

## Accessing and Visualizing Scientific Spatiotemporal Data

Daniel S. Katz  
Jet Propulsion Laboratory  
California Institute of Technology  
*Daniel.S.Katz@jpl.nasa.gov*

Attila Bergou  
Jet Propulsion Laboratory  
California Institute of Technology  
*attila@alumni.cmu.edu*

G. Bruce Berriman  
NASA Infrared Processing and Analysis Center  
*gbb@ipac.caltech.edu*

Gary L. Block  
Jet Propulsion Laboratory  
California Institute of Technology  
*Gary.L.Block@jpl.nasa.gov*

Jim Collier  
Jet Propulsion Laboratory  
California Institute of Technology  
*James.C.Collier@jpl.nasa.gov*

David W. Curkendall  
Jet Propulsion Laboratory  
California Institute of Technology  
*David.W.Curkendall@jpl.nasa.gov*

John Good  
NASA Infrared Processing and Analysis Center  
*jcg@ipac.caltech.edu*

Laura Husman  
Jet Propulsion Laboratory  
California Institute of Technology  
*Laura.Husman@jpl.nasa.gov*

Joseph C. Jacob  
Jet Propulsion Laboratory  
California Institute of Technology  
*Joseph.C.Jacob@jpl.nasa.gov*

Anastasia Laity  
NASA Infrared Processing and Analysis Center  
*laity@ipac.caltech.edu*

P. Peggy Li  
Jet Propulsion Laboratory  
California Institute of Technology  
*P.P.Li@jpl.nasa.gov*

Craig Miller  
Jet Propulsion Laboratory  
California Institute of Technology  
*Craig.D.Miller@jpl.nasa.gov*

Tom Prince  
Jet Propulsion Laboratory  
California Institute of Technology  
*Thomas.A.Prince@jpl.nasa.gov*

Herb Siegel  
Jet Propulsion Laboratory  
California Institute of Technology  
*Herbert.L.Siegel@jpl.nasa.gov*

Roy Williams  
California Institute of Technology  
*roy@cacr.caltech.edu*

## Abstract

*This paper discusses work done by JPL's Parallel Applications Technologies Group in helping scientists access and visualize very large data sets through the use of multiple computing resources, such as parallel supercomputers, clusters, and grids. These tools do one or more of the following tasks: visualize local data sets for local users, visualize local data sets for remote users, and access and visualize remote data sets. The tools are used for various types of data, including remotely sensed image data, digital elevation models, astronomical surveys, etc. The paper attempts to pull some common elements out of these tools that may be useful for others who have to work with similarly large data sets.*

## 1. Introduction

The Jet Propulsion Laboratory (JPL) is run by the California Institute of Technology for NASA, as the lead NASA center for robotic space exploration. To date, most of the data taken by JPL's spacecraft, as well as other NASA spacecraft, are sent to Earth for processing and analysis. As the spaceborne instruments improve, more and more data is taken and then returned to Earth.

Many common data sets are now each 5-10 TB, and the number of data sets is also increasing. Desktop tools often have difficulty working with data sets of this size. Simply storing the data and accessing it is difficult. Gaining knowledge from data is still hard, but one very effective method is to visualize the data.

This paper discusses work done by JPL's Parallel Applications Technologies (PAT) Group. The PAT Group's goal is to help JPL and Caltech scientists deal with large data sets. The remainder of this paper will discuss a number of tools that the PAT group has developed, for a few particular situations. The first is working with local data sets that the scientists themselves have generated. The second is providing access and information from data developed by the scientist to others located elsewhere. Finally, the third is accessing and working with data developed by other scientists.

## 2. Working with local data

Supercomputing is often defined by comparison with desktop systems, where the exact definition frequently changes. One example is computing that is 2 orders of magnitude more powerful than that available on a desktop. When data is generated on a supercomputer, it is often similarly larger in size than data that could be generated on a desktop. Not coincidentally, data sets of this size are also difficult to deal with in a desktop system.



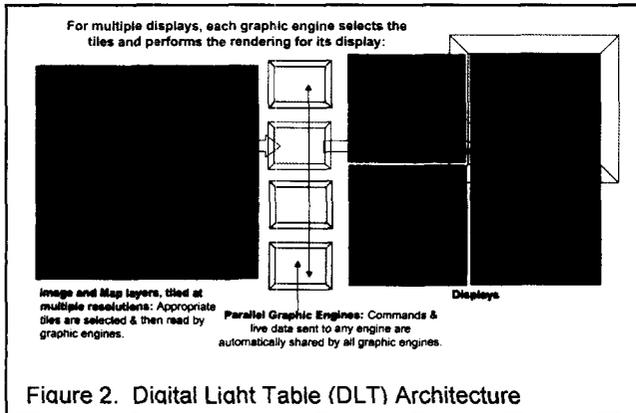
Figure 1. Synthetically-enhanced Martian terrain. (30 m resolution Viking image enhanced to 2 cm resolution.)

A common method for working with such data sets is to leave them on the supercomputer on which they were developed, and to return just the visual data to the user's desktop.

