Trimble®

eCognition Developer

Tutorial - Working with LiDAR (point cloud) files

www.eCognition.com

Introduction

About this Tutorial

In this tutorial, you will learn how to import LiDAR (point cloud) *.las files, convert and interpolate them so that they can be used for segmentation and classification routines.

This Module has four lessons:

- Lesson 1 – Introduction working with LiDAR *.las (point cloud) files in eCognition
- Lesson 2 – Using the 'rasterize point cloud' algorithm
- Lesson 3 – Interpolating raster data (DSM / Number of returns) based on image objects
- Lesson 4 – Segmenting and classifying with interpolated image layers

Further information about eCognition products is available on our website: www.eCognition.com

Requirements

To perform this tutorial, you will need:

- **eCognition Developer** installed on a computer
- A computer **mouse** is highly recommended

All steps of this tutorial, except the exporting of the results can be done using the **eCognition Developer** or the free-trial version (see www.eCognition.com for download).

Data included with the Tutorial

Image data

- '**imagery_Subset.img**' contains the optical aerial RGB and NIR data

LiDAR data

- '**454_266_allpoints-subset01.las**'

Rule Sets

A Rule Sets is available representing the final state of Rule Set development. Whenever the tutorial refers to a Rule Set, it can be found in the tutorial folder.

Project

An eCognition Project is provided for this tutorial and can be found in the tutorial folder.

Lesson 1 – Introduction working with LiDAR *.las (point cloud) files in eCognition

1.1 Lesson contents

In this first lesson we will have a look at the following topics:

- Working with LiDAR (point cloud) files in this tutorial
- Loading LiDAR (point cloud) files into eCognition Developer

eCognition supports the integration of point cloud data in two formats:

- **.las** or
- the compressed **.laz** version

When dealing with LiDAR (point cloud) processing and analysis, there are several components which provide you with means to directly load and analyze LiDAR (point cloud) as well as export raster result images such as DSM / DTM / nDSM.

In this tutorial we will try to create raster datasets derived from the point cloud representing elevation.

- Loading a LiDAR (point cloud) will automatically create a raster representing a quick resampling of the intensity data. It is also possible to display the un-rasterized point cloud (in 2D or 3D).
- Once loaded, we will create additional layers using the 'rasterize point cloud' algorithm.
- Gaps within the LiDAR data will be interpolated based on image objects and pixel information.

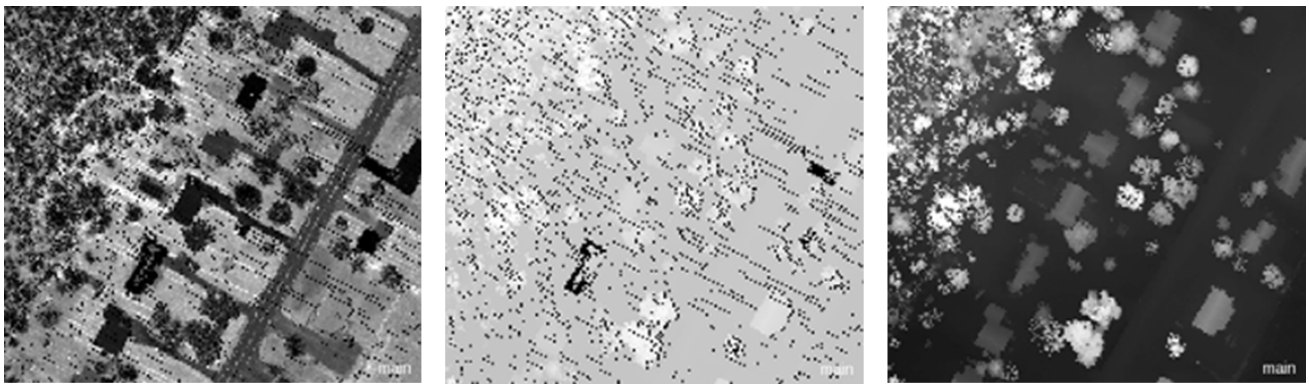

Figure 1: Rasterized LiDAR intensity data (left); first return from elevation after LiDAR conversion (middle); last return from elevation after interpolation (right).

1.2 Working with LiDAR in this tutorial

In this tutorial, you will use elevation information and the information about the number of returns to classify trees and buildings.

This information is not represented in the per default loaded LiDAR (point cloud) image layer. Using an algorithm, two additional image layers are created which contain the desired information. These layers are then used for segmentation and classification.

1.3 Loading LiDAR point cloud files

To allow for quick display of the point cloud, a rasterization is implemented in a simple averaging mode based on intensity values. However you can create rasterized representations of any feature value stored in the point cloud using the '**rasterize point cloud**' algorithm.

If you load a point cloud the resolution of the project is set to 1 per default, which is the optimal value for LiDAR data with a point density of one point per square meter. For coarser resolution data, set the value to 2 or above; for higher resolution data set it to 0.5 or below.

1.3.1 Creating a new Project using LiDAR (point cloud) data

A *.las file is imported, like any other image, via the 'Import Image Layers' dialog box or simply via Drag & Drop functionality.

1. Start eCognition Developer and switch to the Layout '**Rule Set Development Layout**'
2. In the main menu '**File**' choose '**New Project...**' or click on the 'Create New Project' button in the toolbar.
3. Browse to '**454_266_subset02.las**' which is located in the tutorial folder.
4. Select it and hit the '**Open**' button to import it.

The .las file is added to the Project and the Project automatically assumes the name of the imported .las file.

In addition to the *.las file, please load the corresponding image file:

5. Click '**Insert**' within the Image Layer section of the '**Create Project**' dialog
6. Navigate to the tutorial folder and select the '**imagery_Subset.img**' file and confirm with OK.

The image file is added to the Project. At this point it is time to assign image layer aliases and adjust the order of image layers (this is not required but recommended for simplified display options). This also can be done in a later stage but for this tutorial we will do the assigning of the aliases straight away during project creation.

7. Please assign the following aliases:
    - Layer 1 = LiDAR
    - Layer 2 = red
    - Layer 3 = green
    - Layer 4 = blue
    - Layer 5 = nir
8. Finally, assign a meaningful name to the project such as: '**Working with LiDAR**' and confirm with OK.

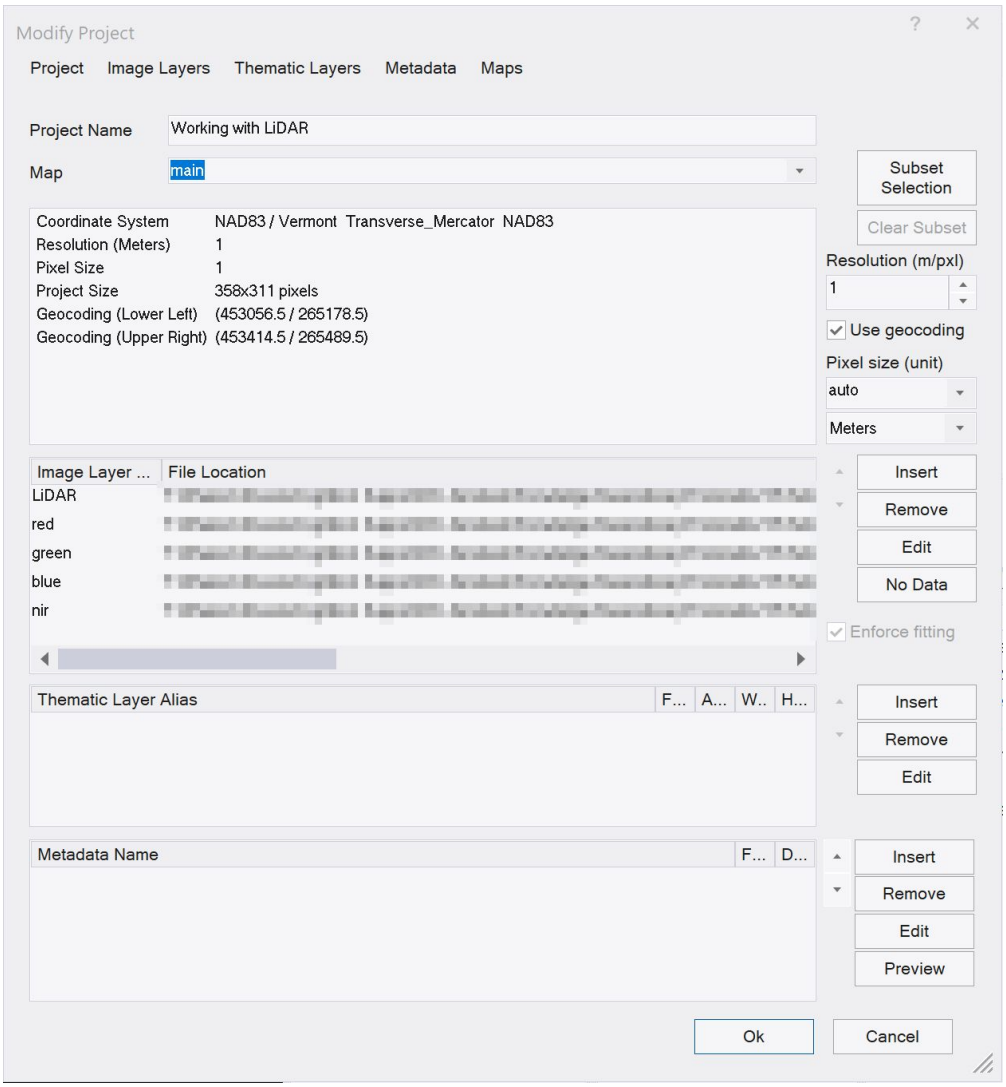

Figure 2: 'Create Project' dialog settings.

Now you should see your data in an eCognition project. Hit the '**Rule Set Development Layout**' button and you will have the View settings on the left hand side of your screen. Here you actually can change the order of the layers, rename the aliases, change band combinations, display or remove layers from the view, remove layers from the project etc.. To adjust the order of the image layers, simply drag & drop the layers to their desired position in the view settings window.. Move the LiDAR layer to the bottom of the list.

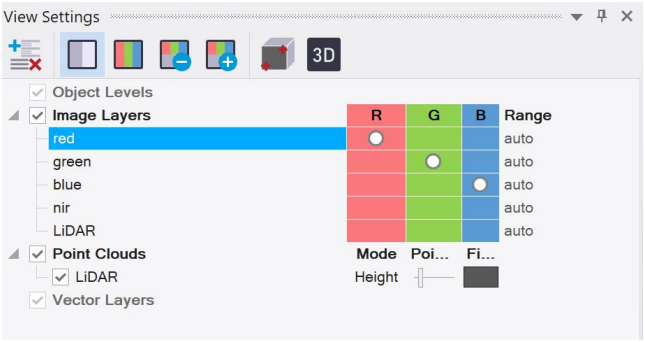

Figure 3: 'The View Settings window.

1.3.2 Review the input data

By default, LiDAR point clouds are loaded and displayed using a quick resampling of the intensity data. Use the '**View Settings**' window to look at just the LiDAR layer in this rasterized format.

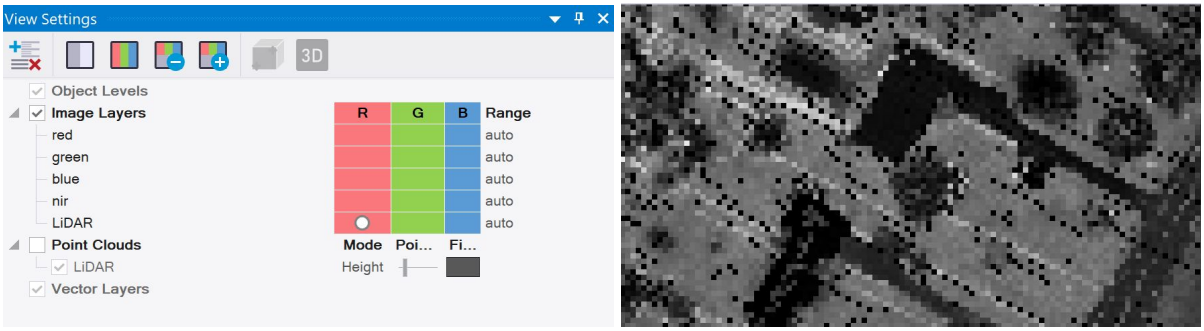

Figure 4: How to display the rasterized LiDAR in the View Settings window (left) and detail of loaded LiDAR data (right).

Alternatively, you can display the point cloud itself by checking the checkbox for '**Point Clouds**' available in the '**View Settings**'. You can display the point cloud in 3D using one of the 3D subset buttons (see figure 5).

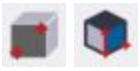

Figure 5: 3D subset buttons. 3D Subset selection (left) and 3 point Subset selection (right)

Using these subset tools will split the view and you now can have a look at your loaded point cloud in 3D.

