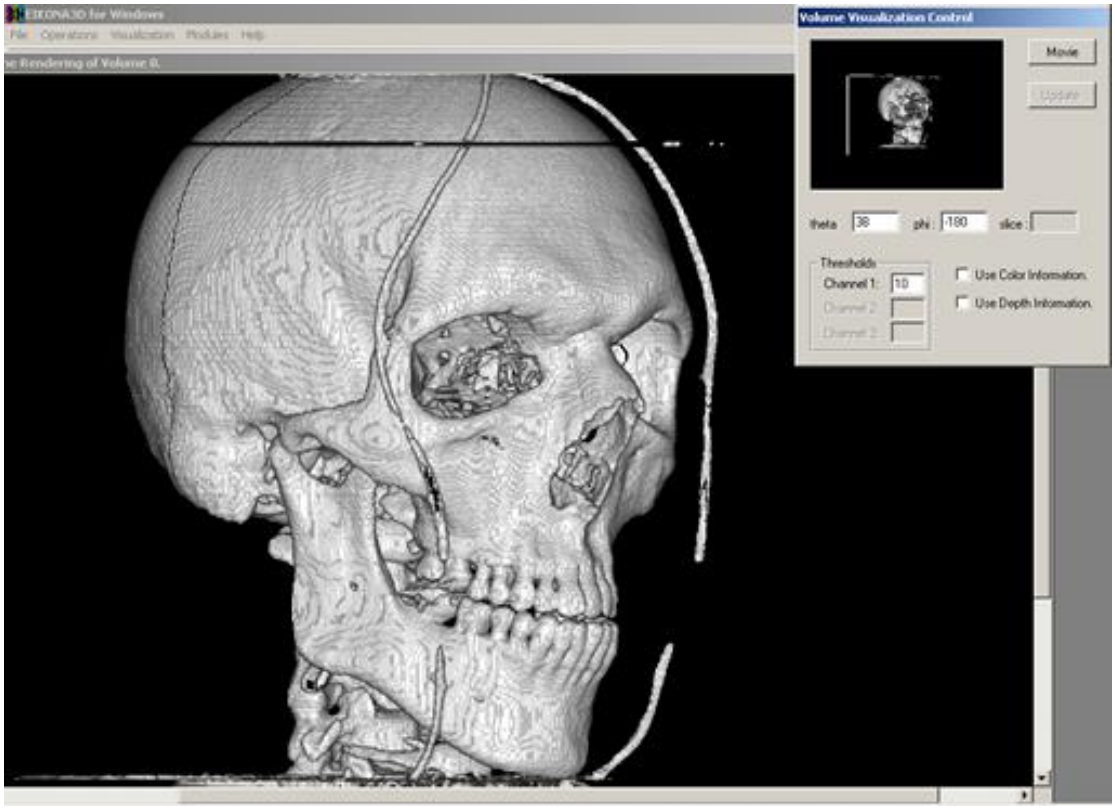
EIKONA3D

A software package for 3D Image Processing, Analysis and Visualization.

Version 3.2.3.1

Part II: EIKONA3D modules



Prof. I. Pitas
Aristotle University of Thessaloniki
www.aiia.csd.auth.gr
pitas@csd.auth.gr

Distribution:

Prof. I. Pitas
Aristotle University of Thessaloniki
www.aiia.csd.auth.gr
pitas@csd.auth.gr

EIKONA3D Manual

PART II: EIKONA3D modules

CONTENTS

PART II:EIKONA3D modules

- **3D image I/O**
  - DICOM module
- **3D Visualization**
  - Volume visualization module
  - Surface rendering module
- **3D Image processing/analysis**
  - Automatic alignment module
  - Marching cubes module
  - Surface modeling module
  - 3D skeletonization module
  - ICP algorithm module
  - Tooth drilling for dental purposes module

DICOM

By choosing *DICOM Module* from the *Module* menu of the menu bar, the user is allowed to open or load a DICOM image. The images being processed by the *DICOM Module* are:

- 8-bit grayscale
- 16-bit grayscale (signed/unsigned, Intel/Sun byte order)
- 8-bit color
- RGB
- Planar RGB

In the following lines the way a user can handle this module is briefly described.

Open DICOM File: When the user selects *Open DICOM File*, the only thing that he/she has to do is to select the file from the Open File Window (Figure 1). After that, the image is opened and displayed.

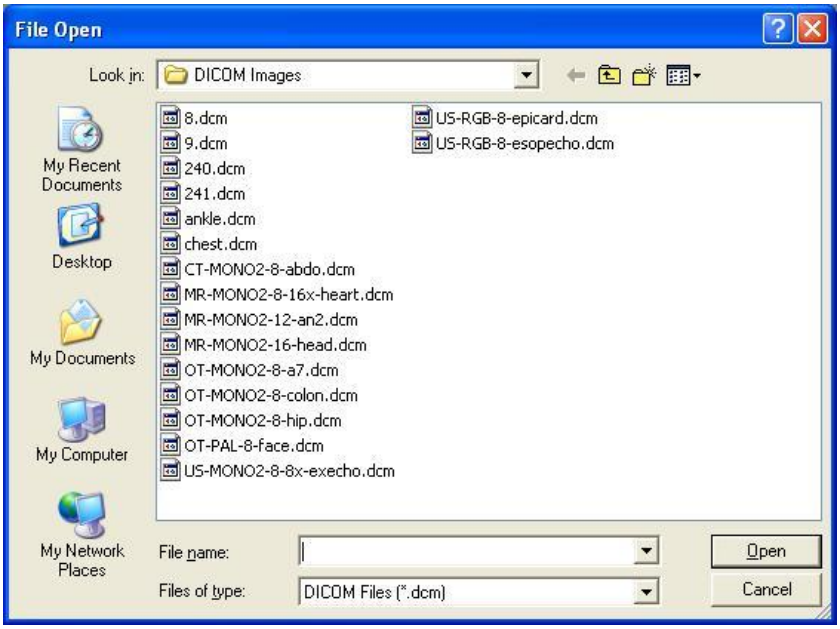


Figure 1: File Open window.

NOTE: If the selected image file is an 8-bit color one, the user will be asked if he/she wants to save the look-up table of the image. If the answer is positive, the user will be able to choose the filename for saving the look-up table, and after that, the image will be displayed. It is worth to note that the look-up table files must be saved in a specific

folder, which is named `luts` and should exist in the EIKONA3D directory. If there is no such directory create one before performing a lut save operation

Save DICOM File:At first the user must select the volume to be saved from the *Select Input Volume* dialog box of EIKONA3D (Figure 2).

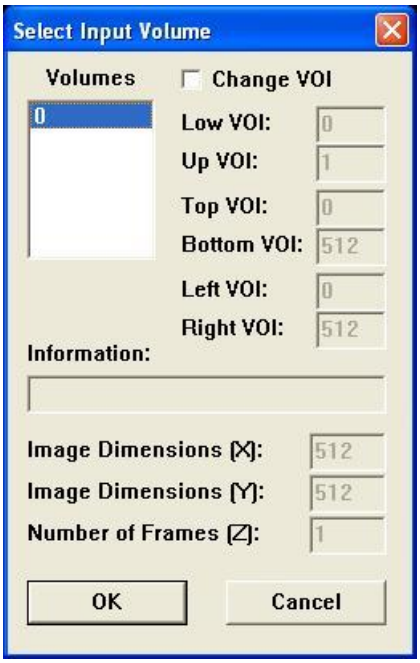


Figure 2: EIKONA3D *Select Input Volume* window.

If the selected volume belongs to an already opened DICOM file, the following dialog box (Figure 3) will be displayed:

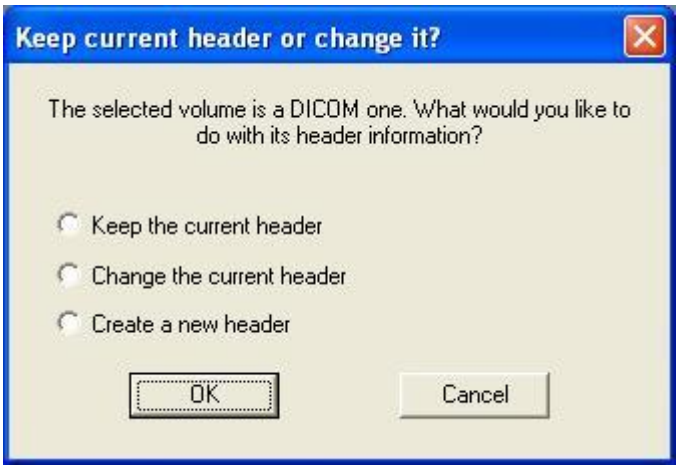


Figure 3: *Keep current header or change it* window.

In this dialog box the user must select if he/she wants to change the current header of the DICOM volume or not (or even create a new header). If the user decides not to

change the header of the DICOM file to be saved, the only thing that he/she has to do after the volume selection is to select the appropriate name for saving the image.

In case the user wants to change the header (or create a new header), the *Change Header* dialog box is displayed, in which the user can change some of the most important parameters of the DICOM header (Figure 4).

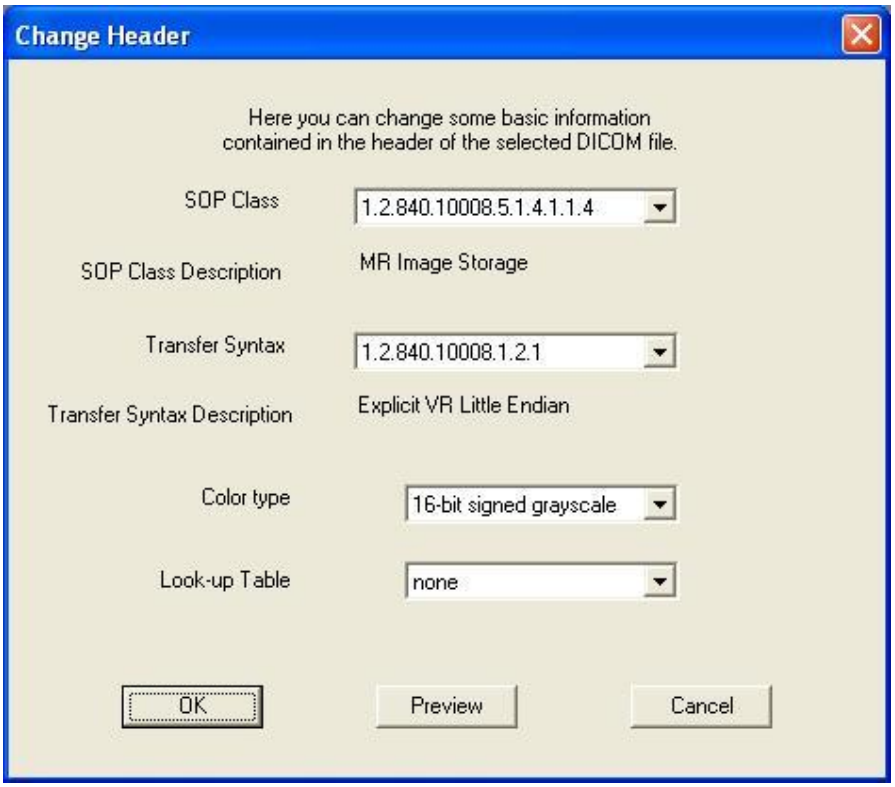


Figure 4: *Change Header* dialog box.

A DICOM file header contains 1769 parameters, so it is virtually impossible to try to change every single parameter. Thus, only the most important parameters are allowed to be changed.

These parameters are:
SOP Class: A tag used to define the kind of medical imaged stored in the DICOM file. Not necessary to be defined.

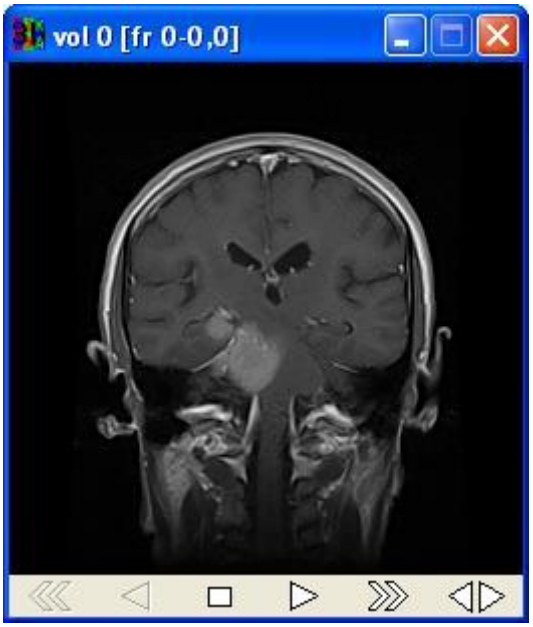
Transfer Syntax: It defines the way bytes will be stored in the DICOM file. It is necessary for the 16-bit grayscale image. The difference between *Little Endian*

transfer syntaxes and the *Big Endian* one is that the bytes are stored in different order (small-great in *Little Endian*, great-small in *Big Endian*).

Color type: It defines the color type of the image to be saved. The user is obliged to define this parameter.

Look-up Table: This option is useful for turning an image (grayscale or color) into a color one and saving it. It is worth to note that, with this option, the user can turn any volume -not necessarily belonging to a DICOM image- into a color one. The names of the look-up tables which are stored in the `luts` folder are shown. One of them must be selected. If the initial image is grayscale, it will be simply turned into a color one. On the other hand, if the initial image is a color one, it will be first turned to a grayscale one and then turned again to a color one by applying the selected look-up table. By selecting the *Black-White* look-up table, the color volume will have the same R, G and B channels with the grayscale channel of the *source* volume. The same thing stands for the selection of *White-Black (Invert)* look-up table, but in this case the color channels of the *target* volume will be the invert of the *source* volume. On the other hand, by selecting *none*, no look-up table correspondence will be made.

It is worth to note, that with the *Preview* button, the user is able to see whether the selected look-up table is the appropriate one or not (Figure 5a,b).



(a)

(b)

Figure 5: Preview window for a selected volume. (a) Original volume, (b) Preview volume.

In case the *Preview* button is pressed, an OpenGL window is displayed, in which the result of the effect of the look-up table is shown to the user. When the preview window is active, the following buttons are active:

Right arrow	Move by *Step* frames to the front
Left arrow	Move by *Step* frames to the back
Home	Move to the first frame
End	Move to the last frame
F1	Decrease the *Step* by one (*Step* $\in$ [1,10])
F2	Increase the *Step* by one (*Step* $\in$ [1,10])

After that, the *File Save* window is displayed (Figure 6).

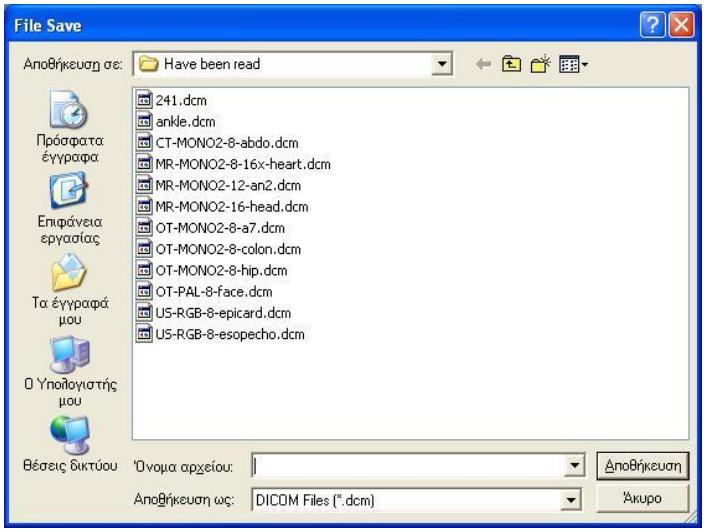


Figure 6: File Save window.

It is worth to note that the user can save a non-DICOM volume (grayscale or color) as a DICOM image. The procedure is the same as if the user had to save a DICOM volume, but in this case the *Keep current header or change it* window is different (Figure 7).

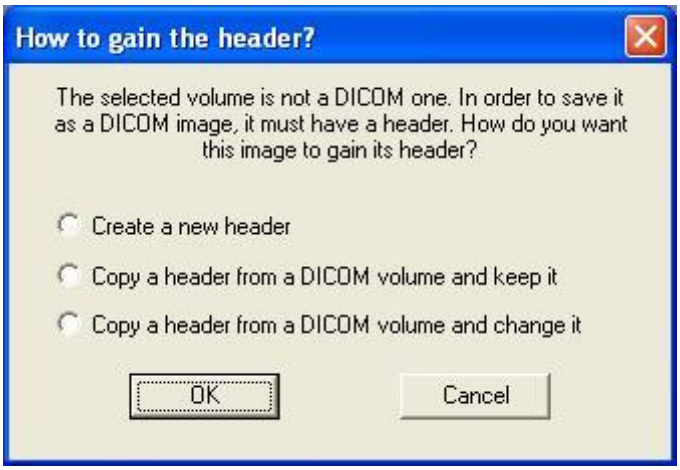


Figure 7: *How to gain the header?* window.

The available options for the user are the following:
Create a new header: The user must define the fields of the new header in the *Change Header* window.

Copy a header from a DICOM volume and keep it: The user must select the volume which will provide the header for the new DICOM file, and then select the appropriate file name for saving the image. It can be used in case a non DICOM image has been produced by processing a DICOM image (e.g. filter a DICOM image).

Copy a header from a DICOM volume and change it: Again, the user must select the volume which will provide the header for the new DICOM file, but after that he/she will be able to change the basic characteristics of the header in the *Change Header* window.

In a nutshell, the DICOM images able to be opened by the "DICOM Module" can be saved in these formats:

Open Format	Save Format
	8-bit grayscale

8-bit grayscale, 16-bit grayscale (various formats)	16-bit signed grayscale (Little Endian)
	16-bit unsigned grayscale (Little Endian)
	16-bit signed grayscale (Big Endian)
	16-bit unsigned grayscale (Big Endian)
	8-bit color (*)
	RGB (*)
	Planar RGB (*)
8-bit color, RGB, Planar RGB	8-bit grayscale
	16-bit signed grayscale (Little Endian)
	16-bit unsigned grayscale (Little Endian)
	16-bit signed grayscale (Big Endian)
	16-bit unsigned grayscale (Big Endian)
	8-bit color (**)
	RGB (**)
	Planar RGB (**)

(*): with look-up table selection

(**): look-up table selection is not compulsory

Volume visualization

The Volume Visualization module is the file `volrend.dll`. If this DLL exists in the directory of EIKONA3D when the program starts, a sub-menu called *Volume Visualization* is added under the *Modules* menu. The menu options of this sub-menu provide several ways for 3D visualization derived directly from the volumetric data of a volume, including parallel projection (normal, average, maximum), sectioning and volume rendering.

For all the available options, the same procedure is followed; the user selects the input volume to visualize (optionally with modified VOI) through the *Select Volume to Visualize* dialog box and then he can control the visualization through the *Volume Visualization Control* dialog, which is common for all different visualizations that may be active simultaneously. For each active visualization, there is a separate display window.

