The Visible Human Slice Sequence Animation Web Server

Jean-Christophe Bessaud, Roger D. Hersch[a]
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

ABSTRACT

Since June 1998, EPFL's Visible Human Slice Server (http://visiblehuman.epfl.ch) allows to extract arbitrarily oriented and positioned slices. More than 300,000 slices are extracted each year. In order to give a 3D view of anatomic structures, a new service has been added for extracting slice animations along a user-defined trajectory. This service is useful both for research and teaching purposes (http://visiblehuman.epfl.ch/animation/). Extracting slices of animations at any desired position and orientation from the Visible Human volume (Visible Man or Woman) requires both high throughput and much processing power. The I/O disk bandwidth can be increased by accessing more than one disk at the same time, i.e. by striping data across several disks and by carrying out parallel asynchronous disk accesses. Since processing operations such as slice and animation extraction are compute-intensive, they require the program execution to be carried out in parallel on several computers. In the present contribution, we describe the new slice sequence animation service as well as the approach taken for parallelizing this service on a multi-PC multi-disk Web server.

Keywords: Internet, Java applet, MPEG-1, parallel I/O, parallel file striping, Slice sequence animation, Visible Human.

1. INTRODUCTION

Web-based slicing services for extracting slices from the Visible Human dataset exist since 1995[1], but they authorize only to extract slices perpendicular to the main axes. Since June 1998, EPFL's Visible Human Slice Server (http://visiblehuman.epfl.ch)[2] allows to extract arbitrarily oriented and positioned slices. More than 300,000 slices are extracted per year. A surface extracting service has been added and allows to define, extract and flatten curved surfaces. But extracted slices and surfaces are only 2D images which do not reveal the real 3D anatomic structures. To give an impression of 3D, we created a service allowing to extract successive slices along a user-defined trajectory (see http://visiblehuman.epfl.ch/animation/). It is a kind of video on demand service (VOD), where Web clients specify the trajectory of the desired slice sequence animation across the body of the Visible Human.

To offer a slice sequence animation service, we had to meet the following challenges:

1. The user interface must be intuitive and allow to define the trajectory of the animation.
2. Server side extraction of slices and synthesis of a compressed slice animation must be carried out quickly (few seconds). The progress of the animation download must be shown.
3. The currently displayed animation slice of the slice sequence animation should be shown on the user-defined trajectory.
4. In order to avoid blocking new incoming requests, the server should allow the simultaneous computation of several slice sequence animations.
5. It should be possible to save the slice sequence animation on the local client computer despite security restrictions.
6. The animation should be compressed as an MPEG-1 movie, supported by most browsers.

The creation of an animation requires reading of up to a few Gigabytes from disks, extracting many successive oblique slices and compressing the resulting sequence of slices into an MPEG-1 stream. These operations may take several minutes and need therefore to be parallelized. The parallel program should be executed on a multi-PC cluster and disk accesses

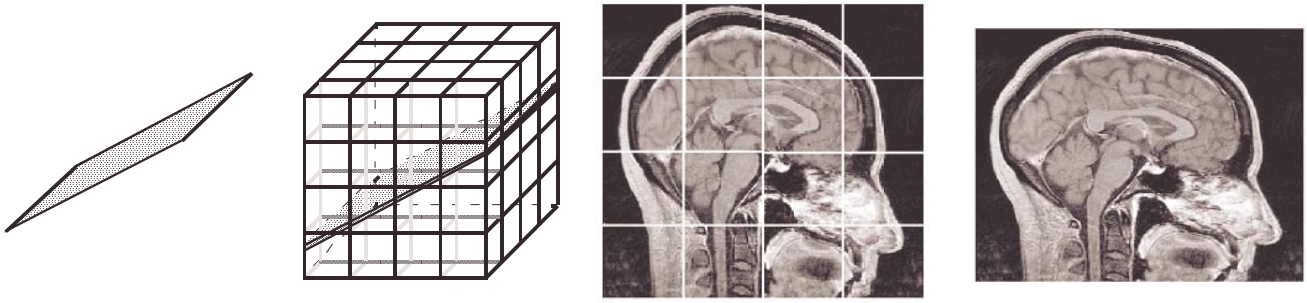
a. {jean-christophe.bessaud,rd.hersch}@epfl.ch, http://visiblehuman.epfl.ch

3rd VisibleHuman Project Conf., National Institutes of Health, Bethesda, Oct. 5-6, 2000 (USA),
http://www.nlm.nih.gov/research/visible/vhpconf2000/MAIN.HTM,
extended version published in Conf. on Internet Imaging II, SPIE Vol. 4311, 2001, 341-348,

carried out asynchronously and in parallel. Section 2 describes the slice web server and its performances. Section 3 describes the new slice sequence animation service. The client user interface is presented in section 4 and the design of the parallel Web server is described in section 5.

