Computer Science Project

Image Processing Using QT & Open CV

University of Bourgogne

Umamaheswaran Raman Kumar

May 19, 2016

Contents

1 Objective

The main objective of this project is to create a Qt/C++ based GUI application with attention to follow proper Object Oriented Programming concepts.

2 Introduction

The project is a Qt widget application which does image processing using OpenCV which is an open source computer vision library. Image processing is a technique which applies different mathematical operations on a given image to obtain a processed image which provides more information about the original image. This report provides a detailed view about the project implementation and the learning outcomes.

3 Development Tools

- QT v5.5.1
- OpenCV v2.4.12

4 Use Case Diagram

The figure below shows the high level view of the features provided by the application to the user.

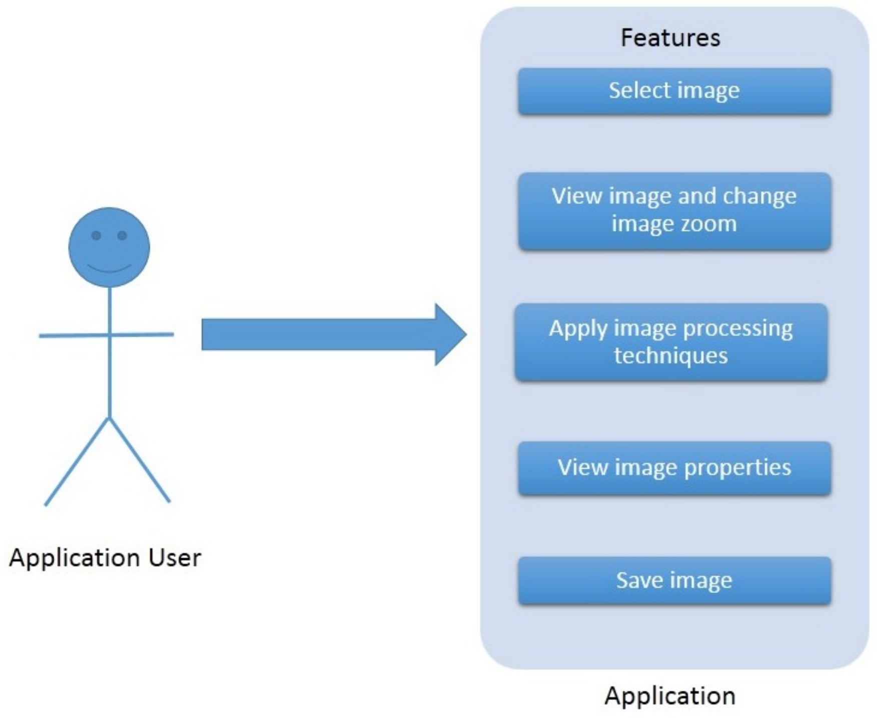


Figure 1: *Application Use Case*

5 Implementation

The main idea behind the implementation is to come up with a robust base design so that it becomes easy to add additional features at any later stage of the development. Modularizing the classes properly, increases the readability, reusability and maintainability of the code.

5.1 Class Diagram

Figure 2 shows the class diagram of the application and it clearly visualizes the hierarchy of the classes and their associations.

