Efficient Multiscale Template Matching with Orthogonal Wavelet Decompositions

Sumit Basu
Perceptual Computing Section, The MIT Media Laboratory
20 Ames St., Cambridge, MA 02139 USA
sbasu@media.mit.edu

Abstract:

I develop a method for efficient, multiscale template matching using orthogonal wavelet representations. I also develop an efficient approximate mechanism for initially finding the "correct" values for the basis coefficients a_{jk} at the finest scale (as opposed to using the sample values as coefficients). I then present applications of these techniques to two important image coding/computer vision tasks: motion estimation and multiple model matching (model recognition).

Keywords: wavelets; template matching; multiscale representations; multiscale algorithms; motion estimation; multiple model matching; model recognition; approximate signal processing.

1 Introduction

In many signal processing applications, the task of template matching is an important one. The basic idea is simple: we have some piece of signal $r(t)$ (the template), and we want to find where (or whether) it occurs within another signal $x(t)$ (see figure 1). In 1D, this problem appears in radar, where we want to find the known "pulse" signal after it has been reflected by an obstacle. It appears in communications channels, where we want to distinguish various symbols from each other [3]. It also shows up in simple speech recognition systems, where we wish to match an input signal with a set of known utterances [4]. In 2D, the problem comes up in motion estimation/tracking tasks, where we want to see where a piece of an image in one frame goes to in the next frame. It is also seen in model matching, when an image (or piece thereof) is to be matched with a series of known images (models).

Unfortunately, performing the template match at every point in a target signal is very expensive. In this study, I present a multiresolution approach to template matching using orthogonal wavelet representations of the signal and the template. I develop a coarse to fine scheme which finds a rough match at the coarsest scale of representation and then refines the estimated position/match value by moving through increasing levels of resolution. This is far cheaper computationally than searching the entire space at the finest scale. This is certainly not the first instance of a multiresolution approach to template matching. However, there are three significant advantages to using orthogonal wavelets over other decompositions (*e.g.* the Laplacian pyramid of [1], which is another compact-support multiresolution representation). First, the computations performed at the coarse scale can be reused at finer scales. The second advantage is that the wavelet representation may be available for free if a given video/audio device is using a wavelet representation for coding as well. Last but certainly not least, there is a *precise* interpretation of the operations performed at the different scales, as I will later explain.

After presenting the theory behind my approach, I will also show the applications of the technique to two image processing/computer vision problems: motion estimation (in an egomotion task) and model matching (in a simple facial expression recognition task).

1.1 Template Matching with Normalized Correlation

In all of the applications described above, the signal processing technique most often used for template matching is the "matched filter" [3]. The matched filter is simply a time-reversed version of the template:

$$r'(t) = r(-t) \tag{1}$$

This filter is then convolved with the input:

$$d(t) = r'(t) * x(t) \tag{2}$$

This technique derives from the properties of the autocorrelation function: when $r(t)$ is convolved with $r(-t)$, the output is maximized at the origin. As a result, we expect to see a peak in $d(t)$ where $x(t)$ contains $r(t)$. The autocorrelation signal and the convolution with the test sequence for the example in figure 1 is shown in figure 2. We see the expected peak

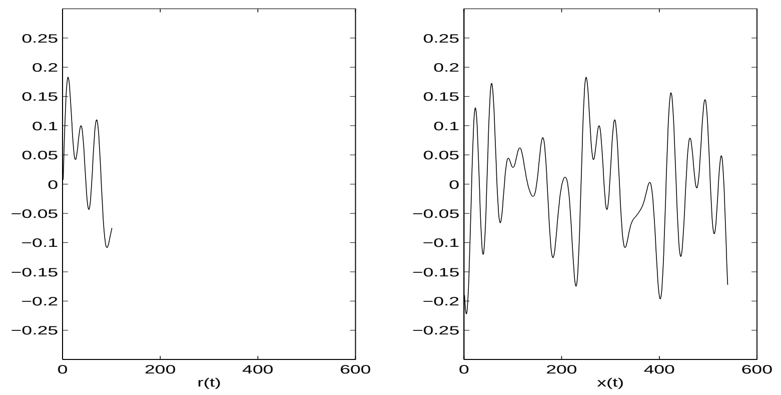

Figure 1: The template matching problem with template $r(t)$ and signal $x(t)$

at the origin for the autocorrelation function and a peak in the convolution with the test signal at $t = 240$. This is also expected, as the template for this example was chosen as the piece of $x(t)$ from $t = 240$ to $t = 340$ (see 1).

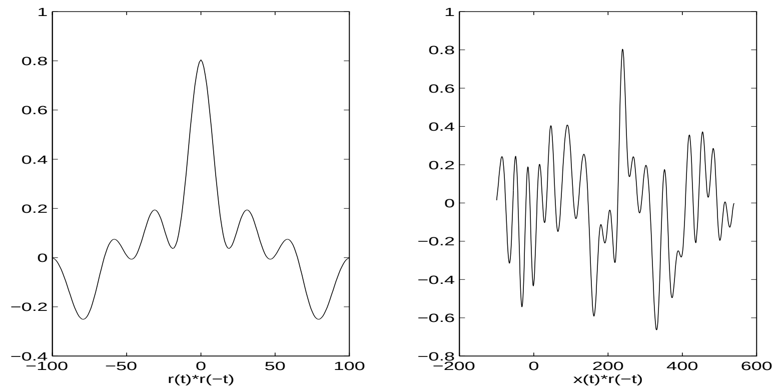

Figure 2: The autocorrelation of $r(t)$ and the convolution of $r(-t)$ with the test sequence $x(t)$

Another way to view this operation is in terms of an inner product integral. We can rewrite the convolution result $d(t)$ at time T as the integral of the inner product of the signal and the shifted template:

$$d(t) = \int_{-\infty}^{\infty} x(t)r(t-T)dt \tag{3}$$

or in discrete time,

$$d[N] = \sum_{k=-\infty}^{\infty} x[k]r[k-N]dt \tag{4}$$

Because the template is typically finite support, the product signal inside the integral/summation is as well. This inner product view of template matching will be critical in the development of my wavelet-based template matching scheme.