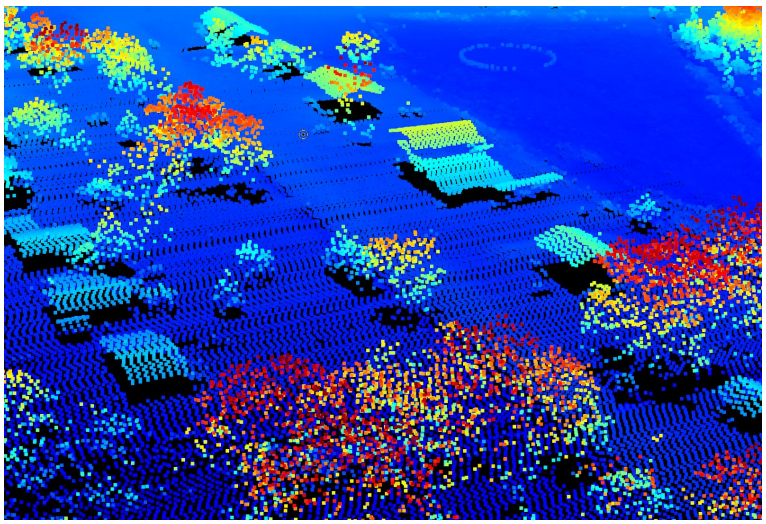

Figure 6: 3D view of point cloud.

To navigate through the point cloud, you can use the various buttons of your computer mouse:

- Use the mouse wheel to zoom in and out while moving the mouse
- Hold the right mouse button to rotate the point cloud view
- Hold both mouse wheel to pan through the scene

Once the 'Point Cloud View' is activated, a number of '**Point Cloud View Settings**' become available in the '**View Settings**' window.

Depending on the characteristics of the point cloud file, various display modes are available:

- Render Mode: Fixed Color | Classification | Intensity | Color Coded Intensity | RGB | Height
- Point Size can be used to adjust the size of the rendered points.

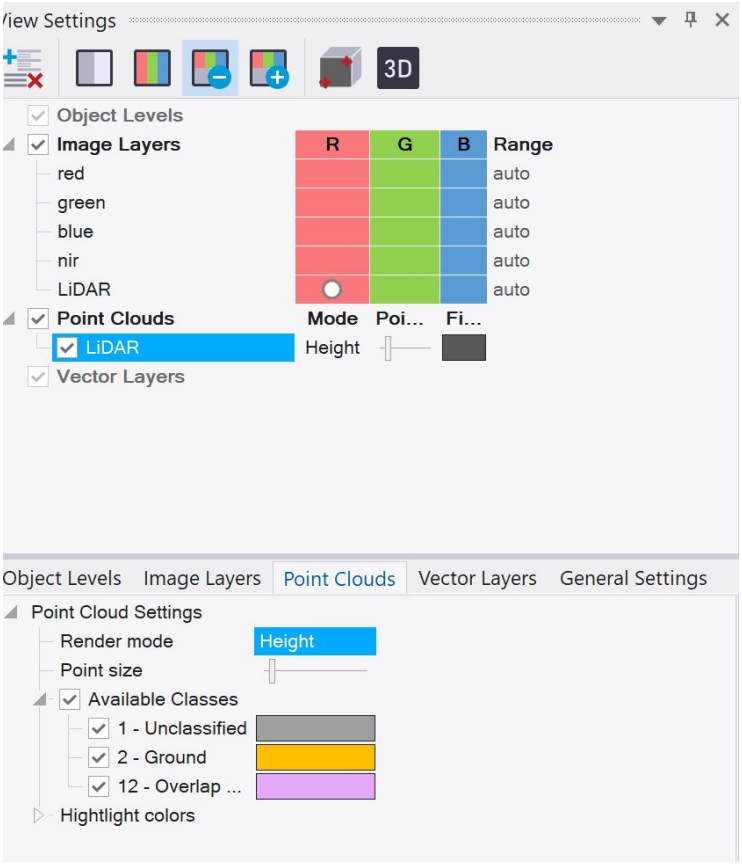

Figure 7: 'Point Cloud View Settings' dialog.

1. Explore the LiDAR file with the display tools in the View Settings window
2. Adjust the view settings to see how they affect the view

If you feel comfortable adjusting different view settings and navigating in 3D, please go ahead with the next Lesson of this Tutorial.

Lesson 2 – Using the 'rasterize point cloud' algorithm

2.1 Lesson content

- Introduction to the '**rasterize point cloud**' algorithm
- Creating an image layer - '**DSM**'
- Creating an image layer - '**Number of Returns**'

In this and the following lessons a Project will be used which contains LiDAR data in combination with optical data.

Figure 8: Optical image layers (left); loaded LiDAR data (right).

From the loaded LiDAR data, additional information can be extracted using the algorithm '**rasterize point cloud**'. This algorithm can create new image layers, containing information about elevation, number of returns and more.

In later exercises, the image layers created in this lesson are used for further segmentation and classification of trees and buildings in the loaded subset. The number of returns is used to classify trees, the elevation is used to classify buildings.

2.2 Introduction to the 'rasterize point cloud' algorithm

This algorithm can create new image layers, containing information about elevation, number of returns and more.

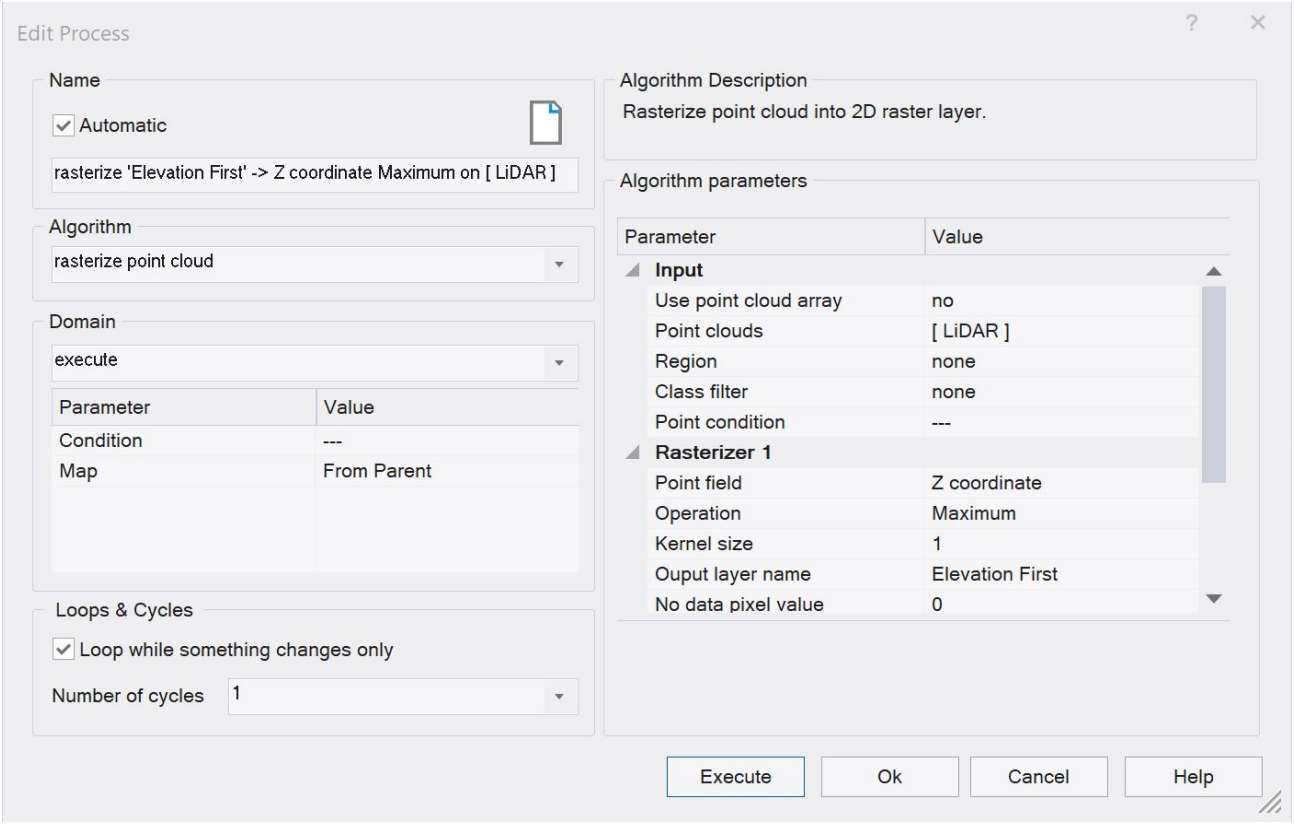

Figure 9: Process settings to convert the LiDAR (point cloud) to a new image layer.

2.2.1 Define image Layer

In the field '**Point CLouds**' in the parameter section on the right of the Edit Process window, you define the point cloud to be used for converting to a raster (rasterization).

2.2.2 Filtering settings

In the fields '**Region**', '**Class filter**', '**Point Condition**' you are allowed to limit the points used for generating the raster representation based on available point properties, location or classification. Leave the default settings if you don't want to filter the point cloud.

- Region - Define a region of interest to filter the point cloud.
- Class filter - You can select a classification filter.
- Point condition - You can select threshold condition(s) to filter points.

2.2.3 Conversion parameters

In the field '**Point filed**' under Rasterizer 1, you define which information to use for generating the new image layer. The following parameters can be set:

- X coordinate: X coordinate value of each point.
- Y coordinate: Y coordinate value of each point.

- Z coordinate: The point elevation values.
- Intensity: The point intensity values.
- Return number: The Return Number is the pulse return number for a given output pulse.
- Number of returns: The Number of Returns is the total number of returns for a given pulse.
- Classification flags: Classification flags are used to indicate special characteristics associated with the point (0 Synthetic, 1 Key-point, 2 Withheld, 3 Overlap).
- Scanner channel: Used to indicate the channel (scanner head) of a multichannel system, channel 0 is used for single scanner systems.
- Scan direction flag: Denotes the direction at which the scanner mirror was traveling at the time of the output pulse (value of 1 is a positive scan direction, value of 0 is a negative scan direction).
- Edge of flight line: The last point on a given scan line before it changes direction. It has a value of 1 only when the point is at the end of a scan.
- Classification: The class information stored in the points.
- User data: This field may be used at the user's discretion.
- Scan angle: Represents the rotational position of the emitted laser pulse with respect to the vertical of the coordinate system of the data
- Point source ID: This value indicates the file from which this point originated.
- GPS time: Point time tag value at which the point was acquired.
- Red: The red channel value associated with this point.
- Green: The green channel value associated with this point.
- Blue: The blue channel value associated with this point.
- NIR: The NIR (near infrared) channel value associated with this point.
- Number of points: number of points that are mapped to the pixel.

Since the data is resampled into a 2D Grid with a fixed spacing during import, several LiDAR points can be within one pixel (or no point within one pixel)..

2.2.4 Calculation parameters

In the field '**Operation**' the calculation mode can be defined, which determines how to compute the resulting pixel value out of several points. Since the data is resampled into a 2D grid with a fixed spacing, several point cloud points can be within one pixel. The following modes are available:

- Average: The (mean) average of all point properties is calculated
- Standard Deviation: standard deviation of all point properties is calculated
- Minimum: The minimum value of all available point properties is used
- Maximum: The maximum value of all available point properties is used
- Median : The median of all available point properties is used
- Mode: The mode of all available point properties is used
- Closest to camera (camera view only): Only the closest points to the camera are used for calculation

2.2.5 Kernel Size

Select the size of the kernel used for rasterization. This value should always be an odd, positive integer number [3,5,7,...], if kernel size is '1' the filter is not applied.

2.2.6 Output layer options

In the field '**Output Layer**' an individual name for the layer to be created can be defined.

2.3 Creating an image layer - 'DSM'

In the example Project, the first converted layer shall contain the values of the elevation '**Z**' stored in the point cloud using the condition '**maximum**' so we assign the highest value to a pixel if more than one point falls within its boundary. This layer will later be used to classify buildings.

The Project has optical data and an imported LiDAR *.las file (from previous Lesson 1). From the .las file the intensity is shown in the image layer '**LiDAR**'. A Rule Set is provided for this exercise and needs to be loaded into the Project.

1. Right-click in the Process Tree window and select '**Load Rule Set...**'
2. Navigate to the tutorial folder and select the '**Working With LiDAR (point cloud) Files.dcp**' and confirm with OK
3. Finally, view both the optical and LiDAR data using a split screen viewer (Window → Split vertically).

