Towards an interactive navigation in large virtual microscopy images on 3D displays

J. Sarton, N. Courilleau, Y. Remion, Laurent Lucas

TOWARDS AN INTERACTIVE NAVIGATION IN LARGE VIRTUAL MICROSCOPY IMAGES ON 3D DISPLAYS

J. Sarton[1], N. Courilleau[1, 2], Y. Remion[1], and L. Lucas[1]

1. Université de Reims Champagne-Ardenne, CReSTIC, France
2. Neoxia, France

ABSTRACT

Acquiring biomedical images - multichannel, 3D, and/or time-lapse - using modern techniques such as slide scanners, confocal, or electron microscopy generates today huge data streams. This trend raises a large number of issues related to the management of these massive datasets. Efficient solutions for data storage and processing must be developed in order to deliver increasingly reliable and faster analyses. In addition, the improvement of workflows also requires the reinforcement of visualization capabilities. In this article, we introduce a new approach for high capacity screening of slices based on a tool combining pyramidal images representation and 3D visualization on multi-stereo display, in order to simulate virtual microscopy (VM). The preliminary results suggest that the proposed solution facilitates the reading and the understanding of data essentially because they are spatialized.

Index Terms— Virtual high-resolution microscopy, 3D display, Out-of-core hierarchical visualization, GPU rendering.

1. INTRODUCTION

Current biomedical acquisition devices can generate ultra-high resolution image data. Electron microscopy like digital histology allows, for instance, to deliver volumes of brain tissue with a resolution that may contain terabytes of raw data [4]. Interactive visualization of these volumetric data is essential partly because; *i)* our screens are only able to display a fraction of the resolutions of large images; and *ii)* they help to mentally understand the three-dimensional structures of images - how cells are organized to form tissues and organs or how a pathology changes the nature of the tissues.

In practice and thanks to the advances in computer technology, the development of VM [6, 17, 15] has partially addressed this issue. As a reminder, a VM could be defined as a system that simulates the observation of microscopic samples on computer mimicking a conventional microscope allowing to observe, navigate, and annotate virtual slides.

Our solution is based on this latter concept of VM. While many of these issues were addressed on the web, no formal study has ever been conducted in order to assess the contribution of 3D vision in this context. Our system, based on a multiresolution pyramidal representation of stacked images, provides a multiview display to improve the quality of the user experience.

Like DeepZoom [1] or other equivalent methods [7, 2], the basic idea of our tool consists in streaming visualization of large multiresolution datasets. Nevertheless, Hortsch [12] notes a drawback in such system used as a virtual microscope to study virtual slides. He reports that users of such equivalent tools were frustrated to have only a single plane focus and to lose three-dimensional perception. To compensate these disadvantages, we propose a rendering of each sight on an autostereoscopic screen [11]. In addition, this extension allows the user to freely navigate - zooming in or out like Google Earth - in the whole volume rather than in a single slide.

In order to reach this quality of navigation, we draw our inspiration from out-of-core methods management of large volume data on GPU used in the context of volume visualization. Usually, in these approaches, a pyramidal level of resolutions is created from the datasets. Each level of the pyramid is subdivided into blocks of data (called bricks) of the same size regardless the level. The Gigavoxel approach by Crassin et al. [8] stores the bricks in a tree structure. We rather turned to the proposed method of Hadwiger et al. [10] who provides a virtual memory approach with a multi-level, multi-resolution page table mechanism. This approach has already been validated by a concrete application with interactive exploration of petascale volumes [5].

The remainder of this paper is organized as follows. In Section 2, a brief overview of the system is provided. In Section 3, our visualization-driven pipeline is detailed. Experimental results are then presented and discussed in Section 4. Finally, conclusions are given in Section 5.

This work is supported by the French national funds (PIA2'program) under contract No. P112331-3422142 (3DNS project). We would like to thank the three clusters (Cap Digital and Systematic for ICT and Medicen for Health) that support this project.

