An Architecture For Rapid Stereoscopic Image Generation

Shaun M[c]Cann, Paul Lister

Centre for VLSI and Computer Graphics, University of Sussex, Brighton, BN1 9QT, UK

Abstract

A cost effective architecture for the rasterisation of stereo-imagery based on image derivation is presented. The architecture is a simple and scaleable augmentation of a classic monocular graphics pipeline and integrates stereoscopic photorealistic capability at comparatively little extra cost. Our architecture performs polygon derivation based on purely incremental operations and is well suited to implementation in hardware.

1. Introduction

Conventional computer generated images contain only monocularly encoded depth information, i.e. the left eye and the right eye view the same image. Monocular visual cues can be used to enhance this two-dimensional representation providing the viewer with a greater sense of image depth, however the image is still essentially flat. To produce an image which is truly three dimensional we must incorporate binocular encoding of depth information. This is achieved by providing the left eye and the right eye with computer generated views of a scene that reflect the relative lateral displacement of the two eyes of an observer. Such images are termed stereoscopic images.

2. Efficient Stereoscopic Rasterisation

There are two ways we can generate stereoscopic images using a conventional graphics pipeline — we specifically consider hardware implementations here. First, we might use a single pipeline twice producing first the left-eye view and subsequently the right-eye view, or vice-versa. This method is potentially slow but requires basic hardware resources to implement. Secondly, we can use two graphics pipelines operating in parallel where each pipeline is dedicated to producing the images for each view independently. This approach is faster but requires twice as much hardware to implement.

Ideally we desire a graphics pipeline implementation that is both fast, producing stereoscopic image pairs in parallel, while requiring minimal extra hardware over a single graphics pipeline implementation. To reduce the time taken or hardware resources required to generate a stereoscopic image pair we must take advantage of any visual and computational coherencies that exist between left- and right-eye views. We can produce stereoscopic images significantly more efficiently if we simultaneously generate the left- and right-eye views by deriving one eye view from the other. Here we shall describe and evaluate a stereoscopic rasteriser hardware architecture based on view derivation.

3. Stereoscopic Image Derivation

Most modern graphics systems accelerate the rasterisation of a triangularly tessellated database using scan line based algorithms. Three approaches suitable for scan line based stereoscopic rasterisation have been published; z-shear techniques [10,7]; rotation techniques [6,13,8]; and techniques using two centres of projection [12,1,9].

Using these techniques we can develop equations defining left and right projected x and y coordinates and apply them in two different ways. First, we can use left and right coordinate equations to simply construct a stereoscopic viewing transform [16,4]. Left- and right-eye polygonal projections defined using these viewing transforms are rendered separately using standard (monocular) scan line based algorithms — independently, and serially or in parallel. This is represented by P_1 and P_2 in figure 1. Alternatively, we can construct a parallax relation between projected left and right coordinates — represented by P_3. This parallax relation defines a spatial association between left- and right-eye projected coordinates and thus allows us to derive one view from the other simultaneously during scan conversion, i.e. stereoscopic image derivation.

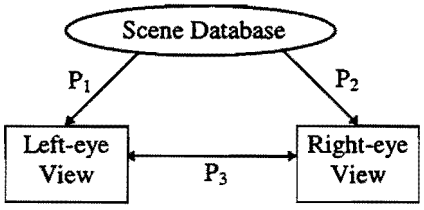


Figure 1 - Stereoscopic Image Generation

We have implemented a simple yet effective derivation technique, the stereoscopic affine mapping (SAM) algorithm, that performs stereoscopic image generation using purely incremental operations which can be readily implemented in hardware — see Appendix for further details.

We shall use the following terminology when referring to stereoscopic projections of a polygon: the *primary polygon* refers to the left- or right-eye polygonal projection with the larger area, and the *secondary polygon* refers to the left- or right-eye polygonal projection with the smaller area.