There is a significant problem with this approach as it stands: it does not account for the relative magnitudes of the signal and the template. If $r(t)$ is always positive, then the larger (more positive) the values of $x(t)$, the greater the inner product integral will be, regardless of how well the signal matches the template. To account for this, the following "normalized correlation" form is often used:

$$d(T) = \frac{\langle x(t), r(t-T)\rangle}{\|x_r(t)\|\|r(t)\|} \tag{5}$$

Where $x_r(t)$ is the relevant piece of $x(t)$, i.e., that piece which lies within the support of $r(t-T)$. This is the inner product of the two signal pieces divided by their norms, which can be interpreted as the cosine between the signals in vector space. The value is thus bounded at 1 and -1. Because of the normalization, this result is invariant to the relative scaling of the signals. The closer signals are to each other, the closer the result will be to 1. We can rewrite this expression more explicitly as

$$d(T) = \frac{\int x(t)r(t-T)dt}{\sqrt{\int x_r^2(t)dt}\sqrt{\int r^2(t)dt}} \tag{6}$$

In discrete time, the vector space interpretation is the same, and we can rewrite equation 6 as

$$d[N] = \frac{\sum_k x[k]r[k-N]}{\sqrt{\sum_k x^2[k]}\sqrt{\sum_k r^2[k]}} \tag{7}$$

Though this is a simple expression, it can be very expensive to compute. In the tracking/detection applications described above, we must compute the expressions in equations 6 and 7 for all T in the test signal $x(t)$. Only then can we take the maximum of the resulting $d(t)$ to find the best match. We would like to have a computationally more efficient way of finding this match. As the reader may expect by now, wavelets will provide the better way.

2 The Wavelet Advantage

The wavelet decomposition breaks a function down into a basis of function pieces. For orthogonal wavelets, it begins by finding the projections a_{Nk} of the function onto the shifts of the finest resolution scaling function $\phi(2^N t-k)$ (we will get to how we find these projections in the next section). This is the representation of the function in the basis V^N. Using Mallat's technique, we can then break these coefficents into the coefficients for V^{N-1} and W^{N-1}. We then continue to break down the coefficients until we reach V^0, which will be the coarsest level of representation. The basis for V^0 is $\phi(t-k)$. By combining this basis and W^0, we can reconstruct the function in basis V^1 (i.e., we can recover the a_{1k}'s). The breakdown and the paths for splitting and recombining bases is shown in figure 3 below.

Because the basis functions are compact support, we can represent a finitely supported function in V^j for any j with a finite number of coefficients. This is because the projections onto the basis functions outside the support of the target function will always be zero.

Notice that this breakdown gives us a multiresolution representation of our signal. By appropriately recombining coefficients, we can find the representation of the signal in terms

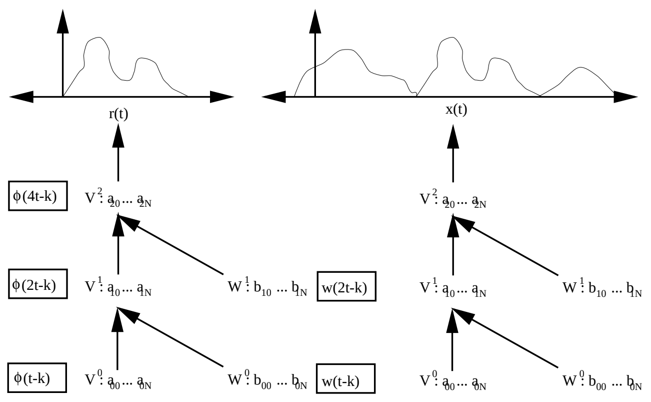

Figure 3: A breakdown of two signals into a wavelet basis

of $\phi(2^j t - k)$ for any j between 0 and N. This is where template matching can become easier. Since we can have a representation of the signal and the template at multiple resolutions (from coarse to fine), we can find an approximate match between the small, coarse images and then refine our estimate at the higher levels. The key point here is that since we will already have a coarse estimate of the position (in a detection task), we only need to search in a small neighborhood in the next finer resolution. The *global* search is necessary only at the coarsest (and computationally cheapest!) scale.

This still leaves us with a problem. We still need to compute the inner product integral of equation 3. In general, if two functions $f(t)$ and $g(t)$ are built up of function pieces $s_0(t)$, $s_1(t)$, etc., so that

$$f(t) = c_0 s_0(t) + c_1 s_1(t) + \cdots \tag{8}$$
$$g(t) = d_0 s_0(t) + d_1 s_1(t) + \cdots \tag{9}$$

then the inner product will contain all the pairwise inner products of all of the function pieces:

$$f(t)g(t) = c_0 d_0 s_0^2(t) + c_0 d_1 s_0(t) s_1(t) + \cdots \tag{10}$$
$$= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} c_m d_n s_m(t) s_n(t) \tag{11}$$

This results in n^2 inner products for a basis with n functions. For the first few levels (V^0,V^1, etc.), this is still a relatively small amount of computation. But by the time we get to V^N, which has as many coefficients (n) as our original signal had samples, the inner product in an arbitrary basis takes n^2 computations whereas it only takes n computations in the original signal representation!

2.1 Orthogonality Has Its Privileges

Of course, we will not be using an arbitrary basis. We will not even be using arbitrary wavelets! We will use only *orthogonal* wavelets. Orthogonal wavelets have the wonderful property that they are orthogonal to everything else in their basis family. Not only are the $\phi(2^j t - k)$'s orthogonal to their integer shifts, they are orthogonal to the $w(2^l t - k)$'s for all l and the $\phi(2^l t - k)$'s for all $l \neq j$. As a result, most of the terms in equation 11 are now zero! The only terms that remain involve the inner products of a basis piece with itself:

$$\begin{aligned} f(t)g(t) &= c_0 d_0 s_0^2(t) + c_1 d_1 s_1^2(t) + \cdots && (12) \\ &= \sum_{n=0}^{N-1} c_n d_n s_n^2(t) && (13) \end{aligned}$$

This takes us from n^2 terms to only n terms! Let us now return to the quantity we are trying to compute (equation 3. We need to find the *integral* of the inner product function. In terms of our basis, this is

$$\int \sum_{n=0}^{N-1} c_n d_n s_n^2(t) dt \qquad (14)$$

$$= \sum_{n=0}^{N-1} \int c_n d_n s_n^2(t) dt \qquad (15)$$

If we choose an *orthonormal* wavelet basis, we have the further advantage that the norm of each basis function is unity:

$$\int s_n^2(t) dt = 1 \qquad (16)$$

As a result, we can rewrite equation 15 as a simple summation:

$$\sum_{n=0}^{N-1} c_n d_n \qquad (17)$$

At the finest resolution V^R (where a shift in k corresponds to a shift of one sample), this involves the same number of computations as the time domain implementation (in discrete time), and at each lower resolution, the number of operations is cut by a factor of 2 for 1D signals and a factor of 4 for 2D.

