3D Video Sprites

Stephen Pollard, Sean Hayes
Digital Media Department
HP Laboratories Bristol
HPL-98-25
February, 1998

image processing, digital video processing, image based rendering, computer graphics

Here we introduce the concept of a 3D video sprite. We outline an approach for their automatic creation and manipulation. 3D sprites consists of a number of synchronous video streams and mark up information that allows virtual viewpoints with respect to a live action video to be rendered and combined with traditional 3D virtual environments. As such they constitute a form of image based rendering.

Our generation method uses trinocular stereo matching to provide a morphing channel. The matching of corresponding locations in the images is edge-based and relies on the extraction of suitable epipolar geometries. The resulting edge correspondences are then interpolated and used in an efficient morphing algorithm that operates one scan-line at a time to perform image synthesis.

Easily-obtained viewing geometry is used to combine 3D video sprites with computer generated 3D environments.

Internal Accession Date Only

Contents

1. 3D video sprites

3D video sprites are an extension of existing digital blue screen techniques to allow greater flexibility in rendering video footage against computer generated three dimensional backdrops. In current systems the location of the live action camera is either fixed or follows a precisely calibrated path during capture and the rendered scene is required to be consistent with it.

3D video sprites, on the other hand, can dynamically combine blue screen footage captured simultaneously from a number of viewpoints to create video sequences from novel viewing directions. This allows the 3D video sprite to be treated more like a graphic object and allows the vantage with respect to the video overlay, and the backdrop, to be determined after the fact.

Depending upon the availability of real-time capture and rendering hardware, a large number of applications of 3D video sprites exist. Off line capture and rendering can be employed for the kind of post production effects eluded to above that give greater flexibility to the film maker. Adding real-time rendering creates a new digital video media, similar to computer graphic games, that allows truly immersive interaction with respect to a viewer who changes their viewpoint either directly through a sensed feed back loop or indirectly through a traditional pointing paradigm (e.g. the mouse/joystick). Extended yet further to include automatic capture and image processing, allows us to consider the presence in a rendered virtual world of real life avatars that present themselves with full three-dimensionality and support the creation of natural meeting places in cyberspace.

3D video sprites are not true 3D objects; they consists of a number of synchronous video streams and mark-up information that allows other views to be morphed (interpolated and blended) from the original set. As such they constitute a form of image based rendering.

The advantage of image based approaches is a naturalness that is very hard to achieve with graphic objects, and the rendering time is not proportional to the complexity of a model, but only the number of output pixels – a significant advantage for real time applications.

2. Image based rendering

A growing number of researchers are exploring ways of constructing static and temporally varying immersive scenes using real world image data alone.

One approach is to capture a large number of viewpoints and use these as an environment map [4] to be applied as a texture on some imaging surface. New viewpoints can be generated by projecting the texture back onto the imaging plane corresponding to the user's current view. Environment maps can be obtained as panoramas composed from multiple images [2] or based upon plenoptic capture by lens and/or mirror arrangements [8]. See [12] for a good overview of image mosaic generation to capture virtual environments.

It is possible to go beyond the exploration of 2D worlds (where the viewer is constrained to a single, or predetermined set of discrete locations in the 3D environment) by recovering a dense depth map from multiple discrete viewpoints and employing a standard texture mapping technique to view that surface from an alternative viewpoint (e.g., see [6]). Chen and Williams [3] studied the interpolation of intermediate views from 3D data. Laveau and Faugeras. [7] bypass the reconstruction-projection paradigm, and use *transfer* with projective invariants to predict, from dense correspondences, where pixels end up in virtual projectively distorted images. A more recent approach is to directly represent the light field in the vicinity of an object [5].

An approach part way between the 2D and 3D paradigms is that proposed by Seitz and Dyer [10], which uses *image morphing* techniques to synthesise viewpoints between two original images. They observe that under fairly general assumptions, veridical virtual viewpoints can be constructed by linearly interpolating corresponding uniform regions from the two images.

The method of constructing 3D video sprites [9] builds directly upon the earlier approach of Seitz and Dyer. This method uses a different matching and morphing strategy that does not require the images to be pre-warped and then post-warped for each viewpoint for more efficient rendering. More importantly, we extend the approach beyond the two image case, thereby allowing the user to explore the change of viewpoint in two dimensions, which considerably enhances the 3D experience.

Figure1: 3D Sprite Video Example

3. The morphing channel

Typically, as illustrated in figure 1, we represent 3D video sprites as 3 synchronous streams of video data plus a morphing channel. Each video stream is a view of the same foreground subject against a suitable blue or green screen background. Typically the subject will be a person or other unconstrained animate object. The 3 cameras are best arranged at the corners of an approximate equal sided triangle where 2 of the cameras form a horizontal baseline and the 3rd camera is either above or below it. Only approximate geometry is required (i.e. that that can be estimated with a tape-measure) for integrating 3D video sprites with true 3D geometrical primitives

The translation between the cameras orthogonal to their principal axes induces an apparent 3D rotation of the viewed object. The magnitude of the rotation is dependent upon the relationship between the length of the baseline between the cameras and the viewing distance. By arranging our cameras at the corners of an equal sided triangle with the principal axes of each camera roughly orthogonal to the plane passing through the optic centres of each (the vantage plane) we induce rotations between the images in the horizontal and vertical directions.

If we were to place a camera at an arbitrary point on the vantage plane within the extent of the original camera triangle we would produce a particular view of our subject. Using 3D sprites we produce a similar image by using the information in the morphing channel to interpolate approximate versions of the views on the vantage plane.

The morphing channel relates the images of the subject in each of the 3 views. It must be extracted automatically for effective 3D video sprite creation. A blue/green screening process is employed to delimit background and subject regions of each image. Only the foreground subject needs to be represented in the morphing channel.

A number of methods can be used to extract and represent the morphing channel. We have considered mesh and edge based approaches. In the former a contiguous set of triangles (or other simple polygonal mesh) is used to tessellate the active area of each view such that correspondences amongst the vertices refer to the same physical locations in the scene. Morphing constitutes an interpolation of the vertices of each individual triangle into a new virtual view. The contents of the triangles are then obtained by texture mapping and alpha blending the originals.

Despite the suitability of the above approach to real-time rendering on state of the art graphic processors we prefer to use an edge based approach, as this greatly simplifies the job of automatically extracting correspondence information between the 3 views. Edges obtained from all 3 views are matched using standard computer vision techniques (as described in [9]) to give strings of matched edge points that correspond to the primary intensity discontinuities of the subject. Edge strings are linked according to the local adjacency of edge points in each of the 3 views.

