SHEAR-RESIZE FACTORIZATIONS FOR FAST MULTI-MODAL VOLUME REGISTRATION *

Ying Chen[1,2] *Pengwei Hao*[1,2] *Jian Yu*[3]

[1]Dept. of Computer Science
Queen Mary, Univ. of London
London, E1 4NS, UK
phao@dcs.qmul.ac.uk

[2]Center for Information Science
Peking University
Beijing, 100871, China
phao@cis.pku.edu.cn

[3]Dept. of Computer Science
Beijing Jiaotong University
Beijing, 100044, China
jianyu@center.njtu.edu.cn

ABSTRACT

Intensity-based methods work well for multi-modal image registration owing to their effectiveness and simplicity, but the computation for geometric transforms is a heavy load. To accelerate the transformation, in this paper, we present two shear-resize matrix factorizations for general 3D linear transformation, to factorize a transform matrix into three shears and a fixed resize, or four shears and a customizable resize. Shears can be implemented very fast by memory shift, and a resize can be done by axis-aligned resampling. The factorizations can be applied to both rigid-body and affine transformations. The experiments on MRI and CT volumes show that our method is 10 times faster than the naive transformation. The method is quite promising for hardware and parallel implementation, and is also valid for mono-modal image registration.

1. INTRODUCTION

Entropy-based registration method was first proposed by Viola [1] and Collignon [2] independently. The idea leads to much research and it's very popular with multi-modal medical image registration in recent years. It maximizes the mutual information between the two given images, and needs no complicated feature extraction.

For multi-modal volume registration, sometimes the minutiae are not easy to detect and not necessarily the same even if two volumes are well aligned. But according to information theory, the better the two volumes aligned, the better one volume can be explained by the other [3]. So the entropy-based method works well without concerning much about the minutiae.

Virtually, joint entropy and mutual information are two measures of correlation between volumes. Intensity-based methods repeatedly apply geometric transformations to the floating volume so as to find the best registration parameters to correlate with the reference volume. All the voxels in the floating volume must be relocated after the transformation. Therefore, geometric transformations are a heavy load for registration, and a fast transformation largely accelerates the correlation-based optimization algorithms for image registration.

Shears can be implemented faster by hardware-level memory shift than by voxel-by-voxel relocation. A resize can be done by simple axis-aligned resampling. So if a transform can be factorized into a series of shears and resizes, the transformation must be faster. By using some hardware or parallelism, the transform can be much faster.

Research on shear factorizations dates back to early 1980s, and a number of such algorithms have been published in the literature, such as Catmull and Smith's two-pass pseudo-shear (shearing coupled with scaling) factorization for 2D rotations [4], Paeth's three-shear factorization for 2D rotations [5], Hanrahan's three-pass pseudo-shear factorization for 3D affine transforms [6], Wittenbrink and Somani's three-shear factorization for 3D rotations [7], Chen and Kaufman's four-shear factorization for 3D rotations [8].

For volume registration, Thevenaz and Unser presented a four-pseudo-shear factorization for 3x3 matrices to interpolate volume data fast in three separate dimensions [9]. Cox and Jesmanowicz proposed a real-time 3D image registration for functional MRI by using 3D shear factorization of matrices [10]. The factorization has four pure shears, but only for orthogonal matrices.

In this paper, we propose two shear-resize factorizations to accelerate volume registration, which can be applied to both rigid-body and affine transforms. Some experiments on multi-modal medical volumes show their efficiency, and they can be run much faster if implemented with hardware.

2. ELEMENTARY TRANSFORMS

We have 3 cases of 2D slice shears: yz/x (shearing yz slices by using x coordinate), xz/y and xy/z, and 3 cases of

* This work was supported by the Foundation for the Authors of National Excellent Doctoral Dissertation of China, under Grant 200038.

