Astronomy and Computing 10 (2015) 43–53

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

Web-based visualization of very large scientific astronomy imagery☆

E. Bertin [a,d], R. Pillay [b,*], C. Marmo [c,d]

[a] Univ Pierre et Marie Curie, Institut d'Astrophysique de Paris, UMR7095, Paris, F-75014, France
[b] C2RMF, Palais du Louvre - Porte des Lions, Paris 75001, France
[c] Univ Paris-Sud, Laboratoire GEOPS, UMR8148, Orsay, F-91405, France
[d] CNRS, France

ARTICLE INFO

Article history:
Received 21 March 2014
Accepted 15 December 2014
Available online 23 December 2014

Keywords:
Visualization
Scientific data
Web application
High resolution
HTML5

ABSTRACT

Visualizing and navigating through large astronomy images from a remote location with current astronomy display tools can be a frustrating experience in terms of speed and ergonomics, especially on mobile devices. In this paper, we present a high performance, versatile and robust client–server system for remote visualization and analysis of extremely large scientific images. Applications of this work include survey image quality control, interactive data query and exploration, citizen science, as well as public outreach. The proposed software is entirely open source and is designed to be generic and applicable to a variety of datasets. It provides access to floating point data at terabyte scales, with the ability to precisely adjust image settings in real-time. The proposed clients are light-weight, platform-independent web applications built on standard HTML5 web technologies and compatible with both touch and mouse-based devices. We put the system to the test and assess the performance of the system and show that a single server can comfortably handle more than a hundred simultaneous users accessing full precision 32 bit astronomy data.

1. Introduction

Although much of the extraction of information from astronomy science images is now performed "blindly" using computer programs, astronomers still rely on visual examination for a number of tasks. Such tasks include image quality control, assessment of morphological features, and debugging of measurement algorithms.

The generalization of standardized file formats in the astronomy community, such as FITS (Wells et al., 1981), has facilitated the development of universal visualization tools. In particular, SAOIMAGE (Vanhilst, 1990), ALADIN (Bonnarel et al., 1994), SKYCAT (Albrecht et al., 1997), GAIA (Draper, 2000) and DS9 (Joye and Mandel, 2003). These packages are designed to operate on locally stored data and provide efficient access to remote image databases by downloading sections of FITS data which are subsequently read and processed locally for display; all the workload, including image scaling, dynamic range compression, color compositing and gamma correction, is carried out client-side.

However, the increasing gap between storage capacities and data access bandwidth (Budman, 2011) makes it increasingly efficient to offload part of the image processing and data manipulations to the server, and to transmit some form of pre-processed data to clients over the network.

Thanks to the development of wireless networks and light mobile computing (tablet computers, smartphones), more and more scientific activities are now being carried out on-the-go outside an office environment. These possibilities are exploited by an increasing number of scientists, especially experimentalists involved in large international collaborations and who must interact remotely, often in real-time, with colleagues and data located in different parts of the world and in different time zones. Mobile devices have increasingly improved display and interfacing capabilities, however, they offer limited I/O performance and storage capacity, as well as poor battery life when under load. Web-based clients, or simply *Web Apps*, are the applications of choice for these devices, and their popularity has exploded over the past few years.

Thanks to the ubiquity of web browsers on both desktop and mobile platforms, *Web Apps* have become an attractive solution for implementing visual interfaces. Modern web browsers feature ever faster and more efficient JavaScript engines, support for advanced standards such as HTML5 (W3C, 2012) and CSS3 (W3C, 2011), not to mention interactive 3D-graphics with the recent

☆ This code is registered at the ASCL with the code entry ascl:1408.009.

* Corresponding author.
E-mail addresses: bertin@iap.fr (E. Bertin), ruven.pillay@culture.gouv.fr (R. Pillay), chiara.marmo@u-psud.fr (C. Marmo).

http://dx.doi.org/10.1016/j.ascom.2014.12.006
2213-1337/

WebGL API (Khronos Group, 2013). As far as data visualization is concerned, web applications can now be made sufficiently feature-rich so as to be able to match many of the functions of standalone desktop applications, with the additional benefit of having instant access to the latest data and being embeddable within web sites or data portals.

One of the difficulties in having the browser deal with science data is that browser engines are designed to display gamma-encoded images in the GIF, JPEG or PNG format, with 8-bits per Red/Green/Blue component, whereas scientific images typically require linearly quantized 16-bit or floating point values. One possibility is to convert the original science data within the browser using JavaScript, either directly from FITS (Lowe, 2011; Kapadia, 2013), or from a more "browser-friendly" format, such as e.g., a special PNG "representation file" (Mandel, 2014), or compressed JSON (Federl et al., 2011). In practice this is currently limited to small rasters, as managing millions of such pixels in JavaScript is still too burdensome for less powerful devices. Moreover, lossless compression of scientific images is generally not very efficient, especially for noisy floating-point data (e.g. Pence et al., 2009). Hence, currently, server-side compression and encoding of the original data to a browser-friendly format remains necessary in order to achieve a satisfactory user experience on the web client, especially with high resolution screens.

Displaying images larger than a few megapixels on monitors or device screens requires panning and/or pixel rebinning, such as in "slippy map" implementations (Google Maps™, OpenStreetMap[1] etc.). On the server, the images are first decomposed into many small tiles (typically 256 × 256 pixels) and saved as PNG or JPEG files at various levels of rebinning, to form a "tiled pyramid". Each of these small files corresponds to a URL and can be loaded on demand by the web client. Notable examples of professional astronomy web apps based on this concept include the Aladin Lite API (Schaaff et al., 2012), and the Mizar plugin[2] in SITools2 (Malapert and Marseille, 2012).

However, having the data stored as static 8-bit compressed images means that interaction with the pixels is essentially limited to passive visualization, with little latitude for image adjustment or interactive analysis. Server-side dynamic processing/conversion of science-data on the server and streaming of the processed data to the web client are necessary to alleviate these limitations. Visualization projects featuring dynamic image conversion/streaming in Astronomy or Planetary Science have mostly relied on browser plugins implementing proprietary technologies (Federl et al., 2012) or Java clients/applets (Muller et al., 2009; Kitaeff et al., 2012). Notable exceptions include Helioviewer (Hughitt et al., 2008), which queries compressed PNG tiles directly from the browser with the tiles generated on-the-fly server-side from JPEG2000 encoded data.

In this paper we describe an open source and multi-platform high performance client–server system for the processing, streaming and visualization of full bit depth scientific imagery at the terabyte scale. The system consists of a light-weight C++ server and W3C standards-based JavaScript clients capable of running on stock browsers. In Section 2, we present our approach, the protocols and the implementation of both the server and the client. Sections 3 and 4 showcase several applications in Astronomy and Planetary Science. In Section 5, we assess the performance of the system with various configurations and load patterns. Finally in Section 6, we discuss future directions in the light of current technological trends.

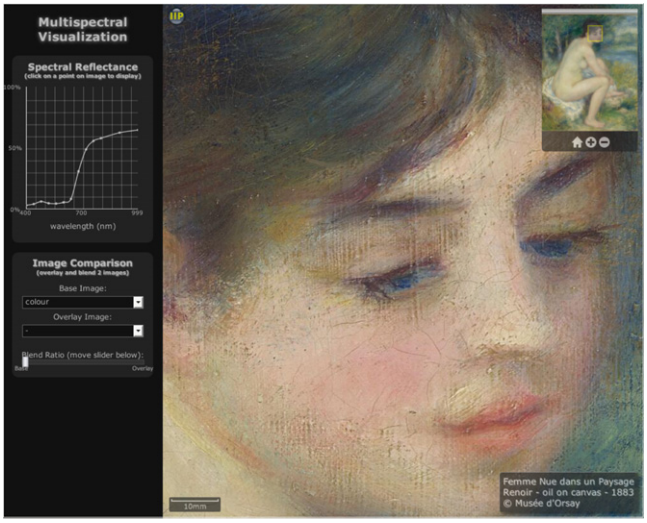

Fig. 1. Spectral visualization of Renoir's *Femme Nue dans un Paysage*, Musée de l'Orangerie, showing spectral reflectance curve for any location and controls for comparing different imaging modalities.

