Virtual review of large scale image stack on 3D display

J. Sarton, N. Courilleau, A-S. Herard, T. Delzescaux, Y. Remion, Laurent Lucas

VIRTUAL REVIEW OF LARGE SCALE IMAGE STACK ON 3D DISPLAY

J. Sarton[1], N. Courilleau[1, 2], A-S. Hérard[3], T. Delzescaux[3], Y. Remion[1], and L. Lucas[1]

1. Université de Reims Champagne-Ardenne, CReSTIC, France
2. Neoxia, France
3. CEA-CNRS-UMR 9199, MIRCen, France

ABSTRACT

Large scale images allow pathologists to perform reviews, using computer workstations instead of microscopes. This trend raises a wide range of issues related to the management of these massive datasets. In particular, efficient solutions for data storage and processing have to 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 paper, we present a new virtual microscopy (VM) approach for interactive-time navigation in large images stack with 3D visualization on multi-stereo display. Our work is based on a tool that combines 3D pyramidal representation and out-of-core data management for interactive on-demand streaming of large datasets on GPUs. The preliminary results suggest that the proposed solution facilitates the reading and the understanding of data essentially because they are spatialized.

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

1. INTRODUCTION

Current image-generating technologies are essential tools within modern biology despite the large amounts of data they generate [1]. 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 understand the three-dimensional structures of images and determine spatial relationships between regions of interest.

In practice, and thanks to the advances in computer technology, the development of VM [2, 3, 4] 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, by mimicking a conventional microscope enabling to observe, navigate [5, 6], and annotate virtual slides [7]. Nevertheless, Hortsch [8] notes a drawback in such system used as a virtual microscope to study virtual slides. Indeed, 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 to extend visualization to 3D by using autostereoscopic display [9] in order to improve the quality of the user experience. In addition, this extension allows the user to freely navigate – zooming in or out – 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, one creates a pyramidal level of resolutions, subdivided into blocks of data (called bricks). The Gigavoxel approach [10], stores the bricks in a tree structure. We rather turned to the proposed method of Hadwiger et al. [11] who provide a virtual memory approach with a multi-level, multi-resolution page table mechanism for ray casting rendering. This approach has already been validated by a concrete application with interactive exploration of petascale volumes [12].

Based on this out-of-core method, we extend it to propose a visualization-driven pipeline which generates stereoscopic multi-view frames on GPU at interactive time from data bigger than GPU memory.

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. Issues on dataset representation are specifically studied. Experimental results are then presented and discussed in Section 4. Finally, conclusion is given in Section 5.

2. MATERIALS AND METHODS

The data used in this article was obtained from a *C57BL/6J* mouse brain processed to improve block-face photograph resolution compared to a previous research realized on Alzheimer's Disease [13]. The photographic volume produced was composed of block-face images of the surface of the frozen brain taken before the cutting process (slice thickness of $20\,\mu\mathrm{m}$ to derive histological sections not treated in this work). Every image was recorded with a digital camera (Oscar, Allied Vision). These photographs were taken at the end of the cryostat wheel crank course, ensuring that the brain was in the same position for all sections (x and y-axes) and at the same distance from the camera (z-axis). A laptop was used to control the camera, making it possible to take images remotely; the images were stored directly on this laptop. For the processed brain, we obtained a series of 800 blockface photographs with an in-plane resolution of $2.16 \times 2.16\ \mu\mathrm{m}$ per pixel. Image processing steps was performed using BrainVISA software (http://brainvisa.info) to reconstruct the volume. Such 3D photographic information was proposed in many papers to perform efficient histological 3D reconstructions based on series of sections.

This dataset comprise a preliminary result with a 3D high resolution volume (far superior to MRI images) even if the precision does not reach the one given by 3D histological data. In addition, it provides a good overview and interesting anatomical information (white matter – gray matter).

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.

