Chapter 2
Image Pre-processing

2.1 Image Acquisition

Image acquisition was carried out in the Corvis tonometer. The results can be exported to various formats directly from the original software (OCULUS Corvis ST ver 1.02r 1126). The following formats have been used in practice:

- *U12*—a compressed file with a database of patients,
- *CST*—a compressed file of one patient,
- **.avi*—an AVI file with a sequence of images of the cornea,
- **.jpg*—a sequence of images in jpg format from *patient's_name_date_000.jpg* to *patient's_name_date_139.jpg*.

A block diagram of the data acquisition and connection, through file conversion, with Matlab is shown in Fig. 2.1.

Later in the monograph the operation of the algorithm individual parts was verified using the Operating System: Microsoft Windows 7 Version 6.1 (Build 7601: Service Pack 1) Java VM Version: Java 1.6.0_17-b04 with Sun Microsystems Inc. Java HotSpot(TM) 64-Bit Server VM mixed mode in Matlab Version 7.11.0.584 (R2010b) with Image Processing Toolbox Version 7.1 (R2010b). The PC computer was equipped with Intel Xeon X5680 @ 3.33 GHz with 12 GB of RAM.

The last two data formats (**.avi* and **.jpg*) are most convenient for data processing. Loading and showing a sequence of images previously saved in the folder *C:/data* in **.jpg* format to Matlab can be done using the following code:

R. Koprowski, *Image Analysis for Ophthalmological Diagnosis*,
Studies in Computational Intelligence 631, DOI 10.1007/978-3-319-29546-6_2

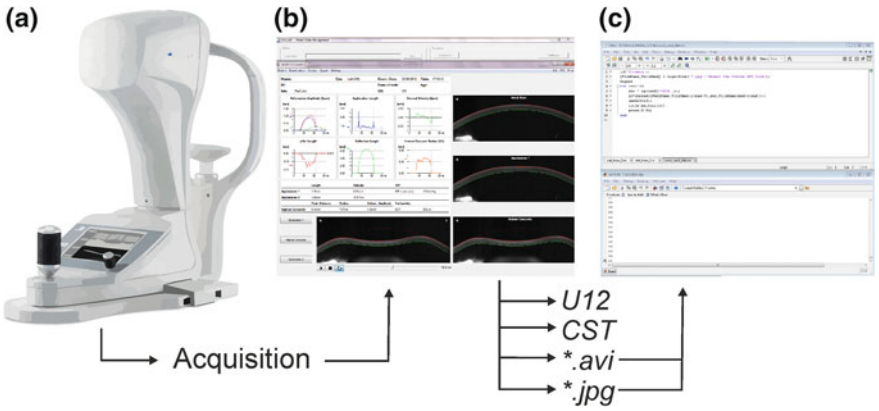

Fig. 2.1 Block diagram of the data acquisition and connection, through file conversion, with Matlab: **a** Corvis tonometer, **b** application OCULUS Corvis ST, **c** Matlab with the algorithm discussed in this monograph

```
cd('C:/data');
[FileName,PathName] = uigetfile('*.jpg','Select the
Corvis JPG file');
figure
for i=0:139
    str = sprintf('%03d',i);
    LG=imread([PathName,FileName(1:end-
7),str,FileName(end-3:end)]);
    imshow(LG);
    title(mat2str(i))
    pause(0.01)
end
```

Function *cd* is designed to set the path, *uigetfile* function enables to open a dialog where the user can select any **.jpg* file from the image sequence. The file selection is followed by sequential reading of successive *jpg* images from $i = 0$ to $i = 139$. With the use of *sprintf* function it is possible to convert the value i to an appropriate string form, "000" instead of "0", "001" instead of "1" etc. *Imread* function enables to load the appropriate file to the matrix L_G. Next, functions title and *mat2str* enable to show the image number in its header. With *pause* function it is possible to delay the loop operation due to the need to display individual images from a sequence.

In the case of *.*avi* file, the following notation can be used:

```
cd('C:/data');
[FileName,PathName] = uigetfile('*.avi','Select the Cor-
vis AVI file');
Obj = mmreader([PathName,FileName]);
figure
for i=1:Obj.NumberOfFrames
    LGi = read(Obj, i);
    imshow(LGi);
    title(mat2str(i))
    pause(0.01)
end
```

Function *mmreader* (*aviread* in older versions of Matlab) enables to download the movie file handle. Then *read* function allows for loading individual frames as 2D images.

In this loop, and also in the previous one, it is possible to perform three basic tasks useful for further analysis:

- visualization of an image sequence—as it has been done in the presented source code,
- saving an image sequence as subsequent files with the extension *.*mat*,
- saving an image sequence as one file in the Matlab data format—*.*mat*.

The choice of the appropriate method depends on the direction of further analysis. As the essential elements in further analysis are the cornea and its contour, saving all the images on the disk in the form of one matrix is not necessary. Additionally, the creation of one matrix containing the full sequence of 140 images occupies in the Matlab space 16,128,000 bytes for data in *uint8* format ($M \times N \times I = 200 \times 576 \times 140$ pixels). Compared to a single 2D image, it is 115,200 bytes in *uint8* format ($M \times N = 200 \times 576$ pixels).

The subsequent images loaded into Matlab will be the variable L_G. The notation $L_G(m, n)$, where m—row coordinates, n—column coordinates, will be used equally often. Later in the monograph, other variables will be indexed at the bottom in accordance with the notation of the following image matrices for subsequent analysis phases. In the notation of source codes in Matlab, subscripts and superscripts will be replaced with normal letters. In addition, readers who will not use the *m*-files added to the monograph should pay attention to too long strings which were divided between the text lines. This division is important because the division between the lines in the source code in Matlab is marked with an ellipsis "…". However, it is not included in this monograph.

