Int J Comput Vis (2010) 90: 198–235
DOI 10.1007/s11263-009-0287-0

Learning Active Basis Model for Object Detection and Recognition

Ying Nian Wu · Zhangzhang Si · Haifeng Gong · Song-Chun Zhu

Received: 28 March 2008 / Accepted: 3 August 2009 / Published online: 26 August 2009

Abstract This article proposes an active basis model, a shared sketch algorithm, and a computational architecture of sum-max maps for representing, learning, and recognizing deformable templates. In our generative model, a deformable template is in the form of an active basis, which consists of a small number of Gabor wavelet elements at selected locations and orientations. These elements are allowed to slightly perturb their locations and orientations before they are linearly combined to generate the observed image. The active basis model, in particular, the locations and the orientations of the basis elements, can be learned from training images by the shared sketch algorithm. The algorithm selects the elements of the active basis sequentially from a dictionary of Gabor wavelets. When an element is selected at each step, the element is shared by all the training images, and the element is perturbed to encode or sketch a nearby edge segment in each training image. The recognition of the deformable template from an image can be accomplished by a computational architecture that alternates the sum maps and the max maps. The computation of the max maps deforms the active basis to match the image data, and the computation of the sum maps scores the template matching by the log-likelihood of the deformed active basis.

Keywords Deformable template · Generative model · Shared sketch algorithm · Sum maps and max maps · Wavelet sparse coding

Y.N. Wu (✉) · Z. Si · H. Gong · S.-C. Zhu
Department of Statistics, University of California, Los Angeles, USA
e-mail: ywu@stat.ucla.edu

Z. Si
e-mail: zzsi@stat.ucla.edu

H. Gong
e-mail: hfgong@stat.ucla.edu

S.-C. Zhu
e-mail: sczhu@stat.ucla.edu

H. Gong · S.-C. Zhu
Lotus Hill Research Institute, Ezhou, China

1 Introduction

Deformable template is an important element in object recognition (Ullman 1996; Yuille et al. 1992; Lades et al. 1993; Cootes et al. 2001; Weber et al. 2000; Amit and Trouve 2007). In this article, we propose a generative model, a model-based algorithm, and a computational architecture for representing, learning and recognizing deformable templates.

1.1 Form of Representation

We call our model the active basis model. An active basis consists of a small number of Gabor wavelet elements at selected locations and orientations. These elements are allowed to slightly perturb their locations and orientations before they are linearly combined to generate the observed image. Figure 1 illustrates the basic idea. The lower half of Fig. 1 shows an active basis, where each element is illustrated by a thin ellipsoid at a certain position and with a certain orientation. The upper half of Fig. 1 illustrates the perturbation of one basis element. Intuitively, each Gabor wavelet element can be considered a "stroke." The template is formed by a composition of a number of strokes. These strokes can be slightly perturbed, so that the template is deformable.

Figure 2 shows a real example. It displays 7 images of cars at the same scale and in the same pose. These images are defined on a common image lattice, which is the bounding box of the cars. These images are represented by an active basis consisting of 60 Gabor wavelet elements, as displayed in the first block of Fig. 2. Each wavelet element is represented symbolically by a bar at the same location and with the same length and orientation as the wavelet element. The length of each element is about 1/10 of the length of the image patch. These elements do not have much overlap and are well connected. They form a common template or an average sketch of the training image patches. The 60 elements of the active basis in the first block of Fig. 2 are allowed to locally change their locations and orientations to code each observed image, as illustrated by the remaining 7 blocks of Fig. 2. Within each block, the left plot displays the observed car image, and the right plot displays the 60 Gabor wavelet elements that are actually used for encoding the corresponding observed image. They form the deformed active basis that sketches the observed image.

