Volume Rendering Based on Oblique Projections

Hubert W.J. Borst Pauwels, Óscar Mealha, Beatriz Sousa Santos, José M.R. Nunes

Resumo - **Este artigo introduz um método novo para visualizar dados representados por voxels utilizando projecções oblíquas. Em vez da abordagem tradicional em que os objectos são rodados e depois ortogonalmente projectados, investigamos a possibilidade de de usar projecções oblíquas para visualizar um objecto de vários pontos do espaço com ganhos significativos em velocidade de cálculo. Provamos que projecções oblíquas conduzem a uma superfície igual à que é gerada quando o objecto é rodado no espaço 3-D em torno dos eixos Z- e X- seguido de projecção ortogonal. Quando desejado a distorção da superfície pode ser retirada com uma transformação geométrica sobre a imagem final. Indicamos como determinamos e exploramos projecções oblíquas específicas com a propriedade de projectar os vertices dos voxels exclusivamente em coordenadas discretas de forma a obter uma representação precisa da superfície.**

Abstract - **This paper introduces a new method for visualization of voxel represented data based on oblique projections. Instead of the traditional approach in which objects are rotated and then orthogonally projected, we investigated the possibility of using oblique projections to view an object under multiple different viewing positions with a speed advantage. We prove that oblique projections will lead to the same surface that is rendered when an object is rotated in 3-D space about Z- and X- axes followed by orthogonal projection. Moroever, when desired, the oblique distortion of the surface can be removed by a geometrical transformation of the final image. We will indicate how we can determine and exploit specific oblique projections with the property of mapping vertices of voxels exclusively on discrete coordinates in order to obtain very accurate rendering of surfaces.**

I. INTRODUCUCTION

Nowadays scientists can choose out of a range of different methods for visualizing 3-D data. According to the taxonomy of Elvins, [1] two main streams of algorithms can be distinguished, surface fitting algorithms and direct volume rendering algorithms.

Surface fitting algorithms detect surfaces in 3-D data and subsequently transform them in polygonal descriptions. The geometric primitives involved are usually triangles which can be used either to connect previously detected surface contours [2] or more directly, to describe surfaces within logical cubes [3]. In [4] an alternative approach is described as the cuberille model where surfaces are modeled with faces of cubes instead of triangles.

Direct volume rendering methods project surface elements directly into screen space without using geometric primitives as an intermediate representation. Surface elements can be projected either directly from the volume array into the image plane or in the reverse way by casting rays from an image plane into the volume array. An early direct volume rendering method that projects surface elements is described by Frieder et al. [5]. Their method traverses slices in back-to-front order whereby the image coordinates of voxels are calculated according to a 3-D rotation matrix and scaled to avoid artifact holes. More recently developed direct volume rendering methods that are based on orthogonal projections use so called splats as projection primitives [6], [7].

In this paper we describe a new direct volume rendering method based on oblique projections that can be used for back-to-front display of voxel based data. We prove that a surface obtained by rotating an object about Z- and X- axes in 3-D space, and then orthogonaly projected can also be obtained by an oblique projection followed by a 2-D transformation, essentially a 2-D scaling, which results in a speed advantage. Furthermore, we will indicate how we can determine and exploit specific oblique projections with the property of mapping vertices of voxels exclusively on discrete coordinates in order to obtain very accurate rendering of textures on objects.

II. OBLIQUE PROJECTIONS

An oblique projection of an object is obtained using projectors that are not perpendicular to the projecting plane [8]. We can describe an oblique projection as shown in fig-

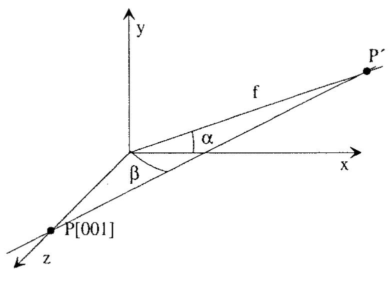

Fig. 1 - Oblique projection on projection plane z=0.

