CEIG - Spanish Computer Graphics Conference (2015)
Jorge Lopez-Moreno and Mateu Sbert (Editors)

Surfel Octrees: A New Scheme for Interactive Inspection of Anatomy Atlases in Client-Server Applications

J. Surinyac[1], P. Brunet[2]

[1]Department of Digital Technologies and Information, University of Vic, Vic, Spain
[2]Department of Software, Polytechnic University of Catalonia, Barcelona, Spain

Abstract
Nowadays, an increasing interest on tele-medicine and tele-diagnostic solutions can be observed, with client/server architectures for remote inspection of volume image-based medical data which are becoming more and more popular. The use of portable devices is gradually spreading due to their portability and easy maintenance. In this paper, we present an efficient data model for segmented volume models based on a hierarchical data structure of surfels per anatomical structure. Surfel Octrees are compact enough for transmission through networks with limited bandwidth, and provide good visual quality in the client devices at a limited footprint. Anatomy atlases are represented as octree forests, supporting local interaction in the client device and selection of groups of medical organs. After presenting the octree generation and interaction algorithms, we present several examples and discuss the interest of the proposed approach in low-end devices such as mobiles and tablets.

Keywords: *client-server architectures, segmented volume models, interactive inspection, surfels, octree, anatomy information, anatomic atlas.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Interactive inspection of segmented volume models and anatomy atlases in client-server architectures is becoming more and more popular, now that 3D medical volume models are continuously improving in quality and detail.

Present systems for the acquisition of medical images are providing better and better resolution images which end up in bigger amounts of data, even in scenarios with learning or training purpouses. However, transmission time for the volumetric information and low performance hardware properties make quite complex to build efficient visualization systems on these devices.

Server-client model transmission requires good compression techniques, being as lossless as possible and having a high compression rate, to obtain a reduced data flow through the network. So advanced new algorithms for compression and progressive transmission are required, because of user interest on displaying the models and interacting with them in low-end, portable and mobile clients.

In this paper we present an efficient data model for segmented volume models based on a hierarchical data structure of surfels per anatomical structure or organ (*organ* in the following). The volume models we use are already segmented, process which is beyond the scope of this work. Surfel Octrees are compact enough for transmission through networks with limited bandwidth, and provide good visual quality in the client devices at a limited footprint. Anatomy atlases are represented as octree forests, supporting local interaction in the client device and selection of groups of medical organs. After presenting the octree generation and interaction algorithms, we present several examples and discuss the interest of the proposed approach in low-end devices. The main contributions of our approach are:

- A client/server architecture for the visualization of medical atlases in low-end devices.
- A hierarchical structure (Surfel Octree) to store segmented medical models.
- A very compact scheme for storing and transmitting Surfel Octrees.

We present our approach in Sections 3 and 4, after review-

DOI: 10.2312/ceig.20151201

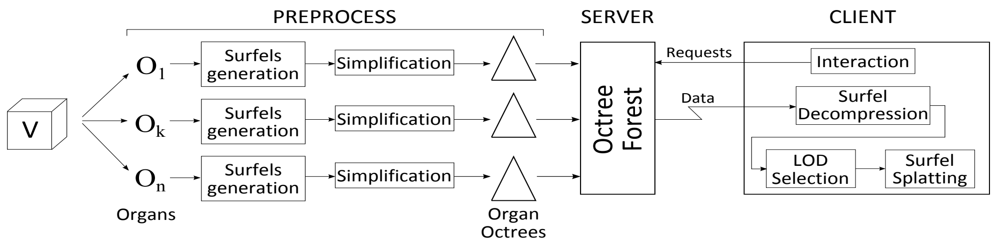

Figure 1: *Overview of the approach. Per organ preprocess and Client-Server interaction.*

ing the previous work in Section 2. Results are presented and discussed in Section 5.

2. Previous work

Tele-medicine and tele-diagnostic are elements of growing interest during the last years. Interactive anatomy atlases have also appeared as new methods of medicine teaching. Such elements are usually based in portable low-end devices. Tablets and mobiles are increasingly equipped with larger memories and faster GPUs. But the improvement of capture devices is continuously increasing the amount of medical data to be processed and it is still far away from the limits of today's devices. So applications can take advantage of hierarchical models for the visualization of medical data and client-server architectures to supply the necessary data from the server's repository. Compression methods of the octrees are also used for a fast transmission of data and to avoid a bottleneck in the network.

Compression methods

Client-server architectures and progressive transmission algorithms started almost two decades ago with mesh-specific methods. Hoppe [Hop96] proposed an algorithm for the transmission of triangle meshes as a sequence of progressive smaller meshes.

Gobbetti and Marton [GM04] worked on client-server architectures for huge point data clouds, by using a *k*-d tree hierarchical data structure. In their approach each tree node contains a region of space with a similar amount of equally distributed samples that are complemented by the samples in upper levels. The rendering traverses the tree from the top and continues until a predefined level/density is reached.

P. Callahan et al. [CBPS06] implemented a client-server architecture for progressive volume rendering of unstructured data. They used the server as a data repository and clients as rendering machines that accumulate the incoming geometry and display it in a progressively improving way. The server processed the model into and octree, by this way traversing it by depth ranges from front to back. When the client requested a certain packet, the server culled the geometry outside the current depth range and sent visible volumes to the client, also incrementing depth ranges for futures steps.

For a more complete survey on volume rendering architectures, client-server schemes and compression techniques, we refer to the work of Balsa et al, [BGI*13].