The operations performed in the *Volume Visualization Control* dialog affect the display window that has the focus. This dialog contains a small display window that displays a preview of the visualization together with the axons *x*, *y*, *z* of the orthogonal co-ordinate system. The preview window works with a sub-sampled version of the input volume, thus achieving better interactivity. The user can interactively manipulate the preview window with the mouse to set the desired viewpoint, which is defined by two angles: (1) a rotation by *theta* degrees around *z* axis and (2) a rotation by *phi* degrees around *x* axis. The manipulation of the preview window is accomplished by clicking and dragging inside it with the mouse - left and right to change the *theta* angle, up and down to change the *phi* angle. Alternatively, the user can directly set these angles in the respective edit boxes of the dialog. Also, the user can change the threshold values that define the maximum value of the background points. For a gray-level volume the threshold defines that the useful information inside the volume is composed of the points with gray-level higher than it. For a color volume the thresholds define that the useful information inside the volume is composed of the points with at least one of the RGB values higher than the respective channel threshold.

The *Update* button of the *Volume Visualization Control* dialog can be used for updating the normal display window, which displays the visualization in real size, using the current settings. The displayed image in the normal display window can be saved in disk by selecting the *Save Image* menu option from its system menu.

The *Movie* button of the *Volume Visualization Control* dialog can be used for producing a series of frames (movie), each one representing a view of a volume calculated with successive increments of the *theta* angle (in order to complete a 360 degrees cycle) and the current *phi* angle. The more the frames that are produced, the smaller the difference between successive angles. The user is first prompted to specify the required number of frames through the *Number of Frames* dialog box and the volume where the movie is to be written in through the *Select Output Volume* dialog box. In the case of the *Section* menu option (see below), the movie is composed of frames with increasing depth percent.

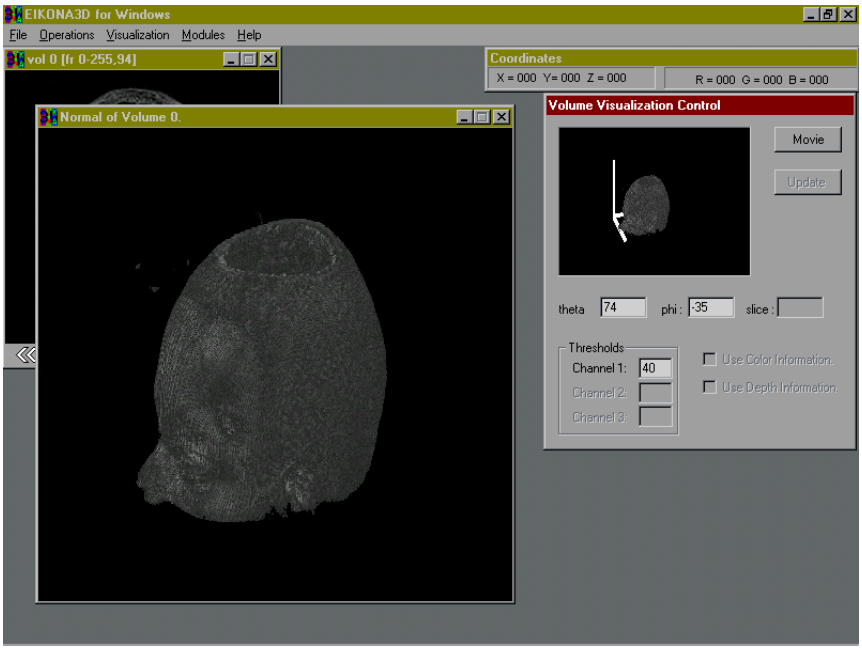


Figure 1: An example of normal projection of volume containing an image sequence of cross-sections of a human head.

The special characteristics of each of the available visualisation methods are described below [1].

Normal Projection: This menu option performs the parallel projection method. The projection is composed of the gray-level/RGB value of the first point each parallel ray hits. An example of normal projection of a volume is seen in Figure 1.

Average Projection: This menu option performs the parallel average projection method. The projection is computed by averaging the gray-1evels/RGB values of all the points that each parallel ray goes through. Here, the threshold values have no effect.

Maximum Projection: This menu option performs the parallel maximum projection method. The projection is composed of the maximum gray-level/RGB values of all the points each parallel ray goes through. Here, the threshold values have no effect.

Section: This menu option performs sectioning of the volume, which enables the exploration of the inner invisible structure. The result image is computed by cutting away a section of the volume with a plane defined by the selected *theta* and *phi* angles with respect to the volume (the cutting plane is always parallel to the screen) and the depth percent, which is chosen through the provided scroll-bar next to the preview window or the *slice* edit box. Here, the threshold values have no effect.

Volume Rendering: This menu option performs a three-dimensional volume rendering technique. The input volume can be binary, grayscale or color. Volume rendering differs from projection in the fact that it illustrates the 3D structure of the objects; it is not based on the intensity values but on the angle between the normal of the object surface and the viewing direction. An example can be seen in Figure 2.

There are two options that affect the rendering which are described below.

Use depth information: This is a toggle option in the *Volume Visualization Control* dialog. When it is enabled, the rendering considers the information of the depth of the rendered object surface. By this way, the surface points that are of more distance from the viewing plane appear darker. If this option is disabled (default), then the luminance of the rendered surface points is varied only with the angle of the surface normal with the viewing direction. An example is shown in Figure 3.

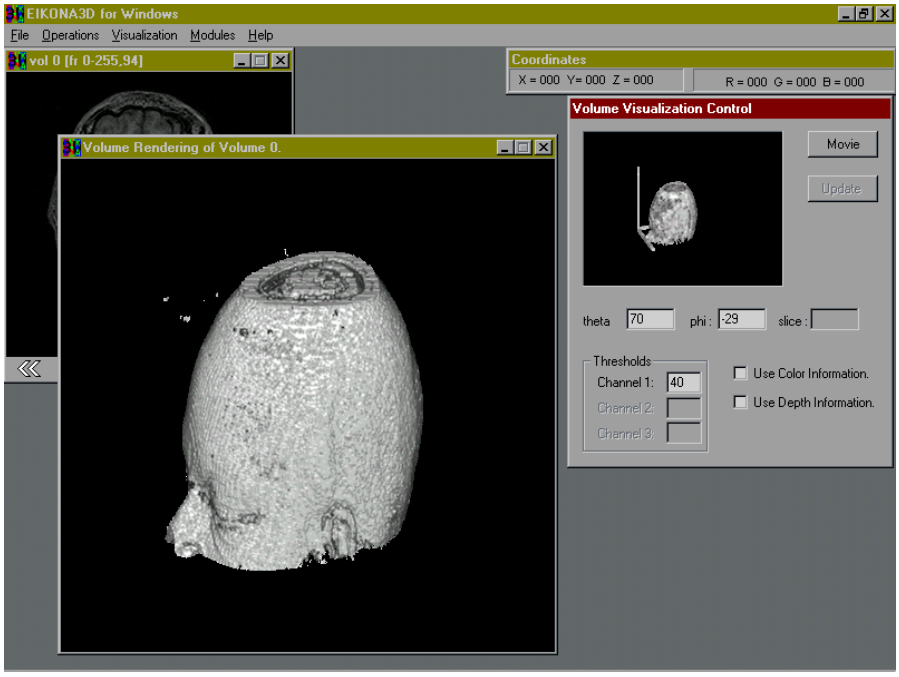


Figure 2: An example of interactive volume rendering of a volume containing an image sequence of cross-sections of a head.

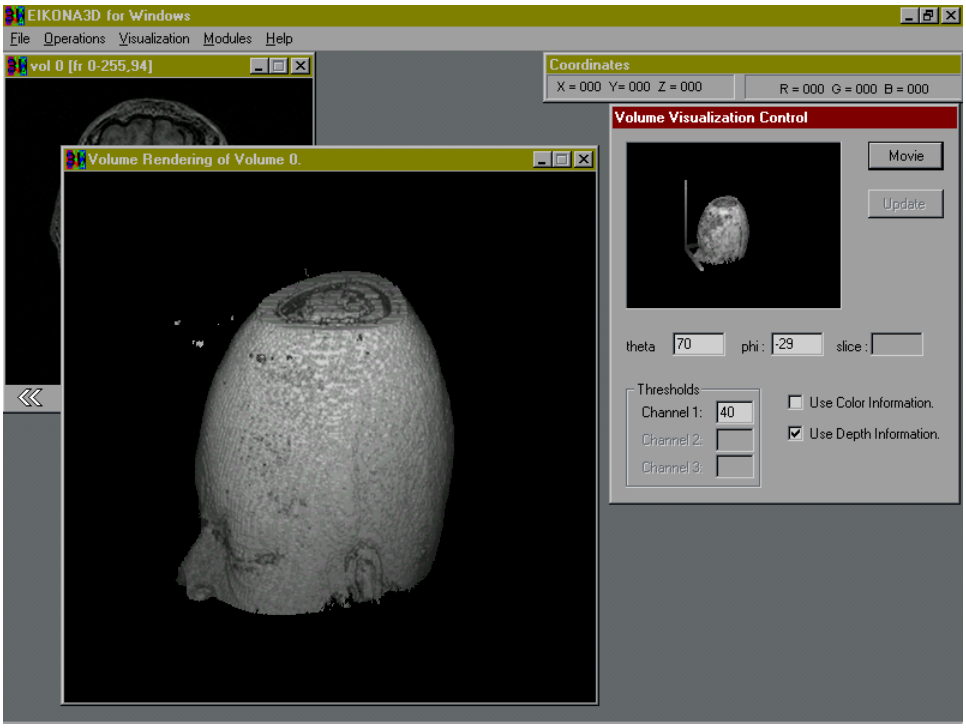


Figure 3: An example of interactive volume rendering of a volume containing an image sequence of cross-sections of a head. Here, depth information is used.

Use color information: This is a toggle option in the *Volume Visualization Control* dialog. When it is enabled, the rendering considers the information of the original

color of the rendered object surface. By this way, the rendering seems more like a projection. If this option is disabled (default), then the luminance of the rendered surface points is varied by scaling the whole range of color levels (256 different levels of the first channel – gray levels for binary or grayscale volumes and red component levels for color volumes). An example is shown in Figure 4.

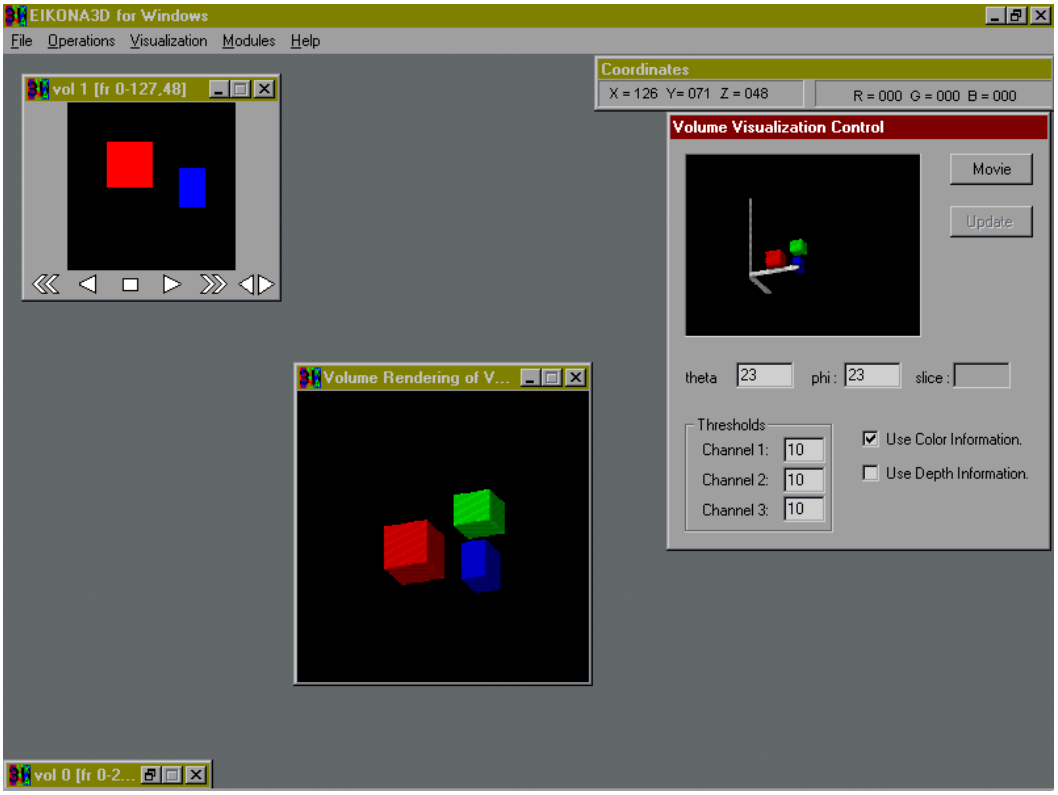


Figure 4: An example of interactive volume rendering of a volume containing three 3D objects (red green, blue). Here, color information is used.

References

[1] N.Nikolaidis, I.Pitas '3D image processing algorithms', Wiley 2000.

Surface Rendering

The Surface Rendering module is the file `surfrend.dll`. If this DLL exists in the directory of EIKONA3D when the program starts, a sub-menu called *Surface Rendering* is added under the *Modules* menu. The menu options of this sub-menu provide the necessary tools for three-dimensional surface reconstruction using triangulation and surface rendering. The source data for surface reconstruction is required in the form of a volume (image sequence) that has been manually processed using the *Color Contour Follow* option (and probably the *Manual Frame Alignment* option) under the *Operations* menu of EIKONA3D (or any other externally created image sequence with similar content). The manual tracking of the closed contours of the object surface(s) in each slice (frame) of the volume, gives a different color to the contours of each object of interest (see the description of *Build Surface Script* menu option below for more details on color selection restrictions), whereas the background remains black. If alignment between successive frames is needed, it can be performed on the source image sequence or on the processed image sequence that contains the colored contours. Usually, the source image sequence contains more information that can be of help in the alignment and it is preferred for performing the alignment procedure. In this case, the contour following is performed directly on the aligned image sequence. The following menu options then use the contours to construct the object surface(s) via triangulation (or even just a 3D representation of the contours only), write it to an industry-standard AutoCAD script file, and produce surface renderings, which can be optionally saved to disk.

***Build Surface Script*:** Through this menu item, the user is first prompted to specify the input color volume (which is supposed to contain the frames of the image sequence with the colored contours that will be connected to form the object surface) through the *Select Input Volume* dialog box. Then, he is prompted to specify the name of the output script file through the *Select Output Script File* dialog box and, finally, the interlayer distance (distance between two successive layers in proportion to the 2-D resolution of the frames) through the *Interlayer Distance* dialog box. Then, the volume is processed to create an AutoCAD script file (the default extension is `.scr`). During this processing, the user is prompted through the *Specify Object Connections* dialog to specify possible connections between different objects (differently colored)

that cannot be guessed automatically, thus providing a way to reconstruct complex objects (see next paragraph for more details). The produced output script file can subsequently be used by AutoCAD or other third-party applications that support this format, or be directly displayed using the *Interactive Rendering* menu option.

Care should be taken in the formation of the input volume containing the color contours in order to avoid unexpected results. A few rules should be followed. First of all, the background pixels (those that do not belong to any contour) in each frame should be black (R=G=B=0). If a contour (in any frame) does not contain any other internal contours of different objects, it makes no difference if it is filled with the same color as the contour or with the background color, since the algorithm tracks and uses only the external boundary of each connected colored shape. For the same reason, the width of a contour makes no difference to the final result. Also for the same reason, care should be taken so that a contour not filled with the contour color does not have any discontinuities (holes), in which case the tracking will continue to the internal boundary of the contour and represent it as a petal-like shape. Another important issue is the coloring of the contours. The group of contours (in two or more successive frames) belonging to the same object should have exactly the same color, so that the algorithm can automatically connect them to form the surface of the object. Furthermore, the color of an object should be different than the color of any other object, in order to avoid unexpected connections between different objects. We define a 3D object as single, if it has only one contour in each volume frame. In order to reconstruct a complex object, it should be separated in two or more single objects. A complex object contains one or more branches. In the case of a branch, we have a body that splits in two or more branches (the body can be formed by one or more successive contours). In such a case, we should form a single object (with respect to coloring) that contains the body and one of the branches and each of the rest branches should form a new single object (this holds recursively for the case of a branch that splits in other branches). As the algorithm cannot automatically connect the branches on the body, the user should specify a connection (in the *Specify Object Connections* dialog) for each branch that forms a new object. Specifically, if the branch occurs in the direction of increasing *z* (frame index), it should be specified that the lower chain (that one in the frame with the lower index) of each branch that forms a new single object (selected as object A in the *Specify Object Connections* dialog) is to be connected on the object that contains the body (selected as object B in the *Specify*

Object Connections dialog). If the branch occurs in the direction of decreasing *z* (frame index), it should be specified that the upper chain (that one in the frame with the higher index) of each branch that forms a new single object (selected as object A in the *Specify Object Connections* dialog) is to be connected on the object that contains the body (selected as object B in the *Specify Object Connections* dialog). The selection of an object in the *Specify Object Connections* dialog is leaded by the object colors, as they are found in the input volume. In the output surface script, each branch that forms a new single object remains as a separate object, including also the part of the surface that connects it on the object that contains the body. If the surface script is rendered, in order to view a complex object as one object, it suffices to give all its component objects the same characteristics (color or material, depending on the application).

***Build Lines Script*:** This menu item is similar to the previous one, with the difference that it produces a 3D representation of the contours without connecting them to produce a surface. Thus, it can help for previewing the orientation in the 3D space of the contours that exist in the frames of a volume suitable to be processed with the *Build Surface Script* procedure.

***Read Lines Script*:** This menu enables the conversion of a lines script file back to a volume. First, it asks for the name of the input script file through the *Select Script File to Open* dialog box, then asks for the output volume through the *Select Output Volume* dialog box, and finally asks for the interlayer distance (the same interlayer distance used when creating the lines script must be given for normal results).

***Interactive Rendering*:** This menu item enables the visualization of a previously created script file (containing a surface/lines representation) through surface rendering. The user needs only to specify the directory and the name of the script file from the displayed *Open* dialog box. In the rendering window the user can rotate the 3D object by dragging the mouse in any orientation. Clicking once with the right mouse button in the window resets the viewing angles. The *Display Coordinates* option is also available for the content of this window. The following menu options (*Set Object Colors*, *Take Snapshot*, *Make Movie*), which are enabled during a surface rendering operation, provide further useful operations related to the content of the

surface rendering window. Examples of interactive rendering can be seen in Figure 1 and Figure 2.

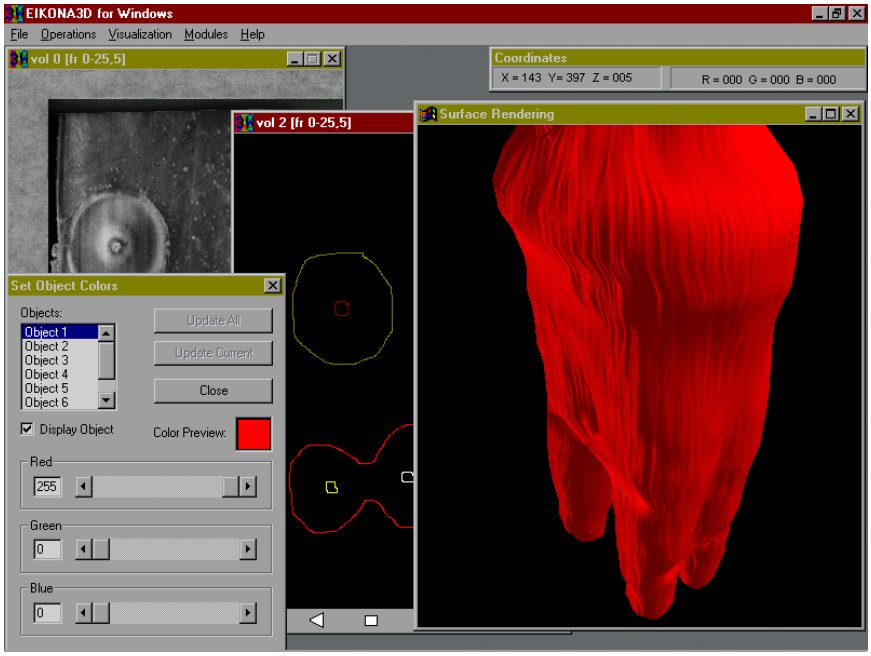


Figure 1: An example of interactive surface rendering of a 3D surface representation of a tooth, produced from an aligned sequence of frames containing contours. A view of outer surface is shown.