2. Material and methods

The proposed system consists of a (or several) central image server(s) capable of processing 32 bit floating point data on-demand and of transcoding the result into an efficient form usable by both light-weight mobile devices or desktop computers.

2.1. Image server

At the heart of the system is the open source IIPIMAGE[3] image server (Pitzalis et al., 2006). IIPIMAGE is a scalable client–server system for web-based streamed viewing and zooming of ultra high-resolution raster images. It is designed to be fast, scalable and bandwidth-efficient with low processor and memory requirements.

IIPIMAGE has a long history and finds its roots in the mid 1990s in the cultural heritage field where it was originally created to enable the visualization of high resolution colorimetric images of paintings (Martinez et al., 1998). The original system was designed to be capable of handling gigapixel size, scientific-grade imaging of up to 16 bits per channel, colorimetric images encoded in the CIEL*a*b* color space and high resolution multispectral images (Martinez et al., 2002) (Fig. 1). It had hitherto been very difficult to simply even view such image data locally, let alone access it remotely, share or collaborate between institutions. The client–server solution also enabled integration of full resolution scientific imaging such as infra-red reflectography, Xray, multispectral and hyperspectral imagery (Fig. 2) into museum research databases, providing for unprecedented levels of interactivity and access to these resources (Lahanier et al., 2002).

Beyond cultural heritage, the system has also been adapted for use in the field of biomedical imaging. For example, to visualize ultra-large high resolution electron microscopy maps created by ultra-structural mapping or *virtual nanoscopy* (Faas et al., 2012), or to explore high resolution volumetric 3D cross-sectional atlases (Husz et al., 2012).

In practice the IIPIMAGE platform consists of a light-weight C++ Fast-CGI (Brown, 1996) server, `iipsrv`, and an AJAX-based web interface. Image data stored on disk are structured

[1] http://www.openstreetmap.org.
[2] https://github.com/TPZF/RTWeb3D.
[3] http://iipimage.sourceforge.net.

Fig. 2. Hyperspectral imaging of paintings.

in order to enable efficient and rapid multi-resolution random access to parts of the image, allowing terapixel scale data to be efficiently streamed to the client. As only the region being viewed needs to be decoded and sent, large and complex images can be managed without onerous hardware, memory or network requirements by the client. The IIPIMAGE server performs on-the-fly JPEG compression for final visualization, but as the underlying data is full bit depth uncompressed data, it can operate directly on scientific images, and perform operations such as rescaling or filtering before sending out the results to the client.

IIPImage, therefore, possessed many of the attributes necessary for astronomy data visualization and rather than develop something from scratch, it was decided to leverage this existing system and extend it. A further benefit to this approach would be access to a larger scientific community beyond that of astronomy with certain similar data needs. Moreover, as IIPImage forms part of the standard Debian, Ubuntu and Fedora Linux distributions, access to the software, installation and maintenance would be greatly simplified and sustainable in the longer term.

Hence a number of modifications were made to the core of IIPIMAGE in order to handle astronomy data. In particular, to extend the system to handle 32 bit data (both integer and IEEE floating point), FITS metadata, functionality such as dynamic gamma correction, colormaps, intensity cuts, and to be capable of extracting both horizontal and vertical data profiles. The resulting code has been integrated into the main IIPIMAGE software development repositories and is available from the project website,[4] where it will form part of the 1.0 release of `iipsrv`.

2.2. Data structures and format

Extracting random image tiles from a very large image requires an efficient storage mechanism. In addition to tile-based access, the possibility to rapidly zoom in and out imposes some sort of multi-resolution structure on the data provided to the server. The solution adopted for "slippy map" applications is often simply to store individual tiles rebinned at the various resolution levels as individual image files. For a very large image this can translate into hundreds of thousands of small files being created. This approach is not convenient from a data management point of view, and for IIPIMAGE a single file approach has always been preferred.

The current version of IIPIMAGE supports both TIFF and JPEG2000 formats. Multi-resolution encoding is one of the major features of JPEG2000, but the lack of a robust, high performance open source library has been a serious issue until recently. Nevertheless, the encoding of floating point values spanning a large dynamic range remains a concern with current open-source libraries, as in practice input data is managed with only fixed point precision (Woodring et al., 2011; Kitaeff et al., 2014).

The combining of tiling and multi-resolution mechanisms is also possible with TIFF. TIFF is able to store not only 8 bit and 16 bit data, but also 32 bit integers, and single or double precision floating point numbers in IEEE format. As a well supported and mature standard with robust and widely used open source development libraries readily available, TIFF was adopted as the main server-side storage format, rather than creating a completely new format or adapting existing science formats in some way.

2.3. Image transcoding

Astronomy imaging data are usually stored in the FITS format (Wells et al., 1981). FITS is a flexible container format that can handle data encoded in up to 64 bits per value. FITS supports image tiling, whereby the original raster is split into separate rectangular tiles, which can be retrieved quickly and read and decoded independently (Pence et al., 2000). Versions of the same image could be stored at multiple resolution levels in different extensions, at the price of an increased file size. However currently neither tiling nor multi-resolution is present in archived FITS science images. Hence, regardless of the adopted storage format (TIFF in our case), a considerable amount of pixel shuffling and rebinning must be carried out in order to convert FITS data before they can be handled by the server.

Transcoding from basic FITS to multi-resolution tiled TIFF is carried out via the STIFF conversion package (Bertin, 2012). The multi-resolution structure consists of an image "pyramid" whereby pixels in each image are successively rebinned 2×2 and stored in separate TIFF virtual "directories" in tiled format. Tile size remains constant across all resolution levels (Fig. 3). The total number of pixels stored in the pyramid is increased by approximately one third compared to the original raster, but TIFF's widespread support for various lossless compression algorithms (e.g., LZW, Deflate) mitigates some of this extra structural overhead. Note that using pixel rebinning instead of decimation (as in traditional astronomy image display tools) averages out background noise as one zooms out: this makes faint background features such as low surface brightness objects or sky subtraction residuals much easier to spot.

The default orientation for TIFF images (and most image formats) is such that the first pixel resides in the upper left corner of the viewport, whereas FITS images are usually displayed with the first pixel in the lower left corner. To comply with these conventions, STIFF flips the original image content along the y direction by proceeding through the FITS file backwards, line-by-line.

STIFF takes advantage of the TIFF header "tag" mechanism to include metadata that are relevant to the IIPIMAGE server and/or web clients. For instance, the `ImageDescription` tag is used to carry a verbatim copy of the original FITS header. Another set of information of particular importance, especially with floating point

[4] http://iipimage.sourceforge.net.

Table 1
Main commands available in IIPImage.

Command	Description
FIF	Image path *p*. [FIF = p]
OBJ	Property/ies *text* to be retrieved from image and server metadata. [OBJ = text]
QLT	JPEG quality factor *q* between 0 (worst) and 100 (best). [QLT = q]
SDS	Specify a particular image within a set of sequences or set of multi-band images. [SDS = s1, s2]
CNT	Contrast factor *c*. [CNT = c]
CVT	Return the full image or a region, in JPEG format. [CVT = jpeg]
WID	Width *w* in pixels of the full sized JPEG image returned by the CVT command (interpolated from the nearest resolution). [WID = w]
HEI	Height *h* in pixels of the full sized JPEG image returned by the CVT command (interpolated from the nearest resolution). [HEI = h]
RGN	Define a region of interest starting at relative coordinates *x*, *y* with width *w* and height *h*. [RGN = x, y, w, h]
ROT	Rotate the image by *r* (90°, 180° or 270°). [ROT = r]
JTL	Return a tile with index *n* at resolution level *r*, in JPEG format. [JTL = r, n]
SHD	Apply hillshading simulation with azimuth and altitude angles *a*, *b*. [SHD = a, b]
SPECTRA	Return pixel values in all image channels for a particular point *x*, *y* on tile *t* at resolution *r* in XML format. [SHD = r, t, x, y]

Table 2
List of new commands implemented in IIPImage.

