Robert Koprowski, Zygmunt Wróbel

Image Processing in Optical Coherence Tomography

using Matlab

University of Silesia 2011

CONTENTS

PREFACE

Dear Readers, the book you have in your hands is a summary of research carried out at the Department of Computer Biomedical Systems, Institute of Computer Science, University of Silesia in Katowice in cooperation with the team of Prof. Edward Wylęgała, D.Sc., M.D. This cooperation resulted in the creation of methods for ophthalmologists support in OCT images automated analysis. These methods, like the application developed on their basis, are used during routine examinations carried out in hospital.

The monograph comprises proposals of new and also of known algorithms, modified by authors, for image analysis and processing, presented on the basis of example of Matlab environment with Image Processing tools. The results are not only obtained fully automatically, but also repeatable, providing doctors with quantitative information on the degree of pathology occurring in the patient. In this case the anterior and posterior eye segment is analysed, e.g. the measurement of the filtration angle or individual layers thickness.

To introduce the Readers to subtleties related to the implementation of selected fragments of algorithms, the notation of some of them in the Matlab environment has been given. The presented source code is shown only in the form of example of implementable selected algorithm. In no way we impose here the method of resolution on the Reader and we only provide the confirmation of a possibility of its practical implementation.

The book is addressed both to ophthalmologists willing to expand their knowledge in the field of automated eye measurements and also primarily to IT specialists, Ph.D. students and students involved in the development of applications designed for automation of measurements for the needs of medicine.

This book is available free of charge in an electronic version. The authors agree to disseminate, duplicate and use in any way free of charge this book. A commercial use of algorithms and images presented is protected by law.

The authors thank cordially Prof. Edward Wylęgała, D.Sc., M.D. and his team for the provided images and valuable guidance and consultations.

1 INTRODUCTION

An optical tomography is a modern, non-invasive technique for a tissue section imaging, in this case of anterior and posterior eye segment, using the light scattered on individual layers of the examined tissue. The spectral tomography, as compared with the hitherto solutions (e.g. time tomography), features much higher resolution. The elimination of a moving mirror, necessary to scan deep into the object examined, allows also shortening the examination time (object scanning) approx. a hundred times. A short time of scan performance as well as its sequentiality and maintaining a constant shift allows obtaining 3D images [7]. Many instruments available now allow such imaging. Instruments, used to acquire images used in this book, have been selected from them, i.e.:

- SOCT Copernicus HR,
- Zeiss Cirrus HD-OCT,
- Zeiss Stratus OCT Zeiss Visante OCT.

Overall, the algorithms presented below have been tested on a group of more than 100,000 images of patients, both healthy and with a significant degree of eye pathology.

The most interesting fragments of algorithms presented have been recorded in the Matlab environment, version 7.2.0.232 (R2006a) and Image Processing Toolbox, version 5.2 (R2006a). Individual fragments of algorithms are separated only with text, comments and after integration create a whole allowing a full image analysis. Despite that, the authors assume that the Reader is familiar with basic functions and possibilities of the Matlab software, with special emphasis on the operations carried out on matrices. If it is not the case, the authors refer Readers to familiarise themselves with Matlab basics, e.g. references [41].

In terms of the imaged object, the description of OCT images analysis and processing methods has been divided into two parts: the anterior eye segment has been presented in the first part, while the posterior eye segment has been presented in the second part of this monograph, in accordance with Fig. 1-1.

PART I

SEQUENCE OF IMAGES

LENS

HYALINE

RETINA

PART II

0.1mm
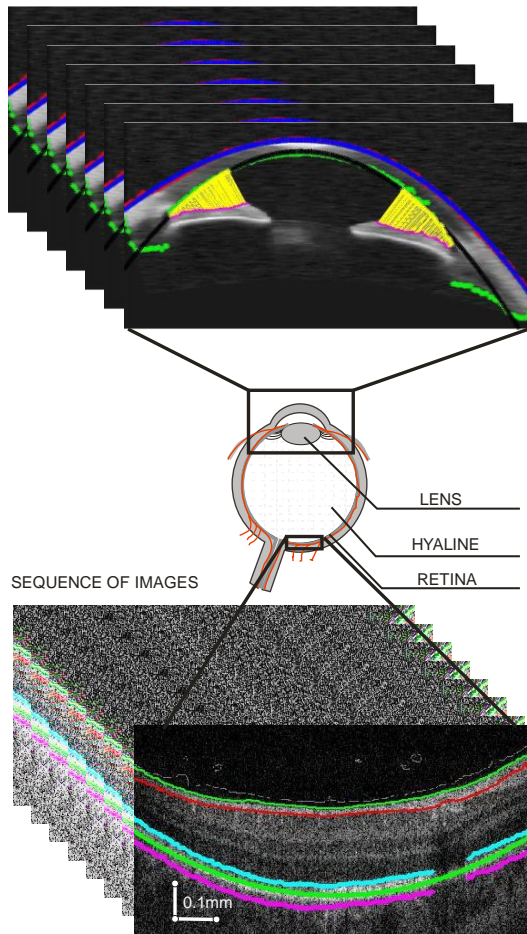
Fig. 1-1 Cross-section of the front and back of the eye with a marked characteristic of the location areas

2 ACQUISITION OF IMAGE DATA

Difficulties with reading and appropriate interpretation of data recorded for individual patients in OCT equipment result primarily from manufacturers' fears of developing own competitive software. Frequently the information is a company secret. Fortunately the OCT equipment software produced nowadays more and more often records the data acquired in a DICOM or similar format. The Optopol OCT is an exception here, recording the acquisition data in one compressed file.

On the other hand DICOM images may be read in Matlab using the dicomread function available in the Image Processing package. Unfortunately, the usability of this function, in version 5.2 of the Image Processing package possessed now by the authors to read DICOM images originating from reputed OCT manufacturers, is small. Missing header tags and frequently a specific record of the image (JPEG2000) are the reason for which the reading of files is difficult. Such files cannot be read also by majority of freeware available in the Internet and designed for viewing typical DICOM images.

Let us look at the header read from the track

```
path_name='D:/source/1.DICOM'
```

as follows:

```
fid = fopen(path_name, 'r');
dataa = fread(fid,'uint8');
fclose(fid);
```

then we obtain the result directly from OCT Carl Zeiss Meditec file for example for the first thousand of characters

```
char(dataa(1:1000)')
```

we obtain the result:

```
ans =

DIC C ORIGINA  UI  1.2.826.0.1.3680043.2.139.3.1.1 UI:
1.2.826.0.1.3680043.2.139.3.1.1001.1017.20070928114546359
D 2007092 ! D 2007092 " D 2007092 0 TM11435 1 TM11452 2
TM11452 P SH  p LO Carl Zeiss Meditec Inc.  LO& Uniwersytet
Slaski w Katowicac SH  0LO  >LO  pPN
```

```
 test^test^^
□LO
```

```
1000 PN Koprowski^Robert^^^ L KOWALSKI ! LO  0 D 20060115@
CS F   @LT
LO
```

```
1054  L 1.2.0.1 0L Chamber
 UI:
1.2.826.0.1.3680043.2.139.3.1.1001.1017.20070928114359062
UI:
1.2.826.0.1.3680043.2.139.3.1.1001.1017.20070928114546312
IS 0   IS 0   IS0   R UI:
1.2.826.0.1.3680043.2.139.3.1.1001.1017.20070928114546312
` CS OD   @LT  ( US  (
```

```
CS MONOCHROME2 ( US  ( US
```

```
( US  ( US  ( US  (───────────────────US @ SQ
SH
 ALL_SCANS SH 99CZM ──────────────────...
```

The information available in the Internet specify clearly the place, where the data is located, e.g.

- 0010,0010 PatientName N
- 0020,0013 InstanceNumber N
- 0002,0010 TransferSyntaxUID N
- 0028,0100 BitsAllocated N
- 0028,0111 LargestImagePixelValueInPlane N

This means that patient's Name and Surname given in hexadecimal notation in appropriate sequence starting from LSB and ending at MSB will be preceded with values read from the file, i.e.: `16 0 16 0`. In the example file presented these are the values comprised by elements from 480 to 506 range, i.e.: `char(dataa(480:506)')`, `dataa(480:506)'` as a result we obtain:

```
ans =
 PN Koprowski^Robert

ans =

  Columns 1 through 23

    16     0    16     0    80    78    20     0    75
 111   112   114   111   119   115   107   105    94    82
 111    98   101   114
```

```
  Columns 24 through 27

   116    94    94    94
```

The reading of the remaining information comes down only to finding appropriate tag and then the record content. The skeleton of example function `OCT_head_read`, returning the information on the header `header_dicom` and the matrix `Ls` of image, designed to read the data originating from OCT Visante, are presented below:

```
function [header_dicom,Ls]=OCT_head_read(dataa)
flagi=zeros([1 100]);
iu=1;
header_dicom=[];
for i=1:30000
    te=dataa(i:(i+3));
    if (sum((te'==[16 0 16 0]))==4)&(flagi(1)==0)
        Patinet_Name=char(dataa(i+8:(i+8+dataa(i+6)-1))');
        header_dicom.Patinet_Name{iu}=Patinet_Name;
        flagi(1)=1;
    end
    if (sum((te'==[32 0 19 0]))==4)&(flagi(2)==0)
            Instance_Number=char(dataa(i+8));
        header_dicom.Instance_Number{iu}=Instance_Number;
        flagi(2)=1;
    end
    if (sum((te'==[2 0 16 0]))==4)&(flagi(3)==0)
        UID=dataa(i:i+30);
        header_dicom.UID{iu}=char(UID)';
        flagi(3)=1;
    end
    if (sum((te'==[40 0 0 1]))==4)&(flagi(4)==0)
        Bp=dataa(i+8);
        header_dicom.bits_per_pixel{iu}=dataa(i+8);
        flagi(4)=1;
    end
    if (sum((te'==[40 0 17 0]))==4)&(flagi(5)==0)
        M=dataa(i+9)*256+dataa(i+8);
        header_dicom.Mlumn{iu}=M;
        flagi(5)=1;
    end
    if (sum((te'==[224 127 0 0]))==4)&(flagi(6)==0)

header_dicom.length_pixel_data{iu}=dataa(i+10)*256*256+data
a(i+9)*256+dataa(i+8) ;
        flagi(6)=1;
    end
    if (sum((te'==[40 0 0 1]))==4)&(flagi(7)==0)
```

```
            header_dicom.bits_allocated{iu}=dataa((i+8))';
            flagi(7)=1;
        end
        if (sum((te'==[40 0 1 1]))==4)&(flagi(8)==0)
            header_dicom.bits_stored{iu}=dataa((i+8))';
            flagi(8)=1;
        end
        if (sum((te'==[40 0 2 1]))==4)&(flagi(9)==0)
            header_dicom.high_bit{iu}=dataa((i+8))';
            flagi(9)=1;
        end
        if (sum((te'==[40 0 2 0]))==4)&(flagi(10)==0)
            header_dicom.samples_per_pixel{iu}=dataa((i+8))';
            flagi(10)=1;
        end
        if (sum((te'==[40 0 16 0]))==4)&(flagi(11)==0)
            N=dataa(i+9)*256+dataa(i+8);
            header_dicom.Nws{iu}=N;
            flagi(11)=1;
        end
        if (sum((te'==[8 0 32 0]))==4)&(flagi(12)==0)
            Date_=dataa((i+8):(i+15))'-48;
            header_dicom.date_study{iu}=Date_;
            %(dataa(i:(i+28))')
            flagi(12)=1;
        end
        if (sum((te'==[224 127 16 0]))==4)&(flagi(13)==0)
            ipp=i;
            header_dicom.pixel_start_data{iu}=i;
            flagi(13)=1;
        end
        if sum(flagi)==13;
            break
        end
    end
end
if (length(dataa)-ipp)<N*M
    disp('Small pixel')
end
if Bp==8
    L=dataa(ipp+12:ipp+11+N*M);
    L=reshape(L,[M N]);
    Ls=L;
end
if Bp==16
    LH=dataa(ipp+12:2:ipp+11+N*M*2);
    LL=dataa(ipp+13:2:ipp+11+N*M*2);
    L1L=reshape(LL,[M N]);
    L1H=reshape(LH,[M N]);
    Ls=L1H+L1L*256;
    Ls(Ls>2^14)=Ls(Ls>2^14)-2^16;
```

```
end
```

The above function reads the header from a DICOM file, seeking appropriate tags. The functionality of individual tags and their full list may be easily checked in an Internet browser entering "DICOM tags" or on http://medical.nema.org/ website. Having read the selected information from the header, it converts, reorganising the data (`reshape` function), to size M x N. The final image to be recorded

```
path_name='d:/OCT/SOURCES/1.DCM';
fid = fopen(path_name, 'r');
dataa = fread(fid,'uint8');
fclose(fid);
 [header_dicom,Ls]=OCT_head_read(dataa);
figure; imshow(Ls,[]); colormap('jet')
```

OCT image read using the function `OCT_head_read` has been shown in Fig. 2-1.
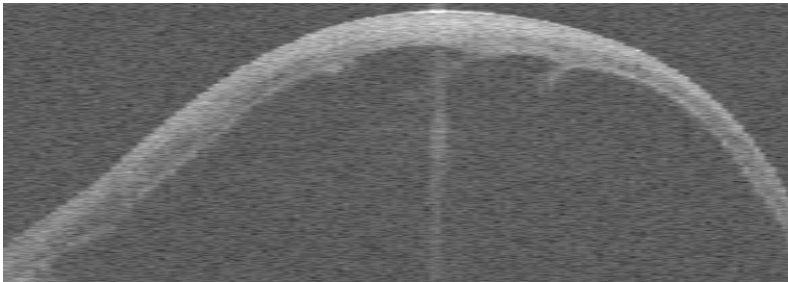
Fig. 2-1 OCT image read using the function `OCT_head_read`

It is necessary to remind here that this is only an example function and it does not fully uses up very broad (described in the place referred to above) scope of possibilities to record image, video and other sequences in a DICOM file.

Another way of recording is possessed by the company Optopol, packing the data in one file. After unpacking (using any unpacking software) the images of image sequences recorded in a bmp format are available as well as a file of inf extension containing the information on patient's data and locations of individual images on the xy axis. Assuming that files from OCT equipment are available in the path 'd:/OCT/SOURCES/' and that results in the form of directories of the same names as names of files to be unpacked should be in 'd:/OCT/FOLDERS/' the script for automatic unpacking can be written as:

```matlab
dr=dir('d:/OCT/SOURCES/');
for i=1:size(dr)
    cc=getfield(dr,{i},'name');
    iscc=getfield(dr,{i},'isdir');
    if (iscc==0)&(strcmp(cc(end-2:end),'OCT'))
      unzip(['d:/OCT/SOURCES/',cc],['d:/OCT/FOLDERS/',cc]);
    end
end
```

The images of anterior eye segment, obtained using the function presented above, have resolution of 256x1024 pixels, what at the example measuring range of 8mm x 16 mm gives 0.0313 mm/pixel. In the case of posterior eye segment the resolution of images obtained e.g. from Copernicus tomograph is 1010x684.

These functions are only examples, very limited, of methods resolving the problem of data reading from OCT instruments. Instead, they were used to enter the images to the Matlab space.

PART I

3 ANALYSIS OF ANTERIOR EYE SEGMENT

The first part of this monograph presents the issues related to the analysis of anterior eye segment in terms of selection of algorithms analysing the filtration angle and the anterior chamber volume. These are among fundamental issues not resolved so far in applications available in modern tomographs. These calculations are either not possible at all or not fully automated. The algorithms presented below not only fully resolve the problem mentioned but also indicate other possible ways to resolve it.
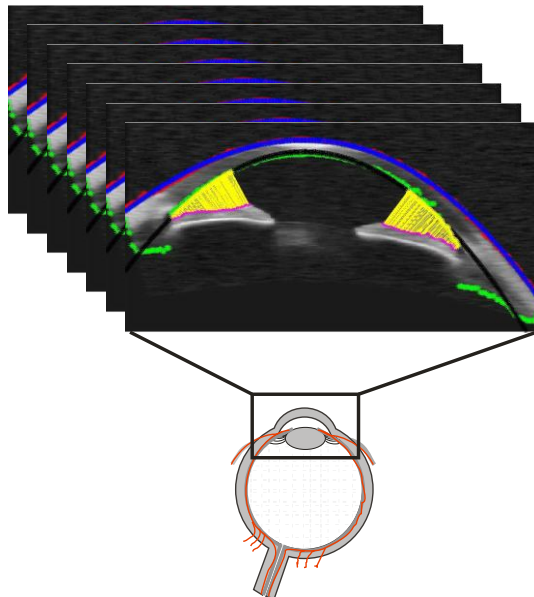
3.1   Introduction to Anterior Eye Segment Analysis

The filtration angle, i.e. the iridocorneal angle (Fig. 3-1, Fig. 3-2), is the structure responsible for the aqueous humour drainage from the eye's anterior chamber. Both a correct production of the aqueous humour by the ciliated epithelium and a correct rate of aqueous humour drainage through the filtration angle are conditions for a correct intraocular pressure. All anatomical anomalies, the angle narrowing and the angle closing result in a more difficult drainage and in a pressure increase. The examination allowing to determine the angle width is named the gonioscopy. Based on the angle width the glaucoma may be broken down to the open angle glaucoma and to the closed angle glaucoma [16], [18].
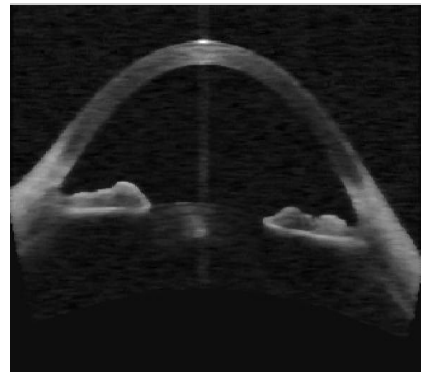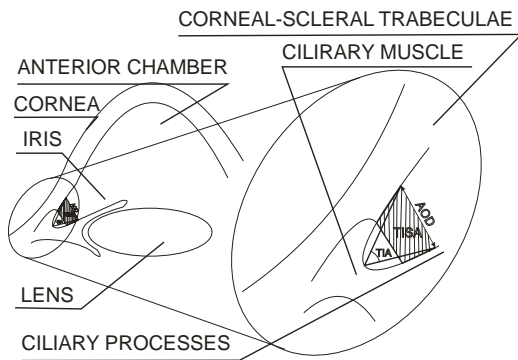
Fig. 3-1 A section of the anterior segment of an eye with marked positions of characteristic areas

Fig. 3-2 An example of the image of the anterior segment of an eye

The methods presented are not precisely defined and doctors each time must choose the measuring method used. In consequence, the results obtained are not reliable and difficult to verify and to compare with the standard and with other doctors.

Fig. 3-3 Methods for the filtration angle measurement: a) AOD (Angle Opening Distance) method, b) TIA (Trabecular-Iris Angle) method, c) TISA (Trabecular-Iris-Space Area) method

So far all the measurements mentioned have been performed manually indicating appropriate characteristic points. However, in the cases of individual variation or pathology these methods have different accuracy and repeatability of measurements resulting primarily from their nature and from the measured quantities. The AOD (Angle Opening Distance) method (Fig. 3-3.a)) consists in the measurement of distance, TIA (Trabecular-Iris Angle) (Fig. 3-3.b)) in the measurement of angle and (Fig. 3-3.c)) TISA (Trabecular-Iris Space Area) method consists in the measurement of area, respectively [20] (the methods have been shown together in Fig. 3-4).

Fig. 3-4 Methods for the filtration angle measurement: AOD (Angle Opening Distance) method, TIA (Trabecular-Iris Angle) method, TISA (Trabecular-Iris-Space Area) method

As it can be seen from the measurement data presented (Fig. 3-3) the AOD method does not cope sufficiently well with pathological cases,

what makes that the results obtained are not reliable in diagnostic terms. What is even worse, using this method in accordance with the definition a doctor makes consciously a pretty large (depending on the degree of pathology) error of the method. Therefore an 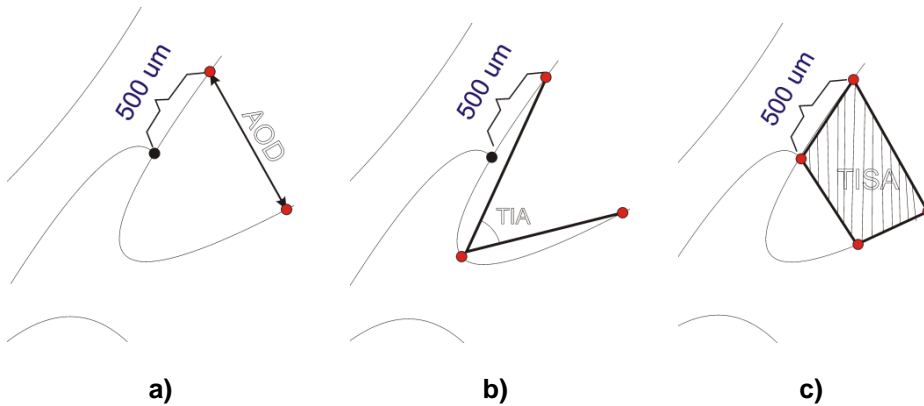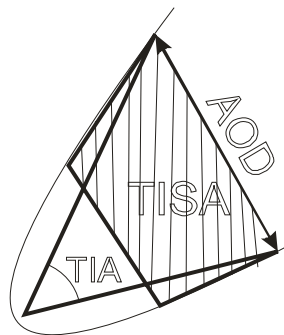automatic method for the filtration angle measurement has been proposed and an original measurement method (based on the aforementioned AOD, TISA and TIA) free of the errors mentioned above. However, further considerations should be preceded by showing the hitherto methods, which are comprised by the software delivered with an OCT instrument.

3.2   Review of Hitherto Filtration Angle Analysis Methods

The hitherto filtration angle analysis methods may be easily assessed, because in each software attached to each tomograph these are manual methods. An operator indicates reference points characteristic of specific measurement method (Fig. 3-5). Partial automation of angle analysis method by "dragging" the marked measuring line to the object contour is rare. However, irrespective of whether the method is computer assisted or fully manual the measurement is not automated and its result is affected by the precision of point indication by the operator. Hence these methods are not free of errors, both of the operator and of the measurement methodology itself. The error related to the lack of measurements repeatability is especially troublesome at statistical calculations.
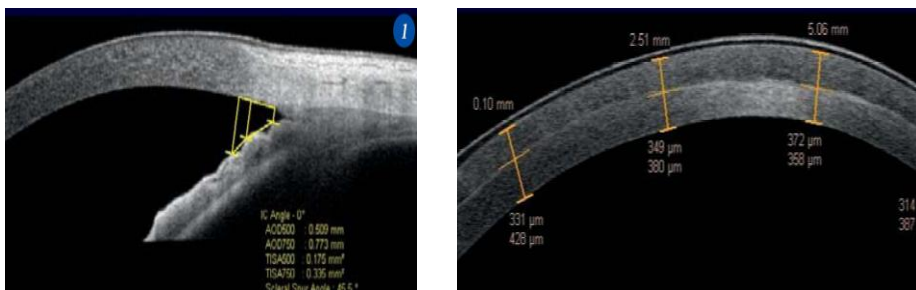

Fig. 3-5 Fragments of commercial software [38], attached to OCT Visante instruments, operation [37]

Summing up the software available now has the following deficiencies:

- missing 3D reconstruction and thereby a possibility to perform calculations of the volume of selected parts of anterior eye segment,

- missing full automation,
- calculations, which may be carried out manually, are possible only to a very limited extent,
- large measurement errors e.g. in the case of filtration angle measurement for pathological conditions.

Taking into account the aforementioned deficiencies an own profiled algorithm has been suggested, designed for automated analysis excluding any involvement of the operator. The description of the algorithm itself and of its parameters has been preceded by sections on reading the files from OCT instruments and the assessment of errors at manual measurements.

3.3   Verification of the Sensitivity of the Proposed Methods

This section is aimed at the analysis of properties (mainly sensitivity to parameters change) of methods specified in the previous section (AOD, TIA and TISA).

The need to evaluate and verify the precision of individual measurement methods at the presence of disturbances results from situations occurring in the case of inaccurate manual method for indication of characteristic points coordinates (marked red in Fig. 3-3). The location of points mentioned strictly depends on the measurement method chosen and on operator's accuracy and is forced by all types of software delivered by the OCT vendor. The calculated values of errors obtainable at manual points indication are the subject of these considerations. A reliable final result, documented by error values, consists of analysed method (AOD, TIA, TISA) sensitivity to operator's error. Conditions related strictly to the operator have been formulated in the summary based on that and referring to the fact, which coordinate of a point indicated by the operator in what way affects the final error of the filtration angle measurement.

3.3.1   Methodology for Measuring Methods Sensitivity to Parameters Change

The verification of AOD, TIA and TISA methods sensitivity to parameters change (operator's error) was carried out, likewise in the previous section, taking into account and not taking into account semi-automation implemented in commercial software. Semi-automatic

marking of points characteristic of individual methods is related to dragging the point marked to the edge, most often using the active contour method. However, in both methods mentioned the result is affected by the place indicated by the operator. Preliminary measurements have confirmed, depending on the operator, an error of points indication of around ± 10 pixels, giving an error at the resolution of 32 pixels/mm of around ± 0.31 mm. For the sake of comparison of sensitivity to parameters change for AOD, TIA and TISA methods the scope of analysis and comparisons has been narrowed to two points p_1 and p_2 (Fig. 3-6).
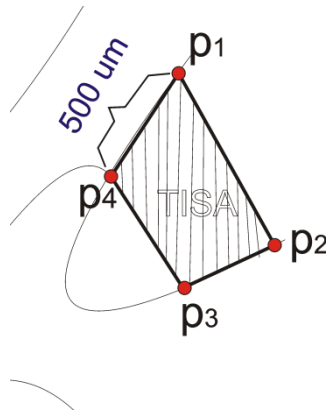
Fig. 3-6 Location of points p_i indicated by the operator
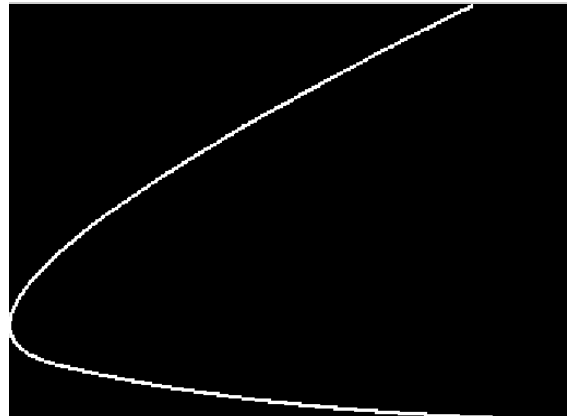
Fig. 3-7 Binary test image illustrating the filtration angle

On this basis the following assumptions related to the studies carried out have been formulated:

- the range of characteristic points position variability ± 10 pixels,
- verified software in semi-automatic version,
- analysis, due to comparative reasons, narrowed to points p1 and p2,
- the analysed image resolution of 32 pixels/mm.

The measurement error calculated for the AOD method − δ_{AOD}, for TIA − δ_{TIA}, for TISA − δ_{TISA} will be calculated as the difference between the measured and the correct value, expressed as the percentage of notional value, where the notional values is most often understood as the correct value, i.e.:

$$\delta_{AOD} = \frac{d_M - d_w}{d_w} \bullet 100 \ [\%], \quad \delta_{TIA} = \frac{\alpha_M - \alpha_w}{\alpha_w} \bullet 100 \ [\%], \tag{1}$$

$$\delta_{TISA} = \frac{s_M - s_W}{s_W} \bullet 100 \ [\%]$$

where:

d_M, d_W- measured and standard distance, respectively, defined as:

$$d_M = \sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2} \quad \text{and} \atop d_w = \sqrt{(y_{w1} - y_{w2})^2 + (x_{w1} - x_{w2})^2}$$ (2)

α_M, α_W- measured and standard angle, respectively

s_M, s_W − measured and standard area, respectively.

The method sensitivity to parameters change will be understood as a change of the measured value caused by a change of one parameter, indicated by the points operator, referred to the measured value and expressed as percentage, i.e.:

$$S_{AOD(p_i)} = \frac{\delta d_M}{d_M \bullet \delta x_i} \bullet 100 \ [\%]$$ (3)

for small increments it is possible to write

$$S_{AOD(p_i)} \approx \frac{\Delta d_M}{d_M \bullet \Delta d_i} \bullet 100 \ [\%]$$ (4)

where:

x_i − coordinates of points p_i indicated by the operator, the next i-th number in accordance with Fig. 3-6 (in accordance with the assumptions only two points, p_1 and p_2, are analysed).

Appropriately for the other methods:

$$S_{TIA(p_i)} \approx \frac{\Delta \alpha_M}{\alpha_M \bullet \Delta \alpha_i} \bullet 100 \ [\%] \qquad S_{TISA(p_i)} \approx \frac{\Delta s_M}{s_M \bullet \Delta s_i} \bullet 100 \ [\%]$$ (5)

The calculations have been carried out for an artificial image shown in Fig. 3-7, which may be downloaded from this book site http://robert.frk.pl and which, after entering to the Matlab space, should be converted to sorted coordinates x,y, i.e.:

```
L1=imread('D:\OCT\reference.jpg');
L1=1-double(L1(:,:,1)>128);
figure; imshow(L1);
[xx,yy]=meshgrid(1:size(L1,2),1:size(L1,1));
yy(L1~=1)=[];
xx(L1~=1)=[];
xy=sortrows([xx',yy'],2);
```

```
podz=750;

xl=xy(1:podz,1);
yl=xy(1:podz,2);
xp=xy(podz:end,1);
yp=xy(podz:end,2);
figure; plot(xl,yl,'-r*'); hold on; grid on
plot(xp,yp,'-g*'); xlabel('x'); ylabel('y')
```

From the notation we obtain coordinates (x_l, y_l) and (x_p, y_p) of the left and right hand side of the measured angle, respectively.

The next section will present the results obtained using this artificial image.

3.3.2   Methods Sensitivity to Parameters Change

Measurements were carried out changing the position of points p_1 and p_2 in coordinate x within $x_w \pm 10$ pixels, assuming automated dragging to the contour line on the y axis (Fig. 3-8). An example of measured quantities values variability range for individual methods has been shown in the following graphs (Fig. 3-9 - Fig. 3-11).
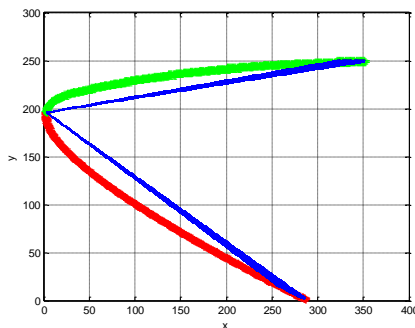
Fig. 3-8 Contour obtained from the image from Fig. 3-7 with marked range of points p_1 and p_2 fluctuation on x axis
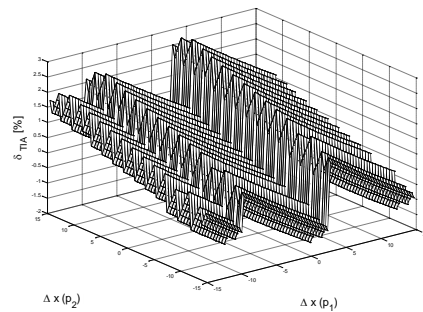
Fig. 3-9 Graph of δ_{TIA} error values changes vs. changes of p_1 and p_2 points position on the x axis within the range of $x_w \pm 10$ pixels
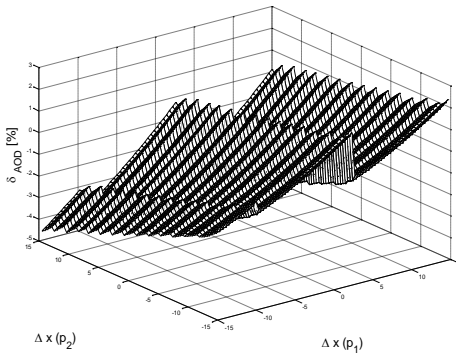
Fig. 3-10 Graph of δ_{AOD} error values changes vs. changes of p_1 and p_2 points position on the x axis within the range of $x_w \pm 10$ pixels
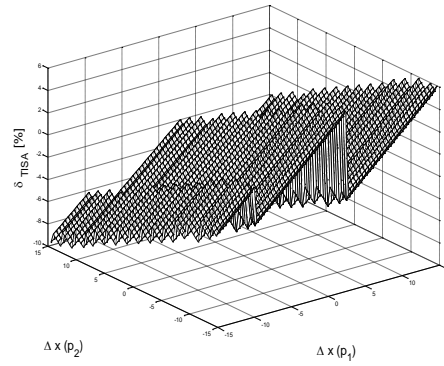
Fig. 3-11 Graph of δ_{TISA} error values changes vs. changes of p_1 and p_2 points position on the x axis within the range of $x_w \pm 10$ pixels

For coordinates of contour right and left hand side position the errors may be calculated as follows:

```
prz=30;
pam=[]; pam0=[];
xx=xp(end,:)-xp(:,:); yp(xx>(prz))=[]; yyyp=yp;
xx=xp(end,:)-xp(:,:); xp(xx>(prz))=[]; xxxp=xp;
xx=xl(1,:) - xl(:,:); yl(xx>(prz))=[]; yyyl=yl;
xx=xl(1,:) - xl(:,:); xl(xx>(prz))=[]; xxxl=xl;
po2p=round(length(xxxp)/2);
po2l=round(length(xxxl)/2);
pam=[];
for pol=1:length(xxxl)
    for pop=1:length(xxxp)
        xl=xy(1:podz,1);
        yl=xy(1:podz,2);
        xp=xy(podz:end,1);
        yp=xy(podz:end,2);
        xxl=xl(end):xxxl(pol);
        xxp=xp(1):xxxp(pop);
        Pl = POLYFIT([xl(end) xxxl(pol)],[yl(end)
yyyl(pol)],1);
        Yl = POLYVAL(Pl,xxl);
        Pp = POLYFIT([xp(1) xxxp(pop)],[yp(1)
yyyp(pop)],1);
        Yp = POLYVAL(Pp,xxp);
        plot(xxl,Yl)
        plot(xxp,Yp)
        katl=180+atan2([xl(end)-xxxl(pol)],[yl(end)-
yyyl(pol)])*180/pi;
```

```
        katp=atan2([xxxp(pop)-xp(1)],[yyyp(pop)-
yp(1)])*180/pi;
        pam(pol,pop)=katl-katp;
        if (pop==po2p)&(pol==po2l)
            pam00=katl-katp;
        end
    end
end
pam=(pam-pam00)./pam00*100;
sl=round(size(pam,1));
sp=round(size(pam,2));
[xx,yy]=meshgrid((1:sp)./sp*30-15, (1:sl)./sl*30-15);
figure; mesh(xx,yy,pam);
xlabel('\Delta x (p_1) [pixel]','fontSize',20);
ylabel('\Delta x (p_2) [pixel]','fontSize',20);
zlabel('\delta _{TIA} [%]','fontSize',20)
axis([-15 15 -15 15 min(pam(:)) max(pam(:))])
colormap([0 0 0])
```

The results obtained, for three methods AOD, TIA and TISA, of error value and of sensitivity to change of points p_1 and p_2 position are shown in the table below.

Tab 3-1 Table of methods sensitivity to points positions change

Method	$S_{(P_1)}[\%]$	$S_{(P_2)}[\%]$
AOD	*0.12*	*≈0*
TIA	*0.35*	*0.04*
TISA	*0.55*	*-0.25*

The table above and the graphs presented (Fig. 3-9 - Fig. 3-11) show the measurement error for individual methods, AOD, TIA and TISA, when changing positions of points p_1 and p_2 in the x coordinate, assuming "dragging" by a semi-automatic to the correct y coordinate. The measurement error, at incorrect indication of points p_1 and p_2 position for AOD and TISA methods, affects the result with the sign opposite to that for the TIA method. When moving point p_1 or p_2 towards the filtration angle, the measurement value is understated for AOD and TISA methods and overstated for the TIA method.

As it can be seen from the graphs presented and from the method sensitivity (Tab 3-1) to a change of the points mentioned, the TISA method is the most sensitive to operators errors. The sensitivity value of around 0.55% for TISA results from the nature of measurement, where very small changes of point p_1 and p_2 position have a significant impact on the calculated area. The AOD methods is the least sensitive to operators error, because a change of point p_1 and p_2 position on a contour

nearly parallel to the line, which length is calculated, affects the result only slightly.

The results obtained admittedly show an advantage of AOD method, in which a change of points position by the operators affects the total error to the least extent and at the same time this method is least sensitive to operators errors, but only in cases of ideal determination of the contour. Unfortunately it turns out that in the case of disturbances, personal variability and other factors causing sudden local contour changes/fluctuations, the situation is slightly different (Fig. 3-12 - Fig. 3-15). The disturbances may be added like in the case of calculations in the previous section, i.e.:

```
xyrand=rand(size(xy))*40;
xy=xy+xyrand;
```
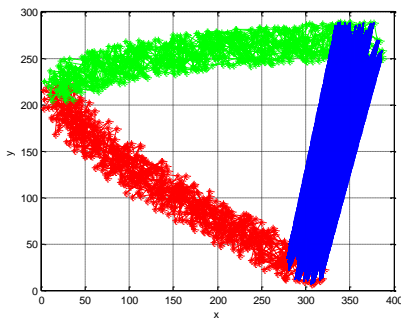

Fig. 3-12 Contour obtained from the image from Fig. 3-7 after adding noise of uniform distribution on ±20 range with marked range of points p_1 and p_2 fluctuation on the x axis
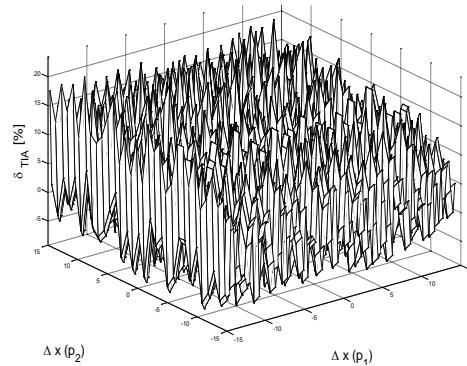
Fig. 3-13 Graph of δ_{TIA} error values changes vs. changes of p_1 and p_2 points position on the x axis within the range of x_w±10 pixels
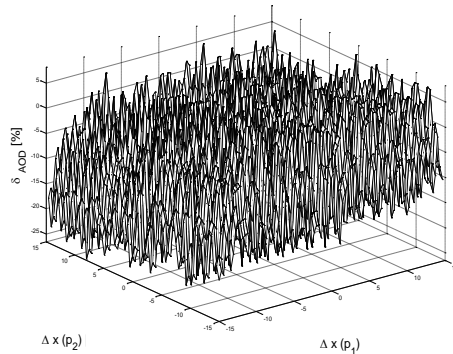
Fig. 3-14 Graph of δ_{AOD} error values changes vs. changes of p_1 and p_2 points position on the x axis within the range of $x_w \pm 10$ pixels
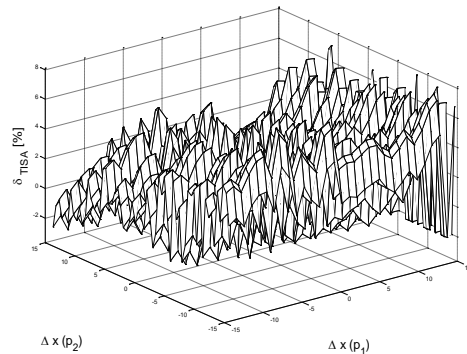
Fig. 3-15 Graph of δ_{TISA} error values changes vs. changes of p_1 and p_2 points position on the x axis within the range of $x_w \pm 10$ pixels

The above measurements were carried out for the same contour (Fig. 3-7) adding disturbances of random nature and uniform distribution on ±20 range, as a result obtaining contour shown in Fig. 3-12 and results as error values δ_{AOD}, δ_{TIA}, δ_{TISA} shown in Fig. 3-13, Fig. 3-14 and Fig. 3-15. As it is seen from the graphs presented (Fig. 3-13, Fig. 3-14, Fig. 3-15) the error has totally different distribution for individual methods AOD, TIA and TISA than in the case from Fig. 3-9, Fig. 3-10 i Fig. 3-11. In the case of disturbances existence the lowest error value is achievable for the TISA method, the largest for the TIA method. Based on that the following summary may be formulated.

3.3.3   Conclusions From the Sensitivity Analysis Methods

For AOD, TIA and TISA methods of filtration angle measurement and for an example contour analysed as an ideal (possible approximated in the software attached to the tomography) and featuring random disturbances the conclusions presented in the following table may be drawn.

Tab 3-2 Summary of method errors impact for AOD, TIA and TISA measurements

METHOD	*Measurement error without added disturbances*	*Measurement error with added disturbances*	*When x_M increases*	*When x_M decreases*
AOD	*Small*	*Large*	d_M *increases*	d_M *decreases*
TIA	*Medium*	*Medium*	α_M *decreases*	α_M *increases*
TISA	*Large*	*Small*	s_M *increases*	s_M *decreases*

On the basis of presented Tab 3-2 the following premises may be drawn for the operator – the person indicating manually the measurement points (supported by a semi-automatic implemented in the software delivered with the instrument or not):

- the AOD method gives results burdened with the smallest error in the case of contour line approximation. Precise manual indication of measurement points makes this method to be the least accurate.
- the TIA method, irrespective of the way of operation, help, software delivered with the tomography, shows average error values at the indication of measurement points,
- the TISA method is burdened with the smallest error if the contour is not approximated and the operator indicates measurement points very precisely.

Summing up, the AOD method is the best for a contour in which the filtration angle measured is approximated by lines, in other cases it is the TISA method.

3.4 The Results of Automatic Analysis Chamber Angle Obtained Using Well-Known Algorithms

The justification of the necessity to use a profiled algorithm in this case is related with insufficient results obtained from other known algorithms intended for detection of lines and/or areas on images:

- the Hough transform enables detecting lines on images of predetermined shape.
- the wavelet analysis method gives incorrect results in the case, where the objects are poorly visible and lines can overlap,

- also the methods for elongated objects analysis are not applicable here due to a possibility of large change of dimensions both of the object itself and also of its thickness and to a possibility of its division to many parts.

Based on that, taking also into consideration the medical premises presented below, a profiled algorithm for analysis and processing of anterior eye segment has been proposed.

3.5 Proposed Modifications to the Well-Known Method of Measuring

Fig. 3-16 presents again the anterior chamber for different degrees of pathology with marked distances at various points for one selected method, i.e. AOD [31].
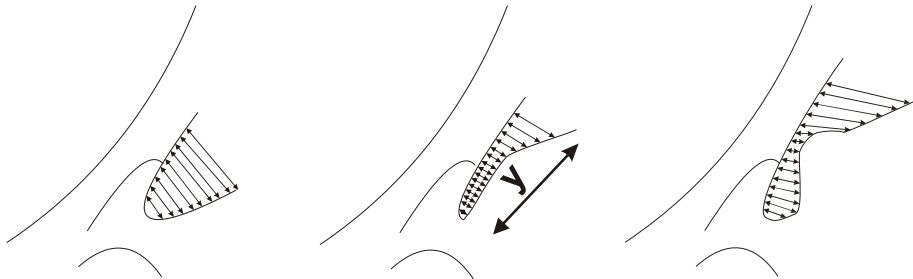
Fig. 3-16 The anterior chamber for different degrees of pathology with measured quantities for the AOD method and distance y marked with arrows

As it can be seen (Fig. 3-16) and as previously mentioned the AOD method does not cope sufficiently well with pathological cases, what makes that the results obtained are not reliable in diagnostic terms. The new method, proposed by the authors, consists in continuous measurements via modified AOD, TIA and TISA methods. A continuous measurement will be understood here as a series of measurements for a distance of 500 μm (Fig. 3-16) decreasing by 1 pixel. At a typical resolution of the image of 32 pixels/1 mm this gives on average around 16 measurements. Because of the resolution error the measurements for a small number of pixels are burdened with a larger error. However, this does not affect the advantage of the method proposed over the commonly used methods. For the measurement method defined this way its precision and sensitivity to disturbance have been verified. To this end

the shape of contour analysed and also x and y coordinates have been preliminary modelled, e.g. as follows.

```
figure
% green plot
x=[.1:0.1:4, -4:0.1:-0.1]; y=x.^2;
x(x<0)=x(x<0)*2; x(x<0)=fliplr(x(x<0));
x(x>0)=fliplr(x(x>0)); x=x-min(x);y=max(y)-y;
plot(x,y,'-gs'); grid on; hold on

% red plot
xs1=[sqrt(y)]; xs2=[sqrt(y)+8];
x=[flipud((8-(xs1*2)));(xs2)];
ys1=flipud(y); y=[ys1;(y)];
plot(x',y','-r+'); grid on; hold on

% blue plot
x=[-4:0.1:0,.1:0.1:4];
y=x.^2; y(x<=0)=[]; x(x<=0)=[];
x(x>0)=fliplr(x(x>0)); x=x+8; y=max(y)-y;
y_=fliplr(y/max(y)*3*pi); x_=1*cos(y_)-1;
x__=0:(6/(length(x_)-1)):6;
x=[x,x_+8-x__]; y=[y,y_/3/pi*16];
plot(x,y,'-b+'); grid on; hold on
```

For each of these curves the filtration angle was calculated according to individual AOD, TIA and TISA methods, i.e.

```
xl=[]; xp=[];
TIA=[];
TISA=[];
AOD=[];
xr=8; yr=0;
    for i=round(length(x)/2):-1:1
        line([x([i,length(x)-i+1])], [y([i,length(x)-
i+1])],'Color',[0 1 0])
        Pl = POLYFIT([xr x(i)],[yr y(i)],1);
        Pp = POLYFIT([xr x(length(x)-i+1)],[yr y(length(x)-
i+1)],1);
        TIA=[TIA; [y(i) -atan(Pp(1)-Pl(1))*180/pi]];
        AOD=[AOD; [y(i) -(x(length(x)-i+1)-x(i))]];
        TISA=[TISA; [y(i) sum(AOD(:,2))]];
    end

figure;
plot(AOD(:,1), AOD(:,2)./max(max([AOD(:,2)])),'-r+');
hold on; grid on
xlabel('y [piksel]');
ylabel('D (AOD), D (TISA),D (TIA), [\\]')
plot(TISA(:,1), TISA(:,2)./max(max([TISA(:,2)])),'-g+');
```

```
plot(TIA(:,1), TIA(:,2)./max(max([TIA(:,2)]))),'-b+');
legend('AOD','TISA','TIA')
```

The results obtained at pathologies for a diminishing distance y for images in Fig. 3-16 have been shown in the following figures.
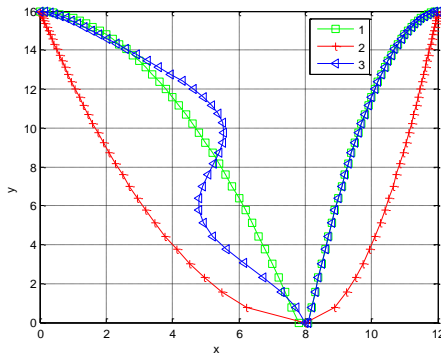
Fig. 3-17 Contours of the filtration angle measured for three examples of patients
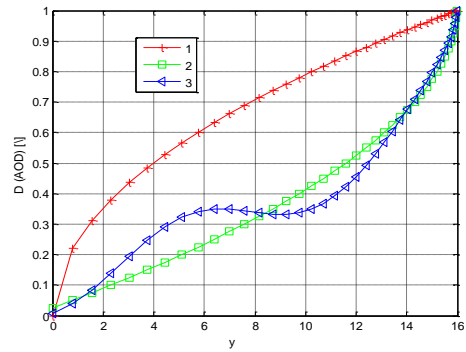
Fig. 3-18 Values of distance D measurements for the AOD method vs. y for different shapes of the filtration angle (Fig. 3-17)
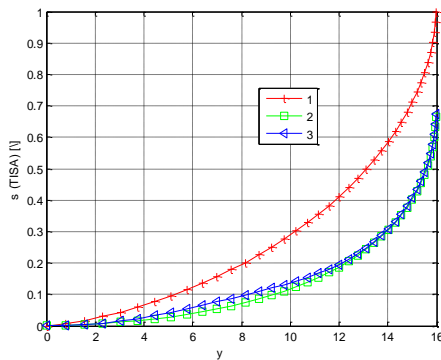
Fig. 3-19 Values of area s measurements for the TIA method vs. y for different shapes of the filtration angle (Fig. 3-17)
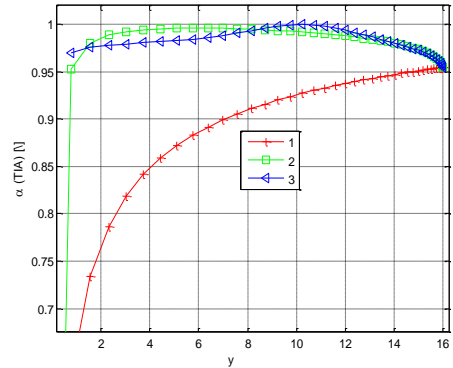
Fig. 3-20 Values of angle α measurements for the TIA method vs. y for different shapes of the filtration angle (Fig. 3-17)

The results obtained for an actual image case with the presence of noise (random steady interference in the 0÷1 range) are presented in the following figures (Fig. 3-21 - Fig. 3-24)
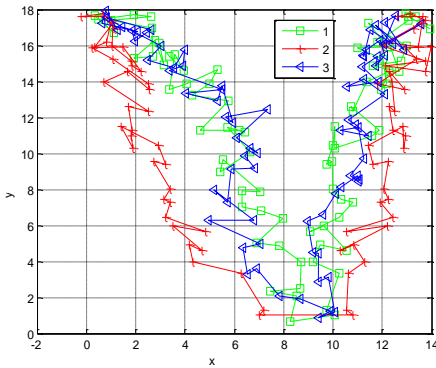
Fig. 3-21 Contours of the filtration angle measured for three examples of patients, together with noise
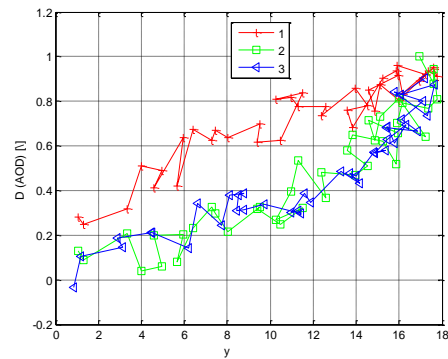
Fig. 3-22 Values of distance D measurements for the AOD method vs. y for different shapes of the filtration angle (Fig. 3-21).
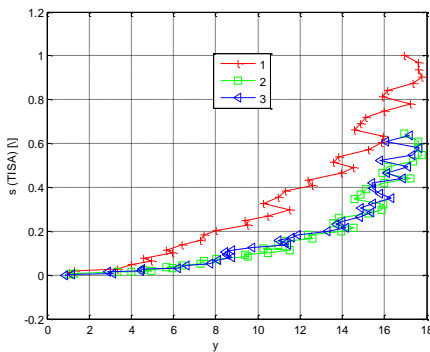
Fig. 3-23 Values of area s measurements for the TIA method vs. y for different shapes of the filtration angle (Fig. 3-17).
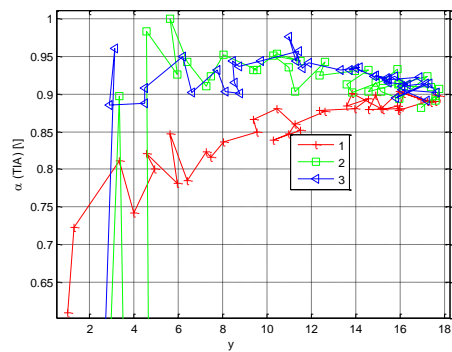
Fig. 3-24 Values of angle α measurements for the TIA method vs. y for different shapes of the filtration angle (Fig. 3-17)

The disturbance was random added as follows:

```
x=x+rand(size(x))*2;
y=y+rand(size(y))*2;
```

The following conclusions may be drawn from the graphs presented above:

- increasing value of y (place of the measurement) to the least extent affects the results obtained from the TIA method – Fig. 3-20 and Fig. 3-24.