What does it mean to use the coefficents at a lower resolution (i.e., using a_{jk} where $k < R$? For the wavelet basis, there is a precise and pleasing answer: it simply means that we are computing the *exact* inner product integral of the function and the template *as represented in the space* V^j. In fact, it is also the *exact* inner product integral of the two functions with one represented *exactly* (i.e., in V^∞) and the other represented in V^j. This is because the "higher order" components of the former would be completely orthogonal to the latter, so their contribution to the inner product integral would be zero. This is a very powerful interpretation. We are using a numerical method (summing mutiples of coefficients) to compute an inner product integral exactly. This is not ordinary approximate signal processing - in this case, we know exactly what the approximation to the signal is. In addition, Parseval (orthogonality strikes again) allows us to express exactly what fraction of power a given level of representation holds. If either the signal or the template does not have any power above a certain level (i.e., if it is "band-limited" in the wavelet sense as opposed to the Fourier sense), we can compute the inner product integral exactly by just doing the computation at this level.

Furthermore, since the wavelet bases we are interested in are orthogonal, we have

$$V^0 \bigoplus W^0 = V^1 \tag{18}$$

$$V^1 \bigoplus W^1 = V^0 \bigoplus W^0 \bigoplus W^1 = V^2 \tag{19}$$

This leads us to an elegant computational efficiency. Essentially, we can find the inner product integral at a finer degree of resolution V^{j+1} by taking the result in V^j and adding on the contribution from the "detail basis" W^j. This contribution is of course the sum of the products of the corresponding coefficients for W^j between the signal and the template. This brings us to a key point: *we can use the results of computations at the coarse level in computing results at finer levels.* For 1D, this means we need to do only half the work to refine our estimate. If we have an N coefficient representation in V^j, the representation in V^{j+1} will have $2N$ coefficients resulting in 2N computations for the inner product. However, we can simply add on the products of the N coefficients in W^j to get the same result! The gains are not quite as dramatic for 2D, since for an N by N coefficient representation in V^j, there are three times as many coefficients for the W^j [2]. As a result, we have to do $\frac{3}{4}$ of the computation that would have been necessary if we were starting from scratch.

There is a bit of a problem with this last insight. Though we can certainly refine our inner product integrals using the above method, it only makes sense if we already have the result at the lower level. Unfortunately, we can only have this result at dyadic points (multiples of two), as this is the only place the coefficients are defined for the lower level. In V^{j+1}, the shifts of $\phi(2^j t - k)$ correspond to shifts by 2. In V^j, we simply cannot find the "coarse" estimates at the odd points of V^{j+1}. There are two possible workarounds. The first is to shift one of the signals (most practically the template) by integers in V^R and then decompose each of these down to V^j. We can't shift by $\frac{1}{2^j}$ in V^j, so we use the shifted representation instead. Unfortunately, I found this technique somewhat impractical in practice, as the number of "shifted templates" that must be kept quickly grows large (32 version of the template for R=5). The other (more practical) workaround is to use the coefficients for V^j and W^j to rebuild the coefficents for V^{j+1} and then perform the sum of products with the resulting coefficients. In a practical implementation, it makes sense to keep around the a_{jk}'s as well as the b_{jk}'s so that this rebuilding does not have to performed at each step.

2.2 Comparison to Other Multiresolution Approaches

An appropriate question at this point is how this method fares against a more traditional multiresolution representation, such as the Gaussian/Laplacian pyramid image representation of Burt and Adelson [1]. In their technique, the original image forms the highest resolution representation. The next level is then a quarter of the size (half the size in each dimension). The values for this next level come from "a weighted average of the surrounding pixels" (this averaging is typically a five by five point approximation of a 2D Gaussian). The succeeding levels are defined analagously.

Why not simply compute this representation of the image and the template and compute the inner products at the various scales? The answer lies in the interpretation, or the lack thereof. Basically, we can make none of the statements of the section above for this representation. There is no precise meaning to an inner product of the "low-resolution" image and template. Parseval does not apply, so we have no precise idea how much of a signal's power is represented at a given level. Lastly (and perhaps most importantly in the minds of engineers), we cannot reuse computations from coarse levels in compute inner products at finer levels. The second application I will present will show why this can be an expensive drawback.

3 Getting the "Right" Coefficients from Discrete Time

The signals we will be dealing with are in discrete time, which brings up an important question: how are we to get the initial coefficients a_{Rk} from our discrete-time signal $x[n]$? If we are to talk about meaning, we cannot now turn around and simply take the sample values of the signal as the coefficients. We begin with the requirement that in the finest level of resolution using basis $\phi(2^R t - k)$, integer shifts of k correspond to shifts of one sample. We then have approximately as many coefficients as we have samples. The continuous time signal represented by these coefficients is

$$f(t) = \sum a_{Rk}\phi(2^R t - k) \tag{20}$$

As Strang points out in [6], a minimal requirement is for the samples of $f(t)$ above to equal the samples $x[n]$. This condition can be rewritten as:

$$a_R[n] * \phi[n] = x[n] \tag{21}$$

where $\phi[n]$ is the samples of $\phi(t)$ at the integers and $a_R[n] = a_{Rn}$. To find $a_R[n]$, we must convolve $x[n]$ with $\tilde{\phi}[n]$, the inverse of $\phi[n]$. $\phi[n]$ is an FIR sequence since $\phi(t)$ is compact support, and thus its inverse is always IIR. In addition, the stable form of the IIR filter may contain both causal and anti-causal components. For finite length signals, such an IIR filter is a feasible (but still unattractive) option to pursue. For infinite length signals and "real-time" applications, an anti-causal IIR filter is simply not an option.

