# Setting up a new view in mpx3gui software using QtCreator

1. The 'view' are QWidgets that are part of a StackedWidget. In QtCreator, go to the design form of the main window and right-click on the stackedWidget -> Insert page. A new QWidget is then added to the stackedWidget.
2. Create a new Qt Designer Form Class with QWidget as its base class.
3. Promote the QWidget to the newly made class.
4. Add functionalities to the newly made class (widget) as you would for a normal widget, dialog or window.

* Make sure to have a private variable *\_mpx3gui* (or something similar) and a public setter (*setMpx3GUI*) that sets the private variable to the main GUI, to have access to the other objects, in particular *dataset*.

1. Add an Action to the menubar in the designer of Qt to mpx3gui.ui. Add an icon to the *icons* folder in the *SpidrMpx3Eq* folder.
2. Optional: add a shortcut to the view in mpx3gui.cpp

# Dlib

*“Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.”* [http://dlib.net/]

This library can be used freely, license: <http://dlib.net/license.html>.

The library is used for its optimization and fast Fourier transform algorithms. It uses its own data structures, which have to be used in order to properly use the functions. The Qt data structures used in the rest of the code are converted to these dlib structures when a dlib function is needed, through simple for loops.

Simpler and/or better libraries are available for the optimization and fft, separately. To reduce the number of dependencies, this library was chosen, even if it means temporarily converting to other (unwieldy) data structures.

If however having separate libraries may prove to be superior/preferred, a developer is free to change all this.

FFT – used in *calcMTFdata()*

The number of rows and columns of the input matrix to calculate the fast Fourier transform, must be powers of 2. This means the data used must be ~~‘cropped’ in some way or~~ padded with zeros to assure this. To do this a dlib function called *order\_of\_two* is also used.

Optimization – used in *fitESFparams()*

For fitting the ESF data, a least squares method was used. Dlib provides a function that expects its own input \_vector and parameter\_vector data types. Converting to these data types requires small loops. Two prototype functions (not memberfunctions) have to be created: model and residual. The first describes the function that is to be fitted to the data, the second calculates the difference between the value that comes out of this model function and the data, at each data point.

It however does not provide a ‘goodness of fit’ parameter. This has to be calculated separately. For now, the Mean Squared Error and R squared is reported. (χ2 is not suitable for a non-linear model).

R2– used in *plotFitESF()*

Dlib provides a function *r\_squared* that calculates the R2, given two vectors: the fitvalues and the datavalues. R2 is also called the coefficient of determination and ranges from 0 – 1. 1 being a perfect fit and 0 being a poor fit.

# Calculation of MTF, NPS, DQE:

## Overview DQE calculation:

**DQE (n) = NEQ(n) / SNR2ideal**

For the DQE we need :

* **NEQ  
  NEQ = G2 \* [MTF(n)]2 / NPS(n)**  
  for this we need:
  + **(E or P)MTF (n)** (Dobbins, 1995) or (Flynn & Samei, 1999) resp.EMTF can be estimated by averaging several PMTF:P(resampling)MTF is the fourier transform of:
    - * **LSF**Line spread function.   
        Derivative of:
        + **ESF**Edge spread function. Pixel value against distance from edge.  
          We need:

**Sharp edge image**

* + **NPS (n)**for this we need:
    - **Fourier Transform**   
      of RoI’s open beam image(s)
  + **G2**  
    The system gain.
* **SNR2ideal**for this we need:
  + **SNR2ideal/exposure**a quantity that can be obtained by:
    - Tabulated values for specific beam qualities. (Dobbins, 1995) (Hoeschen, 2001)
    - Estimation by computational modelling. (Flynn & Samei, 1999) See included screen captures.
      * They use mass-energy absorption coefficients from: http://physics.nist.gov/PhysRefData/XrayMassCoef/  
        In dry air: http://physics.nist.gov/PhysRefData/XrayMassCoef/ComTab/air.html
  + The **exposure**used in the measurements of MTF and NPS.

## MTF

The user should be given the option to calculate the MTF in different ways.

The way that is implemented works by the following steps. In brackets are the corresponding class members.