3D video sprite rendering is then based upon the linear interpolation of the matched edges to form a virtual line sketch of the new view. The regions between the interpolated edges is then filled using the texture data from each of the 3 views.

3.1 Interpolating edges

Figure 2 depicts how matched edges are interpolated to generate virtual viewpoints within the original image triple. Each string of matched edge points is interpolated according to the parameter pair (α, β) that specify the location **V** of the new view with respect to the original set. Physically, α specifies a view between image 0 and image 1 and β specifies the location between that point and the location in

image 2. Thus, the i^{th} edge point along the string has projection into the three views at $\mathbf{p}_{0i}$, $\mathbf{p}_{1i}$ and $\mathbf{p}_{2i}$ and into the synthesised view at location $\mathbf{p}_{Si}$ given by

$$\mathbf{p}_{Si} = (1-\beta)((1-\alpha)\mathbf{p}_{0i}+\alpha\mathbf{p}_{1i}) + \beta\mathbf{p}_{2i}$$

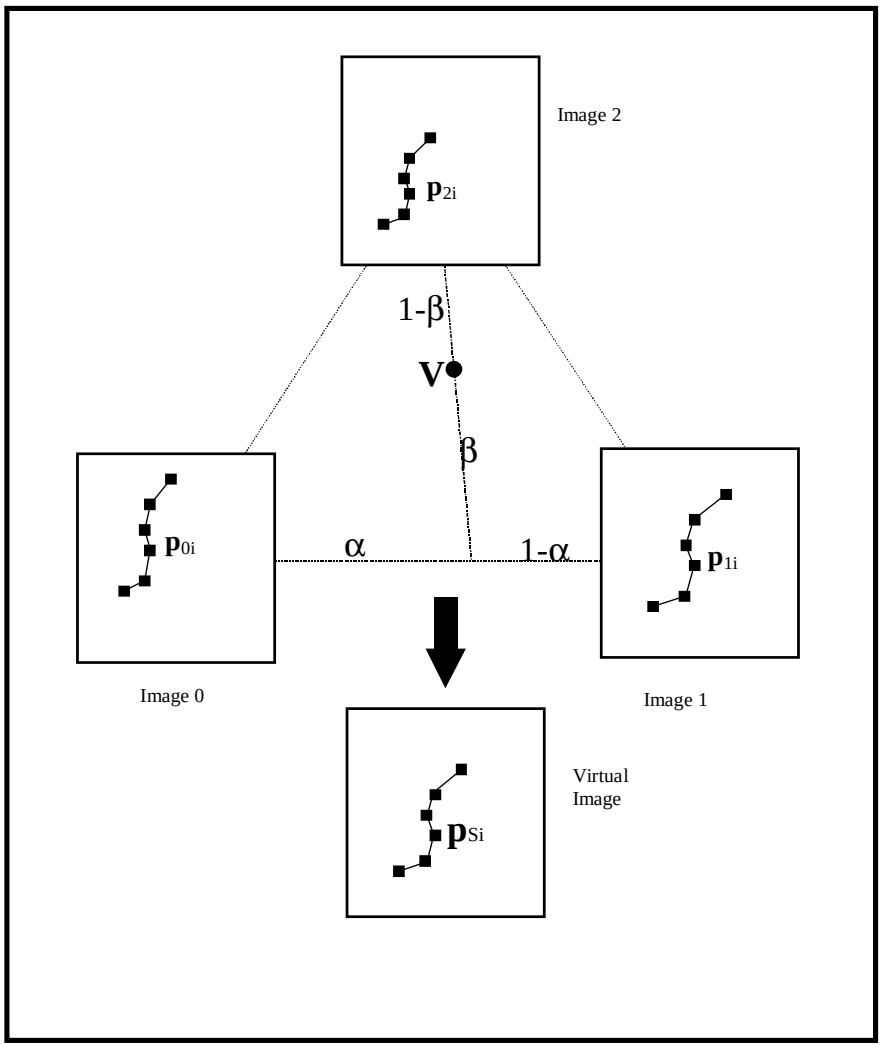

Figure 2: Interpolating Edges

When creating video sprites we are interested in producing a video object that can be rotated in a virtual 3D world (or can match the rotation induced by motion of a point being tracked in another video sequence). However, some degree of image plane translation of the subject may also be present between the 3 views, unchecked this results in an unwanted translation of the video sprite across the image as the edge geometry between the different views is interpolated. The solution is to absorb the

unwanted translation into the interpolation process itself by shifting the center of gravity of each set of edges to a common image location. Thus the interpolation becomes:

$$\mathbf{p}_{Si} = (1-\beta)((1-\alpha)\mathbf{p}_{0i} + \alpha(\mathbf{t}_1+\mathbf{p}_{1i}) + \beta(\mathbf{t}_2+\mathbf{p}_{2i})$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are the translation of the centre of gravity of the matched edges in image 0 with respect to images 1 and 2 respectively.

