PlantScan3d Documentation

Release 1.5

Frederic Boudon, Julien Benoit

Oct 22, 2020

Contents

Version 1.1.0

Release 1.5

Date Oct 22, 2020

This manual details functions, modules, and objects included in PlantScan3D, describing what they are and what they do.

Warning: This "User Guide" is still very much in progress. Many aspects of VPlants.treeeditor3D are not covered.

More documentation can be found on the openalea wiki.

CHAPTER 1

User guide

This is the documentation of how to use Plantscan3d.

1.1 Install and run

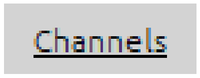
This part explain how to install Plantscan3d.

1.1.1 Anaconda

OpenAlea is distributed through Anaconda. So first we have to install it. All information for installation is available from the Anaconda documentation website. Then, we will use Anaconda navigator, a GUI for managing Anaconda environments.

1.1.2 Environment

Open Anaconda navigator. This can take a while (10 seconds to 1 minute). Then, go to "Environments". Click on the `Create` button :

Name it `openalea` and create it. It will create an isolated environment with an installation of python (*e.g.* Python 3.7). We now have to import the different packages for plantscan3d. The packages are bundled in two channels, the one from conda-forge, and the one from Frédéric Boudon. Add a channel by clicking on the channels button: 

Then, click on `Add...`, and add these two channels (just copy/paste it as is):

- https://anaconda.org/fredboudon
- conda-forge

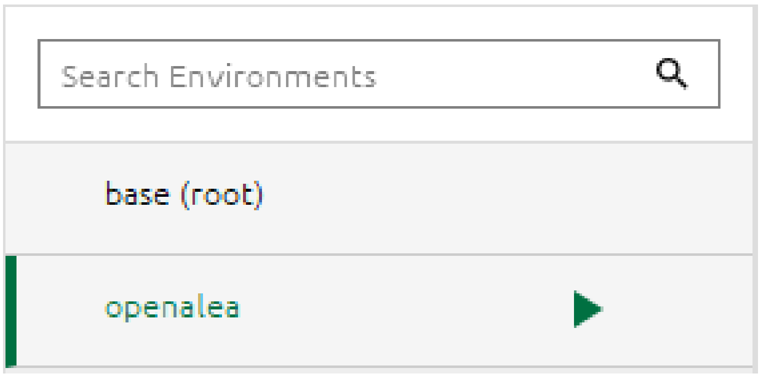
Click on update index:

Then click on the ribbon named `Installed`, and replace by `Not Installed`. Search plantscan3d, check the box and click on apply (twice). The package will be installed along with all its dependencies.

And that's it !

1.1.3 Run

To run Plantscan3D, open your openalea environment, and click on the play button: ▶ which is located on the "environments" panel:



Choose "open a terminal", and type the following command:

```
plantscan3D
```

1.2 Point cloud Treatment

This part explain how to manipulate a point cloud with Plantscan3d's tools.

1.2.1 Basic usage

This part explain how to use basic commands of Plantscan3d.

Import

Point cloud

You can import a point cloud from a LiDAR scan using `File -> Import Points`, or `ctrl+P`. Here is an example point cloud of a branch that we can imported:

You can download the example file here: `Branch point cloud`.

Topology

You can import the plant toology by importing an mtg file using `File -> Open MTG`, or `ctrl+O`. Here is an example MTG based on the previous point cloud:

You can download the example file here: `Branch MTG`.

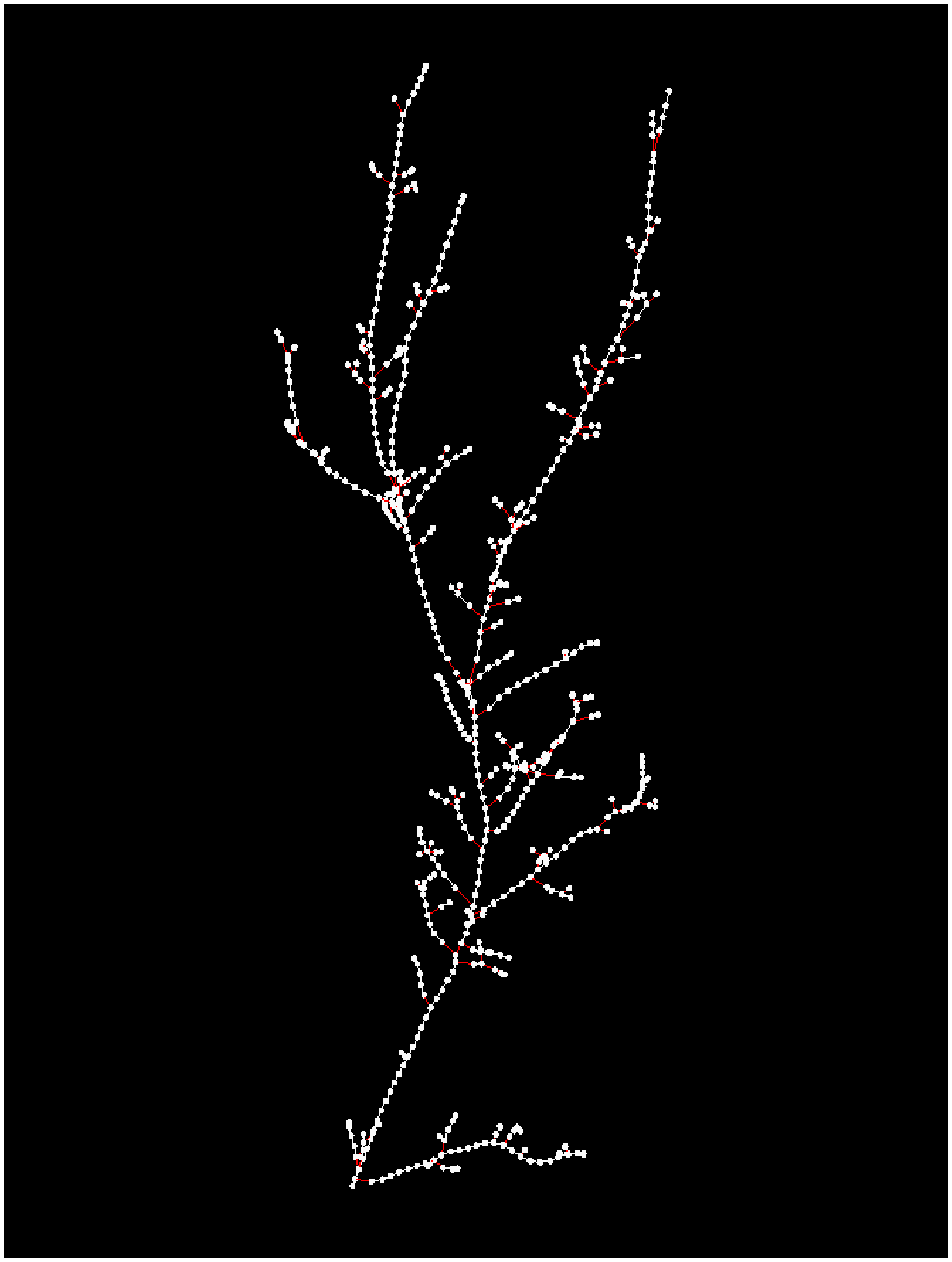
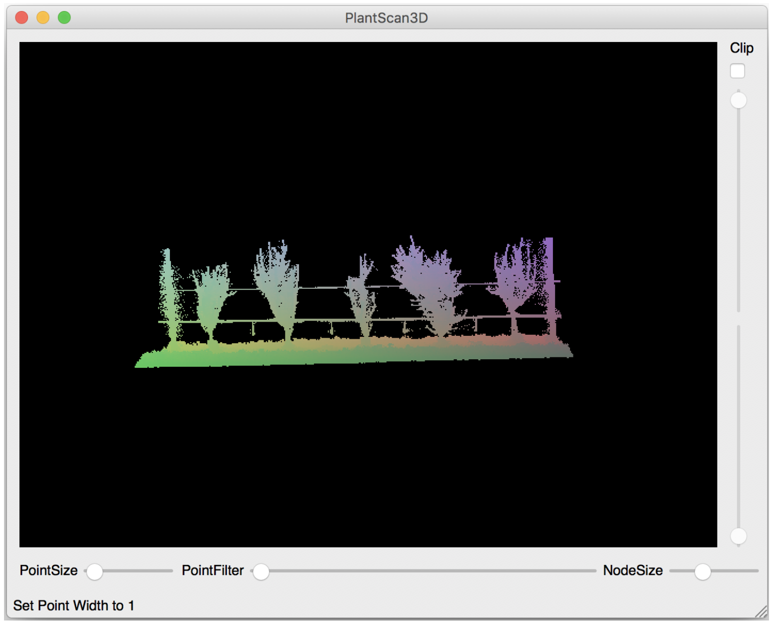


Fig. 2: Topology

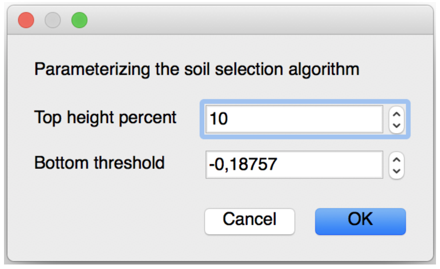
1.2.2 Cleaning Point cloud

This section explain how to use the tools to clean a point cloud and segment it. All algorithms not delete directly the points but select its with the selection system of PlanScan3d (see also: *Anaconda*).



