eCognition Developer

Tutorial 5 - working with LiDAR *.las files

www.trimble.com

TRANSFORMING THE WAY THE WORLD WORKS

Introduction

About this Tutorial

In this tutorial, you will learn how to import *.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 to working with LiDAR *.las files in eCognition
- Lesson 2 Using the LiDAR converter algorithm
- Lesson 3 Interpolating LiDAR data based on image objects
- Lesson 4 Segmenting and classifying with the interpolated image layers

Further information about eCognition products is available on our website:

www.eCognition.com

Requirements

To perform this Guided Tour, you will need:

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

All steps of this tutorial can be done using the **eCognition Developer** or the free-trial version.

This tutorial is designed for self-study.

Data included with the Tutorial

Image data

We will be working with a single image (*.img) file in this tutorial:

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

LiDAR data

In addition to the image file, a LiDAR file is available in *.las format:

- '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 files in eCognition

1.0 Lesson contents

- Working with LiDAR in this tutorial
- Loading LiDAR point cloud files

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

- LAS or
- the compressed LAZ version

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

Working with LiDAR point clouds in eCognition is implemented in a three step approach.

- LiDAR point clouds are loaded and displayed using a quick resampling of the intensity data. It is also possible to display the un-rasterized point cloud.
- Once loaded, additional layers can be generated using a special LiDAR converter algorithm.
- Gaps within the LiDAR data can then be interpolated based on image objects

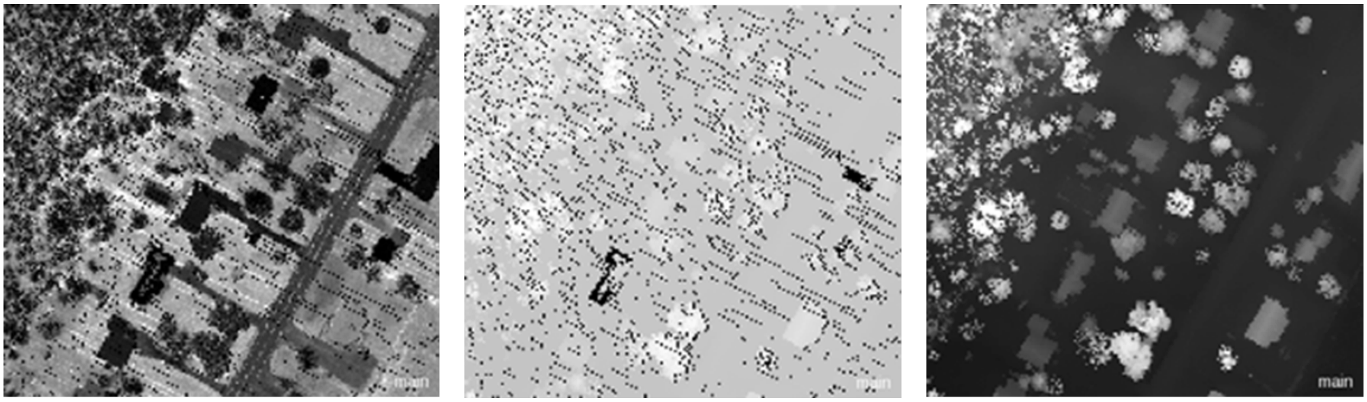


Figure 1: Rasterized LiDAR intensity data (left); first return from elevation after LiDAR conversion (middle); last return from elevation after interpolation (right).

1.1 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.

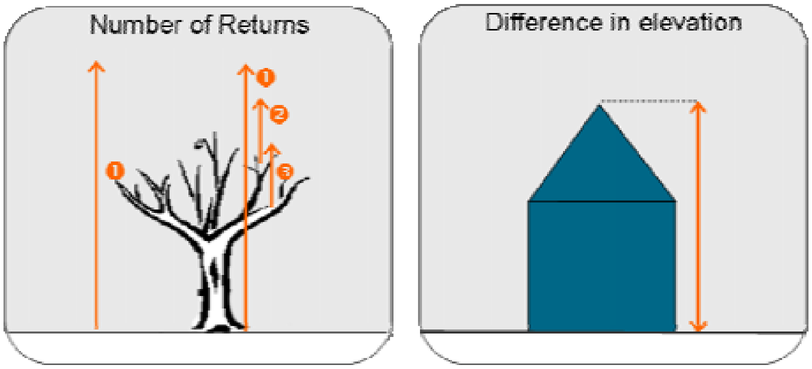


Figure 2: Schematic figure showing the different number of returns.

This information is not represented in the per default loaded LiDAR 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.

1.2 Loading LiDAR point cloud files

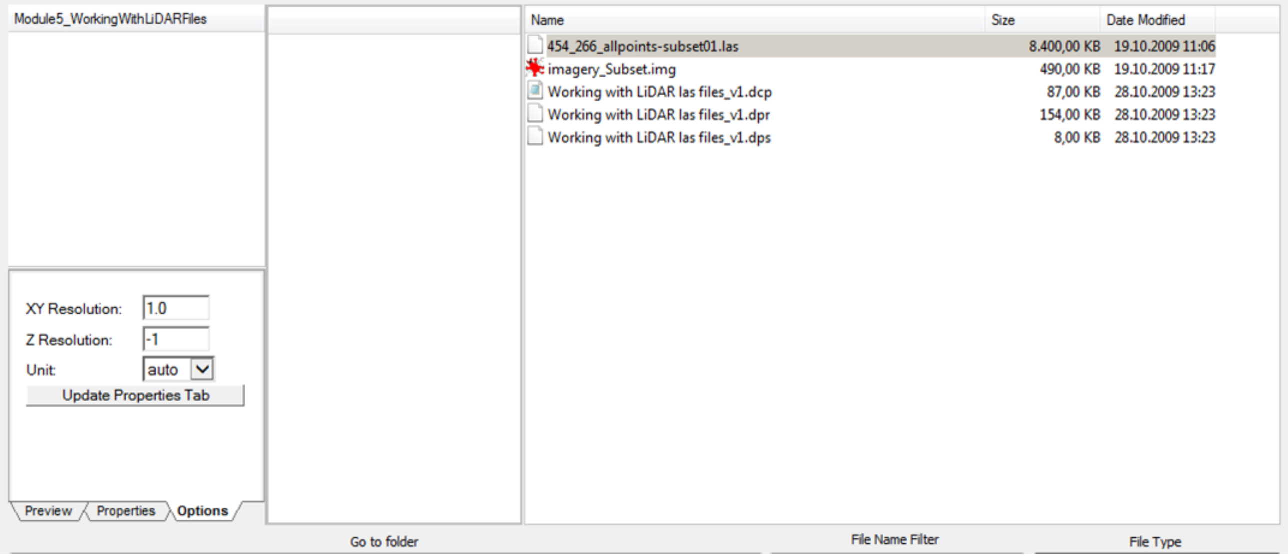
To allow for quick display of the point cloud, a rasterization is implemented in a simple averaging mode based on intensity values. Complex interpolation of data can be done based on the LiDAR file converter algorithm.

In the loading process a resolution needs to be set which determines the grid spacing of the raster generated out of the *.las or *.laz file. The resolution 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.

1.2.1 Creating a new Project using LiDAR data

A *.las file is imported, like any other image, via the 'Import Image Layers' dialog box.

1. Start **eCognition Developer** and switch to predefined view setting number 4 '**Develop rulesets**'
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 the '454_266_subset02.las' to import it. In the left hand window of the 'Import Image Layers…' dialog, the 'Options' tab contains the default point cloud settings:
    - XY Resolution = 1.0
    - Z Resolution = -1
    - Unit = 'auto'
5. Leave the default values and click OK.


Module5_WorkingWithLiDARFiles

Name	Size	Date Modified
454_266_allpoints-subset01.las	8.400,00 KB	19.10.2009 11:06
imagery_Subset.img	490,00 KB	19.10.2009 11:17
Working with LiDAR las files_v1.dcp	87,00 KB	28.10.2009 13:23
Working with LiDAR las files_v1.dpr	154,00 KB	28.10.2009 13:23
Working with LiDAR las files_v1.dps	8,00 KB	28.10.2009 13:23

XY Resolution: 1.0
Z Resolution: -1
Unit: auto

Figure 3: LiDAR file properties within the 'Options' tab.

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

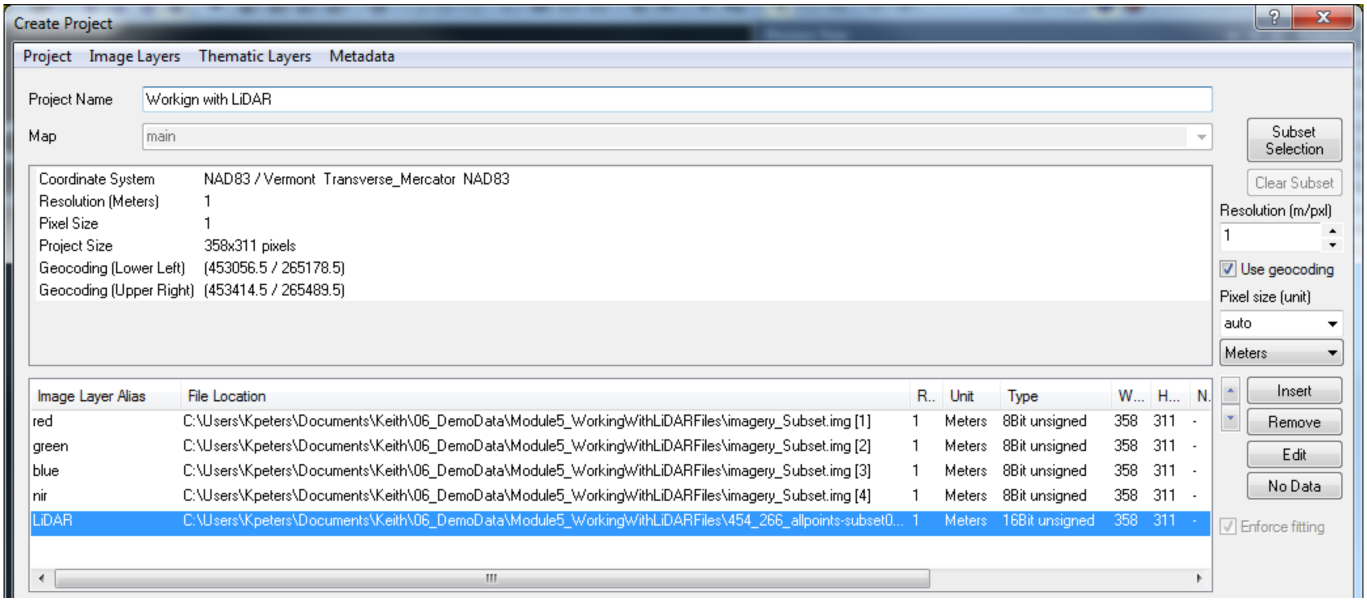
In addition to the *.las file, please load the corresponding image file:

6. Click '**Insert**' within the Image Layer section of the '**Create Project**' dialog

7. 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).

8. Please assign the following aliases:
   - **Layer 1 = LiDAR**
   - **Layer 2 = red**
   - **Layer 3 = green**
   - **Layer 4 = blue**
   - **Layer 5 = nir**
9. To adjust the order of the image layers, use the up and down arrows to the right of the Image Layer window. Move the LiDAR layer to the bottom of the list.
10. Finally, assign the Project a meaningful name, ‘Working with LiDAR’ and confirm with OK.



Project Name: Workign with LiDAR

Map: main

Coordinate System: NAD83 / Vermont Transverse_Mercator NAD83
Resolution (Meters): 1
Pixel Size: 1
Project Size: 358x311 pixels
Geocoding (Lower Left): (453056.5 / 265178.5)
Geocoding (Upper Right): (453414.5 / 265489.5)

Resolution (m/pxl): 1
☑ Use geocoding
Pixel size (unit): auto
Meters
☑ Enforce fitting

Image Layer Alias	File Location	R..	Unit	Type	W...	H...	N.
red	C:\Users\Kpeters\Documents\Keith\06_DemoData\Module5_WorkingWithLiDARFiles\imagery_Subset.img [1]	1	Meters	8Bit unsigned	358	311	-
green	C:\Users\Kpeters\Documents\Keith\06_DemoData\Module5_WorkingWithLiDARFiles\imagery_Subset.img [2]	1	Meters	8Bit unsigned	358	311	-
blue	C:\Users\Kpeters\Documents\Keith\06_DemoData\Module5_WorkingWithLiDARFiles\imagery_Subset.img [3]	1	Meters	8Bit unsigned	358	311	-
nir	C:\Users\Kpeters\Documents\Keith\06_DemoData\Module5_WorkingWithLiDARFiles\imagery_Subset.img [4]	1	Meters	8Bit unsigned	358	311	-
LiDAR	C:\Users\Kpeters\Documents\Keith\06_DemoData\Module5_WorkingWithLiDARFiles\454_266_allpoints-subset0...	1	Meters	16Bit unsigned	358	311	-

Figure 4: ‘Create Project’ dialog settings.

NOTE:
It is not possible to adjust the order of image layers after the Project has been completed. This must be done in the initial Project setup.

1.2.2 Review the input data

By default, LiDAR point clouds are loaded and displayed using a quick resampling of the intensity data. Use the ‘Edit Image Layer Mixing’ tool to look at just the LiDAR layer in this rasterized format.

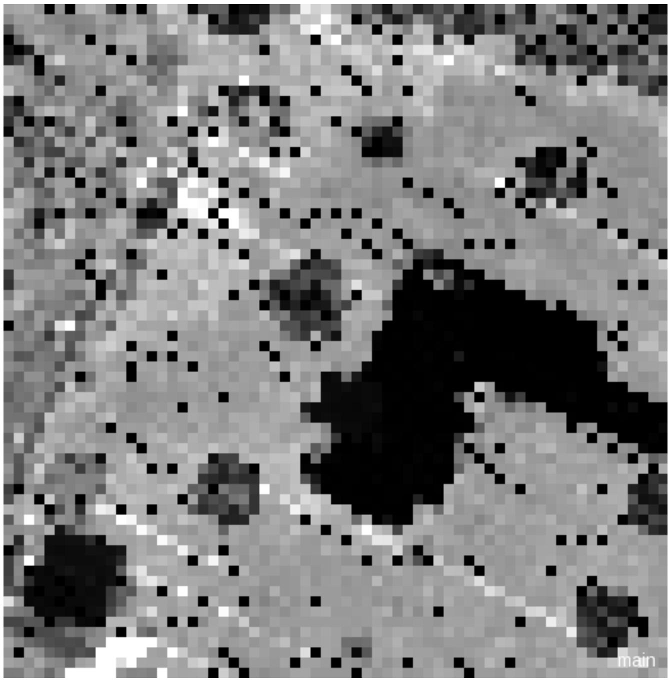


Figure 5: Detail of loaded LiDAR data.

