COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 41, 333–345 (1988)

NOTE

Some Algorithms for Approximating Convolutions

DIANNE P. O'LEARY

Mathematical Analysis Division, National Bureau of Standards, Gaithersburg, Maryland 20899
and Computer Science Department and Institute for Advanced Computer Studies,
University of Maryland, College Park, Maryland 20742

Received July 9, 1986; accepted September 3, 1987

This paper presents some algorithms for approximating two-dimensional convolution operators of size $n \times n$, n odd, by a product, or sum of products, of 3×3 convolutions. Inaccuracies resulting from the approximation as well as from fixed point computation are discussed and examples are given. © 1988 Academic Press, Inc.

1. INTRODUCTION

This work involves the approximation of two-dimensional convolutions of size $n \times n$, n odd, by a product (or sum of products) of 3×3 convolutions. The motivation for the work is related to the PIPE, a parallel image processing machine developed in the Center for Manufacturing Engineering of the National Bureau of Standards and described by Kent *et al.* [4]. Standard image processing algorithms require the application of convolutions of size $n \times n$ to an image, but the PIPE hardware can only handle 3×3 convolutions. For a survey of the uses of convolutions in image processing, see Huang [3]. For some background on the algebra and properties of convolutions, see Mitra and Ekstrom [6].

The PIPE is a parallel processing machine consisting of a number of stages. Images flow through the stages in real time, and during each time period, each stage can perform a limited number of pointwise arithmetic or Boolean operations on its three current 256×256 images, or apply a 3×3 convolution operator to an image. The time period is not sufficient to allow 5×5 or larger convolutions. The operations performed at a given stage, and the data paths routing the images, are under program control.

Thus, in defining algorithms for the PIPE and similar real-time processing machines, the standard measure of computational complexity, the number of arithmetic operations, is not applicable. The relevant measure is simply the number of stages of the PIPE machine needed to perform a given operation.

In Section 2 we summarize some of the properties of convolutions and derive exact representations for some classes of convolutions. In Section 3 we apply these methods and others to the problem of approximating a general convolution. The effects of inexact arithmetic are discussed in Section 4. Section 5 provides some examples.

Many of the ideas in this work derive from conversations with Chris Witzgall. Comments of a referee were very helpful.

0734-189X/88 $3.00

2. EXACT FACTORIZATIONS OF CONVOLUTIONS

2.1. *Properties of Convolutions*

We begin this section with some examples establishing notation and illustrating the basic properties of convolutions and their factorizations. Convolutions are related to polynomials in two variables, and to block-banded Toeplitz matrices of infinite dimension. We exploit the first relationship, but not the second. For simplicity, we restrict our examples to 5×5, but the principles are general.

EXAMPLE 1. Convolutions obey the algebra of polynomial multiplication. The coefficients in the product

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} * \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \tag{1}$$

are defined by the relationship

$$\sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij}x^{5-i}y^{5-j} = \left(\sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}x^{3-i}y^{3-j}\right)\left(\sum_{i=1}^{3}\sum_{j=1}^{3} b_{ij}x^{3-i}y^{3-j}\right).$$

From this, we conclude that multiplication of convolutions is commutative.

In general, an $n \times n$ convolution corresponds to a polynomial in two variables with highest order term $x^{n-1}y^{n-1}$, and if a factorization into 3×3 convolutions exists, the factorization contains $\alpha = (n-3)/2 + 1$ factors. There are n^2 conditions on the parameters in the factors in order to make the product of α terms equal to a given convolution, and there are 9α parameters.

EXAMPLE 2. If a convolution has a factorization as a product of 3×3 convolutions, then that factorization is nonunique:

$$\begin{bmatrix} 1 & 0 & -5 & 0 & 4 \\ 2 & 0 & -10 & 0 & 8 \\ 3 & 0 & -15 & 0 & 12 \\ 4 & 0 & -20 & 0 & 16 \\ 2 & 0 & -10 & 0 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -4 \\ 0 & 0 & 0 \\ 2 & 0 & -8 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & -2 \\ 4 & 0 & -4 \\ 2 & 0 & -2 \end{bmatrix} * \begin{bmatrix} .5 & 0 & -2 \\ 0 & 0 & 0 \\ 1 & 0 & -4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -3 & 2 \\ 2 & -6 & 4 \\ 1 & -3 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 2 \\ 0 & 0 & 0 \\ 2 & 6 & 4 \end{bmatrix}.$$

The second factorization is formed by multiplying the convolutions in the first factorization by scale factors whose product is 1. The third factorization cannot be derived in a simple way from the first two.