- the noise introduced to the contour of the image measured to the least extent affects the measurement by the TISA method;
- in the cases of moving the place of measurement, of increasing the value of y, the results of measurements for all TISA and AOD methods are overestimated, while for the TIA method they strictly depend on the shape of the contour measured (Fig. 3-20).
- changes of the shape of the analysed image contour only slightly affect the results obtained from the TIA method;
- the TISA method, stable in terms of the occurring noise (Fig. 3-23), has a drawback in the form of a nonlinear dependence of the measurement results on the place of measurement – the value of y. As it results from Fig. 3-23 this nonlinearity causes sudden changes of the measured value for the increasing values of y.
- the drawback of the method used consists of necessary full automation of the measurement due to high amounts of time consumed for individual calculations.

Summing up, the AOD method is the best for a contour in which the filtration angle measured is approximated by lines, in other cases it is the TISA method.

The method proposed, because of the laborious obtaining of partial results, requires a full automation of the measurement.

So it is already known, which of methods is most appropriate in terms of sensitivity to personal characteristics (degree of pathology); further on it is interesting to assess the sensitivity to change of parameters, but set by the operator (characteristic points indication). These are measurements necessary to assess the precision obtained during manual measurement of parameters.

3.6 Algorithm for Automated Analysis of the Filtration Angle

Two main directions of algorithm operation:

- automated calculation of the filtration angle,
- automated determination of sclera layers for 3D reconstruction.

On the basis of the above medical premises [21], [30] and preliminary tests performed the following block diagram of the algorithm has been suggested (Fig. 3-25).

Acquisition of an 256x1024 image
Filtering by median filter, mask size 3x3
Analysis of maximum value in all columns
Automatic choice of threshold
Filling the holes
Detection of limits of the scleris
Approximation of the inside and outside of the anterior part
Analysis of the iris and ciliary process
Detection of boundary points of the iris
Calculation with TIA, AOD and TISA methods
Result

Fig. 3-25 Block diagram of the algorithm

As mentioned in the introduction the input image of 256x1024 resolution and on average of 0.0313 mm/pixel is entered in the DICOM format to the Matlab software space. The source code may be divided into two parts: the readout of the file as a set of bytes and the conversion to one of image recording formats including acquiring necessary information from the header.

The readout of 3.dcm file was carried out in accordance with the information provided in the initial section, i.e.:

```
path_name='d:/OCT/SOURCES/3.DCM';
fid = fopen(path_name, 'r');
dataa = fread(fid,'uint8');
fclose(fid);
[header_dicom,Ls]=OCT_head_read(dataa);
```

Further on the algorithm comprises filtration using a median filter of Ls image of 3x3 mask size, changing resolution to accelerate calculations and individual columns analysis [32].

```
Ls=medfilt2(Ls,[7 7]);
Ls=imresize(Ls,[256 512 ]);
figure; imshow(Ls,[]);
L2=Ls
```

This analysis results in the calculation for each column of the binarisation threshold (images are calibrated) as 10% of the brightest of the existing pixels (Fig. 3-26) i.e.:

```
przed=1;
```

```
L22=imclose(Ls>max(max(Ls))*0.10,ones([3 3]));
```

The binary values for each column are consecutively analysed considering the criterion of the largest object length. An example record to delete all objects larger than 100 pixels looks as follows:

```
L2lab=bwlabel(L22);
L33=zeros(size(L22));
for ir=1:max(L2lab(:))
    L2_=L2lab==ir;
    if sum(L2_(:))>100
        L33=L33|L2_;
    end
end
figure; imshow(L33,[]);
```

Then – to eliminate small inclusions and separations of layers – a method of holes filling is implemented.

```
L22=bwfill(L33,'holes');
figure; imshow(L22,[]);
L55=bwlabel(xor(L22,L33));
for ir=1:max(L55(:))
    L5_=L55==ir;
    if sum(L5_(:))<100
        L33=L33|L5_;
    end
end
L22=L33;
figure; imshow(L22,[]);
```

Obviously in this case the function bwfill (…,'holes') would be sufficient itself, however, all holes would be filled and not only those, which have the number of pixels (area) smaller than 100.

The image preliminary prepared in this way is used to perform the operation of the sclera boundaries determination and the approximation of the boundaries determined by a third degree polynomial (Fig. 3-30).:

```
linie_12=[];
for i=1:size(L22,2)
    Lf=L22(:,i);
    Lff=bwlabel(Lf);
    if sum(Lff)>0
        clear Lnr
        for yt=1:max(Lff(:))
            Lffd=Lff==yt;
            if sum(Lffd(:))>10
                Lnr=[(1:length(Lffd))',Lffd];
Lnr(Lnr(:,2)==0,:)=[];
                break
```

```
            end
        end
        if (exist('Lnr')>0)&(~isempty(Lnr))
            linie_12=[linie_12; [i Lnr(1,1) Lnr(end,1)]];
        end
    end
end
hold on;
plot(linie_12(:,1),linie_12(:,2),'r*'); grid on
plot(linie_12(:,1),linie_12(:,3),'g*'); grid on
```

The next stage is the filtration using a median filter, i.e.:

```
linie_12(:,2)=medfilt2(linie_12(:,2),[5 1]);
linie_12(:,3)=medfilt2(linie_12(:,3),[5 1]);
```

The obtained values of (x,y) coordinates in the variable linie_12 are analysed with regard to differences in oy axis exceeding the threshold set, e.g. 5 pixels (selected taking into account medical premises), i.e.:

```
x=linie_12(:,1);
y=linie_12(:,2);
ybw=bwlabel(abs([diff(y') 0])<5);
```

For each pair of coordinate sets obtained for all combinations of labels, the approximation by a third degree polynomial is performed.

```
    rzad=3;
    toler=10;
P=polyfit(x,y,rzad);
    Y=polyval(P,x);
    yyy=Y-y;
pamm=[0 0 sum( abs(yyy)<toler )/length(yyy)];
for ir=1:(max(ybw)-1)
    for irr=(ir+1):max(ybw)
        y_=[y(ybw==ir); y(ybw==irr)];
        x_=[x(ybw==ir); x(ybw==irr)];
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
        hold on; plot(x,Y,'-g*');
        yyy=Y-y;
        pamm=[pamm; [ir irr sum( abs(yyy)<toler
)/length(yyy) ]];
    end
end
```

Then this combination of such pairs of coordinate sets is chosen, for which around the tolerance set.

```
pamm_=sortrows(pamm,3); ir=pamm_(end,1); irr=pamm_(end,2);
    if ir==0;
        y_=y;
```

```
        x_=x;
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
        yyy=Y-y;
        y_=y(abs(yyy)<toler);
        x_=x(abs(yyy)<toler);
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
    else
        y_=[y(ybw==ir); y(ybw==irr)];
        x_=[x(ybw==ir); x(ybw==irr)];
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
        yyy=Y-y;
        y_=y(abs(yyy)<toler);
        x_=x(abs(yyy)<toler);
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
    end
plot(x,Y,'b*'); grid on
```

After the analysis of the iris and of the ciliary processes the analysis of iris endings is carried out, using the information originating within the sclera boundaries (Fig. 3-31).

Fig. 3-26 A binary image originated from the original image after the binarisation with a threshold of 90% of the maximum value
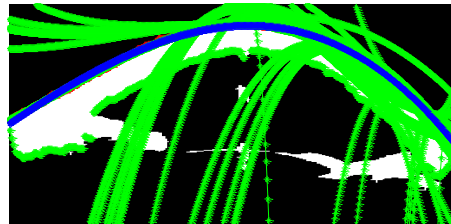
Fig. 3-27 A binary image after the operation of holes filling with approximation lines marked green, and the best fit marked blue

The contour of internal boundary is analysed in a similar way:

```
y_1=Y;
x=linie_12(:,1);
y=linie_12(:,3);
ybw=bwlabel(abs([diff(y') 0])<5);
rzad=3; toler=15;
P=polyfit(x,y,rzad); pamm=[];
    Y=polyval(P,x);
    yyy=Y-y;
        if sum( (Y(:)-y_1(:))<0      )==0
```

```
            pamm=[0 0 sum( abs(yyy)<toler )/length(yyy)];
        end
 for ir=1:(max(ybw)-1)
    for irr=(ir+1):max(ybw)
        y_=[y(ybw==ir); y(ybw==irr)];
        x_=[x(ybw==ir); x(ybw==irr)];
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
        yyy=Y-y;
        if sum( (Y(:)-y_1(:))<0     )==0
            pamm=[pamm; [ir irr sum( abs(yyy)<toler
)/length(yyy) ]];
        end
    end
 end
if  size(pamm,1)>1
pamm_=sortrows(pamm,3);
ir=pamm_(end,1); irr=pamm_(end,2);

    if ir==0;
        Y_=y;
        x_=x;
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
        yyy=Y-y;
        y_=y(abs(yyy)<toler);
        x_=x(abs(yyy)<toler);
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
    else
        y_=[y(ybw==ir); y(ybw==irr)];
        x_=[x(ybw==ir); x(ybw==irr)];
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
        yyy=Y-y;
        y_=y(abs(yyy)<toler);
        x_=x(abs(yyy)<toler);
        P=polyfit(x_,y_,rzad);
        Y=polyval(P,x);
    end
else
    x=[]; Y=[];P=[];
end
plot(x,Y,'m*'); grid on
y_2=Y;
```

The results of contour analysis are shown in (Fig. 3-28). The input image within approximated boundaries, red – the approximation result

marked blue, and green – the approximation result marked white, respectively.
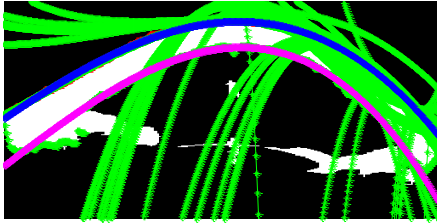
Fig. 3-28 The input image with detected boundaries of anterior eye segment marked red and green

Fig. 3-29 OCT image of the anterior eye part with marked analysis area (red and turquoise)

The next phases of algorithm operation consist in analysing the area situated under the contour marked red in Fig. 3-28. Because of that it is necessary to draw a straight line normal to the tangent at each point of the contour. The algorithm performing such calculations is shown below, while the results in Fig. 3-29 i Fig. 3-30.

```
figure; imshow(L33,[]); hold on
    sf=zeros( [ 1 length(Y) ]  ); sf_=zeros( [ 1 length(Y)
] );pole_=zeros([1 length(Y)]);
    pole_x=zeros([1 length(Y)]);  pole_y=zeros([1
length(Y)]);
      p_zn=0; zakres_=40;
Lwys=zeros([zakres_ length(Y)-1]);
Lwys_bin=zeros([zakres_ length(Y)-1]);
L_gridXX=[]; L_gridYY=[];
for nb=1:(length(Y)-1)
    PP=polyfit(x(nb:nb+1),Y(nb:nb+1),1);
    PP2(2)=x(nb)/PP(1)+Y(nb);
    PP2(1)=-1/PP(1);
    if Y(nb)>Y(nb+1)
       XX=x(nb):1:(x(nb)+zakres_);
    else
       XX=x(nb):-1:(x(nb)-zakres_);
    end
    YY=polyval(PP2,XX);
    if  (max(YY)-min(YY))>(zakres_+1)
        YY=Y(nb):1:(Y(nb)+zakres_);
        PP3(1)=1/PP2(1);
        PP3(2)=-PP2(2)/PP2(1);
        XX=polyval(PP3,YY);
       plot(XX,YY,'r*'); grid on; hold on
       XX(round(YY)>size(L2,1))=[];
YY(round(YY)>size(L2,1))=[];
```

```
        YY(round(XX)>size(L2,2))=[];
XX(round(XX)>size(L2,2))=[];
        for vc=1:length(XX);   if
(round(YY(vc))>0)&(round(XX(vc))>0)  ;Lwys(vc,nb)=L2(
round(YY(vc)), round(XX(vc))  ); Lwys_bin(vc,nb)=L22(
round(YY(vc)), round(XX(vc))  );    end; end
        L_gridXX(1:length(XX),nb)=XX;
L_gridYY(1:length(YY),nb)=YY;
    else
        plot(XX,YY,'c*'); grid on; hold on
        XX(round(YY)>size(L2,1))=[];
YY(round(YY)>size(L2,1))=[];
        YY(round(XX)>size(L2,2))=[];
XX(round(XX)>size(L2,2))=[];
        for vc=1:length(XX);   if
(round(YY(vc))>0)&(round(XX(vc))>0); Lwys(vc,nb)=L2(
round(YY(vc)), round(XX(vc))  ); Lwys_bin(vc,nb)=L22(
round(YY(vc)), round(XX(vc))  ); end; end
        L_gridXX(1:length(XX),nb)=XX;
L_gridYY(1:length(YY),nb)=YY;
    end
end
figure; imshow(Lwys,[]);
```
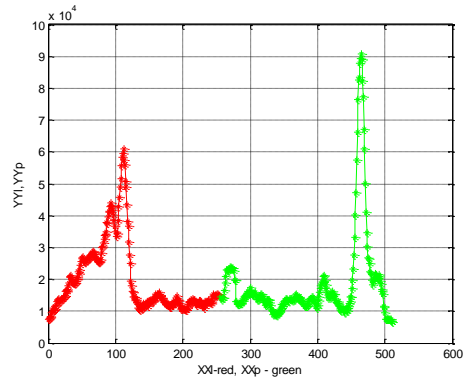

Fig. 3-30   The image of separated analysed area Lwys

Fig. 3-31 The diagram of sum of pixel brightness values for individual columns (XXI,YYI) – red (XXp,YYP) – green

The image originated from marked area pixels is analysed in the next stage of algorithm operation. The area is divided into two equal parts and the filtration angle is analysed independently in each of them. This is the last common part of algorithm for both angles calculation.

```
Lwys_bin=imopen(Lwys_bin,ones(3));
Lss=sum(Lwys);
XX=1:length(Lss);
YY=Lss;
Lm=max(Lss(:));
XXl=XX(1:round(length(XX)/2));
XXp=XX(round(length(XX)/2):end);
YYl=YY(1:round(length(YY)/2));
YYp=YY(round(length(YY)/2):end);
YYlb=YYl>(Lm/4);
YYpb=YYp>(Lm/4);
nr_XXl=1:length(XXl);
nr_XXp=1:length(XXp);
XXl_max=nr_XXl(YYl==max(YYl));
XXp_max=nr_XXp(YYp==max(YYp));
 figure
plot(XXl,YYl,'-r*'); hold on
plot(XXp,YYp,'-g*'); grid on
xlabel('XXl-red, XXp - green')
ylabel('YYl,YYp')
```

The obtained diagram of sum values calculated for individual columns is presented in Fig. 3-31.

An automated finding of the filtration angle vertex and determination of the correspondence between contour points (pixels forming the angle edges) is one of more difficult fragments of the algorithm operation. This analysis was started from automated finding the place on the contour, in which a normal to the tangent `zakres_=40` long for the first time comprises a ciliary process, i.e.:

```
YYlb_=bwlabel(YYlb);
pam_l=[];
for ty=1:max(YYlb_)
    YYt=YYl(YYlb_==ty);
    XXt=XXl(YYlb_==ty);
    pam_l=[pam_l; [ty sum(YYt) XXt(end)]];
end
if size(pam_l,1)>0
    pam_l=pam_l(YYlb_(XXl_max),:);
    plot(pam_l(1,3),Y(pam_l(1,3)),'rs','MarkerSize',10)
end
```

Further on, having the contour point mentioned, the fragment comprising the interesting measured filtration angle is analysed. For the filtration angle situated on the left-hand side of the image the algorithm has the form:

```
xy_g_l=[];
```

```
xy_d_l=[];
for vv=pam_l(1,3):-1:1
    pp=Lwys_bin(:,vv);
    ppl=bwlabel(pp);
    pam_lab=[];
    for jk=1:max(ppl)
        ppl_=ppl==jk;
        y_ppl=1:length(ppl_);
        y_ppl(ppl_==0)=[];
        pam_lab=[pam_lab;[vv jk y_ppl(1) sum(ppl_)
y_ppl(end)]];
    end
    if (size(pam_lab,1)>1)&(pam_lab(1,3)~=1);
                pam_lab(1,:)=[];
pam_lab=sortrows(pam_lab,4);
            if linie_12(round(L_gridXX(1,vv)),3)<
L_gridYY(pam_lab(end,3),vv)
                xy_g_l=[xy_g_l;[L_gridXX(1,vv)
linie_12(round(L_gridXX(1,vv)),3)]];
                xy_d_l=[xy_d_l;[L_gridXX(pam_lab(end,3),vv)
L_gridYY(pam_lab(end,3),vv) ]];
            end
    end
    if (size(pam_lab,1)==1)&(pam_lab(1,3)~=1);
            if
linie_12(round(L_gridXX(1,vv)),3)<L_gridYY(pam_lab(1,3),vv)
                xy_d_l=[xy_d_l;[L_gridXX(pam_lab(1,3),vv)
L_gridYY(pam_lab(1,3),vv) ]];
                xy_g_l=[xy_g_l;[L_gridXX(1,vv)
linie_12(round(L_gridXX(1,vv)),3)]];
            end
    end
    if (size(pam_lab,1)==2)&(pam_lab(1,3)==1);
            if
L_gridYY(pam_lab(1,5),vv)<L_gridYY(pam_lab(end,3),vv)
                xy_d_l=[xy_d_l;[L_gridXX(pam_lab(end,3),vv)
L_gridYY(pam_lab(end,3),vv) ]];
                xy_g_l=[xy_g_l;[L_gridXX(pam_lab(1,5),vv)
L_gridYY(pam_lab(1,5),vv) ]];
            end
            pam_lab(1,:)=[];
    end
    if (size(pam_lab,1)>2)&(pam_lab(1,3)==1);
            pam_lab(1,:)=[]; pam_lab=sortrows(pam_lab,4);
            if
L_gridYY(pam_lab(1,5),vv)<L_gridYY(pam_lab(end,3),vv)
                xy_g_l=[xy_g_l;[L_gridXX(pam_lab(1,5),vv)
L_gridYY(pam_lab(1,5),vv) ]];
                xy_d_l=[xy_d_l;[L_gridXX(pam_lab(end,3),vv)
L_gridYY(pam_lab(end,3),vv) ]];
```

```matlab
            end
    end
    if (size(pam_lab,1)==1)&(pam_lab(1,3)==1)
        pam_lab=pam_lab(1,1);
        break
    end
end
hold on; plot(pam_lab,Y(pam_lab),'k*'); hold on; grid on
if size(xy_g_l)>1
plot(xy_g_l(:,1),xy_g_l(:,2),'-y*')
plot(xy_d_l(:,1),xy_d_l(:,2),'-m*')
for ib=1:size(xy_g_l,1)
    line([xy_g_l(ib,1) xy_d_l(ib,1)],[xy_g_l(ib,2)
xy_d_l(ib,2)],'Color','y','LineWidth',1);
end
end
```

Instead, the algorithm analysing the filtration angle situated on the right-hand side of the image is provided below.

```matlab
YYpb_=bwlabel(YYpb);
pam_p=[];
for ty=1:max(YYpb_)
    YYt=YYp(YYpb_==ty);
    XXt=XXp(YYpb_==ty);
    pam_p=[pam_p; [ty sum(YYt) XXt(1)]];
end
if size(pam_p,1)>0
    pam_p=pam_p(YYpb_(XXp_max),:);
    plot(pam_p(end,3),Y(pam_p(end,3)),'rs','MarkerSize',10)
end
xy_g_p=[];
xy_d_p=[];
for vv=(pam_p(1,3)+1):size(Lwys_bin,2)
    pp=Lwys_bin(:,vv);
    ppl=bwlabel(pp);
    pam_lab=[];
    for jk=1:max(ppl)
        ppl_=ppl==jk;
        y_ppl=1:length(ppl_);
        y_ppl(ppl_==0)=[];
        pam_lab=[pam_lab;[vv jk y_ppl(1) sum(ppl_)
y_ppl(end)]];
    end
    if (size(pam_lab,1)>1)&(pam_lab(1,3)~=1);
                pam_lab(1,:)=[];
pam_lab=sortrows(pam_lab,4);
            if linie_12(round(L_gridXX(1,vv)),3)<
L_gridYY(pam_lab(end,3),vv)
```

```
                    xy_g_p=[xy_g_p;[L_gridXX(1,vv)
linie_12(round(L_gridXX(1,vv)),3)]];
                    xy_d_p=[xy_d_p;[L_gridXX(pam_lab(end,3),vv)
L_gridYY(pam_lab(end,3),vv) ]];
                end
        end
        if (size(pam_lab,1)==1)&(pam_lab(1,3)~=1);
                if
linie_12(round(L_gridXX(1,vv)),3)<L_gridYY(pam_lab(1,3),vv)
                    xy_d_p=[xy_d_p;[L_gridXX(pam_lab(1,3),vv)
L_gridYY(pam_lab(1,3),vv) ]];
                    xy_g_p=[xy_g_p;[L_gridXX(1,vv)
linie_12(round(L_gridXX(1,vv)),3)]];
                end
        end
        if (size(pam_lab,1)==2)&(pam_lab(1,3)==1);
                if
L_gridYY(pam_lab(1,5),vv)<L_gridYY(pam_lab(end,3),vv)
                    xy_d_p=[xy_d_p;[L_gridXX(pam_lab(end,3),vv)
L_gridYY(pam_lab(end,3),vv) ]];
                    xy_g_p=[xy_g_p;[L_gridXX(pam_lab(1,5),vv)
L_gridYY(pam_lab(1,5),vv) ]];
                end
                pam_lab(1,:)=[];
        end
        if (size(pam_lab,1)>2)&(pam_lab(1,3)==1);
                pam_lab(1,:)=[]; pam_lab=sortrows(pam_lab,4);
                if
L_gridYY(pam_lab(1,5),vv)<L_gridYY(pam_lab(end,3),vv)
                    xy_g_p=[xy_g_p;[L_gridXX(pam_lab(1,5),vv)
L_gridYY(pam_lab(1,5),vv) ]];
                    xy_d_p=[xy_d_p;[L_gridXX(pam_lab(end,3),vv)
L_gridYY(pam_lab(end,3),vv) ]];
                end
        end
        if (size(pam_lab,1)==1)&(pam_lab(1,3)==1)
            pam_lab
            pam_lab=pam_lab(1,1);
            disp('kuku')
            break
        end

end
hold on; plot(pam_lab,Y(pam_lab),'k*'); hold on; grid on
if size(xy_g_p)>1
plot(xy_g_p(:,1),xy_g_p(:,2),'-y*')
plot(xy_d_p(:,1),xy_d_p(:,2),'-m*')
for ib=1:size(xy_g_p,1)
    line([xy_g_p(ib,1) xy_d_p(ib,1)],[xy_g_p(ib,2)
xy_d_p(ib,2)],'Color','y','LineWidth',1);
```

```
end
end
```

Based on the presented algorithm fragment it is possible to analyse automatically the filtration angle determined by the yellow and turquoise lines

Fig. 3-32 The result of algorithm fragment automatically determining the walls contours (yellow and red) necessary to calculate the filtration angle – left-hand side
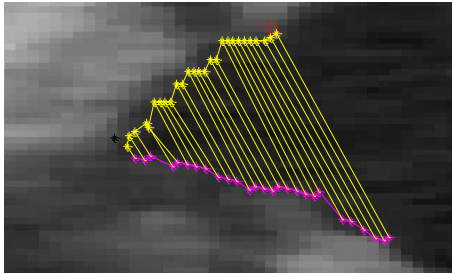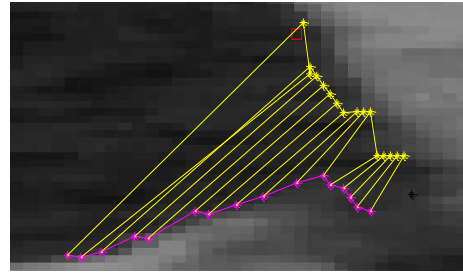

Fig. 3-33 The result of algorithm fragment automatically determining the walls contours (yellow and red) necessary to calculate the filtration angle – right-hand side

The filtration angle calculated traditionally as the angle between tangents formed from the edge lines (yellow and red colour - Fig. 3-33) may be implemented as follows:

```
PPgl=polyfit(xy_g_l(:,1),xy_g_l(:,2),1);
PPdl=polyfit(xy_d_l(:,1),xy_d_l(:,2),1);
PPgp=polyfit(xy_g_p(:,1),xy_g_p(:,2),1);
PPdp=polyfit(xy_d_p(:,1),xy_d_p(:,2),1);

x=1:size(L33,2);
y=polyval(PPgl,x); plot(x,y,'r*')
y=polyval(PPdl,x); plot(x,y,'g*')
y=polyval(PPgp,x); plot(x,y,'b*')
y=polyval(PPdp,x); plot(x,y,'y*')
al=atan(PPdl(1))*180/pi-atan(PPgl(1))*180/pi;
ap=atan(PPgp(1))*180/pi-atan(PPdp(1))*180/pi;
al=round(al*10)/10;
ap=round(ap*10)/10;
title(['Left - ',mat2str(al),'^o', '    Right -
',mat2str(ap),'^o'])
```

As a result, we obtain the filtration angle value calculated traditionally – these are values in `al` and `ap` variables.
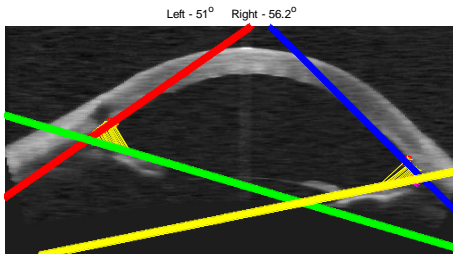
Fig. 3-34 The result of algorithm operation, where the inclination angle of straight lines relative to each other for the right and left filtration angle is the measured value

Fig. 3-35 Fragments selected from Fig. 3-34

Based on the algorithm, presented above, for the inter-sclera analysis it is possible to estimate the position of filtration angles and to calculate AOD, TISA and TIA values during approx. 3 s/image on a computer with a 64-bit operating system, Intel Core Quad CPU 2.5 GHz processor, 2GB RAM (Fig. 3-36).
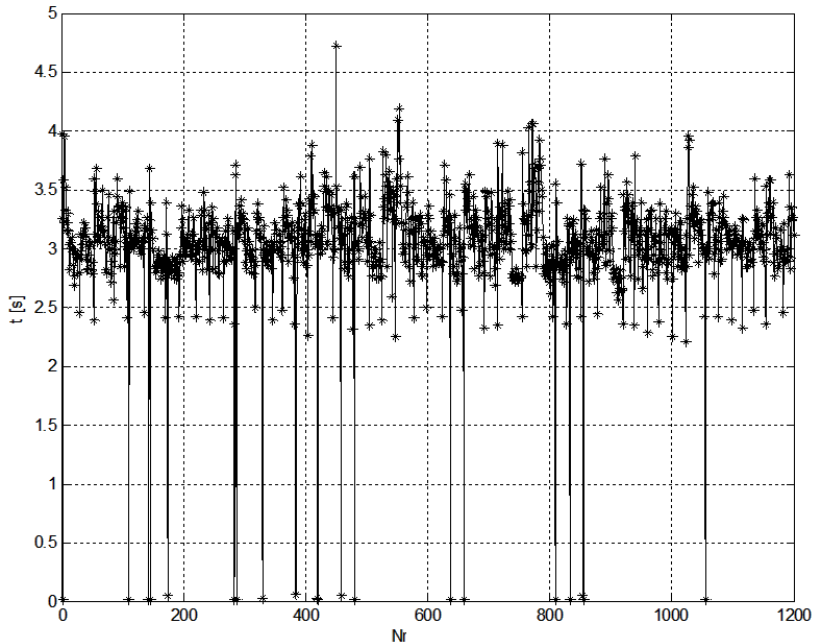
Fig. 3-36 Algorithm operation time for consecutive images calculated on a computer with a 64-bit operating system, Intel Core Quad CPU 2.5 GHz processor, 2GB RAM

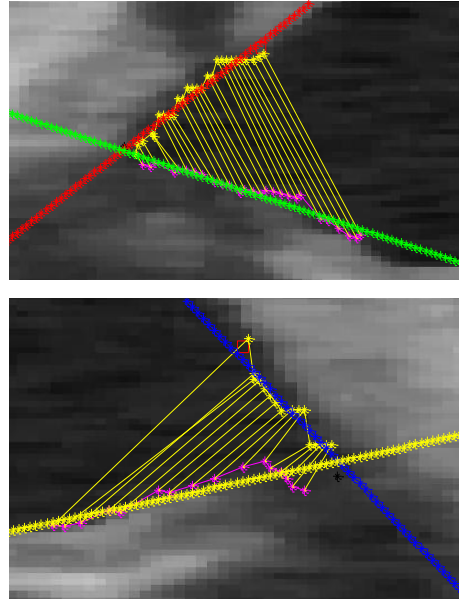AOD, TISA and TIA methods have drawbacks consisting in difficulties to cope with a large degree of pathology. Such situations occur in the case of partial narrowing of the filtration angle. Therefore the analysis of distances between appropriate points has been suggested in accordance with Fig. 3-38 using the previous calculations.

```
dist_l=[];
for ib=1:size(xy_g_l,1)
    r_x=xy_g_l(ib,1) - xy_d_l(ib,1);
    r_y=xy_g_l(ib,2) - xy_d_l(ib,2);
    dist_l(ib,1:2)=[ib sqrt( (r_x).^2  +  (r_y).^2  )];
end
dist_p=[];
for ib=1:size(xy_g_p,1)
    r_x=xy_g_p(ib,1) - xy_d_p(ib,1);
    r_y=xy_g_p(ib,2) - xy_d_p(ib,2);
    dist_p(ib,1:2)=[ib sqrt( (r_x).^2  +  (r_y).^2  )];
end
figure; plot(dist_l(:,1), dist_l(:,2),'-r*'); hold on
plot(dist_p(:,1), dist_p(:,2),'-g*'); grid on
```

The graph obtained using the above fragment of the algorithm is shown below.

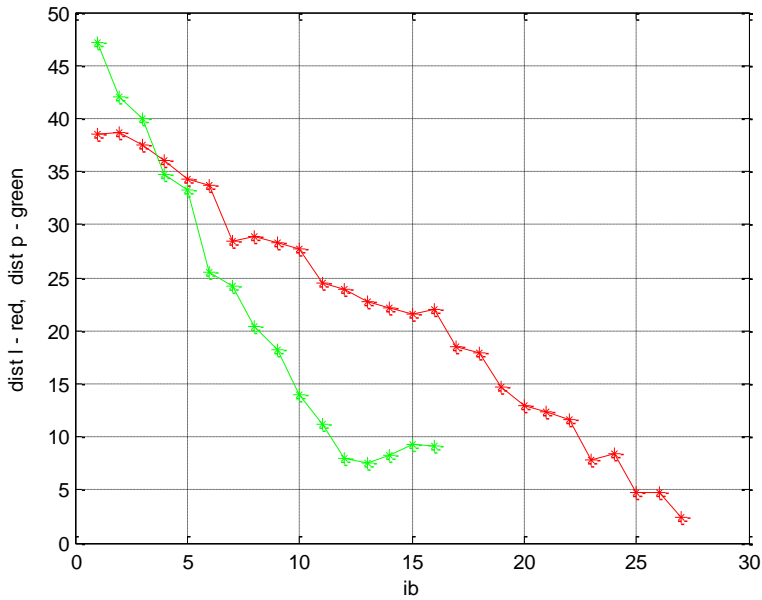**Fig. 3-37 Result of distance measurement at the filtration angle measurement**

The following images show examples of results obtained for other patients.
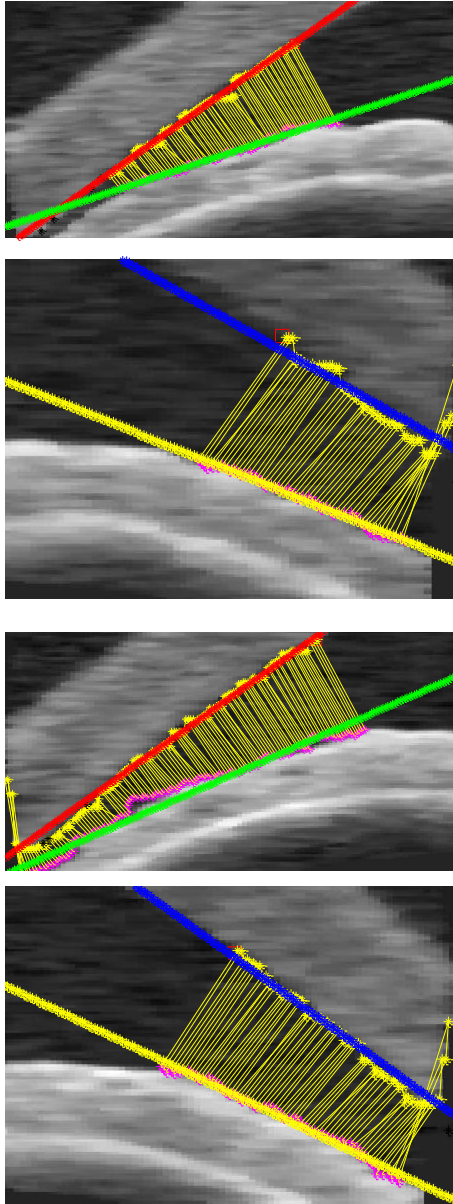
Fig. 3-38 Result of operation of algorithm for automated filtration angle measurement
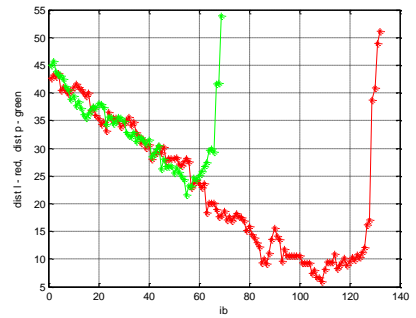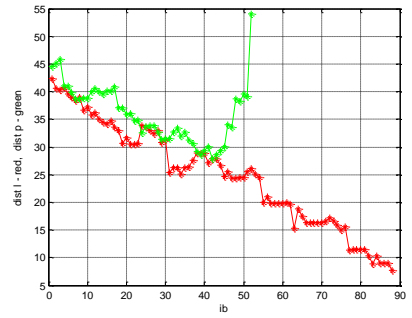

Fig. 3-39 Result of operation of algorithm for automated filtration angle measurement

The results obtained for the authors' method show it copes much better with large degrees of pathology, which is confirmed by the graph of distance changes shown in Fig. 3-39.

3.6.1   Advantages of the Algorithm Proposed

An automated analysis of anterior eye segment allows obtaining reliable results during a period of time not longer than 3.5 s. The assessment of algorithm sensitivity to parameter changes, and in particular - to

- The area of iridocorneal angle searching shows the largest dependence on the width of the iris searching area for pathological cases,
- For 70,736 images correct results were obtained for around 55,000 cases. An approximate result indicating the number of properly measured cases results from the difficulties in the assessing and suggesting how the algorithm should properly respond,
- The greatest measurement error, excluding the impact of method errors and its sensitivity, occurred for AOD and TIA methods,
- On the basis of experience gained in the measurement of the filtration angle an own authors' measurement method has been suggested.

Summarising this section it is necessary to emphasise the fact that the presented Matlab source code does not exhaust the issue. It is short of both protections, e.g. related to the detection of proper position of filtration angles and also of preliminary analysis of the analysed object position on the scene (Results for `path_name=` `'d:/OCT/SOURCES/1.DCM'` – Fig. 3-40 and Fig. 3-41).
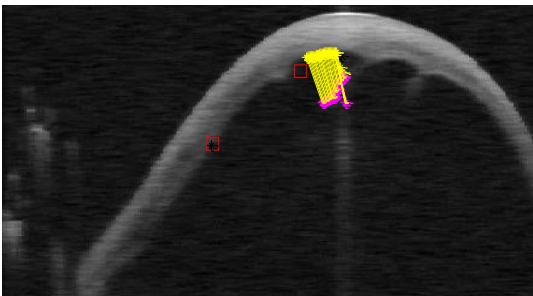

Fig. 3-40 Result of erroneous operation of algorithm automatically determining the filtration angle (yellow)
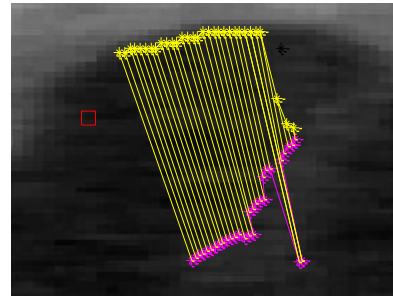
Fig. 3-41 Enlarged fragment from Fig. 3-40.

In this place we encourage the Readers to create on their own such simple safeguards in the algorithm. Readers can also optimise the graph (Fig. 3-39.) to useful values. These are such parameters, which will allow rough approximation of graph Fig. 3-39. These situations apply to cases presented in (Fig. 3-42).
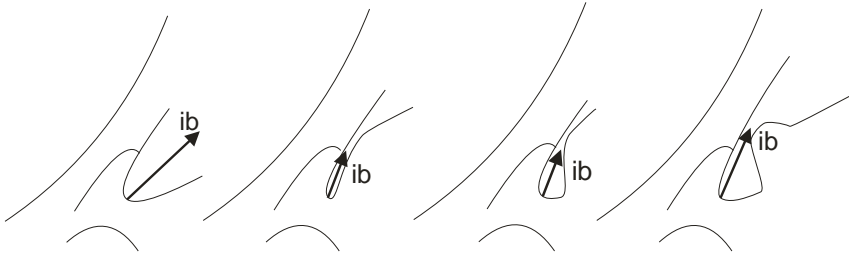
Fig. 3-42 Demonstrative figure showing filtration angles and problems with describing the results obtained using the algorithm presented

A description of pathological cases of filtration angle prevails here, such, for which there are cases of local narrowing or local closure of the angle (Fig. 3-42). The notation, which a Reader can suggest, must consist of a few digits (symbols) automatically determined from the graph, Fig. 3-39. For example, the alphabet created may look as follows:

- symbols:

    / - increasing distance for consecutive id-s,
    ^ - local minimum,
    v – local maximum,
    _ - invariable value of distance for a changing id.

- numerical parameters:

    - angular value,
    - maximum, minimum or constant distance for defined id-s,
    - id range, in which a specific situation does not occur.

For example, the notation _80,100 /30 consists of two symbols "_" and "/", where according to the interpretation adopted the former stands for a narrowing, a slit, in the filtration angle of dist = 80 value in the range id = 100 um and the latter – a typical angle of 30° (corresponding to the state from the second image in Fig. 3-42).

3.7 Determination of Anterior Chamber Volume Based on a Series of Images

The analysis of anterior chamber (Fig. 3-43) is based on contours of boundaries determined in the previous section and presented in Fig. 3-27. They were determined on images performed at preset angles acc. to (Fig. 3-44).

Fig. 3-43 Anterior chamber position in eye cross-section
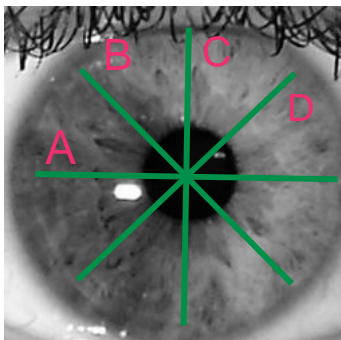
Fig. 3-44 Arrangement of individual eye scans.
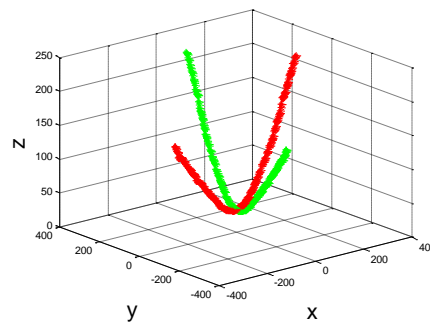
Fig. 3-45 Contours of external boundary of sclera on scans A and B (Fig. 3-43)) in a Cartesian coordinate system

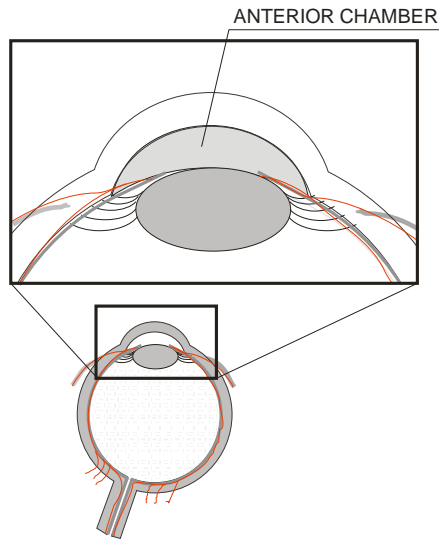To simplify the notation, let us further assume that the function determining necessary contours and the filtration angle will be defined as:

```
[Ls,L22,xy_g_l, xy_d_l, xy_g_p,
xy_d_p,linie_12]=OCT_angle_line(Ls);
```

where the OCT image is an input parameter and at the output we obtain, in accordance with the previous section:

- `xy_d_l` coordinates x and y of the filtration angle, left hand lower contour,
- `xy_g_l` coordinates x and y of the filtration angle, left hand upper contour,
- `xy_d_p` coordinates x and y of the filtration angle, right hand lower contour,
- `xy_g_p` coordinates x and y of the filtration angle, right hand upper contour,
- `linie_12` contour lines.

To obtain the result presented in Fig. 3-34 using the function defined this way, it is necessary to write:

```
path_name='d:/OCT/SOURCES/3.DCM';
fid = fopen(path_name, 'r');
dataa = fread(fid,'uint8');
fclose(fid);
[header_dicom,Ls]=OCT_head_read(dataa);
[Ls,L22,xy_g_l, xy_d_l, xy_g_p,
xy_d_p,linie_12]=OCT_angle_line(Ls);
figure; imshow(Ls,[]); hold on
PPgl=polyfit(xy_g_l(:,1),xy_g_l(:,2),1);
PPdl=polyfit(xy_d_l(:,1),xy_d_l(:,2),1);
PPgp=polyfit(xy_g_p(:,1),xy_g_p(:,2),1);
PPdp=polyfit(xy_d_p(:,1),xy_d_p(:,2),1);
x=1:size(Ls,2);
y=polyval(PPgl,x); plot(x,y,'r*')
y=polyval(PPdl,x); plot(x,y,'g*')
y=polyval(PPgp,x); plot(x,y,'b*')
y=polyval(PPdp,x); plot(x,y,'y*')
al=atan(PPdl(1))*180/pi-atan(PPgl(1))*180/pi;
ap=atan(PPgp(1))*180/pi-atan(PPdp(1))*180/pi;
al=round(al*10)/10;
ap=round(ap*10)/10;
title(['Left - ',mat2str(al),'^o', '    Right -
',mat2str(ap),'^o'])
```

The basic difficulty in an attempt to calculate the volume of anterior eye chamber is a correct determination of sclera boundaries and selection of appropriate method for approximation of intermediate spaces (Fig. 3-44), existing between scans A-B, B-C, C-D, D-A.