4. System Architecture

Figure 2 shows an outline of a stereoscopic rendering architecture. Polygonal data from the host processor is passed over the system bus via the host interface. Primary and secondary graphics pipelines rasterise primary and secondary polygons respectively, the secondary pixel stream data being predominantly derived from the primary pixel stream data.

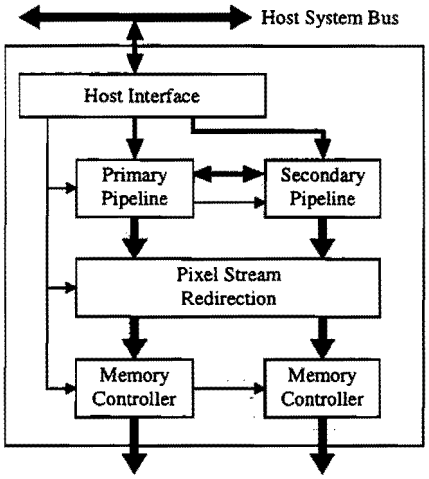


Figure 2 - Stereoscopic Rasteriser

Certain polygons may be visible to both views and others may be exclusive to a particular view. Derivation can only be performed for polygons that are common to both views — simply deriving one view from the other will result in erroneously inserted or omitted polygons. Also, image derivation should not consistently produce polygons in one view from the other, e.g. always derive right polygons from left polygons. The source polygon should always be the larger projected polygon, i.e. the primary polygon. View exclusive polygons are always handled in monocular mode by the primary pipeline only. The pixel stream redirection unit accommodates such derivation characteristics.

Primary and secondary pixel streams are allocated to one of two memory controllers that access display memory. The memory controllers may arbitrate for display memory access with the host, for example via shared frame buffer support, or may have exclusive display memory access.

5. The Secondary Pipeline

The primary pipeline is essentially a classic monocular graphics pipeline. This is augmented by a smaller secondary pipeline that derives secondary polygonal views. Where view *independent* functionality is present in the primary pipeline the secondary pipeline will use primary pixel data, possibly in some manipulated form, to construct a secondary pixel — no functional replication is necessary. Where view *dependent* functionality, e.g. antialiasing, alpha blending, etc., is incorporated into the primary pipeline it must be replicated in the secondary pipeline. The primary pipeline functionality therefore dictates to some extent the nature of the secondary pipeline.

Also, the generation of secondary polygons from primary polygons using pure derivation techniques based on spatial mapping only may produce a variety of artifacts in a derived polygonal image. In order to eliminate these artifacts we extend our derivation technique beyond a simple spatial mapping operation. The degree to which we extend the secondary pipeline functionality is dependent on the cost-performance compromise we wish to achieve — a variety of secondary pipeline configurations are thus possible. Figures 3 and 4 show two possible secondary derivation based rasteriser architectures.

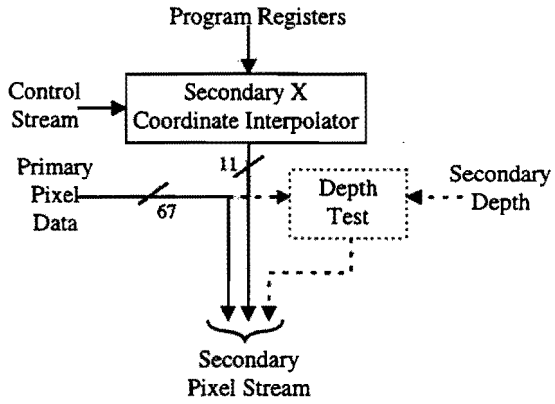


Figure 3 - Minimal System

The first, figure 3, is a minimal system implementation that performs pure derivation using spatial mapping only. The secondary pipeline samples the primary pixel stream to obtain the data used to construct a secondary pixel. This system is very simple and will produce stereoscopic images but its functionality is obviously restricted and image artifacts are probable.