2.2 Image Filtration

Immediately after loading the image L_G into Matlab, noise and small artefacts were filtered. Initial analyses confirmed that the greatest distortions, when taking proper care of the tonometer optics (its purity), are clusters consisting of no more than 3, 4 pixels. In this case, the median filter is the best option due to its characteristics, mainly the ability to remove impulse noise. The available function representing the median filter is *medfilt2*. In addition, the size of the mask h_1 is given, namely $M_{h1} \times N_{h1}$. This is the size of the window within which the median will be calculated. Since, as mentioned earlier, distortions are clusters that do not exceed 4 pixels, the sufficient size of the median filter mask is $M_{h1} \times N_{h1} = 3 \times 3$ pixels—because $4 < (M_{h1} \cdot N_{h1})/2$. In practice, this provides 9 pixels, from which the middle one is chosen after sorting. With *imnoise* function, it is possible to trace the relationship between the level of distortions (image L_{NO}) and the results obtained from median filtering—image L_{MED}. For a sample image for $i = 70$ and the selected area, $m \in (80, 120)$, $n \in (80, 120)$:

```
cd('C:/data');
[FileName,PathName] = uigetfile('*.jpg','Select the Cor-
vis JPG file');
i=70;
str = sprintf('%03d',i);
LG=imread([PathName,FileName(1:end-7),str,FileName(end-
3:end)]);
for d=0.01:0.2:0.61
    LNO=imnoise(LG,'salt & pepper',d);
    figure; imshow(LNO);
     axis([80 120 80 120])
    xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
    ylabel('m
[pixel]','FontSize',20,'FontAngle','Italic');
    LMED=medfilt2(LNO,[3 3],'symmetric');
    figure; imshow(LMED);
     axis([80 120 80 120])
    xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
    ylabel('m
[pixel]','FontSize',20,'FontAngle','Italic');
end
```

The value d is noise density and $d \in \{0.01, 0.21, 0.41, 0.61\}$. The results obtained can be traced in Fig. 2.2.

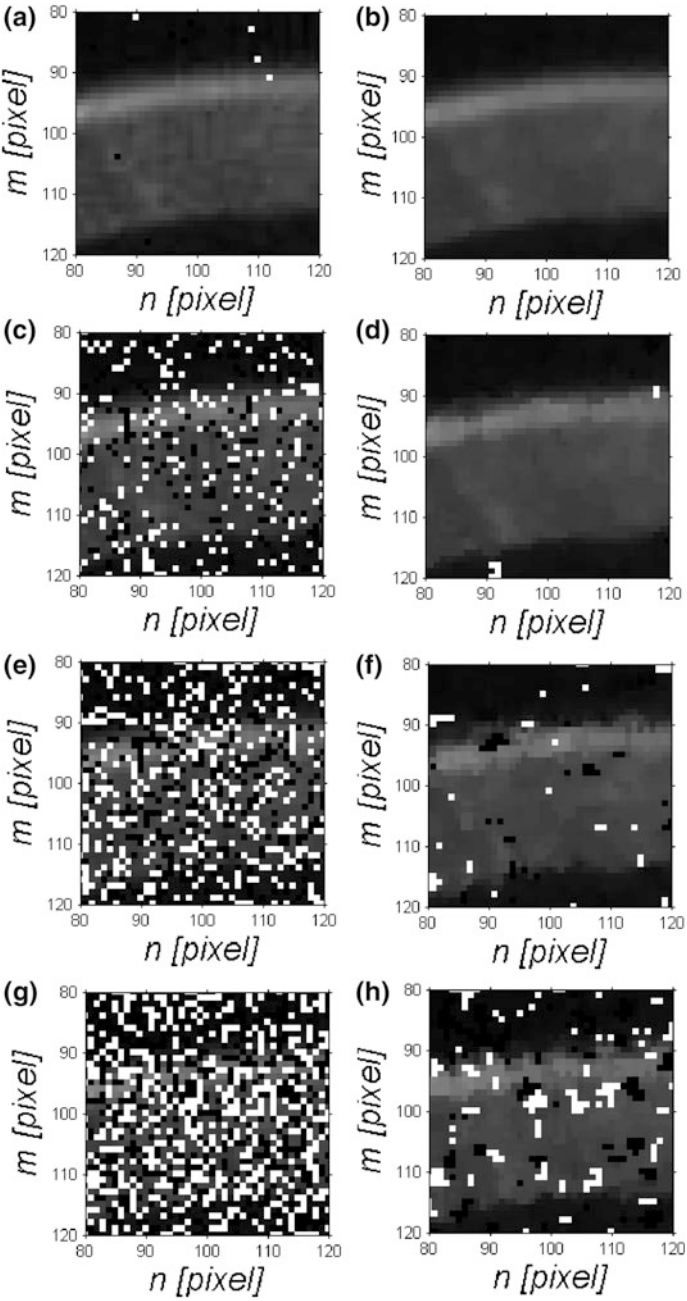

Fig. 2.2 Image sequence obtained for different d values (noise density) of noise added to the image L_G. The following images show the portion of the image L_{NO} with noise (*first column*) and the image L_{MED} after median filtering (*second column*) for: **a**, **b** $d = 0.01$, **c**, **d** $d = 0.21$, **e**, **f** $d = 0.41$, **g**, **h** $d = 61$

The image sequence (presented in Fig. 2.2) obtained for different d values (noise density) of noise added to the image L_G confirms, in all cases, the necessity and reasonableness of using the median filter. The next step of image pre-processing is normalization.

2.3 Image Normalization

Images L_G coming from different human populations obtained under various conditions are characterized by various parameters relating to brightness. In general, these images have a histogram shifted towards the darker pixels. Therefore, it is necessary to carry out normalization. Image normalization can be carried out in two different ways:

- normalization of the entire image to the range of brightness values from 0 to 1,
- normalization of individual columns or rows to a range of brightness values from 0 to 1.

The function *mat2gray*, being a simplification of *imadjust*, will be used in both types of normalization. The following source code:

```
cd('C:/data');
[FileName,PathName] = uigetfile('*.jpg','Select the Cor-
vis JPG file');
i=70;
str = sprintf('%03d',i);
LG=imread([PathName,FileName(1:end-7),str,FileName(end-
3:end)]);
figure; imshow(LG);
xlabel('n [pixel]','FontSize',20,'FontAngle','Italic');
ylabel('m [pixel]','FontSize',20,'FontAngle','Italic');
LRM1=mat2gray(LG);
figure; imshow(LRM1);
xlabel('n [pixel]','FontSize',20,'FontAngle','Italic');
ylabel('m [pixel]','FontSize',20,'FontAngle','Italic');
LRM2=zeros(size(LG));
for n=1:size(LG,2)
    LRM2(:,n)=mat2gray(LG(:,n));
end
figure; imshow(LRM2);
xlabel('n [pixel]','FontSize',20,'FontAngle','Italic');
ylabel('m [pixel]','FontSize',20,'FontAngle','Italic');
```

provides the results shown in the image in Fig. 2.3.