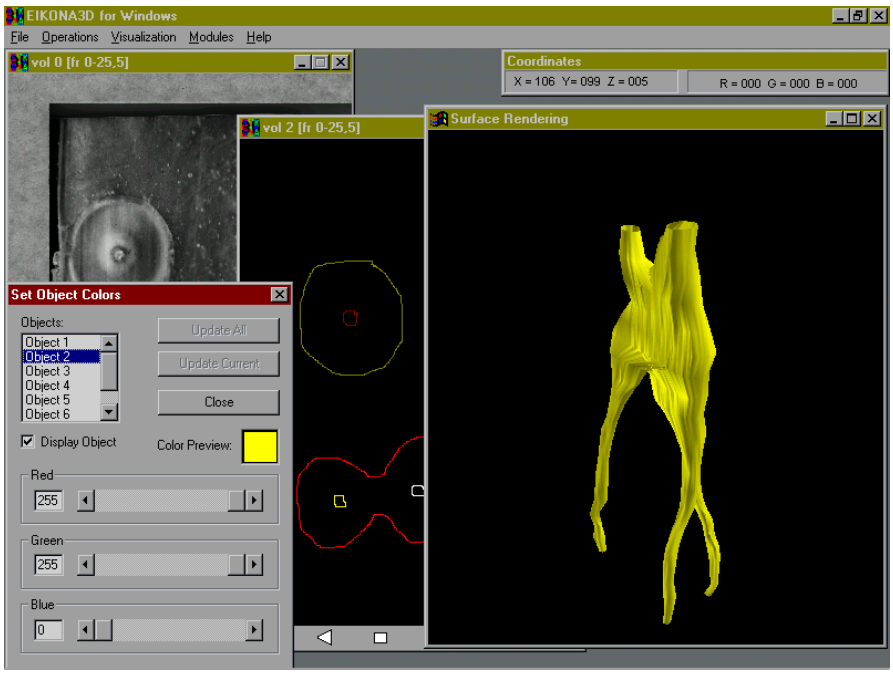


Figure 2: An example of interactive surface rendering of a 3D surface representation of a tooth, produced from an aligned sequence of frames containing contours. A view of root canals is shown.

***Set Object Colors*:** This menu item is enabled during an interactive rendering procedure of a script file. When it is selected, it displays the *Set Objects Color* dialog, which enables the user to set the RGB values for each identified object through the corresponding sliders or edit boxes, or hide/unhide any object through the *Display Object* toggle. Any changes are reflected to the rendering window through the *Update All* and *Update Current* buttons. The dialog is closed through the *Close* button and can be recalled many times while the rendering procedure is active. It should be noted that the script files do not encode the color of the objects. Thus, the colors assigned to the objects when an interactive rendering procedure is initialized are not the same as in the input volume from which the script file was built, but they result from a default color separation scheme. Through the *Set Objects Color* dialog the user can assign his preferred colors to the objects.

***Take Snapshot*:** This menu item is enabled during an interactive rendering procedure of a script file. It enables the transfer of the current content of the rendering window to the first frame of a volume selected through the *Select Output Volume* dialog box. By this way it can be saved to disk with the appropriate *File* menu options.

***Make Movie*:** This menu item is enabled during an interactive rendering procedure of a script file. It enables the production of a movie displaying a 360 degrees rotation of the surface around the vertical axis. The angle difference between two successive frames is inversely analogous to the specified number of movie frames. The user first specifies the number of movie frames through the *Make Movie* dialog, and then selects the output volume where the movie frames are to be written into through the *Select Output Volume* dialog box. Then, the movie can be saved to disk as a normal volume with the appropriate *File* menu options.

Automatic frame alignment module

The frames of certain image volumes, e.g. the ones coming from mechanical slicing in 3D microscopy, are usually misaligned to each other. Such an example is shown in Figure 1.

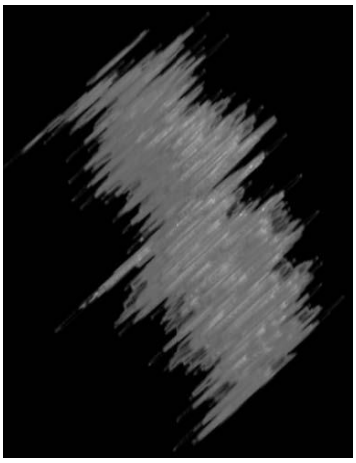


Figure 1: Misalign image volume.

Such volumes need manual or automatic alignment. EIKONA3D kernel provides a software tool for manual frame alignment. Automatic frame alignment is provided by the automatic alignment module. The volume Alignment module is implemented in `alignment.dll` file. If this DLL exists in the EIKONA3D folder, then, when EIKONA3D starts, it loads the DLL in a submenu with the name *Automatic Alignment* under the *Modules* menu. This submenu has the choice *Alignment*, which aligns a volume with no human intervention. If a user wants to align automatically a volume, he selects the input volume using the *Select Volume to Align* dialog box and he initiates the alignment procedure by pressing the button *OK*. A dialog box appears on screen asking for some necessary alignment initializations (Figure 2).

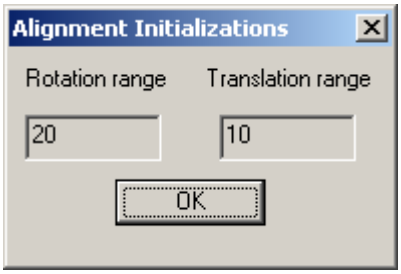


Figure 2: Dialog box with the necessary alignment initializations.

Initializations contain the rotation and the translation angles of the volume frames under consideration. The default initialization, covering a large number of alignment

cases, is the 20º (-20º, 20º) for the rotation and 10 (-10, 10) pixels for the **x**, **y** translation range respectively. However, if user believes that the volume under alignment has a smaller or larger rotation and translation range, then he can decrease or increase respectively those values. Attention! Those ranges MUST be positive.

When user finishes with the necessary initializations, the program is ready to start the alignment of the volume under examination, by just pressing the button *OK*. A status dialog box appears on screen indicating the status of the alignment procedure at each stage (Figure 3).

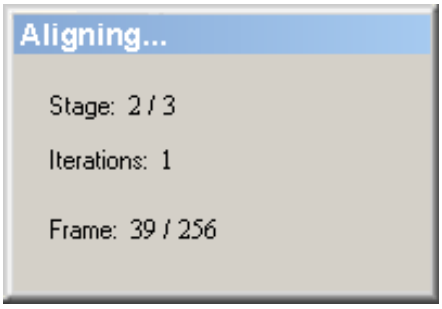


Figure 3: Status dialog box showing the status of the alignment procedure.

A result of automatic image volume alignment is shown in Figure 4.

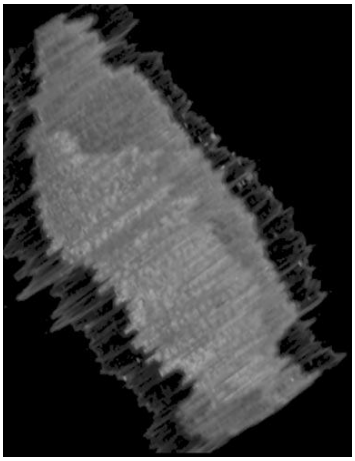


Figure 4: Aligned image volume.

Finally, when the volume alignment finished, the program ask user to give the name of the file that will contain all the alignment transformations. If the user wants to save the transformation file, he selects the desired path and name of the file and presses the *OK* button. This is a text file that contains the rotation (in degrees) and the translation in (x,y) coordinates for each volume frame. The (x, y) coordinates are saved as float

variables for increased accuracy but they are rounded prior to display. The point (0, 0) is the upper, left corner of the frame. Saving this file is optional, that is, if user presses the *Cancel* button, the program just by-passes this procedure and continues to the last alignment stage, where the program asks user to select the output buffer in order to display the aligned volume (Figure 5).

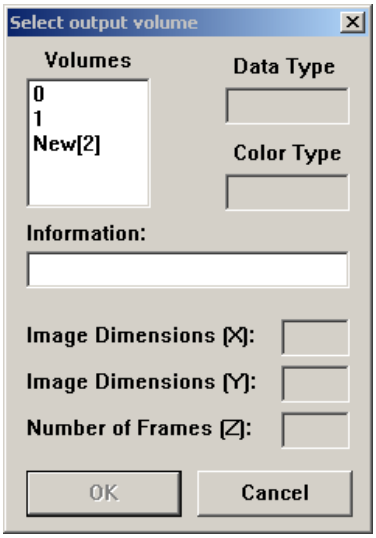


Figure 5: Dialog for selecting the output volume.

Marching Cubes Module

1. Introduction

Marching Cubes is an algorithm that transforms volumetric 3D object representation to 3D surface representation [1]. The 3D surface is represented in a triangular form. The Marching Cubes module initially reads the volumetric data the user has provided and creates a surface model of a 3D object. The user has the ability to save the surface model in various file formats as well as to directly show it on screen.

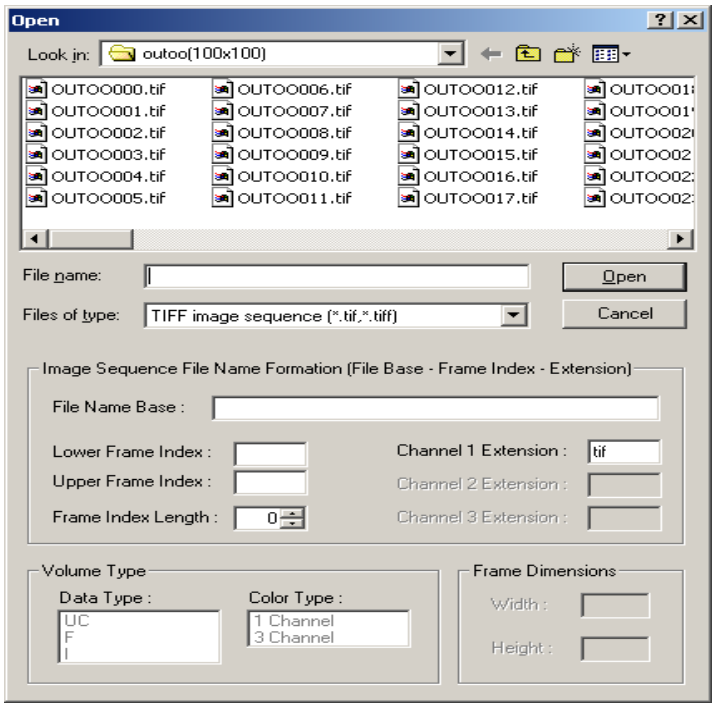


Figure 1: The *Open* dialog box.

Marching Cubes is a single document interface (SDI) Windows (Win32 architecture) application. It is totally coded in C++ programming language . All operations are executed through menu commands and dialogs which extend the user-friendliness of the EIKONA3D software. Each 3D model is displayed in an OpenGL platform inside the EIKONA3D workspace. The module comes in a single DLL (Dynamic Link Library) file called `marching_cubes.dll`. In order to install the module you only have to copy the file in the installation directory of EIKONA3D (where the `eikona3d.exe` file resides in) and restart EIKONA3D to recognize the new module. After launching the EIKONA3D package, the Marching Cubes module will create a new submenu under the name *Marching Cubes*. By choosing *File>Open*

from EIKONA3D main menu you can load a 3D object volume, i.e. volumetric data (serial object slices) to application's workspace from a file in disk.

Executing *Open* command will show up the file open dialog (Figure 1) from which navigation to the file system and selection of the desired volumetric data can be performed. The user has to choose one of the serially assigned slices (whichever) for the volumetric data to be loaded.

The *Marching Cubes* command under the *Modules* menu executes the Marching Cubes algorithm on the volumetric data the user has loaded. This command causes the execution of the Open Volume dialog (Figure 2) in which the user chooses the volume to be triangulated.

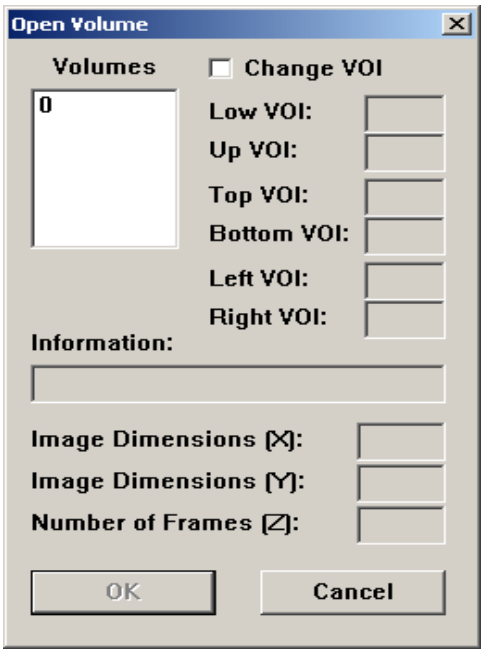


Figure 2: The *Open Volume* dialog box.

2. Marching Cubes Procedures dialog

Options for 3D Model tab: The Model dialog supplies the Marching Cubes Options for the 3D model (Figure 3).

Distance between slices text field: It represents the distance between the slices in the Z axis. Its value must vary from 1 to 50 voxels, otherwise an error message is displayed.

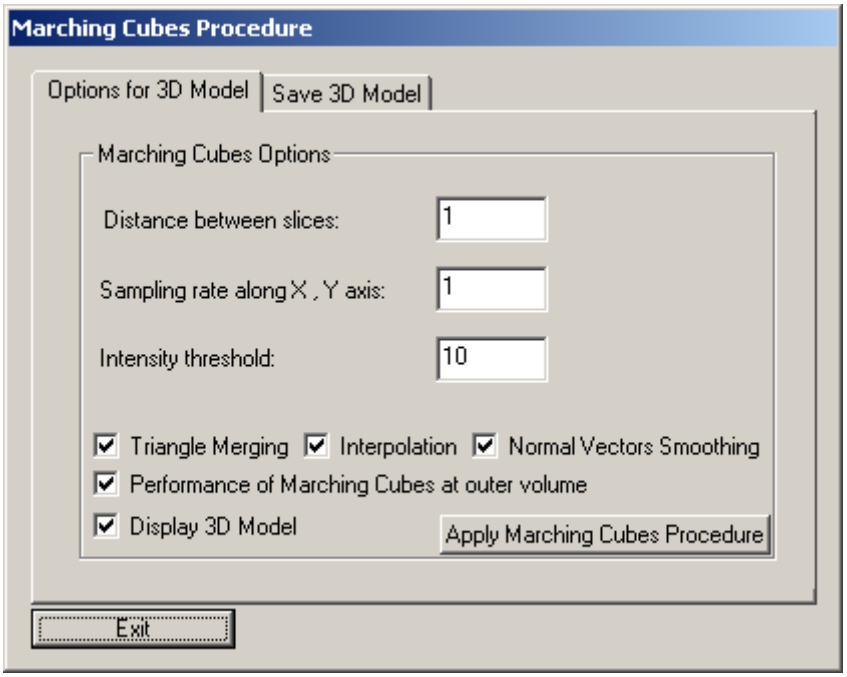


Figure 3: The *Options for 3D Model* dialog box tab.

Sampling Rate along X,Y axis text field: It represents the sampling rate of the pixels to be followed in order to create the new reduced in dimension slices from the original ones. Its value must vary from 1 to 20, otherwise an error message is displayed.

Intensity threshold: It represents the threshold according to which voxels that have intensity greater than or equal to that value are the ones that compose the 3D object, while the rest are background voxels. Its value must vary from 1 (black) to 255 (totally white), otherwise an error message is displayed.

Triangle Merging check box: If checked, the algorithm merges all coplanar triangles into larger polygons, so that the triangle number is reduced. No vertex coordinates are computed here. If not checked, the coplanar triangles are not merged.

Interpolation check box: If checked, the algorithm calculates the vertex locations by linear interpolation of local cube vertices. The corresponding normals are also computed here. If not checked, the mean of each local cube is taken into consideration for the calculation of the triangle vertex coordinates.

Normal Vector Smoothing check box: If checked, the algorithm calculates the normals for each vertex, estimates their arithmetic mean value and assigns it to that vertex. If not checked, no normal vector smoothing is applied.

Extraction of exterior surface only check box: If checked, the algorithm produces only the exterior object surface (i.e. it does not describe object holes).

Show 3D model button: When checked, an OpenGL platform is opened, where the 3D model is displayed (Figure 4).

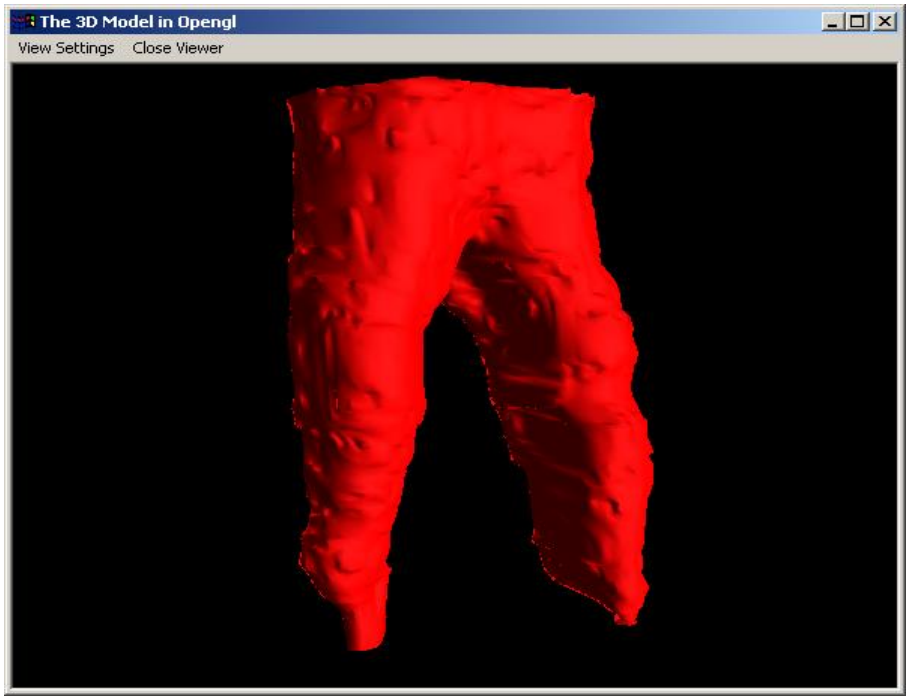


Figure 4: The *3D model in OpenGL* display window.

If the *Apply Marching Cubes* button has not been selected previously, an error message is displayed.

