A simple web-based tool for multi-spectral surface visualization

Snehal Padhye, David Messinger, James A. Ferwerda
Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, One Lomb Memorial Drive
Rochester, NY USA 14623

ABSTRACT

In this paper we describe a web-based application we have developed that allows visualization of the multi-spectral reflectance properties of three-dimensional surfaces. To develop this application, we employ a 3D computer graphics framework that enables us to combine spectral data with other object features such as topography and reflectance properties. The cross-platform application runs in any modern web browser, has an interactive interface, and does not require the installation or maintenance of any custom or proprietary software packages. The application is configured as a standard HTML/JavaScript client-server system, which provides flexibility for working with data in local or remote setups.

Keywords: Multi-spectral Imaging, web-based visualization, open-source, cross-platform

1. INTRODUCTION

Multi-spectral imaging has been used to great advantage in astronomy, remote sensing, document analysis, and medical diagnosis [1][2][3]. Each spectral band provides unique information about the surface under observation. Precise measurement of the spectral band properties is crucial for applications and much research has been done in developing techniques in capturing and processing multi-spectral data [4][5]. Correct interpretation and visualization of the spectral information is equally important. Similarly, 3D modeling provides an edge over 2D imaging for better understanding of surfaces with topographic features. Combining multi-spectral imaging and 3D information in a visualization tool may help to gain larger insights in properties of complex surfaces.

Many packages provide multi-spectral and/or 3D visualization. However, simultaneous visualization of spectral bands is a rare feature. Additionally, many of these packages are platform-dependent and require maintenance of the packages to ensure compatibility. Also, these packages usually require local installations which may also lead to system security issues. In this paper, we introduce a simple and easy to use browser-based visualization tool that enables users to visualize the different spectral and topographical properties of a surface at the same time. It does not require installation of any software packages, that helps to ensure system security. The application can be used across various platforms and accessed over the Web. In the following sections we describe the context, design, use, and applications of this visualization tool.

2. PREVIOUS WORK

Multi-spectral image visualization is supported by many standard image processing and analysis software packages. ENVI [6] is one popular software for visualization and analysis of spectral data, that is used in remote sensing and document imaging applications. Different bands can be rendered in different windows to visualize the spectral information beside each other. A more generic and widely used image processing software is MATLAB [7]. Spectral bands can be loaded into its standard display window, and multiple such windows can be used to visualize each individual spectral band. Similar multi-spectral image analysis is supported by ImageLab [8]. All these packages are loaded with features for multi-spectral visualization and analysis, but their interface can be cumbersome to use due to their generic nature. They usually also require heavy local installation of their executables and supporting files. Moreover, they are usually proprietary and have very limited features in their free versions.

Other than commercially available software, there has been plenty of research on developing free and open-source multi-spectral imaging and visualization tools. MultiSpec [9] is one such package largely designed for remote sensing applications. Opticks [10] is another freeware package. Gerbil [11] is an open-source software with an interactive support for multi-spectral visualization. While all these packages provide powerful features, they still require local installation of

Algorithms, Technologies, and Applications for Multispectral and Hyperspectral Imaging XXVII, edited by Miguel Velez-Reyes, David W. Messinger, Proc. of SPIE Vol. 11727, 1172716
 · doi: 10.1117/12.2587799

the software and supporting libraries. Since, spectral bands can have correlated information, dynamic simultaneous visualization of individual bands becomes useful, which is also missing in the existing software.

3D and multi-spectral data has always been considered separately in visualization tools. WebGL [12] has enabled development of many such 3D visualization platforms [13]. The most popular is Sketchfab [14]. It provides diverse tools for interaction with user-created 3D surfaces. Smithsonian museum X3D [15] is a package that was developed by Autodesk to create 3D models of the vast collection at Smithsonian museum. 3DHOP [16] has a rich set of tools for cultural heritage model visualization. While Smithsonian museum X3D is proprietary, 3DHOP is completely free and open source. While all these tools are of great value, there is a disconnect between the 3D and multi-spectral data. We have developed a tool that enables simultaneous visualization of different bands of spectral data with the flexibility of adding 3D topographical features for a rich and intuitive visualization.

