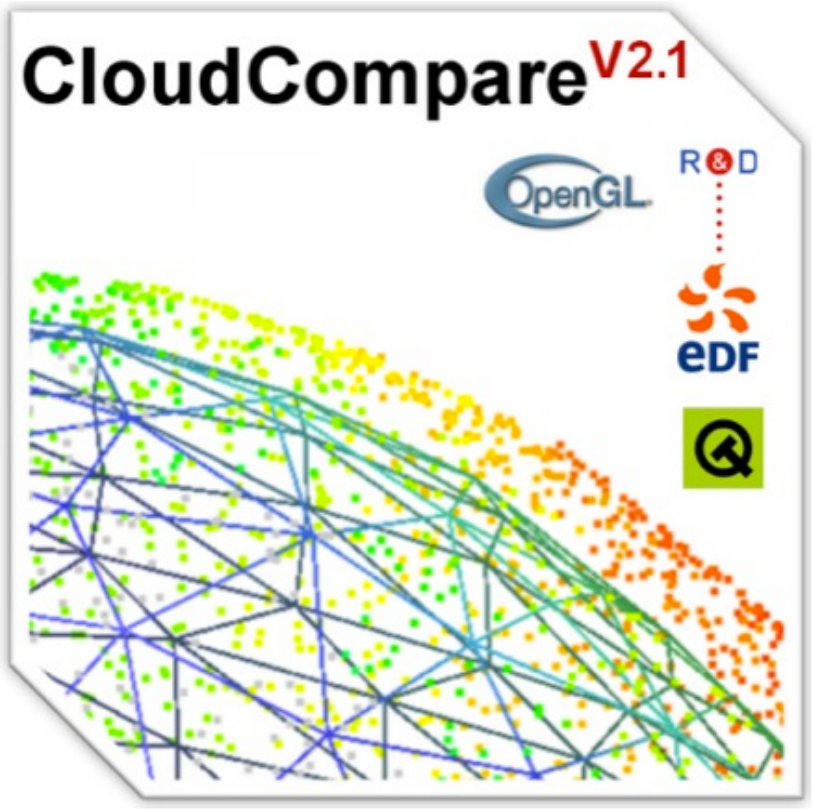


CloudCompare

User's Manual for version 2.1

Authors: DGM, AB, RM

Translator: Rosemary Le Faive

Funding for Translation Provided by Vehiclemetrics Inc.

www.cloudcompare.net

Table of Contents

Introduction

CloudCompare is an application for managing and comparing 3D point clouds (and, to some extent, surface meshes). Its development began in 2004 as part of a CIFRE thesis, funded by EDF R&D and housed at the *École Nationale Supérieure des Télécommunications* (ENST – Telecom Paris, TSI Laboratory, TII team). Work has continued after the thesis was completed in 2006. It is a platform to demonstrate algorithms that were studied in the thesis, and developed thereafter. In this sense, it is not designed for commercial use.

This program was primarily created to deal with point clouds. It was largely developed by Daniel Giradeau-Montaut, with the participation of:

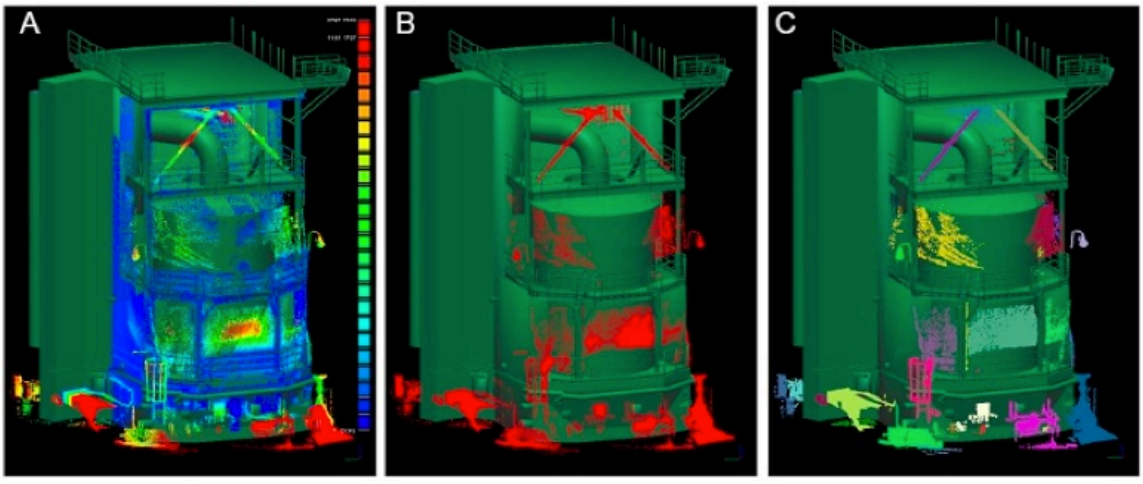
- Salma Bougacha (intern at EDF R&D in 2004)
- Aurélien Bey (PhD student at EDF R&D since the start of December 2008)
- Raphaël Marc (research engineer at EDF R&D)

along with elements graciously provided by Florent Duguet, then a PhD student at Telecom Paris.

CloudCompare has been mainly used to deal with interiors of power plants, mapped by EDF surveyors using laser scanners, and it has occasionally been used to deal with readings from dams or cooling towers.

This software can, among other things:

- calculate the local distances between two dense point clouds (figure below, left);
- filter out the measurement noise of the laser scanner to identify true differences (figure below, center);
- identify the individual objects (or parts of objects) that differ between two data sets (figure below, right).



The originality of *CloudCompare* comes from many aspects:

- the data structures used: an "octree" that allows large point clouds (many millions of points in 3D) to be stored in memory and displayed, as well as allowing the differences between two

large data sets to be calculated rapidly (that is to say, in a few seconds); as well as a "Kd-tree" which is used to quickly align two point clouds.
- two novel distance measurements between point clouds: a precise measure based on a *Hausdorff* distance (distance to the nearest neighbour); and a very fast though less precise measure based on the *chamfer* distance;
- measurement noise filtering;
- accounting for the sampling differences between the compared data sets;
- accounting for the visibility of the scanner for each data set;
- graphically mapping the PCV (Portion of Visible Sky) onto the point cloud, which improves the display of point clouds on the screen.

Even though *CloudCompare* is able to manage triangular meshes or sets of 3D polylines, these entities are primarily, to *CloudCompare*, point clouds (sets of vertices) that have particular structures, in addition to numerous other structures (octree, kd-tree, colors, normals, scalar fields, calibrated photos, etc.). The user is therefore advised to keep this aspect in mind while using *CloudCompare*; and should in particular pay attention to the role of each type of 3D entity in the processing tools offered by this software.

0.1. License

The *CloudCompare* software is comprised of the two following programs:

- the CClib library;
- the qCC executable which uses the CClib library.

Installing and using the two component programs of the *CloudCompare* software signifies that you accept the terms and conditions of their respective licenses. Version 2, dated March 2007, and previous versions of the two component programs are the property of EDF and TELECOM ParisTech.

License for the CClib library:
The CClib library is distributed under the GNU LGPL (GNU Lesser General Public License) as published by the FSF (Free Software Foundation) at: http://www.gnu.org/licenses/gpl.html

License for the qCC executable:
The qCC program is distributed under the GNU GPL (GNU General Public License) as published by the FSF (Free Software Foundation) at: http://www.gnu.org/licenses/lgpl.html

EDF and TELECOM ParisTech grant the user the rights to install and to use the *CloudCompare* software after having downloaded it from the site http://rd.edf.com. The software is provided as-is, with no explicit or implicit warranties. The authors take no responsibility for any harm caused directly or indirectly. The user assumes all risks and responsibilities with regard to the quality of the *CloudCompare* software and its use.

0.2. Installing the binary

CloudCompare works under the Windows 2000 and Windows XP operating systems. It is no longer being ported to Linux. The binary distribution of *CloudCompare* does not have an installer. Simply decompress the .zip archive containing the executable and the DLLs. Listed below are the files (.exe or .dll) that you will find after expanding the archive:

For *CloudCompare* version 2.0 (IHM based on Fltk/Flu):

- CloudCompare.exe (primary executable)
- CC_Dll.dll (CClib library)
- cv100.dll (DLL of the openCV library used for image processing)
- cxcore100.dll (DLL of the openCV library used for image processing)
- DevIL.dll (DLL of the Developer's Image Library which permits importing numerous image formats)
- glew32.dll (DLL of the OpenGL Extension Wrangler Library)
- highgui100.dll (DLL of the openCV used for image processing)
- ILU.dll (DLL of the Developer's Image Library which permits importing numerous image formats)
- ILUT.dll (DLL of the Developer's Image Library which permits importing numerous image formats)
- libguide40.dll (allows linking to Intel®'s DLL Math Kernel Library)
- xerces-c_2_7.dll (C++ library for parsing XML)

Remark: it is possible that running *CloudCompare* 2.0 will require the presence of the msvcp71.dll and msvcr71.dll DLLs. These DLLs are part of the Microsoft C++ Runtime Library (available for download on the Microsoft web site).

For version 2.1 of *CloudCompare* (IHM based on Qt 4):

- qCC.exe (primary executable)
- CC_Dll.dll (CClib library)
- FreeImage.dll (a library from the open source project FreeImage, which enables reading and writing of the latest image formats)
- mingwm10.dll (DLL for the minGW compiler which allows multi-threading)
- QtCore4.dll (DLL required by Qt)
- QtGui4.dll (DLL required by Qt)
- QtOpenGL4.dll (DLL required by Qt)

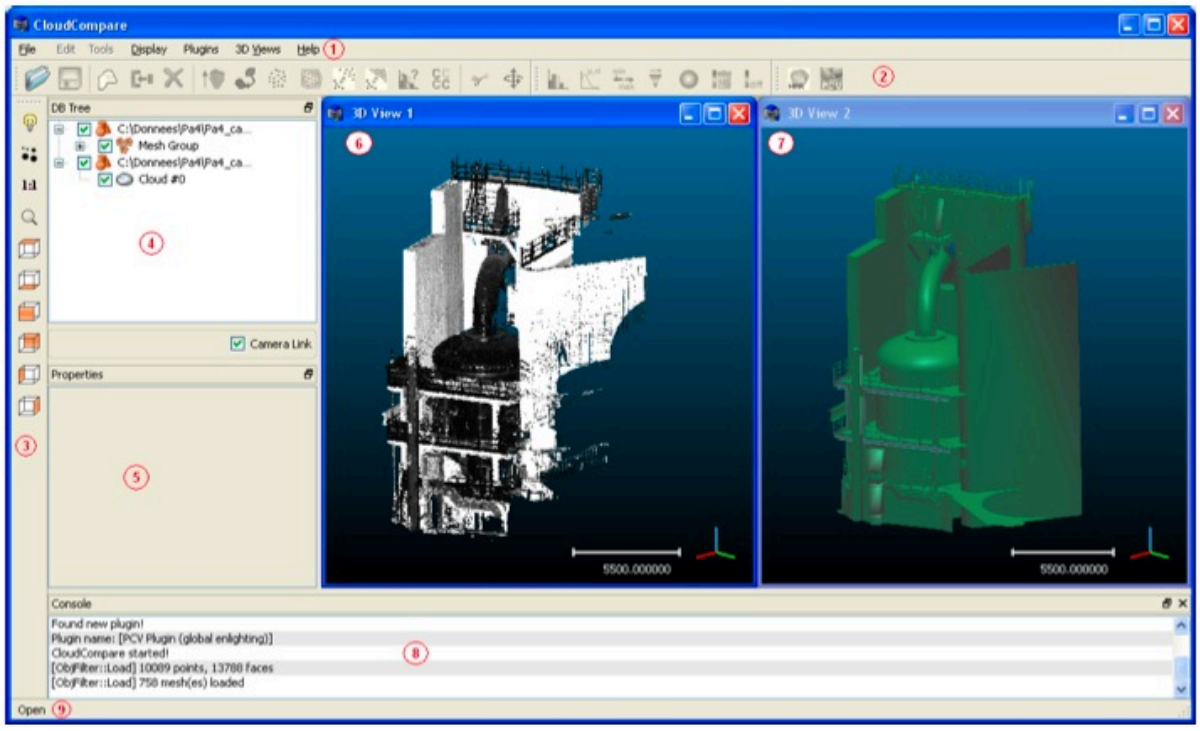
And the optional components, in the *plugins* directory:

– qHPRPlugin.dll (plugin for the "Hidden Point Removal" function)
– qPCVPlugin.dll (plugin for the PCV (portion of visible sky) rendering function)

Chapter 1. Interface

1.1. Main Window

The main window of *CloudCompare* has the following sections:



- 1: **Application menu bar**. This bar provides access to the functions available in *CloudCompare*. Sometimes, as a function of the selected objects, certain menus may not be available. For example, the "Edit" and "Tools" menu are inactive if no object is selected (visible in the screenshot above).
- 2 and 3: **Toolbars**. The different icons visible on these toolbars provide quick access to selected functions. These functions are also available through the appropriate menus in the menu bar (1). The toolbar (2) provides functions related to processing of objects, while the toolbar (3) is purely for changing the visualization parameters.
- 4: **Navigation Tree**. This area provides a tree visualization of the set of objects imported or created by the program. Objects can be selected by clicking on them in this area. Subsequently, we will use the term "*list*" for the set of elements grouped under the same tree structure (on the screenshot above, we can see two lists).
- 5: **Current object properties window.** If a single object is selected, this area lets you view and edit some properties of this object.
- 6 and 7: **3D views**. These areas let you visualize the objects available in memory "in 3D". In *CloudCompare*, objects can be visualized together in a shared 3D view, or in separate windows (in the screenshot above, two 3D viewers are open).

- 8: **Console**. This zone contains the log of information generated by the execution of *CloudCompare* (typically, supplementary non-essential information produced by the algorithms). It can be displayed or hidden via the "Console" command on the Display menu.
- 9: **Status bar.** Quick help messages related to certain functions of *CloudCompare* are displayed here. When you position the mouse cursor over a control button in the menus, the related help message will appear in this bar. However, not all the controls necessarily have such a message.

All the entities available in *CloudCompare* can therefore be accessed via the navigation tree (4) and possibly in the 3D view(s) (6,7).

It is possible to move or remove the different toolbars. To do this, simply grasp the manipulable part situated at the end of the bar by clicking on it, then move it while holding down the mouse button:

The bar can be left floating, or docked in a new position, in which case the neighboring toolbars will rearrange themselves. The toolbars can also be displayed or hidden via the Display / Toolbar submenu: the checked items correspond to the visible toolbars; unchecked items are hidden.

Similarly, it is possible to move the navigation window and the properties window by grasping their title bars and dragging them to the desired position.

In the rest of this document, we will describe the basic functionalities of *CloudCompare*, and describe all the available manipulations that affect the visualization of entities. The user can refer to the next part of this document (Chapter 2) to get detailed information on the advanced functions.

1.2. Available Objects

1.2.1. Navigation tree

As previously seen, *CloudCompare* displays the set of available objects (loaded into or created by this application) in the navigation tree window, which is located by default on the left side of the main window. We can find the following elements there:


A group of entities. If this corresponds to an open file, it will group all the elements loaded from this file, and will be named with the source file path.


A point cloud


A simple mesh


A group of meshes, treated here as a standard mesh (we can also find the vertices of these meshes as point clouds within this element type).


A structure


An object


A photo (possibly calibrated).

Note: groups of entities are not, technically speaking, manipulable entities. Typically, they cannot be used as entities for *CloudCompare* functions: they serve only as a "folder".

In the usual way, the tree can be opened by clicking on the buttons [+] and [-] situated at the left of the junctions of the tree (that respectively show or hide the sub-entities that are attached there).

The checkbox situated at the left of an element activates or deactivates the corresponding entity (respectively, when it is checked or unchecked). The concept of deactivation is stronger than simply showing/hiding an entity (the active status is a generic property of any 3D object -- see further). The inactive entities (and all their attached sub-entities) are not displayed, but further they are neither affected by the interactive operations like graphical segmentation, and more generally, they can no longer be selected (and can therefore not be used as inputs for *CloudCompare* functions).

1.2.2. Selecting Objects

To select an entity, the user can do one of two things: click with the left mouse button either on the entity in the navigation tree, or on its representation in the 3D view. In both cases, the selected object appears underlined in the navigation tree, and highlighted by a surrounding box in the graphical context or object figure. When an entity is selected, the information corresponding to it appears in the *Properties* window. Some properties displayed in this window can be modified; others are presented for information only.

It is possible to select multiple entities, by using the classic multi-selection commands in *Windows*: clicking multiple objects (in the navigation tree or in a graphical context) while holding down the CTRL key. To de-select one object while keeping the others already selected, click once more on that object while holding down the CTRL key. Alternately, while holding the SHIFT key down, then selecting two objects (in the navigation tree only), all objects visible between the two selected objects will also be selected. A final possibility is to hold down the left mouse button while moving the cursor over the objects to select in the navigation tree.

The majority of functions in *CloudCompare* only apply to selected entities. Their commands are therefore deactivated until the user has selected the appropriate number and type of objects.

