Navigable movies: A Real QuickTime VR

Paul Bourke

WASP, University of Western Australia

Abstract

This paper introduces a navigable movie player based upon the QuickTime API. Navigable does not mean (as in QuickTime VR) that one can only look around within a static image, rather one can navigate within a movie. A key application of this is to present interactive movie content not just on a flat screen but also within immersive displays, in particular, within hemispherical domes. As such the player supports a very general method of mapping the source movie projection geometry onto the geometry of the presentation hardware. The result is an abstraction of the details of the input movie geometry and the output image projection, the exact mapping and navigation mode is contained with a data file rather than within the movie player software itself. A description of the player will be presented along with details of the mapping between source and destination geometries, and finally a number of practical examples.

Introduction

In 1996 Apple added a number of extensions to the QuickTime movie format[1] and corresponding support within the QuickTime movie player, these were called "Panoramic movies" or QuickTime VR (QTVR). This allowed a cylindrical panoramic image to be imbedded within what appeared to be a QuickTime movie, and on opening this movie with the QuickTime Player one was presented with a perspective projection. The use could interactively (in real time) change the view direction and zoom level. Most panoramic images created for QTVR were stitched from multiple camera shots and were therefore usually cylindrical projections, 360 degrees horizontally but with a limited vertical field of view. While QTVR is able to support spherical panoramic images which capture the entire visual field, more recently Apple[2] added cubic map support that solves the issues at the poles and is also easier to create using computer based modelling and rendering software. Although QTVR "movies" appeared as movie icons and were referred to as movies (sometimes even as navigable movies) the name was somewhat of an exaggeration since they consisted of just a single image captured from one position in space and time. QTVR was really an interactive panoramic viewer and although it is possible to click and jump to another discrete position and experience a panoramic view from that new position, it is hardly a movie experience.

Here I present navigable movies in the true sense of the word, that is, one navigates in a similar fashion as in a QTVR movie except that instead of navigating within a single image one is navigating within a movie! Additionally the approach taken is much more general in that while QTVR and similar players present only a perspective view the solution here is general enough that it can readily present other projections such as those required for more immersive display environments. In particular it supports fisheye projections[3] for planetarium domes that employ a projector and fisheye lens as well as the warped fisheye images required for immersive projection using a spherical mirror[4]. QTVR is able to accept three source projection geometries (cylindrical, spherical, and cubic) and present a single output geometry (perspective). The player software introduced here needs to potentially accept any source geometry and create any output projection. The different source image geometries and output projection are not built into the player but are described by an external file that defines the mapping between the input and output image. At the time of writing the player has been used to present output views as perspective, fisheye, and warped fisheye. It has been used to accept source movie geometries including cylindrical panoramic, spherical panoramic, fisheye, and perspective. This last option may seem unusual but, for example, it allows one to zoom and pan around within a very high resolution movie while it is playing (see example 4).

Implementation

The navigable movie player is implemented as a Cocoa application that makes extensive use of the QuickTime API. The basic operation involves opening a movie and a warp mesh file, it is this mesh file that describes the source geometry and how to create the output view projection. Each frame of the movie is considered to be a texture and is mapped onto the mesh. Each node of the mesh consists of a position (x,y) in normalised screen coordinates, a texture coordinate (u,v), and a multiplicative intensity value. The intensity value can be used to correct for variable intensities due to different light paths that arise in some projection environments and can support edge blending[5] for multiple projector arrangements. The warping mesh files are typically created mathematically given an understanding of the geometry or by simulation as in the case of the warped fisheye images required for spherical mirror projection.

The position of each mesh node (x,y) is specified in normalised screen coordinates, that is, horizontally the range is –aspect to aspect, vertically -1 to 1. The aspect is the ratio of the image width to height, aspect ratios in common usage for data projectors are 4:3, 5:4, and 16:9. From an OpenGL programming point of view this would represent a full window display given an orthographic camera as follows:

```
glOrtho(-aspect,aspect,-1,1,near,far);
```

Normalised screen coordinates are convenient since they are independent of resolution, that is, the number of pixels in the display. It does mean that a particular warping map file is explicitly linked to an aspect ratio, this isn't a serious limitation since most warping is aspect ratio dependent for other reasons.

