Template Learning from Atomic Representations: A Wavelet-based Approach to Pattern Analysis

Clayton Scott, Robert Nowak

The authors are with the Department of Electrical and Computer Engineering, MS 366, Rice University, Houston, TX 77005 (e-mail: cscott@rice.edu; nowak@rice.edu).

Abstract

Despite the success of wavelet decompositions in other areas of statistical signal and image processing, current wavelet-based image models are inadequate for modeling patterns in images, due to the presence of unknown transformations (e.g., translation, rotation, location of lighting source) inherent in most pattern observations. In this paper we introduce a hierarchical wavelet-based framework for modeling patterns in digital images. This framework takes advantage of the efficient image representations afforded by wavelets, while accounting for unknown pattern transformations. Given a trained model, we can use this framework to synthesize pattern observations. If the model parameters are unknown, we can infer them from labeled training data using TEMPLAR (Template Learning from Atomic Representations), a novel template learning algorithm with linear complexity. TEMPLAR employs minimum description length (MDL) complexity regularization to learn a template with a sparse representation in the wavelet domain. We discuss several applications, including template learning, pattern classification, and image registration.

Keywords

Wavelets, pattern analysis, MDL, supervised learning

I. Introduction

Wavelet decompositions often provide very parsimonious image representations, and this feature has been exploited to devise powerful compression, denoising and estimation methods [1]. Although wavelets provide sparse representations for many real world images, it is difficult to develop a wavelet-based statistical model for a given pattern based on observations of the pattern. This is because in many (perhaps most) applications, the pattern of interest undergoes an unknown or random transformation during data acquisition (e.g. variations in illumination, orientation, translation, and perspective). Modeling the wavelet expansion of such transformed data leads to distorted components, or even components that model the transformations instead of the structure of the underlying object or pattern.

The objective of this work is to develop a wavelet-based framework for modeling pattern observations that have undergone random transformations in the observation process. We introduce TEMPLAR (Template Learning from Atomic Representations), an algorithm that combines the efficiency of wavelet representations with minimum description length (MDL) complexity-regularization to automatically learn a low-dimensional pattern template from noisy, randomly transformed observations [2]. The resulting template may be applied to classification or pattern synthesis. The learning algorithm itself may be applied to image registration and denoising from multiple, unaligned observations. Our approach is similar in spirit to that of Frey and Jojic [3], although in that work, the dimension of the template is fixed in advance and the basis vectors are adaptive. We work with a fixed basis, and allow the dimension of the template to vary.

The essential ingredients underlying the learning algorithm are: (i) an orthogonal basis that is well-matched to the pattern structure (i.e., a basis that provides a parsimonious representation of the pattern template); and (ii) MDL-based complexity penalization to promote a low-dimensional template by allowing only the *significant* basis coefficients to contribute to template. A low-dimensional template is advantageous because it facilitates the pattern matching process (in both template learning can classification) by giving more weight to significant template coefficients, which model the defining structure of the pattern, and less weight to surrounding clutter.

In this paper, we emphasize discrete wavelet bases since they tend to provide very sparse representations for a wide variety of spatial patterns, especially those patterns characterized by their edges. Thus, wavelet-domain representations will lead to lower-dimensional templates than, for example, pixel-domain representations. Furthermore, wavelet bases admit a reasonable and tractable statistical model for describing significant and insignificant coefficients.

In Section 2 we introduce our hierarchical framework and statistical model for describing patterns. In Section 3 we present TEMPLAR, an iterative algorithm for learning a pattern template from training observations. In Sections 4 we illustrate applications of TEMPLAR to a variety of problems. Section 5 concludes with a summary and discussion.

II. Hierarchical Framework and Statistical Model

When a pattern is observed in an image, it can appear at any number of locations, orientations, scales, etc., in the image, depending on the spatial relationship between the image forming device and the pattern. Further uncertainty in pattern observations can be caused by lighting sources, background clutter, observation noise, and deformations of the pattern itself (if the pattern is not rigid, like a human face). Figure 2 (a) shows several observations of an airplane with variable location, orientation, lighting conditions, and background. We model these uncertainties in pattern observations with a hierarchical framework, based on the notion of deformable templates [4]. For our purposes, a template is a noise-free observation of a pattern that can be transformed into an arbitrary observation of the same pattern by applying a deformation to the template, and adding observation noise. Examples of deformations include global operations such as translations and rotations, as well as transformations with localized support, to represent local perturbations of the pattern.

Our hierarchical framework attempts to de-couple three different aspects of the pattern observations: the observation noise, the unknown transformations, and the pattern itself. Thus, a pattern observation is synthesized by taking a realization of the template, (which we treat as a random vector), applying a randomly selected transformation, and adding a realization of the observation noise. The novelty of our approach is that we model the template in the wavelet domain. As we show in Section 3, this allows us to develop a template learning algorithm that takes advantage of the properties of the wavelet transform. We now describe in more detail the individual stages of the hierarchical framework.

We model template deformations with a finite set of linear transformations $\mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_L$. Restricting to linear transformations still allows for a wide range of transformations, such as translation, rotation, scaling, and shearing. Furthermore, the aforementioned transforms all have sparse matrix representations, i.e., if they are operating on images with N pixels, their matrix representations have $O(N)$ non-zero entries. This allows for linear-time computation and matrix inversion. In addition to linearity, we will also assume the transformation operators are invertible and, moreover, orthogonal. Of the transformations listed above, only translation is truly orthogonal when implemented on a computer (see appendix A), but many other transformations are "nearly" orthogonal in the following sense: If $\mathbf{\Gamma}_\ell$ is a matrix representation of the operator, then $\mathbf{\Gamma}_\ell \mathbf{\Gamma}_\ell^T$ differs from the identity matrix by $O(N)$ entries. The error that results from this assumption is negligible in the examples presented in this paper. (When scaling or shearing to a high degree, the non-orthogonality of the transformation can no longer be ignored. Translation and rotation are the most suitable transformations for this framework.)

These transformations are fixed before any analysis is done, and are chosen to cover (or at least reasonably approximate) the suspected range of possible transformations. This list may be reduced if we assume each training image has been crudely preprocessed to compensate for very gross transformations such as large translations. To limit the complexity of our model, we do not directly account for local pattern deformations. However, we are still able to capture variability in the pattern itself by treating the template as a random vector. In particular, we model the statistics of the template in the wavelet domain.

Let the random vector $\mathbf{W} = (W_1, \ldots, W_N)^T$ denote the wavelet coefficients (as computed by some orthogonal wavelet transform $\mathcal{W}$) of the pattern template, where N is the total number of pixels in a pattern observation. In real-world images, we observe two "flavors" of wavelet coefficients: large coefficients, corresponding to wavelets that match edges in the image, and small coefficients, corresponding to smooth regions of the image or noise. We refer to the former as *significant* coefficients, and the latter as *insignificant* coefficients. We assign to each W_i a state variable s_i taking on the values 0 or 1, with $s_i = 1$ indicating a significant coefficient, and $s_i = 0$ indicating an insignificant coefficient. This dichotomy is captured as follows: we model the marginal pdf of the insignificant wavelet coefficients as $f_{W_i|s_i}(w_i|s_i = 0) = \mathcal{N}(w_i|\mu_{i,0}, \sigma_{i,0}^2)$; and, we model the marginal pdf of the significant wavelet coefficients as $f_{W_i|s_i}(w_i|s_i = 1) = \mathcal{N}(w_i|\mu_{i,1}, \sigma_{i,1}^2)$. Here, $\mathcal{N}(x|\mu, \sigma^2)$ denotes a Gaussian density with mean μ and variance σ^2, as a function of the variable x. For $m = 0, 1$, define $\boldsymbol{\mu}_m \equiv (\mu_{1,m}, \ldots, \mu_{N,m})^T$, and define $\boldsymbol{\Sigma}_m = \mathrm{diag}(\sigma_{1,m}^2, \ldots, \sigma_{N,m}^2)$, where $\mathrm{diag}(a_1, \ldots, a_n)$ denotes the $n \times n$ diagonal matrix with diagonal entries $a_1, \ldots, a_n$. To simplify our model, we assume $\mu_{i,0} = 0$ and $\sigma_{i,0}^2 = \sigma_0^2$ for each i. This reflects our belief that the insignificant wavelet coefficients model smooth regions or noise. Thus, significant coefficients have unconstrained mean and variance, while significant coefficients are constrained to have mean zero and common variance σ_0^2. We collect the means and variances of the significant and insignificant wavelet coefficients together in the parameter vector $\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\}$. Because of our simplifying assumptions, we may also write $\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \sigma_0^2\}$.

