Graphical Models **62,** 308–322 (2000)
doi:10.1006/gmod.2000.0525, available online at http://www.idealibrary.com on IDEAL®

3D Volume Rotation Using Shear Transformations

Baoquan Chen

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, Minnesota 55455
E-mail: baoquan@computer.org

and

Arie Kaufman

Center for Visual Computing and Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11794-4400
E-mail: ari@cs.sunysb.edu

Received June 15, 1999; accepted February 8, 2000; published online May 24, 2000

We present a group of methods for decomposing an arbitrary 3D volume rotation into a sequence of simple shear (i.e., regular shift) operations. We explore different types of shear operations: *2D beam shear*, a shear in one coordinate based on the other two coordinates; *2D slice shear*, a shear of a volume slice (in two coordinates) according to the third coordinate; and *2D slice–beam shear*, the combination of a beam shear and a slice shear. We show that an arbitrary 3D rotation can be decomposed into four 2D beam shears. We use this decomposition as a basis to obtain the sequence of 3D rotation decomposition into four 2D slice shears or three 2D slice–beam shears. Moreover, we observe that two consecutive slice shears can be achieved by shifting beams in 3D space, a transformation we call a *3D beam shear*. Therefore, an arbitrary 3D rotation can be decomposed into only two 3D beam shears. Because of the regularity and simplicity of the shear operation, these decompositions are suitable for implementations on a multipipelined hardware or a massively parallel machine. In addition, we present a resampling scheme in which only a single-pass resampling is required for performing multiple-pass shears to achieve the 3D volume rotation.

1. INTRODUCTION

Three-dimensional volume transformation plays a key role in volume modeling and manipulation, registration of multiple volumes, and volume rendering [4]. Straightforward hardware implementation of volume rotation is very expensive [3, 5]. Naive implementation on parallel machines is also inefficient because rotation requires global communication and

1524-0703/00 $35.00

could cause contention while data is written back to the distributed memory modules. Shear transformations, however, capitalize on nearest neighbor connections and lend themselves to a feasible hardware or parallel implementation. Any hardware with a barrel shifter can efficiently perform shear transformation. Utilizing the neighboring connection, a barrel shifter is able to shift an entire beam of voxels by m places in one shift cycle [2]. Shear or pseudo shear has also been implemented on parallel machines [11, 15].

Existing algorithms decomposing 3D volume rotation into shears are usually direct extensions of the multiple-pass algorithms for 2D image rotation. Drebin *et al.* [4] decomposed an arbitrary 3D rotation into sequences of major axis rotations, each of which could be performed as a 2D rotation of every slice perpendicular to the axis. They further applied Catmull and Smith's [1, 12] two-pass algorithm for each 2D rotation. Later, Hanrahan [6] generalized the two-pass image transformation method to a three-pass algorithm for volume affine transformation. This generalization is significant; however, the caveat is that each pass is a more complicated pseudo shear transformation—a shift coupled with scaling. This scaling not only complicates the resampling, but also may cause a situation (called *bottleneck*) where a beam (a volume row) is first shrunk and then magnified so that the original beam cannot be recovered.

A three-shear decomposition of a 2D image rotation was introduced independently by Paeth [8] and Tanaka *et al.* [13], expressed as

$$R_{2D}(\alpha)=\begin{bmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{bmatrix}=\begin{bmatrix}1 & 0\\ -\tan\frac{\alpha}{2} & 1\end{bmatrix}\begin{bmatrix}1 & \sin\alpha\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ -\tan\frac{\alpha}{2} & 1\end{bmatrix}. \tag{1}$$

A 1D shear operation does not suffer from any bottleneck problems and its resampling is also much simpler. A straightforward extension of this method to 3D was proposed by Schröder and Salem [11]. From the initially obtained nine-shear decomposition sequence, they managed to merge two neighboring shears into a single shear, resulting in an eight-shear decomposition. The first attempt at directly performing decomposition on 3D rotation was taken by Wittenbrink [17] and recently by Toffoli and Quick [14]:

$$R = R_x(\phi)R_y(\theta)R_z(\alpha)=\begin{bmatrix}1 & a_{12} & a_{13}\\ 0 & 1 & a_{23}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ b_{21} & 1 & 0\\ b_{31} & b_{32} & 1\end{bmatrix}\begin{bmatrix}1 & c_{12} & c_{13}\\ 0 & 1 & c_{23}\\ 0 & 0 & 1\end{bmatrix}. \tag{2}$$

Each matrix in Eq. (2) is an upper/lower triangular matrix, which can be interpreted as a *general shear* operation—first sliding (*shearing*) volume slices (a volume plane perpendicular to a major axis) along one another and then sliding (*shearing*) beams within each slice along one another.

In this paper, we present a group of methods for decomposing an arbitrary 3D rotation into sequences of different types of shear. We start in Section 2 with a *2D beam shear* decomposition of four passes, in which each shear pass only involves sliding beams along each other. A 2D beam shear is the simplest of the shear operations, as it involves only 1D shifts. We then use this decomposition as a basis for obtaining all the other types of shear decompositions. First, we observe that transposing a 2D beam shear matrix gives us a matrix defining a *2D slice shear*, in which slices slide along each other, but each slice itself remains rigid. Consequently, we transpose a 2D beam shear sequence and obtain a 2D slice shear sequence (Section 3). More attractively, we observe that two consecutive slice shears

can be achieved by shifting (translating) a beam in 3D space, an operation we define as a *3D beam shear*. Therefore, by merging two neighboring shears in the 2D slice shear sequence, we obtain a two-pass 3D beam shear (Section 4). Second, since a 2D beam shear can be replaced with two 1D shears, in a 2D beam shear sequence we can split the second shear in this manner and merge the two 1D shears with the first and third 2D shears, resulting in a *2D slice–beam* shear for each (Section 5). This results in a three-pass 2D slice–beam shear decomposition. The first two passes of this decomposition are of the same complexity as those Toffoli and Quick's three-pass decomposition, but our last pass is simpler, a mere 2D beam shear. In addition, our 2D slice–beam shear matrix is not restricted to being either a lower or an upper triangular matrix as in Toffoli and Quick's decomposition. This yields more combinations of decomposition sequences. Multiple decomposition gives us a fine implementation trade-off. The type of shear ranges from the simplest 2D beam shear to a 3D beam shear with increasing complexity, while the number of decomposition passes ranges from 4 to 2. Each type of shear operation features a different implementation. Nevertheless, all shears guarantee rigid translation of beams along at least one axis.