2. THE SLICE SERVER ARCHITECTURE

The Slice Web Server[2] is composed of one Bi-Pentium PC and 16 disks. It is a scaled down version of a powerful parallel server comprising 5 Bi-Pentium Pro PC's and 60 disks[3]. All PC's are interconnected through a 100 Mbits/s Fast Ethernet switch. The parallel server program was created thanks to the computer-aided parallelization framework (CAP)[3], which takes over the task of creating a multi-threaded pipelined parallel program from a high-level parallel program description.
For enabling parallel access to its data, the Visible Human 3D volume[4] is segmented into sub-volumes (extents) of size 32x32x17 RBG voxels (i.e. 51 Kbytes), which are striped over the disks residing on the server. In order to extract an image slice from the 3D image, the extents intersecting the slice are read on the disks and the slice parts contained in these volumic extents are extracted and resampled in the display gird (Fig. 1).



Extraction slice specification | Extraction of the digital slice from the 3D image sub-volumes (extents) | Extracted slice parts | Merging the extracted and resampled slice parts into one displayable slice

Fig. 1. Extraction of slice parts from volumic file extents

The Visible Human slice web server application consists in a server interface residing on the Web server PC and server processes running on the server's PC's. The Web server interface interprets the slice location and the orientation parameters defined by the user and determines the volumic extents (sub-volumes) which need to be accessed. It sends the extent reading and image slice part extraction requests to the concerned server (server whose disks contain the required extents).

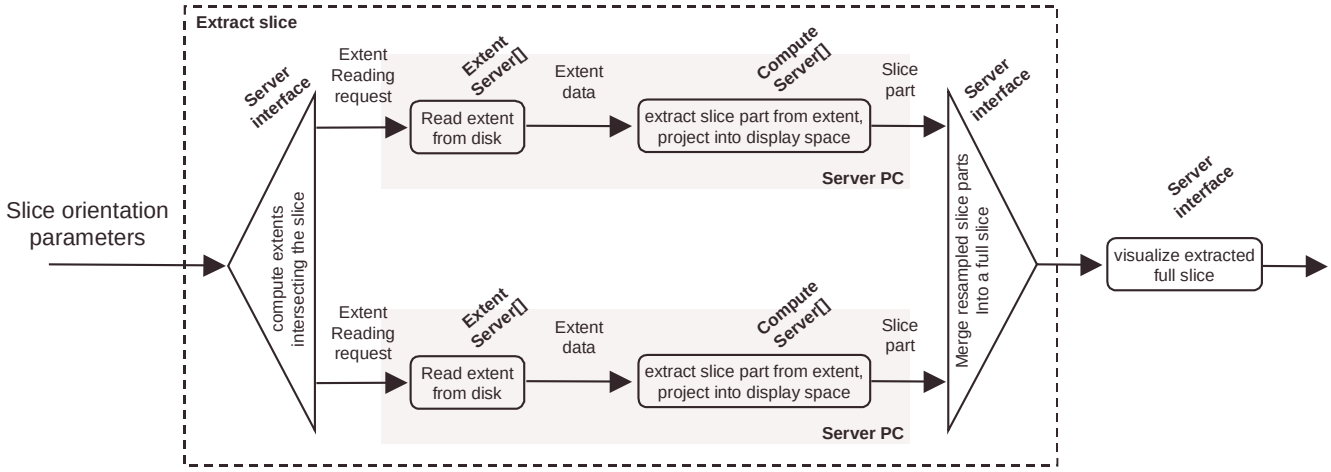


Fig. 2. Graphical representation of the pipelined parallel slice extraction and visualization

These servers execute the requests and transfer the resulting slice parts to the Web server PC which assembles them into the final displayable image slice. This final image is compressed in JPEG format and sent to the corresponding client. The diagram shown in Fig. 2 describes the parallel slice server application.