In [SB13] Surinyac and Brunet presented a framework for the transmission of medical models and a region-based inspection. Each organ is represented by a surfel octree so the region of interest consists in a populated forest of octrees. Each node contains a surfel which is transmitted by the server and stored in the client in 11 bytes. Our scheme uses more efficient methods of transmission of octrees and compression of surfels in order to improve the response of client devices.

Compression techniques consist on decreasing the size of volumetric data by applying a specific encoding algorithm [BGI*13]. In some approaches compression is combined with a brick or a hierarchical partition of the original volume to facilitate the compression process and a level-of-detail rendering/transmission. The compact representation can be lossless or lossy. Medical applications often require a lossless compression when rendering models, even at coarser resolutions.

Vector quantification is a lossy compression method [NH92]. The volume dataset is represented as indexes into a small codebook of representative blocks. This scheme is suitable for a CPU-based ray-casting render. Schneider and Westermann [SW03] proposed a Laplacian pyramid vector quantification approach that allows relatively fast volume decompression and render on the GPU. However this method does not support GPU linear filtering capabilities and its rendering cost increases when using high zoom factors.

Wavelet transforms offer considerable compression ratios in homogeneous regions of a volume while they conserve the detail in the non-uniform ones [Mur92] [CNB13]. Ihm and Park [IP99] proposed an effective 3D 16^3 block based compression/decompression wavelet scheme. Guthe et al. [GS01] proposed an alternative hierarchical wavelet representation which allows to store very large volume datasets in compact but lossy data structures.

Lazaro et al [CNB13] proposed a compact, progressive transmission that is very efficient to send and reconstruct the octree structure. In their scheme isosurfaces of transfer functions are computed, storing only edge/grey nodes in a Gradient Octree. Data information and octree structure are separated and the information needed for a node is whether it is a grey or a nil node. That way only consumes one bit of storage per node, giving a very compact compression of an octree. This implementation is used in our approach to obtain better rates of data transmission.

Mobile visualization systems

Low performance of low-end client devices make quite complex to build efficient visualization systems on these devices.

Lamberti et al. [LZS*03] implemented a technique in which storage and computation resources are provided by a server system while mobile devices are used only as client front ends. User interaction in the mobile device is codified and sent to the server which, in turn, produces the corresponding image in its frame buffer. The image is sent via TCP to the client which shows it in its display. With low bandwidth, the image is compressed to jpeg or gzip, but requires time to decodification in the client part that decreases the frame rate. The image can also be sent in half resolution when the model is being rotated.

To visualize at the same time interior and exterior parts of a volume Monclús et al. [MDNV09] propose a metaphor named Virtual Magic Lantern which uses a 3D pointing device to define a visualization cone. The volume outside the cone is rendered using the original Transfer Function whereas the inside one can use a different function, for instance rendering only specific organs or parts of the model. Although this metaphor was designed for a raycasting algorithm, it can be adapted to other visualization methods. Moser and Weikpof [MW08] introduced an interactive technique for volume rendering on mobile devices that adopts the 2D texture slicing approach. Noguera et al. [NJOS12] propose an algorithm that overcomes the 3D texture limitation of mobile devices and achieves interactive frame rates by caching the geometry of the slices in a vertex buffer object. The method proposed by Díaz et al. [DMNV12] produces expressive medical illustrations. Segmented medical datasets can be cut by a clipping plane defined by the user and then selected organs can be extruded out of the plane by dragging them with the mouse.

Octree-based, GPU-friendly ray-casting algorithms have been proposed in [CNE09], [LK10] and others. In [CNE09] the octree leaves represent bricks containing 16^3 or 32^3 voxels, while octree nodes in [LK10] contain 64-bit child descriptors, both being too verbose for progressive transmission purposes. More recently, Suter et al. [SIM*11] have proposed a multiscale volume representation based on a tensor approximation within a GPU-accelerated rendering framework. It can be better than wavelets at capturing non-axis aligned features at different scales.

Gobetti et al. [GIM12] have proposed an algorithm for rendering gigantic volume models. In their client-server approach, the volume is decomposed into a multiresolution hierarchy of bricks. Each brick is further subdivided into smaller blocks, which are compactly described by sparse linear combinations of prototype blocks stored in an overcomplete dictionary. The dictionary is learned, using limited computational and memory resources, by applying the K-SVD algorithm to a re-weighted non-uniformly sampled subset of the input volume.

Faut and Kwan-Liu Ma propose in [FM07] propose a method which uses volumetric compression and decompression plus rendering in the GPU for large datasets. This method is based in a block classification scheme that allows real-time rendering.

Suter et al. [SMP13] presented a hierarchical tensor approximation volume visualization approach. They create a new global TA hierarchy for multiresolution DVR and multiscale feature visualization.

In [AGIM10] Agus et al. implemented a method for simplifying blocks of voxels from segmented volume data, providing a multiresolution scheme. Their method allows interactive manipulation of the transfer function and a GPU ray-casting framework.

Last but not least, Movania and Feng [MF12] presented a single-pass ray caster and a 3D texture slicer rendering algorithm for the WebGL platform, capable of handle dynamic transfer functions in mobile devices.