Soil Selection

The goal of this algorithm is to select the points that belongs to the soil and not keep the weed that do up to the soil (**Points -> Selection -> Soil**). To do that, we select a percent of points that are on the top of the scan (see below: the first parameter) and we go down through the point's neighborhood. For each point, we check its height and test if it is below a threshold (see below: the second parameter).

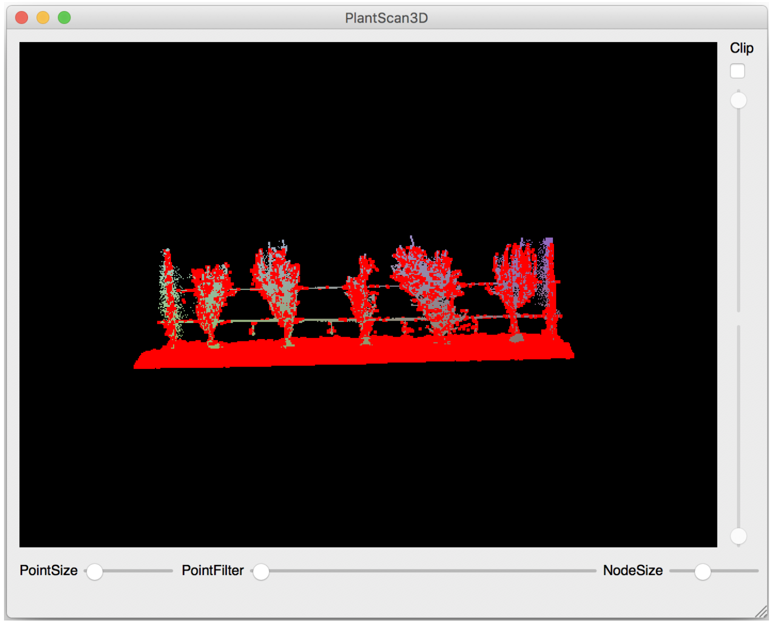


Parameterizing the soil selection algorithm

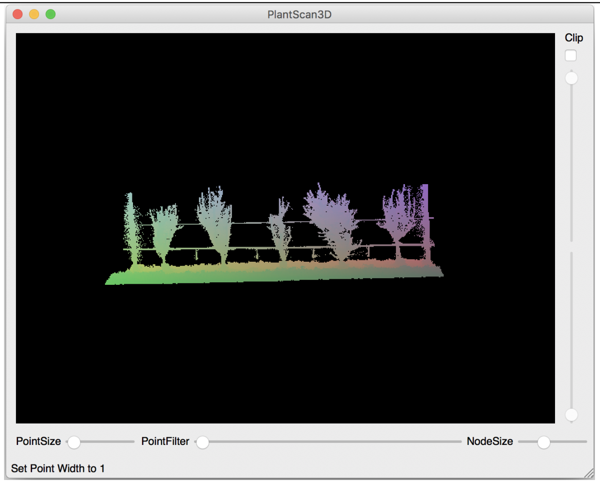
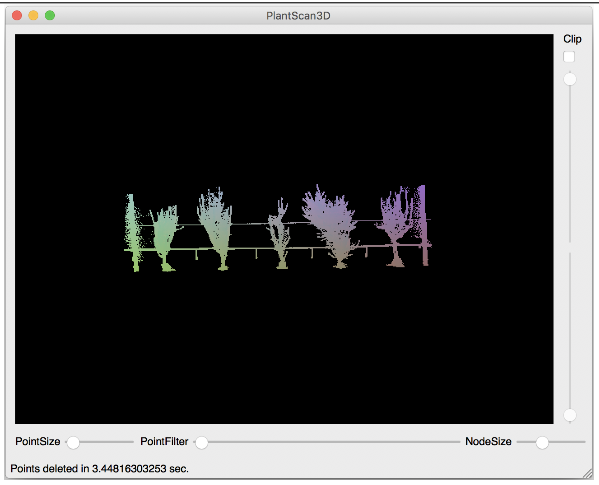
Top height percent	10
Bottom threshold	-0,18757

The parameters of this algorithm are :

- Top height percent : this parameter is the percent of the height of the scan that will be taken as start points. The default value is 10% but you can change it if you want but not exceed 70% because it's possible that the algorithm will keep the weed as start.
- Bottom threshold : this parameter control the height were the algorithm will stop go down, this parameter is estimated with the barycenter of the scan. We recommend to keep the value that is calculated.



You can notice that a few points above the soil are select, this is normal because there is isolated points.

Before	After
 Set Point Width to 1	 Points deleted in 3.44816303253 sec.

You can also test this algorithm in a python script (See: `point cloud used of this example`)

```
from openalea.plantgl.all import *

scene = Scene('winter.ply')
points = scene[0].geometry.pointList
topPercent = 10
minZ = points[points.getZMinIndex()].z
bottomThreshold = minZ + (points.getCenter().z - minZ) * 0.5

soil_indexes = select_soil(points, IndexArray(0), topPercent, bottomThreshold)

soil_points, other_points = points.split_subset(soil_indexes)
shape1 = Shape(PointSet(soil_points), Material('red', Color3.RED))
shape2 = Shape(PointSet(other_points), Material('blue', Color3.BLUE))
```

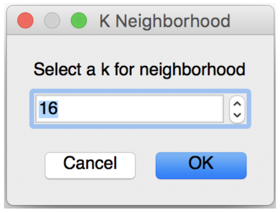
(continues on next page)

(continued from previous page)

```
Viewer.display(Scene([shape1, shape2]))
```

Wires Selection

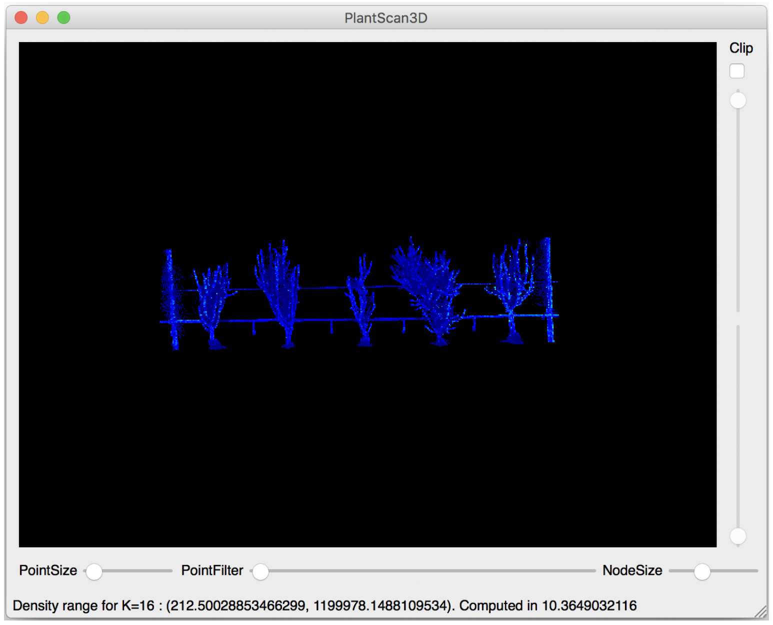
The second step of the treatment pipeline is the clean of the wires. The first things to do is to remove noise of the LIDAR, we calculate the densities of all points according of there neighborhood (**Points -> Density -> K-Density**).



Select a k for neighborhood: 16

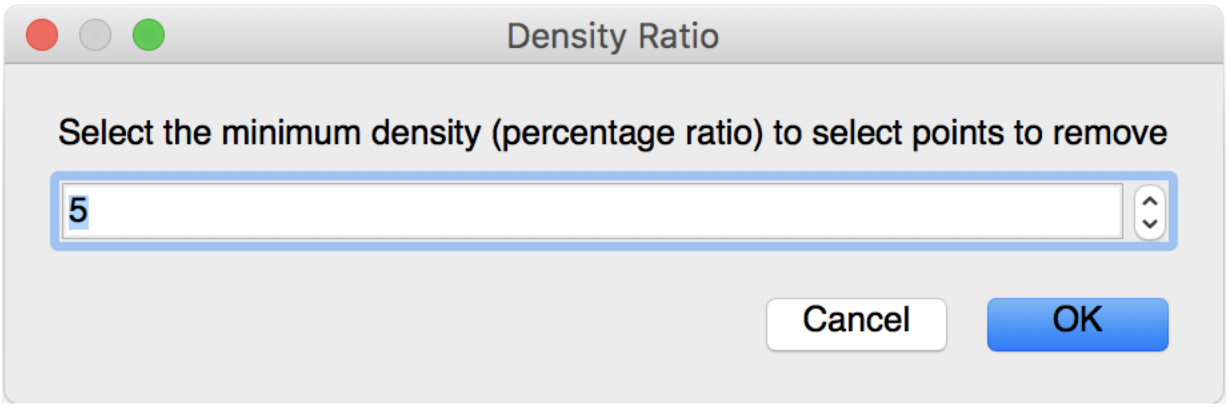
The parameter of the k density is:

- k is the number of neighborhood that is calculate for one point. We recommend to use the default number (16 neighborhood).

