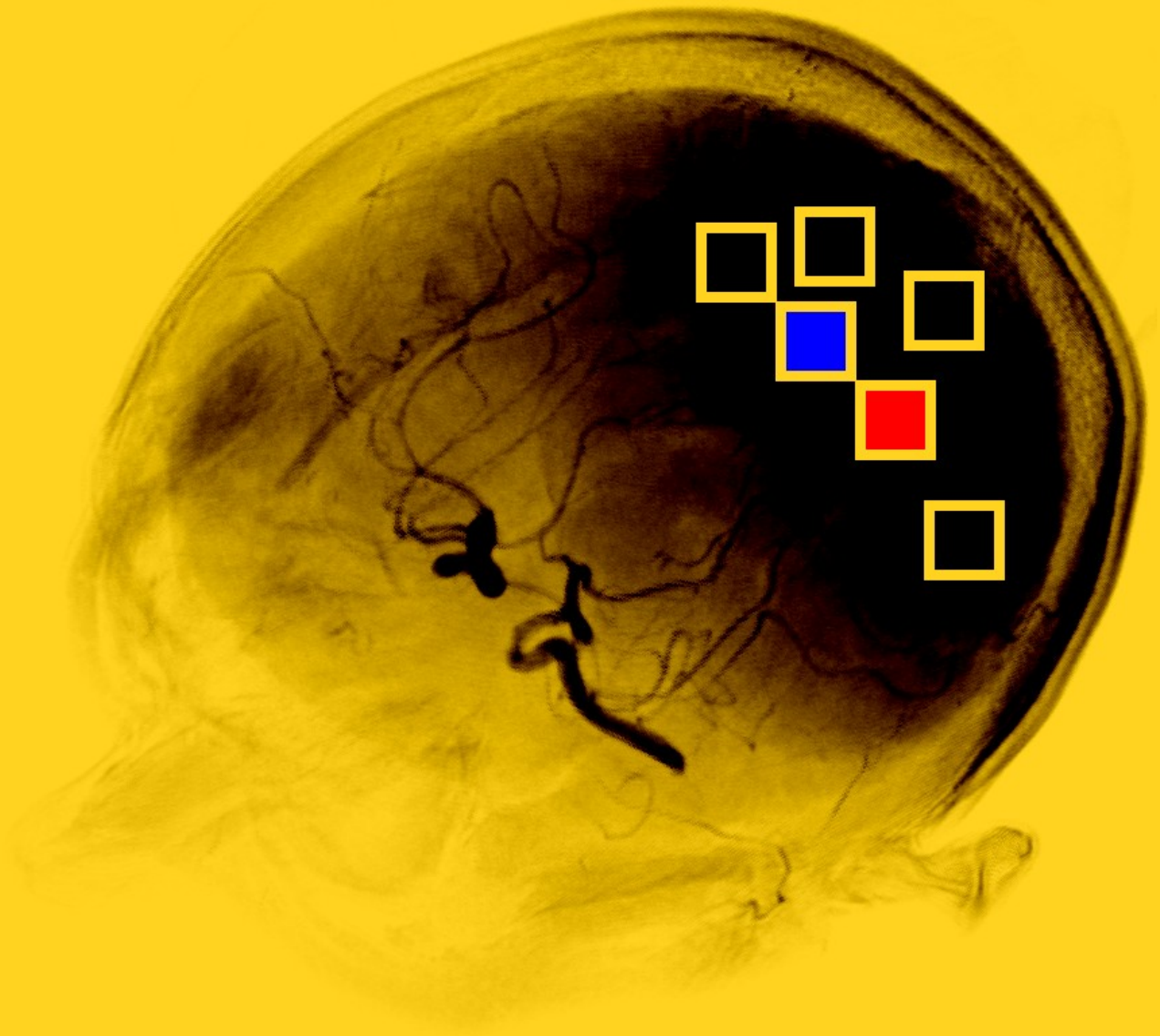


qmazda manual

Piotr M. Szczypiński

2020-12-02

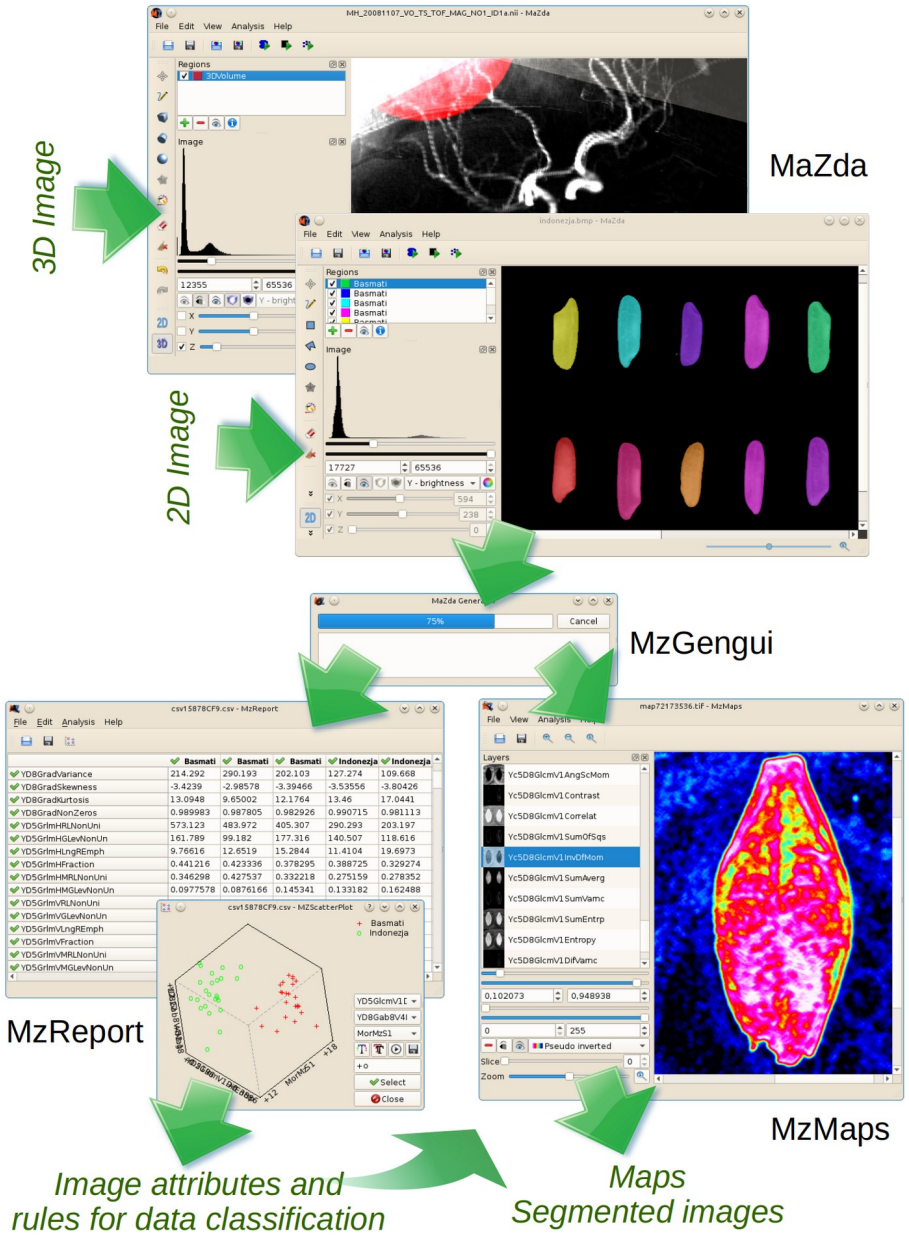
Introduction

Qmazda is a software package for digital image analysis. It computes shape, color and texture attributes of arbitrary regions of interest. Also, it implements algorithms of discriminant analysis and machine learning. *MaZda*, which was the previous version of the texture analysis software, has been developed since 1996. The first algorithm implemented in this software derived attributes from the gray-level co-occurrence matrix (*Macierz Zdarzeń* in Polish). The name *MaZda* is thus an acronym of *Macierz Zdarzeń* and has no connotation with the Japanese car make. The *qmazda* project is to further develop *MaZda* program, make it an open source and port the implemented algorithms previously available for Windows systems also to Linux and OS X platforms.

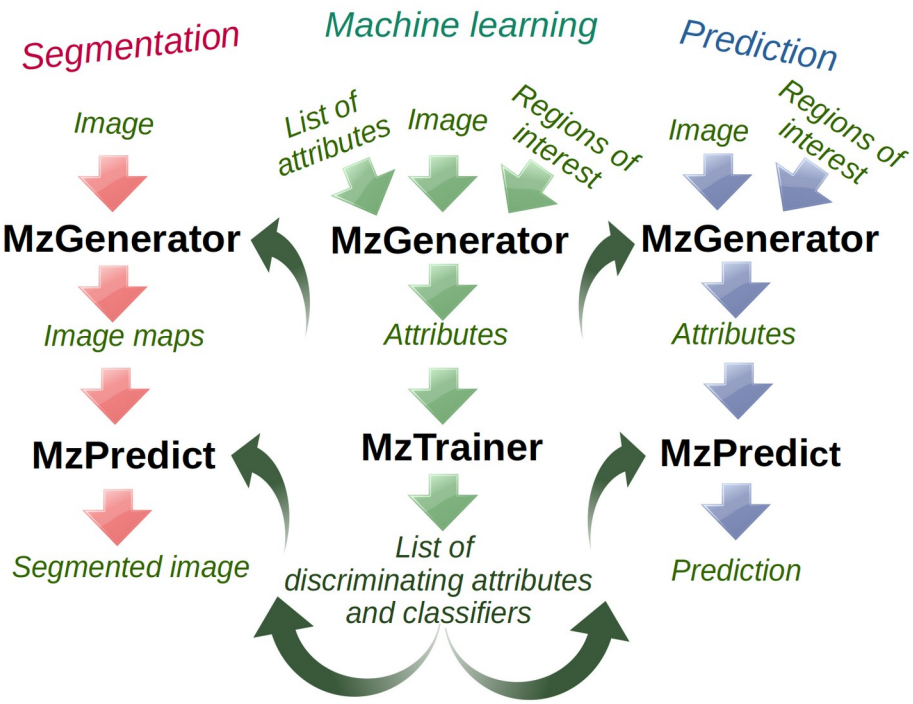
Qmazda package consists of four programs, which have graphical user interfaces. These are *MaZda*, *MzGengui, MzReport* and *MzMaps*. These programs together create an image analysis workflow. The *MaZda* module enables to define regions of interest in the image and to make choice which image attributes are going to be extracted from the image. The second module, *MzGengui*, computes color, texture and morphological attributes. The resulting data may be represented as feature vectors (each vector characterizing individual region of interest) or by feature maps (images, which gray-levels represent local values of the attributes). The results of computation are then submitted to the *MzReport* or to the *MzMaps* module. *MzReport* presents the vectors of computed attributes in a spreadsheet and enable visualization of their distributions. Moreover, it enables discriminant analysis, selection of most discriminative attributes, machine learning and tests of data classification. *MzMaps* shows the feature maps and enables image segmentation based on color and texture attributes.

Qmazda also includes three console applications. These programs are *MzGenerator*, *MzTrainer* and *MzPredict*. The function of *MzGenerator* reflects functionality of the *MzGengui*, and thus it computes region-based feature attributes and feature maps. The input list of attributes to be computed, input image and regions of interest are specified through the command line arguments. *MzTrainer* covers selected functions of the *MzReport*, among others they are procedures for selection of the most discriminating attributes and machine learning. The result of the *MzTrainer* is a file with classification rules. The third application, *MzPredict*, enables data classification or image segmentation based on the rules produced by the *MzTrainer* or *MzReport*.

The following diagrams present workflows for image analysis, machine learning and classification paths by means of the programs with a graphical user interfaces and by their console counterparts.



Workflow available with interactive applications: image processing, data extraction, machine learning, data classification and texture-based image segmentation



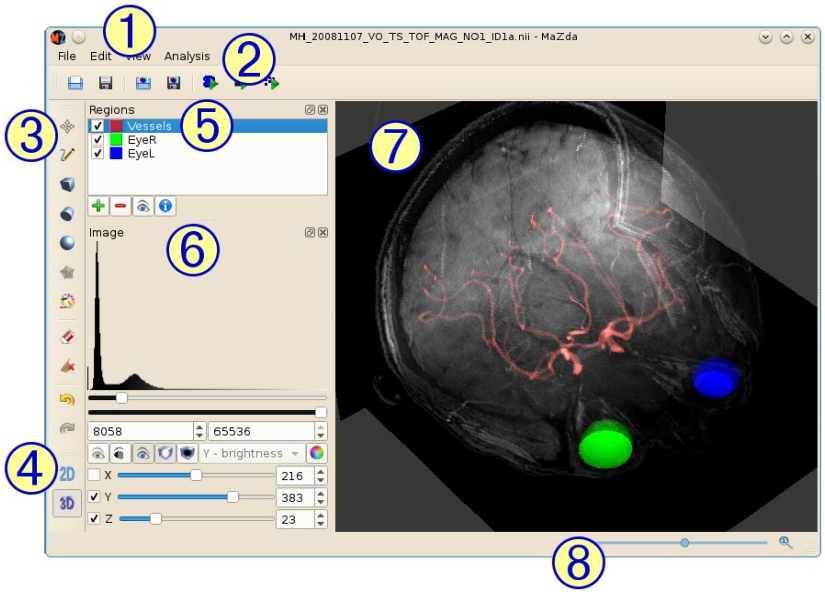
Workflows available with console applications: data extraction, machine learning, data classification and texture-based image segmentation

MaZda

MaZda module enables loading and visualization of 2D and 3D images. The loaded images are always presented in grey-scale. Color images are converted to monochrome (grey-scale) images and then displayed. Color overlays are used in MaZda to indicate regions of interest. Such the regions are used to limit further analysis to the selected image fragment. MaZda enables defining (drawing) regions of arbitrary size and shape.

Working with the MaZda module consists in loading the image, defining regions (or volumes) of interest for the following analysis, compiling a list of features to be extracted, and execution of the analysis. Working with the MaZda module consists in loading the image, defining regions (or volumes) of interest for the following analysis, compiling a list of features to be extracted, and execution of the analysis.

