Reconstructing Solid Model from 2D Scanned Images of Biological Organs for Finite Element Simulation

This work presents a methodology to reconstruct 3D biological organs from image sequences or other scan data using readily available free softwares with the final goal of using the organs (3D solids) for finite element analysis. The methodology deals with issues such as segmentation, conversion to polygonal surface meshes, and finally conversion of these meshes to 3D solids. The user is able to control the detail or the level of complexity of the solid constructed. The methodology is illustrated using 3D reconstruction of a porcine liver as an example. Finally, the reconstructed liver is imported into the commercial software ANSYS, and together with a cyst inside the liver, a nonlinear analysis performed. The results confirm that the methodology can be used for obtaining 3D geometry of biological organs. The results also demonstrate that the geometry obtained by following this methodology can be used for the nonlinear finite element analysis of organs. The methodology (or the procedure) would be of use in surgery planning and surgery simulation since both of these extensively use finite elements for numerical simulations and it is better if these simulations are carried out on patient specific organ geometries. Instead of following the present methodology, it would cost a lot to buy a commercial software which can reconstruct 3D biological organs from scanned image sequences.

Contents

more

Introduction

Anatomical dimensions and geometry of body organs differ from patient to patient due to various factors such as age, body size and weight and due to the presence of pathologies such as cysts and cancer. Scanning procedures like Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) are often used in getting patient specific organ details including dimensions, and they can identify pathological conditions, if any. Constructing a 3D model of an organ of interest from these images would not only result in better visualization of the organ and its pathologies, but also, more importantly, would benefit patient specific surgery planning and surgery simulation. Accurate surgery planning and surgery simulations both require the use of numerical techniques like finite element analysis and the actual organ

geometry is the first requirement for any analysis to be really useful.

The main aim of the present paper is to describe a procedure to get patient specific organ geometry from patient specific scanned images. Also, in this paper, it is demonstrated that the geometry thus obtained can readily be used for finite element analysis. Hence the paper is organized as follows. Section 2 describes the procurement and installation of the necessary softwares. The features which were found to be most useful for the present work have also been taken note of here. Section 3 gives the general procedure to get a 3D geometry from scanned images. Section 4 illustrates this procedure using the 3D reconstruction of a liver as an example. Section 5 uses the liver geometry thus obtained for a nonlinear analysis, confirming that a finite element analysis may be performed on patient specific organ geometries obtained using the procedure explained in this work.

This paper is an extended and enlarged version of [1]. The present work gives very detailed information (like the kind of information one can get in a manual, e.g., details of every mouse click) on the 3D reconstruction using a few softwares, while [1] does not go into such a level of detail. Also, the present work, when compared to [1], gives more information on the softwares used, including the installation details of the softwares under both Linux and Windows, whenever found necessary. While [1] does not, the present work gives information on the hardware configuration and the operating systems used. Also, the present paper takes note of, analyzes, and solves the different kinds of problems encountered while employing the methodology (or the procedure) given in the present work. While [1] uses only free softwares for 3D reconstruction, present work optionally makes use of MATLAB (which is a commercial software) also. The finite element analysis that is carried out in the present paper is a nonlinear one, while just a linear elastostatic analysis was carried out in [1]. These are the additional works which make the present paper an extended and highly enlarged version of [1] which is a conference paper.

Software Details

There exit a few commercial softwares which can easily do 3D reconstruction from 2D image sequences (e.g., 3D Doctor [2], AMIRA® [3]). But, the procedure presented in this paper mostly makes use of free softwares. Following this procedure (outlined in Section 3) results in a 3D solid which represents the organ of interest. Next, in Section 5, a finite element analysis is performed on the solid organ obtained by following the procedure explained in Section 3.

The free softwares used are: ImageJ, ITK-SNAP, MeshLab. Commercial softwares used are: MATLAB, Rhinoceros, ANSYS. In the following sections, some of these (mostly free softwares) are explained in some detail. Here, only those salient features which have been found to be of great use in the present work have been looked into. The details are by no means exhaustive and one has to consult the respective software documentations for further details.

At this point, it is informative to note that the 3D reconstruction involves two phases. During the first phase, one starts with scanned image sequences and finally gets a ***3D surface***. All the free softwares listed above are used to complete this phase only and only these free softwares are sufficient for this task. MATLAB is optional and may be used in this stage of the work, just to cross verify the image information. ***Also, it should be noted that many commercial softwares like 3D Doctor just do this first phase of the work only***. The second phase of the 3D reconstruction process is the ***conversion of 3D surface to 3D solid***. Present work makes use of the commercial software Rhinoceros (free but fully functional evaluation version available for download) for this purpose. The role of the commercial software ANSYS in this work is just to show that a nonlinear finite element analysis can be performed on a geometry obtained by using the rest of the softwares.

Another point to be noted here is that it is better to run image processing softwares like ImageJ, ITK-SNAP and MeshLab under Linux because of its stability while running memory intensive tasks. Both

Linux and Windows versions of all these three softwares are available for download and authors have tried running these softwares under both Linux (Fedora 9, 64 bit) and Windows XP Professional SP2 (32 bit). Authors have found that, although installation might be somewhat tedious sometimes because of the need for manual compilation of the source code or the need to resolve and install for dependencies, it is worth spending some time installing these softwares under Linux, since it was found to make a robust environment for processing large image sequences. However, because of ease of installation, Windows versions can be used and they work fine as long as the problem size is not too large or the processing task does not require too much memory. As far as the present work is concerned, both ImageJ and MeshLab could be used both under Linux and Windows without any observable difference in the performance. But, the tasks requiring ITK-SNAP could be completed only under Linux.

The commercial software Rhinoceros is available only for Windows, and ANSYS, although Linux version is also available, was used under Windows only. MATLAB was used under both Linux and Windows.

The hardware configuration of the computer used for this work is: AMD Athlon 64 X2 Dual Core Processor 5200+, 2.61 GHz, 8 GB RAM. However, although Linux could use all of the available RAM, Windows could detect only 2.75 GB of RAM. This might be the reason why ITK-SNAP had trouble while executing a memory intensive task like semi-automatic segmentation under Windows.

The softwares used in the present work are explained in some detail now.

