = true`

- `T1 * Data = nullptr`
- `uint32_t * bins = nullptr`
- `double * CFD = nullptr`
- `bool Calculated = false`
- `float Mean = 0.0`
- `uint32_t n = 0`
- `uint32_t noBins = 0`
- `T1 Range = 0`
- `T1 min = 0`
- `T1 max = 0`
- `T1 Startbin = 0`
- `T1 EndBin = 0`
- `T1 binRange = 0`
- `float Std = 0.0`
- `T3 Sum = 0`
- `uint16_t Rows = 0`
- `uint16_t Cols = 0`
- `bool StartAtZero = true`
- `double * BinRanges = nullptr`
- `double HighestPDF = 0.`
- `CFD {new double[rhs.noBins]{}}`
- `BinRanges`

Friends

- class `boost::serialization::access`

6.64.1 Detailed Description

`template<typename T1, typename T2, typename T3> class SoilMath::Stats< T1, T2, T3 >`

`Stats` class.

Usage `Stats<type1, type2, type3>Stats()` type 1, 2 and 3 shoudl be of the same value and concecutive in size

Definition at line [39](#) of file `Stats.h`.

6.64.2 Constructor & Destructor Documentation

`6.64.2.1 template<typename T1, typename T2, typename T3> SoilMath::Stats< T1, T2, T3 >::Stats (const Stats< T1, T2, T3 > & rhs) [inline]`

`Stats` Constructor.

Parameters

<code>rhs</code>	Right hand side
------------------	-----------------

Definition at line [112](#) of file `Stats.h`.

6.64.3 Member Function Documentation

`6.64.3.1 template<typename T1, typename T2, typename T3> uint32_t* SoilMath::Stats< T1, T2, T3 >::begin () [inline]`

pointer to the first bin

Definition at line [65](#) of file `Stats.h`.

`6.64.3.2 template<typename T1, typename T2, typename T3> uint32_t* SoilMath::Stats< T1, T2, T3 >::end () [inline]`

pointer to the last + 1 bin

Definition at line [66](#) of file `Stats.h`.

`6.64.3.3 template<typename T1, typename T2, typename T3> void SoilMath::Stats< T1, T2, T3 >::getCFD () [inline]`

getCFD get the CFD matrix;

Definition at line [629](#) of file `Stats.h`.

```
6.64.3.4 template<typename T1, typename T2, typename T3> template<class Archive> void SoilMath::Stats< T1, T2, T3 >::serialize ( Archive & ar,  
const unsigned int version ) [inline]
```

serialize the object

Parameters

<i>ar</i>	argument
<i>version</i>	

Definition at line 651 of file [Stats.h](#).

6.64.3.5 template<typename T1, typename T2, typename T3> bool **SoilMath::Stats< T1, T2, T3 >::WelchTest** (**SoilMath::Stats< T1, T2, T3 > & statComp**) [inline]

WelchTest Compare the sample using the Welch's Test.

(source: 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)

Parameters

<i>statComp</i>	Statics Results of which it should be tested against
-----------------	--

Returns

Definition at line 75 of file [Stats.h](#).

6.64.4 Friends And Related Function Documentation

6.64.4.1 template<typename T1, typename T2, typename T3> friend class **boost::serialization::access** [friend]

Serialization class

Definition at line 643 of file [Stats.h](#).

6.64.5 Member Data Documentation

6.64.5.1 template<typename T1, typename T2, typename T3> T1 **SoilMath::Stats< T1, T2, T3 >::binRange** = 0

the range of a single bin

Definition at line 55 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.2 template<typename T1, typename T2, typename T3> double* **SoilMath::Stats< T1, T2, T3 >::BinRanges** = nullptr

Definition at line 62 of file [Stats.h](#).

6.64.5.3 template<typename T1, typename T2, typename T3> **SoilMath::Stats< T1, T2, T3 >::BinRanges**

data end counter used with mask

Definition at line 114 of file [Stats.h](#).

6.64.5.4 template<typename T1, typename T2, typename T3> uint32_t* **SoilMath::Stats< T1, T2, T3 >::bins** = nullptr

the histogram

Definition at line 44 of file [Stats.h](#).

Referenced by [Vision::Segment::GetBlobList\(\)](#), [Vision::Segment::GetThresholdLevel\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), [VSAMainWindow::setAngularityHistogram\(\)](#), and [VSAMainWindow::setRoundnessHistogram\(\)](#).

6.64.5.5 template<typename T1, typename T2, typename T3> bool **SoilMath::Stats< T1, T2, T3 >::Calculated** = false

indication if the data has been calculated

Definition at line 46 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.6 template<typename T1, typename T2, typename T3> double* **SoilMath::Stats< T1, T2, T3 >::CFD** = nullptr

the CFD

Definition at line 45 of file [Stats.h](#).

Referenced by [QReportGenerator::QReportGenerator\(\)](#), and [VSAMainWindow::SetPSDgraph\(\)](#).

6.64.5.7 template<typename T1, typename T2, typename T3> **SoilMath::Stats< T1, T2, T3 >::CFD** {new double[rhs.noBins]{}}

Definition at line 113 of file [Stats.h](#).

6.64.5.8 template<typename T1, typename T2, typename T3> **uint16_t SoilMath::Stats< T1, T2, T3 >::Cols** = 0

number of cols from the data matrix

Definition at line 59 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.9 template<typename T1, typename T2, typename T3> **T1* SoilMath::Stats< T1, T2, T3 >::Data** = nullptr

Pointer the data

Definition at line 43 of file [Stats.h](#).

Referenced by [VSAMainWindow::on_actionLoadSample_triggered\(\)](#).

6.64.5.10 template<typename T1, typename T2, typename T3> **T1 SoilMath::Stats< T1, T2, T3 >::EndBin** = 0

End bin value

Definition at line 54 of file [Stats.h](#).

Referenced by [Vision::Segment::GetBlobList\(\)](#), and [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.11 template<typename T1, typename T2, typename T3> **double SoilMath::Stats< T1, T2, T3 >::HighestPDF** = 0.

