DEFINIENS®

The Image Intelligence Company

Definiens

Professional 5

Reference Book

Definiens AG

www.definiens.com

Imprint and Version

Document Version 5.0.6.1

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 **www.definiens.com**

Dear User,

thank you for using **Definiens Professional** 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 **www.definiens.com/support/index.htm**.

Thank you.

Legal Notes

Table of Contents

1 Algorithms Reference

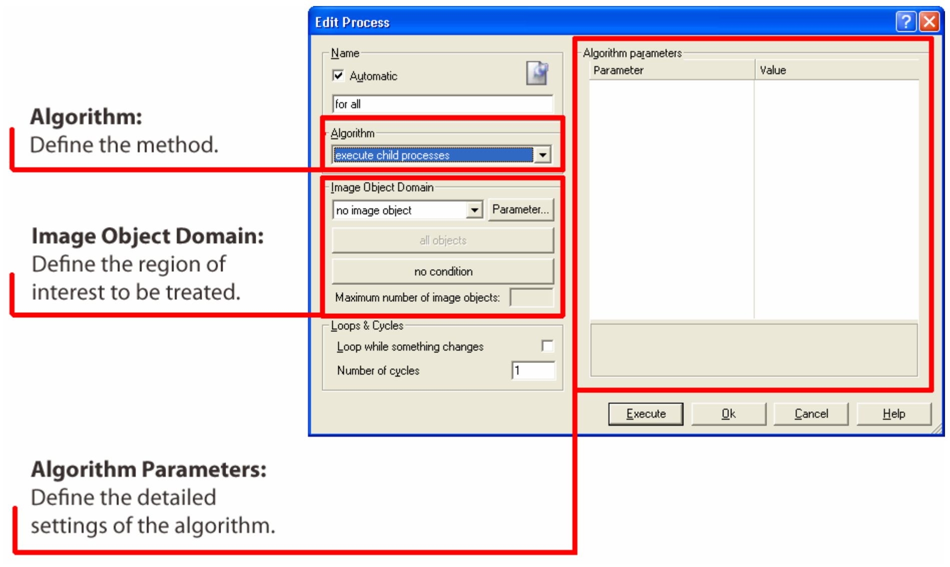
Contents in This Chapter

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 refered to as the current image object.

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

- 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.


Edit Process
Algorithm parameters
Parameter
Value
Execute
Ok
Cancel
Help
Parameter...
Algorithm: Define the method.
Image Object Domain: Define the region of interest to be treated.
Algorithm Parameters: Define the detailed settings of the algorithm.

Name
☑ Automatic
for all

Algorithm
execute child processes

Image Object Domain
no image object
all objects
no condition
Maximum number of image objects:

Loops & Cycles
Loop while something changes ☐
Number of cycles 1

Figure 1: Edit Process dialog box with highlighted group boxes.

Specify Algorithm Parameters

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

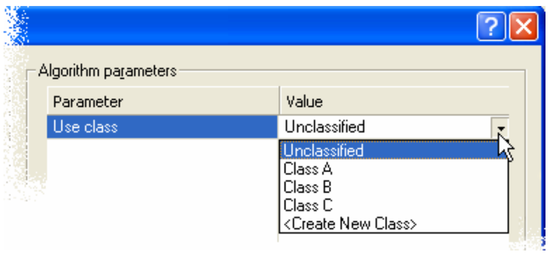
1. Define the individual settings of the algorithm in the **Algorithms Parameters** 2 group box. If available, click a plus sign (+) button to expand the table to access additional parameters.

⊞ (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 placed inside the value field. A dialog box opens allowing you to configure the value .
   - Click the drop-down arrow button placed inside the value field. Select from a drop-down list to configure the value.

... (**ellipsis button**)

▾ (**drop-down arrow button**)



Parameter	Value
Use class	Unclassified

Unclassified
Class A
Class B
Class C
<Create New Class>

Figure 2: Select an Algorithm Parameter for editing .

1.1 Process Related Algorithms

1.1.1 Execute Child Processes

Execute all child processes of the process.

▪ **execute child processes**

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 an container for a sequence of functional related processes.

Use the **execute child processes** algorithm in conjunction with other image object domains (e.g. 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 cases the child processes usually use one of the following as image object domain: **current image object, neighbor object, super object, sub objects**.

1.2 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.

eCognition 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.

1.2.1 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 this grid lines.

Object Size

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

Level Name

Enter the name for the new image object level.

Condition: 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.

Condition: 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.

Example

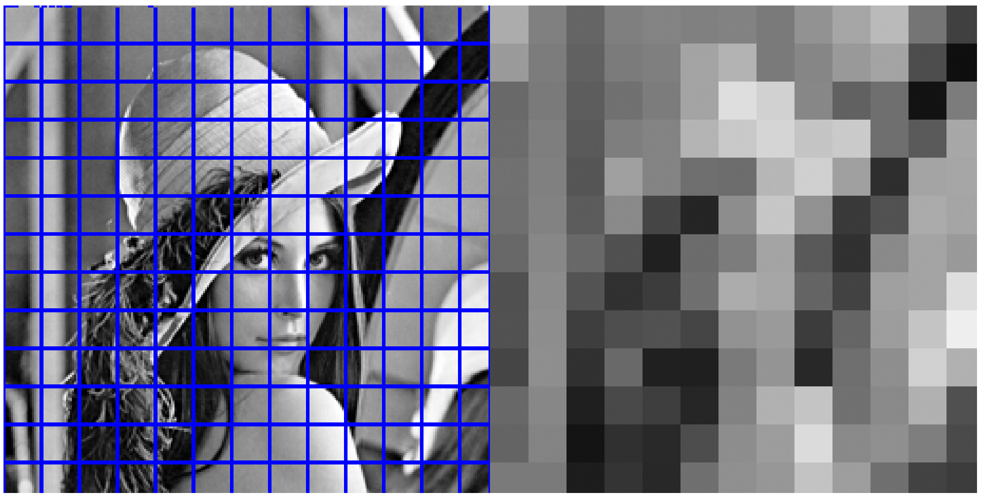


Figure 3: Result of chessboard segmentation with object size 20.

1.2.2 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.

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. Condition: 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.

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

Level Name

Enter the name for the new image object level.

Condition: 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.

Condition: Thematic layers must be available.

Example

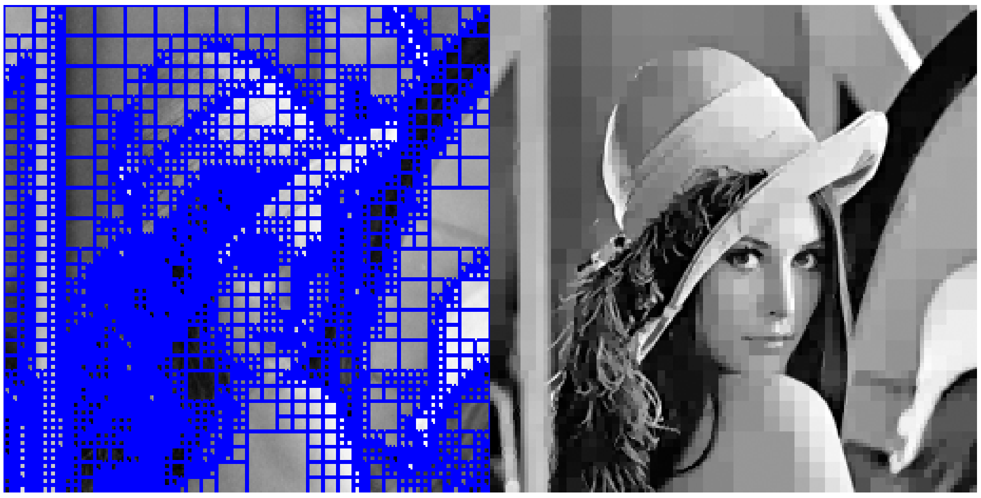


Figure 4: Result of quad tree based segmentation with mode color and scale 40.

1.2.3 Multiresolution Segmentation

The **multiresolution segmentation** algorithm is a heuristic 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.

multiresolution segmentation

Level Name

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

Condition: 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.

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

Image Layer Weights

This parameter groups allows you to assess image layers differently 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. Consequently, image layers that do not contain the information intended for representation by the image objects should be given little or no weight. Along with the image layer weight, the standard deviations of the image layer values for each single image layer over the entire scene are displayed to provide information about the layer dynamics.

Example: When segmenting a LANDSAT scene, 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.

Condition: 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.

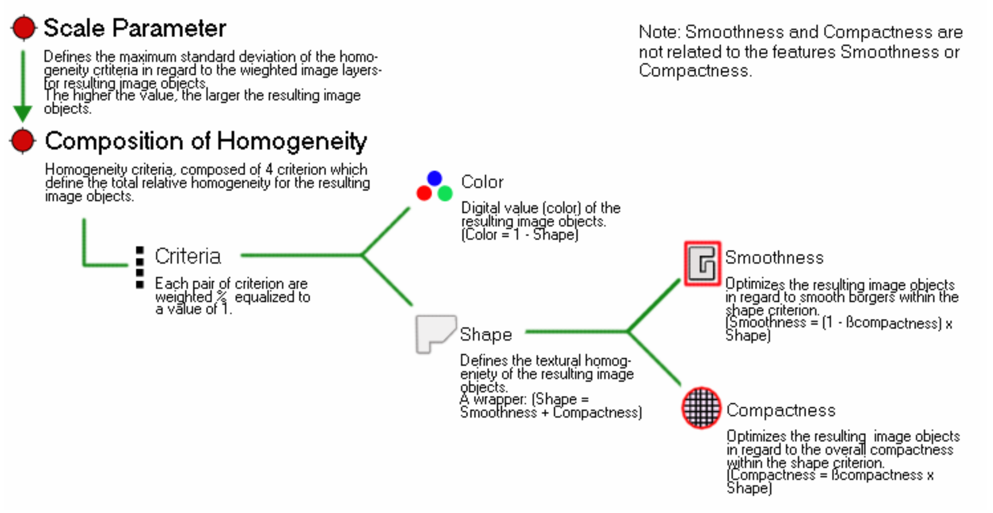
Tip

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 criterion are especially helpful in avoiding highly fractured image object results in strongly textured data (e.g., radar data).



Defines the maximum standard deviation of the homogeneity crtiteria in regard to the wieghted image layers-for resulting image objects.
The higher the value, the larger the resulting image objects.

Homogeneity criteria, composed of 4 criterion which define the total relative homogeneity for the resulting image objects.

Each pair of criterion are weighted % equalized to a value of 1.

Digital value (color) of the resulting image objects.
(Color = 1 - Shape)

Defines the textural homogeniety of the resulting image objects.
A wrapper: (Shape = Smoothness + Compactness)

Optimizes the resulting image objects in regard to smooth borgers within the shape criterion.
(Smoothness = (1 - ßcompactness) x Shape)

Optimizes the resulting image objects in regard to the overall compactness within the shape criterion.
(Compactness = ßcompactness x Shape)

Note: Smoothness and Compactness are not related to the features Smoothness or Compactness.

Figure 5: 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.