ure 1, by the values α and f where f (the foreshortening factor) is the projection of the unit z-axis vector [001] on the projection plane (z=0) and α is the angle between this projection and the x-axis. Figure 1 also shows β the angle between the oblique projector (direction of projection) and the plane of projection, $\beta = \cot^{-1}(f)$. Representing points by row matrices and using homogeneous coordinates, the transformation matrix for producing any parallel projection of 3D object on the projecting plane z=0 is defined as:

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -f\cos\alpha & -f\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

THEOREM 1
Let
1-X be a 3-D object
2-$[3DRPz]$ be the transformation matrix that rotates X an angle δ about the local z'-axis, followed by a rotation ϕ about the local x'-axis and subsequently followed by an orthogonal projection on the $z = 0$ plane Then, there exists an oblique projection $[POz]$ and a 2-D transformation $[2D]$ such that

$$[X]\,[3DRPz] = [X]\,[POz]\,[2D]$$

end of theorem 1
For a proof of theorem 1 see appendix A.
The significance of theorem 1, for visualizing voxel-represented objects, is twofold. Provided the voxels are projected in the same order as if they were projected with the rotation matrix:

1. The same set of visible surface faces obtained after rotating an object about Z- and X- axes followed by orthogonal projection can be obtained by a specific oblique projection.
2. The same set of visible surface faces with the same geometrical proportions obtained after rotating an object about Z- and X- axes followed by orthogonal projection can be obtained by a specific oblique projection followed by a specific 2-D transformation.

III. THE DISCRETE COORDINATES PARADIGM

Important for the quality of a 3-D visualization method is the accuracy with which coordinates in 3-D space (object space) can be transformed in a discrete 2-D space (image space). In methods that use a 3-D rotation matrix we always encounter the problem that 3-D coordinates may be transformed into non-discrete coordinates, which will have to be rounded or truncated. In this section we will indicate how we can avoid the occurrence of non-discrete coordinates during a 3-D to 2-D transformation by using a set of specific oblique projections with the property of projecting vertices of voxels exclusively onto discrete coordinates. We will indicate how to choose particular values α and f to obtain this property by means of the following theorem.
THEOREM 2
Let S_1 be an axis aligned square of size R with its left bottom vertex positioned on point (0,0) (figure 2).
Let S_2 be a square obtained from S_1 using a translation vector (T_x, T_y).
Let C be an axis aligned cube of size R, with one of its vertices on the origin and the others with positive coordinates (figure 2).
Then an oblique projection defined by:

$$\alpha = \arctan\frac{T_y}{T_x} \quad (2)$$

and

$$f = \frac{T_x}{-R\cos\alpha} \quad (3)$$

projects the vertices of C onto the vertices of S_1 and S_2

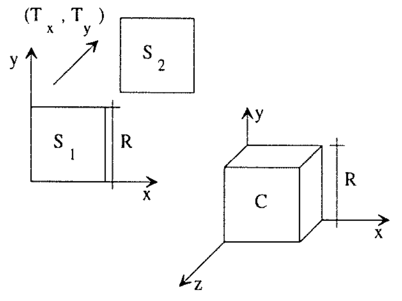

Fig. 2 - An oblique projection of a cube can be described as a translation of a square.

end of theorem 2.
For a proof of theorem 2 see appendix B.
With the help of theorem 2, we can now establish a set of oblique projections with the property of projecting voxel vertices exclusively on discrete coordinates. The procedure is simple. Draw an oblique projected voxel of size R on a grid using a translation vector (T_x, T_y), using integer values for R, T_x and T_y, calculate the oblique parameters α and f with equations 2 and 3. According to theorem 1, we can subsequently calculate the associated 3-D rotation angles with equations 17 and 18, as described in appendix A.
For example, from an oblique voxel defined by $T_x = T_y = -1$ and $R = 1$ we can calculate the oblique parameters α and f and therefore the associated 3-D rotation angles, which in this case will lead to an isometric projection. In contrast with this oblique projection an isometric projection does not exclusively map voxel vertices onto discrete coordinates.

IV. THE OBLIQUE VOXEL METHOD

The oblique voxel method taverses a 3-D volume slice by slice starting with the slice furthest away from the observer. Voxels in each slice are traversed in back-to-front order guided as in case the object was rotated by the associated 3-D rotation matrix. We can use different discrete projecting voxels as projection primitives as can be identified with theorem 2. In order to avoid artifact holes each voxel is scan converted on its bounding rectangular area as depicted in the example of figure 3.

