3D Display: Synthetic Image Generation and Visual Effect Simulation

T. A. Theoharis[*], A. R. L. Travis[†] and N. E. Wiseman[‡]

Abstract

Two viewing models are presented for a promising type of three dimensional display; they are based on parallel oblique and perspective oblique projections respectively. A detailed simulation compares the quality of the images that will be produced by each projection type and points out potential problems. The time necessary to alter the image of the three dimensional display will be comparable to that of conventional displays by the use of a simple parallel processing scheme.

1. Introduction

Two dimensional (2D) images produced by current displays contain a number of cues which help the viewer to perceive the third dimension; perspective, hidden surface elimination, shadows and varying object brightness with distance from the viewpoint are some of these cues.

Two fundamental cues can not be provided by 2D images however. These are the stereo effect obtained by each of the two eyes seeing a different image and the change of the image as the viewer moves his head.

A number of devices have been built in order to provide the two missing cues. Varifocal mirrors, stereo pairs and holograms are among the techniques that have been tried; surveys and short descriptions of these and other techniques can be found in Hodges et al.[1] and Collender[2]. The main disadvantages of most of these techniques are that they are hard, hence slow, to produce by computation (e.g. the production of a hologram is far from real-time), have a poor image quality (e.g. varifocal mirrors) or are inconvenient to use (e.g. stereo glasses must be worn).

We believe that future computer displays will use techniques which allow a three dimensional (3D) image to be updated in real-time, have a good quality 3D image and require no special equipment to be worn by the viewer. A technique which promises to possess the above qualities is being developed by one of us (ARLT). The logical operation of the technique is described in section 2. Section 3 deals with two viewing models for the 3D display and outlines a simple parallel processing scheme. Finally section 4 describes a simulation of the images that the 3D display will produce. A description of the 3D display hardware can be found in Travis[3] and Travis and Lang[4].

2. The 3D Display: Logical Operation

When an observer looks at a solid object, what he effectively receives are two 2D pictures formed on the back of his retina. His perception is of a 3D object consistent with these two views. As he moves round the object, these two pictures change. The aim of the display described here is to project a group of pictures in such a way that whatever his position the observer will see the same pictures as if he were looking at the original object.

Imagine photographing a series of views of a solid object from a series of positions at successive incremental angles to the central axis and all in the horizontal plane. These *P* views are stored in *P* frame buffers in a frame store where they comprise the image.

The views are transferred one by one to a conventional two dimensional display where each is briefly presented. Simultaneously an optical apparatus on the front of the display controls the optical output in such a way that the picture displayed is visible from a single (but variable) direction - the action of this apparatus can be likened to a set of vertical parallel shutters or ''venetian'' blinds.

The optical apparatus comprises an imaging lens, a liquid crystal display, and a collimating lens. The optical process is most easily thought of as a combination of two processes. Considering firstly the imaging lens and the CRT on their own, the imaging lens produces an image of the CRT picture in a certain plane (Figure 1).

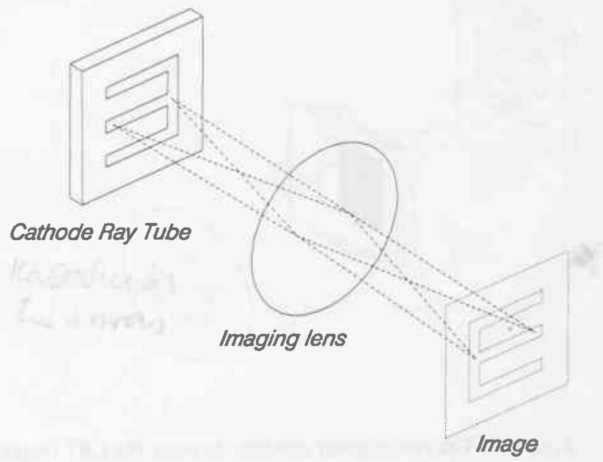

Figure 1. A lens forms an image of the CRT picture

[*] St Catharine's College
University of Cambridge, U.K.
[†] Engineering Department
University of Cambridge, U.K.
[‡] Computer Laboratory
University of Cambridge, U.K.

Secondly considering the liquid crystal display and the collimating lens on their own (Figure 2), the liquid crystal display is placed in the focal plane of the collimating lens so that any light passing through a single slit of the liquid crystal display is collimated into approximately parallel rays.

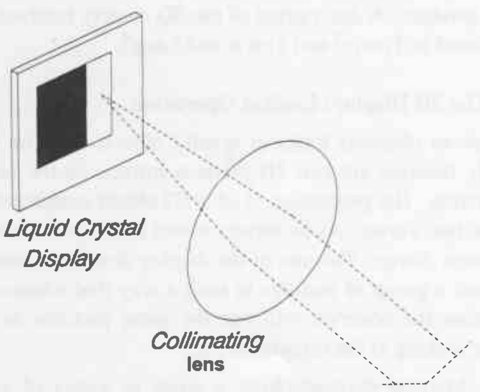

Figure 2. A second lens confines rays from LCD slit to one direction

The two systems are combined into a single system in such a way that the collimating lens is placed in the same plane as that where the imaging lens produces an image of the CRT (Figure 3). The result is that a picture of the CRT appears on the collimating lens as if it were a screen, but the liquid crystal display determines the direction of light rays from the picture so that it can be made visible from a single direction at a time.

During the sequential display of pictures of the solid object the slit on the liquid crystal is moved sideways. Movement of the slit is synchronised with the picture sequence so that each picture is made visible at the same angle from the central axis as that subtended by the camera when it took the original photograph. This sequence of pictures is repeated continuously without regard to the position of the viewer. Because the liquid crystal display used to perform the shuttering operation is fast, each picture is made visible to its respective point of view sufficiently often for there to be no apparent flicker.

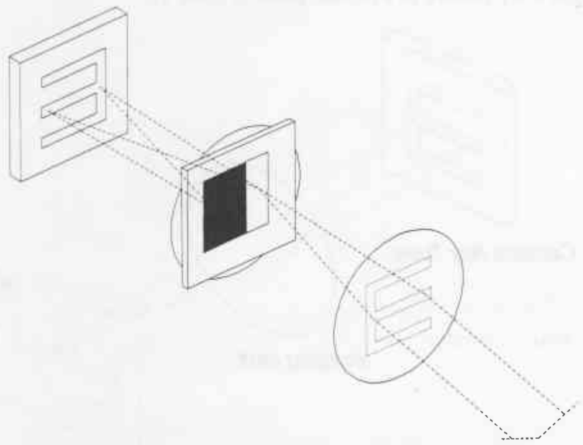

Figure 3. The two systems combine to make the CRT image visible from a single direction

The result is a display showing a static parallax image similar to those found in laterally multiplexed (or "stereographic") holograms.