3. Surfel Octrees: Definition and Generation Algorithm

Definition

Surfel Octrees are a compact representation of segmented volumes. They recursively subdivide the space into Void and Grey nodes and store this subdivision in a compact way. Void nodes correspond to cubic regions of the space that are not intersecting the boundary surface of the organ being represented, while Grey nodes cubic regions do intersect this boundary surface. By approximating the organ boundary surface within each Grey node by a planar surfel, Surfel Octrees become multiresolution compact representations of the corresponding organs. Surfel Octrees are generated in two steps. In the first one, the organ bounding cubic box is recursively subdivided and the octree structure with Void and Grey nodes is created in a top-down way. Subdivision stops when the tree node size equals the voxel size in the dataset. Then, in a second step, surfels in the leaf Grey octree nodes are computed by estimating and smoothing normal vectors in a similar way to [SB13] and Grey octree nodes are computed by an efficient bottom-up simplification process, also based on the Organ Octree generation algorithm, [SB13].

Our algorithm starts from a segmented binary volume model V without density information. In our case, every voxel in V contains an index to its segmented organ and the local color attribute of the organ tissue. A preprocess in the server computes a Surfel Octree for each organ in the region to be studied. The server stores the set of all Surfel organ octrees (Octree Forest from now on) and sends parts of it to the clients based on their interactive requests. Figure 1 presents the overall approach of our proposal. The Octree Forest is rather compact and can be stored in the server main memory. It is represented as an array of surfel octrees, each of them being indexed by its organ label.

Transmission to the clients is performed on demand and in a progressive way. Client requests are driven by their interaction events. Initially, a coarse representation of all organs in the Octree Forest is sent to all clients, to guarantee a minimum level of local interaction, as discussed in Section 5. Afterwards, any client can ask for a refinement of a certain organ, and the next octree levels for this organ in the server Octree Forest will be sent to this client. Clients render their volume models by octree traversal and surfel splatting. Progressive transmission and interaction in the clients is detailed in the next Section.

Efficient Compact Representation

The main problem with previous approaches like Organ Octrees ([SB13]) is twofold: individual surfels require a 11 Bytes encoding, and the application must also take into account the storage overhead due to the octree structure. Our present proposal addresses these two problems in a novel and efficient way, as presented in the next paragraphs.

Regarding surfel representations, we have concluded that a very efficient and compact choice is to use a Conn ected-Cubes plane parameterization [ACB*04] together with projective surfels. We compute a set of texture images $T_1 .. T_n$ from the high resolution surfels of the segmented volume V by rendering all maximum resolution surfels from n initial directions, see Figure 2. The rendering direction corresponding to any T_i is always the normal direction of the T_i plane. Some surfels like s_1 will be captured by more than one image (T_1 and T_2 in this case), but some other surfels like s_2 in Figure 2-(a) will not be captured due to occlusions. We detect these situations by performing a second render of the surfels with false color, and we add a few more directions and texture images T_k in a way that surfels like s_2 become unoccluded, Figure 2-(b). By computing surfel importances in every image T_k as dot products between surfel normals and render directions, we are able to compute the best texture image for every surfel. At the end of the preprocess, surfels encode their plane equation and a one-Byte index k to their texture T_k. During render, surfel appearances are computed by projecting T_k onto them in the texture normal direction. This is valid for surfels at any level in the octree.

One of the key ingredients of the Surfel Octree representation is the use of an extremely compact encoding for the

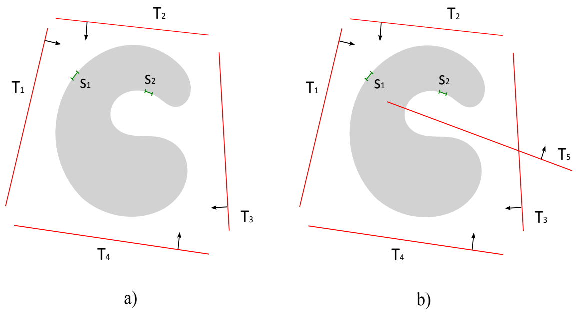

Figure 2: *a) Surfel s_1 is captured by both T_1 and T_2 but is projected to T_1. s_2 is occluded. A rendering with false color is performed to detect such surfels. b) New texture image T_5 is generated to capture s_2.*

octree structure, which has been inspired in [CNB13]. We encode the octree structure as a single bit array, with one Bit per octree node, assigning a "0" to Void nodes and a "1" to Grey nodes. Octrees are encoded as a sequence of octree levels, each level k being a block of bits that represent all octree nodes at that level. The number of bits of the $(k+1)$-block is obviously $8 * g_k$ where g_k is the number of Grey (or "1") nodes in the previous k-block. Sets of 8 bits corresponding to the sons of all Grey nodes in the k-block are encoded in the $(k+1)$-block in the same order their parents had in the k-block.

Moreover, we use a very compact encoding of individual surfels in the octree, with 5 bytes per surfel. The first 33 bits encode the oriented plane of the surfel while the last 7 bits contain an index to its texture T_k. To maximize the encoding accuracy in our plane parameterization, we locally relate this parameterization to the cube C of the octree node, by using four alternate vertices of the surfel's octree node to define a local frame and the vertices of a supporting base tetrahedron. The size and location of the cube C is automatically obtained from the octree traversal information during the rendering process. Oriented surfel planes are encoded in 33 bits by using an efficient Connected-Cubes plane parameterization, [ACB*04]. We use 3 bits to encode the CC cube $W_0^+ ... W_3^-$ and 30 bits to encode the three coordinates of the oriented surfel plane in its CC-cube.

