VOLUMETRIC VIDEO

Innovations Report 2008

Luke Harris

ABSTRACT

By stacking frames from an image sequence one behind the other, to form a volume of pixels (or voxels), new approaches can be taken in how this data is viewed and manipulated. Viewing is no longer restricted to a 2 dimensional snapshot at specific moments but can mix spatial and temporal information to form a new, sometimes remarkable, image or sequence. Likewise, standard image processes, are no longer considered as ordinary 2D operations, but as 3D operations which act on the whole volume.

INTRODUCTION

There seems to be a lack of widely available tools that provide the user with the power to extract coherent data from video in the way described in this report. Even sophisticated compositing software is largely insufficient in offering this type of treatment, instead preferencing a 'frames over time' representation. This of course makes perfect sense for the vast majority of work performed in editing and compositing packages, however some additional power can be gained by treating the video as a volume, as will be shown here.

From a visual effects perspective, this method solves some problems that are inherent in the conventional format for viewing video. For example, a video recording from a camera panning 360° around a room contains all the information to construct a perfect panoramic image. However obtaining this panoramic image is cumbersome – standard video tools are not designed to make extracting that kind of information easy. This project presents a more generalized approach for dealing with video data.

I use the terms z-axis (z-depth) and t-axis (time) interchangeably to mean successive frames of video.

AIMS & OBJECTIVES

My aim was to produce a visualization of a regular piece of video/image sequence so that it can be seen and manipulated in unconventional ways. The software should have a practical use as well as be able to make aesthetically pleasing images.

The user would need to be able to specify a source clip to perform the operations on. The operations should be easy to understand and responsive to user interaction. There should be the freedom to experiment inside the program in order to achieve new looks. Finally, it should be possible to output the resulting images for other uses.

I would like to use the project to develop my programming and to look at areas such as GUI creation and basic volume rendering techniques. I'd also like to explore some of the possibilities offered by 3d slicing.

RELATED WORK

There have been several examples of related work, ranging from old photographs to experimental films, music videos, software applications or plug-ins and art installations. Some are practical, others are purely aesthetic. I shall describe some of the more influential ones for me. What I find appealing is a visually interesting image having a practical application. I also discovered a small subculture of slit-scan enthusiasts on the internet on photo/video websites like flickr and vimeo. The quality of their images encouraged me to pursue the project.

SLIT SCAN PHOTOGRAPHY

This is a well established photographic technique whereby a camera with a focal plane shutter traverses the film gate relatively slowly with a narrow opening slit. The film is exposed progressively from one side to the other over a period of time. Thus, the left hand side of the image may have recorded the scene from an earlier time than the right hand side. Figure 1 Shows this effect with the shutter moving from top to bottom.

The effect may have been more noticeable with older cameras which were not able to accelerate and decelerate the shutter curtains as rapidly as newer cameras[i]

The same effect can be reproduced with my video cube by simple rotating the slice plane around X or Y.

Seeing these types of images in motion brings a whole new dimension to them which the still photographers were not able to enjoy.

Figure 1. A couple spinning on a turntable taken with a vertically moving shutter. Robert Doisneau

INTERACTIVE VIDEO CUBISM

Slicing a video cube based on an arbitrary slice plane or sphere, not restricted to axis aligned viewing is introduced by Fels and Mase 1999ii. The slicer can be manipulated in real time and operate on streaming video, providing an opportunity to interactively explore the video cube. I wished my implementation to be similar to this. This is the interactive, moving equivalent of slit-scan photography.

" The video data buffer is formed from frames of video data. The virtual cube is the representation of the video data in virtual coordinates. Finally, the cut surface cuts through the virtual video cube which in turn displays the corresponding video data... The main purpose is to explore some of the aesthetics of looking at video data from a variety of perspectives."[iii]

One distinction between our implementations is that they texture-map the surfaces of the cube with the corresponding video images whereas I ray-march through the volume which permits image transparency to produce lofted volumes thus reinforcing the 3D volumetric approach.

COMPOSITING TOOLS

Adobe After Effects and Houdini's compositor Halo (and likely many more) support a technique called "Time Displacement Mapping" out of the box or available through plugins such as GenArts' Time Displace. They work using a black and white map the same size as the video frame whereby black looks up the pixel at frame `t` and white looks up at `t+maxdisplacement`. Often they are used with a black/white gradient which can be recreated in my program by using a diagonal slice, however they also offer the freedom of using arbitrary values (for example a noisy/wavy pattern) in the lookup map which cannot be done with the bilinear patch I use. The limitation with this technique is it is really only remapping the t coordinate, leaving x and y the same. Taking a ZY slice for example would not be possible. Figure 2 (top) shows the distortion resulting from this time-remapping.

A Shake macro TxTransform[iv] written by Jean First exsits that will transpose the X or Y axis with T. The resulting image can be of use to compositors as it allows their filters to be time sensitive etc. A good example is temporal median filtering[v] which can work as a grain suppression technique.