Alternatively, you can display the point cloud itself using the ‘**Point Cloud View**’ tool available in the ‘View Settings’ toolbar. This view setting is automatically available when a point cloud is loaded into a Project.

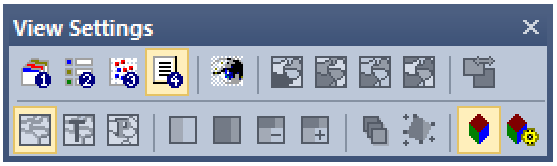


Figure 6: ‘View Settings’ toolbar with the ‘Point Cloud View’ mode selected.

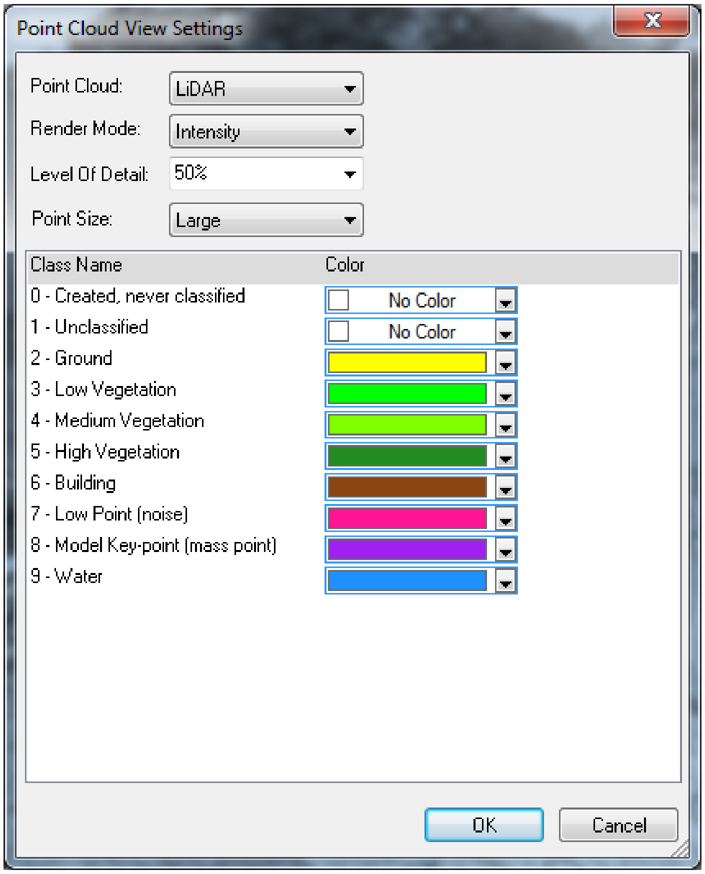
To navigate through the point cloud, you can use the various buttons of your computer mouse:

- Hold the **left** mouse button to zoom in and out while moving the mouse
- Hold the **right** mouse button to toggle the point cloud view
- Hold **both** mouse buttons 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’ toolbar.

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

- **Render Mode**: RGB, Intensity, Class/Intensity and Class/RGB
  By default the intensity mode is used.
- **Level of detail** can be used to adjust the rendering detail. When working with large, high resolution LiDAR, you may want to decrease the level of detail to increase display speed.
- **Point Size** can be used to adjust the size of the rendered points.



Point Cloud View Settings

Point Cloud: LiDAR

Render Mode: Intensity

Level Of Detail: 50%

Point Size: Large

Class Name	Color
0 - Created, never classified	□ No Color
1 - Unclassified	□ No Color
2 - Ground	🟨
3 - Low Vegetation	🟩
4 - Medium Vegetation	🟩
5 - High Vegetation	🟩
6 - Building	🟫
7 - Low Point (noise)	🟥
8 - Model Key-point (mass point)	🟪
9 - Water	🟦

Figure 7: 'Point Cloud View Settings' dialog.

1. **Explore** the LiDAR file with the 'Point Cloud View'
2. Adjust the **view settings** to see how the effect the view
3. Once you are comfortable with the 'Point Cloud View' tool, click the 'Point Cloud View' button to exit and return to the standard viewer.

Lesson 2 - Using the LiDAR converter algorithm

2.0 Lesson content

- Introduction to the ‘LiDAR File Converter’ algorithm
- Creating an image layer - ‘Elevation First’
- 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.

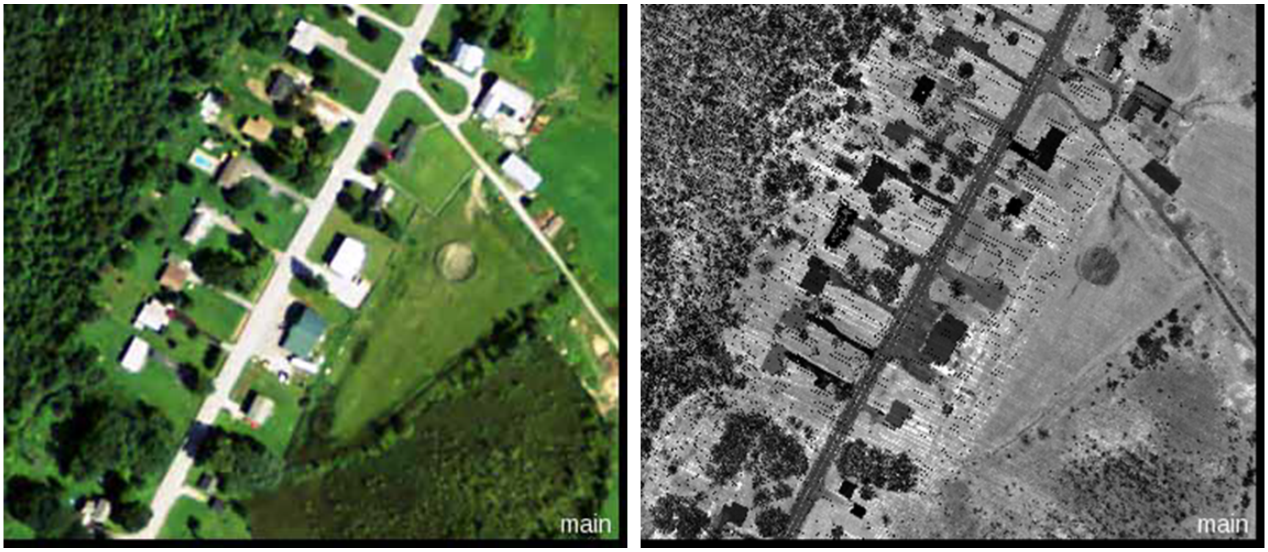


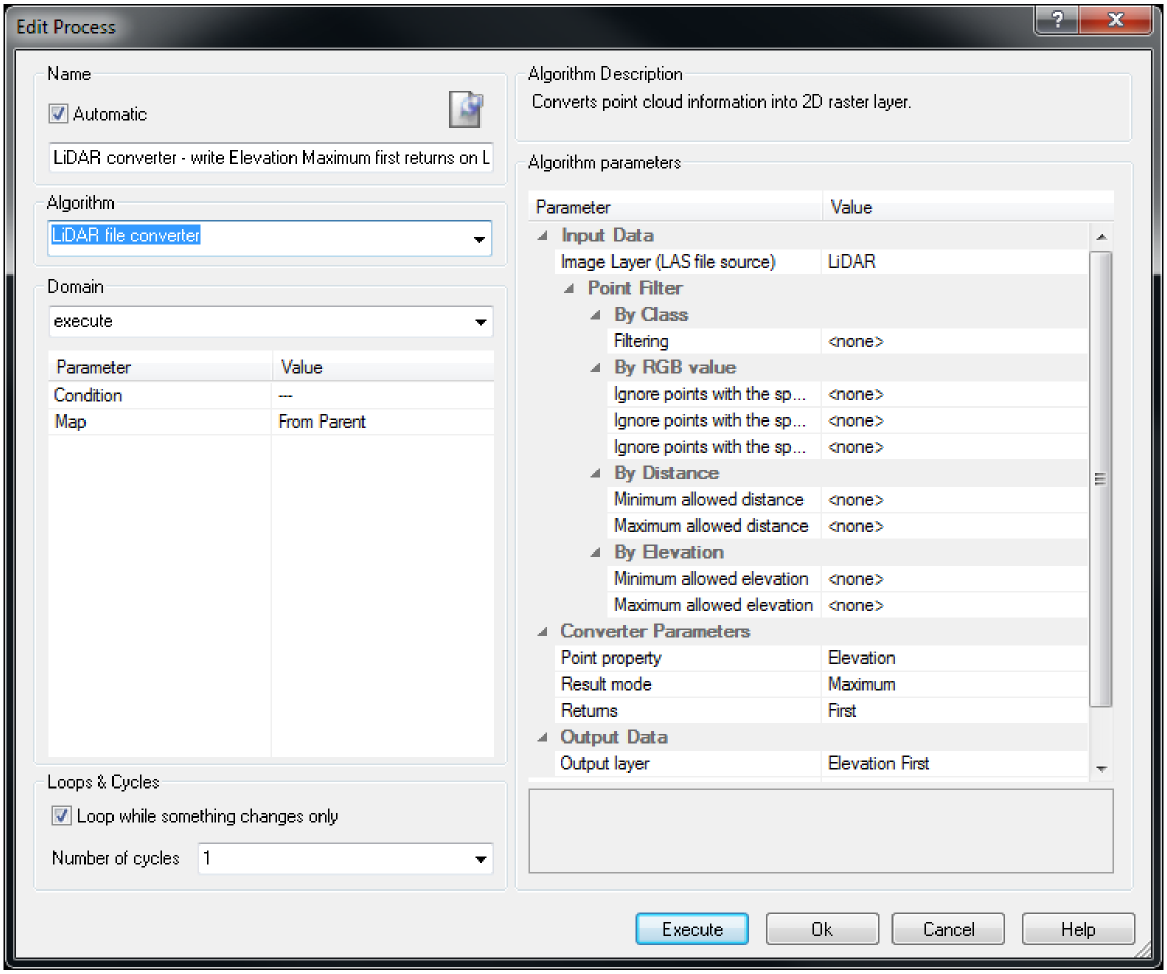
Figure 8: Optical image layers (left); loaded LiDAR data (right).

From the loaded LiDAR data, additional information can be extracted using the algorithm ‘LiDAR File Converter’. 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.

2.1 Introduction to the ‘LiDAR File Converter’ algorithm

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



Edit Process

Name

☑ Automatic

LiDAR converter - write Elevation Maximum first returns on L

Algorithm: LiDAR file converter

Domain: execute

Parameter	Value
Condition	---
Map	From Parent

Loops & Cycles

☑ Loop while something changes only

Number of cycles: 1

Algorithm Description

Converts point cloud information into 2D raster layer.

Algorithm parameters

Parameter	Value
Input Data	
Image Layer (LAS file source)	LiDAR
Point Filter	
By Class	
Filtering	<none>
By RGB value	
Ignore points with the sp...	<none>
Ignore points with the sp...	<none>
Ignore points with the sp...	<none>
By Distance	
Minimum allowed distance	<none>
Maximum allowed distance	<none>
By Elevation	
Minimum allowed elevation	<none>
Maximum allowed elevation	<none>
Converter Parameters	
Point property	Elevation
Result mode	Maximum
Returns	First
Output Data	
Output layer	Elevation First

Figure 9: Process settings to convert the LiDAR to an new image layer.

2.1.1 Define image Layer

In the field '**Image Layer (LAS source file)**', you define the layer to be use for converting, respectively the dataset from which to extract additional layers.

2.1.2 Conversion parameters

In the field '**Convert Parameter**', you define which information to use for generating new image layers. The following parameters can be set:

- **Intensity -** the point intensity values are used to generate the raster layer.
- **Elevation** - the point elevation values are used to generate the raster layer.
- **Class -** the class information stored in the points is used to generate the raster layer.
- **Number of returns** - a raster layer representing the number of returns is generated.

Since the data is resampled into a 2D Grid with a fixed spacing during import, several LiDAR points can be within one pixel.

2.1.3 Returns

In the field '**Returns**' it can be defined which points to use in generating the raster. The following settings are available.

- **First** - only first return points are used
- **Last** - only last return points are used
- **All** - all points are used

2.1.4 Calculation parameters

In the field '**Calculation**' the calculation mode can be defined, which allows determines how to compute the resulting pixel value out of several LiDAR points. The following modes are available:

- **Average** - the average out of all point values is calculated
- **Minimum** - the minimum value out of all available point values is used
- **Maximum** - the maximum value out of all available point values is used
- **Median (lower)** - the lower median out of all available point values is used
- **Median (upper)** - the upper median out of all available point values is used
- **Most frequent value** - the most frequent value out of all available point values is used

2.1.5 Filtering settings

In the field '**Filtering**' different filtering modes allow to limit the points used for generating the raster representation based on available point properties. The following options are available:

- **No filtering** - no filtering is applied. This is the default setting
- **Selected classes** - all operations are applied to points of the selected classes only

If filtering is applied, a list of classes as standardized for the *.las file format is provided. Standard classes allow using predefined standardized class types, for custom classes the other classes selection is available.