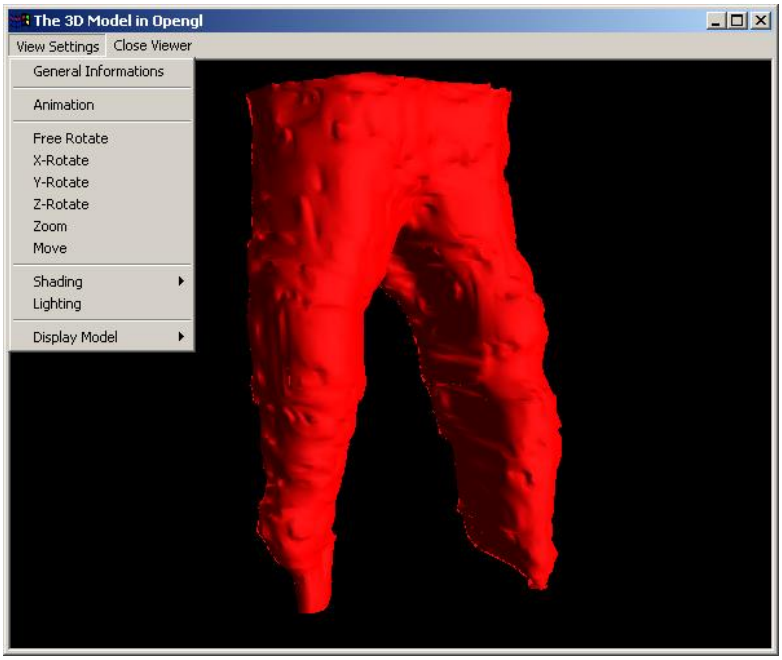


Figure 5: The *View Settings* menu.

The *View settings* menu (Figure 5) contains useful commands for manipulating the 3D model. Each command of the menu described in details bellow.

General Informations command: It opens the *Informations* dialog (Figure 6).

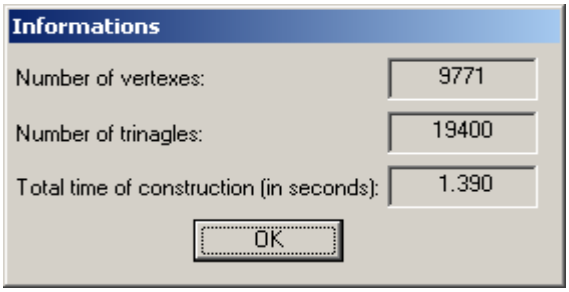


Figure 6: The *Informations* dialog box.

In this dialog the number of vertices, the number of triangles and the total calculation time (in seconds) of the 3D surface model are given.

OpenGL Animation command: It opens an animation settings dialog (Figure 7).

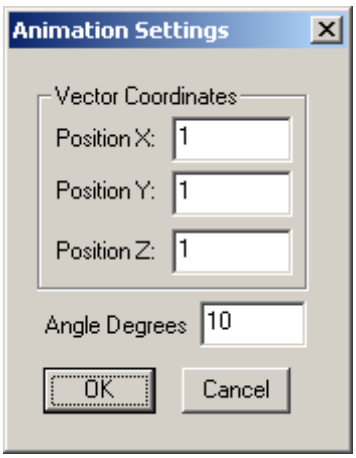


Figure 7: Animation Settings.

Its fields are:

Vector Coordinates panel:

Position X : It represents the X coordinate of the vertex around which the 3D model is to be rotated.

Position Y: It represents the Y coordinate of the vertex around which the 3D model is to be rotated.

Position Z: It represents the Z coordinate of the vertex around which the 3D model is to be rotated.

For all of the above coordinates, their values must not be zero concurrently. Otherwise, an error message is displayed.

Angle degrees: It represent the degrees around which the vector is to be rotated. Its value must vary from 1 to 359 degrees. Otherwise, an error message is displayed.

Free Rotate command: It allows the user to freely rotate the 3D surface model using the mouse.

X-Rotate command: It allows the user to rotate the 3D model along the X axis using the mouse.

Y-Rotate command: It allows the user to rotate the 3D model along the Y axis using the mouse.

Z-Rotate command: It allows the user to rotate the 3D model along the Z axis using the mouse.

Zoom command: it allows the user to zoom in moving the mouse to the upper side of the window and to zoom out moving the mouse to the opposite side of the window.

Move command: Allows the user to translate the model using the mouse.

Shading submenu:
Flat command: No smoothing of the triangles is applied.
Smooth tab: Smoothing of the triangles is applied. The smoothing procedure is applied by default.

Lighting menu command: It enables/disables the lighting of the 3D model. The default choice is the enabled lighting.

Display Model submenu:
Solid command: Ist displays polygon filled with texture (Figure 8).

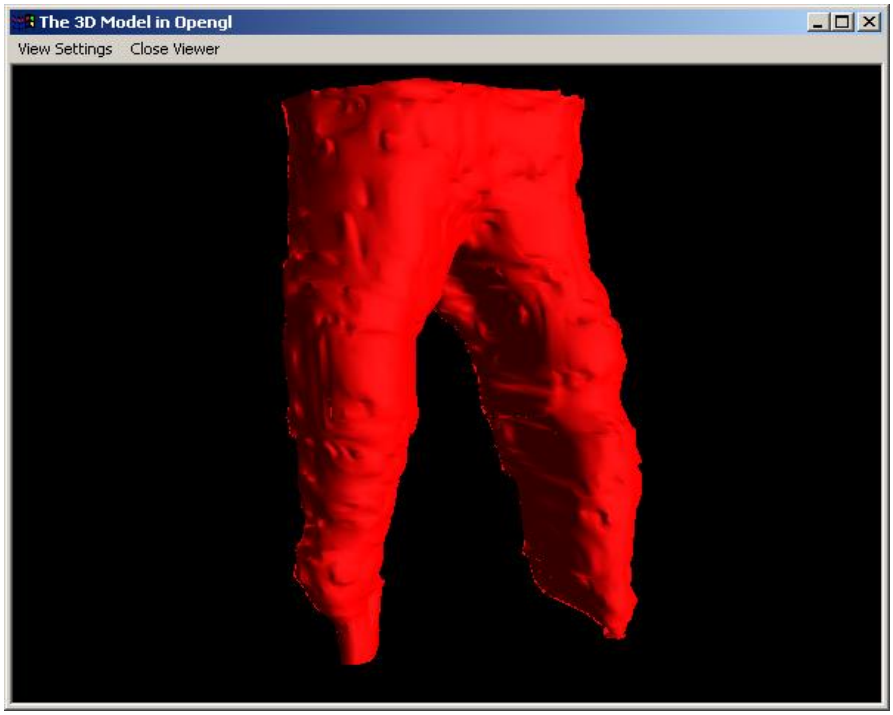


Figure 8: Displayed solid 3D surface model.

Wireframe command: It displays a 3D surface model as wireframe (Figure 9).

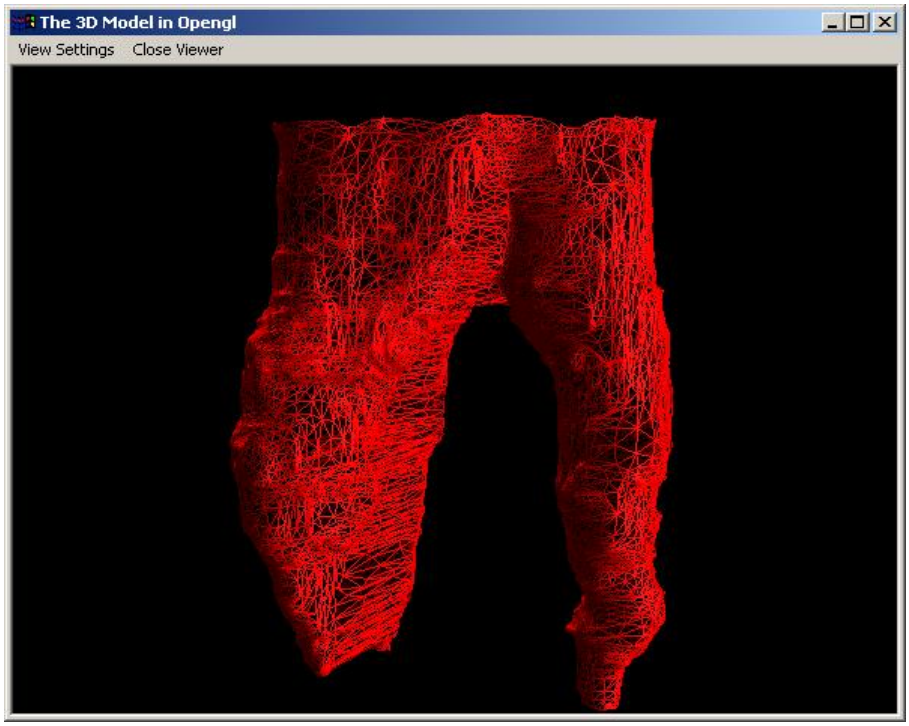


Figure 9: Wireframe display of a 3D surface model.

Vertices command: It displays only the vertices of a 3D model (Figure 10).

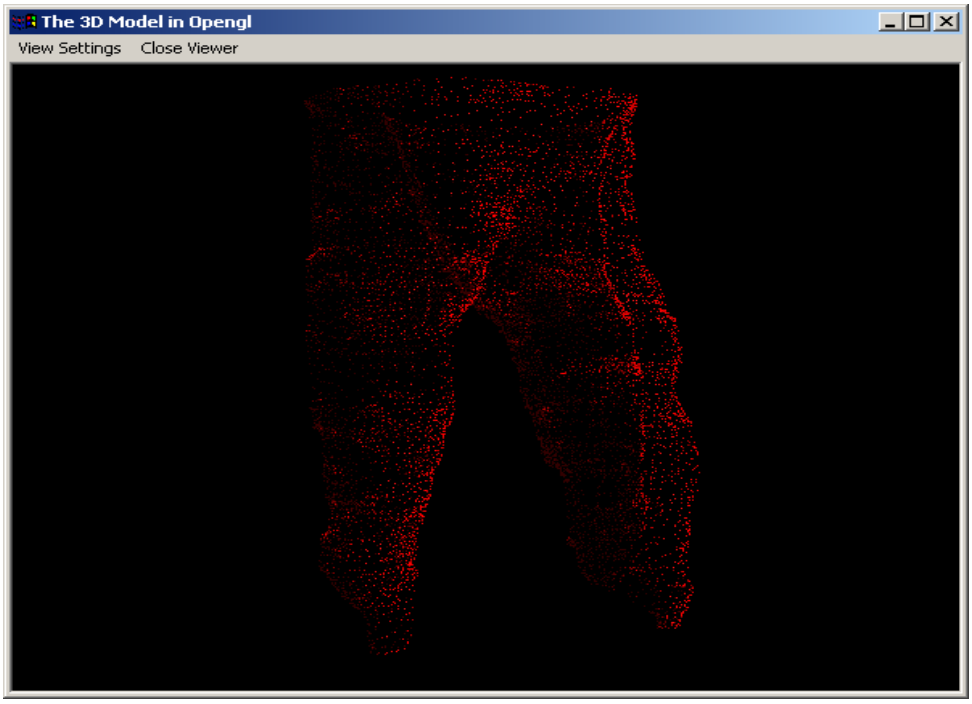


Figure 10: Vertex display of a 3D surface model.

Close Viewer menu:

Close command: It closes the OpenGL platform.

When pressing the *Show 3D Model* button, besides having the 3D model displayed, a short message '*Wait, the 3D Model is displayed'* is presented on the left side of the button, inside the small box, otherwise, a short message '*3D Model is not displayed'* is presented on the right side of the button.

Save 3D Model tab:

Folder text field:

The user must select the destination directory where the 3D model will be saved (Figure 11). The destination directory can be selected by the browse function.

The name of the file is provided in the *File Name* text field. In case of missing destination path or file name when the user has chosen to export the result in a file, an error message is displayed.

Export Model panel:

Export model to wrl file checkbox: If checked, the 3D model is exported to a VRML2 format file.

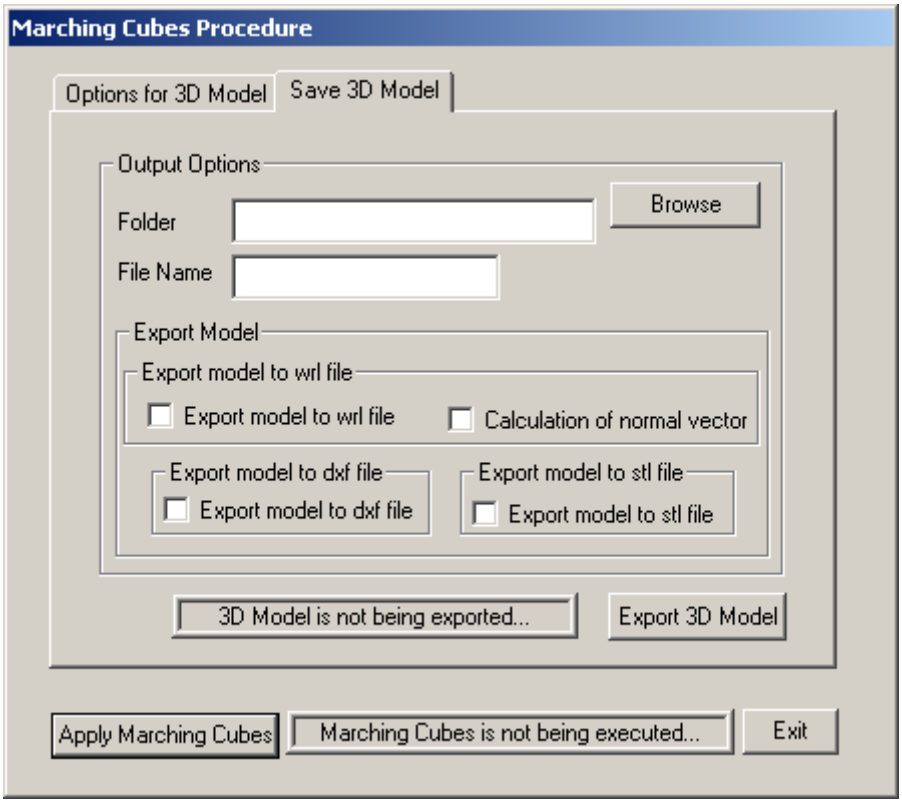


Figure 11: The *Save 3D Model* dialog box.

Calculation of normal vector checkbox: If checked, it supplies the exported file with the normal vectors that have already been calculated. Otherwise, the exported file does not carry normal vector information.
Export model to dxf file checkbox: If checked, the 3D model is exported to a AUTOCAD file format.

Export model to stl file checkbox: If checked, the 3D model is exported to a stereo lithography (STL) file format, that is used in 3D scanners as well as in 3D printers.

Export 3D Model button: When pressing the *Export 3D Model* button, besides having the 3D model being exported, the message *Wait, the 3D Model is being exported* is presented on the left side of the button, inside the small box, otherwise , the message *3D Model is not being exported* is presented on the right side of the button.

Apply Marching Cubes button: When pressing the *Apply Marching Cubes* button, besides having the Marching Cubes algorithm executed, the message *Wait, Marching Cubes is being executed* is presented on the right side of the button, inside the small

box, otherwise, the message *Marching Cubes is not being executed* is presented on the right side of the button.

Exit button: It is used to exit the Marching Cubes module.

3. Error messages

Can not allocate internal buffer: can not allocate memory for marching cubes procedure.

fopen error: Can not open file .

4. References

[1] N.Nikolaidis, I.Pitas '3D image processing algorithms', Wiley 2000.

3D Surface modeling module

This module performs the dynamic surface modeling of a 3D object by surface deformations. The deformations performed exploit modal analysis [1]. The input volume for this module should be a grayscale (thresholded) volume, like the one shown in Figure 1.

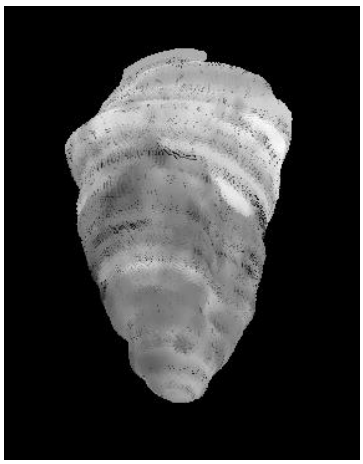


Figure 1: 3D object used for surface modeling.

1. Modal Analysis Method

This object surface modeling method is described in [1]. The 3D object is originally described by a 3D spherical surface containing N×M×L nodes. During deformation, this surface is fit to the 3D object surface. Once a volume containing a 3D object has been opened using *File>Open*, its surface modeling can be performed by selecting the option *Modules>Surface Deformation>Modal Analysis* from the main menu which brings up the Interactive Modal Analysis dialog box depicted in Figure 2.

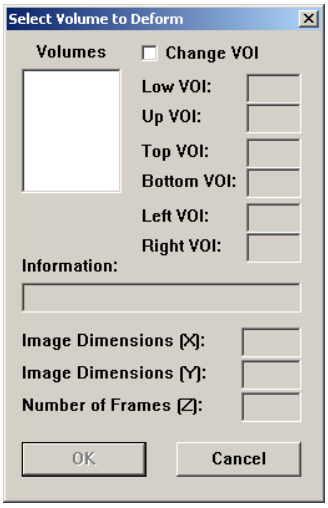


Figure 2: The Interactive Modal Analysis dialog box.

The dialog box is used to select the source volume buffer that contains the 3D object whose surface should be modeled. Afterwards, the *Model Properties* dialog box appears, which is used to provide the properties of the deformation model that is going to be used. The four parameters used in this dialog box are:

a) the characteristic value of the model, which characterizes how deformable the model will be. It should be greater than 0.
b) The percentage of the low-frequency eigenvalues used in the modal analysis algorithm.
c) The height and the width of the model (in number of nodes).

The dialog box can be seen in Figure 3.

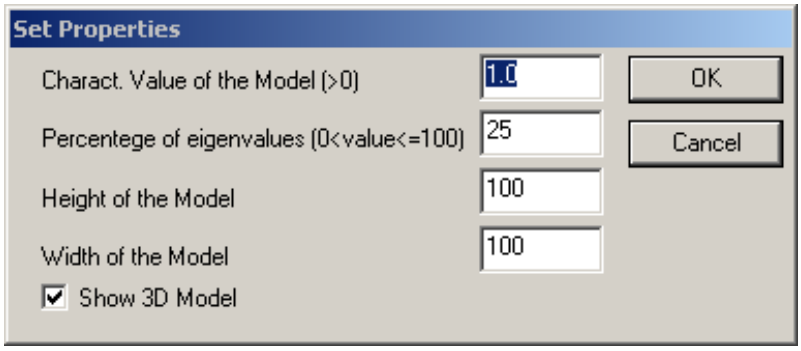


Figure 3: Model properties dialog box used by Modal Analysis.

Also in this dialog box there is the option *Show 3D Model*. If this option is checked, an OpenGL platform is opened, where the 3D surface model is displayed (see Figure6).

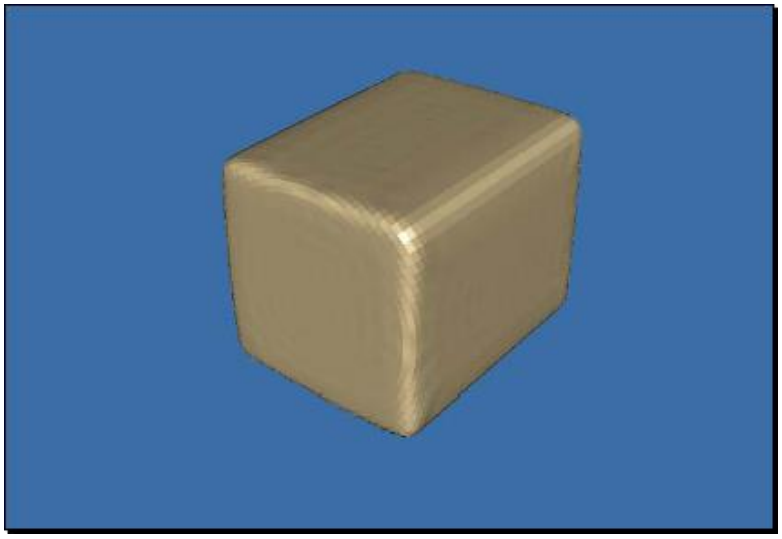


Figure 4: VRML Surface model of a cubic 3D object.