The texture coordinates are defined the same as in most graphical APIs, such as OpenGL. They range from 0 to 1 in both directions (u,v) and are therefore independent of the relative dimensions or the resolution of the texture image. While the warping can be encapsulated in either or both the mesh node positions (x,y) or the texture coordinates (u,v), it is more usual to use a regular mesh and encode the warping within the texture coordinates.

As expected all compression codecs and audio modes supported by the QuickTime API are supported in this player. The movie is opened by drag/drop, the warping mesh file can either be specified in the preferences or by giving it a specific file name and locating it in the same directory as the player application. The only other requirement is the player must be able to operate in full screen mode (no menus or window decoration), this is a requirement for data projector based environments (see figure 3b).

Navigation

Navigation is typically implemented by modification of the (u,v) coordinates of the warping mesh. For example, rotating left and right within a cylindrical panoramic image is simply a matter of adding an offset to the current (u,v) value at each mesh node and ensuring texture coordinate wrapping occurs across the interface at the left and right edge of the image. The source input image geometry and the output projection determine what forms of navigation are appropriate, for example, it is not appropriate to zoom into a hemispherical fisheye projection since the result would no longer be a fisheye image.

Source movie projection	Output projection	
	Perspective	Fisheye or warped fisheye
Perspective (Example 4)	Pan, zoom, roll	-
Fisheye (Example 3)	Rotate, zoom, roll	Roll
Cylindrical (Example 2)	Rotate, zoom	Rotate, roll
Spherical (Example 1)	Rotate, zoom, roll	Rotate, roll

Table 1. Matrix showing navigation modes for each combination of input image projection and output projection.

Example 1: Spherical projection

Figure 1 is an example of a movie where each frame is a spherical projection ranging from 0 to 360 degrees in longitude and 90 (north pole) to -60 degrees in latitude, captured using the LadyBug[6] camera (courtesy iCinema[7], UNSW). Each frame is 4096 pixels wide by 2048 pixel high. Figure 1a is a single frame from this movie, figure 1b is a perspective view presented by the player being described here, figure 1c is a fisheye view intended for projection into an immersive hemispherical dome using a projector with a fisheye lens. Navigation in the fisheye view is (currently) limited to panning left and right, in the perspective view one can pan and zoom.