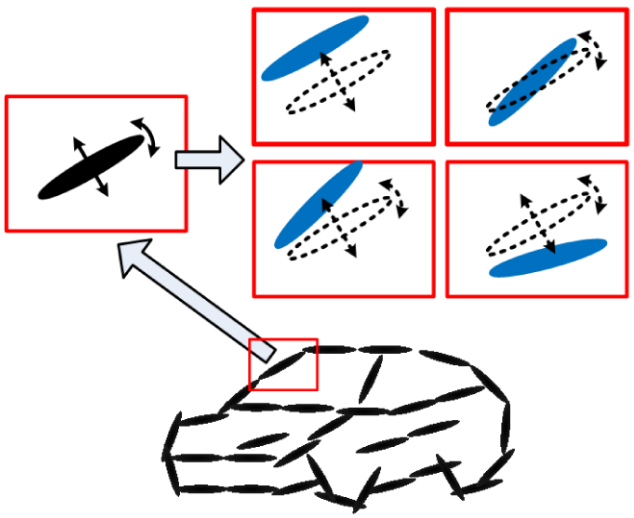


Fig. 1 Active basis. Each basis element is illustrated by a *thin ellipsoid* at certain location and orientation. The upper half shows the perturbation of one basis element. By shifting its location or orientation or both within a limited range, the basis element (illustrated by a *black ellipsoid*) can change to other Gabor wavelet elements (illustrated by the *blue ellipsoids*)

1.2 Scheme of Learning

The active basis, in particular, the locations and the orientations of the basis elements, can be learned from training image patches by the shared sketch algorithm. The algorithm selects the elements of the active basis sequentially from a dictionary. The dictionary consists of Gabor wavelets at a dense collection of locations and orientations. Figure 3 illustrates the selection of three elements by learning from a sample of training images of cars. When an element is selected, the element is shared by all the training images in the sense that a perturbed version of this element is added to improve the encoding of each image. Specifically, the element is perturbed to a location and orientation that achieves the local maximum response within a small neighborhood of the selected element, that is, the perturbed version of the selected element seeks to sketch a nearby edge segment in each training image. For instance, when the green element is selected, it is attracted to the nearby edge in each training image. The same is true for the red element and the blue element.

For each element, a distribution of filter responses is pooled over all the training images at the perturbed locations and orientations. The elements are selected in an order according to the Kullback-Leibler divergence between

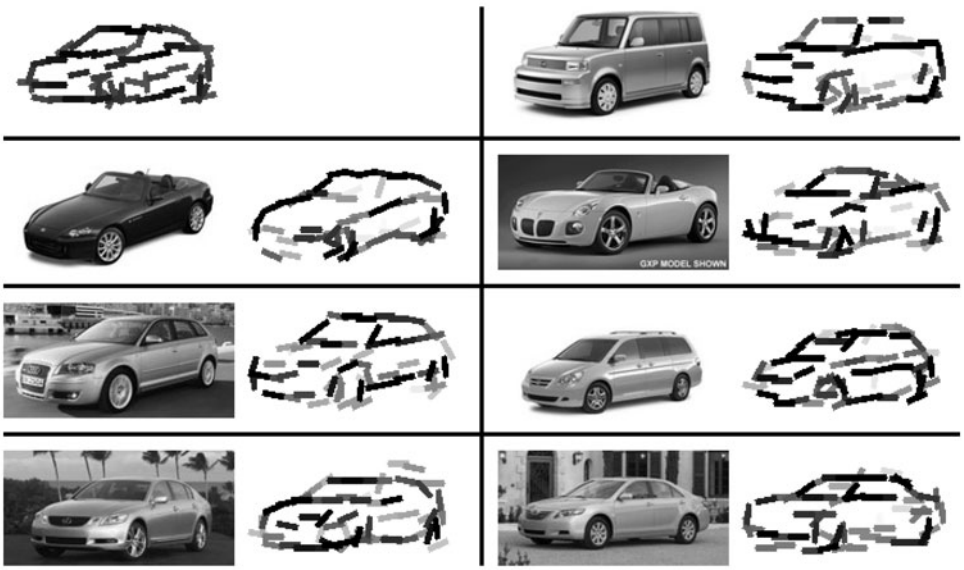


Fig. 2 Active basis formed by 60 Gabor wavelet elements. The first block displays the 60 elements, where each element is represented by a bar. For each of the other 7 blocks, the left plot is the observed image, and the right plot displays the 60 Gabor wavelet elements resulting from locally shifting the 60 elements in the first block to fit the corresponding observed image

