# Techniques for Interactive Video Cubism

Sidney Fels
Dept. of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada
ssfels@ece.ubc.ca

Eric Lee
Dept. of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada
elee@ece.ubc.ca

Kenji Mase
ATR M.I & C. Research Laboratories
Seika-cho, Soraku-gun
Kyoto, Japan
mase@mic.atr.co.jp

## ABSTRACT

This paper presents an interactive video visualization technique called video cubism. With this technique, video data is considered to be a block of three dimensional data where frames of video data comprise the third dimension. The user can observe and manipulate a cut plane or cut sphere through the video data. An external real-time video source may also be attached to the video cube. The visualization leads to images that are aesthetically interesting as well as being useful for image analysis.

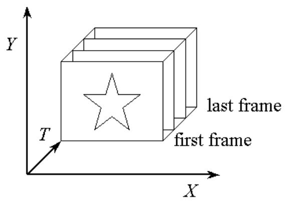
## 1. INTRODUCTION

We introduce a new technique for visualizing video data. In this novel scheme, video data is considered to be a volume of data. The dimensions of width and height are the usual X and Y axes of a frame of video data. The third dimension is derived from layering frames of video data sequentially in time as shown in the diagram (figure 1). Normal video viewing can be considered a cut plane that is parallel to the X-Y plane and advancing from the first frame to the last frame along the T axis as shown in figure 2 .

Now, imagine rotating the cut plane to a different location and moving it. For example, consider moving the cut plane so that it is parallel to the X-T axis and advancing it along the Y dimension. At each cut you are seeing all of the X dimension values for all the frames at a given position in the Y dimension as shown in the diagram in figure 3. Figure 4 shows an arbitrary rotation and positioning of the cut plane. Next, imagine a cut sphere instead of a plane as shown in figure 5. Here, we get a non-linear cut through the video data. We have implemented both a cut plane and a cut sphere which can be manipulated in real-time. Additionally, real-time video data can be streamed into the video cube.

## 2. RELATED WORK

Viewing video data along the X-T axis and Y-T axis has appeared in several forms in the literature. Most recently, [4] has developed a technique, called the *tx-transform*, for use with film. In their work, the different cut planes are always aligned with the basis axis and are used for creating aesthetically interesting dynamic viewpoints of film data. The work, *The Invisible Shape of Things Past* [5] also represents video as a three dimensional object where the topology is determined by the characteristics of the video camera. In [1], they describe epipolar-plane analysis for tracking objects in motion. In this work, the cut plane images through the video cube are are analysed for straight lines or hyberbolic curves to track objects during camera motion on a mobile robot. In their work, they consider the effect of moving a camera in straight lines relative to a fixed scene. In video cubism, the camera and the objects are free to move. The complex patterns that form are due to the plane or sphere cutting



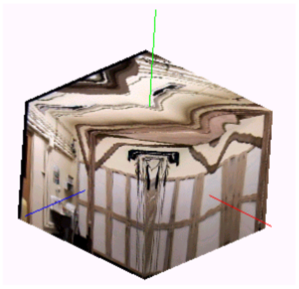


**Figure 1: Video frames are stacked together to form a volume. The right figure shows a video cube with 7 seconds of data (210 frames; 212X160 pixels). The scene is a room with camera panning and zooming.**

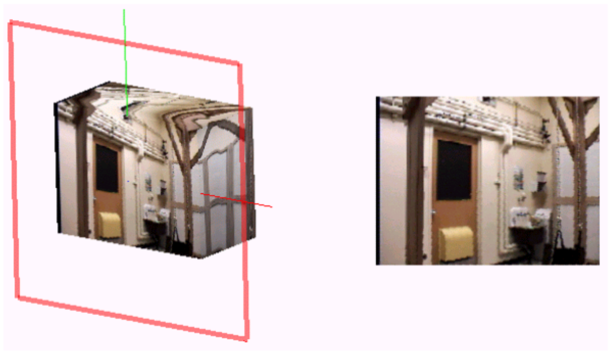


**Figure 2: Cutting the video cube parallel to the X-Y plane shows a single video frame at some point in time. Animating along the T axis in this manner displays normal video. The image on the right is the cut plane viewed head-on.**

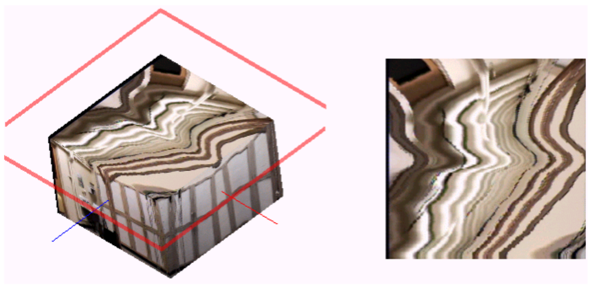


**Figure 3: Cutting the video cube parallel to the X-Z plane. Stationary objects leave a smooth "trail" due to camera movement.**