In short, we succeed in representing the octree structure with one bit per node and surfel data with 5 Bytes per surfel. Our representation is pointerless and consists of two bit streams, see Figure 3. The first bit stream (node list) represents all octree nodes in a breadth-first traversal, with one bit per node. In this bit stream, Void nodes are encoded as "0" and Grey nodes are encoded as "1". The second bit stream (data list) represents all octree surfels, with a 5 Bytes/surfel encoding as discussed. The number of surfels in the data list equals the number of "1" bits in the node list, and the k-element in the node list corresponds to the k-Grey node in

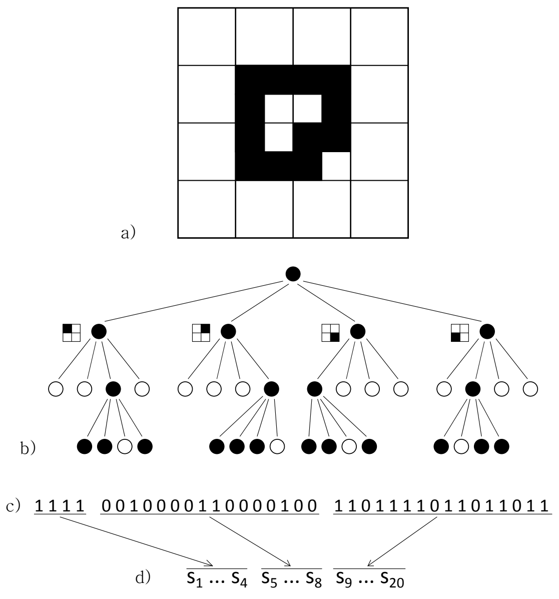
a)
b)
c) 1111 0010000110000100 1101111011011011
d) $s_1 \dots s_4$ $s_5 \dots s_8$ $s_9 \dots s_{20}$

Figure 3: *Surfel Octree representation, 2D example. In (a), shaded regions contain parts of the organ boundary and are represented by surfels. In (b), octree structure and ordering of the brother sons during spacial subdivisions. In (c), bit stream encoding the octree structure. Separate levels are underlined, and the root node is not represented. A total ol 20 surfels are represented in the data stream, (d).*

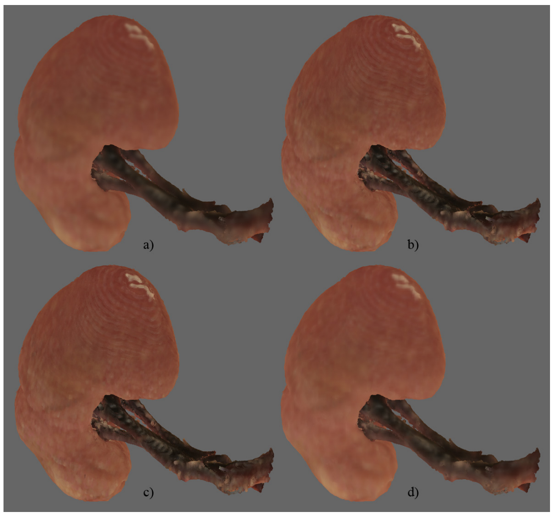

Figure 4: *Kidney and renal vein shown in the three approaches. a) Original model before transmission. b) Explicit representation. c) Color-Gradient. d) Projective textures.*

the node list (in other words, the k occurrence of a bit "1" in the node list).

Figure 4 compares the color-gradient approach in [SB13] with an explicit representation of the colors in the six vertices of an hexagon inscribed in the surfel and with our present approach. As discussed in Section 5, using projective textures produces good visual results with less than 50% of the memory requirements, when compared with the approach in [SB13].

On the other hand, and regarding the octree structure, our proposal derives from the Gradient Octrees data structure [CNB13] and is able to encode the Organ Octree structures with only one bit per node. This is detailed in the next Section.

By taking advantage of the octree structure, we are able to encode surfels in a very compact way. Surfel geometries are simply represented as oriented planes in the cubes of their octree nodes. We have chosen the Connected Cubes Plane Parameterization (CCPP from now on) from [ABC*04], to encode surfel planes with respect to their cubic domain. The parameterization is based on four weights w_k which define a linear scalar field, the weights being attributes of the vertices of the base tetrahedron defined by alternate vertices of the cubic domain. The represented plane is the zero-isosurface of the linear scalar field. The plane can be represented by a line (the plane line) through the origin in the four dimensional w_0 .. w_3 space, as the field isosurface is invariant to any weights common scaling. To remove redundancy, the CCPP plane parameterization represents planes by the intersection between their plane lines and the 3D faces of a hypercube centered at the origin. This is a set of eight 3D cubes, corresponding to the maximum and minimum values of the four weights. The CCPP parameterization is faithful (it uniformly represents all possible planes in the domain) and 2-concise, see [ABC*04]. The 3D position $s.P$ of the surfel is implicit in its CCPP representation. We can always compute it from the surfel plane $s.pl$ by defining it as the centroid of the intersections between the surfel plane and the edges of the boundary of its cubic domain.

4. Atlas Transmission and Remote Interaction

The total amount of memory required for the representation of the structure of Surfel Octrees can be bounded under the assumption that the fractal dimension of the organ boundaries is 2 (in other words, we assume that the organ surfaces are not fractals). Let us note by v_k and g_k the number of Void and Grey nodes at the octree level k, $k = 0..n$. The number of surfels s_n at the highest resolution is obviously $s_n = g_n$. We know that $v_k + g_k = 8 * g_{k-1}$ for all k and that g_k is similar to $4 * g_{k-1}$ for the deepest levels of the octree, the first equation coming from well-known octree properties and the second one from our assumption that organ surfaces are not fractals, [LT89] (box counting for the estimation of the fractal dimension D is based on the well-known equation $g_k = 2^D \cdot g_{k-1}$). By using these two equations and adding the contribution of the different octree levels, we obtain that $v_k \simeq s_k$ for k values close to n. Then, the total number n_n of octree nodes can be approximated as $n_n = 2*s_n + s_n/2 + s_n/8 + s_n/32 + \ldots < 2.688*s_n$. Taking into account that our representation requires one bit per node to represent the octree structure, we can conclude that, in well-formed organs, the overhead of the proposed octree structure representation is less than three bits per surfel at the highest resolution. In other words, the octree of an organ like the Calcaneus in Table 1 having 90058 surfels at the deepest resolution (level n) requires 90058*3=270174 bits which is equivalent to 34 KBytes.