A method, preliminary consisting in creating the contour of internal and external sclera boundary, adjusted by the filtration angle boundaries, has been presented below, i.e.:

```
linie_m_x=[flipud(xy_d_l(:,1))',xy_d_l(1,1):xy_d_p(1,1) ,
xy_d_p(:,1)'];
linie_m_y=[flipud(xy_d_l(:,2))',linspace(xy_d_l(1,2),xy_d_p
(1,2),length(xy_d_l(1,1):xy_d_p(1,1) )) , xy_d_p(:,2)'];
plot(linie_m_x,linie_m_y,'-c*')
```

The obtained boundary is shown in Fig. 3-46 and Fig. 3-47.
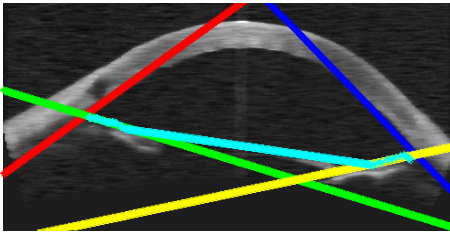
Fig. 3-46 Determined boundary after correction with the values of filtration angle boundary
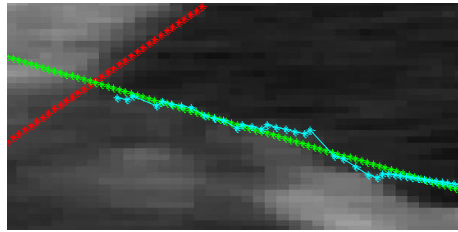
Fig. 3-47 Enlarged fragment from Fig. 3-46

As visible in the image presented (Fig. 3-46 i Fig. 3-47) the contour marked with a turquoise line is not drawn in a perfect way. The correction consists in using the method of modified active contour (Fig. 3-48).
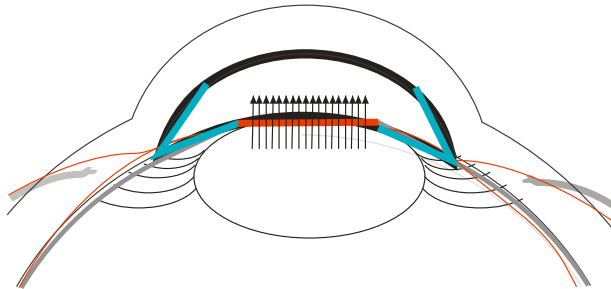
Fig. 3-48 Demonstrative figure showing the idea of straight line (red) dragging to the lens contour

The method of modified active contour consists in maximisation of external – internal energy F_{ZW}. This energy is calculated as the difference between average values of pixels brightness inside and outside the declared area. In a general case the calculations start from the determination of characteristic points w_i (w_1, w_2,..., w_{k-1}, w_k, w_{k+1},...,w_K).

For each determined point w_k a straight line is drawn, perpendicular to adjacent points and passing the point considered. For example, for point w_k a straight line is drawn passing through it and perpendicular to the straight line connecting points w_{k-1}, w_{k+1}. In the next stage the outside and inside areas are defined and weights for individual pixels are determined. In the simplest case this is the average value of brightness at weights of individual pixels calculated as 1. Any shape of inside and outside area may be chosen, however, a rectangular area is most often used - Fig. 3-49.
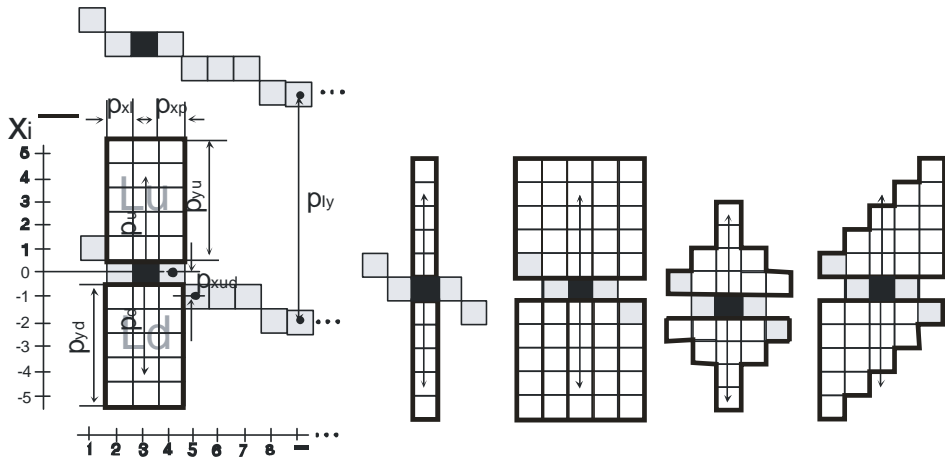
Fig. 3-49 Demonstrative diagram of pixels arrangement in the analysis of operation of the modified active contour method and examples of analysis area

If we assume, in accordance with the nomenclature from Fig. 3-49 Lu as an outside area, Ld as an inside area and their dimensions in the sense of the number of rows and columns in a rectangular case as p_{yd} x ($p_{xl}+p_{xp}+1$) and p_{yu} x ($p_{xl}+p_{xp}+1$) then the matrix of differences may be written as follows:

$$\Delta_S = \begin{array}{r} \cdots \\ -4 \\ -3 \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ \cdots \end{array} \left\{ \begin{array}{cccc} \cdots & & & \\ \Delta_{S,-4,1} & \Delta_{S,-4,2} & \Delta_{S,-4,3} & \Delta_{S,-4,4} \\ \Delta_{S,-3,1} & \Delta_{S,-3,2} & \Delta_{S,-3,3} & \Delta_{S,-3,4} \\ \Delta_{S,-2,1} & \Delta_{S,-2,2} & \Delta_{S,-2,3} & \Delta_{S,-2,4} \\ \Delta_{S,-1,1} & \Delta_{S,-1,2} & \Delta_{S,-1,3} & \Delta_{S,-1,4} \\ \Delta_{S,0,1} & \Delta_{S,0,2} & \Delta_{S,0,3} & \Delta_{S,0,4} & \cdots \\ \Delta_{S,1,1} & \Delta_{S,1,2} & \Delta_{S,1,3} & \Delta_{S,1,4} \\ \Delta_{S,2,1} & \Delta_{S,2,2} & \Delta_{S,2,3} & \Delta_{S,2,4} \\ \Delta_{S,3,1} & \Delta_{S,3,2} & \Delta_{S,3,3} & \Delta_{S,3,4} \\ \Delta_{S,4,1} & \Delta_{S,4,2} & \Delta_{S,4,3} & \Delta_{S,4,4} \\ & \cdots & & \end{array} \right.$$

where Δ_S is the difference in average values of Lu and Ld areas, i.e.:

$$\Delta_S = \frac{\sum\limits_{x,y} Ld}{p_{yd} \cdot (p_{xl} + p_{xp})} - \frac{\sum\limits_{x,y} Lu}{p_{yu} \cdot (p_{xl} + p_{xp})} \tag{6}$$

p_{yu} - number of rows Lu,

p_{yd} - number of rows Ld,

p_u - range of movement and areas of the pixel Lu i Ld top,

p_d - range of movement and areas of the pixel Lu i Ld down,

p_{xl} - number of columns on the left part of the analyzed pixel,

p_{xp} - number of columns on the left part of the analyzed piel,

p_{ly} - distance in the axis oy with y_{RPEC},

p_{xud} - distance between neighboring pixels in the axis oy.

In the next stage elements are sorted separately for each column of matrix Δ_S. As a result we obtain, for example:

$$sort(\Delta_S) = \begin{matrix}
0.8_{1,1} & 0.4_{-2,2} & 0.9_{2,3} & 0.8_{-2,4} & 0.4_{-2,5} & 0.1_{2,6} \\
0.7_{2,1} & 0.4_{-1,2} & 0.9_{-1,3} & 0.5_{-1,4} & 0.4_{-3,5} & 0.1_{1,6} \\
0.7_{-3,1} & 0.4_{-3,2} & 0.6_{0,3} & 0.5_{0,4} & 0.3_{-1,5} & 0.1_{-4,6} \\
0.4_{-4,1} & 0_{1,2} & 0.3_{-2,3} & 0.2_{1,4} & 0.1_{2,5} & 0_{-3,6} \\
0.4_{-2,1} & -0.1_{0,2} & 0.3_{-4,3} & 0.2_{2,4} & 0_{4,5} & -0.1_{-2,6} \\
-0.1_{3,1} & -0.3_{2,2} & -0.1_{3,3} & 0.1_{3,4} & -0.1_{3,5} & -0.1_{0,6} \\
-0.1_{4,1} & -0.4_{-4,2} & -0.2_{1,3} & 0.1_{-3,4} & -0.1_{-2,5} & -0.4_{3,6} \\
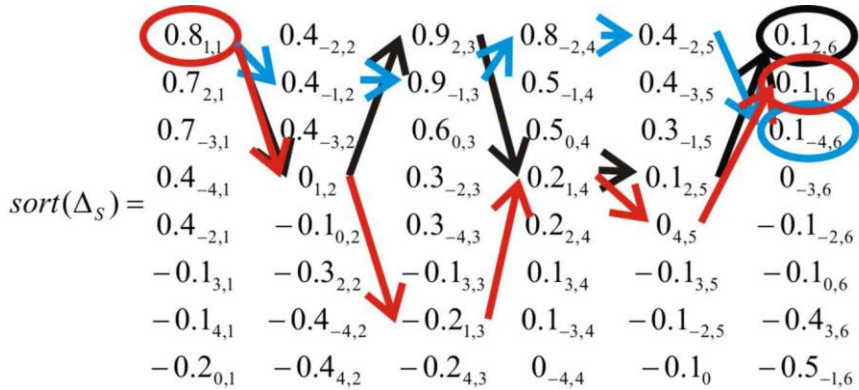-0.2_{0,1} & -0.4_{4,2} & -0.2_{4,3} & 0_{-4,4} & -0.1_{0} & -0.5_{-1,6}
\end{matrix}$$

Fig. 3-50 Demonstrative figure of the method for selecting the optimum path

sort(Δ_S) is the basis to determine the target new position of pixels. The analysis of searching for the best solution is close to a problem of path seeking at the criterion of maximising the difference in the average values and minimising the difference in adjacent pixels positions. As against the latter ones, a coefficient p_{xud} has been suggested, defined as a permissible difference in the position on the oy axis of pixels neighbouring in consecutive positions on the ox axis. Fig. 3-50 shows the selection of optimum path for $p_{xud}=0$ − red colour, $p_{xud}=1$ − black colour and $p_{xud}=2$ − blue colour. Let us assume that we consider the case for $p_{xud}=2$.

Starting from the point of coordinates (1,1) we obtain positions of the next pixels (-1,2), because $|-1-1|<= p_{xud}$, then (-1,3), (-2,4), (-2,5) and (-4,6). While selecting the next points we should consider two elements: permissible change of the location on the ox axis defined by parameter p_{xud} and the position of the largest values (the higher is a full element in the column, the better).

Reducing the value of p_{xud} we obtain smaller differences on the oy axis between consecutive pixels, at a cost of increased error of contour fit. Instead, increasing the value of p_{xud} we allow a possibility of greater fluctuation of neighbouring pixels on the oy axis, obtaining this way more precise representation of the contour. Looking at the matrix sort(Δ_S) it is possible to notice a trend of finding the highest situated path for consecutive columns; this feature has been used in a practical implementation, i.e. in the function OCT_activ_cont

```
function
[yy,i]=OCT_activ_cont(L1,x,y,pud,pyud,pxud,pxlp,polaryzacja
)
x=x(:);
y=y(:);
pam_grd=[];
pam_num=[];
if polaryzacja==1
for i=1:size(x,1)
    gr_gd=[];
    for j=-pud:pud
        wgp=(y(i)-pyud+j);
        wgk=(y(i)+j);
        kgp=(x(i)-pxlp);
        kgk=(x(i)+pxlp);
        wdp=(y(i)+j);
        wdk=(y(i)+pyud+j);
        kdp=(x(i)-pxlp);
        kdk=(x(i)+pxlp);
        if wgp<=0; wgp=1; end
        if wdp<=0; wdp=1; end
        if wgk>size(L1,1); wgk=size(L1,1); end
        if wdk>size(L1,1); wdk=size(L1,1); end
        if kgp<=0; kgp=1; end
        if kdp<=0; kdp=1; end
        if kgk>size(L1,2); kgk=size(L1,2); end
        if kdk>size(L1,2); kdk=size(L1,2); end
       Lu=L1(wgp:wgk,kgp:kgk);
       Ld=L1(wdp:wdk,kdp:kdk);
       gr_gd=[gr_gd;mean(Lu(:))-mean(Ld(:))];
    end
    pam_grd=[pam_grd,gr_gd];
    gr_num=[gr_gd,( (y(i)-pud) : (y(i)+pud) )'];
    gr_num=sortrows(gr_num,1);
    pam_num=[pam_num,gr_num(:,2)];
end


elseif polaryzacja==-1

for i=1:size(x,1)
    gr_gd=[];
    for j=-pud:pud
        wgp=(y(i)-pyud+j);
        wgk=(y(i)+j);
        kgp=(x(i)-pxlp);
        kgk=(x(i)+pxlp);
        wdp=(y(i)+j);
        wdk=(y(i)+pyud+j);
        kdp=(x(i)-pxlp);
```

```
        kdk=(x(i)+pxlp);
        if wgp<=0; wgp=1; end
        if wdp<=0; wdp=1; end
        if wgk>size(L1,1); wgk=size(L1,1); end
        if wdk>size(L1,1); wdk=size(L1,1); end
        if kgp<=0; kgp=1; end
        if kdp<=0; kdp=1; end
        if kgk>size(L1,2); kgk=size(L1,2); end
        if kdk>size(L1,2); kdk=size(L1,2); end
       Lu=L1(wgp:wgk,kgp:kgk);
       Ld=L1(wdp:wdk,kdp:kdk);
       gr_gd=[gr_gd;mean(Ld(:))-mean(Lu(:))];
     end
    pam_grd=[pam_grd,gr_gd];
    gr_num=[gr_gd,( (y(i)-pud) : (y(i)+pud) )'];
    gr_num=sortrows(gr_num,1);
    pam_num=[pam_num,gr_num(:,2)];
end

else
    disp('polaryzation ?')
end
i_hh=[];
for hh=1:7
i=ones([1 size(pam_num,2)]);
i(1)=hh;
j=1;
while (j+1)<size(pam_num,2)
    if abs(pam_num(i(j),j)-pam_num(i(j+1),j+1))<pxud
            j=j+1;
    else
            if i(j+1)<size(pam_num,1)
                i(j+1)=i(j+1)+1;
            else
                i(j+1)=i(j);
                j=j+1;
            end
    end
end
i_hh=[i_hh;i];
end
[d_,smiec]=find(sum(i_hh,2)==min(sum(i_hh,2)));
i=i_hh(d_(1),:);

yy=y;
for i__=1:length(i)
    yy(i__)=pam_num( i(i__),i__);
end
```

The input arguments for the functions are:

`L1` – input image,
`x` – position of input points on the ox axis,
`y` – position of input points on the oy axis,

polarisation – parameter responsible for a feature of searched contour, "1" stands for a white object against a dark background, while value of "-1" – the opposite situation.

As a result, new coordinates on the oy axis are obtained. The presented implementation of modified active contour function has many limitations and assumptions made, related for instance to making an assumption that the contour searched for is situated horizontally.

However, the function presented has very interesting properties depending on the parameters adopted. These properties will be the subject of further considerations in one of the next sections. Using the function as follows:

```
pud=10;
pyud=10;
pxud=2;
pxlp=10;
polaryzacja=1;
[yy,i]=OCT_activ_cont(mat2gray(Ls),linie_m_x,linie_m_y,pud,
pyud,pxud, pxlp,  polaryzacja);
plot(linie_m_x,yy,'-w*')
linie_12(:,2)=medfilt2(linie_12(:,2),[15 1]);
linie_12(:,3)=medfilt2(linie_12(:,3),[15 1]);
linie_mm_x=[flipud(xy_g_l(:,1)); linie_12(
(linie_12(:,1)>xy_g_l(1,1)) & (linie_12(:,1)<xy_g_p(1,1))
,1) ; xy_g_p(:,1)];
linie_mm_y=[flipud(xy_g_l(:,2)); linie_12(
(linie_12(:,1)>xy_g_l(1,1)) & (linie_12(:,1)<xy_g_p(1,1))
,3) ; xy_g_p(:,2)];
plot(linie_mm_x,linie_mm_y,'-w*')
```

We obtain the results presented in Fig. 3-51. The determined boundaries, marked white, have been obtained using the function `OCT_activ_cont`
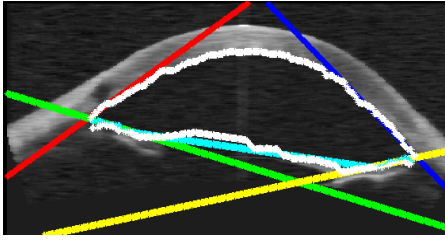
Fig. 3-51 Determined boundaries marked white, using the function `OCT_activ_cont`
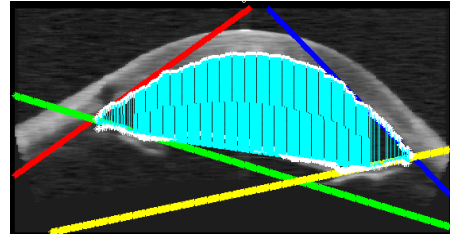
Fig. 3-52 Determined boundaries marked white, using the function `OCT_activ_cont`

Fig. 3-52 shows determined boundaries, marked white, obtained using the function `OCT_activ_cont` and connected corresponding points are marked with turquoise lines.

To prepare the final fragment of the algorithm for anterior chamber volume calculation it is necessary to connect the lines and to allocate correspondence to individual points, which is a simple procedure, i.e.:

```
if length(linie_m_x)>=length(linie_mm_x)
    for ht=1:length(linie_mm_x)
        linie_r=sortrows ([ abs(linie_m_x'-
linie_mm_x(ht)),linie_m_x',linie_m_y' ]);
        line([linie_r(1,2) linie_mm_x(ht)],[linie_r(1,3)
linie_mm_y(ht)]);
    end
else
    for ht=1:length(linie_m_x)
        linie_r=sortrows ([ abs(linie_mm_x-
linie_m_x(ht)),linie_mm_x,linie_mm_y ]);
        line([linie_r(1,2) linie_m_x(ht)],[linie_r(1,3)
linie_m_y(ht)]);
    end
end
```

The above stage of inside boundaries determination on a single OCT image is a compact whole, which has been located in the function `OCT_edge_inside` returning values of boundaries contour line coordinates, i.e.:

```
[linie_m_x1,linie_m_y1,linie_121,linie_mm_x1,linie_mm_y1]=O
CT_edge_inside(Ls);
```

The last stage of the algorithm presented consists of calculation of anterior chamber volume on the basis of reconstruction presented. There are many practical methods used in such calculations.

The first group of methods is based on the definition for calculation of solid of revolution volume formed as a result of function $f(x)$ revolution around axis ox and using the formula for volume V:

$$V = \pi \int_{x2}^{x1} (f(x))^2 dx \qquad (7)$$

In this case there is a difficulty in defining the analytical shape of function $f(x)$. Instead, accuracies obtained using this method are very high.

The second method consists in the calculation of average value V, calculated from revolutions of contour solid for each image. This method features lower accuracy, however, the results are obtained pretty quickly.

The third group of methods is based on the calculation of a sum of binary images pixels of image sequence originated on the xyo axis. This method is accurate and fast, but only in the case of discrete structures – 3D matrices existence. Unfortunately in this case the necessary conversion to 3D matrices causes an unnecessary increase in the algorithm computational complexity.

The fourth method consists of two stages:

- digitisation of anterior chamber 3D contour calculations,
- summing up spherical sectors formed from consecutive points of the wall, i.e.:

$$S = \left(\frac{\Delta\alpha}{360}\pi - \frac{\sin\Delta\alpha}{2}\right)r^2 \qquad (8)$$

which after simple transformations for constant values $\Delta\alpha=3.6°$ gives:

$$S = 0.2527r^2 \qquad (9)$$

The fifth method (practically implemented) consists in counting the areas of unit triangles formed by vertices (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3), (x_4,y_4,z_4). By definition $x_1=x_2=x_3=x_4$ have been ensured for consecutive iterations, for which there is a unit increment in the value on the ox axis.

The Pole (Area) variable contains the result of summing up areas of calculated triangles located on the oyz axis for x values featuring unit increments. The basic relationship for a triangle area has been used here, i.e.

$$S = \frac{1}{2}\det\begin{vmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{vmatrix} \tag{10}$$

A demonstrative figure is shown below, presenting the methodology adopted in the algorithm and designed to calculate the anterior chamber value based on the partial areas sum.
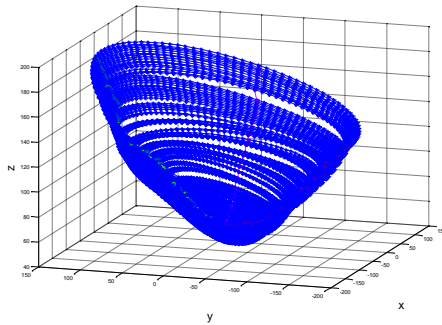

Fig. 3-53 Contour lines, based on which vertices of triangles analysed have been formed
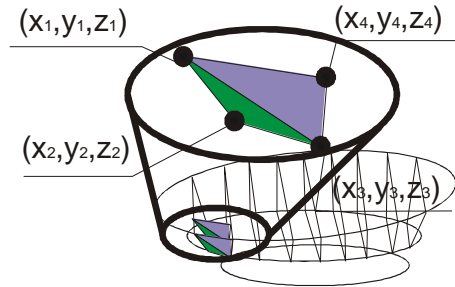
Fig. 3-54 Arrangement of triangle vertices (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3), (x_4,y_4,z_4)

To perform a practical implementation of one of the methods for anterior chamber volume calculation first it is necessary to use the function OCT_edge_inside returning the values of contour boundary lines to two OCT images made at an angle of 90°:

```
path_name='d:/OCT/SOURCES/3.DCM';
fid = fopen(path_name, 'r');
dataa = fread(fid,'uint8');
fclose(fid);
[header_dicom,Ls1]=OCT_head_read(dataa);
[linie_m_x1,linie_m_y1,linie_121,linie_mm_x1,linie_mm_y1
]=OCT_edge_inside(Ls1);
path_name='d:/OCT/SOURCES/3.DCM';
fid = fopen(path_name, 'r');
dataa = fread(fid,'uint8');
fclose(fid);
[header_dicom,Ls2]=OCT_head_read(dataa);
[linie_m_x2,linie_m_y2,linie_122,linie_mm_x2,linie_mm_y2
]=OCT_edge_inside(Ls2);
```

In the results obtained it is necessary to modify the sequence of points coordinates occurrence, i.e.:

```
xa1=[linie_mm_x1(end:-1:1)];
```

```
xa2=[linie_mm_x2(end:-1:1)];
za1=[linie_mm_y1(end:-1:1)];
za2=[ linie_mm_y2(end:-1:1)];
```

Then, if we assume, that the apparatus axis is in the middle of coordinates correction image, i.e.:

```
mm1=median(xa1);
mm2=median(xa2);
xa1=xa1-median(xa1);
xa2=xa2-median(xa2);
ya1=zeros(size(za1));
ya2=zeros(size(za2));
```

In the next stage, in the case of both images situated against each other at an angle of 90°, an appropriate correction, i.e.:

```
 [THETAa,RHOa,Za] = cart2sph(xa1,ya1,za1);
THETAa=THETAa+90*pi/180;
[xa1,ya1,za1] = sph2cart(THETAa,RHOa,Za);
```

The division, necessary to carry out further steps of the algorithm, into the left and right part looks as follows:

```
xa1_a=xa1(ya1<=0);
xa1_b=xa1(ya1>0);
ya1_a=ya1(ya1<=0);
ya1_b=ya1(ya1>0);
za1_a=za1(ya1<=0);
za1_b=za1(ya1>0);
xa2_a=xa2(xa2<=0);
xa2_b=xa2(xa2>0);
ya2_a=ya2(xa2<=0);
ya2_b=ya2(xa2>0);
za2_a=za2(xa2<=0);
za2_b=za2(xa2>0);
figure
plot3(xa1_a,ya1_a,za1_a,'-r*'); grid on; hold on
plot3(xa1_b,ya1_b,za1_b,'-g*');
plot3(xa2_a,ya2_a,za2_a,'-b*');
plot3(xa2_b,ya2_b,za2_b,'-m*');
xlabel('x','FontSize',20); ylabel('y','FontSize',20);
zlabel('z','FontSize',20)
```

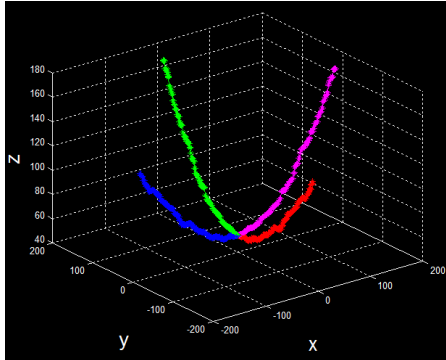The result obtained is presented in Fig. 3-55.

Fig. 3-55 Determined boundaries
`xa1_a,ya1_a,za1_a,`
`xa1_b,ya1_b,za1_b`
`xa2_a,ya2_a,za2_a,`
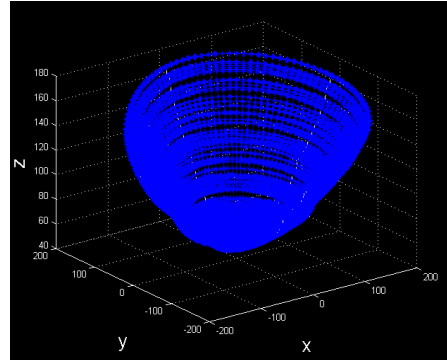`xa2_b,ya2_b,za2_b,` **marked in colours**

Fig. 3-56 Figure after reconstruction

For further calculations it turns out necessary to unify the number of elements existing for each of 4 edges visible in Fig. 3-55 as follows:

```
s_m=max([length(xa1_a), length(xa1_b), length(xa2_a),
length(xa2_b)]);
xa1_aa=[]; xa1_bb=[];
ya1_aa=[]; ya1_bb=[];
za1_aa=[]; za1_bb=[];
xa2_aa=[]; xa2_bb=[];
ya2_aa=[]; ya2_bb=[];
za2_aa=[]; za2_bb=[];
for it=1:s_m
    xa1_aa(it)=xa1_a( round( (length(xa1_a)/s_m) *it));
    xa1_bb(it)=xa1_b( round( (length(xa1_b)/s_m) *it));
    ya1_aa(it)=ya1_a( round( (length(ya1_a)/s_m) *it));
    ya1_bb(it)=ya1_b( round( (length(ya1_b)/s_m) *it));
    za1_aa(it)=za1_a( round( (length(za1_a)/s_m) *it));
    za1_bb(it)=za1_b( round( (length(za1_b)/s_m) *it));

    xa2_aa(it)=xa2_a( round( (length(xa2_a)/s_m) *it));
    xa2_bb(it)=xa2_b( round( (length(xa2_b)/s_m) *it));
    ya2_aa(it)=ya2_a( round( (length(ya2_a)/s_m) *it));
    ya2_bb(it)=ya2_b( round( (length(ya2_b)/s_m) *it));
    za2_aa(it)=za2_a( round( (length(za2_a)/s_m) *it));
    za2_bb(it)=za2_b( round( (length(za2_b)/s_m) *it));
end
plot3(xa1_aa,ya1_aa,za1_aa,'-w*'); grid on; hold on
plot3(xa1_bb,ya1_bb,za1_bb,'-w*');
plot3(xa2_aa,ya2_aa,za2_aa,'-w*');
```

```
plot3(xa2_bb,ya2_bb,za2_bb,'-w*');
```

The spline function is the basis for missing points reconstruction. Function spline returns the piecewise polynomial form of the cubic spline interpolan. For values `xa1_aa,ya1_aa,za1_aa` etc. unified in such a way, the following notation using the spline function has been introduced:

```
xc=[]; yc=[]; zc=[];
pam_p=[];
for i=1:s_m
        xi = pi*[0:.5:2];
        xyzi = [xa1_aa(end-i+1),  xa2_aa(end-
i+1),xa1_bb(i), xa2_bb(i) xa1_a(end-i+1);
                ya1_aa(end-i+1),  ya2_aa(end-
i+1),ya1_bb(i), ya2_bb(i) ya1_aa(end-i+1);
                za1_aa(end-i+1),  za2_aa(end-
i+1),za1_bb(i),  za2_bb(i) za1_aa(end-i+1)];
        pp = spline(xi,xyzi);
        xyz_ = ppval(pp, linspace(0,2*pi,101));
        plot3(xyz_(1,:),xyz_(2,:),xyz_(3,:),'-*b')
        xc=[xc,xyz_(1,:)'];
        yc=[yc,xyz_(2,:)'];
        zc=[zc,xyz_(3,:)'];
end
```

The result obtained is presented in Fig. 3-56.

After calculations using the spline function the next stage comprises calculation of anterior chamber volume, i.e.:

```
xc=round(xc); yc=round(yc); zc=round(zc);
Objetosc=0;
min_x=min(min(xc))-1;
for iuu=1:(size(xc,2)-1)
for iu=1:49
        xq=[]; yq=[]; zq=[];

        xq1=linspace( xc(iu,iuu),xc(end-
iu,iuu),abs(xc(iu,iuu)-xc(end-iu,iuu)) );
        xq(1,(xc(iu,iuu)-min_x):(xc(iu,iuu)-
min_x+length(xq1)-1))=xq1;

        yq1=linspace( yc(iu,iuu),yc(end-
iu,iuu),abs(xc(iu,iuu)-xc(end-iu,iuu)) );
        yq(1,(xc(iu,iuu)-min_x):(xc(iu,iuu)-
min_x+length(yq1)-1))=yq1;
        zq1=linspace( zc(iu,iuu),zc(end-
iu,iuu),abs(xc(iu,iuu)-xc(end-iu,iuu)) );
        zq(1,(xc(iu,iuu)-min_x):(xc(iu,iuu)-
min_x+length(zq1)-1))=zq1;
```

```
        xq2=linspace( xc(iu,iuu+1),xc(end-
iu,iuu+1),abs(xc(iu,iuu+1)-xc(end-iu,iuu+1)) );
        xq(2,(xc(iu,iuu+1)-min_x):(xc(iu,iuu+1)-
min_x+length(xq2)-1))=xq2;
        yq2=linspace( yc(iu,iuu+1),yc(end-
iu,iuu+1),abs(xc(iu,iuu+1)-xc(end-iu,iuu+1)) );
        yq(2,(xc(iu,iuu+1)-min_x):(xc(iu,iuu+1)-
min_x+length(yq2)-1))=yq2;
        zq2=linspace( zc(iu,iuu+1),zc(end-
iu,iuu+1),abs(xc(iu,iuu+1)-xc(end-iu,iuu+1)) );
        zq(2,(xc(iu,iuu+1)-min_x):(xc(iu,iuu+1)-
min_x+length(zq2)-1))=zq2;
        xq3=linspace( xc(iu+1,iuu),xc(end-
iu+1,iuu),abs(xc(iu+1,iuu)-xc(end-iu+1,iuu)) );
        xq(3,(xc(iu+1,iuu)-min_x):(xc(iu+1,iuu)-
min_x+length(xq3)-1))=xq3;
        yq3=linspace( yc(iu+1,iuu),yc(end-
iu+1,iuu),abs(xc(iu+1,iuu)-xc(end-iu+1,iuu)) );
        yq(3,(xc(iu+1,iuu)-min_x):(xc(iu+1,iuu)-
min_x+length(yq3)-1))=yq3;
        zq3=linspace( zc(iu+1,iuu),zc(end-
iu+1,iuu),abs(xc(iu+1,iuu)-xc(end-iu+1,iuu)) );
        zq(3,(xc(iu+1,iuu)-min_x):(xc(iu+1,iuu)-
min_x+length(zq3)-1))=zq3;
        xq4=linspace( xc(iu+1,iuu+1),xc(end-
iu+1,iuu+1),abs(xc(iu+1,iuu+1)-xc(end-iu+1,iuu+1)) );
        xq(4,(xc(iu+1,iuu+1)-min_x):(xc(iu+1,iuu+1)-
min_x+length(xq4)-1))=xq4;
        yq4=linspace( yc(iu+1,iuu+1),yc(end-
iu+1,iuu+1),abs(xc(iu+1,iuu+1)-xc(end-iu+1,iuu+1)) );
        yq(4,(xc(iu+1,iuu+1)-min_x):(xc(iu+1,iuu+1)-
min_x+length(yq4)-1))=yq4;
        zq4=linspace( zc(iu+1,iuu+1),zc(end-
iu+1,iuu+1),abs(xc(iu+1,iuu+1)-xc(end-iu+1,iuu+1)) );
        zq(4,(xc(iu+1,iuu+1)-min_x):(xc(iu+1,iuu+1)-
min_x+length(zq4)-1))=zq4;
         plot3(xq1,yq1,zq1,'r*');
        for tu=1:size(xq,2)
            if sum(xq(:,tu)~=0)==4
                    Objetosc=Objetosc+0.5*...
                        abs(det([yq(1,tu) zq(1,tu) 1;
yq(2,tu) zq(2,tu) 1; yq(3,tu) zq(3,tu) 1]))...
                        +0.5*...
                        abs(det([yq(2,tu) zq(2,tu) 1;
yq(3,tu) zq(3,tu) 1; yq(4,tu) zq(4,tu) 1]));
            end
        end
end
end
Objetosc
```

We obtain the result for the anterior chamber expressed in pixels, shown in (Fig. 3-57):
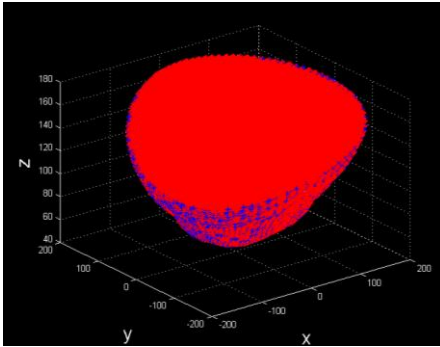
```
Objetosc =

   2.7584e+006
```


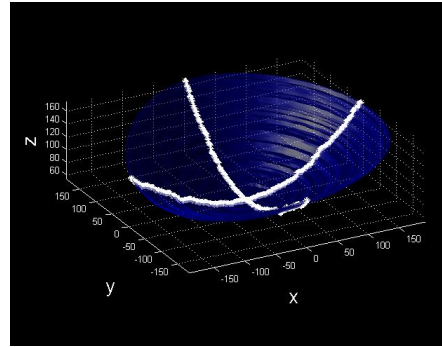**Fig. 3-57 Anterior chamber with calculated volume marked red.**

Fig. 3-58 Anterior chamber with calculated volume marked in a form of blue envelope.

Fig. 3-58 presents the outside envelope of the measured volume, i.e.:

```
figure
fd=surf(xc,yc,zc,'FaceColor',[0 0 1],...
'EdgeColor','none',...
'FaceLighting','phong')
daspect([5 5 1])
view(-50,30)
camlight lef
set(fd,'FaceAlpha',.5)
hold on
plot3(xa1_aa,ya1_aa,za1_aa,'-w*'); grid on; hold on
plot3(xa1_bb,ya1_bb,za1_bb,'-w*');
plot3(xa2_aa,ya2_aa,za2_aa,'-w*');
plot3(xa2_bb,ya2_bb,za2_bb,'-w*');
axis equal
xlabel('x','FontSize',20); ylabel('y','FontSize',20);
zlabel('z','FontSize',20)
```

To confirm visual correctness of calculations and of automation it is possible to overlap component images Ls1 and Ls2 on the envelope formed (Fig. 3-59) tj.:

```
Ls1=imresize(Ls1,[256 512 ]);
Ls2=imresize(Ls2,[256 512 ]);
 [XX,YY]=meshgrid(1:size(Ls1,2),1:size(Ls1,1));
 Ls1=mat2gray(Ls1);
```

```
Ls1=uint8(round(histeq(Ls1)*255));
Ls1=cat(3,Ls1,Ls1,Ls1);
surface(ones(size(XX)),XX-
mm1/(size(Ls1,1)/256),YY,(Ls1),...
        'FaceColor','texturemap',...
        'EdgeColor','none',...
        'CDataMapping','direct')
    [XX,YY]=meshgrid(1:size(Ls2,2),1:size(Ls2,1));
 Ls2=mat2gray(Ls2);
Ls2=uint8(round(histeq(Ls2)*255));
Ls2=cat(3,Ls2,Ls2,Ls2);
surface(XX-
mm2/(size(Ls2,1)/256),ones(size(XX)),YY,(Ls2),...
        'FaceColor','texturemap',...
        'EdgeColor','none',...
        'CDataMapping','direct')
```
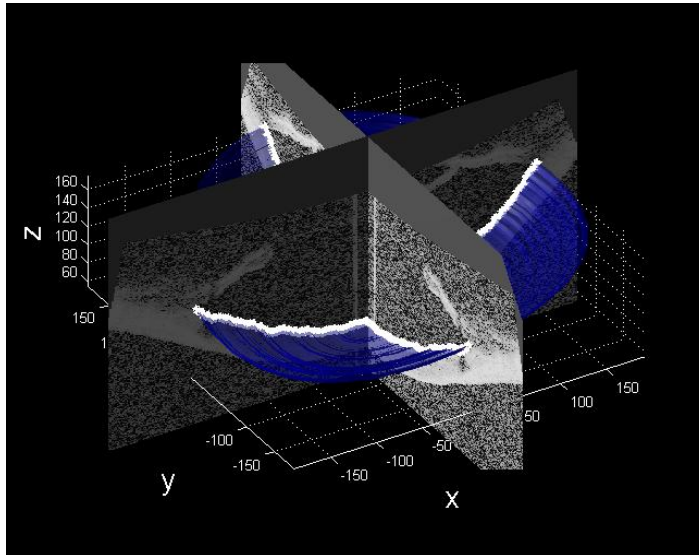

Fig. 3-59 Anterior chamber with calculated volume together with component flat images

The algorithm presented has numerous drawbacks and we encourage Readers to remove them. These drawbacks include:

- the calculated volume is understated,
- the shape of the top surface is not considered at all,
- the calculated volume is expressed only in pixels – it should be converted to appropriate unit of volume, reading the unit of distance falling per pixel from the file header.

Summarising, the algorithm calculates the anterior chamber volume in a fully automated way. The computation time for a PC class computer with the Windows Vista operating system, Intel Core Quad CPU Q9300, 2.5 GHz processor, 8GB RAM amounts to approx. 2 s.

PART II

4 ANALYSIS OF POSTERIOR EYE SEGMENT

The second part of this monograph presents the issues of posterior eye segment with special emphasis on automated methods for individual layers detection. Also the optic nerve head and the degree of retinal detachment will be fully automatically analysed. The measurements performed provide a possibility of not only obtaining quantitative data but also of automated determination of individual layers thickness maps.
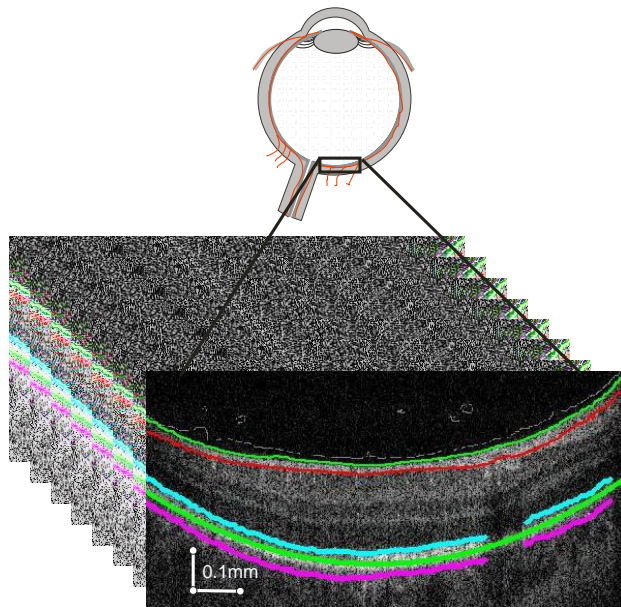
4.1 Introduction to the fundus of the eye analysis

The analysis of the fundus of the eye in its initial part is similar to the analysis of the anterior eye segment [5], [11], [12], [13]. This applies to the DICOM image acquisition and entering to the Matlab space as well as to acquiring the header and comprised by it patient and other data. Methods and tools intended for that have been discussed in detail in the first section of this monograph. The methodology for the image analysis has been presented below assuming that it already had been introduced to the Matlab space.

The input images L_{GRAY} acquired e.g. from an optical tomograph SOCT Copernicus of the following parameters: the light source wavelength: 840nm, spectrum width of 50nm, axial (longitudinal) resolution: 6μm, transverse resolution: 12-18 μm, tomogram window width: 2mm, measurement rate: 25,000 A scans per second, the maximum scanning width: 10mm, the maximum number of A scans falling per a B scan: 10'500, were saved as grey levels of MxN = 722x928 resolution, where 8 bits falls per each pixel.
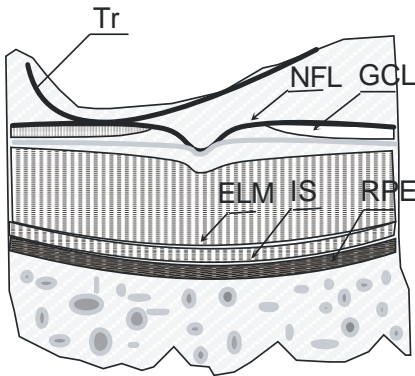
Fig. 4-1 Diagram of individual layers cross-section with marked characteristic measured areas, where: Tr – traction, NFL neural fibre layer – internal retina Bondary, RPE retinal pigment epithelium
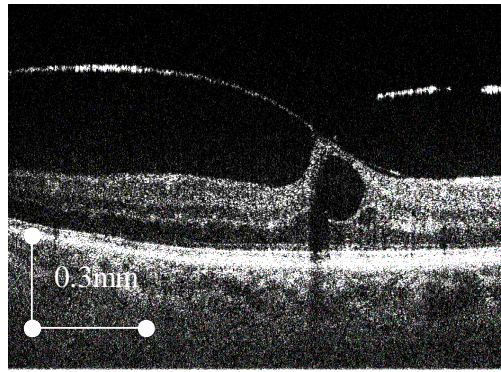
Fig. 4-2 Example image acquired from SOCT Copernicus.

The identification of individual layers position, starting from the nerve fibre layer (NFL), ganglion cell layer (GCL), inner plexiform layer (IPL), inner nuclear layer (INL), outer plexiform layer (OPL), inner and outer segment of photoreceptors (IS/OS) and ending at retinal pigment

epithelium (RPE) and choriocapillaris (CC) situated between the inner limiting membrane (ILM) and the CC has been shown in Fig. 4-1 and Fig. 4-2.
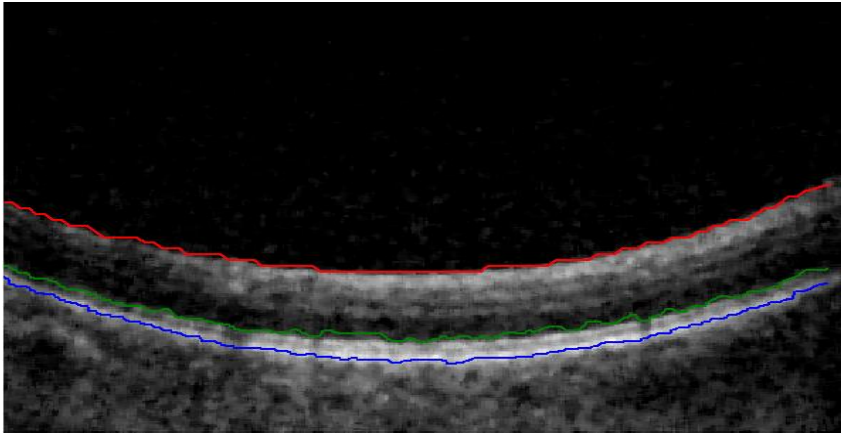
Fig. 4-3 Example tomograph image with marked layers NFL – red, ONL - green, RPE – blue

Na Fig. 4-3 shows layers put on the L_M image detected by means of algorithm described in this monograph, i.e. NFL, ONL and RPE. The position of those layers provides the grounds for further methodology, described in this paper.

Further considerations will refer to methods automatically determining the boundaries of layers visible in Fig. 4-1 i.e.: tractions, internal retina boundary, RNFL/GCL boundary, IS/OS boundary, OS/RPE and RPE boundary preceded by the analysis of results obtained using known algorithms [1], [3], [17], [19], [22], [33], [36], [43].

4.2   Algorithm for Automated Analysis of Eye Layers in the Classical Method

The algorithm proposed by the authors, presented below, has a modular (block) structure, where selected blocks can operate independently of each other - Fig. 4-39.
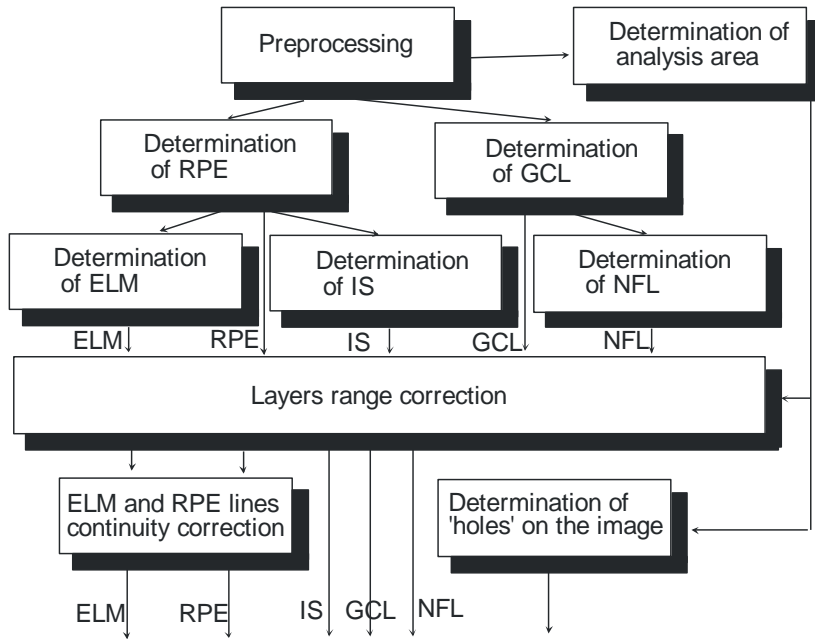
Fig. 4-4 Block diagram of fundus of the eye analysis algorithm

The block diagram presented in Fig. 4-39 divides the algorithm operation into five stages:

- Preprocessing – median filter filtration and normalisation.
- Determination of RPE layer position and then, using a modified active contour method, of ONL and IS.
- Determination of NFL internal retina boundary position and then of GCL areas (usually two).
- Correction of layers obtained with regard to the analysis area – considering the quality by areas of the object presented.
- Determination, based on the image qualitative analysis, of 'holes', local brightness minima.

These stages will be the subject of considerations in the next sections.

4.2.1   Preprocessing

Preliminary algorithms for image processing include filtration with a median filter of square mask, 21x21 in size, to eliminate noise and small artefacts introduced by the measuring system during the image acquisition. The mask size was selected arbitrarily. In addition, the image

was cut at the bottom to correct erroneous instrument readings for the last two lines of the image, i.e.:

```
[Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255; Lorg=Lgray;
Lmed=medfilt2(Lorg,[5 5]);
```

The second component consisted of normalisation from the range of minimum and maximum pixel brightness to a full range between 0 and 1, i.e.:

```
Lmed=mat2gray(Lmed);
figure;
imshow(Lmed)
```

The L_{GRAY} images converted this way were analysed using available algorithms, which in this case – the necessity to detect discontinuous line ranges – did not provide satisfactory results.

4.2.2   Detection of RPE Boundary

The RPE layer is the first and the simplest to determine in an automated determination on an OCT image. It is perfectly visible on the OCT image as the brightest area for each column. This property has been used to create the first part of the algorithm.