2. 3D ROTATION BY 2D BEAM SHEARS

In a 2D beam shear, a beam is merely shifted in its major direction without any change in the other two coordinates. As an example, a 2D x-beam shear is expressed as

$$x = x + a \cdot y + b \cdot z. \tag{3}$$

Figure 1 shows a 2D x-beam shear. The dotted line box is sheared to the thick solid line box position. A 2D beam shear is the simplest shear since a beam is shifted without scaling and without changing the other two coordinates. Therefore, only linear interpolation is required for resampling, as opposed to bilinear interpolation for Toffoli and Quick's general shear.

When a 2D beam shear is constructed directly from a 2D rotation decomposition, it needs at least eight passes [11]. Since Toffoli and Quick have presented a three-pass "general" shear decomposition, intuitively splitting each pass into two 2D beam shears provides a six-pass 2D beam shear sequence. We can further merge two neighboring shears, which

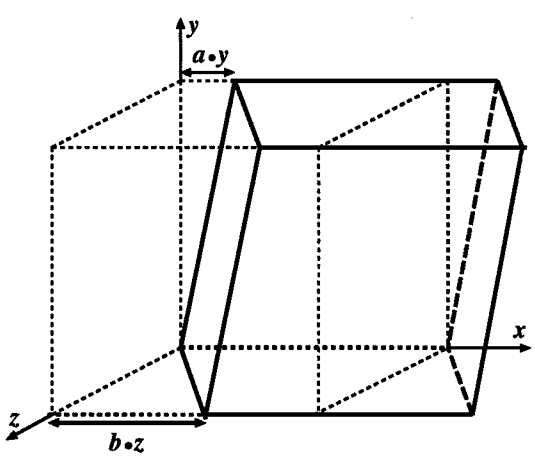


FIG. 1. x-beam shear along y and z by a and b, respectively.

operate on the same directional beam, and obtain a five-pass 2D beam shear decomposition. However, we can do better than that, as described next.

We desire to apply a decomposition directly to the 3D rotation matrix. To elucidate the following description, we write a 2D x-beam shear as $S(x_b, a, b)$, interpreted as a 2D x-beam shear along y and z by a and b, respectively. A 2D y-beam shear, $S(y_b, c, d)$, and a 2D z-beam shear, $S(z_b, e, f)$, are similarly defined.

In 3D, there are three directional beam shears. Consecutive shears along the same axis produce a conforming shear. For example,

$$S(x_b, a, b) \cdot S(x_b, a', b') = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ a' & 1 & 0 \\ b' & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ a + a' & 1 & 0 \\ b + b' & 0 & 1 \end{bmatrix}. \tag{4}$$

Therefore, when we design our shear sequence, neighboring shears have to be in different directions. However, as shown in the following, three distinct beam shears M cannot achieve an arbitrary 3D rotation:

$$\begin{aligned} M &= S(x_b, a, b) \cdot S(y_b, c, d) \cdot S(z_b, e, f) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & c & 0 \\ 0 & 1 & 0 \\ 0 & d & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & e \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & c & e + cf \\ a & ac + 1 & ae + (ac + 1) f \\ b & bc + d & be + (bc + d) f + 1 \end{bmatrix}. \end{aligned} \tag{5}$$

If M is to be a rotation, normality of the first column and row requires that $a = b = c = e \equiv 0$. M is then simplified to

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & f \\ 0 & d & df + 1 \end{bmatrix}. \tag{6}$$

Further, normality of the second column and row demands that $d = f \equiv 0$. Thus, $M \equiv I$. This proves that three distinct beam shears can only achieve the *identity* rotation.

To build the general 3D matrix from 2D shear matrices, we need at least four 2D beam shears. Using an analysis similar to that above, the fourth 2D beam shear cannot be in the same direction as the second shear. It also cannot be in the same direction as the third shear according to Eq. (4). Therefore, the fourth 2D beam shear has to be the same directional shear as the first 2D beam shear. From these constraints, we can have six permutations of shear sequence, (X_b, Y_b, Z_b, X_b), (X_b, Z_b, Y_b, X_b), (Y_b, X_b, Z_b, Y_b), (Y_b, Z_b, X_b, Y_b), (Z_b, X_b, Y_b, Z_b), (Z_b, Y_b, X_b, Z_b), where capital characters indicate the beam directions.