An example data set as shown in figure 1 is synthetically-enhanced Martian terrain data. JPL has collected coarsely-sampled elevation data of over the surface of Mars. When planning future missions, algorithms such as entry, descent, and landing (EDL) and surface navigation must be tested against simulated terrain. The PAT group has developed a terrain server to supply users with such terrain. We have parallelized an application previously developed at JPL, [1] and created a system that allows users to launch jobs and return terrain data generated either by that job or previous jobs. The overall system is called TEDS, the Terrain and Environmental Data Server.

### 2.1. Digital Light Table

Two types of data are often used to represent terrain for a particular geographic area: a digital elevation model (DEM) that gives a height value ( $z$ ) for each location value ( $x,y$ ), and image data that includes one or more values for each pixel. When the pixels are mapped to location values, the two datasets are used together as a terrain model. One tool that has been developed to visualize terrain models is the Digital Light Table (DLT).



The DLT was originally built to allow scientists to explore a SAR (synthetic aperture radar) mosaic of the Amazon basin [2]. This goal led to a number of requirements, including the ability to pan and zoom through the data in real time, along with the ability to scale to large input sizes, over 20 billion pixels, and to scale to large output sizes, over 7 million pixels.

Multiple graphics engines are used to permit scalability of output image sizes, and to assure that the pan and zoom operations operate in real time. Figure 2 shows the architecture of the DLT. The left image shows the input data set broken up into tiles. The middle column shows the graphics engines, and the right image shows the output images (arranged as a single image.) A user interacts with any one graphic engine, which transfers any inputs to all the other graphic engines, using a message passing approach. Other than this, each graphic engine operates independently. Each selects the tiles needed to build its portion of the output image, reads the tiles, and builds the output image. Since the DLT is meant to display terrain data, we must tile the DEM files similarly to the image files. This permits fast panning. When the DLT was built, the only graphics engine fast enough was the Silicon Graphics (SGI) Reality Engine, and thus, the DLT was run on SGIs with multiple graphics pipes.

The use of multiple graphic engines also permits other forms of collaboration. Since one graphic engine can drive one display, we can physically separate the sets of engines and displays. Specifically, we can use the graphics engine and the display in a workstation as one of these units. In other words, scientists in multiple offices can easily work with a single data set. When one takes control and changes the view, the views on all the others change identically.

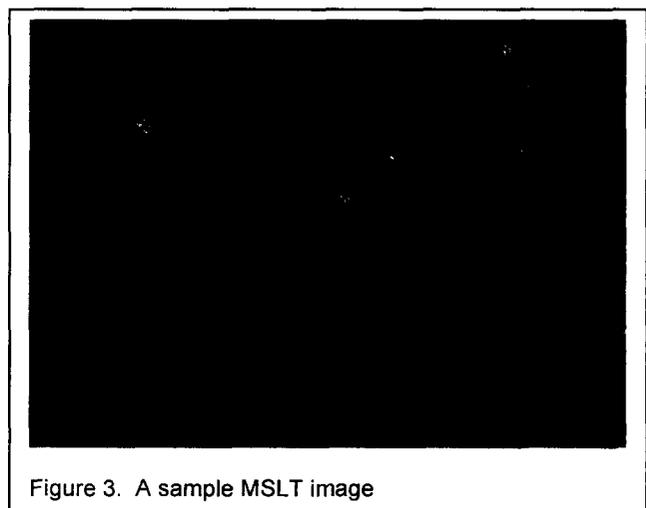
As mentioned, the input image is divided into tiles. For each set of four tiles, a single tile is created with the same number of pixels and half the resolution as one of that set. This process of creating new levels of tiles is continued until a level is obtained where only one tile exists. This scheme, sometimes called an image pyramid [3], adds about 1/3 disk storage overhead, but permits fast

zooming by the graphics hardware because the graphics engines can choose the right level at which to operate, limiting the amount of interpolation needed. The DEM files are pyramided similarly to the image files, for the same reason.

The original data for which the DLT was written included multiple polarizations, and data take in two seasons (wet and dry, about 6 months apart.) For this reason, the DLT permits any band of data to be mapped to any of the three colors (red, green, blue). This allows, for example, for the changes between the two seasons to be clearly viewed while panning and zooming. One can also map bands to each color then flash between the sets as an additional method for visual change detection. Finally, the DLT can also overlay vector data on images.

Recently, a variant of the DLT called the Multi-Surface Light Table (MSLT) has been developed. The MSLT was designed to be used in visualization of data from earthquake simulations. The main modifications from the DLT are the use of both commodity PCs and OpenGL's texture blending capability, which give the MSLT the ability to display multiple surfaces by variable opacity, and linking of image data and metadata.

Figure 3 shows a sample image from the MSLT. One can see terrain, and below the terrain, additional planes. The planes represent fault segments. In this particular image, the color of the segments represents the stress on that segment. Yellow is low stress, and dark blue is high stress. Also shown on the fault segments are arrows that are used to indicate movement along a fault. An additional feature, not shown in the figure 3, is that the user can bring up another window that contains metadata about each of the fault segments. The user can click on one or more fault segments, and they will be highlighted in the image, or the user can select one or more fault segments in the image, and they will be highlighted in the text window.