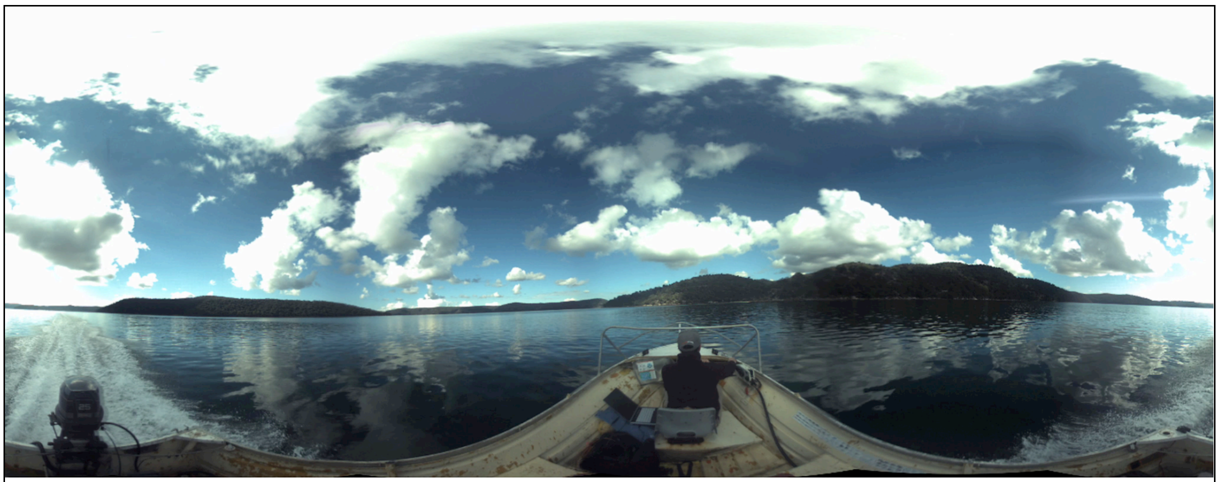


Figure 1a

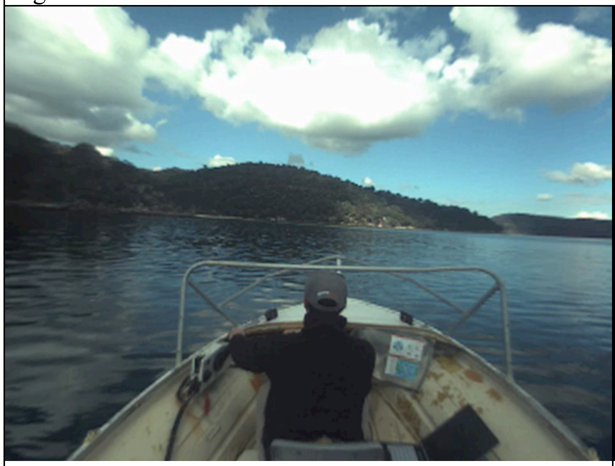


Figure 1b

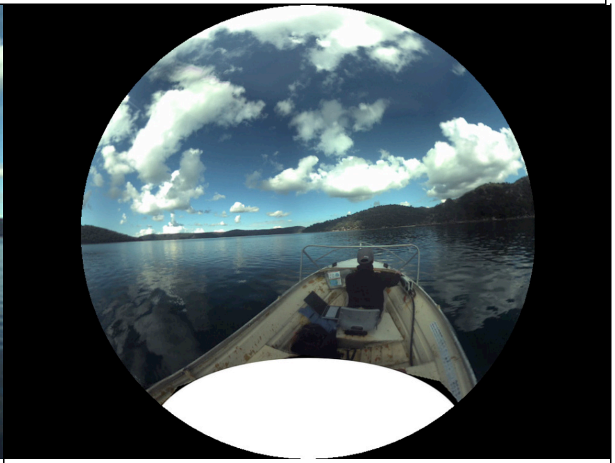
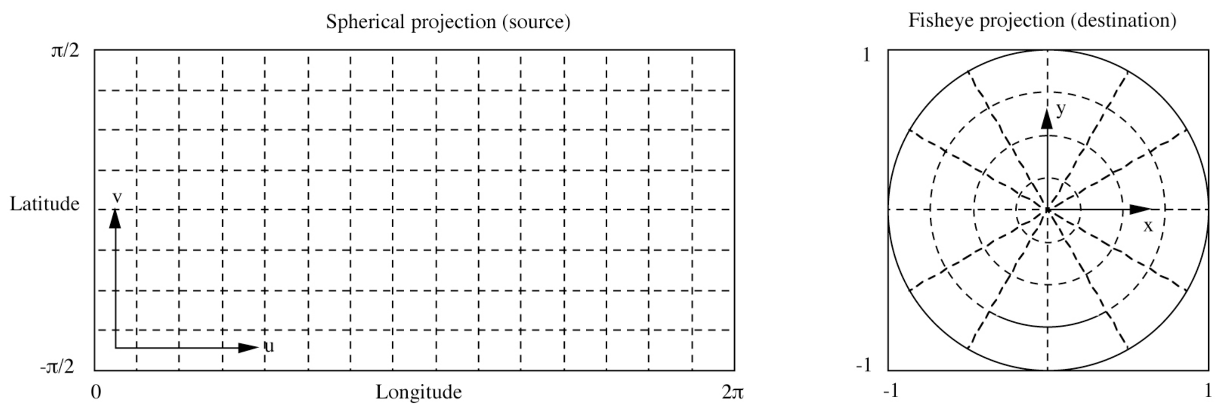


Figure 1c

Figure 1. Example illustrating a spherical projection source movie and a perspective and fisheye output projection.