As a result, I decided to find an FIR approximation $y[n]$ to the IIR inverse $\tilde{\phi}[n]$. Instead of simply truncating the IIR inverse, I made a least-squares formulation of the desired result. We want to construct a "pseudoinverse" that gets us as close as possible to the delta function:

$$y[n] * \phi[n] \approx \delta[n-k] \tag{22}$$

where the error criterion is the norm of our distance to this goal:

$$E = \|y[n] * \phi[n] - \delta[n-k]\| \tag{23}$$

We wish to optimize over $y[n]$ and k (a delay of k does not hurt us; we can simply shift the output back by k samples). An example setup with $\phi[n]$ of length three and $y[n]$ of length three and $k = 0$ is shown below:

$$\begin{bmatrix} \phi[0] & \\ \phi[1] & \phi[0] \\ \phi[2] & \phi[1] \\ & \phi[2] \end{bmatrix} \begin{bmatrix} y[0] \\ y[1] \\ y[2] \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{24}$$

We can rewrite this symbolically as

$$\mathbf{\Phi y} = \mathbf{d} \tag{25}$$

Because this will always be an overconstrained problem, we can only find an approximate solution. The solution minimizing the norm of equation 23 can be found by projecting the goal point into the subspace spanned by Φ [5]:

$$\hat{\mathbf{y}} = (\mathbf{\Phi\Phi^T})^{-1}\mathbf{\Phi^T d} \tag{26}$$

This gives us the optimal $y[n]$ for each k. We then optimize over k by finding this solution over all relevant k (a finite number, since $\phi[n]$ and $y[n]$ are both FIR) and taking the one with the minimum norm.

Some examples of this technique are shown below. The first example uses the orthogonal wavelet 'sym4', a non-minimum-phase factorization of D8. The filter coefficients $h_0[n]$, the continuous scaling function $\phi(t)$ and the wavelet $w(t)$, and the samples of the scaling function $\phi[n]$ are all shown in figure 4 below. Note that the other filters $h_1[n]$,$f_0[n]$, and $f_1[n]$ are derived from $h_0[n]$ using the "alternating flip" technique described on p.110 of [6].

For this filter, the best position for the delta was at 8 ($k = 8$). This resulted in an error norm of 0.0035 or 2.92e-4/sample. The resulting FIR approximate inverse $y[n]$ and the convolution result $\phi[n] * y[n]$ are shown in figure 5 below.

Another example is shown in figure 6 using the D3 wavelet (six coefficients) with the optimal k value. In this case, an eight sample FIR pseudoinverse was necessary to reduce the error to the same level as in the previous example.

The error norm in this case was 0.0013 or 8.84e-5/sample. From these examples, it is clear that a relatively short FIR filter can provide a very good approximation to the desired IIR response. The main reason for this is that the poles of $\frac{1}{\phi(z)}$ are often well behaved: though there are both causal and anti-causal components, they tend to drop quickly to zero, allowing for an effective FIR inverse.

Because the 2D wavelets we will be using are separable (i.e., the tensor products of 1D wavelets), we can use the outer products of our 1D pseudoinverse $y_{2D}[n][m] = \mathbf{y[n]y[n]^T}$ as the pseudoinverse of $\phi_{2D}[n][m] = \phi[\mathbf{n}]\phi[\mathbf{n}]^{\mathbf{T}}$, the samples of the tensor product of $\phi(t)$ with itself. An example showing $\phi_{2D}[n][m]$, the pseudoinverse $y_{2D}[n][m]$, and the convolved result (the approximation to the shifted 2D delta function) for the 'sym4' wavelet is shown below. The error norm in this case is 0.0049 or 3.40e-5/sample.

Again, we have an efficient, effective approximation to the desired IIR filter. As in the 1D case, we can now apply this filter $y_{2D}[n][m]$ to our image and use the resulting values as our coefficient values a_{Rk}. These coefficients can then be broken down to the coefficients

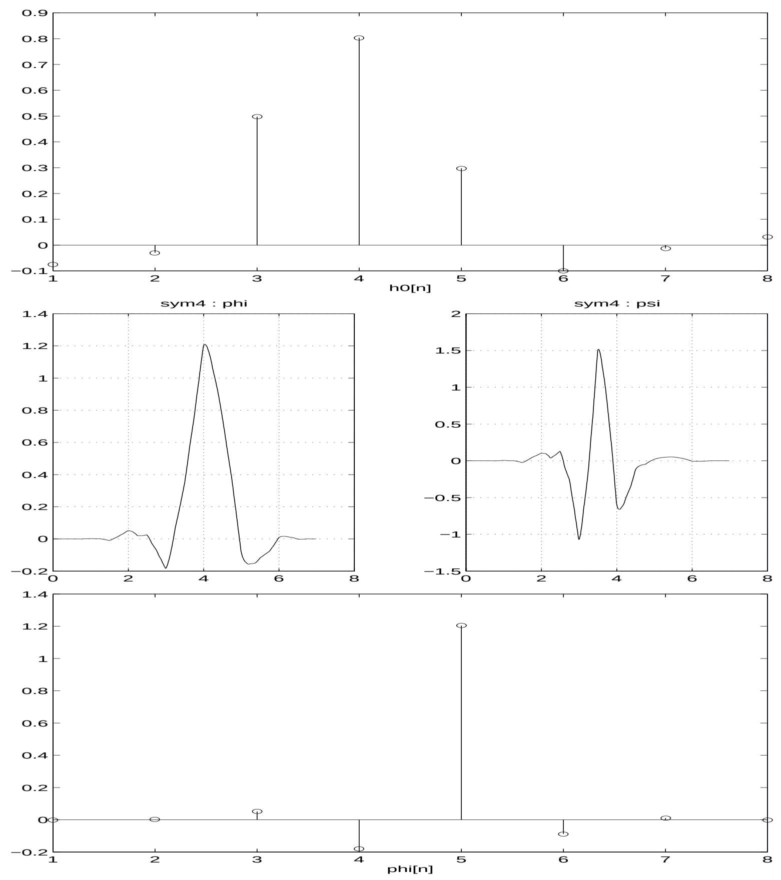

Figure 4: The filter coefficients $h_0[n]$, the continuous scaling function $\phi(t)$ and the wavelet $w(t)$, and the samples of the scaling function $\phi[n]$ for the 'sym4' wavelet

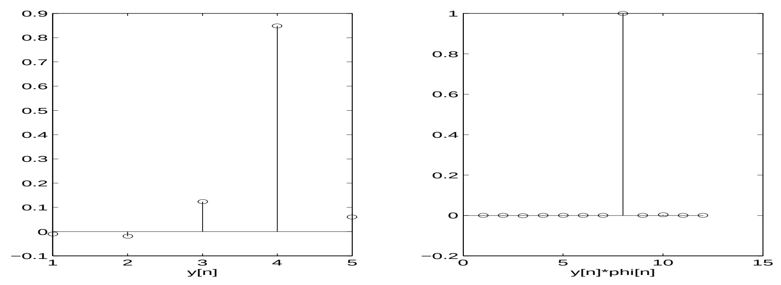

Figure 5: FIR approximate inverse $y[n]$ and the convolution result $\phi[n] * y[n]$

$a_{(R-1)k}$,$b_{(R-1)k}$, ...We can do this with a clear conscience, knowing that the samples of the reconstructed signal using these values will be (almost exactly) the original sample values.