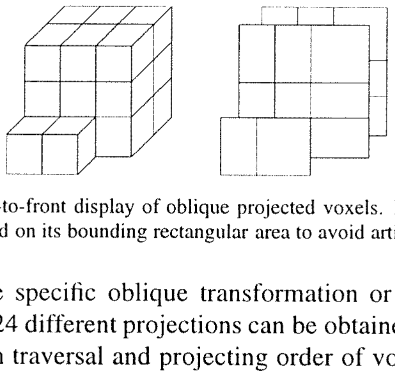
Fig. 3 - Back-to-front display of oblique projected voxels. Each voxel is scan-converted on its bounding rectangular area to avoid artifact holes.

Using one specific oblique transformation or projection primitive, 24 different projections can be obtained by combining both traversal and projecting order of voxels. Specifically, projection of slices can take place in 3 different ways as shown in figure 4.

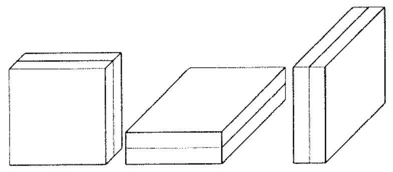

Fig. 4 - Three different ways of projecting oblique slices in back-to-front order.

Traversal of slices can take place either in first-to-last or last-to-first order and traversal of voxels within each slice can take place in 4 ways by alternating the start point of reading at one of the four corners of a slice.

The final step in the method consists of a 2-D transformation, a 2D warping post-process, of the image in order to remove the oblique distortion. We use the transformation matrix $[2D]$ of equation 4 with matrix parameters as can be calculated with equations 11 and 12 as described in appendix A.

V. Computational complexity of oblique voxel method

The computational complexity of any direct volume rendering method that projects voxels in back to front order without pre-processing of the volume data can be expressed as a time function $O(f(n^3))$ where n stands for the maximal dimension of the volume data and f is the time function expressing the computational effort to calculate the image coordinates of a voxel and to scan-convert its associated projection primitive on a display.

In the oblique voxel method, calculation of image coordinates is a relatively inexpensive procedure since no 3-D matrix is required. Image coordinates of voxels are determined while traversing the volume data by tracking two pointers yielding the projected coordinates of the currently accessed voxel. The bottle neck in the computational complexity of the method can be found when it is applied with oblique projections leading to large dimensions for the rectangular projection primitives (see theorem 2). However, significant optimisation for these critical oblique projections could be obtained when the display system supports either software or hardware for fast scan-conversion of rectangles.

VI. Results

We tested the method's expected abilities of rendering surfaces with voxel precision by applying it on several objects. For example, we created a 64x64x64 cube covered with a texture of numbers (figure 5), on each of its faces.

A distance image was produced with the oblique voxel method using the following parameters $T_x = T_y = -1$ and $R = 1$, from which we can calculate the oblique parameters α and according to theorem 2. By means of equations (17) and (18) the associated 3-D rotation angles can be calculated, which in this case will lead to an isometric projection. The oblique distortion can be removed by 2-D transformation with a scaling factor and rotation angle as can be calculated with equations (12) and (17). Figure 6 shows the

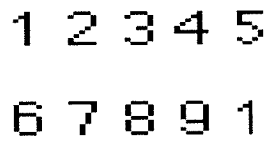

Fig. 5 - Image displaying numbers to be used as a texture.

oblique distance image processed with an image space shading operator [9].

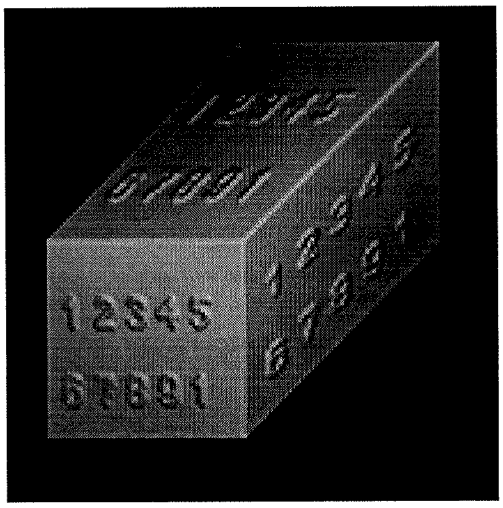

Fig. 6 - An oblique projection of a 64x64x64 cube with a texture of numbers.