For each combination of source image geometry and output view projection a warping mesh file is required. The following illustrates the mapping required to warp a spherical panoramic image (figure 1a) to a fisheye (figure 1c), this is one of the more involved mappings.


Spherical projection (source)
π/2
Latitude
v
u
-π/2
0
Longitude
2π
Fisheye projection (destination)
1
y
x
-1
-1
1

A polar grid in normalised screen coordinates (on the intended fisheye view window) can be defined as follows:

$$(x,y) = (r \cos(2\pi i / N_i), r \sin(2\pi i / N_i)$$

where $r = 1 - j / N_j$ and $0 \le i \le N_i$, $0 \le j \le N_j$. N_i and N_j are the number of mesh nodes horizontally and vertically, or more correctly for this polar mesh, the number of lines of radial arms and rings respectively. For a fisheye each (x,y) position on the image plane corresponds to a unit 3D vector into the scene, this vector is

$$P(x,y,z) = (\sin(\psi) \cos(\theta), \cos(\psi), \sin(\psi) \sin(\theta))$$

where $\psi = \pi r / 2$ and $\theta = \text{atan2}(P_y,P_x)$. Each of these 3D vectors corresponds to a position on the spherical panoramic image, normalised to the range (0,1), as such the texture coordinates (u,v) are given by

$$u = (\text{atan2}(P_x,P_y) + \pi) / 2\pi$$

$$v = (\text{atan2}(P_z,\text{sqrt}(P_x^2 + P_y^2)) + \pi/2) / \pi$$

Note the standard maths library function atan2() is used above, it returns angles in the range $[-\pi, \pi]$ hence the scaling for longitude and latitude (u,v) on the expresssion above.

Example 2: Cylindrical projection

Figure 2 shows a single frame from a movie (Place-Hampi[8] © Sarah Kenderdine and Jeffrey Shaw 2006) where each frame is a full 360 degree horizontal cylindrical projection with a resolution 8000x1000 pixels. The vertical field of view in this case is 43 degrees and the mapping for the perspective views has been designed to give the same (maximum) vertical field of view.