Definition at line 63 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#), [VSAMainWindow::setAngularityHistogram\(\)](#), and [VSAMainWindow::setRoundnessHistogram\(\)](#).

6.64.5.12 template<typename T1, typename T2, typename T3> **bool SoilMath::Stats< T1, T2, T3 >::isDiscrete** = true

indicates if the data is discrete or real

Definition at line 41 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.13 template<typename T1, typename T2, typename T3> **T1 SoilMath::Stats< T1, T2, T3 >::max** = 0

maximum value

Definition at line 52 of file [Stats.h](#).

Referenced by [Vision::Enhance::HistogramEqualization\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), and [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.14 template<typename T1, typename T2, typename T3> **float SoilMath::Stats< T1, T2, T3 >::Mean** = 0.0

the mean value of the data

Definition at line 47 of file [Stats.h](#).

Referenced by [SoilAnalyzer::Particle::getMeanLab\(\)](#), [SoilAnalyzer::Particle::GetMeanRI\(\)](#), [Vision::Segment::GetThresholdLevel\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), [SoilMath::Stats< float, double, long double >::serialize\(\)](#), [VSAMainWindow::setAngularityHistogram\(\)](#), [VSAMainWindow::setRoundnessHistogram\(\)](#), and [SoilMath::Stats< float, double, long double >::WelchTest\(\)](#).

6.64.5.15 template<typename T1, typename T2, typename T3> **T1 SoilMath::Stats< T1, T2, T3 >::min** = 0

minimum value

Definition at line 51 of file [Stats.h](#).

Referenced by [Vision::Enhance::HistogramEqualization\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), and [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.16 template<typename T1, typename T2, typename T3> **uint32_t SoilMath::Stats< T1, T2, T3 >::n** = 0

number of data points

Definition at line 48 of file [Stats.h](#).

Referenced by [QReportGenerator::QReportGenerator\(\)](#), [SoilMath::Stats< float, double, long double >::serialize\(\)](#), and [SoilMath::Stats< float, double, long double >::WelchTest\(\)](#).

6.64.5.17 `template<typename T1, typename T2, typename T3> uint32_t SoilMath::Stats< T1, T2, T3 >::noBins = 0`

number of bins

Definition at line 49 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::end\(\)](#), [SoilMath::Stats< float, double, long double >::getCFD\(\)](#), and [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.18 `template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::Range = 0`

range of the data

Definition at line 50 of file [Stats.h](#).

Referenced by [QReportGenerator::QReportGenerator\(\)](#), and [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.19 `template<typename T1, typename T2, typename T3> uint16_t SoilMath::Stats< T1, T2, T3 >::Rows = 0`

number of rows from the data matrix

Definition at line 58 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.20 `template<typename T1, typename T2, typename T3> bool SoilMath::Stats< T1, T2, T3 >::StartAtZero = true`

indication of the minimum value starts at zero or could be less

Definition at line 60 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.21 `template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::Startbin = 0`

First bin value

Definition at line 53 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

6.64.5.22 `template<typename T1, typename T2, typename T3> float SoilMath::Stats< T1, T2, T3 >::Std = 0.0`

standard deviation

Definition at line 56 of file [Stats.h](#).

Referenced by [Vision::Segment::GetThresholdLevel\(\)](#), [QReportGenerator::QReportGenerator\(\)](#), [SoilMath::Stats< float, double, long double >::serialize\(\)](#), and [SoilMath::Stats< float, double, long double >::WelchTest\(\)](#).

6.64.5.23 `template<typename T1, typename T2, typename T3> T3 SoilMath::Stats< T1, T2, T3 >::Sum = 0`

total sum of all the data values

Definition at line 57 of file [Stats.h](#).

Referenced by [SoilMath::Stats< float, double, long double >::serialize\(\)](#).

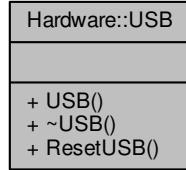
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Stats.h](#)

6.65 Hardware::USB Class Reference

```
#include <USB.h>
```

Collaboration diagram for Hardware::USB:



Public Member Functions

- [USB \(\)](#)
- [~USB \(\)](#)
- void [ResetUSB \(\)](#)

6.65.1 Detailed Description

Definition at line [19](#) of file [USB.h](#).

6.65.2 Constructor & Destructor Documentation

6.65.2.1 Hardware::USB::USB()

Definition at line [11](#) of file [USB.cpp](#).

6.65.2.2 Hardware::USB::~USB()

Definition at line [13](#) of file [USB.cpp](#).

6.65.3 Member Function Documentation

6.65.3.1 void Hardware::USB::ResetUSB()

Definition at line [15](#) of file [USB.cpp](#).

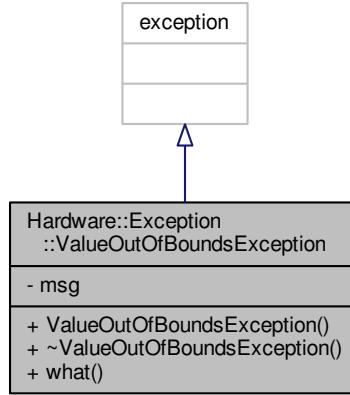
The documentation for this class was generated from the following files:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.h](#)
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.cpp](#)

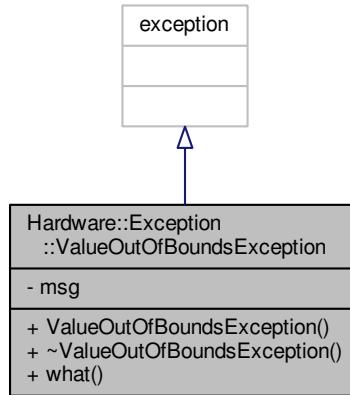
6.66 Hardware::Exception::ValueOutOfBoundsException Class Reference

```
#include <ValueOutOfBoundsException.h>
```

Inheritance diagram for Hardware::Exception::ValueOutOfBoundsException:



Collaboration diagram for Hardware::Exception::ValueOutOfBoundsException:



Public Member Functions

- `ValueOutOfBoundsException` (string m="Value out of bounds!")
- `~ValueOutOfBoundsException` () _GLIBCXX_USE_NOEXCEPT
- const char * `what` () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string `msg`

6.66.1 Detailed Description

Definition at line 17 of file [ValueOutOfBoundsException.h](#).