We assume that the wavelet coefficients are statistically independent. (This is a reasonable assumption because the wavelet basis functions are orthogonal and spatially localized. There are in fact dependencies between wavelet coefficients such as spatial clustering and persistence across scale [5] which we do not treat in this paper.) Therefore the joint distribution of the wavelet coefficients of the pattern template is specified by $\boldsymbol{\theta}$, together with a configuration $\mathbf{s} = (s_1, \ldots, s_N)^T$ of state variables. To synthesize a pattern observation from this template, we first generate a realization $\mathbf{w}$ of wavelet coefficients according to the joint pdf $f_{\mathbf{W}|\mathbf{s}}(\mathbf{w}|\mathbf{s}) = \prod f_{W_i|s_i}(w_i|s_i)$. Next, we obtain a realization $\mathbf{z} = \mathcal{W}^{-1}\mathbf{w}$ of the template by transforming $\mathbf{w}$ into the spatial domain. A transformation $\boldsymbol{\Gamma}_\ell$ is then selected from the list $\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_L$ according to some prior distribution (assumed uniformed unless otherwise indicated), and applied to the spatial template to form a transformed pattern $\mathbf{y} = \boldsymbol{\Gamma}_\ell \mathbf{z}$. Finally, the observed image $\mathbf{x}$ is created by corrupting $\mathbf{y}$ with additive observation noise, which we model as zero-mean iid Gaussian with variance σ^2_{obs}. The conditional density of the observed image is given by

$$p(\mathbf{x}\,|\,\boldsymbol{\theta},\mathbf{s},\ell) \;=\; \mathcal{N}(\,\mathbf{x}\,|\,\mathbf{\Gamma}_\ell \mathcal{W}^T\boldsymbol{\mu}\,,\,\mathbf{\Gamma}_\ell\mathcal{W}^T\mathbf{\Sigma}\mathcal{W}\mathbf{\Gamma}_\ell^T + \sigma^2_{\text{obs}}\mathbf{I}\,),$$

where $\boldsymbol{\mu} \equiv (\mu_{1,s_1},\ldots,\mu_{N,s_N})^T$ and $\mathbf{\Sigma} \equiv \text{diag}(\sigma^2_{1,s_1},\ldots,\sigma^2_{N,s_N})$. Since $\mathbf{\Gamma}_\ell$ and $\mathcal{W}$ are orthogonal operators, we have $\sigma^2_{\text{obs}}\mathbf{I} = \mathbf{\Gamma}_\ell\mathcal{W}^T(\sigma^2_{\text{obs}}\mathbf{I})\mathcal{W}\mathbf{\Gamma}_\ell^T$, and therefore we may write the covariance matrix of the above density as $\mathbf{\Gamma}_\ell\mathcal{W}^T(\mathbf{\Sigma}+\sigma^2_{\text{obs}}\mathbf{I})\mathcal{W}\mathbf{\Gamma}_\ell^T$. In light of this fact, we see that it is not necessary to model the observation noise separate from the wavelet coefficients. Henceforth we assume any observation noise is captured by the wavelet domain statistical model, and that the density on the space of observed images is given by

$$p(\mathbf{x}\,|\,\boldsymbol{\theta},\mathbf{s},\ell) \;=\; \mathcal{N}(\,\mathbf{x}\,|\,\mathbf{\Gamma}_\ell \mathcal{W}^T\boldsymbol{\mu}\,,\,\mathbf{\Gamma}_\ell\mathcal{W}^T\mathbf{\Sigma}\mathcal{W}\mathbf{\Gamma}_\ell^T\,). \tag{1}$$

III. Complexity Regularized Template Learning Via TEMPLAR

In this section we introduce a method for learning template parameters $\boldsymbol{\theta}$ and $\mathbf{s}$ from training data, i.e. a collection $\mathbf{X} = (\mathbf{x}^1,\ldots,\mathbf{x}^T)$ of statistically independent observations of the same underlying pattern, as in Figure 2 (a). In the process, we also learn the indices $\mathcal{L} = (\ell_1,\ldots,\ell_T)$ of the transformations giving rise to each observation. Our approach is to perform penalized maximum likelihood (PML) estimation of the parameters $\boldsymbol{\theta},\mathbf{s}$, and $\mathcal{L}$. In particular, we would like to maximize the objective function

$$F(\boldsymbol{\theta},\mathbf{s},\mathcal{L}) = \log p(\mathbf{X}|\boldsymbol{\theta},\mathbf{s},\mathcal{L}) + c(\mathbf{s})$$

where $c(\mathbf{s})$ is a complexity penalty term that decreases as the number of significant coefficients increases. We include $c(\mathbf{s})$ to balance the trade-off between fitting the data and model complexity, and thus to promote the learning of a template with a sparse wavelet representation.

We select a minimum description length (MDL) based penalty term of the form

$$c(\mathbf{s}) = -2k\log(N),$$

where $k = \sum s_i$ is the total number of significant coefficients [6], [7], [8]. The negative of this quantity is interpreted as the number of bits required to encode the location and values of the means and variances of the significant coefficients. In detail, following a derivation similar to that of Saito [6], each significant coefficient requires approximately $\log_2(N)$ bits to encode its location, $\frac{1}{2}\log_2(N)$ bits to encode its mean, and $\frac{1}{2}\log_2(N)$ bits to encode its variance. Since the negative log-likelihood is the Shannon code length required to encode the data, we see that maximizing F is equivalent to minimizing the total code (description) length required to encode both data and model parameters.

Joint maximization of F over all three parameters is intractable. Our approach is to find a (possibly local) solution to this optimization problem with TEMPLAR (Template Learning from Atomic Representations), an iterative alternating-maximization algorithm. (The name of the algorithm reflects the fact that not only the wavelet transform, but any orthogonal atomic representation that sparsely decomposes the data, could be used.) The premise is to maximize over one parameter at a time, while holding the other two fixed. We first initialize estimates $\mathbf{s}_0$ and $\mathcal{L}_0$ of the states and hidden transformations, by stipulating, for example, that all scaling coefficients are significant, all wavelet coefficients are insignificant, and all transformations are the identity transformation. TEMPLAR then proceeds according to

$$\begin{aligned} \boldsymbol{\theta}_j &= \arg\max_{\boldsymbol{\theta}} \ F(\boldsymbol{\theta}, \mathbf{s}_{j-1}, \mathcal{L}_{j-1}) & (2) \\ \mathbf{s}_j &= \arg\max_{\mathbf{s}} \ F(\boldsymbol{\theta}_j, \mathbf{s}, \mathcal{L}_{j-1}) & (3) \\ \mathcal{L}_j &= \arg\max_{\mathcal{L}} \ F(\boldsymbol{\theta}_j, \mathbf{s}_j, \mathcal{L}) & (4) \end{aligned}$$

Since each step involves a maximization, this produces a non-decreasing sequence of penalized log-likelihood values. The process continues until $F(\boldsymbol{\theta}_j, \mathbf{s}_j, \mathcal{L}_j) = F(\boldsymbol{\theta}_{j-1}, \mathbf{s}_{j-1}, \mathcal{L}_{j-1})$, which is guaranteed to happen in a finite number of iterations, by Theorem 1 below. When the algorithm stops, the current values of $\boldsymbol{\theta}_j$ and $\mathbf{s}_j$ are estimates for the template, and the current value of $\mathcal{L}_j$ contains the best estimate of the transformations that generated the training images from the learned template.

Each step of TEMPLAR is simple and efficient. One iteration requires $O(NLT)$ operations, where N is the total number of pixels in a pattern observation, L is the number of transformations, and T is the number of training images. The use of complexity-regularization combined with the inherent edge detection capabilities of wavelets produces a low-dimensional template that captures the defining features of the pattern of interest, but does not represent background clutter or noise. The details of the algorithm are given in Appendix B.

