Image Processing Algorithms & Techniques

PROCEEDINGS
SPIE—The International Society for Optical Engineering

Image Processing Algorithms and Techniques

Keith S. Pennington
Robert J. Moorhead II
Chairs/Editors

12–14 February 1990
Santa Clara, California

Sponsored by
SPIE—The International Society for Optical Engineering
SPSE—The Society for Imaging Science and Technology

Cooperating Organization
TAGA—Technical Association of the Graphic Arts

SPIE PROCEEDINGS SERIES

Published by
SPIE—The International Society for Optical Engineering
P.O. Box 10, Bellingham, Washington 98227-0010 USA

Volume 1244

SPIE (The Society of Photo-Optical Instrumentation Engineers) is a nonprofit society dedicated to advancing engineering and scientific applications of optical, electro-optical, and optoelectronic instrumentation, systems, and technology.

Please use the following format to cite material from this book:
Author(s), "Title of Paper," *Image Processing Algorithms and Techniques,* Keith S. Pennington, Robert J. Moorhead II, Editors, Proc. SPIE 1244, page numbers (1990).

Library of Congress Catalog Card No. 90-52670
ISBN 0-8194-0291-5

Published by
SPIE—The International Society for Optical Engineering
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone 206/676-3290 (Pacific Time) • FAX 206/647-1445

Printed in the United States of America.

IMAGE PROCESSING ALGORITHMS AND TECHNIQUES

Volume 1244

CONFERENCE COMMITTEE

Conference Chairs
Keith S. Pennington, IBM/Thomas J. Watson Research Center
Robert J. Moorhead II, Mississippi State University

Session Chairs
Session 1—Image Coding and Processing
V. Ralph Algazi, University of California/Davis

Session 2—Image Processing/Analysis Algorithms
Sanjit K. Mitra, University of California/Santa Barbara

Session 3—Image Compression I
Majid Rabbani, Eastman Kodak Company

Session 4—Image Compression II
Ronald B. Arps, IBM/Almaden Research Center

Session 5—Color Image Processing
Robert J. Moorhead II, Mississippi State University

Session 6—Video Image Processing/Coding
Didier J. Le Gall, Bell Communications Research

Session 7—Image Processing/Detection Algorithms
Keith S. Pennington, IBM/Thomas J. Watson Research Center

Conference 1244, *Image Processing Algorithms and Techniques,* was part of a four-conference program on Image Processing held at the 1990 SPIE/SPSE Symposium on Electronic Imaging Science and Technology, 11–16 February 1990, in Santa Clara, California. The other conferences were:

Conference 1245, *Biomedical Image Processing*
Conference 1246, *Parallel Architectures for Image Processing*
Conference 1247, *Nonlinear Image Processing.*

Program Chairs: **Jorge L. C. Sanz,** IBM/Almaden Research Ctr.
Alan C. Bovik, Univ. of Texas/Austin.

INTRODUCTION

The Image Processing Algorithms and Techniques conference was one of four conferences in the Image Processing program at the 1990 SPIE/SPSE Symposium on Electronic Imaging Science and Technology. The intent of the conference was to update the image processing community on recent developments and novel ideas in the areas of image compression, communication, enhancement, and analysis. In addition, there were several papers on image processing environments and color spaces. The image content ranged from spectroscopic to celestial, from freeze-frame to HDTV, and from binary to true-color photographic quality. The conference was divided into seven sessions spanning three full days and had a definite international flavor, with some of the better papers being presented by authors from Japan and Europe.

The first session, Image Coding and Processing, began with an excellent presentation on a novel DCT algorithm that has many improvements over previous algorithms, such as shorter circuit path, fewer operations, and better roundoff properties. Other topics addressed in this session included various integer convolution operators.

The second session, Image Processing/Analysis Algorithms, concentrated on analysis algorithms, image processing environments and interfaces, as well as some assorted issues such as warping and spread filters.

In the third session, Image Compression I, several new approaches to intraframe coding, as well as several enhancements to existing techniques, were presented. The techniques included wavelets, pyramid coding, Lempel-Ziv-Welch, DPCM, and VQ.