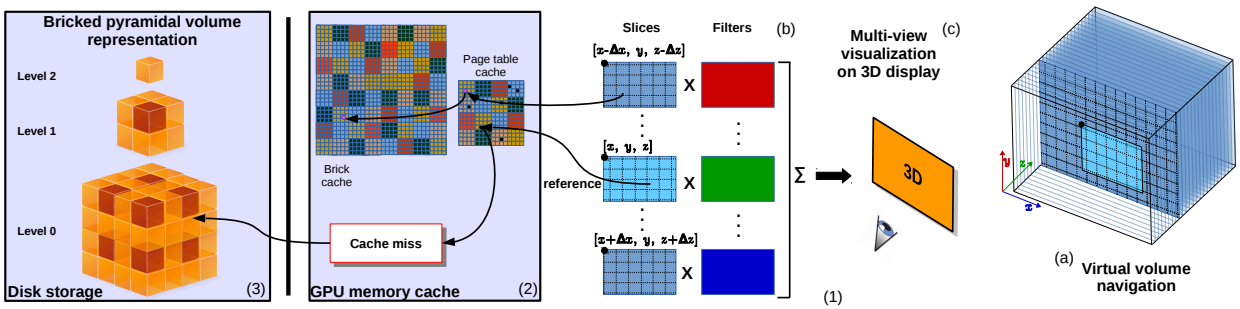

Fig. 1. **Visualization-driven system overview**: The navigation in a virtual image stack and in a given slice (1.a) induces a streaming and a construction stage to compose a multi-view image (1.b) that can be visualized on a multiscopic 3D display (1.c). (2) An address translation system with a multi-level 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.

2.1. System Overview

The system we propose aims to response to the following needs by providing *i)* a way to navigate interactively through a large multi-resolution image stack, *ii)* a mechanism to manage very large volumes of data, and *iii)* a visualization on 3D multi-view displays.

Based on a GPU architecture (Fig. 1), it includes a specific approach for handling datasets larger than GPU memory capabilities in addition to ensure an interactive time navigation for the users. Remember that GPUs do not have a lot of memory, which implies the use of an appropriate data structure. To overcome this limitation, a distinct out-of-core algorithm was designed to manage datasets and to reduce I/O bottleneck. In a preprocessing step, a bricked multi-resolution pyramidal data representation of the volume is created (Fig. 1.3). In addition, in order to address the entire volume, the system uses a multi-level, multi-resolution page table hierarchy (Fig. 1.2) and a caching system to store and manage the bricks.

The user can navigate through the volume by performing one of the following actions: *i)* a zoom in / out on a slide, which implies a change of precision level in the pyramid; *ii)* a move on the current slide, which means a $[x, y]$ pan; or *iii)* a move in the depth of volume which is a change of slices. As shown in figure 1, the system was built on a visualization driven pipeline. In other words, if one of these actions is observed, the system runs automatically the following sequence.

First step, according to the new view position, the system calculates all bricks needed to build the final multi-view image. Second step, the system tries to get the access to the bricks in the GPU memory (Fig. 1.2). In the case the system does not find them (cache misses), it automatically sends asynchronous request to the CPU to fetch them. They are either, in a simple cache in the CPU memory or in the storage device, where the bricked multi-level volume representation is stored (Fig. 1.3). Third step, the system creates the final multi-view image and displays it on a 3D multi-view screen (Fig. 1.1.c).

2.2. Data representation

In order to reach an interactive time navigation ($\geq$ 15 FPS), a well-designed data architecture is required. The architecture used to handle huge volume of data is based on the work of Hadwiger et al. [11] who proposed a bricked pyramidal data representation also called 3D mipmap (Fig. 1.3). Rather than trying to use the whole volume, this data representation makes possible the use of small independent bricks (e.g., 16^3 or 32^3 voxels). However, the approach proposed in [11] was based on a complete interactive-time pipeline, from data acquisition to visualization, which is not the case here. Indeed, the volume acquisition was already performed, and the construction of the pyramid can be done outside the stream. In addition, to obtain better performances, the data stored on the hard drive could be compressed; however, since the system aims to offer a interactive-time navigation, it implies to have an extraction algorithm with a throughput satisfying this constraint.

The study proposed by Fogal et al. [14] gives a good overview to choose correctly the optimal brick size. They highlight two important points: *i)* it is, more efficient to use a large brick size to transfer and store the data (e.g., 128^3 or 256^3); *ii)* for rendering, it is more interesting to use small brick size (e.g., 32^3) (Sec. 4).

As a bonus, with the use of this data structure, it is possible to apply different down-sampling ratios on each axes of the volume for each level in the pyramidal representation. It is important to note that acquiring biomedical data may introduce an anisotropic representation, and this data structure can easily deal with this by applying appropriate ratios.

