
MaZda User's Manual



MaZda User's Manual

1.1. Introduction

MaZda is a computer program for calculation of texture parameters (features) in digitized images. It has been under development since 1998, to satisfy the needs of the participants of COST B11 European project "Quantitative Analysis of Magnetic Resonance Image Texture" and the subsequent COST B21 "Physiological modelling of magnetic resonance image formation". The program code has been written in C++ and compiled for PC computers that use MS Windows® XP operating system.

MaZda was originally developed in 1996 at the Institute of Electronics, Technical University of Lodz (TUL), Poland, by Michał Strzelecki and Piotr Szczypinski, for texture analysis of mammograms. The statistical parameters computed by the early version of the program were derived from the co-occurrence matrix. Consequently, the name of the program is an acronym derived from 'Macierz Zdarzen' that is Polish counterpart of the English term 'co-occurrence matrix'. Thus **MaZda** has no connotation of the famous Japanese car maker.

MaZda User's Manual

1.2. Functionality

A flowchart illustrating typical steps in image texture analysis is shown in Fig. 1.2.1. Image acquisition is the first step. Although in the case of images considered in COST B11 action, this step is performed by means of a magnetic resonance scanner, the following analysis steps can be carried out on images of different origins, e.g. visual ones taken by a digital camera.

The next two steps cover selection of a region of interest (ROI) and computation of texture parameters within the ROI. **MaZda** provides tools for performing these two steps. Lists of parameters computed for each ROI are displayed in a report window ([Section 1.4](#)).

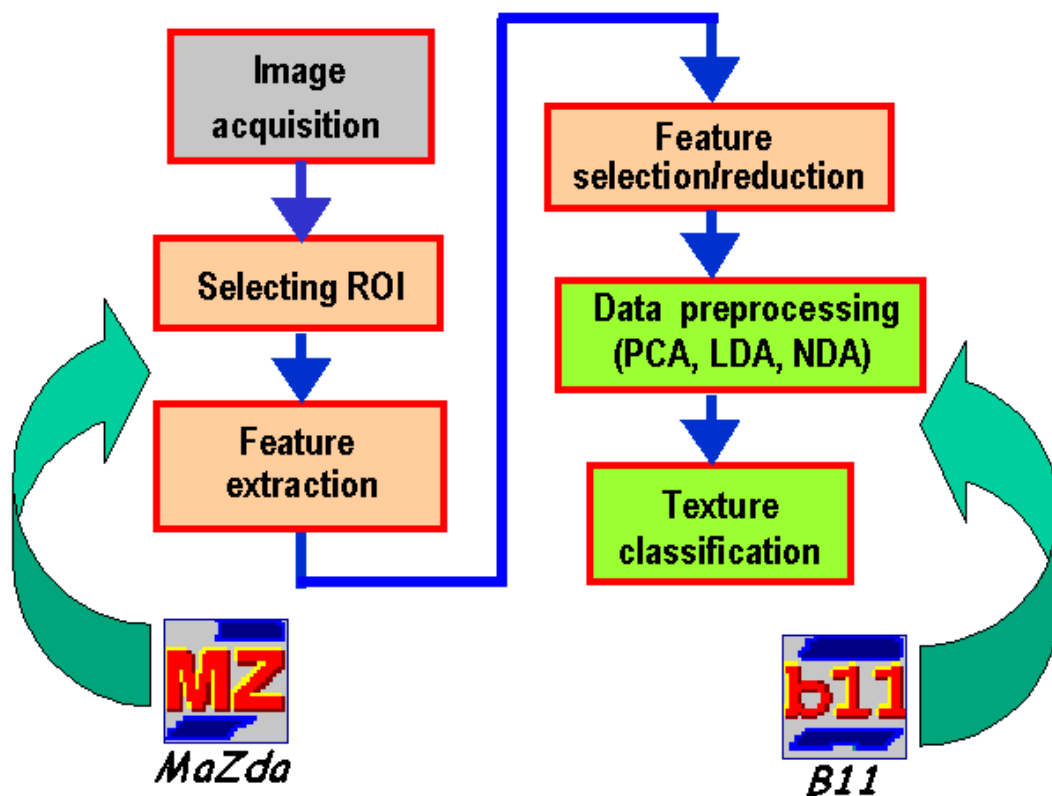


Fig. 1.2.1 Main steps in texture analysis

It can not be said, in advance, which parameters computed for a given texture are best to distinguish one texture from another. Moreover, some of them are highly correlated to each other and thus they do not carry any new information about the texture at hand. Thus there is a need for parameter selection and reduction. Several techniques of feature selection implemented in **MaZda** are described in more detail in [Section 3.2](#).

Yet another program module was developed, for selected texture parameters preprocessing, exploratory analysis and texture data classification. This is called **B11**; it is called from within **MaZda**, or can be run on its own. Using the numerical routines available in this program, one is able to perform principal component analysis (PCA), linear discriminant analysis (LDA), nonlinear discriminant analysis (NDA) and unsupervised data clustering in the feature vector space. These data preprocessing methods are described in [Section 4](#). The raw or preprocessed parameter vectors can be classified with **B11**, by means of a k-NN classifier or an artificial neural network ([Section 4.5](#)).

MaZda User's Manual

1.3. Main window

The main window of **MaZda**, as it looks like after loading an image into its working area, is shown in Fig. 1.3.1. Besides the image panel, this window contains 8 elements - toolbars for image manipulation and/or processing (Fig. 1.3.2a - 1.3.2h).

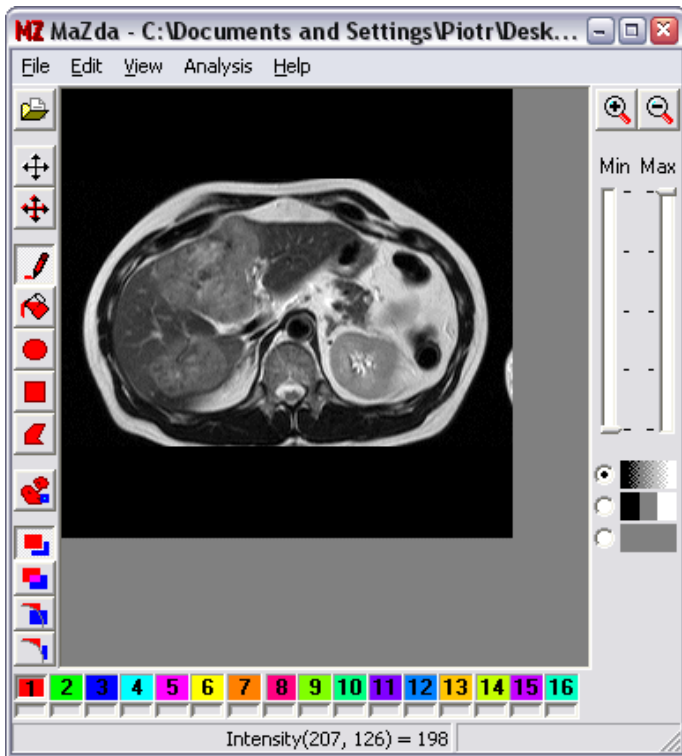


Fig 1.3.1 The main window of MaZda with an example MR image loaded

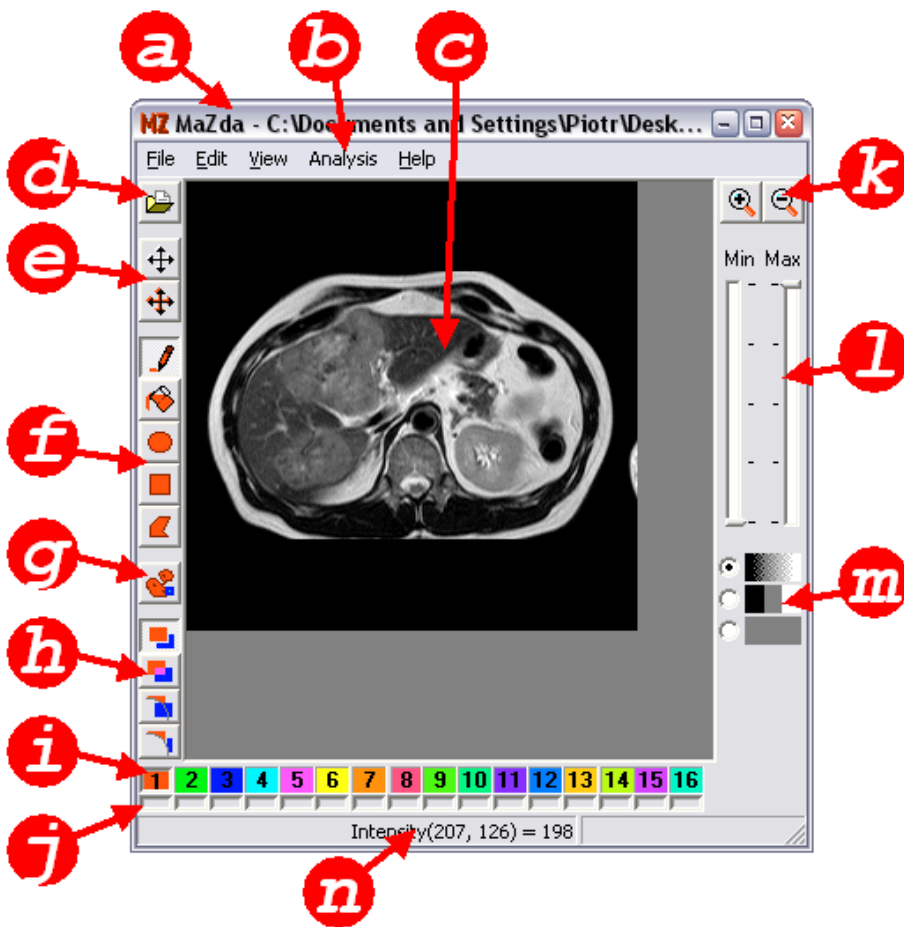


Fig 1.3.2 Elements of MaZda main window: a) window title bar, b) menu bar, c) image panel, d) load file button, e) copy and move buttons, f) graphics toolbar for ROI edition, g) morphological tools for ROI edition, h) drawing mode selection buttons, i) ROI color selector, j) ROI on/off switches, k) zoom in/out buttons, l) sliders for adjustment of grey-scale palette, m) image view mode selector, n) status bar.

An image displayed in the image panel consists of two main layers. One of them is the original image to be analyzed, as e.g. the image presented in Fig. 1.3.1. On top of this layer, the region of interest (ROI) image can be visualized, as e.g. shown in Fig. 1.3.3. To see the ROI image only, one has to click on the lowest radio button in the layer selector (Fig. 1.3.4). More information about the display layers available, and how to switch them on and off, can be found in [Section 2.3](#). To add ROI to an image, one has to either load it from a file by means of **File=>Load ROI** menu options, or paint it using the graphics tool bar and ROI color selector.

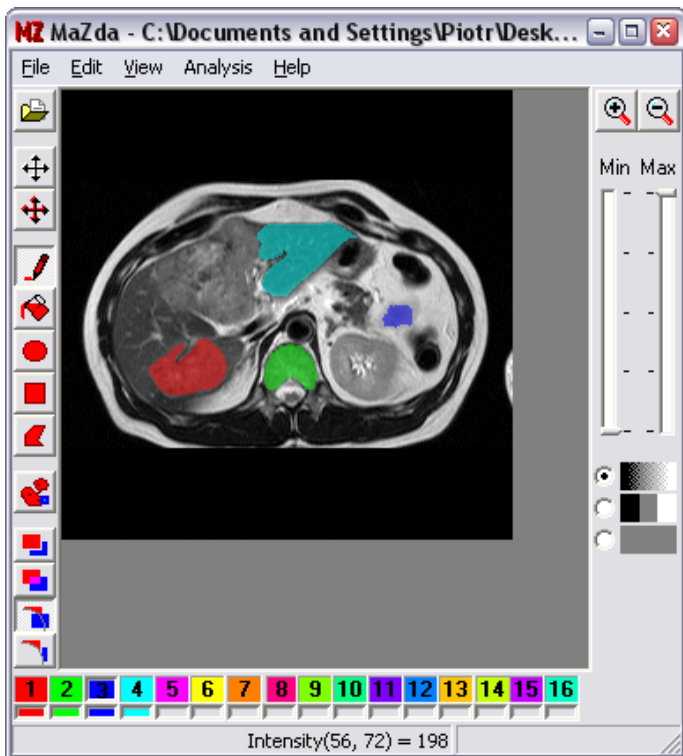


Fig 1.3.3 Region of interest (ROI) image, superimposed on the analysed image

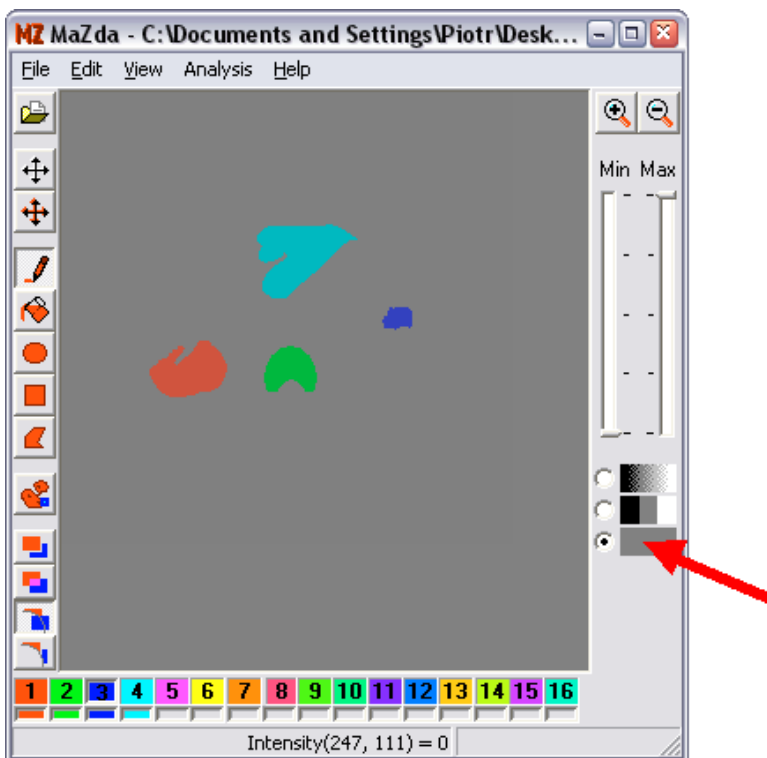


Fig 1.3.4 ROI image layer displayed using layer selector

By setting the position of two sliders on the right-hand side of the window, one can adjust the boundaries of a conceptual window, on the brightness scale, through which the image is observed. Then, what is seen through this window is

stretched out to the full brightness scale, available in the computer system. In this way, the contrast of the image can be enhanced (for visualization only, not for analysis!), as demonstrated in Fig. 1.3.5.

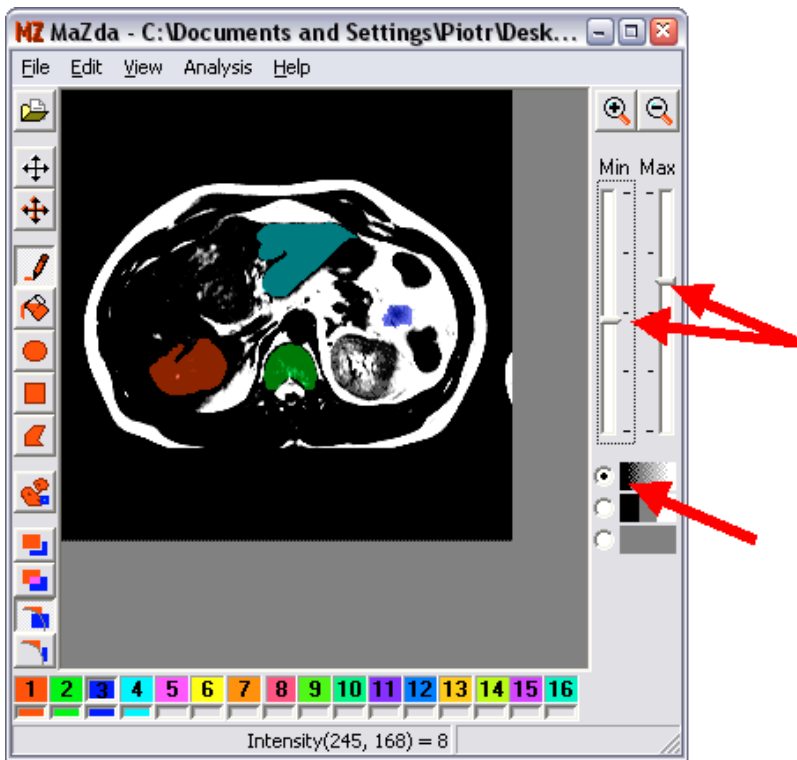


Fig 1.3.5 The effect of sliders' position on image contrast

By choosing the middle button of the image layer selector, one can change the function of the image display in yet another mode. In this mode, the sliders' positions define two thresholds on the brightness scale, respectively of low and high value. The image is converted into its 3-brightness-value representation, using black, grey and white colors. If the actual image brightness is lower than the lower threshold, black color is used. The original pixels whiter than the higher threshold are displayed as white points. Those pixels, whose brightness falls within the range defined by low and high threshold are painted in grey (Fig. 1.3.6). This functionality allows one to quickly segment an image - to identify its dark, medium brightness and white objects.

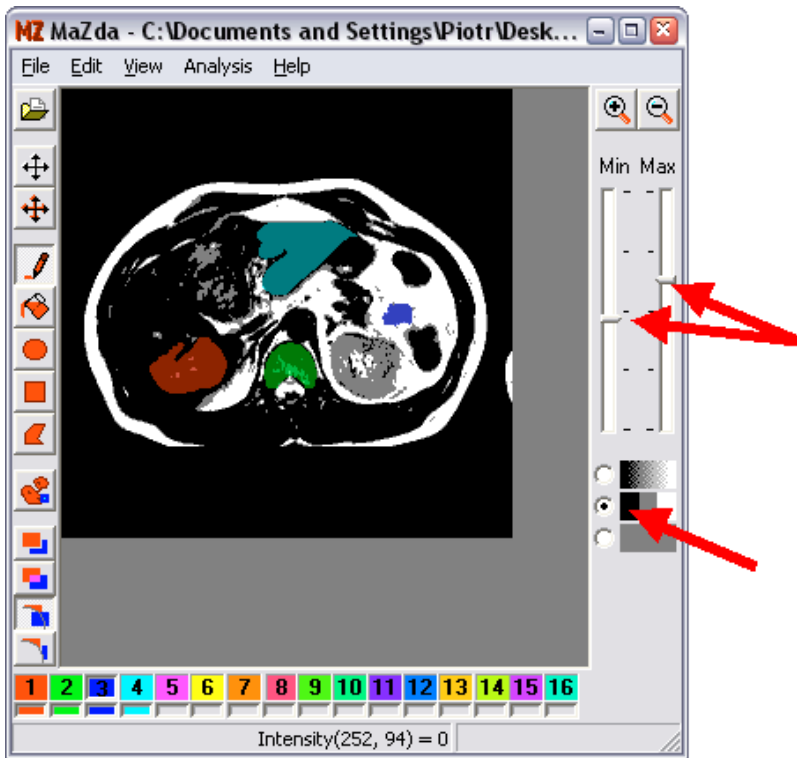


Fig 1.3.6 Image thresholding by means of sliders

MaZda User's Manual

1.4. Histogram window

After loading an image, MaZda computes its histograms, separately for each ROI defined. As an example, consider the image shown in Fig. 1.4.1, that contains 4 ROIs. By choosing **View=>Histogram** menu item, or by pressing **Ctrl+H** keyboard keys, one can display a window that contains these histograms (Fig. 1.4.2a - 1.4.2d).

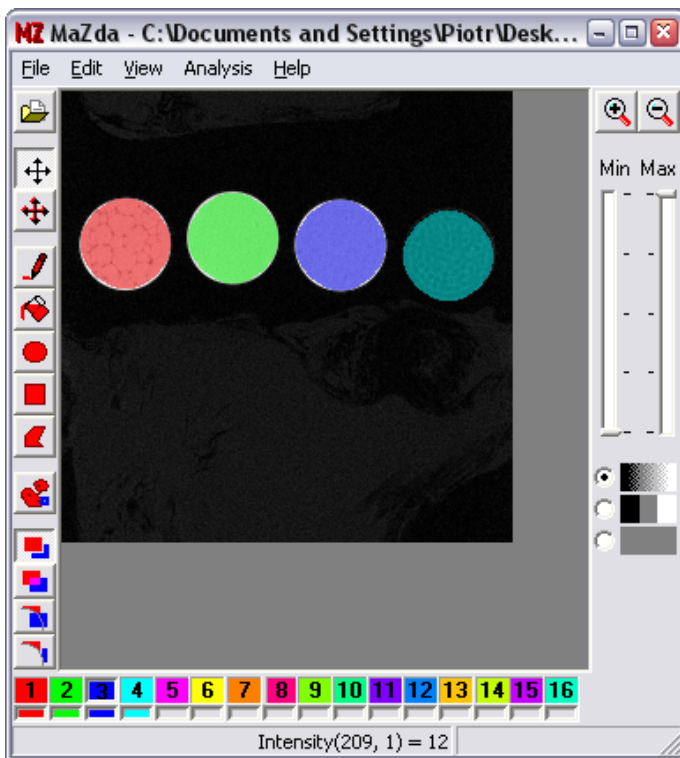
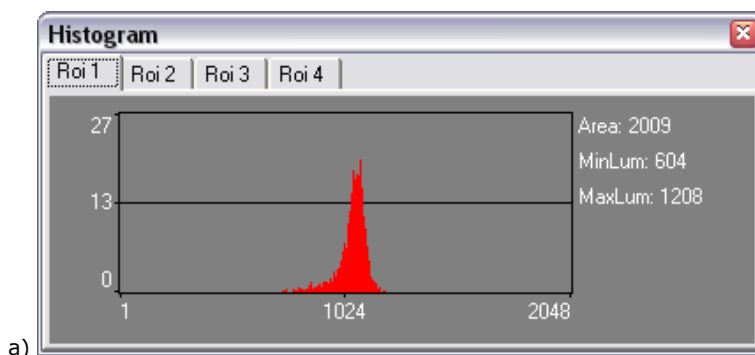


Fig. 1.4.1 An example of MRI image with 4 ROIs superimposed



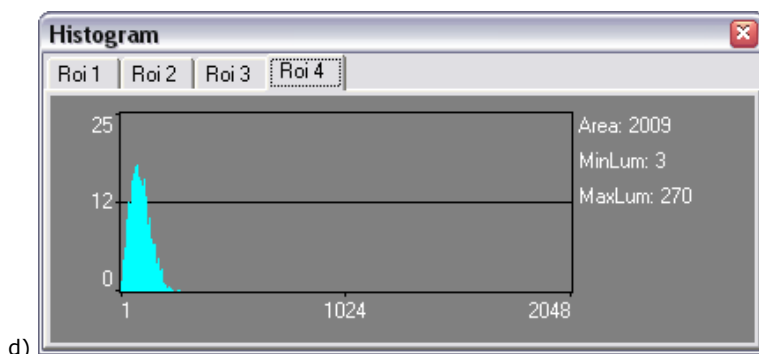
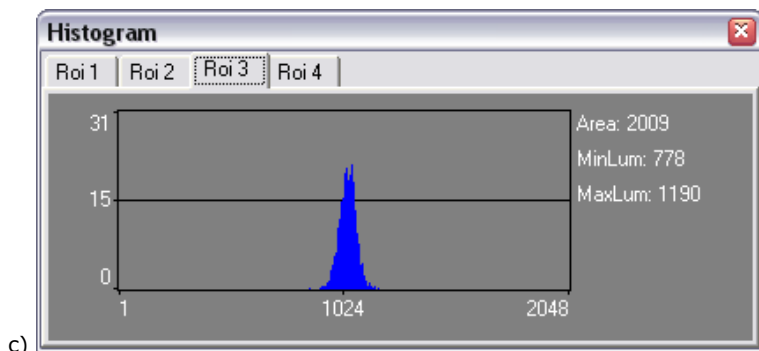
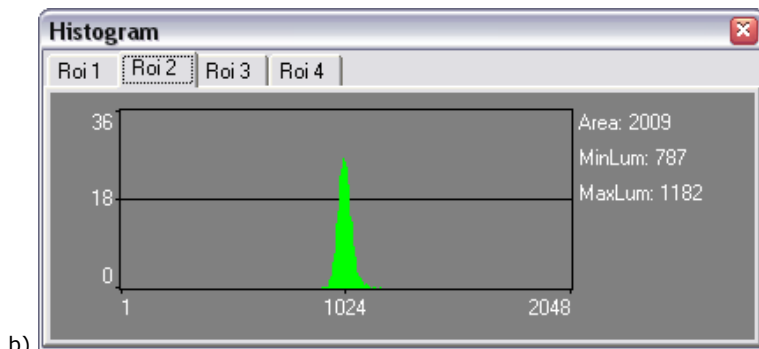


Fig. 1.4.2 Histograms computed for the image and ROI in Fig. 1.4.1

MaZda User's Manual

1.5. Report window

Consider the main window after loading the "text0324.ima" MR image and 4-region ROI image, as shown in Fig. 1.5.1. To compute texture parameters for these ROI, one can invoke **Analysis=>Run** menu item. Then MaZda produces a report tab-page with a 4-column table (one column for each ROI), that can be browsed through in a report window. This window is displayed either by means of **View=>Report** menu item or by pressing **Ctrl+R** keyboard keys. The report window corresponding to the example image from Fig. 1.5.1 is presented in Fig. 1.5.2.

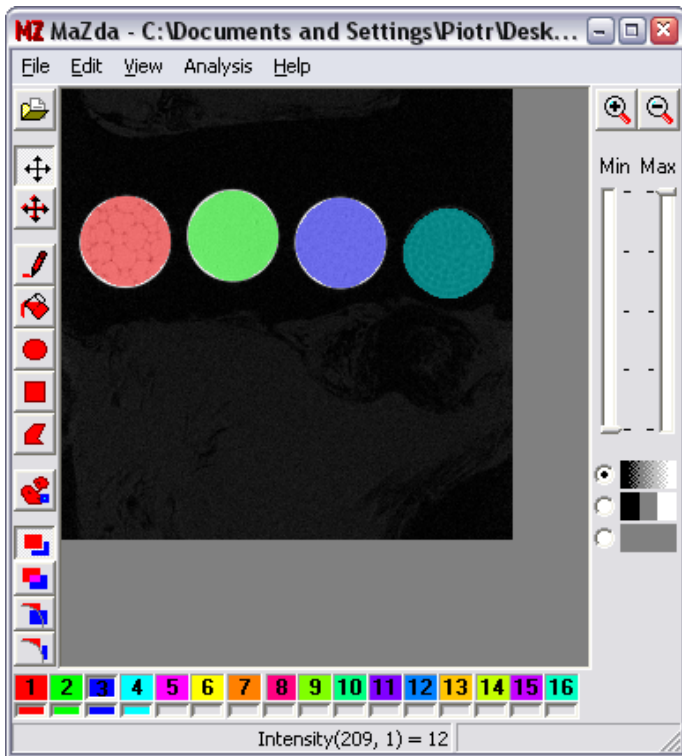


Fig. 1.5.1 An example of MRI image "text0324.ima" with 4 ROIs superimposed, defined in "newdraw.roi" file

Feature name	✓ 1	✓ 2	✓ 3	✓ 4	5
• _Area	2009	2009	2009	2009	0
• _MinNorm	844	917	920	-44	0
• _MaxNorm	1262	1120	1161	224	0
• _Area_S(1,0)	3916	3916	3916	3916	0
✓ S(1,0)AngScMom	0.0023138	0.0012696	0.0011695	0.0010423	0
✓ S(1,0)Contrast	111.26	168.72	183.72	154.75	0
✓ S(1,0)Correlat	0.40151	0.11395	0.074166	0.28962	0
✓ S(1,0)SumOfSqs	92.952	95.206	99.218	108.92	0
✓ S(1,0)InvDfMom	0.151	0.10729	0.089699	0.10168	0
✓ S(1,0)SumAverg	64.985	64.484	64.497	64.555	0
✓ S(1,0)SumVarc	260.55	212.11	213.15	280.93	0
✓ S(1,0)SumEntrp	1.7154	1.7537	1.7586	1.8173	0
✓ S(1,0)Entropy	2.8212	3.0052	3.0375	3.0704	0

Fig. 1.5.2 Report window generated for the image and ROI in Fig. 1.5.1

The report window consists of a menu bar and tab-pages, each containing a list of texture parameters computed for images analysed during a **MaZda** session. If, for example, another image, say "text0321.ima", is loaded (Fig. 1.5.3) and

then analysed, the report window will contain two tab-pages, as shown in Fig. 1.5.4.