Measurements made on the parallel slice server[2] have shown that up to 4.8 slices/s (512x512 colour slices) either the parallel server disks (60 disks) or the parallel server processing power (5 Bi-Pentium Pro PCs, 200 MHz) are the bottleneck. Beyond 4.8 slices/s, the server interface PC assembling the slice parts is not able to receive from the Fast Ethernet more than 7.8 Megabytes/s of image slice data. These performances are close to the performances offered by the underlying hardware, operating system (Windows NT) and network (TCP/IP over Fast Ethernet). The scaled down architecture offering its slice extraction services on the Web comprises a single Bi-Pentium II (450 MHz) and 16 disks. It runs all the system's functions: Web server, server interface, extent server and compute server (Fig. 2). This server is dedicated to the service of Web requests and offers a global access throughput of approximately two slices per second (512x512 colour slices). The access time comprises the following steps: request launching, slice retrieval, JPEG compression, slice transmission to client and slice display at the client site.

3. SLICE SEQUENCE ANIMATION ARCHITECTURE

The slice animation server provides a 3D view of anatomic structures by showing animations made of many successive slices along the user-defined trajectory. This service must be accessible by all Internet clients, even those having a low bit rate connection. It would have been easy to gather the animation as many JPEG-compressed slices. However, we prefer to allow the clients to save the resulting animation as one MPEG-1 video, reusable and playable with many video programs and utilities. We therefore decided to compress the successive extracted slices in the server as a video animation. The client only needs to send one request containing all animation parameters (animation size, distance between slices, trajectory control points, etc.).

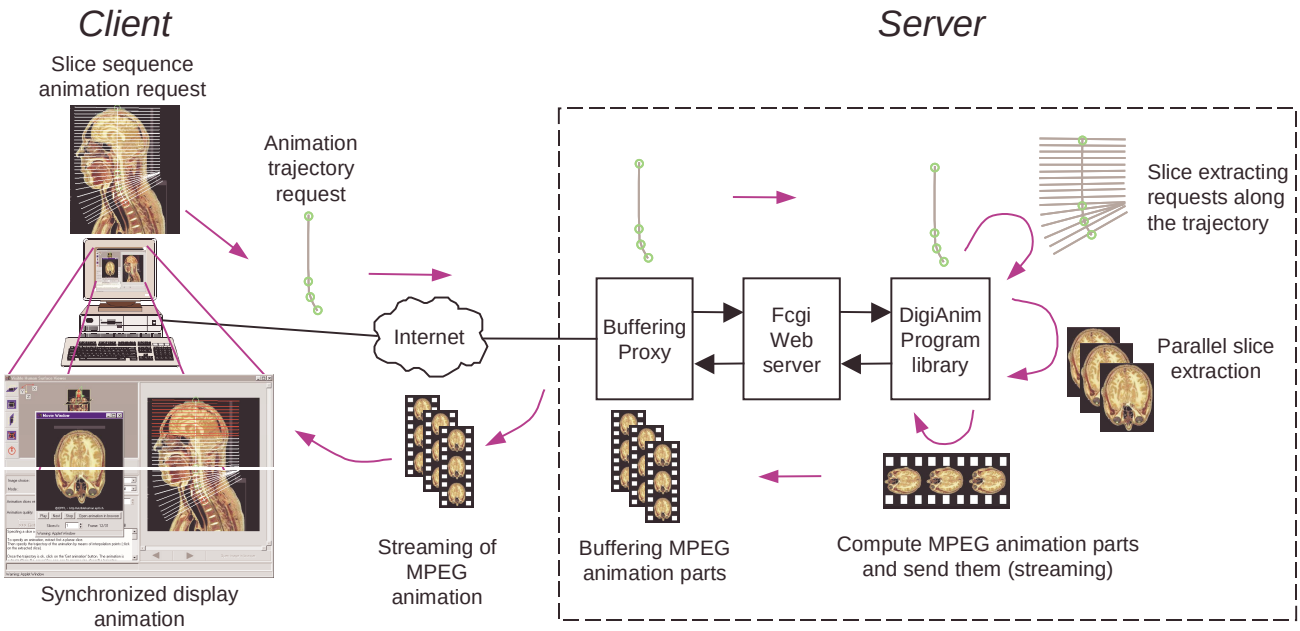


Fig. 3. Block diagram of the Visible Human Slice Sequence Animation Server

The slice sequence animation architecture (Fig. 3) comprises two parts:

1. The client side whose user interface was developed in Java 1.1.
2. The server side, a parallel program developed with the CAP[3] parallelization framework, and interfaced to the Web by FastCGI[6].

To extract a slice sequence animation, an Internet client defines a user-defined trajectory on a previously extracted slice and sends it to the server over the Internet. When the server receives the animation request, the request is transformed into many slice extracting requests. The animation slices are extracted in parallel and compressed in MPEG-1 video format. In pipeline, the MPEG parts are sent to the client. The client can within its applet visualize the resulting animation.

4. CLIENT SIDE USER INTERFACE

The interface for specifying the slice sequence animation parameters is the same as the interface for the Visible Human Slice Web server[2] including an additional "Slice sequence animation" mode (Fig. 4) allowing to modify the animation parameters (frame size, quality). Users define the animation trajectory by clicking on a previously extracted slice in order to create the trajectory's control points. The animation trajectory is given by the natural spline interpolating these control

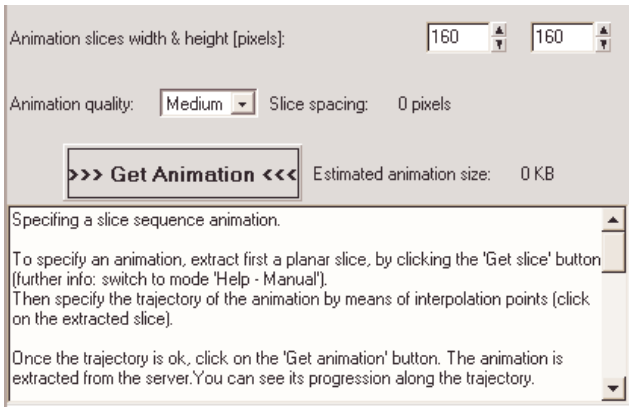


Fig. 4. Animation parameters panel

points. On this trajectory, each slice of the resulting animation is represented by an orthogonal line segment (Fig. 5, right part). The succession of line segments gives a feedback about slice size and slice spacing. These parameters determine the size of the final MPEG animation. Each modification of these parameters is automatically updated on the user-defined

