Shear-Resize Factorizations for Fast Image Registration*

Ying Chen[1] Pengwei Hao[1,2] Chao Zhang[2]
[1]Department of Computer Science, Queen Mary, University of London, E1 4NS, UK
[2]Center for Information Science, Peking University, Beijing, 100871, China
E-mail: phao@dcs.qmul.ac.uk, phao@cis.pku.edu.cn, chzhang@cis.pku.edu.cn

Abstract—**Owing to its effectiveness and simplicity, intensity-based method works well for registration of images. However, it needs a large amount of computation for geometric transformation. In this paper, we present two shear-resize matrix factorizations to accelerate the transformation. A transform matrix can be factorized into two shears and a fixed non-uniform resize, or three shears and a customizable resize. A customizable resize can be uniform in all dimensions or scaling just in one dimension. Shears can be implemented very fast by memory-shift, and a resize can be done by simple axis-aligned interpolation. The factorizations can be applied to both rigid-body and affine transformations. Their efficiency is performed by experiments on some standard test images and fingerprint images. The methods are quite promising for hardware implementation, and can also be extended to 3D or higher dimensional fast geometric transformation.**

Keywords – PLUS factorization; image registration; shear-resize factorization; transform acceleration.

I. 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 recently years. The method is an intensity-based method. It maximizes the mutual information between the two given images, and it needs no complex feature extraction.

In biometrics, fingerprint matching algorithms compare two given fingerprints and find the similarity. Most matching algorithms are performed in a feature (e.g. minutiae) space. However, minutiae are difficult to detect for some poor quality fingerprint images. Recently, correlation-based matching techniques attract much attention in order to avoid such minutiae extraction [3].

Virtually, joint entropy and mutual information are measures of correlation between images. The above methods repeatedly apply geometric transformations to the floating image so as to find the best registration parameters correlating with the reference image. After each geometric transformation, all the pixels (or voxels in 3D) in the floating image must be relocated. Therefore, geometric transformations are a heavy load for image registration, and a fast geometric transformation largely accelerates the correlation-based optimization algorithms for image registration.

Shears can be implemented faster by hardware-level memory-shift than by pixel-by-pixel relocation. A resize can be done by simple axis-aligned interpolation, which is faster than a generic image warp. Therefore, it must be faster if a transform is implemented with resizes and shears. By using hardware or parallelism, geometric transforms can be much faster. Additionally, the transformed data can be losslessly recovered if the determinant of the transform matrix is equal or greater than 1 [4, 14].

Research on shear factorization can be dated back to early 1980s, and a number of such algorithms have been published in the literature, such as Catmull and Smith's two-pass shear-scale factorization of 2D rotations [5], Paeth's three-shear factorization of 2D rotations [6], Hanrahan's three-pass shear-scale factorization of 3D affine transforms [7], Wittenbrink and Somani's three-shear factorization of 3D rotations [8], Chen and Kaufman's four-shear factorization of 3D rotations [9]

For image registration, Thevenaz and Unser presented a factorization of 4 shear-scaling matrices for 3x3 matrices to interpolate images fast in three separate dimensions [10]. Cox and Jesmanowicz proposed a real-time 3D image registration for functional MRI by using 3D shear factorization of matrices [11]. The factorization has 4 pure shears, but only for orthogonal matrices.

In this paper, we propose two shear-resize factorizations to accelerate image registration, which can be applied to both rigid-body and affine transformations. Some experiments on fingerprints and some standard test images show their efficiency, and they can be much faster if implemented with hardware.

II. SHEAR-RESIZE FACTORIZATIONS

For simple description, we name the pure shears in x and y directions as *x-shear* and *y-shear*, respectively. In matrix notations, x-shear and y-shear are respectively:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad (2)$$

If a 2D memory is used for an image, shearing in both directions are the same fast, and both x-shear and y-shear can be done by memory-shift. However, if the 2D data is a one-

* Partially supported by the Foundation for the Authors of National Excellent Doctoral Dissertation of China (200038) and a National Key Basic Research Project of China (2004CB318005).

dimensional row packed array in memory, x-shear can be done by row memory shifting, but y-shear cannot.