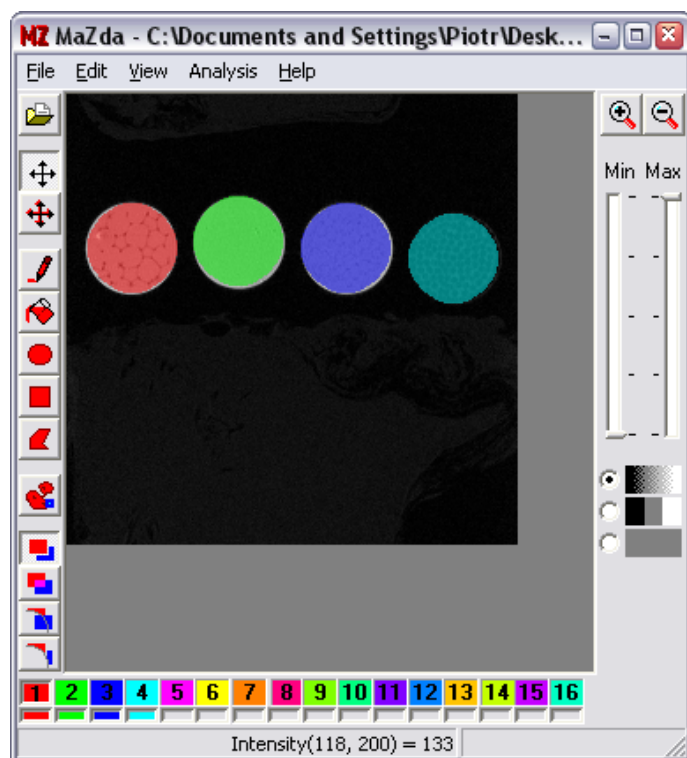


Fig. 1.5.3 Main window for "text0321.ima" image and "newdraw.roi" ROI file

The screenshot shows the MaZda Report window. The title bar reads "MaZda Report". The menu bar includes "File", "Feature selection", and "Tools". The report text includes the following information:

- Image File: TEXT0334.IMA
- ROI File: newdraw.roi
- Image size: 256 x 256
- Min. lum.: 1
- Max. lum.: 1501
- Bits/pixel: 11
- Normalisation = 3 sigma
- Histogram analysis = No
- CD matrix analysis = Yes, Dimensions = 6 x 6, Distances = 1 2
- RL matrix analysis = No
- Gradient analysis = No

Below the text is a table with 5 columns representing different ROIs (1, 2, 3, 4, 5) and a final column for a specific feature. The table contains the following data:

Feature name	✓ 1	✓ 2	✓ 3	✓ 4	5
• _Area	2009	2009	2009	2009	0
• _MinNorm	722	788	821	-60	0
• _MaxNorm	1271	1121	1159	233	0
• _Area_S(1,0)	3916	3916	3916	3916	0
✓ S(1,0)AngScMom	0.0037536	0.0028703	0.0019502	0.0011129	0
✓ S(1,0)Contrast	111.59	83.394	119.22	153.85	0
✓ S(1,0)Correlat	0.34175	0.085746	0.11029	0.30379	0
✓ S(1,0)SumOfsq	84.762	45.608	67	110.49	0
✓ S(1,0)InvDfMom	0.18075	0.15604	0.1198	0.098028	0
✓ S(1,0)SumAverg	65.434	65.956	65.804	64.586	0
✓ S(1,0)SumVarnc	227.46	99.036	148.78	288.12	0
✓ S(1,0)SumEntrp	1.6383	1.5333	1.6549	1.8138	0
✓ S(1,0)Entropy	2.682	2.6712	2.8448	3.0456	0

Fig. 1.5.4 Report window generated for images "text0324.ima" and "text0321.ima", both analyzed using the same ROI, "newdraw.roi"

Each tab-page of the report window displays the information about the texture analysis options, in its upper panel. One can read, for example, from the panel shown in Fig. 1.5.4 that

- the image and ROI file names were respectively "text0321.ima" and "newdraw.roi",
- the image minimum and maximum brightness values were 1 and 1023,
- the image was digitally encoded using 11 bits per pixel,
- the image was normalized before analysis using the "±3 sigma" technique,
- the co-occurrence matrix-derived parameters were computed ...

et ceatera.

The lower panel of the report window contains a list of texture parameters and their values computed for each ROI. Most of them can be used for texture characterisation, except of area parameters, such as \bullet _Area, which merely indicate the number of pixels (or pixel neighbourhoods) used for parameters computation. Columns in the lower panel represent individual regions of interest, whereas rows represent features computed within these regions.

The computed parameter values are listed in columns. The number of columns is equal to the number of different color ROIs, superimposed on the image. In the case of the examples shown here, the number of ROIs is equal to 4. Typically, each ROI corresponds to a separate class of textures; however, it can just be an example of the same class if a number of ROIs is superimposed on a homogeneous texture area in an image. Thus each column in the report file should be attributed a name corresponding to a texture class the user wants it to represent. This can be done by clicking on the column title bar; after that a suitable dialog window is displayed (Fig. 1.5.5).

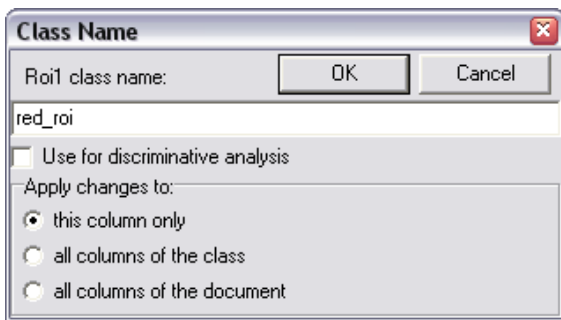


Fig. 1.5.5 Window for editing the name of the class a feature column is meant to represent

The column selected can be excluded from further analysis by unchecking the appropriate checkbox in the window of Fig. 1.5.5. The new class name can be applied to a single column, all columns corresponding to a ROI of a given color, or to all columns in the report.

Any parameter from the list can either be selected (\checkmark S(1,0)Contrast) for further inclusion in the analysis, or deselected from it (\times S(1,0)Contrast). This can respectively be done by clicking on the icon on the left-hand side of the parameter name.

The rows of the parameter table in the lower panel of the report window can be highlighted by clicking on the row content and/or using the **Ctrl** or **Shift** keyboard keys in a standard MS Windows® way (Fig. 1.5.6).

Feature name	\times red_roi	\checkmark 2	\checkmark 3	\checkmark 4	5
\bullet _Area	2009	2009	2009	2009	0
\bullet _MinNorm	722	788	821	-60	0
\bullet _MaxNorm	1271	1121	1159	233	0
\bullet _Area_S(1,0)	3916	3916	3916	3916	0
\checkmark S(1,0)AngScMom	0.0037536	0.0028703	0.0019502	0.0011129	0
\times S(1,0)Contrast	111.59	83.394	119.22	153.85	0
\checkmark S(1,0)Correlat	0.34175	0.085746	0.11029	0.30379	0
\checkmark S(1,0)SumOfSqs	84.762	45.608	67	110.49	0
\checkmark S(1,0)InvDfMom	0.18075	0.15604	0.1198	0.098028	0
\checkmark S(1,0)SumAverg	65.434	65.956	65.804	64.586	0
\checkmark S(1,0)SumVarnc	227.46	99.036	148.78	288.12	0
\checkmark S(1,0)SumEntrp	1.6383	1.5333	1.6549	1.8138	0
\checkmark S(1,0)Entropy	2.682	2.6712	2.8448	3.0456	0

Fig. 1.5.6 A set of highlighted rows of texture parameters

By using the **Feature selection=>Select highlighted** or **Feature selection=>Deselect highlighted** menu items, one can respectively select the highlighted parameters for further analysis, or exclude them from it (Fig. 1.5.7).

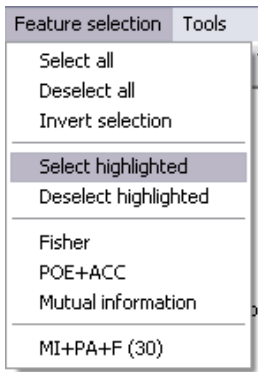


Fig. 1.5.7 The Feature selection menu items of the report window

The selected (checked on the list) texture parameters can be used as an input to feature selection/reduction procedures. There are two such procedures implemented in **MaZda**: one based on Fisher coefficient (a ratio of between-class to within-class variance) and the other that selects parameters that classify the given textures with the smallest error and are least correlated with each other (Fig. 1.5.7). Each of them selects 10 texture parameters from the input list. These procedures were previously the core modules of the **Convert** program.

The features of the tabs of the report windows can be saved to disk, one by one by using the **File=>Save report** menu item. The selected features only can be saved by means of **File=>Save selected** menu item.

The texture parameters selected, either manually or by means of the automated "Fisher"/"POE+ACC" procedures, can be further used for texture analysis by means of the **B11** program, which can be called from within **MaZda** by **Tools=>B11 analysis** menu item of the report window (Fig. 1.5.8). The maximum allowable number of input parameter to **B11** is 30.

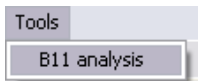


Fig. 1.5.8 The B11 analysis menu item of the report window

MaZda User's Manual

1.6. Image View window

MaZda not only allows one to compute texture parameters within a ROI, as discussed in [Section 1.5](#). Also, which is a very useful functionality, it generates parameter maps which are texture parameters computed in a small window sliding over the image. To obtain such maps one has to select proper analysis options in the [options window](#).

Suppose "271-9-77.bmp" MR image has been loaded to the image panel (Fig. 1.6.1). Assume also GrMean, S(1,0) AngScMom, S(3,0) Contrast, Mean and Variance features have been selected in the options window (Fig. 1.6.2). After running the image analysis using the **Analysis=>Run** menu item of the main window, the Image View window is displayed on the screen. It contains tab pages, each with a feature map, as illustrated in Fig. 1.6.3a-1.6.3e.

If one points at a given pixel in a feature map by means of mouse-driven cursor, the pixel coordinates and feature value are displayed in the status bar of the Viewer window (Fig. 1.6.3).

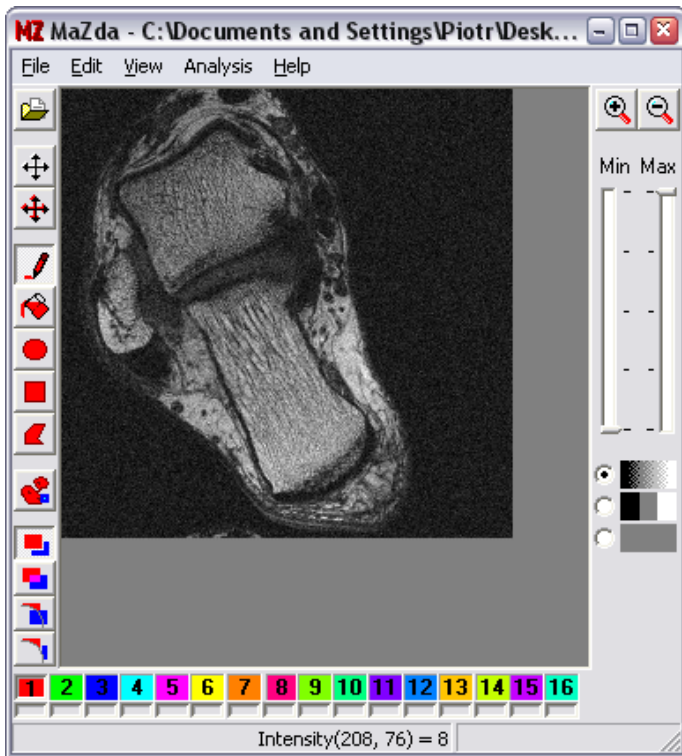


Fig. 1.6.1 An example of MR image "271-9-77.bmp"

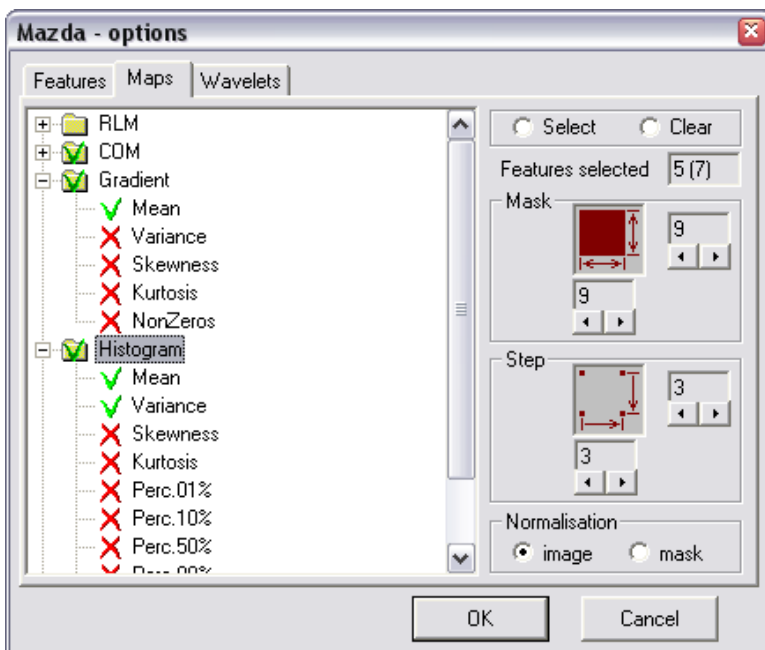
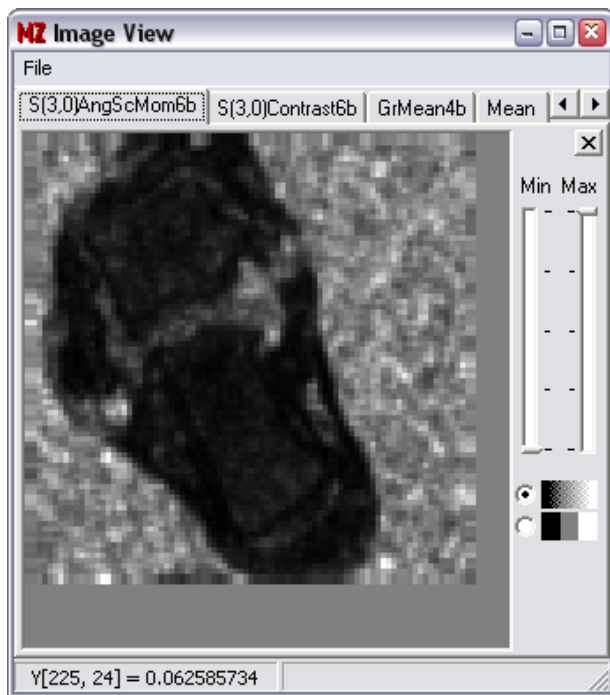


Fig. 1.6.2 An example of the options window



a)



b)



c)



d)



Fig. 1.6.3 Image maps generated for image "271-9-77.bmp"

MaZda User's Manual

1.7. Image information window

If an image is loaded into the **MaZda** working space, one can see the header information extracted from the image file. This can be obtained either by selecting the **View=>Image info** menu item of the main window (Fig. 1.7.1), or by pressing the **Ctrl+F** keyboard keys. The image header information is then displayed in the Image info window, as shown in Fig. 1.7.2.

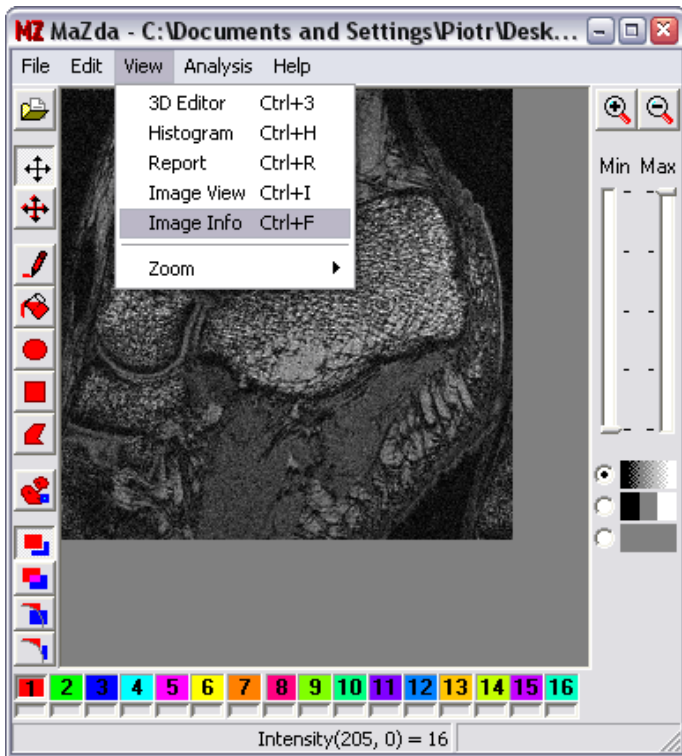


Fig. 1.7.1 An example of MR image "rimml047.ima" and View=>Image info menu item selected

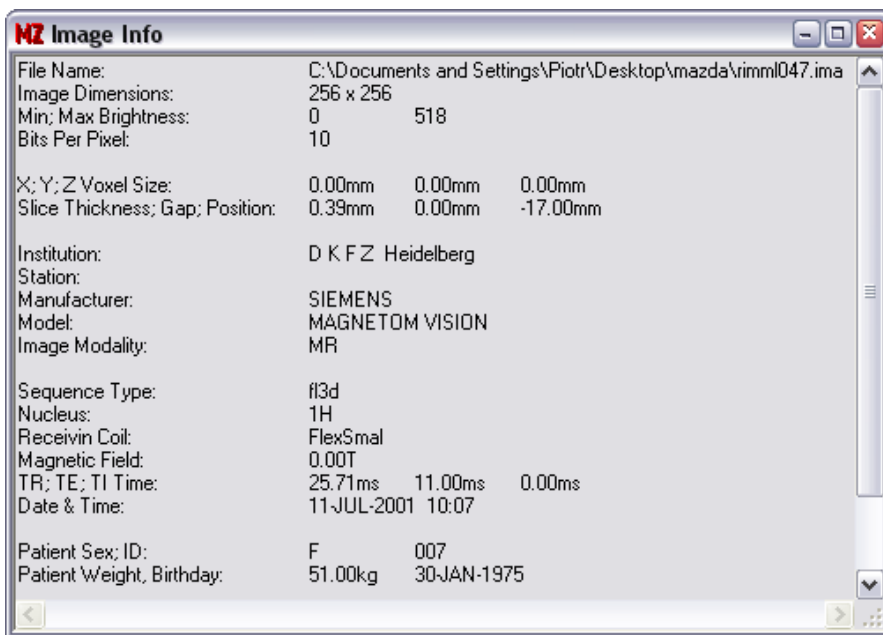


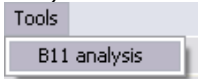
Fig. 1.7.2 The contents of the header of MR image "rimml047.ima" presented in the image information window.

MaZda User's Manual

1.8. B11 window

If a number of texture features has been selected in the [report window](#), either manually or automatically, one can call

the **B11** program for the obtained data exploration, and eventually texture classification. The **B11 analysis** menu item of the report window can be used for the purpose (Fig. 1.8.1). Once it is invoked, the **B11 analysis** program window is displayed on the screen (Fig. 1.8.2). More information about the functionality and operation of **B11** can be found in



[Section 4](#) of this User's Manual.

Fig. 1.8.1 The **B11 analysis** menu item of the report window

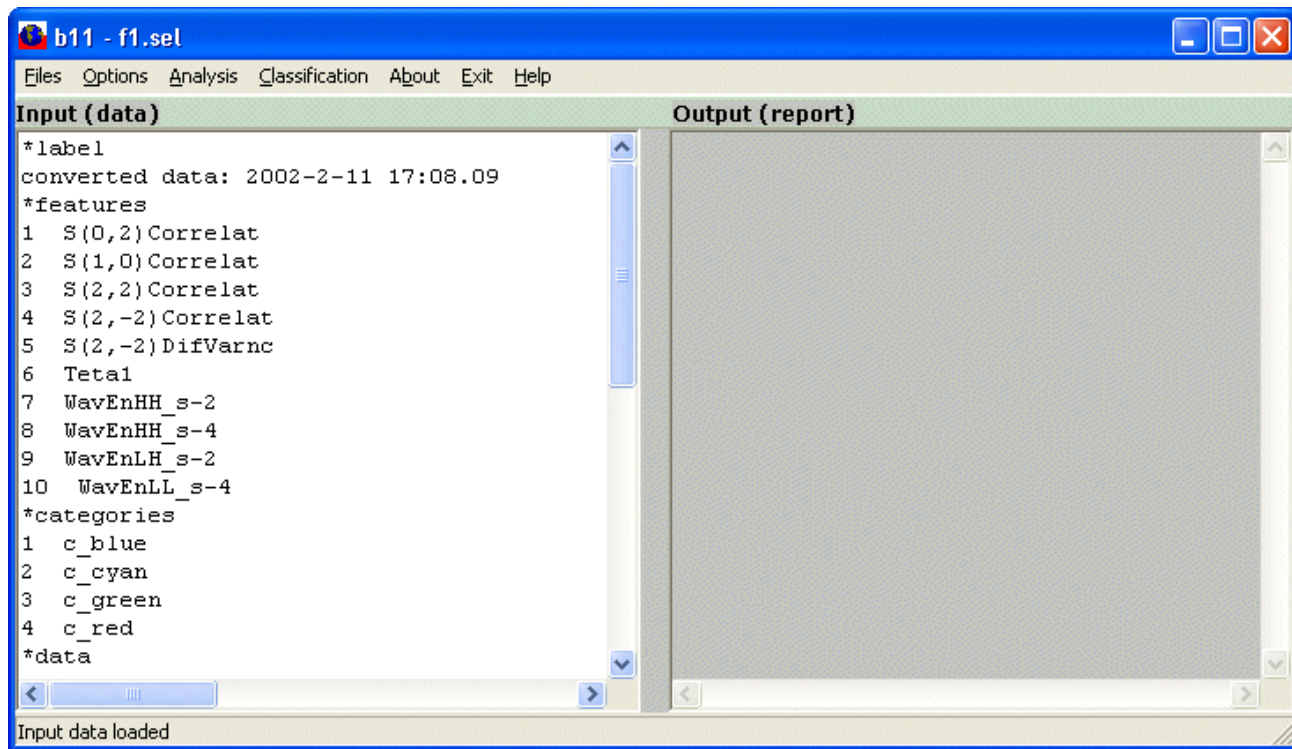
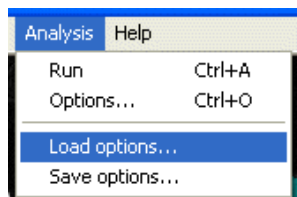


Fig. 1.8.2 the **B11 analysis** program window

MaZda User's Manual

1.9. Options window

The options window appears on the screen (Fig. 1.9.1) if **Analysis=>Options** menu item is selected in the main window. It contains 3 tab-pages, corresponding respectively to setting options to compute texture parameters within ROI ("Features" tab-page), selection of parameters to generate feature maps ("Maps"), and setting options to compute parameters derived from wavelet transform ("Wavelets"). The optional parameter values, that affect operation of **MaZda**, can be saved in a *.ini file for further loading when needed.



1.9.1 ROI feature options

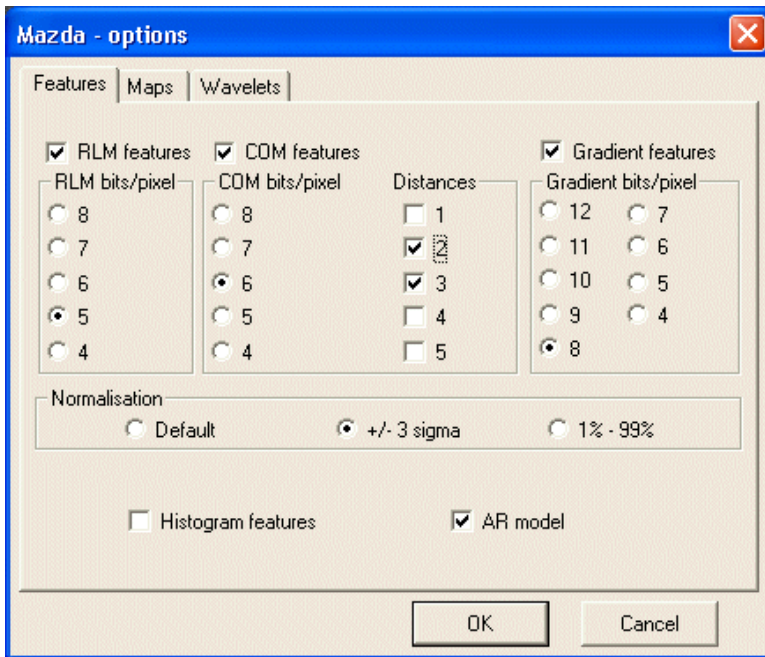


Fig. 1.9.1 The options window (features)

Texture parameters derived from 6 different statistical image descriptors can be computed by **MaZda**. The parameters are derived respectively from

- image histogram,
- image gradient,
- run-length matrix (RLM),
- co-occurrence matrix (COM),
- autoregressive model (AR),
- wavelet transform,

each calculated for up to 16 predefined ROIs.

Calculation of features belonging to a selected group can be enabled or disabled by clicking on a corresponding check box. As an example, for the settings shown in Fig. 1.9.1, the histogram, RLM and COM features are enabled, whereas the gradient-based and autoregressive model features are disabled.

To compute texture parameters derived from the RL and CO matrices, as well as texture parameters derived from the image gradient, the image fragments inside the ROIs are normalized first. The normalization includes

- image intensity range selection,
- image quantization, using selected number of bits per pixel.

To select the intensity range, one has to click on one of the radio buttons inside the **Normalisation** frame in Fig. 1.9.1. There are three normalisation options available in MaZda:

- **default** setting, which is the intensity range of the image under analysis, from 1 to $N_g = 2^k$, where k is the number of bits per pixel used to encode the image under analysis (see [Note 1](#)),
- **± 3 sigma** option, which is equivalent to the range $[\mu - 3\sigma, \mu + 3\sigma]$ where μ is the image mean and σ denotes its standard deviation (both μ and σ are computed separately for every ROI),
- **1% – 99%** option, which is the range between the brightness level at which image accumulated histogram is equal to 1% of its total to the level where the accumulated histogram is equal to 99% of its total (typically, different ROIs give different 1%- and 99%-levels).

The **± 3 sigma** normalization option is selected in the example of Fig. 1.9.1.

In the case of computation of AR model parameters, there is only one normalization method employed for ROI preprocessing. In this case, each ROI is standardized, independent of the setting of the radio buttons inside the **Normalization** frame in Fig. 1.9.1. To do the standardization, mean value and standard deviation are computed, taking into account all the image points inside the ROI considered. Then mean value is subtracted from original image intensity (within the ROI) and the differences are divided by the standard deviation. In this way, all the parameters of the AR model are confined to the range from -1 to 1, independent of the actual image variance.

To select the number of bits used for image quantization before computing the RLM (Haralick, 1979), COM (Haralick, 1973) and gradient features (Lerski, 1993), one has to click on corresponding radio buttons within the respective frame inside the **Options** dialog box. In the example of Fig. 1.9.1, 5 bits per pixel were selected for RLM parameters, 6 bits per pixel were used for COM features, and 8 bits per pixel were chosen for gradient-derived parameters.

Additionally, the COM parameters depend on the distance between pairs of image points that has to be specified to compute the co-occurrence matrix (Haralick, 1973). Up to five different values of the distance are supported. For the example shown in Fig. 1.9.1, two values of distance parameter are enabled: $d = 2$ and $d = 3$. More details about texture parameter definitions can be found in Section 3.3 and in references.

Note 1

To keep consistency with the formulas used in the standard references (Haralick, 1973), (Haralick, 1979) on texture analysis, it is assumed in MaZda that the intensity of image under analysis changes from 1 to N_g , where $N_g = 2^k$, and k is the number of bits per pixel. Thus if originally the image intensity changes from 0 to N_g-1 , MaZda converts this image internally, such that its intensity changes from 1 to N_g . Consequently, the summation indices in the formulas listed in Section 3.3 span the range from 1 to N_g .

1.9.2 Feature map options

The feature maps option window is shown in Fig. 1.9.2. Using mouse, one can indicate an element in the tree diagram (on the left-hand side of the window), and then activate "select" or "clear" button (on the right-hand side of the window) to, respectively, select or unselect group of features for maps generation. Features can be selected in groups (Fig. 1.9.2) or individually (Fig. 1.9.3). Values of selected features are computed for a sliding window that is moved in steps over the image area. The size of the window and values of steps in horizontal and vertical directions can be adjusted by means of arrows of the control elements in the options window (right-hand side of Figs. 1.9.2 and 1.9.3).

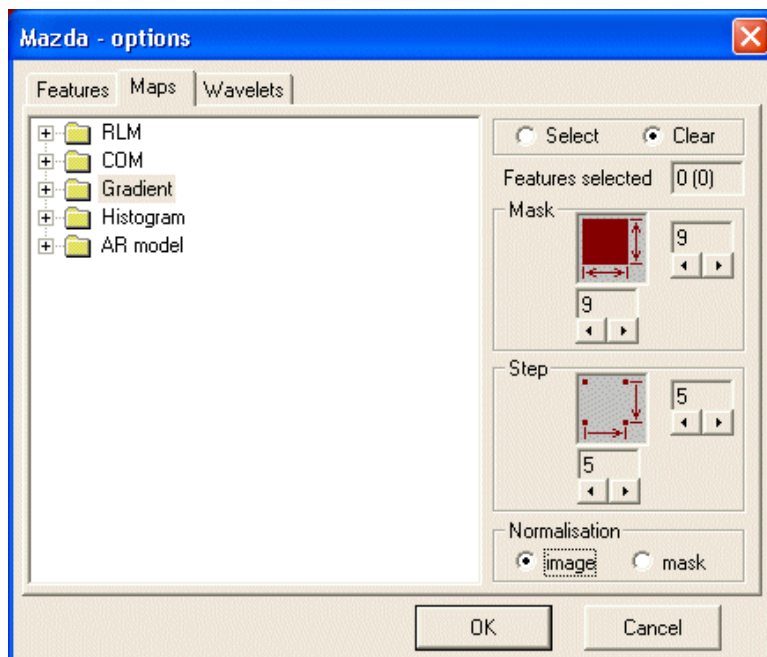


Fig. 1.9.2 The options window (feature maps)

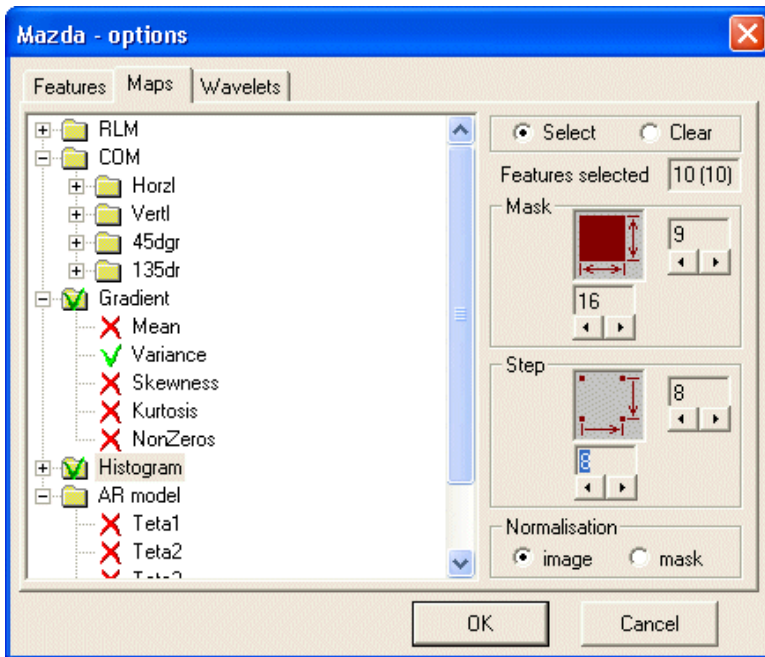


Fig. 1.9.3 Example of individual feature selection for maps calculation

Note 2

Computation of feature maps is, generally, a time consuming process, especially at large size of "Mask" and small value of "Step" (Figs. 1.9.2 and 1.9.3). It is recommended to do some experiments with a small number of feature maps (e.g. one or two), to assess the time needed for map generation on a particular computer, before attempting calculation of a large number of maps.