MaZda window layout



① main menu

② load, save and analysis tools

③ region drawing tools

④ dimensionality switches

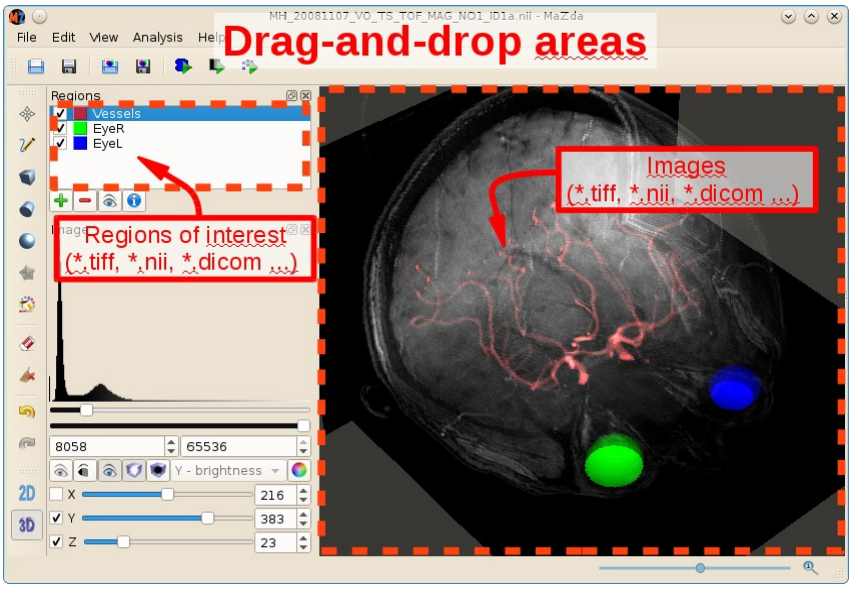
⑤ region of interest panel

⑥ image viewing options panel

⑦ image rendering area

⑧ status bar with a zoom slider

Loading and saving



To load an image select *File* > *Load image...* option from the menu or press the load image button. Then, use the dialog to select an image file to load. Optionally image file can be dragged-and-dropped to the image rendering area. MaZda automatically recognizes image format and dimensionality, and appropriately sets initial visualization modes. Note that *MaZda* enables loading 3D images from stacks of files. To do this select a list of appropriate files in the image loading dialog or drag-and-drop a list of files. To save the loaded image in another format select *File* > *Save image...* from the menu.

To load regions previously created in MaZda select *File* > *Load regions...* option from the menu or press the load regions button. Optionally regions' file can be dragged-and-dropped onto the regions of interest list area. To save the regions select *File* > *Save regions...* from the menu.

Image viewing options

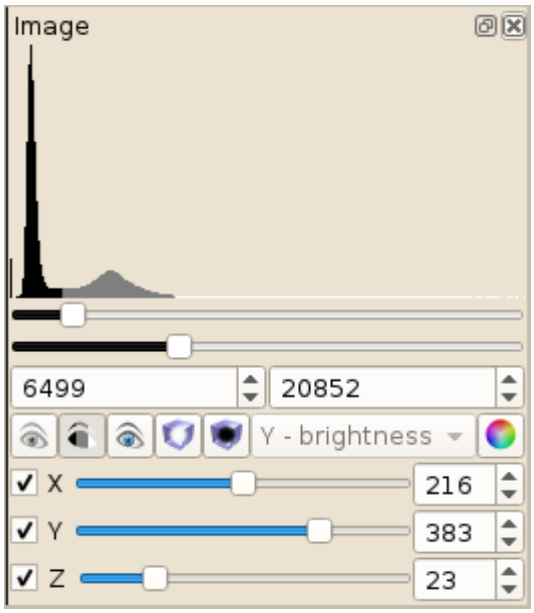


Image view options docking window enables switching the view modes. The upper part shows the image histogram with two sliders and two spin boxes to set gray-scale range.

The *eye* buttons enable selection of visualization modes. The eye button enables visualization in gray-scale range. The three-level mode is used to choose grey-level thresholds. The grayed eye hides an image to enable exclusive visualization of overlays. The two cube buttons enable volumetric rendering of 3D images of white and black objects. The combo (list) box enables selection of color-to-monochrome conversion. The color selection button invokes dialog to define background color for image rendering.

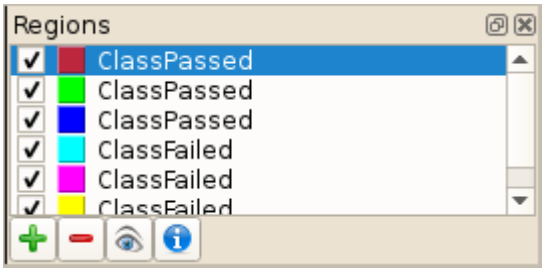
The bottom part consists of three groups of check boxes, sliders and spin boxes to select 3D image cross-sections to be rendered.

The image can be zoomed in and out by means of the slider located at the right-hand-side of the status bar. Also the zoom can be controlled by the menu options (*View*) or by the mouse wheel.

Proportions (voxel spacing) of the 3D image can be modified by means of voxel spacing dialog. To open the dialog select *View* > *Voxel spacing...* from the menu, edit the values appropriately and press *OK*.

It must be noted that the above mentioned options affect image visualization (rendering) only. They do not affect the original image data. Also, modifying these options does not affect image analysis (feature extraction), since the analysis is performed on the original image data.

Regions of interest list



Regions of interest docking window present a list of regions of interest. The name (class name) of any region can be modified by double-clicking and editing of its text field. After feature extraction the name becomes a label of feature vector and may be used in supervised learning procedures. The color of the region and associated image overlay may be changed by double-clicking on the color box. The check-boxes can be used to hide and show corresponding color overlays. The eye button toggles all the regions' check boxes on the list. It must be noted that unchecking particular region does not erase or remove it. Such the region is still used in the analysis. A selected region can be removed from the list, and from the analysis, by pressing the minus button. A new region can be added to the list by pressing the plus button. The information button invokes a dialog showing histogram and information about the image fragment corresponding to the selected item on the list.

Editing regions of interest

First select a region for editing from the list. Regions of interest (2D image) can be drawn by means of drawing tools available from the toolbox. All the modifications will apply to the currently selected region. Be sure the selected region stays checked while drawing.

The tools enable drawing lines, rectangles, polygons and ellipses. To use particular tool the relevant button has to be pressed. Then, the user can draw by means of a mouse and keyboard. The line is drawn as the left mouse button is pressed and the mouse cursor is moved.

To draw a rectangle press the mouse left button as the cursor is located at the rectangle's corner and move the mouse cursor toward location of the opposite corner. As the left mouse button is pressed you can press *Ctrl* button to draw square or *Shift* button to rotate. The drawing is completed when the mouse button is released.

To draw a polygon user has to click on the consecutive vertices of the polygon. To finalize and enclose the polygon double-click the left mouse button or press *Enter*.

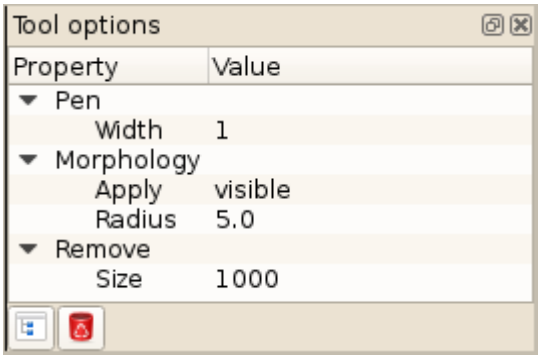
To draw an ellipse user has to press the mouse as the cursor is located at the center of the ellipse and then move the mouse cursor. As the left mouse button is pressed user can press *Ctrl* button to draw a circle or *Shift* button to rotate. The drawing is completed when the mouse button is released.

The flood-filling tool can be used when the image is shown in three-level mode. You need to select the grey-level thresholds by means of sliders located at the bottom of the histogram. Then, click the left mouse button as the cursor is within the area to fill.

The existing region can be moved or copied . To move the selected region press the mouse, move the cursor to the new location and release the mouse button. To copy and merge keep *Shift* pressed when releasing the mouse button. To copy and create a new region keep *Ctrl* pressed.

All the drawing tools can be used in combination with the eraser. As the eraser stays pressed the tools are used to erase fragments of existing region areas. The cleaning tool can be used to immediately erase the currently selected region. Note that the region is cleaned, however it is not removed from the list.

There are additional tools available from the *Edit* menu. The tools enable image thresholding, joining or splitting regions, and mathematical morphology operations.



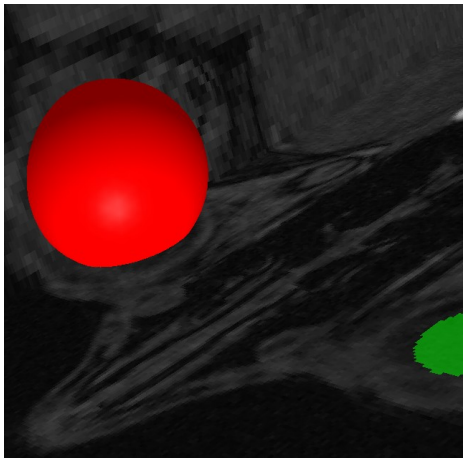
To set drawing tools attributes use *Tool options* dialog (*Edit* > *Tool options*).

Editing volumes of interest

If the 3D image is loaded the image features are computed within volumes of interest. The regions of interest docking window present a list of such the volumes. The toolbox with drawing tools changes to make available tools for shaping 3D volumes.

The tools enable drawing lines and inserting blocks, such as cubes, tubes or ellipsoids. To use particular tool the relevant button has to be pressed. All the modifications will apply to the currently selected volume. The volume can be shaped by means of a mouse and keyboard.

The line is drawn as the left mouse button is pressed and the mouse cursor is moved. The lines are drawn on the surface of cross-sections displayed on the screen.



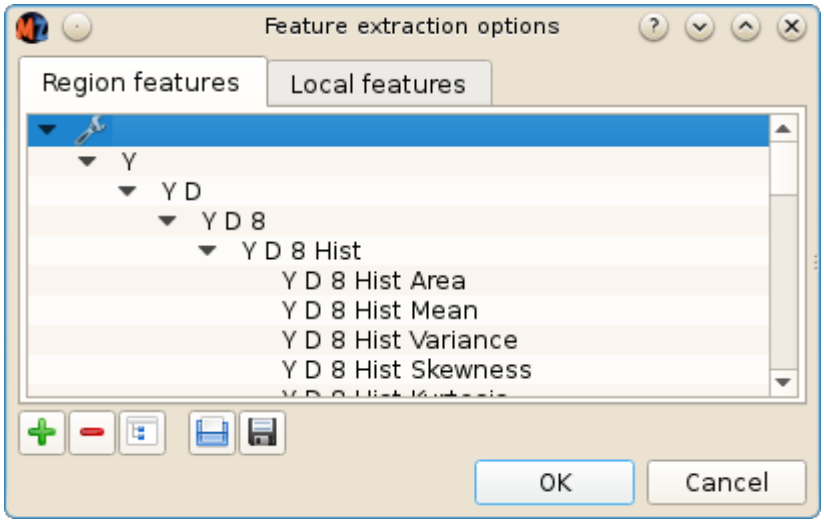
To insert a block, one of the three buttons cube, tube, or ellipsoid, have to be pressed. To insert the block click on the rendering area. The block is initially positioned at the center of the image. Alternatively, if the *Shift* is pressed while clicking, the block will be positioned at the location of previously inserted block. Next, the block can be moved, rotated, resized or its proportions can be changed. To do this move the mouse cursor while pressing the left mouse button or turn the mouse wheel. If no keys on the keyboard are pressed the block is rotated (mouse move) or resized (mouse wheel). If the *Shift* is pressed the block is moved. Pressing the *Alt* (or *Alt*+*Shift*) enables changes of the block proportions. To change orientation and location of the image together with the block keep pressing on the *Ctrl* or *Ctrl*+*Shift* buttons. To complete the procedure double-click the left mouse button or press *Enter*.

The flood-filling tool can be used when the image is shown in three-level mode. You need to select the grey-level thresholds by means of sliders located at the bottom of the histogram. Then, click the left mouse button as the cursor points on appropriate location on one of the displayed cross-sections.

The existing volume of interest can be moved . To move the selected volume click the left mouse button. Then move, rotate or scale the volume in the similar way as described above. To complete the procedure double-click the left mouse button or press *Enter*.

You can edit individual cross-sections of volumes of interest by means of 2D editing tools. To do so select appropriate cross-section to edit by means of *X*, *Y* or *Z* slider. Turn the image in such a way that the cross-section to edit is directed toward you. Press the 2D mode button and edit the volume's cross-section. When you finish editing, press 3D mode button to return to 3D image view mode.

Feature list dialog



Select menu option *Analysis* > *Options* to open *Feature extraction options* dialog and to edit list of features for computation. The dialog consists of two tab-pages with two sets of options, the first one for feature extraction from regions of interest, and the second for computation of features locally applied for map computation or point driven computation.

The feature names are presented as a tree view. Individual tree branches represent specific parameters or stages of computation algorithms. Leaves define complete names of the features for extraction. Double-clicking on the individual branch opens a drop box or a spin box which enable selection of some alternative option. Branch or a leave can be removed by selection it and pressing the button. A new branch can be created by selecting a parent branch and pressing the button.

The edited list of feature can be stored to a file or loaded from a file after pressing buttons available at the bottom of the dialog. It must be noted that loading feature list from a file will augment the existing list with the feature names from the file. To replace the existing list with the new one select the root of the tree and press before loading data from the file.

Refer to the Feature naming section for more information on the feature naming convention, meaning of parameters and the algorithm name abbreviations.

MzGenerator and MzGengui

MzGenerator and *MzGengui* are programs for image feature extraction. *MzGenerator* is a purely console application that does not create any graphical interface. In contrast, *MzGengui* creates a simple window with a progress bar. Except this difference, the two programs accept the same list of command line arguments and implement the same procedures for feature computation.

The programs work in three feature extraction modes. They enable

(1) computation of feature values within predefined regions of interest,
(2) computation of features within specified neighborhoods around indicated locations, and
(3) computation of feature maps.

In mode (1) the input for the program is an image from which the features are extracted together with the regions of interest masks. In mode (2) the input is the image from which the features are extracted, and another image with dots indicating the locations (centers of neighborhoods). In mode (3) the input consists of the image for feature maps computation. The input image from which the features are extracted should be in TIFF (Tagged Image File Format) or NIFTI (Neuroimaging Informatics Technology Initiative) formats. By default *MaZda* uses TIFF in case of two-dimensional images and NIFTI for 3D tomographic images. The masks, if they overlap, should be delivered in multi-page TIFF format. If the masks do not overlap, they may by provided as image with black background and individual region areas filled with unique colors.