MATLAB

MATLAB [4] is a widely available commercial software. MATLAB 7.9.0 (R2009b) has been used in the present work. Image processing toolbox in MATLAB can read images in different formats and display those images on a window. The most useful functions from this toolbox are: ***imread, imtool, whos***. One can also directly load the image file into the MATLAB workspace using ***File → Open***. This will display some of the image information in the workspace window, like the dimensions of the array which represents the image, number of bits per pixel (also whether signed or unsigned). From this array data one can infer the number of pixels in each direction of each image and one can infer the total number of images contained in the image sequence represented by the input file.

For example, to view the image information for a file 'liver.tif', one can use ***File → Open***, then browse to the file location, select ***All Files*** in the ***Files of type*** drop down box, press the ***Open*** push button, press ***Finish*** push button in the ***Import Wizard***. This creates a variable named 'liver' in the workspace and the variable information may be read from the workspace window.

MATLAB is optional and usually it is possible to obtain the image file information from the source which supplies the file.

ImageJ

ImageJ [5][6][7] is an image processing software. This software is not subject to copyright protection and is in the public domain. It is written in Java and hence may be installed on any computer with pre-installed Java. Also, ImageJ bundled with Java is available for different platforms like Linux (32 bit), Linux (64 bit), Windows (32 bit), Windows (64 bit).

Present work uses ImageJ 1.42q (together with Java 1.6.0). Its role is to import an image stack from the file, getting image information, selecting the image portions with a rectangle and processing the selected portion only, removing selected portion of the image, subtracting a particular intensity value from each

pixel in the image, multiplying each pixel intensity value by a particular number, applying the image processing procedures to a single image or to the whole image stack, reversing the order of images in an image stack, concatenating image stacks, saving the processed image stack in different file formats etc. One has to consult the software documentation for a detailed description of each of these features.

ITK-SNAP

ITK-SNAP [8][9] is a software application used to segment (*image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics*) structures in 3D medical images. It provides semi-automatic segmentation using active contour methods, as well as manual delineation and image navigation. It can export a segmented image sequence as a 3D surface mesh. ITK-SNAP is a free software, provided under the General Public License [10] and as per the license, the source code and binaries are provided free of charge for academic or commercial use. ITK-SNAP 2.0.0 is used in the present work.

ITK-SNAP is capable of importing image stacks of different file formats. The present work uses the software for 3D segmentation. Through a rectangular box, one can select the region of interest and the software does the segmentation within this region of interest. The region of interest may be selected on each image in the stack separately and each image in the stack can be segmented separately, or, the region of interest may be selected for the whole image stack at once by making the rectangular selection include the region of interest of all the images in the stack so that the 3D segmentation can be done on all the images in the stack at once.

The software can do manual and semi-automatic segmentation. Many options are available and a lot of theory is there behind the semi-automatic segmentation. These theories and options are not explained in this paper but understanding these theories helps in the understanding of the algorithms behind different features, and it may help to select a proper option. A detailed understanding of all of these theories is not essential however, if ones aim is just to get a correct 3D segmentation; one can try out different options, and by trial and error can arrive at the correct segmentation.

To be short, automatic (may also be called semi-automatic) segmentation first involves the selection of the region of interest. The region of interest applies to all the images in the stack. Snake initialization is the next step. Snake is a closed curve or collection of closed curves in 2D, and a closed surface or collection of closed surfaces in 3D. Snake can be initialized using a circular bubble and more than one bubble can be used to initialize the snake. The snake is in fact the initial approximation for the segmented volume or the volume of interest. In the next step, the snake is made to evolve or the initial segmentation represented by a spherical bubble/bubbles changes its shape towards the shape of the final segmented volume which represents the surface of the biological organ to be extracted from the image stack. This morphing of the snake is done in steps and is handled by different algorithms. One algorithm modifies the shape of the snake taking into account intensity variations across different pixel locations in the images and the other algorithm uses the edges in the images to calculate the morphing. The iterations (and hence morphing) should continue until the snake represents the surface of the organ of interest. Of course, different options are available to control the snake evolution. But, since ImageJ is used in this work to preprocess the input for ITK-SNAP, it does not matter whether one chooses the intensity based algorithm or the image edge based algorithm to drive the snake evolution. In fact, preprocessing with ImageJ keeps in the images only the organ of interest and hence it does not matter which algorithm one uses to drive snake evolution, and also it was found that the results are practically the same for different parameter values one has the provision to choose from. The final step in using ITK-SNAP is to export the 3D segmented volume as a 3D surface. Thus, in this work, although ImageJ itself virtually (***it was not used for the actual segmentation in this work***) does all the segmentation work, ITK-SNAP is a must since the images are not to scale in ImageJ and it does not have the provision to export the image stack as a 3D surface.

MeshLab

MeshLab [11][12] is an open source system for the processing and editing of unstructured 3D triangular meshes. It is a tool developed with the support of the ***3D-CoForm*** project. The system is aimed to help the processing of the typical not-so-small unstructured models arising in 3D scanning, providing a set of tools for editing, cleaning, healing, inspecting, rendering and converting this kind of meshes. It is licensed under the GNU General Public License. MeshLab v1.2.2 has been used in the present work. MeshLab installer is available for Windows. But only source is available for Linux. One needs a C++ compiling environment and a software called qmake for compiling the source code. Also, the following libraries are essential: Qt 4.4 (if one wants to install MeshLab v1.2.2), a svn-client, the VCG libraries. These softwares should also be installed for external dependencies: glew, lib3ds-1.3.0, bzip2-1.0.3, muParser 1.30. All these softwares should be installed (together with their own dependencies if any) before starting the MeshLab compilation. Then, one should follow the compilation instructions in the installation documentation to complete the MeshLab installation. It takes some effort to install the software under Linux. But it may be noted that authors have been able to process files of around 85 MB size without any problem under Windows itself. Some of the numerous useful features available in the software are explained next.

Edit → Fill Hole checks whether the 3D surface is water tight. MeshLab should report that there are no holes in the mesh to edit.