Surfel render in the client devices is based on a preorder traversal of the Surfel Octrees of the Forest. The octree traversal keeps track of the current depth in the octree and the path from the root, the 3D location and size of the cube of any visited node being easily computed from the depth and path information. The geometry of individual surfels is created on the fly by decoding the Connected-Cubes plane information, transforming this local plane to the World coordinate system and rendering an hexagon that circumscribes the polygon of intersection between the surface plane and the cube of the octree node. A specific fragment shader creates a circular surfel by removing hexagon fragments outside the circle and assigns colors to fragments by projecting the surfel indexed texture.

We start interactive sessions by defining an inspection region, obtaining its list of organs and sending the organs list and the octree structure of the Surfel Octrees of the corresponding Forest to the clients. Our results, as shown in the next Section, confirm that the amount of involved information is small enough for interaction purposes. The user selects a group of organs to inspect from the organs list, and the client sends requests for each of these organs to the server, which in turn sends the set of textures for each organ and a set of surfels corresponding to a low resolution octree level. In our present implementation, and assuming that a certain Surfel Octree has depth equal to n, we initially send all surfels at octree level $n-2$. Requests for increased detail (surfels at levels $n-1$ or n) can be issued by the client at any moment during the interactive inspection session. Observe that textures and octree structures are *LOD*-independent: they they must be sent only once per organ to the clients. Memory requirements of textures and surfels at different octree levels are discussed in the next Section.

During interactive sessions, users can select groups of objects, add or remove some of them at any time, and can decide to inspect the whole volume model at a low resolution (with $(n-2)$-level surfels) or ask for a better resolution in certain specific organs. Since the whole structure of the Surfel Octrees is available at the client side, rotation and zooming operations can be autonomously performed in the client without any further transmission from the server. If a region of the model needs to be detailed, surfels from levels $n-1$ or n can be displayed on demand. Additional feaures include the use of a Precision Inspection Sphere (see next Section) which only displays the maximum resolution inside the interest region, and planar sections that are shown with high-resolution pictures of the corresponding sections of the initial volume data V.

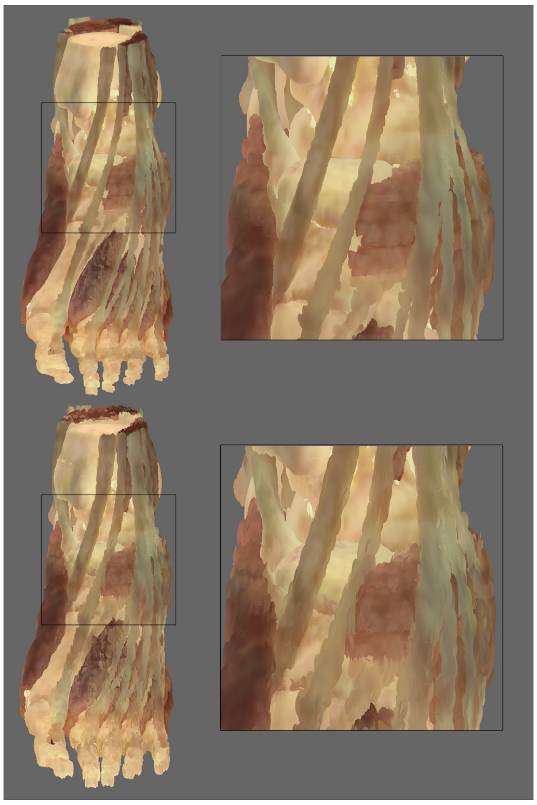
Figure 5: *Top row: Foot at the most detailed level (n) and zoom of a region. Bottom row: The same but at coarser level ($n-2$). Individual surfels can be seen at the outline of the ankle at the coarser resolution (level $n-2$).*

5. Results and Discussion

We present examples on three different regions of the Visible Human model. In Figure 6 there are several views of the left foot. In this case, the total number of processed organs is 96, resulting in an octree forest of 96 components. This figure illustrates the client capacity of interactively adding or removing organs. Users can also inspect cuts of the organs.

We can appreciate the similarities between the reconstructed organ and a real one as is shown in Figure 7. In the original dataset the organ is still inside the body, so the

