International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 4, Number 2, pages 369–391, July 2014

Communication Platform for Image Analysis and Sharing in Biology

Masahiko Morita, Takehiro Tawara, Masaomi Nishimura, Shin Yoshizawa
Image Processing Research Team, Center for Advanced Photonics, RIKEN,
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Bukai Chou, Ippei Kuroki
Advanced Center for Computing and Communication, RIKEN,
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Takashi Ijiri, Yuki Tsujimura
Image Processing Research Team, Center for Advanced Photonics, RIKEN,
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Ryutaro Himeno
Advanced Center for Computing and Communication, RIKEN,
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Hideo Yokota
Image Processing Research Team, Center for Advanced Photonics, RIKEN,
2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Received: February 16, 2014
Revised: May 7, 2014
Accepted: May 26, 2014
Communicated by Takio Kurita

Abstract

Image analysis is crucial to medical and biological applications. Recent advances in imaging technology have led to the demand for processing and visualizing a large amount of three-dimensional (3D) biomedical images. In addition, cloud computing has become popular for managing big data. Unfortunately, conventional image-processing systems either lack cloud computing services or advanced 3D processing abilities. In this paper, we present a novel cloud-based system for sharing, processing, and visualizing 3D biomedical images. Our system employs a standard web browser as a client interface that interactively communicates with high-performance servers. Thus, an inexpensive tablet PC without an advanced graphics processing unit (GPU) can be used for 3D image processing and visualization. Our system provides the sharing of limited software and hardware resources, and it allows for effective collaboration between researchers. We demonstrate the applicability and functionality of the system by examining typical case studies on biomedical images. We also examine the performance of our system numerically.

Keywords: Cloud System, Biomedical, Image Analysis, Image Processing

1 Introduction

The development of image-processing tools has become popular in biological and medical applications; in addition, rapid advances are being made in biomedical imaging technology [20, 57, 19]. In particular, tools that can manage large amounts of volumetric data (i.e., three-dimensional (3D) images) are required in advanced applications such as surgery simulations based on Computed Tomography (CT) volumes and quantitative analysis of live-cell images in molecular cell biology.

Many useful computational tools have been developed for biomedical image analysis, including ImageJ [53], CellProfiler [11], and Fiji [56]. However, most existing tools are stand-alone systems that are not equipped with a communication platform to facilitate collaboration among users, or the sharing of images, software, or central processing unit/graphics processing unit (CPU/GPU) resources. Moreover, these conventional tools are either designed for a specific image-processing task or lack the 3D image processing functions required for advanced applications.

Thus, biologists often purchase or develop proprietary software and hardware for every research project/group. Conversely, developing proprietary image processing systems has been become difficult because of the expensive hardware, such as a GPU clusters and/or supercomputers, and related software that are required for large-scale 3D image analysis.

(a) (b) (c)

Figure 1: (a) Intracellular volume rendering of plasma and nuclear membranes and mitochondria using our system. (b) Medical CT volume processing using our system via a tablet PC. (c) Our system allows volume rendering with curved cross-sectional images.

Our approach. In this paper[1], we propose a new cloud-based communication system for biomedical images that allows collaboration among users and facilitates the effective and seamless management of images, software, and CPU/GPU resources. Our system consists of a set of computer servers and databases, such as a web server, a biomedical image database, a set of CPU/GPU servers, and a remote computing server in order to provide the cloud computing services required for collaboration, management, and large-scale 3D analysis of biomedical images.

The key idea of our system is to employ standard web browsers (e.g., Internet Explorer, Google Chrome, and Mozilla FireFox) as the user interface. We developed a novel cloud-based interactive technique to provide user interaction and rendering results through the web browser; therefore, our system can control the data management, visualization, and processing services through the web browser via inexpensive hardware devices such as tablet PCs, mobile phones, and low-cost PCs without advanced GPUs, see Fig. 1. The images are uploaded to the biomedical image database using any standard file transfer protocol (FTP) software through the web server. The user selects a CPU/GPU server with images in the biomedical image database

[1]This is an extension of our previous work [44]. The main differences from [44] are more detailed descriptions and performance evaluations (Sections 6 and 7) of our system.

through the web server. The CPU/GPU server is equipped with a set of image processing software (including our own [23], open source software, and commercial software). The remote computing server controls user interaction via background processes in our system.

Our system also includes a unified method for managing biomedical images for collaborative research, and an intuitive user interface for advanced processing of complex 3D biomedical images. We demonstrate our system by examining several case studies on biomedical images. The contributions and benefits of the paper are summarized as follows:

- A new communication platform for effective biomedical image analysis and management.
- An efficient method for sharing limited software and hardware resources.
- Support for a unified approach to collaborative research 3D images.
- A novel cloud-based interactive technique using a standard web browser for user interaction and rendering results.
- A quantitative performance evaluation of the proposed cloud-based interactive technique.
- Three case studies for typical biomedical image analysis: unsupervised segmentation and filtering, supervised machine learning, and interactive volume visualization and processing.

Paper organization. The remainder of this paper is organized as follows. Section 2 presents related work on the communication systems of biomedical images. We describe our platform overview in Section 3. Our cloud-based interactive technique is explained in Section 4. Section 5 provides case studies on biomedical images using our system. We evaluate the performance of our system in Section 6, and present a simple user study on our system in Section 7. We conclude the paper in Section 8.

2 Related Work

Image processing, including pattern recognition and computer vision, is one of the most intensively studied fields in computer science; therefore, a large number of software systems have been proposed over the past 60 years. Image analysis in biology has become popular more recently compared with other scientific fields, such as astronomical physics and geology. We briefly review the conventional biomedical image processing systems and screen-updating techniques related to our system, see [18, 20] and references therein for excellent surveys on general biomedical image processing tools.

2.1 Stand-Alone Systems

ImageJ [53] is a common tool for image analysis in biology, see [57] for its historical survey (although the Java programming language has less than 18 years of history). Fiji [56] is a set of ImageJ software resources focused on biomedical image analysis. Cell Profiler [11] is a specialized tool for two-dimensional (2D) segmentation and quantitative analysis; its extension [31] can manage and analyze multidimensional images using machine learning techniques. Icy [17] and BioImageXD [33] also include a multidimensional image processing ability; Icy optimizes its processing by multi-cores with OpenCL implementation, and BioImageXD provides deconvolution and segmentation functions. Table 1 summarizes these conventional systems in terms of their functionalities for extensibility, 3D processing and visualization, and user communication. All of these systems are very useful, but they are stand-alone systems that do not provide any data sharing or database functions. Any of these systems can be used as image processing engines in our CPU/GPU servers. In other words, our system is not a competitor to any these stand-alone systems; rather, it is a collaborative engine where these systems can complement each other.

	Extensibility	3D Processing	3D Visualization	User Communication
ImageJ [53]	High	Low	Yes (plugin)	No
Fiji [56]	High	Middle	Yes (plugin)	No
Cell Profilers [11, 31]	Low	Low	No	No
Icy [17]	Middle	Middle	Yes (plugin)	Yes
BioImageXD [33]	Middle	Middle	Yes	No

Table 1: Conventional systems for biomedical images.

2.2 Cloud-based Systems

There are many general cloud-based systems, such as remote desktops (Microsoft and Google), Citrix (Proprietary software), Virtual Network Computing (VNC, Open source), RVEC (Fujitsu), GRID (NVIDIA), and ORBX.js (Mozilla). Furthermore, many cloud-based systems have been proposed for specific tasks, such as CT reconstruction using PC clusters [63], as well as for particular techniques, such as transferring compressed images and videos [36, 39, 37]. Our system is designed for general purposes in biomedical image visualization, processing, and analysis; see Fig. 1 (c) for a representative example of 3D volume rendering with curved surface image visualization by our system.

In contrast to the specialized system, OMERO [2] and Bisque [35] are both web-based systems that focus mainly on data management for multidimensional images accessed and used via a web browser (OMERO also has its own client) on Mac, Windows, and Linux operating systems. Unfortunately, OMERO and Bisque are only equipped with simple image processing functions. PSLID [45] is a database for 3D cell images that works on a web browser. Shen et al. [59] proposed a cloud-based classification tool for multidimensional biomedical images with a web browser interface. Wu et al. [66] proposed a cloud-based image processing system that efficiently processes large data sets with distributed parallel computing. This system is designed only for non-interactive image processing and does not provide visualization functions. Although the pattern recognition methods implemented in [59] are powerful, such as a genetic algorithm and Support Vector Machine (SVM), and PSLID includes an image classification function, all the conventional cloud-based systems mentioned in this paragraph [2, 35, 45, 59, 66] lack either 3D visualization or advanced 3D image processing abilities. In addition, it is difficult to incorporate conventional image processing tools [53, 31, 33] into these systems [2, 35, 45, 59, 66].

Engel et al. [21] proposed an interactive visualization system that combines local and remote hardware for medical CT images. Vazhenin [65] proposed a cloud-based system that allows the general populace access to medical health information. The idea of these system architectures is similar to ours, and these systems include 3D visualization. However, their purpose is very different, for example, our system is directed at scientists and engineers who might not have sufficient knowledge about detailed image processing algorithms, but who would want to apply advanced computations on their images. Therefore, our system is capable of incorporating image processing tools; in addition, the systems described in [21, 65] only provide visualization and are not equipped with advanced image processing tools. Table 2 summarizes these conventional cloud-based systems for biomedical images and our system.

	Image Sharing	3D Processing	3D Visualization	Call Stand-Alones
X-ray Tools [63]	No	Limited Use	Yes	No
Wu et al. [66]	No	Low	No	Middle
OMERO [2]	High	No	No	No
Bisque [35]	High	Low	Yes	No
PSLID [45]	Limited Use	Low	No	No
Shen et al. [59]	No	Limited Use	No	No
Vazhenin [65]	No	No	Yes	No
Engel et al. [21]	No	No	Yes	No
Our System	Middle	High	Yes	High

Table 2: Conventional cloud-based systems for biomedical images.

2.3 Updating Screens

Displaying screen images has been well studied in computer graphics and related research fields. Recent advances in GPU and multi-core CPU make tile-based approaches usable for parallel processing of a screen, e.g., NVIDIA's SLI [68] and ATI's Crossfire [3]. Both multi-GPU environments support tile-based rendering at the level of the graphics driver. The Split Frame Rendering mode of NVIDIA's SLI horizontally splits the screen into two regions such that the rendering workload becomes identical. The splitting position is dynamically changed by the performance feedback. The Super Tiling mode of ATI's Crossfire splits the screen into tiles based on the number of GPUs. See [46] for a survey of tile-based approaches in both stand-alone and cloud-based systems.