Command	Description
CMP	Set the colormap for grayscale images. Valid colormaps include GREY, JET, COLD, HOT, RED, GREEN and BLUE. [CMP = JET]
INV	Invert image or colormap. [Doesnotrequireanargument]
GAM	Set gamma correction to *g*. [GAM = g]
MINMAX	Set minimum min and maximum *max* for channel *c*. [MINMAX = c : min, max]
PFL	Request full bit-depth data profile for resolution *r* along the line joining pixel *x*1, *y*1 to *x*2, *y*1. [PFL = r : x1, y1 − x2, y2] *Note: Only horizontal* ($y1 = y2$) *and vertical profiles* ($x1 = x2$) *currently supported*

Fig. 3. Illustration of tiled multi-resolution pyramid with 4 levels of resolution.

data, is stored in the SMinSampleValue and SMaxSampleValue tags: these are the minimum and maximum pixel values ($S_{\min}$ and $S_{\max}$) that define the display scale. These values do not necessarily represent the full range of pixel values in the image, but rather a range that provides the best visual experience given the type of data. STIFF sets $S_{\max}$ to the 999th permil of the image histogram by default. $S_{\min}$ is computed in a way that the sky background $S_{\rm sky}$ should appear on screen as a dark gray $\rho_{\rm sky} \approx 0.001$ (expressed as a fraction of the maximum display radiant emittance: $1 \equiv$ full white):

$$S_{\min} = \frac{S_{\rm sky} - \rho_{\rm sky} S_{\max}}{1 - \rho_{\rm sky}}. \quad (1)$$

STIFF currently takes simply the median of all pixel values in the FITS file to compute $S_{\rm sky}$, although better estimates could be computed almost as fast (Bertin and Arnouts, 1996).

Transcoding speed can be a critical issue, for instance in the context of real-time image monitoring of astronomy observations. On modern hardware, the current STIFF conversion rate for transcoding a FITS file to an IIPImage-ready tiled pyramidal TIFF ranges from about 5 Mpixel/s to 25 Mpixel/s (20–100 MB/s) depending on the chosen TIFF compression scheme and system I/O performance. This means that FITS frames with dimensions of up to 16k × 16k pixels can be converted in a matter of seconds, and just-in-time conversion is a viable option for such images. Note that although STIFF is multithreaded, all calls to libtiff for writing tiles are done sequentially in the current implementation and there may, therefore, be some room for significant performance improvements.

2.4. Protocol and server-side features

IIPImage is based on the Internet Imaging Protocol (IIP), a simple HTTP protocol for requesting images or regions within an image, which allows the user to define resolution level, contrast, rotation and other parameters. The protocol was originally defined in the mid-1990s by the *International Imaging Industry Association* (Hewlett Packard, Live Picture, Eastman Kodak, 1997), but has since been extended for IIPImage. The use of such a protocol provides a rich RESTful-like interface to the data, enabling flexible and consistent access to imaging data. IIPImage is also capable of communicating using the simpler tile request protocols used by Zoomify or Deepzoom and the more recent IIIF image access API (International Image Interoperability Framework, Sanderson et al., 2013).

Table 1 lists the main commands already available in the original, cultural heritage-oriented version of IIPImage. For a complete description of the protocol, see the full IIP protocol specification (Hewlett Packard, Live Picture, Eastman Kodak, 1997).

For this project the entire IIPImage codebase was updated and generalized to handle up to 32 bits per pixel, with support for single precision floating point data. Support for double precision (at the expense of performance) would require a relatively simple code update. In addition, several extensions were implemented that allow the application of predefined colormaps to grayscale images, adjust the gamma correction, change the minimum and maximum cut-offs of the pixel value range, and that enable the export of image data profiles. A list of the new available commands is given in Table 2.

2.4.1. Examples

In order to better understand how these commands can be used, here are several examples showing the typical syntax and usage for applying colormaps, setting a gamma correction and for obtaining a full bit-depth profile.

All requests take the general form:

```
<protocol>://<server address>/<iipsrv>?<IIP Commands>
```

The first IIP command must specify the image path and several IIP command–value pairs can be chained together using the separator & in the following way:

```
FIF=<image path>&<command>=<value>&<command>=<value>
```

Thus, a typical request for the tile that fits into the smallest available resolution (tile 0 at resolution 0) of a TIFF image named `image.tif` is:

```
http://server/iipsrv.fcgi?FIF=image.tif&JTL=0,0
```

Let us now look at some more detailed examples using the new functionality created for IIPImage. For example, in order to export a profile in JSON format from pixel location x_1, y_1 horizontally to pixel location x_2, y_2 at resolution r on image `image.tif`, the request would take the form:

```
FIF=image.tif&PFL=r:x1,y1-x2,y2
```

In order to request tile t at resolution r and apply a standard *jet* colormap to image `image.tif`, the request would take the form:

```
FIF=image.tif&CMP=JET&JTL=r,t
```

and the equivalent inverted colormap request:

```
FIF=image.tif&CMP=JET&INV&JTL=r,t
```

In order to obtain metadata containing the minimum and maximum values per channel:

```
FIF=image.tif&OBJ=min-max-sample-values
```

In order to request tile t at resolution r and apply a gamma correction of g and specify a minimum and maximum of m_1 and m_2 respectively for image band b:

```
FIF=image.tif&MINMAX=b:m1,m2&GAM=g&JTL=r,t
```

Commands are not order sensitive excepting JTL and CVT that must always be specified last.

2.5. Security

A client–server architecture also has the advantage in terms of control and security of the data as the raw data at full bit depth does not necessarily need to be made fully available to the end user. Indeed, the raw data need never be directly accessible by the public and can be stored on firewalled internal storage and only accessible via the IIPImage server. Thus only 8 bit processed data is ever sent out to the client and restrictions and limits can be applied if fully open access is not desired. The IIPImage server also contains several features for added security, such as a path prefix, which limits access to a particular subdirectory on the storage server. Any requests to images higher up or outside of this subdirectory tree are blocked.

If an even greater level of security is required on the transmitted data, the IIPImage server can also dynamically apply a watermark to each image tile with a configurable level of opacity. Watermarking can be randomized both in terms of which tiles they are applied to as well as their position within the tile itself, making removal of watermarks extremely difficult.

2.6. Web clients

Two web clients, developed using different approaches and different goals in mind, are presented in this paper as examples to illustrate the capabilities of the system.

The first one, known as VisiOmatic, is built on top of the Leaflet JavaScript mini-framework, and is designed to display large celestial images through a classic image tile-based view.

The second client builds on the existing IIPMooViewer client to demonstrate two experimental features more specifically relevant to planetary surface studies: hillshading and advanced compositing/filtering performed at the pixel level within the browser.

