Simulating Depth Perception in Virtual Microscopy

N. Courilleau, Y. Remion, Laurent Lucas

SIMULATING DEPTH PERCEPTION IN VIRTUAL MICROSCOPY

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

1. Université de Reims Champagne-Ardenne, CReSTIC EA3804, France
2. Neoxia, France

ABSTRACT

3D datasets production capacity in bioimaging has widely evolved in recent years. This trend also results in a growing demand of more suitable display procedures. In this paper, we propose a new virtual microscopy approach aiming at recovering the third dimension, in order to get closer to analogue microscopes. For this purpose, we rely on multi-view autostereoscopic display with off-centered parallel virtual cameras to ensure 3D perception for a more realistic user experience. Also, this approach handles very large volume data size thanks to an out-of-core data management structure which offers interactive navigation by using complete GPU algorithms.

Index Terms— Virtual microscopy, 3D display, GPU rendering.

1. INTRODUCTION

Nowadays, the volume of 3D datasets tends to increase in nearly all application fields. Such an observation is especially true with the current image acquisition technologies used in biology [1]. Despite the large amount of generated data, efficient visualization tools are needed to improve three-dimensional structures understanding. Among the possible methods available to overcome this problem, virtual microscope is a tool of choice in the biological field.

A virtual microscope could be defined as a system simulating the observation of microscopic samples on a computer. It aims to mimic a conventional microscope, enabling observation, navigation [2, 3], and annotation of virtual slides [4]. The latest virtual microscopy developments [5, 6, 7] have addressed the issue of huge data size. However, as of today, another crucial issue has not been considered: users were frustrated by the availability of only one single plain focus, and the induced loss of three dimensional perception [8]. To improve the user experience quality and to recover the depth perception, we propose to use autostereoscopic display [9] for the 3D visualization [10, 11]. To complete the system, we allow the users to freely navigate and zoom in or zoom out in the whole volume rather than in a single slide.

To address this problem, we rely on an out-of-core data management architecture to handle the large volume of data. In most cases, a resolution level pyramid is created where each level is subdivided into data blocks (bricks). Conversely to tree based structures (like Gigavoxels [12]) that can lead to deep tree traversal to access data, we focus our work on the method proposed by Hadwiger *et al.* [13]. They provide a virtual memory approach with a multi-level, multi-resolution page table mechanism. Validated with a concrete application with interactive exploration of petascale volumes by Beyer *et al.* [14], this structure offers constant access time between all resolution levels.

Inspired by this out-of-core method, we propose a visualization-driven approach which generates stereoscopic multi-view frames on GPU in interactive time. The aim is to recover the depth perception in a most realistic way with the use of off-centered parallel virtual cameras focused on a shared point of interest.

A brief overview of the system and data access is provided in section 2. After a detailed description of our virtual microscope approach in section 3), experimental results are presented and discussed in section 4. Finally, section 5 provides concluding remarks and perspective works.

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.