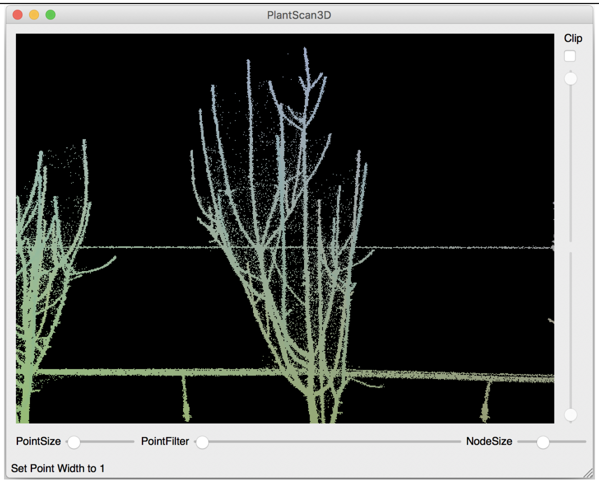
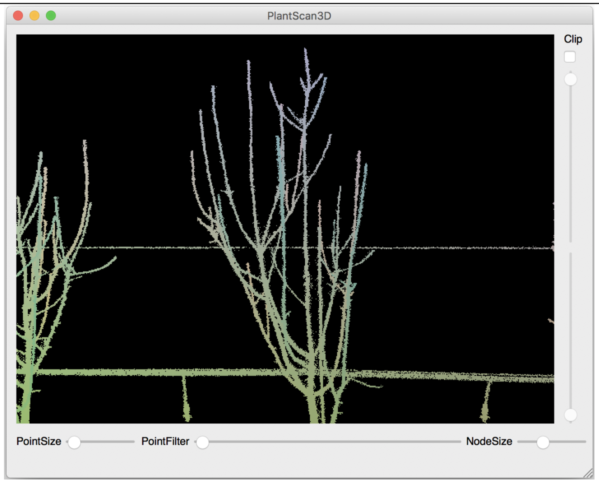


Density range for K=16 : (212.50028853466299, 1199978.1488109534). Computed in 10.3649032116

The next step is to remove the points that have a low density, so we use the filter min algorithm (**Points -> Filter -> Filter Min Density**).


Density Ratio
Select the minimum density (percentage ratio) to select points to remove
5
Cancel
OK

The single parameter of this algorithm is the percent of the low densities that will be deleted. The default value is 5 percent but I recommend to set 3 or 2 percent because 5 percent could delete too much points and the algorithm to select the wire could fail.

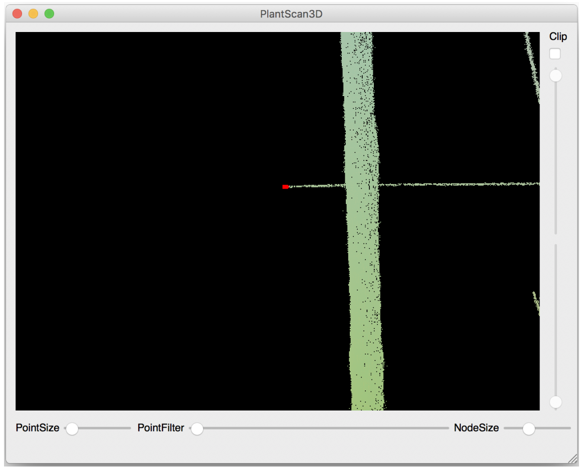
Before	After
	

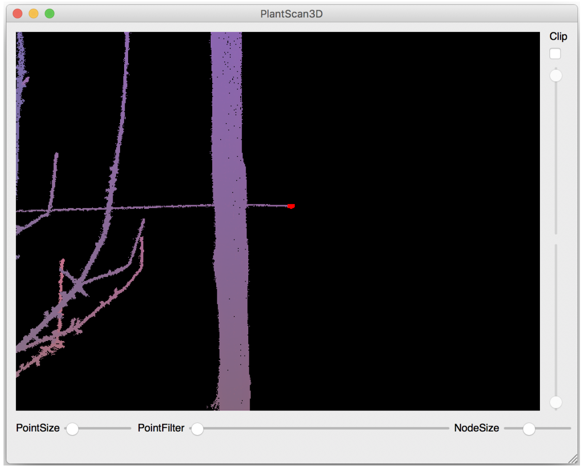
You can also test this algorithm in a python script (See: `point cloud used of this example`)

```
from openalea.plantgl.all import *

scene = Scene('winter_step_01.ply')
points = scene[0].geometry.pointList
k = 16
densityratio = 3

kclosests = k_closest_points_from_ann(points, k)
densities = densities_from_k_neighborhood(points, kclosests, k)

filter_indexes = filter_min_densities(densities, densityratio)

isolate_points, other_points = points.split_subset(filter_indexes)
shape1 = Shape(PointSet(isolate_points), Material('red', Color3.RED))
shape2 = Shape(PointSet(other_points), Material('blue', Color3.BLUE))

Viewer.display(Scene([shape1, shape2]))
```

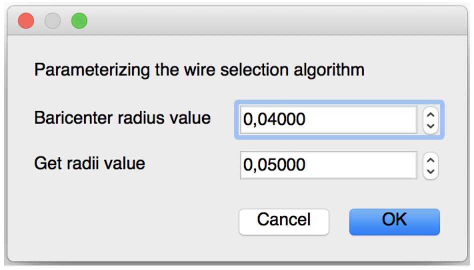
The next step is to select two extremities of the wire with the selection tool and validate the selection with the action

(**Points -> Selection -> Use selection for wire algorithm**).





Next start the wire algorithm (Points -> Selection -> Wire). The algorithm calculate the shortest path between the extremities passing to the neighborhood and add a barycenter for each points of this path according to their neighborhood (see below: the first parameter). Next the algorithm estimate radii of the wire for each barycenters (see below: the second parameter) and take the points that is inside a cylinder between two barycenters with the radius.

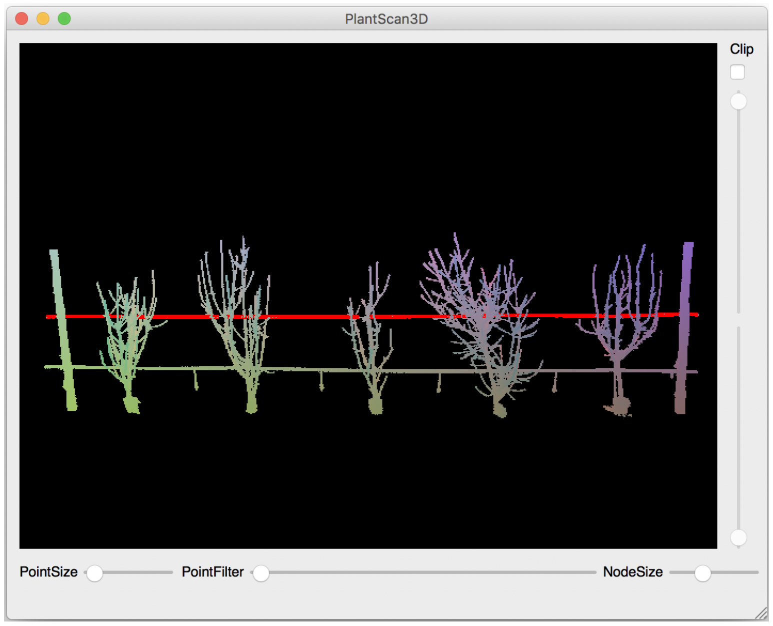


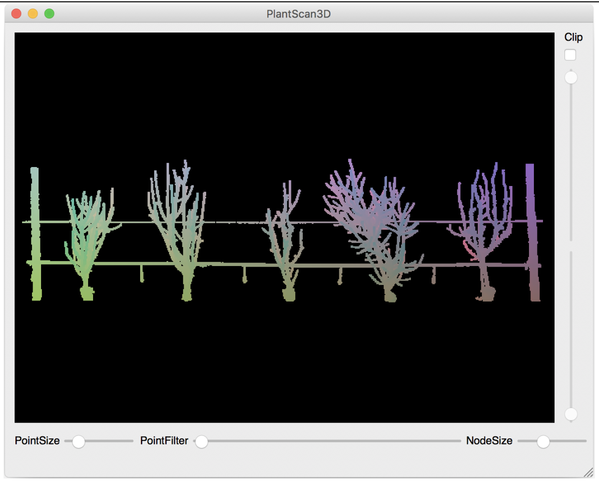
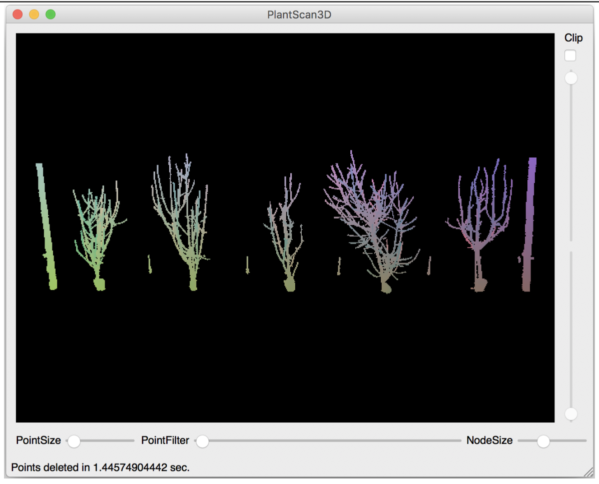
Parameterizing the wire selection algorithm

Baricenter radius value	0,04000
Get radii value	0,05000

The parameters of this algorithm are:

- Barycenter radius value : this value is the radius of the neighborhood for each point on the path.
- Get radii value : this is the radius of the neighborhood for each barycenters.



Before	After
	

Points deleted in 1.44574904442 sec.

