DEFINIENS
Understanding Images

Definiens

Developer 7

Reference Book

Definiens AG

www.definiens.com

Imprint and Version

Document Version 7.0.0.843

Copyright © 2007 Definiens AG. All rights reserved.

This document may be copied and printed only in accordance with the terms of the **Frame License Agreement for End Users** of the related **Definiens** software.

Published by

Definiens AG
Trappentreustr. 1
D-80339 München
Germany

Phone +49-89-231180-0
Fax +49-89-231180-90

E-mail **info@definiens.com**
Web **http://www.definiens.com**

Dear User,

Thank you for using **Definiens** software. We appreciate being of service to you with image intelligence solutions.

At Definiens we constantly strive to improve our products. We therefore appreciate all comments and suggestions for improvements concerning our software, training, and documentation.

Feel free to contact us via web form on the Definiens support website **http://www.definiens.com/support/index.htm**.

Thank you.

Legal Notes

Definiens®, **Definiens Cellenger®** and **Definiens Cognition Network Technology®** are registered trademarks of Definiens AG in Germany and other countries. **Cognition Network Technology™**, **Definiens eCognition™**, **Enterprise Image Intelligence™**, and **Understanding Images™**, are trademarks of Definiens AG in Germany and other countries.

All other product names, company names, and brand names mentioned in this document may be trademark properties of their respective holders.

Protected by patents US 7146380, US 7117131, US 6832002, US 6738513, US 6229920, US 6091852, EP 0863485, WO 00/54176, WO 00/60497, WO 00/63788 WO 01/45033, WO 01/71577, WO 01/75574, and WO 02/05198. Further patents pending.

Table of Contents

1 Introduction

This **Reference Book** lists detailed information about algorithms and features, and provides general reference information. For individual image analysis and rule set development you may wish to keep a printout ready at hand.

→ **Algorithms Reference** on page 11
→ **Features Reference** on page 83

2 About Rendering a Displayed Image

The **eCognition** image renderer creates the displayed image in two steps.

1. The first step reads out the displayed area from the selected image layers according to the screen size and zoom settings. Then **image layer equalization** is applied. The result is an 8-bit raw gray value image for each image layer. These gray value images are mixed into one raw RGB image by a layer mixer according to the current layer mixing settings.

→ About Image Layer Equalization on page 7

2. Finally the **image equalizing** is applied to create the output RGB image that is displayed on the screen.

→ About Image Equalization on page 8

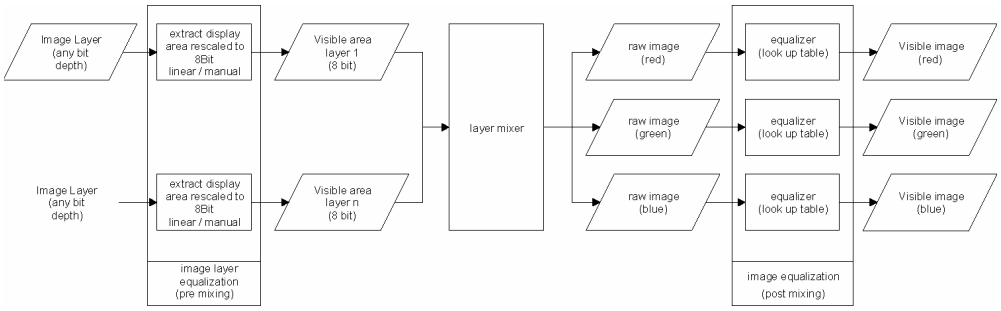


Figure 1: The rendering process of an image displayed in the project view.

2.1 About Image Layer Equalization

Image layer equalization is part of the rendering process of the image display within the project view.

Image layer equalization maps the input data of an image layer which may have different intensity ranges to the unified intensity range [0...255] of an 8-bit gray value image. For 8-bit data no image layer equalization is necessary. All other data types have to be converted into an 8-bit representation at this step of the rendering process.

This function is implemented as a mapping of the input range to the display range of [0...255]. Image layer equalization can be either linear or manual.

About Linear Image Layer Equalization

By default, the input data is mapped to the gray value range by a linear function.

Data Type	Input Range	Mapping Function
8-bit	[0...255]	$c_s = c_k$ (no transformation)
16-bit unsigned; 32-bit unsigned	$[0...max_2(c_k)]$	$c_s = 255 * c_k / max_2(c_k)$
16-bit signed; 32-bit signed	$[min_2(c_k)...max_2(c_k)]$	$c_s = 255*(c_k-min_2(c_k))/(max_2(c_k)-min_2(c_k))$
32-bit float	$[min_{10}(c_k)...max_{10}(c_k)]$	$c_s = 255*(c_k-min_{10}(c_k))/(max_{10}(c_k)-min_{10}(c_k))$

$\mathbf{c}_k$: intensity value in image layer k

$\mathbf{c}_s$: intensity value on the screen

$\mathbf{min}_2(\mathbf{c}_k)$ = max { x : x = -2^n; x <= c'^{min}_k } is the highest integer number that is a power of 2 and darker than the darkest actual intensity value of all pixel values of the selected layer.

$\mathbf{max}_2(\mathbf{c}_k)$ = min { x : x = 2^n; x >= c'^{max}_k } is the lowest integer number that is a power of 2 and is brighter than the brightest actual intensity value of all pixel values of the selected layer.

$\mathbf{min}_{10}(\mathbf{c}_k)$ = max { x : x = -10^n; x <= c'^{min}_k } is the highest integer number that is a power of 10 and darker than the darkest actual intensity value of all pixel values of the selected layer.

$\mathbf{max}_{10}(\mathbf{c}_k)$ = min { x : x = 10^n; x >= c'^{max}_k } is the lowest integer number that is a power of 10 and is brighter than the brightest actual intensity value of all pixel values of the selected layer.

About Manual Image Layer Equalization

With manual image layer equalization you can specify the mapping function for each layer individually by defining both the input range [c_{min}...c_{max}] and the mapping function.

→ **Manual Image Layer Equalization** section in User Guide

Data Type	Mapping Function
Linear	$c_s = 255*(c_k-c_{min})/(c_{max}-c_{min})$
Linear - inverse	$c_s = 255 - 255*(c_k-c_{min})/(c_{max}-c_{min})$
Gamma correction (positive)	$c_s = 255*((c_k-c_{min})/(c_{max}-c_{min}))^2$
Gamma correction (negative)	$c_s = 255*((c_k-c_{min})/(c_{max}-c_{min}))^{0,5}$
Gamma correction (positive) - inverse	$c_s = 255 - 255*((c_k-c_{min})/(c_{max}-c_{min}))^2$
Gamma correction (negative) - inverse	$c_s = 255 - 255*((c_k-c_{min})/(c_{max}-c_{min}))^{0,5}$

$\mathbf{c}_k$: intensity value in image layer k

$\mathbf{c}_s$: intensity value on the screen

$\mathbf{c}_{min}$: smallest input intensity value (adjustable)

$\mathbf{c}_{max}$: largest input intensity value (adjustable)

2.2 About Image Equalization

Image equalization is performed after all image layers are mixed into a raw RGB (Red, Green, Blue) image. . If more than one image layer is assigned to one screen color (Red, Green or Blue) this approach leads to higher quality results. Where only one image layer is assigned to each color, as is common, this approach is the same as applying equalization to the individual raw layer gray value images

There are different modes for image equalization available. All of them rely on image statistics. These are computed on the basis of a 256x256 pixel sized thumbnail of the current raw RGB image.

→ **Edit the Image Layer Mixing** section in User Guide

None

No (**None**) equalization allows you to see the image data as it is, which can be helpful at the beginning of rule set development, when looking for an approach.

The output from the image layer mixing is displayed without further modification.

Input Range	Mapping Function
[0...255]	[0...255]

Linear Equalization

Linear equalization with **1.00%** is the default for new scenes. Commonly it displays images with a higher contrast as without image equalization.

→ About Image Layer Equalization on page 7

Linear equalization maps each color Red, Green, and Blue (RGB) from an input range $[c_{min}...c_{max}]$ to the available screen intensity range [0...255] by a linear mapping. The input range can be modified by the percentage parameter **p**. The input range is computed such that p percent of the pixels are not part of the input range. In case p=0 this means the range of used color values is stretched to the range [0....255]. For p>0 the mapping ignores p/2 percent of the darkest pixels and p/2 percent of the brightest pixels. In many cases a small value of p lead to better results because the available color range can be better used for the relevant data by ignoring the outliers.

$c_{min} = \max \{ c : \#\{ (x,y) : c_k(x,y) < c_{min}\} / (sx*sy) >= p/2 \}$

$c_{max} = \min \{ c : \#\{ (x,y) : c_k(x,y) > c_{max}\} / (sx*sy) >= p/2 \}$

Input Range	Mapping Function
$[c_{min}...c_{max}]$	$c_s = 255*(c_k-c_{min})/(c_{max}-c_{min}))$

For images with no color distribution (i.e. all pixels having the same intensity) the result of **Linear** equalization will be a black image independent of the image layer intensities.

Standard Deviation Equalization

With its default parameter of 3.0, **Standard deviation** renders a similar display as **Linear** equalization. Use a parameter around 1.0 for an exclusion of dark and bright outliers.

Standard deviation equalization maps the input range to the available screen intensity range [0...255] by a linear mapping. The input range $[c_{min}...c^{max}]$ can be modified by the width **p**. The input range is computed such that the center of the input range represents the mean value of the pixel intensities mean($\mathbf{c}_k$). The left and right borders of the input range are computed by taking the **n** times the standard deviation σ_k to the left and the right. You can modify 8 the parameter n.

$c_{min} = mean(c_k) - n * \sigma_k$

$c_{max} = mean(c_k) + n * \sigma_k$

Input Range	Mapping Function
$[c_{min}...c_{max}]$	$c_s = 255*(c_k-c_{min})/(c_{max}-c_{min}))$

Gamma Correction Equalization

Gamma correction equalization is used to improve the contrast of dark or bright areas by spreading the corresponding gray values.

Gamma correction equalization maps the input range to the available screen intensity range [0...255] by a polynomial mapping. The input range $[c_{min}...c_{max}]$ cannot be be modified and is defined by the smallest and the largest existing pixel values.

$c_{min} = c'^{min}_k$

$c_{max} = c'^{max}_k$

Input Range	Mapping Function
$[c_{min}...c_{max}]$	$c_s = 255*(((c_k-c_{min})/(c_{max}-c_{min}))^n)$

You can be modify the exponent of the mapping function by editing the equalization parameter 8. Values of n less than 1 emphasize darker regions of the image, values larger than one emphasize darker areas of the image. A value from n=1 represents the linear case.

Histogram Equalization

Histogram equalization is well suited for **LANDSAT** images but can lead to substantial over-stretching on many normal images. It can be helpful in cases you want to display dark areas with more contrast.

Histogram equalization maps the input range to the available screen intensity range [0...255] by a nonlinear function. Simply said, the mapping is defined by the property that each color value of the output image represents the same number of pixels. The respective algorithm is more complex and can be found in standard image processing literature.

Manual Image Layer Equalization

Manual image layer equalization allows you to control equalization in detail. For each image layer individually, you can set the equalization method specifying the mapping function. Further you can define the input range by setting minimum and maximum values.

→ **Manual Image Layer Equalization** section in User Guide

3 Algorithms Reference

Contents in This Chapter

A single process executes an algorithm on an image object domain. It is the elementary unit of a rule set providing a solution to a specific image analysis problem. Processes are the main working tools for developing rule sets. A rule set is a sequence of processes which are executed in the defined order.

The image object domain is a set of image objects. Every process loops through this set of image objects one by one and applies the algorithm to each single image object. This image object is referred to as the current image object.

Create a Process

A single process can be created using the **Edit Process** dialog box in which you can define:

→ **Use Processes** section ot the **User Guide**

- the method of the process from an algorithm list, for example **multiresolution segmentation** or **classification**,
- the image object domain on which an algorithm should be performed,
- detailed parameter settings of the algorithm.

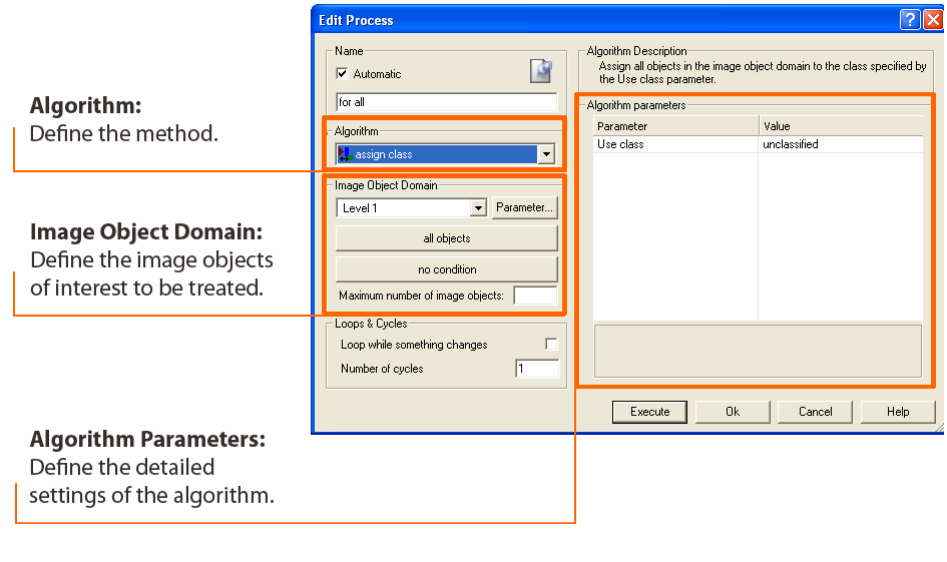


Figure 2: Edit Process dialog box with highlighted group boxes.

Specify Algorithm Parameters

Depending on the chosen algorithm, you have to specify different parameters.

1. Define the individual settings of the algorithm in the **Algorithms Parameters** group box. If available, click a plus sign (+) button to expand the table to access additional parameters.

 (expand)

2. To edit **Values** of **Algorithm Parameters**, select the parameter name or its value by clicking. Depending on the type of value, change the value by one of the following:
   - Edit many values directly within the value field.
   - Click the ellipsis button located inside the value field. A dialog box opens allowing you to configure the value.

 (ellipsis button)

   - Click the drop-down arrow button placed inside the value field. Select from a drop-down list to configure the value.

 (drop-down arrow button)

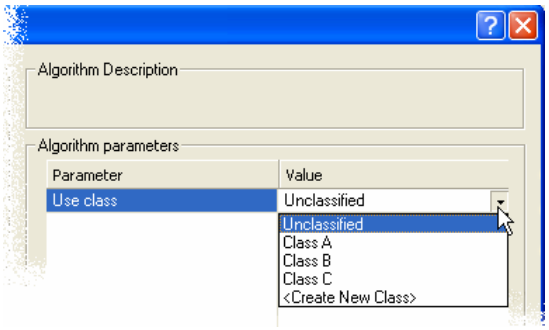


Figure 3: Select an Algorithm Parameter for editing values.

3.1 Process Related Operation Algorithms

The Process Related Operation algorithms are used to control other processes.

3.1.1 Execute Child Processes

- **execute child processes**

Execute all child processes of the process.

Use the **execute child processes** algorithm in conjunction with the **no image object** domain to structure to your process tree. A process with this settings serves an container for a sequence of functional related processes.

Use the **execute child processes** algorithm in conjunction with other image object domains (for example, the **image object level** domain) to loop over a set of image objects. All contained child processes will be applied to the image objects in the image object domain. In this case the child processes usually use one of the following as image object domain: **current image object, neighbor object, super object, sub objects**.

3.1.2 Set Rule Set Options

Select settings that control the rules of behavior of the rule set.

This algorithm enables you to control certain settings for the rule set or for only part of the rule set. For example, you may want to apply particular settings to analyze large objects and change them to analyze small objects. In addition, because the settings are part of the rule set and not on the client, they are preserved when the rule set is run on a server.

Apply to Child Processes Only

Value	Description
Yes	Setting changes apply to child processes of this algorithm only.
No	Settings apply globally, persisting after completion of execution..

Distance Calculation

Value	Description
Smallest enclosing rectangle	Uses the smallest enclosing rectangle of an image object for distance calculations.
Center of gravity	Uses the center of gravity of an image object for distance calculations.
Default	Reset to the default when the rule set is saved.
Keep Current	Keep the current setting when the rule set is saved,

Current Resampling Method

Value	Description
Center of Pixel	Resampling occurs from the center of the pixel.
Upper left corner of pixel	Resampling occurs from the upper left corner of the pixel.
Default	Reset to the default when the rule set is saved.
Keep Current	Keep the current setting when the rule set is saved.

Evaluate Conditions on Undefined Features as 0

Value	Description
Yes	Ignore undefined features.
No	Evaluate undefined features as 0.
Default	Reset to the default when the rule set is saved.
Keep Current	Keep the current setting when the rule set is saved.

Polygons Base Polygon Threshold

Set the degree of abstraction for the base polygons.

Default: **1.25**

Polygons Shape Polygon Threshold

Set the degree of abstraction for the shape polygons. Shape polygons are independent of the topological structure and consist of at least three points. The threshold for shape polygons can be changed any time without the need to recalculate the base vectorization.

Default: **1**

Polygons Remove Slivers

Enable **Remove slivers** to avoid intersection of edges of adjacent polygons and self-intersections of polygons.

Sliver removal becomes necessary with higher threshold values for base polygon generation. Note that the processing time to remove slivers is high, especially for low thresholds where it is not needed anyway.

Value	Description
No	Allow intersection of polygon edges and self-intersections.
Yes	Avoid intersection of edges of adjacent polygons and self-intersections of polygons.
Default	Reset to the default when the rule set is saved.
Keep Current	Keep the current setting when the rule set is saved.

3.2 Segmentation Algorithms

Segmentation algorithms are used to subdivide the entire image represented by the pixel level domain or specific image objects from other domains into smaller image objects.

Definiens provides several different approaches to this well known problem ranging from very simple algorithms like chessboard and quadtree based segmentation to highly sophisticated methods like multiresolution segmentation or the contrast filter segmentation.

Segmentation algorithms are required whenever you want to create new image objects levels based on the image layer information. But they are also a very valuable tool to refine existing image objects by subdividing them into smaller pieces for a more detailed analysis.

3.2.1 Chessboard Segmentation

chessboard segmentation

Split the pixel domain or an image object domain into square image objects.

A square grid aligned to the image left and top borders of fixed size is applied to all objects in the domain and each object is cut along these grid lines.

Example

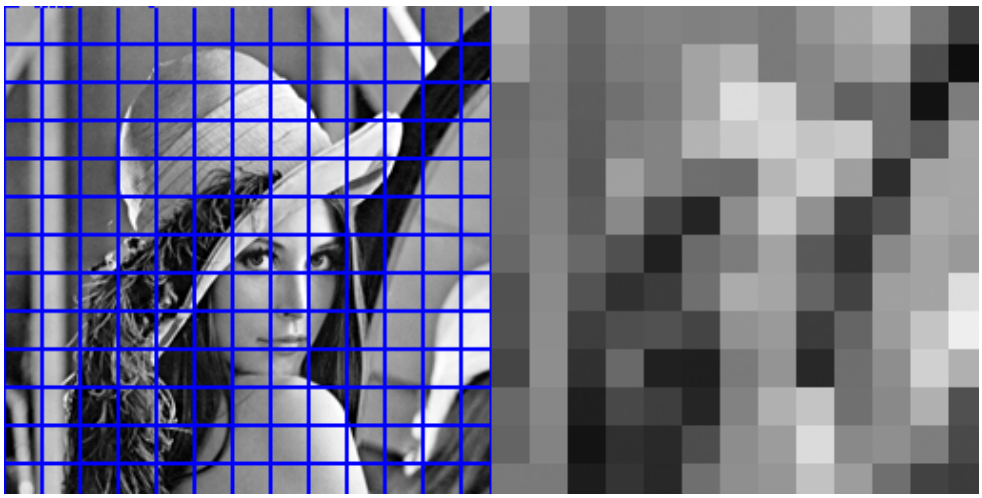


Figure 4: Result of chessboard segmentation with object size 20.

Object Size

The **Object size** defines the size of the square grid in pixels.

> **Note**
>
> Variables will be rounded to the nearest integer.

Level Name

Enter the name for the new image object level.

Precondition: This parameter is only available, if the domain **pixel level** is selected in the process dialog.

Thematic Layers

Specify the thematic layers that are to be considered in addition for segmentation.

Each thematic layer that is used for segmentation will lead to additional splitting of image objects while enabling consistent access to its thematic information. You can segment an image using more than one thematic layer. The results are image objects representing proper intersections between the thematic layers.

Precondition: Thematic layers must be available.

If you want to produce image objects based exclusively on thematic layer information, you can select a chessboard size larger than you image size.

3.2.2 Quad Tree Based Segmentation

quad tree based segmentation

Split the pixel domain or an image object domain into a quad tree grid formed by square objects.

A quad tree grid consists of squares with sides each having a power of 2 and aligned to the image left and top borders is applied to all objects in the domain and each object is

cut along this grid lines. The quad tree structure is build in a way that each square has first maximum possible size and second fulfills the homogeneity criteria as defined by the mode and scale parameter.

Example

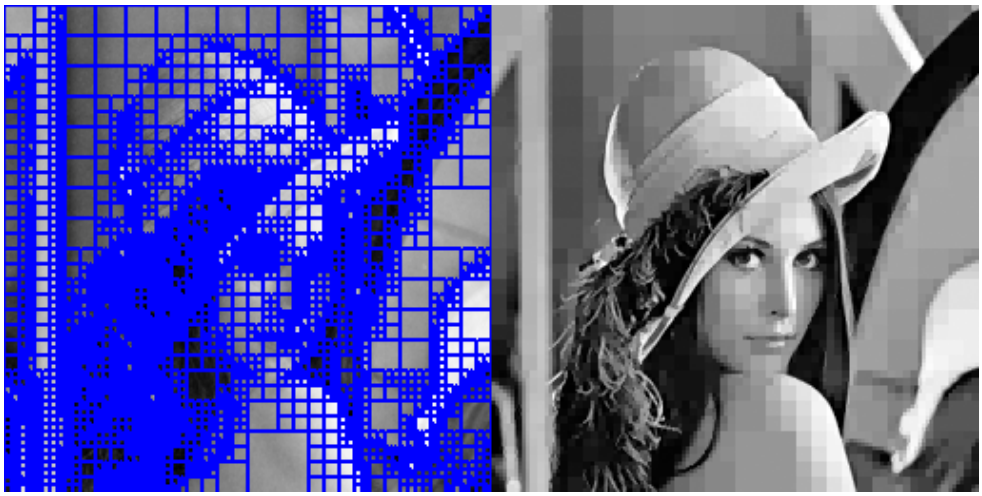


Figure 5: Result of quad tree based segmentation with mode color and scale 40.

Mode

Value	Description
Color	The maximal color difference within each square image object is less than the **Scale** value.
Super Object Form	Each square image object must completely fit into the superobject. Precondition: This mode only works with an additional upper image level.

Scale

Defines the maximum color difference within each selected image layer inside square image objects.

Precondition: Only used in conjunction with the **Color** mode.

Level Name

Enter the name for the new image object level.

Precondition: This parameter is only available, if the domain **pixel level** is selected in the process dialog.

Thematic Layers

Specify the thematic layers that are to be considered in addition for segmentation.

Each thematic layer that is used for segmentation will lead to additional splitting of image objects while enabling consistent access to its thematic information. You can segment an image using more than one thematic layer. The results are image objects representing proper intersections between the thematic layers.

Precondition: Thematic layers must be available.

3.2.3 Contrast Split Segmentation

▪ **contrast split segmentation**

Use the **contrast split segmentation** algorithm to segment an image or an image object into dark and bright regions.

The contrast split algorithm segments an image (or image object) based on a threshold that maximizes the contrast between the resulting bright objects (consisting of pixels with pixel values above threshold) and dark objects (consisting of pixels with pixel values below the threshold).

The algorithm evaluates the optimal threshold separately for each image object in the image object domain. If the pixel level is selected in the image object domain, the algorithm first executes a chessboard segmentation, and then performs the split on each square.

The algorithm achieves the optimization by considering different pixel values as potential thresholds. The test thresholds range from the minimum threshold to the maximum threshold, with intermediate values chosen according to the step size and stepping type parameter. If a test threshold satisfies the minimum dark area and minimum bright area criterion, the contrast between bright and dark objects is evaluated. The test threshold causing the largest contrast is chosen as best threshold and used for splitting.

Chessboard Tile Size

Available only if **pixel level** is selected in the **Image Object Domain.** Enter the chessboard tile size.

→ Chessboard Segmentation on page 15

Default: **1000**

Level Name

Select or enter the level that will contain the results of the segmentation.
Available only if the **pixel level** is in the image object domain.

Minimum Threshold

Enter the minimum gray value that will be considered for splitting.
The algorithm will calculate the threshold for gray values from the **Scan Start** value to the **Scan Stop** value.

Default: **0**

Maximum Threshold

Enter the maximum gray value that will be considered for splitting.
The algorithm will calculate the threshold for gray values from the **Scan Start** value to the **Scan Stop** value.

Default: **255**

Step Size

Enter the step size by which the threshold will increase from the **Minimum threshold** to the **Maximum threshold.** The value will either be added to the threshold or multiplied by the threshold, according to the selection in the **Stepping type** field.

The algorithm recalculates a new best threshold each time the threshold is changed by application of the values in the **Step size** and **Stepping type** fields, until the **Maximum threshold** is reached.

Higher values entered for **Step size** will tend to execute more quickly; smaller values will tend to achieve a split with a larger contrast between bright and dark objects.

Stepping Type

Use the drop-down list to select one of the following:

add: Calculate each step by adding the value in the **Scan Step** field.

multiply: Calculate each step by multiplying by the value in the **Scan Step** field.

Image Layer

Select the image layer where the contrast is to be maximized.

Class for Bright Objects

Create a class for image objects brighter than the threshold or select one from the drop-down list.
Image objects will not be classified if the value in the **Execute splitting** field is **No.**

Class for Dark Objects

Create a class for image objects darker than the threshold or select one from the drop-down list.
Image objects will not be classified if the value in the **Execute splitting** field is **No.**

Contrast Mode

Select the method the algorithm uses to calculate contrast between bright and dark objects. The algorithm calculates possible borders for image objects and the border values are used in two of the following methods.

a = the mean of bright border pixels.

b = the mean of dark border pixels.

Value	Description
edge ratio	**a** – **b**/**a** + **b**
edge difference	**a – b**
object difference	The difference between the mean of all bright pixels and the mean of all dark pixels.

Execute Splitting

Select **Yes** to split objects with best detected threshold. Select **No** to simply compute the threshold without splitting.

Best Threshold

Enter a variable to store the computed pixel value threshold that maximizes the contrast.

Best Contrast

Enter a variable to store the computed contrast between bright and dark objects when splitting with the best threshold. The computed value will be different for each **Contrast mode.**

Minimum Relative Area Dark

Enter the minimum relative dark area.

Segmentation into dark and bright objects only occurs if the relative dark area will be higher than the value entered.

Only thresholds that lead to a relative dark area larger than value entered are considered as best threshold.

Setting this value to a number greater than 0 may increase speed of execution.

Minimum Relative Area Bright

Enter the minimum relative bright area.

Only thresholds that lead to a relative bright area larger than value entered are considered as best threshold.

Setting this value to a number greater than 0 may increase speed of execution.

Minimum Contrast

Enter the minimum contrast value threshold.
Segmentation into dark and bright objects only occurs if a contrast higher than the value entered can be achieved.

Minimum Object Size

Enter the minimum object size in pixels that can result from segmentation.

Only larger objects will be segmented. Smaller objects will be merged with neighbors randomly.

The default value of **1** effectively inactivates this option.

3.2.4 Multiresolution Segmentation

multiresolution segmentation

Apply an optimization procedure which locally minimizes the average heterogeneity of image objects for a given resolution. It can be applied on the pixel level or an image object level domain.

Example

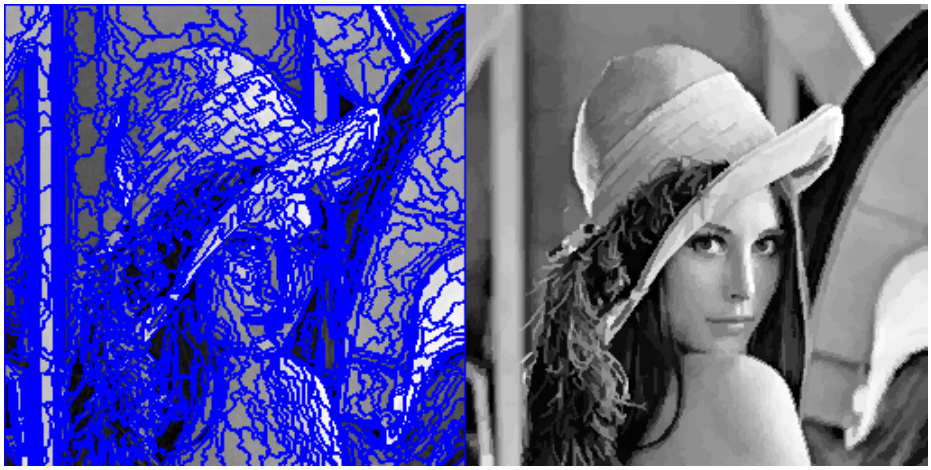


Figure 6: Result of multiresolution segmentation with scale 10, shape 0.1 and compactness 0.5.

Level Name

The **Level name** defines the name for the new image object level.

Precondition: This parameter is only available, if a new image object level will be created by the algorithm. To create new image object levels use either the image object domain **pixel level** in the process dialog or set the level mode parameter to **create above** or **create below**.

Level Usage

Use the drop down arrow to select one of the available modes. The algorithm is applied according to the mode based on the image object level that is specified by the image object domain.

Value	Description
Use current	Applies Multiresolution Segmentation to the existing image object level. Objects can be merged and split depending on the algorithm settings.
Use current (merge only)	Applies Multiresolution Segmentation to the existing image object level. Objects can only be merged. Usually this mode is used together with stepwise increases of the scale parameter.
Create above	Creates a copy of the image object level as super objects.
Create below	Creates a copy of the image object level as sub objects.

Precondition: This parameter is not visible if **pixel level** is selected as image object domain in the **Edit Process** dialog box.

Image Layer Weights

Image layers can be weighted differently to consider image layers depending on their importance or suitability for the segmentation result.

The higher the weight which is assigned to an image layer, the more of its information will be used during the segmentation process , if it utilizes the pixel information. Consequently, image layers that do not contain the information intended for representation by the image objects should be given little or no weight.

Example: When segmenting a geographical LANDSAT scene using multiresolution segmentation, the segmentation weight for the spatially coarser thermal layer should be set to **0** in order to avoid deterioration of the segmentation result by the blurred transient between image objects of this layer.

Thematic Layers

Specify the thematic layers that are to be considered in addition for segmentation.

Each thematic layer that is used for segmentation will lead to additional splitting of image objects while enabling consistent access to its thematic information. You can segment an image using more than one thematic layer. The results are image objects representing proper intersections between the thematic layers.

Precondition: Thematic layers must be available.

Scale Parameter

The **Scale parameter** is an abstract term which determines the maximum allowed heterogeneity for the resulting image objects. For heterogeneous data the resulting objects for a given scale parameter will be smaller than in more homogeneous data. By modifying the value in the **Scale parameter** value you can vary the size of image objects.

Tip

Produce Image Objects that Suit the Purpose (1)

Always produce image objects of the biggest possible scale which still distinguishes different image regions (as large as possible and as fine as necessary). There is a tolerance concerning the scale of the image objects representing an area of a consistent classification due to the equalization achieved by the classification. The separation of different regions is more important than the scale of image objects.

Composition of Homogeneity Criterion

The object homogeneity to which the scale parameter refers is defined in the **Composition of homogeneity criterion** field. In this circumstance, homogeneity is used as a synonym for minimized heterogeneity. Internally three criteria are computed: Color, smoothness, and compactness. These three criteria for heterogeneity maybe applied multifariously. For most cases the color criterion is the most important for creating meaningful objects. However, a certain degree of shape homogeneity often improves the quality of object extraction. This is due to the fact that the compactness of spatial objects is associated with the concept of image shape. Thus, the shape criteria are especially helpful in avoiding highly fractured image object results in strongly textured data (for example, radar data).

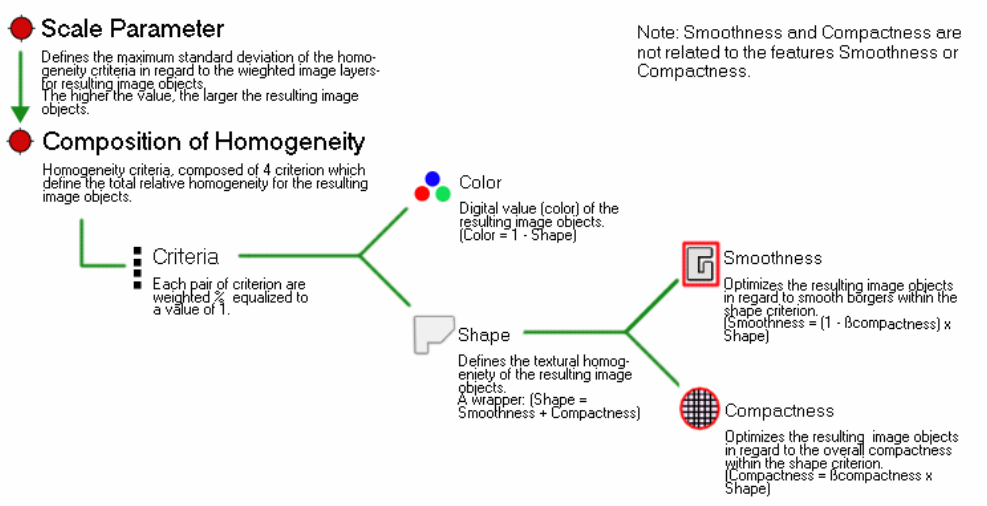


Figure 7: Multiresolution concept flow diagram.

Color and Shape

By modify the shape criterion, you indirectly define the color criteria. In effect, by decreasing the value assigned to the **Shape** field, you define **to which percentage** the spectral values of the image layers will contribute to the entire homogeneity criterion. This is weighted against the percentage of the shape homogeneity, which is defined in the **Shape** field. Changing the weight for the **Shape** criterion to **1** will result in objects more optimized for spatial homogeneity. However, the shape criterion cannot have a value more than **0.9**, due to the obvious fact that without the spectral information of the image, the resulting objects would not be related to the spectral information at all.

Use the slider bar to adjust the amount of **Color** and **Shape** to be used for the segmentation.

