



TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií



Hochschule
Zittau/Görlitz
UNIVERSITY OF APPLIED SCIENCES

**TECHNICKÁ UNIVERZITA V LIBERCI
HOCHSCHULE ZITTAU/GÖRLITZ**

Educational Application for Image Processing Library

Study course: Mechatronics

Supervisor: Prof. Dr. rer. nat. Stefan Bischoff

2015

Bc. Zbyšek Zapadlík

Assignment for the master thesis

Course of studies: Mechatronics

Student's name: Zbyšek Zapadlík

Subject:

Development of educational software for image processing library

Assignment:

- Create application for windows operating system, which allows to user use most of OpenCV library functions without text-oriented programming
- Results and intermediate results can be saved into different files
- Application can work with images and video streams
- Every function that can be used will have own description of meaning and input parameters

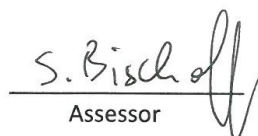
Assessor: Prof. Dr. Stefan Bischoff (Hochschule Zittau/Görlitz)

Supervisor: Prof. Dr. Stefan Bischoff (Hochschule Zittau/Görlitz)

Date of issue: 12.03.2015

Date of submission: 12.07.2015

Registry-Number.: MA/EMIm13 - 06/ 15


Assessor


Dean

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Dedication

First, I would like to dedicate this thesis to my family for standing with me all the time. Thanks to my parents i was able to complete the academic life with success and I know, that not everybody has this possibility.

Secondly, i would like dedicate this thesis to my girlfriend, she was my source of strength and my savior since we met.

At last I would like to sincerely thank to my supervisor, always kindly and understanding person.

Abstract

The aim of this diploma thesis is to develop a desktop application for the operating system *Windows*. This application will be primarily used as an educational tool on the lectures of image processing at Hochschule Zittau/Goerlitz.

Some of participants in this lectures does have none or not sufficient experience with a text oriented programming language, and they are not able to learn it during one semester subject. This application will allow the students to use a functions of image processing library called *OpenCV* without knowledge of text-based language, show an influence of used function to an image and adjust a parameters and see immediate result. The application include short description of each function and also direct link to the information contained in *OpenCV* manual.

The application is programmed with use of *Microsoft Foundation Classes* (MFC) in *Visual Studio 2013*. Chapters at beginning of this paper work are concerning about MFC conventions and structure of application itself.

For more experienced students or even programmers can be this work also useful. Well commented application code and the chapter about how to set-up *Visual Studio 2013* with *OpenCV* library and the chapter about how to add new functions gives possibility to extend the code without too much effort.

The application can be used for quickly testing with real images or live video from camera and only then eventually use the text based language.

Keywords

Master Thesis, Image Processing, OpenCV, Microsoft Foundation Library, Desktop Application, Educational tool

Abstrakt

Cílem této diplomové práce je vyvinout *desktopovou* aplikaci pro operační systém Windows. Tato aplikace bude primárně používána jako výuková pomůcka na hodinách předmětu *Zpracování obrazu* na škole Hochschule Zittau/Görlitz .

Určitá část studentů účastnících se těchto hodin nemá žádné, nebo nedostatečné zkušenosti s textově orientovaným programovacím jazykem a není schopna se ho naučit v rámci jednosemestrového předmětu. Tato aplikace umožní právě těmto studentům použít funkce knihovny *OpenCV* bez znalosti jakéhokoliv textově orientovaného jazyka. Umožní aplikovat jednotlivé funkce knihovny na libovolný obrázek a modifikovat jejich parametry a zároveň vidět mezi výsledky. Aplikace obsahuje krátký popis každé funkce a také odkaz na informace obsažené v *OpenCV* manuálu.

Aplikace je naprogramována s použitím *Microsoft Foundation Classes* (MFC) ve vývojovém prostředí *Visual Studio 2013*. Kapitoly na začátku této práce se zabírají MFC konvencemi a strukturou programu jako takového.

Pro zkušenější studenty nebo pro programátory tato práce může být také užitečná. Dobře okomentovaný kód, kapitola o nastavení *Visual Studia 2013* pro práci s *OpenCV* knihovnou a kapitola uvádějící detailní popis jak přidat novou funkci dává možnost rozšiřovat kód aplikace snadno dle požadavků.

Aplikace může být použita na velmi rychlé testování na reálných obrázcích nebo videa z kamery a teprve až pak případné použití textově orientovaného jazyka.

Klíčová slova

Diplomová práce, Zpracování obrazu, OpenCV, Microsoft Foundation Library, Desktopová aplikace, Výuková pomůcka

Content

1 INTRODUCTION.....	8
1.1 LIBRARY FOR IMAGE PROCESSING.....	8
1.2 MICROSOFT FOUNDATION CLASSES.....	9
1.2.1 MFC and C++.....	9
1.2.2 MFC Conventions.....	10
2 DEVELOPING AN APPLICATION.....	11
2.1 Setting up Visual Studio – OpenCV.....	11
2.2 Requirements on program structure.....	13
2.3 Realization.....	14
2.3.1 General classes.....	15
2.3.2 General dialog classes.....	15
2.3.3 Function classes.....	15
2.3.4 Parameters dialog classes.....	16
3 TESTING THE APPLICATION.....	17
3.1 Overview of the application.....	17
3.1.1 Project managing.....	18
3.1.2 Load/save image.....	20
3.1.3 Other.....	20
3.2 Implemented functions.....	22
3.2.1 Threshold.....	22
3.2.2 Blur.....	24
3.2.3 Median blur.....	25
3.2.4 Gaussian Blur.....	26
3.2.5 CvtColor.....	27
3.2.6 Canny edge.....	29
3.2.7 Connected components labeling.....	30
3.2.8 Adaptive threshold.....	31
3.3 Combining of functions.....	32
4 VIDEO SUPPORT.....	34
4.1 Difference between image and video input.....	34
4.2 Implementation.....	34
5 NEW FUNCTIONS.....	38
6 CONCLUSION.....	40
7 RESOURCES.....	41

1 INTRODUCTION

In chapter 1.1 we will make an overview of image recognition library which is center of interest in this work.

Next, in chapter 1.2, will be described library used for develop the desktop application.

1.1 LIBRARY FOR IMAGE PROCESSING

The following text is taken from [1]

Abbreviation OpenCV means **Open Source Computer Vision** library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are also many startup projects. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of special instructions when available. OpenCV is written natively in C++.

1.2 MICROSOFT FOUNDATION CLASSES

The text in this chapter is taken from [2]

In essence, MFC is a SDK interface, a library consisting in a set of classes that act as wrappers around portions of the Windows API, so that C++ programmers may program Windows using some concepts of the object-oriented programming (OOP) paradigm and the C++ language (the Win32 API is based on C).

Some tools, such as Microsoft Visual Studio, are capable of automatically generating large amounts of MFC skeleton code for use in a project.

MFC was first oriented mostly for enterprise-level programming projects, created in an age most code was done in C and Object Oriented Programming was only small part. Since the release of Visual Studio 2008 Service Pack 1, Microsoft seems to once again be actively supporting the MFC.

Many users today find acceptable .NET Framework for a low complexity program to have a memory footprint of 30-80Mb (this is common in Java or .Net applications), low response times or "outside of your control" applications like the internet now provides. It is therefore debatable if the impact of use of MFC in small applications outweighs the benefits the libraries provides. Most of the software made specifically for Windows today uses MFC.

1.2.1 MFC and C++

The MFC design principle is an attempt for simplification. The wrapper classes were designed so to simplify some tasks and automates others. Because of those facts, however, a certain amount of fine-tunable control was lost from the raw Win32 API or excessive automation was archived. The MFC has been recognized as having serious design flaws and inconsistencies and has not been actively maintained. The C++ language and best practices have evolved and today this constitutes a barrier for the utilization of the framework.

As MFC predates the STL standardization in to the C++ language, it implements its own versions of the STL containers, not as complete and even inconsistent, this simplistic solutions the MFC implementations tend to be faster, however you should prefers to use the STL when ever you can, it will make the code more C++ standard and permit easier portability in converting the code to multi-platform.

1.2.2 MFC Conventions

The MFC uses the Hungarian notation. It uses prefixes, like "m_" to indicate a member variable or "p" to indicate a pointer, and the rest of the name is normally written out in CamelCase (the first letter of each word is capitalized).

2 DEVELOPING AN APPLICATION

2.1 Setting up Visual Studio – OpenCV

If we are planning to use Microsoft Visual Studio with the library OpenCV we must set up the project properties for the compiler to know, where is the folder with OpenCV files. For clarity it will be shown in this sub-chapter.

In this document, we will use *Microsoft Visual Studio 2013 Ver. 12* with *.NET framework 4.5.51209* and *OpenCV 2.4.11*.

Our PC station is 64-bit, but if you have 32-bit this should not be any complication, just replace in every path in this subchapter the architecture label “x64” by “x86”.

Firstly, run installer for OpenCV, it will just extract files to destination folder. We have chosen “E:\opencv\”.

Create new Project in Visual Studio, language choose Visual C++ and type MFC application, dialog based option and static library.