## 2.2. RIVA

The Remote Interactive Visualization and Analysis System (RIVA) [4, 5] is another tool that can visualize terrain data, but unlike the DLT/MSLT, it is designed with the underlying geometric model of a sphere, and thus can accommodate global (planetary) datasets.

**2.2.1. RIVA Architecture.** Figure 4 shows the RIVA system architecture. Around the core renderer, RIVA is equipped with a suite of Graphic User Interface (GUI) programs for data navigation, display and animation editing. The main RIVA data navigator program, *Flexible Flyer*, resides on an SGI workstation. A low-resolution copy of the dataset is loaded into the Flexible Flyer, where the user can navigate the dataset and select the desired views. As the user navigates, his/her viewpoint is transmitted to the whole earth renderer residing on a remote supercomputer via a network interface program, *NetHost*. The renderer renders the image using a full resolution copy of the dataset and sends the resulting image back to a display window, *receive\_display*, on the user's workstation.

RIVA is designed for both interactive exploration of large datasets and batch generation of animations. The Flexible Flyer has a key frame editor built in where key frames can be inserted, appended, modified, and previewed. A separate 2-D map display window, *xshow*, displays the key frames or the flight path on a 2D map of the data. It helps a user to identify his location and direction in a global orientation and also help a user to select key frames. Once the key frames are selected, the flight path is calculated using a cubic spline algorithm. The renderer then renders the flight path in the batch mode and saves the animation frames into disk.

The middle section of Figure 4 is the network interface programs for RIVA. *NetHost* is the interface

(or, host) program between the GUI and the renderer. RIVA uses a text-based command language to communicate with the renderer. *NetHost* processes and dispatches commands, and receives and distributes results. The commands may come from a network interface, such as a socket stream fed by the GUI program, or from a disk file. The *Router* is a routing daemon that facilitates dynamic, reliable, multi-cast message passing services. It detaches the physical connection between the renderer and the GUI program, thus allowing either part of the program to run as a stand-alone entity, a renderer to feed multiple displays on different workstations, or even one GUI program to control two parallel renderers.

**2.2.2. Parallel Algorithm.** Rather than using geometric objects (such as triangular strips) to represent the digital terrain as all the hardware-based terrain renderers do, RIVA represents and renders the terrain pixel by pixel. It uses a parallel forward projection rendering algorithm with both object space decomposition and image space decomposition. The input data, both image and elevation, are equally divided into small tiles and distributed to each rendering processor. Those processors apply the transformation matrix to their local data and transform them into image space coordinates. Each rendering processor produces patches of images scattered in the final image. The image patches are then merged and composited into the final image using a binary-swap method. The image tiles can be rendered in any order and there is no communication required in the transformation stage. The rendering processors synchronize globally before the final compositing begins. The rendering speed is thus determined by the slowest processor in the transformation stage.

RIVA is careful in storing and accessing data in order to render as efficiently as possible. RIVA stores its data sets in data pyramids, similar to that of the DLT/MSLT, but it generates the data pyramids on-the-fly to save memory at the cost of some computation overhead. RIVA also uses two culling techniques to eliminate input tiles that fall outside the field of view of a given viewpoint. *Horizon Test* calculates the distance from the viewpoint to the center of each tile and eliminates the tiles that fall behind the horizon. This test can eliminate almost half of the tiles in a global dataset. *Tile Test* eliminates the tiles whose projected areas fall outside the viewport.

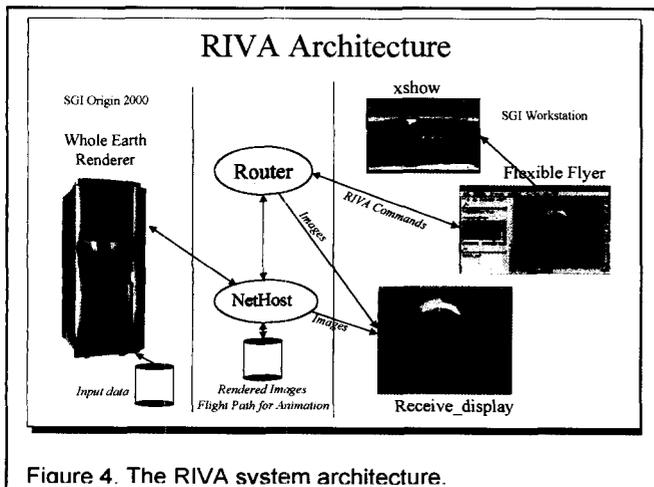


Figure 4. The RIVA system architecture.



Figure 5. Multiple surfaces composited by blending: (a) 30-meter LandSat, (b) two surfaces blended with the 2.25-meter dataset at opacity=0.58. (c) 2.25-meter grayscale dataset.

**2.2.3. RIVA Features.** There are several unique features that distinguish RIVA from other terrain renderers. The first is multiple data representation. Internally, RIVA represents the data using a spherical model regardless of whether it is a global dataset or a regular gridded dataset. Externally, RIVA can process data in either 2D Cartesian, 3D Cartesian, or 3D Polar coordinates. The dataset can be stored in either Sinusoidal projection to save space or in Cylindrical projection for efficient processing. RIVA is flexible and tries to accommodate different application needs and different data representations.