The oblique distortion was removed by a transformation using a bi-cubic interpolation of the oblique image leading to an isometric projection of the cube as displayed in figure 7.

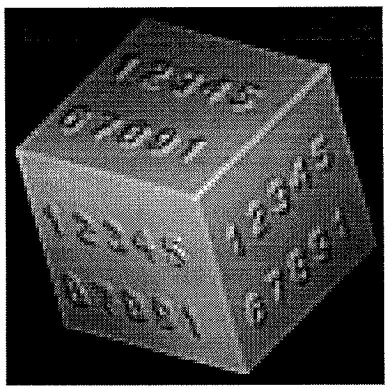

Fig. 7 - Resulting image after transforming the oblique projection into an isometric one by means of a 2-D transformation based on a bi-cubic interpolation.

Figure 8 shows the same isometric projection of the cube obtained after applying a direct volume rendering method that uses a 3-D rotation matrix and projects voxels as pixels. The resulting z-buffer image was processed with the same image space shading operator as in the case of the oblique image displayed in figure 6. The significant loss in texture information, which is clearly visible at all three faces of the cube, can be explained by the fact that pixels are not very suitable projection primitives for the hexagonal shape of rotated and orthogonally projected voxels and by the fact that the image coordinates of the rotated voxels may have to be rounded.

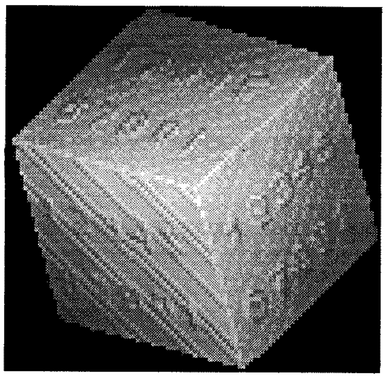

Fig. 8 - Isometric projection obtained with a method that uses a 3-D rotation matrix and projects voxels as pixels.

Figure 9 shows the same isometric projection of the cube obtained after applying a ray casting method that uses nearest neighbour interpolation. The ray caster is capable of preserving the texture information but some irregularities are visible, for example in the boundary of the numbers and the "false" edgės on the originally smooth surface parts of the cube.

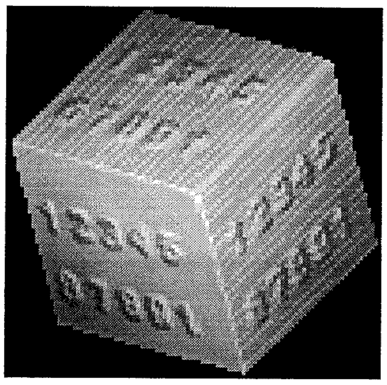

Fig. 9 - Isometric projection obtained with a ray caster that uses nearest neighbour interpolation.

A second test object consisted of a 256x256x129 volume containing data of a human thorax (obtained by Computarised Axial Tomography). A distance image was produced with the oblique voxel method using the following parameters, $T_x = T_y = -1$ and $R = 1$ which, as described above, is equivalent to an isometric projection. Figure 10 shows the oblique distance image after being processed with the image space shading operator.

The oblique distortion was removed by transformations based on a bi-cubic interpolation and a bi-linear interpolation of the original oblique image. The resulting image (figure 11) is far more sharper than the slightly blurred image resulting from transformation based on bi-linear interpolation (figure 12).

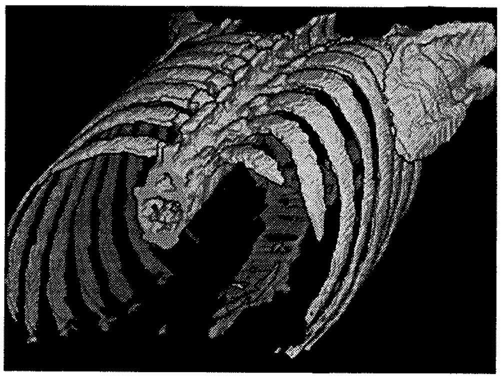

Fig. 10 - An oblique projection of a 256x256x129 volume containing CAT data of a human thorax.