2.1.6 Output layer options

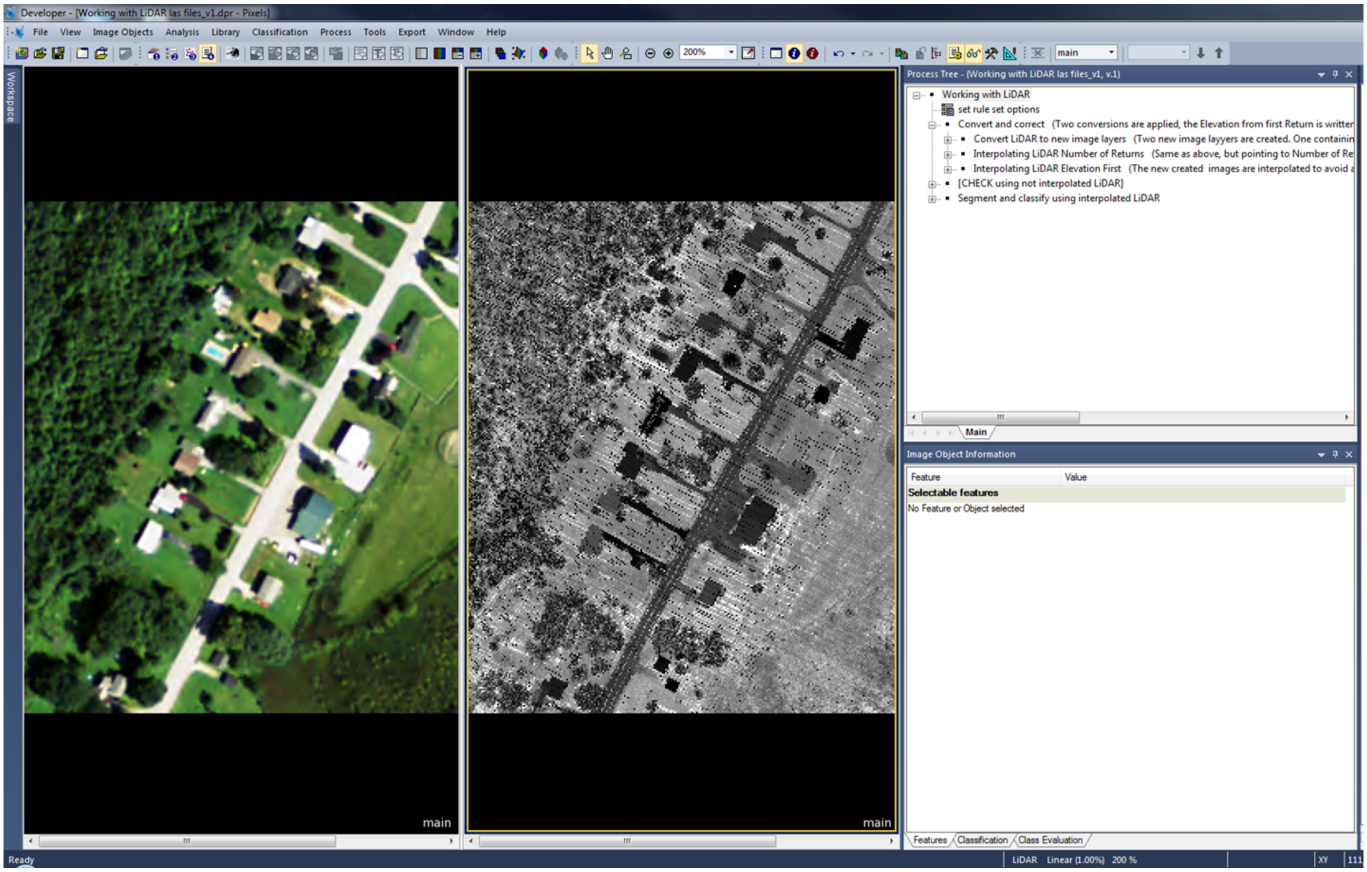
In the field '**Output Layer**' an individual name for the layer to be created can be defined.

2.2 Creating an image layer - 'Elevation First'

In the example Project, the first converted layer shall contain the value of the laser point with the maximum value of first return for elevation. This layer will later be used to classify buildings.

The Project has optical data and an imported LiDAR *.las file. 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 las files_v1.dcp**' and confirm with OK
3. Finally, view both the optical and LiDAR data using a split screen viewer.


Developer - [Working with LiDAR las files_v1.dpr - Pixels]
File View Image Objects Analysis Library Classification Process Tools Export Window Help
main
Image Object Information
Feature
Value
Selectable features
No Feature or Object selected
Features
Classification
Class Evaluation
Ready
LiDAR
Linear (1.00%)
200 %

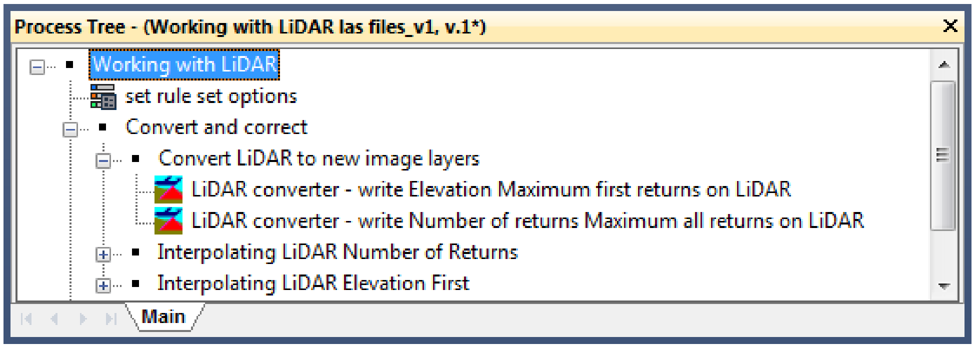
Process Tree - (Working with LiDAR las files_v1, v.1)

- Working with LiDAR
  - set rule set options
  - Convert and correct (Two conversions are applied, the Elevation from first Return is writter
    - Convert LiDAR to new image layers (Two new image layyers are created. One containin
    - Interpolating LiDAR Number of Returns (Same as above, but pointing to Number of Re
    - Interpolating LiDAR Elevation First (The new created images are interpolated to avoid a
  - [CHECK using not interpolated LiDAR]
  - Segment and classify using interpolated LiDAR

Figure 10: The Project with Rule Set loaded.

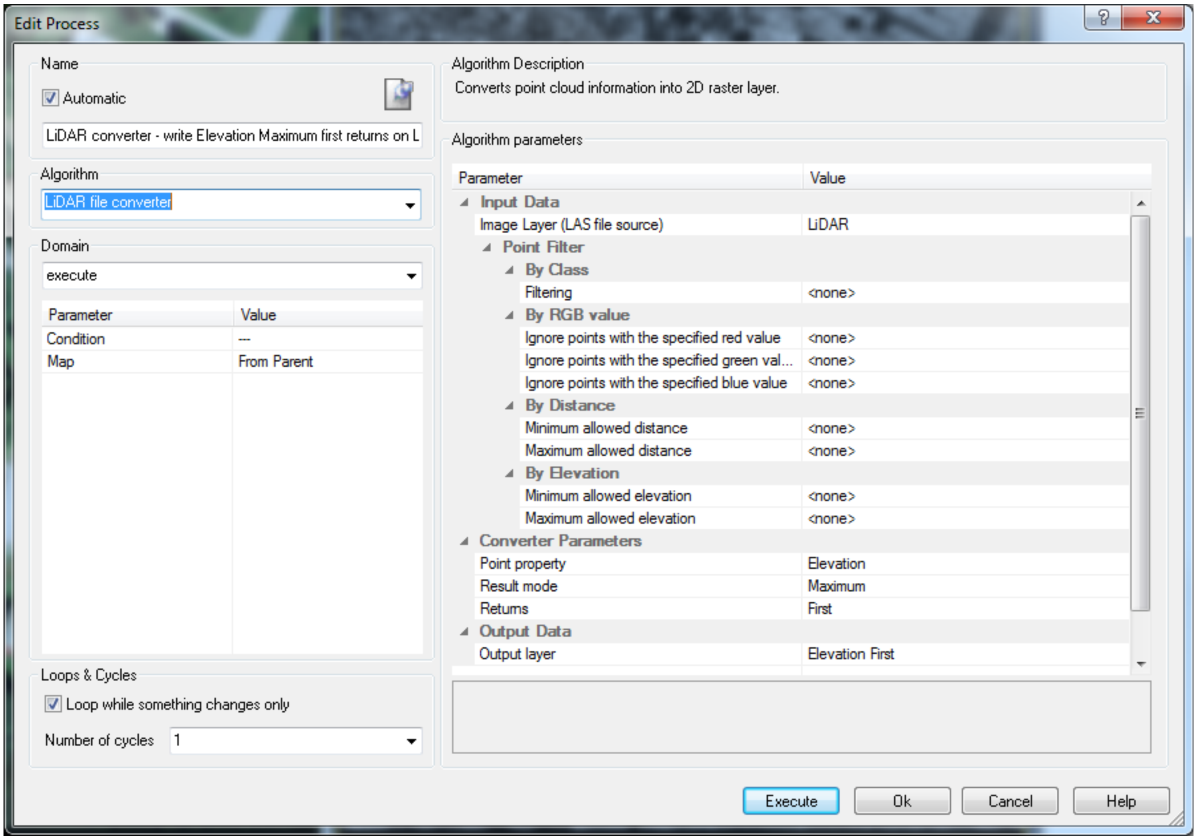
2.2.1 Create the image layer

1. In the Process Tree **expand** the parent processes ‘Working with LiDAR’, ‘Convert and correct’ and ‘**Convert LiDAR to new image layers**’.
2. Double-click on the process ‘**LiDAR Converter - write Elevation Maximum first returns on LiDAR)**’ to open it.


Process Tree - (Working with LiDAR las files_v1, v.1*)

- Working with LiDAR
  - set rule set options
  - Convert and correct
    - Convert LiDAR to new image layers
      - LiDAR converter - write Elevation Maximum first returns on LiDAR
      - LiDAR converter - write Number of returns Maximum all returns on LiDAR
    - Interpolating LiDAR Number of Returns
    - Interpolating LiDAR Elevation First

Figure 11: Process Tree with process highlighted to convert to ‘Elevation First Return’.



Name

☑ Automatic

LiDAR converter - write Elevation Maximum first returns on L

Algorithm: LiDAR file converter

Domain: execute

Parameter	Value
Condition	---
Map	From Parent

Loops & Cycles

☑ Loop while something changes only

Number of cycles: 1

Algorithm Description

Converts point cloud information into 2D raster layer.

Algorithm parameters

Parameter	Value
Input Data	
Image Layer (LAS file source)	LiDAR
Point Filter	
By Class	
Filtering	<none>
By RGB value	
Ignore points with the specified red value	<none>
Ignore points with the specified green val...	<none>
Ignore points with the specified blue value	<none>
By Distance	
Minimum allowed distance	<none>
Maximum allowed distance	<none>
By Elevation	
Minimum allowed elevation	<none>
Maximum allowed elevation	<none>
Converter Parameters	
Point property	Elevation
Result mode	Maximum
Returns	First
Output Data	
Output layer	Elevation First

Figure 12: Process settings to convert the LiDAR to 'Elevation First' layer.

The selected algorithm 'LiDAR File Converter' can be found in the 'LiDAR tools' section of the algorithms list.

The algorithm uses the following parameters:

- In the field '**Image Layer (LAS source file)**', 'LiDAR' is chosen from the list and defined as the sources of the .las file.
- In the field '**Convert Parameter**' the mode 'Elevation' is chosen from the drop down list.
- In the field '**Returns**', 'First' is chosen from the drop-down list.
- In the field '**Calculation**', the calculation mode 'Maximum' is chosen.
- In the field '**Output Layer**', the name 'Elevation First' 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 new created '**Elevation First**' layer.
5. Hover with the mouse over the new layer 'Elevation First' and evaluate the different values of this layer.

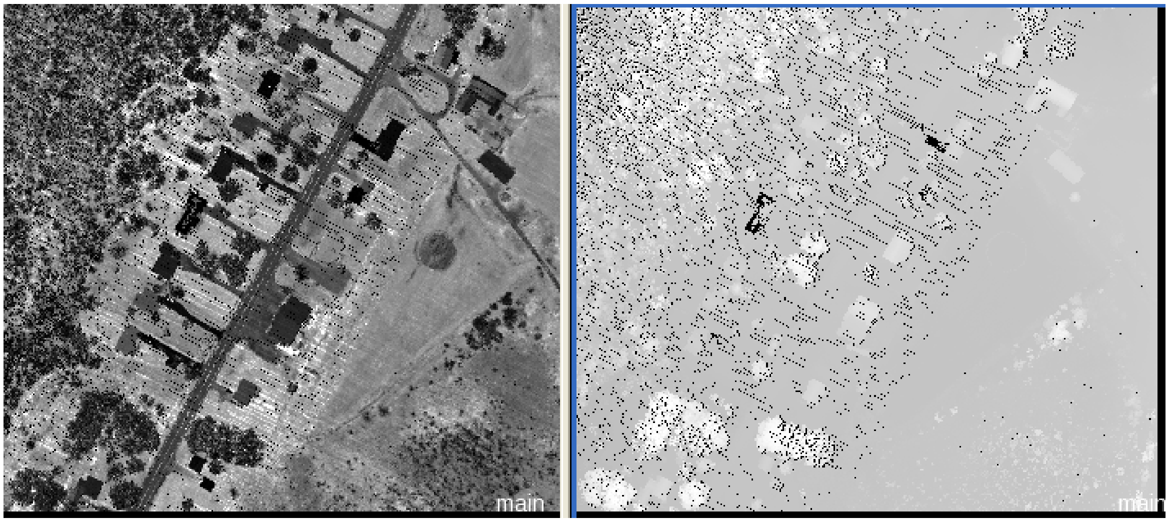


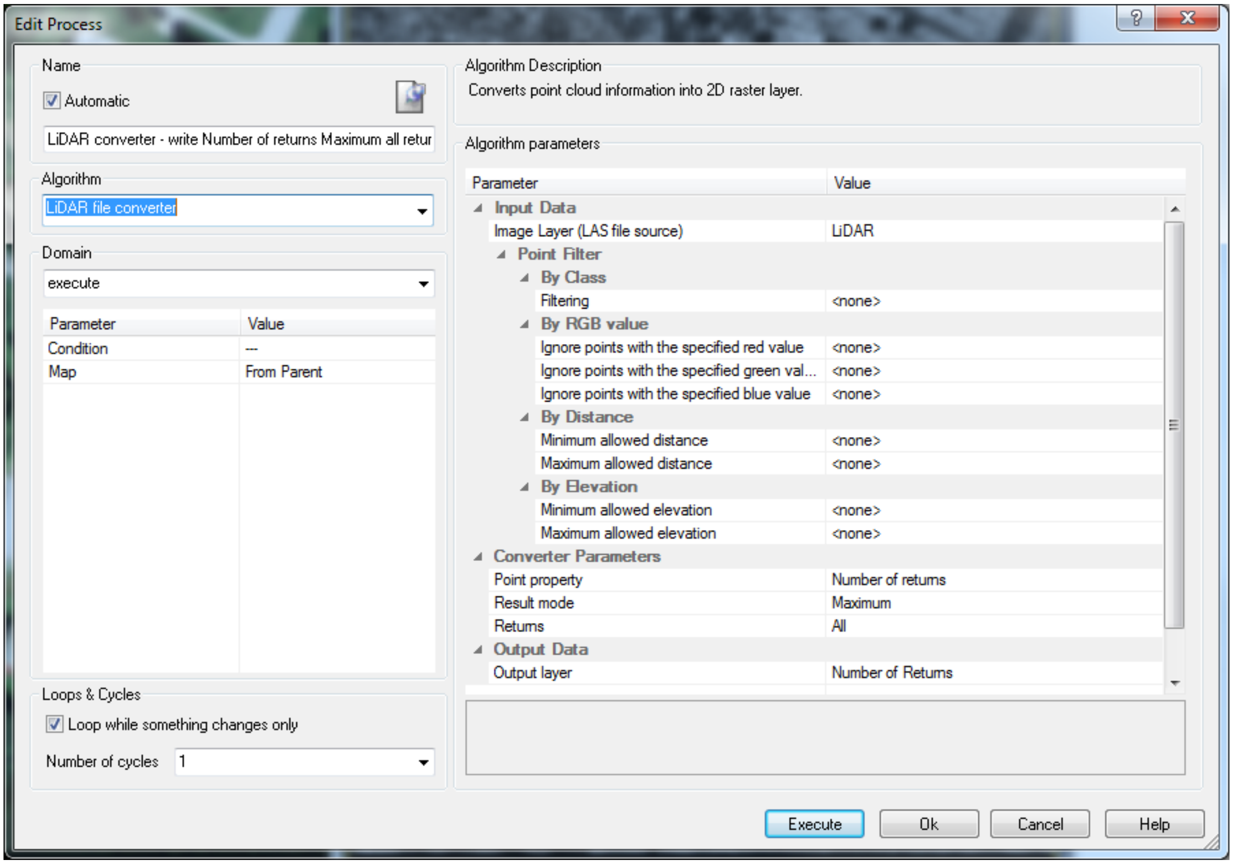
Figure 13: Initial LiDAR data (left); converted image layer 'Elevation First' (right).