The second feature is Multiple Surface Rendering. RIVA can render multiple terrain surfaces with different resolutions, different data formats, and different coverages, somewhat similarly to the MSLT. RIVA's multiple surfaces can be combined using various blending methods. Figure 5 is an example of alpha-blending of two surfaces. The top surface is a grayscale image of Coronado Island at 2.25-meter resolution (c) and the bottom surface is a color LandSat image at 30-meter

resolution (a). Setting the opacity of the top surface to 0.58 produces a colored image at 2.25-meter resolution (b). Figure 6 is an example of zbuffer compositing. The top surface is a North Atlantic Ocean surface with color representing ocean surface temperature. The bottom surface is a topographic map of the ocean bottom. The top surface is raised up so that the two surfaces can be separated.

The third RIVA feature is Large Dataset Rendering. RIVA allows out-of-core rendering for datasets that exceed the capacity of the physical memory. A lower resolution sample of the original dataset has to be prepared in advance. RIVA loads the lower resolution dataset and renders it until the data pyramiding algorithm identifies that a higher resolution image tile is needed. The full resolution image tile will then be loaded into memory. A memory cache is used to keep the most recent tiles used to reduce disk I/O. RIVA also renders time-varying datasets generated by simulations. Similar to out-of-core rendering, only the data for the first time step reside in the memory. The remaining data will be loaded into memory when the animation starts.

The last RIVA feature is High Resolution Animation. RIVA is not only scalable to large input datasets but also scalable to large image outputs. RIVA images are not limited to the framebuffer size or the screen resolution as with other terrain renderers. RIVA can render a large image in multiple passes by partitioning the images into multiple viewports. Theoretically, there is no limit to the image size in RIVA.

### 3. Sharing Data Visually

A common problem at NASA is that of having a collection of data that has been taken and possibly manipulated in some way, which is intended to be shared with others. At one time, the method for distributing such data was to put it on a tape and mail it to the end user. For small amounts of data, this has now generally been replaced by providing ftp or web access. However, for

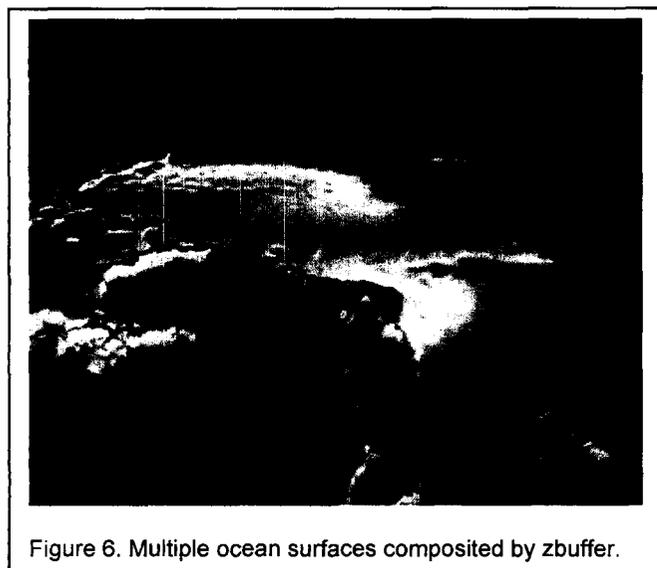


Figure 6. Multiple ocean surfaces composited by zbuffer.

large data sets, sending files on a tape has often been replaced only by sending them on CDs or DVDs. Our intent is to use the Internet for both small and large files, and specifically, to allow users to obtain just the data they desire. We can do this because we can take advantage of both falling disk prices to put all the data on disk for fast access, and we can use parallel computing that's hidden from the user to provide particular data products derived from the data on disk.

MapUS is an example of a project that demonstrates this idea. MapUS is a Landsat mosaic of the continental United States. This 6-band, 150-GB (compressed), 215,000 by 95,000 pixels image with 1-arc-second resolution was built from 428 individual Landsat patches of the Multi-Resolution Land Characteristics (MRLC) dataset, using custom-designed software based on SGI ImageVision framework and run on JPL supercomputers to accomplish this task as a single step operation, with minimal operator input. As has been discussed previously in this paper, we store the mosaic in a tiled, pyramided format. However, the tiles are not stored as separate files, but rather, a new, custom file-format was developed. [6]

We used SGI's ImageVision (IV) Library as a framework for both building the mosaic and building custom images from the mosaic. It supports a large selection of image file formats, but none of the existing ones provided all the features we needed to build and store the mosaic. Since it is easily extendible, a custom file format was designed and implemented. The DLT pyramided data format was used as the starting point, as it provides hierarchical, multispectral and paged image storage. This format was also greatly enhanced to supply additional features. The DLT format specified the order of the pages within the file, the offset at which each page is stored being uniquely determined by a linear combination of resolution level, page corner position within the whole image and spectral channel.