The analysis of L_{GRAY} images after images preprocessing (filtration and normalisation, obtaining L_{MED}) was started analysing the position of maximum for consecutive columns. If m and n denote rows and columns of image matrix, then the new image:

$$L_{BIN\_RPE}(m,n) = \begin{cases} 1 & \text{dla} \quad L_{MED}(m,n) > \max_{m \in \{1,2,...M\}}(L_{MED}(m,n)) \cdot pr \\ 0 & \text{dla} \qquad \qquad \text{pozostale} \end{cases} \tag{11}$$

  dla $n \in \{1,2,3,...,N-1,N\}$

where pr – parameter of decimal-to-binary conversion threshold, assumed as 0.9 (90%).

The L_{BIN_RPE} image contains values '1' in places, where pixels in a given column are brighter than 90% of the maximum occurring brightness for this column. Values '0' occur in the other places. The image obtained this way is shown below.
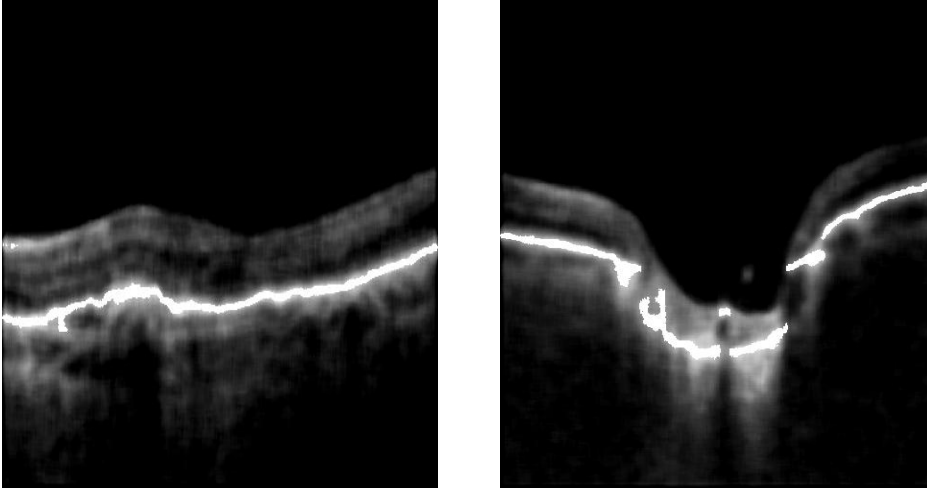
Fig. 4-5 Sum of L$_{BIN\_RPE}$ images with weight 50% and L$_{MED}$ with 50%; a) image with properly detected lp area and b) image, where RPE area is discontinuous in ranges.

In the next stage the position of the longest section centre for each column of L$_{BIN\_RPE}$ image was calculated, obtaining y$_{RPE}$, i.e.:

$$y_{RPE}(n) = \sum_{m=1}^{M} y_W(m,n) \Big/ \sum_{m=1}^{M} L_{BIN\_RPE}(m,n) \tag{12}$$

where:

$$y_W(m,n) = \begin{cases} m & \text{dla} & L_{BIN\_RPE}(m,n) \neq 0 \\ 0 & \text{dla} & L_{BIN\_RPE}(m,n) = 0 \end{cases} \tag{13}$$

$n \in \{1,2,3,...,N-1,N\}$

The obtained course of y$_{RPE}$ and the source code are shown below:

```
x=(1:size(Lmed,2))';
yyy=(1:size(Lmed,1))';
yrpe=[];
Lk=zeros(size(Lmed));
for ik=1:size(Lmed,2)
    xx_best=[];
    Llabp=bwlabel(Lmed(:,ik)>(max(Lmed(:,ik))*0.9));
    Lk(:,ik)=Llabp;
    for tt=1:max(Llabp)
        xxl=yyy(Llabp==tt);
        xx_best=[xx_best;mean(xxl)] ;
```

```
    end
    if ~isempty(xx_best)
        yrpe(ik)=max(xx_best);
    else
        yrpe(ik)=0;
    end
end
figure; imshow(mat2gray(Lk*0.5+Lmed));hold on;
plot(yrpe,'r*-')
```
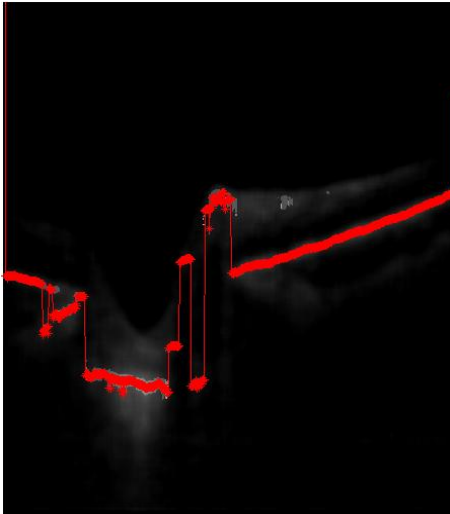

Fig. 4-6 Sum of L$_{BIN\_RPE}$ images with weight 50% and L$_{MED}$ with 50% and marked course of y$_{RPE}$
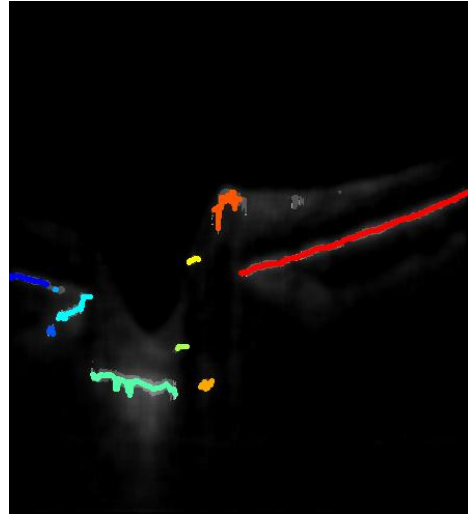
Fig. 4-7 Sum of L$_{BIN\_RPE}$ images with weight 50% and L$_{MED}$ with 50% and marked course of y$_{RPES}$

The course of y$_{RPE}$ function is further analysed for clusters using k-means method, obtaining y$_{RPES}^{(k)}$ for each k-cluster. Then (y$_{RPES}^{(k1,k2)}$) is approximated by a 3rd order polynomial for each pair y$_{RPES}^{(k1)}$ and y$_{RPES}^{(k2)}$ for k$_1 \neq$k$_2$. All obtained polynomial functions y$_{RPES}^{(k1,k2)}$ determined for all possible cluster pairs (k$_1$, k$_2$) are shown in Fig. 4-8 and an appropriate part of algorithm is given below:

```
yg=gradient(yrpe);
ygg=ones([1 length(yrpe)]); ygg(abs(yg)>20)=0;
ygl=bwlabel(ygg);
figure; imshow(mat2gray(Lbinrpe*0.5+Lmed));hold on;
palett=jet(max(ygl));
for iiih=1:max(ygl(:))
        plot(x(ygl==iiih),
yrpe(ygl==iiih),'Color',palett(iiih,:),'LineWidth',4);
end
pam_dl=[];
```

```
figure; imshow(mat2gray(Lbinrpe*0.5+Lmed)); hold on
for iiik=1:max(ygl(:))
    for iiikk=iiik:max(ygl(:))
        if iiik<=iiikk
            ygk=[yrpe(ygl==iiik),yrpe(ygl==iiikk)];
            xgk=[x(ygl==iiik);x(ygl==iiikk)];
        else
            ygk=[yrpe(ygl==iiikk),yrpe(ygl==iiik)];
            xgk=[x(ygl==iiikk);x(ygl==iiik)];
        end
        if length(ygk)>10
            P = POLYFIT(xgk',ygk,2); yrpes =
round(POLYVAL(P,x));
            plot(yrpes,'g*-')
            pam_dl=[pam_dl;[iiik iiikk sum(abs(yrpe-
yrpes')<20)]];
        end
    end
end
```


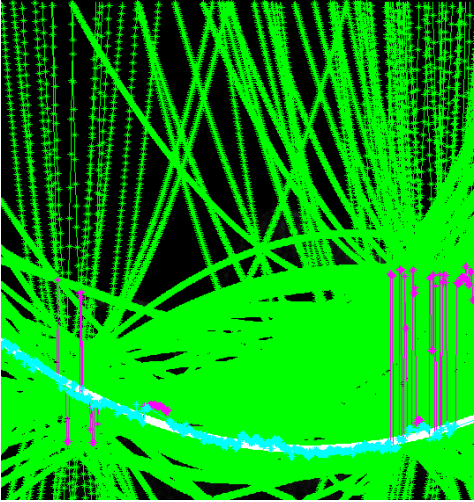**Fig. 4-8 3rd order functions $y_{RPES}(k_1, k_2)$ for all possible cluster pairs**

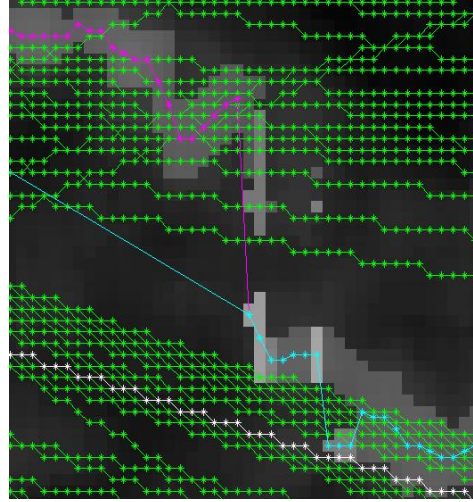**Fig. 4-9 Enlarged fragment of image from Fig. 4-8**

$$y_{RPE}(n) = \sum_{m=1}^{M} y_W(m,n) / \sum_{m=1}^{M} L_{BIN\_RPE}(m,n)$$

The number of points from the range ± 15 pixels, i.e. $pr_1 = 15$ and $pr_2 = 15$ is determined for each function.

$$s(k_1, k_2) = \sum y_{RPEB}^{(k1, k2)}(n) \tag{14}$$

$$y_{RPEB}^{(k_1, k_2)}(n) = \begin{cases} 1 & \text{dla} & pr_1 < y_{RPES}^{(k_1, k_2)}(n) < pr_2 \\ 0 & \text{dla} & \text{other} \end{cases} \tag{15}$$

Then this pair (k_1, k_2) is determined, for which:

$$s(k_1^{*}, k_2^{*}) = \max_{k_1, k_2}(s(k_1, k_2)) \tag{16}$$

The pair determined achieves the maximum value at selected $y_{RPES}^{(k1*, k2*)}$ later on named simply y_{RPEC} function. The implementation of the algorithm fragment described above is provided below:

```
pam_s=sortrows(pam_dl,-3);
if size(pam_s,1)==1
        ygk=[yrpe(ygl==pam_s(1,1))];
        xgk=[x(ygl==pam_s(1,1))];
else
        ygk=[yrpe(ygl==pam_s(1,1)),yrpe(ygl==pam_s(1,2))];
        xgk=[x(ygl==pam_s(1,1));x(ygl==pam_s(1,2))];
end
    P = POLYFIT(xgk',ygk,2); yrpes = round(POLYVAL(P,x));
    plot(x,yrpes,'w*-');
yrpe=yrpe(:);
    plot(x,yrpe,'m*-');
```

In further considerations also these points of y_{RPE} are important, which fall within the tolerance predetermined regarding $y_{RPES}^{(k1*, k2*)}$, i.e.:

```
dx=x; dx(abs(yrpe-yrpes)>20)=[];
yrpe(abs(yrpe-yrpes)>20)=[];
dxl=bwlabel(diff(dx)<125);
pdxl=[];
for qw=1:max(dxl)
   pdxl=[pdxl;[qw, sum(dxl==qw)]];
end
pdxl(pdxl(:,2)<50,:)=[];
dxx=[]; dyy=[];
for wq=1:size(pdxl,1)
    dxx=[dxx; dx(dxl==pdxl(wq,1))];
    dyy=[dyy; yrpe(dxl==pdxl(wq,1))];
end
dx=dxx; yrpe=dyy;
    plot(dx,yrpe,'c*-');
```

```
figure
imshow(Lgray); hold on
    plot(dx,yrpe,'c*-');
```

The results obtained are presented in the following figure (Fig. 4-10, Fig. 4-11).
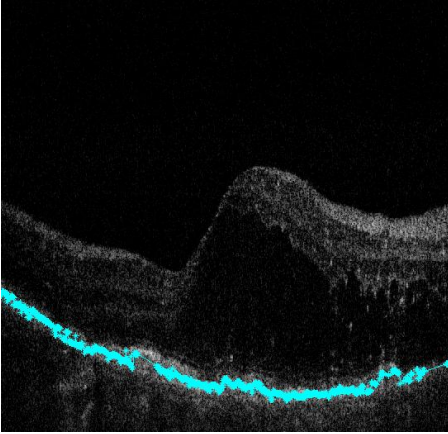
Fig. 4-10 Function y$_{RPEC}$ satisfying the conditions given
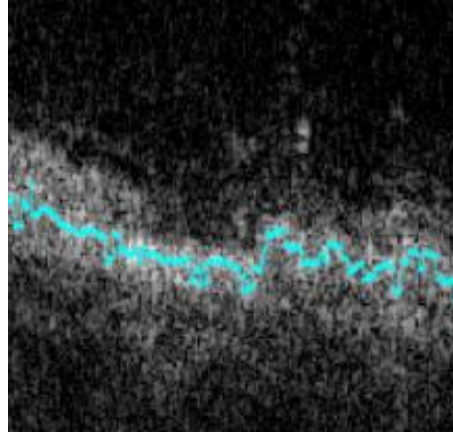
Fig. 4-11 Enlargement of image from Fig. 4-10

The y$_{RPEC}$ values will further, in the next section, provide the basis to determine IS and ONL boundaries.

4.3   Detection of IS, ONL Boundaries

Boundaries of IS and ONL were determined on the basis of y$_{RPEC}$ limit. In both cases algorithms were very similar and in their largest fragment applied to the modified active contour method [29], [41]. This method was used to analyse the anterior eye segment in the first part of this monograph and the function intended for its proper operation noted as OCT_activ_cont. This operation could also be performed (obtaining similar results) using other methods, e.g. of the convolution with mask h presented below (Fig. 4-12) or of filtration by a median filter and calculating differences between pixels situated on the oy axis distant from each other by the number of mask rows.

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$
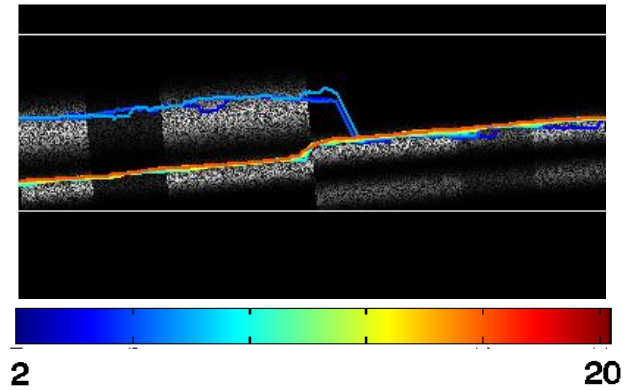
Fig. 4-12 Mask h used for independent calculations of modified active contour method

Fig. 4-13 Artificial input image with y_{IS} courses for parameters $p_{yu} = p_{yd}$ changing within the range from 2 – blue colour to 20 – red colour

The change of operation selectivity in the sense of individual layers distinction accuracy is obtained depending on the selection of parameters p_{yu} and p_{yd}. Such situation is illustrated by Fig. 4-13 where p_{yu} and p_{yd} were changing between 2 and 20 for an artificial image created as follows:

```
L1=rand([201 200]);
xx=-1:0.01:1;
y=gauss(xx+0.5,0.2)+0.5*gauss(xx-0.1,0.05);
Ly=y'*ones([1 200]);
Ly=mat2gray(Ly);
Lw1=L1.*Ly;
L1=rand([201 200]);
y=gauss(xx,0.2)+0.5*gauss(xx-0.4,0.05);
Ly=y'*ones([1 200]);
Ly=mat2gray(Ly);
Lw2=L1.*Ly;
Lw=[Lw1,Lw2];
Lw(:,300:350)=Lw(:,300:350)*.5;
Lw(:,50:100)=Lw(:,50:100)*.2;
Lw=imrotate(Lw,5,'crop');
figure; imshow(Lw)
```

where the `gauss` function has the following form:

```
function y = gauss(x,std)
y = exp(-x.^2/(2*std^2)) / (std*sqrt(2*pi));
```

The change of parameters p_{yu} and p_{yd} values affects the selectivity of algorithm operation. The remaining parameters, such as p_u or p_d, determine the range of search on the vertical axis. Parameters p_{xl} and p_{xp} are the range on the ox axis, from which values Lu and Ld are calculated. They have a direct influence on the algorithm behaviour in places, where shadows occur. Fig. 4-14 shows the influence of parameters p_{xl} and p_{xp} settings on the results obtained.
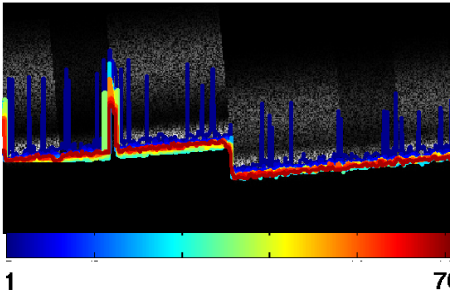
Fig. 4-14 Artificial input image with y_{IS} courses for parameters $p_u= p_d=50$, $p_{xud}=\infty$ and $p_{xl} =p_{xp}$ changing within the range from 1 – blue colour to 70 – red colour	**Fig. 4-15 Artificial input image with y_{IS} courses for parameters $p_u= p_d=50$, $p_{xud}=2$ and $p_{xl} =p_{xp}$ changing within the range from 1 – blue colour to 70 – red colour**
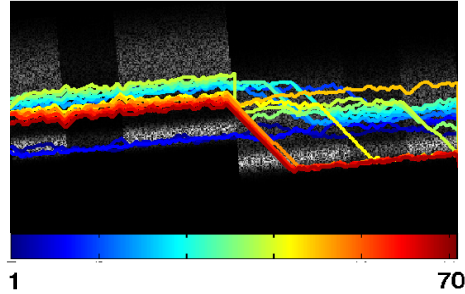
Images have been obtained at the following implementation in Matlab:

```
x=1:size(Lw,2);
y=round(   [ones([1 size(Lw,2)/2])*size(Lw,1)/3 ones([1
size(Lw,2)/2])*size(Lw,1)/2]   );
map=jet(70);
for pyud=1:4:70
pud=50;
pxud=2;
pxlp=1;
polaryzacja=-1;
[yy,i]=OCT_activ_cont(Lw, x,y+20, pud,  pyud, pxud,  pxlp,
polaryzacja);
hold on
plot(x,yy,'Color',map(pyud,:),'LineWidth',3)
pause(0.001)
end
```

As can be seen from Fig. 4-14 and Fig. 4-15 small values of p_{xl} and p_{xp} in the range from around $1 \div 10$ result in the origination of large changes in positions of its consecutive values on the oy axis of y_{IO} course. Values of parameters p_{xl} and p_{xp} changed in the range from around $10 \div 70$ 'stabilise' the course of y_{IO} due to which it becomes less sensitive to sudden changes of brightness (e.g. shadows) on the image.

The influence of parameters p_{xl}, p_{xp} and p_{yu}, p_{yd} can be best followed on the graph of error $\delta_{IO}(p_{xl=}p_{xp}, p_{yu}=p_{yd})$ defined as:

$$\delta_{IO}\left(p_{xl} = p_{xp}, p_{yu} = p_{ud}\right) = \frac{1}{N} \cdot \sum_{n=1}^{N} \left| \frac{y_{IO}(n) - y_{IOW}(n)}{y_{IOW}(n)} \right| \cdot 100 \quad \% \tag{17}$$

where y_{ISW} – a model course of y_{IS}.
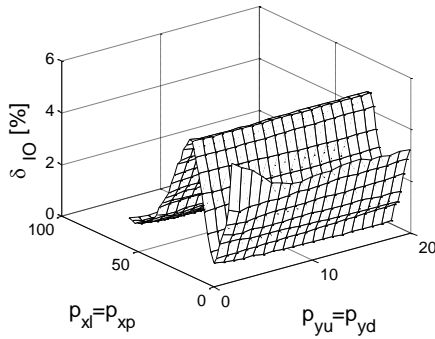
Fig. 4-16 Graph of error δ_{IS} values changes for changes of parameters $p_{xl} = p_{xp}$ within the range 1-70 and $p_{yu} = p_{yd}$ within the range 1-20 for $p_{xud}=\infty$
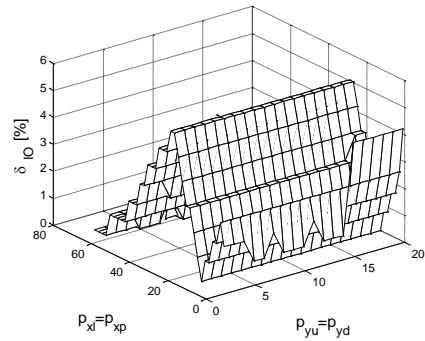
Fig. 4-17 Graph of error δ_{IS} values changes for changes of parameters $p_{xl} = p_{xp}$ within the range 1-70 and $p_{yu} = p_{yd}$ within the range 1-20 for $p_{xud}=1$

In accordance with the graph presented in Fig. 4-16 parameter $p_{xl} = p_{xp}$ for $p_{xud}=\infty$ has the largest influence on the value of δ_{IS} error. Because of two characteristic areas visible on the L_{GRAY} image (Fig. 4-16) the course of error has a local maximum for $p_{xl}=p_{xp}\cong40$. The course of error δ_{IS} value for $p_{xud}=1$ (Fig. 4-17) is similar, where parameter p_{xud} had no significant impact on its value.

The graphs discussed were generated using the function:

```
L1=rand([201 200]);
xx=-1:0.01:1;
y=gauss(xx+0.5,0.2)+0.5*gauss(xx-0.1,0.05);
```

```
Ly=y'*ones([1 200]);
Ly=mat2gray(Ly);
Lw1=L1.*Ly;
Lw=Lw1;
Lw(:,50:100)=Lw(:,50:100)*.2;
figure; imshow(Lw)
x=1:size(Lw,2);
y=round(   [ones([1 size(Lw,2)/2])*size(Lw,1)/3 ones([1
size(Lw,2)/2])*size(Lw,1)/2]    );
map=jet(70);
hold on
plot(x,y,'r','LineWidth',3)
d3_wy=[];
        pub=50;
        pxud=1;
        polaryzacja=-1;
jj=1;
for pxlp=2:1:20
    ii=1;
    for pyud=1:2:70
        [yy,i]=OCT_activ_cont(Lw, x,y+20,  pub, pyud, pxud,
pxlp, polaryzacja);
        d3_wy(ii,jj)=sum(abs(119-yy)./119)/length(yy)*100;
        ii=ii+1;
        [ii,jj]
    end
    jj=jj+1;

end
[XX,YY]=meshgrid(2:1:20,1:2:70);
figure; mesh(XX,YY,d3_wy);
ylabel('p_{xl}=p_{xp}','FontSize',20)
xlabel('p_{yu}=p_{yd}','FontSize',20)
zlabel('\delta _{IO} [%]','FontSize',20)
colormap([0 0 0])
set(gca,'FontSize',15)
```

The sensitivity to a Gaussian noise, which may appear on the image, is a totally different feature of the algorithm discussed. To evaluate the quality of algorithm proposed a Gaussian noise of variance σ changed between 0 and 0.9 was added to the L_{GRAY} image.
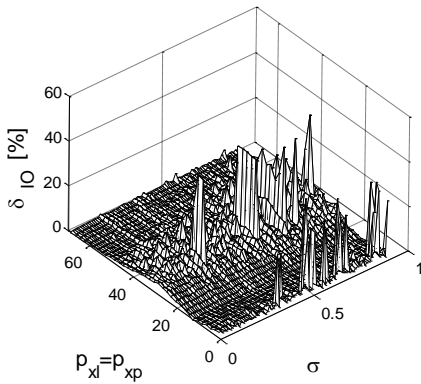
Fig. 4-18 Graph of error δ_{IS} values changes for changes of parameters $p_{xl} = p_{xp}$ within the range 1-70 and of variance σ within the range 0÷0.9 for p_{xud}=2
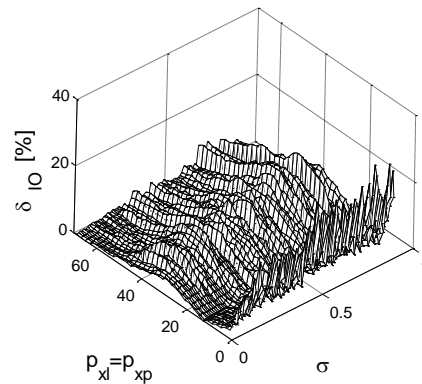
Fig. 4-19 Graph of error δ_{IS} values changes for changes of parameters $p_{xl} = p_{xp}$ within the range 1-70 and of variance σ within the range 0÷0.9 for p_{xud}=∞

Graphs in Fig. 4-18 and Fig. 4-19 show changes of error δ_{IS} values for changes of parameters $p_{xl}=p_{xp}$ within the range 1-70 and of variance σ within the range 0÷0.9 for p_{xud}=2 pixels and p_{xud}=∞. For both graphs at the change of σ values within the range 0-0.3 and $p_{xl}=p_{xp}$ within 50-70 pixels the δ_{IO} error does not exceed 5%. The dependence of error δ_{IS} value on p_{xud} is insignificant, mainly due to its definitions), where large changes of isolated points of y_{IS} course have no significant impact on the δ_{IS} error. The nature of error δ_{IS} values changes shown in Fig. 4-16 and Fig. 4-17 as well as in Fig. 4-18 and Fig. 4-19 regarding changes of parameters $p_{xl}=p_{xp}$ within their full range depends mainly on the nature and arrangement of objects on the scene and therefore it will not be discussed here. The form of algorithm intended to generate the above results is similar to the previous case.

4.4   Detection of NFL Boundary

The NFL boundary position was determined in two stages, of which the second stage of individual points positions correction is the most complicated and the analysis laborious.

The first stage comprises determination of decimal to binary conversion for each column of L_{MED} image acc. to previously mentioned relationship for parameter pr assumed arbitrarily around 0.1 (10%). Then, for each column of L_{BIN_NFL} image, the position of the first pixel for each

k_n- cluster of value '1' for each column – n is calculated. Assuming further that each column n has K_n clusters it is possible to write:

$$y_{NFL\_P}(k_n, m^*, n) = \min_{m \in (1,M)} \left(y_{NFL\_W}(m, n, k_n) \right) \tag{18}$$

where:

$$y_{NFL\_W}(m, n, k_n) = \begin{cases} M & dla & L_{ET\_N}(m, n) \neq k_n \\ m & dla & L_{ET\_N}(m, n) = k_n \end{cases} \tag{19}$$

and L_{ET_N} - image formed as a result of labelling each cluster for each column irrespective of L_{BIN_NFL} image for $k_n \in \{1,2,3,...,K_n\text{-}1,K_n\}$.

Fig. 4-20 and Fig. 4-22 show L_{ET_N} images for artificial input image L_{MED} without the added noise (Fig. 4-21) and with added Gaussian noise of variance σ=0.2 (Fig. 4-23).
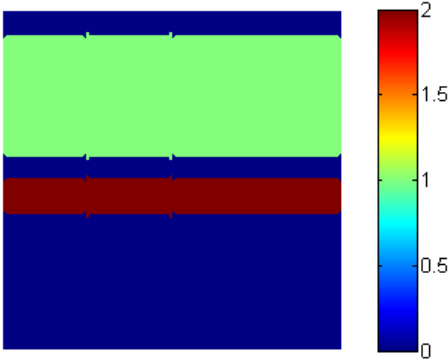
Fig. 4-20 Image L_{ET_N} formed from the input L_{MED} image shown in Fig. 4-21
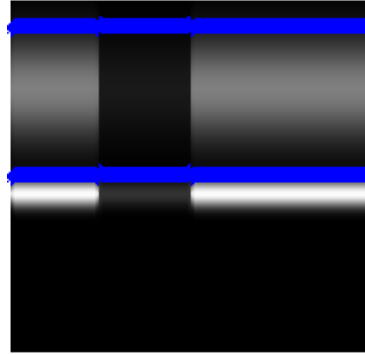
Fig. 4-21 Image L_{MED} resulting from the filtration, using a median filter, of artificial image L_{GRAY} with marked blue points y_{NFL_P}
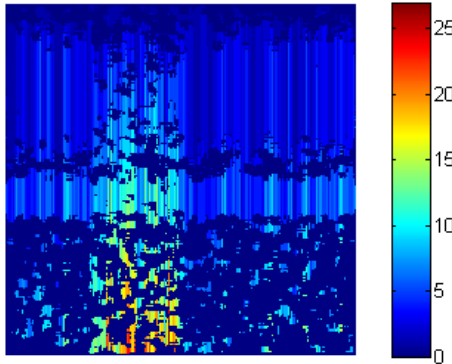
Fig. 4-22 Image L_{ET_N} formed from the input L_{MED} image shown in Fig. 4-23
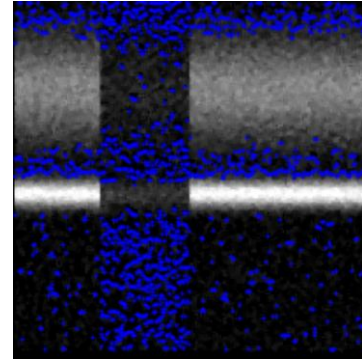
Fig. 4-23 Image L_{MED} resulting from the filtration, using a median filter, of artificial image L_{GRAY} with added Gaussian noise and with marked blue points y_{NFL_P}

The image from Fig. 4-23 originated at the following implementation:

```
L1=rand([201 200]);
xx=-1:0.01:1;
y=gauss(xx+0.5,0.2)+0.5*gauss(xx-0.1,0.05);
Lmed=y'*ones([1 200]);
Lmed=mat2gray(Lmed);
Lmed(:,50:100)=Lmed(:,50:100)*.2;
Lmed = imnoise(Lmed,'gaussian',0.02);
Lmed=medfilt2(Lmed,[3 3]);
figure; imshow(Lmed); hold on
xyinfy=[];
xyinfdl=[];
for ik=1:size(Lmed,2)
        grL1=Lmed(:,ik)>(max(Lmed(:,ik))*0.1);
        lgrL1=bwlabel(grL1);
            for jju=1:max(lgrL1)
                xyinfdl(jju,ik)=sum(lgrL1==jju);
                cuu=1:length(lgrL1);
cuu(lgrL1~=jju)=[];
                xyinfy(jju,ik)=cuu(1);
                plot(ik,cuu(1),'b*')
            end
end
```

As shown in Fig. 4-20 - Fig. 4-23 the relationship (18) and (19) is very sensitive to noise and to small artefacts on the image, which are the reason for origination of additional erroneous points y_{NFL_P}. In practice,

however, this problem is not too arduous because even in the case of proper distribution of points y_{NFL_P} the determination of NFL line is not an unambiguous and simple process, which is illustrated by Fig. 4-24.
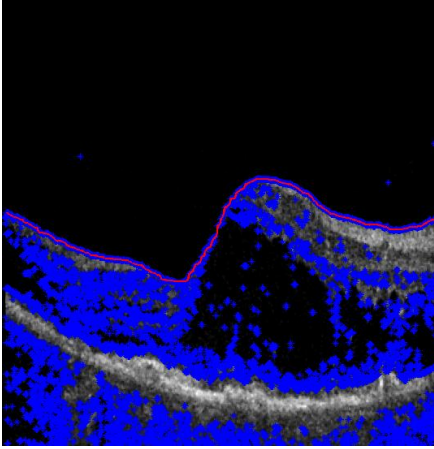
Fig. 4-24 Image L$_{MED}$ of actual image L$_{GRAY}$ with marked blue corresponding points y$_{NFL\_P}$
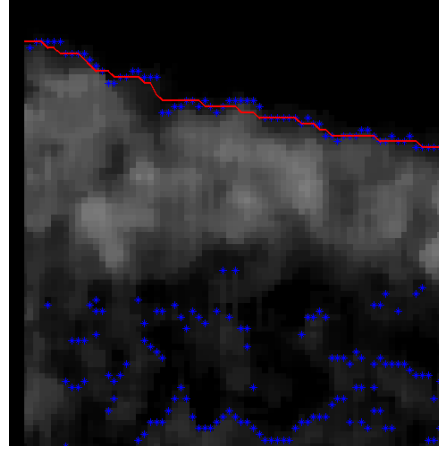
Fig. 4-25 Enlarged L$_{MED}$ image from Fig. 4-25

This figure was obtained from the following algorithm.

```
  [Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255; Lorg=Lgray;
Lmed=medfilt2(Lorg,[5 5]);
Lmed=mat2gray(Lmed);
figure; imshow(Lmed)
grad_y_punkt=30;
figure; imshow(Lmed); hold on
  [xNFL,yNFL,xyinfdl,xyinfy,ggtxnn,ggtynn,ggdlnn,xyinfdl_o
ld,xyinfy_old]=OCT_NFL_line(Lmed,grad_y_punkt);
plot(xNFL,yNFL,'r','LineWidth',2)
```

where function `OCT_NFL_line` is intended to analyse the course of NFL line and is described below.

The second stage of NFL line determination is related to the analysis of y_{NFL_P} points on the ox axis. For the next y_{NFL_P} points a derivative for the ox axis was calculated and then the clusters analysis was performed, obtaining this way k_m clusters and y_{NFL_D} where for each $k_m \in \{1,2,3,...,K_m-1,K_m\}$ the following condition is satisfied:

$$y_{NFL\_D}(k_m, m, n) = \left| \frac{\partial y_{NFL\_P}(k_n, m^*, n)}{\partial n} \right| < p_{rd} \tag{20}$$

where p_{rd} is the threshold limiting the maximum value of the derivative for consecutive points on the ox axis. This threshold is directly responsible for the obtained number of clusters and thereby the number of sections analysed in further part of the algorithm.

Clusters containing too small number of elements (less than 20% of the largest cluster) are automatically cut off. Instead, the others are analysed in terms of arrangement on the image (coordinate m) and of the number of pixels existing in a specific cluster (y_{NFL_H}).

$$y_{NFL\_H}(k_m) = \sum_{n=1}^{N} \sum_{m=1}^{M} \left(y_{NFL\_D}(k_m, m, n) \right) \tag{21}$$

So analysing the position of individual y_{NFL_S} points and the number of pixels in specific y_{NFL_H} cluster for which they were determined it is possible to create weights y_W for analysed clusters (points groups), i.e.:

$$y_W(k_m) = y_{NFL\_H}(k_m) / \max_{k_m \in (1, K_m)} \left(y_{NFL\_H}(k_m) \right) \cdot \varepsilon_P + y_{NFL\_S}(k_m) / \max_{k_m \in (1, K_m)} \left(y_{NFL\_S}(k_m) \right) \cdot \varepsilon \tag{22}$$

where ε_S and ε_P are constants arbitrarily selected from the 0-1 range and

$$y_{NFL\_S}(k_m) = \max_{m \in (1, M), n \in (1, N)} \left(y_{NFL\_D}(k_m, m, n) \right) \tag{23}$$

In the next stage this cluster k_m is selected, which has the largest weight k_{m^*}. Later on it is used as a start vector for the modified active contour method described in section 0. This way the results presented in Fig. 4-26 are obtained.
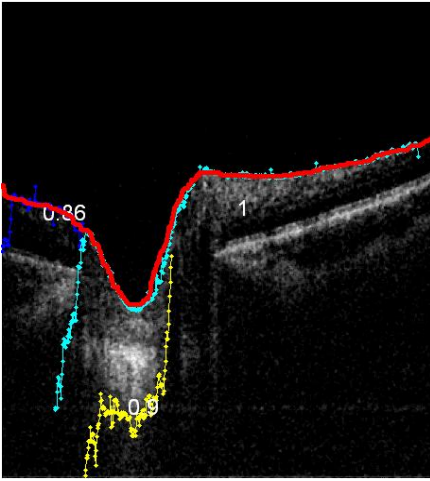
Fig. 4-26 Image L_MED with marked red y_NFL points for the best, with respect to the criterion set, cluster k_m* (turquoise) and with results obtained for the active contour method (red)
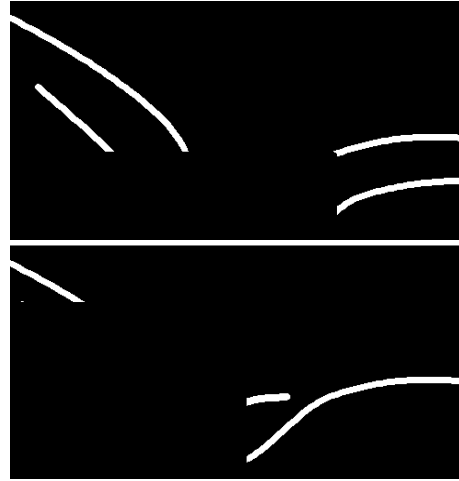
Fig. 4-27 Demonstrative images of layers arrangement on an OCT image, for which the algorithm described operates improperly

Fig. 4-26 shows points for the best, with respect to the criterion set, cluster k_{m*} in turquoise and y_{NFL} results obtained for the active contour method in red.

Taking into account the above analysis the final shape of `OCT_NFL_line` function was formulated as follows:

```
function
[xNFL,yNFL,xyinfdl,xyinfy,ggtxnn,ggtynn,ggdlnn,xyinfdl_old,
xyinfy_old]=OCT_NFL_line(Lmed,grad_y_punkt)

xyinfy=[];
xyinfdl=[];
for ik=1:size(Lmed,2)
        grL1=Lmed(:,ik)>(max(Lmed(:,ik))*0.1);
        lgrL1=bwlabel(grL1);
            for jju=1:max(lgrL1)
                xyinfdl(jju,ik)=sum(lgrL1==jju);
                cuu=1:length(lgrL1);
cuu(lgrL1~=jju)=[];
                xyinfy(jju,ik)=cuu(1);
                plot(ik,cuu(1),'b*')
            end
end
xyinfdl_old=xyinfdl;
xyinfy_old=xyinfy;
```

```
ggtxnn=[];
ggtynn=[];
ggdlnn=[];
while sum(sum(xyinfy(:,1:(end-1))))~=0
    ggtx=[];
    ggty=[];
    for hvi=1:(size(xyinfy,2)-1)
        if sum(xyinfy(:,hvi))~=0
            break
        end
    end
    for hv=hvi:(size(xyinfy,2)-1)
        if  (min(  abs(xyinfy(1,hv)-xyinfy(:,hv+1))
)<grad_y_punkt)&(xyinfy(1,hv)~=0)
            vff=1:size(xyinfy,1); vff(abs(xyinfy(1,hv)-
xyinfy(:,hv+1))>=grad_y_punkt)=[]; vff=vff(1);
xypam=xyinfy(1,hv);
            vff__=1:size(xyinfy,1); vff__(vff)=[];
            xyinfy(1:end,hv+1) = [xyinfy(vff,hv+1);
xyinfy(vff__,hv+1)];
            xyinfdl(1:end,hv+1)= [xyinfdl(vff,hv+1);
xyinfdl(vff__,hv+1)];
            xyinfy(1:end,hv)=[xyinfy(2:end,hv);0];
            xyinfdl(1:end,hv)=[xyinfdl(2:end,hv);0];
            ggtx=[ggtx,hv];
            ggty=[ggty,xypam];
        else
            xyinfy(1:end,hv)=[xyinfy(2:end,hv);0];
            xyinfdl(1:end,hv)=[xyinfdl(2:end,hv);0];
            break
        end
    end
    if length(ggty)>10
        ggtxnn(size(ggtxnn,1)+1,1:length(ggtx))=ggtx;
        ggtynn(size(ggtynn,1)+1,1:length(ggty))=ggty;
        ggdlnn=[ggdlnn;[length(ggty) min(ggty)]];
    end
end
ggdlnn_leng=ggdlnn(:,1);
ggdlnn=[(1:size(ggdlnn,1))',ggdlnn];
ggdlnn(:,2)=ggdlnn(:,2)-min(ggdlnn(:,2));
ggdlnn(:,2)=ggdlnn(:,2)./max(ggdlnn(:,2));
ggdlnn_leng(ggdlnn(:,2)<(0.2),:)=[];
ggdlnn(ggdlnn(:,2)<(0.2),:)=[];
for bniewazne=1:(size(ggdlnn,1).^2)
if size(ggdlnn,1)>=2
usun_=zeros([1 size(ggdlnn,1)]);
    nr1=ggdlnn(1,1);
    x11=ggtxnn(nr1,:);
    y11=ggtynn(nr1,:);
```

```matlab
        x11(y11==0)=[];
        y11(y11==0)=[];
    for nr_=2:size(ggdlnn,1)
        nr2=ggdlnn(nr_,1);
        x22=ggtxnn(nr2,:);
        y22=ggtynn(nr2,:);
            x22(y22==0)=[];
            y22(y22==0)=[];
        for iy=1:length(x11)
            xbn=1:length(x22);
            xbni=xbn(x22==x11(iy));
            if ~isempty(xbni)
                if y11(iy)<y22(xbni(1))
                    usun_(nr_)=usun_(nr_)+1;
                end
            end
        end
    end
    if sum(usun_)~=0
        ggdlnn(usun_>(ggdlnn_leng'*0.2),:)=[];
        ggdlnn_leng(usun_>(ggdlnn_leng'*0.2))=[];
        ggdlnn=[ggdlnn(2:end,:);ggdlnn(1,:)];
        ggdlnn_leng=[ggdlnn_leng(2:end);ggdlnn_leng(1,:)];
    else
        ggdlnn=[ggdlnn(2:end,:);ggdlnn(1,:)];
        ggdlnn_leng=[ggdlnn_leng(2:end);ggdlnn_leng(1,:)];
    end
end
end
ggdlnn_s=sortrows(ggdlnn,-2);
if size(ggdlnn_s,1)==2
    xNFL1=ggtxnn(ggdlnn_s(1,1),:);
    yNFL1=ggtynn(ggdlnn_s(1,1),:);
    xNFL2=ggtxnn(ggdlnn_s(2,1),:);
    yNFL2=ggtynn(ggdlnn_s(2,1),:);
    xNFL1(xNFL1==0)=[];
    yNFL1(yNFL1==0)=[];
    xNFL2(xNFL2==0)=[];
    yNFL2(yNFL2==0)=[];
    yNFL1_poczg=yNFL1(1)+std(yNFL1);
    yNFL1_poczd=yNFL1(1)-std(yNFL1);
    yNFL2_poczg=yNFL2(1)+std(yNFL2);
    yNFL2_poczd=yNFL2(1)-std(yNFL2);
    if min(xNFL1)<min(xNFL2)
        if (abs(yNFL1(end)-
yNFL2_poczd)<std(yNFL1))|(abs(yNFL1(end)-
yNFL2_poczg)<std(yNFL1));
                xNFL=[xNFL1 xNFL2];
        else
            if length(yNFL1)>length(yNFL2)
```

```
                xNFL=[xNFL1];
            else
                xNFL=[xNFL2];
            end
        end
    else
        if (abs(yNFL2(end)-
yNFL1_poczd)<std(yNFL2))|(abs(yNFL2(end)-
yNFL1_poczg)<std(yNFL2));
                xNFL=[xNFL2 xNFL1];
        else
            if length(yNFL1)>length(yNFL2)
                xNFL=[xNFL1];
            else
                xNFL=[xNFL2];
            end
        end
    end
else
    xNFL=ggtxnn(ggdlnn_s(1,1),:);
    xNFL(xNFL==0)=[];
end
filtr_med=50;
[xNFL,yNFL]=OCT_NFL_line_end(xNFL,xyinfdl_old,xyinfy_old,gr
ad_y_punkt,filtr_med);
przyci_po_obu_x_proc=0.2;
y_dd=abs(diff(yNFL));
y_dd_lab=bwlabel(y_dd<(grad_y_punkt)/2);
num_1=y_dd_lab(round(length(y_dd_lab)*przyci_po_obu_x_proc)
);
num_end=y_dd_lab(round(length(y_dd_lab)*(1-
przyci_po_obu_x_proc)));
x_sek=1:length(y_dd_lab);
x_sek_1=x_sek(y_dd_lab==num_1); x_sek_1=x_sek_1(1);
x_sek_end=x_sek(y_dd_lab==num_end);
x_sek_end=x_sek_end(end);
xNFL=xNFL(x_sek_1:x_sek_end);
yNFL=yNFL(x_sek_1:x_sek_end);
```

and function `OCT_NFL_line_end` intended for filtration of the left and right side of the course:

```
function
[xNFL,yNFL]=OCT_NFL_line_end(xNFL_old,xyinfdl,xyinfy,grad_y
_punkt,filtr_med)
x_start=xNFL_old(round(end/2));
xNFL=[];
yNFL=[];
xyinfy(1,:)=medfilt2(xyinfy(1,:),[1 filtr_med]);
    for hv=x_start:(size(xyinfy,2)-1)
```

```
        if  (min(  abs(xyinfy(1,hv)-xyinfy(:,hv+1))
)<grad_y_punkt)&(xyinfy(1,hv)~=0)
            vff=1:size(xyinfy,1); vff(abs(xyinfy(1,hv)-
xyinfy(:,hv+1))>=grad_y_punkt)=[]; vff=vff(1);
xypam=xyinfy(1,hv);
            vff__=1:size(xyinfy,1); vff__(vff)=[];
            xyinfy(1:end,hv+1) = [xyinfy(vff,hv+1);
xyinfy(vff__,hv+1)];
            xyinfdl(1:end,hv+1)= [xyinfdl(vff,hv+1);
xyinfdl(vff__,hv+1)];
            xyinfy(1:end,hv)=[xyinfy(2:end,hv);0];
            xyinfdl(1:end,hv)=[xyinfdl(2:end,hv);0];
            xNFL=[xNFL;hv];
            yNFL=[yNFL;xypam];
        else
            xyinfy(1:end,hv)=[xyinfy(2:end,hv);0];
            xyinfdl(1:end,hv)=[xyinfdl(2:end,hv);0];
            break
        end
    end
    for hv=(x_start-1):-1:2
        if  (min(  abs(xyinfy(1,hv)-xyinfy(:,hv-1))
)<grad_y_punkt)&(xyinfy(1,hv)~=0)
            vff=1:size(xyinfy,1); vff(abs(xyinfy(1,hv)-
xyinfy(:,hv-1))>=grad_y_punkt)=[]; vff=vff(1);
xypam=xyinfy(1,hv);
            vff__=1:size(xyinfy,1); vff__(vff)=[];
            xyinfy(1:end,hv-1) = [xyinfy(vff,hv-1);
xyinfy(vff__,hv-1)];
            xyinfdl(1:end,hv-1)= [xyinfdl(vff,hv-1);
xyinfdl(vff__,hv-1)];
            xyinfy(1:end,hv)=[xyinfy(2:end,hv);0];
            xyinfdl(1:end,hv)=[xyinfdl(2:end,hv);0];
            xNFL=[hv;xNFL];
            yNFL=[xypam;yNFL];
        else
            xyinfy(1:end,hv)=[xyinfy(2:end,hv);0];
            xyinfdl(1:end,hv)=[xyinfdl(2:end,hv);0];
            break
        end
    end
xNFL=round(xNFL);
yNFL=round(yNFL);
```

Unfortunately, the method described provides the expected results not in all analysed cases. The situation presented in Fig. 4-27 is an example here, fortunately seldom occurring in practice. Such situations occur for actual images if there is a lot of noise on them or if large eye pathologies exist or shadows are strongly visible. Such cases (where even for an OCT

operator it is difficult to answer clearly a question where an individual layer starts and ends) occur pretty seldom in practice.