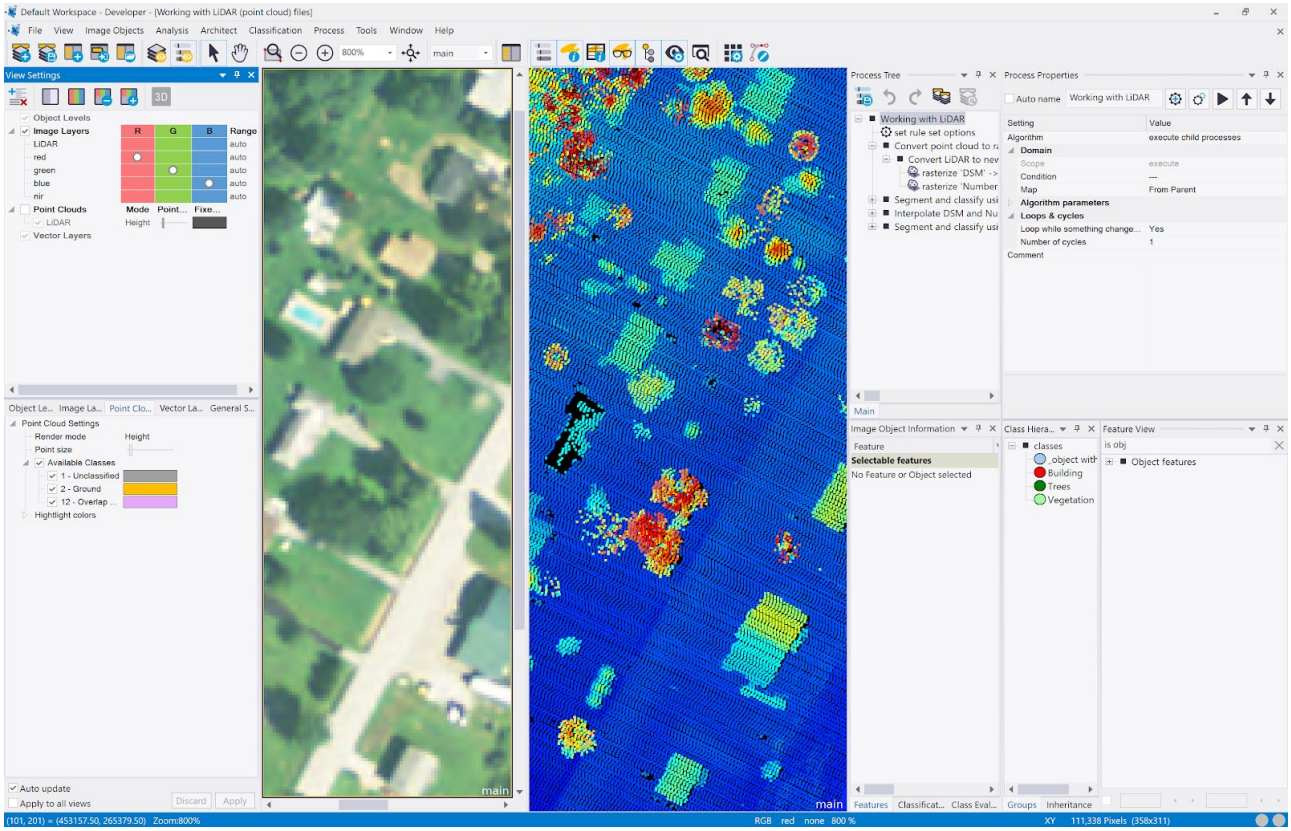

Figure 10: The Project with Rule Set loaded.

2.3.1 Create the image layer

1. In the Process Tree expand the parent processes '**Working with LiDAR (point cloud)**', '**Convert and correct**' and '**Convert LiDAR to new image layers**'.
2. Double-click on the process '**rasterize 'DSM' -> Z coordinate Maximum on [LiDAR]**' to open it.

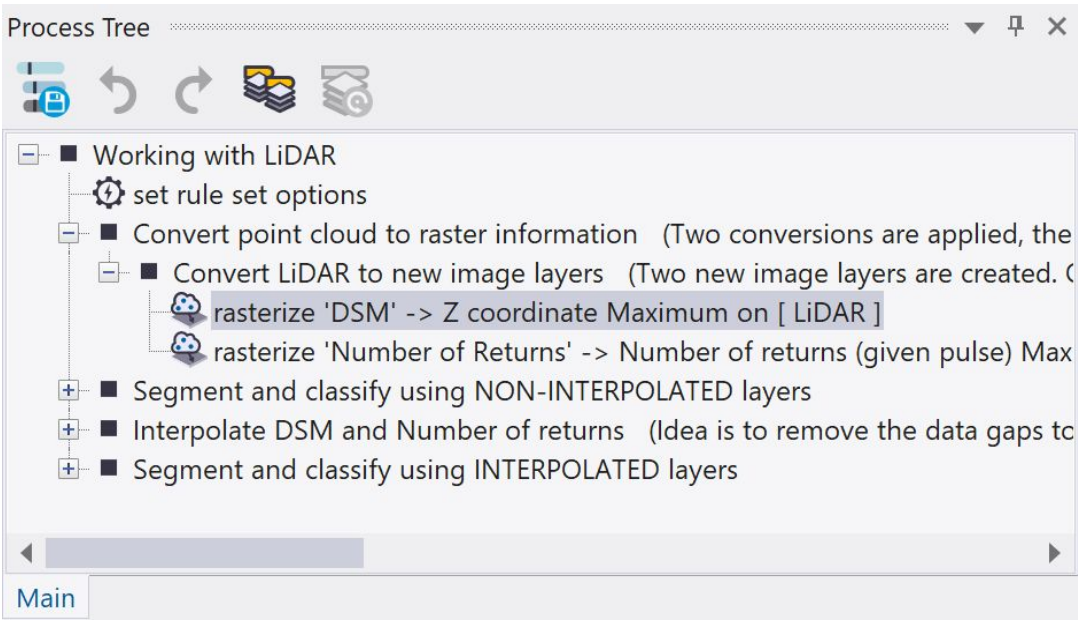

Figure 11: Process Tree with process highlighted to convert to 'DSM'.

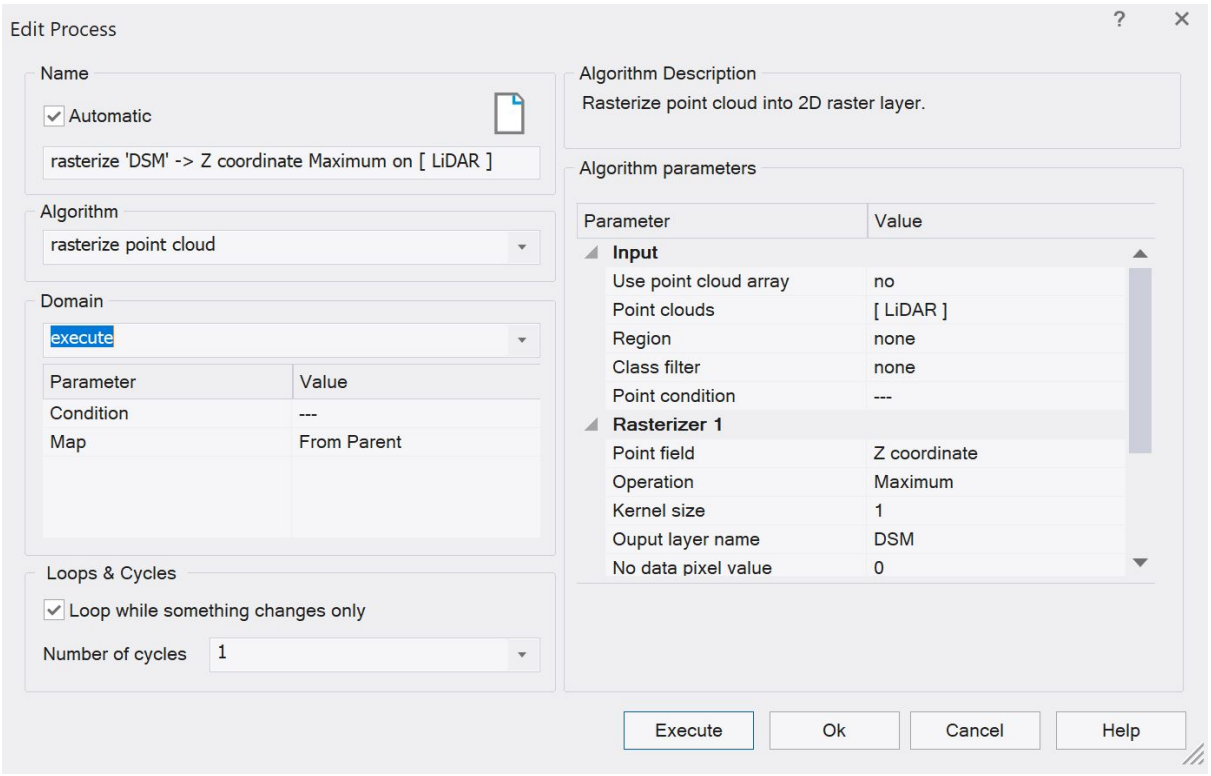

Figure 12: Process settings to convert the LiDAR point cloud to a 'DSM' layer.

The algorithm uses the following parameters:

- In the field '**Point clouds**', '**LiDAR**' is chosen from the list and defined as the source point cloud.
- In the field '**Point filed**' we have selected '**Z coordinate**' from the drop down list.
- In the field '**Operation**', the calculation mode '**Maximum**' is chosen.
- In the field '**Output** Layer **name**', the name '**DSM**' is defined. This will be the name for the new created image layer.

Execute the algorithm and review the layer created:

3. Execute the process.
4. Display in one viewer the original '**LiDAR**' image file, in a second viewer display the newly created '**DSM**' layer.
5. Hover with the mouse over the new layer '**DSM**' and evaluate the different values of this layer.

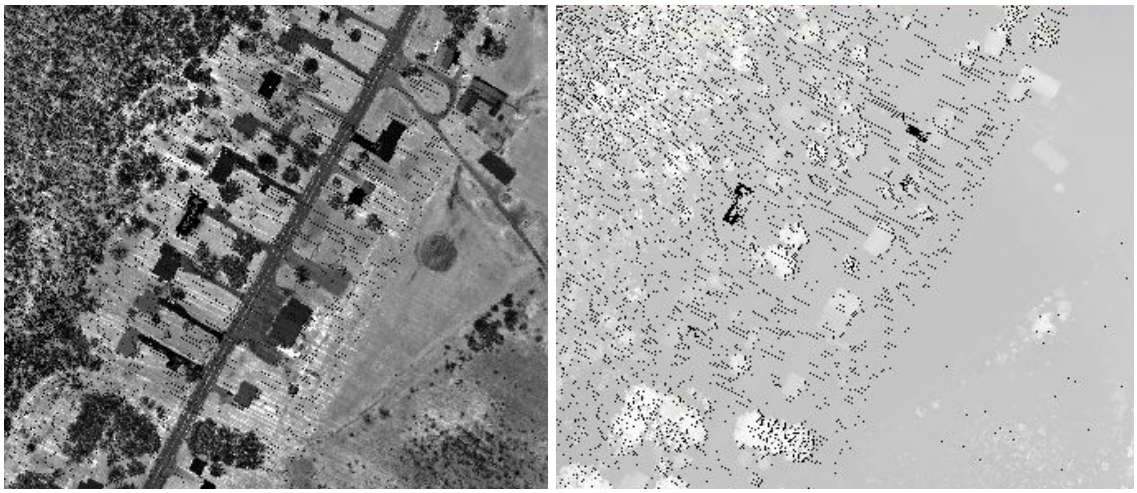

Figure 13: Initial LiDAR data (left); converted image layer 'DSM' (right).

2.4 Creating the image layer 'Number of Returns'

In the example Project, the second converted layer shall contain the value of the laser point with the maximum number of all returns. This layer will be later used to classify trees.

2.4.1 The process settings

1. Double-click on the process '**rasterize 'Number of Returns' -> Number of returns (given pulse) Maximum on [LiDAR]**' to open it.

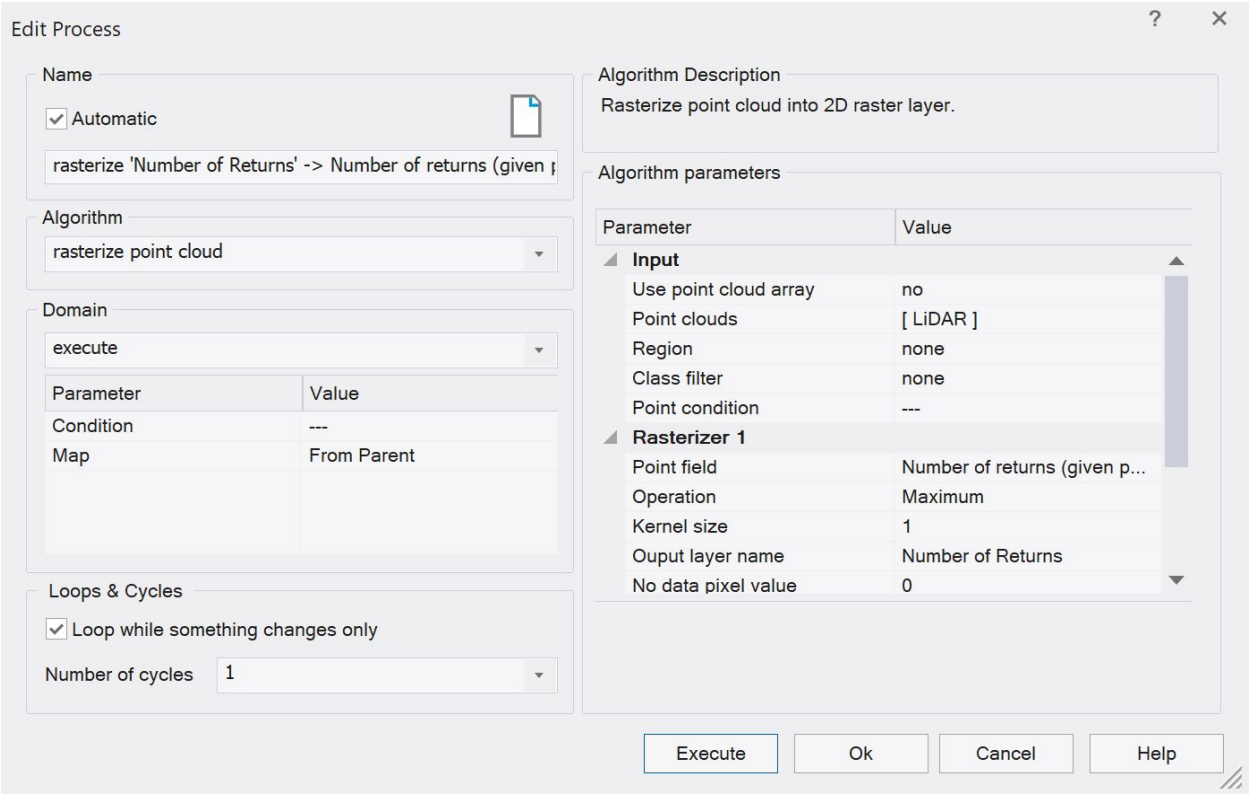

Figure 14: Process settings to convert the point cloud to 'Number of Returns'.