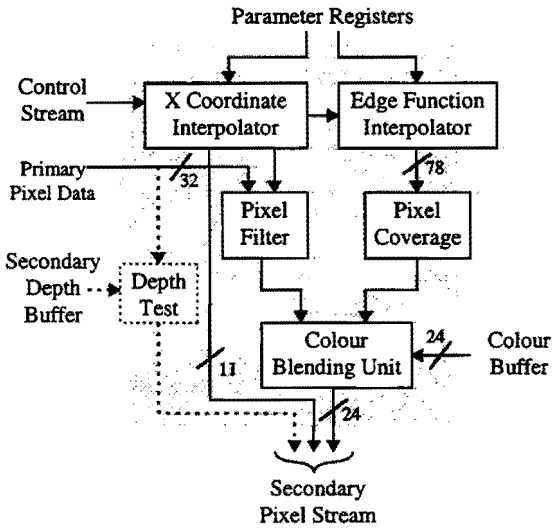


Figure 4 - Hybrid System

The second, figure 4, is a hybrid derivation approach that incorporates additional functionality to prevent artifacts specific to polygons from arising. It also supports enhanced rendering capability, i.e. antialiasing and blending functions. Depth testing may be performed internally or externally.

5.1 X coordinate Interpolation

Primary x coordinate increments are generally unitary integer values, whereas secondary x coordinate increments are fixed point, with values of one or less. — the secondary x coordinate uses an extended data format of 11.14. Unfortunately, such a secondary pixel coordinate format complicates interpolation of other secondary parameters, notably secondary linear edge functions, as we must round secondary parameters to appropriate pixel centred values before inside/outside boundary tests can be performed. To avoid this complication extra functionality is incorporated into the secondary x coordinate interpolator that allows us to use secondary parameter increments based on purely integer secondary coordinate increments[1]. This modification eliminates the need for several costly multiplication and addition operations per primary pixel while requiring little additional interpolator logic.

5.2 Y coordinate Interpolation

An implementation suitable for immersive applications does not require a y coordinate interpolator for the secondary pipeline as y coordinates of all projected left and right-eye points are identical [9]. If, however, we wish to support alternative display modes, for example that of the Cyberscope [15], a secondary y coordinate interpolator is necessary. Also, in non-immersive applications the viewer may move or orientate independently of the display device. Therefore the axis along which parallax occurs is not generally horizontal and projected left- and right-eye points may have differing y coordinates — a secondary y coordinate interpolator is required. Unfortunately simple spatial mapping algorithms suitable for such non-immersive applications may produce artifacts. If considered too distracting[2] these can be eliminated, but the solution becomes sub-optimal. Alternatively, the immersive SAM formulation can be used in a non-immersive environment to generate artifact free images based on viewer position alone — viewer orientation might be used as an additional navigation control.

5.3 Linear Edge Function Interpolation

It would seem correct to assume that the derivation process implicitly defines the boundary of our derived polygon. Therefore if a source pixel lies inside the primary polygonal bounds it will result in a derived pixel that is inside the secondary polygonal bounds. Similarly, all pixels lying outside the source polygonal bounds will result in derived pixels that are outside the secondary polygonal bounds. Such an assumption is fundamental to pure derivation techniques and it is generally wrong.

Pure derivation techniques have no representation of the geometry of the derived polygon. It is therefore not possible to determine if a secondary *point* is rounded to a *pixel centre* that lies inside or outside the secondary polygon geometry when displayed—consider figure 4. Artifacts are especially visible on lower resolution displays such as those found in many head-mounted display designs. We maintain secondary polygon edge quality comparable to the primary using linear edge functions [5] to distinguish valid and invalid secondary pixels.