4 Applications

Now that we have the correct coefficient values at each scale, we can move forward to the applications. I will demonstrate how the techniques described above can be applied to two

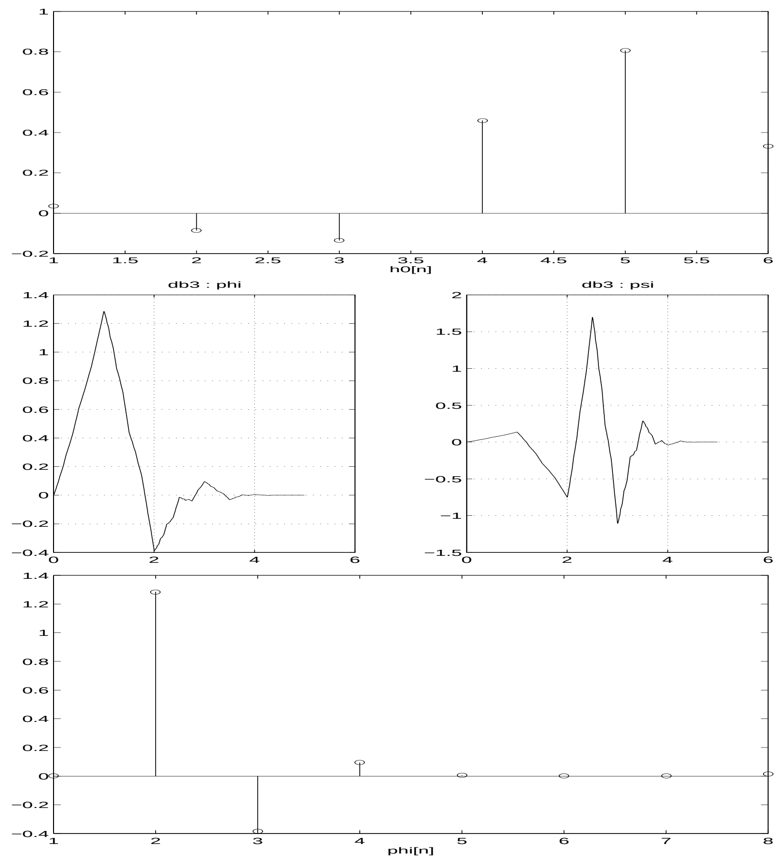

Figure 6: The filter coefficients $h_0[n]$, the continuous scaling function $\phi(t)$ and the wavelet $w(t)$, and the samples of the scaling function $\phi[n]$ for the 'd3' wavelet

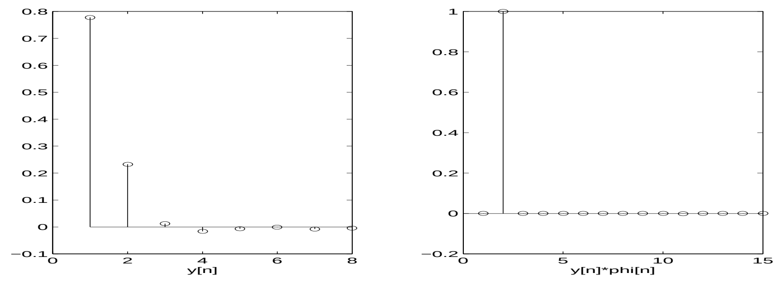

Figure 7: FIR approximate inverse $y[n]$ and the convolution result $\phi[n] * y[n]$

important problems in image processing/video coding/computer vision: motion estimation and multiple model matching. For all of the examples below, we will be using separable 2D wavelets built up from the 'sym4' wavelet described above. We apply these in 2D as described by Daubechies in Chapter 10 of [2].

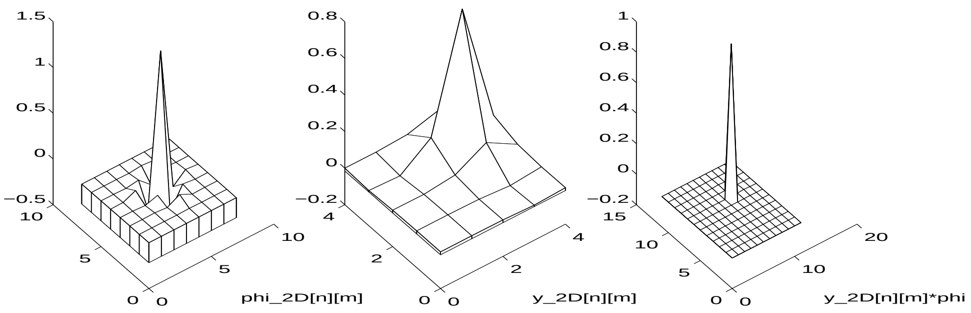

Figure 8: $\phi_{2D}[n][m]$, the pseudoinverse $y_{2D}[n][m]$, and the convolved result (the approximation to the shifted 2D delta function)

4.1 Motion Estimation

Motion estimation is one of the most fundamental problems in video coding. The basic issue is to take a piece of an image in a video sequence and see where it came from in the previous frame. This is exactly the template matching problem I described earlier: we have a template (the piece of the current frame) and we want to find it within another signal (the previous frame). Of course, this assumes that the piece of the image (or a large portion of it) will actually have appeared in the previous frame. For most frames in a video sequence, this is a reasonable assumption. Once we've found the optimal location of the template in the previous frame, we can describe the piece of the current frame by simply referring back to the previous frame. In practice, we code the error between the motion-predicted version of the current frame and the actual current frame. If the current frame is composed of simple motions of pieces of the previous frame, this error will be zero — thus the advantage for coding.

The case of motion estimation I will deal with is *egomotion* or self-motion. This corresponds to the motion of the camera itself rather than objects in the scene. If there is no motion in the scene, our objects are all at infinity, and our camera moves by pure translation, this scheme could perfectly explain the majority of the new frame (all but the new portion which is uncovered by the camera motion). Unfortunately, this is rarely the case. There is always motion in the scene, the camera is often rotating as well as translating, and objects are close enough to the camera for perspective effects to be significant. All of these problems occurr in the examples I will show below. However, as we will see, the *majority* of the scene is fairly well behaved, allowing the scheme to be effective in estimating the optimal alignment between the frames and providing a coding advantage.