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `Hardware::Exception::ValueOutOfBoundsException::ValueOutOfBoundsException (string m = "Value out of bounds!")` [inline]

Definition at line 19 of file [ValueOutOfBoundsException.h](#).

6.66.2.2 `Hardware::Exception::ValueOutOfBoundsException::~ValueOutOfBoundsException ()` [inline]

Definition at line 20 of file [ValueOutOfBoundsException.h](#).

6.66.3 Member Function Documentation

6.66.3.1 `const char* Hardware::Exception::ValueOutOfBoundsException::what () const` [inline]

Definition at line 21 of file [ValueOutOfBoundsException.h](#).

6.66.4 Member Data Documentation

6.66.4.1 `string Hardware::Exception::ValueOutOfBoundsException::msg` [private]

Definition at line 21 of file [ValueOutOfBoundsException.h](#).

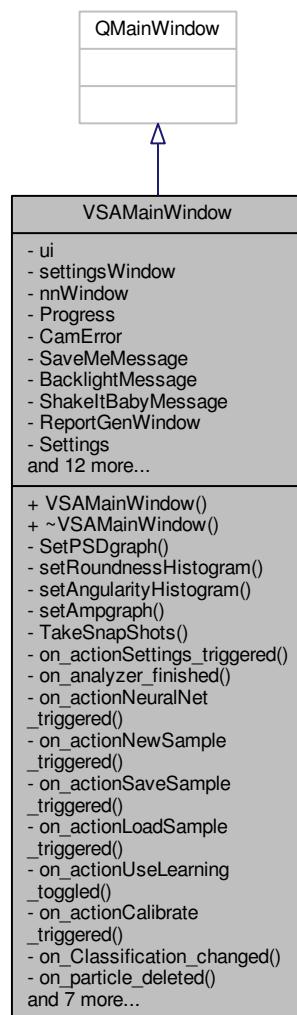
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ValueOutOfBoundsException.h](#)

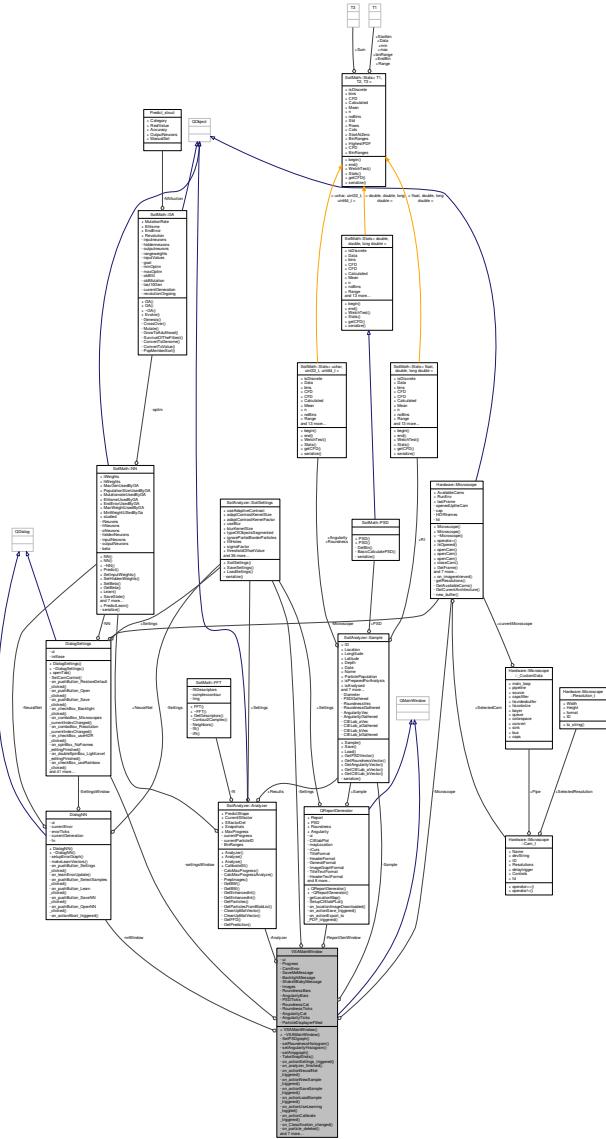
6.67 VSAMainWindow Class Reference

```
#include <vsamainwindow.h>
```

Inheritance diagram for VSAMainWindow:



Collaboration diagram for VSAMainWindow:



Public Member Functions

- **VSAMainWindow** (QWidget *parent=0)
 - **~VSAMainWindow** ()

Private Slots

- void **on_actionSettings_triggered** ()
 - void **on_analyzer_finished** ()
 - void **on_actionNeuralNet_triggered** ()
 - void **on_actionNewSample_triggered** ()
 - void **on_actionSaveSample_triggered** ()
 - void **on_actionLoadSample_triggered** ()
 - void **on_actionUseLearning_toggled** (bool arg1)
 - void **on_actionCalibrate_triggered** ()
 - void **on_Classification_changed** (int newValue)

- void `on_particle_deleted()`
- void `on_actionAutomatic_Shape_Pediction_triggered(bool checked)`
- void `on_reset_graph(QMouseEvent *e)`
- void `on_actionReport_Generator_triggered()`
- void `on_particleChanged(int newPart)`
- void `on_PSD_contextMenuRequest(QPoint point)`
- void `on_compare_against()`
- void `on_restore_PSD()`

Private Member Functions

- void `SetPSDgraph()`
- void `setRoundnessHistogram()`
- void `setAngularityHistogram()`
- void `setAmpgraph()`
- void `TakeSnapShots()`