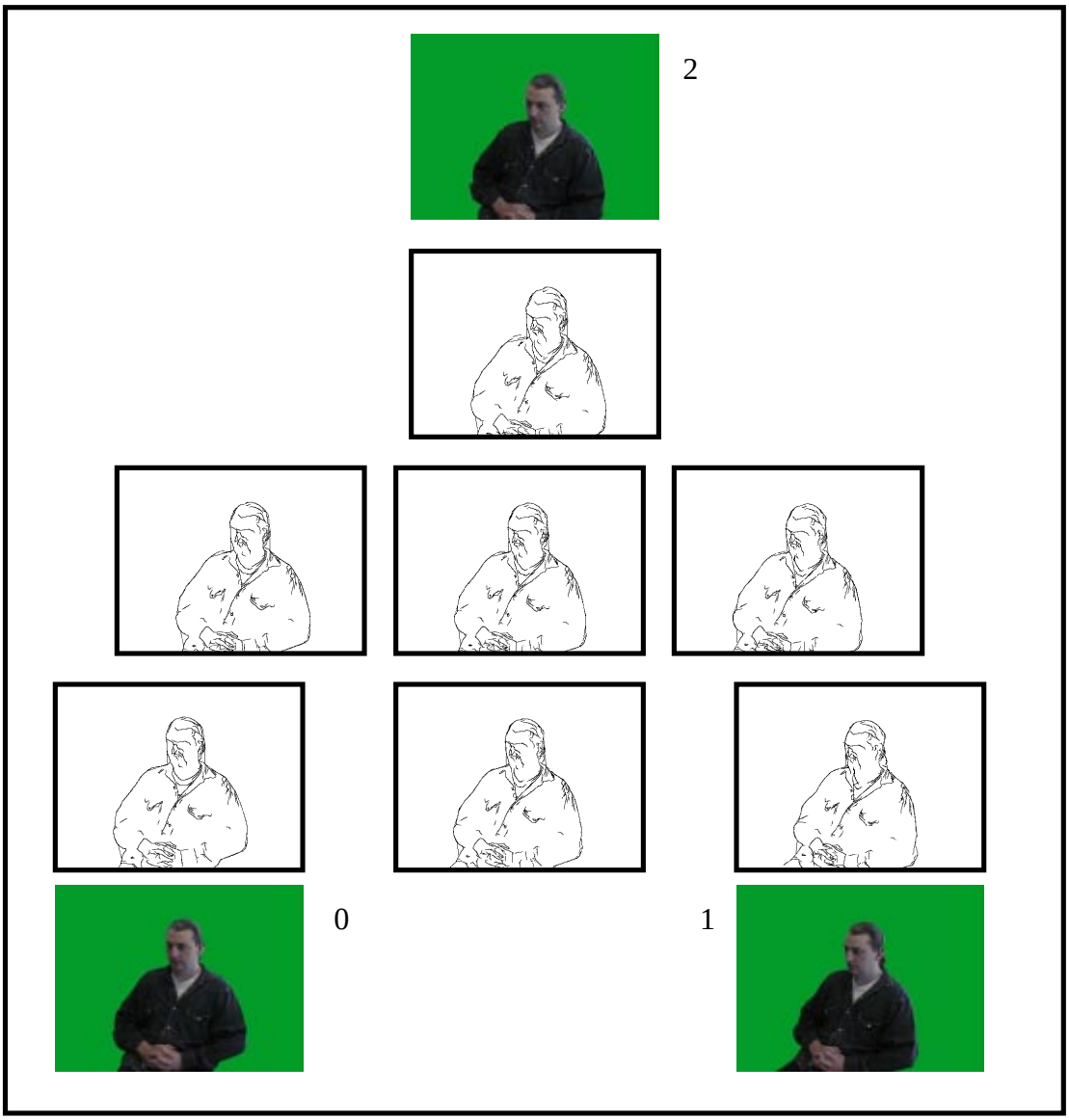

Figure 3: Sketch Interpolation

Figure 3 gives an example of interpolating edge sketches obtained from a test image triple which have have mock blue screen properties (the region around the subject was eliminated by hand). The 3 corner

images are shown along with their associated edge sketches plus those for a number of intermediate locations.

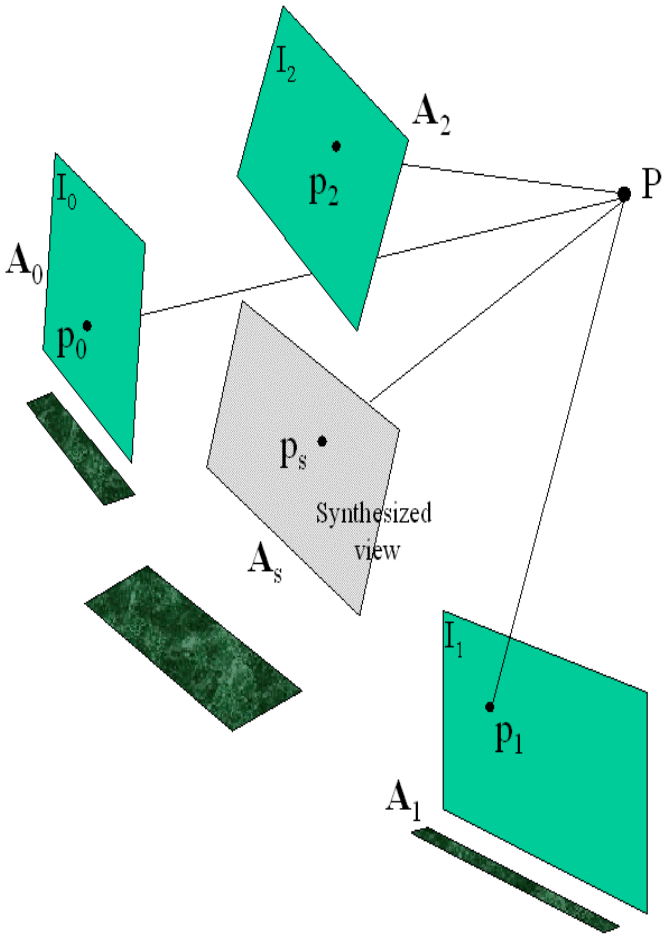

Figure 4: Trinocular Geometry

4. The geometry of view interpolation

In this section we outline a simple physical interpretation of what interpolating in the image plane means. Note that the arguments presented here for 3 cameras can be extended to any number of cameras.

An important early work by Ullman and Basri [13] shows the conditions under which linearly interpolating orthographic views produces other veridical views in the context of model recognition.

Chen and Williams [3] used interpolation between range data to synthesise intermediate views and again found that valid views are produced only under some circumstances. More recently Seitz and Dyer [10] demonstrated that interpolating parallel camera images always produces valid in-between views.

If we assume that the viewing distance is reasonably large with respect to the depth in the scene, which is typically the case for 3D video sprites. Then the cameras can be approximated with an affine model (see, e.g., [11]). Let us refer to figure 4 and consider a point $\mathbf{P}=[X\ Y\ Z\ 1]^T$ in the space imaged by three affine, uncalibrated cameras defined by affine projection matrices $\mathbf{A}_0$, $\mathbf{A}_1$ and $\mathbf{A}_2$. Each matrix is of the form

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \end{bmatrix}$$

and is scaled, without loss of generality, so that $a_{34} = 1$

The projections of a point **P** onto the image planes of the three cameras are given by

$\mathbf{p}_0=[x_0\ y_0\ 1]^T=\mathbf{A}_0\mathbf{P}$

$\mathbf{p}_1=[x_1\ y_1\ 1]^T=\mathbf{A}_1\mathbf{P}$

$\mathbf{p}_2=[x_2\ y_2\ 1]^T=\mathbf{A}_2\mathbf{P}$

Let the interpolation of these three points in image plane co-ordinates be given by:

$$\mathbf{p}_s = (1-\beta)((1-\alpha)\mathbf{p}_0 + \alpha\mathbf{p}_1) + \beta\mathbf{p}_2$$
$$\mathbf{p}_s = (1-\beta)((1-\alpha)\mathbf{A}_0\mathbf{P} + \alpha\mathbf{A}_1\mathbf{P}) + \beta\mathbf{A}_2\mathbf{P}$$
$$\mathbf{p}_s = \mathbf{A}_2\mathbf{P}$$

where

$$\mathbf{A}_s = (1-\beta)((1-\alpha)\mathbf{A}_0 + \alpha\mathbf{A}_1) + \beta\mathbf{A}_2 .$$

Thus interpolation in the image plane produces the same effect as having an another interpolated affine camera $\mathbf{A}_s$.