Adaptively adjusting the screen quality has also been well studied in order not to disturb human-computer interactions. In such techniques, the quality of the rendered image becomes low during user interactions. Progressive and multiresolutional rendering techniques are popular for achieving adaptive adjustments. For example, path-tracing [32] employed progressive image synthesis by stochastically sampling eye rays on a screen. QSplat [55] used multiresolutional point-based rendering for large meshes.

These techniques are performed to avoid rendering latency for large-scale data in a stand-alone system. On the other hand, network latency is a bigger problem for our purpose, which is interactive image processing via a network.

Network-based techniques. Most of existing remote desktop solutions, such as VNC [54] and Remote Desktop Protocol [42], are firmly designed for traditional computer desktop architecture, which constructs with graphic primitives such as a window and a popup menu. Such remote desktop solutions update a desktop screen based on each graphic primitive. With these systems, it is difficult to exploit hardware acceleration that is essential for our biomedical image processing. Furthermore, the compression techniques [13, 60, 10, 40] employed in general video streaming may introduce artifacts because of their high compression rate, and such artifacts are not desirable for biomedical image applications.

Cedilnik et al. [12] proposed a parallel rendering framework, most likely an application programming interface, for large data visualization. Although this framework has a screen division functionality and can be used for parallel volume rendering [43], it cannot be applied to accelerate visualization of existing software such as ImageJ and Fiji. Engel et al. [22] presented an image-streaming framework for the remote visualization of large-scale volume data sets. Unfortunately, modifications to the application software are required in order to use this framework.

Lu et al. [39] proposed a virtual screen system that employs a strategy similar to our cloud-based interactive technique in terms of updating screens. The system detects the difference between the previous and the current screens, and divides the screens as local windows. In addition, the system classifies the windows into skip, text, and picture-blocks, and applies different coding schemes for each block. Thus, the system clients require special browser software in order to decode the blocks. Whereas our technique focuses on the interactions for processing biomedical images using general web browsers, Lu et al.'s system focuses on text information using their system for both servers and clients.

3 Platform Overview

Our system consists of a set of computer servers, and it provides cloud computing services. Here, we briefly describe the types of services available in our system and how servers are employed to construct the system.

3.1 System Services

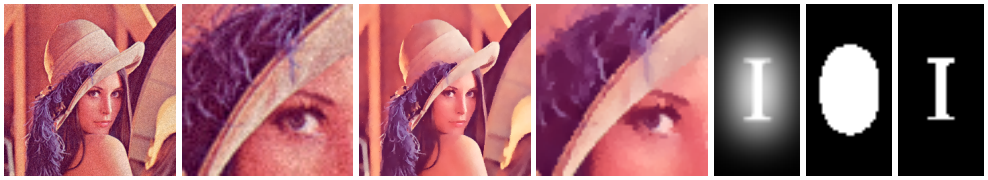
Our system provides 3D image data management, visualization, and processing services through a web browser. The main features of the services are as follows.

Data management. The system allows users to store, search, and delete images with information that describes the experimental conditions for imaging and explanations of the data, e.g., a description of what is observed in the image, microscopy parameters, image aspect-ratio, fluorescent staining methods, etc. The

system also includes user accounts and group management that controls permissions for the groups and users in order to support collaborative research and to manage software licenses.

Visualization and processing. The user can employ any conventional image processing tool (e.g., [53, 31, 33]) that the administrator installs on the CPU/GPU servers. In the default setting, we install our software VCAT [23] as a core image processing engine. VCAT provides a set of plug-in image processing functions, a macro design interface, and visualization methods. Standard three-axial views and volume rendering with multi-dimensional transfer functions [34] are implemented in VCAT; see Fig. 1 (a) for an example of volume rendering for intracellular organelles and Fig. 2 for simple filtering examples.

Currently, the VCAT plug-ins include standard noise reduction filters (Gaussian, median, adaptive median [26], bilateral [64], and non-local means [9]), morphological operations [26], fast edge-aware filters [52, 67, 24], feature extraction methods (HoG [16], HLAC [49], Saliency [30], and SIFT [38]), SVM via the sequential minimal optimization [51], traditional (Otsu [48], graph cuts [8], region growing [1], and mean shift [15]) and interactive [28, 29] segmentation methods, meshing segmented volumes (tetra- and hex-dominant tetra meshes, see Fig. 3), and other functions such as labeling, counting, and measuring segmented regions for quantitative analysis. VCAT employs a multi-page TIFF format that is translated from standard biomedical image formats using the Bio-formats library [6]. Obviously, any other open-source/free/commercial software can also be employed and their licenses can be managed by the system appropriately with user accounts.



(a) (b) (c) (d) (e)

Figure 2: Image processing examples using VCAT. Since biomedical images are often corrupted with noise, denoising is important for processing such images. The image (a) was corrupted with additive Gaussian noise, artificially. The image (b) demonstrates a result of denoising with edge-preservation, which was obtained by applying Domain Transformations (DT) [24] to the corrupted input image (a). Biomedical images may also consist of a gradation of background intensity due to their imaging conditions. The image (c) mimics a such condition, and applying the Otsu method [48] to (c) gave a undesired segmentation result (d). Employing Top-Hat transformation [26] prior to the Otsu method provided a better segmentation result (e).

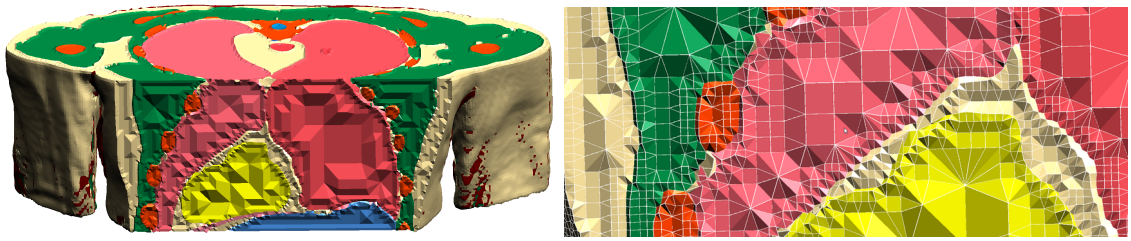


Figure 3: Volumetric mesh generation for a multi-material biomedical volume. Each organ was segmented and labeled by a simple thresholding with manual corrections. The hexa-dominant tetrahedral mesh was generated on the multi-labeled segmented organ volume by VCAT with the meshing method [47]. Such volumetric meshing is useful for mechanical and biochemical computations in surgical or metabolic simulations from real-world images.

3.2 System Architecture

To provide the services mentioned in the previous section, the proposed system consists mainly of the components described in this section (CPU/GPU servers, clients, databases, and web server). The key components

required to achieve interactivity with our system are remote computing and capturing servers in the web server and CPU/GPU servers, respectively. Our computational schemes built on these servers are described in Section 4.

Clients. Any standard PC, including tablet PCs, mobile phones, and inexpensive PCs without advanced GPUs.

CPU/GPU servers. High-performance computers with advanced GPUs. The image processing engines (e.g., VCAT, ImageJ, etc.) are installed and a screen-capturing daemon process (*capture server*) runs on each server. The capture server sends the screen rendering results to the remote computing server (Section 4).

Web server. A network server that handles connections between the client web browsers and the transmission control protocol/Internet protocol (TCP/IP) ports of a server machine, and provides access to a content management system (CMS) such as Drupal. Our CMS provides four sets of graphical user interface (GUI)-based contents such as saving (downloading) images to (from) a biomedical image database, searching the database, converting image formats, and managing/processing images. The network server also provides connections between client web browsers, a remote computing server, and CPU/GPU servers. Currently, we employ the Bio-formats library [6] for an image converter and a meta-information reader in our system. Figure 4 illustrates our configuration for the web server.

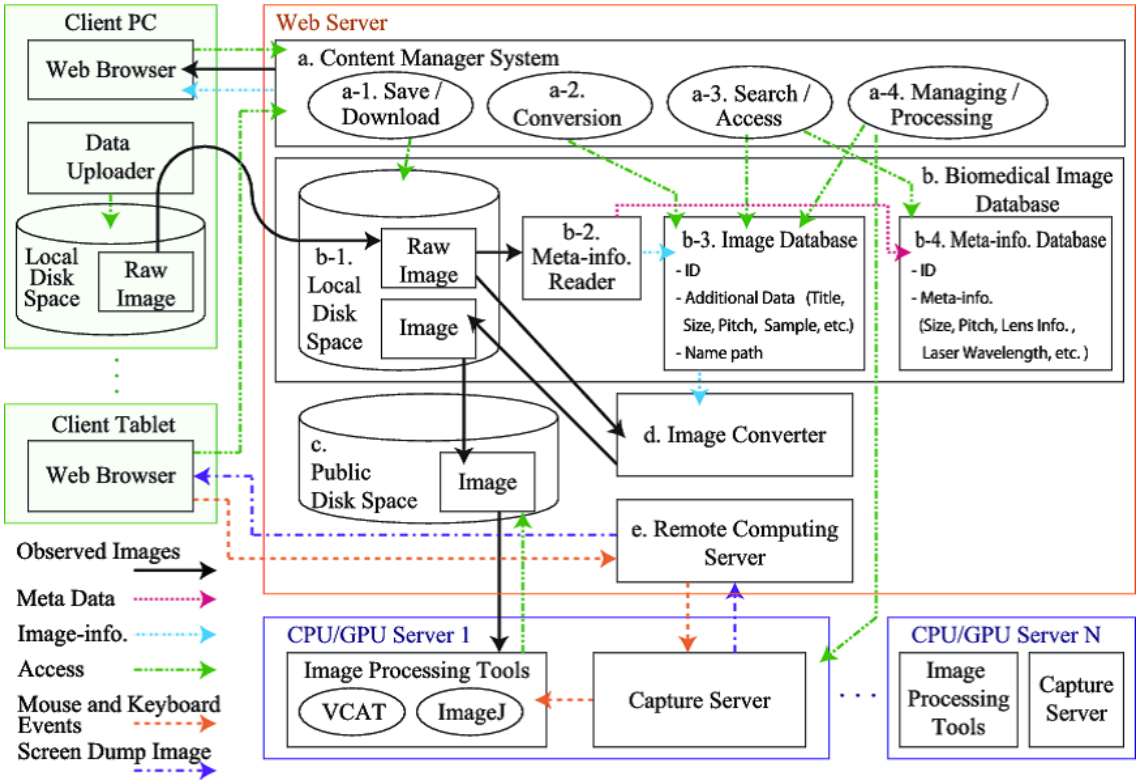


Figure 4: Web server architecture. (a) The CMS provides the GUI-based services (from (a-1) to (a-4)) to users of the biomedical image database. (b) The biomedical image database manages images. (b-1) The local disk space stores the images. (b-2) The image information (e.g., image pathnames) and meta-information are extracted by using a meta-information reader, and are stored in the image (b-3) and meta-information (b-4) databases, respectively. (c) The CPU/GPU servers access to images in the disk space. (d) The image format is unified via TIFF using a image converter. (e) The remote computing server manages user interactions between the CPU/GPU servers and clients.