The derived surface of the volume is saved in a VRML file after prompting the user for the desired output file and path name, using a *Save Deformed Volume As…* dialog box. The obtained VRML output file of a volume containing a cube is shown in Figure 4.

The result of the surface modeling of the object shown in Figure 1 is illustrated in Figure 5.

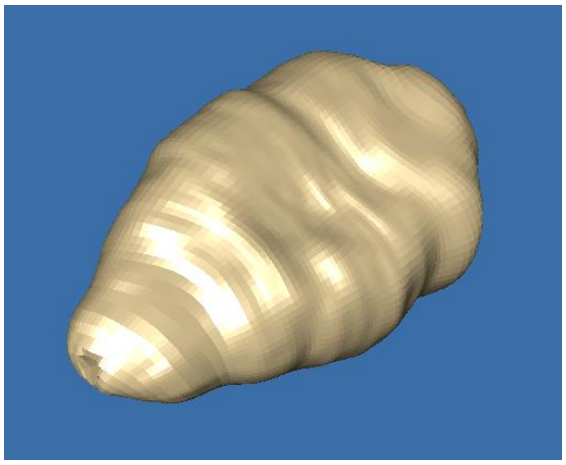


Figure 5: VRML Surface model of the object shown in Figure 1.

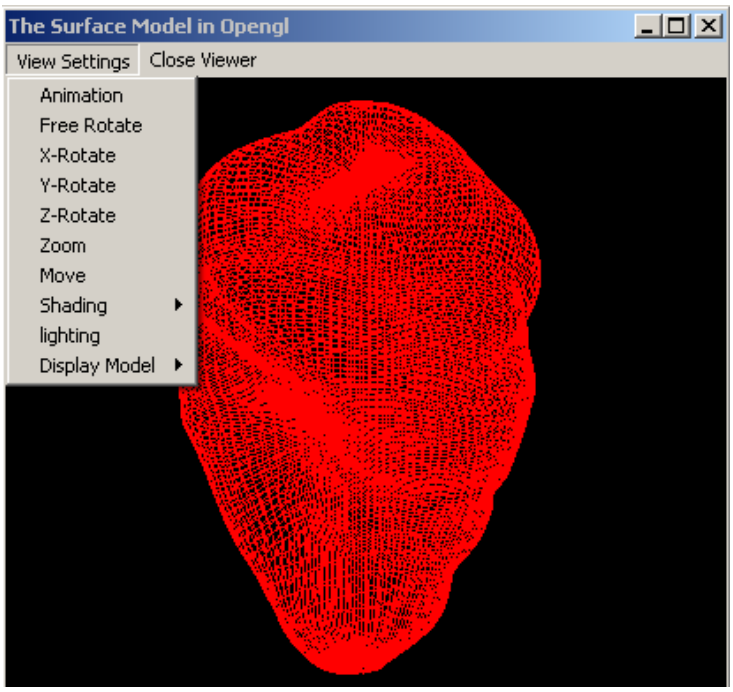


Figure 6: The Surface Model *in OpenGL* display window.

The *View settings* menu (Figure 6) contains useful commands for manipulating the 3D model. Each command of the menu is described in details below.

OpenGL Animation command: It opens an animation settings dialog (Figure 7).

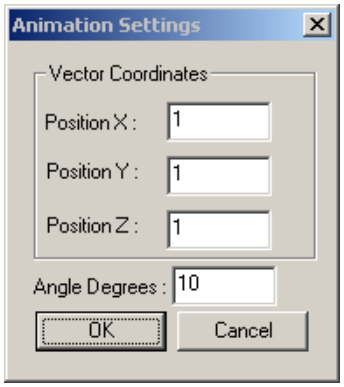


Figure 7: Animation Settings.

Its fields are:

Vector Coordinates panel:

Position X: It represents the X coordinate of the rotation axis.

Position Y: It represents the Y coordinate of the rotation axis.

Position Z: It represents the Z coordinate of the rotation axis.

For all of the above coordinates, their values must not be zero concurrently. Otherwise, an error message is displayed.

Angle degrees: It represents the rotation angle in degrees. Its value must vary from 1 to 359 degrees. Otherwise, an error message is displayed.

Free Rotate command: It allows the user to rotate the 3D surface model freely using the mouse.

X-Rotate command: It allows the user to rotate the 3D model along the X axis using the mouse.

Y-Rotate command: It allows the user to rotate the 3D model along the Y axis using the mouse.

Z-Rotate command: It allows the user to rotate the 3D model along the Z axis using the mouse.

Zoom command: It allows the user to zoom in by moving the mouse to the upper side of the window and to zoom out by moving the mouse to the opposite side of the window.

Move command: It allows the user to move the model using the mouse.

Shading submenu:
Flat command: No smoothing of the surface model is applied.
Smooth tab: Smoothing of the surface model is applied. The smoothing procedure is applied by default.

Lighting menu command: It enables/disables the lighting of the 3D model. The default choice is the enabled lighting.

Display Model submenu:
Solid command: It displays a polygonal surface as a solid surface (Figure 8).

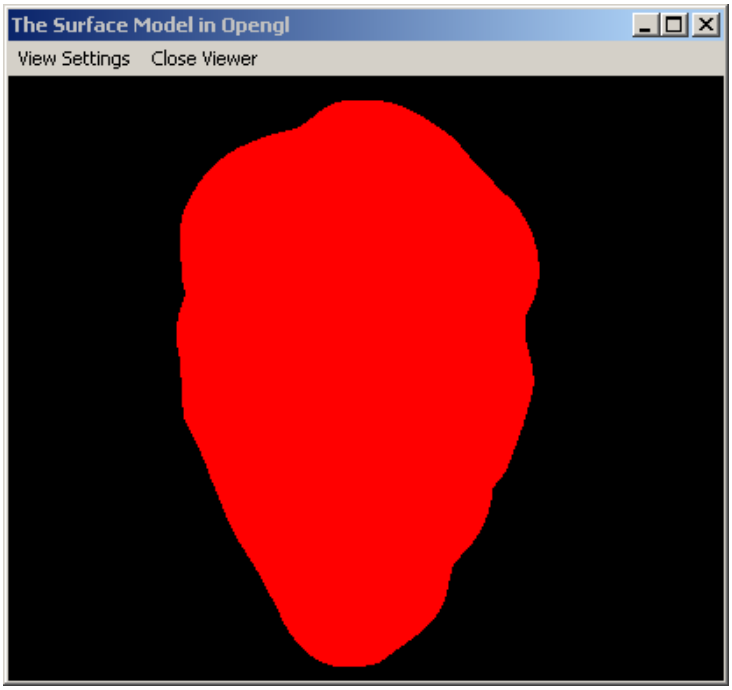


Figure 8: Displayed solid 3D surface model.

Wireframe command: It displays a 3D surface model as wireframe (Figure 9).
Close Viewer menu:
Close command: It closes the OpenGL platform.

2. Solve Equations method

This module performs dynamic surface modeling of a 3D object. The method used is described in [2]. The 3D object is originally described by a 3D spherical surface containing N×M×L nodes. During deformation, this surface is fit to the 3D object

surface. The input volume for this module should be a grayscale (thresholded) volume.

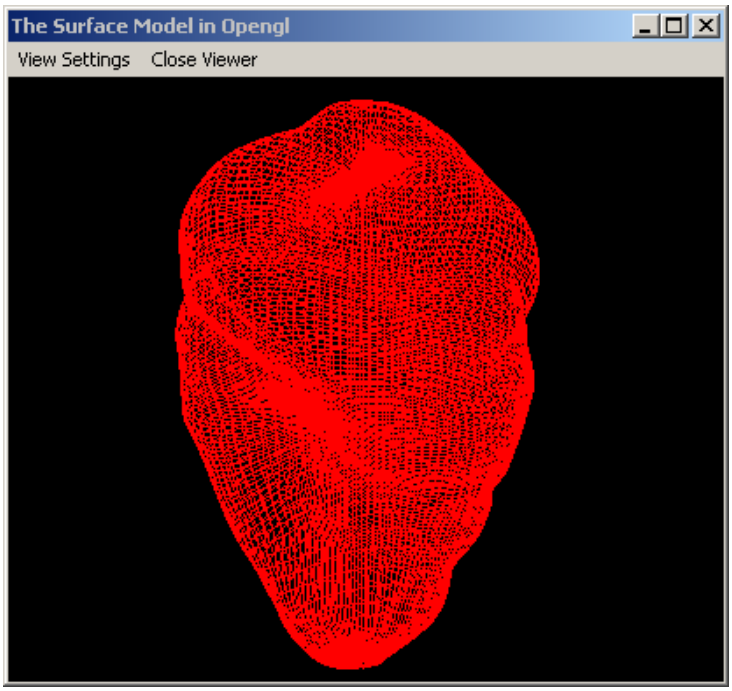


Figure 9: Wireframe display of a 3D surface model.

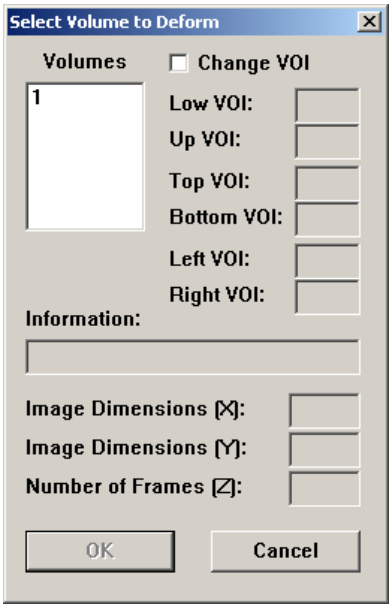


Figure 10: The Interactive Solve Equations dialog box.

Once a volume has been opened using *File>Open*, surface deformations can be performed by selecting the option *Modules>Surface Deformation>Solve Equations* which brings up the *Interactive Solve Equations* dialog box depicted in Figure 10.

The dialog box is used to select the source volume buffer containing the 3D object to be modeled. Afterwards, the Model Properties dialog box appears, which is used to provide the properties of the deformation model that is going to be used (the dialog box can be seen in Figure 11).

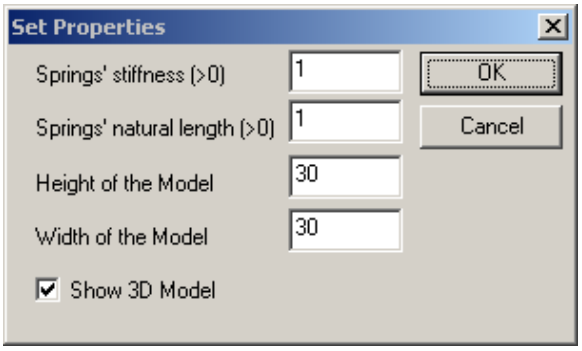


Figure 11: Model properties dialog box used by Modal Analysis.

The four parameters used in this dialog box are:

a) the *spring stiffness* (should be greater than zero).

b) The *natural length* of the springs used by the model (which must be positive).

c) The *height* and the *width* of the model (in number of nodes).

Also in this dialog box there is the option *Show 3D Model*. If this option is checked, an OpenGL platform is opened, where the 3D surface model is displayed (see Figure6).

The deformed surface of the volume is saved in a VRML file after prompting the user for the desired output file and path name using a *Save Deformed Volume As...* dialog box. The VRML output file of the 3D object of Figure 1 is shown in Figure 12.

The VRML output files can be viewed by using a VRML player.

Figure 12: VRML surface model of the object shown in Figure 1.

3. References

[1] S. Krinidis, I. Pitas. ‘Fast Free Vibration Modal Analysis of 2D Physics-Based Deformable Objects’. IEEE Transactions on Image Processing, to be published on March 2005.

[2] K. J. Bathe. ‘Finite Element Procedure’. Prentice Hall, Englewood Cliffs, New Jersey, 1996.

3D skeletonization module

1. Background

A binary 3D object can be represented by its skeleton, which consists of the spines of the object parts. A 3D skeleton may consist of 3D curves (spines), isolated 3D points and, possibly, 3D surfaces. More information on 3D skeletons can be found in [1]. The generation of a skeleton is realized by applying an iterative process which erodes the object layer by layer until only the object spines remain, which form the skeleton of the 3D object. This iterative process is called *thinning*. A thinning algorithm contains a set of voxel deleting conditions, which enable it to erode the object iteratively. The thinning process should support the following properties, so that the thinning result can be characterized as a skeleton of the 3D binary object:

- *Geometry preservation* is a major concern of thinning algorithms. For example, an object like “b” should not be converted into an object like “o”. To preserve the geometry of the original image, a thinning algorithm must contain certain geometry preserving conditions.
- *Topology preservation* is the second major concern of thinning algorithms. For example, an object like “o” should not be converted into an object like “c”.
- Finally, a skeleton of an object should be, ideally, as thin as possible (one voxel wide) and should represent the object through its spine or *medial axis*.

A skeleton of an object can have a variety of applications in 3D image processing, such as:

- *Automatic path generation*. 3D volume visualization of objects, especially in medical applications, can be beneficial from the patient’s side. Such a process can implement virtual endoscopy, where the navigation of a virtual camera can be guided by a path inside an organ. The skeleton of the organ can serve as such an automatically generated path.
- *Shape matching, identification, classification*. As skeletons retain only vital information about the shape of an object, the object modeling is simplified.
- *Length measurement*. By representing an object through its medial axis length measurement can be simplified in 3D space.

- *Compression*. A 3D skeleton may need substantial less information for its description than the entire 3D object itself.

2. 3D Skeletonization Module Overview

3D Skeletonization is a module of EIKONA3D software package. It conforms to the module expansion capability of EIKONA3D and offers a thinning function available for the user. The function is called *3D Pattern Thinning* function which based on a pattern-based thinning algorithm and extracts the skeleton of a single volume. The module can be applied only on *binary* volumes. A binary volume contains only two intensity values, a *foreground* voxel value (255), and a *background* voxel value (0). A simple way to create a binary volume is by *thresholding* (a basic operation of EIKONA3D). This module is designed to work for all versions of EIKONA3D.

3. Installation

The module comes in a single DLL (Dynamic Link Library) file called `3dthin.dll`. In order to install the module you only have to copy the file in the installation directory of EIKONA3D (where the `eikona3d.exe` file resides in) and restart EIKONA3D to recognize the new module.

4. Use of 3D Skeletonization Module

After launching the EIKONA3D package, the skeletonization module will create a new submenu under the name *3D Skeletonization,* as illustrated in Figure 1.

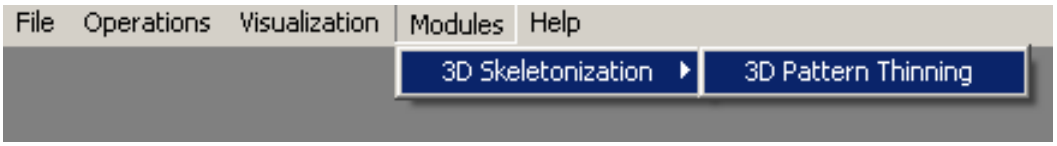


Figure 1: *3D Skeletonization* menu function in EIKONA3D.

The skeletonization module operates on a single input volume that EIKONA3D handles. The output is a volume containing the generated skeleton of the object(s) comprising the input volume. It is noted again that both the input and the output volumes of 3D Skeletonization should be binary volumes. Even if the user selects a grayscale volume, it is converted inside the module to a binary one by thresholding it with a voxel intensity threshold equal to 1. This means that every voxel whose value is greater than or equal to the threshold takes a value of 255.

It is worth noting that not every 3D object can be thinned to generate a well formed 3D skeleton. A *well-composed* object is likely to be thinned to a better result. 3D binary object preprocessing by 3D interpolation and/or filtering can help producing a satisfactory result.

5. 3D Skeletonization operations

5.1 3D Pattern Thinning

When the user selects the 3D Pattern Thinning operation a dialog box appears permitting the user to select the input volume. The input dialog filters only the grayscale volumes already loaded in EIKONA3D. After the selection of an input volume, a second dialog box appears permitting the user to select the output volume. The output volume can be placed either on an existing buffer or on a new one. A dialog will appear displaying the progress of the thinning operation as illustrated in Figure 2.

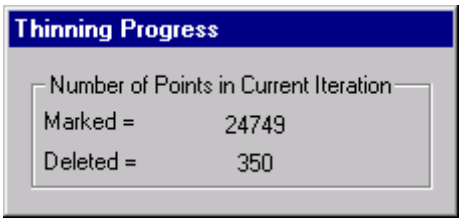


Figure 2: Dialog which displays the thinning progress.

The algorithm works iteratively and it is time-consuming. In each iteration, a number of object voxels is marked in order to be tested against a set of deleting conditions. The box *Marked* in the dialog shows the number of voxels currently marked for checking. If an object voxel satisfies the deleting conditions, it is deleted. The box *Deleted* in the dialog shows the number of object voxels deleted so far from the set of the marked object voxels.

5.2 Thinning Example

Figure 3 illustrates the skeleton extracted using the 3D pattern thinning operation on a pulmonary tree. The 3D data were obtained from CT slices.

Figure 3: a) 3D rendering of pulmonary airway tree. b) The skeleton produced by the 3D pattern thinning algorithm.

6. Performance

The performance of 3D Pattern Thinning depends on the number of object points included in the volume and the object shape itself. A well-shaped object could be more easily thinned and take less time than a non-smooth object. Generally speaking, the thinning process using this operation is a time consuming operation for even medium sized volumes (having frame size 128×128 or 256×256 pixels). For example, the thinning process on the volume displayed in Figure 3 (size 128×128×128) took about 5 minutes to complete on a single Pentium II class CPU at 400 MHz or above. The math morphology thinning runs significantly faster because of the type of operations it performs.

ICP module

The Iterative Closest Point Algorithm module is the file `icpa.dll`. If this DLL exists in the directory of EIKONA3D when the program starts, a sub-menu called *ICP Algorithm* is added under the *Modules* menu. The menu options of this sub-menu provide two methods (*Classical* and *Morphological*) of Iterative Closest Point Algorithm (ICPA).

The Classical method accomplishes the original ICPA which proposes a solution to a key registration problem in computer vision: given a cloud of 3D model points and a cloud of 3D data points, estimate the optimal translation and rotation that register the two point clouds by minimizing the mean square Euclidean distance between them. This method is described in [1]. A crucial drawback of this method is the high computational complexity of the closest point operator. The Morphological method solves this problem by using a 3D array representing a volume and by constructing the Voronoi diagram of the model points within this volume, by using the morphological Voronoi tessellation method. Then, the ICP algorithm is employed, with distance calculations substituted by simple array references.