Example

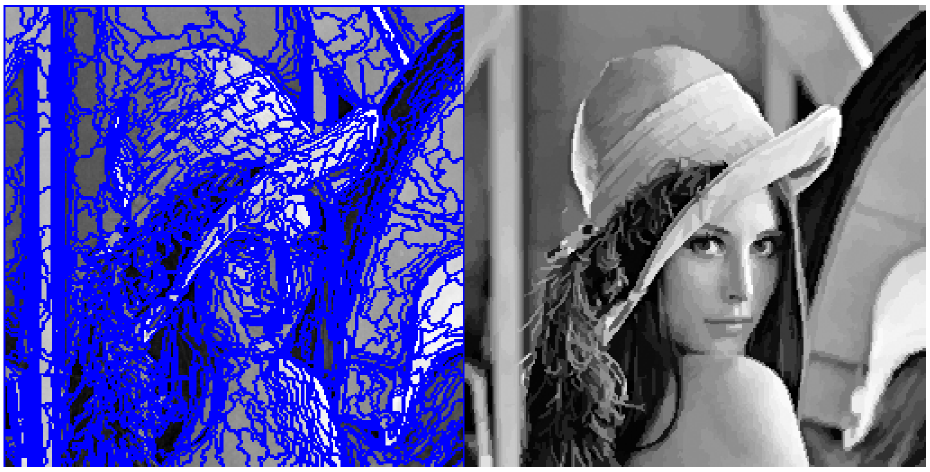


Figure 6: Result of multiresolution segmentation with scale 10, shape 0.1 and compactness 0.5.

1.2.4 Spectral Difference Segmentation

The **spectral difference segmentation** algorithm is used to merge neighboring objects according to their mean layer intensity values. Neighboring objects are merged, if their difference in layer mean intensities is below this value given by the mean spectral difference. This algorithm is designed to refine existing segmentation results, by merging spectrally similar image objects produced by previous segmentations.

spectral difference segmentation

> ***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.

Condition: 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

This parameter groups allows you to assess image layers differently 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. Consequently, image layers that do not contain the information intended for representation by the image objects should be given little or no weight. Along with the image layer weight, the standard deviations of the image layer values for each single image layer over the entire scene are displayed to provide information about the layer dynamics.

Example: When segmenting a LANDSAT scene, 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.

Condition: Thematic layers must be available.

1.3 Basic Classification Algorithms

Classification algorithms compute the membership value of an image object to one or more classes and modify the object classification based on this information.

1.3.1 Classification

The **classification** algorithm 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.

classification

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 options 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 15

1.3.2 Assign Class

Assign all objects of the image object domains 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 .

assign class

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.

1.3.3 Hierarchical Classification

The **hierarchical classification** algorithm evaluates the membership value of an image object to a list of selected classes.

hierarchical classification

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 Profession 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.

1.4 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.

1.4.1 Find Enclosed by Image Object

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

find enclosed by image object

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.

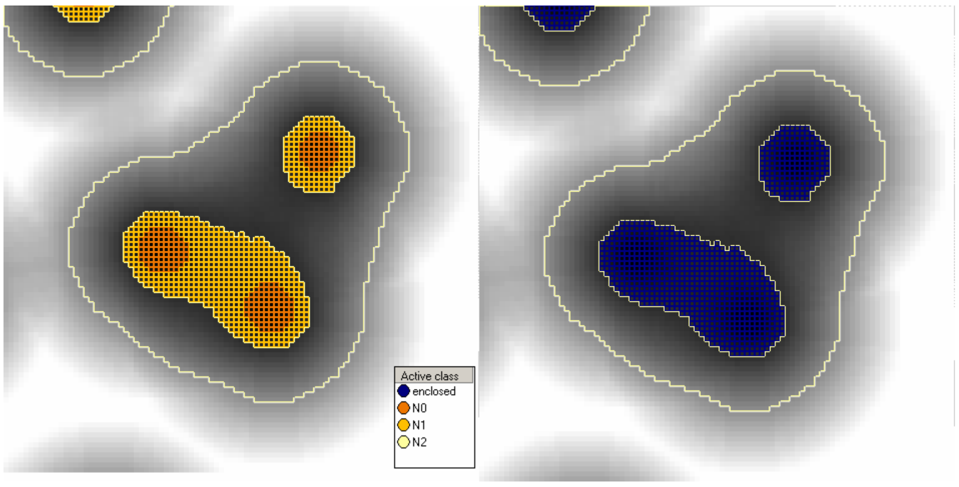
Classification Settings

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

See **classification** algorithm for details.

→ Classification on page 14

Example



Left: Input of **find enclosed by image object**:image object domain: **image object level**, class filter: **N3**. Right: Result of find enclosed by image object:enclosed objects get classified with the class **enclosed.** Note that the objects at the upper image border are classified as **enclosed**.

1.4.2 Find Enclosed by Class

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

find enclosed by class

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.

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.

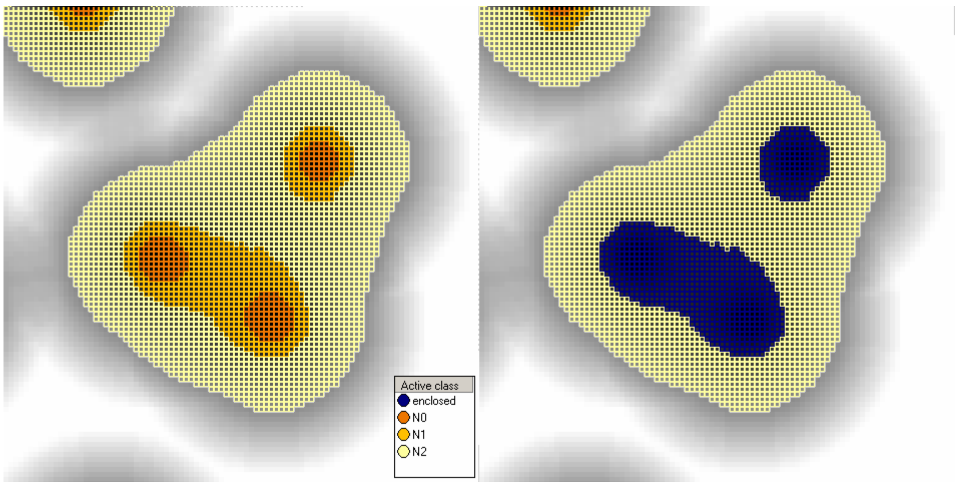
Classification Settings

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

See **classification** algorithm for details.

→ Classification on page 14

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**.

1.4.3 Find Local Extrema

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

find local extrema

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.

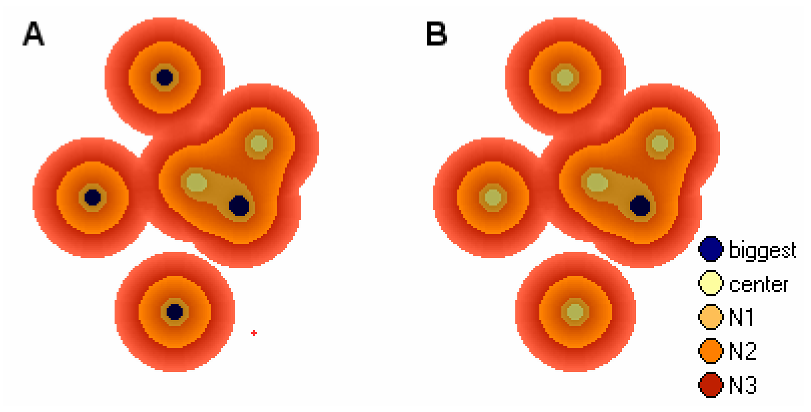
See **classification** algorithm for details.

→ Classification on page 14

Note

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

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

1.4.4 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.

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 behaviour 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 14

> **_Note_**
>
> _At least one class needs to be selected in the active class list for this algorithm_

Example

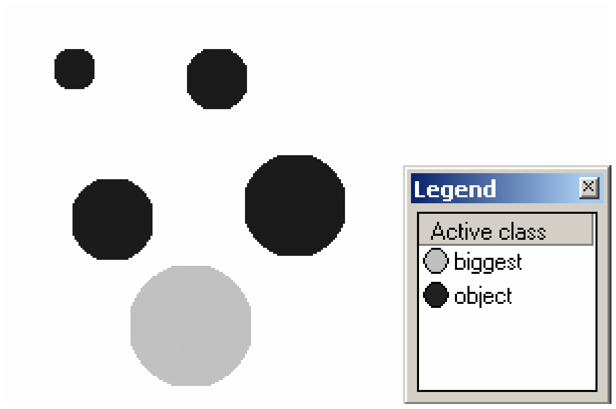


Figure 7: Result of find domain extrema using Extrema Type Maximum and Feature Area.

1.4.5 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 14

1.5 Level Operation Algorithms

Level operating algorithms allow you to modify entire image object levels within the image object hierarchy.

1.5.1 Copy Image Object Level

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

copy image object level

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.

1.5.2 Delete Image Object Level

Delete the image object level selected by the image objects domain.

delete image object level

1.5.3 Rename Image Object Level

Renames an image object level.

rename image object level

Level to Rename

Select the image object level to be renamed.

New Level New

Edit the new name for the image object level.

1.6 Reshaping Operation 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.

1.6.1 Grow Region

Enlarges image objects defined in the image object domain by neighboring image objects. The image object will be enlarged by merging in all neighboring image objects (candidates) that match the criteria specified in the parameters.

grow region

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.

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

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.

Example

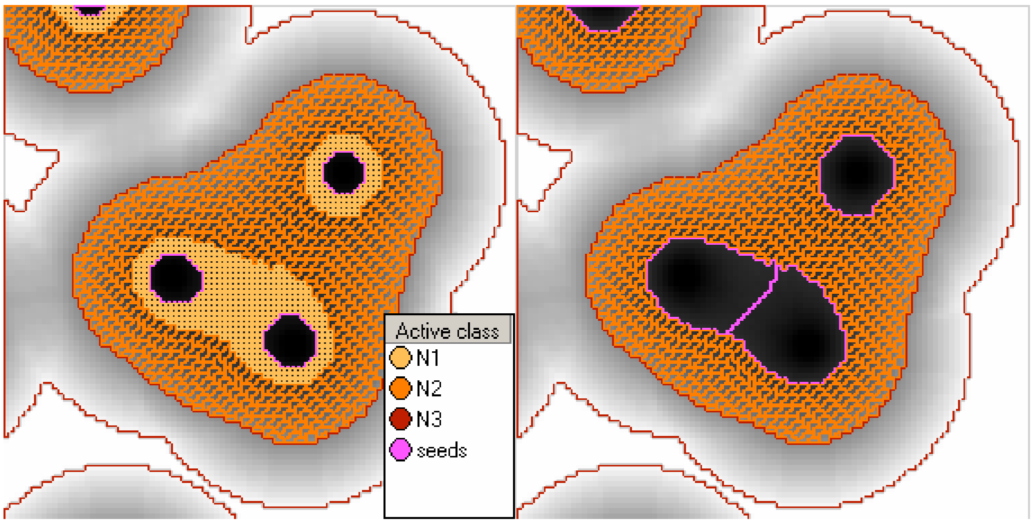


Figure 8: Result of looped grow region algorithm on image objects of class seed and candidate class N1.
Note that the two seed objects in the image center grow to fill the entire space originally covered by objects of class N1 while still being two separate objects.

1.6.2 Merge Region

Merge all image objects chosen in the image object domain.

merge region

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.

Example

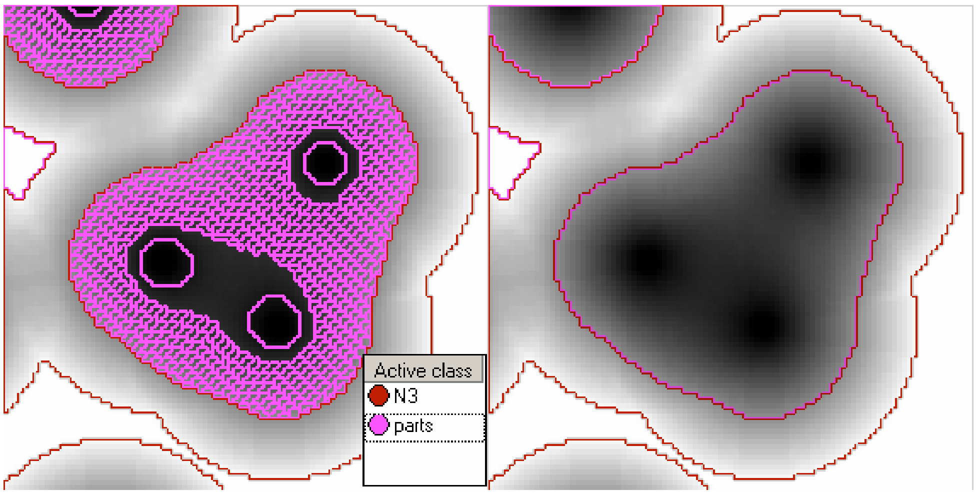


Figure 9: Result of merge region algorithm on all image objects classified as parts.