1.9.3 Wavelet feature options

The wavelet-features tab-page of the option window is shown in Fig. 1.9.4.

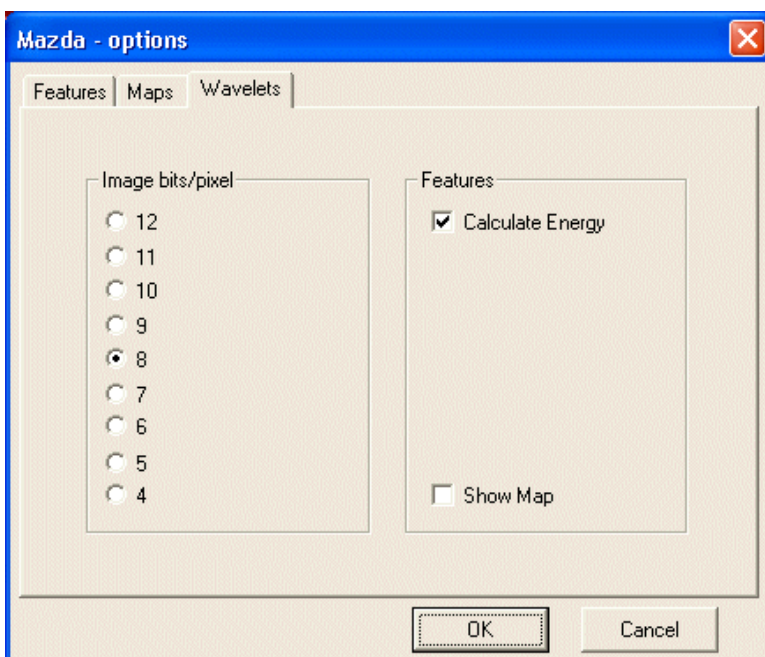


Fig. 1.9.4 The options window (wavelet features)

MaZda User's Manual

2. Getting started with MaZda

MaZda User's Manual

2.1. Input and output data

MaZda allows computation of selected statistical parameters of digital image texture, within a region of interest (ROI) defined by the user. Up to 16 regions of interest can be specified, as shown in Fig. 2.1.1. They are labelled with different colours – to be distinguished from each other.

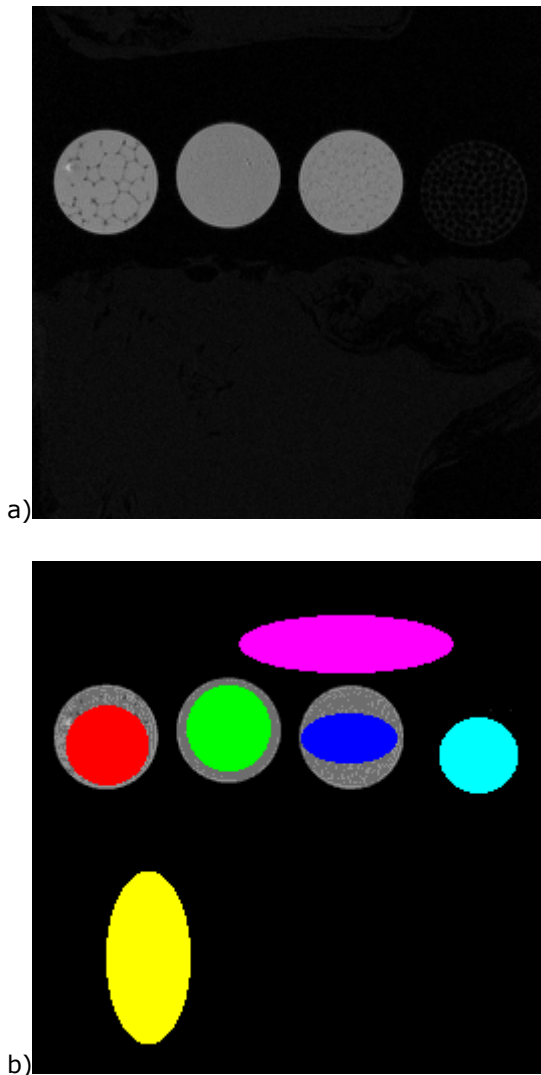


Fig. 2.1.1 An example of a digital image (a) and 6 ROIs within the field of it (b).

The input to the program are then two images. One is the image to be analysed. It can be loaded from disk in one of 12 file formats supported. The other is the ROI image. This image can either be defined by the user or loaded from a disk file. Logically, the ROI image is superimposed by the program on the image under analysis.

The statistical features, calculated for each ROI by MaZda are listed in columns, in the form of a table. Such tables, with added information about the analysis options, are displayed in a [report window](#). They can be saved on disk as a text file (analysis report file). The default name of the report file is ***.par**. This is one of the forms of output information generated by **MaZda**.

Typically, a report file contains a very large amount of data which is almost impossible to study by the

user directly. For example, the number of parameters included in such file is about 300. Then, there is a need for retaining only the most relevant parameters in the context of textures under consideration. To do that, there are two procedures, "Fisher" and "POE+ACC" that can be run from **Feature selection** menu item of the [report window](#). They produce a list of 10 parameters each, placed in a text file with extension ***.sel**. These files form the second form of output information from **MaZda**.

The ***.par** and ***.sel** file can be used as an input to any statistical analysis program to do further image analysis, e.g. aimed at texture classification. The simplest, yet fully useful solution is to apply MS Excel® for the processing of the analysis report files. For the user convenience, the **B11** program has been designed and linked to **MaZda**. This program takes ***.sel** files as input and allows one data preprocessing (with PCA, LDA and NDA techniques) and texture classification (k-NN and artificial neural network classifiers). It can be called from **Tools=>B11 analysis** menu item of the report window, and is described in [Section 4](#) of the Users' Guide.

Yet another information that **MaZda** extracts from the input images are their [histograms](#) computed for each ROI.

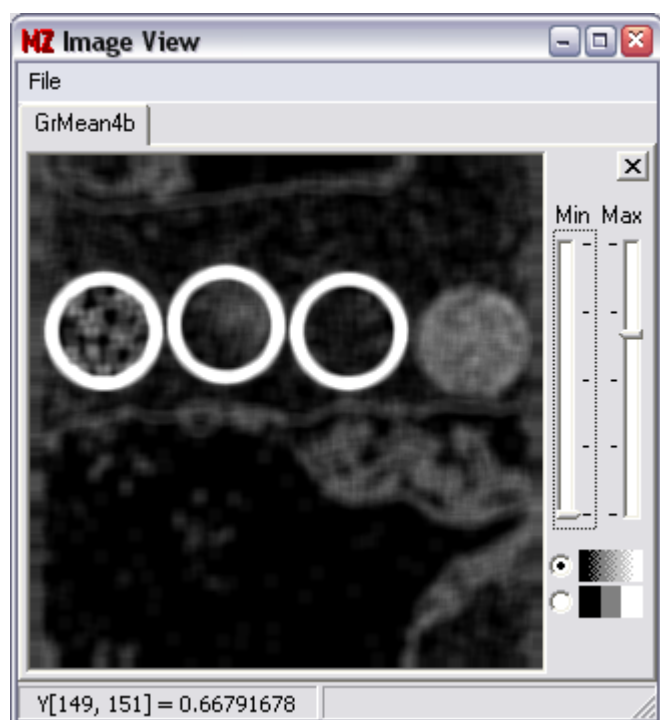



Fig. 2.1.2 A mean-gradient feature map computed for Fig. 2.1.1a (the input image was quantized to 4 bits per pixel, prior to feature computation)

MaZda allows also computation of [feature maps](#), obtained from the input image. A sliding window of adjustable size is moved (with adjustable steps) along the x and y coordinates. For each window position, selected image features are computed and presented in a form of another image - the feature map. An example of feature image (map), which is in this case a gradient magnitude computed for the image of Fig. 2.1.1(a), is demonstrated in Fig. 2.1.2. The feature maps are displayed as tab-pages of the [Image View window](#).

MaZda User's Manual

2.2. Loading the image

There are 4 ways of loading an image to be analyzed into the **MaZda** working space:

- selecting **File=>Load image...** menu item and going through a standard MS Windows[®] procedure,
- clicking on  button and going through a standard MS Windows[®] procedure,
- double clicking on a image file name registered as **MaZda** file name (this option will not work on the first run of the software on a given computer),
- indicating the image file name on the list in the MS Windows Explorer[®] program window, pressing the left-hand mouse button, moving the cursor to inside the **MaZda** main window, and releasing the left mouse button ("drag'n'drop" option).

2.2.1. Raw image options

The assumed structure of a raw-image file is illustrated in Fig. 2.2.1. It has a form of a vector of consecutive binary words used to encode the image data, with a header placed before the images.



Fig. 2.2.1 Assumed structure of a raw-data image file

Each word represents the brightness of corresponding image pixels. The words can be either 8-bit or 16-bit long. The choice of the word length ("Bits per pixel") can be made by means of two radio-buttons located in the upper right corner of the dialog window (Fig. 2.2.2). In the case of 16-bit (2-byte) words, the order of the bytes can be selected by means of another pair of radio buttons, indicated as "Bytes order" in Fig. 2.2.2. The "Intel" option denotes the arrangement in which the most significant byte (MSB) precedes the least significant byte (LSB) in every word stored in a file. The "Motorola" option is the opposite: the LSB comes first, then MSB follows.

The header information is not read by MaZda; however, the length of the header should be provided by the user to give the program proper address reference for reading the images ("Header length"). In the case of no header, the zero digit should be entered in the edit box. The size of the images (in pixels) can be entered using the upper pair of edit boxes in the dialog window ("Image dimensions" in Fig. 2.2.2) - horizontal dimension on the left, and vertical dimension on the right. Finally, the lower edit box ("Image number") allows users to indicate the number of the image that is to be read by MaZda from the input file, first image being numbered by 0 (Fig. 2.2.1).

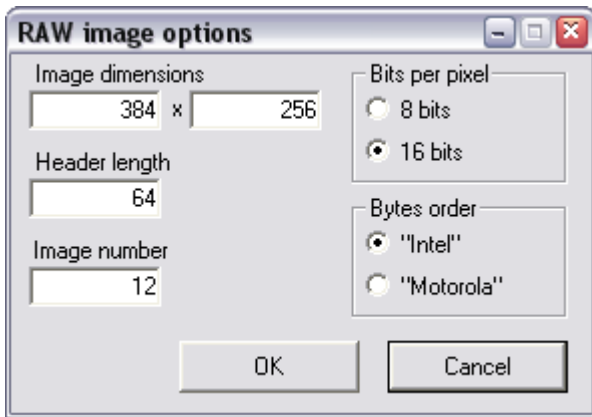


Fig. 2.2.2 Dialog window of *File=>Raw options...* menu item

2.2.2. Loading color image

MaZda is capable of analysis of gray-scale image only. However, when loading the color image bitmap the user may select what kind of information or color channel to load into MaZda. While loading the color image MaZda converts it into a gray-scale with one of eight algorithms. Thus, it may convert color image according to CCIR Recommendation 601-1 (Y channel), it may extract red (R), green (G), blue (B), U or V color components. Moreover, it may compute Saturation (S) or Hue (H) of color image. The Fig. 2.2.3 shows an example color image and its color channels computed by the MaZda image loader.

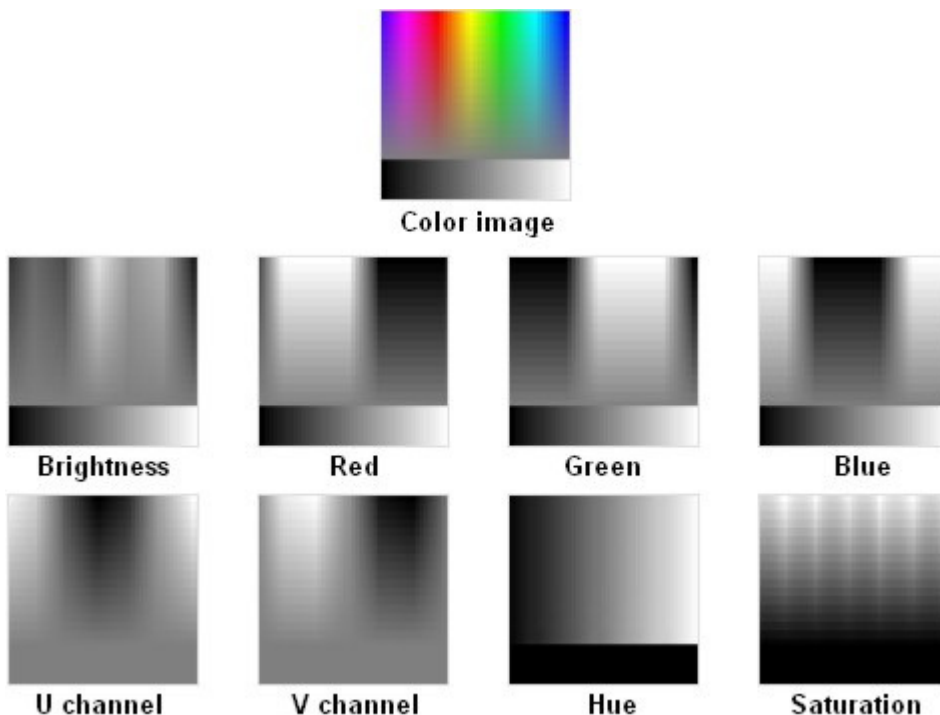


Fig. 2.2.3 Example of color image and its color channels

To switch a color conversion mode one has to select **File->Color conversion...** option from the MaZda menu, and then, within a **Color conversion** dialog window, to select appropriate color conversion algorithm (Fig. 2.2.4). MaZda will load a selected color channel of the opened image after pressing the **Reload** button. Pressing **OK** button will not automatically reload image.

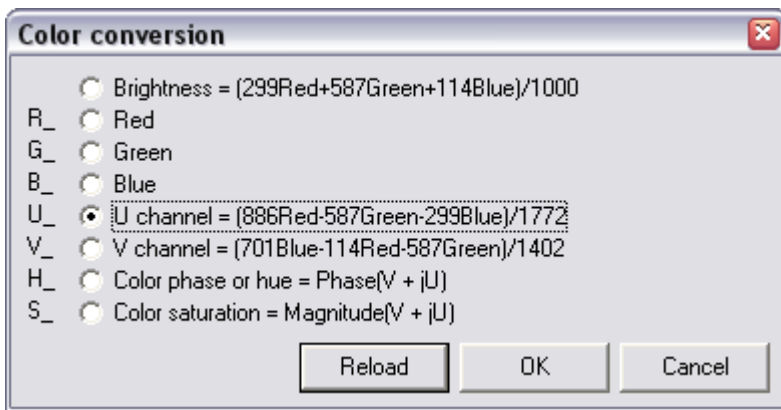


Fig. 2.2.4 Color conversion dialog window

MaZda User's Manual

2.3. Editing ROI

It is possible to define up to 16 regions in the image (image fragments) for which texture parameters will be computed. These regions of interest (ROIs) are marked with different colours. (Immediately after any image is loaded, only one region - marked in red - covering the whole image is defined. It can be cleared or reset by means of appropriate **Edit** menu items.)


To define or redefine the ROI one has to

- choose the number and colour of the region to define (or redefine) from the region panel (at the bottom of the window) by clicking ROI switch (a row of coloured boxes with numbers on it),
- use drawing tools (left panel) to draw the ROI shape on the surface of the image.

As an example, consider the image in Fig. 2.3.1. To better see the image structure, switch the image



palette to three-level mode, adjust "Min" / "Max" track-bars (Fig. 2.3.2), choose "Draw line" tool, adjust the line thickness and paint the ROI over the image (Fig. 2.3.3).

Basic mathematical-morphology operators (erosion, dilation, opening and closing, e.g. [Giardinia 1988](#)) can be applied to the manually-painted ROI - to fill the holes and smooth boundaries - by pressing the  button (available from version 3.08 onwards, as can be seen in Fig. 2.3.1).

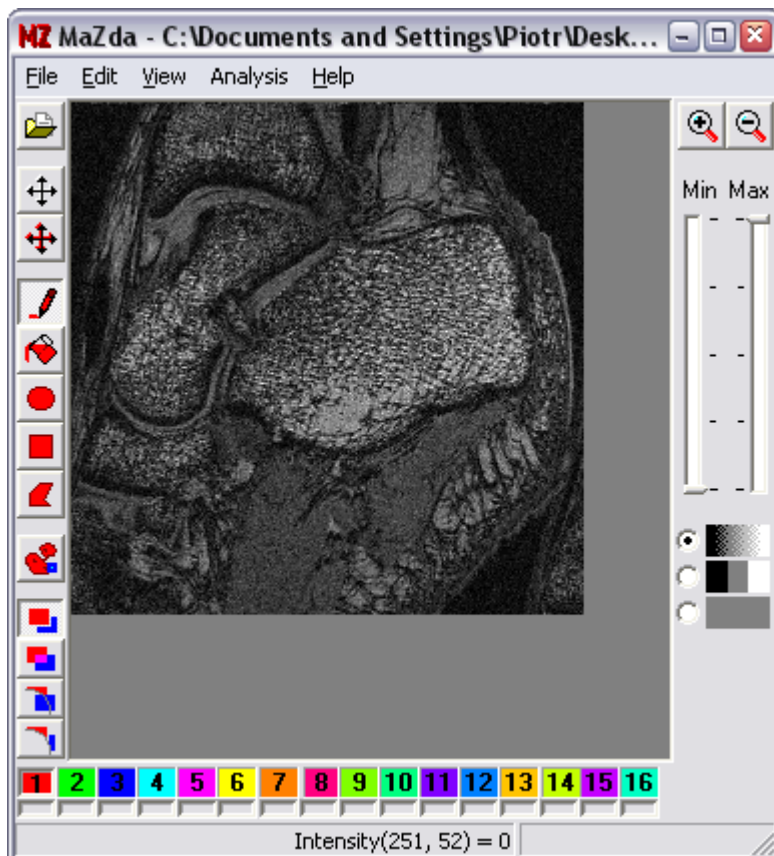


Fig. 2.3.1 An MR image of a trabecular bone

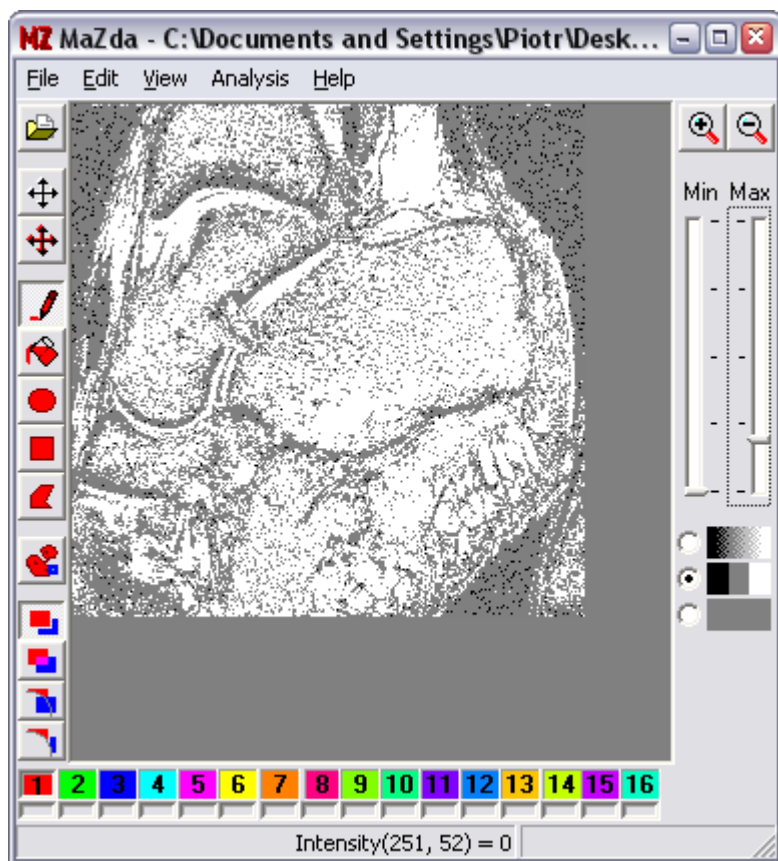


Fig. 2.3.2 The MR image of Fig. 2.3.1 adjusted for editing a ROI

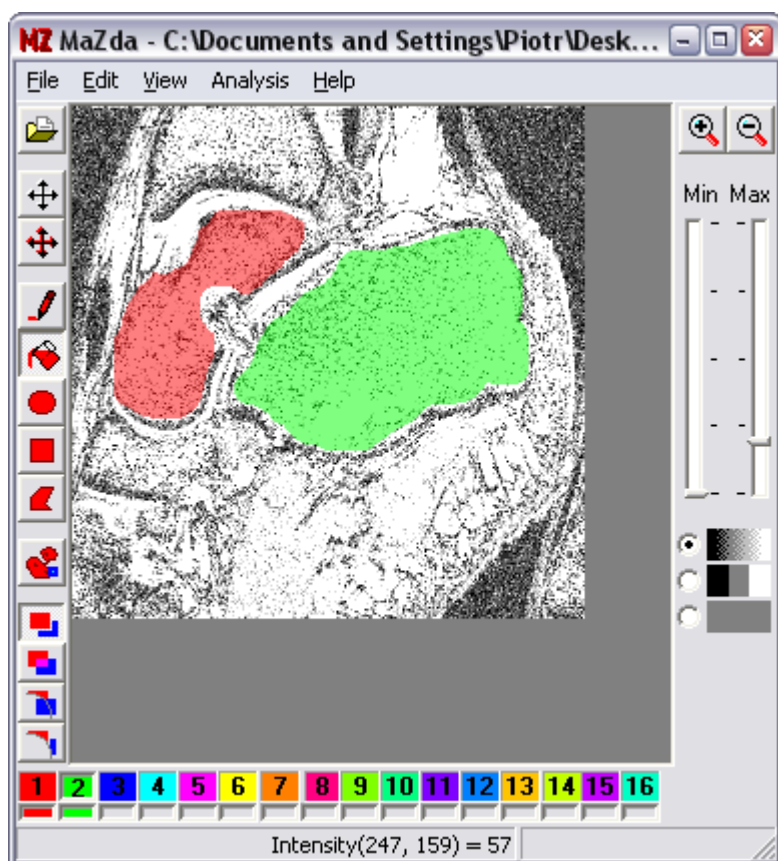


Fig. 2.3.3 Two ROIs edited by means of a "Draw line" tool for MR image of Fig. 2.3.1

Any region can be moved ("Move" icon), copied ("Copy" icon) or deleted (click on "Move" icon and move chosen region outside of image border). Also, a fragment of chosen region can be deleted (choose "Clear" and use any drawing tool for clearing out a fragment of region).

The regions can be defined as non-overlapping or overlapping ("Non overlapping ROI" and "Overlapping ROI" icons). If non-overlapping option is chosen, when ROI is being drawn, it destroys other regions drawn at its location. If overlapping option is chosen, drawing will produce regions that share a common image area.

Any region may be switched on and off by clicking ROI on/off box (coloured, smaller box without number on the panel at the bottom of the window). **Please note that switching the region off doesn't affect the analysis process, the region is still taken into consideration during computation of texture parameters!**

The image [histogram](#) plot within regions may be viewed **View=>Histogram**. The histogram is updated every time the region changes.

Every region can be assigned a class name. To define a class name of a region, double-click or right-click on the ROI switch box and fill up the form displayed with a name (Fig. 2.3.4). The region name appears when the mouse cursor stays over ROI switch box for a while. Regions defined together with their names can be saved to a file or loaded from a file (**File=>Save ROI...** and **File=>Load ROI...** menu items).

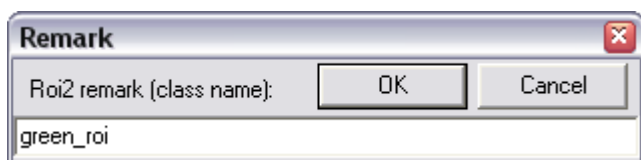


Fig. 2.3.4 Dialog box for giving a name to ROI

MaZda User's Manual

2.4. Image analysis

Image analysis covers two tasks: texture parameter computation within ROIs and feature map generation. Before running the analysis, by selecting **Analysis=>Run** menu item, one has to select the required analysis [options](#) using the **Analysis=>Options** menu item.

After analysis is done, its results are displayed in [Report](#) and/or [Image View](#) windows.

MaZda User's Manual

2.5. Parameter selection and reduction

Selection of parameters can be done manually, as described in [Chapter 1.5](#), or automatically, using four selection methods. First one is based on selection of feature subsets (they can contain one, two or three features) which provide the lowest classification error with 1-nearest neighbor classifier. Second method is based on Fisher coefficient, third one uses both minimisation of classification error probability and average correlation coefficients (POE+ACC), and the fourth one implements the Mutual Information coefficient (MI).

In the case of automatic selection, calculations are performed for all features that have been selected in parameter files opened in the **Report** window. All those files must have the same structure, i.e. the same definition and number of features.

Texture classes are recognised based on ROI names. By default, ROI names are numbered from 1 to 16. They correspond to different ROI colours (red ROI corresponds to 1, green ROI corresponds to 2 and so on). ROI name can be manually changed to provide more meaningful names to texture classes. Particular ROI can be disabled for discriminant analysis. The procedure for ROI name changing and disabling is described in [Chapter 1.5](#) (Fig. 1.5.5). The number of texture classes (different ROIs) is limited to 16.

To perform feature selection based on automatic techniques, one should use **Feature selection=>Optimal subsets** (next, number of features in the subset should be selected), **Fisher**, **Feature selection=>Fisher**, **Feature selection=>POE+ACC** or **Feature selection=>Mutual Information** menu items in the **Report** window. After that, one of six windows shown in Fig. 2.5.1, will appear.



a)

Feature pairs

Feature1 name	Feature2 name	k-NN
S(1,0)DiVarnc	Perc.99%	1
S(2,0)DiVarnc	GrSkewness	2
Perc.99%	Perc.01%	2
S(1,-1)DiVarnc	Perc.99%	2
S(2,0)DiVarnc	Perc.99%	2
S(1,0)DiVarnc	S(1,0)DiEntrp	2
S(1,0)InvDfMom	S(1,0)DiVarnc	2
S(2,0)InvDfMom	S(1,0)DiVarnc	2
S(2,2)DiEntrp	S(1,0)DiVarnc	2
S(5,-5)Correlat	S(1,0)DiVarnc	2

Accept Discard

b)

Feature triples

Feature1 name	Feature2 name	Feature3 name	k-NN
S(0,1)DiVarnc	GrMean	GrKurtosis	0
S(0,1)DiVarnc	GrNonZeros	GrKurtosis	0
S(0,1)DiVarnc	GrSkewness	GrKurtosis	0
S(0,1)DiVarnc	GrVariance	GrKurtosis	0
S(0,1)Contrast	Mean	GrKurtosis	0
S(0,2)Contrast	Mean	GrKurtosis	0
S(0,2)DiVarnc	Mean	GrKurtosis	0
S(1,1)DiVarnc	Mean	GrKurtosis	0
S(0,5)SumAverg	Perc.50%	GrKurtosis	0
S(5,5)SumAverg	Perc.50%	GrKurtosis	0

Accept Discard

c)

Fisher coefficient

Feature name	F
S(4,0)AngScMom	312.3768
S(2,0)AngScMom	307.7508
S(3,0)AngScMom	305.4635
S(5,-5)AngScMom	294.7775
S(4,-4)AngScMom	282.1180
S(1,-1)AngScMom	273.8941
S(5,0)AngScMom	271.6925
S(3,-3)AngScMom	268.4229
S(0,2)AngScMom	266.8662
S(4,4)AngScMom	255.2278

Accept Discard

d)

e)

POE+ACC	
Feature name	P
Teta3	0.3612
Horz_RLNonUni	0.3647
S(1,0)Correlat	0.3770
45dgr_GLevNonU	0.3944
Teta1	0.4249
Teta2	0.4389
S(0,3)Correlat	0.4400
Vertl_RLNonUni	0.4408
S(1,0)DiVarnc	0.4541
S(0,3)DiEntrp	0.5000

f)

Mutual Information	
Feature name	MI
S(0,3)Correlat	1.8299
S(0,2)Correlat	1.7290
S(0,3)Contrast	1.7039
S(2,-2)DiVarnc	1.6949
GrMean	1.6941
S(0,3)DiVarnc	1.6877
S(1,0)Correlat	1.6533
WavEnLL_s-5	1.6484
S(1,0)DiVarnc	1.6376
WavEnLL_s-4	1.5810

Fig. 2.5.1 Window of features (a), feature pairs (b), feature triples (c) with lowest 1-NN classification error; features with: Fisher coefficients (d), POE+ACC coefficients (e), Mutual Information coefficients (f)

These windows contain respectively a list of up to 10 features with largest F coefficients or a list of up to 10 features with lowest POE+ACC coefficients, depending which selection method has been applied. After pressing **Accept** button, only the features displayed in this window will be selected. Selected features can be saved as a *.sel file using **File=>Save selected** menu item.

Manual and automatic feature selection techniques can be mixed, i.e. some features can be manually disabled (i.e. unchecked in the **Report** window) before automatic feature selection analysis. It is also possible to manually disable (i.e. uncheck on the list in the **Report** window) features chosen by the Optimal subsets/Fisher/POE+ACC/Mutual Information methods.

During automatic analysis, the error message shown in Fig. 2.5.2 can appear. It means, that

- there is only one ROI defined (one texture class identified),
- there is only one sample of some texture class

To continue automatic selection in such situation, one should define or enable more ROIs (texture classes) or load more samples of a given texture.