1D beam shears: x/yz (shearing x beams by using y and z coordinates), y/xz and z/xy. They are denoted in this paper as $\boldsymbol{S}_S(x,a,b)$, $\boldsymbol{S}_S(y,a,b)$, $\boldsymbol{S}_S(z,a,b)$ (for slice shears) and $\boldsymbol{S}_B(x,a,b)$, $\boldsymbol{S}_B(y,a,b)$, $\boldsymbol{S}_B(z,a,b)$ (for beam shears), respectively. The matrices of the shears are respectively:

$$\begin{bmatrix}1&0&0\\a&1&0\\b&0&1\end{bmatrix}\begin{bmatrix}1&a&0\\0&1&0\\0&b&1\end{bmatrix}\begin{bmatrix}1&0&a\\0&1&b\\0&0&1\end{bmatrix}\begin{bmatrix}1&a&b\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\a&1&b\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\a&b&1\end{bmatrix}$$

Generally, slice shears are faster than beam shears because the data are better connected during each slice translation. If the volume data is in a 3D memory, all slice shears can be implemented the same fast by using memory shift. However, in our experiments on PC, physically only 1D memory architecture is provided. For 3D volume, our data is stored in row-major ordering, first by row (x), then by column (y) and last by layer (z), so the shears are not the same fast.

In order to capitalize on the neighboring connections in the 1D memory, the intrinsic functions of "memory-move" and "memory-copy" can be used to translate whole or partial xy-slices and x-dimensional row beams. For example, xz/y shear and xy/z shear can be implemented by memory shift while yz/x shear can only be done by voxel-by-voxel moving, which is time-consuming.

Table 1 shows the average running time for some elementary transforms and for naive rotation and naive general linear transformation on a PC (See Section 6).

Table 1. Average time for 3D transforms (ms)

transforms \ volume size		64^3	128^3	256^3	512^3
translations	x	0.26	5.73	46.97	361.55
	y	0.10	**3.94**	**36.52**	**293.40**
	z	**0.09**	4.13	50.88	445.38
	3D	0.22	5.61	58.19	462.84
slice shears	xy/z	**0.10**	**4.04**	**35.17**	**287.66**
	xz/y	0.19	7.20	51.82	366.14
	yz/x	1.98	183.28	1866.25	19261.0
beam shears	x/yz	**0.48**	**5.75**	**37.10**	**297.76**
	y/xz	1.37	14.46	115.89	2117.59
	z/xy	4.83	203.84	5422.13	34434.7
resizes	x	10.85	87.73	683.58	5435.71
	y	0.39	6.78	47.67	380.84
	z	**0.12**	**6.39**	**44.96**	**363.26**
	3D	11.37	91.33	709.62	5608.07
transposes	xy	**0.31**	5.95	**41.71**	**490.67**
	yz	0.38	**5.66**	48.21	1097.42
naive transforms	rotation	50.27	403.18	3570.23	29338.0
	linear	56.08	471.58	4252.21	34347.3

According to Table 1, another factor that affects the transformation speed is the usage of the CPU cache. For example, the slowest yz/x shear, the slowest z/xy shear and the slowest x resize can only be implemented by voxel-by-voxel moving and the CPU cache is not efficiently used.

To accelerate a transform, the problem is how to find a factorization with those fast elementary transforms.

3. SHEAR-RESIZE FACTORIZATIONS

As known in linear algebra, LU factorization ***A=PLU*** exists if and only if ***A*** is non-singular, where ***P*** is a permutation matrix, ***L*** is a unit lower triangular, and ***U*** is an upper triangular matrix. It is easy to prove that a so-called pseudo-permutation matrix (a unit upper triangular matrix whose elements are all 0, 1 or –1, still denoted as ***P***) can play the role of the permutation matrix to convert all leading principle minors into non-zeros, and the nonsingular triangular matrix ***U*** can be further factorized into a unit triangular matrix (denoted as ***U*** as well) and a diagonal matrix ***D***, and ***D*** can be flexibly moved to in front of, in between, or behind those unit triangular matrices.

In paper [11], it is proved that the factorization ***A=PLUS*** exists if and only if ***A*** is non-singular, where ***P*** is a pseudo-permutation matrix, ***L*** is a unit lower triangular, ***U*** is an upper triangular matrix whose diagonal elements can all be customized as long as the determinant is the same as det(***A***), and ***S*** is a special unit lower triangular matrix, such as a matrix with the first column or the last row off-diagonal elements being non-zeros. The customizable diagonal elements of ***U*** can be further extracted to form a diagonal matrix ***D*** and such a resize matrix's position can also be flexible. Thus, we can find a factorization of ***A=DPLUS***, where the axis-aligned scaling factors of the resize are almost freely customizable.