In modes (1) and (2) the results are stored into a text file in CSV (comma separated vectors) format. The file can be loaded into *MzReport* module or into spreadsheet programs (eg. *LibreOffice Calc*). The mode (3) produces stacks of images (maps) which grey-scales represent local feature values. The maps are stored in multi-page TIFF format and can be loaded into *MzMaps* module. Also, this format is accepted by selected image processing software (eg. *ImageJ*). In all the modes the additional input for *MzGenerator* is a list of names of features to be extracted from the image.

The list of the program arguments is listed when the program is run with *--help* or */?* switch. The list is as follows:

-m, --mode <mode>	Set <mode> of the analysis to: 'local' or 'roi',
-q, --query	Queries for feature names template
-i, --input <file>	Load image from <file>. 16bit tiff is available
-f, --features <file>	Load feature list from <file>
-r, --regions <file>	Load masks or regions of interest form <file>
-o, --output <file>	Save results to csv or floating-point multipage tiff <file>
-c, --category <string>	Force category name By default a category name is retrieved from regions file
-s, --step <int>	Step in pixels in map computation, default is 1
-x, --save-hex	Save double precision results in hexadecimal format
-a, --append	Append data to output file if available

-d, --debug-log	Set filename for debug info
/?, --help	Display this help and exit

The examples of usages, which correspond to the above listed working modes, are as follows:

(1) MzGenerator -m roi -i image.tiff -r masks.tiff -f featurenames.txt -o result.csv
(2) MzGenerator -m local -i image.tiff -r dots.tiff -f featurenames.txt -o result.csv
(3) MzGenerator -m local -i image.tiff -f featurenames.txt -o result.tiff

Explanation of other options:

-q, --query is used by the MaZda module to query for available computation algorithms. When *-q* is combined with the *-m* switch, MzGenerator prints a tree of available image processing and feature extraction algorithms.

-s, --step <int> defines an integer value *n* used for maps computation (3). To speed up computation the feature values are computed only for every *n*-th line and every *n*-th row of an image. The other feature values in the map are interpolated.

-x, --save-hex the feature values in CSV file are hexadecimally-coded to preserve accuracy of double precision numbers.

-a, --append this is to append new feature vectors to already created CSV file. The feature names list is acquired from the first line of the output CSV file. Therefore, the *-f* switch should not be used together with the *-a* switch.

Scripting

To automate feature extraction for multiple images and masks, the user can write a script (Unix/Linux) or a batch file (Dos/Windows). The below examples show how to extract features from image files stored in the current directory. The images are stored in TIFF format, recognized by a *.tiff* extension, and are accompanied by mask files of the same names but with a *.roi* extensions.

Linux Bash

```
#!/bin/bash
MzGenerator –m roi –f ./features.txt –o ./output.csv
for i in *.tiff
do
  MzGenerator –m roi –i "$i" –a –r "${i%.tiff}.roi" –o ./output.csv
done
```

Windows batch

```
@echo off
SetLocal EnableDelayedExpansion
MzGenerator.exe –m roi –f features.txt –o output.csv
for %%i in (*.tiff) do (
    set a=%%i
    set b=!a:~0,-5!
    set c=!b!.roi
  MzGenerator.exe –m roi –i %%i –a –r !c! –o output.csv
)
```

or optional

```
@echo off
MzGenerator.exe –m roi –f features.txt –o output.csv
```

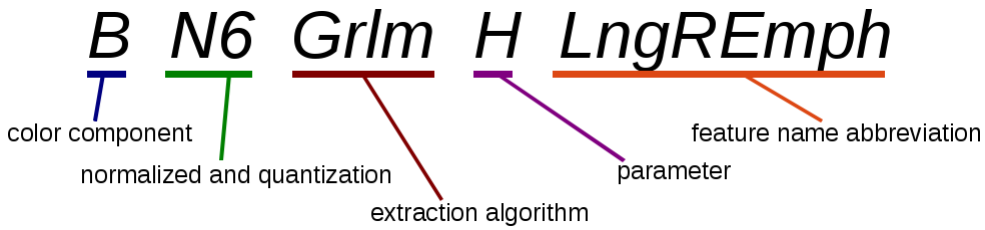
```
for %%i in (*.tiff) do (
  MzGenerator.exe -m roi -i %%i -a -r %%~ni.roi -o output.csv
)
```

Feature extraction algorithms

Feature naming

Feature values in qmazda result from several steps of image processing and feature extraction procedures. Moreover, the values can be computed at different parameter setups. Therefore, the multi-stage feature computation creates a problem of feature name uniqueness. Using traditional feature names such as mean, variance or entropy would be insufficient to identify particular characteristics. To prevent this situation, feature names are combinations of symbols that uniquely identify the way in which the value was generated. The symbols uniquely identify the complete image processing scheme, feature extraction algorithm and the algorithm parameters.

Feature name in *qmazda* is composed of components, called feature name stubs. They identify specific image preprocessing step, feature value computation algorithm or corresponding parameters. The stubs in feature name are arranged from the left to the right, coding consecutive computation steps.



For example, in region-based extraction, a name *BN6GrlmHLngREmph* indicates that the blue (*B*) color component's intensity was normalized (*N*) and then quantized with six (*6*) bits per pixel. The *Grlm* stub identifies the gray-level run-length matrix algorithm. The letter *H* identifies horizontal direction of runs and *LngREmph* is an abbreviation of *Long Run Emphasis* feature name.

Feature names for maps computation or for computed within specified neighborhoods, differ from feature names for region-based extraction. They consist of additional stub defining shape and size of the neighborhood or a sliding window. For example *c7* denotes the circular shape with radius equal to seven. The stub can fall on the right or on the left from the stub defining intensity normalization and quantization, e.g. *BN6c7GrlmHLngREmph* or *Bc7N6GrlmHLngREmph*. In the first case the normalization is performed within the entire image, in the second case the normalization is applied locally to the neighborhood or to the sliding window area.

Color components

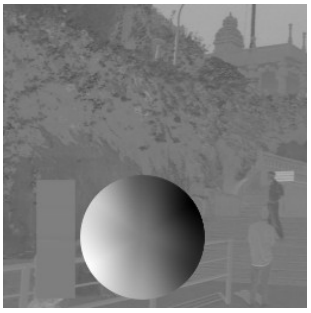
To apply feature extraction algorithms, which were designed for analysis of monochrome images, the color image is converted to monochrome color component images. There are different color models which define the components in different ways. *MzGenerator* applies conversion based on the following models: RGB, YIQ, YUV, HSB, CIE XYZ and CIE L*a*b*. The color components are identified by letter-codes and conversions are performed by means of the corresponding formulas.

Name	Letter-code	Color model	Conversion formula
Brightness	Y	YUV	$(299\,R + 587\,G + 114\,B) / 1000$
Red	R	RGB	R
Green	G	RGB	G
Blue	B	RGB	B
U-channel	U	YUV	$(886\,B - 587\,G - 299\,R + 886\,\Theta) / 1772$
V-channel	V	YUV	$(- 114\,B - 587\,G + 701\,R + 701\,\Theta) / 1402$
Hue (UV)	H	HSB	$\frac{\Theta}{2\pi} arg\left(\frac{886\,B - 587\,G - 299\,R}{886} + j\frac{-114\,B - 587\,G - 701\,R}{701}\right)$
Saturation	S	HSB	$0{,}937\sqrt{\left(\frac{886\,B - 587\,G - 299\,R}{886}\right)^2 + \left(\frac{-114\,B - 587\,G - 701\,R}{701}\right)^2}$
I-channel	I	YIQ	$(-3213\,B - 2744\,G + 5957\,R + 5958\,\Theta) / 11916$
Q-channel	Q	YIQ	$(-3111\,B - 5226\,G + 2115\,R + 5226\,\Theta) / 10452$
U normalized	u	YUV	$0.114\,\Theta\left(\left(\frac{(886\,B - 587\,G - 299\,R)}{(299\,R + 587\,G + 114\,B + 1)}\right) + 1\right)$
V normalized	v	YUV	$0.299\,\Theta\left(\left(\frac{(-114\,B - 587\,G + 701\,R)}{(299\,R + 587\,G + 114\,B + 1)}\right) + 1\right)$
I normalized	i	YIQ	$0.20786\,\Theta\left(\left(\frac{(-321.3\,B - 274.4\,G + 595.7\,R)}{(299\,R + 587\,G + 114\,B + 1)}\right) + 2.8185\right)$
Q normalized	q	YIQ	$0.26817\,\Theta\left(\left(\frac{(311.1\,B - 522.6\,G + 211.5\,R)}{(299\,R + 587\,G + 114\,B + 1)}\right) + 0.8903\right)$
Hue (IQ)	h	HSB	$\frac{\Theta}{2\pi} arg\left(-\frac{886\,B - 587\,G - 299\,R}{886} - j\frac{-114\,B - 587\,G - 701\,R}{701}\right)$

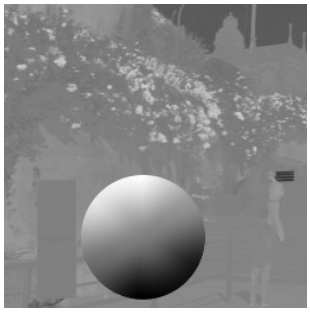
Parameter $\Theta = 2^{16} - 1$ represents maximum grey level.

Example image and its color components below.

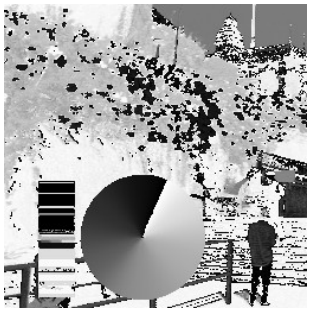
Y



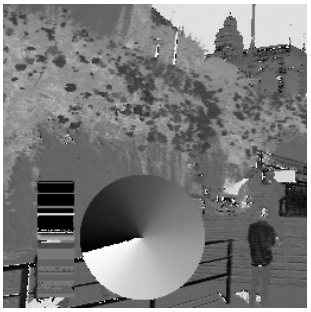
U



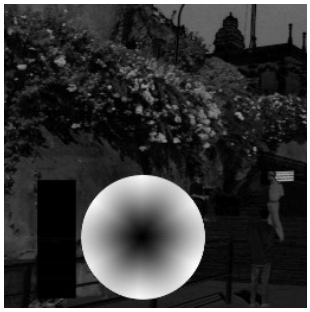
V



h



H



S

Normalization and quantization

Images obtained from different equipment or images acquired at different settings vary with brightness and contrast. This difference may affect feature values in unwanted way. To overcome this problem, *MzGenerator* implements image gray-level normalization procedures, which can be used optionally. The normalization is applied to the region of interest area for region based computation, or for all the image or the neighborhood area in local mode.

The particular procedure is identified by one of the following characters appearing in the feature name. The character is followed by *n* parameter defining number of bits used for gray-level coding.

D - the image is analyzed as is, no normalization is performed.

S - the maen value μ and standard deviation σ of grey-levels are computed. The range for further computation is $<\mu\text{-}3\sigma, \mu\text{-}3\sigma>$.

M - the minimum and a maximum grey-levels found in the region of interest define a new range.

N - the area grey-level histogram percentiles are computed. The new range is defined by first and ninety-ninth percentiles *<p1, p99>*.

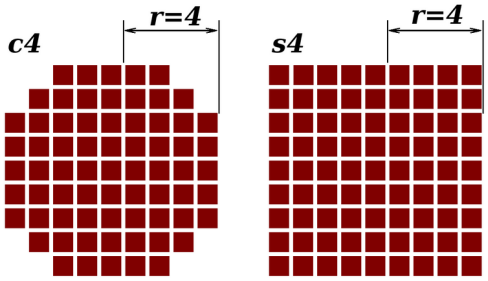
The normalized image I_{out} is computed pixelwise, according to the following formula:

$$I_{out} = 2^n \frac{(I_{in} - min)}{(max - min + 1)}$$

where I_{in} is an original monochrome image, *<min, max>* defines a new range and *n* defines number of bits per pixel.

Sliding window

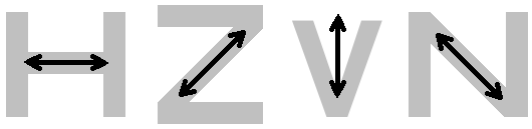
Names of feature maps or these of locally computed features consist of additional stub, a character and number, defining shape and size of the neighborhood. The character *c* indicates a neighborhood of a circular shape and the letter *s* a square. The following number *r* defines a radius (in pixels).



In case of 3-dimensional images the characters *c* and *s* define a ball-shaped neighborhood or a cube respectively. It must be noted, the *r* defines a radius in voxels. The voxel size (or spacing) does not affect the created neighborhood.

Directions coding

In algorithms such as Grey-level run-length matrix or Grey-level co-occurrence matrix, features are computed for specified directions or orientations. In the feature name, the directions are coded by single characters: *H* - 0º (horizontal), *Z* - 45º, *V* - 90º (vertical) and *N* - 135º. The below diagram can be useful to match the particular character with a direction.



In some algorithms (e.g. Gabor transform) intermediate angles are used, coded by characters: *U* - 22.5º, *S* - 67.5º, *G* - 112.5º and *J* - 157.5º.

In three-dimensional images the above mentioned directions apply to cross-sectional plane. In some algorithms the direction along the depth direction (perpendicular to the cross-sectional plane) is coded by *X*.

Character coding the orientation	The orientation vector
H	(1.000, 0.000, 0.000)
U	(0.924, 0.383, 0.000)
Z	(0.707, 0.707, 0.000)
S	(0.383, 0.924, 0.000)
V	(0.000, 1.000, 0.000)
G	(-0.383, 0.924, 0.000)
N	(-0.707, 0.707, 0.000)
J	(-0.924, 0.383, 0.000)
X	(0.000, 0.000, 1.000)

Histogram statistics

Image brightness histogram of the region of interest or the neighborhood is normalized (divided by the number of pixels). The feature name consist of the *Hist* stub to indicate the histogram-based feature extraction and the name of one of the following statistics.

$$Area = \sum_{(x,y)\in ROI} 1$$

$$Mean = \mu = \sum_{k=0}^{\Theta} k\, p(k)$$

$$Variance = \sum_{k=0}^{\Theta} (k-\mu)^2\, p(k)$$

$$Skewness = Variance^{-\frac{3}{2}} \left(\sum_{k=0}^{\Theta} (k-\mu)^3\, p(k) \right)$$

$$Kurtosis = Variance^{-2} \left(\sum_{k=0}^{\Theta} (k-\mu)^4\, p(k) \right) - 3$$

$$Perc\,01 = min(K) : \sum_{k=0}^{K} p(k) \geq 0{,}01$$

$$Perc\,10 = min(K) : \sum_{k=0}^{K} p(k) \geq 0{,}10$$

$$Perc\,50 = min(K) : \sum_{k=0}^{K} p(k) \geq 0{,}50$$

$$Perc\,90 = min(K) : \sum_{k=0}^{K} p(k) \geq 0{,}90$$

$$Perc\,99 = min(K) : \sum_{k=0}^{K} p(k) \geq 0{,}99$$

The $p(k)$ is the normalized histogram function.

$$p(k) = \frac{1}{Area} \sum_{(x,y)\in ROI} \begin{cases} 1 : I(x,y) = k \\ 0 : I(x,y) \neq k \end{cases}$$

Parameter $\Theta = 2^n - 1$ represents maximum grey level, n is a number of bits per pixel, $I(.)$ is an image, and (x, y) are pixel coordinates.

