Rapid Generation of Patient-specific Anatomical Models for Usage in Virtual Environment

Gim Han Law[1], Melvin Eng[2], Calvin Lim[3], Yi Su[4], Weimin Huang[5], Jiayin Zhou[6], Jing Zhang[7], Tao Yang[8], Chee Kong Chui[9]and Stephen Chang[10]

[1]Institute of High Performance Computing, lawgh@ihpc.a-star.edu.sg
[2]Institute of High Performance Computing, engyf@ihpc.a-star.edu.sg
[3]Institute of High Performance Computing, limcw@ihpc.a-star.edu.sg
[4]Institute of High Performance Computing, suyi@ihpc.a-star.edu.sg
[5]Institute for Infocomm Research, wmhuang@i2r.a-star.edu.sg
[6]Institute for Infocomm Research, jzhou@i2r.a-star.edu.sg
[7]Institute for Infocomm Research, jzhang@i2r.a-star.edu.sg
[8]Institute for Infocomm Research, tyang@i2r.a-star.edu.sg
[9]National University of Singapore, mpecck@nus.edu.sg
[10]National University Hospital (University Surgical Cluster), surv7@nus.edu.sg

ABSTRACT

This paper presents a framework for generating patient-specific, visually realistic anatomical models from medical scan images suitable for usage in a virtual environment. The goal is to design a robust and repeatable procedure for rapid preparation of anatomical models with minimal user intervention. The proposed framework is an integration of algorithmic componentssuch as image processing and segmentation, 3D surface reconstruction, mesh decimation, remeshing and visualization. Based on this framework, we have successfully generated the anatomical models of the human liver and the gall bladder from real clinical CT scan data. Our proposed framework took less than 20 minutes for a typical CT scan of 70 slices, demonstrating that it is efficient for real clinical applications which demand rapid turnaround time, such as for pre-surgical virtual training.

Keywords:anatomical modeling, 3D reconstruction, realistic rendering, virtual reality.
DOI: 10.3722/cadaps.2011.927-938

1 INTRODUCTION

In traditional surgical planning and training, surgeons have been relying on medical imaging data for a pre-surgical examination. Some commercial systems, such as da Vinci Surgery, LaparoscopyVR [1], Lap Mentor [2] and Xitact™IHP [3], are available to assist surgical planning and training. Also, the increasing popularity of robot-assisted and minimally-invasive surgical procedures, such aslaparoscopic surgery and non-invasive virtual endoscopy [4], demands new methods in surgical training. In particular, due to space constraints, narrow field of view, and diminished tactile perception, laparoscopic surgery requires honing of advanced skills in instrument manipulation, as

well as intensive training on different patient's anatomy. Therefore, accurate depiction of patient-specific data is particularly important in such applications.

Many existing commercial offerings are passive training systems for general surgical training. The lack of variety in surgical scenarios limits the effectiveness of the training outcome. This is due to the fact that these systems used generic anatomical models which are not patient-specific. Surgeons are still required to experience real life supervised surgery, which might cast unwanted consequence in the operation. As such, it is more beneficial and effective to have a patient-specific anatomical model to be used as the subject of a surgical training and planning system.

From a technology point of view, the ability to automatically generate patient-specific anatomical models from medical imaging data (e.g., CT/MRI) is a key component for such surgical training systems. The process must be fast, reliable and capable of preserving anatomical details. The reconstructed digital models should capture accurate shapes, sizes and locations of the relevant anatomic structures with optimum number of triangles and good mesh quality for efficient usage and manipulation. This is especially crucial for deployment in a virtual environment. Inspecting patient's anatomy in a virtual environment or using virtual reality allows natural control of the visualization process and high level of interactivity. This greatly enhances the scientific and clinical value of imaging data produced by medical imaging systems.

In this paper, we present a complete framework for automatic reconstruction of realistic, patient-specific anatomical model. This framework is an integration of three main modules: image segmentation, anatomical model reconstruction and visualization. The processes are fast, accurate and reliable, and the output is suitable for deployment in a virtual reality (VR) environment.

2 OVERVIEWOF FRAMEWORK

The input to this framework is a stack of medical scanned images (e.g., MRI/CT scans). As mentioned earlier, there are three main modules in the framework. They are the image segmentationmodule, anatomical modeling module which comprises of 3D surface reconstruction and mesh processing, and visualization module (refer to Fig. 1). The following sections describe each module in detail: Section 3 describes the image segmentation methodology; Sections 4 and 5 explain 3D surface reconstruction and mesh processing modules, respectively; Section 6 describes the approach to generate realisticvisualization.

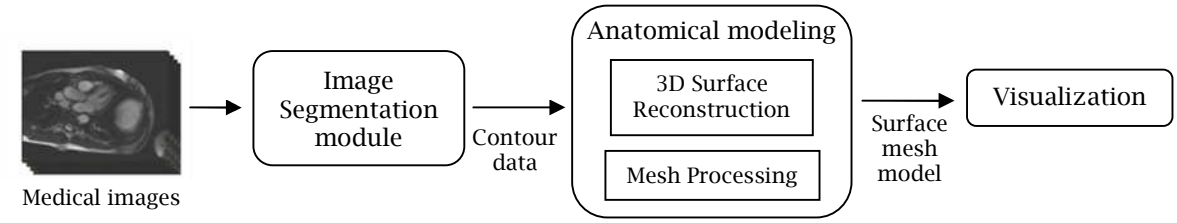


Fig. 1: Workflow of framework.