Note that under standard measures of computational complexity, the first two factorizations would be preferred to the third, since the number of arithmetic operations is proportional to the total number of nonzeroes in the convolutions: 10 in the first two, vs. 15 in the third and in the original 5×5 convolution. All three factorizations require two terms, however, and therefore run in equal time on a machine like the PIPE.

EXAMPLE 3. Convolutions can also be decomposed, either by diagonals or by subarrays, as a sum of products of 3×3 convolutions:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 3 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The convolutions with a single 1 are just shift operators, and many computer architectures (although not the PIPE) can take advantage of this.

EXAMPLE 4. Convolutions can also be decomposed as a sum of nontrivial convolutions of decreasing size:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & -24 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$* \begin{bmatrix} 0 & -2 & 0 \\ -2 & -25 & -2 \\ 0 & -2 & 0 \end{bmatrix},$$

where the first term in the decomposition also has an exact representation as a product of 3×3 convolutions.

EXAMPLE 5. Not every convolution has an exact factorization as a product of 3×3 convolutions. Here is an example of one which does not:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c_{22} & c_{23} & c_{24} & X \\ 0 & c_{32} & c_{33} & c_{34} & c_{35} \\ 0 & c_{42} & c_{43} & c_{44} & c_{45} \\ 0 & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix}.$$

The c_{ij} represent arbitrary values and the X represents any nonzero value. Then, equating terms on the left- and right-hand sides in Eq. (1), we obtain the conditions $a_{12} = a_{13} = a_{21} = a_{31} = b_{12} = b_{13} = b_{21} = b_{31} = 0$. The condition on X requires that $a_{13}b_{23} + a_{23}b_{13} = X$, which is a contradiction. Thus the 25 equality conditions on the 18 parameters of the 3×3 factors are inconsistent, and no set of parameters exists.

The class of convolutions which have an exact factorization are those for which the n^2 conditions of Eq. (1) yield a solution. (For a detailed mathematical discussion of these conditions, see Chakrabarti, Bose, and Mitra [1].) These conditions are not very revealing, however, and it is useful to have classes of matrices which can be determined by inspection to have exact factorizations. We now study two such classes: the convolutions which, considered as a matrix, have rank 1, and convolutions whose nonzeros lie on a single diagonal or anti-diagonal.

2.2. *Convolutions Which are Matrices of Rank 1*

The set of convolutions which are matrices of rank 1 have exact factorizations. We illustrate this for 5×5 convolutions. Since each row of such a matrix is a multiple of the first one, every rank-1 matrix has a representation as uv^T, where u and v are column vectors. In polynomial representation, this corresponds to

$$p(x, y) = \sum_{j=1}^{5} u_j y^{5-j} \sum_{i=1}^{5} v_i x^{5-i}.$$

Since the u_j are real numbers, the fourth degree polynomial in y has a factorization as the product of two quadratic polynomials with real coefficients, and similarly for the polynomial in x. We thus have a decomposition of $p(x, y)$ as

$$\begin{aligned} p(x, y) = {} & (a_1y^2 + b_1y + c_1)(a_2y^2 + b_2y + c_2)(d_1x^2 + e_1x + f_1) \\ & \times (d_2x^2 + e_2x + f_2), \end{aligned}$$

where all of the coefficients are real. (If enough of the roots of the polynomial in x or the polynomial in y are real, then this decomposition can be done in a different way by pairing the roots in a different way.) We now have a factorization of the original polynomial, or, equivalently, a factorization of the convolution, as

$$\begin{aligned} uv^{\mathrm{T}} &= \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} [d_1\ e_1\ f_1] * \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} [d_2\ e_2\ f_2] \\ &= \begin{bmatrix} a_1d_1 & a_1e_1 & a_1f_1 \\ b_1d_1 & b_1e_1 & b_1f_1 \\ c_1d_1 & c_1e_1 & c_1f_1 \end{bmatrix} * \begin{bmatrix} a_2d_2 & a_2e_2 & a_2f_2 \\ b_2d_2 & b_2e_2 & b_2f_2 \\ c_2d_2 & c_2e_2 & c_2f_2 \end{bmatrix}. \end{aligned}$$