1.6.3 Convert to Subobjects

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

convert to subobjects

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

1.6.4 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.

border optimization

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 14

1.7 Vectorization Algorithms

Vectorization algorithms are used to create and delete polygons.

Polygons are vector objects provide detailed information for the characterization of image objects by their shape. They are also needed for the export of image object outlines into vector file formats.

1.7.1 Create Polygons

Defines basic parameters used to create polygons.

create polygons

> ***Note***
>
> *A polygon view is only available for image object levels with polygons, while outlines of image objects can be shown after the first segmentation. Furthermore, if the polygons cannot be clearly distinguished due to a low zoom factor, they are automatically deactivated. Therefore, a higher zoom factor must be chosen in order to see the polygons.*

Threshold

Set the degree of abstraction for the base 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.

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.

1.7.2 Delete Polygons

Delete the polygons that are defined by the image object domain.

delete polygons

> ***Note***
>
> *The polygons for the entire image object hierarchy will be deleted.*

1.8 Sample Operation Algorithms

Use sample operation algorithms to perform sample operations.

1.8.1 Classified Image Objects to Samples

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

classified image objects to samples

1.8.2 Cleanup Redundant Samples

Remove all samples which membership value is higher than the given 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.*

1.8.3 Nearest Neighbor Configuration

Add a nearest neighbor classifier with specific configuration to each selected class.

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.

1.9 Thematic Layer Operation Algorithms

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

1.9.1 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.

- **synchronize image object hierarchy**

Thematic Layers

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

1.9.2 Not Available

1.9.3 Write Thematic Attributes

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

- **write thematic attributes**

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.

1.10 Export Algorithms

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

1.10.1 Export Classification View

Export the classification view to file.

- **export classification view**

Export 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 Geo information.

Default Export Driver

Select the export file type

Desktop Export Folder

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

1.10.2 Export Current View

Export the current project view to file.

- **export current view**

Export 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.

... **(ellipsis button)**

Default Export Driver

Select the export file type.

Desktop Export Folder

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

1.10.3 Export Thematic Raster Files

Export thematic raster files.

▪ **export thematic raster files**

Export 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 as many features as you like.

Default Export Driver

Select the export file type.

Desktop Export Folder

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

1.10.4 Export Domain Statistics

Export a domain statistics to file.

▪ **export domain statistics**

Export Name

Use default name or edit it.

Features

Select as many features as you like.

Default Export Driver

Select the export file type.

Statistical Operations

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

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

Desktop Export Folder

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

1.10.5 Export Project Statistics

Exports statistics concerning the project to file.

- **export project statistics**

Export Name

Use default name or edit it.

Features

Select as many features as you like.

Default Export Driver

Select the export file type.

Desktop Export Folder

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

1.10.6 Export Object Statistics

Export object statistics to file.

▪ **export object statistics**

Export Name

Use default name or edit it.

Features

Select as many features as you like.

Default Export Driver

Select the export file type.

Desktop Export Folder

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

1.10.7 Export Vector Layers

Export vector layers to file.

▪ **export vector layers**

Export Name

Use default name or edit it.

Features

Select as many features as you like.

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**

Default Export Driver

Select the export file type.

Desktop Export Folder

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

2 Features Reference

Contents in This Chapter

2.1 Basic Features Concepts

Basic features concepts offers an overview of the most important features available in Definiens.

2.1.1 Image Layers Related Features

Images and Scenes

Scene

Scene is a rectangular area in a 2D space. It has an origin $\mathbf{(x_0, y_0)}^{geo}$ an extension $\mathbf{sx}^{geo}$ in **x** and an extension $\mathbf{sy}^{geo}$ in **y**. The size of a pixel (in meters) is denoted in $\mathbf{u}_{geo}$. If a scene is geo-coded, $\mathbf{(x_0, y_0)}^{geo}$, $\mathbf{sx}^{geo}$, $\mathbf{sy}^{geo}$ or in geo-coordinates (i.e. these values refer to the coordination system defined by the geo-coding). If a scene contains pixel coordinates then $\mathbf{(x_0, y_0)}$ is its origin and **sx, sy** is its size (in pixels).The formula is defined as follows:

$$(x_0, y_0)^{geo} = (x_0, y_0)^{geo} + (x\text{-}y)^{pxl} * u_{geo}$$

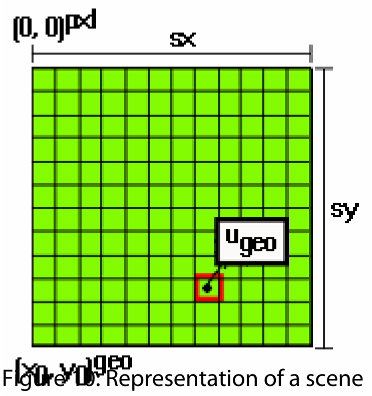


Figure 10: Representation of a scene

Scenes can consist of an arbitrary number of image layers **(k =1,...,K)** and thematic layers **(t=1,...,T)**. The pixel value (layer intensity) of an image layer **k** at pixel **(x,y)** is denoted as **c_k(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 c_k^{min}, whereas the largest possible value as c_k^{max}. The dynamic range is given by $c_k^{range}:=c_k^{max} - c_k^{min}$. The supported layer data types are:

Type	c_k^{min}	c_k^{max}	c_k^{range}
8-bit unsigned (int)	0	255	255
16-bit unsigned (int)	0	65535	65535
16-bit signed (int)	-32767	32767	65534
32-bit unsigned (int)	0	4294967295	4294967295
32-bit signed (int)	-2147483647	2147483647	4294967294
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.

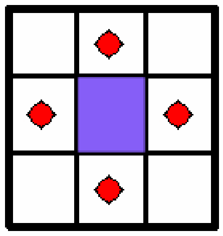


Figure 11: 4-pixel neighborhood

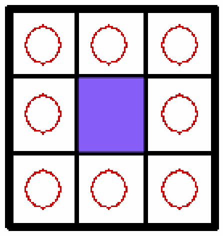


Figure 12: 8-pixel neighborhood

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

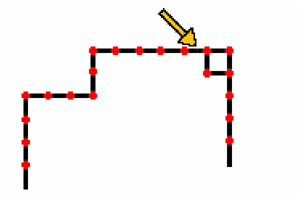


Figure 13: Pixel Borders

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:

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

The standard deviation is defined as:

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

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

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

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:

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

2.1.2 Image Object Related Features

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.

$$\boxed{\forall_{i<j} \ \forall_{v \in V_i} \ \exists_{u \in V_j} \ P_v \subset P_u}$$

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 **$U_v(d)$**. Similar, all subobjects with a level distance **d** is denoted as **$S_v(d)$**.

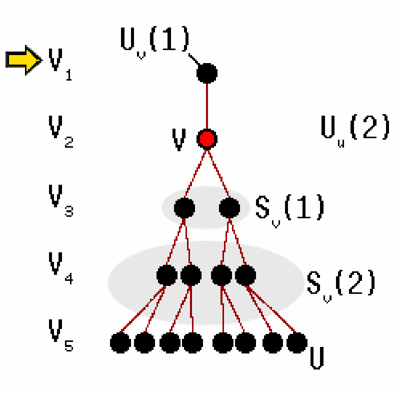


Figure 14: Image object Hierarchy

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

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

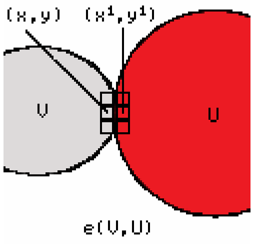


Figure 15: Topological relation between neighbors

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 $\mathbf{N_v(d)}$.

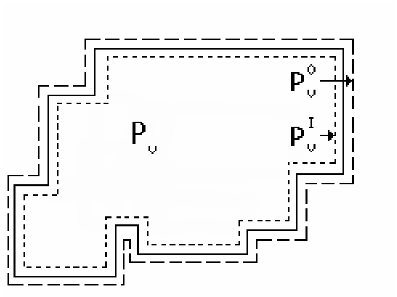


Figure 16: Boundaries of an image object v.

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 $\mathbf{P_v}^{\text{Inner}}$.

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

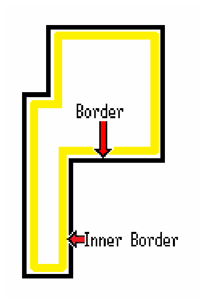


Figure 17: Inner borders of a image object v

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

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

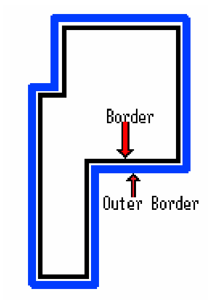


Figure 18: Outer borders of a image object v

Bounding Box

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** (**x_{min}(v), x_{max}(v)** and **y_{min}(v), y_{max}(v)).**

The bounding box **B_v(d)** can be also extended by a number of pixels.

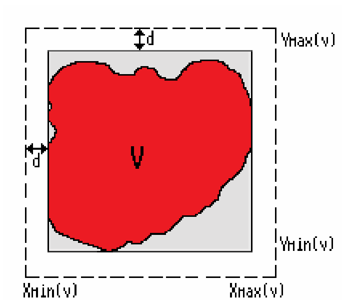


Figure 19: Bounding box of an image object v

Border Length

The border length **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.

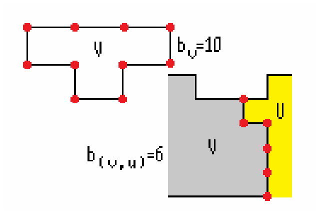


Figure 20: Border length of an image object v or between two objects v, u

2.1.3 Class Related Features

Class-Related Sets

Let **M=(m_1,...,m_a)** be a set of classes with **m** being a specific class and **m**∈**M**. Each object has a fuzzy membership value of **ϕ(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:

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

e.g.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 $\bar{\Delta}_k$**(v,N_v(d,m)).**

2.1.4 Shape Related Features

> Object Features
> Shape

Many of the form features provided by **Definiens Professional** 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 Professional** 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.

Shape Approximations based on Eigenvalues

This approach measures the statistical distribution of the pixel coordinates **(x,y)** of a set **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.

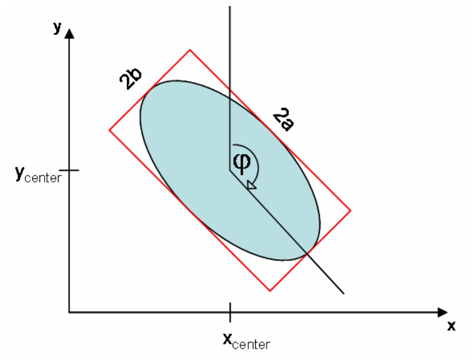


Figure 21: Elliptic approximation

Elliptic Approximation

The elliptic approximation uses the eigenvectors **(λ_1,λ_2)**of the covariance matrix and computes an ellipsis with axis along **e_1** and **e_2** with length.

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

$$\frac{a}{b} = \frac{\lambda_1}{\lambda_2}$$

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

The eigenvector of the main axis defines the main direction.

2.1.5 Distance Related Features

Distance Measurements

Many features allow you to enter a spatial distance parameter. Distances are usually measured in pixel units. Since 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 efficient but it can get 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.