Note

The color criterion is indirectly defined by the **Shape** value.

The **Shape** value can not exceed **0.9**.

Tip

Produce Image Objects that Suit the Purpose (2)

Use as much color criterion as possible while keeping the shape criterion as high as necessary to produce image objects of the best border smoothness and compactness. The reason for this is that a high degree of shape criterion works at the cost of spectral homogeneity. However, the spectral information is, at the end, the primary information contained in image data. Using too much shape criterion can therefore reduce the quality of segmentation results.

In addition to spectral information the object homogeneity is optimized with regard to the object shape. The shape criterion is composed of two parameters:

Smoothness

The smoothness criterion is used to optimize image objects with regard to smoothness of borders. To give an example, the smoothness criterion should be used when working on very heterogeneous data to inhibit the objects from having frayed borders, while maintaining the ability to produce non-compact objects.

Compactness

The compactness criterion is used to optimize image objects with regard to compactness. This criterion should be used when different image objects which are rather compact, but are separated from non-compact objects only by a relatively weak spectral contrast.

Use the slider bar to adjust the amount of **Compactness** and **Smoothness** to be used for the segmentation.

Note

It is important to notice that the two shape criteria are not antagonistic. This means that an object optimized for compactness might very well have smooth borders. Which criterion to favor depends on the actual task.

3.2.5 Spectral Difference Segmentation

spectral difference segmentation

Merge neighboring objects according to their mean layer intensity values. Neighboring image objects are merged if the difference between their layer mean intensities is below the value given by the maximum spectral difference.

This algorithm is designed to refine existing segmentation results, by merging spectrally similar image objects produced by previous segmentations.

Note

This algorithm cannot be used to create new image object levels based on the pixel level domain.

Level Name

The **Level name** defines the name for the new image object level.

Precondition: This parameter is only available, if a new image object level will be created by the algorithm. To create new image object levels use either the image object domain **pixel level** in the process dialog or set the level mode parameter to **create above** or **create below**.

Maximum Spectral Difference

Define the amount of spectral difference between the new segmentation for the generated image objects. If the difference is below this value, neighboring objects are merged.

Image Layer Weights

Image layers can be weighted differently to consider image layers depending on their importance or suitability for the segmentation result.

The higher the weight which is assigned to an image layer, the more of its information will be used during the segmentation process , if it utilizes the pixel information. Consequently, image layers that do not contain the information intended for representation by the image objects should be given little or no weight.

Example: When segmenting a geographical LANDSAT scene using multiresolution segmentation, the segmentation weight for the spatially coarser thermal layer should be set to **0** in order to avoid deterioration of the segmentation result by the blurred transient between image objects of this layer.

Thematic Layers

Specify the thematic layers that are to be considered in addition for segmentation.

Each thematic layer that is used for segmentation will lead to additional splitting of image objects while enabling consistent access to its thematic information. You can segment an image using more than one thematic layer. The results are image objects representing proper intersections between the thematic layers.

Precondition: Thematic layers must be available.

3.2.6 Contrast Filter Segmentation

contrast filter segmentation

Use pixel filters to detect potential objects by contrast and gradient and create suitable object primitives. An integrated reshaping operation modifies the shape of image objects to help form coherent and compact image objects.

The resulting pixel classification is stored in an internal thematic layer. Each pixel is classified as one of the following classes: no object, object in first layer, object in second layer, object in both layers, ignored by threshold.

Finally a chessboard segmentation is used to convert this thematic layer into an image object level.

Use this algorithm as first step of your analysis to improve overall image analysis performance substantially.

Chessboard Segmentation

The settings configure the final chessboard segmentation of the internal thematic layer.

See **chessboard segmentation** reference.

→ Chessboard Segmentation on page 15

Input Parameters

These parameters are identical for the first and the second layer.

Layer

Choose the image layer to analyze form the drop-down menu. Use **<no layer>** to disable one of the two filters. If you select **<no layer>**, then the following parameters will be inactive.

Scale 1-4

You can define several scales to be analyzed at the same time. If at least one scale is tested positive, the pixel will be classified as image object.

By default, no scale is used what is indicated by a scale value of 0. To define a scale, edit the scale value.

The scale value **n** defines a frame with a side length of 2d' with **d** := {all pixels with distance to the current pixel ≤□|n|*2+1 but > (|n|-2)*2+1} with the current pixel in its center. The mean value of the pixels inside this frame is compared with the mean value of the pixels inside a cube with a side length of 2d' with **d'** := {all pixels with distance to the current pixel ≤ (|n|-2)*2+1 but not the pixel itself}. In case of |n|≤3 it is just the pixel value.

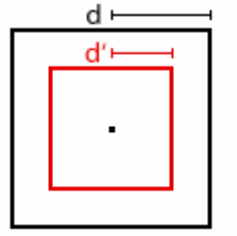


Figure 8: Scale testing of the contrast filter segmentation.

Select a positive scale value to find objects that are brighter than their surroundings on the given scale. Select a negative scale value to find objects that are darker than their surroundings on the given scale.

Gradient

Use additional minimum gradient criterion for objects. Using gradients can increase the computing time the algorithm. Set this parameter to 0 to disable the gradient criterion.

Lower Threshold

Pixels with layer intensity below this threshold will be assigned to the **ignored by threshold** class.

Upper Threshold

Pixels with layer intensity above this threshold will be assigned to the **ignored by threshold** class.

ShapeCriteria Settings

If you expect coherent and compact image objects, the shape criteria parameter provides an integrated reshaping operation which modifies the shape of image objects by cutting protruding parts and filling indentations and hollows.

ShapeCriteria Value

Protruding parts of image objects are declassified if a direct line crossing the hollow is smaller or equal than the **ShapeCriteria value**.

Indentations and hollows of image objects are classified as the image object if a direct line crossing the hollow is smaller or equal than the **ShapeCriteria value**.

If you do not want any reshaping, set the **ShapeCriteria value** to 0.

Working on Class

Select a class of image objects for reshaping.

Classification Parameters

The pixel classification can be transferred to the image object level using the class parameters.

Classification Parameters

Enable Class Assignment

Select **Yes** or **No** in order to use or disable the **Classification parameters**. If you select **No**, then the following parameters will be inactive.

No Objects

Pixels failing to meet the defined filter criteria will be assigned the selected class.

Ignored by Threshold

Pixels with layer intensity below or above the **Threshold** value will be assigned the selected class.

Object in First Layer

Pixels than match the filter criteria in **First layer**, but not the **Second layer** will be assigned the selected class.

Objects in Both Layers

Pixels than match the filter criteria value in both **Layer**s will be assigned the selected class.

Objects in Second Layer

Pixels than match the **Scale** value in **Second layer**, but not the **First layer** will be assigned the selected class.

3.3 Basic Classification Algorithms

Classification algorithms analyze image objects according defined criteria and assign them each to a class that best meets the defined criteria.

3.3.1 Assign Class


assign class

Assign all objects of the image object domain to the class specified by the **Use class** parameter. The membership value for the assigned class is set to 1 for all objects independent of the class description. The second and third best classification results are set to 0 .

Use class

Select the class for the assignment from the drop-down list box. You can also create a new class for the assignment within the drop-down list.

3.3.2 Classification


classification

Evaluates the membership value of an image object to a list of selected classes. The classification result of the image object is updated according to the class evaluation result. The three best classes are stored in the image object classification result. Classes without a class description are assumed to have a membership value of 1.

Active classes

Choose the list of active classes for the classification.

Erase old classification, if there is no new classification

Value	Description
Yes	If the membership value of the image object is below the acceptance threshold (see classification settings) for all classes, the current classification of the image object is deleted.
No	If the membership value of the image object is below the acceptance threshold (see classification settings) for all classes, the current classification of the image object is kept.

Use Class Description

Value	Description
Yes	Class descriptions are evaluated for all classes. The image object is assigned to the class with the highest membership value.
No	Class descriptions are ignored. This option delivers valuable results only if **Active classes** contains exactly one class.

If you do not use the class description, it is recommended to use the algorithm **assign class** algorithm instead.

→ Assign Class on page 28

3.3.3 Hierarchical Classification


hierarchical classification

Evaluate the membership value of an image object to a list of selected classes.

The classification result of the image object is updated according to the class evaluation result. The three best classes are stored as the image object classification result. Classes without a class description are assumed to have a membership value of 0. Class related features are considered only if explicitly enabled by the according parameter.

Note

This algorithm is optimized for applying complex class hierarchies to entire image object levels. This reflects the classification algorithm of **eCognition Professional 4**. When working with domain specific classification in processes the algorithms **assign class** and **classification** are recommended.

Active classes

Choose the list of active classes for the classification.

Use Class-Related Features

Enable to evaluate all class-related features in the class descriptions of the selected classes. If this is disabled these features will be ignored.

3.3.4 Remove Classification


remove classification

Delete specific classification results from image objects.

Classes

Select classes that should be deleted from image objects.

Process

Enable to delete computed classification results created via processes and other classification procedures from the image object.

Manual

Enable to delete manual classification results from the image object.

3.4 Advanced Classification Algorithms

Advanced classification algorithms classify image objects that fulfill special criteria like being enclosed by another image object or being the smallest or the largest object in a hole set of object.

3.4.1 Find Domain Extrema

find domain extrema

Classify image objects fulfilling a local extrema condition within the image object domain according to an image object feature. This means that either the image object with smallest or the largest feature value within the domain will be classified according to the classification settings.

Example

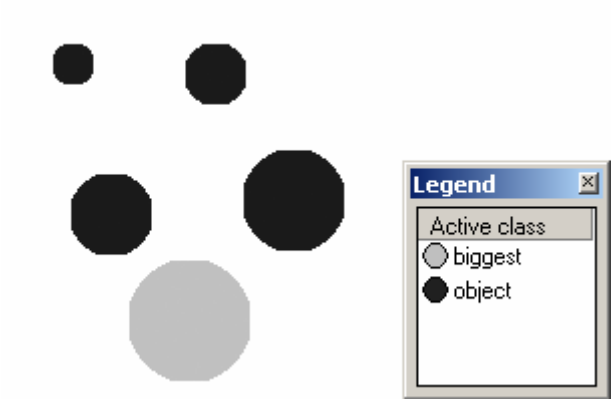


Figure 9: Result of find domain extrema using Extrema Type Maximum and Feature Area.

Extrema Type

Choose **Minimum** for classifying image objects with the smallest feature values and **Maximum** for classifying image objects with largest feature values.

Feature

Choose the feature to use for finding the extreme values.

Accept Equal Extrema

Enable the algorithm to **Accept equal extrema**. This parameter defines the behavior of the algorithm if more than one image object is fulfilling the extreme condition. If enabled all image objects will be classified. If not none of the image objects will be classified.

Compatibility Mode

Select **Yes** from the **Value** field to enable compatibility with older software versions (version 3.5 and 4.0). This parameter will be removed with future versions.

Classification Settings

Specifies the classification that will be applied to all image objects fulfilling the extreme condition.

See **classification** algorithm for details.

→ Classification on page 28

Note

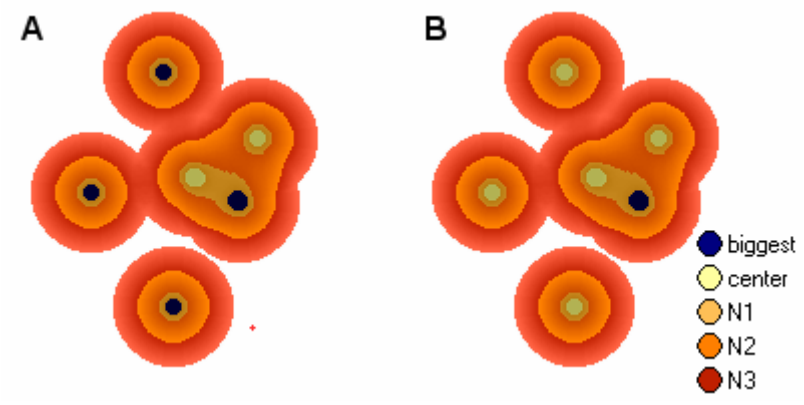
At least one class needs to be selected in the active class list for this algorithm

3.4.2 Find Local Extrema

find local extrema

Classify image objects fulfilling a local extrema condition according to an image object feature within a search domain in their neighborhood. Image objects with either the smallest or the largest feature value within a specific neighborhood will be classified according to the classification settings.

Example



Parameter	Value
Image Object Domain	all objects on level classified as center
Feature	Area
Extrema Type	Maximum
Search Range	80 pixels
Class Filter for Search	center, N1, N2, biggest
Connected	A) true B) false

Search Settings

With the **Search Settings** you can specify a search domain for the neighborhood around the image object.

Class Filter

Choose the classes to be searched. Image objects will be part of the search domain if they are classified with one of the classes selected in the class filter.

Note

Always add the class selected for the classification to the search class filter. Otherwise cascades of incorrect extrema due to the reclassification during the execution of the algorithm may appear.

Search Range

Define the search range in pixels. All image objects with a distance below the given search range will be part of the search domain. Use the drop down arrows to select zero or positive numbers.

Connected

Enable to ensure that all image objects in the search domain are connected with the analyzed image object via other objects in the search range.

Compatibility Mode

Select **Yes** from the **Value** field to enable compatibility with older software versions (version 3.5 and 4.0). This parameter will be removed with future versions.

Conditions

Define the extrema conditions.

Extrema Type

Choose **Minimum** for classifying image objects with the smallest feature values and **Maximum** for classifying image objects with largest feature values.

Feature

Choose the feature to use for finding the extreme values.

Extrema Condition

This parameter defines the behaviour of the algorithm if more than one image object is fulfilling the extrema condition.

Value	Description
Do not accept equal extrema	None of the image objects will be classified.
Accept equal extrema	All of the image objects will be classified.
Accept first equal extrema	The first of the image objects will be classified.

Classification Settings

Specifies the classification that will be applied to all image objects fulfilling the extremal condition.

See **classification** algorithm for details.

→ Classification on page 28

Note

At least one class needs to be selected in the active class list for this algorithm.

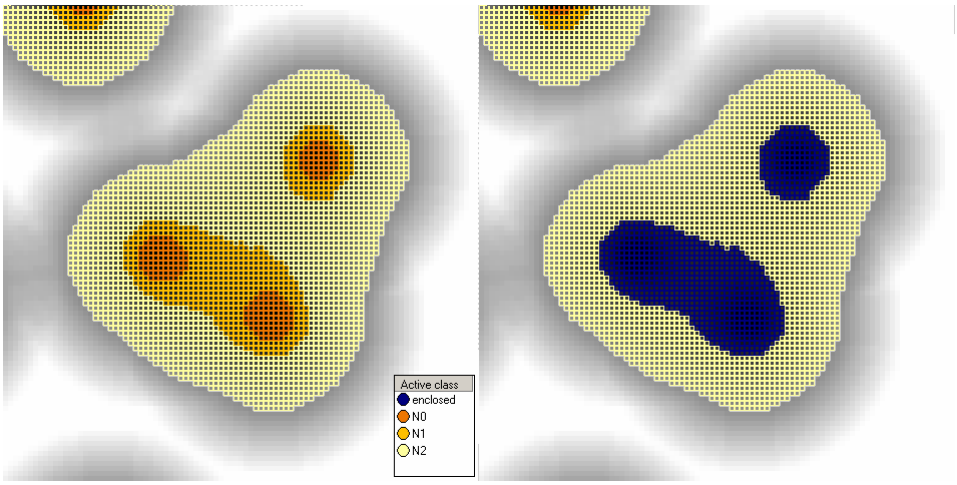
3.4.3 Find Enclosed by Class

find enclosed by class

Find and classify image objects that are completely enclosed by image objects belonging to certain classes.

If an image object is located at the border of the image, it will not be found and classified by **find enclosed by class**. The shared part of the outline with the image border will not be recognized as enclosing border.

Example



Left: Input of find enclosed by class:image object domain: image object level, class filter: N0, N1. Enclosing class: N2 Right: Result of find enclosed by class:
Enclosed objects get classified with the class **enclosed.** You can notice that the objects at the upper image border are not classified as **enclosed**.

Enclosing Classes

Choose the classes that might be enclosing the image objects.

Compatibility Mode

Select **Yes** from the **Value** field to enable compatibility with older software versions (version 3.5 and 4.0). This parameter will be removed with future versions.

Classification Settings

Choose the classes that should be used to classify encloses image objects.

See **classification** algorithm for details.

→ Classification on page 28

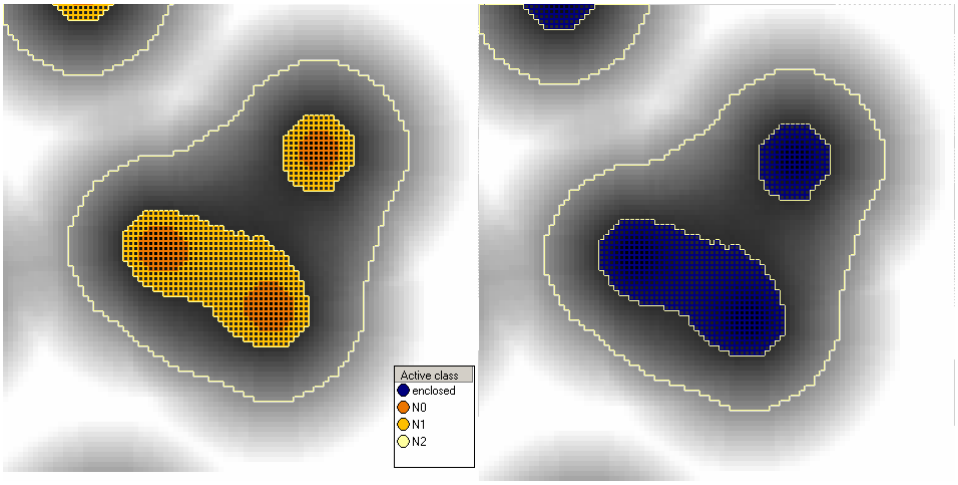
3.4.4 Find Enclosed by Image Object

find enclosed by image object

Find and classify image objects that are completely enclosed by image objects from the image object domain.

Enclosed image objects located at the image border will be found and classified by **find enclosed by image object**. The shared part of the outline with the image border will be recognized as enclosing border.

Example



Left: Input of **find enclosed by image object**:image object domain: **image object level**, class filter: **N2**. Right: Result of find enclosed by image object:enclosed objects are classified with the class **enclosed.** Note that the objects at the upper image border are classified as **enclosed**.

Classification Settings

Choose the class that will be used to classify enclosed image objects.

See **classification** algorithm for details.

→ Classification on page 28

3.4.5 Connector

connector

Classify the image objects which connect the current image object with the shortest path to another image object that meets the conditions described by the connection settings.

The process starts from the current image object to search along objects that meet the conditions as specified by **Connect via** and **Super object mode via** until it reaches image objects that meet the conditions specified by **Connect to** and **Super object mode to**. The maximum search range can be specified in **Search range in pixels**. When the algorithm has found the nearest image object that can be connected to it classifies all image objects of the connection with the selected class.

Connector Via

Choose the classes you wish to be connected.

Super Object Mode Via

Limit the shorted path use for **Super object mode via** using one of the following:

Value	Description
Don't Care	Use any image object.
Different Super Object	Use only images with a different superobject than the **Seed** object.
Same Super Object	Use only image objects with the same superobject as the **Seed** object

Connect To

Choose the classes you wish to be connected.

Super Object Mode To

Limit the shorted path use for **Super Object Mode To** using one of the following:

Value	Description
Don't Care	Use any image object.
Different Super Object	Use only images with a different superobject than the **Seed** object.
Same Super Object	Use only image objects with the same superobject as the **Seed** object

Search Range

Enter the **Search Range** in pixels that you wish to search.

Classification Settings

Choose the class that should be used to classify the connecting objects.

See **classification** algorithm for details.

→ Classification on page 28

3.4.6 Optimal Box

optimal box

Generate member functions for classes by looking for the best separating features based upon sample training.

Sample Class

For target samples

Class that provides samples for target class (class to be trained). Select a class or create a new class.

For rest samples

Class that provides samples for the rest of the domain.

Select a class or create a new class.

Insert Membership Function

For target samples into

Class that receives membership functions after optimization for target. If set to unclassified, the target sample class is used.

Select a class or create a new class.

For rest samples into

Class that receives inverted similarity membership functions after optimization for target. If set to unclassified, the rest sample class is used.

Select a class or create a new class.

Clear all membership functions

When inserting new membership functions into the active class, choose whether to clear all existing membership functions or clear only those from input feature space.

Value	Description
No, only clear if associated with input feature space	Clear membership functions only from the input feature space when inserting new membership functions into the active class.
Yes, always clear all membership functions	Clear all membership functions when inserting new membership functions into the active class.

Border membership value

Border y-axis value if no rest sample exists in that feature direction.

Default: **0.66666**

Feature Optimization

Input Feature Set

Input set of descriptors from which a subset will be chosen.

Click the ellipsis button to open the **Select Multiple Features** dialog box and select features by double-clicking in the Available pane to move features to the Selected pane. The **Ensure selected features are in Standard Nearest Neighbor** feature space checkbox is selected by default.


(ellipsis button)

Minimum number of features

Minimum number of features descriptors to employ in class.

Default: **1**

Maximum number of features

Maximum number of features descriptors to employ in class.

Optimization Settings

Weighted distance exponent

0: All distances weighted equally.

X: Decrease weighting with increasing distance.

Enter a number greater than 0 to decrease weighting with increasing distance.

Default: **2**

Optimization Settings

False positives variable

Variable to be set to the number of false positives after execution.

Enter a variable or select one that has already been created. If you enter a new variable, the **Create Variable** dialog will open.

→ **User Guide** chapters: **Use Variables in Rule Sets** and **Create a Variable**

False negatives variable

Variable to be set to the number of false positives after execution.

Enter a variable or select one that has already been created. If you enter a new variable, the **Create Variable** dialog will open.

Show info in message console

Show information on feature evaluations in message console.

3.5 Variables Operation Algorithms

Variable operation algorithms are used to modify the values of variables. They provide different methods to perform computations based on existing variables and image object features and store the result within a variable.

3.5.1 Update Variable

Perform an arithmetic operation on a process variable.

 Update Variable Algorithm

Variable Type

Select **Object, Scene, Feature, Class,** or **Level.**

Variable

Select an existing variable or enter a new name to add a new one. If you have not already created a variable, the **Create** *variable type* **Variable** dialog box will open.

Feature/Class/Level

Select the variable assignment, according to the variable type selected in the **Variable Type** field. This field does not display for **Object** and **Scene** variables.

To select a variable assignment, click in the field and do one of the following depending on the variable type:


(ellipsis button)


Select **Single** Feature (drop-down arrow button)

- For feature variables, use the ellipsis button to open the **Select Single Feature** dialog box and select a feature or create a new feature variable.
- For class variables, use the drop-down arrow to select from existing classes or create a new class.
- For level variables, use the drop-arrow to select from existing levels.
- For object variables, use the drop-arrow to select from existing levels.

Operation

This field displays only for **Object** and **Scene** variables.

Select one of the following arithmetic operations:

Value	Description
=	Assign a value.
+=	Increment by value.
–=	Decrement by value.
*=	Multiply by value.
/=	Divide by value.

Assignment

This field displays only for **Scene** and **Object** variables.

You can assign either by value or by feature. This setting enables or disables the remaining parameters.

Value

This field displays only for **Scene** and **Object** variables.

If you have selected to assign **by value**, you may enter either a value or a variable. To enter text use quotes. The numeric value of the field or the selected variable will be used for the update operation.

Feature

This field displays only for **Scene** and **Object** variables.

If you have chosen to assign **by feature** you can select a single feature. The feature value of the current image object will be used for the update operation.

Comparison Unit

This field displays only for **Scene** and **Object** variables.

If you have chosen to assign **by feature**, and the selected feature has units, then you may select the unit used by the process. If the feature has coordinates, select

Coordinates to provide the position of the object within the original image or **Pixels** to provide the position of the object within the currently used scene.

3.5.2 Compute Statistical Value

Perform a statistical operation on the feature distribution within an image object domain and stores the result in a process variable.


compute statistical value

Variable

Select an existing variable or enter a new name to add a new one. If you have not already created a variable, the **Create Variable** dialog box will open.

Operation

Select one of the following statistical operations:

Value	Description
Number	Count the objects of the currently selected image object domain.
Sum	Return the sum of the feature values from all objects of the selected image object domain.
Maximum	Return the maximum feature value from all objects of the selected image object domain.
Minimum	Return the minimum feature value from all objects of the selected image object domain.
Mean	Return the mean feature value of all objects from the selected image object domain.
Standard Deviation	Return the standard deviation of the feature value from all objects of the selected image object domain.
Median	Return the median feature value from all objects of the selected image object domain.
Quantile	Return the feature value, where a specified percentage of objects from the selected image object domain have a smaller feature value.

Parameter

If you have selected the **quantile** operation specify the percentage threshold [0;100].

Feature

Select the feature that is used to perform the statistical operation.

Precondition: This parameter is not used if you select **number** as your operation.

Unit

If you have selected a feature related operation, and the feature selected supports units, then you may select the unit for the operation.

3.5.3 Apply Parameter Set

apply parameter set

→ **User Guide**: **About Parameter Sets**

Writes the values stored inside a parameter set to into the related variables. For each parameter in the parameter set the algorithm scans for a variable with the same name. If this variable exists, then the value of the variable is updated by the value specified in the parameter set.

Precondition: You must first create at least one parameter set.

Parameter Set Name

Select the name of a parameter set.

3.5.4 Update Parameter Set

update parameter set

→ **User Guide**: **About Parameter Sets**

Writes the values of variable into a parameter set. For each parameter in the parameter set the algorithm scans for a variable with the same name. If this variable exists, then the value of the variable is written to the parameter set.

Precondition: You must first create at least one parameter set.

Parameter Set Name

Select the name of a parameter set.

Tip

Create Parameters

*Parameters are created with the **Manage Parameter Sets** dialog box, which is available on the menu bar under **Process** or on the tool bar.*

3.6 Reshaping Algorithms

Reshaping algorithms modify the shape of existing image objects. They execute operations like merging image objects, splitting them into their subobjects and also sophisticated algorithm supporting a variety of complex object shape transformations.

3.6.1 Remove Objects

remove objects

Merge image objects in the image object domain. Each image object is merged into the neighbor image object with the largest common border.

This algorithm is especially helpful for clutter removal.

3.6.2 Merge Region

merge region

Merge all image objects chosen in the image object domain.

Example

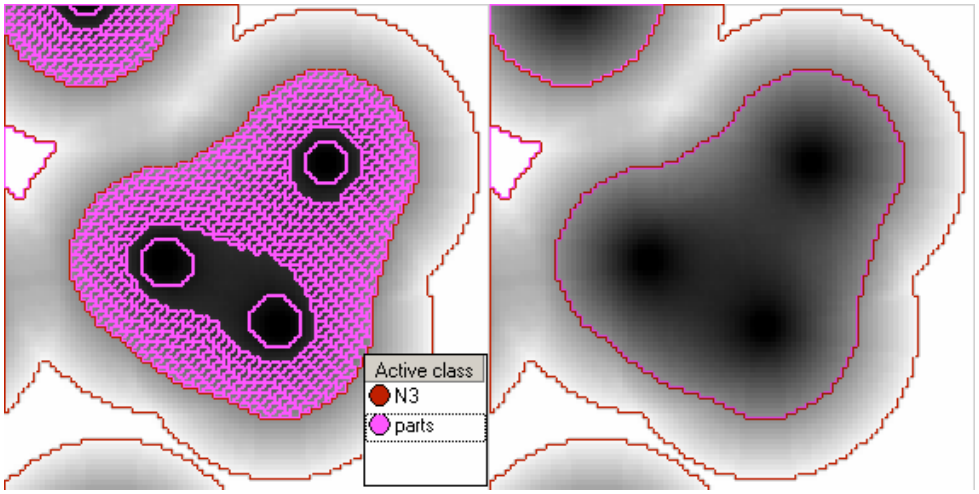


Figure 10: Result of merge region algorithm on all image objects classified as parts.

Fusion Super Objects

Enable the fusion of affiliated super objects.

Use Thematic Layers

Enable to keep borders defined by thematic layers that where active during the initial segmentation of this image object level.

3.6.3 Grow Region

grow region

Enlarge image objects defined in the image object domain by merging them with neighboring image objects ("candidates") that match the criteria specified in the parameters.

→ **Repeat Process Execution** in the **User Guide**

The **grow region** algorithm works in sweeps. That means each execution of the algorithm merges all direct neighboring image objects according to the parameters. To grow image objects into a larger space, you may use the **Loop while something changes** check box or specify a specific number of cycles.

Example

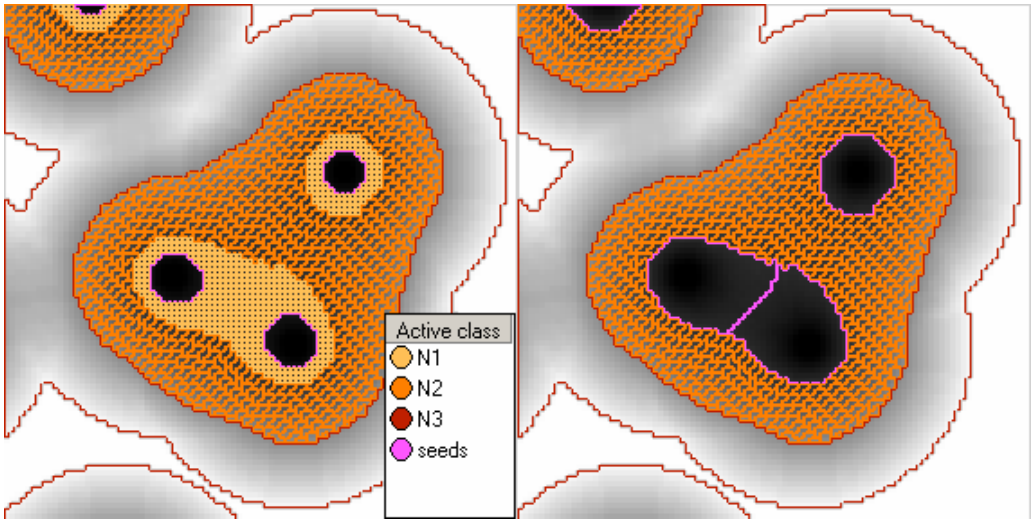


Figure 11: Result of looped grow region algorithm on image objects of class seed and candidate class N1.
Note that the two seed objects in the image center grow to fill the entire space originally covered by objects of class N1 while still being two separate objects.

Candidate Classes

Choose the classes of image objects that can be candidates for growing the image object.

Fusion Super Objects

Enable the fusion of affiliated super objects.

Candidate Condition

Choose an optional feature to define a condition that neighboring image objects need to fulfill in addition to be merged into the current image object.

Use Thematic Layers

Enable to keep borders defined by thematic layers that where active during the initial segmentation of this image object level.

3.6.4 Multiresolution Segmentation Region Grow

Grow image objects according to the multiresolution segmentation criteria.

Precondition: The project must first be segmented by another segmentation process.

For detailed description of all parameters see the algorithm **multiresolution segmentation**.

multiresolution segmentation region grow

→ Multiresolution Segmentation on page 21

3.6.5 Image Object Fusion

Define a variety of growing and merging methods and specify in detail the conditions for merger of the current image object with neighboring objects.

image object fusion

Tip

*If you do not need a fitting functions, we recommend that you use the algorithms **merge region** and **grow regions**. They require fewer parameters for configuration and provide higher performance.*

Image object fusion uses the term **seed** for the current image object. All neighboring image objects of the current image object are potential **candidates** for a fusion (merging). The image object that would result by merging the seed with a candidate is called the **target** image object.

A class filter enables users to restrict the potential candidates by their classification. For each candidate, the fitting function will be calculated. Depending on the fitting mode, one or more candidates will be merged with the seed image object. If no candidate meets all fitting criteria no merge will take.

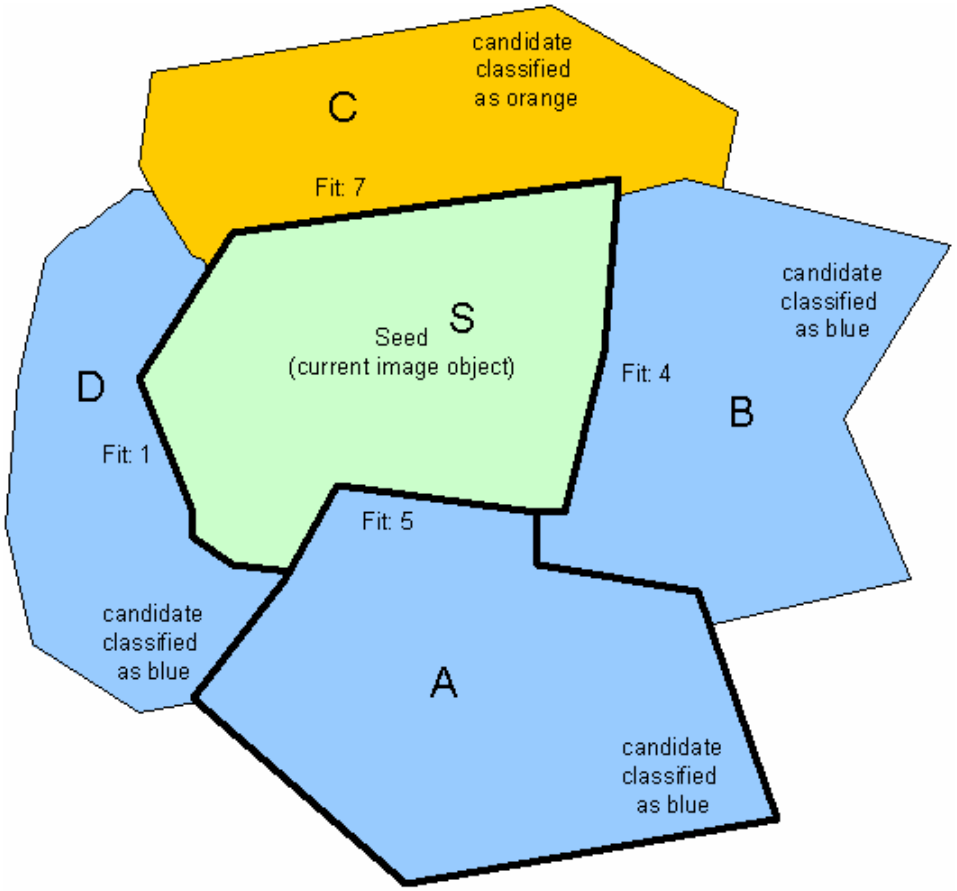


Figure 12: Example for image object fusion with seed image object S and neighboring objects A, B, C and D.