3. WEB-BASED MULTI-SPECTRAL VISUALIZATION

Traditionally, spectral data visualization is similar to a 2D image visualization, with each pixel denoting a value of the selected individual band. We use 3D computer graphics in our visualization tool to add the flexibility of combining the spectral data with its surface and material properties. The development of the tool has following objectives:

- Multi-spectral data visualization
- Simultaneous display of multiple bands
- Browser-based
- Intuitive and easy-to-use interface
- No applications or libraries to install and maintain
- No security issues
- Free and open-source
- Support for richer datasets (shape, texture, material)

The first step towards developing such a tool requires a platform independent framework that is not limited by libraries or drivers of a given system environment. Browsers are available in all devices and requires no additional installations which make them the best candidate for implementation of our tool. Moreover, WebGL has brought the power of 3D graphics to browsers, and we can easily ensure support for rich datasets. A browser-based application enables users to access the tool with standard URLs, can be used on any browser-compatible system, and protects users from security threats arising from third-party software installations. Having decided on the basic technology, we develop an intuitive interface for simultaneous visualization of spectral data using WebGL.

The application ecosystem consists of the application code running on a browser and a user interacting with the interface. The application ecosystem can be seen pictorially in Figure 1. The application code can itself be represented in three layers [17]. The lowermost layer is the core layer consisting of graphics enabling libraries. The lowermost layer in our application is WebGL. The middle layer consists of APIs to provide higher level graphics and interactive functionalities. They are built over the lower-level libraries, and for this we use Three.js [18], based on WebGL. The uppermost layer consists of interface specific code for providing features such as simultaneous visualization of spectral bands. The interface code is based on HTML and JavaScript.

The Three.js pipeline of the application is shown in Figure 2. Geometry and material properties are important for a surface. They together form a mesh which is added to a Three.js scene. The scene also requires an illumination source. A camera is added to the renderer which is responsible for bringing all the components together to render images of the scene. It also supports interaction by updating scene transformations in real time.

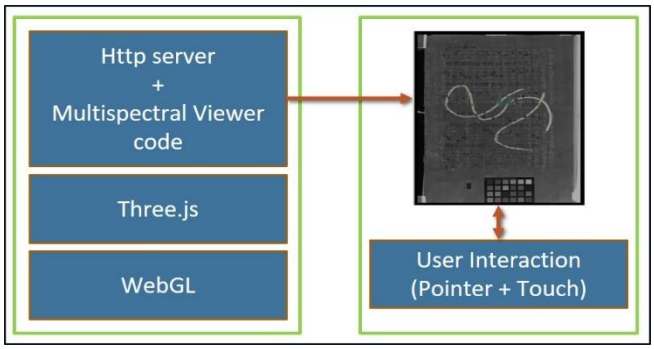

Figure 1. Application Ecosystem: WebGL forms the lowermost layer providing the basic graphics capability. Three.js uses WebGL to provide higher level functionalities to manage a rendered scene as a middle layer. The topmost layer consists of our application code, built over Three.js, to provide multi-spectral visualization. The application can be run through an HTTP server to display multiple spectral bands of a surface. The user can then observe and interact with the rendered surface.

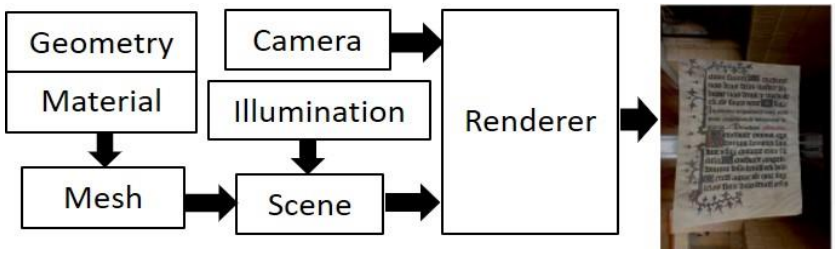

Figure 2. Three.js Pipeline: The geometry and material properties of a surface are specified and turned into a mesh that is a component in a scene that also includes light source. The renderer takes this information along with a camera and produces an image. Using Three.js all these steps can be performed in real-time on a standard web-browser.

Three.js provides the ability to represent the properties of a surface in terms of different 2D maps. We take advantage of this property, and use the spectral bands along with the 3D surface variations as texture and depth maps respectively. The application code is not able to fetch these maps, physically located on the system, and render them together due to the Cross Origin Resource Sharing (CORS) mechanism that is a security feature of Web software. Therefore, we use a simple webserver to solve this problem and thus, our application runs in a client-server configuration. There are two different ways of running the application, described below.