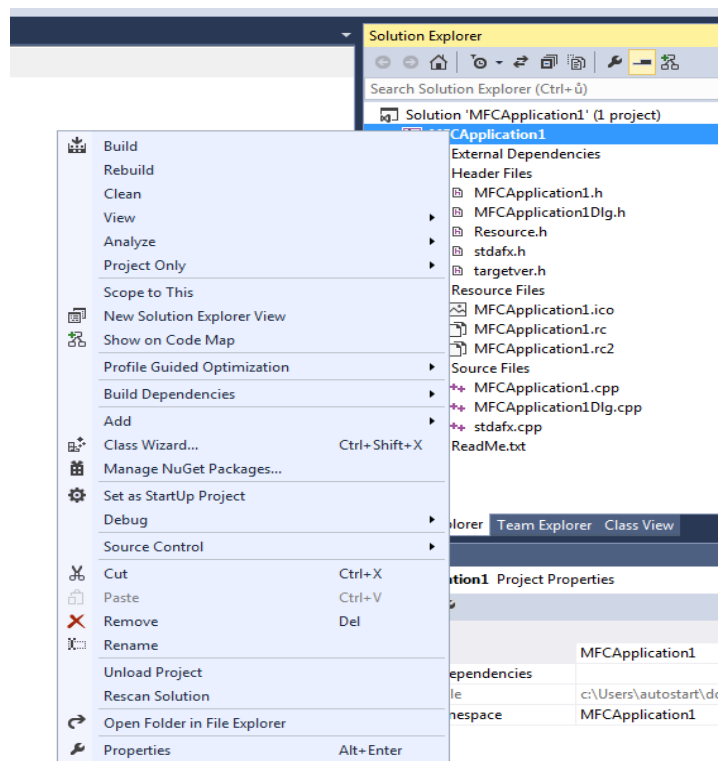


Figure 1: Project pop-up menu

Every settings we will made in Project property pages, that can be open by left clicking on project name in solution explorer and choosing *properties*.

Check if the platform configuration corresponds to your 64 or 32 bit system. Leave debug setting.

The first properties that we must set are in *VC++ Directories*. Add the following to executable directories: *"E:\opencv\build\x64\vc12\bin"*
to include directories: *"E:\opencv\build\include"*
to library directories: *"E:\opencv\build\x64\vc12\bin"* and *"E:\opencv\build\x64\vc12\lib"*
Then to the General under C/C++ add to Additional Include Directories *"E:\opencv\build\include"*
Then to the General under Linker add to Additional Include Directories *"E:\opencv\build\x64\vc12\bin"* and *"E:\opencv\build\x64\vc12\lib"*
And to the input add according the following picture:
(libraries are *opencv_core2411d.lib;opencv_highgui2411d.lib;opencv_imgproc2411d.lib*)

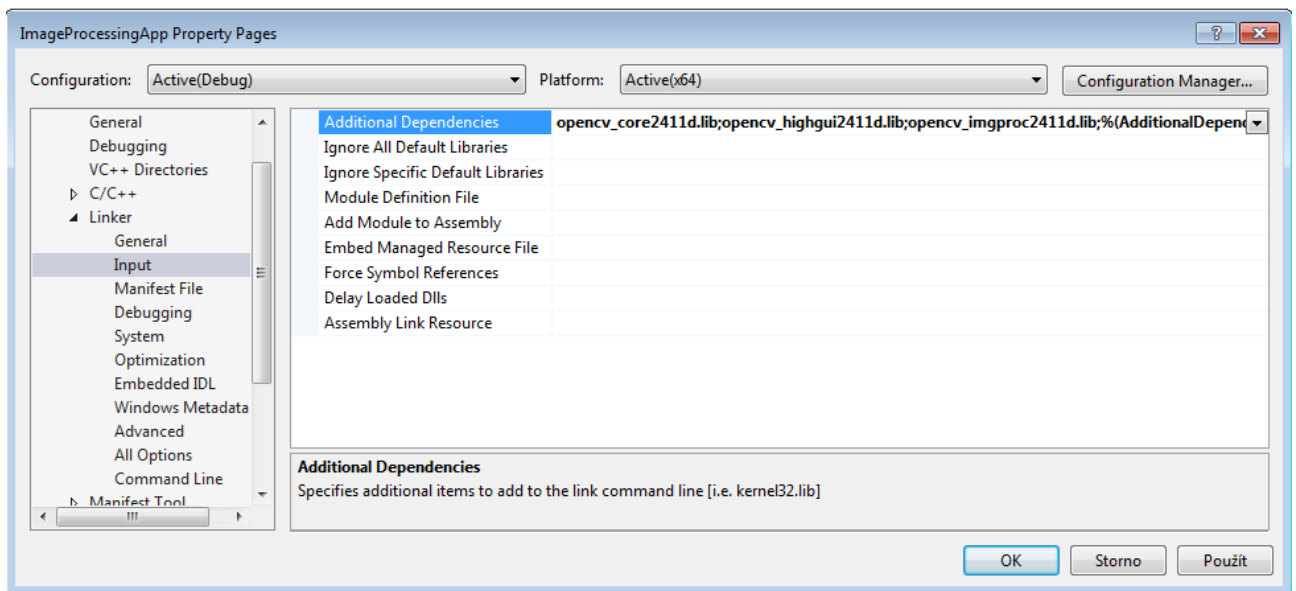


Figure 2: Property pages for settings of project

2.2 Requirements on program structure

When developing such an application (complex structure, many classes, use of inheritance, pointers and messages between classes) we can create few rules, that will simplify our work. That applies also for everyone who will possibly extend our code. At first point we will be compliant with MFC conventions described in chapter 1.2.2. Secondly, name of all our classes will be in this form: `c[describing_name]`. All classes that belong to dialogs in this form: `c[describing_name]Dlg`. The appropriate source code files (`*.cpp`) and the header files (`*.h`) will have the same name as the class name.

We know, that application will use many functions of library OpenCV. Therefore we can assume, that we need universal class for such a function. This class will have some arguments and method where we will directly call OpenCV function. There we will use inheritance of C language.

We know, that we want to use more of these functions and we have to see all of them in some list and have possibility to sort it. Therefore, we will use array of pointers and the type of those pointers will be the base class.

2.3 Realization

This chapter will describe program that was realized. We can split all classes to three types (General, Function, Parameter Dialog) described in following sub chapters. On next picture we can see an overview, that clarifies a process of changing parameters of the function blur. This also applies for any other function, we choose blur just as an example.

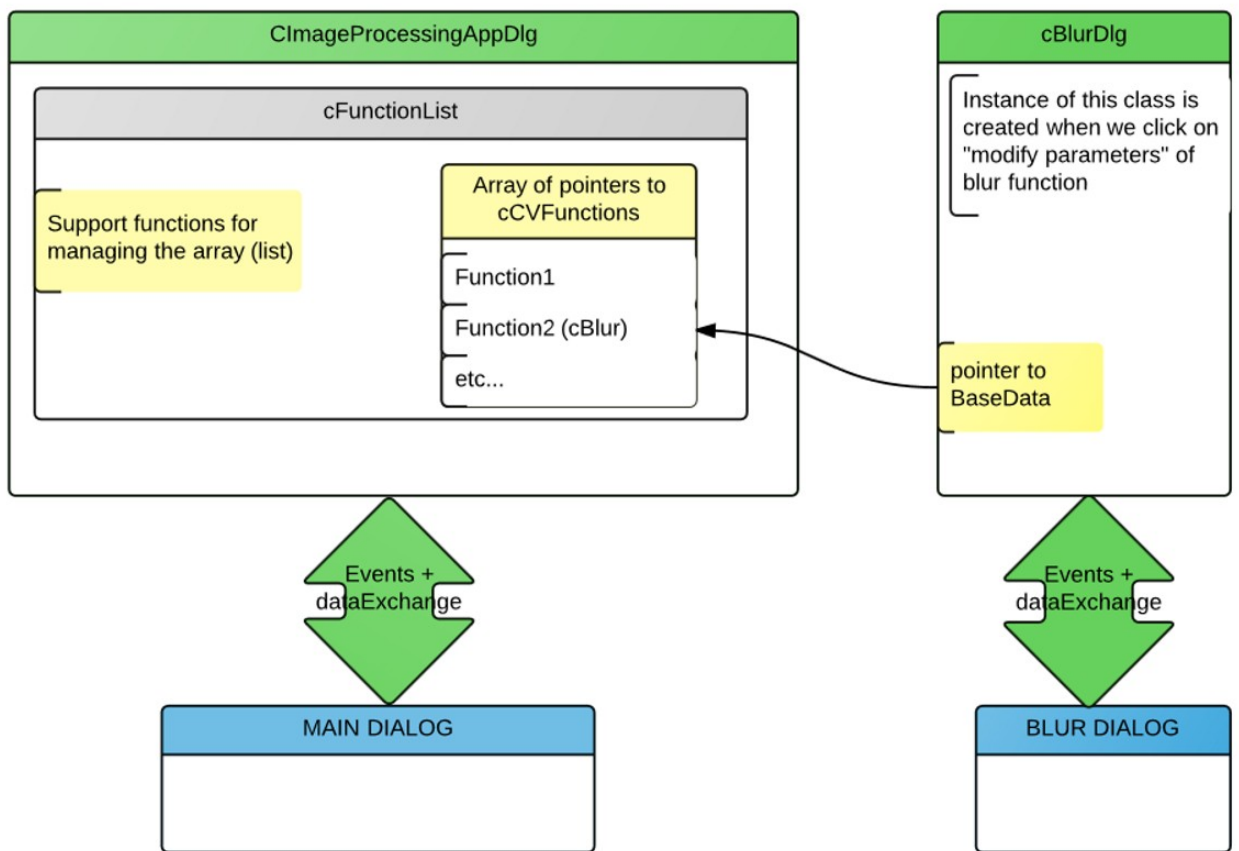


Figure 3: Overview of process - blur function

2.3.1 General classes

ImageProcessingApp – general entry class for application, it is generated by default, by MFC wizard

cCVFunction – base class for all Function classes in chapter 2.3.3. Prescribes general methods that all derived class must implement. For example every function has *Call()*, *getFunction()* and *Serialize()* methods and the source and the destination images.

cDataTemp – static class that serves as a storage, where the source, the destination and the temporary images are contained

cFunctionList – class for manage array (or list) of OpenCV functions contained in it. Several sorting methods and for example *AddFunction()* and *Call()* methods.

cMeasurementCalc – class that serves as a storage, where data from user mouse actions and few calculation constants are contained. For data exchange between *CimageProcessingAppDlg* and *cMeasurementDlg*.

