Object browsing using the Internet Imaging Protocol

Kirk Martinez*, Steve Perry^ and John Cupitt+

* *Electronics and Computer Science Department, University of Southampton, UK.*
^ *ANT Ltd, Cambridge, UK.*
+*Scientific Department, The National Gallery, London, UK.*

Abstract

This paper builds on the results from the Viseum project where we built an image server/client system to allow browsing of very large images. In the follow-on European ACOHIR project we built systems capable of acquiring colour calibrated high resolution views of objects from many positions. A Java viewer allows the user to closely examine objects in a similar way to Quicktime VR but with much higher resolution. The Internet Imaging Protocol is used to allow the viewer to request 64x64 pel tiles on demand to allow fast browsing of the objects in a Web browser. The original image data occupy typically around 200 Mbytes yet we can provide almost instantaneous views with zooming and acceptable performance across the Internet or a modem. The approach taken in the Java viewer is modular and easily customised using JavaScript. Caching at both the server and client provide improved performance. This paper shows how the techniques developed for large images have been applied and modified to handle high resolution object views.

Keywords

object movies, Java, JavaScript, Internet Imaging Protocol

Introduction

Previous research into high resolution imaging [1, 2] has produced systems capable of creating very large colorimetric images of works of art. These could be as large as 1.6 GB each and were colour calibrated. This means that the images can be reproduced accurately and show fine detail. The problem of how to browse these large 2D images over the Web was tackled in the Euro-Canadian project Viseum [3, 4]. This designed a client-server system for displaying the images and investigated the effects of long distance ATM networking. Since then the European project ACOHIR has worked on imaging 3D objects and placing high quality views on the Web. This included making systems with high quality digital cameras and controllable turntables to capture the object views. These are colour calibrated to CIE Lab but stored compressed as sRGB values. The multiresolution tiled JPEG in TIFF format from Viseum is still used and the client/server system has been enhanced to cope with the object data.

The aim of the ACOHIR project was to provide high quality images of 3D objects, in particular sculptures from the Louvre and Greek museums, furniture and porcelain in Spain

and archaeological finds used in teaching. These often have fine details such as scratch marks or engravings. To capture a texture mapped 3D graphics model of an object with a sufficiently high level of detail would require expensive hardware. It also requires a 3D processor at the client in order to display the data. The approach used in Apple's Quicktime VR [5] is to capture images of objects from many views and store them as a movie. For this reason they are often known as object movies. This is played in a special way so that a sequence of 30 frames for example is seen as rotating an object and moving it in the up/down axis is handled by jumping to different 30 frame segments. This approach provides a very fast interface and the images can be quite detailed for each view compared to a rough 3D object. High quality images give a much better idea of surface texture which is very difficault to do with 3D graphics. Companies such as Kadian [6] also produce a wide range of hardware to help capture objects including turntables with controllable camera arms to automate the whole process. Companies such as Live Picture [7] make software to produce and browse object movies and panoramas.

In ACOHIR we wanted to be able to use images of around 3kx2k resolution for each view so that "zooming" into details was possible. This makes sending a full set of images to the client impractical: 36 views of 3kx2k three times would make 216 MBytes of raw data which could be compressed to around 20 MBytes. Transmitting this is impractical on the Internet compared to our solution which initially only transmits around 25kBytes and only ever send the areas the user requires. We decided to adapt the technique used before for high resolution images: to supply images on demand.

Capturing object views

In ACOHIR new turntables were designed by AIDIMA in Spain, the Louvre used a heavy turntable capable of handling very large sculptures while Southampton made a low cost turntable shown in Figure 1 below. Existing hardware can also be used to capture images which can then be colour calibrated using our software and a MacBeth Colorchecker chart [8].

