IEEE Int. Conf. Image Processing, CD paper no. 200, Santa Barbara CA, USA, October 1997.

SEPARABLE LEAST-SQUARES DECOMPOSITION OF AFFINE TRANSFORMATIONS

Philippe Thévenaz and Michael Unser
National Institutes of Health
Bethesda MD 20892–5766, USA

ABSTRACT

We decompose 2D and 3D invertible affine transformations into a series of elementary shears and skews along the coordinate axis. These elementary operations are one-dimensional, which disposes of the need for non-separable interpolation methods and allows higher-quality approaches for a given processing time. We propose a framework where the transformed function is projected on the original function space in a least-squares sense, which ensures optimal anti-aliasing filters.

1. INTRODUCTION

The Paeth algorithm [1] introduced a separable approach to the rotation of an image, which has several benefits. First, data can be processed one line at a time, which is economical in terms of memory requirements, especially for very large images [2]. Also, this is well adapted to off-the-shelf DSP architectures [3]. Second, 1D processing is inexpensive compared to non-separable processing in higher dimensions [4]. Therefore, it is possible to achieve faster results for a given quality, or better results within a given computation time [5].

In this paper, we extend the Paeth algorithm in a new way that allows us to perform general affine transformations, not only in 2D but also in 3D, while using 1D operations only. Volumetric imaging is particularly relevant to biomedical applications, where the amount of data available in any single acquisition can be considerable; the benefit of using separable transformations is even more pronounced there than in 2D.

The 1D transformation $f(a\,x+b)$ is an essential operation in our framework. To achieve high quality, we perform this operation in the least-squares sense developed in [6]. An important element of this methodology is an expansion kernel that can be pre-computed for each scaling factor a. We show how this pre-computation is best performed in the Fourier domain, an approach that was not pursued before.

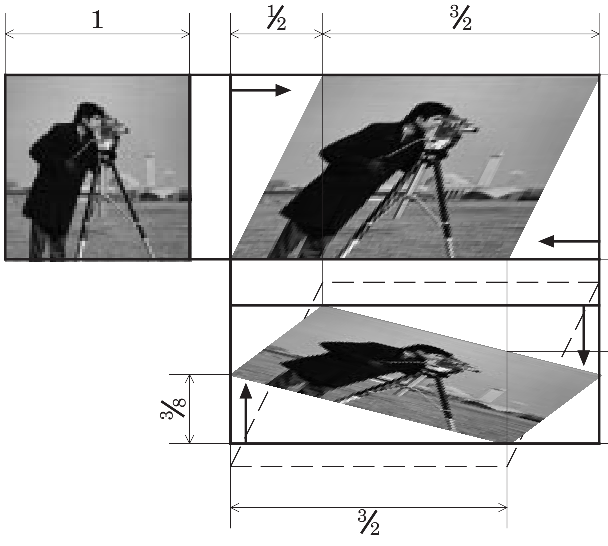

Figure 1: Example of an 2D two-pass affine transformation.

The decomposition is $\begin{pmatrix} \frac{3}{2} & \frac{1}{2} \\ \frac{-3}{8} & \frac{3}{8} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{-1}{4} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{3}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}$.

2. FACTORIZATION

We present several ways to decompose an affine transformation into 1D operations. One important aspect is the number of elementary operations involved. In 2D, we show that selecting from two-pass and three-pass approaches is sufficient to decompose any invertible affine transformation matrix. In 3D, we need to select from three-pass and four-pass decompositions to achieve the same result.

2.1. Affine decomposition in 2D

The Paeth algorithm is restricted to the decomposition of a pure rotation matrix into a series of separable operations along the rows and columns of an image. General affine transformations can be decomposed in a similar way. The simplest case involves a series of two operations only, one along the rows, and one along the columns, as can be seen in Figure 1. The sequence order is arbitrary, and this freedom of choice results in two different solutions, one of which has been presented in [7]. The two solutions are

$$\begin{aligned} \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} &= \begin{pmatrix} \alpha_1 & \beta_1 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 1 & 0 \\ \gamma_1 & \delta_1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \phi_1 \end{pmatrix} \right) + \begin{pmatrix} \varepsilon_1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ \gamma_2 & \delta_2 \end{pmatrix} \left(\begin{pmatrix} \alpha_2 & \beta_2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \varepsilon_2 \\ 0 \end{pmatrix} \right) + \begin{pmatrix} 0 \\ \phi_2 \end{pmatrix}, \end{aligned} \tag{1}$$