In general and also in the analysed case (Fig. 2.3), normalization with the first method is more often used. This is due to the fact that the relationships between

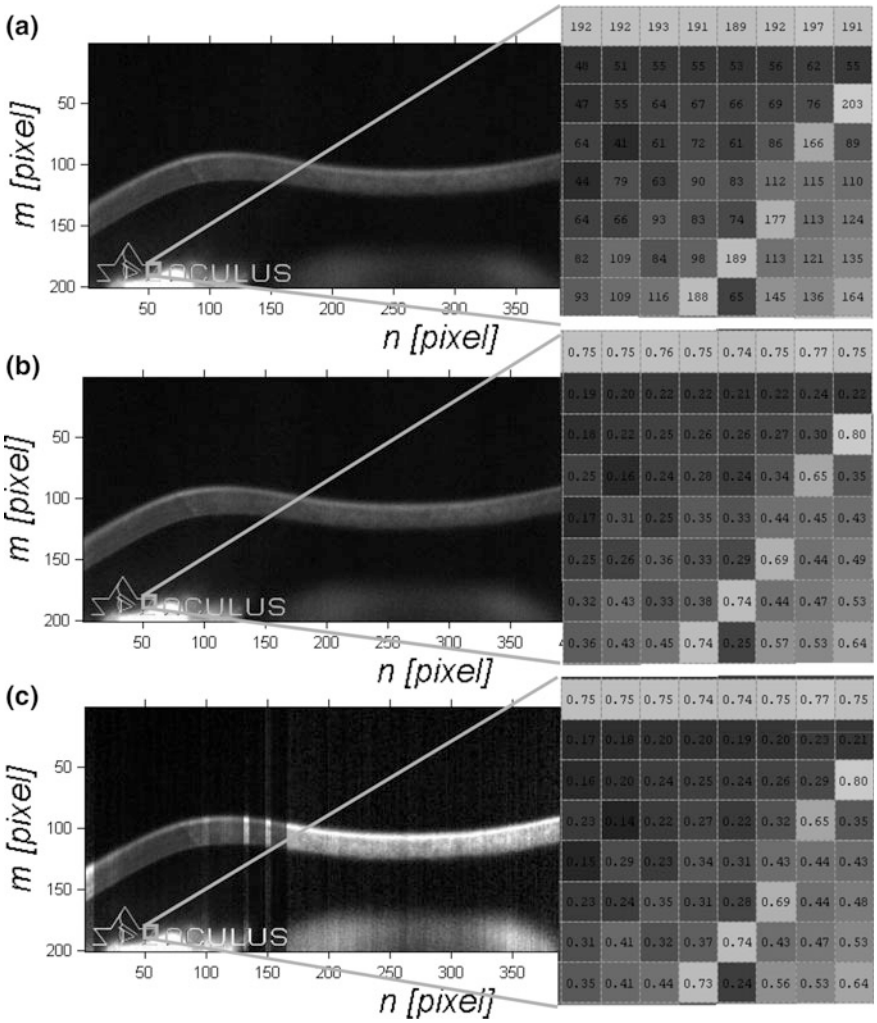

Fig. 2.3 Image L_G (**a**) and images L_{RM} after normalization with the first method (**b**) and the second method (**c**) as well as the zoom of their selected fragment

pixels remain unchanged. Subsequent columns are modified in a linear manner. The results obtained with the second normalization method are much more impressive. However, their usefulness in practice, as will be shown in the following chapters, is smaller.

2.4 Histogram Equalization and Removal of Uneven Background

The next step is histogram equalization, partly related to the normalization discussed in the previous subchapter, and the removal of uneven background. Histogram equalization is reduced to the use of *histeq* function and the function intended to visualize the histogram, namely *imhist*. The results provided in Fig. 2.4 were obtained using the source code shown below.

```
figure; imshow(LG);
xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
ylabel('m
[pixel]','FontSize',20,'FontAngle','Italic');
figure; imhist(LG)
text('Units','normalized','Position',[0.4,-
0.16],'String','intensity','FontSize',20,'FontAngle','
Italic');
ylabel('number of pixels
[pixel]','FontSize',20,'FontAngle','Italic');
grid on

LQ=histeq(LG);
figure; imshow(LQ);
xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
ylabel('m
[pixel]','FontSize',20,'FontAngle','Italic');
figure; imhist(LQ)
text('Units','normalized','Position',[0.4,-
0.16],'String','intensity','FontSize',20,'FontAngle','
Italic');
ylabel('number of pixels
[pixel]','FontSize',20,'FontAngle','Italic');
grid on
```

Figure 2.4 shows the image L_G and its histogram as well as the image after histogram equalization L_Q together with its histogram.

Histogram equalization shown in Fig. 2.4 significantly increases the contrast between adjacent pixels. The corneal contour is more visible. Non-uniformity of brightness for the rows from 1 to about 90 is also visible. It is illustrated in the graph in Fig. 2.5.

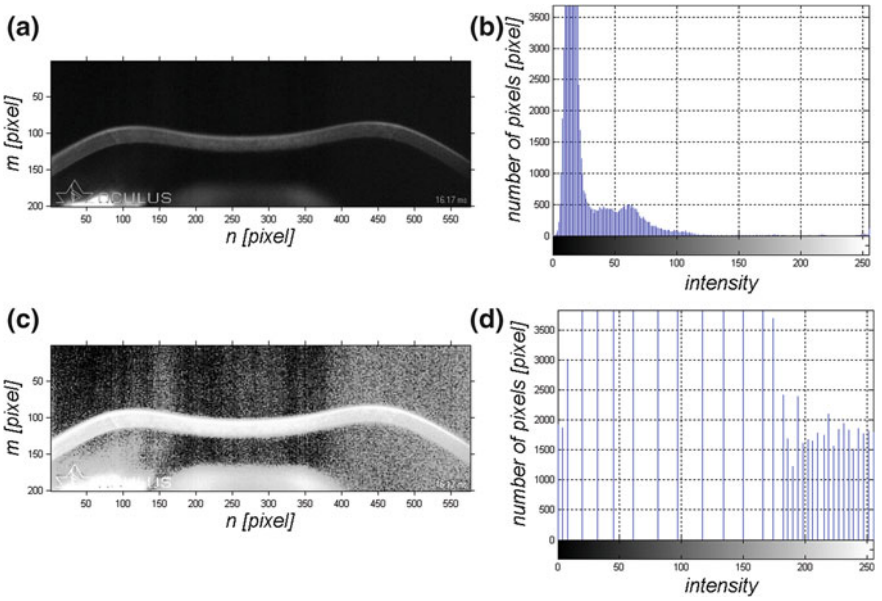

Fig. 2.4 Image L_G (**a**), and its histogram (**b**), image after histogram equalization L_Q (**d**), and its histogram