Filters → Remeshing, simplification and reconstruction → Quadric Edge Collapse Decimation is used to control the level of detail of the 3D surface by reducing the total number of 3D triangular mesh faces to the target number of faces. Target number of faces should be lesser than the original number of faces and hence this only reduces the level of detail of the model. This reduces the mesh using a quadric based edge collapse strategy. More number of faces means more level of detail but many a times too much detail is unnecessary. The big problem in using too many faces to describe a surface arises if one wants use the 3d watertight surface for finite element meshing. As the total number of faces increases, the total number of elements required to adequately mesh the surface and the 3d solid also increases. Also, finite element analysis requires all elements to be well shaped to give accurate results and hence the number of ***well shaped*** elements required to mesh a very detailed solid represented by too many number of faces could be very high. Further, if the analysis is a nonlinear one, elements should be strictly too well shaped because elements could distort a lot during a large deformation analysis. Poorly shaped triangles in the surface mesh result in more number of elements and sometimes it may not be possible to mesh the solid at all. It is next to impossible to ensure that all the triangles forming the surface are well shaped (even with the ***Triangle quality*** menu explained in the next paragraph), if the total number of faces is too large. All these result in very large number of elements in a good finite element mesh that is suitable for large deformation analysis, if the solid to be meshed is represented by too many numbers of surface triangles. If the total number of elements goes too high, it may not be possible to do the finite element analysis because of memory limitations. Finally, the optimum number of faces to describe the organ of interest has to be decided upon considering all these factors: the level of detail that is essential for a particular purpose, whether the analysis is linear or nonlinear, whether the solid is homogeneous or heterogeneous, the system memory available. ***Using the present menu, MeshLab can convert the geometry described by very large number of triangle faces which is usually the case, to a geometry which is a very good approximation to the original one but described with lesser number of faces.***

Filters → Color Creation and Processing → Triangle quality displays the quality measures for each of the triangle surfaces using a color code. The triangles which are nearly equilateral (well shaped triangles) are coded using colors from the violet side and triangles with very small or very large angles (bad shaped or distorted triangles) are coded using colors from the red end of the spectrum of visible colors. One can click on the ***Wireframe*** icon to switch to the wireframe display of the surface to see the triangles more clearly. One should process the mesh using the below mentioned ***Smoothing, Fairing and Deformation*** filters until all the triangles forming the mesh become well shaped. This is very important if the mesh is to be used for any finite element analysis later. This is because the creation of 3D finite element meshes

requires a well shaped surface mesh in the first place; 3D meshing algorithms usually mesh the 3D surface made of the surface of the 3D solid first, the inside volume of the solid being meshed with solid elements later. Analysis results may go inaccurate or if the surface triangle shapes are too bad, sometimes it may not be possible to create a 3D finite element mesh at all. The present menu item ***Triangle quality*** should be executed after each application of any ***Smoothing, Fairing and Deformation*** filters explained next to observe the improvement in triangle quality upon the application of the said filters.

Filters → Smoothing, Fairing and Deformation → MLS projection is used to improve the triangle quality of surface triangles. It accomplishes this by projecting the whole mesh onto the MLS surface defined by itself [13].

Filters → Smoothing, Fairing and Deformation → Laplacian Smooth is another filter used for improving triangle quality. Here, the smoothing is accomplished by calculating the average position for each vertex with nearest vertex.

Filters → Smoothing, Fairing and Deformation → Taubin Smooth is also found to be very useful and it makes two steps of smoothing, forth and back, for each iteration. All the three ***Smoothing, Fairing and Deformation*** filters explained here may slightly change the geometry being processed. Care should be taken to ensure that these filters do not change the geometry too much. If the shape is retained but if it is found that the geometry got scaled, one has to rescale the whole geometry using known distance between known features (like distance between visibly identifiable features like sharp corners, distance between consecutive image slices etc.).

Rhinoceros

Rhinoceros [14] is a low cost, commercial, NURBS based software for windows. It can create, edit, analyze, document, render, animate, and translate NURBS curves, surfaces, and solids with no limits on complexity, degree, or size. It supports polygon meshes and point clouds also. The evaluation version can be downloaded free of cost. The evaluation version is fully functional but will ***save*** only 25 times. Version 3.0 has been used in the present work to convert the ***3D surface*** which represents the organ of interest to a ***3D solid***.

The command ***MeshToNurb*** is to be typed on the command prompt to convert the selected mesh entities on the screen to NURBS entities. Then one can save the resulting solid in any of the available 3D solid formats.

ANSYS

ANSYS [15] is a very popular engineering simulation software. It is a commercial software and is widely used as a standard tool for finite element analysis. ANSYS Multiphysics Release 10.0 has been used in the present work just to demonstrate that a finite element analysis is possible on the 3D geometry obtained making use of the softwares explained earlier. Very good documentation (including tutorials) is available with the software.

General Procedure for Extracting 3D Organs from 2D Image Slices

In general one has to perform these steps in sequence:

1. Open ImageJ, import the image file (e.g., .tif), process the image stack with the goal of keeping the pixels which belong to the organ of interest but with the goal of removing the pixels which do not belong to the organ of interest. This makes the next step (segmentation) much easier. Many a times, to reduce size, image files give only the half of the information (the other half of the organ may be considered to be symmetric to the first half). In that case, one can reverse the order of images in the image stack and then concatenate this image stack with the original image stack to get the full image stack. One should save the processed image stack in a file format which can be read by ITK-SNAP (e.g., .raw).
2. Open ITK-SNAP and open the file saved in the previous step. If the image file format does not include the image header information (like the total number of pixels in each direction, absolute distance between pixels in both the directions, total number of such images in the image stack considered, distance between the consecutive images in the stack, voxel representation scheme etc.), ITK-SNAP requests the user to enter this information. In this case, the information has to be obtained from the source which supplied the image file. Optionally, MATLAB may also be used to obtain some of the information. Then, ITK-SNAP displays the image stack. Next, one has to complete segmentation (manual or automatic). Then, the segmented volume has to be exported as a 3D surface mesh using the menu item ***Segmentation → Save as Mesh…*** in a file format supported by MeshLab (e.g., .stl).
3. Open MeshLab and open the file saved in the previous step. While opening, MeshLab asks whether it is all right to unify duplicate vertices and one can select ***OK***. Apply ***Edit → Fill Hole*** to ensure that the mesh does not contain any holes. If the surface mesh is not watertight or in other words if it contains any holes, it does not properly represent the 3D geometry of interest. Next, ***Quadric Edge Collapse Decimation*** filter is used to reduce the number of triangle faces to the target number of faces. ***Triangle Quality*** is used to color code the triangles to identify ill shaped triangles and the filters ***MLS projection***, ***Laplacian Smooth*** and ***Taubin Smooth*** are used to make those triangles well shaped. Finally, the processed mesh is saved.
4. Open Rhinoceros and ***Import*** the geometry saved in the previous step. Issue the command ***MeshToNurb***, select the whole mesh using a rectangular window, and press 'Enter'. Now use ***File → Save As…*** to save the 3D geometry in a format which can be read by ANSYS (e.g., .sat).
5. This step is optional and it just ensures that the 3D geometry obtained by following the previous steps can be used for an analysis supported by ANSYS. In this step, one considers the organ geometry obtained in the previous step as a 3D CAD model and follows the same analysis procedures as those followed for analyzing any CAD model.