The fourth session, Image Compression II, continued on the theme of the previous session. The first of several presentations on the standardization activities currently being pursued within the image processing arena opened this session. The session also included a presentation on digital still cameras and an excellent presentation on subband coding.

In the fifth session, Color Image Processing, two papers were presented. One addressed the applicability of a vector space approach and the other addressed the issues of color transformation while proposing an algorithm to solve many of the problems.

In the sixth session, Video Image Processing/Coding, eleven papers were presented. The subjects varied from algorithms that have been implemented to proposals for algorithms that should become part of a standard. A thorough overview of the capabilities and possibilities of video image coding was obtained.

(continued)

In the final session, Image Processing/Detection Algorithms, the predominant issue of discussion was edge detection.

The authors and session chairs are to be complimented for the excellent program and presentations. We graciously acknowledge and appreciate their efforts and expertise.

Keith S. Pennington
IBM/Thomas J. Watson Research Center

Robert J. Moorhead II
Mississippi State University

IMAGE PROCESSING ALGORITHMS AND TECHNIQUES

Volume 1244

CONTENTS

(continued)

(continued)

SESSION 1

Image Coding and Processing

Chair

V. Ralph Algazi
University of California/Davis

A fast scaled-DCT algorithm

Ephraim Feig
IBM Research Division, Thomas J. Watson Research Center
P. O. Box 218, Yorktown Heights, New York 10598

ABSTRACT

The Discrete Cosine Transform (DCT) followed by scaling and quantization is an important operation in image processing. Because of the scaling, the DCT itself need not be computed, but rather a scalar multiple of the DCT might do, with appropriate compensation incorporated into the scaling. We present a fast method for computing such scaled output of the 2-dimensional DCT on 8×8 points. We also present a similar algorithm for the inverse scaled DCT. [1]

1. INTRODUCTION

The discrete cosine transform (DCT) plays an important role in digital image processing. Of particular interest is the two-dimensional DCT followed by scaling and quantization. This has applications in data compression of continuous tone images [1, 11]. Because the DCT is so often used, research into fast algorithms for its implementation has been rather active [3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15].

Given an array $y(k)$, $0 \leq k \leq K-1$, of input data, its one-dimensional DCT output is

$$\hat{y}(n) = c(n) \sum_{k=0}^{K-1} cos\left(\frac{\pi n(2k+1)}{2K}\right) y(k),$$

where $c(0) = 1/\sqrt{K}$ and $c(n) = \sqrt{2/K}$ for $1 \leq n \leq K-1$. Let C denote the $K \times K$ matrix such that for $0 \leq k, n \leq K-1$, its $(n+1, k+1)$ entry is $c(n)\, cos\,(\pi n(2k+1)/2K)$, and let $\bar{y}$ denote the K-dimensional column vector whose $k+1$-st entry is $y(k)$, for $0 \leq k \leq K-1$. The output of the one-dimensional DCT on $\bar{y}$, written in vector form, equals the matrix-vector product $C\bar{y}$.

Given a two-dimensional $K \times K$ array $x(k_1, k_2)$, $0 \leq k_1, k_2 \leq K-1$, of input data, the 2-D DCT output is

$$\hat{x}(n_1, n_2) = c(n_1)\, c(n_2) \sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} cos\left(\frac{\pi n_1(2k_1+1)}{2K}\right) cos\left(\frac{\pi n_2(2k_2+1)}{2K}\right) x(k_1, k_2).$$

[1]Sections 1 and 2 of this paper were originally published internally as Confidential IBM RC 14713 (#65954), 6/22/89. The algorithms described in this paper are included in a US patent application, Docket YO989-073, 9/6/1989. These results will be presented at the 1990 SPIE/SPSE Symposium of Electronic Imaging Science and Technology, February 12, 1990, Santa Clara, California, and some version of this paper will appear in the proceedings.