You can also test this algorithm in a python script (See: `point cloud used of this example`)

```python
from openalea.plantgl.all import *
import numpy

scene = Scene('winter_step_02.ply')
points = scene[0].geometry.pointList
Ymin, Ymax = points.getYMinAndMaxIndex()
bariRadius = 0.04
radiiValue = 0.05

kclosest_wire = IndexArray(0)
wire_path = get_shortest_path(points, kclosest_wire, Ymin, Ymax)
newpoint, baricenters = add_baricenter_points_of_path(points, kclosest_wire, wire_
↪path, bariRadius)
```

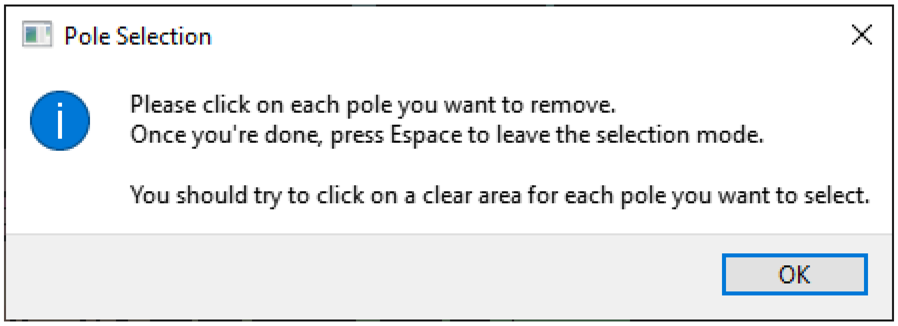
(continues on next page)

(continued from previous page)

```python
kclosest = k_closest_points_from_ann(newpoint, 20, True)

radii = get_radii_of_path(newpoint, kclosest, baricenters, radiiValue)
radius = numpy.average(radii)

indexes = select_wire_from_path(newpoint, baricenters, radius, radii)
wire_indexes = Index([])

for i in indexes:
    if i not in baricenters:
        wire_indexes.append(i)

wire_points, other_points = points.split_subset(wire_indexes)
shape1 = Shape(PointSet(wire_points), Material('red', Color3.RED))
shape2 = Shape(PointSet(other_points), Material('blue', Color3.BLUE))

Viewer.display(Scene([shape1, shape2]))
```

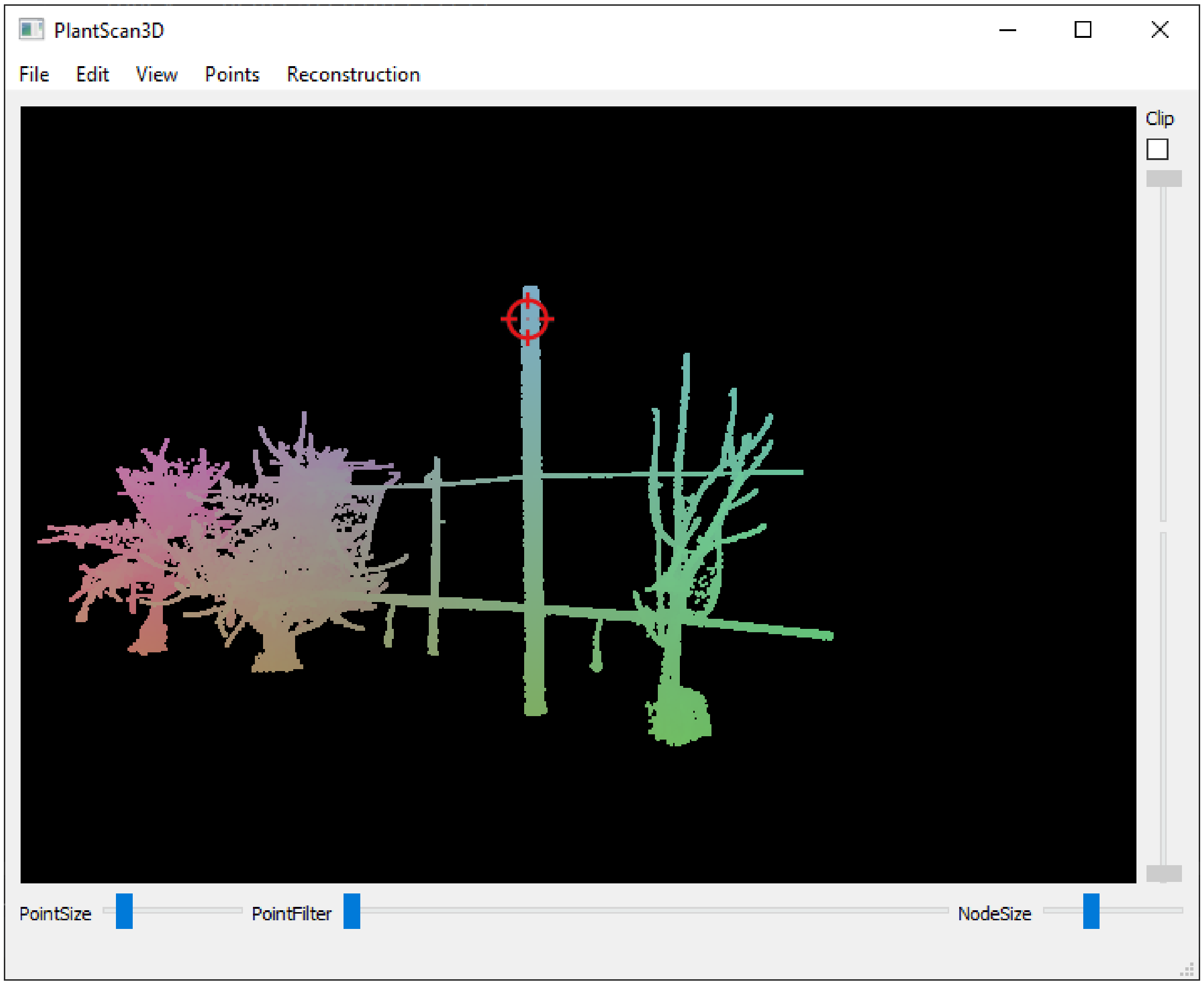
Pole Selection

The pole selection (**Points -> Selection -> Pole**) is based on the RANSAC algorithm. Using the graphical interface, you will be asked the click on each pole you want to select. When you're done, press Escape to leave the selection mode. Selecting a point allows the RANSAC algortihm to concentrate on a specific area of the point cloud, therefore significantly decreasing the computation time.