A. Convergence Analysis

We begin with some theoretical results, before shifting to a more intuitive discussion of TEMPLAR's convergence. For notational convenience, we define $F_j \equiv F(\boldsymbol{\theta}_j, \mathbf{s}_j, \mathcal{L}_j)$. Also, let

$\mathcal{T}_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ be the template corresponding to the j-th iteration, where $\boldsymbol{\mu}_j = (\mu_{1,s_1}, \ldots, \mu_{N,s_N})^T$ and $\boldsymbol{\Sigma}_j = \mathrm{diag}(\sigma^2_{1,s_1}, \ldots, \sigma^2_{N,s_N})$, and μ_{i,s_i} and σ^2_{i,s_i} are determined by the current estimates $\boldsymbol{\theta}_j$ and $\mathbf{s}_j$. As noted earlier, the sequence $\{F_j\}$ is non-decreasing. Note that the particular sequence, as well as the terminal value of $(\boldsymbol{\theta}_j, \mathbf{s}_j, \mathcal{L}_j)$, depends on the initial settings $\mathbf{s}_0$ and $\mathcal{L}_0$.

Theorem 1: The sequence of penalized log-likelihood values $\{F_j\}$ generated by TEMPLAR (according to Equations (2-4)) converges and reaches its limit in a finite number of iterations.

Proof: Recall that the observations are N-dimensional, there are L transformations, and T training images. Observe that there are 2^N possible configurations of the state vector $\mathbf{s}$, and L^T possible configurations of $\mathcal{L}$. Furthermore, each $\boldsymbol{\theta}_j$ is determined by $\mathbf{X}, \mathbf{s}_{j-1}$, and $\mathcal{L}_{j-1}$, and $\mathbf{s}_j$ is determined by $\mathbf{X}$ and $\boldsymbol{\theta}_j$. Thus, there are $2^N L^T$ different possibilities for $(\boldsymbol{\theta}_j, \mathbf{s}_j, \mathcal{L}_j)$, and therefore there are at most $2^N L^T$ different values for F_j. A non-decreasing sequence that takes on a finite number of values must converge and reach its limit after finitely many terms in the sequence. $\blacksquare$

It follows that TEMPLAR terminates in a finite number of iterations. However, Theorem 1 does not tell us about the convergence of the sequence of template estimates $\mathcal{T}_j$, which is what we are ultimately interested in. The following result allows us to address this issue:

Theorem 2: With probability one, if $F_j = F_{j-1}$, then $\mathcal{T}_j = \mathcal{T}_{j-1}$.

The proof is found in Appendix C. An underlying assumption of the proof is that the training images $\mathbf{x}^1, \ldots, \mathbf{x}^T$ are drawn from a Gaussian distribution. Thus, when we say "with probability one," we mean that the set of training images $\mathbf{x}^1, \ldots, \mathbf{x}^T$ such that $F_j = F_{j-1}$ and $\mathcal{T}_j \neq \mathcal{T}_{j-1}$ (for some j) has measure zero with respect to the Gaussian probability measure.

Theorem 3: The final estimate of the template and hidden transformations produced by TEMPLAR is the limit of the sequence $\{\mathcal{T}_j, \mathcal{L}_j\}$, with probability one.

To see this, suppose the algorithm stops after k iterations, i.e. $F_k = F_{k-1}$. By Theorem 2, $\mathcal{T}_k = \mathcal{T}_{k-1}$, with probability one. Since $\mathcal{L}_j$ depends only on $\mathcal{T}_j$ and the data, we also have $\mathcal{L}_k = \mathcal{L}_{k-1}$ with probability one. Since each term in the sequence $\{\mathcal{T}_j, \mathcal{L}_j\}$ depends only on the previous term and the data (which is fixed), once two consecutive terms are equal, all terms are equal from that point on. This is what we wanted to show. This result eliminates the possibility of "cycling," i.e. having the algorithm endlessly alternate between two or more different configurations of $\mathcal{T}$ and $\mathcal{L}$ that yield the same penalized log-likelihood value.

It is difficult, if not impossible, to prove either global or local optimality of the final estimate of the parameters $\boldsymbol{\theta}, \mathbf{s}$, and $\mathcal{L}$. The final estimate produced by TEMPLAR does satisfy a weaker criterion, however; it is a *partial optimal solution*. The parameters $\boldsymbol{\theta}^*, \mathbf{s}^*, \mathcal{L}^*$ are a partial optimal solution to the optimization problem at hand if they satisfy:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \; F(\boldsymbol{\theta}, \mathbf{s}^*, \mathcal{L}^*) \\ \mathbf{s}^* &= \arg\max_{\mathbf{s}} \; F(\boldsymbol{\theta}^*, \mathbf{s}, \mathcal{L}^*) \\ \mathcal{L}^* &= \arg\max_{\mathcal{L}} \; F(\boldsymbol{\theta}^*, \mathbf{s}^*, \mathcal{L}) \end{aligned}$$

This criterion pairs naturally with alternating-maximization algorithms, and is employed frequently in numerical optimization problems of this sort [9], [10]. That TEMPLAR produces a partial optimal solution is an immediate consequence of Theorem 2.

As the final stage of our analysis, we give an intuitive explanation for why TEMPLAR works. We have already discussed how wavelets decompose the pattern into a few significant coefficients and several insignificant coefficients, and how the MDL principle can be applied to form a sparse template by selecting only the significant coefficients. But how does a sparse template aid in the learning process?

The answer to the preceding question is that a sparse template improves the pattern matching (i.e., registration) procedure in Equation (4). Presumably, the significant coefficients model the defining features of the pattern. Thus, when matching/registering a training image to the template, more weight is given to the defining features of the pattern than to non-essential features such as clutter, because the significant coefficients tend to have smaller variances than the insignificant coefficients.

Furthermore, if a training image is matched to the template in one iteration, those significant coefficients will be "stronger" for the next iteration. Hence, other training images will become more likely to align themselves with the template. In other words, once a few training images share a common alignment, other training images are likely to gravitate toward that same alignment. We refer to this phenomenon as "jumping on the bandwagon." This feature of TEMPLAR enables it to converge quickly in most cases (fewer than T iterations).

In summary, TEMPLAR generally converges quickly to a template with a sparse wavelet representation. These properties are illustrated in the synthetic example below, and in the applications given in Section IV. In nearly all of our experimentation with TEMPLAR, it converges in 4 to

8 iterations. Of course, the probability of TEMPLAR producing what we would call a good template decreases as the observation noise increases, as the range of transformations considered decreases, or as the variability of the pattern itself increases.

