A Performance Analysis of Fast Gabor Transform Methods

Troy T. Chinen
Department of Mathematics, University of California at Davis, California

and

Todd R. Reed
Department of Electrical and Computer Engineering, University of California at Davis, California

Received February 20, 1996; revised January 10, 1997; accepted February 19, 1997

Computation of the finite discrete Gabor transform can be accomplished in a variety of ways. Three representative methods (matrix inversion, Zak transform, and relaxation network) were evaluated in terms of execution speed, accuracy, and stability. The relaxation network was the slowest method tested. Its strength lies in the fact that it makes no explicit assumptions about the basis functions; in practice it was found that convergence did depend on basis choice. The matrix method requires a separable Gabor basis (i.e., one that can be generated by taking a Cartesian product of one-dimensional functions), but is faster than the relaxation network by several orders of magnitude. It proved to be a stable and highly accurate algorithm. The Zak–Gabor algorithm requires that all of the Gabor basis functions have exactly the same envelope and gives no freedom in choosing the modulating function. Its execution, however, is very stable, accurate, and by far the most rapid of the three methods tested.

1. INTRODUCTION

The Gabor transform is a time–frequency transform which holds strong potential for utility in a variety of image processing applications. It was initially seen as a purely theoretical tool since there were no effective means by which it could be computed. Recently, however, there have emerged several algorithms to compute the Gabor transform; some of these algorithms are claimed to be very fast. Order-of-computation analyses have been made for some of these algorithms, but there is a lack of actual implementation data.

The Gabor expansion was introduced in 1946 by Dennis Gabor [1] to perform simultaneous time–frequency analysis of signals. He argued that the optimal representation for a signal is one which combines frequency and locality information. For this purpose he introduced a set of basis functions consisting of Gaussian windows modulated by complex exponentials. Bastiaans [2] dubbed this family of functions the Gabor elementary functions (GEF). He also generalized the notion of Gabor’s expansion to include non-Gaussian windows.

The Gabor transform has several features which make it attractive for image representation. First, the nonstationary nature of many natural signals [3, 4] mean that local, not global, frequency information is desirable; the Gabor transform provides this. The GEF also enjoy the property of minimal joint uncertainty. Each GEF is thus maximally concentrated in space and spatial frequency; this property carries over to higher dimensional GEF also [5, 6]. It has also been shown empirically [7] that the entropy of the Gabor transform of an image is much lower than the entropy of the pixel representation of the image. In fact, recent work indicates that Gabor expansions can provide better signal compression that can the DCT for very low bit rates [8]. The Gabor expansion is therefore recognized as a potential image compression tool [9–11].

1.1. Definitions

The Gabor transform, like the Fourier transform, has several forms depending on the dimensionality of the domain and range spaces. Previous work on Gabor representations has addressed the continuous case [12–15] and the relationship of the continuous to the discrete case [16]. This paper will focus on the finite dimensional (hence discrete) Gabor transform.

The GEF in 1D have the form

$$g_{mn}(x) = g_s(x - nD)e^{imW(x-nD)},$$

where \(g_s\) is a Gaussian window with variance \(\sigma^2\). If \(P\) is the number of points in the signal, we define \(N = P/D\) and \(W\) by the relation \(WM = 2\pi\). In this paper we confine the discussion to complete bases (i.e., \(D = M\)), although...
We briefly outline the role of biorthogonal functions in nonorthogonal expansions. Suppose that we have a signal $f$, a (nonorthogonal) basis $G = \{v_i\}$, and we desire coefficients $h_{ai}$ such that

$$f = \sum a_i v_i.$$  \hfill (4)

If a set of biorthogonal functions, $\Gamma = \{\gamma_i\}$, can be found with the defining property

$$\langle v_i, \gamma_i \rangle = \delta_{ij},$$  \hfill (5)

then the coefficients are easily determined since

$$\langle f, \gamma_i \rangle = \left\langle \sum_i a_i v_i, \gamma_i \right\rangle = \sum_i a_i \langle v_i, \gamma_i \rangle = a_i.$$

Thus, the signal–biorthogonal inner products, $\langle f, \gamma_i \rangle$, give the desired expansion coefficients. A similar argument shows that the signal–basis inner products $\langle f, v_i \rangle$ give the coefficients necessary for an expansion of $f$ in terms of the biorthogonal functions. Biorthogonal functions are actually computed quite often. Observe that (5) is really a matrix equation which states that the product of the row matrix of basis vectors with the conjugate of the column matrix of biorthogonal vectors is the identity matrix. Therefore, inverting a matrix whose columns are formed by a basis corresponds to computing the biorthogonal vectors.