2.3 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.

2.3.1 The process settings

1. Double-click on the process '**LiDAR Converter - write Number of returns Maximum all returns on LiDAR**' to open it.



Name

☑ Automatic

LiDAR converter - write Number of returns Maximum all retur

Algorithm

LiDAR file converter

Domain

execute

Parameter	Value
Condition	---
Map	From Parent

Loops & Cycles

☑ Loop while something changes only

Number of cycles 1

Algorithm Description

Converts point cloud information into 2D raster layer.

Algorithm parameters

Parameter	Value
Input Data	
Image Layer (LAS file source)	LiDAR
Point Filter	
By Class	
Filtering	<none>
By RGB value	
Ignore points with the specified red value	<none>
Ignore points with the specified green val...	<none>
Ignore points with the specified blue value	<none>
By Distance	
Minimum allowed distance	<none>
Maximum allowed distance	<none>
By Elevation	
Minimum allowed elevation	<none>
Maximum allowed elevation	<none>
Converter Parameters	
Point property	Number of returns
Result mode	Maximum
Returns	All
Output Data	
Output layer	Number of Returns

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

The algorithm uses the following parameters:

- In the field '**Image Layer (LAS source file)**', again 'LiDAR' is chosen from the list and defined as the sources of the .las file.
- In the field '**Convert Parameter**' the mode 'Number of Returns' is chosen from the drop down list.
- In the field '**Returns**', 'All' is chosen from the drop-down list.
- In the field '**Calculation**' the calculation mode 'Maximum' is chosen.
- In the field '**Output Layer**' the name 'Number of Returns' is defined. This will be the name for the new created image layer.

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 new 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.

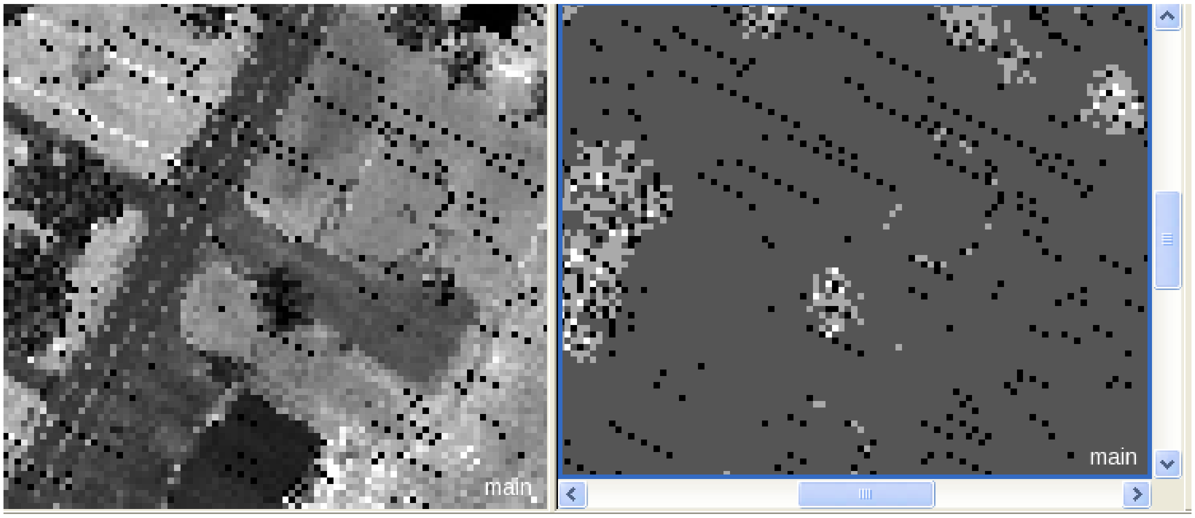


Figure 15: Initial LiDAR data (left); converted image layer 'Number of Returns' (right).

Lesson 3 – Interpolating LiDAR data based on image objects

3.0 Lesson contents

- Why interpolation?
- Introduction to interpolating LiDAR data based on image objects
- Storing current feature values in an object variable
- Creating new values for those objects without a value
- Creating the new image layer from an object variable
- Creating the second interpolated layer 'Elevation First'

The imported and converted LiDAR images shall be used to help **segment** and **classify** the trees and buildings in this subset.

Trees are expected to have multiple returns and buildings are expected to have a high difference in elevation to the surrounding neighbors.

Before using the image layers they must be **interpolated** because there are pixels with **0 values**, disturbing the object creation and feature calculation.

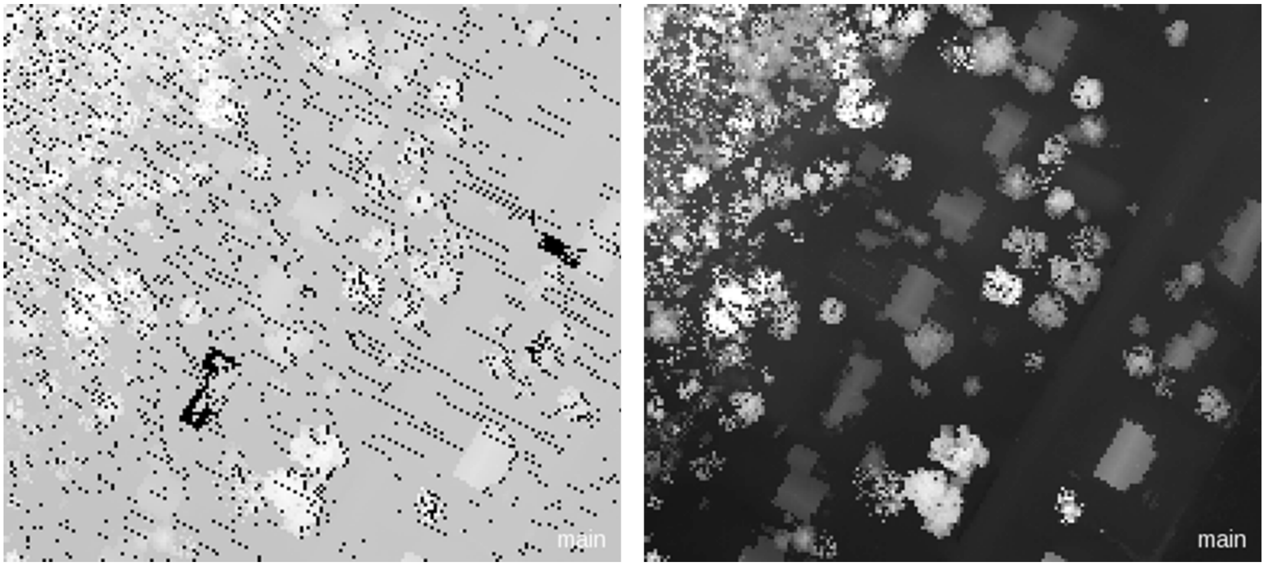


Figure 16: LiDAR first return from elevation as displayed after applying the LiDAR converter algorithm (left); LiDAR first return from elevation as displayed after interpolating (right).

3.1 Why interpolation?

To outline the objects best, both LiDAR image layers are used for segmentation. But to create meaningful objects, the converted layers have to be interpolated. Also if later calculating the mean difference in elevation for buildings, a 0 value object would give back misleading values.

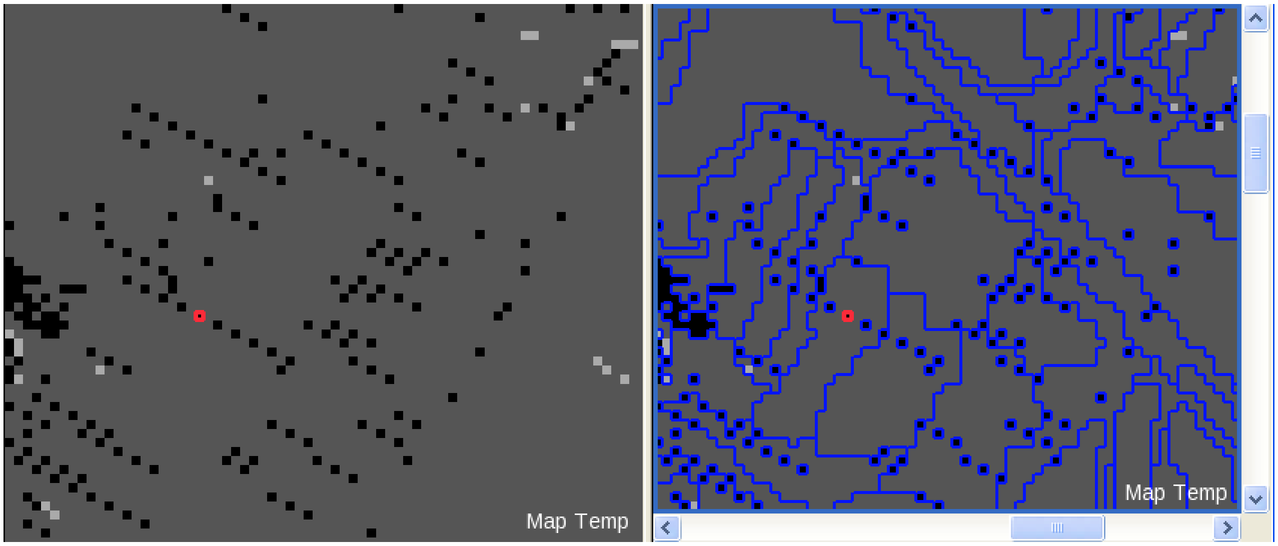


Figure 17: Not interpolated image layer (left); objects created using this layer (right).

3.1.1 Segmentation without interpolating image layers

All optical layers are used for segmentation. The two converted 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.

The processing without interpolated layers is done on a separate map.

1. In the Process Tree **collapse** the parent processes 'Convert and correct'.
2. Expand the parent processes '**CHECK using not interpolated LiDAR**'.
3. Execute the process '**copy map to 'Map Temp''**'.
4. Switch the viewer to the 'Map Temp'.
5. Execute the multiresolution segmentation process
6. Turn on the object outlines and review the objects

Where the image layer have 0 values, small objects are created (as seen in *Figure 17*).

3.1.2 Classify trees without interpolating image layers

The mean value of the image layer 'Number of Returns' is used to classify Trees. In areas where the values of the image layers is 0, no Trees are classified, a lot of holes are in the classification.

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

The classification of 'Trees' has a lot of holes.

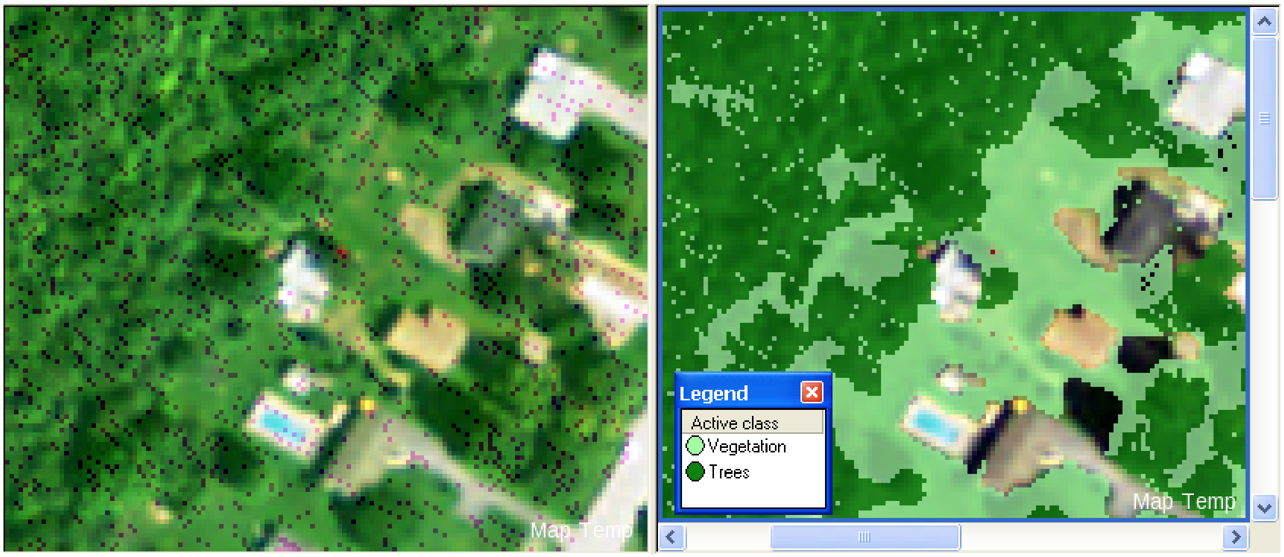


Figure 18: Optical layers and Number of Returns in mixed view (left); classification Result (right).

3.1.3 Classify buildings without interpolating image layers

Objects with difference 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_Mean Elevated First_10> 0.5 and Compactness < 2 at L2: Building'.
2. Switch on the classification view and review the classification.

A lot of objects are misclassified as 'Building'.