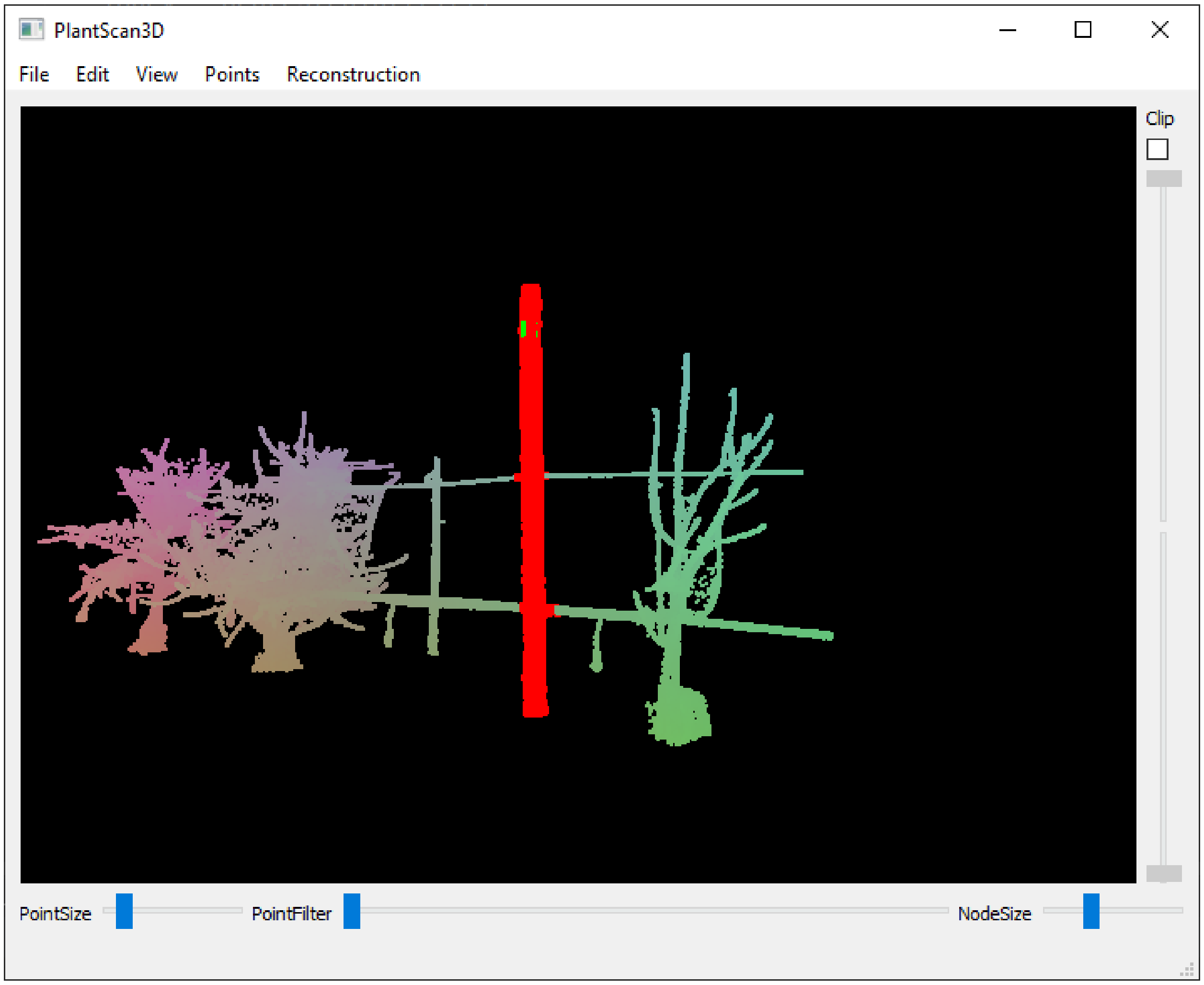

Pole Selection

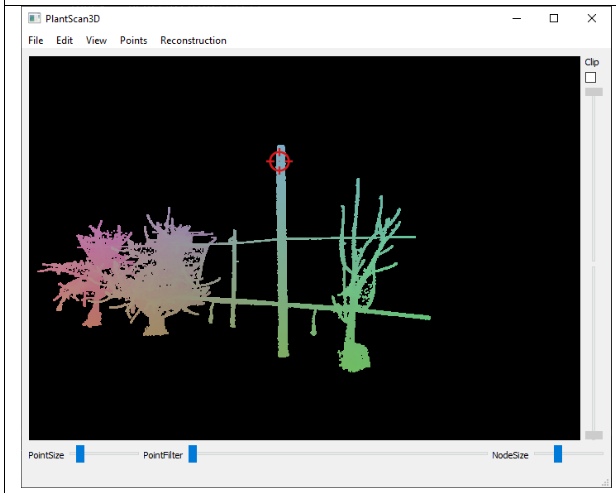
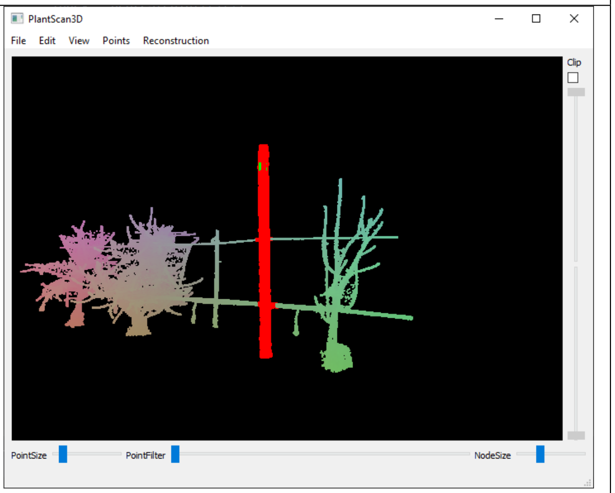
Please click on each pole you want to remove.
Once you're done, press Espace to leave the selection mode.

You should try to click on a clear area for each pole you want to select.



This algorithm constructs cylinders between two randomly chosen points. Each of these cylinders must pass through the previously selected point. Then, it counts the number of points inside and compute a score with it. This step is repeated 10.000 times. Finally, the algorithm selects the cylinder with the best score.



Before	After
	

You can also use this algorithm in a Python script (See: `point cloud used of this example`)

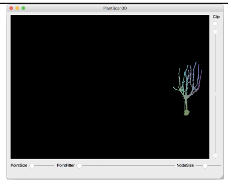
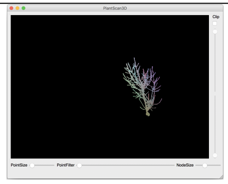
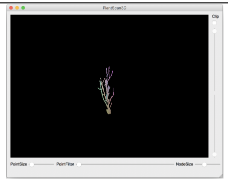
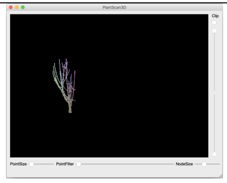
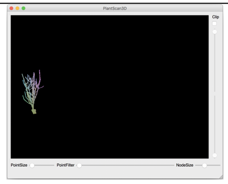
The parameters are:

- The radius of the cylinder created by the Ransac.
- The percent of tolerance inside and outside the cylinder to take the points.
- The number of cylinders generated before take the best one. This parameters impact the processing time.

```
from openalea.plantgl.all import *

scene = Scene('winter_step_03.ply')
points = scene[0].geometry.pointList
poleRadius = 0.175
tolerance = float(60) / 100
iteration = 10000

pole_indexes, score = select_pole_points_mt(self.points.pointList, poleRadius,␣
↪iteration, tolerance)
print 'score = ' + str(score)

pole_points, other_points = points.split_subset(pole_indexes)
shape1 = Shape(PointSet(pole_points), Material('red', Color3.RED))
shape2 = Shape(PointSet(other_points), Material('blue', Color3.BLUE))

Viewer.display(Scene([shape1, shape2]))
```

Point cloud Segmentation

Here we segment the point cloud to get the trees separately (**Points -> Segment**). You can change to the next tree the next action (**Points -> Next Segmented Tree**). This algorithm is only base on the connex components so it not efficient.

Tree1	Tree2	Tree3	Tree4	Tree5
				

You can also test this algorithm in a python script (See: `point cloud used of this example`)

```
from openalea.plantgl.all import *
import random

scene = Scene('winter_step_04.ply')
points = scene[0].geometry.pointList

kclosest = k_closest_points_from_ann(points, 10, True)
connexsIndex = get_all_connex_components(points, kclosest)

connexPoints = []  # type: List[Point3Array]
for c in connexsIndex:
    if len(c) < 10000:
        continue
    connexPoints.append(points.split_subset(c)[0])

mats = []  # type: List[pglsg.Material]
```

(continues on next page)

(continued from previous page)

```
while len(mats) != len(connexPoints):
    r = random.randrange(0, 256)
    g = random.randrange(0, 256)
    b = random.randrange(0, 256)
    color = Color3(r, g, b)
    inmats = False
    for m in mats:
        if m.ambient == color:
            inmats = True
            break
    if not inmats:
        mats.append(Material("mat" + str(len(mats)), color))

shapes = []  # type: List[pglsg.Shape]
for i in range(len(connexPoints)):
    shapes.append(Shape(PointSet(connexPoints[i]), mats[i]))

Viewer.display(Scene(shapes))
```

1.2.3 Reconstruction

This section is a step-by-step guide for the process of the reconstruction of the topology of a plant from a point cloud.

The guide use a 3d point cloud of a branch scanned using a Riegl VZ400 LiDAR as an example. It is available for download `here`. The branch looks as follows after import in plantscan3d:

Points contraction

A first step before reconstructing the topology is to contract the points around the main axis. To do so, click on `Points->Contraction`, and choose a contraction algorithm in the list. The Euclidean algorithm is simple but efficient, so it can be used to contract the points very quickly. The algorithm giving best results is the algorithm of Riemannian, which is longer but more precise.

Here is an example result using the algorithm of Riemannian:

We can see that the points were contracted along the main axis. This is a key step because it will help the reconstruction algorithm. The objective here is to contract the points to reduce the noise while keeping small structures (a high contraction deforms the structure).

Topology reconstruction

Add the root

First, we have to place the first point at the "root" of the structure: `Reconstruction->Add Root->Bottom`. It can be the base of a trunk, or a branch if the point cloud is an isolated branch. You can then move the point so it is close to the root of the structure. Then, use `right-click->T`, or more simply select the point and press `T`. It will stick the node to the nearest points.

Skeletization

The reconstruction of the topology is made using `Reconstruction->Skeletization`. Choose an algorithm in the list. It is recommended to use the algorithm from Xu et al. Then, enter the number of nodes required from the root

until the top of the structure. For example 50 is a good approximation for a branch.

Now save the resulting MTG `File->Save MTG`, or press `ctrl+S`. Please save frequently in case the application crash.

Display helpers

The window in the top right corner (Display) proposes 3 helper sliders for visualisation:

- `Point Size` to change the size of the LiDAR points

Fig. 5: Point size

- `Point Filter` to filter the LiDAR points that are far from the skeleton. It is used to point out potential reconstruction errors, where some structure is potentially not reconstructed:

Fig. 6: Filter points

- `Node Size`, to control the size of the skeleton nodes:

Fig. 7: Node size

Simple corrections

It is possible to correct the reconstruction by moving, deleting and adding nodes, or by changing the link between nodes (parent and child).

- Moving a node: click on a point and drag it. Change your viewpoint to check if its position is right for all axes. Then, it can be useful to press `T` to "stick" the node to the nearest points, *i.e.* move it to the barycentre of the local cluster of points.
- Add a new node: `N`;
- Delete a node: `right-click->Remove node`;
- Add a new node between two nodes: `E`;
- Correct the parent of a node (wrong connection): select the node, press `P`, and select the new parent;
- Correct the relationship between two nodes:

A node can either be part of the axis than the previous (it is following in MTG terms), or branching. The relationship is denoted by the color of the segment between the nodes: white for following, red for branching. It can be changed using `right-click->Set Branching Points` or `right-click->Set Axial Points` (or `M`).

Note: it is often preferable to re-use the original LiDAR point cloud to correct the skeleton instead of the contracted one. To do so, just re-import the point cloud. It will simply replace the previous one.

Sub-tree corrections

Sometimes a whole sub-tree need reconstruction. In that case it is possible to delete a sub-tree and to force its reconstruction:

Fig. 8: Work with sub-trees

Smoothing the MTG

The MTG can be smoothed spatially using `Reconstruction->Skeleton->Smooth`

Geometry estimation

The 3D geometry of the tree can be computed using an estimation of the diameter of each node. The first step is to re-import the original LiDAR point cloud to get the right dimensions: `ctrl+P`.

