eCognition Developer

Tutorial 8 — Working with Point Clouds in eCognition

www.trimble.com

TRANSFORMING THE WAY THE WORLD WORKS

Trimble.

Introduction

About this Tutorial

This tutorial applies different rule sets to point cloud data. The lessons show and explain approaches how to classify point cloud data, extract changes between two different dates and how to generate terrain models.

This module has three lessons:

- Lesson 1 - Create DTM, DSM, nDSM and classify ground points in point clouds
- Lesson 2 - General point cloud data classification in ground, vegetation and buildings
- Lesson 3 - Classify differences in multitemporal point cloud data

Requirements

To perform this Tutorial, you will need:

- **eCognition Developer** version 9.3 (or higher) installed on a computer
- A computer **mouse** is highly recommended

All steps of this tutorial can be done using **eCognition Developer.** This tutorial is designed for self-study.

Image data - Rule sets - Projects

We will be working with point cloud data sets. These are included in the tutorial zip file. Please download the files needed (projects and image data) in the eCognition Community: www.ecognition.com/community.

Data specification:

- Location: Cobbs Creek, West Philadelphia, Pennsylvania, United States, 39.951782, -75.234095
- Coordinate System: NAD83 / Pennsylvania South (ftUS) Lambert Conformal Conic 2SP
- Acquisition time: 2008, Change detection 2008 & 2010
- Copyright/legal information: Department of Conservation and Natural Resources, Bureau of Topographic and Geologic Survey, Pennsylvania, USA. PASDA (www.pasda.psu.edu) provides public access to PAMAP orthoimagery, DEMs, contours, and lidar processing/contour enhancement lines as downloadable files and web services.

Data needed for the Tutorials

The tutorial folder contains for lesson 1:

- Point cloud data file **testPointCloud01.las**
- eCognition project **TerrainModelGeneration.dpr**

The tutorial folder contains for lesson 2:

- Point cloud data file **testPointCloud01.las**
- eCognition project **GeneralClassification.dpr**

The tutorial folder contains for lesson 3:

- Two point cloud data files of the same area for two different times (**time01.las** and **time02.las**)
- eCognition project **ChangeDetection.dpr**

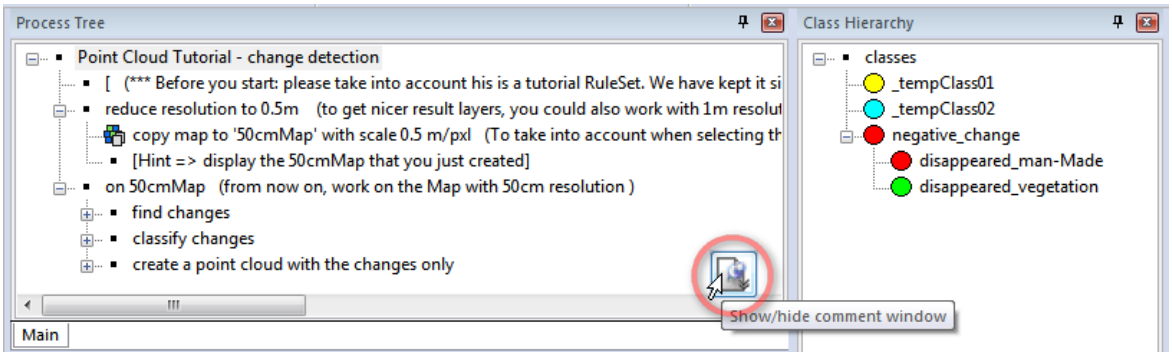


Figure 0.1: *Rule set with pin to visualize comments*

Lesson 1 – Create Terrain models and Classify Ground Points

1.1 Learning objective and contents

This lesson shows how to extract DTM, DSM and nDSM information from point cloud data. The rule set also assigns ground points with classification rules using an approach based on customized object features.

1. First step - Reduce resolution
2. Create DSM - create rasterized layer with highest elevation per pixel cell
3. Create DTM - classify ground objects and interpolate values
4. Create nDSM - create difference layer using layer arithmetics
5. Classify point cloud ground points

Estimated completion time: 30 minutes to 1 hour depending on the users familiarity with the eCognition software.

1.2 Description of Rule Set - Create DTM, DSM, nDSM and classify ground points

Please start **eCognition Developer** and open the project via the menu **File > Open project > TerrainModelGeneration.dpr.**

If not already opened please click on the ***pin*** in the lower right corner of the **process tree window** to visualize comments included in this rule set.

Overview Rule Set

Please execute the following processes now step-by-step, read through the comments included in the rule set and follow the instructions in this tutorial. Deepen your knowledge on how to visualize and handle point cloud data based on the **User Guide > Starting eCognition Developer > Navigating in 3D**.

1.2.1 Reduce resolution

In eCognition, a project including only point clouds is set to a resolution of 1 meter by default. This can be changed during project creation or later based on the rule set by creating a new map with a different resolution. The project has a pixel size of 1m/pixel for the rasterized intensity layer when opening the project. If you check the small drop-down menu **Select active map** before executing the first algorithm of the rule set - only **main** is available.

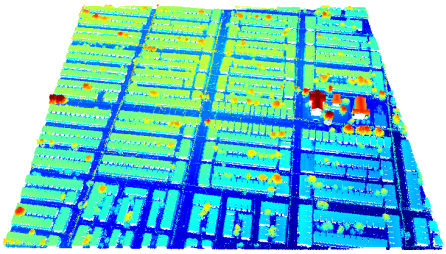


Figure 1.1: *Input point cloud in render mode 'Height'.*

1.2.2 on 50cmMap

If you execute the first process, the resolution is changed to 0,5m/pixel creating a new map called **50cmMap**. From now on, all further algorithms are applied to this resolution and you have to change to this map using the **Select active map** drop-down to see the results of all processes.

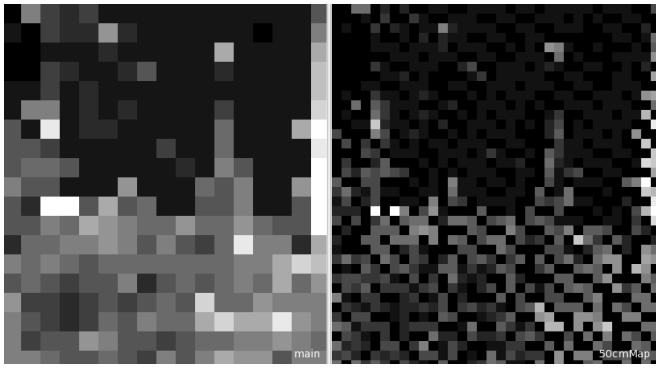


Figure 1.2: *Subset of main view with 1 m resolution (left) and 50cm Map (right)*

Create DSM

In the first set of rules, eCognition's algorithms are used to extract a Digital Surface Model (DSM) based on the point cloud data:

- The algorithm "rasterize point cloud" writes the elevation maximum (z-coordinate) from point cloud, **Layer 1**, to a new raster layer, **'DSM'.** This algorithm produces a raster file where the highest elevation of the point cloud is assigned to the pixel cell. It assigns the maximum elevation value from the point cloud to the pixel value because this represents vegetation and building edges better than the calculation of a mean value.
- Fill gaps - Based on the new raster layers, gaps between pixels have to be filled in the next steps with valid elevation values from the point cloud. The approach is suited to fill *small* gaps only:
    - In a first segmentation the "multi-threshold segmentation" creates a new level based on the **'DSM'**, where all pixels are segmented to a large object and classified as **class '_tempClass01'**

with an elev ≠ 0. All pixels with a value = 0 are assigned to 'unclassified' (Note that there might be data with negative elevations. Because this segmentation does not allow to insert the value < 0 and > 0, a workaround is used for the lower border of the interval set to _tempClass01 <= -0.0001)
    - Now, a “median filter” is applied to this object level to interpolate the gaps using the class _tempClass01, where all unclassified objects are filtered using a kernel size of 3. The resulting values are written into the raster layer **'DSM'** again.
    - Because the object level is not needed anymore it is deleted ('New Level').
- Finally, by applying 2 additional cycles of the “median filter”, outliers - single pixels with extremely high or low elevation - are removed that result for example from power lines or vegetation.

Figure 1.3: *Rasterized DSM (left) with gaps filled (middle) and median filter applied (right) to remove outliers.*

Create DTM

To create a DTM based on the point cloud data, in a first step, ground areas are extracted and in a second step non-ground areas are refined.

Classify ground objects

The ground areas are extracted based on a smart mixture of calculating minimum elevations in the point cloud, segmentation steps resulting in comparable image objects and various filtering techniques.

[Approach 1 - using automatic ground classification - not explained in this tutorial]

Approach 2 - with a customized OBIA-based ground classification

- Classify flat areas
    - Produce raster with lowest elevation per pixel cell - the algorithm “rasterize point cloud” writes the elevation minimum (z-coordinate) from point cloud **Layer 1** to a new raster layer **'DTM'**
    - Fill gaps & Filtering - as described in the section above (Create DSM) the gaps are filled and outliers filtered using the same approach

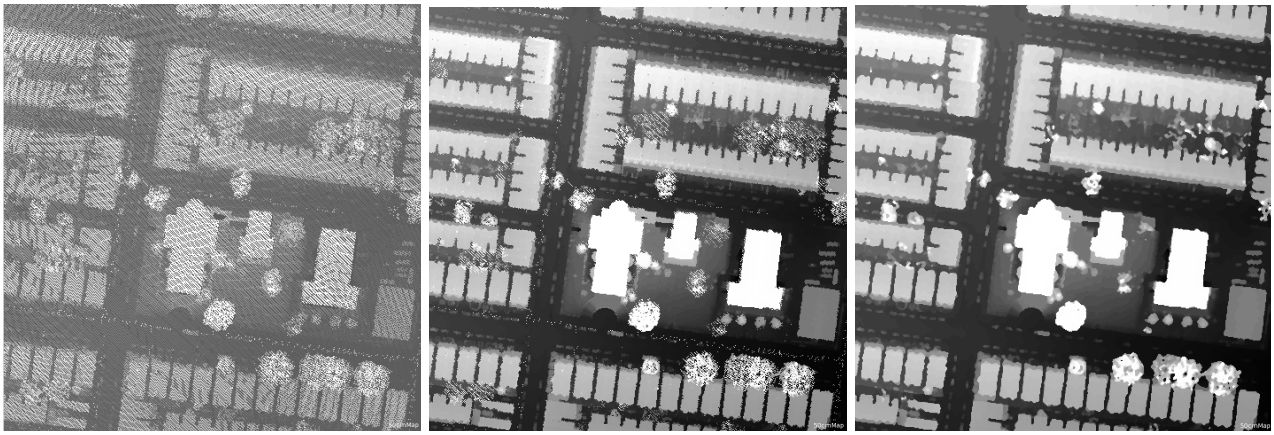


Figure 1.4: *Rasterized DTM (left) with gaps filled (middle) and median filter applied (right) to remove outliers.*

- Find all flat surfaces:
  - First a “min/max pixel filter” is applied to the current **'DTM'** layer in mode Diff. brightest to darkest. The result is image layer **'ElevMaxMin'** showing all edges a high difference from elevated areas to low areas.
  - Based on this new image layer, a “multi-threshold segmentation” is applied where image object level 'New Level' is created, assigning the classification of preliminary ground objects for all areas with a value <= 1 in **'ElevMaxMin'**. All areas > 1 stay unclassified.
  - Now image layer 'ElevMaxMin' can be deleted.

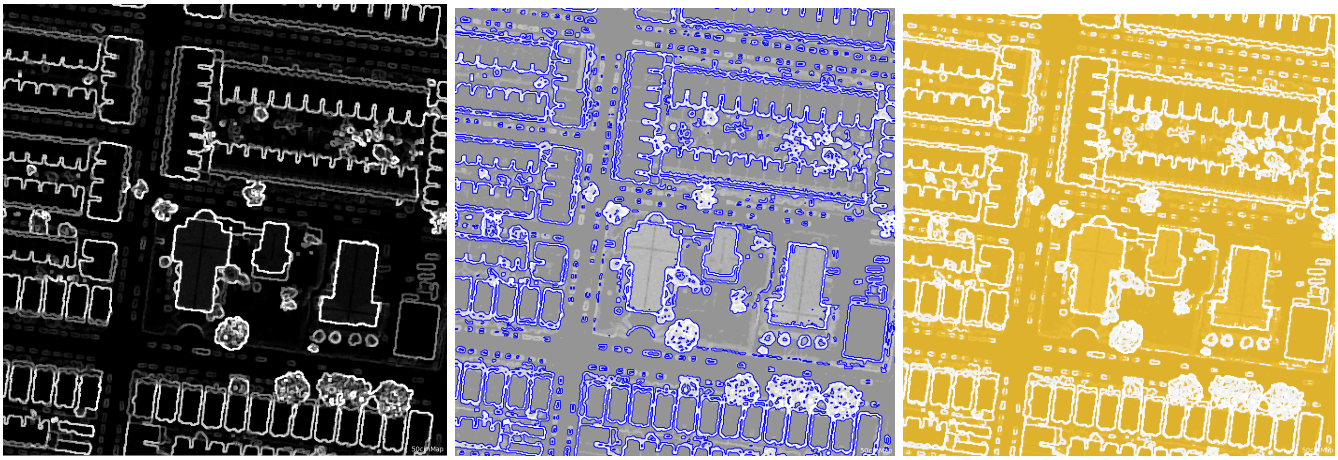


Figure 1.5: *Image layer* **'ElevMaxMin'** *(left) serving as basis for segmentation (middle) and classification result (right) to extract flat areas.*

- Eliminate small and thin objects
  - Based on 4 cycles of “pixel-based object resizing” using the **shrinking** mode, the class ground is reduced on its edges and these areas are assigned to _tempClass01.
  - Then 10 cycles of the same algorithm in mode **growing** - enlarges the class ground into class _tempClass01.
  - in a next step the remaining small _tempClass01 objects are set to **unclassified**.
  - merge all - fuses all image objects classified as ground to one large image object using the algorithm “convert image objects”. Keep in mind, this step really fuses all image objects belonging to one class - even if not adjacent - to one large area. In case single objects are needed a segmentation is necessary, dividing this large object again. (Alternative algorithm: **merge region**)

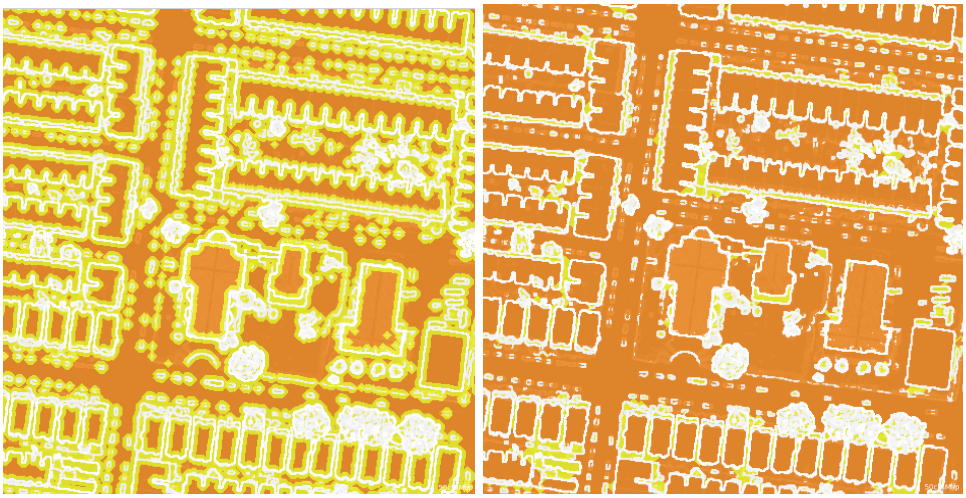


Figure 1.6: *Pixel-based object modification with shrink (left) and grow (right) approach within preliminary class ground to improve DTM object layer and eliminate small areas.*

- Reshape - the following algorithms have the purpose to divide big non-compact objects into meaningful image objects:
    - First a “quadtree segmentation” creates a sub level based on the current DTM layer resulting in New Level-1.
    - Then within the class **ground** all small image objects are lifted up to New Level using the algorithm “convert to sub-objects”.
    - New Level-1 is not needed anymore and deleted.