1.3. **Gabor Transform vs STFT**

The STFT is computed by taking the Fourier transform of the product of the signal with a sliding window. We introduce the notation $\text{STFT}(w)$ to indicate the STFT using window $w$. A discrete version of the STFT using the 2D Gaussian window, the $\text{STFT}(g)$, is often confused with the Gabor transform. It is defined as the set of complex coefficients $\{c_{m,m,n,n}\}$, such that

$$c_{m,m,n,n} = \langle f, g_{m,m,n,n} \rangle.$$

We pause briefly to introduce the functions biorthogonal to the GEF analytically. It can be shown that these functions are formed by translation and modulation of a single 2D window. Bastiaans first computed the following analytic expression for the 1D version of this window [15].
Equations (8) and (7) show the duality between the two sets of functions \( \{ g_{m,m,n,n} \} \) and \( \{ \gamma_{m,m,n,n} \} \). Taking inner products with one family gives expansion coefficients for the other.

2. COMPUTATIONAL COMPLEXITIES OF AVAILABLE ALGORITHMS

Orr’s work [17] gives a complexity analysis of several methods of computing the Gabor transform. Specifically, biorthogonal, matrix, FFT-based biorthogonal, FFT-based the GEF, matrix, STFT, and Zak-transform methods are analyzed; the Zak-transform method was found to have the same complexity as the FFT. His theoretical results motivated the present work which, among other things, aims to carry out actual software implementation performance comparisons.

In this section we give complexities of the 2D versions of several computation schemes. Although the Gabor transform as stated in Section 1.1 is of primary interest, we will also point out which methods can compute more general transforms. We assume that the image under consideration is square with dimension \( P \times P \) and define a

where

The previous section shows that the expansion coefficients of \( f \) can be found by computing \( \langle f, \gamma \rangle \). This implies that the STFT(\( \gamma \)) computes the Gabor transform, i.e.,

Conversely, STFT(\( g \)) gives the expansion coefficients of \( f \) in terms of the functions \( \{ g \} \) which are biorthogonal to the GEF. In other words,

2.1. Biorthogonal Functions

The discussion of nonorthogonal bases (Section 1.2) shows that one method for calculating the Gabor coefficients is to use the functions which are biorthogonal to the GEF. The Gabor coefficients are then given by the signal–dual inner products. Straightforward computation of these inner products gives an operation count of \( P^4 \). Since the only requirement is that the basis is indeed a basis, decompositions much more general than (3) can be computed by this method (e.g., for a Gabor decomposition where \( \sigma_s \) may be spatially varying).

2.2. Matrix

Treating the expansion (3) as a system of linear equations means that matrix methods can be brought to bear on the problem. Taking advantage of the fact that (1) and (2) imply that the GEF form a separable basis yields a large reduction in computation compared to the previous method. The operation count in this case is \( 2P^3 \). Since the
involves finding the 2D Zak transform of the image and the window function, taking their quotient, and then performing a 4D IFFT. The division step requires \( P^2 \) operations, and since the 2D Zak transform consists of \( M^2 N \times N \) 2D FFTs, the operation count is \( P^2 [(2 + 2\alpha) \log_2 P + 1] \). This makes Zak–Gabor the only algorithm which has the same order of computation as the FFT.

2.6. Relaxation Network

The relaxation network proposed by Daugman [9] is an iterative scheme. It employs gradient descent over the space of Gabor coefficients. At each step, the STFT(\( g \)) coefficients are computed; the Gabor coefficients are then adjusted and the process repeats. The computation of each STFT(\( g \)) coefficient requires a fixed number of multiplies; in our experiments this number was 16 \( \times \) 16. Since there are as many STFT(\( g \)) values as pixels in the image, the resulting operation count is \( 2 \times 256^P \) per iteration.

only requirement is that the basis be separable, this method allows for much more general decompositions than (3).

2.3. FFT-Based Biorthogonal

Several methods use the biorthogonal functions and FFT methods together to increase efficiency over the previously described methods. The best of these [18] has an operation count of \( P^2 [P^{2\alpha} + 2(1 - \alpha) \log_2 P] \). We point out that all FFT based methods require GEF which are linearly spaced in frequency and are generated by a single window function.