Candidate Settings

Enable Candidate Classes

Select **Yes** to activate candidate classes. If the candidate classes are disabled the algorithm will behave like a region merging.

Candidate Classes

Choose the candidate classes you wish to consider.

If the candidate classes are distinct from the classes in the image object domain (representing the seed classes), the algorithm will behave like a region growing.

Fitting Function

The fusion settings specify the detailed behavior of the image object fusion algorithm.

Fitting Mode

Choose the fitting mode.

Value	Description
all fitting	Merges all candidates that match the fitting criteria with the seed.
first fitting	Merges the first candidate that matches the fitting criteria with the seed.
best fitting	Merges the candidate that matches the fitting criteria in the best way with the seed.
all best fitting	Merges all candidates that match the fitting criteria in the best way with the seed.
best fitting if mutual	Merges the best candidate if it is calculated as the best for both of the two image objects (seed and candidate) of a combination.
mutual best fitting	Executes a mutual best fitting search starting from the seed. The two image objects fitting best for both will be merged. Note: These image objects that are finally merged may not be the seed and one of the original candidate but other image objects with an even better fitting.

Fitting Function Threshold

Select the feature and the condition you want to optimize. The closer a seed candidate pair matches the condition the better the fitting.

Use Absolute Fitting Value

Enable to ignore the sign of the fitting values. All fitting values are treated as positive numbers independent of their sign.

Weighted Sum

Define the fitting function. The fitting function is computed as the weighted sum of feature values. The feature selected in the **Fitting function threshold** will be calculated

for the **seed**, **candidate**, and the **target** image object. The total fitting value will be computed by the formula

Fitting Value = (**Target** * Weight) + (**Seed** * Weight) + (**Candidate** * Weight)

To disable the feature calculation for any of the three objects, set the according weight to 0

Target Value Factor

Set the weight applied to the target in the fitting function.

Seed Value Factor

Set the weight applied to the seed in the fitting function.

Candidate Value Factor

Set the weight applied to the candidate in the fitting function.

Typical Settings (TVF, SVF, CVF)	Description
1,0,0	Optimize condition on the image object resulting from the merge.
0,1,0	Optimize condition on the seed image object.
0,0,1	Optimize condition on the candidate image object.
2,-1,-1	Optimize the change of the feature by the merge.

Merge Settings

Fusion Super Objects

This parameter defines the behaviour if the seed and the candidate objects that are selected for merging have different super objects. If enabled the super objects will be merged with the sub objects. If disabled the merge will be skipped.

Thematic Layers

Specify the thematic layers that are to be considered in addition for segmentation.

Each thematic layer that is used for segmentation will lead to additional splitting of image objects while enabling consistent access to its thematic information. You can segment an image using more than one thematic layer. The results are image objects representing proper intersections between the thematic layers.

Precondition: Thematic layers must be available.

Compatibility Mode

Select **Yes** from the **Value** field to enable compatibility with older software versions (version 3.5 and 4.0). This parameter will be removed with future versions.

Classification Settings

Define a classification to be applied to the merged image objects.

See **classification** algorithm for details.

→ Classification on page 28

3.6.6 Convert to Subobjects

convert to subobjects

Split all image objects of the image object domain into their subobjects.

Precondition: The image objects in the domain need to have subobjects.

3.6.7 Border Optimization

border optimization

Change the image object shape by either adding subobjects from the outer border to the image object or removing subobjects from the inner border from the image object.

Candidate Classes

Choose the classes you wish to consider for the subobjects. Subobjects need to be classified with one of the selected classes to be considered by the border optimization.

Destination

Choose the classes you wish to consider for the neighboring objects of the current image object. To be considered by the **Dilatation**, subobjects need to be part of an image object classified with one of the selected classes. To be considered by the **Erosion** subobjects need to be moveable to an image object classified with one of the selected classes. This parameter has no effect for the **Extraction**.

Operation

Value	Description
Dilatation	Removes all **Candidate** subobjects from its **Destination** superobject inner border and merges them to the neighboring image objects of the current image object.
Erosion	Removes all **Candidate** objects from its **Seed** superobject inner border and merges them to the neighboring image objects of **Destination** domain.
Extraction	Splits an image object by removing all subobjects of the **Candidate** domain from the image objects of **Seed** domain.

Classification Settings

The resulting image objects can be classified.

See **classification** algorithm for details.

→ Classification on page 28

3.6.8 Morphology

morphology

Perform the pixel based binary morphology operations **Opening** or **Closing** on all image objects of an image object domain.

This algorithm refers to image processing techniques based on mathematical morphology.

Operation

Decide between the two basic operations **Opening** or **Closing**. For a first approach, imagine that you may use opening for sanding image objects and closing for coating image objects. Both will result in a smoothed border of the image object:

Open Image Object removes pixels from an image object. Opening is defined as the area of an image object that can completely contain the mask. The area of an image object that cannot contain the mask completely is separated.

Figure 13: Opening operation of the morphology algorithm.

Close Image Object adds surrounding pixels to an image object. Closing is defined as the complementary area to the surrounding area of an image object that can completely contain the mask. The area near an image object that cannot contain completely the mask is filled; thus comparable to coating. Smaller holes inside the area are filled.

Figure 14: Closing operation of the morphology algorithm.

Mask

Define the shape and size of mask you want. The mask is the structuring element, on which the mathematical morphology operation is based.

In the Value field text, the chosen Mask pattern will be represented on one line.

To define the binary mask, click the ellipsis button. The **Edit Mask** dialog box opens.

... (**ellipsis button**)

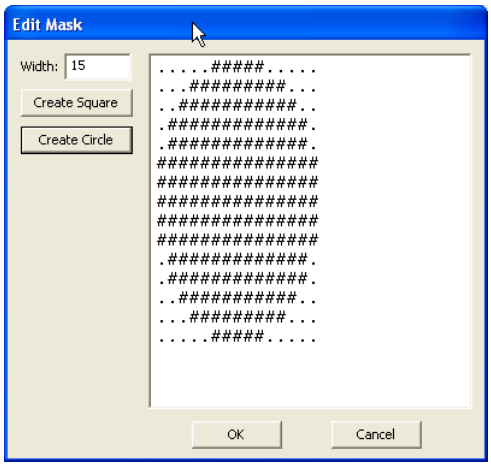


Figure 15: Edit Mark dialog box.

To modify the binary mask you have the following options:

- Change the **Width** of the mask by entering new positive number.
- **Create Square** helps you to create a quadratic mask. Enter the dimensions. Start trying with values similar to the size of areas +1 you want to treat by sanding or to fill by coating.
- **Create Circle** helps you to create a circular mask. Enter the side length. Start trying with values similar to the size of areas +1 you want to treat by sanding or to fill by coating.
- Alternatively, you can directly define a binary mask in the mask text field using
  . for FALSE and
  # for TRUE.

> **Note**
>
> Square masks perform more rough operations and produce fewer artifacts than circle masks do.

Compatibility Mode

Select **Yes** from the **Value** field to enable compatibility with older software versions (version 3.5 and 4.0). This parameter will be removed with future versions.

Classification Settings

When the operation **Open Image Object** is active a classification will be applied to all image objects sanded from the current image object. When using the **Close Image Object** operation, the current image object will be classified if it gets modified by the algorithm.

See **classification** algorithm for details.

→ Classification on page 28

3.6.9 Watershed Transformation

watershed transformation

The **watershed transformation** algorithm calculates an inverted distance map based on the inverted distances for each pixel to the image object border. Afterwards, the minima are flooded by increasing the level (inverted distance). Where the individual catchment basins touch each other (watersheds), the image objects are split.

Example of purpose: The **Watershed Transformation** algorithm is used to separate image objects from others.

Precondition: Image objects that you wish to split should already be identified and classified.

Length Factor

The **Length factor** is the maximal length of a plateau, which is merged into a catchment basin. Use the toggle arrows in the **Value** field to change to maximal length.

> **Note**
>
> The **Length Factor** must be greater or equal to zero.

Classification Settings

Define a classification to be applied if an image object is cut by the algorithm.

See **classification** algorithm for details.

→ Classification on page 28

3.7 Level Operation Algorithms

Level operation algorithms allow you to add, remove or rename2 entire image object levels within the image object hierarchy.

3.7.1 Copy Image Object Level

copy image object level

Insert a copy of the selected image objects domain above or below the existing one.

Level Name

Enter the name for the new image object level.

Copy Level

Level copy maybe placed **above** or **below** the input level specified by the domain.

3.7.2 Delete Image Object Level

Delete the image object level selected in the image object domain.


delete image object level

3.7.3 Rename Image Object Level

Rename an image object level.


rename image object level

Level to Rename

Select the image object level to be renamed.

New Level Name

Select or edit an image object level to be changed, and select or edit the new name for the level.

If the new name is already assigned to an existing level, that level will be deleted.

This algorithm does not change names already existing in the process tree.

3.8 Training Operation Algorithms

Interactive operation algorithms are used for user interaction with the user of actions in **Definiens Architect**.

3.8.1 Show User Warning

Edit and display a user warning.


show user warning

Message

Edit the text of the user warning.

3.8.2 Create/Modify Project

Create a new project or modify an existing one.


create/modify project

→ **User Guide**: **Create a New Project**

Image File

Browse for an image file containing the image layers. Alternatively, you can edit the path.

Image Layer ID

Change the image layer ID within the file. Note, that the ID is zero-based.

Image Layer Alias

Edit the image layer alias.

Thematic File

Browse for a thematic file containing the thematic layers. Alternatively, you can edit the path.

Attribute Table File

Browse for an attribute file containing thematic layer attributes. Alternatively, you can edit the path.

Attribute ID Column Name

Edit the name of the column of the attribute table containing the thematic layer attributes of interest.

Thematic Layer Alias

Edit the thematic layer alias.

Show Subset Selection

Opens the **Subset Selection** dialog box when executed interactively.

Enable Geocoding

Activate to select the bounding coordinates based on the respective geographical coordinate system.

3.8.3 Update Action from Parameter Set

Synchronize the values of an action according to the values of a parameter set.

update action from parameter set

Parameter Set Name

Select the name of a parameter set.

Action Name

Type the name of an action.

3.8.4 Update Parameter Set from Action

Synchronize the values of a parameter set according to the values of an action.


update parameter set from action

Action Name

Type the name of an action.

Parameter Set Name

Select the name of a parameter set.

3.8.5 Manual Classification

Enable the user of an action to classify image objects of the selected class manually by clicking.


manual classification

→ **User Guide**: **Classify Image Objects Manually**

Class

Select a class that can be assigned manually.

3.8.6 Configure Object Table

Display a list of all image objects together with selected feature values in the **Image Object Table** window .


configure object table

→ **User Guide**: **Compare Multiple Image Objects by Using the Image Object Table**

Classes

Select classes to list all of its image objects.

Features

Select the features to display the feature values of the image objects.

3.8.7 Display Image Object Level

Display a selected image object level.


display image object level

→ **User Guide**: **Navigate Within the Image Object Hierarchy**

Level Name

Select the image object level to be displayed.

3.8.8 Select Input Mode

Set the mode for user input via graphical user interface.

select input mode

Input Mode

Select an input mode:

Value	Description
Normal	Return to normal input mode, for example, selection of image objects by clicking them.
Manual object cut	Activate the **Cut Objects Manually** function.

3.8.9 Activate Draw Polygons

Use the **activate draw polygons** algorithm to activate thematic editing, creates thematic layer and enable the cursor for drawing. It is designed to be used with actions.

▪ **activate draw polygons**

Layer Name

Select the name of the image layer where the polygons will be enabled.

Cursor Actions Available After Execution

- Click and hold the left mouse button as you drag the cursor across the image to create a path with points in the image.
- To create points at closer intervals, drag the cursor more slowly to create points at closer intervals or hold the control key while dragging.
- Release the mouse button to automatically close the polygon.
- Click along a path in the image to create points at each click. To close the polygon, double-click or select **Close Polygon** in the context menu.
- To delete the last point before the polygon is complete, select **Delete Last Point** in the context menu.

3.8.10 Select Thematic Objects

Use the **select thematic objects** algorithm to enable selection of thematic objects in the user interface. The algorithm activates thematic editing and enables cursor selection mode. It is designed to be used with actions.

▪ **select thematic objects**

Layer Name

Enter the name of the layer where thematic objects are to be selected.

Selection Mode

Choose the type of selection:

- **Single:** enables selection of single polygons.
- **Polygon:** enables selection of all shapes within a user-drawn polygon.
- **Line:** enables selection of all shapes crossed by a user-drawn line.
- **Rectangle:** enables selection of all shapes within a user-drawn rectangle.

Cursor Actions After Execution

Depending on the Selection Mode, you can select polygons in the following ways. Selected polygons will be outlined in red. After making a selection, delete any selected polygons using the context menu or press Del on the keyboard.

- **Single:** Click on a polygon to select it.
- **Polygon:** Left-click and drag around polygons. When the polygon is closed, any enclosed polygons will be selected.
- **Line:** Left-click and drag in a line across polygons.
- **Rectangle:** Draw a rectangle around polygons to select them.

3.8.11 End Thematic Edit Mode

end thematic edit mode

Use the **end thematic edit mode** algorithm to switch back from thematic editing to image object editing and save the shape file. It is designed to be used with actions.

Shapes File

Enter the name of the shape file.

3.9 Vectorization Algorithms

Tip

Vectorization algorithms available in earlier versions have been removed because polygons are available automatically for any segmented image. You can use the algorithm parameters in the ***set rule set options*** *algorithm to change the way polygons are formed.*

➔ ***Set Rule Set Options*** *on page 13*

3.10 Sample Operation Algorithms

Use sample operation algorithms to perform sample operations.

3.10.1 Classified Image Objects to Samples

classified image objects to samples

Create a sample for each classified image object in the image object domain.

3.10.2 Cleanup Redundant Samples

Remove all samples with membership values higher than the membership threshold.

cleanup redundant samples

Membership Threshold

You can modify the default value which is 0.9.

Note

This algorithm might produce different results each time it will be executed. This is because the order of sample deletion is random.

3.10.3 Nearest Neighbor Configuration

Select classes, features and function slope to use for nearest neighbor classification.

nearest neighbor configuration

Active classes

Choose the classes you wish to use for nearest neighbor classification.

NN Feature Space

Select as many features as you like for the nearest neighbor feature space.

Function Slope

Enter the function slope for the nearest neighbor.

3.10.4 Delete All Samples

Delete all samples.

delete all samples

3.10.5 Delete Samples of Class

Delete all samples of certain classes.

delete samples of class

Class List

Select the classes for which samples are to be deleted.

3.10.6 Disconnect All Samples

Disconnect samples from image objects, to enable creation of samples that are not lost when image objects are deleted. They are stored in the solution file.

disconnect all samples

This algorithm has no parameters.

3.10.7 Sample Selection

Use the sample selection algorithm to switch the curser to sample selection mode using the selected class.

• **sample selection**

Class

Choose a class to use in selecting samples.

3.11 Image Layer Operation Algorithms

Image layer operation algorithms are used to create or to delete image object layers. Further you can use the image layer operation algorithms to apply filters to image layers at the pixel level.

→ Apply Pixel Filters with Image Layer Operation Algorithms on page 66

3.11.1 Create Temporary Image Layer

Create a temporary image layer with values calculated from a selected feature for the image objects selected in the image object domain.

create temporary image layer

Layer Name

Select the default name for the temporary image layer or edit it.

Feature

Select a single feature that is used to compute the pixel values filled into the new temporary layer.

3.11.2 Delete Image Layer

Delete one selected image layer.

delete image layer

Tip

*This algorithm is often used in conjunction with the **create temporary image layer** algorithm to remove this image layer after you finished working with it.*

Layer to be Deleted

Select one image layer to be deleted.

3.11.3 Convolution Filter

- **convolution filter**

The **convolution filter** algorithm applies a convolution filter to the image. It offers two options; a preset Gaussian smoothing filter and a user-defined kernel.

A convolution filter uses a kernel, which is a square matrix of a values that is applied to the image pixels. Each pixel value is replaced by the average of the square area of the matrix centered on the pixel.

Type

The **Gauss Blur** is a convolution operator used to remove noise and detail.

The **Custom Kernel** enables the user to construct a kernel with customized values.

Advanced Parameter

Displays for **Gauss Blur.**

Enter a value for the reduction factor of the standard deviation. A higher value results in more blur.

Custom Kernel

Displays only when **Custom Kernel** is selected.

Click the ellipsis button on the right to open the **Kernel** dialog box and enter the numbers for the kernel.


(ellipsis button)

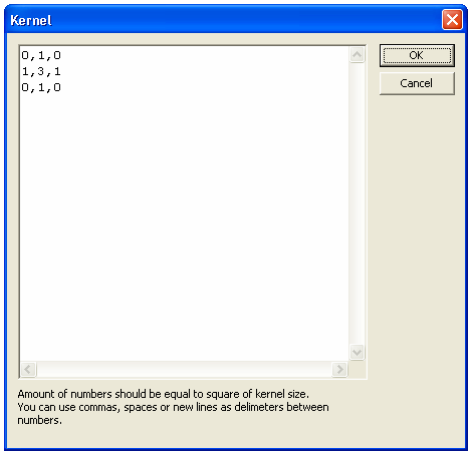


Figure 16: Kernel dialog box.

The number of entries should equal the square of the kernel size entered in the **2D kernel size** field. Use commas, spaces or lines to separate the values.

2D Kernel Size

Default: **3**

Enter an odd number only for the filter kernel size.

Input Layer

Select a layer to be used as input for filter.

Output Layer

Enter a layer name to be used for output. A temporary layer will be created if there is no entry in the field or if the entry does not exist.

Caution

If an existing layer is selected it will be deleted and replaced.

Output Layer Type

Select an output layer type from the drop-down list.

Select **as input layer** to assign the type of the input layer to the output layer.

Formulas

Gauss Blur

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Figure 17: Gauss blur formula.

where σ is the standard deviation of the distribution.

3.11.4 Layer Normalization

- **layer normalization**

The **layer normalization** algorithm offers two options to normalize images. The linear normalization filter stretches pixel values to the entire pixel value range. The histogram

normalization changes pixel values based on the accumulated histogram of the image. The general effect is illustrated in the histograms below.

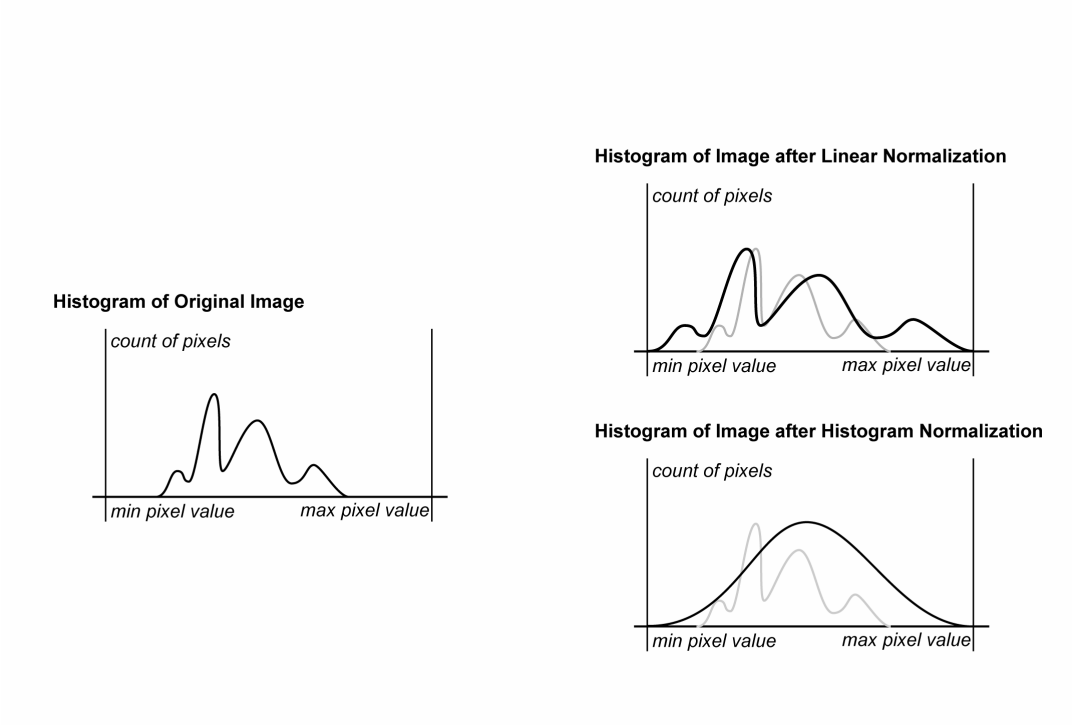


Figure 18: Example histogram changes after normalization.

Type

Value	Description
Linear	Applies a linear stretch to the layer histogram.
Histogram	Applies a histogram stretch to the layer histogram.

Input Layer

Select a layer to be used as input for filter.

Output Layer

Enter a layer name to be used for output. If left empty, a temporary layer will be created.

Caution

If an existing layer is selected it will be deleted and replaced.

3.11.5 Median Filter

- **median filter**

Use the **median filter** algorithm to replace the pixel value with the median value of neighboring pixels.

The median filter may preserve image detail better than a mean filter. Both can be used to reduce noise.

2D Kernel Size

Enter a number to set the kernel size in one slice.

Default: **3**

Input Layer

Use the drop-down list to select a layer to be used as input for filter.

Output Layer

Enter a name for the or use the drop-down list to select a layer name to be used for output. If left empty, a temporary layer will be created.

Caution 

If an existing layer is selected it will be deleted and replaced.

3.11.6 Pixel Frequency Filter

- **pixel frequency filter**

The **pixel frequency filter** algorithm scans the input layer and select the color that is found in the greatest number of pixels. The frequency is checked in the area defined by the size of the kernel.

2D Kernel Size

Enter a number to set the kernel size.

Default: **3**

Input Layer

Select a layer to be used as input for filter.

Output Layer

Enter a layer name to be used for output. If left empty, a layer will be created.

Caution 

If an existing layer is selected it will be deleted and replaced.

3.11.7 Edge Extraction Lee Sigma

- **edge extraction lee sigma**

Use the **edge extraction lee sigma** algorithm to extract edges. This is a specific edge filter that creates two individual layers from the original image. One layer represents bright edges, the other one dark edges.

To extract two layers, one with bright, one with dark edges, this algorithm must be applied two times with the appropriate settings changed.

If two edge layers are created, it is important to give them two individual image layer aliases. Otherwise, the first existing layer would be overwritten by the second generated layer.

Sigma

Set the Sigma value.
The Sigma value describes how far away a data point is from it's mean, in standard deviations. A higher Sigma value results in a stronger edge detection.

Default: **5**

Edge Extraction Mode

Value	Description
Dark	Extract edges of darker objects.
Bright	Extract edges of brighter objects.

Input Layer

Use the drop-down list to select the input layer.

Output Layer

Enter a name for the output layer or use the drop-down box to select a layer.

Formula

For a given window, the sigma value is computed as:

$$\Sigma = \sqrt{\frac{\sigma^2}{(\bar{x})^2}}$$

Figure 19: Sigma value, Lee Sigma preprocessing algorithm.

If the number of pixels P within the moving window that satisfy the criteria in the formula below is sufficiently large (where W is the width, a user-defined constant), the average of these pixels is output. Otherwise, the average of the entire window is produced.

$$(1 - W\Sigma)P_{Center} \leq P \leq (1 + W\Sigma)P_{Center}$$

Figure 20: Moving window criteria for Lee Sigma edge extraction.

3.11.8 Edge Extraction Canny

- **edge extraction canny**

Use the **edge extraction canny** filter algorithm to enhance or extract feature boundaries, using Canny's algorithm. Edge extraction filters may be used to enhance or extract feature boundaries. The resulting layer typically shows high pixel values where there is a distinctive change of pixel values in the original image layer.

Algorithm

The **Canny Algorithm** is provided.

Lower Threshold

Lower Threshold is applied after **Higher Threshold**. During the first step, edges are detected and pixels with values lower than **Higher Threshold** are removed from detected edges. During the final step, non-edge pixels (those previously removed because values were less than **Higher Threshold**) with values higher than **Lower Threshold** are marked as edge nodes again.

After applying the algorithm the first time, you can check results (edge pixel values) and the value for the threshold.

Usually values for this field are from 0.0 to 5.0.

Default: **0**

Higher Threshold

After edges are detected, pixels with values lower than this threshold will not be marked as edge pixels. This allows removal of low intensity gradient edges from results.

After applying the algorithm once, users can check the results (values of edge pixels) and find the correct value for the threshold.

Usually values for this field are from 0.0 to 5.0.

Default: **0**

Gauss Convolution FWHM

Enter the width of the Gaussian filter in relation to full width at half maximum of the Gaussian filter. This field determines the level of details covered by Gaussian filter. A higher value will produce a wider Gaussian filter and less detail will remain for edge detection. Thus, only high intensity gradient edges will be detected by Canny's algorithm.

Range of the field is 0.0001 till 15.0.

Default: **1.0**

Input Layer

Use the drop-down list to select a layer to use for input.

Output Layer

Use the drop-down list to select a layer to use for output or enter a new name. Output is 32 Bit float. If the name of an existing 32Bit float temporary layer is entered or selected, it

will be used. If there is an existing temporary layer with a matching name but of a different type, it will be recreated.

Sample Results

Original Layer	Lower Threshold: 0 Higher Threshold: 0 Gauss Convolution FWHM: 0.2	Lower Threshold: 0.3 Higher Threshold: 0.6 Gauss Convolution FWHM: 0.2	Lower Threshold: 0.3 Higher Threshold: 0.69 Gauss Convolution FWHM: 0.2
			

3.11.9 Surface Calculation

- surface calculation

Use the **surface calculation** algorithm to derive the slope for each pixel of a digital elevation model (DEM). This can be used to determine whether an area within a landscape is flat or steep and is independent from the absolute height values.

There is also an option to calculate aspect using Horn's Method.

Layer

Select the layer to which the filter will be applied.

Algorithm

Value	Description
Slope Zevenbergen, Thorne (ERDAS)	Uses the Zevenbergen Thorne method to calculate slope. See: Quantitative analysis of land surface topography. Zevenbergen, L W; Thorne, C R Earth Surface Processes and Landforms [EARTH SURF. PROCESS. LANDFORMS.], vol. 12, no. 1, pp. 47-56, 1987 #
Aspect (Horn's Method)	Uses Horn's Method to calculate aspect. See: Horn, B. K. P. (1981). Hill Shading and the Reflectance Map, Proceedings of the IEEE, 69(1):14-47.

Gradient Unit

Available for slope.

Select **Percent** or **Degree** from the drop-down list for the gradient unit.

Unit of Pixel Values

Enter the ratio of the pixel height to pixel size.

Input Layer

Use the drop-down list to select a layer for input.

Output Layer

Select a layer for output or enter a new name.

3.11.10 Layer Arithmetics

- layer arithmetics

The **layer arithmetic** algorithm uses a pixel-based operation that enables the merger of up to four layers by mathematical operations (**+ – * /**). The layer created displays the result of this mathematical operation. This operation is performed on the pixel level which means, that all pixels of the image layers are used.

For example Layer 2 can be subtracted from Layer 1. This would mean that whenever the same pixel value in both layers exist, the result would be 0.

Before or after the operation, the layers can be normalized. Furthermore, weights can be used for each individual layer to influence the result.

Layer Name

Select or enter a raster layer name to which the filter will be applied. A layer will be created if the entry does not match an existing layer.

Output Layer Data Type

Select a data type for the raster channel if it must be created:

- **float**
- **int 8bit**
- **int 16bit**
- **int 32bit**

Minimum Input Value

Enter the lowest value that will be replaced by the output value.

Default: **0**

Maximum Input Value

Enter the lowest value that will be replaced by the output value.

Default: **255**

Output Value

The value that will be written in the raster layer. May be a number or an expression. For example, to add Layer 1 and Layer 2, enter **Layer 1 + Layer 2**.

3.11.11 Line Extraction

The **line extraction** algorithm creates a layer and classifies the pixels of the input layer according to their line filter signal strength.

- **line extraction**

Line Direction

Enter the direction of the extracted line in degrees, between 0 and 179.

Default: **0**

Line Length

Enter the length of the extracted line.

Default: **12**

Line Width

Enter the length of the extracted line.

Default: **4**

Border Width

Enter the width of the homogeneous border at the side of the extracted line.

Default: **4**

Max Similarity of Line to Border

Enter a value to specify the similarity of lines to borders.

Default: **0.9**

Min. Pixel Variance

Enter a value to specify the similarity of lines to borders.

Use **-1** to use the variance of the input layer.

Default: **0**

Min. Mean Difference

Enter a value for the minimum mean difference of the line pixels to the border pixels. If positive, bright lines are detected. Use **0** to detect bright and dark lines.

Input Layer

Use the drop-down list to select the layer where lines are to be extracted.

Output Layer

Enter or select a layer where the maximal line signal strength will be written.

3.11.12 Apply Pixel Filters with Image Layer Operation Algorithms

Use the **Image Layer Operation** algorithms to apply filters to image layers at the pixel level.

Before digital images are analyzed, preprocessing is usually a necessary step for optimal results, and typically includes radiometric correction.

In addition, filter techniques can be applied which may improve the quality of the extracted information.

In **Definiens Developer** the term preprocessing refers to the application of filters (such as Gaussian, edge detection, or slope calculation from a digital elevation model).

Image layer operation algorithms can be applied on an existing image layer or a combination of existing image layers, which means, the existing layers are used as the basis for the new layer to be created.

The identifier for a layer created by an image layer operation algorithm is recognized in the same way a physical layer is recognized; by its image layer alias. If a layer exists and any algorithm is programmed to create a layer with an existing alias, this existing layer is overwritten by the newly generated layer. Physical layers can not be overwritten with the described procedure.

The result is one or more new raster layers, which are generated by the algorithm. The newly generated layers can be accessed in the project in the same way as other image layers, which means that all features related to image layers can be applied.

To avoid excessive hard disk use, preprocessed layers are only available temporarily. When other temporary layers are no longer needed, they can be deleted with the **delete image layer** algorithm.

Preprocessed layers are typically used within the segmentation process or for the classification process and can be used to improved the quality of the information extraction.

The key to the use of preprocessed layers is to be clear when they might be useful in a segmentation or classification step and when they would not. For example, if a multiresolution segmentation is executed with the goal of extracting a certain feature that is mainly distinguished by its spectral properties, the preprocessed layer should not be used in this step, because the image layer properties would influence the image object primitives. In this situation, the preprocessed layer might be used in the classification step, where it could help distinguish two features with similar spectral properties.

3.12 Thematic Layer Operation Algorithms

Thematic layer operation algorithms are used to transfer data from thematic layers to image objects and vice versa.

3.12.1 Synchronize Image Object Hierarchy

synchronize image object hierarchy

Change an image object level to exactly represent the thematic layer. Image objects smaller than the overlapping thematic object will be merged, image objects intersecting with several thematic objects will be cut.

Thematic Layers

Select the **Thematic layers** for the algorithm.

3.12.2 Read Thematic Attributes

read thematic attributes

Create and assign local image object variables according to a thematic layer attribute table. A variable with the same name as the thematic attribute will be created, attached to each image object in the domain and filled with the value given by the attribute table.

Thematic Layer

Select the **Thematic layer** for the algorithm.

Thematic Layer Attributes

Choose attributes from the thematic layer for the algorithm. You can select any numeric attribute from the attribute table of the selected thematic layer.

3.12.3 Write Thematic Attributes

write thematic attributes

Generate a attribute column entry from an image object feature. The updated attribute table can be saved to a **.shp** file.

Thematic Layer

Select the **Thematic layers** for the algorithm.

Feature

Select the feature for the algorithm.

Save Changes to File

If the thematic layer is linked with a shape file the changes can be updated to the file.

3.13 Export Algorithms

Export algorithms are used to export table data, vector data and images derived from the image analysis results.

3.13.1 Export Classification View

Export the classification view to a raster file.

 **export classification view**

Export Item Name

Use default name or edit it.

Export Unclassified as Transparent

Activate to export unclassified image objects as transparent pixels.

Enable Geo Information

Activate to add geographic information.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

3.13.2 Export Current View

Export the current project view to a raster file.

 **export current view**

Export Item Name

Use default name or edit it.

Enable GEO Information

Activate to add GEO information.

Save Current View Settings

Click the ellipsis button to capture current view settings. Transparency settings may affect the appearance of the exported view as explained in the note following.

 (ellipsis button)

Note

Projects created with prior versions of **Definiens Developer** will display with the current transparency settings. If you want to use the **export current view** algorithm and preserve the current transparency settings, access the **Algorithm parameters.** Then select **Click to capture current view settings** in the **Save current view settings** field. If you want to preserve the original transparency settings, do not select **Click to capture current view settings.**

Scale

1. If you do not want to keep the current scale of the scene for the copy, click the ellipsis button to open the **Select Scale** dialog box.


(ellipsis button)

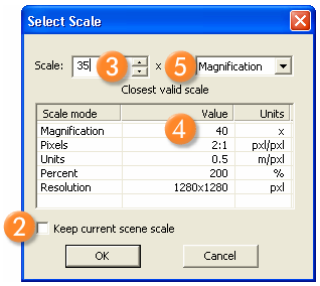


Figure 21: Select Scale dialog box.

2. To keep the scale of the scene for the current view to export click **OK**. If you want to change the scale, clear the **Keep current scene scale** check box 2.
3. You can select a different **Scale** 3 compared to the current scene scale. That way you can export the current view at a different magnification/resolution.
4. If you enter an invalid **Scale** factor, it will be changed to the closed valid one as displayed in the table 4 below.
5. To change the current scale mode, select from the drop-down list box 5. We recommend that a scaling method be used consistently within a rule set as the scaling results may differ.

Note

The scaling results may differ depending on the scale mode. Example: If you enter 40, you work at the following scales, which are calculated differently:

Options dialog box setting	Scale of the scene copy or subset to be created
Units (m/pixel)	40m per pixel
Magnification	40x
Percent	40% of the resolution of the source scene
Pixels	1 pxl per 40 pxl of the source scene

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

3.13.3 Export Thematic Raster Files

Export thematic raster files.

export thematic raster files

Export Item Name

Use default name or edit it.

Export Type

Select the type of export:

Value	Description
Image Objects	Export feature values.
Classification	Export classification by unique numbers associated with classes.

Features

Select one or multiple features for exporting their values.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

3.13.4 Export Domain Statistics

Select an image object domain and export statistics regarding selected features to a file.

export domain statistics

Export Item Name

Use default name or edit it.

Features

Select one or multiple features for exporting their values.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

Statistical Operations

Select the statistical operators with a **Yes** or **No** from the drop-down arrow.

- **Number**
- **Sum**
- **Mean**
- **Std. Dev.**
- **Min**
- **Max**

3.13.5 Export Project Statistics

Export values of selected project features to a file.

