Report from 2015 OHBM Hackathon (HI)

NiftyView: A Zero-Footprint Web Application for viewing DICOM and NIfTI files

Project URL: http://www2.hawaii.edu/%7Eweiran/NiftyView.html

Weiran Deng

1 Introduction

The purpose of developing yet another web-based image viewer, NiftyView, is to use WebGL to take advantage of the parallel computing power in Graphics Processing Units (GPU) hardware for the acceleration of rendering and processing of medical images in web applications. Although several web-based medical image viewers such as Papaya, BrainBrowser and Slice:Drop are currently available, the slow performance of the web-based applications is still one of the major limitations of web-based image viewers. NiftyView is a free web application developed in JavaScript. It has zero footprint; only a web browser and an Internet connection are needed to run NiftyView. It's advantageous over conventional desktop applications in that NiftyView doesn't require installation and constant updates. The current version supports NIfTI and DICOM format. As a minimal image viewer, it's a convenient tool for users who need a quick and easy tool for viewing medical images. Currently, the beta version of NiftyView is freely available at http://www2.hawaii.edu/~weiran/NiftyView.html .

2 Approach

NiftyView is developed in JavaScript with jQuery for HTML document manipulation and event handling, jQueryUI for user interface, and DicomParser for parsing DICOM files. It's compatible with popular web browsers including Internet Explorer, Safari, Firefox, and Opera. Either DICOM or NIfTI files can be loaded by dragging files into the browser window. Loaded images can be displayed in single-slice mode or tiled mode. After loading, images are automatically arranged according to the scan IDs for DICOM files and the file name for NIfTI files, respectively. Current functions include image zooming and adjustment of image brightness and contrast. The number of image columns can be adjusted in the tiled mode to maximize the use of the display space. The contrast and brightness of images can be adjusted by clicking and holding the right mouse button or using a double slider widget in the horizontal tool bar at the top of the window. For proof-of-concept, functions such as pixel windowing and scaling are programmed using WebGL by translating the arithmetic operations in image processing to 3D graphics primitives using WebGL's programmable shaders. The pixel values of an image is loaded into a frame buffer. A vertex shader is programmed to define vertices corresponding to the coorddinates of the image, and a fragment shader is programmed to perfrom arithmetic operations, which are performed in parallel to a massive number of image pixels.

Correspondence: weirandeng@gmail.com
University of Hawaii John A. Burns School of Medicine, Honolulu, Hawaii, USA, Honolulu, Hawaii, USA
Full list of author information is available at the end of the article

3 Results

See Figures 1 and 2.

4 Discussion

One of the major limitations of current web-based image viewers is the slow performance compared to their desktop counterparts.

There are collective efforts in industry to develop new technologies such as WebAssmebly and WebGL to narrow this performance gap. The highly parallel nature in processing image pixels independently allows the use of WebGL to achieve a signficant speedup, as shown in this abstract. Currently, there are several similar existing web applications such as Papaya,

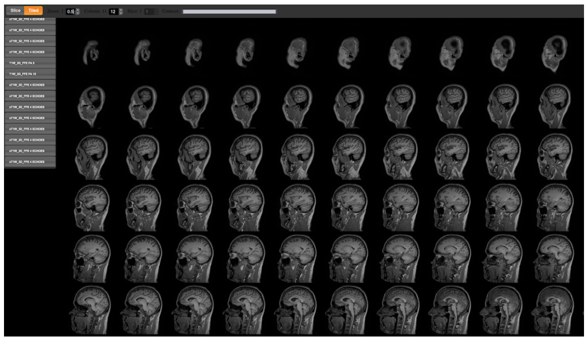

Figure 1 A few sagittal MRI images displayed in titled mode after loaidng approximately 1,500 DICOM files from 11 MRI scans. It took approximately ten seconds to load all the DICOM files into NiftyView. The images are organized into different vertical tabs by the sequence names stored in the DICOM files.

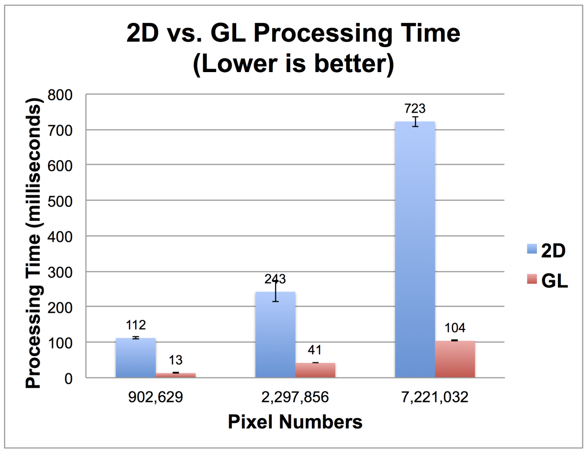

Figure 2 Image of WebGL vs. Canvas Comparison. Shows a comparison of processing time as a function of the number of image pixels in JavaScript(blue) and WebGL (red). WebGL shows a factor of six to eight accelerations.

BrainViewer, and slicedrop.com, which are more mature and offer varieties of features. However, the main goal of the continuing effor in the development of NiftyView is to achieve a high performance for image processing using GPU via WebGL. NiftyView has a minimal boilerplate and can handle a large number of files with relative ease. Future work will be focused on developing a WebGL-accelerated version, adding more image processing features, and adding support of accessing files stored in HIPAA (Health Insurance Portability and Accountability Act) compliant cloud storage servies such as Box and Amazon S3. The stable version of NiftyView will be released under a General Public License that allows end users to freely run, modify, and share the program.

5 Conclusion NiftyView is a free and convenient web application for

quick and easy viewing of NIfTI and DICOM medical images. We have shown that a factor of six to eight acceleration can be achieved using WebGL for image processing.

Availability of Supporting Data
More information about this project can be found at: http://www2.hawaii.edu/%7Eweiran/NiftyView.html. Further data and files supporting this project are hosted in the *GigaScience* repository doi:10.5524/100222.

Competing interests
None

Author's contributions
Author's contributions statement is missing.

Acknowledgements
Acknowedgements section is missing.

References