- *Biomedical Image Database*: The database manages uploaded images with their additional information. Meta-information (e.g., lens type, laser power and frequency, and imaging time) stored in raw image format depends on its format and imaging device, and has wide variations. Thus, our system employs two databases, such as image and meta-information databases, for regular and free-form data, respectively. The image database currently stores absolute image pathnames and additional information corresponding to the image (title, size, pitch, channel number, time-lapse ID, and format type), object (e.g., cell type), imaging device and method (e.g., microscope type), staining technique, dates, and text descriptions (e.g., experiment memo).
- *Remote Computing Server*: A server program that manages user interactions from the web browsers and rendering results from the CPU/GPU servers. See Section 4 for technical details.

Figure 5 presents a simplified flowchart that describes how the image data and the aforementioned components interact in terms of individual hardware. The flowchart for the web server is shown in Fig. 4.

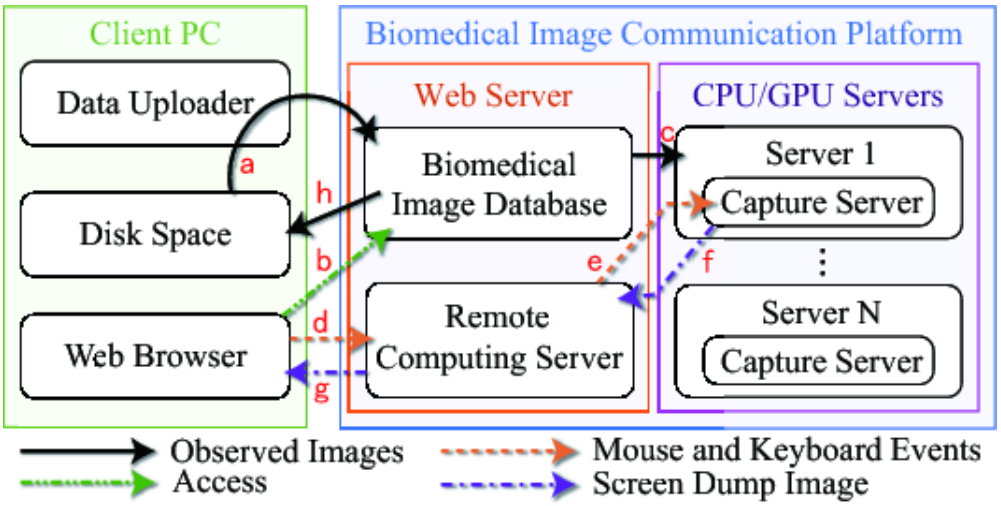


Figure 5: Data flowchart and interaction within our system architecture. (a,h) Image data are uploaded (downloaded) to (from) the biomedical image database via standard FTP software. (b) User logs in to the web server via the web browser to access the biomedical image database and selects a CPU/GPU server. (c) The selected CPU/GPU server downloads the images from the biomedical image database in a background process. (d,e) The remote computing server sends the user's mouse and keyboard events on the web browser to the CPU/GPU server. (f,g) The capture server (daemon process in the CPU/GPU server) automatically sends the desktop screen rendering results to the web browser through the remote computing server.

4 Cloud-based Interactive Technique

Manipulating the image processing engines in the CPU/GPU server *interactively* from the client is crucial to cloud-based 3D image processing. Such manipulation is especially important when the user attempts to adjust the volume rendering parameters (e.g., transfer functions and view information) and apply semi-manual segmentations [1, 8, 28, 29]. If the user is not able to manipulate the engines without delay and jumping of interactions and visualization results, it can be extremely difficult to obtain the desired 3D image processing results (for quality evaluation, immediate visualization is important, even for time-consuming off-line filters). To resolve this issue, we developed a novel cloud-based interactive technique based on the three schemes described in this section to improve the system interactivity. Each scheme is inspired by different conventional techniques developed in image processing and computer graphics, and in contrast to them, our technical contribution includes combining all three schemes to be available simultaneously on standard web browsers.

4.1 Screen Transfer by Detecting Differences

Updating the partial regions of a screen that a user has changed is a popular strategy in video-streaming and cloud-based systems (e.g., [39]). In our system, the capture server detects a change in the screen according to differences, and then sends the screen to the remote computing server if and only if the screen has changed. The capture server always maintains an image of the previous screen. Figure 6 illustrates our strategy.

The sum of squared distances among uniformly sampled pixels between the previous and the current images (as specified in the configuration file) is computed for every frame. If the computed value is greater than a user-specified threshold, the current image is determined to have differences. The difference is measured as

$$\frac{1}{N}\sum_{i=0}^{N-1}(I_{curr}(x_i) - I_{prev}(x_i))^2 > \text{threshold},$$

where N is the number of samples, x_i is a pixel on the screen, $I_{curr}(x_i)$ is the luminance of x_i on the current screen, and $I_{prev}(x_i)$ is the luminance of x_i on the previous screen.

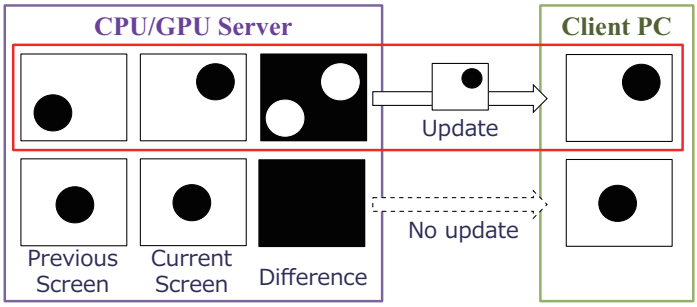


Figure 6: An illustration of screen transfer by the detection of the screen differences. We only update the screen if it has changed.

4.2 Spatially Divided Screen Transfer

Dividing a screen spatially into regions and processing the regions separately is a method that has been used in computer graphics (e.g., multi-GPU rendering for one screen). In our system, the desktop screen of the CPU/GPU server is subdivided, and each region is processed separately to send the corresponding screen part to the remote computing server. The screen is divided into tiles as shown in Fig. 7, and the number of divisions can be changed dynamically via the menu displayed on the browser.

The client web browser periodically queries the remote computing server to determine whether the screen on the CPU/GPU server has changed. If it has changed, the CPU/GPU server sends the indices of the changed tiles (and their corresponding tile images) to the client web browser. The client web browser then updates only the corresponding tile images.

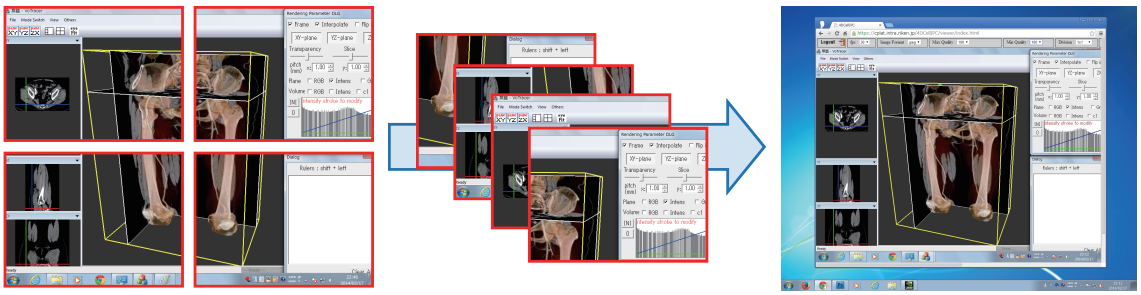


Figure 7: An illustration of screen subdivision with 2×2 image tiles. The screen is divided into smaller image tiles, and we separately update the tile if it has changed.

4.3 Quality Adjusted Screen Transfer

Image and video compression techniques have been well studied [13, 60, 10, 40]; however, the techniques for interactive applications have not been well investigated compared with the field of geometry compression. Existing methods of such techniques are either for simple [62] or predefined [41] regions, specialized with volume rendering [4, 61], or application-dependent [60], although MPEG presumably has interactive 3D content functionality [10].

In our system, the screen compression rate is dynamically changed according to user interaction. A low-quality screen image is employed for high-speed interactions and vice versa. This scheme also includes a buffer (queue) for interaction events so that stroke-based interactions, such as curve drawing, can be managed interactively. Such interactive management is important, for example, for editing transfer functions of volume rendering [34] or sketch-based interfaces of [28, 29].

The image is compressed in the JPEG format. The image quality decreases exponentially between the user-specified minimum and maximum quality values ($\min$ and $\max$) during a mouse operation (e.g., clicking and dragging). Here, we use the image quality (reciprocal of compression rate) as $100q^{\alpha} \in [\min, \max]$ where q is a user-specified constant (we use 0.8 as the default value), and α is a number of user interactions (i.e., mouse operations). All tile images are rendered at the highest quality to guarantee the quality of the still image when the mouse button is released. The graph in Fig. 8 illustrates the change in image quality during a user interaction. Figure 9 demonstrates how this scheme affects the user's view when we rotate a medical CT volume by 90 degrees via our system. As can be seen in the lower images of Fig. 9, the quality of the screen images is decreased by compressions, but such decrease does not disturb user interaction because such change is not time-consuming; for example, the instance described in this paragraph only took an average of 54.7 ms for one frame (18.3 frames per second, or approximately 1 s for 19 frames).