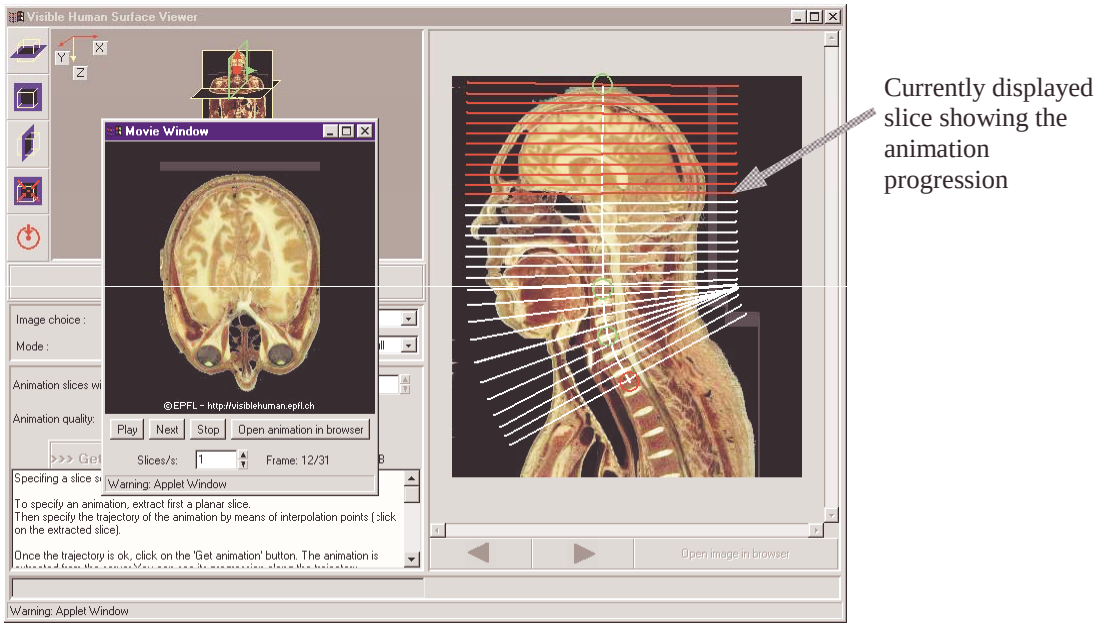


Fig. 5. Animation display in synchronization with slice progression

trajectory. The succession of line segments is also used as a progress bar indicating the animation download progression (Web streaming). When playing the animation, the succession of line segments is used to locate the currently displayed slice (Fig. 5, right part).

To visualize the animation, a Java MPEG decoder[5] has been integrated in the applet. Once the animation is completely downloaded, a movie window is opened. It allows to play, pause, stop and step-by-step display of the resulting animation, always in synchronism with the slice progress displayed on the user-defined trajectory (Fig. 5). It is also possible to change the animation display speed (slices per second).

Within the Java display window, we allow to save the resulting animation ("Open animation in browser" button). Due to security restrictions, it is not possible to directly access the local disks of an Internet client. To save the animation, we use the Web browser's cache. When the same animation request is opened in a new browser page, the browser recognizes the request and locates it in its own local cache without asking the server again. Saving animations by relaunching the request in a new browser page works well with Microsoft Internet Explorer. With Netscape, the animation request is not cached and therefore requested again from the server. This considerably slows down the saving of animations.

5. PARALLEL WEB SERVER

The Visible Human Slice Animation Server (Fig. 6) runs with FastCGI[6] and is easily portable across platforms (Windows NT, Linux, Unix).

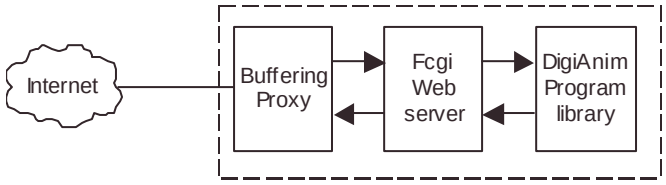


Fig. 6. Parallel server Web achitecture

In front of the server Web interfaced by FastCGI, a buffering proxy has been added in order to cache the computed animation for clients having a slow Internet connection. This buffering proxy is necessary since FastCGI accepts only one request at a time per process.

5.1. Creation of slices animations by parallel extraction of slices and serial MPEG-1 compression

The parallel animation server program (DigiAnim program library) extracts the slices required for the animation request in parallel on several PCs and, at the same time, compresses the slices into the MPEG-1 format. In pipeline, MPEG-1 slice sequences are sent to the client in streaming mode, while computation of the remaining parts of the animation proceeds.

The DigiAnim program is described by the pipelined parallel schedule shown in Fig. 7. The MPEG-1 video encoder[7] is able to encode 18 slices per second with independent frames coding (Pentium II, 450Mhz, 320x320 color slices). Since the time to encode a slice is smaller (1/18 s) than the time to extract a slice from the Visible Human dataset (1/11 s), we extract the slices in parallel and put them in a slice buffer for serial compression (producer-consumer scheme).

To create a slice sequence animation, the animation request (animation size, distance between slices, trajectory control points, etc.) is converted into a list of slices orthogonal to the trajectory (Fig. 7). The diagram is divided by an initial split function into two parts working in parallel: (1) the parallel extraction of the animation's slices (the producer), and (2) the MPEG-1 encoder (the consumer).

The list of orthogonal slices is split into individual slice request. Each slice is extracted by the ***Extract Slice*** function in parallel on several PCs. This function is the same as the Extract slice function of the single slice server (Fig. 2). Once a slice is extracted, the merge function writes the slice into the ***slice buffer*** (Fig. 7). This sequence of operations is realized in pipeline for all slice requests.