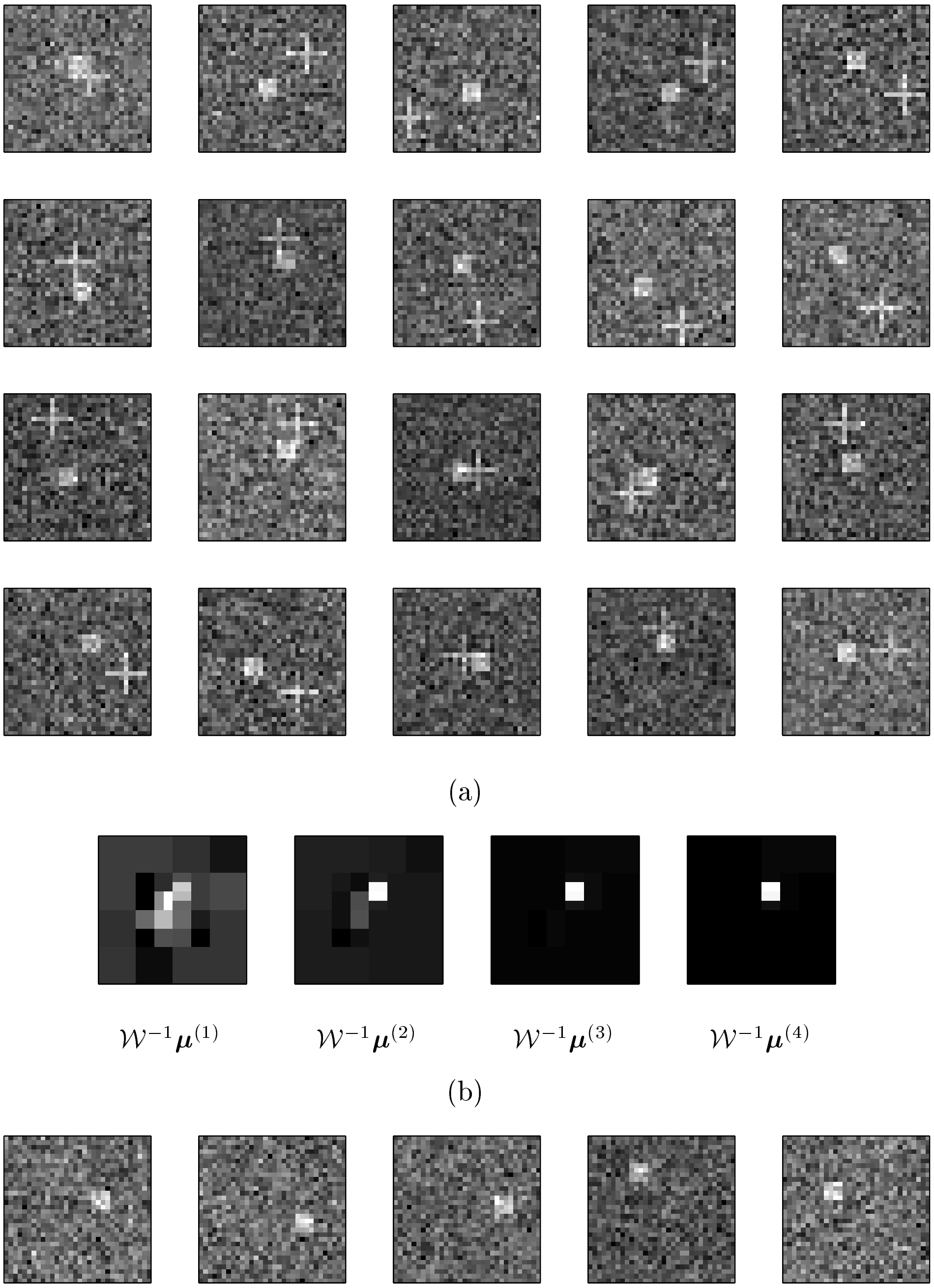
B. A Synthetic Example

Figure 1 (a) shows twenty training images. Each image is 32×32 pixels, and was formed by adding iid Gaussian noise (with variance $\sigma^2_{\mathrm{obs}} = 0.1$) to a particular binary (0 or 1 valued) image consisting of a 4×4 square and a cross. The square occurs within a ± 4 pixel translation (horizontally and/or vertically) of the center of the image and is considered the pattern of interest. The cross occurs at more widely varying translations, independent of the square's location, and is considered clutter.

We apply TEMPLAR to this data using the Haar wavelet transform, and using transformations $\mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_L$ that cover translations of up to ± 8 pixels horizontally or vertically, for a total of $(17)^2 = 289$ transformations. The algorithm converges after four iterations for this particular realization of the training data. Figure 1 (b) shows $\mathcal{W}^{-1}\boldsymbol{\mu}^{(j)}$, the template mean transformed into the spatial domain, for each iterarion.

The final template has 10 significant coefficients out of 1024 total coefficients. If we consider a binary image with a 4×4 square of 1s and a background of 0s, then the Haar wavelet transform of such an image can have from 10 to 56 nonzero wavelet transform coefficients, depending on the location of the square in the image. Thus the sparsest representation of such a square would have 10 nonzero coefficients. The template found by TEMPLAR has the sparsest possible representation for the square, and the 10 nonzero coefficients needed to reconstruct the square at that location are the significant coefficients of the template.

This example illustrates how TEMPLAR converges to a sparse template via jumping on the bandwagon (defined in the previous subsection). In the first iteration $\mathcal{W}^{-1}\boldsymbol{\mu}_1^{(1)}$ is the average of the 20 training images, because the transformations are initialized to be the identity. After it is computed, some wavelet coefficients are "turned on," meaning their states become significant. When the transformations are updated, 12 of the training patterns are aligned at a specific location. At that location, the binary images of the square would have 10 nonzero coefficients, the minimum possible. This is because the square pattern is best represented at this location, given the limited number of significant coefficients. Six more training patterns match this alignment in the second iteration, and two in the third. Upon repeating this experiment 100 times, the



(a)

$\mathcal{W}^{-1}\boldsymbol{\mu}^{(1)}$ $\mathcal{W}^{-1}\boldsymbol{\mu}^{(2)}$ $\mathcal{W}^{-1}\boldsymbol{\mu}^{(3)}$ $\mathcal{W}^{-1}\boldsymbol{\mu}^{(4)}$

(b)

(c)

Fig. 1. Synthetic example: (a) Training data: randomly translated squares, with randomly translated crosses in the background, plus iid Gaussian noise. (b) Sequence of template means, transformed into the spatial domain, for each iteration of the learning algorithm. (c) Observations synthesized from the trained model.

average dimension of the template was 10.90, and the average number of iterations needed to reach convergence was 4.65.

We also observe that the final template does not represent any of the clutter present in the training images, Although each training image contains a "cross" pattern in the background, the template does not. This is because the crosses' spatial relationship to the square varies from image to image. It is conceivable that TEMPLAR would try to align the crosses instead of the squares, but this does not happen because the range of transformation used by TEMPLAR does not cover the range expressed by the cross pattern, and because the square is a "stronger" pattern, in that it has more energy.

The common variance of the insignificant coefficients in this example is $\sigma_0^2 = 0.113$, which is slightly larger than the observation noise variance of $\sigma^2_{\text{obs}} = 0.1$. This discrepancy is to be expected, since background clutter (i.e. the crosses) also contributes to the variance of the insignificant coefficients. Figure 1 (c) shows five pattern observations synthesized from the trained model. The synthesized images closely resemble the training data, except that the clutter (crosses) is no longer present.

C. Variations of TEMPLAR

Some variations on the basic learning algorithm presented earlier in this section are possible. First, we discuss the issue of initialization. The algorithm requires that two of the three parameters $\boldsymbol{\theta}, \mathbf{s}$, and $\mathcal{L}$ be initialized. Our first proposal was to initialize the states (scaling coefficients are significant, wavelet coefficients are insignificant) and transformations (all equal to the identity map). This initialization reflects a prior belief that no particular training image looks more like a good template than any other. But suppose that one specific training image $\mathbf{x}^t$ does have this property (as judged by the user) of being a more typical realization of the pattern template than any of the other training images. It is possible to initialize $\boldsymbol{\theta}$ and $\mathbf{s}$ to reflect his belief. For example, we may set $s_i = 1$ for each i, $\boldsymbol{\mu} = \mathcal{W}\mathbf{x}^t$, and $\boldsymbol{\Sigma} = \mathbf{I}_{N \times N}$. Then, when $\mathcal{L}$ is computed for the first time, the effect is that all training images are registered with $\mathbf{x}^t$ according to a squared error loss function. Now, when $\boldsymbol{\theta}$ is updated on the next iteration, it is the original initialization scheme, except that the transformations have been adjusted to favor the alignment of $\mathbf{x}^t$.

The optimization in (4) is the most time consuming of the three steps. There are two possibilites for improving this situation. The first is to modify the transformation search space $\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_L$ with each iteration. The first iteration would perform a coarse search over a large range of the search space, and subsequent iterations would reduce the range of the search space but increase the accuaracy of the search. For example, we might start by searching over translations which cover a range of 40 pixels in increments of 5 pixels, and in later iterations, we might cover an eight pixel range, but at single pixel or sub-pixel accuracy. This multiresolution search strategy reflects how the template converges in a coarse to fine fashion. We would not lose much by adopting this approach, since the highest resolutions coefficients of the template are not learned until later iterations.

Another possibility for speeding up convergence is to alternately update $\boldsymbol{\theta}$ and $\mathbf{s}$ until they reach a fixed point, before updating $\mathcal{L}$. All of the convergence results above still apply in this situation. In practice, however, this approach is not necessarily desireable, because it leads to over-fitting the model, and training images get stuck where they are, instead of aligning with each other.