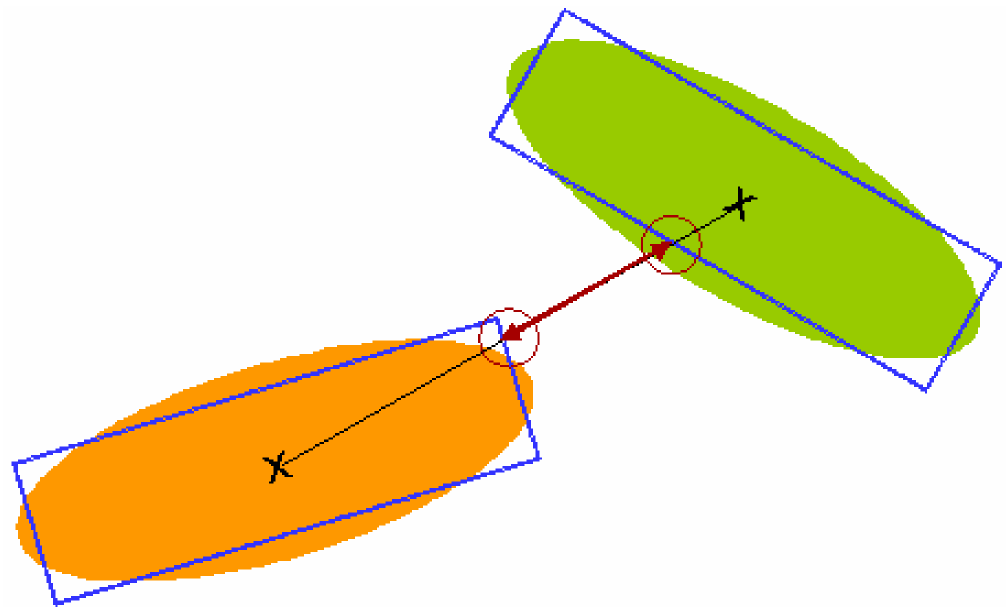


Figure 22: Distance calculation between image objects.
Black line: center of gravity approximation. Red line: Smallest enclosing rectangle approximation.

We recommend to use 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, e.g. by creating border objects. To avoid performance problems it is also important to restrict the total number of objects involved into distance calculations to a small number.

You can edit the distance calculation in the **Options** dialog box and set the **Distance calculation** option to your preferred value.

Note

Changing the distance calculation changes the results rule sets produce. It is strongly recommended not to change distance calculations within a a rule set or mix rule sets developed for different distance calculations.

When you load a rule set that does not fit to your current settings of the distance calculations a warning dialog pops up and the distance calculation of the rule set is activated.

***Definiens eCognition Server** is able to handle rule sets using different calculations for different workspaces.*

2.2 Object Features

> Object Features

2.2.1 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]**

2.2.2 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.

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

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:

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

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 (menu item **Classification > Advanced Settings > Select Image**

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

Layers for Brightness).

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

Parameters:

- **w_k^B**: brightness weight of layer **k**
- **$\bar{c}_k(v)$**: mean intensity of layer **k** of an image object **v**
- **c_k^{min}**: darkest possible intensity value of layer **k**
- **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:

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

Feature value range:

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

Conditions:

- Feature available only for scenes with more than one layer.

Max. diff.

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

Parameters:

- **i, j** : image layers
- **$\bar{c}(v)$** : brightness
- **$\bar{c}_i(v)$** : mean intensity of layer **i**
- **$\bar{c}_j(v)$** : mean intensity of layer **j**
- **c_k^{max}** : brightest possible intensity value of layer **k**
- **K_B** : layers with positive brightness weight
  K_B $:=\{k \in K : w_k=1\}$, w_k:layer weight

Formula:

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

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.

Standard Deviation

[name of a layer]

> 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.

Parameters:

- $\sigma_k(\mathbf{v})$: standard deviation of layer **k** of an image object **v**
- $\mathbf{P_v}$**:** set of pixels of an image object **v**
- $\#\mathbf{P_v}$**:** total number of pixels contained in $\mathbf{P_v}$
- $c_k(x,y)$: 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:

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

Feature value range:

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

Pixel Based

Ratio

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

> Object Features
> Layer Values
> Pixel Based
> **Ratio**

Parameters:

- w_k^B: brightness weight of layer **k**
- $\bar{c}_k(v)$: mean intensity of layer **k** of an image object **v**
- $\bar{c}(v)$: brightness

Formula:

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

If $w_k^B=0$ or $\bar{c}(v)=0$ 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.

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

Parameters:

- **(x,y)**: pixel coordinations
- $c_k(x,y)$: 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:

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

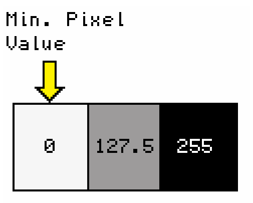


Figure 23: Minimum pixel value of an image object v

Feature value range:

$$[c_k^{min}, c_k^{max}]$$

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 coordinations
- **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:

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

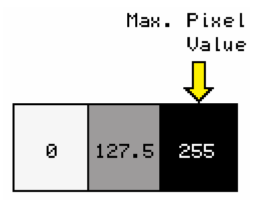


Figure 24: Maximum pixel value of an image object v

Feature value range:

$$[c_k^{min}, c_k^{max}]$$

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.

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

Parameters:

- **P_v** :set of pixels of an image object **v**
- **P_v^{Inner}**: inner border pixels of **P_v**
  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**
- **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:

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

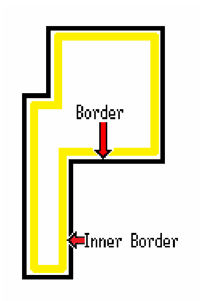


Figure 25: Inner borders of a image object v

Feature value range:

$$[c_k^{min}, c_k^{max}]$$

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.

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

Parameters:

- **P_v** :set of pixels of an image object **v**
- **P_v^{Outer}** : outer border pixels of **P_v**
  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**
- **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:

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

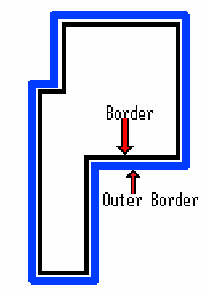


Figure 26: Outer borders of a image object v

Feature value range:

$$[c_k^{min}, c_k^{max}]$$

Contrast to neighbor pixels

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

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

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:

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

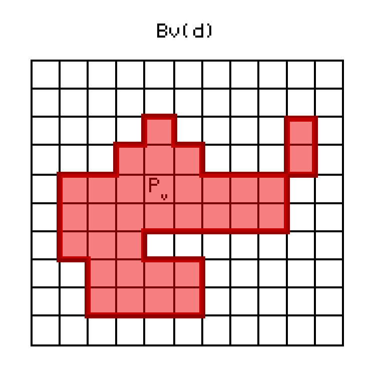


Figure 27: Contrast to neighbor pixels

Feature value range:

[-1000, 1000]

Conditions:

- If **$d=0$**, then **$B_v(d)=B_v$**, and if **$B_v=P_v$** $\therefore$ the formula is invalid.
- If unsigned data exist then maybe **$\bar{c}_k(P_v) = -1$** $\therefore$ 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:

- **P_v** :set of pixels of an image object **v**
- **$B_v(d)$:** extended bounding box of an image object **v** with distance **d**

Formula:

$$\boxed{\sigma_k(B_v(d) - P_v)}$$

Feature value range:

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

Conditions:

- If **$d=0$**, then **$B_v(d)=B_v$**, and if **$B_v=P_v$** $\therefore$ the formula is invalid.

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

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

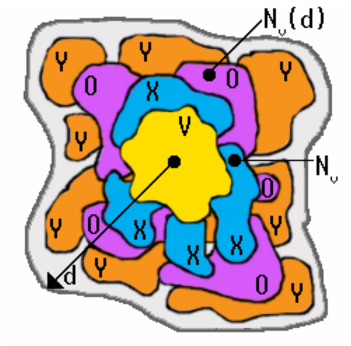


Figure 28: Direct and distance neighbors

Feature value range:

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

Conditions:

- 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:

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

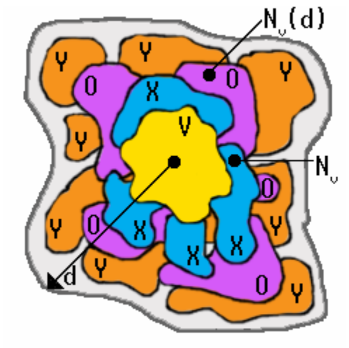


Figure 29: Direct and distance neighbors

Feature value range:

$[\ 0, c_k^{range}\ ]$

Conditions:

- If **w=0** $\Rightarrow$ the mean difference to neighbors is **0** therefore the formula is invalid.

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.

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

Parameters:

- **v,u** : image objects
- **b(v,u)**: top 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
- **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**
  $N_v(d):=\{u\in V_i: d(v,u)\leq d\}$
- $N_v^D(d)$: darker neighbors to **v** at a distance **d**
  $N_v^D(d):=\{u\in N_v(d): \bar{c}_k(u)< \bar{c}_k(v)\}$

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

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

Formula:

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

Feature value range:

$$\left[\,-c_k^{range}, c_k^{range}\,\right]$$

Conditions:

- If **w=0** then $\overline{\Delta}_k^D$**(v) =0**∴ the formula is invalid.
- If N_v^D**(d)=∅** ∴ the formula is invalid.

Mean diff. to brighter neighbors

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

Parameters:

- **v,u** : image objects
- **b(v,u)**: top 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
- **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**
  $N_v(d):=\{u\in V_i: d(v,u)\leq d\}$
- N_v^B**(d)**: brighter neighbors to **v** at a distance **d**
  $N_v^B(d):=\{u\in N_v(d): \bar{c}_k(u)> \bar{c}_k(v)\}$

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

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

Formula:

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

Feature value range:

$$\left[-c_k^{range}, c_k^{range} \right]$$

Conditions:

- If **w=0** then $\overline{\Delta}_k^B(v) = 0 \therefore$ the formula is invalid.
- If $N_v^B(d)=\varnothing \therefore$ 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:

- $N_v^B(d)$: brighter neighbors to **v** at a distance **d**
  $N_v^B(d):=\{u \in N_v(d): \ \overline{c}_k(u)>c_k(v)\}$
- b_v : image object border length
- **b(v,u)**: top relation border length
- **d**: distance between neighbors

Formula:

$$\boxed{\sum_{u \in N_v^B(d)} \frac{b(v,u)}{b_v}}$$

Feature value range:

[0,1]

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**
- c_k^{range} : data range of **k**
  $c_k^{range} := c_k^{max} : - c_k^{min}$
- $S_v(d)$: 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:

$$\boxed{\bar{c}_k(v) - \bar{c}_k(U_v(d))}$$

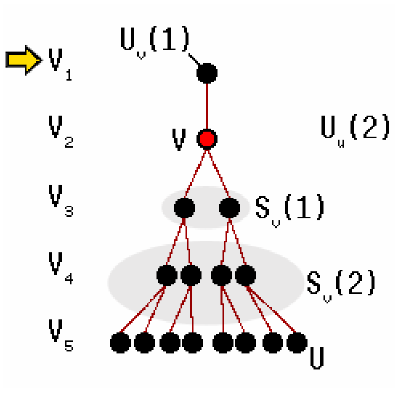


Figure 30: Image object Hierarchy

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

Formula:

$$\boxed{\frac{\bar{c}_k(v)}{\bar{c}_k(U_v(d))}}$$

Feature value range:

[0, ∞]

Conditions:

- If **U_v(d)=∅** ∴ the formula is undefined.
- If **U_v(d)=0** →∞ ∴ 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**
- $\sigma_k\mathbf{(v)}$: std. deviation of object **v** on layer **k**
- $\mathbf{c}_k^{\text{range}}$: data range of layer **k**
  $\mathbf{c}_k^{\text{range}} := \mathbf{c}_k^{\max} :- \mathbf{c}_k^{\min}$

Formula:

$$\boxed{\sigma_k(v) - \sigma_k(U_v(d))}$$

Feature value range:

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

Conditions:

- If $\mathbf{U_v(d)=\varnothing}$ $\therefore$ 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**
- $\sigma_k\mathbf{(v)}$: std. deviation of object **v** on layer **k**

Formula:

$$\boxed{\frac{\sigma_k(v)}{\sigma_k(U_v(d))}}$$

Feature value range:

$[0,\infty]$

Conditions:

- If $\mathbf{U_v(d)=\varnothing}$ $\therefore$ the formula is undefined.
- If $\sigma_k\mathbf{(U_v(d))=0}$ $\Rightarrow$ the std. deviation ratio to $\mathbf{U_v(d)} = \mathbf{1}$.