With theorems in linear algebra and the theorems in [4, 14], it is easy to prove that a nonsingular matrix $\boldsymbol{A}$ can be factorized into two shears $\boldsymbol{L}$ and $\boldsymbol{U}$, a fixed non-uniform resize $\boldsymbol{D}$, and a possible simple shear $\boldsymbol{P}$, $\boldsymbol{A} = \boldsymbol{DPLU}$, or into three shears, $\boldsymbol{L}$, $\boldsymbol{U}$ and $\boldsymbol{S}$, a customizable resize $\boldsymbol{D}$, and a possible simple shear $\boldsymbol{P}$, $\boldsymbol{A} = \boldsymbol{DPLUS}$. In mathematics, $\boldsymbol{P}$ is a unit upper triangular matrix whose elements are all 0 or 1, $\boldsymbol{L}$ is a unit lower triangular matrix, $\boldsymbol{U}$ is a unit upper triangular matrix, $\boldsymbol{S}$ is a special unit lower triangular matrix with only single row or single column off-diagonal non-zero elements, $\boldsymbol{D}$ is a diagonal matrix of scaling factors for axis-aligned resize. The scaling factors of a customizable resize can be freely chosen by users as long as the determinant is kept the same.

For almost every nonsingular matrix, $\boldsymbol{P}$ can be the identity matrix, and then the factorization can be of two shears and one fixed non-uniform resize, $\boldsymbol{A} = \boldsymbol{DLU}$, or three shears and one customizable resize, $\boldsymbol{A} = \boldsymbol{DLUS}$.

An *N*-by-*N* matrix made of $\boldsymbol{LU}$ can be further factorized into *N* single-row or single-column special matrices [4, 14], which actually are pure shears.

For a 2D affine transform, the translation can be done simply after the linear transform. By using homogeneous coordinates, the factorization is formulated as:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

In case of $\det \boldsymbol{A} \neq 0$, we have following shear-resize factorizations for the linear transform.

If $a_{11} \neq 0$, the factorization of two shears and one fixed non-uniform resize (*2-shear-resize* for short) for the linear transform is:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \tag{4}$$

where $\alpha_x = a_{11}$, $\alpha_x \alpha_y = \det \boldsymbol{A}$, $a = a_{21} / \alpha_y$, $b = a_{12} / \alpha_x$.

If $a_{22} \neq 0$, the 2-shear-resize factorization is:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix} \cdot \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \tag{5}$$

where $\alpha_y = a_{22}$, $\alpha_x \alpha_y = \det \boldsymbol{A}$, $a = a_{21} / \alpha_y$, $b = a_{12} / \alpha_x$.

If some off-diagonal element is not zero, we can have factorizations of three shears and a customizable resize (*3-shear-resize* for short):

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix} \cdot \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix} \tag{6}$$

where $\alpha_x \alpha_y = \det \boldsymbol{A}$, $a = \alpha_y (a_{11} - \alpha_x) / \alpha_x / a_{21}$, $b = a_{21} / \alpha_y$, $c = (a_{22} - \alpha_y) / a_{21}$ (if $a_{21} \neq 0$).

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix} \tag{7}$$

where $\alpha_x \alpha_y = \det \boldsymbol{A}$, $a = \alpha_x (a_{22} - \alpha_y) / \alpha_y / a_{12}$, $b = a_{12} / \alpha_x$, $c = (a_{11} - \alpha_x) / a_{12}$ (if $a_{12} \neq 0$).

The scaling factors of a customizable resize can be customized in an efficient way, uniform or non-uniform. If $\alpha_x = \alpha_y = \alpha$, the resize is uniform. If $\alpha_x = 1$ or $\alpha_y = 1$, the resize can be done in only one dimension. If $\det \boldsymbol{A} = 1$, we can just have 3 pure shears by setting $\alpha_x = \alpha_y = 1$.

What to employ in a specific system, a uniform resize, a non-uniform resize, or a resize only in one dimension, depends on what resize is the fastest in the system.

