Journal of Electronic Imaging 11(4), 423–433 (October 2002).

Parallel unfolding and visualization of curved surfaces extracted from large three-dimensional volumes

Oscar Figueiredo
CPE Lyon
43 bd du 11 Novembre 69616
Villeurbanne Cedex, France
E-mail: oscar@cpe.fr

Roger D. Hersch
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
E-mail: rd.hersch@epfl.ch

Abstract. *Although many three-dimensional (3D) medical imaging visualization methods exist, 3D volume slicing remains the most commonly used technique for visualizing medical data from modalities such as CT, MRI, and PET. We propose to extend the possibilities of oblique slicing to developable curved surfaces that can be flattened and displayed in two dimensions without deformation. Such surfaces can be used to follow curved anatomical structures while preserving distance metrics at visualization time. They may also be useful for the staging of tumors, i.e., to evaluate the spatial extension of a tumor. We propose an out of core algorithm that runs in parallel on a multi-PC architecture and is able to extract surfaces from very large 3D datasets such as the visible human data set (man: 13 GB, woman: 49 GB). Experimental performance results are presented which demonstrate that parallel surface extraction is scalable and has a reasonable overhead compared with traditional oblique planar slicing. Surface extraction is made available to the public as one of the services offered by EPFLs visible human web server (http://visiblehuman.epfl.ch).* © 2002 SPIE and IS&T.
[DOI: 10.1117/1.1505962]

1 Introduction

Biomedical imaging modalities such as CT, MRI, and PET produce a series of evenly spaced parallel slices which are examined side by side by physicians, mentally rebuilding a spatial representation of the scene. In order to offer more flexibility, slices can be stacked to build a three-dimensional (3D) volume which allows, for instance, the extraction of planar slices of arbitrary orientation (an operation called multiplanar reprojection).[1] However, there are many cases where oblique planar slices do not allow to adequately follow anatomic structures. In contrast to planar slices, surfaces offer the possibility of tracking and visualizing curved anatomic structures. For instance, a surface is needed to visualize the connection between superior vena cava, right atrium, and inferior vena cava (see Appendix B). In addition, extraction of surfaces from tomographic volume images (CT, MRI) may prove to be useful for the staging of tumors, i.e., for delimiting the extension of a tumor in order to evaluate possible therapeutic options.

The very large size 3D volumes produced by medical imaging modalities represent another problem. Indeed, processing such volumes requires high computing power and high-throughput out of core algorithms, i.e., algorithms which instead of accessing data in memory access medical image data directly on disks. To address these issues, we present a system for the parallel out of core visualization of curved surfaces that are extracted from large 3D datasets such as the visible human data set.[2] The software is designed for a parallel architecture based on commodity components, i.e., PCs connected each one to several disks and interconnected by Fast Ethernet. The surfaces, generalized cylinders, extracted with this method nicely extend the possibilities offered by oblique planar slicing. The resulting flattened surfaces allow distances between points along the surface to be measured, for example the distance between successive teeth roots (Fig. 1). The use of integer-based discrete geometry calculations facilitates the parallelization of the algorithm and enables access to 3D volumes striped across several disks. Surface extraction is available on the Web (http://visiblehuman.epfl.ch), in addition to slice extraction,[3] slice animation extraction,[4] real-time navigation,[5] and interactive construction of anatomic structures.[6]

In Sec. 2, we introduce the concepts of ruled surfaces and digital cylinders. In Sec. 3, we present the basic methods for extracting and unfolding surfaces. In Sec. 4, we describe two approaches for the parallel out of core extraction of surfaces. The performance analysis is carried out in Sec. 5 and the conclusions are drawn in Sec. 6.

Paper II-02 received Feb. 7, 2002; revised manuscript received May 28, 2002; accepted for publication Jun. 3, 2002.
1017-9909/2002/$15.00 © 2002 SPIE and IS&T.

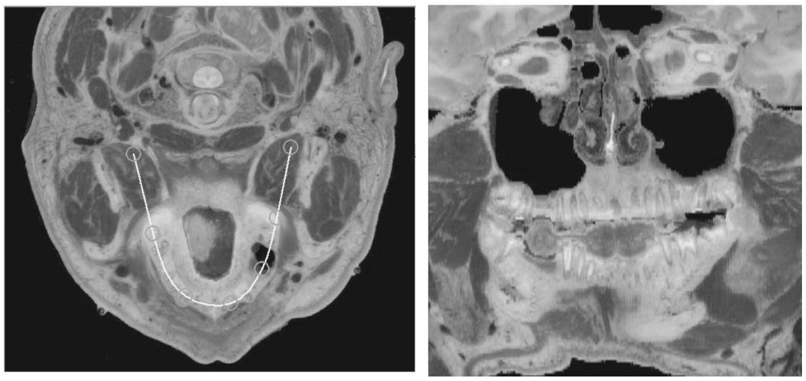


Fig. 1 Surface specification by means of a natural spline (left) and corresponding extracted flattened surface (right).

2 From Oblique Slices to Digital Cylinders

2.1 *Developable Surfaces and Cylinders*

A special class of surfaces, called developable surfaces, is of particular interest for visualization purposes since these surfaces can be ''unfolded'' and ''flattened'' without deformation. Mathematically speaking, developable surfaces are ruled surfaces having a constant tangent plane along each ruling. Ruled surfaces are themselves differentiable maps defined by a curve $\alpha(t)$ in R^3 and a parameterized family of directions $\mathbf{w}(t)$ of R^3:

$$\sigma(t,\nu)=\alpha(t)+\nu\mathbf{w}(t). \tag{1}$$

The curve $\alpha(t)$ is called the directrix of the surface while the family of lines L_t passing through $\alpha(t)$ and parallel to $\mathbf{w}(t)$ are called rulings of the surface. Among developable surfaces we shall focus on cylinders. A cylinder is a ruled surface whose directrix is contained in a plane and whose rulings are of constant direction. Note that this notion encompasses but is not restricted to circle-based cylinders. In the present application, we use spline-based cylinders. If we restrict ourselves to the case of cylinders whose rulings are normal to the directrix plane, user interaction for the specification of the surface to be extracted becomes simple. The plane P containing the directrix can be specified by using the same user interface as the one used for the extraction of oblique planar slices.[3] The directrix itself needs to be defined as a two-dimensional (2D) curve on plane P. This can be done for example by defining a spline by means of interpolation points. Due to the simplicity of their specification, spline-based cylindric surfaces are particularly well-suited for unfolding and visualization.

2.2 *Digital Cylinders*

Several approaches are possible for extracting cylindric surfaces from 3D voxel-based volumes. One may use 3D texture mapping, which allows mapping 2D and 3D images onto free-form surfaces.[7] This technique can be hardware-accelerated.[9] However, generally, the size of the 3D texture is limited to the size of the available main memory. In addition, 3D rendering engines do not support the unfolding of surfaces, i.e., they generate surface projections which are not fully exploitable by physicians since projections do not preserve the original surface geometry (distances and angles).