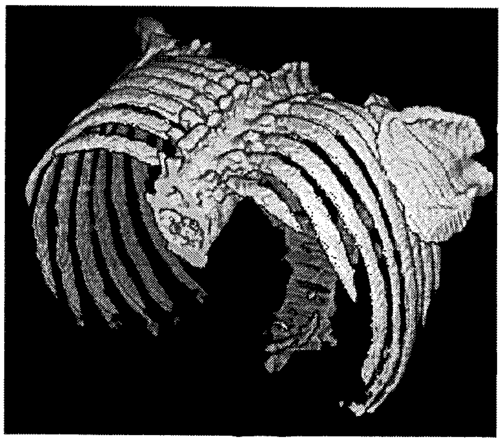

Fig. 11 - Image resulting from transforming the oblique projection into an isometric one using a 2-D transformation based on a bi-cubic interpolation.

However, in contrast with the transformation based on bilinear interpolation, the transformation based on bi-cubic interpolation leads to some "false" dark spots or pixels at the edges of the ribs due to interpolation with the background colour of the image. In comparison with the image produced by the direct volume rendering method that uses a 3-D rotation matrix and projects voxels as pixels (figure 13) some differences with the oblique voxel method are visible especially in the texture of the shoulder blade and spine.

The image produced by the ray casting method that uses nearest neighbour interpolation (figure 14) is comparable in quality to the images produced by the oblique voxel method.

All the methods described in this section were implemented in the C language on a Silicon Graphics, Iris Indigo Elan 4000 with 80MByte of memory. We have measured the average time of 10 trials for each method in order to produce a distance image of the thorax after the entire volume was read into memory. The oblique voxel method used 1.2 seconds, the direct volume rendering method which projects voxels as pixels used 3.3 seconds and finally the ray caster that uses nearest neighbour interpolation used 166.5 seconds. Additionally, we measured the time needed by the oblique voxel method in order to generate a distance image of the same volume but now with oblique parameters $T_x = T_y = -1$ and $R = 2$ which can be associated with a rotation of 45 degrees about both z and x axis. As expected, due to the larger dimension for the rectangular projection primitives, we

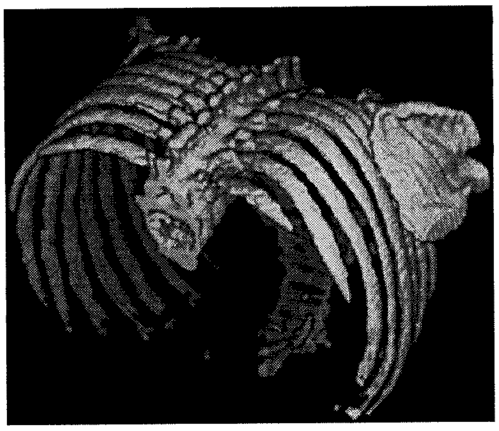
Fig. 12 - Resulting image after transforming the oblique projection into an isometric one using a 2-D transformation based on a bi-linear interpolation.

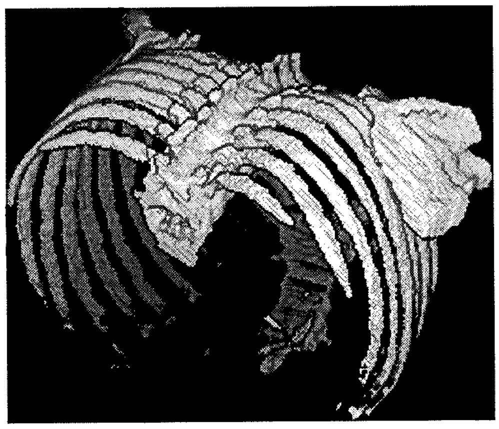
Fig. 13 - Isometric projection obtained with a method that uses a 3-D rotation matrix and projects voxels as pixels.

measured a slightly larger time of 1.3 seconds.