Denote by $C \otimes C$ the tensor (Kroenecker) product of the matrix C with itself. Let $\bar{x}$ denote the K^2-dimensional column vector such that for $0 \leq k_1, k_2 \leq K-1$, its $k_2K + k_1$ entry is $x(k_1, k_2)$. We say that $\bar{x}$ is the column-major order description of the input. The output of the two-dimensional DCT, written in column-major order, equals the matrix-vector product $(C \otimes C)\bar{x}$.

Let $\omega = e^{\pi i/2K}$, and write the Discrete Fourier Transform (DFT) as

$$\hat{x}_n = \sum_{k=0}^{2K-1} x_k \, \omega^{2kn} .$$

If $x_k = x_{2K-k-1}$, then

$$\begin{aligned} \frac{1}{2} \hat{x}_n \, \omega^n &= \frac{1}{2} \sum_{k=0}^{2K-1} x_k \, \omega^{(2k+1)n} \\ &= \frac{1}{2} \sum_{k=0}^{K-1} x_k \left(\omega^{(2k+1)n} + \omega^{-(2k+1)n} \right) \\ &= \sum_{k=0}^{K-1} x_k \cos \left(\frac{\pi n (2k+1)}{2K} \right) . \end{aligned}$$

If also x_k are real, then so are $\hat{x}_n \omega^n$.

Write $\hat{x}_n = a_n + ib_n$, with a, b real. Then

$$\begin{aligned} \hat{x}_n \omega^n &= (a_n + ib_n) \left(\cos \frac{\pi n}{2K} + i \sin \frac{\pi n}{2K} \right) \\ &= \left(a_n \cos \frac{\pi n}{2K} - b_n \sin \frac{\pi n}{2K} \right) + i \left(a_n \sin \frac{\pi n}{2K} + b_n \cos \frac{\pi n}{2K} \right) \end{aligned}$$

For x_n real, the imaginary part above is 0, so that

$$b_n = -a_n \sin \frac{\pi n}{2K} \sec \frac{\pi n}{2K} .$$

Substituting into the real part of the left hand side of the previous equation yields

$$\hat{x}_n \omega^n = a_n \sec \frac{\pi n}{2K} = Re \, \hat{x}_n \sec \frac{\pi n}{2K} ,$$

where $Re \, \hat{x}_n$ denotes the real part of $\hat{x}_n$. Thus, to compute the 1-D DCT on a real vector, one can compute the real parts of the first half of the outputs of a DFT on an appropriate double-length vector, and then multiply them by fixed scaling factors. This observation was already made by [13, 10].

Obtaining the real part of the DFT is a linear operation on real vectors. Therefore, if we let D be the $K \times K$ diagonal matrix whose (n, n) entry is $c(n) \sec((n-1)\pi/2K)$, then the one dimensional DCT matrix can be factored as

$$C = DF , \tag{1}$$

where F is the matrix of the linear transformation acting on the real input vector to yield the real parts of the first K terms of the DFT of the doubled, folded data. The two-dimensional DCT matrix is therefore

$$C \otimes C = (DF \otimes DF) = (D \otimes D)(F \otimes F). \tag{2}$$

The matrix $D \otimes D$ is diagonal. Therefore, when one follows the DCT by scaling, one may compute instead the product by the matrix $F \otimes F$ and absorb the diagonal factor in the scaling.

2. THE 8 × 8 SCALED-DCT

It turns out that for the case $K = 8$, which is of primary importance in image processing, the product by $(F \otimes F)$ can be done very efficiently. In [2] it was suggested that an algorithm based on Winograd's 16-point DFT [16] algorithm be used in row-column fashion to compute this product. This led to an algorithm which used 80 multiplications and 464 additions (we will use term additions to mean additions and/or subtractions). Our method is similar in that it also uses the Winograd construction. But we will only apply the pre- and post-additions of the Winograd algorithm in row-column fashion. The multiplication parts, the so called "core" of the DFT computation, will be done in a direct way, which will be obtained by a closer look at the tensor product of the core factor with itself.