While the liquid crystal display has been been described as being placed in the focal plane of the collimating lens, there is the alternative of bringing the LCD closer to the collimating lens. The effect is that the rays of light from each picture are no longer parallel but convergent. It is as if the venetian blinds mentioned earlier have a continuous change of angle instead of being parallel. Section 3 considers this further.

3. The Viewing Model

The 3D display provides a 3D view of a static 3D model; each of the *P* images[†] of the 3D model is stored in a different frame buffer and is produced using a modified graphics output pipeline. (A conventional graphics output pipeline[5] produces a 2D image from a 3D mathematical model and comprises such operations as coordinate transforms, clipping, normal perspective projection, hidden surface elimination and smooth shading.) The test object (the Utah

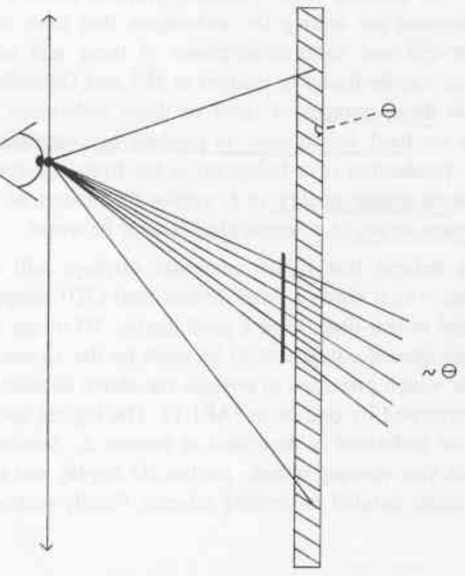

Figure 4. Parallel shutters (top view)

[†]In the rest of this paper *image* will refer to the 2D image resulting from a projection of a 3D model onto the screen plane.

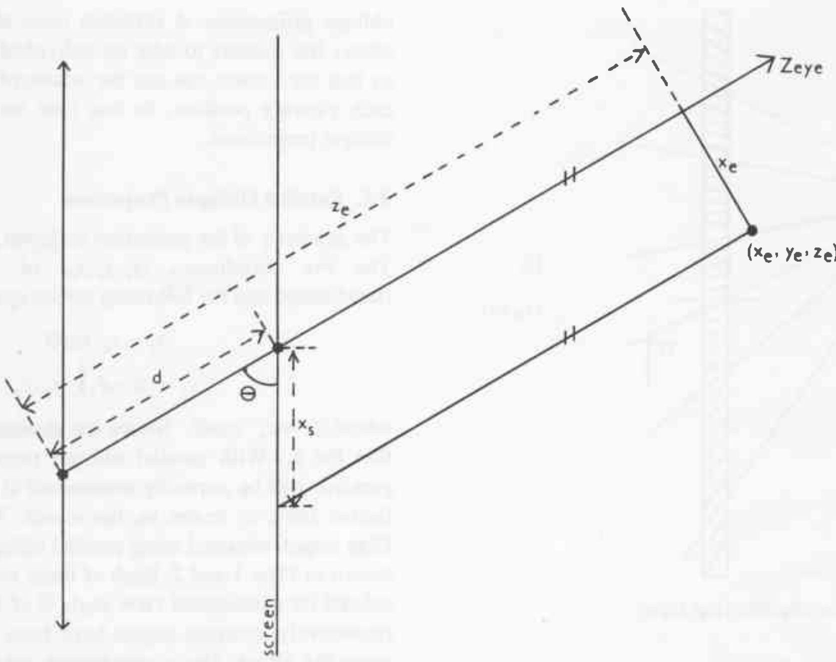

Figure 5. Parallel oblique projection: geometry of x_s

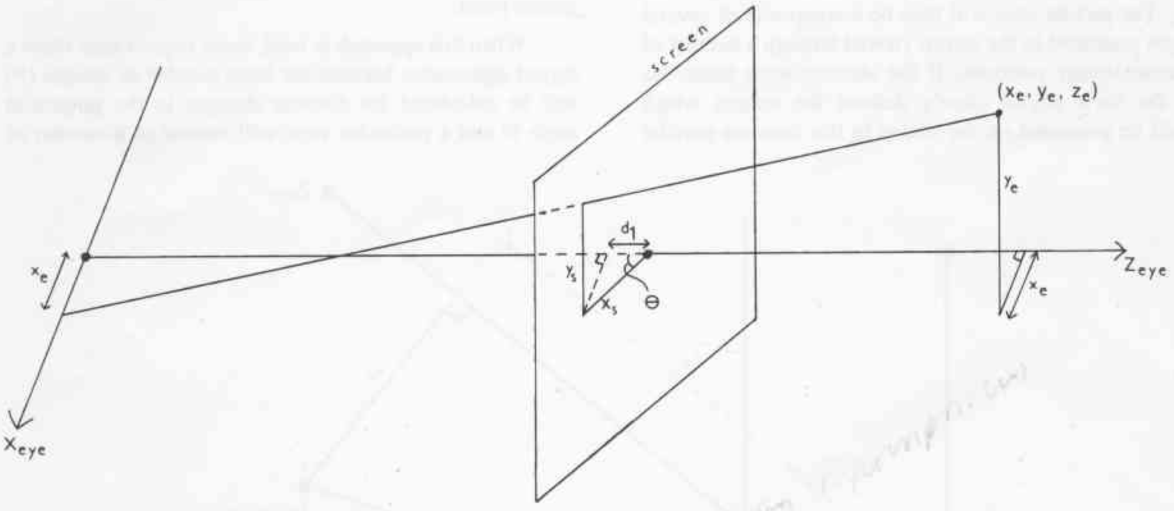

Figure 6. Parallel oblique projection: geometry of y_s

teapot) was a polygon mesh and has been smoothly shaded using the Phong method[6]. Each of the *P* images of the 3D model is produced from a different viewpoint. The *P* viewpoints lie on the horizontal line which is parallel to the screen plane and which is contained in the horizontal plane through the screen centre, see Figure 5 or 8. The angle between the line segments joining adjacent viewpoints to the screen centre is constant and there is an equal number of viewpoints in each of the two halfspaces defined by the vertical plane through the screen centre. The direction of view is defined by the directed line segment from a viewpoint to the screen centre; in the case of parallel projections this line segment defines the direction of projection.