III. Optimization Objectives

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

$$I(A, B) = H(A) + H(B) - H(A, B) \tag{8}$$

where *H*(*A*) and *H*(*B*) are the entropy of image *A* and *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 at al [12], which is more robust when overlap area changes substantially. It is defined as

$$Y(A, B) = \frac{H(A) + H(B)}{H(A, B)} \tag{9}$$

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. Therefore, *I*(*A*, *B*) and *Y*(*A*, *B*) are two valid criterion functions for optimization objectives, and the latter 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 image registration, a pixel in the floating image is usually located between pixels in the reference image 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 4 positions (they may be the same) in the joint histogram. For faster interpolation, a nearest neighbor (NN) method can be used, which just changes one value in the joint histogram for each moved pixel in the floating image and the method fits well for shear transformation.

IV. Optimization

The objective of intensity-based image registration is to find the optimal geometric transform. Our optimization is to maximize the normalized mutual information. The optimal registration parameters (geometric transform $\boldsymbol{T}^*$) are found from

$$\boldsymbol{T}^* = \arg \max_{\boldsymbol{T}} Y(A, \boldsymbol{T}B) \tag{10}$$

A rigid-body transform is a superposition of a rotation and a translation, so we have only 3 arguments for a rigid-body transform in 2D $\boldsymbol{T}$: $\boldsymbol{T}(\theta, t_x, t_y)$. To take scaling into account, we have 2 more arguments: $\boldsymbol{T}(\theta, t_x, t_y, S_x, S_y)$. For an arbitrary 2D affine transform, there are 6 parameters to be found:

$\boldsymbol{T}(a_{11},a_{21},a_{21},a_{22},t_x,t_y)$, where the symbols are consistent to those in Section 2.

Any 2D linear transform is equivalent to a concatenation of a rotation, a shear and a scaling/resize matrix. We use following factorization to combine the transforms into an arbitrary 2D linear transform and to analyze what factors contribute to a transform:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix} \cdot \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad (11)$$

where $\alpha_y = \sqrt{a_{21}^2 + a_{22}^2}$, $\alpha_x = \det \boldsymbol{A} / \alpha_y$, $s = a_{11}a_{21} + a_{12}a_{22}$, $\cos\theta = a_{22}/\alpha_y$, $\sin\theta = a_{21}/\alpha_y$.

There are many optimization strategies can be applied to optimal geometric transform searching, among which Powell's direction set method and downhill simplex method [13] don't require the derivative of the criterion function. For fast optimization, we partition the parameter space of affine transforms into several subspaces, two parameters for translation, two for scaling, one parameter for rotation, and one for shearing. Similar to Powell's method, we search in the subspaces sequentially and iteratively. For one-dimensional subspaces, we use golden section method [13] for searching. For higher-dimensional subspaces, we use downhill simplex method for searching.

V. Experiments And Discussion

We program with Visual C++ under Windows 2000 on a PC with a 2.5GHz CPU and 512MB memory. We use the intrinsic memory-move function for x-shear, but y-shear in our program can only be done pixel by pixel.

The average running time for some elementary transforms, x-shear, y-shear, and naive rotation, are measured with images of various sizes, as listed in Table I. In order to keep all the data after transformation, we use double-sized memory. From the table, we can see that shears are much faster than naive rotation, and x-shear is even faster than y-shear in our one-dimensional memory system. This leads to a conclusion that two-dimensional memory hardware makes transformation remarkably faster.

Our registration experiments are performed on some standard test images with some simulated geometric transforms and some fingerprints in pairs with relative geometric distortion. The standard test images are Lena, Barbara, and Peppers, all of size 256x256. The fingerprints are from Fingerprint Verification Competition website (http://bias.csr.unibo.it/fvc2004), all of size 388x374.

The average running time (ms/iteration) for fingerprint registration is listed in Table II for comparison. Two pairs of the original images and the registered images are shown in Figure 1 and 2.

Table II show that the transformation using the 3-shear-resize factorization is nearly 5 times faster than naive transformation. Added with the time for correlation and optimization, the whole registration time with shear-resize factorization is still more than twice faster.