> Please control first that your skeleton reconstruction is good enough compared to the point cloud because it will have a great influence on the estimation of the nodes diameters.

Several algorithms are provided to get the node diameter:

- Using the maximum point distance. This can lead to an overestimation of the diameter;
- Using the average point distance;
- Using the pipe model: only the maximum diameter at the base and at the tip are required:

Then, the diameters can be filtered and smoothed to avoid extreme values and increasing diameters.

The diameter of each node can be controlled by searching through its properties: `right-click->Properties`, and scroll until the radius property.

Finally, a 3D representation can be computed using the diameters and the length of each segment:

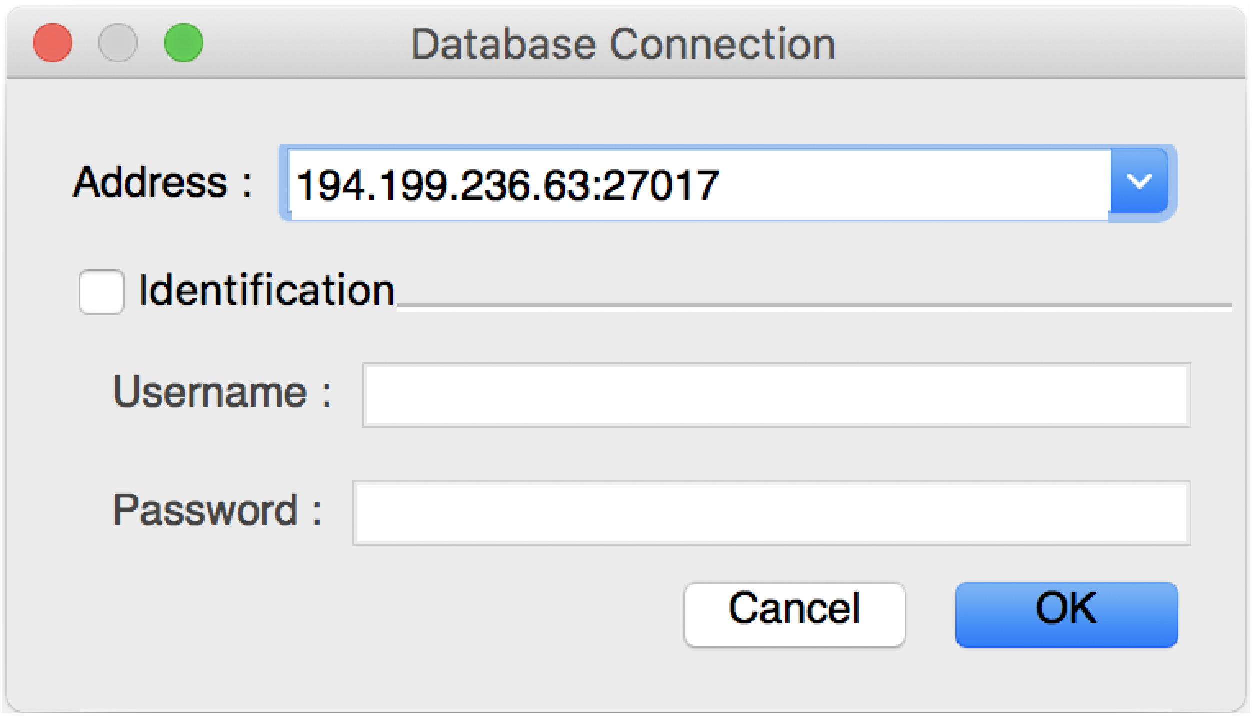
Save and export

The skeleton can be saved as a Multi-scale Tree Graph (MTG). It is highly recommended to save it as often as possible, especially when correcting the skeleton. The keyboard short-code is the same as in any other program: `ctrl+S`. You can also export the MTG as a list (`File -> Export MTG -> As Node List`) and the project as a plantGL file (`File -> Export View -> As PlantGL File`) so you can re-import it using OpenAlea.

1.3 Database Browser

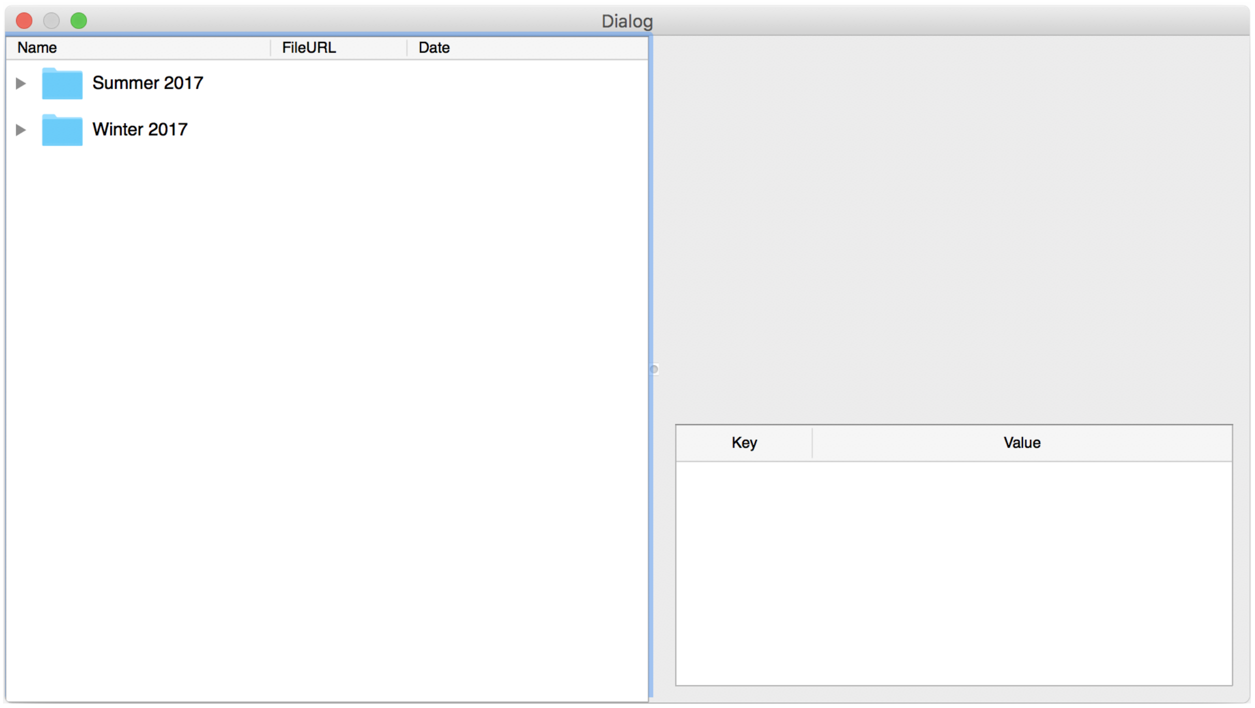
This part explain to use the database browser.