An Example of the 3D Reconstruction of a Liver

Obtaining the Image File and its Details

The image file is obtained from [16]. The image file is a zipped .tif image stack which contains a pig liver volume in the undeformed state and it contains 147 image slices. The image stack has the name: 'LiverNoDeformation.tif' and has a size of around 37 MB. The image stack was obtained using CT-scan and one can refer [17] for the details on the procurement and processing of the liver used for the scan, the scanning equipments used, scanning procedure employed, and further details on the image stack compiled etc. The things which are of interest to us are: 512 X 512 pixels make up each image in the stack, the distance between pixels (in both directions) = 0.29 mm, distance between consecutive images in the stack = 1 mm, each pixel needs 8 bits of memory to describe it and each pixel is described by an 8 bit unsigned

integer, the images are gray scale images, since the images are gray scale images and since each pixel is represented by an 8 bit unsigned integer each pixel can have an intensity between 0 to 255 (totally 256 intensity levels).

Viewing Image Information in MATLAB

Open 'LiverNoDeformation.tif' and import the variable 'LiverNoDeformation' to the MATLAB workspace. MATLAB shows that the variable size is ***(512 X 512 X 147)*** and that the variable class is ***uint8*** (this stands for unsigned 8 bit integer). It also shows that 38535168 bytes of memory is needed to hold this variable. One can note that the information displayed here is consistent with the information noted down in the previous subsection.

Using ImageJ

Open ImageJ and use ***File → Import → Raw…*** to browse to the location of the file 'LiverNoDeformation.tif', select the file and select ***Open*** and a new window appears on the screen. Enter these image information in this window: Image Type = 8-bit, Width = 512 pixels, Height = 512 pixels, Number of Images = 147. Then press ***OK*** and the image stack will be displayed. The images in the stack may be browsed by using the scroll bar located at the bottom of the image window. The 50th and the 100th image are shown in Figure 1 and Figure 2 respectively.

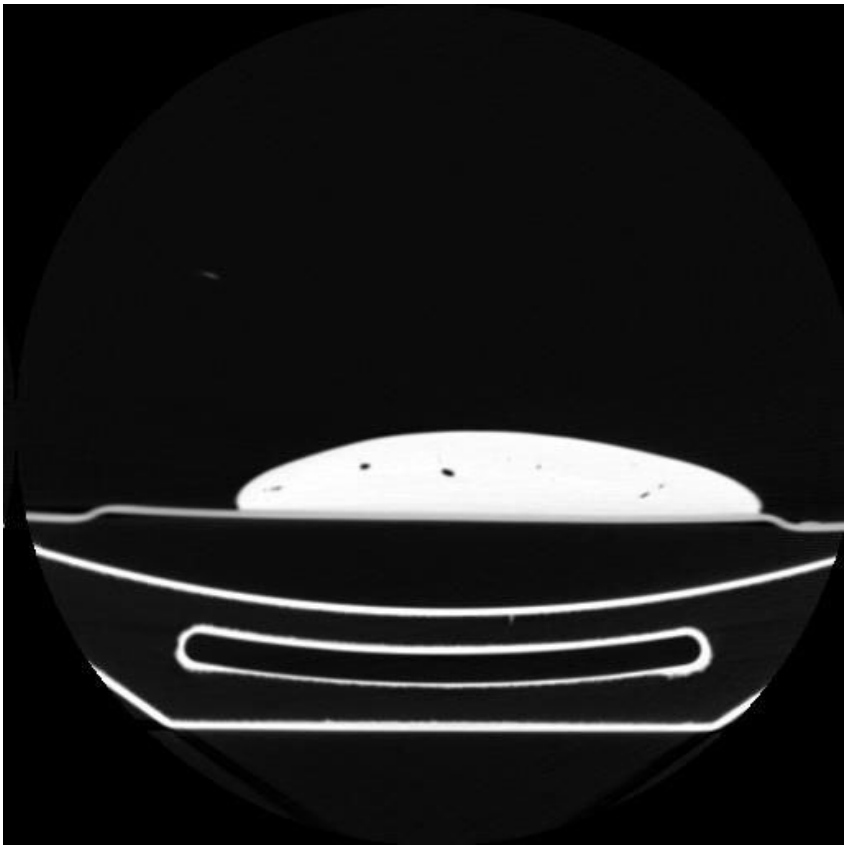

Figure 1 The 50th image.