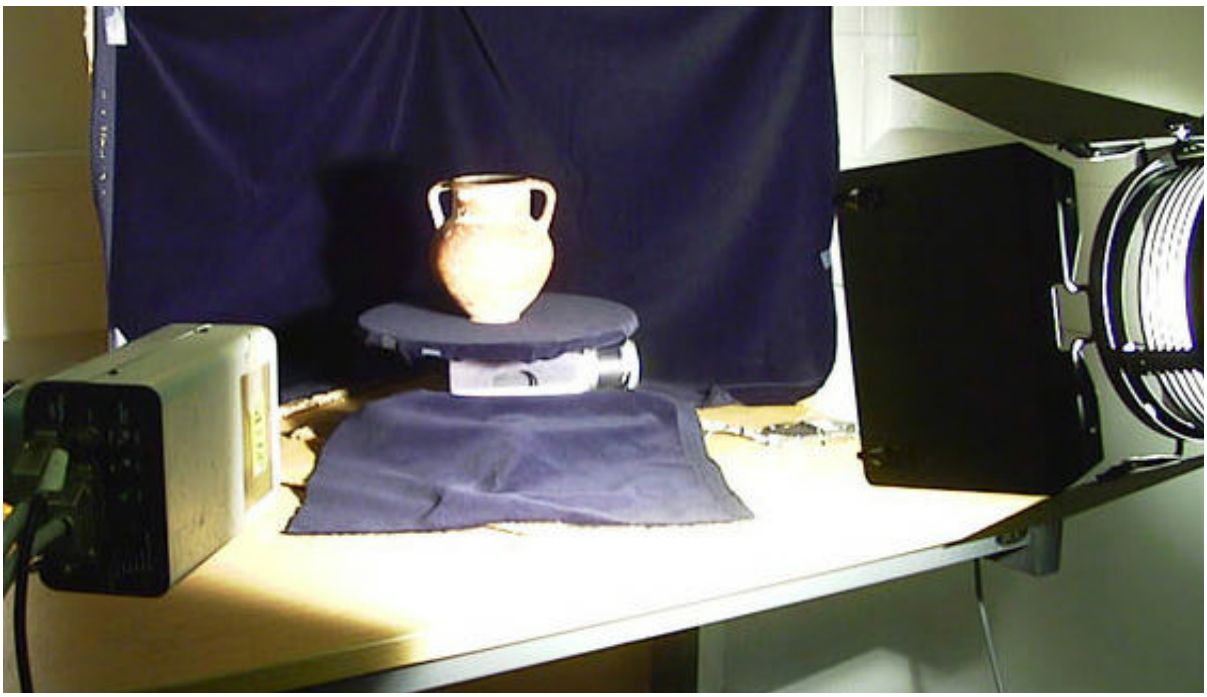


Figure 1. The imaging system in Southampton

We decided to standardise on Kodak professional digital cameras in order to provide an integrated capture package which worked with a range of their cameras. These range from the DCS410 to the DCS560, although the camera shown above is a Kontron camera from the Vasari project. A "consumer" Kodak DC265 has also been tested for a low-cost solution. The colour calibration is described elsewhere [9, 10]. The colour calibration and compression of 10kx10k paintings (aprox. 300MB) was always time consuming and in this project it is still the case: the 216 MByte example cited above is comparable in size.

The use of sRGB

The use of device independent colour has increased considerably, with the general aim of more consistent colour reproduction across devices. sRGB is a fairly new colour space which is used in printers and some cameras as it provides a standard RGB. Unlike CIE colour spaces which we commonly use, sRGB can be displayed well without any computation, which is a great benefit when unknown systems are used.

In the Viseum project we allowed the client to register a colour profile (ICC) with the server so that it could transform CIE colour spaces specifically for the client's display. This allowed the images to be kept in a wider colour space such as CIE Lab but placed a small extra load on the server, as well as adding complexity. This functionality has been kept but now we also allow sRGB images to be stored at the server. This relies on the user making their display conform to sRGB which includes an ambient luminance level of around 64 Lux and a display gamma of 2.2. The server does not need to carry out any computation in this case.

Storage format

For compressed files we generally use a TIFF format file with multiple images for each resolution, each tiled to 64x64 and each tile JPEG compressed. This provides quick access to 64x64 pel tiles at full, half, quarter etc. resolutions, and the JPEG compression is particularly useful for sending images to the Java viewer. Figure 2 illustrates the format with multiple images within the file. Most commercial packages only load the first image and few support JPEG in TIFF although it is supported by the TIFF standard and common libraries. The compression can also be LZW or ZIP as the tiles can be recompressed at the server before transmission to the viewer. Although we sometimes refer to these images as a pyramid there is no inter-resolution compression as used in PhotoCD for example, which is a true pyramidal coding scheme. The major advantage of using TIFF is that it is an open standard.