An important application of the Iterative Closest Point Algorithm is to register actual data sensed from a 3D object with an ideal 3D model. It is also useful in multimodal 3D image registration in medical imaging (e.g. between NMR and CT volumes), in the shape equivalence problem, as well as in estimating the motion between point sets when the point correspondences are not known. The procedure followed for each method in our module is described below:

1. Classical ICP Algortihms

This menu option performs the Classical ICP algorithm. In this method the user has first to select the model and data volumes (Figure 1-2) which can be only grayscale. Normal projection of the object seen in Figures 1-2 can be seen in Figure 3. It is strongly recommended that the model and data volumes should have the same size. Then he has to define the appropriate parameters for the *Initial Rotation Vector* and the *Threshold* through the *Classical ICP Algorithm* dialog box (Figure 3). These two parameters are described in detail below.

Figure 1: A frame of the model volume.

Figure 2: A frame of the data volume (the model volume rotated and translated along x-axes and translated along y-axes).

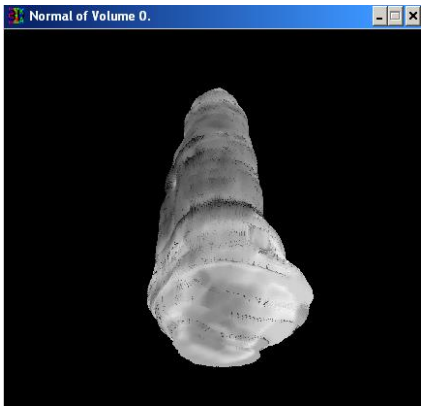


Figure 3: Normal projection of the object seen in Figures 1-2.

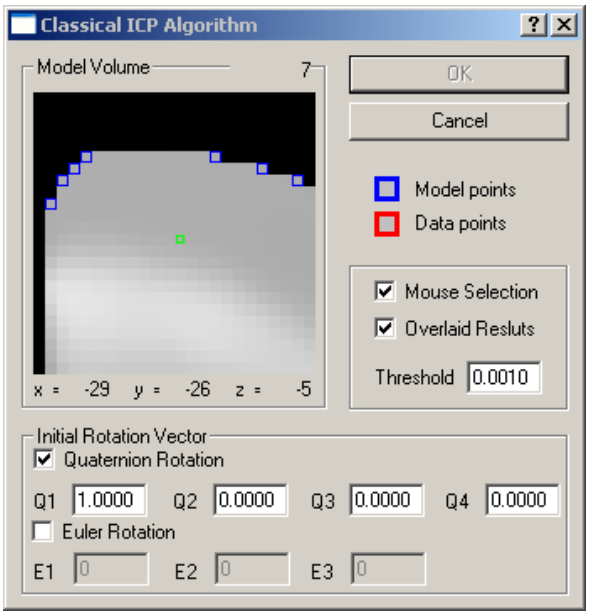


Figure 4: The Classical ICP Algorithm dialog box.

Initial Rotation Vector: There are two options. User can select one initial vector each time. The Quaternion Rotation is a four dimensional vector which is applied to the data points set before the ICP Algorithm starts. A possible convergence of the algorithm in a local minimum depends on this vector. The only way to make sure that we reach the global minimum is to start the algorithm from different initial rotation vector. So if you are not satisfied with the result or if the final mean square error is not small enough, try another initial rotation vector. The rotation vector (1,0,0,0) corresponds to no rotation. The Euler Rotation is a three dimensional vector and works in a similar manner with the previous rotation vector. (The rotation vector (0, 0, 0) corresponds to no rotation).

Threshold: This is used to terminate the algorithm when the change in mean square error, between two iterations of the closest point operator, falls below threshold. You can reduce this threshold if you are not satisfied with the result. Also in this dialog box there is the option *Mouse Select*. If this option is checked, the user is able to select with his mouse the model and data points by clicking on the corresponding volumes. Alternatively, he can define automatically these two points sets by selecting two volumes, after he has pressed the *OK* button and if the option *Mouse Select* is not checked. In such a case, the points of each set are defined by the non-zero voxel values of the selected volumes, which must be grayscale. A simple method to take these two volumes from the model and data volumes is by applying a threshold to them. In the dialog box, there is also the option Overlaid results. If this option is checked, when the process terminates, the user will be prompted to select an output volume where the model and the registered volumes are shown overlaid to the same buffer. Furthermore, there is a view area, where the user can see an enlarged small part of the model or data volume. He can also see the current position of the mouse in the volume and the number of points of the corresponding point set. In this area a model point is displayed with a blue rectangle around it, while a data point is displayed with a red rectangle around it. This allows him to select and view the model and data points.

On the next step, if the user has decided to select the model and data points sets automatically, he will be prompted to select the corresponding volumes. After this the Classical ICP Algorithm starts execution. A progress dialog box shows the number of

model and data points and the mean square error of the current iteration. At this point the user is able to interrupt the whole process by pressing the *Cancel* button.

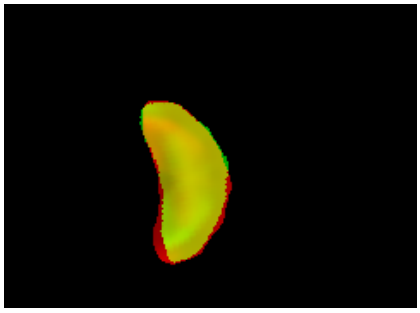


Figure 5: A frame of the overlaid model (red channel) and registered (green channel) volumes of Figures 1-2.

When the algorithm terminates the user is prompted to select the output volumes where the data volume will be registered and where the overlaid model and registered volumes will be shown (Figure 5) and then the *Results* dialog box (Figure 6) will appear. This dialog box shows the results of the algorithm such as the rotation (both in quaternion and Euler space) and translation vectors. Furthermore, it displays the duration of each step of the algorithm:

CPO: This refers to the calculation of the closest point correspondence only.

CPO mean: This is the mean duration of the CPO for one iteration of the algorithm.

ICP pure: This refers to the time needed for ICP algorithm, excluding CPO.

ICP: This is the total duration of the ICP algorithm.

Finally, there is the same view area with the *Classical ICP Algorithm* dialog box, which helps the user to check, if the result of the algorithm is acceptable, since he can see together the model and the registered data points and the correspondence between them.

2. Morphological ICP

This menu option performs the Morphological ICP method. In this method, the user has first to select the model and data volumes and then the model and data points volumes. Attention should be paid to the model and data points sets in order to avoid unexpected results. For example, the model points volume must be large enough to include the data points set and its probable rotation during the ICPA. Otherwise it is possible that some of the data points fall out of the Voronoi tessellation volume and

the result may not be reliable enough. At this point we have to mention that the center of the axes is the middle of the volume. Thus, it is preferable to have the same dimensions for all used volumes.

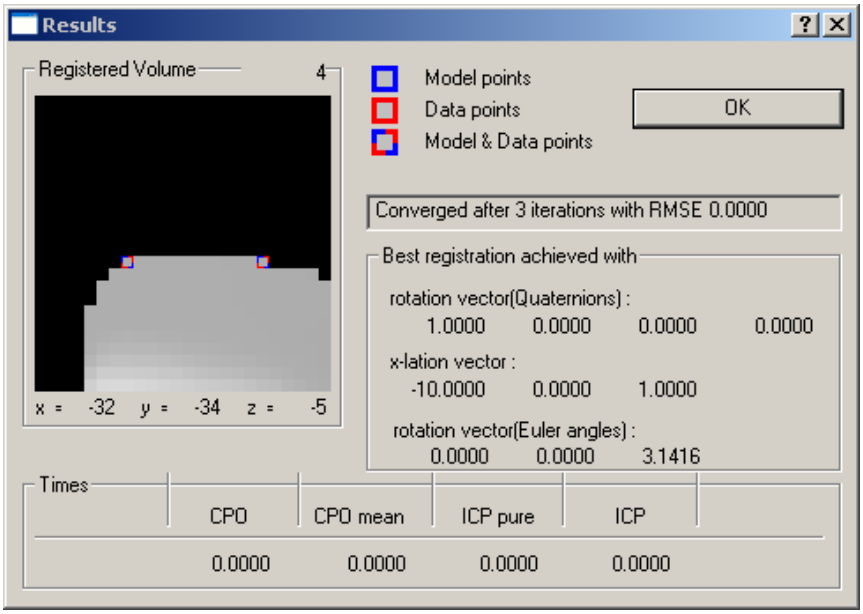


Figure 6: The *Results* dialog box for the Classical ICP Algorithm.

Through the *Morphological ICP Algorithm* dialog box the user can define the *Initial Rotation Vector* and the *Threshold* parameters which are the same to the ones of the *Classical ICP Algorithm* dialog box. In the dialog box, there is also the option Overlaid results. If this option is checked, when the process terminates, the user will be prompted to select an output volume where the model and the registered volumes are shown overlaid to the same buffer.

Also in this dialog box he can decide whether he will save or load the result volume of the Voronoi tessellation process by checking the corresponding options (Figure 7) which are described below:

Load Tessellation: If this option is checked, the user will be prompted to select the tessellation file from the disk before the ICP Algorithm starts. This tessellation file must correspond to the currently selected model points volume, otherwise the result will not be correct.

Save Tessellation: If this option is checked, the user will be prompted to enter a name for the tessellation file after the Voronoi tessellation process. This tessellation file can

be used in the future when we want to register another data volume with the same model volume.

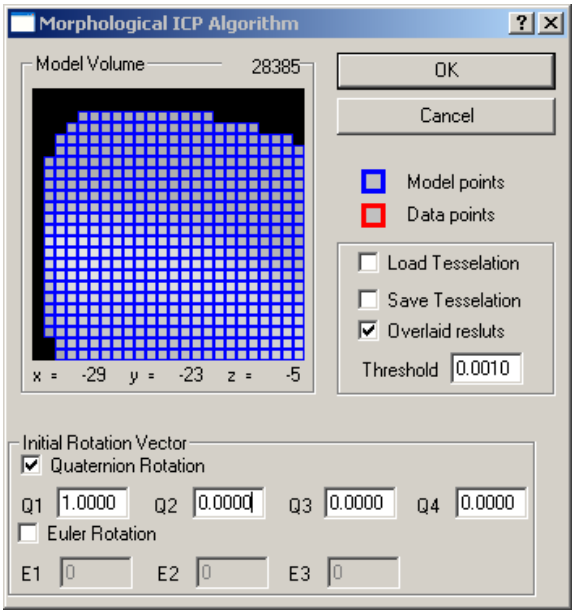


Figure 7: The Morphological ICP Algorithm dialog box.

With these two options we can avoid the time consuming tessellation process, if we want to run the Morphological ICP Algorithm several times for the same model and model points volume.

After this, the Voronoi tessellation starts the execution if the user has not loaded a tessellation file, while a progress dialog box shows the current elapsed time and the estimated time for this process. Then, the Morphological ICP Algorithm starts and the progress dialog box shows the number of model and data points and the mean square error of the current iteration. At this dialog box the user is able to interrupt the whole process by pressing the *Cancel* button.

When the algorithm terminates the user is prompted to select the output volume where the data volume will be registered and where the overlaid model and registered volumes will be shown (Figure 5) and then the *Result* dialog box (Figure 8) will appear.

This dialog box is the same with the Classical ICP method except for two elements:
VT: This refers to the duration of the Voronoi tessellation process.
ICP full: This is the entire duration of the Morphological ICP algorithm.

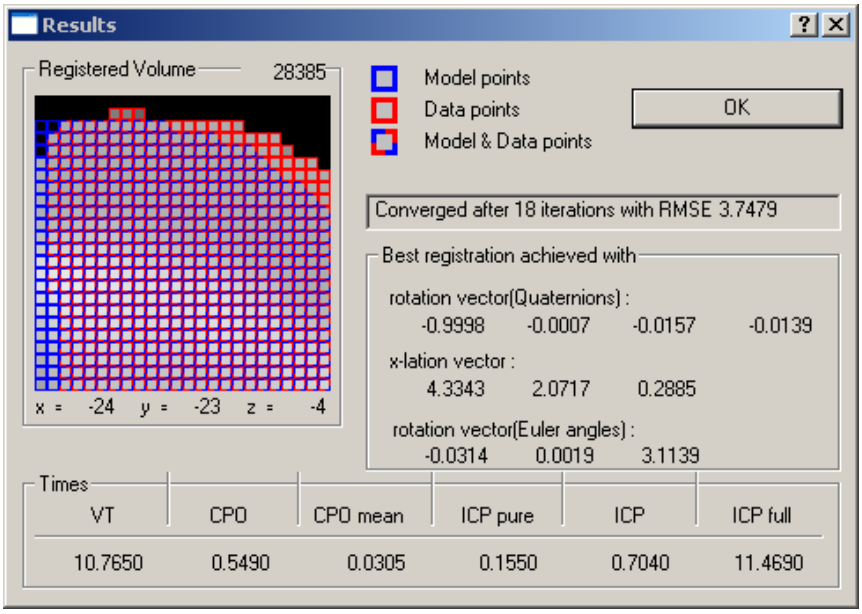


Figure 8: The *Results* dialog box for the Morphological ICP Algorithm. The only difference with the corresponding dialog box of the Classical ICP Algorithm is the times VT and ICP full.

Finally we have to mention that a help context has been implemented so the user can have a small description of each element at any dialog box.

References

[1] N.Nikolaidis, I.Pitas '3D image processing algorithms', Wiley 2000.

Virtual tooth drilling module

The virtual tooth drilling module is a simulator of the operation of the dental bur tool. It can be used as a tool for training students in a Department of Dentistry.

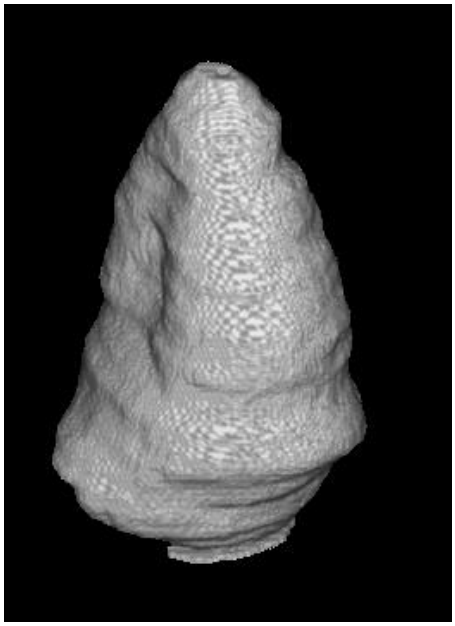


Figure 1: 3D tooth.

1. Opening a Tooth Volume

A 3D virtual tooth volume is a 3D tooth shape representation as a sequence of images corresponding to a set of successive tooth slices, obtained either by mechanical slicing the tooth or by micro CT. Such a virtual tooth is shown in Figure 1. Its cross section is shown in Figure 2.

Figure 2: Tooth cross section obtained by mechanical slicing.

In order to use the Tooth Drilling simulator, we have to load a volume representing a tooth using the EIKONA3D kernel. The result is shown in Figure 3.

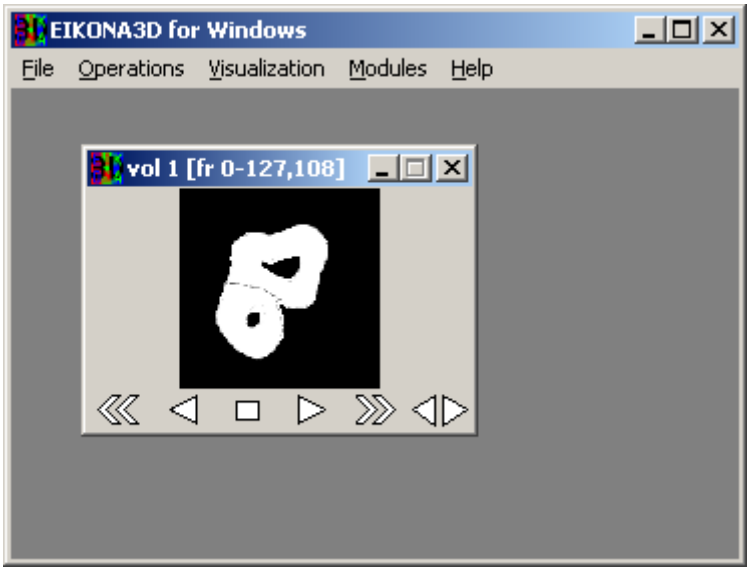


Figure 3: View of EIKONA3D after having opened a file.

2. Loading the *Tooth Drilling* module and 3D Representation

After having opened the tooth volume, we can visualize it. From the main menu of EIKONA3D we choose *Modules>Tooth drilling*. Then we choose the tooth volume to be visualized in 3D before starting drilling and we press the button *OK* (Figure 4). The 3D object rendering appears in a separate window (named *Drill*), as can be shown in Figure 5. Also, a new menu, named *Drill menu*, appears on screen to assist the dentist/user in the virtual tooth drilling.

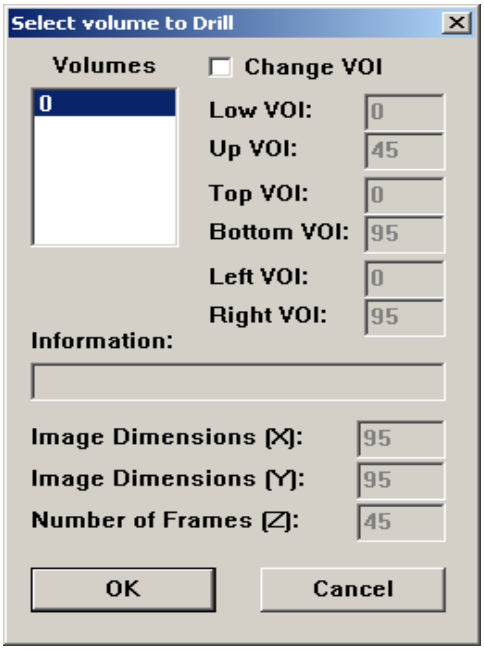


Figure 4: Selecting the tooth volume for visualization.

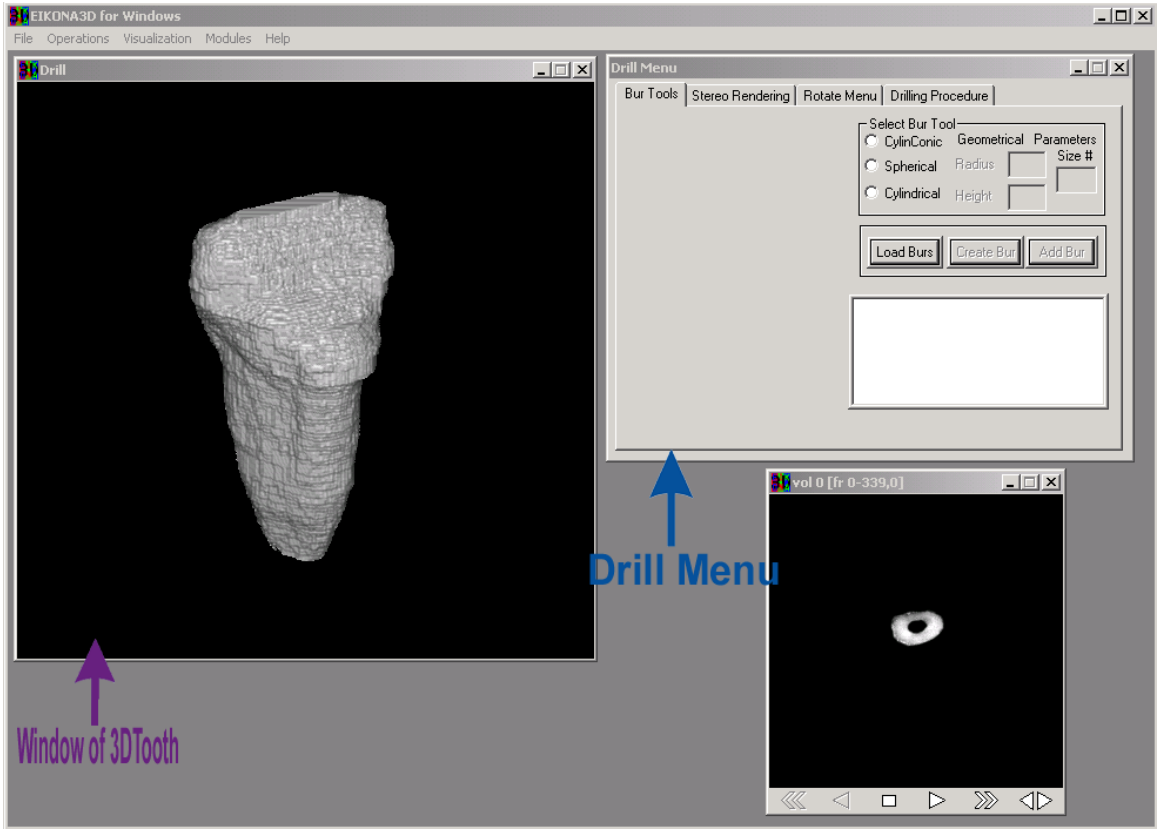


Figure 5: Volumetric 3D tooth rendering before drilling.