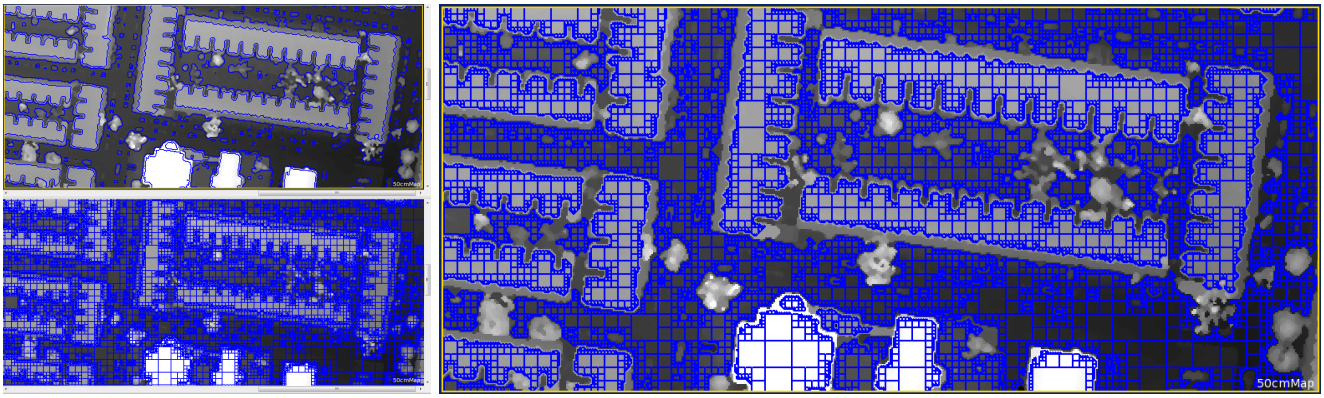


Figure 1.7: ***New Level*** *segmentation detail (left above) and* ***New Level-1*** *quadtree segmentation result based on DTM layer (left below) and its smaller objects lifted up to* ***New Level*** *within preliminary class ground (right).*

- Only now a “multiresolution segmentation” is applied, again using the current DTM layer and again applied within class ground, resulting in compact but meaningful image objects for the next set of rules.

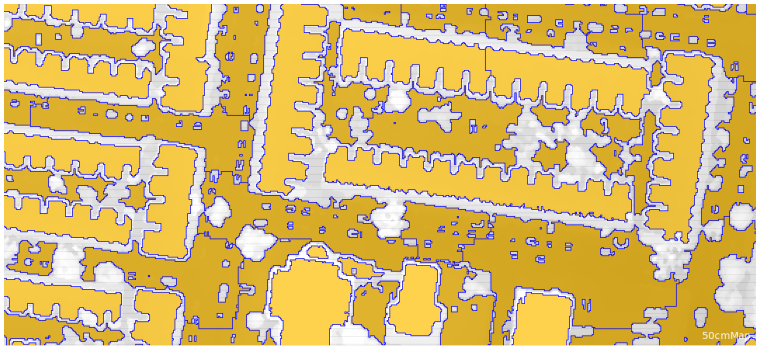


Figure 1.8: *New Level segmentation detail of **multiresolution segmentation** based on current DTM layer.*

- Analyze the border of the current ground candidates
  This set of rules excludes preliminary ground objects where the elevation difference to the neighbourhood is too high:
  - Classify and apply a chessboard segmentation to the border area to be analyzed
    - Here a “pixel-based object resizing” with 5 cycles is applied for class ground in mode **coat** resulting in areas classified as _tempClass01 that surround the current class ground.
    - Furthermore this new class is divided in very small image objects using a “chessboard segmentation” with object size 2.
    This step is necessary, because the two customized features, applied in the following steps, work based on objects. With smaller objects of similar size the feature values are more reliable.

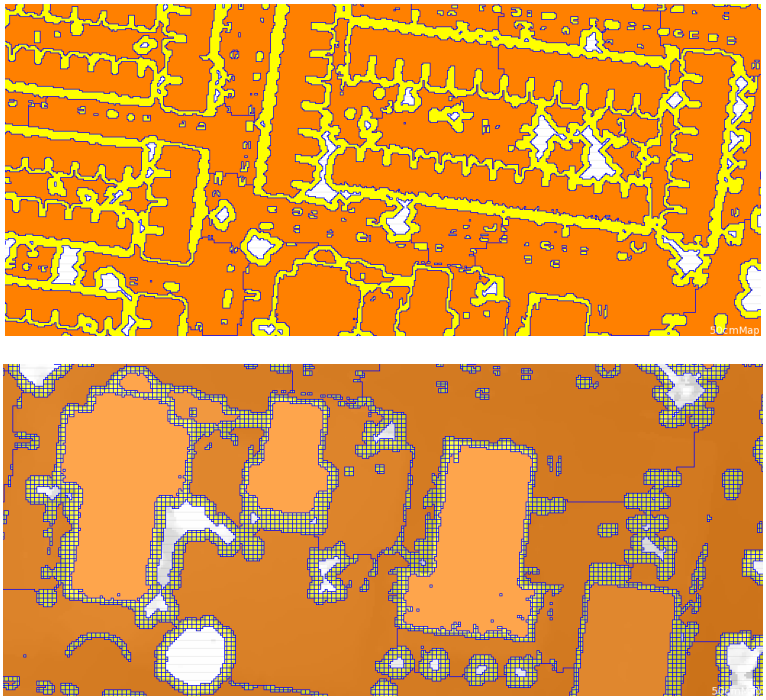
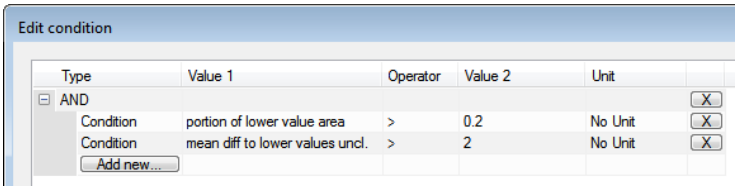


Figure 1.9: *Classification of areas surrounding the current class ground _tempClass01 (above) and small chessboard segmentation result of these areas (below).*

  - Within the potential class ground, the classification step “assign class” applies two customized features to remove objects that are elevated in relation to their surrounding: **ground with portion of lower value area > 0.2** and **mean diff to lower values uncl. > 2** get **unclassified**.



To explain the features have a look at the screenshots below:

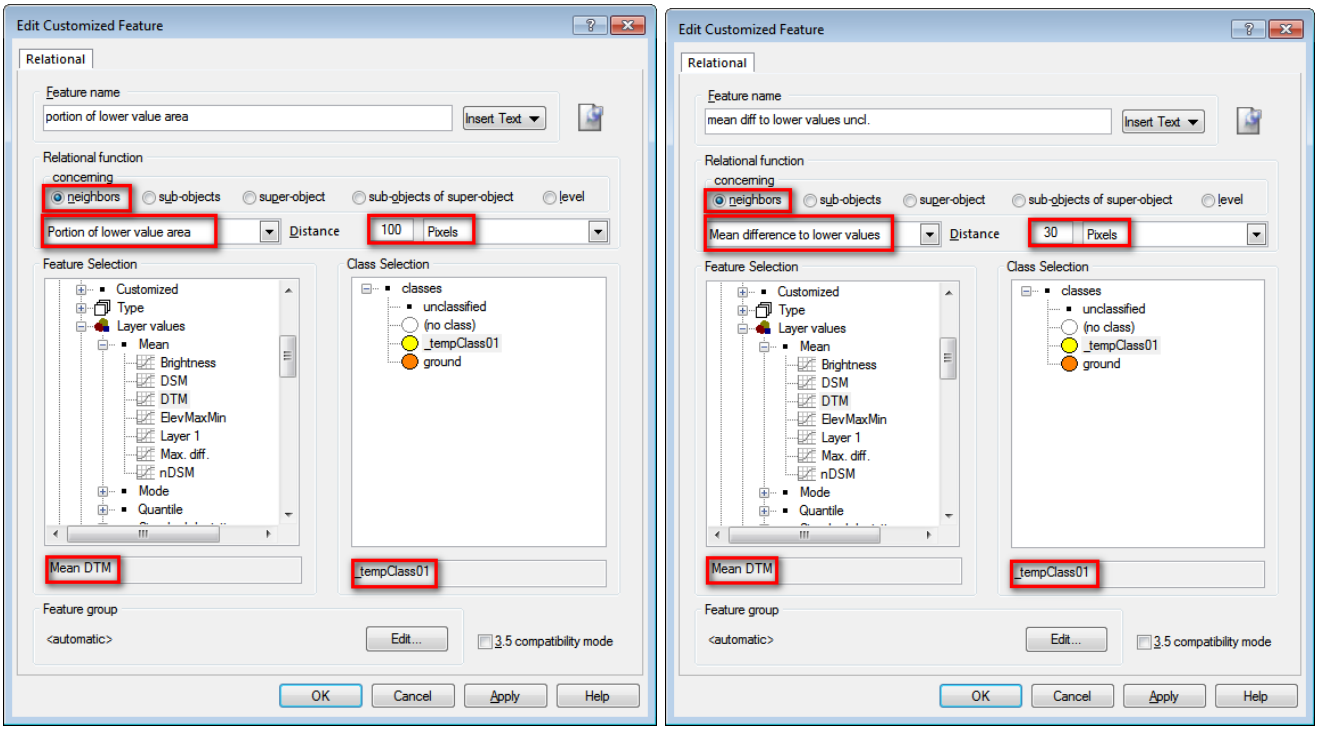


Figure 1.10: *Calculation of customized relational features portion of lower value area and mean diff to lower values uncl.*

To visualize or edit the features, go to the Feature view > Class-related features > Customized > right-click > Manage Customized Features > select feature and select Edit button to open the Edit Customized Feature dialog.

Both features consider the neighboring objects of class _tempClass01 in a defined distance for Mean values of the DTM.

The feature **portion of lower value area** (left) computes in a surrounding of 100 pixels the proportion of objects that are lying lower (Portion of lower value area in Mean DTM). If the proportion of the area surrounding an object within class _tempClass01 is dominated by objects lying below an object, this object is considered to be elevated in relation to its surrounding, and not a ground object. The feature is applied using the threshold portion of lower value area > 0.2.

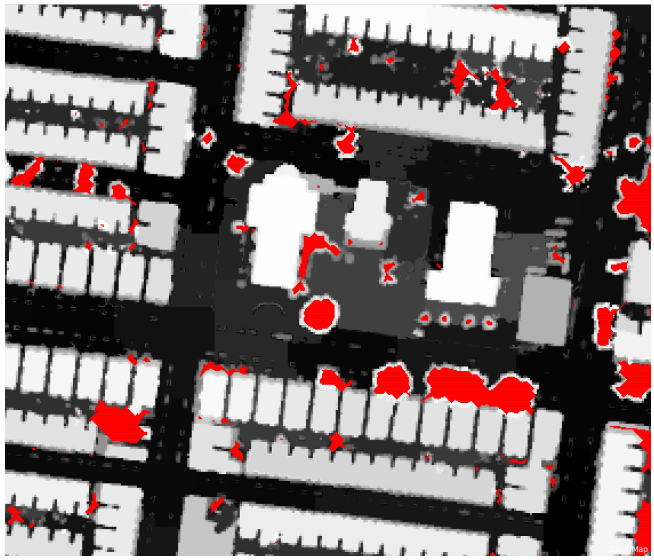


Figure 1.11: *Customized relational feature portion of lower value area (undefined areas in red).*

The feature **mean diff to lower values uncl.** (right) calculates in a smaller surrounding of 30 pixels the mean difference for the DTM, but only to objects with lower values. For all objects, a value is calculated if the object is a neighbor in the defined distance - that lies lower. The feature is applied using the threshold mean diff to lower values uncl. > 2.

Both features follow a similar approach in comparing an object with its surrounding. The combination of both features leads to a reliable result.

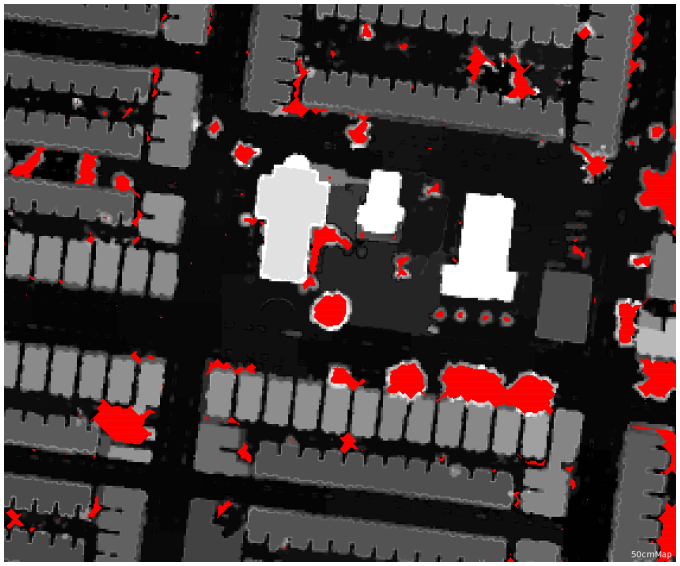


Figure 1.12 *Customized relational features mean diff to lower values uncl. (undefined areas in red).*

The classification step removes all elevated objects from the potential ground class:

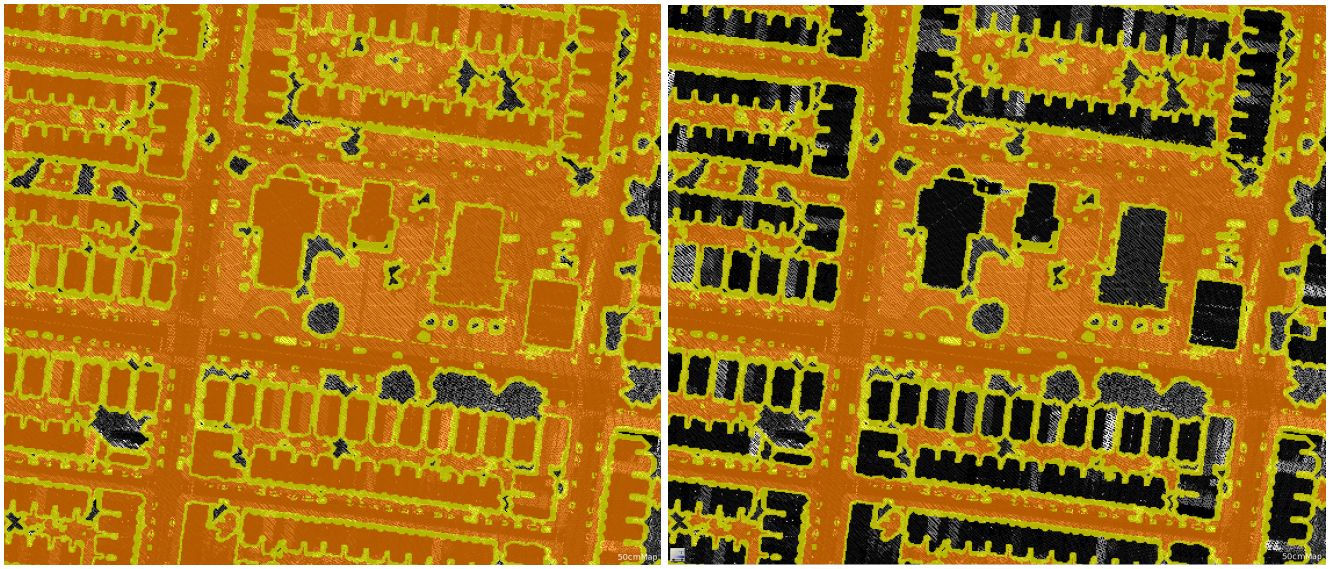


Figure 1.13: *Classification of ground before (left)and after (right) classification using customized relational features.*

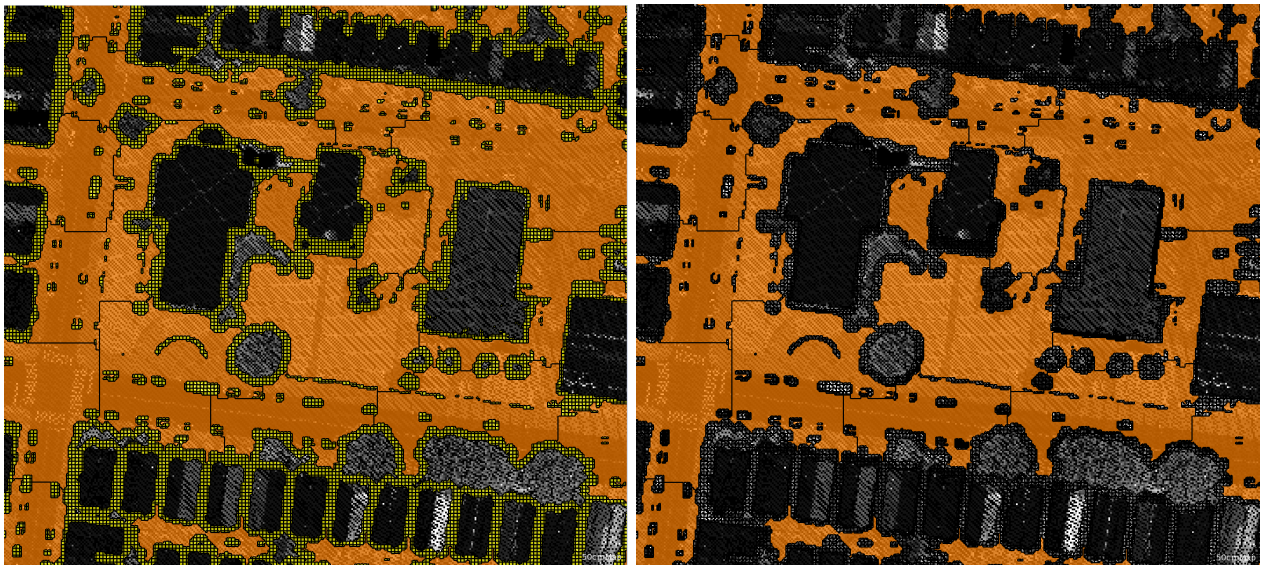


Figure 1.14: *Class ground _tempClass01 (left) and classification removed (right).*

- Clean up: here the classification of the temporary class _tempClass01 is removed and the algorithm “convert image objects” is applied afterwards to all unclassified image objects in mode Disconnected (fusion up) resulting in as large areas as possible for all unclassified image objects (one object for whole class unclassified).
  The class _tempClass01 was used to define the areas close to the potential ground objects as a kind of ‘*buffering*’ class and the customized features are calculated by using only this buffering class. This way it is avoided to compare the elevation of the ground candidates to objects that are not very close to it.

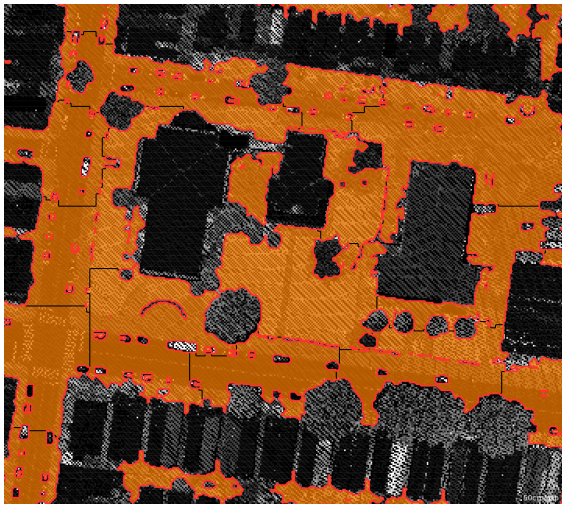


Figure 1.15: *Small unclassified image objects fused to one large unclassified area.*

Interpolate values within the non-ground areas

- Prepare areas to be filled - here in the preliminary DTM layer an elevation value of 0 is assigned to all unclassified areas using the algorithm “layer arithmetics”. In the following set of algorithms these values are increased iteratively.
- Replace all pixels with a value of 1 with values coming from the neighbouring ground objects
    - The algorithm “update variable” defines the scene variable **currentKernel** with a value set to 3 (If you go to Process > Manage Variables you can check the result - value = 3)

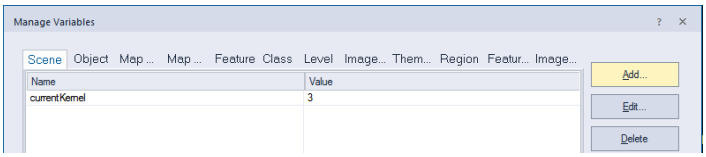


Figure 1.16: *Variable currentKernel initialized to a value of 3*

    - Infinite loop: The next algorithms are wrapped with an infinite loop (Number of cycles set to Infinite), while there are still unclassified objects the sub-processes are executed (Condition - while No. of unclassified > 0). The algorithms start with a kernel of 3 to fill first the "0" pixels located very close to the ground, then the kernel is increased to fill values that are farther away from ground. This operation is repeated until all values have been filled based on the DTM.

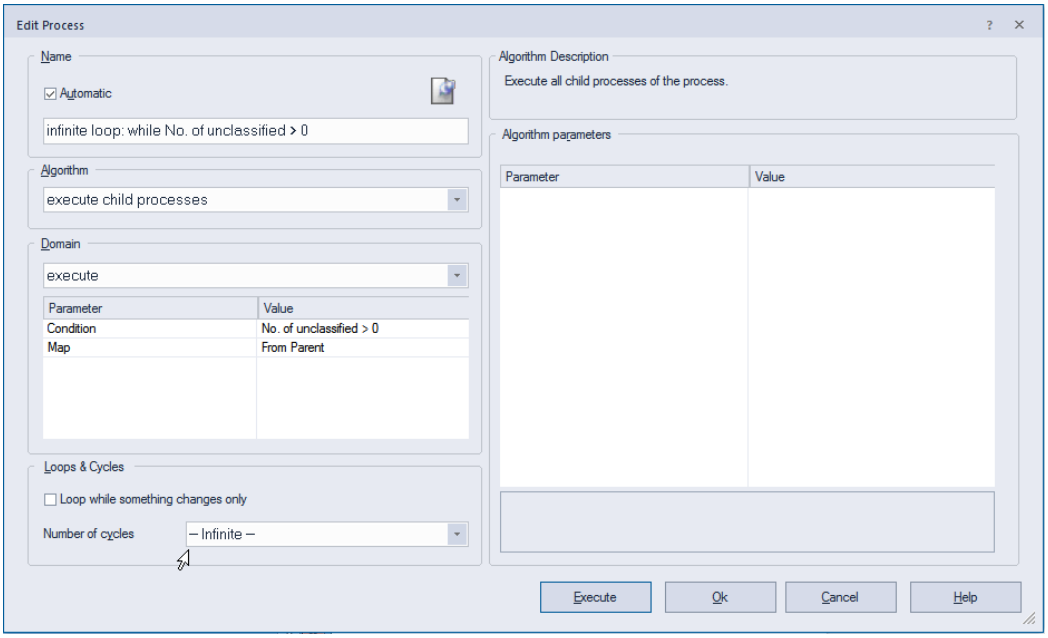


Figure 1.17: *Algorithm to define infinite loop for stepwise recalculation of DTM values*

- Starting within all unclassified objects in the domain, a “convolution filter” (Gauss Blur, currentKernel x currentKernel x 1) is applied to the DTM. In the beginning the currentKernel variable is set to 3, resulting in a 3x3 kernel size. Only classified objects serve as source pixels (ground and _tempClass01). The values of the DTM are overwritten. Each loop increases the value of the variable resulting in an increased kernel size.
- Next step is a “multi-threshold segmentation” based on the DTM values within unclassified objects to classify filled pixels so that they are not used again in the next loop. Starting from the borders of the unclassified areas, all non ground objects are filled with reasonable DTM values. (unclassified <= 0 < _tempClass01).
- The last algorithm “update variable” within the loop increases the scene variable **currentKernel** by: currentKernel = ([currentKernel]*2)+1. This results for the next cycle in currentKernel = 7. The kernel is increased in each cycle to fill more and more pixels.

Figure 1.18: *Stepwise calculation of DTM for currentKernel value 7 (left) and 15 (right)*

- Final refinement
  - In this first refinement step a “convolution filter” is applied to the DTM to remove edges within the filled regions (_tempClass01, Gauss Blur, with large Kernel size 41 x 41)
  - In a last step, to make sure that pixels close to the ground have similar values to the ground pixels, a “convolution filter” with a smaller kernel size is applied to _tempClass01 (Gauss Blur, 5 x 5)

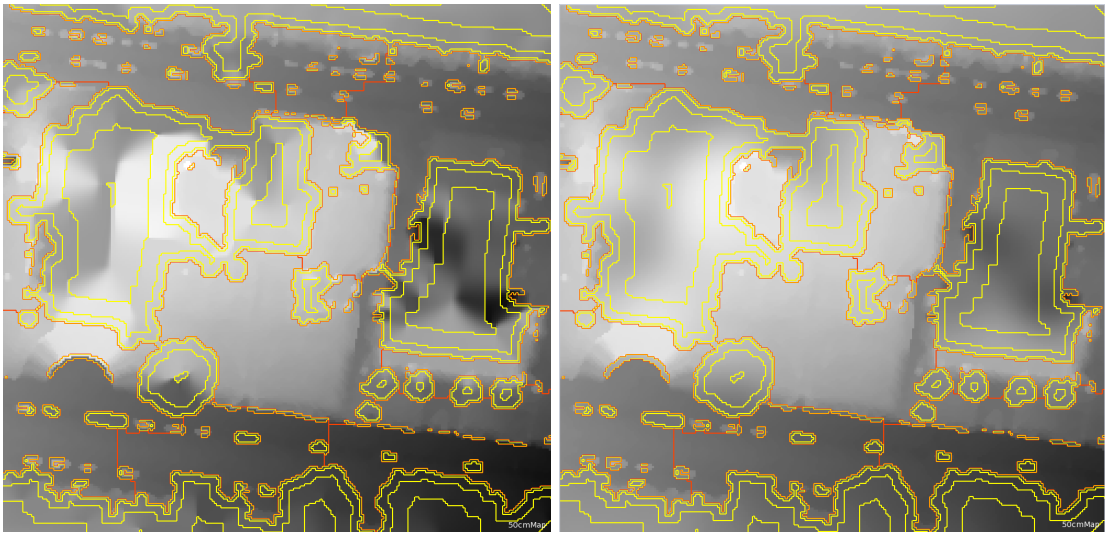


Figure 1.19: *DTM before final refinement steps (left) and filtered DTM data (right)*

- Finally the image object level is deleted, because it is not needed anymore.

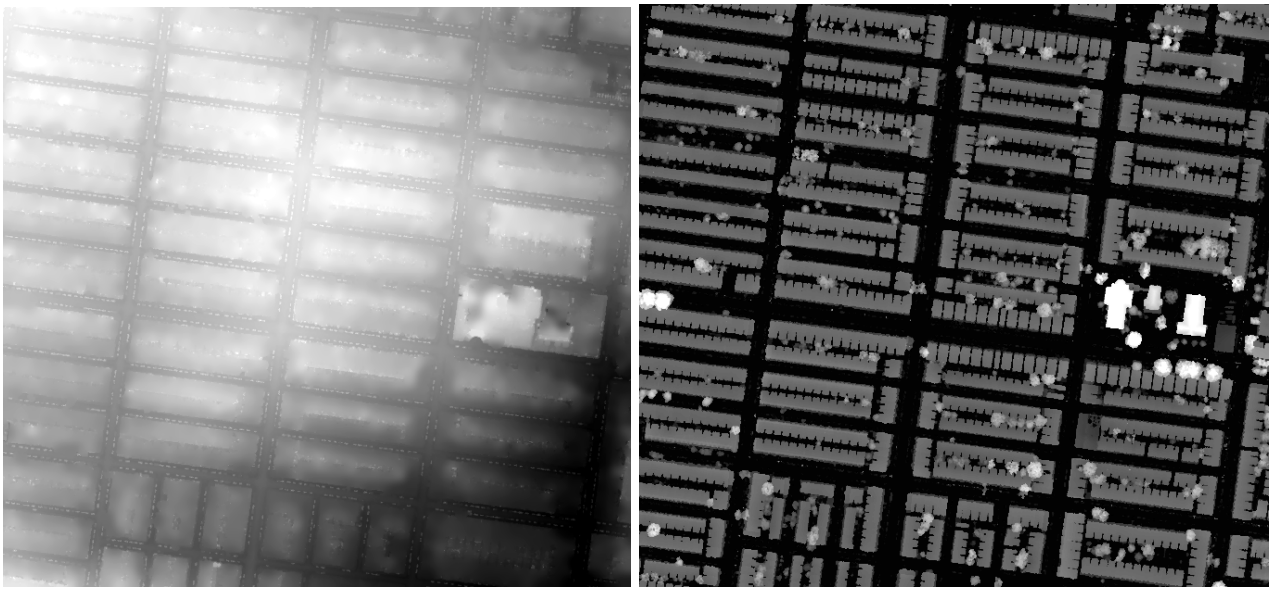


Figure 1.20: *Final DTM layer (left) and nDSM layer (right) calculated based on DSM-DTM*

Create nDSM

- The algorithm “layer arithmetics” calculates the normalized digital surface model by difference between DSM-DTM and writes the results in the according layer nDSM.
  Go to View > Image Layer Mixing and visualize the layer so see the result.