2.4. FFT-Based Matrix

Balart’s FFT based matrix formulation [19] has been given only in the 1D case. It solves for the coefficients \( \{a_{mn}\} \) in

\[
f(x) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} a_{mn}\]

algebraically, i.e., by treating it as a matrix equation \( \mathbf{f} = \mathbf{G} \mathbf{a} \). The method assumes that \( \mathbf{G} \) is block lower diagonal, which, along with other structures present in \( \mathbf{G} \), allows for a solution with an operation count of \( P [P^{2\alpha} + (1 - \alpha) \log_2 P] \). The assumption on \( \mathbf{G} \) implies that as the window is translated to different spatial locations it leaves behind zeros in its wake (see Fig. 3). This assumed behavior means that some symmetric windows, e.g., a Gaussian, cannot be used in this method.

2.5. Zak–Gabor

Computation of the Gabor coefficients via the Zak transform [20], or the Zak–Gabor transform as it is often called, involves finding the 2D Zak transform of the image and the window function, taking their quotient, and then performing a 4D IFFT. The division step requires \( P^2 \) operations, and since the 2D Zak transform consists of \( M^2 N \times N \) 2D FFTs, the operation count is \( P^2 [(2 + 2\alpha) \log_2 P + 1] \). This makes Zak–Gabor the only algorithm which has the same order of computation as the FFT.

3. The Algorithms to Be Tested

There were several considerations which led to selecting the matrix method, Zak–Gabor, and relaxation network as the Gabor transform algorithms to be tested. The Zak–Gabor transform was selected because of its comparability to the FFT in terms of execution speed. The matrix method was chosen as a fast biorthogonal method that does not employ FFT techniques. The algorithm is fairly general in terms of what types of basis functions are allowed, and it functions without requiring an excessively large amount of machine resources. The use of the relaxation network has been documented in many papers, but details on its performance are sparse. It is unique in that it determines the Gabor transform iteratively and puts no explicit constraints on the basis functions.

Following is a description of each algorithm along with a discussion of any choices that had to be made in order to implement them.

3.1. Matrix

The method proposed by Ebrahimi et al. [10, 21] treats (3) as matrix equation. By taking advantage of the separability of the GEF, the equation can be recast in terms of matrices of smaller dimension, thus making a direct solution possible. For simplicity we discuss only the case where the decomposition basis is complete.

The separability of the 2D GEF allows (3) to be expressed as

\[
F = \mathbf{GAG}^T,
\]
where

\[
F = \begin{pmatrix}
f(0, 0) & \ldots & f(0, P - 1) \\
\vdots & \ddots & \vdots \\
f(P - 1, 0) & \ldots & f(P - 1, P - 1)
\end{pmatrix},
\]

\[
A = \begin{pmatrix}
a_{0,0} & \ldots & a_{0,p-1} \\
\vdots & \ddots & \vdots \\
a_{p-1,0} & \ldots & a_{p-1,p-1}
\end{pmatrix},
\]

and

\[
G = \begin{pmatrix}
g_{0}(0) & \ldots & g_{p-1}(0) \\
\vdots & \ddots & \vdots \\
g_{0}(P - 1) & \ldots & g_{p-1}(P - 1)
\end{pmatrix}.
\]

The subscripts \( k \) and \( l \) on \( \{a_{k,l}\} \) represent the space/frequency index pairs \((m_{x}, n_{x})\) and \((m_{y}, n_{y})\), respectively. The subscript on the 1D GEF is similar. If \( G \) is not singular, the solution can be obtained by

\[
\hat{A} = (G^{T}G)^{-1}G^{T}FG(G^{T}G)^{-1}.
\]

Note that the matrix to be inverted, \( G^{T}G \), is \( P \times P \) as opposed to \( P^2 \times P^2 \) as would be in the basic biorthogonal method.

The matrix Gabor decomposition therefore consists of two steps. First precompute the matrix \( H = (G^{T}G)^{-1}G^{T} \) and \( H^{-1} \). The Gabor coefficients are then given by \( \hat{A} = HFH^{-1} \). Since the matrix \( H \) can be precomputed, the speed of computation is dependent only on the matrix multiplication in the second step.