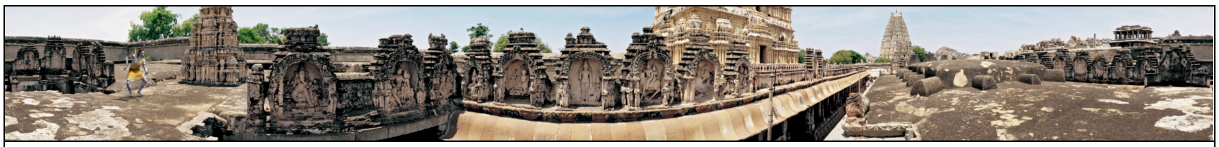


Figure 2a

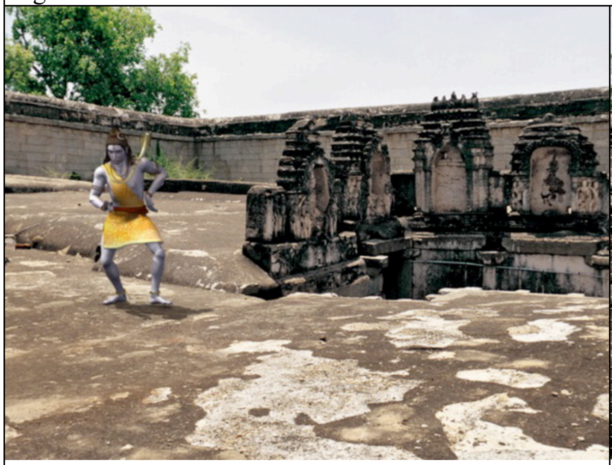


Figure 2b

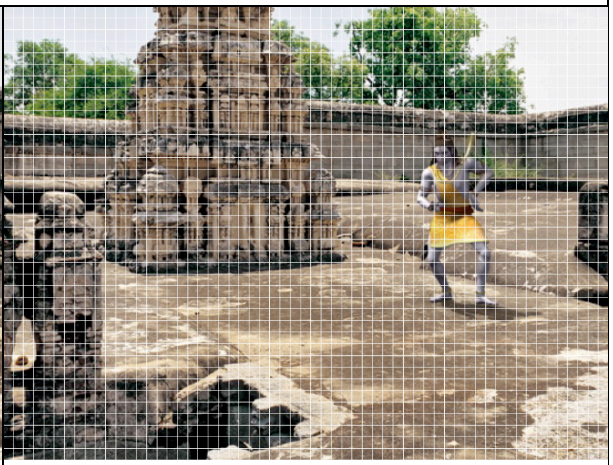


Figure 2c

Figure 2. Example illustrating a source movie where each frame is a cylindrical projection. Two representative output perspective frames are shown, in figure 2c the warping mesh grid (white lines) is also shown.

The mesh grid shown in figure 2c is a 60x40 mesh, typically twice this resolution would be used in order to reduce artefacts arising from the technique OpenGL employs to interpolate texture fragments across a triangle or quad.. Since the mesh is drawn as triangle or quad strips the effect on the performance for modestly high mesh resolutions is negligible. The warping mesh shown here is uniform in (x,y) but it need not be, the image warping itself is achieved with the (u,v) texture coordinate at each mesh node.

The warp mesh that maps a portion of a cylindrical image to a perspective projection is best formed as a regular rectangular mesh rather than a polar mesh as in the previous example. The position of the mesh nodes in the perspective image is then

$$(x,y) = (2\,A\,i\,/\,N_i - A,\ 2\,j\,/\,N_j - 1)$$

where $0 \le i \le N_i$, $0 \le j \le N_j$ and A is the aspect ratio of the perspective view, namely the ratio of the perspective image width to the height. For each node there is an associated 3D vector into the scene

$$P(x,y,z) = (2\,(i - N_i/2)\tan(F_h/2)\,/\,N_i,\ 1,\ 2\,(j - N_j/2)\tan(F_v/2)\,/\,N_j)$$

Where F_h and F_v are the intended horizontal and vertical fields of view of the perspective projection. This vector corresponds to a position on the cylindrical image, normalising that gives the texture coordinates (u,v) as follows

$$u = (\theta + \pi)\,/\,2\pi$$

$$v = (1 + \tan(\psi)\,/\,\tan(F_v/2))\,/\,2$$

where θ = atan2(P_y,P_x) and ψ = atan2(P_z,sqrt($P_x^2 + P_y^2$). If the vertical field of view of the perspective image in intended to match the vertical field of view of the cylindrical panoramic image then they are defined as follows:

$F_v = 2\ \mathrm{atan}(\pi\ C_h / C_w)$

$F_h = 2\ \mathrm{atan}(A\ \tan(F_v/2))$

Where C_h and C_w is the height and width of the cylindrical panoramic image respectively.

Example 3: Fisheye projection

Figure 3 is an example from a cosmological simulation intended for a hemispherical planetarium dome. As the simulation evolves the observer is free to rotate the view. The output view in figure 3c is intended for projection into a planetarium using a spherical mirror where the mirror is located close to the side of the dome. The fading around the rim is to compensate for the higher intensity on that part of the dome close to the mirror and is implemented with the multiplicative intensity value defined at each of the mesh nodes. The warped view in figure 3d is designed for a different hemisphere orientation, namely the upright dome as shown in figure 3b.