Two alternative projections were considered. These are *parallel oblique* and *perspective oblique* and they place different requirements on the 3D display hardware as will

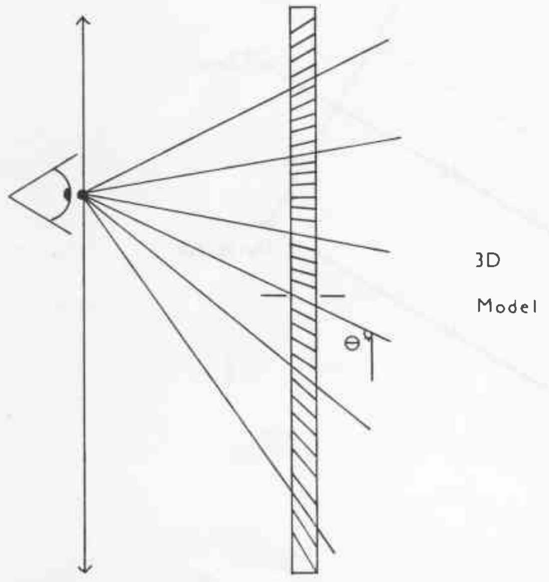

Figure 7. Perspective shutters (top view)

be shown. In Figure 4 all shutters are shown rotated to the same angle from which it is clear that, at the viewpoint shown, the eye will see through some range of shutters only. The picture seen will thus be a composite of several images presented to the screen viewed through a number of different shutter positions. If the shutters were numerous and the view angles closely defined the images which should be presented on the screen in this case are parallel oblique projections. A different case, shown in Figure 7, allows the shutters to take up individually different angles so that the viewer can see the whole of a single image at each viewing position. In this case we want perspective oblique projections.

3.1. Parallel Oblique Projection

The geometry of the projection is shown in Figures 5 and 6. The eye coordinates (x_e, y_e, z_e) of a point will be transformed into the following screen space coordinates:

$$x_s = x_e / \sin\Theta$$

$$y_s = (d - d_1) \cdot y_e / z_e$$

where $d_1 = x_s \cdot \cos\Theta$. Notice the absence of d in the equation for x_s. With parallel oblique projections, horizontal parallax will be correctly maintained if the viewer moves further from, or nearer to, the screen. Two images of the Utah teapot, obtained using parallel oblique projections, are shown in Plate 1 and 2. Each of these images has been calculated for a horizontal view angle Θ of 70 and 110 degrees respectively; extreme angles have been chosen to demonstrate the effect. The y coordinates are perspectively projected and the combined effect is that there is a projection axis (the horizontal X_{eye} axis of Figure 6) rather than a projection point.

When this approach is used, some objects may show a jagged appearance because the finite number of images (P) will be calculated for discrete changes in the projection angle Θ and a particular view will consist of a number of

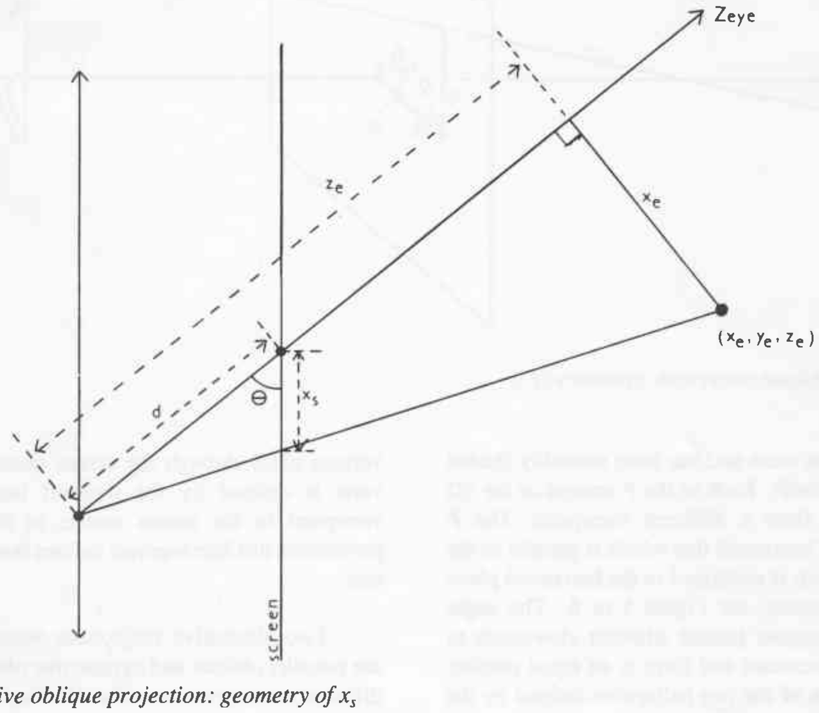

Figure 8. Perspective oblique projection: geometry of x_s

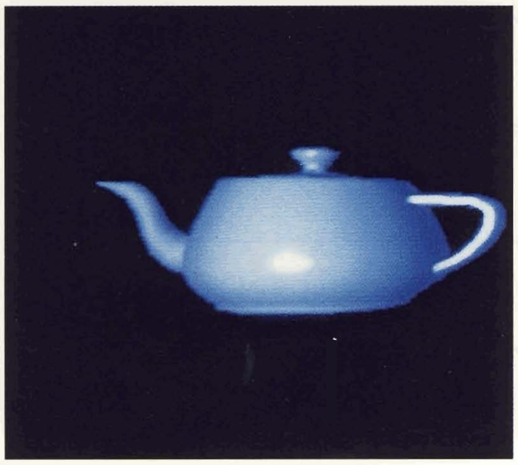
Plate 1. Parallel oblique projection (θ = 70°)

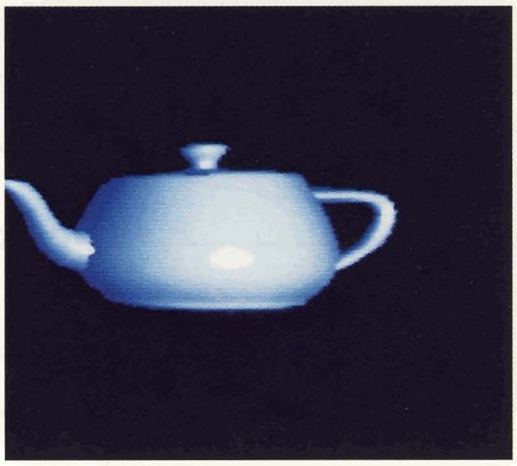
Plate 2. Parallel oblique projection (θ = 110°)

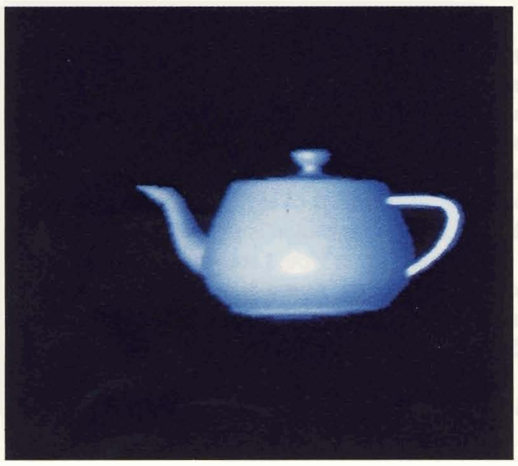
Plate 3. Perspective oblique projection (θ = 70°)

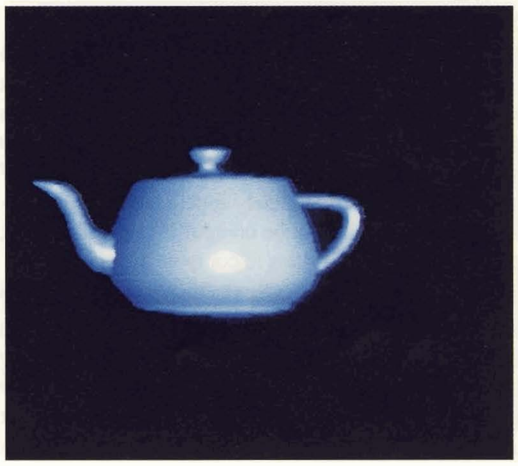
Plate 4. Perspective oblique projection (θ = 110°)

οδοντωτό

vertical strips from different images. For example, for a typical 20 degree field of view the viewer would see parts of 10 of the total number of images if these were spaced by 2 degrees. Of course the jagged appearance will be reduced as the number of images is increased but each extra image requires an extra frame buffer, takes up an extra share of the update bandwidth and reduces the maximum acceptable image decay time.