Using the ideas introduced by Winograd, we factor F as

$$F = P R_1 M R_2, \tag{3}$$

where $R_1 = B_1 B_2$, $R_2 = A_1 A_2 A_3$, with

$$B_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix},$$

$$B_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix},$$

$$
M = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \sqrt{2}/2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\cos(\pi/8) & 0 & -\cos(3\pi/8) & 0 \\
0 & 0 & 0 & 0 & 0 & \sqrt{2}/2 & 0 & 0 \\
0 & 0 & 0 & 0 & -\cos(3\pi/8) & 0 & \cos(\pi/8) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix},
$$

$$
A_1 = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix},
$$

$$
A_2 = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix},
$$

$$
A_3 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{pmatrix},
$$

and

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

As with the Winograd factorization of the DFT, our factroization of F suggests an algorithm for multiplying by F which sandwitches its multiplications between subroutines that only use additions, and concludes with a permutation.

We are interested in multiplication by $F \otimes F$. A standard result on tensor products of matrices yields the equation

$$\begin{aligned} F \otimes F &= P R_1 M R_2 \otimes P R_1 M R_2 \\ &= (P \otimes P)(R_1 \otimes R_1)(M \otimes M)(R_2 \otimes R_2). \end{aligned} \tag{4}$$

This equation is the key to our new algorithm. The matrix $P \otimes P$ is a permutation matrix, and the implementation of its product is trivial (the size of the input, 64 data points, is sufficiently small so that data-management is not an issue). Next, observe that evaluating products of arbitrary 8-dimensional vectors by the matrices B_1, B_2, A_1, A_2, A_3 require additions only. Indeed, these products can be computed with 4, 4, 3, 7, 8 additions, respectively. We first evaluate the product with $R_2 \otimes R_2$ using the standard row-column method; that is, we first multiply all the rows of the input matrix by R_2 and then we multiply all the columns of the resulting matrix by R_2. The product by R_2 is done via the factorization $R_2 = A_1 A_2 A_3$ with 18 additions. Since this is done 16 times, computing the product by $R_2 \otimes R_2$ is done with 288 additions. Similarly, the product by $R_1 \otimes R_1$ is done in row-column fashion with 128 additions.

The computation of the product by $M \otimes M$ will not be done in row-column fashion. Rather, the first, second, fourth and eighth columns of $x(k_1, k_2)$ will each be multiplied by M. Each of these will involve 2 multiplications by $\sqrt{2}/2$ plus a product of a 2-vector by the 2×2 block

$$N = \begin{pmatrix} -\cos(\pi/8) & -\cos(3\pi/8) \\ -\cos(3\pi/8) & \cos(\pi/8) \end{pmatrix},$$

which can be done with 3 multiplications and 3 additions. The third and sixth columns of $x(m_1, m_2)$ will be multiplied by $(\sqrt{2}/2)M$. Each of these can be done with 4 multiplications by $(\sqrt{2}/2)$, 2 multiplications by $1/2$, plus a product of a 2-vector by the 2×2 matrix $(\sqrt{2}/2)N$, which can be done with 3 multiplications and 3 additions. The fifth and seventh columns will be handled simultaneously to account for the product by $N \otimes M$. A 16-dimensional column vector $\bar{v}$ is formed by interleaving the entries of these two columns. The first, second, fourth and eighth pairs of entries are each multiplied by N, while the third and sixth pairs are multiplied by $(\sqrt{2}/2)\,N$. Each of these takes 3 multiplications and three additions. Finally, the fifth and seventh pairs of entries are handled simultaneously as a 4-vector multiplied by $N \otimes N$. One observes that

$$N \otimes N = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \sqrt{2}/4 & \sqrt{2}/4 & 0 & 0 \\ \sqrt{2}/4 & -\sqrt{2}/4 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & -1/2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix}.$$

Using this factorization, the product by $N \otimes N$ can be done with 2 multiplications by $\sqrt{2}/4$ and 10 addition. Altogether, the entire algorithm calls for 54 multiplications, 462 additions, and 6 multiplications by 1/2.