In relation to the results we refer briefly to the discussion in [10] describing the specific advantages and disadvantages of oblique projections in comparison with orthogonal projections. Oblique projections may be particularly suitable for objects with much detail or irregular shapes on one principal face since they have the property of displaying the exact shape of one face of an object. For example, see the front face of the oblique projected cube in figure 6 which displays the numbers with preservation of angles and lengths. A disadvantage of oblique projections is the typical oblique distortion which may lead to an unrealistic perception of the object, however our 2D warping post-process eliminates this problem. For example the oblique projected cube in figure 6 appears to be a parallelepiped object instead of a cube as is perceived correctly in the orthogonal projection in figure 7.

VII. DISCUSSION

In addition to the current spectrum of volume rendering methods, oblique projections can be helpful to render voxel data with voxel precision. We gave the mathematical foundations for a direct volume rendering method allows a 3-D object to be viewed under different viewing positions without using a 3-D rotation matrix. We have indicated how

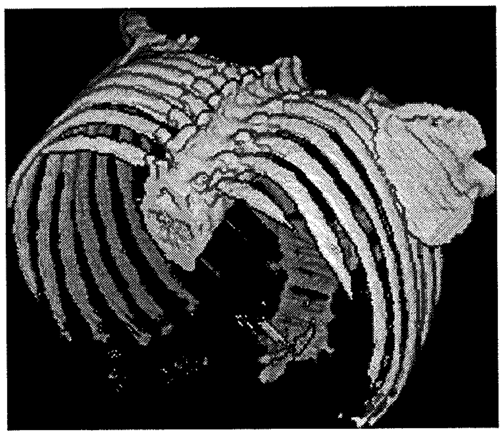
Fig. 14 - Isometric projection obtained with a ray caster that uses nearest-neighbour interpolatioray caster that uses nearest-neighbour interpolation.

we can use specific oblique projections, that project vertices of voxels exclusively onto discrete coordinates, in order to avoid the introduction of rounding errors of coordinates while the surface is generated. As a consequence, surfaces are rendered with hardly any loss in texture information. A unique property of the method is that it produces two images displaying the same set of points of an object, with both an oblique and an orthogonal projection with the advantage that each projection has its specific characteristics.

We conclude that the oblique voxel method is capable of very fast and accurate volume rendering with the constraint that it allows an object only to be viewed under a limited set of different rotation angles corresponding to specific oblique projections. As future work we intent to investigate the possibility of a more general projection method that allows the user to view the volume from any angle, allowing for example, quick rendering of images or real time animation sequences.

ACKNOWLEDGEMENTS

This work was partially funded by the Portuguese Agency for Cientific Research and Technology (JNICT) with a scholarship BD/1733/91-IA.

REFERENCES

[1] T. Todd Elvins, "A survey of algorithms for volume visualization", *Computer Graphics*, vol. 26, no. 3, pp. 194–201, Aug. 1992.

[2] H. Fuchs, Z. M. Kedem, and S. P. Uselton, "Optimal surface reconstruction from planar contours", *Comunications of the ACM*, vol. 20, no. 10, pp. 693–702, Oct. 1977.

[3] William E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm", *ACM Computer Graphics*, vol. 21, no. 4, pp. 163–169, July 1987.

[4] Gabor T. Herman and Hsun Kao Liu, "Three-dimensional display of human organs from computed tomograms", *Computer Graphics and Image Processing*, vol. 9, pp. 1–21, 1979.

[5] Gideon Frieder, Dan Gordon, and R. Anthony Reynolds, "Back-to-front display of voxel-based objects", *IEEE Computer Graphics & Applications*, pp. 52–60, Feb. 1985.

[6] J. Wilhelms and A.V. Gelder, "A coherent projection approach for direct volume rendering", *Computer Graphics*, vol. 25, no. 4, pp. 275–284, 1991.

[7] Lee Westover, "Footprint evaluation for volume rendering", *ACM Computer Graphics*, vol. 24, no. 4, pp. 367–376, Aug. 1990.