Figure 2 After Effects 2 second time displacement using black/white grad[vi] (below) TxTransform

Tx-Transform is also the name of a film and software by Martin Reinhart[vii]. It simply swaps the t axis with either the x or y axes and has been used in several of his films since 1992. However I do not believe the software is publically available.

STYLIZED VIDEO CUBES

Perhaps the most interesting demonstration of video cubes has been undertaken by Klein et al[viii] 2002. By creating 'rendering solids' (see Figure 3), which exist over time as well as spatially then interesting NPR effects can be made that do not suffer from undesirable temporal aliasing artifacts evident in individual (per-frame) processing. This is achieved by first performing an optical flow on the video to determine the directions that all the pixels are moving in. This provides greater temporal resolution than the 30fps that the original sequences were captured in. The rendering solids can take the form of 3 dimensional shards, or 'worms' which grow along curves along the t axis or importance based KD-tree. Each can be manipulated interactively by the user e.g. offset in time.

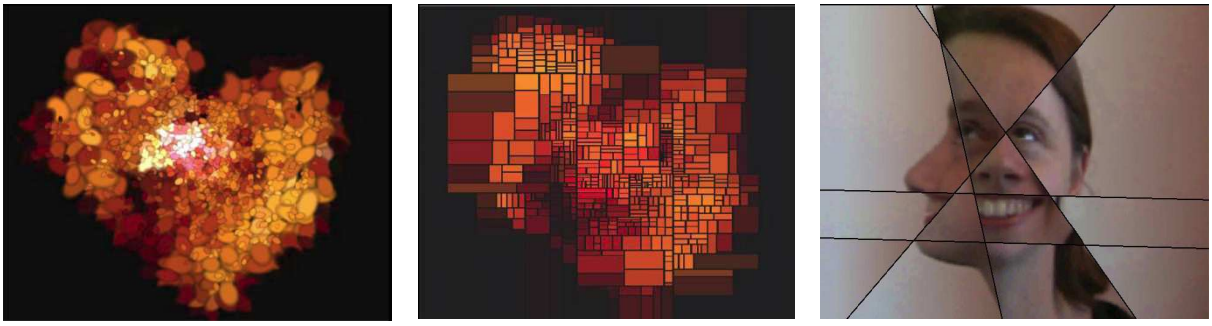


Figure 3 3D Rendering solids used to create temporally coherent NPR effects. Worm, KD-Tree and Shard methods respectively[ix]

LIGHT FIELD PHOTOGRAPHY

Shortly before the end of the project I discovered an expanding area in computer graphics and photography – light field photography. By utilizing techniques such as placing a microlens array behind the main lens of a camera (a plenoptic camera) it's possible to capture more of the light ray information present in the scene.

"Plenoptic Video Geometry is the study of the space of light rays as observed by a moving imaging sensor. The space of light rays is the most complete representation of visual information possible."[x]

Photographic depth of field and exposure become decoupled and can be changed after the fact by selecting the appropriate pixels from the plenoptic image[xi]. Similarly, small differences in parallax caused by the many spatially offset lenses can be used to construct 3D data of the scene. A related project, The Stanford CityBlock Project involved taking video of a street from a truck moving perpendicularly to the camera. The video can be stitched together to form a single orthographic (or multi-perspective) image of the street. This could be straightforwardly reproduced with the software.

DYNAMIC MOSAICS

Rav-Acha et al. present the idea of Time Fronts, and the creation of dynamic panoramas from panned video footage, provided the footage is comprised of primarily vertical motion e.g. a waterfall.

"In this time flow pattern [Figure 4], the initial time front is passing through the right side of each input frame, where regions are captured as they first enter the camera's field of view. Thus, the first time slice is a panoramic image capturing the earliest appearance of each point in the scene. The final time front is passing through the left side of each frame, where regions are captured just before leaving the field of view. This time slice shows the last appearance of each point in the scene."[xii]

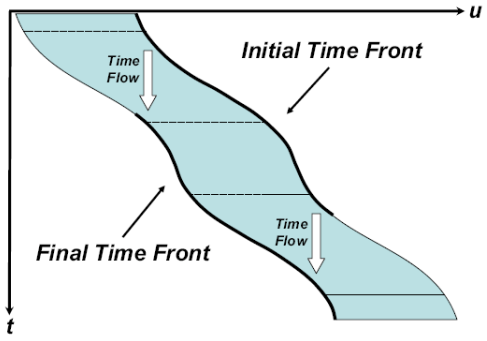


Figure 4 This time flow pattern assumes a camera panning from left to right and will generate a dynamic panoramic video of the scene captured between the dark black lines. These are the time fronts (left and right edges of the frame in this case). Diagram: A. Rav-Acha

PRODUCTION

I had been very impressed with some of the work that had been developed in Processing[xiii] and had initially decided to develop in that. It was a good experience in getting to grips with 3D data structures and cameras. However as the project developed some of my ideas would have been difficult to implement properly (ie a user interface) so I decided to switch back C++. Figure 5 and Figure 6 show some 3d particles rendered in OpenGL using Processing.