3 IMAGE SEGMENTATION

The first step in preparing a patient-specific 3D anatomical model is to delineate the region-of-interest (ROI) on the input stack of CT/MRI images. There are several mature approaches that are commonly used for 3D anatomy segmentation like deformable model-based methods (e.g., active contour (snake) [5]) and level-set method. However, these methods face issues/difficulties in handling anatomy with abnormalities, which causes the mixture of densities/signals in the CT images. In our proposed framework, we have implemented two alternatives to perform the image segmentation task - a propagation learning method and a flipping-free mesh deformation method [6].

In the first method, the segmentation of the ROI is based on voxel classification and propagation learning. First a support vector machine (SVM) classifier was trained to extract the ROI from a single image slice in the intermediate part of the anatomy of interest (of the stack of images) by supervised learning-based voxel classification. After some morphological operations, the extracted contour is then projected to the neighboring slices for automated re-sampling, learning and further voxel classification. This propagation procedure continues until all the anatomy-containing slices are processed. The method we developed had been proven and validated to be able to handle abnormal anatomy [7].

In the second method, the ROI was extracted by deforming a 3D mesh model iteratively. In each iteration, a flipping avoidance was performed before mesh deformation in which the algorithm detects and avoids possible flippings which will cause self-intersection of the mesh. This will prevent undesired segmentation results. Next, Laplacian mesh deformation is performed with various constraints in terms of geometry and shape smoothness.

For small organs with relatively homogenous regions, the propagation learning method would prove to be effective and efficient. For large organs with high inhomogeneity, the flipping-free mesh deformation is more suitable. In general, after an automatic segmentation, the quality of the output can be further enhanced by using an interactive contour editing tool to improve the fit to the actual anatomical boundary on the medical images.

4 3D SURFACE RECONSTRUCTION

Contour data obtained from the segmentation are used as an input to the reconstruction algorithm. The contour data residing on the same planar slice (same image plane), are defined as a list of closed loop straight-line segments which do not intersect each other.This reconstruction process is broken down into several iterations.In each iteration, only one pair of consecutive parallel planar slices of contours is considered.The algorithm consists of four stepsin each iteration:Data Preparation, Contour Proximity Matching, Triangulation [8] and Surface Elevation/Interpolation [9]. The following sub-sections willexplain each step in more detail.

4.1 Data Preparation

First the contour or image plane is orientedto align with the *xy* plane. Next we compute the contours nesting or containment hierarchy using CGAL [10] point location query. With the contour nesting information, the contour points are re-oriented in a consistent directionsuch that the outermost contour loop is always counterclockwise and the loop just inside the outer loop in clockwise direction.-If there are multiple levels of nesting, the contours in the subsequent levels are oriented by alternating the direction of the contours in its parent hierarchy. After this, realignment or re-spacing of the contour vertices is needed so that the distance between two consecutive vertices is of a given re-spacing parameters$P_{spacing}$. Lastly, we do a “contour pruning” so as to trim off any irregularities or noises occurring in each slice of the contours. These irregularities or noises are errors arising fromthe segmentation processinduced by bad images. Appropriate contour pruning prevents errors from propagating to the next step of the algorithm and helps in achieving a smoother surface reconstruction results in the later stage.

4.2 Contour Proximity Matching

After the data preparation, the next step is to obtain a match between portions ofcontours from adjacent slices which are in close proximity to each other.This matching process identifies and extracts sequences of contour points from different slices that are close to each other in term of Euclidean distance between the *xy*-projections of the points. In this step, a system of votes and scores matrix are used. Votes are given to a good point-to-point match only if (a) the Euclidean distance between points are less than or equal to some chosen voting parameterT_{dist},such that $T_{dist} > 0$and is sufficiently large with respect to the re-spacing parameter, and (b) the angle between their edge direction vectors is less thanT_{angle}(to ensure that matching is only carried out on consistently orientated contours). For one vote or one match, a score is computed based on$1/(d+0.01)$ where d is the *xy* distance between the two points. This matching process will only match contour portions that are consistently oriented so as to

guarantee that the material region lies on the same side of the two contours. Hence consistency in re-orientating the contour points in the previous step is important.

After all the votes and scores computation are done, matched candidates are selected if $\sum votes \geq Vmin \cap scores > Smin$ for each contour point. Based on our experimentation, $Vmin$ is best selected to be 4, S_{min} is best selected to be 20.0 and $T_{dist} = 3P_{spacing}$.

4.3 Triangulation

Each matched sequence of contour points extracted has to be connected/stitched together which will form a portion of the surface mesh to be reconstructed. A simple advancing rule is used to triangulate the matched sequence of points. We start from one end of the sequence, taking two points, u_i and v_j, each from one of the contour sequences. Assume that u_i is followed by u_{i+1} and v_j is followed by v_{j+1}. Then, if $|u_i u_{i+1}| + |v_j u_{i+1}| < |u_i v_{j+1}| + |v_j v_{j+1}|$, we advance the first chain. Otherwise we advance the second chain. This advancing process terminates when we reach the last points of both chains.

As for the remaining unmatched portions, 2D Constrained Delaunay Triangulation (CDT) is performed on the remaining contour points projected onto the *xy* plane. This initial surface triangulation serves as a basis for generating medial axes in the next step to introduce additional Steiner points. These additional points will improve the result in the surface elevation process.