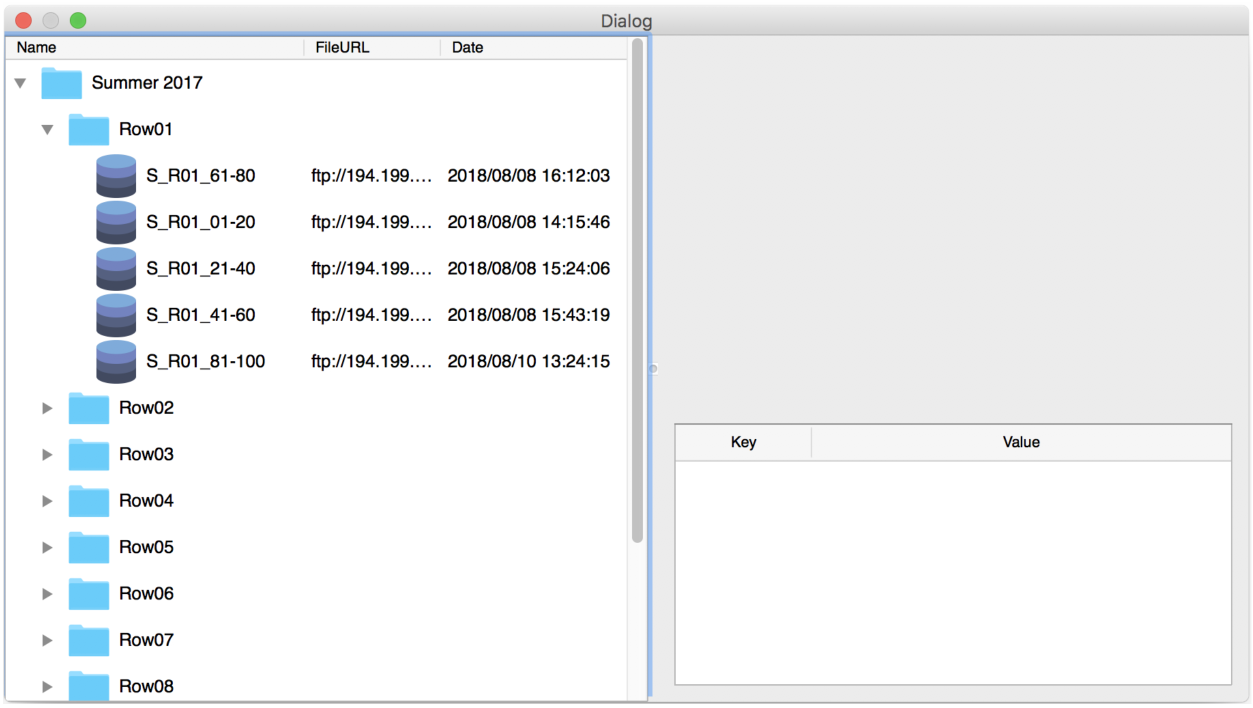
1.3.1 Database Connection



194.199.236.63:27017

1.3.2 Browse Database

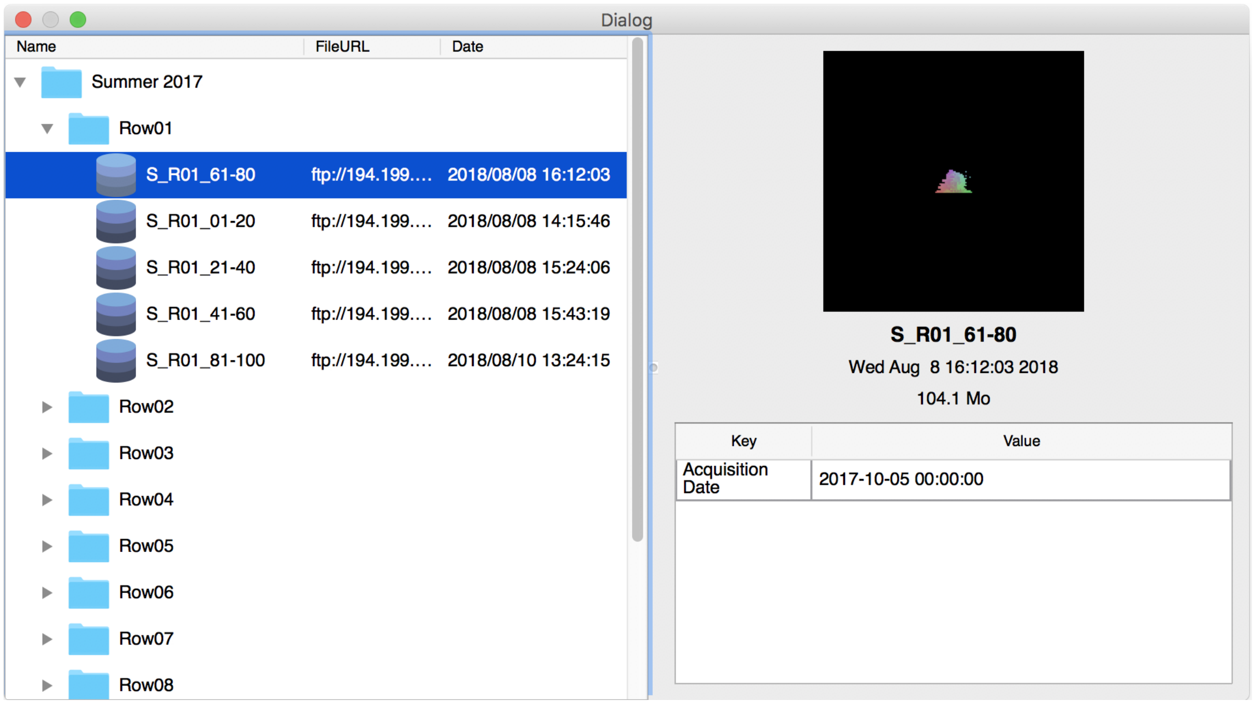




Dialog

Name	FileURL	Date
Summer 2017		
Row01		
S_R01_61-80	ftp://194.199.…	2018/08/08 16:12:03
S_R01_01-20	ftp://194.199.…	2018/08/08 14:15:46
S_R01_21-40	ftp://194.199.…	2018/08/08 15:24:06
S_R01_41-60	ftp://194.199.…	2018/08/08 15:43:19
S_R01_81-100	ftp://194.199.…	2018/08/10 13:24:15
Row02		
Row03		
Row04		
Row05		
Row06		
Row07		
Row08		

Key	Value



Dialog

Name	FileURL	Date
Summer 2017		
Row01		
S_R01_61-80	ftp://194.199.…	2018/08/08 16:12:03
S_R01_01-20	ftp://194.199.…	2018/08/08 14:15:46
S_R01_21-40	ftp://194.199.…	2018/08/08 15:24:06
S_R01_41-60	ftp://194.199.…	2018/08/08 15:43:19
S_R01_81-100	ftp://194.199.…	2018/08/10 13:24:15
Row02		
Row03		
Row04		
Row05		
Row06		
Row07		
Row08		