Kaufman *et al.* proposed several algorithms to perform the 3D scan conversion of various geometric objects including free-form Bézier surfaces.[8] Their method consists in incrementally drawing the voxels derived from the continuous surface equation. Error accumulation in incremental fixed-point or floating-point algorithms may yield ambiguous rasterizations, i.e., rasterizations which depend on the scanning direction.[9] We need to ensure that surface parts extracted in parallel from several subvolumes precisely fit side by side. We therefore base our algorithms on discrete geometry,[10] where only rational numbers having integer numerators and denominators are used. Algorithms derived from discrete geometry are efficient since they use integer arithmetic and avoid the pitfalls incurred by going back and forth between the discrete and continuous spaces. The rigorous definitions for digital lines and planes given in Refs. 10 and 11 have been used to derive an efficient incremental parallel oblique slice extraction algorithm.[12] The definition and discrete geometry representation of complex digital surfaces is, however, a more difficult problem. For the purpose of cylinder surface extraction, we represent a digital surface by a mesh of digital polygons.[13] In the same way as a digital curve is strictly equivalent to a discrete polygonal line, i.e., a sequence of digital straight line segments, a thin digital surface can be conveniently represented as a sequence of digital plane patches. Thanks to the geometric properties of cylinders, we can easily determine the equivalent representation of a digital cylinder by a mesh of adjacent digital plane patches.

Let us consider a cylinder in Euclidian space R^3 verifying Eq. (1). Its directrix $\alpha(s)$ is a natural spline which can be represented as a sequence of Bézier curves.[14] We build an equivalent digital cylinder by first polygonalizing $\alpha(s)$ on the directrix plane P by recursive subdivision of its Bézier spline control polygons. This yields a polygonal representation $\pi_\alpha(s)$ of the curve $\alpha(s)$ whose digitization is the same as the digitization of the original curve $\alpha(s)$ in a given 2D pixel grid.[15] The new cylinder defined by $\pi_\alpha(s)$ and $\mathbf{w}(t)$ consists of a series of adjacent Euclidian planar facets (Fig. 2). The discrete equivalent to this surface,

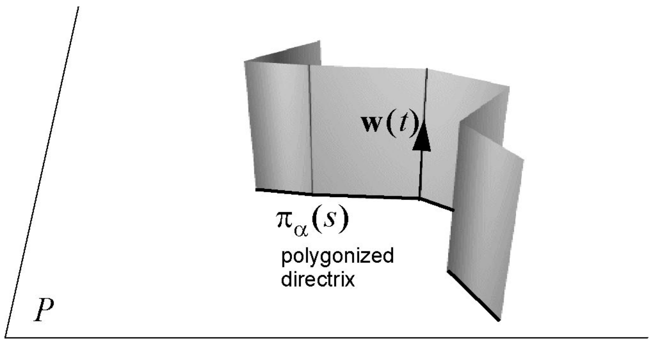


Fig. 2 A polygonal cylinder.

called the digital cylinder, is the union of a set of digital plane patches (digital facets), each fitting one of the Euclidian facets in the sense of a best approximation of an Euclidean plane by a naive digital plane.[11]

The extraction of cylindric surfaces can be seen as an extension of the planar slicing problem: planar facets are extracted and combined together to render the final flat nondistorted view of the cylinder surface. To respect the original surface geometry (distances), successive facets need to be resampled according to a single continuous 2D grid traversing the facets boundaries (Sec. 3).

3 Surface Extraction

3.1 *Algorithm Overview*

It should be possible to extract surfaces from virtually unlimited 3D volume sizes. We therefore describe surface extraction from 3D volumes distributed over multiple disks. We consider the 3D visible human cryosection data set comprising 1840 1 mm spaced slices, each scanned at a resolution of 3 pixels/mm, yielding slices of size 2048 $\times$1216 pixels. The 3D visible human data set is reorganized into subvolumes called extents of size 32$\times$32$\times$17 RGB voxels. For the purpose of resampling in the z direction by linear interpolation, we use "fat extents." Fat extents incorporate as their last 32$\times$32 surface layer the first layer of the next extent in the z direction (vertical body axis). Extents are distributed across the available disks using the PS^2 parallel file striping system[16] so as to provide a high degree of load balancing between the contributing disks.[17]

The extraction of the cylinder to be visualized proceeds as follows. In a first step, the directrix is polygonalized by recursive subdivision of its Bézier spline control polygons. The subdivision is stopped when the resulting digital polyline is identical to the corresponding digital representation of the directrix.[15] The polyline defines a digital cylindric surface (digital cylinder) that fits the Euclidian cylinder. The facets of that digital cylinder are extracted as digital rectangles, resampled and merged as horizontal strips into the final display buffer (Fig. 3).

3.2 *Extraction of the Facets*

The extraction of each facet uses a parallel version of the digital plane scan-conversion algorithm.[11] The algorithm proceeds by first determining the extents (subvolumes) that intersect the required facet given its geometric specifications. Each extent is read from the disks and the portion of the facet, called a facet part, that is contained in each of these extents is extracted. Facet voxels are read from the extent, (in order to obtain facets within an isometric 3D grid, a linear interpolation in the z direction is also performed), and, after positioning of the display grid (Sec. 3.3), resampled by nearest-neighbor interpolation. Facet parts are merged (a simple OR operation) at their relative position in the final display buffer.

3.3 *Facet Merging*

To visualize the entire surface on the display grid, individual facets need to be combined together to form a single image. For an accurate display of the surface, distances should be preserved. However, in the general case, the arc lengths of the polygonalized directrix are irrational whereas facets extracted as discrete rectangles have an integer size. Therefore, the final display grid needs to be laid out successively on all contributing facets. Figure 4 shows the

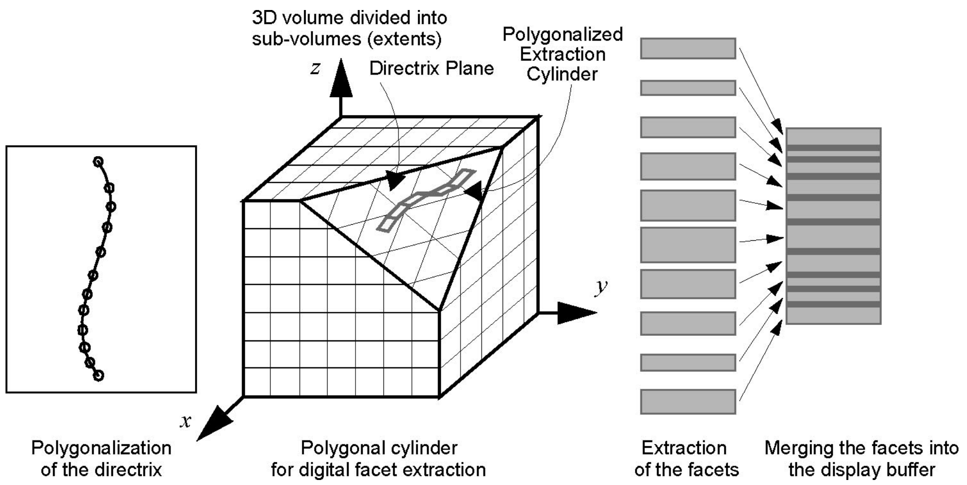

3D volume divided into sub-volumes (extents)
Polygonalized Extraction Cylinder
z
Directrix Plane
y
x
Polygonalization of the directrix
Polygonal cylinder for digital facet extraction
Extraction of the facets
Merging the facets into the display buffer

Fig. 3 Overview of the surface extraction algorithm.