A 3D rotation matrix can be expressed as the concatenation of three major axis rotations, $R_x(\phi)$,$R_y(\theta)$, and $R_z(\alpha)$. A different order of this concatenation results in a different 3D rotation. Without losing generality, we choose $R = R_x(\phi)R_y(\theta)R_z(\alpha)$ as our underlying 3D rotation matrix. For each of the shear sequences, we compute the product of the consecutive shear matrices and make it equal to the target 3D rotation matrix. As an example, we take the sequence (Y_b, Z_b, X_b, Y_b)

$$R = R_x(\phi)R_y(\theta)R_z(\alpha) = S(y_b, c, d) \cdot S(z_b, e, f) \cdot S(x_b, a, b) \cdot S(y_b, g, h). \tag{7}$$

This matrix equation implies nine trigonometric equations with eight variables, a, b, c, d, e, f, g, and h. In solving these equations we obtain

$$
\begin{aligned}
a &= -\cos\theta\sin\alpha \\
b &= \frac{\cos\theta\sin\alpha + \sin\phi\sin\theta\cos\alpha - \cos\phi\sin\alpha}{\sin\phi\cos\theta} \\
c &= \frac{\sin\theta\sin\alpha(\cos\phi - \cos\theta) + \sin\phi(\cos\theta - \cos\alpha)}{\sin\phi\sin\alpha\cos^2\theta} \\
d &= \frac{\cos\phi\cos\theta - 1}{\sin\phi\cos\theta} \\
e &= -\frac{\cos\phi\sin\theta\sin\alpha - \sin\phi\cos\alpha + \sin\phi\cos\theta}{\cos\theta\sin\alpha} \\
f &= \sin\phi\cos\theta \\
g &= -\frac{\cos\theta\cos\alpha - 1}{\cos\theta\sin\alpha} \\
h &= \frac{\sin\phi\sin\theta(\cos\theta - \cos\alpha) + \sin\alpha(\cos\phi - \cos\theta)}{\sin\phi\sin\alpha\cos^2\theta}.
\end{aligned}
\tag{8}
$$

In the same way, we can obtain the remaining five shear sequences for a given 3D rotation matrix. We use volume data similar to that used by Toffoli and Quick [14] to demonstrate our decomposition. The original volume has a resolution of 64^3 with each voxel of 1 byte. Volumes are rendered using ray-casting. The original volume is shown in Fig. 2a. Figures 2b–2e show one such sequence of the four consecutive 2D beam shear transformations achieving the 3D rotation of $\phi = 45^\circ$, $\theta = 30^\circ$, and $\alpha = 30^\circ$.

From Eq. (8) we see that when angle θ is 90° or when angle ϕ or α is 0°, the denominator becomes zero. This is a singularity problem, which also exists in all previous decompositions. However, since for each 3D rotation we provide six decomposition sequences, the angles causing singularity for each sequences are different. We can choose different sequences according to the rotation angles to avoid the singularity problem. The following is the solution for all the special cases of Eq. (8):

1. When both ϕ and α are zero, the 3D rotation degenerates to a 2D rotation of $R_y(\theta)$.
2. When $\theta = 90^\circ$ and $\phi = \alpha$, it degenerates to a 2D rotation of $R_y(90^\circ)$.
3. When $\theta = 90^\circ$ and $\phi \neq \alpha$, we choose the sequence (Z_b, Y_b, X_b, Z_b).
4. When $\phi = 0^\circ$, we choose the sequence (X_b, Z_b, Y_b, X_b).
5. When $\alpha = 0^\circ$, we choose the sequence (Z_b, Y_b, X_b, Z_b).

3. 3D ROTATION BY 2D SLICE SHEARS

In a 2D slice shear, a slice is merely shifted within its plane. For example, a 2D y-slice shear is expressed as

$$
\begin{aligned}
x &= x + c \cdot y \\
z &= z + d \cdot y.
\end{aligned}
\tag{9}
$$

Figure 3 illustrates the 2D y-slice shear. The thick solid line box is the shear result of the dotted line box. We write a 2D y-slice shear as $S(y_s, c, d)$, interpreted as a y-slice shear along x and z by c and d, respectively. A 2D x-slice shear, $S(x_s, a, b)$, and a 2D z-slice

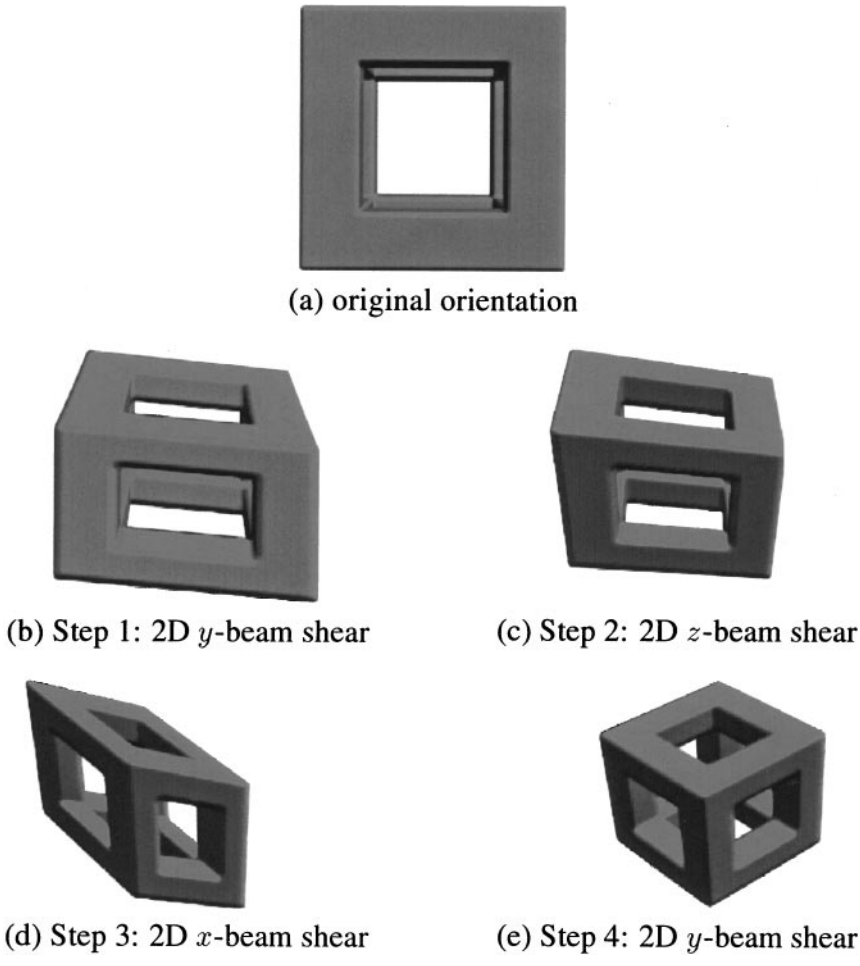


FIG. 2. 3D rotation ($\phi = 45^\circ$, $\theta = 30^\circ$, and $\alpha = 30^\circ$) achieved by four consecutive 2D beam shears.