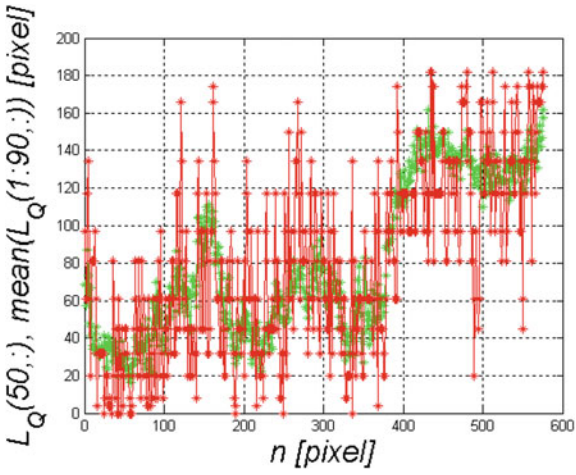

Fig. 2.5 Graph of brightness changes for a sample row $m = 50$ (*red*) and the average range for $m \in (1.90)$ (*green*), the image after histogram equalization L_Q

The graph above was obtained according to the source code being the continuation of the previous fragment, i.e.:

```
figure;
plot(mean(LQ(1:90,:)),'-g*'); hold on; grid on
plot(LQ(50,:),'-r*');
xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
ylabel('L_Q(50,:), mean(L_Q(1:90,:))
[pixel]','FontSize',20,'FontAngle','Italic');
```

where *mean(LQ(1:90,:))* expresses the value of the variable L_{QS}, i.e.:

$$L_{QS}(n) = \frac{1}{91} \cdot \sum_{m=1}^{m=90} L_Q(m,n) \tag{2.1}$$

As is apparent from the presented results (Fig. 2.5), the brightness of the image background is not fixed. Therefore, the method for the removal of uneven background was proposed. This method is based on the use of an averaging filter with a large mask size or morphological methods also with a large size of the structural element.

In the first case this is the convolution operation expressed by the relation (1.8) and a varying size of the mask h_2, from $M_{h2} \times N_{h2} = 31 \times 31$ pixels to $M_{h2} \times N_{h2} = 90 \times 90$ pixels. This size is due to the typical corneal thickness of 500 µm, which at 16.5 µm/pixel gives 31 pixels, and image resolution $M \times N = 200 \times 576$ pixels. In the second case when applying morphological methods, it is most often the operation of opening with the structural element SE_1. The size $M_{SE1} \times N_{SE1}$ of the structural element was chosen following the same rationale as in the selection of the mask size in the case of the previously discussed averaging filter. The results of using morphological operations to remove uneven background are shown in Fig. 2.6 and they were obtained for the following source code:

```
cd('C:/data');
[FileName,PathName] = uigetfile('*.jpg','Select the
Corvis JPG file');
i=70;
str = sprintf('%03d',i);
LG=imread([PathName,FileName(1:end-
7),str,FileName(end-3:end)]);
LCS=[];
```

```
for MSE1=[1 11 21 31 91]
    NSE1=MSE1;
    SE1=ones([MSE1 NSE1]);
    LO=imopen(LG,SE1);
    LC=abs(LG-LO);
    figure; imshow(histeq(LC))
    xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
    ylabel('m
[pixel]','FontSize',20,'FontAngle','Italic');
    LCS=[LCS,mean(LC(1:90,:))'];
end
figure;
plot(LCS);  grid on
xlabel('n
[pixel]','FontSize',20,'FontAngle','Italic');
ylabel('L_{CS}(n)
[pixel]','FontSize',20,'FontAngle','Italic');
legend('1x1','11x11','21x21','31x31','91x91')
```

The results shown in Figs. 2.6 and 2.7 refer to the image L_G with adjusted uneven lighting, i.e.:

$$L_C(m,n) = |L_G(m,n) - L_O(m,n)| \tag{2.2}$$

where:

$$L_O(m,n) = \max_{SE1}\left(\min_{SE1} L_G(m,n)\right) \tag{2.3}$$

and the value L_{CS} defined as:

$$L_{CS}(n) = \frac{1}{91} \cdot \sum_{m=1}^{m=90} L_C(m,n) \tag{2.4}$$

The graph presented in Fig. 2.7 shows that the smallest background brightness changes are for the structural element sized $M_{SE1} \times N_{SE1} = 11 \times 11$ pixels. However, this value of the structural element size causes degradation of the cornea (Fig. 2.6a). The optimum values are therefore the structural element sizes in the range $M_{SE1} = N_{SE1} \in (11, 21)$ pixels. Evidence of this is the analysis of the brightness of the image section—for the selected column of the image L_C, for example, for 100 pixels—Fig. 2.8.

The graphs in Fig. 2.8 reach the highest dynamics for $M_{SE1} \times N_{SE1} = 31 \times 31$ and $M_{SE1} \times N_{SE1} = 91 \times 91$ pixels. However, only for the former mask size ($M_{SE1} \times N_{SE1} = 31 \times 31$ pixels) the background brightness for $m \in (120, 160)$ pixels is comparable with the results obtained for the smaller size of the structural