Classify Point Cloud

- In case some points are already classified the algorithm “assign class to point cloud” sets all points to class '1 - Unclassified' within the point cloud 'Layer 1'.
- The same algorithm applied a second time assigns finally the class '2 - Ground' to all point cloud points of 'Layer 1' with the condition that their elevation value are low (<20cm). Here a object feature is used (Feature View > Object features > Customized > point elev - DTM (abs)) that calculates the absolute value of the difference between the elevation (Z coordinate) and the feature 'Layer pixel value at point DTM' (Point cloud-related features > Layer pixel value at point).
  The intention here is to classify all points with an elevation similar (<20cm) to the ground elevation raster that was calculated. The absolute value is used because it is not important if the point has an elevation value higher or lower than the ground.

Now, visualize the final point cloud classification: Open the Point Cloud View Settings dialog and select Point cloud > Layer 1 > Show. Furthermore select render mode 'Classification' to see the result. Select a 3D subset to navigate in 3D mode with the classification visualized.

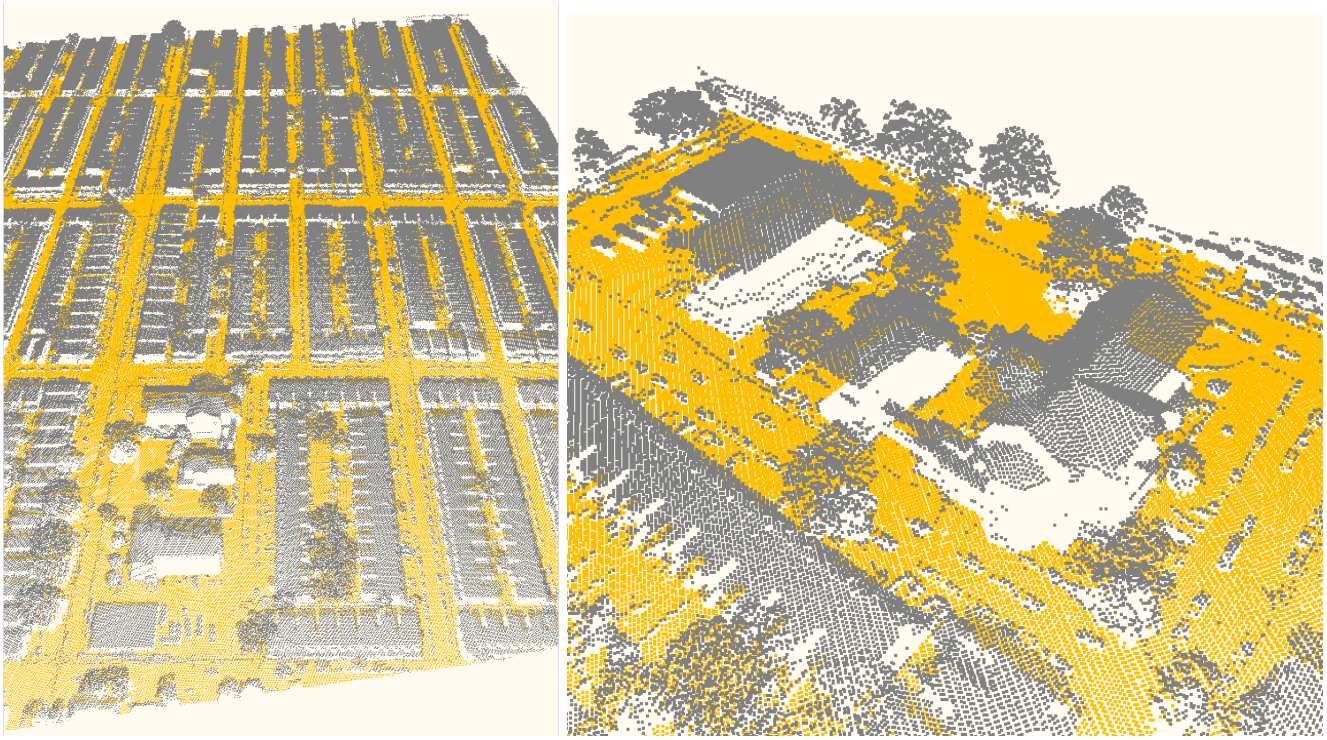


Figure 1.21: *Final point cloud classification of layer 1 (left) and detail (right).*

Lesson 2 – General Point Cloud Classification

2.1 Learning objective and contents

This lesson shows how to classify point cloud data in 3 main classes ground, buildings/roofs and vegetation.

1. Classify ground points
2. Classify buildings and roofs
3. Classify vegetation

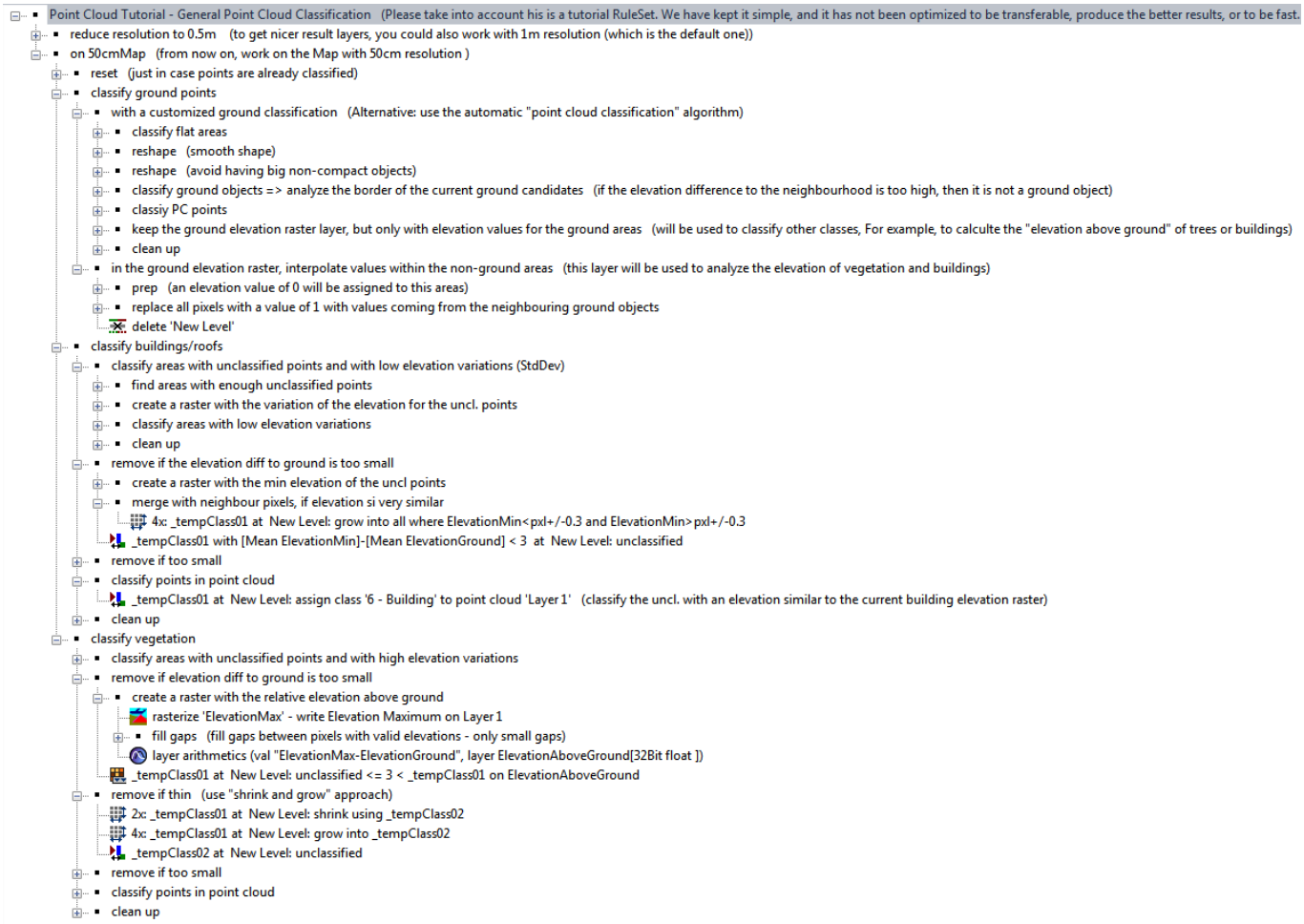
Estimated completion time: 30 minutes

2.2 Description of Rule Set - General Point Cloud Classification

Please start **eCognition Developer** and open the project via the menu **File > Open project > GeneralClassification.dpr.**

If not already opened please click on the ***pin*** in the lower right corner of the **process tree window** to visualize comments included in this rule set.

Overview Rule Set



Please execute the following processes now step-by-step, read through the comments included in the rule set and follow the instructions in this tutorial. Deepen your knowledge how to visualize and handle point cloud data based on **User Guide > Starting eCognition Developer > Navigating in 3D**.

2.2.1 Reduce resolution

As described in Lesson 1, the project has a pixel size of 1m/pixel for the rasterized intensity layer when opening the project. If you check the small drop-down menu **Select active map** before executing the first algorithm of the rule set - only **main** is available.

2.2.2 on 50cmMap

If you execute the first process, the resolution is changed to 0,5m/pixel creating a new map called **50cmMap**. From now on, all further algorithms are applied to this resolution and you have to change to this map using the **Select active map** drop-down to see the results of all processes.

Reset

Because the tutorial wants to demonstrate how to classify point cloud data and in case some points are already classified - this process removes the classification and assigns '0 - Created, never classified' to point cloud 'Layer 1'.

Classify Ground Points

With a customized ground classification

This part of the rule set is reused in this project and described in more detail in Lesson 1 (Project Create Terrain Models > Create DTM > approach 2 - with a customized ground classification > Classify flat areas):

- Classify flat areas
    - Produce raster with lowest elevation per pixel cell - the algorithm “rasterize point cloud” writes the elevation minimum (z-coordinate) from point cloud **Layer 1** to a new raster layer **'ElevationGround'**
    - Fill gaps & Filtering - the gaps are filled and outliers filtered
    - Find all flat surfaces:
      - First a “min/max pixel filter” is applied to the current **'ElevationGround'** layer in mode Diff. brightest to darkest. The result is image layer **'ElevMaxMin'** showing all edges a high difference from elevated areas to low areas.
      - Based on this new image layer a “multi-threshold segmentation” is applied where image object level 'New Level' is created, assigning the classification of preliminary ground objects for all areas with a value <= 1 in **'ElevMaxMin'**. All areas > 1 stay unclassified.
      - Now image layer 'ElevMaxMin' can be deleted.
    - Eliminate small and thin ones
      - Based on 4 cycles of “pixel-based object resizing” using the **shrinking** mode, the class _tempClass01 is reduced on its edges and these areas are assigned to _tempClass02.
      - Then 10 cycles of the same algorithm in mode **grow** - enlarges the class _tempClass01 into class _tempClass02.
      - in a next step the remaining small _tempClass02 objects are set to **unclassified**.
      - merge all - fuses all adjacent unclassified image objects using the algorithm “merge region” and in a second step all objects classified as _tempClass01 are merged to as large but separate objects as possible.
- Reshape - here the objects are smoothed using a shrinking and growing approach with minor modifications:
    - Based on 1 cycle of “pixel-based object resizing” using the **shrinking** mode, the class _tempClass01 is reduced on its edges and these areas are assigned to _tempClass02.
    - Then 1 cycle of the same algorithm in mode **growing** - enlarges the class _tempClass01 into class _tempClass02 slightly.

- In a next step the remaining small _tempClass02 objects are set to **unclassified**.

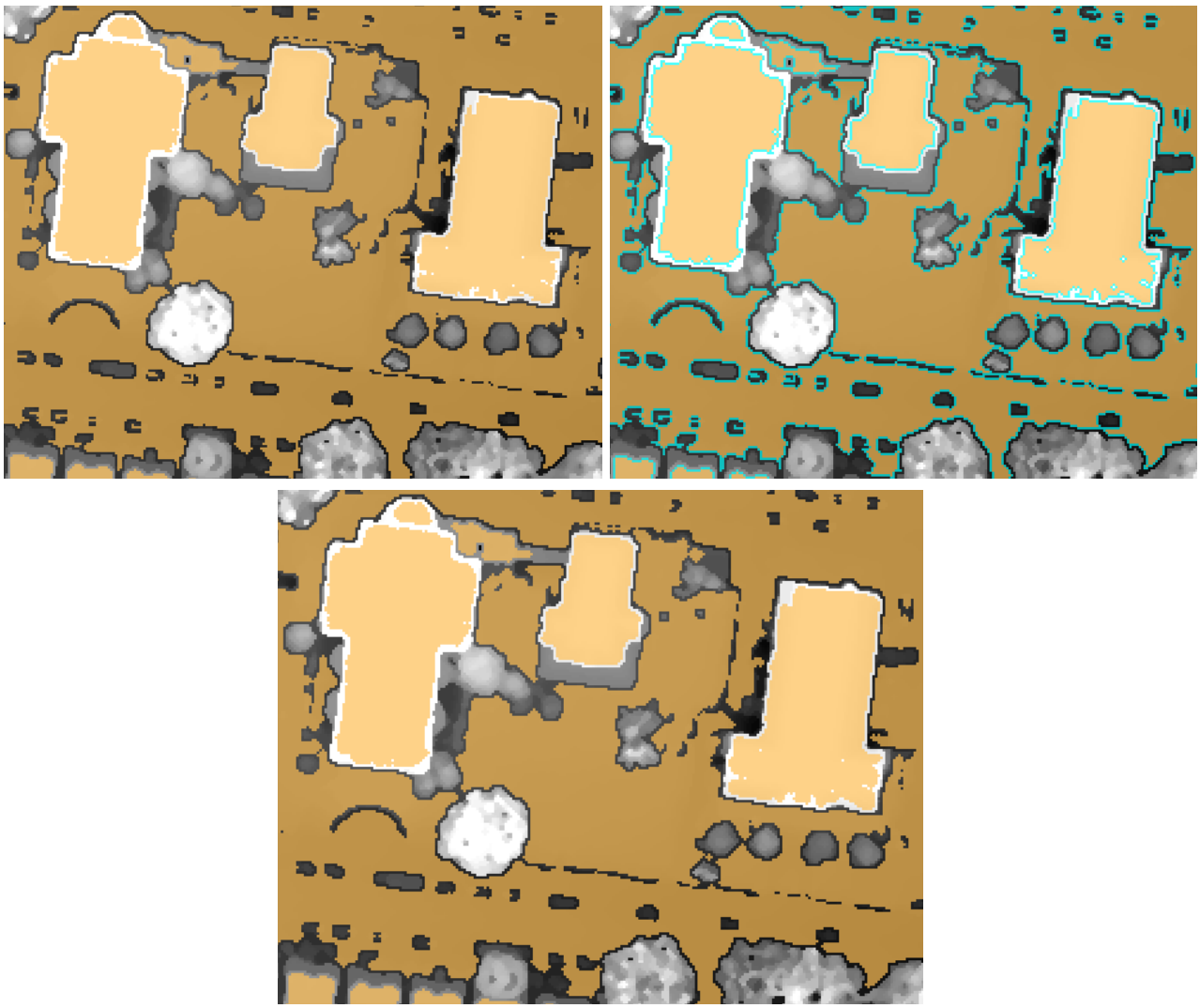


Figure 2.1: *Approach to smooth outline slightly - initial image objects (above, left), by shrinking (above, right) and growing (below).*

- Reshape - the following algorithms have the purpose to divide big non-compact objects into meaningful image objects. Similar basic principles are explained in Lesson 1 (Create DTM > approach 2 - with a customized ground classification > reshape (avoid having big non-compact objects)):
  - First a “chessboard segmentation” divides the image object level into very large square image objects (1000x1000). This step is necessary because the next step uses a quadtree segmentation with mode "super object form" and this algorithm needs an upper level.
  - A “quadtree segmentation” creates now a sub level based on the current ElevationGround layer resulting in New Level-1.
  - Then within the class **_tempClass01** all small image objects are lifted up to New Level using the algorithm “convert to sub-objects”.
  - New Level-1 is not needed anymore and deleted.
  - Only now a “multiresolution segmentation” is applied, again using the current ElevationGround layer and again applied within class _tempClass01, resulting in compact but meaningful image objects for the next set of rules.