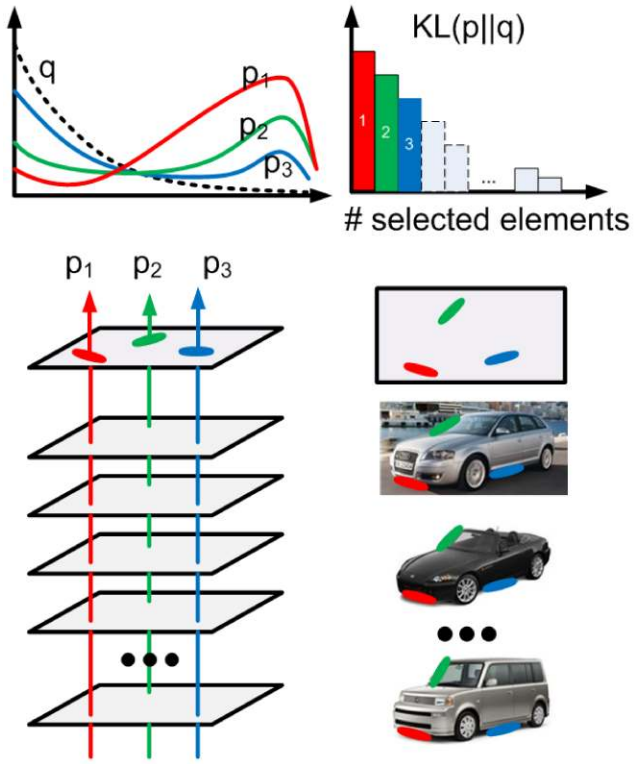


Fig. 3 Shared sketch algorithm. A selected element (*colored ellipsoid*) is shared by all the training images. For each image, a perturbed version of the element seeks to sketch a local edge segment near the element by a local maximization operation. The elements of the active basis are selected sequentially according to the Kullback-Leibler divergence between the pooled distribution (*colored solid curve*) of filter responses and the background distribution (*black dotted curve*). The divergence can be simplified into a pursuit index, which is the sum of the transformed filter responses. The sum essentially counts the number of edge segments sketched by the perturbed versions of the element

the pooled distribution (solid curve) and a background distribution (dotted curve). The background distribution is pooled over natural images. With proper parametrization, the Kullback-Leibler divergence can be reduced to a pursuit index that drives the selection of the elements. This index takes the form of the sum of the transformed filtered responses, summed over all the training images. The transformation is an increasing function that discounts large filter responses. So the pursuit index can be interpreted as a voting of the training images, and the index favors the element whose perturbed versions sketch as many edge segments as possible. After an element is selected, its perturbed version explains away a small part of each training image, and thereby inhibits nearby Gabor wavelet elements from coding the same part of the image. So the selected elements of the active basis are well spaced, and usually form a clear template.

The active basis displayed in Fig. 2 is learned by the shared sketch algorithm. It is worth noting that for the last two examples in Fig. 2, the strong edges in the background are not sketched, because these edges are not shared by other examples, and such edges are ignored by the shared sketch algorithm.

1.3 Architecture of Inference

After learning the active basis from training images, the detection and recognition of the deformable template from a testing image can be accomplished by a computational architecture of sum-max maps. This architecture alternates between sum maps and max maps. The sum maps result from local filtering operations for detecting edge segments and shapes. The max maps result from local maximization operations that track shape deformations. Figure 4 illustrates this architecture. It starts from convolving the image with Gabor filters at all the locations and orientations. The filtered images become the first layer of the sum maps, or SUM1 maps, because each Gabor filter is a local summation operator. In Fig. 4, the thin ellipsoids in the SUM1 maps illustrate the local filtering or summation operation. Then a layer of max maps, or MAX1 maps, is computed by applying a local maximization operator to the SUM1 maps. In Fig. 4, the arrows in the MAX1 maps illustrate that the local maximization is taken over small perturbations of the Gabor wavelets. This local maximization tells us how to deform the active basis to match the image data.