 **export project statistics**

Export Item Name

Use default name or edit it.

Features

Select one or multiple features for exporting their values.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

3.13.6 Export Object Statistics


export object statistics

Export image object statistics of selected features to file. This generates one file per project.

Export Item Name

Use default name or edit it.

Features

Select one or multiple features for exporting their values.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

3.13.7 Export Object Statistics for Report


export object statistics for report

Export image object statistics to a file. This generates one file per workspace.

Export Item Name

Use the default name or edit it.

Features

Select one or multiple features for exporting their values.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

3.13.8 Export Vector Layers

Export vector layers to file.

 **export vector layers**

Export Name

Use default name or edit it.

Features

Select one or multiple features for exporting their values.

Shape Type

Select a type of shapes for export:

- **Polygons**
- **Lines**
- **Points**

Export Type

Select a type of export:

- **Center of main line**
- **Center of gravity**

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

Write Shape Attr to CSV File

The column width for data in **.dbf** files is 255 characters.

Yes: Save the shape attributes as **.csv** file.

No: Save the shape attributes as **.dbf** file.

3.13.9 Export Image Object View

Export an image file for each image object.

 **export image object view**

Export Item Name

Use default name or edit it.

Border Size Around Object

Add pixels around the bounding box of the exported image object. Define the size of this bordering area.

Desktop File Format

Select the export file type used for desktop processing. If the algorithm is run in desktop mode, files will stored in this format. In server processing mode, the file format is defined in the export settings specified in the workspace.

Desktop Export Folder

Specify the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. In server processing mode, the file location is defined in the export settings specified in the workspace.

Save Current View Settings

Click the ellipsis button to capture current view settings.

 (ellipsis button)

3.14 Workspace Automation Algorithms

Workspace automation algorithms are used for working with subroutines of rule sets. These algorithms enable you to automate and accelerate the processing of workspaces with especially of large images. Workspace automation algorithms enable multi-scale workflows, which integrate analysis of images at different scales, magnifications, or resolutions.

3.14.1 Create Scene Copy

Create a scene copy that is a duplicate of a project with image layers and thematic layers, but without any results such as image objects, classes, or variables. This algorithm enables you to use subroutines.

 **create scene copy**

Scene Name

Edit the name of the scene copy to be created.

Scale

1. If you do not want to keep the current scale of the scene for the copy, click the ellipsis button to open the **Select Scale** dialog box.


(ellipsis button)

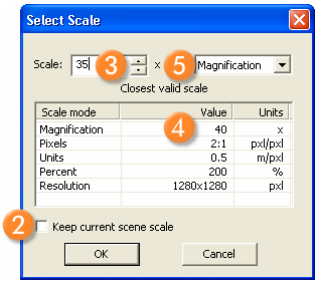


Figure 22: Select Scale dialog box.

2. To keep the scale of the scene for the copy click **OK**. If you want to change the scale, clear the **Keep current scene scale** check box 2.
3. You can select a **Scale** 3 different from the current scene scale, so you can work on the scene copy at a different magnification/resolution.
4. If you enter an invalid **Scale** factor, it will be changed to the closest valid scale as displayed in the table 4 below.
5. To change the current scale mode, select from the drop-down list box 5. We recommend that you use the scaling method consistently within a rule set as the scaling results may differ.

Note

The scaling results may differ depending on the scale mode. Example: If you enter 40, you work at the following scales, which are calculated differently:

Options dialog box setting	Scale of the scene copy or subset to be created
Units (m/pixel)	40m per pixel
Magnification	40x
Percent	40% of the resolution of the source scene
Pixels	1 pxl per 40 pxl of the source scene

Additional Thematic Layers

Edit the thematic layers you wish to load to a scene copy. This option is used to load intermediate result information that has been generated within a previous subroutine and exported to a geocoded thematic layer.

Use semicolons to separate multiple thematic layers, for example, **ThematicLayer1.tif;ThematicLayer2.tif**.

3.14.2 Create Scene Subset


create scene subset

Copy a portion (subset) of a scene as a project with a subset of image layers and thematic layers. The copy does not include results such as image objects, classes, or

variables.

The algorithm uses the given coordinates (geocoding or pixel coordinates) of the source scene. You can create subset copies of an existing subset.

Scene Name

Edit the name of the copy of a scene subset to be created.

Scale

1. If you do not want to keep the current scale of the scene for the copy, click the ellipsis button to open the **Select Scale** dialog box.


(ellipsis button)

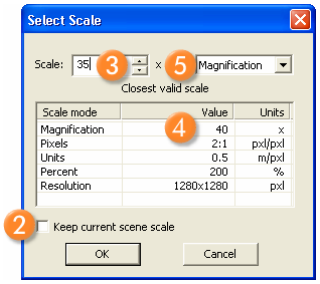


Figure 23: Select Scale dialog box.

2. To keep the scale of the scene for the subset click **OK**.If you want to change the scale, clear the **Keep current scene scale** check box 2.
3. You can select a different **Scale** 3 compared to the current scene scale. That way you can work on the scene subset at a different magnification/resolution.
4. If you enter an invalid **Scale** factor, it will be changed to the closed valid one as displayed in the table 4 below.
5. To change the current scale mode, select from the drop-down list box 5. We recommend that you use the scaling method consistently within a rule set as the scaling results may differ.

Note

The scaling results may differ depending on the scale mode. Example: If you enter 40, you work at the following scales, which are calculated differently:

Options dialog box setting	Scale of the scene copy or subset to be created
Units (m/pixel)	40m per pixel
Magnification	40x
Percent	40% of the resolution of the source scene
Pixels	1 pxl per 40 pxl of the source scene

Additional Thematic Layers

Edit the thematic layers to load to a scene copy. This option is used to load intermediate result information which has been generated within a previous subroutine and exported to a geocoded thematic layer.

Use semicolons to separate multiple thematic layers, for example **ThematicLayer1.tif;ThematicLayer2.tif**.

Define the Cutout

The cutout position is the portion of the scene to be copied.

Depending on the selected **Image Object Domain** of the process you can define the cutout position and size:

- Based on coordinates: If you select **no image object** in the **Image Object Domain** drop down list box, the given coordinates (geocoding or pixel coordinates) of the source scene are used.
- Based on classified image objects: If you select an image object level in the **Image Object Domain** drop down list box you can select classes of image objects. For each image object of the selected classes a subset is created based on a rectangular cutout area around the image object.
  Other image objects of the selected classes are commonly located inside the cutout rectangle, typically near the border. You can choose to include or to exclude them from further processing. Thus, you can extract regions of interest as separate subsets by extracting classified image objects as subset scenes.

Cutout Position Based on Coordinates

Min X Coord, Max X Coord, Min X Coord, Max-Y Coord

Edit the coordinates of the subset.

For the default **Coordinates Orientation** (below) of **(0,0) in Lower left corner** the different coordinates are defined as follows:

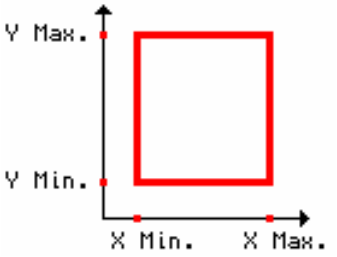


Figure 24: Coordinates of a subset.

The minimum X coordinates describes the left border.

The maximum X coordinates describes the right border.

The minimum Y coordinates describes the lower border.

The maximum Y coordinates describes the upper border.

Alternatively, click the drop-down arrow button to select from available variables. Entering a letter will open the **Create Variable** dialog box.

 (drop-down arrow button)

Coordinates Orientation

You can change the corner of the subset that is used as the calculation base for the coordinates. The default is **(0,0) in Lower left corner**.

Cutout Position Based on Classified Image Objects

Border Size

Edit the size of the border in pixel that is added around the rectangular cutout area around the image objects when creating subsets from.

Exclude Other Image Objects

Commonly it occurs that other image objects of the selected classes are located inside the cutout rectangle, typically near the border. Select **Yes** to exclude them from further processing. For each scene subset, a **.tif** file is created describing the excluded areas as a no-data-mask. The **.tif** file is loaded as additional image layer to each scene subset project.

Desktop Export Folder

If **Exclude Other Image Objects** is selected, you can edit the file export folder used for desktop processing. If the algorithm is run in desktop mode, files will stored at this location. The default **{:Scene.Dir}** is the directory storing the image data.

In server processing mode, the file location is defined in the export settings specified in the workspace.

3.14.3 Create Scene Tiles

create scene tiles

Create a tiled copy of the scene. Each tile is a separate project with its own image layers and thematic layers.

Together the tile projects represent the complete scene as it was before creating the tiled copy. The given coordinates (geocoding or pixel coordinates) of the source scene of the rule set are used. Results are not included before the tiles are processed.

After processing, you can stitch the tile results together and add them to the complete scene within the dimensions as it was before creating the tiled copy.

→ Submit Scenes for Analysis on page 78

You can tile scenes and subsets several times.

Tile Width

Edit the width of the tiles to be created. Minimum with is 100 pixels.

Tile Height

Edit the height of the tiles to be created. Minimum height is 100 pixels.

3.14.4 Submit Scenes for Analysis

submit scenes for analysis

Execute a subroutine.

This algorithm enables you to connect subroutines with any process of the main process tree or other subroutines. You also can also choose whether to stitch the results of the analysis of subset copies.

Type of Scenes

Select the type of scene to submit to analysis: the **Current Scene** itself, **Tiles**, or **Subsets and Copies**.

Scene Name Prefix

Enter the prefix of the names of scene copies to be selected for submitting. A prefix is defined as the complete or the beginning of the scene name. Enter the unique part of the name to select only that scene, or the beginning of the name to select a group with similar or sequential names. For example, if you have scene names 7a, 7b and 7c, you can select them all by entering a **7,** or select one by entering **7a, 7b** or **7c.**

Process Name

Address a subroutine or a process in the process tree of a subroutine for execution by using a slash mark / before hierarchy steps, for example, *subroutine/process name*.

Parameter Set for Processes

Select a parameter set to transfer variables to the following subroutines.

Percent of Tiles to Submit

If you do not want to submit all tiles for processing but only a certain percentage you can edit the percentage of tiles to be processed. If you change the default 100, the tiles are picked randomly. If the calculated number of tiles to be picked is not integer it is rounded up to the next integer.

Stitching Parameters

Stitch Subscenes

Select **Yes** to stitch the results of subscenes together and add them to the complete scene within its original dimensions.

Overlap Handling

If **Subsets and Copies** are stitched, the overlapping must be managed. You can opt to create **Intersection** image objects (default) or select **Union** to merge the overlapping image objects.

Class for Overlap Conflict

Overlapping image objects may have different classifications. In that case, you can define a class to be assigned to the image objects resulting from overlap handling.

Post-Processing

Request Post-Processes

Select Yes to execute another process.

Post-Process Name

Address a subroutine or a process in the process tree of a subroutine for execution by using a slash mark **/** before hierarchy steps, for example, *subroutine/process name*.

Parameter Set for Post-Processes

Select a parameter set to transfer variables to the following subroutines.

3.14.5 Delete Scenes

Delete the scenes you do not want to use or store any more.

Type of Subscenes

Select the type of scene copy to be deleted: **Tiles** or **Subsets and Copies**.

Scene Name Prefix

Enter the prefix of the names of scene copies to be selected for deleting. A prefix is defined as the complete or the beginning of the scene name. Enter the unique part of the name to select only that scene, or the beginning of the name to select a group with similar or sequential names. For example, if you have scene names 7a, 7b and 7c, you can select them all by entering a **7,** or select one by entering **7a, 7b** or **7c.**

3.14.6 Read Subscene Statistics

- **read subscene statistics**

Read in exported result statistics and perform a defined mathematical summary operation. The resulting value is stored as a process variable that can be used for further calculations or export operations concerning the main scene.

This algorithm summarizes all values in the selected column in selected export item, using the selected summary type.

In cases the analysis of subscenes results in exporting statistics per each scene, the algorithm allows you to collect and merge the statistical results of multiple files. The advantage is that you do not need to stitch the subscenes results for result operations concerning the main scene.

Preconditions:

- For each subscene analysis, a project or domain statistic has been exported.
- All preceding subscene analysis including export has been processed completely before the **read subscene statistics** algorithm starts any result summary calculations. To ensure this, result calculations are done within a separate subroutine.

Type of Subscenes

Select the type of scene copy to summarize their results: **Tiles** or **Subsets and Copies**.

Scene Name Prefix

Enter the prefix of the names of scene copies to be selected for reading. A prefix is defined as the complete or the beginning of the scene name. Enter the unique part of the name to select only that scene, or the beginning of the name to select a group with similar or sequential names. For example, if you have scene names 7a, 7b and 7c, you can select them all by entering a **7,** or select one by entering **7a, 7b** or **7c.**

Summary Type

Select the type of summary operation:

- **Mean**: Calculate the average of all values.
- **Sum**: Sum all values of appropriate statistics table columns.
- **Std. Dev.**: Calculates the standard deviation of all values.
- **Min**: Returns the minimal value of all values.
- **Max**: Returns the maximal value of all values.

Export Item

Enter the name of the export item as you defined it in the related exporting process of the subscenes (tiles or subsets).

Column

After defining the **Export Item** above, click the drop-down arrow button to select from the available columns from which values are read used for the summary operation.

 (drop-down arrow button)

Variable

Enter the name of the variable that stores the resulting value of the summary operation.

3.15 Customized Algorithms

Customized algorithms enable you to reuse process sequences several times in one or different rule sets. Based on a developed process sequence, representing the developed code, you can create and reuse your own customized algorithms.

In contrast to duplicating a process, the main advantage of creating customized algorithms, is that when you want to modify the duplicated process you need to

perform the changes to each instance of this process. However, with customized algorithms you only need to modify the customized algorithm template and the changes will take effect to every instance of this algorithm.

Note

Customized algorithms are created within the **Process Tree** window. They **do not** appear within the **Algorithm** drop-down list box in the **Edit Process** dialog box unless you first created them.

→ **User Guide** section: **Reuse Process Sequences with Customized Algorithms**

4 Features Reference

Contents in This Chapter

This **Features Reference** lists all available features in detail.

4.1 About Features as a Source of Information

Image objects have spectral, shape, and hierarchical characteristics. These characteristic attributes are called **Features** in **Definiens** software. Features are used as source of information to define the inclusion-or-exclusion parameters used to classify image objects.

There are two major types of features:

- **Object features** are attributes of image objects, for example the area of an image object.
- **Global features** are not connected to an individual image object, for example the number of image objects of a certain class.

4.2 Basic Features Concepts

Basic features concepts offer an overview on concepts and basic definitions of features.

4.2.1 Image Layer Related Features

4.2.1.1 Scene

Scene is a rectangular area in a 2D space. It has an origin $(x_0, y_0)^{geo}$, an extension **sx** in **x**, and an extension **sy** in **y**. The size of a pixel (in coordinate system unit) is denoted in u_{geo}.

If a scene is geocoded, $(x_0, y_0)^{geo}$ or in geo coordinates (in other words, these values refer to the coordination system defined by the geocoding).

If a scene contains pixel coordinates then (x_0, y_0) is its origin and **sx, sy** is its size (in pixels).

The formula is defined as follows:

$$x^{geo} = x_0^{geo} + x^{pxl} * u_{geo}$$
$$y^{geo} = x_0^{geo} + y^{pxl} * u_{geo}$$

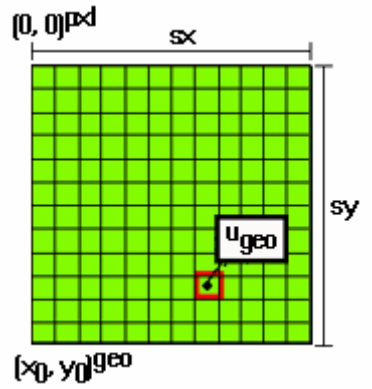


Figure 25: Representation of a scene

Scenes can consist of an arbitrary number of image layers **(k =1,...,K)** and thematic layers **(t=1,...,T)**.

Conversions of Feature Values

The conversion of feature values is handled differently, depending on the kind of values:

- Values identifying a position. These values are called **position values**.
- Values identifying certain distance measurements like **Length** or **Area**. These values are called **unit values**.

Conversion of Position Values

Position values can be converted from one coordinate system to another. The following position conversions are available:

- If the unit is **Pixel**, a position within the pixel coordinate system is identified.
- If the unit is **Coordinate**, a position within the user coordinate system is identified.

→ Pixel Coordinate System on page 93

→ User Coordinate System on page 93

The position conversion is applied for image object features like **Y center**, **Y max**, **X center** and others.

Conversion of Unit Values

Distance values, like **Length**, **Area** and others are initially calculated in pixels. They can be converted to a distance unit.

To convert a pixel value to a unit, the following information is needed:

- Pixel size in meters.
- Value dimension, for example 1 for length, 2 for area and so forth.
- Unit factor, relative to the meter, for example 1 for meter, 100 for centimeter, 0.001 for kilometer and so forth

The following formula is valid for converting value from pixel to a unit:

u: pixel size in meters,

F: unit factor

dim: dimension

$$val_{unit} = val_{pixel} * u^{dim} * \mathbf{F}$$

4.2.1.2 Image Layer

The pixel value—that is the layer intensity—of an image layer **k** at pixel **(x,y)** is denoted as $\mathbf{c}_k\mathbf{(x,y)}$.

The dynamic range of image layers depends on the layer data type. The smallest possible value of an image layer is represented as $\mathbf{c}_k^{min}$, whereas the largest possible value as $\mathbf{c}_k^{max}$. The dynamic range is given by $\mathbf{c}_k^{range} := \mathbf{c}_k^{max} - \mathbf{c}_k^{min}$. The supported layer data types are:

Type	$\mathbf{c}_k^{min}$	$\mathbf{c}_k^{max}$	$\mathbf{c}_k^{range}$
8-bit unsigned (int)	0	255	256
16-bit unsigned (int)	0	65535	65536
16-bit signed (int)	-32767	32767	65535
32-bit unsigned (int)	0	4294967295	4294967296
32-bit signed (int)	-2147483647	2147483647	4294967295
32-bit float	1.17e–38	3.40e+38	n/a

The mean value of all pixels of a layer is computed by:

$$\bar{c}_k := \frac{1}{sx * sy} \sum_{(x,y)} c_k(x,y)$$

The standard deviation of all pixels of a layer is computed by:

$$\sigma_k := \sqrt{\frac{1}{sx * sy}\left(\sum_{(x,y)}(c_k(x,y))^2 - \frac{1}{sx * sy}\sum_{(x,y)} c_k(x,y)\sum_{(x,y)} c_k(x,y)\right)}$$

On raster pixels there are two ways to define the Neighborhood: 4-pixel Neighborhood or 8-pixel Neighborhood.

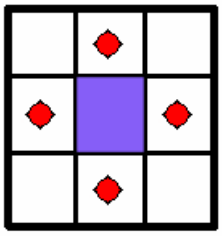


Figure 26: 4-pixel neighborhood

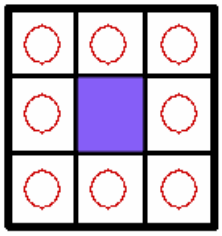


Figure 27: 8-pixel neighborhood

Pixel borders are counted as the number of the elementary pixel border.

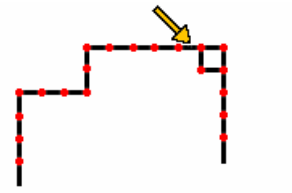


Figure 28: Pixel Borders

4.2.1.3 Image Layer Intensity on Pixel Sets

A fundamental measurement on a pixel set **S** and an image object **v** is the distribution of the layer intensity. First of all the mean intensity within the set is defined by:

$$\bar{c}_k(S) := \frac{1}{\#S} \sum_{(x,y)\in S} c_k(x,y)$$

The standard deviation is defined as:

$$\sigma_k(S) := \sqrt{\frac{1}{\#S}\left(\sum_{(x,y)\in S}(c_k(x,y))^2 - \frac{1}{\#S}\sum_{(x,y)\in S} c_k(x,y) \sum_{(x,y)\in S} c_k(x,y)\right)}$$

An overall intensity measurement is given by the brightness which is the mean value of $\bar{\mathbf{c}}_k\mathbf{(S)}$ for selected image layers.

$$\bar{c}(S) := \frac{1}{w^B} \sum_{k=1}^{K} w_k^B \bar{c}_k(S)$$

If **v** is an image object and **O** a set of other image objects then the mean difference of the objects within **O** to an image object **v** is calculated by:

$$\overline{\Delta}_k(v,O) := \frac{1}{w}\sum_{u\in O} w_u\left(\bar{c}_k(v) - \bar{c}_k(u)\right)$$

4.2.2 Image Object Related Features

4.2.2.1 Image Object Hierarchy

An image object **v** or **u** is a 4-connected set of pixels in the scene. The pixels of an object **v** are denoted by $\mathbf{P}_v$. The image objects are organized in levels **(V**i, **i=1,...,n)** in where each object on each level creates a partition of the scene **S**.

- $\underset{v\in V_i}{U} P_v = \{(x,y)\}$
- $\underset{u,v\in V_i}{\forall} P_u \cap P_v = \varnothing$

The image object levels are hierarchically structured. This means that all image objects on a lower level are complete contained in exactly one image object of a higher level.

$$\underset{i<j}{\forall}\ \underset{v\in V_i}{\forall}\ \underset{u\in V_j}{\exists}\ P_v \subset P_u$$

There are two types of feature distance:

- The **level distance** between image objects on different image object levels in the image object hierarchy.
- The **spatial distance** between objects on the same image object level in the image object hierarchy.

Level Distance

The level distance represents the hierarchical distance between image objects on different levels in the image object hierarchy. Starting from the current image object level, the number in brackets indicates the hierarchical distance of image object levels containing the respective image objects (subobjects or superobjects).

Since each object has exactly 1or 0 superobject on the higher level, the superobject of **v** with a level distance **d** can be denoted as $\mathbf{U_v(d)}$. Similar, all subobjects with a level distance **d** is denoted as $\mathbf{S_v(d)}$.

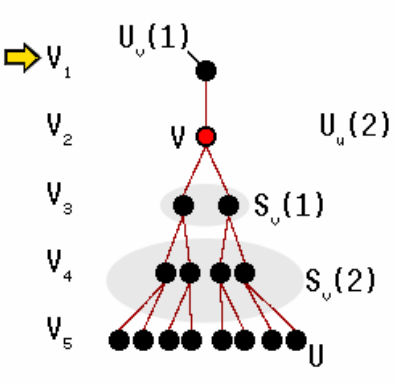


Figure 29: Image object Hierarchy

Two image objects **u** and **v** are considered neighboring each other if this is at least on pixel $\mathbf{(x,y) \in P_v}$ and one pixel $\mathbf{(x',y') \in P_u}$ so that **(x',y')** is part of $\mathbf{N_4(x,y)}$. The set of all image objects neighboring **v** is denoted by $\mathbf{N_v(d)}$.

$$\mathbf{N_v} := \{u \in V_i : \exists (x,y) \in P_v \exists (x',y') \in P_u : (x',y') \in N_4(x,y)\}$$

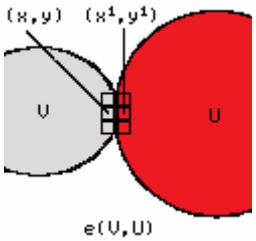


Figure 30: Topological relation between neighbors

The border line between **u** and **v** is called topological relation and it is represented as **e(u,v)**.

Spatial Distance

The spatial distance represents the distance between image objects on the same level in the image object hierarchy.

If you want to analyze neighborhood relations between image objects on the same image object level in the image object hierarchy, the feature distance expresses the spatial distance (in pixels) between the image objects. The default value is **0** (i.e., only neighbors that have a mutual border are regarded). The set of all neighbors within a distance **d** are denoted by $\mathbf{N_v(d)}$.

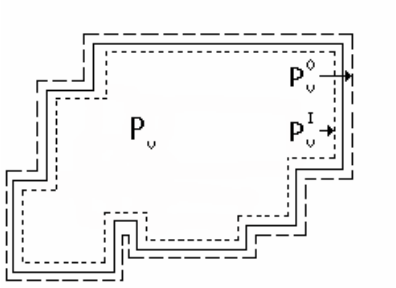


Figure 31: Boundaries of an image object v.

4.2.2.2 Image Object as a Set of Pixels

Image objects are basically pixel sets. The number of pixels belonging to an image object **v** and its pixel set $\mathbf{P_v}$ is denoted by $\mathbf{\#P_v}$.

The set of all pixels in $\mathbf{P_v}$ belonging to the inner border pixels of an object **v** is defined by $\mathbf{P_v}^{Inner}$.

$$\mathbf{P_v}^{Inner} := \{(x,y) \in P_v : \exists (x',y') \in N_4(x,y) : (x',y') \notin P_v\}$$

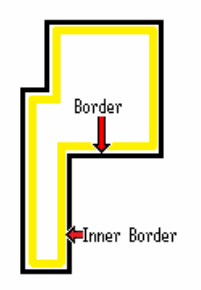


Figure 32: Inner borders of a image object v

The set of all pixels in $\mathbf{P_v}$ belonging to the outer border pixels of an object **v** is defined by $\mathbf{P_v}^{Outer}$.

$$\mathbf{P}_v^{Outer} := \{(x,y) \notin P_v : \exists (x',y') \in N_4(x,y) : (x',y') \in P_v\}$$

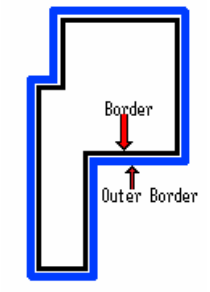


Figure 33: Outer borders of a image object v

4.2.2.3 Bounding Box of an Image Object

The bounding box $\mathbf{B}_v$ of an image object **v** is the smallest rectangular area that encloses all pixels of **v** along **x** and **y** axes.

The bounding box is defined by the minimum and maximum values of the **x** and **y** coordinates of an image object **v** ($\mathbf{x}_{min}(\mathbf{v})$, $\mathbf{x}_{max}(\mathbf{v})$ and $\mathbf{y}_{min}(\mathbf{v})$, $\mathbf{y}_{max}(\mathbf{v})$).

The bounding box $\mathbf{B}_{v}(\mathbf{d})$ can be also extended by a number of pixels.

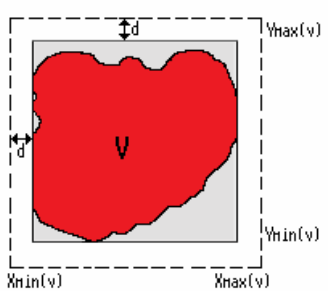


Figure 34: Bounding box of an image object v

Border Length

The border length $\mathbf{b}_v$ of an image object **v** is defined by the number of the elementary pixel borders. Similar, the border length **b(v,u)** of the topological relation between two image objects **v** and **u** is the total number of the elementary pixel borders along the common border.

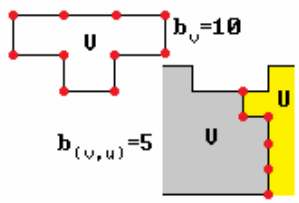


Figure 35: Border length of an image object v or between two objects v, u.

4.2.3 Class-Related Features

4.2.3.1 Class-Related Sets

Let $\mathbf{M=(m_1,...,m_a)}$ be a set of classes with **m** being a specific class and $\mathbf{m \in M}$. Each object has a fuzzy membership value of $\mathbf{\phi(v,m)}$ to class **m**. In addition each image object also carries the stored membership value $\tilde{\phi}(v,m)$ that is computed during the last classification algorithm. By restricting a set of objects **O** to only the image object that belong to class **m** many interesting class related features can be computed:

$N_v(d,m) := \{u \in N_v(d) : \tilde{\phi}(u,m) = 1\}$

$S_v(d,m) := \{u \in S_v(d) : \tilde{\phi}(u,m) = 1\}$

$U_v(d,m) := \{u \in U_v(d) : \tilde{\phi}(u,m) = 1\}$

$V_i(m) := \{u \in V_i(m) : \tilde{\phi}(u,m) = 1\}$

For example, the mean difference of layer **k** to a neighbor object within a distance **d** and that object belongs to a class **m** is defined as $\bar{\Delta}_k(\mathbf{v,N_v(d,m)})$.

4.2.4 Shape-Related Features

> Object Features
> Shape

Many of the form features provided by **Definiens Developer** are based on the statistics of the spatial distribution of the pixels that form an image object. As a central tool to work with these statistics **Definiens Developer** uses the covariance matrix:

Parameters:

- **X** = **x**-coordinates of all pixels forming the image object
- **Y** = **y**-coordinates of all pixels forming the image object

Formula:

$$S = \begin{pmatrix} Var(X) & Cov(XY) \\ Cov(XY) & Var(Y) \end{pmatrix}$$

Another frequently used technique to derive information about the form of image objects (especially length and width) is the bounding box approximation. Such a bounding box can be calculated for each image object and its geometry can be used as a first clue of the image object itself.

The main information provided by the bounding box is its length a, its width b, its area a * b and its degree of filling f, which is the area A covered by the image object divided by the total area a * b of the bounding box.

4.2.4.1 Shape Approximations based on Eigenvalues

This approach measures the statistical distribution of the pixel coordinates **(x,y)** of a set $\mathbf{P_v}$.

$$x_{center} := \frac{1}{\#P_v} \sum_{(x,y)} x$$
$$y_{center} := \frac{1}{\#P_v} \sum_{(x,y)} y$$

and the variances:

$$C_{xx} := \frac{1}{\#P_v} \sum_{(x,y)} x_i^2 - \left(\frac{1}{\#P_v} \sum_{(x,y)} x_i \right)^2 = E_{xx} - E_x^2$$
$$C_{yy} := \frac{1}{\#P_v} \sum_{(x,y)} y_i^2 - \left(\frac{1}{\#P_v} \sum_{(x,y)} y_i \right)^2 = E_{yy} - E_y^2$$
$$C_{xy} := \frac{1}{\#P_v} \sum_{(x,y)} x_i y_i - \left(\frac{1}{\#P_v} \sum_{(x,y)} x_i \right) * \left(\frac{1}{\#P_v} \sum_{(x,y)} y_i \right) = E_{xy} - E_x E_y$$

The eigenvalues of the covariance matrix:

$$\begin{pmatrix} C_{xx} - C_{xy} \\ C_{xy} \; C_{yy} \end{pmatrix}$$

The diagonalization of the pixel coordinates covariance matrix gives two eigenvalues which are the main and minor axis of the ellipse.

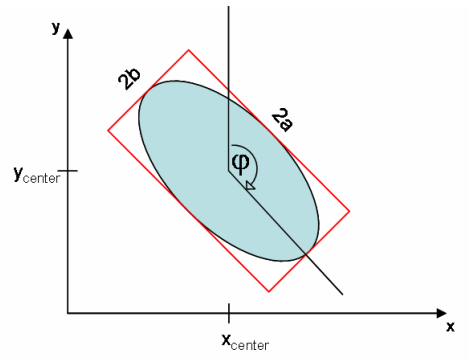


Figure 36: Elliptic approximation

Elliptic Approximation

The elliptic approximation uses the eigenvectors **(λ_1,λ_2)**of the covariance matrix and computes an ellipsis with axis along $\mathbf{e}_1$ and $\mathbf{e}_2$ with length.

$\frac{a}{b} = \frac{\lambda_1}{\lambda_2}$	and a*b* $\bar{n} = \#P_v$

(e.g. the ellipsis with the asymmetry and direction are defined by the CooVar)

The eigenvector of the main axis defines the main direction.

4.2.5 About Coordinate Systems

Definiens software uses different coordinate systems:

- The **pixel coordinate system** is used for identifying pixel positions within the scene.
- The **user coordinate system** allows using geocoding information within the scene.
- The **internal pixel coordinate system** is used only for internal calculations by the Analysis Engine Software.

4.2.5.1 Pixel Coordinate System

The pixel coordinate system is used to identify pixel position within the image. It is used for calculating position features like **X center**, **Y Center** or others in cases where the unit used is pixel.

This coordinate system is oriented from bottom to top and from left to right. The origin position is (0, 0), which is located at the bottom left corner of the image. The coordinate is defined by the offset of the left bottom corner of the pixel from the origin.

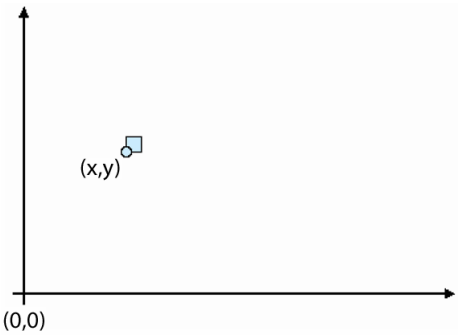


Figure 37: The pixel coordinate system.

4.2.5.2 User Coordinate System

The user coordinate system enables the use of geocoding information within the scene.

The values of the separate user coordinate system are calculated from the pixel coordinate system. In the user interface, the user coordinate system is referred to as **coordinate system**.

This coordinate system is defined by geocoding information:

- Lower Left X position
- Lower Left Y position
- Resolution that is the size of a pixel in coordinate system unit. Examples: If the coordinate system is metric the resolution is the size of a pixel in meters. If the coordinate system is Lat/Long then the resolution is the size of a pixel in degrees.
- Coordinate system name.
- Coordinate system type.

The origin of coordinate system is at the left bottom corner of the image (x_0, y_0). The coordinate defines the position of the left bottom corner of the pixel within user coordinate system.

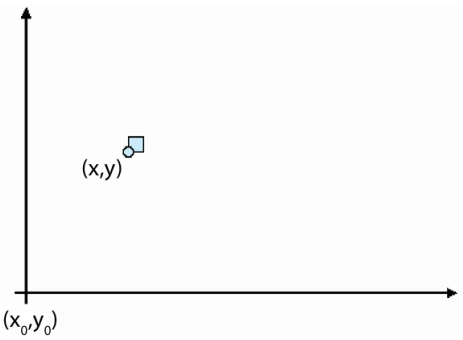


Figure 38: The user coordinate system.

To convert a value from the pixel coordinate system to the user coordinate system and back, the following transformations are valid:

(x, y): coordinates in user coordinate system

U: pixel resolution.

$x = x_0 + x_{pixel} * u$	$x_{pixel} = (x - x_0)/u$
$y = y_0 + y_{pixel} * u$	$y_{pixel} = (y - y_0)/u$

4.2.6 Distance-Related Features

4.2.6.1 Distance Measurements

Many features enable you to enter a spatial distance parameter. Distances are usually measured in pixel units. Because exact distance measurements between image objects are very computing-intensive, Definiens uses approximation approaches to estimate the distance between image objects. There are two different approaches: **center of gravity** and **smallest enclosing rectangle**. You can configure the default distance calculations.