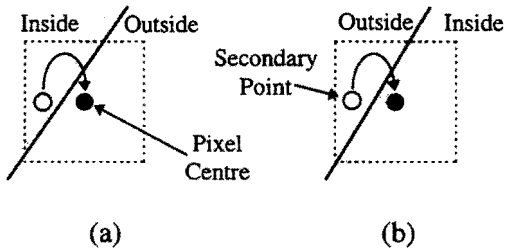


Figure 5 - Pixel Rounding at Edges

In addition, when rendering in a purely monocular mode we may use the redundant secondary linear edge function interpolators to support rendering of more complex and higher order planar primitives.

5.4 Anti-aliasing

Antialiasing is performed based on the extent to which a polygon geometry covers a pixel. Linear edge functions can be used to calculate pixel coverage values for polygon edge pixels [14]. However, the geometries and pixel coverage of the primary and secondary polygons are generally different and so primary and secondary pixel coverage values must be calculated independently. The polygon traversal algorithm will generally be a function of primary linear edge function values so aliasing of secondary polygon edges may be compromised, although less optimal traversal solutions incorporating secondary linear edge function dependence are possible. Our initial simulations indicate that antialiasing of polygonal edges is desirable to avoid retinal rivalry between stereo-image pairs— such rivalry produces an ambiguous and disturbing binocular perception of 'folds' where polygonal edge depth becomes indeterminate.

5.5 Specular Illumination

There are several approaches that can be used to support view exclusive specular highlights. Here we propose three alternatives :

- *Approximation* — we simply ignore the view dependency of specularities. This produces errors in perceived light position and viewed surface characteristics [2].
- *Exception Handling* — do not use derivation when scan converting any polygon that has a significant specular illumination component. This reduces the system performance but produces correct results.
- *Integrated Support* — use a second set of R, G, and B interpolators for the secondary pipeline. This does not reduce system performance and produces visually correct results but requires a significant increase in system hardware.

Given that the number of specularities in a scene is usually small, exception handling is attractive. If high performance is of primary concern the more expensive integrated support approach might be used. A similar set of alternatives may be

[1] Although not noted by the authors, line drawing theory described in [3] is applicable to stereoscopic image derivation. Our approach is different, however, as we do not explicitly use a residual function.

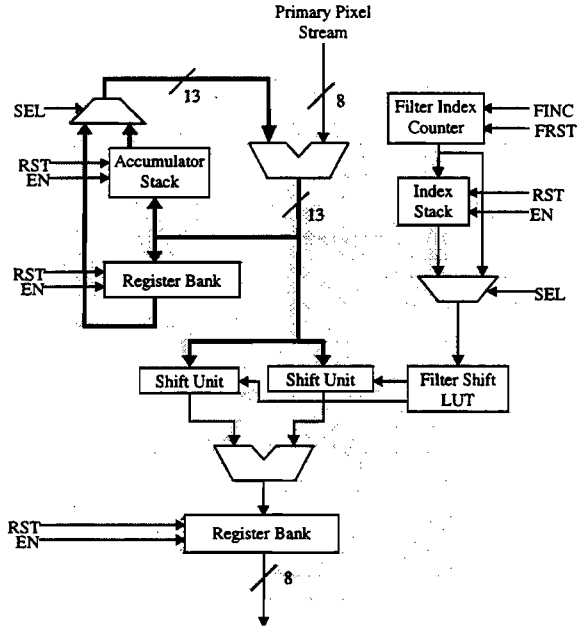
[2] Certain stereoscopic viewing techniques actually rely on the ability of an observer to ignore gaps visible in an image, e.g. parallax barriers [7].

considered for accommodating stereoscopic environment mapping.

5.6 *Pixel Filtering*

It is possible for more than one primary pixel to be mapped within a single secondary pixel extent. We can obtain a suitable secondary pixel parameter value from multiple primary pixel parameter values in a variety of ways, although a point sampling strategy is easy to implement in hardware and is valid for parameters that vary gradually over the extent of a polygon.