3.1 Local Setup

In the Local setup, both the webserver and the client run on a local network in a local configuration as shown in Figure 3. To start the visualization interface, a user can enter the local network URL for the server (http://<server IP address>:<port number>/) in a browser. For example, a simple Python webserver can be run by typing 'python -m HTTP.Server 8000' in a command terminal. This starts the webserver in the local system at port 8000. To launch the application interface on the local system itself, the user would type 'http://localhost:8000'. To run the interface on a different device on the same local network, the user would replace 'localhost' by IP address of the machine on which the server is running.

3.2 Remote Setup

When the webserver-running machine is accessible over the Web, the application interface can run on any remote system. The interface is launched by putting the correct server IP address and port in the URL. Any user can take advantage of this configuration to continue working remotely without running the server locally on the system. The remote mode configuration is also shown in Figure 3.

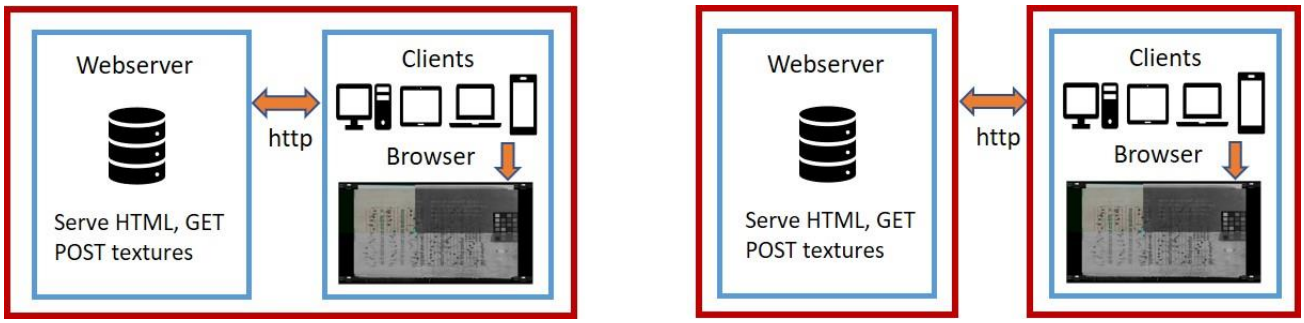

Figure 3. Left: Local setup: The server and the client run locally (represented by the outer boundary). The client can be any browser-running device. The client enters the server's local address and the port on which it is running, and the application runs on the client through the http protocol. The user can then have seamless interaction with the surface through the client system. Right: Remote Setup: The server and the client are on separate systems connected over Web. The client runs the application by connecting to the server through its IP address and its port over http protocol.

4. MULTI-SPECTRAL VISUALIZATION TOOLS

The multi-spectral visualization application has three modes for spectral band analysis: the Quad Viewer, Multi-spectral Lens and Multi-spectral highlighter. The user is prompted to provide the location of a folder containing the spectral band images. The application in the selected mode is launched and is initialized with the first few images given in the uploaded folder. The application provides a drop-down menu to change the displayed spectral bands in each mode. An important point to note here is that none of the image files from the uploaded folder are shared with the server. The application supports static files and the operation is always local to the user. This also means that even though the application can be accessed remotely, a user must have a local copy of their data.

4.1 Quad Viewer

As the name suggests, the Quad Viewer enables simultaneous visualization of four spectral bands of the target surface. Figure 4 shows a snapshot of Quad Viewer visualizing a page from an illuminated manuscript. The interface is composed of four quadrants with dynamically varying area controlled by a cyan colored knob, initially provided at the center. Each quadrant has a small GUI in its corner which allows a user to change the currently displayed spectral band. The Viewer uses the Three.js API to construct a scene for each quadrant. The Renderer is then asked to render each scene independently according to the current dimensions of each quadrant. This gives the user the ability to analyze interesting bands in the manuscript in all proportions of the quadrants as desired. The layout of the quadrants is manipulated by moving the cyan knob across the scene, allowing the user to quickly view a particular feature in all four bands in a common window.