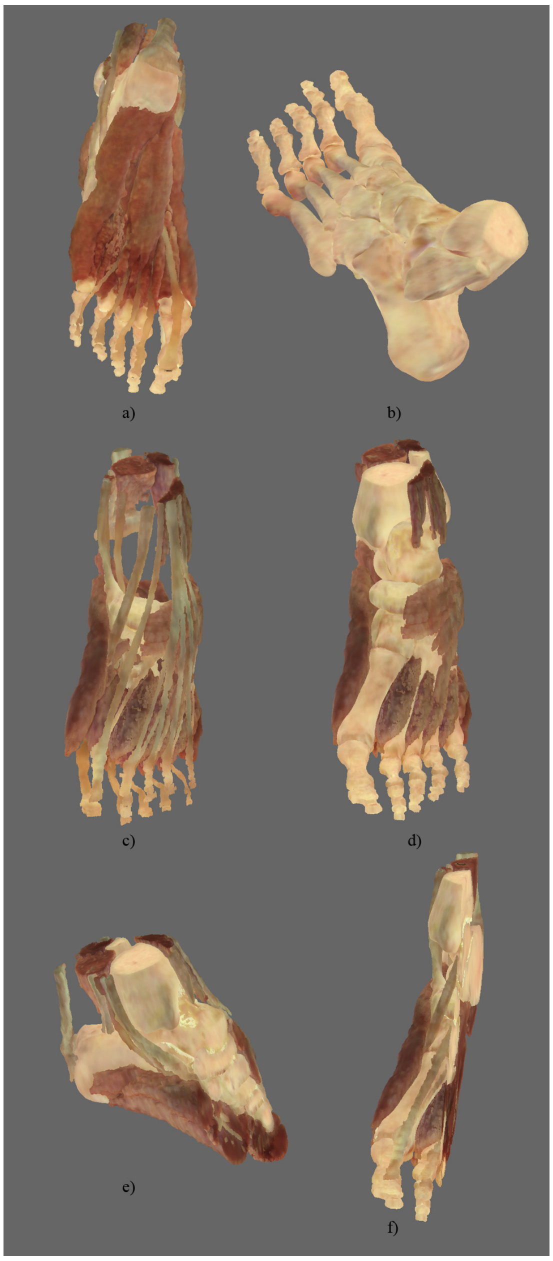

Figure 6: *a) Plantar view. b) Only bones. c) Muscles removed. d) Only bones and muscles. e) Medial cut. f) Sagital cut.*

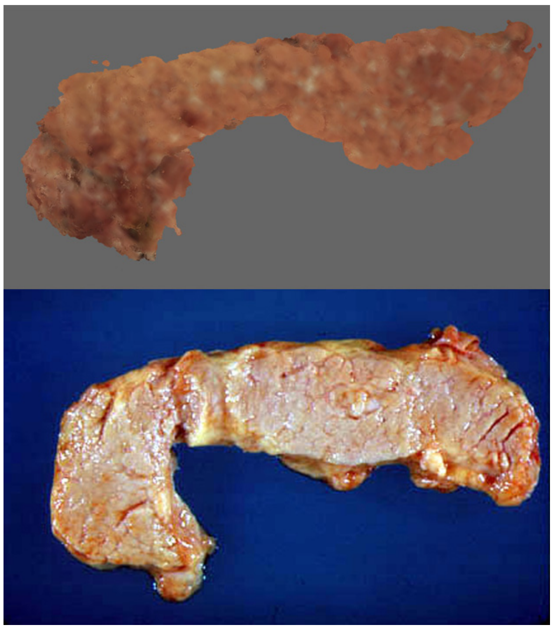

Figure 7: *Reconstructed and real pancreas.*

shape of the real, surgically extracted organ can show some differences to the reconstructed one.

The level of detail shown must not be the same among all organs. Each organ can be displayed specifically at any required precision level. It is also possible to display simultaneously two levels of the same organ. To take advantage of this ability the client implements a Precision Inspecting Sphere, consisting in a sphere which acts as a "resolution magnifying lens". The surfels inside the sphere are rendered at the finest level, while the rest of them are rendered at a coarse level. In Figure 8 all the bones of the hand are displayed at level $n-2$ except for part of two metacarpals which are displayed at level n.

The complete foot consists in 96 organs. In table 1 there are relevant information of 25 of them, ordered in decreasing number of surfels. Second and third column are the number of surfels in levels n and $n-2$ of the Surfel Octree of the given organ. The total of the complete model is 1686034 and 95465 surfels, respectively. Fourth column is the maximum edge (in millimeters) of the bounding box of the organ; it defines the size of the octree that contains it. In fifth column there is the depth of the corresponding octree.

The data transmitted from the server to the client is divided in three parts: textures, surfels and octree structure. A calculation of the total memory transmitted for the 96 organs of the foot model is made next.

- **Textures**. There are 20 textures per organ, each one of 128x128 texels of 3 bytes of color, and 96 organs. That

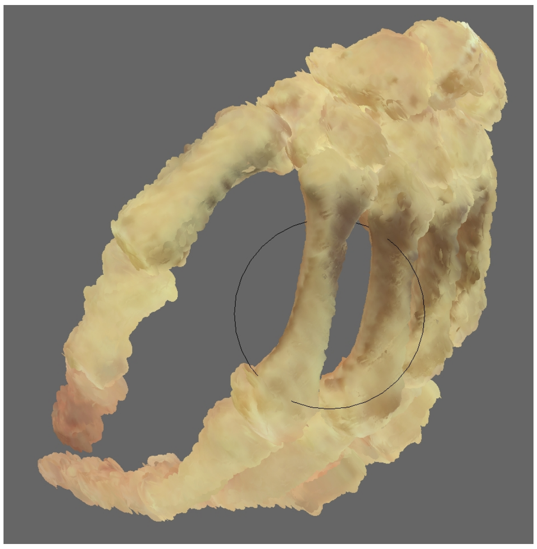

Figure 8: *Precision Inspection Sphere. Inside the sphere surfels are represented at level n whereas outside it they are shown at level $n-2$.*

makes 90 MBytes, 960 KBytes per organ. Textures are transmitted only once per session.