4.4 Surface Elevation/Interpolation

With the initial 2D triangulation, we perform a chordal axis transform (CAT) [11] to obtain the skeletons or medial axes of the remaining contour region. Points along the medial axis will be used as Steiner points to elevate the surface mesh. In order to lift the surface up to three dimensions, a height value (in the *z* direction) will have to be assigned to the additional Steiner points. The height value is based on the *xy* distance between the points on the medial axis to the edge of the upper or lower contour slices. Once the height value has been computed, we re-triangulate the contour region with the additional points and simply assign back the *z* coordinates to all the points of the remaining contour region including the Steiner points. By this step, we would have successfully generated the surface mesh between the two contour/image slices.

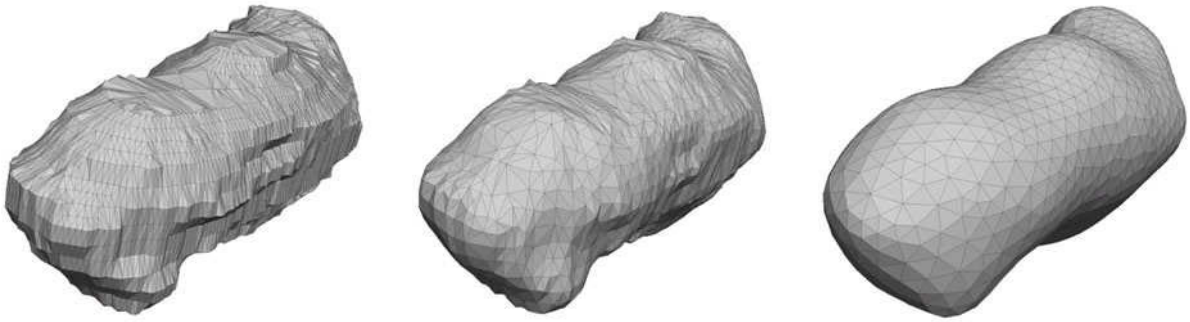
5 MESH PROCESSING

After the contours from all image slices have been processed, we obtain the initial reconstructed watertight mesh model of the anatomy of interest. However, this initial mesh usually contains various geometrical artifacts, such as stepped/terracing and sharp surfaces. In order to reduce such undesirably artifacts which is typical of data extracted from images, the mesh model is subjected to a series of post-processing procedures: mesh simplification and remeshing. With these procedures, the reconstructed surface mesh model will then be suitable for visualization purposes to achieve visual realism.

5.1 Mesh Simplification

Cignoni *et al* [12] and Heckbert *et et al* [13] had presented detailed taxonomies to categorize and compare the myriads of existing mesh simplification algorithms. In most methods, some form of error control is incorporated to prevent excessive deviation from the original mesh. Such decimation criteria include controlling the classical distance-to-plane value, minimizing the Hausdorff distances [14], using discrete curvatures as constraints [15], or simply by optimizing the shape function of the triangles [16], to name a few. Brodsky and Watson [17] presented R-Simp – a simplification algorithm designed for speed and interactivity. It works by recursively subdividing the model domain based on curvature measures and uses re-triangulation to generate the simplified mesh. The work by Cohen *et al.* [18] uses simplification envelope to simplify a polygonal model to a user-defined tolerance. The method has the advantage of preventing self-intersection and can intrinsically preserve sharp features. Hoppe [19] proposed a progressive mesh representation through an optimization procedure which preserves the overall appearance of a mesh. This method is useful for mesh simplification, transmission, compression and level-of-detail approximation.

In our proposed system, we have chosen to use the simplification method based on quadric error metrics proposed by Garland and Heckbert [20] due to its efficiency and the ease of controlling the surface error. The procedure performs arbitrary vertex pair contraction which is able to join unconnected regions of the mesh to facilitate better approximation. While the original quadric error simplification algorithm allows topological joining, we have imposed the constraint of non-manifold topology in our system as the reconstructed models are all strictly 2-manifold or watertight models. An example of a simplified mesh is shown in Fig. 2(b).



(a) (b) (c)

Fig. 2: Sample gallbladder mesh model: (a) Initial reconstructed mesh, (b) simplified mesh and (c) remeshed and regularized mesh.

5.2 Remeshing

To further enhance the regularity of the mesh, an advancing front triangulation algorithm was used to remesh the simplified mesh. As its name implies, the advancing front algorithm is a class of mesh generation algorithm which begins by taking a front (in our case, the front is a closed chain of mesh edges) as input. Mesh elements are created progressively from the front until the whole domain is filled. There had been significant research work done in the finite element mesh generation community for CAD/CAE-centric model [21, 22, 23]. However, in our application, a definite analytical surface definition is unavailable since the underlying simplified mesh is a piecewise linear model. As such, we have developed an advancing front technique by incorporating a Virtual Geometry kernel [24, 25] so that we can approximate the underlying surface definition accurately. The basis of the Virtual Geometry kernel is to interpolate each triangle of the underlying mesh using a quartic Bézier patch generated by taking into consideration the normal and tangent vectors at the vertices of each triangle. Since our input to the advancing front algorithm is a watertight model, we select an arbitrary point on the model as the input and generate a seed triangle to produce an initial front. An example of a mesh generated using the advancing front algorithm is shown in Fig. 2(c).

6 VISUALISATION

The visualization module is responsible for rendering the reconstructed mesh model in real-time 3D graphics, allowing the user to pan the viewpoint and light source around freely for visual inspection from any desired perspective and under various lighting conditions. Rendering is accomplished via OpenGL, with custom low-level shader code written to provide greater visual realism beyond the default provided by OpenGL.