Based on the above analysis, we can find a shear-resize factorization of ***A=DPLU***, where the resize is fixed by the transform matrix and ***P***, and a shear-resize factorization of ***A=DPLUS***, where the resize is customizable. A customizable resize can be a uniform resize, a non-uniform resize, or a resize only in one dimension. To employ what factorization in a specific system depends on what resize is the fastest in the system.

For a non-singular 3D linear transform matrix, if all the leading principle minors are not zero, we have the factorization of ***L***, ***U*** and a fixed resize ***D*** as:

$$\begin{bmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{bmatrix}=\begin{bmatrix}\alpha_x&0&0\\0&\alpha_y&0\\0&0&\alpha_z\end{bmatrix}\cdot\begin{bmatrix}1&0&0\\l_1&1&0\\l_2&l_3&1\end{bmatrix}\cdot\begin{bmatrix}1&u_1&u_2\\0&1&u_3\\0&0&1\end{bmatrix}$$

or $\boldsymbol{A}=\boldsymbol{DLU}$, where $\alpha_x\alpha_y\alpha_z=\det(\boldsymbol{A}), \alpha_x=a_{11}$,

$$\alpha_x\alpha_y=\det\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}, u_1=a_{12}/\alpha_x,\ \ u_2=a_{13}/\alpha_x, l_1=a_{21}/\alpha_y,$$

$$l_2=a_{31}/\alpha_z, l_3=a_{32}/\alpha_z-l_2u_1,\ \ u_3=a_{23}/\alpha_y-l_1u_2.$$

If some off-diagonal minors are not zero, we can also find a factorization of a unit lower triangular ***L***, a unit upper triangular ***U***, a slice shear $\boldsymbol{S}_S$ or a beam shear $\boldsymbol{S}_B$, and a customizable resize ***D*** (without permutation matrix):

$$\boldsymbol{A} = \boldsymbol{DLUS}_S(x, s_1, s_2)$$

where $\alpha_x\alpha_y\alpha_z = \det(\boldsymbol{A}), u_1 = a_{12}/\alpha_x,\ u_2 = a_{13}/\alpha_x,$

$m_{31} = a_{12}a_{23} - a_{13}a_{22}, m_{33} = a_{11}a_{22} - a_{12}a_{21},$

$l_1 = \alpha_x(a_{22} - \alpha_y)/a_{12}/\alpha_y,\ u_3 = a_{23}/\alpha_y - l_1u_2,$

$s_2 = (\alpha_x\alpha_y - m_{33})/m_{31},\ s_1 = (a_{11} - a_{13}s_2 - \alpha_x)/a_{12},$

$l_2 = (a_{31} - a_{32}s_1 - a_{33}s_2)/\alpha_z,\ l_3 = a_{32}/\alpha_z - l_2u_1.$

or
$$\boldsymbol{A} = \boldsymbol{DLUS}_B(z, s_1, s_2)$$

where $\alpha_x\alpha_y\alpha_z = \det(\boldsymbol{A}),$

$m_{22} = a_{11}a_{33} - a_{13}a_{31}, m_{31} = a_{12}a_{23} - a_{13}a_{22}, m_{32} = a_{11}a_{23} - a_{13}a_{21},$

$u_2 = a_{13}/\alpha_x,\ s_1 = (a_{11} - \alpha_x)/a_{13},\ l_1 = (a_{21} - a_{23}s_1)/\alpha_y,\ l_2 = (a_{31} - a_{33}s_1)/\alpha_z,$

$u_3 = m_{32}/(\alpha_x\alpha_y),\ l_3 = (m_{22} - \alpha_x\alpha_z)/(\alpha_x\alpha_z u_3),\ u_1 = (m_{31} + a_{13}\alpha_y)/(\alpha_x\alpha_y u_3),$