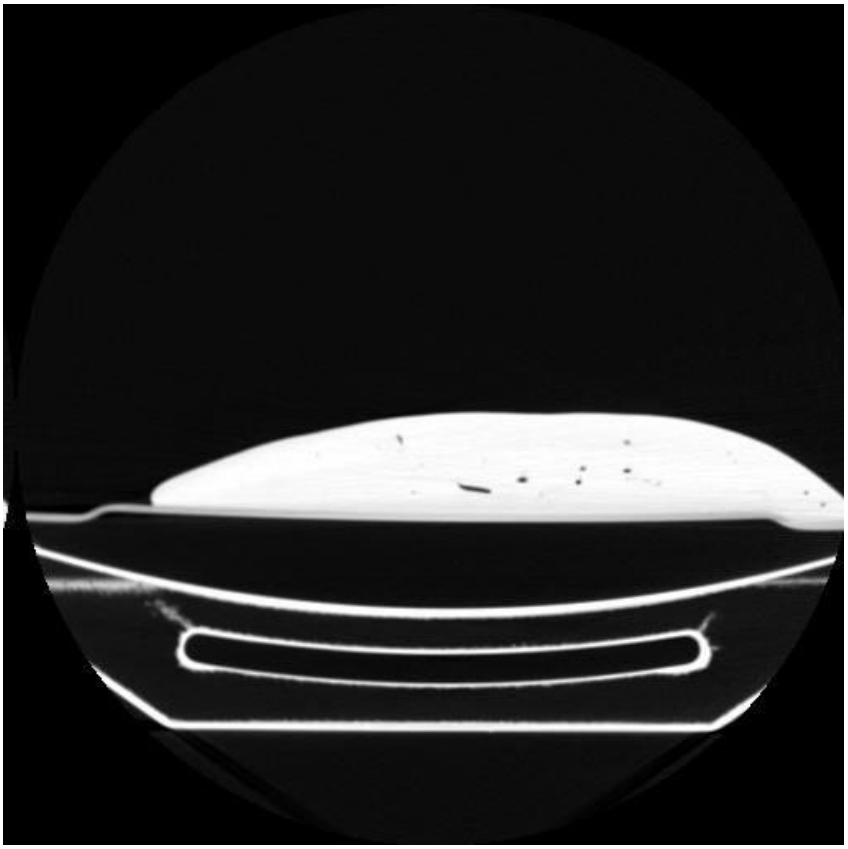

Figure 2 The 100th image.

It can be observed that the liver is located around the centre of the images and the bottom portion of the images shows the part of the experimental set up supporting the liver. Our goal here is to make the pixels belonging to the liver very bright and making the intensity values of other pixels (which do not belong to liver) zero; we should do this for all the images in the image stack. This makes the next step (segmentation) which uses ITK-SNAP very trivial, since all the pixels which belong to the liver volume have some positive intensity values but all the pixels which do not belong to the liver volume have their intensity values equal to zero. This is accomplished as follows.

The pixels belonging to the liver look brighter. Use ***Process → Math → Subtract…***, enter ***Value = 200***, select ***Yes*** to accept the processing of all 147 images. Now select the bottom portion which does not belong to the liver in a yellow rectangular box and use ***Process → Math → Multiply…***, then enter ***Value = 0***, and process all 147 images. This will set the intensity values of the pixels selected in the rectangular box to zero. Follow the same procedure to eliminate the left side portion of the image stack which does not belong to the liver. Now only those pixels which belong to the liver volume have some positive intensity value. Now use ***Process → Math → Multiply…***, then enter ***Value = 3*** to multiply these intensity values by a factor of three. This brightens up the image of the liver. The 50th and 100th image, after undergoing processing with ImageJ are shown in Figure 3 and Figure 4.

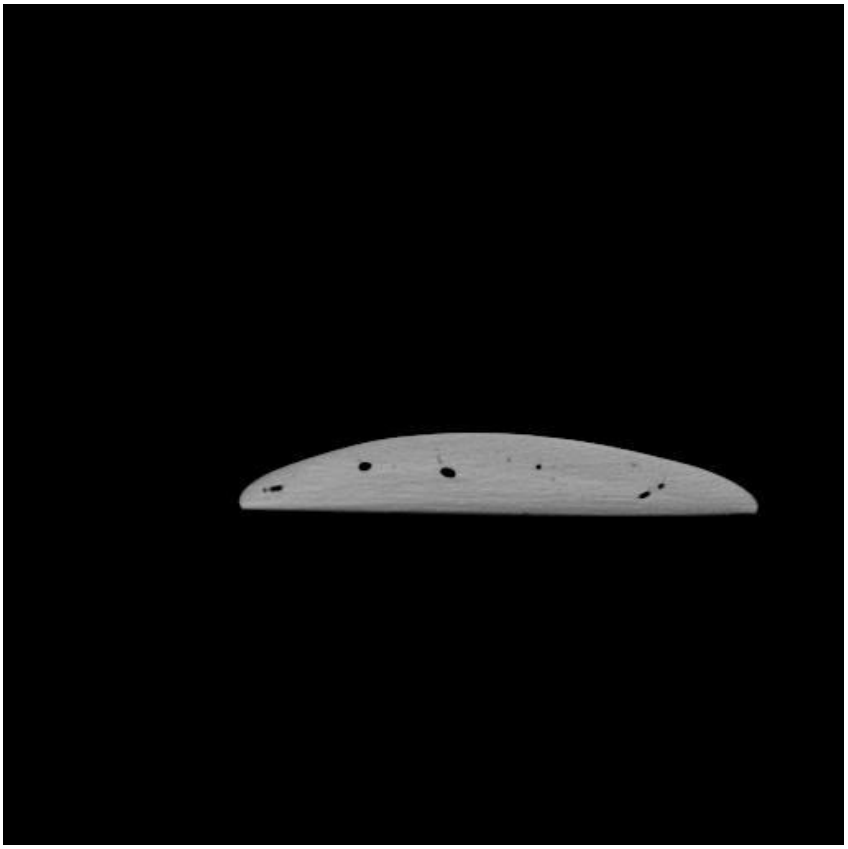

Figure 3 The 50th image (after using ImageJ).