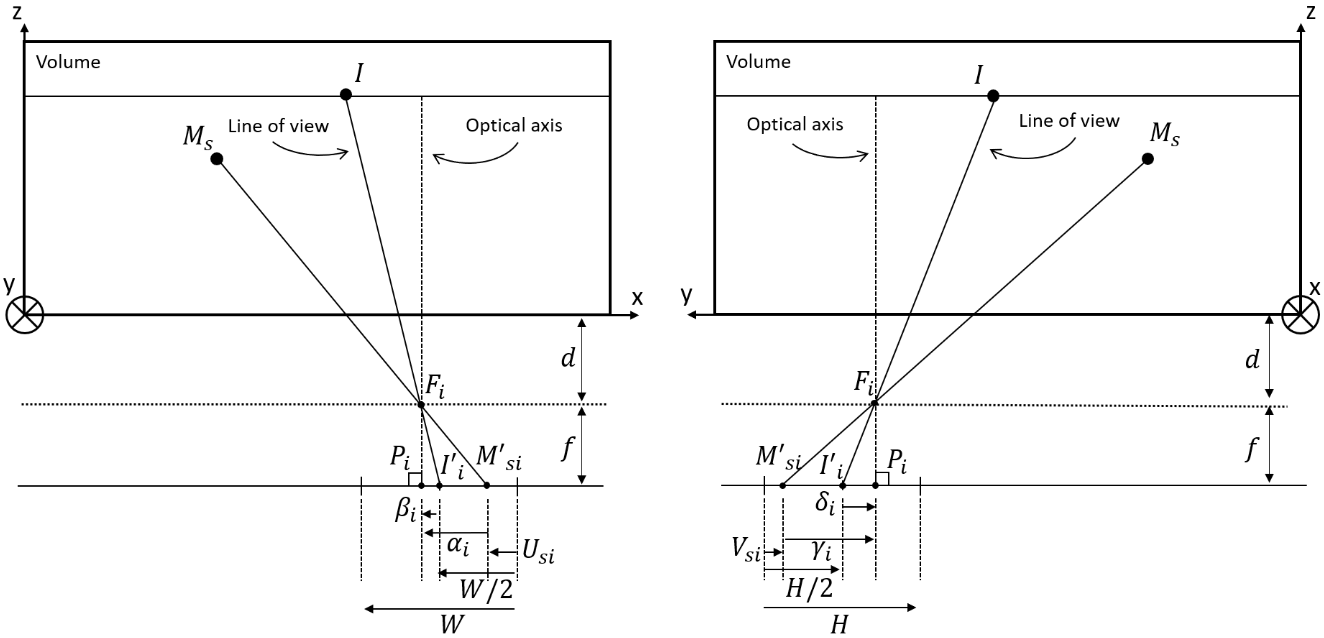
2. SYSTEM OVERVIEW

The proposed system fully runs on GPUs [15] and is composed of the virtual microscope and the out-of-core data management. The whole pipeline is called *visualization-driven*. According to the camera position given by the virtual microscope, data is accessed using the out-of-core data manager. We base our approach on the one proposed by Hadwiger *et al.* [13]. The data is represented with a multi-resolution bricked pyramid and addressed using a virtualized page table hierarchy. When a chunck of data is missing in the hierarchy, a brick request is raised and handled by the CPU. Bricks are then loaded from its own cache or a mass storage to the GPU. This method allows to get the data on-demand and to address very large volume of data. The whole process is triggered by the virtual microscope depending on the required data.

Data addressing. The voxel access is performed using a virtual volume representation. A pair $[l, p]$ is composed to get through the virtualization hierarchy. The level of detail is determined by l while $p \in [0, 1)^3$ is the 3D normalized coordinate vector of the requested voxel position. Using normalized coordinate vector allows the use of the hardware trilinear interpolation. Finally, the pair $[l, p]$ is sent to the GPU out-of-core memory manager which sends back three possible answers: the requested voxel itself, an *empty* status if the brick contains no values, or *unmapped*, warning the system that the brick is not in cache and needs to be fetched.

3. VIRTUAL MICROSCOPY

The notion of depth perception is usually lost in conventional 2D virtual microscopy. In this work, we provide an approach to recover this perception somehow available with genuine microscopy of thick slices. To achieve this goal, we use multi-view autostereoscopic displays. Such displays rely on N multi-view images shot according to a specific geometry implying off-centered parallel cameras. The number of views N is directly dependent on the display. In our application, the multi-view image generation follows three steps: *i)*



(a) (b)

Fig. 1. **Representation of the top and side views of the virtual scene and camera geometry.** (a) top view, (b) left side view (with the top view as reference). For both views, the volume is represented on the top and the camera focal points F_i and sensors are at the bottom. Off-centered parallel cameras are used with a common point of interest I. I_i' is the projection of I on the view i given the focal point F_i. The line F_i, I is the line of view i implying $I_i' \in (I_i' I)$.

projecting the volume slices on the N views according to the specific multi-camera geometry, *ii)* applying an alpha blending algorithm in the associated view for each slice projection, and *iii)* compositing the N views, *i.e.* creation of the final multi-view image.