3. Astronomy applications

3.1. Celestial images

Two essential features of astronomy image browsers are missing in the IIPMooViewer client originally developed for cultural heritage applications: the handling of celestial coordinates and a comprehensive management system for vector layers (overlays). It soon became clear that developing such a system from scratch with limited human resources would raise severe maintenance issues and portability concerns across browsers and platforms. We investigated several JavaScript libraries that would provide such functionality and decided to build a new client, VisiOmatic, based on the Leaflet library (Agafonkin, 2010). Leaflet is open-source and provides all the necessary functions to build a web client for browsing interactive maps. It is, in fact, not simply a client, but a small framework, offering features not directly available in standard JavaScript such as class creation and inheritance. It has a well-documented, user-friendly API and a rich collection of plug-ins that significantly boost its potential, while providing many advanced programming examples. Indeed, VisiOmatic operates as a Leaflet plug-in and as such comes bundled as a NodeJS package. Documentation for the VisiOmatic API is available on the VisiOmatic GitHub page.[5]

Once the `iipsrv` server has been installed, embedding a zoomable astronomy image in a web page with the VisiOmatic and Leaflet JavaScript libraries is very simple and can be done with the following code:

```
<div id="map"></div>
 <script>
  var map = L.map('map'),
      layer = L.tileLayer.iip('/fcgi-bin/iipsrv.fcgi?
          FIF=/path/to/image.ptif').addTo(map);
 </script>
```

Leaflet was built from the ground up with mobile device support in mind. VisiOmatic capitalizes on this approach by defining the current map coordinates at the center of the viewport instead of the mouse position. This also makes the coordinate widget display area usable for input, copy or paste as coordinates do not change while moving the mouse. Celestial coordinates are handled through a custom JavaScript library that emulates a small subset of the WCS standard (Calabretta and Greisen, 2002), based on the FITS header content transmitted by the IIPImage server. Our simplified WCS library fits into Leaflet's native latitude–longitude coordinate management system, giving access to all layer contents directly in celestial coordinates. This makes it particularly easy to synchronize maps that do not use the same projection for e.g., orientation maps, "smart" magnifiers, or multi-band monitoring.

Changing image settings is done by appending the relevant IIP commands (see e.g., Table 2) to the `http GET` tile requests. Metadata and specific data queries, such as profile extractions, are carried out through AJAX requests. VisiOmatic also uses AJAX requests for querying catalogs from other domains, with the

[5] https://github.com/astromatic/visiomatic.

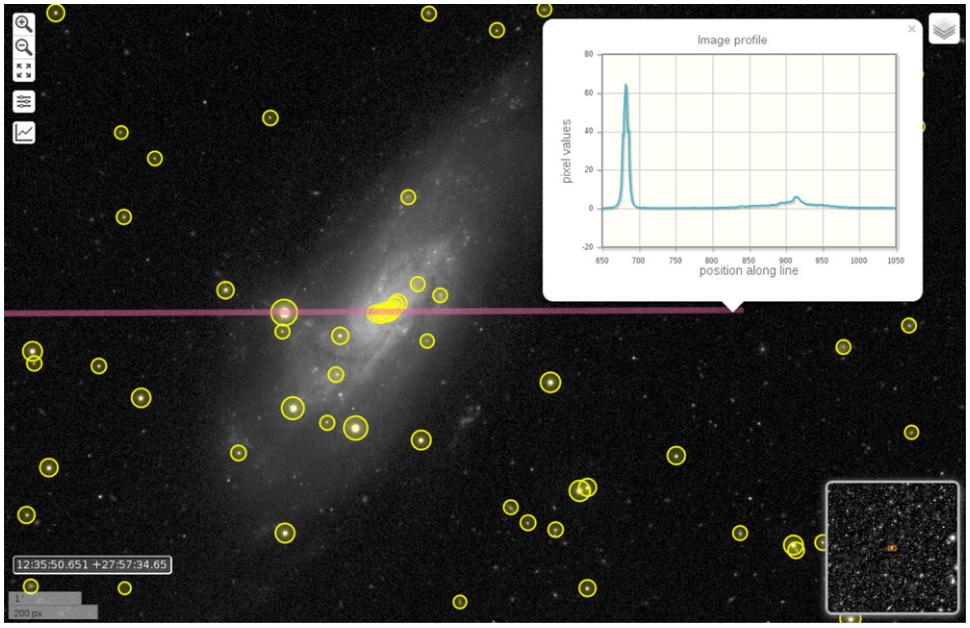

Fig. 4. The VisiOmatic web client showing a part of an SDSS release nine image stack (Ahn et al., 2012) provided by the IIPImage server in the main layer, plus two vector layers superimposed. Yellow: local detections from the photometric SDSS catalog provided by the Vizier service (Ochsenbein et al., 2000). Purple: horizontal profile through the image extracted by the IIPImage server.

restriction that the *same origin security policy*[6] present in current browsers requires that all requests transit through the image server domain, which must, therefore, be configured as a web proxy.

The VisiOmatic website[7] showcases several examples of applications built with the VisiOmatic client. They involve large images of the deep sky stored in floating point format, including a one terabyte (500,000 × 500,000 pixels) combination of 250,000 exposures from the 9th Sloan Digital Sky Survey data release (Ahn et al., 2012), representing about 3 TB worth of raw image data (Fig. 4).

Display performance with the VisiOmatic client varies from browser to browser. Browsers based on the WebKit rendering engine (e.g., Chrome, Safari) generally offer the smoothest experience on all platforms, especially with complex overlays. User experience may also vary because of the different ways browsers are able to deal with data. For example, examining images at exceedingly high zoom levels and scrutinizing groups of pixels displayed as blocks is common practice among astronomers. Leaflet takes advantage of the built-in resampling engines in browsers to allow image tiles to be zoomed in smoothly through CSS3 animations. VisiOmatic uses the `image-rendering` CSS property to activate nearest-neighbor interpolation and have the pixels displayed as blocks at higher zoom levels. Although this works in, for example, Firefox and Internet Explorer 11, other browsers, such as Chrome, do not offer the possibility to turn off bilinear interpolation at the present time, and zoomed images will not appear pixelated in those browsers. Hopefully, it is expected that such residual differences will eventually disappear as browser technology converges over standards.

4. Planetary science

Planetary Science data are largely heterogeneous with respect to the physical quantities they describe (chemical abundances, atmospheric composition, magnetic and gravitational fields of Earth-like planets and satellites, reflectance, surface composition) and with respect to the formats they are encoded in (raster, vector, time-series, in ASCII or various binary formats). Two scientific communities are essentially involved in Planetary Science research: astronomers and geologists/geophysicists.

Geographical Information Systems (GIS) are the basis for planetary surface studies but they often suffer from a lack of systematic and controlled access to pixel values for quantitative physical analyses on raster data (Marmo, 2012).

In Earth Sciences, distributors of GIS commercial software have been ready to exploit the potential of the Web. This is the case, for example, of the online ArcGIS WebMap Viewer (ESRI ArcGIS).

However, the difficulty in sending 16 or 32 bit precision scientific data using current web technologies has, hitherto, limited web visualization to public outreach applications such as the Microsoft World Wide Telescope available for images of Mars (Scharff et al., 2011) or Google Earth for Mars (Google Inc.).

Nevertheless, remote scientific visualization has been achieved with tools such as JMars (NASA/JPL-Caltech/Arizona State University) (Christensen et al., 2009) and HiView (University of Arizona), which are both Java clients that aim to visualize both remote and local data. The first is GIS based (layer superposition oriented) while HiView is more a remote sensing software (raster manipulation oriented) that is an ad-hoc product for HiRISE (HiRISE Team) and which uses the JPIP protocol for remote access to JPEG2000 imagery.

The visualization system we propose already supports basic manipulations on raster layers, raster layer superposition and could easily manage vector layer creation and superposition. It, moreover, enables access to full precision pixel values and provides a simple and generic solution for planetary applications, efficiently and elegantly blending both GIS and remote sensing approaches.

4.1. Color compositing

Color compositing is an essential feature in both GIS and remote sensing applications and is used to point out differences in surface composition by performing on-demand composition

[6] The *Cross-Origin Resource Sharing* (CORS) mechanism implemented in modern browsers could in principle prevent that, but it is not supported by the main astronomy data providers at this time.

[7] http://visiomatic.org.