Gradient map features

The gradient magnitude map is computed within the region of interest (*ROI*) according to the following formula:

$$|G(x,y)| = \sqrt{(I(x,y+1) - I(x,y-1))^2 + (I(x+1,y) - I(x-1,y))^2}$$

where I is an image, and x, y are pixel coordinates, the horizontal and the vertical.

For normalization the *Area* is computed:

$$Area = \sum 1$$

The summation Σ includes all the pixels such that:

$$(x,y) \in ROI \wedge (x,y-1) \in ROI \wedge (x,y+1) \in ROI \wedge (x-1,y) \in ROI \wedge (x+1,y) \in ROI$$

The feature name consist of the *Grad* stub to indicate the gradient-map-based feature extraction and the name of one of the following statistics.

$$Mean = \frac{1}{Area} \sum |G(x,y)|$$

$$Variance = \frac{1}{Area} \sum (|G(x,y)| - Mean)^2$$

$$Skewness = Variance^{-\frac{3}{2}} \frac{1}{Area} \sum (|G(x,y)| - Mean)^3$$

$$Kurtosis = Variance^{-2} \frac{1}{Area} \sum (|G(x,y)| - Mean)^4 - 3$$

$$NonZeros = \frac{1}{Area} \sum \begin{cases} 1 : |G(x,y)| > 0 \\ 0 : |G(x,y)| \leq 0 \end{cases}$$

Grey-level run-length matrix features

The grey-level run-length matrix holds counts $p(k, l)$ of runs of pixels having the same grey level k and length l. The runs are established optionally in various directions. The feature name consist of the *Grlm* stub to indicate the grey-level run-length matrix feature extraction algorithm, then the character to indicate direction of runs: *H*, *V*, *Z*, *N* or *X*. The following features are computed based on the matrix.

$$Area = \sum_{k=0}^{\Theta} \sum_{\forall l} p(k,l)$$

$$ShrtREmph = \frac{1}{Area} \sum_{k=0}^{\Theta} \sum_{\forall l} \frac{p(k,l)}{l^2}$$

$$LongREmph = \frac{1}{Area} \sum_{k=0}^{\Theta} \sum_{\forall l} l^2 p(k,l)$$

$$GLevNonUni = \frac{1}{Area} \sum_{k=0}^{\Theta} \left(\sum_{\forall l} p(k,l) \right)^2$$

$$MGLevNonUni = \frac{1}{Area^2} \sum_{k=0}^{\Theta} \left(\sum_{\forall l} p(k,l) \right)^2$$

$$RLNonUni = \frac{1}{Area} \sum_{\forall l} \left(\sum_{k=0}^{\Theta} p(k,l) \right)^2$$

$$MRLNonUni = \frac{1}{Area^2} \sum_{\forall l} \left(\sum_{k=0}^{\Theta} p(k,l) \right)^2$$

$$Fraction = \frac{\sum_{k=0}^{\Theta} \sum_{\forall l} p(k,l)}{\sum_{k=0}^{\Theta} \sum_{\forall l} l\, p(k,l)}$$

Parameter $\Theta = 2^n - 1$ represents maximum grey level, where n is a number of bits per pixel.

Grey-level co-occurrence matrix features

The co-occurrence matrix holds counts of co-occurrences of pixels having some specified gray-levels. The pairs of pixels are considered, such that one of the pixels is situated at the offset (Δx, Δy) from the other one. The co-occurrence matrix (asymmetric) is defined as:

$$C_{\Delta x,\Delta y}(i,j)=\sum_{(x,y)\in ROI}\begin{cases}1 : I(x,y)=i \wedge I(x+\Delta x, y+\Delta x)=j\\ 0 : otherwise\end{cases}$$

Optionally the co-occurrence matrix (symmetric) is defined as:

$$C_{\Delta x,\Delta y}(i,j)=\sum_{(x,y)\in ROI}\begin{cases}1 : (I(x,y)=i \wedge I(x+\Delta x, y+\Delta x)=j) \vee (I(x+\Delta x, y+\Delta x)=i \wedge I(x,y)=j)\\ 0 : otherwise\end{cases}$$

The size of the matrix in both the cases is $2^n \times 2^n$ where n is a number of bits per pixel.

The normalized matrix, or probability of co-occurrence, is defined as:

$$p_{\Delta x,\Delta y}(k,l)=\frac{C_{\Delta x,\Delta y}(k,l)}{A}$$

where $A=\sum_{k=0}^{\Theta}\sum_{l=0}^{\Theta} C_{\Delta x,\Delta y}(k,l)$

The features are computed from the following formulas (we omit Δx, Δy for simplicity):

$$AngScMom=\sum_{k=0}^{\Theta}\sum_{l=0}^{\Theta} p^2(k,l)$$

$$Contrast=\sum_{m=1}^{2^{n+1}}\left(m^2 p_{\mathrm{dif}}(m)\right)$$

$$Correlat=\frac{1}{\rho_k \rho_l}\sum_{k=0}^{\Theta}\sum_{l=0}^{\Theta}\left((k+1)(l+1)p(k,l)-\mu_k\mu_l\right)$$

$$SumOfSqs=\sum_{k=1}^{N_g}\sum_{l=1}^{N_g}\left(k-\mu_k\right)^2 p(k,l)$$

$$InvDfMom=\sum_{k=0}^{\Theta}\sum_{l=0}^{\Theta}\frac{p(k,l)}{1+(k-l)^2}$$

$$SumAverg=\sum_{m=1}^{2^{n+1}}\left(m\, p_{\mathrm{sum}}(m)\right)$$

$$SumVarnc=\sum_{m=1}^{2^{n+1}}\left((m-SumAverg)^2 p_{\mathrm{sum}}(m)\right)$$

$$SumEntrp=-\sum_{m=1}^{2^{n+1}} p_{\mathrm{sum}}(m)\log\left(p_{\mathrm{sum}}(m)\right)$$

$$Entropy=-\sum_{k=0}^{\Theta}\sum_{l=0}^{\Theta} p(k,l)\log\left(p(k,l)\right)$$

$$DifVarnc=\sum_{m=1}^{\Theta}\left(i-\mu_{\mathrm{dif}}\right)^2 p_{\mathrm{dif}}(m)$$

$$DifEntrp=-\sum_{m=1}^{2^n} p_{\mathrm{dif}}(m)\log\left(p_{\mathrm{dif}}(m)\right)$$

where:

$$\Theta = 2^n - 1$$

$$\mu_k = \sum_{k=0}^{\Theta} \sum_{l=0}^{\Theta} k\, p(k,l)$$

$$\mu_l = \sum_{k=0}^{\Theta} \sum_{l=0}^{\Theta} l\, p(k,l)$$

$$\sigma_k = \sum_{k=0}^{\Theta} \sum_{l=0}^{\Theta} (k-\mu_k)^2\, p(k,l)$$

$$\sigma_l = \sum_{k=0}^{\Theta} \sum_{l=0}^{\Theta} (l-\mu_l)^2\, p(k,l)$$

$$p_{\text{sum}}(m) = \sum_{k=0}^{m-1} p(k, m-k)$$

$$p_{\text{dif}}(m) = \begin{cases} \sum_{k=0}^{\Theta-m} \left(p(k, m+k) + p(m+k, k) \right), & m \neq 0 \\ \sum_{k=0}^{\Theta-m} \left(p(k,k) \right), & m = 0 \end{cases}$$

$$\mu_{\text{dif}} = \sum_{m=0}^{2\Theta} m\, p_{\text{dif}}(m)$$

The feature name stub for grey-level co-occurrence matrix consists of *Glcm* (if features are derived from the symmetric matrix) or *Glch* (if features are derived from the asymmetric matrix), offset direction and distance, and the name given by the above feature extraction formulas. The direction is identified by one of the characters: *H*, *V*, *Z*, *N* or *X*. The distance is given by a number d = 1, 2, ...9.

Name stub	Offset (Δx, Δy)	Offset (Δx, Δy, Δz)
Hd	(*d*, 0)	(*d*, 0, 0)
Vd	(0, *d*)	(0, *d*, 0)
Zd	(*d*, *d*)	(*d*, *d*, 0)
Nd	(*d*, -*d*)	(*d*, -*d*, 0)
Xd	N/A	(0, 0, *d*)

Autoregressive model

Autoregressive model assumes dependence of pixel intensity on intensities of adjacent pixels. In *MzGenerator* the dependence is given by the formula:

$$I(x,y)=\theta_1(I(x-1,y)-\mu)+\theta_2(I(x,y-1)-\mu)+\theta_3(I(x-1,y-1)-\mu)+\theta_4(I(x+1,y-1)-\mu)+\mu+e(x,y)$$

where

$$\mu=\frac{\sum_{x,y\in ROI} I(x,y)}{\sum_{x,y\in ROI} 1}$$

and θ_1, θ_2, θ_3 and θ_4 are parameters of the model, and $e(.)$ is an error function. The parameters are computed to minimize mean squared error of $e(.)$ within the given region of interest (*ROI*).

Autoregressive model parameters are identified by the *Arm* stub, and names *Teta1*, *Teta2*, *Teta3*, *Teta4*. In addition a standard deviation of the error is computed identified by a *Sigma* name.

Gabor transform

The Gabor transform locally decomposes an image signal to its frequency components. The frequency components are computed by convolution with a complex number kernel, defined as:

$$g(x,y)=e^{\frac{-(x^2+y^2)}{2\sigma^2}}(\cos(\beta)+j\sin(\beta))$$

where

$$\beta=\omega(x\cos(\alpha)+y\sin(\alpha))$$

The parameters define frequency (ω), orientation (α) and standard deviation of the Gaussian envelope (σ). It is assumed the kernel size equals $6\sigma+1$ pixels in both horizontal and vertical directions, with center at $(x, y) = (0, 0)$.

The features computed by this algorithm are identified by *Gab* stub, size of the Gaussian envelope (σ), orientation (*H*, *U*, *Z*, *S*, *V*, *G*, *N*, *J* or *X*), period ($2\pi/\omega$) of the sinwave, and a name *Mag*.

The *MzGenerator* computes average magnitudes of Gabor transform within the morphologically eroded region of interest.

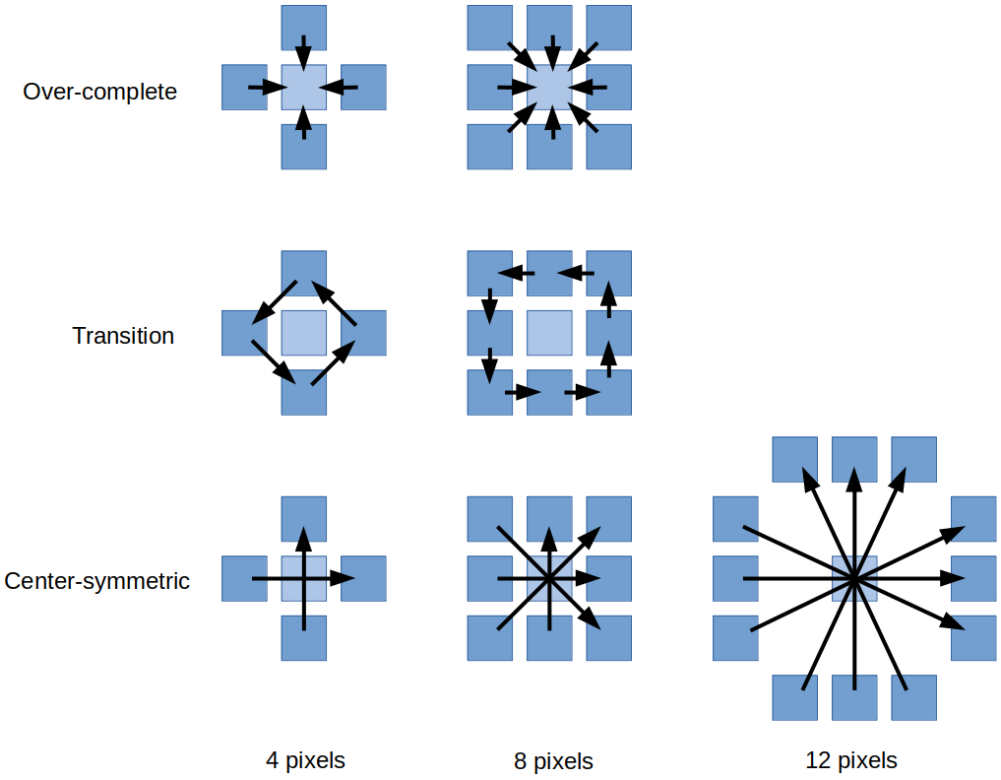
$$Mag=\frac{\sum_{(x,y)\in ROI'} g(x,y)}{\sum_{(x,y)\in ROI'} 1}$$

The ROI' indicates the region of interest morphologically eroded by means of circular structuring element of a radius equal to 3σ (the radius of the kernel). The erosion prevents image fragments located outside of the region of interest, to have any impact on the computation results.

Local binary patterns

Local binary patterns (LBPs) algorithm assess inequality relations in brightness between pixel in some neighborhood. In a specific pair of pixels in such the neighborhood, brightness of one of the pixels may be higher, equal or lower than the brightness of the other pixel. Situation when the value is higher is coded by one. Otherwise it is coded by zero. Several pairs in the neighborhood are examined to establish their codes, and the codes for a particular neighborhood are arranged together to form a binary pattern. Histogram of binary patterns computed for different neighborhoods within a specified region of interest becomes a descriptor of the region.