The specific geometry implies that N aligned cameras with parallel optical axes and camera line of view converge in a single 3D point (I in Fig. 1). We chose to align the N cameras along the volume x-axis for two reasons: to get volume slices parallel to the sensors and the views, and to get the view lines parallel to the camera baseline, therefore parallel to the volume x-axis. The convergence point I must be displayed at the center of the display. The views are captured in a rectangular region of interest (ROI) of the sensor which is centered on the line of view intersecting each point I_i' (Fig. 1). This implies that slices are projected on views according to a ROI which is also centered on the camera line of view. Thus, projecting a given volume slice on a view relies on identifying the involved slice ROI projection on the view ROI.

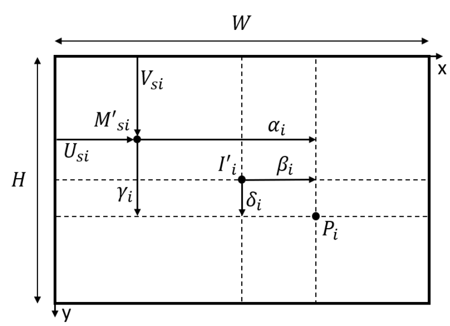


Fig. 2. **Screen space representation.** A view is considered as a rectangular ROI in the camera sensor. On this figure, the view has a $W \times H$ size (display size), and is centered at I_i' around the camera line of view. M_{si}' is the projection of the point M_s on the view at coordinates $[U_{si}, V_{si}]$ where i refers to the i-th view and s the slice in which M_s lies.

3.1. Views generation

The first step of the pipeline begins with the determination of all the parameters required to project the volume slices on the views. In order to meet autostereoscopic multi-view requirements, each virtual camera must aim at a shared point of interest I whose projections are centered in each of the N view (Fig. 1), numbered $i \in [0, N-1]$. I_i' is I projection (Fig. 2). As shown on figure 2, a view i has a $W \times H$ size, implying that the screen space coordinates of I_i' are always $[\frac{H}{2}, \frac{W}{2}]$. Considering that a slice s is projected on a view i, each pixel coordinates of this view are $[U_{si}, V_{si}]$. Identification of the slice s ROI projected on the view i relies on the geometric relation between the space coordinates of voxels M_s and the pixel coordinates $[U_{si}, V_{si}]$ of their projection M_{si}' in the view i. Let consider the shooting geometry is set: I, F_i are chosen as well as the focal length f (Fig. 1). The calculation of the relation between M_s spacial coordinates and M_{si}' then rely on f and the coordinates of $I = [I_x, I_y, I_z]$ and $F_i = [F_{ix}, F_{iy}, F_{iz}]$. Because the point I is not necessarily aligned on the camera i optical axis, its projection I_i' is shifted from P_i the orthographic projection of F_i. These shifts, called β_i for x-axis and δ_i for y-axis. In a first step, they allow to calculate the $I'_i = [I_{ix} + \beta_i, I_{iy} + \delta_i, -d - f]$ coordinates knowing that $I'_i F_i \| F_i I$:

$$[\beta_i, \delta_i] = \begin{cases} (F_{ix} - I_x) \times \frac{f}{d+I_z} \\ (F_{iy} - I_y) \times \frac{f}{d+I_z} \end{cases} \quad (1)$$

where d stands for the camera – volume distance on z-axis and f for the focal length. Because $[\beta_i, \delta_i]$ are related to the view i and its focal point, they need to be calculated only once for a view.