3. THE 8 × 8 INVERSE SCALED-DCT

The matrix C is orthogonal: its inverse is its transpose. Therefore from equation 1 we have

$$C^{-1} = F^t D \tag{5}$$

where the superscript ^{t} denotes the transpose. Similarly, analogous to equation 2 we have

$$\begin{aligned} (C \otimes C)^{-1} &= C^{-1} \otimes C^{-1} = F^t D \otimes F^t D \\ &= \left(F^t \otimes F^t\right)(D \otimes D)\,. \end{aligned} \tag{6}$$

Multiplication by the diagonal matrix $(D \otimes D)$ can be incorporated into the descaling, and so we consider only the remaining computation of the product by $(F^t \otimes F^t)$. From equation 3, observing that M is symmetric, we have

$$F^t = R_2^t\, M\, R_1^t\, P^t\,, \tag{7}$$

and using the notation of the previous section, $R_1^t = B_2^t B_1^t$ and $R_2^t = A_3^t A_2^t A_1^t$. Therefore, analogous to equation 4 we have

$$\begin{aligned} F^t \otimes F^t &= R_2^t\, M\, R_1^t\, P^t \otimes R_2^t\, M\, R_1^t\, P^t \\ &= (R_2^t \otimes R_2^t)(M \otimes M)(R_1^t \otimes R_1^t)(P^t \otimes P^t)\,. \end{aligned} \tag{8}$$

This equation yields an algorithm for the inverse scaled-DCT computation. The product by the permutation matrix $(P^t \otimes P^t)$ is trivial. The products by $(R_1^t \otimes R_1^t)$ and $(R_2^t \otimes R_2^t)$ can be done in row-column fashion with 128 and 288 additions, respectively. And the middle "essential" computation, the product by $M \otimes M$ has already been discussed. Hence the inverse algorithm for the scaled-DCT can also be done with 54 multiplications and 462 additions plus 6 multiplications by 1/2.

An alternate way of computing the inverse DCT is via the more direct consequence of equation 1, namely

$$C^{-1} = F^{-1} D^{-1}, \tag{9}$$

from which we obtain from equation 2

$$(C \otimes C)^{-1} = \left(F^{-1} \otimes F^{-1}\right)\left(D^{-1} \otimes D^{-1}\right). \tag{10}$$

Again, multiplication by the diagonal matrix $(D^{-1} \otimes D^{-1})$ can be incorporated into the descaling. From equation 3, we have

$$F^{-1} = R_2^{-1} M^{-1} R_1^{-1} P^{-1}, \tag{11}$$

and $R_1^{-1} = B_2^{-1} B_1^{-1}$ and $R_2^{-1} = A_3^{-1} A_2^{-1} A_1^{-1} t$. Therefore,

$$F^{-1} \otimes F^{-1} = (R_2^{-1} \otimes R_2^{-1})(M \otimes M)(R_1^{-1} \otimes R_1^{-1})(P^{-1} \otimes P^{-1}). \tag{12}$$

Here too the product by $(P^{-1} \otimes P^{-1})$ is trivial. The products by $(R_1^{-1} \otimes R_1^{-1})$ and $(R_2^{-1} \otimes R_2^{-1})$ involve not only additions but also divisions by 2. As for the product by $(M^{-1} \otimes M^{-1})$, after we observe that $N^{-1} = -N$, an algorithm similar to the one in section 2 can be constructed using the methods we have outlined so far; we leave this to the reader. Rather, we present an alternative factorization obtained from equation 11 which, after appropriate massaging, incorporates the divisions by 2 of the "addition" steps into the middle "multiplication" factor. We obtain the following factorization:

$$C^{-1} = \tilde{F} \tilde{D}, \tag{13}$$

where $\tilde{D}$ is the 8×8 diagonal matrix whose (1,1) entry is $\sqrt{2}/4$ and whose (n, n) entry, for $1 \le n \le 8$, is $\frac{1}{2} \cos(2\pi(n-1)/16)$, and

$$\tilde{F} = \tilde{R}_2 \tilde{M} \tilde{R}_1 P^t, \tag{14}$$

with $\tilde{R}_1 = \tilde{B}_2 \tilde{B}_1$ and $\tilde{R}_2 = \tilde{A}_3 \tilde{A}_2 \tilde{A}_1$, and

$$\tilde{B}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$