- Classify ground objects => analyze the border of the current ground candidates
  This set of rules excludes preliminary ground objects where the elevation difference to the neighbourhood is too high. (The customized features used here are explained in more detail in lesson 1):
    - Classify and chessboard the border area to be analyzed
      - Here a “pixel-based object resizing” with 5 cycles is applied for class _tempClass01 in mode **coat** resulting in areas classified as _tempClass02 that surround the current class _tempClass01.
      - Furthermore this new class is divided in very small image objects using a “chessboard segmentation” with object size 2.
    - Within the class _tempClass01, the classification step “assign class” applies two customized features to remove objects that are elevated in relation to their surrounding: **_tempClass01 with portion of lower value area > 0.2** and **mean diff to lower values uncl. > 2** are classified as **_tempClass02**.
      The classification step removes all elevated objects of class _tempClass01.
    - Clean up: removes the classification of the temporary class _tempClass02.
- Classify PC points
    - Here a “median filter” is applied to ElevationGround to interpolate the values of the image layer using a kernel size of 3.
    - Afterwards a “convolutional filter” is applied to _tempClass01 (Gauss Blur, 5 x 5)

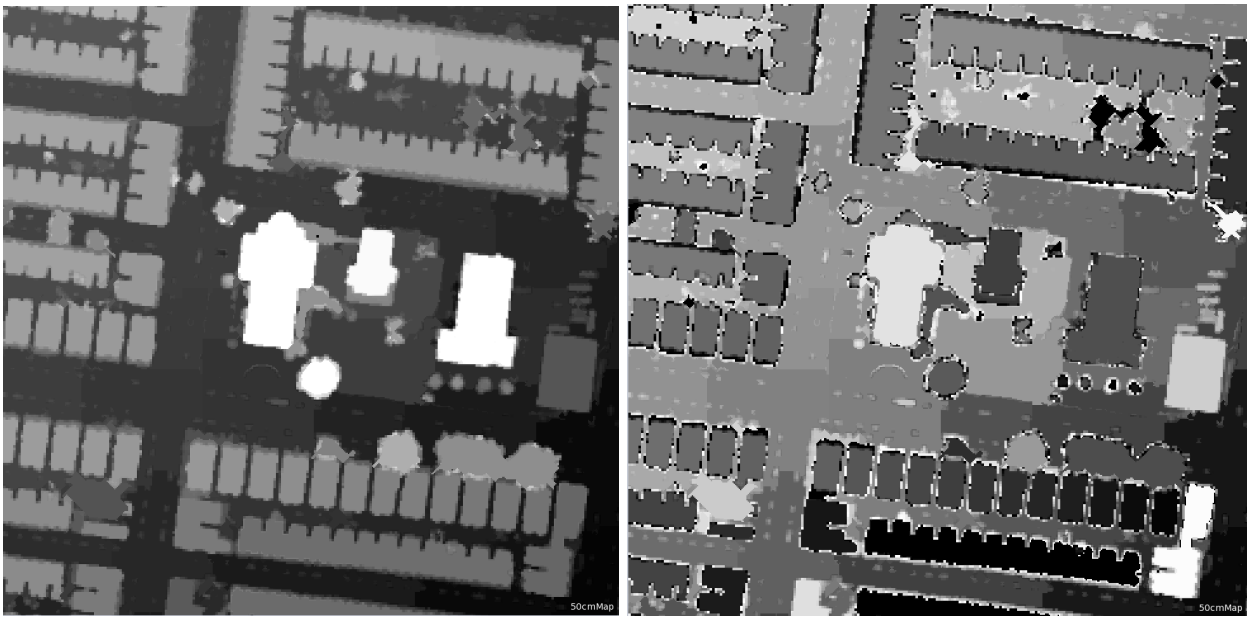


Figure 2.2: *Elevation ground (left)and median and convolution filter applied (right).*

- The algorithm “assign class to point cloud” uses the customized feature **point elev - Ground Elevation (abs)**. This feature calculates the absolute difference between elevation and the pixel value in the ElevationGround image layer: abs ([Z coordinate]-[Layer pixel value at point ElevationGround]).
  The **point condition** point elev - Ground Elevation (abs) < 0.2 has to be fulfilled so that a point gets assigned to class **'2 - Ground'**.

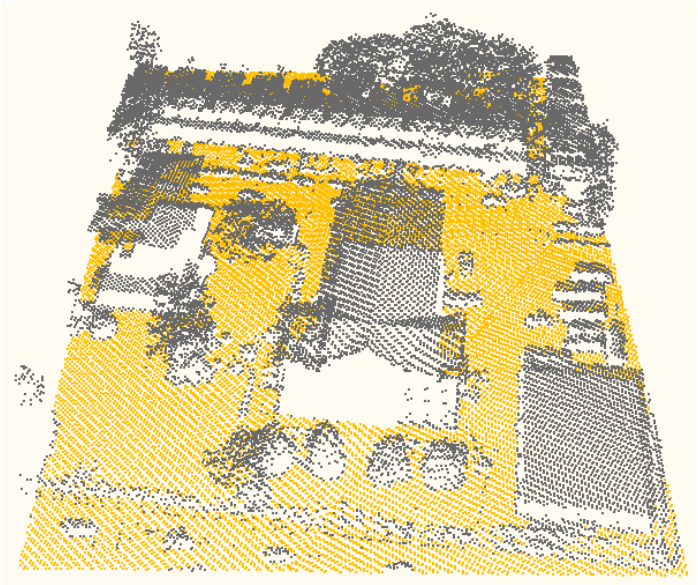


Figure 2.3: *Classification result of class 2 - Ground (yellow).*

- Keep the ground elevation raster layer, but only with elevation values for the ground areas
  For all unclassified areas corresponding to elevated areas, the algorithm “layer arithmetics” overwrites the values with a value of 0 in layer ElevationGround.

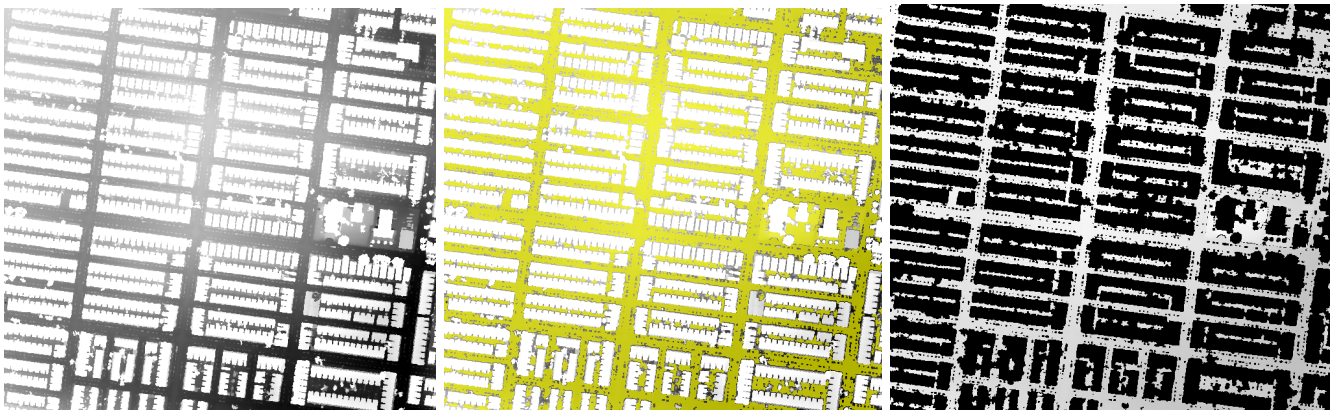


Figure 2.4: *Elevation ground (left) with classified areas (middle) and unclassified areas set to 0 (right).*

- Clean up - deletes the current level.

In the ground elevation raster, interpolate values within the non-ground areas

The layer created here will be used to analyze the elevation of vegetation and buildings (main steps explained in Lesson 1 already):

- Prep - now all ground areas are fused to one large, unclassified area using a “multi-threshold segmentation” in layer ElevationGround (unclassified <= 0 < _tempClass01)
- Replace all pixels with a value of 1 with values coming from the neighbouring ground objects
  - A variable is defined, that is increased in cycles to increase a kernel size applied to ElevationGround. The variable **'currentKernel'** is defined and set to 3, resulting in a 3x3 kernel size in the next filtering algorithm.
  - Infinite loop: while No. of unclassified > 0
    The next algorithms are wrapped with an infinite loop (Number of cycles set to Infinite), while

there are still unclassified objects the sub-processes are executed:
- In the beginning of this set of algorithms for all unclassified objects in the domain, a "convolution filter" (Gauss Blur) is applied to **ElevationGround**. Objects of the classes _tempClass01 and _tempClass02 serve as source pixels. The values of ElevationGround are overwritten.
- Next step is a "multi-threshold segmentation" based on ElevationGround values within unclassified objects to classify filled pixels so that they are not used again in the next loop. Starting from the borders of the unclassified areas, all non ground objects are filled with reasonable ElevationGround values. (unclassified <= 0 < _tempClass02).
- The last algorithm "update variable" within the loop increases the scene variable **currentKernel** by: currentKernel = ([currentKernel]*2)+1. This results for the next cycle in currentKernel = 7. The kernel is increased in each cycle to fill more and more pixels.

    - final refinement
      - In this first refinement step a "convolution filter" is applied to the **DTM** to remove edges within the filled regions (_tempClass02, Gauss Blur, with large Kernel size 41 x 41)
      - In a last step, to make sure that pixels close to the ground have similar values to the ground pixels a "convolution filter" with a smaller kernel size is applied to _tempClass02 (Gauss Blur, 5 x 5)

- Because the object level is not needed anymore it is deleted ('New Level').

Classify Buildings/Roofs

Classify areas with unclassified points and with low elevation variations (StdDev)

The classification of buildings and roofs uses the variation of the elevation in point cloud data. Building *edges* and vegetation have high variation - where buildings and roofs have low values for elevation variation. Correlated with this the number of points that is recorded, scattering at building edges and also within vegetation having a rough surface in comparison to man made features.
Our approach creates a layer with the number of unclassified points per pixel and then objects for all pixels with at least a certain number of unclassified points.

- Find areas with enough unclassified points
    - Based on the initial point cloud Layer 1 the algorithm "rasterize point cloud" creates an image layer called **'NumberOfPoints'** - based on all points that have no classification yet (Class filter: 0 - Created, never classified). The point property used is the 'number of points' with kernel size 3. The new layer represents then the number of points in the point cloud that are mapped to the pixel.
    - Based on this new image layer (NumberOfPoints) the "multi-threshold segmentation" creates an image object level 'New Level' splitting the image in two areas - **unclassified for values <= 4** and **_tempClass01 > 4**.
    - Clean up - the image layer 'NumberOfPoints' is deleted.
- Create a raster with the variation of the elevation for the uncl. Points
    - In case the image layer 'ElevationStdDev' exists already within this rule set it is deleted here. If you reutilize parts of rule sets, this is a recommended step to reset the data and make your rule set transferable. The layer will be created in the next step.
    - The algorithm "rasterize point cloud" creates within class _tempClass01 the image layer 'ElevationStdDev' using the point property standard deviation of elevation (Z coordinate). Edges and vegetation have high values in this image layer.

Figure 2.5: *Layer 'NumberOfPoints' that represents the number of points in the point cloud that are mapped to a pixel (left) and 'ElevationStdDev' based on point property standard deviation of elevation (right).*

- Classify areas with low elevation variations - here the algorithm “multi-threshold segmentation” reclassifies _tempClass01 based on ElevationStdDev values in **_tempClass01 <= 0.5** and **removes** the classification for all values **> 0.5**. The result is the removal of mainly building edges and vegetation of class _tempClass01, representing now the core parts of buildings and roofs.
- Clean up - At this point the image layer 'ElevationStdDev' is deleted.

Remove if the elevation diff to ground is too small

The approach here assumes that a building is at least 3 meter elevated above ground. By looking at the *minimum* elevation of the points for an object (building candidate) the elevation difference between its lowest part and the ground can be measured.

- Create a raster with the minimum elevation of the unclassified points
    - Within all unclassified points of the point cloud, the algorithm “rasterize point cloud” produces the image layer '**ElevationMin'** based on the *minimum value* of the elevation (Z coordinate) with a kernel size 1.

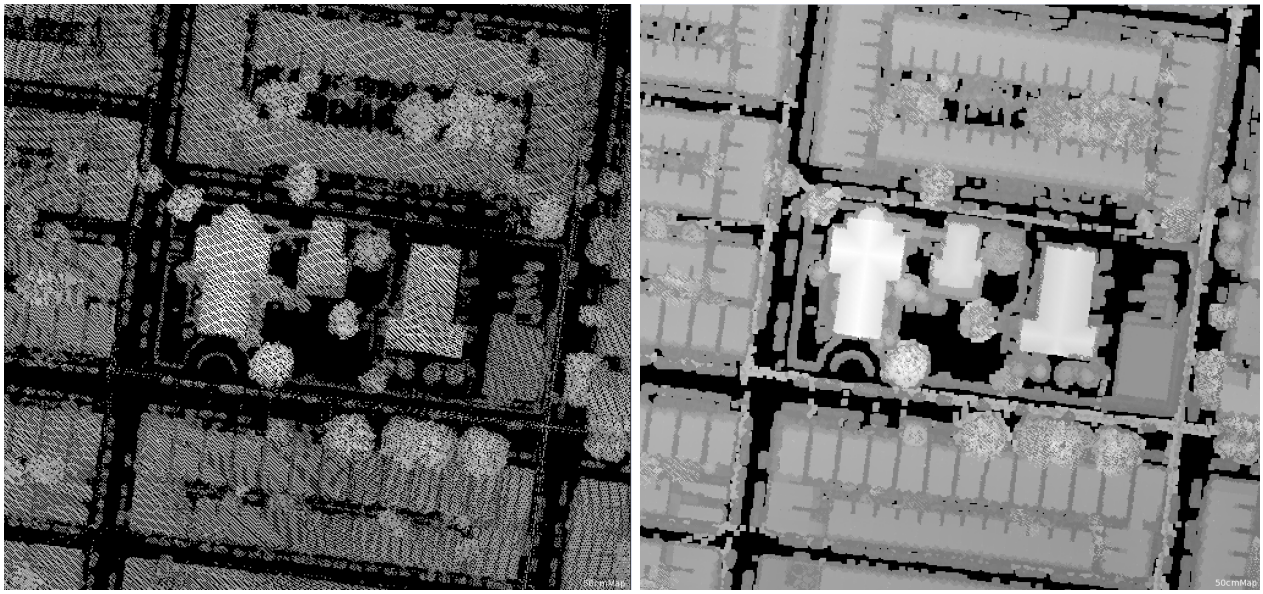


Figure 2.6: *Preliminary image layer 'ElevationMin' (left) refined after filtering in two cycles (right).*

- fill gaps (fill gaps between pixels with valid elevations - only small gaps)
  - all 2x: do - this algorithm controls that the following two sub algorithms are executed twice to fill small gaps (in case larger gaps have to be filled the number of iterations can be increased):

    - In a first segmentation the “multi-threshold segmentation” creates a new level **'New Level-1'** based on **ElevationMin**, where all pixels are segmented to a large object and classified as **class '_tempClass01'** with an elev ≠ 0. All pixels with a value = 0 are assigned to **'**unclassified**'** (Note that there might be data with negative values. Because this segmentation does not allow to insert the value < 0 and > 0, this workaround is used for the lower border of the interval set to _tempClass01 <= -0.0001).
    - Now a “median filter” is applied to this object level to interpolate the gaps using the class _tempClass01, where all unclassified objects are filtered using a kernel size of 3. The resulting values are written into the raster layer **'ElevationMin'** again.