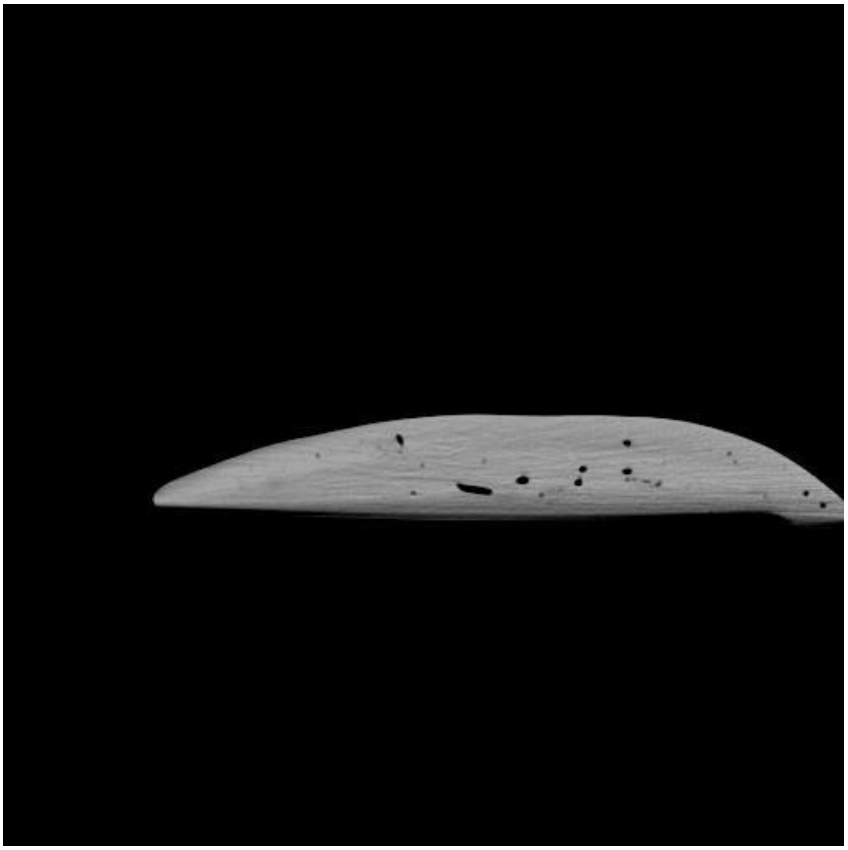

Figure 4 The 100th image (after using ImageJ).

Now, one has to save this processed image stack. Here, ***File → Save As → Raw Data…*** has been used to save the image in the .raw format. File name chosen is: a.raw. Now, we have to note that the original image stack represents only half of the complete image stack. Hence, the complete stack is obtained as follows. Use ***Plugins → Stacks → Stack Reverser*** to reverse the order of images in the image stack. Now, save this new stack using ***File → Save As → Raw Data…*** and the file name chosen here is b.raw. Now,

close all image windows, and then, ***Import*** a.raw and b.raw. Now, use ***Image → Stacks → Concatenate…***, and put a.raw as Stack1 and b.raw as Stack2. Title chosen here is ***LiverNoDeformation***. Now we have got the full stack (with 147 + 147 = 294 images) which we ***Save As*** 'LiverNoDeformation.raw'.

Use ITK-SNAP

Use ***File → Open Greyscale Image…*** to browse to the location of 'LiverNoDeformation.raw'. ***Raw Binary File*** should be selected as the image file format. Now press ***Next >*** and supply the following header information. We can remember here that we have got these information from the source which supplied the image file. We enter ***Image dimensions: X:512 Y:512 Z:29, Voxel spacing: X:0.29 Y:0.29 Z:1.00, Voxel representation: 8 bit, unsigned***. Press ***Next >*** and ***Finish***, and the image stack will be displayed. Now select ***SNAP tool*** in the ***IRIS Toolbox***. Now make sure that the region selected for segmentation (through red dotted lines) includes the whole liver, and then press ***Segment 3D***. Now select the ***Intensity regions*** radio button, and press ***Preprocess Image*** and ***Intensity Region Filter*** window pops up. Make sure that ***Preview result*** check box is ticked, select the radio button ***Below and Above***, set Lower threshold to ***0.00*** and Upper threshold to the maximum possible value, smoothness to the lowest possible value. Then press ***Okay***. Now press ***Next >***. Adjust location of cross hairs such that in each of the three windows, cross hairs are positioned at approximately the centre of the image. Adjust ***Radius*** to ***4.5***, so that after pressing ***Add Bubble***, the bubble created lies inside the image in all three windows. Now after adding the bubble, press ***Next >***. Set ***Step size*** to ***10*** and press ***Iterate continuously***. Now we can see how bubble slowly changes its shape to the shape of the image on the screen. This can take quite a bit of time and we can see the number of iterations completed from ***Iteration***. When shape of the bubble assumes the shape of the image on the screen (in all the windows), segmentation is complete and we should press ***Finish***. One should remember here that automatic segmentation has been followed here and not only the image seen on the screen, but all the images in the image stack have been segmented by now. Also, now we can note that the segmentation procedure followed here is a very simple one and the process has become easier only because we processed the image stack with ImageJ earlier. Now use ***Segmentation → Save As Mesh…*** to save the segmented volume which represents the liver as a surface mesh made of triangle faces. Here, the surface mesh file is named as 'liver.stl'. Size of the file is 86.0 MB.

Use MeshLab

Use ***File → Open*** in MeshLab to open 'liver.stl'. The liver as seen in MeshLab is shown in Figure 5. Here, the liver is represented by 288190 vertices or 576376 triangle faces. We use ***Quadric Edge Collapse Decimation*** filter to reduce the total number of faces representing the organ. Now, the same liver represented by 1002 vertices (2000 faces) is shown in Figure 6 and represented by 102 vertices (200 faces) is shown in Figure 7. We use ***MLS projection***, ***Laplacian Smooth*** and ***Taubin Smooth*** filters to improve triangle quality. The liver represented by 102 vertices (or 200 faces) only but with well shaped triangles is shown in Figure 8. Now, we will save this model (liver represented by 102 vertices or 200 faces, all triangles being well shaped) as 'liver102vprocessed.stl'. The size of the file is around 60 KB.

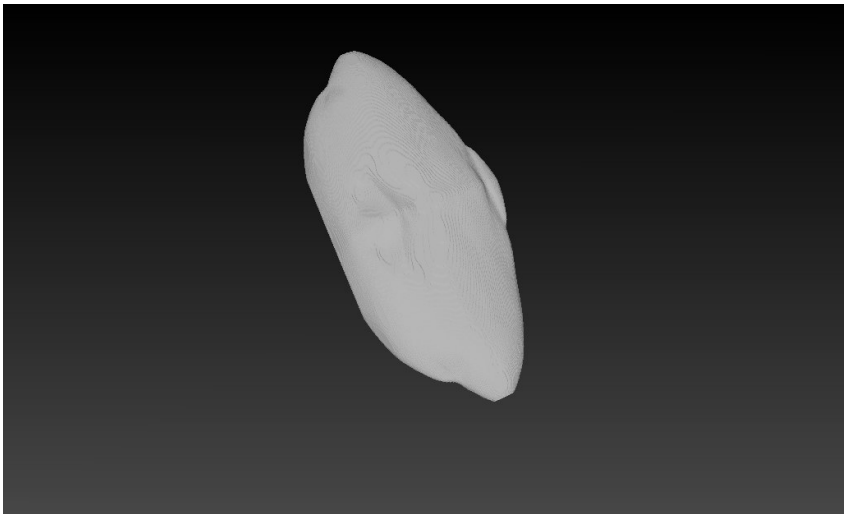

Figure 5 The liver displayed in MeshLab (288190 vertices or 576376 faces).