2.3.2 General dialog classes

CimageProcessingAppDlg – main dialog class, contains all events of main window dialog. It also contains instances one of from each type: *cFunctionList*, *cMeasurementCalc*

cChooseFunctionDlg – dialog class for dialog where user can choose OpenCV function that will be added to project list.

cMeasurementDlg – dialog class of dialog that is used for calibration and then measuring in image in millimeters. In this class, timer is set up, and at every tick it checks instance of *cMeasurementCalc* for user mouse actions.

2.3.3 Function classes

Here is listed function classes specific to each OpenCV function. Every following class is derived class from *cCVFunction*. Each class contains *Call()* method, where the function from OpenCV library is called. Each class contains also description of function, that is shown in choose function dialog.

cBlur – realizes normalized box filter called *blur*

cGaussianBlur – realizes Gaussian blur

cMedianBlur – realizes median blur

cCvtColor – realizes conversion from color space to gray-scale

cThreshold – realizes binary threshold

cCanny – realizes *canny* algorithm (edge detection)

cConCom – realizes connected component labeling function (*blob* detection)

cAdaptiveThreshold – realizes special variant of threshold

2.3.4 Parameters dialog classes

Parameters dialogs belongs to each OpenCV function (see previous chapter) for setting up parameters – inputs to function. In every dialog the source and the destination image can be set. In every dialog is also possibility to apply changes immediately to see any changes on images during the adjustment. Here is the list of the dialogs:

cBlurDlg

cGaussianBlurDlg

cMedianBlurDlg

cCvtColorDlg

cThresholdDlg

cCannyDlg

cConComDlg

3 TESTING THE APPLICATION

In this chapter, we will show the capabilities of the developed application. The sub-chapter *General capabilities* describe project managing, saving and loading of an images, online help and language selection. Sub-chapter *Implemented functions* describes all OpenCV functions that can be used nowadays in our application. Further functions can be easily implemented – according to the small “tutorial” in chapter 5.

3.1 Overview of the application

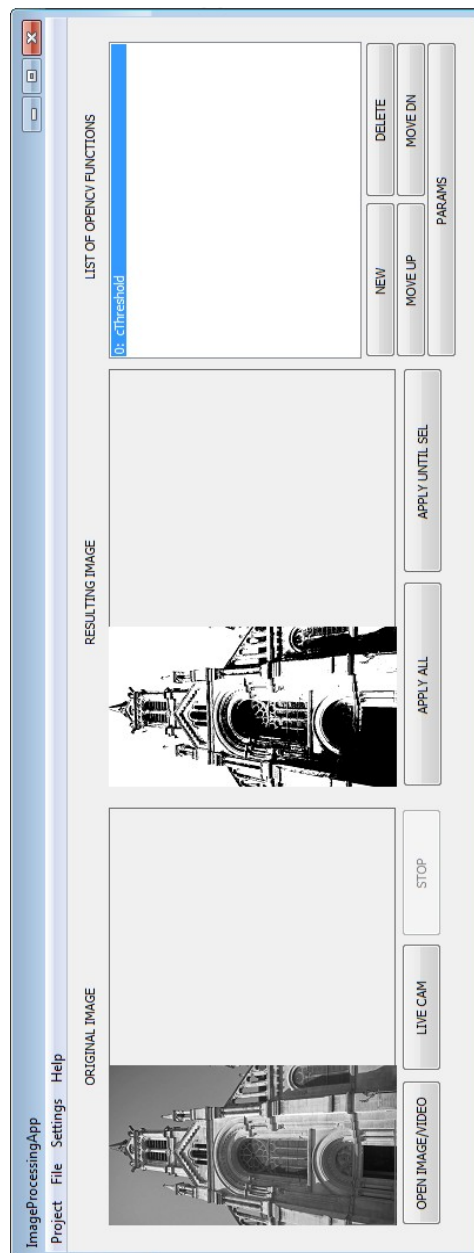


Figure 4: Main window

On a picture above, we can see main window of our application. There is several buttons and one menu. From menu we can manage project and file, change language and so on. Detailed descriptions of those functions are in next chapters. On the left part of the main window we can see a list, where we can see OpenCV functions that will be applied to our source image on left. We can manage order, create and delete the functions in list by use of a buttons located under that list. To adjust parameters of the specific function, just double click on name of function in the list or press the *PARAMS* button.

3.1.1 Project managing

Like every bigger application, even this one can load and save data from and to project file. We can create a new project – it will erase all what is on screen – the loaded image and the list of functions.

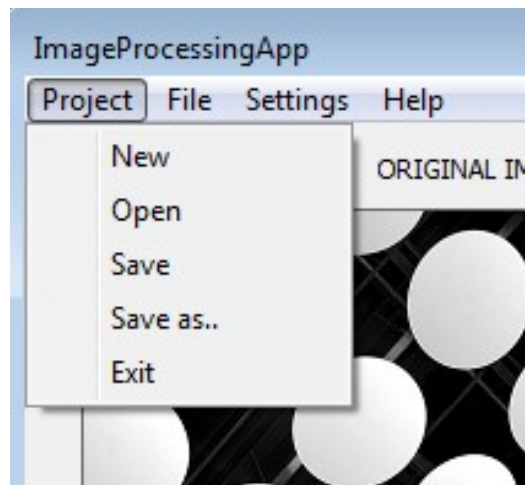


Figure 5: Project pop-up menu

Open and save commands is done by serializing the actual instance classes to a project file. We are also using special file to keep information about recent project path (the last opened project). That file is called *recentProject.txt* and it is located in application folder.

Here is the code from function `void CImageProcessingAppDlg::OpenProject(CString path)` that loads data from project file located in *path*.

```

void CImageProcessingAppDlg::OpenProject(CString path){
    CFile file;
    cCVFunction* p_fcion;
    if (!file.Open(path, CFile::modeRead)){
        MessageBox(L"Error loading project file! path:"+path);
        return;
    }
    CArchive ar(&file, CArchive::load);
    int nOfFcions;
    ar >> nOfFcions;//load number of functions to var
    ar >> recentFile;//load information about last opened image
    ar >> measurementCalc.mmPxRatio; // load last ratio for calculation
    for (int i = 0; i < nOfFcions; i++)
    {
        p_fcion = cCVFunction::Serialize_(ar);//we will call this
function to get instance
        functionList.functionList.Add(p_fcion);//add instance to list
    }
    ar.Close();
    file.Close();

    listBox.ResetContent();//update listBox
    functionList.WriteToListBox(&listBox);

    if (recentFile != _T("")){//show the source image from project on screen
        CT2CA pszConvertedAnsiString(recentFile);
        // construct a std::string using the LPCSTR input
        std::string strStd(pszConvertedAnsiString);
        src = cv::imread(strStd);
        if (!src.empty())ShowImg(src, 0);
    }

    //save last opened project path to txt file (when program again runs, that
    project he will try open)
    CFile file2;
    file2.Open(L"recentProject.txt", CFile::modeCreate | CFile::modeWrite);
    CArchive ar2(&file2, CArchive::store);

    ar2 << recentProject;

    ar2.Close();
    file2.Close();
}

```

Figure 6: Code of method for open project routine

3.1.2 Load/save image

Loading and saving an images is done by simple OpenCV functions *imwrite()* and *imload()*.

```
void CImageProcessingAppDlg::OnImageOpen()
{
    CFileDialog dlg(TRUE); // TRUE is to tell the dialog
                           // is used as an open CFileDialog.

    if (dlg.DoModal() == IDOK)
    {
        CString fullPathName = dlg.GetPathName(); // get the full path name of the
selected file.
        recentFile = fullPathName;//save this path for later use

        // Convert a TCHAR string to a LPCSTR
        CT2CA pszConvertedAnsiString(fullPathName);
        // construct a std::string using the LPCSTR input
        std::string strStd(pszConvertedAnsiString);

        src = cv::imread(strStd);

        //clear both windows, we are changing dimensions of images
        EraseWindow(0);
        EraseWindow(1);

        if (!src.empty())ShowImg(src, 0);
    }
    SetMenu(GetMenu());
    dlg.DestroyWindow();
}
```

Figure 7: Code for loading image from file to instance container *cv::Mat*

3.1.3 Other

In the main menu, there is an option to switch language from English to German. Labels for buttons and menus are always in English, only the descriptions of *OpenCV* functions are bi-lingual.

This is programmed like following: menu items sets or resets *boolean* variable. This variable is given to *Choose dialog* (the one which pops up when user want to add new function) and this dialog draw either English or German string in the description dialog.

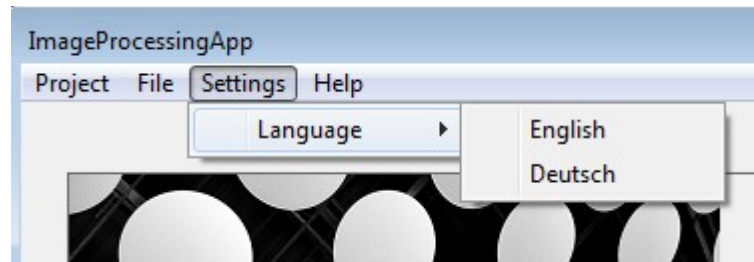


Figure 8: Language selection menu

There is also possible to open online OpenCV documentation [15] from Help→Online documentation. Default browser will open this website.

3.2 Implemented functions

In this chapter, there will be presented available OpenCV functions with a print-screens from the application.

3.2.1 Threshold

There is a print-screen of our application(the main window), that applies one of the possible functions.