6.1 Lighting

Real-time lighting for the mesh model has been implemented, with the freedom to pan the light source around as well as to move it towards or away from the mesh. Lighting quality is adjustable and increases in the level of visual realism from standard OpenGL-default Gouraud shading, to custom smooth Phong shading and finally custom bump shading.

6.1.1 Standard OpenGL-default Gouraud shading

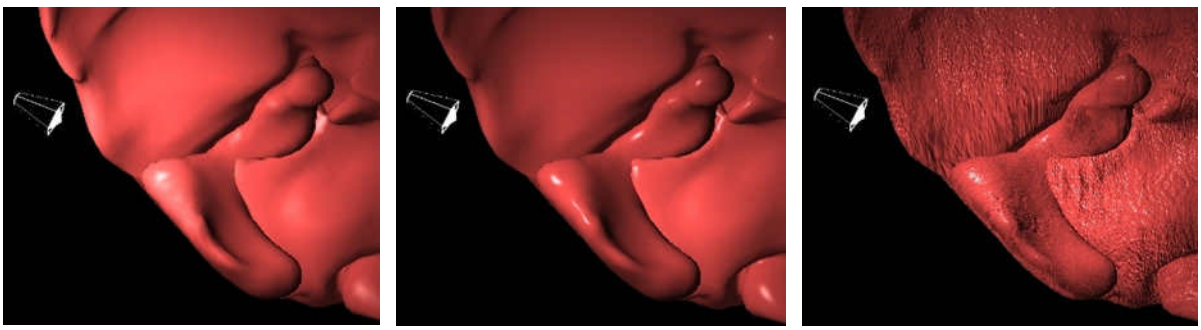
The standard OpenGL-default Gouraud shading is implemented internally by OpenGL where lighting values are calculated using the normal vector at each vertex, and then interpolated across the pixels[26]. The resulting shading quality will be acceptable only if the mesh is sufficiently tessellated. Blocky artifacts may be evident particularly with the rendering of glossy highlights on even reasonably dense meshes(see Fig. 3(a)).

6.1.2 Custom Smooth Phong Shading

Custom smooth Phong shading is accomplished via shader programming that computes lighting values using the interpolated normal vector at each pixel[26]. This yields smooth shading quality that is consistently free of blocky artifacts even for low resolution meshes, which is especially evident for glossy highlights(see Fig. 3(b)).

6.1.3 Custom Bump Shading

Custom bump shading is also achieved via shader programming except that it computes lighting values using the texture-defined normal vector at each pixel, thus making possible the appearance of complex bumpy surface details without requiring the underlying geometry to be highly tessellated(see Fig. 3(c)).

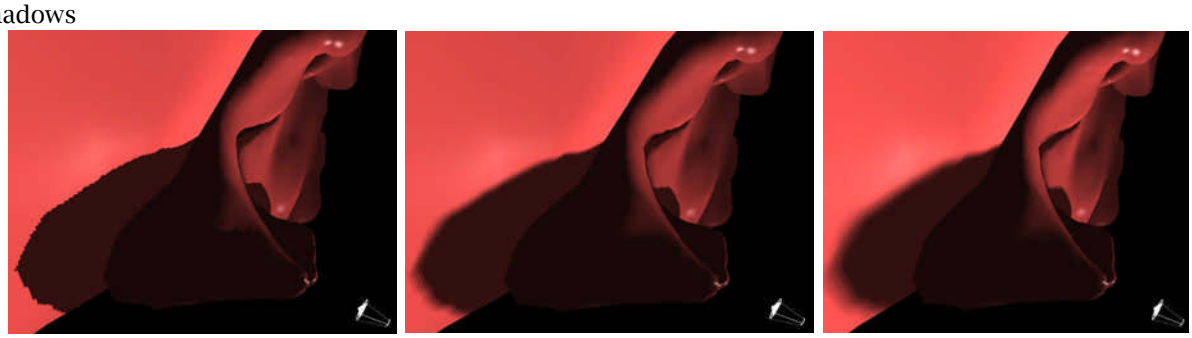


(a) (b) (c)

Fig. 3: (a) OpenGL-default Gouraud shading, (b) custom smooth Phong shading, and (c) custom bump shading.

6.2 Shadowing

Real-time shadowing, the counterpart to real-time lighting, has also been implemented. Shadows will be cast on the mesh where the light is blocked, be it by another mesh or another part of the same mesh(i.e., self-shadowing). Shadow quality is also adjustable and increases in the level of visual realism from hard-edged shadows, to uniform-filtered soft-edged shadows, to jittered-filtered soft-edged shadows



(a) (b) (c)

Fig. 4: (a) Hard-edged shadows, (b) uniform-filtered soft-edged shadows, and (c) ijtter-filteredsoft-edged shadows.

6.2.1 Hard-edged Shadows

Hard-edged shadows are rendered using standard shadow mapping[27], with the shadow boundary either in shadow or not, thus accounting for the hard edge(see Fig. 4(a)).

6.2.2 Uniform-filtered Soft-edged Shadows

Uniform-filtered soft-edged shadows are also rendered using shadow mapping, but with the additional process of performing box-filtering on the shadow outcomes of the neighboring pixels around the shadow boundary so that the pixels along the boundary vary in shades of gray[28], thus producing a softer edge(see Fig. 4(b)).