4.5   Correction of Layers Range

y_{IO}, y_{RPE}, y_{NFL} obtained at earlier stages will be now subject to common analysis to eliminate additional disturbances and to improve their quality. The y_{IO}, y_{RPE}, y_{NFL} courses must fulfil the following conditions resulting from medical premises of eye structure (the conditions will be given in a Cartesian coordinate system):

- $y_{RPE} < y_{IO} < y_{NFL}$ for each x,
- y_{IO} - $y_{RPE} \approx 0.1$ mm – being the initial value starting the operation of modified active contour method,
- y_{NFL} - $y_{IO} \approx$ from 0 to 1 mm, for different x may be even $y_{IO} > y_{NFL}$ or/and $y_{RPE} > y_{NFL}$.

The implementation of this moderately simple correction of layers arrangement we leave to the Reader.

4.6   Final Form of Algorithm

Based on considerations carried out in previous sections the final form of algorithm was formulated in the following form:

```
 [Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255; Lorg=Lgray;
Lmed=medfilt2(Lorg,[5 5]);
Lmed=mat2gray(Lmed);
[xRPE,yRPE,xRPEz,yRPEz]=OCT_global_line(Lmed);
grad_y_punkt=30;
[xNFL,yNFL,xyinfdl,xyinfy,ggtxnn,ggtynn,ggdlnn,xyinfdl_o
ld,xyinfy_old]=OCT_NFL_line(Lmed,grad_y_punkt);
z_gd1=60;
z_gd2=60;
z_sr1=16;
z_sr2=16;
z_kat1=12;
z_kat2=12;
z_us_xy1=12;
z_us_xy2=12;
[yRPEd,ygRPEd]=OCT_activ_cont(Lmed,xRPE,yRPE+50,z_gd1,z_
sr1,z_kat1,z_us_xy1,-1);
```

```
    [yONL,ygONL]=OCT_activ_cont(Lmed,xRPE,yRPE-
50,z_gd2,z_sr2,z_kat2,z_us_xy2,1);
    figure; imshow(Lmed); hold on
    plot(xRPE,yRPE,'-r*','LineWidth',2);
    plot(xRPEz,yRPEz,'-g*','LineWidth',2);
    plot(xNFL,yNFL,'b','LineWidth',2)
    plot(xRPE,yONL,'y','LineWidth',2)
    plot(xRPE,yRPEd,'m','LineWidth',2)
```

Consequently, the following results were obtained - Fig. 4-28 and Fig. 4-29.
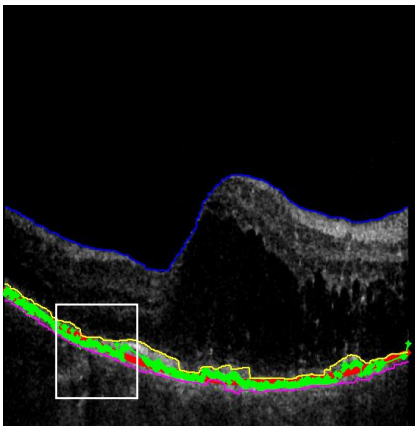
Fig. 4-28 Image L$_{MED}$ with marked in colours layer boundaries y$_{NFL}$, y$_{RPE}$, y$_{ONL}$ and y$_{RPED}$ as the limit of RPE layer analysis
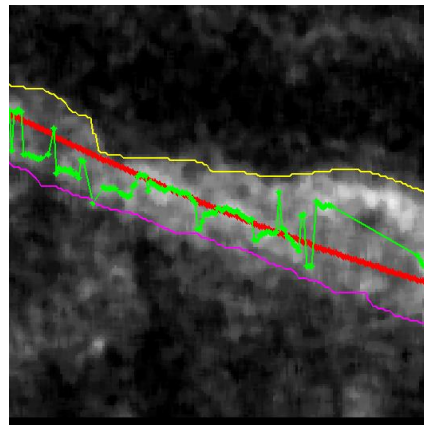
Fig. 4-29 Enlarged image L$_{MED}$ from Fig. 4-28

In the source code presented the following functions have been used, previously presented `OCT_activ_cont` and `OCT_global_line`, which has the following form:

```
function [x,yrpes,dxx,dyy]=OCT_global_line(Lmed)
x=(1:size(Lmed,2))';
yyy=(1:size(Lmed,1))';
yrpe=[];
Lbinrpe=zeros(size(Lmed));
for ik=1:size(Lmed,2)
    xx_best=[];
    Llabp=bwlabel(Lmed(:,ik)>(max(Lmed(:,ik))*0.9));
    Lbinrpe(:,ik)=Llabp;
    for tt=1:max(Llabp)
        xxl=yyy(Llabp==tt);
        xx_best=[xx_best;mean(xxl)] ;
    end
```

```
    if ~isempty(xx_best)
        yrpe(ik)=max(xx_best);
    else
        yrpe(ik)=0;
    end
end
figure; imshow(mat2gray(Lbinrpe*0.5+Lmed));hold on;
plot(yrpe,'r*-')
yg=gradient(yrpe);
ygg=ones([1 length(yrpe)]); ygg(abs(yg)>20)=0;
ygl=bwlabel(ygg);
figure; imshow(mat2gray(Lbinrpe*0.5+Lmed));hold on;
palett=jet(max(ygl));
for iiih=1:max(ygl(:))
        plot(x(ygl==iiih),
yrpe(ygl==iiih),'Color',palett(iiih,:),'LineWidth',4);
end
pam_dl=[];
figure; imshow(mat2gray(Lbinrpe*0.5+Lmed)); hold on
for iiik=1:max(ygl(:))
    for iiikk=iiik:max(ygl(:))
        if iiik<=iiikk
            ygk=[yrpe(ygl==iiik),yrpe(ygl==iiikk)];
            xgk=[x(ygl==iiik);x(ygl==iiikk)];
        else
            ygk=[yrpe(ygl==iiikk),yrpe(ygl==iiik)];
            xgk=[x(ygl==iiikk);x(ygl==iiik)];
        end
        if length(ygk)>10
            P = POLYFIT(xgk',ygk,2); yrpes =
round(POLYVAL(P,x));
            plot(yrpes,'g*-')
            pam_dl=[pam_dl;[iiik iiikk sum(abs(yrpe-
yrpes')<20)]];
        end
    end
end
pam_s=sortrows(pam_dl,-3);
if size(pam_s,1)==1
        ygk=[yrpe(ygl==pam_s(1,1))];
        xgk=[x(ygl==pam_s(1,1))];
else
        ygk=[yrpe(ygl==pam_s(1,1)),yrpe(ygl==pam_s(1,2))];
        xgk=[x(ygl==pam_s(1,1));x(ygl==pam_s(1,2))];
end
    P = POLYFIT(xgk',ygk,2); yrpes = round(POLYVAL(P,x));
    plot(x,yrpes,'w*-');
yrpe=yrpe(:);
    plot(x,yrpe,'m*-');
dx=x; dx(abs(yrpe-yrpes)>20)=[];
```

```
yrpe(abs(yrpe-yrpes)>20)=[];
dxl=bwlabel(diff(dx)<125);
pdxl=[];
for qw=1:max(dxl)
   pdxl=[pdxl;[qw, sum(dxl==qw)]];
end
pdxl(pdxl(:,2)<50,:)=[];
dxx=[]; dyy=[];
for wq=1:size(pdxl,1)
    dxx=[dxx; dx(dxl==pdxl(wq,1))];
    dyy=[dyy; yrpe(dxl==pdxl(wq,1))];
end
```

The result presented is affected mainly by the arguments of `OCT_activ_cont` function, which in accordance with the description quoted determine the type of layer recognised.

The algorithm presented makes a uniform whole related to the analysis of layers within the fundus of the eye on flat OCT images. The results obtained may be enhanced by an automated analysis of 'holes' on the image – presented below.

4.7   Determination of 'Holes' on the Image

To determine holes on the image a method of binary image L_{BIN_IP} labelling was applied (11) obtaining image L_{ET} shown in Fig. 4-30.
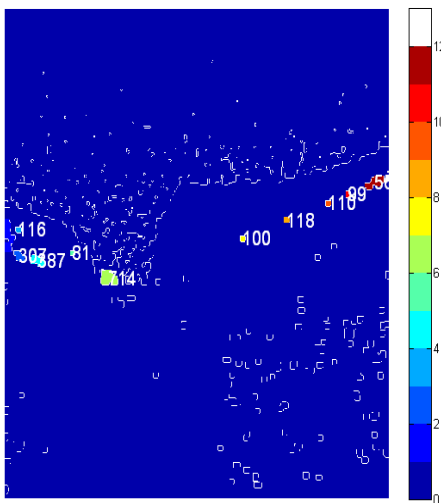
k_o	m_o	n_o	P_o
1	28	420	307
2	26	375	116
3	64	428	387
4	131	415	81
5	201	459	714
6	460	390	100
7	546	359	118
8	626	330	110
9	664	315	99
10	717	295	560
11	740	287	70

Fig. 4-30 L_{ET} image

Fig. 4-31 Table of results obtained for consecutive clusters k_o (position of the centre of gravity (m_o,n_o) and area of surface P_o)

Examples of results obtained are shown in the specification in Fig. 4-31. Each object (cluster) k_o received a label and determined coordinates (m_o, n_o) of its centre of gravity position. In addition, the area of surface P_o is also calculated. The source code is provided below:

```
[Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255; Lorg=Lgray;
Lmed=medfilt2(Lorg,[5 5]);
Lmed=mat2gray(Lmed);
[xRPE,yRPE,xRPEz,yRPEz]=OCT_global_line(Lmed);
L11=filter2(ones(3),Lmed)/(3*3);
L12=imregionalmin(L11);
L13=~imopen(L12,ones(9));
[Lbin,L18]=OCT_areaa(L13,xRPE,yRPE);
Let=bwlabel(Lbin);
Let_=Let;
Let_(edge(double(L13))==1)=max(Let(:))+1;
figure; imshow(Let_,[]); pall=jet(max(Let(:)));
colormap([pall; [1 1 1]]); colorbar; hold on
[XX,YY]=meshgrid(1:size(Let,2),1:size(Let,1));
kmnp=[];
for ju=2:max(Let(:))
    Let4=Let==ju;
    Letx=Let4.*XX; Letx(Letx==0)=[];
    Lety=Let4.*YY; Lety(Lety==0)=[];

text(median(Letx),median(Lety),mat2str(sum(Let4(:))),'FontS
ize',15,'Color',[1 1 1])
    kmnp=[kmnp;[ju
median(Letx),median(Lety),sum(Let4(:))]];
end
kmnp
```

For diagnostic reasons the position of clusters analysed (given in Fig. 4-30) was narrowed to those, which position of the centre of gravity falls within the range between y_{RPE} and y_{NFL}.

4.8 Assessment of Results Obtained Using the Algorithm Proposed

An example of algorithm implementation intended for analysis of layers occurring on an OCT image has been presented. This methodology has been applied to the analysis of around 500 cases, where during verification it has erroneously determined layers for 5% of images.

Examples of properly and improperly recognised layers are shown in Fig. 4-32 and Fig. 4-33.
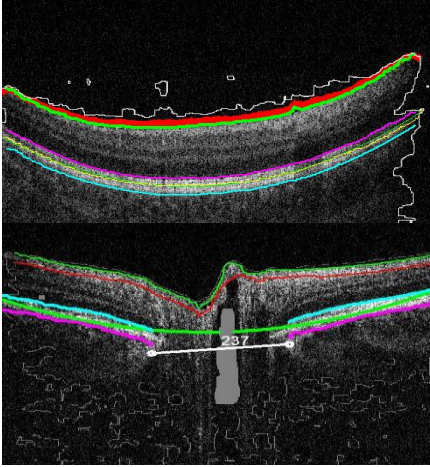

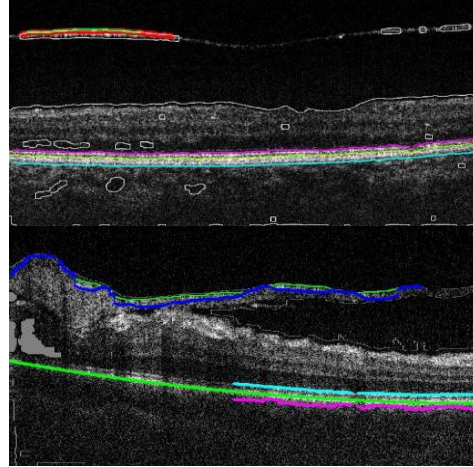**Fig. 4-32 Examples of OCT resultant images with marked properly recognised y_{RPE}, y_{IO}, y_{NFL}**

Fig. 4-33 Examples of OCT resultant images with marked improperly recognised y_{RPE}, y_{IO}, y_{NFL}

The algorithm proposed was implemented in the Matlab environment and operates at a rate of one image per 15s for a P4 CPU 3GHz processor, 2GB RAM. Additionally, an application in language C was developed, which after time optimisation on the same computer analyses the same image within 0.85s.

The Reader implementing the above function must notice delays introduced by the graphic card during image displaying. In particular, the resultant images are the point here, for which results were presented in the form of graphs or points on a flat image in greyness levels.

4.9 Layers Recognition on a Tomographic Eye Image Based on Random Contour Analysis

4.9.1 Determination of Direction Field Image

Like in [25] and [40] the input image L_{GRAY} is initially subject to filtration using a median filter of (M_hxN_h) size of h=3x3 mask. The obtained image L_M is subject to the analysis presented in the next sections.

The first stage of the edge detection method used [14], [35], [41] consists of making a convolution of input image L_M of M_MxN_M resolution, i.e.

$$L_{GX}(m,n) = \sum_{m_h=-M_h/2}^{M_h/2} \sum_{n_h=-N_h/2}^{N_h/2} L_M(m+m_h, n+n_h) \cdot h_x(m_h, n_h) \qquad (24)$$

$$L_{GY}(m,n) = \sum_{m_h=-M_h/2}^{M_h/2} \sum_{n_h=-N_h/2}^{N_h/2} L_M(m+m_h, n+n_h) \cdot h_y(m_h, n_h) \qquad (25)$$

with Gauss filters masks, e.g. of 3x3 size [14], [35], [41]. Based on that the matrix of gradient in both directions, necessary to determine the edges, has been determined in accordance with a classical dependence:

$$L_{GXY}(m,n) = \sqrt{L_{GX}(m,n)^2 + L_{GY}(m,n)^2} \qquad (26)$$

And in particular its normalised form, i.e.:

$$L_G(m,n) = \frac{L_{GXY}(m,n)}{\max\limits_{m,n}(L_{GXY}(m,n))} \qquad (27)$$

The image of Lα direction field has been determined for each pair of pixels $L_{GX}(m,n)$ and $L_{GY}(m,n)$, and in general L_{GX} and L_{GY} images, i.e.:

$$L\alpha(m,n) = \operatorname{atan}\left(\frac{L_{GY}(m,n)}{L_{GX}(m,n)}\right) \qquad (28)$$

The implementation of the above relationships in Matlab looks as follows:

```
Lm=zeros(100); Lm(10:30,10:20)=1; Lm(40:80,50:70)=1;
Lm=imnoise(Lm,'gaussian',0.2);
Lm=medfilt2(Lm,[3 3]);
Lm=mat2gray(Lm);
figure; imshow(Lm,'notruesize')
    Nx1=5;
    Sigmax1=24;
    Nx2=5;
    Sigmax2=24;
    Theta1=pi/2;
    Ny1=5;
    Sigmay1=24;
    Ny2=5;
    Sigmay2=24;
    Theta2=0;
    alfa=0.15;
```

```
hx=OCT_NOISE_gauss(Nx1,Sigmax1,Nx2,Sigmax2,Theta1);
Lgx= conv2(Lm,hx,'same');
hy=OCT_NOISE_gauss(Ny1,Sigmay1,Ny2,Sigmay2,Theta2);
Lgy=conv2(Lm,hy,'same');
Lalp=atan2(Lgy,Lgx);
Lalp=Lalp*180/pi;
Lg=mat2gray(abs(Lgx)+abs(Lgy));
figure; imshow(Lg,[],'notruesize'); colormap('jet');
colorbar
figure; imshow(Lalp,[],'notruesize'); colormap('jet');
colorbar
```

where `OCT_NOISE_gauss`

```
function h = OCT_NOISE_gauss(n1,sigma1,n2,sigma2,theta)
r=[cos(theta) -sin(theta);
   sin(theta)  cos(theta)];
for i = 1 : n2
    for j = 1 : n1
        u = r * [j-(n1+1)/2 i-(n2+1)/2]';
        h(i,j) = gauss(u(1),sigma1)*OCT_gauss(u(2),sigma2);
    end
end
h = h / sqrt(sum(sum(abs(h).*abs(h))));

function y = OCT_gauss(x,std)
y = -x * gauss(x,std) / std^2;

function y = gauss(x,std)
y = exp(-x.^2/(2*std^2)) / (std*sqrt(2*pi));
```

As a result the images presented below are obtained.
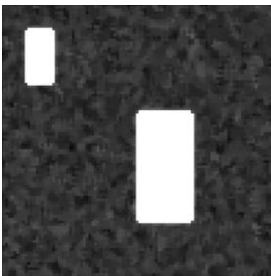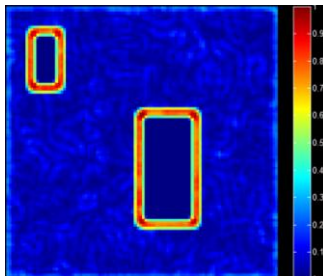

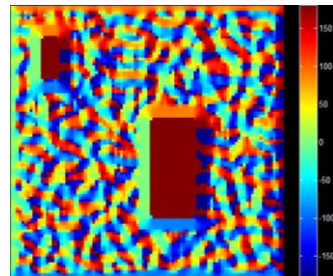| **Fig. 4-34 Artificial Lm image** | **Fig. 4-35 Artificial L_G image** | **Fig. 4-36 Artificial Lα image** |

Those images, Lα and L_G, are further used in the analysis, where the starting points random selection is the next step.

4.9.2   Starting Points Random Selection and Correction

Starting points, and − based on them − the next ones will be used in consecutive stages of algorithm operation to determine parts of layers contours. The initial position of starting points was determined at random. Random values were obtained from uniform range (0,1) for each point of image matrix L_o with image resolution L_M, i.e.: MxN. For a created this way (random) image L_o a decimal to binary converion is carried out with threshold p_r, which is the first and one of matched (described later) parameters of the algorithm, the obtained binary matrix L_u is described by the relationship:

$$L_u(m,n) = \begin{cases} 1 & \text{dla} \quad L_G(m,n) > L_M(m,n) \cdot p_r \\ 0 & \text{dla} \qquad\qquad \text{other} \end{cases} \tag{29}$$

In this case:

```
figure; imshow(Lg,[],'notruesize'); hold on
pr=0.3;
Lrand=rand(size(Lg));
[n,m]=meshgrid(1:size(Lrand,2),1:size(Lrand,1));
n(Lrand>(Lg*pr))=[];
m(Lrand>(Lg*pr))=[];
plot(n,m,'r.');
```

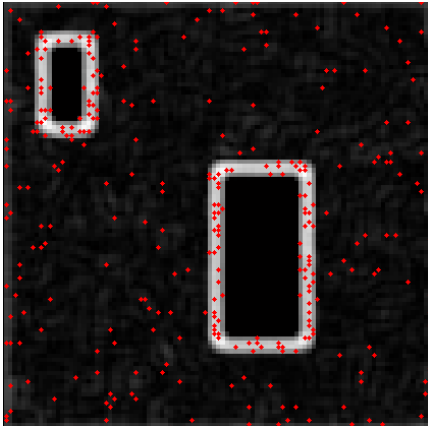The result obtained is presented in the following figure (Fig. 4-37).

Fig. 4-37 Image L_G with marked random selected points
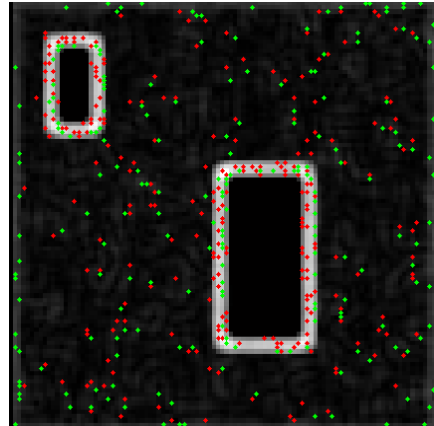
Fig. 4-38 Image L_G with random selected points marked red and their correction marked green

Starting points $o^*_{i,j}$ (where index 'i' marks the next starting point, while 'j' subsequent points created on its basis) satisfy condition $L_u(m,n)=1$ – that is starting points are $o^*_{i,1}$. This way the selection of the threshold value p_r within the range $(0,1)$ influences the number of starting points, which number is the larger, the brighter is the grey level (contour) in the L_G image. In the next stage the starting points' position is modified in the set area H of $M_H x N_H$ size. Modification consists in the correction of points $o^*_{i,1}$ position of coordinates $(m^*_{i,1}, n^*_{i,1})$ to new coordinates $(m_{i,1}, n_{i,1})$, where shifts within the range $m_{i,1}= m^*_{i,1}\pm(M_H)/2$ and $n_{i,1}= n^*_{i,1}\pm(N_H)/2$ are possible. A change of coordinates occurs in the area of $\pm(M_H)/2$ and $\pm(N_H)/2$, in which the highest value is achieved $L_G(m^*_{i,1}\pm(M_H)/2, n^*_{i,1}\pm(N_H)/2)$, i.e.:

$$L_G\left(m_{i,1},n_{i,1}\right)= \max_{m^*_{i,1}\pm\frac{M_H}{2},n^*_{i,1}\pm\frac{N_H}{2}}\left(L_G\left(m^*_{i,1} \pm \frac{M_H}{2}, n^*_{i,1} \pm \frac{N_H}{2} \right)\right) \qquad (30)$$

Then the correction of repeating points is carried out – points of the same coordinates are removed. The source code looks here as follows:

```
H=ones(5);
[n,m]=OCT_NOISE_area(n,m,Lg,H);
plot(n,m,'g.'); hold on
```

where `OCT_NOISE_area`

```
function [n,m]=OCT_NOISE_area(n,m,Lg,H)
xn=[];
yn=[];
[xr,yr]=meshgrid(1:size(H,2),1:size(H,1));
for iw=1:length(n)
    ddx=size(H,2)/2;
    ddy=size(H,1)/2;
    xp=round(n(iw)-ddx); xk=round(n(iw)+ddx-1);
    yp=round(m(iw)-ddy); yk=round(m(iw)+ddy-1);

    if (xp<1)|(yp<1)|(xk>size(Lg,2))|(yk>size(Lg,1))
        xn(iw)=n(iw);
        yn(iw)=m(iw);
    else
        Lff=Lg(yp:yk,xp:xk);
        xr_=xr; yr_=yr;
        xr_(Lff~=max(max(Lff)))=[];
        yr_(Lff~=max(max(Lff)))=[];
        xn(iw)=n(iw)+xr_(1)-ddx;
        yn(iw)=m(iw)+yr_(1)-ddy;
    end
```

```
end
n=round(xn); m=round(yn);
n(n<=0)=1; m(m<=0)=1;
n(n>size(Lg,2))=size(Lg,2);
m(m>size(Lg,1))=size(Lg,1);
```

The obtained results are presented in Fig. 4-38.

4.9.3   Iterative Determination of Contour Components

To determine layers on an OCT image, contour components have been determined in the sense of its parts subject to later modification and processing in the following way. For each random selected point $o^*_{i,1}$ of coordinates $(m^*_{i,1}, n^*_{i,1})$ and then modified (in the sense of its position) to $o_{i,1}$ of coordinates $(m_{i,1}, n_{i,1})$ an iterative process is carried out consisting in looking for consecutive points $o_{i,2}$ $o_{i,3}$ $o_{i,4}$ $o_{i,5}$ etc. and local modification of their position (described in the previous section) starting from $o_{i,1}$ in accordance with the relationship:

$$\begin{cases} m^*_{i,j+1} = m_{i,j} + A_{i,j} \cdot \sin\big(L_\alpha\big(m_{i,j}, n_{i,j}\big)\big) \\ n^*_{i,j+1} = n_{i,j} + A_{i,j} \cdot \cos\big(L_\alpha\big(m_{i,j}, n_{i,j}\big)\big) \end{cases} \tag{31}$$

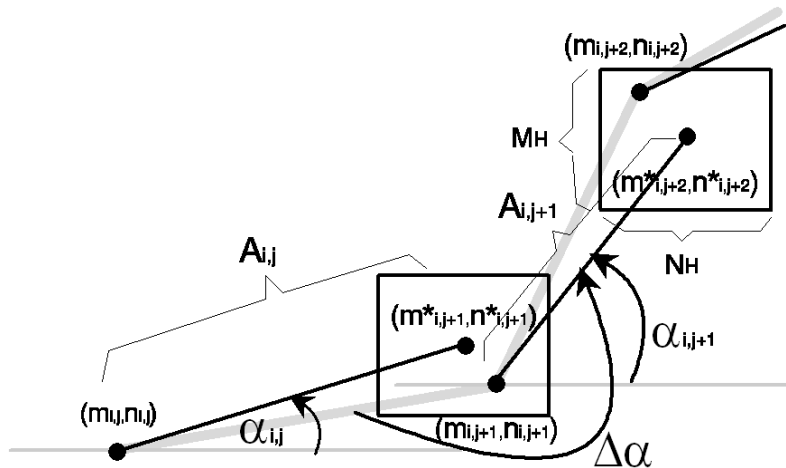A demonstrative illustration of the iterative process is shown in Fig. 4-39.

Fig. 4-39 Demonstrative diagram of iterative process of contour components determination

In the case of described iterative process of contour components determination it is necessary to introduce a number of limitations (next parameters), comprising:

- j_{MAX} – maximum iterations number – limitation aimed at elimination of algorithm looping if each time points $o_{i,j}$ of different position are determined and the contour will have the shape of e.g. a spiral.

- Stopping the iterative process, if it is detected that $m_{i,j} = m_{i,j+1}$ and $n_{i,j} = n_{i,j+1}$. Such situation happens most often if $A_{i,j}$ is close to or higher than M_H or N_H. Like in the case of starting points random selection and correction, also here a situation may occur that after the correction $m_{i,j} = m_{i,j+1}$ and $n_{i,j} = n_{i,j+1}$.

Stopping the iterative process if $m_{i,j} > M_M$ or $n_{i,j} > N_M$ that is in the cases, when indicated point $o_{i,j}$ will be outside the image.

Stopping the iterative process if $|L_\alpha(m_{i,j}, n_{i,j}) - L_\alpha(m_{i,j+1}, n_{i,j+1})| > \Delta\alpha$ where $\Delta\alpha$ is the next parameter set for acceptable contour curvature.

At this stage consecutive contour components for set parameters are obtained. These parameters comprise:

- mask size h_x and h_y ((24), (25)) closely related to the image resolution and to the size of areas identified, adopted for $M_M \times N_M = 864 \times 1024$ on $M_H \times N_H = 23 \times 23$,

- p_r – threshold responsible for the number of starting points (29) – changed practically within the range 0-0.1,

- j_{MAX} – the maximum acceptable iterations number – set arbitrarily at 100,

- $\Delta\alpha$ - angle range set within the range 10-70°,

- $M_H \times N_H$ – size of the correction area, a square area, changed within the range from $M_H \times N_H = 5 \times 5$ to $M_H \times N_H = 25 \times 25$,

- A_{ij} – amplitude, constant for individual i,j, set at $A_{i,j} = M_H$,

- $\Delta\alpha$ - acceptable maximum change of angle between consecutive contour points, set within the range 10-70°.

For the artificial image presented in Fig. 4-40 an iterative process of contours determination has been performed, assuming $p_r = 0.1$, $\Delta\alpha = 45^\circ$, $M_H \times N_H = 5 \times 5$. The results obtained are presented in Fig. 4-40.
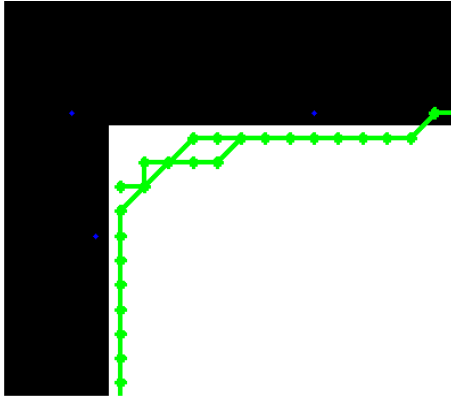
Fig. 4-40 Artificial input image with marked contour components
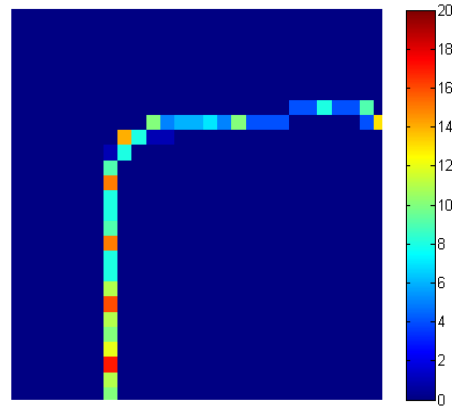
Fig. 4-41 Artificial input image with marked overlapping contour components – the number of overlapping points of the same coordinates is shown in pseudocolours

The source code of the iterative process of contour components determination is presented below:

```
Lz=zeros(size(Lalp));
Lz2=zeros(size(Lalp));
A=5;
delta_alph=50;
n_1=[]; m_1=[];
al_1=[];
for i=1:length(n)
    ns_=[];
    ms_=[];
    ks_=[];
    ns_(1)=[n(i)];
    ms_(1)=[m(i)];
    ii=1;
    alp_1=Lalp(ms_(ii),ns_(ii));
    al_1(i,1)=[alp_1];
    kat_r=0;
    while kat_r<delta_alph
        alp_1=Lalp(ms_(end),ns_(end));
        n_p1=round(ns_(end)+A*cos((alp_1+90)*pi/180));
        m_p1=round(ms_(end)+A*sin((alp_1+90)*pi/180));
        if
(n_p1<1)|(m_p1<1)|(n_p1>size(Lalp,2))|(m_p1>size(Lalp,1))
            break
        end
```

```
        [n_pp1,m_pp1]=OCT_NOISE_area(n_p1,m_p1,Lg,H);
        if
sum(sum([round(m_pp1)==ms_',round(n_pp1)==ns_'],2)==2)>1
            disp('zabezpiecz')
            break
        end
        if ii>100
            [i, ii]
            break
        end
        ii=ii+1;

[nss,mss]=OCT_NOISE_line([ns_(end),n_pp1],[ms_(end),m_pp1])
;
        ns_=[ns_;round(nss')];
        ms_=[ms_;round(mss')];
        ks_(ii)=alp_1;
        kat_r=abs(alp_1-Lalp(ms_(end),ns_(end))); if
kat_r>180; kat_r=180-kat_r; end
    end
        n_1(i,1:length(ns_))=ns_;
        m_1(i,1:length(ms_))=ms_;
        al_1(i,1:length(ks_))=ks_;
        for im=1:length(ns_)
            Lz(ms_(im),ns_(im))=Lz(ms_(im),ns_(im))+1;
        end
    plot(ns_,ms_,'g-*','LineWidth',3)
    pause(0.0000001)
end
figure; imshow(Lz,[],'notruesize'); colormap('jet');
colorbar
```

where `OCT_NOISE_line` is a function intended for generation of discrete points on the section connecting the points given, i.e.:

```
function [n_,m_]=OCT_NOISE_line(n,m)
if (abs(n(1)-n(2))==0)&(abs(m(1)-m(2))==0)
    n_=n; m_=m;
else
    if abs(n(1)-n(2))<abs(m(1)-m(2))
        if m(1)<m(2)
            m_=m(1):m(2);
        else
            m_=m(1):-1:m(2);
        end
        if n(1)~=n(2)
            n_=n(1):((n(2)-n(1))/(length(m_)-1)):n(2);
        else
            n_=ones(size(m_))*n(1);
        end
```

```
    else
        if n(1)<n(2)
            n_=n(1):n(2);
        else
            n_=n(1):-1:n(2);
        end
        if m(1)~=m(2)
            m_=m(1):((m(2)-m(1))/(length(n_)-1)):m(2);
        else
            m_=ones(size(n_))*m(1);
        end
    end
end
```

When analysing results presented in Fig. 4-40 it should be noticed that the iterative process is stopped only when $m_{i,j} = m_{i,j+1}$ and $n_{i,j} = n_{i,j+1}$ (as mentioned before). That is only if points $o_{i,j}$ and $o_{i,j+1}$ have the same position. Instead, this condition does not apply to points $o_{i,j}$ which have the same coordinates but for different 'i' that is originated at specific iteration point from various starting points. Easing of this condition leads to origination of overlapping contour components (Fig. 4-41) which will be analysed in the next sections.

4.9.4   Determination of Contours from Their Components

As presented in Fig. 4-41 in the previous section, the iterative process carried out may lead to overlapping of points $o_{i,j}$ of the same coordinates originated from various starting points $(m_{i,j}, n_{i,j})$. This property is used for final determination of layers contour on an OCT image. In the first stage the image L_z from Fig. 4-41, is subject to decimal to binary conversion, i.e. the image that originated as follows:

$$L_{Z,j}(m,n) = \begin{cases} 1 & \text{jezeli} \quad m = m_{i,j} \wedge n = n_{i,j} \\ 0 & \text{other} \end{cases} \qquad (32)$$

For j=1,2,3,… and finally $L_z(m,n)$:

$$L_Z(m,n) = \sum_j L_{Z,j}(m,n) \qquad (33)$$

$$L_{ZB}(m,n) = L_Z(m,n) > p_b \qquad (34)$$

Where L_{ZB} is a binary image originated from decimal to binary conversion of image L_z with threshold p_b. The selection of threshold p_b is a key element for further analysis and correction of the contour generated. In a general case a situation may occur, where despite relatively low value p_r of threshold assumed a selected starting point $o_{i,1}$ is situated outside the object's edge. Then the next iterations may 'connect' it (in consecutive processes (32), (33), with the remaining part. In such case the process of protruding branches removing should be carried out – like branch cutting in skeletonisation. In this case the situation is a bit easier – there are two possibilities of this process implementation: increasing the threshold value p_b or considering the brightness value $L_G(m_{i,j}, n_{i,j})$ - Fig. 4-42.
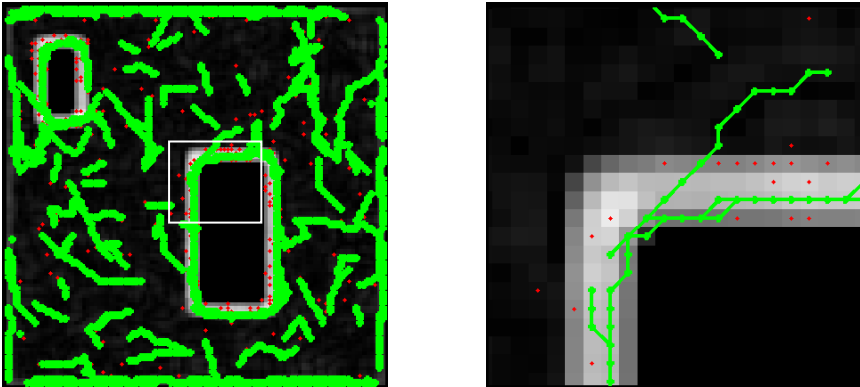

Fig. 4-42 Artificial input image including an enlargement of example area with contour components marked in green, and preliminary random selected points – in red

4.9.5   Setting the Threshold of Contour Components Sum Image

The selection of threshold p_b during image L_{ZB} receiving on the one hand for high values leads to obtaining those contour components, for which the largest number of points overlapped at various 'i' of $o_{i,j}$ points (Fig. 4-43, Fig. 4-44). On the other hand contour discontinuities may occur. Therefore the second mentioned method of obtaining the final form of contour, which consists of considering values $L_G(m_{i,j}, n_{i,j})$ for $L_z(m_{i,j}, n_{i,j}) = 1$ and higher, was selected.
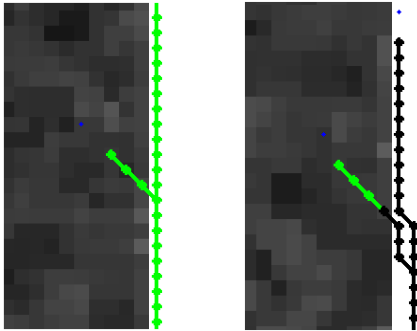
Fig. 4-43 Protruding contour branch (green) as an artefact occurring for the method described particularly visible for noise-affected images and results of removing the protruding branches (black)
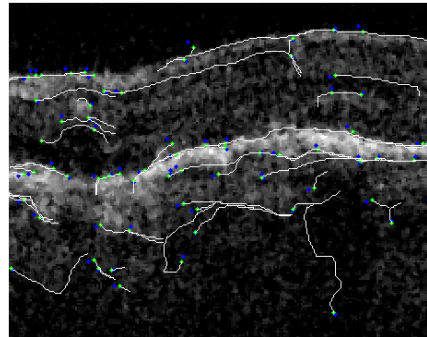
Fig. 4-44 Protruding contour branch as an artefact occurring for the method described in a real OCT image

Assuming that two non-overlapping points $o_{1,j}$ and $o_{2,j}$ have been random selected, such that $m_{1,j} \neq m_{2,j}$ or $n_{1,j} \neq n_{2,j}$, $L_M(m_{1,j}, n_{1,j})$ and $L_M(m_{2,j}, n_{2,j})$ values were determined for consecutive j – Fig. 4-45.

Fig. 4-45 Graph of $L_M(m_{1,j}, n_{1,j})$ – red and $L_M(m_{2,j}, n_{2,j})$ – green values changes for consecutive points j
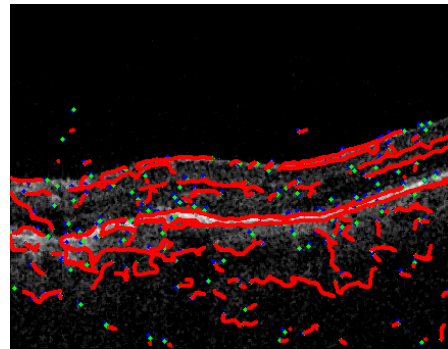
Fig. 4-46 Example of results obtained for a real OCT image for p_r=0.02, $\Delta\alpha$=80°, M_HxN_H=35x35, p_b=2, p_j=0.8

Then a maximum value was determined for each sequence of $o_{i,j}$ points:

$$O_m(i) = \max_j \left(L_M\left(m_{i,j}, n_{i,j} \right) \right)$$

(35)

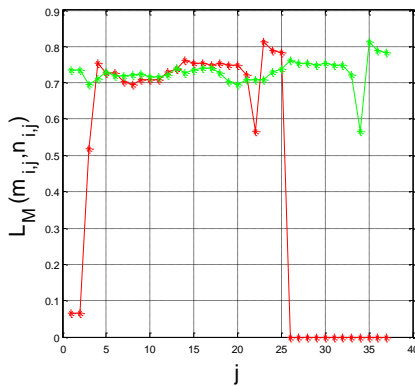Then all $o_{i,j}$ points were removed, which satisfied the condition $o_{i,j} < (O_m(i) \cdot p_j)$, where p_j is the threshold (precisely the percentage value of $O_m(i)$ below which all points are removed). To prevent introduction of discontinuities, only points at the beginning of the component contour are removed. The value was arbitrarily set to $p_j = 0.8$. The obtained results are shown in Fig. 4-43 and Fig. 4-46. Example results shown in Fig. 4-46 are obtained for a real OCT image for $p_r = 0.02$, $\Delta = 80°$, $M_H \times N_H = 35\times35$, $p_b = 2$, $p_j = 0.8$. Correctly determined contour components and other contour fragments, which because of the form of relationship (34) and limitation for $O_m(i)$ have not been removed, are visible. However, on the other hand the number and form of parameters available allows pretty high freedom in such their selection as to obtain the expected results. The final form of algorithm was formulated on this basis.

```
[Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255; Lorg=Lgray;
L=imresize(Lgray,0.5);
Lm=medfilt2(L,[3 3]);
Lm=mat2gray(Lm);
figure; imshow(Lm)
     Nx1=5;
     Sigmax1=24;
     Nx2=5;
     Sigmax2=24;
     Theta1=pi/2;
     Ny1=5;
     Sigmay1=24;
     Ny2=5;
     Sigmay2=24;
     Theta2=0;
     alfa=0.15;
hx=OCT_NOISE_gauss(Nx1,Sigmax1,Nx2,Sigmax2,Theta1);
Lgx= conv2(Lm,hx,'same');
hy=OCT_NOISE_gauss(Ny1,Sigmay1,Ny2,Sigmay2,Theta2);
Lgy=conv2(Lm,hy,'same');
Lalp=atan2(Lgy,Lgx);
Lalp=Lalp*180/pi;
Lg=mat2gray(abs(Lgx)+abs(Lgy));
figure; imshow(Lg,[],'notruesize'); colormap('jet');
colorbar
figure; imshow(Lalp,[],'notruesize'); colormap('jet');
colorbar
figure; imshow(Lg,[],'notruesize'); hold on
pr=0.05;
Lrand=rand(size(Lg));
```

```
[n,m]=meshgrid(1:size(Lrand,2),1:size(Lrand,1));
n(Lrand>(Lg*pr))=[];
m(Lrand>(Lg*pr))=[];
plot(n,m,'r.');
H=ones(5);
[n,m]=OCT_NOISE_area(n,m,Lg,H);
plot(n,m,'g.'); hold on
Lz=zeros(size(Lalp));
A=5;
delta_alph=50;
n_1=[]; m_1=[];
al_1=[];
for i=1:length(n)
    ns_=[];
    ms_=[];
    ks_=[];
    nma_=[];
    ns_(1)=[n(i)];
    ms_(1)=[m(i)];
    ii=1;
    alp_1=Lalp(ms_(ii),ns_(ii));
    al_1(i,1)=[alp_1];
    kat_r=0;
    while kat_r<delta_alph
        alp_1=Lalp(ms_(end),ns_(end));
        n_p1=round(ns_(end)+A*cos((alp_1+90)*pi/180));
        m_p1=round(ms_(end)+A*sin((alp_1+90)*pi/180));
        if
(n_p1<1)|(m_p1<1)|(n_p1>size(Lalp,2))|(m_p1>size(Lalp,1))
            break
        end
        [n_pp1,m_pp1]=OCT_NOISE_area(n_p1,m_p1,Lg,H);
        if
sum(sum([round(m_pp1)==ms_',round(n_pp1)==ns_'],2)==2)>1
            disp('zabezpiecz')
            break
        end
        if ii>100
            [i, ii]
            break
        end
        ii=ii+1;
        [nss,mss]=line_([ns_(end),n_pp1],[ms_(end),m_pp1]);
        ns_=[ns_;round(nss')];
        ms_=[ms_;round(mss')];
        ks_(ii)=alp_1;
        kat_r=abs(alp_1-Lalp(ms_(end),ns_(end))); if
kat_r>180; kat_r=180-kat_r; end
    end
        n_1(i,1:length(ns_))=ns_;
```

```
        m_1(i,1:length(ms_))=ms_;
        al_1(i,1:length(ks_))=ks_;
        for im=1:length(ns_)
            Lz(ms_(im),ns_(im))=Lz(ms_(im),ns_(im))+1;
            nma_(im)=Lg(ms_(im),ns_(im));
        end
    ns_s=ns_; ms_s=ms_; m_nma_=max(nma_(:));
    for bg=1:length(nma_)
        if nma_(bg)<(m_nma_*0.8)
            ns_s(1)=[];ms_s(1)=[];
        else
            break
        end
    end
    plot(ns_s,ms_s,'r','LineWidth',3)
pause(0.0000001)
end
```

In most cases the obtaining of intended contour shape is possible for one fixed $M_H \times N_H$ value. However, it may turn out necessary to use a hierarchical approach, for which the $M_H \times N_H$ size will be reduced, thanks to which a higher precision of the proposed method will be obtained and the weight (hierarchy) of individual contours importance will be introduced. Examples of results obtained for the algorithm given ultimately in this form are as follows.
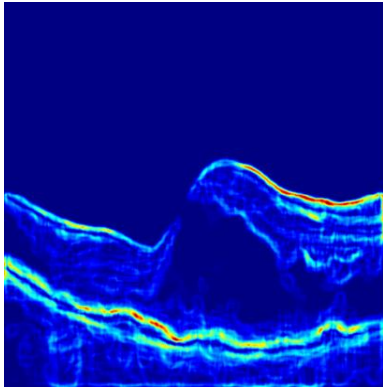
Fig. 4-47 Image L_G   **Fig. 4-48 Image L_α**

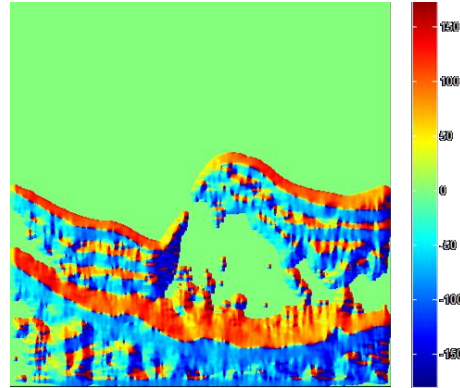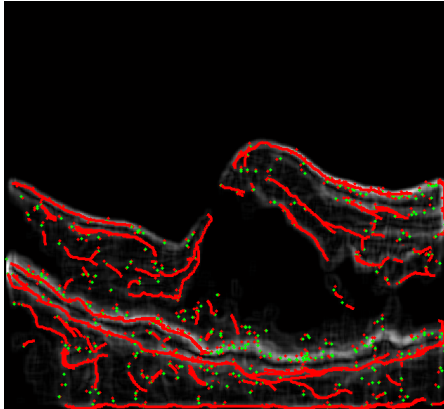**Fig. 4-49 Image L$_z$ with determined contours marked red**
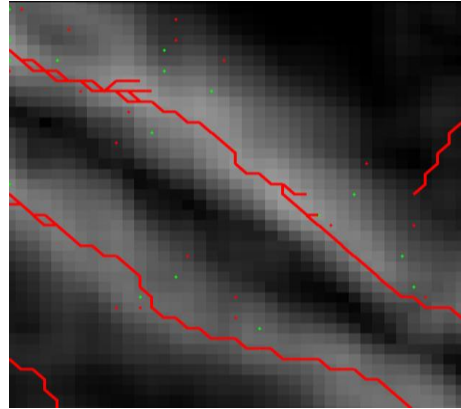

Fig. 4-50 Enlarged fragment of L$_z$ image

4.9.6   Properties of the Algorithm Proposed

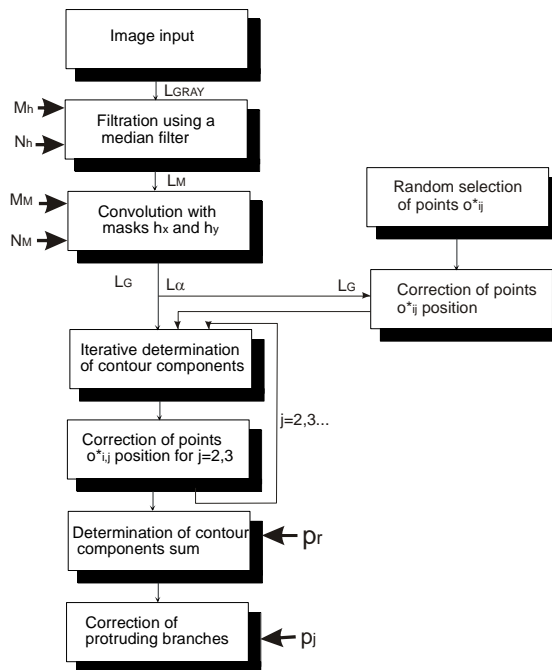The algorithm created is presented in a block diagram – Fig. 4-51.