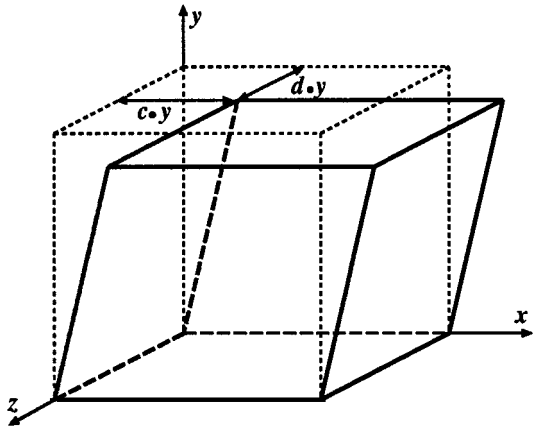


FIG. 3. y-slice shear along x and z by a and b.

shear, $S(z_s, e, f)$, are similarly defined. We can derive the 2D slice shear decompositions in the same way as in Section. 2. However, more straightforwardly, we can derive them directly from the 2D beam shear decompositions. A slice shear matrix can be obtained by transposing a beam shear matrix. Therefore, by transposing a 2D beam shear sequence, we can obtain a 2D slice shear sequence. For example, take Eq. (7) and transpose each side:

$$
\begin{aligned}
R^{\mathrm{T}} &= \mathbf{R_z}(-\alpha)\mathbf{R_y}(-\theta)\mathbf{R_x}(-\phi) \\
&= [S(y_b, c, d) \cdot S(z_b, e, f) \cdot S(x_b, a, b) \cdot S(y_b, g, h)]^{\mathrm{T}} \\
&= \left[\begin{bmatrix} 1 & c & 0 \\ 0 & 1 & 0 \\ 0 & d & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & e \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & g & 0 \\ 0 & 1 & 0 \\ 0 & h & 1 \end{bmatrix} \right]^{\mathrm{T}} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ g & 1 & h \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & a & b \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ e & f & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ c & 1 & d \\ 0 & 0 & 1 \end{bmatrix} \\
&= S(y_s, g, h) \cdot S(x_s, a, b) \cdot S(z_s, e, f) \cdot S(y_s, c, d). \qquad (10)
\end{aligned}
$$

Since the transpose of a rotation is equal to the inverse of the rotation, this resulting slice shear sequence achieves an inverse of the original rotation. To obtain 2D slice shear decomposition for a given rotation R, we need to first calculate the 2D beam shear decompositions for the inverse rotation R^{T} and then apply the transpose transformation to the 2D beam shear sequence. Since each 2D beam shear decomposition corresponds to a 2D slice shear decomposition, we can also obtain six sequences of 2D slice shear decompositions. Figures 4a–4d show one such sequence of four consecutive 2D slice shear transformations achieving the 3D rotation of $\phi = 45°$, $\theta = 30°$, and $\alpha = 30°$.

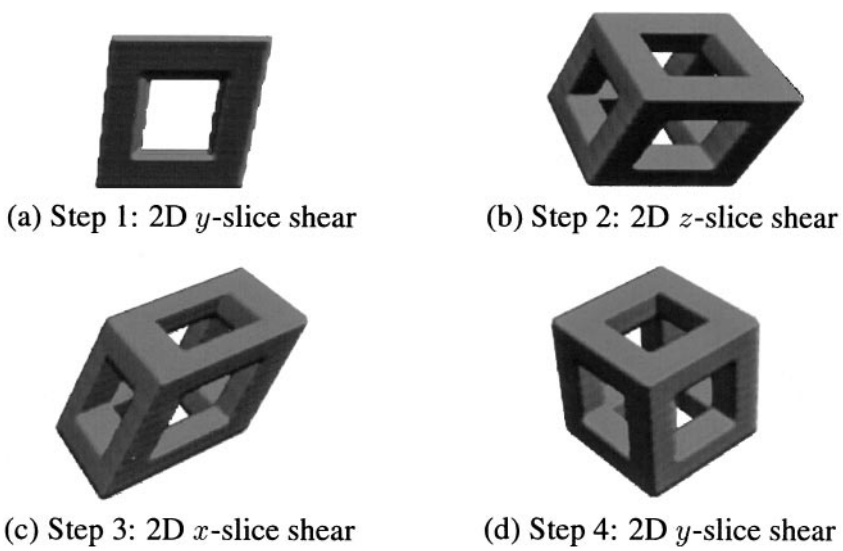


FIG. 4. 3D rotation ($\phi = 45°$, $\theta = 30°$, $\alpha = 30°$) achieved by four 2D slice shears.