On top of that, a sum map, or SUM2 map, is computed by applying a local summation operator to the MAX1 maps. Specifically, we scan the active basis template over the whole image lattice, and for each pixel of the SUM2 map, we compute a weighted sum of the values of the MAX1 maps, where the summation is over the locations and orientations of the elements of the active basis centered at this pixel. So this is another layer of filtering operation, and can be considered a shape filter. It computes the log-likelihood of the deformed active basis. In Fig. 4, the car template in the SUM2 map illustrates the active basis centered at one pixel. We scan this template over all the pixels to obtain the SUM2 map, which scores the template matching.

The SUM2 map is obtained by a local summation operator of fixed shape. However, because the local summation is applied to the MAX1 maps, shape deformation is automatically accounted for, and the template matching score is invariant to shape deformation.

Besides the log-likelihood scoring for template matching, we also develop a non-probabilistic scoring method based on active correlation between the template and the image. Essentially, the active basis defines the notions of "average" and "correlation" of image patches that are invariant of local shape deformations.

What is described above is a bottom-up scoring process for object detection. After an object is detected, a top-down sketching process is triggered. This process deforms the

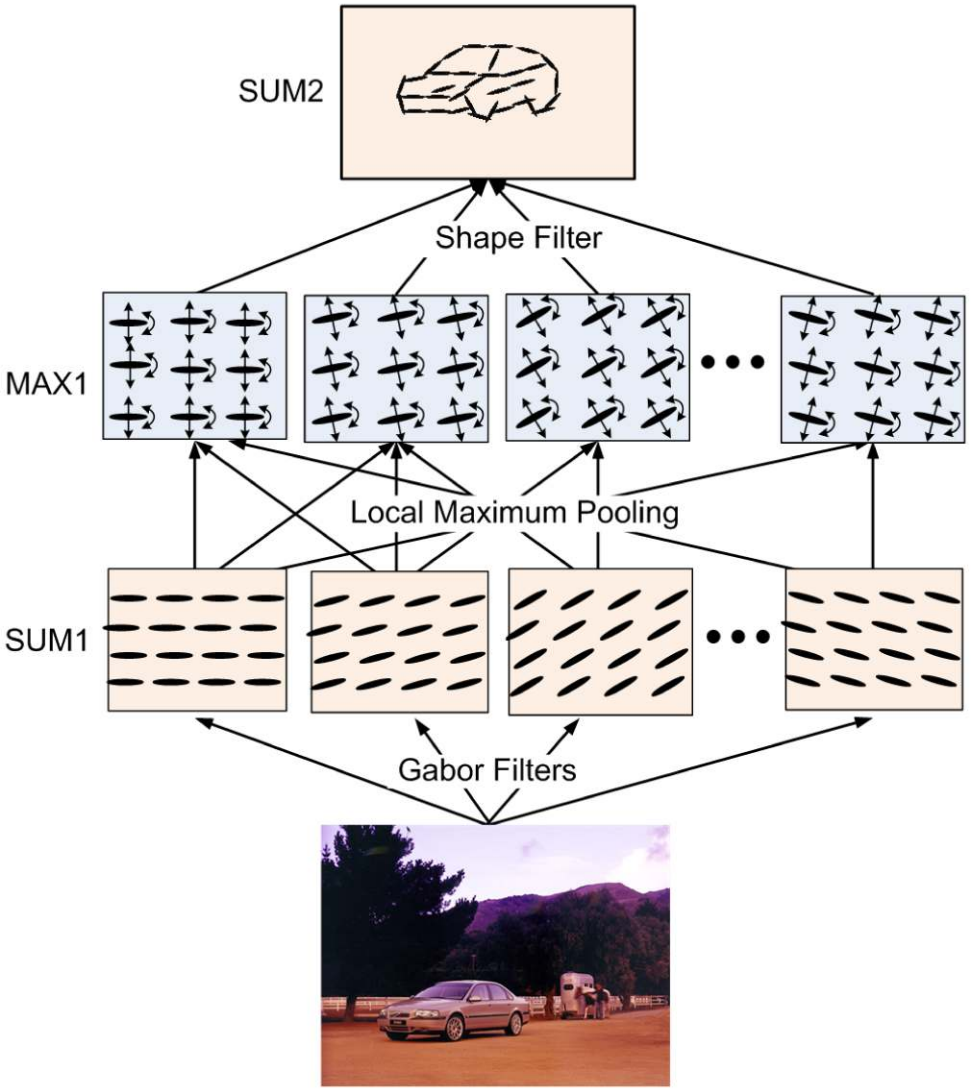


Fig. 4 Sum-max maps. The SUM1 maps are obtained by convolving the input image with Gabor filters at all the locations and orientations. The *ellipsoids* in the SUM1 maps illustrate the local filtering or summation operation. The MAX1 maps are obtained by applying a local maximization operator to the SUM1 maps. The *arrows* in the MAX1 maps illustrate the perturbations over which the local maximization is taken. The SUM2 map is computed by applying a local summation operator to the MAX1 maps, where the summation is over the elements of the active basis. This operation computes the log-likelihood of the deformed active basis, and can be interpreted as a shape filter

template at the detected location, to match the deformable template to the image. This is accomplished by retrieving the locations and orientations of the corresponding Gabor wavelets that achieve the local maxima in the computation of the MAX1 maps.

1.4 Review of Literature

This work is a continuation of our search for generative models of visual patterns, as well as our attempt to understand these models within a common information-theoretical framework (Wu et al. 2008).