[8] Ludwig Adams, Werner Krybus, Dietrich Meyer-Ebrecht, Rainer Rueger, Joachim M. Gilsbach, Ralph Moesges, and George Schloendorff, "Computer-assisted surgery", *IEEE Computer Graphics & Applications*, pp. 43–51, May 1990.

[9] Dan Gordon and R. Anthony Reynolds, "Image space shading of 3-dimensional objects", *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 361–376, 1985.

[10] I. Carlbom and J. Paciorek, "Planar geometric projections and viewing transformations", *Computing Surveys*, vol. 10, no. 4, pp. 465–502, 1978.

APPENDIX A

Proof of theorem 1

Consider X to be oblique projected with $[POz]$ followed by a rotation about the z axis with an angle β, subsequently followed by a scaling along the y axis with a scaling factor S_y and finally followed by a translation along the x axis and y axis with factors T_x and T_y, respectively. The combined transformation is then described as:

$$[POz2D] = [POz]\,[2D] \tag{4}$$

where

$$[POz2D] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -f\cos\alpha & -f\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \circ$$

$$\begin{bmatrix} \cos\beta & S_y\sin\beta & 0 & 0 \\ -\sin\beta & S_y\cos\beta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ T_x & T_y & 0 & 1 \end{bmatrix} \tag{5}$$

(Note that although $[2D]$ describes a 2-D transformation it is written in the form of a 3-D transformation matrix) for which we calculate

$$[POz2D] = \begin{bmatrix} \cos\beta & S_y\sin\beta & 0 & 0 \\ -\sin\beta & S_y\cos\beta & 0 & 0 \\ A & S_y.B & 0 & 0 \\ T_x & T_y & 0 & 1 \end{bmatrix} \tag{6}$$

where

$$A = -f\cos\alpha\cos\beta + f\sin\alpha\sin\beta$$

and

$$B = -f\cos\alpha\sin\beta - f\sin\alpha\cos\beta$$

Now consider X to be rotated about a local axis system x'y'z' with its origin on C. Where $C = (C_x, C_y, C_z)$ is the centroid of X. A rotation of X about the local z' axis with angle δ followed by a rotation about the local x' axis with angle ϕ can be performed by translating X to make C coincident to that of the origin of the global axis system, susequently followed by the required rotations and finally by translating X back with C to its original position. After rotation, X will then be projected orthogonally onto the $z = 0$ plane. The combined transformation is then described as:

$$[3DRPz] = [3DR]\,[Pz] \tag{7}$$

where

$$[3DR] = [TR]\,[Rz]\,[Rx]\,[TR]^{-1}$$

specifically

$$[3DR] = \begin{bmatrix} \cos\delta & \cos\phi\sin\delta & \sin\phi\sin\delta & 0 \\ -\sin\delta & \cos\phi\cos\delta & \sin\phi\cos\delta & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ C & D & E & 1 \end{bmatrix} \tag{8}$$

where

$$C = -C_x\cos\delta + C_y\sin\delta + C_x$$

$$D = \cos\phi(-C_x\sin\delta - C_y\cos\delta) + C_z\sin\phi + C_y$$

and

$$E = \sin\phi(-C_y\cos\delta - C_x\sin\delta) - C_z\cos\phi + C_z$$

$$[3DRPz] = \begin{bmatrix} \cos\delta & \cos\phi\sin\delta & \sin\phi\sin\delta & 0 \\ -\sin\delta & \cos\phi\cos\delta & \sin\phi\cos\delta & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ C & D & E & 1 \end{bmatrix} \circ$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

from which we calculate

$$[3DRPz] = \begin{bmatrix} \cos\delta & \cos\phi\sin\delta & 0 & 0 \\ -\sin\delta & \cos\phi\cos\delta & 0 & 0 \\ 0 & -\sin\phi & 0 & 0 \\ C & D & 0 & 1 \end{bmatrix} \tag{10}$$