To make this format more flexible, one level of indirection between the page file offset and the page location within the image was introduced. This was done by the introduction of a separate index file, containing a small record for every page. This record specifies the storage size of the page and the offset of the page within the data file. The order of records in the index file is predetermined, the same as the page order in the original image file format. This addition makes possible per-page compression since the size of a page is now flexible, and also adds efficient sparse image storage. We support both JPEG and LZW image compression since both are readily

available and are used extensively. In addition to straightforward versions of these, a few custom variations were attempted, but they are still in the early development phase.

Another useful feature of this image file format is the fact that the image file is consistent as soon as an empty data file and a zero filled index file is created. A record containing a size of zero specifies that the whole page is black, and no additional storage space is required. A newly created file, with a zero size data file and an index file filled with zeros is thus a perfectly valid black image. As pages are written or modified, they are added at the end of the file, and the record in the index file is populated. Thus, the output file can be opened for inspection while it is being created, allowing early detection of potential problems and an easy visual inspection of the generated images. Using this format, the image may be modified by simply appending the new pages of image data to the data file and modifying the corresponding records in the index file. If a copy of the previous index file is made and preserved, simple access to both versions of the file is possible at a minimal storage cost.

MapUS is actually two servers. One is a standard web server that is also a client to the second server. The second uses a standard called the Web Map Service [7] to accept requests from a client (the web server) and deliver the image data back, ultimately to the user. By building this as two servers, we can satisfy two sets of customers: users who want to look at the data through our web server, and users who want to build products based on our data just using our WMS server. Our web server performs the actual image generation using the IV library to take advantage of the parallelism available on our SGI computers. While this allows us to return most results in a few seconds, it limits image generation to the IRIX platform. To ensure portability, the image generation is confined to a separate binary that is executed as a subprocess. Unfortunately, this adds significant overhead as the image must be encoded and then decoded for each pipe between the processes. To overcome this, there is a compile-time option which will include the image generation code in the server, allowing much more efficient operation by avoiding unnecessary image encoding, piping, and decoding. When this option is disabled, the server does not rely on the ImageVision library, which preserves portability, but if the ImageVision library is available, significant speed gains can be realized [8].

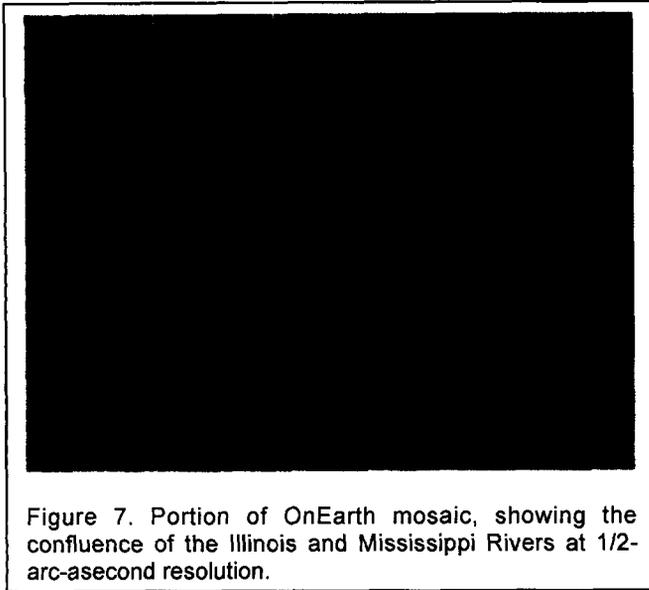


Figure 7. Portion of OnEarth mosaic, showing the confluence of the Illinois and Mississippi Rivers at 1/2-arc-second resolution.

We are also now near the end of development of a new server that will serve a mosaic of the land surface of the Earth at 1/2-arc-second resolution. Figure 7 shows a portion of this mosaic. The mosaic will be about 20 times larger than the MapUS mosaic. Past experience with storage systems together with budget as a consideration led to the selection of a Linux storage cluster. Other projects with similar storage requirements joined the development, with the result being a ten-system Linux storage cluster offering 40TB of storage space. The system, named Raid Again Storage using Commodity Hardware And Linux (RASCHAL), has been operational since April 2003. Each unit is built around a rack-mountable PC case containing one Intel XEON 2.4GHz CPU hosting two separate 3-ware IDE RAID controllers, each controller being configured as a 7+1 RAID 5 disk. The IDE drives used are Maxtor 250GB drives, selected for maximum capacity and minimal power requirements. At the next level, two virtual drives residing in the same case are striped together using the Linux metadrive facility, thus the RAID again name. Dual Gigabit ethernet is standard on every RASHCAL unit, and a 24 port Gigabit Ethernet router provides the communication matrix to the client systems. To make the whole system simpler to maintain, the storage units are configured to boot from a remote host machine.

The OnEarth WMS mosaic project is the first user of this storage system, having, to date, restored more than 7000 of the original Landsat 7 scenes, and produced a few continent size mosaics. This storage system is being used for the mosaic building project and for hosting the data for the WMS site [9].

## 4. Accessing and Visualizing Remote Data

Another general problem is working with data stored in remote archives. Typical problems are understanding what data exists in the archive, what data exists in which file in the archive, and how these files are accessed. Another problem is building visualizations from this remote data. The PAT group has worked on a number of projects related to these problems, but in this paper, we will focus on astronomy applications. In these projects, we attempt to hide the complexity of accessing data from remote archives and building custom visualizations behind a simple interface, which is again that of a web browser.