Since we do not use parallel camera rectification as in [9], we have to understand what an interpolated affine camera matrix represents.

An affine transformation can be seen as a parametric mapping from $\Re^3 \rightarrow \Re^2$

$$\mathbf{A}_i = \mathbf{A}_i(\mathbf{v}) = \mathbf{A}_i(\theta, \vartheta, \psi, t_x, t_y, t_z, S, S_x, S_y)$$

function of the camera reference frame orientation and position, plus a shearing and two scaling components, respectively.

Now, since the transformation is linear in translation, scaling and shearing, if no rotation between the cameras $\mathbf{A}_0$, $\mathbf{A}_1$ and $\mathbf{A}_2$ is involved, $\mathbf{A}_s$ represents a perfectly valid new viewpoint $\mathbf{V}$.

On the other hand, when rotation is involved this is no longer true. However, provided the relative rotations between the cameras are small, there is a near-linear relationship between changes in the elements of the affine matrices and changes in the gaze angles. Hence, under these conditions, in general we can write:

$$\mathbf{A}_s \approx \mathbf{A}_0((1 - f_\beta(\beta))((1 - f_\alpha(\alpha))\mathbf{v}_0 + f_\alpha(\alpha)\mathbf{v}_1) + f_\beta(\beta)\mathbf{v}_2)$$

where $f_\alpha(\alpha)$ $f_\beta(\beta)$ are non-linear functions of α and β. Thus the synthesised viewpoint, neglecting second order effects, simulates the camera on the hyper-plane through $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$.

5. Epipolar geometry

Each pair of cameras are related by an epipolar geometry [15]. This determines for each point in one image the single epipolar line along which corresponding points must lie in the other. This in effect reduces the matching problem between the images from 2D to 1D. Intuitively, the epipolar geometry results from the fact that each point in the world forms a plane through the optic centres of each camera and that plane intersects the imaging planes of each camera as a pair of corresponding epipolar lines. In fact the whole set of epipolar lines corresponds to the set of planes that includes the axis connecting the optical centres of each camera.

The epipolar geometry can be computed from the intrinsic and extrinsic parameters of the cameras but in practice this involves complex calibration processes. We prefer direct computation of the epipolar geometry from image data alone using the property that the relationship between two corresponding points $\mathbf{p_0}$ and $\mathbf{p_1}$, extracted from images 0 and 1 respectively, is given by $\mathbf{p_1}^T \mathbf{F_{01}}\, \mathbf{p_0} = 0$ where $\mathbf{F_{01}}$ is the so called *Fundamental Matrix* from images 0 to 1. The matrix $\mathbf{F_{01}}$ is a 3x3 matrix of rank 2 and relates features in one image to the corresponding epipolar lines in the other. It has the form:

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{22} \\ f_{31} & f_{32} & 1 \end{bmatrix}$$

Given the fundamental matrix $\mathbf{F_{01}}$ relating a pair of images, then a point $\mathbf{p_0}$ in image 0 defines an epipolar line $\mathbf{e_{10}}$ in image 1 (the double subscript indicates the epipolar in image 1 based upon image 0) according to $\mathbf{e_{10}} = \mathbf{F_{01}}\, \mathbf{p_0}$ **(**such that the components of the vector $\mathbf{e_{10}}$ are ***(a, b, c)*** and define the line by way of the equation $\boldsymbol{ax + by + c = 0)}$. Each and every point $\mathbf{p_i}$ on the epipolar line $\mathbf{e_{10}}$ in image 1 defines the same corresponding epipolar line in image 0 according to $\mathbf{e_{01}} = \mathbf{F_{10}}\, \mathbf{p_i} = \mathbf{F_{01}}^T \mathbf{p_i}$.

In practice for subjects of limited depth with respect to viewing distance an affine form of the fundamental matrix [11] is preferred.

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & f_{13} \\ 0 & 0 & f_{22} \\ f_{31} & f_{32} & 1 \end{bmatrix}$$

In this case the epipolar lines are restricted to be parallel which is sufficient for the kind of image of image geometries employed here (provided the overall depth variation is small).

Epipolar geometry need only be computed once for fixed geometry video sequences. We use the method of Zhang & Deriche [16] to automatically extract epipolar information. The epipolar geometry is used to allow us to match up the images properly rather than recover either 3D structure or calibration parameters.