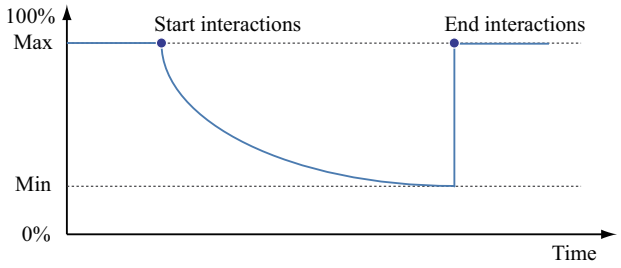


Figure 8: An illustration of quality adjustment during user interaction. The vertical and horizontal axes are the image quality and the interaction time, respectively. We change JPEG compression rate according to mouse dragging operations.

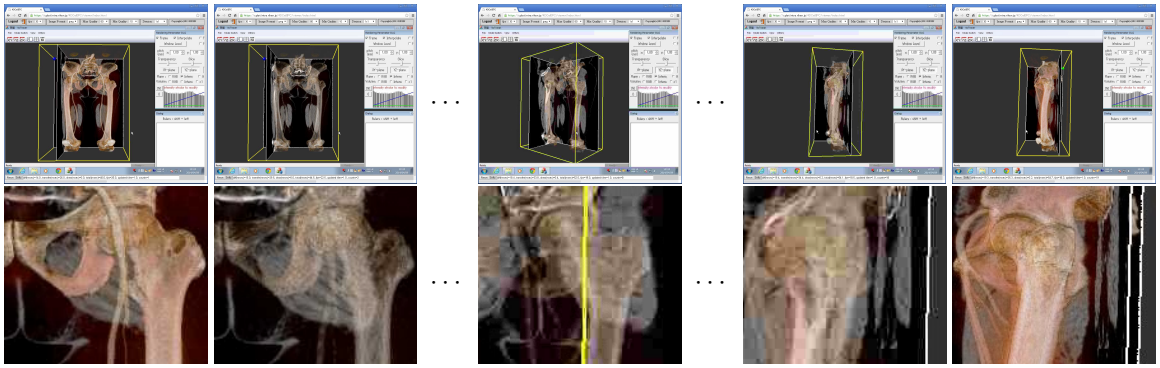


Figure 9: An example of quality adjustment for rotating the medical CT volume via VCAT with our system. The bottom images are magnifications of the corresponding top images. We employed the minimum and maximum image qualities at $[\min, \max] = [10, 80]\%$. The interaction and compression start from frame 2 and end at frame 18; frames 1 and 19 possess 80% quality images.

5 Case Studies on Biomedical Images

In this section, we demonstrate our system by examining three case studies on biomedical images. The case studies typify the use of our system: unsupervised segmentation and filtering, supervised machine learning, and interactive volume visualization and processing.

System implementation. All numerical experiments presented in this paper were executed on two workstations with VCAT. For our system, we employed Drupal 7.20, MySQL 5.5.22, and MongoDB 2.4.6 for CMS, the image database, and the meta-information database, respectively. The biomedical image database and remote computing server were implemented on an Intel Xeon 3.3 GHz CPU workstation with 16 GB of memory and the Apache 2.2.20 web server. The CPU/GPU server was an Intel Xeon 3.6 GHz CPU workstation with 64 GB of memory and NVIDIA GeForce GTX 670. We employed an Intel Core2 Duo 1.83 GHz with 2 GB of memory PC for the client, and ran Google Chrome 34.0 as the web interface. The network was a 100 Mbps LAN.

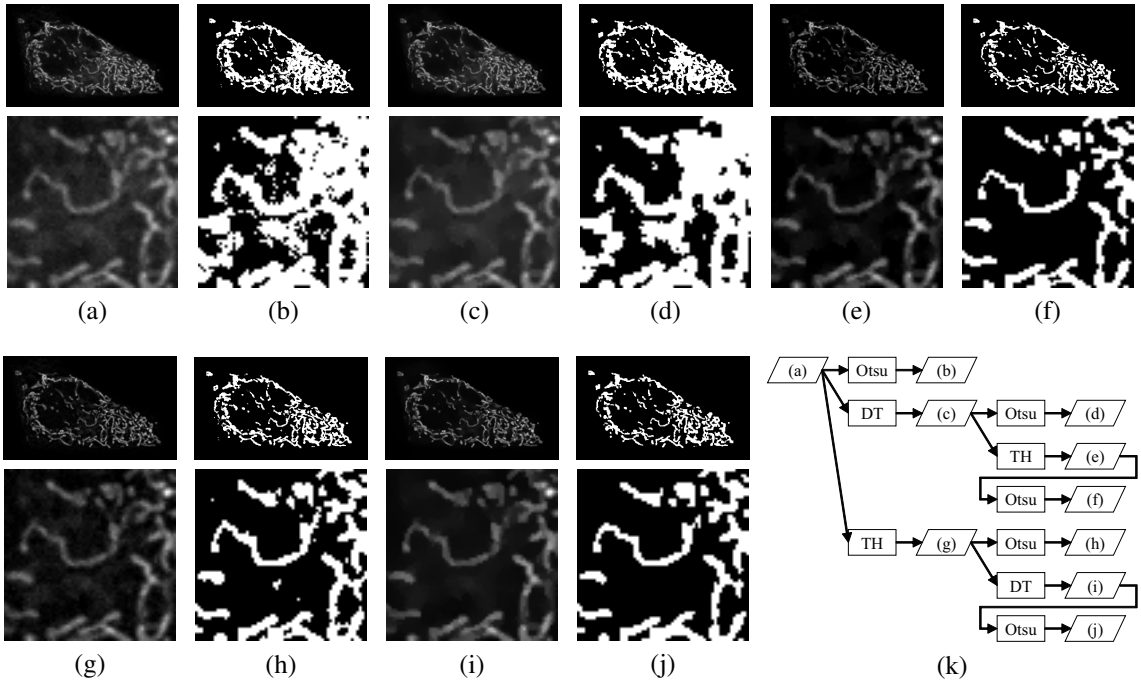


Figure 10: Mitochondria segmentation results. All segmentations were obtained by the Otsu method. (a) Input image. (b) Segmented image of (a). (c) Denoised image of (a) via DT. (d) Segmented image of (c). (e) Top-Hat image of (c). (f) Segmented image of (e). (g) Top-Hat image of (a). (h) Segmented image of (g). (i) Denoised image of (g) via DT. (j) Segmented image of (i). (k) Processing flowchart with the resulting image IDs (a-j).

5.1 Automatic Segmentation of Mitochondria

Intracellular image analysis has become a crucial tool in the field of molecular cell biology because of rapid advances in fluorescence microscopy [50, 19]. A cell contains many internal compartments that consist of lipid bilayer membranes: organelles, cytoskeletons, and vesicles. Thus, separating the regions of interest from the background of cell images is important for quantitative analysis [5].

In this case study, we apply unsupervised image segmentations on a HeLa cell observed with a confocal laser scanning microscope and a fluorescent stain (MitoTracker Deep Red 633 [25]). We extract the mitochondria regions using different procedures, and compare them visually. Figure 10 demonstrates the mitochondria segmentation results.

We employed the following methods. DT [24], which is a state-of-the-art fast edge-preserving smoothing filter, was employed to eliminate unessential noise, with the spatial parameter set to 0.4 and the color range parameter set to 2000 and 8000 for (c-f) and (i) of Fig. 10, respectively, to obtain the best results. The Top-Hat transform that consists of morphological operations [26] was applied to compensate for differences in intensity between the background and mitochondria; a disk with radius 8 was used as the Top-Hat structural element.

Segmentation was performed with the famous Otsu's discriminant thresholding [48], which is the most frequently used unsupervised fully automatic segmentation method. As shown in Fig. 10, filtering with the Otsu method worked well for this case.

5.2 Organellar Classifications via SVM with HLAC

Supervised machine learning is a very powerful tool in computer vision and pattern recognition [7]. Moreover, such a tool has been used in medical image registrations. Thus, its application to biology has garnered considerable attention in biology [58, 14, 50].

We applied organellar classification to three sets of intracellular images obtained by a confocal laser scanning microscopy with fluorescent staining (ER: Endoplasmic Reticulum (GFP[2]), Golgi Apparatus (RFP[3]), and Mitochondria (MitoTracker Deep Red 633)).