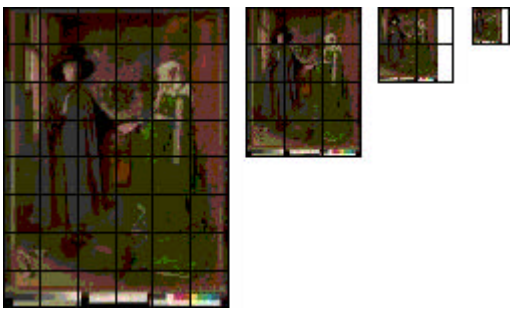


Figure 2 Representation of multiresolution tiled JPEG TIFF file

The Java viewer

The viewer has been written in Java, as it runs on a number of different platforms, and can be embedded in Web pages to provide an easy way to present the acquired objects to the largest possible audience. The applet is highly configurable, and the user interface can be created either as components of the applet itself, or using HTML on the page in which the applet is embedded which communicates with the applet using JavaScript. The viewer reads image tiles from the server and displays them in a window. As the window is typically considerably smaller than the actual size of the image, the viewer facilitates navigation around the image, to different resolutions and to different frames in the image sequence. A cache of tiles is kept at the client to prevent unnecessary requests to the server and to increase performance. When dealing with a multiframed image, this cache may optionally be preloaded with low resolution tiles. Thus, when the user changes the current frame being displayed in that resolution the tile will quickly be retrieved from the cache rather than the server, and so a smooth turning effect can be achieved.

Figure 3 shows the viewer in use for a typical archaeological object with descriptive text. The page consists of two Java applets rather than a new applet containing two image areas with a fixed layout and controls. The small top view has hidden controls and is for turning the object around. The applets search for all compatible applets and communicate with each other to update views. When it is released it sends a position update message to the lower applet which shows the appropriate view. The user controls the viewer in one of four ways - either by clicking in the viewing window, using keyboard shortcuts, activating the controls seen at the bottom of the screen, or by activating JavaScript code which communicates with the viewer. The lower viewer also has its internal controls hidden and JavaScript is used in text links instead. Clicking and dragging with the left mouse button in the viewing window allows the user scroll the viewport around the image, and clicking and dragging with the middle button changes the current frame (Alt is used with single button mice). For a zoomed out view with the tiles for each frame in the cache, dragging the middle button results in a smooth rotation of the object. The frame containing the image views is generated dynamically in the browser using JavaScript, by passing the object name and other details as parameters to the page. This make it possible to easily generate a Web page for viewing any object from a database for example, without having to manually write it. This combination of JavaScript controlling modular applets means the user interface can easily be tailored for different applications or image types without modifying any Java code.