There are three algorithms for computation of local binary patterns (LBPs) implemented in *qmazda*, the over-complete, transition and center-symmetric. The algorithms differ with the arrangement of pixels in the neighborhoods and in for which pixels the inequality relations is established. The figure presents the example of 8-pixel neighborhood and arrows to indicate which pairs of pixels are used to establish the inequality relation. The *qmazda* implementation enables computation of LBPs in neighborhoods of three different sizes – of 4, 8 and 12 pixels. The below table presents schemes of all the available neighborhood sizes and algorithms.



The LBP features (the histogram values) are identified by the *Lbp* stub, followed by one of the three algorithm identifiers: *Oc* (over-complete), *Tr* (transition) or *Cs* (center-symmetric), and followed by the number of neighbors: *4n*, *8n* or *12n*.

Histogram of oriented gradients

Histogram of oriented gradients (HOG) counts occurrences of gradient orientations. The HOG features are identified by the *Hog* stub, followed by the number of angular bins: *4b*, *8b*, *16b* or *32b*.

Discrete wavelet transform

Qmazda enables computation of Haar discrete wavelet transform and the energies in its frequency sub-bands. The computation is performed excursively within the user-defined region of interest. The related features are identified by the *DwtHaar* stub followed by the sub-band identifier. The sub-band identifier defines a scale of the transformation: *S1*, *S2*, *S3* or *S4*, and the configuration of the low-pass and high-pass filters in vertical and horizontal directions: *HH*, *HL* or *LH*.

Morphological (shape) attributes

Morphological attributes are specific since they are not really extracted from the image. The features are computed exclusively from regions of interest masks, which are binary images. Such the regions of interest may be defined by the user. In *MaZda* application there are drawing tools which can be used for this purpose. Another possibility to create such the regions is to use programs for automatic image segmentation. *Qmazda* package do not support such the programs, however they can be found elsewhere or they can be created and tailored to a specific image processing problem.

All the attribute names related to the shape of image regions begin with *Mor*. There are three groups of morphological attributes computed by *MzGenerator* and *MzGengui*. The first group of features is computed by the algorithms implemented in the original *MaZda* project. Attribute names which belong to this group start with *MorMz* prefix. The other group of features is computed based on the algorithms implemented in *OpenCV* library. These attribute names begin with *MorCv* prefix. The third group are the selected algorithms implemented in *Insight Toolkit* library, and the feature names in this case begin with *MorItk*.

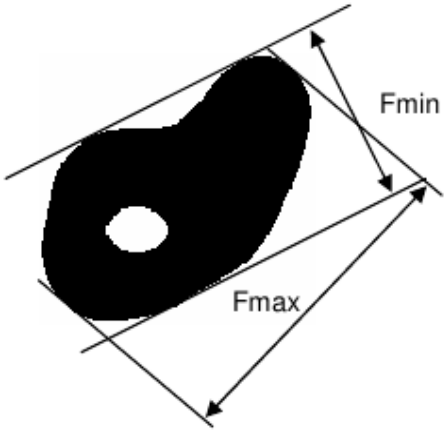
It must be noted that some basic features such as an area or a perimeter are computed by all the three approaches and thus may be multiplicated. Moreover, attributes which refer to the same property may slightly differ in value, since the algorithm implementations may use different approximation approaches. What is more, the *MorMz* and *MorCv* attributes are computed in pixel related image space and they do not care about the image resolution related to mm or inches. In contrast, implementations in *Insight Toolkit* library (with *MorItk* prefix) make use of image resolution information if it is available from the image file.

The list of attribute names computed by the custom-made implementation.

MorMzX horizontal coordinate of gravity center

MorMzY vertical coordinate of gravity center

MorMzF	area, number of the object pixels
MorMzSpol	diameter of the area equivalent circle
MorMzSmax	maximum diameter, distance between the most distant contour points
MorMzAox	orientation angle
MorMzUg	specific perimeter, sum of distances between the contour pixels
MorMzUw	convex perimeter



MorMzFmin mainimum Feret's diameter measured as distance between two parallel lines

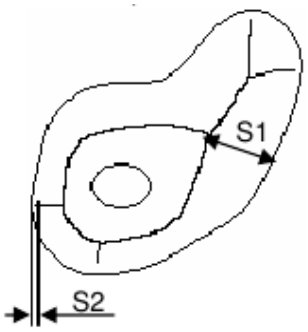
MorMzFmax maximum Feret's diameter measured as distance between two parallel lines

MorMzMmin minimum Martin's radius, a distance between gravity center and contour pixels

MorMzMmax maximum Martin's radius, a distance between gravity center and contour pixels

MorMzMaver Martin's average radius

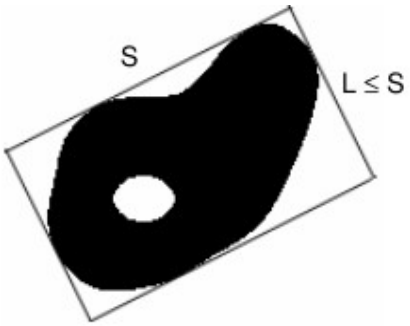
MorMzUl profile specific perimeter



MorMzS1 contour-skeleton maximal thickness

MorMzS2 contour-skeleton minimal distance

MorMzLsz skeleton length



MorMzS length of the circumscribing rectangle of minimal area

MorMzL width of the circumscribing rectangle

MorMzD1 diameter of profile inscribed circle of maximum area

MorMzD2 diameter of circumscribing circle

MorMzW1 ratio of radii (big to small) of circumscribing ellipsis of minimal area

MorMzW2 4π (profile area) / (profile perimeter)^2

MorMzW6	1 / W2
MorMzW3	(profile perimeter)^2 / (profile area)
MorMzW4	Ul / Uw
MorMzW5	F / Lsz
MorMzW7	D2 / D1
MorMzRs	perimeter^2 / 4π area
MorMzRf	ratio of horizontal to vertical Feret's diameters
MorMzRc	circularity
MorMzRm	Malinowska ratio
MorMzRb	Blair-Bliss ratio
MorMzRd	Danielsson ratio
MorMzRh	Haralic ratio
MorMzW8	L / S
MorMzW9	L S / F
MorMzM2x	horizontal second order moment of inertia
MorMzM2y	vertical second order moment of inertia
MorMzEr	average distance from gravity center
MorMzEl	average distance from contou
MorMzNc	number of contour pixels
MorMzNv	number of cavities
MorMzNl	number of profile contour pixels
MorMzNsz	number of skeleton pixels
MorMzNi	number of skeleton branches
MorMzNx	number of skeletal junctions
MorMzNo	number of skeletal loops

The list of attribute names computed by algorithms implemented in the OpenCV.

MorCvPixelsCount	area in number of pixels
MorCvX	
MorCvY	
MorCvProfileArea	the area limited by the outer contour
MorCvEquivDiameter	diameter of a circle of equivalent area
MorCvPerimeter	perimeter computed from the outer contour
MorCvRoundness	roundness
MorCvElipsAngle	angle of the fitted ellipse
MorCvElipsHeight	long diameter of the fitted ellipse
MorCvElipsWidth	short diameter of the fitted ellipse

MorCvElipsElong	elongation, the ratio of long to short diameter of the ellipse
MorCvMomTheta	properties computed from the moments of inertia
MorCvMomElong	
MorCvHuMom1	this one and the following are the Hu moments
MorCvHuMom2	
MorCvHuMom3	
MorCvHuMom4	
MorCvHuMom5	
MorCvHuMom6	
MorCvHuMom7	
MorCvChRectWidth	width of the described rectangle of the minimal area
MorCvChRectHeight	height of the described rectangle of the minimal area
MorCvChRectArea	area of the described rectangle of the minimal area
MorCvChArea	area of the convex hull
MorCvChPerim	perimeter of the convex hull
MorCvChMinFeret	
MorCvChMaxFeret	
MorCvRadDist	attributes based on the radii analysis
MorCvRadbb	

The list of attribute names computed by the class *LabelImageToShapeLabelMapFilter* implemented in the Insight Toolkit library. See the documentation of the class for explanation of the attributes.

MorItkArea

MorItkFullyConnectedFalse

MorItkFullyConnectedTrue

MorItkSizeX

MorItkSizeY

MorItkSizeZ (for 3D images only)

MorItkFeretDiameter

MorItkCentroidX

MorItkCentroidY

MorItkCentroidZ (for 3D images only)

MorItkPrincipalMomentX

MorItkPrincipalMomentY

MorItkPrincipalMomentZ (for 3D images only)

MorItkTilt orientation of principal axis

MorItkElongation

MorItkPerimeter

MorItkRoundness

MorItkEquivalentSphericalRadius

MorItkEquivalentSphericalPerimeter

MorItkEquivalentEllipsoidDiameterX

MorItkEquivalentEllipsoidDiameterY

MorItkEquivalentEllipsoidDiameterZ (for 3D images only)

MzReport

The *MzReport* is a program for data analysis, selection of most discriminative features, visualization of feature vector distributions, supervised machine learning and testing of the resulting classifiers. The input for the program are comma-separated vector files generated by the *MzGenerator* or *MzGengui* feature extractors.

Input data format

The input data for the *MzReport* are text files in comma separated vectors format. The first line of the text file is a header defining the feature names. The names in the first line are separated by commas or optionally by tabulators or semicolons. The last name in the header must be a word *Category*. The following lines are feature vectors. Each line contains as many feature values as were defined in the header. The last one is a category or a class name. The machine learning algorithms implemented in *MzReport* require that there are groups of vectors belonging to at least two different categories.

The below example presents a text file with 12 vectors containing four features (YD8HistMean, YD8HistVariance, YD5GlcmH1Contrast and YD5GlcmH1SumEntrp) belonging to three different classes or categories (blask, signora and bordo).

```
YD8HistMean,YD8HistVariance,YD5GlcmH1Contrast,YD5GlcmH1SumEntrp,Category
150.73,640.396,2.47158,1.34435,blask
150.247,990.643,2.86776,1.44421,blask
149.207,1004.67,3.22943,1.46546,blask
159.879,707.333,2.60532,1.36647,blask
145.947,806.024,2.05959,1.40508,signora
151.515,896.406,2.19412,1.37556,signora
151.345,776.586,2.52834,1.39291,signora
143.219,634.31,1.89288,1.32859,signora
145.951,742.408,2.4687,1.32371,bordo
142.599,1010.95,2.90421,1.46354,bordo
146.672,731.912,2.48124,1.37167,bordo
142.627,624.588,1.70794,1.32108,bordo
```

The data can be loaded after the user selects *File > Open report...* or by the drag-and-drop technique. When loading, the data is transposed in such a way that the feature vectors are presented in columns and feature values are arranged in the rows, and then displayed in a spreadsheet. The user may join reports from several files by loading or drag-and-dropping additional files. This will work as long as the feature names in the additional files are the same as the names of the first one. The data from the *MzReport* can be stored to a file by means of the *File > Save…* menu option.

Manual data selection

The user may manually select columns (vectors) or rows (features) to be selectively processed by the data analysis algorithms. For this reason, the rows and columns are accompanied by checkmark symbols. The user may check or uncheck an individual row or a column by double clicking on the symbol next to it. Only the data contained within the checked columns and the checked rows is used in the data processing procedures executed afterwards. The data contained by the checked columns and the checked rows may be saved to a separate file by means of the *File > Save selected…* menu option.

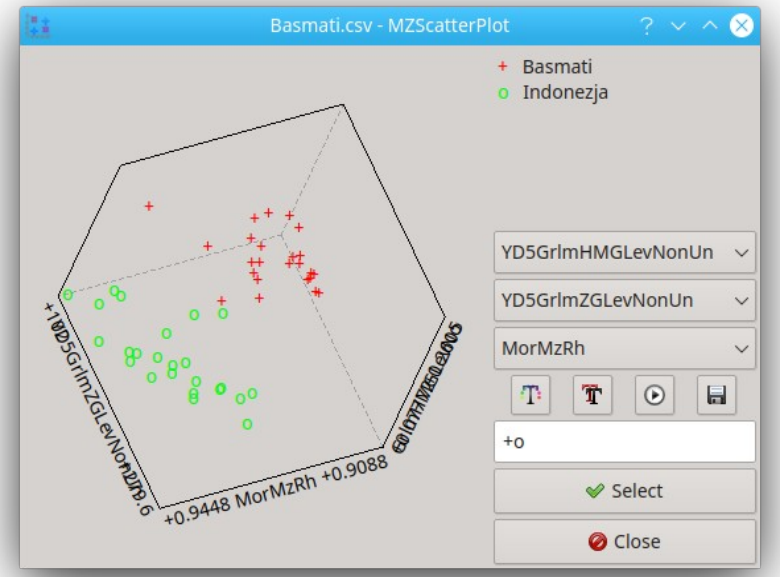
	Indonezja	Indonezja	Basmati	Basmati	Basmat	Bas
YD8GradArea	3432	2932	2441	1987	2183	2261
YD8GradMean	9.86739	8.07642	7.40824	6.30452	5.73554	8.7565
YD8GradVariance	151.389	139.326	87.7018	83.1539	68.8713	142.71
YD8GradSkewness	-3.71991	-3.79711	-3.73999	-3.89555	-3.83611	-3.336
YD8GradKurtosis	17.2055	16.9615	16.0821	18.3546	18.2484	12.736
YD8GradNonZeros	0.989802	nan	0.990578	0.980876	0.972515	0.982
YD8Gab4H2Mag	28.1788	22.2905	15.5239	26.2931	17.5846	25.342
YD8Gab4V2Mag	6.66758	6.33719	4.88022	6.08227	4.61604	5.568
YD8Gab4N2Mag	2.27368	2.56293	2.13921	1.89062	2.68058	3.0057

Manual checking or unchecking hundreds of rows or columns one by one would be time consuming. Therefore the program provides tools to enable checking or unchecking more vectors or features by a single click. These tools are available from *Edit > Vectors* and *Edit > Features* sub menus. The options to check or uncheck highlighted fields refer to the group of fields marked (highlighted) with a mouse. The tool to uncheck *nan* refers to columns or rows containing a *nan* value (not a number). The *nan* indicates that the value of the particular feature was not computed. Since further steps of analysis are performed excursively for the data belonging jointly to the checked columns and the checked rows, unchecking vectors containing the *nan* fields may be necessary to obtain correct results of such the analysis.

Vector distribution visualization

Distributions of the vectors may be inspected visually by choosing the *Analysis > Scatter plot*. This option opens a window with a three dimensional scatter plot. The plot is presented on the left hand side of the window. It can be rotated by pressing the left mouse button and moving the mouse cursor over the plot. The plot resizes along with the window.