978-1-5090-5743-6/16/$31.00

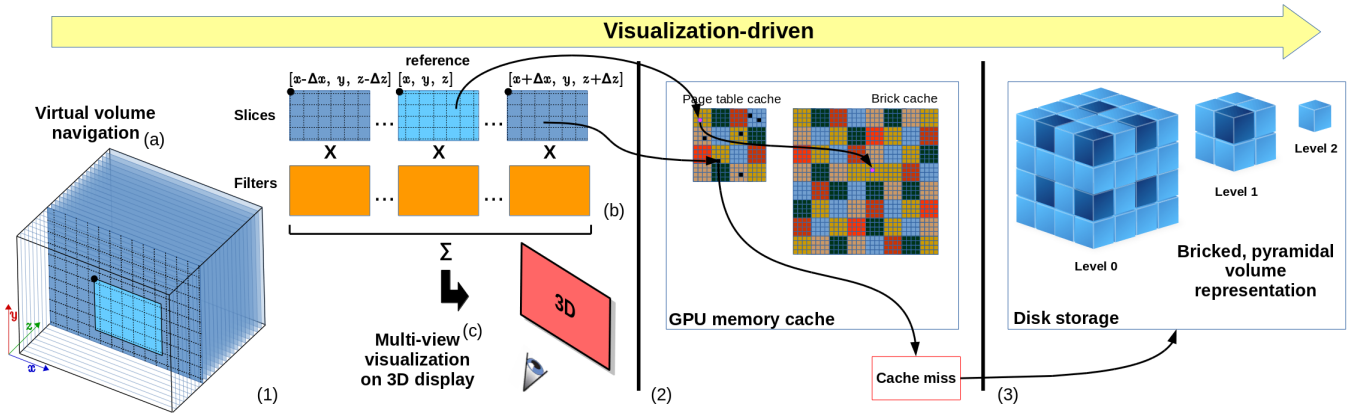

Fig. 1. **System overview**: The navigation in a virtual image stack and in a given slice (1.a) induce a streaming and a construction stage to compose a multi-view image (1.b) that can be visualized on a multiscopic 3D display (1.c). (2) An addresse translation system with a multi-level multi-resolution page table hierarchy and a caching system on the GPU. (3) A bricked multi-resolution pyramidal representation stored on a larger space storage.

2. METHODOLOGY

2.1. System Overview

An overview of our system architecture is shown in Fig.1. Our method proposes a threefold concept:

- an interactive on-demand streaming system to navigate in large multi-resolution images;
- an out-of-core data management with a GPU cache mechanism to manipulate very large volumes;
- a visualization on 3D multi-view displays.

The treatments to apply this rendering method on 3D display (Fig. 1.1.c) are mainly matrix computations (Fig. 1.1.b). They are completely adapted to the massively parallel architecture of GPUs. Furthermore, the use of GPUs is recommended to reach an interactive navigation. However, the restricted memory space on GPUs implies the use of an appropriate data representation and data structure to manipulate huge volumes in an out-of-core way. That is why a bricked multi-resolution data representation is proposed, computed in a preprocessing step (Fig. 1.3). In addition, we propose to use a multi-level, multi-resolution page table hierarchy as data structure to address the entire volume in a virtual way (Fig. 1.2) and, a GPU cache mechanism to store and manage bricks. This allows us to propose an interactive on-demand, bricks streaming system (Fig. 1.2).

The entire pipeline is driven by the visualization. When the view changes, it triggers the following steps.

First, the system determines which bricks are needed to the final multi-view frame construction. Second, using the GPU data structure, it accesses these bricks. For bricks that are not present on the GPU memory (cache miss), it send a request to the CPU to asynchronously get them back. They are either, in the CPU memory or on the storage device, where the bricked multi-level volume representation is stored. Third, when all the needed bricks are present in the GPU cache, it builds the final multi-view frame and visualize it on the 3D multi-view display.