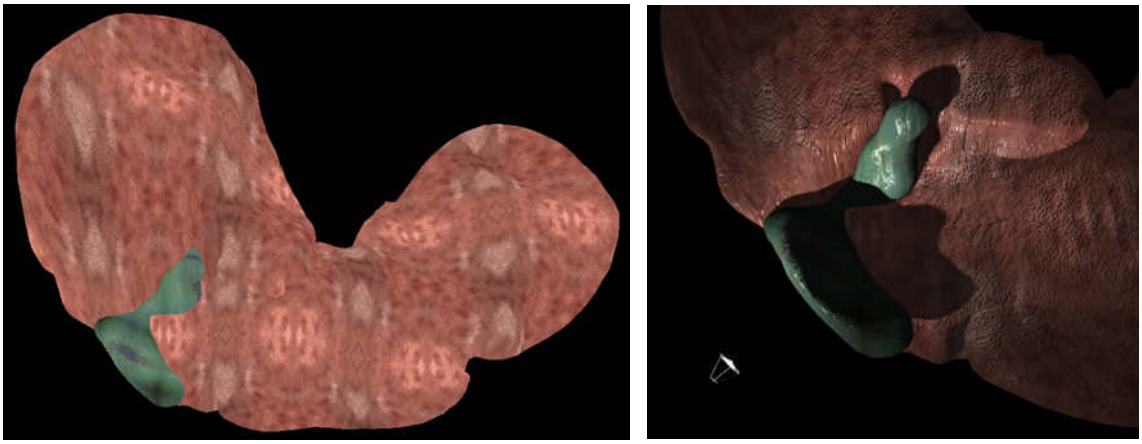
6.2.3 Jittered-filtered Soft-edged Shadows

Jitter-filtered soft-edged shadows improve upon the former by performing jittered-filtering instead of box-filtering, thus reducing boxy artifacts along the shadow boundary(see Fig. 4(c)).

6.3 Texture Mapping

Due to the fact that the topology and shape of the reconstructed mesh model varies from patient to patient, texture coordinates have to be automatically generated for the newly created mesh. This is in contrast to meshes produced by artists using modeling software where texture coordinates are statically assigned.

The current approach, with emphasis on simplicity and expediency, is to generate the texture coordinates employing orthogonal projection whereby the texture is projected onto the mesh with parallel projection lines. This ensures that reasonably appropriate texture coordinates can be generated for meshes of arbitrary topology and shape with minimal processing overheads(see Fig. 5(a)).



(a) (b)

Fig. 5: (a) Texture mapping with texture coordinates automatically generated, and (b) rendering with bump shading, soft-edged shadows and automatic texture mapping.

6.4 Maximum Detail Rendering

Rendering in maximum detail involves employing bump shading; soft-edged shadowing and automatic texture mapping at the same time(see Fig. 5(b)).

7 RESULTS

This section shows some results of digital anatomical models of the human liver and gall bladder generated byour proposed framework, based on real clinical CT scan data. However, this framework is not limited to the above-mentioned human anatomies. Our test data for this framework include3 setsof livers and 20 sets of gallbladders, with a good mixture of healthy gallbladders and gallbladders with gallstones or severe inflammation. The CT scan data are of slice thickness ranging from 0.4mm to 3 mm with in-plane resolution of 0.6-0.9 mm.As the gallbladder is a relatively small organ with more homogeneous region, the propagation learning method is employed inthe gallbladder segmentation. For the liver, which is much bigger with more challenging sample selection,we adopt the flipping-free deformation method using mixture of Gaussian to have the initial segmentation for faster computation. This is followed by interactive editing to obtain a better liver segmentation result. We have tested this framework on a desktop system with Intel Core2 Duo 2GHz, RAM 2GB, Nvidia Geforce 6800 Ultra 512MB graphics memory and 90GB hard disk.