The algorithm for this task closely follows the general multiresolution matching scheme that I have described. The target signal is the previous frame and the template is the entire current frame. The previous frame is zero-padded so that we can compute inner products with the template that overlap only part of the current frame. We start off by comparing the images at the lowest resolution (figure 9) at all possible alignments. Choosing the lowest resolution is an important step — if it is not low enough, the computations at this scale will still be quite expensive. If it is too low, though, the description will be too coarse — there may be many spurious matches. For this example, we began with 320 by 240 images and went down 4 levels, resulting in images about 1/256th of the original size. An image showing the values of the normalized correlation at all points is shown in figure 10. Because the coarse representations of the images are so small, the global search is quite cheap computationally.

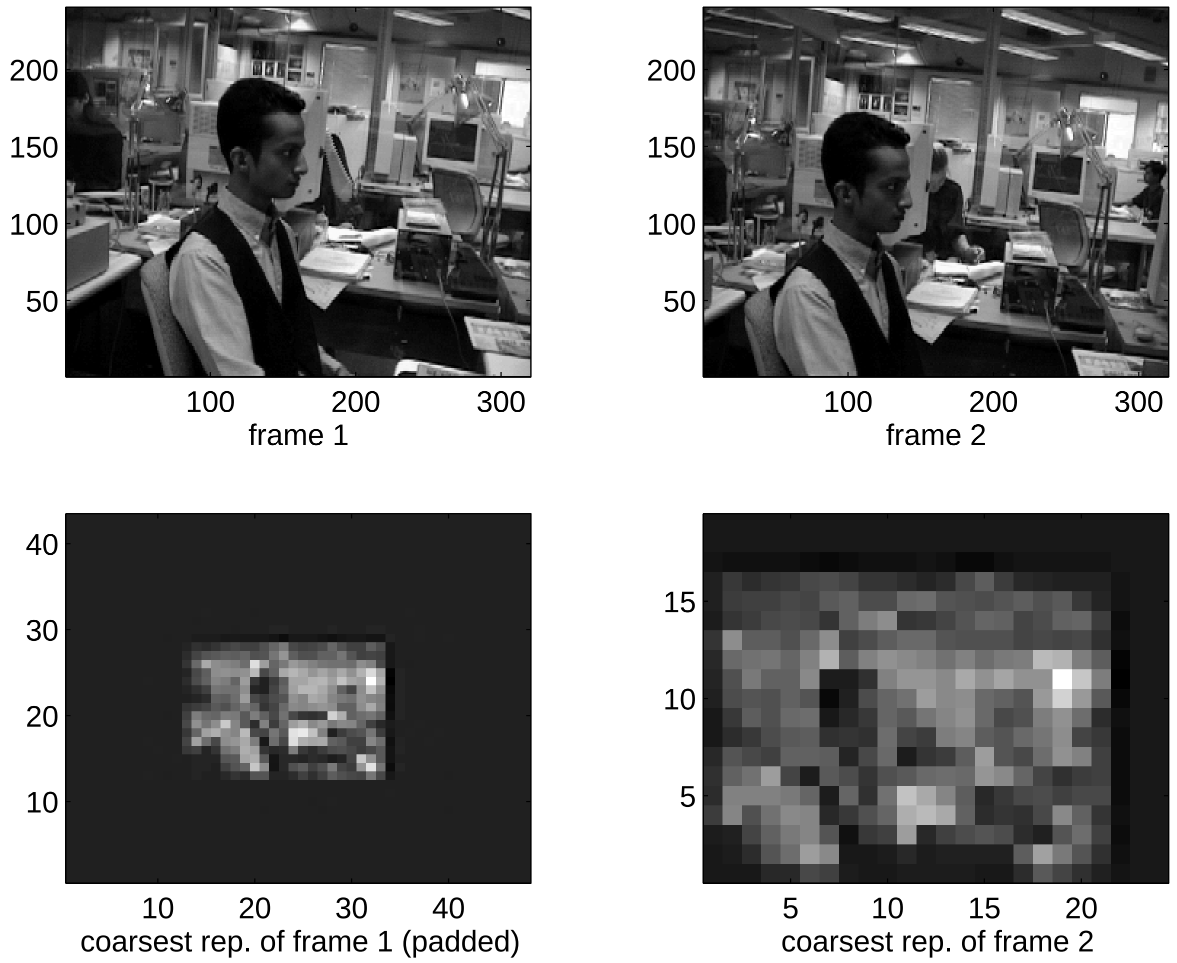

Figure 9: Original images (top) and coefficients at the coarsest scale.

We then take the location of the best match at this resolution and compute the corresponding location at the next higher resolution (i.e., we multiply each coordinate by 2). We then compute the normalized correlation at this higher resolution *but only in a 3x3 neighborhood* surrounding the previous match (see figure 11).

We can reduce the size of the search because we are fairly certain we are near the target from our lower-resolution computation. At this point, we simply need to refine the search. We choose a 3x3 region in order to capture the open interval between the dyadic points at the previous level. One of the nine points in this region (the center) is a dyadic point from the previous level, so we can reuse the computations in refining the estimate there. At the remaining eight points, the computations have to start from scratch. Figure 12 shows the estimated alignment for the two frames.

To show the benefits of this technique for coding, I have taken the current frame and

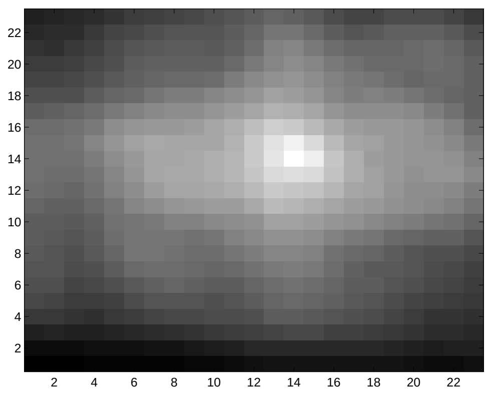
Figure 10: Normalized correlation values of the second frame against the padded first frame for the coarsest scale. Note the clear maximum signifying the best match.

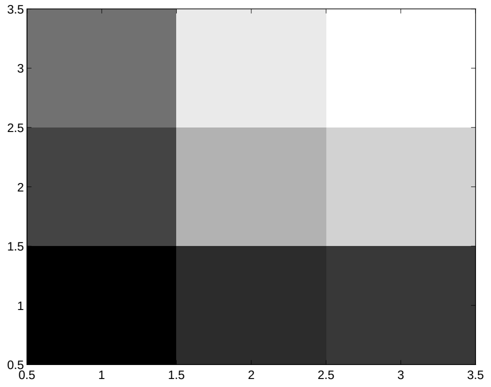
Figure 11: Normalized correlation at next finer scale.

subtracted the previous frame shifted by the estimated translation. The result can be seen in figure 13 below.