While one can write down the equations governing the warping required when using a spherical mirror to scatter the light across a wide angle, the equations cannot be readily solved in a closed form. The warping maps in this case are derived numerically by tracing rays from the data projector, bouncing them off the spherical mirror, and seeing where they strike the final projection surface and thus their (u,v) coordinates on the source fisheye image. This numerical simulation is performed with a precise knowledge of the geometric and optical properties of the hardware components. Any particular installation would conduct this process and derive their warping mesh, content can then be readily shared between the different sites and would look correct and undistorted on each installation.

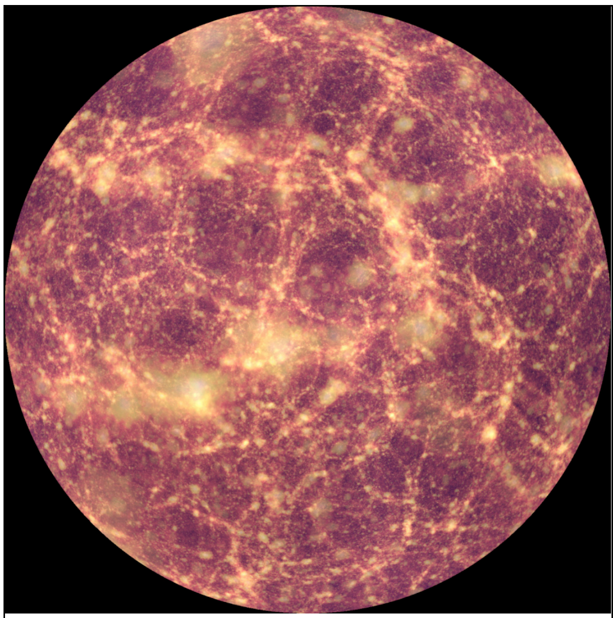


Figure 3a

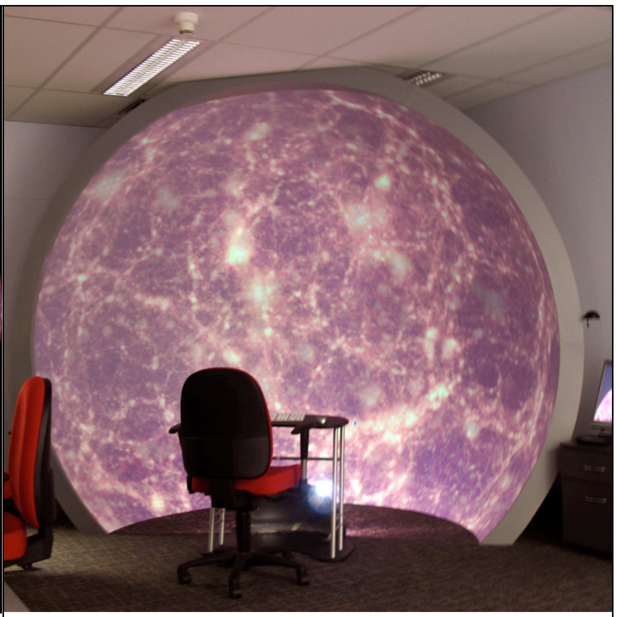


Figure 3b

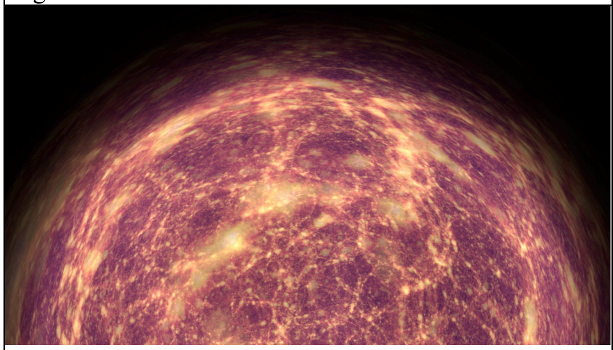


Figure 3c

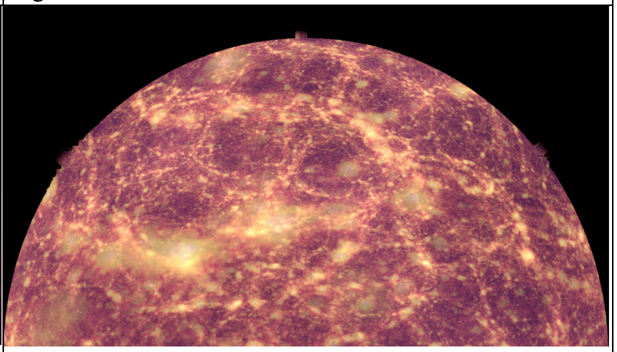


Figure 3d

Figure 3. Sample fisheye frame (3a), warping for conventional planetarium (3c) and warping for upright dome (3d). Figure 3b shows the image in the actual hemispherical environment.

Example 4: Perspective projection

Figure 4 shows a high resolution perspective projection where one can pan around while the movie plays. The dataset is a sufficiently high resolution volumetric dataset from a CAT scan that the isosurface generation cannot be performed in realtime at the resolution required. By rendering the frames at sufficiently high resolution (larger than the resolution of a display) one can zoom in and pan across the image while the isosurface value changes, effectively viewing any isosurface value in detail from any position on a plane.