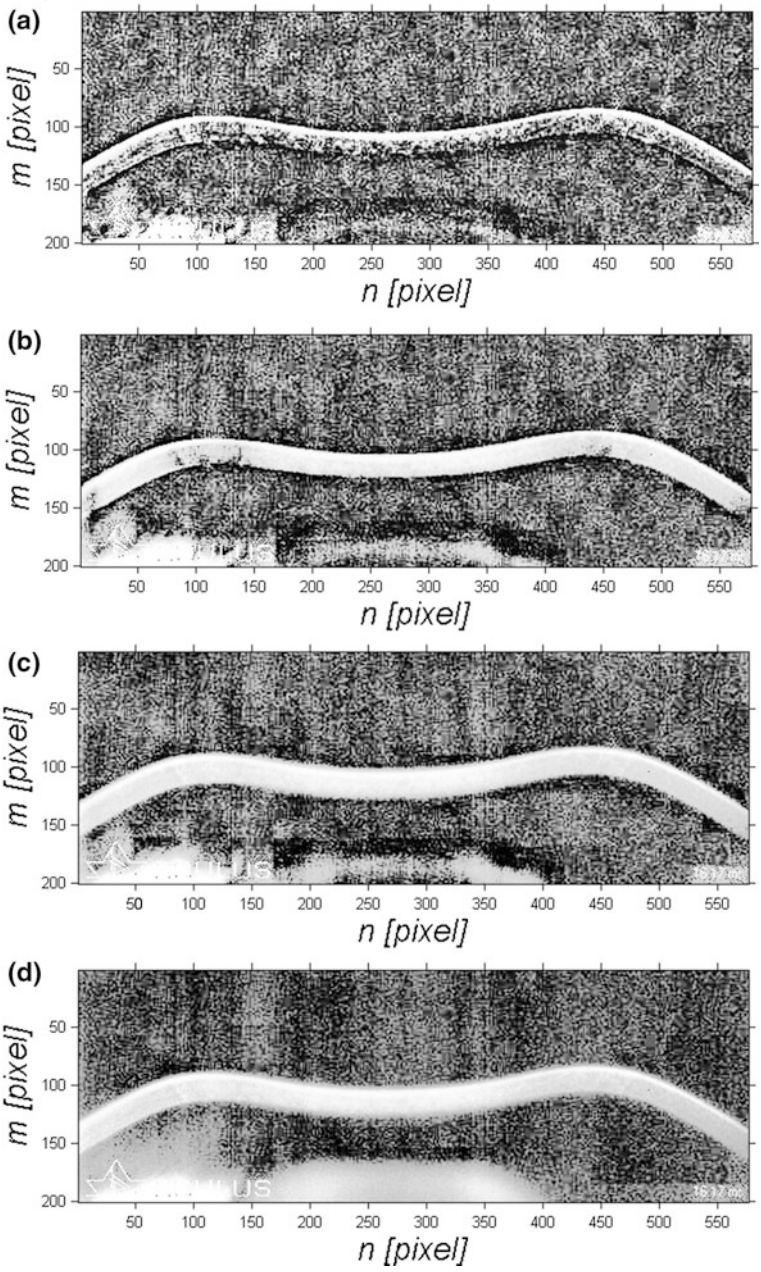

Fig. 2.6 Image L_C obtained for the structural element sized $M_{SE1} \times N_{SE1} \in \{11\ 21\ 31\ 91\}$ after histogram equalization: **a** for 11 × 11 pixels, **b** for 21 × 21 pixels, **c** for 31 × 31 pixels and **d** for 91 × 91 pixels

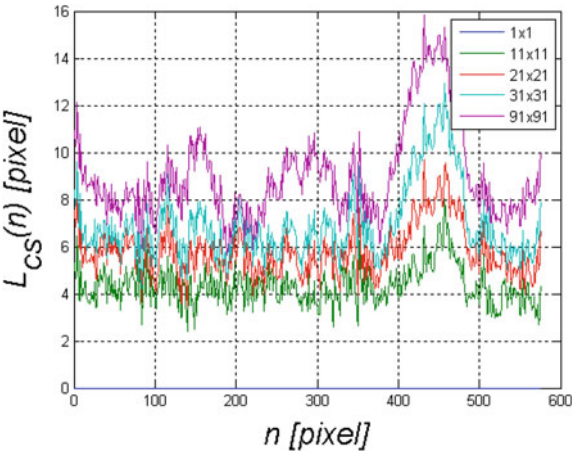

Fig. 2.7 Graphs L_{CS} for the image L_C with uneven background correction at $M_{SE1} \times N_{SE1} = 1 \times 1$ pixel (no correction), 11×11, 21×21, 31×31 and 91×91 pixels

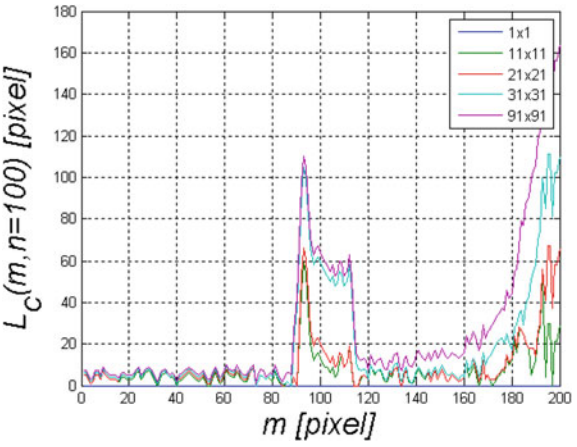

Fig. 2.8 Graphs of brightness changes for the column number 100 of the image L_C at $M_{SE1} \times N_{SE1} = 1 \times 1$ pixel (no correction), 11×11, 21×21, 31×31 and 91×91 pixels

elements (it does not exceed 10 Fig. 2.8). Therefore, when removing uneven background from the images L_G, morphological opening operations with a structural element sized $M_{SE1} \times N_{SE1} = 31 \times 31$ pixels were used.

2.5 Summary of Image Pre-processing

The individual blocks of image pre-processing discussed in the previous subchapters are shown in Fig. 2.9

The whole discussed image pre-processing algorithm has been provided in the form of a graphical user interface (GUI). The individual values of masks and structural elements are those parameters that are set by the user. The best, in terms

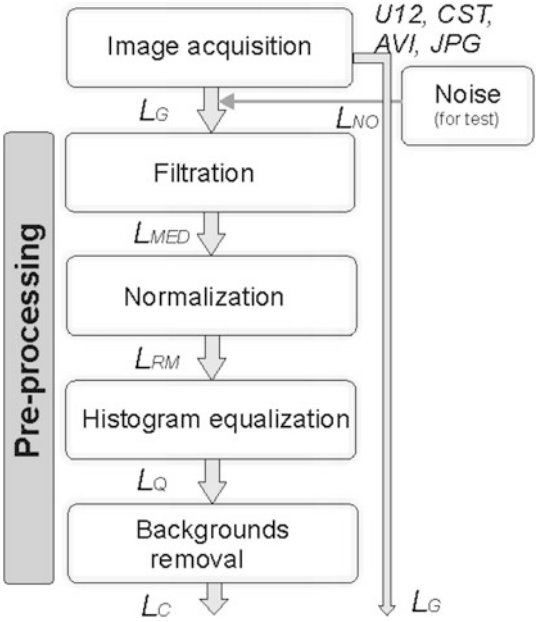

Fig. 2.9 Block diagram of individual stages of image pre-processing

of the results obtained, are the default settings. The designed GUI consists of three *m*-files. The first one *CorvisGUI.m* contains functions responsible for the arrangement of buttons, images and other windows. The second m-file named *CorvisFunctions.m* includes the above pre-processing stages. The third m-file *CorvisCalcPre.m* is responsible for preliminary image analysis. The main window of the proposed application is shown in Fig. 2.10.