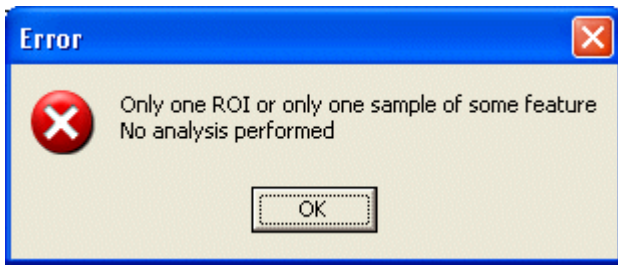


Fig. 2.5.2 Error message window

MaZda User's Manual

2.6. Analysis automation

The process of texture analysis can be automated by running a so-called macro. This can be done by choosing **File=>Run macro...** menu option from the main menu. Macro is a text file, usually with **.txt** extension, which contains consecutive **MaZda** commands. It governs the required operations of loading and saving files, setting analysis options, running calculation of texture parameters, and so on.

The macro commands that can be recognised by **MaZda** are as follows:

1. **LoadImage *file_path_and_name*** – loads an image for analysis from a given file,
2. **LoadROI *file_path_and_name*** – loads regions of interest from a given file,
3. **LoadOptions *file_path_and_name*** – loads options from a given file,
4. **LoadReport *file_path_and_name*** – loads a report into the report window,
5. **ColorChannel [*arg = Y, R, G, B, U, V, H or S*]** – selection of color conversion mode,
6. **ReloadImage** – reloading an opened image (e.g. after changing a color conversion mode),
7. **RunAnalysis** – starts the analysis process,
8. **Execute *file_path_and_name argument_list*** – executes program from a given file,
9. **AggregateReports** – joins data from two reports,
10. **SaveImage** – saves an opened image,
11. **SaveSelected *file_path_and_name*** – saves selected features to a given file,
12. **SaveReport *file_path_and_name*** – saves report from opened tab-page of report window to a given file,
13. **SaveMap *file_path_and_name*** – saves feature map from opened tab-page to a given file,
14. **CloseReport** – closes opened tab-page of report window,
15. **CloseMap** – closes opened tab-page of feature map,
16. **Exit** – unconditional MaZda termination.

Any line starting with `'/'` or `;'` is a comment which is ignored by the interpreter.

Plugins are implemented in MaZda through its macro technology. Macro files given a `“.plugin”` extension, located in MaZda’s working directory are recognized as plugins, and are accessible from the MaZda **Tools** menu (Fig. 2.6.1).

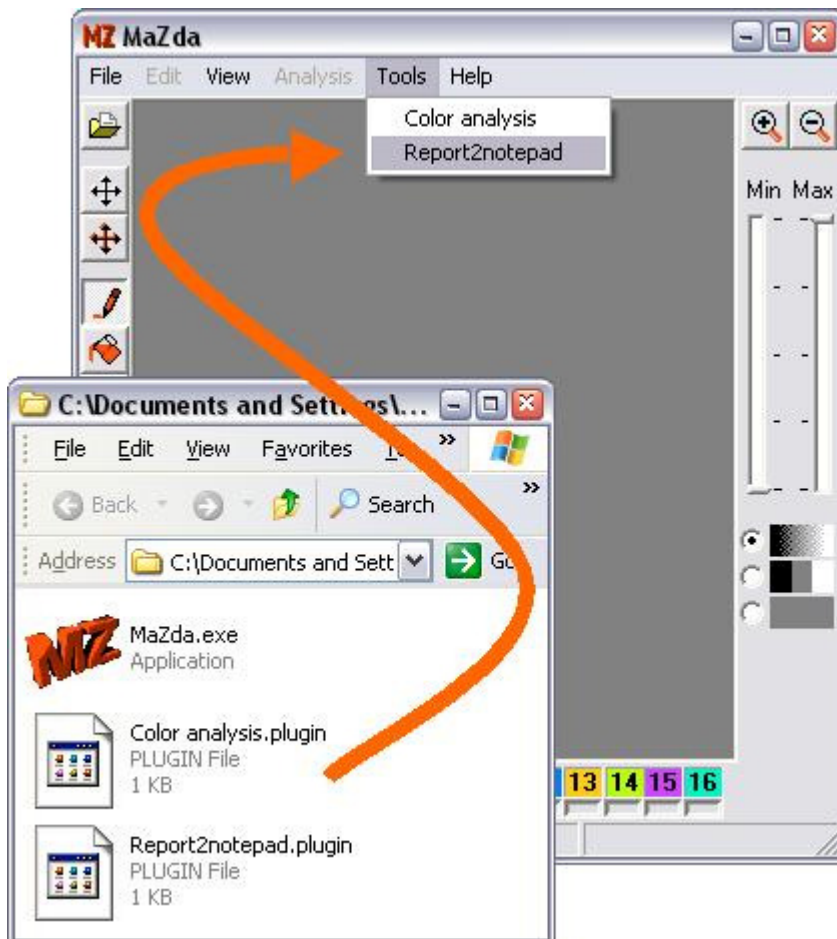


Fig. 2.6.1 MaZda menu extended with plugins

Examples:

```
// Load_and_analyze.txt - MaZda macro example
// The macro loads image, region of interest and options set,
// performs the textural analysis and saves its results.
LoadImage siem2.dcm
LoadROI testroi.roi
LoadOptions options1.ini
RunAnalysis
SaveReport result1.par
SaveMap map3.bmf
CloseMap
SaveMap map2.bmf
CloseMap
SaveMap map1.bmf
CloseMap
```

```
// View_in_notepad.txt - MaZda macro example
// Simple macro that saves a report to a file
// and opens it with a notepad editor.
SaveReport temp.par
Execute notepad.exe temp.par
```

```
// Color_analysis.plugin - MaZda plugin example
// The plugin analyzes color image YUV channels.
// It joins all the computed features within a single report.
ColorChannel Y
```

ReloadImage
RunAnalysis
ColorChannel U
ReloadImage
RunAnalysis
AggregateReports
ColorChannel V
ReloadImage
RunAnalysis
AggregateReports
ColorChannel Y
ReloadImage

MaZda User's Manual

3. Theoretical background

MaZda User's Manual

3.1. Texture analysis methods

A literature survey, on main techniques of image texture analysis has been reported by A. Materka and M. Strzelecki in Brussels, during one of the COST B11 workshops in 1998. It is available in pdf form at http://www.eletel.p.lodz.pl/cost/pdf_1.pdf as **Texture analysis methods - a review**.

MaZda User's Manual

3.2. Feature reduction methods

The **MaZda** software generates almost 300 texture parameters. With such a large number of features, it is very difficult to predict, which parameters will be most useful for texture classification. Also, the large number of features requires very large number of data samples to provide reliable discriminant analysis results, from the statistical point of view. The required large number of samples are not normally available. Finally, almost 300 features are difficult to manage (for presentation, displaying, analysis). For these reasons, there is a need for feature reduction, to provide a compact parameter set useful for texture discrimination and classification.

Generally, there are two approaches for feature reduction. First, called feature selection, is based on choosing of some features according to given mathematical criterion. As a consequence, a subset of features is obtained, which best satisfies such a criterion. Examples of feature selection methods are the one based on Fisher criterion and another one, based on minimisation of classification error and correlation coefficient (POE+ACC). Both methods are further described in this chapter.

Yet different feature reduction approach is called feature extraction or projection. In this case, the original feature space is transformed into a different space, of lower dimension compared to the input space. This provides a set of new features, smaller in number than the original feature set. Feature extraction techniques of this type, namely principal component analysis (PCA), linear discriminant analysis (LDA) and nonlinear discriminant analysis (NDA) are implemented in **B11** program and described in [B11 help file](#).

Optimal subsets

This feature selection method is based on selection of feature subsets (from the whole feature set), which can contain one, two or three features. These subsets are optimal in the sense that they provide minimal classification error when 1-nearest neighbor (1-NN) classifier is used.

Using this **MaZda** option, up to 10 features (feature pairs or feature triples) with the lowest 1-NN

classification error can be selected. To activate this option, one should use **Feature selection=>Optimal subsets=>Single feature/Feature pairs/Feature triples** menu item in the **Report** window.

Fisher coefficient

This feature selection method is based on Fisher coefficient. It is defined as a ratio of between-class variance to within-class variance (Schürmann 1996)

$$F = \frac{D}{V} = \frac{\frac{1}{K} \sum_{k=1}^K \sum_{j=1}^K P_k P_j (\mu_k - \mu_j)^2}{1 - \sum_{k=1}^K P_k^2 - \sum_{k=1}^K P_k V_k}$$

where D denotes between-class scatter, V - within-class variance, μ_i and V_i - mean and variance of class i , P_i - probability of class i (ratio of number of data samples from class i to the number of all data samples).

Using this **MaZda** option, up to 10 features with the largest Fisher coefficient can be selected. To activate this option, one should use **Feature selection=>Fisher** menu item in the **Report** window.

Low POE+ACC feature selection method

This method is based on minimisation of both classification error probability (POE) and average correlation coefficients (ACC) between chosen features (Mucciardi 1971, Dash 1997). The first feature, f^1 , is chosen to minimise POE for all classes:

$$f^1 = f_j : \min_j [POE(f_j)]$$

where $POE(f_j)$ is the probability of classification error for feature f_j . This probability is defined as a ratio of a number of not properly classified samples to the whole number of samples in the analysed data set, using only feature f_j for classification. Fig. 3.2.1 shows sample distribution of feature f_j for two classes. Samples of the feature f_j marked in blue belong to class 1 while the samples marked in red belong to class 2. The samples marked in black can be assigned both to class 1 or 2. In this case, these samples can not be properly classified. The POE calculated for this feature is given by the following equation:

$$POE(f_j) = \frac{\text{(number of samples not properly classified, marked in black)}}{\text{(total number of samples)}}$$

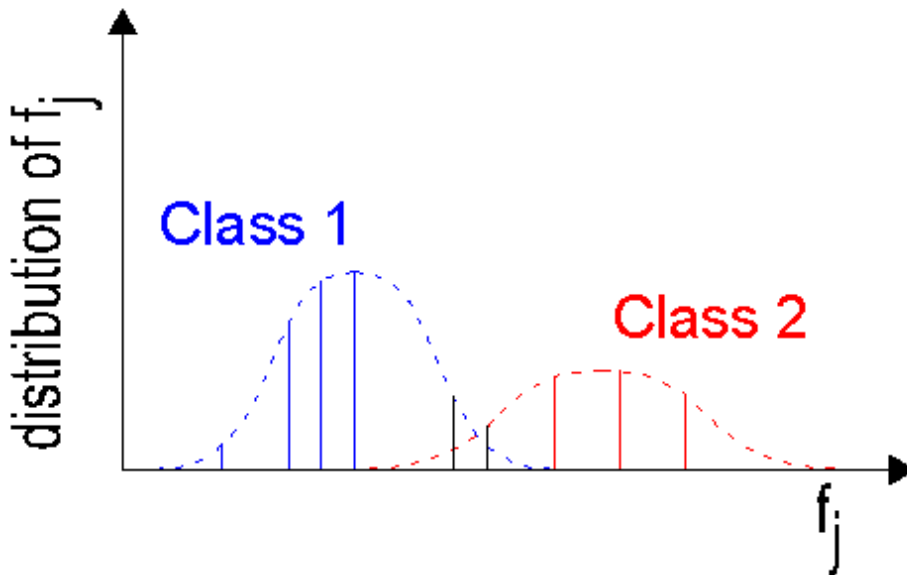


Fig. 3.2.1 A sample distribution of feature f_j

The next feature is then selected by minimising the following sum for all features, excluding f_j :

$$f^2 = f_j : \min_j [POE(f_j) + |CC(f^1, f_j)|]$$

The $|CC(f^l, f_j)|$ is the absolute value of correlation coefficient between previously chosen feature f^l and the new feature f_j . The n -th feature is chosen by minimising the following sum for all remaining features (except for the features already selected):

$$f^n = f_j : \min_j [POE(f_j) + \frac{1}{n-1} \sum_{k=1}^{n-1} |CC(f^k, f_j)|] = \min_j [POE(f_j) + ACC(f_j)]$$

where the averaged sum is extended for correlation coefficients between previously selected features and feature f_j . This sum is called an average correlation coefficient (ACC).

To control influence of POE and ACC components on the feature selection process, two weight values can be added to the previous formula:

$$f^n = \min_j [w_1 POE(f_j) + w_2 ACC(f_j)]$$

In the present version of **Convert** module implemented in **MaZda** it is assumed $w_1 = w_2 = 0.5$.

Using this technique, features with the lowest POE+ACC coefficient will be selected. To activate this option in **MaZda**, one has to select **Feature selection=>POE+ACC** menu item in **Report** window.

Mutual information measure

Mutual information is a measure of dependence between two or more random variables. In the case of two random variables X and Y , it is defined as follows (Tourassi 2001):

$$MI(X, Y) = H(X) + H(Y) - H(X, Y)$$

where H is the entropy. It can be considered, that random variables X stores values of texture features, while Y stores the classification decision. Then, large value of mutual information between texture feature X and class category variable Y means that this feature carries information about class membership (its values are correlated with given class categories). In consequence, such a feature is useful for classification. On the other hand, when class category does not depend on the feature value X , mutual information equals to zero because $H(X,Y)=H(X)+H(Y)$ for independent variables.

In **MaZda**, the mutual information for each feature f_j is estimated according to the formula (Tourassi 2001, Kwak 2002):

$$MI(f_j, d) = \sum_{d=1}^{Nb} \sum_{k=1}^{Nc} P(f_j^d, c_k) \log_2 \left[\frac{P(f_j^d, c_k)}{P(f_j^d)P(c_k)} \right]$$

where P means the probability, f_j^d is discretised value of feature f_j , Nb is a number of histogram bins used for feature discretisation., $d=\{c_1, c_2, \dots, c_{Nc}\}$ means the class category, and Nc is the total number of categories (classes). In this version of Mazda Nb is constant and equals to $\text{round}[\log_2(M)+1]$, where M is a total number of samples of feature f_j (Tourassi 2001).

Using this **MaZda** option, up to 10 features with the largest Mutual Information coefficient can be selected. To activate this option, one should use **Feature selection=>Mutual Information** menu item in the **Report** window.

MaZda User's Manual

3.3. Texture parameters summary

MaZda allows computation of a variety of statistical parameters that are derived from image histogram, absolute gradient, run-length matrix co-occurrence matrix and autoregressive model (Table 3.3.1). One of the latest additions to **MaZda** are texture parameters derived from wavelet analysis.

Table 3.3.1. Texture parameters computed by **MaZda**

Histogram	Absolute gradient	Run-length matrix	Co-occurrence matrix	AR model
<ul style="list-style-type: none"> • mean • variance • skewness • kurtosis, • 1-% percentile • 10-% percentile • 50-% percentile • 90-% percentile • 99-% percentile 	<ul style="list-style-type: none"> • mean • variance • skewness • kurtosis • percentage of pixels with nonzero gradient. 	<ul style="list-style-type: none"> • run length nonuniformity • grey level nonuniformity • long run emphasis • short run emphasis • fraction of image in runs 	<ul style="list-style-type: none"> • angular second moment • contrast • correlation • sum of squares • inverse difference moment • sum average • sum variance • sum entropy • entropy • difference variance • difference entropy 	<ul style="list-style-type: none"> • θ_1 • θ_2 • θ_3 • θ_4 • σ

The run-length matrix-based parameters are computed 4 times for each ROI (for vertical, horizontal, 45-

degree and 135-degree directions). The co-occurrence matrix-based parameters are [computed up to 20 times, for $(d,0)$, $(0,d)$, (d,d) , $(d,-d)$ where the distance d can take values of 1, 2, 3, 4, and 5].

Definition and discussion of the above parameters can be found in (Haralick 1973, Haralick 1979, Hu 1994, Lerski 1993). A quick texture parameter reference is provided below for the user convenience.

Note

To keep consistency with the formulas used in the standard references (Haralick 1973, Haralick 1979) on texture analysis, it is assumed in MaZda that the intensity of image under analysis changes from 1 to N_g , where $N_g = 2^k$, and k is the number of bits per pixel. Thus if originally the image intensity changes from 0 to N_g-1 , MaZda converts this image internally, such that its intensity changes from 1 to N_g . Consequently, the summation indices in the formulas listed below span the range from 1 to N_g .

3.3.1 Histogram-based features

In the formulas that follow, $p(i)$ is a normalized histogram vector (i.e. histogram whose entries are divided by the total number of pixels in ROI), $i=1,2,\dots, N_g$, and N_g denotes the number of intensity levels.

$$\mu = \sum_{i=1}^{N_g} i p(i)$$

Mean:

$$\text{Variance: } \sigma^2 = \sum_{i=1}^{N_g} (i - \mu)^2 p(i)$$

$$\text{Skewness: } \mu_3 = \sigma^{-3} \sum_{i=1}^{N_g} (i - \mu)^3 p(i)$$

$$\text{Kurtosis: } \mu_4 = \sigma^{-4} \sum_{i=1}^{N_g} (i - \mu)^4 p(i) - 3$$

3.3.2 Gradient-based parameters

For the gradient feature calculation the following neighborhood for image pixel $x(i,j)$ is defined:

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>	<i>J</i>
<i>K</i>	<i>L</i>	$x(i,j)$	<i>N</i>	<i>O</i>
<i>P</i>	<i>Q</i>	<i>R</i>	<i>S</i>	<i>T</i>
<i>U</i>	<i>V</i>	<i>W</i>	<i>Y</i>	<i>Z</i>

Based on this neighborhood, the absolute gradient value ($ABSV(i,j)$) is calculated for each pixel:

a) for 5x5 pixel neighborhood: $ABSV5(i,j) = \sqrt{(W - C)^2 + (O - K)^2}$

b) for 3x3 pixel neighborhood: $ABS\bar{V}3(i, j) = \sqrt{(R - H)^2 + (N - L)^2}$

The $ABS\bar{V}3$ definition is used in this version of **MaZda**. For the $ABS\bar{V}=ABS\bar{V}3$ matrix of M elements (which contains absolute gradient values for ROI pixels), the gradient features are defined as follows:

Mean absolute gradient:

$$GrMean = \frac{1}{M} \sum_{i,j \in ROI} ABS\bar{V}(i, j)$$

Variance of absolute gradient:

$$GrVariance = \frac{1}{M} \sum_{i,j \in ROI} (ABS\bar{V}(i, j) - GrMean)^2$$

Skewness of absolute gradient:

$$GrSkewness = \frac{1}{(\sqrt{GrVariance})^3} \frac{1}{M} \sum_{i,j \in ROI} (ABS\bar{V}(i, j) - GrMean)^3$$

Kurtosis of absolute gradient:

$$GrKurtosis = \frac{1}{(\sqrt{GrVariance})^4} \frac{1}{M} \sum_{i,j \in ROI} (ABS\bar{V}(i, j) - GrMean)^4 - 3$$

where ROI is a region of interest.

Percentage of non-zero $ABS\bar{V}$ matrix elements (**Grads>0**)

3.3.3 Run length matrix-based parameters

Let $p(i, j)$ be the number of times there is a run of length j having grey level i . Let N_g be the number of grey levels and N_r be the number of runs. Definitions of the parameters of the run-length matrix $p(i, j)$, as adopted in **MaZda**, are given below.

Short run emphasis inverse moments:

$$ShrtREmph = \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \frac{p(i, j)}{j^2} \right) / C$$

Long run emphasis moments:

$$LngREmph = \left(\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} j^2 p(i, j) \right) / C$$

Grey level nonuniformity:

$$\text{GLevNonUni} = \left(\sum_{i=1}^{N_g} \left(\sum_{j=1}^{N_g} p(i,j) \right)^2 \right) / C$$

Run length nonuniformity:

$$\text{RLNonUni} = \left(\sum_{j=1}^{N_r} \left(\sum_{i=1}^{N_g} p(i,j) \right)^2 \right) / C$$

Fraction of image in runs:

$$\text{Fraction} = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} p(i,j)}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} j p(i,j)}$$

The coefficient C is defined as

$$C = \sum_{i=1}^{N_g} \sum_{j=1}^{N_r} p(i,j)$$

The above 5 features are calculated for 4 directions: horizontal (Horzl_), vertical (Vertl_), slanted at 45 degrees (45dgr_) and slanted at 135 degrees (135dgr_). The RL feature name in **MaZda** software contains a prefix that defines direction and feature type, for example 45dgr_RLNonUni means the run length nonuniformity calculated for 45-degree direction.

3.3.4 Co-occurrence matrix-derived parameters

The second-order histogram is defined as the co-occurrence matrix $h_{d\theta}(i,j)$ [1]. When divided by the total number of neighboring pixels $R(d,\theta)$ in ROI, this matrix becomes the estimate of the joint probability, $p_{d\theta}(i,j)$, of two pixels, a distance d apart along a given direction θ having particular (co-occurring) values i and j . Formally, given the image $f(x,y)$ with a set of N_g discrete intensity levels, the matrix $h_{d\theta}(i,j)$ is defined such that its (i,j) th entry is equal to the number of times that

$$f(x_1, y_1) = i \text{ and } f(x_2, y_2) = j,$$

$$\text{where } (x_2, y_2) = (x_1, y_1) + (d \cos \theta, d \sin \theta).$$

This yields a square matrix of dimension equal to the number of intensity levels in the image, for each distance d and orientation θ . In **MaZda**, the distances $d = 1, 2, 3, 4$ and 5 pixels with angles $\theta = 0^\circ, 45^\circ, 90^\circ$ and 135° are considered. Reduction of the number of intensity levels (by quantization to fewer levels of intensity) helps increase the speed of computation, with some loss of textural information.

The co-occurrence matrix-derived parameters computed by **MaZda** are defined by the equations that follow, where μ_x, μ_y and σ_x, σ_y denote the mean and standard deviations of the row and column sums of the co-occurrence matrix, respectively [related to the marginal distributions $p_x(i)$ and $p_y(j)$].

Angular second moment:

$$\text{AngScMom} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)^2$$

Contrast:

$$\text{Contrast} = \sum_{x=0}^{N_x-1} x^2 \sum_{\substack{i=1 \\ |i-j|=x}}^{N_x} \sum_{j=1}^{N_x} p(i, j)$$

Correlation:

$$\text{Correlat} = \frac{\sum_{i=1}^{N_x} \sum_{j=1}^{N_x} ij p(i, j) - \mu_x \mu_y}{\rho_x \rho_y}$$

Sum of squares:

$$\text{SumOfSqs} = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} (i - \mu_x)^2 p(i, j)$$

Inverse difference moment:

$$\text{InvDfMom} = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} \frac{1}{1+(i-j)^2} p(i, j)$$

Sum average:

$$\text{SumAverg} = \sum_{i=1}^{2N_x} i p_{x+y}(i),$$

where

$$p_{x+y}(k) = \sum_{\substack{i=1 \\ i+j=k}}^{N_x} \sum_{j=1}^{N_x} p(i, j) \quad k = 2, 3, \dots, 2N_x$$

Sum variance:

$$\text{SumVarnc} = \sum_{i=1}^{2N_x} (i - \text{SumAverg})^2 p_{x+y}(i)$$

Sum entropy:

$$\text{SumEntrp} = - \sum_{i=1}^{2N_x} p_{x+y}(i) \log(p_{x+y}(i))$$

Entropy:

$$\text{Entropy} = - \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \log(p(i, j))$$

Difference variance:

$$\text{DifVarnc} = \sum_{i=0}^{N_x-1} (i - \mu_{x-y})^2 p_{x-y}(i)$$

where μ_{x-y} is a mean value of difference distribution p_{x-y} :

$$p_{x-y}(k) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_x} p(i, j) \quad k = 0, 1, \dots, N_x - 1$$

$$|i - j| = k$$

Difference entropy:

$$\text{DifEntrp} = - \sum_{i=1}^{N_x} p_{x-y}(i) \log(p_{x-y}(i))$$

3.3.5 Autoregressive model parameters

The autoregressive (AR) model assumes a local interaction between image pixels in that pixel intensity is a weighted sum of neighbouring pixel intensities. Assuming image f is a zero-mean random field, an AR causal model can be defined as

$$f_s = \sum_{r \in N_s} \theta_r f_r + e_s$$

where f_s is image intensity at site s , e_s denotes an independent and identically distributed (i.i.d.) noise, N_s is a neighbourhood of s , and θ is a vector of model parameters. The local neighbourhood for AR model implemented in **MaZda**, represented by 4 parameters, is shown in Fig. 3.3.1. Shaded area in Fig. 3.3.1 indicates region where valid causal half-plane AR model neighbourhood may be located, in general.

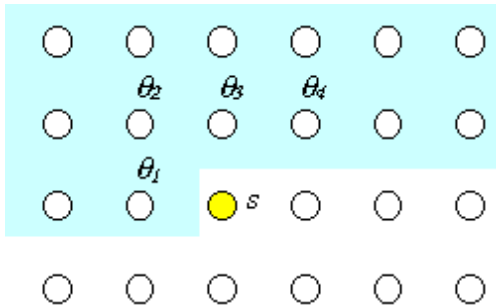


Fig. 3.3.1. Local neighbourhood of image element f_s

Using the AR model for image segmentation consists in identifying the model parameters for a given image region and then using the obtained parameter values for texture discrimination. In the case of simple pixel neighbourhood shown in Fig. 3.3.1, that comprises 4 immediate pixel neighbours, there are 5 unknown model parameters – the standard deviation σ of the driving noise e_s and the model parameter vector $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$. The parameters can be estimated by minimizing the sum of squared error

$$\sum_s e_s^2 = \sum_s (f_s - \hat{f}_s)^2$$

which leads to the following linear equations:

$$\hat{\theta} = \left(\sum_s \mathbf{w}_s \mathbf{w}_s^T \right)^{-1} \left(\sum_s \mathbf{w}_s f_s \right) \quad \sigma^2 = N^{-2} \sum_s (f_s - \hat{\theta} \mathbf{w}_s)^2$$

where $\mathbf{w}_s = \text{col}[f_i, i \in N_s]$, and the square $N \times N$ image is assumed. The above set of equations is numerically solved in **MaZda** for each ROI of interest.

As an example, AR model parameters were used in [5] for unsupervised texture segmentation.

3.3.6 Wavelet parameters

The discrete wavelet transform (DWT) is a linear transformation that operates on a data vector whose length is an integer power of two, transforming it into a numerically different vector of the same length. It is a tool that separates data into different frequency components, and then studies each component with resolution matched to its scale. DWT is computed with a cascade of filters followed by a factor 2 subsampling (Fig 3.3.2).

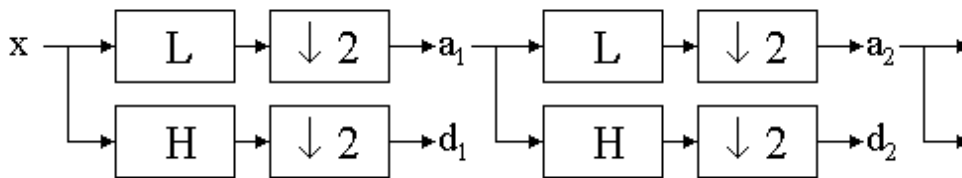


Fig. 3.3.2 DWT tree.

H and L denotes high and low-pass filters respectively. Boxes after each filter denote subsampling. Outputs of these filters are given by the following equations

$$a_{j+1}[p] = \sum_{n=-\infty}^{+\infty} l[n-2p] a_j[n]$$

$$d_{j+1}[p] = \sum_{n=-\infty}^{+\infty} h[n-2p] a_j[n]$$

Elements a_j are used for next step (scale) of the transform and elements d_j , called wavelet coefficients, determine output of the transform. $l[n]$ and $h[n]$ are coefficients of low and high-pass filters, respectively. One can assume that on scale $j+1$ there is only half of the number of a and d elements at scale j . This causes that DWT can be done until only two a_j elements remain in the analyzed signal. These elements are called scaling function coefficients.

DWT algorithm for pictures is similar. The DWT is performed firstly for all image rows and then for all columns (Fig.3.3.3).

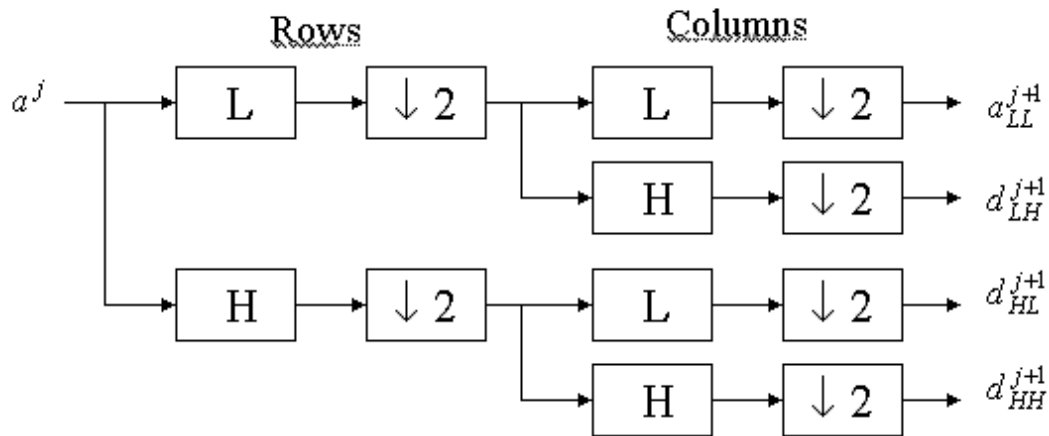


Fig. 3.3.3 Wavelet decomposition for two-dimensional pictures.

In this version of **MaZda**, only one set of DWT derived features is considered. It is a vector, which contains energies of wavelet coefficients calculated in subbands at successive scales.

To compute the wavelet features in the first step, Harr wavelet is calculated for whole image. As a result of this transform there are 4 subband images at each scale (Fig.3.3.4).

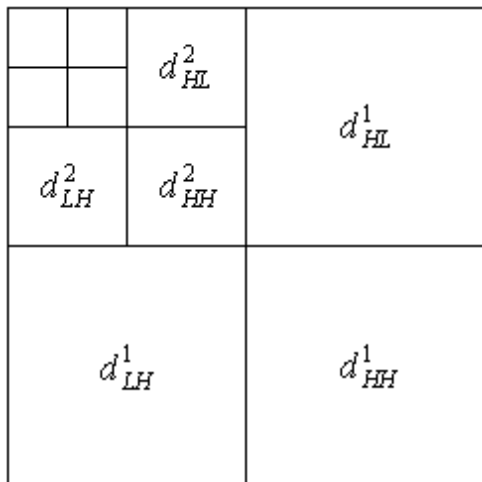


Fig. 3.3.4 Subband images.