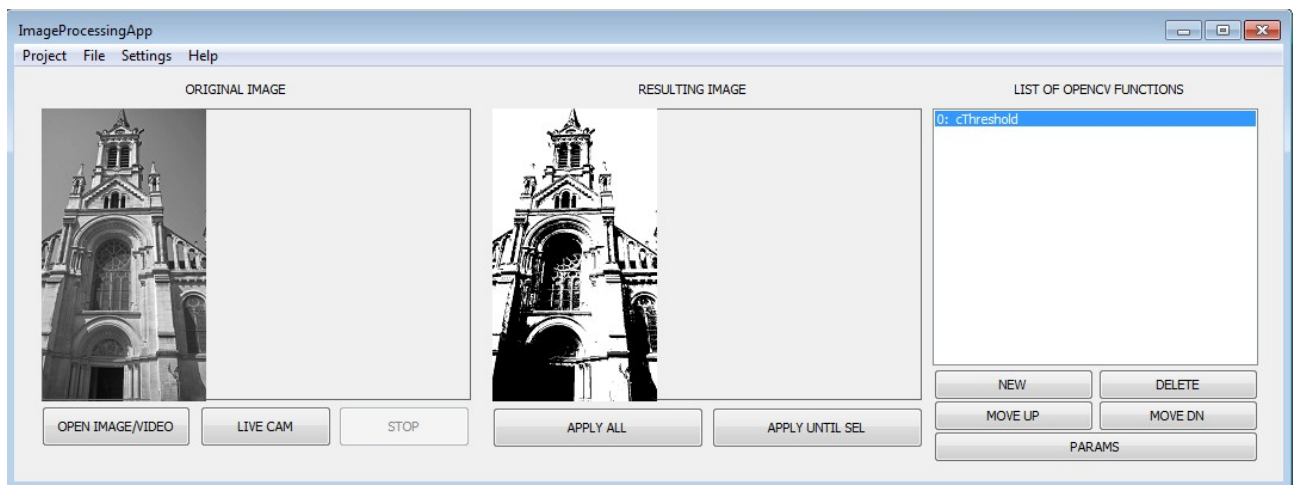


Figure 9: Main window when threshold function is applied

In following dialog we can set all input parameters of that function. In the right upper corner we can set also the source and the destination image and the temporary bitmap containers named 0 to 9.

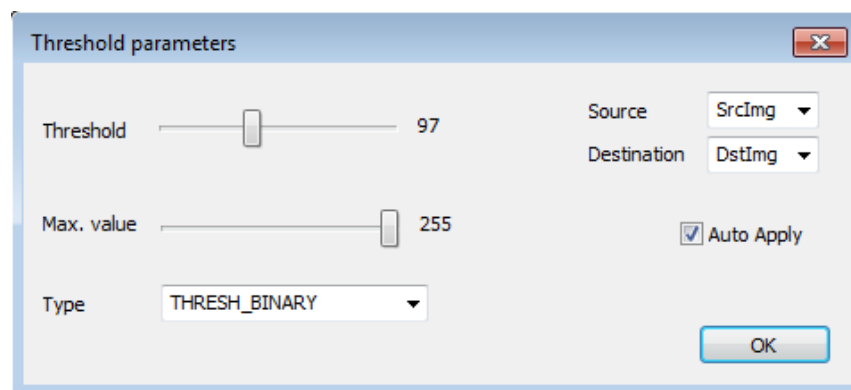


Figure 10: Parameters window for threshold function

This function has two parameters: threshold and maximal value. In the type combo-box we can choose specific type of binary thresholding. Overview of those types is on next picture.

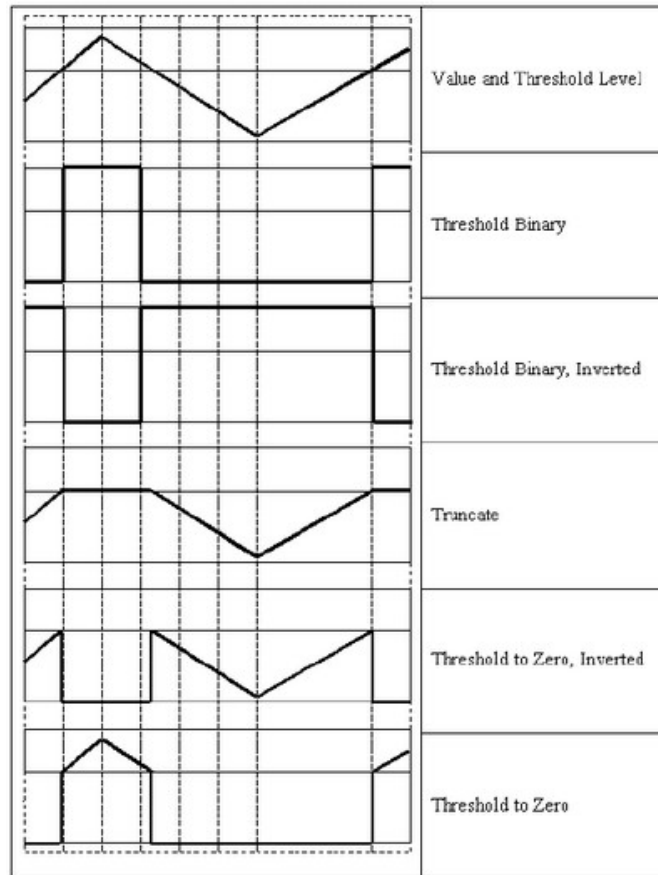


Figure 11: List of threshold types, taken from [16]

3.2.2 Blur

There is a print-screen of our application(the main window), that applies one of the possible functions. Blur is one of possible OpenCV smoothing functions. Smoothing functions are typically used for removing a different types of noises from images.

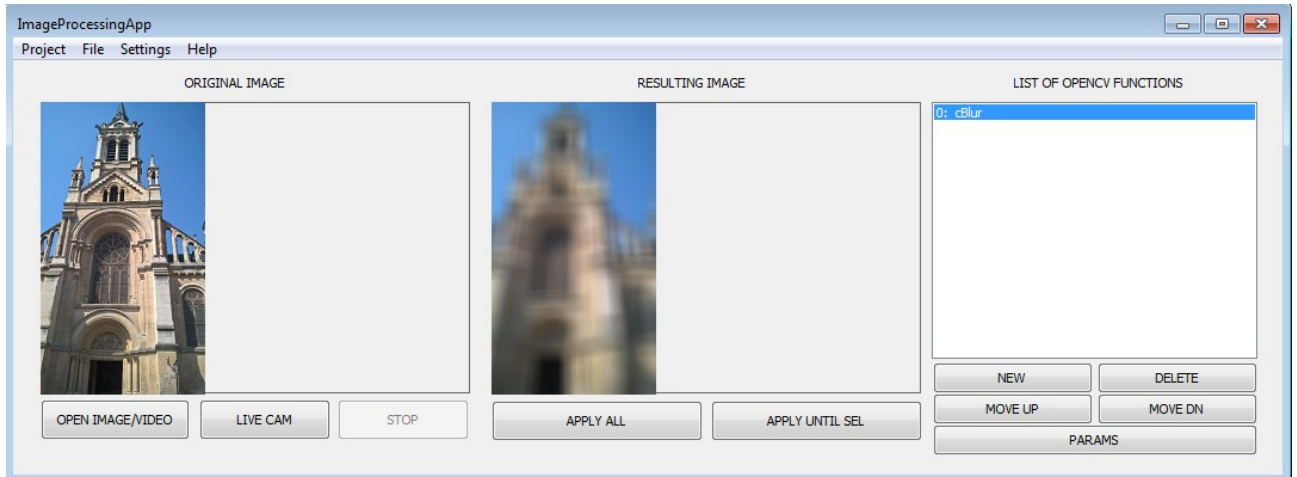


Figure 12: Main window when blur function is applied

In following dialog we can set all input parameters of that function. In the right upper corner we can set also the source and the destination image and the temporary bitmap containers named 0 to 9.



Figure 13: Parameters window for blur function

There is also two parameters. Anchor components (x and y) must be always less then components of k-size. It is given by the principle.

3.2.3 Median blur

There is a print-screen of our application(part of the main window), that applies one of the possible functions. Median blur is one of possible OpenCV smoothing functions. Smoothing functions are typically used for removing a different types of noises from images.

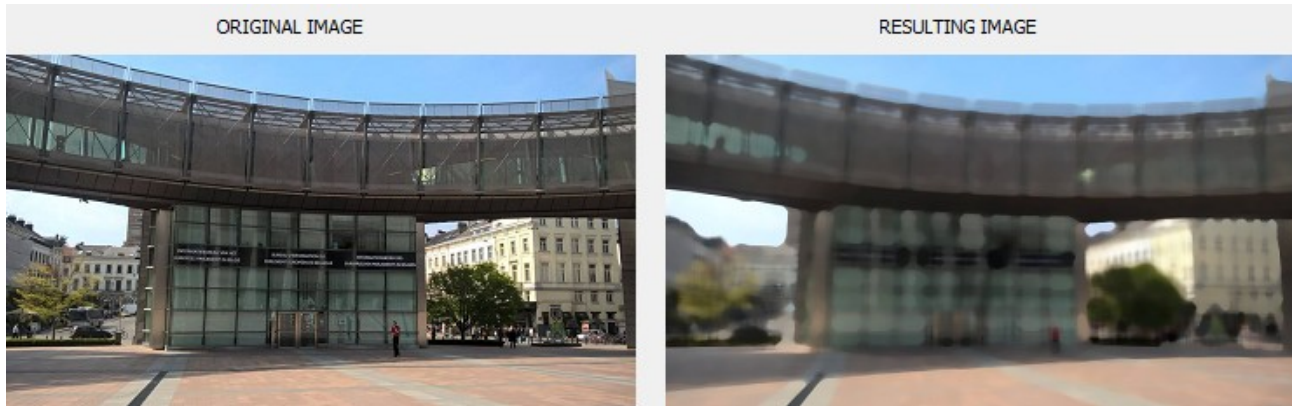


Figure 14: Main window when median blur function is applied

In following dialog we can set all input parameters of that function. In the right upper corner we can set also the source and the destination image and the temporary bitmap containers named 0 to 9.