For a long time, we have been trying to understand what is beyond the Olshausen and Field's linear sparse coding model (Olshausen and Field 1996). The work of Viola and Jones (2004) based on adaboost (Freund and Schapire 1997) motivated us to apply Olshausen and Field's representation to modeling specific image ensembles of object categories, instead of the generic ensemble of natural images. This led us to retool our previous work on textons (Zhu et al. 2005), in particular, to parallelize the matching pursuit algorithm of Mallat and Zhang (1993) in order to pursue a sparse coding for multiple training images simultaneously.

While the Olshausen and Field's model is intended to explain the role of simple cells in primary visual cortex or V1, the theory of Riesenhuber and Poggio (1999) holds that the V1 complex cells perform local maximum pooling of responses of simple cells. This motivated us to add local perturbations to the locations and orientations of the linear basis elements in the Olshausen and Field's model, so that the linear basis becomes active, and the active basis becomes a deformable template (Yuille et al. 1992). This connects the Olshausen and Field's model to shape models such as active contours (Kass et al. 1988) and active appearance model (Cootes et al. 2001). In the context of the active basis model, the local maximum pooling of Riesenhuber and Poggio can

be interpreted as deforming the active basis to explain the image data.

The active basis model is a simplest instance of the and-or graph (Zhu and Mumford 2006) in the compositional framework (Geman et al. 2002). The and-or grammar naturally suggests that one can further compose multiple active bases to represent more articulate shapes. Such a recursive active basis leads to a recursive architecture of sum-max maps for inference.

2 Representation, Learning, and Inference

This section presents the active basis representation, and describes the shared sketch algorithm and the sum-max maps. We leave theoretical underpinnings and justifications to the next section.

2.1 Gabor Wavelets and Sparse Coding

A dictionary of Gabor wavelets. To fix notation, a Gabor function (Daugman 1985) is of the form $G(x, y) \propto \exp\{-[(x/\sigma_x)^2 + (y/\sigma_y)^2]/2\}e^{ix}$, where $\sigma_x < \sigma_y$. We can translate, rotate, and dilate $G(x, y)$ to obtain a general form of Gabor wavelets: $B_{x,y,s,\alpha}(x', y') = G(\tilde{x}/s, \tilde{y}/s)/s^2$, where $\tilde{x} = (x' - x)\cos\alpha + (y' - y)\sin\alpha$, $\tilde{y} = -(x' - x)\sin\alpha + (y' - y)\cos\alpha$. (x, y) is the central position, s is the scale parameter, and α is the orientation. The Gabor wavelets give reasonable fit to the receptive fields of the simple cells in V1 (Daugman 1985).