The change of the view position can result to one of the following actions: *i)* a zoom in/out on a given slide, which induces a pyramid navigation; *ii)* a $[x, y]$ pan in a given slide in the same pyramid level; or *iii)* a slice change (volume depth) in the same pyramid level.

2.2. Data representation

Before being able to use the data in the real-time pipeline for the navigation, the volume needs to be transformed. This step consists in the creation of a bricked pyramidal data representation (Fig. 1.3). The bricked representation allows to manipulate small (e.g., 16^3 or 32^3 pixels) independent bricks rather than the whole large volume. In contrast to Hadwiger et al. [10], the acquisition of the whole volume is considered to be already completed and it does not need to handle tile streams from a microscope. Thus, the brick creation step can be done once on the whole volume in a preprocessing step.

This data representation is then fully stored on a large storage space like a hard drive disk. It is possible to stock these data in a compressed format in order to manipulate smaller files. The chosen compression algorithm should be able to compute the extraction step in real-time before providing the brick to the GPU memory.

Fogal et al. [9] propose a study on the optimal size for the bricks. It is shown to be more interesting to have a brick size of 128^3 or 256^3 for the storage or for the data transfer but it is preferable, for the renderer, to manipulate smaller

bricks, such as 32^3. However, this study was made for the visualization via volume ray-casting. In our case, it could be more interesting to use even smaller brick size in our renderer. To perform this, the use of the rebricking system presented by Fogal et al. [9] could considered.

It can note that acquisitions of biomedical data may provide an anisotropic representation. The data representation shown is able to deal with this, by applying different downsampling ratios for each axes of the volume and for each level in the pyramidal representation.

3. VISUALIZATION-DRIVEN PIPELINE

3.1. Volume data construction

To compute the final multi-view image that will be displayed on the multi-view screen (Sec. 3.2), it needs to create the N images (N = number of view of the display) required to its composition. The navigation is performed in a virtual volume (Fig. 1.1.a) that represents the size of the whole volume to the highest resolution. In this virtual volume, a 3D position and a level of resolution in the pyramid are determined. The 3D position could make reference to upper-left corner of the reference area to visualize. The level of resolution is chosen by the user. From the previously described information and the definition of the screen, the system deduce which bricks will be required to compose the N images. Finally, for each required bricks, brick requests are sent to the cache manager.

The data structure used is a multi-level multi-resolution page table hierarchy [10]. It allows the addressing of the whole volume with the concept of virtual memory. The volume is virtualized and we reach it with the translation of a virtual address to a physical address thanks to the page table hierarchy. In the case where the volume is too massive, we can apply the same concept of virtualization on the page table itself. This is why we talk about multi-level table. According to Hadwiger et al. [10], such hierarchy with two levels allows the addressing of several peta-bytes of data and is more appropriate than a tree structure. To handle this data structure we use the memory of the GPUs. Two pools are created, one for the page table hierarchy and the other one for the data (bricks, Fig. 1.2). The cache updates are managed with a *Least Recently Used* (LRU) cache.

When the navigation is only done through the pyramid of resolution (meaning a fixed $[x, y]$ camera position), it could be interesting to use a strategy to handle a progressive loading of levels of resolution. When this case occurs, the bricks of lower resolution could be used while the ones from the requested resolution are not loaded in the GPU memory.

3.2. 3D display

Our system uses an N-view autostereoscopic display. We now describe our proposal to create a viewable frame on such screens. This process has a strong dependency on the display hardware (definition and view number N).

To create a viewable frame, it is required to generate the N images that compose it. These images are created according to a reference image. The reference image is the one the user is interested in; it is the one corresponding to the 3D position used during the virtual volume navigation.