Fig. 4-51 Block diagram of proposed contour detection algorithm (and hence layers on an OCT eye image)

The assessment of proposed algorithm properties (Fig. 4-51) was carried out evaluating error δ in contour determination for changing parameters p_r, $\Delta\alpha$, $M_H x N_H$, p_b, p_j within the range $p_r \in (0,0.1)$, $\Delta\alpha$, $M_H x N_H \in (3,35)$ p_b, p_j. An artificial image of rectangular object located centrally in the scene (Fig. 4-52) has been used in the assessment.
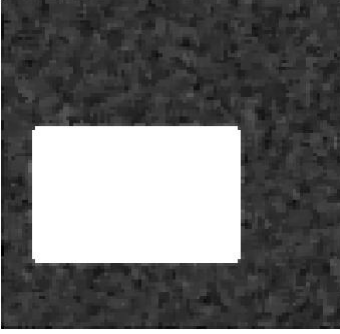
Fig. 4-52 Artificial input image used for error assessment
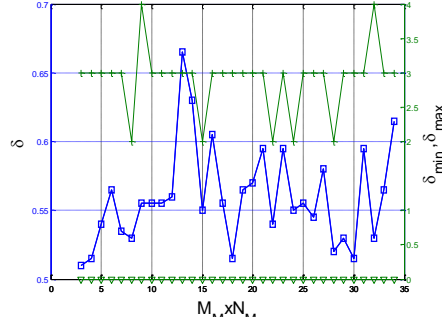
Fig. 4-53 Graph of error δ values changes and its minimum δ_{min} and maximum δ_{max} value vs. $M_M x N_M$

Instead, the error was defined as follows:

$$\delta = \frac{1}{j} \cdot \sum_j \left(\left| m_{i,j} - m_{w,j} \right| + \left| n_{i,j} - n_{w,j} \right| \right) \tag{36}$$

$$\delta_{min} = \min_j \left(\left| m_{i,j} - m_{w,j} \right| + \left| n_{i,j} - n_{w,j} \right| \right) \tag{37}$$

$$\delta_{max} = \max_j \left(\left| m_{i,j} - m_{w,j} \right| + \left| n_{i,j} - n_{w,j} \right| \right) \tag{38}$$

assuming that only one point, i.e. i=1, was random selected. The second part of the assessment consists of points of discontinuity against the standard contour.

Fig. 4-53 shows the graph of error δ values changes and its minimum δ_{min} and maximum δ_{max} value vs. $M_M x N_M$ changing between 3 and 35. The algorithm intended for properties analysis comprises the already presented source code (as a fundamental part) supplemented with fragments related to the specific nature of the object (Fig. 4-42) and measurements of its properties.

```
MN_w=[];
```

```matlab
for MN=3:34
L1=zeros(100);
L1(40:80,10:70)=1;
[xw,yw]=meshgrid(1:size(L1,2),1:size(L1,1));
L111=xor(L1,imerode(L1,ones(3)));
xw(L111==0)=[];
yw(L111==0)=[];
L2=imnoise(L1,'gaussian',0.2);
L3=medfilt2(L2,[3 3]);
L4=mat2gray(L3);
    Nx1=8;
    Sigmax1=MN;
    Nx2=8;
    Sigmax2=MN;
    Theta1=pi/2;
    Ny1=8;
    Sigmay1=MN;
    Ny2=8;
    Sigmay2=MN;
    Theta2=0;
    alfa=0.15;
hx=OCT_NOISE_gauss(Nx1,Sigmax1,Nx2,Sigmax2,Theta1);
Lgx= conv2(L4,hx,'same');
hy=OCT_NOISE_gauss(Ny1,Sigmay1,Ny2,Sigmay2,Theta2);
Lgy=conv2(L4,hy,'same');
alp=atan2(Lgy,Lgx);
Lalp=alp*180/pi;
Lg=mat2gray(abs(Lgx)+abs(Lgy));
figure; imshow(L4,'notruesize');
hold on
Lrand=rand(size(Lg));
[n,m]=meshgrid(1:size(Lrand,2),1:size(Lrand,1));
n(Lrand>(Lg*0.02))=[];
m(Lrand>(Lg*0.02))=[];
plot(n,m,'b.');
Lz=zeros(size(Lalp));
delta_alph=50;
Lz2=zeros(size(Lalp));
H=ones(5);
A=5;
z_kat=80;
[n,m]=OCT_NOISE_area(n,m,Lg,H);
plot(n,m,'g.'); hold on
n_1=[]; m_1=[];
al_1=[];
…
…
    plot(xw,yw,'k*','LineWidth',3)
nmabs_=[];
for jjx=1:size(n_1,1)
```

```
    for jjy=1:size(n_1,2)
        if (m_1(jjx,jjy)+n_1(jjx,jjy))>0
      nmabs_(jjx,jjy)=Lg(m_1(jjx,jjy),n_1(jjx,jjy));
        end
    end
end
blad_=[];
for cd=1:length(xw)
    blad_(cd)=min(min(abs(n_1-xw(cd))+abs(m_1-yw(cd))));
end
MN_w=[MN_w;[MN, sum(blad_)./length(blad_) min((blad_))
max((blad_))]];
end
figure;
[AX1,H1,H2] =
plotyy(MN_w(:,1),MN_w(:,2),MN_w(:,1),MN_w(:,4),'plot');
set(get(AX1(1),'Ylabel'),'String','\delta','FontSize',20,'C
olor','k')
set(get(AX1(2),'Ylabel'),'String','\delta_{min},\delta_{max
}','FontSize',20,'Color','k')
set(H1,'LineStyle','-','Marker','s','LineWidth',2)
set(H2,'LineStyle','-','Marker','+')
set(AX1(2),'Ylim',[min(min(MN_w(:,3:4))),max(max(MN_w(:,3:4
)))])
xlabel('M_MxN_M','FontSize',20)
grid on
hold on
[AX2,H1,H2] =
plotyy(MN_w(:,1),MN_w(:,2),MN_w(:,1),MN_w(:,3),'plot');
set(H2,'LineStyle','-','Marker','v');
set(AX2(2),'Ylim',[min(min(MN_w(:,3:4))),max(max(MN_w(:,3:4
)))]);
legend([AX1,AX2(2)],'\delta','\delta_{min}','\delta_{max}')
```

As it can be seen (Fig. 4-53)) the values of δ error fall within the 0.5-0.7 range, what is a small value as compared with the error originating during the algorithm operation for wide changes of other parameters.
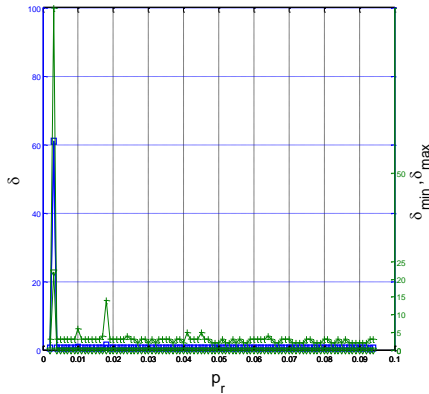
Fig. 4-54 Graph of error δ values changes and its minimum δ$_{min}$ and maximum δ$_{max}$ value vs. p$_r$
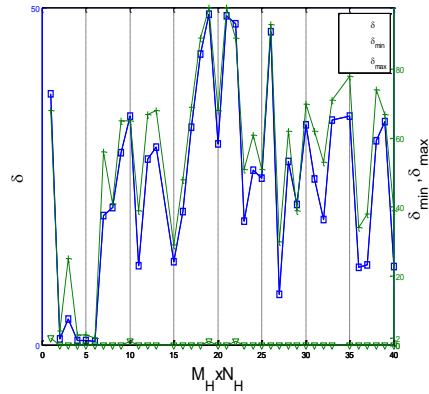
Fig. 4-55 Graph of error δ values changes and its minimum δ$_{min}$ and maximum δ$_{max}$ value vs. M$_H$xN$_H$

Fig. 4-54 shows the graph of error δ values changes and its minimum δ$_{min}$ and maximum δ$_{max}$ value vs. p$_r$. As it results from (29), the change of threshold p$_r$ value is directly connected with the number of selected points. For p$_r$=0.02 and higher values the number of random selected points is that large that it is possible to assume that starting from this value their number does not have a significant influence on error δ value. Fig. 4-55 shows the graph of error δ values changes and its minimum δ$_{min}$ and maximum δ$_{max}$ value vs. M$_H$xN$_H$. Both the choice of the points position correction area M$_H$xN$_H$ and the amplitude A$_{i,j}$ which in practical application is constant for various 'i' and 'j' is a key element affecting the error and thereby the precision of contours reconstruction. As may be seen from Fig. 4-55 the value of δ versus M$_H$xN$_H$ is relatively large for A$_{i,j}$=const=9 (for variables 'i' and 'j'), for which the computations were carried out. A strict relationship between error δ values changes vs. M$_H$xN$_H$ and A$_{i,j}$ is visible in Fig. 4-56 and Fig. 4-57 the maximum value δ$_{max}$ Fig. 4-57. Based on this it is possible to determine the relationship between M$_H$=N$_H$ and A$_{i,j}$, i.e.: M$_H$=N$_H$≈1.4*A$_{i,j}$ (in graphs in Fig. 4-56 and Fig. 4-57 for the minimum error value it may read e.g. M$_H$=N$_H$=25 at A$_{i,j}$=35).

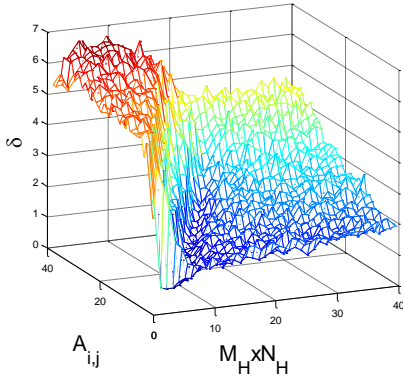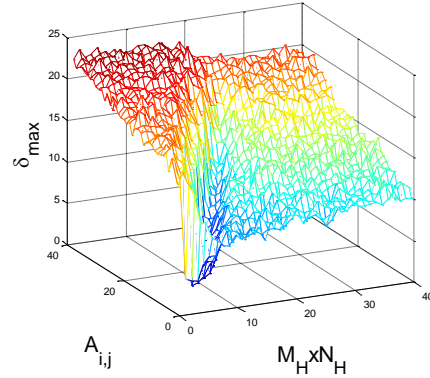**Fig. 4-56 Graph of error δ values changes versus $M_H x N_H$ and $A_{i,j}$**

Fig. 4-57 Graph of maximum error $δ_{max}$ values changes versus $M_H x N_H$ and $A_{i,j}$

From Fig. 4-56 and Fig. 4-57 it may be noticed that high error values occur for small $M_H x N_H$ values and high $A_{i,j}$. This results from the fact that the consecutive points $o_{i,j+1}$ are separated from $o_{i,j}$ by $A_{i,j}$ and their local position correction occurs within a small $M_H x N_H$ range. At high $A_{i,j}$ the rounding originating in computations of Lα value formula (28) causes large deviations of $o_{i,j+1}$ points from the standard contour, what substantially affects the δ and $δ_{max}$ error. Verification of these parameters may be implemented in a similar way as the previous source code with modifications in appropriate places. An attentive Reader will successfully introduce necessary modifications in appropriate place of the previous source code.

4.9.7   Assessment of Results Obtained from the Random Method

The method described gives correct results at contours determination (layers separation) both on OCT images as well as on others, for which classical methods of contours determination do not give results or the results do not provide a continuous contour. The algorithm drawbacks include a high influence of noise on the results obtained. This results from relationship (29) where pixels of pretty high value, resulting from a disturbance, increase the probability of selecting in this place a starting point and hence a component contour. The second drawback is the computations time, which is the longer the larger is the number of selected points and/or the reason, for which searching for the next points $o_{i,j+1}$ was stopped (these are limitations specified in section 4).

Fig. 4-58 below presents the enlarged results obtained for an example of OCT image.
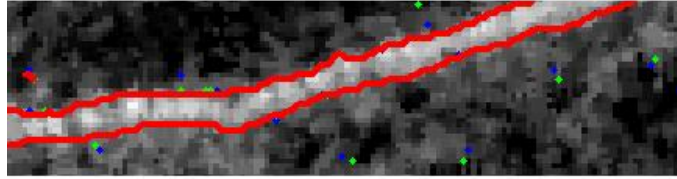
Fig. 4-58 Example of final enlarged result obtained for a real OCT image for p_r=0.02, $\Delta\alpha$=45°, M_HxN_H=35x35, p_b=2, p_j=0.8, $A_{i,j}$=25

The algorithm presented may be further modified and parametrised, e.g. through $A_{i,j}$ change for various 'i' and 'j' acc. to the criterion suggested or having considered weights of individual $o_{i,j}$ points and taking them into account as the iteration stopping condition etc.

4.10 Layers Recognition on Tomographic Eye Image Based on Canny Edge Detection

4.10.1 Canny Filtration

The input image L_{gray} is initially subject to filtration using a median filter of (M_hxN_h) size of h=13x13 mask. The obtained L_{MED} image is subject to another filtration using a modified Canny filter, for which the next filtration stages are presented in the next sections – as a reminder:

```
[Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255;
Lmed=medfilt2(Lgray,[13 13]);
Lmed=mat2gray(Lmed);
figure; imshow(Lmed)
```

The first stage of the edge detection method used [14], [35], [40], [41] consists of making a convolution of input image L_{MED} [6], i.e.:

$$L_{GX}(m,n) = \sum_{m_h=-M_h/2}^{M_h/2} \sum_{n_h=-N_h/2}^{N_h/2} L_{MED}(m+m_h, n+n_h) \cdot h_x(m_h,n_h) \qquad (39)$$

$$L_{GY}(m,n) = \sum_{m_h=-M_h/2}^{M_h/2} \sum_{n_h=-N_h/2}^{N_h/2} L_{MED}(m+m_h, n+n_h) \cdot h_y(m_h,n_h) \qquad (40)$$

with the following Gauss filters masks, e.g. of dimensions 3 x 3 (Fig. 4-59, Fig. 4-60):

$$\begin{bmatrix} 0.325 & 0.536 & 0.325 \\ 0 & 0 & 0 \\ -0.325 & -0.536 & -0.325 \end{bmatrix} \qquad \begin{bmatrix} 0.325 & 0 & -0.325 \\ 0.536 & 0 & -0.536 \\ 0.325 & 0 & -0.325 \end{bmatrix}$$

Fig. 4-59. Mask h_x of filter for the ox axis   **Fig. 4-60. Mask h_y of filter for the oy axis**

The matrix of gradient in both directions necessary to determine the edges has been determined in accordance with a classical dependence:

$$L_{GXY}(m, n) = \sqrt{L_{GX}(m, n)^2 + L_{GY}(m, n)^2} \tag{41}$$

and p_{xy} threshold:

$$p_{xy} = \varepsilon \cdot \left(\max_{m,n \in L_{GXY}} (L_{GXY}(m, n)) - \min_{m,n \in L_{GXY}} (L_{GXY}(m, n)) \right) + \min_{m,n \in L_{GXY}} (L_{GXY}(m, n)) \tag{42}$$

where ε is a coefficient selected within the range $\varepsilon \in (0,1)$.

A practical implementation of this, initial, phase of algorithm should not give rise to any difficulties:

```
    Nx1=13;
    Sigmax1=2;
    Nx2=13;
    Sigmax2=2;
    Theta1=pi/2;
    Ny1=13;
    Sigmay1=4;
    Ny2=13;
    Sigmay2=4;
    Theta2=0;
    epsilon=0.15;
hx=OCT_NOISE_gauss(Nx1,Sigmax1,Nx2,Sigmax2,Theta1);
Lgx= conv2(Lmed,hx,'same');
Lgx(Lgx<0)=0;
figure; imshow(Lgx,[])
hy=OCT_NOISE_gauss(Ny1,Sigmay1,Ny2,Sigmay2,Theta2);
Lgy=conv2(Lmed,hy,'same');
Lgy(Lgy<0)=0;
figure; imshow(Lgy,[])
Lgxy=sqrt(Lgx.*Lgx+Lgy.*Lgy);
```

```
figure; imshow(Lgxy)
I_max=max(max(Lgxy));
I_min=min(min(Lgxy));
pxy=epsilon*(I_max-I_min)+I_min;
Lgxym=max(Lgxy,pxy.*ones(size(Lgxy)));
figure; imshow(Lgxym,[])
```

The obtained images are shown below (Fig. 4-61 - Fig. 4-64).
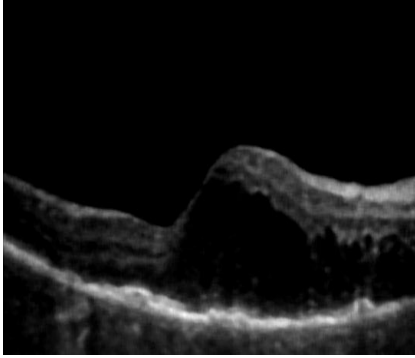
Fig. 4-61 Image L$_{MED}$
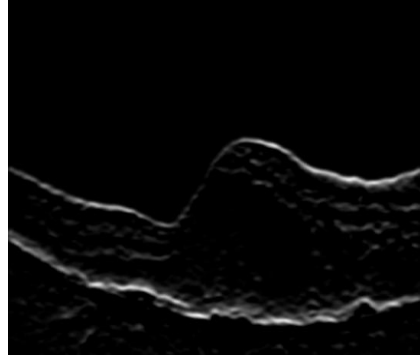
Fig. 4-62 Image L$_{GX}$
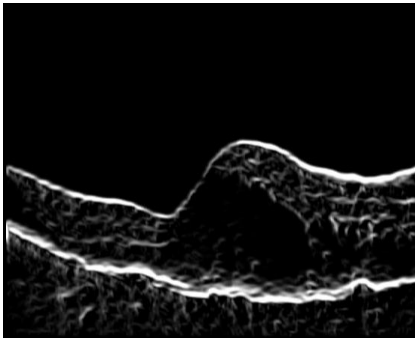
Fig. 4-63 Image L$_{GY}$
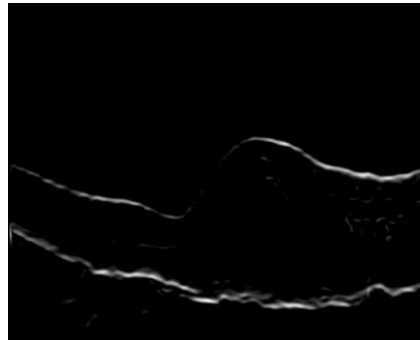
Fig. 4-64 Image L$_{GXYM}$

For the final form of the formula for the matrix of edges containing image, i.e. L$_{BIN\_KR}$ it is necessary to define L$_{GXYM}$, i.e.:

$$L_{GXYM}(m,n) = \begin{cases} p_{xy} & \text{if} \quad L_{GXY}(m,n) < p_{xy} \\ L_{GXY}(m,n) & \text{if} \quad L_{GXY}(m,n) \ge p_{xy} \end{cases} \tag{43}$$

and (x_i, y_i) and (x_j, y_j) coordinates of i_{xy} and j_{xy} values, respectively, determined from the relationship

$$x_i = \cos(\alpha(m,n)) \text{ oraz } x_j = -\cos(\alpha(m,n)) \tag{44}$$

$$y_i = \sin(\alpha(m,n)) \text{ oraz } y_j = -\sin(\alpha(m,n)) \tag{45}$$

where angle α was determined for each pair of pixels L_{GX} and L_{GY}:

$$\alpha(m,n) = \operatorname{atan}\left(\frac{L_{GY}(m,n)}{L_{GX}(m,n)}\right) \tag{46}$$

and then the i_{xy} and j_{xy} values, which assume the level of saturation acc. to values interpolated on the plane determined from the area of 3 x 3 resolution from $L_{GXYM}(m\pm\Delta m, n\pm\Delta n)$, where Δm and Δn are equal to 1 (Fig. 4-65, Fig. 4-66).
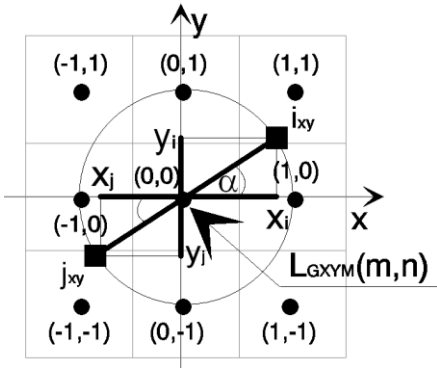
Fig. 4-65 Graphic interpretation of i_{xy} and j_{xy} points location in a fragment of $L_{GXYM}(m\pm1, n\pm1)$ image
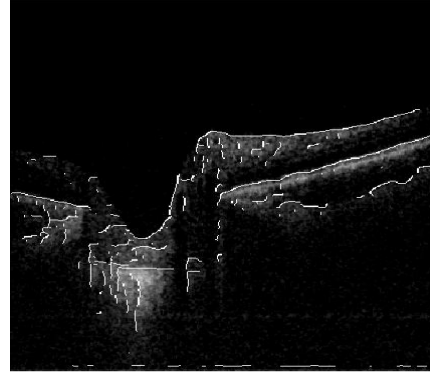
Fig. 4-66 Input image L_{MED} and white pixels of L_{BIN_KR} image

Hence the output image of edges determined using the Canny method L_{BIN_KR} is equal to:

$$L_{BIN\_KR}(m,n) = \begin{cases} 0 & \text{if} & L_{GXYM}(m,n) \le p_{xy} \\ 1 & \text{if} & (L_{GXYM}(m,n) > p_{xy}) \wedge (L_{GXYM}(m,n) > i_{xy}(m,n)) \wedge (L_{GXYM}(m,n) > j_{xy}(m,n)) \\ 0 & \text{if} & (L_{GXYM}(m,n) > p_{xy}) \wedge [(L_{GXYM}(m,n) \le j_{xy}(m,n)) \vee (L_{GXYM}(m,n) \le i_{xy}(m,n))] \end{cases} \tag{47}$$

An example of OCT image generated for $\varepsilon = 0.15$, where for better assessment of results obtained white pixels of L_{BIN_KR} image have been shown in Fig. 4-66. The source code of this part is given below

```
[M,N]=size(Lgxym);
Lkr=zeros(size(Lgxym));
for m=2:M-1,
for n=2:N-1,
    if Lgxym(m,n) > pxy,
        X=[-1,0,+1;-1,0,+1;-1,0,+1];
        Y=[-1,-1,-1;0,0,0;+1,+1,+1];
        Z=[Lgxym(m-1,n-1),Lgxym(m-1,n),Lgxym(m-
1,n+1);Lgxym(m,n-1),Lgxym(m,n),Lgxym(m,n+1);Lgxym(m+1,n-
1),Lgxym(m+1,n),Lgxym(m+1,n+1)];
        alp=atan2(Lgy(m,n),Lgx(m,n));
        ss=sin(alp);
        cc=cos(alp);
        XI=[cc, -cc];
        YI=[ss, -ss];
        ZI=interp2(X,Y,Z,XI,YI);
        if Lgxym(m,n) >= ZI(1) & Lgxym(m,n) >= ZI(2)
            Lkr(m,n)=I_max;
        else
            Lkr(m,n)=I_min;
        end
    else
        Lkr(m,n)=I_min;
    end
end
end
figure; imshow(Lkr,[]);
Lbin_kr=Lkr>0;
figure; imshow(Lbin_kr)
```

The results obtained are presented in Fig. 4-67, Fig. 4-68.

Fig. 4-67 Image L$_{KR}$
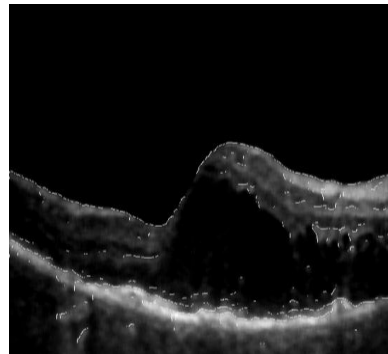
Fig. 4-68 Image L$_{BIN\_KR}$ imposed on image L$_{MED}$

The L$_{BIN\_KR}$ image further on provides the basis for the next steps of the algorithm operation.

4.10.2 Features of Line Edge

For the L_{BIN_KR} image a labelling operation has been carried out, where each cluster (of values '1') has its label $e_t = 1, 2,...,E_t-1, E_t$.

```
Lind=bwlabel(Lbin_kr);
figure; imshow(Lind,[]); colormap('jet'); colorbar
```

Then for each label e_t a dilatation operation is performed for a rectangular structural element SE_d of dimension 5 x 1 oriented acc. to the value of angle $\alpha(m,n)$, where the origin of coordinates was placed in its first row [26]. The obtained L_{IND} image in pseudocolours is shown in Fig. 4-69.

e_t	P_{et}	I_{et}
1	26	0.2524
2	33	1.000
3	43	0.2038
4	25	0.1344
5	4	0.1250
6	1	0.1250
7	10	0.1250
8	16	0.1250
9	36	0.1250
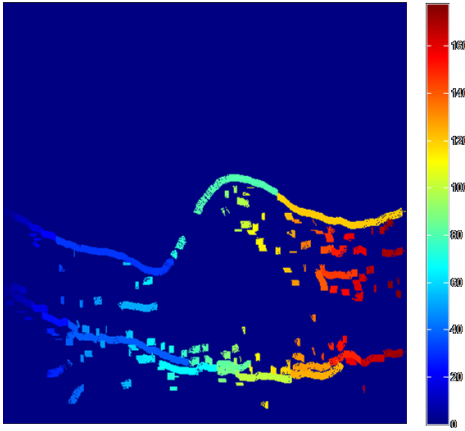
Fig. 4-69 Image L_{IND} in pseudocolours (label 178)

Fig. 4-70 Table of weights with examples of values for objects with first labels e_t

Fig. 4-70 shows weight values for consecutive (from among the initial ones) labels of L_{IND} image (Fig. 4-69) i.e. binary images L_{et}, where P_{et} is the surface of object for label e_t and I_{et} is the average value of its level of grey, i.e.:

$$P_{et} = \sum_{m=1}^{M}\sum_{n=1}^{N} L_{et}(m,n) \tag{48}$$

$$I_{et} = \frac{1}{M \cdot N} \cdot \sum_{m=-1}^{M}\sum_{n=-1}^{N}\left(L_{et}(m,n) \cdot L_{MED}(m,n)\right) \tag{49}$$

The determined P_{et} and I_{et} values will be later on used as features during ultimate analysis of edge lines. These values have been written in order in the data variable in the following source code:

```
data=[]; xd=[]; xdpk=[]; yd=[]; ydpk=[];
Let_=zeros(size(Lind));
for et=1:max(Lind(:))
Let=(Lind==et);
[xx_,yy_]=meshgrid(1:size(Let,2),1:size(Let,1));
xx_(Let==0)=[];
yy_(Let==0)=[];
xd(et,1:length(xx_))=xx_;
yd(et,1:length(yy_))=yy_;
xdpk(et,1:2)=[xx_(1),xx_(end)];
ydpk(et,1:2)=[yy_(1),yy_(end)];
Let2=Let;
Let3=Let;
for i=8:(size(Let,1)-8)
    for j=8:(size(Let,2)-8)
        p=Let(i,j);
        if p>0;
            alp=atan2(Lgy(i,j),Lgx(i,j));
            ss=sin(alp);
            cc=cos(alp);
            Let2(round(i+ss),round(j+cc))=p;
            Let2(round(i+2*ss),round(j+2*cc))=p;
            Let2(round(i+3*ss),round(j+3*cc))=p;
            Let2(round(i+4*ss),round(j+4*cc))=p;
            Let2(round(i+5*ss),round(j+5*cc))=p;
            Let2(round(i+6*ss),round(j+6*cc))=p;
            Let2(round(i+7*ss),round(j+7*cc))=p;
            Let3(round(i-ss),round(j-cc))=p;
            Let3(round(i-2*ss),round(j-2*cc))=p;
            Let3(round(i-3*ss),round(j-3*cc))=p;
            Let3(round(i-4*ss),round(j-4*cc))=p;
            Let3(round(i-5*ss),round(j-5*cc))=p;
            Let3(round(i-6*ss),round(j-6*cc))=p;
            Let3(round(i-7*ss),round(j-7*cc))=p;
        end
    end
end
Let_((Let2+Let3)>0)=et;
data(et,1)=et;
data(et,2)=sum(sum(Let));
Lmed_1=Let2.*Lmed; Lmed_1(Let2==0)=[];
Lmed_2=Let3.*Lmed; Lmed_2(Let3==0)=[];
Lmed_3=Let.*Lmed; Lmed_3(Let3==0)=[];
data(et,4)=mean(Lmed_1)-mean(Lmed_2);
data(et,3)=mean(Lmed_3);
```

```
end
figure; imshow(Let_,[]); colormap('jet'); colorbar
```

Matrices L_{et2} and L_{et3} have been used in the above source code, being the result of dilatation on the one and on the other side of analysed pixel of the e_t area. In addition, coordinates of the beginning and of the end of the analysed e_t area have been written in variables z_{dpk} and y_{dpk}. This data will be necessary at a further stage of connecting individual contour fragments.

4.10.3 Contour Line Correction

Each solid line of edge visible in L_{et} image for labels e_t=1,2,...,E_t-1,E_t is transformed into the form of x_{et} and y_{et} vector of points' coordinates in a Cartesian coordinate system. The method of contour line correction is applied to 'elongation' of each edge in both directions. To this end for the first two pairs of coordinates of the first edge $(x_1(1), y_1(1))$ and $(x_1(2), y_1(2))$ as well as for the last two $(x_1(end-1), y_1(end-1))$ and $(x_1(end), y_1(end))$ a straight line passing through those points is determined (end – means the last element), i.e. in accordance with demonstrative illustration below (Fig. 4-71):
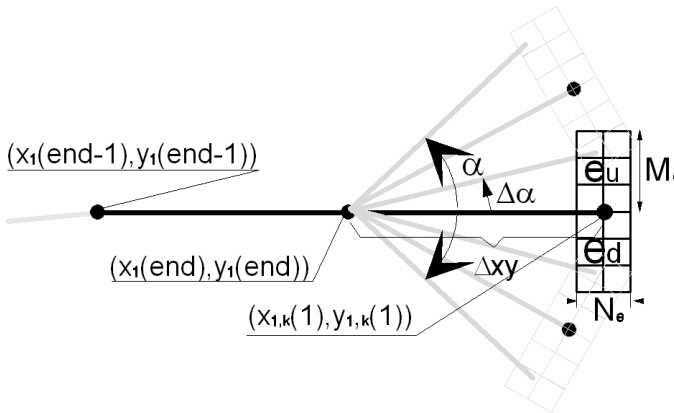
Fig. 4-71 Graphic interpretation of the contour correction method to determine consecutive points starting from the position of points (x_1(end-1),y_1(end-1)) and (x_1(end),y_1(end)) for a new point (pixel) to be determined ($x_{1,k}$(1),$y_{1,k}$(1)). To simplify, the angle of inclination of end points of the edges has been set as β=0°

Fig. 4-71 presents the ideas of contour correction method, where starting from the position of points $(x_1(end-1),y_1(end-1))$ and

$(x_1(end),y_1(end))$ the straight line passing through them is determined with a slope β_1, i.e.:

$$\beta_1\left(x_1(end), y_1(end)\right) = \operatorname{atan}\left(\frac{y_1(end) - y_1(end-1)}{x_1(end) - x_1(end-1)}\right) \qquad (50)$$

and at the distance of Δxy the position of new point $(x_{1,k}(1), y_{1,k}(1))$ is determined for its various potential positions (within the angle range $\beta_1(1)\pm\alpha$ every $\Delta\alpha$). The selection of right position of a contour point obtained by adding consecutive points to the existing edge is obtained based on the analysis of mean value from $e_{u1}(x_u,y_u,\alpha,1)$ and $e_{u1}(x_d,y_d,\alpha,1)$ areas of M_e x N_e size. The difference ΔS is determined for each position of point $(x_{1,k}(1), y_{1,k}(1))$:

$$\Delta S(1,\alpha) = \frac{1}{M_e \cdot N_e} \cdot \qquad (51)$$

$$\left(\sum_{y_u=1}^{M_e}\sum_{x_u=1}^{N_e} e_u(x_u,y_u,\alpha,1)\cdot h_u(x_u,y_u) - \sum_{y_d=1}^{M_e}\sum_{x_d=1}^{N_e} e_d(x_d,y_d,\alpha,1)\cdot h_d(x_d,y_d)\right)$$

where:

x_u, y_u – coordinates of consecutive elements of matrix e_u and h_u situated atop relative to the analysed point $(x_{1,k}(1),y_{1,k}(1))$, for which $x_u \in \{1,2,..., N_u-1,N_u\}$ and $y_u \in \{1,2,..., N_u-1,N_u\}$

x_d, y_d – coordinates of consecutive elements of matrix e_d and h_d situated at the bottom relative to the analysed point $(x_{1,k}(1),y_{1,k}(1))$, for which $x_d \in \{1,2,..., N_d-1,N_d\}$ and $y_d \in \{1,2,..., N_d-1,N_d\}$

and h_u and h_d masks for M_exN_e =3x2

$$h_u = \begin{bmatrix} 0.5 & 0.5 \\ 0.7 & 0.7 \\ 1 & 1 \end{bmatrix} \qquad\qquad h_d = \begin{bmatrix} 1 & 1 \\ 0.7 & 0.7 \\ 0.5 & 0.5 \end{bmatrix}$$

Fig. 4-72 Mask h_u for M_exN_e =3x2   **Fig. 4-73 Mask h_d for M_exN_e =3x2**

The areas (matrices) e_u and e_d of M_exN_e size are created based on angle β and α every $\Delta\alpha$ in the following way:

$$e_{u1}(x_u, y_u, \alpha, 1) =$$
$$= L_{MED}(y_{1,k}(1) + y_u \cdot \cos(\beta_1(1) + \alpha + 90), x_{1,k}(1) + x_u \cdot \sin(\beta_1(1) + \alpha + 90)) \tag{52}$$

$$e_{d1}(x_d, y_d, \alpha, 1) =$$
$$= L_{MED}(y_{1,k}(1) - y_d \cdot \cos(\beta_1(1) + \alpha + 90), x_{1,k}(1) - x_d \cdot \sin(\beta_1(1) + \alpha + 90)) \tag{53}$$

where $x_u \in \{1,2,3,... N_e-1, N_e\}$ and $x_u \in \{1,2,3,... N_e-1, N_e\}$ and $\beta_1(1)$, in general $\beta_1(v_1)$:

$$\beta_1(v_1) = a\tan\left(\frac{y_{1,k}(v_1) - y_{1,k}(v_1 - 1)}{x_{1,k}(v_1) - x_{1,k}(v_1 - 1)}\right) \tag{54}$$

for $v_1 \in \{2,3,... V_1-1, V_1\}$, V_1 – a total number of points of contour correction implemented for line 1 of the contour.

The angle, for which there is the best fit of the analysed point $(x_{1,k}(v_1), y_{1,k}(v_1))$, is calculated as α^* for which $\Delta S(v_1, \alpha)$ reaches a maximum or minimum depending on the position and brightness of the analysed object.

$$\Delta S(v_1, \alpha^*) = \max_\alpha(\Delta S(v_1, \alpha)) \tag{55}$$

Consecutive points determined for increasing v_1 must be limited. The minimum value $\Delta S(v_1, \alpha^*)$ limited by threshold p_r is this bound.

The suggested method of contour correction has very interesting properties. Parameters of this part of algorithm include:

α - the angle, within which the best fit is sought with regard to the given criterion,

$\Delta\alpha$ - accuracy, with which the best fit is sought,

Δxy - the distance between the current and the next sought point of the active contour,

M_e - height of analysed area e_u and e_d,

N_e - width of analysed area e_u and e_d.

The function constructed on this basis is presented below.

```
function
[x_out,y_out,wagi,iter]=OCT_COR_LINE(Lmed,x_in,y_in,udxy_,m
ene,alpha,iter_,pr,dxy)
wagi=[];
xi=x_in(end); yi=y_in(end);
```

```
beta=atan2((y_in(end)-y_in(end-1)),(x_in(end)-x_in(end-
1)));
x_out=xi;y_out=yi;
for iter=1:iter_
    eu=[]; ed=[]; deltaS=[];
    for alpha_=-alpha:alpha
        for udxy=0:udxy_
            yi_=yi+udxy*sin(beta+alpha_*pi/180);
            xi_=xi+udxy*cos(beta+alpha_*pi/180);
            al_be=beta+(alpha_+90)*pi/180;
            ss=sin(al_be);
            cc=cos(al_be);
            for mene_=1:mene
                yy=round(yi_+mene_*ss);
                xx=round(xi_+mene_*cc);
                if
(yy>1)&(yy<=size(Lmed,1))&(xx>1)&(xx<=size(Lmed,2))
                    eu(udxy+1,mene_)=Lmed(yy,xx)/mene_;
                else
                    eu(udxy+1,mene_)=0;
                end
            end
            for mene_=1:mene
                yy=round(yi_-mene_*ss);
                xx=round(xi_-mene_*cc);
                if
(yy>1)&(yy<=size(Lmed,1))&(xx>1)&(xx<=size(Lmed,2))
                    ed(udxy+1,mene_)=Lmed(yy,xx)/mene_;
                else
                    ed(udxy+1,mene_)=1;
                end
            end
        end
            deltaS=[deltaS;[alpha_,mean(ed(:))-
mean(eu(:))]];
    end
    deltaS=sortrows(deltaS,2);
    if  deltaS(1,2)>pr
        break
    end
        wagi(iter)=deltaS(1,2);
    al_be_=beta+deltaS(1,1)*pi/180;
            yi=yi+dxy*sin(al_be_);
            xi=xi+dxy*cos(al_be_);
    beta=al_be_;
    xyxy=[x_out',y_out'];
    if  sum(((round(xyxy(:,1))==round(xi))  +
(round(xyxy(:,2))==round(yi)))==2)>=2
        break
    end
```

```
    x_out=[x_out,xi];
    y_out=[y_out,yi];
end
end
```

Fig. 4-74 - Fig. 4-77 below present the results obtained for an artificial image of a square for modified aforementioned parameters α, Δα, Δxy, M_e, N_e changed within the range $α \in \{1,2,3,...,19,20\}$, $Δxy=N_e \in \{1,2,3,...,19,20\}$, $M_e \in \{1,2,3,...,19,20\}$ for Δα=1, and p_r=-0.001. Also the number of iterations was limited to 50.
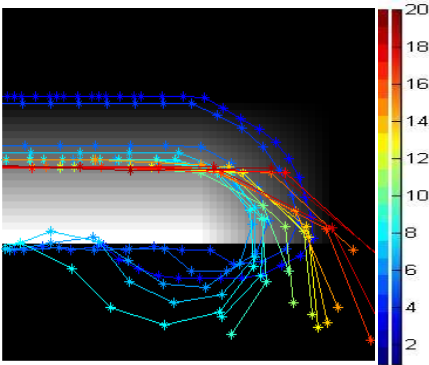
Fig. 4-74 Artificial image and fragment of contour correction action for α=40, Δα=1, M_e=10, Δxy=N_e changed within the range (1,20)
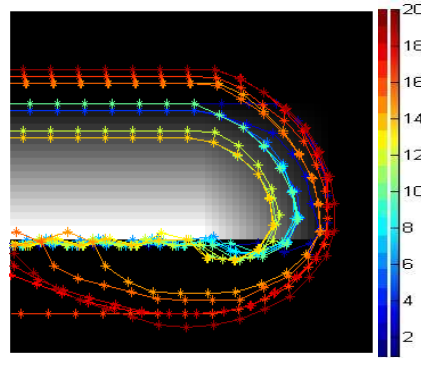
Fig. 4-75 Artificial image and fragment of contour correction action for α=40, Δα=1, Δxy=N_e=4, M_e changed within the range (1,20)

Fig. 4-76 Artificial image and fragment of contour correction action for α=40, M_e=10, Δxy=N_e=10, Δα changed within the range (1,20)

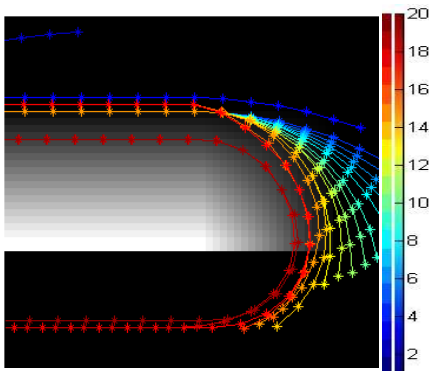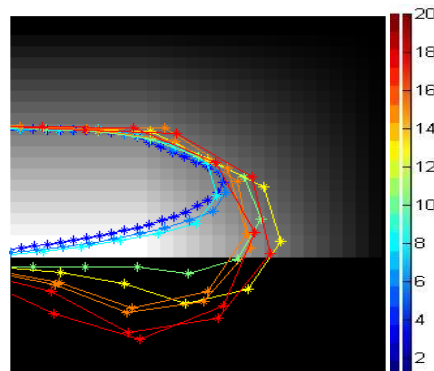**Fig. 4-77 Artificial image and fragment of contour correction action for α=45, M_e=10, N_e=10, Δα=1, and Δxy changed within the range (1,20)**

The presented contour correction method has the following properties:

α - angle defining the range sought in the sense of degree of object edge curvature,

$\Delta\alpha$ - accuracy, with which the degree of edge curvature is sought,

Δxy - distance between the current and next sought point affecting the extent of generalisation and approximation of intermediate values (placed between points),

M_e - height of analysed area affecting the algorithm capability to find objects of higher level of detail,

N_e - width of analysed area averaging the contour sought along edges.

The experiments and algorithm parameters measurements presented (Fig. 4-74 - Fig. 4-77) can be easily followed using a short source code:

```
Lmed=zeros(300); Lmed(200:250,100:250)=1;
Lmed=conv2(Lmed,ones(19))./sum(sum(ones(19)));Lmed(220:end,
:)=0;
figure; imshow(Lmed)
x_in=[100,101]; y_in=[200,200];
hold on; plot(x_in,y_in,'*g-')
map=jet(20);
udxy_=4;
iter_=70;
pr=-0.0001;
dxy=4;
alpha=45;
for mene=1:20
[x_out,y_out,wagi,iter]=OCT_COR_LINE(Lmed,x_in,y_in,udxy_,m
ene,alpha,iter_,pr,dxy)
    hold on; plot(x_out,y_out,'*-','color',map(mene,:))
    axis([222 275 186 236])
    pause(0.05)
end
```

We encourage Readers here to perform independent changes of x_in,y_in,udxy_,mene,alpha,iter_,pr,dxy values and to experimentally verify these parameters influence on the result obtained.

4.10.4 Final Analysis of Contour Line

The obtained individual lines of edges e_t and corresponding values I_{et} and P_{et} (average value of brightness and surface) have been adjusted. Because those edges have been removed, which have $I_{et} < p_r \cdot \max_{et \in \{1,2,3,...Et\}} (I_{et})$ and for which $P_{et} < p_r \cdot \max_{et \in \{1,2,3,...Et\}} (P_{et})$ where threshold p_r was arbitrarily taken as 0.2 (20%). For the other edges e_k,

which have not been removed, the adjustment was made using on their ends the active contour method. The values of active contour parameters were taken as $\alpha=45$, $\Delta\alpha=1$, $\Delta xy=1$, $M_e=11$, $N_e=11$. Iterations for individual e_k edges of active contour method were interrupted, when one of the following situations occurred:

- the acceptable iterations number was exceeded – set arbitrarily at 1000,
- for that point the condition $\Delta S(v_{ek}, \alpha^*)<p_s$ has not been met, where p_s was set at -0.02,
- at least two points have the same coordinates – this prevents looping of the algorithm.

Results obtained for parameters determined this way are presented below (Fig. 4-78, Fig. 4-79).
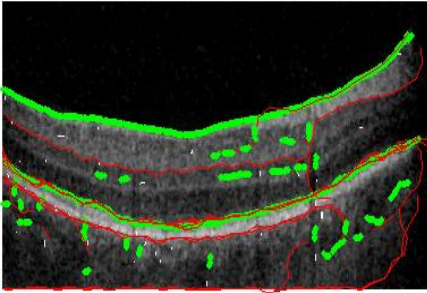
Fig. 4-78 Action of modified active contour on a real image for $\alpha=40$, $\Delta\alpha=1$, $\Delta xy=N_e=11$, $M_e=10$. The green line marks the contour obtained from the Canny method, the red line marks consecutive points of active contour method
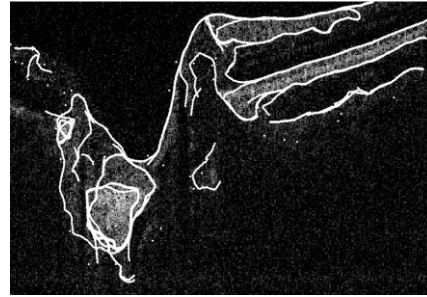
Fig. 4-79 Action of modified active contour after the described correction on a real image for $\alpha=40$, $\Delta\alpha=1$, $\Delta xy=N_e=11$, $M_e=11$

As shown in the figures above (Fig. 4-78, Fig. 4-79) the method suggested correctly detects individual layers on an OCT eye image. Further stages, which are planned in this approach continuation, are related to a deeper analysis of the algorithm in terms of parameters selection. The discussed algorithm fragment looks as follows:

```
figure; imshow(Lmed,[]); hold on
hh=waitbar(0,'Please wait...')
for et=1:max(Lind(:))
    Let=(Lind==et);
    [x_in,y_in]=meshgrid(1:size(Let,2),1:size(Let,1));
```

```matlab
    x_in(Let==0)=[];
    y_in(Let==0)=[];
    mene=15;
    udxy_=10;
    alpha=45;
    dxy=1;
    pr=-0.01;
        if length(x_in)>5
        [x_out,y_out,wagi,iter]=OCT_COR_LINE(Lmed,
x_in,y_in, udxy_,mene,alpha,1275,pr,  dxy);
        hold on;
        plot(x_out,y_out,'w.')
        pause(0.1)
        end
  waitbar(et/max(Lind(:)))
end
close(hh)
```

We encourage the Reader again to modify parameters of function OCT_COR_LINE allowing obtaining proper results and enabling learning the function capabilities. A few artefacts, resulting from improper selection of OCT_COR_LINE function parameters, are presented below.
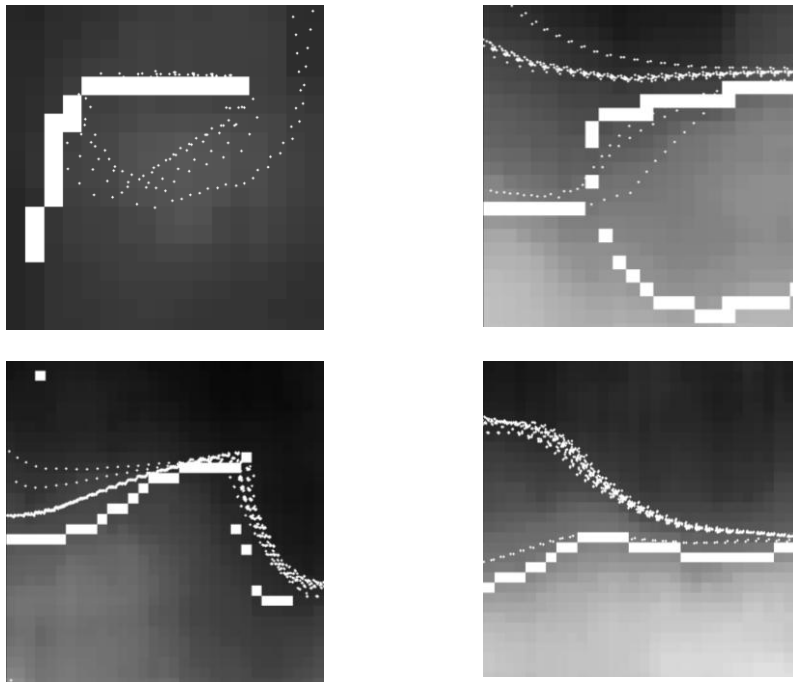
Fig. 4-80 Examples of artefacts resulting from improper selection of function OCT_COR_LINE parameters

The presented method of combination of Canny edge detection algorithm with the modified active contour algorithm is applied in detection of external limiting membranes on tomographic OCT eye images. The method proposed may be used during images segmentation into other contents than presented, provided that values of parameters mentioned are modified [23]. Despite satisfactory results presented above there is a pretty large area for research related to modification of the algorithm presented in terms of operation time optimisation. The time of analysis in this, as well as in many other cases of image analysing applications, is of crucial importance in practical use. In terms of functionality, implementation difficulties, the speed of operation, this method may be classified as an average one.