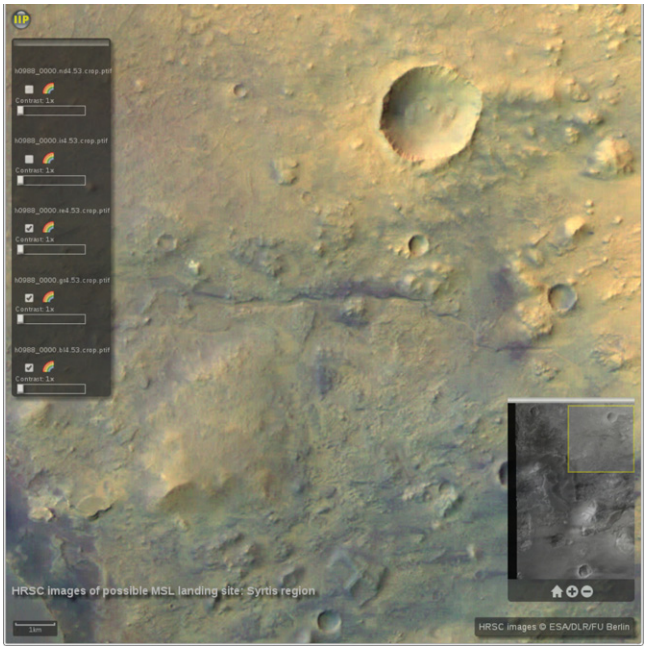

Fig. 5. Example of a planetary application using HRSC Mars images (ESA/DLR/FU Berlin/G. Neukum). The resulting color image is a linear combination of input channels. The mixing matrix is defined by a user-adjustable combination of Red, Green, Blue and a contrast factor for each input channel.

of specific color bands. Interactive color composition on the Web can be achieved using the HTML5 `canvas` element which allows us to directly access and manipulate image pixel values. This, therefore, enables more complex real-time image processing directly within the client and we have developed a version of our client making extensive use of HTML5 `canvas` properties[8] in order to implement on-demand color composition with multiple input channels (Fig. 5).

Color compositing performance depends essentially on canvas size (the overall image size is irrelevant as only the displayed part of the image needs to be processed). For the example cited above the processing time is about 1–3 ms per tile (256 × 256 pixels), depending on browser and client hardware.

4.2. Terrain maps—Hillshading

High resolution 3D data is not easy to stream or to make multi-resolution. Furthermore as IIPIMAGE is essentially image-centric, a 2D rendering approach to visualization was favored. In order to facilitate the use of DEM (digital elevation map) data, two approaches have been developed in our IIPIMAGE framework.

The first approach is the dynamic application of custom colormaps to grayscale images. A new command `CMP` has been added to the IIP protocol, which can also be useful for the visualization of other physical map data such as gas density, temperature or chemical abundances in real or simulated data.

In the second approach, elevation point data is converted to vector normal and height data at each pixel. In this form, they are also able to be stored within the standard TIFF format. The *XYZ* normal vectors can be packed into a 3 channel "color" TIFF, whereas the height data can be packed into a separate 1 channel monochrome TIFF. They are both, therefore, able to be tiled, compressed and structured into a multi-resolution pyramid for streaming with IIPIMAGE. A basic rendering technique for DEM data is that of hill-shading (Horn, 1981) where a virtual directional illumination is used to create shading on virtual "hills". A fast hill-shading algorithm has been implemented server-side in IIPIMAGE and extended to 32 bit data allowing the user to interactively set the angle of incidence of the light source and view a dynamically rendered hill-shaded relief map. An example showing a Mars terrain map from Western Arabia Terra can be seen online[9] and in Fig. 6.

5. Performance analysis

Although the use of JPEG compression as a delivery format significantly reduces the bandwidth required, data access, dynamic processing and compression of 32 bit data can impose significant server-side overhead, that will ultimately dictate the maximum number of users that a server will be able to handle. Timings and memory usage depend on image type, server settings and commands in the query; we chose to focus on the typical case of browsing a large, single channel, single precision floating-point image stored with tiles of 256 × 256 pixels in size. In order to fully test this, we created a large 131, 072 × 131, 072 pixel FITS image by combining contiguous SDSS i-band images using the SWARP package (Bertin et al., 2002). This large image was then converted to a 92 GB multi-resolution TIFF comprising 9 resolution levels using STIFF.

Our tests were performed on two Dell PowerEdge servers running GNU/Linux (Fedora distribution with kernel 3.11) and equipped with 2.6 GHz processors, 32 and 48 GB of RAM and a Perc5i internal RAID controller. In order to check the influence of the I/O subsystem on server performance, we installed the TIFF file on two different types of RAID:

(a) a RAID 6 array of 12 × 3 TBytes SAS (6 Gb/s) hard drives formatted with the XFS filesystem.
(b) a RAID 5 array of 6 × 1 Tbytes SATA3 (6 Gb/s) solid-state drives (SSDs) formatted with the Ext4 filesystem.

On both systems we obtain a typical sequential read speed of 1.2 GB/s for large blocks; but obviously access times are much lower on the RAID of SSDs (<1 ms vs. 15 ms for the one with regular hard drives).

The client consists of a third machine sending requests to any of the two servers through a dedicated 10 GbE network. We used a modified version of AB, the APACHEBENCH HTTP server benchmarking package, to send sequences of requests to random tiles among the 262,144 that compose the highest image zoom level. Appropriate system settings, as prescribed in Veal and Foong (2007), were applied server-side and client-side to ensure that both ends would stand the highest possible concurrency levels with minimum latencies and maximum throughput. We conducted preliminary tests through Apache's httpd,[10] Lighty Labs' lighttpd,[11] a combination of Nginx[12] and Lighty Labs' `spawn-fcgi` and finally LiteSpeed Technologies' OpenLiteSpeed.[13] We found the latter to offer the best combination of performance and robustness, especially at high concurrency levels; hence all the requests to `iipsrv` in the tests reported below were served through OpenLiteSpeed (one single `lshttpd` instance).

5.1. Timings

Fig. 7 shows the distribution of timings of the main tasks involved in the server-side processing of a tile, for several system

[8] http://image.iap.fr/iipcanvas/hrsc.html.

[9] http://image.iap.fr/iipdiv/hirise.html.
[10] http://httpd.apache.org/.
[11] http://www.lighttpd.net.
[12] http://nginx.org.
[13] http://open.litespeedtech.com.

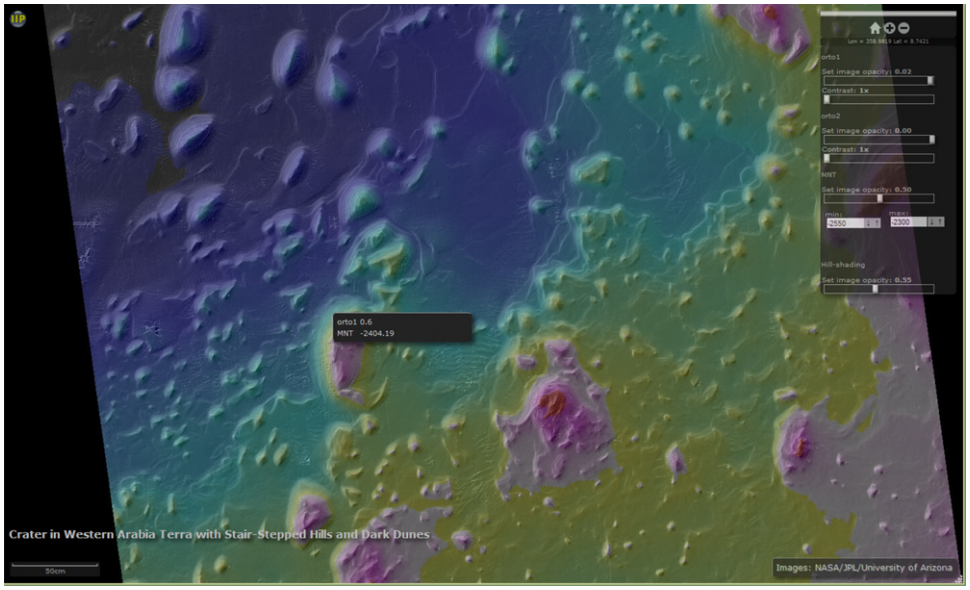

Fig. 6. Example of a planetary web application using HiRISE Mars images (NASA/JPL/University of Arizona). The digital elevation model (a floating point raster) is displayed using the JET colormap with cuts set by the user from the control panel. Superimposed is the hill-shading layer computed by the IIPImage server from the DEM; the azimuth incidence angle can be adjusted from the control panel.

and `iipsrv` cache settings. In order to probe the impact of I/O latencies, we set up an experiment where the server system page cache is flushed and the `iipsrv` internal cache is deactivated prior to running the test (upper row of Fig. 7). With such settings most accesses to TIFF raw tiles do not benefit from caching. As a consequence, `iipsrv` timings are dominated by random file access times when the data are stored on spinning hard disks, with access latencies reaching up to $\approx$ 25 ms in unfavorable situations. As expected, switching to SSDs reduces the uncached file access latencies to less than 1 ms.