Center of Gravity

The center of gravity approximation measures the distance between the center of gravity between two image objects. This measure can be computed very efficiently but it can be quite inaccurate for large image objects.

Smallest Enclosing Rectangle

The smallest enclosing rectangle approximation tries to correct the center of gravity approximation by using rectangular approximations of the image object to adjust the basic measurement delivered by the center of gravity.

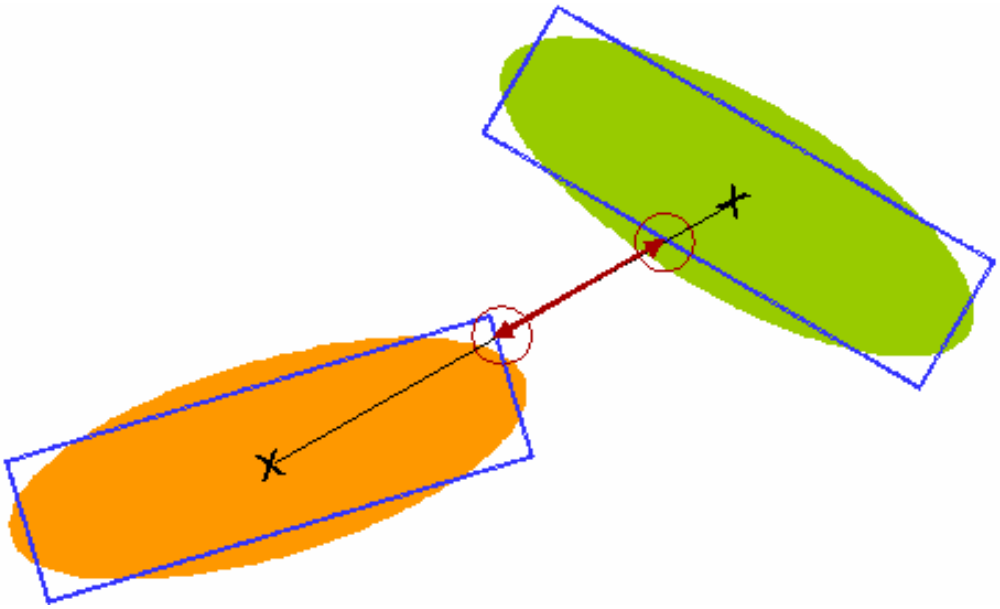


Figure 39: Distance calculation between image objects.
Black line: center of gravity approximation. Red line: Smallest enclosing rectangle approximation.

We recommend use of the center of gravity distance for most applications although the smallest enclosing rectangle may lead to more accurate results. A good strategy for exact distance measurements is to use center of gravity and try to avoid large image objects, for example, by creating border objects. To avoid performance problems, restrict the total number of objects involved in distance calculations to a small number.

You can edit the distance calculation in the **Algorithm parameters** of the **set rule set options** algorithm and set the **Distance Calculation** option to your preferred value.

→ Set Rule Set Options on page 13

4.3 Object Features

Object features are obtained by evaluating image objects themselves as well as their embedding in the image object hierarchy.

> **Object Features**

Object Features are grouped as follows:

- **Customized:** All features created in the **Edit Customized Feature** dialog box referring to object features.
- **Layer Values:** Layer values evaluate the first and second statistical moment (mean and standard deviation) of an image object's pixel value and the object's relations to other image object's pixel values. Use these to describe image objects with information derived from their spectral properties.
- **Shape:** Shape features evaluate the image object's shape in a variety of respects. The basic shape features are calculated based on the object's pixels. Another type of shape features, based on sub-object analysis, is available as a result of the hierarchical structure. If image objects of a certain class stand out because of their shape, you are likely to find a form feature that describes them.
- **Texture:** The image object's texture can be evaluated using different texture features. New types of texture features based on an analysis of sub-objects. These are especially helpful for evaluating highly textured data. Likewise, a large number of features based upon the co-occurrence matrix after Haralick can be utilized.
- **Variables:** Define variables to describe interim values related to variables.
- **Hierarchy:** This feature provides information about the embedding of the image object in the image object hierarchy. These features are best suited for structuring a class hierarchy when you are working with an image object hierarchy consisting of more than one image object level.
- **Thematic Attributes:** If your project contains a thematic layer, the object's thematic properties (taken from the thematic layer) may be evaluated. Depending on the attributes of the thematic layer, a large range of different features become available.

4.3.1 Customized

> Object Features
> **Customized**

[name of a customized feature]

If existing, customized features referring to object features are listed in the feature tree.

> Object Features
> Customized
> **[name of a customized feature]**

4.3.2 Layer Values

> Object Features
> **Layer Values**

4.3.2.1 Mean

> Object Features
> Layer Values
> **Mean**

[name of a layer]

> Object Features
> Layer Values
> Mean
> [name of a layer]

Layer mean value $\bar{c}_k(P_v)$ calculated from the layer values $c_k(x,y)$ of all $\#P_v$ pixels forming an image object.

Parameters:

P_v: set of pixels of an image object **v**
$P_v := \{(x,y) : (x,y) \in v\}$

$\#P_v$: total number of pixels contained in P_v

$c_k(x,y)$: image layer value at pixel **(x,y)**

c_k^{min}: darkest possible intensity value of layer **k**

c_k^{max} : brightest possible intensity value of layer **k**

$\bar{c}_k$: mean intensity of layer **k**

Formula:

$$\bar{c}_k(v) := \bar{c}_k(P_v) = \frac{1}{\#P_v} \sum_{(x,y) \in P_v} c_k(x,y)$$

Feature value range:

$$\left[c_k^{min}, c_k^{max} \right]$$

Brightness

> Object Features
> Layer Values
> Mean
> Brightness

Sum of the mean values of the layers containing spectral information divided by their quantity computed for an image object (mean value of the spectral mean values of an image object).

To define which layers provide spectral information, use the **Define Brightness** dialog box. Select **Classification > Advanced Settings > Select Image Layers for Brightness** from the main menu. The **Define Brightness** dialog box opens.

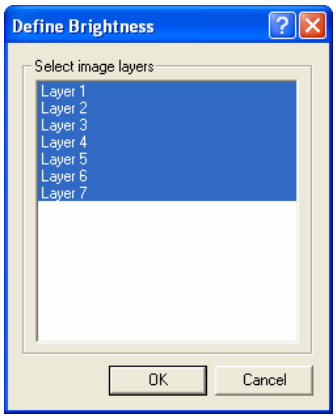


Figure 40: Define Brightness dialog box.

Select image layers and click **OK**.

Because combined negative and positive data values would create an erroneous value for brightness, this feature is only calculated with layers of positive values.

Parameters:

$\mathbf{w_k^B}$: brightness weight of layer **k**

$\bar{\mathbf{c}}_\mathbf{k}$**(v)**: mean intensity of layer **k** of an image object **v**

$\mathbf{c_k^{min}}$: darkest possible intensity value of layer **k**

$\mathbf{c_k^{max}}$: brightest possible intensity value of layer **k**

$$w_k^B := \begin{cases} 0 \\ 1 \end{cases}$$

$$w^B := \sum_{k=1}^{K} w_k^B$$

Formula:

$$\bar{c}(v) := \frac{1}{w^B} \sum_{k=1}^{K} w_k^B \bar{c}_k(v)$$

Feature value range:

$$\left[c_k^{min}, c_k^{max} \right]$$

Condition:

Feature available only for scenes with more than one layer.

Max. diff.

To calculate Max Diff. the minimum mean value belonging to an object is subtracted from its maximum value. To get the maximum and minimum value the means of all layers belonging to an object are compared with each other. Subsequently the result is divided by the brightness.

> Object Features
> Layer Values
> Mean
> **Max. diff.**

Parameters:

i, j : image layers

$\bar{\mathbf{c}}$**(v)** : brightness

$\bar{\mathbf{c}}_\mathbf{i}$**(v)** : mean intensity of layer **i**

$\bar{\mathbf{c}}_\mathbf{j}$**(v)** : mean intensity of layer **j**

$\mathbf{c_k^{max}}$: brightest possible intensity value of layer **k**

$\mathbf{K_B}$: layers with positive brightness weight
$\mathbf{K_B}$:={k ∈ K : w_k=1}, w_k:layer weight

Formula:

$$\frac{\max_{i,j \in K_B} \left| \bar{c}_i(v) - \bar{c}_j(v) \right|}{\bar{c}(v)}$$

Feature value range:

$$\left[0, \frac{1}{K_B} c_k^{\max} \right]$$

Normally, the values given to this range are between 0 and 1.

Conditions:

Feature available only for scenes with more than one layer.

If $\bar{c}(v)=0$ then the formula is undefined.

4.3.2.2 Standard Deviation

> Object Features
> Layer Values
> **Standard Deviation**

[name of a layer]

Standard deviation calculated from the layer values of all n pixels forming an image object.

> Object Features
> Layer Values
> Standard Deviation
> **[name of a layer]**

Parameters:

$\sigma_k(v)$: standard deviation of layer **k** of an image object **v**

P_v: set of pixels of an image object **v**

$\#P_v$: total number of pixels contained in P_v

$c_k(x,y)$: image layer value at pixel **(x,y)**

(x,y) : pixel coordinates

c_k^{range} : data range of layer **k**
$c_k^{range} := c_k^{max} - c_k^{min}$

Formula:

$$\sigma_k(v) := \sigma_k(P_v) = \sqrt{\frac{1}{\#P_v}\left(\sum_{(x,y)\in P_v} c_k^2(x,y) - \frac{1}{\#P_v} \sum_{(x,y)\in P_v} c_k(x,y) \sum_{(x,y)\in P_v} c_k(x,y) \right)}$$

Feature value range:

$$\left[0, \frac{1}{2} c_k^{range} \right]$$

4.3.2.3 Pixel Based

Ratio

The ratio of layer **k** reflects the amount that layer **k** contributes to the total brightness.

Parameters:

$\mathbf{w_k^B}$: brightness weight of layer **k**

$\bar{\mathbf{c}}_\mathbf{k}\mathbf{(v)}$: mean intensity of layer **k** of an image object **v**

$\bar{\mathbf{c}}\mathbf{(v)}$: brightness

Formula:

If $\mathbf{w_k^B=1}$ and $\mathbf{c(v)\neq 0}$ then $\frac{\bar{c}_k(v)}{\bar{c}(v)}$

If $\mathbf{w_k^B=0}$ or $\bar{\mathbf{c}}\mathbf{(v)=0}$ then the ratio is equal to **0.**

Feature value range:

[0,1]

Conditions:

- For scenes with more than one layer.
- Only layers containing spectral information can be used to achieve reasonable results.
- Since combined negative and positive data values would create an erroneous value for ratio, this feature is only calculated with layers of positive values.

Note

The results get meaningless if the layers have different signed data types.

Min. pixel value

Value of the pixel with the minimum intensity value of the image object.

Parameters:

(x,y): pixel coordinates

$\mathbf{c_k(x,y)}$: image layer value at pixel **(x,y)**

$\mathbf{c_k^{min}}$: darkest possible intensity value of layer **k**

$\mathbf{c_k^{max}}$: brightest possible intensity value of layer **k**

P_v: set of pixels of an image object **v**

Formula:

$$\min_{(x,y)\in Pv} c_k(x,y)$$

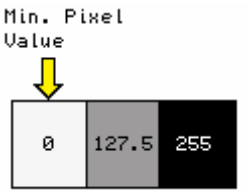


Figure 41: Minimum pixel value of an image object v

Feature value range:

$$\left[c_k^{min}, c_k^{max} \right]$$

Max. pixel value

Value of the pixel with the maximum value of the image object.

> Object Features
> Layer Values
> Pixel Based
> **Max. pixel value**

Parameters:

(x,y): pixel coordinates

c_k(x,y): image layer value at pixel **(x,y)**

c_k^{min}: darkest possible intensity value of layer **k**

c_k^{max} : brightest possible intensity value of layer **k**

P_v: set of pixels of an image object **v**

Formula:

$$\max_{(x,y)\in Pv} c_k(x,y)$$

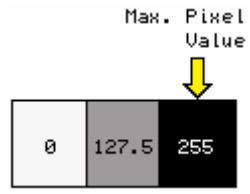


Figure 42: Maximum pixel value of an image object v

Feature value range:

$$\left[c_k^{min}, c_k^{max} \right]$$

Mean of inner border

> Object Features
> Layer Values
> Pixel Based
> **Mean of inner border**

Mean value of the pixels belonging to this image object and sharing their border with some other image object, thereby forming the inner border of the image object.

Parameters:

$\mathbf{P}_v$:set of pixels of an image object **v**

$\mathbf{P}_v^{Inner}$: inner border pixels of $\mathbf{P}_v$
$\mathbf{P}_v^{Inner} := \{(x,y) \in P_v : \exists (x',y') \in N_4(x,y) : (x',y') \notin P_v\}$: Set of inner border pixels of **v**

$\mathbf{c}_k^{min}$: darkest possible intensity value of layer **k**

$\mathbf{c}_k^{max}$: brightest possible intensity value of layer **k**

$\bar{\mathbf{c}}_k$: mean intensity of layer **k**

Formula:

$$\bar{c}_k\left(P_v^{Inner}\right)$$

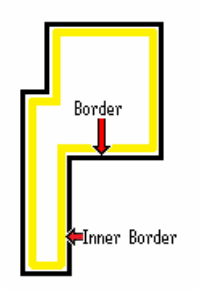


Figure 43: Inner borders of a image object v

Feature value range:

$$\left[c_k^{min}, c_k^{max} \right]$$

Mean of outer border

> Object Features
> Layer Values
> Pixel Based
> **Mean of outer border**

Mean value of the pixels not belonging to this image object, but sharing its border, thereby forming the outer border of the image object.

Parameters:

$\mathbf{P}_v$:set of pixels of an image object **v**

$\mathbf{P}_v^{Outer}$: outer border pixels of $\mathbf{P}_v$
$\mathbf{P}_v^{Outer}$: := $\{(x,y) \notin P_v : \exists (x',y') \in N_4(x,y) : (x',y') \in P_v)\}$: Set of outer border pixels of **v**

$\mathbf{c}_k^{min}$: darkest possible intensity value of layer **k**

$\mathbf{c}_k^{max}$: brightest possible intensity value of layer **k**

$\bar{\mathbf{c}}_k$: mean intensity of layer **k**

Formula:

$$\bar{c}_k\left(P_v^{Outer}\right)$$

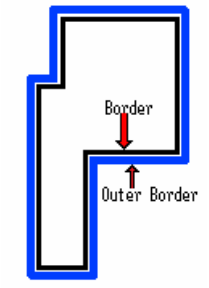


Figure 44: Outer borders of a image object v

Feature value range:

$$\left[\, c_k^{min}, c_k^{max} \,\right]$$

Contrast to neighbor pixels

The mean difference to the surrounding area. This feature is used to find borders and gradations.

> Object Features
> Layer Values
> Pixel Based
> **Contrast to neighbor pixels**

Parameters:

$B_v(d)$: extended bounding box of an image object **v** with distance **d**
$B_v(d)$:= $\{(x,y) : x_{min}(v)-d \le x \le x_{max}(v)+d\ ,\ y_{min}(v)-d \le y \le y_{max}(v)+d\}$

P_v : set of pixels of an image object **v**

$\bar{c}_k$: mean intensity of layer **k**

Formula:

$$1000 * \left(1 - \frac{\bar{c}_k\left(B_v(d) - P_v\right)}{1 + \bar{c}_k\left(P_v\right)}\right)$$

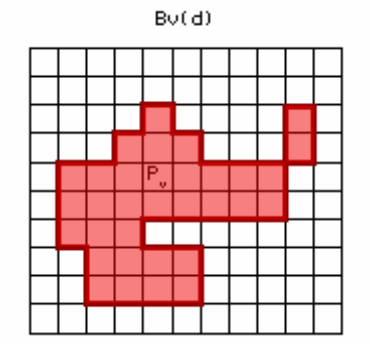


Figure 45: Contrast to neighbor pixels

Feature value range:

[-1000, 1000]

Conditions:

- If **d=0**, then $\mathbf{B_v(d)=B_v}$, and if $\mathbf{B_v=P_v}$ ∴ the formula is invalid.
- If unsigned data exist then maybe $\bar{c}_k(P_v) = -1$ ∴ the formula is invalid.
- If $\bar{c}_k(P_v)=0$ then the values are meaningless.
- The distance should always be greater than **0**, **(d>0)**.

Std. deviation to neighbor pixels

Computes the standard deviation of the pixels not belonging to the image object in the extended bounding box.

> Object Features
> Layer Values
> Pixel Based
> **StdDev. to neighbor pixels**

Parameters:

$\mathbf{P_v}$:set of pixels of an image object **v**

$\mathbf{B_v(d)}$**:** extended bounding box of an image object **v** with distance **d**

Formula:

$\sigma_k(B_v(d) - P_v)$

Feature value range:

$[0, c_k^{max}/2]$

Condition:

If **d=0**, then $\mathbf{B_v(d)=B_v}$, and if $\mathbf{B_v=P_v}$ ∴ the formula is invalid.

4.3.2.4 To Neighbors

> Object Features
> Layer Values
> **To Neighbors**

Mean diff. to neighbors

For each neighboring object the layer mean difference is computed and weighted with regard to the length of the border between the objects (if they are direct neighbors, feature distance = 0) or the area covered by the neighbor objects (if neighborhood is defined within a certain perimeter (in pixels) around the image object in question, feature distance > 0).

> Object Features
> Layer Values
> To Neighbors
> **Mean diff. to neighbors**

The mean difference to direct neighbors is calculated as follows:

Parameters:

u,v : image objects

b(v,u): topological relation border length

$\bar{c}_k$:mean intensity of layer **k**

c_k^{max} : brightest possible intensity value of **k**

c_k^{min} : darkest possible intensity value of **k**

#P_u: total number of pixels contained in P_u

d: distance between neighbors

w_u: weight of image object **u**

w: image layer weight

N_v: direct neighbors to an image object **v**
N_v:={u∈V_i : ∃(x,y)∈P_v∃(x',y')∈P_u :(x',y')∈N_4(x,y)}

N_v**(d)**: neighbors to **v** at a distance **d**
N_v**(d)**:={u∈V_i: d(v,u)≤d}

$$w_u := \begin{cases} b(v,u), d = 0 \\ \#P_u, d > 0 \end{cases}$$

$$w := \sum_{u \in N_v(d)} w_u$$

Formula:

$$\bar{\Delta}_k(v) := \frac{1}{w} \sum_{u \in N_v(d)} w_u \left(\bar{c}_k(v) - \bar{c}_k(u) \right)$$

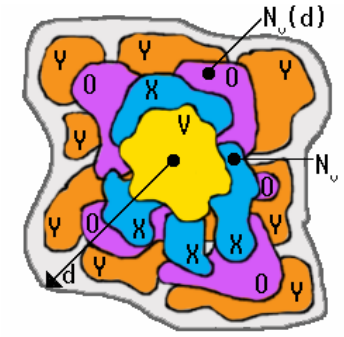


Figure 46: Direct and distance neighbors.

Feature value range:

$$\left[c_k^{min}, c_k^{max} \right]$$

Condition:

If **w=0** ⇒ the mean difference to neighbors is **0** therefore the formula is invalid.

Mean diff. to neighbors (abs)

The same definition as for **Mean diff. to neighbors**, with the difference that absolute values of the differences are averaged:

Parameters:

v,u : image objects

b(v,u): topological relation border length

$\bar{c}_k$:mean intensity of layer **k**

c_k^{max} : brightest possible intensity value of **k**

c_k^{min} : darkest possible intensity value of **k**

c_k^{range} : data range of **k**
$c_k^{range}=c_k^{max} - c_k^{min}$

d: distance between neighbors

#P$_u$ total number of pixels contained in **P**$_u$

w: image layer weight

w$_u$: weight of image object **u**

N$_v$: direct neighbors to an image object **v**
$N_v:=\{u \in V_i : \exists(x,y) \in P_v \exists(x',y') \in P_u : (x',y') \in N_4(x,y)\}$

N$_v$**(d)**: neighbors to **v** at a distance **d**
$N_v(d):=\{u \in V_i : d(v,u) \leq d\}$

$$w_u := \begin{cases} b(v,u), d = 0 \\ \#P_u, d > 0 \end{cases}$$

$$w := \sum_{u \in N_v(d)} w_u$$

Formula:

$$\bar{\Delta}_k(v) := \frac{1}{w} \sum_{u \in N_v(d)} w_u \left| \bar{c}_k(v) - \bar{c}_k(u) \right|$$

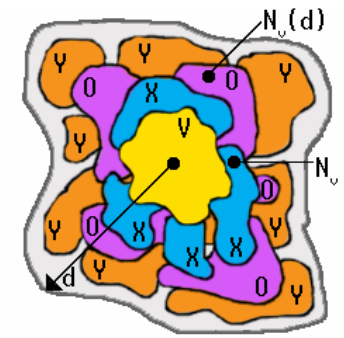


Figure 47: Direct and distance neighbors.

Feature value range:

$$\left[\ 0, c_k^{range}\ \right]$$

Condition:

If **w=0** ⇒ the mean difference to neighbors is **0** therefore the formula is invalid.

Mean diff. to darker neighbors

> Object Features
> Layer Values
> To Neighbors
> **Mean diff. to darker neighbors**

This feature is computed the same way as **Mean diff. to neighbors**, but only image objects with a layer mean value less than the layer mean value of the object concerned are regarded.

Parameters:

v,u : image objects

b(v,u): top relation border length

$\bar{c}_k$:mean intensity of layer **k**

$\mathbf{c}_k^{max}$: brightest possible intensity value of **k**

$\mathbf{c}_k^{min}$: darkest possible intensity value of **k**

$\mathbf{c}_k^{range}$: data range of **k**
$\mathbf{c}_k^{range} = c_k^{max} - c_k^{min}$

d: distance between neighbors

w: image layer weight

$\mathbf{w}_u$: weight of image object **u**

$\mathbf{N}_v$: direct neighbors to an image object **v**
$\mathbf{N}_v := \{u \in V_i : \exists (x,y) \in P_v \exists (x',y') \in P_u : (x',y') \in N_4(x,y)\}$

$\mathbf{N}_v(\mathbf{d})$: neighbors to **v** at a distance **d**
$\mathbf{N}_v(\mathbf{d}) := \{u \in V_i : d(v,u) \le d\}$

$\mathbf{N}_v^D(\mathbf{d})$: darker neighbors to **v** at a distance **d**
$\mathbf{N}_v^D(\mathbf{d}) := \{u \in N_v(d) : \bar{c}_k(u) < \bar{c}_k(v)\}$

$$w_u := \begin{cases} b(v,u), d = 0 \\ \#P_u, d > 0 \end{cases}$$

$$w := \sum_{u \in N_v(d)} w_u$$

Formula:

$$\bar{\Delta}_k^D(v) := \frac{1}{w} \sum_{u \in N_v^D(d)} w_u \left(\bar{c}_k(v) - \bar{c}_k(u) \right)$$

Feature value range:

$$\left[-C_k^{range}, C_k^{range} \right]$$

Conditions:

If **w=0** then $\bar{\Delta}_k^D$**(v) =0** ∴ the formula is invalid.

If $\mathbf{N}_v^D$**(d)=**∅ ∴ the formula is invalid.

Mean diff. to brighter neighbors

This feature is computed the same way as **Mean diff. to neighbors**, but only image objects with a layer mean value larger than the layer mean value of the object concerned are regarded.

> Object Features
> Layer Values
> To Neighbors
> **Mean diff. to brighter neighbors**

Parameters:

v,u : image objects

b(v,u): top relation border length

$\bar{c}_k$:mean intensity of layer **k**

$\mathbf{c}_k^{max}$: brightest possible intensity value of **k**

$\mathbf{c}_k^{min}$: darkest possible intensity value of **k**

$\mathbf{c}_k^{range}$: data range of **k**
$\mathbf{c}_k^{range}=c_k^{max} - c_k^{min}$

d: distance between neighbors

w: image layer weight

$\mathbf{w}_u$: weight of image object **u**

$\mathbf{N}_v$: direct neighbors to an image object **v**
$\mathbf{N}_v$:={u∈V_i : ∃(x,y)∈P_v∃(x',y')∈P_u :(x',y')∈N_4(x,y)}

$\mathbf{N}_v$**(d)**: neighbors to **v** at a distance **d**
$\mathbf{N}_v$**(d)**:={u∈V_i: d(v,u)≤d}

$\mathbf{N}_v^B$**(d)**: brighter neighbors to **v** at a distance **d**
$\mathbf{N}_v^B$**(d)**:={u∈N_v(d): $\bar{c}_k$(u)> $\bar{c}_k$(v)}

$$w_u := \begin{cases} b(v,u), d = 0 \\ \#P_u, d > 0 \end{cases}$$

$$w := \sum_{u \in N_v(d)} w_u$$

Formula:

$$\bar{\Delta}_k^B(v) := \frac{1}{w} \sum_{u \in N_v^B(d)} w_u \left(\bar{c}_k(v) - \bar{c}_k(u) \right)$$

Feature value range:

$$\left[-C_k^{range}, C_k^{range} \right]$$

Conditions:

If **w=0** then $\overline{\Delta}_k^B$**(v) =0** ∴ the formula is invalid.

If $\mathbf{N}_v^B$**(d)=∅** ∴ the formula is invalid.

Rel. border to brighter neighbors

Ratio of shared border with image objects of a higher mean value in the selected layer and the total border of the image object concerned.

> Object Features
> Layer Values
> To Neighbors
> **Rel. border to brighter neighbors**

Parameters:

$\mathbf{N}_v^B$**(d)**: brighter neighbors to **v** at a distance **d**
$\mathbf{N}_v^B$**(d)**:={u∈N_v(d): $\overline{c}_k(u) > c_k(v)$}

$\mathbf{b}_v$: image object border length

b(v,u): top relation border length

d: distance between neighbors

Formula:

$$\sum_{u \in N_v^B(d)} b(v,u) / b_v$$

Feature value range:

[0,1]

4.3.2.5 To Superobject

> Object Features
> Layer Values
> **To Superobject**

Mean Diff. to Superobject

Difference between layer L mean value of an image object and the layer L mean value of its superobject. You can determine in which image object level the superobject is selected by editing the feature distance.

> Object Features
> Layer Values
> To Superobject
> **Mean diff. to superobject**

Parameters:

$\overline{c}_k$:mean intensity of layer **k**

$\mathbf{c}_k^{range}$: data range of **k**
$\mathbf{c}_k^{range}$ **:=** $\mathbf{c}_k^{max}$ **:-** $\mathbf{c}_k^{min}$

$\mathbf{S}_v$**(d)** : subobject of **v** with hierarchical distance **d**

$U_v(d)$: superobject of **v** with hierarchical distance **d**

V_i: image objects level, **i=1,...,n**

Formula:

$$\overline{c}_k(v) - \overline{c}_k(U_v(d))$$

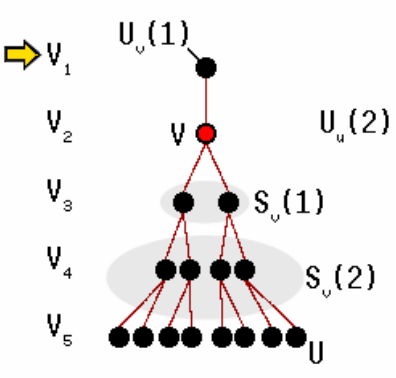


Figure 48: Image object Hierarchy

Feature value range:

$$\left[-C_k^{range}, C_k^{range} \right]$$

Ratio to Superobject

Ratio of the layer **k** mean value of an image object and the layer **k** mean value of its superobject. You can determine in which image object level the superobject is selected by editing the feature distance.

> Object Features
> Layer Values
> To Superobject
> **Ratio to superobject**

Parameters:

$U_v(d)$: superobject of **v** with hierarchical distance **d**

$\overline{c}_k$:mean intensity of layer **k**

Formula:

$$\frac{\overline{c}_k(v)}{\overline{c}_k(U_v(d))}$$

Feature value range:

[0, ∞]

Conditions:

If **$U_v(d)$**=∅ ∴the formula is undefined.

If $\mathbf{U_v(d)=0 \rightarrow \infty}$ ∴ the formula is undefined.

Stddev. Diff. to Superobject

Difference between layer **k** Stddev value of an image object and the layer **k** Stddev of its superobject. You can determine in which image object level the superobject is selected by editing the feature distance.

> Object Features
> Layer Values
> To Superobject
> **Stddev. diff. to superobject**

Parameters:

$\mathbf{U_v(d)}$: superobject of **v** with hierarchical distance **d**

$\mathbf{\sigma_k(v)}$: std. deviation of object **v** on layer **k**

$\mathbf{c_k^{range}}$: data range of layer **k**
$\mathbf{c_k^{range}} := \mathbf{c_k^{max}} - \mathbf{c_k^{min}}$

Formula:

$$\sigma_k(v) - \sigma_k(U_v(d))$$

Feature value range:

$$\left[-\frac{1}{2}c_k^{range}, \frac{1}{2}c_k^{range}\right]$$

Condition:

If $\mathbf{U_v(d)=\varnothing}$ ∴ the formula is undefined.

Stddev. Ratio to Superobject

Ratio of the layer **k** standard deviation of an image object and the layer **k** standard deviation of its superobject. You can determine in which image object level the superobject is selected by editing the feature distance.

> Object Features
> Layer Values
> To Superobject
> **Stddev. ratio. to superobject**

Parameters:

$\mathbf{U_v(d)}$: super object of **v** with hierarchical distance **d**

$\mathbf{\sigma_k(v)}$: std. deviation of object **v** on layer **k**

Formula:

$$\frac{\sigma_k(v)}{\sigma_k(U_v(d))}$$

Feature value range:

[0,∞]

Conditions:

If $\mathbf{U_v(d)}=\varnothing$ ∴ the formula is undefined.

If $\sigma_k(\mathbf{U_v(d)})=\mathbf{0}$ ⇒the std. deviation ratio to $\mathbf{U_v(d)}$ =**1**.

4.3.2.6 To Scene

> Object Features
> Layer Values
> **To Scene**

Mean diff. to scene

Difference between layer **K** mean value of an image object and the layer **K** mean value of the whole scene.

> Object Features
> Layer Values
> To Scene
> **Mean diff. to scene**

Parameters:

$\bar{\mathbf{c}}_k$: mean intensity of layer **k**

$\bar{\mathbf{c}}_k\mathbf{(v)}$: mean intensity of layer **k** of an image object **v**

$\mathbf{c}_k^{range}$: data range of layer **k**
$\mathbf{c}_k^{range} := \mathbf{c}_k^{max} : \mathbf{c}_k^{min}$

Formula:

$$\bar{c}_k(v) - \bar{c}_k$$

Feature value range:

$$\left[-c_k^{range}, c_k^{range} \right]$$

Ratio to scene

Ratio to scene of layer **k** is the layer **k** mean value of an image object divided by the layer **k** mean value of the whole scene.

> Object Features
> Layer Values
> To Scene
> **Ratio to scene**

Parameters:

$\bar{\mathbf{c}}_k$: mean intensity of layer **k**

$\bar{\mathbf{c}}_k\mathbf{(v)}$: mean intensity of layer **k** of an image object **v**

Formula:

$$\frac{\bar{c}_k(v)}{\bar{c}_k}$$

Feature value range:

[-∞, ∞]

Condition:

If $\bar{\mathbf{c}}_k$=**0** ⇔the feature is undefined as the image object is black.

4.3.2.7 Hue, Saturation, Intensity

> Object Features
> Layer Values
> **Hue, Saturation, Intensity**

Performs a transformation of values of the RGB color space to values of the HSI color space. You can create three different types of **HSI Transformation** features as **Output** here:

- Hue
- Saturation
- Intensity

When creating a new HSI transformation, you have to assign a corresponding image layers to red (R), green (G) and blue (B). By default these are the first three image layers of the scene.

Hue

> Object Features
> Layer Values
> Hue, Saturation, Intensity
> **Hue**

The hue value of the HSI color space representing the gradation of color.

Parameters:

R, G, B: values expressed as numbers from 0 to 1

MAX: the greatest of the (R, G, B) values

MIN: the smallest of the (R, G, B) values

Formula:

$$H = \begin{cases} undefined & if\ MAX = MIN \\ 60° x \dfrac{G-B}{MAX-MIN} \Big/ 360°, & if\ MAX = R \\ 60° x \dfrac{B-R}{MAX-MIN} + 120° \Big/ 360°, & if\ MAX = G \\ 60° x \dfrac{R-G}{MAX-MIN} + 240° \Big/ 360°, & if\ MAX = B \end{cases}$$

Feature value range:

[0,1]

Condition:

When creating a new HSI transformation, you have to assign the corresponding image layers to red (R), green (G) and blue (B).

Saturation

The saturation value of the HSI color space representing the intensity of a specific hue.

> Object Features
> Layer Values
> Hue, Saturation, Intensity
> **Saturation**

Parameters:

R, G, B: values expressed as numbers from 0 to 1

MAX: the greatest of the (R, G, B) values

MIN: the least of the (R, G, B) values

Formula:

$$S = \begin{cases} 0 & if\ MAX = 0 \\ \dfrac{MAX - MIN}{MAX}, & if\ MAX \neq 0 \end{cases}$$

Feature value range:

[0,1]

Conditions:

When creating a new HSI transformation, you have to assign the according image layers to red (R), green (G) and blue (B).

Intensity

The intensity value of the HSI color space representing the lightness spanning the entire range from black through the chosen hue to white.

> Object Features
> Layer Values
> Hue, Saturation, Intensity
> **Intensity**

Parameters:

R, G, B: values expressed as numbers from 0 to 1

MAX: the greatest of the (R, G, B) values

MIN: the least of the (R, G, B) values

Formula:

I=MAX

Feature value range:

[0,1]

Condition:

When creating a new HSI transformation, you have to assign the according image layers to red (R), green (G) and blue (B).

4.3.3 Shape

> Object Features
> **Shape**

4.3.3.1 Generic

> Object Features
> Shape
> **Generic**

Area

In non-georeferenced data the area of a single pixel is 1. Consequently, the area of an image object is the number of pixels forming it. If the image data is georeferenced, the area of an image object is the true area covered by one pixel times the number of pixels forming the image object.

> Object Features
> Shape
> Generic
> **Area**

Parameters:

#P_v: total number of pixels contained in **P_v**

Feature value range:

[0; scene size]

Asymmetry

The more longish an image object, the more asymmetric it is. For an image object, an ellipse is approximated which can be expressed by the ratio of the lengths of the minor and the major axis of this ellipse. The feature value increases with the asymmetry.

> Object Features
> Shape
> Generic
> **Asymmetry**

Note

We recommend to use the **Length/Width** ratio because it is more accurate.