* Determination of the edge line by finding the point of change from bright to dark or vice versa.
* Calculating the distance to the edge line for each point and plot the value of all points against the distance from the edge.
* Binning these points using a binsize set by the user. This gives the discrete ESF.
* Making a fit through the (binned) datapoints using an error function.
* Differentiation of this fitted function gives the LSF.
  + Differentiation by using the fitting parameters or numerical differentiation are both possible (see options).
* Fast Fourier Transforming of the LSF gives the MTF.
* Maximization of MTF (TODO, see below).

Another implementation is discussed by Samei et. al. (1998). This method consists of the following steps:

* Exponential-to-linear transformation.
* Image is converted to an 8-bit binary image by supplying a threshold value (average of the signal of the two sides).
* Determination of angle and position of the edge with double Hough transformation.
* Projection of points to the distance s from the edge. And collection into bins of 0.1\*pixelsize. This gives the ESFk array, the discrete ESF.
* This ESFk array is smoothed using a 4th order, Gaussian-weighted, moving polynomial fit. Local smoothing does not confine the ESF to a particular mathematical form: for each element in the ESFk array, a polynomial function is fit using adjacent elements and the initial element value is replaced by the value predicted by the fit.  
  “A least-squares fit is employed for which values near the center-point are strongly weighted by entering a different variance value for each point.” ?? Weighting function is a Gaussian (see form in article), of which parameters were chosen based on a test performed on simulated edge images… (Can we use this then? Do we need to do a similar test?)
* The ESFk array is numerically differentiated.
* The baseline of the LSF is substracted using a linear fit to the 10-nm-long portions of the LSF tails.
* A Hanning filter with a window width of 20 mm is applied to the LSF to establish a sampling rate of 0.05 cycles/mm.
* Presampling MTF is obtained by a Fast Fourier Transform of the LSF.
* This is repeated for 17 angles in an +-0.16 degrees, by varying by 0.02 degrees each time. The MTF’s are each integrated from 0 – 2 cycles/mm and the angle associated with the maximum of this MTF integral is identified as the best estimate for the edge angle and its MTF is the result.

The extent of the regions used by Samei are not feasible for the Medipix3… The LSF looks significantly different than ours, so the Hanning filter and baseline substraction or extrapolation of the tails (as in other papers) might not be the way to go. (?)  
But we CAN use the iterative MTF maximization by varying the angle. Only one parameter changes and then the entire “Calc MTF” procedure can be repeated:

(still) TO DO for MTF calculation:

* Maximize MTF by angle variation.
* (?)Use a Hanning filter.
* (?) Use baseline substraction or extrapolation of the tails.
* Implement slit method. (Try Fujita’s slit method. (“Simple”).)
* Get the calculation to work for horizontal edges.
* (?) Better quantization of goodness of fit. 🡪 R2 and mean squared error are given.

## NPS calculation

To calculate the NPS, the method as described in the Handbook of Medical Imaging (Beutel et al.) and by (Dobbins, 1995). The method is described by the steps below in order of the calculation. Which is not necessarily the order of implementation.

TO DO:

* Determine characteristic curve.
* Convert pixel values to be linearly proportional to exposure (using characteristic curve).
* [User] Take multiple flat field images (at multiple exposures).
* [User] Determine best size of ROI (N) and number of ROI. (M). Choose ROI.
* ~~Correct for background trends from ROI by fitting a planar ramp.~~
* ~~Calculate the 2D squared Fourier transforms of the ROI.~~
* ~~Take the ensemble average of all these FT of ROIs 🡪 NPS.~~
* ~~Calculate 1D NPS.~~

## DQE

According to the method used for the NPS, it is then not too difficult to calculate the DQE. (see overview)

TO DO:

* Calculate NEQ.
* Use EMTF for NEQ calculation (in addition to the PMTF, both should be reported). (Dobbins, 1995)
* Determine (given by manufacturer or measurements and simulations (Flynn & Samei, 1999)) the SNR2ideal of the incident x-ray beam.
* Calculate DQE.
* Calculate DQE(0) by extrapolation of the low-frequency part back to zero.

# QCstmDQE class

The user can select a region in the visualization tab by dragging the mouse with the right mousebutton pressed. An option window pops up and one can then choose to use the region for DQE calculation. The GUI then directly switches to the DQEtab, where the correct data is already imported and plotted. The data is also binned and plotted. Whether the user sees the unbinned data or not is determined by the *show prebinning data* checkbox.