Private Attributes

- `Ui::VSAMainWindow * ui`
- `DialogSettings * settingsWindow = nullptr`
- `DialogNN * nnWindow = nullptr`
- `QProgressBar * Progress`
- `QErrorMessage * CamError = nullptr`
- `QMessageBox * SaveMeMessage = nullptr`
- `QMessageBox * BacklightMessage = nullptr`
- `QMessageBox * ShakeItBabyMessage = nullptr`
- `QReportGenerator * ReportGenWindow = nullptr`
- `SoilAnalyzer::SoilSettings * Settings = nullptr`
- `Hardware::Microscope * Microscope = nullptr`
- `SoilAnalyzer::Sample * Sample = nullptr`
- `SoilAnalyzer::Analyzer * Analyzer = nullptr`
- `SoilAnalyzer::Analyzer::Images_t * Images = nullptr`
- `QCPBars * RoundnessBars = nullptr`
- `QCPBars * AngularityBars = nullptr`
- `std::vector<double> PSDTicks`
- `QVector<QString> RoundnessCat = {"High", "Medium", "Low"}`
- `std::vector<double> RoundnessTicks = {1, 2, 3}`
- `QVector<QString> AngularityCat`
- `std::vector<double> AngularityTicks = {1, 2, 3, 4, 5, 6}`
- `bool ParticleDisplayerFilled = false`

6.67.1 Detailed Description

Definition at line 27 of file `vsamainwindow.h`.

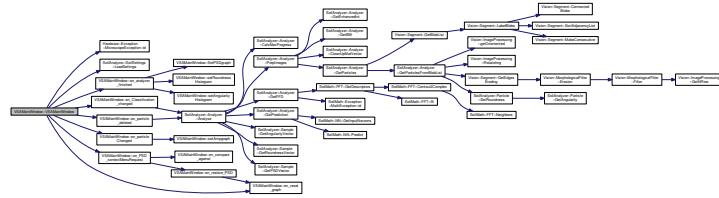
6.67.2 Constructor & Destructor Documentation

6.67.2.1 VSAMainWindow::VSAMainWindow (QWidget * parent = 0) [explicit]

Definition at line 4 of file `vsamainwindow.cpp`.

References `Analyzer`, `AngularityBars`, `AngularityCat`, `AngularityTicks`, `BacklightMessage`, `CamError`, `SoilAnalyzer::SoilSettings::defaultWebcam`, `EXCEPTION_NOCAMS_NR`, `EXCEPTION_OPENCAM_NR`, `Hardware::Exception::MicroscopeException::id()`, `Images`, `SoilAnalyzer::SoilSettings::LoadSettings()`, `SoilAnalyzer::Analyzer::MaxProgress`, `SoilAnalyzer::Analyzer::NeuralNet`, `nnWindow`, `on_analyzer_finished()`, `on_classification_changed()`, `on_particle_deleted()`, `on_particleChanged()`, `on_PSD_contextMenuRequest()`, `on_reset_graph()`, `SoilAnalyzer::SoilSettings::PredictTheShape`, `Progress`, `PSDTicks`, `RoundnessBars`, `RoundnessCat`, `RoundnessTicks`, `Sample`, `SaveMeMessage`, `SoilAnalyzer::SoilSettings::selectedResolution`, `Settings`, `settingsWindow`, `ShakeItBabyMessage`, and `ui`.

Here is the call graph for this function:



6.67.2.2 VSAMainWindow::~VSAMainWindow ()

Definition at line 254 of file [vsmainwindow.cpp](#).

References [Analyzer](#), [BacklightMessage](#), [CamError](#), [Images](#), [Microscope](#), [nnWindow](#), [Sample](#), [SaveMeMessage](#), [Settings](#), [settingsWindow](#), [ShakeltBabyMessage](#), and [ui](#).

6.67.3 Member Function Documentation

6.67.3.1 void VSAMainWindow::on_actionAutomatic_Shape_Prediction_triggered (bool checked) [private], [slot]

Definition at line 541 of file [vsmainwindow.cpp](#).

References [SoilAnalyzer::SoilSettings::PredictTheShape](#), and [Settings](#).

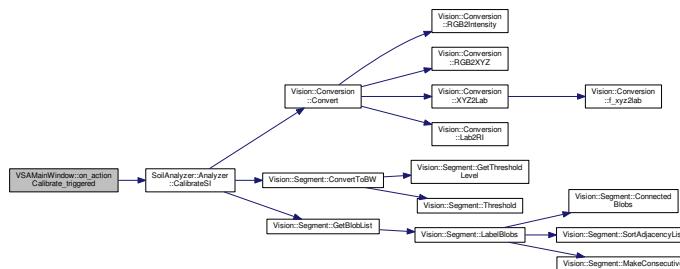
6.67.3.2 void VSAMainWindow::on_actionCalibrate_triggered () [private], [slot]

Definition at line 516 of file [vsmainwindow.cpp](#).

References [Analyzer](#), and [SoilAnalyzer::Analyzer::CalibrateSI\(\)](#).

Referenced by [TakeSnapShots\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

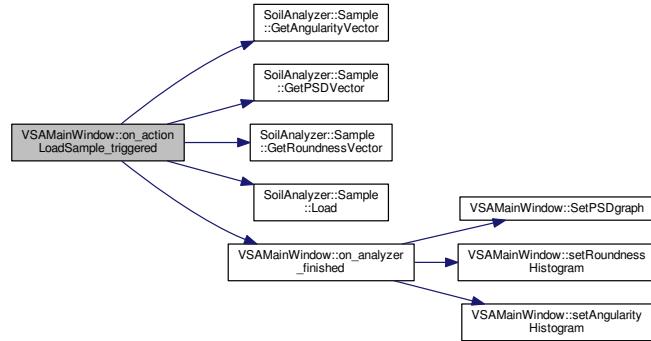


6.67.3.3 void VSAMainWindow::on_actionLoadSample_triggered () [private], [slot]

Definition at line 475 of file [vsmainwindow.cpp](#).