### 3.2. Zak–Gabor

The properties of the continuous Zak transform [22] and its relation to the Gabor transform have been well studied [20]. The following derivation of the Zak–Gabor transform reveals the methods’ strengths and weaknesses, namely, the reason for its high speed, as well as its tacit use of wrapped GEF (GEF near the edge of the image which have support, say, on both opposite edges), and the singularity associated with the Gaussian window [23, 24, 4, 16, 25].

#### 3.2.1. The 1D Zak–Gabor transform

Suppose that a \( P \) point 1D signal \( f \) has the Gabor representation

\[
f(x) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} a_{m,n} g(x - nD)e^{imWx}.
\]

The finite, discrete Zak transform of a signal \( f \) is defined to be

\[
Zf(r, \theta) = \sum_{k=0}^{N-1} f(r + kD)e^{-ikV\theta},
\]

where \( 0 \leq r < D, 0 \leq \theta < N \), and \( VN = 2\pi \). Taking the Zak transform of both sides of (10), multiplying and dividing by \( \exp(imV\theta) \), and rearranging leads to

\[
Zf(r, \theta) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} a_{m,n} e^{imWr-imV\theta}
\]

\[
\cdot \sum_{k=0}^{N-1} g(r + (k - m)D)e^{i(k-m)V\theta},
\]

which, under conditions to be discussed, reduces to

\[
\frac{Zf(r, \theta)}{Zg(r, \theta)} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} a_{m,n} e^{imWr-imV\theta}.
\]

We see that the coefficients of the Gabor expansion can be recovered by inverting the 2D discrete Fourier series. Since the computation of \( Zg, Zf \), and the final inversion are all DFTs, the Zak–Gabor algorithm operates with an execution speed proportional to that of the FFT. Also, note that a Zak–Gabor inverse transform can be obtained simply by reversing the steps in the forward transform.

Working out the details to obtain (12) shows that the structure of the GEF has been fully exploited in this algorithm. The facts that the same 2D Gaussian is employed at every spatial location and that modulating frequencies are linearly spaced are necessary conditions. It is also important that the set of GEF form a complete basis, i.e., \( D = M \). Following is a discussion of some other Zak–Gabor considerations.

#### 3.2.2. Periodicity of Zak–Gabor basis functions

The latter factor in (12)

\[
\sum_{k=0}^{N-1} g(r + (k - m)D)e^{i(k-m)V\theta}
\]

must coincide with \( Zg \) (the Zak transform of the window \( g \)) for every \( m \) for the derivation to go through. It suffices to extend \( g \) by periodicity to definition on \( \{-P, \ldots, P\} \). Using (13) in a computational setting therefore means that the GEF are implicitly periodized or “wrapped around.”
The Zak–Gabor coefficients are therefore recovered by performing a 4D IFFT on the quotient \( \frac{Zf}{Zg} \).

### 3.3. Relaxation Network

Daugman [9] developed a method for iteratively computing the Gabor transform coefficients. It places no explicit restrictions on the basis function, although in practice the support of each basis function must be limited to keep the computational load reasonable.

#### 3.3.1. Development of the Iteration Scheme

The relaxation networks’ iterative scheme is based on minimizing the error functional

\[
E = \left\| f - \sum_i a_i g_i \right\|^2
\]

where the bar indicates complex conjugation. Computing the partial derivatives of \( E \) (i.e., the gradient of \( E \)) gives

\[
\frac{\partial E}{\partial a_k} = -\langle g_k, f \rangle + \langle g_k, \sum_i a_i g_i \rangle
\]

by assuming \( \frac{\partial(z)}{\partial z} = 0 \) (see [26]). Applying the method of gradient descent then gives the update rule

\[
a_k^{(n+1)} \leftarrow a_k^{(n)} + \mu \left( \langle g_k, f \rangle - \langle g_k, \sum_i a_i g_i \rangle \right),
\]

where the \( \mu > 0 \) ensures that movement proceeds in the direction opposite to the gradient.

#### 3.3.2. Comments about the Relaxation Network

The relaxation network allows for very general decompositions such as incomplete or wavelet-like expansions. It has been shown that Daugman’s relaxation network actually performs Jacobi iteration on a set of simultaneous linear equations [27]. Faster convergence is therefore possible by using alternative weight adjustment strategies such as Gauss–
Seidel iteration, successive over relaxation, conjugate gradient methods, and multigrid methods. Block partitioning strategies have also been suggested to gain additional speed [28], but this seems to assume orthogonality of the GEF.