with

$$\begin{cases} \begin{bmatrix} \alpha_1 & \beta_1 \\ \gamma_1 & \delta_1 \end{bmatrix} = \begin{bmatrix} (ad-bc)/d & b/d \\ c & d \end{bmatrix} & \begin{bmatrix} \varepsilon_1 \\ \phi_1 \end{bmatrix} = \begin{bmatrix} (ed-fb)/d \\ f \end{bmatrix} \\ \begin{bmatrix} \alpha_2 & \beta_2 \\ \gamma_2 & \delta_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c/a & (ad-bc)/a \end{bmatrix} & \begin{bmatrix} \varepsilon_2 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} e \\ (fa-ec)/a \end{bmatrix} \end{cases}. \tag{2}$$

In order to understand the basic operation, it is enough to consider any one of the elementary operations. We decide to take as example

$$\begin{pmatrix} \alpha_2 & \beta_2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \varepsilon_2 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha_2\,x + \beta_2\,y + \varepsilon_2 \\ y \end{pmatrix}. \tag{3}$$

For this particular elementary transformation, it happens that the image can be processed row by row since the y coordinate is invariant. The basic operation is then to transform a particular row $f(x)$ into $f(a\,x+b)$. In our example, we have $a=\alpha_2$ and $b=\beta_2\,y+\varepsilon_2$. Note that the scaling factor a does not depend on the coordinate y, although the translation b does.

The relative magnitude of the quantities $|a|$ and $|d|$ are critical because they appear as denominators in the solutions. If they are small compared to the determinant of the affine transformation, the elementary transformations will introduce a substantial scaling. The worst case is $a=d=0$, for which there is no two-pass decomposition. In addition, when applied to a rotation matrix, the two-pass decomposition has the disadvantage that some unnecessary aliasing is introduced [8], essentially because the diagonal terms α_i and δ_i are not unity. Disregarding translation for the moment, we propose the following two solutions to these problems, at the cost of one additional (1D) transformation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \alpha_1'' & \beta_1'' \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \gamma_1 & \delta_1 \end{pmatrix}\begin{pmatrix} \alpha_1' & \beta_1' \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \gamma_2'' & \delta_2'' \end{pmatrix}\begin{pmatrix} \alpha_2 & \beta_2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \gamma_2' & \delta_2' \end{pmatrix}. \tag{4}$$

These two solutions are under-determined because each one has two extra coefficients that provide two additional degrees of freedom. The limit to this freedom can be summarized by the necessary constraint

$$ad-bc = \alpha_1''\,\delta_1\,\alpha_1' = \delta_2''\,\alpha_2\,\delta_2', \tag{5}$$

where the arbitrary selection of a pair of coefficients determines the third one. For example, a natural choice in the case of a rotation leads to $\alpha_1'' = \delta_1 = \alpha_1' = 1$ or to $\delta_2'' = \alpha_2 = \delta_2' = 1$, which corresponds to the Paeth algorithm. We propose to use either one of the following two strategies to satisfy the constraint (5): distribute the determinant evenly, in a cubic root fashion (best accuracy); alternatively, set two diagonal coefficients to a unit value, and load the only remaining coefficient with the full weight of the determinant (best speed). Once the diagonal coefficients have been selected, the remaining coefficients are fully constrained. For example, $\beta_1' = \alpha_1'(d - \delta_1)/c$, $\beta_1'' = (a - \alpha_1'\alpha_1'')/c$ and $\gamma_1 = c/\alpha_1'$. The value of these coefficients does not change if translation is brought back into play.

As announced, the decomposition (4) deals gracefully with the case $a = d = 0$, and a rotation matrix can be decomposed without any scaling. In return, the case $b = c = 0$ does not lend itself to a three-pass decomposition, which means that an affine transformation close to identity (or essentially diagonal) leads to an unstable three-pass decomposition. If the transformation is a small rotation, the stability is recovered because trigonometric relations ensure that β_1' and β_1'' tend toward zero when the rotation angle vanishes. This is not the case for a general affine transformation. Therefore, we prefer to use a two-pass decomposition when the transformation is close enough to identity, because the aliasing is negligible in this case. Finally, a completely general affine transformation includes translation as well. It is trivial to incorporate this translation into the two last elementary operations. Alternatively, one may express the transformation as an 3x3 homogenous matrix [7] and use the results of the next Section.