References [Analyzer](#), [SoilAnalyzer::Sample::Angularity](#), [SoilAnalyzer::Sample::ChangesSinceLastSave](#), [SoilMath::Stats< T1, T2, T3 >::Data](#), [SoilAnalyzer::Sample::GetAngularityVector\(\)](#), [SoilAnalyzer::Sample::GetPSDVector\(\)](#), [SoilAnalyzer::Sample::GetRoundnessVector\(\)](#), [Images](#), [SoilAnalyzer::Sample::Load\(\)](#), [on_analyzer_finished\(\)](#), [ParticleDisplayerFilled](#), [SoilAnalyzer::Sample::PSD](#), [SoilAnalyzer::Analyzer::Results](#), [SoilAnalyzer::Sample::Roundness](#), [Sample](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), [SaveMeMessage](#), [Settings](#), and [ui](#).

Here is the call graph for this function:



6.67.3.4 void VSAMainWindow::on_actionNeuralNet_triggered () [private], [slot]

Definition at line 353 of file [vsamainwindow.cpp](#).

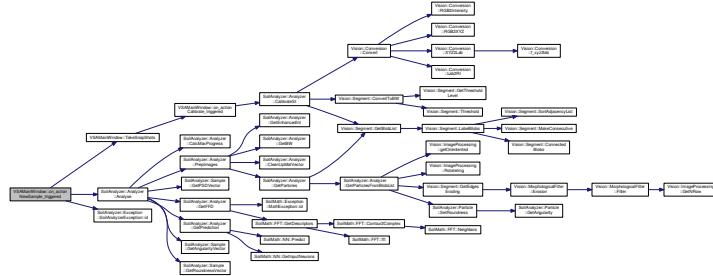
References [Analyzer](#), [SoilAnalyzer::Analyzer::NeuralNet](#), [nnWindow](#), [Settings](#), and [settingsWindow](#).

6.67.3.5 void VSAMainWindow::on_actionNewSample_triggered() [private], [slot]

Definition at line 361 of file [vsamainwindow.cpp](#).

References `SoilAnalyzer::Analyzer::Analyse()`, `Analyzer`, `CamError`, `SoilAnalyzer::Sample::ChangesSinceLastSave`, `EXCEPTION_NO_SN`, `APSHOTS_NR`, `SoilAnalyzer::Exception::SoilAnalyzerException::id()`, `Images`, `SoilAnalyzer::Sample::ParticlePopulation`, `Sample`, `SaveMe`, `Message`, `Settings`, `TakeSnapShots()`, and `ui`.

Here is the call graph for this function:



6.67.3.6 void VSAMainWindow::on_actionReport_Generator_triggered() [private], [slot]

Definition at line 552 of file [vsamainwindow.cpp](#).

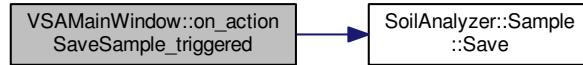
References [ReportGenWindow](#), [Sample](#), [Settings](#), and [ui](#).

6.67.3.7 void VSAMainWindow::on_actionSaveSample_triggered() [private], [slot]

Definition at line 460 of file [vsamainwindow.cpp](#).

References `SoilAnalyzer::Sample::ChangesSinceLastSave`, `SoilAnalyzer::Sample::IsLoadedFromDisk`, `Sample`, `SoilAnalyzer::SoilSettings::SampleFolder`, `SoilAnalyzer::Sample::Save()`, and `Settings`.

Here is the call graph for this function:

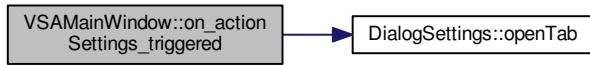


6.67.3.8 void VSAMainWindow::on_actionSettings_triggered() [private], [slot]

Definition at line 270 of file [vsmainwindow.cpp](#).

References [DialogSettings::openTab\(\)](#), and [settingsWindow](#).

Here is the call graph for this function:



6.67.3.9 void VSAMainWindow::on_actionUseLearning_toggled(bool arg1) [private], [slot]

Definition at line 512 of file [vsmainwindow.cpp](#).

References [Analyzer](#), and [SoilAnalyzer::Analyzer::PredictShape](#).

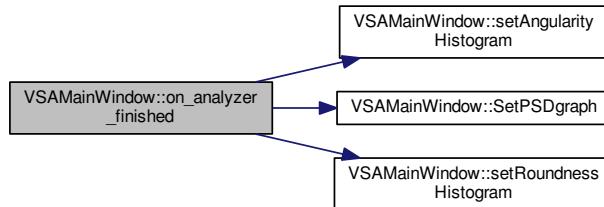
6.67.3.10 void VSAMainWindow::on_analyzer_finished() [private], [slot]

Definition at line 275 of file [vsmainwindow.cpp](#).

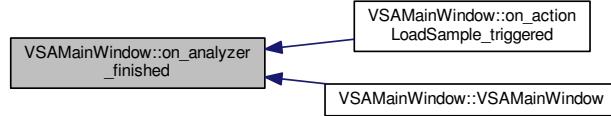
References [ParticleDisplayerFilled](#), [SoilAnalyzer::Sample::ParticlePopulation](#), [Sample](#), [setAngularityHistogram\(\)](#), [SetPSDgraph\(\)](#), [setRoundnessHistogram\(\)](#), and [ui](#).

Referenced by [on_actionLoadSample_triggered\(\)](#), and [VSAMainWindow\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



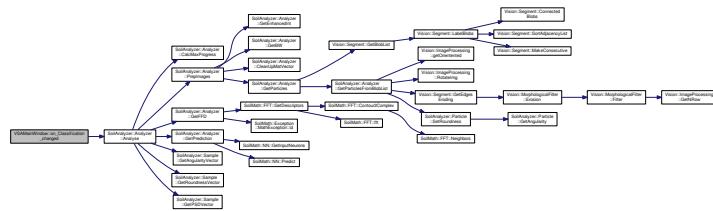
6.67.3.11 void VSAMainWindow::on_Classification_changed (int newValue) [private], [slot]

Definition at line 522 of file `vsamainwindow.cpp`.