1.3. Object Display

1.3.1. 3D view

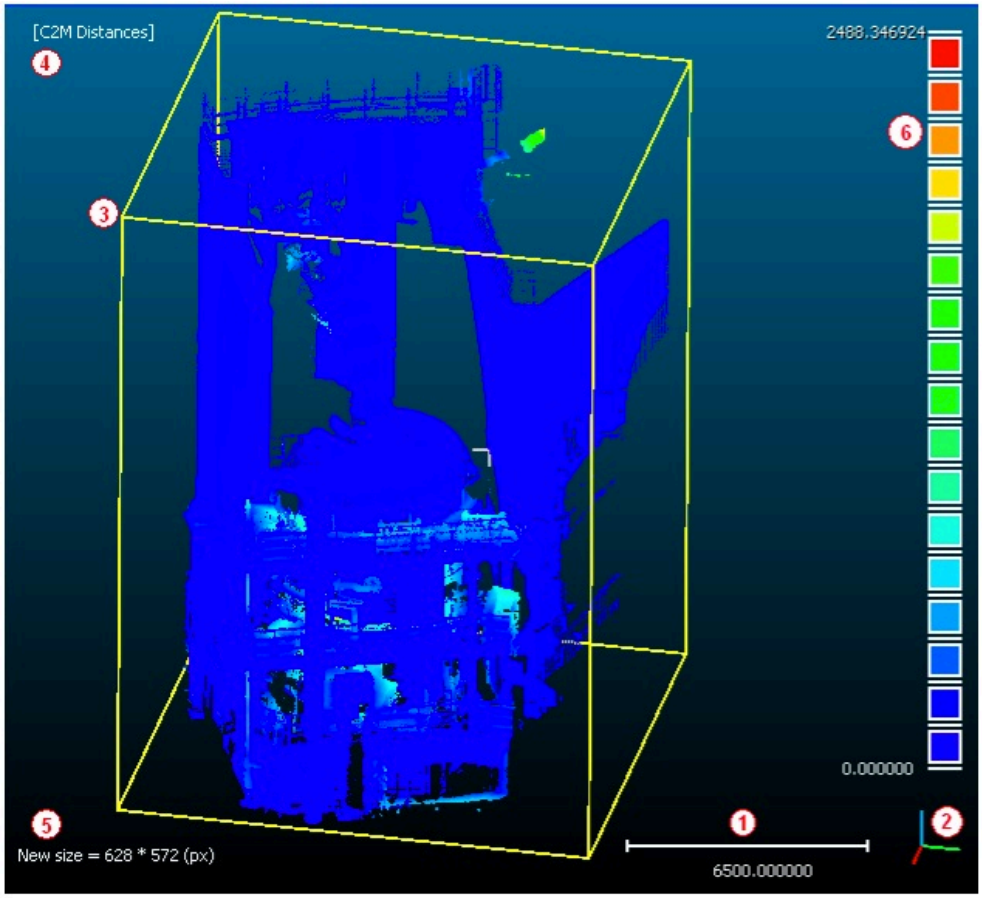


Figure 1.1 – Standard elements of a 3D view

The 3D views (or "graphical contexts" - see figure 1.1) are the sub-windows in which the entities can be visualized. They are all arranged in the following manner:

- 1: **Scale**. Provides a reference to estimate distances. The scale is displayed using the distance units in which the coordinates are expressed (*CloudCompare* does not use explicit distance units).
- 2: **Orientation triad**. The three displayed axes represent the X axis (red), Y axis (green) and Z axis (blue). The triad indicates the current orientation of the viewer.
- 3: **Bounding box**. This is the smallest parallelepiped (rectangle) that contains the entirety of the selected objects. The axes of the box are aligned on the X, Y, and Z axes, as per the orientation triad (2).
- 4: **Name of active scalar field** (refer to section 1.4.2.2 for more information about scalar fields).
- 5: **Temporary information concerning the display**. It may be the current dimension of the context (in pixels, after resizing the window), the type of projection used, etc.

- 6: **Color gradient**. Identifies the values associated with the colors when a scalar field is active and displayed.

1.3.2. Multiple 3D views

The majority of entities loaded in memory (and visible in the navigation tree) can be displayed in 3D views. Any number of these visualization windows can be created, and the entities can each be assigned individually to a different window.

To create a new 3D view, click on the command *New* in the *3D Views* menu, or simultaneously type CTRL and F3. A new window will then appear (by default it will fill the entire area dedicated to the 3D views).

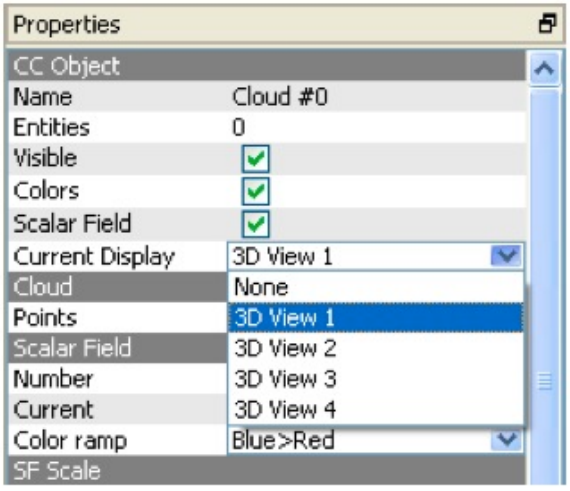
To divide the display area between the different open windows, different methods exist:

- Partitioning: each window occupies a portion of the available space, and the windows do not overlap. To partition the display, click on *Tile* in the *3D Views* menu.
- Cascading Display: each window occupies the same portion of the allotted space, and the windows all overlap each other. To cascade the views, click *Cascade* in the *3D Views* menu.
- Resize every context "by hand", notably with the assistance of the and  buttons (minimize this window and reduce this window, respectively), available at the top right of each window.

It is also possible to navigate between the different open views, using the commands *Next* and *Previous* on the *3D Views* menu, or accessing a context directly by clicking on its name in the *3D Views* menu.

To close a 3D view window, click *Close* in the *3D Views* menu after selecting the context, or directly via the  button on the window.

Finally, to change the window in which an object appears, select the object, and then in the properties window, choose the destination window in the *Current Display* dropdown list.



An object will not displayed in any view if *None* is chosen as the destination display.

1.3.3. Interactivity

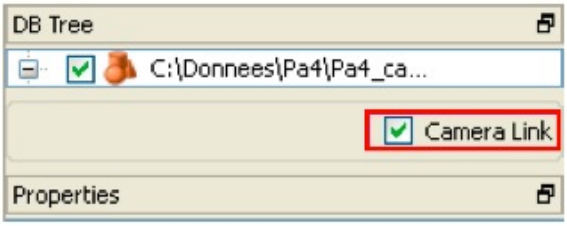
You can change the 3D point of view in a display by using the mouse:

Figure 1.2 – Mouse commands (L: select/rotate; M: zoom; R: pan)

- L - left click: [SELECTION] click on an object in a 3D display to select it.
- L - held down: [ROTATION] hold the left button down and move the mouse to perform a rotation around the current center of the display (cf. sections 2.4.4 and 2.4.5).
- M - scrolling: [ZOOM] scroll the mouse will forward (toward the top) to zoom in (get closer). Conversely, scroll down ("backwards") to zoom out (move farther away).
- R - held down: [PAN] hold the right mouse button down while moving the mouse to perform a translation of the point of view in the plane of the screen.

By default, the center of the rotation of the point of view is located in the center of the scene, and the visualization is done according to an orthographic projection (without perspective). While changing the point of view (rotating the viewing vector), large point clouds are temporarily sampled in a manner to permit interactive rendering of the movements.

The graphical displays (3D views) are independent from each other, and changes in the point of view in one viewer don't necessarily have any repercussions in the others. All the same, it is possible to synchronize the camera movements so they will be applied to all the windows at once. To do this, simply check the *Camera link* option under the navigation window. To re-establish the normal functioning of the graphical displays, un-check this box.



1.3.4. Display Options

The following commands allow the user to modify the display properties. Some of them are notably visible via the *Viewing tools* toolbar.

 Change the lighting parameters in the scene (details below).

 Change the display size of the points.

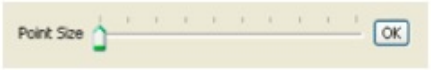
 Reset zoom and re-center the camera on the whole scene in the current 3D view.

 Zoom and re-center the camera on the selected objects. This has no effect if no objects are selected.

 Use these 6 buttons to toggle between different predefined points of view (in the order that the buttons appear: top, bottom, front, back, left, right). The predefined points of view are defined according to the orientation triad.

The  button opens the lighting and materials parameters window. In it, the user can modify the mood of the light source, the default color of point clouds, the default color of meshes, distinguish between "front" and "back" faces, and set the colors of various elements that make up the graphical display. This functionality is described in more detail in section 2.4.7.

The  button permits you to access the point size parameter, via a dialogue box visible over the top right of the graphical display area:



To change the displayed size of the points, simply slide the tracker: the farther to the right, the larger the points will be. The display will update interactively. Click OK to confirm and close the dialogue box.

Other functions are accessible via the *Display* menu:

- *Full screen*: show the main window in full-screen mode. To return to the normal display mode, press the F11 key.
- *Refresh*: update the display.
- *Test frame rate*: start testing the frame rate (expressed in Frames Per Second) of the current context. In theory, the more triangles or points to display, the slower the frame rate will be. Between 10 and 20 images per second, a human perceives the display moving in discrete steps; above 24 fps., the display is perceived as moving fluidly. If the refresh rate is too slow, it is advised to limit the number of objects to render by only displaying the objects of interest.

- *Toggle viewer based perspective*: switch to a perspective projection and position the center of rotation of the point of view on the camera itself. To re-establish the center of rotation to its original position (in the center of the frame), click once more on this command.
- *Toggle centered perspective*: switch to a perspective projection and position the center of the rotation of the point of view on the center of the observed scene. To reset default projection (parallel orthographic) click once more on this command.
- *Light & Materials/Toggle sun light*: turn on/off the global light source.
- *Light & Materials/Toggle custom light*: turn on/off the personalized light source (which can be moved by holding down the CTRL key while panning with the cursor over the screen).

Note: shadow effects are not visible unless at least one light source is active.

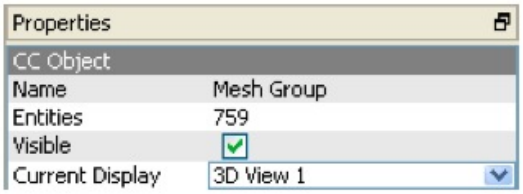
It is also possible to change the individual display color of a point cloud. The color is only visible if no scalar fields are actively displayed (cf. section 1.4.2.2). To change the color of a cloud, select it then open the *Edit/Colors* sub-menu.

- The *Set unique* command lets the user select a single color to apply to all the points in the cloud (cf. section 2.2.1).
- The *Colorize* command enables the user to combine a new color with the current color of the point cloud (cf. section 2.2.2). This operation consists of multiplying the colors component by component, the colors being expressed in the form of a [Red, Green, Blue] vector. For example, if the cloud is yellow at any point ($[1,\ 1,\ 0]$), and we apply Colorize with a purple color ($[1,\ 0,\ 1]$), we obtain a red cloud ($[1*1,\ 1*0,\ 0*1] = [1,\ 0,\ 0]$).
- The *Height ramp* command enables the user to apply a gradient of colors along one of the 3 principal axes (cf. section 2.2.3). The Clear command removes the current color of the cloud. After applying this command, the points will display in the default color.

1.4. Properties of Objects

1.4.1. Common properties

Certain fields visible in the properties window are common to all entities in *CloudCompare*. We find them in the CCObject section:



- *Name*: name associated with an object. It can be modified by double-clicking on the name (right side of the field) or by pressing the F2 key.
- *Entities*: this field shows the number of sub-entities attached to the current object. In the screenshot above, for example, the selected object is a group of meshes comprising 759

sub-meshes (and point clouds). The sub-entities can be accessed independently by expanding the current object in the navigation tree window (cf. section 1.2.1).
- *Visible*: if this box is checked and if the object is not deactivated in the navigation tree (cf. section 1.2.1), then the object is displayed in the 3D view selected as *Current Display*. Otherwise, it stays hidden and only its bounding box can appear (when it is selected).
- *Current display*: selects the 3D view in which the object is displayed (cf. section 1.3.2).

The next section presents the properties specific to each type of entity. There are certain fields (or even whole sections) that are available only after certain manipulations or for certain object types.

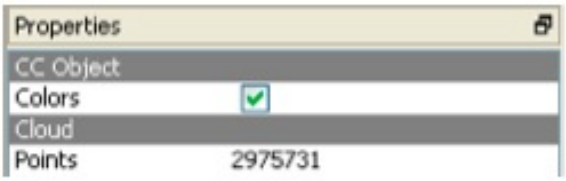
1.4.2. Point Clouds

1.4.2.1. Common Properties

Two fields are common to all point clouds:

- *Color*s: available in the *CCObject* section, it activates (when checked) or deactivates (when not) the display of the color parameter for the point cloud (cf. section 1.3.4). Warning: do not confuse the colors of points with scalar fields, which, though they are displayed as colors, actually represent real scalar values associated with each point, for which color is merely a mode of visualization – cf. section 1.4.2.2. It is therefore possible that the points remain colored on the screen even though the "color" of the cloud is deactivated.
- *Points*: available in the *Cloud* section, indicates the number of points composing the cloud.

Point clouds that have normals (normal vectors) have a *Normals* option for whether these vectors should be used in rendering the graphical display (via a semi-realistic shadow effect).

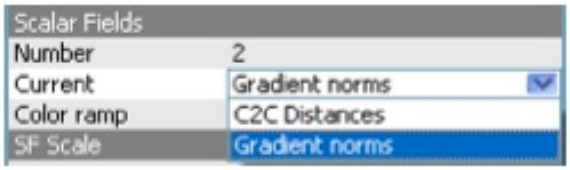


1.4.2.2. Scalar fields

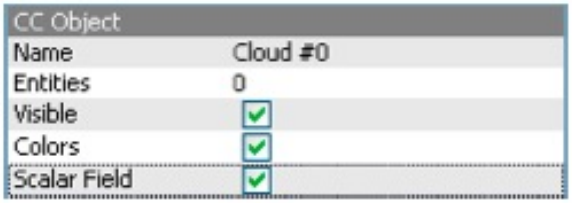
Some functions, when applied to point clouds, associate a real value (a scalar) with each point. The set of these scalars is saved in a structure called a *Scalar field* and is linked with the point cloud.

It is possible to link multiple scalar fields to the same cloud, but only one can be active at a given time. The number of scalar fields associated with a point cloud can be seen via the *Number* entry in the *Scalar field(s)* section.

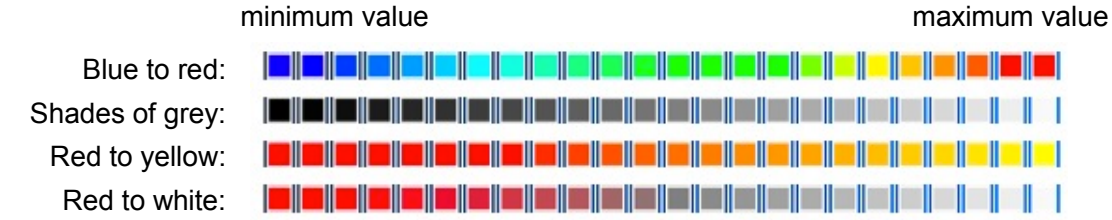
To choose the active scalar field, you must select it in the *Properties* window of the cloud, in the *Current* drop-down list in the *Scalar field(s)* section. We note in passing, that the scalar fields are named automatically by *CloudCompare* as a function of the analysis from which they were generated. One field can be displayed at a time.



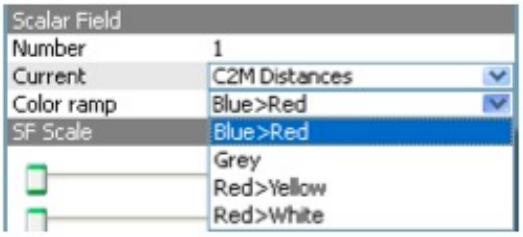
The user can visualize the values in the active scalar field by means of a color gradient, applied to the points in the cloud. To do this, simply check the *Scalar field* box in the *CCObject* section. Each point of the cloud will then be colored as a function of its value, according to the chosen color scale. To return to the normal display for the point cloud, uncheck this same box.



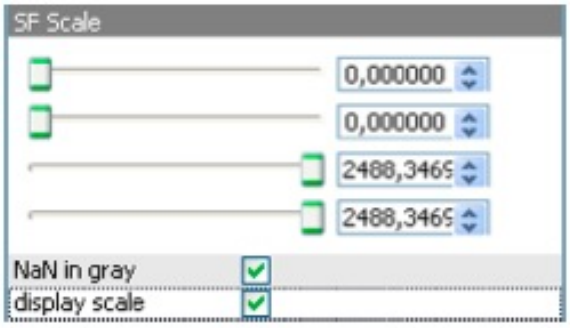
Different color scales are available. Each corresponds to a gradient of colors, the first color being associated with the minimum value of the point cloud, the last to the maximum value, and the rest are associated with intermediate values according to a linear scale.



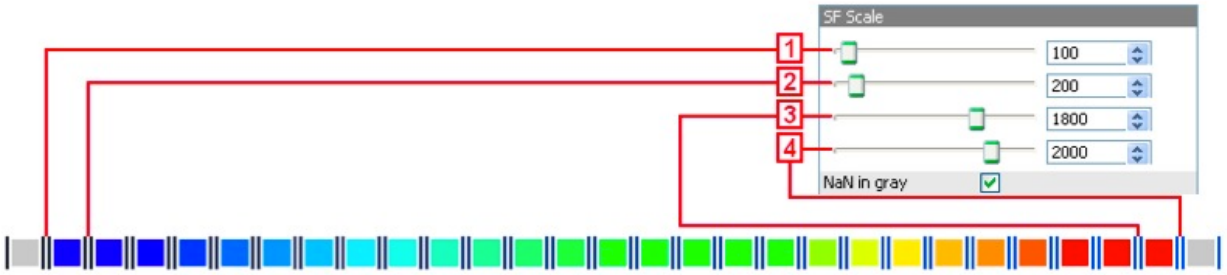
To change the color scale of the scalar field visualization, select the desired gradient in the *Color Ramp* drop-down list in the *Scalar field(s)* section:



You can display the color scale in the 3D view where the point cloud is displayed (cf. figure in section 1.3.1), by checking the *Display Scale* box in the *SF scale* section. To hide the scale, uncheck this box:



Finally, *CloudCompare* offers the user the possibility to more finely adjust the visualization parameters of the current scalar field, by manipulating the four sliders available in the *SF scale* section (or by modifying the associated text fields which are found to their right). In case of doubt, to know what a slider corresponds to, just position the cursor of the mouse over it and wait a few seconds until a description appears (you can also read the help text visible immediately in the status bar at the bottom of the main window). The role of each slider is defined in the following manner:



- 1: Minimum displayed value. All the scalar values below (1) are ignored in the display.
- 2: Minimum saturation value. All the points having an associated scalar value below (2) are displayed in the "weakest" color (here, in blue).
- 3: Maximum saturation value. All the points having an associated scalar value above (3) are displayed in the "strongest" color (here, in red).
- 4: Maximum displayed value. All scalar values above (4) are ignored in the display.

The display of the ignored values is governed by the *NaN in grey* property: if it is checked, points associated with ignored values are displayed in grey. If not, the points are not displayed.

The four values presented above define two ranges:

- values to be displayed, for which the points are effectively displayed in color. The behaviour outside of this range depends on whether *NaN in grey* is selected.
- the saturation range, within which the gradient is used. The lower bound of this interval is associated with the first color of the scale, the upper bound to the last color, and the rest of the colors are distributed linearly onto the values comprising the interval.

CloudCompare verifies the chosen values in real time so you cannot have (1) > (4) or (2) > (3). Any valid modification to one of these values is interactively reflected in the current display.