Thus, any convolution which is a matrix of rank 1 has an exact factorization of this form, unique only up to scale factors for each factor, and there may be other

factorizations corresponding to different pairings, as in Example 2. In that case we had

$$(y^4 + 2y^3 + 3y^2 + 4y + 2)(x^4 - 5x^2 + 4)$$
$$= (y + 1)(y + 1)(y^2 + 2)(x + 1)(x - 1)(x + 2)(x - 2),$$

and two alternative pairings of the x factors with the y factors give the first and third factorizations.

To perform this factorization on a computer, we find the roots of the polynomials of a single variable, and then pair these roots to obtain quadratic factors with real coefficients. There are many standard algorithms for finding roots of polynomials; one is to use Newton's method on the system of equations obtained from the symmetric functions of the roots.

2.3. *Single-Diagonal Matrices*

The set of convolutions which have only one nonzero diagonal also have exact factorizations. We illustrate this for 5×5 convolutions.

Suppose that the convolution A has only one nonzero diagonal, and we call the coefficients on that diagonal $u_1, u_2, \ldots, u_k$. Then again we can form and factor the polynomial

$$\sum_{i=1}^{k} u_i x^{k-i} = (a_1 x^2 + b_1 x + c_1)(a_2 x^2 + b_2 x + c_2),$$

where some of the leading coefficients are zero if the degree $k - 1$ is less than 4. A product of convolutions with these coefficients and some shift convolutions then reproduces the original. For example,

$$\begin{bmatrix} u_1 & 0 & 0 & 0 & 0 \\ 0 & u_2 & 0 & 0 & 0 \\ 0 & 0 & u_3 & 0 & 0 \\ 0 & 0 & 0 & u_4 & 0 \\ 0 & 0 & 0 & 0 & u_5 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & b_1 & 0 \\ 0 & 0 & c_1 \end{bmatrix} * \begin{bmatrix} a_2 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & c_2 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ u_2 & 0 & 0 & 0 & 0 \\ 0 & u_3 & 0 & 0 & 0 \\ 0 & 0 & u_4 & 0 & 0 \\ 0 & 0 & 0 & u_5 & 0 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & b_1 & 0 \\ 0 & 0 & c_1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ b_2 & 0 & 0 \\ 0 & c_2 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & u_3 & 0 & 0 \\ 0 & 0 & 0 & u_4 & 0 \\ 0 & 0 & 0 & 0 & u_5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & b_1 & 0 \\ 0 & 0 & c_1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Again, the critical computational task is finding roots of polynomials in one variable.

A similar decomposition can be performed for matrices with a single antidiagonal, as illustrated in Example 3.

3. ALGORITHMS FOR EXACT AND APPROXIMATE FACTORIZATIONS

We discuss in this section five methods for decomposing an $n \times n$ covolution as a product (or sum of products) of 3×3 terms.

3.1. *Method 1: Rank-1 Approximations*

An $n \times n$ convolution, considered as a matrix A, can be decomposed as the sum of at most n rank-1 matrices. Each of these rank-1 matrices can be factored exactly. Therefore, one exact representation of a convolution is as a sum of at most n products, each of which represents a rank-1 matrix. Since it may be desirable to represent a convolution by only a few of these terms, it is important to be able to obtain the *best* rank-1 (or rank-2, etc.) approximation for a given matrix. One criterion for "best" is least squares: among all rank-k matrices B, chose the one for which

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - b_{ij})^2$$

is minimized. This problem has been well studied (see, for example, Stewart [10, 1973, p. 322]) and the solution is known to be given by the sum of the first k terms of the "singular value decomposition"

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^{\mathrm{T}}$$

where the scalars σ_i are ordered $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ and u_i and v_i are column vectors satisfying the orthogonality conditions

$$u_i^{\mathrm{T}} u_j = v_i^{\mathrm{T}} v_j = 0, \qquad i \neq j,$$
$$u_i^{\mathrm{T}} u_i = v_i^{\mathrm{T}} v_i = 1,$$

for $i, j = 1, \ldots, n$. This method produces an exact representation of the convolution A if k terms are taken, where k is the number of nonzero singular values, or, equivalently, k is the rank of the matrix A. Treitel and Shanks (1971) have also used low rank approximations to convolutions.