→ **Length/Width** on page 119

→ **Shape-Related Features** on page 91

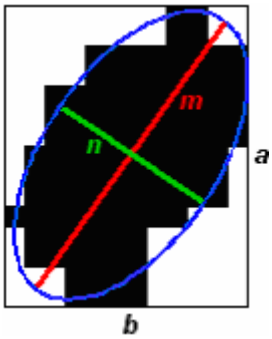
Parameters:

VarX : variance of X

VarY: variance of Y

Expression:

$$\frac{2\sqrt{\frac{1}{4}(VarX+VarY)^2+(VarXY)^2-VarX\cdot VarY}}{VarX+VarY}$$



Feature value range:

[0, 1]

Border index

> Object Features
> Shape
> Generic
> **Border index**

Similar to shape index, but border index uses a rectangular approximation instead of a square. The smallest rectangle enclosing the image object is created. The border index is then calculated as the ratio of the **Border length** of the image object to the **Border length** of this smallest enclosing rectangle.

Parameters:

$\mathbf{b}_v$: image object border length

$\mathbf{l}_v$: length of an image object **v**

$\mathbf{w}_v$: width of an image object **v**

Expression:

$$\frac{b_v}{2(l_v + w_v)}$$

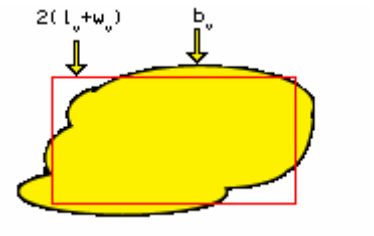


Figure 49: Border index of an image object v

Feature value range:

[1,∞], **1**=ideal.

The more fractal an image object appears, the higher its border index.

Border length

The border length e of an image object is defined as the sum of edges of the image object that are shared with other image objects or are situated on the edge of the entire scene. In non-georeferenced data the length of a pixel edge is 1.

> Object Features
> Shape
> Generic
> **Border length**

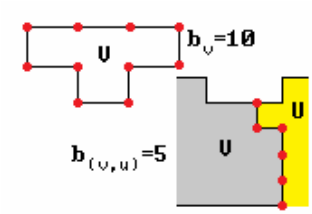


Figure 50: Border length of an image object v or between two objects v, u.

Feature value range:

[0, ∞]

Compactness

This feature is similar to the border index, however instead of border based it is area based.

> Object Features
> Shape
> Generic
> **Compactness**

The compactness of an image object **v**, used as a feature, is calculated by the product of the length **l** and the width **w** and divided by the number of its pixels **#P**$_v$.

Parameters:

l$_v$: length of an image object **v**

w$_v$: width of an image object **v**

#P$_v$**:** total number of pixels contained in **P**$_v$

Expression:

$$\frac{l_v * w_v}{\#P_v}$$

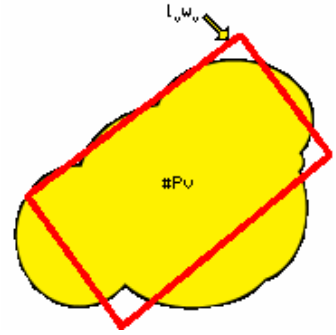


Figure 51: Compactness of an image object v

Feature value range:

[0, ∞], **1**=ideal.

The more compact an image object appears, the smaller its border.

Density

The density an be expressed by the area covered by the image object divided by its radius. **Definiens Developer** uses the following implementation, where n is the number of pixels forming the image object and the radius is approximated using the covariance matrix. Use the density to describe the compactness of an image object. The ideal compact form on a pixel raster is the square. The more the form of an image object is like a square, the higher its density.

> Object Features
> Shape
> Generic
> **Density**

Parameters:

$\sqrt{\#\mathbf{P}_v}$: diameter of a square object with $\#\mathbf{P}_v$ pixels.

$\sqrt{\mathbf{VarX+VarY}}$: diameter of the ellipse

Expression:

$$\frac{\sqrt{\#P_v}}{1+\sqrt{VarX+VarY}}$$

Feature value range:

[0, depended on shape of image object]

Elliptic fit

As a first step in the calculation of the elliptic fit is the creation of an ellipse with the same area as the considered object. In the calculation of the ellipse the proportion of the length to the width of the Object is regarded. After this step the area of the object outside the ellipse is compared with the area inside the ellipse that is not filled out with the object. While 0 means no fit, 1 stands for a complete fitting object.

> Object Features
> Shape
> Generic
> **Elliptic fit**

Parameters:

$\varepsilon_v(x,y)$: elliptic distance at a pixel **(x,y)**

$\mathbf{P}_v$**:** set of pixels of an image object **v**

$\#\mathbf{P}_v$**:** total number of pixels contained in $\mathbf{P}_v$

Formula:

$$\phi := 2 \cdot \frac{\#\{(x,y) \in P_v : \varepsilon_v(x,y) \le 1\}}{\#P_v} - 1$$

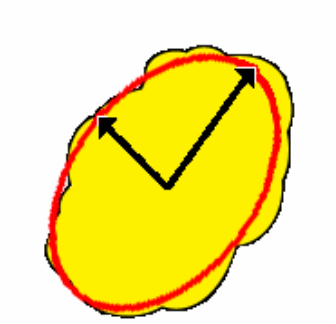


Figure 52: Elliptic fit of an image object v

Feature value range:

[0,1], **1**=complete fitting, whereas **0** = only 50% or less pixels fit inside the ellipse.

Length

The length can be calculated using the length-to-width ratio derived from a bounding box approximation.

> Object Features
> Shape
> Generic
> **Length**

Parameters:

#P_v: total number of pixels contained in $\mathbf{P}_v$

γ_v : length/width ratio of an image object **v**

Expression:

$$\sqrt{\#P_v \cdot \gamma_v}$$

Feature value range:

[0; ∞]

Length/Width

There are two ways to approximate the length/width ratio of an image object:

> Object Features
> Shape
> Generic
> **Length/Width**

- The ratio length/width is identical to the ratio of the eigenvalues of the covariance matrix with the larger eigenvalue being the numerator of the fraction:

$$\gamma_v^{EV} = \frac{\lambda_1(v)}{\lambda_2(v)}$$

- The ratio length/width can also be approximated using the bounding box:

$$\gamma_v^{BB} = \frac{\left(k_v^{bb'}\right)^2}{\#P_v}$$

Definiens Developer uses both methods for the calculation and takes the smaller of both results as the feature value.

Parameters:

#Pv :Size of a set of pixels of an image object **v**

λ_1, λ_2:eigenvalues

γ_v^{EV} :ratio length of **v** of the eigenvalues

γ_v^{BB} :ratio length of **v** of the bounding box

γ_v : length/width ratio of an image object **v**

$\mathbf{k}_v^{bb'}$:

$\mathbf{h}_v^{bb'}$:

a: Bounding box fill rate

$\mathbf{\#P}_{xl}$

h :

w : image layer weight

$$k_v^{bb'} = \sqrt{(k_v^{bb'})^2 + (1-a)(h_v^{bb'})^2}$$

$$a = {\#Pv}\Big/{k_v^{bb'} \cdot h_v^{bb'}}$$

$$k \cdot h = \#P_v \Rightarrow k = \frac{\#P_v}{w}, h = \frac{\#P_v}{k} \Rightarrow \frac{k}{w} = \frac{k^2}{\#P_{xl}} = \frac{\#P_{xl}}{w^2}$$

Formula:

$$\gamma_v := \min \gamma_v^{EV}, \min \gamma_v^{BB}$$

Feature value range:

[0; ∞]

Main direction

In **Definiens Developer**, the main direction of an image object is the direction of the eigenvector belonging to the larger of the two eigenvalues derived from the covariance matrix of the spatial distribution of the image object.

> Object Features
> Shape
> Generic
> **Main direction**

Parameters:

VarX : variance of X

VarY: variance of Y

λ_1 : eigenvalue

Expression:

$$\frac{180^\circ}{\pi}\tan^{-1}(VarXY, \lambda_1 - VarY) + 90^\circ$$

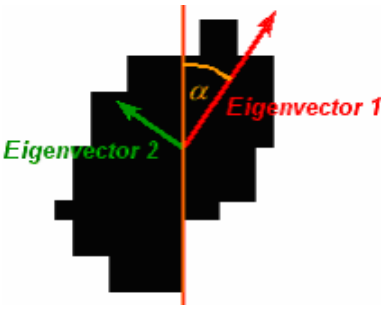


Figure 53: Ellipse approximation using eigenvalues.

Feature value range:

[0; 180]

Radius of largest enclosed ellipse

An ellipse with the same area as the object and based on the covariance matrix. This ellipse is then scaled down until it's totally enclosed by the object. The ratio of the radius of this largest enclosed ellipse to the radius of the original ellipse is returned for this feature.

> Object Features
> Shape
> Generic
> **Radius of largest enclosing ellipse**

Parameters:

$\varepsilon_v(x,y)$: elliptic distance at a pixel **(x,y)**

Expression:

$\varepsilon_v(x_o,y_o)$

with $(x_o,y_o) = \max \varepsilon_v(x,y), (x,y) \in P_v$

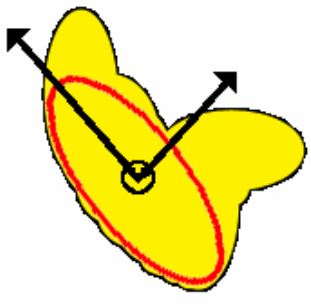


Figure 54: Radius of largest enclosed ellipse of an image object v

Feature value range:

[0,∞]

Radius of smallest enclosing ellipse

An ellipse with the same area as the object and based on the covariance matrix. This ellipse is then enlarged until it's enclosing the object in total. The ratio of the radius of this smallest enclosing ellipse to the radius of the original ellipse is returned for this feature.

> Object Features
> Shape
> Generic
> **Radius of smallest enclosing ellipse**

Parameters:

$\varepsilon_v(x,y)$: elliptic distance at a pixel **(x,y)**

Expression:

$\varepsilon_v(x_o,y_o)$

with $(x_o,y_o) = \min \varepsilon_v(x,y), (x,y) \notin P_v$

Figure 55: Radius of smallest enclosing ellipse of an image object v

Feature value range:

[0,∞]

Rectangular fit

A first step in the calculation of the rectangular fit is the creation of a rectangle with the same area as the considered object. In the calculation of the rectangle the proportion of the length to the width of the object in regarded. After this step the area of the object outside the rectangle is compared with the area inside the rectangle, which is not filled out with the object.

> Object Features
> Shape
> Generic
> **Rectangular fit**

Parameters:

$\rho_v(x,y)$: rectangular distance at a pixel **(x,y)**

Expression:

$$\frac{\#\{(x,y) \in P_v : \rho_v(x,y) \leq 1\}}{\#P_v}$$

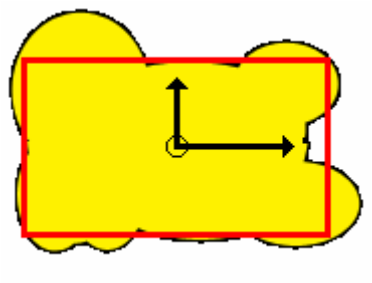


Figure 56: Rectangular fit of an image object v

Feature value range:

[0,1], **1**=complete fitting, whereas **0** = 0% fit inside the rectangular approximation.

Roundness

Object Features
> Shape
> Generic
> **Roundness**

> Difference of enclosing/enclosed ellipse as the radius of the largest enclosed ellipse is subtracted from the radius of the smallest enclosing ellipse.

Parameters:

ε_v^{max}: radius of smallest enclosing ellipse

ε_v^{min}: radius of largest enclosed ellipse

Expression:

$\varepsilon_v^{max} - \varepsilon_v^{min}$

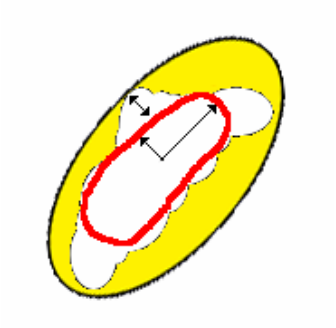


Figure 57: Roundness of an image object v

Feature value range:

[0,∞], **0**=ideal.

Shape index

Mathematically the shape index is the border length e of the image object divided by four times the square root of its area A. Use the shape index s to describe the smoothness of the image object borders.

> Object Features
> Shape
> Generic
> **Shape index**

Parameters:

$\mathbf{b}_v$: image object border length

$4\sqrt{\#P_v}$: border of square with area $\mathbf{\#P}_v$

Expression:

$$\frac{b_v}{4\sqrt{\#P_v}}$$

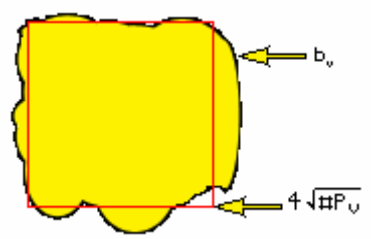


Figure 58: Shape index of an image object v

Feature value range:

[1,∞], **1**=ideal.

The more fractal an image object appears, the higher its shape index.

Width

The width of an image object is calculated using the length-to-width ratio.

> Object Features
> Shape
> Generic
> **width**

Parameters:

$\#P_v$: total number of pixels contained in $\mathbf{P}_v$

γ_v : length/width ratio of an image object **v**

Expression:

$$\frac{\#P_v}{\gamma_v}$$

Feature value range:

[0; ∞]

4.3.3.2 Line Features Based on Subobject Analysis

> Object Features
> Shape
> **Line features based on sub-object analysis**

The information for classification of an object can also be derived from information provided by its subobjects. A specific method is to produce compact sub-objects for the purpose of line analysis. The basic idea is to represent the shape of an object by compact subobjects and operate from center point to center point to get line information. Nevertheless it is possible to determine to which image object level the feature should refer to.

As mentioned above, this method is superior to bounding box approximation, if you want to extract features out of lengthy and curved image objects (e.g., image objects representing rivers or roads).

Note

These features are provided for backward compatibility only. It is not recommended to use them in new rule sets.

Width (line so)

> Object Features
> Shape
> Line features based on sub-object analysis
> **Width (line so)**

The image object width calculated on the basis of sub-objects is the area A (in pixels) of the image object divided by its length derived from sub-object analysis.

Parameters:

•

Formula:

$$w_{so} = \frac{A}{l_{so}}$$

Feature value range:

[1; depending on image object shape]

Length/Width (line so)

> Object Features
> Shape
> Line features based on sub-object analysis
> **Length/width (line so)**

The length-to-width ratio based on subobject analysis is the square length derived from subobject analysis divided by the object area (in pixels).

Parameters:

•

Formula:

$$\gamma_{SO} = \frac{l_{SO}}{w_{SO}} = \frac{l_{SO}^2}{A}$$

Feature value range:

[0; 1]

Length (line so)

Of the image object of concern, the object center is known. Among all the sub-objects those two objects are detected which are situated furthest from this center point. From one end point to the other, the distances between the center points of adjacent sub-objects are added together (red lines). The radii of the end objects are also considered to complete the approximation (green)

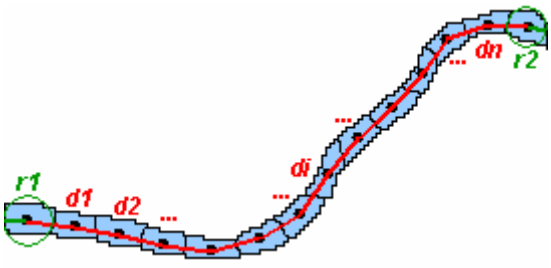
> Object Features
> Shape
> Line features based on sub-object analysis
> **Length (line so)**

Parameters:

•

Formula:

$$l_{SO} = r_1 + r_2 + \sum_{i=1}^{n} d_i$$



Feature value range:

[1; depending on image object shape]

Curvature/length (line so)

The curvature of an image object divided by its length. Both curvature and length are based on analysis of subobjects. The curvature is the sum of all changes in direction (absolute values) when iterating through the subobjects from both ends to the subobject that is situated closest to the center of the image object of concern.

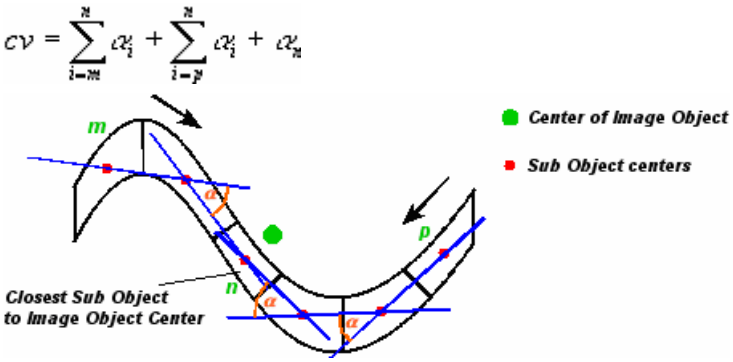
> Object Features
> Shape
> Line features based on sub-object analysis
> **Curvature/length (line so)**

Parameters:

•

Formula:

The curvature is calculated as follows:



Feature value range:

[0; depending on image object shape]

Stddev. curvature (line so)

> Object Features
> Shape
> Line features based on sub-object analysis
> **Stddev. curvature (line so)**

The standard deviation of all changes in direction (IMAGE) when iterating through the subobjects from both ends to the subobject situated closest to the center of the image object of concern. If an image object can be characterized by a high standard deviation of its curvature, this means that there are a large number of changes in direction when iterating through the subobjects. On the other hand, an image object may appear curved, but if it follows a circular line, the standard deviation of its curvature will be small, since the changes in direction when iterating through its subobjects are more or less constant.

The polygon shape features are based on the vectorization of the pixels that form an image object. The following figure shows a raster image object with its polygon object after vectorization.

The lines that are shown in red colors are the edges of the polygon object of the raster image object.

4.3.3.3 Position

Position features refer to the position of an image object relative to the entire scene. These features are of special interest when working with geographical referenced data as an image object can be described by its geographic position.

> Object Features
> Shape
> **Position**

Distance to line

Distance to a line, which could be manually defined by the enter of two points that are a part of this line. Note, that the line has neither a start nor an end. Click with the right mouse button on the feature, select **Edit Feature** and adapt the coordinates to your analysis.

> Object Features
> Shape
> Position
> **Distance to line**

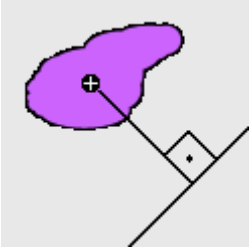


Figure 59: Distance between an image object and a line

Feature value range:

[0; square root of (rows^2+columns^2) or depending on the coordinates]

Distance to image border

Distance to the nearest border of the image.

> Object Features
> Shape
> Position
> **Distance to image border**

Parameters:

minx :minimum distance from the image border at **x**-axis

maxx : maximum distance from the image border at **x**-axis

miny : minimum distance from the image border at **y**-axis

maxy : maximum distance from the image border at **y**-axis

(sx,sy) : scene size

Formula:

min {minx, sx-maxx, miny, sy-maxy}

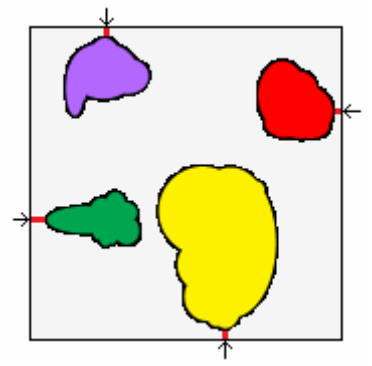


Figure 60: Distance between the nearest border and the image object

Feature value range:

[0,max{sx-1, sy-1}]

X center

X-position of the image object center (center of gravity, mean value of all X-coordinates).

> Object Features
> Shape
> Position
> **X center**

Parameters:

$\bar{\mathbf{x}}_v$: **x** center of an image object **v**

#P$_v$: total number of pixels contained in **P**$_v$

Formula:

$$\bar{x}_v := \frac{1}{\#P_v} \cdot \sum_{(x,y) \in P_v} x$$

Figure 61: Center of gravity of an image object v

Feature value range:

[0, x_coordinate. of the center of gravity]

X distance to image left border

> Object Features
> Shape
> Position
> **X distance to image left border**

Horizontal distance to the left border of the image.

Parameters:

sx-1 : scene size at the left border

minx :minimum distance from the image border at **x**-axis

Formula:

$$\min_{(x,y)\in Pv} x$$

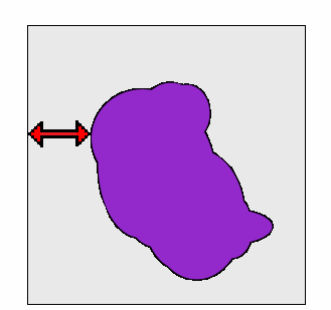


Figure 62: X _distance between the image object and the left border

Feature value range:

[0,sx-1]

X distance to image right border

> Object Features
> Shape
> Position
> **X distance to image right border**

Horizontal distance to the right border of the image.

Parameters:

sx : scene size at the right border.

maxx : maximum distance from the image border at **x**-axis

Formula:

$$sx - \max_{(x,y)\in Pv} x$$

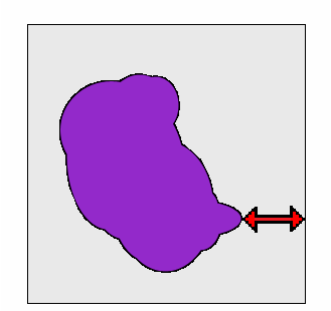


Figure 63: X distance to the image object right border

Feature value range:

[0,sx-1]

X max.

Maximum X-position of the image object (derived from bounding box).

> Object Features
> Shape
> Position
> **X max**

Formula:

$$\max_{(x,y)} x$$

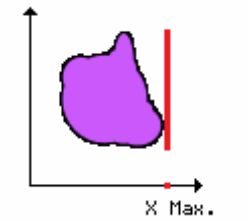


Figure 64: Maximum value of X_Cord at the image object border

Feature value range:

[0, y_coordinate. of the center of gravity]

X min.

Minimum X-position of the image object (derived from bounding box).

> Object Features
> Shape
> Position
> **X min**

Formula:

$$\min_{(x,y)} x$$

Figure 65: Minimum value of X_Cord at the image object border

Feature value range:

[0, y_coordinate. of the center of gravity]

Y max.

Maximum Y-position of the image object (derived from bounding box).

> Object Features
> Shape
> Position
> **Y max**

Formula:

$$\max_{(x,y)} y$$

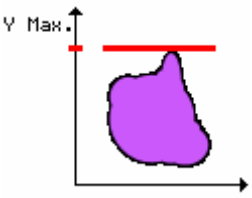


Figure 66: Maximum value of Y_Cord at the image object border.

Feature value range:

[0, y_coordinate. of the center of gravity]

Y center

Y-position of the image object center (center of gravity, mean value of all Y-coordinates).

> Object Features
> Shape
> Position
> **Y center**

Parameters:

$\bar{\mathbf{y}}_v$: **y** center of an image object **v**

#P$_v$: total number of pixels contained in **P**$_v$

Formula:

$$\bar{y}_v := \frac{1}{\#P_v} \cdot \sum_{(x,y)\in P_v} y$$

Figure 67: Center of gravity of an image object v

Feature value range:

[0, y_coordinate. of the center of gravity]

Y min.

Minimum Y-position of the image object (derived from bounding box).

> Object Features
> Shape
> Position
> **Y min**

Formula:

$$\min_{(x,y)} y$$

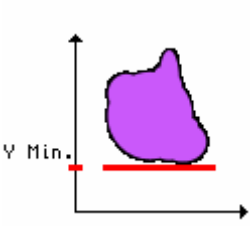


Figure 68: Minimum value of Y_Cord at the image object border

Feature value range:

[0, y_coordinate. of the center of gravity]

Y distance to image bottom border

Vertical distance to the bottom border of the image.

> Object Features
> Shape
> Position
> **Y distance to image bottom border**

Parameters:

sy : scene size at the bottom border

miny : minimum distance from the image border at **y**-axis

Formula:

$$\min_{(x,y)\in Pv} y$$

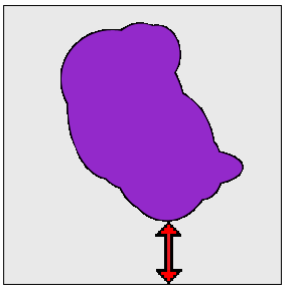


Figure 69: Y _distance between the image object and the bottom border

Feature value range:

[0,sy-1]

Y distance to image top border

Vertical distance to the top border of the image.

> Object Features
> Shape
> Position
> **Y distance to image top border**

Parameters:

sy : scene size at the top border.

maxy : maximum distance from the image border at **y**-axis

Formula:

$$sy - \max_{(x,y)\in Pv} y$$

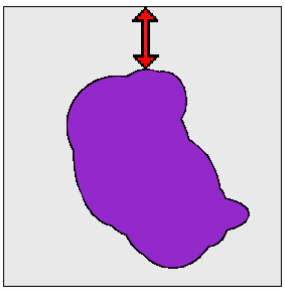


Figure 70: Y _distance between the image object and the top border

Feature value range:

[0,sy-1]

4.3.3.4 To Superobject

Use **To Superobject** features to describe an image object by its form relations to one of its superobjects (if there are any). Which superobject is to be referred to is defined by editing the feature distance (n). Especially when working with thematic layers these features might be of great interest.

> Object Features
> Shape
> **To Superobject**

Rel. area to superobject

The feature is computed by dividing the area of the image object of concern by the area covered by its superobject. If the feature value is 1, then the image object is identical to its superobject. Use this feature to describe an image object by the amount of area it covers of its superobject.

> Object Features
> Shape
> To Superobject
> **Rel. area to superobject**

Parameters:

#P$_v$: total number of pixels contained in **P**v

#**P**$_{Uv(d)}$: the size of the superobject of **v**

Formula:

$$\frac{\#\ P_v}{\#\ P_{U_v(d)}}$$

Condition:

If $\mathbf{U_v(d)} = \varnothing$ ∴ the formula is undefined.

Feature value range:

[0,1]

Rel. rad. position to superobject (n)

The feature value is calculated by dividing the distance from the center of the image object of concern to the center of its superobject by the distance of the center of the most distant image object which has the same superobject.

Use this feature to describe an image object by its position relative to the center of its superobject.

> Object Features
> Shape
> To Superobject
> **Rel. rad. position to superobject**

Parameters:

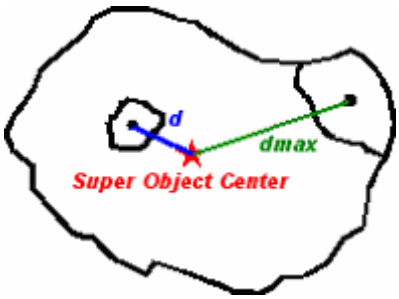
#P$_v$: total number of pixels contained in **P**v

#**P**$_{Uv(d)}$: the size of the superobject of an image object **v**

$\mathbf{d_g(v,U_v(d))}$: distance of **v** to the center of gravity of the superobject $\mathbf{U_v(d)}$

Formula:

$$\frac{d_g(v, U_v(d))}{\max_{u \in S_{U_v(d)}(d)} d_g(u, U_v(d))}$$



Feature value range:

[0; 1]

Condition:

If $\mathbf{U_v(d)} = \varnothing$ ∴ the formula is undefined.

Rel. inner border to superobject (n)

This feature is computed by dividing the sum of the border shared with other image objects that have the same superobject by the total border of the image object. If the relative inner border to the superobject is 1, the image object of concern is not situated on the border of its superobject. Use this feature to describe how much of an image object is situated at the edge of its superobject.

> Object Features
> Shape
> To Superobject
> **Rel. inner border to superobject**

Parameters:

$\mathbf{N_U(v)}$:Neighbors of **v** that exist within the superobject
$\mathbf{N_U(v)} := \{u \in N_v : U_u(d) - U_v(d)\}$

$\mathbf{b_v}$: image object border length

Formula:

$$\frac{\sum_{u \in N_U(v)} b(v, m)}{b_v}$$

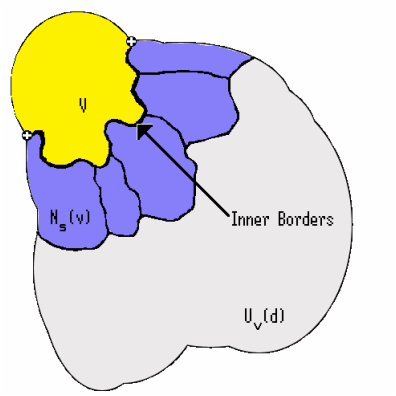
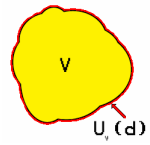


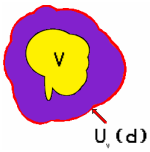
Figure 71: Relative inner border of an image object v to super object U

Conditions:

If the feature range is **0 ⇒ v=U_v(d).**



If the feature range is **1 ⇒ v** is an inner object.



Feature value range:

[0; 1]

Distance to superobject center

The distance of this image objects center to the center of the superobject of this image object. This might not be the shortest distance between the two points, since the way to the center of the superobject has to be within the borders of the superobject.

d_g(v,U_v(d)) : distance of **v** to the center of gravity of the superobject **U_v(d)**

> Object Features
> Shape
> To Superobject
> **Distance to superobject center**

Feature value range:

[0; sx*sy]

Elliptic distance to superobject center

Distance of objects to the center of the superobject.

> Object Features
> Shape
> To Superobject
> **Elliptic distance to superobject center**

Expression:

d_e(v,U_v(d))

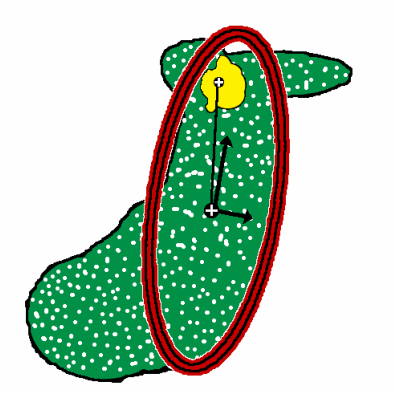


Figure 72: Distance between the distance from the superobject's center to the center of a subobject.

Feature value range:

typically [0; 5]

Is end of superobject

This feature is true only for two image objects **a** and **b**, both being sub-objects of the same superobject if:
a is the image object with the maximum distance to the superobject,
b is the image object with the maximum distance to a.

> Object Features
> Shape
> To Superobject
> **Is end of superobject**

Is center of superobject

This feature is true, if the image object is the center of its superobject.

> Object Features
> Shape
> To Superobject
> **Is center of superobject**

Rel. x position to super-object

This feature returns the relative x postion of an image object with regard to its superobject, based on the centers of gravity of both objects.

> Object Features
> Shape
> To Superobject
> **Rel. x position to super-object**
> → **User Guide Basic Concepts**

Parameter

Distance in image object hierarchy: Select the distance (upward) in the image object hierarchy between subobject and superobject.

Feature Range

[– scene width/2 , + scene width/2]

Formula

$\Delta x = x_{CG}$ of current image object – x_{CG} of superobject, where x_{CG} is the center of gravity.

Rel. y position to super-object

This feature returns the relative y postion of an image object with regard to its super-object, based on the centers of gravity of both objects.

> Object Features
> Shape
> To Superobject
> **Rel. y position to super-object**
➔ **User Guide**
Basic Concepts

Parameter

Distance in image object hierarchy: Select the distance (upward) in the image object hierarchy between sub-object and super -object.

Feature Range

[– scene height/2 , + scene height/2]

Formula

$\Delta y = y_{CG}$ of current image object – y_{CG} of superobject, where y_{CG} is the center of gravity.

4.3.3.5 Based on Polygons

The polygon features provided by **Definiens Developer** are based on the vectorization of the pixels that form an image object. The following figure shows a raster image object with its polygon object after vectorization:

> Object Features
> Shape
> **Based on Polygons**

The lines that are shown in red colors are the edges of the polygon object of the raster image object.

Edges longer than

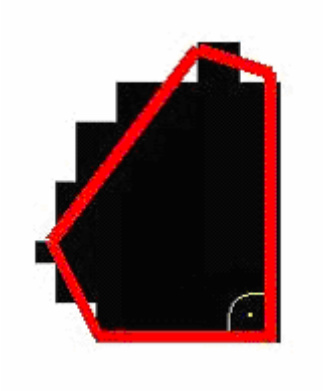
This feature reports the number of edges that have lengths exceeding a threshold value. The user defines the threshold value.

> Object Features
> Shape
> Based on Polygons
> **Edges longer than**

Number of right angles with edges longer than

This feature value gives the number of right angles that have at least one side edge longer than a given user defined threshold. The following figure shows a polygon with one rectangular angle:

> Object Features
> Shape
> Based on Polygons
> **Number of right angles with edges longer**



Area (excluding inner polygons)

> Object Features
> Shape
> Based on Polygons
> **Area (excluding inner polygons)**

Calculating the area of a polygon is based on Green's Theorem in a plane. Given points (x_i, y_i), i = 0,…, n, with $x_0 = x_n$ and $y_0 = y_n$, the following formula can be used for rapidly calculating the area of a polygon in a plane:

Parameters:

a_i :

Formula:

$$Area = \frac{1}{2}\sum_{i=0}^{n-1} a_i$$

where

$$a_i = x_i y_{i+1} - x_{i+1} y_i$$

This value does not include the areas of existing inner polygons.

Area (including inner polygons)

> Object Features
> Shape
> Based on Polygons
> **Area (including inner polygons)**

The same formula as for area (excluding inner polygon) is used to calculate this feature. The areas of the existing inner polygons in the selected polygon are taken into account for this feature value.

Figure 73: The area of an image object v including an inner polygon.

The above picture shows a polygon with one inner object

Average length of edges (polygon)

This feature calculates the average length of all edges in a polygon.

> Object Features
> Shape
> Based on Polygons
> **Average length of edges (polygon)**

Parameters:

$\mathbf{X}_i$: length of edge **i**

n : total number of edges

Formula:

$$Average = \frac{\sum_{i=1}^{n} X_i}{n}$$

Compactness (polygon)

Compactness is defined as the ratio of the area of a polygon to the area of a circle with the same perimeter. The following formula is used to calculate the compactness of the selected polygon:

> Object Features
> Shape
> Based on Polygons
> **Compactness (polygon)**

$$compactness = \frac{4 * \pi * Area}{Perimeter^2}$$

Feature value range:

[0; 1 for a circle]

Length of longest edge (polygon)

The value of this feature contains the length of the longest edge in the selected polygon.

> Object Features
> Shape
> Based on Polygons
> **Length of longest edge (polygon)**

Number of edges (polygon)

This feature value simply represents the number of edges which form the polygon.

> Object Features
> Shape
> Based on Polygons
> **Number of edges (polygon)**

Number of inner objects (polygon)

If the selected polygon includes some other polygons (image objects), the number of these objects is assigned to this feature value. The inner objects are completely surrounded by the outer polygon.