3.2. Perspective Oblique Projection

In this case the geometry is as shown in Figures 8 and 9. A simple calculation shows that the eye coordinates (x_e, y_e, z_e) of a point will be mapped onto the following screen space coordinates:

$$x_s = d \cdot x_e / (z_e \cdot \sin\Theta + x_e \cdot \cos\Theta)$$

$$y_s = (d - d_1) \cdot y_e / z_e$$

where $d_1 = x_s \cdot \cos\Theta$. Two perspective oblique images of the teapot for horizontal view angles of 70 and 110 degrees are shown in Plate 3 and 4 respectively.

3.3. Rapid 3D Image Generation

The hardware of the 3D display provides a 3D view of a static 3D model. The user is able to inspect it from a number of different angles but will eventually want to see the model from an angle that is not provided by the horizontal parallax effect, or zoom into or out of the model, or even change the relative positions of objects within the model (modelling transformations). To achieve such a change, all of the *P* frame buffers must be updated.

Updating a single frame buffer is a matter of performing the operations of the graphics output pipeline on the object data base using the new viewing (and perhaps

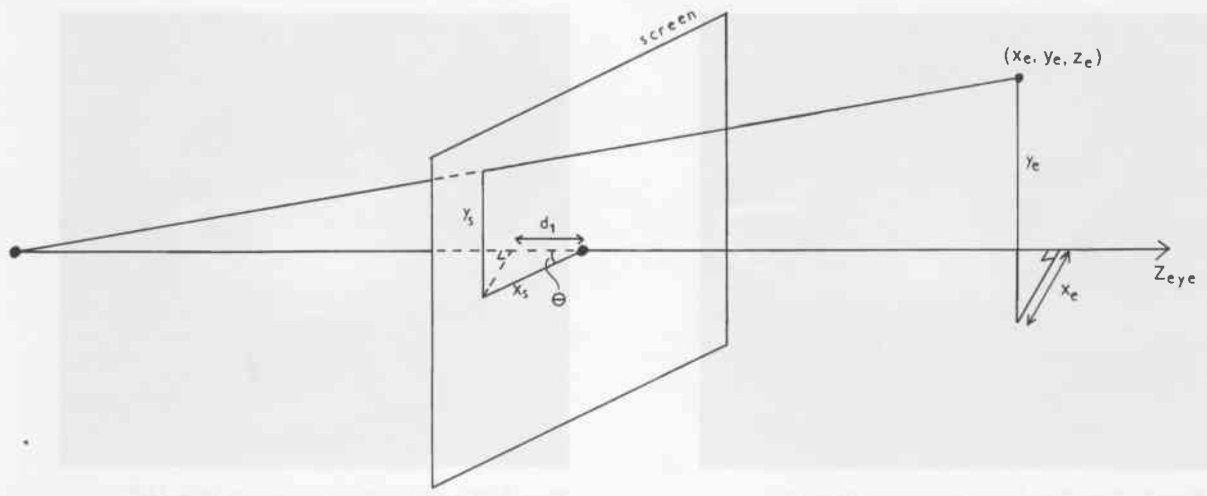

Figure 9. Perspective oblique projection: geometry of y_s

modelling and lighting) parameters. These operations are computationally expensive. Using Phong shading it takes several seconds to perform the above operations on a data base consisting of a few thousand polygons on a T800 floating point transputer[7]. Fortunately the computation of each of the *P* images is completely independent of the others and they can therefore all be computed in parallel using *P* processors in the time it takes to compute a single image.

To this effect, we propose the close-coupling of *P* T800 transputers with each of the *P* frame buffers. This can be achieved by using Video Random Access Memory (VRAM)[8] for the part of the memory of each transputer corresponding to the frame buffer. Each transputer must also have sufficient memory to store the entire object data base plus a copy of the graphics output pipeline code. The transputers must be connected in a sensible manner (e.g. tree) to allow the broadcasting of the viewing parameters. These parameters consist of 3 vectors: *from*, the viewpoint, *at*, the screen centre and *up*, the view up vector. These can be augmented to include lighting and modelling parameters. In any case these parameters only occupy a few tens of bytes and can be broadcast very rapidly through the 10Mbit/sec transputer links. Each transputer must vary the value of the *from* parameter according to the view angle Θ for which it is responsible. Notice that no local communication between the transputers is required.

Our current system consists of 4 T800 transputers hosted by a PC. They do not possess VRAM's and the extraction of the images from the frame buffers is a slow process.

4. Simulation

The 3D display hardware is currently under construction. By producing a computer simulation of the effect of merging the 2D images that the hardware will achieve, we hope to identify the preferred projection type (parallel or perspective), spot potential problems and gain an advance insight into the 3D images that the 3D display will produce. The production of such a simulation requires a detailed knowledge of the workings of the 3D display (section 4.1). The objective of this simulation is to produce the 3D image that a single eyed viewer would see when positioned on an axis perpendicular to the screen at a distance *d* away from the screen. Thus we are only interested in the quality of a static 3D image (the 3D effect resulting from the movement of the viewer's head and the disparity between his two eyes is difficult to simulate).

4.1. Details of 3D Display Function

For simplicity of description, it was mentioned in section 2 that the shutters (angular light filters) could take on one of *P* orientations in synchrony with the display of each of the *P* images. What actually happens is a bit more complicated. The horizontal angular field from which a pixel may be viewed is divided into $n*P$ arcs as shown in Figure 10. The arcs emanating from each pixel lie on the horizontal plane which includes the screen row that the pixel is on and is perpendicular to the screen. We talk of an arc *opening* or *closing* at the times when the corresponding angular light filter becomes transparent or opaque for the range of angles that the arc includes. The hardware being designed allows arcs to open and close independently of each other. Each arc stays open for the time *T* that it takes to refresh one of the *P* images onto the screen; subsequent arcs open at intervals of T/n (Figure 10). Thus there is an average of *n* adjacent arcs open at any time. It is believed that the provision of the $n*P$ arcs will provide a smooth transition between temporally adjacent images as the viewer changes his horizontal position.