The central frequency of $B_{x,y,s,\alpha}$ is $\omega = 1/s$. $B_{x,y,s,\alpha} = (B_{x,y,s,\alpha,0}, B_{x,y,s,\alpha,1})$, where $B_{x,y,s,\alpha,0}$ is the even-symmetric Gabor cosine component, and $B_{x,y,s,\alpha,1}$ is the odd-symmetric Gabor sine component. We always use Gabor wavelets as pairs of cosine and sine components. We normalize both the Gabor sine and cosine components to have zero mean and unit ℓ_2 norm. For each $B_{x,y,s,\alpha}$, $B_{x,y,s,\alpha,0}$ and $B_{x,y,s,\alpha,1}$ are orthogonal to each other.

Let D be the domain of image lattice. The dictionary of Gabor wavelet elements is Dictionary $= \{B_{x,y,s,\alpha}, \forall(x, y, s, \alpha)\}$, where (x, y, s, α) are densely sampled: $(x, y) \in D$ with a fine sub-sampling rate (e.g., every 1 pixel or every 2 pixels), and $\alpha \in \{a\pi/A, a = 0, \ldots, A - 1\}$ (e.g., $A = 15$).

Filtering operation. For an image $\mathbf{I}$ defined on domain D, the projection coefficient of $\mathbf{I}$ onto $B_{x,y,s,\alpha,\eta}$, or the filter response, is $\langle \mathbf{I}, B_{x,y,s,\alpha,\eta}\rangle = \sum_{x',y'} \mathbf{I}(x', y')B_{x,y,s,\alpha,\eta}(x', y')$, where $\eta = 0, 1$. We write $\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle = (\langle \mathbf{I}, B_{x,y,s,\alpha,0}\rangle, \langle \mathbf{I}, B_{x,y,s,\alpha,1}\rangle)$. The local energy is $|\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2 = \langle \mathbf{I}, B_{x,y,s,\alpha,0}\rangle^2 + \langle \mathbf{I}, B_{x,y,s,\alpha,1}\rangle^2$. $|\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2$ measures the local spectrum of $\mathbf{I}$. The local maxima of $|\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2$ can be used to detect edges in $\mathbf{I}$.

Whitening normalization. To make filter responses comparable between different training images, we need to normalize them. Let

$$\sigma^2(s) = \frac{1}{|D|A} \sum_{\alpha} \sum_{(x,y)\in D} |\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2, \tag{1}$$

where $|D|$ is the number of pixels in $\mathbf{I}$, and A is the total number of orientations. $\sigma^2(s)$ measures the power spectrum of $\mathbf{I}$ around frequency $1/s$. For each input image $\mathbf{I}$, we normalize $|\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2$ by changing it to $|\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2/\sigma^2(s)$. This is a whitening normalization, because it makes the power spectrum flat over s.

Linear additive model that explains the image data. A deeper perspective than local filtering is offered by the sparse coding theory of Olshausen and Field (1996), where $B_{x,y,s,\alpha}$ serves as a representational, instead of operational, element. Specifically, for an image $\mathbf{I}$, we can represent it by

$$\mathbf{I} = \sum_{i=1}^{n} c_i B_i + U, \tag{2}$$

where $B_i = B_{x_i,y_i,s_i,\alpha_i}$, (c_i) are the coefficients, and U is the unexplained residual image. Recall that each B_i is a pair of Gabor cosine and sine components. So $B_i = (B_{i,0}, B_{i,1})$. Accordingly, $c_i = (c_{i,0}, c_{i,1})$ and $c_i B_i = c_{i,0}B_{i,0} + c_{i,1}B_{i,1}$. The set of Gabor wavelet elements $(B_i, i = 1, \ldots, n)$ are selected from the dictionary. If the $(B_i, i = 1, \ldots, n)$ are orthogonal, i.e., if they do not overlap in spatial domain or frequency domain, then $c_i = \langle \mathbf{I}, B_i\rangle$.