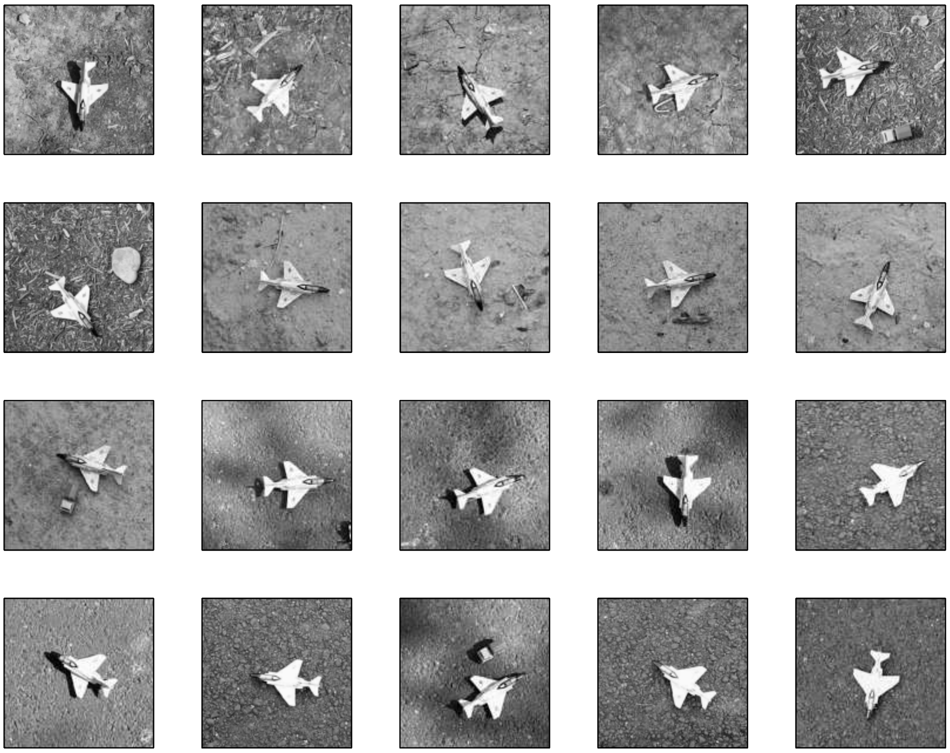
IV. Applications of TEMPLAR

A. Template Learning

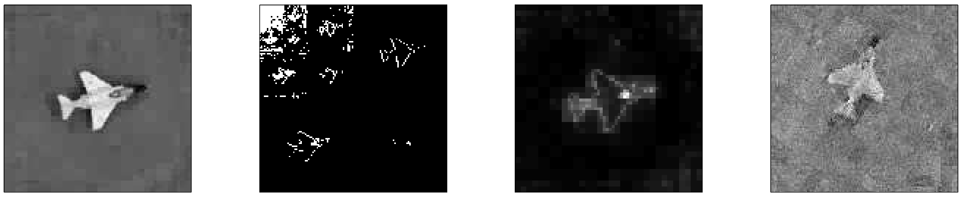
In this section we illustrate template learning on real data. Figure 2 (a) shows 20 observations of a randomly translated and rotated toy airplane. These $128 \times 128 = 16,384$ dimensional images were obtained with a digital camera. Note the varying background, lighting intensity, and location of the lighting source. In learning a template for this data, we use transformations $\boldsymbol{\Gamma}_1, \ldots, \boldsymbol{\Gamma}_L$ that cover rotations up to one degree accuracy, and translations up to one pixel accuracy within a radius of 40 pixels. Thus the total number of transformations modelled is approximately $\pi(40)^2(360) \approx 1.8$ million. Because this data requires a large number of transformations, we successively refined the transformation search space, as described in section III-C, bringing the total number of transformations per iteration down to around 10,000 or less. Other variations in the observations were not explicitly modeled by transformations.

Using the Haar wavelet transform, TEMPLAR converges to an 853-dimensional template in seven iterations. The mean of the template, transformed into the spatial domain, is shown in Figure 2 (b). The most important observation is that TEMPLAR produces a template that captures the structure of the pattern of interest, while ignoring clutter. The details of the airplane are represented fairly well, while the background is smooth.

Figure 2 (c) is a wavelet-domain map of the significant wavelet coefficients. We see (by looking



(a)



(b) (c) (d) (e)

Fig. 2. Template learning: (a) Training data: randomly translated and rotated airplanes, with variable background and lighting conditions. (b) Template mean, transformed into the spatial domain. (c) Map of significant wavelet coeficients (indicated by white). (d) Template variance, transformed into the spatial domain. (e) Observation synthesized from the trained model.

at the highest resolution subbands) that the significant coefficients model *edges* in the pattern, which is where the defining information in the pattern is contained. This is what we would expect from complexity-regularized learning: the significant coefficients should be those wavelet coefficients that contribute the most to defining the structure of the pattern. Figure 2 (d) is a map of the variance of the template in the spatial domain. This map was formed by computing the square of each wavelet basis function, multiplying by the variance of the corresponding

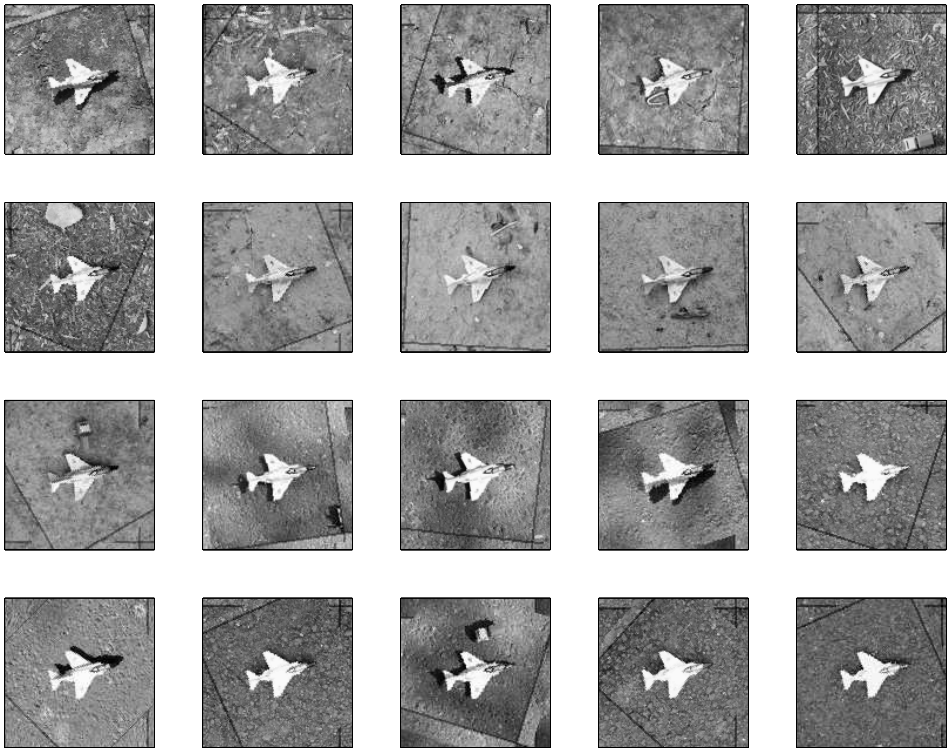


Fig. 3. As a byproduct of TEMPLAR, the training images are registered. This figure shows the registered training images.

wavelet coefficient, and summing over all coefficients. Figure 2 (e) is an image synthesized from the trained model. Since we are assuming a Gaussian noise model, the noise looks Gaussian, although it has the same variance as the pixels in the background of the training patterns. Figure 3 displays the registered images produced by TEMPLAR. The template mean in Figure 2 (b) is simply the superposition of these images, reconstructed using only the significant wavelet coefficients.