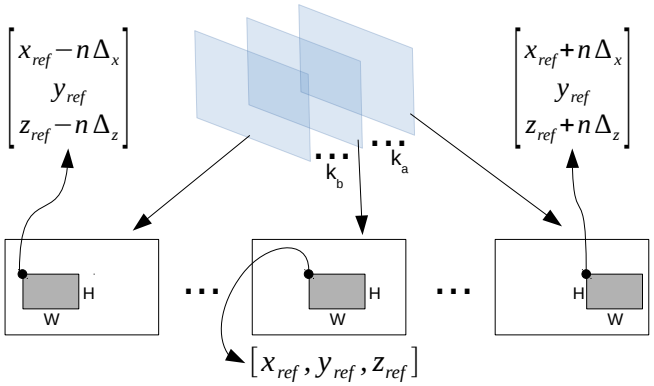

Fig. 2. Computing area positions in the selected slices.

Let *W* be the width of the display definition, *H* the height of the display definition and $p = [x_{ref}, y_{ref}, z_{ref}]$, the position of the upper-left corner for the reference image. To create this image, it has to get all the needed bricks $\in [x_{ref}, x_{ref} + W]$ and $\in [y_{ref}, y_{ref} + H]$ in the slice z_{ref}.

We select, in the volume, k_b slices before and k_a slices after the reference slice position (z_{ref}). The value k_a may be different from k_b, depending on the number of views of the display (e.g., if the display requires an even number of views) and where the camera is positioned in the volume (e.g., if the camera position is at the edge - front or back - of the volume).

To pick up the images around the reference one, we apply an offset on the x-axis (noted Δ_x), encoding the horizontal disparity [13, 14] and a Δ_z for the slice position. In this way and according to the previous notations, the positions for the images before, with $n \in [1; k_b]$, are $p_n = [x_{ref} - n\Delta_x, y_{ref}, z_{ref} - n\Delta_z]$ and for the images after, with $n \in [1; k_a]$, are $p_n = [x_{ref} + n\Delta_x, y_{ref}, z_{ref} + n\Delta_z]$ (Fig. 2). In the same way as the reference image, we need all the bricks from positions p to $p.x + W$ and $p.y + H$ in the corresponding $p.z$ slice.

All the required information are now known and the brick requests can be sent to the cache, in order to construct the N images. However, cache misses may appear if the camera position was changed from the previous rendering pass. In this case, we could consider different strategies. While the cache is fetching missing bricks, we can start building images as soon as all their required bricks are available. This case could occur when the position of the camera changes on the x or z-axis. Sadly, because all images required to produce

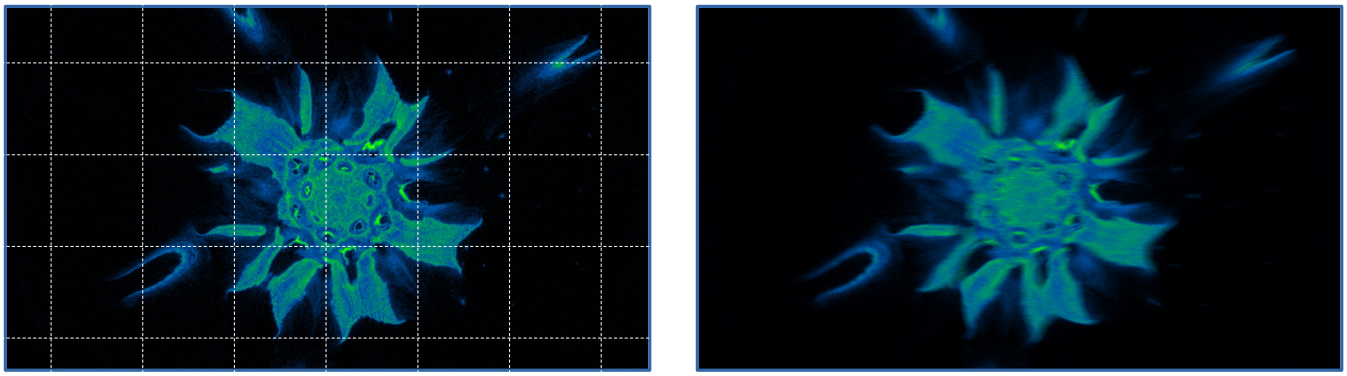

Fig. 3. **Results**: The left image is the reference image used, rebuilt from the bricks. The right image shows the result of the 8-views frames composition.

the multi-view frame are sharing the same y_{ref}, cache misses could happened when the camera moves on this axis, there is no other choice than waiting for all missing bricks from the cache.