> Object Features
> Shape
> Based on Polygons
> **Number of inner objects (polygon)**

Perimeter (polygon)

The sum of the lengths of all edges which form the polygon is considered as the perimeter of the selected polygon.

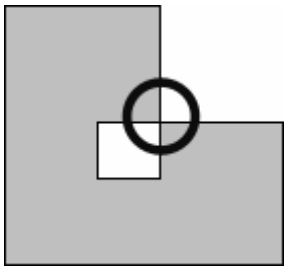
> Object Features
> Shape
> Based on Polygons
> **Perimeter (polygon)**

Polygon self-intersection (polygon)

The feature polygon intersection allows identifying a rarely occurring special constellation of image objects which leads to a polygon self-intersection when exported as a polygon vector file.

This feature enables you to identify the affected objects and take measures to avoid the self-intersection. All objects with a value of 1 will cause a polygon self-intersection when exported to a shape file.

The type of object pictured below will lead to a self-intersection at the circled point. To avoid the self-intersection, the enclosed object needs to be merged with the enclosing object.



Tip

*Use the **image object fusion** algorithm to remove polygon intersections. To do so, set the domain to all objects which have a value larger than 0 for the polygon intersection feature. In the algorithm parameter, set the fitting function threshold to polygon intersection = 0, in the weighted sum setting set Target value factor to 1. This will merge all objects with a value of 1 for the feature polygon intersection so that the resulting object will not sport a self intersection.*

→ **Image Object Fusion** on page 43

Stddev of length of edges (polygon)

This feature value shows how the lengths of edges deviate from their mean value. The following formula for standard deviation is used to compute this value.

> Object Features
> Shape
> Based on Polygons
> **Stddev of length of edges**

Parameters:

$\mathbf{X}_i$: length of edge i

$\bar{X}$: mean value of all lengths

n : total number of edges

Formula:

$$stddev = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n}}$$

4.3.3.6 Based on Skeletons

For the better understanding of the following descriptions the skeleton is divided in a main line and branches as above mentioned. Each mid-point of the triangles created by the Delaunay Triangulation is called a node.

> Object Features
> Shape
> **Based on Skeletons**

Number of segments of order

Number of segments of order calculates the number of line segments of branches with a selected order. Note, that only segments are counted that do not belong to a lower order. Define the branch order in the dialog Edit Parametrized Features. To open this dialog select Edit Features from the pop up menu that is opened with a right click on the corresponding feature. For more information see **Parametrized Features** section.

> Object Features
> Shape
> Based on Skeletons
> **Number of segments of order**

Feature value range:

[0; depending on shape of objects]

Number of branches of order

Number of branches of order calculates the number of branches of a predefined order. Define the branch order in the dialog Edit Parametrized Features. To open this dialog select Edit Feature from the pop up menu that is opened with a right click on the corresponding feature. For more information see **Parametrized Features** section.

> Object Features
> Shape
> Based on Skeletons
> **Number of branches of order**

Feature value range:

[0; depending on shape of objects]

Average length of branches of order

Average length of branches of order calculates the average length of branches of a selected order. The length of the branch of the selected order is measured from the intersect point of the whole branch and the main line to the end of the branch. The order can be manually defined. With a right click on the feature a pop up menu opens where you have to select Edit Feature. In the dialog it is possible to select the order of the branches to select. For more information see **Parametrized Features** section.

> Object Features
> Shape
> Based on Skeletons
> **Average length of branches of order**

Feature value range:

[0; depending on shape of objects]

Number of branches of length

Number of branches of length calculates the number of branches of a special length up to a selected order. At this all ends of branches are counted up to the selected order. Since it is a parametrized feature it is possible to select the branch order and the length in a special range manually. To do so, select Edit Feature from the pop up menu you can open with a right click. For more information see **Parametrized Features** section.

> Object Features
> Shape
> Based on Skeletons
> **Number of branches of length**

Feature value range:

[0; depending on shape of objects]

Average branch length

Average branch length calculates the average length of all branches of the corresponding object.

> Object Features
> Shape
> Based on Skeletons
> **Average branch length**

Feature value range:

[0; depending on shape of objects]

Avrg. area represented by segments

Calculates the average area of all triangles created by the Delaunay Triangulation (see fig. 11).

> Object Features
> Shape
> Based on Skeletons
> **Avg. area represented by segments**

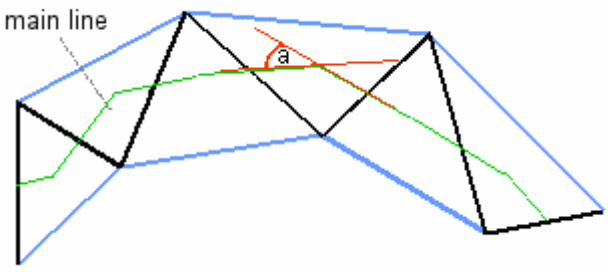
Feature value range:

[0; depending on shape of objects]

Curvature/length (only main line)

The feature Curvature/length (only main line) is calculated by the ratio of the curvature of the object and its length. The curvature is the sum of all changes in direction of the main line. Changes in direction are expressed by the acute angle a in which sections of the main line, built by the connection between the nodes, cross each other.

> Object Features
> Shape
> Based on Skeletons
> **Curvature/length (only main line)**



Feature value range:

[0; depending on shape of objects]

Degree of skeleton branching

The degree of skeleton branching describes the highest order of branching in the corresponding object.

> Object Features
> Shape
> Based on Skeletons
> **Degree of skeleton branching**

Feature value range:

[0; depending on shape of objects]

Length of main line (no cycles)

The length of the main line is calculated by the sum of all distances between its nodes. No cycles means that if an object contains an island polygon, the main line is calculated without regarding the island polygon. In this case the main line could cross the island polygon. Note that this is an internal calculation and could not be visualized like the skeletons regarding the island polygons.

> Object Features
> Shape
> Based on Skeletons
> **Length of main line (no cycles)**

Feature value range:

[0; depending on shape of objects]

Length of main line (regarding cycles)

The length of the main line is calculated by the sum of all distances between its nodes. regarding cycles means that if an object contains an island polygon, the main line is calculated regarding this island polygon. Consequently the main line describes a path around the island polygon. This way also the skeletons for visualization are calculated.

> Object Features
> Shape
> Based on Skeletons
> **Length of main line (regarding cycles)**

Feature value range:

[0; depending on shape of objects]

Length/Width (only main line)

In the feature Length/width (only main line) the length of an object is divided by its width.

> Object Features
> Shape
> Based on Skeletons
> **Length/width (only main line)**

Feature value range:

[0; depending on shape of objects]

Maximum branch length

Maximum branch length calculates the length of the longest branch. The length of a branch is measured from the intersect point of the branch and the main line to the end of the branch.

> Object Features
> Shape
> Based on Skeletons
> **Maximum branch length**

Feature value range:

[0; depending on shape of objects]

Number of segments

Number of segments is the number of all segments of the main line and the branches.

> Object Features
> Shape
> Based on Skeletons
> **Number of segments**

Feature value range:

[0; depending on shape of objects]

Stddev. curvature (only main line)

The standard deviation of the curvature is the result of the standard deviation of the changes in direction of the main line. Changes in direction are expressed by the acute angle in which sections of the mainline, built by the connection between the nodes, cross each other.

> Object Features
> Shape
> Based on Skeletons
> **Stddev. curvature (only main line)**

Feature value range:

[0; depending on shape of the objects]

Stddev. of area represented by segments

Calculates the standard deviation of all triangles created by the Delaunay Triangulation.

> Object Features
> Shape
> Based on Skeletons
> **Stddev. of area represented by segments**

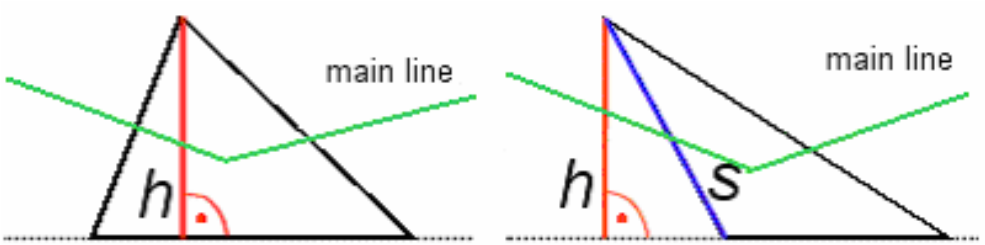
Feature value range:

[0; depending on shape of the objects]

Width (only main line)

To calculate the width of the objects the average height **h** of all triangles crossed by the main line is calculated. An exception are triangles in which the height **h** does not cross one of the sides of the corresponding triangle. In this case the nearest side **s** is used to define the height.

> Object Features
> Shape
> Based on Skeletons
> **Width (only main line)**



Feature value range:

[0; depending on shape of objects]

4.3.4 Texture

All features concerning texture are based on subobject analysis. This means you must have a image object level of subobjects to be able to use them.

> Object Features
> **Texture**

The image object level of subobjects to use can be defined by editing the feature distance.

→ **Level Distance** on page 88

The texture features are divided in the following groups:

- Texture concerning the spectral information of the subobjects

→ **Layer Value Texture Based on Subobjects** on page 147

- Texture concerning the form of the subobjects

→ **Shape Texture Based on Subobjects** on page 148

- Texture after Haralick based on the gray level co-occurrence matrix (GLCM), which is a tabulation of how often different combinations of pixel gray levels occur in an image.

→ **Texture After Haralick** on page 152

4.3.4.1 Layer Value Texture Based on Subobjects

These features refer to the spectral information provided by the image layers.

> Object Features
> Texture
> **Layer value texture based on sub-objects**

Mean of sub-objects: stddev.

Standard deviation of the different layer mean values of the sub-objects.

> Object Features
> Texture
> Layer value texture based on sub-objects
> **Mean of sub-objects: stddev.**

At first this feature might appear very similar to the simple standard deviation computed from the single pixel values (layer values), but it might be more meaningful since (a reasonable segmentation assumed) the standard deviation here is computed over homogeneous and meaningful areas. The smaller the sub-objects, the more the feature value approaches the standard deviation calculated from single pixels.

Parameters:

$\mathbf{S_v(d)}$: subobject of an image object **v** at distance **d**

$\bar{\mathbf{c}}_k\mathbf{(u)}$: mean intensity of layer **k** of an image object **u.**

d : level distance

Formula:

$$\sqrt{\frac{1}{\#S_v(d)}\left(\sum_{u\in S_v(d)}(\bar{c}_k(u))^2-\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)}\bar{c}_k(u)\sum_{u\in S_v(d)}\bar{c}_k(u)\right)}$$

Feature value range:

[0; depending on bit depth of data]

Conditions:

If **Sv(d) =** $\varnothing$ $\therefore$ the formula is invalid.

Avrg. mean diff. to neighbors of subobjects

The contrast inside an image object expressed by the average mean difference of all its subobjects for a specific layer.

> Object Features
> Texture
> Layer value texture based on sub-objects
> **Avrg. mean diff. to neighbors of subobjects**

This feature has a certain spatial reference, as a local contrast inside the area covered by the image object is described. For each single subobject the layer L mean difference (absolute values) to adjacent subobjects of the same superobject is calculated. The feature value is the mean value of the layer L mean differences.

Parameters:

$\mathbf{S_v(d)}$: subobject of an image object **v** at distance **d**

$\bar{\Delta}_k\mathbf{(u)}$: mean difference to neighbor of layer **k** of an image object **u.**

d : level distance

Formula:

$$\frac{1}{\#S_v(d)} \sum_{u \in S_v(d)} \overline{\Delta}_k(u)$$

Feature value range:

[0; depending on bit depth of data]

Conditions:

If $\mathbf{S_v(d)} = \varnothing$ ∴ the formula is invalid

4.3.4.2 Shape Texture Based on Subobjects

The following features refer to the form of the sub-objects. The premise to use these features properly is an accurate segmentation of the image, because the sub-objects should be as meaningful as possible.

> Object Features
> Texture
> **Shape texture based on sub-objects**

Area of subobjects: mean

Mean value of the areas of the sub-objects.

> Object Features
> Texture
> Shape texture based on sub-objects
> **Area of sub-objects: mean**

Parameters:

$\mathbf{S_v(d)}$: subobject of an image object **v** at distance **d**

$\mathbf{\#P_u}$: total number of pixels contained in **u**

d : level distance

Formula:

$$\frac{1}{\#S_v(d)} \sum_{u \in S_v(d)} \#Pu$$

Feature value range:

[0; scene size]

Condition:

If $\mathbf{Sv(d)} = \varnothing$ ∴ the formula is invalid.

Area of subobjects: stddev.

Standard deviation of the areas of the sub-objects.

> Object Features
> Texture
> Shape texture based on sub-objects
> **Area of subobjects: stddev.**

Parameters:

$\mathbf{S_v(d)}$: subobject of an image object **v** at distance **d**

#P$_u$: total number of pixels contained in **u**

d : level distance

Formula:

$$\sqrt{\frac{1}{\#S_v(d)}\left(\sum_{u\in S_v(d)}\#P_u^2-\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)}\#P_u\sum_{u\in S_v(d)}\#P_u\right)}$$

Feature value range:

[0; scene size]

Condition:

If **Sv(d)** = ∅ ∴ the formula is invalid.

Density of subobjects: mean

Mean value calculated from the densities of the subobjects.

> Object Features
> Texture
> Shape texture based on sub-objects
> **Density of subobjects: mean**

Parameters:

S$_v$**(d)** : subobject of an image object **v** at distance **d**

a(u) : density of **u**

d : level distance

Formula:

$$\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)}a(u)$$

For more details on density see the **Density** topic under shape features.

→ **Density** on page 118

Feature value range:

[0; depending on image object shape]

Condition:

If **Sv(d)** = ∅ ∴ the formula is invalid.

Density of subobjects: stddev.

Standard deviation calculated from the densities of the subobjects.

> Object Features
> Texture
> Shape texture based on sub-objects
> **Density of subobjects: stddev.**

Parameters:

S$_v$**(d)** : subobject of an image object **v** at distance **d**

a(u) : density of **u**

d : level distance

Formula:

$$\sqrt{\frac{1}{\#S_v(d)}\left(\sum_{u\in S_v(d)} a^2(u) - \frac{1}{\#S_v(d)}\sum_{u\in S_v(d)} a(u)\sum_{u\in S_v(d)} a(u)\right)}$$

Feature value range:

[0; depending on image object shape]

Condition:

If **Sv(d)** = ∅ ∴ the formula is invalid.

Asymmetry of subobjects: mean

Mean value of the asymmetries of the subobjects.

> Object Features
> > Texture
> > Shape texture based on sub-objects
> > **Asymmetry of subobjects: mean**

Parameters:

S_v(d) : subobject of an image object **v** at distance **d**

a(u) : asymmetry of **u**

d : level distance

Formula:

$$\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)} a(u)$$

For more details on asymmetry see the **Asymmetry** section under shape features.

→ Asymmetry on page 115

Feature value range:

[0; depending on image object shape]

Condition:

If **Sv(d)** = ∅ ∴ the formula is invalid.

Asymmetry of subobjects: stddev.

Standard deviation of the asymmetries of the sub-objects.

> Object Features
> > Texture
> > Shape texture based on sub-objects
> > **Asymmetry of sub-objects: stddev.**

Parameters:

S_v(d) : subobject of an image object **v** at distance **d**

a(u) : asymmetry of **u**

d : level distance

Formula:

$$\frac{1}{\#S_v(d)} \sum_{u \in S_v(d)} a(u)$$

For more details on asymmetry see the **Asymmetry** section under shape features.

→ Asymmetry on page 115

Feature value range:

[0; depending on image object shape]

Condition:

If **Sv(d)** = ∅ ∴ the formula is invalid.

Direction of subobjects: mean

Mean value of the directions of the sub-objects. In the computation, the directions are weighted with the asymmetry of the respective sub-objects (the more asymmetric an image object, the more significant its main direction).

Before computing the actual feature value, the algorithm compares the variance of all sub-object main directions with the variance of the sub-object main directions, where all directions between 90°; and 180°; are inverted (direction - 180°). The set of sub-object main directions which has the lower variance is selected for the calculation of the main direction mean value weighted by the sub-object asymmetries.

> Object Features
> Texture
> Shape texture based on sub-objects
> **Direction of sub-objects: mean**

Parameters:

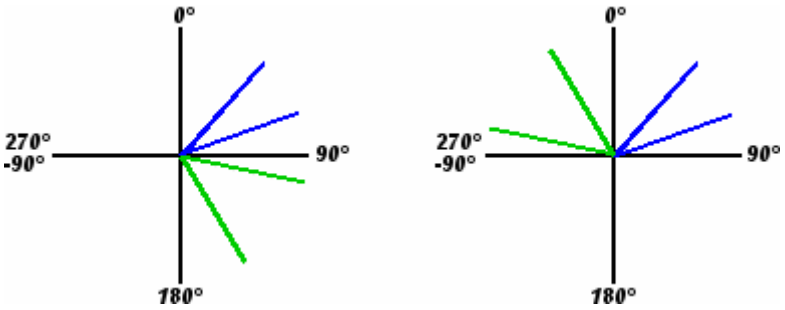
S_v(d) : subobject of an image object **v** at distance **d**

a(u) : main direction of **u**

d : level distance

Formula:

$$\frac{1}{\#S_v(d)} \sum_{u \in S_v(d)} a(u)$$



For more details on main direction see the **Main Direction** section under shape features.

→ Main direction on page 120

Feature value range:

[0-180°]

Condition:

If **Sv(d) = ∅ ∴** the formula is invalid.

Direction of subobjects: stddev.

Standard deviation of the directions of the sub-objects. Again, the sub-object main directions are weighted by the asymmetries of the respective sub-objects. The set of sub-object main directions of which the standard deviation is calculated is determined in the same way as explained above (Direction of SO: Mean).

> Object Features
> Texture
> Shape texture based on sub-objects
> **Direction of sub-objects: stddev.**

4.3.4.3 Texture After Haralick

The gray level co-occurrence matrix (GLCM) is a tabulation of how often different combinations of pixel gray levels occur in an image. A different co-occurrence matrix exists for each spatial relationship. To receive directional invariance all 4 directions (0°, 45°, 90°, 135°) are summed before texture calculation. An angle of 0° represents the vertical direction, an angle of 90° the horizontal direction. In **Definiens** software, texture after Haralick is calculated for all pixels of an image object. To reduce border effects, pixels bordering the image object directly (surrounding pixels with a distance of one) are additionally taken into account. The directions to calculate texture after Haralick in **Definiens** software are:

> Object Features
> Texture
> **Texture after Haralick**

Parameters:

i : the row number

j : the column number

$\mathbf{V}_{i,j}$: the value in the cell i,j of the matrix

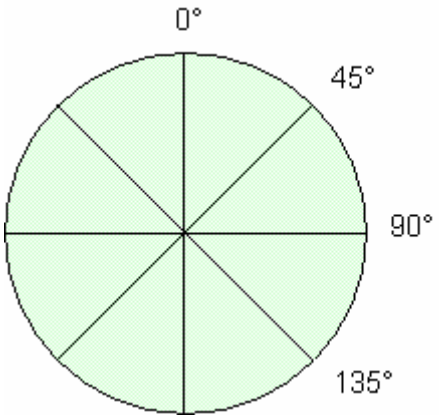
$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

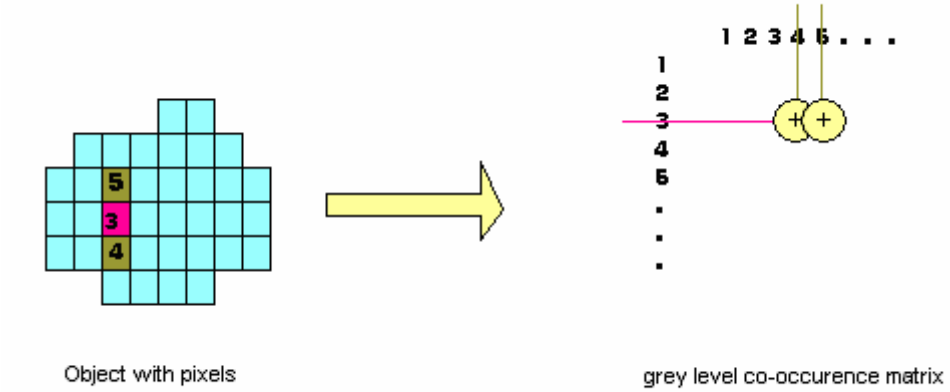
Formula:

Every GLCM is normalized according to the following operation:

$$P_{i,j} = \frac{V_{i,j}}{\sum_{i,j=0}^{N-1} V_{i,j}}$$



The normalized GLCM is symmetrical. The diagonal elements represent pixel pairs with no gray level difference. Cells, which are one cell away from the diagonal, represent pixel pairs with a difference of only one gray level. Similarly, values in cells, which are two pixels away from the diagonal, show how many pixels have a 2 gray levels and so forth. The more distant to the diagonal, the greater the difference between the pixels' gray levels is. Summing-up the values of these parallel diagonals, gives the probability for each pixel to be 0, 1, 2 or 3 etc. different to its neighbor pixels.



Another approach to measure texture is to use a gray-level difference vector (GLDV) instead of the GLCM. The GLDV is the sum of the diagonals of the GLCM. It counts the occurrence of references to the neighbor pixels' absolute differences. In **Definiens** software the GLCM and GLDV are calculated based on the pixels of an object. They are computed for each input layer. Within each Texture after Haralick feature you have the choice of either one of the above directions or of all directions.

The calculation of Texture after Haralick is independent of the image data's bit-depth. The dynamic range is interpolated to 8 bit before evaluating the co-occurrence. However, if 8 bit data is used directly the results will be most reliable. When using data of higher dynamic than 8 bit, the mean and standard deviation of the values is calculated. Assuming a Gaussian distribution of the values, more than 95% is in-between the interval:

$$\bar{x} - 3 * \sigma < x < \bar{x} + 3 * \sigma$$

The interval is subdivided into 255 equal sub-intervals to obtain an 8 bit representation.

The calculation of the features

In the following for each **Texture after Haralick** feature its general calculation is described. The usable features are sorted by their direction of concern: All directions, Direction 0°, Direction 45°, Direction 90° and Direction 135°. Further, each feature is calculated based upon the gray values of one selectable layer.

Note

The calculation of any Texture after Haralick feature is very CPU demanding because auf the calculation of the GLCM.

Tip

GLCM (quick 8/11) features

*For each Haralick texture feature there is a performance optimized version labeled **quick 8/11**. The performance optimization works only on data with a bit depth of 8bit or 11bit. Hence the label **quick 8/11**. Use the performance optimized version whenever you work with 8 or 11 bit data. For 16 bit data, use the conventional Haralick feature.*

References

Haralick features were implemented in **Definiens** software according to the following references:

- R. M. Haralick, K. Shanmugan and I. Dinstein, Textural Features for Image Classification, IEEE Tr. on Systems, Man and Cybernetics, Vol SMC-3, No. 6, November 1973, pp. 610-621.
- R. M. Haralick, Statistical and Structural Approaches to Texture, Proceedings of the IEEE, Vol. 67, No. 5, May 1979, pp. 786-804.
- R. W. Conner and C. A. Harlow, A Theoretical Comparison of Texture Algorithms, IEEE Tr. on Pattern Analysis and Machine Intelligence, Vol PAMI-2, No. 3, May 1980

GLCM homogeneity

> Object Features
> Texture
> Texture after Haralick
> **GLCM homogeneity**

If the image is locally homogeneous, the value is high if GLCM concentrates along the diagonal. Homogeneity weights the values by the inverse of the Contrast weight with weights, decreasing exponentially according to their distance to the diagonal.

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

Formula:

$$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2}$$

Feature value range:

[0; 90]

GLCM contrast

Contrast is the opposite of homogeneity. It is a measure of the amount of local variation in the image. It increases exponentially as (i-j) increases.

> Object Features
> Texture
> Texture after Haralick
> **GLCM Contrast**

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

Formula:

$$\sum_{i,j=0}^{N-1} P_{i,j}(i-j)^2$$

Feature value range:

[0; 90]

GLCM dissimilarity

Similar to contrast, but increases linearly. High if the local region has a high contrast.

> Object Features
> Texture
> Texture after Haralick
> **GLCM dissimilarity**

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

Formula:

$$\sum_{i,j=0}^{N-1} P_{i,j}\left|i-j\right|$$

Feature value range:

[0; 90]

GLCM entropy

The value for entropy is high, if the elements of GLCM are distributed equally. It is low if the elements are close to either 0 or 1. Since ln(0) is undefined, it is assumed that 0 * ln(0) = 0.

> Object Features
> Texture
> Texture after Haralick
> **GLCM entropy**

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

Formula:

$$\sum_{i,j=0}^{N-1} P_{i,j}(-\ln P_{i,j})$$

Feature value range:

[0; 90]

GLCM ang. 2nd moment

> Object Features
> Texture
> Texture after Haralick
> **GLCM ang. 2nd moment**

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

Formula:

$$\sum_{i,j=0}^{N-1} P_{i,j}^{\ 2}$$

Feature value range:

[0; 90]

GLCM mean

The GLCM mean is the average expressed in terms of the GLCM. The pixel value is not weighted by its frequency of occurrence itself, but by the frequency of its occurrence in combination with a certain neighbor pixel value.

> Object Features
> Texture
> Texture after Haralick
> **GLCM Mean**

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

Formula:

$$\mu_{i,j} = \frac{\sum_{i,j=0}^{N-1} P_{i,j}}{N^2}$$

Feature value range:

[0; 90]

GLCM stddev

> Object Features
> Texture
> Texture after Haralick
> **GLCM StdDev.**

GLCM standard deviation uses the GLCM, therefore it deals specifically with the combinations of reference and neighbor pixels. Thus, it is not the same as the simple standard deviation of gray levels in the original image.

Calculating the standard deviation using i or j gives the same result, since the GLCM is symmetrical.

Standard deviation is a measure of the dispersion of values around the mean. It is similar to contrast or dissimilarity.

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

$\mu_{i,j}$: GLCM mean

Formula:

$$\sigma_{i,j}^2 = \sum_{i,j=0}^{N-1} P_{i,j}(i, j - \mu_{i,j})$$

Standard Deviation:

$$\sigma = \sqrt{\sigma_{i,j}^2}$$

Feature value range:

[0; 90]

GLCM correlation

> Object Features
> Texture
> Texture after Haralick
> **GLCM Correlation**

Measures the linear dependency of gray levels of neighboring pixels.

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

$\mu_{i,j}$: GLCM mean

$\sigma_{i,j}$: GLCM std. deviation

Formula:

$$\sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$$

Feature value range:

[0; 90]

GLDV angular 2nd moment

> Object Features
> Texture
> Texture after Haralick
> **GLDV Ang. 2nd moment**

High if some elements are large and the remaining ones are small. Similar to GLCM Angular Second Moment: it measures the local homogeneity.

Parameters:

N : the number of rows or columns

$\mathbf{V}_k$: image object level, k=1,...n

Formula:

$$\sum_{k=0}^{N-1} V_k^2$$

Feature value range:

[0; 90]

GLDV entropy

The values are high if all elements have similar values. It is the opposite of GLDV Angular Second Moment.

Parameters:

i : the row number

j : the column number

$\mathbf{P}_{i,j}$: the normalized value in the cell **i,j**

N : the number of rows or columns

$\mathbf{V}_k$: image object level, k=1,...n

Formula:

Since ln(0) is undefined, it is assumed that 0 * ln(0) = 0:

$$\sum_{k=0}^{N-1} V_k(-\ln V_k)$$

Feature value range:

[0; 90]

GLDV mean

The mean is mathematically equivalent to the GLCM Dissimilarity measure above. It is only left here for compatibility reasons.

Parameters:

N : the number of rows or columns

$\mathbf{V}_k$: image object level, k=1,...n

Formula:

$$\sum_{k=0}^{N-1} k(V_k)$$

Feature value range:

[0; 90]

GLDV contrast

It is mathematically equivalent to the GLCM Contrast measure above. It is only left here for compatibility reasons.

> Object Features
> Texture
> Texture after Haralick
> **GLDV Contrast**

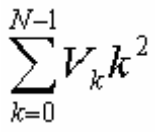
Parameters:

N : the number of rows or columns

$\mathbf{V}_k$: image object level, k=1,...n

k :

Formula:



Feature value range:

[0; 90]

GLCM and GLDV (quick 8/11)

For each Haralick texture feature there is a performance optimized version labeled quick 8/11.

The performance optimization works only on data with a bit depth of 8bit or 11bit. Hence the label quick 8/11. Use the performance optimized version whenever you work with 8 or 11 bit data. For 16 bit data, use the conventional Haralick feature.

4.3.5 Variables

All object variables are listed here.

> Object Features
> **Variables**

[name of a local variable]

Define variables to describe interim values. Variables are used as:

> Object Features
> Variables
> **[name of a local variable]**

- constants
- fixed and dynamic thresholds
- store temporary and final results

Variables should be used to store "tools" with which you may fine-tune your rule-sets for similar projects.

For detailed description on how to create a variable refer to the **Create a Variable** section.

Feature value range:

[-∞; ∞]

4.3.6 Hierarchy

Hierarchy features refer to the embedding of an image object in the entire image object hierarchy.

> Object Features
> **Hierarchy**

Level

The number of the image object level an image object is situated in. You will need this feature if you perform classification on different image object levels to define which class description is valid for which image object level.

> Object Features
> Hierarchy
> **Level**

Parameters:

U_v(d) : superobjects of an image object **v** at distance **d**

Formula:

$$\min_d U_v(d) = 0$$

Feature value range:

[1; number of image object levels]

Conditions:

To use this feature you need to have more than one image object levels.

Number of higher levels

The number of image object levels situated above the image object level the object of concern is situated in. This is identical to the number of superobjects an image object may have.

> Object Features
> Hierarchy
> **Number of higher levels**

Parameters:

d : distance between neighbors

S_v(d) : subobjects of an image object **v** at a distance **d**

Formula:

$$(\min_d S_v(d) = \varnothing) - 1$$

Feature value range:

[1; number of image object levels -1]

Number of neighbors

The number of the direct neighbors of an image object (i.e., neighbors with which it has a common border) on the same image object level in the image object hierarchy.

> Object Features
> Hierarchy
> **Number of neighbors**

Parameters:

$N_v(d)$: neighbors of an image object **v** at a distance **d**

Formula:

$\#N_v(d)$

Feature value range:

[0; number of pixels of entire scene]

Number of subobjects

Concerning an image object, the number of subobjects that are located on the next lower image object level in the image object hierarchy.

> Object Features
> Hierarchy
> **Number of sub-objects**

Parameters:

$S_v(d)$: subobjects of an image object **v** at a distance **d**

Formula:

$\#S_v(d)$

Feature value range:

[0; number of pixels of entire scene]

Number of sublevels

The number of image object levels situated below the image object level the object of concern is situated in.

> Object Features
> Hierarchy
> **Number of sublevels**

Parameters:

d : distance between neighbors

$U_v(d)$: superobjects of an image object **v** at a distance **d**

Formula:

$$(\min_d U_v(d) = \varnothing) - 1$$

Feature value range:

[1; number of image object levels -1]

4.3.7 Thematic Attributes

> Object Features
> **Thematic attributes**

Thematic attributes are used to describe an image object using information provided by thematic layers.

If your project contains a thematic layer, the object's thematic properties (taken from the thematic layer) can be evaluated. Depending on the attributes of the thematic layer, a large range of different features becomes available.

Note

If the currently open project does include a thematic layer, **Thematic attributes** features are not listed in the feature tree.

[name of the thematic objects attribute]

> Object Features
> Thematic Attributes
> **[name of the thematic objects attribute]**

If existing, **Thematic Objects Attribute** features referring to a thematic layer are listed in the feature tree. Available only for image objects that overlap with one or no thematic object.

Thematic object ID

> Object Features
> Thematic Attributes
> **Thematic object ID**

The identification number (ID) of a thematic object. Available only for image objects that overlap with one or no thematic object.

Number of overlapping thematic objects

> Object Features
> Thematic Attributes
> **Number of overlapping thematic objects**

The number of overlapping thematic objects. Available only for image object overlaps with one or no thematic object. Available only for image objects that overlap with several thematic objects.

Feature value range:

[0; number of thematic objects]

4.4 Class-Related Features

> Class-Related Features

4.4.1 Customized

> Class-Related Features
> **Customized**

[name of a customized feature]

> Class-Related Features
> Customized
> **[name of a customized feature]**

If existing, customized features referring to other classes are displayed here.

4.4.2 Relations to Neighbor Objects

Use the following features to describe an image object by the classification of other image objects on the same image object level in the image object hierarchy.

> Class-Related Features
> Relations to neighbor objects

Existence of

Existence of an image object assigned to a defined class in a certain perimeter (in pixels) around the image object concerned. If an image object of the defined classification is found within the perimeter, the feature value is 1 (= true), otherwise it would be 0 (= false). The radius defining the perimeter can be determined by editing the feature distance.

> Class-Related Features
> Relations to neighbor objects
> **Existence of**

Formula:

0 if $N_v(d,m) = \varnothing$
1 if $N_v(d,m) \neq \varnothing$

Feature value range:

[0,1]

Number of

Number of objects belonging to the selected class in a certain distance (in pixels) around the image object.

> Class-Related Features
> Relations to neighbor objects
> **Number of**

Parameters:

v : image object

d : distance between neighbors

m : a class containing image objects

Expression:

$\#N_v(d,m)$

Feature value range:

[0, ∞]

Border to

The absolute border of an image object shared with neighboring objects of a defined classification. If you use georeferenced data, the feature value is the real border to image objects of a defined class; otherwise it is the number of pixel edges shared with the adjacent image objects, as by default the pixel edge-length is 1.

> Class-Related Features
> Relations to neighbor objects
> **Border to**

Parameters:

b(v,u) : topological relation border length

N_v(d) : neighbors to an image object **v** at a distance **d**

Expression:

$$\sum_{u \in N(d,m)} b(v,u)$$

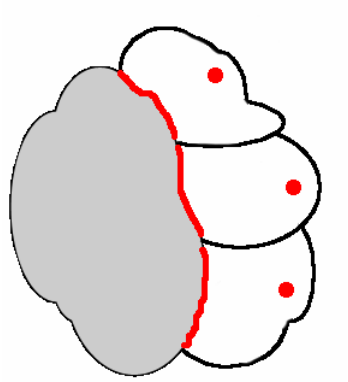


Figure 74: The absolute border between unclassified and classified image objects.