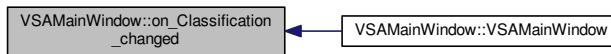
References `SoilAnalyzer::Analyzer::Analyse()`, `Analyzer`, `SoilAnalyzer::Sample::ChangesSinceLastSave`, `SoilAnalyzer::Sample::ParticleChangedStateAngularity`, `SoilAnalyzer::Sample::ParticleChangedStateRoundness`, `Sample`, and `ui`.

Referenced by [VSAMainWindow\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.67.3.12 void VSAMainWindow::on_compare_against() [private], [slot]

Definition at line 570 of file `vsamainwindow.cpp`.

References [PSDTicks](#), [SoilAnalyzer::SoilSettings::SampleFolder](#), [Settings](#), and [ui](#).

Referenced by [on_PSD_contextMenuRequest\(\)](#).

Here is the caller graph for this function:



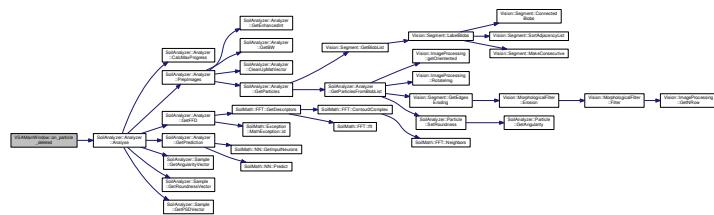
6.67.3.13 void VSAMainWindow::on_particle_deleted() [private], [slot]

Definition at line 539 of file [vsamainwindow.cpp](#).

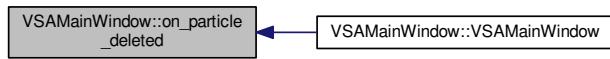
References `SoilAnalyzer::Analyzer::Analyse()`, and `Analyzer`.

Referenced by [VSAMainWindow\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.67.3.14 void VSAMainWindow::on_particleChanged(int newPart) [private], [slot]

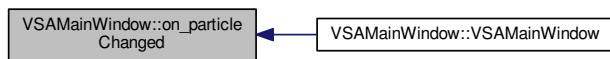
Definition at line 351 of file [vsamainwindow.cpp](#).

References [setAmpgraph\(\)](#).

Referenced by [VSAMainWindow\(\)](#)

```
graph LR; A[VSAMainWindow::on_particle_Changed] --> B[VSAMainWindow::setAmpgraph]
```

Here is the caller graph for this function:



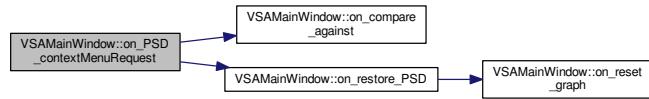
6.67.3.15 void VSAMainWindow::on_PSD_contextMenuRequest (QPoint point) [private], [slot]

Definition at line 561 of file [vsamainwindow.cpp](#).

References `on_compare_against()`, `on_restore_PSD()`, and `ui`.

Referenced by [VSAMainWindow\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



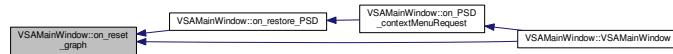
6.67.3.16 void VSAMainWindow::on_reset_graph (QMouseEvent * e) [private], [slot]

Definition at line 545 of file [vsamainwindow.cpp](#).

References [ui](#).

Referenced by [on_restore_PSD\(\)](#), and [VSAMainWindow\(\)](#).

Here is the caller graph for this function:



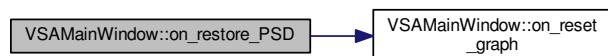
6.67.3.17 void VSAMainWindow::on_restore_PSD () [private], [slot]

Definition at line 628 of file [vsamainwindow.cpp](#).

References [on_reset_graph\(\)](#), and [ui](#).

Referenced by [on_PSD_contextMenuRequest\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



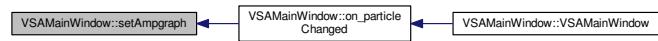
6.67.3.18 void VSAMainWindow::setAmpgraph() [private]

Definition at line 339 of file [vsamainwindow.cpp](#).

References [ui](#).

Referenced by [on_particleChanged\(\)](#).

Here is the caller graph for this function:



6.67.3.19 void VSAMainWindow::setAngularityHistogram() [private]

Definition at line 315 of file [vsamainwindow.cpp](#).

References [SoilAnalyzer::Sample::Angularity](#), [AngularityBars](#), [AngularityTicks](#), [SoilMath::Stats< T1, T2, T3 >::bins](#), [SoilMath::Stats< T1, T2, T3 >::HighestPDF](#), [SoilMath::Stats< T1, T2, T3 >::Mean](#), [Sample](#), and [ui](#).

Referenced by [on_analyzer_finished\(\)](#).

Here is the caller graph for this function:



6.67.3.20 void VSAMainWindow::SetPSDgraph() [private]

Definition at line 285 of file [vsamainwindow.cpp](#).

References [SoilMath::Stats< T1, T2, T3 >::CFD](#), [SoilAnalyzer::Sample::PSD](#), [PSDTicks](#), [Sample](#), and [ui](#).

Referenced by [on_analyzer_finished\(\)](#).

Here is the caller graph for this function:



6.67.3.21 void VSAMainWindow::setRoundnessHistogram() [private]

Definition at line 291 of file [vsamainwindow.cpp](#).

References [SoilMath::Stats< T1, T2, T3 >::bins](#), [SoilMath::Stats< T1, T2, T3 >::HighestPDF](#), [SoilMath::Stats< T1, T2, T3 >::Mean](#), [SoilAnalyzer::Sample::Roundness](#), [RoundnessBars](#), [RoundnessTicks](#), [Sample](#), and [ui](#).

Referenced by [on_analyzer_finished\(\)](#).