### 4.1 The yourSky server

The only client software required to use the yourSky custom astronomical image mosaic server is the ubiquitous web browser. By filling out and submitting the form at <http://yourSky.jpl.nasa.gov/>, users have custom access on their desktops to all the publicly released data from the member surveys. In this context, “custom access” refers to new technology that enables on-the-fly astronomical image mosaicking to meet user-specified criteria for: region of the sky to be mosaicked, data set to be used, resolution, coordinate system, projection, data type, and image format [10, 11]. All mosaic requests are fulfilled from the original archive data so that the domain experts maintain control and responsibility for their data and data corruption due to resampling is minimized because only one reprojection is done from the raw input data to the end product. Currently, the data archives that are accessible with yourSky are the Digitized Palomar Observatory Sky Survey (DPOSS) [12] and the Two Micron All Sky Survey (2MASS) [14]. DPOSS has captured the entire northern sky at 1 arc second resolution in three visible wavelengths. 2MASS has captured the entire sky at 1 arc second resolution in three infrared wavelengths. The yourSky architecture supports expansion to include other surveys, without regard to the native image partitioning scheme used by a particular survey.

The architecture for yourSky is illustrated in Figure 8. In the figure, the numbered descriptions on some of the arrows give the steps take to fulfill a typical mosaic request. The procedure is:

1. The clients at the top left of the illustration are the web browsers that may be used to submit requests to yourSky, through use of a simple HTML form interface.
2. The yourSky Mosaic Request Manager polls for mosaic requests in the request queue and hands them off to the Mosaic Request Handler, using a user-priority scheme where the priority is

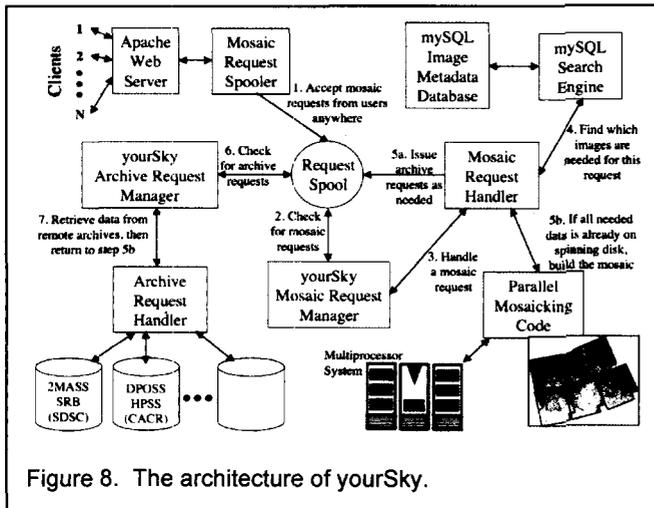


Figure 8. The architecture of yourSky.

- inversely related to the number of pixels generated for the user in the last 24 hours.
3. The Mosaic Request Handler queries the Plate Coverage Database (containing metadata describing the extent of the sky covered by the files in the raw input data) to determine which input image plates from DPOSS or 2MASS are required to fulfill the mosaic request.
  4. A fixed-size data cache is maintained on the yourSky server containing copies of recently-used input files is checked for the needed files.
  5. If all the files exist in the cache, the parallel mosaicking code is started to build the desired mosaic. If all the files are not in the cache, an archive request is inserted in the request queue.
  6. The Archive Request Manager polls for archive requests and hands them off to the Archive Request Handler.
  7. The Archive Request Handler retrieves the images for the archives, using the correct clients for each archive. When all the files have been retrieved, the parallel mosaicking code is started.
  8. When the mosaic is completed, an email is sent to the user with a URL that can be used to access the mosaic.

This process is fairly general. To add new sources of data, two changes need to be made. First, the plate database must be updated with the extents of the plates in the new data source. Second, the mechanism to access the data from the archives must be implemented within the Archive Request Handler. yourSky is one of a number of applications designed to provide access to these data sources. All of these applications need to do these two things. Fortunately, the astronomy and computer science communities, working together as the US National Virtual Observatory (NVO), are converging on standard methods for them called Conesearch and SIAP (simple image access protocol), respectively [15].

Another issue with yourSky is user authentication and accounting. This is something that could be handled within the parallel mosaicking code. This code is currently run on dedicated resources so that the authors of yourSky are really responsible for all yourSky usage. This model is probably appropriate for small mosaicking jobs, in the same way that users of web services are not charged for their use of a web server. However, for large mosaics, it is desirable to be able to tie the processing to a unique user. The Grid community is studying this issue, and some answers are starting to appear. A version of yourSky that is Grid-enabled has been developed, called yourSkyG [16].

## 4.2 Montage

One issue with the current version of yourSky is that, while it operates quickly and thus is good for generating “browse” images, it does not preserve the calibration and astrometric fidelity of the data. Another project, Montage, builds on yourSky through the following improvements:

- Preservation of scientific fidelity in the mosaics
- Support for the “Drizzle” algorithm [17]
- Application of physically based background subtraction models
- Improved performance throughput
- Interoperability with grid infrastructure
- Compliance with NVO architecture

Montage’s architecture is shown in Figure 9. Montage consists of two independent but interoperable components: a background rectification engine, responsible for matching background radiation across images, and a coaddition/reprojection engine, responsible for computing the mosaic. Montage will support all reprojections defined in the World Coordinate System (WCS) [18].