2.2 Affine decomposition in 3D

In 3D, a general affine transformation can be decomposed as a three-pass operation (six solutions) or a four-pass operation (six solutions, too). Qualitatively, the behavior is very similar to the 2D case: again, it is enough to apply a series of 1D affine transformations to each row (or column, or z-line) of the volume; also, the scaling is the same for all rows, while the translation differs from row to row. Similarly to the 2D two-pass scheme, the 3D three-pass decomposition introduces some unnecessary aliasing, but it is stable when the transformation is nearly identity, or nearly diagonal. It is fully constrained. The 3D four-pass decomposition is unstable near the identity, but does not introduce extra intermediate scaling when the determinant is unity. It offers three degrees of freedom that pertain to the diagonal coefficients, whose product must equal the determinant of the transformation. Also, in 3D as in 2D, there are special invalid cases, but we can show that at least one out of the twelve solutions is stable if the affine transform to decompose is invertible. We give these twelve solutions in the Appendix.

We conjecture that, in a space of dimension N, it is possible to decompose any full-rank affine transformation in either N or $N+1$ separable passes, with $N!$ solutions each.

3. LEAST-SQUARES INTERPOLATION

We must complete each pass of a decomposition before we can proceed with the next pass. Because of this sequential aspect, we need to perform each operation with the greatest accuracy to minimize the propagation of errors. We present below a way to perform optimal 1D affine transformations in an optimal least-squares sense.

3.1. Affine Algorithm in 1D

We use the methodology presented in [6] for computing $g(x) = f(ax+b)$. It uses a B-spline representation of signals and performs the least-squares projection of $f(ax+b)$ on $V^{(n)}$, the space of polynomial splines of degree n. In contrast with simple interpolation, this approach ensures that the optimal anti-aliasing filter is used. A brief description of the algorithm follows (see [9] for implementation details and further explanations).

I) Given a scaling factor a, pre-compute the expansion kernel

$$\xi_a^{(n)}(x) = \left\langle \beta^{(n)}(at), \beta^{(n)}(t+x) \right\rangle_t . \tag{6}$$

This first step is data-independent, and needs to be performed only once for the whole set of rows of the image or volume for which a is constant

II) Given a sequence of signal samples $f(k)$, find its B-spline coefficients $c(k)$. This is achieved by applying to $f(k)$ a filter $1/B^{(n)}(z)$ and satisfies

$$f(k) = \sum_{i=-\infty}^{\infty} c(i)\, \beta^{(n)}(k-i), \tag{7}$$

where the recursive digital filter of degree n is

$$B^{(n)}(z) = \sum_{k=-\infty}^{\infty} \beta^{(n)}(k)\, z^{-k}. \tag{8}$$

III) Given a translation b, compute the coefficients $d(l)$. This new sequence incorporates the change of rate expected under scaling

$$d(l) = \sum_{k=-\infty}^{+\infty} c(k)\, \xi_a^{(n)}(ak - l - b). \tag{9}$$

IV) Finally, reconstruct the signal by applying a digital post-filter $B^{(n)}(z)/B^{(2n+1)}(z)$ to $d(l)$.

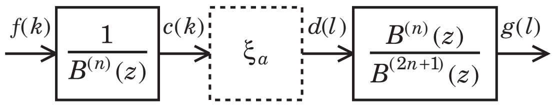

Figure 2: Overview of the 1D least-squares rescaling algorithm.

In this algorithm, all the kernels have a finite-support, which ensures that there is a way to perform the computations in an exact fashion (the sums involve a finite number of terms if $f(x)$ is finite-support). Moreover, the zeroes of the recursive filters being real, the filtering operations can be implemented in a very efficient way. The only difficult step is to get a closed form for the expansion kernel $\xi_a^{(n)}(x)$, which is built of many polynomial regions that do not coincide with the grid of integers. In [6], we proposed a Gaussian approximation to this kernel. Here, we pursue a different approach.

3.2. Kernel computation

The kernel $\xi_a^{(n)}(x)$ given in (6) can be rewritten as the convolution of a B-spline and a scaled B-spline, because $\beta^{(n)}$ is a symmetric function. In the Fourier domain, this amounts to multiply together two sinc functions, a canonical one and a stretched one. Hence, the closed form of $\hat{\xi}_a^{(n)}(\omega)$ is easy to obtain

$$\hat{\xi}_a^{(n)}(\omega) = \left(\mathrm{sinc}(\omega/2)\,\mathrm{sinc}(\omega/2a)\right)^{n+1} / |a|. \tag{10}$$

This function has an infinite frequency support that decays as fast as ω^{-2n-2}.