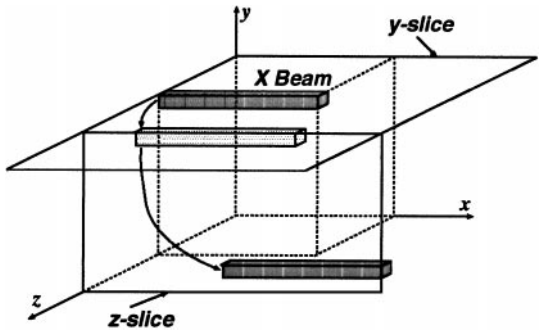


FIG. 5. A 3D x-beam shear.

4. 3D ROTATION BY 3D BEAM SHEARS

We further observe that two neighboring slice shears have one directional beam in common, and the combination of two can be achieved by shifting (or translating) these common beams in 3D space. For example, an x-beam is a common beam of y-slice and z-slice shears. We call this kind of beam translation in 3D space a 3D beam shear. Figure 5 shows a 3D x-beam shear, which is equivalent to the concatenation of two consecutive 2D slice shears $S(y_s, c, d)S(z_s, e, f)$. Similarly, we define the other two directional 3D beam shears: 3D z-beam shear, equivalent to $S(x_s, a, b)S(y_s, c, d)$ or $S(y_s, c, d)S(x_s, a, b)$, and 3D y-beam shear, equivalent to $S(x_s, a, b)S(z_s, e, f)$ or $S(z_s, e, f)S(x_s, a, b)$. Every 3D beam shear involves only one major beam. In Fig. 5, the darker x-beam is translated to a new 3D location following the arrows. The lighter beam indicates the intermediate position if we interpret this as two consecutive 2D slice shears.

Therefore, given a slice shear decomposition, the number of shears can be reduced to merely 2 by introducing 3D beam shear. We denote the three directional 3D beam shears as $S_{x_{3D}}$, $S_{y_{3D}}$, and $S_{z_{3D}}$. Now, an arbitrary 3D rotation can be decomposed into only two consecutive 3D beam shears. For example, directly from the 2D slice shear sequence $S(y_s, c, d) \cdot S(z_s, e, f) \cdot S(x_s, a, b) \cdot S(y_s, g, h)$, a 3D beam shear sequence is obtained as

$$R = S_{x_{3D}} \cdot S_{z_{3D}}, \tag{11}$$

where

$$\begin{aligned} S_{x_{3D}} &= S(y_s, c, d) \cdot S(z_s, e, f) = \begin{bmatrix} 1 & 0 & 0 \\ c + de & 1 + df & d \\ e & f & 1 \end{bmatrix} \\ S_{z_{3D}} &= S(x_s, a, b) \cdot S(y_s, g, h) = \begin{bmatrix} 1 + ag & a & b + ah \\ g & 1 & h \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{12}$$

Figures 4b and 4d show the two consecutive 3D shear transformations achieving the 3D rotation of $\phi = 45^\circ$, $\theta = 30^\circ$, and $\alpha = 30^\circ$. A unique property of this decomposition is that a 3D rotation involves a minimum of two major beam transformations. However, an intuitive implementation of this decomposition leads to complex interpolation. We provide more discussion on this in Sections 6 and 7.

5. 3D ROTATION BY 2D SLICE–BEAM SHEARS

In a 2D slice–beam shear, slices shift within their planes, while beams within each slice shift along one another. For example, a 2D y-slice–x-beam shear is expressed as

$$\begin{aligned} x &= x + c \cdot y + g \cdot z \\ z &= z + d \cdot y. \end{aligned} \tag{13}$$

We write a 2D y-slice–x-beam shear as $S(y_s, x_b, c, d, g)$, interpreted as a y-slice shear along x and z by c and d, respectively, and an x-beam shear along z by g. Figure 6 illustrates this shear operation. The dotted line box is sheared to the thick solid line box position.

In Toffoli and Quick's decomposition, each pass is a lower or upper triangular matrix, which is a 2D slice–beam shear. However, a 2D slice–beam shear matrix is not necessarily a lower or upper triangular matrix. Here we make this kind of shear more general. From a 2D beam shear decomposition, we can obtain a 2D slice–beam shear by simply rearranging the matrices. For example, if we take Eq. (7), split the second pass into two 1D shears, and merge them with the first and third passes, we form the following three-pass decomposition:

$$\begin{aligned} R &= S(y_b, a, b) \cdot S(z_b, e, f) \cdot S(x_b, c, d) \cdot S(y_b, g, h) \\ &= \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & b & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & e \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ d & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & g & 0 \\ 0 & 1 & 0 \\ 0 & h & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & a & e \\ 0 & 1 & 0 \\ 0 & b & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ c + df & 1 & f \\ d & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & g & 0 \\ 0 & 1 & 0 \\ 0 & h & 1 \end{bmatrix} \\ &= S(x_s y_b, a, e, b) \cdot S(y_s x_b, c + df, f, d) \cdot S(y_b, g, h). \end{aligned} \tag{14}$$

In this three-pass decomposition, the first two shears are 2D slice–beam shears, while the third one remains a 2D beam shear. We can obtain the other 2D slice–beam shear decomposition sequences by applying the same operation to the other 2D beam shear

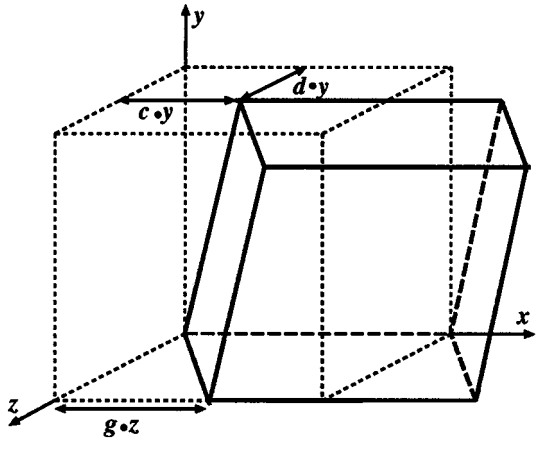
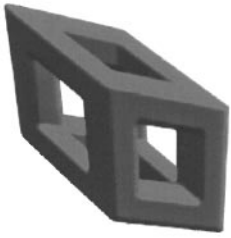


FIG. 6. 2D y-slice-x-beam shear: 2D y-slice shear along x and z by c and d, respectively, combined with 2D x-beam shear along z by g.