The algorithm uses the following parameters:

- In the field '**Point clouds**', '**LiDAR**' is chosen from the list and defined as the source point cloud.
- In the field '**Point filed**' the mode '**Number of** Re**turns (given Puls)**' is chosen from the drop down list.
- In the field '**Operation**' the calculation mode '**Maximum**' is chosen.
- In the field '**Output Layer name**' the name '**Number of Returns**' is defined.

Execute the algorithm and review the layer created:

2. Execute the process.
3. Display in one viewer the original '**LiDAR**' image file, in a second viewer display the newly created '**Number of Returns**' layer.
4. Hover with the mouse over the new layer '**Number of Returns**' and evaluate the different values of this layer.

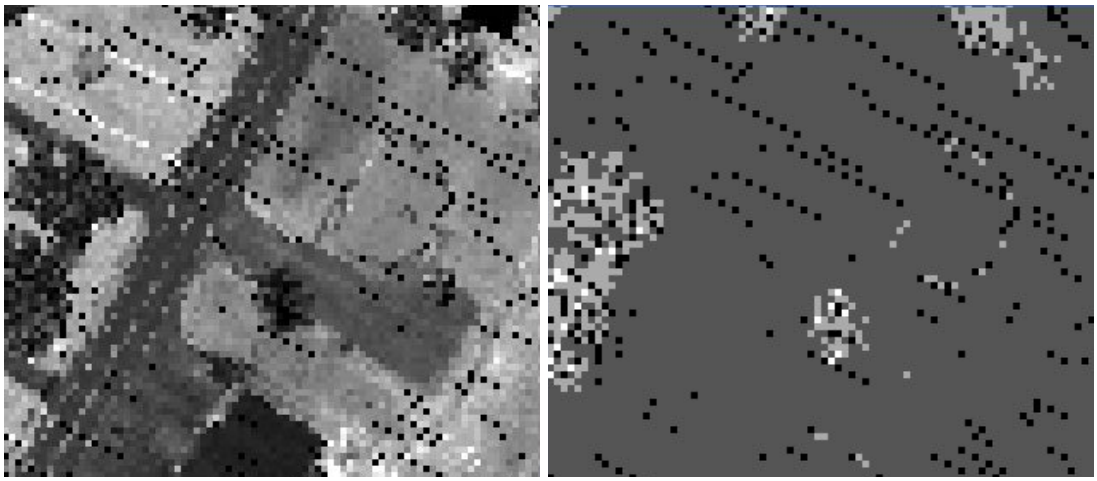

Figure 15: Initial LiDAR data (left); converted image layer 'Number of Returns' (right).

Lesson 3 – Interpolating raster data (DSM / Number of returns) based on image objects

3.1 Lesson contents

- Why interpolation?
- Introduction to interpolation of raster data based on image objects
- Storing current feature values in an object variable
- Creating new values for those objects without a value based on neighborhood properties
- Creating the new image layer from an object variable

The previously created raster representations '**DSM**' and '**Number of Returns**' shall be used to help segment and classify the trees and buildings in this subset. We first will try to segment and classify using the layers that we previously have created. Afterwards it should become clear why we need to improve our layers '**DSM**' and '**Number of Returns**'. In a later step we will apply the same segmentation and classification steps on improved (interpolated) '**DSM**' and '**Number of Returns**' to showcase what difference it makes and why it is important to remove the data gaps within our created layers. But first let's have a look at how the original layers that we have created perform in a segmentation and classification rule set.

Overall, trees are expected to have multiple returns and buildings are expected to have a high difference in elevation to the surrounding neighbors. This will serve as a basis for classification.

3.2 Segmentation without interpolating image layers

3.2.1 Creating a second map and apply a segmentation

We will use all optical layers for segmentation. Additionally, we want to use the two computed layers in the segmentation algorithm as well. The weighting is set to 10 to balance the lower value range of these layers compared to the optical values.

The processing without interpolated layers is done on a separate map to illustrate the upcoming issues if no interpolation is done. You can afterwards compare the results of the two maps in your project.

1. In the Process Tree collapse the parent processes '**Convert point cloud to raster information**'.
2. Expand the parent processes '**Segment and classify using NON-INTERPOLATED layers**'.
3. Execute the process '**copy map to** '**Map Temp**".
4. Switch the viewer to the '**Map Temp**'.
5. Expand the parent processes '**on Map Temp**'
6. Execute the multiresolution segmentation process
7. Turn on the object outlines and review the objects

Where we have 0 values, small objects are created.

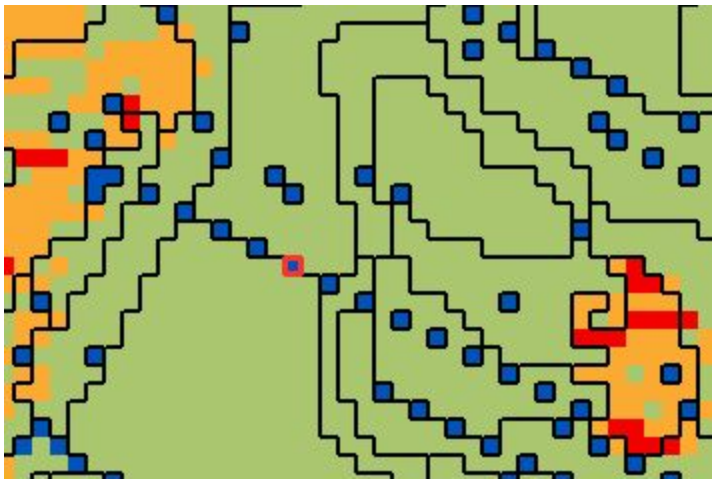

Figure 16: Pixel-sized objects are created as a result of the data gaps in one of the layers ('DSM' or 'Number of Returns').

3.2.2 Classify trees without interpolating image layers

First we classify the vegetation based on the NDVI. Afterwards we use the mean value of the image layer '**Number of Returns**' to classify Trees. In areas where the values of the image layers is 0, no Trees are classified, you should notice a lot of holes in the tree classification as a result of this.

1. Execute the process '**with NDVI > 0 at L2: Vegetation**'
2. Execute the process '**Vegetation with Mean Number of Returns > 1.1 at L2: Trees**'.
3. Display the classified objects and check your result.

You should notice that our classification '**Trees**' has a lot of holes which correspond to the data gaps of the underlying raster layer..

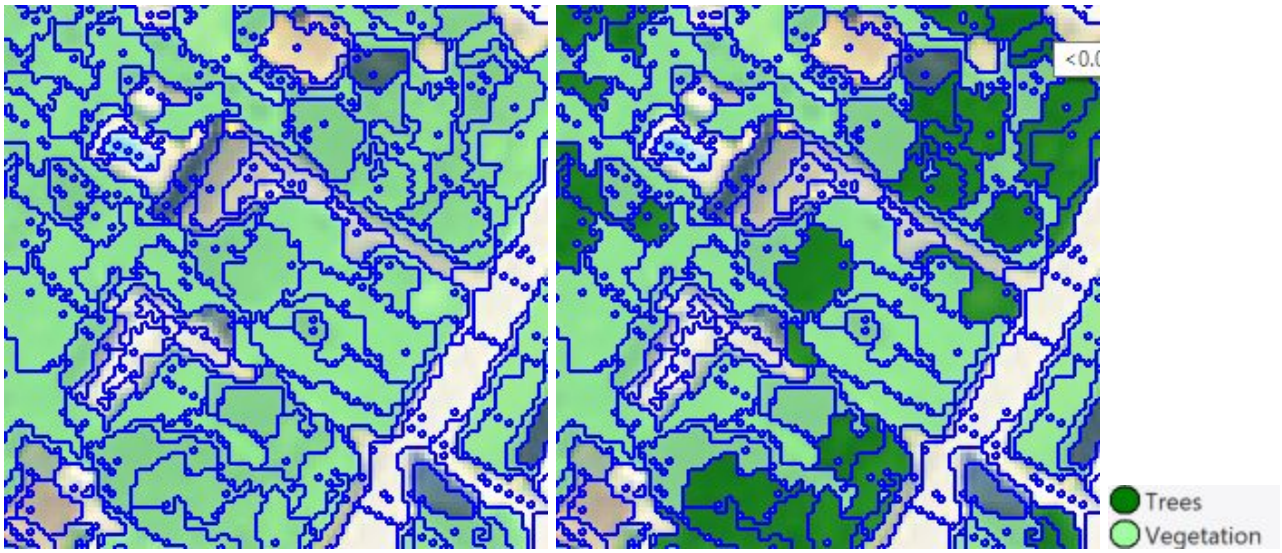

Figure 17: Classification of 'Vegetation' (left); Next step also classifying trees (right).

3.2.3 Classify buildings without interpolating image layers

Objects with differences in elevation in comparison to their neighbors are classified as '**Building**'. In areas where the values of the image layers is 0, the '**Mean difference**' calculation gives back values not fitting for '**Building**' objects, respectively other objects actually not being a building fulfill the condition. A lot of misclassifications are the result.

1. Execute the process '**loop: unclassified with MeanDiff_DSM_10 > 0.5 and Compactness < 2 at L2: Building**'.
2. Review your classification result.

A lot of objects are misclassified as '**Building**'.

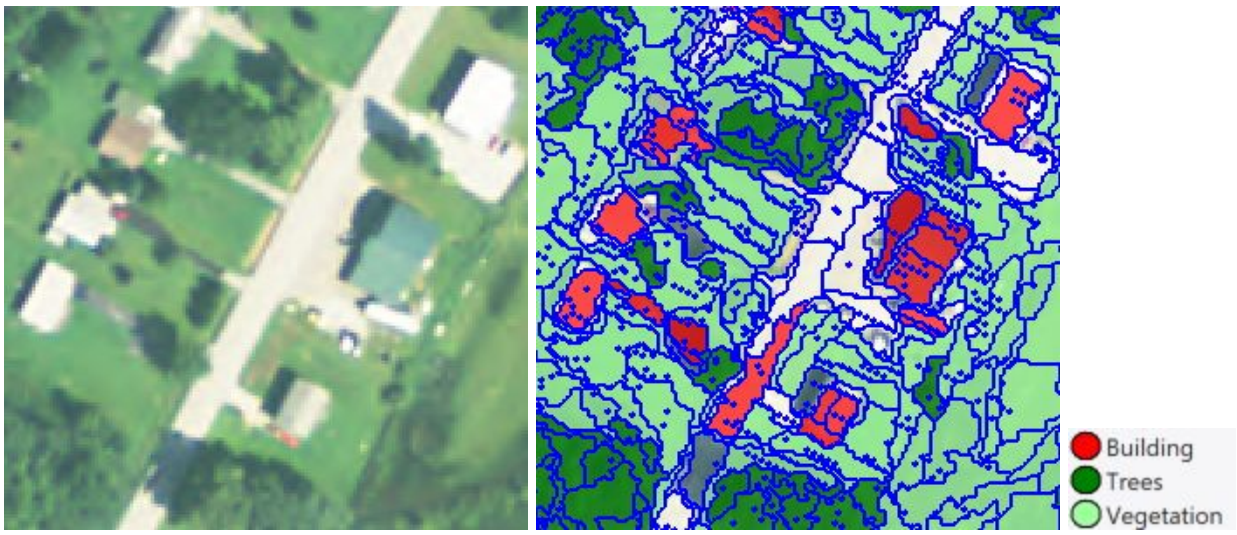

Figure 18: Optical layers (left); classification Result (right).

The issues with our segmentation, classification and the features that we are using is caused by the data gaps of the '**DSM**' and '**Number of Returns**' layer. Before using these image layers they must be interpolated to remove the pixels (to fill) with 0 values, which are disturbing the object creation and feature calculation.

3.3 Interpolation approach

To be able to create objects delineating the features that we want to extract (buildings / trees) nicely, we will need layers without data gaps as input for segmentation. If for example we want to calculate the mean difference in elevation for buildings, a 0 value object would give back misleading values.