1.4.3. Meshes / groups of meshes

These two types of entities share the same properties:

- *Normals*: available in the CCObject section, this field activates the use of the normals in the object rendering. The normals permit us to create a semi-realistic visual rendering of objects, as they serve to calculate the lighting of the faces. If this option is deactivated, the lighting calculation cannot be performed and the surface of the object is displayed in a uniform color ("silhouette" effect, losing the effect of depth).
- *Faces*: available in the *Mesh* section, indicates the number of faces composing the mesh. In the case of a group of meshes, the number corresponds to the total number of faces in the sub-meshes that are part of this group.
- *Wireframe*: available in the *Mesh* section, activates (if the box is checked) the "wireframe" rendering of the object.

The "wireframe" displays only the edges of the facets that compose the mesh (as line segments). It is created without masking hidden edges: all edges are displayed, even those which are not visible from the point of view of the camera as a result of being hidden by faces (see figure 1.3).

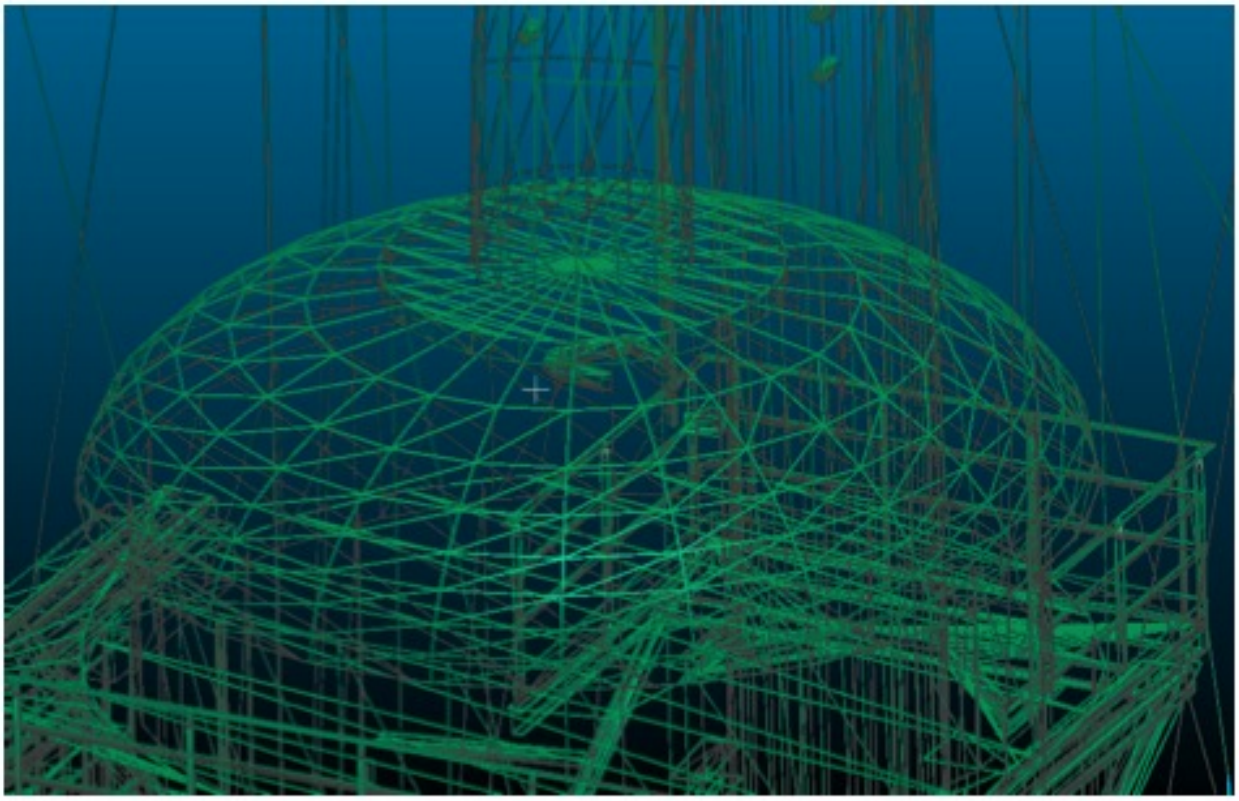


Figure 1.3 – Example of a "wireframe" of a mesh

1.4.4. Octree

1.4.4.1. Description

The octree is a structure designed to speed up the processing of spatial data. It is a recursive and hierarchical slicing of space (decomposing the space into cubes). In most cases, *CloudCompare* constructs one of these structures over a point cloud in order to accelerate its functions. This type of octree is especially fast to construct.

From a general point of view, the octree is defined by its *subdivision level*:

- The first level (level 0) is the smallest cube that entirely contains the point cloud.
- At level N+1, the octree is built by dividing each of the cubes in level *N* into 8 sub-cubes of the same size (in practice, we only store sub-cubes that contain at least one cloud point).

It is important that the user understands the principle of this structure, because it is a central part of *CloudCompare*. The higher the octree level is raised:

- the higher the number of cubes to process: potentially $8^N = 2^{3N}$ cubes at level N. For instance, though it is very improbable, we could have up to $2^{21} = 2\ 097\ 152$ cubes at the 7th level of an octree. At level 10, we have 2^{30} cubes, which would be a little over a billion (again, this is very unlikely, especially since it would require at least as many points). In practice, many cubes are empty and are not stored in memory, resulting in a structure that is generally relatively compact. Finally, *CloudCompare* uses a special coding for the octree, which ensures that its size is always proportional to the number of points in the cloud, and doesn't depend on the cloud's spatial distribution. It is also for this reason that the maximum level of an octree is fixed at 10 in *CloudCompare*, so the encoding per point is not too large.
- the smaller the cubes are: if $[a_N]$ is the length of the edge of a cube of level [N] (therefore a_0 is the size of the initial cube), $[a_N = a_0/2^N]$. In effect, each time we go down a level (increase N), we half the lengths of the cubes along each dimension. For example, the cells of an octree at level 5 are 32 times smaller (in side length) than the cloud's bounding cube (level 0).
- the fewer number of cloud points per cube: intuitively, the smaller the cubes are, the fewer points they contain.
- the closer the surface of the octree (the global surface formed by the collection of external surfaces of the cubes) is to the point cloud.

The 3 screenshots in figure 1.4 give an outline of the division and the size of the cells of the octree at different levels of subdivision.

Many functions that apply to point clouds make use of an octree. As much as possible, the optimal level of an octree for the chosen calculation is determined automatically and transparently by the program. All the same, certain calculations ask the user to estimate a level to use. In some situations, the user will have to find a level that provides the best compromise between the number of cubes to deal with (not too large, therefore not too high a level) and the average number of points per cube (as small as possible, therefore not too low a level). Finding the right level can require some experience.

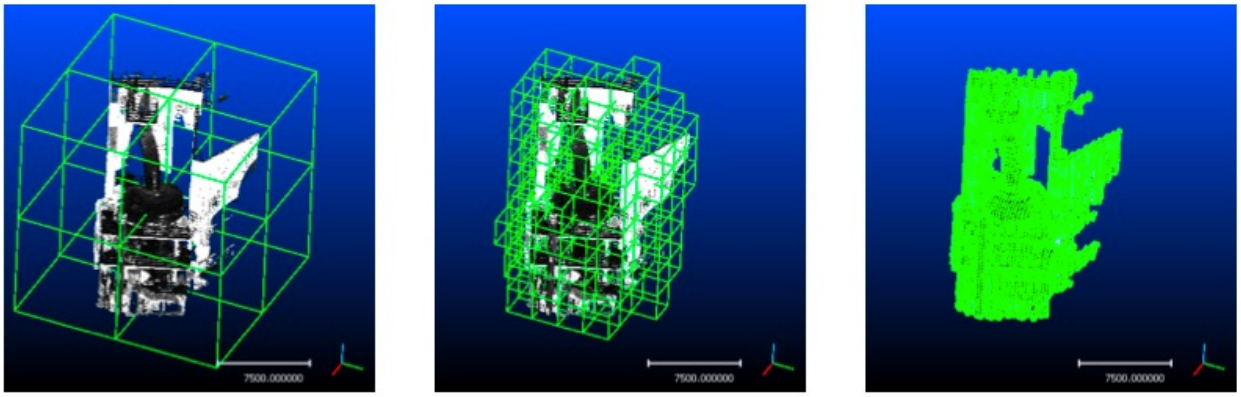


Figure 1.4 – Cells at different levels of octree construction: level 1 (left), 3 (middle), and 6 (right).

1.4.4.2. Display

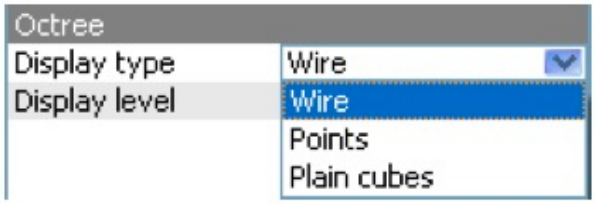
The octree is inseparable from the point cloud to which it is attached. It is an abstract structure, available in the *CloudCompare* interface only as a visual. The display proposed by *CloudCompare* only allows visualization of one level at a time. You can change the level of display of the octree by raising/lowering the *Display level* field in the *Octree* section.

The display levels can be between 1 and 10. Level 0 can not be selected (it is never used in practice, not even for calculations, because a level 0 octree is simply the entire cloud.

The display might slow down considerably at a certain level of subdivision (a function of the capabilities of the computer on which the program is running) as a result of the number of elements to display. In this case, choose lower display levels as much as possible.

The octree can be visualized in different forms:
- *Wireframe*: only the edges of the cubes are represented.
- *Points*: each cube is represented by the center of gravity of the points contained within it.
- *Plain cubes*: the surface of the cubes is fully displayed.

To change the display of the octree, select a mode in the *Display Type* drop-down list in the *Octree* section:



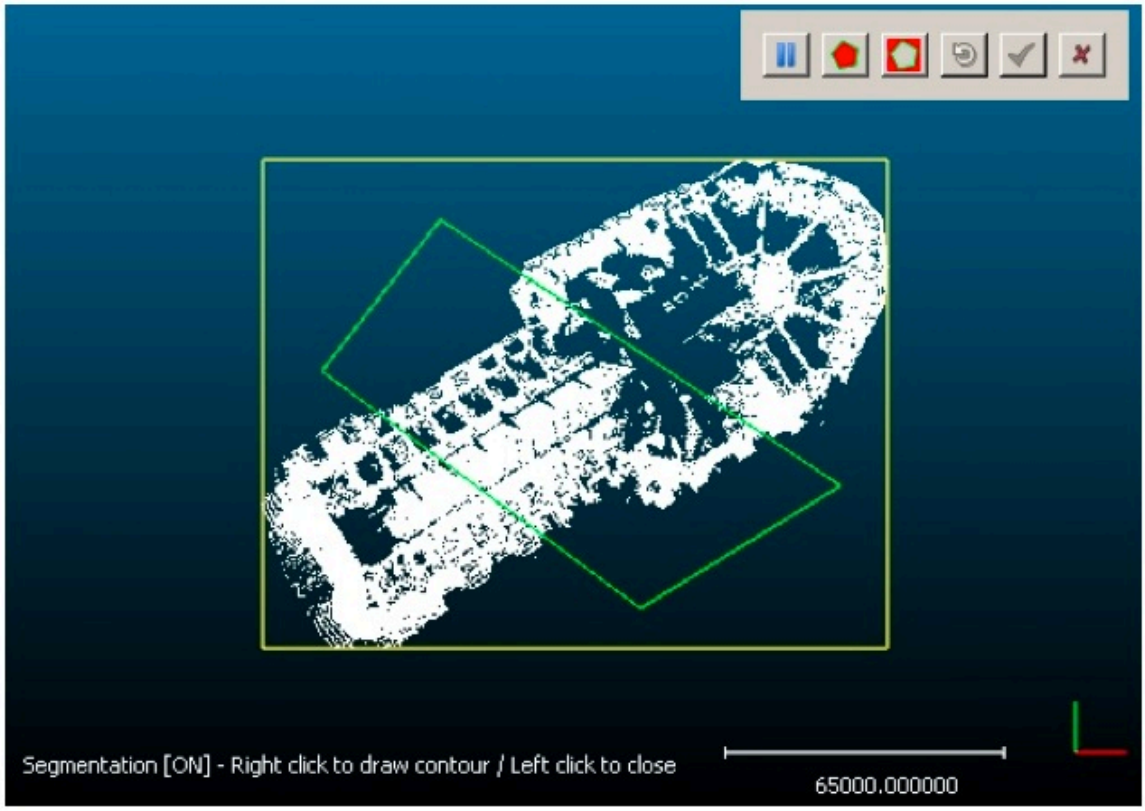
1.5. Interactive modification of entities

1.5.1. Manual segmentation

This tool is accessible via the icon.

It is possible to manually "cut" the screen containing the selected entities. The user may define a contour by a successive series of clicks, and decide whether to "keep" the points inside or outside of this contour. The process is repeatable at will, and the rejected points are hidden. If the user confirms this "trim", the current cloud is replaced by the points corresponding to the selected points, and a new cloud with the remaining points is created.

This can also be done to meshes, but the triangles on the border are not technically "cut" (triangles completely within the border are kept, and the rest are considered to be outside of the selection).



The tool activates an icon bar, which appears at the top right of the screen:

- the first icon pauses the selection (allowing the 3D entity to be rotated, for example, between two "cuts"). Caution: if the segmentation is paused, the currently drawn contour will disappear. If the cut is performed (see the next two points), the segmentation is automatically paused. If the tool is not paused, the user can define the contour by successive left-clicks on the screen. The first will not appear to do anything, but the next ones will add line segments connecting the current location to that from the previous click. A right click will close the

contour (necessary to perform the selection). Also, any right click performed after closing the contour will reset the contour.
- The next icon corresponds to a selection of points inside the contour. The outside points are hidden and the segmentation is paused. To do this, the contour must be closed.
- The next icon corresponds to selecting points outside of the contour. The points on the interior are hidden and the segmentation is paused. To do this, the contour must be closed.
- The next icon resets the selection completely (all the points become visible again).
- The second-to-last icon validates the trim, and quits.
- The last icon cancels the cut and quits.

1.5.2. Manual rotation/translation

This tool is accessible via the  icon.

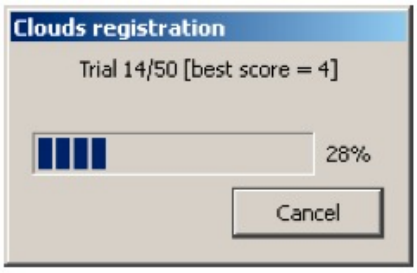
It applies a rotation or a translation manually to the selected entities. When it is active, 3 icons appear at the top right:

- The first resets the transformation applied to the selected entities.
- The second confirms the transformation, and quits.
- The last cancels the transformation, and quits.

The transformation is done interactively with the mouse, using the same conventions as for transforming the point of view (cf. section 1.3.3). The left button allows you to turn the entities (around the center of gravity of the bounding box of the selected items) and the right button translates the entities on the plane of the screen.

1.6. Progress bars

Some functions in *CloudCompare* require a relatively long processing time, which can be between several seconds to several minutes depending on the computer on which the program is running, as well as the parameters of the function and the complexity inherent in the particular function. The progress bar is a window that appears during non-immediate calculations, giving a visualization of the advancement of the calculation (when the bar is full, the processing is done).



Among the commands on the progress bar, some allow one to stop the calculation during its execution – via the *Cancel* button (the stopping of the program isn't necessarily immediate; the

function will stop when it is able to do so). When this button is not visible, the calculation cannot be interrupted before it finishes.

1.7. Toolbars

Listed below are the buttons available in the various toolbars in *CloudCompare*.

Button	Command	Description
Main Tools Toolbar 		
	Open	cf. section 2.1.1
	Save	cf. section 2.1.2
	Clone	cf. section 2.1.24
	Fuse	cf. section 2.1.25
	Delete	Delete the selected objects
	Register	cf. section 2.3.4
	Align	cf. section 2.3.3
	Subsample	cf. section 2.2.28
	Sample points	cf. section 2.2.10
	Cloud/Cloud distance	cf. section 2.3.5
	Cloud/Mesh distance	cf. section 2.3.6
	Statistical test	cf. section 2.3.9
	Label connected components	cf. section 2.3.10
	Segment	cf. section 1.5.1
	Translate/Rotate	cf. section 1.5.2

Scalar Field Tools Toolbar		
		
	Show histogram	display the histogram of the scalar field
	Compute statistical parameters	cf. section 2.3.8
	Filter by value	cf. section 2.2.20
	Gradient	cf. section 2.2.18
	Gaussian filter	cf. section 2.2.19
	Delete current scalar field	removes the active scalar field of the selected cloud
	Difference	cf. section 2.2.21
		
	Hidden Points Removal	cf. section 2.5.2
	Global enlightening	cf. section 2.5.1
Viewing Tools Toolbar		
		
This toolbar is explained in section 1.3.4.		

The toolbars are displayed or hidden according to whether they are checked or not in the *Display*/*Toolbars* sub-menu. To change the visibility of a toolbar, click on its name in *Display/Toolbars*.

1.8. Keyboard shortcuts

The following table lists the keyboard shortcuts available in *CloudCompare*:

Key(s)	**Command**	**Function**	**Remarks**
F1	Help	Display the help documentation for *CloudCompare*	
F2	Rename	Rename the selected entity	

Key(s)	Command	Function	Remarks
F3	Toggle centered perspective	Position the center of rotation of the point of view on the center of the scene; activate/deactivate the projection perspective.	cf. section 1.3.4
CTRL+F3	New 3D View	Open a new 3D Viewer	
F4	Toggle viewer based perspective	Position the center of rotation of the point of view on the camera / re-establish the default point of view	cf. section 1.3.4
CTRL + F4	Close 3D View	Close the current 3D view window (before applying this shortcut, at least one object in the window to close must be selected)	
F5	Refresh	Refresh the display	
F6	Toggle sunlight	Turn on/off the global light source	cf. section 1.3.4
F7	Toggle custom light	Turn on/off the secondary light source	cf. section 1.3.4
F8	Set unique color	Apply a color to the selected objects	cf. section 1.3.4
F11	Full screen	Display the main window in full screen/ return to the default display	
F12	Test frame rate	Calculate the refresh rate of the current 3D view	cf. section n1.3.4
CTRL + O	Open	Open an object from a file	cf. section 2.1.1
CTRL + D	Delete list	Remove the selected list from the application	

Chapter 2. Functions

In this section, we will describe the functions accessible through the main menu. They are arranged by sub-menu.