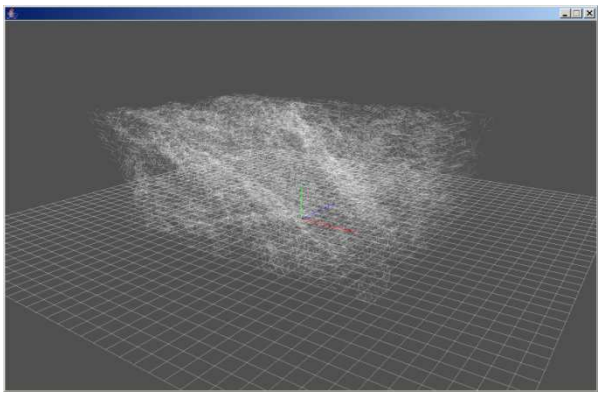


Figure 5. A noisy volumetric cube written in Processing

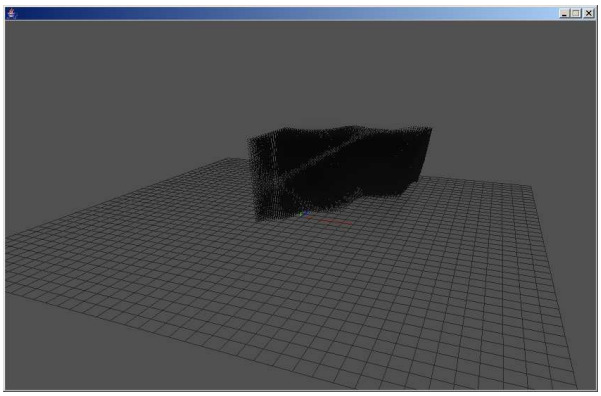


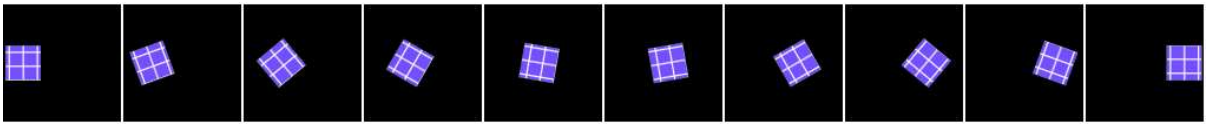
Figure 6. fBm volume noise slice in Processing

Ian Stephenson had developed the NCCAPixmap library which is a cross platform GUI and image handling library which I was going to use and extend to handle voxmaps. In the end however, I eventually settled on wxWidgets[xiv], another cross platform windowing library due to its extensive documentation and native image handling format, wxImage.

Figure 7. Slice 5 of a 10 voxel deep 3D blurred animation

Figure 7 shows my first proof of concept image using wxWidgets. At this point the user could load in a sequence of images which would get stored in a voxel map and apply a simple box blurring effect. It is a 3D blur, so simulates motion blurring as well as spatial blurring. It is based on a 2D algorithm by Jim Scott[xv] which I expanded into 3D. The idea of 3d image processes was something I wanted to enforce as much as possible in an attempt to avoid treating the data as frames.

This algorithm was later rewritten to use 3D convolution kernels, which were more difficult to create but allow the implementation of other filter effects e.g. find edges more easily.



I wanted two viewports in the interface. One is a 3d viewport which shows the block of voxels in space. The other viewport shows the extracted data by the intersecting screen (the slicer). In between are the controls for positioning the slicer.

The image data is stored as a simple one dimensional array of voxels which is accessed using accessor functions in the form `getVoxel(x, y, z)`. I think storing them in this format can sometimes be unstable. There were unexpected crashes when loading long or high resolution images. A voxel is four unsigned chars (0-255) for red, green, blue and alpha. I was tempted to store them as floats instead to support higher bit-depth images and higher quality filtering operations however due to its tight integration with the wxImage object used extensively in the program (which uses unsigned chars) I followed suit.

To position the voxmap in space a world space position vector is subtracted in the `getVoxel()` and `setVoxel()` methods to convert them to object space lookups. See the Appendix Figure 18 for an

illustration of how the items are arranged. A matrix is used to position the camera to allow rotation around the camera's pivot. The pivot itself is set to the centre of the voxmap. Another matrix is used to position, orientate and scale the slicer. The slicer is a bilinear patch[xvi] defined by 4 corners. The patch is surface evaluated (given 2d coordinates – u and v) based on the image resolution, and the returned 3d coordinates are looked up in the voxmap, then the resulting colour from that is written to a framebuffer (a wxImage).

```
pixel = voxelmap->getVoxel( slicer->SrfEval( u, v ) );
```

The four slicer vertices are positioned as offsets from the slicer pivot. This allows them to be transformed in object space to produce non-planar slices, although there is no user interface for this control. For an intuitive way of moving the slicer the matrix operations are multiplied in this order: Rotation*Scale*Translation. I found that other combinations can lead to unexpected behavior from the slice such as shearing. Figure 8 shows a non-axially aligned slice (in green) intersecting the volume, and the image that that slice results in. Moving the green slice is reasonably responsive because it is implemented as a separate layer, so there is no need to remarch the voxels.