B. Pattern Classification

If we are given training images for two or more classes of patterns, we can apply TEMPLAR to produce templates for each class, and use the resulting pattern models for likelihood-based classification. Specfically, if we have trained models $\{\boldsymbol{\theta}_c, \mathbf{s}_c\}_{c=1}^{C}$ for C different classes of patterns, we can use these models to classify an unlabeled test image $\mathbf{x}$ according to a generalized likelihood ratio test (GLRT):

$$c^* = \arg\max_c \left[\max_\ell p(\mathbf{x}|\boldsymbol{\theta}_c, \mathbf{s}_c, \ell) \right]. \tag{5}$$

Fig. 4. Images of the first subject in the Yale Face Database, displaying the variety of facial expressions and lighting conditions used. In our expirement, the images were not registerd, and the background was not cropped out, as it is in this figure.

The GLRT selects the class that has the highest likelihood when evaluated using the most likely transformation for that class. This results in a classifer that is independent of the transformation $\mathbf{\Gamma}_\ell$ that gave rise to $\mathbf{x}$. The number of operations required to classify an image is $O(NM)$.

We illustrate pattern classification by a face recognition example using the Yale Face Database. This database was assembled at the Yale Vision and Robotics Lab, and is available on the Internet at `http://cvc.yale.edu/projects/`. The database consists of 165 images of 15 people, with 11 images of each subject, each having different lighting and/or facial expression. The images for one subject (with the background cropped out) are shown in Figure 4. The same variations in lighting and facial expression were used for each subject.

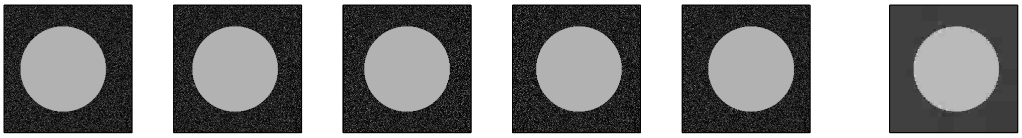
Belhumeur, Hespanha and Kriegman [11] compare the performance of five different face recognition methods on this data set: the "Fisherface" method, a correlation-based method, the Linear Subspace method, and two variations of the Eigenface method. The performance of each method was evaluated using the "leave-one-out" strategy: to classify an image, that image was removed from the data set and a classifier was constructed for each class using the remaining data. TEMPLAR ranks third out of the six methods, with a misclassification rate of 16.5 % when the background was not cropped out of the picture.

This example is not meant to demonstrate that TEMPLAR is a superior method for face recognition. TEMPLAR is a general-purpose framework for pattern analysis, and undoubtedly some methods that are designed specifically for face recognition will outperform it. What is important to notice is that TEMPLAR, while in no way designed for face recognition, performs comparably to these algorithms, *without any pre-processing*. In the experiment reported in [11], each image was manually registered before classification took place. TEMPLAR however requires no pre-processing: the images are automatically registered during the template learning process. Even if the other methods had used an automatic algorithm to register the images beforehand, this registration would by no means align the images in a way that was optimal for each method. With TEMPLAR, however, registration and template learning are jointly optimized under a unified framework.

C. Image Registration

In the process of learning the template parameters $\boldsymbol{\theta}$ and $\mathbf{s}$ associated with a given set of training images, TEMPLAR also learns the transformations (indexed by $\mathcal{L} = (\ell_1, \ldots, \ell_T)$) which align the training images with the pattern template. (See Figure 3.) This suggests that TEMPLAR may be useful for image registration. The standard image registration problem involves finding the transformation taking one image to another. TEMPLAR performs well at the two-image registration problem, but does not appear to offer any advantage over other image registration techniques. Where TEMPLAR may offer advantages over other registration techniques is at the problem of registering multiple observations of the same pattern. We discuss two specific applications below.

The first application we consider is simply the problem of registering a very large number of images. This is often a necessary pre-processing step in many applications of interest. For example, if cross sections of the brain depicting mental activity (acquired under the same modality, e.g., MRI or PET) were obtained from a population under study, the images would need to be aligned before a comparative analysis could take place. Two standard solutions to this problem are as follows: one option is for an individual to manually align the images; another option is to register each image individually to a fixed image of the same pattern. The first solution is time-consuming, tedious, and liable to human error. The second solution requires the use of a rigid template which may or may not be representative of that particular population, and which does not take into consideration the variability of certain features in the pattern. Registration via TEMPLAR is able to overcome these limitations. In the first case, TEMPLAR is fully automatic, and consistently registers the images with respect to a penalized maximum likelihood criterion, thus eliminating human errors due to fatigue or imprecision. In the second case, TEMPLAR benefits from treating the template as a random variable. Thus, if a particular feature of the pattern is highly variable, it will give greater weight to other features during registration. Indeed, the second case may be viewed as the first iteration of TEMPLAR, where we initialize the template instead of the transformations (See Section III-C). Then the final registration produced by TEMPLAR only improves upon the registration produced by the second method.



(a) (b)

Fig. 5. Image registration: (a) Training data from a synthetic example, viewed as a video sequence of a circle moving with respect to a fixed background. Correlation based registration aligns the backgrounds, but TEMPLAR aligns the circles. (b) Mean of the final template.

The second image registration application we consider is the problem of aligning observations where the pattern of interest occurs at different positions relative to a fixed background. This situation may arise, for example, when registering a pattern in a sequence of video frames, where the object of interest is moving but the camera is still. This occurs with surveillance cameras or geosynchronous surveillance satellites. If the background is not smooth but has several edges, a registration algorithm may align the structures in the background, rather than aligning the patterns. The five images in Figure 5 (a) reflect this situation. Here the pattern of interest is a circle, which is located at different positions with respect to a fixed background. A correlation based image registration algorithm aligns the backgrounds in these images, and not the circles. This happens because of the presence of large spikes in the background. However, TEMPLAR is able to successfully align the circles. This is due to the multiresolution nature of the algorithm, whereby TEMPLAR learns the coarse scale coefficients of the pattern first. The circle has significant coefficients at coarse and fine scales, but the textured background has significant coefficients primarily at finer scales. Figure 5 (b) shows the final template mean (transformed into the spatial domain).

D. Denoising from Multiple, Unaligned Observations

We mention briefly one final application of TEMPLAR: wavelet denoising from multiple, unaligned observations. This application is simply another way to view template learning. Previous work on this problem assumes that the pattern is aligned in each of the observations [12]. With our framework, we no longer require this assumption. TEMPLAR automatically registers the observations, and performs denoising by averaging and then setting some wavelet coefficients to zero (although not according to a thresholding rule). The coefficients that are set to zero are precisely the insignificant coefficients.

V. Conclusion

In this paper we present a wavelet-based approach for modeling observations of patterns that have undergone unknown transformations. We introduce TEMPLAR, an iterative, linear-time algorithm that combines the edge-detection property of wavelets with MDL complexity-regularization to learn a low-dimensional template from training data. The dimension of the template is automatically inferred from the data. Once the template has been learned, the resulting model can be used for pattern synthesis or pattern classification. TEMPLAR also applies to image registration or denoising from unaligned observations. We illustrate these applications with real and synthetic examples.

Beyond the specifics of the hierarchical framework and associated learning algorithm presented here, we would like to emphasize two general principles that contribute to TEMPLAR's success, and which might be incorporated into subsequent work on pattern analysis. They are (i) incorporating unknown transformations into the pattern learning process, and (ii) viewing the template as a random variable. Most current wavelet and multiresolution methods either ignore the existence of transformations, or perform registration independently, in a preprocessing step. In the latter case, the pattern learning process depends on how the patterns are registered. With TEMPLAR, the transformations may be updated progressively as more and more details of the pattern are learned.

Regarding (ii), an advantage of viewing the template as a random variable is that when trying to align the template with a given image, certain features which express more variability are not given as much weight. In contrast, when attempting to register an image with a fixed, non-random template, the variability of different features of the pattern is not taken into account. Another advantage is that local deformations in the pattern need not be modeled explicitly by transformations, thus reducing computational complexity. Rather, they can be treated as random fluctuations in the template itself.

While the above properties are important ingredients of TEMPLAR, they are not necessarily novel ideas. The primary contribution of this paper is the incorporation of atomic representations into a pattern theoretic framework. In recent years, considerable effort has been invested in finding good atomic representations (e.g., wavelets) for various kinds of data in order to facilitate signal processing tasks. Unfortunately, most such atomic representations are not invariant to transformations of the image, and therefore it is difficult to construct statistical models for observations of a particular pattern using such representations. Pattern theory provides a framework for de-coupling the modeling of transformations from the modeling of the pattern itself. The hierarchical framework presented in this paper employs this concept to allow a wavelet domain model of pattern observations.