$s_2 = (a_{12} - \alpha_x u_1)/a_{13}.$

A matrix made of two unit triangular matrices ***LU*** can be further factorized into 3 slice shears or 3 beam shears:

$$\boldsymbol{LU} = \boldsymbol{S}_S(x, a_1, a_2) \cdot \boldsymbol{S}_S(y, b_1, b_2) \cdot \boldsymbol{S}_S(z, c_1, c_2)$$

where $a_1 = l_1,\ a_2 = l_2,\ b_1 = u_1,\ b_2 = l_3,\ c_1 = u_2 - u_1u_3,\ c_2 = u_3 \cdot$

$$\boldsymbol{LU} = \boldsymbol{S}_B(z, a_1, a_2) \cdot \boldsymbol{S}_B(y, b_1, b_2) \cdot \boldsymbol{S}_B(x, c_1, c_2)$$

where $c_1 = u_1,\ c_2 = u_2,\ b_1 = l_1,\ b_2 = u_3,\ a_1 = l_2 - l_1l_3,\ a_2 = l_3$

It should be mentioned that symmetrically-permuted resizes are also resizes, symmetrically-permuted shears are shears as well. For instance, $\boldsymbol{P}_{xy}\boldsymbol{S}_S(x, a, b)\boldsymbol{P}_{xy}^T = \boldsymbol{S}_S(y, a, b) \cdot$

Now that some transposes are faster than some shears and resizes (see Table 1), we can also use two transposes and a faster shear or a faster resize to accelerate another time-consuming shear or resize. For example, let $\boldsymbol{D}_x(\alpha)$, $\boldsymbol{D}_y(\alpha)$ and $\boldsymbol{D}_z(\alpha)$ be the resizes in x, y and z directions and $\boldsymbol{P}_{xy}$ be the xy transpose, we have:

$$\begin{aligned}\boldsymbol{A} = \boldsymbol{DLU} &= \boldsymbol{DS}_S(x, a_1, a_2)\boldsymbol{S}_S(y, b_1, b_2)\boldsymbol{S}_S(z, c_1, c_2)\\ &= \boldsymbol{D}_z(\alpha_z)\boldsymbol{D}_y(\alpha_y)\boldsymbol{D}_x(\alpha_x)\boldsymbol{P}_{xy}^T\boldsymbol{P}_{xy}\boldsymbol{S}_S(x, a_1, a_2) \cdot\\ &\quad \boldsymbol{P}_{xy}^T\boldsymbol{P}_{xy}\boldsymbol{S}_S(y, b_1, b_2)\boldsymbol{S}_S(z, c_1, c_2)\\ &= \boldsymbol{D}_z(\alpha_z)\boldsymbol{D}_y(\alpha_y)\boldsymbol{P}_{xy}^T\boldsymbol{D}_y(\alpha_x)\boldsymbol{S}_S(y, a_1, a_2) \cdot\\ &\quad \boldsymbol{P}_{xy}\boldsymbol{S}_S(y, b_1, b_2)\boldsymbol{S}_S(z, c_1, c_2)\end{aligned}$$

After an xy transpose, we avoid the inefficient yz/x shear and the slow x-resize or 3D resize.

4. OPTIMIZATION FOR REGISTRATION

Based on Shannon's information theory, the mutual information *I* for two images *A* and *B* is defined as

$$I(A, B) = H(A) + H(B) - H(A, B)$$

where *H*(*A*) and *H*(*B*) are the entropy of image *A* and image *B*, respectively, and *H*(*A*, *B*) is the joint entropy of the two images.

An alternative normalized version of mutual information was proposed by Studholme et al [12], which is more robust when overlap area changes substantially. It is defined as $Y(A, B) = (H(A) + H(B))/H(A, B)$.

The larger *I*(*A*, *B*) or *Y*(*A*, *B*) is, the more image *A* can be predicted by image *B*, and thus the better they are registered. The latter measure is used in our experiments.