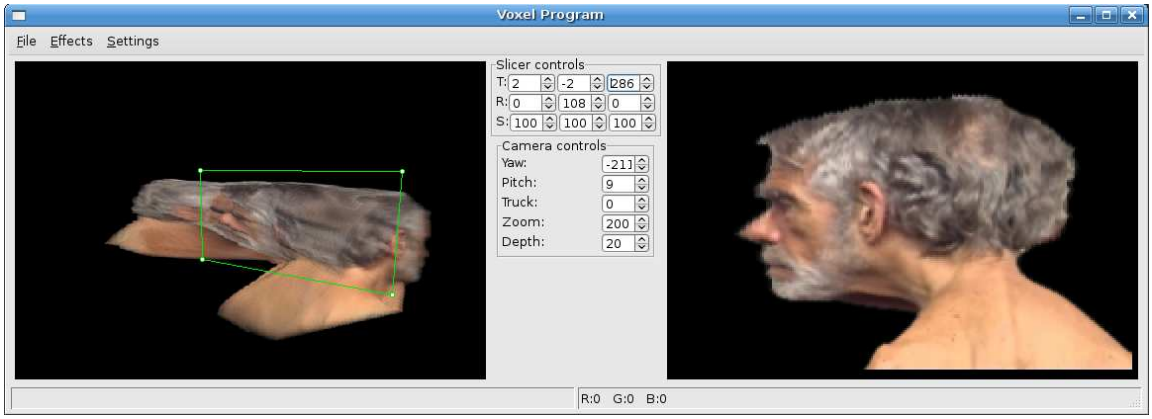


Figure 8 The 3D ray-marcher (left) showing the keyed video in volume form. The green slice plane is overlaid. (Right) The cubism like result of that slice. Video source: The Human Motion Show

RESULTS

FEATURES

- Interactive GUI
- Interactive 3D viewport displaying the video as voxel data with wireframe slicer overlaid
- 2D viewport displaying the result of the current slice
- User controllable quality settings include anti-aliasing, ray marcher resolution, ray step length, sample influence (depth), viewport scaling.
- Two proof of concept 3D convolution kernels
- Controllable colour keyer as a visualization aid
- Sequence image loading in variety of formats (TIFF, BMP, JPEG, PNG)
- Image saving
- Batch mode for multiple image saving with animated parameters

- Progress bars on most time consuming operations
- Cross platform (tested on Linux and Windows)

KNOWN BUGS

- Pitching in the 3D view can produce a divergence of the video cube and the overlaid slice plane
- Windows does not recognize user text input into the spinners as an event, they must be clicked
- With lower resolution clips, the anti-aliasing option can produce artifacts in one half of the slice plane
- The rendering progress bar crashes on Linux, it has to be compiled without it
- On Windows, the last imported frame is copied to the front

IMAGES

Please see the videos for animated versions of some of these images

Figure 9 Temporal aliasing artifacts appear on quickly moving objects. A 10° slice around Y. Source: BBC Motion Gallery

As discussed above, Klein used optical flow to improve the results for the NPR. Some utilization of this would have been nice in my case as well – the time axis is too low resolution compared to spatial. Figure 9 shows this quite clearly, where tearing occurs around edges as the slicer goes from one frame to the next. It is especially evident on higher resolution frames. A PAL resolution 300 frame video (720*576*300) is much less cube like than a half resolution one (360*288*300) which is what I did most of my tests with.

My oversight was I assumed the cube would scale uniformly to full resolution PAL, which of course it didn't as the temporal resolution was the same on both. Thus, slicer numbers that produce one result with one resolution will produce a different result at a different resolution (provided the temporal resolution remains the same).

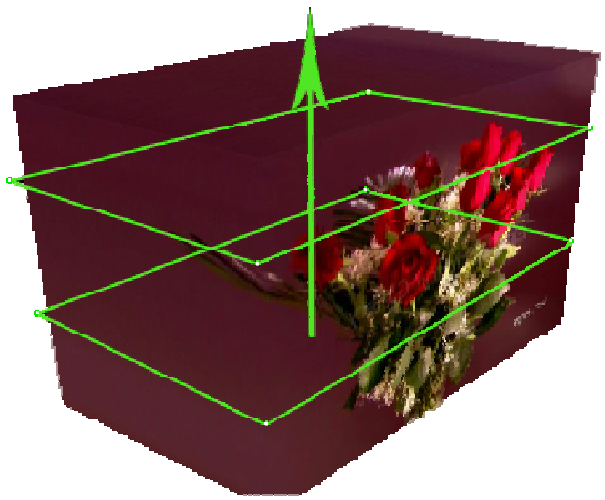


Figure 10 Shows the ray marched cube and two slice planes (green). Figure 11 shows the results of these slices. The attached video is an animation of the slice plane moving in the direction of the arrow. It is interesting to see that there is still a sense of organic structure in the sliced video, despite its abstract form. Interestingly, by following the same process, the original video can be reconstructed from the sliced video as all the data is in there, just reordered. The software is of course unbiased as to which axis is which.