2.1. 'File' Menu

2.1.1. Open (file)

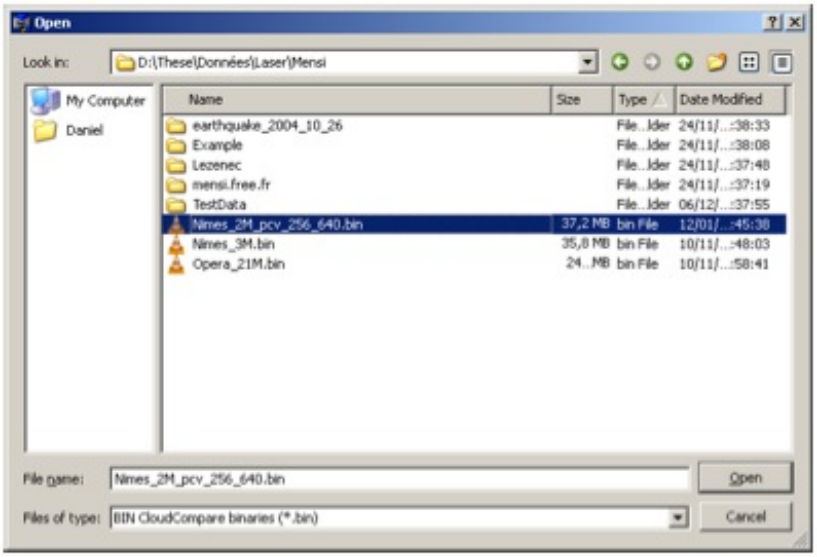


Figure 2.1 – File selection interface

Loads a 3D entity into memory via the standard interface (figure 2.1).

Notes:

- Keyboard shortcut: CTRL+O
- If opening the file is successful, the entity will be automatically displayed in the active 3D view.
- Use the *Look in* dropdown list (see above) to get access to the usual locations, as well as recently used locations.
- Use the *Files of type* dropdown list (at the bottom) to choose a filter to apply to files in the current folder (if the filter is "*.*", the format of the selected file is automatically detected by *CloudCompare* according to its extension).

2.1.2. Save (file)

Saves the selected entity to file (or multiple entities, if the chosen file format permits). The file is saved using the standard interface (figure 2.1).

Notes:

- Use the *Look in* dropdown list to access the usual locations, and the recently visited locations.
- Use the *Files of type* dropdown list (at the bottom) to filter the currently displayed files and to choose the file format in which to save.
- An extension will be automatically added to the file name according to the chosen file format (if the user does not specify an extension).

2.2. 'Edit' Menu

2.2.1. Colors > Set Unique

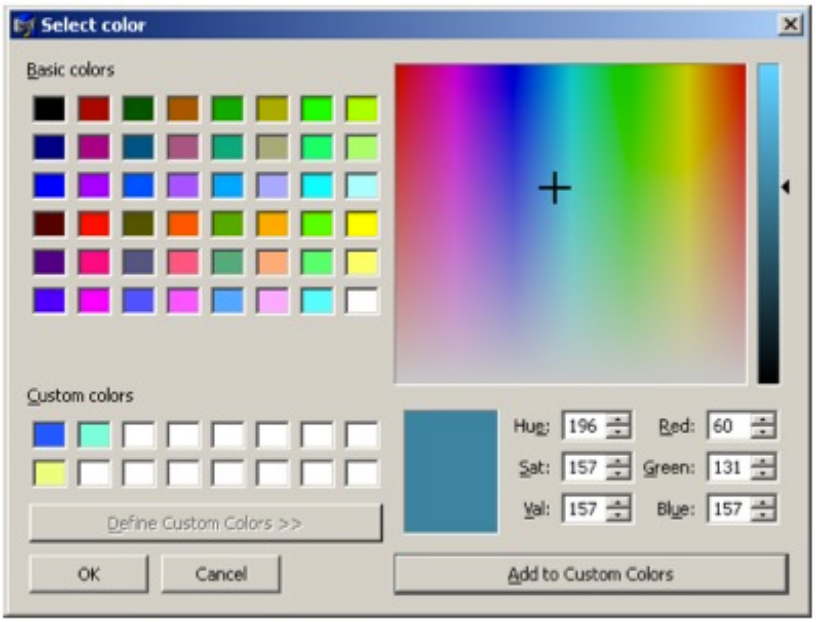


Figure 2.2 - Interface for selecting an individual color

Define a color that will be applied to all the points in the selected 3D entity. The choice is manual, and is made using a classic interface allowing various modes of selection (figure 2.2):

- One can choose a basic color (at the top left), or a previously saved color (*custom colors* - at the bottom left)
- or one can click on the colored zone (at the top right) and can vary the intensity using the sliding indicator at the right
- or one can manually enter the color parameters in the RGB or HSV fields (at the bottom right).

Keyboard shortcut: F8

2.2.2. Colors > Colorize

Uses the same interface as the "Set Unique" function (2.2.1). This function lets the user modify the current colors of the points (by multiplying the components of the current color with those of the selected color). If the entity does not have a color, then this function is equivalent to "Set Unique".

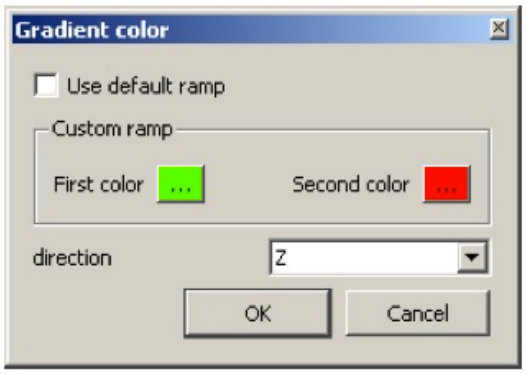


Figure 2.3 - interface to define a color gradient

Figure 2.4 - Default color gradient

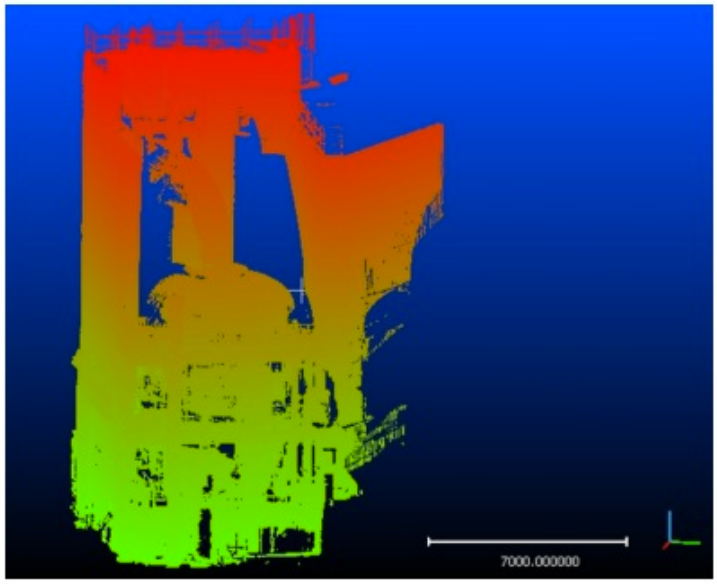


Figure 2.5 - Example of a red-to-green gradient applied on the Z axis

2.2.3. Colors > Height Ramp

The user can choose between applying the default color ramp (figure 2.4) or a custom ramp for which he can define the two extreme colors (figure 2.3). In the latter case, the user must deselect the *Use default ramp* box. The two colors at the extremes of the gradient can be chosen by clicking on the *First color* and *Second color* fields (which load the standard color selection interface as seen for the *Set unique* method – cf. section 2.2.1).

In either case, the user can choose along which of the three principal directions (refer to the triad in the bottom right of the display) to apply the gradient.

2.2.4. Normals > Compute

This function is no longer included in version 2.1 of CloudCompare.

This function calculates the normals (unsigned) of a point cloud that lacks normals, and can do this via different methods. To get a signed field of normals, use the *Resolve direction* function.(cf. section 2.2.6).

The two methods currently available are:

- LS: obtained by local adjustment according to the least squares method (fast method, but noisy)
- HF: obtained by interpolation of the points by a function of quadratic height (more robust but slower).

2.2.5. Normals > Invert

Invert the normals (for the components corresponding to the tridimensional unit vectors, we replace each by its vector opposite).

Note: this usually fixes problems that have to do with the sense (direct or indirect) of the mesh triangles, because the normal is not shared by all programs that process or generate meshes.

2.2.6. Normals > Resolve direction

This function is no longer included in version 2.1 of CloudCompare.

This algorithm resolves the correct sense of the normal vectors associated with a point cloud. The resolution is performed gradually, by propagating one or more fronts on the cloud (using the *Fast Marching* algorithm).

The propagation follows a 3D grid (here, an octree) and one must therefore choose the level of octree on which to apply the algorithm. Choosing a good parameter may take some trial and error, because too low a level will result in large cells, which propagate quickly but will miss local convolutions, while too high a level will result in the opposite. Furthermore, the more difficult the propagation - i.e. piece by piece - the higher the risk of having areas near each other take on

opposite senses. Try the algorithm at different levels of the octree, starting typically at 5 or 6, and raise the level until you find a satisfactory result. The algorithm is very rapid, and it can be executed as many times as needed (it doesn't modify the normals, only their sense).

Note: the resolution is very fine; it may still be necessary to use the Invert function (cf. section 2.2.5) to obtain the final desired result.

2.2.7. Octree > Compute

This function calculates an octree structure (recursive spatial subdivision) on top of a selected point cloud (or vertices, in the case of a mesh). Once the octree is successfully calculated, it is attached automatically on top of the selected entity (cf. section 1.4.6).

Note: the maximum level of subdivision of an octree is 10 in this version of *CloudCompare*.

2.2.8. Octree > Resample

Function for (coarse) point cloud re-sampling. The user can specify an approximate number of points (via the interface in figure 2.6), and *CloudCompare* will determine the level of octree having a number of cells closest to this value. The re-sampled point cloud will be formed by replacing each cell by a *representative* point (actually, the center of gravity of the points present inside that cell).

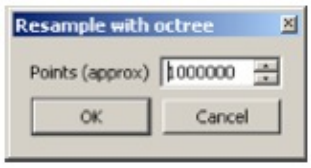


Figure 2.6 – Interface to select the number of points in the re-sampling

Note: this method is different than the *Subsample* method (cf. section 2.2.28) because it creates points, unlike *Subsample*, which selects points that already exist in the source cloud.

2.2.9. Mesh > Compute

This function calculates a mesh in 2.5D on top of a point cloud (via Delaunay triangulation). Because the method produces a 2.5D mesh from a 3D cloud, we must project it onto a plane before calculating the triangulation. So the user has two choices (the function is available in two versions):

- *Axis aligned plane*: by default, *CloudCompare* estimates that the altitudes are carried by the principal Z axis and the points are then projected onto the XY plane
- *Best LS plane*: a more generic approach, which projects the points on the best plane by interpolating the cloud in the least-squares sense (it is suited to clouds that are rather flat, with elevations not necessarily along the Z axis).

2.2.10. Mesh > Sample Points

This function randomly samples points on a surface described by a mesh (see figure 2.7).

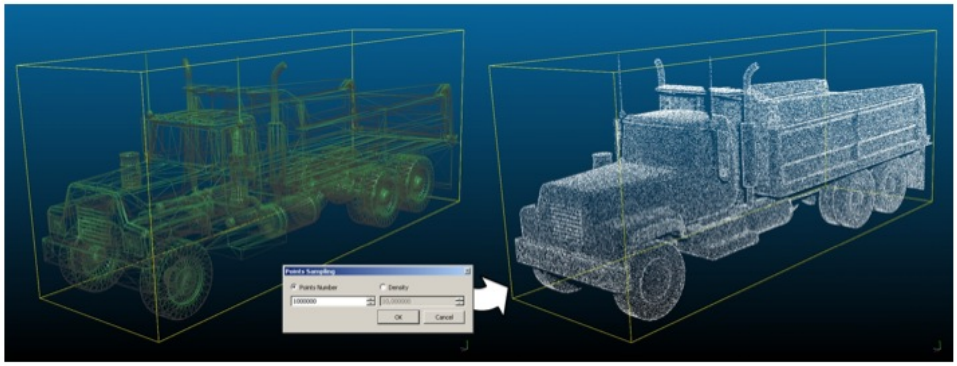


Figure 2.7 -- Illustration of sampling points on a mesh

A new point cloud is created. The user can specify, via the interface:

- the total number of points desired.
- or a density per unit on the surface. Be aware that the surface is expressed in the same units (squared) that is used for the coordinates of the mesh points. To get the total surface area of the mesh, call the function *Mesh* > *Measure surface* - section 2.2.11.

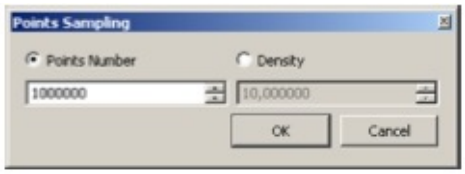


Figure 2.8 -- Interface to choose the parameters for sampling the points of a mesh

2.2.11. Mesh > Measure Surface

Calculate the surface area of a mesh.

This surface is expressed in the same unit (squared) that is used for the coordinates of the vertices of the mesh.

2.2.12. Mesh > Scalar Field > Smooth

Apply a smoothing of a scalar field that is associated with a mesh, by using the topology of the mesh. The scalar value assigned to a vertex is replaced by a mean of the values at the neighboring vertices.

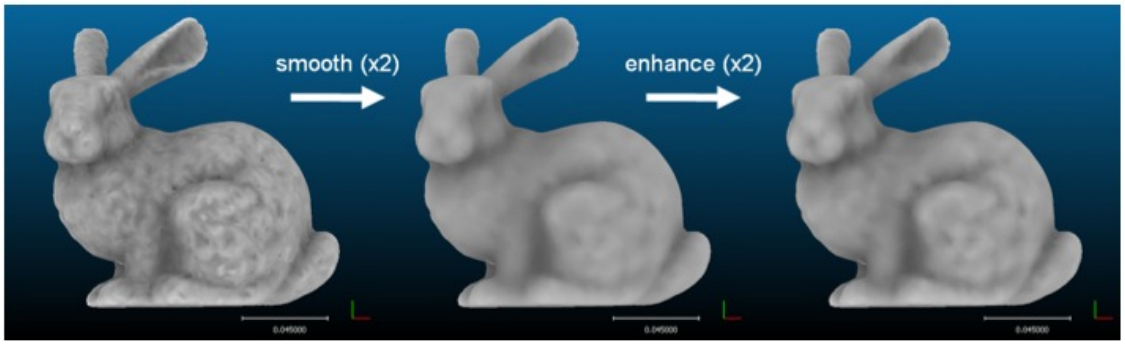


Figure 2.9 -- Example of results obtained with the *smooth* and *enhance* options on a scalar field over a mesh

Remark: this function is much faster than the *Gaussian filter* (section 2.2.19) for an approximate result, but on one hand it requires a mesh associated with the cloud of points (which correspond to the vertices of the mesh) and on the other hand, the size of the smoothing kernel cannot be changed.

2.2.13. Mesh > Scalar Field > Enhance

Applies a contrast enhancement on a scalar field associated with a mesh, by using the topology of the mesh. The scalar value at a vertex is modified to augment the contrast between it and the values at neighboring vertices.

This function is the inverse of *Mesh* > *Scalar Field* > *Smooth* (section 2.2.12).

2.2.14. Sensor > Ground-Based Lidar > Show depth buffer

This function is no longer part of version 2.1 of CloudCompare.

Display the depth map associated with a sensor.

2.2.15. Sensor > Ground-Based Lidar > Export depth buffer

This function exports the depth buffer associated with a sensor as a text file. It is applied to a cloud that "contains" the selected sensor (it is necessary to first select a sensor entity). The user is invited to specify a filename under which all the information relating to the depth map will be saved (see section A.1.3.2).

2.2.16. Sensor > Create

This function creates a sensor (a virtual representation of a laser scanner) associated with a cloud.

In general, the user wants to re-create the sensor that acquired the point cloud after the fact. In particular, this allows applying smart filtering when calculating the distance between two point clouds (cf. section 2.3.5).

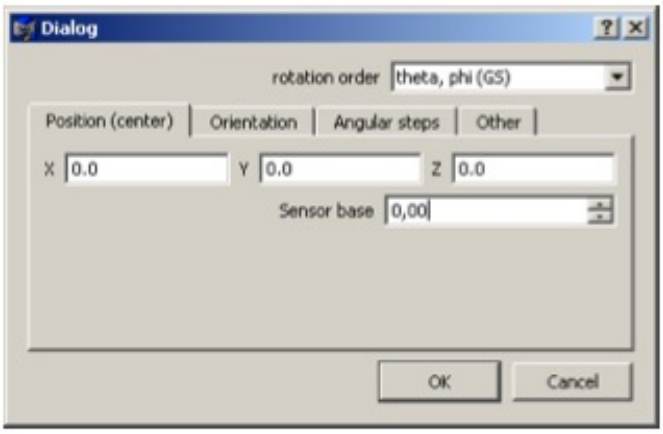


Figure 2.10 -- Configuration interface for creating a sensor

When creating a sensor, numerous parameters can be configured (via the different tabs in interface 2.10):

- *rotation order*: order of rotations of the scanner (motor, mirror). Here we use the angles θ and φ as per the usual conventions for spherical coordinates: θ represents the horizontal angle (the declination) of the scanner, φ represents the vertical angle. There are now two choices: θ then φ (Trimble GS type) or φ then θ (Trimble SOISIC type).
- *Position (center)/(X,Y,Z)*: XYZ position of the optical center of the scanner (expressed in the reference frame of the point cloud)
- *Position (center)/Sensor base*: gap between the laser emitter and the receptor (used for triangulation captures such as typical SOISIC)
- *Orientation*: Reference point of the sensor expressed from the reference point of the cloud (three vectors). By default, the matrix formed by these three vectors is left as the identity matrix, which makes it aligned with the current reference point.
- *Angular steps/dPhi*: angular step (in degrees) of the sensor in the φ direction.
- *Angular steps/dTheta*: angular steps (in degrees) of the sensor in the θ direction.
- *Other/Uncertainty*: Uncertainty in the laser measurement, in percentage (calculated automatically from the projection).
- *Other/Max. range*: The maximum scope (deduced automatically from the projection).

Once the parameters are entered, *CloudCompare* creates a sensor object that it will associate with the cloud. This sensor object contains, among other things, a map of the depth of the cloud (obtained by projecting the points along the reference angle of the sensor - and equivalent to a *Z-buffer*) which permits smart filtering when calculating the distances as mentioned above, but which may also be viewed (section 2.2.14) or exported (2.2.15). Finally, the sensor object can be displayed in place in the form of a small schematic of a 3D sensor (see figure 2.11).