3. VISUALIZATION-DRIVEN PIPELINE

3.1. Volume data construction

The navigation is performed in a virtual volume (Fig. 1.1.a) that represents the whole stack images at the highest resolution. In this virtual area, a 3D position representing the upper-left corner of the section to visualize is determined. Moreover, the appropriate level of resolution in the pyramid is chosen according to the pixel size projection of the image in the screen space. In addition to the screen definition, it is possible to deduce which bricks will be required to compose the N images (N = number of view of the display) needed to compute the multi-view image that will be displayed afterwards (Sec. 3.3). Then, a brick request is sent to the GPU cache manager for each of these required bricks.

In order to address the whole large volume, a virtual memory scheme is used with an address translation mechanism. This is ensured directly on the GPU by a multi-level, multi-resolution page table hierarchy. In the case where the volume is too massive, the same concept of virtualization can be applied on the page table itself. To handle this data structure, two pools are created on the GPU memory (global or texture memory), one for the page table hierar-

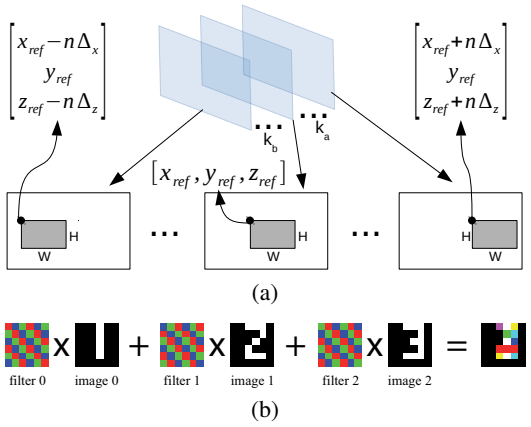

Fig. 2. (a) Computing area positions in the selected slices. (b) Multiplexing filters application on views. [15]

chy, the other one for the data bricks (Fig. 1.2). The cache updates are managed with an LRU policy.

Strategies can be applied when the navigation is only done through the pyramid of resolutions (meaning a fixed camera position) to handle a progressive loading, and preserve an interactive rate. The available bricks of lower resolution could be used while those from the requested level are not loaded in the GPU memory.

3.2. Views generation

To produce the image that will be displayed on the N-view autostereoscopic screen, it is necessary to generate the N views and to compose them (Sec. 3.3). These views are created according to a reference area, which is the one the user is interested in, the one corresponding to the 3D position used during the virtual volume navigation. This process has a strong dependency on the display hardware (definition and view number N).

Let *W* and *H* be the width and the height of the display definition respectively, and $p = [x_{ref}, y_{ref}, z_{ref}]$, the position of the upper-left corner for the reference image. All the bricks covering the area from the position $[x_{ref}, y_{ref}]$ to the position $[x_{ref} + W, y_{ref} + H]$, in the slice z_{ref}, are required to create this reference image.

The images around the reference image are selected by applying an offset on the x-axis (noted Δ_x), encoding the horizontal disparity [16, 17] and a Δ_z for the slice position. In the volume, k_b slices before and k_a slices after the reference slice (z_{ref}) are selected. 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). In this way, 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.a). In the same way as the reference image, all the bricks from positions p to $[p.x + W, p.y + H]$ in the corresponding $p.z$ slice are needed.

When all the previously described information are known, the brick requests can be sent to the cache, in order to construct the N images. Different strategies can be considered to handle cache misses. As soon as all the required bricks of an image are present in the cache, this image can be built while the cache is fetching missing bricks of the other images.

This case could occur when the position of the camera changes on the x or z-axis. Unfortunately, because all images required to produce the multi-view frame are sharing the same y_{ref}, cache misses may happen when the camera moves on this axis. Then, there is no other choice than waiting for all missing bricks from the cache.

Rendering quality and user interaction are improved by adding an alpha transparency information during the pixel compositing. A transfer function (TF) is used to determine this transparency coefficient (and eventually the color) for each intensity values. Then, to integrate this transparency through the thickness of the N slices, an alpha-blending computation is operated on the images.

It is possible to consider that a fully transparent brick, according to the TF, will not be loaded into the GPU cache. This TF can be easily updated by the user with a visualization impact in interactive time. Indeed, a TF update will trigger the pipeline only from the alpha-blending step since all the required bricks are already in the cache.