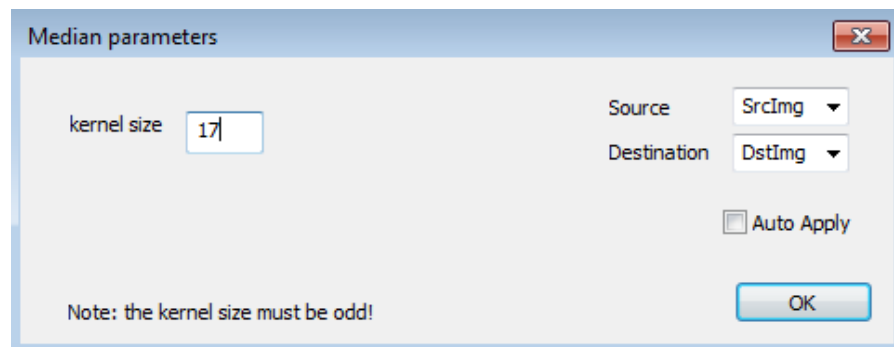


Figure 15: Parameters window for median blur function

The kernel size is only parameter of this function. This must be odd, otherwise we get an error while executing the function.

3.2.4 Gaussian Blur

There is a print-screen of our application(part of the main window), that applies one of the possible functions. Gaussian blur is one of possible OpenCV smoothing functions. Smoothing functions are typically used for removing a different types of noises from images.

Gaussian smoothing is commonly used with edge detection. Most edge-detection algorithms are sensitive to noise; the 2-D Laplacian filter, built from a discretization of the Laplace operator, is highly sensitive to noisy environments. Using a Gaussian Blur filter before edge detection aims to reduce the level of noise in the image, which improves the result of the following edge-detection algorithm. This approach is commonly referred to as Laplacian of Gaussian, or LoG filtering. This paragraph is taken from [Fisher, Perkins, Walker & Wolfart (2003). "Spatial Filters - Laplacian of Gaussian". Retrieved 2010-09-13.]

Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function [17].

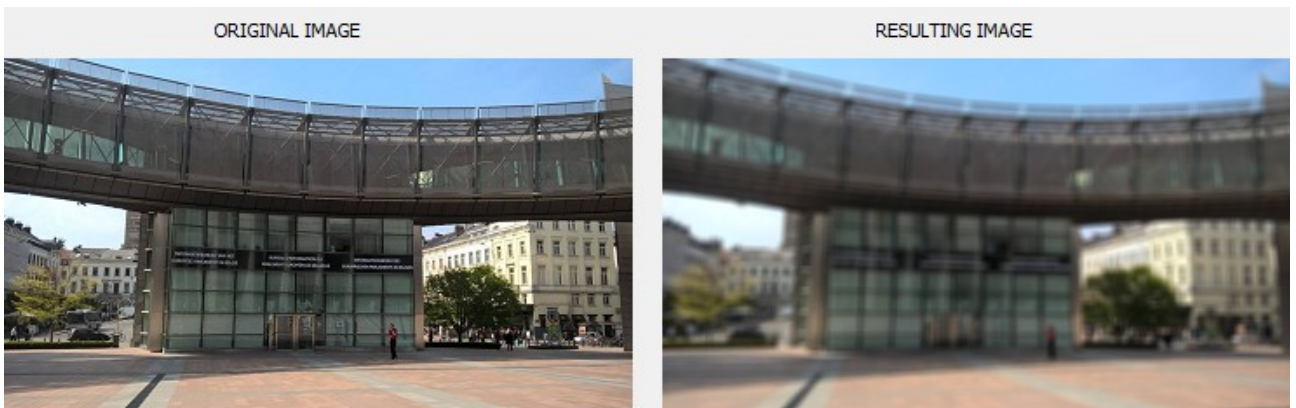


Figure 16: Main window when gaussian blur function is applied

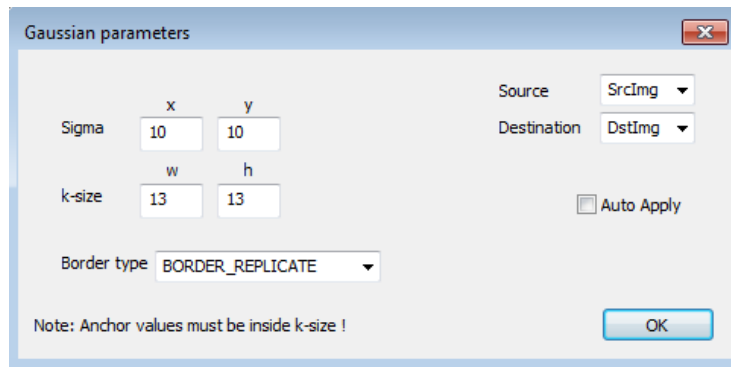


Figure 17: Parameters window for gaussian function

In dialog on parameters dialog we can set all input parameters of that function. In the right upper corner we can set also the source and the destination image and the temporary bitmap containers named 0 to 9. We can also set up parameters sigma, k-size and border type.

3.2.5 CvtColor

There is a print-screen of our application(main window), that applies one of the possible functions. CvtColor is abbreviation of Convert Color which is one of possible OpenCV functions for converting between color spaces. More specific, this function just convert color image to gray scale image. No other conversions is supported. This color to gray conversion is the most common in image processing.

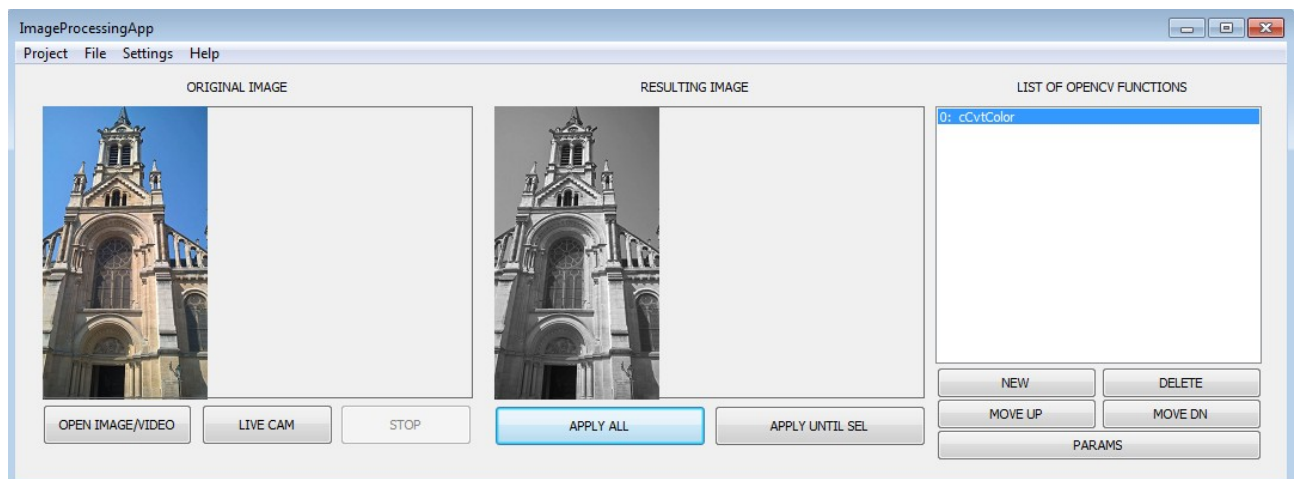


Figure 18: Main window when cvtColor function is applied

In following dialog we can set just the source, the destination and the temporary bitmap containers named 0 to 9. No other parameters can be changed.

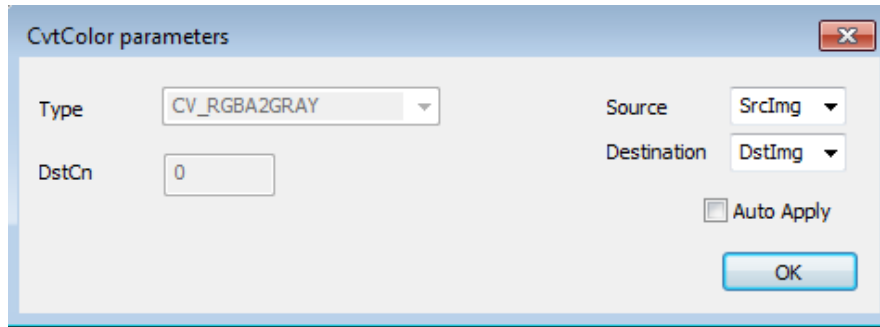


Figure 19: Parameters window for convert to gray function

3.2.6 Canny edge

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986 [18].

The algorithm can be split to five steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

All this handles function of OpenCV library. Our task is just to call it. On next images we can see print-screen of main window and parameters of function.



Figure 20: Main window when Canny function is applied

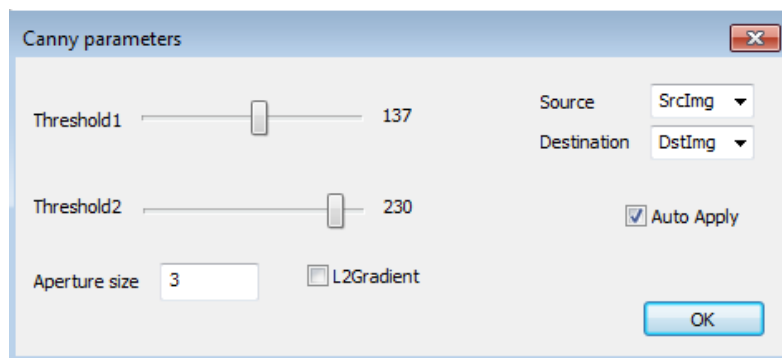


Figure 21: Parameters window for canny function

3.2.7 Connected components labeling

This function is special and it is not implemented directly in *OpenCV* library. Inside our custom algorithm (see code for details) is used function *floodFill* and that is part of *OpenCV* library.