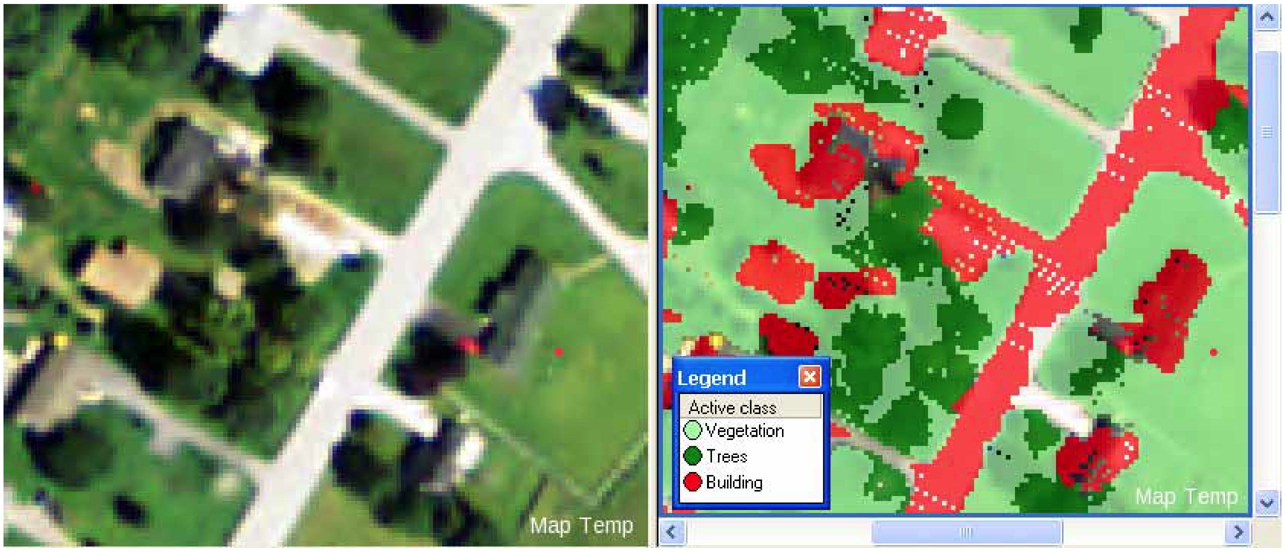


Figure 19: Optical layers (left); classification Result (right).

3.2 Introduction to interpolating LiDAR data based on image objects

To avoid problems in object creation and feature calculation, as explained in the previous chapter, the 0 value pixels will be interpolated in both converted LiDAR image layers. To do so an object oriented approach is used resulting in an interpolated image layer within the Project.

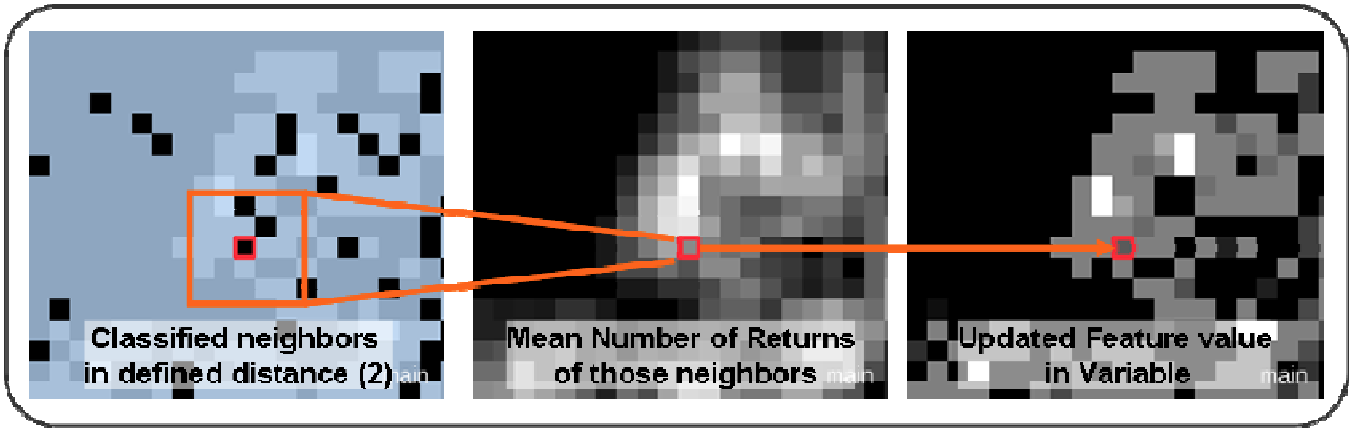


Figure 20: The 'Mean of neighbors' is basis for updating the objects with 0 values.

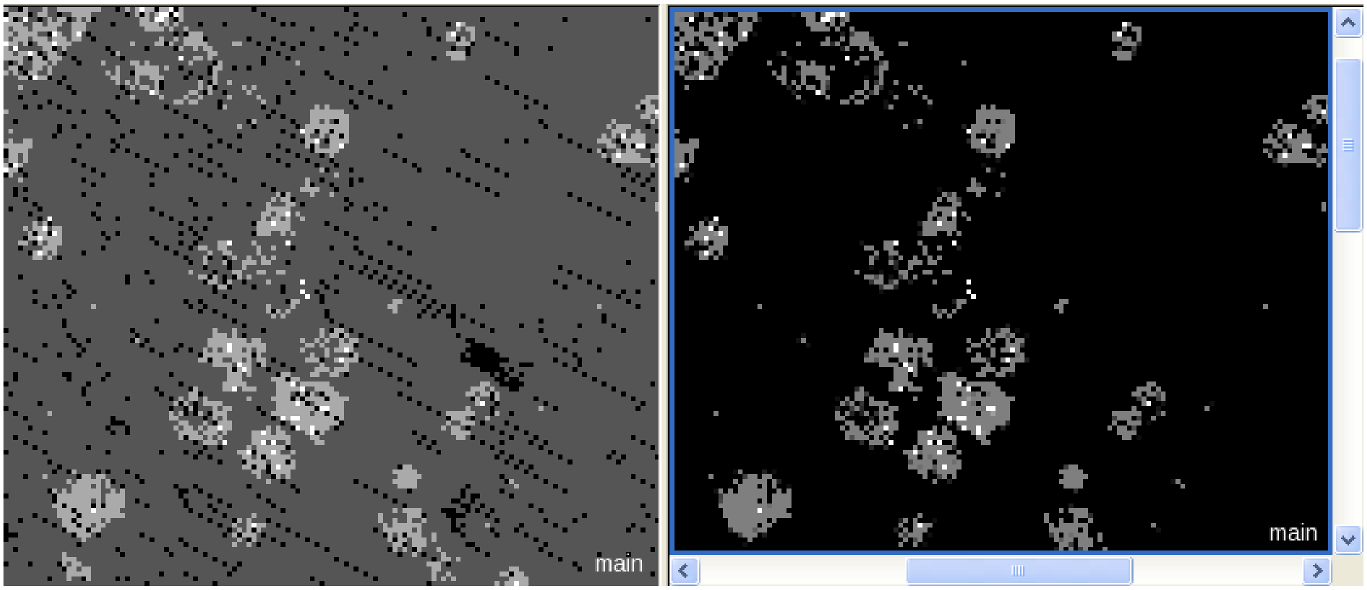


Figure 21: Not interpolated image layer (left); interpolated image layer (right).

3.3 Storing current feature values in an object variable

This chapter covers the following:

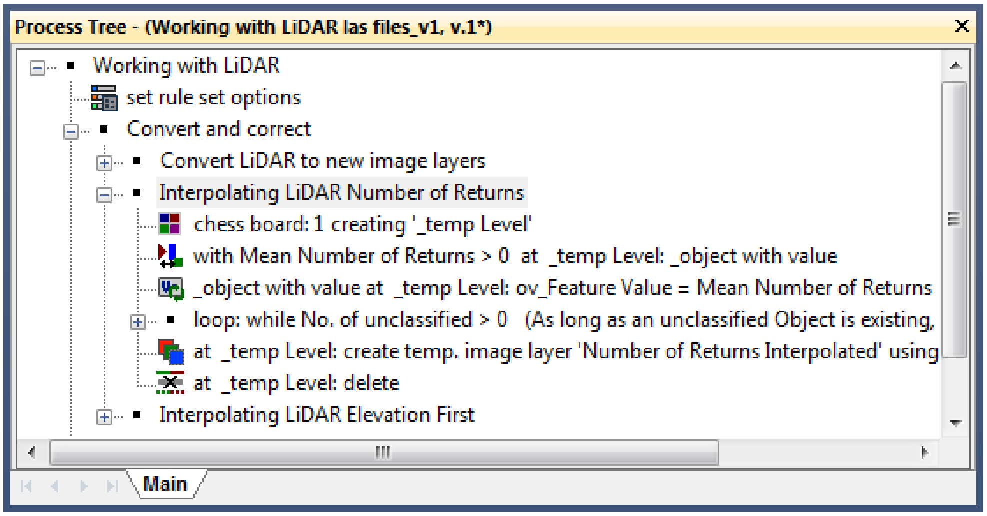
- Classifying objects with a value
- Storing feature values in an object variable

3.3.1 Classifying objects with a value

The first step of this process sequence is to classify those objects which have a value for the 'Number of Returns' image layer. Therefore, a pixel-sized level is created and those objects are classified. Later a customized class-related feature points to these classified objects.

1. Switch in the viewer to the **main map**.

2. Collapse the parent process ‘CHECK using not interpolated LiDAR’.
3. Expand the process ‘Convert and correct’ and ‘**Interpolating LiDAR Number of Returns**’.



Process Tree - (Working with LiDAR las files_v1, v.1*)

```
⊟ ▪ Working with LiDAR
    set rule set options
  ⊟ ▪ Convert and correct
    ⊞ ▪ Convert LiDAR to new image layers
    ⊟ ▪ Interpolating LiDAR Number of Returns
          chess board: 1 creating '_temp Level'
          with Mean Number of Returns > 0  at  _temp Level: _object with value
          _object with value at  _temp Level: ov_Feature Value = Mean Number of Returns
      ⊞ ▪ loop: while No. of unclassified > 0  (As long as an unclassified Object is existing,
          at  _temp Level: create temp. image layer 'Number of Returns Interpolated' using
          at  _temp Level: delete
    ⊞ ▪ Interpolating LiDAR Elevation First
```

Figure 22: Process Tree with processes highlighted to interpolate ‘Number of Returns’.

4. Execute the process ‘chessboard: 1 creating '_temp Level'’.
5. Execute the process ‘with Mean Number of Returns > 0 at _temp Level: _object with value’.

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

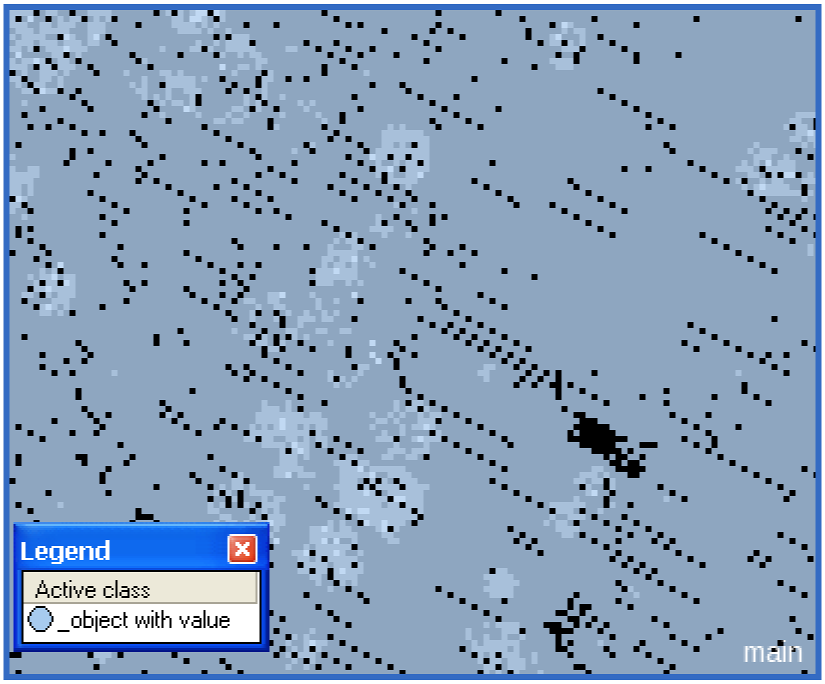
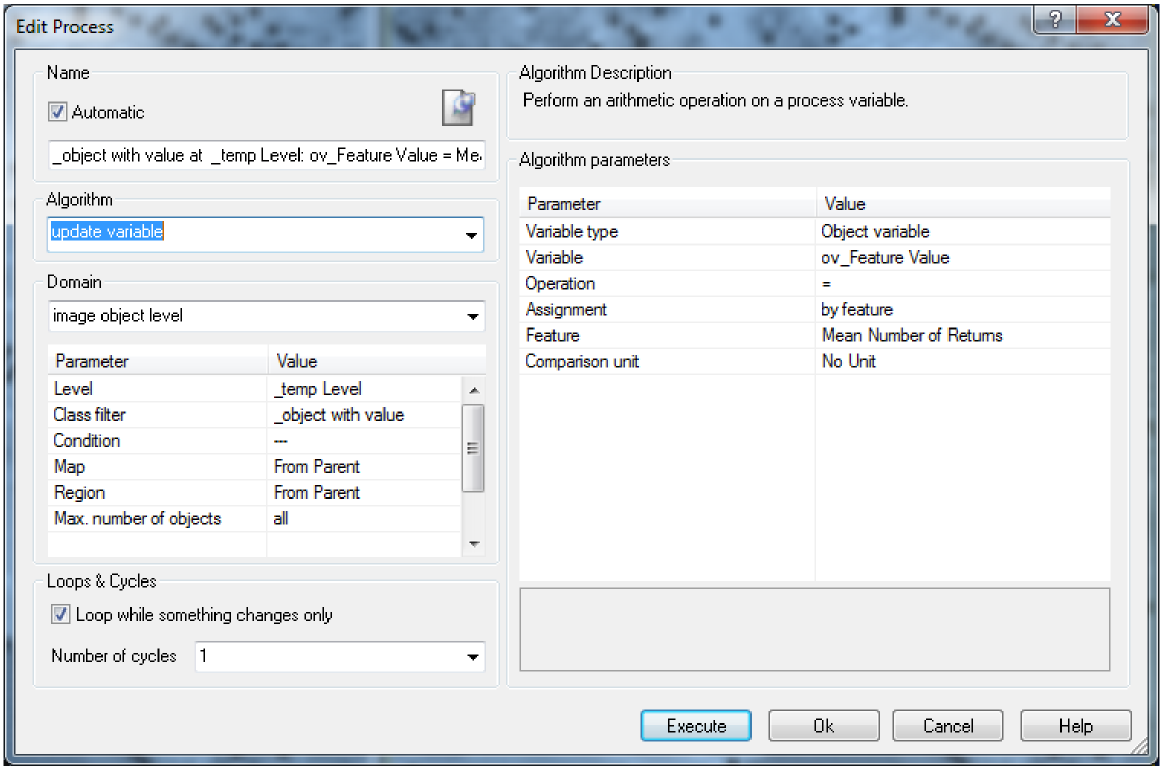


Figure 23: Classification View.