The main application window shown in Fig. 2.10 consists of a few characteristic elements:

- *Default menu*—Matlab default menu, it includes image recording, reading, printing, changing the settings of menu items and so on.
- *Main menu*—menu created for pre-processing of images from the Corvis tonometer. It includes:
  - *Open*—button intended to point to *avi* or *jpg* files containing a sequence of images
  - *ReCal*—button responsible for the recalculation of parameters for a full sequence of *i* images.
  - *MedFilter*—textbox—non-editable—median filtering.
  - *3 × 3*—sample setting from the drop-down menu associated with the size of the mask h_1 during median filtering. The following settings are possible: "*None|3 × 3|7 × 7|9 × 9|11 × 11|15 × 15|23 × 23*"
  - *Normal*—textbox—non-editable—normalization.

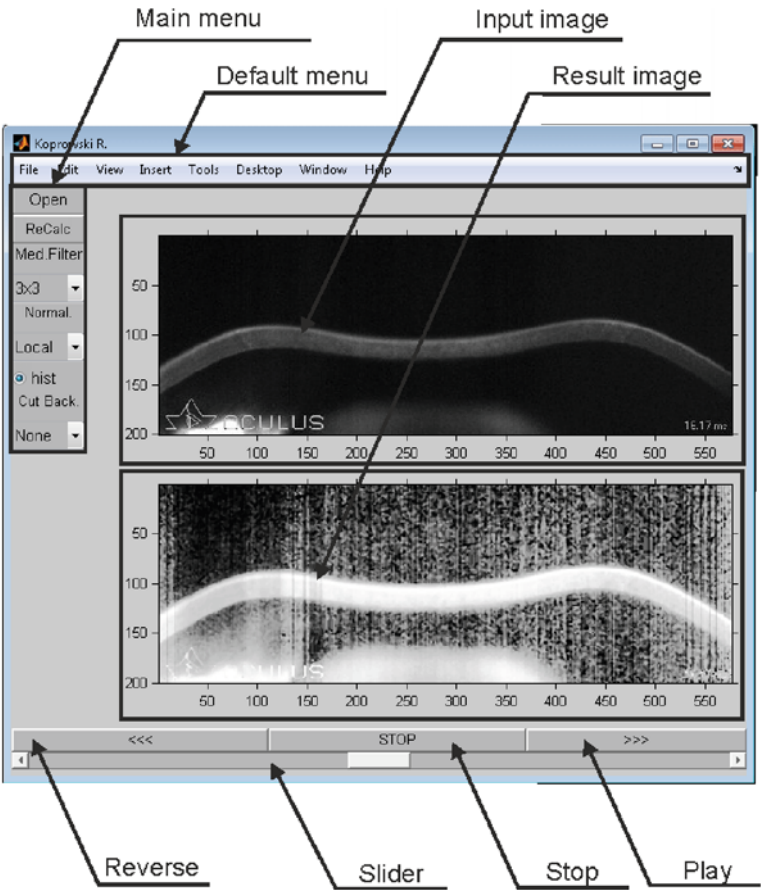

Fig. 2.10 Main application window (GUI)

- *Local*—one of the settings of the drop-down menu associated with normalization. The following settings are possible: "*None*|*Global*|*Local*"
- *Hist*—radio button designed to indicate whether histogram equalization was or was not performed in the image.
- *Cut. Back.*—textbox—non-editable—cut background.
- *None*—one of the settings of the drop-down menu associated with background removal. The following settings are possible: "*None*|*21* × *21*|*27* × *27*|*31* × *31*|*37* × *37*|*45* × *45*"

- *Input image*—input image L_G.
- *Result image*—result image as the result of analysis, in this case, the result of image pre-processing. In the general case, it is the result of image pre-processing and main processing (see Fig. 2.9).

- *Reverse*—button for playing the image sequence backwards.
- *Play*—button for playing the image sequence forward.
- *Stop*—playback stop button
- *Slider*—slider for manual viewing of images and result analysis.

Three files (source code) *CorvisGUI.m, CorvisFunctions.m* and *CorvisCalcPre.m* are presented below.

CorvisGUI.m file consists of a commentary that provides the file name and the last update. These data are displayed after calling the Matlab command >> *help corvisGUI* in the main window. The second part of the file mostly contains the function *uicontrol* designed for embedding the menu items in the main window—*figure*. The first code line (after the comment "%") relates to the declaration of global variables. This is the variable *hObj* carrying, in the form of a matrix, information about handles to individual menu items. Then there is the default path to the folder containing the sample data—in this case "*C:/data"*. The subsequent paths are the use of *uicontrol* function to create the next menu items. The handles from *hObj(2)* to *hObj(10)* concern *main menu* (Fig. 2.10), handles from *hObj(11) to hObj(15)* concern *Input image* and *Result image,* handles from *hObj(15) to hObj (18)* concern buttons *Reverse, Play, Stop, Slider* (Fig. 2.10). The full source code is presented below.

```
%CorvisGUI graphical user interface
%
%   See also CorvisFunction CorvisCalcPre
%   Copyright 2015 Robert Koprowski
%
%   $Revision: 2.1 $  $Date: 2015/07/09 10:40:11 $
global hObj

cd('C:/data');
path='C:/data';

hObj(1)=figure('Name','Koprowski
R.','NumberTitle','off');
col=get(hObj(1),'Color')*0.9;
```

```
hObj(2)=uicontrol('Style', 'pushbut-
ton','units','normalized','FontUnits','normalized',
'String', 'Open',   'Position', [0.001 0.95 0.1
0.05],'Callback', 'CorvisFunc-
tion(1)','BackgroundColor',col);
hObj(3)=uicontrol('Style', 'pushbut-
ton','units','normalized','FontUnits','normalized',
'String', 'ReCalc',   'Position', [0.001 0.9 0.1
0.05],'Callback', 'CorvisFunc-
tion(2)','BackgroundColor',col);
hObj(4)=uicontrol('Style',
'text','units','normalized','FontUnits','normalized','
String', 'Med.Filter','Value',3,'Position', [0.001
0.85 0.1 0.05],'BackgroundColor',col);
hObj(5)=uicontrol('Style',
'popup','units','normalized','FontUnits','normalized',
'String',
'None|3x3|7x7|9x9|11x11|15x15|23x23','Value',3,'Positi
on', [0.001 0.8 0.1
0.05],'Callback','CorvisFunction(3)','BackgroundColor'
,col);
hObj(6)=uicontrol('Style',
'text','units','normalized','FontUnits','normalized','
String', 'Normal.','Value',3,'Position', [0.001 0.75
0.1 0.05],'BackgroundColor',col);
hObj(7)=uicontrol('Style',
'popup','units','normalized','FontUnits','normalized',
'String', 'None|Global|Local','Value',3,'Position',
[0.001 0.7 0.1
0.05],'Callback','CorvisFunction(4)','BackgroundColor'
,col);
```

```
hObj(8)=uicontrol('Style', 'radiobut-
ton','units','normalized','FontUnits','normalized','St
ring', 'hist','Value',1,'Position', [0.001 0.65 0.1
0.05],'Callback','CorvisFunction(5)','BackgroundColor'
,col);
hObj(9)=uicontrol('Style',
'text','units','normalized','FontUnits','normalized','
String', 'Cut Back.','Value',2,'Position', [0.001 0.6
0.1 0.05],'BackgroundColor',col);
hObj(10)=uicontrol('Style',
'popup','units','normalized','FontUnits','normalized',
'String',
'None|21x21|27x27|31x31|37x37|45x45','Value',4,'Positi
on', [0.001 0.55 0.1
0.05],'Callback','CorvisFunction(6)','BackgroundColor'
,col);

hObj(11)=axes('Parent',hObj(1),'units','normalized','P
osition', [0.2 0.52 0.78 0.45]);
hObj(12)=imshow(rand(200,576),'InitialMagnification','
fit','Parent',hObj(11)); hold on
hObj(13)=axes('Parent',hObj(1),'units','normalized','P
osition', [0.2 0.1 0.78 0.45]);
hObj(14)=imshow(rand(200,576),'InitialMagnification','
fit','Parent',hObj(13)); hold on

hObj(15)=uicontrol('Style',
'slider','units','normalized','FontUnits','normalized'
,'BackgroundColor',col,'Position', [0.001 0.01 0.999
0.03],'Min',0,'Max',139,'Value',0,'Callback', 'Corvis-
Function(7)');
hObj(16)=uicontrol('Style', 'pushbut-
ton','units','normalized','FontUnits','normalized',
'String', '<<<',  'Position', [0.001 0.04 0.35
0.04],'Callback', 'CorvisFunc-
tion(8)','BackgroundColor',col);
hObj(17)=uicontrol('Style', 'pushbut-
ton','units','normalized','FontUnits','normalized',
'String', 'STOP','Position', [0.351 0.04 0.349
0.04],'Callback', 'CorvisFunc-
tion(9)','BackgroundColor',col);
hObj(18)=uicontrol('Style', 'pushbut-
ton','units','normalized','FontUnits','normalized',
'String', '>>>','Position', [0.70 0.04 0.3
0.04],'Callback', 'CorvisFunc-
tion(10)','BackgroundColor',col);
```