At the same time as the pipelined parallel slice extraction is carried out, the MPEG-1 encoder (***Compute MPEG animation*** function) gets the extracted slices from the ***slice buffer***, compresses them and sends the resulting animation part to the client (Web streaming).

In the current parallel implementation, the split function (***Split list into slice request***) generates the tokens (slice coordinates data structure) at a much higher rate than they can be consumed, i.e. processed by operations and merged to yield the result. Tokens may therefore accumulate in front of operations and merge functions. Such accumulation of tokens may require considerable amounts of memory and induce disk swapping (transfer of virtual memory to and from disk). To control this phenomenon, we created a flow control mechanism which limits the number of circulating request tokens between a split and a merge function. In order to avoid slice buffer overflow, we also created a synchronization (semaphore) between the split function of the parallel slice extraction and the MPEG-1 encoder. With this parallel program version, only the extraction of slices is carried out in parallel on several PCs. The MPEG compression is not parallelized. The performance analysis shows that the parallelization of MPEG compression is of interest when more than two PCs contribute to the computation of animations.

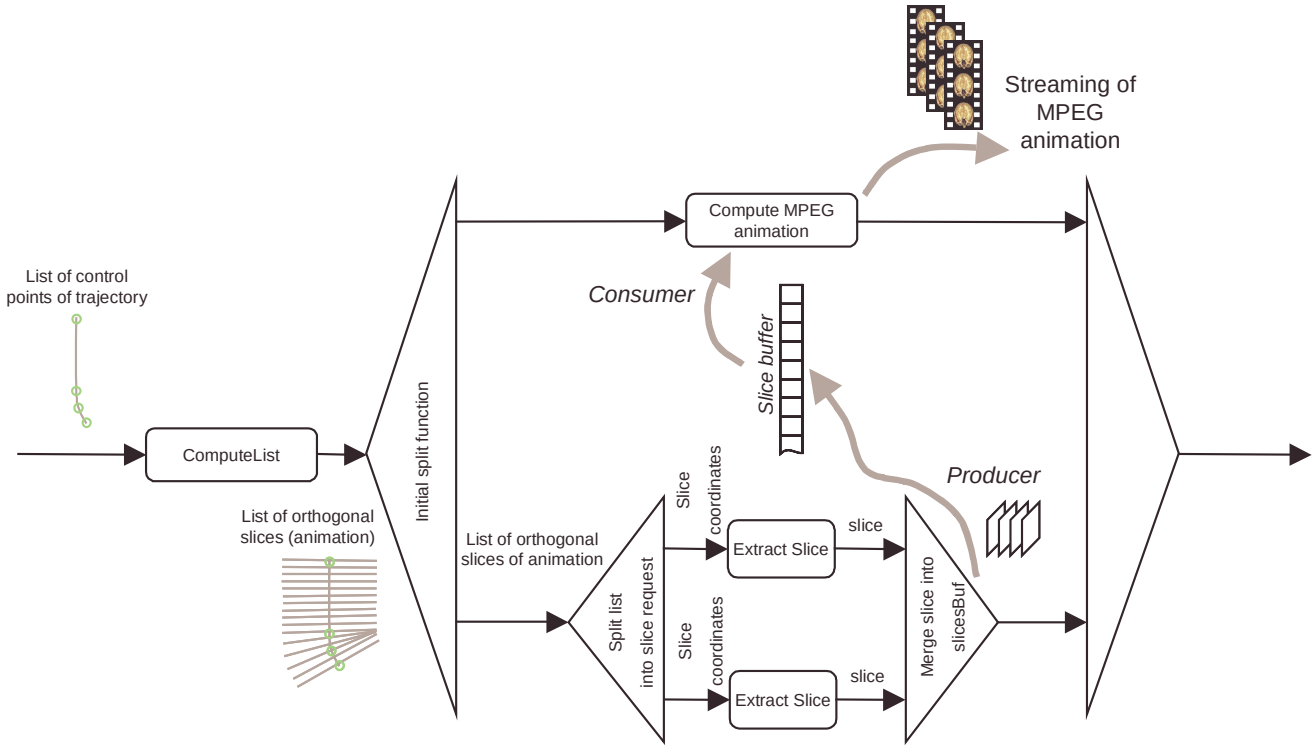


Fig. 7. Graphical representation of the pipelined-parallel slice sequence animation extraction library

On a single PC configuration (Bi-Pentium II, 450Mhz, 8 U2W disks), the DigiAnim program and compresses 7.7 slices per second, of size 320x320 pixels. In order to generate 7.7 slices/s, 40 MB/s of 3D Visible Human data must be accessed, partly from cache and partly from striped disk files. The MPEG output throughput is approximately 67 Kbytes/s.