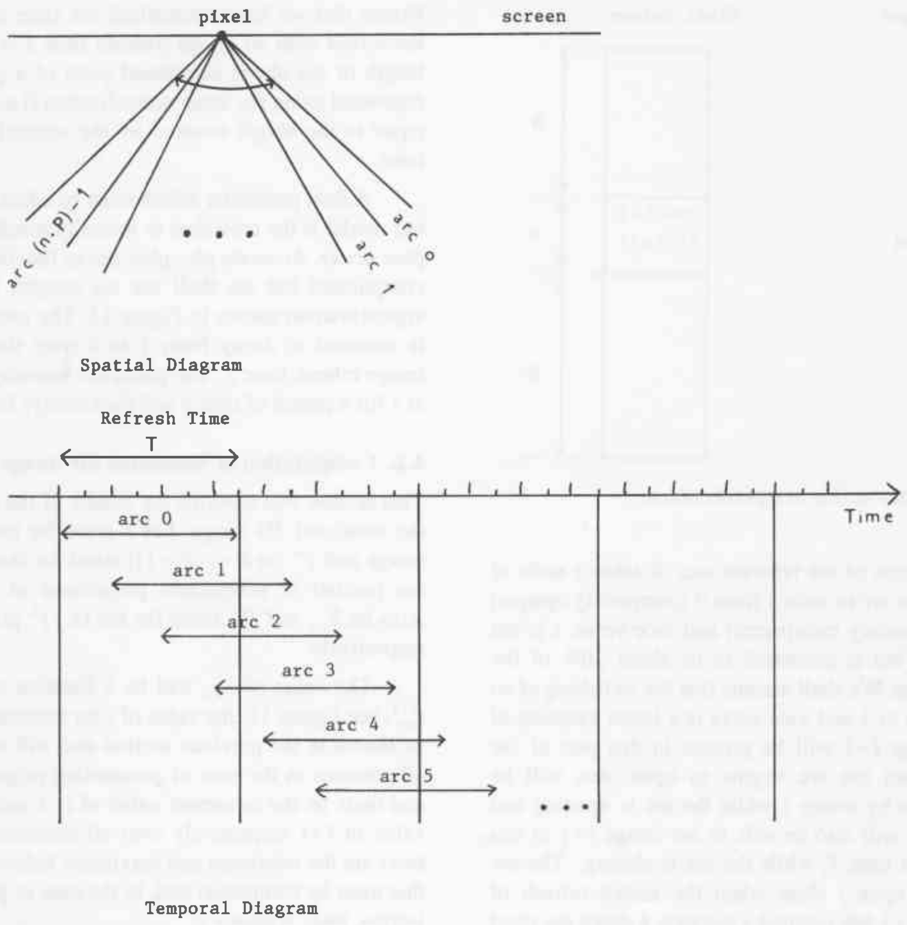

Figure 10. Arcs in space and time

Given a pixel and a viewpoint, it is simple to calculate which of the arcs that emanate from the pixel the viewpoint lies in. Since there is only horizontal parallax the viewpoint will lie in the same arc for all the pixels that belong to the same column. Having calculated the arc α that the viewpoint lies in relative to a pixel column x, it is possible to determine which of the P images is visible at that column. Since there are n arcs per image, the visible image has index:

$$i = \lfloor \alpha / n \rfloor = f(x)$$

for constant n and fixed viewpoint. $\lfloor\ \rfloor$ represents the floor function. Notice that when arc α opens, the refresh of image i will have covered a part of the screen proportional to the value:

$$k = (\alpha \, REM \, n)/n = g(x) \qquad (0 \le k < 1)$$

for constant n and fixed viewpoint. The expression $\alpha \, REM \, n$ represents the remainder of the division α / n when the quotient is $\lfloor \alpha / n \rfloor$. The $(1-k)^{th}$ portion of the screen that has not yet been refreshed with image i will still contain image $i-1$ (which will have deteriorated due to phosphor decay). Also, since an arc stays open for an amount of time equal to a complete screen refresh, the viewer will also see the k^{th} portion of image $i+1$.

The above calculation of i and k holds true in the case of parallel projections. In the case of perspective projections those pixel-arcs that contain the viewpoint are open at the same time and the viewer can therefore only see a single image (as well as parts of its temporal neighbours). In this case i and k have the same value for all pixel columns. Their calculation is based on the position of the viewpoint relative to the centre of the screen.

A pixel column can be divided into 3 parts (see Figure 11). Assuming that the screen is refreshed starting from the top, and that the screen height is equal to 1 then:

- The top a^{th} part of the column, where $a = k$, contains image i when the relevant arc opens. Image $i+1$ will also be seen in this part before the arc closes.
- The next s^{th} part of the column represents the opening

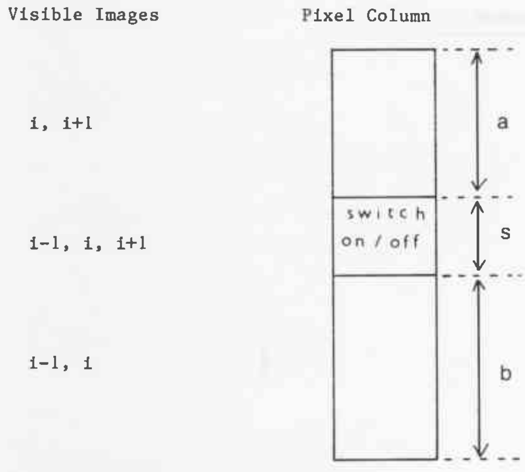

Figure 11. Images visible in a pixel column

/ closing time of the relevant arc. It takes s units of time for the arc to switch from 0 (completely opaque) to 1 (completely transparent) and vice-versa. s is not negligible but is estimated to be about 20% of the refresh time. We shall assume that the switching of an arc from 0 to 1 and vice-versa is a linear function of time. Image $i-1$ will be present in this part of the screen when the arc begins to open, this will be overwritten by image i while the arc is opening and the viewer will also be able to see image $i+1$ in this part after a time T, while the arc is closing. The arc begins to open / close when the screen refresh of image i / $i+1$ has reached a distance k down the pixel column. Thus images $i-1$, i and $i+1$ will all be visible in this part of the column.

- The bottom b^{th} part of the column, where $b = 1-(a+s)$, will contain images $i-1$ and i during the period that the relevant arc is open.