Once this step is done, the projection M'_{si} of $M_s = [x, y, z_s]$ on the view i can be computed. This new step consists of finding M'_{si} pixel coordinates. Starting with $M'_{si} = [F_{ix} + \alpha_i(x), F_{iy} + \gamma_i(y), -d - f]$ space coordinates and knowing that $M'_{si} F_i \| F_i M_s$, $[\alpha_i(x), \gamma_i(y)]$ are calculated as follow:

$$[\alpha_i(x), \gamma_i(y)] = \begin{cases} (F_{ix} - x) \times \frac{f}{d+z_s} \\ (F_{iy} - y) \times \frac{f}{d+z_s} \end{cases} \quad (2)$$

Finally, the pixel coordinates $[U_{si}, V_{si}]$ of M'_{si} are recovered. Based on figure 2, $[U_{si}, V_{si}]$ are expressed as follow:

$$[U_{si}, V_{si}] = \begin{cases} \frac{W}{2} + \beta_i - \alpha_i(x) \\ \frac{H}{2} + \delta_i - \gamma_i(y) \end{cases} \quad (3)$$

Equations (3) express the M'_{si} pixel coordinates from M_s space coordinates. Reverting those equations will result in expressing the space coordinates $[x, y, z_s]$ in slice s of any pixel of the view i:

$$[x, y] = \begin{cases} F_{ix} - (\frac{W}{2} + \beta_i - U_{si}) \times \frac{d+z_s}{f} \\ F_{iy} - (\frac{H}{2} + \delta_i - V_{si}) \times \frac{d+z_s}{f} \end{cases} \quad (4)$$

The focal point F_i and $[\beta_i, \delta_i]$ are known and have been calculated previously. The sizes W and H are related to the display definition. Note that z_s is the z-axis coordinate of the slice s projected on the view i. Once $[x, y]$ are calculated, their values must be normalized to be used in the addressing structure (see Sec. 2). Thanks to GPU threading, and texture handling, equations (4) are used to compute each pixel $[U_{si}, V_{si}] \in [0, W) \times [0, H)$ in a dedicated thread by alpha blending (see Sec. 3.2) of its previous value and the 3D interpolation of slice s at $[x, y]$.

Multi-camera geometry. A shooting geometry is required to fit the display geometry and to ensure non-distorted 3D perception. This is the reason why off-centered parallel geometry is needed. Moreover, non-distorted 3D perception implies homothetic connection display and camera geometry. The display geometry implies N off-centered (or favorite) user eyes positions O_i lying in front of the screen. They are aligned with the display lines at a distance D from the screen. Those eyes positions are spread by the average human binocular distance ($b = 65\,\text{mm}$). The display and camera geometries are compared according to the eyes O_i – centre of display C, and the focal points F_i – convergence point I relative positions. Non-distorted 3D perception requires $O_i C \| F_i I$ with $i \in [0, N)$. Knowing the display geometry (Fig. 3):

$$O_i = \begin{bmatrix} (i - \frac{N}{2})b \\ 0 \\ 0 \end{bmatrix} \text{ and } C = \begin{bmatrix} 0 \\ C_y \\ D \end{bmatrix}$$

Assuming the convergence point I and the focal length f are chosen by the user, this leads to:

$$[F_{ix}, F_{iy}] = \begin{cases} I_x + (i - \frac{2}{N})b \times \frac{d+I_z}{D} \\ I_y - C_y \times \frac{d+I_z}{D} \end{cases}$$

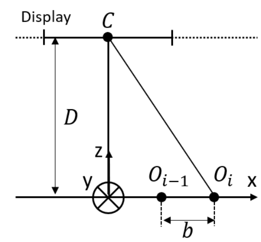


Fig. 3. **User favorite position in front of the display.** The O_i are the focal points of the user favorite eyes positions. These points are aligned on the x-axis, shifted by the average binocular distance b (65 mm). The display is positioned at a distance D from the user.