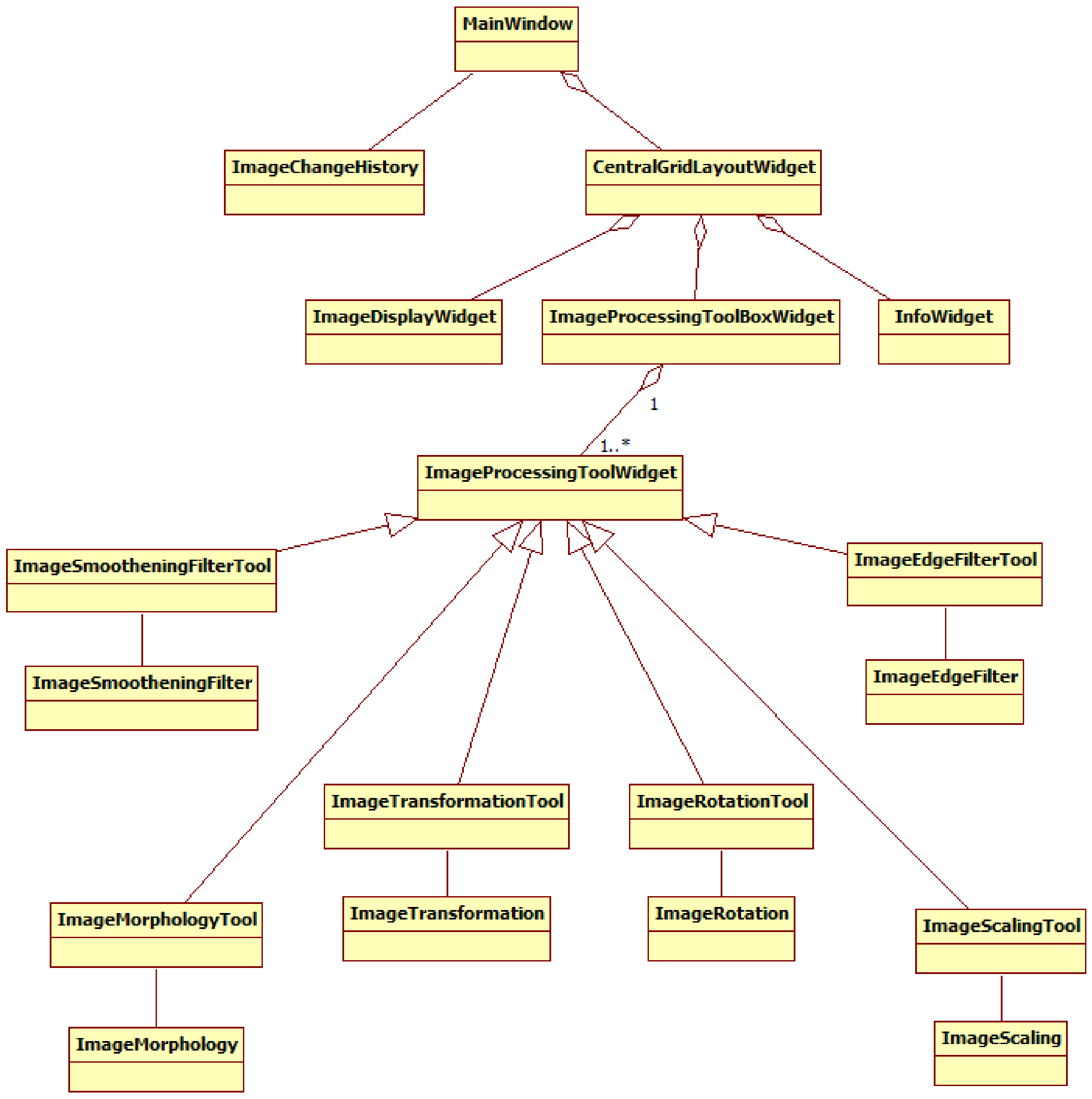


Figure 2: *Class Diagram*

5.2 Detailed Description of Classes

5.2.1 Class: MainWindow

The 'MainWindow' class is the main UI class and all the other UI classes are directly or indirectly a part of this class. It has 3 major parts and are the central widget, tool bar and status bar. The central widget holds most of the functionalities like displaying image, information about the image and tools for image processing and so it is promoted as a custom class 'CentralGridLayoutWidget' for better maintainability.

The tool bar holds functions that are mostly not related to image processing. The functionalities implemented in the tool bar are:

- Browse for a new image to be loaded in the UI
- Undo the changes applied on an image
- Redo the changes applied on an image
- Save the modified image as a new image or overwrite the existing image
- Zoom in the image display
- Zoom out the image display
- Activate selection tool to crop the image or to set it as the region of interest. They are the two dependent actions which are activated only when the selection tool is checked and a region is selected
- Improve image contrast

The status bar is mainly used to show temporary messages or warnings to the user related to the status of the image being currently processed or about the supported image conversion formats. It also displays a permanent message showing the current percentage zoom of the image being displayed.

The 'MainWindow' class also has the 'ImageChangeHistory' class as member and so it connects this class with all the Tool widgets to share images between them. It also holds the connection to equip the Tool widgets to show status messages.

5.2.2 Class: ImageChangeHistory

The 'ImageChangeHistory' class is used for storing all the images created in the course of running the application. This class also stores the image path from where the original image was loaded, so the user can be pointed to the same path when he/she saves the image. When the user selects a new image, all the images stored previously are discarded. This change history is mainly used for the undo and redo options.