In general, connected components labeling colors regions that is separated of each other with different colors. Following print-screen shows picture of some grid. On resulting image we can see that separated areas have different colors. On upper part of image is grid getting lost, so there is common color.

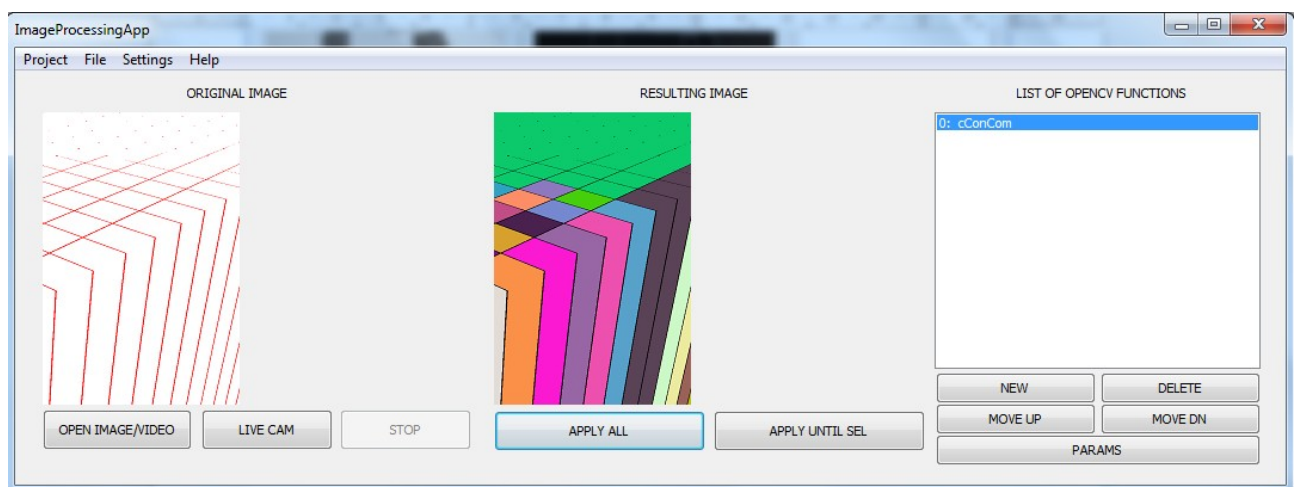


Figure 22: Main window when ConCom function is applied

There are no parameters except source and destination image for connected components labeling function.

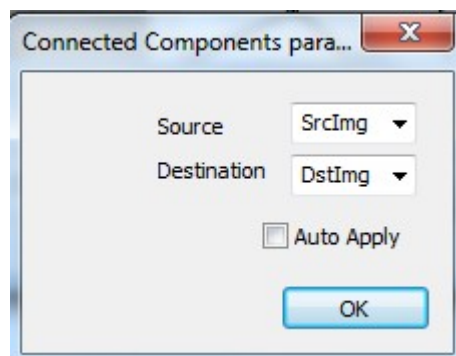


Figure 23: Parameters window for connected components labeling function

3.2.8 Adaptive threshold

For images, that does not have ideal illumination (not homogenous) we can use adaptive variant of threshold function.

This function calculates threshold for every region in image. The types are following:

- ADAPTIVE_THRESH_MEAN_C : threshold value is the mean of neighborhood area.
- ADAPTIVE_THRESH_GAUSSIAN_C : threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.

The Block Size decides the size of neighborhood area. The constant C is subtracted from the mean or weighted mean calculated.

The print-screen of main window and the parameter window are shown on next two pictures.

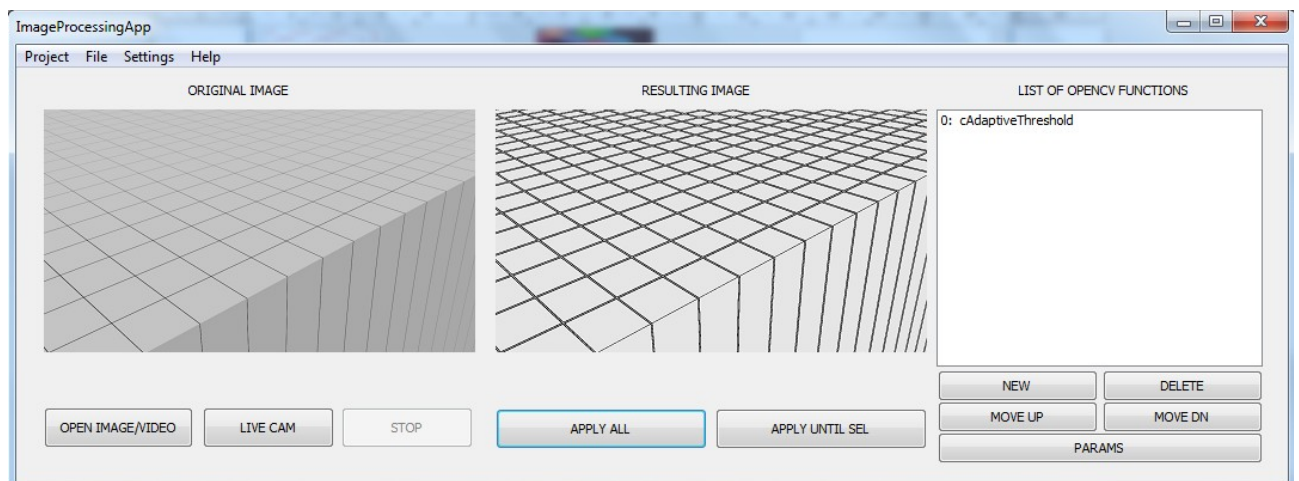


Figure 24: Main window when adaptive threshold function is applied

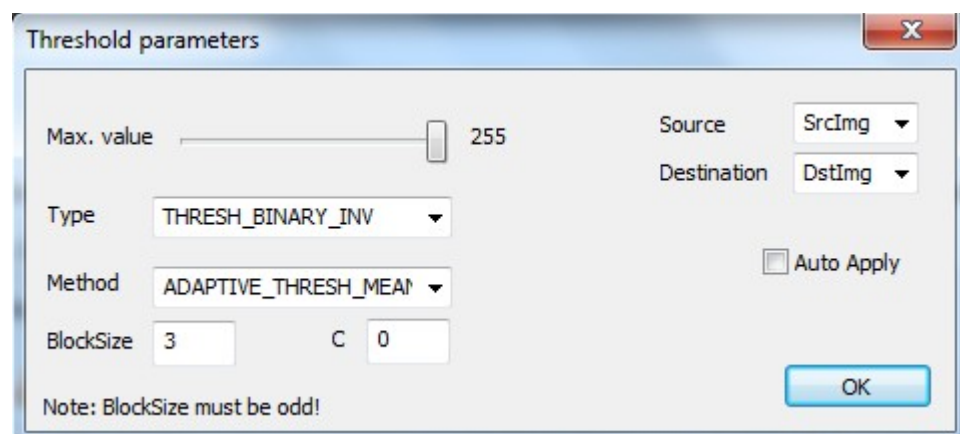


Figure 25: Parameters window for adaptive threshold function

3.3 Combining of functions

Every function has own source and the destination image. We can choose assignation in a combo-box in the function's parameter dialog.

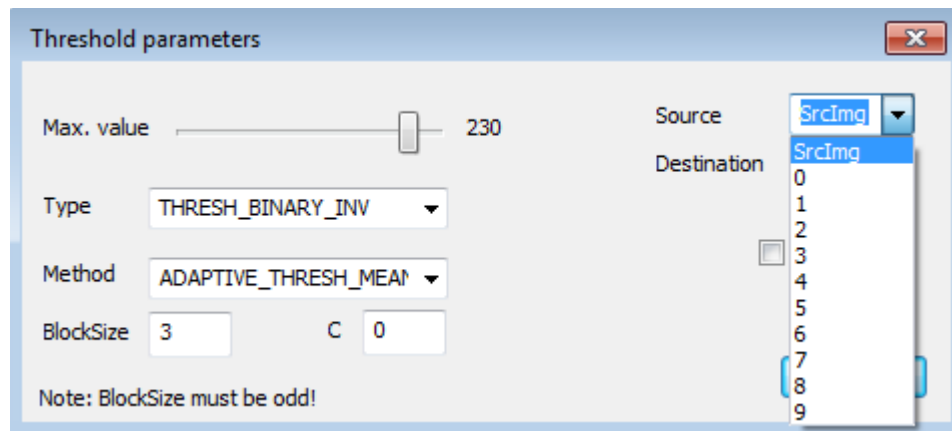


Figure 26: Determining source and destination images the functions.
After click there is list for choose

SrcImg and *DstImg* are clear, that's the images shown in the main frame. The numbers 0 to 9 represents temporary image containers for our custom purpose.

We can use multiple of functions to a single source image easily with the advantage of temporary bitmap containers – explained in table.

	Source Image	Destination Image
<i>Function1</i>	SrcImg	0
<i>Function2</i>	0	0
<i>Function3</i>	0	DstImg

This is how we can use more than one function at the same time. We can do also more complex structures for example – we create mask of some image, we store it to image container for example no. 5 and then do other image processing with original image, and then use mask stored in image container no. 5 to our actual image and then we put it to *DstImg*.

Such a structure for example combination adaptive threshold – connected components labeling – median blur is shown on next three print screens. We use a special button “APPLY UNTIL SEL” which shows us intermediate steps.