The second file *CorvisFunction.m* relates to action that is taken when pressing any button or selecting any item from *Main menu*. The function argument is a variable *sw* which is dependent on the element from *Main menu* which was selected with a mouse or pressed—it is linked with, e.g. notation, as in the case of the slider *'Callback','CorvisFunction(7)'* (see the source code above). Any *avi* or *jpg* file is read by opening the window intended for the file selection using *uigetfile* function and then reading the image sequence. In the case of the sequence of *jpg* images, it was assumed, according to the data obtained directly from the Corvis tonometer, that they are numbered from **000.jpg* to **139.jpg*. In another case or in the absence of any file, Matlab will report an error. For images saved as subsequent frames of *avi* video, all the available frames are read, regardless of their number—see the variable *aviObj.NumberOfFrames*. At this point I encourage readers to test the program thoroughly and to create their own security. For example, information can be shown to the operator in the absence of the full sequence of *jpg* files or in the case of incorrect image resolution, e.g. using the function *warndlg*. The source code of the discussed CorvisFunction is shown below:

```
function []=CorvisFunction(sw)
global hObj PathName FileName break_ LGi LMED LRM LQ
LC h1 Norm Hist MSE1
if (sw==1)|(sw==2) % OPEN OR RECALCULATION
    if sw==1 % if OPEN
        [FileName,PathName] = uiget-
file('*.jpg;*.avi','Select the Corvis JPG or AVI
file');
    end
    if FileName~=0 % if select file
        if strcmp(FileName(end-
2:end),'jpg')|strcmp(FileName(end-2:end),'JPG')
                set(hObj(14),'CData',rand(200,576))
            for i=0:139
                str = sprintf('%03d',i);
                LGi=imread([PathName,FileName(1:end-
7),str,FileName(end-3:end)]);
                LGi=double(LGi)/255;
                set(hObj(12),'CData',LGi);
                set(hObj(15),'Value',i);
                set(hObj(15),'Max',139,'Min',0);
                CorvisFunction(3);
                pause(0.01)

            end
```

```
            elseif strcmp(FileName(end-2:end),'avi')
                set(hObj(14),'CData',rand(200,576))
                aviObj = mmreader([PathName,FileName]);
                for i=1:aviObj.NumberOfFrames
                    LGi = read(aviObj, i);
                    LGi=double(LGi(:,:,1))/255;
                    set(hObj(12),'CData',LGi);
                    set(hObj(15),'Value',i);

    set(hObj(15),'Max',aviObj.NumberOfFrames,'Min',1);
                    CorvisFunction(3);
                    pause(0.01)
                end
            else
                FileName(end-2:end)
                disp('Please select AVI or JPG')
            end
        end
    end
```

Subsequent parts of *CorvisFunction* relate to assigning values to variables *h1, Norm, Hist, MSE*. The values are selected by the user by means of the drop-down menu. This drop-down menu has values of the variable *sw* equal to 3, 4, 5 or 6. Accordingly:

- *h1* is the mask size of the averaging filter,
- *Norm* is the variable indicating whether normalization needs to be performed, or whether normalization should be local or global,
- *Hist* is the variable indicating whether histogram equalization needs to be performed or not,
- *MSE1* is the size of the structural element necessary to remove uneven background, this element is always square with a declared number of rows and columns.

The source code responsible for this part is given below:

```
if (sw==3)|(sw==4)|(sw==5)|(sw==6)
% MEDIAN None|3x3|7x7|9x9|11x11|15x15|23x23
    if get(hObj(5),'Value')==1
        h1=0;
    end
    if get(hObj(5),'Value')==2
        h1=3;
    end
    if get(hObj(5),'Value')==3
        h1=7;
    end
    if get(hObj(5),'Value')==4
        h1=9;
    end
    if get(hObj(5),'Value')==5
        h1=11;
    end
    if get(hObj(5),'Value')==6
        h1=15;
    end
    if get(hObj(5),'Value')==7
        h1=23;
    end
% normalization %None|Global|Local
    if get(hObj(7),'Value')==1
        Norm=0;
    end
    if get(hObj(7),'Value')==2
        Norm=1;
    end
    if get(hObj(7),'Value')==3
        Norm=2;
    end
% Hist %None|HIST
    Hist=get(hObj(8),'Value');
%Backgroung %None|21x21|27x27|31x31|37x37|45x45
    if get(hObj(10),'Value')==1
        MSE1=0;
    end
    if get(hObj(10),'Value')==2
        MSE1=21;
    end
    if get(hObj(10),'Value')==3
        MSE1=27;
    end
    if get(hObj(10),'Value')==4
        MSE1=31;
    end
```

```
    if get(hObj(10),'Value')==5
        MSE1=37;
    end
    if get(hObj(10),'Value')==6
        MSE1=45;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[LMED,LRM,LQ,LC]=CorvisCalcPre(LGi,h1,Norm,Hist,MSE1);
        set(hObj(14),'CData',LC);

end
```