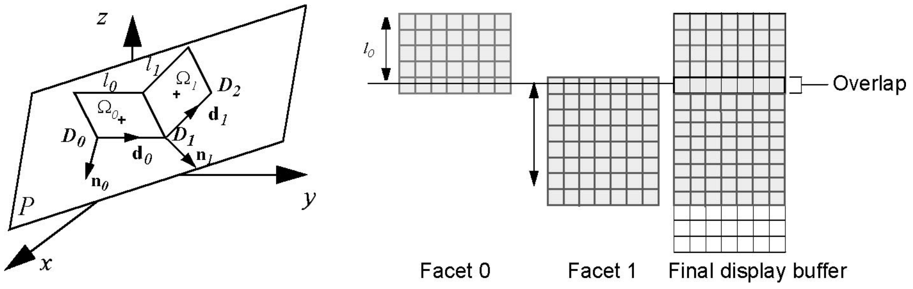


Fig. 4 Merging facets into the final display buffer.

junction of two successive facets. Since the facet length $l_0 = D_0D_1$ is not an integer, the grid of the second facet must be shifted by a fractional amount.

By translating the center of the facets and slightly incrementing their size, we can align the facet grids with the display grid. The facet translation and enlargement are calculated starting from the real size of each facet. Figure 5 represents the polygonalized surface projected on the directrix plane P and shows how the translation $\alpha_i = F_iD_i$ is computed.

In Fig. 5, small dots represent the horizontal pixel boundaries of the display grid. The directrix vertices (after recursive subdivision) are denoted D_i. The new length and position of the ith facet are defined by F_iG_i where F_i and G_i are computed according to the recurrence given in Eq. (2):

$$
\begin{aligned}
&F_0 = D_0,\\
&\overline{F_iD_i} = \mathrm{frac}(\overline{F_{i-1}D_i}) \quad i \geqslant 1\\
&\qquad \mathrm{frac}(x)\text{: fractional part of } x, \qquad (2)\\
&\overline{F_{i-1}G_{i-1}} = \mathrm{ceil}(\overline{F_{i-1}D_i}) \quad i \geqslant 1\\
&\qquad \mathrm{ceil}(x)\text{: smallest integer greater or equal to } x.
\end{aligned}
$$

F_i is computed so as to provide an offset that corresponds exactly to the fractional part of the length of the i-1th facet, Thus, the grid of facet i is properly aligned with the display grid.

By proceeding this way, extracted and resampled facets can be merged into the surface display buffer without additional resampling. At the junction of two facets, pixel lines of each of the two facets contribute to the resulting display pixel line (Fig. 4) and need therefore to be blended together. The contribution of each facet to the junction line is simply determined by their respective fractional sizes [E_0D_1 and D_1H_1 in Fig. 5(b)]. The blending weights $w_i(i)$ and $w_i(i+1)$ of facets i and $i+1$ at their junction are given by

$$
\begin{aligned}
&w_i(i) = F_{i+1}D_{i+1},\\
&w_i(i+1) = \overline{D_{i+1}H_{i+1}} = 1 - w_i(i).
\end{aligned} \qquad (3)
$$

An alternative to the presented algorithm would consist in mapping the target display grid onto the continuous cylinder, with one display grid axis mapped onto the directrix and the other mapped perpendicular to it. Display grid point intensities would be obtained by resampling. i.e., interpolation between neighboring volume voxels. Such a solution would not require facet merging. However, the use of floating-point numbers would require special care in order to avoid ambiguities when resampling grid points located at extent boundaries and would therefore make a parallel implementation much more awkward.

4 Parallel Implementation

4.1 *Parallelization Framework*

The recent improvements of medical imaging modalities produce images requiring an ever increasing storage space and computing power which has traditionally been available only on expensive specialized hardware configurations. Fortunately, the increase in performance of commodity hardware such as Pentium-based PCs as well as the possibility of connecting arrays of disks through EIDE or SCSI interfaces enables building much cheaper interactive 3D image storage and visualization devices. Striping a volumic data set onto a set of disks and reading from them in parallel offers an aggregate bandwidth compatible with the requirements of medical imaging. Furthermore, several PCs can be connected through commodity network switches such as Fast Ethernet and provide scalable com-

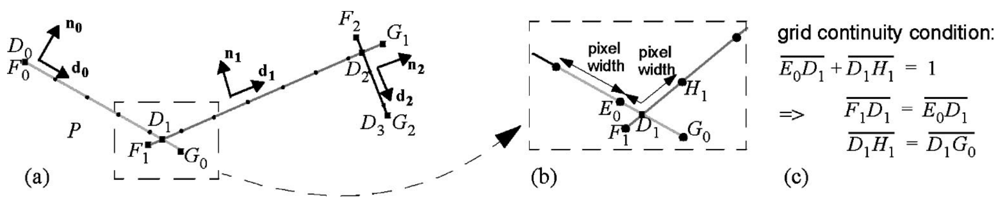


Fig. 5 Conditions for facet translation (projection on the directrix plane).

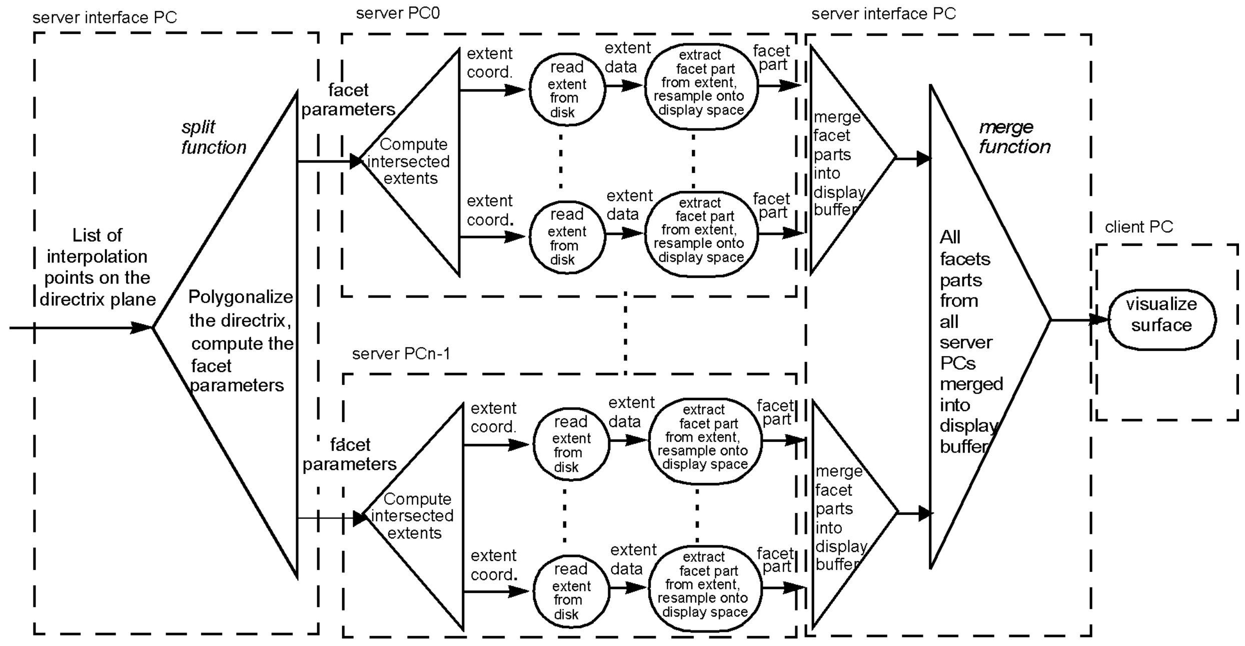

server interface PC
server PC0
server interface PC
List of interpolation points on the directrix plane
split function
Polygonalize the directrix, compute the facet parameters
facet parameters
Compute intersected extents
extent coord.
read extent from disk
extent data
extract facet part from extent, resample onto display space
facet part
merge facet parts into display buffer
merge function
All facets parts from all server PCs merged into display buffer
client PC
visualize surface
server PCn-1

Fig. 6 Schedule of the pipelined parallel surface extraction.

puting power for algorithms designed to run in parallel. To achieve the best performance on a multi-PC multidisk architecture, I/O intensive algorithms may hide disk transfer times by pipelining asynchronous disk accesses and computations.