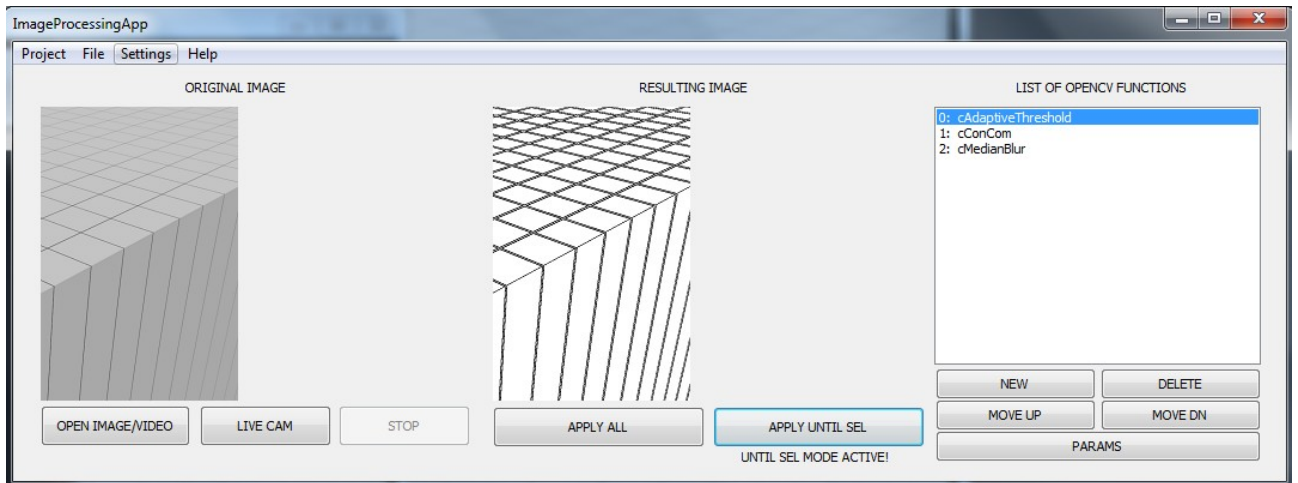


Figure 27: Applied sequence - step 1

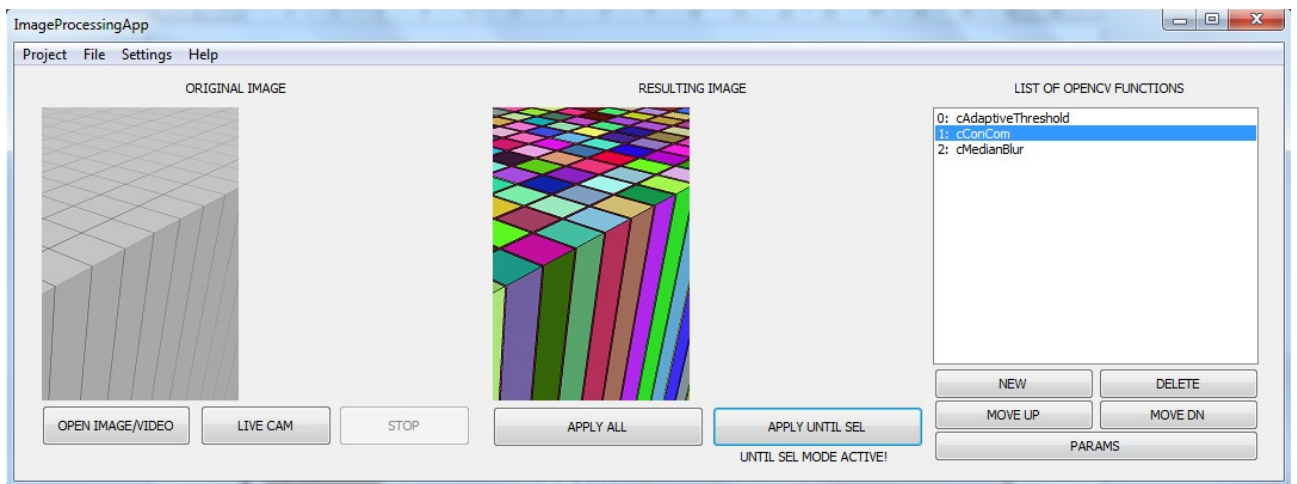


Figure 28: Applied sequence - step 2

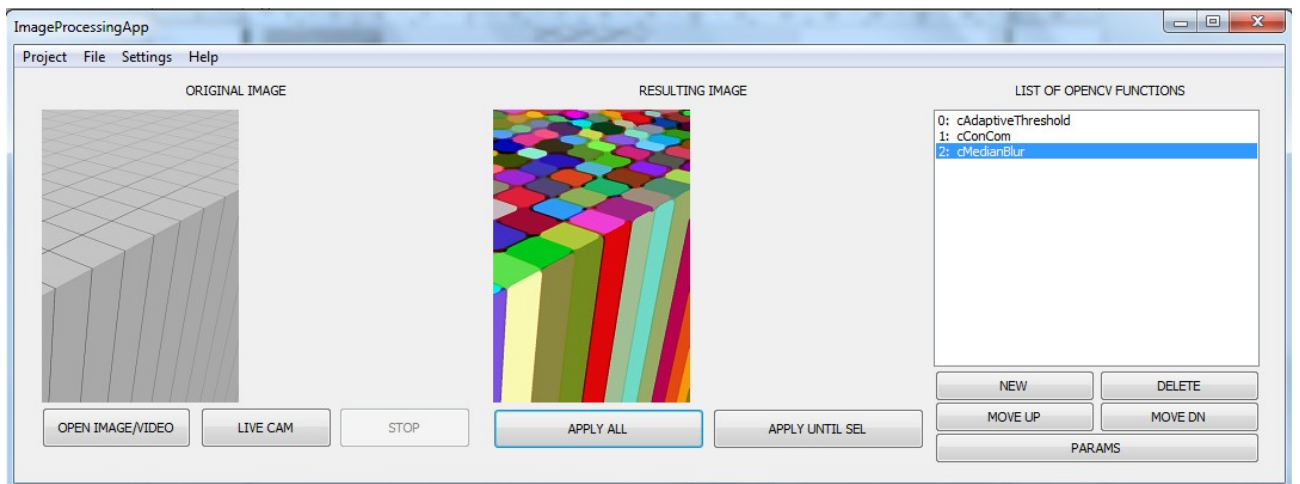


Figure 29: Applied sequence - step 3

4 VIDEO SUPPORT

4.1 Difference between image and video input

In the *OpenCV*, difference between a static image and a video is, in fact, really small. A video is at the end just a sequence of images. *OpenCV* library offers us simple function for grabbing the image from a video device.

So to support the video we need to add few events to our application for handling a video device connection. The rest of processing is the same as for the image loaded from disk.

4.2 Implementation

We must modify the code in function `void CimageProcessingAppDlg::OnImageOpen()` described in chapter “3.1.1 Project Managing”

```
//we will check if we opened video file
if (fullPathName.Find(L".avi", 0) > 0 || fullPathName.Find(L".mp4", 0)
> 0){
    videoFile = true;
    cap.release();
    StartTimer();
    return;
}
else videoFile = false;
```

Figure 30: Simple modification of code for detecting that file is video type file

The variable *videoFile* is global flag for keeping information if we are working with image or a video file. This method is called when user clicks on open image/video button. Detection that we are opening video is by checking the file extension.

```

void CImageProcessingAppDlg::OnTimer(UINT_PTR nIDEvent){
    if (!cap.isOpened()) {
        // Convert a TCHAR string to a LPCSTR
        CT2CA pszConvertedAnsiString(recentFile);
        // construct a std::string using the LPCSTR input
        string strStd(pszConvertedAnsiString);

        if (videoFile)cap.open(strStd);
        else cap.open(0);
    }
    Mat frame; // Current Frame

    cap >> frame; // get first frame for size

    frame.copyTo(src);
    ApplyMultiFunctions(false);
    if (!src.empty())ShowImg(src, 0);
}

```

Figure 31: Timer for grabbing images from webcam or video file stream

The timer method is called periodically. The purpose is to serve as a grabber.

On the following two pages are two print-screens of the application while live capturing from web camera on a notebook is active. First print-screen shows us use of multiple functions to detect edges in video stream from web camera. The second shows us use of adaptive threshold function to video stream.