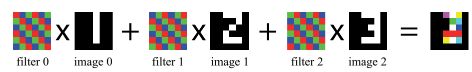


Fig. 4. **Multiplexing filters application.** [16]

Slice selection. The most important slice is the one containing the point of interest. Consequently, the slice I_z is always projected on the first view. However, the slices projected on the other views need to be determined. In order to recover the depth perception, we use the slice behind I_z. Determining which slice to consider can be done using a parameter Δ_z. For a view i, the associated slice will be $s = I_z + i \times \Delta_z$. The choice of Δ_z value is at user's discretion. Nevertheless, the data characteristics need to be studied to select the best value for Δ_z (see Sec. 4).

3.2. Alpha generation

The alpha generation is performed using a two-step process. First, the pixels of each view are assigned an alpha value. This step can be compared to a standard classification where, for instance, four classes could be used: background, interest tissues, non-interest tissues, and artefacts. The user defines an alpha value for each class, and each pixel will be assigned the apha value corresponding to its closest class alpha value. What is noteworthy is that the choice of the classification algorithm to use will have an impact on the result quality, as well as on the computational time needed to classify all pixels

Once the alpha values are set, an alpha blending process is performed, consisting in applying a back-to-front alpha blending following the painter's algorithm. This process determines the contribution of the back views, according to a certain amount, in the front views. The depth perception is therefore recreated.

3.3. Compositing

The compositing is the final step of the pipeline, providing a final multi-view frame $\mathcal{F}$. This frame is generated using the N pre-build views $\mathcal{V}^i$, and is composed by using filtering masks $\mathcal{M}$ on the views (Fig. 4). These masks are binary filters ($\mathcal{M}^i = \{0, 1\}^3\,[\mathbb{Z}^2]$), and determine, for all views, which color components will contribute to the final frame composition. This step can be expressed as $\mathcal{F}_c = \sum_i \mathcal{V}^i_c\ \mathcal{M}^i_c$ where $c \in \{R, G, B\}$. The filtering masks mechanism is dependent of the display. This process may differ, and take more or less computing time, depending on the used hardware.

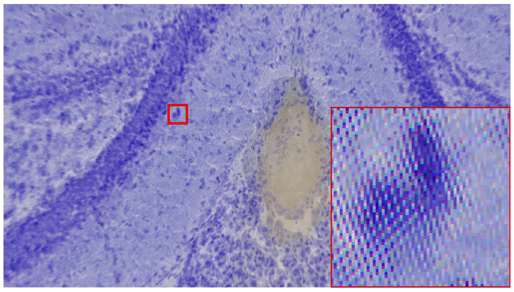


Fig. 5. **Multi-view frame generated.** Example resulting from the histological volume.

4. RESULTS AND DISCUSSION

The experiments were made using three different datasets:

- (a) A 114 histological slices stack of a mouse brain with a resolution of 64000×50000 RGBA pixels ($1.459\,\mathrm{TB}$);
- (b) A $2160 \times 2560 \times 1072$ volume with grayscale 16 bits voxels ($11\,\mathrm{GB}$) of a primate hippocampus from a light sheet microscope;
- (c) A 645 blockface slices stack of the mouse brain (a) with a resolution of 823×202 RGBA pixels ($428\,\mathrm{MB}$).

The dataset (a) is the proof of concept of our system (Fig. 5). The dataset (b) demonstrates that our system can be applied to different modalities. Finally, the dataset (c) is used to compare the result with the dataset (a), acknowledging the former has a strongest anisotropy than the latter. The display used was a HD (16:10) autosteroscopic 8-views display with a 1920×1200 viewport and requiring a distance – display of $2\,\mathrm{m}$. The same test scenario was applied on the three datasets, and consists of a zoom-in, a pan navigation, then a navigation in the stacks. Table 1 shows the average time recorded to generate a complete multi-view frame.

	(a)	(b)	(c)
Generate frame	11	12	7
Set alpha	8	5	8
Alpha blending	5	5	6
Compositing	5	5	5
Sum	29	27	26

Table 1. Average rendering time in milliseconds of a frame for three different datasets.