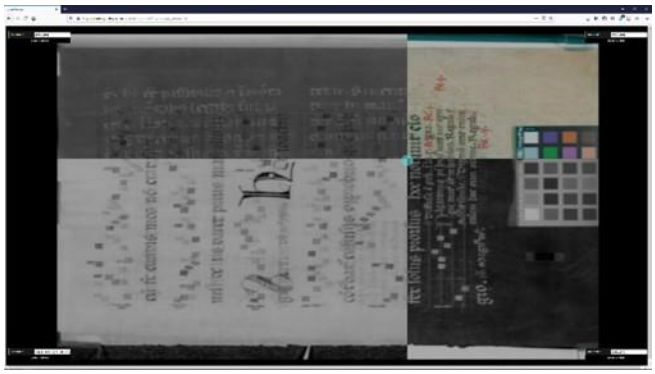

Figure 4. Web-based multi-spectral Quad Viewer: To access a multi-spectral dataset, a user types its URL into a browser. The browser then renders the image data and allows the user to interactively split the image into four quadrants and select the spectral bands shown in each quadrant.

4.2 Multi-spectral Lens Viewer

While the Quad Viewer provides simultaneous visualization of four spectral bands in distinct quadrants, the Multi-spectral lens viewer, enables simultaneous visualization of two bands but in overlapping regions. This allows direct comparison of

information between the two bands. Figure 5 shows a snapshot of the Multi-spectral Lens Viewer with viewing area as both window and lens. The figure shows different spectral bands of the same manuscript. The interface also has a small GUI at its right corner that allows selection of the spectral band to be displayed in each of the overlapping planes. The viewing area can be window shaped or a lens shaped and can be selected from the GUI. It also provides an option to change the size of the lens or the window.

4.3 Multi-spectral Highlighter

Both Quad Viewer and Multi-spectral Lens Viewer have geometrically well-defined areas to visualize spectral components. In contrast, the Multi-spectral Highlighter, provides simultaneous visualization of spectral bands in arbitrarily shaped user defined areas. Figure 6 shows the manuscript viewed using the Highlighter. It enables users to mark regions of interest by pointing/dragging and/or touching. A GUI at the top right corner of the interface provides the option to change the spectral band displayed in both the background and highlighted regions. It also provides the option to change the width of the region affected by a stroke.

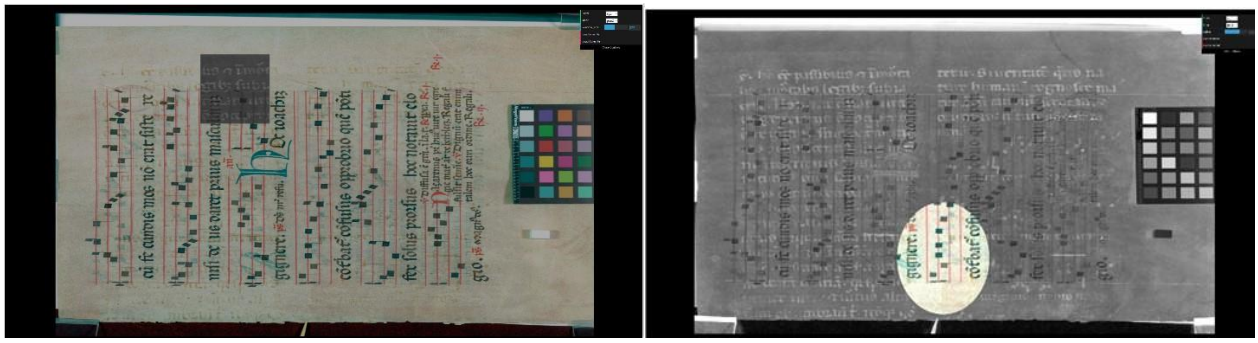

Figure 5. Multi-spectral Lens Viewer: In this viewer, the user can specify the size, position, and spectral properties of a rectangular or a circular region that acts as a 'lens' or filter to reveal particular surface properties. This provides increased flexibility in targeted visualization over the Quad Viewer.

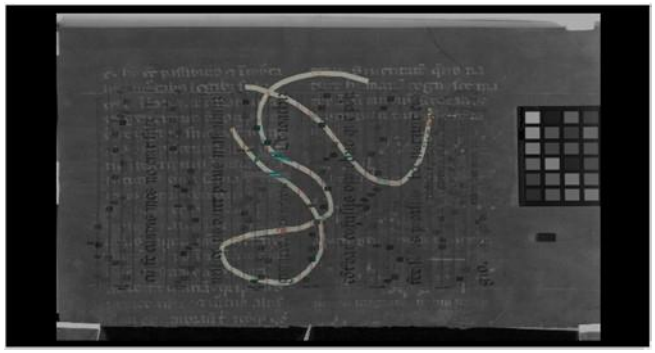

Figure 6. Multi-spectral Highlighter: Using a mouse or touch screen, the user can define the spectral rendering properties of precisely targeted and arbitrarily shaped region.