The primary disadvantage of any iterative scheme is often execution time. Furthermore, the learning constant $\mu$ must be adjusted for maximal convergence rate on each image. It is also not known in advance how many iterations will be necessary to reach an acceptable error level.

4. ALGORITHM IMPLEMENTATION

The matrix, Zak–Gabor, and relaxation network algorithms were implemented in software and tested on an HP 715, 75 MHz workstation. To facilitate a machine-independent comparison, Table 1 shows the performance of several other computers via the Spec benchmark, relative to that used for this performance analysis. All code was written in C; computations were performed in double precision. Three $64 \times 64$ images were used in the testing process. The standard “lenna” and “fax1” images were subsampled to the lower resolution and saved at 8 bit resolution. The third “image” consisted of the GEF $g_{0,0,3,4}$, i.e., an unmodulated Gaussian having spatial center index $(3, 4)$. We set $M_x = M_y = D_x = D_y = \sigma _x^2 = \sigma _y^2 = 8$ pixels as this gives a reasonable number of frequencies and sufficient overlap of the GEF. The exception to this was in the case of the relaxation network, where convergence difficulties necessitated the use of narrower Gaussians, i.e., $\sigma ^2 < 8$ (in the experiments which follow, $\sigma ^2 = 6$ and 8 pixels).

Image processing considerations dictate that the GEF should be shifted so that the union of their support is centered on the image. The GEF $g_{0,0,0,0}$, for example, should be spatially centered at $((D_x - 1)/2, (D_y - 1)/2)$ instead of at the origin as $(1)$ would imply. This spatial shift necessitates a complementary shift in the phase of the GEF so that the symmetry of the GEF is preserved.

In the matrix and relaxation networks these changes are straightforward; we show in section 4.2 how this is done in the Zak–Gabor case.

For reasons to be discussed, the decomposition bases used in each method were slightly different. Figure 5 graphically illustrates these differences in the decomposition bases. While using the same reconstruction basis for all three methods would be optimal (to facilitate comparison of the algorithms), theoretical and practical considerations made this impossible. The matrix basis was used for reconstruction in the matrix and Zak–Gabor algorithms, but the relaxation network reconstruction was effected using its decomposition basis set. Details are given in Section 4.3.

In the following sections we describe implementation issues specific to each of the methods, e.g., matrix solution methods and basis choice.

4.1. Matrix

It has been verified experimentally that the conditioning of the matrix which must be inverted becomes worse as the variance of the Gaussian increases. If it is necessary to work with such a matrix, singular value decomposition (SVD) can be used to stabilize the inversion process [29]. For these experiments the LU decomposition was used.

4.2. Zak–Gabor Issues

It is not obvious how to shift the Zak–Gabor basis functions in space and phase since they are defined implicitly in the algorithm. By substituting these shifted GEF into (16) we obtain the relation

$$\begin{align*}
Zf(r_x, r_y, \theta _x, \theta _y) &= \sum_{m_x=0}^{M_x-1} \sum_{m_y=0}^{M_y-1} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} a_{m_x, m_y, n_x, n_y} e^{im_x W x \phi _x + in_x W y \phi _y} \\
Zg(r_x, r_y, \theta _x, \theta _y) &= e^{(-m_x V x \phi _x - m_y V y \phi _y) n_x r_x + n_y r_y} \tag{22}
\end{align*}$$

where $\tilde{g}$ is the spatially shifted window and $(\phi _x, \phi _y)$ is the phase shift.

The difference between the analysis (wrapped) and synthesis bases (unwrapped) implies that the reconstructed image will not coincide with the original image even if the Zak–Gabor transform is invertible. Since the wrapped and unwrapped bases differ near the edges of the image, we anticipate error in the reconstructed image near its edges. We emphasize that this reconstruction error is distinct from error in the algorithm itself. To make meaningful comparisons, the MSE was computed after 22 pixels had been trimmed from the edges of the reconstructed image.