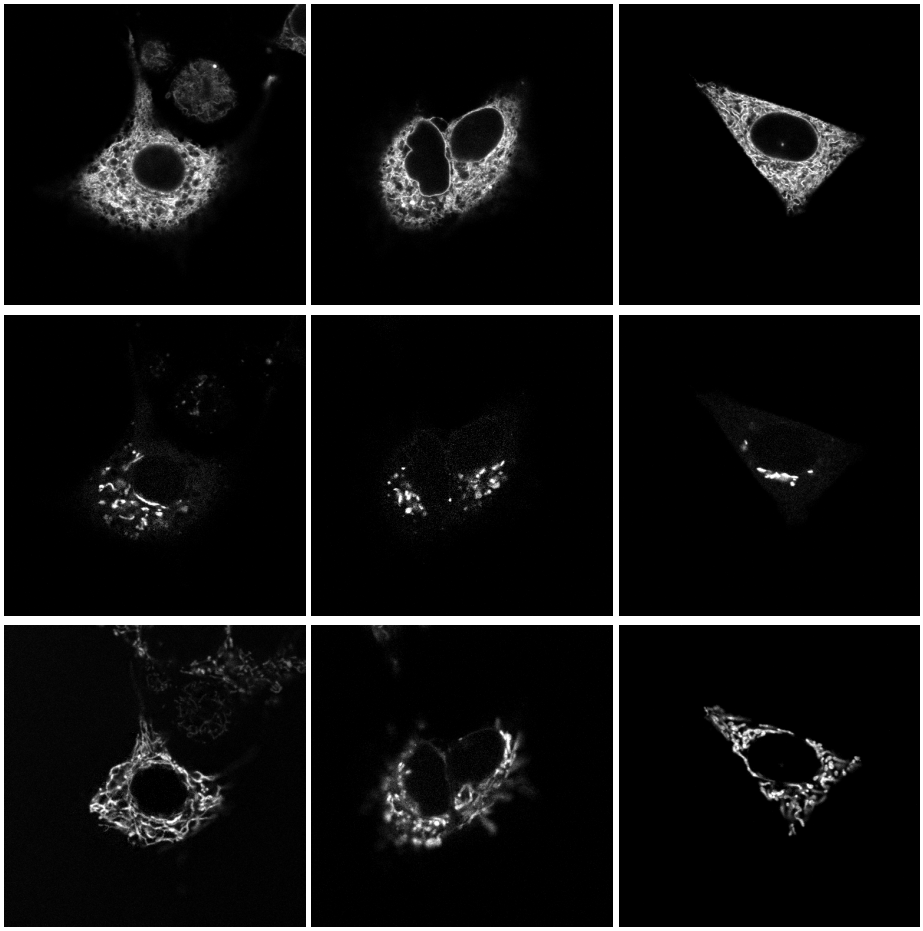
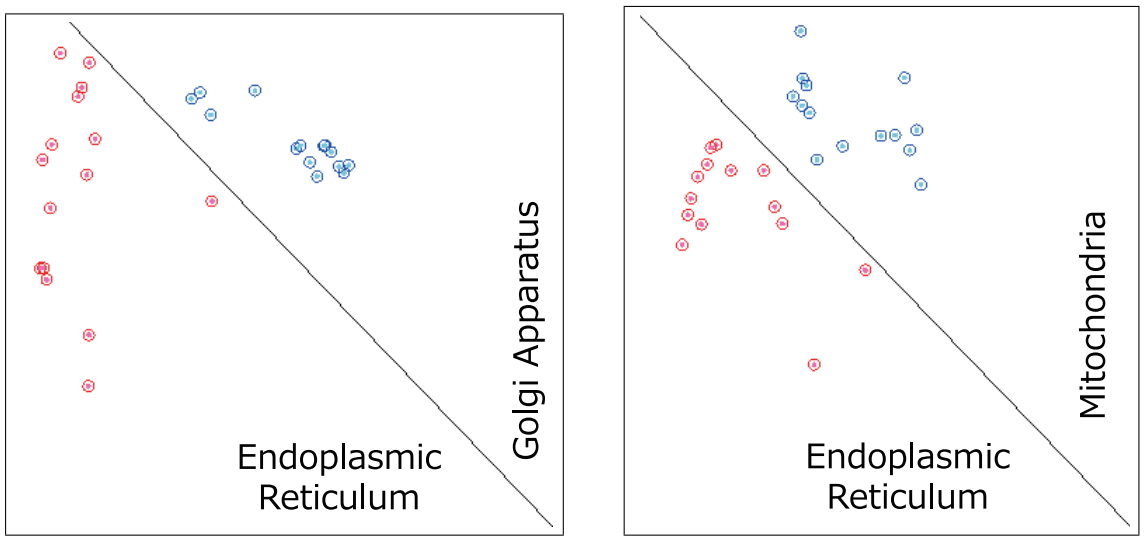


Figure 11: Examples of training images for SVM; top: ER, middle: Golgi Apparatus, and bottom: Mitochondria.

[2]Organelle Lights ER-GFP (O36212), Thermo Fisher Scientific.
[3]Organelle Lights Golgi-RFP (O10098), Thermo Fisher Scientific.

The image features were extracted using HLAC [49], which approximates with higher order local autocorrelation on the image and is proven to be a very good image feature in pattern recognition. A fast implementation of a linear SVM [51] was employed as the classifier, where a hyperplane in feature space was constructed such that the distance from the plane to the nearest sample was maximized allowing for the failure of a sample to reach the correct distance. The C parameter in the linear SVM was $C = 100$, where C exchanges a wide margin with a small number of margin failures [51].

Figure 11 shows representative training images for SVM where 15 images were used for each class. Figure 12 illustrates the learning results of ER vs. Golgi Apparatus (a) and ER vs. Mitochondria (b). Figures 12 and 13 visualize the learned and classified results, respectively, by mapping a high-dimensional feature (HLAC) space (25 dimensions) onto a 2D plane that is orthogonal to the corresponding hyperplane in the HLAC space. The hyperplane is represented by a straight line in the 2D plane (long straight lines in Figs. 12 and 13). The 2D coordinate axes coincide with the normal and tangent directions of the line in the 2D plane. The distance from the 2D sample to the line is equal to the distance from the sample in the HLAC space to the hyperplane. The tangential coordinate is provided by the distance from the projected sample to a fixed sample on the hyperplane.



(a) (b)

Figure 12: Training results using SVM with HLAC for ER vs. Golgi Apparatus (a) and ER vs. Mitochondria (b). The straight line represents the SVM's hyperplane. In this visualization, the training samples are projected onto a 2D plane, where the distance from each sample to the hyperplane is preserved, and the tangential direction of the straight line corresponds to distance from a fixed sample projected onto the hyperplane.

As shown in Fig. 13 and Table 3, ER and Golgi Apparatus are well recognized, whereas the system has some difficulty distinguishing between ER and Mitochondria. In other words, ER and Mitochondria are well correlated with each other. Further investigation is needed in order to understand the correlation in terms of biological evidence (in recently, geometrical and biological correlations between ER and Mitochondria have been reported [27]).

	N	M	Accuracy		N	M	Accuracy
ER	122	120	98.36%	ER	122	85	69.67%
Golgi Apparatus	120	118	98.33%	Mitochondria	119	107	89.92%

Table 3: Classification accuracy of ER with Golgi Apparatus, and Mitochondria, where N and M are the sample size and number of correctly classified images, respectively. The accuracy is measured by M/N as a percentage (%) of the classification.

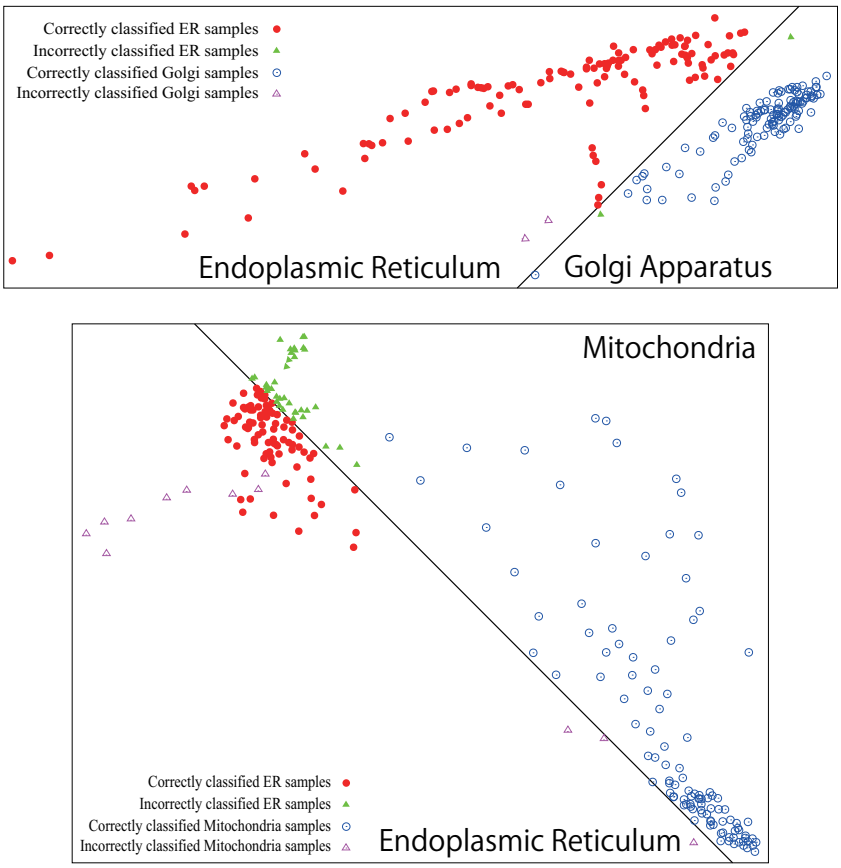


Figure 13: Classification results for ER vs. Golgi Apparatus (top) and ER vs. Mitochondria (bottom). The straight line represents the SVM's hyperplane. In this visualization, the classified samples are projected onto a 2D plane, where the distance from each sample to the hyperplane is preserved, and the tangential direction of the straight line corresponds to the distance from a fixed sample projected onto the hyperplane (similar to the coordinate system used in Fig. 12).

5.3 Interactive Visualization and Segmentation of Medical CT

Data visualization is a key ingredient for image analysis. Our system provides volume rendering [34], orthogonal planar cross-sections, and curved cross-sections [28]. Figure 14 shows the volume rendering and curved cross-section visualization of a CT image. Such high-quality volume visualizations usually require high-end, expensive computers. However, because our system allows the user to share a computer (i.e., the CPU/GPU server), it does not force the user to own a high-end computer.

Fully automatic segmentation is difficult to apply to biomedical images because the images often contain blurred boundaries and noise. Therefore, user input during the segmentation process is crucial. To help the user by providing subjective interpretation during the segmentation process, our system includes a set of interactive segmentation tools, such as seeded region growing [1], graph cuts [8], and contour-based [29] segmentations. With the seeded region growing tool, the user places several seed points on the cross-sections. The system gradually grows the foreground region by considering the local differences in voxel features (Fig. 15a). With the graph-cut segmentation tool, the user places foreground/background seeds. Then, the system computes the foreground voxels by minimizing an energy function. This method solves the energy minimization problem using the minimum cut of a flow network (Figs. 15b,c). With the contour-based segmentation tool, the user specifies a set of contours in the 3D space. The system generates a signed scalar field from the contours and obtains the segmentation boundary surface using the zero-level-set of the signed scalar field (Fig. 16). These interactive tools normally require high-end computing capacity. Our cloud-based system allows users to access these tools with inexpensive devices (e.g., tablet PCs).