However, in practice much better timings will be obtained with regular hard drives, as tiles are generally not accessed randomly. Moreover, leaving the system page cache un-flushed between test sessions when using spinning hard drives reduces access latencies to a few milliseconds (lower row in Fig. 7). Activating `iipsrv`'s internal cache system will further reduce latencies close to zero for tiles that were recently visited.

Further testing with TIFF images of different size was carried out in order to ensure that the system would also scale in terms of file size and the timings reported above remain roughly identical as file size increases up to at least 1.8 TB.

5.2. Concurrency and data throughput

Each single-threaded FastCGI process takes about 5–10 ms to complete, and is therefore capable of serving up to 100–200 256 $\times$ 256 "new" tiles per second. Higher tile serving rates are obtained by running several instances of `iipsrv` on servers with multiple CPU cores. But how is the system able to keep up with a large number of concurrent requests?

As Fig. 8 shows, the tile serving rate remains remarkably flat, and latency scales linearly with the concurrency level when the number of concurrent requests exceeds that of CPU cores. Setting a limit for average latency to $\sim$500 ms for comfortable image browsing, we see that a single 12-core web server can handle $\sim$700 concurrent 256 $\times$ 256 tile requests, which corresponds to about 100 users frantically browsing large, uncompressed, single-channel floating-point images. This estimation is well verified in practice, although it obviously depends on tile size and on the amount of processing carried out by `iipsrv`. Note that the average tile serving rate obtained with a single 12-core web server corresponds to a sustained data rate of 60 MB/s for 256 $\times$ 256 tiles encoded at a JPEG quality factor of 90; higher JPEG quality factors bring the data rate close to the saturation limit of a 1 GbE connection.

6. Conclusion and future work

A high performance web-based system for remote visualization of full resolution scientific grade astronomy images and data has been developed. The system is entirely open-source and capable of efficiently handling full resolution 32 bit floating point image and elevation map data.

We have studied the performance and scalability of the system and have shown that it is capable of handling terabyte-size scientific-grade images that can be browsed comfortably by at least a hundred simultaneous users, on a single server.

By using and extending an existing open source project, a system for astronomy has been put together that is fully mature, that will benefit from the synergies of the wider scientific imaging community and that is ready for use in a busy production environment. In addition the IIPImage server, is distributed as part of the default Debian, Ubuntu and Fedora package repositories, making installation and configuration of the system very straightforward. All the code developed within this project for `iipsrv` has been integrated into the main code base and will form an integral part of the 1.0 release. However, there are still many potential directions for improvements, both server-side and client-side. Most importantly:

- The TIFF storage format used on the server currently restricts pixel bit depth, the number of image channels, and I/O performance (through `libtiff`). A valid alternative to TIFF could be the Hierarchical Data Format Version 5 (HDF5) (HDF Group, 2000), which provides a generic, abstract data model that enables POSIX-style access to data objects organized hierarchically within a single file; some radio-astronomers have been trying to promote the use of HDF5 for storing massive and complex astronomy datasets (Masters et al., 2012). A more radical approach would be to adopt JPEG2000 as the archival storage format for astronomy imaging archives (Kitaeff et al., 2014), which could also remove the need for transcoding images for visualization purposes.

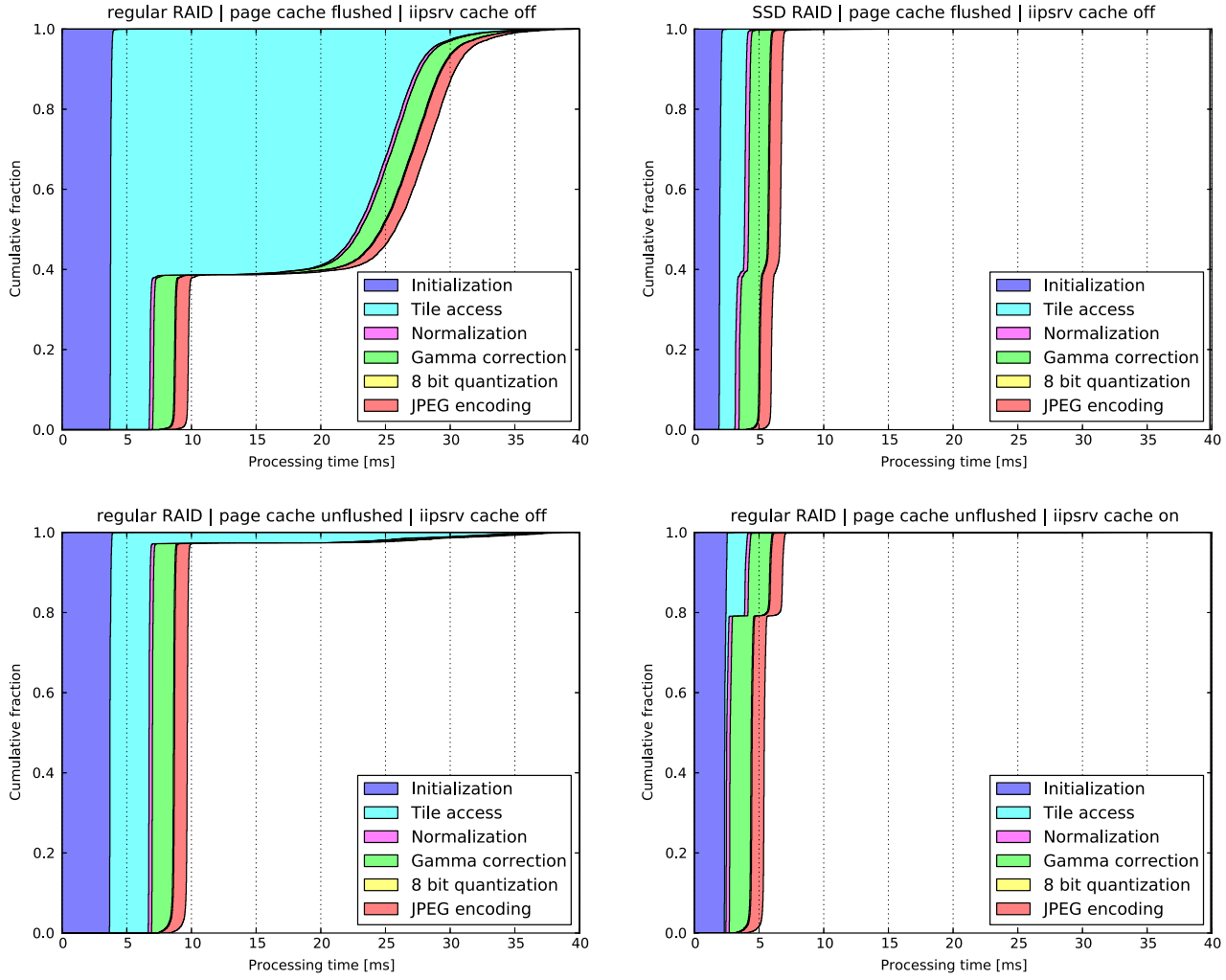

Fig. 7. Cumulative distributions for the timings of the main tasks involved in the processing of random 256 × 256 pixel tiles in four different contexts (see text for details). "Initialization" is the time taken to initialize various objects and (re-)open the TIFF file that contains the requested raw tile (call to TIFFOpen()). "Tile access" is the time spent accessing and reading the content of a raw tile. "Normalization" and "Gamma correction" respectively measure the time it takes to apply intensity cuts and to compress the dynamic range of pixel values for the whole tile. "8 bit quantization" is the time spent converting the tile to 8 bit format, while "JPEG encoding" is the time taken to encode the tile in JPEG format.