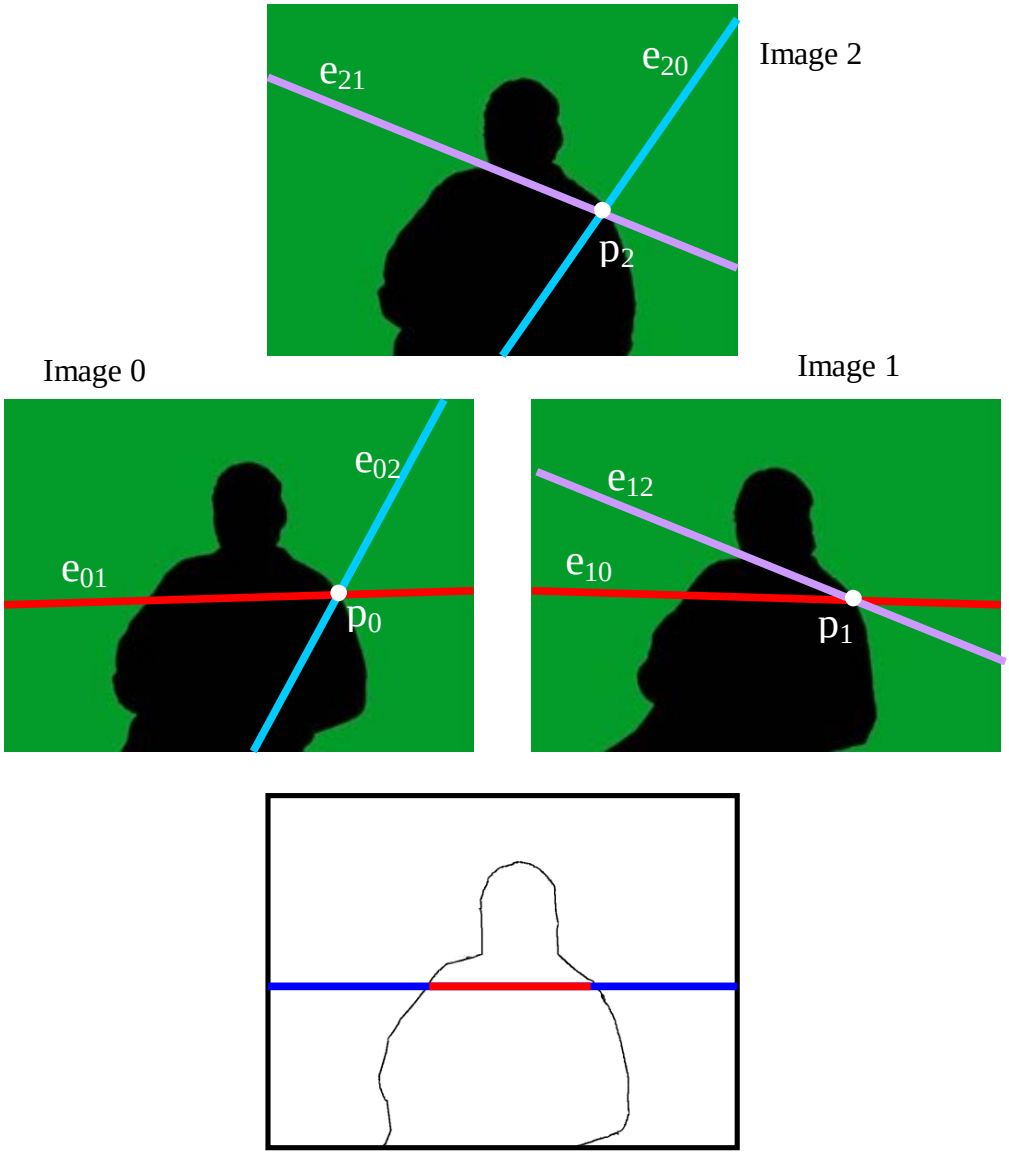

Figure 5: Blue Mask Rendering.

6. Raster rendering

Consider first just the edge around the blue screen mask. Matching and interpolating this corresponds, approximately, to treating the viewed object as lying on a lamina surface. In the middle 2 images of figure 5 consider the intersection of the red epipolar pair from images 0 and 1 with the outline masks. A point $\mathbf{p_0}$ along the epipolar $\mathbf{e_{01}}$ on the boundary of the mask in image 0 will lie at the point along epipolar line $\mathbf{e_{10}}$ in image 1 where it to intersects the boundary of the mask, call it $\mathbf{p_1}$. It follows that the corresponding point in image 2 (at the top of figure 5) will lie at (or close to) the intersection of the epipolar lines $\mathbf{e_{20}}$ and $\mathbf{e_{21}}$ that are defined by the points $\mathbf{p_0}$ and $\mathbf{p_1}$ respectively. In this way each point on

the boundary in image 0 can be brought into correspondence with matching boundary points in each of the other 2 images.

The viewpoint interpolation of boundary edge triples according to interpolation scheme outlined in section 3.1 gives intermediate boundary representations such as the one shown at the bottom of figure 5 (obtained with interpolation parameters $\alpha=\beta=0.5$). Also shown is a raster of a virtual image. The interval of the raster within the boundary corresponds to a line in each of the primary images. The respective locations of the start and end of this line are determined by the boundary locations whose interpolation gave rise to the start and end of the raster interval itself. In general, these will be different locations in each of the 3 images and will involve interpolation along the synthesised boundary to give improved accuracy of location.

Rendering the virtual viewpoint based upon boundary information alone would introduce considerable distortion. However internal edges tend to delimit regions of uniform intensity or slow intensity variations. So provided that the order of edges is preserved along the raster of the virtual view (this is the so-called monotonicity constraint of [9]) with respect to each of the original images then linear interpolation between interpolated edges should be sufficient to preserve the intensity patterns of the original.

Hence the interval between each successive pair of edge intersections within a raster in the virtual-viewpoint sketch is filled using a combination of the image data obtained from corresponding intervals in the primary images. A similar method was adopted by Seitz and Dyer [9] for binocular images obtained from parallel camera geometries (obtained from more general viewing geometries by image rectification), however it is not straightforward to extend their approach to situations involving more than two cameras.

Figure 6 shows, in the bottom left image, a selected raster within a virtual viewpoint of which the section between a pair of successive interpolated edges has been marked. In the top 3 images the corresponding intervals in the primary images have also been marked. The algorithm uses an *intersection table* to efficiently identify the projection of the raster interval with respect to the 3 original

views. This table is built incrementally during edge interpolation stage (Section 3.1). Each entry consists of an edge intersection with respect to a raster of the virtual view and the co-ordinates of corresponding points in each of the three views. The table is indexed spatially, ordered along the raster, so that intervals are efficiently obtained from successive entries.