4.11 Hierarchical Approach in the Analysis of Tomographic Eye Image

4.11.1 Image Decomposition

Images originating from a Copernicus tomograph due to its specific nature of operation are obtained in sequences of a few, a few dozen 2D images within approx. 1s, which provide the basis for 3D reconstruction [42]. Because of their number, the analysis of a single 2D image should proceed within a time not exceeding 10, 20, 30, 40, 50 ms, so that the time of operator's waiting for the result would not be onerous (as it could be easily calculated for the above value, for a few dozen images of resolution usually M x N = 740 x 820 in a sequence, this time will be shorter than 1 s).

At the stage of image preprocessing the input image L_{GRAY} is initially subject to filtration using a median filter of $(M_h x N_h)$ size of mask h equal to $M_h x N_h = 3x3$ (in the final software version this mask may be set as $M_h x N_h = 5x5$ to obtain a better precision of algorithm operation for certain specified group of images), i.e.:.

```
[Lgray,map]=imread(['D:\OCT\FOLDERS\2.OCT\SKAN7.bmp']);
Lgray=Lgray(1:850,:);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255;
Lm=medfilt2(Lgray,[3 3]);
Lm=mat2gray(Lm);
figure; imshow(Lm)
```

Image L_M obtained this way is subject consecutively to decomposition to an image of lower resolution and analysed in terms of layers detection.

As an assumption, different than those presented in previous algorithm sections, the algorithm described should provide satisfactory results mainly from the operation speed criterion point of view. Although methods (algorithms) described feature high precision of computations, however, they are not fast enough (it is difficult to obtain the speed of single 2D image analysis on a PII 1.33 GHz processor in a time not exceeding 10 ms). Therefore a reduction of image L_M resolution by approx. a half was proposed to such value of pixels number in lines and columns, which is a power of '2', i.e.: MxN=256x512 (L_{M2}) applying further on its decompositions to image L_{D16} (where symbol 'D' means decompositions, while '16' the size of block, for which it was obtained), i.e.:

```
d=16;
fun = @(x) median(x(:));
Ld16 = blkproc(Lm,[d d],fun);
```

Each pixel of the input image after decomposition has a value equal to a median of the area (block) of 16x16 size of the input image, acc. to Fig. 4-81.
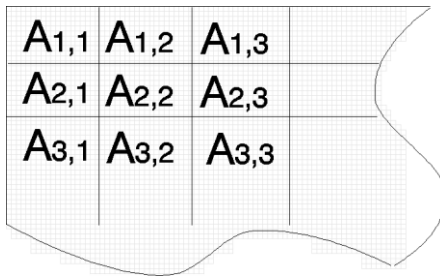
Fig. 4-81 Blocks arrangement on the L_M image
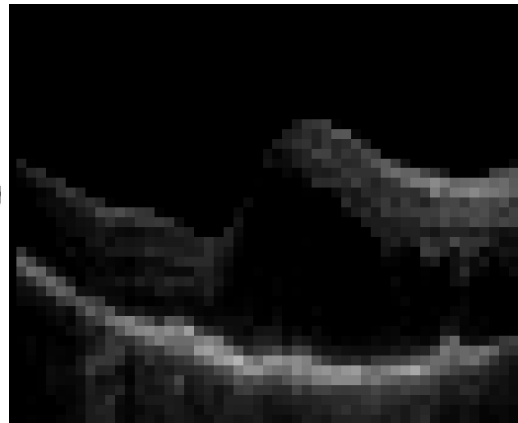
Fig. 4-82 OCT image after decomposition – L_{D16}

An example of result L_{D16} is presented in **Błąd! Nie można odnaleźć źródła odwołania.**Fig. 4-82. Image L_{D16} is then subject to determination of pixels position of maximum value for each column, i.e.:

$$L_{DM16}(m,n) = \begin{cases} 1 & \text{if} \quad L_{D16}(m,n) = \max_{m}(L_{D16}(m,n)) \\ 0 & \text{other} \end{cases} \quad (56)$$

where   m – means a row numbered from one,

n – means a column numbered from one.

Appropriate record in Matlab

```
Ldm16=Ld16==(ones([size(Ld16,1) 1])*max(Ld16));
figure; imshow(Ldm16,'notruesize')
```

Using the described method of threshold setting for the maximum value in lines, in 99 percent of cases only one maximum value in a column is obtained (Fig. 4-83).
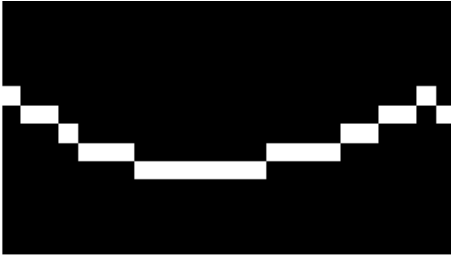
Fig. 4-83 Example of L$_{DM16}$ image   **Fig. 4-84 Example of L$_{DB16}$ image**

To determine precisely the position of NFL and RPE boundaries (Fig. 4-82) it turned out necessary to use one more L$_{DB16}$ image, i.e.:

$$L_{DB16}(m,n) = \begin{cases} 1 & \text{if} & |L_{D16}(m,n) - L_{D16}(m+1,n)| > p_r \\ 0 & \text{other} \end{cases} \tag{57}$$

for $m \in (1, M-1)$, $n \in (1,N)$, where p_r – the threshold assumed within the range (0, 0.2).

A record in Matlab looks as follows:

```
Ldb16_=zeros(size(Ld16));
for n=1:size(Ld16,2)-1
        Ldb16_(1:end-1,n)=diff(Ld16(:,n));
end
pr=0.1;
Ldb16=Ldb16_>pr;
figure; imshow(Ldb16,'notruesize')
```

As a result, coordinates of NFL and RPE boundaries position points are obtained as such positions of values '1' on L$_{DB16}$ image, for which $y_{NFL} \leq y_{RPE}$ and y_{RPE} is obtained from L$_{DB16}$ image in the same way.

This method for p_r threshold selection at the level of 0.01 gives satisfactory results in around 70 percent of cases of not composed images (i.e. such, which are not images with a visible pathology). Unfortunately for the other 30 percent cases the selection of p_r threshold in the adopted limits does not reduce the originated errors (Fig. 4-84).

The correction on this level of erroneous recognitions of NFL and RPE layers is that important, that for this approach these errors will not be duplicated (in the hierarchical approach presented below) for the subsequent more precise approximations.

4.11.2 Correction of Erroneous Recognitions

In L_{DB16} image (Fig. 4-84) white pixels are visible in an excess number for most columns. Two largest objects arranged along 'maxima' in columns entirely coincide with NFL and RPE limits position. Based on that and having carried out the above analysis for several hundred images, the following limitations were adopted:

- for coordinates y_{RPE} found on L_{DM16} image there must be at the same time $L_{DM16}(m,n)=1$ in other cases this point is considered as disturbance or as a point of $G_w(n)$ layer,

- if only one pixel of value '1' occurs on image L_{DM16} and L_{DB16} for the same position, i.e. for the analysed n there is $L_{DM16}(m,n) = L_{DB16}(m,n)$ the history is analysed for n>1 and it is checked, whether $|y_{NFL}(n-1) - y_{NFL}(n)| > |y_{RPE}(n-1) - y_{RPE}(n)|$, i.e.:

$$Rp(n) = \begin{cases} m & \text{if} \quad \begin{aligned} & L_{DB16}(m,n) = L_{DM16}(m,n) = 1 \wedge \\ & \wedge |y_{NFL}(n-1) - y_{NFL}(n)| > |y_{RPE}(n-1) - y_{RPE}(n)| \end{aligned} \\ 0 & \text{other} \end{cases} \quad (58)$$

for $m \in (1,M)$, $n \in (2,N)$

- if $|y_{NFL}(n-1)-y_{NFL}(n)| \leq |y_{RPE}(n-1)-y_{RPE}(n)|$, the condition $y_{NFL}(n-1)-y_{NFL}(n)= \pm 1$ is checked (giving thereby up fluctuations against history n-1 within the range ± 1 of area A (Fig. 4-81)). If so, then this point is the next $y_{NFL}(n)$ point. In the other cases the point is considered as a disturbance. It is assumed that lines coincide $y_{NFL}(n)=y_{RPE}(n)$ if $y_{RPE}(n-1)-y_{RPE}(n)= \pm 1$ and only one pixel occurs of value '1' on L_{DM16} image.

- in the case of occurrence in specific column of larger number of pixels than 2, i.e. if $\sum_m(L_{DB16}(m,n)) > 2$ a pair is matched (if occurs) $y_{NFL}(n-$

1), $y_{RPE}(n-1)$ so that $|y_{NFL}(n-1)-y_{NFL}(n-1)|-|y_{RPE}(n)-y_{RPE}(n)|=\pm1$ would occur. In this case it may happen that lines $y_{NFL}(n)$ and $y_{RPE}(n)$ will coincide. However, in the case of finding more than one solution, that one is adopted, for which $L_{D16}(y_{NFL}(n),n)+L_{D16}(y_{RPE}(n),n)$ assumes the maximum value (the maximum sum of weights in L_{D16} occurs).

The presented correction gives for the above class of images the effectiveness of around 99% of cases. Despite adopted limitations the method gives erroneous results for the initial value n=1, unfortunately these errors continue to be duplicated.

Fig. 4-85 Examples of L_{DB16} images for p_r=0.01 with incorrectly marked $y_{NFL}(n)$, $y_{RPE}(n)$ points (layers)

Unfortunately, the adopted relatively rigid conditions of acceptable difference $|y_{NFL}(n-1)-y_{NFL}(n-1)|$ or $|y_{RPE}(n)-y_{RPE}(n)|$ cause origination of large errors for another class of tomographic images, on which a pathology occurs in any form (Fig. 4-86).
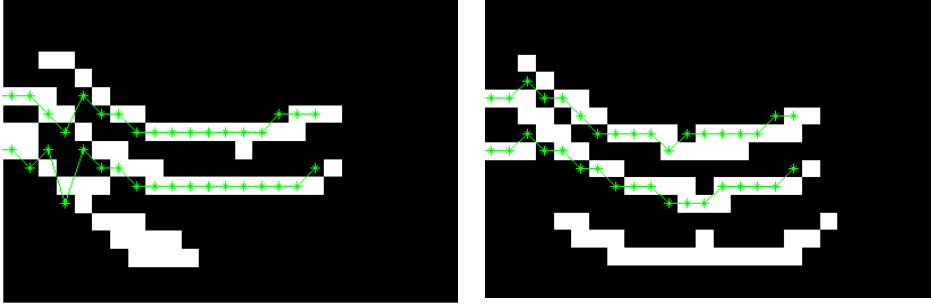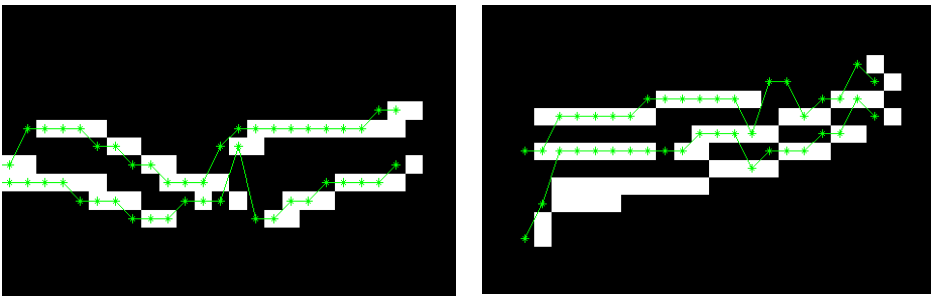

Fig. 4-86 Examples of L_{DB16} images for p_r=0.01 with incorrectly marked $y_{NFL}(n)$, $y_{RPE}(n)$ points (layers)

As it may be seen in Fig. 4-85 and Fig. 4-86 problems occur not only for the initial n values, but also for the remaining points. The reason for

erroneous recognitions of layers positions consists of difficulty in distinguishing proper layers in the case of discovering three 'lines', three points in a specific column, which position changes in acceptable range for individual n.

These errors cannot be eliminated at this stage of decomposition into 16x16 pixels areas (or 32x32 image resolution). They will be the subject of further considerations in the next sections.

The present form of algorithm is a little extended as against the description presented above, what results from the necessity to introduce numerous limitations and algorithm blocks. As the blocks mentioned are not technically related to the OCT image analysis, they will not be discussed here in detail. However, we encourage the Reader to follow this, apparently, complicated algorithm.

```
pr=0.005;
[mss,nss,waga_p,L5,L6]=HIERARHICALL_STEP(Lm,fun,d,pr);
fg=figure; imshow(Lm); hold on
plot(nss'*d-d/2,mss'*d-d/2,'-*')
```

where function `HIERARHICALL_STEP` is:

```
function
[ynfl_rpe,xnfl_rpe,waga_p,Ld16d,Ldb16z]=HIERARHICALL_STEP(L
m,fun,d,pr)
ynfl_rpe=[]; xnfl_rpe=[]; waga_p=[];
Ld16  = blkproc(Lm,[d d],fun);
fun2=@(x) max(x(:));
Ld16__ = blkproc(Lm,[d d],fun2);
Ld16__=[Ld16__(2:end,:);Ld16__(end,:)];
Ldm16=Ld16__==ones([size(Ld16__,1),1])*max(Ld16__);
for n=1:size(Ld16,2); Ld16(:,n)=mat2gray(Ld16(:,n)); end
Ld16d=zeros(size(Ld16));
for n=1:size(Ld16,2)
      Ld16d(1:end-1,n)=diff(Ld16(:,n)).*Ld16(2:end,n);
end
Ldm16=zeros(size(Ld16d));
for n=1:size(Ld16d,2)
      Ldm16(1:end,n)=Ld16d(1:end,n)==max(Ld16d(1:end,n));
end
Ldb16=Ld16d>pr;
Ldb16=bwmorph(Ldb16,'clean');
figure; imshow(Ldb16,[],'notruesize'); hold on
Ldb16_lab=bwlabel(Ldb16);
Ldb16z=zeros(size(Ldb16_lab));
for et=1:max(Ldb16_lab(:))
    Ldb16i=(Ldb16_lab==et);
    if sum(sum(Ldb16i.*Ldm16))>0
        Ldb16z=Ldb16z|Ldb16i;
```

```
    end
end
Ldb16z=bwmorph(Ldb16z,'clean');
Ldb16_lab2=bwlabel(Ldb16);
L77=zeros(size(Ldb16z));
for iw=1:size(Ldb16z,2)
    L77(:,iw)=bwlabel(Ldb16z(:,iw));
end
if (max(L77(:))<2)&(max(Ldb16_lab2(:))==2)
    Ldb16z=Ldb16;
end
ynfl_rpe=[]; xnfl_rpe=[];
for iu=1:size(Ld16d,2)
if sum(Ldb16z(:,iu))>0
        Ldb16z_lab=bwlabel(Ldb16z(:,iu)|Ldm16(:,iu));
        if max(Ldb16z_lab(:))<=2
            Ldb16z_nr=1:size(Ld16d,1);
            Ldb16z_nr(Ldb16z(:,iu)==0)=[];
            Ld16d_nr=1:size(Ld16d,1);
            Ld16d_nr(Ldb16(:,iu)==0)=[];
            if Ld16d_nr(1)==Ldb16z_nr(end)
                if size(ynfl_rpe,2)>0
                    if min(abs(ynfl_rpe(:,end)-
Ldb16z_nr))<=2
                        if  abs(ynfl_rpe(1,end)-
Ld16d_nr(1))<abs(ynfl_rpe(2,end)-Ld16d_nr(1))

ynfl_rpe=[ynfl_rpe,[Ld16d_nr(1);ynfl_rpe(2,end)]];
                            xnfl_rpe=[xnfl_rpe,[iu;iu]];
                        else

ynfl_rpe=[ynfl_rpe,[Ld16d_nr(1);Ldb16z_nr(end)]];
                            xnfl_rpe=[xnfl_rpe,[iu;iu]];
                        end
                    end
                else

ynfl_rpe=[ynfl_rpe,[Ld16d_nr(1);Ldb16z_nr(end)]];
                    xnfl_rpe=[xnfl_rpe,[iu;iu]];
                end
            else

ynfl_rpe=[ynfl_rpe,[Ld16d_nr(1);Ldb16z_nr(end)]];
                xnfl_rpe=[xnfl_rpe,[iu;iu]];
            end
        else
            et_Ldb16=[];
            for et=1:max(Ldb16z_lab)
```

```
et_Ldb16=[et_Ldb16;[et,max((Ldb16z_lab==et).*Ld16__(:,iu))]
];
            end
            et_Ldb16=sortrows(et_Ldb16,-2);
            if et_Ldb16(2,2)*8>et_Ldb16(1,2)
                if size(ynfl_rpe,2)>0
                    Ld16d_nr2=1:size(Ld16d,1);

Ld16d_nr2(Ldb16z_lab~=et_Ldb16(1,1))=[];
                    if abs(ynfl_rpe(2,end)-
Ld16d_nr2)<abs(ynfl_rpe(1,end)-Ld16d_nr2)

et_Ldb16(et_Ldb16(:,1)>et_Ldb16(1,1),:)=[];
                        et_Ldb16=sortrows(et_Ldb16,-2);
                    else
                        et_Ldb16=sortrows(et_Ldb16,-2);
                    end
                end
            end
            et_Ldb16(3:end,:)=[];
            et_Ldb16=sortrows(et_Ldb16,1);
            Ldb16z_nr=1:size(Ld16d,1);
            if size(et_Ldb16,1)==1
                Ldb16z_nr(Ldb16z_lab~=et_Ldb16(1,1))=[];
            else
                Ldb16z_nr(Ldb16z_lab~=et_Ldb16(2,1))=[];
            end
            Ld16d_nr=1:size(Ld16d,1);
            Ld16d_nr(Ldb16z_lab~=et_Ldb16(1,1))=[];

ynfl_rpe=[ynfl_rpe,[Ld16d_nr(1);Ldb16z_nr(end)]];
            xnfl_rpe=[xnfl_rpe,[iu;iu]];
        end
end
end
```

4.11.3 Reducing the Decomposition Area

The increasing of accuracy and thereby reducing the $A_{m,n}$ area size (Fig. 4-81) – block on L_M image is a relatively simple stage of tomographic image processing with particular focus on the operating speed. It has been assumed that $A_{m,n}$ areas will be sequentially reducing by half in each iteration – down to 1x1 size. The reduction of $A_{m,n}$ area is equivalent to performance of the next stage of lines NFL and RPE position approximation.

The increasing of accuracy (precision) of NFL and RPE lines position determined in the previous iteration is connected with two stages:

- concentration of (m,n) coordinates in the sense of determining intermediate ((m,n) points situated exactly in the centre) values by means of linear interpolation method;

- change of concentrated points position so that they would better approximate the limits sought.

If the first part is intuitive and results only in resampling, the second requires more precise clarifications. The second stage consists in matching individual points to the layer sought. As on the ox axis the image by definition is decomposed and pixel's brightness on the image analysed corresponds to the median value of the original image in window A (Fig. 4-81), the modification of points RPE and NFL position occurs only on the vertical axis. The analysis of individual RPE and NFL points is independent in the sense of dependence on n-1 point position, as was the case in the previous section.
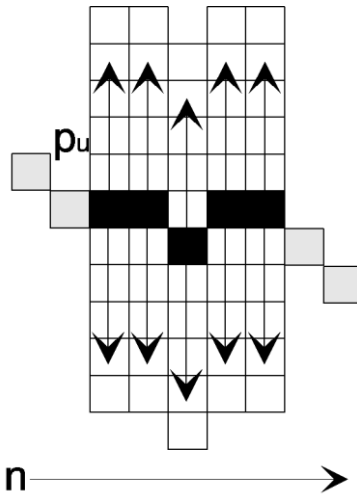
Fig. 4-87 Demonstrative diagram of the process of RPE course matching to the edge of the layer sought. Individual pixels independent of each other may change the position within the $\pm p_u$ range
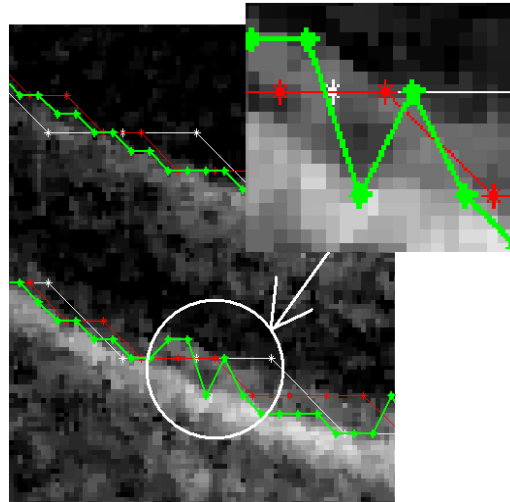
Fig. 4-88 Results of matching for two iterations White colour marks input RPE points and red and green – consecutive approximations

Each of RPE points, left from the previous iteration, and newly created from interpolation, in the consecutive algorithm stages is matched with increasingly high precision to the RPE layer. Point's

RPE(n) position changes within the range of $\pm p_u$ (Fig. 4-87) where the variation range does not depend on the scale of considerations (size of A area) and strictly results from the distance between NFL and RPE (Fig. 4-88). For blocks A of 16x16 to 1x1 size p_u is constant and amounts to 2. This value has been assumed on the basis of, typical for the analysed several hundred L_{GRAY} images, average distance between NFL and RPE, equal to around 32 pixels, what means that after decomposition into blocks A of 16x16 size these are two pixels, that is $p_u=2$. The maximum on the L_{DM} image is sought in this ± 2 range and a new position of RPE or NFL point is assumed for it. Thus the course of RPE or NFL is closer to the actual course of the layer analysed.

The obtained results of matching are presented in Fig. 4-88. White colour shows input RPE values as input data for this stage of algorithm and decomposition into blocks A of size 16 x 16 (L_{DM16} and L_{DB16} images), red colour – results of matching for blocks A of size 8 x 8 (L_{DM8} and L_{DB8} images), and green colour – results of matching for blocks A of size 4 x 4 (L_{DM4} and L_{DB4} images). As may be seen from Fig. 4-88 the next decompositions into consecutive smaller and smaller areas A and thus image of higher resolution, a higher precision is obtained at the cost of time (because the number of analysed RPE, NFL points and their neighbourhoods $\pm p_u$ increases).

This method for A of 16 x 16 size has that good properties of global approach to pixels brightness that there is no need to introduce at this stage additional actions aimed at distinguishing layers situated close to each other (which have not been visible so far due to image resolution). While at areas A of 4x4 size other layers are already visible, which should be further properly analysed. At increased precision, ONL layer is visible, situated close to RPE layer (Fig. 4-88). Thereby in the area marked with a circle there is a high position fluctuation within the oy axis of RPE layer. Because of that the next step of algorithm has been developed, taking into account separation into RPE and ONL layers for appropriately high resolution. In a practical implementation this fragment looks as follows:

```
  function
[mss2,nss2]=HIERARHICALL_PREC(Lm,mss,nss,fun,d,z,pu)
mss=mss*z;
nss=nss*z;
[mss,nss]=HIERARHICALL_DENSE(mss,nss);
Ld16 = blkproc(Lm,[d/z d/z],fun);
Ld16d=zeros(size(Ld16));
for n=1:size(Ld16,2)
        Ld16d(1:end-1,n)=diff(Ld16(:,n));
```

```
end
mss2=[]; nss2=[];
for m=1:size(mss,1)
for n=1:size(mss,2)
    if mss(m,n)~=0 %
        ms2=mss(m,n);
        ns2=nss(m,n);
        m2=ms2+pu;
        m1=ms2-pu;
        if m1<=0; m1=1; end
        if m2>size(Ld16d,1); m2=size(Ld16d,1); end
            mm12=round(m1:m2);
            if ~isempty(mm12)
                Ld16dmm=Ld16d(mm12,ns2);
                mm12(Ld16dmm~=max(Ld16dmm))=[];
                if ~isempty(mm12)
                    mss2(m,n)=mm12(1);
                    nss2(m,n)=ns2;
                end
            end
    end
end
end
```

Where function `HIERARHICALL_DENSE` designed to condense the number of points on determined layers has the following form:

```
function [y_out,x_out]=HIERARHICALL_DENSE(y_in,x_in)
y_out=[0;0]; x_out=[0;0];
y_in(:,x_in(1,:)==0)=[];
x_in(:,x_in(1,:)==0)=[];
for i=1:(size(y_in,2)-1)
    m_1=y_in(1,i:i+1);
    n_12=x_in(1,i:i+1);
    m_2=y_in(2,i:i+1);

x_out(1:2,1:end+length(n_12(1):n_12(2)))=[[x_out(1,:),n_12(
1):n_12(2)];[x_out(2,:),n_12(1):n_12(2)]];
    x_out(:,end)=[];
    if (m_1(2)-m_1(1))~=0
        w1=m_1(1):(m_1(2)-
m_1(1))/(length(n_12(1):n_12(2))-1):m_1(2);
    else
        w1=ones([1 length(n_12(1):n_12(2))])*m_1(1);
    end
    if (m_2(2)-m_2(1))~=0
        w2=m_2(1):(m_2(2)-
m_2(1))/(length(n_12(1):n_12(2))-1):m_2(2);
    else
        w2=ones([1 length(n_12(1):n_12(2))])*m_2(1);
```

```
        end
y_out(1:2,1:end+length(n_12(1):n_12(2)))=[y_out(1:2,:),[w1;
w2]];
        y_out(:,end)=[];
    end
    y_out=y_out(:,2:end);
    x_out=x_out(:,2:end);
```

Hence the function `HIERARHICALL_PREC` is designated to 'match' layers position at any precision.

Both functions – `HIERARHICALL_PREC` and nested `HIERARHICALL_DENSE` – will be used below in the next stages of detected layers approximation to the proper position.

```
z=2;
pu=2;
[mss,nss]=HIERARHICALL_PREC(Lm,mss,nss,fun,d,z,pu);
    plot(nss'*d/z-d/z/2,  mss'*d/z,'-r*')

z=4;
pu=3;
[mss,nss]=HIERARHICALL_PREC(Lm,mss/2,nss/2,fun,d,z,pu);
plot(nss'*d/z-d/z/2,  mss'*d/z,'-g*')
```

The obtained results are shown below in Fig. 4-89 and Fig. 4-90.
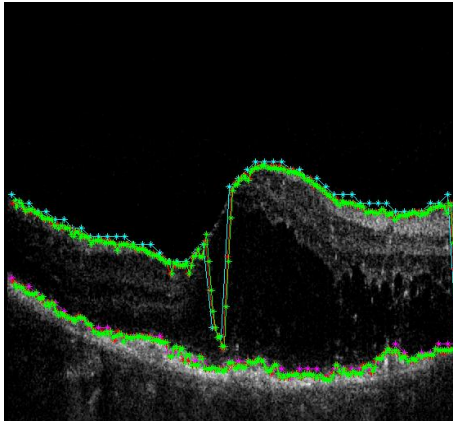

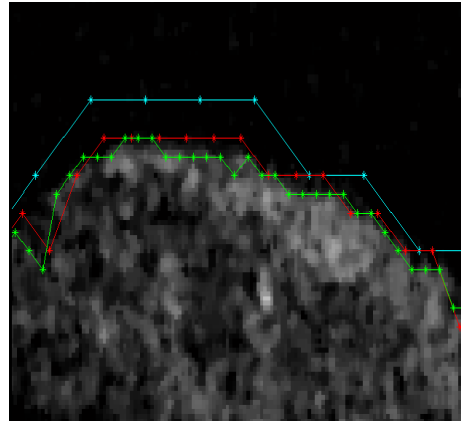**Fig. 4-89 Obtained results of RPE, NFL layers detection on the Lm image**

Fig. 4-90 Obtained results of NFL layer detection on the Lm image – enlargement of Lm image

The results shown in Fig. 4-89 and Fig. 4-90 are not perfect. A visible minimum of NFL layer results from the lack of filtration at the initial stage of y_{NFL} course. Because of that function `HIERARHICALL_MEDIAN`

presented below has been suggested, intended to filtrate using a median filter.

```
function [m_s,n_s]=HIERARHICALL_MEDIAN(mss,nss,Z)
for j=1:size(nss,1)
    for io=1:size(nss,2)
        p=io-round(Z/2); k=io+round(Z/2); if k>size(nss,2);
k=size(nss,2); end; if p<1; p=1; end
        m_s(j,io)=median(mss(j,p:k));
    end
end
n_s=nss;
```

The considerations presented above, related to a hierarchical approach, lead to suggesting the final version of algorithm detecting the ONL, RPE and NFL layers.

```
[Lgray,map]=imread(['D:\OCT\SOURCES\3.bmp']);
Lgray=ind2gray(Lgray,map);
Lgray=double(Lgray)/255;
Lorg=Lgray;
Lm=medfilt2(Lorg,[5 5]);
Lm=mat2gray(Lm);
szer_o=16;
Lm=[Lm(:,1)*ones([1 szer_o]),Lm,Lm(:,end)*ones([1
szer_o])];
fun = @(x) median(x(:));
[mss,nss,waga_p,L5,L6]=HIERARHICALL_STEP(Lm,fun,szer_o,0.03
);
[mss,nss]=HIERARHICALL_PREC(Lm,mss,nss,fun,szer_o,2,2);
[mss,nss]=HIERARHICALL_PREC(Lm,mss/2,nss/2,fun,szer_o,4,3);
[yrpe_onl,xrpe_onl]=HIERARHICALL_MEDIAN(mss(1,:)*4,nss(1,:)
*4,5);
[ynfl,xnfl,Lgr]=HIERARHICALL_PREC2(Lm,mss*4,nss*4,20,20);
xnfl(:,xnfl(1,:)==0)=[];
ynfl(:,ynfl(1,:)==0)=[];
xnfl(:,xnfl(2,:)==0)=[];
ynfl(:,ynfl(2,:)==0)=[];
[ynfl,xnfl]=HIERARHICALL_MEDIAN(ynfl,xnfl,5);
figure; imshow(Lm,'notruesize'); hold on
plot(xnfl',ynfl','LineWidth',2)
plot(xrpe_onl,yrpe_onl,'r','LineWidth',2)
```

where function `HIERARHICALL_PREC2` looks as follows:

```
function
[mss2,nss2,Lgr]=HIERARHICALL_PREC2(Lm,mss,nss,pu,pu2)
[mss,nss]=HIERARHICALL_DENSE(mss,nss);
mss2=[]; nss2=[];
Lgr=[];ngr=[];
for n_=1:size(mss,2);
```

```
    n=round(nss(2,n_));
    m1=round(mss(2,n_))-pu;
    m2=round(mss(2,n_))+pu;
    if m1<1; m1=1; end; if m2>size(Lm,1); m2=size(Lm,1);
end
    Lmn=Lm(m1:m2,n);
    Lmnr2=1:length(Lmn);
    Lmf=[Lmnr2',Lmn];
    Lmf=sortrows(Lmf,-2);
Lmf(Lmf(:,2)<(0.9*Lmf(1,2)),:)=[]; Lmf=sortrows(Lmf,-1);
    Lmnr2=Lmf(1,1);
    nss2=[nss2,n];
    mss2=[mss2,m1+Lmnr2(1)-1];
    m11=m1+Lmnr2(1)-1-pu2;
    m22=m1+Lmnr2(1)-1+pu2;
    if m11<1; m11=1; end; if m22>size(Lm,1);
m22=size(Lm,1); end
    if length(m11:m22)==(pu2*2+1)
        Lmn=Lm(m11:m22,n);
        Lgr=[Lgr,Lmn];
        ngr=[ngr,n_];
    end
end
    Lgr=filter2(ones([3 3]),Lgr)/9;
for n=1:size(Lgr,2)
  po_=Lgr(:,n);
   P = POLYFIT(1:length(po_),po_',5);
  po = POLYVAL(P,1:length(po_));
  dpo=diff(po);
  dpo(round(length(dpo)/2):end)=0;
  dnr=1:length(dpo);
  if max(dpo)>0.03
      dnr(dpo~=max(dpo))=[];
      dnr_=dnr;
   nss2(2,ngr(n))=nss2(1,ngr(n));
   mss2(2,ngr(n))=mss2(1,ngr(n))+dnr-pu2;
      for itt=(n+1):size(Lgr,2)
          po_=Lgr(:,itt);
          P = POLYFIT(1:length(po_),po_',4);
          po = POLYVAL(P,1:length(po_));
          dpo2=diff(po);
          dnr1=dnr-3;
          dnr2=dnr+3;
          if dnr1<1; dnr1=1; end; if dnr2>length(dpo2);
dnr2=length(dpo2); end
          dpo2([1:dnr1,dnr2:end])=0;
          dnr2=1:length(dpo2);
          if max(dpo2)>0
              dnr2(dpo2~=max(dpo2))=[];
              dnr=dnr2(1);
```

```
        nss2(2,ngr(itt))=nss2(1,ngr(itt));
        mss2(2,ngr(itt))=mss2(1,ngr(itt))+dnr-pu2;
                end
            end
                dnr=dnr_;
            for itt=(n-1):-1:1
                po_=Lgr(:,itt);
                P = POLYFIT(1:length(po_),po_',4);
                po = POLYVAL(P,1:length(po_));
                dpo2=diff(po);
                dnr1=dnr-4;
                dnr2=dnr+4;
                if dnr1<1; dnr1=1; end; if dnr2>length(dpo2);
dnr2=length(dpo2); end
                dpo2([1:dnr1,dnr2:end])=0;
                dnr2=1:length(dpo2);
                if max(dpo2)>0
                    dnr2(dpo2~=max(dpo2))=[];
                    dnr=dnr2(1);
        nss2(2,ngr(itt))=nss2(1,ngr(itt));
        mss2(2,ngr(itt))=mss2(1,ngr(itt))+dnr-pu2;
                end
            end
            break
        end
end
```

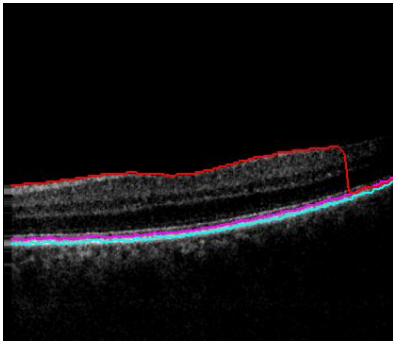The results obtained are shown in Fig. 4-91 and Fig. 4-92.
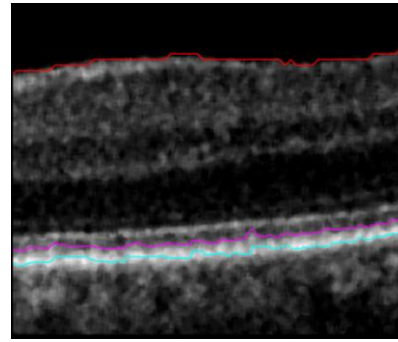

Fig. 4-91 Detected ONL, RPE and NFL layers

Fig. 4-92 Enlargement of detected ONL, RPE and NFL layers from the image aside

4.11.4 Analysis of ONL Layer

This analysis consists in separating line ONL from line RPE originating from previously executed stages of the algorithm. The issue is

facilitated by the fact that on average approx. 80, 90% pixels on each tomographic image have the maximum value in each column exactly at point RPE (this property has been already used in the previous section). So the only problem is to detect the position of ONL line. One of possible approaches consists of an attempt to detect the contour of the layer sought on L_{IR} image. This image originated from L_M image thanks to widening of $y_{RPE}(n)$ layer range within oy axis within the range of $\pm p_I = 20$ pixels. L_{IR} image has been obtained with the number of columns consistent with the number of L_M image columns and with the number of lines $2 \cdot p_I + 1$. Fig. 4-93 shows image $L_{IR} = L_M(m - y_{RPE}(n), n)$ originating from L_M image from Fig. 4-88.
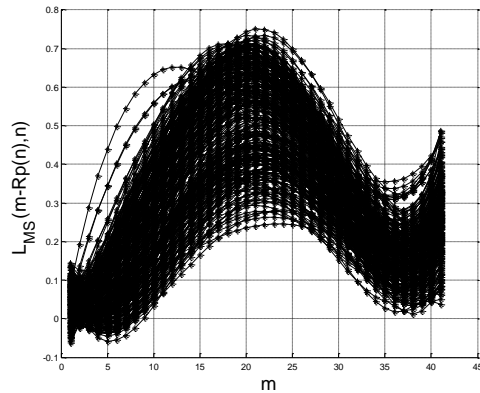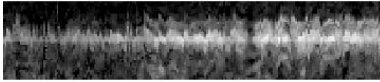

Fig. 4-93 Image $L_{IR} = L_M(m - y_{RPE}(n), n)$ **Fig. 4-94 Courses of $L_{IRS} = L_{MS}(m - y_{RPE}(n), n)$ versus m**

The upper layer visible in Fig. 4-93 as a pretty sharp contour is the sought course of ONL. Unfortunately, because of a pretty high individual variation within the ONL layer position relative to RPE, the selected p_I range in further stages of the algorithm may be increased even twice (that will be described later). To determine consecutive points of ONL layer position interpolations with 4th degree polynomial of grey level degree for individual columns of L_{IR} image obtaining this way L_{IRS}, which changes of grey levels in individual columns are shown in Fig. 4-94. The position of point ONL(n) occurs in the place of the highest gradient occurring within the range $(RPE(n) - p_I) \div RPE(n)$ relative to L_{MS} image or $1 \div p_I$ relative to L_{IRS} image.
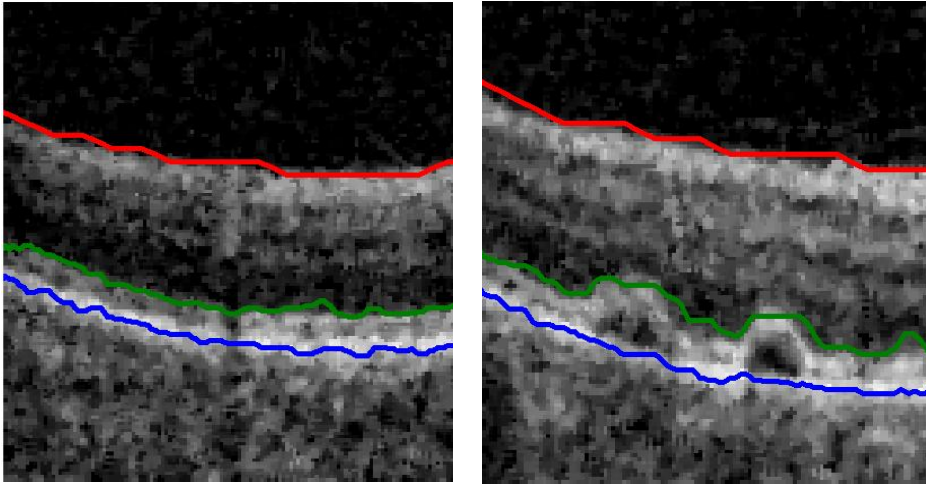
Fig. 4-95 Parts of L_M images with marked courses of NFL – red, RPE – blue, and ONL - green

As may be seen in Fig. 4-95 the method presented perfectly copes with detecting NFL, RPE and ONL layers marked in red, blue and green, respectively.

There is another solution of this problem – presented below.

4.11.5 Determination of the Area of Interest and Preprocessing

Having coordinates for consecutive n-columns, points $y_{NFL}(n)$ and $y_{RPE}(n)$ the area of interest has been determined as the area satisfying the condition $y_{NFL}(n)<y<y_{RPE}(n)$. An example of area L_{GR} originated from the L_M image presented in Fig. 4-3 after filtration using a median filter of 7x7 size (the size was arbitrarily chosen) is shown in Fig. 4-96. The L_{GR2} image is related to a similar fragment of L_M image, but before filtration.

Fig. 4-96 Image L_{GR}

Fig. 4-97 Image L$_{GR2}$

Images presented in Fig. 4-96 and Fig. 4-97 originated from the algorithm

```
yrpe_onl=round(yrpe_onl);
xrpe_onl=round(xrpe_onl);
[yrpe_onl,xrpe_onl]=HIERARHICALL_DENSE2(yrpe_onl,xrpe_onl);
ynfl=round(ynfl(1,:));
xnfl=round(xnfl(1,:));
Lgr=[];
Lgr2=[];
fun2 = @(x) median(x(:))*ones(size(x));
Lmf=blkproc(Lm,[3 3],[3 3],fun2);
m1n2=[];
for ix=1:length(yrpe_onl)
    m1=yrpe_onl(ix); n1=xrpe_onl(ix);
    xynfl=[ynfl',xnfl']; xynfl_=xynfl(xynfl(:,2)==n1,:);
    m1n2(ix,1:2)=[m1,0];
    if ~isempty(xynfl_)
       m2=xynfl_(1,1); n2=xynfl_(1,2);
        Lgr2(1:(m2-m1+1),ix)=Lm(m1:m2,n2);
        Lgr(1:(m2-m1+1),ix)=Lmf(m1:m2,n2);
        m1n2(ix,1:2)=[m1,n2];
    end
end
figure; imshow(Lgr);
figure; imshow(Lgr2);
```

   where function `HIERARHICALL_DENSE2`

```
function [y_out,x_out]=HIERARHICALL_DENSE2(y_in,x_in)
y_out=[0]; x_out=[0];
y_in(:,x_in==0)=[];
x_in(:,x_in==0)=[];
for i=1:(length(y_in)-1)
    m_1=y_in(i:i+1);
    n_12=x_in(i:i+1);

x_out(1:end+length(n_12(1):n_12(2)))=[x_out(:)',n_12(1):n_1
2(2)];
    x_out(:,end)=[];
    if (m_1(2)-m_1(1))~=0
```

```
        w1=m_1(1):(m_1(2)-m_1(1))/(length(n_12(1):n_12(2))-
1):m_1(2);
    else
        w1=ones([1 length(n_12(1):n_12(2))])*m_1(1);
    end
    y_out(1:end+length(n_12(1):n_12(2)))=[y_out(:)',[w1]];
    y_out(end)=[];
end
y_out=y_out(2:end);
x_out=x_out(2:end);
```

The first stage of algorithm operation is sequential performance of convolution with mask h, i.e.:

$$h\big(m_h, n_h, \theta = 0\big) = \begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \tag{59}$$

for angles θ from the range 80° to 100°, every 1°.

$$L_{SGR}\big(m, n, \theta\big) = \sum_{m_h=-M_h/2}^{M_h/2} \sum_{n_h=-N_h/2}^{N_h/2} L_{GR}\big(m+m_h, n+n_h\big) \cdot h\big(m_h, n_h, \theta\big) \tag{60}$$

$$L_{GGR}\big(m, n\big) = \max_{\theta \in (80,100)} \big(L_{SGR}\big(m, n, \theta\big)\big) \tag{61}$$

where m – row, n – column, θ – angle of mask h rotation, M_h, N_h – number of mask h rows and columns.

This fragment implementation is presented below:

```
t=-4:1:4; f=OCT_GAUSS(t,1); f=f/max(f(:)); f=f*(4+1)-
abs(2);
h=ones([9 1])*f;
h=imresize(h,[3 3],'bicubic');
h(:,round(size(h,2)/2):end)=max(h(:));
h=imresize([-2 -2 0 2 2],[15 5],'bicubic');
Lggr=zeros(size(Lgr));
Lphi=zeros(size(Lgr));
for phi=-100:10:-80
    h_=imrotate(h,phi,'bicubic');
    Lsgr=conv2(Lgr,h_,'shape');
    Lpor=Lggr>Lsgr;
    Lphi=Lpor.*Lphi+(~Lpor)*phi;
    Lggr =max(Lggr,Lsgr);
```

```
end
figure
imshow([mat2gray(Lggr)]);
figure;
imshow(Lggr,[0 0.5])
```

where `OCT_GAUSS`:

```
function y = OCT_GAUSS(x,std)
y = exp(-x.^2/(2*std^2)) / (std*sqrt(2*pi));
```

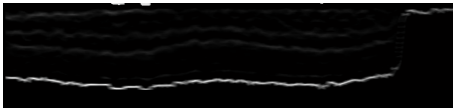The resultant images are shown in Fig. 4-98 and Fig. 4-99.