In this paper, we use an m-point IDFT (inverse discrete Fourier transform) to recover $\tilde{\xi}_a^{(n)}(k\,\Delta x)$, which gives us a discrete approximation of the expansion kernel. To determine the spatial sampling step Δx, we first note that the support of $\xi_a^{(n)}(x)$ is finite, given by

$$\begin{aligned} \mathrm{support}\left\{\xi_a^{(n)}(x)\right\} &= \mathrm{support}\left\{\beta^{(n)}(x)\right\} + \mathrm{support}\left\{\beta^{(n)}(ax)\right\} \\ &= (n+1)\left(1 + 1/|a|\right) \end{aligned}. \tag{11}$$

Once m is selected, the spatial resolution can be no better than $\Delta x \geq (n+1)(1+|a|)/m|a|$ without introducing spatial aliasing or overlap. In turn, this spatial resolution determines the bandwidth of the discrete approximation.

4. ERROR ANALYSIS

The IDFT introduces two types of error in the computation of the kernel. First, the finiteness of the set of samples requires one to approximate the kernel by a band-limited version. Second, the discreteness of the set requires one to interpolate the expansion kernel itself. We analyze below the influence of these two types of approximation.

4.1. Band-limited kernel

We compute the amount of energy that is ignored when band-limiting $\hat{\xi}_a^{(n)}(\omega)$

$$\begin{aligned}\left(\varepsilon_a^{(n)}\right)^2 &= \frac{1}{2\pi}\int_{-\infty}^{-\frac{m}{2}\Delta\omega}\left|\hat{\xi}_a^{(n)}(\omega)\right|^2 d\omega + \frac{1}{2\pi}\int_{\frac{m}{2}\Delta\omega}^{\infty}\left|\hat{\xi}_a^{(n)}(\omega)\right|^2 d\omega \\ &= \frac{1}{\pi}\int_{\frac{m}{2}\Delta\omega}^{\infty}\left(\frac{\sin(\omega/2|a|)}{\omega/2|a|}\right)^{2n+2}\left(\frac{\sin(\omega/2)}{\omega/2}\right)^{2n+2} d\omega\end{aligned}. \qquad (12)$$

Since $|\sin(\theta)| \le 1$, this error is bounded as in

$$\left(\varepsilon_a^{(n)}\right)^2 \le \frac{2^{8n+7}}{(4n+3)\pi} a^{2n+2}(m\,\Delta\omega)^{-4n-3}. \qquad (13)$$

Selecting a critical frequency sampling $\Delta\omega$ such that there is no spatial aliasing leads to

$$\Delta\omega = \frac{2\pi}{(n+1)(1+1/|a|)} \qquad (14)$$

and

$$\varepsilon_a^{(n)} \le \frac{(2(n+1)/\pi)^{2n+2}}{\sqrt{(n+1)(4n+3)}}\frac{(1+|a|)^{2n+3/2}}{|a|^{n+1/2}} m^{-2n-3/2} = C_\varepsilon\, m^{-2n-3/2}. \qquad (15)$$

The cases where the scaling factor $|a|$ takes a value significantly different from unity are rare in practice. Finally, the error of approximation due to band-limiting is essentially dependent on the term $m^{-2n-3/2}$, which is consistent with the decay of $\hat{\xi}_a^{(n)}(\omega)$.

4.2. Kernel re-sampling and interpolation

Our procedure yields a kernel given by a set of samples. Since we have to evaluate this kernel for a continuous argument $x = ak - l - b$, we need to introduce interpolation for recovering the values between the samples. For convenience purposes, we fit through the samples a B-spline model of degree p, where p can be chosen independently from n.

Intuitively, the use of a small number m of samples for the representation of $\tilde{\xi}_a^{(n)}(x)$ calls for a high interpolation degree p if one desires to reach the best quality results for a given computational cost. Conversely, a large number of samples can compensate for a low-order interpolation scheme. The trade-off between p and m will depend on the amount of data because it determines how many times one needs to interpolate the expansion kernel, while the time needed for computing the IDFT is spent but once. For images and volumes, we typically use nearest neighbor interpolation, for which we have $p = 0$.