5. ENHANCED SURFACE PROPERTIES

The multi-spectral visualization application has the ability to add topographical and material properties of the surface, if available. Figure 7 shows one such example of the Quad Viewer with multi-spectral and topographical data of a Snap Bean field. Figure 7 also shows another example of a painting with its material properties. The top left quadrant shows the painting with its diffuse, specular and normal maps while the other quadrants show them individually.

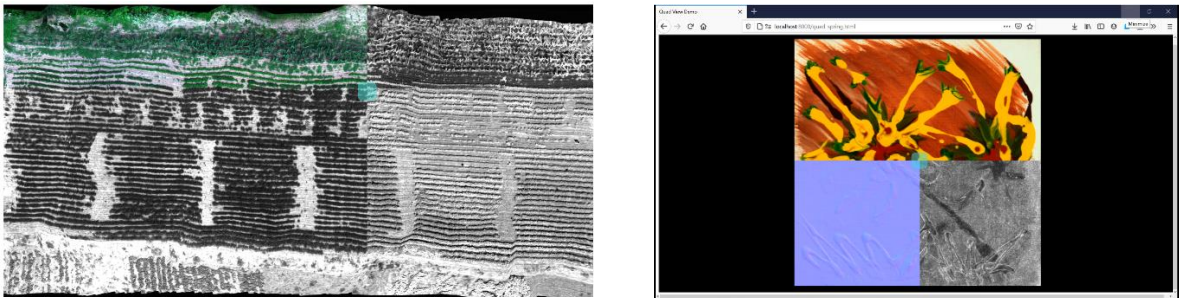

Figure 7. Left: Quad Viewer with topographical features of a Snap Bean field. Right: Quad Viewer with material properties of a painting.

6. ASSESSMENT

In this section we assess the application against factors important for any software, program or package.

6.1 Performance

The performance of any web-based visualization application depends on its ability to handle large data files and its efficiency in rendering the data [19]. Three.js uses the Nexus engine [20] that employs progressive streaming of view-dependent representations along with data compression to streamline transmission. Since, our application does not require very complicated scenes, the only bottleneck should be file size. Very large files (30MB+) may slow the rendering initially but does not affect real-time interaction, once the surface is rendered.

6.2 Security

Protecting systems from security threats due to installation of third-party packages was one of our core objectives in developing the tools. Browsers operate in a sandboxed environment that makes accessing local computer resources very difficult. This is usually seen as a disadvantage from a developer's perspective because it makes operations like writing temporary data to the disk very difficult. However, from a user's perspective, this feature ensures that their system resources are protected and safe. This is a common concern particularly for libraries, museums, and archives who have potentially significant, and expensive, collections.

6.3 Device Support

Device or hardware support is another important property of an application. Three.js use WebGL to for graphics rendering and hence the application can be run on any desktop or mobile browser that supports WebGL. The latest list includes [18] Google Chrome 9+, Firefox 4+, Opera 15+, Internet Explorer 11, Safari 5.1+ and Microsoft Edge.

6.4 Provenance and Intellectual Property Protection

Two of the greatest concerns in the cultural heritage community about using web-based visualization and dissemination tools are data provenance and intellectual property (IP) protection. Our application framework provides support for both of these concerns. In both the local and remote setups, data is uploaded and used at the client side. The application just hosts static files that are not stored anywhere other than temporarily in user's browser. This protects the data from being exposed on a network, however the user must ensure that they have the data they want to analyze on the client system they are using.

6.5 Cost and Accessibility

Cost and accessibility are also important factors in choosing a visualization tool. Our tool is completely open-source and free. We can also add features to customize for a specific application domain. For example, the current version supports data from the client side for IP protection, but for open-source projects, the data can be stored and accessed from a server so that every user can visualize the data without having a local copy. The code is available freely on GitHub and can be used by accessing our server at - https://github.com/snehalpadhye/MultispectralViewer.

7. LIMITATIONS AND FUTURE WORK

Our tools are currently limited to visualizing near-planar surfaces such that the geometry and material maps are rendered on a planar surface mesh and the maps can only be in formats that are supported by WebGL and Three.js. We plan to extend our tools to support non-planar objects. We also plan to add the ability to rotate the objects in 3D to enhance the rendering of texture and material features along with simultaneous visualization of spectral bands. If required, we also plan to add an option for server-side storage for open-source projects and add image processing features as required by domain-specific applications.