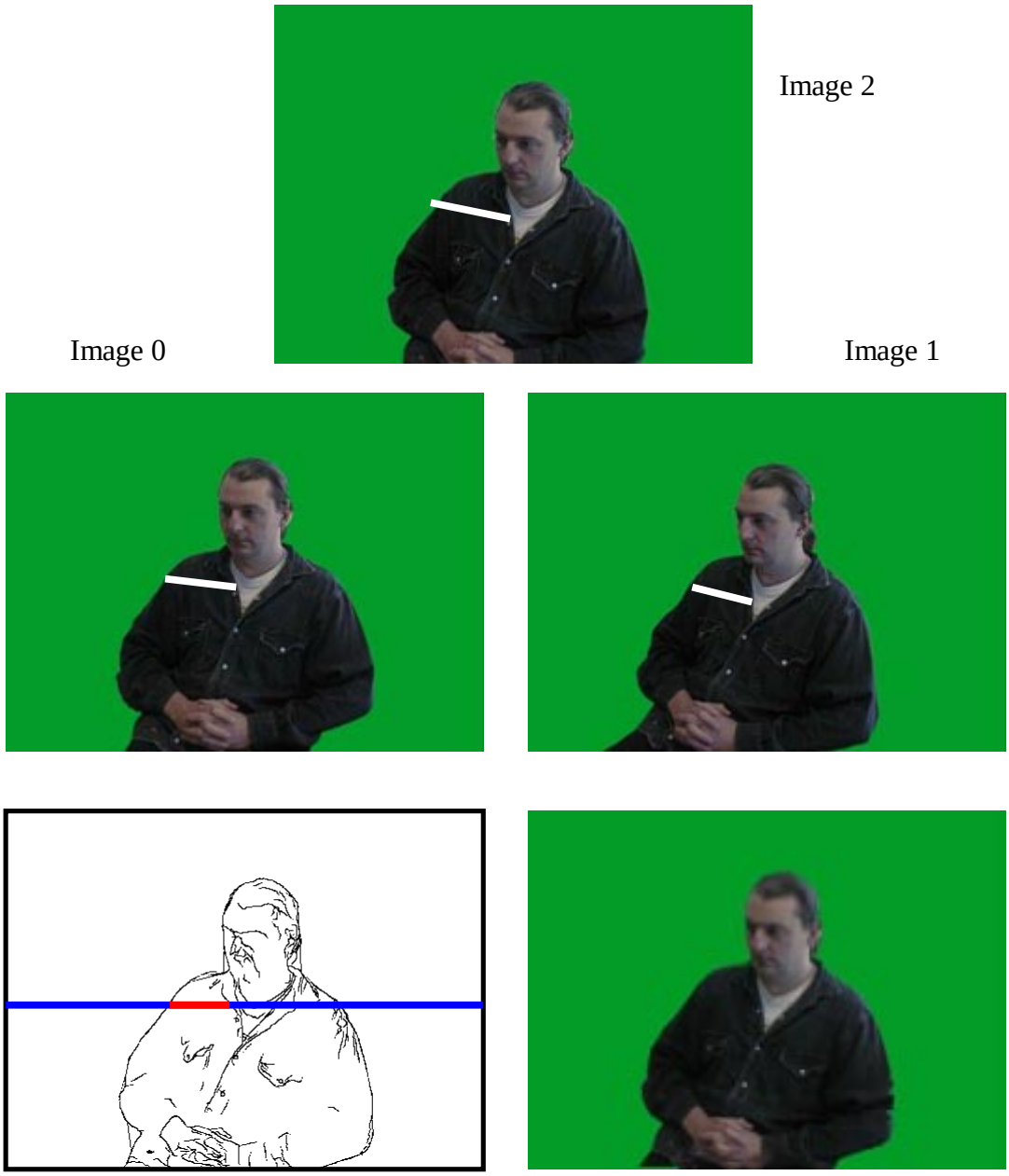

Figure 6: Edge Based Rendering.

The rendered pixels in the raster interval are thus obtained by blending the pixels from the three corresponding intervals in the primary images. As with standard image morphing techniques [14] the blend of the pixel contributions from each of the three images is linearly weighted according to the viewpoint parameter pair (α, β)**.** Aliasing artefacts are reduced by using pixel-level bilinear interpolation when sampling the primary images [14]. The rendered image is shown at the bottom right of figure 6.

7. Frequency controlled blending

It is possible for some small degree of matching and/or interpolation error to occur between the 3 views. This has the effect of blurring the virtual images as a result of image blending.

Alternatively we can eliminate blending and simply use the closest of the primary images as texture data. This introduces influence zones with hard edges, for example if α and β are both less than 0.5 we always use image 0 for texture mapping, if only α exceeds 0.5 we use only image 1 and whenever β exceeds 0.5 we use only image 2. Whilst sharp definition is maintained in any given image this method introduces unsightly transitions in image sequences as the parameters move from one influence zone to another.

A computationally more expensive but visually pleasing compromise is to introduce a form of frequency controlled blending. This employs the same influence zones as the single image approach but uses the distance to the transition between zones to determine the contribution of each frequency. The method follows the spline based image registration approach of Burt and Adelson [1].

In the method each image is represented by a stack of 4 frequency component images. These are created by gaussian smoothing the original and subtracting to give a high frequency component and a lower frequency residual (the smooth image). We start with a gaussian convolution profile with standard deviation 2.0 pixels. The residual is then smoothed at twice the standard deviation and the process repeated to give 4 images with a doubling of the smoothing factor between each. Summation of the frequency component images will return us to the original.

The lowest frequency component (the final smoothed image) is blended linearly across the whole range of (α, β) as with standard morphing method. The highest frequency, on the other hand, uses the step transition approach. The intermediate frequencies transition over intermediate intervals. The lower of the intermediate frequencies transitions linearly over half the interval (α, β between 0.25 and 0.75) and the higher intermediate frequency transitions linearly over one quarter the interval (α, β between 0.375 and 0.625).

Figure 7: Standard blending on the left; frequency controlled blending on the right

This approach maintains high frequency components (and hence reduces blurring) while not introducing unsightly transitions (spatial or intensity). Figure 7 shows how face detail is preserved when using frequency controlled blending. Both images were obtained using interpolation parameters $\alpha=\beta=0.4$.

8. Augmenting virtual worlds

We have experimented at combining virtual reality and video sprites using the viewpoint parameters extracted at the start to provide the glue between the geometry of the virtual world and the interpolation parameters of the sprite. Essentially we scale the sprite and position it within our virtual 3D world (e.g. anchoring it to one or two points in the scene). As we update our viewpoint with respect to the 3D world we scale the sprite to mirror simple motion in Z and interpret changes in orientation according to the image interpolation parameters they imply.

Figure 8 shows the simple geometry used in our experiments to relate the morph parameters (α, β) of the video sprite to the orientation parameters of the 3D world. Given the baseline *A* between the optical centers of cameras 0 and 1 and a viewing distance of *D* then from simple trigonometry (assuming symmetric viewing conditions) we approximate the overall rotation of the subject between these cameras as

$$\theta = 2\tan^{-1}(A/2D)$$

We then relate α as a proportion the rotation θ about the axis perpendicular to the horizontal base plane.

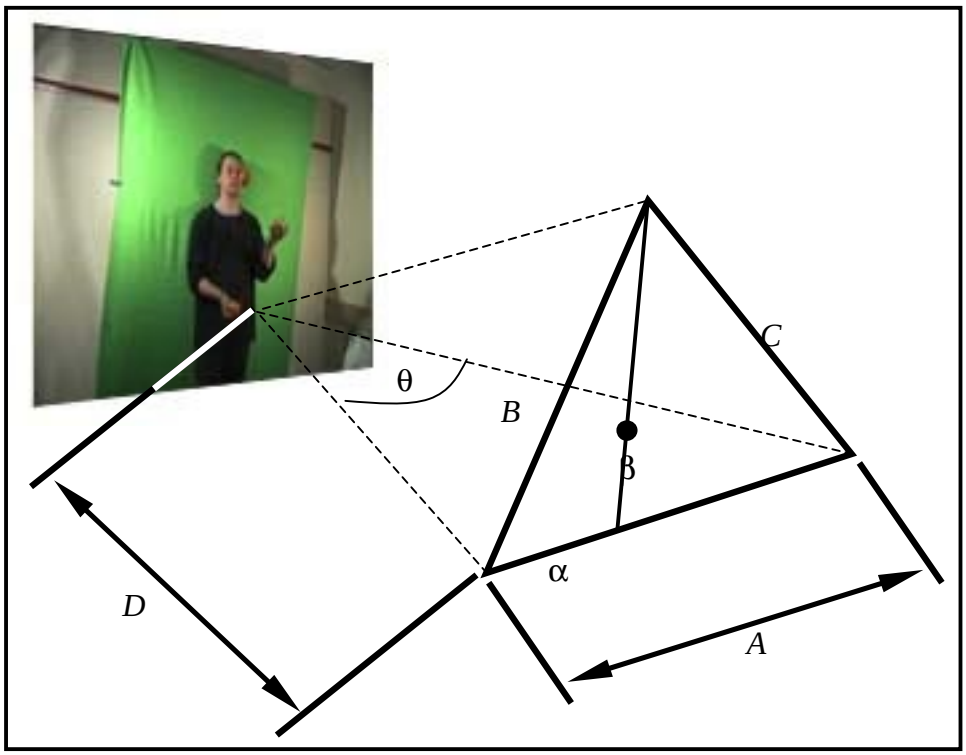

Figure 8: Estimating Orientation Parameters.