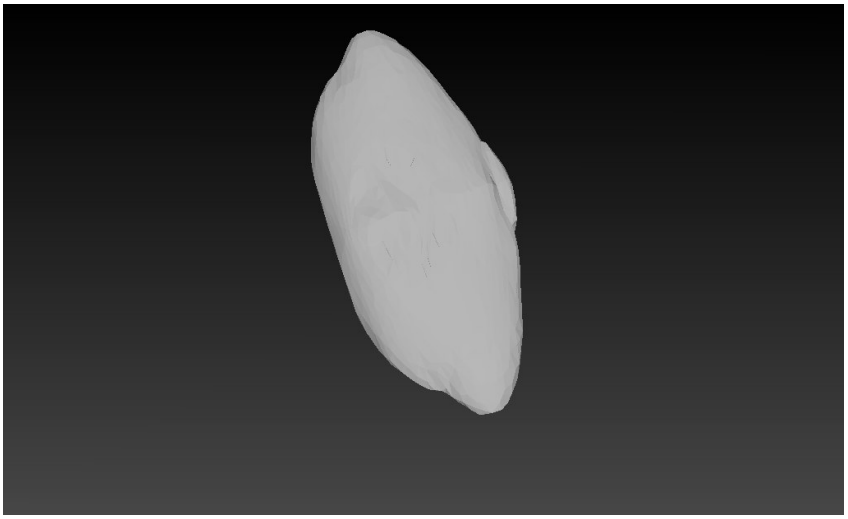

Figure 6 The liver displayed in MeshLab (1002 vertices or 2000 faces).

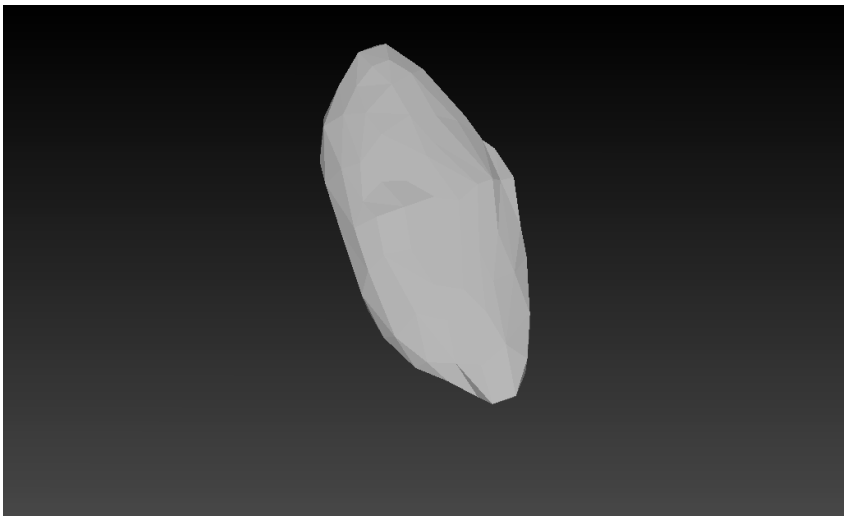

Figure 7 The liver displayed in MeshLab (102 vertices or 200 faces).

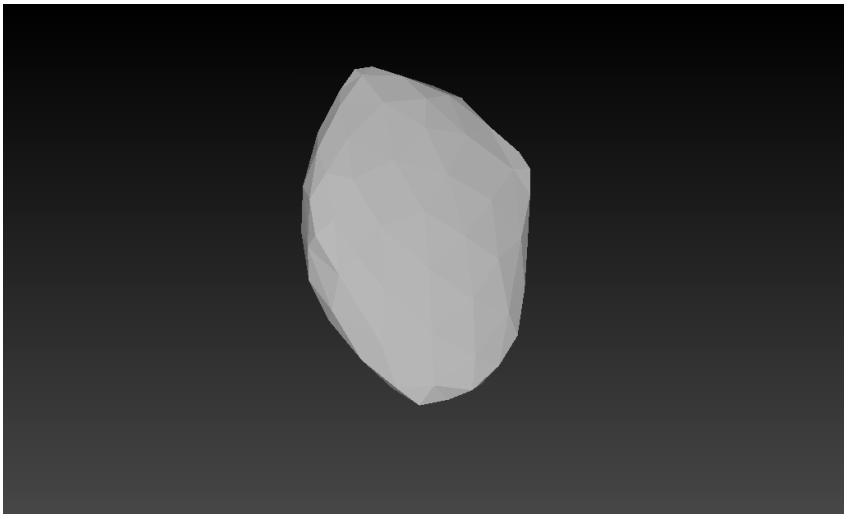

Figure 8 The liver displayed in MeshLab (102 vertices or 200 faces, well shaped triangles).

Use Rhinoceros to Convert 3D Surface to 3D Solid

Now, open 'liver102vprocessed.stl' in Rhinoceros. Use the command ***MeshToNurb***, and the resulting solid may be saved in ACIS (.sat) format using ***Save As…***. Here, the file is named as 'liver102vprocessed.sat'. Size of the file is around 500 KB. ***Now, we have obtained the organ of interest 'liver102vprocessed.sat' (3D solid) from the image stack 'LiverNoDeformation.tif'***.

Using the Organ Geometry in a Finite Element Analysis

Now the liver (or the 3D solid 'liver102vprocessed.sat') is ***imported*** into ANSYS and used in a finite element analysis. This is a dummy analysis which demonstrates that the geometry obtained by following the procedure illustrated in the previous section could be used in a nonlinear finite element analysis. A box shaped imaginary cyst is created inside the liver. Cyst is placed approximately at the center of the liver. Figure 9 shows the liver and the cyst.