An expression similar to (15) bounds the re-sampling and interpolation error $\gamma_a^{(p)}$ [10]. As expected, this second type of error depends essentially on $1/m$, with some proportional constant C_γ that incorporates the norm of the $(p+1)$-th derivatives of $\xi_a^{(n)}(x)$

$$\gamma_a^{(p)} \le C_\gamma\, m^{-p-1}. \qquad (16)$$

4.3. Overall error

By a triangular inequality argument, the total kernel error is bounded by the sum $\varepsilon_a^{(p)} + \gamma_a^{(p)}$. The comparison of (15) and (16) shows that the driving term is the interpolation degree p rather than band-limiting, because the constants C_ε and C_γ play no role in the comparison when $1/m$ is sufficiently small.

It is important to remember that the present Section deals with errors in the *kernel* approximation. The analysis of their influence on the overall *data* least-squares interpolation process is more difficult. Qualitatively, the kernel $\xi_a^{(n)}(x)$ is a very smooth function, even though it is not band-limited. Hence, we expect the constant C_γ to be small and the influence of p to be minor, because we can choose to perform an IDFT with a sufficiently large number of points for the error in the kernel approximation to be negligible. Therefore, we can be arbitrarily close to the true least-squares projection of $f(ax+b)$ on $V^{(n)}$. In short, we have replaced the problem of interpolating an unknown, possibly agitated function f by the problem of interpolating a known, smooth kernel ξ.

APPENDIX

$$\begin{cases} R = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ b_{21} & b_{22} & b_{23} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \\ c_{31} = r_{31} \quad c_{32} = r_{32} \quad c_{33} = r_{33} \\ b_{21} = \dfrac{r_{21}r_{33} - r_{23}r_{31}}{r_{33}} \quad b_{22} = \dfrac{r_{22}r_{33} - r_{23}r_{32}}{r_{33}} \quad b_{23} = \dfrac{r_{23}}{r_{33}} \\ a_{11} = \dfrac{\mathrm{Det}(R)}{b_{22}r_{33}} \quad a_{12} = \dfrac{r_{12}r_{33} - r_{13}r_{32}}{b_{22}r_{33}} \quad a_{13} = \dfrac{r_{13}r_{22} - r_{12}r_{23}}{b_{22}r_{33}} \end{cases} \qquad (17)$$

$$\begin{cases} R = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_{31} & b_{32} & b_{33} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix} \\ c_{21} = r_{21} \quad c_{22} = r_{22} \quad c_{23} = r_{23} \\ b_{31} = \dfrac{r_{22}r_{31} - r_{21}r_{32}}{r_{22}} \quad b_{32} = \dfrac{r_{32}}{r_{22}} \quad b_{33} = \dfrac{r_{22}r_{33} - r_{23}r_{32}}{r_{22}} \\ a_{11} = \dfrac{\mathrm{Det}(R)}{b_{33}r_{22}} \quad a_{12} = \dfrac{r_{12}r_{33} - r_{13}r_{32}}{b_{33}r_{22}} \quad a_{13} = \dfrac{r_{13}r_{22} - r_{12}r_{23}}{b_{33}r_{22}} \end{cases} \qquad (18)$$

$$\begin{cases} R = \begin{pmatrix} 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \\ c_{31} = r_{31} \quad c_{32} = r_{32} \quad c_{33} = r_{33} \\ b_{11} = \dfrac{r_{11}r_{33} - r_{13}r_{31}}{r_{33}} \quad b_{12} = \dfrac{r_{12}r_{33} - r_{13}r_{32}}{r_{33}} \quad b_{13} = \dfrac{r_{13}}{r_{33}} \\ a_{21} = \dfrac{r_{21}r_{33} - r_{23}r_{31}}{b_{11}r_{33}} \quad a_{22} = \dfrac{\mathrm{Det}(R)}{b_{11}r_{33}} \quad a_{23} = \dfrac{r_{11}r_{23} - r_{13}r_{21}}{b_{11}r_{33}} \end{cases} \qquad (19)$$

$$\begin{cases} R = \begin{pmatrix} 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b_{31} & b_{32} & b_{33} \end{pmatrix}\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ c_{11} = r_{11} \quad c_{12} = r_{12} \quad c_{13} = r_{13} \\ b_{31} = \dfrac{r_{31}}{r_{11}} \quad b_{32} = \dfrac{r_{11}r_{32} - r_{12}r_{31}}{r_{11}} \quad b_{33} = \dfrac{r_{11}r_{33} - r_{13}r_{31}}{r_{11}} \\ a_{21} = \dfrac{r_{21}r_{33} - r_{23}r_{31}}{b_{33}r_{11}} \quad a_{22} = \dfrac{\mathrm{Det}(R)}{b_{33}r_{11}} \quad a_{23} = \dfrac{r_{11}r_{23} - r_{13}r_{21}}{b_{33}r_{11}} \end{cases} \qquad (20)$$