In order to speedup the development of parallel applications and to specify parallel I/O and processing operations at a high level of abstraction, we use the computer-aided parallelization (CAP) tool.[17] This tool enables application programers to hierarchically specify the schedules of parallel operations and the flow of parameters and data between operations. Operations consist of sequential code performed by a single execution thread and characterized by input and output values. The input and output values of an operation are called tokens. In the context of this paper, tokens consist of image data (3D or 2D) and of additional application dependent parameters. Each parallel CAP construct consists of a split function splitting an input request into subrequests sent in a pipelined parallel manner to the operations of the available threads and of a merge function collecting the results. The merge function also acts as a synchronization means terminating its execution and passing its result to the higher level program after the arrival of all subresults.

The CAP schedule shown in Fig. 6 describes how the individual operations required for surface extraction can be executed as a series of pipelined parallel operations.

The server interface processor (one of the server's PCs) receives the list of interpolation points defining the directrix on the directrix plane, polygonalizes that directrix, computes the facet parameters (position, orientation, size), and sends them in parallel to the contributing server PCs. The server PCs compute for each facet the list of extents (subvolumes) which partly cover that facet (intersecting extents). Extent reading requests are asynchronously launched for intersecting extents located on the PCs local disks. The callback function associated to the asynchronous disk reading request forwards the received extent to a computing operation which extracts and resamples the desired facet part from the extent, and sends it to the server interface PC for merging into the display buffer. The server interface PC merges directly the facet parts at the correct position of the display buffer. Once all facet parts are merged, the display buffer is send to the client PC for visualization.

The surface extraction is carried out in a pipelined parallel manner. Within a single server PC, pipelining occurs by extracting facet parts from extents while new extents are read from disks. Since facet parts are merged on the server interface PC, there is a further pipelining step between facet part extraction and facet part merging into the display buffer. All server PCs contributing to the surface extraction run this pipeline in parallel.

The pipelining of operations is made possible by the CAP preprocessor which creates the code for queuing several requests in front of each operation.[18] Pipelining and request queuing allow to make a maximal use of the underlying hardware. Possible bottlenecks are either disk I/O, facet extraction running in the server PCs, facet merging running on the server's interface PC or facet transmission to the client PC. Increasing the number of disks per PC allows to increase the I/O bandwidth up to the bandwidth available on the PCs I/O channels (SCSI-2). Increasing the number of server PCs allows to increase both the available processing power and the disk I/O bandwidth.

4.2 *Improving the Parallelization Strategy*

The parallelization strategy described in the previous section is not optimal since, in the case of a large number of facet parts, the same extents are read several times. A large disk cache may, however, reduce the overhead of reading the same extents several times.

A more efficient strategy consists in first computing all extents intersected by the surface and grouping the facet

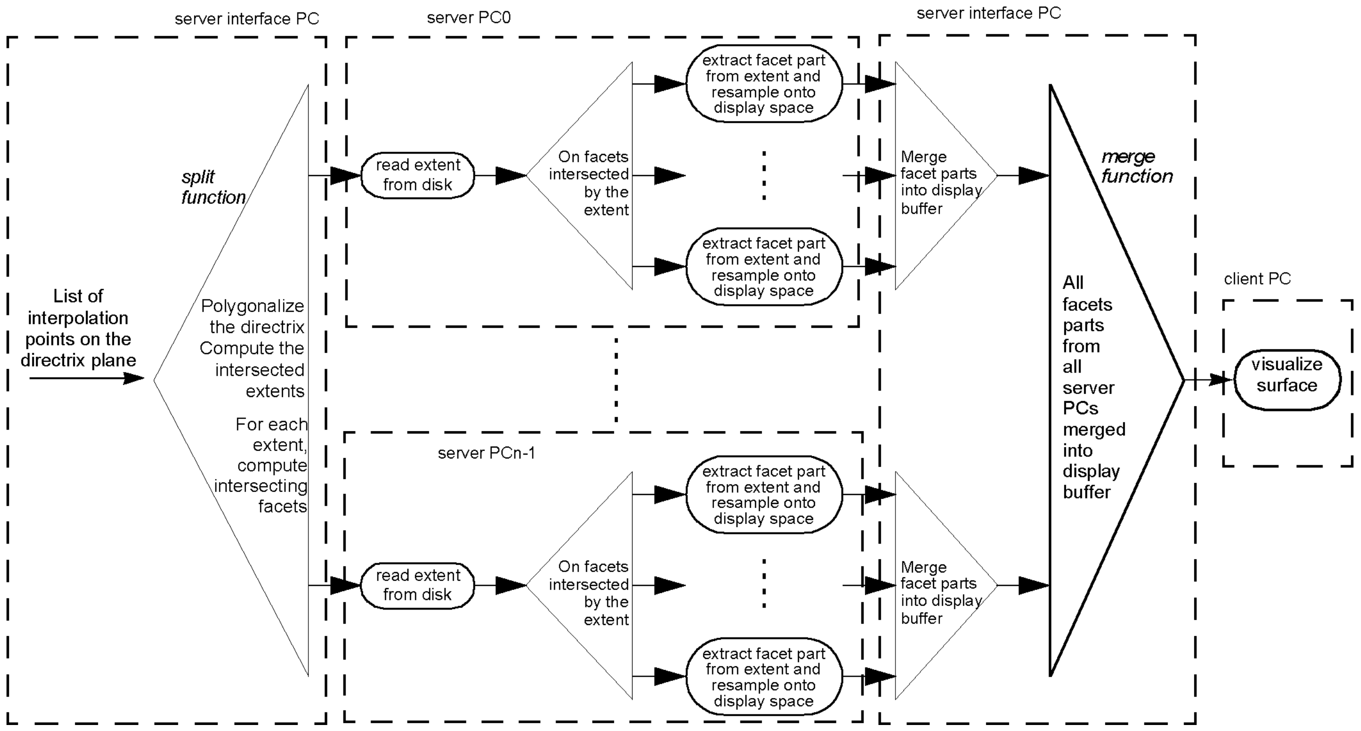

server interface PC
server PC0
server interface PC
List of interpolation points on the directrix plane
split function
Polygonalize the directrix
Compute the intersected extents
For each extent, compute intersecting facets
read extent from disk
On facets intersected by the extent
extract facet part from extent and resample onto display space
Merge facet parts into display buffer
merge function
All facets parts from all server PCs merged into display buffer
client PC
visualize surface
server PCn-1

Fig. 7 Extraction of facet parts extent by extent.

extraction requests on a per extent basis. Then, for each extent, the facet parts corresponding to each facet it intersects are extracted. With this method, each extent needs to be read only once, thus avoiding the overhead of multiple extent reads.