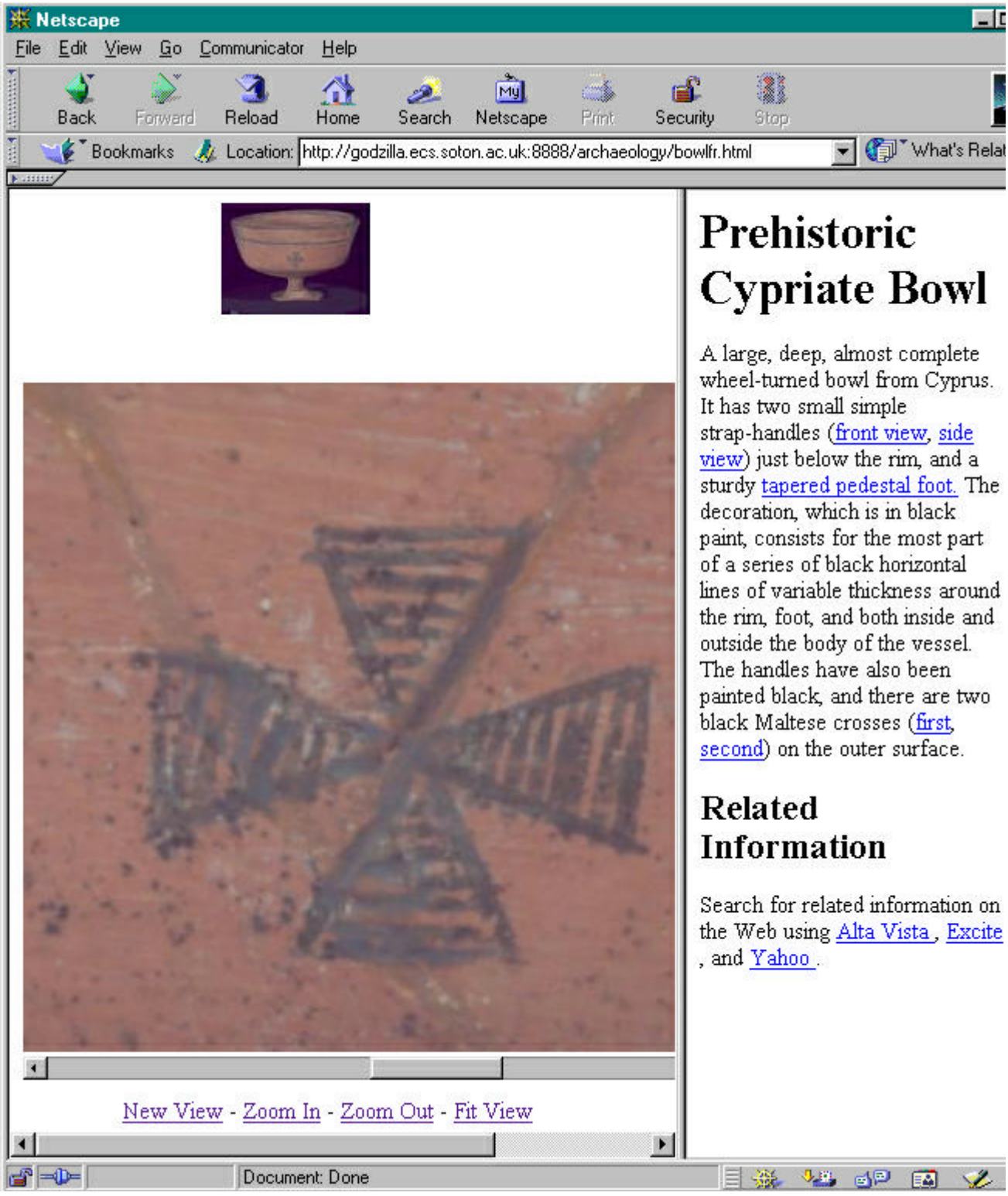


Figure 3. Screendump of viewer

In the text frame on the right of the image, hyperlinks in the text may be used to update the current view. When the link is activated a piece of JavaScript code instructs the viewing applets to change to the appropriate location and frame. This enables the person viewing the document to easily refer to the correct part of the object to which the author is referring. In the example shown, four of the links may be used to zoom in and show details of the surface of the pot. Pages containing applets are typically partially created using JavaScript to enable a number of integration facilities such as personal linking. In this situation the Web page may contain a link or button which invokes JavaScript to query the applet for it's

current location. It can then add this to a list on the page created using HTML forms, entries in which when clicked will update the viewer to the stored location. This can be useful for temporarily `bookmarking' interesting locations within an object which may be returned to later.These bookmarks are stored using cookies, and when the user returns to a page which has been visited previously the bookmarks are reloaded and are again available for selection. In addition to placing links on the page, the author of the document can also define a set of initial bookmarks to be present in the list. This is achieved by creating a special links file for the page which the server is aware of. Whenever a page is requested the server checks for the existence of a links file, and if present encodes the information within into a cookie which is then sent to the browser and presented in the bookmarks list by the JavaScript that is used when creating the page.

IIP server

Our image server uses the Internet Imaging Protocol (IIP) [11] which provides a framework for CGI calls to request parts of images and details about the image such as its resolution. We added extensions to handle directories with object images. Each view of the object is usually stored as a multiresolution JPEG tiled TIFF image, although other or no compression can be used. The client requests get_image_resolutions and then get_num_images to initialise. It can then request JFIF tiles, using JTL requests from the appropriate TIFF file. The server is capable of handling other formats but the efficiency will be very low if it is not tiled or multiresolution. It can decompress tiles from the source file and recompress them to JFIF, which means the user could request a higher compression or other processing. The server is written in C and runs as a permanently resident fastCGI process. It has been tested under Solaris, Linux and Irix. On a SUN e450 the server load is usually under 1% per user. The load on the client always seems higher, with a 450 MHz Pentium II, running NT, being loaded to around 80% decompressing and displaying images.