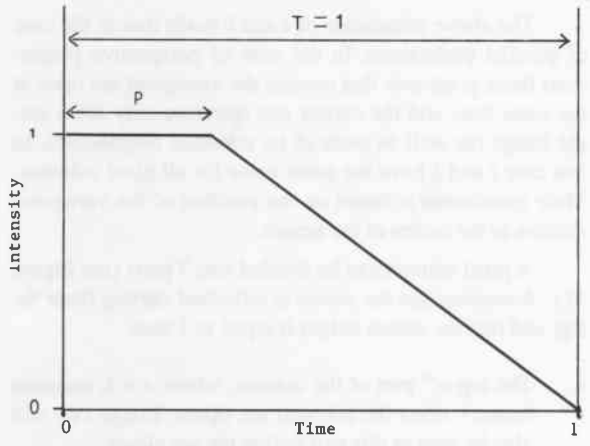

Figure 12. Simple phosphor decay function

Notice that we have normalised the time taken to refresh the screen with an image (refresh time $T = 1$) and that the length of the above mentioned parts of a pixel column is expressed using the same normalisation (i.e. a length of 1 is equal to the length covered by the vertical screen resolution).

A final parameter which must be taken into account in our model is the reduction in intensity resulting from phosphor decay. Accurate phosphor decay functions can be very complicated but we shall use the simple, but reasonable, approximation shown in Figure 12. The phosphor intensity is assumed to decay from 1 to 0 over the length of the image refresh time T. The phosphor intensity initially stays at 1 for a period of time p and then decays linearly to 0.

4.2. Computation of Simulated 3D Image

This section will describe the details of the computation of the simulated 3D image. Let S stand for the simulated 3D image and I^m $(m: 0 \cdots P-1))$ stand for the P images that are parallel or perspective projections of the 3D world. Also let $S_{x,y}$ and $I^m_{x,y}$ stand for the $(x,y)^{th}$ pixel of S and I^m respectively.

The value of $S_{x,y}$ will be a function of $I^{i-1}_{x,y}$, $I^{i}_{x,y}$ and $I^{i+1}_{x,y}$, see Figure 11; the value of i for column x is calculated as shown in the previous section and will be the same for all columns in the case of perspective projection. Let *imin* and *imax* be the minimum value of $i-1$ and the maximum value of $i+1$ respectively over all columns (i.e. *imin* and *imax* are the minimum and maximum indices of the images that must be computed) and, in the case of perspective projection, $imax = imin + 2$.

The computation of images I^{imin} to I^{imax} takes place and the appropriate columns of each of these images are copied into three frame buffers, $FB.M1$, FB and $FB.P1$, so that:

$$\left.\begin{aligned} FB.M1_{x,*} &= I^{f(x)-1}_{x,*} \\ FB_{x,*} &= I^{f(x)}_{x,*} \\ FB.P1_{x,*} &= I^{f(x)+1}_{x,*} \end{aligned}\right| \quad x: 0 \cdots (\textit{horizontal resolution}) - 1$$

where $*$ in the row index indicates that the whole column is selected and $f(x)$ is the function of the pixel column x that gives the image index (see section 4.1). In other words each of the three frame buffers $FB.M1$, FB and $FB.P1$ contains in its pixel column x, column x of the image whose index is $f(x)-1$, $f(x)$ and $f(x)+1$ respectively; the images with these indices are the ones that are (partly) visible at column x of the 3D display from the given viewpoint. The reason for using the above 3 frame buffers is to avoid the unnecessary storage cost of allocating a frame buffer to each of the computed images (these are likely to be many more than 3 in the case of parallel projections).

The next step in obtaining the simulation is to combine $FB.M1$, FB and $FB.P1$ into the simulated image S.

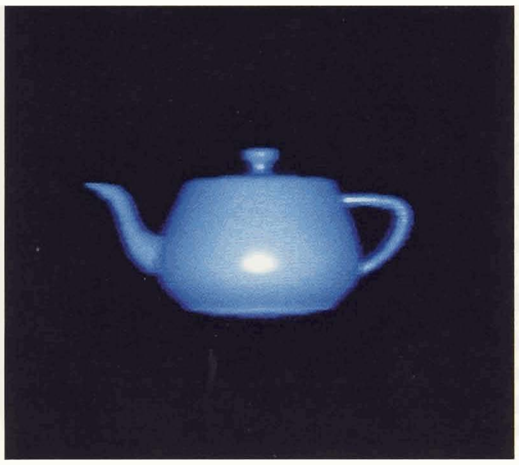
Plate 5. 3D image simulation: perspective proj., size = x, distance = y

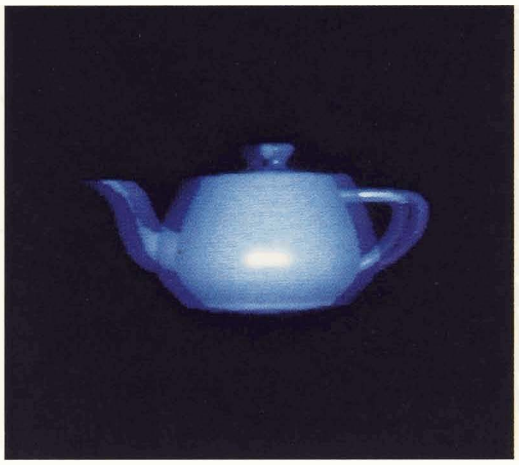
Plate 6. 3D image simulation: perspective proj., size = $2x$, distance = $2y$

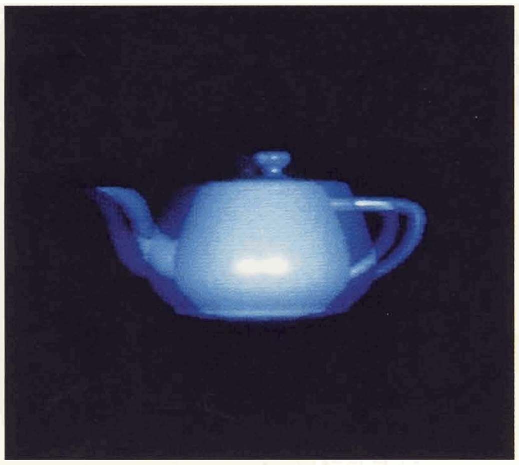
Plate 7. 3D image simulation: perspective proj., size = $4x$, distance = $4y$

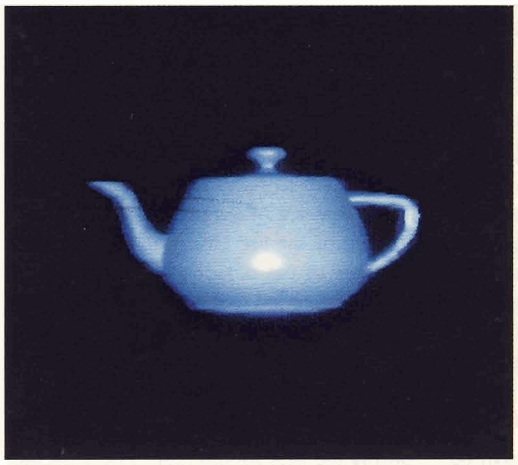
Plate 8. 3D image simulation: parallel proj., size = x, distance = y