In order to obtain the joint entropy *H*(*A*, *B*), we first find the joint probability distribution, which is also called joint histogram in image processing. In intensity-based volume registration, a voxel in the floating volume is usually located between voxels in the reference volume after a geometric transformation. An interpolation method such as the trilinear partial volume distribution (PV) interpolation [2] can be used to update the joint histogram. It uses the interpolation weights to change the 8 positions (they may be the same) in the joint histogram. For faster interpolation, the nearest neighbor (NN) method can be used, which just changes one value in the joint histogram for each nearest transformed voxel in the floating volume and the method fits well for shear transformation.

The objective of volume registration is to find the optimal transform. Our optimization is to maximize the normalized mutual information. The optimal registration parameters (geometric transformation) are found by

$$\boldsymbol{T}^* = \arg \max_{\boldsymbol{T}} Y(A, \boldsymbol{T}B)$$

A rigid-body transform is a superposition of a rotation and a translation, so it needs 6 arguments in 3D. In some applications, rigid-body transformation is not enough. To take scaling into account, we have 3 more arguments: $\boldsymbol{T}(\theta_x, \theta_y, \theta_z, t_x, t_y, t_z, S_x, S_y, S_z) \cdot$

There are many optimization strategies can be applied to optimal geometric transform searching, among which downhill simplex method [13] doesn't require the derivative of the criterion function. For fast optimization, we partition the parameter space of affine transforms into three subspaces, three parameters for translation, three for rotation and three for resize. For each 3D subspace, we use downhill simplex method for searching.

5. EXPERIMENTS AND DISCUSSION

We use Visual C++ under the Windows 2000 operating system to run our experiments. The computer we use is a PC with an AMD CPU (128K L1 and 512K L2 on-chip cache, FSB 333MHz) and 512MB memory.

Our experiments are performed on a 256x256x109 MRI data set of a human head and a 256x256x113 CT data set of a cadaver head (www.siggraph.org/education/materials/). In order to make the measured experimental time comparable, we resample the data into cubes of 64^3, 128^3 and 256^3. The running time for the linear transformation is tested by the registration of the two volume data sets. We take the MRI data as the reference volume and the CT data as the floating.

In order to keep the data without loss during the shears, the shears are implemented in a double-sized cube (eight times the memory). After the shears, it's reasonable to cut it into the original size for fast registration. Thus, we implement the transpose only in valid data region. Memory reallocation is needed for some auxiliary work.

The average running time for transforms during registration are listed in Table 2. The shear-resize transformation is about 10 times faster than the naive.

Some slices and rendered images are shown in Figure 1. We can see that the two volumes are well registered.

Table 2. Time for 3D image registration (ms/iteration)

Transform \ Image size		64^3	128^3	256^3
Transform using Shear-Resize Factorization	Translation	0.30	6.39	54.07
	Resize	1.79	28.15	208.80
	3 shears	1.52	27.12	160.20
	Memory Reallocation	1.46	19.13	135.42
	Total	**5.06**	**80.79**	**558.50**
Naive Transformation		63.33	508.31	4163.69

6. CONCLUSION

Some 3D shear-resize factorizations are proposed to make intensity-based volume registration fast. A 3D linear transform can be factorized into three shears and a fixed non-uniform resize, or four shears and a customizable resize. The factorizations can be applied to both rigid-body and affine transformations. With some transpose, the method is 10 times faster than naive transformation. If shears and resize can all be implemented with hardware, affine transformation using shear-resize factorizations is remarkably faster than the naive transformation, and the hardware architecture is not complicated for pure shears and axis-aligned resize. Anyway, the quantitative evaluation and more efficient implementation of our shear-resize factorizations are needed for our future work.

7. REFERENCES

[1] P. Viola, and W.M. Wells III, "Alignment by maximization of mutual information," *Int. Conf. on Computer Vision*, pp. 16-23, 20-23 June 1995.

[2] A. Collignon, F. Maes, et al, "Automated multi-modality image registration based on information theory," in *Information Processing in Medical Imaging*, Kluwer, pp. 263-274, 1995.

[3] D. L. G. Hill, P. G. Batchelor, M. Holden, D. J. Hawkes "Medical Image Registration," *Physics in Medicine and Biology*, vol 46, pp 1-45, 2001.