Subband image a^{LL} is used for DWT calculation at the next scale.

For a given image, the maximum of 8 scales can be calculated by **MaZda**. The Harr wavelet is calculated only if output subbands have dimensions at least 8 by 8 points.

In the next step, energy of a^{LL} , d^{LH} , d^{HL} and d^{HH} is calculated at any considered scale in marked ROIs.

$$E_{subband, scale} = \frac{\sum_{x,y \in ROI} (a_{x,y}^{subband})^2}{n}$$

where n is the number of pixels in ROI, both at given scale and subband.

Of course, ROIs are reduced in size in successive scales in order to correspond to subband image dimensions. In a given scale the energy is calculated only if ROI at this scale contains at least 4 points.

Output of this procedure is a vector of features containing energies of wavelet coefficients calculated in subbands at successive scales.

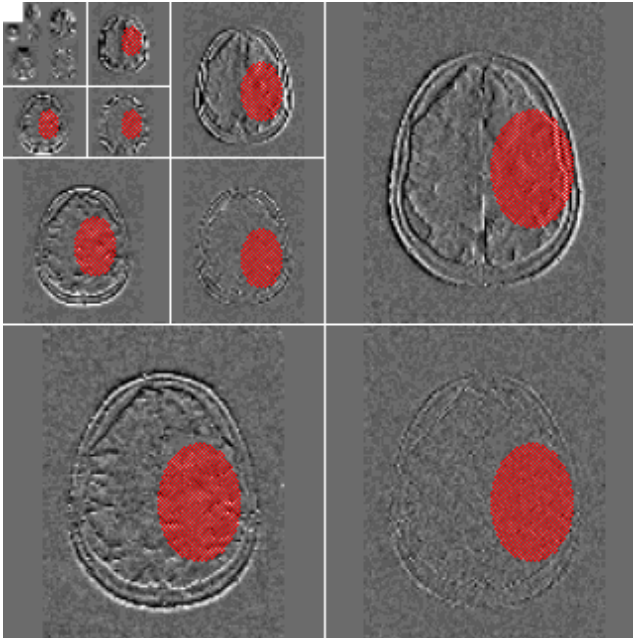


Fig. 3.3.5 Example of ROI reduced in successive scales

MaZda User's Manual

B11 program for data analysis and classification

B11 - program for data analysis and classification

4.1. Introduction

Program **b11** has been developed for quantitative analysis of magnetic resonance images. It is devised to be one of the software tools for carrying out research within the framework of COST B11 European project "Quantitation of Magnetic Resonance Image Texture" (1998-2002).

It is assumed that any magnetic resonance image (MRI) investigated comprises a number of nonoverlapping, homogeneous regions. The regions differ from each other in terms of their *texture*.

Apart from its perceptual properties, such as coarseness, granulation, regularity, etc., the image texture can be quantitatively described by means of local statistics of the image. These statistics, e.g. parameters derived from the so-called co-occurrence matrix, are the texture *features*. A program called **MaZda** was developed to compute more than 250 different features for an image. Feature values obtained from **MaZda** can be stored in a text file.

Usually, only a small fraction of features calculated by **MaZda** can be used to distinguish textures present in an investigated image. The remaining features do not carry the discriminative information with regard to the textures at hand. Also, it is very time-consuming and difficult to analyse feature vectors in a high-dimensional space. Therefore, yet another program, **Convert**, is used to pre-select a subset of features (e.g. no more than 10) from **MaZda** output file to form a low-dimensional feature vector, for more detailed analysis by means of **b11**. In **MaZda** version 3, **Convert** has been integrated with **MaZda** for more convenient parameter set [selection and reduction](#).

The input to **b11** program is a text file (produced either by **Convert**, produced automatically by **MaZda** or prepared manually) that contains feature vectors, each representing a different category (class) of textures. The program investigates this input-data file to assess the features ability to distinguish the texture categories. The Fisher coefficient is calculated to quantitatively describe discriminative power of the features. Three feature vector transformations are available in **b11** to transform the data to a new feature space, in order to either reduce the feature vector dimension or increase the discriminative power. These are principal component analysis ([PCA](#)), linear discriminant analysis ([LDA](#)) and nonlinear discriminant analysis ([NDA](#)).

Program **b11** also allows performing [classification](#) tests on the input data using different classifiers, such as *k-NN* classifier.

The three programs being developed within COST B11 (**MaZda**, **Convert** and **b11**) will be integrated in future to made a single software tool for MRI texture analysis. Their independent development helps better utilisation of limited human resources within the 'Software and Statistics' working group of the COST B11 action.

B11 - program for data analysis and classification

4.2. Input and output data

The input to **b11** program is a text file of predefined structure. Such a file contains data organized in four logical blocks: label, feature numbers and names, category numbers and names, and feature values. An example of the contents of an input file (**m25.sel**) is presented in Fig. 4.2.1. The **m25.sel** file contains 25 feature vectors, each of 5 elements long. The feature vectors represent 5 data categories, each category represented by 5 vectors.

The ***label** keyword (identifier) must be placed in the first line of the file. Second line contains the user-defined label (tag) attached to the file, which is ignored by **b11**.

The third line contains the ***features** keyword. Then a list of feature names follows, such that each line starts with a number of the feature and its name.

The ***categories** identifier starts the next block, which contains all the categories listed, line by line, number preceding the name.

Feature vectors come after the ***data** identifier. First element of each row is the number of category (class). Elements of the feature vector are then placed in each row, in an ascending order (feature #1 placed in column 2, feature #2 in column 3, etc.).

The ***end** identifier must be placed at the end of the input file. Contents and structure of input files are checked for some errors during loading.

```
AR model, phantom image analysis
*features
1 teta1
2 teta2
3 teta3
4 teta4
5 sigma
*categories
1 Large-size bubbles
2 Glass beads
3 Medium-size bubbles
4 Small-size bubbles
5 Background noise
*data
1 3.004e-1 -5.460e-2 4.466e-1 4.100e-2 8.374e-1
1 2.936e-1 -4.150e-2 4.428e-1 7.440e-2 8.198e-1
1 3.630e-1 -7.520e-2 4.042e-1 5.020e-2 7.922e-1
1 3.140e-1 -9.010e-2 4.068e-1 8.180e-2 8.132e-1
1 3.827e-1 -1.067e-1 4.486e-1 2.760e-2 8.016e-1
2 3.273e-1 -1.175e-1 4.293e-1 5.090e-2 8.315e-1
2 3.419e-1 -1.748e-1 4.454e-1 4.960e-2 8.270e-1
2 3.574e-1 -1.415e-1 4.279e-1 4.880e-2 8.296e-1
2 3.121e-1 -1.729e-1 4.648e-1 2.280e-2 8.236e-1
2 2.848e-1 -7.850e-2 4.431e-1 1.450e-2 8.491e-1
3 9.310e-2 -2.400e-2 3.770e-1 2.350e-2 9.384e-1
3 1.054e-1 -2.920e-2 2.856e-1 -7.700e-2 9.216e-1
3 4.530e-2 -2.030e-2 3.093e-1 -2.280e-2 9.426e-1
3 1.096e-1 -1.930e-2 2.278e-1 4.620e-2 9.555e-1
3 9.190e-2 -2.640e-2 2.658e-1 -7.700e-3 9.551e-1
4 -8.750e-2 -1.050e-2 1.741e-1 -3.280e-2 9.845e-1
4 -9.280e-2 -2.600e-2 2.305e-1 1.600e-3 9.473e-1
4 -1.048e-1 2.150e-2 2.021e-1 2.910e-2 9.799e-1
4 -6.150e-2 5.040e-2 1.992e-1 -9.100e-3 9.643e-1
4 8.900e-3 3.520e-2 2.771e-1 4.540e-2 9.278e-1
5 6.900e-3 -2.490e-2 5.270e-2 3.800e-3 9.851e-1
5 -8.500e-3 4.260e-2 1.880e-2 4.790e-2 9.962e-1
5 1.270e-2 -1.590e-2 -4.290e-2 -5.250e-2 9.868e-1
5 6.300e-3 4.620e-2 5.210e-2 2.310e-2 1.002e+0
5 -4.000e-3 -2.500e-2 1.000e-3 3.700e-2 9.990e-1
*end
```

Fig. 4.2.1. Contents of m25.sel input file.

After loading any input file, its text is displayed in the left (input) panel of the main window of **b11** program. (It is displayed using blue-color font if the file contains errors.)

Depending on [analysis](#) or [classification](#) procedure selected, **b11** produces an output report which is displayed in the right-hand panel of the main program window. Reports can be saved as text files. The extension part of a file-name corresponds to analysis procedure that produced that file:

- ***.rda**: raw-data analysis report,
- ***.pca**: principal component analysis report,
- ***.lda**: linear discriminant analysis report,
- ***.nda**: nonlinear discriminant analysis report,
- ***.cls**: classification report.

The ***.nda** files, as well as those ***.cls** files which were generated by neural-network classifiers, contain the information about architecture and weights of neural networks. These files are needed for [testing](#) of classifiers trained in, respectively, **Analysis|NDA**, **Classification|ANN: one-class (training)** and **Classification|ANN: n-class (training)** options.

B11 - program for data analysis and classification

4.3. Getting started with b11

The main window of **b11** is displayed on the computer screen once the program is executed. It contains a menu bar with **Files, Options, Analysis, Classification, About, Exit** and **Help** menu items, and two panels for displaying processed information. The left-hand (input) panel will contain the input data. Reports from analysis/classification will be displayed in the right-hand (output) panel.

Load the supplied **m25.sel** input file (use **Files|Get Input File** menu items and go through a standard MS Windows 'load file' dialog). The contents of the file will appear in the left-hand panel of the main window of **b11** (Fig. 4.3.1).

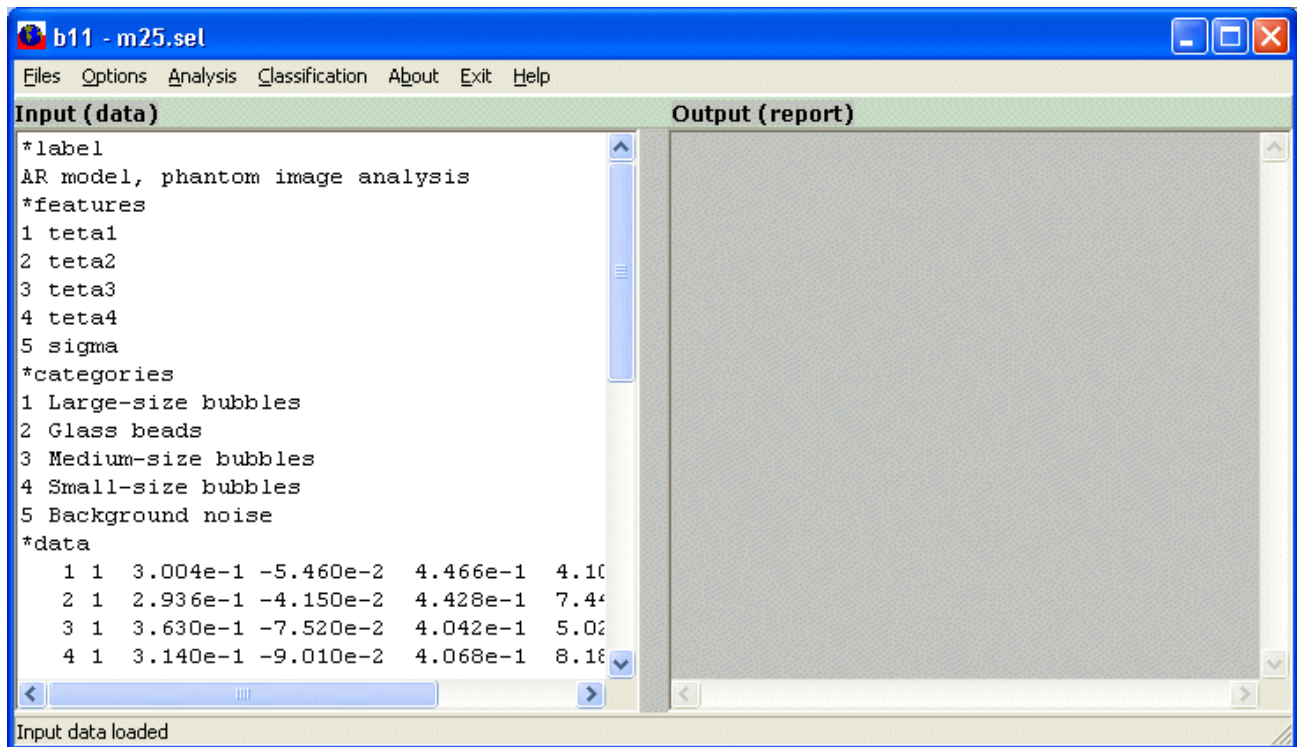


Fig. 4.3.1. The main window of **b11** after loading the **m25.sel** file.

Now select **Analysis|Raw data** menu items. A report of the analysis of the input data will be displayed in the right-hand panel of the main window (Fig. 4.3.2), and a graphic window will appear on the screen (Fig. 4.3.3). The plot shows 5 data clusters in the (*teta1*, *teta2*, *teta3*) space, corresponding to 5 data categories defined in the input file. Each number on the plot represents a unique feature vector and the category it belongs to. Click the **Flip "teta2" axis** menu item of the graphics window. Now the plot will show the clusters from another perspective, to better visualize the data categories (Fig. 4.3.4).

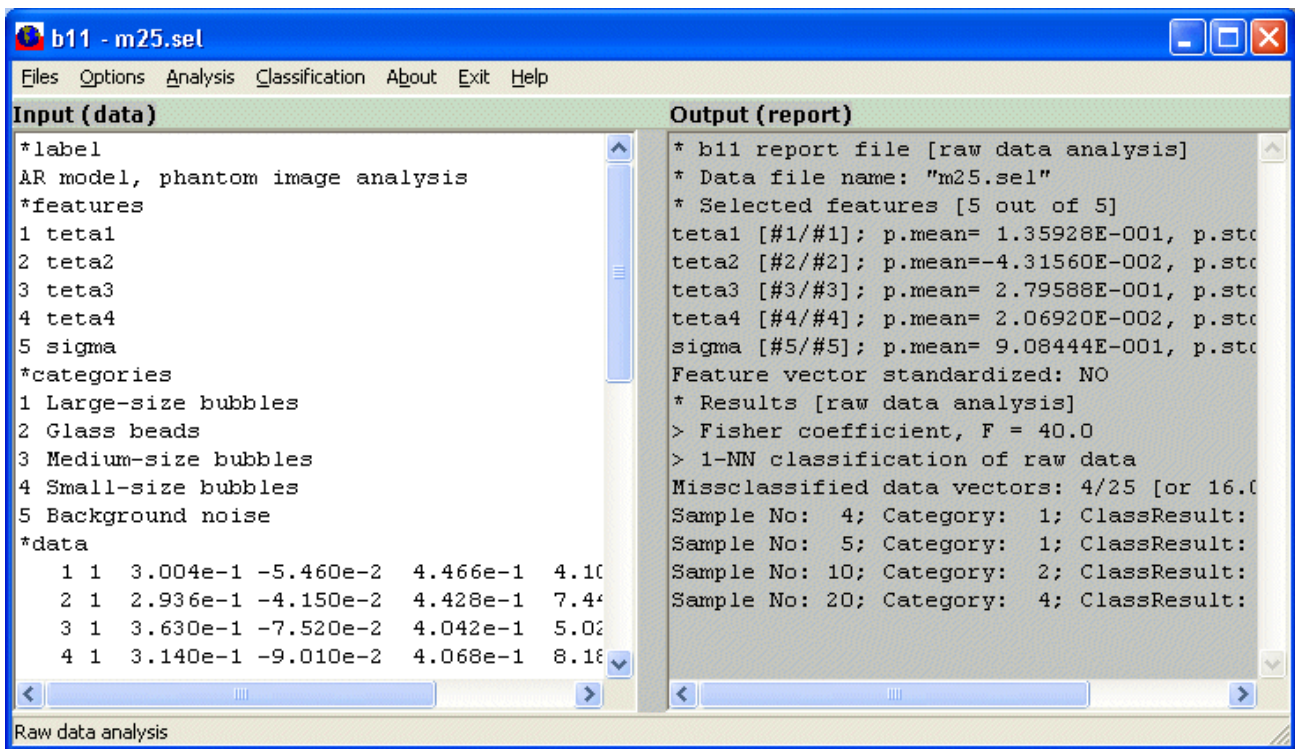


Fig. 4.3.2. Main window of b11 after running 'Raw Data' analysis of m25.sel file (all features enabled).

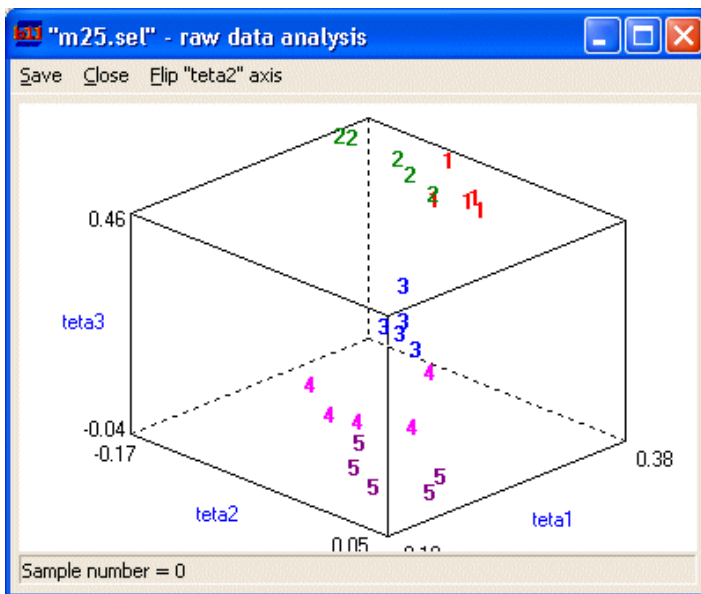


Fig. 4.3.3. Clusters of data vectors in (teta1,teta2,teta3) space for m25.sel file.

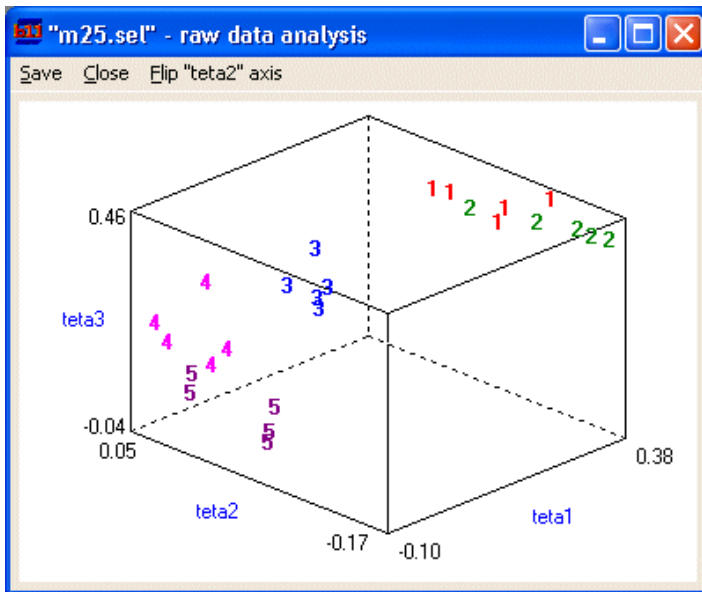


Fig. 4.3.4. Clusters of data vectors in (teta1,teta2,teta3) space for m25.sel file (flipped teta2 axis).

The report displayed in the right-hand panel of the main window of **b11** can be saved on disk by using **Files|Save Report** main menu items.

Other [analysis](#) and [classification](#) procedures can be invoked by using appropriate menu items.

B11 - program for data analysis and classification

4.4. Analysis

The maximum number of features that can be processed by **b11** is limited to 30. By using **Options** menu item, and disabling some of the features present in input file, one can further reduce the number of features considered for analysis to N_x , which however can not be smaller than 2. Each N_x -dimensional feature vector $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{iN_x}]$ is allocated to feature category d_i , where $d_i \in \{c_1, c_2, \dots, c_{N_c}\}$ and N_c is the total number of categories (classes). The maximum number of categories is 16 in the present version of **b11**. The number M of pairs $\{\mathbf{x}_i, d_i\}$, $i=1,2,\dots,M$ is limited to $M_{\max}=2000$. The examples $\{\mathbf{x}_i, d_i\}$, $i=1,2,\dots,M$, so-called *patterns*, form the *training set* for classifiers, and M is the size of this set.

Assume there are M_k , $k=1,2,\dots,N_c$ training patterns for category c_k , such that $M_1+M_2+\dots+M_{N_c}=M$. Mean $\boldsymbol{\mu}^{(k)}$ and standard deviation $\boldsymbol{\sigma}^{(k)}$ vectors are computed for each category. Pooled mean $\boldsymbol{\mu}$ and pooled standard deviation $\boldsymbol{\sigma}$ are also calculated using all M patterns, and applied for feature standardization. Let \mathbf{x}'_i be a standardized feature vector. Its p -th element, $p=1,2,\dots,N_x$ is computed as follows

$$x'_{ip} = \frac{x_{ip} - \mu_p^{(k)}}{\sigma_p}$$

To evaluate the selected features ability to discriminate the categories of interest, Fisher F coefficient is computed for the training set. It is defined ([Shürmann 1996](#)) as the ratio of mean-squared between-class distance D^2 (between the class means) to the mean of mean-squared within-class distances V_k^2 (between the samples \mathbf{x} of class k and the corresponding class mean $\boldsymbol{\mu}^{(k)}$).

4.4.1. Raw Data Analysis

For this analysis type, Fisher coefficient is calculated for the original data included in the input file. The features can be standardized or not, depending on user choice (options window).

Raw data analysis example

Load the supplied **m25.sel** input file as described in [Getting started](#) help section. Click **Options** menu item and disable *teta2* and *teta4* features (Fig. 6.2. in [Options](#) help section). Close options window and activate **Analysis|Raw Data** menu items. Close graphics window. The report file will be displayed in the right-hand panel of the program main window (Fig. 4.1). It can be saved on disk by using **Files|Save Report** menu items.

The first line of the report file contains the header and information about the type of analysis (Fig. 4.1). Input file name is included in the second line. Third line gives information about the number of features enabled for analysis. These features are listed in the subsequent lines (3 lines in the present example). Each line of this type contains feature name (e.g. *sigma*), its number in the reduced feature set (#3), the number in the input file data set (#5), pooled mean (*p.mean*), and pooled standard deviation (*p.std*). In the following line of the report file (7 in this example), information is displayed about feature normalization prior to analysis. Finally, the value of Fisher coefficient is displayed in the last line.

Comparing Fig. 3.2 in [Getting started](#) with Fig. 4.1, one can notice that the selection of features affects the Fisher coefficient. In both examples data were not normalized. The value of $F = 40.0$ was obtained for 5 features, whereas F increased to 66.5 when 2 features (*teta2* and *teta4*) were excluded from the data set. Of course it is not true, in general, that elimination of some features increases the class separability. Usually, adding features may help discriminate different data categories. It is left as an exercise for the reader to show that feature standardization also affects the value of Fisher coefficient F .

```
* b11 report file [raw data analysis]
* Data file name: "m25.txt"
* Selected features [3 out of 5]
teta1 [#1/#1]; p.mean= 1.35928E-001, p.std= 1.70216E-001
teta3 [#2/#3]; p.mean= 2.79588E-001, p.std= 1.63285E-001
sigma [#3/#5]; p.mean= 9.08444E-001, p.std= 7.51631E-002
Feature vector standardized: NO
* Results [raw data analysis]
> Fisher coefficient, F = 66.5
```

Fig. 4.1. Report of 'Raw data' analysis for *m25.sel* (*teta2* and *teta4* disabled).

4.4.2. Principal Component Analysis

Subtracting the pooled mean μ from the features in the training set is the first step in PCA analysis ([Krzanowski 1988](#)). The data covariance matrix is then estimated and its eigenvalues and eigenvectors computed using Jacobi algorithm ([Press et al 1988](#)). The eigenvalues $[\lambda_1, \lambda_2, \dots, \lambda_{Nx}]$ are listed in the decreasing order. Their corresponding eigenvectors form columns of a projection matrix Φ whose dimension is $Nx \times Np$. The Np principal components of the data set can be obtained by a linear transform

$$\mathbf{p}_i = \Phi^T (\mathbf{x}_i - \mu)$$

where $\mathbf{p}_i = [p_{i1}, p_{i2}, \dots, p_{iNp}]$, $i=1,2,\dots,M$. One can show that the covariance matrix of the transformed data is a diagonal matrix, $\text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{Nx})$, which means that the principal components are uncorrelated. Typically, only a small number Np (smaller than Nx) of principal components is needed to retain all the variance of the original data. These correspond to the largest eigenvalues of the covariance matrix. Thus PCA provides a compact, lower dimensionality representation of the original feature vectors. To evaluate the usefulness of PCA to data representation, the *linear dimensionality* coefficient d_j ([Mao and Jain, 1995](#)) is calculated, which is equal to the number of significant eigenvalues (more than 97% of the total variance is retained in first d_j principal components). Due to the principal components ability to optimally represent sets of data, they are called 'most expressive features', MEF ([Swets and Weng, 199f](#)).

PCA analysis example

A report file generated for **m25.sel** input after choosing **Analysis|PCA** menu items is presented in Fig. 4.4.2. The linear dimensionality of the **m25.sel** data is 3. One can notice that for the considered data, Fisher coefficient $F=48.7$ in the principal component space is slightly larger compared to its value in the original data space ($F=40.0$). However, in general, principal components may not be as effective for data discrimination as they are for data representation.


```

* b11 report file [PCA analysis]
* Data file name: "m25.txt"
* Selected features [5 out of 5]
teta1 [#1/#1]; p.mean= 1.35928E-001, p.std= 1.70216E-001
teta2 [#2/#2]; p.mean=-4.31560E-002, p.std= 6.36152E-002
teta3 [#3/#3]; p.mean= 2.79588E-001, p.std= 1.63285E-001
teta4 [#4/#4]; p.mean= 2.06920E-002, p.std= 3.81224E-002
sigma [#5/#5]; p.mean= 9.08444E-001, p.std= 7.51631E-002
Feature vector standardized: NO
* Results [PCA analysis]
Eigenvalues of data covariance matrix:
    5.86870E-002
    5.58017E-003
    1.43091E-003
    8.05602E-004
    2.81619E-004
Projection matrix for MEF:
    6.74660E-001  -6.22029E-001  1.14657E-001  2.84947E-001  2.52138E-001
   -2.18473E-001  1.79955E-001  7.19914E-001  6.33553E-001  -1.48323E-002
    6.32314E-001  7.56052E-001  4.64289E-003  1.97474E-003  1.68933E-001
    7.53454E-002  -8.56647E-002  6.79176E-001  -7.19307E-001  9.11136E-002
   -3.02664E-001  4.17459E-002  -8.53041E-002  2.90635E-003  9.48349E-001
Linear dimensionality: 3
> Fisher coefficient, F = 48.7

```

Fig. 4.4.2. Report of 'PCA' analysis for m25.sel (all features enabled).

4.4.3. Linear Discriminant Analysis

Let $\mathbf{x}^{(k)}$, denote the i th pattern in class i , $i=1,2,\dots,M_k$, $k=1,2,\dots,Nc$. Define the within-class scatter matrix \mathbf{C}_W as

$$\mathbf{C}_W = \frac{1}{M} \sum_{k=1}^{Nc} \sum_{i=1}^{M_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})^T$$

where $\boldsymbol{\mu}^{(k)}$ is the mean vector of class k . Similarly, define the between-class scatter matrix \mathbf{C}_B as

$$\mathbf{C}_B = \frac{1}{M} \sum_{k=1}^{Nc} M_k (\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T$$

where $\boldsymbol{\mu}$ is the mean vector of the pooled data. The total scatter matrix is, then

$$\mathbf{C}_T = \frac{1}{M} \sum_{k=1}^{Nc} \sum_{i=1}^{M_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu})(\mathbf{x}_i^{(k)} - \boldsymbol{\mu})^T$$

The goal of linear discriminant analysis is to find a linear transform matrix Φ such that the ratio of determinants

$$\frac{|\Phi^T \mathbf{C}_T \Phi|}{|\Phi^T \mathbf{C}_W \Phi|}$$

is maximized (Fukunaga 1991, Mao and Jain 1995). It can be proved that such a transform Ψ is composed of eigenvectors corresponding to largest eigenvalues of $\mathbf{C}_W^{-1} \mathbf{C}_T$. Transformation of original data by means of matrix Ψ

$$\mathbf{q}_i = \Psi^T (\mathbf{x}_i - \boldsymbol{\mu})$$

produces most discriminating features, MDF, (Swets and Weng 1996), such that $\mathbf{MDF}_i = [MDF_{i1}, MDF_{i2}, \dots, mdf_{iNq}] = \mathbf{q}_i = [q_{i1}, q_{i2}, \dots, q_{iNq}]$, $i=1,2,\dots,M$, where Nq is usually smaller than Nx .

To evaluate the usefulness of LDA to classes discrimination, the *linear separability* coefficient I_s (Mao and Jain, 1995) is

calculated, which is defined as the largest eigenvalue of $C_T^{-1}C_B$ (Gallinari et al 1991). As I_s changes from 0.0 to 1.0, the data set becomes more and more linearly separable. Similarly to PCA analysis, an LDA dimensionality factor is calculated, which is equal to the number of largest eigenvalues of $C_T^{-1}C_B$ whose sum is greater than 0.97.