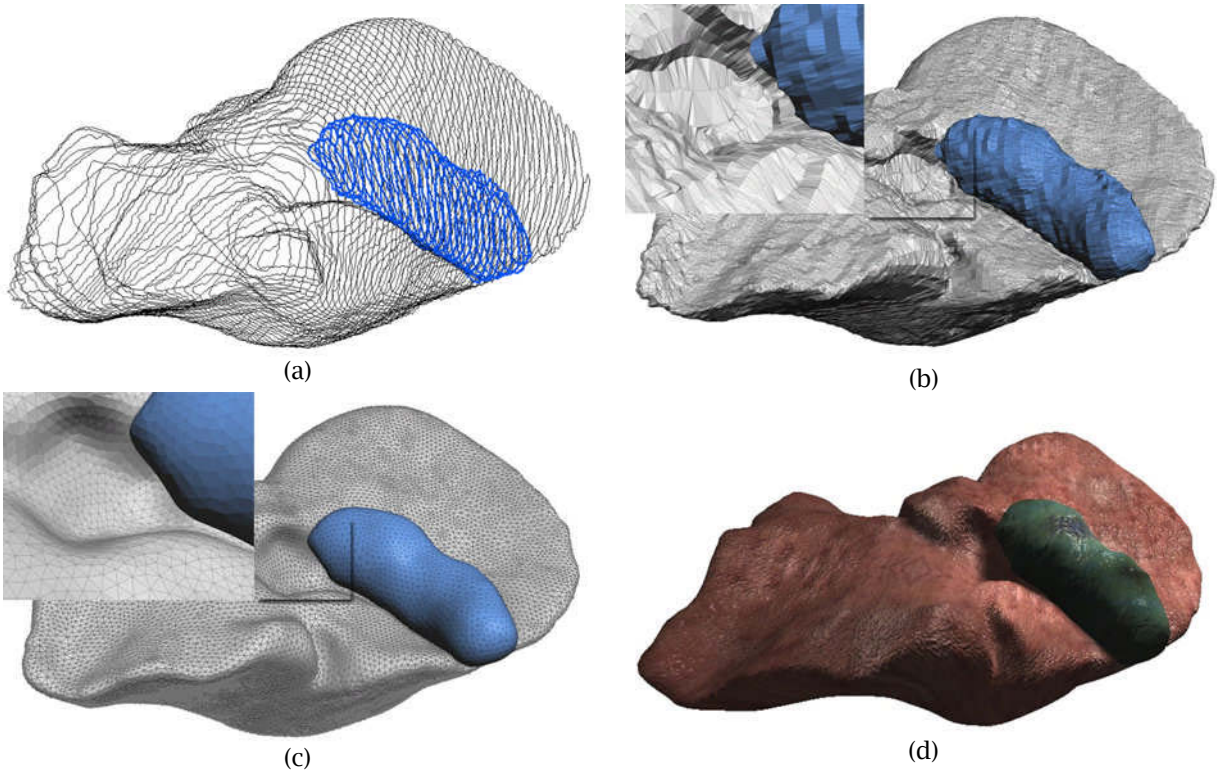
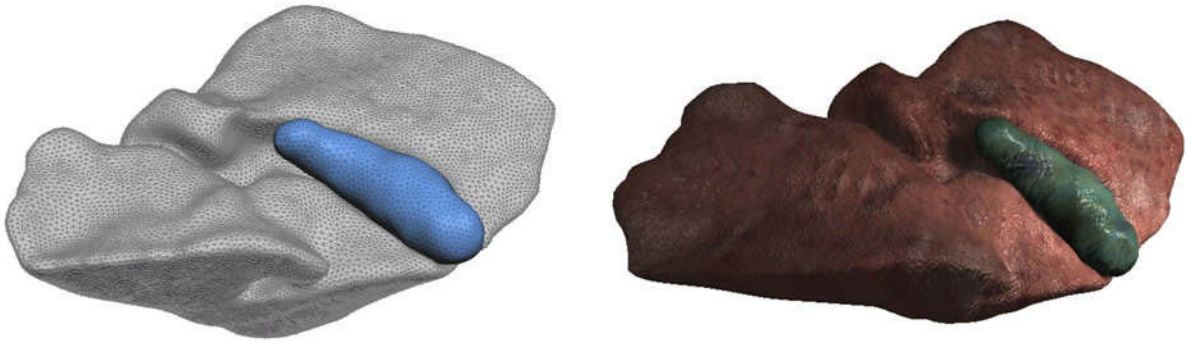


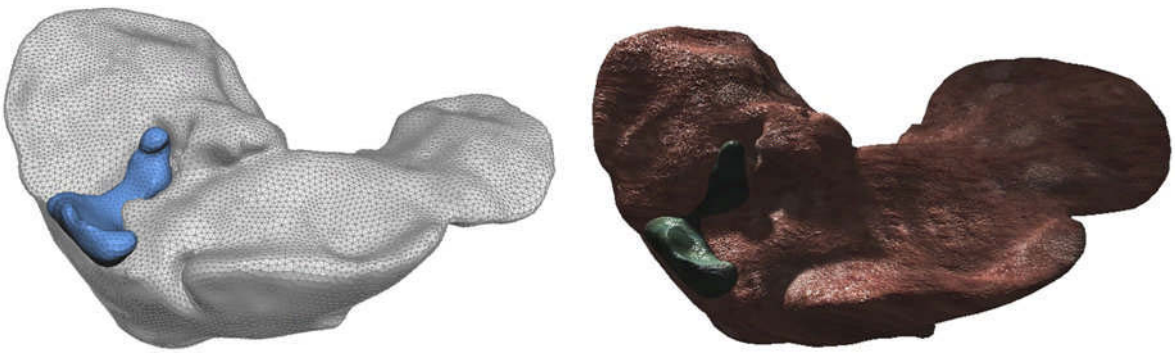
Fig. 6: Liver and gallbladder dataset 1 from input CT scans of 59 slices: (a) Contour data, (b) initial reconstructed surface mesh model, (c)processed surface mesh model, and (d) realistic rendered 3D mesh model with auto-texturing.



(a) (b)

Fig. 7: Liver and gallbladder dataset 2 from input CT scans of 62 slices: (a) Final mesh model, (b) realistic rendered 3D mesh model with auto-texturing.

Fig. 6 shows some intermediate results of an input CT scan of around 60 slices with inter-slice distance of 3mm. From Fig. 6(b), the mesh resolution is too dense and unnecessary for achieving accurate realistic rendering of the anatomy. There are also several visible geometrical artifacts, such as stepped and sharp surfaces. With the mesh simplification and remeshing process, we managed to successfully reduce the mesh resolution to what is appropriate for real-time rendering,remove undesirable geometrical artifacts and sharp surfaces, and achieve good mesh quality for rendering purposes(see Fig. 6(c)). Fig. 6(d) shows the anatomical model captured with texture mapping and realistic effect using the visualization module.Using this model, we have achieved an interactive frame rate of 20 fps during the visualization runtime.



(a) (b)

Fig. 8: Liver and gallbladder dataset 3 from input CT scans of 306 slices: (a) Final mesh model,and (b) realistic rendered 3D mesh model with auto-texturing.