  - Because the object level is not needed anymore it is deleted (**'New Level-1')**.

- Merge with neighbour pixels, if elevation is very similar - In this step the algorithm “pixel-based object resizing” is applied within _tempClass01 in the growing mode. Objects grow if ElevationMin < pxl+/-0.3 and ElevationMin > pxl+/-0.3 in 4 cycles. This step enlarges the areas of _tempClass01 slightly.
- “Assign class” - now this classification algorithm sets image objects of class _tempClass01 to unclassified if **[Mean ElevationMin]-[Mean ElevationGround] < 3**.

  To edit the feature go to the Feature view > Object features > Customized > right-click feature [Mean ElevationMin]-[Mean ElevationGround] > Edit > to open the Edit Customized Feature dialog or double-click the feature to visualize it.

  The feature compares the minimum elevation of a point cloud point with those points that are classified within the point cloud as ground. Those objects where the difference to ground points is too small are removed.

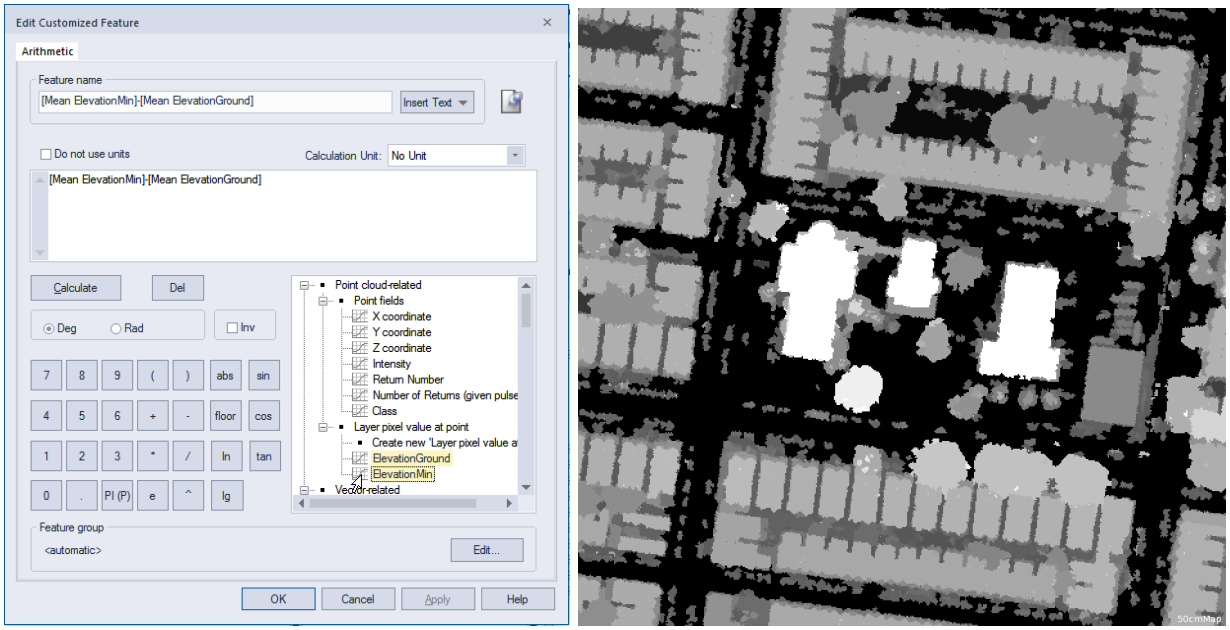


Figure 2.7: *Calculation of the customized feature [Mean ElevationMin]-[Mean ElevationGround]' and visualisation in view*

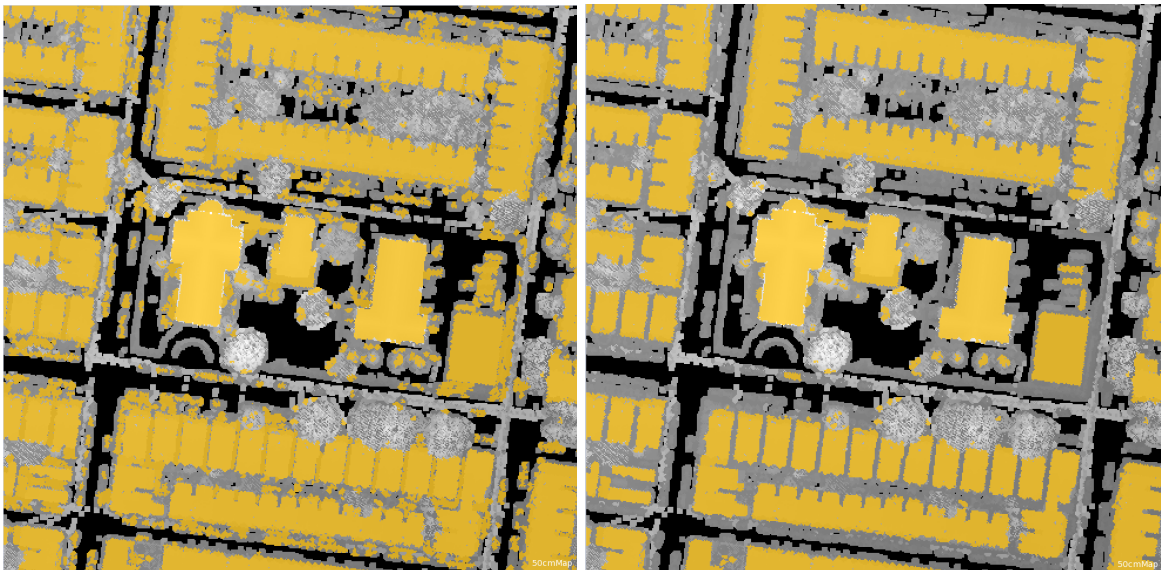


Figure 2.7: *Classification of _temClass01 before and after application of customized feature*

Remove if too small

- _tempClass01 at New Level: “merge region” fuses all adjacent _tempClass01 image objects.
- _tempClass01 with Number of pixels < 150 at New Level: unclassified - here all small _tempClass01 objects are changed to unclassified.

Classify points in point cloud

- Finally, the algorithm “assign class to point cloud” modifies the point cloud points for class '6 - Building' for all points that fulfill the condition for the customized feature point elev - ElevationMin (abs) < 0.2 (explanation see Lesson 1 > last algorithm set > **Classify Point Cloud**).

Clean up

Here the current image object level and image layer 'ElevationMin' are deleted.

- delete 'New Level'
- delete image layer 'ElevationMin'

Classify Vegetation

To distinguish vegetation, the rule set is using the characteristic that these areas have a very high elevation variation it their surface. The property used is the *maximum* value of the elevation representing the diversified surface of the vegetation as good as possible.

- classify areas with unclassified points and with high elevation variations
    - Find areas with enough unclassified points
      - As described in the classification for buildings and roofs in this lesson, the algorithm “rasterize point cloud” creates the image layer **'NumberOfPoints'** by writing the point property Number of points with a Kernel size 3 for points with no classification.
      - Creating 'New Level': unclassified <= 4 < _tempClass01 on NumberOfPoints applies a “multi-threshold segmentation” to this image layer and assigns all values > 4 to _tempClass01.
      - Clean up - Then image layer 'NumberOfPoints' is deleted.
    - Create a raster with the variation of the elevation for the unclassified points
      - First, the current image layer 'ElevationStdDev' is deleted.
      - Now the algorithm “rasterize point cloud” creates within class _tempClass01 the image layer 'ElevationStdDev' using the point property standard deviation of elevation (Z coordinate).
    - Classify areas with high elevation variations - applies a “multi-threshold segmentation” to _tempClass01 reducing its area slightly based on the condition - unclassified <= 0.5 < _tempClass01 for values of the image layer 'ElevationStdDev'.
    - Clean up - deletes image layer 'ElevationStdDev'
- remove if elevation diff to ground is too small
    - Create a raster with the minimum elevation of the unclassified points
        - Within all unclassified points of the point cloud, the algorithm “rasterize point cloud” produces the image layer '**ElevationMax**' based on the maximum value of the elevation (Z coordinate) with a kernel size 1.
        - fill gaps (fill gaps between pixels with valid elevations - only small gaps)
          - all 2x: do - this algorithm controls that the following two sub algorithms are executed twice to fill small gaps (in case larger gaps have to be filled the number of iterations can be increased):
          - In a first segmentation the “multi-threshold segmentation” creates a new level **'New Level-1'** based on **ElevationMax**, where all pixels are segmented to a large object and classified as **class '_tempClass01'** with an elev ≠ 0. All pixels with a value = 0 are assigned to **'**unclassified**'** (Note that there might be data with negative values. Because this segmentation does not allow to insert the value < 0 and > 0, this workaround is used for the lower border of the interval set to _tempClass01 <= -0.0001).
          - Now a “median filter” is applied to this object level to interpolate the gaps using the

class _tempClass01, where all unclassified objects are filtered using a kernel size of 3. The resulting values are written into the raster layer **'ElevationMax'** again.
- Because the object level is not needed anymore it is deleted (**'New Level-1'**).

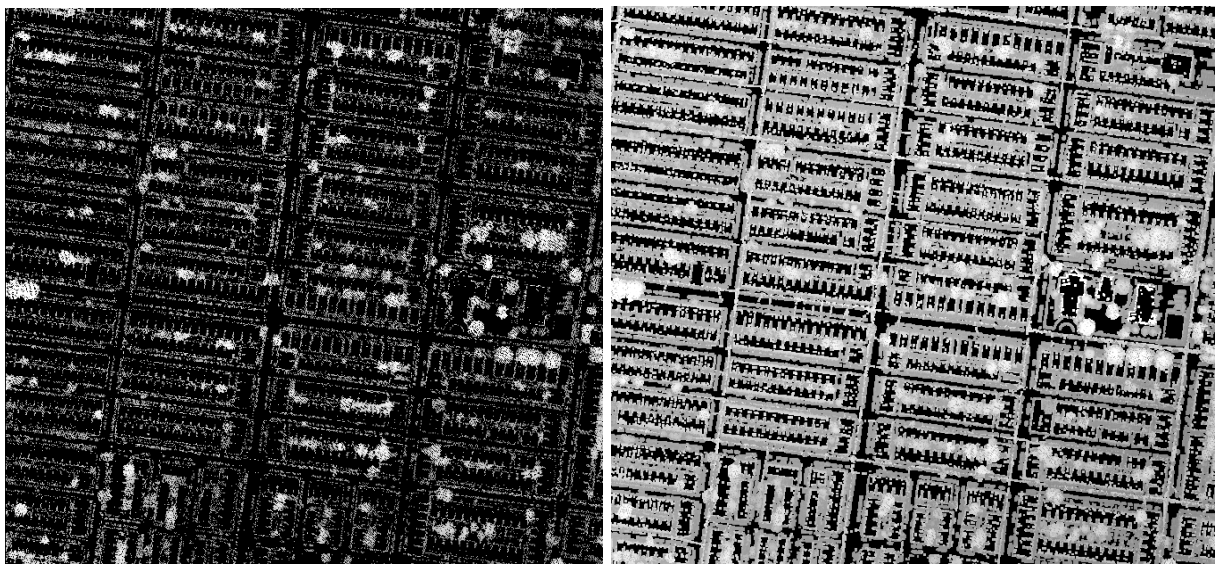


Figure 2.8: *Image layer 'ElevationMax' before (left) and after filling of gaps in 2 cycles (right)*

- Afterwards, the algorithm “layer arithmetics” calculates the difference between 'ElevationMax-ElevationGround' creating image layer **'ElevationAboveGround'**.

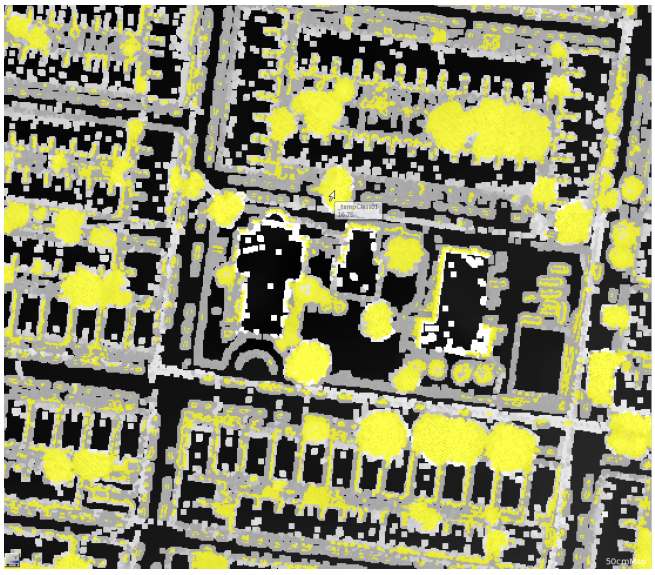


Figure 2.9: *Detail of difference layer 'ElevationAboveGround' with _tempClass01*

  - Now a “multi-threshold segmentation” creates a new objects within _tempClass01 with unclassified <= 3 < _tempClass01 based on image layer 'ElevationAboveGround'.
- remove if thin (use "shrink and grow" approach)
  This part of the rule set smoothes object shapes and removes non-compact objects assuming that vegetation and trees are larger and compact areas.
  - During this improvement step based on the algorithm “pixel-based object resizing” only objects that are large and compact are kept, while non-compact objects are removed using a "shrink and grow" approach:

- In 2 cycles _tempClass01 **shrinks** into _tempClass02 using “pixel-based object resizing”.
- In 4 cycles _tempClass01 is **grown** into the _tempClass02.

○ Then _tempClass02 is unclassified.