It is clear from from the result that there is more than pure translation going on. However,

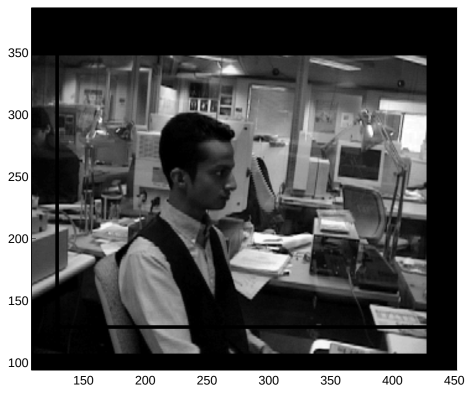
Figure 12: Estimated alignment of frames. The box shows where frame 2 lies in relation to frame 1.

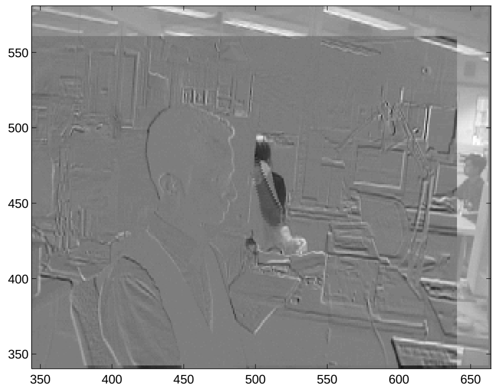

Figure 13: Result of subtracting overlapping portion of frame 1 from frame 2

the majority of the power (brighter region of the image) is concentrated in the new region uncovered by the camera motion and where there was motion in the scene (the phantom arm behind the monitor made a sudden move in this case). As a result, we expect the difference of the new frame and the prediction to be much cheaper to code than the entire new frame.

Another application of this motion estimation technique is shown below. Instead of using the estimation for coding, we wish to now fit several frames together to form a panoramic shot of the scene. This is known in image processing as "image mosaicing." The estimation technique is the same as before. Once we have the estimates, we composite the two frames using the offset. If our estimate is perfect, the resulting image should look seamless. An example of mosaicing four frames together is shown in figure 14 below. Again, the various non-ideal factors described above are at play, so the final image is not entirely seamless. Note that the background alignment is almost perfect while there are several mismatches in the foreground. This is because the effect of perspective (which is not accounted for in our pure translational model of motion) is much more pronounced in the foreground. A more sophisticated scheme would smooth the transition regions between frames. However, this simple example illustrates my motion estimation algorithm's applicability to this task.

4.2 Multiple Model Matching

Recognition is a very important problem in computer vision. Often in a recognition problem, we have several possible models that a candidate may match against, and we must test each one to see which fits the data best. Normalized correlation is again a powerful tool for determining the quality of the match. The way in which we are using it is different, though. Instead of moving around a template to find the best fit (the detection/estimation task), we are comparing multiple templates *to the same part of the signal* (the recognition/classification task).

In this case, the orthogonal wavelet inner product algorithm truly shines. Because we are only computing the correlations at one location in the test signal, we can reuse *all* of our computations in refining our estimates. To perform the recognition, we start off by computing the inner products of the test signal with all of the model templates at the coarsest scale. We then refine each estimate to the next finer scale, reusing the coarse scale's computations. We continue this process until one model clearly wins out over the others

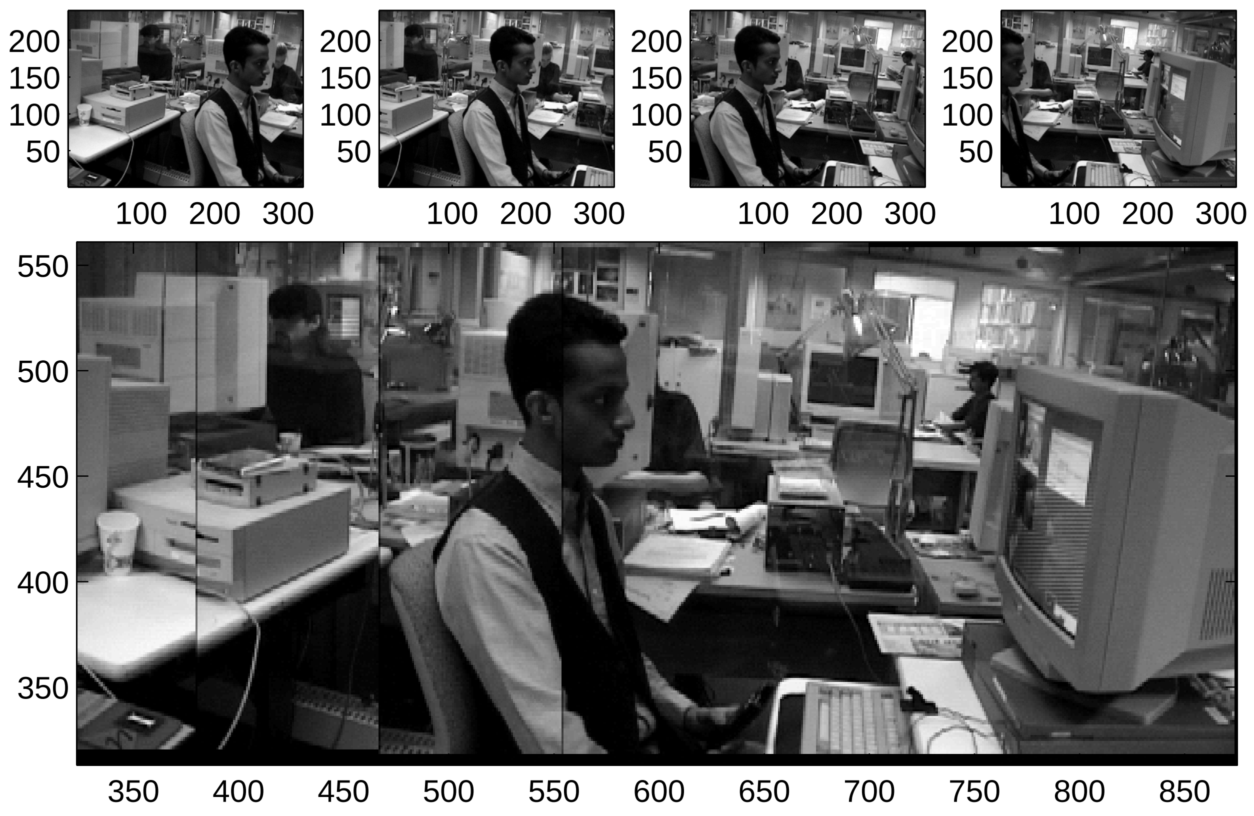

Figure 14: Five frames in a motion sequence and the mosaiced result.