The other more advanced server is written in Java. It is completely self contained, and acts as a combined Web and IIP server, and is therefore capable of serving not only the high resolution images, but also the HTML documents in which the applets are embedded. Alternatively, the Java server may simply be used as a set of servlets running under any servlet capable Web server. This flexibility provides a convenient single step solution to the problem of serving images in a cross-platform environment.

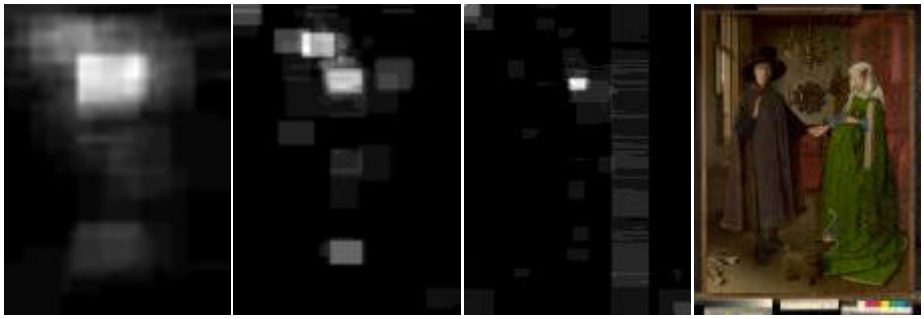
The Java server uses dynamically loaded code to enable a variety of image types to be handled. JPEG images may be served using pure Java code, although this is rather slow due to the lack of tiling in the images, and the fact that decompression, tiling, and recompression must be done in Java. For optimal performance tiled TIFF images are usually used, and the server uses dynamically loaded native libraries for handling the JPEG and TIFF image files. These libraries are themselves extremely portable and run under a number of platforms, including Linux, Solaris, Irix, and Windows 95/98/NT. Although marginally slower than the native C server, in practice the Java server is more advanced for a number of reasons. It is highly cross platform, and will run on any system possessing a Java virtual machine. Additionally, the only native code that needs to be ported are the image handling libraries rather than the whole server itself, as is the case with the C server. The use of Java also provides a convenient mechanism for accessing the underlying threading capabilites of the host platform, thereby giving considerably enhanced multiuser performance when running on a multiprocessor system. One side benefit of the standalone Java server has been that it is easier to upgrade and maintain as Apache does not need to be restarted, as is the case with the C version.

Caching strategies

The effective use of caching has a significant effect on the system's performance. When IIP is tunneled through HTTP each image tile is flagged as not cacheable to prevent them from flooding the browser cache or intermediate proxy servers. A separate cache in the Java client maintains recently used tiles using a simple scoring algorithm. This also reduces the possibility of a potentially slow disc access. The scoring causes the cache to retain low resolution tiles in preference to higher resolution ones, especially because they can be used for the object overview. The table below shows the improvements from increasing cache size and from better cache scoring for a typical user session. Typical usage was recorded and then results were calculated for the same session with different cache strategies.

Cache size (tiles)	Tiles Downloaded - Naive MRU cache	Tiles Downloaded - Cache with improved scoring
200	2270	2130
400	1992	1920
800	1745	1697

Prefetching commonly visited areas while the viewer is quiet can improve performance at the cost of higher network use. The server can maintain statistics of tile usage in each image and provide hints to the client at start-up which are used to prefetch later. Figure 4 shows an image with the most visited areas marked to illustrate this idea, for example in this case most people visit the mirror in the background at high resolution.



Low res Mid res High res Original

Figure 4 showing statistics of image use

An IIP extension allows the viewer to get a list of tiles to preload during quiet periods. This is made easier by the threaded nature of the cache filling code, which can be interrupted to provide better response. The server stores tile usage statistics to generate the cache hints. If the user looks at the low resolution image and the controls for a while before doing anything then most of the popular tiles for an image such as the one in Figure 4 can be preloaded. This has the effect of an instant response if the user follows a typical browsing pattern but places a heavier load on the viewer's caching system and network if they do not. This is more complex for objects due to the large number of images involved but the principle is the same. However the heavy preloading of the small images for the icon applet place a heavy load on the client already, so any further prefetching will have to take place in other quiet periods.