(a) Step 1: 2D x-slice-y-beam shear



(b) Step 2: 2D y-slice-x-beam shear

(c) Step 3: 2D y-beam shear

FIG. 7. 3D rotation ($\phi = 45^\circ$, $\theta = 30^\circ$, $\alpha = 30^\circ$) achieved by two 2D slice-beam shears and one 2D beam shear.

decompositions. Therefore, not only are our three-pass decompositions slightly simpler than Toffoli and Quick's decomposition, but we can also provide multiple decomposition sequences. Figures 7(a)–7(c) show one such sequence of two consecutive 2D slice–beam shear transformations and one 2D beam shear transformation achieving the 3D rotation of $\phi = 45^\circ$, $\theta = 30^\circ$, and $\alpha = 30^\circ$.

6. RESAMPLING

When multiple-pass algorithms are used, the resampling techniques chosen are the key to achieving high quality. We present two resampling approaches: *multiple resampling* and *single resampling*. Intuitively, resampling is necessary for each pass because a continuous shear transformation may move voxels off the grid points. The shear results shown in Figs. 2, 4, and 7 are results with resampling in each pass. The problem with multiple resampling is that the resulting volume quality is lower than the straightforward one-pass rotation. Therefore, we devise a technique to perform only single-pass resampling while taking advantage of the regular data flow of multiple shears.

When multiple resampling is performed, each resampling pass is usually a simple 2D or even 1D interpolation (except 3D shear decomposition) because of the simplicity of each shear pass. A 2D beam shear requires a simple linear interpolation, while a 2D slice shear requires a bilinear interpolation. In a 2D slice shear, the whole slice shifts as a rigid plane; all voxels within the same slice have the same displacement such that they share the same bilinear interpolation weights. A 2D slice–beam shear also requires bilinear interpolation, but due to an additional beam shear within each slice, each beam has a different bilinear interpolation weight. However, voxels within a beam still share the same weights. Compared with the other shears, 3D beam shear involves the most complicated interpolation—trilinear interpolation.

One problem with multiple resampling is the quick degradation of the volume quality when consecutive rotations are applied to a volume. It is thus desirable to sample the volume only once. Here we propose such a method. The idea is to precompute a sampled volume and then use a multipass shear sequence to shuffle sampled voxels to their destinations. In this procedure only zero-order (nearest neighbor) interpolation is performed in each shear pass. Wittenbrink and Somani had taken a similar approach in their permutation warping [16, 17]. However, there the sampled voxels are sent to their destinations using global communication, while in our case each shear features only local communication.

Given an original volume (source volume) and the desired rotated volume (target volume), we first set up one-to-one correspondence between a source voxel and a target voxel. This one-to-one mapping is guaranteed by our multipass shear decomposition, because each shear is a one-to-one transformation when zero-order interpolation is used [16]. The concatenation of a sequence of one-to-one mappings remains one-to-one. Once this one-to-one correspondence is set up, we calculate for each source voxel (v_s) its corresponding target voxel (v_t) and store it in the source voxel position. The sampling position of each target voxel (v_t') is computed using the direct backward transformation of the rotation. During this procedure, the resampling is performed by interpolation on the local voxels; therefore, no global communication is required.

After we obtain the values for all target voxels and store them in the source voxels, we must shuffle them to their destinations in the target volume. Intuitively, this involves global communication. However, global communication is expensive to perform for parallel implementation. Therefore, we use multiple shears with a nearest neighbor placement scheme to achieve this voxel shuffling. As shear is a regular, conflict-free transformation, each pass can thus be very efficiently performed. Using our 3D beam shear decomposition, we only need a minimum of two passes of regular local communication to achieve the effect of global communication.

Note that care must be taken to avoid the overlap of beams in 3D beam shear. Take the 3D x-beam shear in Eq. (12) as an example. While each x-beam is preserved, that is, while an x-beam remains rigid after a 3D x-beam shear, several x-beams may overlap with each other. To maintain the required one-to-one mapping, we have to utilize the fact that a 3D beam shear is the concatenation of two 2D slice shears (Section 4). A 2D slice shear maintains a one-to-one mapping when zero-order interpolation is used. Therefore, our solution is to calculate the destination coordinates using the same order as that of two consecutive 2D slice shears (but we perform communication only once). For a 3D x-beam shear, while the x-coordinate is calculated directly using the 3D shear matrix, the y and z coordinates of each beam are calculated as

$$\begin{aligned} z' &= \mathit{round}(z + b \cdot y) \\ y' &= \mathit{round}(y + f \cdot z'), \end{aligned} \tag{15}$$

where $\mathit{round}(w)$ is a function of rounding w to the nearest integer. Coordinates (y', z') determine the integral coordinates of the whole beam for the nearest neighbor storage. This way, no overlaps occur.