Such an approach produces aliasing artifacts when sampling primary textured pixel values obtained from a texture map that contains high spatial frequencies. Experimental evidence has shown that high spatial frequency data should only be used with small disparities [17]. In this case few primary pixels will be mapped to a single secondary pixel and aliasing artifacts will occur less frequently. However, avoidance of such cases can not be guaranteed To reduce aliasing artifacts we filter all primary textured pixels mapped to a single secondary pixel.



Although strictly a two dimensional problem, we treat secondary image filtering as a one dimensional operation along scan lines — this reduces distortion due to secondary image scaling but does not eliminate shearing distortion. We use a simple box filtering process accumulating n primary pixel colour values and averaging them with a division by n.

5.6.1 Filter Division

True division is expensive to implement so we make the approximation :

$$\frac{1}{n} \approx \frac{1}{2^p} + \frac{1}{2^q} \qquad \{1\}$$

where n is the number of accumulated primary pixel values. Appropriate values for p and q are indexed from an LUT dependent on the number of accumulated pixels we are averaging. We can now perform an approximate division using two shift and one accumulate operations, as shown in figure 6.

Using equation 1 to approximate 1/n, where $1 \le n \le 29$, gives a maximum error of 12.5% but an average error of only 4.26% — colour artifacts are generally negligible. We choose a 29 entry filter function as this provides sufficient support for high primary to secondary pixel mapping ratios while not requiring overly complex hardware. Rasteriser operation defaults to monocular operation for primary to secondary mapping ratios of greater than 29:1.

Greater filtering accuracy can be achieved using a true divison operation. Alternatively, we can extend the number of terms in the divisor of our approximation in equation 1 beyond two.

5.7 *Colour Blending*

The colour blending unit combines the various secondary pixel colour sources into a single derived pixel colour value. The complexity of the secondary blending unit is dictated by the degree to which we transform the pure derivation algorithm to a hybrid form and the rendering functionality supported by the primary pipeline.

The blending unit produces secondary pixel values defined by the blending factor derived from the secondary pixel coverage and alpha blending values — equation 2 shows the blending function used :

$$C = St + (1 - t)B \qquad \{2\}$$

where S represents the current secondary pixel colour, B is the background or current secondary colour buffer value, t is the blending factor, and C is the new pixel colour. For some rendering modes it may be possible to assume that C is the same for both views. The primary and secondary blending factors may be modified independently during antialiasing. We eliminate the complexities of blending support for specular highlights and environment maps by using the exception handling scheme noted in section 5.5.

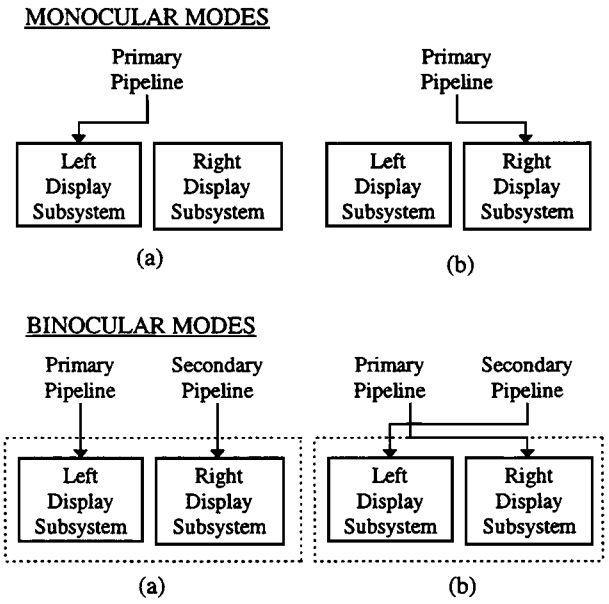


Figure 7 - Pixel Stream Redirection Modes

5.8 *Pixel Stream Redirection*

Primary and secondary polygons can occur arbitrarily in either the left- or right-eye views. Also, view exclusive polygons may occur in both left and right-eye views — this requires rendering in a monocular mode using a directed primary pipeline only. Pixel allocation in both cases is provided by the pixel stream redirection unit. This unit is implemented using simple multiplexer functionality.