Dataset		Segmentation& Mesh Initialization	Surface Reconstruction	Mesh Processing
Gallbladder 1 (28 slices, 3mm)	Time taken	7 mins	0.770s	0.334s
	Mesh size	4,590 vertices	4,274 vertices 8,544 faces	2,047 vertices 4,090 faces

	Mesh quality (°)	-	[7.078°, 151.841°]	[34.791°, 101.144°]
Liver 1 (59 slices, 3mm)	Time taken	14.75 mins	14.906s	2.683s
	Mesh size	39,625 vertices	50,808 vertices 101,612 faces	19,352 vertices 38,700 faces
	Mesh quality (°)	-	[1.169°, 171.702°]	[24.522°, 113.935°]
Gallbladder 2 (24 slices, 3mm)	Time taken	6 mins	0.765s	0.290s
	Mesh size	3,296 vertices	3,998 vertices 7,992 faces	1,974 vertices 3,944 faces
	Mesh quality (°)	-	[4.482°, 153.282°]	[30.867°, 112.547°]
Liver 2 (62 slices, 3mm)	Time taken	15.5 mins	15.593s	2.592s
	Mesh size	34,296 vertices	53,225 vertices 106,446 faces	20,860 vertices 41,716 faces
	Mesh quality (°)	-	[1.302°, 168.530°]	[25.878°, 118.430°]
Gallbladder 3 (69 slices, 0.4mm)	Time taken	17.25 mins	3.391s	2.538s
	Mesh size	12,314 vertices	17,741 vertices 35,478 faces	1,158 vertices 2,312 faces
	Mesh quality (°)	-	[0.536°, 167.318°]	[25.845°, 122.136°]
Liver 3 (306 slices, 0.4mm)	Time taken	76.5 mins	78.576s	14.112s
	Mesh size	205,248 vertices	253,859 vertices 507,718 faces	41,446 vertices 82,892 faces
	Mesh quality (°)	-	[0.226°, 176.836°]	[14.255°, 148.309°]

Tab. 1: Numerical results of three liver and gallbladder datasets.

A more detailed numerical result for each datasets is shown in the Tab. 1. This table contains the compute time for each module, mesh size in term of number of vertices and faces and mesh quality expressed in terms of the minimum and maximum angle of all the triangles in the mesh models. From these results, it is observed that the image processing and segmentation module is considered the most compute intensive. This is mainly due to the fact that the image processing and segmentation module is currently implementedin Matlab, which results in slower processing speed. Efforts are now ongoing to convert the code to an optimized C++ version.

The implementation of the propagation learning segmentation algorithm is based on propagating a segmented 2D contour from the central to the peripheral part of the anatomy. The 2D anatomy contour segmented in one slice, after necessary morphological processing, will be used as the ROI and sampling zone for its neighboring slices. In practice, if the ROI and the sampling zone are obviously undesired, a re-defining procedure will be performed interactively to improve the segmentation accuracy. In fact, this interactive re-defining procedure costs a substantial part of the processing time. Having some initial estimation on the contours using spatial coherence might allow us to further optimize our segmentation pipeline. In the flipping-free mesh deformation method, the computational cost for segmentation depends on the slice thickness of the medical images, which in turn determines the size of the data to process. For the liver models, we are processing CT scans (with thin slice thickness) of a large organ (involves many slices). Therefore, the computational cost is hefty, which explains its significant contribution to the total processing time. In actual practice, apart from the compute time of the algorithms, the total actual turnaround time is also affected by the requirement to perform manual contour adjustment. If the quality of initial segmentation is poor, moderate to considerable time is needed for manual editing. If the quality of initial segmentation is rather good, only minor editing is required and time can be greatly saved.

We also observed that the amount of processing time taken is actually linearly proportional to the number of input image slices, with average time of 15.17 seconds taken for one slice. Naturally with the increase in the number of slices, the resulting contour data points output from segmentation will also increase, thus more computation/processing time will be required to reconstruct the surface when more contour points have to be considered. In terms of the mesh quality, we have achieved good improvement from a minimum average angle of 2.465° to 26.026° and have reducedthe maximum average angle from 164.918° to 119.416° between the initial reconstructed surface mesh model and the mesh model after mesh processing. This improvement is also clearly reflected in Figs. 6(c), 7(a) and 8(a). In each case, we have achieved an interactive frame rate of at least 20 fps for the visualisation module.

8 CONCLUSION

In this paper,we proposed a framework for rapid generationof patient-specific anatomical models for usage in a virtual environment. This framework served to overcome the current limitationsof most surgical training or planning system in terms of the lack of real specific operative anatomical model. While only human liver and gallbladder datasets were used to demonstrate the efficacy of this framework, it could well be extended to more complex and critical surgical scenarios like cardiac and neurosurgical operation and planning.

Our method was able to generate patient-specific models with a relatively fast processing time while achieving 3D triangular mesh modelsof sufficient mesh resolution and good mesh quality with minimum triangle angle of around 30° and maximum triangle angle of 120°, together with auto-texturing and realistic visual rendering from real medical images.

There are plans to extend this framework to include physics-based simulation for modeling anatomical motion and behaviour. The purpose is to create a realistic user experience for real-time interaction with the virtual 3D anatomical models, such as in simulating the experience of cutting and deformationof anatomical models.In addition, we will includerealisticvisualeffectsto simulate bleeding in organs and develop a closed-loop robotic system with haptics feedback to improve user experience.

REFERENCES

[1] The LaparoscopyVR Virtual-Reality System, Immersion.http://www.immersion.com/markets/medical/index.html.

[2] LAP Mentor, SimBionix. http://www.simbionix.com/LAP_Mentor.html.