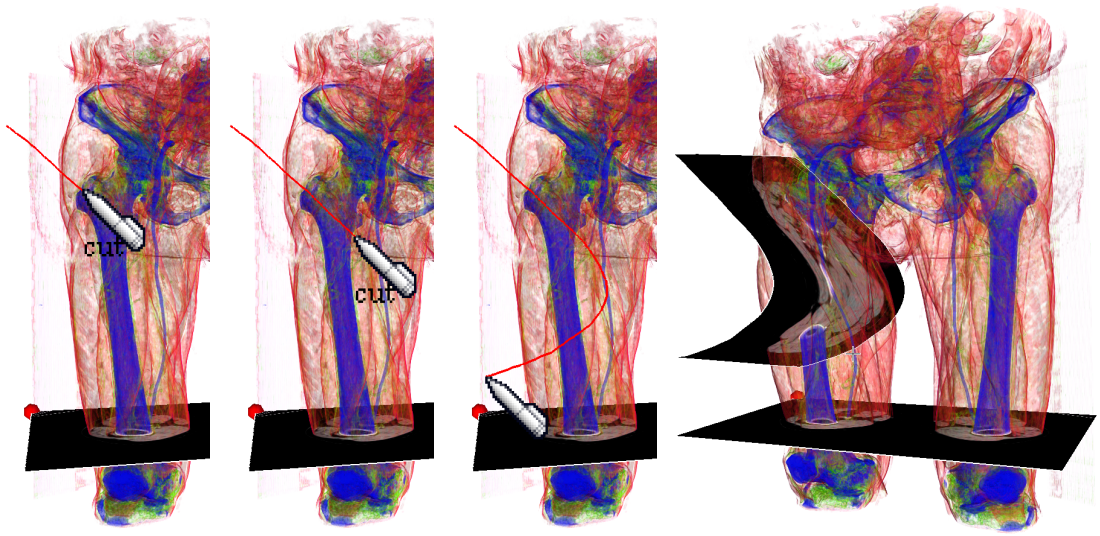
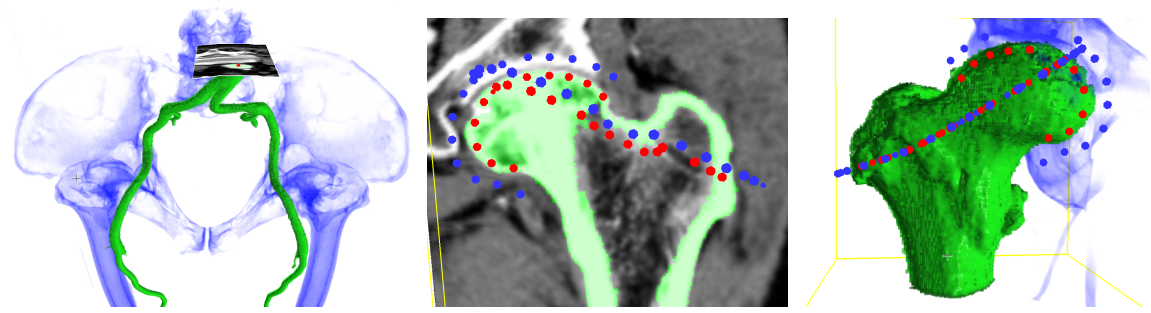


Figure 14: Typical volume visualization using our system.



(a) (b) (c)

Figure 15: The seeded region growing (a) and graph-cut segmentation (b,c) tools. (a) The red mark is a seed. A segmented region is highlighted in green. (b,c) The user specifies the foreground (red) and background (blue) seeds. The resulting segmented region is highlighted in green.

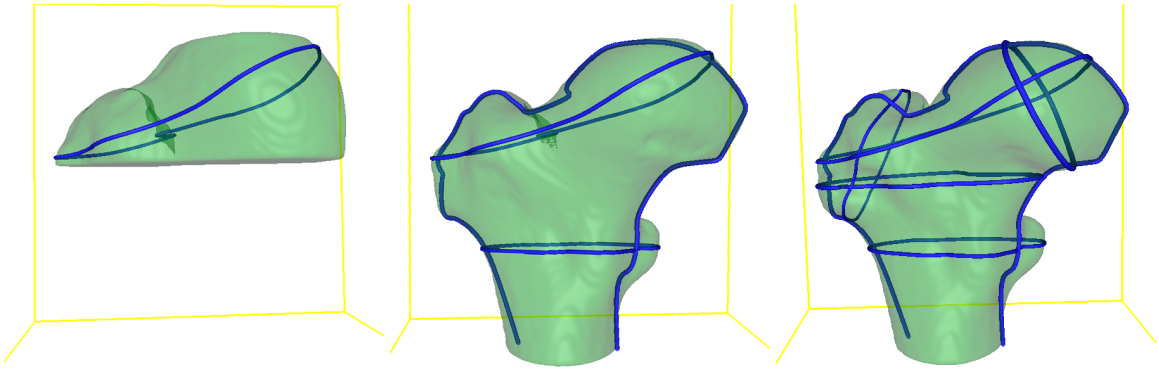


Figure 16: The contour-based segmentation tool. The user specifies contours (blue), and the system computes a segmentation boundary (green surface) that interpolates both contours and image edges.

6 Performance Evaluation

We examined the performance of our cloud-based interactive technique described in Section 4 using the implementation and case studies described in Section 5. Let D, S, and Q be the abbreviations for our schemes proposed in Sections 4.1, 4.2, and 4.3, respectively; see Table 4. Consider the three case studies (A, B, and C) presented in Sections 5.1, 5.2, and 5.3, respectively; see Table 5. Figures 17 and 18 illustrate the average response times for one frame and the volume of downloaded data (KB/s) for experiments in the case studies (A, B, and C) and combinations of the proposed schemes (D, S, and Q). The response times were measured through a web browser. The display resolution was 1024×768. For scheme S, a screen subdivision number of 6×6 was used. The maximum and minimum values for scheme Q were 100% and 20%, respectively.

Symbol	Scheme Name	Section	On	Off
D	Screen Transfer by Detecting Differences	4.1	D:On	D:Off
S	Spatially Divided Screen Transfer	4.2	S:On	S:Off
Q	Quality Adjusted Screen Transfer	4.3	Q:On	Q:Off

Table 4: Abbreviations for our schemes employed in this section and in Figs. 17 and 18. Here, On and Off correspond to switching on and off the functions of our schemes, respectively.

Symbol	Study Name	Section	Case Name
A	Automatic Segmentation of Mitochondria	5.1	Case A
B	Organellar Classifications via SVM with HLAC	5.2	Case B
C	Interactive Visualization and Segmentation of Medical CT	5.3	Case C

Table 5: Abbreviations for our case studies employed in this section and in Figs. 17, 18, and 19. The experiments A, B, and C were performed for each measurement carefully in order to match every results as closely as possible (because the case studies include user interactions).

Applying scheme Q always provides a better response time according to the comparison between Q:Off and Q:On shown in Fig. 17. In other words, image compression during user interaction works well in all cases and combinations of schemes. Additionally, Fig. 18 shows that scheme Q greatly reduces the average volume of the downloaded data for all cases and schemes.

On the other hand, the response time is not improved using scheme S (S:On). In particular, the response time is not good when transferring screens without detecting differences (D:Off and S:On). This is because of the overhead of dividing an image into small tiles. The screen transfer using both D and S (D:On and S:On) schemes provides a reasonable performance compared with the image response time obtained by not using scheme S (S:Off with D:On/Off). Therefore, scheme S should be always used in combination with scheme D.

Subdivision number analysis. We further investigated a significant number of screen subdivisions. Figure 19 presents the average image response time when we vary the subdivision number of scheme S (D:On, S:On, and Q:On). Figure 6 details the results of the response time, the frames per second (FPS), and the updated screen area ratio (USAR) calculated by averaging the three case studies. Here, a smaller response time (larger FPS, smaller USAR) is better.

As can be seen by the USAR shown in Fig. 6, subdividing the screen decreases the screen area that is updated. On the other hand, the cases that have more than 6×6 tiles do not gain actual speed (see the FPS shown in Fig. 6) because of the overhead of processing too many small area tiles. This numerical experiment suggests that the use of 2×2 tiles is the best parameter for all case studies, see Fig. 6. The case of 2×2 tiles demonstrates the fastest response time, and approximately 1.3 FPS better than the no subdivision case (1×1). The case of 4×4 tiles is similar to the no subdivision case, according to its FPS.

Overall, the case of all three schemes employed simultaneously (D:On, S:On, and Q:On) with 2×2 subdivision is the best performance in our case studies. This performance evaluation indicates that our technique is useful and effective for the interactivity of a cloud-based system that provides a rich variety of biomedical image-processing applications.

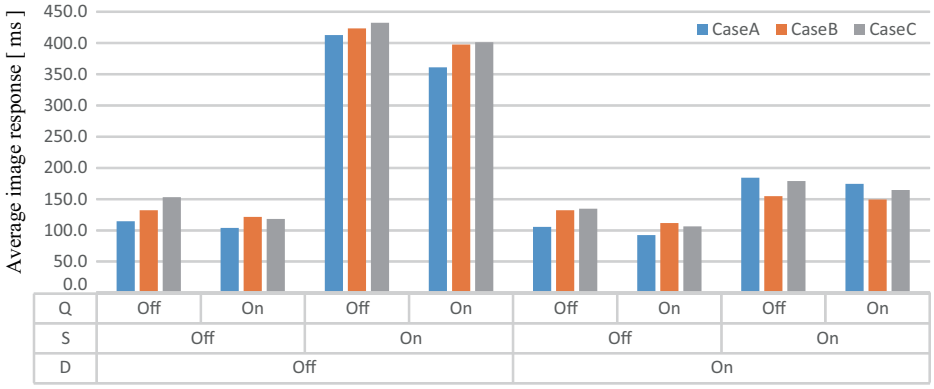


Figure 17: The average image response time (ms) for one frame with switching on/off of our three schemes. The symbols Q, S, and D are given in Table 4.

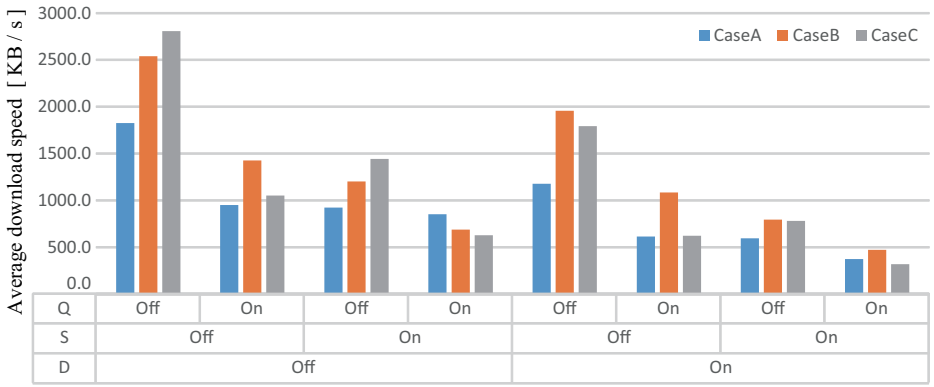


Figure 18: The average rate of data downloaded (KB/s) with switching on/off of our three schemes. The symbols Q, S, and D are given in Table 4.