The Montage processing paradigm consists of three main parts: reprojection of images to a common scale/coordinate system; background adjustment of images to a common flux scale and background level; and coaddition of reprojected/background-corrected images into a final mosaic. The background adjustment process involves fitting the differences between overlapping images on a local (for small mosaics) or global scale and determining the parameters for smooth surfaces to be subtracted from each image to bring them to the common scale. These parameters can either be determined on the fly or done once and saved in a database for any future mosaics done with the same images. The advantage of the former is that it allows variations in the fitting algorithms to deal with the special cases and, for small regions, will probably be more sensitive to local variations than a global fit. The advantage of the latter is that it provides a uniform view of the sky and a tested “best fit” that can be

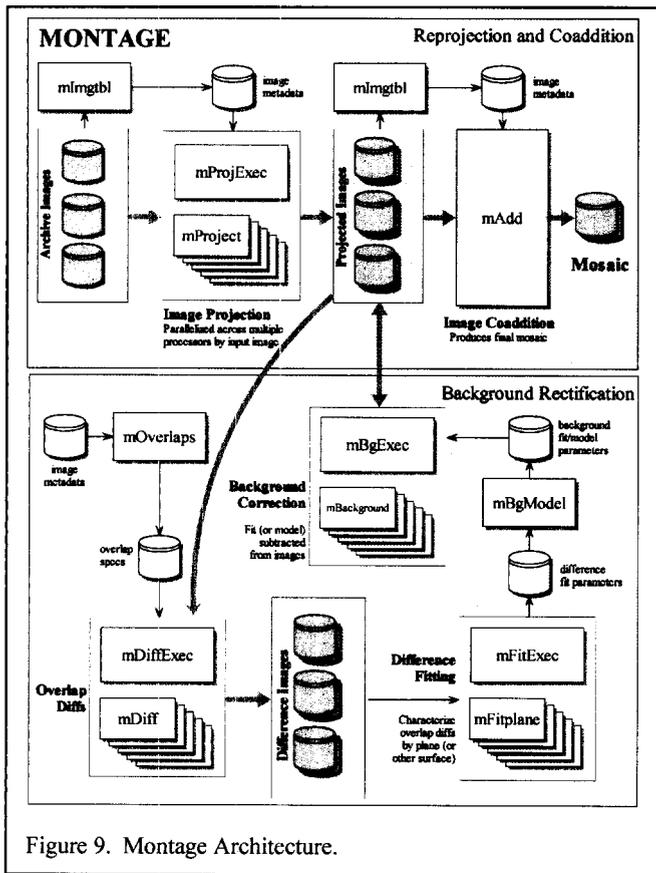


Figure 9. Montage Architecture.

certified as such by the data provider. We plan to use both approaches, deriving and storing in a relational DBMS at least one set of background fit parameters for the full sky for each image collection, but allowing the user the option to invoke custom background processing if they think it will provide a better mosaic for a local region.

The image reprojection within Montage takes up nearly all the run time. This step however, is inherently parallelizable, and can be run on however many processors are available to it. When deployed in 2005, the final version of Montage is required to sustain a throughput of at least 30 square degrees (e.g. thirty 1 degree x 1 degree mosaics, one 5.4 degrees x 5.4 degrees mosaic, etc.) per minute on a 1024 x 400 MHz R12K Processor Origin 3000 or machine equivalent. Currently, Montage has been released as a single processor application, built as a set of executables [19]. The time to run this code on our test system, a PC with a 1.4-GHz IA32-processor running Linux is about 100 seconds per 512x1024 pixel 2MASS image.

We have also demonstrated two prototypes of a parallel Montage. In one case, we scripted the process and data flow, assuming that all processors share a file system. This prototype, called Atlasmaker [20], obtained a speedup of approximately 60 on 64 nodes. The second prototype [21] is more general. It relies on a Montage-

specific web server that builds an abstract DAG (directed acyclic graph, a definition of the data and processing dependencies), a general grid software package called Pegasus that builds a concrete DAG (a DAG that is tied to specific computational resources) from the abstract DAG, and another program called Condor DAGman [22] that then runs the concrete DAG on a collection of grid computers. This prototype is, however, currently much slower and only permits a small speedup.

All three versions of Montage share the use of NVO services to determine which input images need to be used and to access those images. The second prototype can save intermediate files as well as output files using a replica location service (RLS, a Globus component [23]) to allow future mosaics to either be simply retrieved or built more quickly.

## 5. Conclusions

This paper has described a number of existing tools developed by the PAT group to make sense of large amounts of data: Digital Light Table (DLT), MSLT, RIVA, MAPUS, OnEarth, yourSky, and Montage. These tools all were developed in collaboration with scientists. They combine three ideas in three areas: visualizing a scientist's data; allowing others to view images generated from a scientist's data; and accessing and visualizing remote data. Many of these applications try to hide the complexities of accessing remote data from archives and running jobs on supercomputers from the users. These applications have given us a set of tools that we or others can use for new applications.