Multi-view frame generation. The first noticeable fact is the independence between the frame rendering time and the volume data size or the acquisition modality. Considering the time allowed to other required processes (out-of-core data management) the method offers a real-time navigation with an average of 30 fps. In the presented test, the rendering pipeline was performed at each frame to simulate the worst case scenario in which this needs to be done every time. Nevertheless, in concrete situation, the whole pipeline should be triggered only when a change is detected in the camera position. The first step recorded is the volume projection on the views only. The average time to generate the views was around 11-12 ms. The dataset (c) took less time to project because of its size, as the volume did not fill entirely the views. In fact, on pixels in the outside borders of the views, we try to project information that is outside the volume. In that case, we are outside the bound of normalized volume sizes and there is no need to continue to process the projection; therefore the frame are generated faster.

The *set alpha* step consists in computing the Euclidean distance between the voxels and different clusters. In the tests, we used five clusters computed beforehand using a k-means algorithm. The choice in the classification algorithm is of importance: the computation runtime can affect the time to render a view, therefore the navigation fluidity. The difference noticed in the results is related to the data type differences between the volumes. While the dataset (b) uses grayscale pixels, the others use RGBA pixels. Therefore, the pixels assignment is performed faster on (b) than (a) and (c).

Depth perception. As stated in section 3.1, the value Δ_z is of importance in the proposed method, as a non optimal value may induce an uncomfortable visualization. During the tests, we noticed that the differences between two slices were small in the dataset (b). The depth perception was recovered by using $\Delta_z = 3$. However, this also underlines that visualizing a grayscale works better with high-contrast or color data. An extra step may be required to colorize the data via LUTs. The extreme case happened with the dataset (a). As expected, the volume anisotropy was significant and required to use a $\Delta_z < 1$. Depending on Δ_z value, the number of slices the user can see in a multi-view frame may be drastically reduced, however the final result is softened. Having $\Delta < 1$ is possible because we are addressing the data using normalized coordinates. The used out-of-core data structure relies on the GPU texture memory and allows trilinear interpolations. However, misusing this value may reduce the depth effect and in turn make the approach irrelevant. Yet, the data anisotropy is an important factor to consider when using this approach.

Camera positioning. In the given approach, the views are positioned on the same plane as the slices. This could be improved using an orbital camera. One could be able to analyse the volumes in a different view angle. However, the data anisotropy would be a major concern and would require a better alternative than simply using a Δ_z parameter.

To test our method, an autostereoscopic display was used, where the user has to move in front of the display to move inside the multi-view frame. With current technologies, it could be interesting to use the same approach on tablet devices, as it may be more convenient for the user to move the device left and right than to move in front of a screen. Finally, all tests were performed and assessed by the authors. The method is ready for statistical studies with the end-users, using lightsheet or histological datasets and using different parameters.

5. CONCLUSION

We introduced a proof of concept to virtualize a microscope using off-centered parallel cameras and an autostereoscopic display. With the help of current GPUs, this approach allows the user to interactively navigate through multi-resolution images. This allows data visualization similar to an analogue microscope visualization by recovering the depth perception. Taking advantage of the out-of-core data management, the proposed approach works independently of the data size, from MB to several TB. In addition, the concept offers a similar user experience accross different volume modalities, without significant differences in rendering time. In the future, the development of an approach offering the ability to use an orbital cameras would be of great benefits to this method.

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] "Deep zoom," `https://www.microsoft.com/silverlight/deep-zoom`, [Online; accessed 24-September-2018].

[3] "Openseadragon," `http://openseadragon.github.io`, [Online; accessed 24-September-2018].

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] J. Sarton, N. Courilleau, A. Hérard, T. Delzescaux, Y. Remion, and L. Lucas, "Virtual review of large scale image stack on 3d display," in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 2229–2233.

[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)*, 2012, pp. 1–8.