Dependent on display subsystem design, primary and secondary pixel streams might be directed to a single display device without compromising pipeline operating speed — as indicated by the dashed boxes in figure 7.

6. Implementation

Secondary pipeline components have been modeled using synthesisable VHDL code. The post-optimisation gate counts for synthesised functional units of the secondary pipeline are shown in table 1 below. For completeness we include a gate count for a y coordinate interpolator.

Secondary Functional Unit	Gate Count
X coordinate Interpolator	2423
Y coordinate Interpolator	392
Edge-function Interpolator	2768
x3 =	8304
Filter Unit	1873
Pixel Coverage Unit	4000[3]
Pixel Blending Unit	4520
Pixel Redirection Unit	1242

Table 1 - Secondary Pipeline Gate Counts

The figures in table 1 indicate that a stereoscopic image derivation architecture can be produced by augmenting a standard monocular pipeline with a secondary pipeline at relatively little extra cost. We can construct secondary rasterisation architectures requiring between approximately 5,000 gates, for the pure derivation architecture of figure 3, and approximately 30,000 gates for the hybrid derivation architecture of figure 4. These values are significantly smaller than current gate count estimates in excess of approximately 100,000 gates for compatible primary pipeline architectures. (System gate estimates are generous as the amount of registers required to maintain synchronisation between primary and secondary pipelines and extended state machine control have yet to be fully quantified).

Synthesis of VHDL models is initially targeted at the Matra-Harris ASIC gate library to obtain a realistic generic gate count estimate. Future design iterations will also be targeted at Altera field programmable gate array (FPGA) technology. This reconfigurable technology is currently being used to develop a rapid prototyping test bench that will allow us to perform the complete design cycle 'in-house'.

[3] Preliminary estimate.

7. Conclusion

We have presented a rasteriser architecture capable of producing stereoscopic image pairs from polygonal scene descriptors using polygon derivation. We have shown how this approach can be extended to overcome limitations common to pure derivation techniques, and incorporate support for enhanced rendering functionality.

The purely incremental nature of our technique makes it well suited to hardware implementation. The approach allows for a scaleable system trading complexity against cost— starting from the basic requirements of a single coordinate interpolator, additional hardware can provide support for more complex photorealistic effects and increased image quality.

Initial synthesis results are promising and indicate that using image derivation techniques a significant saving in stereoscopic rasteriser gate count is possible compared to dual monocular pipeline implementations.

8. Acknowledgements

Many thanks are given for numerous helpful discussions with and suggestions from Graham Dunnett, Simon Pearce, Marcus Waller, and Martin White during the preparation of this work.

9. References

[1]. Baker, J., "Generating images for a time-multiplexed stereoscopic computer graphics system", *Proceedings of S.P.I.E. - True 3D Imaging Techniques and Display Technologies*, Vol. 761, 1987, pp. 44-52.

[2]. Blake, A., Bülthoff, H., "Does the brain know the physics of specular reflection?", *Nature*, Vol. 343, 1990, pp. 165-168.

[3]. Braccini, C., & Marino, G., "Fast Geometrical Manipulations of Digital Images", *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 127-141.

[4]. Deering, M., "High Resolution Virtual Reality", Computer Graphics, Vol. 26, No. 2, July 1992, pp. 195-202.

[5]. Dunnett, G. J., White, M., Lister, P.F., Grimsdale, R., Glemot, F., "The Image Chip for High Performance 3D Rendering", *IEEE Computer Graphics and Applications*, Vol. 12, No. 6, 1992, pp. 41-52.

[6]. Grotch, S.L., "Three-Dimensional and stereoscopic graphics for scientific data display and analysis", *IEEE Computer Graphics and Applications*, Vol. 3, No. 8, 1983, pp. 31-43.