Conclusions and future work

A new way of serving and browsing images of 3D objects has been successfully produced. This is a logical high-quality step up from object movies as used in Quicktime-VR. Any number of views can be taken at any resolution to provide higher detail views of objects than has been available so far on the Web. In practice the system is usable over modems and long distances on the internet. The initial load of the image area only involves around 25 kBytes and on a 56 kbaud modem the first screen fills in around 6 seconds. The Java code load of 50 kB is actually more time consuming but only happens once. On our 100Mbit Lan the pot image show above is loaded with its Java in 10s with the small view completing in a further 7s. The low load on the server is significant as it means the system could cope with many users, which is an important consideration with new types of service such as this. A demonstration server can be found in:
http://www.ecs.soton.ac.uk/~km/

A small tool will eventually be included in the viewer to shift the user's display gamma to closer to 2.2. At the moment we rely on external third party programs. The viewer will continually gain small improvements such as more controls, hyperlinks and painting interim tiles calculated from low resolution tiles in the cache while the replacement high resolution tile is being downloaded. The caching should be user configurable, as not everyone would want their network loaded by prefetches. It will also become more sophisticated, for example prioritising cached tiles from lower resolutions rather than a simple least recently used algorithm. The release of JPEG2000 will provide interesting improvements due to its inherently multiresolution nature (it uses wavelet compression) but probably issues of support in Java once again. When hyperlink areas are included on the images these will optionally come from a linkbase rather than simply coded in the applet's parameters at startup.

Acknowledgements

Thanks to Nick Lamb for his work on the prefetching. The partners of ACOHIR: ATC, Cobax, AIDIMA, Barco, ENST, Lladro, LRMF, Vasari Ltd, The National Gallery, Aristotle University of Thessaloniki, The Christian and Byzantine Museum in Athens, Museum of Cycladic Art. ACOHIR was funded by the European Commission's ESPRIT programme. Thanks also to the IIP community for many helpful discussions.

References

[1] K. Martinez, J. Cupitt, D. Saunders, "High resolution colorimetric imaging of paintings", *Proc. SPIE*, Vol. 1901, Jan 1993, pp 25-36.

[2] J. Cupitt, K. Martinez, and D. Saunders, "A Methodology for Art Reproduction in Colour: the MARC project", *Computers and the History of Art Journal*, vol. 6, No. 2, 1996, pp 1-20.

[3] K. Martinez, J, Cupitt, S. Perry, "High resolution colorimetric image browsing on the Web", Computer Networks and ISDN Systems, 30, pp 399-405, 1998 - online

[4] Viseum project: www.InfoWin.org/ACTS/RUS/PROJECTS/ac238.htm and www.ecs.soton.ac.uk/~km/projs/viseum

[5] Quicktime VR: www.apple.com/quicktime/.

[6] Kadian: www.kadian.com

[7] Live Picture: www.livepicture.com/

[8] Gretag: www.gretagmacbeth.org

[9] Internet Imaging Protocol, Hewlett Packard, Live Picture and Eastman Kodak, 1997 - available from www.digitalimaging.org

[10] D. Saunders, J. Cupitt, R. Pillay and K. Martinez, “Maintaining colour accuracy in images transferred across the Internet”, in Colour Imaging: Vision and Technology, Eds L.W. MacDonald and M.R. Luo, John Wiley, pp 215-231, 1999.

[11] M. Stokes, M. Anderson, S. Chandrasekar, R. Motta, A Standard Default Color Space for the Internet, 1996 - www.color.org/sRGB.html

Vitae

Dr Kirk Martinez gained a BSc in Physics from the University of Reading and a PhD in Image Processing at the University of Essex. He has worked on several European projects such as VASARI, MARC and Viseum. His research interests include high resolution colorimetric imaging, parallel image processing, Web multimedia. He is Director of the Centre for Digital Libraries Research at the University of Southampton. km@ecs.soton.ac.uk.

Dr. Stephen Perry obtained a Bsc. in Computer Science and a PhD. in image and multimedia from the University of Southampton, and subsequently worked on a number of European projects including VISEUM and ACOHIR. He is currently working for ANT Ltd. on their embedded Web browser, Fresco. stephen@ant.co.uk

Dr John Cupitt: since completing his PhD in Theoretical Computer Science at the University of Kent, he has worked in the Scientific Department of the National Gallery London on the European Community-funded VASARI, MARC and VISEUM projects. He has published papers on camera calibration, image processing I/O systems, user-interface design, the measurement of colour change in paintings and infrared imaging of paintings. john.cupitt@ng-london.org.uk