Fig. 4-98 Image L$_{GGR}$

Fig. 4-99 Image L$_{GGR}$ after normalisation to [0 0.5] range

The range of θ angle values was selected because of the position of layers sought, which in accordance with medical premises should be 'nearly' parallel with small angular deviations. Because each pathology featuring a significant angular change of $y_{NFL}(n)$ and $y_{RPE}(n)$ layers will be corrected after the conversion to the L_{GR} image. The methodology for consecutive convolutions performance (60) for successively changing θ angle values and then the calculation of the maximum occurring for consecutive resultant images (61) was described in detail in [25] and [40]. The created resultant image L_θ obtained on the basis of code presented above and:

```
figure; imshow(Lphi,[-100 -80]); colormap('jet'); colorbar
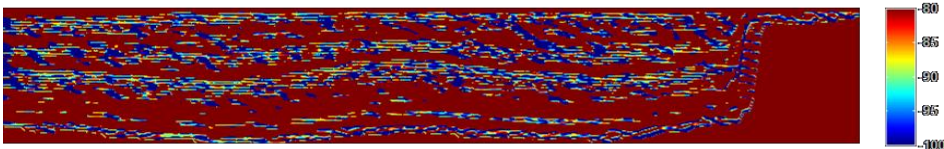```

is shown in Fig. 4-100.



**Fig. 4-100 Image L$_\theta$**

The division into individual layers consists here in tracking changes of individual points position of individual layers changing their position for consecutive n-columns of the $L_{GGR}$ image. This issue is not a trivial one, mainly due to difficulties in identification of both (in a general case) of

the number of layers visible and due to the lack of their continuity and also very often due to their decay because of e.g. existing shadows [2], [4], [18]. These issues are illustrated by the graph of changes of $L_{GGR}$ image grey level changes presented in Fig. 4-101. The change of grey levels has been marked in red and in green for consecutively occurring columns on the $L_{GGR}$ image (for example for presented n=120 and 121). The tracking consists here in suggesting a method to connect individual peaks of courses presented, what will happen in the next section.

## 4.11.6  Layers Points Analysis and Connecting

The localisation and determination of layer position, having NFL, RPE and ONL layers, is one of the most difficult issues. The graph shown in Fig. 4-101 clearly confirms this presumption.

In the first stage it is necessary to find the maximums positions on the graph from Fig. 4-101. To this end the following operation was carried out:

$$L_{UGR}(m, n) = L_{GGR}(m, n) - L_{SR}(m, n) \tag{62}$$

where $L_{SR}$ is the image originated as a result of $L_{GGR}$ image filtration using an averaging filter of mask with experimentally chosen 9x9 size, i.e.:

```
Lgr=(Lggr-conv2(Lggr,ones(9),'same')/81);
```

The procedure enables cutting out the unevenness of lighting visible on the image Fig. 4-96 and thereby on the graph from Fig. 4-101. The graph of the same range of rows and columns, i.e. n=120 and n=121 for m∈(5,35) of the L_{UGR} image is shown in Fig. 4-101.

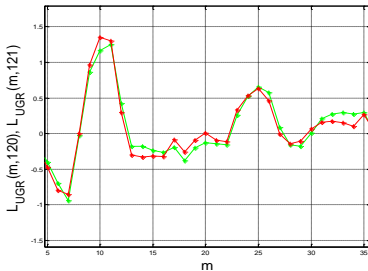**Fig. 4-101 Examples of grey level changes for n=120 – red and n=121 – green colour of L_{UGR} image for m∈(5,35)**
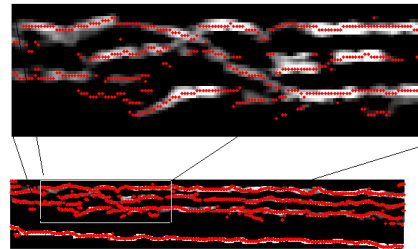

Fig. 4-102 Image L_{UGR} with marked points p(i,n) of the maximum position for consecutive areas and its enlargement

The implementation in Matlab of the course described looks as follows:

```
figure;
    plot(Lggr(:,121),'-g*'); grid on; hold on
    plot(Lggr(:,121-1),'-r*'); hold off
ylabel('L_{GGR}(m,120), L_{GGR}(m,121)','FontSize',20)
xlabel('m','FontSize',20)
Lugr=(Lggr-conv2(Lggr,ones(9),'same')/81);
figure;
    plot(Lugr(:,121),'-g*'); grid on; hold on
    plot(Lugr(:,121-1),'-r*'); hold off
ylabel('L_{UGR}(m,120), L_{UGR}(m,121)','FontSize',20)
xlabel('m','FontSize',20)
```

Points $p(i,n)$ (where i – index of a consecutive point in the nth column) are shown in Fig. 4-102 on the L_{UGR} image. The position of individual $p(i,n)$ points for the L_{UGR} image was determined based on the method of finding consecutive maximum values for binary columns (the decimal to binary conversion threshold was set at 0). The source code responsible for this part is presented below:

```
      figure
imshow(Lugr); hold on
for n=1:size(Lugr,2)
    Lnd=Lugr(:,n);
    Llab=bwlabel(Lnd>0.01);
    Lnr=1:length(Llab);
    for io=1:max(Llab)
       Lnd_=Lnd;
       Lnd_(Llab~=io)=0;
       Lnrio=Lnr(Lnd_==max(Lnd_(:)));
       plot(n,Lnrio(1),'.r')
    end
end
```
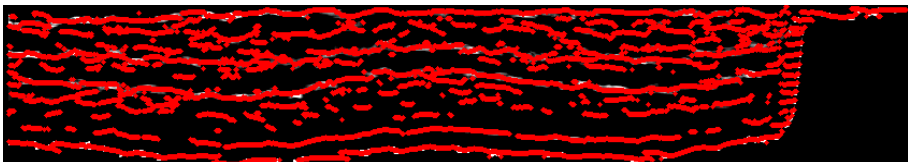
The image generated is shown below

Fig. 4-103 L_{UGR} image with marked p(i,n) points

The following assumptions were made in the process of individual $p(i,n)$ points connecting:

- p_{zx} – parameter responsible for permissible range of points connecting (analysing) on the ox axis,
- p_{zy} – parameter responsible for permissible range of points connecting (analysing) on the oy axis,
- p_{zc} – parameter responsible for permissible range on the ox axis, where the optimum connection points are sought.
- each new point, if it does not fulfil the assumptions on p_{zx} and p_{zy} distance, is assumed as the first point of a new layer,
- each point may belong to only one line, what by definition limits a possibility of lines division or connection.

As an illustration the process of connecting for typical and extreme cases is shown below (Fig. 4-104).

a)　　　　　b)　　　　　c)　　　　　d)

e)　　　　　　　　　　f)
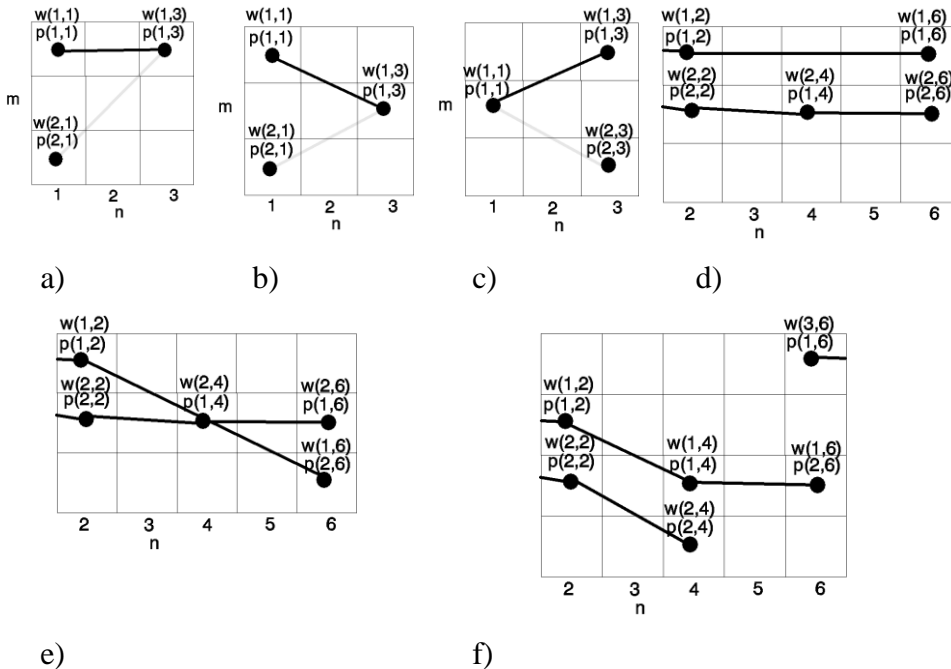
Fig. 4-104 Demonstrative diagrams of typical and extreme cases of individual layers' p(i,n) points connecting. Results are shown for parameters $p_{zx}=2$, $p_{zy}=2$, $p_{zc}=6$

Fig. 4-104 shows demonstrative diagrams of typical and extreme cases of p(i,n) points connecting into lines marked as w(j,n), where j – is the line number and n – column. Fig. 4-104 a) shows a typical case, where having two points p(1,1) and p(2,1) because of a smaller distance on the oy axis p(1,1) was connected with p(1,3). Fig. 4-104 b) shows a reverse

more difficult situation as compared with Fig. 4-104 a), because points p(1,1) and p(2,1) are equidistant. In this case, because points p(i,n) for each column are determined top-down, this connection will be carried out between p(1,1) and p(1,3). Fig. 4-104 c) shows a similar situation to Fig. 4-104 b). In Fig. 4-104 d) the system of connections is visible for the case, where there are points of discontinuity in determination of points comprised by individual layers. Parameter p_{zc} is responsible for that. In the case of Fig. 4-104 e) there was an erroneous lines crossing. Points p(2,2), p(1,4) and p(2,6) were properly connected, while point p(1,2) was improperly connected with p(2,6). Such action results in adopting a principle of connecting with the nearest point and in a too large range of p_{zc} parameter values, which in this case 'allowed' connecting p(1,2) and p(2,6). Fig. 4-104 f) is a typical example, where the line formed from points p(2,2) and p(2,4) ends and a new line starts from point p(1,6). This example is interesting to the extent that if parameters p_{zx}, p_{zx} and p_{zx} would allow that, as a result lines created from points p(1,2), p(1,4) and p(1,6) should be obtained as well as the second line p(2,2), p(2,4) and p(2,6). Obviously, having only such data (p(i,n) points coordinates) it is not possible to determine, which solution is the right one. Situations presented in Fig. 4-104 a), b) and c) have another significant feature, by definition they do not allow individual analysed layers (Fig. 4-104 a), b)) to be connected and to be divided (Fig. 4-104 c)).

For parameters $p_{zx}=2$, $p_{zy}=2$, $p_{zc}=6$ and points p(i,n) of L_{UGR} image shown in Fig. 4-102 the following results were obtained - Fig. 4-105.
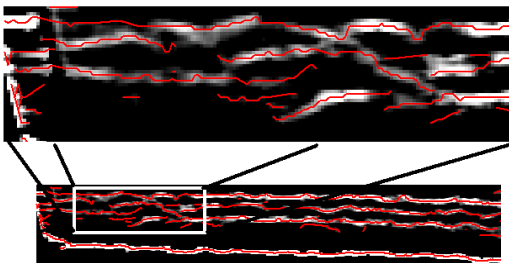

Fig. 4-105 Image L_{UGR} with marked grouped p(i,n) points for parameters $p_{zx}=2$, $p_{zy}=2$, $p_{zc}=6$ and its enlargement
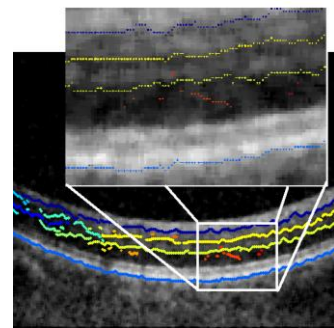

Fig. 4-106 Image L_M and its enlargement with marked groups of connected p(i,n) points for consecutive j[th] w(j,n) lines

The implementation of the discussed algorithm fragment is presented below. The Reader should be familiar with the first part from the previous implementation, i.e.:

```
figure
imshow(Lugr); hold on
rr_d_o=0;
rr_u_o=0;
r_pp=[];
rrd=[]; rru=[];
rrd_pol=[];rrd_nr=[];
rrd_pam=[];
rru_pam=[];
for n=1:size(Lugr,2)
    Lnd=Lugr(:,n);
    Llab=bwlabel(Lnd>0.01);
    Lnr=1:length(Llab);
    rr_d=[];
    for io=1:max(Llab)
       Lnd_=Lnd;
       Lnd_(Llab~=io)=0;
       Lnrio=Lnr(Lnd_==max(Lnd_(:)));
       rr_d=[rr_d,Lnrio(1)];
    end
…
```

Instead, in the second part there is the right part of described problem solution, i.e.:

```
…
pzc=10;
pzy=4;
    rrd_pol(1:length(rr_d),n)=rr_d;
if n==1
    rrd_nr(1:length(rr_d),n)=(1:length(rr_d))';
else
    rrd_nr(1:length(rr_d),n)=0;
end
wu=[]; wd=[];
wuiu=[]; wdiu=[];
rrd(1:length(rr_d),n)=rr_d;
rr_dpp=rr_d;
for ni=(n-1):-1:(n-pzc)
    if ni>0
            rr_d=rrd(:,n);
            rr_d_o=rrd(:,ni);
            rrd_nr_iu=rrd_nr(:,ni);
            if (~isempty(rr_d))&(~isempty(rr_d_o))
                uu=ones([length(rr_d) 1])*rr_d_o';
                nrnr=ones([length(rr_d) 1])*rrd_nr_iu';
```

```
                dd=rr_d*ones([1 length(rr_d_o)]);
                ww=ones([size(dd-uu,1) 1])*min(abs(dd-
uu))==abs(dd-uu);
                ww_=min(abs(dd-uu),[],2)*ones([1 size(dd-
uu,2)])==abs(dd-uu);
                ww=ww_.*ww;
                ww(abs(dd-uu)>pzy)=0; ww(dd==0)=0;
ww(uu==0)=0; ww(rr_d==0,:)=0; ww(:,rr_d_o==0)=0;
                wu_=ww.*uu; wu_(wu_==0)=[]; wu=[wu,wu_];
                wd_=ww.*dd; wd_(wd_==0)=[]; wd=[wd,wd_];
                wuiu_=ones(size(wu_))*(ni);
wuiu=[wuiu,wuiu_];
                wdiu_=ones(size(wd_))*(n);
wdiu=[wdiu,wdiu_];
                nrnr=sum(nrnr.*ww,2);
                nrnrw=sum(ww,2);
                niu=max(rrd_nr(:))+1;
                for gf=1:length(nrnr)
                    if (nrnr(gf)==0)&&(nrnrw(gf)==1)
                            nrnr(gf)=niu;
                            wvv=ww(gf,:);
                            rrd_nr(wvv==1,ni)=niu;
                            niu=niu+1;
                    end
                end
                rpnr=rrd_nr(:,n);  rpnr=rpnr+nrnr;
rrd_nr(:,n)=rpnr;
                rr_d(sum(ww,2)~=0)=0;
                rr_d_o(sum(ww,1)~=0)=0;
                rrd(1:length(rr_d),n)=rr_d;
                rrd(1:length(rr_d_o),ni)=rr_d_o;

            end
    end
end
rrd(1:length(rr_dpp),n)=rr_dpp;
for j=1:length(wu)
    line([wuiu(j)  wdiu(j)],[wu(j)
wd(j)],'LineWidth',2,'Color','r')
end
n
end
```

Fig. 4-106 shows the arrangement of individual j[th] w(j,n) lines on the input image L_M. Instead, Fig. 4-107 shows other results of points p(i,n) grouping for parameters $p_{zx}=2$, $p_{zy}=2$, $p_{zc}=6$ at other L_{UGR} images.
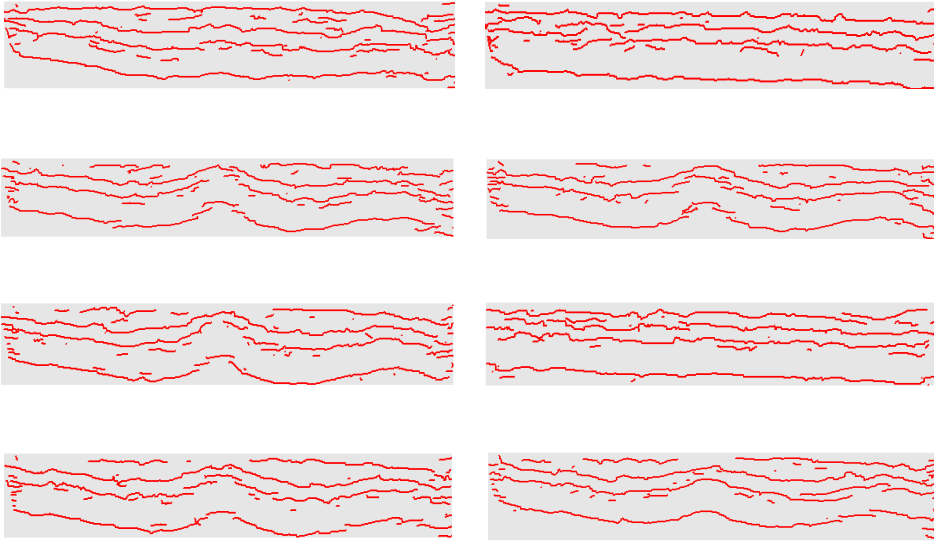
Fig. 4-107 Example L$_{UGR}$ images with marked grouped p(i,n) points for parameters p$_{zx}$=2, p$_{zy}$=2, p$_{zc}$=6 and its enlargement

Two characteristic elements may be noticed. The first of them is related to the existence of short lines, which are a disturbance (short is understood here as such, which are not longer than 10, 20 points). The second characteristic element is the determination of transition borders (looking in the sequence of rows occurrence – top-down) by a lighter and darker area. This is caused by an asymmetric form of mask h (59). Hence a supplementary approach consists of performance of operations presented starting from the relationship (59) for the suggested h but for angles θ from the range -80° to -100° every 1°. The results obtained are presented below (Fig. 4-108).
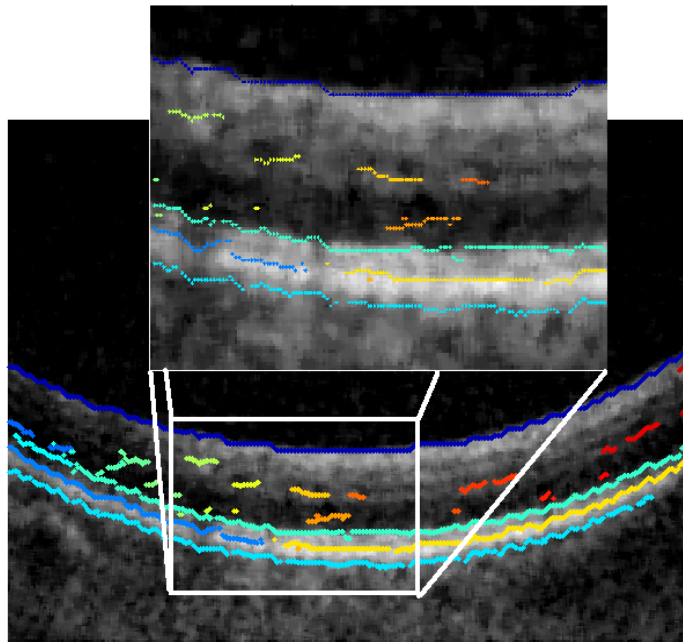
**Fig. 4-108 Image L_M and its enlargement with marked groups of
connected p(i,n) points for consecutive j^{th} w(j,n) lines at h for θ angles
from the -80° to -100° range**

Further on, denoting w(j,n) lines obtained for h rotated within θ angles
from the range -80° to -100° as $w_1(j_1,n)$ and w(j,n) lines obtained for h
rotated within θ angles from the range 80° to 100° as $w_2(j_2,n)$, the
following operations were performed:

- the location of last p(i,n) points positions of consecutive $w_1(j_1,n)$ and
  $w_2(j_2,n)$ lines has been checked,

- the approximation by the second degree polynomial of the last points
  of $w_1(j_1,n)$ and $w_2(j_2,n)$ lines was carried out,

- it has been checked, whether the obtained next points extending the
  analysed line j_1* connect with another line $j_1 \neq j_1*$ (or similarly $j_2 \neq j_2*$).

These operations have been precisely described in the next section.

4.11.7 Line Correction

The determined $w_1(j_1,n)$ and w(j,n) lines are shown as an example in
Fig. 4-108. The lines correction consists in connecting them, provided
that the extension of consecutive points of the approximated line

coincides in a specific range with the beginning of the next one. The following assumptions were made in the process of individual $w(i,n)$ lines connecting:

- P_{kx} – parameter responsible for permissible range of lines connecting (analysing) on the ox axis,
- P_{ky} – parameter responsible for permissible range of lines connecting (analysing) on the oy axis,
- p_{kc} – parameter responsible for the range on the ox axis, in which the line end is approximated,
- p_{ko} – parameter responsible for the size of ox axis analysis window,
- the process of lines connecting applies only to those, which end and start – branches connecting is not carried out,
- only those lines are connected, which have minimum 90% of analysed points falling within the range $\pm p_{ky}$ with respect to the approximated line (Fig. 4-109),
- lines connection consists in changing their labels – in the case of connecting e.g. $w(1,n)$ with $w(2,n)$ lines, the label is changed from '2' to '1'.

The presented methodology works pretty well for tested image resolutions in the case, when the approximation is carried out using a first or second degree polynomial and when the following values of parameters are assumed $p_{kx}=20$, $p_{ky}=4$, $p_{kc}=10$, $p_{ko}=10$. The obtained example results for the last two points (marked - wa'), three last points (marked – wa'') and four last points (marked – wa''') are shown in Fig. 4-110.

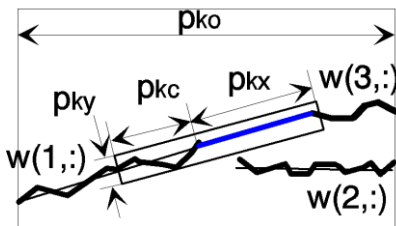**Fig. 4-109 Demonstrative lines correction diagram with marked algorithm parameters p_{kc}, p_{kx}, p_{ky} and p_{ko}**
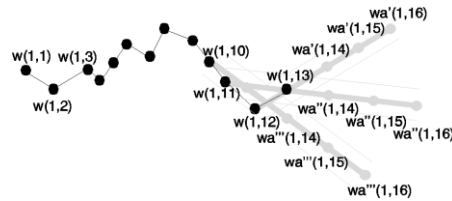
Fig. 4-110 Demonstrative diagram of lines approximation results using order 1 polynomial for different numbers of end points
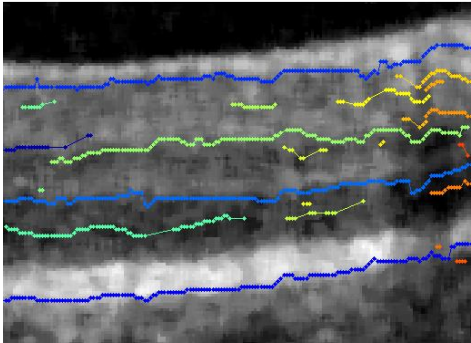
Fig. 4-111 Image fragment before lines correction
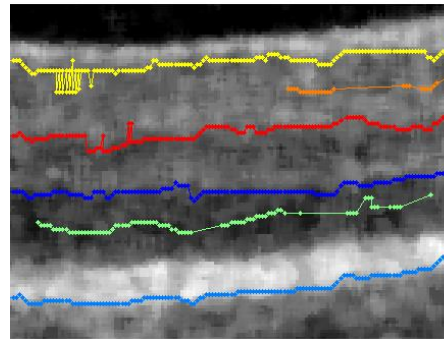
Fig. 4-112 Image fragment after lines correction obtained for parameters $p_{kx}=20$, $p_{ky}=4$, $p_{kc}=10$, $p_{ko}=10$

A direct relationship between obtaining correct results from $w(j,n)$ lines connecting and the number of analysed points at their end is visible from the results obtained at the initial analysis of approximation results. In particular, when allowing connecting lines, which – looking at the x axis – have the same values, a situation shown in Fig. 4-113 may occur.

Fig. 4-113 Result of connecting lines overlapping each other for a few pixels with regard to the ox axis

As this fragment implementation in Matlab is trivial, we leave this part to be written by the Reader.

4.11.8 Layers Thickness Map and 3D Reconstruc

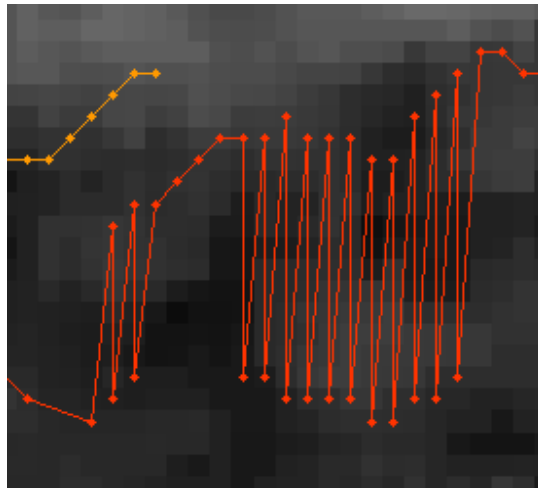The analysis of L_M images sequence and precisely the acquiring of layers NFL, RPE and ONL allows performing 3D reconstruction and layers thickness measurement. A designation for an image sequence with an upper index (i) has been adopted, where i = {1,2,3,...,k-1,k} i.e. $L_M^{(1)}$, $L_M^{(2)}$, $L_M^{(3)}$,.., $L_M^{(k-1)}$, $L_M^{(k)}$. For a sequence of 50 images the position of NFL layers (Fig. 4-114), RPE (Fig. 4-115) and ONL (Fig. 4-116) was measured as well as ONL - RPE layer thickness (Fig. 4-117).
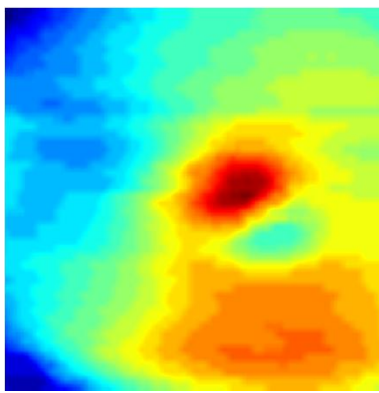
Fig. 4-114 NFL spatial position
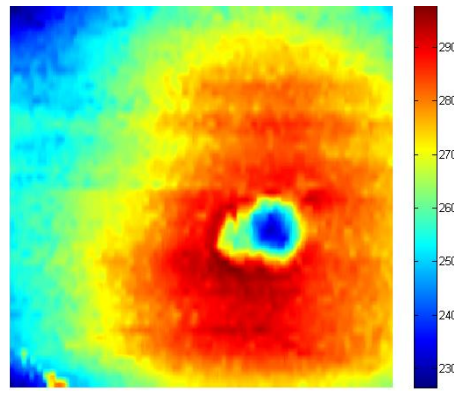
Fig. 4-115 RPE spatial position
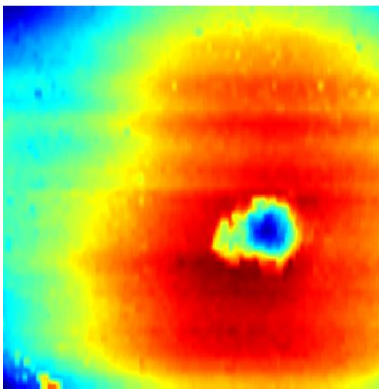
Fig. 4-116 ONL spatial position
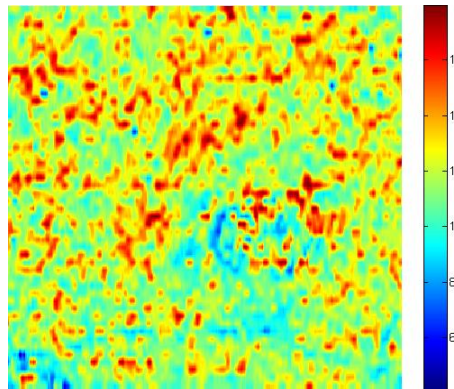
Fig. 4-117 ONL-RPE layer thickness

3D reconstruction performed based on $L_M^{(i)}$ images sequence is the key element crowning the results obtained from the algorithm suggested. The sequence of images, and more precisely the sequence of $NFL^{(i)}(n)$, $RPE^{(i)}(n)$ and $ONL^{(i)}(n)$ layers position, provides the basis for 3D reconstruction of a tomographic image. For an example of 50 images

sequence and one image resolution $L_M^{(i)}$ at the level of MxN = 256x512, a 3D image is obtained composed of three layers NFL, RPE and ONL of 50x512 size. The results are shown in Fig. 4-118 for an example of original images reconstruction (without the sample described above) based on i pixels brightness Fig. 4-119 – reconstruction performed using the algorithm described above, on the basis of $NFL^{(i)}(n)$, $RPE^{(i)}(n)$ and $ONL^{(i)}(n)$ information.
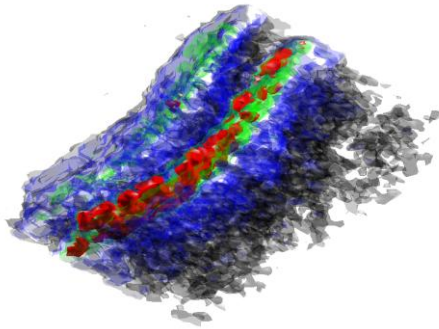
**Fig. 4-118 Example of 3D reconstruction of layers NFL and ONL – green, RPE - red**
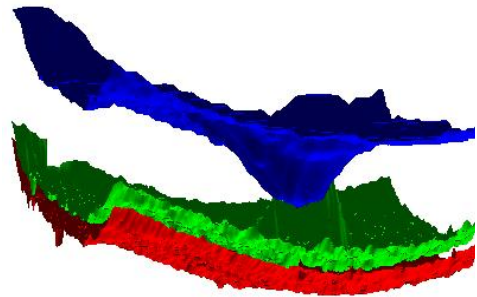
Fig. 4-119 Example of 3D reconstruction of layers NFL – blue, RPE – red and ONL – green

In an obvious way a possibility of automatic determination of the thickest or the thinnest places between any points results from layers presented in Fig. 4-119.

4.11.9 Evaluation of Hierarchical Approach

The algorithm presented, after a minor time optimisation, detects NFL, RPE and ONL layers with up to a few dozen milliseconds on a computer with a 2.5GHz Intel Core 2 Quad processor. The time was measured as an average value of 700 images analysis dividing individual images into blocks A (Fig. 4-81) of consecutive sizes 16x16, 8x8, 4x4, 2x2. This time may be reduced by the modification of approximation blocks number and at the same time increasing the layer position identification error – results are shown in the table below (Tab 4-1).

Tab 4-1 Percentage execution time of algorithm for NFL, RPE and ONL layers detection

Processing stage	Total time since processing start [%]
Preprocessing	20
Initial breakdown into NFL and RPE+ONL	26
NFL and RPE+ONL approximation for A – 16x16	32
NFL and RPE+ONL approximation for A – 8x8	46
Accurate RPE and ONL breakdown	100

The specification of individual algorithm stages' analysis times presented in the table above clearly shows the longest execution of the first stage of image preprocessing, where filtration with a median filter is of prevailing importance (in terms of execution time) as well as of the last stage of precise determination of RPE and ONL layers position. Because precise RPE and ONL breakdown is related to the analysis and mainly to the correction of RPE and ONL points position in all columns of the image for the most precise approximation (because of a small distance between RPE and ONL it is not possible to perform this breakdown in earlier approximations). So the reduction of computation times may occur only at increasing the error of layers thickness measurement. And so for example for the analysis in the first approximation for A of 32 x 32 size and then for 16 x 16 gross errors are obtained generated in the first stage and duplicated in the next ones. The greatest accuracy is obtained for approximations of A of 16x16 size, and then of 8x8, 4x4, 2x2 and 1x1, however the computation time nearly doubles.

4.12 Evaluation and Comparison of Suggested Approaches Results

The methods presented: classical, Canny, random [28] or hierarchical [27] give correct results at the detection (recognition) of RPE, IS/OS, NFL or OPL layers on a tomographic eye image. Differences in the methods proposed are visible only when comparing their effectiveness in the analysis of mentioned several hundred tomographic images. When comparing the methods mentioned it is necessary to consider the accuracy of layer recognition, algorithm responses to pathologies and

optic nerve heads and the operating speed, in this case for a computer (P4 CPU 3GHz, 2GB RAM).

The following table Tab 4-2 presents a cumulative comparison of algorithms proposed and Tab 4-3 a comparison of results obtained using the algorithms discussed, taking into account typical and critical fragments of individual algorithms operation.

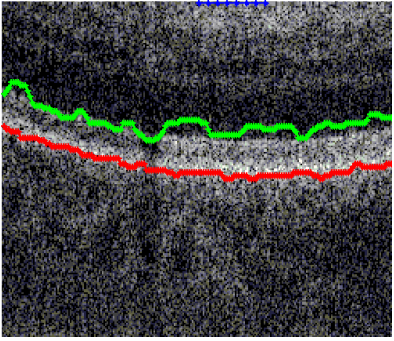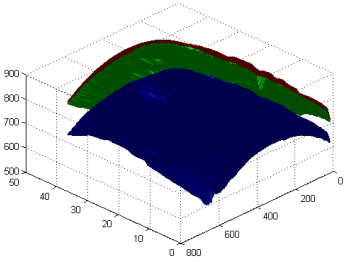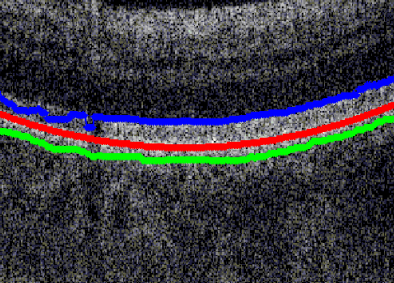**Tab 4-2 Cumulative comparison of algorithms proposed**
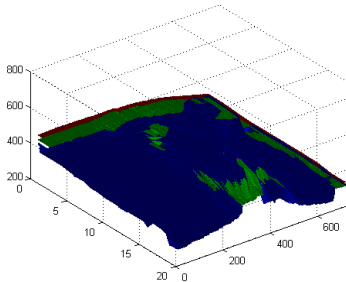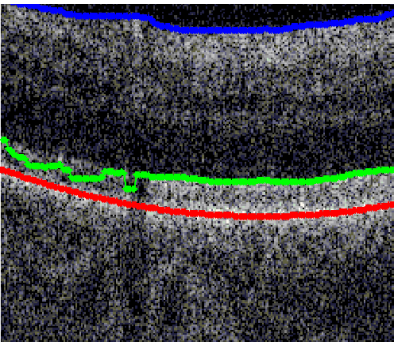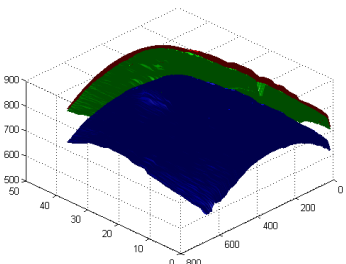
Algorithm/Feature	classical	Canny	random	hierarchical
Total error in layers recognition	5%	4%	7%	2%
Speed of RPE layer recognition – MATLAB	15 s	5s	10s	1s
Speed of RPE layer recognition – C++	0.85 s	0.27s	1.2s	50ms

The random method described as an example in this monograph gives correct results at contours determination (layers separation) both on OCT images as well as on others, for which classical methods of contours determination do not give results or the results do not provide a continuous contour. The algorithm drawbacks include a high influence of noise on the results obtained. This results from a relationship that the number of pixels of pretty high value, resulting from a disturbance, increases the probability of selecting in this place a starting point and hence a component contour. The second drawback is the computations time, which is the longer the larger is the number of selected points and/or the reason, for which searching for the next points $o_{i,j+1}$ was stopped.

The specification of hierarchical algorithm individual stages' analysis times presented in the table above clearly shows the longest execution of the first stage of image preprocessing, where filtration with a median filter is of prevailing importance (in terms of execution time) as well as of the last stage of precise determination of RPE and IS/OS layers position. Because precise RPE and IS/OS breakdown is related to the analysis and mainly to the correction of RPE and IS/OS points position in all columns of the image for the most precise approximation (because of a small distance between RPE and IS/OS it is not possible to perform this breakdown in earlier approximations). So the reduction of computation times may occur only at increasing the error of layers thickness measurement. And so for example for the analysis in the first

approximation for A of 32 x 32 size and then for 16 x 16 gross errors are obtained generated in the first stage and duplicated in the next ones. The greatest accuracy is obtained for approximations of A of 16x16 size, and then of 8x8, 4x4, 2x2 and 1x1, however the computation time nearly doubles.

Tab 4-3 of results obtained using algorithms discussed

Method	*Case of wrong recognition – resulting from specific method nature*	*Example of 3D reconstruction of layers NFL – blue, RPE – red and IS/OS – green*
classical		
Canny		
random		

3D reconstruction performed based on $L_M^{(i)}$ images sequence is the key element crowning the results obtained from the algorithm suggested. The sequence of images, and more precisely the sequence of $y_{NFL}^{(i)}(n)$, $y_{RPE}^{(i)}(n)$ and $y_{IS/OS}^{(i)}(n)$ layers position, provides the basis for 3D reconstruction of a tomographic image. For an example sequence of 50 images and one $L_M^{(i)}$ image resolution of MxN= 256 x 512 a 3D image is obtained, composed of three NFL, RPE and IS/OS layers of 50 x 512 size. Results are shown in Fig. 4-118 for an example reconstruction of original images (without processing described above) based on pixels brightness and in Fig. 4-119 – the reconstruction performed using the algorithm described above was carried out based on $y_{NFL}^{(i)}(n)$, $y_{RPE}^{(i)}(n)$ and $y_{IS/OS}^{(i)}(n)$ information.

5 SUMMARY

The considerations presented confirm the thesis that it is possible to develop a fully automated IT tool assisting doctor's work. The algorithms presented in fragments provide a foundation for their further modifications and profiling for a specific OCT instrument. These modifications should comprise not only the selection of algorithm parameters but also a change of spatial or colour resolution. It is not excluded that a correction of function responsible for reading a DICOM image will be possible. All the corrections mentioned already constitute a marginal contribution as compared with development and testing of a specific solution – what has been presented in this monograph. However, it is necessary to remember that the field of image analysis and processing has been developing very dynamically and with time better and faster, than presented here, methods for OCT images analysis and processing will be appearing. Despite that the authors hope that this monograph will be helpful to Readers not only during developing applications assisting doctors in OCT images diagnostics, but also will provide a basis to develop new original algorithms.

The most recent version of the monograph will be always available for downloading from the site http://robert.frk.pl under 'books' bookmark.

In addition, examples of algorithms presented in this monograph including test images are displayed on this site.

6 SUPPLEMENT

On the http://robert.frk.pl website under 'books' bookmark also source images and the source code presented in this monograph are available, apart from the most recent monograph version.

The source images are placed in the **images.zip** file, which the Reader must unzip onto disk D:/, to the main directory. Matlab files located in the **sources.zip** archive should be unzipped to any directory, to which the access path should be given in the Matlab package. Matlab files (m files) have been specified below; character 'c' at the beginning of a file name stands for the content described in specific section, while missing character 'c' stands for a function, also included in the text (Fig. 6-1).

```
├──READ_DICOM
│      OCT_head_read.m
│      OCT_unzip.m
│      cread_head2.m
│      cread_head.m
├──ANTERIOR
│      cOCT_angle_3D.m
│      cOCT_angle_3D_2.m
│      cOCTplot_AOD_error.m
│      cOCTread_oct_angle.m
│      cOCTreferencje_method.m
│      OCT_activ_cont.m
│      OCT_angle_line.m
│      OCT_edge_inside.m
└──POSTERIOR
   ├──STANDARD
   │      OCT_hole.m
   │      OCT_NFL.m
   │      OCT_corr_line.m
   │      OCT_ALL_GLOBAL.m
   │      OCT_global_line.m
   │      OCT_NFL_artyfic_noise.m
   │      OCT_NFL_line_end.m
   │      OCT_NFL_line.m
   │      OCT_areaa.m
   │      OCT_global_line_mod.m
   │      OCT_activ_cont_noise.m
   │      cOCT_fundus_filtr.m
   │      OCT_activ_cont_other.m
   │      cOCT_fundus_artyfical_pic.m
   ├──NOISE
   │      cOCT_NOISE_artyfical.m
   │      OCT_NOISE_gauss.m
   │      OCT_NOISE_area.m
   │      OCT_NOISE_line.m
   │      cOCT_NOISE_real.m
   │      cOCT_NOISE_artyfical_param.m
   ├──CANNY
   │      cOCT_CANNY_FUN.m
   │      OCT_COR_LINE.m
   │      cOCT_COR_artifical.m
   └──HIERARHICALL
          cOCT_HIERARHICALL_RESIZE.m
          HIERARHICALL_STEP.m
          HIERARHICALL_DENSE.m
          HIERARHICALL_PREC.m
          HIERARHICALL_MEDIAN.m
          HIERARHICALL_PREC2.m
          HIERARHICALL_DENSE2.m
          OCT_GAUSS.m
          cOCT_HIERARHICALL_RPE_NFL.m
```
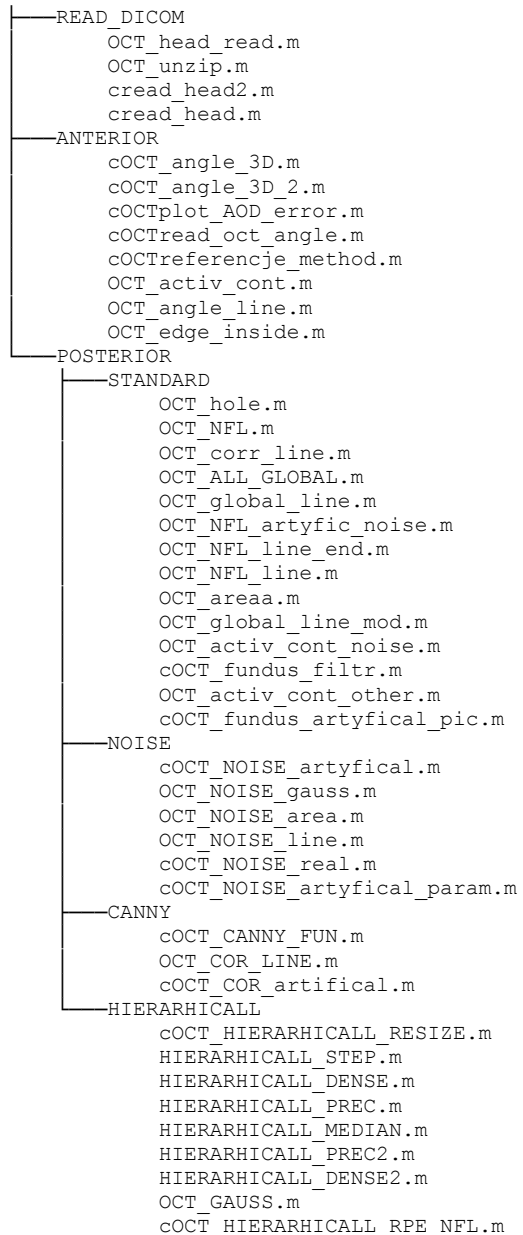
Fig. 6-1 Files tree

BIBLIOGRAPHY

[1]   Adler D. C., Ko T. H., and Fujimoto J. G.: Speckle reduction in optical coherence tomography images by use of a spatially adaptive wavelet filter, Opt. Lett. 29, (2004).

[2]   Akiba, M., Chan, K. P., Tanno, N.: Full-field optical coherence tomography by twodimensional heterodyne detection with a pair of CCD camera, Optics Letters 28, (2003).

[3]   Barry C.: Optical Ccoherence tomography for retinal imaging dissertation, De promotor: Prof. Dr. T.G. van Leeuwen, De copromotor: Prof. Dr. J.F. de Boer (2005).

[4]   Bauma B. E., Tearney G. J.: Handbook of Opticall Coherence Thomography, MarcelDekker (2002).

[5]   Brezinski M.: Optical Coherence Tomography Principles and Applications, Academic Press; 1 edition (2006).

[6]   Canny J.: A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, (1986).

[7]   Choe T. E., Medioni G., Cohen I., Walsh A. C., Sadda S.V. R.: 2-D registration and 3-D shape inference of the retinal fundus from fluorescein images, Medical Image Analysis, Volume 12, Issue (2008).

[8]   Davies E.: Machine Vision: Theory, Algorithms and Practicalities, Academic Press, (1990).

[9]   Farsiu S, Chiu SJ, Izatt JA, Toth CA. Fast detection and segmentation of drusen in retinal optical coherence tomography images. Proceedings of Photonics West, San Jose, CA, February 2008; 68440D1-12 and Proc. SPIE, Vol. 6844, (2008).

[10]   Fercher, A. F., Hitzenberger, C. K., et al.: Measurement of Intraocular Distances by Backscattering Spectral Interferometry. Optics Communications, (1995).

[11]   Fernando J. A.: Retinal Angiography and Optical Coherence Tomography, Springer Science + Business Media, LLC, (2009).

[12]   Gnanadurai D. and Sadasivam V., Undecimated wavelet based speckle reduction for SAR images, Pattern Recognition Letters, 26, (2005).

[13]   Golubovic, B., Bouma, B. E., et al.: Optical frequency-domain reflectometry using rapid wavelength tuning of a Cr4+:forsterite laser. Optics Letters 22, (1997).

[14]   Gonzalez R., Woods R.: Digital Image Processing, Addison-Wesley Publishing Company, (1992).

[15] Gupta S. et al.: A wavelet based statistical approach for speckle reduction in medical ultrasound images, in Proc. IEEE TENCON, 2, (2003).

[16] Hee MR., Puliafito CA., Duker JS, et al.: Topography of diabetic macular edema with optical coherence tomography. Ophthalmology, (1998).

[17] Huang D.: Optical Coherence Thomography Doctor of Philosophy in Medical Engeneering AT the Massachusetts Institute of Technology, Mat, (1993).

[18] Huang, D., Swanson, E. A., et al.: Optical Coherence Tomography. Science, (1991).

[19] Jeoung JW, Park KH, Kim TW, et al.: Diagnostic ability of optical coherence tomography with a normative database to detect localized retinal nerve fiber layer defects. Ophthalmology, (2005).

[20] Kai-shun L. Ch., Li H., Neal W. R., Liu J.,Yim L.C., Yiu K. L. R., Pui P. Ch. Shun Ch. D.: Anterior Chamber Angle Measurement with Anterior Segment Optical Coherence Tomography: A Comparison between Slit Lamp OCT and Visante OCT IOVS, Vol. 49, No. 8, (2008).

[21] Kaluzny, J. J., Wojtkowski, M., Kowalczyk, A.: Imaging of the anterior segment of the eye by spectral optical coherence tomography. Optica Applicata, (2002).

[22] Klinder T., Ostermann J., Ehm M., Franz A., Kneser R., Lorenz C.: Automated model-based vertebra detection, identification, and segmentation, Medical Image Analysis 13 (2009).

[23] Koprowski R., Izdebska-Straszak G., Wróbel Z., Adamek B., Wiczkowski A.: The photometric analysis of selected cell structures. Pattern Recognition and Image Analysis, Vol. 15, No. 4, (2005).

[24] Koprowski R., Wróbel Z. Kucypera K.: 3D Modeling of the growth and division of shoot apex meristem, Machine Graphics and Vision, (2008).

[25] Koprowski R., Wróbel Z.: Analysis of the inclination of elongated biological objects – microtubules, Machine Graphics and Vision, Vol. 14 (2008).

[26] Koprowski R., Wróbel Z.: Automatic segmentation of biological cell structures based on conditional opening and closing, Machine Graphics and Vision, Vol. 14, No. 3, (2005).

[27] Koprowski R., Wróbel Z.: Hierarchic Approach in the Analysis of Tomographic Eye Image Advances in Soft Computing, Springer Berlin / Heidelberg Volume 57, (2009).

[28] Koprowski R., Wróbel Z.: Layers recognition in tomographic eye image based on random contour analysis., Advances in Intelligent and Soft Computing, 57, Computer Recognition Systems, Springer (2009).

[29] Liang J, McInerney T., Terzopoulos D.: United Snakes, Medical Image Analysis, Volume 10, (2006).

[30] Ozcan A., Bilenca A., Desjardins A. E., B. E. Bouma B. E., and Tearney G. J.: Speckle reduction in optical coherence tomography images using digital filtering, J. Opt. Soc. Am. A. 24, (2007).

[31] Radhakrishnan S, ,See J. Smith S.D., Nolan W. P.,Ce Z., Friedman D. S.,Huang D., Li Y., Aung T., Chew P. T. K.: Reproducibility of Anterior Chamber Angle Measurements Obtained with Anterior Segment Optical Coherence Tomography IOVS Vol. 48, No. 8, (2007).

[32] Rogowska J. and Brezinski M. E.: Evaluation of the adaptive speckle suppression filter for coronary optical coherence tomography imaging, IEEE Trans. Med. Imaging, 19, (2000).

[33] Schuman JS., Pedut-Kloizman T., Hertzmark E., et al.: Reproducibility of nerve fiber layer thickness measurements using optical coherence tomography. Ophthalmology (1996).

[34] Scott A. W., Farsiu S., Toth C. A.: Novel techniques to evaluate the infant retina with portable, handheld spectral domain optical coherence tomography. Presented at: the ARVO Annual Meeting, (2008).

[35] Sonka M., Michael Fitzpatrick J.: Handbook of Medical Imaging - Volume 2, Medical Image Processing and Analysis, SPIE, Belligham, (2000).

[36] Thrane L.: Optical Coherence Tomography: Modeling and Applications, Risø National Laboratory, Roskilde, Denmark, May (2001).

[37] User Manual Stratus OCT Model 3000, Carl Zeiss Meditec Inc (2003).

[38] Uset Manual SOCT Copernicus HR, Optopol, (2009)

[39] Witkin A., Terzopoulos D., Int. J.Active Contour Models, M. Kass, Computer, 1(4), pages 321−331, (1987).

[40] Wróbel Z,. Koprowski R.: Automatyczne metody analizy orientacji mikrotubul, Wydawnictwo Uniwersytet Śląski, (2007).

[41] Wróbel Z., Koprowski R.: Przetwarzanie obrazów w programie Matlab, Wyd. Uniwersytet Śląski, (2001).

[42] Yeow, J. T. W., Yang, V. X. D., et al., Micromachined 2-D scanner for 3-D optical coherence tomography. Sensors and Actuators a-Physical, (2005).

[43] Zeimer R. C., Mori M. T., Khoobehi B.: Feasibility test of a new method to measure retinal thickness noninvasively. Invest Ophthalmol Vis Sci, (1989).