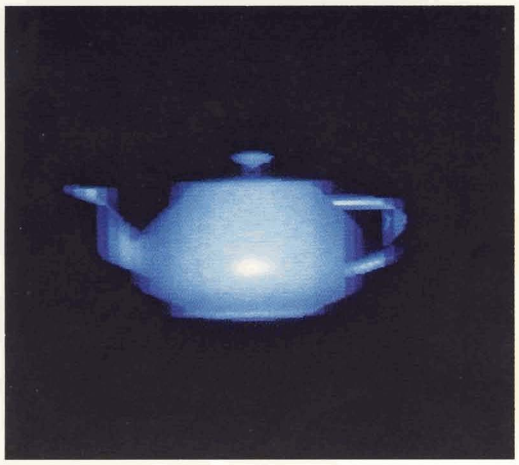
Plate 9. 3D image simulation: parallel proj., size = $2x$, distance = $2y$

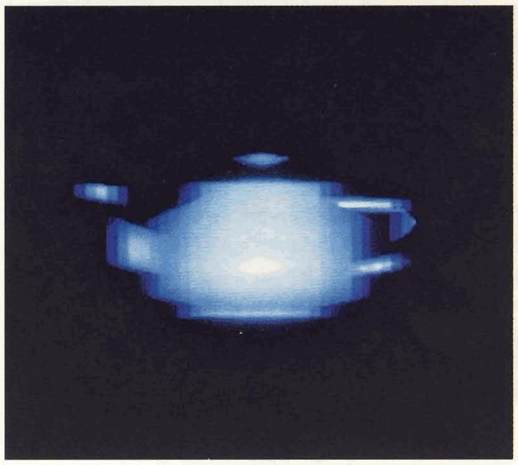
Plate 10. 3D image simulation: parallel proj., size = $4x$, distance = $4y$

Let us again consider a particular pixel column x. $S_{x,y}$ will be a combination of ($FB_{x,y}$, $FB.P\,1_{x,y}$) or ($FB.M\,1_{x,y}$, $FB_{x,y}$, $FB.P\,1_{x,y}$) or ($FB.M\,1_{x,y}$, $FB_{x,y}$) depending on the row index y, see Figure 11. Suppose that $y.norm < a$ (y is normalised to a value between 0 and 1 by dividing it by the value of the vertical resolution). In this case $S_{x,y}$ is a combination of $FB_{x,y}$ and $FB.P\,1_{x,y}$. The latter two values must be combined in proportion to:

- The amount of time that each value is displayed during the time interval T that the relevant arc is open.
- The opacity of the shutters during the display of each value.
- The relevant phosphor intensity.

We can partition the time interval T into subintervals which add up to 1 (T normalised) and during which a single colour value is displayed ($FB_{x,y}$ or $FB.P\,1_{x,y}$). If we then multiply the colour value for each interval by 3 weights (one for each of the above 3 factors) and then sum the results, we shall get the value for $S_{x,y}$. The weights are all normalised to the range (0..1).

We shall next determine the values of the weights for the case under consideration i.e. $y.norm<a$. The time interval T during which the arc is open is partitioned into 4 subintervals; a fifth subinterval of length s is also included in order to take account of the arc closing time. Subintervals are defined in terms of the position of the refresh on the screen; the refresh is assumed to proceed top-down, see Figure 11. The duration of the first subinterval is s (s:0..1). During this time the arc switches from completely opaque (0) to completely transparent (1); since the variation in opaqueness is a linear function of time, the weight for this factor will be $\frac{1}{2}$ during the first subinterval. The third factor is intensity reduction due to phosphor decay. The intensity function of Figure 12 is used to determine the average phosphor intensity of pixel (x,y) during the subinterval $a..(a+s)$. Time 0 of the phosphor decay function is $y.norm$, the instant when pixel (x,y) was refreshed. Let the weight derived for this factor be w_1. Then the contribution of the first subinterval to the colour value of pixel (x,y) is $\frac{1}{2} \cdot s \cdot w_1 \cdot FB_{x,y}$.

There are 4 more subintervals; two are of length b and $y.norm$ respectively and the relevant colour value is $FB_{x,y}$ while the other two are of length $a - y.norm$ and s respectively and the relevant colour value is $FB.P\,1_{x,y}$. (The last subinterval, s, corresponds to the time it takes for the arc to close.) With the aid of Figures 11 and 12 it is not difficult to work out that the contributions of the last 4 subintervals are:

$$b \cdot w_2 \cdot FB_{x,y}$$

$$y.norm \cdot w_3 \cdot FB_{x,y}$$

$$(a - y.norm) \cdot w_4 \cdot FB.P\,1_{x,y}$$

$$\frac{1}{2} \cdot s \cdot w_5 \cdot FB.P\,1_{x,y}$$

($w_1..w_5$ are the weights derived from the phosphor decay function). The sum of all 5 contributions is $S_{x,y}$, the grey scale colour value of pixel (x,y) in the simulated 3D image.

The cases when $a \le y.norm < (a+s)$ and $y.norm \ge (a+s)$ are dealt with similarly.

4.3. Simulation Results

The results of the simulation are shown in Plates 5-10. The simulated 3D images of Plates 5-7 have been calculated using perspective oblique projections while in the images of Plates 8-10 parallel oblique projections were used. The teapot of Plates 5 and 8 is very close to the screen (projection) plane. In Plates 6 and 9 the teapot is twice as far as the original while in Plates 7 and 10 the teapot is four times further than the original.

Ideally, all the Figures would be identical. However, the disparity between adjacent images of the object increases as the object gets further away from the screen

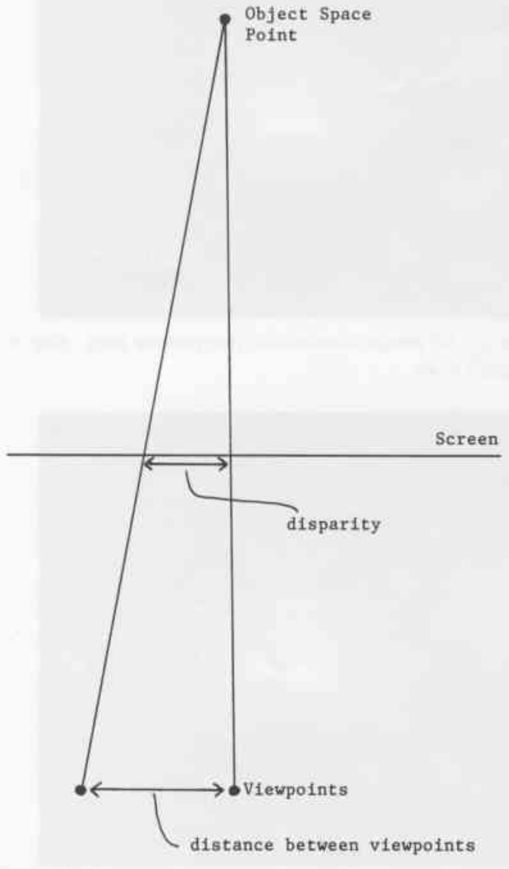

Figure 13. Perspective projection: object disparity limited by distance between viewpoints

plane. This disparity manifests itself in different ways in the cases of perspective and parallel projections.