The computation of the extents hit by the surface is carried out facet by facet. For each intersected extent, a list of intersecting facets is built. Figure 7 shows the schedule corresponding to this parallelization strategy. Extents are read in parallel from several disks located on several server nodes. For each extent, corresponding facets intersected by that extent are computed and the facet parts inside the extent are extracted, resampled and sent to the interface PC to be merged into the display buffer. Disk access operations and computing operations are carried out in a pipelined parallel manner, as described in Sec. 4.1.

5 Performance Analysis

The architecture used for the performance experiments consists of one client node and one to four server nodes, all of them running the Windows NT 4.0 operating system. The client node is a Pentium II 333 MHz with 256 MB of memory. The server nodes are dual Pentium-Pro 200 MHz machines with 64 MB of memory. Each server node has ten SCSI-2 disks attached, distributed across three SCSI strings. The disks are of a variety of brands and models including IBM DPES 31080, IBM DCAS 32160, Conner CFP 1080S, Seagate ST51160N, and Seagate ST32155S. The slowest disks have a measured read data transfer throughput of 3.5 MBytes/s and a mean latency time (seek time plus rotational latency time) of 12.2 ms. Thus, when accessing extents made of 51 KByte blocks, the system is expected to achieve an effective throughput of 1.88 Mbytes/s per disk.[17] The client and all server nodes are interconnected through a 100 MHz Fast Ethernet switch.

Performance experiments have been carried out in order to assess the scalability of the surface extraction algorithm both when the number of disks and the number of server nodes increase. Surface extraction performance is also compared to planar slice extraction performance in order to evaluate its relative overhead.

Planar slice extraction reference times were obtained by measuring the average extraction time of four 512×512 pixel slices having different orientations. Four sample surfaces with an unfolded size of approximately 512×512 pixels were also chosen. These four surfaces have a size similar to the planar slices but vary in curvature as shown in Appendix A. Appendix A also gives the number of facets, the number of intersected extents, and the number of facet parts per surface.

A higher number of facets denotes a higher curvature of the surface since more planar subdivisions are necessary to approximate the corresponding real cylinder. The number of intersected extents corresponds to the actual amount of data that is to be read from the disks (elementary I/O operations). The number of facet parts denotes the number of planar facet extraction operations to be performed (elementary computation operations).

5.1 *Varying the Number of Disks*

In order to evaluate the scalability of the algorithm when increasing the number of disks, the first experiment comprises a client node and a single server node with a varying number of disks. Figure 8 shows the number of planar slices, respectively, surfaces, that are extracted per second as a function of the number of disks. For planar slice ex-

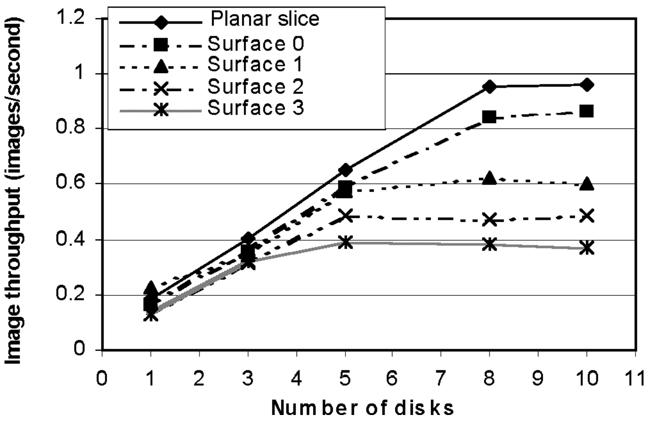


Fig. 8 Performance with a single server and a varying number of disks.

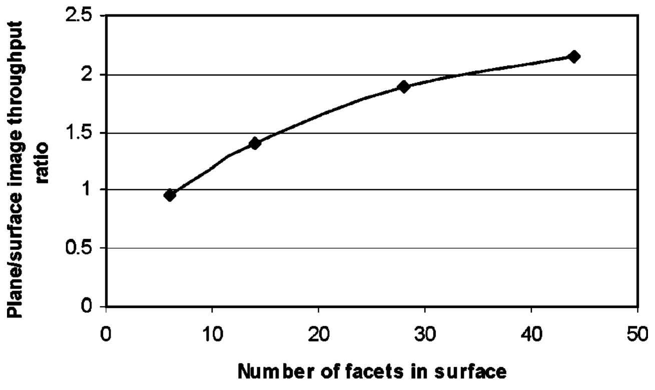


Fig. 10 Overhead of the surface extraction vs plane extraction.

traction, the peak performance is reached when the server contains eight disks delivering an aggregate I/O throughput of 17 MB/s. With less than eight disks the system is bound by the I/O throughput and performance scales linearly with the number of disks. With more than eight disks, slice extraction becomes computation bound.

For a constant surface size, when the curvature of the surface increases, the ratio of computation time and time taken to read data from disks increases considerably. A highly curved surface fits in a smaller data volume and requires more computing power for extracting more facets. Therefore, when curvature increases, the system becomes compute bound with fewer disks. For a small curvature (surface 0 made of only four facets) the surface extraction algorithm behaves as planar slice extraction. For higher curvatures (surfaces 1–3, made of respectively 14, 28, and 44 facets), the system quickly becomes compute-bound, i.e., instead of eight disks only five disks are enough to saturate the system (Fig. 8).

5.2 *Varying the Number of Server Nodes*

In order to evaluate the scalability of the algorithm when increasing the number of server nodes from one to four, the second experiment involved one client node and a varying number of server nodes (Fig. 9).

In this experiment, the server PCs were each connected to ten disks. The results show that surface extraction algorithm scales close to linearly. Extracting a surface with a low curvature such as surface 0 is carried out at the same speed as extracting a planar slice. Independently of their respective curvature, the parallelization of surface extraction yields similar speedups. The next bottleneck for planar and surface slice extraction (from up to five server nodes) resides in the server interface PC which needs to receive thousands of separate facet parts and assemble them into the display buffer. With latencies in the order of a millisecond, the TCP–IP Fast Ethernet protocol limits the number of separate packets that can be received by a single PC.[17] However, in order to reduce the number of transmitted packets, one may carry out the facet merging operation in two separate steps. Within each server node, facet parts may be first partially merged into an intermediate buffer. The resulting partially merged facet parts may then be transmitted to the server interface PC for final merging.

Figure 10 shows in a compute-bound configuration (1 server node with ten disks) the overhead of surface extraction in respect to planar slice extraction as a function of the estimated surface curvature expressed in number of facets.

Surfaces with up to ten facets incur no noticeable performance penalty in comparison to planar slices. Surfaces with high curvatures take up to twice as much time to extract compared with planar slices of the same size. Since high curvature surfaces are segmented into a much larger number of elementary facet parts, surface extraction requires smaller loops and more logic instead of the few large loops that are necessary for planar slice extraction.