METHOD 1.

1. Compute the singular value decomposition of the matrix A.
2. For each rank-1 term in the decomposition:
   Factor the rank-1 matrix as a product of 3×3 convolutions, and replace A by A minus the rank-1 matrix, leaving the residual stored in A.
   Quit if the residual is small enough.

3.2. *Method 2: Decomposition by Diagonals*

The matrix A can also be decomposed as the sum of its diagonals running from upper left to lower right: for example, a 3×3 matrix can be expressed as the sum of 5 terms:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ a_{31} & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ 0 & a_{32} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & 0 \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

In general, a convolution of dimension $n \times n$ can be decomposed as the sum of $2n - 1$ diagonal convolutions, and each diagonal can then be represented exactly as a product of 3×3 convolutions.

METHODS 2 AND 3.

1. Consider each diagonal of A as a vector, and order them by decreasing 2-norm.
2. For each diagonal:
   Factor the 1-diagonal matrix as a product of 3×3 convolutions, and replace A by A minus the 1-diagonal matrix, leaving the residual stored in A.
   Quit if the residual is small enough.

3.3. *Method 3: Decomposition by Anti-Diagonals*

The matrix A can also be decomposed as the sum of its diagonals running from upper right to lower left: for example, a 3×3 matrix can be expressed as the sum of 5 terms:

$$A = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & 0 \\ a_{21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & 0 \\ a_{31} & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a_{23} \\ 0 & a_{32} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & a_{33} \end{bmatrix}.$$

Each of the $2n - 1$ 1-diagonal matrices of dimension $n \times n$, can then be represented exactly as a product of 3×3 convolutions, as was done in the first part of Example 3.

3.4. *Method 4: Approximation by Least Squares*

The approximation problem can also be expressed as a nonlinear least squares problem

$$\min_{b_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - b_{ij})^2, \tag{2}$$

where B is a matrix formed from the product of $\alpha = (n-3)/2 + 1$ convolutions of size 3×3. This rather innocent-looking problem is complicated by the fact that the solution is nonunique, as discussed in Section 2. This means that the matrix of second derivatives of the function in (2) will be singular at a solution, and this greatly complicates the numerical methods and makes them much less reliable. If the scale factors were the only source of nonuniqueness, we could easily eliminate the complication by removing some of the degrees of freedom (e.g., setting some of the coefficients in the 3×3 convolutions to constants), but, as we saw above, even this will not guarantee a unique solution.

Similar approximation problems have been studied by Pistor [8] and Maria and Fahmy [5], who approximate $n \times n$ unstable filters by a product of stable ones of smaller dimension.

There are many standard algorithms which can be applied to (2) (see, for example, Dennis, Gay, and Welsch [2]), but most of these methods are only guaranteed to find a local minimum of the function; i.e., perturbing the coefficients in the 3×3 convolutions in the solution will give a higher sum of squared residuals, but there may be a different combination of coefficients which results in a better approximation. To find the global rather than the local minimum is a much more expensive and less-understood task, and reliable algorithms are rare.

METHOD 4. Solve the nonlinear least squares problem (2) to obtain an approximation to A.

3.5. *Method 5: Least Squares Approximation by a Sum of Products*

This last method is similar to Method 4 except that positive weights w_{ij} are included in the objective function:

$$\min_{b_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(a_{ij} - b_{ij})^2. \tag{3}$$

In the first step of the calculation, we choose the weights to be 1 for the interior coefficients of the convolution ($i, j = 2, \ldots, n-1$) and large for the border coefficients. This forces a better approximation to the border coefficients and leaves a residual which is concentrated in the interior, in a convolution of order $n-2$. The process can then be iterated, giving an approximation of A as a sum of convolutions of size $n, n-2, \ldots, 3$, each of which has an exact factorization. This form of decomposition is similar to Example 4. Example 5 tells us that the minimum may be nonzero even if the interior weights are zero.