The extraction of slices (***Extract Slice*** function) is carried out in parallel on several PC's. The server interface PC which gathers together all slice parts cannot receive more than 8 MB/s through its Fast Ethernet interface. Since the slice parts needed to generate a full slice may be up to 49% larger than the final slice size (the worst-case is a diagonal slice), the server interface PC can not receive more than 12.8 slices/s of size 320 by 320 pixels. Another bottleneck resides in the fact that one PC can only compress up to 18 slices/s. In order to offer more parallelization potential, we can duplicate the Visible Human dataset and synthesize in parallel different parts of the MPEG-1 animation. Instead of the slice parts, the compressed MPEG-1 animation parts are transferred to the server interface PC for creating the full animation and streaming it to the client (see next section).

5.2. Synthesis of the animation by parallel extraction of slices and parallel MPEG-1 compression

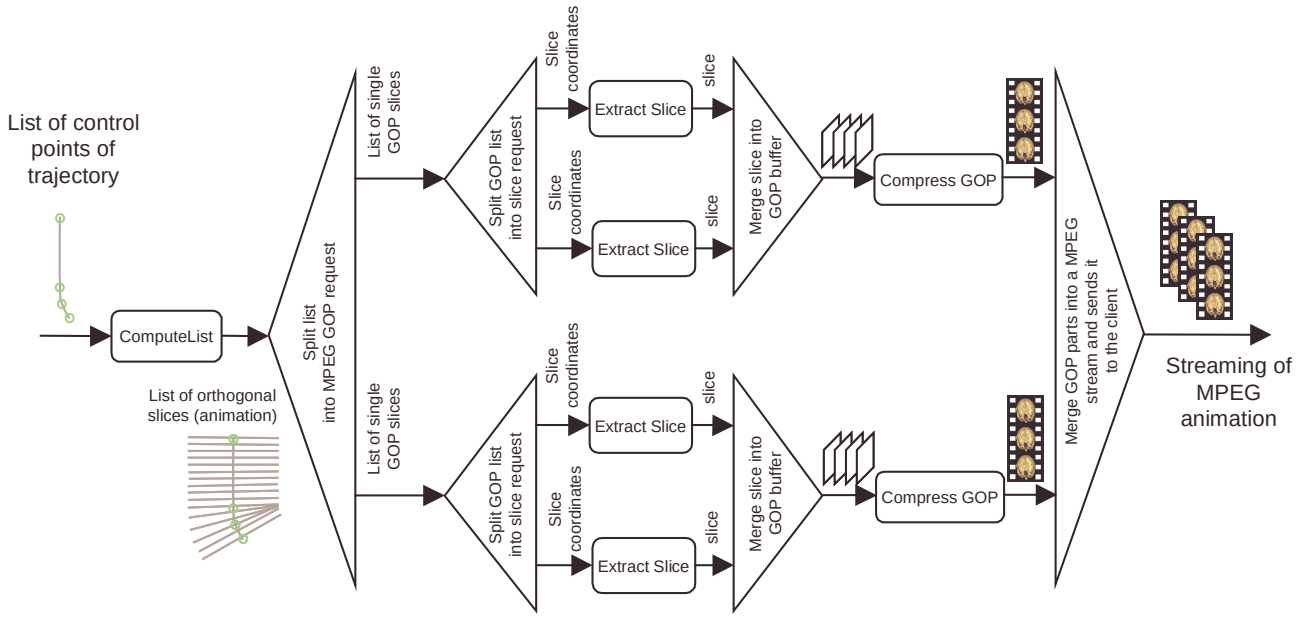


Fig. 8. Graphical representation of the pipelined parallel animation extraction with parallel MPEG compression