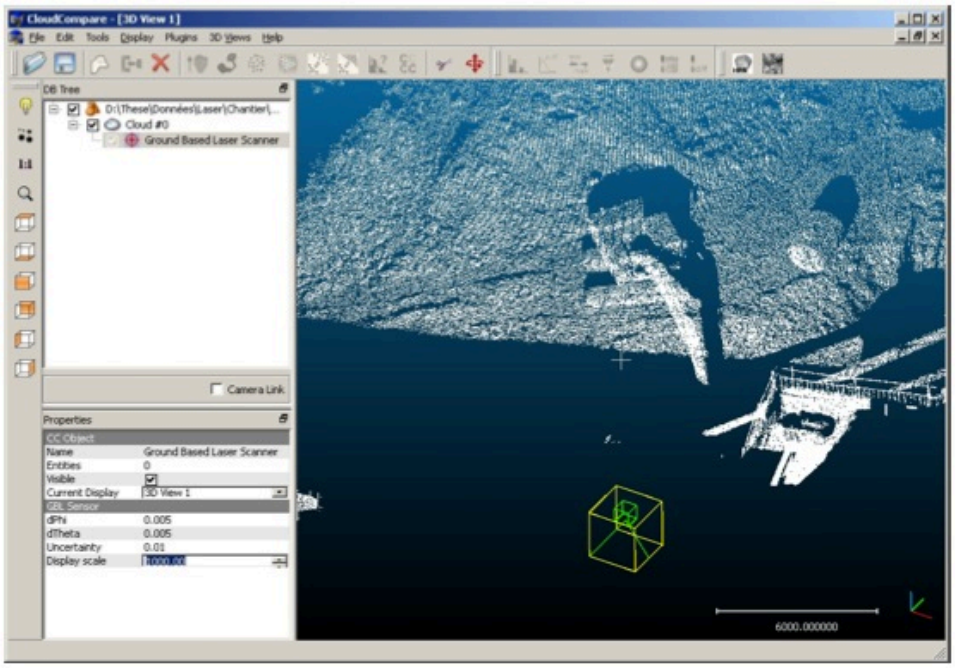


Figure 2.11 – *Sensor* object displayed *in situ*

2.2.17. Sensor > Modify

Configure the parameters of the sensor.

The same dialog box as was shown in figure 2.10 for the creation of a sensor object appears for its modification, but here, the configuration values of the sensor correspond to those which were captured during its creation (and not the default parameters).

2.2.18. Scalar Fields > Gradient

This function calculates the magnitude of the gradient of the active scalar field.

When this function is called, *CloudCompare* asks the user to specify if the scalar field corresponds to a Euclidean distance (such as distances calculated between two clouds or between a cloud and a mesh). If yes, the algorithm will filter the outlying values, which in this case are easily detectable (in fact, in this case the gradient in absolute value may never exceed 1 - c.f. D. Girardeau-Montaut's thesis).

Remarks:

- The algorithm creates a new type of scalar field (Gradient norms).
- As in classic 2D image processing, the gradient notably highlights zones of strong variation in the scalar field, which may be taken as evidence of a zone boundary - see figure 2.12).
- As in 2D image processing, it is often necessary to apply a Gaussian filter to the data before and/or after calculating the gradient (cf. section 2.2.19).

- The fact that the absolute value of the gradient is never greater than 1 is true in reality for all scalar fields in which the values vary proportionally to the distance between the points (it is thus for the case of a distance field).

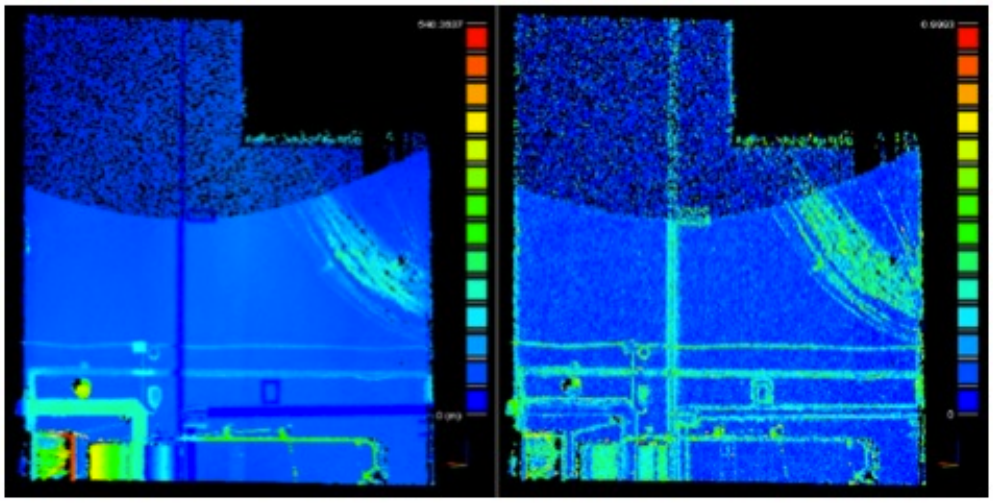


Figure 2.12 – Configuration interface for calculating the normals

2.2.19. Scalar Fields > Gaussian Filter

Apply a Gaussian filter to the active scalar field.

The only parameter to define is the size of the sigma kernel of the Gaussian filter. To quickly estimate this parameter, one can refer to the octree, taking for the size of the kernel the size of a cell at level 8 (for a soft filter), 7 (for a relatively strong filter), etc. The size of a cell is displayed at the level of the console when an octree rendering is displayed (cf. section 1.4.6).

Remarks:

- Sigma gives us the radius of the sphere in 3D that delimitates the "neighborhood" which will be considered around each point. What is actually calculated for each point is the weighted average of the scalar values of the neighbors, weighted by their distance according to a Gaussian law. Given that 3σ corresponds to an overwhelming 99.9%, it is not useful to consider more distant points.
- The larger the kernel, the slower the calculation will be.
- This function is very useful for smoothing the result of a gradient calculation (section 2.2.18) but also for a Portion of Visible Sky calculation (section 2.5.1) for example.

2.2.20. Scalar Fields > Filter by Value

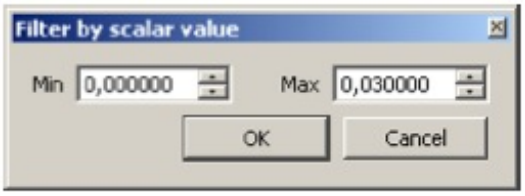


Figure 2.13 -- Configuration interface for filtering points by the value of their scalar field

This function segments a cloud by defining an interval of scalar values (figure 2.13). A new cloud will be created with all the points for which the scalar values (for the active scalar field) are included in this interval.

2.2.21. Scalar Fields > Difference

This function is no longer part of version 2.1 of CloudCompare.

Tool for calculating the point-to-point differences between two scalar fields applied to identical clouds. This function requires two spatially identical clouds, each with an applied scalar field, and calculates the differences between the values of homologous points.

This algorithm requires selecting two (and only two) clouds. *CloudCompare* asks the user to define the role of the clouds, via the generic interface for choosing roles - see section 2.3.4. The calculation is signed, and the order for subtraction to take place, their order must be defined ($field_{diff} = field_A - field_B$). The first cloud (cloud *A*) will receive a new scalar field corresponding to this difference.

Remarks:

- The term "identical" is used loosely. It suffices that all the points in cloud *A* are found in cloud *B*. That is, cloud *A* must be a subset of the points in cloud *B*.
- The algorithm creates a new type of scalar field (*Diff*), which is signed.

2.2.22. Scalar Fields > Multiply

This function is no longer part of version 2.1 of CloudCompare.

This function multiplies the scalar fields of two clouds for the selected points. The two clouds must have the same number of points. The scalar field of the first is updated with the result of the multiplication.

Remark: to call this function, one must select two (and only two) point clouds with scalar fields.

2.2.23. Scalar Fields > Convert to RGB

Assigns a color to each point from the current scalar field values and the current display parameters.

If the entity already has been assigned a color, it is possible to remove the existing color or to mix the two values (*CloudCompare* asks the user when needed).

2.2.24. Clone

Creates a new 3D entity that is point-for-point identical to (but independent from) the selected entity. A modification to the cloned object has no impact on the original (and vice versa).

2.2.25. Fuse

Fuse two (or more) selected entities.

Note: the fused lists are minimized when the function is executed.

All characteristics of the 3D entities are conserved. Lists that do not possess a particular characteristic will be given default elements (no normals, white color, etc.).

2.2.26. Translate

Apply a translation to the selected entity or entities. The user supplies the three components of the translation (X, Y, and Z) via an interface.

2.2.27. Multiply

Multiplies the coordinates of the points of the selected entities by constants. The user chooses the 3 coefficients to multiply each component along its axis (f_X, f_Y, f_Z) via a dialog box.

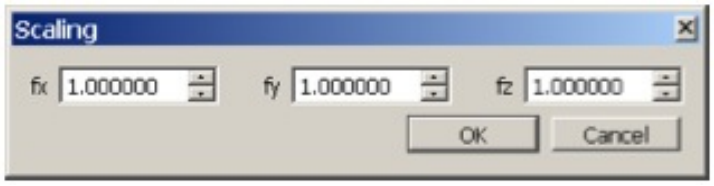


Figure 2.14 – Dialog box for multiplication of coordinates

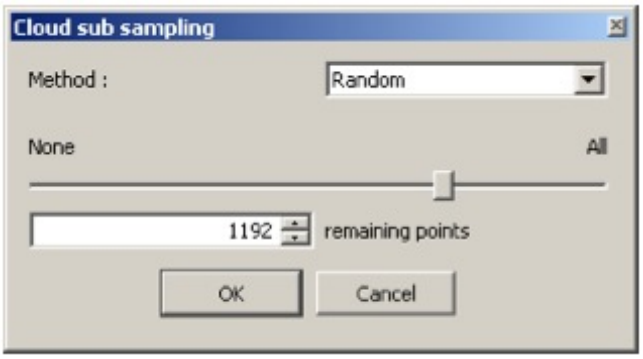


Figure 2.15 – Configuration interface for sub-sampling of clouds

2.2.28. Subsample

This function creates a subsample of a point cloud. Different methods are now available. The choice (as well as the configuration) is accessible via the dialog box shown in figure 2.15:

- Random: random subsampling
- Space: spatial subsampling (the resulting local density is constant and configurable)
- Octree: rapid subsampling via the octree (keeping one point per cell of the octree at the given level of subdivision)

Remarks:
- The subsampling differs from re-sampling (cf. section 2.2.8) in the sense that it doesn't create new points but only selects a sub-set of points from the source cloud.
- The quick method of subsampling via the octree (cf. section 2.2.8) chooses the point closest to the center of each cell. Therefore the distance between the points is closer to constant (if the initial cloud is sufficiently dense)

2.2.29. Synchronize

Aligning two selected entities: this function simply applies a translation to make the centers of gravity of the two entities coincide.

Remarks:
- To call this function, it is necessary to have selected two (and only two) entities.
- Again, we use the generic interface for choosing the role of each entity (see section 2.3.4), which allows the user to specify which is the entity to displace and which entity is the reference.

2.3. Tools Menu

2.3.1. Tools > Projection > Unroll

This function will "unroll" a point cloud from a cylindrical (or conical) shape onto a plane. See figure 2.17.

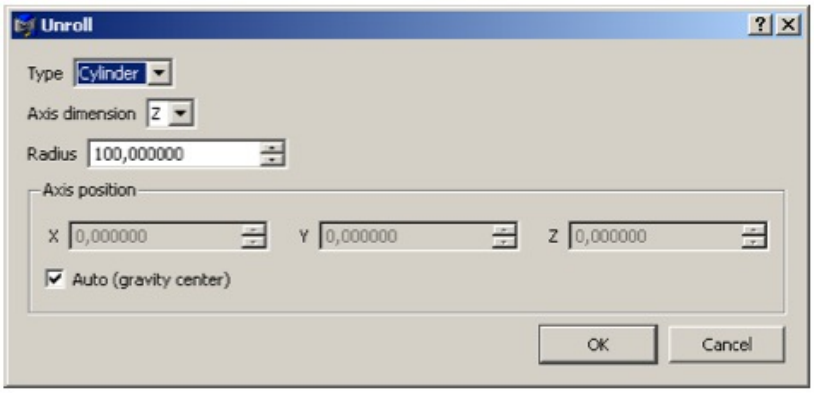


Figure 2.16 – Configuration interface for the cloud unroll tool

To do this, supply the parameters that define the object of revolution:

- the type (cylinder or cone)
- the dimension along which the axis is positioned (must be X, Y, or Z axis for the moment)
- a point through which the axis passes (if the auto axis checkbox is checked, the point used will be the center of gravity of the cloud.
- the radius of the cylinder or base of the cone
- and the opening angle of the cone in the applicable case

Warning: to optimize memory usage, this function applies the transformation directly on the selected entity. It may be necessary to use the clone tool first - see section 2.2.24.

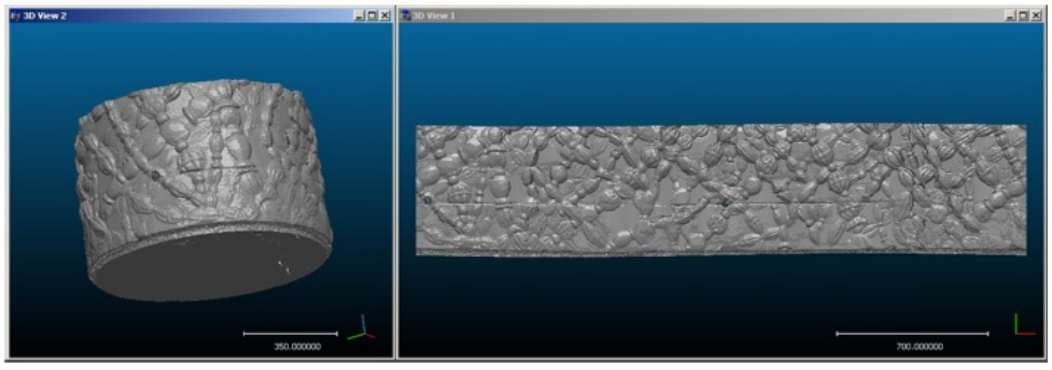


Figure 2.17 – Example of results: a cylindrical point cloud (left) and its unrolled version (right)

2.3.2. Tools > Projection > Height Grid Generation

This function projects a point cloud onto a flat grid normal to the Z-axis.

An interface (figure 2.18) allows the user to choose various parameters:
- *grid step*: size of a grid section expressed in the units of the point cloud coordinates
- *type of projection*: this parameter may take one of the following two values:
  - *maximum height*: Let E_{ij} be the subset of cloud points that is projected in the (i,j)th square of the grid. For each grid square (i,j), we keep the Z altitude of the highest point in E_{ij}.
  - *average height*: for each grid square (i,j), we keep the mean Z-altitude of the points in E_{ij}.
- *fill empty cells with*: some grid squares may remain empty after projection (no cloud points were projected there). This parameter indicates the value with which to fill these squares and can take one of the following three values:
  - *minimum height*: the empty squares are filled with the minimum Z altitude of all the points in the cloud.
  - *average height*: the empty squares are filled with the mean Z altitude of all the points in the cloud.
  - *maximum height*: the empty squares are filled with the maximum Z altitude of all the points in the cloud.

This function creates two files (located by default in the binary folder of *CloudCompare*):
- *height_grid_image.tiff*: the 2D raster image coded in 256 shades of grey corresponding to the Z-altitudes projected in the squares of the grid;
- *height_grid_text_file.txt*: the height data of the grid in ASCII format (easily read by a program or script).

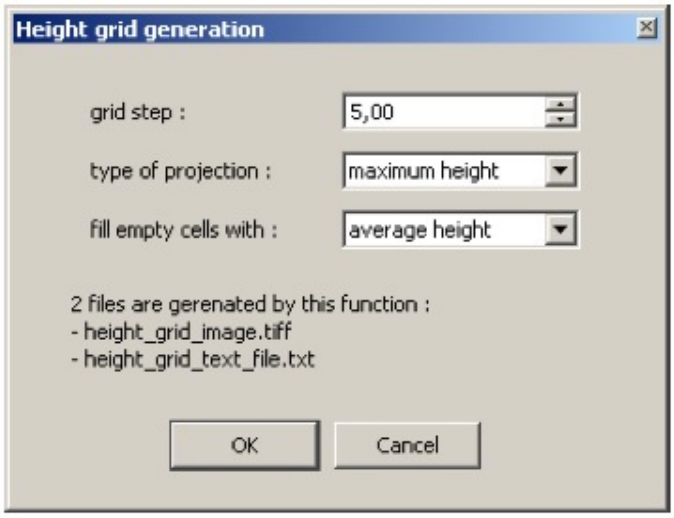


Figure 2.18 – Interface for configuring the height grid generation tool

See figure 2.19 for an example of the results produced by this function.

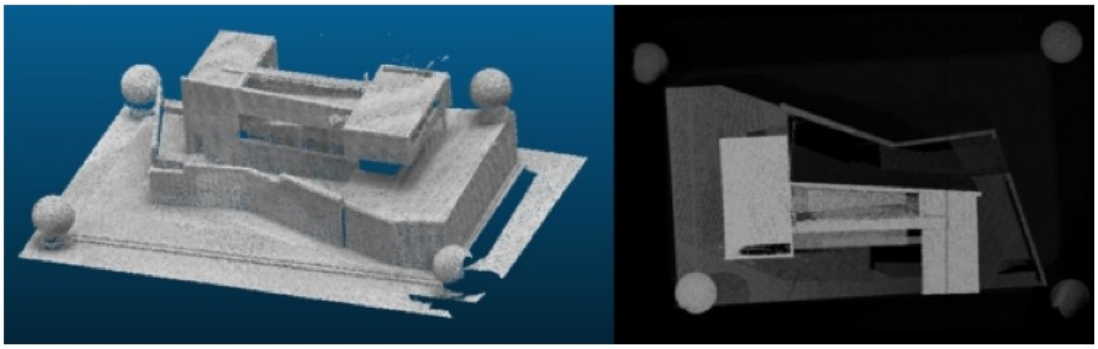


Figure 2.19 – Example results: 3D view at left, 2D height image at right

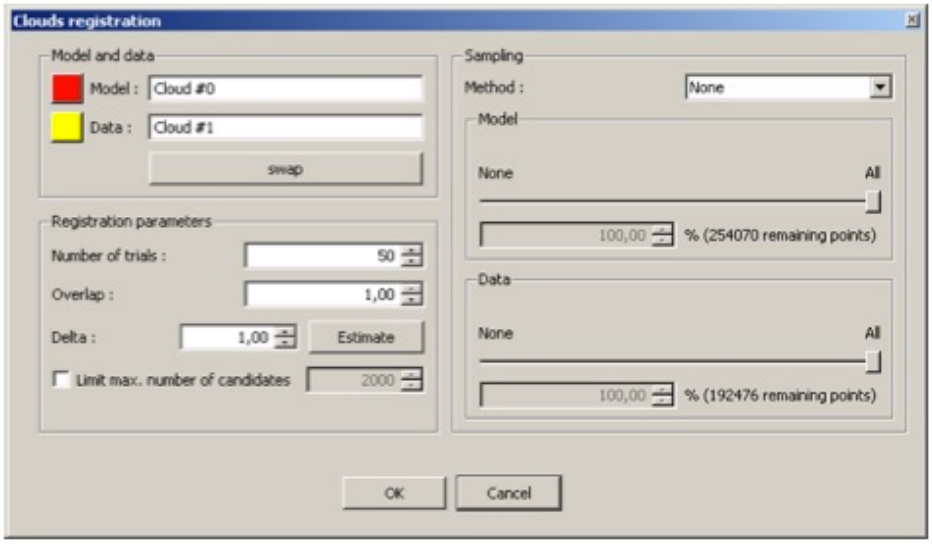


Figure 2.20 – Configuration interface for the coarse registration between two entities

2.3.3. Tools > Registration > Align

This function coarsely aligns two point clouds (using the "*4 points Congruent Sets For Robust Registration*" algorithm by Aiger, Mitra and Cohen-Or, Siggraph 2008).