Figure 10 The 3d volume of the flowers video showing 2 slices. Source: BBC Motion Gallery

Figure 11 Rotating vase of flower cut 90° around X, Slice translating along Y.

Creating unwrapped images for texture projection in computer graphics still presents a problem for artists. In Figure 12 the software is used to create a turntable image which could be applied as a cylindrical map to a 3D model. This avoids the long image editing process of assembling photographs from multiple angles. Reflective objects, such as vases and soft drink cans would be largely unsuitable for conventional multiple angle photographing, and would create a higher quality image with turntable slicing as the reflection would at least be consistent across the object.

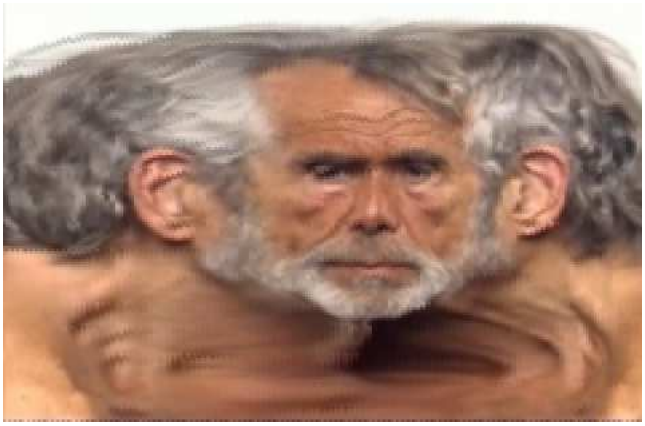


Figure 12 A Cylindrical map is generated by placing the subject on a turntable. Source: The Human Motion Show

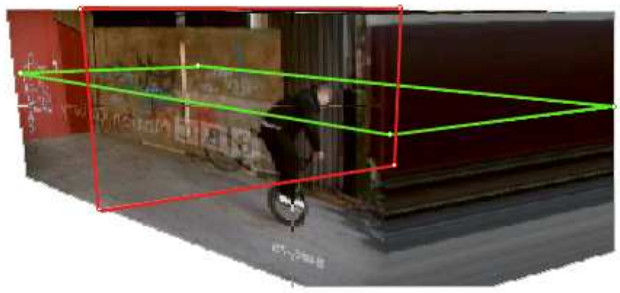


Figure 13 Traversing along Y (green) and X (red). Source: BBC Motion Gallery

Moving the slice plane along X and Y axes as opposed to T is demonstrated again in Figure 13, resulting in slit-scan images like those below. Figure 14. Elements that are not in motion are stretched out across the frame, disappearing. These images show only movement of a certain row or column of pixels at all times in the video. A BMX flatlander (Keelan Phillips) clip was chosen to demonstrate the effects of vertical, horizontal and rotational movement.

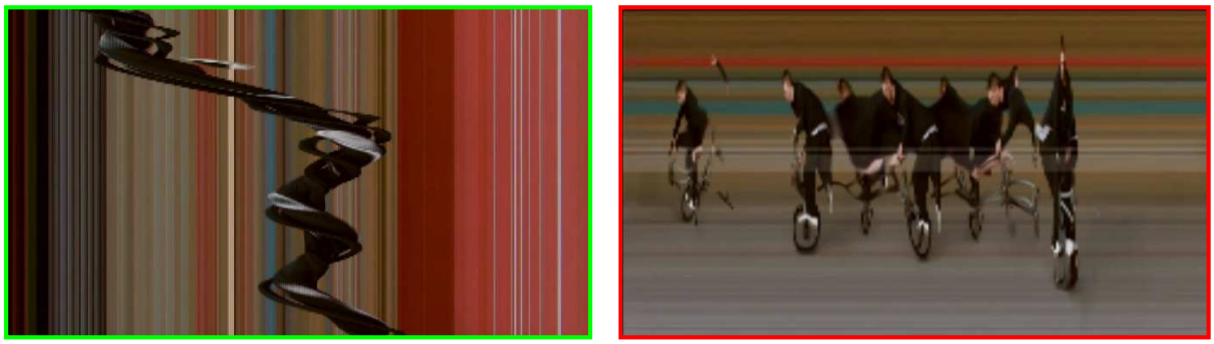


Figure 14 (Left) An XZ slice. (Right) A YZ slice

Figure 15 360° Panoramic image generated from panning video footage. Dynamic panoramas as described previously are also possible, by moving the slice diagonally in an XZ direction. This is very easy to visualize by examining the 3D view. See Appendix Figure 17.