3. Working with the *Drill Menu*

The Drill Menu consists of four tabs: B*ur Tools*, *Stereo Rendering*, *Rotate Menu* and *Drilling Procedure*. To move from one tab to another one, just click on the name of the tab and the menu of this tab will be displayed automatically. The functions of each tab are described below:

3.1 Bur Tools tab menu:

It is the most important tab of the *Drill Menu*. This menu can be used by the dentist/user to create a virtual drilling bur tool or select one from those that already exist. Drilling is impossible without previously having chosen a bur tool. *Bur Tools* tab menu has four subregions (Figure 6).

The first one, called *Create Bur Tool*, has a double role. Firstly, it is used to create a virtual drilling bur tool and determine its shape parameters. Secondly, every time a selection is made from the tools that already exist, the characteristics of this tool are presented in this region.

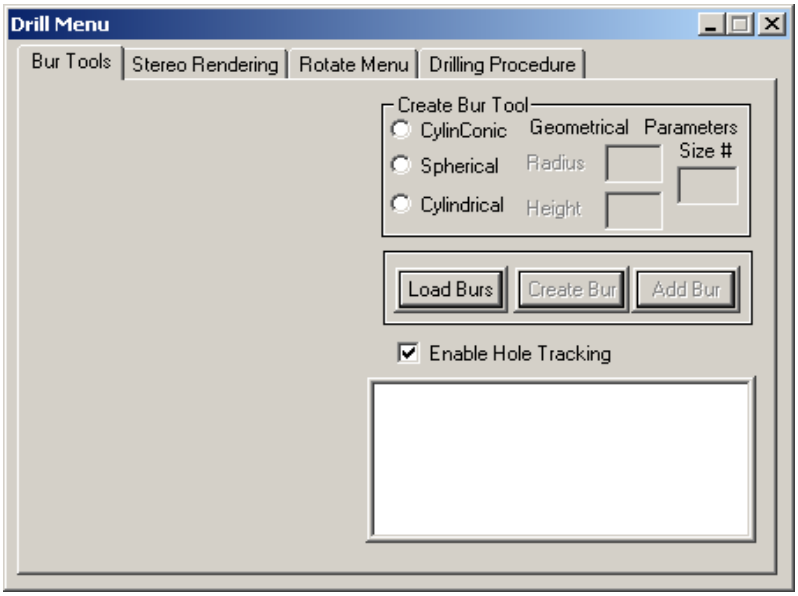


Figure 6: The *Bur Tools* tab menu.

The shape characteristics to be chosen define the shape of the tool, namely its radius and height (both measured in voxels). The radius and height can be transformed in length units (e.g. mm) if we assume that the voxel is cubic and know its size in mm. They are shown in Figure 7.

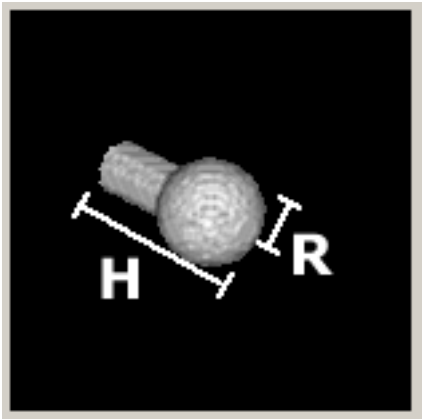


Figure 7: Shape parameters of a virtual drilling bur tool (H: height, R: radius).

There are three choices of the bur shape: *spherical* (only its radius has to be chosen), *cylindrical* or *cylinder-conical.* In the last two cases, both the *Radius* and *Height* must be defined. Furthermore, there is another edit box, named *Size #*, which is the size number of existing commercial burs that appear in commercial catalogues, e.g. in that of Dentsply [1]. The usage of this box is optional and has no effect to the drilling procedure.

The second region has three buttons named *Load Burs*, *Create Bur* and *Add Bur*. The *Load Burs* button is used to load the file that contains a list of tool burs selected from the commercial catalogue of Dentsply. The *Create bur* button is used to create the tool and activate the drilling process. The *Add bur* button is inactivated in the current version of Tooth Drilling modulator, but the intention is to be used for adding new created bur tools to the file that contains the existing bur tools.

The third region consists of a window that shows the characteristics of the bur tools that have been loaded from the file that was opened by *Load Burs* button. The information's shown are the Dental Company name, the type of shape (*radius*, *cylindrical* or *cylindrical-conical*), the *Size* number and the *Radius*. Every time the user selects a tool from this list with the mouse, the respective characteristics appear in the *Create Bur Tool* region.

Finally, the last region is where we can see an image of the created bur tool. Initially, this region is blank and is activated only after the *Create bur* button is pressed.

3.2 Stereo Rendering tab menu:

Stereo Rendering tab menu has options about the 3D stereo visualization of the tooth in the *Drill* window (Figure 8).

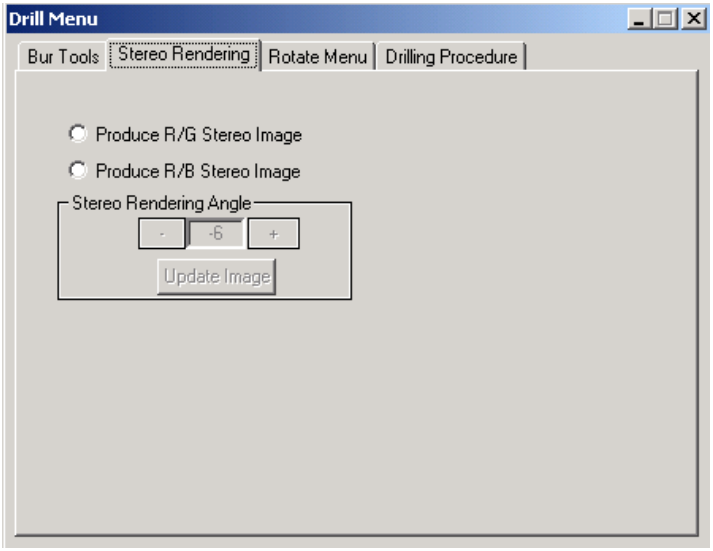


Figure 8: The *Stereo Rendering* tab menu.

In order to view the 3D stereo image, it is necessary to have a pair of 3D stereo glasses, either red/green or red/blue ones. The stereo image, actually, consists of two

same images, one with red color and the other one either with green or blue color, depending on what type of stereo glasses we use. These two images are projected at different angle creating the illusion of the 3D stereo image.

The options of *Stereo Rendering* tab menu allow firstly the choice about the mode of the stereo image to be produced (if it will be red/green or red/blue) and, secondly, the choice of the projection angle between the two stereo images. Changing the angle between the two image projections may improve the visual quality of the stereo image. Every time the stereo rendering angle changes, we have to press the *Update Image* button to activate this change.

3.3 Rotate Menu tab menu:

Using the *Rotate Menu*, we can rotate the 3D tooth volume image that exists in the *Drill* window (Figure 9), by changing the projection angle.

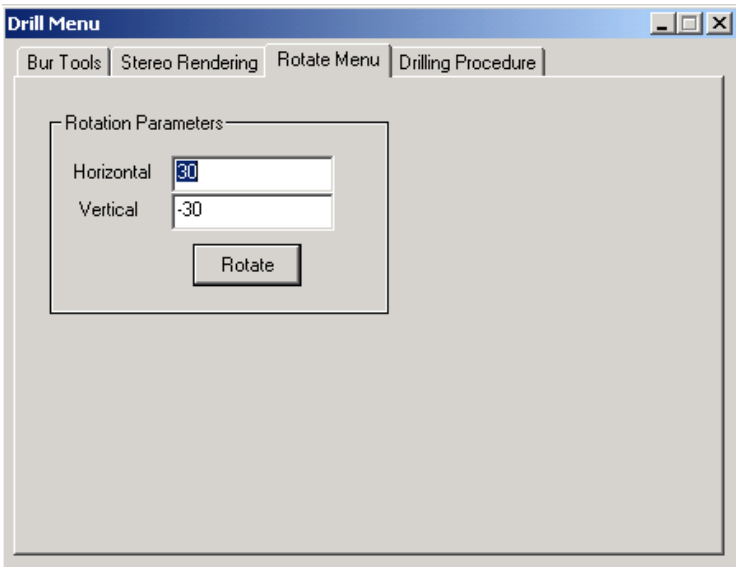


Figure 9: The *Rotate Menu* tab menu.

There are two editing boxes in *Rotate Menu*, named *Horizontal* and *Vertical* and the button *Rotate*. Changing the *Horizontal* angle (in degrees), the 3D tooth representation will be rotated horizontally. Changing the *Vertical* angle (in degrees), the 3D tooth representation will be rotated vertically. As in the *Stereo Rendering* tab menu, in to order to activate the changes in the perspective angle of the projection, the user has to press the *Rotate* button. By default, *Horizontal* and *Vertical* boxes have the values 30 and –30 degrees respectively, which are the values of the angles that are used from the *Tooth Drilling* module, in order to project the tooth volume.

4. The *Drill* Window and the tooth rotation operation

The *Drill* window, is used to visualize the virtual 3D tooth volume. The tooth slice sequence and its virtual 3D visualization in the Drill window are shown in Figure 10.

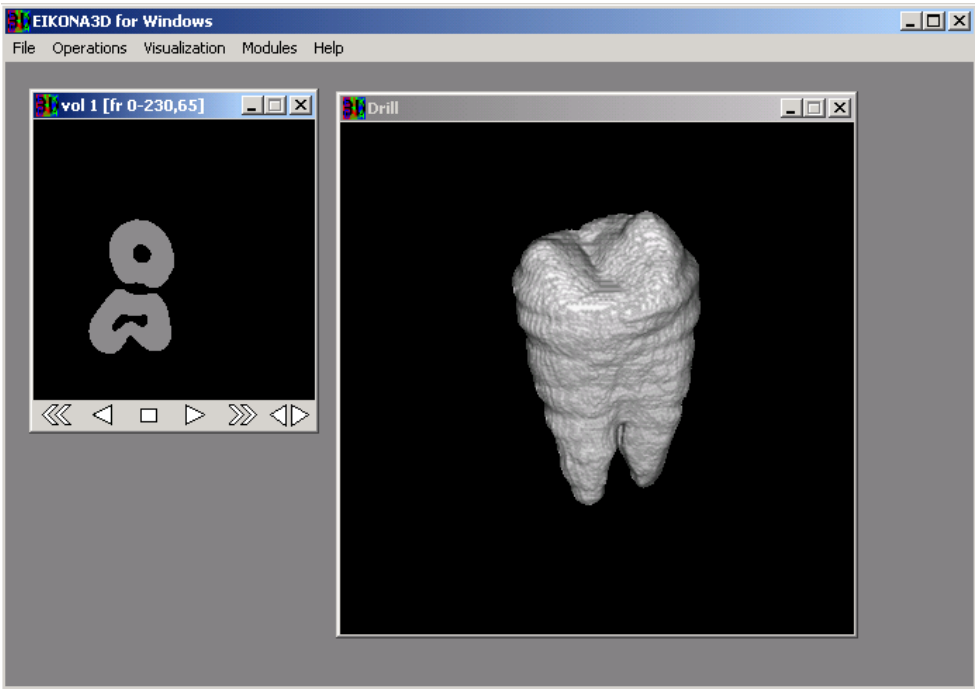


Figure 10: A tooth image sequence and its visualization in the Drill window.

There are two operations that can be done in the *Drill* window. First of all, it is the virtual tooth drilling operation that will be described in the next section and, secondly, it is the 3D tooth rotation using the mouse. The tooth will rotate by moving the mouse to the right or left, while keeping the right mouse button pressed. Then the 3D tooth will be visualized from a different angle. The direction of the mouse movement will correspond to the direction of rotation of the 3D tooth. When the tooth is brought in a chosen position, we can start drilling.

Alternatively, the 3D rotation of the virtual tooth can be achieved from the *Rotate Menu* tab of the *Drill Menu*, according to the following steps (Figure 11):

1. Enter a value for the Horizontal rotation angle in degrees (e.g. 220)
2. Enter a value for the Vertical rotation angle in degrees (e.g. 235)
3. Press the *Rotate* button.

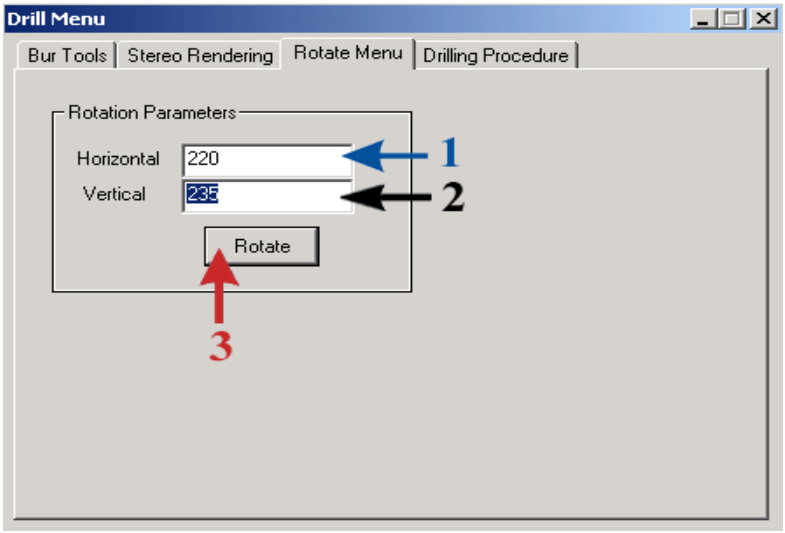
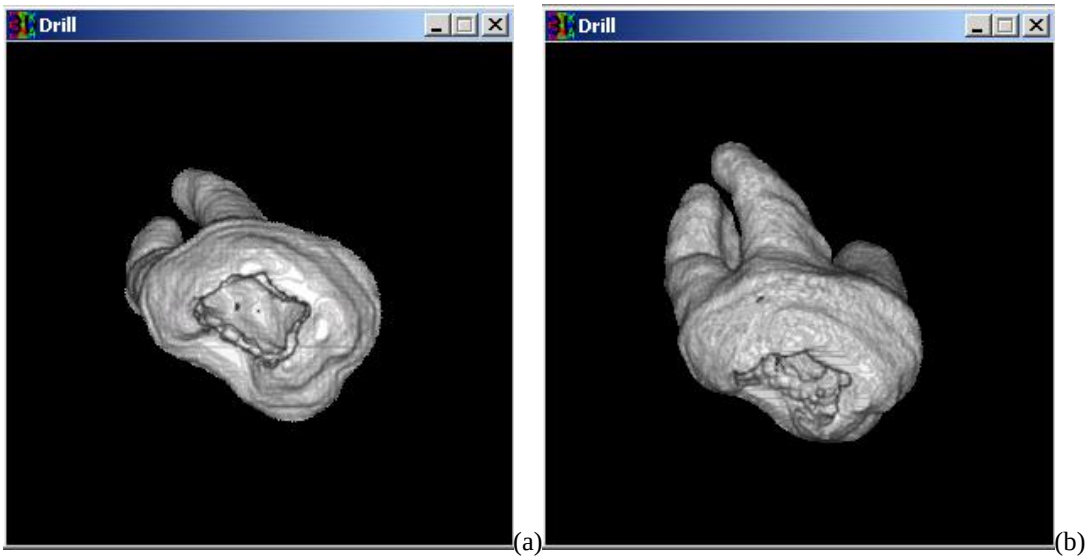


Figure 11: Three steps for the rotation operation.

The *Drill* window before and after the 3D tooth rotation operation is shown in Figure 12.



(a) (b)

Figure 12 (a) Tooth before rotation, (b) Tooth after rotation.

5. Drilling Bur tool selection and its use in drilling

To start drilling, we have to choose first a bur tool. There are two possibilities: either a) to create a virtual bur tool and determine its parameters or b) to load one corresponding to a commercial bur from a bur parameter file. If we follow the first way, we have to do the steps described below (Figure 13):

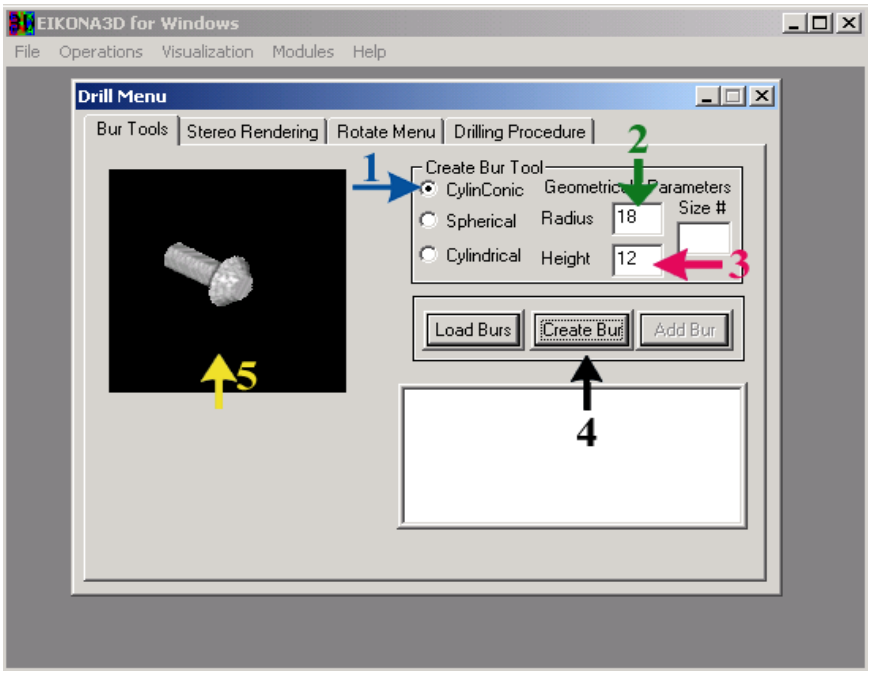


Figure 13: Creating a cylindrical-conical bur tool.

1. Choose the type of tool you want to use among a *Cylindrical - Conical*, *Spherical* or *Cylindrical* tool (for example *Cylindrical - Conical*).
2. Enter a value for the *Radius* parameter of the tool (for example 18 voxels),
3. Enter a value for the *Height* parameter of the tool (for example 12 voxels),
4. Press the *Create Bur* button
5. The created tool is displayed in the left side of the *Drill Menu*.