- Additional image operations could be implemented within `iipsrv`, including real-time hyperspectral image processing and compositing.
- Although the IIPImage image tile server already supports simple standard tile query protocols and interfaces easily with most image panning clients, a welcome addition would be to offer support for the more GIS-oriented WTMS (Web Map Tile Service) protocol (Open Geospatial Consortium, 2010).
- The International Virtual Observatory Alliance (IVOA) has agreed on a standard set of specifications for discovering and accessing remote astronomical image datasets: the Simple Image Access Protocol (SIAP) (Tody et al., 2011). The response to an SIAP query consists of metadata and download URLs for matching image products. Current SIAP specifications[14] do not provide specific ways to access pyramids of tiled images. Still, support for SIAP could be implemented within or outside of `iipsrv` for generating, for example, JPEG cutouts or lists of tiles that match a given set of coordinates/field of view/pixel scale.
- Both IIPMooViewer and Leaflet clients require all layers displayed on a map at the same moment to share the same "native" pixel grid (projection). Although this limitation does not prevent "blinking" images with different pixel grids, it precludes

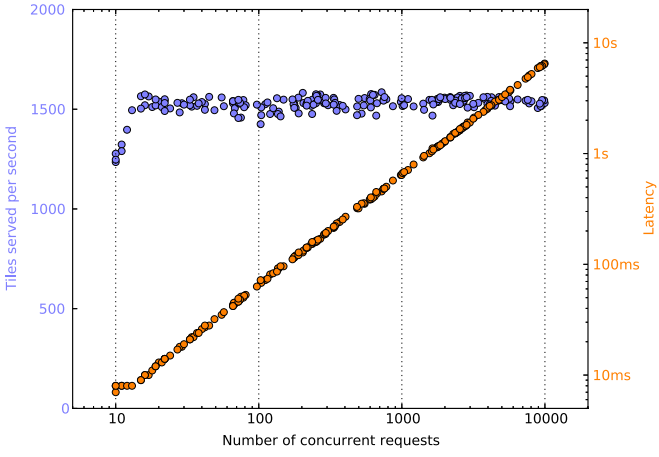

Fig. 8. Tile serving rate (in blue) and latency (in orange) as a function of the number of concurrent tile requests using 12 instances of `iipsrv` on a 12-core server equipped with an SSD RAID. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

[14] http://www.ivoa.net/documents/SIA/.

overlapping different observations/exposures on screen. For instance it makes it impossible to display accurately the entire focal plane of a mosaic camera on a common viewport, without prior resampling. Having different images with different native pixel grids sharing the same map would require the web-client to perform real-time reprojection. Client-side reprojection should be possible e.g., with version 3 of the OpenLayers library.[15]

Acknowledgments

The authors would like to thank the anonymous referees whose comments helped not only in improving the clarity of this paper, but also the performance of the code.

CM wishes to acknowledge Prof. Joe Mohr for hospitality at USM, Munich, and the SkyMapper team, in particular Prof. Brian Schmidt, Dr. Patrick Tisserand and Dr. Richard Scalzo for support during her stay at MSO-ANU, Canberra where part of this work was completed. EB thanks Raphael Gavazzi, Valérie de Lapparent, and the Origin and Evolution of Galaxies group at IAP, Paris for financial support with the VisiOmatic hardware, and Dr. Hervé Bouy at CAB, Madrid for providing content for the VisiOmatic demos.

The VisiOmatic client implements services provided by the Sesame Name Resolver and the VizieR catalog access tool developed at CDS, Strasbourg, France.

Some of our demonstration data are based on SDSS-III[16] images. Funding for SDSS-III has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Science Foundation, and the US Department of Energy Office of Science. SDSS-III is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS-III Collaboration including the University of Arizona, the Brazilian Participation Group, Brookhaven National Laboratory, Carnegie Mellon University, University of Florida, the French Participation Group, the German Participation Group, Harvard University, the Instituto de Astrofisica de Canarias, the Michigan State/Notre Dame/JINA Participation Group, Johns Hopkins University, Lawrence Berkeley National Laboratory, Max Planck Institute for Astrophysics, Max Planck Institute for Extraterrestrial Physics, New Mexico State University, New York University, Ohio State University, Pennsylvania State University, University of Portsmouth, Princeton University, the Spanish Participation Group, University of Tokyo, University of Utah, Vanderbilt University, University of Virginia, University of Washington, and Yale University.

References

Agafonkin, V., 2010. Leaflet—a JavaScript library for mobile-friendly maps. http://www.leafletjs.com.

Ahn, C.P., Alexandroff, R., Allende Prieto, C., Anderson, S.F., Anderton, T., Andrews, B.H., Aubourg, É., Bailey, S., Balbinot, E., Barnes, R., et al., 2012. The ninth data release of the sloan digital sky survey: first spectroscopic data from the SDSS-III Baryon oscillation spectroscopic survey. Astrophys. J. Suppl. Ser. 203, 21. 1207.7137.

Albrecht, M.A., Brighton, A., Herlin, T., Biereichel, P., Durand, D., 1997. Access to data sources and the ESO SkyCat tool. In: Hunt, G., Payne, H. (Eds.), Astronomical Data Analysis Software and Systems VI. p. 333.

Bertin, E., 2012. Displaying digital deep sky images. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. p. 263.

Bertin, E., Arnouts, S., 1996. SExtractor: software for source extraction. Astron. Astrophys. Suppl. Ser. 117, 393–404.

Bertin, E., Mellier, Y., Radovich, M., Missonnier, G., Didelon, P., Morin, B., 2002. The TERAPIX pipeline. In: Bohlender, D.A., Durand, D., Handley, T.H. (Eds.), Astronomical Data Analysis Software and Systems XI. p. 228.

Bonnarel, F., Paillou, P., Ochsenbein, F., Creze, M., Egret, D., 1994. Aladin: towards an interactive atlas of the digitized sky. In: Crabtree, D.R., Hanisch, R.J., Barnes, J. (Eds.), Astronomical Data Analysis Software and Systems III. pp. 215–218.

Brown, M., 1996. FastCGI Specification. http://www.fastcgi.com.

Budman, G., 2011. Price Gap: Storage vs Bandwidth. http://blog.backblaze.com/2011/06/22/price-gap-storage-vs-bandwidth (Last accessed 07/03/2012).

Calabretta, M.R., Greisen, E.W., 2002. Representations of celestial coordinates in FITS. Astron. Astrophys. 395, 1077–1122. astro-ph/0207413.

Christensen, P.R., Engle, E., Anwar, S., Dickenshied, S., Noss, D., Gorelick, N., Weiss-Malik, M., 2009. JMARS—A Planetary GIS. AGU Fall Meeting Abstracts, A6.

Draper, P.W., 2000. GAIA: recent developments. In: Manset, N., Veillet, C., Crabtree, D. (Eds.), Astronomical Data Analysis Software and Systems IX. p. 615.

ESRI ArcGIS, ArcGIS Web Map Viewer. http://www.arcgis.com/home/webmap/viewer.html (Last accessed 4 February 2014).

Faas, F.G.A., Avramut, M.C., Berg, B.M.v.d., Mommaas, A.M., Koster, A.J., Ravelli, R.B.G., 2012. Virtual nanoscopy: generation of ultra-large high resolution electron microscopy maps. J. Cell Biol. 198, 457–469.