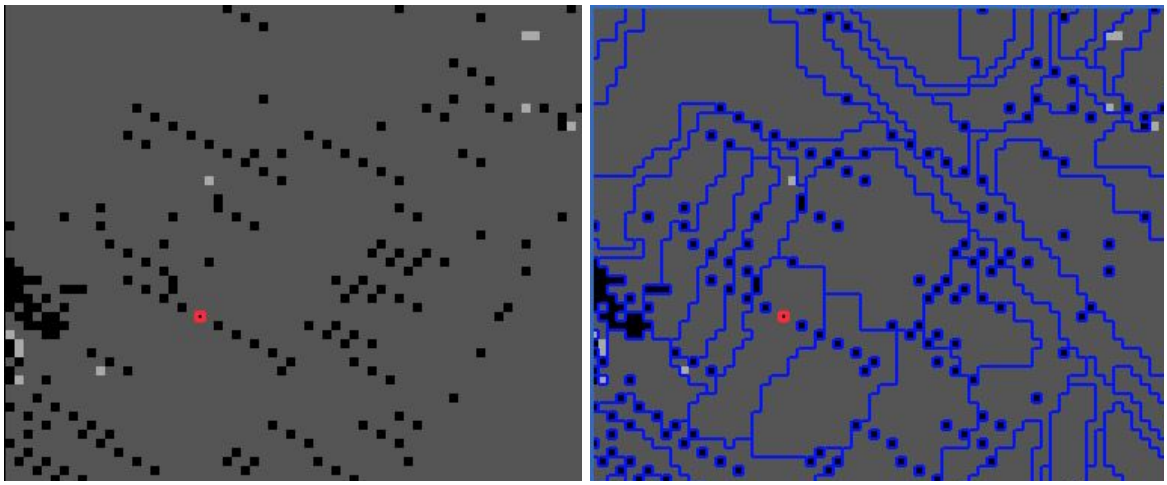

Figure 19: Not interpolated image layer (left); objects created using this layer (right).

3.3.1 Introduction to interpolation approach based on image objects

To avoid problems in object creation, feature calculation and resulting miss-classifications we will close (interpolate) our data gaps and replace the 0 value pixels based on local neighborhood interpolation of mean values. To do so an object oriented approach is used resulting in an interpolated image layer within the Project. The workflow is the following:

1. Pixel-sized segmentation (chessboard segmentation algorithm with a size of 1)
2. Classify objects that have a value >0
3. Compute a feature which gives you for each object the mean value of its surrounding objects
4. Using this feature to assign values to the objects that are not classified (0-values)
5. Repeat step 2-4 until all objects have a value
6. Create a raster based on the object values (this is the interpolated result)

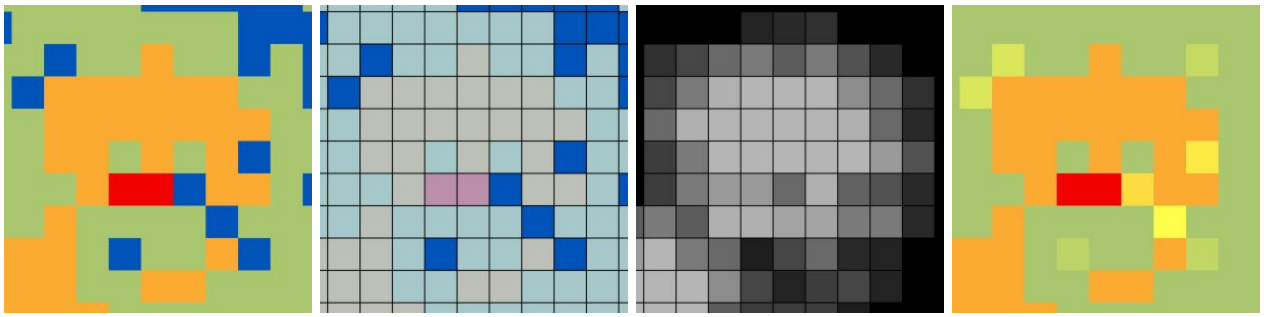

Figure 20: Showing the workflow to fill the gaps in the data set. First image from left showing the number of Returns in color scale. Blue indicates the value of '0', green '1', orange '2' and red '3'. Second image from the left shows the next step, creating a chessboard segmentation and classifying objects that have >0 Returns which leaves our data gaps as unclassified. Step three (third image from left) shows the feature we are using to fill the gaps which represents the mean value of its neighbors with a distance of 2 Pixel. Last step (image on the right) is to assign these values to the unclassified objects (data gaps). We are using a loop to do this as there are data gaps that are further away than 2 Pixels from and information..

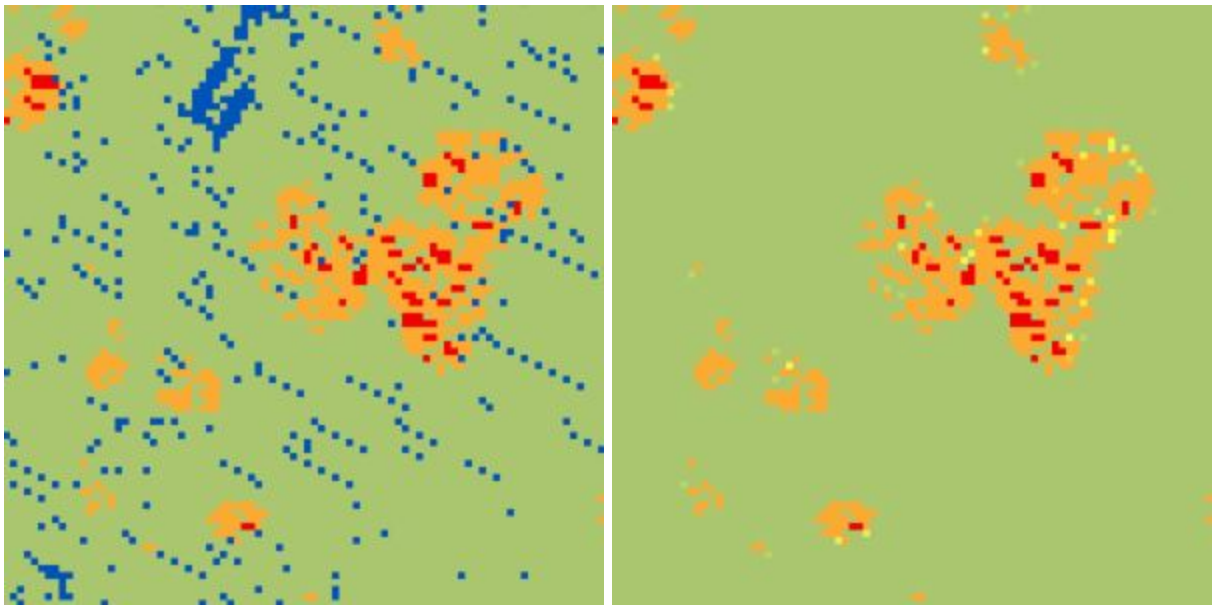

Figure 21: Not interpolated image layer (left); interpolated image layer (right).

3.3.2 Executing the rule set section for interpolation

The first step of this process sequence is to classify those objects which have a value for the '**Number of Returns**' image layer. Therefore, objects of a pixel-size are created (chessboard segmentation) and those objects are classified if they hold values >0. Later a customized class-related feature points to these classified objects.

1. Collapse the parent process '**Segment and classify using NON-INTERPOLATED layers**'.
2. Expand the process '**Interpolate DSM and Number** of **returns**' and '**Interpolating DSM**'.
3. Change the view to see the '**main**' map.

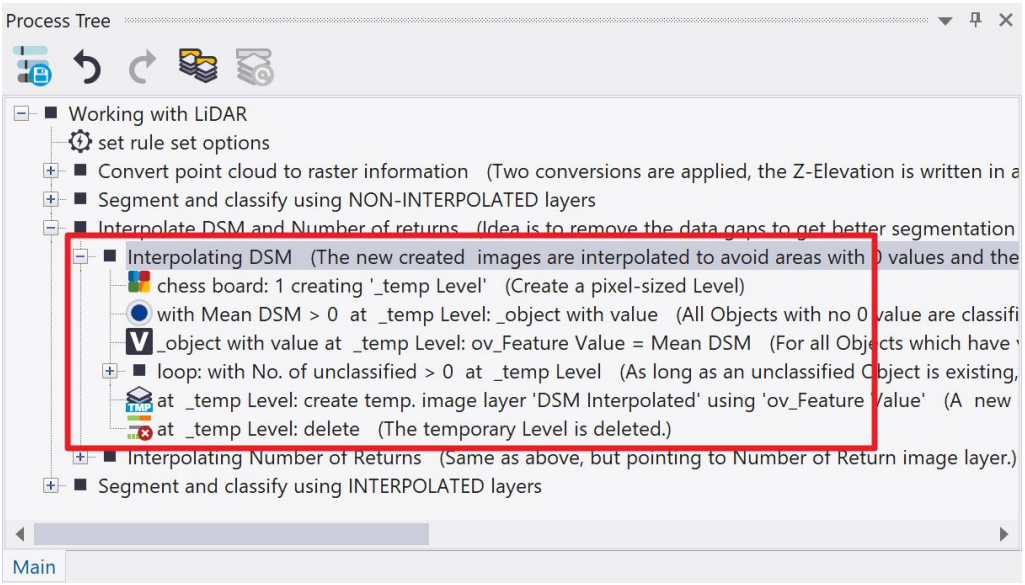

Figure 22: Process Tree with processes highlighted to interpolate the 'DSM'.

4. Execute the process '**chessboard: 1 creating '_temp Level**'.
5. Execute the process '**with Mean DSM > 0 at _temp Level: _object with value**'.

The objects with 0 value stay unclassified, the others are classified.

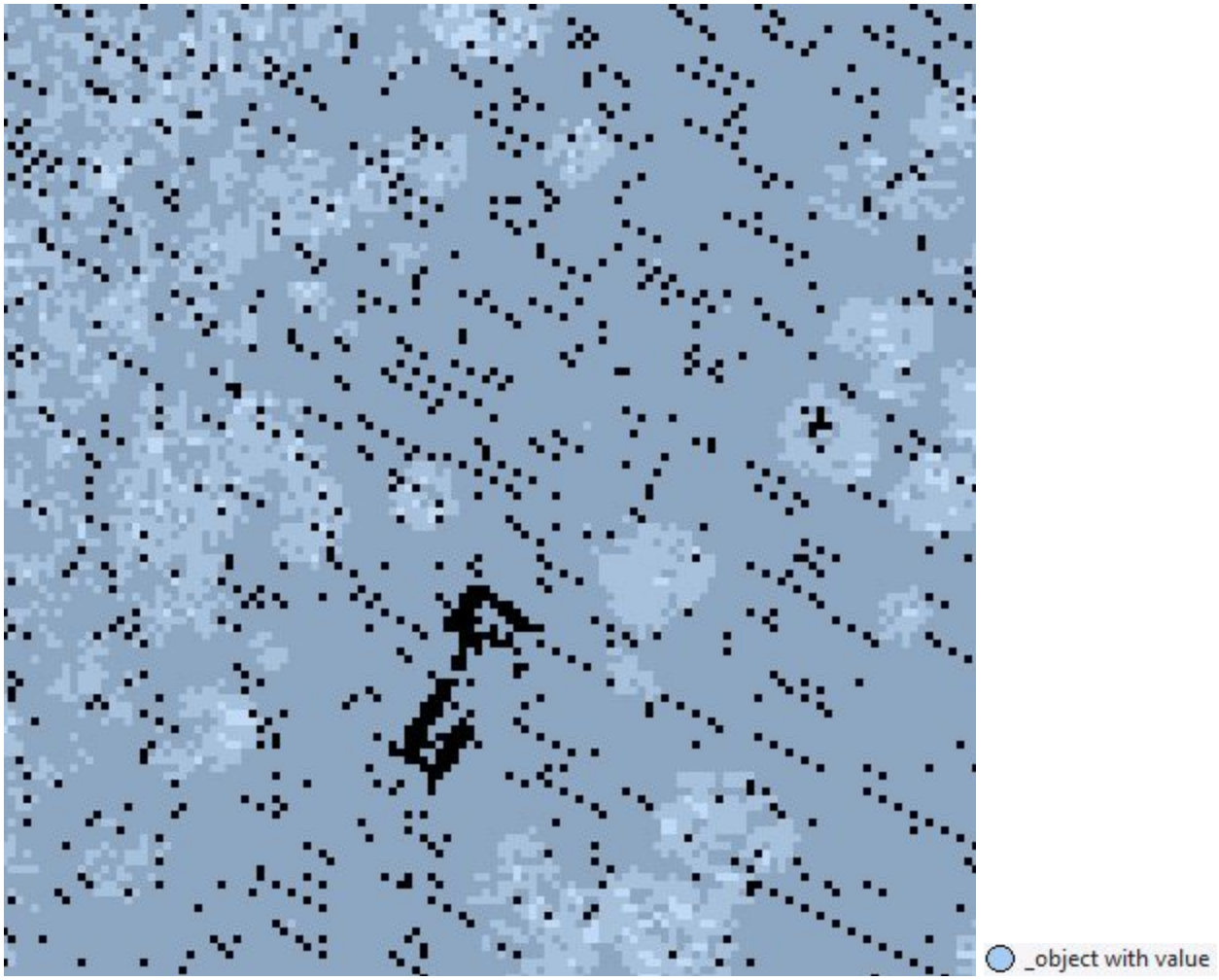

Figure 23: Classification View.