HDRI PANORAMAS

Producing HDR images is often time consuming and requires specialized equipment. Here I describe a possible method for quickly producing low distortion HDRI panoramas with only a neutral density filter.

HDRI images are commonly captured for use in computer graphics by taking bracketed exposures of a mirrored ball, usually from more than one angle or by using a wide angle lens on a tripod and taking multiple exposures in different directions. Mirrored balls result in fewer pictures necessary but can cause severe distortion whereas the second method produces a lot of images and requires a complex stitching process to produce undistorted HDR images.

To generate my panoramas I used a 90° slice around Y. This tells the software to pick a vertical column of pixels from each frame and place them next to each other in the output image. The user can choose which column to use (by moving the rotated slice in X) and the panorama will look pretty much the same. Figure 15 shows one of these possible panoramas. The amount of unique (but redundant) panoramas that can be created this way

is equal to the horizontal resolution of the video – 720 in standard PAL. By utilizing a gradated neutral density filter (ND) of the required attenuation factor, oriented horizontally, then each vertical column of pixels will capture the scene at a unique exposure. All of these individually exposed panoramic images can be captured in one pan of the camera and have very good registration with each other, provided the user panned the camera at any constant speed. The panning speed merely defines the horizontal resolution of the panorama as the slice can be cut at any angle.

KEYING POTENTIAL

While testing the 3D ray marcher, I imported some video footage that I had keyed in shake. This displayed a sweeping volume of the subject in the 3D view. Interestingly it presented a number of flaws in the key that I had not noticed while in the compositor. Spill was evident on the surface of the object – the surface of the object represents the important matte edge achieved with the keyer. To fix these issues with a conventional compositor would be very laborious as it involves dealing with very fine details only 1 pixel wide. If there had been paint tools in the ray marcher these problematic areas would have been trivial to fix, and it would not suffer from jittery frame-to-frame incoherencies that manual rotoscoping can.

ANALYSIS

OVERALL

I'm reasonably happy with the final program. I particularly like when I'm surprised by the unusual slices it shows me when new, unremarkable source clips are loaded in. The rotating flowers slice was my favourite example of this. It would have been nice to test it with a wider variety of footage, such as timelapses, rack-focusing, aperture changes, tracking shots as I'm sure there are some unexpected results to be found. I especially would have liked to test the HDR panoramic idea and the Stanford CityBlock orthographic image reproduction to include in the report. High frame rate video would also have been useful as, at only 25fps, that axis seems to be the worst for aliasing artifacts unless the movement is very slow.

What I found particularly exciting was the extent of research and development into related areas. It made me realize that recording a scene using a standard camera can, at times, be quite limiting and there may be better ways of sampling data, or of revealing the data you didn't know you had.

PROGRAMMING

Programming with wxWidgets was an incredibly steep learning curve because I knew very little about interfaces and C++ OOP in general. Parts were immensely frustrating, but other things I got almost for free, like the loading bars which I think is a nice touch. Although I'm still a relatively weak programmer it has given me the confidence to undertake programming challenges which I would never even consider before. I understand to a greater degree the importance of modularity and consistency. Parts of my program are a bit inconsistent, for example the method to transform the slicer is implemented differently to the one which transforms the camera. The slicer transformations are implemented as a member functions in the slicer class, whereas the camera's are in the GUI class rather than in the camera class. It was satisfying to implement some of the things we had covered in maths over the 3 years (vectors, matrices, interpolation, coordinate systems and surfaces) and have them do something useful. I was particularly proud of the ray marcher which I had been quite worried about but actually it was relatively easy and demonstrated the power of simple vector operations. I've established a significant fondness for Processing, even though I didn't use it in the end I suspect I will be playing with it in the future and think it would have been great to have been exposed to it sooner.

SHORTCOMINGS

A number of things didn't get done that I had hoped for at the start. One was a volume shutter. Currently the slice plane is infinitely thin, and will only consider voxels that intersect its surface. By giving the slice plane some depth as in Figure 16, and having the voxels that fall inside this shape blend together somehow, some interesting results may be possible. Glassner[xvii] 1999 states that a more realistic shutter blurring can be achieved in rendering programs by modeling the 'shape' of the shutter i.e. the exposed area over time. His paper was also useful in helping to visualize time as a spatial axis.