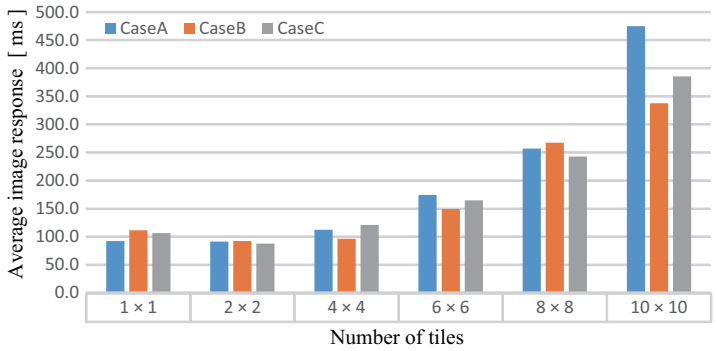


Figure 19: Average image response time (ms) for one frame with different numbers of tiles.

	1×1	2×2	4×4	6×6	8×8	10×10
Response Time	103.4	90.7	109.9	162.9	255.7	399.3
Frames Per Second: FPS	9.7	11.0	9.2	6.2	3.9	2.6
Updated Screen Area Ratio: USAR	90.0%	57.5%	38.3%	33.7%	34.1%	33.4%

Table 6: Average image response time (ms), FPS, and number of updated tiles per frame for different division numbers of a screen.

7 Preliminary User Test

Because our system is targeted to scientists and engineers who might not have sufficient knowledge about image processing, it is better to evaluate our system in terms of the feasibility and usability for non-computer-scientists. Here, we report the results of a simple user test, although extensive user studies remain for future work.

We performed the user tests described in this section with eight non-expert participants that consisted of three biologists, two medical scientists, and three mechanical and precision engineers who had never used our system (or the conventional system employed in this test) before the test, with the exception of one of the participants, see Table 7. Here, two of the biologists, one of the medical scientists, and one of the engineers had limited experience with a stand-alone VCAT; the remaining participants had no experience. In addition, one of the participants is one of authors of this paper and contributed to this research partially, but never had the opportunity of using our system before the test.

Symbol	Character	Number	Task	Number $\times$ Task	VCAT Experience
B	Biologist	3	2	6	2
M	Medical Scientist	2	1	2	1
E	Engineer	3	1	3	1

Table 7: Abbreviations for the eight participants. Because the three biologists performed the tasks twice, the total number of tasks is equivalent to 11.

Test setting. We asked the participants to perform a segmentation task similar to the case study A described in Section 5.1 on two systems: our system with Google Chrome and VCAT, and a conventional system that consisted of VCAT, OMERO [2] 4.4.4, and Chrome Remote Desktop (CRD) 35.0. In our system, we employed all schemes simultaneously (D:On, S:On, and Q:On) with 2×2 tiles in scheme S and the minimum and maximum image qualities of scheme Q as $[\min, \max] = [20, 80]\%$.

7.1 Segmentation Task

First, we explained to the participants how to use both systems and asked the participants to segment ten mitochondria images by showing the images (a) and (h) of Fig. 10 as the input and the preferred result, respectively. We also provided a simple instruction note for how to proceed with this test. The task was provided with the following instructions:

Our System : Login to our system via a web browser. For the ten images, select an image and a CPU/GPU server, and perform the following VCAT procedure.

Conventional System : Access a server via CRD. Run OMERO via a shortcut icon. For the ten images, download an image from OMERO and perform the following VCAT procedure.

Common Task on VCAT : Run VCAT via a shortcut icon and load the image. Apply DT and Top-Hat filtering according to case study A. (We allowed the participants to freely choose the parameters and the order of the DT and Top-Hat, but provided these tips: $\alpha = 0.4$, $\sigma \in [1000, 10000]$ for DT, and the radius of Top-Hat lives $[1, 20]$.) Apply the Otsu method to the filtered image and save the resulting image.

Test conditions. Obviously, the order in which the systems are tested affects the results, especially with regard to the time required to finish the task, because the VCAT portions are common. Thus, we asked four of the participants (two biologists, one medical scientist, and one engineer) to attempt to use the conventional system first, and then our system second. We asked the other four participants to attempt the reverse order. Moreover, we asked the three biologists to attempt a reverse order with different image sets (ten ER images used in the case study B described in Section 5.2).

7.2 Test Results

We measured the total time required to finish the task and the mean squared error (MSE) between the preferred and task results. The preferred segmented results (ten images for both mitochondria and ER) were produced by an experienced user with knowledge of both our system and the image processing tools used in this test. Table 8 provides the total and average processing times for each group of participants. As expected, the biologists required more time than the medical scientists and the engineers. It is because their biological knowledge tend to spend more time to segment the organelles accurately. The total and average times required for our system is somewhat longer than that required for the conventional system (4 min in average for ten image segmentations). It is probably because of good affordance of CRD. On the other hand, the processing time the biologists required to use our system is slightly faster than that required to use the conventional system.

Table 9 shows the total and average MSE for each group of participants. As expected, the engineers and biologists scored the best and worst, respectively. The total and average MSE for our system is clearly better than for the conventional system.

	Total				Average: Total/(Number × Task)			
Character	(B,M,E)	B	M	E	(B,M,E)	B	M	E
Number × Task	11	6	2	3	11	6	2	3
Conventional System	6.9	5.5	0.56	0.84	0.63	0.92	0.28	0.28
Our System	7.33	5.21	0.75	1.37	0.67	0.87	0.37	0.46

Table 8: Total (left) and average (right) processing times (hours) and (hours/persons), respectively.

	Total				Average: Total/(Number × Task)			
Character	(B,M,E)	B	M	E	(B,M,E)	B	M	E
Number × Task	11	6	2	3	11	6	2	3
Conventional System	1.66	1.198	0.194	0.255	0.15	0.2	0.097	0.085
Our System	1.271	0.879	0.198	0.195	0.116	0.146	0.099	0.065

Table 9: Total (left) and average (right) MSE where the segmented image consists of zero or one pixel values.

	Total 11					Total 8				
	Q1	*Q2*	*Q3*	*Q4*	*Q5*	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
Conventional System	7	7	1	1	8	7	4	1	1	6
Our System	8	7	10	6	8	7	5	8	4	6

Table 10: Total numbers of affirmative (Yes) answers to the questions. The left table includes the second segmentation task of the three biologists, whereas the right table consists of the first segmentation task of all eight participants.

We also asked five questions (yes or no) to the participants. Table 10 summarizes the number of affirmative (Yes) answers given to the questions. Unfortunately, the questions regarding latency (*Q3* and *Q4*) provide a better result for the conventional system. On the other hand, questions (*Q1*, *Q2*, and *Q5*) provide similar results for both the conventional and our systems. Considering the timing and segmentation quality discussed in the previous paragraphs, the participants' feeling of latency affects somewhat on the processing time, and this might produce opposite results in terms of quality because users might become more prudent when processing the images. Nevertheless, extensive user studies that include perceptual evaluation are required in order to obtain a conclusive analysis.

The five questions are as follows:

- *Q1*: Are you satisfied with your segmentation results ?
- *Q2*: Was it easy to access the images ?
- *Q3*: Did you feel latency for mouse or keyboard interactions ?
- *Q4*: Did you feel latency for displaying a screen ?

- *Q5*: Do you want to use the system in future ?

8 Conclusion

We proposed a novel cloud-based communication system for biomedical images. The system is based on a set of computer servers that manage collaborative users, their images and software, and CPU/GPU resources effectively and seamlessly. Because our system can control data management, visualization, and processing services through a standard web browser on inexpensive hardware, users can collaborate and share limited software and hardware resources for biomedical image processing and analysis. The paper's technical contribution also includes the novel cloud-based interactive technique. We believe that our system can significantly contribute to quantitative analysis in modern biology.

Future Work We employed an intermediate upload server between the biomedical image database and the client FTP software in our current implementation. Future work will simplify this intermediate server and include evaluations via end-user studies.

Acknowledgements We would like to thank the participants of preliminary user test (described in Section 7) for their efforts, Yutaka Ohtake for his meshing VCAT plug-in program, and the anonymous reviewers of this paper for their helpful comments and suggestions. The human CT volume (employed in Figs. 1, 7, 9, 14, 15, and 16) is courtesy of Kazuaki Fukasaku (Himonya hospital). This work was supported in part by Strategic Programs for R&D (Presidents Discretionary Fund) of RIKEN and Grants-in-Aid for Scientific Research of Japan (20113007, 24700182, and 25700011).

References

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[2] C. Allan, J.-M. Burel, J. Moore, C. Blackburn, M. Linkert, S. Loynton, D. MacDonald, W. Moore, C. Neves, A. Patterson, M. Porter, A. Tarkowska, B. Loranger, J. Avondo, I. Lagerstedt, L. Lianas, S. Leo, K. Hands, R. Hay, A. Patwardhan, C. Best, G. Kleywegt, G. Zanetti, and J. R Swedlow. OMERO: flexible, model-driven data management for experimental biology. *Nature Methods*, 9(3):245–253, 2012.

[3] ATI. ATI crossfire technology white paper. Technical report, ATI Technologies, 2005.

[4] C. Bajaj, I. Ihm, and S. Park. 3D RGB image compression for interactive applications. *ACM Trans. on Graphics*, 20(1):10–38, 2001.

[5] H. Bhaskar and S. Singh. Live cell imaging: a computational perspective. *J. of Real-Time Image Processing*, 1:195–212, 2007.

[6] Univ. Wisconsin-Madison Biophotonics Research Lab. Bio-formats library, 2013.

[7] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[9] A. Buades and B. Coll. A non-local algorithm for image denoising. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 60–65. IEEE-CS, 2005.

[10] A. Buchowicz. Video coding and transmission standards for 3D television - a survey. *Opto-Electronics Review*, 21(1):39–51, 2013.