Here is the caller graph for this function:



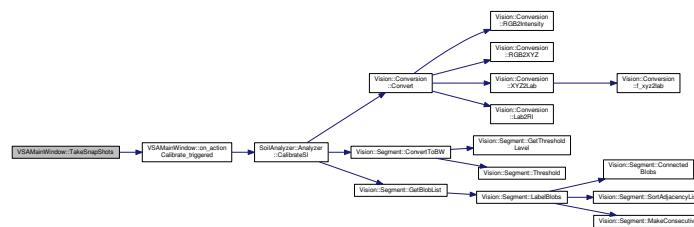
6.67.3.22 void VSAMainWindow::TakeSnapShots() [private]

Definition at line 391 of file [vsamainwindow.cpp](#).

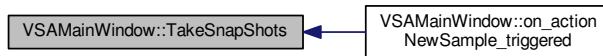
References [Analyzer](#), [SoilAnalyzer::Analyzer::Image_t::BackLight](#), [BacklightMessage](#), [SoilAnalyzer::Analyzer::CurrentSlfactor](#), [SoilAnalyzer::Analyzer::Image_t::FrontLight](#), [SoilAnalyzer::SoilSettings::HDRframes](#), [Images](#), [on_actionCalibrate_triggered\(\)](#), [Settings](#), [ShakeItBaby::Message](#), [SoilAnalyzer::Analyzer::SlfactorDet](#), [SoilAnalyzer::Analyzer::Image_t::SIPixelFactor](#), [SoilAnalyzer::SoilSettings::StandardNumberOfShots](#), [SoilAnalyzer::SoilSettings::useBacklightProjection](#), and [SoilAnalyzer::SoilSettings::useHDR](#).

Referenced by [on_actionNewSample_triggered\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.67.4 Member Data Documentation

6.67.4.1 SoilAnalyzer::Analyzer* VSAMainWindow::Analyzer = nullptr [private]

Definition at line 83 of file [vsamainwindow.h](#).

Referenced by [on_actionCalibrate_triggered\(\)](#), [on_actionLoadSample_triggered\(\)](#), [on_actionNeuralNet_triggered\(\)](#), [on_actionNewSample_triggered\(\)](#), [on_actionUseLearning_toggled\(\)](#), [on_Classification_changed\(\)](#), [on_particle_deleted\(\)](#), [TakeSnapShots\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.2 QCPBars* VSAMainWindow::AngularityBars = nullptr [private]

Definition at line 86 of file [vsamainwindow.h](#).

Referenced by [setAngularityHistogram\(\)](#), and [VSAMainWindow\(\)](#).

6.67.4.3 QVector<QString> VSAMainWindow::AngularityCat [private]

Initial value:

```
= {"Very Angular", "Angular", "Sub Angular",
    "Sub Rounded", "Rounded", "Well Rounded"}
```

Definition at line 92 of file [vsamainwindow.h](#).

Referenced by [VSAMainWindow\(\)](#).

6.67.4.4 `std::vector<double> VSAMainWindow::AngularityTicks = {1, 2, 3, 4, 5, 6}` [private]

Definition at line 94 of file [vsamainwindow.h](#).

Referenced by [setAngularityHistogram\(\)](#), and [VSAMainWindow\(\)](#).

6.67.4.5 `QMessageBox* VSAMainWindow::BacklightMessage = nullptr` [private]

Definition at line 76 of file [vsamainwindow.h](#).

Referenced by [TakeSnapShots\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.6 `QErrorMessage* VSAMainWindow::CamError = nullptr` [private]

Definition at line 74 of file [vsamainwindow.h](#).

Referenced by [on_actionNewSample_triggered\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.7 `SoilAnalyzer::Analyzer::Images_t* VSAMainWindow::Images = nullptr` [private]

Definition at line 84 of file [vsamainwindow.h](#).

Referenced by [on_actionLoadSample_triggered\(\)](#), [on_actionNewSample_triggered\(\)](#), [TakeSnapShots\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.8 `Hardware::Microscope* VSAMainWindow::Microscope = nullptr` [private]

Definition at line 81 of file [vsamainwindow.h](#).

Referenced by [~VSAMainWindow\(\)](#).

6.67.4.9 `DialogNN* VSAMainWindow::nnWindow = nullptr` [private]

Definition at line 72 of file [vsamainwindow.h](#).

Referenced by [on_actionNeuralNet_triggered\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.10 `bool VSAMainWindow::ParticleDisplayerFilled = false` [private]

Definition at line 96 of file [vsamainwindow.h](#).

Referenced by [on_actionLoadSample_triggered\(\)](#), and [on_analyzer_finished\(\)](#).

6.67.4.11 `QProgressBar* VSAMainWindow::Progress` [private]

Definition at line 73 of file [vsamainwindow.h](#).

Referenced by [VSAMainWindow\(\)](#).

6.67.4.12 `std::vector<double> VSAMainWindow::PSDTicks` [private]

Initial value:

```
= {0.0, 0.038, 0.045, 0.063, 0.075,
 0.09, 0.125, 0.18, 0.25, 0.355,
 0.5, 0.71, 1.0, 1.4, 2.0}
```

Definition at line 87 of file [vsamainwindow.h](#).

Referenced by [on_compare_against\(\)](#), [SetPSDgraph\(\)](#), and [VSAMainWindow\(\)](#).

6.67.4.13 `QReportGenerator* VSAMainWindow::ReportGenWindow = nullptr` [private]

Definition at line 78 of file [vsamainwindow.h](#).

Referenced by [on_actionReport_Generator_triggered\(\)](#).

6.67.4.14 `QCPBars* VSAMainWindow::RoundnessBars = nullptr` [private]

Definition at line 85 of file [vsamainwindow.h](#).

Referenced by [setRoundnessHistogram\(\)](#), and [VSAMainWindow\(\)](#).

6.67.4.15 QVector<QString> VSAMainWindow::RoundnessCat = {"High", "Medium", "Low"} [private]

Definition at line 90 of file [vsamainwindow.h](#).

Referenced by [VSAMainWindow\(\)](#).