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:

- $\bar{c}_k$: mean intensity of layer **k**
- $\bar{c}_k(v)$: mean intensity of layer **k** of an image object **v**
- c_k^{range} : data range of layer **k**
  $c_k^{range} := c_k^{max} :- c_k^{min}$

Formula:

$$\boxed{\bar{c}_k(v) - \bar{c}_k}$$

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:

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

Formula:

$$\boxed{\frac{\bar{c}_k(v)}{\bar{c}_k}}$$

Feature value range:

$[-\infty, \infty]$

Conditions:

- If $\bar{c}_k = 0$ ⇔the feature is undefined as the image object is black.

2.2.3 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]

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 **P**$_v$
- γ_v : length/width ratio of an image object **v**

Formula:

$$\boxed{\sqrt{\#P_v \cdot \gamma_v}}$$

Feature value range:

[0; ∞]

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 **P**$_v$
- γ_v : length/width ratio of an image object **v**

Formula:

$$\boxed{\frac{\#P_v}{\gamma_v}}$$

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.

$$\boxed{\gamma_v^{EV} = \frac{\lambda_1(v)}{\lambda_2(v)}}$$

- The ratio length/width can also be approximated using the bounding box.

$$\boxed{\gamma_v^{BB} = \frac{\left(k_v^{bb'}\right)^2}{\#P_v}}$$

Definiens Professional uses both methods for the calculation and takes the smaller of both results as the feature value.

Parameters:

- **#P**v :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:

$$\boxed{\gamma_v := \min \gamma_v^{EV}, \min \gamma_v^{BB}}$$

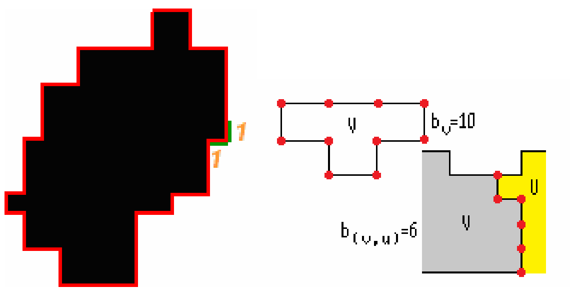
Feature value range:

[0; ∞]

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**



Feature value range:

[0, ∞]

Asymmetry

The lengthier 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 minor and major axes of this ellipse. The feature value increases with the asymmetry.

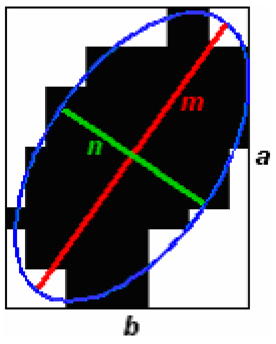
> Object Features
> Shape
> Generic
> **Asymmetry**

Parameters:

- **VarX :** variance of X
- **VarY:** variance of Y

Formula:

$$\boxed{\frac{2\sqrt{\frac{1}{4}(VarX + VarY)^2 + (VarXY)^2 - VarX \cdot VarY}}{VarX + VarY}}$$



Feature value range:

[0; ∞]

> ***Note***
>
> *The length/width ratio is more accurate.*

Main direction

> Object Features
> Shape
> Generic
> **Main direction**

In **Definiens Professional**, 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.

Parameters:

- **VarX :** variance of X
- **VarY:** variance of Y
- λ_1 : eigenvalue

Formula:

$$\boxed{\frac{180^\circ}{\pi}\tan^{-1}(VarXY, \lambda_1 - VarY) + 90^\circ}$$

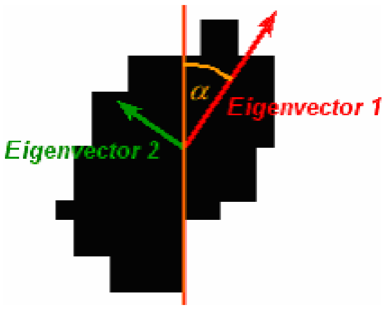


Figure 31: Ellipse approximation using eigenvalues

Feature value range:

[0; 180]

Density

The density an be expressed by the area covered by the image object divided by its radius. **Definiens Professional** 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:

- $\sqrt{\#P_v}$: diameter of a square object with $\#P_v$ pixels.
- $\sqrt{VarX+VarY}$: diameter of the ellipse

Formula:

$$\frac{\sqrt{\#P_v}}{1+\sqrt{VarX+VarY}}$$

Feature value range:

[0, depended on shape of image object]

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:

- b_v: image object border length
- $4\sqrt{\#P_v}$: border of square with area $\#P_v$

Formula:

$$\frac{b_v}{4\sqrt{\#P_v}}$$

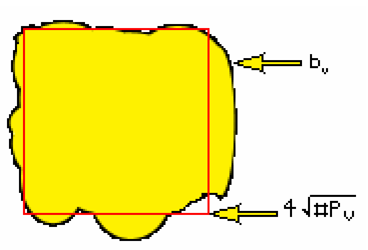


Figure 32: Shape index of an image object v

Feature value range:

[1,∞], **1**=ideal.

The more fractal an image object appears, the higher its shape index.

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.

> Object Features
> Shape
> Generic
> **Border index**

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**

Formula:

$$\boxed{\frac{b_v}{2(l_v + w_v)}}$$

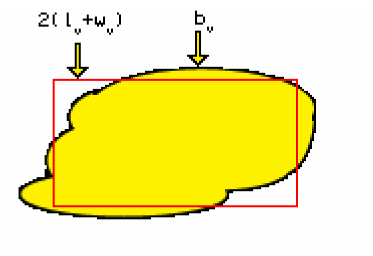


Figure 33: Border index of an image object v

Feature value range:

[1,∞], **1**=ideal.

The more fractal an image object appears, the higher its border index.

Compactness

This feature is similar to the border index, however instead of border based it is area based.

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**$_v$.

> Object Features
> Shape
> Generic
> **Compactness**

Parameters:

- $\mathbf{l}_v$: length of an image object **v**
- $\mathbf{w}_v$: width of an image object **v**
- **#P**$_v$**:** total number of pixels contained in **P**$_v$

Formula:

$$\boxed{\frac{l_v * w_v}{\# P_v}}$$

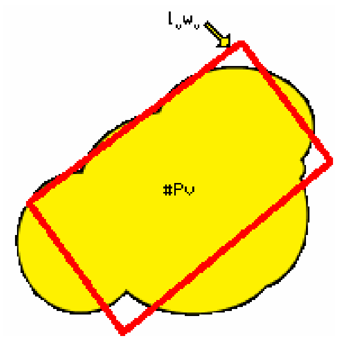


Figure 34: Compactness of an image object v

Feature value range:

[0, 1], **1**=ideal.

The more compactness an image object appears, the smaller its border.

Roundness

> Object Features
> Shape
> Generic
> **Roundness**

Formerly called: Diff. 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

Formula:

$$\boxed{\varepsilon_v^{max} - \varepsilon_v^{min}}$$

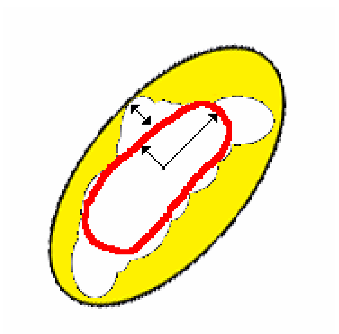


Figure 35: Roundness of an image object v

Feature value range:

[0,∞], **0**=ideal.

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)**
- **P_v:** set of pixels of an image object **v**
- **#P_v:** total number of pixels contained in **P_v**

Formula:

$$\phi := 2 \cdot \frac{\#\{(x,y) \in P_v : \varepsilon_v(x,y) \leq 1\}}{\# P_v} - 1$$

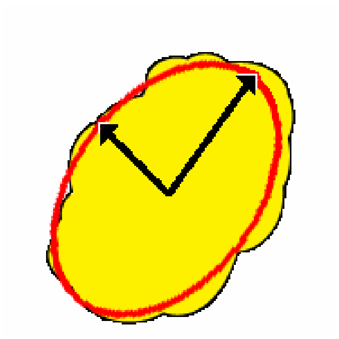


Figure 36: Elliptic fit of an image object v

Feature value range:

[0,1], **1**=complete fitting, whereas **0** = only 50% or less pixels fit inside the ellipse.

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:

- ρ_v**(x,y)** : rectangular distance at a pixel **(x,y)**

Formula:

$$\frac{\#\{(x,y) \in P_v : \rho_v(x,y) \leq 1\}}{\# P_v}$$

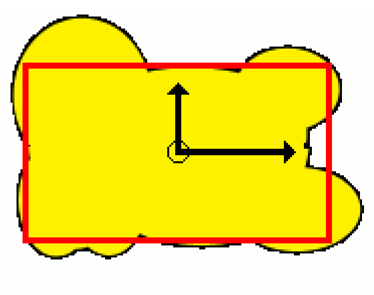


Figure 37: Rectangular fit of an image object v

Feature value range:

[0,1], **1**=complete fitting, whereas **0** = 0% fit inside the rectangular approximation.

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:

- ε_v**(x,y)** : elliptic distance at a pixel **(x,y)**

Formula:

$$\boxed{\varepsilon_v(x_o,y_o)}$$

with $(x_o,y_o) = \min \varepsilon_v(x,y), (x,y) \notin P_v$

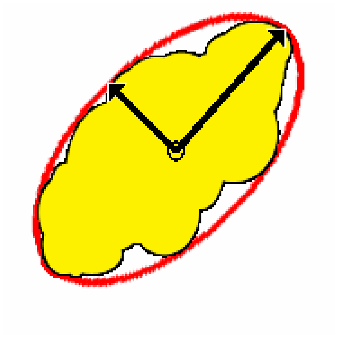


Figure 38: Radius of smallest enclosing ellipse of an image object v

Feature value range:

[0,∞]

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:

- ε_v**(x,y)** : elliptic distance at a pixel **(x,y)**

Formula:

$\varepsilon_v(x_o, y_o)$

with $(x_o, y_o) = \max \varepsilon_v(x,y), (x,y) \in P_v$

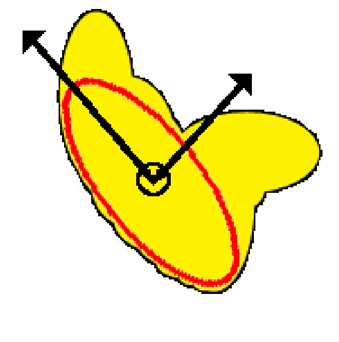


Figure 39: Radius of largest enclosed ellipse of an image object v

Feature value range:

$[0,\infty]$

Position

The following 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.