The visualization of the drilling tool will give the user an idea about the effect of the drilling tool on the virtual 3D tooth to be drilled.

For a spherical tool type the only parameter that must be determined is the radius. For the *cylindrical* and *cylindrical–conical* type, we have to define both the *Radius* and *Height* parameter. If, for any reason, we give a zero value to either *Radius* or *Height* parameter, the program will complain.

If we follow the second way, the appropriate steps to create a virtual bur tool are shown in Figure 14.

a) Press the *Load Burs* button

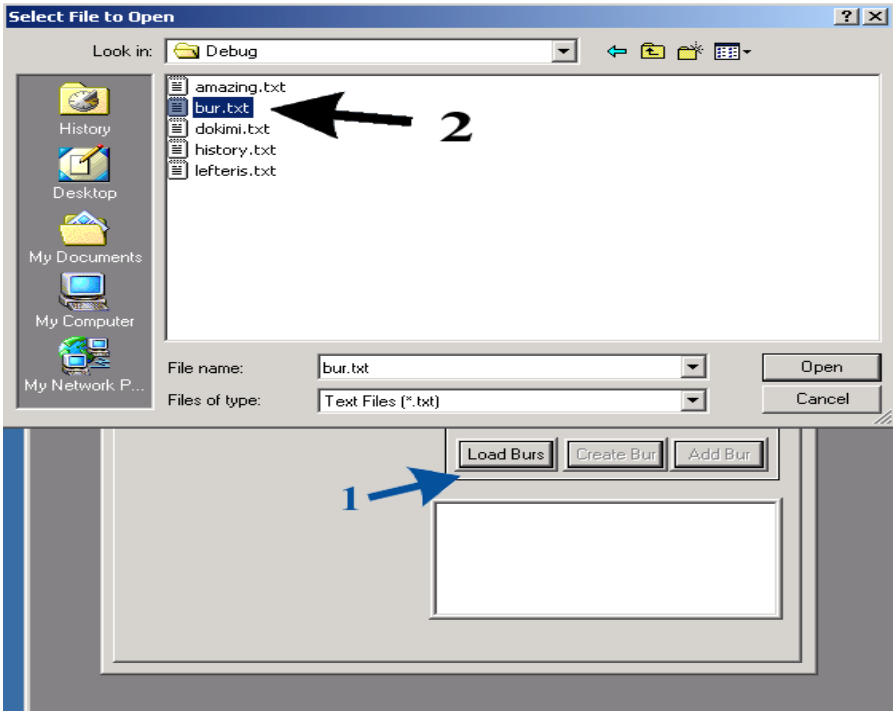


Figure 14: Opening the `bur.txt` file.

b) Find and select the file `bur.txt`. This file has a variety of parameters for virtual bur tools that correspond to the ones existing in the catalogue of Dentsply. The use of any other file is incorrect and an attempt to load another file than `bur.txt` will result in a message error. You will find the file `bur.txt` in the directory of EIKONA3D.

c) Select the appropriate tool from the displayed tool list. The parameters of the bur will appear automatically at the *Create Bur Tool* region. As you can see, the height parameter is not determined. Also, the radius parameter in the file is in mm, where at the *Create Bur Tool*, has been converted in pixels multiplying the value in mm by 10. So, enter a value for the *Height* parameter having in mind this conversion.

d) Press the *Create Bur* button. In the left side of *Drill* menu the created tool will be displayed.

The software keeps in memory one selected set of geometrical parameters for each geometrical bur shape. Each time the respective tool is recalled, the geometrical parameters appear in their corresponding slots. The user must press the *OK* button in order to display the selected tool and to confirm the chosen parameters. The currently used tool is the one displayed in the left part of the window.

Now that the bur tool has been created, we are ready to proceed with virtual tooth drilling. We apply drilling in the 3D tooth volume that is displayed in the *Drill* window. The effects of the drilling can be seen both in the *Drill* window and in the window containing the slice sequence of the tooth (Figure 15).

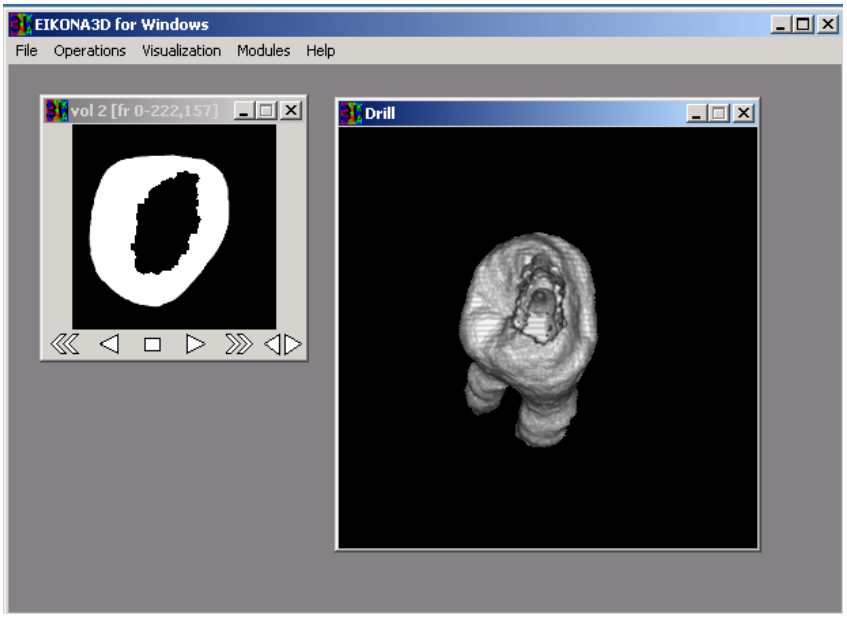


Figure 15: Display of a drilled tooth.

The drilling process is quite simple. The user/dentist applies the bur tool to the appropriate surface point of the virtual 3D tooth by clicking on it with the left button of the mouse. Each time the user presses the left mouse button, a hole having same shape to that of the bur tool is created in the tooth volume. For continued drilling, the user must press the left mouse button repeatedly. Furthermore, by keeping the left mouse button pressed and moving the mouse at the same time, the mouse will drill along its motion trajectory. Whenever the user desires, he/she can rotate the object and visualize the drilling results from a desired angle, as described in the previous section. The drilling results can be seen in Figure 15.

If the bur tool accidentally reaches the root canal, a red square appears in the *Drill* window at the surface point where the root canal is reached. Also, a warning message "*You reached the root canal. If you wish Undo"* is shown at the lower left side of *Drill Menu*, as can be seen in Figure 16.

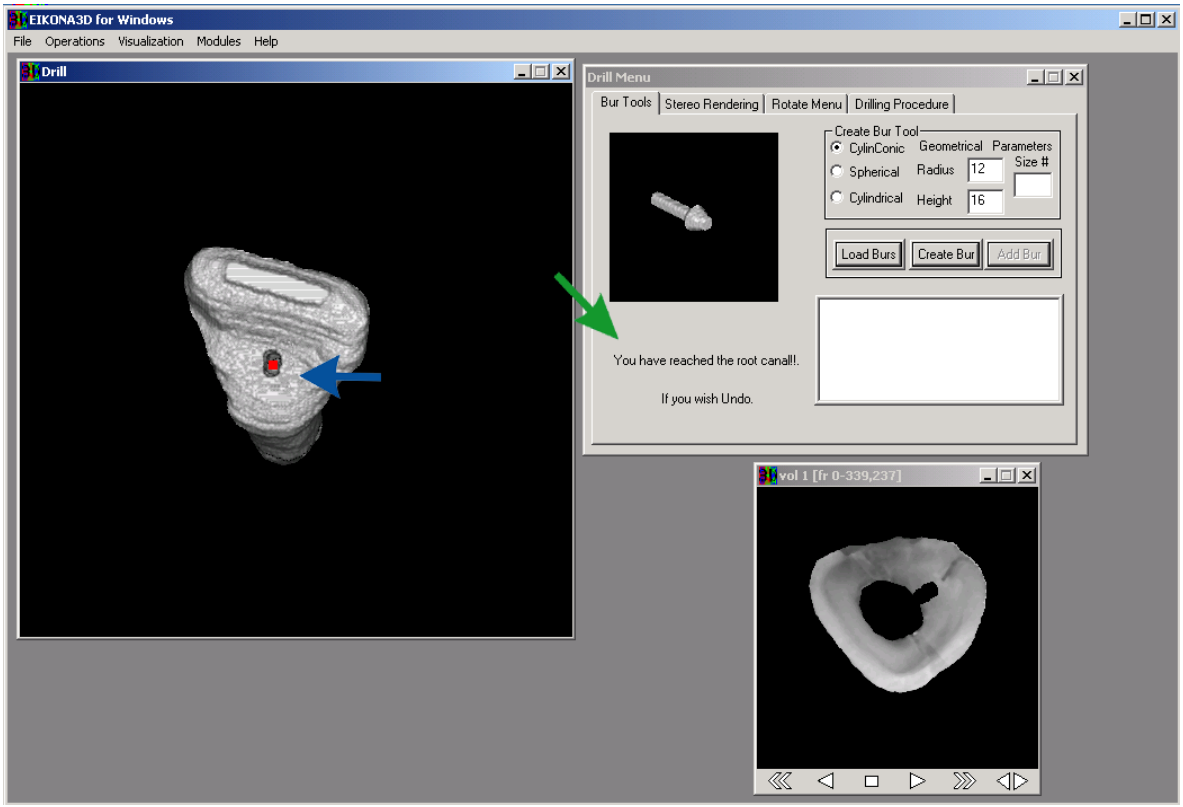


Figure 16: Reaching the root canal.

6. 3D stereo tooth display

If 3D stereo glasses are available, then we can view the 3D stereo tooth image. In order to produce and view the stereo image, we can go to the *Stereo Rendering* tab menu and follow the steps below (Figure 17):

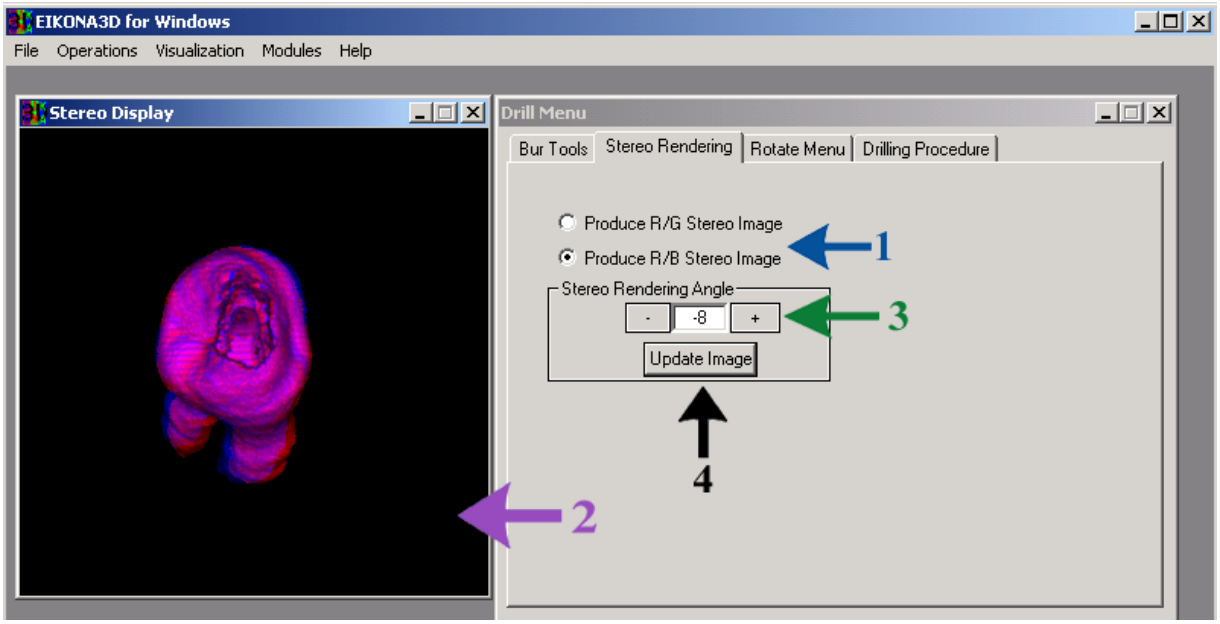


Figure 17: Producing a stereo image.

1. Select the preferable type of stereo image to be produced (red/green or red/blue).

2. The stereo image will be displayed in new window named *Stereo Display*.

3. The quality of stereo image can be improved by changing the stereo rendering angle by pressing the [-] and [+] buttons.

4. Having selected the proper stereo angle, press the *Update Image* button and the new stereo image will be seen in the *Stereo Display* window.

Stereo image is not updated immediately. So, every time we make changes to *Drill* window we have to produce a new stereo image. The stereo image cannot be saved.

7. Drilling Procedure tab menu

As we can see in Figure 18 the *Drilling Procedure* tab menu consists of two regions: the *Job Menu* and the *Undo Menu* region.

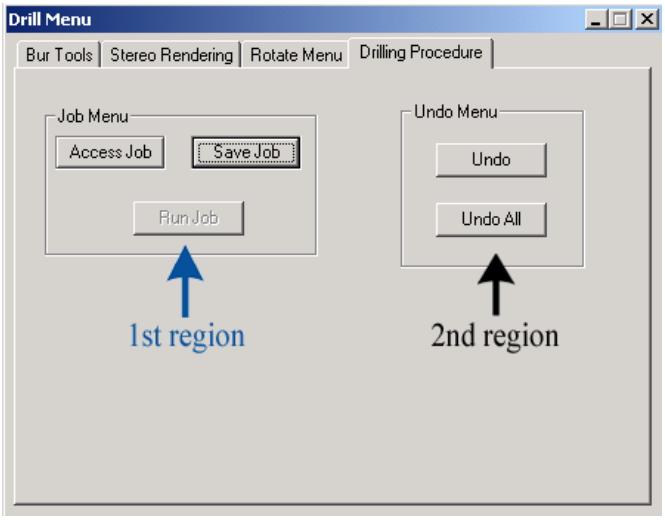


Figure 18: The *Drilling Procedure* tab menu.

The drilling steps in each virtual tooth drilling form a drilling job. The *Job Menu* region is responsible for saving a tooth drilling job or loading a saved job. It has three buttons, named *Access Job* - *Save Job* and *Run Job*, simulating the actions mentioned above. The *Save Job* button saves the present job as it is. The save operation saves the required parameters of the drilling job in a text file. This file must be opened using the *Access Job* button, in order to load a saved drilling job. To run a past drilling job, we have to load it first using the *Access Job* button. Furthermore, to run the drilling job, the initial tooth volume must be already opened and visualized in the *Drill* window.

The second region, namely the *Undo Menu*, is responsible for the undo action, i.e. for undoing a wrongly executed drilling job. It has two buttons, the *Undo* and the *Undo*

All button. When, for any reason, it is desired to undo only the last drilling step, the user can press the *Undo* button. Otherwise, if the user wants to delete the whole drilling job, as it has been carried out so far, then he has to press the *Undo All* button. The undo operations have effect only in the present tooth volume and the results appear immediately in the tooth display in the *Drill* window.

If, for any reason, you wish to undo the last drilling action or the whole drilling process then you have to move to the *Drilling Procedure* tab menu and press the *Undo* or *Undo All* button. The effect of undo action will appear immediately in the *Drill* window, as can be seen in Figures 19 and 20.

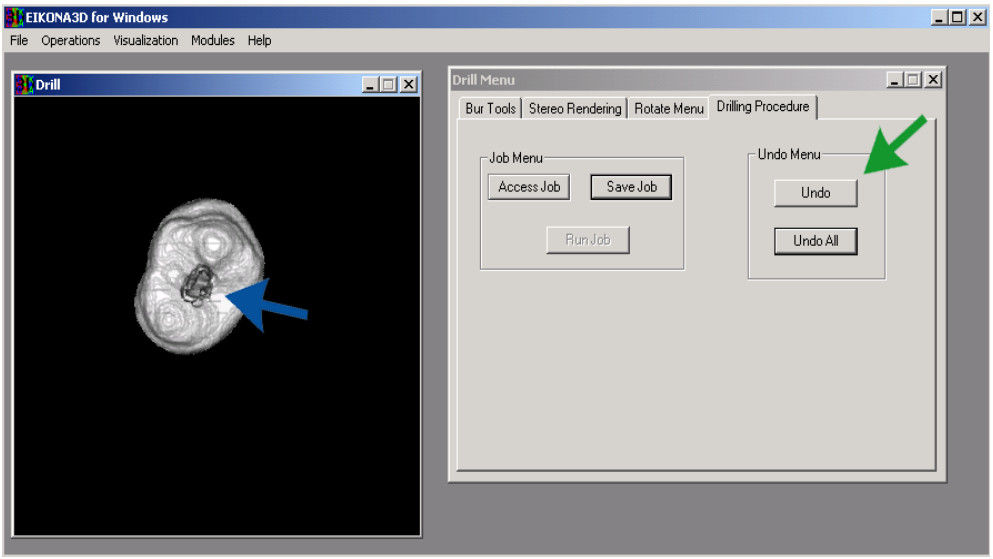


Figure 19: Snapshot before the undo action.

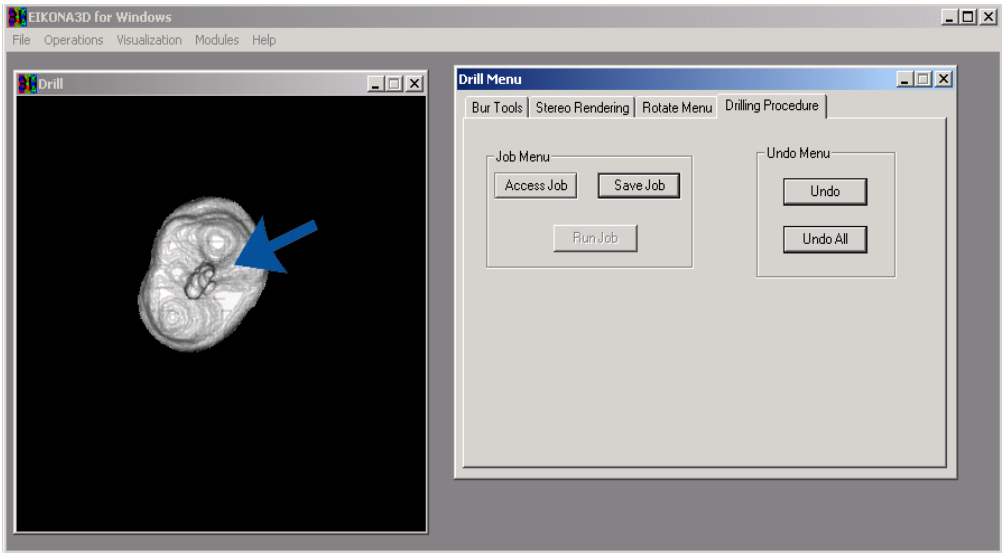


Figure 20: Snapshot after having pressed twice the *Undo* button.

8. Saving the drilled tooth volume

Having completed the drilling job, the user may want to save not only the job (from *Drilling Procedure* tab menu), but the entire drilled tooth volume as well by following the steps:

1. From EIKONA3D menu, choose *Modules>Create\Open\Save Volume>Save a Tooth Volume*.
2. Select the volume you want to save and press the *OK* button.

9. References

[1] http://www.dentsply.com