6.67.4.16 std::vector<double> VSAMainWindow::RoundnessTicks = {1, 2, 3} [private]

Definition at line 91 of file [vsamainwindow.h](#).

Referenced by [setRoundnessHistogram\(\)](#), and [VSAMainWindow\(\)](#).

6.67.4.17 SoilAnalyzer::Sample* VSAMainWindow::Sample = nullptr [private]

Definition at line 82 of file [vsamainwindow.h](#).

Referenced by [on_actionLoadSample_triggered\(\)](#), [on_actionNewSample_triggered\(\)](#), [on_actionReport_Generator_triggered\(\)](#), [on_actionSaveSample_triggered\(\)](#), [on_analyzer_finished\(\)](#), [on_Classification_changed\(\)](#), [setAngularityHistogram\(\)](#), [SetPSDgraph\(\)](#), [setRoundnessHistogram\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.18 QMessageBox* VSAMainWindow::SaveMeMessage = nullptr [private]

Definition at line 75 of file [vsamainwindow.h](#).

Referenced by [on_actionLoadSample_triggered\(\)](#), [on_actionNewSample_triggered\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.19 SoilAnalyzer::SoilSettings* VSAMainWindow::Settings = nullptr [private]

Definition at line 80 of file [vsamainwindow.h](#).

Referenced by [on_actionAutomatic_Shape_Pediction_triggered\(\)](#), [on_actionLoadSample_triggered\(\)](#), [on_actionNeuralNet_triggered\(\)](#), [on_actionNewSample_triggered\(\)](#), [on_actionReport_Generator_triggered\(\)](#), [on_actionSaveSample_triggered\(\)](#), [on_compare_against\(\)](#), [TakeSnapShots\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.20 DialogSettings* VSAMainWindow::settingsWindow = nullptr [private]

Definition at line 71 of file [vsamainwindow.h](#).

Referenced by [on_actionNeuralNet_triggered\(\)](#), [on_actionSettings_triggered\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.21 QMessageBox* VSAMainWindow::ShakeItBabyMessage = nullptr [private]

Definition at line 77 of file [vsamainwindow.h](#).

Referenced by [TakeSnapShots\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

6.67.4.22 Ui::VSAMainWindow* VSAMainWindow::ui [private]

Definition at line 70 of file [vsamainwindow.h](#).

Referenced by [on_actionLoadSample_triggered\(\)](#), [on_actionNewSample_triggered\(\)](#), [on_actionReport_Generator_triggered\(\)](#), [on_analyzer_finished\(\)](#), [on_Classification_changed\(\)](#), [on_compare_against\(\)](#), [on_PSD_contextMenuRequest\(\)](#), [on_reset_graph\(\)](#), [on_restore_PSD\(\)](#), [setAmpgraph\(\)](#), [setAngularityHistogram\(\)](#), [SetPSDgraph\(\)](#), [setRoundnessHistogram\(\)](#), [VSAMainWindow\(\)](#), and [~VSAMainWindow\(\)](#).

The documentation for this class was generated from the following files:

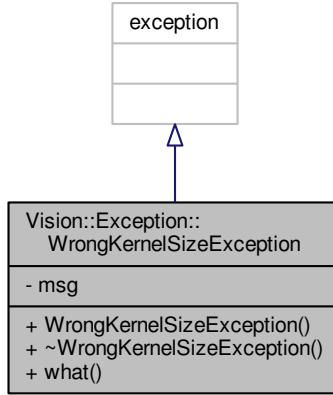
- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/vsamainwindow.h](#)

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/vsamainwindow.cpp](#)

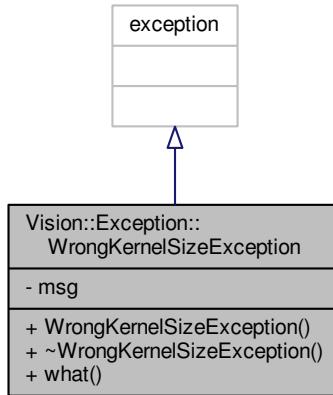
6.68 Vision::Exception::WrongKernelSizeException Class Reference

```
#include <WrongKernelSizeException.h>
```

Inheritance diagram for Vision::Exception::WrongKernelSizeException:



Collaboration diagram for Vision::Exception::WrongKernelSizeException:



Public Member Functions

- `WrongKernelSizeException` (string m="Wrong kernel dimensions!")
- `~WrongKernelSizeException` () `_GLIBCXX_USE_NOEXCEPT`
- const char * `what` () const `_GLIBCXX_USE_NOEXCEPT`

Private Attributes

- string `msg`

6.68.1 Detailed Description

Definition at line 20 of file [WrongKernelSizeException.h](#).

6.68.2 Constructor & Destructor Documentation

6.68.2.1 `Vision::Exception::WrongKernelSizeException::WrongKernelSizeException (string m = "Wrong kernel dimensions!")` [inline]

Definition at line 22 of file [WrongKernelSizeException.h](#).

6.68.2.2 `Vision::Exception::WrongKernelSizeException::~WrongKernelSizeException ()` [inline]

Definition at line 23 of file [WrongKernelSizeException.h](#).

6.68.3 Member Function Documentation

6.68.3.1 `const char* Vision::Exception::WrongKernelSizeException::what () const` [inline]

Definition at line 24 of file [WrongKernelSizeException.h](#).

6.68.4 Member Data Documentation

6.68.4.1 `string Vision::Exception::WrongKernelSizeException::msg` [private]

Definition at line 24 of file [WrongKernelSizeException.h](#).

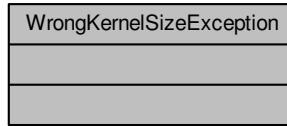
The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/WrongKernelSizeException.h](#)

6.69 WrongKernelSizeException Class Reference

`#include <WrongKernelSizeException.h>`

Collaboration diagram for WrongKernelSizeException:



6.69.1 Detailed Description

Exception class which is thrown when a wrong kernelsize is requested

The documentation for this class was generated from the following file:

- [/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/WrongKernelSizeException.h](#)