3.3. 3D display

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:

$$\mathcal{V}_c^{final}(x, y) = \sum_i \mathcal{V}_c^i(x, y)\ \mathcal{F}_c^i(x, y) \quad \text{with } c \in \{R, G, B\}$$

As shown in figure 2.b, 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; it consists of N $[H \times W]$ matrix product and a global sum. The multiplexing filters are of the size of the display definition, and they are always the same; so they can be loaded once on the GPU during an initialization step.

4. RESULTS AND DISCUSSION

4.1. Memory usage

According to the following settings: a display with a $H \times W$ screen size, a volume with p-bytes encoded pixels and a brick size of b^3 pixels, in the worst case scenario, the working-set size can be determined with:

$$\left(\frac{H}{b} + 1\right) \times \left(\frac{W}{b} + 1\right) \times 2 \times (b^3 \times p) \qquad (1)$$

With an HD 1920×1080 display and a volume with RGBA32 pixels, this working-set size will take ~560 MB with $b = 32$ and ~270 MB with $b = 16$. We can notice than modern GPUs memory can easily handle this, meaning there is no need to consider strategies for the GPU memory overflow.

Table 1 shows that bigger are the bricks better is the memory bandwidth which is in line with the study of Fogal et al. [14] about the brick size for the data transfers.

Their study also points out that it is interesting, in the case of GPU volume ray-casting, to use small bricks for the renderer. In the same way for our multi-stereoscopic frame renderer, using small bricks provides a finer granularity on data that will be required to build a frame and avoid the loading of unnecessary data. Thus, the memory space used on the GPU can be optimized with smaller bricks, as shown with equation (1). In addition, it could be interesting to take in consideration the L1 cache size of the GPUs. Hence, in order to optimize the number of cycles, it could be interesting to use

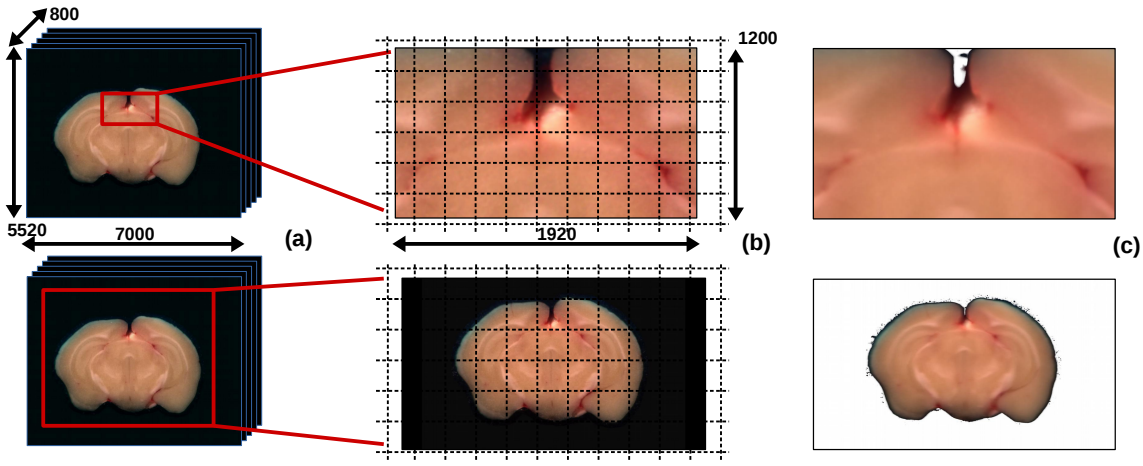

Fig. 3. **Results**: Mouse brain block-face volume (~86 GB) render in ~30 ms. (a) A full large-scale slide inside an images stack. (b) The reference images used, rebuilt from the bricks. (c) The results of 8-views frames composition with transfer function and alpha-blending.

bricks that can entirely fit in L1 cache to avoid to load / unload brick chunks. In modern GPUs architecture the L1 cache has a size of 24 - 48 kB, it induces to take in consideration the use of 16^3 bricks (16 kB).

To conclude, it would be better to use a rebricking strategy in order to handle large bricks for data transfers and small bricks in the GPU cache for the renderer. An other idea could be to use non-cubic bricks and to link the brick size to the sizes and views the display requires (e.g., $[x, y, z] = [256, 128, 32]$).