6 Conclusions

We have presented a new approach for the parallel out of core extraction of ruled surfaces from large 3D medical images. Surface extraction extends the traditional oblique slicing operation commonly used by radiologists to examine 3D medical scans. Curved surfaces represented as spline-based cylinders are extracted and unfolded in order to provide a flat nondistorted 2D surface representation. The application runs in parallel on a cluster of PCs, each PC being connected with up to ten SCSI disks.

Surface extraction is easily affordable, since it is only two times slower than planar oblique slicing. With up to four server nodes offering sufficient I/O bandwidth (5–8 disks), surface extraction scales linearly with the number of server nodes.

The surface extraction service has been interfaced to the visible human web server (http://visiblehuman.epfl.ch) and

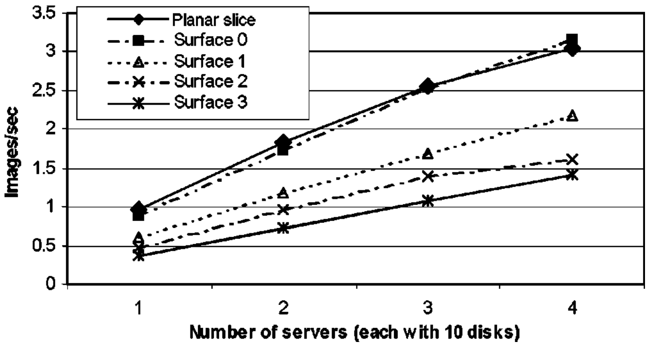


Fig. 9 Performance as a function of the number of server nodes.

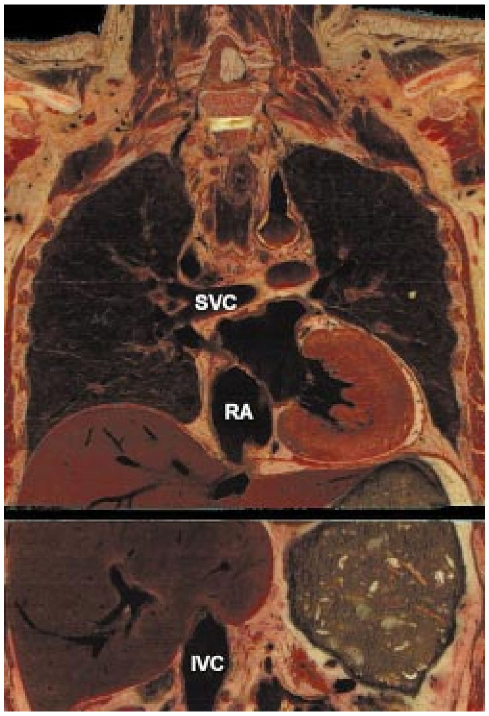


Fig. 12 Coronal slice illustrating the localizations of superior vena cava, right atrium, and inferior vena cava.

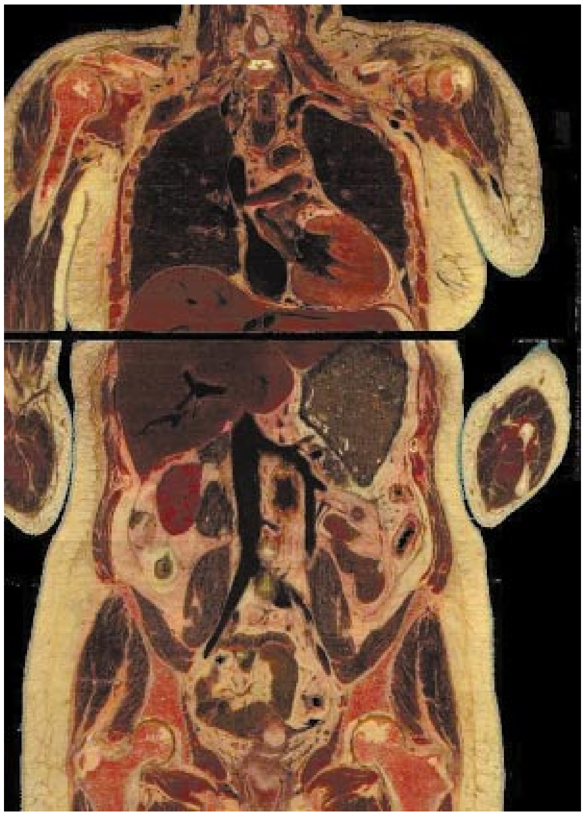


Fig. 13 Oblique slice illustrating the bifurcation of the inferior vena cava into the illiac veins.

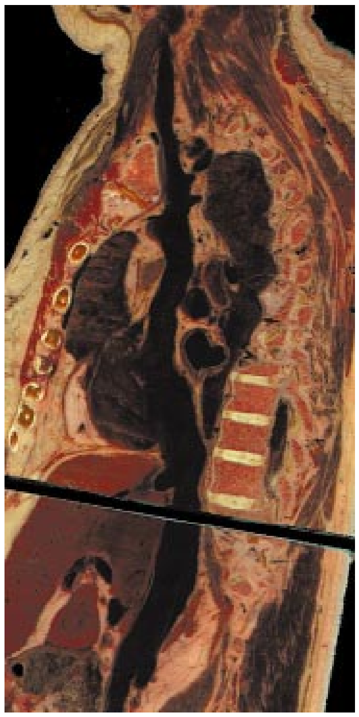


Fig. 15 Resulting unfolded quasisagittal surface showing the continuity of superior vena cava, right atrium, and inferior vena cava.

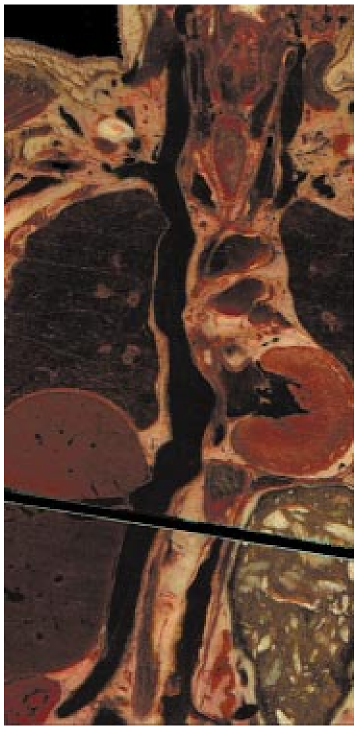


Fig. 16 Quasicoronal surface obtained by a trajectory specified on an oblique quasi sagittal slice, also showing the continuity of superior vena cava, right atrium, and inferior vena cava.