An MPEG-1 video sequence is composed of a series GOP's, each GOP being composed of several successive frames (slices). The GOP structure is intended to provide random access into a sequence. A GOP is an independently decodable unit that can be of any size as long as it begins with an independent frame (I-frame). Parallel MPEG-1 compression relies on the separate compression of the GOP parts of an MPEG stream. Parallel MPEG-1 compression is described by the pipelined parallel schedule of Fig. 8. The initial part of this diagram is the same as the diagram of Fig. 7. The animation request is also transformed into a list slices which need to be extracted. This list is split into several ***Lists of single GOP slices***. For each of them, each slice is extracted by the ***Extract Slice*** function in parallel on several PCs as in the previous version. The difference is that all extracted slices of a GOP request are directly encoded into an MPEG-1 part by the ***Compress GOP*** function. The last merge function merges the MPEG-1 GOP parts according to their order within the MPEG animation and sends the resulting MPEG animation to the client in streaming mode. This sequence of operations can is pipelined and carried out in parallel. Pipelining is achieved at two levels:

1. A GOP part can be compressed while the next one is being extracted.
2. A compressed GOP part can be merged into the final MPEG-1 stream and sent to the client while the next one is extracted and compressed.

Parallelism can be achieved at two levels:

3. Several slices can be extracted at the same time for one ***List of single GOP slices***.
4. Several GOP parts can be compressed at the same time.

Again, a flow control mechanism limits the number of request tokens in circulation and therefore the size of memory used by the application. The fully parallelized animation synthesis is running. Its performances will be tested both on small and large parallel configurations (up to 32 computers). The performance analysis should allow us to reduce as much as possible the number duplications of the dataset (disk space) and at the same time obtain the highest possible performances.

6. PERSPECTIVES AND CONCLUSIONS

The Visible Human volume is an exceptional data set on top of which many different services can be build in order to create the full atlas of the human body. Since 2D slices do not reveal the shape of 3D anatomic structures, we created the Visible Human Slice Sequence Animation server (http://visiblehuman.epfl.ch/animation/), which represents an additional milestone towards the interactive digital atlas of the human body. This new service which allows to extract and display animations along a user-defined trajectory is useful both for research and teaching purposes. The pipelined parallel slice animation extraction service runs efficiently on available commodity hardware such as PCs, SCSI disks and a Fast Ethernet switch.

In the future, we are interested in creating tools for associating sound to animations and creating a repository of annoted and commented slice sequence animations.

7. ACKNOWLEGEMENTS

We thank Benoit Gennart, Oscar Figueiredo, Marc Mazzariol, Joaquin Tarraga as well as our partners from the faculties of medicine Jean-Pierre Hornung, University of Lausanne, Luc Bidaut, Geneva University Hospital and Jean Fasel, University of Geneva. We thank the National Library of Medicine, Dr Ackerman, for making the Visible Human data set available. The research has been partly financed by the Swiss National Fund, SPP priority program in informatics and communications (grants 5003-45348/2 and 5003-51332).

References

1. “The NPAC Visible Human Viewer”, http://www.npac.syr.edu/projects/vishuman/VisibleHuman.html
2. R.D. Hersch, B. Gennart, O. Figueiredo, M. Mazzariol, J. Tarraga, S. Vetsch, V. Messerli, R. Welz, L. Bidaut, “The Visible Human Slice Web Server: A first Assessment”, *Proc. IS&T/SPIE Conf. on Internet Imaging*, San Jose, Jan. 2000, SPIE Vol. 3964, 253-258
3. V. Messerli, O. Figueiredo, B. Gennart, R.D. Hersch, “Parallelizing I/O intensive Image Access and Processing Applications”, *IEEE Concurrency*, Vol. 7, No. 2, April-June 1999, 28-37
4. M. Ackerman, “The Visible Human Project”, *Proc. of the IEEE*, Vol.86, No.3, March1998, 504-511
5. Carlos Hasan, “Java Mpeg Player software”, chasan@dcc.uchile.cl, Department of Computer Science, University of Chile, Santiago, Chile, downloaded at http://www.mpeg.org, November 1998
6. FastCGI for Windows, http://www.fastserv.com
7. K.L. Gong and L.A. Rowe, "Parallel MPEG-1 Video Encoding", *1994 Picture Coding Symposium*, Sacramento, CA, September 1994.