4.3. Relaxation Network

Equation (21) implies the computation of $P^2$ inner products each requiring $P^2$ multiplies. To make this a viable

### Table 1

<table>
<thead>
<tr>
<th>System</th>
<th>FPU (MHz)</th>
<th>SPEC fp92</th>
<th>SPEC int92</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGI Power Challenge</td>
<td>75.0</td>
<td>310.6</td>
<td>108.7</td>
</tr>
<tr>
<td>HP 9000/715</td>
<td>75.0</td>
<td>108.2</td>
<td>54.8</td>
</tr>
<tr>
<td>HP 9000/715</td>
<td>50.0</td>
<td>72.1</td>
<td>36.5</td>
</tr>
<tr>
<td>HP 9000/715</td>
<td>33.0</td>
<td>45.0</td>
<td>24.2</td>
</tr>
<tr>
<td>Sun SPARCstation 10/41</td>
<td>40.0</td>
<td>63.4</td>
<td>53.3</td>
</tr>
<tr>
<td>Sun SPARCstation 10/30</td>
<td>36.0</td>
<td>41.1</td>
<td>40.0</td>
</tr>
</tbody>
</table>

*Note. Values for the HP 9000/715 75 MHz were linearly extrapolated from the 50 and 33 MHz data.

*Source. Anonymous ftp at ftp.nosc.mil in 'pub/aburto'.

The primary disadvantage of any iterative scheme is often execution time. Furthermore, the learning constant $\mu$ must be adjusted for maximal convergence rate on each image. It is also not known in advance how many iterations will be necessary to reach an acceptable error level.

4. ALGORITHM IMPLEMENTATION

The matrix, Zak–Gabor, and relaxation network algorithms were implemented in software and tested on an HP 715, 75 MHz workstation. To facilitate a machine-independent comparison, Table 1 shows the performance of several other computers via the Spec benchmark, relative to that used for this performance analysis. All code was written in C; computations were performed in double precision. Three $64 \times 64$ images were used in the testing process. The standard “lenna” and “fax1” images were subsampled to the lower resolution and saved at 8 bit resolution. The third “image” consisted of the GEF $g_{0,0,3,4}$, i.e., an unmodulated Gaussian having spatial center index $(3, 4)$. We set $M_x = M_y = D_x = D_y = \sigma _x^2 = \sigma _y^2 = 8$ pixels as this gives a reasonable number of frequencies and sufficient overlap of the GEF. The exception to this was in the case of the relaxation network, where convergence difficulties necessitated the use of narrower Gaussians, i.e., $\sigma ^2 < 8$ (in the experiments which follow, $\sigma ^2 = 6$ and 8 pixels).

Image processing considerations dictate that the GEF should be shifted so that the union of their support is centered on the image. The GEF $g_{0,0,0,0}$, for example, should be spatially centered at $((D_x - 1)/2, (D_y - 1)/2)$ instead of at the origin as $(1)$ would imply. This spatial shift necessitates a complementary shift in the phase of the GEF so that the symmetry of the GEF is preserved.

In the matrix and relaxation networks these changes are straightforward; we show in section 4.2 how this is done in the Zak–Gabor case.

For reasons to be discussed, the decomposition bases used in each method were slightly different. Figure 5 graphically illustrates these differences in the decomposition bases. While using the same reconstruction basis for all three methods would be optimal (to facilitate comparison of the algorithms), theoretical and practical considerations made this impossible. The matrix basis was used for reconstruction in the matrix and Zak–Gabor algorithms, but the relaxation network reconstruction was effected using its decomposition basis set. Details are given in Section 4.3.

In the following sections we describe implementation issues specific to each of the methods, e.g., matrix solution methods and basis choice.

4.1. Matrix

It has been verified experimentally that the conditioning of the matrix which must be inverted becomes worse as the variance of the Gaussian increases. If it is necessary to work with such a matrix, singular value decomposition (SVD) can be used to stabilize the inversion process [29]. For these experiments the LU decomposition was used.

4.2. Zak–Gabor Issues

It is not obvious how to shift the Zak–Gabor basis functions in space and phase since they are defined implicitly in the algorithm. By substituting these shifted GEF into (16) we obtain the relation

$$\begin{align*}
Zf(r_x, r_y, \theta _x, \theta _y) &= \sum_{m_x=0}^{M_x-1} \sum_{m_y=0}^{M_y-1} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} a_{m_x, m_y, n_x, n_y} e^{im_x W x \phi _x + im_y W y \phi _y} \\
Zg(r_x, r_y, \theta _x, \theta _y) &= e^{(-m_x V x \phi _x - m_y V y \phi _y) n_x r_x + n_y r_y} \tag{22}
\end{align*}$$

where $\tilde{g}$ is the spatially shifted window and $(\phi _x, \phi _y)$ is the phase shift.