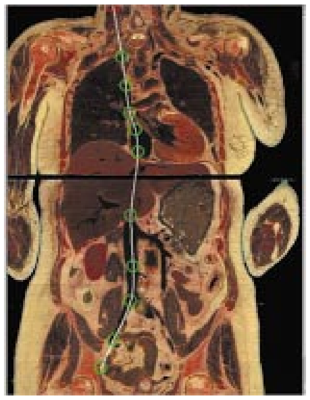
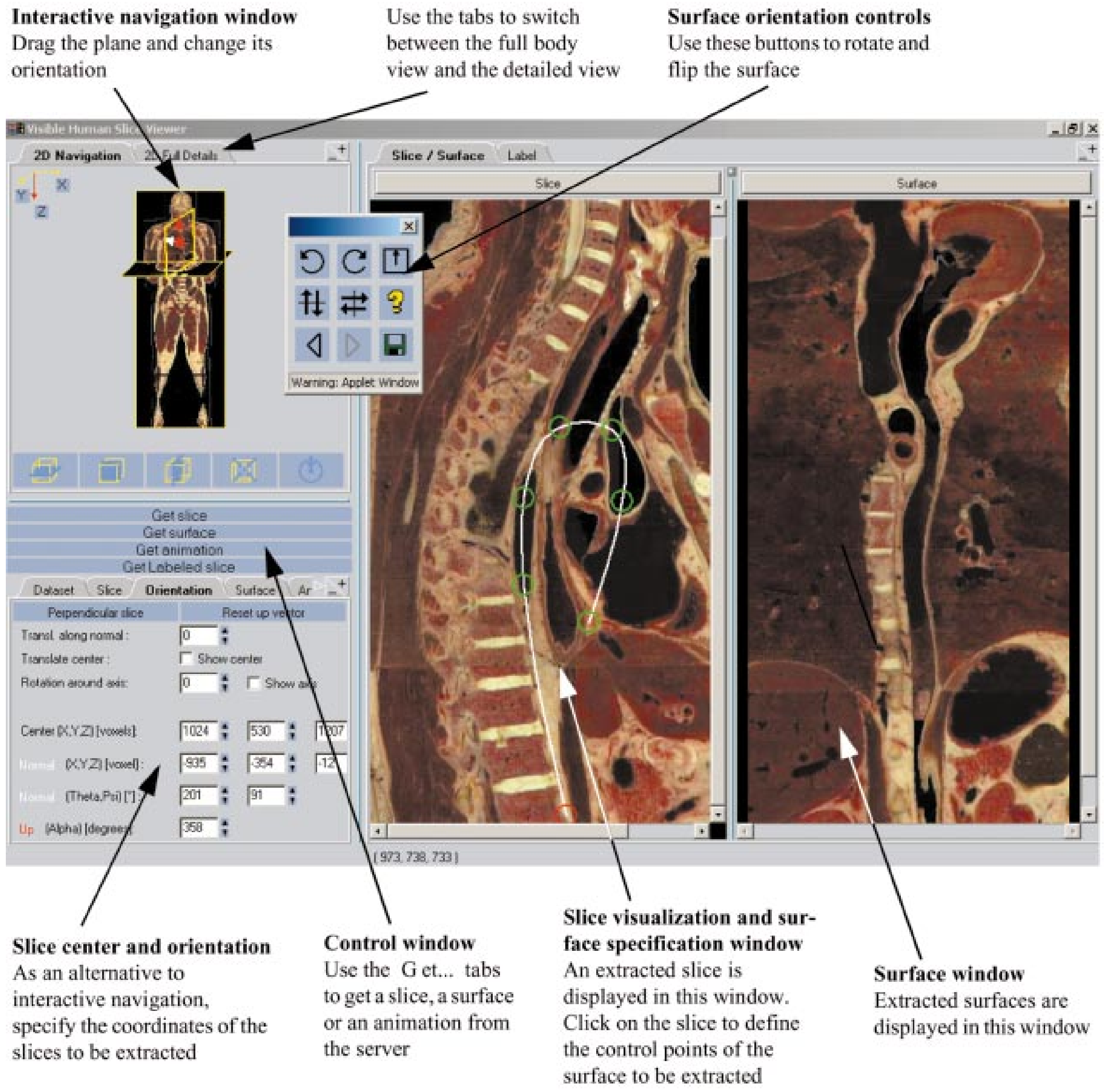


Fig. 14 Specification of a trajectory along the superior vena cava, right atrium, and inferior vena cava



Interactive navigation window
Drag the plane and change its orientation

Use the tabs to switch between the full body view and the detailed view

Surface orientation controls
Use these buttons to rotate and flip the surface

Transl. along normal : 0
Translate center : Show center
Rotation around axis: 0 Show axis
Center (X,Y,Z) [voxels]: 1024 530 1207
Normal (X,Y,Z) [voxel] : .935 -.354 -.12
Normal (Theta,Psi) [°] : 201 91
Up (Alpha) [degrees]: 358

[973, 738, 733]

Slice center and orientation
As an alternative to interactive navigation, specify the coordinates of the slices to be extracted

Control window
Use the G et... tabs to get a slice, a surface or an animation from the server

Slice visualization and surface specification window
An extracted slice is displayed in this window. Click on the slice to define the control points of the surface to be extracted

Surface window
Extracted surfaces are displayed in this window

Fig. 17 Screenshot of the user interface (Java applet) available for the extraction of surfaces.

is available as one of the server's on-line services. Surface unfolding is mainly used by professionals for visualizing curved anatomic structures. Integrated into tomographic equipment, surface extraction and unfolding may also prove to be useful for the staging of tumors, i.e., for locating the extension of tumors in tomographic volume images.

Appendix A: Sample Sets for Surface Extraction Performance Measurements

The images in Fig. 11 show the directrices of the surfaces used in the performance measurements. They were interactively specified using the applet of the visible human web server (http://visiblehuman.epfl.ch).

Appendix B: Using Surfaces for Visualizing the Continuity of Anatomic Structures

(written in collaboration with Peter Ratiu, Harvard Medical School)
As an example, suppose we wish to visualize the continuity of the superior vena cava (SVC), right atrium (RA), and inferior vena cava (IVC). This fact is clinically important in cardiac catheterizations. One must use the marking on the catheter, because in the absence of an anatomical obstacle, the catheter inserted through the external jugular vein will run past the right atrium into the inferior vena cava. Axial, coronal (Fig. 12) and sagittal slices do not allow to visualize the continuity of SVC, RA, and IVC. An oblique slice such as the one in Fig. 13 allows to visualize the branches of the IVC and the bifurcation into the common illiac veins, but the continuity of the IVC is obscured by the liver.

Let us specify on an oblique slice a trajectory for a surface through the SVC, RA, and IVC as shown in Fig. 14. The surface obtained (quasisagittal surface) shows the clear continuity of the three structures (Fig. 15). A similar surface can be obtained by specifying a trajectory on a slice in sagittal orientation, resulting in a quasicoronal surface (Fig. 16).

At Harvard Medical School, it is planned to use the surface specification and extraction feature offered by the visible human web server in many clinical and anatomical applications.

Appendix C: The Visible Human Web Server User Interface

Fig. 17 shows a screenshot of the user interface (Java applet) available for the extraction of surfaces (see http://visiblehuman.epfl.ch/).

To extract a surface, the user needs first to extract a slice. A slice is specified either in the interactive navigation window or by entering its location and orientation in the control window (select the ''orientation'' tab). By clicking on the ''get slice'' bar in the control window, the specified slice is requested from the server and displayed. To specify a surface perpendicular to the slice, the user clicks on the slice in order to define the control points of the surface directrix. Then by clicking on the ''get surface'' bar in the control window, the request for extracting the corresponding surface is sent to the server which extracts the surface and sends it back to the client applet for display.