Feature value range:

[0, ∞]

Rel. border to

The feature **Relative border to (Rel. border to)** refers to the length of the shared border of neighboring image objects. The feature describes the ratio of the shared border length of an image object with a neighboring image object assigned to a defined class to the total border length. If the relative border of an image object to image objects of a certain class is **1**, the image object is totally embedded in these image objects. If the relative border is 0.5 then the image object is surrounded by half of its border.

> Class-Related Features
> Relations to neighbor objects
> **Rel. border to**

Parameters:

b(v,u) : topological relation border length

$\mathbf{N_v(d)}$: neighbors to an image object **v** at a distance **d**

$\mathbf{b_v}$: image object border length

Expression:

$$\frac{\sum_{u \in N_v(d,m)} b(v,u)}{b_v}$$

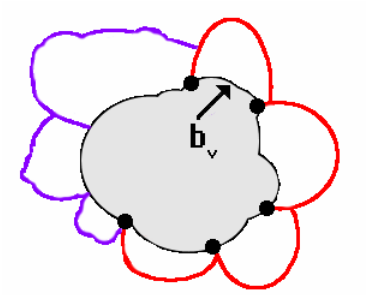


Figure 75: Relative border between neighbors.

Feature value range:

[0,1]

Conditions:

If the relative border is **0** then the class **m** does not exist.

If the relative border is **1** then the object **v** is completely surrounded by class **m**

Rel. area of

Area covered by image objects assigned to a defined class in a certain perimeter (in pixels) around the image object concerned divided by the total area of image objects inside this perimeter. The radius defining the perimeter can be determined by editing the feature distance.

> Class-Related Features
> Relations to neighbor objects
> **Rel. area of**

Parameters:

$N_v(d)$: neighbors to an image object **v** at a distance **d**

#P_u : total number of pixels contained in P_u

Expression:

$$\frac{\sum_{u \in N_v(d,m)} \#P_u}{\sum_{u \in N_v(d)} \#P_u}$$

Feature value range:

[0; 1]

Conditions:

If the relative border is **0** then the class **m** does not exist.

If the relative border is **1** then the object **v** is completely surrounded by class **m**.

Distance to

> Class-Related Features
> Relations to neighbor objects
> **Distance to**

The distance (in pixels) of the image object's center concerned to the closest image object's center assigned to a defined class. The image objects on the line between the image object's centers have to be of the defined class.

Parameters:

d(v,u) : distance between **v** and **u**

$\mathbf{V_i(m)}$: image object level of a class **m**

$\mathbf{b_v}$: image object border length

Expression:

$$\min_{u \in V_i(m)} d(v,u)$$

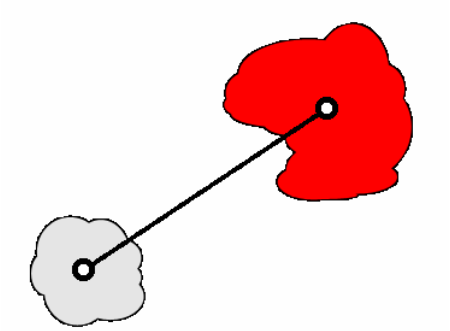


Figure 76: Distance between the centers of neighbors.

Feature value range:

[0,∞]

Mean diff. to

> Class-Related Features
> Relations to neighbor objects
> **Mean diff. to**

The mean difference of the layer L mean value of the image object concerned to the layer L mean value of all image objects assigned to a defined class.

Parameters:

v : image object

$\mathbf{N_v(m)}$: neighbors to an image object **v** of a class **m**

Expression:

$$\overline{\Delta}(v, N_v(m))$$

Feature value range:

[0, ∞]

4.4.3 Relations to Subobjects

These features refer to existing class assignments of image objects on a lower image object level in the image object hierarchy. Which of the lower image object levels to refer to can be determined by editing the feature distance.

> Class-Related features
> **Relations to sub objects**

Existence of

Checks if there is at least one subobject assigned to a defined class. If there is one, the feature value is 1 (= true), otherwise the feature value is 0 (= false).

> Class-Related features
> Relations to sub objects
> **Existence of**

Parameters:

v : image object

d : distance between neighbors

m : a class containing image objects

Formula:

0 if $S_v(d,m) = \varnothing$
1 if $S_v(d,m) \neq \varnothing$

Feature value range:

[0,1]

Number of

The number of subobjects assigned to a defined class.

> Class-Related features
> Relations to sub objects
> **Number of**

Parameters:

v : image object

d : distance between neighbors

m : a class containing image objects

Expression:

$\#S_v(d,m)$

Feature value range:

$[0, \infty]$

Area of

The absolute area covered by subobjects assigned to a defined class. If your data are georeferenced, the feature value represents the real area.

> Class-Related features
> Relations to sub objects
> **Area of**

Parameters:

d: distance

m: class

M: subobjects in class M

Expression:

$$\sum_{MES_v(d,m)} \#P_M$$

Feature Value Range:

[0;∞]

Rel. area of

The area covered by subobjects assigned to a defined class divided by the total area of the image object concerned.

> Class-Related features
> Relations to sub objects
> **Rel. area of**

Parameters:

d: distance

m: class

M: subobjects in class M

Expression:

$$\frac{\sum_{MES_v(d,m)} \#P_M}{\#P_v}$$

Feature Value Range:

[0;1]

Clark aggregation index

> Class-Related features
> Relations to sub objects
> **Clark aggregation index**

For a superobject the **Clark aggregation index** gives evidence about the spatial distribution of its subobjects of a certain class.

Parameters:

D(x) : mean spacial Distance to next neighbor of the subobjects of the class x

N(x) : Number of subobjects of class x

A: Number of pixels of the superobject (Area)

Obs_mean_dist : Observed mean distance of sub objects to their spatial nearest neighbor

$$\text{Obs\_mean\_dist} = \frac{\text{Sum}(D(x))}{N(x)}$$

Exp_mean_dist : Expected mean distance of sub objects to their spatial nearest neighbor

$$\text{Exp\_mean\_dist} = \frac{1}{2\sqrt{\frac{N(x)}{A}}}$$

CAI : Clark aggregation index

Formula:

$$\text{CAI} = \frac{\text{Obs\_mean\_dist}}{\text{Exp\_mean\_dist}}$$

Feature Value Range:

[0, 2.149]

0 : heavily clumped subobjects

1 : homogeneous spatial distribution of subobjects

2.149 : hexagonal distribution (edges of a honeycomb) of the subobjects

4.4.4 Relations to Superobjects

This feature refers to existing class assignments of image objects on a higher image object level in the image object hierarchy.

> Class-Related features
> **Relations to super objects**

Existence of

Checks if the superobject is assigned to a defined class. If this is true, the feature value is 1, otherwise 0.

> Class-Related features
> Relations to super objects
> **Existence of**

Parameters:

v : image object

d : distance between neighbors

m : a class containing image objects

Formula:

$$\begin{cases} \mathbf{0} \text{ if } U_v(d,m) = \varnothing \\ \mathbf{1} \text{ if } U_v(d,m) \neq \varnothing \end{cases}$$

Feature Value Range:

[0,1]

4.4.5 Relations to Classification

Membership to

In some cases it is important to incorporate the membership value to different classes in one class. This function allows explicit addressing of the membership values to different classes. If the membership value is below the assignment threshold, this value turns to 0.

Parameters:

v : image object

m : a class containing image objects

$\tilde{\phi}(v,m)$: stored membership value of an image object **v** to a class **m**

Expression:

$\tilde{\phi}(v,m)$

Feature value range:

[0,1]

Classified as

The idea of this feature is to enable the user to refer to the classification of an image object without regard to the membership value. It can be used to freeze a classification.

Parameters:

v : image object

m : a class containing image objects

Expression:

m(v)

Feature value range:

[0,1]

Classification value of

> Class-Related Features
> Relations to Classification
> **Classification value of**

This feature **Classification value of** allows you to explicitly address the membership values to all classes. As opposed to the feature **Membership to** it is possible to apply all membership values to all classes without restrictions.

Parameters:

v : image object

m : a class containing image objects

φ(v,m) : fuzzy membership value of an image object **v** to a class **m**

Expression:

φ(v,m)

Feature value range:

[0,1]

4.4.5.1 Class Name

> Class-Related Features
> Relations to Classification
> **Class Name**

The **Class name** feature returns the name of the class (or superclass) of an image object (or its superobject).

Parameters:

Distance in class hierarchy specifies the number of hierarchical levels when navigating from class to superclass. Using a distance of '0' the class name is returned, a distance of '1' will return the superclass name and so on.

Distance in image object hierarchy specifies the number of hierarchical levels when navigating from object to superobject. Using a distance of '0' the class of the image object is used as a starting point for the navigation in the class hierarchy, a distance of '1' will start at the class of the superobject.

4.4.5.2 Class Color

> Class-Related Features
> Relations to Classification
> **Class Color**

The **Class color** feature returns either the Red, Green or Blue color component of the class (or superclass) of an image object (or its superobject).

Parameters:

Color component is **Red, Green** or **Blue.**

Distance in class hierarchy specifies the number of hierarchical levels when navigating from class to superclass. Using a distance of '0' the class name is returned, a distance of '1' will return the superclass name and so on.

Distance in image object hierarchy specifies the number of hierarchical levels when navigating from object to superobject. Using a distance of '0' the class of the image object is used as a starting point for the navigation in the class hierarchy, a distance of '1' will start at the class of the superobject.

4.5 Scene Features

> Scene Features

4.5.1 Variables

All scene variables are listed here.

> Scene Features
> **Variables**

[name of a scene variable]

Define variables to describe interim values.

> Scene features
> Variables
> [**name of a global variable**]

4.5.2 Class-Related

> Scene features
> **Class-Related**

Number of classified objects

The absolute number of all image objects of the selected class on all image object levels.

> Scene features
> Class-Related
> **Number of classified objects**

Parameters:

V(m) : all image objects of a class **m**

m : a class containing image objects

Expression:

```
#V(m)
```

Feature value range:

[0,number of image objects]

Number of samples per class

The number of all samples of the selected class on all image object levels.

> Scene features
> Class-Related
> **Number of samples per class**

Parameters:

m : a class

Feature value range:

[0,number of samples]

Area of classified objects

The absolute area of all image objects of the selected class on all image object levels in pixel.

> Scene features
> Class-Related
> **Area of classified objects**
→ Area of classified objects page 115

Parameters:

v : image object

m : a class containing image objects

V(m) : all image objects of a class **m**

#P$_v$: total number of pixels contained in **P**$_v$

Expression:

$$\sum_{v \in V(m)} \#P_v$$

Feature Value Range:

[0,sx*sy]

Layer mean of classified objects

The mean of all image objects of the selected class on the selected image object levels.

> Scene features
> Class-Related
> **Layer mean of classified objects**

Parameters:

v : image object

m : a class containing image objects

V(m) : all image objects of a class **m**

$\bar{c}_k$**(v)** : mean intensity layer of an image object **v**

Expression:

$$\frac{1}{\#V(m)} \sum_{v \in V(m)} \bar{c}_k(v)$$

Feature value range:

[0,1]

Layer stddev. of classified objects

The standard deviation of all image objects of the selected class on the selected image object levels.

> Scene features
> Class-Related
> **Layer stddev. of classified objects**

Parameters:

v : image object

m : a class containing image objects

V(m) : all image objects of a class **m**

c_k**(v)** : image layer value of an image object **v**

Formula:

$$\sigma_k(V(m)) := \sqrt{\frac{1}{\#V(m)}\left(\sum_{v \in V(m)} (c_k(v))^2 - \frac{1}{\#V(m)} \sum_{v \in V(m)} c_k(v) \sum_{v \in V(m)} c_k(v)\right)}$$

Feature value range:

[0,1]

4.5.2.1 Class Variables

All class variables are listed here.

> Scene features
> Class-Related
> **Class Variable**

[name of a class variable]

A variable that use classes as values. In a rule set they can be used instead of an ordinary classes where needed.

> Scene features
> Class-Related
> Class Variable
> [**name of a class variable]**

4.5.3 Scene-Related

> Scene features
> **Scene-Related**

Existence of object level

Existence of a defined image object level. If the image object level with the given name exists within the project the feature value is 1 (= true), otherwise it is 0 (= false).

> Scene features
> Scene-Related
> **Existence of object level**

Parameter:

- Image object level name

Feature value range:

[0,1]

Existence of image layer

Existence of a defined image layer. If the image layer with the given alias exists within the project the feature value is 1 (= true), otherwise it is 0 (= false).

> Scene features
> Scene-Related
> **Existence of image layer**

Parameter:

- Image layer alias

Feature value range:

[0,1]

Existence of thematic layer

Existence of a defined thematic layer. If the thematic layer with the given alias exists within the project the feature value is 1 (= true), otherwise it is 0 (= false).

> Scene features
> Scene-Related
> **Existence of thematic layer**

Parameter:

- Thematic layer alias

Feature value range:

[0,1]

Mean of scene

Mean value for the selected layer.

> Scene features
> Scene-Related
> **Mean of Scene**

Expression:

$\overline{c_k}$

Stddev.

Standard deviation for the selected layer.

> Scene features
> Scene-Related
> **StdDev**

Expression:

σ_k

Smallest actual pixel value

Darkest actual intensity value of all pixel values of the selected layer.

> Scene features
> Scene-Related
> **Smallest actual pixel value**

Expression:

c'^{min}_k

Feature value range:

$[c_k^{min}, c_k^{max}]$

Smallest actual pixel value

Brightest actual intensity value of the selected layer.

> Scene features
> Scene-Related
> **Largest actual pixel value**

Expression:

c'^{max}_k

Feature value range:

$[c_k^{min}, c_k^{max}]$

Image size X

Vertical size **x** of the image in the display unit.

> Scene features
> Scene-Related
> **Image size X**

Expression:

```
sx
```

Image size Y

Horizontal size **y** of the image in the display unit.

> Scene features
> Scene-Related
> **Image size X**

Expression:

```
sy
```

Number of image layers

Number of layers **K** which are imported in the scene.

> Scene features
> Scene-Related
> **Number of objects**

Number of objects

Number of image objects of any class on all image object levels of the scene including unclassified image objects.

> Scene features
> Scene-Related
> **Number of objects**

Expression:

```
#V
```

Feature value range:

[0,1]

Number of pixels

Number of pixels in the pixel layer of the image.

> Scene features
> Scene-Related
> **Number of pixels**

Parameters:

sx : image size **x**

sy : image size **y**

(sx,sy) : scene size

Expression:

```
sx*sy
```

Feature value range:

[0,1]

Number of Samples

Number of all samples on all image object levels of the scene.

> Scene features
> Scene-Related
> **Number of samples**

Feature value range:

[0,number of samples]

Number of thematic layers

Number of layers **T** which are imported in the scene.

> Scene features
> Scene-Related
> **Number of thematic layers**

4.5.3.1 User name

This feature returns the user name.

> Scene features
> Scene-Related
> **User name**

Pixel Resolution

The resolution of the scene as given in the metadata of the project. The resulting number represents the size of a pixel in coordinate system unit.

The value is 1 if no resolution is set.

> Scene features
> Scene-Related
> **Pixel resolution**

4.6 Process-Related Features

> **Process-Related features**

4.6.1 Customized

> Process-Related features
> **Customized**

diff. PPO

> Process-Related features
> Customized
> **diff. PPO**

Parameters:

v : image object

f : any feature

ρ :

Formula:

$f(v) - f(\rho)$

Feature value range:

The range depends on the value of the feature in use.

Conditions:

If **f**(ρ) =**0** ∴ the formula is undefined

ratio PPO

> Process-Related features
> Customized
> **ratio PPO**

Parameters:

v : image object

f : any feature

ρ :

Formula:

$$\frac{f(v)}{f(\rho)}$$

Feature value range:

The range depends on the value of the feature in use.

Conditions:

If **f**(ρ) =**0** ∴ the formula is undefined

[name of a customized feature]

If existing, customized features referring to a Parent Process Object (PPO) are listed in the feature tree.

Border to PPO

> Process-Related features
> **Border to PPO**

Parameters:

b(v,ρ) : topological relation border length with the PPO

Formula:

$b(v,\rho)$

Feature value range:

[0,max size]

Elliptic Dist. from PPO

Measures elliptic distance of an object to its Parent Process Object (PPO).

> Process-Related features
> **Elliptic Dist. from PPO**

Parameters:

$\bar{x}_v$:

$\bar{y}_v$:

Formula:

$$\varepsilon_\rho(\bar{x}_v, \bar{y}_v)$$

Feature value range:

[0,∞]

Rel. border to PPO

The ratio of the border length of an image object shared with the Parent Process Object (PPO) to its total border length.

> Process-Related features
> **Rel. border to PPO**

Parameters:

$\mathbf{b}_v$: image object border length

$\mathbf{b}(v,\rho)$: topological relation border length with the PPO

Formula:

$$\frac{b(v,\rho)}{b_v}$$

Feature value range:

[0,1]

Same superobject as PPO

Checks whether this image object and its Parent Process Object (PPO) are parts of the same superobject.

> Process-Related features
> **Same super object as PPO**

Parameters:

v : image object

ρ :

$\mathbf{U}_v\mathbf{(d)}$: superobject of an image object **v** at a distance **d**

Formula:

$$\begin{cases} 1 : U_v(d) = U_\rho(d) \\ 0 : U_v(d) \neq U_\rho(d) \end{cases}$$

Feature value range:

[0,1]

4.7 Customized

> Customized

4.7.1 Largest possible pixel value

This feature returns the largest possible pixel value for a chosen layer. For example, the value displayed for a 8 bit image would be 255 and the value for a 16 bit image would be 65536.

> Customized
> **Create new "Largest possible pixel value"**

Parameter

Layer: Use the drop-down list to select an image layer.

4.7.2 Smallest possible pixel value

This feature returns smallest possible pixel value for a layer. This value will often be 0, but can be a negative value for some types of image data.

> Customized
> **Create new "Smallest possible pixel value"**

Parameter

Layer: Use the drop-down list to select a layer for which you want to display the lowest possible value.

[name of a metadata item]

A metadata item that can be used as a feature in rule set development.

> Metadata
>  **[name of a metadata item]**
→ **About Metadata as a Source of Information** on page 188

To make external metadata available to the feature tree, you have to convert it within data import procedures to get an internal metadata definitio

→ **User Guide** sections: **Create Project** and **Customized Import**

4.8 Metadata

All metadata items are listed here.

> **Metadata**

4.9 Feature Variables

All feature variables are listed here.

> **Feature Variables**

[name of a feature variable]

A variable that use features as values.

> Feature variables
> [**name of a feature variable**]

In a rule set they can be used like that feature. It returns the same value as the feature to which it points. It uses the unit of whatever feature is assigned as a variable. It is possible to create a feature variable without a feature assigned, but the calculation value would be invalid.

4.10 Use Customized Features

Customized features allow you to create new features that are adapted to your needs. Customized features are composed of arithmetic and relational features. All customized features are based on the features shipped with **Definiens Developer** as well as newly created customized features.

- **Arithmetic features**, are composed of existing features, variables (**Definiens Developer** only), and constants, which are combined via arithmetic operations. Arithmetic features can be composed of multiple features but apply only to a single object.
- **Relational features**, are used to compare a particular feature of one object to those of related objects of a specific class within a specified distance. Related objects are surrounding objects (neighbors), sub-objects, superobjects, sub-objects of a superobject or a complete image object level. Relational features are composed of only a single feature but refer to a group of related objects.

4.10.1 Create Customized Features

The **Manage Customized Features** dialog box allows you to add, edit, copy, and delete customized features. It enables you to create new arithmetic as well as relational features based on the existing ones.

1. To open the **Manage Customized Features** dialog box, do one of the following:
   - On the menu bar click on **Tools** and then select **Manage Customized Features.**
   - On the **Tools** toolbar click on the **Manage Customized Features** icon.


Manage Customized Features

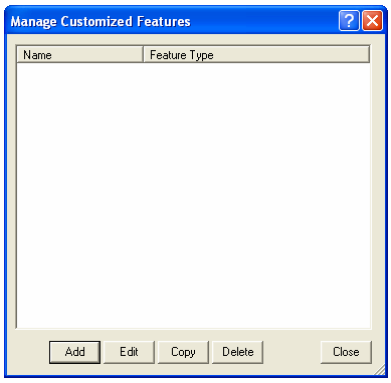


Figure 77: Manage Customized Features dialog box.

2. Click **Add** to create a new customized feature. The **Customized Features** dialog box will open, providing you with tools for the creation of arithmetic and relational features.

3. To edit a feature first you need to select it and then click **Edit** . This opens the **Customized Features** dialog in which you can modify the feature.

4. To copy or delete a feature you first need to select it and then depending on the action you want to perform you click either **Copy** or **Delete.**

Find Out More

Where Else to Find Customized Features

*Newly created features can also be found under **Customized** in the **Feature View**. To edit a customized feature, right-click the respective feature and select **Edit Feature**. To delete the feature, select **Delete Feature**.*

*New customized features can be named and saved separately. Use **Tool > Save Customized Features** and **Tools > Load Customized Features** to reuse customized features.*

4.10.2 Arithmetic Customized Features

The procedure below guides you through the steps you need to follow when you want to create an arithmetic customized feature.

1. Open the **Manage Customized Features** dialog box and click **Add**. The **Customized Features** dialog opens, make sure you currently viewing the **Arithmetic** tab.

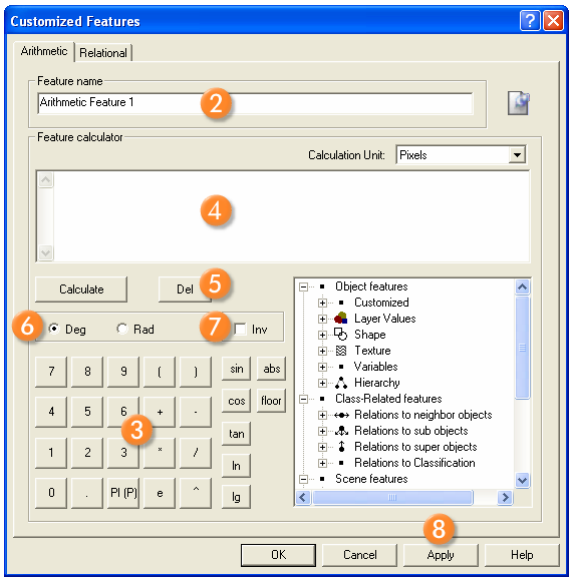


Figure 78: Creating an arithmetic feature in the Customized Features dialog box.

2. Insert a name 2 for the customized feature to be created.
3. Use the calculator 3 to create the arithmetic expression. You can:
   - Type in new constants.
   - Select features or variables (**Definiens Developer** only) in the feature tree on the right.
   - Choose arithmetic operations or mathematical functions.

Find Out More

About Calculating Customized Features

The calculator provides the following arithmetic operations and mathematical functions:

+ *addition*

– *subtraction*

***** *multiplication*

/ *division*

^ *power of (e.g. x^2 means x^2). You can use x^0.5 for the square root of x.*

sin *trigonometric function sine*

cos *cosine*

tan *tangent*

ln *natural logarithm to base e*

lg *logarithm to base 10*

abs *for absolute value*

floor *to round down to the next lowest integer (whole value). You can use floor(0.5+x) to round to the next integer value.*

4. The expression you create is displayed 4 at the text area above the calculator.

5. To calculate or delete 5 an arithmetic expression first you need to highlight the expression with the cursor and then click either **Calculate** or **Del** depending on the action you want to take.
6. You can switch between degrees (**Deg**) or radians (**Rad**) 6 measurements.
7. You can invert 7 the expression.
8. To create the new customized feature do one of the following:
   - Click **Apply** 8 to create the feature without leaving the dialog box or
   - Click **OK** to create the feature and close the dialog box.
9. After creation, the new arithmetic feature can be found in either one of the following locations:
   - In the **Image Object Information** window
   - In the **Feature View** window under **Object features>Customized.**

Note

Avoid invalid operations such as division by 0. Invalid operations will result in undefined values.

4.10.3 Relational Customized Features

The following procedure will assist you with the creation of a relational customized feature.

1. Open the **Manage Customized Features** dialog box and click **Add**. The **Customized Features** dialog opens, make sure you currently viewing the **Relational** tab.
2. Insert a name 2 for the relational feature to be created.
3. Select the relation 3 existing between the image objects.
4. Choose the relational function 4 to be applied.
5. Define the distance 5 of the related image objects. Depending on the related image objects, he distance can be either horizontal (units, e.g. pixels) or vertical (image object levels)
6. Select the feature 6 for which to compute the relation.

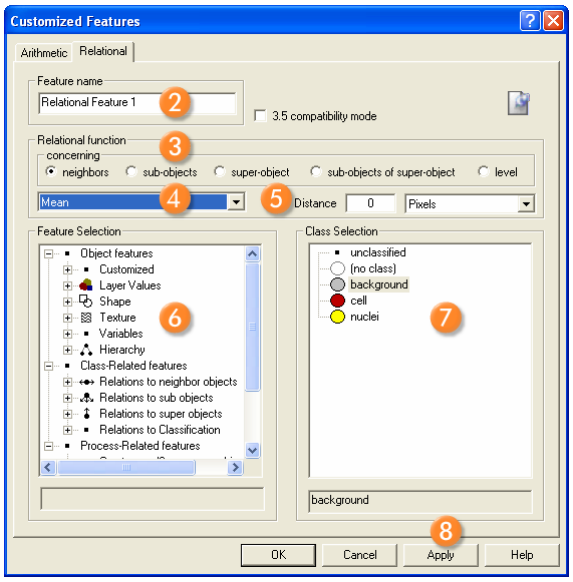


Figure 79: Creating a relational feature at the Customized Features dialog box.

7. Select a class, a group or **no class** 7 to apply the relation.
8. To create the new customized feature do one of the following:
   - Click **Apply** 8 to create the feature without leaving the dialog box or
   - Click **OK** to create the feature and close the dialog box.
9. After creation, the new relational feature can be found in the **Feature View** window under **Class-Related features > Customized.**

Note

As with class-related features, the relations refer to the groups hierarchy. This means if a relation refers to one class, it automatically refers to all subclasses of this class in the groups hierarchy.

Relations between surrounding objects can exist either on the same level or on a level lower or higher in the image object hierarchy:

neighbors	Related image objects on the same level. If the distance of the image objects is set to 0 then only the direct neighbors are considered. When the distance is greater than 0 then the relation of the objects is computed using their centers of gravity. Only those neighbors whose center of gravity is closer than the distance specified from the starting image object are considered. The distance is calculated either in metric units or pixels. For example, a direct neighbor might be ignored if its center of gravity is further away from the specified distance.
subobjects	Image objects that exist under other image objects (superobjects) whose position in the hierarchy is higher. The distance is calculated in levels.
superobject	Contains other image objects (subobjects) on lower levels in the hierarchy. The distance is calculated in levels.

sub-objects of superobject	Only the image objects that exist under a specific super-object are considered in this case. The distance is calculated in levels.
level	Specifies the level on which an image object will be compared to all other image objects existing at this level. The distance is calculated in levels.

The following table gives an overview of all functions existing in the drop-down list under the **Relational function** section:

Mean	Calculates the mean value of selected features of an image object and its neighbors. You can select a class to apply this feature or no class if you want to apply it to all image objects.
Standard deviation	Calculates the standard deviation of selected features of an image object and its neighbors. You can select a class to apply this feature or no class if you want to apply it to all image objects.
Mean difference	Calculates the mean difference between the feature value of an image object and its neighbors of a selected class. Note that for averaging, the feature values are weighted by the size of the respective objects.
Mean absolute difference	Calculates the mean absolute difference between the feature value of an object and the feature values of its neighbors of a selected class. Note that for averaging, the absolute difference to each neighbor is weighted by the respective size.
Ratio	Calculates the proportion between the feature value of an image object and the mean feature value of its neighbors of a selected class. Note that for averaging the feature values are weighted with the size of the corresponding image objects.
Sum	Calculates the sum of the feature values of the neighbors of a selected class.
Number	Calculates the number of neighbors of a selected class. The feature you have selected is of no account. But it has to be selected for working of the feature.
Min	Returns the minimum value of the feature values of an image object and its neighbors of a selected class.
Max	Returns the minimum value of the feature values of an image object and its neighbors of a selected class.
Mean difference to higher values	Calculates the mean difference between the feature value of an image object and the feature values of its neighbors of a selected class, which have higher values than the image object itself. Note that for averaging the feature values are weighted by the size of respective image objects.
Mean difference to lower values	Calculates the mean difference between the feature value of an image object and the feature values of its neighbors of a selected class, which have lower values than the object itself. Note that for averaging the feature values are weighted by the size of the respective image objects.
Portion of higher value area	Calculates the portion of the area of the neighbors of a selected class, which have higher values for the specified feature than the object itself to the area of all neighbors of the selected class.
Portion of lower value area	Calculates the portion of the area of the neighbors of a selected class, which have lower values for the specified feature than the object itself to the area of all neighbors of the selected class.

Portion of higher values	Calculates the feature value difference between an image object and its neighbors of a selected class with higher feature values than the object itself divided by the difference of the image object and all its neighbors of the selected class. Note that the features are weighted with the size of the corresponding image objects.
Portion of lower values	Calculates the feature value difference between an image object and its neighbors of a selected class with lower feature values than the object itself divided by the difference of the image object and all its neighbors of the selected class. Note that the features are weighted with the size of the corresponding image object.

4.11 Use Variables as Features

The following variables can be used as features:

→ **Use Variables** section of the **User Guide**

- Scene variables → **Variables** on page 173
- Object variables → **Variables** on page 160
- Feature variables → **Feature Variables** on page 182

They display in the feature tree of for example the **Feature View** window or the **Select Displayed Features** dialog box.

4.12 About Metadata as a Source of Information

Many image data formats include metadata providing information about the related image, for example the acquisition time. Considering metadata might be beneficial for image analysis if you relate it to features.

The available metadata depends on the image reader or camera used, the industry-specific environment, and settings. Industry-specific examples are:

- Satellite image data may contain metadata providing cloudiness information.
- Microscopy image data may contain metadata providing information about the used magnification.

Definiens Developer can provide a selection of the available metadata. This selection is defined in a metadata definition which is part of the rule set.

The provided metadata can be displayed in the **Image Object Information** window. Further, it is listed together with features and variables in the feature tree of for example the **Feature View** window or the **Select Displayed Features** dialog box.

Convert Metadata to Provide it to the Feature Tree

When importing data, you can provide a selection of available metadata. To do so, you have to convert external metadata to an internal metadata definition. This provides a selection of the available metadata to the feature tree and allows its usage in rule set development.

When developing rule sets, metadata definitions will be included in rule sets allowing the serialization of metadata usage.

Metadata conversion is available within the following import functions:

- Within the **Create Project** dialog box.
- Within the **Customized Import** dialog box on the **Metadata** tab.

4.13 Table of Feature Symbols

This section contains a complete feature symbols reference list.

4.13.1 Basic Mathematical Notations

Basic mathematical symbols used in expressions:

$:=$	Definition
$\therefore$	Therefore
$\varnothing$	Empty set
$a \in A$	**a** is an element of a set **A**
$b \notin B$	**b** is not an element of a set **B**
$A \subset B$	Set i**A**s a proper subset of set **B**
$A \not\subset B$	Set **A** is not a proper subset of set **B**
$A \subseteq B$	Set **A** is a subset of set **B**
$A \cup B$	Union of sets **A** and **B**
$A \cap B$	Intersection of sets **A** and **B**
$A \setminus B$	A symmetric difference of sets **A** and **B**
$\#A$	The size of a set **A**
$\exists$	It exists, at least one
$\forall$	For all
$\Rightarrow$	It follows
$\Leftrightarrow$	Equivalent
$\sum_{i=1}^{u}$	Sum over index **i**
[a,b]	Interval with $\{ x \mid a \leq x \leq b \}$

4.13.2 Images and Scenes

Variables used to represent image objects and scenes.

$k = 1,..., K$	Image layer **k**
$t = 1,..., T$	Thematic layer **t**
(x,y)	Pixel coordinates
(sx,sy)	Scene size
$c_k(x,y)$	Image layer value at pixel **(x,y)**
c_k^{max}	Brightest possible intensity value of layer **k**
c_k^{min}	Darkest possible intensity value of layer **k**
c_k^{range}	Data range of layer **k**
$\bar{c}_k$	Mean intensity of layer **k**
σ_k	Std. deviation of layer **k**
$N_4(x,y)$	4-pixel Neighbors **(x,y)**
$N_8(x,y)$	8-pixel Neighbors **(x,y)**

4.13.3 Image Objects Hierarchy

Variables that represent the relations between image objects.

u, v	Image objects
$U_v(d)$	Superobject of an image object **v** at a distance **d**
$S_v(d)$	Subobjects of an image object **v** at a distance **d**
V_i **,** $i=1,...,n$	Image object level
N_v	Direct neighbors of an image object **v**
$N_v(d)$	Neighbors of an image object **v** at a distance **d**
$e(u,v)$	Topological relation between the image objects **u** and **v**

4.13.4 Image Object as a Set of Pixels

Variables representing an image object as a set of pixels.

P_v	Set of pixels of an image object **v**
$\#P_v$	Total number of pixels contained in $\mathbf{P}_v$
P_v^{Inner}	Inner border pixels of $\mathbf{P}_v$
P_v^{Outer}	Outer border pixels of $\mathbf{P}_v$

4.13.5 Bounding Box of an Image Object

Variables that represent the boundaries of an image object:

B_v	Bounding box of an image object **v**
$B_v(d)$	Extended bounding box of an image object **v** with distance **d**
$x_{min}(v)$	Minimum **x** coordinate of **v**
$x_{max}(v)$	Maximum **x** coordinate of **v**
$y_{min}(v)$	Minimum **y** coordinate of **v**
$y_{max}(v)$	Maximum **y** coordinate of **v**
b_v	Image object border length
b(v,)	Topological relation border length

4.13.6 Layer Intensity on Pixel Sets

Variables representing the layer intensity.

S	Set of pixels
O	Set of image objects
$\bar{c}_k(S)$	Mean intensity of layer **k** of a set **S**
$\sigma_k(S)$	Standard deviation of a set **S**
$\bar{c}(S)$	Brightness
w_k^B	Brightness weight of layer **k**
$\bar{\Delta}_k(v,O)$	Mean difference of an image object **v** to image objects in a set **O**

4.13.7 Class Related Sets

Variables representing the relation between classes:

M	Set of classes **M={m1,..., m_a}**
m	A class,**(m∈M)**
$N_v(d,m)$	Neighbors of class **m** within a distance **d**
$S_v(d,m)$	Subobjects of class **m** with hierarchical distance **d**
$U_v(d,m)$	Superobject of class **m** with hierarchical distance **d**
$V_i(m)$	All image objects at level **i** of class **m**
$\phi(v,m)$	Fuzzy membership value of object an image object **v** to a class **m**
$\tilde{\phi}(v,m)$	Stored membership value of an image object **v** to a class **m**

5 Index