8. CONCLUSION

We have presented a set of browser-based visualization applications for multi-spectral image data. They are free, and easy to use for visualization and analysis of multi-modal data (3D geometry, material, multi-spectral color) without the problems associated with installing software, or concerns about storage and security issues. The tools are written in HTML and JavaScript and can run on modern desktop, laptop, and mobile devices with just a URL. Web-based tools for visualization of multi-spectral data provide powerful, simple-to-use, and secure means to for analyzing and understanding multi-spectral data in remote sensing, medical imaging, and cultural heritage imaging. Our hope is that these tools will allow multi-spectral imaging to provide greater insights into the objects under study.

ACKNOWLEDGEMENT

We would like to thank Cary collection and Dr Jan Van Aardt and team at Rochester Institute of Technology (RIT) for sharing the manuscript data and the Snap Bean field data, respectively, and for allowing us to use the data for demonstration of this tool.

REFERENCES

[1] Coffey, V. C., "Multispectral Imaging Moves into the Mainstream," Optics and Photonics News. 23 (4): 18 (1 April 2012).
[2] Easton Jr, R., Knox, K. and Christens-Barry, W., “Multispectral imaging of the Archimedes palimpsest,” in Proceedings of 32nd Applied Imagery Pattern Recognition Workshop, IEEE-AIPR (2003).
[3] Ortega, S., Halicek, M., Fabelo, H., Callico, G. M. and Fei, B., "Hyperspectral and multispectral imaging in digital and computational pathology: a systematic review [Invited]," Biomed. Opt. Express 11, 3195-3233 (2020).
[4] Chabries, D., Booras, S., and Bearman, G.,” Imaging the past: Recent applications of multispectral imaging technology to deciphering manuscripts,” Antiquity, 77(296), 359-372 (2003).
[5] Wellington, D. F., Bell, J. F., Johnson, J.R., Kinch, K. M., Rice, M. S., Godber, A., Ehlmann, B. L., Fraeman, A. A., Hardgrove, C., the MSL Science Team,” Visible to near-infrared MSL/Mastcam multispectral imaging: Initial results from select high-interest science targets within Gale Crater, Mars,” *American Mineralogist*, 102 (6): 1202–1217 (2017).
[6] ENVI Software, https://www.l3harrisgeospatial.com/Software-Technology/ENVI.
[7] MATLAB Software, https://www.mathworks.com/products/matlab.html.
[8] ImageLab Software, http://www.imagelab.at/.
[9] Biehl, L., and Landgrebe, D., “MultiSpec: a tool for multispectral--hyperspectral image data analysis,” Computers & Geosciences. 28. 1153-1159. 10.1016/S0098-3004(02)00033-X (2002).
[10] Opticks Software, https://opticks.org/.
[11] Jordan, J. and Angelopoulou, E., “Gerbil - A Novel Software Framework for Visualization and Analysis in the Multispectral Domain, “in Proceedings Vision, Modeling, and Visualization, The Eurographics Association (2010).
[12] WebGL API, https://www.khronos.org/webgl/.
[13] Scopigno, R., Callieri, M., Dellepiane, M., Ponchio, F. and Potenziani, M., “Delivering and using 3D models on the Web: Are we ready?,” in Virtual Archaeology Review (2017).

[14] Sketchfab, https://sketchfab.com/.
[15] Smithsonian 3D Digitization, https://3d.si.edu/.
[16] Potenziani, M., Callieri, M., Dellepiane, M., Corsini, M., Ponchio, F. and Scopigno, R., “3DHOP: 3D Heritage Online Presenter,” Computers & Graphics,Volume 52 (2015).
[17] Potenziani, M., Callieri, M., Dellepiane, M. and Scopigno, R., "Publishing and Consuming 3D Content on the Web: A Survey", Foundations and Trends in Computer Graphics and Vision (2018).
[18] Three.js API, https://threejs.org/.
[19] Boutsi, Ioannidis and Soile,” An Integrated Approach to 3D Web Visualization of Cultural Heritage Heterogeneous Datasets,” Remote Sens (2019).
[20] Ponchio F. and Dellepiane M.,” Fast decompression for web-based view-dependent 3D rendering,” in Proceedings of the 20th International Conference on 3D Web Technology (2015).