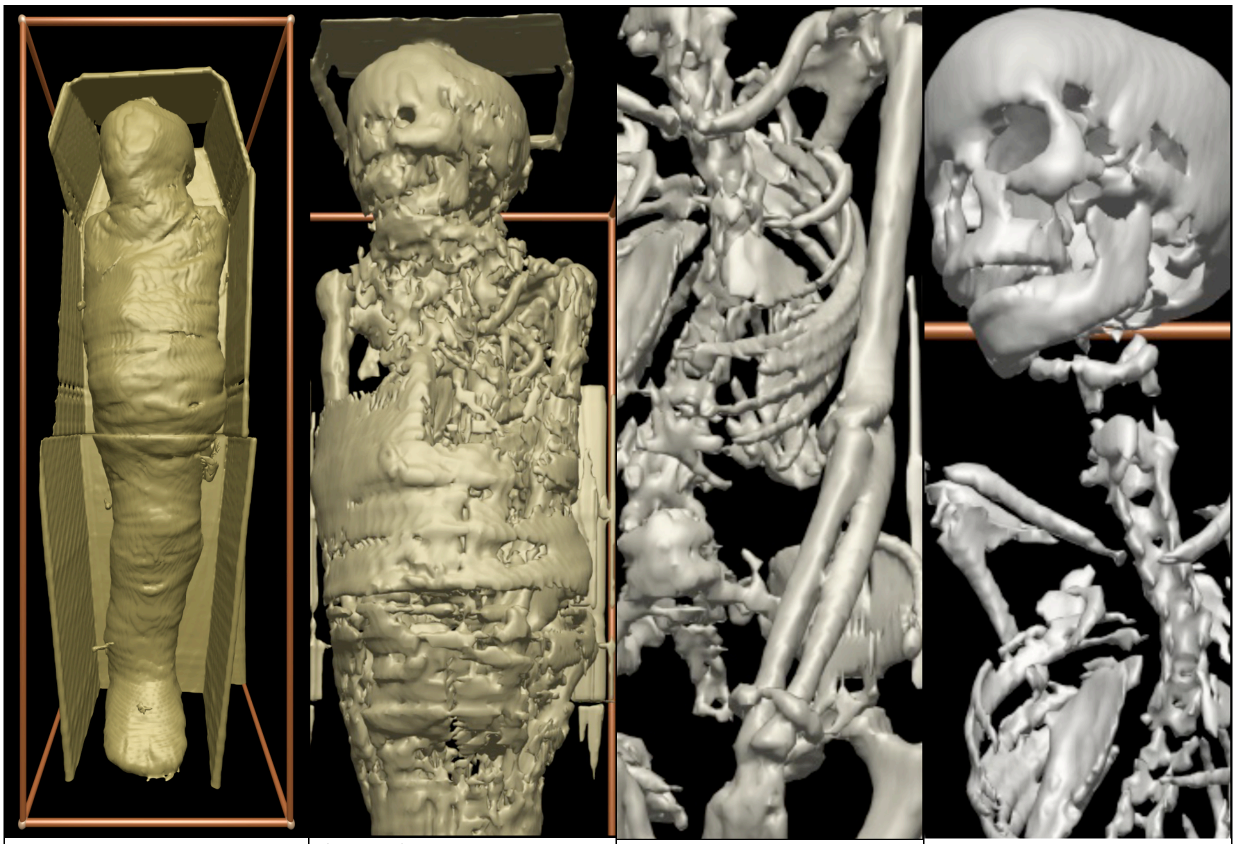


Figure 4a Figure 4b Figure 4c Figure 4d

Figure 4. Large format perspective movie, one frame shown in (4a). Sample zoomed in frames at different times and different positions (4b 50%, 4c 25%, 4d 25%).

Conclusion and future work

A true navigable movie player has been demonstrated that implements a very general image mapping/warping algorithm. Within reason this allows movies made up of frames with an arbitrary image geometry to be mapped onto arbitrary view projections. To date warping map files have been created for cylindrical and spherical panoramic, fisheye, orthographic and perspective projections. A more exotic warping is required for an increasingly popular projection system for planetariums and other immersive environments, namely the use of a low cost spherical mirror instead of a fisheye lens.

Future work includes extending the existing player to support various stereoscopic modes, not just standard perspective stereoscopic pairs but also stereoscopic panoramic[9] and fisheye movies.

Acknowledgement to Ian Hooper for the Cocoa implementation based upon largely imprecise specifications and iCinema for the opportunity to test the ideas demonstrated here on real surround video footage.

Appendix: Mapping file format

The format of the mapping file is relatively straightforward and is a human readable text only file. There is a two line header, first line of which contains a single number that defines the source image/movie projection geometry, a warp map is always defined with respect to a particular input projection. The projection types currently defined are