A related problem has been studied by Pratt, Abramatic, and Faugeras [9], who applied for a patent on the idea of approximating a convolution as a linear

combination of powers of a single small convolution C,

$$A = \sum_{i=1}^{\alpha} b_i C^i,$$

and proposed a hardware configuration implementing this summation. Mutluay and Fahmy [7] do work related to that of Pratt *et al.* [9].

METHOD 5.

For $$\alpha = \frac{n-3}{2} + 1, \frac{n-3}{2}, \ldots, 1$$

Solve the nonlinear least squares problem (3) with weights of 1 for interior coefficients and a large number for border coefficients, obtaining a product with α terms.

Replace A by the convolution of dimension 2 smaller, formed from the residuals on the interior coefficients.

4. THE EFFECTS OF INEXACT ARITHMETIC

The PIPE computer, like many image processing machines, operates in fixed-point binary arithmetic. The images are represented to 8-bits including sign, the convolutions to 12-bits, and computation intermediate to forming the application of a convolution with an image is performed to 20-bits. For convenience, we will consider the binary point to be located in front of the first bit. Overflow causes an erroneous result (wrap-around arithmetic) with no error flag.

These arithmetic characteristics have several implications for the design of the approximation algorithms. We now discuss these implications.

There is a tolerance for the approximation of a convolution beyond which the algorithm need not go. If the convolution has been represented to the fixed-point round-off precision, then there is no sense in further work to reduce the residual. Algorithms for approximating convolutions should contain a tolerance *tol* which states that a residual convolution of less than *tol* times the size of the original convolution should be considered to be zero. One way to measure the size of a convolution C is by

$$\left(\sum_i \sum_j c_{ij}^2 \right)^{1/2}.$$

This tolerance also affects the choice of weights in Method 5. We ask that the border coefficients be approximated $1/tol$ times more accurately than the interior coefficients. This essentially asks that the border residuals be (fixed point) zero relative to the interior ones, in case all of them cannot be made zero.

The fixed point arithmetic also means that the best floating point approximation to a given convolution is not necessarily the best fixed point approximation. There are two parts to this problem: scaling and rounding.

Under floating point, the product of several 3×3 convolutions is invariant if the convolutions are multiplied by various scale factors which are powers of 2 and

whose product is 1, as long as intermediate underflow and overflow are avoided. This is not true in fixed point, and scaling has a tremendous effect on the accuracy of an approximation to a convolution. It can be shown that the best scalings are those which make the 3×3 convolutions of approximately the same magnitude. This is implemented by choosing scale factors which make the largest magnitude components in each convolution equal. Choosing powers of 2 which approximate these scale factors can be better on some examples, because the resulting coefficients are more likely to have exact representations on machines with binary arithmetic.

This brings us to the second problem, that of rounding. In approximating a 5×5 convolution, we have 18 coefficients, and each of these must be rounded up or down to represent them on image processing machines. This gives 2^{18} possible convolution products for binary arithmetic, and it is obviously impractical to test them all. The choice needs to be made by machine-assisted intuition.

5. COMPUTATIONAL EXAMPLES

5.1. *Computational Example* 1

As an illustration of the approximation algorithms, consider the convolution

$$A = \begin{bmatrix} 0 & 0 & 0.125 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.125 & 0 & -0.500 & 0 & 0.125 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.125 & 0 & 0 \end{bmatrix}.$$

This is a Laplacian-type operator, scaled to avoid overflow on machines which represent numbers in the interval $(-1, 1)$.