5.2.3 Class: CentralGridLayoutWidget

The 'CentralGridLayoutWidget' is partitioned into 3 sections and each section is implemented in a separate class extending the 'QWidget' class. The classes are:

- ImageDisplayWidget: Display the image
- InfoWidget: Display image properties
- ImageProcessingToolBoxWidget: Display the toolbox for used for image processing

5.2.4 Class: ImageDisplayWidget

The 'ImageDisplayWidget' displays the images to the user. The same region is updated for the below actions:

- New image is loaded
- Image processed by any of the image processing tool
- Image zoomed in or zoomed out
- Image region selected
- Image cropped

5.2.5 Class: InfoWidget

The 'InfoWidget' displays the below properties of an image:

- Name of the image
- Width in pixels
- Height in pixels
- Image format
- Absolute path of the image

5.2.6 Class: ImageProcessingToolBoxWidget

The 'ImageProcessingToolBoxWidget' displays all the image processing toolsets by adding them to the 'QToolBox' widget. The 'QToolBox' shows multiple widgets in a cascaded view and allows the user to select only the toolset he/she wishes to work on. This gives more space to add other widgets to the main UI and also makes the UI look simple and elegant.

5.2.7 Class: ImageProcessingToolWidget

The 'ImageProcessingToolWidget' is the parent class for all the widgets added to the 'ImageProcessingToolBoxWidget'. This class equips the widgets to get equipped with functions common to all the classes that extend it. The main functions included in this class are:

- Get and set input image
- Get and set output image
- Signals to notify:

  - Change of status of the image being currently processed
  - Output image updated

- Slots to handle:
  - Update of input image
  - Update of region of interest

5.2.8 Class: ImageSmootheningFilterTool & ImageSmootheningFilter

The 'ImageSmootheningFilterTool' is the widget that displays the list of smoothening filters and the filter mask radius from which the user can select and apply on the image. It also has a checkbox for applying the filter only on the region of interest and not on the whole image. This checkbox is enabled only if the user had selected a region of interest.

The 'ImageSmootheningFilter' class has separate functions implemented for each filter which internally calls the OpenCV function to apply the filter on the image. When the user clicks on the apply button, based on the user's selection, the corresponding function is called with the options selected by the user as parameters. After the image is processed and the output image is updated, a signal is sent from the widget with the new image. Below are the list of filters the user can select from:

- Bilateral filter
- Box filter (Normalized)
- Gaussian filter
- Median filter

5.2.9 Class: ImageEdgeFilterTool & ImageEdgeFilter

The 'ImageEdgeFilterTool' is the widget that displays the list of edge filters which the user can select and apply on the image. The 'ImageEdgeFilter' class has separate functions implemented for each edge filter which internally calls the OpenCV function to apply the edge detection on the image. When the user clicks on the apply button, based on the user's selection, the corresponding function is called with the options selected by the user as parameters. After the image is processed and the output image is updated, a signal is sent from the widget with the new image. Below are the list of edge filters the user can select from:

- Canny
- Scharr
- Sobel

5.2.10 Class: ImageMorphologyTool & ImageMorphology

The 'ImageMorphologyTool' is the widget that displays the list of morphological operations and the structuring element radius from which the user can select and apply on the image. It also has a checkbox for applying the operation only on the region of interest and not on the whole image. This checkbox is enabled only if the user had selected a region of interest.

The ‘ImageMorphology’ class has separate functions implemented for each morphology operation which internally calls the OpenCV function to apply the morphology operation on the image. When the user clicks on the apply button, based on the user’s selection, the corresponding function is called with the options selected by the user as parameters. After the image is processed and the output image is updated, a signal is sent from the widget with the new image. Below are the list of morphology operations the user can select from:

- Closing
- Dilate
- Erode
- Opening

5.2.11 Class: ImageTransformationTool & ImageTransformation

The ‘ImageTransformationTool’ is the widget that displays the list of colour transformations which the user can select and apply on the image. The ‘ImageTransformation’ class has separate functions implemented for each transformation which internally calls the OpenCV function to apply the transformation on the image. When the user clicks on the apply button, based on the user’s selection, the corresponding function is called with the options selected by the user as parameters. After the image is processed and the output image is updated, a signal is sent from the widget with the new image. Below are the list of colour transformations the user can select from:

- Binary transformation (applied separately for each channel if the image is RGB)
- Grayscale transformation
- Negative transformation

5.2.12 Class: ImageRotationTool & ImageRotation

The ‘ImageRotationTool’ is the widget that displays the list of rotation directions and the angle of rotation which the user can select and apply on the image. The ‘ImageRotation’ class has separate functions implemented for each rotation direction which internally calls the OpenCV function to apply on the image. When the user clicks on the apply button, based on the user’s selection, the corresponding function is called with the options selected by the user as parameters. After the image is processed and the output image is updated, a signal is sent from the widget with the new image. Below are the list of rotation directions the user can select from:

- Clockwise rotation
- Anticlockwise rotation

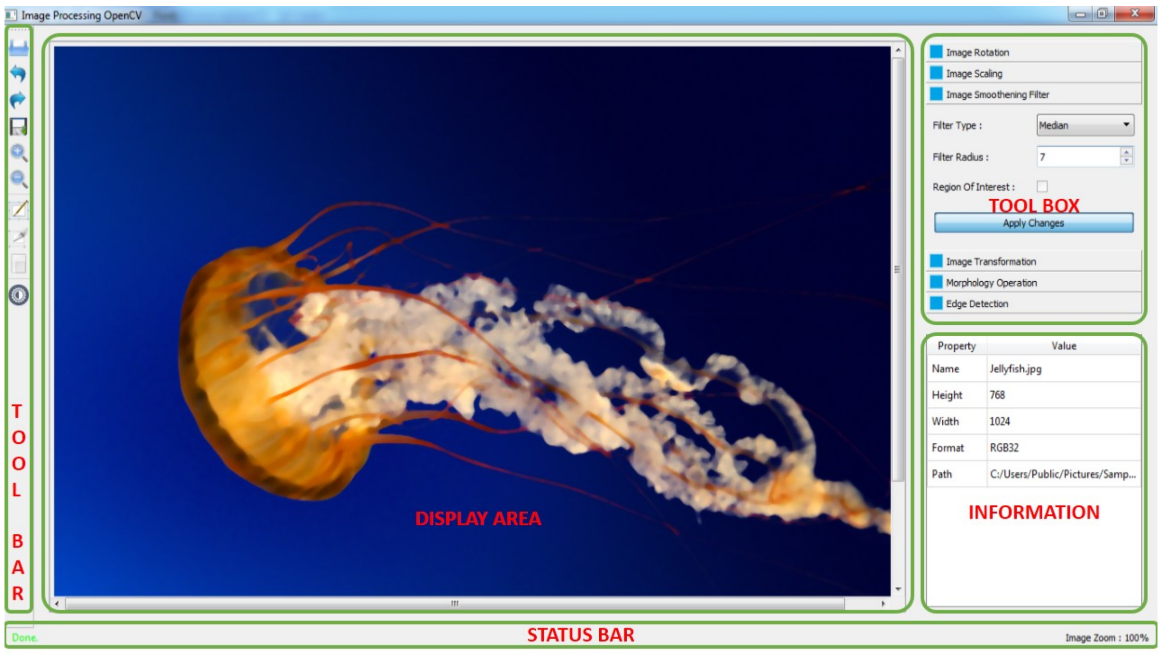
5.2.13 Class: ImageScalingTool & ImageScaling

The ‘ImageScalingTool’ is the widget that displays the list of scale types and the scale percentage which the user can select and apply on the image. The ‘ImageScaling’ class has separate functions implemented for scale type which internally calls the OpenCV function to apply the scaling on the image. When the user clicks on the apply button, based on the user's selection, the corresponding function is called with the options selected by the user as parameters. After the image is processed and the output image is updated, a signal is sent from the widget with the new image. Below are the list of scale types the user can select from:

- Scale down
- Scale up

6 GUI Design

Figure 3 shows the final GUI of the application highlighting the main divisions in the main window.



Filter Type : Median
Filter Radius : 7
Region Of Interest : ☐

Property	Value
Name	Jellyfish.jpg
Height	768
Width	1024
Format	RGB32
Path	C:/Users/Public/Pictures/Samp...

Done.
Image Zoom : 100%

Figure 3: *Application GUI*

7 Future Enhancements

Listed below are the areas where the application can be enhanced further:

- Additional image processing tools in the tool box widget
- Edit multiple images simultaneously
- Support for more image formats (currently tested with jpeg, png and bmp)
- Improve GUI look and feel

8 Important links

Code repository: https://github.com/umaatgithub/ImageProcessingOpenCV

9 References

[1] http://doc.qt.io/qt-5/signalsandslots.html
[2] http://stackoverflow.com/questions/15881913/how-to-link-opencv-in-qtcreator-and-use-qt-library
[3] http://docs.opencv.org/3.0-beta/modules/imgproc/doc/imgproc.html
[4] http://opencvexamples.blogspot.com/p/learning-opencv-functions-step-by-step.html
[5] https://asmaloney.com/2013/11/code/converting-between-cvmat-and-qimage-or-qpixmap/
[6] http://stackoverflow.com/questions/15007304/histogram-equalization-not-working-on-color-image-opencv
[7] https://sourceforge.net/projects/openiconlibrary/?source=typ_redirect