Figure 8 shows the two 3D beam shear steps achieving 3D volume rotation when only single resampling is used. We use a voxelized gazebo (Fig. 8a) to demonstrate the results. Figure 8b shows the intermediate sampled volume by calculating and storing in each source voxel its assigned destination voxel. Figures 8c and 8d show that after two 3D beam shears,

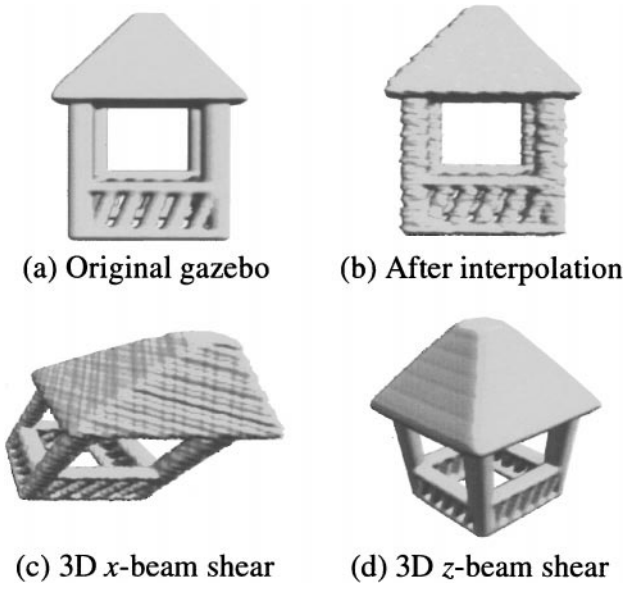


FIG. 8. 3D rotation of the gazebo using two 3D beam shears and single-pass resampling.

using only a nearest neighbor placement scheme, sampled voxels move exactly to their destination locations. Note that in an implementation, the interpolation operation can be merged with the first 3D beam shear pass. There is no need to create an intermediate sampled volume; instead, once the samples of an x-beam are computed, they are shifted using the 3D x-beam shear. The second beam shear performs no interpolation, but merely shifting.

To evaluate the quality of different approaches, we rotate an original volume with $R = R_x(\phi)R_y(\theta)R_z(\alpha)$ using slice–shear decomposition (Y_s, Z_s, X_s, Y_s) and then rotate the result back to its original orientation with $R = R_z(-\alpha)R_y(-\theta)R_x(-\phi)$. The difference volume between the twice-rotated volume and the original is then calculated and volume rendered. We use a forth and back rotation of $\phi = 45$, $\theta = 30$, and $\alpha = 30$. for demonstration. The transfer function used is a linear ramp between 0 and 50 changing from zero to full opacity. Figure 9(a) shows that using no resampling (nearest neighbor placement)

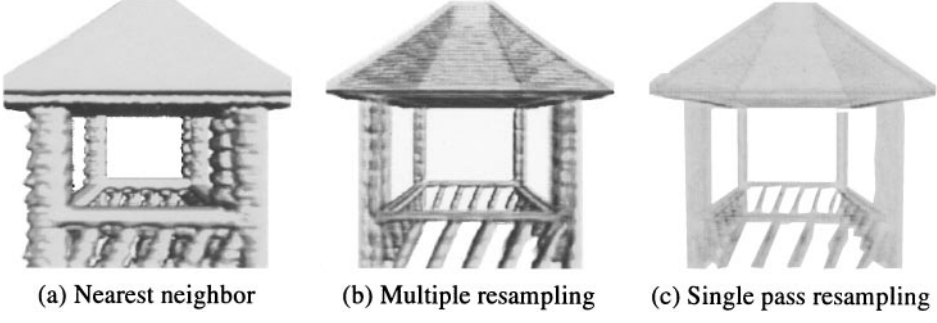


FIG. 9. Volume rendering of the difference volume for the various resampling methods. The original volume is rotated forth and back with the same angles using the various methods. Then the difference volume between the twice rotated volume and the original volume is calculated and volume rendered. The transfer function used is a linear ramp between 0 and 50 changing from 0 to full opacity.

produces the largest error. Multiple resampling (non-zero-order interpolation) produces much less error than the nearest neighbor approach, as shown in Fig. 9b. Single-pass resampling creates the least error, as can be seen from Fig. 9c, which is identical to direct 3D volume rotation.

7. SHEARING ON THE CUBE ARCHITECTURE

Shear transformation has been implemented on several massively parallel distributed memory machines, such as the Connection Machine (CM-200) [11], Maspar MP-1 [15–17], and CAM-8 [14]. Shearing can also be very efficiently implemented on any hardware with a barrel shifter [2].

We describe a hardware design to perform efficient shearing. It is the Cube architecture [10], a special-purpose hardware design for real-time volume rendering, developed at the State University of New York at Stony Brook. Mitsubishi Electric VolumePro board [9], based on the Cube-4 design, has been available on the market since the second quarter of 1999. In the Cube architecture, volume rotation is especially important for rendering overlapping volumes. Consider the scenery where smoke rises up through a cloud, or a radiation beam penetrates through a human organ. Because the Cube design features a slice-by-slice processing order, slices from different overlapping volumes should be interlaced for a correct rendering. The slice is determined by the storage order in memory; therefore, it is critical to align the overlapping volumes so that their memory storages reflect their physical positions. This involves a rotation transformation. Straightforward implementation of 3D rotation in hardware is very expensive. Rotation requires global communication and could cause memory contention while data is written back to the distributed memory modules. However, as the shear transformation capitalizes on the nearest neighbor connections, it lends itself to an extremely feasible multipipelined hardware implementation.

We use a distributed skewed volume buffer [7] in the Cube design. A voxel with space coordinates (x, y, z) is mapped onto the kth out of n memory modules by

$$k = (x + y + z) \bmod n \quad (0 \le k, x, y, z \le n - 1). \tag{16}$$

The data is distributed and skewed across the volume memory modules. By providing direct connections from each of the n Cube processing units to its dedicated volume memory module, this 3D skewed organization of the n^3 voxels allows conflict-free access to any directional beam of n voxels.