$$\begin{cases} R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ c_{21} & c_{22} & c_{23} \\ 0 & 0 & 1 \end{pmatrix} \\ c_{21} = r_{21} \quad c_{22} = r_{22} \quad c_{23} = r_{23} \\ b_{11} = \dfrac{r_{11}r_{22} - r_{12}r_{21}}{r_{22}} \quad b_{12} = \dfrac{r_{12}}{r_{22}} \quad b_{13} = \dfrac{r_{13}r_{22} - r_{12}r_{23}}{r_{22}} \\ a_{31} = \dfrac{r_{22}r_{31} - r_{21}r_{32}}{b_{11}r_{22}} \quad a_{32} = \dfrac{r_{11}r_{32} - r_{12}r_{31}}{b_{11}r_{22}} \quad a_{33} = \dfrac{\mathrm{Det}(R)}{b_{11}r_{22}} \end{cases} \qquad (21)$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}1&0&0\\0&1&0\\a_{31}&a_{32}&a_{33}\end{pmatrix}\begin{pmatrix}1&0&0\\b_{21}&b_{22}&b_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}c_{11}&c_{12}&c_{13}\\0&1&0\\0&0&1\end{pmatrix}\\
c_{11}=r_{11}\quad c_{12}=r_{12}\quad c_{13}=r_{13}\\
b_{21}=\dfrac{r_{21}}{r_{11}}\quad b_{22}=\dfrac{r_{11}r_{22}-r_{12}r_{21}}{r_{11}}\quad b_{23}=\dfrac{r_{11}r_{23}-r_{13}r_{21}}{r_{11}}\\
a_{31}=\dfrac{r_{22}r_{31}-r_{21}r_{32}}{b_{22}r_{11}}\quad a_{32}=\dfrac{r_{11}r_{32}-r_{12}r_{31}}{b_{22}r_{11}}\quad a_{33}=\dfrac{\mathrm{Det}(R)}{b_{22}r_{11}}
\end{array}\right. \tag{22}$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}\lambda_1&a_{12}&a_{13}\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\b_{21}&\lambda_2&b_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\c_{31}&c_{32}&\lambda_3\end{pmatrix}\begin{pmatrix}\lambda_4&d_{12}&d_{13}\\0&1&0\\0&0&1\end{pmatrix}\\
\lambda_1\lambda_2\lambda_3\lambda_4=\mathrm{Det}(R)\\
b_{21}=\dfrac{r_{21}r_{33}-r_{23}r_{31}}{\lambda_3\lambda_4}\quad b_{23}=\dfrac{r_{21}-b_{21}\lambda_4}{r_{31}}\\
c_{31}=\dfrac{r_{31}}{\lambda_4}\quad c_{32}=\dfrac{r_{31}\lambda_2+r_{21}r_{32}-r_{22}r_{31}}{b_{21}\lambda_4}\\
d_{12}=\dfrac{r_{22}-\lambda_2-b_{23}r_{32}}{b_{21}}\quad d_{13}=\dfrac{(r_{33}-\lambda_3)\lambda_4}{r_{31}}\\
a_{12}=\dfrac{r_{11}r_{33}-r_{13}r_{31}-\lambda_1\lambda_3\lambda_4}{b_{21}\lambda_3\lambda_4}\quad a_{13}=\dfrac{r_{11}-\lambda_1\lambda_4-a_{12}r_{21}}{r_{31}}
\end{array}\right. \tag{23}$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}\lambda_1&a_{12}&a_{13}\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\b_{31}&b_{32}&\lambda_2\end{pmatrix}\begin{pmatrix}1&0&0\\c_{21}&\lambda_3&c_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}\lambda_4&d_{12}&d_{13}\\0&1&0\\0&0&1\end{pmatrix}\\
\lambda_1\lambda_2\lambda_3\lambda_4=\mathrm{Det}(R)\\
b_{31}=\dfrac{r_{31}r_{22}-r_{32}r_{21}}{\lambda_3\lambda_4}\quad b_{32}=\dfrac{r_{31}-b_{31}\lambda_4}{r_{21}}\\
c_{21}=\dfrac{r_{21}}{\lambda_4}\quad c_{23}=\dfrac{r_{21}\lambda_2+r_{23}r_{31}-r_{21}r_{33}}{b_{31}\lambda_4}\\
d_{12}=\dfrac{(r_{22}-\lambda_3)\lambda_4}{r_{21}}\quad d_{13}=\dfrac{r_{33}-\lambda_2-b_{32}r_{23}}{b_{31}}\\
a_{12}=\dfrac{r_{11}-\lambda_1\lambda_4-a_{13}r_{31}}{r_{21}}\quad a_{13}=\dfrac{r_{11}r_{22}-r_{12}r_{21}-\lambda_1\lambda_3\lambda_4}{b_{31}\lambda_3\lambda_4}
\end{array}\right. \tag{24}$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}1&0&0\\a_{21}&\lambda_1&a_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}\lambda_2&b_{12}&b_{13}\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\c_{31}&c_{32}&\lambda_3\end{pmatrix}\begin{pmatrix}1&0&0\\d_{21}&\lambda_4&d_{23}\\0&0&1\end{pmatrix}\\
\lambda_1\lambda_2\lambda_3\lambda_4=\mathrm{Det}(R)\\
b_{12}=\dfrac{r_{12}r_{33}-r_{13}r_{32}}{\lambda_3\lambda_4}\quad b_{13}=\dfrac{r_{12}-b_{12}\lambda_4}{r_{32}}\\
c_{31}=\dfrac{r_{32}\lambda_2+r_{12}r_{31}-r_{11}r_{32}}{b_{12}\lambda_4}\quad c_{32}=\dfrac{r_{32}}{\lambda_4}\\
d_{21}=\dfrac{r_{11}-\lambda_2-b_{13}r_{31}}{b_{12}}\quad d_{23}=\dfrac{(r_{33}-\lambda_3)\lambda_4}{r_{32}}\\
a_{21}=\dfrac{r_{22}r_{33}-r_{23}r_{32}-\lambda_1\lambda_3\lambda_4}{b_{12}\lambda_3\lambda_4}\quad a_{23}=\dfrac{r_{22}-\lambda_1\lambda_4-a_{21}r_{12}}{r_{32}}
\end{array}\right. \tag{25}$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}1&0&0\\a_{21}&\lambda_1&a_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\b_{31}&b_{32}&\lambda_2\end{pmatrix}\begin{pmatrix}\lambda_3&c_{12}&c_{13}\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\d_{21}&\lambda_4&d_{23}\\0&0&1\end{pmatrix}\\
\lambda_1\lambda_2\lambda_3\lambda_4=\mathrm{Det}(R)\\
b_{31}=\dfrac{r_{32}-b_{32}\lambda_4}{r_{12}}\quad b_{32}=\dfrac{r_{11}r_{32}-r_{12}r_{31}}{\lambda_3\lambda_4}\\
c_{12}=\dfrac{r_{12}}{\lambda_4}\quad c_{13}=\dfrac{r_{12}\lambda_2+r_{13}r_{32}-r_{12}r_{33}}{b_{32}\lambda_4}\\
d_{21}=\dfrac{(r_{11}-\lambda_3)\lambda_4}{r_{12}}\quad d_{23}=\dfrac{r_{33}-\lambda_2-b_{31}r_{13}}{b_{32}}\\
a_{21}=\dfrac{r_{22}-\lambda_1\lambda_4-a_{23}r_{32}}{r_{12}}\quad a_{23}=\dfrac{r_{11}r_{22}-r_{12}r_{21}-\lambda_1\lambda_3\lambda_4}{b_{32}\lambda_3\lambda_4}
\end{array}\right. \tag{26}$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}1&0&0\\0&1&0\\a_{31}&a_{32}&\lambda_1\end{pmatrix}\begin{pmatrix}\lambda_2&b_{12}&b_{13}\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\c_{21}&\lambda_3&c_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\d_{31}&d_{32}&\lambda_4\end{pmatrix}\\
\lambda_1\lambda_2\lambda_3\lambda_4=\mathrm{Det}(R)\\
b_{12}=\dfrac{r_{13}-b_{13}\lambda_4}{r_{23}}\quad b_{13}=\dfrac{r_{13}r_{22}-r_{12}r_{23}}{\lambda_3\lambda_4}\\
c_{21}=\dfrac{r_{23}\lambda_2+r_{13}r_{21}-r_{11}r_{23}}{b_{13}\lambda_4}\quad c_{23}=\dfrac{r_{23}}{\lambda_4}\\
d_{31}=\dfrac{r_{11}-\lambda_2-b_{12}r_{21}}{b_{13}}\quad d_{32}=\dfrac{(r_{22}-\lambda_3)\lambda_4}{r_{23}}\\
a_{31}=\dfrac{r_{22}r_{33}-r_{23}r_{32}-\lambda_1\lambda_3\lambda_4}{b_{13}\lambda_3\lambda_4}\quad a_{32}=\dfrac{r_{33}-\lambda_1\lambda_4-a_{31}r_{13}}{r_{23}}
\end{array}\right. \tag{27}$$