[3] Xitact™ IHP Instrument Haptic Port. http://mentice46.kaigan.se/archive/pdf_products/Product sheet Xitact IHP_2007-10-12.pdf.

[4] Robb, R.; Aharon, S.; Cameron, B.: Patient-specific anatomic models from three dimensional medical image data for clinical applications in surgery and endoscopy, Journal of Digital Imaging, 10(1), 1997, 31-35.DOI:10.1007/BF03168651

[5] Michael, K.; Andrew, W.; Demetri, T.: Snakes: Active contour models,International Journal of Computer Vision, 1(4), 1987, 321-331.DOI:10.1007/BF00133570

[6] Ding, F.; Yang, W.; Leow, W. K.; Venkatesh, S.: Segmentation of Soft Organs by Flipping-Free Mesh Deformation, in Proc. 2009 Workshop on Applications of Computer Vision (WACV), 2009, 1-7. DOI:10.1109/WACV.2009.5403096

[7] Zhou, J.;Huang, W.; Zhang, J.; Yang, T.; Liu, J.; Chui, C. K.: Segmentation of gallbladder from CT images for a surgical training system,in Proc. 3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010), Yantai, China, 4, 2010, 536 - 540.DOI:10.1109/BMEI.2010.5639989

[8] Barequet, G.;Sharir, M.: Piecewise-linear interpolation between polygonal slices,Comput. Vision Image Understanding, 63(2), 1996, 251-272.DOI:10.1006/cviu.1996.0018

[9] Goodrich, M. T.; Barequet, G.; Levi-Steiner, A.; Steiner, D.: Contour interpolation by straight skeletons, Graph. Models, 66(4), 2004, 245-260.DOI:10.1016/j.gmod.2004.05.001

[10] CGAL, http://www.cgal.org, Computational Geometry Algorithms Library.

[11] Andres, E.; Damiand, G.; Lienhardt, P.; Prasad, L.: Rectification of the chordal axis transform and a new criterion for shape decomposition,Discrete Geometry for Computer Imagery - Lecture Notes in Computer Science,3429, 2005, 263-275.DOI:10.1007/978-3-540-31965-8_25
[12] Cignoni, P.; Montani, C.; Scopigno, R.: A comparison of mesh simplification algorithms, Comput. Graph.,22(1), 1998, 37–54.DOI:10.1016/S0097-8493(97)00082-4
[13] Heckbert, P. S.; Garland, M.: Survey of polygonal surface simplification algorithms. Tech. report, Carnegie-Mellon Univ., School of Computer Science, 1997, ftp://ftp.cs.cmu.edu/afs/cs/project/anim/ph/paper/multi97/release/heckbert/simp.pdf.
[14] Klein, R.; Liebich, G.; Straer, W.: Mesh reduction with error control, in Proc.Visualization, 1996, 311–318.DOI:10.1109/VISUAL.1996.568124
[15] Kim, S. J,; Kim, C. H.; Levin, D.: Surface simplification using a discrete curvature norm, Comput. Graph., 26, 2002, 657–663.DOI:10.1016/S0097-8493(02)00121-8
[16] Ollivier-Gooch, C.: Coarsening unstructured meshes by edge contraction, Int. J. Numer. Meth. Engng., 57(3), 2003, 391–414.DOI:10.1002/nme.682
[17] Brodsky, M.; Watson, B.: Model simplification for interactive applications,in Proc. IEEE Virtual Reality Conference 2000 (VR'00), 2000, 286.DOI:10.1109/VR.2000.840514
[18] Cohen, J.; Varshney, A.; Manocha, D.; Turk, G.; Weber, H.; Agarwal, P.; Brooks, F.; Wright, W.: Simplification envelopes,in Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques, 1996, 119–128.DOI:10.1145/237170.237220
[19] Hoppe, H.: Progressive meshes,in Proc. 23rd Annual Conference on Computer Graphics and Interactive Techniques, 1996, 99–108.DOI:10.1145/237170.237216
[20] Garland, M.; Heckbert, P. S.: Surface simplification using quadric error metrics, in Proc. SIGGRAPH, 1997, 209–216.DOI:10.1145/258734.258849
[21] Löhner, R.; Oñate, E.: Advancing front techniques for filling space with arbitrary separated objects, Finite Elements in Analysis and Design, 46(1-2), 2010.DOI:10.1016/j.finel.2009.06.032
[22] El-Hamalawi,A.: A 2D combined advancing front-Delaunay mesh generation scheme, Finite Elements in Analysis and Design, 40(9-10), 2004 967–989.DOI:10.1016/j.finel.2003.04.001
[23] Hartmann,E.: A marching method for the triangulation of surfaces, The Visual Computer, 14(3), 1998, 95–108.DOI:10.1007/s003710050126
[24] Su, Y.; Chua, K. S.; Chong, C. S.: Mesh processing using virtual geometry, WSEAS Transactions on Computers, 5(4), 2006, 697-704.
[25] Su, Y.; Senthil Kumar, A.: Templatized refinement of triangle meshes using surface interpolation, Int. J. Numer. Methods Eng., 65(9), 2006, 1472-1494.DOI:10.1002/nme.1503
[26] Pfenning, F.: Shading In OpenGL, 15-462 Computer Graphics I Lecture, 8, 2003.
[27] Everitt, C.; Rege, A.; Cebenoyan, C.: Hardware Shadow Mapping, http://developer.nvidia.com/attach/8456.
[28] Bunnell, M.; Pellacini, F.: Shadow map aliasing, in: Fernando, R. (Ed.) GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics, Addison-Wesley Professional, 2004.