A first input box (top left) lets the user specify the two clouds and their respective roles (*Model* and *Data*: The *Model* is the cloud of reference (which does not move) on which the program will align (if possible) the *Data* cloud (which will be moved if necessary). As the alignment calculated is a rigid alignment, only translations and rotations can be applied to the *Data* cloud. Several other parameters must be entered for optimal use of this method:

Sampling: this area relates to the prerequisite subsampling of the point clouds. This can give a perceptible improvement in the efficiency of the algorithm. Indeed, tens of thousands of points are generally sufficient to obtain a good registration, while the complexity of the algorithm increases rapidly with the number of points. The user should always seek to minimize the number of points taken into account, restarting the algorithm with more points if needed. Here are the parameters of the sub-sampling:

- *Method*: subsampling method (see section 2.2.28). Select *None* to not sub-sample
- *Model*: a slider and/or a variable field allows the user to choose the number of points kept for the reference cloud
- *Data*: same as above, a slider and/or a variable field allows the user to choose the number of points kept for the adjusted cloud

Registration parameters: this zone corresponds to parameters of the registration algorithm itself. We will explain these parameters in detail:

- *Number of trials*: The algorithm proceeds by successive iterations and only retains the trial that produces the best result. This field lets you choose the number of trials to perform. The higher the chosen value, the longer the calculation will take, but the better the probability of obtaining good results. It may therefore be necessary to adapt this parameter as a function of the number of points in the clouds to obtain a good alignment in a reasonable time. To give a general idea, to align two clouds of 5000 points each, about fifty tries will produce a decent result in a few minutes (about 2-5 minutes, depending on the speed of the computer).
- *Overlap*: This parameter, which can be between 0.0 and 1.0, corresponds to an estimation of the overlap between two clouds when they are correctly aligned. An overlap of 1 signifies that the two clouds overlap completely; 0 signifies that they are disjoint (in this case, alignment doesn't make much sense). A very approximate estimation of overlap is generally sufficient.
- *Delta*: This parameter is an estimation of the mean distance between points in the two clouds when they are correctly aligned. This parameter functions like an error tolerance: the closer it is to 0.0, the closer we constrain the clouds to be, but the likelihood of finding a good solution is lowered. In principle, if the Delta is zero, the program can never find an alignment between the two clouds. As a general rule, to obtain good results, Delta should correspond to the resolution (inverse of the density) of the reference cloud. The interface contains an *Estimate* button that automatically estimates the parameter based on the mean density of the reference cloud.
- *Limit max. number of candidates*: when this field is activated (the checkbox is checked), it is possible to constrain the maximum number of candidates that the program may calculate for each trial. In the "*4-points Congruent Sets*" algorithm, the program chooses a set of 4 coplanar points in the reference cloud, and seeks congruent sets (under translation and rotation) within the data cloud (candidates). These sets are calculated as a function of the previously mentioned parameters, and the program may find an enormous number of candidates (hundreds of thousands of sets). This parameter allows the algorithm to select only those candidates which are considered to be the "best", and can therefore considerably shorten the calculation time for each trial. As a result, you potentially deprive yourself from finding the best alignment as a result of the heuristic used to retain the best candidates. When this field is deactivated, the maximum number of candidates is unlimited, which can result in a very long calculation time.

The *Delta* and *Overlap* parameters require that the user have a prior idea of how the clouds will be after alignment.

Figure 2.21 presents the results obtained from aligning two scans of a toy taken from two substantially different angles. In theory, the *Align* function can be applied to clouds with much less overlap than is shown in the example.

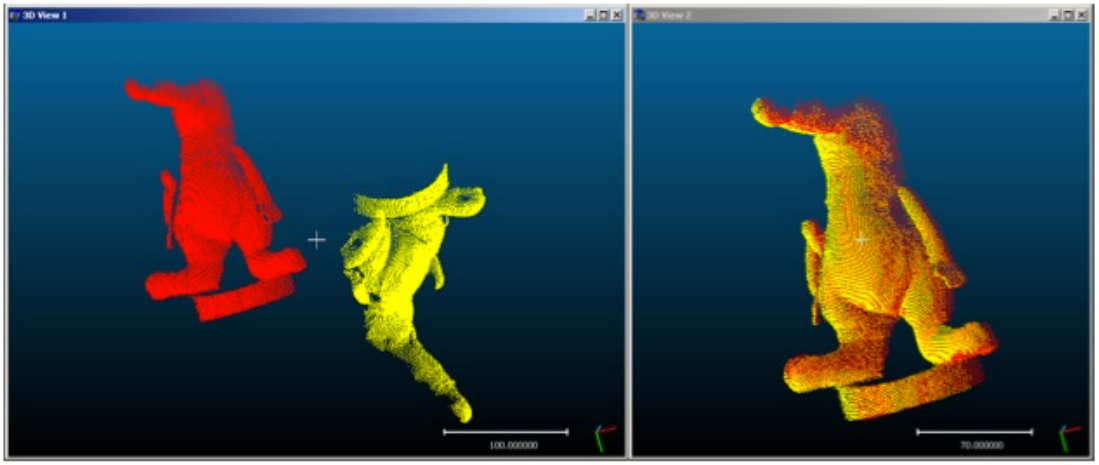


Figure 2.21 – Alignment of two clouds with partial overlap. On the left is the initial configuration, and on the right is the result of alignment with an estimated overlap of 90% (0.9) and 20 iterations

The calculated alignment using this function depends largely on the configuration of the initial clouds. Their geometry and degree of resemblance will make them more or less difficult to compare. Therefore, it is possible that in some cases, the results will be fairly poor. In these situations, you can then use the fine alignment function described in section 2.3.4. It is recommended, in the general case, to resort to the fine alignment after using this function.

This function creates a copy of the Data cloud aligned with the Model cloud. It is therefore not necessary to clone the clouds beforehand, since they will not be modified directly.

2.3.4. Tools > Registration > Register

This function aligns two point clouds (using the "*Iterative Closest Point*" algorithm by Besl and McKay, IEEE Trans. PAMI 1992).

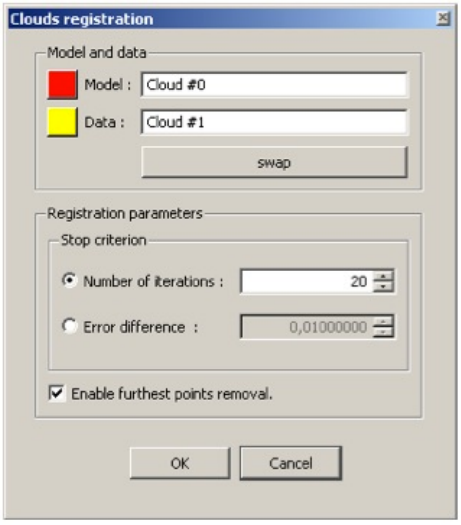


Figure 2.22 – Configuration interface for the registration tool

Warning: for this registration to work, it is necessary that the two clouds be more or less aligned.

This function cannot align clouds that are positioned and oriented haphazardly. Its role is primarily to refine the registration of two clouds that are roughly aligned. The *Align* function described in section 2.3.3 can ensure that the clouds are approximately aligned, and the *Register* function can be used on the resulting clouds.

In the top zone (*Model* and *Data*) of the configuration window, the user may interactively assign roles to each entity. The *Model* is the reference cloud (which does not move) and the *Data* cloud is the one that will be registered (and will move if necessary). To help the user, *CloudCompare* colors the entities in their display (*Model* in red and *Data* in yellow) in the same way as in the interface for choosing roles when calculating distances (see paragraph below). The *Swap* button switches the roles if needed.

The lower part (*Registration parameters*) corresponds to the parameters of the algorithm itself. In detail:

– *Stop criterion*: the user chooses either a fixed number of iterations (this prevents an excessively long calculation times, but does not guarantee the quality of registration), or a minimal decrease in the error between two iterations to justify more iterations (which guarantees a better quality but potentially may take a long time).

– *Enable furthest point removal*: heuristics adapted to align entities that differ only slightly (while *CloudCompare* is made to compare potentially different point clouds, this algorithm assumes the clouds represent the same objects!). This heuristic removes the farthest points at each iteration of the alignment, to avoid that the differences between the clouds pull the final position from its proper alignment.

Therefore this option should not be checked if the two clouds represent the same object.

Choice of roles (generic interface)

This generic interface (figure 2.23) is used by all the distance calculation methods, and a few other methods (which use it in a similar manner). It lets the user interactively assign a role to the two entities which are selected at the same time. *CloudCompare* colors the entities according to their assigned roles. In the distance case, for example, the reference cloud is represented in yellow and the cloud to compare (which will be assigned a scalar field after calculation), in red. A *Swap* button will invert the roles (and therefore the colors) of the two entities.

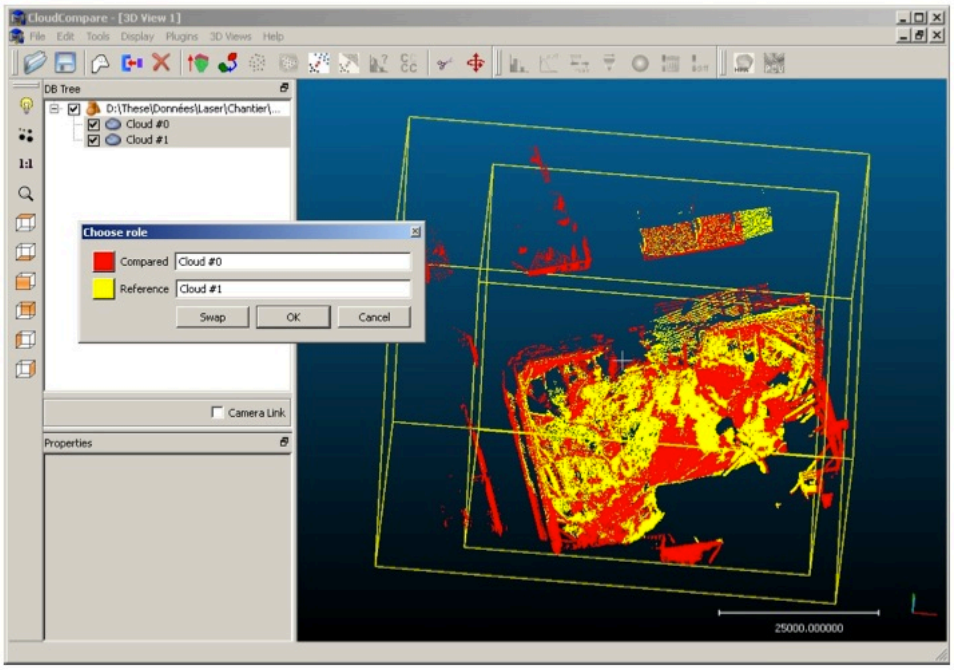


Figure 2.23 – Standard interface for choosing the roles of entities

2.3.5. Tools > Distances > Cloud/Cloud dist.

This function calculates the distances (approximate or exact) between two clouds of points.

When this function is called, and after having chosen the role of each cloud (see section 2.3.4), a first approximate distance calculation between the clouds (chamfer distance, calculated via the octree) is automatically performed. When this is done, the upper part of the display in figure 2.24 (*Approx. results*) shows various distance statistics that can help configure the refined distance calculations.

This information includes:

- *Min. dist*.: approximate minimal distance
- *Max. dist*.: max. approximate distance
- *Mean dist*.: approximate mean distance
- *Sigma*: standard deviation
- *Max relative error*: maximum relative error of the approximation (expressed as a function of *d*, the distances, because this error is dependent on the actual distance between the points, and generally decreases rapidly when *d* grows, which means that the approximation of the minimal distance is generally very bad, but that of the maximal distance can be reliable enough.

The user can lastly display the histogram of approximate distances (by clicking the  icon), but these are usually not very detailed given that they are chamfer distances calculated via the octree.

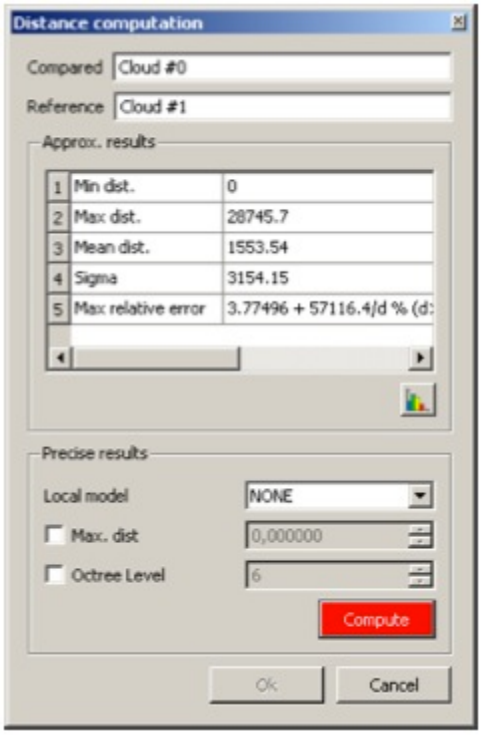


Figure 2.24 – Configuration interface to calculate the distances between two point clouds.

The lower part (*Precise results*) configures the precise distance calculation. The user can choose the following values:

- *Local model*: indicates that a local model will be applied to the reference cloud to improve the precision of the cloud-to-cloud distance calculation (to a certain extent). This technique improves the overall precision (but not necessarily the local) This improvement depends on the chosen model, and has the cost of slowing the calculation (the extent of which depends again on the chosen model):
  - *NONE*: no local modeling (default), calculates distances from point to nearest point.
  - *Least Square Plane*: local approximation of the cloud by a plane (adjusted to least squares) – less precise but fast.
  - *2.5D triangulation*: local approximation of the cloud by a 2.5D Delaunay triangulation (after projection of the points on a plane adjusted to least squares) – intermediate speed and precision.
  - *Height Function*: local approximation of the cloud by a height function of the type $z = ax + by + cx^2 + dy^2 + exy$ (here again, after a projection of the points onto a least-squares plane) – best precision but reduced speed.
- *Max. dist*: define a distance above which it is not necessary to calculate a precise distance. This may greatly improve the calculation's performance, in particular on clouds having few zones in common (by avoiding calculating the far distances - the most costly - while their precise knowledge is generally not useful). The points in question retain their approximate distances. The information displayed in the upper part may be of help in choosing this limiting value.
- *Octree level*: this parameter of the algorithm is normally optimized automatically by *CloudCompare*, but it is possible to override it in cases where the determining heuristic fails.

Remarks:
- This function adds a scalar field of *C2C* (cloud-to-cloud) *distances* to the reference cloud.
- To calculate the precise distances, click the red Compute button. Otherwise, only the approximate distances are saved.
- All the distances calculated by this function or entered as parameters are expressed in the same unites as the coordinates of the point cloud (there is no explicit unit in *CloudCompare* 2.1).

2.3.6. Tools > Distances > Cloud/Mesh dist.

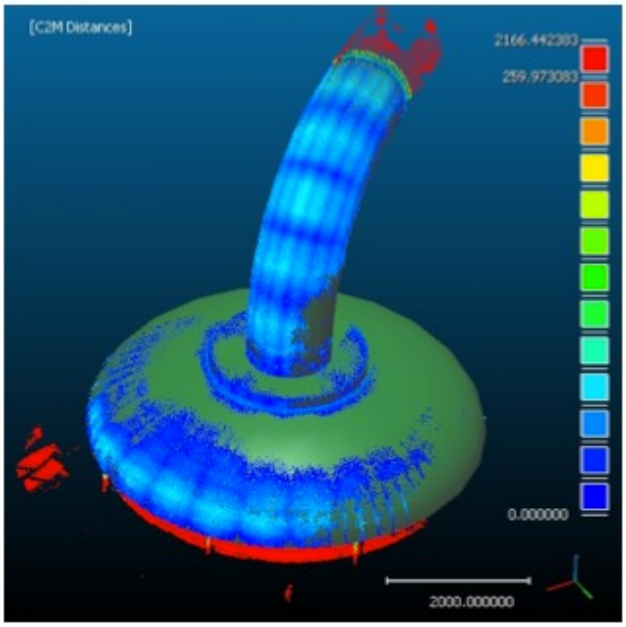


Figure 2.25 – Example of the results of calculating the distances between a cloud and a mesh

This function calculates the distance (approximate or exact) between a cloud of points and a mesh.

This function is largely equivalent to the distance calculation between clouds (section 2.3.5) except for a few details:
- if one of the two selected entities is a mesh, the choice of roles (section 2.3.4) is not necessary (the mesh is assigned as the reference entity).
- if the two selected entities are meshes, the vertices of the *compared* entity will be used as the *cloud*. It may be of interest to use the sampling function for points on a mesh (section 2.2.10) beforehand, to have a better view of the differences between meshes (if this is the expected result).
- this function adds a *C2M Distances* (Cloud-to-mesh) scalar field to the reference cloud.
- The choice of a local model is not possible because the reference entity here is already a mesh.

2.3.7. Tools > Distances > Closest Point Set

This function calculates the closest point in the *compared* cloud for each point in the *reference* cloud. The set of these "closest points" forms a new cloud (Closest Point Set or CPS).

Remarks:

- To call this function, you must have selected exactly two point clouds.
- We see again the generic role choice interface (cf. section 2.3.4, which lets the user choose the cloud from which to extract the CPS points (*compared* cloud) and which is the *reference* cloud.
- The result is a cloud which has exactly the same number of points as the reference cloud, and in which each point is a member of the compared cloud (by definition). By construction, there may be duplicate points. This result is used, for example, by the alignment algorithm between two point clouds (cf. section 2.3.4).