The last part of the code contains a reference to the function *CorvisCalcPre* which is responsible for performing computations in the image *LGi* (*i*-th input image). The last part of the discussed source code of the function *CorvisFunction* relates to navigation while browsing back and forth the images before and after image pre-processing. At the end of each loop and performed calculations (*CorvisCalcPre* function), there is *pause* function with a parameter (0.01 s) which refers to the time needed to display an image. In the absence of the *pause* function, the last image in the performed loop will be displayed (*for*). This function fragment includes another variable *break_* which is the flag directly related to pressing the *Stop* button. The *set* functions refresh the image and other parameters according to their calculated values.

```
    if sw==7 % SLIDER
        i=round(get(hObj(15),'Value'));
        if strcmp(FileName(end-
    2:end),'jpg')|strcmp(FileName(end-2:end),'JPG')
            str = sprintf('%03d',i);
            LGi=imread([PathName,FileName(1:end-
    7),str,FileName(end-3:end)]);
            LGi=double(LGi)/255;
            set(hObj(12),'CData',LGi);
        elseif strcmp(FileName(end-2:end),'avi')
            aviObj = mmreader([PathName,FileName]);
            LGi = read(aviObj, i);
            LGi=double(LGi(:,:,1))/255;
            set(hObj(12),'CData',LGi);
        else
            disp('Please select AVI or JPG')
        end
        CorvisFunction(3);
    end
    if sw==8 % <<<<<<<<<<<<
```

```
        if strcmp(FileName(end-
2:end),'jpg')|strcmp(FileName(end-2:end),'JPG')
            for i=round(get(hObj(15),'Value')):-1:0
                str = sprintf('%03d',i);
                LGi=imread([PathName,FileName(1:end-
7),str,FileName(end-3:end)]);
                LGi=double(LGi)/255;
                set(hObj(12),'CData',LGi);
                set(hObj(15),'Value',i);
                set(hObj(15),'Max',139,'Min',0);
                CorvisFunction(3);
                if break_==1
                    break_=0;
                    break
                end
                pause(0.01)

            end
        elseif strcmp(FileName(end-2:end),'avi')
            aviObj = mmreader([PathName,FileName]);
            for i=round(get(hObj(15),'Value')):-1:1
                LGi = read(aviObj, i);
                LGi=double(LGi(:,:,1))/255;
                set(hObj(12),'CData',LGi);
                set(hObj(15),'Value',i);

set(hObj(15),'Max',aviObj.NumberOfFrames,'Min',1);
                CorvisFunction(3);
                if break_==1
                    break_=0;
                    break
                end
                pause(0.01)
            end
        else
            FileName(end-2:end)
            disp('Please select AVI or JPG')
        end
end
if sw==9
  break_=1;
end
if sw==10 % >>>>>>>>>>>>>>>>>>>>>>
        if strcmp(FileName(end-
2:end),'jpg')|strcmp(FileName(end-2:end),'JPG')
            for i=round(get(hObj(15),'Value')):139
                str = sprintf('%03d',i);
                LGi=imread([PathName,FileName(1:end-
7),str,FileName(end-3:end)]);
```

```
                    LGi=double(LGi)/255;
                    set(hObj(12),'CData',LGi);
                    set(hObj(15),'Value',i);
                    set(hObj(15),'Max',139,'Min',0);
                    CorvisFunction(3);
                    if break_==1
                        break_=0;
                        break
                    end
                    pause(0.01)

                end
            elseif strcmp(FileName(end-2:end),'avi')
                aviObj = mmreader([PathName,FileName]);
                for
i=round(get(hObj(15),'Value')):aviObj.NumberOfFrames
                    LGi = read(aviObj, i);
                    LGi=double(LGi(:,:,1))/255;
                    set(hObj(12),'CData',LGi);
                    set(hObj(15),'Value',i);

set(hObj(15),'Max',aviObj.NumberOfFrames,'Min',1);
                    CorvisFunction(3);
                    if break_==1
                        break_=0;
                        break
                    end
                    pause(0.01)
                end
            else
                FileName(end-2:end)
                disp('Please select AVI or JPG')
            end
end
```

The third and last discussed m-file, namely *CorvisCalcPre*, is related to performing calculations in the image L_G. The order of operations is directly related to the block diagram—Fig. 2.9. When a given operation is not performed, the variable

value is equal to zero, e.g.: *h1 = 0*. Details of the source code of *CorvisCalcPre* are shown below:

```
function
[LMED,LRM,LQ,LC]=CorvisCalcPre(LG,h1,Norm,Hist,MSE1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if h1==0
    LMED=LG;
else
    LMED=medfilt2(LG,[h1 h1],'symmetric');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Norm==0 % None
    LRM=LMED;
end
if Norm==1 % LOCAL
    LRM=mat2gray(LMED);
end
if Norm==2 % global
    LRM=zeros(size(LMED));
    for n=1:size(LMED,2)
       LRM(:,n)=mat2gray(LMED(:,n));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Hist==0 % None
    LQ=LRM;
else % Hist
    LQ=histeq(LRM);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if MSE1==0
    LC=LQ;
else
    SE1=ones([MSE1 MSE1]);
    LO=imopen(LQ,SE1);
    LC=abs(LQ-LO);
end
```

Individual actions are intuitive and an interested reader can easily trace them by analysing the successive lines of the presented source code.

Consequently, the results obtained from the three discussed functions, *CorvisGUI.m, CorvisFunctions.m* and *CorvisCalcPre.m*, form the basis for further analysis, in particular, the image L_C and the source image L_G resulting from image pre-processing. Details of further analysis are discussed in the next chapter.

Springer

http://www.springer.com/978-3-319-29545-9

Image Analysis for Ophthalmological Diagnosis
Image Processing of Corvis® ST Images Using Matlab®
Koprowski, R.
2016, XIII, 125 p. 73 illus., 48 illus. in color., Hardcover
ISBN: 978-3-319-29545-9