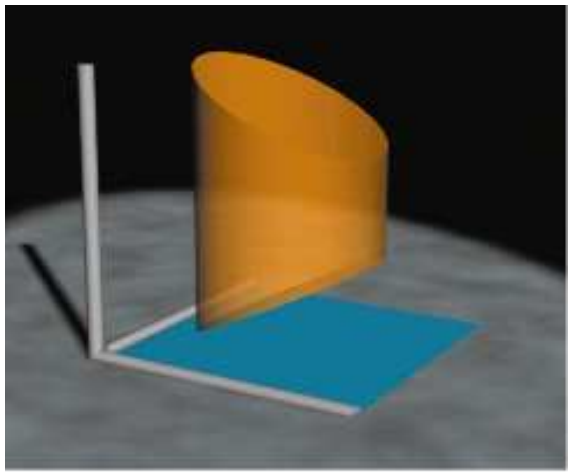


Figure 16 A circular shutter. The shutter cover moves in a left-to-right motion, stays open for a while, then slides back into place - Glassner

An important missing feature from a user's perspective is the lack of interactive controls for the slice plane in the 3D view. All translation is in world space also, so for example moving in a perpendicular direction to the slice plane when it is rotated arbitrarily is difficult. An object space switch would be very useful. Manipulating the slider parameters can be clunky, particularly in Windows where the slider speed accelerates after it has been held down for a few seconds. It is also difficult to tell where in space the slicer is, and where it intersects with the voxels. Some sort of graphic displaying this would have been useful.

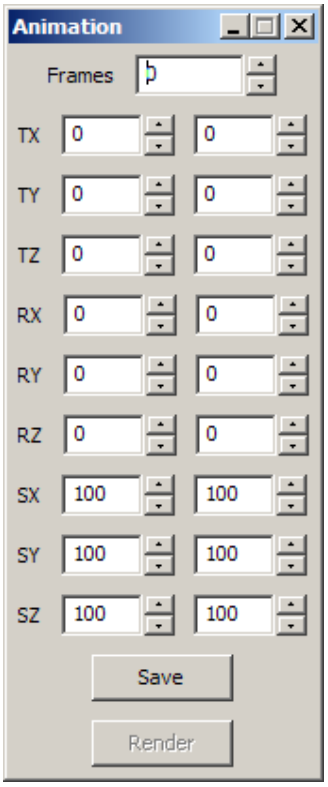
The implementation of the voxels is quite specific. Adding additional properties to a voxel such as velocity would be possible, but it would require changes in many areas of the program. Changing the size of a voxel would be extremely difficult as the whole program makes the assumption that a voxel is a one unit cube. Having variable sized voxels would be useful in supporting anamorphic video, but more practically as an optimization. For example on initialization the program would only create a new voxel if it had different properties to the one next to it. This could allow for huge memory savings in transparent images. Currently the memory is allocated as a single huge chunk and can easily exceed hundreds of megabytes. This can lead to instability in the event of memory failures.

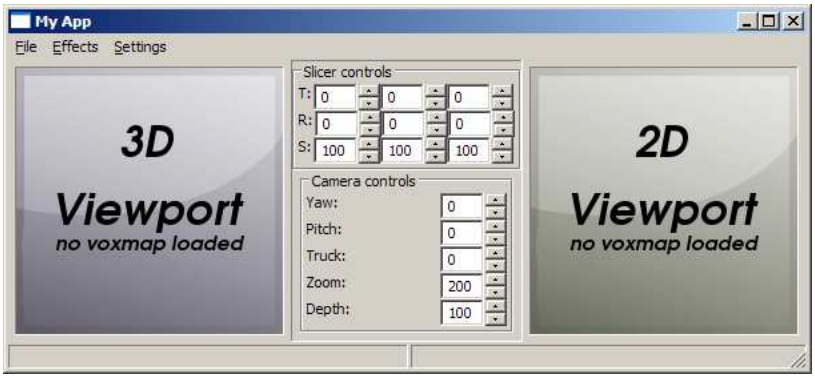
It can also be quite slow, particularly when dealing with PAL resolution video. The ray-marcher is sometimes so slow that it can be frustrating to use at a decent quality setting. The slicer is faster than I thought it would be, however with anti-aliasing it slows down considerably as it is doing 8 look-ups instead of one. Most slow of all are the 3D effects. They involve a 3D box (3x3x3) or (5x5x5) or (7x7x7) - the convolution kernel, traversing the entire voxmap performing weighting operations. These are completely unoptimised, most voxels get looked up 27 times each which is somewhat ridiculous. One way to improve this would be to only load in the 9 new voxels every time it goes onto a new base voxel instead of the full 27. I found creating and using these kernels the most unpleasant part of the project as it was difficult to troubleshoot and the 3d arrays became confusing. That is probably the reason there are only 2 filters and they are very slow.

CONCLUSION

I believe there is tremendous potential in treating video in this way. I was surprised by the relatively small amount of commercial development in the area for visual effects production as the method could have several applications. With some of the developments in plenoptic video photography it would be interesting to see how future acquisition devices differ from conventional still and video cameras and indeed how that captured data is then processed. I like the idea of capturing 'scene data' rather than simply capturing 'pictures' - digital versions of what our eyes see, and then trying to coax the relevant data out of it. Certainly for me, using this

unintuitive but fundamentally simple video cube technique has created an awareness of some of the potential that those developments might have.





By translating the slice in the direction indicated by the arrow, dynamic (moving) panoramas can be achieved. Though not a feature in the program, it may be possible to lay these blocks beside each other and create full 360° moving panoramas. Wavy lines on the roof of the box are an indication of non uniform panning speed.