In order to obtain a valuable image, each of the N pre-build views $\mathcal{V}_c^i(x,y)$ must be combined conforming to the autostereoscopic display device according to the expression $\mathcal{V}_c^{final}(x,y) = \sum_i \mathcal{V}_c^i(x,y)\ \mathcal{F}_c^i(x,y)$ with $c \in \{R,G,B\}$. As shown in Figure 4, this combination is achieved by selecting each color component of the final image in the view locally indicated by dedicated filtering masks $\mathcal{F}^i : \{0,1\}^3\ [\mathbb{Z}^2]$ (specific to each 3D display). This process can be easily implemented with a simple GPU kernel. Regarding the filters, they are provided as inputs. So they can be loaded once on the GPU during an initialization step.

Finally, the final frame is ready to be displayed on the 3D display once the sum of the N resulting matrix is done.

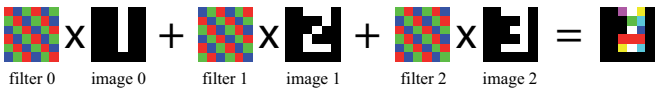

Fig. 4. Filters application on views. [16]

4. RESULTS AND DISCUSSION

Given the following configuration: an high definition screen (1920×1080), a brick size of 32^3, $\Delta_x = 4$ and $\Delta_z = 1$. The number of bricks required to compose the final multi-view image will be, in the worst case, $60 \times 34 \times 2 = 4080$ bricks. For a volume with 8-bit pixel encoding, it will take around $130MB$. Modern GPU memory can easily handle this, meaning there is no need to consider strategies for the GPU memory overflow. It allows us more freedom on the number of bricks loaded in the cache. This given 32^3 brick size allows the user to navigate freely and decently in the volume before getting cache misses.

In order to perform the tests of our system we were using an HD autostereoscopic display using a 16:10 aspect ratio. This display is an 8-view autostereoscopic display with a definition of 1920×1200. The sample we used was representation of a flower (from [3]) in a discretized volume 1024^3 of unsigned 8-bit where the voxels are uniformly spaced by $60\,\mu\mathrm{m}$. In order to get a better rendering, we decided to do a preliminary processing on the volume to transform it to a 2048^3 volume, using a bicubic interpolation. We applied this transformation in order to have one voxel per pixel for a high definition display. In addition, we applied a LUT in order to have a volume encoded on 3 channels in colours. The result is shown in Figure 3.

We chose the delta values $[\Delta_x, \Delta_z] = [4, 1]$. It appears that the value of Δ_z is directly limited by the physical distance between two slices. With a distance too large between slices, even if $\Delta_z = 1$, the visualization will not be smooth. However, if the distance between two slices is very short and Δ_z is too small, the user will not have the feeling to move in the depth of volume. In the same way, the problem appears on the x-axis where a high Δ_x gives an unpleasant feeling when the user changes of perspective; furthermore, it introduces an undesirable blur effect.

Thus, the objective to merge the concept of Deep Zoom applied on a 3D volume displayed on a multiscopic display was conclusive and a real depth effect can be observed. However, perception was only visually assessed by the authors of this paper, and a statistical study on a larger sample of users is required. In order to increase the quality of the system, the study should be focused on the feedback of a pool of persons on their perceptions on multiple samples with different settings. Moreover, the strategies used to create the images may be improved when cache misses on y-axis appear.

5. CONCLUSION

We have proposed in this article a 3D vision extension to VM. This approach allows users to interactively navigate through large multiresolution datasets while preserving the depth information. To achieve this we suggest a GPU visualization-driven pipeline based on an adapted data structure. The preliminary results suggest that the proposed solution facilitates the reading and the understanding of data essentially because they are spatialized.