We then are able to relate the parameter β as a proportion of the overall rotation about the axis perpendicular to a plane that passes through the line between the optical centre of camera 2 and the point on the baseline implied by α (this axis is simple to compute but is algebraicly messy). The magnitude of the overall rotation about this axis is given by:

$$\varphi = \tan^{-1}\left(\sqrt{\alpha A^2(\alpha-1) + B^2(1-\alpha) + \alpha C^2}\,/D\right)$$

The results of a side to side motion with respect to a juggling sprite are shown in figures 9 and 10 for 9 successive frames extracted from a 14 second video sequence. The frames in figure 9 show views of the sprite video at approximately 0.1 degree intervals reconstructed at approximately the mid point between cameras 0 and 1. The congruent set in figure 10 show the same sprite sequence superimposed against a computer generated 3D backdrop.

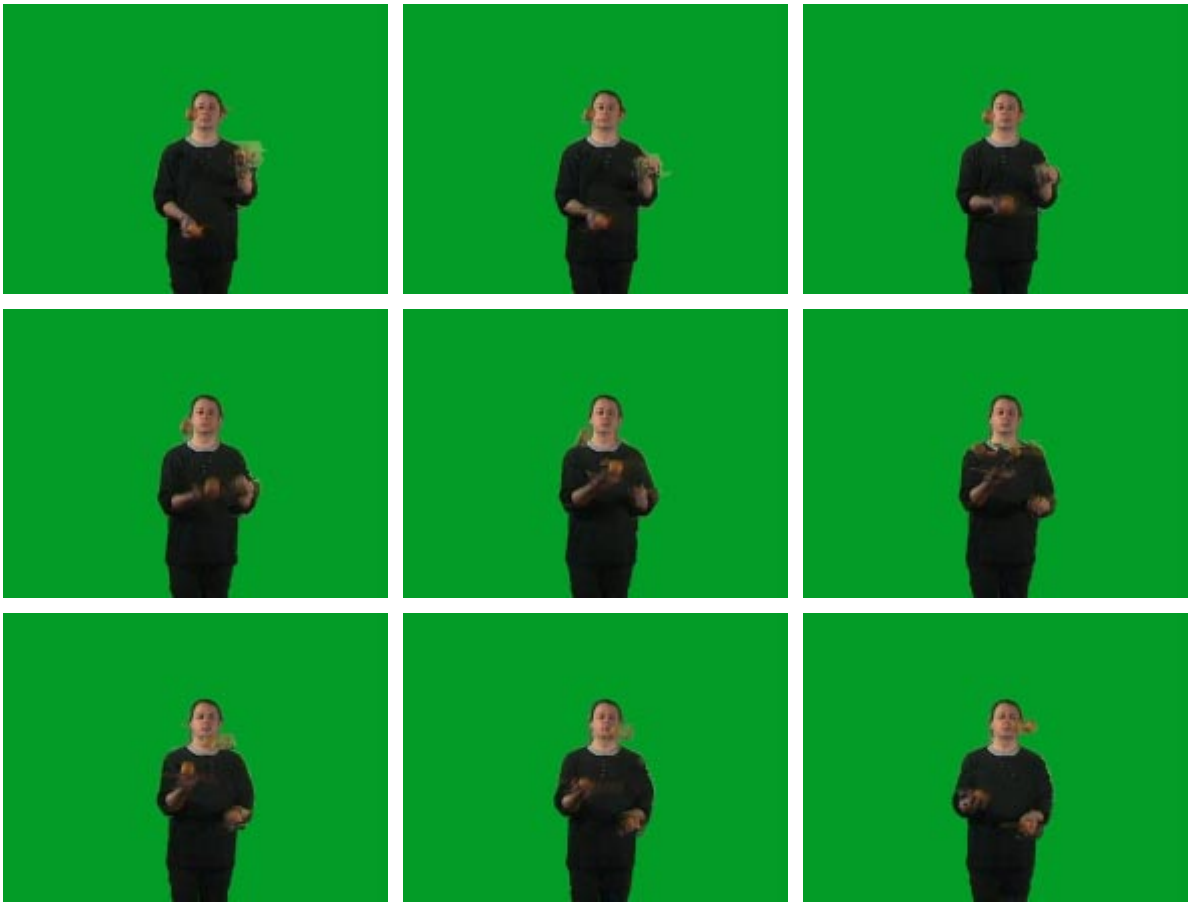

Figure 9: Synthesised Video Sprite Frames

9. Summary

Image-based rendering has become a viable alternative to both 3D graphic modelling and rendering on the one hand and the extraction of full range data and associated texture maps on the other. Its potential for application covers diverse areas such as photo-realistic immersive environment creation,

entertainment and even graphics HW architectures themselves. Against this background, this paper presented a method for the automatic generation of a new type of image based graphic object, which we call a 3D video sprite. It lies part way between the traditional video layer technology used in the digital video post-production industry and a full 3D graphic object. As such it would seem to form a natural component to be implemented within the emerging MPEG 4 standard. 3D video sprites raise the potential to create video objects with which it is possible to interact more fruitfully and the myriad of applications that this presents.