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}

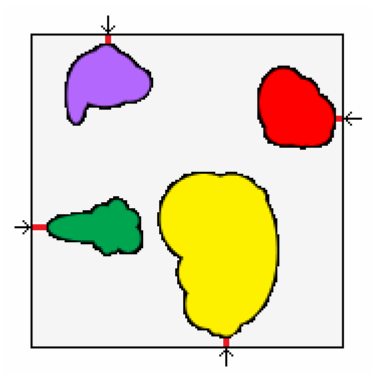


Figure 40: Distance between the nearest border and the image object

Feature value range:

[0,max{sx-1, sy-1}]

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:

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

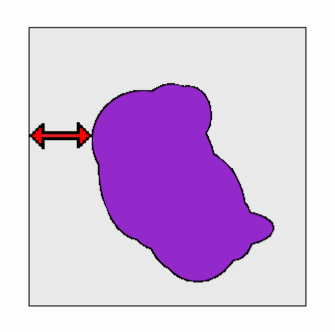


Figure 41: X _distance between the image object and the left border

Feature value range:

[0,sx-1]

X distance to image right border

Horizontal distance to the right border of the image.

> Object Features
> Shape
> Position
> **X distance to image right border**

Parameters:

- **sx** : scene size at the right border.
- **maxx** : maximum distance from the image border at **x**-axis

Formula:

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

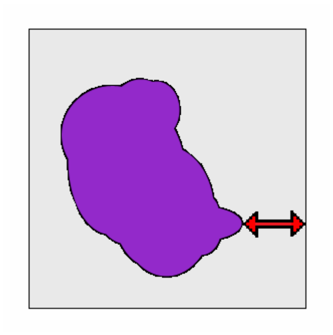


Figure 42: X distance to the image object right border

Feature value range:

[0,sx-1]

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:

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

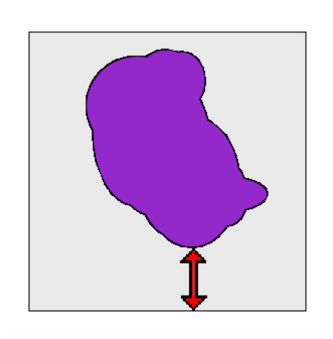


Figure 43: Y _distance between the image object and the bottom border

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:

$$sy - \max_{(x,y)\in P_v} y$$

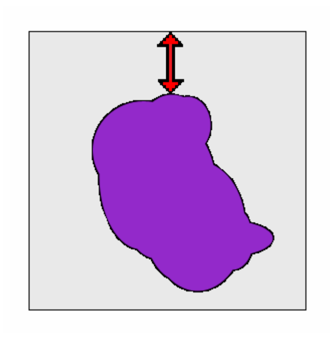


Figure 44: Y _distance between the image object and the top border

Feature value range:

[0,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:

- $\bar{x}_v$: **x** center of an image object **v**

- **#P**$_v$: total number of pixels contained in **P**$_v$

Formula:

$$\bar{x}_v := \frac{1}{\#P_v} \cdot \sum_{(x,y)\in P_v} x$$

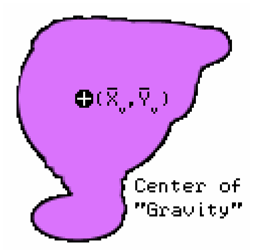


Figure 45: Center of gravity of an image object v

Feature value range:

[0, x_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:

$$\bar{y}_v := \frac{1}{\#P_v} \cdot \sum_{(x,y)\in P_v} y$$

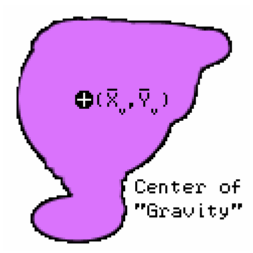


Figure 46: Center of gravity of an image object v

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:

$$\boxed{\min_{(x,y)} x}$$

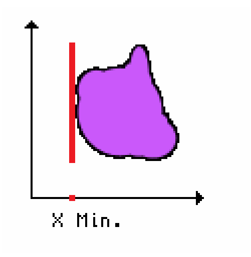


Figure 47: Minimum value of X_Cord at the image object border

Feature value range:

[0, y_coordinate. of the center of gravity]

X max.

Maximum X-position of the image object (derived from bounding box).

> Object Features
> Shape
> Position
> **X max**

Formula:

$$\boxed{\max_{(x,y)} x}$$

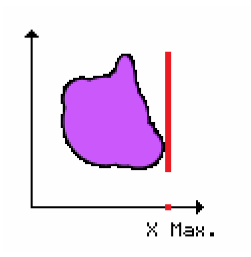


Figure 48: Maximum value of X_Cord at the image object border

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:

$$\boxed{\min_{(x,y)} y}$$

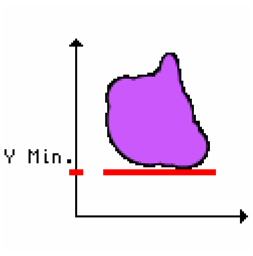


Figure 49: Minimum value of Y_Cord at the image object border

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:

$$\boxed{\max_{(x,y)} y}$$

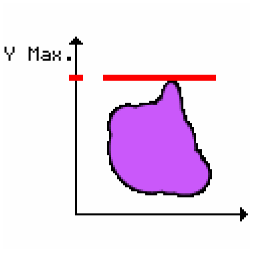


Figure 50: Maximum value of Y_Cord at the image object border

Feature value range:

[0, y_coordinate. of the center of gravity]

To Superobject

Use the following 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.

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:

$$\frac{\# P_v}{\# P_{U_v(d)}}$$

Conditions:

- If $U_v(d) = \varnothing$ ∴ the formula is undefined.

Feature value range:

[0,1]

Rel. rad. position to superobject (n)

> Object Features
> Shape
> To Superobject
> **Rel. rad. position to superobject**

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.

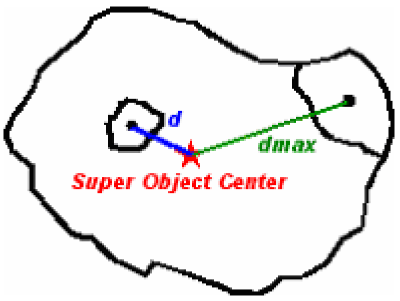
Use this feature to describe an image object by its position relative to the center of its superobject.

Parameters:

- $\#P_v$: total number of pixels contained in **P**v
- $\#P_{Uv(d)}$: the size of the superobject of an image object **v**
- $d_g(v,U_v(d))$: distance of **v** to the center of gravity of the superobject $U_v(d)$

Formula:

$$\frac{d_g(v,U_v(d))}{\max\limits_{u \in S_{Uv(d)}(d)} d_g(u,U_v(d))}$$



Feature value range:

[0; 1]

Conditions:

- If $U_v(d) = \varnothing$ ∴ 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.

Parameters:

- **$N_U(v)$** :Neighbors of **v** that exist within the superobject
  $N_U(v)$ $:=\{u \in N_v : U_u(d) - U_v(d)\}$
- **b_v** : image object border length

Formula:

$$\boxed{\frac{\sum_{u \in N_U(v)} b(v,m)}{b_v}}$$

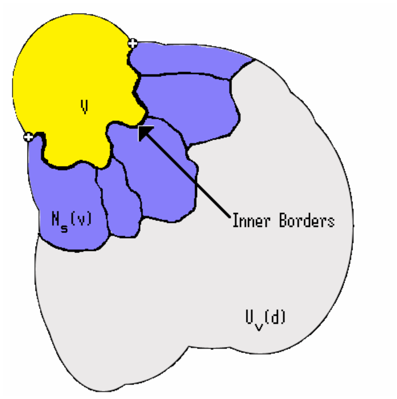
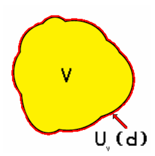


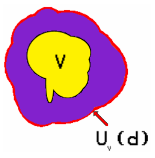
Figure 51: Relative inner border of an image object v to super object U

Conditions:

- If the feature range is **0** $\Rightarrow$ **$v=U_v(d)$**



- If the feature range is **1** $\Rightarrow$ **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.

> Object Features
> Shape
> To Superobject
> **Distance to superobject center**

- $\mathbf{d_g(v,U_v(d))}$: distance of **v** to the center of gravity of the superobject $\mathbf{U_v(d)}$

Formula:

$\mathbf{d_g(v,U_v(d))}$

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**

Formula:

$\mathbf{d_e(v,U_v(d))}$

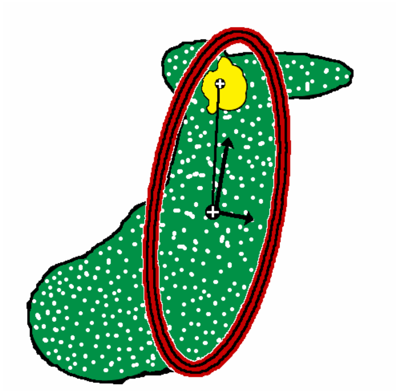


Figure 52: Distance between the distance from the superobject's center to the center of a subobject.

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:

> Object Features
> Shape
> To Superobject
> **Is end of superobject**

a is the image object with the maximum distance to the superobject,

b is the image object with the maximum distance to a.

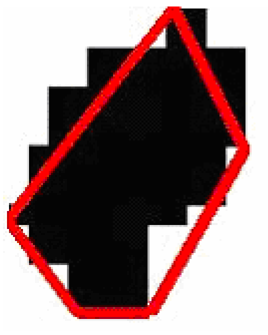
Is center of superobject

This feature is true, if the image object is the center of its superobject.

> Object Features
> Shape
> To Superobject
> **Is center of superobject**

Based on Polygons

The polygon features provided by **Definiens Professional** 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.

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)

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.

> Object Features
> Shape
> Based on Polygons
> **Area (including inner polygons)**

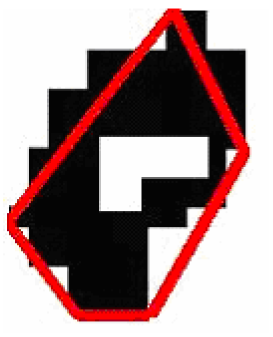


Figure 53: The area of an image object v including an inner polygon.

The above picture shows a polygon with one inner object

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)**

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]

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)**

Stddev of length of edges

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:

- X_i : length of edge i
- $\overline{X}$: mean value of all lengths
- **n** : total number of edges

Formula:

$$stddev = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n}}$$

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:

- X_i : length of edge **i**
- **n** : total number of edges

Formula:

$$Average = \frac{\sum_{i=1}^{n} X_i}{n}$$

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 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)**

Edges longer than

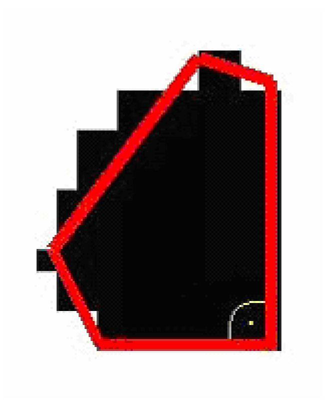
This feature reports the number of edges that have lengths exceeding a threshold value. The user defines the threshold value.

> 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**



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.

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/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]

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)**

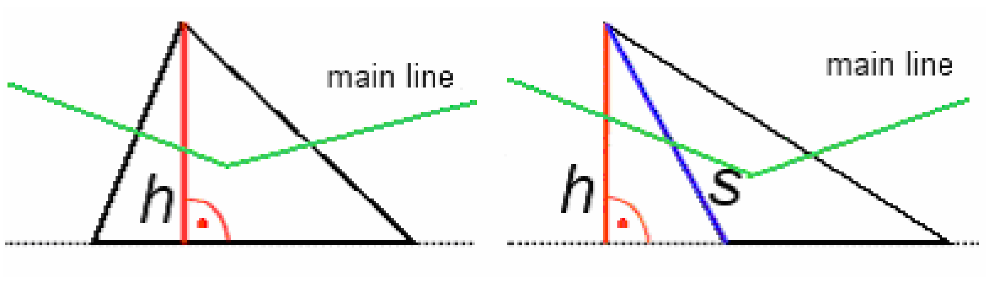
Feature value range:

[0; depending on shape of objects]

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)**



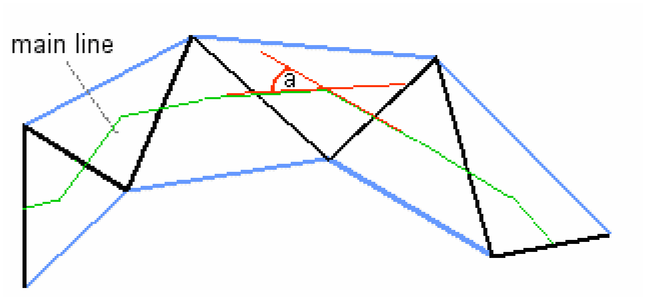
Feature value range:

[0; depending on shape of objects]

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]

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]

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]

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]

Stddev. of area represented by segments

Calculates the standard deviation of all triangles created by the Delaunay Triangulation (see fig. 11).