For all objects which have values, the mean values for the '**DSM**' will be written in an object variable (this equals the pixel value as the objects represent the size of a single pixel). This object variable is later used to do the interpolation of those objects with no values for the '**DSM interpolated**' target layer.

6. Double-click on the process '**_object with value at _temp Level: ov_Feature Value = Mean DSM**'.

    - The algorithm used is the '**update variable**' to update a variable
    - In the Image Object Domain, '**_objects with value**' is defined as Class.
    - As '**Variable Type**' '**Object variable**' is defined. This variable type assigns a value to every object.
    - In the field '**Variable**' the name of the variable is defined, here '**ov_Feature Value**'.

- In the '**Operations**' field '=' is defined.
- In the field '**Assignment**' '**by feature**' is defined. The value of the specified feature is written in the objects specified in the Image Object Domain.
- In the field '**Feature**' the feature '**Mean DSM**' is defined.

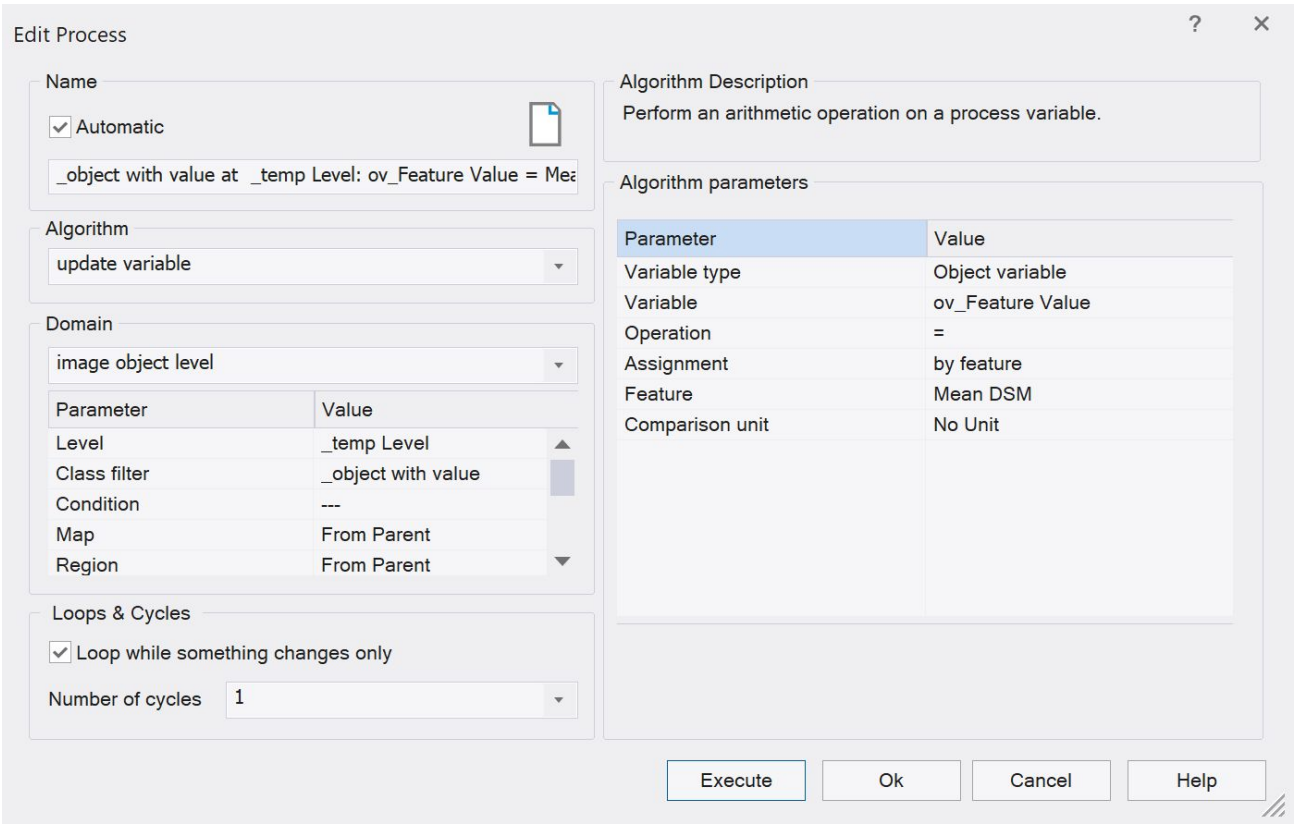

Figure 24: Process settings to update an object variable with values from feature 'Mean Number of Returns'

7. Execute the process.
8. Go to the '**Feature View**' window and browse to '**Object Features > Variables**'
9. Double-click on '**ov_Feature Value**'.

The object variable and the feature '**Mean DSM**' currently have the same values.

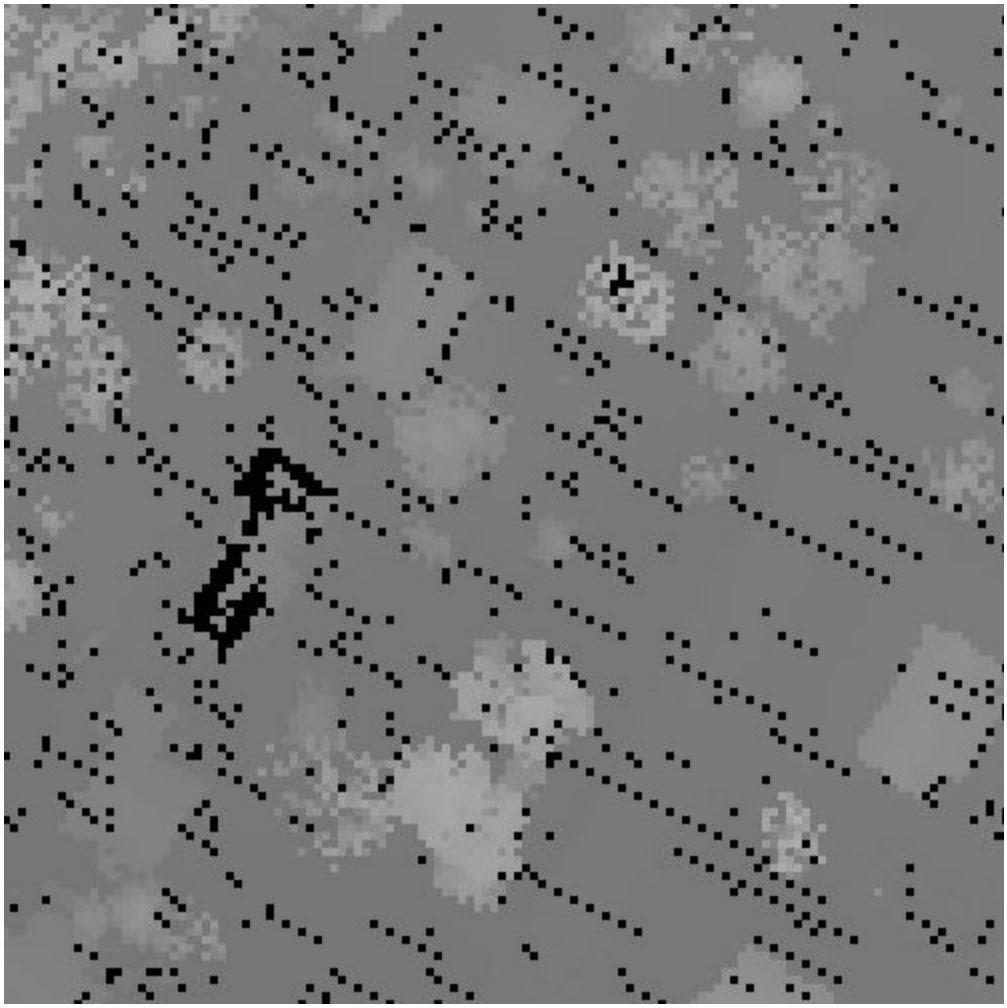

Figure 25: Feature 'ov_Feature Value' displayed.

The processing sequence to write interpolated values for objects with no value is repeated until no unclassified object remains.

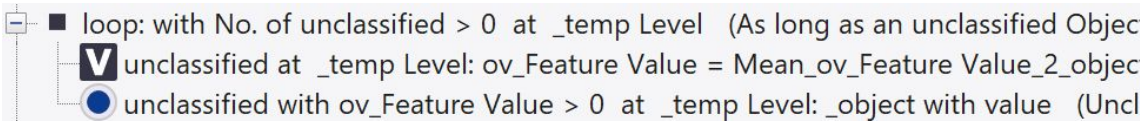

Figure 26: Loop sequence to assign values to the no data objects (unclassified).

Within the loop, for all unclassified objects, the value of a Customized Relational feature is written in the object variable. The features calculate the mean of the neighboring objects. This value is written in the object variable of the unclassified objects.

The unclassified objects which now have values for the object variable are then classified as '**_object with value**' too. This is repeated until all objects are classified, which means they have values >0.

To calculate the interpolation a customized class-related feature is used, which computes the mean of the mean values of all neighboring '**_objects with value**' objects in a certain distance. This value is then stored in the object variable for those objects, which have currently 0 values.

10. Go to the '**Feature View**' window and browse to '**Customized features → Objects → Relations to Classification**'.
11. Double-click on '**Mean_ov_Feature Value_2_objects with value**'.

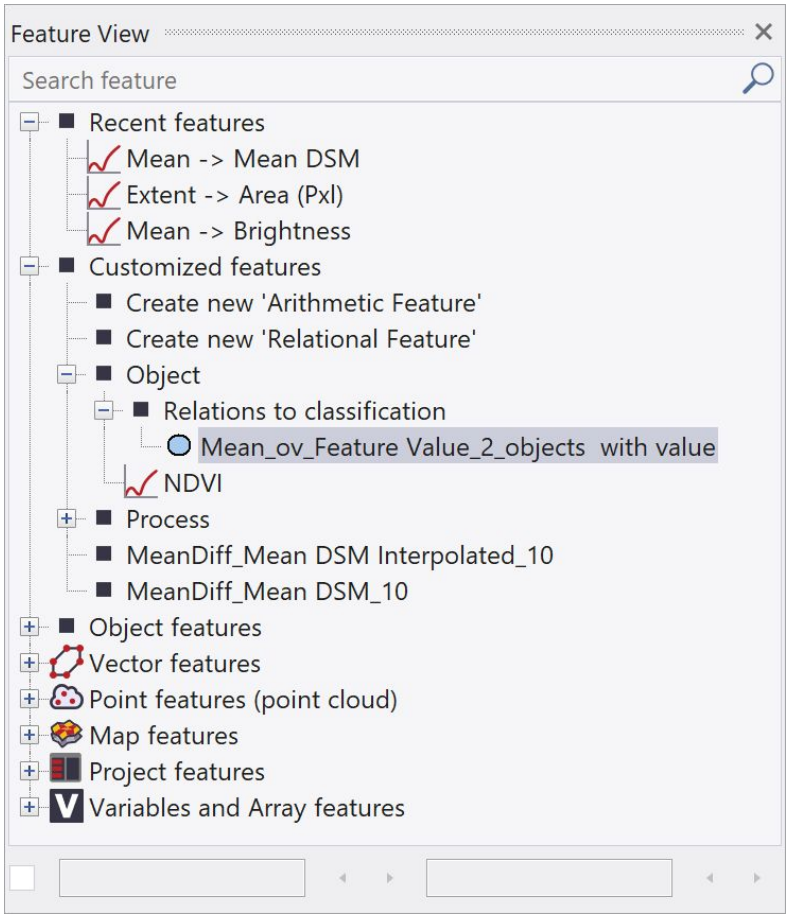

Figure 27: The customized feature calculates the mean from the mean values of the neighbors.

12. Select and execute the parent process '**loop: while No. of unclassified > 0**'.

Objects which formerly had 0 values are updated and contain now the mean of the mean values of its neighbors. To test if this worked, again display the feature '**ov_Featuer Value**' using the feature view.