LDA example

A report file generated for **m25.sel** input after choosing **Analysis|LDA** menu items is presented in Fig. 4.2. The linear separability of the **m25.sel** data is 0.98, which means that this set is well separable by hyperplanes in the MDF space. One can notice that for the considered data, Fisher coefficient in the MDF space ($F=64.3$) is larger compared to its values in the original ($F=40.0$) and MEF ($F=48.7$) data spaces.

```
* b11 report file [LDA analysis]
* Data file name: "m25.txt"
* Selected features [5 out of 5]
teta1 [#1/#1]; p.mean= 1.35928E-001, p.std= 1.70216E-001
teta2 [#2/#2]; p.mean=-4.31560E-002, p.std= 6.36152E-002
teta3 [#3/#3]; p.mean= 2.79588E-001, p.std= 1.63285E-001
teta4 [#4/#4]; p.mean= 2.06920E-002, p.std= 3.81224E-002
sigma [#5/#5]; p.mean= 9.08444E-001, p.std= 7.51631E-002
Feature vector standardized: NO
* Results [LDA analysis]
Eigenvalues of [inv(St)*Sb]:
    9.79820E-001
    8.42142E-001
    7.71618E-001
    3.45186E-001
   -4.65573E-017
Projection matrix for MDF:
    7.84288E-001  -5.80625E-001  1.85203E-001  3.47844E-001  7.27552E-002
   -2.34161E-001  7.55683E-002  -1.44937E-001  8.35113E-001  1.22662E-002
    5.67639E-001  3.01854E-001  2.15120E-001  8.71321E-002  1.50459E-001
   -6.98187E-002  -3.74772E-002  -1.55091E-001  -3.10235E-001  7.43399E-001
    5.45158E-002  -7.51427E-001  9.35074E-001  2.78843E-001  6.47516E-001
Linear separability: 0.98
LDA dimensionality: 4
> Fisher coefficient, F = 64.3
```

Fig. 4.4.3. Report of LDA analysis for m25.sel (all features enabled).

4.4.4. Nonlinear Discriminant Analysis

Often, one deals with data sets where clusters of points belonging to different categories can not be separated using linear classifiers (i.e. by means of hyperplanes in the feature space). Both PCA and LDA described above use linear transformation of the input patterns. These two techniques provide means for feature compression (both MEF and MDF type vectors have less elements than the original feature vectors). However, since they are based on linear transformations, they can not help in classification of linearly non-separable data. Such data need hypersurfaces instead of hyperplanes for clusters separation. Nonlinear classifiers, such as artificial neural networks (Hecht-Nielsen 1989, Freeman and Skapura 1991) or k -NN classifiers (Duda and Hart 1973, Fukunaga 1991) may be appropriate to perform the classification task.

Another solution, which is employed in **b11** 'NDA analysis' is based on finding a nonlinear transformation of the data and then applying a linear classifier to the resulting features. In other words, the aim of nonlinear discriminant analysis (NDA) is to find a nonlinear transformation of the feature vectors, such that the input patterns are projected on a space (possibly of lower dimensionality as in the case of PCA and LDA) in which they become linearly separated (Gallinari et al 1991, Mao and Jain 1995). This technique is implemented here by using a feedforward artificial neural network (ANN) with two hidden layers of sigmoid-type neurons (Fig. 4.4.4).

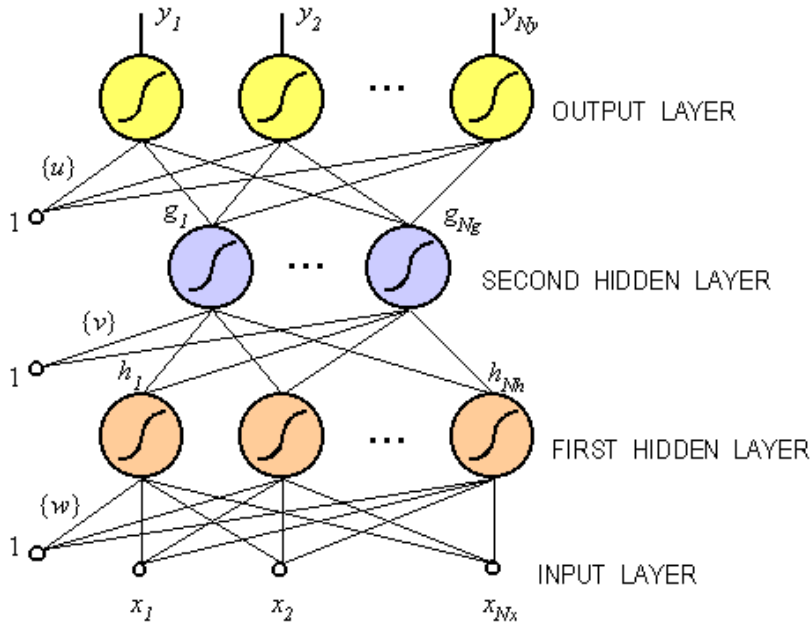


Fig. 4.4.4 Artificial neural network architecture for nonlinear discriminant analysis.

Feature vector \mathbf{x} is the input to ANN. The number of input terminals is equal to Nx , i.e. to input pattern dimension. The output of ANN is a vector \mathbf{y} , whose dimension Ny is equal to the number of categories in the data set. Thus the ANN has Ny output terminals. There are 3 layers of processing elements (neurons) in the network of Fig. 4.4. Each of them implements a simple single-input single-output nonlinear function $\xi=f(\alpha)$, described by

$$\xi(\alpha) = \frac{1}{1 + \exp(-\alpha)}$$

which is sometimes called a logistic function. Its plot has a 'sigmoidal' shape. It is equal to 0.5 for $\alpha=0$, and smoothly changes from 0 to 1 for α varying from large negative values to large positive values.

The neurons whose outputs are the elements of the \mathbf{y} vector, form the output layer of the ANN. Their inputs are linked to the outputs g_k , $k=1,2,\dots,Ng$ of the so-called second hidden layer. This layer contains Ng neurons. The inputs of the neurons in the second hidden layer are driven by output signals h_j , $j=1,2,\dots,Nh$ of Nh neurons in the first hidden layer. Finally, the first hidden layer is excited by the feature vector \mathbf{x} . Each layer of neurons has an extra input, of unit value, so-called bias.

Each connection between layers is allocated a weight coefficient. There are 3 sets of weights in the network of Fig. 4.4 - $\{w\}$, $\{v\}$, and $\{u\}$. Thus input α to any neuron is a weighted sum of the appropriate bias and output signals from all the neurons located in the preceding layer. It follows that the elements of the output vector \mathbf{y} are described by

$$y_n = \xi \left(u_{n0} + \sum_{k=1}^{Ng} u_{nk} g_k \right), \quad n = 1, 2, \dots, Ny$$

the outputs of the second hidden layer are given as

$$g_k = \xi \left(v_{k0} + \sum_{j=1}^{Nh} v_{kj} h_j \right), \quad k = 1, 2, \dots, Ng$$

and the outputs of the first hidden layer can be calculated from the formula

$$h_j = \xi \left(w_{j0} + \sum_{i=1}^{Nx} w_{ji} x_i \right), \quad j = 1, 2, \dots, Nh$$

It can be proven (Hecht-Nielsen 1989) that the network of the discussed architecture is able to approximate any smooth nonlinear mapping from the input space \mathbf{x} to the second hidden layer space \mathbf{g} . The output layer performs the function of a linear classifier, provided a decision-making block is added. (Vector \mathbf{g} belongs to class c_k if the output y_k is close to 1.0

and all the remaining outputs are close to 0.0. Decision is inconclusive if more than one output is close to 1.0. Vector \mathbf{g} that causes such situation is allocated to a category c_0 that is not present in the input data set.)

The nonlinear transformation considered is performed by the two hidden layers of the network in Fig. 4.4. The input patterns \mathbf{x} of dimension $N \times x$ are transformed into \mathbf{g} vectors whose dimension is $Ng < Nx$. The limited number of \mathbf{g} vector elements forms a "bottleneck" through which the information passes flowing from input \mathbf{x} to output \mathbf{y} .

The neural network has to be trained to be able to perform its function. The training involves adjustment of the ANN weight coefficients such that the actual output of the network \mathbf{y} is close to a desired output \mathbf{d} . Supervised training techniques are used in **b11**, based on examples of input patterns and correct categories they belong to, $\{\mathbf{x}_i, d_i\}$, $i=1,2,\dots,M$. For this purpose, the following error function

$$E = \frac{1}{2} \sum_{i=1}^M \sum_{n=1}^{Np} (d_{in} - y_{in}(\mathbf{x}_i; \mathbf{u}, \mathbf{v}, \mathbf{w}))^2$$

is minimized by adjustment of weights \mathbf{u} , \mathbf{v} , and \mathbf{w} . Training the neural network relies in fact on solving a multivariable numerical minimization problem. Two training methods are implemented in **b11**, which are realized sequentially. The first is the well-known backpropagation technique ([Hecht-Nielsen 1989](#), [Freeman and Skapura 1991](#)). It is done iteratively, in a limited number of steps, to provide coarse values of weights. A multivariable optimization routine ([Press et al. 1996](#)) is then invoked as the second technique, to produce lower values of error and more accurate weights.

One should note, however, that the minimization problem at hand features multiple minima of the error function. Therefore, there is no guarantee that a right solution will be obtained after training, for a given training set. A good practice is to repeat the whole process of training few times, each time starting with randomly initialized weight coefficients, and selecting the solution which provides the best results.

Another word of warning relates to the size of the neural network, measured in the number of neurons in hidden layers. It is well known that as the ANN size increases, it becomes more accurate, in that the mapping $\mathbf{y}(\mathbf{x})$ becomes closer to the desired values \mathbf{d} , over the set of training examples. At the same time, with the increased number of neurons the number of weights increases exponentially. Thus increases the number of unknown degrees of freedom, which are adjusted in the process of training. More training examples are needed to provide at least as many input/output data as there are unknowns (weights). If this condition is not satisfied, the ANN might memorize the training examples "by heart", giving a very low value of error, which could be misleading. Such a network would not possess the required ability to generalize. Namely, when presented unseen input patterns, located "in between" the training patterns, the ANN would produce large errors. This phenomenon is called "overtraining". A good practice is to (1) use as small a network as possible, (2) split the available data set into the training set and *validation set* (or *training test set*) and measure the network performance (in terms of error value) for both sets. If, for given ANN size, the error over the training set decreases and the error measured over the test set starts increasing, the training should be terminated to avoid overtraining ([Hecht-Nielsen 1989](#)).

Since the result of ANN training depends on adjustable parameters whose correct (or best) values may be data-dependent, selection of some parameters is left to the user of **b11**. These parameters are listed in the 'Neural network parameters' box in options window.

Practice shows that training time is shorter if ANN inputs are standardized. Therefore, the training example patterns are standardized for NDA analysis, irrespective of the setting in options window.

NDA example 1

A report file generated for **m25.sel** input after choosing **Analysis|NDA** menu items is presented in Figs. 4.4.5 and 4.4.6. After the standard heading (see the analysis reports in Figs. 4.4.1, 4.4.2, and 4.4.3), information about the ANN size is provided in Fig. 4.4.5 (the number of neurons in the second hidden layer $Ng=3$ was selected in this example). Then, chosen values of backpropagation parameters are displayed and errors obtained in consecutive iterations (in steps of 50000 iterations) are listed. After backpropagation, the numerical weight optimization is performed. One can notice that for the considered data, Fisher coefficient $F=235.9$ in the nonlinear components data space is much much bigger compared to its value in the original data space ($F=40.0$). This shows the potential of the NDA technique to separate category clusters (Fig. 4.4.7). The report generated after NDA analysis includes also values of weight coefficients (Fig 4.4.6), as obtained after training. These values can be easily transferred to e.g. Neural Network Toolbox[®] of Matlab[®], for further utilization. Figure 4.4.7 presents feature clusters, obtained for **m25.sel** data file in the 3-dimensional NDA space ($Ng=3$).

```

* b11 report file [nonlinear discriminant analysis]
* Data file name: "m25.txt"
* Selected features [5 out of 5]
teta1 [#1/#1]; p.mean= 1.35928E-001, p.std= 1.70216E-001
teta2 [#2/#2]; p.mean=-4.31560E-002, p.std= 6.36152E-002
teta3 [#3/#3]; p.mean= 2.79588E-001, p.std= 1.63285E-001
teta4 [#4/#4]; p.mean= 2.06920E-002, p.std= 3.81224E-002
sigma [#5/#5]; p.mean= 9.08444E-001, p.std= 7.51631E-002
Feature vector standardized: YES
* Results [nonlinear discriminant analysis]
> Neural network architecture
    input layer: 5 nodes
    1st hidden layer: 3 neurons
    2nd hidden layer: 3 neurons
    output layer: 5 nodes
> backprop (eta=0.15, bpIterLimit=400000)
iter  rms |-----sample-----|
/1e3 error | # error dy1 dy2 dy3 dy4 dy5 |
  0 0.172 25 0.336-0.3-0.3-0.3 0.3
 50 0.002  8 0.045-0.1 0.1 0.0 0.0 0.0
100 0.006  3 0.076 0.1-0.1 0.0 0.0 0.0
150 0.005  9 0.072-0.1 0.1 0.0 0.0 0.0
200 0.006  3 0.075 0.1-0.1 0.0 0.0 0.0
250 0.001 25 0.032 0.0 0.0 0.0 0.0 0.0
300 0.000 15 0.016 0.0 0.0 0.0 0.0 0.0
350 0.001  1 0.023 0.0 0.0 0.0 0.0 0.0
400 0.000  9 0.015 0.0 0.0 0.0 0.0 0.0
> ANN weight numerical optimization
  (optyIterLimit = 50; WeightCount = 50)
  IterCount  rms error
    0  1.49E-002
    6  6.42E-004
> Missclassified f. vectors: 0/25 [or 0.00%]
> Fisher coefficient, F = 235.9
> Neural network weights

```

Fig. 4.4.5. Report of 'NDA' analysis for m25.sel (all features enabled, Ng=3).

```

> FISHER COEFFICIENT, F = 235.9
> Neural network weights
1st hidden layer -----
  9.94154E-001 -1.14655E+000  8.24999E-001 -4.46543E-001 -3.11560E+000  3.79071E+000
  1.15207E+000 -6.38648E+000 -4.02177E+000  5.90539E+000 -1.01497E-001  2.64391E+000
 -4.68502E+000 -4.61987E+000  3.53868E+000 -1.55306E+000  7.71843E-001 -1.12465E+000
2nd hidden layer -----
 -8.80956E+000  4.47441E+000  9.35154E-001  9.97628E+000
 -5.69806E+000 -4.78787E+000  9.12691E+000  6.44674E+000
 -4.74801E+000  6.93794E+000  5.24536E+000  2.27260E+000
output layer -----
  7.52251E+000 -1.94837E+002 -1.70845E+001 -8.17806E+000
 -8.35346E+000 -2.98283E+002  1.95975E+001 -9.38165E-001
 -9.19801E+000 -1.81903E+001 -9.11713E+000  1.90082E+001
 -1.56654E+001  1.53085E+001  1.37403E+001 -6.55055E+000
 -1.02900E+001  1.79522E+001 -3.42967E+001  3.54418E+000

```

Fig. 4.4.6. Neural network weights extracted from the report file of 'NDA' analysis (Ng=3).

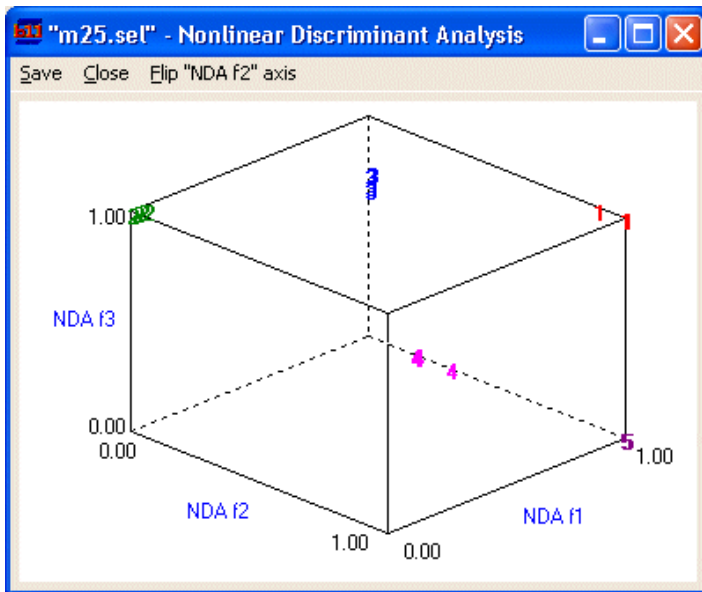


Fig. 4.4.7. Clusters in NDA feature space for m25.sel data file ($N_g=3$).

The **Options** menu item allows selection of the number N_g of neurons in the second (bottleneck) hidden layer of the neural network. The above example showed results that could be obtained for $N_g=3$. The following example was designed to illustrate the other possibility, $N_g=2$, applied to the same data file.

NDA example 2

A report file generated for **m25.sel** input and $N_g=2$, after choosing **Analysis|NDA** menu items, is presented in Figs. 4.8 and 4.9. The differences between this report and the one shown in Figs. 4.5 and 4.6 relate to the reduced number of neurons in the second hidden layer ($N_g=2$) and, incidentally, different value of iterations in the backpropagation algorithm (bpIterLimit=200,000 instead of 500,000). The Fisher coefficient $F=2038.1$ in the nonlinear components data space is much much bigger compared to its value in the original data space ($F=40.0$). Figure 4.10 presents feature clusters, obtained for **m25.sel** data file in the 2-dimensional NDA space ($N_g=2$). The clusters belonging to different categories can be linearly separated in the NDA space in this case. The straight-line segments drawn in color in Fig. 4.10 represent the boundaries between a selected category and all the remaining categories. For example, the green line separates the points of category #2 from all the other categories (#1, #3, #4, and #5). Areas on the NDA (f_1, f_2) plane that form a common part of sets belonging to two or more different categories (e.g. bottom-right corner on the NDA (f_1, f_2) plane, ie. points where f_1 is close to 1 and f_2 is close to 0, which apparently would belong to both category #3 and category #4) represent "inconclusive" cases. Non-existing category #0 is allocated to such points if they occur in practice.

```

* b11 report file [nonlinear discriminant analysis]
* Data file name: "m25.txt"
* Selected features [5 out of 5]
teta1 [#1/#1]; p.mean= 1.35928E-001, p.std= 1.70216E-001
teta2 [#2/#2]; p.mean=-4.31560E-002, p.std= 6.36152E-002
teta3 [#3/#3]; p.mean= 2.79588E-001, p.std= 1.63285E-001
teta4 [#4/#4]; p.mean= 2.06920E-002, p.std= 3.81224E-002
sigma [#5/#5]; p.mean= 9.08444E-001, p.std= 7.51631E-002
Feature vector standardized: YES
* Results [nonlinear discriminant analysis]
> Neural network architecture
    input layer: 5 nodes
    1st hidden layer: 3 neurons
    2nd hidden layer: 2 neurons
    output layer: 5 nodes
> backprop (eta=0.15, bpIterLimit=200000)
iter  rms  |-sample-----|
/1e3 error | # error dy1 dy2 dy3 dy4 dy5 |
  0  0.183 | 7 0.357-0.4 0.4-0.4-0.4-0.4 |
  50 0.011 | 19 0.103 0.0 0.0-0.1 0.2-0.2 |
 100 0.002 | 6 0.043-0.1 0.1 0.0 0.0 0.0 |
 150 0.020 | 8 0.119-0.1 0.2 0.0 0.0 0.0 |
 200 0.002 | 12 0.042 0.0-0.1 0.0 0.0 0.0 |
> ANN weight numerical optimization
  (optyIterLimit = 50; WeightCount = 41)
  IterCount  rms error
    0  3.02E-001
   10  1.60E-001
   20  1.08E-001
   30  2.61E-002
   40  1.30E-002
   50  5.44E-003
   50  5.44E-003
> Missclassified f. vectors: 0/25 [or 0.00%]
> Fisher coefficient, F = 2038.1
> Neural network weights

```

Fig. 4.4.8. Report of 'NDA' analysis for m25.sel (all features enabled, Ng=2).

```

> FISHER COEFFICIENT, F = 2038.1
> Neural network weights
1st hidden layer -----
  3.02791E+000  3.56422E+000 -3.78060E-001  5.19750E+000 -9.01896E-002  3.44745E+000
 -1.14813E+001  1.05098E+001 -3.96498E+000  1.07341E+000 -3.65604E+000 -4.37462E+000
 -5.45642E+000  9.16052E+000  6.79842E+000  3.98358E-001  3.92775E+000  5.03088E-001
2nd hidden layer -----
 -2.99911E+000  1.99489E+001  1.48480E+001 -4.35077E+001
 -1.34393E+001  1.32056E+001  1.57354E+001  2.30143E+000
output layer -----
 -4.50932E+000 -1.44519E+001  1.14128E+001
 -2.20668E+001  1.10044E+001  1.57099E+001
 -2.71146E+001  3.77275E+001 -1.55135E+001
 -6.14519E+000  1.80304E+001 -4.64710E+001
  1.68205E+001 -4.96123E+001 -3.92815E+001

```

Fig. 4.4.9. Neural network weights extracted from the report file of 'NDA' analysis (Ng=2).

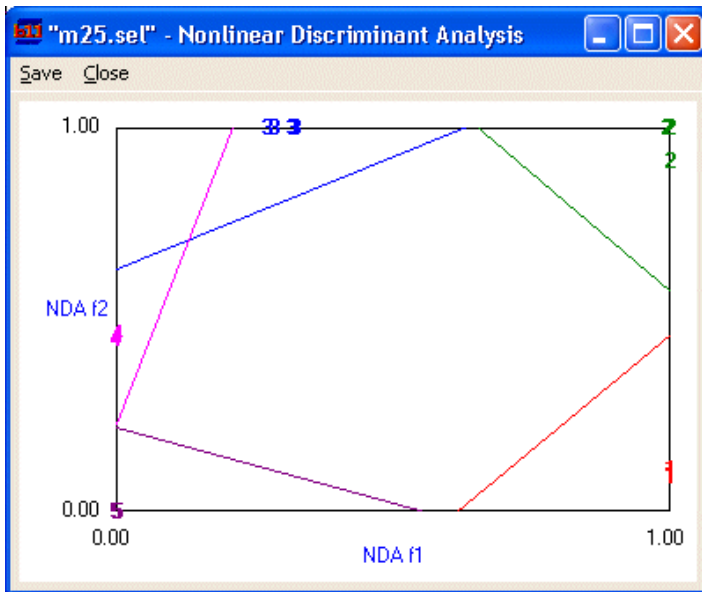


Fig. 4.4.10. Clusters in NDA feature space for m25.sel data file ($N_g=2$).

B11 - program for data analysis and classification

4.5. Classification

At present, two neural-network classifiers and the k -NN nearest-neighbor classifier (Duda and Hart 1973, Fukunaga 1991) with $k=1$ are implemented in **b11**.

4.5.1. Nearest Neighbour Classifier

Let the set $S^M = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ be a set of labelled patterns, and let $\mathbf{x}^n \in S^M$ be the pattern closest to \mathbf{x} . Then the *nearest neighbor rule* for classifying \mathbf{x} is to assign it the label (category) associated with \mathbf{x}^n . The nearest neighbour rule is a suboptimal procedure. Its use will usually lead to an error rate greater than the minimum possible, the Bayes rate. However, with a very large number of samples (patterns), the error rate is never worse than twice the Bayes rate. On the other hand, unlike the Bayesian classifier, the nearest neighbour rule does not require estimation of conditional probability density function for each class, so it is easier to implement.

k-NN classification example

Load the supplied **f48_10.sel** input file, which contains $M=96$ training patterns, which are the co-occurrence matrix-derived features computed with **MaZda** for 48 ROIs in two images - 'Foam1' and 'Foam2' (Materka et al. 1999). Since each image contains a uniform texture, different in view to the texture in the other image, there are 2 categories defined in this example. Run **Raw Data** analysis, with unchecked feature standardization option. The scatter plot of 3 first features you will obtain is illustrated in Fig. 5.1.

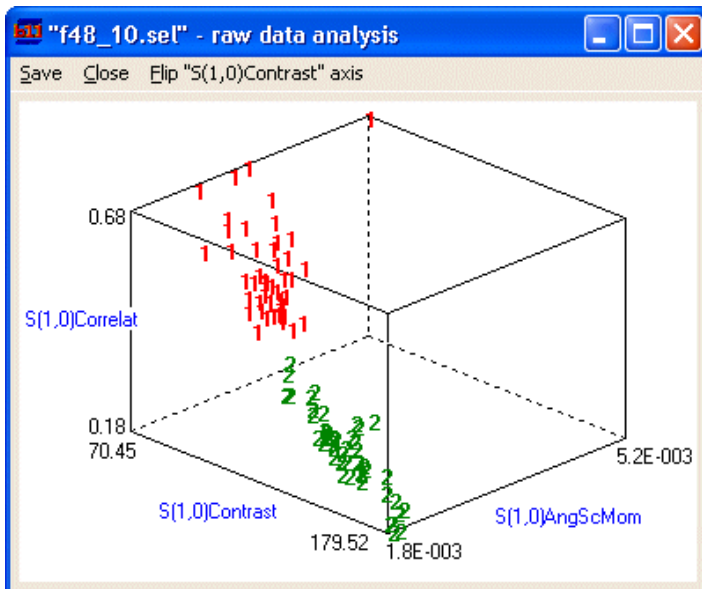


Fig. 4.5.1. Scatter plot of patterns made of first 3 features included in f48_10.sel file.

Close the graphics window and activate **k_NN** menu item. The report that will be displayed in the output panel of the main windows is listed in Fig. 5.2. It indicates that one input pattern is misclassified by the nearest neighbour rule in this case. It is sample #61 of category #2, which has been erroneously classified as belonging to category #1. Now change option settings to enforce data standardization. The **k_NN** report produced after executing **k_NN** classification is shown in Fig. 5.3. No error is made if data are standardized, in the case of **f48_10.sel** file.

```
* b11 report file [k-NN classification]
* Data file name: "f49_10.txt"
* Selected features [10 out of 10]
S(1,0)AngScMom 1.0 [#1/#1]; p.mean= 2.43618E-003, p.std= 5.07853E-004
S(1,0)Contrast 4.5 [#2/#2]; p.mean= 1.26616E+002, p.std= 3.02849E+001
S(1,0)Correlat 4.3 [#3/#3]; p.mean= 4.30601E-001, p.std= 1.36192E-001
S(1,0)InvDfMom 4.3 [#4/#4]; p.mean= 1.57695E-001, p.std= 4.79908E-002
S(1,0)SumAverg 0.3 [#5/#5]; p.mean= 6.43542E+001, p.std= 3.21106E-001
S(1,0)SumVarnrc 3.6 [#6/#6]; p.mean= 3.18161E+002, p.std= 3.05978E+001
S(1,0)SumEntrp 0.8 [#7/#7]; p.mean= 1.80118E+000, p.std= 1.67717E-002
S(1,0)Entropy 1.8 [#8/#8]; p.mean= 2.71159E+000, p.std= 5.25169E-002
S(1,0)DifVarnrc 1.6 [#9/#9]; p.mean= 5.24857E+001, p.std= 7.50406E+000
S(1,0)DifEntrp 4.0 [#10/#10]; p.mean= 1.34353E+000, p.std= 6.43622E-002
Feature vector standardized: NO
* Results [k-NN classification]
Misclassified data vectors: 1/96 [or 1.04%]
Sample No: 61; Category: 2; ClassResult: 1
```

Fig. 4.5.2. Report file generated for f48_10.sel input (data standardization: OFF).

```
* b11 report file [k-NN classification]
* Data file name: "f49_10.txt"
* Selected features [10 out of 10]
S(1,0)AngScMom 1.0 [#1/#1]; p.mean= 2.43618E-003, p.std= 5.07853E-004
S(1,0)Contrast 4.5 [#2/#2]; p.mean= 1.26616E+002, p.std= 3.02849E+001
S(1,0)Correlat 4.3 [#3/#3]; p.mean= 4.30601E-001, p.std= 1.36192E-001
S(1,0)InvDfMom 4.3 [#4/#4]; p.mean= 1.57695E-001, p.std= 4.79908E-002
S(1,0)SumAverg 0.3 [#5/#5]; p.mean= 6.43542E+001, p.std= 3.21106E-001
S(1,0)SumVarnrc 3.6 [#6/#6]; p.mean= 3.18161E+002, p.std= 3.05978E+001
S(1,0)SumEntrp 0.8 [#7/#7]; p.mean= 1.80118E+000, p.std= 1.67717E-002
S(1,0)Entropy 1.8 [#8/#8]; p.mean= 2.71159E+000, p.std= 5.25169E-002
S(1,0)DifVarnrc 1.6 [#9/#9]; p.mean= 5.24857E+001, p.std= 7.50406E+000
S(1,0)DifEntrp 4.0 [#10/#10]; p.mean= 1.34353E+000, p.std= 6.43622E-002
Feature vector standardized: YES
* Results [k-NN classification]
Misclassified data vectors: 0/96 [or 0.00%]
```

Fig. 4.5.3. Report file generated for f48_10.sel input (data standardization: ON).

4.5.2. Neural-network 'n-class' classifier

The 'n-class' classifier employs a feedforward artificial neural network with a single hidden layer of sigmoidal neurons, as shown in Fig. 5.4. Feature vector \mathbf{x} is the input to the ANN. The number of input terminals is equal to Nx , i.e. to input pattern dimension. The output of ANN is a vector \mathbf{y} , whose dimension Ny is equal to the number of categories in the data set. Thus the ANN has Ny output terminals.

The neurons whose outputs are the elements of the \mathbf{y} vector, form the output layer of the ANN. Their inputs are driven by output signals $h_j, j=1,2,\dots,Nh$ of Nh neurons in the hidden layer. Finally, the hidden layer is excited by the feature vector \mathbf{x} . Each layer of neurons has an extra input, of unit value, so-called bias.