3.3.2 Storing feature values in an object variable

For all objects which have values, the mean values for the ‘Number of Returns’ will be written in an object variable. This object variable is later used to do the interpolation of those objects with no values for the ‘Number of Returns’ layer.

1. Double-click on the process '**_object with value at _temp Level: ov_Feature Value = Mean Number of Returns**'.
   - 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 Number of Returns**' is defined.



Name

☑ Automatic

_object with value at _temp Level: ov_Feature Value = Me

Algorithm

update variable

Domain

image object level

Parameter	Value
Level	_temp Level
Class filter	_object with value
Condition	---
Map	From Parent
Region	From Parent
Max. number of objects	all

Loops & Cycles

☑ Loop while something changes only

Number of cycles 1

Algorithm Description

Perform an arithmetic operation on a process variable.

Algorithm parameters

Parameter	Value
Variable type	Object variable
Variable	ov_Feature Value
Operation	=
Assignment	by feature
Feature	Mean Number of Returns
Comparison unit	No Unit

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

2. Execute the process.
3. Go to the 'Feature View' window and browse to '**Object Features > Variables**'
4. Double-click on '**ov_Feature Value**'.

The object variable and the feature 'Mean Elevation First' currently have the same values.

3.4 Creating new values for objects without a value

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

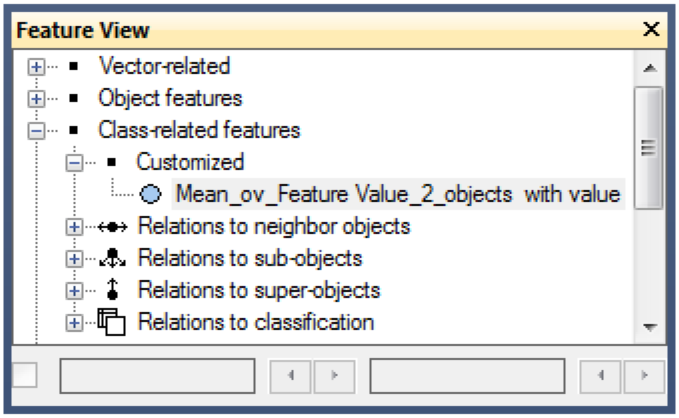
Within the loop, for all unclassified objects, the value of a **Customized Relational** feature is written in the object variable. The features calculates the mean of the neighboring objects. This value is written in the object variable of the unclassified objects.

The unclassified objects which now have values for the object variable are then classified as '_object with value' too.

3.4.1 The customized relational feature to interpolate

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 value.

1. Go to the 'Feature View' window and browse to '**Class-Related Features > Customized**'.
2. Double-click on '**ov_Feature Value**'.

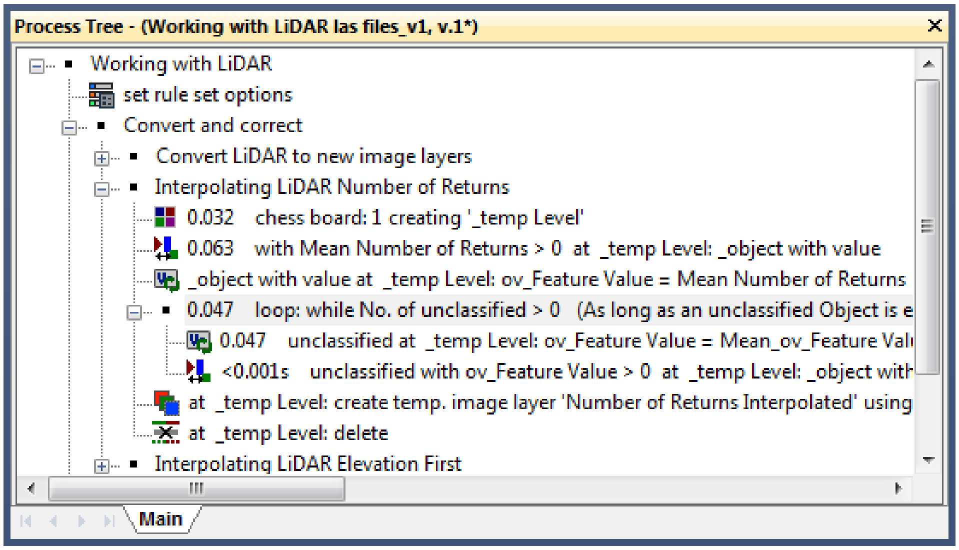


```
Feature View
  Vector-related
  Object features
  Class-related features
    Customized
      Mean_ov_Feature Value_2_objects  with value
    Relations to neighbor objects
    Relations to sub-objects
    Relations to super-objects
    Relations to classification
```

Figure 25: The customized feature calculates the mean from the mean values of the neighbors.

3.4.2 Execute the interpolation

1. Expand the process sequence '**loop: while No. of unclassified > 0**'.
2. Select the process '**unclassified at _temp Level: ov_Feature Value = Mean_ov_Feature Value_2_objects with value**' and execute it.
3. Go to the 'Feature View' window and browse to '**Object Features > Variables**'
4. Double-click on '**ov_Feature Value**'.



```
Process Tree - (Working with LiDAR las files_v1, v.1*)
  Working with LiDAR
    set rule set options
    Convert and correct
      Convert LiDAR to new image layers
      Interpolating LiDAR Number of Returns
        0.032   chess board: 1 creating '_temp Level'
        0.063   with Mean Number of Returns > 0  at  _temp Level:  _object with value
        _object with value at  _temp Level: ov_Feature Value = Mean Number of Returns
        0.047   loop: while No. of unclassified > 0   (As long as an unclassified Object is e
          0.047   unclassified at  _temp Level: ov_Feature Value = Mean_ov_Feature Val
          <0.001s   unclassified with ov_Feature Value > 0  at  _temp Level: _object with
        at  _temp Level: create temp. image layer 'Number of Returns Interpolated' using
        at  _temp Level: delete
      Interpolating LiDAR Elevation First
Main
```

Figure 26: Process Tree with processes highlighted to interpolate.

Objects which formerly had 0 values are updated and contain now the mean of the mean values of its neighbors.

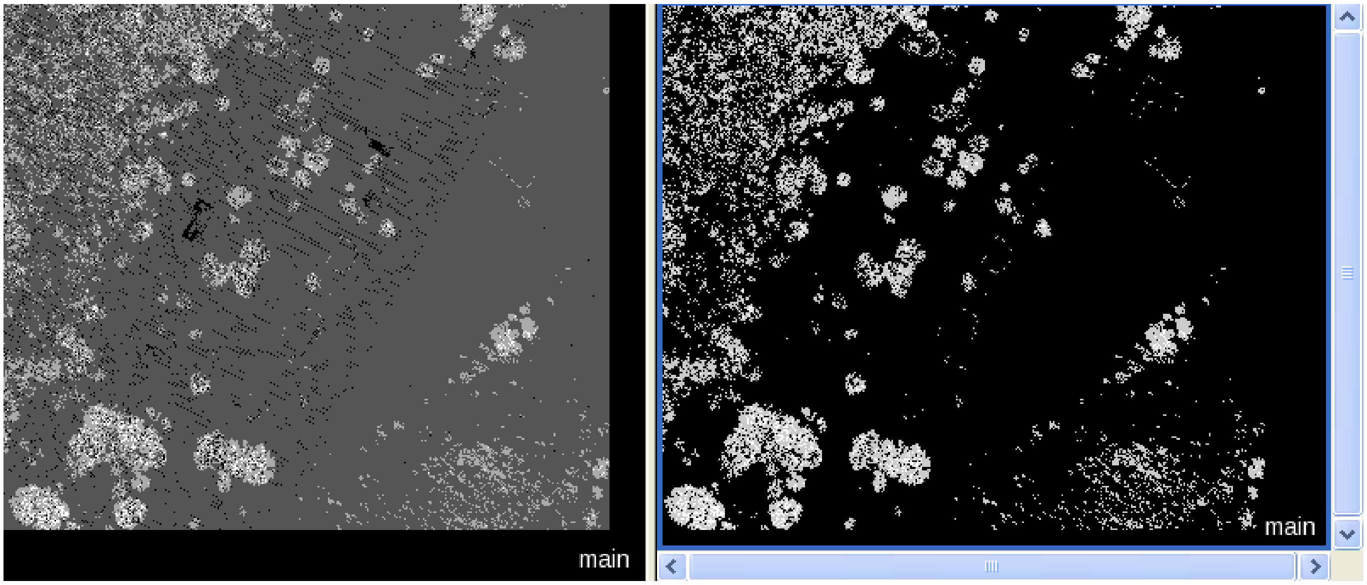


Figure 27: Mean of 'Number of Returns' (left); updated object variable (right).

5. Select and execute the process '**unclassified with ov_Feature Value > 0 at _temp Level: _object with value**'.
6. Select and execute the parent process '**loop: while No. of unclassified > 0**'.

All objects have now interpolated values for 'Number of Returns'.

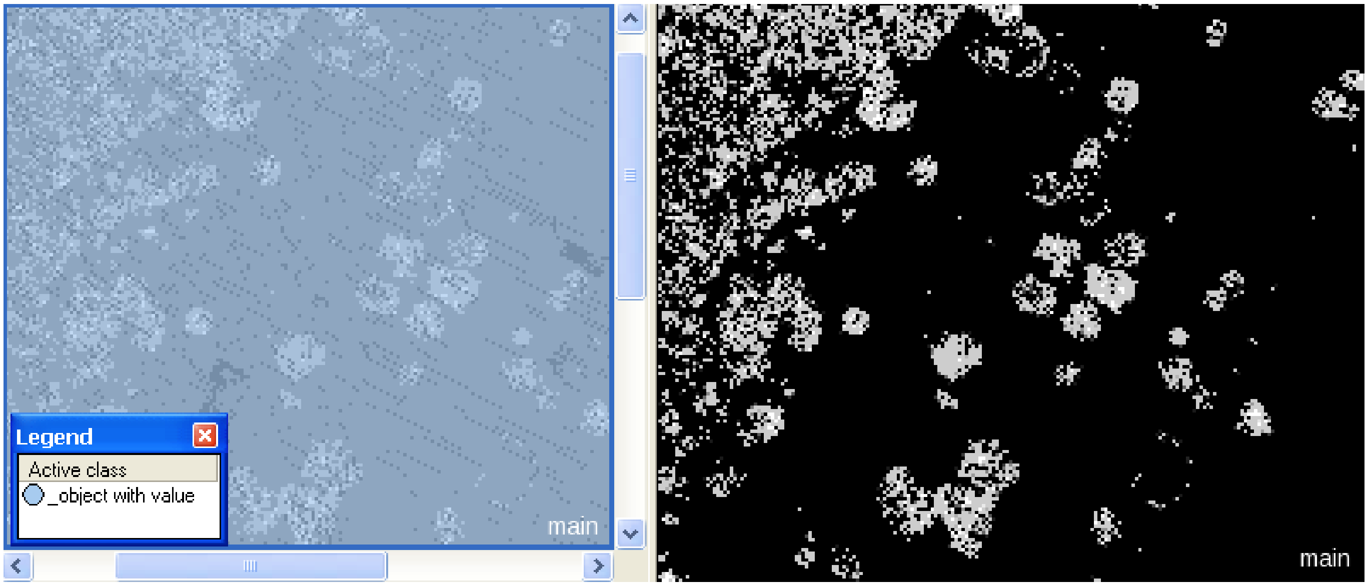


Figure 28: Classification view' (left); updated object variable after executing the loop (right).

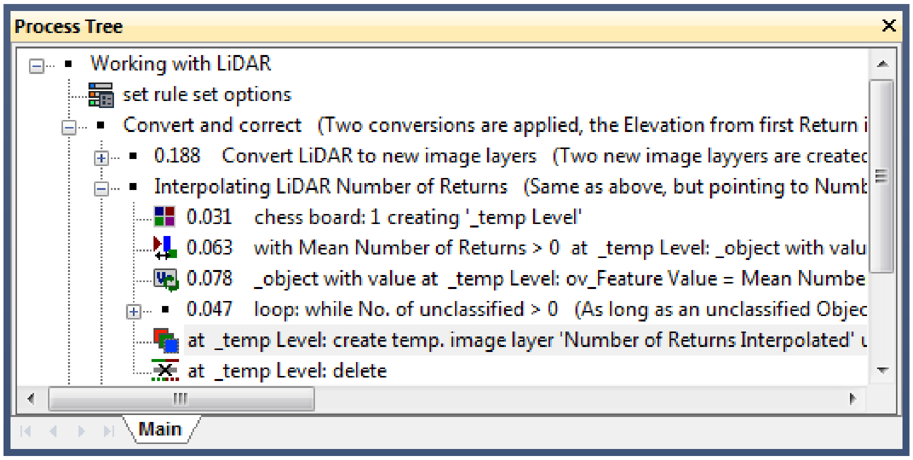
3.5 Creating a new image layer from an object variable

A new image layer 'Number of Returns 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 'Number of Returns' layer.