$$\left\{\begin{array}{l}
R=\begin{pmatrix}1&0&0\\0&1&0\\a_{31}&a_{32}&\lambda_1\end{pmatrix}\begin{pmatrix}1&0&0\\b_{21}&\lambda_2&b_{23}\\0&0&1\end{pmatrix}\begin{pmatrix}\lambda_3&c_{12}&c_{13}\\0&1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\d_{31}&d_{32}&\lambda_4\end{pmatrix}\\
\lambda_1\lambda_2\lambda_3\lambda_4=\mathrm{Det}(R)\\
b_{21}=\dfrac{r_{23}-b_{23}\lambda_4}{r_{13}}\quad b_{23}=\dfrac{r_{11}r_{23}-r_{13}r_{21}}{\lambda_3\lambda_4}\\
c_{12}=\dfrac{r_{13}\lambda_2+r_{12}r_{23}-r_{13}r_{22}}{b_{23}\lambda_4}\quad c_{13}=\dfrac{r_{13}}{\lambda_4}\\
d_{31}=\dfrac{(r_{11}-\lambda_3)\lambda_4}{r_{13}}\quad d_{32}=\dfrac{r_{22}-\lambda_2-b_{21}r_{12}}{b_{23}}\\
a_{31}=\dfrac{r_{33}-\lambda_1\lambda_4-a_{32}r_{23}}{r_{13}}\quad a_{32}=\dfrac{r_{11}r_{33}-r_{13}r_{31}-\lambda_1\lambda_3\lambda_4}{b_{23}\lambda_3\lambda_4}
\end{array}\right. \tag{28}$$