The equation $[POz2D] = [3DRPz]$ will hold a solution if:

$$\beta = \delta \tag{11}$$

$$S_y = \cos\phi \tag{12}$$

$$T_x = -C_x\cos\delta + C_y\sin\delta + C_x \tag{13}$$

$$T_y = \cos\phi(-C_x\sin\delta - C_y\cos\delta) + C_z\sin\phi + C_y \tag{14}$$

$$-f\cos\alpha\cos\beta + f\sin\alpha\sin\beta = 0 \tag{15}$$

$$S_y(-f\cos\alpha\sin\beta - f\sin\alpha\cos\beta) = -\sin\phi \qquad (16)$$

Applying a fixed α and f we can derive β, δ and ϕ. Since $f \neq 0$ equation 15 yields

$$\cos\alpha\cos\beta = \sin\alpha\sin\beta$$

Using the identities $\frac{\sin\alpha}{\cos\alpha} = \tan\alpha$ and $\frac{\sin\beta}{\cos\beta} = \tan\beta$ yields

$$\tan\alpha\tan\beta = 1$$

From which we derive

$$\delta = \beta = \frac{\pi}{2} - \alpha \qquad (17)$$

From equation 16 we can derive ϕ
Applying equation 12 in 16 yields

$$\cos\phi(-f\cos\alpha\sin\beta - f\sin\alpha\cos\beta) = -\sin\phi$$

Using the identity $\frac{\sin\phi}{\cos\phi} = \tan\phi$ yields

$$-f\cos\alpha\sin\beta - f\sin\alpha\cos\beta = -\tan\phi$$

From which we derive

$$\phi = \arctan(f\cos\alpha\sin\beta + f\sin\alpha\cos\beta)$$

Using the identities $\sin\beta = \cos\alpha$ and $\cos\beta = \sin\alpha$ conform equation 17 yields

$$\phi = \arctan(f(\cos^2\alpha + \sin^2\beta))$$

Using the identity $\cos^2\alpha + \sin^2\beta = 1$ yields

$$\phi = \arctan(f) \qquad (18)$$

Applying a fixed δ and ϕ we derive α and f

$$\alpha = \arctan(\frac{1}{\tan\delta}) \qquad (19)$$

$$f = \tan\phi \qquad (20)$$

APPENDIX B

Proof of theorem 2
See figure 2 for symbols.
Let S_1 be defined by the set $\{(0,0)(0,R)(R,R)(R,0)\}$ and S_2 defined by the set $\{(T_x,T_y)(T_x,R+T_y)(R+T_x,R+T_y)(R+T_x,T_y)\}$
Let C be defined by the coordinates formed by the union of C_1 and C_2. Where $C_1 = \{(0,0,0)(0,R,0)(R,R,0)(R,0,0)\}$ and $C_2 = \{(0,0,R)(0,R,R)(R,R,R)(R,0,R)\}$.
Given $\alpha = \arctan(\frac{T_y}{T_x})$ and $f = \frac{T_x}{-R\cos\alpha}$
Applying equation 1, the coordinates of any oblique projected point can be expressed as:

$$x_p = x - zf\cos\alpha \qquad (21)$$

$$y_p = y - zf\sin\alpha \qquad (22)$$

Applying equation (21 and equation (22) with $z = 0$ yields

$$x_p = x \qquad (23)$$

$$y_p = y \qquad (24)$$

Equations (21) and (22) will transform C_1 in S_1
Applying equation (21) with $z = R$, $\alpha = \arctan(\frac{T_y}{T_x})$ and $f = \frac{T_x}{-R\cos\alpha}$ will give

$$x_p = x - R(\frac{T_x}{-R\cos\alpha})\cos\alpha$$

which yields

$$x_p = x + T_x \qquad (25)$$

Applying equation (22) with $z = R$ and $f = \frac{T_x}{-R\cos\alpha}$ will give

$$y_p = y - R(\frac{T_x}{-R\cos\alpha})\sin\alpha$$

which yields

$$y_p = y + T_x\frac{\sin\alpha}{\cos\alpha}$$

Using the identity $\frac{\sin\alpha}{\cos\alpha} = \tan\alpha$ yields $y_p = y + T_x\tan\alpha$
Applied with $\alpha = \arctan(\frac{T_y}{T_x})$ will give $y_p = y + T_x\frac{T_y}{T_X}$ which yields

$$y_p = y + Ty \qquad (26)$$

Equation (25) and (26) will transform C_2 into S_2