(i.e., it has a significantly higher correlation score than any of the other models).

I will demonstrate this algorithm on a simple facial expression recognition task. I begin with a series of expression models (the training set). These are shown in figure 15 below. I then took two test expressions (the test set), shown in figure 16. It is important to note that these images were not in the training set.

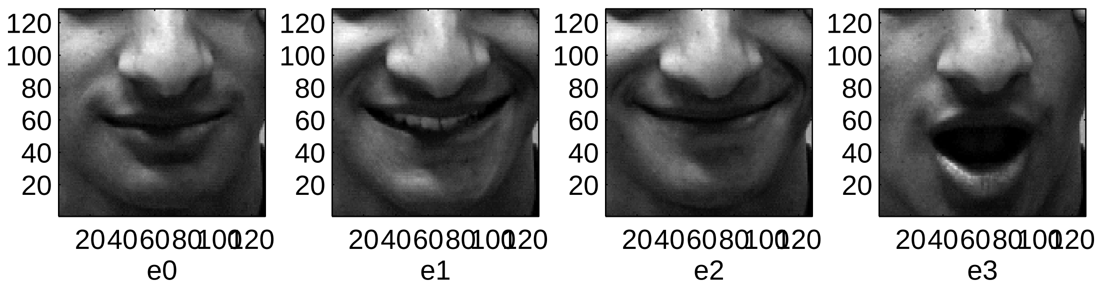

Figure 15: The model templates

Tables 1 and 2 show the correlation values at each level. Note that the basis becomes finer with increasing rows. In the first example, it is clear by the calculation in V^3 that e_2 is the victor. Similarly, in the second example, by V^3, e_3 has won out. In the examples shown above, we could stop the computation at this point. However, one can imagine cases where two templates might look very similar. In such a case, the correlation scores for these *two*

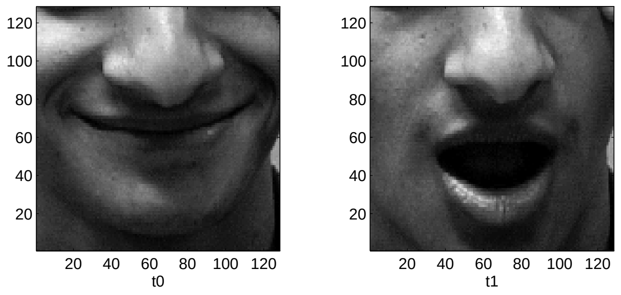

Figure 16: The test signals

Table 1: Reconstruction error per DOF (in normalized coordinates)

k (match in V^k)	e_0	e_1	e_2	e_3
0	0.9955	0.9950	0.9991	0.9789
1	0.9877	0.9815	0.9953	0.9611
2	0.9491	0.9425	0.9868	0.8214
3	0.8782	0.8834	0.9662	0.7193
4	0.7874	0.8158	0.9366	0.6059
5	0.7157	0.7565	0.9068	0.5144
6	0.6485	0.7019	0.8805	0.4367

templates will clearly be greater than that of all the other templates very quickly, but it might take another scale step to see which of the two better matches the target. We must thus refine all (and only) the best matches at every scale (where best implies significantly better performance than the competitors). Making use this insight and the reusability of coarse-scale computations, we can find the best model match very efficiently.

Table 2: Reconstruction error per DOF (in normalized coordinates)

k (match in V^k)	e_0	e_1	e_2	e_3
0	0.9955	0.9849	0.9867	0.9990
1	0.9875	0.9686	0.9766	0.9954
2	0.9231	0.8674	0.8653	0.9815
3	0.8603	0.7518	0.7778	0.9470
4	0.7972	0.6437	0.6845	0.9117
5	0.7461	0.5569	0.6082	0.8844
6	0.7025	0.4827	0.5422	0.8633

5 Conclusions and Future Directions

Through this study, I have shown that an orthogonal wavelet representation of a signal and a template can be used in a very efficient multiscale approach to template matching. I have shown how expensive signal processing operations (inner products in particular) can be performed in a meaningfully approximate way using a wavelet basis. In addition, I have demonstrated that finding the "correct" coefficients for the finest scale of resolution is not so difficult a task if we are willing to accept extremely small errors in the samples of the reconstruction. This makes a powerful case for wavelet representations in video and audio coding. If we have such representations coming from our I/O cards, we get the ability to use these multiscale algorithms for free! We wouldn't need to use Mallat's algorithm to compute the lower scale coefficients: the coder on the other end would have done it for us.

To extend this work, I would like to return to the beginning of our discussion. There I showed how a correlation could be rewritten as an inner product. I apologize if I am restating the obvious, but I was very pleased to recently realize that convolution with an *arbitrary* FIR filter can be rewritten as an inner product. We can thus apply the multiscale techniques presented in this paper to any FIR filter in 1D or 2D. This still leaves us with a question: which filters are the most interesting to pursue? Ideally, they are filters whose responses we would want to refine adaptively (as in the multiple model application above). This would give us the greatest computational win. They may also be filters whose approximate outputs are sufficient for a given task. If other such filters and their applications (in addition to the examples above) can be found, it will further strengthen the arguments for wavelets in signal coding.

I hope to pursue this question and to apply the results to my research in the modeling and tracking of nonrigid facial motions. I also hope to refine the thoughts, conjectures, and example applications above for use in my work, reusing (of course) the computations from this coarsest scale.

References

[1] Peter J. Burt and Edward H. Adelson. "The Laplacian Pyramid as a Compact Image Code". *IEEE Transactions on Communication*, pages 532–540, 1983.

[2] Ingrid Daubechies. *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA, 1992.

[3] Alan V. Oppenheim and Alan S. Willsky. *Signals and Systems.* Prentice Hall, Eaglewood Cliffs, NJ, 1983.

[4] Lawrence Rabiner and Bing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[5] Gilbert Strang. *Linear Algebra and Its Applications (3rd Edition).* Harcourt Brace Jovanovich, San Diego, CA, 1986.

[6] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Wellesley, MA, 1996.