Each connection between layers is allocated a weight coefficient. There are 2 sets of weights in the network of Fig. 5.4 - $\{w\}$ and $\{v\}$. Thus input to any neuron is a weighted sum of the appropriate bias and output signals from all the neurons located in the preceding layer. It follows that the elements of the output vector \mathbf{y} are described by

$$y_k = \xi^k \left(v_{k0} + \sum_{j=1}^{Nh} v_{kj} h_j \right), \quad k = 1, 2, \dots, Ny$$

the outputs of the hidden layer are given as

$$h_j = \xi^j \left(w_{j0} + \sum_{i=1}^{Nx} w_{ji} x_i \right), \quad j = 1, 2, \dots, Nh$$

It can be proven ([Hecht-Nielsen 1989](#)) that the network of the discussed architecture is able to approximate any smooth nonlinear mapping from the input space \mathbf{x} to the output space \mathbf{y} . The output layer performs the function of a classifier, provided a decision-making block is added. (Vector \mathbf{y} belongs to class c_k if the output y_k is close to 1.0 and all the remaining outputs are close to 0.0. Decision is inconclusive if more than one output is close to 1.0. Vector \mathbf{g} that causes such situation is allocated to a category c_0 that is not present in the input data set.)

The neural network has to be trained to be able to perform its function. The training involves adjustment of the ANN weight coefficients such that the actual output of the network \mathbf{y} is close to a desired output \mathbf{d} . Supervised training techniques are used in **b11**, based on examples of input patterns and correct categories they belong to, $\{\mathbf{x}_i, d_i\}, i=1,2,\dots,M$. For this purpose, the following error function

$$E = \frac{1}{2} \sum_{i=1}^M \sum_{k=1}^{Ny} (d_{ik} - y_{ik}(\mathbf{x}_i; \mathbf{v}, \mathbf{w}))^2$$

is minimized by adjustment of weights \mathbf{v} and \mathbf{w} . Training the neural network relies in fact on solving a multivariable numerical minimization problem. Two training methods are implemented in **b11**, which are realized sequentially. The first is the well-known backpropagation technique ([Hecht-Nielsen 1989](#), [Freeman and Skapura 1991](#)). It is done iteratively, in a limited number of steps, to provide coarse values of weights. A multivariable optimization routine ([Press et al. 1996](#)) is then invoked as the second technique, to produce lower values of error and more accurate weights.

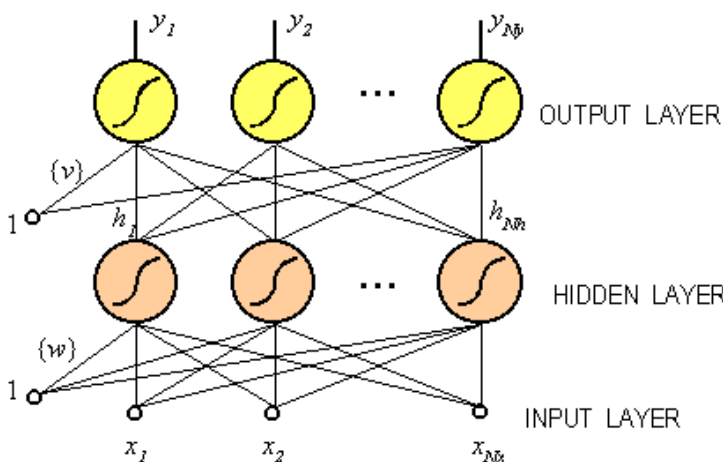


Fig. 4.5.4. Artificial neural network architecture for ' N_y -class' data classification.

One should note, however, that the minimization problem at hand features multiple minima of the error function. Therefore, there is no guarantee that a right solution will be obtained after training, for a given training set. A good practice is to repeat the whole process of training few times, each time starting with randomly initialized weight coefficients (which b11 does, actually), and selecting the solution which provides the best results.

Another word of warning relates to the size of the neural network, measured in the number of neurons in hidden layers. It is well known that as the ANN size increases, it becomes more accurate, in that the mapping $\mathbf{y}(\mathbf{x})$ becomes closer to the desired values \mathbf{d} , over the set of training examples. At the same time, with the increased number of neurons the number of weights increases exponentially. Thus increases the number of unknown degrees of freedom, which are adjusted in the process of training. More training examples are needed to provide at least as many input/output data as there are unknowns (weights). If this condition is not satisfied, the ANN might memorize the training examples "by heart", giving a very low value of error, which could be misleading. Such a network would not possess the required ability to generalize. Namely, when presented unseen input patterns, located "in between" the training patterns, the ANN would produce large errors. This phenomenon is called "overtraining". A good practice is to (1) use as small a network as possible, (2) split the available data set into the training set and *validation set* (or *training test set*) and measure the network performance (in terms of error value) for both sets. If, for given ANN size, the error over the training set decreases and the error measured over the test set starts increasing, the training should be terminated to avoid overtraining ([Hecht-Nielsen 1989](#)).

Since the result of ANN training depends on adjustable parameters whose correct (or best) values may be data-dependent, selection of some parameters is left to the user of **b11**. These parameters are listed in the 'Neural network parameters' box in options window.

Practice shows that training time is shorter if ANN inputs are standardized. Therefore, the training example patterns are standardized for NDA analysis, irrespective of the setting in options window.

Example of 'n-class' classification

Load the supplied **iris.sel** input file, which contains the famous Fisher $M=150$ training patterns, which are the sepal and petal lengths and widths of iris flowers. The dimension of feature vectors is then 4 in this case. There are 3 categories (classes) - (1) Iris Setosa, (2) Iris Versicolor, and (3) Iris Virginica. Run **Raw Data** analysis, with unchecked feature standardization option. The scatter plot of 3 first features you will obtain is illustrated in Fig. 5.1. Patterns belonging to class '1' are easily separable from other classes; however, separation of vectors belonging to classes 2 and 3 does not seem easy.

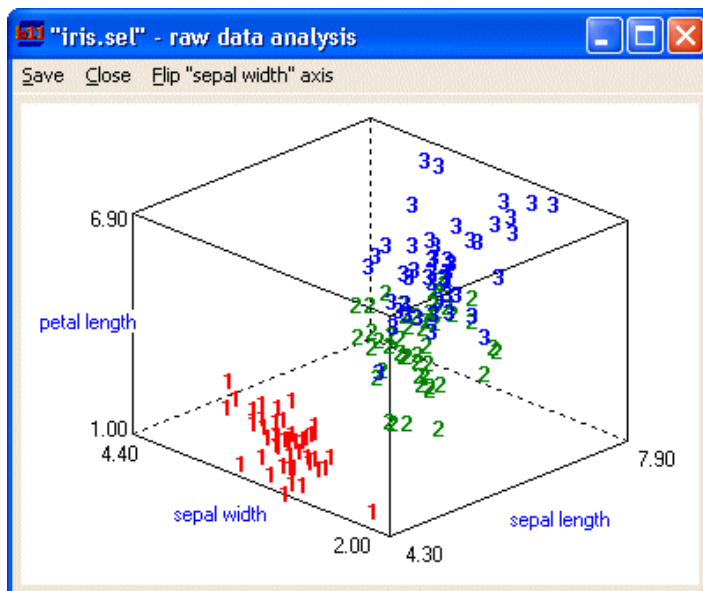


Fig. 4.5.5. Scatter plot of patterns made of first 3 features included in iris.sel file.

Close the graphics window. Use the **Options** menu item to select the ANN parameters as shown in Fig. 4.5.6. Please notice, that the number of neurons in the second hidden layer has no effect on the experiment, since the ANN does not possess it (Fig. 4.5.4). Its only hidden layer is identified as '1st hidden layer' in the options window (Fig. 4.5.6). There are then 2 neurons in the hidden layer in this example, the number of inputs is 4, and the number of outputs is 3. Sometimes, such a network is said to have 4-2-3 architecture.

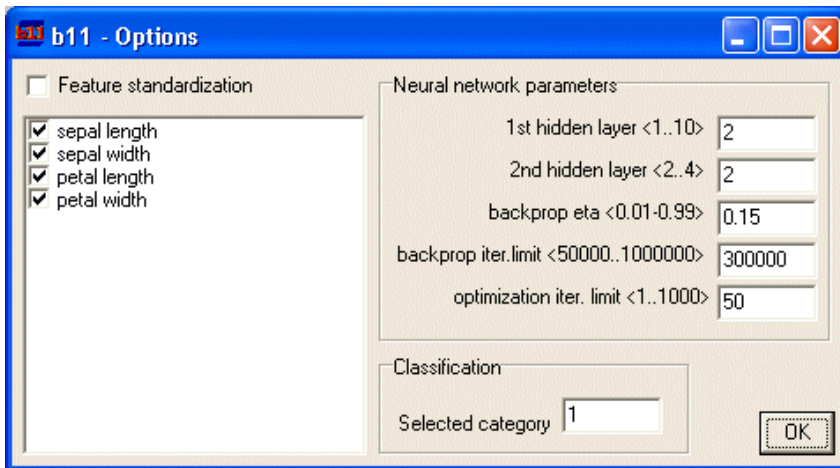


Fig. 4.5.6. Options selected for 'n-class' example with iris.sel file.

Activate **Classification|ANN: n-class (training)** menu item. The report that will be displayed in the output panel of the main windows is listed in Figs. 5.7 and 5.8. It indicates that one input pattern is misclassified by the ANN classifier in this case. It is sample #134 of category #3, which has been erroneously classified as belonging to category #2. Now use the k-NN classifier for the **iris.sel** data, for comparison. Fig 5.9 indicates, that there are 6 misclassification errors made by the k-nN classifier with no feature standardization, all misclassifications related to the confusion between categories 2 and 3. One can easily check that the number of classification errors increases to 8 if feature standardization options is switched on, in the k-NN classifier case. This example illustrates the capacity of artificial neural networks to effectively separate patterns that are otherwise difficult to classify by standard means.

```
* b11 report file [ANN classification of n categories]
* Data file name: "iris.sel"
* Selected features [4 out of 4]
sepal length [#1/#1]; p.mean= 5.84333E+000, p.std= 8.28066E-001
sepal width [#2/#2]; p.mean= 3.85400E+000, p.std= 4.33594E-001
petal length [#3/#3]; p.mean= 3.75867E+000, p.std= 1.76442E+000
petal width [#4/#4]; p.mean= 1.19867E+000, p.std= 7.63161E-001
Feature vector standardized: YES
* Results [ANN classification of n categories]
> Neural network architecture
  input layer:  4 nodes
  hidden layer: 2 neurons
  output layer: 3 nodes
> backprop (eta=0.15, bpIterLimit=300000)
iter  rms  |-sample-----|
/1e3 error | # error dy1 dy2 dy3 |
  0 0.083 119 0.114-0.1-0.1 0.1
  50 0.001  38 0.030 0.0 0.0 0.0
 100 0.000  60 0.017 0.0 0.0 0.0
 150 0.001  52 0.030 0.0 0.0 0.0
 200 0.000 142 0.004 0.0 0.0 0.0
 250 0.000  70 0.007 0.0 0.0 0.0
 300 0.000 117 0.005 0.0 0.0 0.0
> ANN weight numerical optimization
```

Fig. 4.5.7. Report file generated for iris.sel input and 4-2-3 'n-class' classifier.

```

> ANN weight numerical optimization
  (optyIterLimit = 50; WeightCount = 19)
  IterCount  rms error
      0  1.08E-001
     10  8.83E-002
     20  8.07E-002
     30  7.47E-002
     40  6.71E-002
     50  6.67E-002
> Missclassified f. vectors: 1/150 [or 0.67%]
Sample No: 134; Category: 3; ClassResult: 2
> Neural network weights
hidden layer -----
  2.08441E+001  3.80776E+001 -1.73132E+001 -2.55464E+000 -4.81591E+000
-8.99077E+001 -3.26345E+000 -1.12747E+001  5.85810E+001  6.97572E+001
output layer -----
  3.13377E+001 -1.29556E+002 -1.28899E+003
-2.18374E+001  7.20572E+001 -9.34570E+001
-1.05381E+001 -3.47852E+001  8.45404E+001

```

Fig. 4.5.8. Report file generated for iris.sel input and 4-2-3 'n-class' classifier (continued).

```

* b11 report file [k-NN classification]
* Data file name: "iris.sel"
* Selected features [4 out of 4]
sepal length [#1/#1]; p.mean= 5.84333E+000, p.std= 8.28066E-001
sepal width [#2/#2]; p.mean= 3.05400E+000, p.std= 4.33594E-001
petal length [#3/#3]; p.mean= 3.75867E+000, p.std= 1.76442E+000
petal width [#4/#4]; p.mean= 1.19867E+000, p.std= 7.63161E-001
Feature vector standardized: NO
* Results [k-NN classification]
Missclassified data vectors: 6/150 [or 4.00%]
Sample No: 71; Category: 2; ClassResult: 3
Sample No: 73; Category: 2; ClassResult: 3
Sample No: 84; Category: 2; ClassResult: 3
Sample No:107; Category: 3; ClassResult: 2
Sample No:120; Category: 3; ClassResult: 2
Sample No:134; Category: 3; ClassResult: 2

```

Fig. 4.5.9. Report file generated for iris.sel input and k-NN classifier.

4.5.3. Neural-network 'one-class' classifier

The 'one-class' ANN classifier, whose architecture is shown in Fig. 5.10, resembles the above-described 'n-class' system. The only difference is that the neural network has one output in this case. It is then able to distinguish two categories only. These categories are artificially made by allocating any selected category to 'class one' and all the other categories from the training set to 'class zero'. The number of category chosen as 'class one' can be entered in the options window (Classification - Selected category). In the case of N categories represented in a data set, N different neural networks can be trained for data classification, using **Classification|ANN: one-class (training)** menu item.

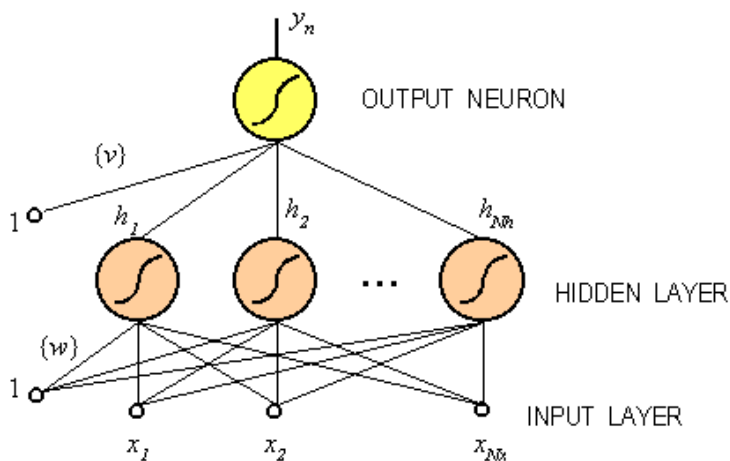


Fig. 4.5.10. Artificial neural network architecture for 'one-class' data classification.

4.5.4. Neural-network classifier testing

Any trained neural network classifier should be tested on a separate test-data set. This applies to neural networks trained as a result of running [NDA analysis](#), as well as [n-class](#) and [one class](#) classification. The **b11** program provides tools for such testing, as shown in Fig. 4.5.11.

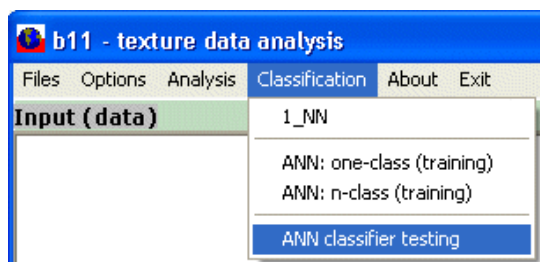


Fig. 4.5.11. Main menu items showing option for neural-network classifier testing.

The test-data set for a neural-network classifier should contain image feature vectors that are different from the ones contained in the set that has been used for training. Both, training and testing data sets can be prepared in **MaZda**, in the form of ***.sel** files.

Prior to calling the testing procedure, one should load the test-data set ***.sel** file using the **Files|Get input data** main menu item.

Information about the architecture and weights of the trained neural network classifier (to be tested) are specified in a corresponding ***.nda** or ***.cls** file. After selecting the testing option shown in Fig. 4.5.11, the appropriate file can be loaded into the **b11** program. Then testing is performed.

The report of classifier testing can be saved to disk in a ***.tst** text-file format, by calling the **Files|Save Report** menu option.

B11 - program for data analysis and classification

4.6. Options

Load the supplied **m25.sel** input file as described in [Getting started](#) help section. Click the **Options** menu item. The options window will appear on the screen (Fig. 4.6.1). Five features (*teta1*, *teta2*, *teta4* and *sigma*) have been found in the input file, as indicated in the 'Feature selection box' of the options window. By default, they are all enabled for further analysis, which is marked by 'tick' signs placed next to the name of each feature. First 3 enabled features

from the list are used to graphically present the input data categories in the feature space (Figs. 4.3.4 and 4.3.5 in [Getting started](#)).

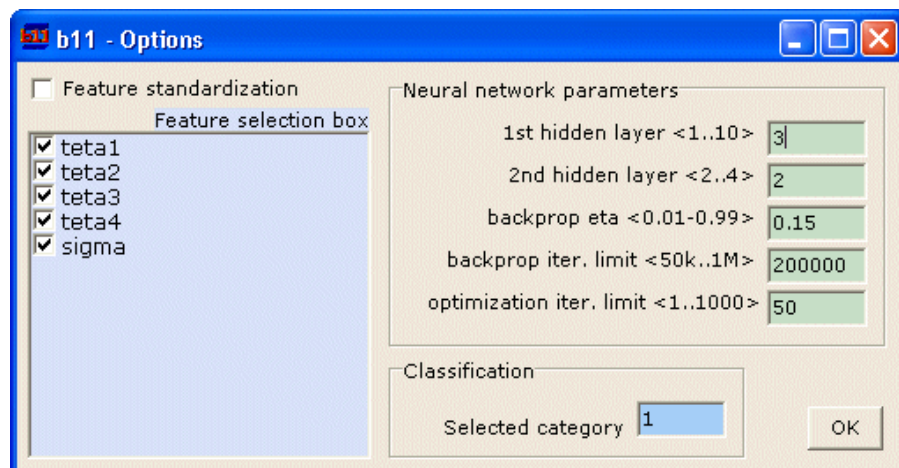


Fig. 4.6.1. Options window after loading m25.sel file.

Uncheck *teta2* and *teta4* features in 'Feature selection box' of the options window (Fig. 4.6.2). Now only 3 out of 5 features are selected for further analysis.

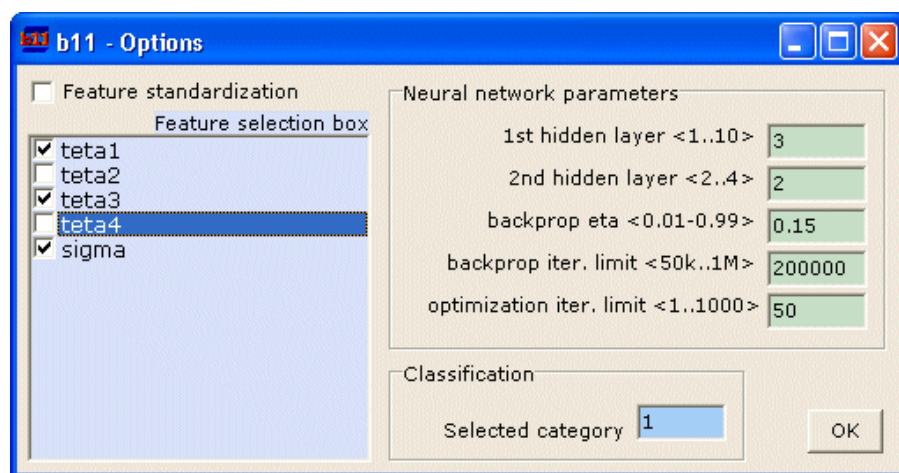


Fig. 4.6.2. Modified options window (two features disabled).

Close the options window (click on **OK** button). Select **Analysis|Raw data** menu item. A graphic window will appear showing five data clusters in the [*teta1*, *teta3*, *sigma*] space. Click **Flip"teta3" axis** menu item to better visualize data categories (Fig. 6.3).

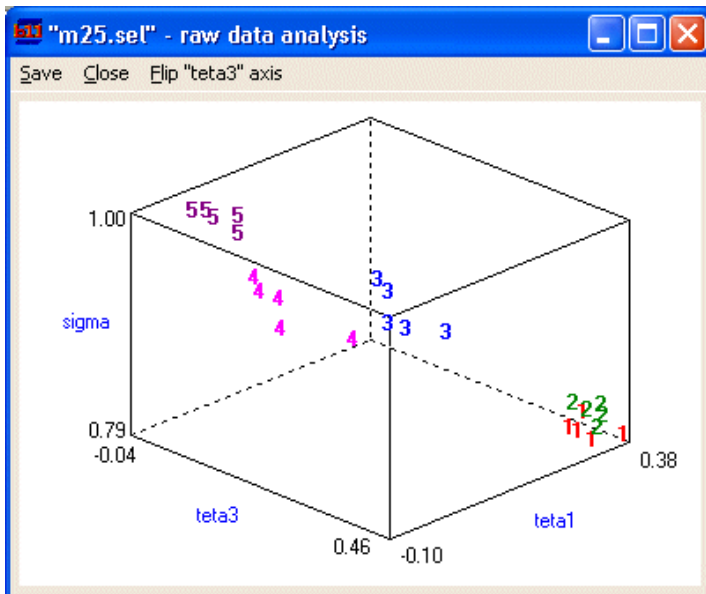


Fig. 4.6.3. Clusters of data vectors in (teta1,teta3,sigma) space for m25.sel file

The 'Feature standardization' option is disabled by default (Fig. 4.6.1); however its is enabled for NDA analysis (Table 4.6.1). If this option is enabled, mean value is subtracted from each feature and the result is divided by the feature standard deviation. The mean value and standard deviation are displayed in the output panel of the main window for an analysis procedure executed.

Table 4.6.1. Default status of 'Feature standardization' option

Analysis type	Default status
Raw data	disabled
PCA	disabled
LDA	disabled
NDA	enabled *)

*) The status can not be changed for this analysis mode.

Inside the options window, there is also a group of neural network adjustable parameters, which are related to [NDA analysis](#), and neural-network '[one-class](#)' and '[n-class](#)' classification (Fig. 4.6.1).

MaZda User's Manual

5. 3D Data Analysis Module

3D Data Analysis Module

5.1. Introduction

MaZda software, since version 4.00, is capable of 3D data analysis. It loads 3D data from image file series, where each file stores single cross-section of 3D volume, or from a single file that stores complete 3D information. Such data is usually produced by medical scanners as CT or MRI.

The **3D Editor** of **MaZda** lets the user define three-dimensional regions of interest (ROI) also called volumes of interest (VOI). The **3D Editor** provides tools for placing volumes of predefined shape within 3D space of image data, resizing them, stretching along selected direction and rotating. There is also a tool implemented for semi-automatic segmentation of 3D image data with elastic surface model.

MaZda computes a number of textural features within selected 3D regions of interest. The resulting feature lists (or feature vectors, one vector per ROI) are presented within the Report window. The feature vectors may be further statistically examined as described in sections [1.5](#) and [2.5](#).

3D Data Analysis Module

5.2. Switching to the 3D analysis mode

After **MaZda** is started it usually shows a window for 2D-image manipulation. To switch to 3D image mode, one has to select **View->3D Editor** menu option or press **[Ctrl]+[3]** from a keyboard. The **3D Editor** window will appear. Through the **3D Editor** window the tools for 3D data loading, ROI editing and 3D volumes analysis are available. See the following sections for details.

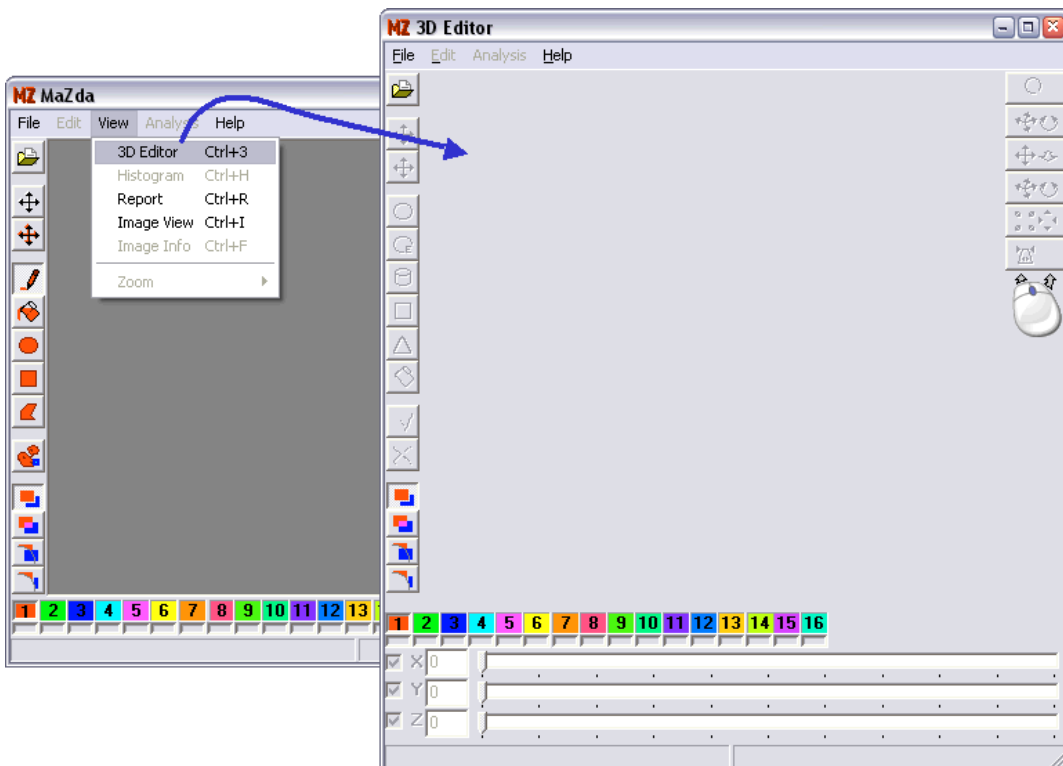


Fig. 5.2.1 Switching to the 3D analysis mode

3D Data Analysis Module

5.3. Loading 3D data

To load 3D image data into the **3D Editor** one has to select **File->Load image...** from **3D Editor** menu, press **[Ctrl]+[L]** from a keyboard or press **Load image** tool button. The **Load 3D data** dialog window appears to choose input file type and to select file or files to be loaded. If several image files (all the

files must be stored in a single folder) are to be loaded, one has to select a group of files by a typical Explorer's **Shift+Ctrl+Mouse** method, and then to press **Open** button.

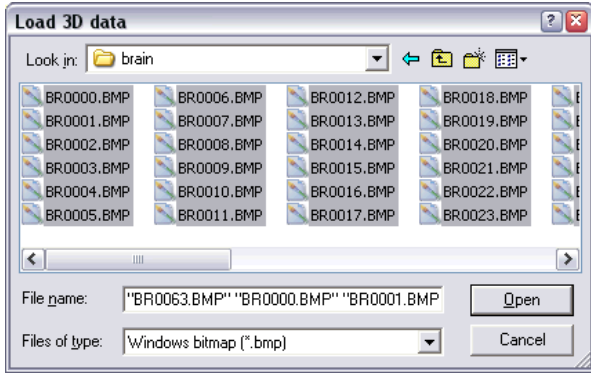


Fig. 5.3.1. Loading 3D data

After the data is loaded it is presented within the **3D Editor** window in a form of three perpendicular cross-sections. Quite often the image is being presented in wrong proportions. The cause of this appearance is a way the image is produced by a medical scanner. Three-dimensional volume is scanned over several parallel cross-sections. The distance between cross-sections is usually higher than distance between adjacent pixels within a cross-section image. Or the other way, the depth of a 3D data voxel (the 3D image element) is different from its width or height. To gain appropriate proportions of presented image, one needs to adjust these proportions manually by invoking the **3D load options** window (**File->Image options...** from **3D Editor** menu) and setting a voxel proportions appropriately.

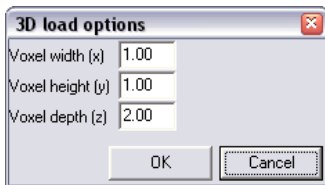


Fig. 5.3.2. Setting a voxel proport

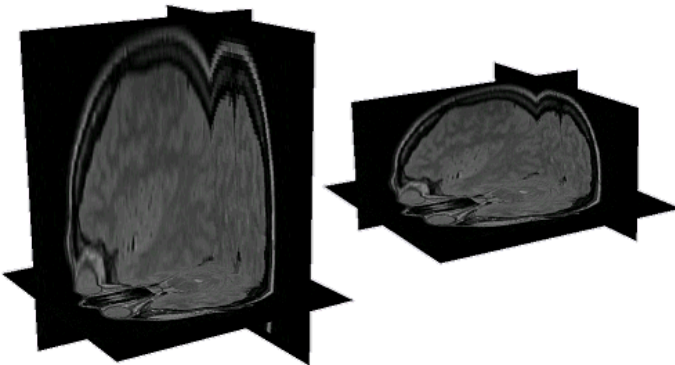


Fig. 5.3.3. 3D data before and after setting the right proportions

3D Data Analysis Module

5.4. Adjusting the 3D data view

The way the 3D data are presented can be adjusted by means of push, pull and turn tools (at the top on the RHS of the **3D Editor** window), and with slide bars (at the bottom of the **3D Editor** window). The view angle can be adjusted after selecting **Turn all** tool (Fig. 5.4.1). The adjustment of pitch and yaw angles is achieved by pressing the left mouse button and sliding the mouse cursor across the **3D Editor** window. The roll angle is adjusted by pressing the right mouse button and moving the mouse cursor horizontally across the window.