As its name implies, TEMPLAR can use any atomic representation to model the template. Thus, the Discrete Fourier Transform, Discrete Cosine Transform, the identity transformation (corresponding to pixel-domain modeling), or any other orthogonal transform are all possible candidates. However, we saw in Section III-A that convergence of the template learning algorithm is assisted by atomic representations that provide sparse decompositions of the data. The wavelet transform is well-known to provide sparse representations (through both theoretical and empirical results) for a wide class of signals, including natural images, which were the object of interest in this paper. This fact, together with the $O(N)$ complexity of the discrete wavelet transform, make wavelets a natural choice. For several of the examples in this paper, we also attempted template learning using the identity transformation instead of a wavelet transform. The learning algorithm often failed to converge to anything reasonable, and the outcome always resulted in a template that captured more background structure and had a greater dimension than when wavelets were used. We should note, however, that for certain types of data, other transforms may be better adapted than wavelets to the structures in that data, and hence would be more suitable for use with TEMPLAR.

Software for TEMPLAR, in the form of MATLAB m-files, is available at the Rice DSP website: `http://www.dsp.rice.edu/software/`.

Acknowledgements

This work was supported by the National Science Foundation, grant no. MIP–9701692, the Army Research Office, grant no. DAAD19-99-1-0349, and the Office of Naval Research, grant no. N00014-00-1-0390.

Appendix

I. Notes on Implementation of Translation and Rotation

In order to make translation and rotation as energy-preserving as possible, the following implementation was used. For translation, the image was treated as a torus. For example, if an image was translated to the right, those pixels on the right edge of the original image would wrap around and appear on the left edge of the translated image. For rotation, when an image was rotated through an angle which was not a multiple of 90 degrees, the corners of the original image which were cropped off by rotation were mapped in a one-to-one manner to the corners of the rotated image. MATLAB m-files implementing these transformations are available with the rest of the TEMPLAR software at `http://www.dsp.rice.edu/software/`.

II. The Details of TEMPLAR

In this appendix we provide explicit solutions to the optimization problems in Equations (2-4) and give the associated computational complexities.

A. *Estimating Gaussian Mixture Means and Variances*

Consider the problem of maximizing F over the parameter $\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \sigma_0^2\}$, given training data $\mathbf{X} = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$, with $\mathbf{s}$ and $\mathcal{L}$ fixed. The penalty term $c(\mathbf{s})$ does not depend on $\boldsymbol{\theta}$, so this is equivalent to maximizing the likelihood $p(\mathbf{X} \mid \boldsymbol{\theta}, \mathbf{s}, \mathcal{L})$. Denote $\mathcal{W}\boldsymbol{\Gamma}_{\ell_t}^{-1}\mathbf{x}^t$ by $\mathbf{w}^t = (w_1^t, \ldots, w_N^t)^T$. By orthogonality of $\boldsymbol{\Gamma}_{\ell_t}$ and $\mathcal{W}$, it follows from Equation (1) that

$$p(\mathbf{w}^t \mid \boldsymbol{\theta}, \mathbf{s}, \ell_t) \;=\; \mathcal{N}(\, \mathbf{w}^t \mid \boldsymbol{\mu} \,, \boldsymbol{\Sigma}\,). \tag{6}$$

Recall that the i-th wavelet coefficient is modeled as $\mathcal{N}(\mu_{i,s_i}, \sigma_{i,s_i}^2)$. Since the wavelet coefficients are independent and the training images are independent, Equation (6) implies

$$\begin{aligned}\hat{\mu}_{i,1} &= \frac{1}{T}\sum_{t=1}^{T} w_i^t \\ \hat{\sigma}_{i,1}^2 &= \frac{1}{T}\sum_{t=1}^{T}(w_i^t - \hat{\mu}_{i,1})^2\end{aligned}$$

are maximum likelihood estimates for the parameters of the Gaussian densities modeling the significant coefficients. If the number of training images T is small, we may wish to normalize by $T-1$ instead of T to achieve unbiased estimates for the variances. To avoid over fitting the model, if $\sigma_{i,1}^2 < \epsilon$ for some ϵ, we set $\sigma_{i,1}^2 = \epsilon$.

For the insignificant coefficients, the maximum likelihood estimate for σ_0^2 is easily seen to be

$$\hat{\sigma}_0^2 \;=\; \frac{1}{N-k}\sum_{i=1}^{N}(1-s_i)\frac{1}{T}\sum_{t=1}^{T}(w_i^t)^2$$

where $k = \sum s_i$, the number of significant wavelet coefficients. Since $\mathcal{W}$ and $\boldsymbol{\Gamma}_{\ell_t}^{-1}$ can be computed in $O(N)$ operations, the process of estimating $\boldsymbol{\theta}$ requires $O(NT)$ operations.

B. Determining States of Gaussian Mixtures

In order to maximize F with respect to $\mathbf{s}$, with $\boldsymbol{\theta}$ and $\mathcal{L}$ fixed, it is useful to write

$$\log p(\mathbf{X} \,|\, \boldsymbol{\theta}, \mathbf{s}, \mathcal{L}) \;=\; \sum_{i=1}^{N} f_i(s_i)$$

where f_i is the contribution to the overall log-likelihood from the i-th coefficient, as a function of the state variable s_i. The change of variables formula says that

$$p(\mathcal{W}\boldsymbol{\Gamma}_\ell^{-1}\mathbf{x} \,|\, \boldsymbol{\theta}, \mathbf{s}, \ell) \left|\mathcal{W}\boldsymbol{\Gamma}_\ell^{-1}\right| \;=\; p(\mathbf{x} \,|\, \boldsymbol{\theta}, \mathbf{s}, \ell).$$

By orthogonality of $\mathcal{W}$ and $\boldsymbol{\Gamma}_\ell$, we have $\left|\mathcal{W}\boldsymbol{\Gamma}_\ell^{-1}\right| = 1$, so from Equation (6),

$$p(\mathbf{x}^t \,|\, \boldsymbol{\theta}, \mathbf{s}, \ell) \;= \mathcal{N}(\,\mathbf{w}^t \,|\, \boldsymbol{\mu}\,,\, \boldsymbol{\Sigma}\,) \;=\; \frac{1}{\sqrt{2\pi\,|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{w}^t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{w}^t - \boldsymbol{\mu})\right\}. \tag{7}$$

Since the observations are assumed statistically independent,

$$p(\mathbf{X} \,|\, \boldsymbol{\theta}, \mathbf{s}, \mathcal{L}) \;=\; \prod_{t=1}^{T} p(\mathbf{x}^t \,|\, \boldsymbol{\theta}, \mathbf{s}, \ell_t)$$

Therefore we have

$$\begin{aligned}
\log p(\mathbf{X} \,|\, \boldsymbol{\theta}, \mathbf{s}, \mathcal{L}) \;&=\; \sum_{t=1}^{T} \log p(\mathbf{x}^t \,|\, \boldsymbol{\theta}, \mathbf{s}, \ell_t) \\
&=\; \sum_{t=1}^{T} \left[-\frac{1}{2}\log(2\pi\,|\boldsymbol{\Sigma}|) - \frac{1}{2}\sum_{i=1}^{N} \frac{(w_i^t - \mu_{i,s_i})^2}{\sigma_{i,s_i}^2}\right] \\
&=\; -\frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\left[\frac{1}{N}\log 2\pi + \log\sigma_{i,s_i}^2 + \frac{(w_i^t - \mu_{i,s_i})^2}{\sigma_{i,s_i}^2}\right] \\
&=\; -\frac{1}{2}\sum_{i=1}^{N}\left[\frac{T}{N}\log 2\pi + T\log\sigma_{i,s_i}^2 + \sum_{t=1}^{T}\frac{(w_i^t - \mu_{i,s_i})^2}{\sigma_{i,s_i}^2}\right].
\end{aligned}$$