When the software is connected to the detector, data can be taken directly from the DQEtab. The view will switch to the visualization tab.

The way the MTF is calculated can be chosen by the radio buttons, checkboxes and lineEdit:

* *Edge vs. Slit radio buttons*These let the user choose between the edge and slit method. Both should be reported for the most accurate MTF. (TO DO: implement slit method.)
* *Bin data*The user can choose to bin the data and even set the binsize (as long as it is smaller than the dataset and larger than 1.
* *Error function* vs *Smoothing*  
  If *Error function* is selected in the combobox, an error function is fitted to the (binned) data. This then represents the ESF and is used to calculate the derivative (LSF) (see next point).  
  When *Smoothing* is selected, the binned data is smoothed by a 4th order (Gaussian weighted) moving polynomial fit, using a window width as specified by the user.
* *Window width*  
  Here the user specifies the window width to be used in the local fitting. This must be an uneven number, since there must be as much data points to the left of the point to be fitted as to the right. It must also be at least 3. And finally the width must be smaller than the entire dataset. The program gives off warnings and changes the value automatically when this happens.
* *Func der*  
  When this is checked and an error function has been used (both checked), the derivative data is calculated by using the formula for the derivative of the error function, using the fitted parameters.  
  If the error function has not been used, but instead the binned data was smoothed, the numerical derivative of this smoothed data is taken as the LSF.
  + When not checked, the numerical derivative (3-point) of the binned ESF data is taken.

# OptionsDialog class

The optionsdialog contains the options for the MTF and NPS calculation all in one place, so as not to clutter the main interface. Only one optionsdialog exists and is shown *on\_optionsPushbutton\_clicked*. When closed, it is not deleted. The object gets deleted when the parent (\_mpx3gui) is deleted.

Three buttons are available.

* *Ok*  
  The settings are applied and saved\* and the dialog is closed (but is not destroyed). When opening the dialog again, the settings are the way the user changed them before.
* *Cancel*The settings are not applied and not saved. The changes the user made in the ui are changed back to how they were when the user opened the window (before any changes were made). The dialog is closed.
* *Apply*The settings are applied and saved\*, but the dialog is not closed. This way the user can see what options are used when calculating the MTF and NPS. (These are also noted in the log (see LOG).)

\*The settings are saved in a private variable of the *optionsDialog* called *\_currentSettings*. This is a QHash, which is a container that stores values by a key (in this case a QString) and a value (in this case an int). This allows for intuitive looking up and setting of the settings. QHash is faster than QMap because the data is stored in an arbitrary order.

## Adding options

To add an option, a few things have to be done:

1. Add the widget of choice to the *optionsDialog.ui* file in designer mode. You do not need to *go to slot…*
2. Choose an appropriate, obvious and short name for the option/setting and rename the widget something similar.   
   [For example: the option for the size of the bins is called *binsize* and the corresponding widghet (a LineEdit) is called *binSizeLineEdit*. It should not be necessary to switch to the ui to look up the name.]
3. In *optionsDialog.cpp*, add the appropriate expression for saving the option to *setCurrentSettings()*.  
   [For example: the binsize is set like this:   
   \_currentSettings["binsize"] = ui->binSizeLineEdit->text().toInt();]
4. Also add an expression for resetting the option in *resetSettings().*
5. In *QCstmDQE.cpp,* add an expression for setting the corresponding private variable in *on\_apply\_options()*. Or start some procedure.  
   [For example: QCstmDQE has a variable called *\_windowW* that is used to locally fit the data (smoothing). This is set by the following expression:  
   \_windowW = options.value("windowW");  
   In the case of *\_binsize*, if it has changed, the binned data is immediately plotted again.]
6. Test.

# Works Cited

Dobbins. (1995). DQE(f) of four generations of computed radiography acquisition devices. *Medical Physics*.

Flynn, & Samei. (1999). Experimental comparison of noise and resolution for 2k and 4k storage phosphor radiography systems. *Medical Physics*.

Hoeschen, D. (2001). DQE of digital x-ray imaging systems: a challenge. *SPIE*.