All types of shear transformations we have proposed, including 3D beam shear, guarantee rigid translation of beams on at least one major axis. According to our skewed memory design, a beam can be read out from the memory without conflict. After it is shifted (translated), it can also be written into the memory conflict-free, because each voxel within a beam is mapped to a different memory module. For example, after a 3D x-beam shear, the new y and z coordinates y' and z' are all the same across the whole beam (Eq. (15)) and the voxel coordinates within a beam differ only in x but are one unit apart between neighboring voxels. According to Eq. (16), a shifted beam is still distributed among n memory modules conflict-free. We utilize the neighboring connections between the Cube processing units to shift a beam across the Cube processing units, much like a barrel shifter (cf. [2]).

8. CONCLUSIONS AND SUMMARY

We have presented several decomposition methods for 3D arbitrary rotation using shear transformations. These decompositions include a four-pass 2D beam shear, a four-pass 2D slice shear, a three-pass 2D slice–beam shear, and a two-pass 3D beam shear. Our decompositions are advantageous over previous methods in various ways. First, all shear transformations of our decompositions are one-to-one mapping and thus are simpler than the pseudo shears [6]. Second, more decompositions are provided to allow implementation alternatives and trade-offs. Two-3D-beam shear decomposition is the best for single-pass resampling because it requires the smallest number of passes. However, it requires 3D interpolation for the resampling. When simplicity of interpolation is preferred, four-2D-beam shear decomposition with multiple resampling is the most appropriate because it only requires the simplest linear interpolation. On the other hand, three-2D-slice–beam shear reduces the number of shear passes by one, requiring slightly more expensive 2D bilinear interpolation for the resampling.

Furthermore, we have experimented with and discussed two different resampling methods: multiple-pass resampling and single-pass resampling. We have concluded that single resampling produces the highest quality results. It offers exactly the same rotation quality as the direct one-pass 3D rotation.

We have further shown a straightforward and efficient implementation of the shear transformation on the Cube architecture, as shearing capitalizes on the neighboring connections between the Cube processing units. In each shear pass, an entire beam can be accessed and processed in parallel.

ACKNOWLEDGMENTS

This work has been supported by NSF Grant MIP9527694 and ONR Grant N000149710402. We thank Alvy Ray Smith for carefully reading our initial paper and providing us with valuable comments. We also thank Amitabh Varshney, Frank Dachille, Kevin Kreeger, Ingmar Bitter, and Hong Qin for many productive discussions.

REFERENCES

1. E. Catmull and A. R. Smith, 3-D transformations of images in scanline order, in *SIGGRAPH'80 Proceedings, Comput. Graph.* **14**(3), 1980, 279–285.
2. D. Cohen and R. Bakalash, The conveyor: An interconnection device for parallel volumetric transformations, in *Advances in Graphics Hardware VI, 1992*, pp. 77–85.
3. M. Doggett and G. Hellestrand, A hardware architecture for video rate shading of volume data, in *International Symposium of Circuits and Systems, May 1995*, pp. 433–436.
4. R. A. Drebin, L. Carpenter, and P. Hanrahan, Volume rendering, in *SIGGRAPH '88 Proceedings, Comput. Graph.* **22**, 1988, 65–74.
5. T. Günther, C. Poliwoda, C. Reinhard, J. Hesser, R. Männer, H.-P. Meinzer, and H.-J. Baur, VIRIM: A massively parallel processor for real-time volume visualization in medicine, in *The 9th Eurographics Hardware Workshop, Sept. 1994*, pp. 103–108.
6. P. Hanrahan, Three-pass affine transforms for volume rendering, in *San Diego Workshop on Volume Visualization, Comput. Graph.* **24**(5), 1990, 71–78.
7. A. Kaufman and R. Bakalash, Memory and processing architecture for 3D voxel-based imagery, *IEEE Comput. Graph. Appl.* **8**(6), 1988, 10–23. [Also in Japanese, *Nikkei Comput. Graph.* **3**(30), 1989, 148–160]
8. A. W. Paeth, A fast algorithm for general raster rotation, in *Proceedings of Graphics Interface '86, May 1986*, pp. 77–81.

9. H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, The VolumePro real-time ray-casting system, in *SIGGRAPH'99 Proceedings*, *Comput. Graph.* **33**, 1999, 251–260.
10. H. Pfister and A. Kaufman, Cube-4: A scalable architecture for real-time volume rendering, in *Proceedings of 1996 Symposium on Volume Visualization*, *Oct. 1996*, pp. 47–54.
11. P. Schröder and J. B. Salem, Fast rotation of volume data on parallel architectures, in *IEEE Visualization '91 Proceedings*, *1991*, pp. 50–57.
12. A. R. Smith, Planar 2-pass texture mapping and warping, in *SIGGRAPH'87 Proceedings*, *Comput. Graph.* **21**, 1987, 263–272.
13. A. Tanaka, M. Kameyama, S. Kazama, and O. Watanabe, A rotation method for raster image using skew transformation, in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, *June 1986*, pp. 272–277.
14. T. Toffoli and J. Quick, Three-dimensional rotations by three shears, *Graph. Models Image Process.* **59**(2), 1997, 89–95.
15. G. Vezina, P. A. Fletcher, and P. K. Robertson, Volume rendering on the maspar MP-1, in *1992 Workshop on Volume Visualization*, *1992*, pp. 3–8.
16. C. M. Wittenbrink and A. K. Somani, 2D and 3D optimal parallel image warping, in *Seventh International Parallel Processing Symposium*, pp. 331–337, Assoc. Comput. Mach., New York, 1993.
17. C. M. Wittenbrink and A. K. Somani, Permutation warping for data parallel volume rendering, in *ACM SIGGRAPH Symposium on Parallel Rendering*, *November 1993*, pp. 57–60.