The difference between the analysis (wrapped) and synthesis bases (unwrapped) implies that the reconstructed image will not coincide with the original image even if the Zak–Gabor transform is invertible. Since the wrapped and unwrapped bases differ near the edges of the image, we anticipate error in the reconstructed image near its edges. We emphasize that this reconstruction error is distinct from error in the algorithm itself. To make meaningful comparisons, the MSE was computed after 22 pixels had been trimmed from the edges of the reconstructed image.

4.3. Relaxation Network

Equation (21) implies the computation of $P^2$ inner products each requiring $P^2$ multiplies. To make this a viable
computation scheme, the support of each was reduced to a $16 \times 16$ square. These basis functions were also wrapped to satisfy the constraint $\langle g_i, g_j \rangle = 1$ while keeping the amplitudes of the GEF uniform. During each iteration, a complete STFT$(g)$ decomposition and a complete inverse Gabor transform (using the wrapped GEF) are carried out. Ideally, the MSE after each iteration would be measured after reconstruction with the unwrapped GEF, but the associated computational costs were prohibitive. The truncated, wrapped basis set was therefore used in both the reconstruction and MSE computation. The truncation introduces a small error, but since the basis functions are modulated Gaussians, the amplitude at which this truncation occurs is very small. In fact, for $\sigma^2 = 6$ pixels, the upper bound on the values which will be set to zero is about 0.004% of the maximal amplitude of the GEF.

The relaxation network requires the choice of a learning constant $\mu$. Choosing $\mu$ too small slows convergence; choosing $\mu$ too large often causes the MSE to grow without bound. In these tests $\mu$ took on the values 3, 2, 1, 0.5, 0.1, and 0.05. The results are given using the largest value of $\mu$ for which convergence was observed.

5. RESULTS

The matrix, Zak–Gabor, and relaxation network algorithms were tested in terms of execution speed, accuracy, and stability under different values of $\sigma^2$, the variance of the Gaussian.

5.1. Speed

The results of the speed tests shown in Table 2 demonstrate convincingly that the Zak–Gabor algorithm is much faster than either the matrix or relaxation algorithms. For a $64 \times 64$ image the Zak–Gabor algorithm executes approximately 9 times faster than the matrix inversion algorithm; for a $256 \times 256$ image it is faster by a factor of 47. The order-of-computation analysis implies that the difference in performance between the Zak–Gabor and matrix algorithms should increase as larger images are considered.

The relaxation network is slower than either of the other methods by several orders of magnitude. Note that the number of computations per iteration is fixed, being determined by the size of the image and the support of the basis functions. Thus, all figures in Table 2 are independent of image content. Of course, the number of iterations could be dependent on the particular image; the data to follow will show this to be the case.

5.2. Accuracy

The error measurements that are given in Table 3 show that the Zak–Gabor and matrix inversion methods have exceptionally high degrees of accuracy. The error values tabulated for the relaxation network are really a measure of convergence speed, since in principle the error can be made as small as desired by taking more and more iterations.

Figure 6 shows semi-log plots of the relaxation MSE versus iteration for the three images tested with $\sigma^2 = 6$ pixels. Observe that aside from the initial drop in MSE, the plots show a linear trend, indicating that the MSE is exponentially decreasing. Note that approximately 10 hours were required to obtain an MSE of $10^{-9}$.

In the relaxation network the STFT$(g)$ coefficients are used as initial approximations to the Gabor coefficients. The MSE graph shows that the initial MSE was very high—on the order of 1000. This experimentally confirms that the STFT$(g)$ coefficients should not be assumed to be good approximations to the Gabor coefficients.

<table>
<thead>
<tr>
<th>TABLE 2 Algorithm Execution Times(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image size</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>$64 \times 64$</td>
</tr>
<tr>
<td>$256 \times 256$</td>
</tr>
</tbody>
</table>
TABLE 3
Mean Squared Error

<table>
<thead>
<tr>
<th>Image</th>
<th>Zak–Gabor(^a)</th>
<th>Matrix</th>
<th>Relaxation network</th>
</tr>
</thead>
<tbody>
<tr>
<td>basis function</td>
<td>$7.55 \times 10^{-28}$</td>
<td>$2.51 \times 10^{-27}$</td>
<td>6.72 \times 10^{-4}</td>
</tr>
<tr>
<td>fax</td>
<td>$3.93 \times 10^{-26}$</td>
<td>$3.45 \times 10^{-26}$</td>
<td>2.57 \times 10^{-3}</td>
</tr>
<tr>
<td>lenna</td>
<td>$2.67 \times 10^{-26}$</td>
<td>$9.97 \times 10^{-27}$</td>
<td>4.43 \times 10^{-3}</td>
</tr>
</tbody>
</table>

\(^a\) The Zak–Gabor reconstructed images were trimmed by 22 pixels before the MSE was computed.