6. REFERENCES

[1] Deep zoom. `https://www.microsoft.com/silverlight/deep-zoom`. [Online; accessed 08-October-2016].

[2] Openseadragon. `http://openseadragon.github.io`. [Online; accessed 08-October-2016].

[3] University of zurich - department of informatics - visualization and multimedia lab. `http://www.ifi.uzh.ch/en/vmml/research/datasets.html`. [Online; accessed 08-October-2016] We acknowledge the Computer-Assisted Paleoanthropology group and the Visualization and MultiMedia Lab at University of Zurich (UZH) for the acquisition of the uCT datasets.

[4] K. Amunts, C. Lepage, L. Borgeat, H. Mohlberg, T. Dickscheid, M-É. Rousseau, S. Bludau, P-L. Bazin, L. B. Lewis, A-M. Oros-Peusquens, N. J. Shah, T. Lippert, K. Zilles, and A. C. Evans. BigBrain: An ultrahigh-resolution 3d human brain model. *Science*, 340(6139):1472–1475, 2013.

[5] J. Beyer, M. Hadwiger, A. Al-Awami, W. K. Jeong, N. Kasthuri, J. W. Lichtman, and H. Pfister. Exploring the Connectome: Petascale volume visualization of microscopy data streams. *IEEE Computer Graphics and Applications*, 33(4):50–61, 2013.

[6] U. Catalyurek, M. D. Beynon, C. Chang, T. Kurc, A. Sussman, and J. Saltz. The virtual microscope. *IEEE Transactions on Information Technology in Biomedicine*, 7(4):230–248, 2003.

[7] G. Corredor, M. Iregui, V. Arias, and E. Romero. Flexible architecture for streaming and visualization of large virtual microscopy images. In *Medical Computer Vision. Large Data in Medical Imaging*, number 8331 in Lecture Notes in Computer Science, pages 34–43. 2013.

[8] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Symposium on Interactive 3D graphics and games*, pages 15–22. ACM, 2009.

[9] T. Fogal, A. Schiewe, and J. Kruger. An analysis of scalable GPU-based ray-guided volume rendering. In *IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, pages 43–51, 2013.

[10] M. Hadwiger, J. Beyer, W-K. Jeong, and H. Pfister. Interactive volume exploration of petascale microscopy data streams using a visualization-driven virtual memory approach. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2285–2294, 2012.

[11] N. S. Holliman, N. A. Dodgson, G. E. Favalora, and L. Pockett. Three-dimensional displays: A review and applications analysis. *IEEE Transactions on Broadcasting*, 57(2):362–371, 2011.

[12] M. Hortsch. From microscopes to virtual reality – How our teaching of histology is changing. *Journal of Cytology & Histology*, 4(3), 2013.

[13] G. R. Jones, D. Lee, N. S. Holliman, and D. Ezra. Controlling perceived depth in stereoscopic images. volume 4297, pages 42–53, 2001.

[14] L. Lucas, C. Loscos, and Y. Rémion. *3D Video: From Capture to Diffusion*. John Wiley & Sons, 2013. Chapters 4 and 14.

[15] J. Molin, A. Bodén, D. Treanor, M. Fjeld, and C. Lundström. Scale Stain: Multi-resolution feature enhancement in pathology visualization. *arXiv preprint arXiv:1610.04141*, 2016.

[16] O. Nocent, S. Piotin, A. Benassarou, M. Jaisson, and L. Lucas. 3d displays and tracking devices for your browser: A plugin-free approach relying on web standards. In *International Conference on 3D Imaging (IC3D)*, pages 1–8, 2012.

[17] C-W. Wang, C-T. Huang, and C-M. Hung. VirtualMicroscopy: Ultra-fast interactive microscopy of gigapixel/terapixel images over internet. *Scientific Reports*, 5:14069, 2015.