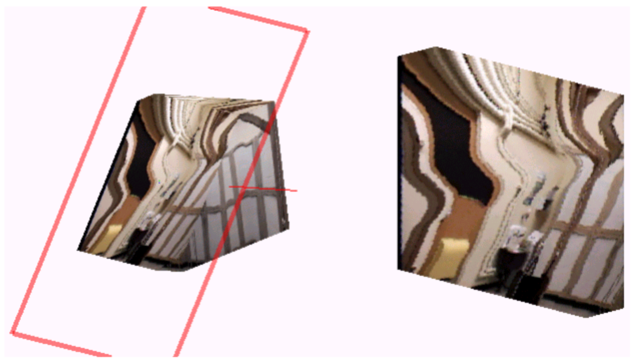


**Figure 4: An arbitrary cut through the video cube. Camera movement translates to an interesting "bending" effect on the door.**

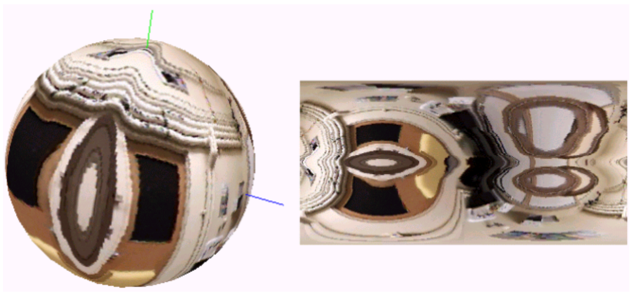


**Figure 5: A spherical cut applied to the cube. The image on the right is the texture map applied to the surface of the sphere. Two "doors" appear in the image; one from front of the original cube and the other from the back.**

through the epipolar lines allowing multiple representations of the spatiotemporal data.

The main distinctions this work has are that the cut plane or sphere used to view the video data can be manipulated in real-time and that real-time video data can be streamed into the video cube. This provides an opportunity to interactively explore the video cube from many different angles to get both aesthetically interesting static images as well as motion effects.

## 3. VIDEO CUBISM

Video cubism has three main parts, the video data buffer, the virtual cube and the cut surfaces. 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.

### 3.1 The Video Data Buffer

The first component of the system is the video data buffer. The complete video data is stored in memory as a 3D array consisting of a sequence of frames of video data. Currently, we are using RGB values for video frames that are 212x160pixels. Using the full 3D array representation made addressing individual video data (vixels) that are on the cut plane simple. In contrast to [3], the video buffer may also dynamically receive data either from a video capture card instead of a file.

The video data is used to form textures which are mapped onto appropriate faces of the video cube and cut surface.

### 3.2 The Video Cube

The video cube is an abstract representation of the video buffer discussed in section 3.1. The appropriate vixel data from the video buffer is texture mapped onto the six faces, with the most recent frame mapped to the front of the cube and the oldest frame at the back. Note that the dimensions of the cube can be selected arbitrarily; the texture map will stretch the vixel data to fit accordingly. This is similar to the technique used by [2]. In this implementation, we use a video cube centred at the origin, with dimensions 1.0x0.75x1.0. The dimensions were chosen to preserve the 4:3 aspect ratio of the video frames.

The video cube is represented as an unordered set of twelve line segments, one for each edge of the cube. The faces of the cube are divided into triangles, with each triangle texture mapped separately. Tessellating the faces simplifies the implementation and makes it consistent with the texture mapping process for the cut plane's intersection polygon, discussed in section 3.3.1. The cube can also be arbitrarily rotated around the origin in the world coordinate system. We use a single composite matrix, composed from rotation, scaling, and translation matrices to convert from video cube (x,y,z) coordinates to video buffer (X,Y,T) coordinates.

### 3.3 The Cut Surfaces

Two cut surfaces have been implemented: the cut plane and the cut sphere. The techniques used for these cuts can be extended to other cut surfaces.

#### *3.3.1 The Cut Plane*

The cut plane allows the user to move a planar window inside the video cube and examine the corresponding imagery (see figure 6). Every time the cube or the plane is moved, an intersection polygon must be calculated and a texture map computed.

The vertices of the intersection polygon can be computed by finding the points of intersection between each line segment representing the edges of the cube with the cut plane.