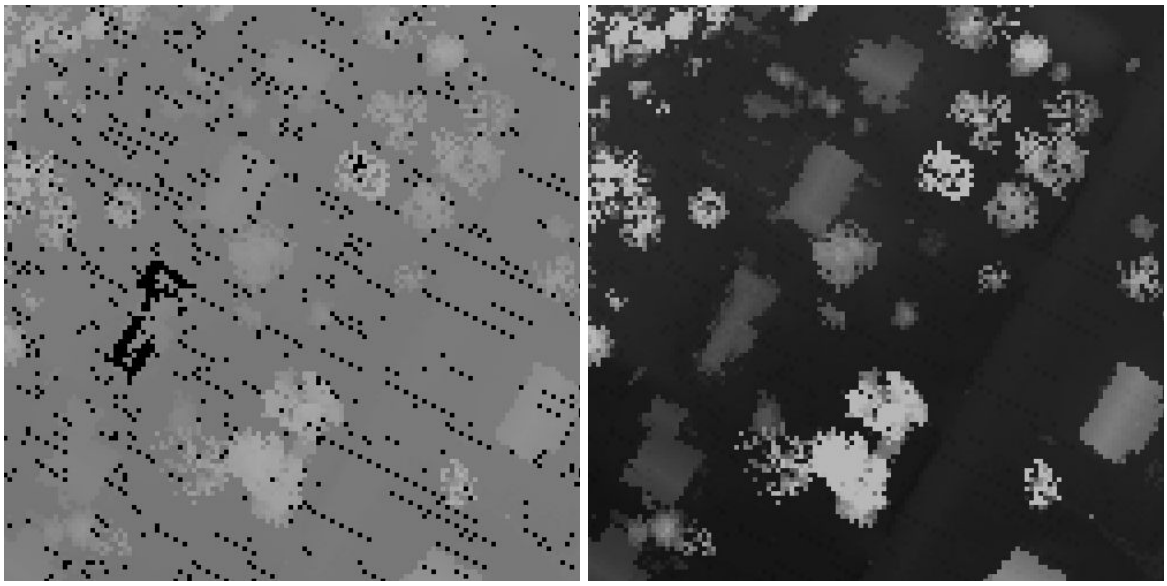

Figure 27: Mean 'DSM (left); updated object variable (right).

Now we want to create a raster layer from these object values that we just have calculated. A new image layer '**DSM Interpolated**' is created from the values of the object variable '**ov_Feature Value**' using the algorithm '**create temporary layer**'. This image layer will then be used for segmentation and classification instead of the original '**DSM**' layer in a next step.

The algorithm '**create temporary image layer**' is chosen. With this algorithm, you can create an image layer containing feature values. It can be found in the 'Image layer operation' section of the algorithms list.

13. Collapse the process sequence '**loop: while No. of unclassified > 0**'.
14. Double-click on the process '**at _temp Level: create temp. image layer 'DSM Interpolated' using 'ov_Feature Value**".
    - The Domain points to all objects of '**temp_Level**'
    - In the field '**Layer name**' the name '**DSM Interpolated**' is defined.
    - In the field '**Feature**' the '**ov_Feature Value**' is defined.

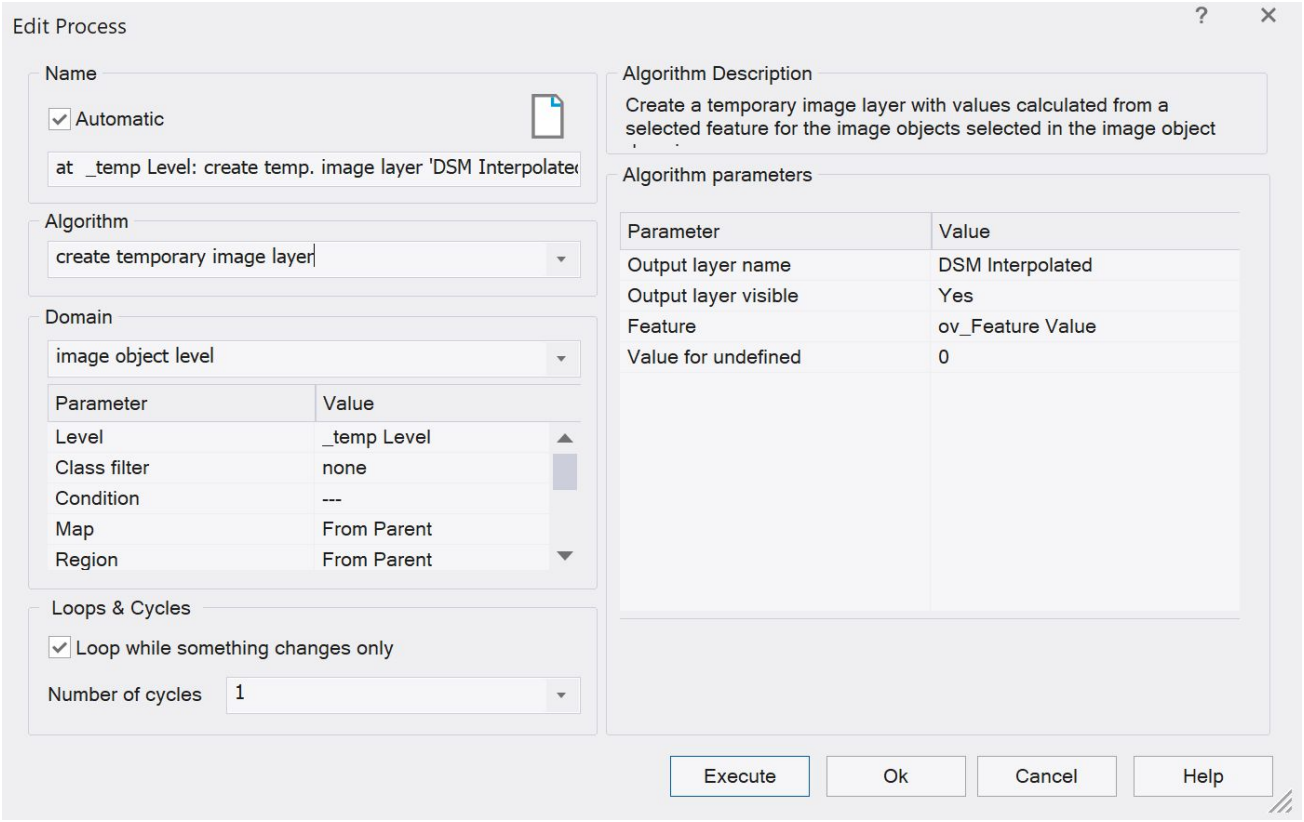

Figure 28: Process settings to create the temporary layer 'DSM Interpolated'.

15. Execute the process '**at _temp Level: create temp. image layer 'DSM Interpolated' using 'ov_Feature Value**".

The new, interpolated image layer is created.

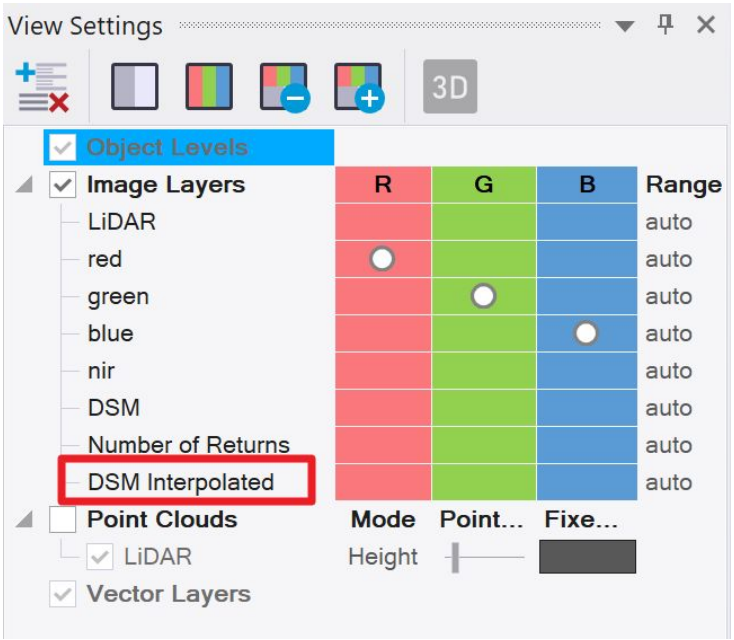

Figure 31: Our newly created raster layer 'DSM Interpolated'.

To be prepared for the next interpolation the existing image object level is deleted.

16. Execute the process '**at _temp Level: delete**'.

The same problems with 0 values also exist for the '**Number of Returns**' layer. The same interpolation process sequence is applied to create an '**Number of Returns Interpolated**' layer.

1. Collapse the process sequence '**Interpolating DSM**'.
2. Execute the parent process '**Interpolating Number of Returns**'.

After execution you will find yet another layer in your View Settings, called '**Number of Returns Interpolated**'.

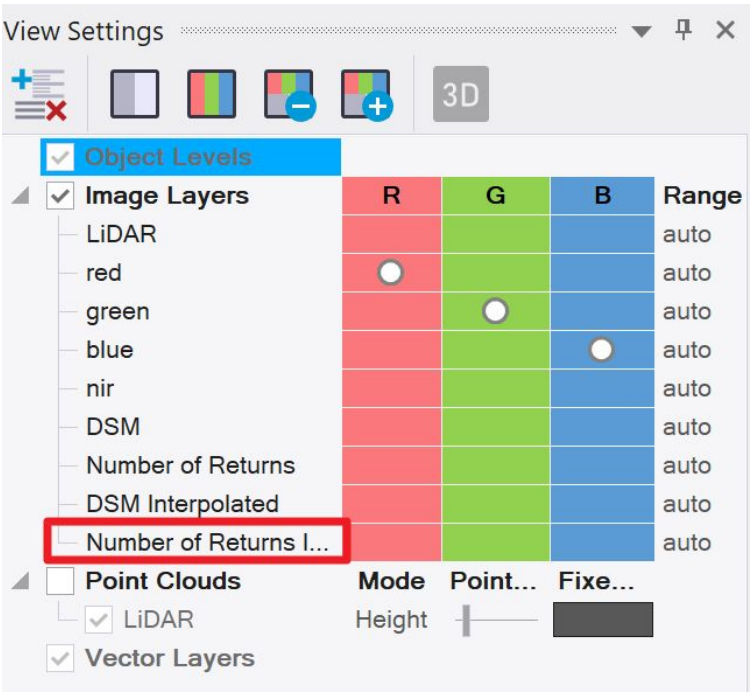

Figure 32: Our newly created raster layer 'Number of Returns Interpolated'.

Lesson 4 – Segmenting and classifying with interpolated image layers

4.1 Lesson content

- Segment with interpolated image layers
- Classify with interpolated image layers
- Classify buildings with interpolated image layers

4.2 Segment with interpolated image layers

All optical layers are used for segmentation. The two interpolated LiDAR images are used too, the weighting is set to 10 to balance the lower value range of these layers compared to the optical values.

As the layers are interpolated now, no pixel-sized objects around the 0 values are created - the segmentation produces meaningful objects.

1. In the Process Tree collapse the parent processes '**Convert and correct**'.
2. Expand the parent processes '**Segment and classify using interpolated LiDAR**'.

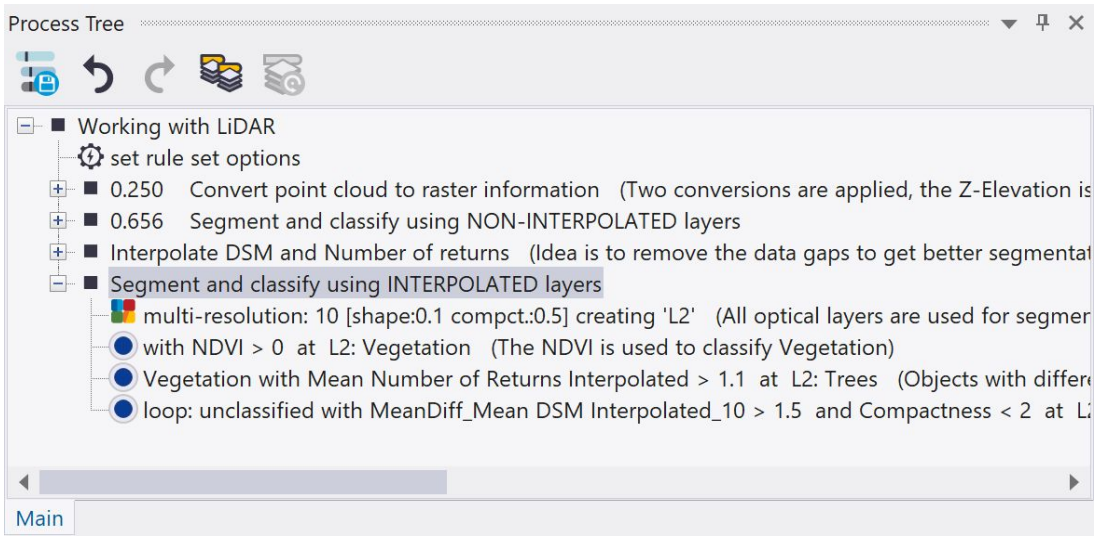

Figure 34: Process Tree with processes highlighted to segment and classify using the interpolated data.

3. Execute the process '**multi-resolution: 10 [shape:0.1 compct.:0.5] creating 'L2**"
4. Switch on the outlines and review the created objects.
5. Display in one viewer the segmentation of the main map, in the other the segmentation of the '**Map Temp**'.

The objects are created in a meaningful way.