Vectors belonging to different categories are presented with various symbols and colors. The symbols and the names of categories are listed at the top-right corner of the window. Below, there are three drop boxes listing feature names. The vectors are presented within the three dimensional space of the features selected by means of these drop boxes. The following are the four buttons, which enable to change colors and typeface of the vector symbols, animate the plot, or save the plot to an image file. Below the buttons there is an editable text field to define character symbols used by the plot.



The window can be closed by pressing the *Select* or the *Close* button. In case of pressing the *Select* button, all the feature names in the main window are unchecked except the three selected in the drop boxes. Pressing the *Close* button does not modify any selections in the main window.

Machine learning

Machine learning algorithms are implemented as separate shared or dynamic linking libraries. Their functionality is available through sub menus of Analysis menu option of *MzReport* program. The algorithms enable selection of most discriminative features, which is a subset of features most suitable for data classification purposes. The simplest are feature filtration methods (*Analysis > Feature filtration*), which examine discrimination ability of individual features one by one. Afterwards, a rank of features having the highest abilities is presented. The user can accept the features. This results in unchecking all the other feature names from the vertical headers in the main window. Currently there are two optional criteria implemented, which quantitatively asses discrimination ability of individual features – these are the Fisher's discriminant and the mutual information.

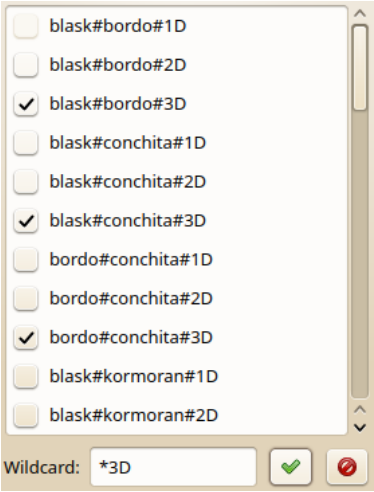
In many cases the discrimination power of individual feature may reveal only when it is accompanied by some other attributes. Therefore, in an alternative wrapper approach, the discrimination ability of several features is evaluated jointly. The discriminative power of such the feature subset can be established in relation to the performance of some classification algorithm. Therefore, in wrapper strategy, attributes are chosen to maximize proficiency of a particular classifier. In the implemented algorithm the wrapper selection starts with examination of individual features, which is equivalent to the filtration approach. Then, the features are ranked according to their discrimination power. Next the pairs of features, or two-dimensional feature subspaces, are examined. The features are combined in pairs starting with the features gained the highest ranked in the previous step. Again, features are ranked according to their best performance in pairs. This procedure continues by increasing the number of jointly examined attributes. The user can limit the maximum number of jointly examined features (subspace dimensionality) as well as the maximum time intended for search.

If the number of selected features is still unacceptably large, it is possible to perform further dimensionality reduction by projection of the original feature space into a new space of lower dimensionality. This yields a less numerous collection of new features having discrimination ability almost as good as the original set of features.

The next step in machine learning is classifier training. The objective of this stage is to gain decision rules to categorize data vectors. The rules may be imagined as boundaries in the feature space – decision boundaries. If such a boundary is a line, a plane or a hyperplane, the classifier is called to be linear. In this case feature vectors appearing on one side of the boundary falls into the one class and the vectors located on the other side falls into the other class. If the shape of the boundary is more complex, the classifier is called nonlinear.

The below table summarizes currently implemented plugins with respect to their abilities in feature selection, feature space projection, classifier learning and classifier testing.

	Feature filtration	**Linear discriminant analysis**	**Convex hull discrimination**	**Support vector machines**
Feature filtration	+	+	+	
Wrapper approach		+	+	
Space projection		+		
Supervised training		+	+	+
Testing		**confusion matrix**	**confusion matrix**	**confusion matrix**
Decision boundary		**hyperplane**	**convex hull**	**optional kernels**
Multiple classes		**classifier ensembles**	**one from the others**	+



Plugins which support training also enable testing. To test the classifier the user have to select *Test classifier…* option from the plugin's submenu. If there are more than one classifier available, the user is provided with the dialog window to select classifier for testing. This usually takes place after feature selection procedure which examines feature sub spaces of various dimensionalities. It is usually reasonable to select only the classifier using the highest number of features (dimensions). Specifically in linear discriminant analysis plugin, the classifier ensembles are used. Therefore in this case all the classifiers with the highest number of dimensions should be selected. It must be noted that if the user selects several classifiers and they make conflicting decisions, the final membership is indeterminate.

As the result of testing procedure the user is presented with the confusion matrix. The rows of the matrix represent actual classes of vectors membership, and the columns represent classes predicted by the classifier. If the actual and predicted classes are of the same names and they are arranged in the same order, then the elements (highlighted with green) located on the diagonal count correct classifications, and the other fields of the matrix count classification errors. Indeterminate decisions are counted in the last column of the matrix (highlighted with yellow).

	blask	bordo	conchita	kormoran	mercada	signora	!
blask	1762	112	226	126	109	169	0
bordo	154	1930	86	122	211	120	0
conchita	228	171	1402	366	181	268	0
kormoran	198	145	338	1402	258	271	0
mercada	130	263	113	196	1727	167	0
signora	215	193	264	256	219	1470	0

Cross-validation

The classification rules should be validated by means of data not used in the training procedure. One of the methods used in machine learning to separate training and test data sets is a k-fold cross-validation, which randomly splits set of all the vectors into a fixed number of subsets – folds. Some number of these folds are used for training whilst the others are used for testing. The procedure of training and testing can be repeated for various combination of the folds. *MzReport* supports the k-fold cross-validation procedure, however it does not automatize it.

To randomly split vectors into several folds one needs to select *Edit* > *Vectors* > *Folds* option from the program's menu. Then, in the *Folds* dialog window the folds number have to be selected. The data subsets are generated when the Random allocation button have been pressed. The list of folds with the check boxes enable selection of active folds. Now, the user needs to Apply selection of folds and close the dialog. In the report window, all the vectors (columns) belonging to selected folds are checked and the others are unchecked. The selected (checked) feature vectors can be now used for feature selection and classifier training. Next, the user can select *Edit* > *Vectors* > *Invert selection* and perform classifier testing with the remaining vectors. The procedure for the classifier training and testing may be repeated for various combination of folds selected in the *Folds* dialog.

Classifiers file format

The classifiers which result from the machine learning procedures can be stored as text files by choosing *Save classifier…* option from the plugin's submenu. The text file specifies a type and number of classifiers, name of each of the classifiers, names of classes recognized by an individual classifier, names of features (attributes) required by the classifier, and numerical parameters which specify decision boundaries. The below example presents a file of three (*@Classifiers 3*) linear (*MzLinearClassifiers2013*) classifiers. The specification of every individual classifier starts with the

word *@ClassNames* followed by the number of classes recognized by the classifier and names of the classes (*Norm* and *Spoiled*). The following lines start with the word @FeatureNames, which in similar way define number of features required by the classifier and names of these features (eg. B_YS5GlcmN5Entropy A_ID8HistPerc10). To identify particular classifier *MzReport* assignes a name to it by concatenation of the class names recognized by the classifier and the number of features it uses. The names are separated by a hash character and the number of features (feature space dimensionality) is followed by the letter D (eg. *Norm#Spoiled#3D*).

```
MzLinearClassifiers2013
@Classifiers 3
@ClassNames 2 Norm Spoiled
@FeatureNames 1 B_YS5GlcmH1SumEntrp
@Values 2
1 -25.7829
2 -36.8988 2.04709
@ClassNames 2 Norm Spoiled
@FeatureNames 2 B_YS5GlcmN5Entropy A_ID8HistPerc10
@Values 2
2 -9.41758 0.266528
2 17.2005 2.72964
@ClassNames 2 Norm Spoiled
@FeatureNames 3 A_ID8HistMean B_YS5GlcmV1Entropy A_QD8HistVariance
@Values 2
3 0.20229 -10.6935 -0.20179
2 12.9349 3.15684
```

The support vector machines plugin was build upon the LibSvm library. Unlike the other plugins, it is able to store only a single classifier in a single file. The file starts with the *SvmMazdaClassifiers2017* identifier, which is followed by the information on number of classifiers (*@Classifiers* – always equal to one), class names (*@ClassNames*), feature names (*@FeatureNames*) and information on data normalization (*@Values*). A specification of the actual classifier is provided after the word *@LibSvmClassifier* and it is compatible with the format established by the *LibSvm* library.

```
SvmMazdaClassifiers2017
@Classifiers 1
@ClassNames 3 Norm Broken Infected
@FeatureNames 4 A_YS5GlcmV1AngScMom A_HD8HistPerc01 B_YD5GrlmHRLNonUni
B_YD5GlcmN1Contrast
@Values 2
4 97.813 0.612372 0.000617165 16.6423
4 -0.0479689 -93 -4123.03 -0.376612

@LibSvmClassifier
svm_type c_svc
kernel_type rbf
gamma 0.008
...
```

Libraries used

MzReport plugins use the following free and open source libraries:

Alglib – linear discriminant analysis (http://www.alglib.net/)

Qhull – convex hull decision boundary (http://www.qhull.org/)

LibSvm – support vector machines (https://www.csie.ntu.edu.tw/~cjlin/libsvm/)

MzTrainer

MzTrainer is a console tool for selecting the most discriminating attributes and for machine learning. It takes csv file with vectors of attributes as an input and produces a text file with classification rules based on the most discriminating attributes. The program can use one of the plugins designed for the *MzReport* program. Thus, it has the same functionality of attribute selection and machine learning as the *MzReport* application. Therefore, to run data analysis the user has to provide information on which plugin (machine learning algorithm) shall be used and also to provide a setup (a list of parameters) for the plugin.

The example command line for the application in Linux may look like this:

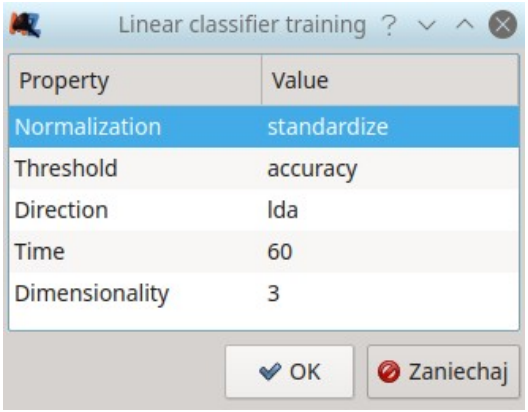
```
./MzTrainer -p ./libLdaPlugin.so -c config.txt -i in.csv -o out.txt
```

Observe that in Windows the plugin library files have *.dll* extension instead of *.so* extension which is used in Linux.

```
MzTrainer.exe -p libLdaPlugin.dll -c config.txt -i in.csv -o out.txt
```

The resulting *out.txt* file consist of classification (prediction) rules and may be used as an input classifier (predictor) for *MzPredict*, *MzReport* or *MzMaps* programs.

The configuration file should be edited by the user prior to calling *MzTrainer*. It should contain text lines with names of parameters (properties) and their values. The simple way to produce such the file is to run the *MzReport* program first and observe the *Options* window of the selected algorithm. When editing the configuration file for *MzTrainer*, one can follow the list of the algorithm options from *MzReport*. The following illustration present example options for training of the linear classifier, both in the window available in *MzReport* and in the text file created for *MzTrainer*.




```
Normalization standardize
Threshold accuracy
Direction lda
Time 60
Dimensionality 3
```

MzPredict

MzPredict is a console tool for data classification, or with other words, prediction of category. The input for the program are a classifier file and a file with vectors in the comma separated values (CSV) format. For more information on the classifier file format and the CSV file format see the MzReport section.

The output are text lines, each line indicating a number and a name of predicted category. The number and the order of lines in the output match the number and the order of vectors in the CSV file.

Examples of use

Let us assume that we have a classifier file named *predictor.txt* prepared by means of the MzReport tool. The content of the file may look like this one:

```
MzLinearClassifiers2013
@Classifiers 1
@ClassNames 2 Bad Good
@FeatureNames 3 height YD8HogO16b7 YD5GlcmN5Area
@Values 2
3 -0.00730388 -0.729595 9.52738e-06
2 -2.09744 29.2809
```

This particualr classifier recognizes two categories: *Bad* and *Good*, and requires three attributes to do this: *height*, *YD8HogO16b7* and *YD5GlcmN5Area*.

Let as assume that the input file name is *input.csv*. It may look like this:

```
height,YD8HogO16b7,YD5GlcmN5Area,vD8HistPerc01,YD8ArmTeta1,Category
301,0.350765,87638,80,0.905263,Broken
248,0.720922,71026,80,0.887093,Broken
259,0.508249,68096,78,0.912827,Broken
239,0.502737,69102,78,0.908557,Broken
234,0.522566,72552,76,0.943935,Broken
468,0.666904,139866,86,0.83612,Fine
530,0.698431,168362,84,0.822966,Fine
500,0.851011,159968,84,0.84404,Fine
535,0.729207,181710,85,0.834936,Fine
448,0.743843,137736,84,0.844527,Fine
```

It consists of several lines. The first line is a header which lists the attribute names. The other lines define vectors of attributes and the corresponding category names. The column with the category name is non obligatory. What is important, the CSV file has to deliver all the attributes which are required by the classifier.

We call the *MzPredict* program in the following way:

```
MzPredict –c predictor.txt –i input.csv –o output.txt
```

or optionally, using standard input and output streams:

```
MzPredict –c predictor.txt < input.csv > output.txt
```

The program saves the classification results into the output file. Every line of the output file consists of the predicted category index and the category name, and corresponds to the feature vectors stored in the input CSV file:

```
1 Bad
1 Bad
1 Bad
1 Bad
1 Bad
2 Good
2 Good
2 Good
2 Good
2 Good
```

Optional *-v* or *-verbose* switch make the *MzPredict* to print additional information. It provides some information on the classifier, the ground-truth categories and the attribute names. Moreover, if we do not provide the input or the output file names, the program will use standard input or standard output streams instead.