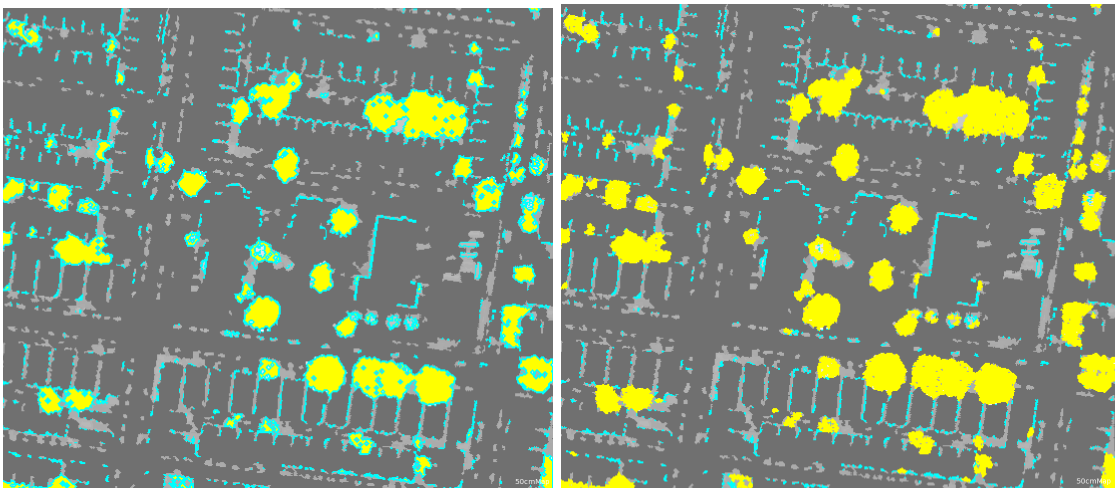


Figure 2.10: *Classification detail of shrink (left) and grow (right) approach.*

- remove if too small
  - To remove small object _tempClass01 is fused using the algorithm “merge region”
  - Now all objects of _tempClass01 with a size Number of pixels < 100 are unclassified.
- classify points in point cloud - Finally based on the algorithm “assign class to point cloud” assigns for points with image objects classified as _tempClass01 now the class '5 - High Vegetation'. The algorithm uses the customized feature **point elev - Ground Elevation (abs)**. This feature calculates the absolute difference between elevation and the pixel value in the ElevationGround image layer: abs ([Z coordinate]-[Layer pixel value at point ElevationGround]).
  The **point condition** point elev - Ground Elevation (abs) > 0.5 has to be fulfilled so that a point gets assigned to class '5 - High Vegetation'.
- clean up - deletes all image layers and levels:
  - delete 'New Level'
  - delete image layer 'ElevationMax'
  - delete image layer 'ElevationAboveGround'

You can now render the point cloud in mode classification to see the classification results:

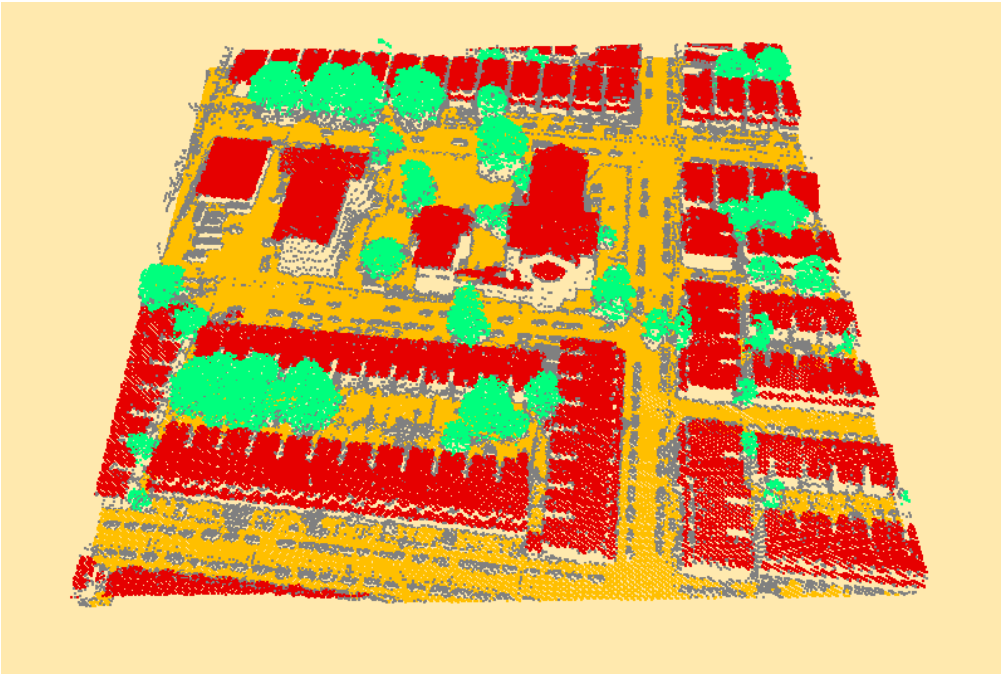


Figure 2.11: *Classification result of point cloud*

Lesson 3 – Classify Differences in Multitemporal in Point Cloud data

3.1 Learning objective and contents

In this exercise you learn how to apply rules to point cloud data in eCognition and how to detect changes between imagery of two acquisition times. In this tutorial you will find these changes based on elevation differences and assign these changes to two different classes. The approach rasterizes point clouds, finds differences and assigns them to man made and vegetation. Finally, you will create a point cloud with the classified changes only.

The rule set comprises the following steps:

1. First step - Reduce resolution
2. Find changes - Find differences & create rasterized elevation layers
3. Find negative changes - Find disappeared features
4. Classify changes - assign areas to vegetation and man-made
5. Create point cloud with the changes only

Estimated completion time: 30 minutes to 1 hour depending on familiarity with the eCognition software.

3.2 Description of Rule Set - Change detection

Please start **eCognition Developer** and open the project **ChangeDetection.dpr** via the menu **File > Open project.**

If not already opened please click on the ***pin*** in the lower right corner of the **process tree window** to visualize comments included in this rule set.

Overview Rule Set

3.2.1 Reduce resolution

As described in Lesson 1, the project has a pixel size of 1m/pixel for the rasterized intensity layer when opening the project changed by the first process to 0,5m/pixel creating a new map called **50cmMap**.

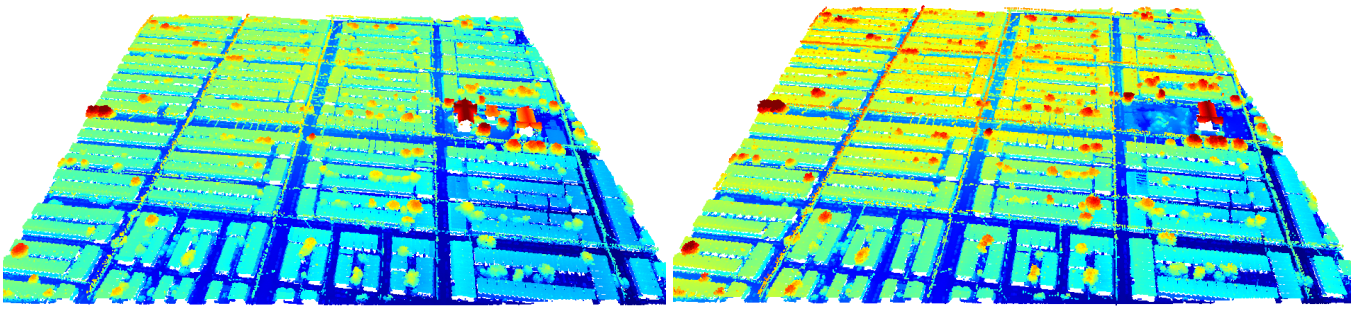


Figure 3.1: *Input point cloud 'Layer/Time 1' and 'Layer/Time 2' (left and right) in 3D render mode 'Height'.*

From now on, all further algorithms are applied to this resolution and you have to change to this map to see the results of all processes.

3.2.2 Find changes & create elevation layers

The algorithms in the next section of the rule set create elevation layers for the two acquisition times (Time1 (before) and Time2 (after) based on the point cloud data. The algorithm assigns the maximum elevation value from the point cloud to the pixel value, representing vegetation and building edges better than calculation the mean value:

- The algorithm “rasterize point cloud” writes the elevation maximum (z-coordinate) from point cloud **Layer 1** to a new raster layer **'ElevationMax_before'**
- Then, algorithm “rasterize point cloud writes” the elevation maximum (z-coordinate) from point cloud **Layer 2** to a new raster layer **'ElevationMax_after'**

To visualize the new layers, split your view (Window > Split) and for the **50cmMap** select the two new layers in **View > Image Layer Mixing**.

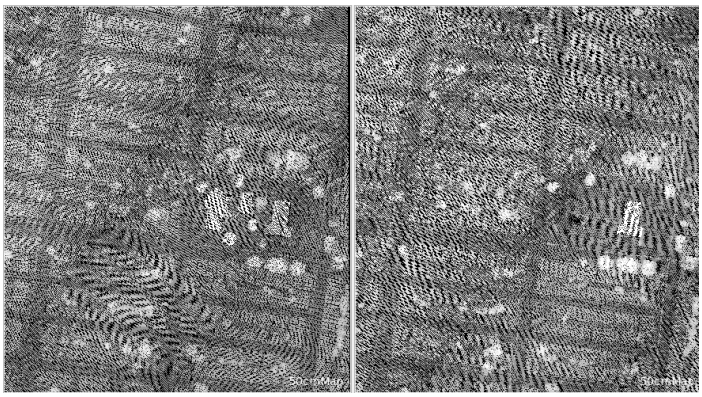


Figure 3.2: *Two new image layers ElevationMax_before (left) and ElevationMax_after (right)*

Fill gaps using a median filter

Based on the two new raster layers, gaps between pixels have to be filled in the next step with valid elevation values from the point cloud. The approach is suited to fill small gaps only.
Time 1:

- In a first segmentation the" multi-threshold segmentation" creates a new level based on **ElevationMax_before***,* where all pixels are segmented to a large object and classified as **class '_tempClass01'** with an elev ≠ 0. All pixels with a value = 0 are assigned to 'unclassified' (Note that there might be data with negative elevations. Because this segmentation does not allow to insert the value < 0 and > 0, a workaround is used for the lower border of the interval set to _tempClass01 <= -0.0001)
- Now a "median filter" is applied to this object level to interpolate the gaps using the class _tempClass01, where all unclassified objects are filtered using a kernel size of 3. The resulting values are written into the raster layer **'ElevationMax_before'** again.
- Because the object level is not needed anymore it is deleted ('New Level').

The same steps described above are applied to time 2 resulting in an interpolated raster layer **'ElevationMax_after'**.

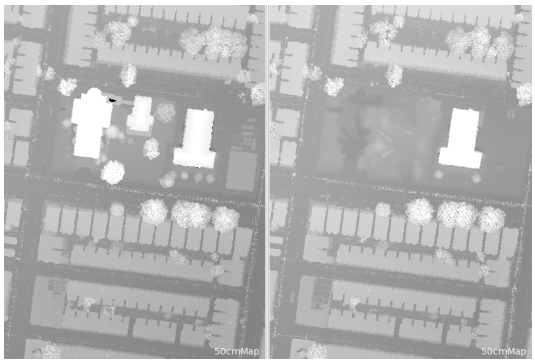


Figure 3.3: *Interpolated new image layers ElevationMax_before (left) and ElevationMax_after (right)*

3.2.3 Find negative changes & disappeared features

- Using the algorithm "layer arithmetics" the difference between 'ElevationMax_after' and 'ElevationMax_before' is calculated resulting in layer **ElevationDiff**.

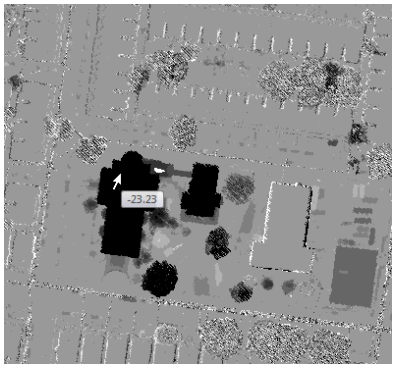


Figure 3.4: *Elevation difference layer* 'ElevationMax_after' -'ElevationMax_before' - *resulting in negative values for disappeared features*

Areas with higher elevation in time 1 than in time 2 thus resulting in negative values are most likely features that have *disappeared*.

- Again, a “multi-threshold segmentation” is applied and a 'New Level' is created where all pixels with a value <= -2 in image layer **ElevationDiff** are assigned to the class **negative_change**.

If you select the view classification button you can see that there are a lot of small image objects that are not classified yet. They are removed in the next set of algorithms.

Reshape - eliminate too small:

- The algorithm “pixel-based object resizing” is applied to New Level - with number of cycles set to 2: all unclassified objects **larger than 100 pixels** (Domain > Condition > Number of pixels > 100) are assigned to _tempClass01 using the coating mode. As a result the seed class negative_change (Candidate Object Domain > Class filter) is surrounded by _tempClass01. The goal is to keep only large compact objects.

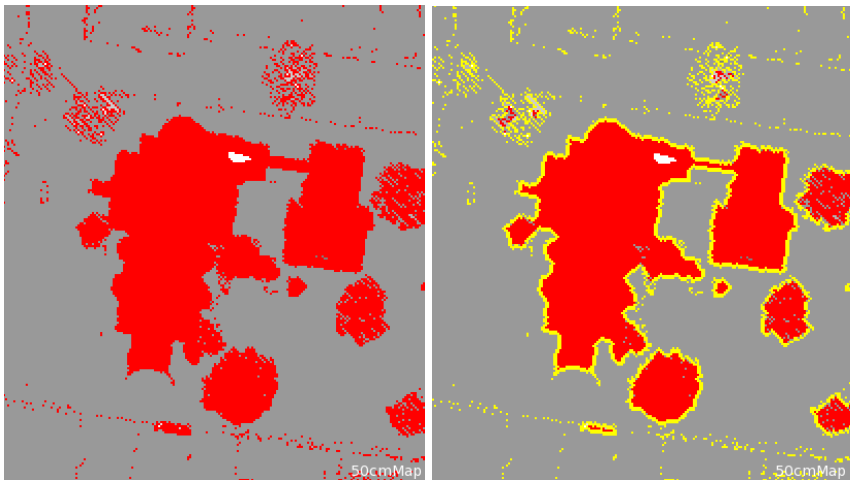


Figure 3.5: *Classified objects expected to be disappeared (red) coated by reshaping class (right, yellow)*

- The next algorithm applies “pixel-based object resizing” using the growing mode. This time, based on 5 cycles all negative_change objects grow into the _tempClass1 objects.