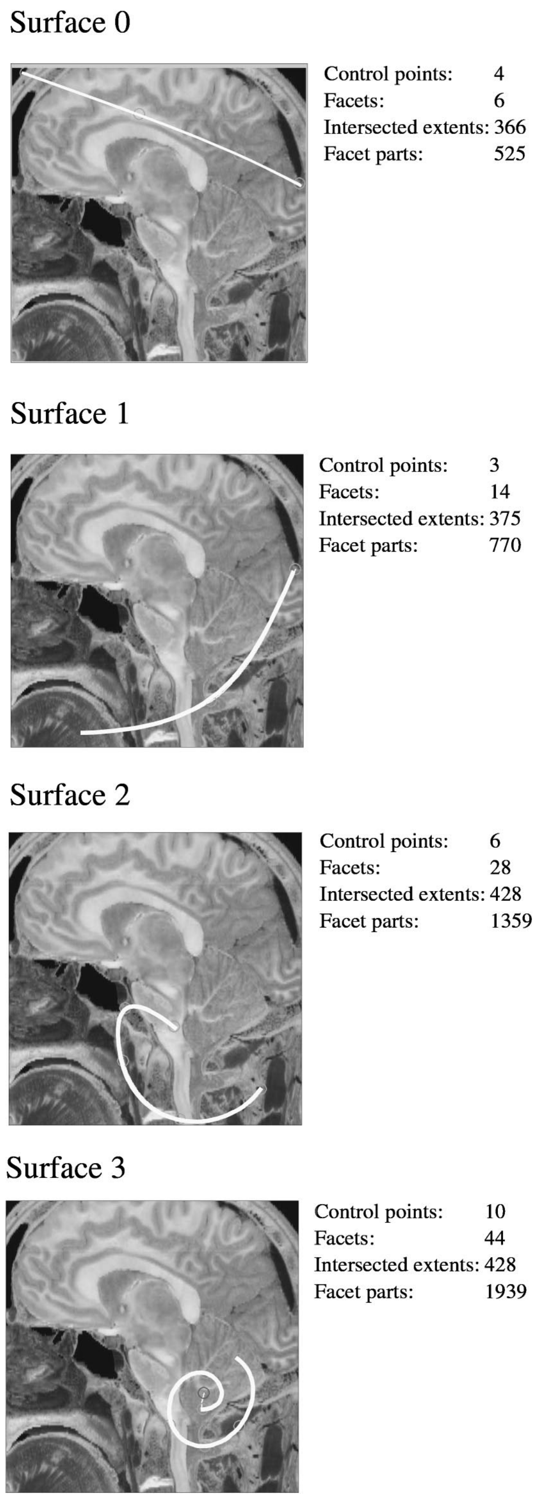


Fig. 11 Directrices of the surfaces used in the performance measurements.

At the present time, it is not possible to save the description of the directrix. However, such a feature could be easily added, since the visible human server incorporates a database allowing to save client specific information.[6]

References

1. D. M. Kramer, L. Kaufman, R. J. Guzman, and C. Hawryszko, ''A general algorithm for oblique image reconstruction,'' *IEEE Comput. Graphics Appl.* **10**, 62–65 (1990).
2. M. J. Ackerman, ''The visible human project,'' *Proc. IEEE* **86**(3), 504–511 (1998).
3. R. D. Hersch *et al.*, ''The visible human slice web server: A first assessment,'' *Proc. SPIE* **3964**, 253–25 (2000).
4. J. C. Bessaud and R. D. Hersch, ''The visible human slice sequence animation web server,'' *Proc. SPIE* **3964**, 253–258 (2000), see also The Third Visible Human Project Conference Proceedings, Bethesda, Maryland, Oct. 2000, http://www.nlm.nih.gov/research/visible/vhpconf2000/.
5. S. Gerlach and R. D. Hersch, ''A real-time navigator for the visible human,'' *IEEE Internet Computing* **6**(2), 27–33 (2002).
6. F. Evesque, S. Gerlach, and R. D. Hersch, *J. Visual. Comput. Animation* **13**, 43–52 (2002).
7. J. D. Foley *et al.*, *Computer Graphics Principles and Practice*, 2nd ed., pp. 1015–1018, Addison-Wesley, (New York 1990).
8. A. Kaufman and E. Shimony, ''3D scan-conversion algorithms for voxel-based graphics,'' Proc. 1986 Workshop on Interactive 3D Graphics, pp. 45–75, 23–24 October 1986, Chapel Hill, NC.
9. J. Bresenham, ''Pixel processing fundamentals,'' *IEEE Comput. Graphics Appl.* **16**(1), 74–82 (1996).
10. J.-P. Reveillès and D. Richard, ''Back and forth between continuous and discrete for the working computer scientist,'' *Ann. Math. Artif. Intell.* **16**, 89–152 (1996).
11. I. Debled-Rennesson and J.-P. Reveillès, ''A new approach to digital planes,'' *Proc. SPIE* **2356**, 12–21 (1994).
12. O. Figueiredo, ''Advances in discrete geometry applied to the extraction of planes and surfaces from 3D volumes,'' Ph.D. thesis No. 1944, 1999, Ecole Polytechnique Fédérale de Lausanne (EPFL), http://diwww.epfl.ch/w31sp/publications/discretegeo/.
13. I. Debled-Rennesson, ''Etude et reconnaissance des droites et plans discrets,'' Thèse de Doctorat no 2234, Université Louis Pasteur, Strasbourg, France, Dec. 1995.
14. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, Academic, New York (2001).
15. O. Figueiredo, J. P. Reveillès, and R. D. Hersch, ''Digitization of Bezier Curves and Patches using Discrete Geometry,'' *Discrete Geometry for Computer Imagery, Proc. of the 8th International Conference DGCI'99, Marne-la-Vallee, France, March 1999*, G. Bertrand, M. Couprie, and L. Perroton, Eds., LNCS 1568, pp. 388–398, Springer, Berlin (1999).
16. V. Messerli, B. Gennart, and R. D. Hersch, ''Performances of the PS2 parallel storage and processing system for tomographic image visualization,'' *Proc. 1997 Int'l Conference on Parallel and Distributed Systems, Seoul, Korea, Dec. 1997*, pp. 514–522, IEEE, New York (1997).
17. V. Messerli, O. Figueiredo, B. Gennart, and R. D. Hersch, ''Parallelizing I/O intensive image access and processing applications,'' *IEEE Concurrency* **7**(2), 28–37 (1999).
18. B. Gennart and R. D. Hersch, ''Computer-aided synthesis of parallel image processing applications,'' *Proc. SPIE* **3817**, 48–61 (1999).

Oscar Figueiredo heads the computer science research and teaching unit at the Ecole Supérieure de Chimie Physique Electronique (CPE Lyon) in France. He received his PhD in computer science in 1999 from the Ecole Polytechnique Federale de Lausanne (EPFL). He holds an engineering degree with a specialization in computer engineering from CPE Lyon and obtained a Master Diploma in computer graphics in 1994. His research interests include theoretical aspects of discrete geometry, their practical applications in computer graphics, high-performance parallel algorithms for visualization, and internet application protocols at large.

Roger D. Hersch is professor of computer science and head of the Peripheral Systems Laboratory at the Ecole Polytechnique Fédérale de Lausanne. He received his engineering and PhD degrees respectively from ETH Zurich in 1975 and from EPFL in 1985. He has published over 100 scientific papers, is the editor of four books, and is inventor or coinventor in several patent applications. The labs research is focused on high-performance server applications (imaging servers, Web servers, PC clusters, etc.) and novel imaging techniques (color prediction, color reproduction, artistic imaging, anticounterfeiting). He directs the visible human web server project, which offers advanced visualization services for exploring human anatomy (see http://visiblehuman.epfl.ch).