If we call the *MzPredict* program in the following way:

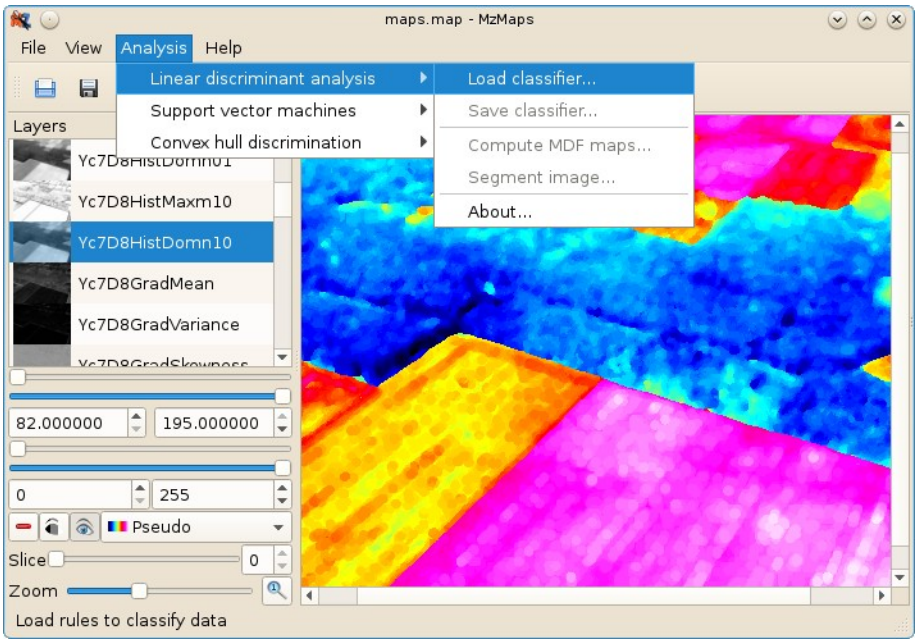
```
MzPredict –v –c predictor.txt –i input.csv
```

The output will be printed to the console and it will contain the additional information:

```
Classifier: MzLinearClassifiers2013
Classes:
  0 !
  1 Bad
  2 Good
Features required:
  height
  YD8HogO16b7
  YD5GlcmN5Area
Features in csv:
  height
  YD8HogO16b7
  YD5GlcmN5Area
  vD8HistPerc01
  YD8ArmTeta1
1 Bad : Broken
1 Bad : Broken
1 Bad : Broken
1 Bad : Broken
1 Bad : Broken
2 Good : Fine
2 Good : Fine
2 Good : Fine
2 Good : Fine
2 Good : Fine
```

MzMaps

MzMaps is a tool for visualization of feature maps and for image segmentation. The input for the program are multi page, floating point data tiff files produced by the *MzGenerator* or *MzGengui* feature extractors. Moreover, the *MzMaps* module accepts classifiers from *MzReport*, which serve as rules for image segmentation.



Maps can be loaded by means of *File* > *Load feature maps* option or by drag and drop technique. When maps have been loaded, the list of all the maps is presented in the *Layers* docking window. User can click on the particular map on the list, which will display the map in the main window. Moreover, the currently selected map can be removed from the list by pressing the button.

The *eye* buttons enable selection of visualization modes. The eye button enables visualization in gray-scale range. The three-level mode is used to choose grey-level thresholds. The grey-levels or thresholds can be modified by means of the four sliders located below the list of the maps. Moreover, the currently selected map can be viewed in pseudo-palette mode, which can be selected from the combo box. The Slice slider enables to select particular cross-section if the 3D map is presented. If the map is 2D the slider is disabled. The image can be zoomed in and out by means of the Zoom slider located at the bottom of the docking window.

User can read the exact value of the feature by moving the mouse button to desired location on the map and then clicking the left mouse button. The value is presented on the status bar at the bottom of the *MzMaps* window.

The main function of the *MzMaps* is to test ability of classifiers created in the *MzReport* to perform

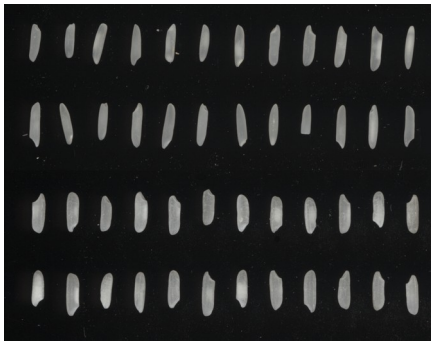
image segmentation. The *MzMaps* use the classifier plugins which are available through the *Analysis* menu option. Particular classifier can be loaded from a file by means of *Load classifier* menu options or by drag and dropping the classifier file to the main window of the *MzMaps*. To perform the segmentation, user has to select a *Segment image* option associated with the particular classification plugin. The results of the segmentation are presented in a new window.

It must be noted that the features required by the classifier must be present on the list feature maps. Otherwise the classification result may be incorrect or the classifier will not work.

Tutorial: Region-based analysis

This tutorial explains:

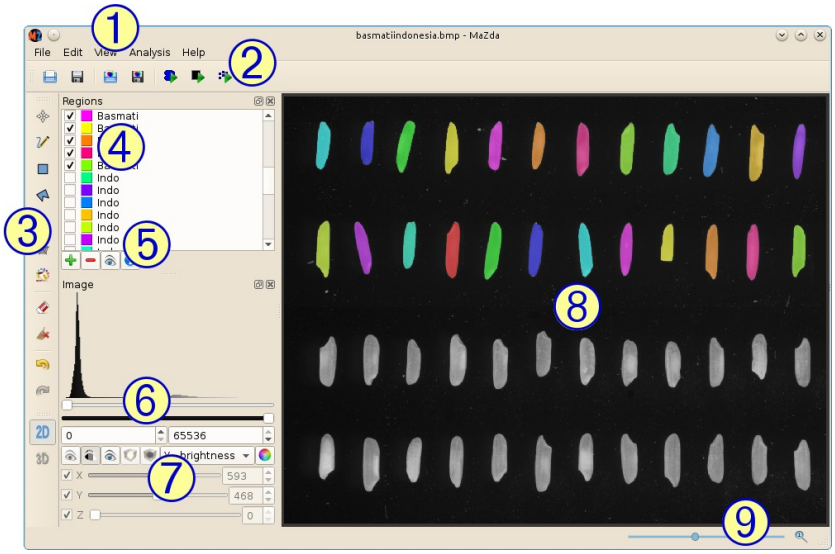
- ✔ How to create regions of interest
- ✔ How to compute feature values
- ✔ How to perform discriminant analysis
- ✔ How to classify new data



In this tutorial we use image showing two varieties of rice. The image was acquired by a flatbed scanner. The upper two rows of the image show Basmati and the lower two rows show Indonesia varieties. The goal of the analysis is to find a way to recognize the variety of individual grains.

How to create regions of interest

Start *MaZda* module. Usually you need to click or double-click on the program's icon. Load an image for analysis. You can use either a menu option *File* > *Load image...* or drag-and-drop the image file to the image area of *MaZda* window.



① main menu
② load, save and analysis tools
③ region drawing tools
④ region of interest list
⑤ region of interest tools
⑥ grey-levels sliders
⑦ image viewing switches
⑧ image area
⑨ zoom slider

The image is shown in grey-scale, with no color overlays. There is only one *New* item present on the region of interest list.

The user can draw the regions manually by means of drawing tools ③. However, in this tutorial we will create regions by grey-level thresholding.

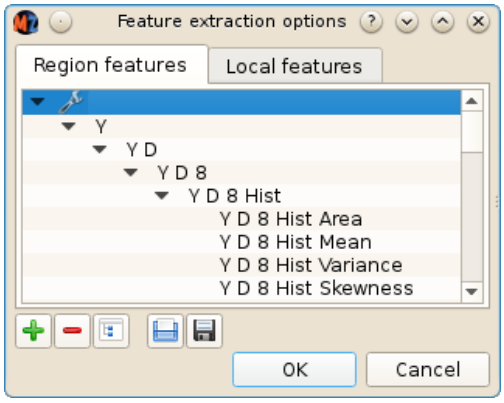
Rename the default regions name. Double-click on the regions name *New* and enter the name *Basmati*, which is a name of the first variety (class). Switch to the three-levels view mode ⑦ and adjust sliders ⑥ to see the image background in black and grains in grey.

Select option *Edit* > *Threshold*. A color overlay covering all the grains should appear on the image. Select *Edit* > *Morphology* > *Median* to remove small areas and dust from the region of interest (color overlay) area. Select *Edit* >*Morphology* to create individual regions for individual grains. The regions should appear with various colors covering individual grains. Now, click on the first region on the list ④. This region still covers all the grains together and is no longer required. Remove it from the list by clicking the button. Verify if all the grains are covered by the color overlays. If you notice small overlays of dust you can remove them from the list manually, or automatically with *Edit* > *Remove small* option.

The check-boxes in the region of interest list ④ enable to hide or show particular color overlays. If the two or more overlays have similar color, clicking on the check-boxes can be useful to verify which color overlay corresponds to a particular item on the list.

One by one, double-click on the names ④ which correspond to grains of Indonesia variety, and change them to *Indonesia*. Now, every name on the list should match the variety of the grain covered by the corresponding region.

How to compute feature values

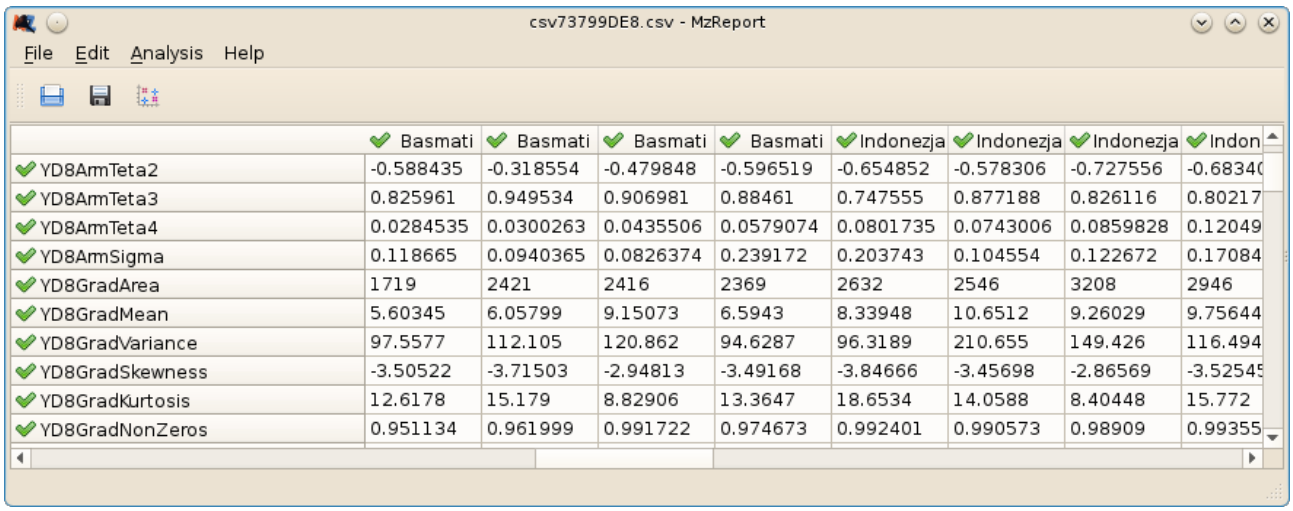


Open the feature extraction options dialog with *Analysis* > *Options...* and select the *Region features* tab-page. Verify if the list of features to be computed is present. Otherwise, load the feature list from a file or create the list manually. Accept the list of features by pressing *OK* button.

In MaZda window select *Analysis* > *Roi based computation* to start feature extraction. Optionally press the button of the *Load, save and analysis* ② toolbox. The progress of feature values computation may be observed in *Mazda generator* window. After successful computation, *MzReport* is started and the results are loaded into this module automatically.

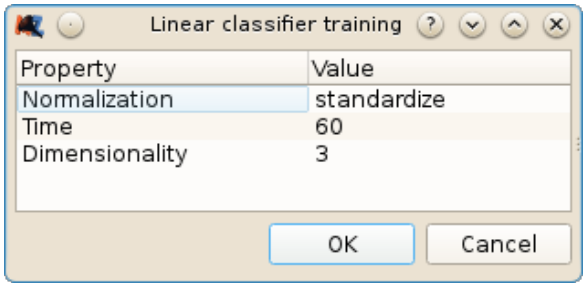
How to perform discriminant analysis

The results are presented as a spread-sheet, where each column represents feature vector of individual grain, and every row represents a value of particular feature. Make sure that all the columns and all the rows are checked to be used for the following analysis. Also, verify if the column headers show the names of the two varietal classes. Save the results to a file (*File* > *Save...*)



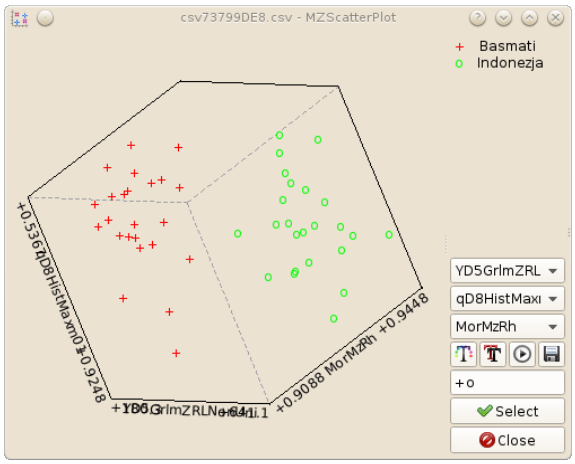
	Basmati	Basmati	Basmati	Basmati	Indonezja	Indonezja	Indonezja	Indon
YD8ArmTeta2	-0.588435	-0.318554	-0.479848	-0.596519	-0.654852	-0.578306	-0.727556	-0.6834(
YD8ArmTeta3	0.825961	0.949534	0.906981	0.88461	0.747555	0.877188	0.826116	0.80217
YD8ArmTeta4	0.0284535	0.0300263	0.0435506	0.0579074	0.0801735	0.0743006	0.0859828	0.12049
YD8ArmSigma	0.118665	0.0940365	0.0826374	0.239172	0.203743	0.104554	0.122672	0.17084
YD8GradArea	1719	2421	2416	2369	2632	2546	3208	2946
YD8GradMean	5.60345	6.05799	9.15073	6.5943	8.33948	10.6512	9.26029	9.75644
YD8GradVariance	97.5577	112.105	120.862	94.6287	96.3189	210.655	149.426	116.494
YD8GradSkewness	-3.50522	-3.71503	-2.94813	-3.49168	-3.84666	-3.45698	-2.86569	-3.52545
YD8GradKurtosis	12.6178	15.179	8.82906	13.3647	18.6534	14.0588	8.40448	15.772
YD8GradNonZeros	0.951134	0.961999	0.991722	0.974673	0.992401	0.990573	0.98909	0.99355

To perform linear discriminant analysis select *Analysis* > *Linear discriminant analysis* > *Selection and training...* from the *MzReport* menu. The analysis options dialog will pop-up. It defines three parameters, *normalization*, *time* and *dimensionality*.