3.2.4 Classify changes - assign areas to vegetation and man-made

Create a raster layer with the elevation Standard Deviation (only within the areas classified as change)

- The algorithm “create temporary point cloud” produces a new point cloud only for **class negative_change**

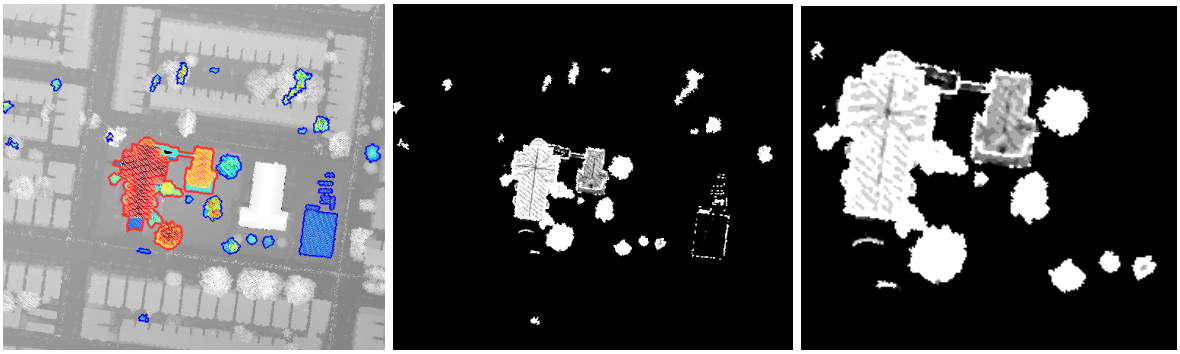


Figure 3.6: *New point cloud _tempPC01 created within negative changes (left) and its rasterized standard deviation for the elevation (middle). Applied median filter to ElevationStdDev (right)*

- In the next step the new point cloud *_tempPC01* is rasterized based on the standard deviation for the elevation. Resulting image layer: **ElevationStdDev**
- Now that the information needed is extracted to the new raster layer ElevationStdDev, the point cloud *_tempPC01* can be deleted.
- The “median filter” overwrites the image layer ElevationStdDev reducing noise in the image.

Classify areas with high Standard Deviation values (rough areas) as disappeared vegetation

- Rough areas, having a high standard deviation for the elevation, are assumed to correspond to vegetation, thus the “multi-threshold segmentation” divides the image ElevationStdDev in the two classes disappeared_vegetation (StdDev value > 0.3) and disappeared_man-Made (StdDev value <= 0.3). The algorithm is applied within the class negative_change only.

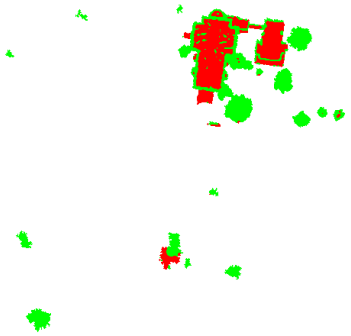


Figure 3.7: *Classes disappeared_vegetation (green) and disappeared_man-Made (red)*

Improve man-made

- In 4 cycles large man-made objects - representing most probably buildings - are grown to merge with their edges:
    - The “pixel-based object resizing” uses the **growing** mode for the class disappeared_man-Made with n**umber of pixels > 500** and grows into the class disappeared_vegetation.
    - Subsequently all man-Made objects are **fused**.
- The remove objects algorithm finally **merges** smaller disappeared_man-Made objects (< 20 pixels) into disappeared_vegetation or unclassified objects, depending on neighbor image objects with the largest common border.

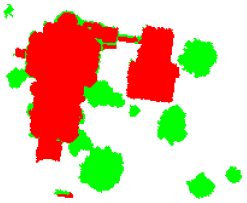


Figure 3.8: *Improved man-made classes disappeared_vegetation (green) and disappeared_man-Made (red)*

Improve vegetation

- During this improvement step only vegetation objects that are large and compact are kept, while non-compact vegetation objects are removed using a "shrink and grow" approach:
    - In 2 cycles disappeared_vegetation **shrinks** into _tempClass01 using “pixel-based object resizing”.
    - As a next step all small disappeared_vegetation objects (< 20 pixels) are also **assigned** to the class _tempClass01.

    - In 4 cycles disappeared_vegetation is **grown** into the _tempClass01.
- The _tempClass01 class is merged using the “merge region” algorithm to achieve meaningful objects.
- The objects are now assigned to disappeared_man-Made or disappeared_vegetation dependent on neighbor image objects with the largest common border.
- Finally, the classification of _tempClass01 objects is changed to unclassified.

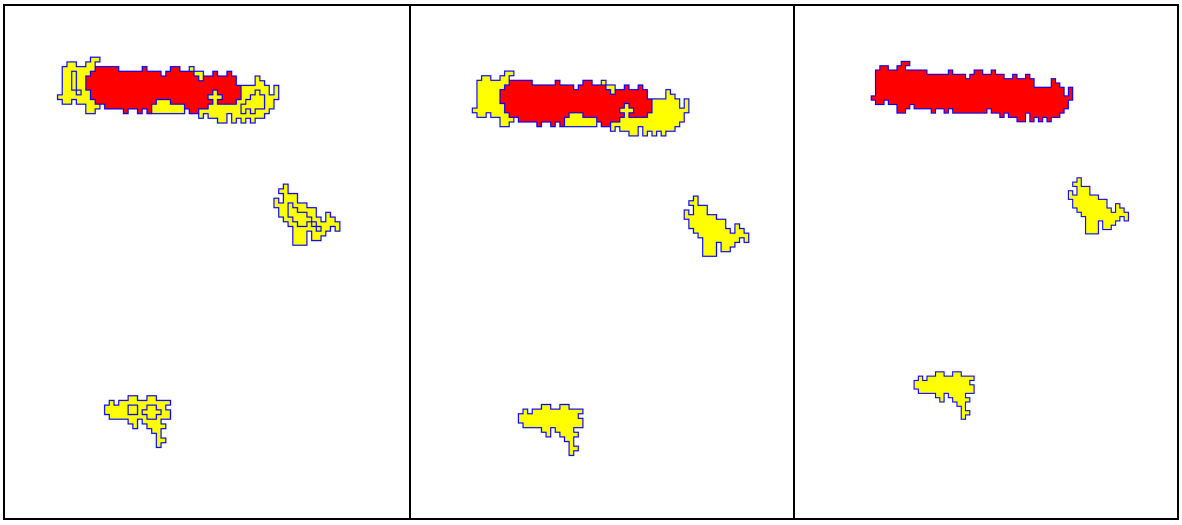


Figure 3.9: *Small objects _tempClass01 (left), merged (middle) and assigned to neighbor objects with the largest common border (right).*

Merge all

- Here, the algorithm “convert image objects” is applied:
    - All image objects in the image object domain are changed to disconnected image objects. The algorithm creates one large image object per class.
      Note - In case single, small objects are needed for further analysis, the algorithm **“merge region”** should be used here instead. To fuse the classes, several merge region algorithms are needed. The convert image objects algorithm solves the fusion using one algorithm, but combines all objects. (In case you have fused the objects using convert image objects - the objects could be separated again using the algorithm **convert image objects** again but in the ‘Connected 2D’ mode: all image objects in the image object domain are changed to separate image objects.)

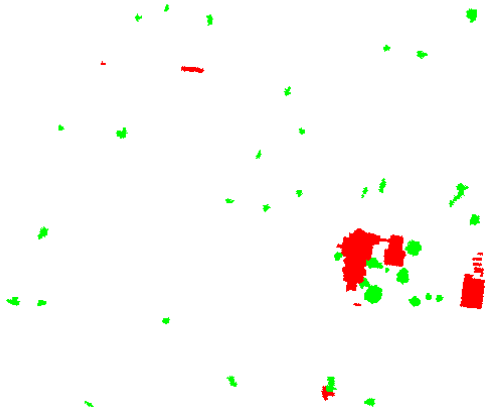


Figure 3.10: *Final classification result for disappeared_vegetation (green) and disappeared_man-Made (red)*

3.2.5 Create a point cloud with the changes only

In this last step of the rule set the final classification is assigned to a point cloud, that contains changes only:

- For all objects classified as negative_change a “temporary point cloud” is created based on the Layer 1 point cloud: 'ChangesPC_disappeared'
- For this new point cloud the class disappeared_man-Made is **assigned** to the point cloud class '6 - Building' to point cloud
- And finally, disappeared_vegetation is assigned to the class **'5 - High Vegetation'**

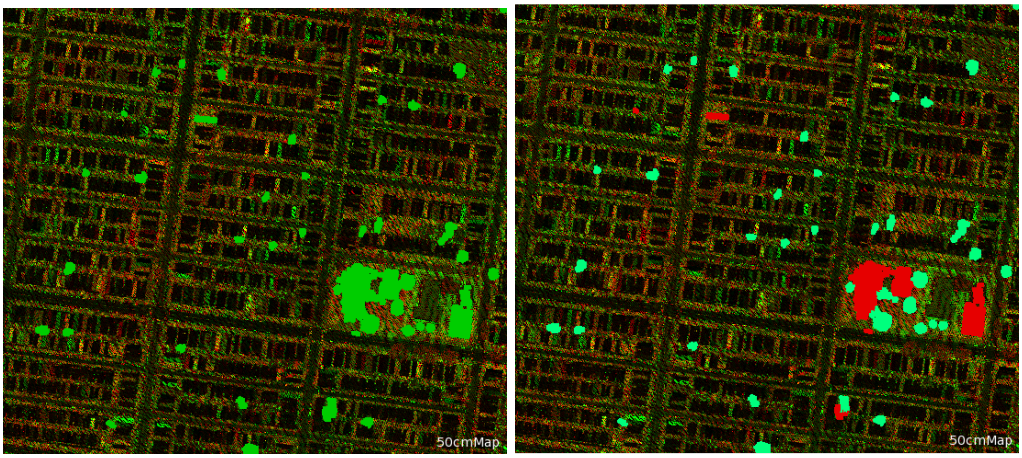


Figure 3.11: *Temporary point cloud 'ChangesPC_disappeared' created for class negative_change (left) and assignment of classes disappeared_man-Made and disappeared_vegetation (right).*

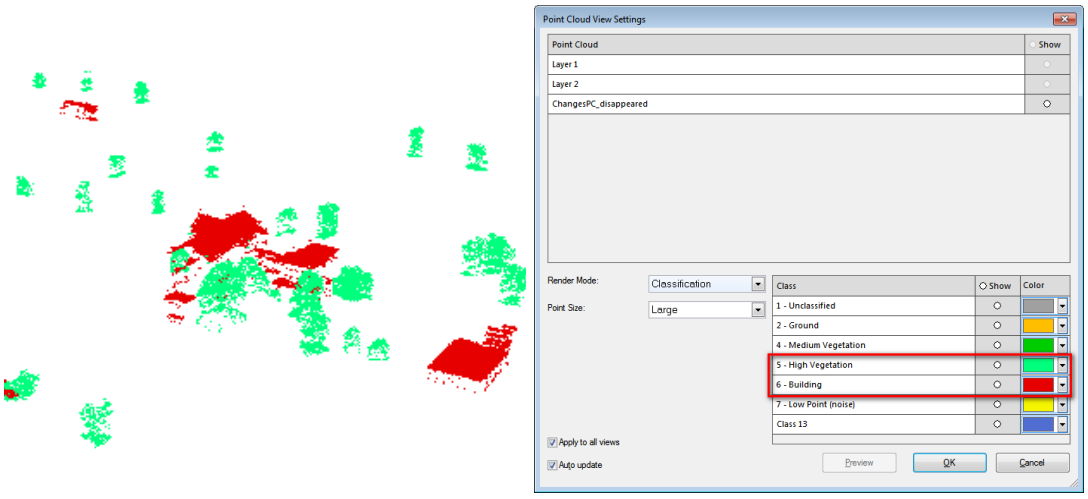


Figure 3.12: *Detail of temporary point cloud 'ChangesPC_disappeared' (left) with additional point cloud classes 'High vegetation' and 'Building' in point cloud view settings dialog (right).*

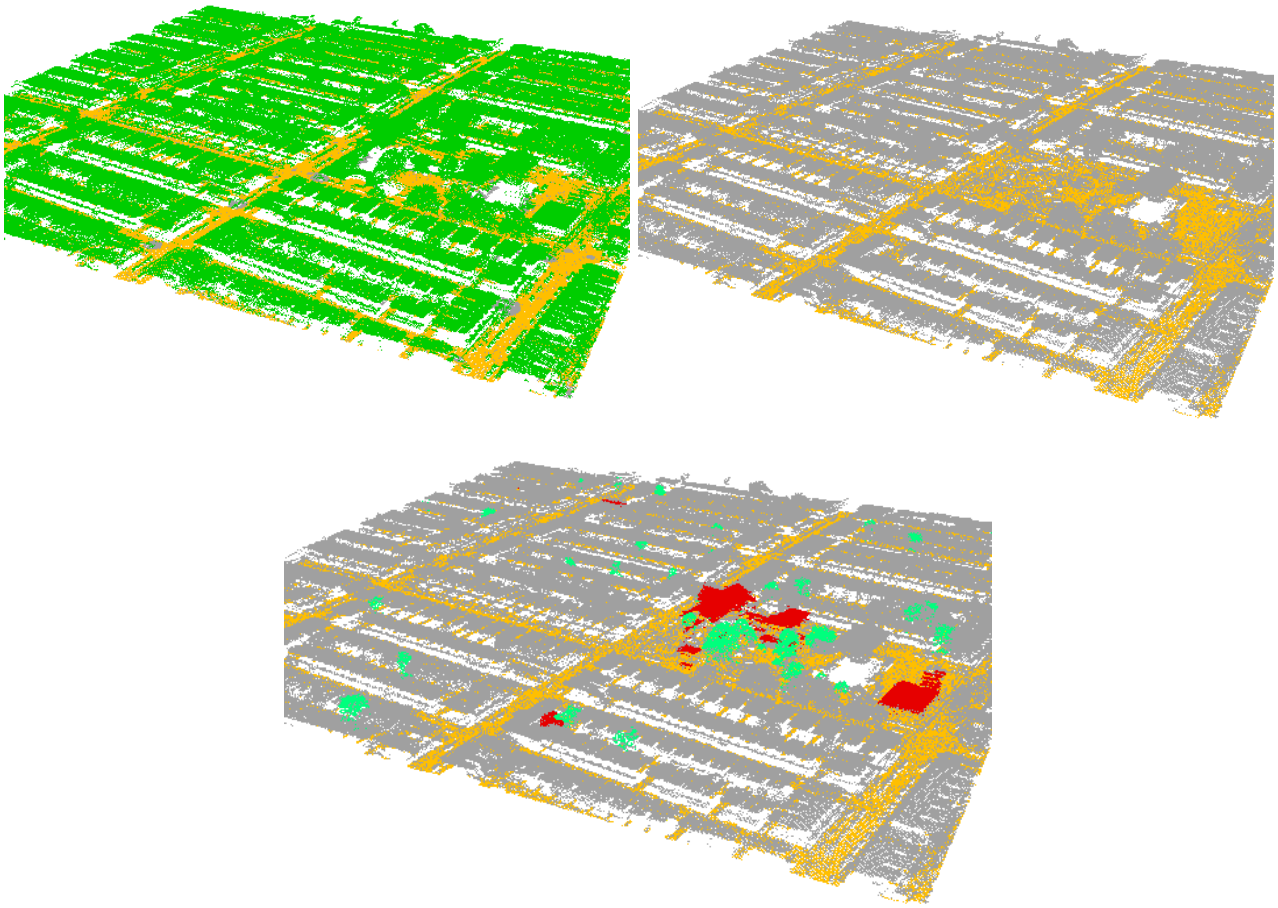


Figure 3.13: *Input point cloud 'Layer 1' and 'Layer 2' (upper left and right) and visualization of temporary point cloud 'ChangesPC_disappeared' together with Layer 2 (below).*

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