5.3. Stability

The stability of each algorithm was measured in terms of its general performance versus the variance, $\sigma^2$, of the Gaussian window. This is an important feature of an algorithm since different applications might require larger or smaller $\sigma^2$. It is therefore desirable to know when a computation scheme breaks down and its accuracy within the limits of proper operation.

The matrix algorithm was the most straightforward to test for stability. The decomposition was performed over a range of values for the variance, and the MSE was noted for each reconstruction. The results in Fig. 7 show that the method is quite robust; the MSE is below $10^{-10}$ for $2 \leq \sigma^2 \leq 34$.

The Zak–Gabor algorithm encountered no problems in the computation of the coefficients for $2 \leq \sigma^2 \leq 10,000$. The data was exactly recovered in each case using the Zak–Gabor inverse transform.

The relaxation network encountered some serious convergence problems as the variance of the Gaussian was increased. In fact, convergence was only observed for $\sigma^2 < 8$ pixels. Figure 8 shows the oscillatory behavior exhibited by the network for $\sigma^2 = 8$ pixels.

5.4. Discussion

The matrix algorithm is fast, but the Zak–Gabor is much faster. It could be argued that comparing the relaxation network on a serial machine to the other two methods is unfair, since networks are designed to be implemented on a parallel architecture. There exist, for example, massively parallel processor arrays which could make the network a
5.5. Sources of Error

The matrix method incurred very little error; the reconstruction is exact, up to machine precision.

The nature and magnitude of the Zak–Gabor reconstruction error are shown in Fig. 9. This artifact is a result of the differing decomposition and reconstruction bases. In fact, perfect reconstruction was demonstrated when the Zak–Gabor inverse transform was used. The reconstruction error in the Zak–Gabor algorithm can be compensated for simply by zero padding around the image. The reconstructed Zak–Gabor images were trimmed by 22 pixels to achieve a very low MSE, but for many practical applications this degree of accuracy is unnecessary.

The relaxation networks’ oscillatory behavior appears to be a manifestation of instability inherent in the algorithm. The fact that the basis functions interact may be the cause of this instability. This hypothesis is supported by the data, since decreasing the Gaussian variance leads to less interaction among the basis functions and better convergence.

6. CONCLUSIONS

The matrix, Zak–Gabor, and relaxation network algorithms for Gabor transform are tested in this work. The data show that both the matrix and Zak–Gabor methods are fast and extremely accurate methods. The network, while not rapid in execution, provides a way of computing not only the Gabor transform, but also decompositions much more general.

We point out that the speed results are exactly in accord with the degree to which the algorithms exploit the structure in the GEF. The relaxation network assumes nothing (explicitly) about the basis functions and consequently must work very hard to obtain the decomposition. The matrix method exploits separability to gain an enormous increase in speed over the network. The Zak–Gabor algorithm uses the linear spacing in frequency, the complete-

**FIG. 8.** Relaxation network log(MSE) vs iteration for $\sigma^2 = 8$ pixels.

**FIG. 9.** Zak–Gabor reconstruction error. The error is concentrated around the edges where the decomposition and reconstruction bases differ.

\[
\bar{A} = BFB',
\]

where $B$ and $B'$ are precomputed, $F$ is the image, and $\bar{A}$ is the transformed image. One possibility is to view this as two successive matrix multiplications and each multiplication as a large number of inner products. If one processor were available for each inner product (i.e., one per pixel), it is conceivable that the computation time could decrease by a factor of $P$, the number of pixels. The Zak–Gabor execution time could be decreased by employing a processor for each 2D frequency pair. Since the Zak transform requires $M_xM_y$ 2D FFTs, it could be computed more quickly by a factor of $M_xM_y$ by employing multiple processors.
ness of the transform, and the single generating window to achieve FFT-like computation speed.

Future areas of research include implementation of the relaxation network on a parallel architecture and an analysis of the convergence properties of the network. A parallel implementation of the Zak–Gabor algorithm could make real time Gabor filtering a possibility.

REFERENCES