## 6. Acknowledgments

Most of the work discussed in this paper has been carried out at the Jet Propulsion Laboratory, California Institute of Technology with funding from NASA and the U.S. Air Force. NASA sponsors include the Earth Science Technology Office's (ESTO's) Computational Technologies (CT) project, the Digital Earth project, and the Geospatial Interoperability Office. The Montage project has been jointly carried out with NASA's Infrared Processing and Analysis Center (IPAC), located at Caltech.

## 7. References

- [1] R. W. Gaskell, L. E. Husman, J. B. Collier, and R. L. Chen; "Synthetic Environments for Simulated Missions," *Proceedings of the IEEE Aerospace Conference*, 2001,
- [2] P. Siqueira, S. Hensley, S. Shaffer, L. Hess, G. McGarragh, B. Chapman, and A. Freeman, "A Continental Scale Mosaic of

the Amazon Basin Using JERS-1 SAR," *IEEE Trans. GeoSci. Rem. Sens.*, v. 38(6), pp. 2638-2644, 2000.

[3] L. Williams, "Pyramidal Parametrics," *Computer Graphics*, v. 17(3), pp. 1-11, 1983.

[4] P. P. Li, W. H. Duquette, and D. W. Curkendall, "RIVA: A Versatile Parallel Rendering System for Interactive Scientific Visualization," *IEEE Transactions on Visualization and Computer Graphics*, v. 2(3), pp. 186-201, 1996.

[5] P. P. Li, "Supercomputing Visualization for Earth Science Datasets", *Proceedings of the NASA Earth Science Technology Office (ESTO) Annual Conference*, 2002.

[6] L. Plesea and J. Jacob, "Building Large Scale Mosaics From Landsat Data," *Proceedings of the 8th ACM Symposium on Advances in Geographic Information Systems*, 2000.

[7] J. de La Beaujardière, "Web Map Service Implementation Specification," OpenGIS Consortium Inc., 2002 (<http://www.opengis.org/>).

[8] R. Schreyer and L. Plesea, "OnEarth: Design and Implementation of a Web Map Server," SURF project final report, Caltech, Sept. 2003.

[9] G. Percival and L. Plesea, "Web Map Services (WMS) Global Mosaic," *Proceedings of the 3rd International Symposium on Digital Earth: Information Resources for Global Sustainability*, 2003.

[10] J. C. Jacob, R. J. Brunner, D. Curkendall, S. G. Djorgovski, J. C. Good, L. Husman, G. Kremenek, and A. Mahabal, "yourSky: Rapid Desktop Access to Custom Astronomical Image Mosaics," *Proceedings of SPIE Astronomical Telescopes and Instrumentation: Virtual Observatories Conference*, 2002.

[11] The yourSky Custom Mosaic Server, <http://yourSky.jpl.nasa.gov/>.

[12] S. G. Djorgovski, R. R. Gal, S. C. Odewahn, R. R. de Carvalho, R. Brunner, G. Longo, and R. Scaramella, "The Palomar Digital Sky Survey (DPOSS)," To appear in: *Wide Field Surveys in Cosmology*, eds. S. Colombi and Y. Mellier.

[14] S. G. Kleinmann, "2MASS - The 2 Micron All Sky Survey," *Robotic Telescopes in the 1990s, (ASP Conf. Ser.)*, 203, 1992.

[15] National Virtual Observatory Standards, 2002 (<http://www.us-vo.org/standards.html> /)

[16] J. C. Jacob, J. B. Collier, L. G. Craymer, and D. W. Curkendall, "yourSkyG: Large-Scale Astronomical Image Mosaicking on the Information Power Grid," submitted to *Parallel and Distributed Computing Practices (PDCP), Special Issue on Grid Computing Infrastructure and Applications*, October 2003.

[17] A.S. Fruchter, and R.N. Hook. "Linear Reconstruction of the Hubble Deep Field," <http://www.stsci.edu/~fruchter/dither/drizzle.html>

[18] E.W. Greisen and M. Calabretta, "Representation of Celestial Coordinates In FITS," <http://www.atnf.csiro.au/people/mcalabre/WCS.htm>.

[19] Montage web site: <http://montage.ipac.caltech.edu/>.

[20] R. Williams, L. Brieger, M. Feldman, J. Jacob, G. Kremenek, and R. Moore, "Atlasmaker: Grid-enabled Multiwavelength Imaging," *Proceedings of Astronomical Data Analysis Software & Systems*, 2003.

[21] G. Singh and E. Deelman, "Montage on the Grid," US Virtual Observatory Technical Report, <http://bill.cacr.caltech.edu/cfdocs/usvo-pubs/list.cfm>

[22] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid," *Grid Computing: Making The Global Infrastructure a Reality*, (F. Berman, A. J. G. Hey, G. Fox, editors,) John Wiley, 2003.

[23] A. Chervenak, et. al., "Giggle: A Framework for Constructing Scalable Replica Location Services," *Proceedings of the SC2002 Conference*, 2002.