Table I. Time for elementary transforms (ms)

transform	128x128	256x256	512x512	776x748
x-shear	0.0180	0.0671	0.2995	2.5346
y-shear	0.1001	0.4056	3.6122	7.6119
rotation	2.874	11.717	46.637	103.639

Table II. Time for fingerprint registration (ms/iteration)

Operation	Pair 1	Pair 2	Pair 3	Pair 4	Pair 5
T	29.41	29.91	31.21	29.56	32.60
S	6.06	6.29	6.42	6.10	5.83
T + G	33.95	34.93	34.79	32.98	37.07
S + G	14.52	15.51	14.21	13.14	14.80

T – time for naive transformation, G – time for registration
S – time for transformation using 3-shear-resize factorization

(a) Reference (b) Floating (c) Registered

Figure 1. Test fingerprint – Pair 1

(a) Reference (b) Floating (c) Registered

Figure 2. Test fingerprint – Pair 2

The naive transformation time in Table II is only about 1/4 of the naive rotation time in Table I for fingerprint images. This is because we don't really rotate the images and we just reversely trace back to find pixels in the original floating image if the pixel is in the overlap area. However, shear-resize factorization is applied to real transformation with double-sized memory.

For the experiments on some standard test images, we take the original as the reference images, and make the distorted floating images by using some geometric transforms, including translation, rotation and scaling. All three pairs are of size 256x256. The original, the floating and the registered images are shown in Figure 3-5, and the average running time is listed in Table III.

Table III shows that the transformation using shear-resize factorizations is about twice faster than the naive transformation. The shearing time is mainly for y-shears, the resize time is also a big load, and uniform resize takes nearly the same time as non-uniform resize. If shears and resize can all be implemented with hardware, the speed-up will be much larger.

For computational complexity, as in our experiments with one-dimensional memory, it is $O(n)$ for x-shears and $O(n^2)$ for y-shears. Shearing of each line only needs one floating-point multiplication, but naive transformation of each pixel needs at least 4 floating-point multiplications.

(a) Reference (b) Floating (c) Registered

Figure 3. Test Image – Lena

(a) Reference (b) Floating (c) Registered

Figure 4. Test Image – Barbara

(a) Reference (b) Floating (c) Registered

Figure 5. Test Image – Peppers

VI. CONCLUSION

Two shear-resize factorizations are proposed to make intensity-based image registration fast. A 2D linear transform can be factorized into two shears and a fixed non-uniform resize, or three shears and a customizable resize. The factorizations can be applied to both rigid-body and affine transformations. If shears and resize can all be implemented with hardware, affine transforms using shear-resize factorizations are remarkably faster than the naive transformation, and the hardware architecture is not complicated for pure shears and axis-aligned resize.

By using shear-resize factorizations, transformation interpolation also brings some error, which will be explored in our future.

REFERENCES

[1] P. Viola and W.M. Wells III, "Alignment by maximization of mutual information," 5th *International Conference on Computer Vision*, 20-23 June 1995, pp. 16-23.

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

[3] Maltoni D., Maio D., Jain A. K., and Prabhakar S., "Handbook of Fingerprint Recognition," Springer, 2003.

[4] P. Hao, and Q. Shi, "Matrix Factorization for Reversible Integer Mapping," *IEEE Transactions on Signal Processing,* 2001 Vol. 49, No. 10, pp. 2314-2324.

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

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

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

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

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

[10] P. Thevenaz, and M. Unser, "Efficient geometric transformations and 3-D image registration," *ICASSP* 1995, vol.5, pp. 2919-2922.

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

[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, W.T. Vetterling, and B.P. Flannery, "Numerical Recipes in C: The Art of Scientific Computing", Cambridge University Press, 1992.

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

TABLE III. AVERAGE TIME FOR REGISTRATION OF STANDARD TEST IMAGES (ms/iteration)

Image	Naive transformation		3-shear-resize factorization				2-shear-resize factorization			
	T	T+G	S	Z	S+Z	S+Z+G	S	Z	S+Z	S+Z+G
Lena	15.91	19.71	5.05	3.15	8.20	13.35	4.43	2.92	7.35	12.31
Barbara	15.22	18.91	4.60	3.05	7.65	13.28	4.75	3.21	7.96	12.59
Peppers	14.73	18.98	4.19	3.03	7.22	12.52	3.83	3.18	7.01	12.48

T – time for naive transformation, G – time for registration, S – time for shears, Z – time for resize