2.3.8. Tools > Statistics > Compute stat. params

This function calculates the parameters of the chosen statistical law (Gauss or Weibull) from the values of the active scalar field of the selected cloud. The function typically returns the mean and standard deviation of the current scalar field if the Normal law is used, or the parameters (*a,b*) for a Weibull law (in which case *CloudCompare* also displays estimates of the mean and standard deviation in the console – see section 1.1).

The method graphically shows the adequacy between the calculated law (white line) and the histogram of the scalar field in a window that appears at the end of the calculation (see figure 2.26). The values of the parameters of the law are displayed at the top of this window. *CloudCompare* returns lastly, in the console, the $\chi 2$ distance between the estimated distribution and the values of the scalar field.

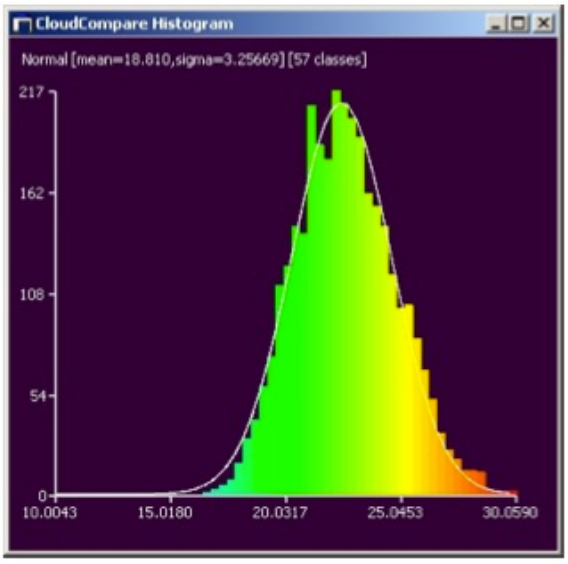


Figure 2.26 – Example of the automatic estimation of the parameters of a normal fit for a scalar field

Remark: the parameters of the law thus estimated can typically be used in the local statistical test function (see section 2.3.9), which permits a point cloud to be filtered based on the distances to a reference cloud (or mesh).

2.3.9. Tools > Statistics > Statistical test

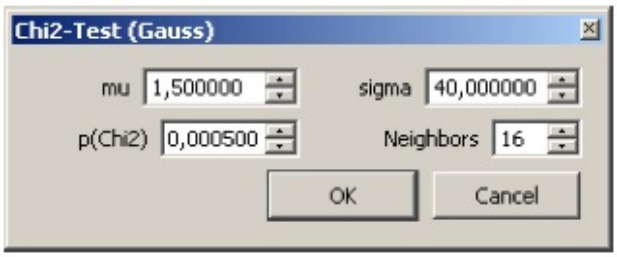


Figure 2.27 – Example of an automatic estimation of the parameters of a normal law for a scalar field

This function, central to *CloudCompare*, lets you apply a local χ2 test on a point cloud that has a scalar field. The χ2 test is applied to each point from the histogram of scalar values of its *n* neighbors (*n* being one of the parameters of the algorithm). The test compares this histogram with a theoretical distribution with two parameters (μ and σ in the case of a normal law, for example).

Before specifying the parameters, the user should choose the type of theoretical distribution (currently, the choice is between Gauss and Weibull). The result is a new scalar field (one value for each point - the χ2 value - which gives an indication of the local concordance between the scalar value and the tested distribution). The theory of the χ2 test gives us a threshold (calculated from the margin of error *p*(χ2), final parameter in the algorithm) which classifies points as a function of their non-adherence to the tested law. This law will typically represent measurement noise, and we obtain from it a set of points where the distance (to the other mesh/cloud) is not accounted for by the measurement noise. From this, we have the points that have actually undergone a change, and we can ignore points that have not actually changed, but have non-zero distance measurements due to noise. Once the cloud is separated into these two classes, one can keep the group of points outside the distribution (see figure 2.28, red) and segment, for example, as a function of the relative proximity of the points (by extracting the various connected components – cf. section 2.3.10).

Remarks:

- To call this function, select a single 3D entity with an active scalar field.
- When adjusting the parameter p(χ2), it is important to understand that the χ2 test only allows us to reject the hypothesis that *the values of the scalar field in the neighborhood of each point follow the tested law*, but not the inverse. So, the smaller the margin of error, the higher the threshold of χ2 will be (thus, we reject the aforementioned hypothesis less often, and we class fewer points as *not following the tested law*, [i.e. fewer points show actual displacement]).
- Inversely, the larger p(χ2), the more points will be found "outside the law", colored in red. Note that this parameter serves uniquely to pre-position the sliders for adjustment of the colors (cut-offs and saturation values of the scalar field) for displaying the result on the screen (cf. section 1.4.2.2). These sliders can then be moved by the user before actually extracting points by calling the function *Scalar Fields* > *Filter by Value*. This will create a new point cloud composed of only the points presently displayed on the screen, which is to say,

the points that do not follow the theoretical distribution. [In practical terms, these are the points that showed meaningful distance measurements, when you take noise into account.] Furthermore, the $\chi 2$ distance is extremely divergent [amplifies significant differences in the original] and this gives a lot of flexibility to the algorithm. Therefore, a relatively large change to the cutoff value will only have a small effect on the classification. At worst, we risk missing a small number of points (at the edges of the boundary zones).

- On the other hand, to obtain precise results, one must know or measure the distribution of measurement noise (as a first approximation it can be modeled as the average noise, including the measurement error due to the sensor, due to the surface that was scanned, to the lighting, to the ambient temperature during the measurement, to the creation of the mesh in the case of comparing clouds/meshes, etc.). The parameters of the statistical distribution can therefore be defined from a priori knowledge, but can also be determined from a scalar field (typically, a portion of a cloud) with the function that computes statistical parameters of a scalar field (cf. section 2.3.8).
- The algorithm creates a new scalar field named "*Chi2 Distances*". This field is assigned to the current point cloud.

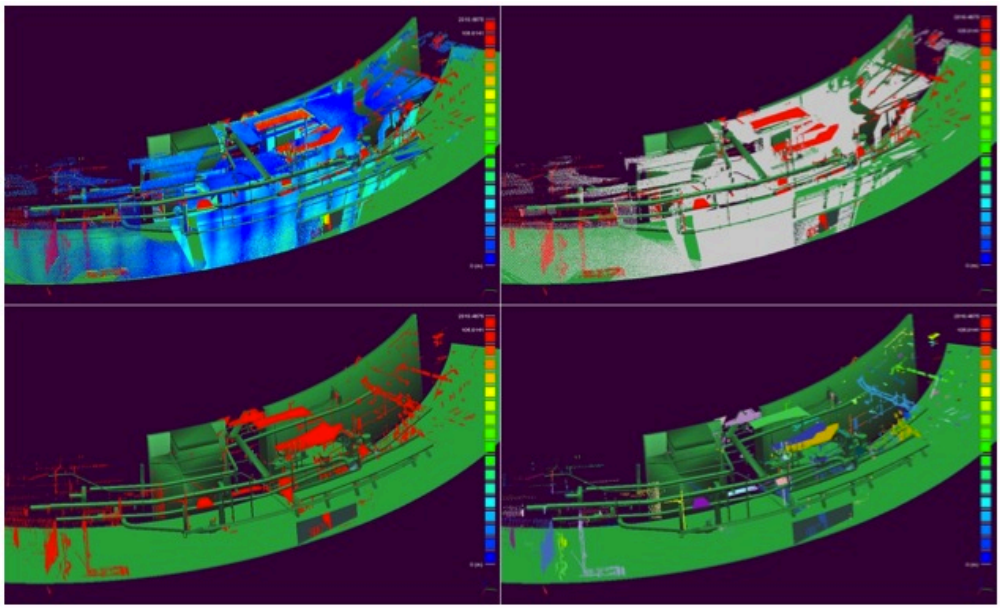


Figure 2.28 – Field of initial differences (top left), statistical filtering (top right), followed by extraction of points out of the theoretical distribution (bottom left), and finally the extraction of the connected components (bottom right)

2.3.10. Tools > Segmentation > Label Connected Components

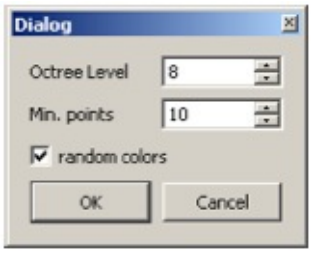


Figure 2.29 – Configuration interface for extracting connected components

This function decomposes a point cloud into compact sub-clouds. If the selected cloud is composed of multiple groups of points sufficiently disassociated (distant) from the others, it is possible to subdivide it fairly simply via the octree. This is done in *CloudCompare* thanks to an approach for *extracting connected components*. This is a common algorithm, generally applied to 2D binary images and which as been extended here to a 3D binary grid. This function outputs one entity per sub-cloud of points (grouped in a new entity group in *CloudCompare*'s navigation tree). Figure 2.28 (bottom right) is a good illustration of its use.

The user usually chooses the octree level at which the algorithm will be applied. This will roughly define the distance threshold beyond which groups of points (*connected components*) will be considered disjoint. The larger the octree level, the smaller the distance threshold will be, the more subgroups will be extracted (which is not necessarily desirable).

A second important parameter is the minimal number of points per connected component (*Min. points*). If a group is composed of fewer points than this number, then it will not be extracted as a new entity. This limits the number of clouds created by the algorithm.

Finally, the *Random Colors* option tells *CloudCompare* to generate random colors for each new cloud.

Remarks:

- The larger the octree level, the more memory (RAM) is required. The octree level is therefore a sensitive parameter which is difficult to adjust a priori, without experience. An approach by trial and error might therefore be necessary (typically, starting at level 7). We can also display the octree (*Wire* or *Cubes* display, c.f. section 1.4.6) to visually estimate the size of the cells at different levels.
- To call this function, you must have selected a single 3D entity.

2.3.11. Tools > Segmentation > K-Means

This function is no longer part of version 2.1 of CloudCompare.

2.3.12. Tools > Segmentation > Front propagation

This function is no longer part of version 2.1 of CloudCompare.

2.4. Display Menu

2.4.1. Display > Full Screen

This function displays the main window of *CloudCompare* in full screen mode. In this mode, the entire screen is taken up by the application. For the sake of visual comfort, the title bar of *CloudCompare* and the Windows task bar are no longer accessible, though this also hides the window manipulation functions (minimizing, changing the active window, etc...).

To return to normal view, click again on the *Full screen* command, or use the keyboard shortcut (F11).

Note: it is still possible to change the active window, even when in full screen mode, by holding down the ALT key, then tapping TAB until you have highlighted the desired application (cf. figure 2.30). When ALT is released, Windows will activate the selected application window. This command works in all applications.

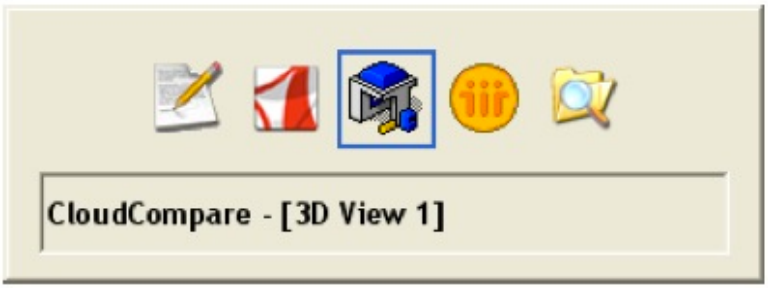


Figure 2.30 – Windows application selector

Keyboard shortcut: F11

2.4.2. Display > Refresh

The Refresh command refreshes the displays in the 3D viewers.

Keyboard shortcut: F5

2.4.3. Display > Test Frame Rate

This function estimates the refresh rate of the displays in *CloudCompare*, measured in frames per second (fps).

The test will take about 10 seconds, and is characterized by a rotation around the objects visible in the active 3D display. Once the test is finished, the result is displayed in the information zone of the 3D viewer.

Note: this rate depends directly on the number of triangles and points to display. For reasons of visual comfort, it is preferable to have a refresh rate of about 25 fps or higher (cf. section 1.3.4).

Keyboard shortcut: F12

2.4.4. Display > Toggle Centered Perspective

In the display of objects, the projection defines the manner in which 3D objects are drawn to the screen in a 2D visualization. *CloudCompare* offers two types of projections:

- *parallel orthographic*: the points are projected orthogonally on the image plane. The field of vision corresponds to a cylinder.
- *perspective*: the points are projected toward a single point that is not part of the image plane. The field of vision corresponds to a cone.

The image plane can be considered to be the visualization screen.

In *CloudCompare*, the *Toggle Centered Perspective* command lets you toggle between the orthographic projection view, which is the default display view, and the projection perspective display. When this command is activated, the center of rotation of the point of view is automatically placed in the center of the observable scene. During the interactive phases (mouse movement in the 3D viewer - cf. section 1.3.3) the camera will turn around the objects in the scene.

If clicked again, the command re-establishes the display mode to orthographic projection.

Keyboard shortcut: F3

2.4.5. Display > Toggle Viewer Based Perspective

This command toggles the 3D views between having a point of view centered on the camera and a point of view centered on the scene.

The first time it is called, it positions the center of rotation on the camera itself. This mode is associated with a projection perspective. During the interactive phases (movement of the mouse in the 3D viewer - cf. section 1.3.3) the camera turns around itself, maintaining its position.

If the command is activated again, it will re-establish the default display centered on the scene and with an orthogonal projection.

Keyboard shortcut: F4

2.4.6. Display > Render to File

Creates a screen capture of the active 3D viewer in a BMP file. This function lets you specify a zoom factor, via the window presented in figure 2.31.

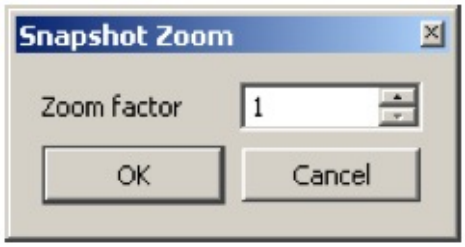


Figure 2.31 – Zoom window for capturing screenshots of the 3D display

2.4.7. Display > Light and Materials > Set Light and Materials

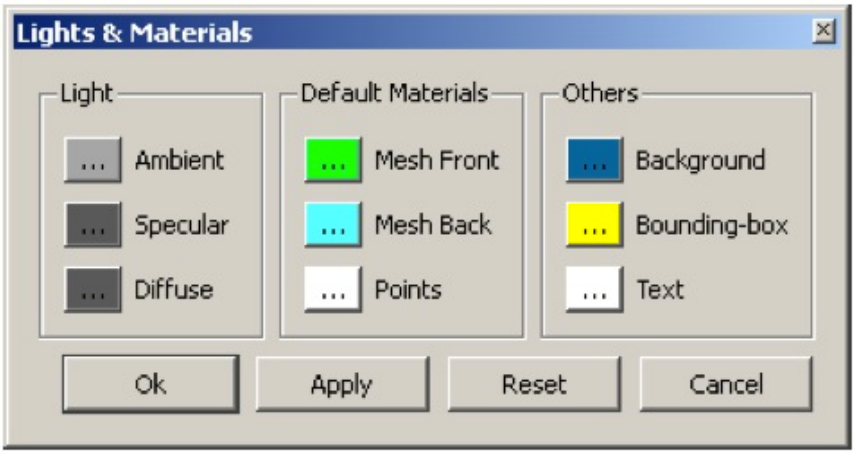


Figure 2.32 – Lighting configuration window

This function sets the parameters of the lighting in *CloudCompare*, via the interface shown in figure 2.32. We see three sections:

The first frame (*Light*) changes the lighting parameters. The user can define a color for each of the three components of light (ambient, specular, and diffuse):

- The *Ambient* component is the constant lighting that bathes the scene: at night, for example, the ambient component is black (no ambient light).
- The *Diffuse* component defines the color reflected by objects independent of the position of the camera.
- Finally, the *Specular* component defines the color re-emitted by objects toward the camera: the brighter this component, the glossier objects appear; the darker, the more they appear matte.

The second frame (*Default materials*) configures the colors applied by default to meshes or point clouds. In the case of meshes, the user can define a different color for each surface (front/back). The *Points* button defines the color of points in point clouds, the same as the command in section 2.2.1.

The final frame (*Others*) modifies some general display options. *Background* sets the color of the background of the 3D viewer. The background always appears in the form of a gradient from the

chosen color to black. *Bounding-box* changes the color of the box encompassing the selected objects. Finally, *Text* changes the color of text in the 3D viewers.

Each of the buttons just described opens the color selection interface described in section 2.2.1.

At the bottom of the window are buttons that let you apply the selected parameters (*OK* and *Apply*), re-set all the values to their defaults (*Reset*) or leave the interface without making modifications (*Cancel*).

2.4.8. Display > Light and Materials > Toggle sun light

Turns the principal light source on and off. You must have at least one light source active to see the lighting effects (shadow, color, ...)

Keyboard shortcut: F6

2.4.9. Display > Light and Materials > Toggle custom light

Turns the personalized light source on and off.

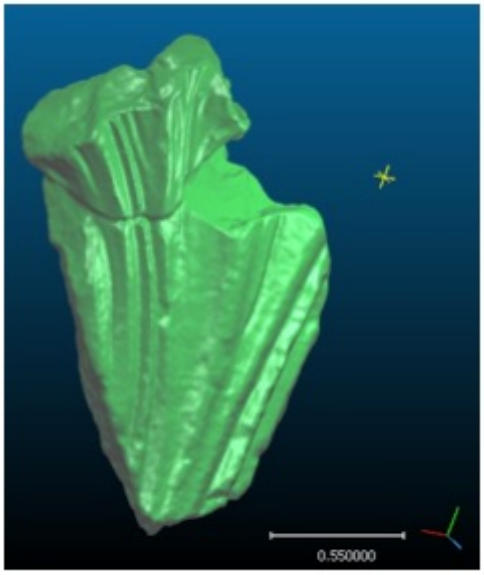


Figure 2.33 – Secondary light source (custom light)

This light source, unlike the principal light source representing "sunlight" - cf. section 2.4.8) is a point light source. It appears as a small yellow star around the objects (see notes below). It has the same properties as the main light source.

It is possible to move the light source by holding down the CTRL key while making a "PAN" with the mouse (holding down the right mouse button).

Notes:
- Keyboard shortcut: F7
- The “star” only appears in the orthographic projection mode (see section 2.4.4. or 2.4.5).
- Sometimes, the star may have an initial position within the object itself.

2.4.10. Display > Console

This command displays or hides the console, visible by default in the lower part of the main window of *CloudCompare* (cf. section 1.1).

2.5. *Plugins* Menu

Plugins are extensions that offer advanced functionalities that are not covered by the main program of *CloudCompare*. They correspond to ".dll" files or dynamic digital libraries (in Windows). The application can perform perfectly well without these functions, as their domain is typically outside of the core purpose of *CloudCompare*.