Bibliography

[1] A.W. Paeth, "A Fast Algorithm for General Raster Rotation," in Proc. *Graphics Interface'86—Vision Interface'86*, 1986, pp. 77–81.

[2] D. Fraser, R.A. Schowengerdt and I. Briggs, "Rectification of Multichannel Images in Mass Storage Using Image Transposition," *CVGIP*, vol. 29, pp. 23–36, 1985.

[3] N. Tsuchida, Y. Yamada and M. Ueda, "Hardware for Image Rotation by Twice Skew Transformations," *IEEE Transactions on Acoustics, Speech, and Image Processing*, vol. ASSP-35, no. 4, pp. 527–532, April 1987.

[4] M. Unser, M.A. Neimark and C. Lee, "Affine Transformations of Images: A Least Squares Formulation," in Proc. *First IEEE International Conference on Image Processing*, Austin, Texas, U.S.A., November 13-16, 1994, vol. 3, of 3, pp. 558–561.

[5] M. Unser, P. Thévenaz and L. Yaroslavsky, "Convolution-Based Interpolation for Fast, High-Quality Rotation of Images," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1371–1381, October 1995.

[6] M. Unser, A. Aldroubi and M. Eden, "Enlargement or reduction of Digital Images with Minimum Loss of Information," *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 247–258, March 1995.

[7] E. Pang and D. Hatzinakos, "An Efficient Implementation of Affine Transformation Using One-Dimensional FFT's," in Proc. *International Conference on Acoustics, Speech and Signal Processing*, Münich, Germany, April 21–24, 1997, vol. 4, of 5, pp. 2885–2888.

[8] D. Fraser, "Comparison at High Spatial Frequencies of Two-Pass and One-Pass Geometric Transformation Algorithms," *CGVIP*, vol. 46, pp. 267–283, 1989.

[9] M. Unser, A. Aldroubi and M. Eden, "B-Spline Signal Processing: Part II—Efficient Design and Applications," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 834–848, February 1993.

[10] M. Unser and I. Daubechies, "On the Approximation Power of Convolution-Based Least Squares Versus Interpolation," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1697–1711, July 1997.