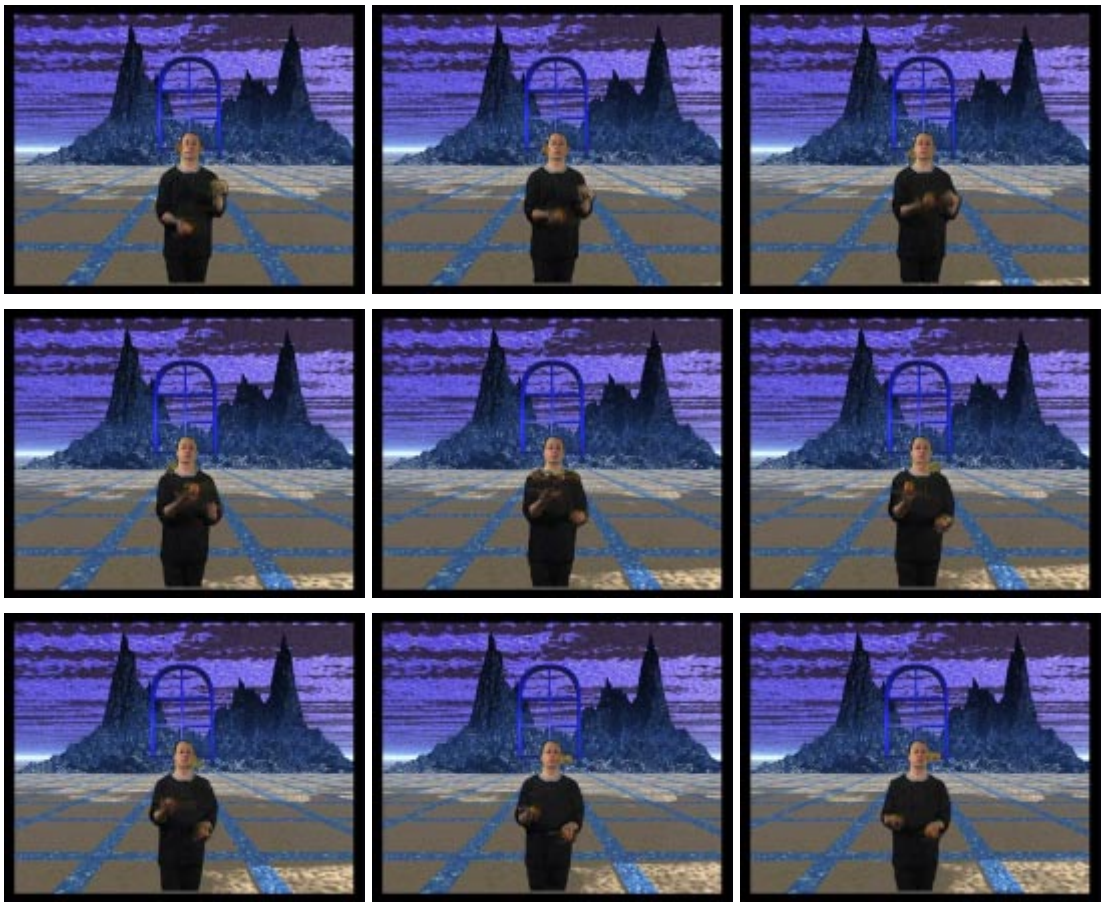

Figure 10: Composite Video Sprite and 3D Graphic

I While for some applications it would be possible to derive 3D video sprites from multiple video streams by employing manual mark up methods, we feel that only through the availability of fully automatic methods of image matching will 3D video sprite technology be really viable. Because of the well-known problem with getting dense (even using edges as we did) correspondences, our method

works best if the baseline between the cameras is reasonably small with respect to the viewing distance, as this improves the quality of stereo matching and reduces the number and degree of occlusions. While this means that the degree of interaction with the video sprite is currently rather limited it has allowed us to demonstrate the potential of the approach. Improvements in image matching technology and the potential for increasing the number and coverage of the video data would both help to improve the effectiveness of the approach.

The image interpolation technique we use has a number of advantages, notably it uses a sketch interpolation and colouring technique that makes it possible to apply the method to three or more images and achieve fast rendering (a key factor if special HW is not available). The method has been successfully applied to video sequences of animated objects (people). Each triplet of corresponding frames is processed afresh to recover matched edge strings. The whole sequence is then played and the viewpoint with respect to it changed in conjunction with a computer generated 3D graphic backdrop.

We are currently investigating better edge matching strategies, in particular investigating ways of exploiting temporal consistency constraints (edge tracking) that could be used to improve quality and matching speed, and ways of smoothly switching between image triples to achieve extended navigable areas. We also recognise the need to achieve improved compression of the multiple video streams by exploiting the redundancy of image data that exists between matching image regions of corresponding frames. It is also important to minimise the not inconsiderable overhead of the morphing channel through a suitable representation of the matched edge strings.

References

1. Burt, P.J. & Adelson, E.H., "A Multiresolution Spline with Application to Image Mosaics", *ACM Trans. Graphics*, Vol 2, No. 4, 217-236, 1983.
2. Chen, S.E., "QuickTime® VR- An Image-Based Approach to Virtual Environment Navigation", *Proc. SIGGRAPH 95*. In *Computer Graphics*, pp29-38, 1995.
3. Chen, S.E. and Williams, L. "View interpolation for image synthesis", *Proc. SIGGRAPH 93*. In *Computer Graphics*, 279-288, 1993.

4. Greene, N., “Environment Mapping and Other Applications of Word Projections”, *IEEE Computer Graphics and Applications*, Vol 6, No 11, pp 21-29, 1986.
5. Gortler, S.J, Grzeszczuk, Szeliski, R. & Cohen, M.F., “The Lumigraph”, *SIGGRAPH 96, In Computer Graphics*, pp 31-42, 1996.
6. Kanade, T., Narayanan, P.J. & Rander, P.W., “Virtualised Reality: Concepts and Early Results”, *Proc. IEEE Workshop on Representation of Visual Scenes*, pp 69-76, 1995.
7. Laveau, S & Faugeras, O., “3D Scene Representation as a Collection of Images and Fundamental Matrices”, INRIA Tech Report 2205, February 1994.
8. Nayer, S.K., “Catapiopetric Omnidirectional Camera”, *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pp 482-488, 1997
9. By the authors, “Automatically Synthesising Virtual Viewpoint by Trinocular Image Interpolation”, *Submitted to IEEE CVPR 1998*.
10. Seitz, S.M. & Dyer, C.R., “Physically-valid view synthesis by image interpolation”, *In Proc. IEEE Workshop on Representation of Visual Scenes*, pp 18-25, 1995
11. Shapiro, L.S., *Affine Analysis of Image Sequences*, Cambridge University Press, 1995.
12. Szeliski, R., “Video Mosaics for Virtual Environments”, *IEEE Computer Graphics and Applications*, 22-30, March 1996.
13. Ullman, S. & Basri, R., “Recognition by Linear Combinations of Models”, *IEEE Trans. PAMI*, Vol 13, No 10, pp 992-1006, 1991.
14. Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, 1990.
15. Xu, G. & Zhang, Z., *Epipolar Geometry in Stereo, Motion and Object Recognition (A Unified Approach)*, Kluwer Academic Press, 1996.
16. Zhang, Z. & Deriche, R., “A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry”, INRIA Tech. Rep. 2273, 1994.