1: planar/perspective image
2: fisheye projection
3: cylindrical panorama
4: spherical panorama
5: stereoscopic pair (only partially implemented at the time of writing).

The second line of the header contains the horizontal and vertical dimensions (N_i,N_j) of the mesh. The subsequent N_i x N_j lines describe the mesh nodes, each consists of 5 numbers: x,y,u,v,i. They are listed in row order, that is, the nodes for the first row are listed first, then the second row, and so on until the last row of mesh. Note that the mesh need not be a regular rectangular mesh but it does need to be topologically rectangular. A polar mesh can equally be described and is often a more natural mesh format for circular projections.

References

[1] Apple Technical Note 1035, QuickTime VR 1.0 Panorama Movie File Format. Feb 1 1996.

[2] Official QuickTime VR site. http://www.apple.com/quicktime/technologies/qtvr/

[3] Fisheye projection systems: Elumenati. http://www.elumenati.com/

[4] Spherical mirror: A new approach to hemispherical dome projection. Planetarian, Vol 34(4), December 2005, pp 5-9 Paul Bourke

[5] Edge blending using commodity projectors.
http://local.wasp.uwa.edu.au/~pbourke/texture_colour/edgeblend/

[6] Point Grey Research's Spherical Vision camera system. http://www.ptgrey.com/

[7] iCinema, UNISW. http://www.icinema.unsw.edu.au/

[8] Place Hampi. http://icinema.unsw.edu.au/projects/prj_hampi.html

[9] Synthetic stereoscopic panoramic images. P.D. Bourke, Lecture Notes in Computer Science (LNCS), Springer, ISBm 978-3-540-46304-7, Volume 4270/2006, pp 147-155.