- **Surfels**. As explained in section 3 the data of a surfel is encoded in 5 bytes, giving 8430170 Bytes (8.04 MBytes) for level n and 477325 Bytes (0.46 MBytes) for level $n-2$.
- **Octree structure**. Following section 4, the number of nodes of the octree is limited by 3 times the number of surfels in level n, which is 1686034 nodes. And given each node is codified in 1 bit, the overall octree hierarchy of the complete forest is stored in 618 Kbytes.

This scheme allows a very compact representation of Surfel Octrees. And the transmission of textures can be enhanced using lossless compression formats of images, like png. With an average compression ration of one to four, the 96 textures can be stored in just 22.5 MBytes, 0.24 MBytes per organ.

In Table 2 the compression ratio of some representative organs from Table 1 is described. First column shows the number of voxels in the model, the second one is the size in bytes of the Surfel Octree and the last column is the bandwidth usage measured in bits/voxel. The compression ratio depends on the shape of organs. Notice that compression values are mainly below 5 bits per voxel. The value of 10 belongs to an small organ with few voxels.

The construction of the Surfel Octree is done in preprocessing time. The complete foot took 265.35 seconds to complete. The rendering of the client achieves 60 fps when all organs are displayed at level $n-2$ and 13 fps when all

Organ	**n**	**n-2**	**size**	**depth**
Calcaneus	90058	5519	83	8
Abductor_Digiti_Minini	70457	4136	98	9
Flexor_Digitorum_Tendon	44798	2404	141	9
Metatarsal_1st	43061	2565	58	8
Talus_Cartilage	40730	2033	57	8
Interosseous_Dorsal_4th	31234	1869	50	8
Flexor_Hallucis_Tendon	28958	1592	152	9
Cuboid	25759	1531	36	7
Lumbricals	25557	1454	56	8
Metatarsal_5th	24830	1504	61	8
Tendo_Calcaneus	23024	1206	31	8
Interosseous_Dorsal_1st	20116	1200	32	7
Extensor_Hallucis_Tendon	18968	997	110	9
Interosseous_Plantar_3rd	14590	842	43	8
Cuneiform_Intermediate	12464	754	28	7
Phalanx_Proximal_3rd	6339	373	26	7
Metatarsal_Cartil_5th	3647	222	89	9
Phalanx_Middle_3rd	2763	168	13	6
Phalanx_Distal_3rd	2117	120	10	5
Tibial_Nerve	1911	107	7	7
Phalanx_Distal_5th	1466	69	10	5
Plantar_Nerve_Lateral	629	35	6	5
Sesamoidal_Cartilage	336	16	5	4
Phalanx_Cartil_Prox_2nd	125	4	5	5
Phalanx_Cartil_Mid_4th	26	2	2	4

Table 1: *Information about of some organs (25 out of 96) of the foot image. Columns are: name, number of surfels at level n (finest) and $n-2$ (more coarse), size of maximum side of bounding box (in mm.) and depth of the octree.*

Organ	**voxels**	**oct.size**	**ratio**
Calcaneus	2311218	717926	2.485
Abductor_Digiti_Minimi	4789325	456032	0.762
Flexor_Digitorum_Tendon	30684420	351452	0.092
Metatarsal_1st	1600452	341378	1.710
Cuboid	318934	204386	5.127
Interosseous_Plantar_3rd	308826	115334	2.988
Phalanx_Cartil_Mid_4th	192	248	10.333

Table 2: *Compression ratios for several organs. Columns: number of voxels in the model, Surfel Octree size in bytes, compression ratio bits/voxel.*

are at level n. In situations where some organs are shown at level n and the rest at $n-2$ (as when the Precision Inspecting Sphere is in use) the frame rate is about 30, depending on the balance of number of surfels rendered at each level.

Additional material (including videos and user evaluation results with a PC server and a tablet client) can be downloaded from `http://www.cs.upc.edu/~jsurinach/CEIG15/AndroidResults.html`.

One limitation of our method is the requirement of a static and not adaptable segmentation. Any modification of the Transfer Function would require a new execution of the preprocess.

The same computer has been used to run both the preprocessing and then the rendering. This is a machine with an Intel Core i7 CPU with 4 cores at 2GHz, 4 GBytes of RAM and a GPU NVidia GeForce GT550M that has 2 GBytes of RAM.

6. Conclusions and Future Work

In this paper, we have presented an efficient data model for the interactive inspection of anatomy Atlases in client/server environments. Our 3D data model uses binary segmented medical volume models, being based on a hierarchical data structure of surfels per organ, namely Surfel Octrees. Surfel Octrees are efficient and compact, supporting real-time transmission through networks with limited bandwidths. Anatomy Atlases are represented as octree forests, supporting local interaction in the client devices and selection of groups of medical organs. Individual surfels are encoded in 5 Bytes, the octree structure requiring 1 bit per octree node.

A new version of our client/server application supporting efficient visualizations of high-resolution volume planar sections is under development, and it can hopefully be completed for the final version of the paper. In the next future, we plan to improve the client/server platform to make it usable in practical applications, and to perform a user evaluation of the system with a PC server and a tablet client.

7. Acknowledgements

This work has been partially supported by the Project TIN2014-52211-C2-1-R of the Spanish Ministerio de Economía y Competitividad.