Figure 32: Sequence for edge detection in live stream

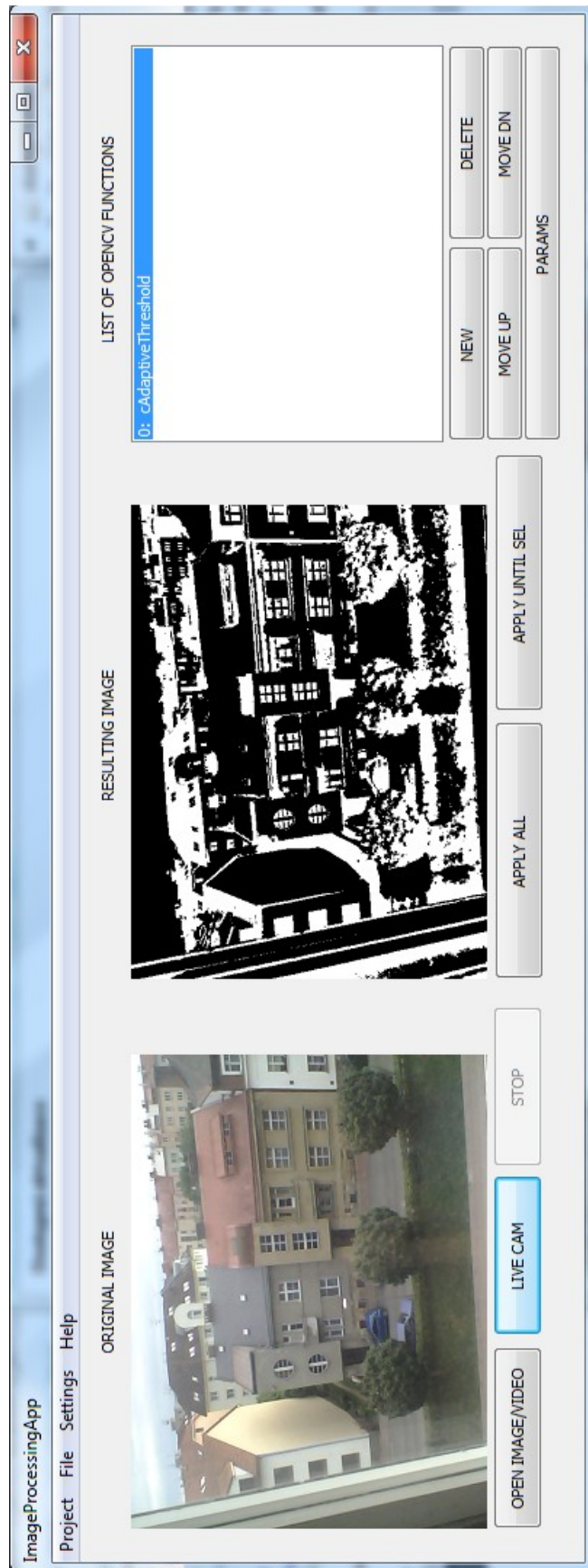


Figure 33: Adaptive threshold applied to live stream

5 NEW FUNCTIONS

In this chapter user, that need to add support of new *OpenCV* function that is not yet implemented, will find recipe how to do that. For example we need to add new function to the list of functions in dialog that will appear after user click “new function”. Then also add dialog for modifying parameters of new function (unique to each of function).

On the picture we can see all nowadays implemented *OpenCV* functions – its the dialog for choosing type of new added function.

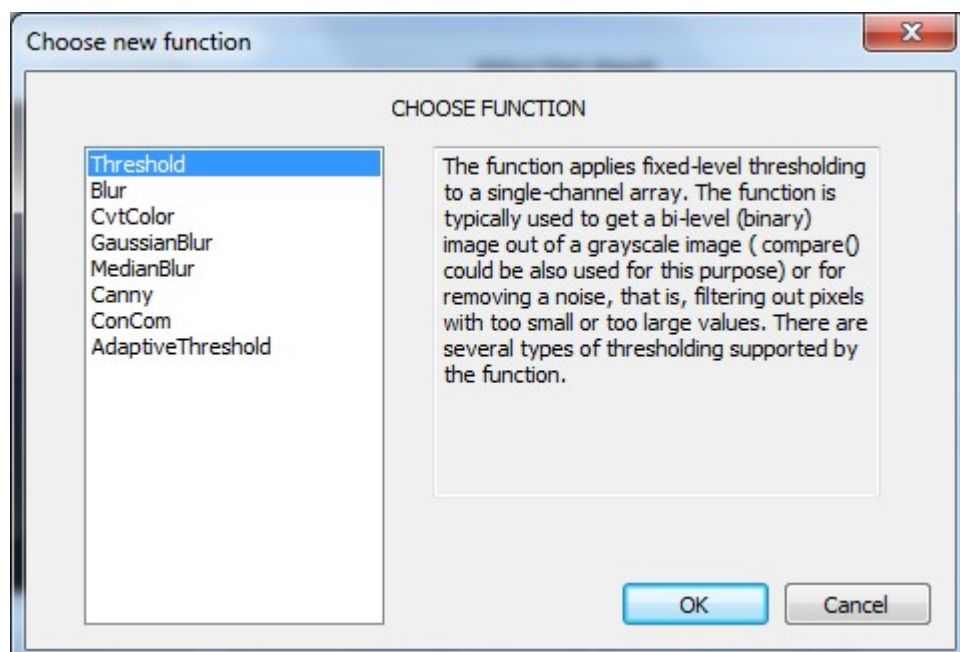


Figure 34: Dialog, where we can choose function for adding to project

Complete guide how to add support of new *OpenCV* function is following:

1. In the project tree, folder function classes → press right button → add class, name it c[describing_name]
2. In the project tree, folder dialog classes → press right button → add class, name it c[describing_name]Dlg

3. In the Resources, make a copy of most similar dialog, rename to IDD_DIALOG_[describing_name]
4. Replace all the code from most similar cpp and the header files to our new files (for example: from cThreshold.cpp to cAdaptiveThreshold.cpp and from cThresholdDlg.cpp to cAdaptiveThresholdDlg.cpp and the two header files)
5. Use function “replace” for example like this: Threshold replace with AdaptiveThreshold
6. Modify in [describing_name]Dlg.cpp the IDD to correct name from point 3.
7. Modify *OpenCV* function calling or comment it for now
8. Add new code to Serialize_ in cCVFunction.cpp and also to that file add new header include on to top
9. Add new code to method AddFcion() in file cFunctionList.cpp and also to that file add new header include on to top
10. Add new code to methods OnActivate() and OnBnSelChangeList() in the file cChooseFunctionDlg.cpp and also to that file add new headers include on to top
11. Add new code to method OnBnClickedParameters() in file ImgProcAppDlg.cpp and also to that file add new header include on to top
12. Finalize the new Parameter Dialog appearance, code for data exchange and call of *OpenCV* functions
13. Add English/German description of new function to c[describing_name].cpp

6 CONCLUSION

We were successful in meaning of task fulfilling and more than that. Creating any application with use of Microsoft Foundation Classes is not that simple as with *.NET* can be. In *MFC*, we must take care for a declarations and also for proper header files connections. We had to write every event handler for every component in our application and there are over eighty components. That, on the other hand, forces us before writing something to think about the structure. Object oriented concepts as inheritance and polymorphism are powerful tools if they are used in proper way. We can say, that we were able to use it with big advantage – this can be seen in chapter 5 where is described how to add support of new *OpenCV* functions. That is much less complicated than it could be without use of that concepts.

Created application can work with an images and save a results of our experiments. It can, and it can be used as an educational tool to provide to all students (not just for those, who can program with use of a text programming) better understanding of possibilities of the *OpenCV* library.

Furthermore, with use of this paper work, the adding new features can be done without too much effort.

7 RESOURCES

[1] PLÍVA, Z., J. DRÁBKOVÁ, J. KOPRNICKÝ a L. PETRŽÍLKA. *Metodika zpracování bakalářských a diplomových prací*. 2. upravené vydání. Liberec: Technická univerzita v Liberci, FM, 2014. ISBN 978-80-7494-049-1. Available from: doi:10.15240/tul/002/2014-11-002

[2] OpenCV. *OpenCV home page*. [online]. © 2015- [cit. 2015-08-03]. Available from: <http://opencv.org/>

[3] Windows Programming/Microsoft Foundation Classes. *Wikibooks*. [online]. 9.8 2013 [cit. 2015-06-24]. Available from: https://en.wikibooks.org/wiki/Windows_Programming/Microsoft_Foundation_Classes

[4] Installing & Configuring with Visual Studio. *OpenCV Tutorial C++*. [online]. [2015] [cit. 2015-08-03]. Available from: <http://opencv-srf.blogspot.cz/2013/05/installing-configuring-opencv-with-vs.html>

[5] Reading and Writing Files. *Microsoft Developer Network*. [online]. [2015] [cit. 2015-08-03]. Available from: <https://msdn.microsoft.com/en-us/library/6337eske.aspx>

[6] Polka Dots picture. *fine art america*. [online]. [2015] [cit. 2015-08-03]. Available from: <http://fineartamerica.com/featured/polka-dots-ann-horn.html>

[7] Image Thresholding. *OpenCV online Docs*. [online]. [2015] [cit. 2015-08-03]. Available from: http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html#gsc.tab=0

[8] Introduction to Dialog Boxes. *Functionx*. [online]. © 2010-2015 [cit. 2015-08-03]. Available from: <http://www.functionx.com/visualc/Lesson04.htm>

[9] OpenCV example. *OpenCV example*. [online]. © 1997-2015 [cit. 2015-08-03]. Available from: <http://lh2078.blog.163.com/blog/static/5681137220131311492631/>

[10] OpenCV study. *Mate computer vision*. [online]. © 1997-2015 [cit. 2015-08-03]. Available from: <http://study.marearts.com/2014/03/opencv-study-iplimage-display-to.html>

[11] Using CStatic control. *Code Project*. [online]. © 1999-2015 [cit. 2015-08-03]. Available from: <http://www.codeproject.com/Articles/807/Using-the-CStatic-control>

[12] X button. *Microsoft developer network*. [online]. © 2015 [cit. 2015-08-03]. Available from: <https://social.msdn.microsoft.com/Forums/en-US/67d9fc46-a17f-4de8-a3a5-ad194543aa36/x-button-and-menu-bar-grayed-after-calling-cfiledialog?forum=vcmfcatl>

[13] Serialization. *Functionx*. [online]. © 2015 [cit. 2015-08-03]. Available from: <http://www.functionx.com/visualc/fileprocessing/serialization.htm>

[14] Reading and writing files. *Microsoft developer network*. [online]. © 2015 [cit. 2015-08-03]. Available from: <https://msdn.microsoft.com/en-us/library/6337eske.aspx>

[15] OpenCV docs. *OpenCV online Docs*. [online]. © 2015 [cit. 2015-08-03]. Available from: <http://docs.opencv.org/index.html>

[16] OpenCV modules. *OpenCV docs*. [online]. © 2015 [cit. 2015-08-03]. Available from: http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html?highlight=threshold#cv2.threshold

[17] Gaussian blur. *Wikipedia*. [online]. 3.8.2015 [cit. 2015-08-03]. Available from: https://en.wikipedia.org/wiki/Gaussian_blur#cite_note-4

[18] Canny edge detector. *Wikipedia*. [online]. 30.7.2015 [cit. 2015-08-03]. Available from: https://en.wikipedia.org/wiki/Canny_edge_detector