[7]. Harrison, L., & McAllister, D.F., *Stereo Computer Graphics and other True 3D Technologies*, edited by D.F. McAllister, Princeton University Press, Princeton, New Jersey, 1993, pp. 119-151.

[8]. Hodges, L. F., & McAllister, D.F., "Rotation algorithm artifacts in stereoscopic images", *Optical Engineering*, Vol. 29, No. 8, 1990, pp. 973-976.

[9]. Hodges, L. F., "Tutorial : Time-Multiplexed Stereoscopic Computer Graphics", *IEEE Computer Graphics & Applications*, Vol. 12, No. 2, March 1992, pp. 20-30.

[10]. Lipscomb, J. S., "Experience with Stereoscopic Display Devices and Output Algorithms", *Proceedings of S.P.I.E. - Three-Dimensional Visualization and Display Technlogies*, Vol. 1083, 1989, pp. 28-34.

[11]. M^{c}Cann, S., White, M., Dunnett, G., Lister, P., "Efficient Stereoscopic Rasterisation", Submitted for Publication to *Computer Graphics Forum.*

[12]. Penna, M. A., & Patterson, R., *Projective geometry and its applications to computer graphics,* Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[13]. Roese, J. A., and McCleary, L. E., "Stereoscopic computer graphics for simulation and modelling", *Computer Graphics,* Vol. 13, No. 2, 1979, pp. 41-47.

[14]. Schilling, A., "A New, Simple and Efficient Antialiasing with Subpixel Masks", *Computer Graphics*, Vol.25, No. 4, 1991, pp. 133-141.

[15]. Simsalabim Systems, Inc., *The Cyberscope^{TM} Reference Manual*, 1993.

[16]. Southard, D. A., "Transformations For Stereoscopic Visual Simulation", *Computers and Graphics*, Vol. 16, No. 4, 1992, pp. 401-410.

[17]. Yeh, Y., *Stereo Computer Graphics and other True 3D Technologies*, edited by D.F. McAllister, Princeton University Press, Princeton, New Jersey, 1993, pp. 50-70.

Appendix

The construction of a parallax relation between homologous points allows us to determine a horizontal coordinate relation between primary and secondary points. This parallax relation is absolute so must be re-evaluated at every primary pixel. Also, the parallax relation involves perspective division — to prevent perspective distortion of derived pixel positions we must avoid the commonly accepted practice of linear interpolation of depth. These problems are common to the parallax relation noted for the z-shear and TCP [7] approaches to stereoscopic projection — *we note here that z-shear and TCP projection techniques are mathematically equivalent, although the authors are unaware of any previous acknowledgement of this point.* We subsequently assume that when referring to projection we imply use of a z-shear or TCP projection, without distinction.

In order to circumvent the problems noted above we approach the problem purely in image space. We wish to map pixels in the primary polygon to points in the secondary polygon — this is achieved using an affine mapping. This affine mapping is converted to an incremental form. Equations 3 and 4 show the standard primary x and y coordinate increment equations :

$$x_{p_{n+1}} = x_{p_n} + 1 \qquad \{3\}$$

$$y_{p_{n+1}} = y_{p_n} + 1 \qquad \{4\}$$

The associated secondary x and y coordinate increment equations are defined by equations 5 and 6 respectively :

$$x_{s_{n+1}} = x_{s_n} + G_1 \qquad \{5\}$$

$$\left.\begin{aligned} x_{s_{n+1}} &= x_{s_n} + G_2 \\ y_{s_{n+1}} &= y_{s_n} + 1 \end{aligned}\right\} \qquad \{6\}$$

where subscripts p and s refer to primary and secondary coordinates respectively. G_1 and G_2 are secondary increments based on the relative geometry of primary and secondary polygons, and effectively represent scale and shear operations respectively. These secondary parameters can be efficiently computed using byproducts of standard primary scan conversion setup computations.

The implementation of the SAM derivation technique described in this paper is based around the preceding theory.