Method 1, the singular value decomposition, yields the essentially exact representation

$$\begin{aligned} A = & \begin{bmatrix} -0.15974 & 0.25000 & 0.15974 \\ 0.25000 & -0.39127 & -0.25000 \\ 0.15974 & -0.25000 & -0.15974 \end{bmatrix} \\ & * \begin{bmatrix} 0.15974 & 0.25000 & -0.15974 \\ 0.25000 & 0.39127 & -0.25000 \\ -0.15974 & -0.25000 & 0.15974 \end{bmatrix} \\ & + \begin{bmatrix} 0.15974 & -0.19890 & 0.15974 \\ -0.19890 & 0.24767 & -0.19890 \\ 0.15974 & -0.19890 & 0.15974 \end{bmatrix} * \begin{bmatrix} 0.15974 & 0.19890 & 0.15974 \\ 0.19890 & 0.24767 & 0.19890 \\ 0.15974 & 0.19890 & 0.15974 \end{bmatrix}. \end{aligned}$$

The roots of the polynomials for the first rank-1 matrix were all real (2.052, -0.487, -2.052, and 0.487 for each factor), and other pairings of these roots would result in a different but equally accurate representation in floating point arithmetic. The second rank-1 component has complex roots (± 0.6226, ± 0.7825), and the representation is unique up to scale factors. A single term representation of A gives a residual of 0.056, compared to 0.559 for the size of A itself. Applying the single term representation to an image requires approximately 18 multiplications per image point, vs. 5 multiplications per point for the original 5×5 convolution, but requires only two stages on the PIPE.

Method 2 reports that there are 3 nonzero diagonals in A, and leads to the decomposition

$$A = \begin{bmatrix} -0.70711 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.70711 \end{bmatrix} + \begin{bmatrix} 0.35355 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.35355 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.35355 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0.35355 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.35355 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0.35355 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Because of the structured pattern of A, Method 3 produces results similar to Method 2.

Method 4, the least squares calculation, produces the decomposition

$$A = \begin{bmatrix} -0.0067334 & 0 & 0.17251 \\ 0 & 0 & 0 \\ 0.17251 & 0 & -0.66703 \end{bmatrix} * \begin{bmatrix} 0.66703 & 0 & -0.17251 \\ 0 & 0 & 0 \\ -0.17251 & 0 & 0.0067334 \end{bmatrix}$$

which gives a residual of norm 0.046, slightly better than the rank-1 approximation. Rounded to 3 significant digits, we obtain

$$\begin{bmatrix} -0.007 & 0 & 0.173 \\ 0 & 0 & 0 \\ 0.173 & 0 & -0.667 \end{bmatrix} * \begin{bmatrix} 0.667 & 0 & -0.173 \\ 0 & 0 & 0 \\ -0.173 & 0 & -0.007 \end{bmatrix} = \begin{bmatrix} -0.0047 & 0 & 0.1166 & 0 & -0.0299 \\ 0 & 0 & 0 & 0 & 0 \\ 0.1166 & 0 & -0.5047 & 0 & 0.1142 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0299 & 0 & 0.1142 & 0 & 0.0047 \end{bmatrix}$$

for a residual of 0.047.

Method 5, the bordering method, gives the (exact) result

$$A = \begin{bmatrix} 0 & 0 & -0.35238 \\ 0 & -0.49203 & 0 \\ -0.35238 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -0.35473 & 0 & 0 \\ 0 & 0.49203 & 0 \\ 0 & 0 & -0.35473 \end{bmatrix}$$

plus the center coefficients

$$\begin{bmatrix} -0.17454 & 0 & 0.17338 \\ 0 & -0.25791 & 0 \\ 0.17338 & 0 & -0.17454 \end{bmatrix}.$$

Although some of the five representations are clearly better than some of the others, the choice of which one to use will depend on the amount of computational time it is possible to devote to this convolution problem, some machine-dependent con-

straints, and how each of the approximations fits in with the other steps in the image processing problem.