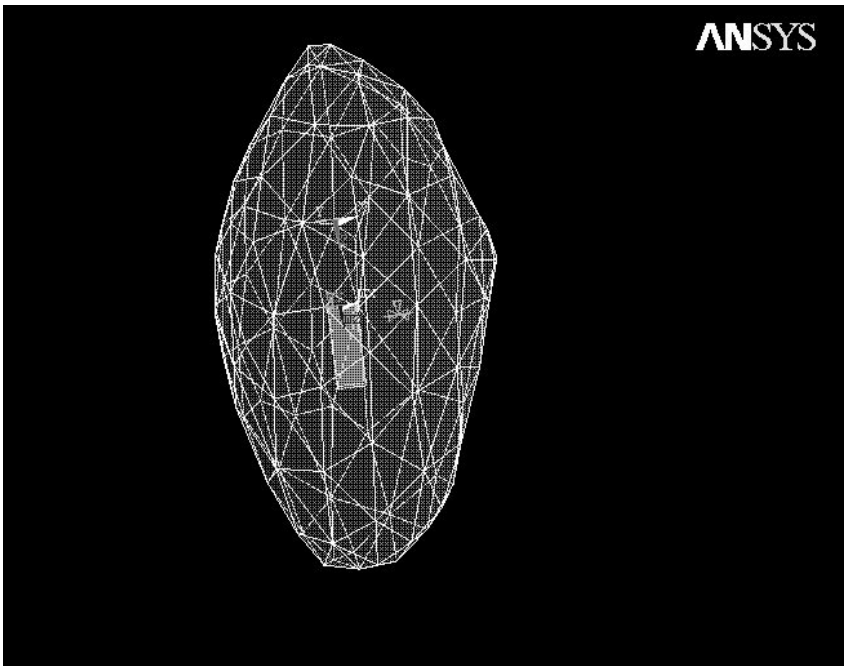

Figure 9 The liver and the cyst displayed in ANSYS.

Both liver and cyst are assumed linear elastic but they both can undergo large deformation, hence making the analysis nonlinear. The Young's modulus for the liver is taken as 200000 N/mm^2 and its Poisson's ratio is taken as 0.33 (this is equivalent to assuming that liver is made out of steel!). For the cyst, Young's modulus is taken as 100000 N/mm^2 and Poisson's ratio is taken as 0.30. Of course, these values are no way near to the real values. Next, one of the triangles forming the bottom surface of the liver is 'fixed' or it is assumed that the displacements in any direction over all of this triangle surface is zero. A 3 mm displacement is applied at a vertex located on the top surface. The displacement is applied in the Y-direction which is roughly along the thickness of the liver. The element used is: ***Solid Tet 10node 187***. The element is capable of handling large deformation. Element shape is ***Tetrahedron*** and ***Free*** meshing has been used; this is to ensure that the complex geometry is properly meshed. ***Solution → Analysis Type → Sol'n Controls → Basic → Analysis Options*** is set to ***Large Displacement Static***. Analysis is carried out and the displacement along Y-direction, at a vertex which is close to the vertex where the point load is applied, is found to be 0.19387 mm. Next, the same analysis is carried out but without any cyst inside the liver and the displacement at the same vertex along Y-direction is noted down again. It is found to be 0.18848 mm. Now, one can clearly see that the difference in the displacements obtained at the same point for the two cases is the result of the presence or the absence of the cyst inside the liver.

Conclusion

This work explains a methodology using which biological organs of interest may be obtained from image stacks produced by scanning procedures like CT-scan. Hence this methodology can be of use to get patient specific organ geometry. The procedure mainly uses free softwares and user can control how fine the organ description should be. The methodology has also been illustrated with the reconstruction of a pig liver as a live example. Result confirms that the methodology can be followed to obtain patient specific organ geometries. This work also demonstrates that finite element simulations may be performed on geometries obtained using the present methodology. The methodology seems to be of use in surgery planning and surgery simulation since both of these extensively use finite elements for numerical simulations and it is better if these simulations are carried out on patient specific organ geometries.

In fact, author is trying to develop finite element algorithms (or other numerical algorithms) which are appropriate for real-time simulations of biological organs, which can in turn be used for surgery simulations or surgery planning by surgeons, and felt the need for realistic organ geometries, and the methodology illustrated in this work is the one that was followed to get an organ geometry; it would have cost a lot if, instead of using the methodology (or the procedure) that is given in the present work for the 3D reconstruction, a commercial software were purchased to accomplish the purpose; the numerical algorithms are being tried out on the reconstructed porcine liver.

Acknowledgments

Author is grateful to the Robotics Lab, Department of Mechanical Engineering & Centre for Product Design and Manufacturing, Indian Institute of Science, Bangalore, INDIA, for providing the necessary infrastructure to carry out this work.

References

1. Kirana Kumara P, Ashitava Ghosal, "A Procedure for the 3D Reconstruction of Biological Organs from 2D Image Sequences," Proceedings of BEATS 2010, 2010, International Conference on

Biomedical Engineering and Assistive Technologies (BEATS 2010), Dr B R Ambedkar National Institute of Technology, Jalandhar.
2. http://www.3d-doctor.com
3. http://www2.amira.com
4. http://www.mathworks.com
5. http://rsbweb.nih.gov/ij/
6. Rasband, W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA, http://rsb.info.nih.gov/ij/, 1997-2009.
7. Abramoff, M.D., Magelhaes, P.J., Ram, S.J., "Image Processing with ImageJ", Biophotonics International, volume 11, issue 7, pp. 36-42, 2004.
8. http://www.itksnap.org
9. Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig, "User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability", Neuroimage, 2006 Jul 1;31(3):1116-28.
10. http://www.gnu.org/licenses/gpl.html
11. http://meshlab.sourceforge.net/
12. MeshLab, Visual Computing Lab - ISTI – CNR
13. Oztireli, Guennebaud and Gross, "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression" Eurographics 2009.
14. http://www.rhino3d.com
15. http://www.ansys.com/
16. http://biorobotics.harvard.edu
17. Amy Elizabeth Kerdok, "Characterizing the Nonlinear Mechanical Response of Liver to Surgical Manipulation", Ph.D Thesis, The Division of Engineering and Applied Sciences, Harvard University, 2006.

Comments

Sign in to write a comment