Brick Size (RGBA32)	Disk → CPU (µs)	CPU → GPU (µs)	Bandwidth CPU → GPU
16^3	944	204	78.4 MB/s
32^3	1 488	246	532.5 MB/s
64^3	6 945	376	2.6 GB/s
128^3	31 970	867	9.2 GB/s

Table 1. Per-brick load and transfer time analysis according to the brick dimensions. The samples were taken using a computer with the following specs: Intel i7 6700HQ @ 2.6GHz, Geforce GTX960M and an SSD in PCIe

4.2. Time analysis

Table 1 illustrates the time needed to fetch a brick from the disk to GPU. As expected, loading a brick from the disk is significantly more time consuming than loading it from the CPU memory. Fortunately, this time is taken in consideration only if a requested brick is not present in our GPU cache and not present in the CPU cache.

Furthermore, the transfer time does not count during the frame creation time. When a brick is missing, the system raises an asynchronous request to fetch it and try to compute the frame by using bricks from lower resolutions. The time to build one multi-view frame is ~30ms, it considers the selection and the composition of the views required for the multi-view frame, alpha blending and multiplexing. This processing time depends on the size of the output 3D screen and the number of views that compose it. It is not affected by the size of the bricks or the selected zooming level.

These times validate the navigation and visualization at an interactive rate (~30 FPS) on volumes that exceeds GPU memory. We can talk about output-sensitive algorithms since render time is directly linked to the size of the output (the screen) and not to the size of the input (the image stack).

4.3. 3D visualization perception

The figure 3 shows two multi-view frames computed for an HD (16:10) autosteroscopic 8-views display, with a volume of $5520 \times 7000 \times 800$ RGBA32 pixels and $[\Delta_x, \Delta_z] = [4, 1]$.

The choices for the values of Δ_x and Δ_z were made in order to maximize the perceived depth. The tests highlight the fact that the physical distance between two slices needs to be considered in the choice of the value Δ_z. In one case, with a large distance between slices, whatever the value of Δ_z, the visualization is not smooth. In the other case, a very short gap combined to a too small Δ_z do not provide the feeling to move in the depth of the volume. In the same way, using a high value for Δ_x cause a major visual discomfort in addition to a non-desired 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, the perception was only visually assessed by the authors and cannot be considered as sufficient. A rigorous statistical study on a large sample of users is required. In order to increase the quality of the system, the study should focus on the feedback of a pool of users, on their perceptions on multiple samples with different settings.

5. CONCLUSION

In this article we introduced a proof of concept to virtualize a microscope on autostereoscopic displays. To achieve this, we propose a visualization driven pipeline based on a GPU rendering using a multilevel, multiresolution data structure in order to create a multi-view frame. Thanks to this architecture, the user is able to perform an interactive navigation through multiresolution images of massive datasets.

6. REFERENCES

[1] 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*, vol. 340, no. 6139, pp. 1472–1475, 2013.

[2] U. Catalyurek, M. D. Beynon, C. Chang, T. Kurc, A. Sussman, and J. Saltz, "The virtual microscope," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, no. 4, pp. 230–248, 2003.

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

[4] 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.

[5] "Deep zoom," `https://www.microsoft.com/silverlight/deep-zoom`, [Online; accessed 07-February-2017].

[6] "Openseadragon," `http://openseadragon.github.io`, [Online; accessed 07-February-2017].

[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, pp. 34–43. 2013.

[8] M. Hortsch, "From microscopes to virtual reality – How our teaching of histology is changing," *Journal of Cytology & Histology*, vol. 4, no. 3, 2013.

[9] N. S. Holliman, N. A. Dodgson, G. E. Favalora, and L. Pockett, "Three-dimensional displays: A review and applications analysis," *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 362–371, 2011.

[10] 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*. 2009, pp. 15–22, ACM.

[11] 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*, vol. 18, no. 12, pp. 2285–2294, 2012.

[12] 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*, vol. 33, no. 4, pp. 50–61, 2013.

[13] A. Dubois, A-S. Hérard, B. Delatour, P. Hantraye, G. Bonvento, M. Dhenain, and T. Delzescaux, "Detection by voxel-wise statistical analysis of significant changes in regional cerebral glucose uptake in an APP/PS1 transgenic mouse model of Alzheimer's disease," *NeuroImage*, vol. 51, no. 2, pp. 586–598, 2010.

[14] 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)*, 2013, pp. 43–51.

[15] 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)*, 2012, pp. 1–8.

[16] G. R. Jones, D. Lee, N. S. Holliman, and D. Ezra, "Controlling perceived depth in stereoscopic images," 2001, vol. 4297, pp. 42–53.

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