3.5.1 The process settings

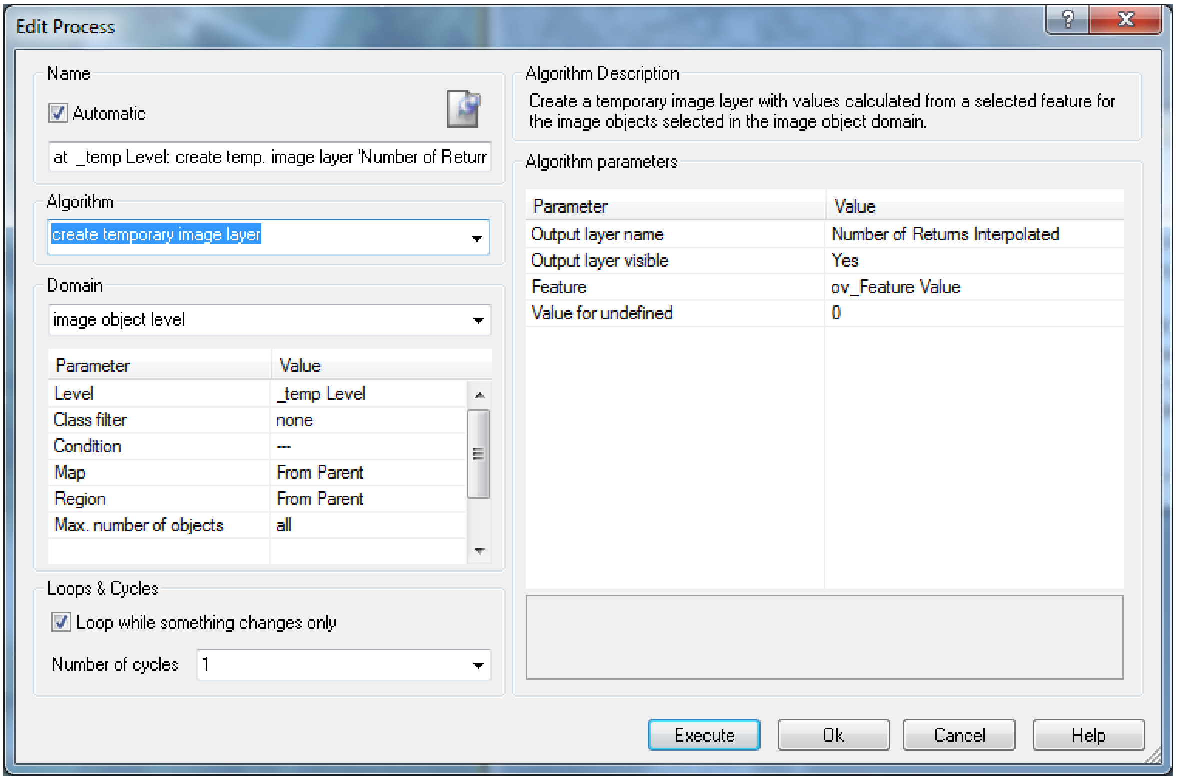
The algorithm ‘create temporary image layer’ is chosen. With this algorithm, you can create an image layer containing feature values. It can be found in the ‘Image layer operation’ section of the algorithms list.

1. Collapse the process sequence ‘**loop: while No. of unclassified > 0**’.
2. Double-click on the process ‘**at _temp Level:create temp. Image layer ‘Number of Returns Interpolated...**’.
    - The Domain points to all objects of ‘**temp_Level**’
    - In the field ‘Layer name’ the name ‘**Number of Return Interpolated**’ is defined.
    - In the field ‘Feature’ the ‘**ov_Feature Value**’ is defined.



```
Working with LiDAR
    set rule set options
    Convert and correct   (Two conversions are applied, the Elevation from first Return i
        0.188   Convert LiDAR to new image layers   (Two new image layyers are createc
        Interpolating LiDAR Number of Returns   (Same as above, but pointing to Numb
            0.031   chess board: 1 creating '_temp Level'
            0.063   with Mean Number of Returns > 0  at  _temp Level: _object with valu
            0.078   _object with value at  _temp Level: ov_Feature Value = Mean Numbe
            0.047   loop: while No. of unclassified > 0   (As long as an unclassified Objec
            at _temp Level: create temp. image layer 'Number of Returns Interpolated' u
            at _temp Level: delete
```

Figure 29: Process Tree with process highlighted to create a temporary layer.



Name

☑ Automatic

at _temp Level: create temp. image layer 'Number of Returr

Algorithm

create temporary image layer

Domain

image object level

Parameter	Value
Level	_temp Level
Class filter	none
Condition	---
Map	From Parent
Region	From Parent
Max. number of objects	all

Loops & Cycles

☑ Loop while something changes only

Number of cycles 1

Algorithm Description

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

Algorithm parameters

Parameter	Value
Output layer name	Number of Returns Interpolated
Output layer visible	Yes
Feature	ov_Feature Value
Value for undefined	0

Figure 30: Process settings to create the temporary layer ‘Number of Returns Interpolated’.

3. Execute the process ‘**at _temp Level:create temp. Image layer ‘Number of Returns Interpolated...**’.

The new, interpolated image layer is created.

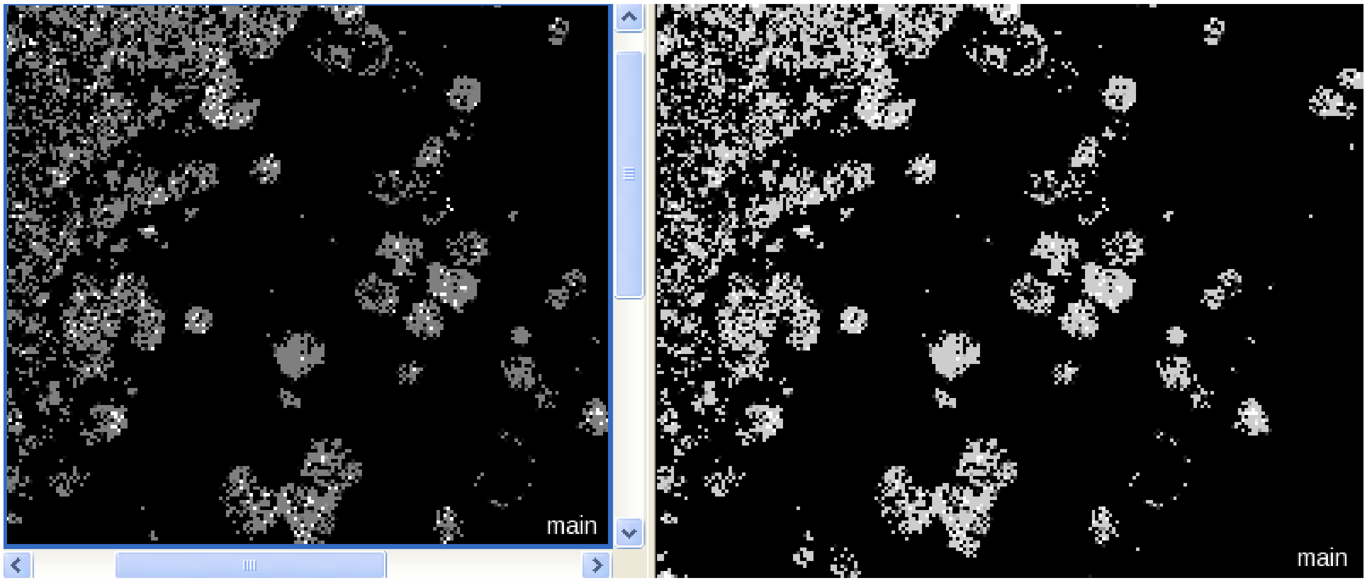


Figure 31: Feature View for ‘ov_Feature Value (left); new created layer (right).

3.5.2 Deleting the object level

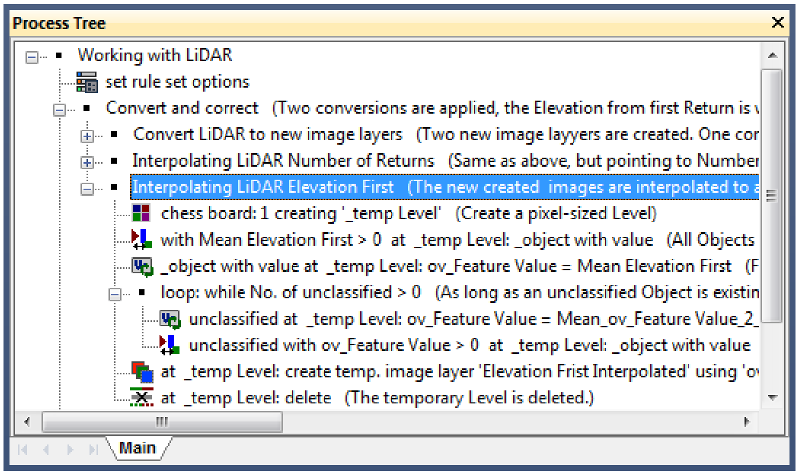
To be prepared for the next interpolation the existing image object level is deleted.

1. Execute the process ‘**at _temp Level: delete**’.

3.6 Creating the second interpolated layer for ‘Elevation First’

The same problems with 0 values also exist for the ‘Elevation First’ layer. A similar interpolation process sequence is applied to create an ‘Elevation First Interpolated’ layer.

1. Collapse the process sequence ‘**Interpolating LiDAR Number of Returns**’.
2. Expand the process sequence ‘**Interpolating LiDAR Elevation First**’.
3. Execute the parent process ‘**Interpolating LiDAR Elevation First**’.



Process Tree

```
Working with LiDAR
    set rule set options
    Convert and correct   (Two conversions are applied, the Elevation from first Return is v
        Convert LiDAR to new image layers   (Two new image layyers are created. One cor
        Interpolating LiDAR Number of Returns   (Same as above, but pointing to Number
        Interpolating LiDAR Elevation First   (The new created  images are interpolated to a
            chess board: 1 creating '_temp Level'   (Create a pixel-sized Level)
            with Mean Elevation First > 0  at  _temp Level: _object with value   (All Objects
            _object with value at  _temp Level: ov_Feature Value = Mean Elevation First   (F
            loop: while No. of unclassified > 0   (As long as an unclassified Object is existin
                unclassified at  _temp Level: ov_Feature Value = Mean_ov_Feature Value_2_
                unclassified with ov_Feature Value > 0  at  _temp Level: _object with value
            at  _temp Level: create temp. image layer 'Elevation Frist Interpolated' using 'ov
            at  _temp Level: delete   (The temporary Level is deleted.)
```

Figure 32: Process Tree with processes highlighted to interpolate the 'Elevation First'.

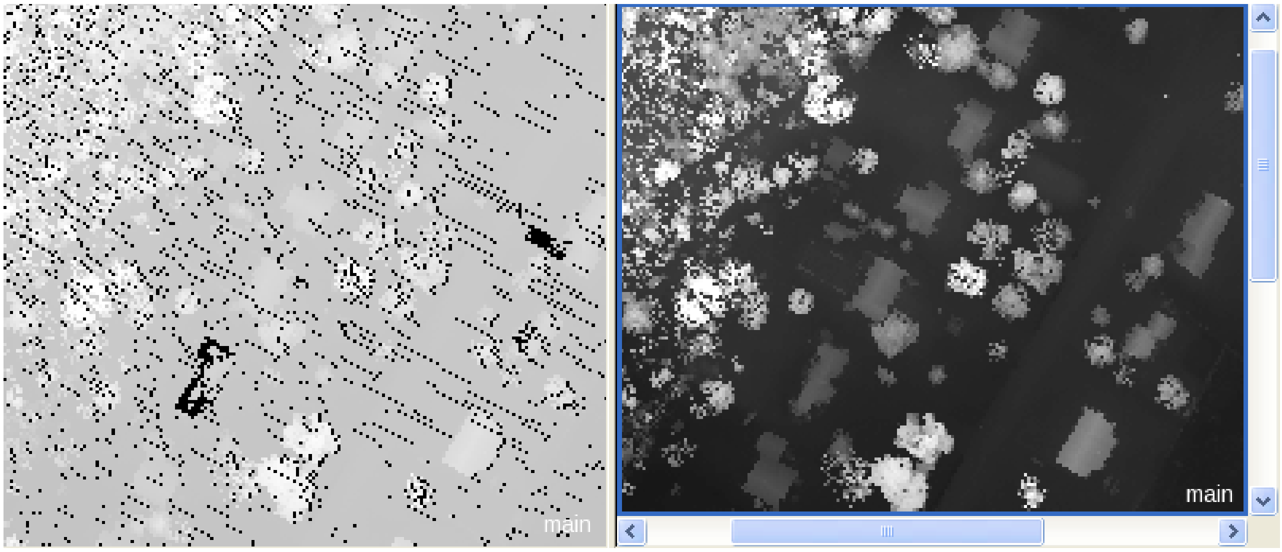


Figure 33: Original, converted image layer (left); new created, interpolated layer (right).

Lesson 4 – Segmenting and classifying with interpolated image layers

4.0 Lesson content

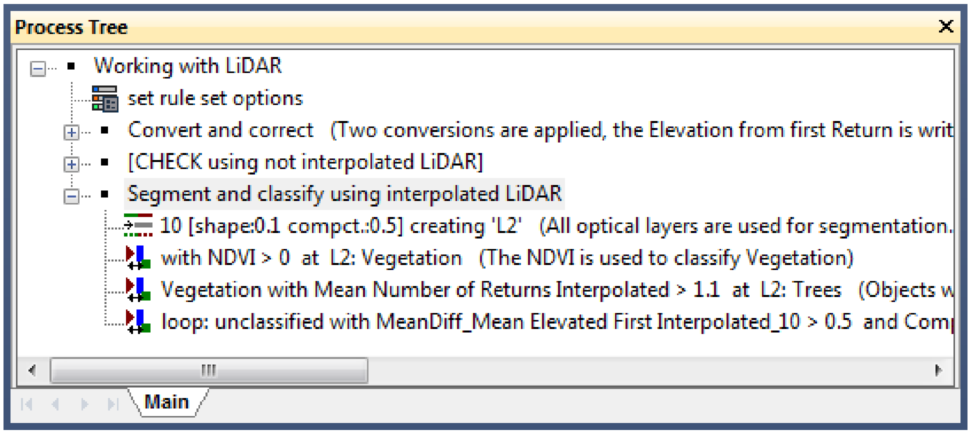
- Segment with interpolated image layers
- Classify with interpolated image layers
- Classify buildings with interpolated image layers

4.1 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**’.