> Object Features
> Shape
> Based on Skeletons
> **Stddev. of area represented by segments**

Feature value range:

[0; depending on shape of the 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]

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]

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]

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]

2.2.4 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. The subobject level to use can be defined by editing the feature distance. The texture features are divided in two groups: texture concerning the spectral information of the subobjects and texture concerning the form of the subobjects.

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. 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.

> Object Features
>▨ Texture
> Layer value texture based on sub-objects
> **Mean of sub-objects: stddev.**

Parameters:

- **$S_v(d)$** : subobject of an image object **v** at distance **d**
- **$\bar{c}_k(u)$** : 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 **$S_v(d)$** = ∅ ∴ 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. 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.

> Object Features
>▨ Texture
> Layer value texture based on sub-objects
> **Avrg. mean diff. to neighbors of subobjects**

Parameters:

- **$S_v(d)$** : subobject of an image object **v** at distance **d**
- **$\bar{\Delta}_k(u)$** : mean difference to neighbor of layer **k** of an image object **u.**
- **d** : level distance

Formula:

$$\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)}\bar{\Delta}_k(u)$$

Feature value range:

[0; depending on bit depth of data]

Conditions:

- If $\mathbf{S_v(d)} = \varnothing \therefore$ the formula is invalid

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:

$$\boxed{\frac{1}{\#S_v(d)} \sum_{u \in S_v(d)} \#Pu}$$

Feature value range:

[0; scene size]

Conditions:

- If $\mathbf{Sv(d)} = \varnothing \therefore$ 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**
- $\#\mathbf{P_u}$: 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]

Conditions:

- 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 62

Feature value range:

[0; depending on image object shape]

Conditions:

- 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]

Conditions:

- 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 60

Feature value range:

[0; depending on image object shape]

Conditions:

- 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:

$$\boxed{\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)} a(u)}$$

For more details on asymmetry see the **Asymmetry** section under shape features.

→ Asymmetry on page 60

Feature value range:

[0; depending on image object shape]

Conditions:

- 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.

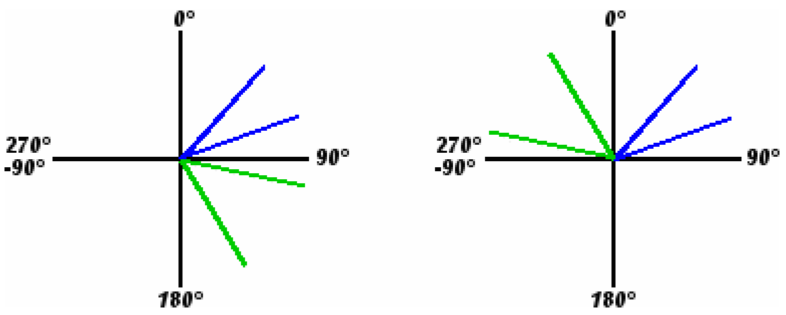
> Object Features
> ▨ Texture
> Shape texture based on sub-objects
> **Direction of sub-objects: mean**

Parameters:

- **S_v(d)** : subobject of an image object **v** at distance **d**
- **a(u)** : main direction of **u**
- **d** : level distance

Formula:

$$\boxed{\frac{1}{\#S_v(d)}\sum_{u\in S_v(d)} a(u)}$$



For more details on main direction see the **Main Direction** section under shape features.

→ Main direction on page 61

Feature value range:

[0-180°]

Conditions:

- If **Sv(d)** = $\varnothing$ $\therefore$ 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.**

Texture After Haralik

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 Haralik 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 Haralik in **Definiens** software are:

> Object Features
> ▨ Texture
> **Texture after Haralik**

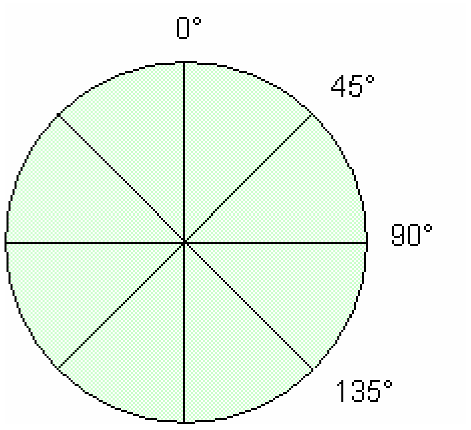
Parameters:

- **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

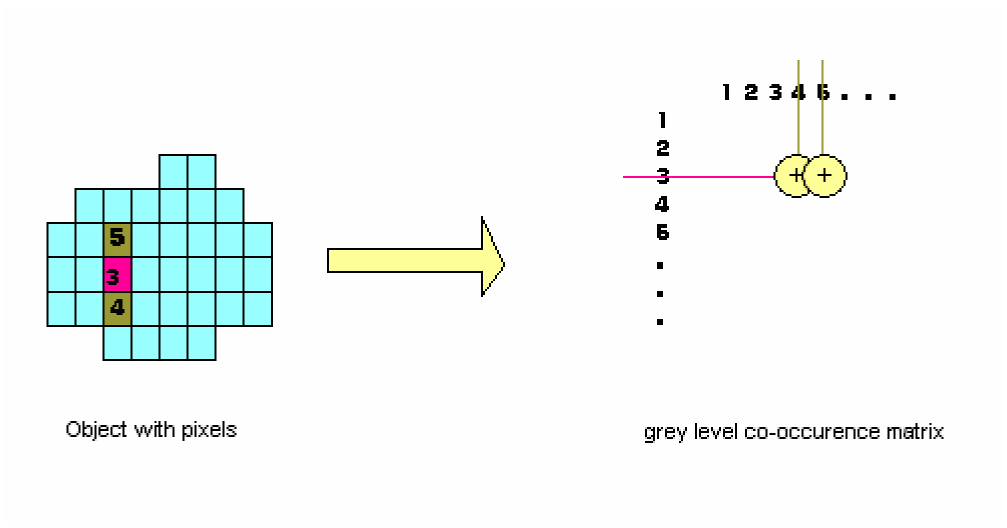
Formula:

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 Haralik feature you have the choice of either one of the above directions or of all directions.

The calculation of Texture after Haralik 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 Haralik 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 Haralik feature is very CPU demanding because auf the calculation of the GLCM.

References for Texture after Haralik:

- Haralik features were implemented in **Definiens** software according to the following references:

- R. M. Haralik, 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. Haralik, 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

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.

> Object Features
> ▨ Texture
> Texture after Haralik
> **GLCM homogeneity**

Parameters:

- **i** : the row number
- **j** : the column number
- **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 Haralik
> **GLCM Contrast**

Parameters:

- **i** : the row number
- **j** : the column number
- **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 Haralik
> **GLCM dissimilarity**

Parameters:

- **i** : the row number
- **j** : the column number
- **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 Haralik
> **GLCM entropy**

Parameters:

- **i** : the row number
- **j** : the column number
- **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 Haralik
> **GLCM ang. 2nd moment**

Parameters:

- **i** : the row number
- **j** : the column number
- **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 Haralik
> **GLCM Mean**

Parameters:

- **i** : the row number
- **j** : the column number
- **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.

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.

> Object Features
> Texture
> Texture after Haralik
> **GLCM StdDev.**

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
- **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 Haralik
> **GLCM Correlation**

Measures the linear dependency of gray levels of neighboring pixels.

Parameters:

- **i** : the row number
- **j** : the column number
- **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. devation

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

High if some elements are large and the remaining ones are small. Similar to GLCM Angular Second Moment: it measures the local homogeneity.

> Object Features
> ▨ Texture
> Texture after Haralik
> **GLDV Ang. 2nd moment**

Parameters:

- **N** : the number of rows or columns
- **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.

> Object Features
> ▨ Texture
> Texture after Haralik
> **GLDV Entropy**

Parameters:

- **i** : the row number
- **j** : the column number
- **P**$_{i,j}$: the normalized value in the cell **i,j**
- **N** : the number of rows or columns
- **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

> Object Features
> Texture
> Texture after Haralik
> **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

> Object Features
> Texture
> Texture after Haralik
> **GLDV Contrast**

It is mathematically equivalent to the GLCM Contrast 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
- k :

Formula:

$$\sum_{k=0}^{N-1} V_k k^2$$

Feature value range:

[0; 90]

2.2.5 Hierarchy

All the following features refer to the embedding of an image object in the entire image object hierarchy.

Level

> 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.

Parameters:

- $\mathbf{U_v(d)}$: superobjects of an image object **v** at distance **d**

Formula:

$$\boxed{\min_d U_v(d) = 0}$$

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 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:

- $\mathbf{N_v(d)}$: neighbors of an image object **v** at a distance **d**

Formula:

$$\boxed{\#N_v(d)}$$

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:

- $\mathbf{S_v(d)}$: subobjects of an image object **v** at a distance **d**

Formula:

$$\boxed{\#S_v(d)}$$

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:

$$\boxed{(\min_d U_v(d)=\emptyset)-1}$$

Feature value range:

[1; number of image object levels -1]

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:

$$\boxed{(\min_d S_v(d)=\emptyset)-1}$$

Feature value range:

[1; number of image object levels -1]

2.2.6 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.

Thematic object ID

The identification number (ID) of a thematic object. Available only for image objects that overlap with one or no thematic object.

> Object Features
> Thematic Attributes
> **Thematic object ID**

[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.

> Object Features
> Thematic Attributes
> **[name of the thematic objects attribute]**

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.

> Object Features
> Thematic Attributes
> **Number of overlapping thematic objects**

Feature value range:

[0; number of thematic objects]

2.3 Class-Related Features

> Class-Related Features

2.3.1 Customized

[name of a customized feature]

If existing, customized features referring to other classes are displayed here.

> Class-Related Features
> Customized
> **[name of a customized feature]**

2.3.2 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:

$\mathbf{0}$ if $N_v(d,m) = \varnothing$

$\mathbf{1}$ 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

Formula:

$\#N_v(d,m)$

Feature value range:

$[0, \infty]$

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**

Formula:

$$\boxed{\sum_{u \in N_v(d,m)} b(v,u)}$$

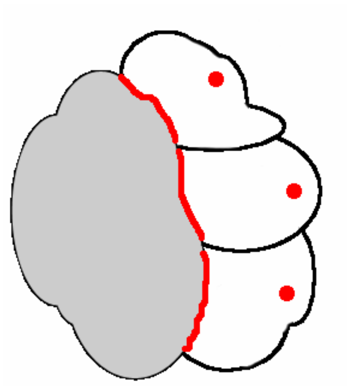


Figure 54: The absolute border between unclassified and classified image objects.

Feature value range:

$[0, \infty]$

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.