In the third equality, we use the fact that $|\boldsymbol{\Sigma}| = \prod_{i=1}^{N} \sigma_{i,s_i}^2$. Therefore we may define for $m = 0, 1$

$$f_i(m) \;=\; -\frac{1}{2}\left[\frac{T}{N}\log 2\pi + T\log\sigma_{i,m}^2 + \sum_{t=1}^{T}\frac{(w_i^t - \mu_{i,m})^2}{\sigma_{i,m}^2}\right].$$

Now define $d_i = f_i(1) - f_i(0)$. This is the increase in the log-likelihood we achieve when the i-th coefficient is significant as opposed to insignificant. Since our penalty on $\mathbf{s}$ is only a function of the *number* of significant coefficients, if we are going to have any significant coefficients, they should be the coefficients with the largest d_i values. Therefore we order the coefficients so that $d_{i_1} \geq d_{i_2} \geq \cdots \geq d_{i_N}$. The maximum increase in the log-likelihood that can occur from having exactly k significant coefficients is $\sum_{j=1}^{k} d_{i_j}$. But having k significant coefficients carries with it a penalty of $c(\mathbf{s}|k)$. Hence, the number of significant coefficients that maximizes the penalized log-likelihood is

$$k^* = \arg\max_k \left[c(\mathbf{s} \mid k) + \textstyle\sum_{j=1}^{k} d_{i_j} \right],$$

and this maximum is attained by setting $s_{i_j} = 1$ for $j = 1, \ldots, k^*$, and $s_{i_j} = 0$ for $j = k^* + 1, \ldots, N$.

Note that the values w_i^t used to compute d_i were already computed in the previous step, so they do not need to be computed again. The number of operations required to compute the d_i is $O(NT)$, while the number of operations required to compute k^* is $O(N \log N)$, due to the sorting and maximizing.

C. Inferring Hidden Transformations

As in Section B-A, the penalty $c(\mathbf{s})$ does not involve $\mathcal{L}$, so PML estimation of $\mathcal{L}$ given $\boldsymbol{\theta}$ and $\mathbf{s}$ is equivalent to maximum likelihood estimation. Furthermore, since the pattern observations are statistically independent, we may estimate the hidden transformation for each training image independently. The ML estimate $\widehat{\mathcal{L}} = (\hat{\ell}_1, \ldots, \hat{\ell}_T)$ is given by

$$\hat{\ell}_t \;=\; \arg\max_{\ell} \log p(\mathbf{x}^t \mid \boldsymbol{\theta}, \mathbf{s}, \ell). \tag{8}$$

We solve this optimization problem by computing the above log-likelihood for all L transformations. This log-likelihood can be computed from Equation (8), which requires $O(N)$ operations. We repeat this computation L times for each of the T training images, so computing $\widehat{\mathcal{L}}$ requires $O(NLT)$ operations. This dominates the other two steps in TEMPLAR, so the overall running time of TEMPLAR is $O(NLT)$.

III. PROOF OF THEOREM 2

Let $\mathbf{X} \in \mathbf{R}^{NT}$ denote the collection of N-dimensional training images $\mathbf{x}^1, \ldots, \mathbf{x}^T$, viewed as a single concatenated vector. Set $U \equiv \{\mathbf{X} \mid F_k = F_{k-1} \text{ and } \mathcal{T}_k \neq \mathcal{T}_{k-1} \text{ for some } k\}$. We will show $\lambda(U) = 0$ where λ denotes Lesbesque measure. Then the probabiliy of U will be zero, since a Gaussian distribution on $\mathbf{R}^{NT}$ is absolutely continuous with respect to the Lebesque measure.

We may write $F(\boldsymbol{\theta}, \mathbf{s}, \mathcal{L})$ as a function of $\mathbf{X}$:

$$F(\boldsymbol{\theta}, \mathbf{s}, \mathcal{L}) \;=\; \log p(\mathbf{X}|\boldsymbol{\theta}, \mathbf{s}, \mathcal{L}) + c(\mathbf{s})$$

$$= \quad -\frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\left[\frac{1}{N}\log 2\pi + \log\sigma_{i,s_i}^2 + \frac{((\mathcal{W}\mathbf{\Gamma}_{\ell_t}^{-1}x^t)_i - \mu_{i,s_i})^2}{\sigma_{i,s_i}^2}\right]$$

Notice that this is a polynomial in the coordinates of $\mathbf{X}$. Now suppose that $\mathbf{s}$ and $\mathcal{L}$ are fixed, and view $\boldsymbol{\theta} = \boldsymbol{\theta}(\mathbf{X};\mathbf{s},\mathcal{L})$ as a function of the data. From the formulas in section B-A, we see that $\boldsymbol{\theta}$, given $\mathbf{s}$ and $\mathcal{L}$, is a polynomial function of the coordinates of $\mathbf{X}$. Therefore, if $\mathbf{X}_0 \in U$, with $F_k = F_{k-1}$ but $\mathcal{T}_k \neq \mathcal{T}_{k-1}$ for some k, then $\mathbf{X}_0$ is a zero of the polynomial

$$r(\mathbf{X}) = F(\boldsymbol{\theta}(\mathbf{X};\mathbf{s}_{k-2},\mathcal{L}_{k-2}),\mathbf{s}_{k-1},\mathcal{L}_{k-1}) - F(\boldsymbol{\theta}(\mathbf{X};\mathbf{s}_{k-1},\mathcal{L}_{k-1}),\mathbf{s}_k,\mathcal{L}_k).$$

Since $\mathcal{T}_k \neq \mathcal{T}_{k-1}$, this function is nonzero. By the following lemma, the zeros of $r(\mathbf{X})$ comprise a set with measure zero. (See pp. 28-29 of [13] for a proof).

Lemma 1: If $p : \mathbf{R}^n \to \mathbf{R}$ is a nonzero polynomial in n variables, then the set of zeros of p has measure zero with respect to the Lebesgue measure.

Moreover, $r(\mathbf{X})$ is one of a finite family of functions. To see this, observe that $\mathbf{s}$ and $\mathcal{L}$ can take on 2^N and L^T possible values, respectively. Therefore, there are at most $(2^N L^T)^3$ possibilities for $r(\mathbf{X})$, corresponding to the choices for $\mathbf{s}_j$ and $\mathcal{L}_j$, $j = k-2, k-1, k$. Hence, we have shown that if $\mathbf{X} \in U$, then $\mathbf{X}$ belongs to one of a finite number of zero measure sets. Since the finite union of zero measure sets has measure zero, we conclude that U has measure zero. As noted above, this proves the theorem.

References

[1] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, CA, 1998.

[2] C. Scott, *A Hierarchical Wavelet-Based Framework for Pattern Analysis and Synthesis*, Masters Thesis, ECE Department, Rice University, Houston, TX, 2000.

[3] B. J. Frey and N. J. Jojic, "Transformed component analysis: Joint estimation of spatial transformations and image components," *Proceedings of the IEEE International Conference on Computer Vision 1999*, 1999.

[4] U. Grenander and M. J. Miller, "Representations of knowledge in complex systems," *J. Roy. Stat. Soc.*, vol. 56, no. 3, pp. 1–33, 1994.

[5] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models," *IEEE Trans. Signal Processing*, vol. 46, pp. 886–902, 1998.

[6] N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion," *Wavelets in Geophysics*, Foufoula-Georgiou and Kumar (eds.), Academic Press, 1994.

[7] J. Rissanen, *Stochastic Complexity in Stastistical Inquiry*, World Scientific, Singapore, 1989.

[8] H. Krim and I.C. Schick, "Minmax description length for signal denoising and optimal representation," *IEEE Transactions on Information Theory*, vol. 45, no. 3, April, 1999.

[9] M. Figueiredo and J. Leitão, "Bayesian esimation of ventricular contours in angiographic images," *IEEE Transactions on Medical Imaging*, vol. 11, no. 3, pp. 416–29, 1992.

[10] L. Lakshmanan and H. Derin, "Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 11, pp. 799–813, 1989.

[11] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–20, 1997.

[12] G. Chang, B. Yu, and M. Vetterli, "Wavelet thresholding for multiple noisy image copies," to appear in *IEEE Trans. Image Processing*, 2000.

[13] W. M. Wonham, *Linear Multivariable Control*, Springer-Verlag, New York, 1979.