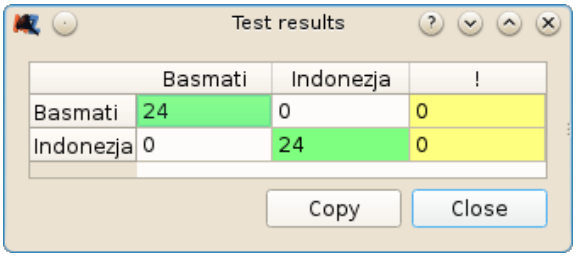
The following analysis will search for feature subsets up to three features (*dimensionality = 3*) in the subset. It will normalize the feature value distributions with respect to their means and standard deviations (*normalization = standardize*). The search will be limited in time to 60 seconds (*time = 60*). Now, press the *OK* button to start feature selection and machine learning procedure.

After the analysis succeeds the dialog with the most discriminative features appears. Press *OK* button to accept. Now, most of the feature names in *MzReport* are unchecked. The only checked feature names are the features found by the selection procedure.



To visually inspect distributions of vectors in the selected features subspace select *Analysis* > *Scatter plot...* . You may change an orientation of the plot by dragging it with the mouse. The + symbols indicate feature vectors of Basmati class and the o symbols indicate vectors of the Indonesia variety. Verify if the vectors belonging to different classes form separate clusters, making it possible to separate the clusters with a plane. Press *Close* to close the dialog.

The procedure also creates rules for classification. To save the rules select *Analysis* > *Linear discriminant analysis* > *Save classifier...* . The classification rules (classifiers) are saved to the text file for further use.



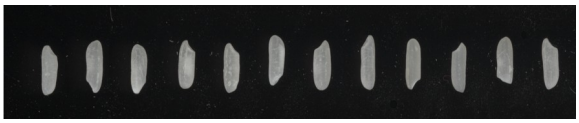
To check the resulting classifiers select *Analysis* > *Linear discriminant analysis* > *Test classifier*... In *Select classifiers* dialog, check the classifier (*Basmati#Indonesia#3D*) which makes use of all the three selected features, uncheck all the others and press *OK*. The test produces confusion matrix to indicate number of correctly and incorrectly recognized grains.

Note that the proper validation of any classifier requires that the test is performed on data not used for the training. The above described classifier's check, makes the rough and quick verification, and does not fulfill this requirement.

How to classify new data

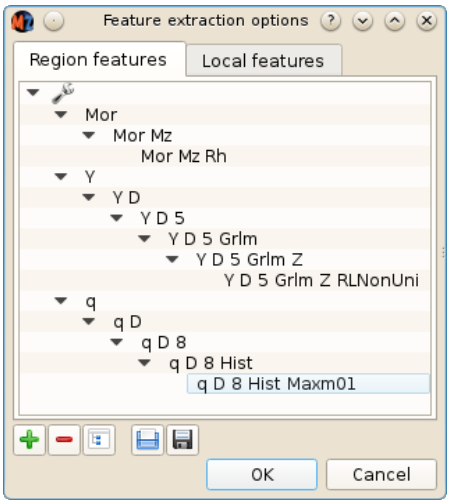
This example explains:

- ✔ How to use classifiers



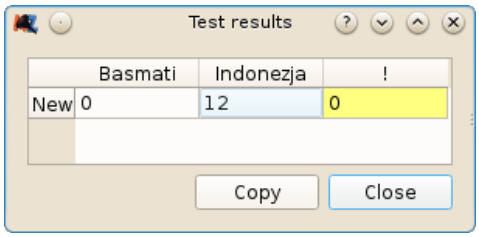
In this tutorial we use image showing rice grains of unknown variety, Basmati or Indonesia.

In MaZda load the image and create regions of interest as explained in the previous tutorial.



Open the feature extraction options dialog with *Analysis > Options...* and select the *Region features* tab-page. Select main root of the feature tree. Press to remove all the feature names. Press button to load feature names from the text file with classification rules created in the previous tutorial. Now the list of feature names should consist of the three features previously selected as the most discriminative. Press *OK* button to accept. Next, press the button, or select *Analysis > Roi based computation*, to start feature extraction.

In the *MzReport* select *Analysis > Linear discriminant analysis > Load classifier...* to load the classification rules created in the previous tutorial. Next, select *Analysis > Linear discriminant analysis > Test classifier...* . In *Select classifiers* dialog, check the *Basmati#Indonesia#3D* classifier, uncheck all the others and press *OK*.

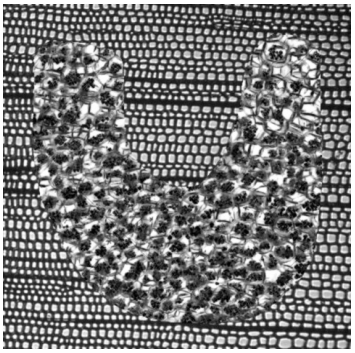


The confusion matrix shows that all the *New* grains were classified as *Indonesia* variety.

Tutorial: Image texture segmentation

This tutorial explains:

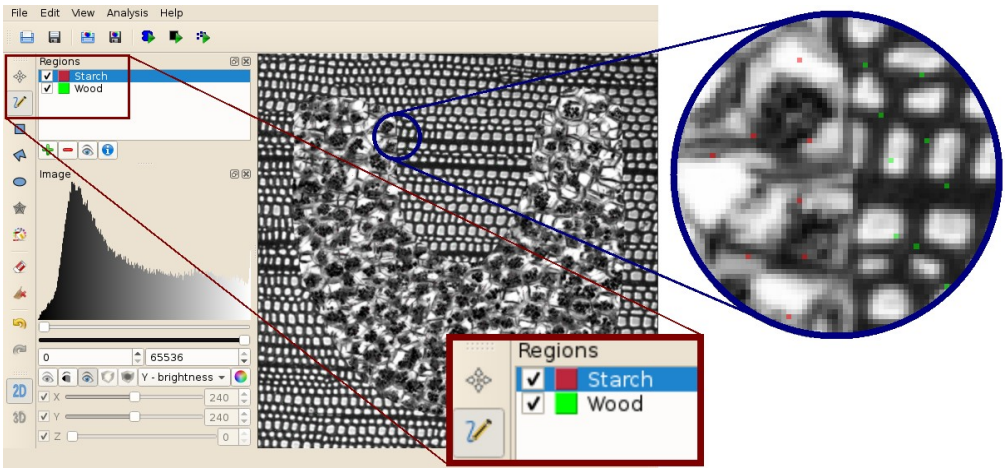
- ✔ How to compute local feature values
- ✔ How to perform discriminant analysis
- ✔ How to segment an image



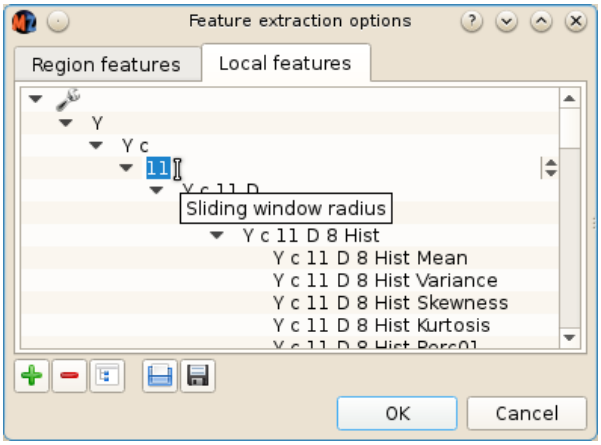
In this tutorial we use a patchwork of two natural textures. The image was artificially created for the need of this tutorial. However, similar problems of texture segmentation can be found also in natural images.

How to compute local feature values

Start *MaZda* module and load the image for analysis. Rename the default regions name to Starch. To do so, double-click on the regions name *New* and enter the name *Starch*. Add another region by pressing the *Add region* button in the *Regions* panel. Change the name of the new region to *Wood*.



Now, select the *Starch* region from the list. Press the *Pencil* tool from the drawing tools toolbox. Mark several dozen dots over the U-shaped starch region. The dots should be placed evenly over the whole area of the region. Select the *Wood* region from the list and mark several dozen dots over the background area of the wood cells texture.

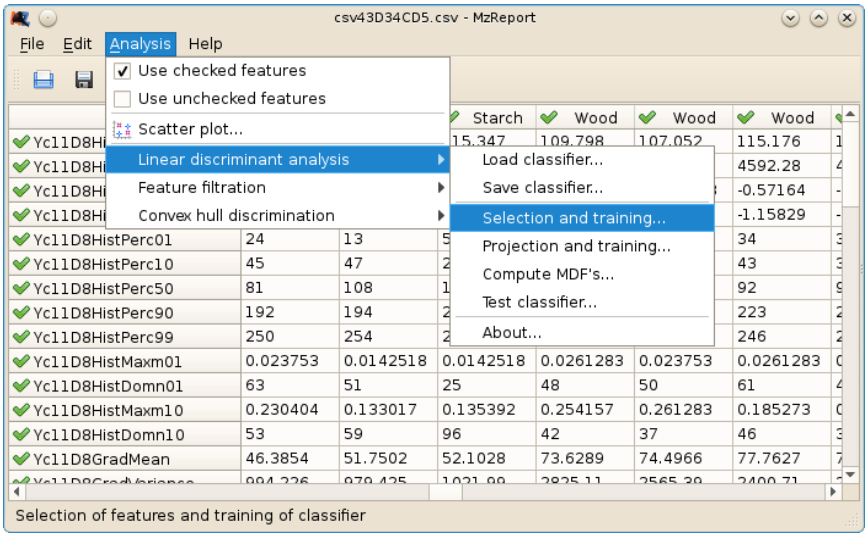


Select *Analysis* > *Options...* to open the feature extraction options dialog and select the *Local features* tab-page. Select the field defining a size of circular neighborhood (or sliding window) and modify its value. The size of the neighborhood should be big enough to cover some representative piece of the texture, so increase the size to 11. Press *OK* button to accept. Next, press the button or select *Analysis* > *Point driven computation*, to start feature extraction.

After successful feature extraction *MzReport* is started and the results are loaded into this module automatically.

How to perform discriminant analysis

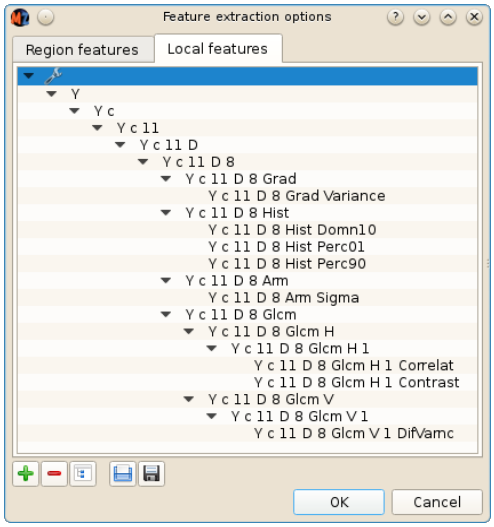
The results are presented as a spread-sheet, where each column represents feature vector of circular region created around one of the dots marked earlier. Make sure that all the columns and all the rows are checked to be used for the following analysis. Also, verify if the column headers show the names of the two texture classes. Save the results to a file (*File* > *Save...*)



To perform linear discriminant analysis select *Analysis* > *Linear discriminant analysis* > *Selection and training...* from the *MzReport* menu. The analysis options dialog will pop-up. It defines three parameters, *normalization*, *time* and *dimensionality*. Set the parameters to *normalization* = *standardize*, *time* = 60 and *dimensionality* = 5. Now, press the *OK* button to start feature selection and machine learning procedure. After the analysis succeeds the dialog with the most discriminative features appears. Press *OK* button to accept. Now, most of the feature names in *MzReport* are unchecked. The only checked feature names are the features found by the selection procedure.

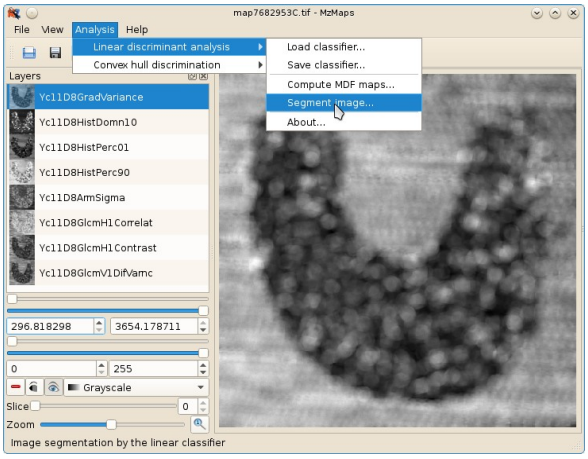
The procedure also creates rules for image segmentation. To save the rules select *Analysis* > *Linear discriminant analysis* > *Save classifier...* . The rules (classifiers) are saved to the text file.

How to segment an image

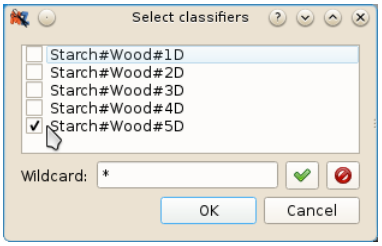


Switch back to the MaZda window with the example image still there. Open the feature options dialog (*Analysis* > *Options...*) and select the *Local features* tab-page. Now select the main root of the feature names tree and press to remove all the feature names. Press  button to load feature names from the text file with segmentation rules just created. Now the list of feature names should consist of the features selected as the most discriminative. Press *OK* button to accept. Next, press the  button, or select *Analysis* > *Maps computation* from the *MaZda* menu

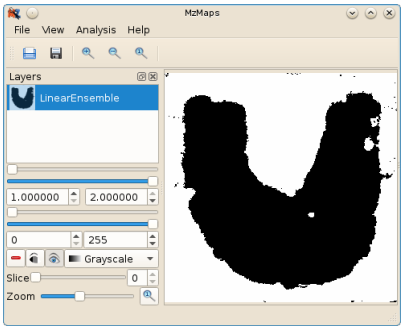
After successful feature extraction the *MzMaps* module is started and the results are loaded into this module automatically.



In *MzMaps* select *Analysis* > *Linear discriminant analysis* > *Segment image...*



In *Select classifiers* dialog check the one with the highest dimensionality (*Starch#Wood#5D*), uncheck the others and press *OK* button.



Another *MzMaps* window opens to show the result of segmentation. The binary image indicates the U-shape starch region in black and the background wood cells texture in white.