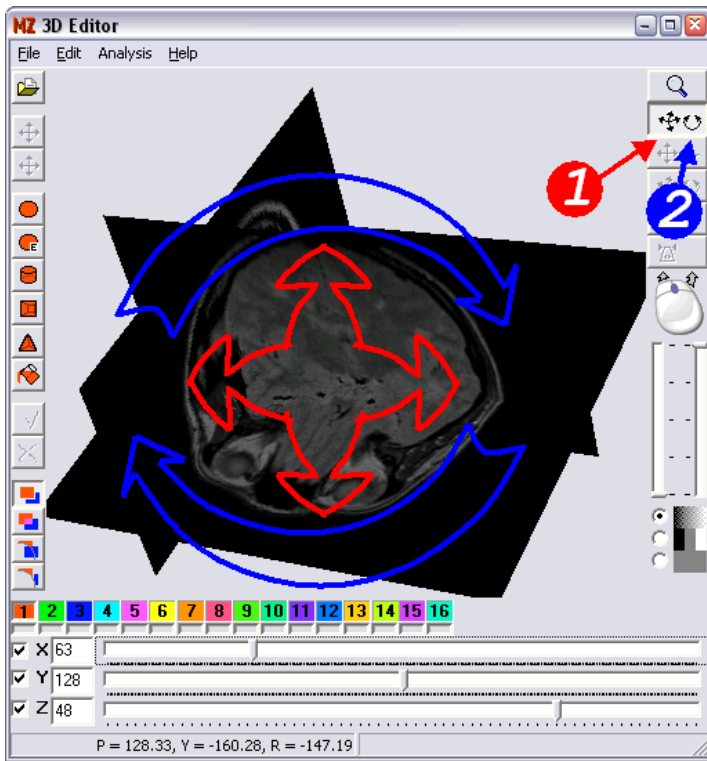


Fig. 5.4.1. Adjusting the view angle

The viewed data can be zoomed in or out, or rather pulled toward or pushed outward, by means of **Push or pull all** tool. The user has to select the tool and then, while pressing the left mouse button, move the mouse cursor vertically across the **3D Editor** window.

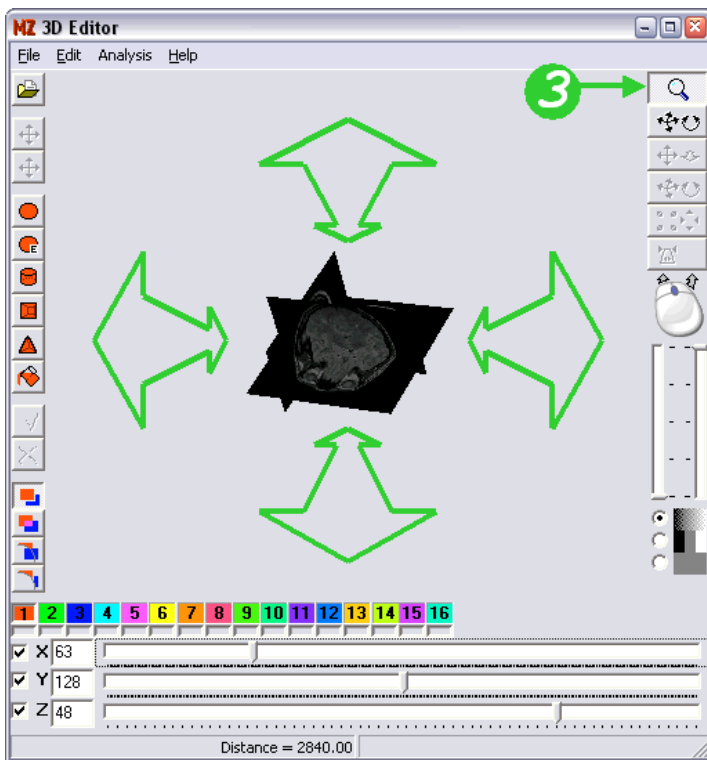


Fig. 5.4.2. Zooming the view

The data is viewed in a form of three perpendicular cross-sections. Some of these cross-sections may be excluded from viewing by unchecking the **X, Y** or **Z** check box. The location of cross-section can be adjusted by one of three slide-bars located at the lower part of **3D Editor** window.

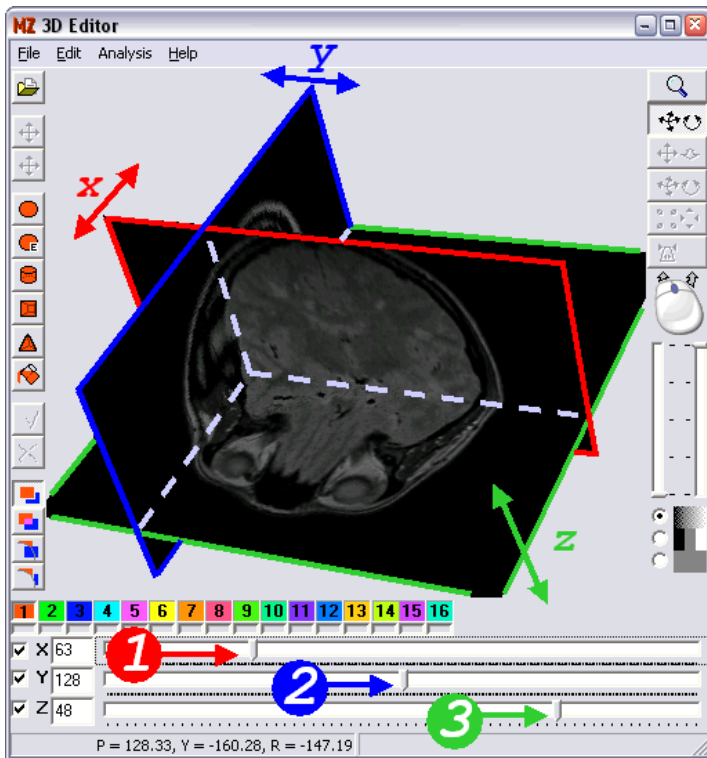


Fig. 5.4.3. Cross-section adjustment

The gray-scale window limits on the brightness scale can be adjusted by setting the position of two slide-bars located vertically on the RHS of the window. These adjustment tools are useful for correcting the image contrast or overall brightness of presented data. Below the slide-bars there are three radio buttons located. Choosing the middle one will cause the image to be displayed in three gray-levels. The slide-bars in this mode serve as gray-level threshold selectors. Choosing the lower radio button will cause the data to disappear. The viewed cross-sections are uniformly gray (if no ROI is defined). These two modes of image presentation are useful when editing regions of interest with a **Flood fill** tool (see section 5.5.3).

3D Data Analysis Module

5.5. Editing regions of interest

Basic purpose of **MaZda's 3D Editor** is to let the user define shapes of regions, within which further analysis is to be performed. Up to 16 regions can be defined. These regions may overlap if it is necessary. The region shape is defined with up to a single voxel accuracy.

5.5.1 Editing regions with predefined blocks

To start the edition process it is required to select the color (number) of region to be created from the palette below the image presentation area of the **3D Editor** window. By default, the red region (no. 1) is selected. Next the user has to select the "drawing" mode. There are four modes, selectable with buttons located at the bottom of toolbox menu on the LHS of the window. In the non-overlapping ROI mode (default) the newly created region will erase other regions already created. In this mode every voxel of 3D data belongs to a single region only. In overlapping ROI mode the newly created ROI will not erase other regions. Thus, the regions will overlap if created at the same locations, and it is possible that a single voxel will belong to several regions. Two other modes are for erasing fragments of already existing regions. In erase active ROI mode, only a region (or its fragment) of selected color will be erased. In erase all mode, erasing will affect all the regions.

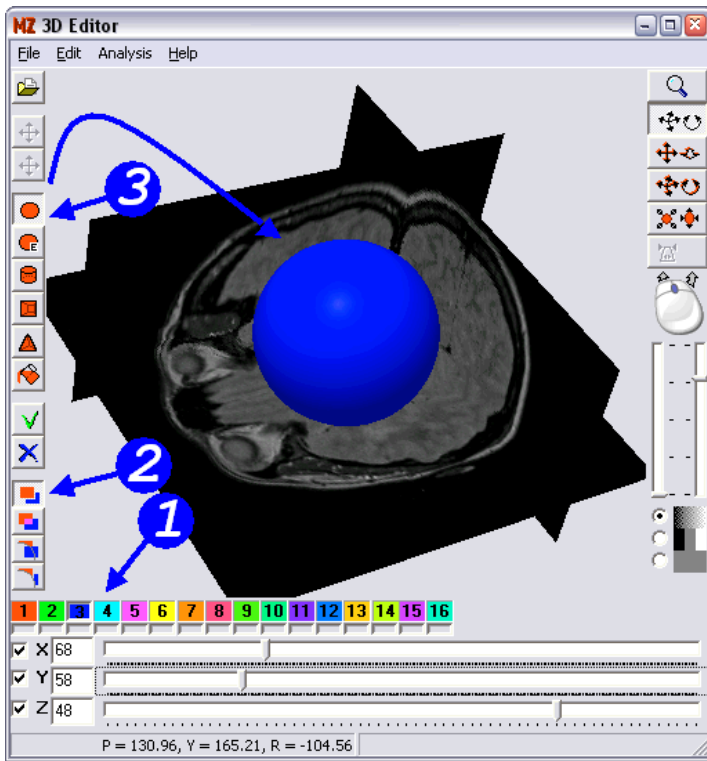


Fig. 5.5.1. Making a ROI of predefined blocks. Selecting: (1) the region's color, (2) drawing mode and (3) a block to be inserted

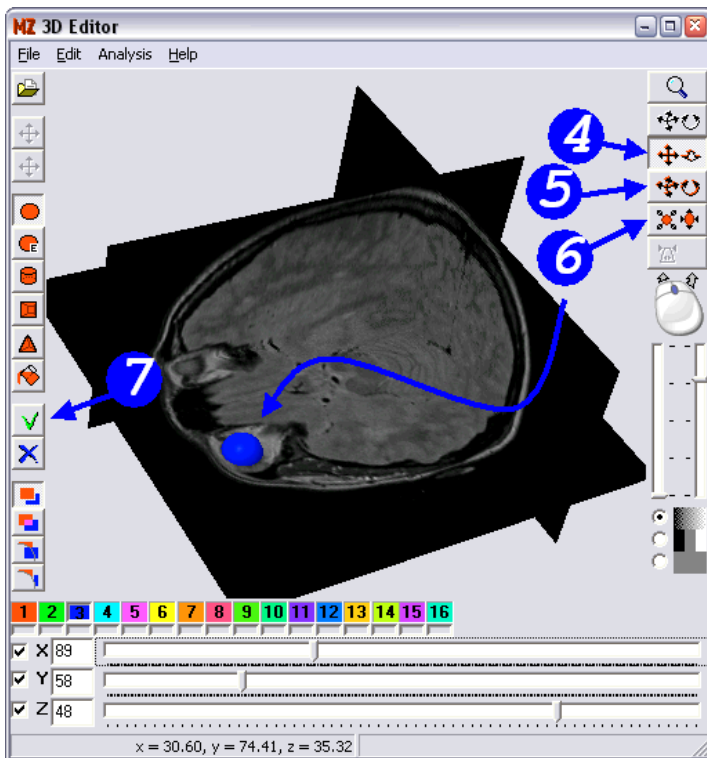


Fig. 5.5.2. Adjusting a (4) location, (5) orientation and (6) sizes of a selected block, and (7) accepting the result.

The simplest way of making a new ROI is to use a predefined blocks. There are four blocks at a choice, sphere, tube, cube and tetrahedron, selectable with the toolbox located on the LHS of the **3D Editor** window (Fig. 5.5.1). At first a block is located at the center of 3D data space. Its location, orientation, sizes, and in some extent shape can be adjusted with tools accessible from the toolbox located on the RHS of **3D Editor** window (Fig. 5.5.2). While adjustments are being done some parameters regarding location, orientation angles or size are printed on the status bar. After adjustments are completed the resulting block can be accepted to augment a ROI of selected color. It can be done by pressing the button with a green checkmark. Clicking the button with a red X will cancel the action.

5.5.2 Switching regions on and off from viewing

The created regions are visible in a form of color highlights. Sometimes it is handy to temporary hide this highlights. The user can switch the ROI on and off from viewing by clicking tiny bars located just below color (ROI number) selection buttons. These tiny bars have also informative function. They are colored if the corresponding region exists and grayed otherwise.

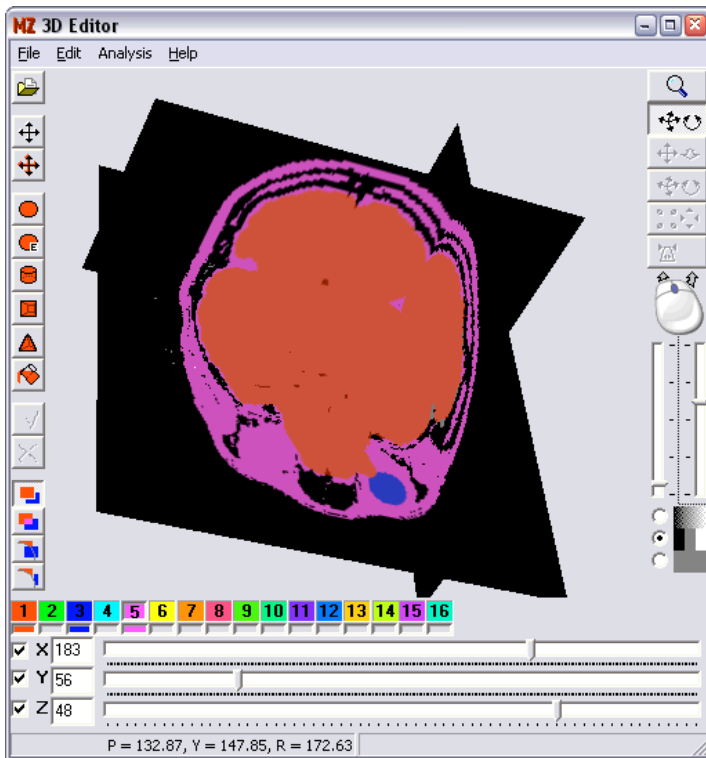


Fig. 5.5.3. Viewing ROI

5.5.3 Flood-filling the volume

The ROI may be also created by means of flood-filling tool. To get the volume flood-filled, it is convenient to switch the image to three gray-level presentation mode (section 5.4) and set up the gray-level scale thresholds to distinguish the considered volume. Next step is to select the **Flood-fill** tool (the button with a jug on it). Now, the user has to point with a mouse cursor at the volume the flood-fill procedure is to fill in. It must be noticed that switching regions off from viewing or the drawing mode will affect the result of flood-filling.

5.5.4 Defining a ROI with an elastic surface model

An elastic (or deformable) surface model is a mathematical model of enclosed surface that deforms upon some local characteristics of 3D image. The deformation is constrained by tensions modeled within the surface. The final shape of the surface is a compromise, in which these two effects are balanced. The final shape of the model strongly depends on initial shape and location of the surface. In **Mazda's 3D Editor** the model of center point deformable surface has been implemented for ROI editing. To start the model, one has to select **Create spherical-elastic ROI** from the toolbox (Fig. 5.5.4). The model by default operates in automatic mode. The image influence and tension parameters are set automatically. The user is able to choose the initial size and location of the surface with use of move, turn, inflate and deflate tools located on the RHS of the window. In addition the user can control the two parameters (**Thresh.** and **Deform.**) after switching to one of non-automatic modes of operation. After the satisfactory shape of elastic surface has been obtained, the user may approve it by pressing the **Accept** button (green checkmark).

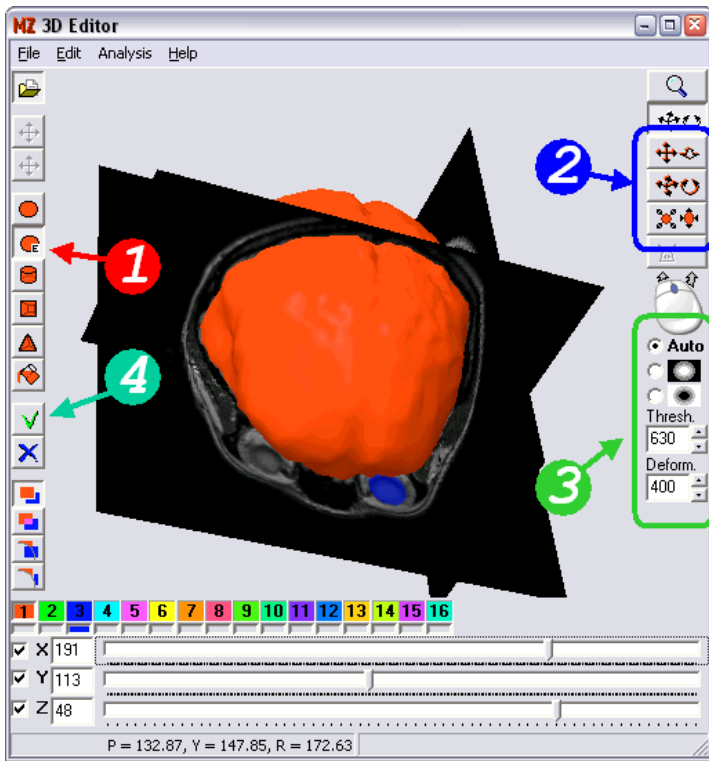


Fig. 5.5.4. Elastic surface tool for ROI editing: (1) selecting the tool, (2) adjusting the initial size and location of spherical surface, (3) adjusting the model parameters and (4) accepting the final shape

5.5.5 Moving and copying regions

Any existing region can be moved or copied. To move or copy the region, user has to select the color (number) region to be moved or copied from a palette toolbox. Next step is to select the **Move** or **Copy** button, both located at the top of the toolbox on the RHS of the window. The three-dimensional representation of selected region will appear. The user can change the location of the region by means of move, push or pull tools and a mouse. After the new location of the region had been obtained it has to be accepted by pressing the **Accept** button (green checkmark).

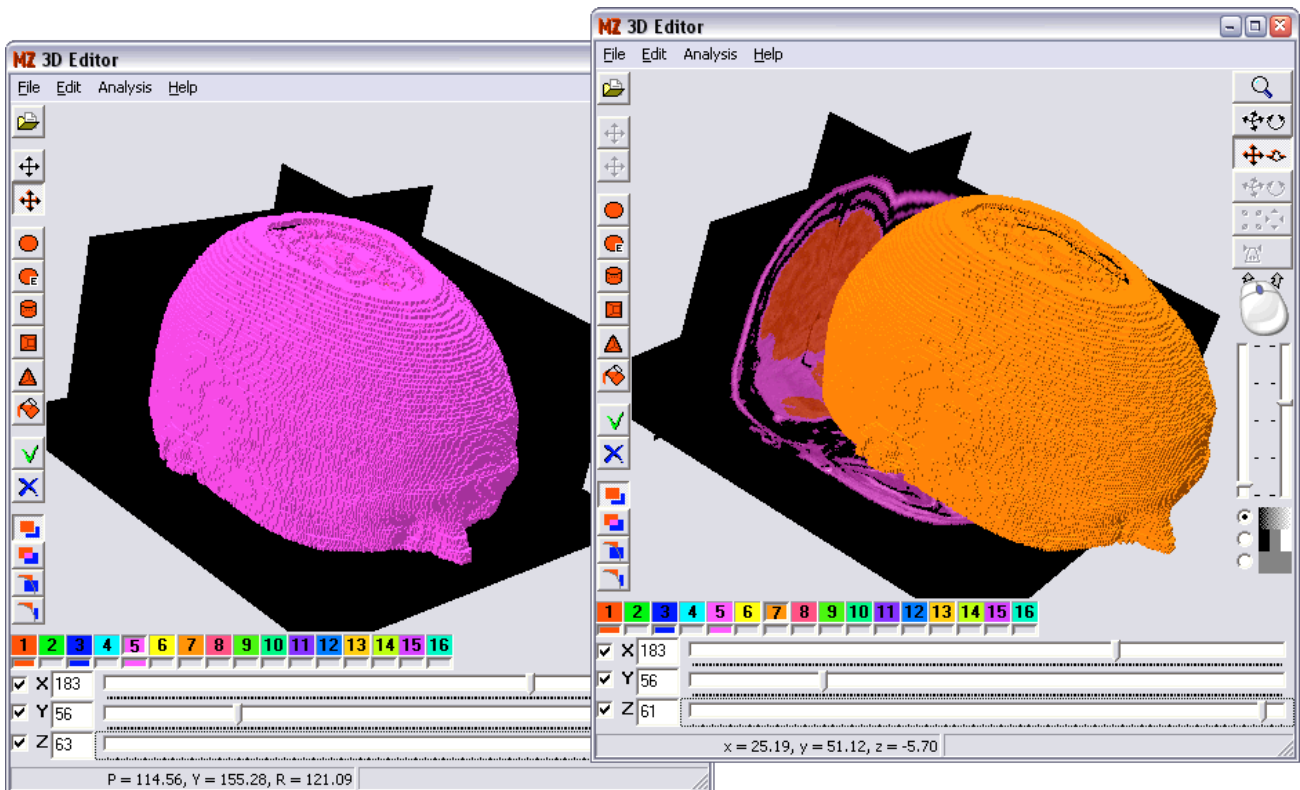


Fig. 5.5.5. Copying the ROI

Comment. It is possible to change the color of ROI before accepting its new location. In such a case the region will be moved or copied given a new selected color.

5.5.6 Editing individual cross-sections in 2D window

The region of interest shape can be edited cross-section by cross-section. After switching to the main (2D) window of **MaZda** program the currently selected cross-section perpendicular to a Z direction is automatically uploaded the 2D window (Fig. 5.5.6). The ROI can be edited by means of 2D editing tools. After switching back to **3D Editor** window the selected cross-section will be updated. On how to use 2D window tools refer to the section [1.3](#) and [2.3](#).

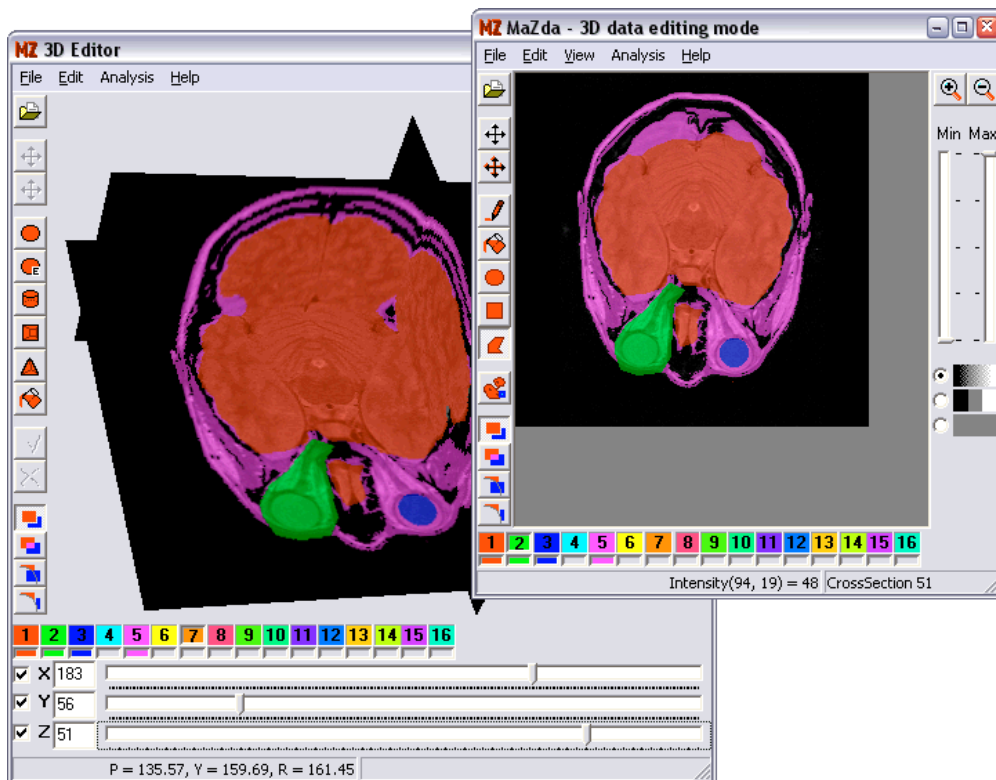


Fig. 5.5.6. 3D Editor window and main MaZda window while editing individual cross-sections

5.5.7 Undoing and redoing editing steps

The latest edition steps can be undone by selecting **Edit->Undo** option from the **3D Editor** menu. If the edition step is required, selecting **Edit->Redo** option from the menu can restore it.

5.5.8 Defining a class name of 3D ROI

The individual ROI can be assigned a class name. The class names of regions are transferred to the report window after computing textural features within regions, and then they are used in a further process of data analysis. To assign a class name to a ROI, the user has to right-click on a color palette toolbox button. The dialog window will appear for entering the class name. After the ROI is assigned a class name, the class name will be shown while hovering with the mouse cursor over the ROI selection buttons of the color palette.

5.5.9 Saving and loading regions of interest

The 3D regions of interest can be saved to a file or load from a disk file. The disk files for 3D ROI data storage are given a **.voi** extension. To save the 3D ROI data the user has to select **File->Save ROI...** option from the **3D Editor** menu. To load 3D ROI from a file the user has to select **File->Load ROI...** option from the menu.

3D Data Analysis Module

5.6. Analysis of 3D data

MaZda computes a number of textural features within selected 3D regions of interest. To select options of the computation process the user has to select **Analysis->Options...** option from the **3D Editor** menu. The analysis options can be then set with the Options dialog window. The image can be analyzed as is (**Normalization** option set to default) or normalized (**+/- 3 Sigma** or **1%-99%** option set) before textural features computation.

There are two types of 3D image analysis implemented in the current version of **MaZda**, based on image histogram computation and on co-occurrence matrix. These types of analysis may be selected by checking or unchecking the **Histogram features** and **COM features** checkboxes. To select the number of bits used for image quantization before computing the textural features, one has to click on corresponding radio buttons within the respective frame inside the dialog box. Additionally, the COM parameters depend on the distance between pairs of image points that has to be specified to compute the co-occurrence matrix. Up to five different values of the distance are supported through the corresponding checkboxes.

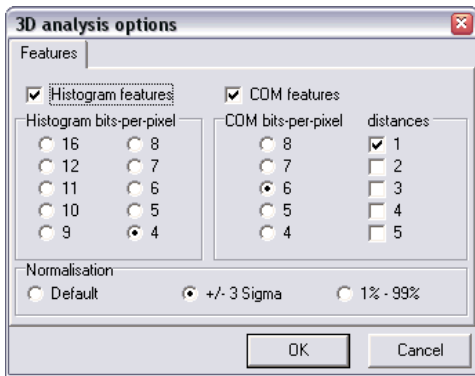


Fig. 5.6.1. 3D analysis options window

Analysis of the 3D data within the defined regions of interest starts after selecting the **Analysis->Run 3D analysis...** option from the **3D Editor** menu. The results of the analysis are presented in the **Report** window. How to further analyze the data is described in sections [1.5](#) and [2.5](#)

6. References

1. M. Dash and H. Liu, Feature selection for Classification (1997) Elsevier Science Inc., <http://www-east.elsevier.com/ida/browse/0103/ida00013/article.htm/>
2. R. Duda and P. Hart (1973) *Pattern Classification and Scene Analysis*, Wiley.
3. C. Giardinia and E. Dougherty (1998) *Morphological Methods in Image and Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.
4. J. Freeman and D. Skapura (1991) *Neural Networks - Algorithms, Applications and Programming Techniques*, Addison-Wesley.
5. K. Fukunaga (1991) *Introduction to Statistical Pattern Recognition*, Academic Press.
6. P. Gallinari, S. Thiria, F. Badran and F. Fogelman-Soulie (1991) On the Relations Between Discriminant Analysis and Multilayer Perceptrons, *Neural Networks*, **4**, 349-360.
7. R. Haralick, K. Shanmugam and I. Dinstein, Textural Features for Image Classification (1973) *IEEE Transactions on Systems, Man and Cybernetics*, **3**, 6, 610-621.
8. R. Haralick, Statistical and Structural Approaches to Texture (1979) *Proceedings of the IEEE*, **67**, 5, 786-804.
9. R. Hecht-Nielsen (1989) *Neurocomputing*, Addison-Wesley.
10. Y. Hu and T. Dennis (1994) Textured Image Segmentation by Context Enhanced Clustering, *IEE Proc.-Visual Image and Signal Processing*, **141**, 6, 413-421.
11. W. Krzanowski (1988) *Principles of Multivariable Data Analysis*, Oxford University Press.
12. N. Kwak and C.H. Choi (2002) Input Feature Selection by Mutual Information Based on Parzen Window, *IEEE Transactions on Patter Analysis and Machine Intelligence*, **24**, 12, 1667-1671.
13. R. Lerski, K. Straughan, L. Shad, D. Boyce, S. Bluml and I. Zuna (1993) MR Image Texture Analysis – An Approach to Tissue Characterization, *Magnetic Resonance Imaging*, **11**, 873-887.
14. J. Mao and A. Jain (1995) Artificial Neural Networks for Feature Extraction and Multivariate Data Projection, *IEEE Trans. on Neural Networks*, **6**, 2, 296-316.
15. A. Materka and M. Strzelecki (1998) [Texture Analysis Methods – A Review](#), COST B11 report (presented and distributed at MC meeting and workshop in Brussels, June 1998), Technical University of Lodz, Poland.
16. A. Materka, M. Strzelecki, R. Lerski and L. Schad (1999) Feature Evaluation of Texture Test Objects For Magnetic Resonance Imaging, *Workshop on Texture Analysis and Machine Vision*, Oulu, Finland, 13-19.
17. A. N. Mucciardi and E. E. Gose (1971) A comparison of Seven Techniques for Choosing Subsets of Patter Recognition Properties, *IEEE Trans. on Computers*, **c-20**, 9, 1023-1031
18. W. Press, S. Teukolsky, W. Vetterling and B. Flannery (1996) *Numerical Recipes in C*, Cambridge University Press.
19. J. Schürman J. (1996) *Pattern classification*, John Wiley & Sons.
20. D. Swets and J. Weng (1996) Using Discriminant Eigenfeatures for Image Retrieval, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **18**, 8, 831-836.
21. G. D. Tourassi, E. D. Frederick, M. K. Markey and C. E. Floyd, Jr. (2001) Application of the mutual information criterion for feature selection in computer-aided diagnosis, *Medical Physics*, **28**, 12, 2394-2402.

MaZda User's Manual

7. Copyright

- MaZda program code ©1998-2006, **MaZda** User's Manual chapter 5 ©2006 by [Piotr Szczypinski](#)
- Image file loader ©1992-1994 by Michael Friedlinger
- Wavelet analysis module ©2001-2002 by [Marcin Kociolek](#)
- B11 data preprocessing and classification module code ©1999-2006, **MaZda** and **B11** User's Manual ©1999-2006 by [Andrzej Materka](#)
- Supervised feature selection module code ©1999-2006 by [Michal Strzelecki](#)
- Mutual information feature selection module code ©2005 by Slawomir Razniewski
- Unsupervised feature selection and segmentation modules ©2005-2006 by [Artur Klepaczko](#)