Federl, P., Grimstrup, A., Kiddle, C., Taylor, A.R., 2011. On-line access and visualization of multi-dimensional FITS data, in: I. N. Evans, A. Accomazzi, D. J. Mink, & A. H. Rots (Ed.), Astronomical Society of the Pacific Conference Series, p. 467–470.

Federl, P., Grimstrup, A., Kiddle, C., Taylor, A.R., Robinson, K., Stephure, M., Yee, G., 2012. Remote visualization of large multi-dimensional radio astronomy data sets. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. p. 95.

Google Inc., Mars in Google Earth. http://www.google.com/earth/explore/showcase/mars.html (Last accessed 3 April 2013).

HDF Group, 2000. Hierarchical data format version 5.

Hewlett Packard, Live Picture, Eastman Kodak, 1997. Internet Imaging Protocol specification. http://iipimage.sourceforge.net/IIPv105.pdf.

HiRISE Team, High Resolution Imaging Science Experiment. http://hirise.lpl.arizona.edu/ (Last accessed 11 April 2013).

Horn, B., 1981. Hill shading and the reflectance map. Proc. IEEE 69, 14–47.

Hughitt, V.K., Ireland, J., Lynch, M.J., Schmeidel, P., Dimitoglou, G., Müeller, D., Fleck, B., 2008. Helioviewer: A Web 2.0 tool for visualizing heterogeneous heliophysics data. AGU Fall Meeting Abstracts -1, 1617.

Husz, Z., Burton, N., Hill, B., Milyaev, N., Baldock, R., 2012. Web tools for large-scale 3D biological images and atlases. BMC Bioinformatics 13, 122.

Joye, W.A., Mandel, E., 2003. New features of SAOImage DS9. In: Payne, H.E., Jedrzejewski, R.I., Hook, R.N. (Eds.), Astronomical Data Analysis Software and Systems XII. p. 489.

Kapadia, A., 2013. AstroJS Library. http://www.astrojs.org (Last accessed 12 August 2013).

Khronos Group, 2013. WebGL Specification Version 1.0.2. http://www.khronos.org/registry/webgl/specs/1.0/ (Last accessed 29/07/2014).

Kitaeff, V.V., Cannon, A., Wicenec, A., Taubman, D., 2014. Astronomical imagery: considerations for a contemporary approach with JPEG2000. ArXiv e-prints 1403.2801.

Kitaeff, V.V., Wu, C., Wicenec, A., Cannon, A.D., Vinsen, K., 2012. SkuareView: client–server framework for accessing extremely large radio astronomy image data. In: Proceedings of the 2012 workshop on High-Performance Computing for Astronomy Date. ACM, New York, NY, USA, pp. 25–32.

Lahanier, C., Aitken, G., Shindo, J., Pillay, R., Martinez, K., Lewis, P., 2002. EROS: an open source multilingual research system for image content retrieval dedicated to conservation-restoration exchange between cultural institutions. In: ICOM Committee for Conservation, ICOM-CC: 13th Triennial Meeting, Rio de Janeiro, 22–27 September 2002. ICOM-CC; James & James, Rio de Janeiro, Brazil, pp. 287–294.

Lowe, S., 2011. jsFITS Repository. https://github.com/slowe/jsFITS (Last accessed 12 August 2013).

Malapert, J.C., Marseille, M., 2012. SITools2: a framework for archival systems. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. p. 821.

Mandel, E., 2014. JS9 Website. http://js9.si.edu/ (Last accessed 17 July 2014).

Marmo, C., 2012. Development of a planetary Web GIS at the "Photothèque Planétaire" in Orsay. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. p. 411.

Martinez, K., Cupitt, J., Perry, S., 1998. High resolution colorimetric image browsing on the Web. Comput. Netw. ISDN Syst. 33, 399–405.

Martinez, K., Cupitt, J., Saunders, D., Pillay, R., 2002. Ten years of art imaging research. Proc. IEEE 28–41.

Masters, J., Alexov, A., Folk, M., Hanisch, R., Heber, G., Wise, M., 2012. The AstroHDF effort. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. p. 871.

Muller, D., Fleck, B., Dimitoglou, G., Caplins, B.W., Amadigwe, D.E., Ortiz, J.P.G., Wamsler, B., Alexanderian, A., Hughitt, V.K., Ireland, J., 2009. JHelioviewer: visualizing large sets of solar images using JPEG 2000. Comput. Sci. Eng. 11, 38–47.

NASA/JPL-Caltech/Arizona State University, JMars Website. http://jmars.mars.asu.edu/ (Last accessed 11 April 2013).

Ochsenbein, F., Bauer, P., Marcout, J., 2000. The VizieR database of astronomical catalogues. Astron. Astrophys. Suppl. Ser. 143, 23–32. astro-ph/0002122.

Open Geospatial Consortium, 2010. OpenGIS Web Map Tile Service Implementation Standard. http://www.opengeospatial.org/standards/wmts (Last accessed 07/28/2014).

Pence, W.D., Seaman, R., White, R.L., 2009. Lossless astronomical image compression and the effects of noise. Publ. Astron. Soc. Pac. 121, 414–427. 0903.2140.

15 http://ol3js.org/.

16 http://www.sdss3.org.

Pence, W., White, R.L., Greenfield, P., Tody, D., 2000. A FITS image compression proposal. In: Manset, N., Veillet, C., Crabtree, D. (Eds.), Astronomical Data Analysis Software and Systems IX. p. 551.
Pitzalis, D., Pillay, R., Lahanier, C., 2006. A new concept in high resolution Internet image browsing. In: 10th International Conference on Electronic Publishing. Bulgarian Academy of Sciences, Bulgaria, Bansko, Bulgaria, pp. 75–85. 10th International Conference on Electronic Publishing, organised by the Bulgarian Academy of Sciences, Bulgaria, 14–16 June 2006.
Sanderson, S., Albritton, B., Enders, M., 2013. International Image Interoperability Framework. http://lib.stanford.edu/iiif.
Schaaff, A., Boch, T., Fernique, P., Kaestlé, V., 2012. The CDS at the age of multitouch interfaces and mobility. In: Ballester, P., Egret, D., Lorente, N.P.F. (Eds.), Astronomical Data Analysis Software and Systems XXI. p. 443.
Scharff, E.B., Beyer, R.A., Broxton, M., Lundy, M., Fay, J., Turcan, P., Fay, D., Messeri, L., 2011. WorldWide telescope mars, in: Lunar and Planetary Institute Science Conference Abstracts, p. 2337–2338.
Tody, D., Plante, R., Harrison, P., 2011. IVOA recommendation: simple image access specification version 1.0. ArXiv e-prints 1110.0499.
University of Arizona, HiView Website. http://hirise.lpl.arizona.edu/hiview/ (Last accessed 11 April 2013).
Vanhilst, M., 1990. SAOimage. Smithsonian astrophysical observatory. Report for the period through Jan 1990. Bull. Am. Astron. Soc. 935.
Veal, B., Foong, A., 2007. Performance scalability of a multi-core Web server. In: Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems. ACM, New York, NY, USA, pp. 57–66.
W3C, 2011. W3C Working Group Note 12 May 2011. http://www.w3.org/TR/css-2010/ (Last accessed 29/07/2014).
W3C, 2012. HTML5 A vocabulary and associated APIs for HTML and XHTML. http://www.w3.org/TR/html5/ (Last accessed 07/03/2012).
Wells, D.C., Greisen, E.W., Harten, R.H., 1981. FITS—a flexible image transport system. Astron. Astrophys. Suppl. Ser. 44, 363.
Woodring, J., Mniszewski, S., Brislawn, C., DeMarle, D., Ahrens, J., 2011. Revisiting wavelet compression for large-scale climate data using JPEG 2000 and ensuring data precision, in: 2011 IEEE Symposium on Large Data Analysis and Visualization, LDAV, pp. 31–38.