5.2. *Computational Example 2*

As a second example, consider the convolution

$$A = \frac{1}{128}\begin{bmatrix} 127 & 0 & 0 & 0 & -128 \\ 127 & 0 & 0 & 0 & -128 \\ 127 & 0 & 0 & 0 & -128 \\ 127 & 0 & 0 & 0 & -128 \\ 127 & 0 & 0 & 0 & -128 \end{bmatrix},$$

which is an example of an edge detector. Since A is a rank-1 matrix, we obtain an exact factorization

$$A = \frac{1}{128}\begin{bmatrix} 14.335 & 0 & 14.391 \\ -8.8595 & 0 & -8.8943 \\ 14.335 & 0 & 14.391 \end{bmatrix} * \begin{bmatrix} 8.8595 & 0 & -8.8943 \\ 14.335 & 0 & -14.391 \\ 8.8595 & 0 & -8.8943 \end{bmatrix}.$$

Rounding to three significant digits gives the approximation

$$A = \frac{1}{128}\begin{bmatrix} 14.3 & 0 & 14.3 \\ -8.9 & 0 & -8.9 \\ 14.3 & 0 & 14.3 \end{bmatrix} * \begin{bmatrix} 8.9 & 0 & -8.9 \\ 14.3 & 0 & -14.3 \\ 8.9 & 0 & -8.9 \end{bmatrix}$$

$$= \frac{1}{128}\begin{bmatrix} 127.27 & 0 & 0 & 0 & -127.27 \\ 125.28 & 0 & 0 & 0 & -125.28 \\ 127.27 & 0 & 0 & 0 & -127.27 \\ 125.28 & 0 & 0 & 0 & -125.28 \\ 127.27 & 0 & 0 & 0 & -127.27 \end{bmatrix}.$$

Experimenting with nearby convolutions yields

$$A = \frac{1}{128}\begin{bmatrix} 13 & 0 & 13 \\ -8 & 0 & -8 \\ 13 & 0 & 13 \end{bmatrix} * \begin{bmatrix} 8 & 0 & -8 \\ 13 & 0 & -13 \\ 8 & 0 & -8 \end{bmatrix} = \frac{1}{128}\begin{bmatrix} 104 & 0 & 0 & 0 & -104 \\ 105 & 0 & 0 & 0 & -105 \\ 104 & 0 & 0 & 0 & -104 \\ 105 & 0 & 0 & 0 & -105 \\ 104 & 0 & 0 & 0 & -104 \end{bmatrix},$$

which, although normalized differently than A, is a very good edge detector.

6. CONCLUSIONS

We have summarized some theory related to the approximation of $n \times n$ convolutions, n odd, by a product or sum of products of 3×3 convolutions, and have presented five algorithms for computing approximations. The purpose of these algorithms is to assist the analyst in the design and approximation of convolution operators on image processing machines.

REFERENCES

1. S. Chakrabarti, N. K. Bose, and S. K. Mitra, Sum and product separabilities of multivariate functions and applications, *J. Franklin Inst.* **299**, 1975, 53–66; also in [6].
2. J. E. Dennis, D. M. Gay, and R. E. Welsch, NL2SOL—An adaptive nonlinear least-squares algorithm, *ACM Trans. Math. Software* **7**, 1981, 369–383.

3. T. S. Huang, Recent advances in picture processing and digital filtering, in *Picture Processing and Digital Filtering* (T. S. Huang, Ed.), pp. 283–292, Springer-Verlag, New York, 1979.
4. E. W. Kent, M. O. Shneier, and R. Lumia, PIPE (pipelined image processing engine), manuscript, CME, National Bureau of Standards, 1984.
5. G. A. Maria, and M. M. Fahmy, An l_p design technique for two-dimensional digital recursive filters, *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-22**, 1974, 15–21; also in [6].
6. S. K. Mitra, and M. P. Ekstrom, *Two-Dimensional Digital Signal Processing*, Dowden, Hutchinson, & Ross, Stroudsburg, PA, 1978.
7. H. E. Mutluay, and M. M. Fahmy, Recursibility of N-dimensional IIR digital filters, *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-32**, 1984, 397–402.
8. P. Pistor, Stability criterion for recursive filters, *IBM J. Res. Dev.* **18**, 1974, 59–71; also in [6].
9. W. K. Pratt, J.-F. Abramatic, and O. Faugeras, *Method and Apparatus for Improved Digital Image Processing*, U.S. Patent 4,330,833, 1982.
10. G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
11. S. Treitel, and J. L. Shanks, The design of multistage separable planar filters, *IEEE Trans. Geosci. Electron.* **GE-9**, 1971, 10–27; also in [6].