The intersection points are sorted into an ordered list of vertices for a convex polygon. Coordinates of the intersecting

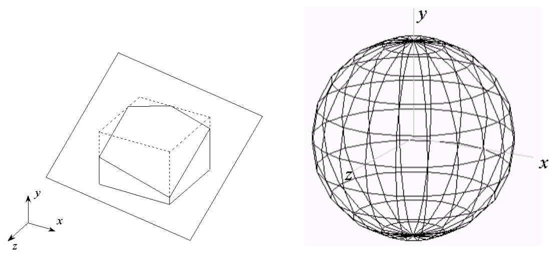


**Figure 6: The cut plane and sphere. The intersection of the cut plane and cube is a polygon. This polygon can have three to six sides, depending on the orientation of the cut plane and cube. Shown is a wire frame cut sphere divided into 10 stacks and 20 slices.**

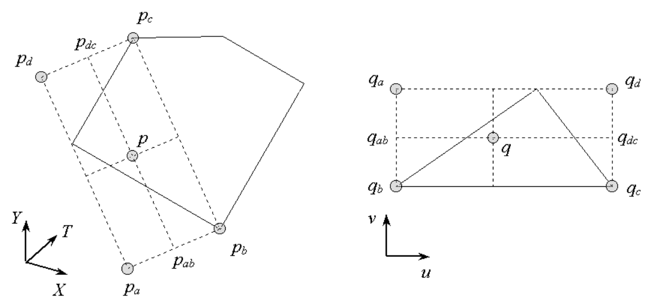


**Figure 7: Computing the video buffer coordinates, $p$, of a point in the texture map, $q$, using a weighting function based on the four corners of a bounding box.**

polygon are used to determine the texture map vixels.

In contrast to [3], we tessellate the intersection polygon into triangles and texture map each triangle separately. Using this procedure, we iterate over the pixels inside each triangle directly, eliminating any per-pixel bounds check. Furthermore, instead of rotating each pixel from texture map coordinates to video buffer coordinates, we rotate only the four corners of the bounding box of the triangle we are mapping. Then, we interpolate the interior texels based on the coordinates of the four corners of the bounding box to determine their video buffer coordinates. Figure 7 illustrates this process.

Given the texture map coordinates $q_{ab}$ and $q_{dc}$ for a particular row of the texture map, the corresponding values $p_{ab}$ and $p_{dc}$ in video buffer coordinates can be calculated. Then $p$ is calculated using the following equation:

$$p = \frac{(p_{dc} - p_{ab})(q - q_{ab})}{q_{dc} - q_{ab}} + p_{ab} = C(q - q_{ab}) + p_{ab}$$

Note that the values C, $q_{ab}$ and $p_{ab}$ are constant for each row; thus, the number of operations has been reduced to only three multiplications and six additions/subtractions for each texel.

Profiling on a 650MHz PentiumIII processor, we found that the computation time for a triangle takes from 7 to 11 milliseconds, depending on the size of the triangle. From this, we estimate that the largest intersection polygon requires 36 milliseconds to calculate the texture.

### 3.3.2 The Cut Sphere

We have also implemented a cut sphere. In this scheme, a sphere is placed in the centre of the cube, and data outside of the sphere is removed. The cut sphere is interesting because, unlike the cut plane, the image on the surface of the sphere is a non-linear distortion of space and time. We use a single texture map wrapped around the surface of the sphere to display the video data. Figure 5 shows an example of the cut sphere.

## 4. INTERACTION CONTROLS

Using the mouse as a virtual trackball, the user is able to rotate or translate the entire scene, the video cube, or the cut plane. Alternatively, the rotations and translations may be specified absolutely using the keyboard, or by manipulating sliders on a control panel. Note that since we assume the cut plane to have an infinite width and height, only translations along the plane normal have any effect.

For the cut plane, the user may animation the cut plane to move along its normal. For the cut sphere, animation dynamically change the size of the sphere which has a "zooming" effect. If the cut plane is aligned along the normal, animating the plane will result in each frame being displayed in its normal or reverse sequence. Rotating the plane slightly induces temporal effects. One interesting temporal effect is that solid objects can be made to appear to bend by rotating the cut plane along the $z$ axis while rotating the object.

## 5. CONCLUSIONS

With video cubism it is possible to interactively explore video data in all three dimensions simultaneously. The main purpose is to explore some of the aesthetics of looking at video data from a variety of perspectives. The images can be abstract or concrete depending upon the orientation and position of the cut surface as well as the movement of the camera and the object in the video data. We also plan to continue investigating ways to explore the dynamic imagery possible with the video cube.

*Acknowledgments*

The authors thank Ivan Poupyrev for helpful discussions.

## 6. REFERENCES

[1] Robert C. Bolles, H. Harlyn Baker, and David H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1(1):7–55, 1987.

[2] S. S. Fels and K. Mase. Iamascope: A graphical musical instrument. *Computers and Graphics*, 2:277–286, 1999.

[3] S. 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.

[4] Martin Reinhart. tx-transform. http://www.tx-transform.com/frame_e.htm , 1998.

[5] Joachim Sauter and Dirk Lusebrink. The invisible shape of things past. Available from: http://www.artcom.de/projects/invisible_shape/welcome.en, 1997.