At the program's launch, *CloudCompare* searches for available plugins in certain pre-defined folders, and only the .dll files (on Windows) are loaded and made available in the application's interface. This manual presents the plugins available with the official version of *CloudCompare*.

It is nonetheless possible that, for one reason or another, you do not have these plugins in your installation. Again, this will not hinder the functioning of *CloudCompare*.

2.5.1. Plugins > PCV

This tool rapidly calculates the illumination of points in a cloud or vertices in a mesh by using the "Portion of Visible Sky" (P.C.V., **p**ortion de **c**iel **v**isible - see figure 2.36).

This lighting consists of calculating for each point the quantity of light it can "see", or in other words the quantity of light energy it would receive if the cloud had been lit uniformly. This colors points as a function of their relative depth, and provides good relief to the micro-geometry. In practice, this calculation is done with an algorithm equivalent to *ShadeVis* (originally proposed by Cignoni et al. of the VCG).

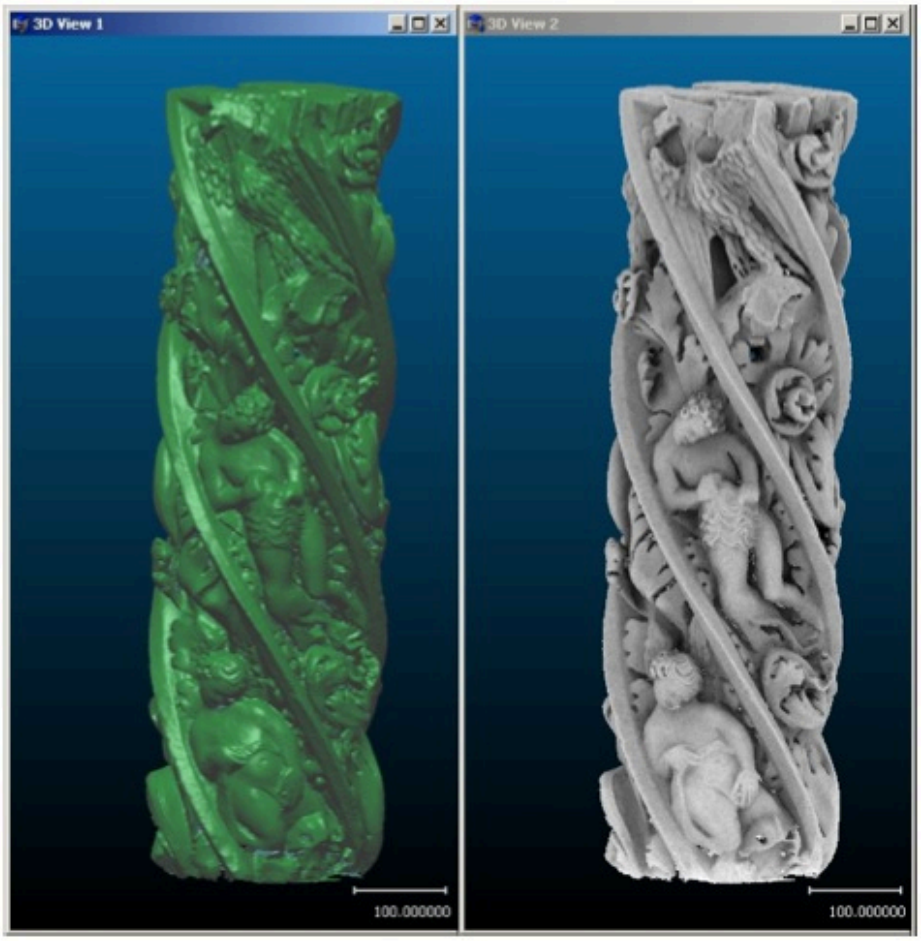


Figure 2.34 – Classic rendering using normals (left) and "PCV" rendering (right)

The two principal parameters of the algorithm, modifiable via the associated dialogue box (figure 2.35) are:

- the number of light "rays". For each direction of lighting (ray), the algorithm projects a graphical map of the entities as seen from that direction and calculates the visibility of the points (or vertices of a mesh). This information is accumulated for each direction to calculate the global lighting. The more rays that are used, the more differences between points will be apparent, and the more dynamic it will appear. On the other hand, the calculation time is proportional to the number of rays.
- the resolution of the rendering buffer. The projection of entities in a viewing direction is done in a video buffer, of which the resolution contributes to the ability to distinguish between points. The higher the resolution, the more distinctly the points will appear, with better calculations of the individual lighting effects and a more the points will appear separate (enabling better lighting calculations and a higher quality of result). On the other hand, if the resolution is too high, calculation time and memory consumption will be unweildly (depending on the performance of the graphics card). Also, be aware that the cloud may become "porous" and the light may shine through (see remark below). In the case of a mesh, this does not pose any problem. Current graphics cards provide good performance in absolute terms, so do not hesitate to use large values for these parameters (such as the default values).

Remarks:

- The algorithm creates a new scalar field of the type "PCV" and the color scale "Grey" is automatically activated.
- The light simulated by the PCV algorithm is considered as coming from the "Z positive" hemisphere. Z corresponds to the vertical direction, so the point cloud must be oriented accordingly before the calculation. If the "360° mode" box is checked, the light will be applied to all directions equally and direction will not play a part.
- As the illumination calculated by this algorithm is a scalar field, it is possible to adjust the sliders to control the saturation and contrast. In the case of a mesh, we can also use the smoothing and enhancing functions (see sections 2.2.12 and 2.2.13). Once the correct parameters are set, we can transform the scalar field into color using the *Scalar fields* > *Convert to RGB* function (section 2.2.23).
- The lighting coming from the sky is represented in a discrete manner by a limited number of light "rays", which are sampled uniformly from the hemisphere (or the full sphere if 360° mode is active). There is not much use of ray tracing in *ShadeVis* (rather, we should talk of viewing angles - cf. the article by Cignoni et al. for more information).
- In the case of meshes, it is possible to accelerate the algorithm if the mesh is closed (see the option "closed mesh", which is active by default).
- In the case of a point cloud, one must pay attention that the resolution is not too large, otherwise "holes" could appear between the points during the internal rendering: this is simply due to the fact that the density of a cloud is limited, and that for a sufficient level of zoom, we always observe empty zones between the points.

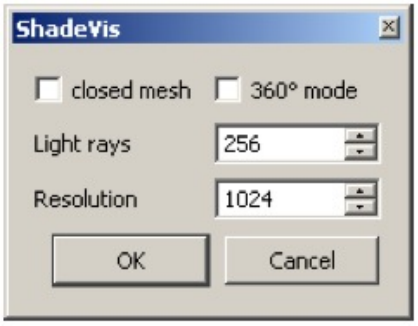


Figure 2.35 – Configuration interface for the PCV rendering

2.5.2. Plugins > HPR

The *Hidden Points Removal* function, as its name indicates, filters the selected point cloud to only keep the visible points (corresponding to the visible surface from the current point of view). The points considered to be thus "masked" are therefore hidden. The result strongly depends on the point of view.

The notion of visibility for points in a cloud is relatively complex to estimate. In reality, it is very unlikely that a point will be actually hidden by other points in a cloud, because that would require perfect alignment between pairs of points, or a density so high that the points are practically in contact. This function approximates the notion of visibility using a calculation of the convex envelope. It is based on the article, *Direct Visibility of Point Sets* by Katz, Tal and Basri, SIGGRAPH 2007.

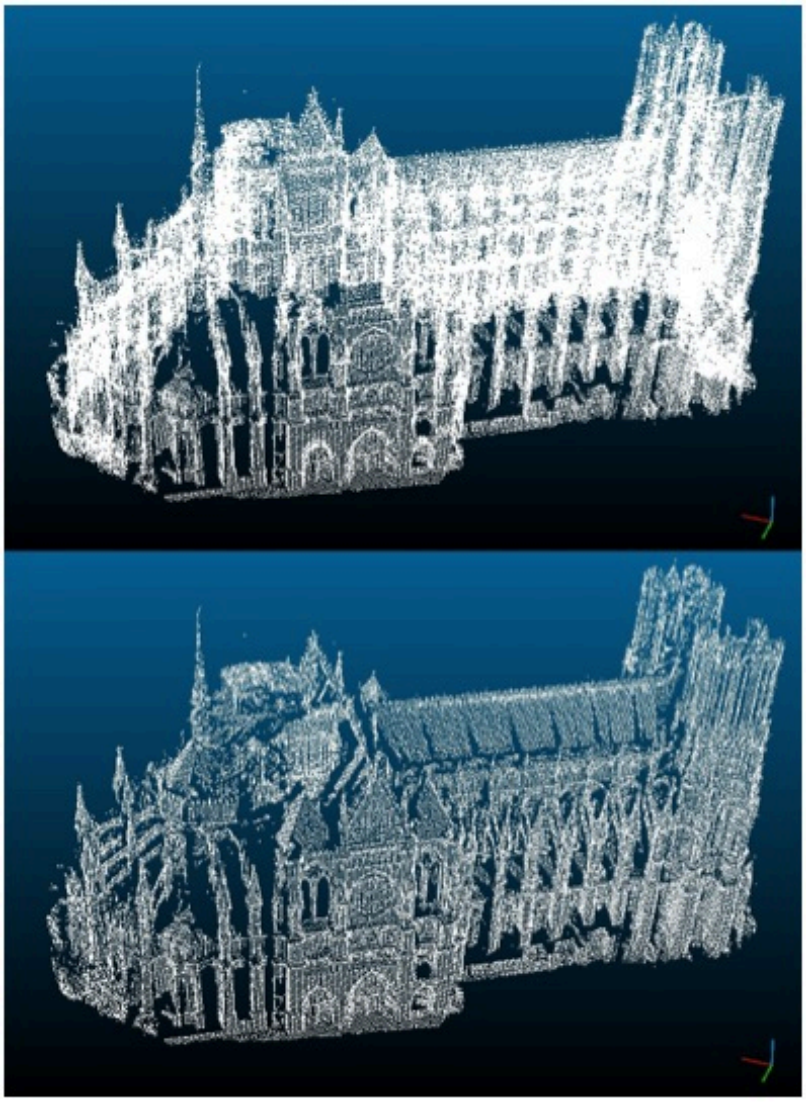


Figure 2.36 – Complete point cloud (top) and point cloud filtered by "HPR" (bottom)

To calculate the occlusions using this function, the 3d viewer must be using the projection perspective (cf. section 2.4.4). Otherwise, an error message will warn the user and ask them to activate projection perspective. The user must then choose the octree level to be used by the function (figure 2.37). The octree level speeds up the calculation of the convex envelope (a rather large structure) by reducing the number of points used via subsampling. The higher the level, the finer the calculation, but it will take longer.

Once the filtering is done, it is only valid for the current camera position (give or take a small amount). The tool must be restarted to update the filtering for each new point of view.

Warning: the points hidden by this method cannot be re-displayed ad-hoc (for the moment). In the meantime, one must use a workaround: activate the tool to manually segment the cloud (the "scissors" icon - see section 1.5.1) which re-calculates the visibility information for each point), and then quit this mode.

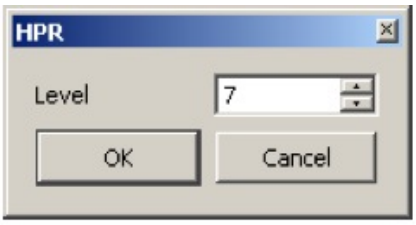


Figure 2.37 – Interface to choose the octree level

2.6. 3D Views Menu

2.6.1. 3D Views > New

Opens a new 3D view (window).

Remarks:
- Keyboard shortcut: CTRL + F3
- The name of the new 3D view is named as a function of the number of views opened since starting *CloudCompare*. If *n* 3D views have been opened in the current session, then regardless of the number of currently open views, the new element will automatically be named "3D View *n*+1".
- 3D viewer windows created this way will be "virgin": they will not contain any objects, and it is up to the user to display available entities as desired (cf. section 1.3.2)

2.6.2. 3D views > Close

Closes the active 3D view (window).

Remarks:
- Keyboard shortcut: CTRL + F4
- The objects in the 3D view are not reassigned to any other window, so are no longer visible. The user can, if desired, re-display the objects in any of the remaining windows (cf. section 1.3.2).

2.6.3. 3D Views > Close all

Close all the open 3D views.

2.6.4. 3D Views > Tile

This command partitions the display space between the different open 3D views. The windows are arranged so that the entire display space is filled, and there is no overlap between them (they form a mosaic).

Note: this arrangement is useful to see all 3D views at once.

2.6.5. 3D Views > Cascade

Organizes the 3D views in a cascade: they are superimposed according to the order in which they were created.

Note: the cascade display is useful to navigate quickly between the current views.

2.6.6. 3D Views > Next

This command activates the next 3D view.

Note: the order of the 3D views is set by their names, and therefore by the order in which they were created. The "next" view is therefore the open window that was created first after the current window.

2.6.7. 3D Views > Previous

This command activates the previous 3d view in place of the current.

Note: the "previous" window is the last one created before the current window.

2.7. Help Menu

2.7.1. Help > Help

Display the user documentation for *CloudCompare*.

Keyboard shortcut: F1.

2.7.2. Help > About

Display the information window for the current version of *CloudCompare* (cf. figure 2.38).

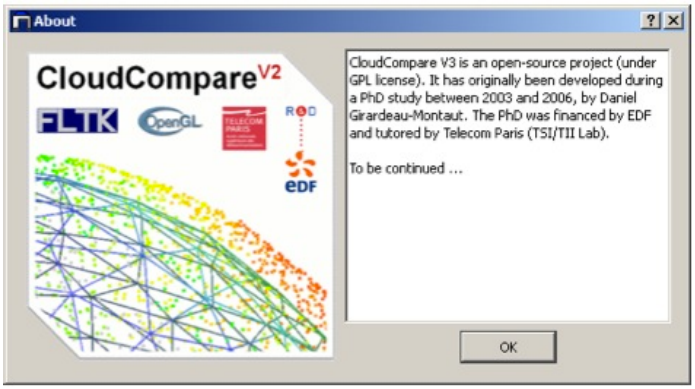


Figure 2.38 – Information window

2.7.3. Help > About plugins

Display the plugin information window (figure 2.39). This window displays the available plugins in the form of a tree. The folders in which *CloudCompare* searches for plugins are displayed at the top of the window.

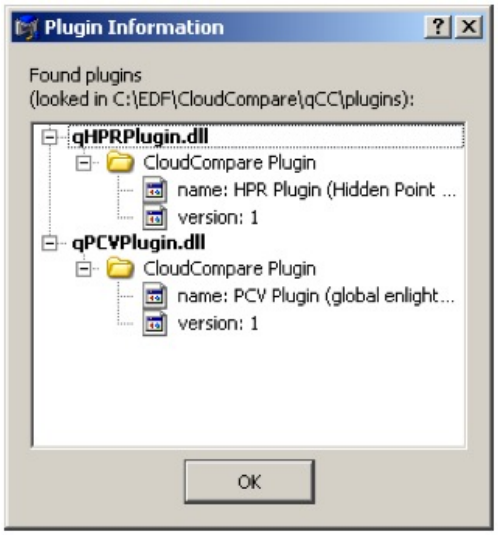


Figure 2.39 – Plugin information window

A Appendix

A.1 File Formats

A.1.1 Recognized 2D/3D file types

Extension	Type	P	M	RGB	LG	N	S	Other	Description
asc, txt, neu, xyz, etc.	ascii	✓		✓	✓	✓	✓		ASCII point cloud
bin	binary	✓		✓		✓	✓		point cloud(s), binary format specific to CloudCompare
ply	ascii/binary	✓	✓	✓	✓	✓	✓	✓	mesh (Stanford)
obj	ascii		✓			✓			mesh (Wavefront)
soi	ascii	✓			✓				cloud(s) (Soisic, Mensi)
(c)bin	binary	✓		✓					cloud (C, Hernandez)
pn	binary	✓				✓			cloud (point + normal)
pv, pcv	binary	✓					✓		cloud (point + value)
icm	ascii							✓	cloud/image association

P : Points

M : Meshes

RGB: Colors (Red, Green, Blue)

LG : Levels of Grey

N : Normals

S : Scalars

A.1.2 Opening and Saving

	Open	Save
asc, txt, neu, xyz, etc.	✓	✓
bin	✓	✓
ply	✓	✓
obj	✓	✓
soi	✓	
(c)bin	✓	
pn	✓	✓
pv, pcv	✓	
icm	✓	

A.1.3 Special Formats

A.1.3.1 ICM files

This type of file associates a point cloud with a VRML file – the latter defines calibrated photos (camera + image file).

Example:

file: toto.icm

```
#CC_ICM_FILE                          // Header
FILENAME=pa4.asc                      // Points file
FILETYPE=ASC                          // Type of points file
IMAGES_DESCRIPTOR=photo_match.wrl     // VRML file describing the calibrated photos
```

file: photo_match.wrl

```
#VRML v2.0 utf8

DEF photo1.jpg Viewpoint{                          //Header photo #1
 fieldOfView 0.621379                              // FOV
position -10.5418 -15.6091 5.95961                 //Optical Center
description «VANNE + PETIT TUYEAU »                //Description
orientation 0.70724 -0.37292 -0.600618 3.74252     //View vector
}

DEF photo2.jpg Viewpoint{                          //Header photo #2
fieldOfView 0.621379                               //etc.
position -3.9782 -21.276 5.95616
description «PORTE »
orientation 0.572629 0.696275 -0.432778 2.02135
}
```

A VRML file like this can be generated quasi-automatically by a program such as *RealWorks* (Mensi).

A.1.3.2 Depth Map Export file

An export, in ASCII, of the depth map associated with a sensor. It may be generated by the function *Sensor* > *Ground-based Lidar* > *Export depth buffer* (section 2.2.15).

Example:

file: ground based laser scanner.txt

```
// CLOUDCOMPARE DEPTH MAP
// Associated cloud : Cloud #0                     associated cloud name (as disp. in CC)
// dPhi = 0.005000 [ -0.383052 : 0.319331 ]        horizontal angular step
// dTheta = 0.005000 [ -1.626588 : 0.137948 ]      vertical angular step
// pMax = 78823.398438                             max depth
// L = 353                                         number of horiz. pixels
// H = 141                                         number of vert. pixels
//////////////////////////
0.000000 0.000000 18132.496094                     1st pixel coordinates (i,j) and depth (z)
1.000000 0.000000 15145.963154                     2nd pixel coordinates (i,j) and depth (z)
...
352.000000 140.000000 132135.321542                L*H pixel coordinates (i,j) and depth (z)
```