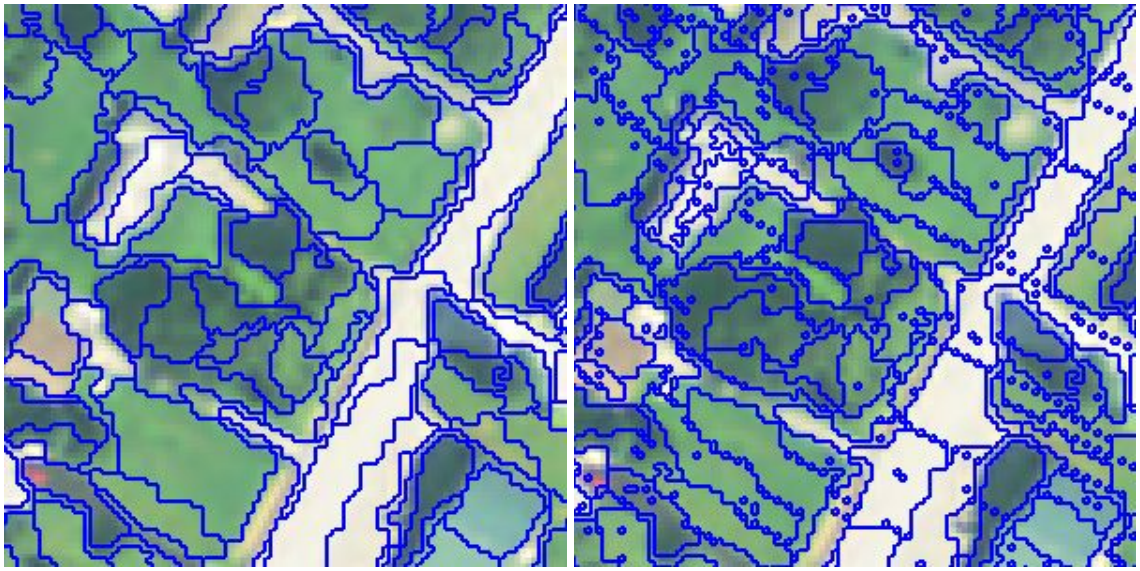

Figure 35: Objects created using the interpolated image layer within the main map (left); objects created using the not interpolated image layer within Map Temp (right).

4.3 Classify trees with interpolated image layers

The mean value of the image layer '**Number of Returns Interpolated**' is used to classify Trees. As the 0 values are corrected, the classification will not contain holes anymore.

1. Execute the process '**with NDVI > 0 at L2: Vegetation**' and '**Vegetation with Mean Number of Returns Interpolated > 1.1 at L2: Trees**'.
2. Switch on the classification view and review the classification.

The trees are classified without holes.

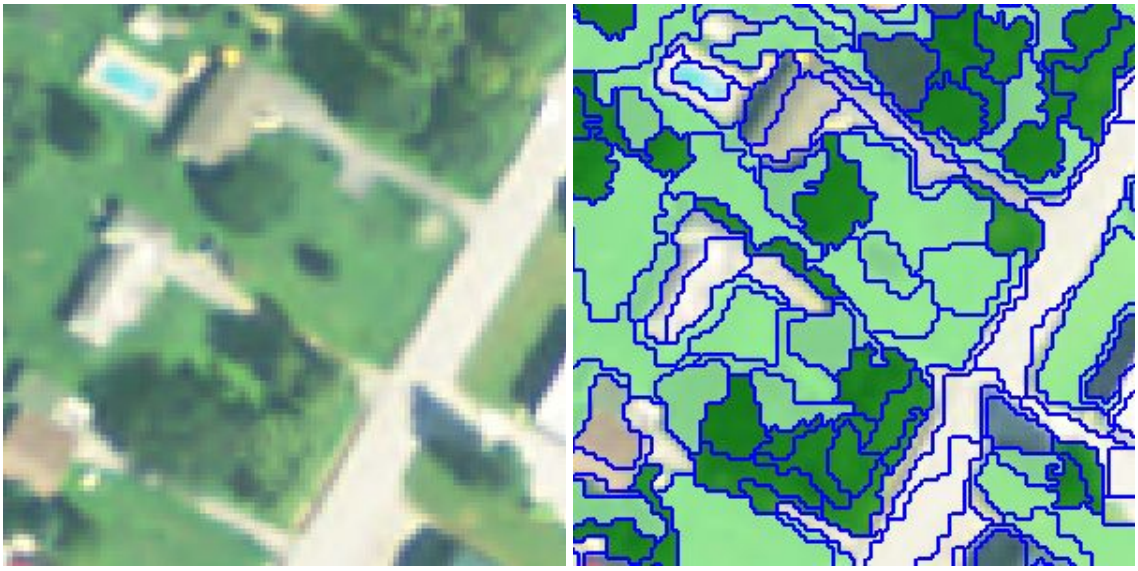

Figure 36: Optical layers (left); classification results using interpolated image layer 'Number of Returns Interpolated' (right).

Compare to results using not interpolated image layer:

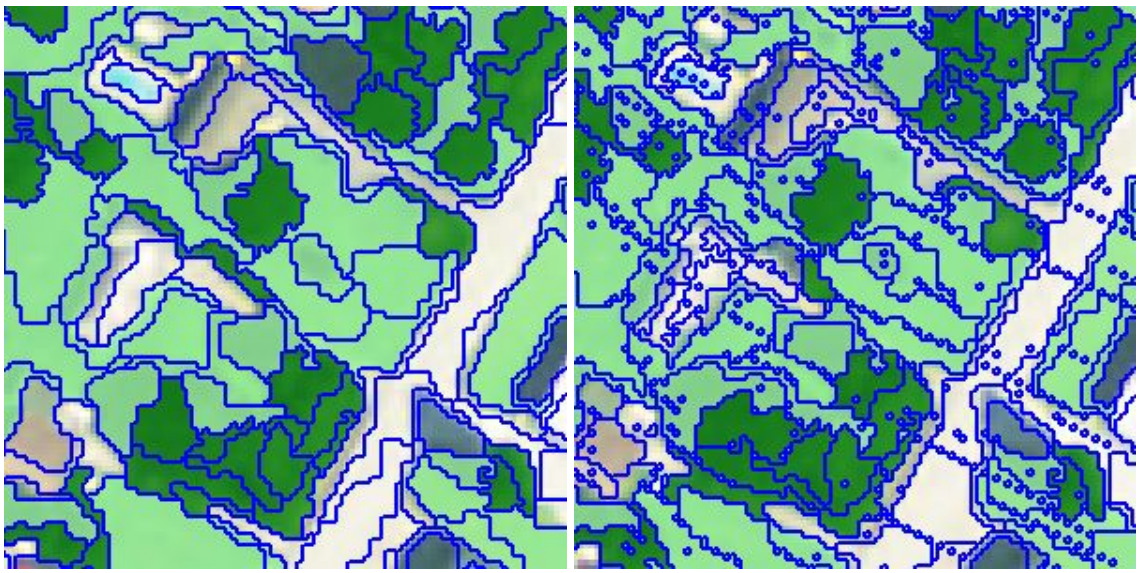

Figure 37: Classification results using interpolated image layer 'Number of Returns Interpolated' (left); classification results based on not interpolated data (right).

4.4 Classify buildings with interpolated image layers

Objects with a difference in elevation in comparison to their neighbors are classified as Building. We follow exactly the same approach as before with the non-interpolated data as input.

As the 0 values of the non-interpolated image layers are corrected, the Mean difference calculation gives back correct values fitting for Building objects, respectively other objects actually not being a building do not fulfill the condition. No misclassifications are the result.

1. Execute the process '**loop: unclassified with MeanDiff_Mean DSM Interpolated_10 > 0.5 and Compactness < 2 at L2: Building**'.
2. Switch on the classification view and review the classification.

Buildings are classified correctly, no misclassification of e.g. the road.

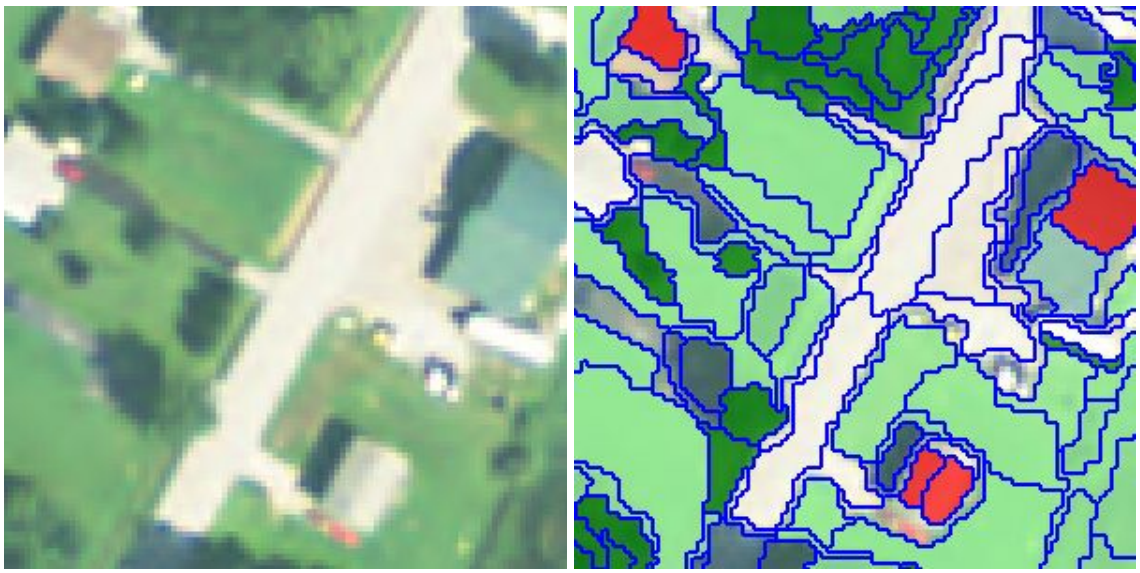

Figure 38: Optical layers (left); classification results using interpolated image layers (right).

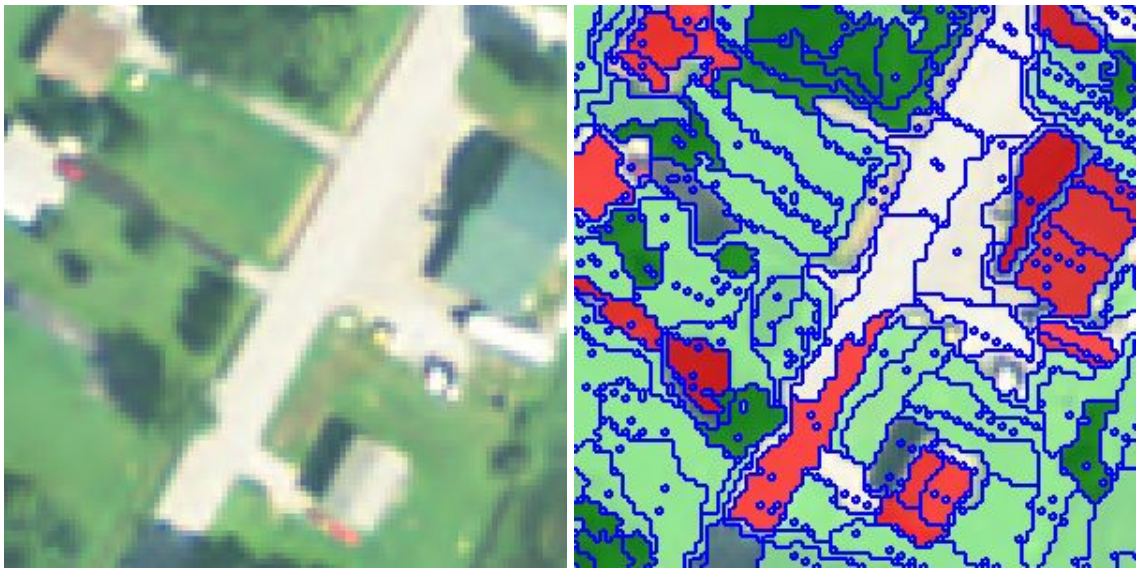

Figure 39: Optical layers (left); classification results using not interpolated image layers (right).

Where to get additional help & information?

The eCognition Community & Knowledge Base

The eCognition Community and Knowledge Base help to share knowledge and information within the user, partner, academic and developer community to benefit from each other's experience.

- The Community and Knowledge Base contain content such as:
- FAQs: Frequently Asked Questions
- Discussions: ask questions and get answers.
- Wish Lists: let us know what features and tools you want to see in upcoming software versions.
- Rule Set Exchange: share any type of eCognition related code such as Rule Sets, Action Libraries, plug-ins...
- Brainwave: tips and tricks that can improve use of the software
- Customer Stories: see how others are using eCognition and get ideas

Share your knowledge and questions with other users interested in using and developing feature extraction intelligence in the eCognition Community.

The User Guide & Reference Book

Together with the software a User Guide and a Reference book is installed. You can access them in the Developer software interface via the Help menu.

The Reference Book lists detailed information about algorithms and features, and provides general reference information.

eCognition tv

eCognition tv is our YouTube channel and contains a variety of helpful video material ranging from in-depth algorithm demonstration to customer testimonials.

The eCognition Blog

The eCogniton Blog includes a wealth of articles exemplifying a wide range of use cases from scientific research to commercial map production. You can learn, discuss and hopefully get inspired.

Additional Tutorials

If you are interested in further tutorial material, you can work through the other tutorials available in the eCognition Knowledge Base.

eCognition Training

eCognition Training Services offer a carefully planned curriculum that provides hands-on, real-world exercises. We are dedicated to enhancing our customers' data analysis skills and helping you to accomplish your eCognition goals.

Our courses are held in our classrooms around the world and on-site in our customer's facilities. We offer regular open training courses, where anyone can register and in-company training. We also offer customized courses to meet a customer's unique data analysis needs, thereby maximizing the training effect.