[11] A. Carpenter, T. Jones, M. R Lamprecht, C. Clarke, I. Kang, O. Friman, D. Guertin, J. Chang, R. Lindquist, J. Moffat, P. Golland, and D. Sabatini. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):R100, 2006.

[12] A. Cedilnik, B. Geveci, K. Moreland, J. Ahrens, and J. Favre. Remote large data visualization in the ParaView framework. In *Proc. of EG Conf. on Parallel Graphics and Visualization*, pages 163–170. EG Association, 2006.

[13] R. J. Clarke. Image and video compression: A survey. *Int. J. of Imaging Systems and Technology*, 10(1):20–32, 1999.

[14] L. Coelho, E. Glory-Afshar, J. Kangas, S. Quinn, A. Shariff, and R. Murphy. Principles of bioimage informatics: Focus on machine learning of cell patterns. In *Proc. of Workshop of the BSIG, Int. Conf. on Linking Literature, Information, and Knowledge for Biology*, pages 8–18. Springer-Verlag, 2010.

[15] D. Comaniciu and P. Meer. Mean Shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[16] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 886–893. IEEE-CS, 2005.

[17] F. de Chaumont, S. Dallongeville, N. Chenouard, N. Hervé, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. Montagner, T. Lagache, A. Dufour, and J. Olivo-Marin. Icy: an open bioimage informatics platform for extended reproducible research. *Nature Methods*, 9(7):690–696, 2012.

[18] R. M. Deserno. *Biomedical Image Processing*. Springer, 2011.

[19] Editorial. The quest for quantitative microscopy. *Nature Methods*, 9(7):627, 2012.

[20] K. Eliceiri, M. Berthold, I. Goldberg, L. Ibáñez, B. Manjunath, M. Martone, R. Murphy, H. Peng, A. Plant, B. Roysam, N. Stuurman, J. Swedlow, P. Tomancak, and A. Carpenter. Biological imaging software tools. *Nature Methods*, 9(7):697–710, 2012.

[21] K. Engel, T. Ertl, P. Hastreiter, B. Tomandl, and K. Eberhardt. Combining local and remote visualization techniques for interactive volume rendering in medical applications. In *Proc. of the IEEE Conf. on Visualization*, pages 449–452. IEEE-CS, 2000.

[22] K. Engel, O. Sommer, and T. Ertl. A framework for interactive hardware accelerated remote 3D-visualization. In *Proc. of EG/IEEE TCVG Symp. on Visualization*, pages 167–177,291, 2000.

[23] M. Nishimura et al. RIKEN: Image Processing Research Team. VCAT: Volume computer aided testing software Ver. 5, 2011-2014. http://logistics.riken.jp/vcat/en.

[24] E. Gastal and M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. on Graphics*, 30(4):69:1–69:12, 2011. Proc. of ACM SIGGRAPH.

[25] K. Gilmore and M. Wilson. The use of chloromethyl-X-rosamine (Mitotracker red) to measure loss of mitochondrial membrane potential in apoptotic cells is incompatible with cell fixation. *Cytometry*, 36:355–358, 1999.

[26] R. Gonzalez and R. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2007.

[27] M. Hamasaki, N. Furuta, A. Matsuda, A. Nezu, A. Yamamoto, N. Fujita, H. Oomori, T. Noda, T. Haraguchi, Y. Hiraoka, A. Amano, and T. Yoshimori. Autophagosomes form at ER-mitochondria contact sites. *Nature*, 495(7441):389–393, 2013.

[28] T. Ijiri and H. Yokota. Contour-based interface for refining volume segmentation. *Computer Graphics Forum*, 29(7):2153–2160, 2010.

[29] T. Ijiri, S. Yoshizawa, Y. Sato, M. Ito, and H. Yokota. Bilateral Hermite radial basis functions for contour-based volume segmentation. *Computer Graphics Forum*, 32(2):123–132, 2013. Proc. of EUROGRAPHICS'13.

[30] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[31] T. Jones, I. Kang, D. Wheeler, R. Lindquist, A. Papallo, D. Sabatini, P. Golland, and A. Carpenter. CellProfiler Analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics*, 9(1):482, 2008.

[32] J. T. Kajiya. The rendering equation. In *Computer Graphics*, pages 143–150, 1986.

[33] P. Kankaanpää, L. Paavolainen, S. Tiitta, M. Karjalainen, J. Päivärinne, J. Nieminen, V. Marjomäki, J. Heino, and D. White. BioImageXD: an open, general-purpose and high-throughput image-processing platform. *Nature Methods*, 9(7):683–689, 2012.

[34] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[35] K. Kvilekval, D. Fedorov, B. Obara, A. Singh, and B. Manjunath. Bisque: a platform for bioimage analysis and management. *Bioinformatics*, 26(4):544–552, 2010.

[36] T. Lin and S. Wang. Cloudlet-screen computing: a multi-core-based, cloud-computing-oriented, traditional-computing-compatible parallel computing paradigm for the masses. In *Proc. of IEEE Int. Conf. on Multimedia and Expo*, pages 1801–1804. IEEE Press, 2009.

[37] T. Lin, K. Zhou, and S. Wang. Cloudlet-screen computing: A client-server architecture with top graphics performance. *Int. J. Ad Hoc Ubiquitous Comput.*, 13(2):96–108, 2013.

[38] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[39] Y. Lu, S. Li, and H. Shen. Virtualized screen: A third element for cloud-mobile convergence. *IEEE MultiMedia*, 18(2):4–11, 2011.

[40] M. Sharif M. Rehman and M. Raza. Image compression: A survey. *J. of Applied Sciences, Engineering and Technology*, 7:656–672, 2014.

[41] R. Mantiuk, K. Myszkowski, and S. Pattanaik. Attention guided MPEG compression for computer animations. In *Proc. of Spring Conf. on Computer Graphics*, pages 239–244. ACM, 2003.

[42] Microsoft. Remote Desktop Protocol, 2009.

[43] K. Moreland, L. Avila, and L. A. Fisk. Parallel unstructured volume rendering in ParaView. In *Proc. of IS&T SPIE Visualization and Data Analysis*, volume 64950F, pages 1–12, 2007.

[44] M. Morita, T. Tawara, M. Nishimura, S. Yoshizawa, B. Chou, I. Kuroki, T. Ijiri, Y. Tsujimura, R. Himeno, and H. Yokota. Biomedical image communication platform. In *Proc. of Int. Symp. on Computing and Networking*, pages 281–287. IEEE, 2013. Int. Workshop on BioImage Recognition.

[45] R. F. Murphy. Location proteomics: a systems approach to subcellular location. *Biochemical Society Transactions*, 33(3):535–538, 2005.

[46] K. Niski and J. D. Cohen. Tile-based level of detail for the parallel age. *IEEE Trans. on Visualization and Computer Graphics*, 13(6):1352–1359, 2007.

[47] Y. Ohtake and H. Kawaharada. Tetrahedral mesh generation of multi-material solids from 3D segmented images. In *Proc. of JSPE (Autumn Meeting)*, pages 323–324, 2007. in Japanese.

[48] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11:23–27, 1975.

[49] N. Otsu and T. Kurita. A new scheme for practical flexible and intelligent vision systems. In *Proc. of IAPR Workshop on Computer Vision*, pages 431–435. IAPR, 1988.

[50] H. Peng, A. Bateman, A. Valencia, and J. Wren. Bioimage informatics: a new category in bioinformatics. *Bioinformatics*, 28(8):1057, 2012.

[51] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. BSchölkopf et al., editor, *Advances in kernel methods*, pages 185–208. MIT Press, 1999.

[52] F. Porikli. Constant time O(1) bilateral filtering. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8. IEEE-CS, 2008.

[53] W. S. Rasband. ImageJ: Image processing and analysis in Java, 1997-2013.

[54] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[55] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. of ACM SIGGRAPH*, pages 343–352. ACM, 2000.

[56] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9(7):676–682, 2012.

[57] C. Schneider, W. Rasband, and K. Eliceiri. NIH image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675, 2012.

[58] L. Shamir, N. Orlov, D. Eckley, T. Macura, and J. Johnston. Wndchrm- an open source utility for biological image analysis. *Source Code for Biology and Medicine*, 3:1–13, 2008.

[59] C.-P. Shen, C.-H. Liu, F.-S. Lin, H. Lin, C.-Y. Huang, C.-Y. Kao, F. Lai, and J.-W. Lin. A multiclass classification tool using cloud computing architecture. In *Proc. of IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining*, pages 765–770. IEEE-CS, 2012.

[60] A. Smolic, K. Mueller, N. Stefanoski, J. Ostermann, A. Gotchev, G. B. Akar, G. Triantafyllidis, and A. Koz. Coding algorithms for 3DTV - a survey. *IEEE Trans. on Circuits and Systems for Video Technology*, 17:1606–1621, 2007.

[61] B.-S. Sohn, C. Bajaj, and V. Siddavanahalli. Volumetric video compression for interactive playback. *Comput. Vis. Image Underst.*, 96(3):435–452, 2004.

[62] P. G. Tahoces, J. R. Varela, M. J. Lado, and M. Souto. Image compression: Maxshift ROI encoding options in JPEG2000. *Comput. Vis. Image Underst.*, 109(2):139–145, 2008.

[63] D. Thompson, A. Khassapov, Y. Nesterets, T. Gureyev, and J. Taylor. X-ray imaging software tools for HPC clusters and the cloud. In *Proc. of IEEE Int. Conf. on e-Science*, pages 1–7. IEEE-CS, 2012.

[64] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. of Int. Conf. on Computer Vision*, pages 839–846. IEEE-CS, 1998.

[65] D. Vazhenin. Cloud-based web-service for health 2.0. In *Proc. of Joint Int. Conf. on Human-Centered Computer Environments*, pages 240–243. ACM, 2012.

[66] T.-Y. Wu, C.-Y. Chen, L.-S. Kuo, W.-T. Lee, and H.-C. Chao. Cloud-based image processing system with priority-based data distribution mechanism. *Computer Communications*, 35(15):1809–1818, 2012.

[67] S. Yoshizawa, A. Belyaev, and H. Yokota. Fast Gauss bilateral filtering. *Computer Graphics Forum*, 29(1):60–74, 2010.

[68] P. Young. SLI best practices. Technical report, NVIDIA Corporation, 2005.