> Class-Related Features
> Relations to neighbor objects
> **Rel. border to**

Parameters:

- **b(v,u)** : topological relation border length
- **N**$_v$**(d)** : neighbors to an image object **v** at a distance **d**

- **b**$_v$: image object border length

Formula:

$$\boxed{\frac{\sum_{u \in N_v(d,m)} b(v,u)}{b_v}}$$

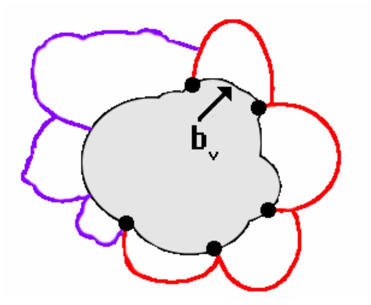


Figure 55: Relative border between neighbors.

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.

Parameters:

- **N**$_v$**(d)** : neighbors to an image object **v** at a distance **d**
- **#P**$_u$: total number of pixels contained in P$_u$

Formula:

$$\boxed{\frac{\sum_{u \in N_v(d,m)} \#P_u}{\sum_{u \in N_v(d)} \#P_u}}$$

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**
- **V_i(m)** : image object level of a class **m**
- **b_v** : image object border length

Formula:

$$\boxed{\min_{u \in V_i(m)} d(v,u)}$$

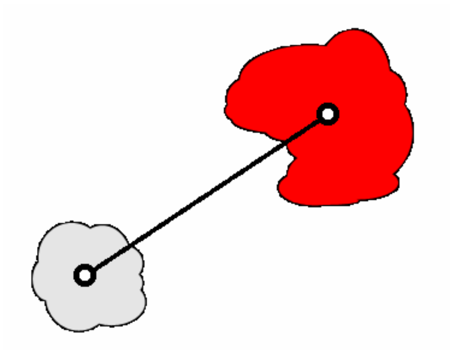


Figure 56: Distance between the center of neighbors.

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
- **N_v(m)** : neighbors to an image object **v** of a class **m**

Formula:

$$\boxed{\bar{\Delta}(v, N_v(m))}$$

Feature value range:

[0, ∞]

2.3.3 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:

$$\boxed{\begin{array}{l} \mathbf{0} \text{ if } S_v(d,m) = \varnothing \\ \mathbf{1} \text{ if } S_v(d,m) \neq \varnothing \end{array}}$$

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

Formula:

$\#S_v(d,m)$

Feature value range:

[0, ∞]

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 (see feature IMAGE).

> Class-Related features
> Relations to sub objects
> **Area of**

Feature value range: [0; scene size]

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**

Feature value range:

[0; 1]

Clark aggregation index

> Class-Related features
> Relations to sub objects
> **Clark Aggregation Index**

2.3.4 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:

$$\boxed{\begin{array}{l} \mathbf{0} \text{ if } U_v(d,m) = \varnothing \\ \mathbf{1} \text{ if } U_v(d,m) \neq \varnothing \end{array}}$$

Feature value range:

[0,1]

2.3.5 Relations to Classification

> Class-Related features
> **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.

> Class-Related Features
> Relations to Classification
> **Membership to**

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**

Formula:

$\boxed{\tilde{\phi}(v,m)}$

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

Formula:

$\boxed{m(v)}$

Feature value range:

[0,1]

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 og an image object **v** to a class **m**

Formula:

$\boxed{\varphi(v,m)}$

Feature value range:

[0,1]

2.4 Scene Features

> Scene Features

2.4.1 Class-related

> Scene features
> **Class-Related**

Number of classified objects

> Scene features
> Class-Related
> **Number of classified objects**

The absolute number of all image objects of the selected class on all image object levels.

Parameters:

- V**(m)** : all image objects of a class **m**
- **m** : a class containing image objects

Formula:

$\boxed{\#V(m)}$

Feature value range:

[0,number of image objects]

Area of classified objects

> Scene features
> Class-Related
> **Area of classified objects**

The absolute area of all image objects of the selected class on all image object levels in pixel.

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**$_v$

Formula:

$$\boxed{\sum_{v \in V(m)} \# P_v}$$

Feature value range:

[0,sx*sy]

Layer mean of classified objects

> Scene features
> Class-Related
> **Layer mean of classified objects**

The mean of all image objects of the selected class on the selected image object levels.

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**

Formula:

$$\frac{1}{\#V(m)}\sum_{v\in V(m)}\bar{c}_k(v)$$

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]

2.4.2 Scene-related

> Scene features
> **Scene-Related**

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

Formula:

$\boxed{sx*sy}$

Feature value range:

[0,1]

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**

Formula:

$\boxed{\#V}$

Feature value range:

[0,1]

Number of layers

Number of layers **K** which are imported in the scene.

> Scene features
> Scene-Related
> **Number of objects**

Pixel resolution

The Resolution of this image as given in the metadata of the scene. The resulting number represents the size of a pixel in the images display unit.

The value is 1 if no resolution is set.

> Scene features
> Scene-Related
> **Pixel resolution**

Image size X

Vertical size **x** of the image in the display unit.

> Scene features
> Scene-Related
> **Image size X**

Notation:

$\boxed{sx}$

Image size Y

Horizontal size **y** of the image in the display unit.

> Scene features
> Scene-Related
> **Image size X**

Notation:

$\boxed{sy}$

Mean of scene

Mean value for the selected layer.

> Scene features
> Scene-Related
> **Mean of Scene**

Notation:

$\boxed{\bar{c}_k}$

Stddev.

Standard Deviation for the selected layer.

Notation:

$\boxed{\sigma_k}$

2.6 (not available)

2.6 Customized Features

Customized features allow you to create new features that are adapted to your needs. Two types of customized features can be created:

- **Arithmetic features** are composed of existing features, variables (**Definiens Developer** only), and constants, which are combined via arithmetic operations.
- **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.

2.6.1 Creating Customized Features

The **Manage Customized Features** dialog box allows you to add, edit, copy, delete and replace 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

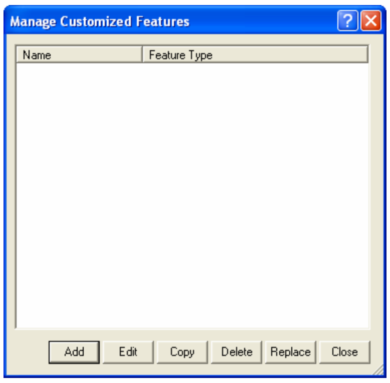


Figure 57: The manage customized features dialog box

Options

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.**
5. Click **Replace** when you want to replace a customized feature with another feature. The procedure is as follows:
    - Select a customized feature and click **Replace**. The **Replace with** dialog box will open.
    - Select the customized feature that should replace the current.

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.*

2.6.2 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.

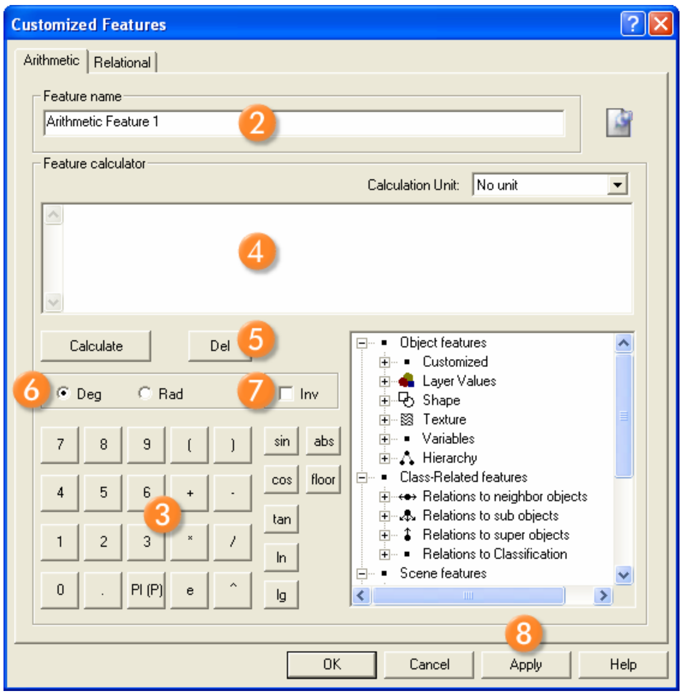


Figure 58: Creating an arithmetic feature at the customized features dialog box.

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.
   - Choose arithmetic operations or mathematical functions.
   - Select features or variables (**Definiens Developer** only) in the feature tree on the right.
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.**

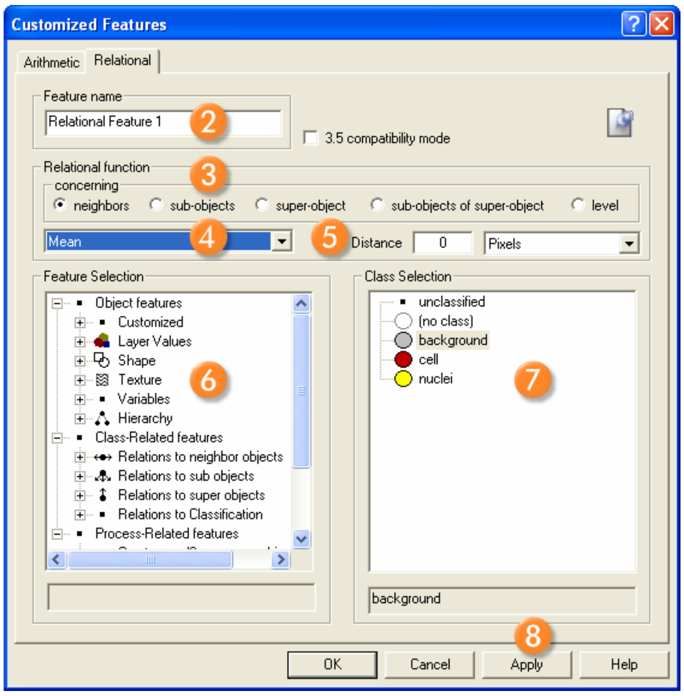
Note

Avoid invalid operations such as division by 0. Invalid operations will result in undefined values.

2.6.3 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 objects.
4. Choose the relational function 4 to be applied.
5. Define the distance 5 of the related objects. The distance can be either horizontal (metric units or pixels) or vertical (levels)
6. Select the feature 6 for which to compute the relation.


Customized Features

Arithmetic | Relational

Feature name: Relational Feature 1

[ ] 3.5 compatibility mode

Relational function concerning: (● neighbors) ○ sub-objects ○ super-object ○ sub-objects of super-object ○ level

Mean

Distance: 0 Pixels

Feature Selection:
- Object features
  - Customized
  - Layer Values
  - Shape
  - Texture
  - Variables
  - Hierarchy
- Class-Related features
  - Relations to neighbor objects
  - Relations to sub objects
  - Relations to super objects
  - Relations to Classification
- Process-Related features

Class Selection:
- unclassified
- no class
- background
- cell
- nuclei

background

OK | Cancel | Apply | Help

Figure 59: 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 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 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	Objects that exist under other objects (superobjects) whose position in the hierarchy is higher. The distance is calculated in levels.
superobject	Contains other objects (subobjects) on lower levels in the hierarchy. The distance is calculated in levels.
sub-objects of superobject	Only the 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 object will be compared to all other 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 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 objects.
Mean difference	Calculates the mean difference between the feature value of an 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 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 object and the feature values of its neighbors of a selected class, which have higher values than the object itself. Note that for averaging the feature values are weighted by the size of respective objects.
Mean difference to lower values	Calculates the mean difference between the feature value of an 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 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 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 object.

2.7 Table of Feature Symbols

This section contains a complete feature symbols reference list.

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
$\sum_{i=1}^{n}$	Sum over index **i**

Images and Scenes

Variables used to represent image objects and scenes.

$k = 1,\ldots, K$	Image layer **k**
$t = 1,\ldots, T$	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)**

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**

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 $\mathbf{P}_v$

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,u)	Topological relation border length

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**

Class Related Sets

Variables representing the relation between classes.

M	Set of classes **$M=\{m1,\ldots, m_a\}$**
m	A class,**$(m \in M)$**
$N_v(d,m)$	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**

3 Index