References

[ABC*04] Andújar C., Brunet P., Chica A., Navazo I., Rossignac J., Vinacua A.: Computing maximal tiles and application to impostor-based simplification. *Computer Graphics Forum 23*, 3 (2004), 401–410. 5

[ACB*04] Andújar C., Chica A., Brunet P., Navazo I., Rossignac J., Vinacua A.: The connected-cubes plane parameterization. *Polytechnic University of Catalonia* (2004). httpAddr//www.lsi.upc.edu/p̃ere/Planes. 4

[AGIM10] Agus M., Gobbetti E., Iglesias J., Marton F.: Split-voxel: A simple discontinuity-preserving voxel representation for volume rendering. *IEEE/EG International Symposium on Volume Graphics* (2010). 3

[BGI*13] Balsa M., Gobbetti E., Iglesias J., Makhinya M., Marton F., Pajarola R., Suter S.: A survey of compressed gpu-based direct volume rendering state of the art report. In *Eurographics STARS* (2013). httpAddr//www.lsi.upc.edu/p̃ere/Planes. 2

[CBPS06] Callahan S. P., Bavoil L., Pascucci V., Silva C. T.: Progressive volume rendering of large unstructured grids. *IEEE Transactions on Visualization and Computer Graphics* (2006), 1307–1314. 2

[CNB13] Campoalegre L., Navazo I., Brunet P.: Gradient octrees: A new scheme for remote interactive exploration of volume models. *Computer-Aided Design and Computer Graphics (CAD/Graphics), International Conference on* (2013), 306–313. 2, 3, 4, 5

[CNE09] Crassin C., Neyret F., Eisemann E.: Gigavoxels: ray-guided streaming for efficient and detailed voxel rendering. *Proc. of the Symposium on Interactive 3D Graphics and Games* (2009), 15–22. 3

[DMNV12] Díaz J., Monclús V., Navazo I., Vàzquez P.: Adaptive cross-sections of anatomical models. *Computer Graphics Forum (Proc. Pacific Graphics), Forum 31(7)* (2012), 2155–2164. 3

[FM07] Fout N., Ma K.-L.: Transform coding for hardware-accelerated volume rendering. *Visualization and Computer Graphics, IEEE Transactions on 13*, 6 (2007), 1600–1607. 3

[GIM12] Gobbetti E., Iglesias J., Marton F.: Covra: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. *EuroVis 2012, Computer Graphics Forum 31*, 3 (2012), 1315–1324. 3

[GM04] Gobetti E., Marton F.: Layered point clouds. a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics 28*, 6 (December 2004), 815–826. 2

[GS01] Guthe S., Strasser W.: Real-time decompression and visualization of animated volume data. *VIS 01: Proceedings of the conference on Visualization 01* (2001), 349–356. 2

[Hop96] Hoppe H.: Progressive meshes. *ACM SIGGRAPH Proceedings* (1996), 99–108. 2

[IP99] Ihm I., Park S.: Wavelet-based 3d compression scheme for interactive visualization of very large volume data. *Computer Graphics Forum 18* (1999), 3–15. 2

[LK10] Laine S., Karras T.: Efficient sparse voxel octrees, proc. of the acm siggraph symposium on interactive 3d. *Graphics and Games* (2010), 55–63. 3

[LT89] Liebovitch L., Tibor T.: A fast algorithm to determine fractal dimensions by box counting. *Physics Letters A 141*, 8-9 (1989), 386–390. doi:10.1016/0375-9601(89)90854-2. 5

[LZS*03] Lamberti F., Zunino C., Sanna A., Fiume A., Maniezzo M.: An accelerated remote graphics architecture for pdas. *Proc. of the 8th International Conference on 3D Web Technology. ACM* (2003), 55–61. 3

[MDNV09] Monclús V., Díaz J., Navazo I., Vàzquez P.: The virtual magic lantern: An interaction metaphor for enhanced medical data inspection. *VRST* (2009), 119–122. 3

[MF12] Mobania M., Feng L.: Mobile visualization of biomedical volume datasets. *Journal of Internet Technology and Secured Transactions (JITST) 1*, 1 (2012). 3

[Mur92] MURAKI S.: Approximation and rendering of volume data using wavelet transforms. *Proc. of the 3rd Conference on Visualization* (1992), 21–28. 2

[MW08] MOSER M., WEISKOPF D.: Interactive volume rendering on mobile devices. In *Workshop on Vision, Modelling and Visualization VMV'08* (2008). 3

[NH92] NING P., HESSELINK L.: Vector quantization for volume rendering. *Proc. of the workshop on Volume Visualization* (1992), 69–74. 2

[NJOS12] NOGUERA J., JIMÉNEZ J., OGÁYAR C. ., SEGURA R.: Volume rendering strategies on mobile devices. *International Conference on Computer Graphics Theory and Applications (GRAPP)* (2012). 3

[SB13] SURINYAC J., BRUNET P.: A client-server architecture for the interactive inspection of segmented volume models. *Proceedings of CEIG - Spanish Computer Graphics Conference 2013* (2013), 99–108. 2, 3, 4, 5

[SIM*11] SUTER S., IGLESIAS J., MARTON F., AGUS M., ELSENER A., ZOLLIKOFER C., GOPI M., GOBBETTI E., PAJAROLA R.: Interactive multiscale tensor reconstruction for multiresolution volume visualization. *IEEE TVCG 17*, 12 (2011), 2135–2143. 3

[SMP13] SUTER S., MAKHYNIA M., PAJAROLA R.: Tamresh: Tensor approximation multiresolution hierarchy for interactive volume visualization. *Eurograpics Conference on Visualization (Euro Vis) 32*, 3 (2013), 151–160. 3

[SW03] SCHNEIDER J., WESTERMANN R.: Compression domain volume rendering. *Proc. of the 14th IEEE Visualization 2003* (2003), 39. 2