[4] E. Catmull, and A.R. Smith, "3-D transformations of images in scanline order," *ACM Computer Graphics (SIGGRAPH)*, vol. 14, n. 3, pp. 279-285, 1980.

[5] A.W. Paeth, "A fast algorithm for general raster rotation," In *Proceedings of Graphics Interface*, pp. 77-81, 1986.

[6] P. Hanrahan, "Three-pass affine transforms for volume rendering," *Computer Graphics*, vol. 24, n. 5, pp. 71-77, 1990.

[7] C.M. Wittenbrink, and A.K. Somani, "Permutation warping for data parallel volume rendering," *ACM SIGGRAPH Symposium on Parallel Rendering*, pp. 57-60, 1993.

[8] B. Chen, and A.E. Kaufman, "3D Volume Rotation Using Shear Transformations," *Graphical Models*, vol. 62, n. 4, pp. 308-322, 2000.

[9] P. Thevenaz, and M. Unser, "Efficient geometric transformations and 3-D image registration," *Int. Conf. on Acoustics, Speech, and Signal Proc.*, vol.5, pp.2919-2922, 1995.

[10] R.W. Cox, and A. Jesmanowicz, "Real-Time 3D Image Registration for Functional MRI," *Magnetic Resonance Medicine*, 42, pp. 1014-1018, 1999.

[11] P. Hao, "Customizable triangular factorizations of matrices," *Linear Algebra and its Applications,* vol. 382, pp. 135-154, 2004.

[12] C. Studholme, D.L.G. Hill, and D.J. Hawkes, "An overlap invariant entropy measure of 3D medical image alignment," *Pattern Recognition*, vol. 32, pp. 71-86, 1999.

[13] W.H. Press, S.A. Teukolsky, et al, "*Numerical Recipes in C: The Art of Scientific Computing*," 2nd ed, Cambridge University Press, 1992.

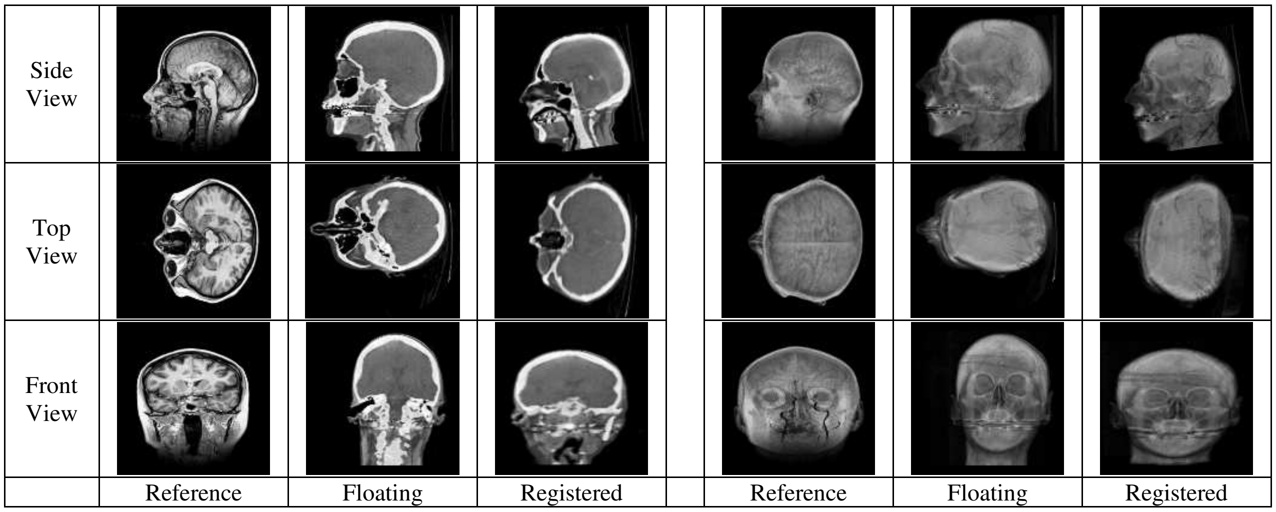

Figure 1. Three-views of the reference, the floating and the registered volume (slices and rendered images)