Consider the perspective projection first. The viewer can see the merging of 3 images: I^{i-1}, I^{i} and I^{i+1}. The lateral disparity in the position of the object in these images is greater the further away the object is, see Plates 5-7. However, this disparity is limited by the distance between the viewpoints that have been used in calculating the respective images, see Figure 13. Thus by reducing the distance between adjacent viewpoints it is possible to reduce the maximum value of the disparity. The distance between adjacent viewpoints can be reduced either by reducing the horizontal range over which the 3D effect is provided or by increasing the number of images P (as technology allows).

In the case of the parallel projection the viewer will see a 3D image made up of groups of pixel columns, each group being part of a different image (parallel projection) of the object. However, the parallel projector for a particular pixel will not usually be exactly collinear with the line through the viewpoint and the pixel (which would be the correct projector, see Figure 14). Therefore there is an error in the projection which is proportional to the distance of the object from the screen plane. Furthermore the further away the object is, the larger the gaps between groups of parallel projectors will be (see Figure 14) hence the smaller the amount of information about the shape and colour of the object that is likely to be conveyed to the viewer. It is for this reason that the teapot loses its ''teapotness'' as it gets further away in Plates 8-10.

5. Further Work

The error that arises when parallel projections are used gives rise to a disturbing object appearance if the latter is sufficiently far away. Small objects may even be invisible if they fall within one of the gaps of Figure 14. Could this error be compensated for by suitable object transformations? Would these transformations maintain the 3D object realism from all the viewpoints? How expensive would their computation be? For how many images P is the effect so diminished as to be acceptable?

Animators have been looking into techniques for exploiting frame-to-frame coherence for some time (frame-to-frame coherence is the similarity between adjacent frames of an animation). A substantial effort has been directed at reducing the computational cost of hidden surface elimination[9, 10] and some recent approximate techniques are aimed at reducing the cost of ray-tracing animation sequences[11, 12]. However, the P projected images of the 3D display offer more scope for exploiting incremental techniques than the frames of an animation for two reasons: first there is a regular difference between the viewpoints of the P images of the 3D display, and second the relative

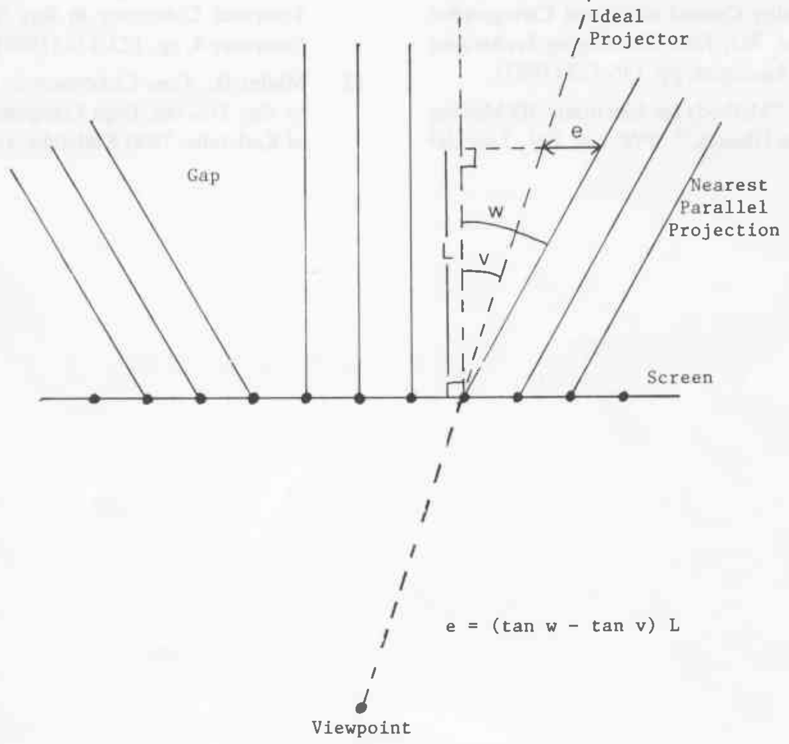

Figure 14. Parallel projection error

positions of the objects in the model are the same for all *P* images (i.e. the modelling transformations are the same for all *P* projections). However, incremental techniques may not be very useful when the *P* images are computed in parallel because data dependencies between the computations of adjacent images will be introduced.

6. Conclusions

We have presented two viewing models for an experimental three dimensional display based on perspective and parallel oblique projections respectively. The parallax effect provided by the parallel projections is not dependent on the viewing distance. A detailed simulation has shown that image quality is better for objects nearer the projection plane but in the case of perspective projections the image deterioration has an upper bound related to the total number of different images presented to the display. With the aid of parallel processing, the time required to change the 3D image will be comparable to the time taken to alter the image of a conventional display.

Acknowledgement

We would like to thank Michael Hearn whose rendering program we modified in order to produce the oblique projections of the teapot and the simulation.

References

1. Hodges L. and D.F. McAllister, "Three-Dimensional Display for Quality Control of Digital Cartographic Data," *SPIE Vol. 761, True 3D Imaging Techniques and Display Technologies*, pp. 146-152 (1987).
2. Collender R.B., "Methods for Electronic 3D Moving Pictures Without Glasses," *SPIE Vol. 761, True 3D Imaging Techniques and Display Technologies*, pp. 2-22 (1987).
3. Travis A.R.L., "An Autostereoscopic 3-D Display," *Applied Optics* **29**(29) (10 Oct 1990).
4. Travis, A.R.L. and S.R. Lang, "A CRT Based Autostereoscopic 3-D Display," in *Eurodisplay'90, Proceedings of SID* (September 25-27 1990).
5. Foley J.D. and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison Wesley (1983).
6. Phong B.T., "Illumination for Computer Generated Pictures," *Comm. ACM* **18**(6), pp. 311-317 (June 1975).
7. INMOS Ltd, *Transputer Reference Manual*, Prentice Hall (1988).
8. Forman S., "Dynamic Video RAM Snaps the Bond Between Memory and Screen Refresh," *Electronic Design*, pp. 117-125 (30 May 1985).
9. Shelley K.L. and D.P. Greenberg, "Path Specification and Path Coherence," *ACM Computer Graphics* **16**(3), pp. 157-166 (July 1982).
10. Hubschman H. and S.W. Zucker, "Frame-to-Frame Coherence and the Hidden Surface Computation: Constraints for a Convex World," *ACM TOG* **1**(2), pp. 129-162 (April 1982).
11. Badt S., "Two Algorithms for Taking Advantage of Temporal Coherence in Ray Tracing," *The Visual Computer* **4**, pp. 123-132 (1988).
12. Muller H., *Time Coherence in Computer Animation by Ray Tracing*, Dept Computer Science, University of Karlsruhe, 7500 Karlsruhe, Germany.