Figure 17 Clockwise from left: The animation window for setting start and end frame parameters. The default screen on Windows. The volume of the kitchen panorama

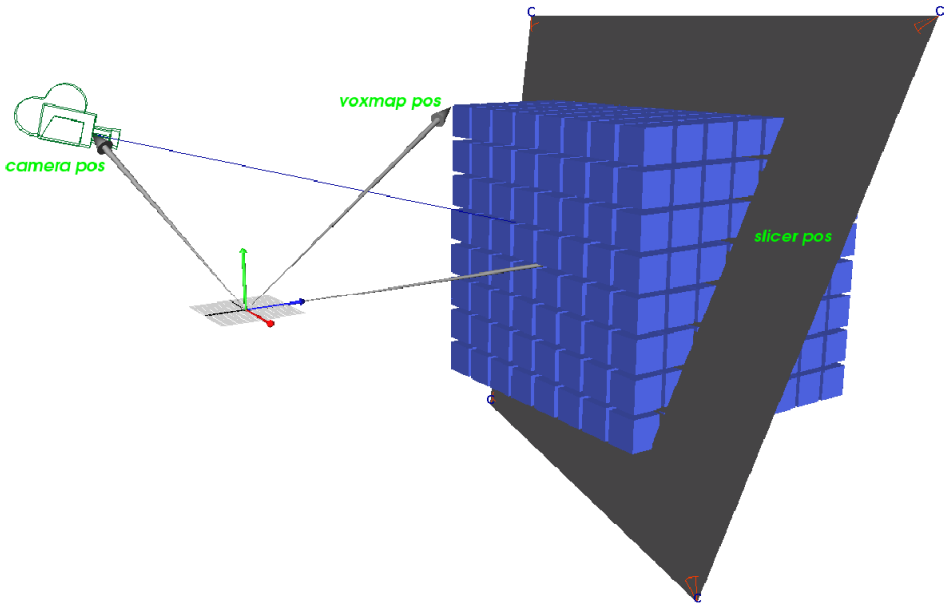


Figure 18. The structure of the 3d scene. Keeping a mental image of how things were laid out in the scene was invaluable when trying to get the pieces to work together

REFERENCES

[i] Andrew Davidhazy "Slit Scan Photography", School of Photographic Arts and Sciences, Rochester Institute of Technology. http://www.rit.edu/~andpph/text-slit-scan.html. Accessed Feb 12th 2008

[ii] S. Fels and K. Mase, "Interactive Video Cubism" In Proceedings of the Workshop on New Paradigms for Interactive Visualization and Manipulation (NPIVM), pages 78–82, Nov 1999

[iii] S. Fels, E. Lee, K. Mase, "Techniques for Interactive Video Cubism" Proceedings of ACM Multimedia. Pages 368-370. Oct. 2000.

[iv] TxTransform shake macro by Jean First available at http://highend3d.com/shake/downloads/macros/image_generation/3495.html

[v] Temporal Median Filter. royalstel: http://www.youtube.com/watch?v=16j91-nksfg

[vi] P. Hodgetts, "Displacement Mapping in Adobe After Effects". http://library.creativecow.net/articles/hodgetts_philip/displacement.php. Accessed 28th Feb 2008

[vii] Tx-tranform ©1998-2001 Martin Reinhart

[viii] A. W. Klein, P. Sloan, A. Finklestein, and M. Cohen, "Stylized video cubes," in Proceedings of SIGGRAPH 2002.

[ix] A. W. Klein, P. Sloan, A. Finklestein, and M. Cohen, "Stylized video cubes," in Proceedings of SIGGRAPH 2002.

[x] J. Neumann http://www.cfar.umd.edu/~jneumann/videogeometry/plenoptic.htm Accessed 9th March 2008

[xi] R. Ng, M Bredif, G Duval, M Levoy, M Horowitz, P Hanrahan "Light Field Photography with a Hand-held Plenoptic Camera" Stanford Tech Report CTSR 2005-02

[xii] A. Rav-Acha, Y. Pritch, D. Lischinski, S. Peleg "Evolving Time Fronts: Spatio-Temporal Video Warping" p5-6 Technical Report 2005-10, The Hebrew University of Jerusalem, 2005.

[xiii] Processing - Processing is an open source programming language and environment for people who want to program images, animation, and interactions. www.processing.org. Accessed continually

[xiv] wxWidgets - wxWidgets lets developers create applications for Win32, Mac OS X, GTK+, X11, Motif, WinCE, and more using one codebase. www.wxwidgets.org. Accessed continually

[xv] 2D blurring algorithm by Jim Scott. www.blackpawn.com/texts/blur Accessed 11th Feb 2008

[xvi] S. D. Ramsey, K. Potter Vector & Bilinear Patch implementation in C. Journal of Graphics Tools http://jgt.akpeters.com/papers/RamseyPotterHansen04/ 2003

[xvii] A. Glassner "An Open and Shut Case" IEEE Computer Graphics and Applications Vol1, Issue 3 1999