- Working with LiDAR
  - set rule set options
  - Convert and correct   (Two conversions are applied, the Elevation from first Return is writ
  - [CHECK using not interpolated LiDAR]
  - Segment and classify using interpolated LiDAR
    - 10 [shape:0.1 compct.:0.5] creating 'L2'   (All optical layers are used for segmentation.
    - with NDVI > 0 at L2: Vegetation   (The NDVI is used to classify Vegetation)
    - Vegetation with Mean Number of Returns Interpolated > 1.1 at L2: Trees   (Objects w
    - loop: unclassified with MeanDiff_Mean Elevated First Interpolated_10 > 0.5 and Comp

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

3. Execute the process ‘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.

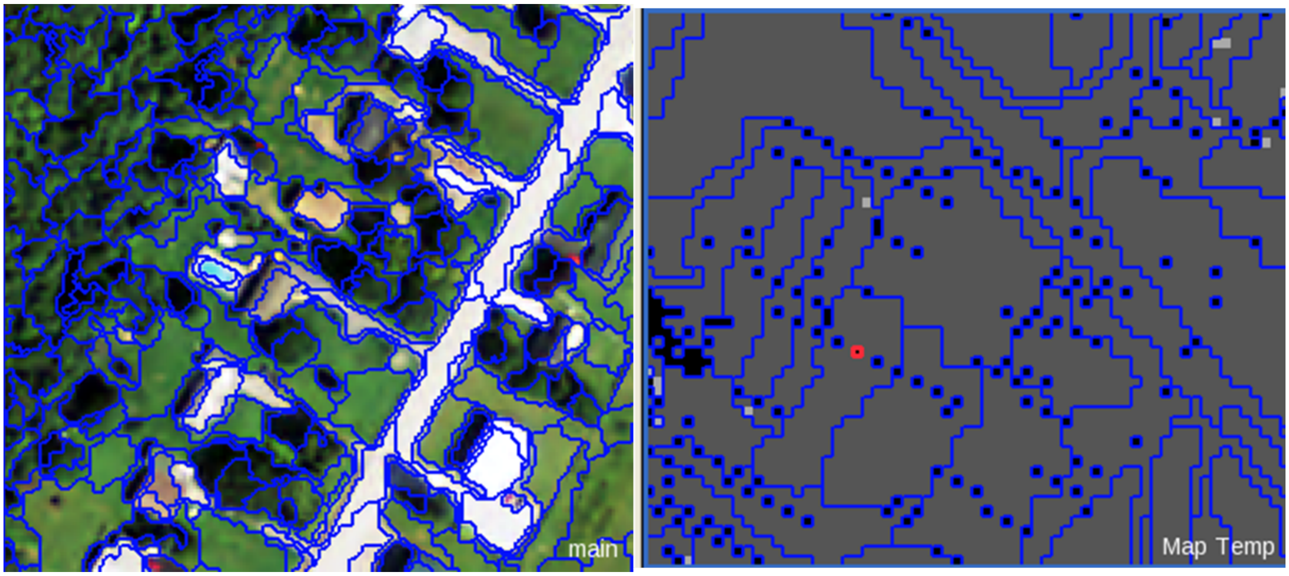


Figure 35: Objects created using the interpolated image layer within the main map (left); objects created using the not interpolated image layer within Map Temp (right).

4.2 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 does not contain holes.

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.

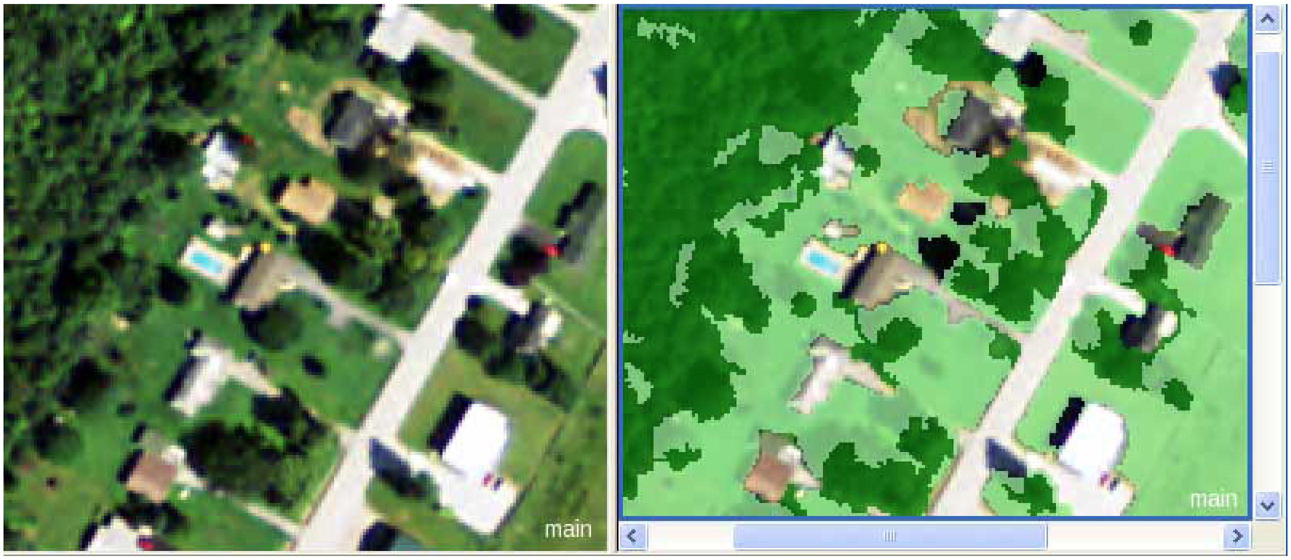


Figure 36: Optical layers (left); classification results using interpolated image layer 'Number of Returns Interpolated' (right).

Compare to results using not interpolated image layer:

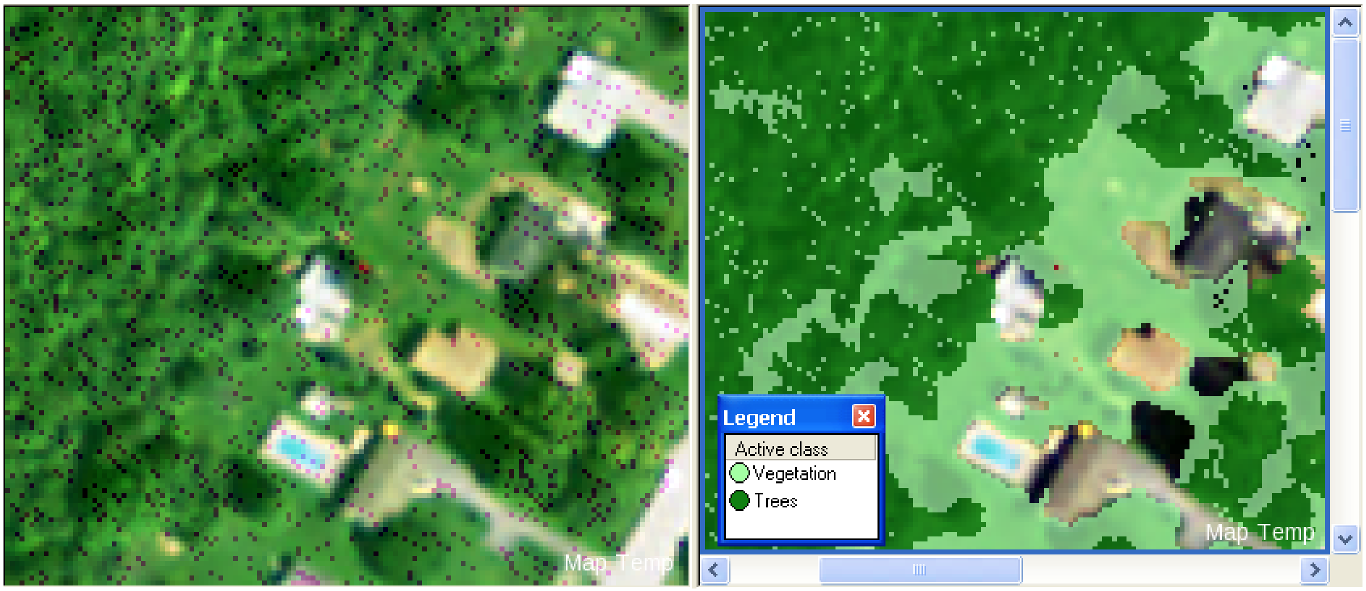


Figure 37: Optical layers and Number of Returns not interpolated in mixed view (left); classification results based on not interpolated data (right).

4.3 Classify buildings with interpolated image layers

Objects with a difference in elevation in comparison to their neighbors are classified as Building.

As the 0 values of the not 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 Elevated First 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.

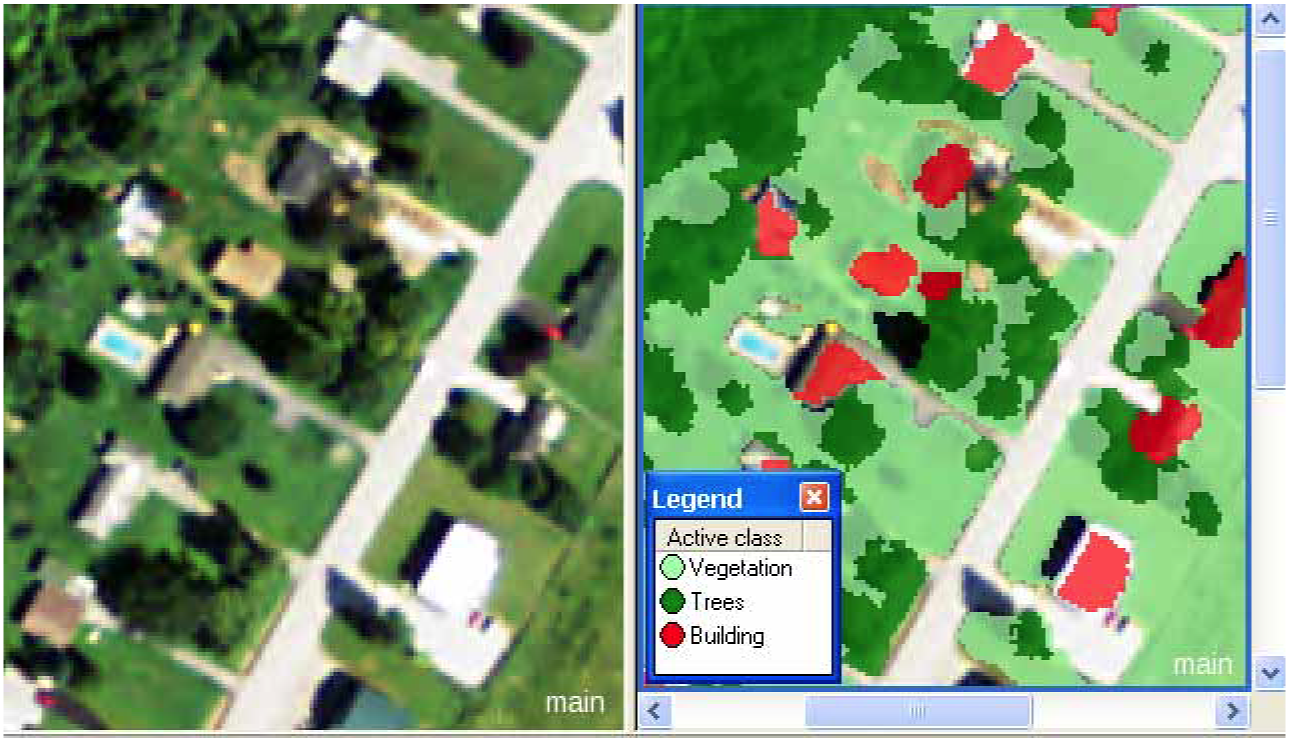


Figure 38: Optical layers (left); classification results using interpolated image layers (right).

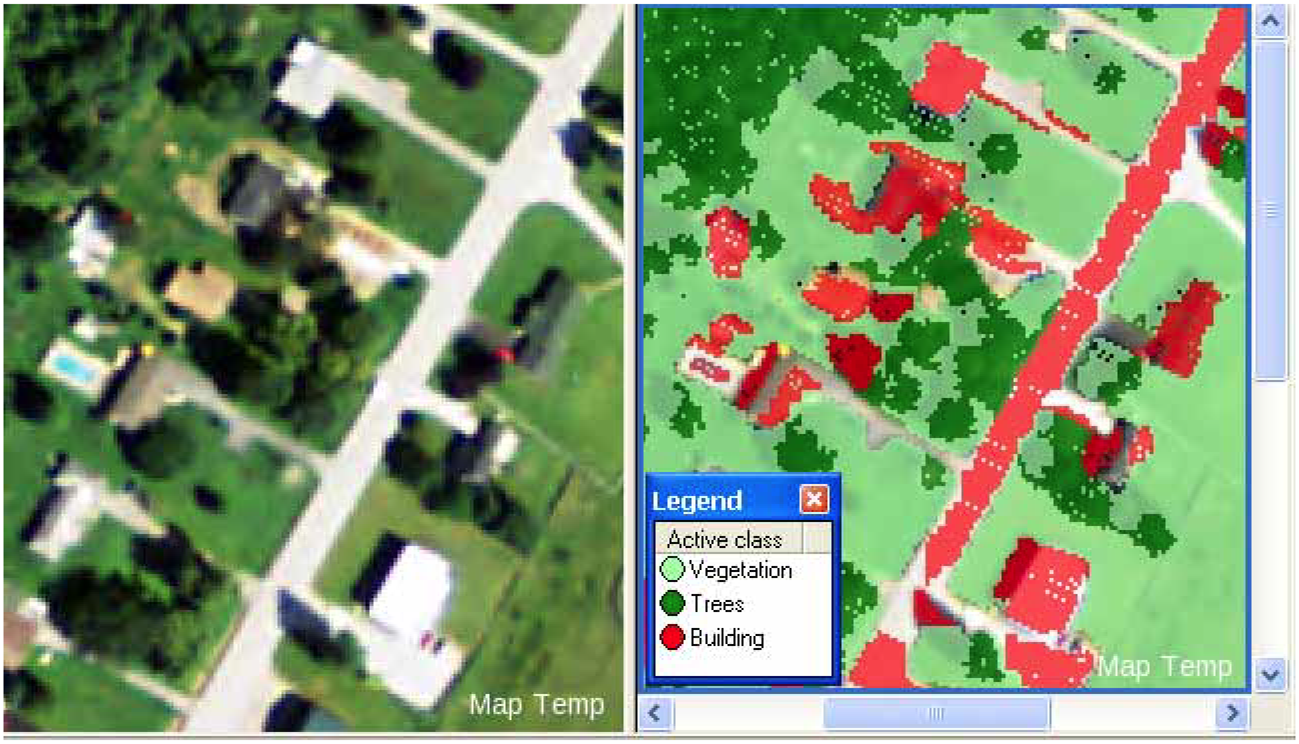


Figure 39: Optical layers (left); classification results using not interpolated image layers (right).

Where to get additional help & information?

The eCognition Community

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

The Community contains content such as:

- **Wiki**: collection of eCognition related articles (e.g. Rule Set tips and tricks, strategies, algorithm documentation...).
- **Discussions**: ask questions and get answers.
- **File exchange**: share any type of eCognition related code such as Rule Sets, Action Libraries, plug-ins...
- **Blogs**: read and write insights about what’s happening around our industry…

Share your knowledge and questions with other users interested in using and developing image intelligence applications for Earth Sciences at:

http://community.ecognition.com/.

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 interface in the main menu ‘Help>eCognition Developer User Guide’ or Reference Book.

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

eCognition Training

eCognition Training Services offer a carefully planned curriculum that provides hands-on, real-world exercises. We are dedicated to enhancing customers’ image analysis skills and helping these organizations to accomplish their 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 image analysis needs, thereby maximizing the training effect.

For more information please see our website or contact us at: eCognition_Training@trimble.com