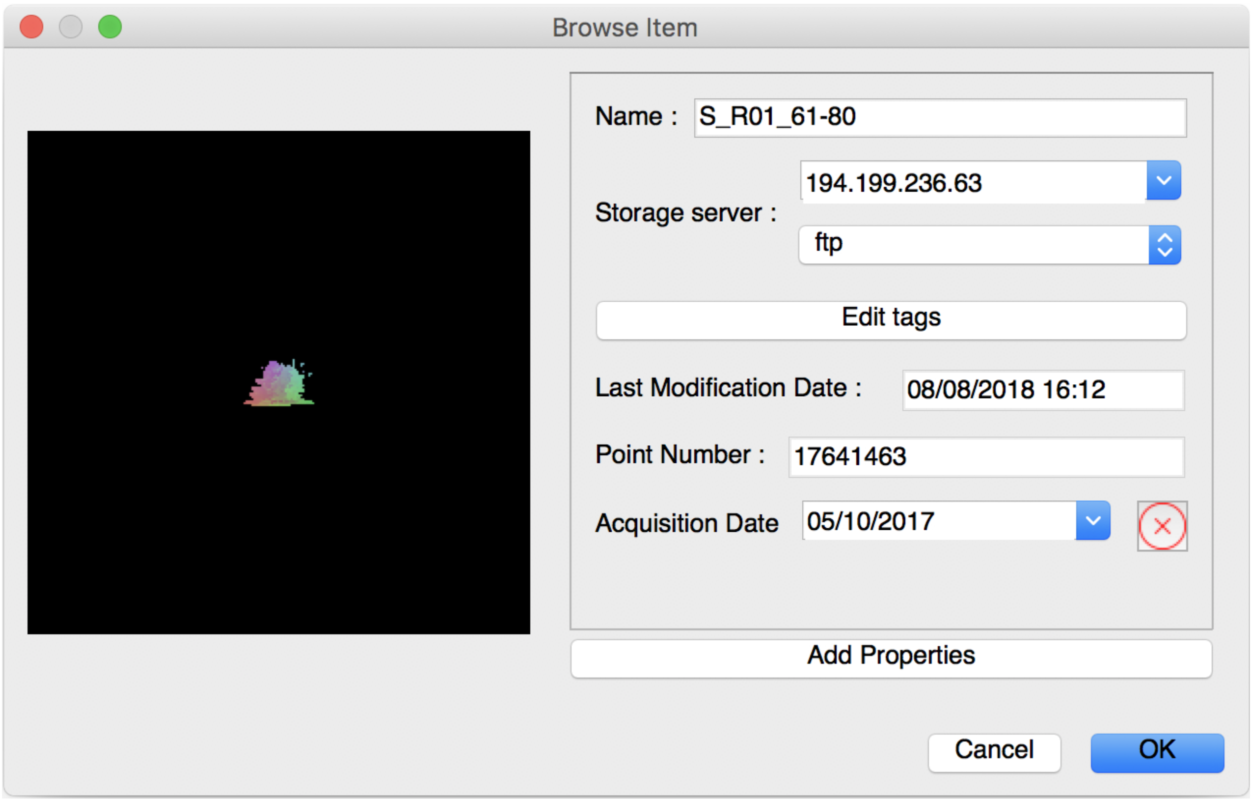
S_R01_61-80

Wed Aug 8 16:12:03 2018

104.1 Mo

Key	Value
Acquisition Date	2017-10-05 00:00:00

1.3.3 Database Item



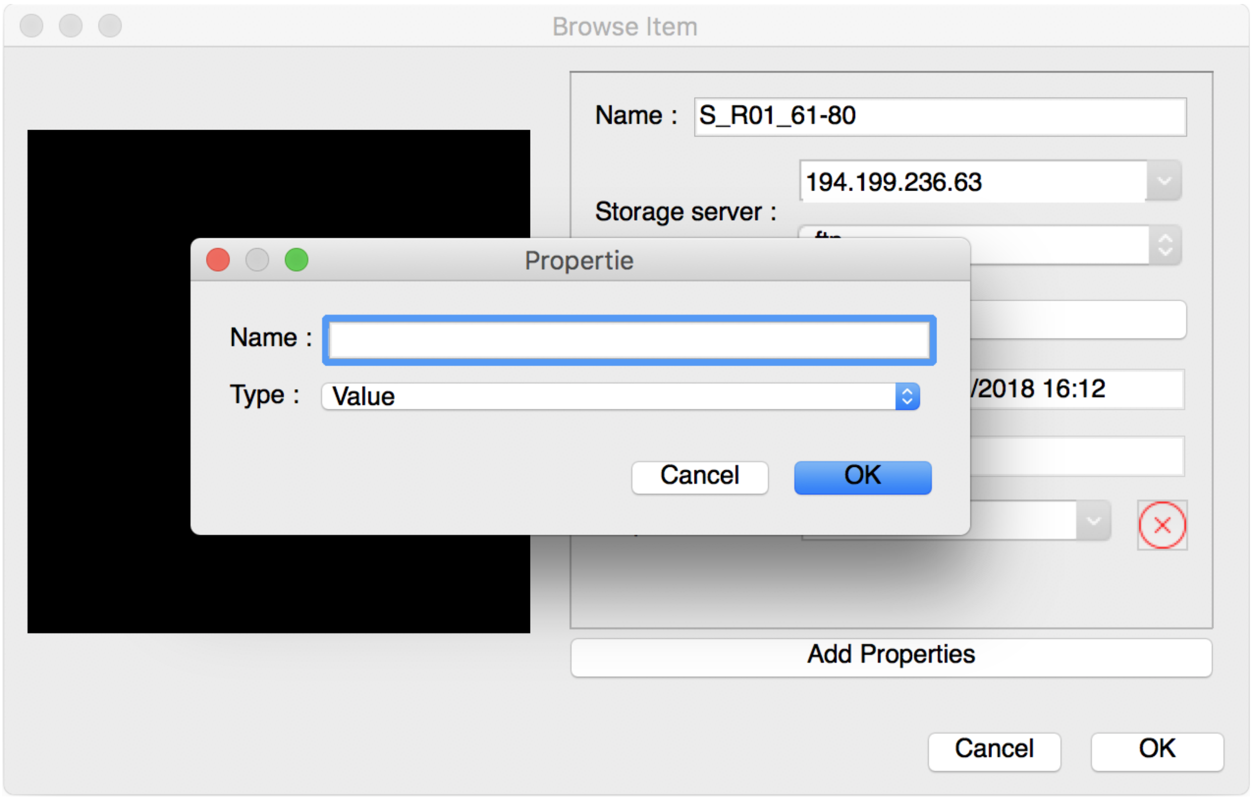
Name : S_R01_61-80

Storage server : 194.199.236.63 ftp

Last Modification Date : 08/08/2018 16:12

Point Number : 17641463

Acquisition Date 05/10/2017

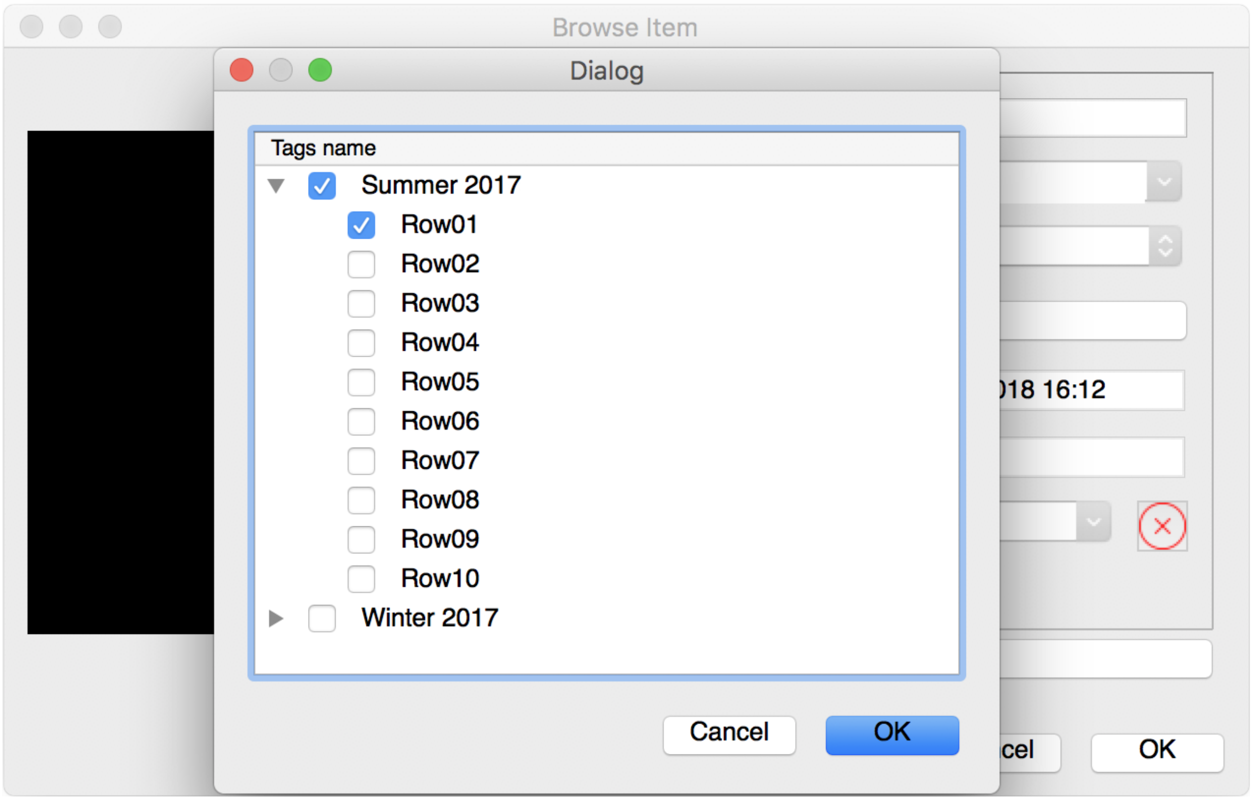


Name : S_R01_61-80

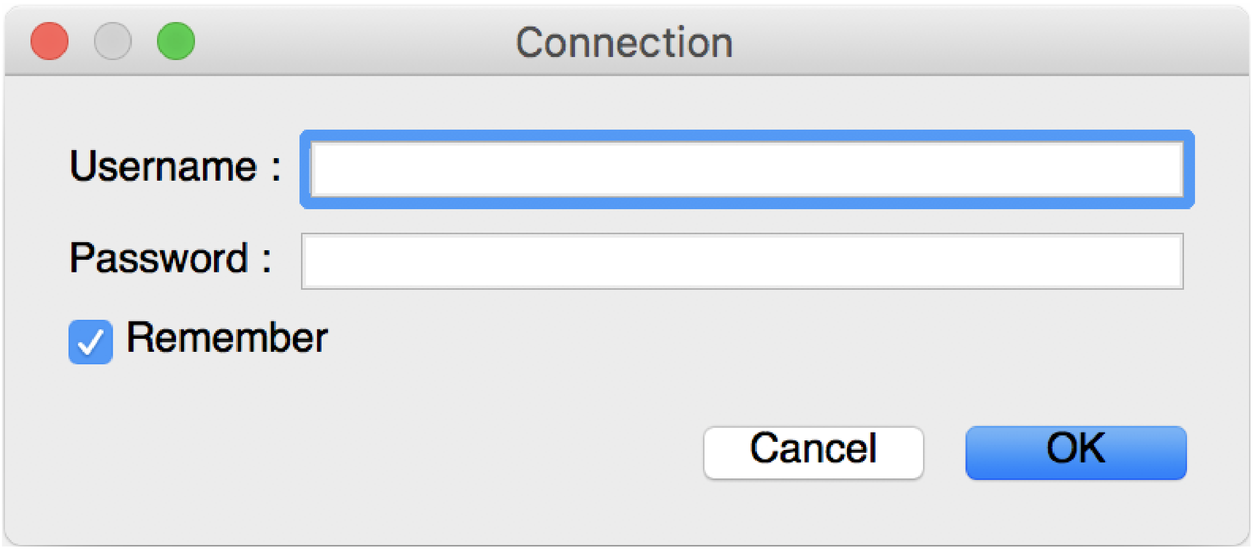
Storage server : 194.199.236.63

Name :

Type : Value



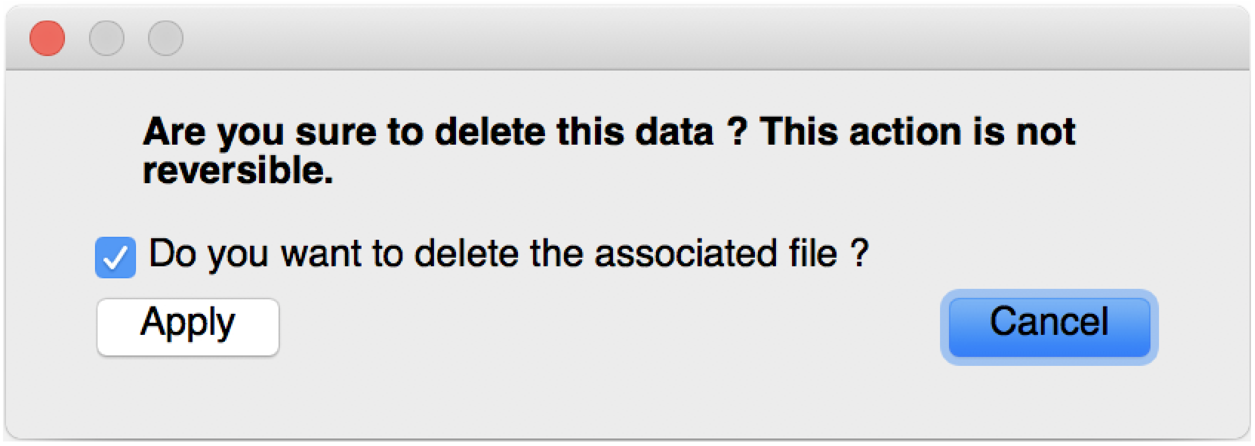
1.3.4 Import from the Database



1.3.5 Export to the Database

1.3.6 Edit a Database Item

1.3.7 Delete a Database Item



Are you sure to delete this data ? This action is not reversible.

✓ Do you want to delete the associated file ?