Sparse coding means that for a typical natural image $\mathbf{I}$, one can usually select a small number n of elements from the dictionary, so that a linear combination of these elements can represent $\mathbf{I}$ with a small residual U. Of course, for different images, one usually selects different sets of elements. The wavelet sparse coding representation (2) reduces an image of tens of thousands of pixels to a small number of wavelet elements or strokes. Using the sparse coding principle, Olshausen and Field (1996) were able to learn from natural image patches a dictionary of Gabor-like wavelet elements that closely resemble the properties of the receptive fields of the simple cells in V1.

Matching pursuit that explains away the image data. The matching pursuit algorithm of Mallat and Zhang (1993) is a commonly used method for fitting model (2). Each step of matching pursuit explains away a small part of image data by selecting a wavelet element, which then inhibits nearby neighboring elements from being included in the linear representation. This idea is used in the shared sketch algorithm.

2.2 Representation: Active Basis Model

The sparse coding model (2) is intended to model the whole ensemble of natural images, where for different $\mathbf{I}$, one may

represent them with completely different wavelet elements $(B_i, i = 1, \ldots, n)$ with different n. In the active basis model, we apply the sparse coding model (2) to image ensembles of various object categories. Then for each category, we require that the images share the same set of wavelet elements $(B_i, i = 1, \ldots, n)$. These elements form a common template. However, when we use $(B_i, i = 1, \ldots, n)$ to encode each individual image, we allow the template to slightly deform, by allowing the elements or strokes to perturb their locations and orientations.

Let $\{\mathbf{I}_m, m = 1, \ldots, M\}$ be a set of training image patches defined on a common rectangle lattice D. We assume that D is the bounding box of the objects in $\{\mathbf{I}_m\}$, and these objects are from the same category and in the same pose. We shall relax this assumption later.

Our method is scale specific. We fix s so that the length of $B_{x,y,s,\alpha}$ (e.g., 17 pixels) is fixed. We can learn templates at multiple scales and then combine them.

Wavelet expansion with perturbations. The active basis model is a composition of strokes that are perturbable:

$$\text{Composition:} \quad \mathbf{I}_m = \sum_{i=1}^{n} c_{m,i} B_{m,i} + U_m, \tag{3}$$

$$\text{Perturbations:} \quad B_{m,i} \approx B_i, \quad i = 1, \ldots, n, \tag{4}$$

where $B_i \in$ Dictionary $B_{m,i} \in$ Dictionary $(c_{m,i}, i = 1, \ldots, n)$ are the coefficients, and U_m is the unexplained residual image. To define the perturbation $B_{m,i} \approx B_i$, suppose

$$B_i = B_{x_i, y_i, s, \alpha_i}, \tag{5}$$

$$B_{m,i} = B_{x_{m,i}, y_{m,i}, s, \alpha_{m,i}}, \tag{6}$$

then $B_{m,i} \approx B_i$ if and only if there exists $(d_{m,i}, \delta_{m,i})$ such that

$$x_{m,i} = x_i + d_{m,i} \cos \alpha_i, \tag{7}$$

$$y_{m,i} = y_i + d_{m,i} \sin \alpha_i, \tag{8}$$

$$\alpha_{m,i} = \alpha_i + \delta_{m,i}, \tag{9}$$

$$d_{m,i} \in [-b_1, b_1], \qquad \delta_{m,i} \in [-b_2, b_2]. \tag{10}$$

That is, we allow B_i to shift its location along its normal direction, and we also allow B_i to shift its orientation. See Fig. 1 for an illustration. We call $(d_{m,i}, \delta_{m,i})$ the activity or perturbation of B_i in image $\mathbf{I}_m$. b_1 and b_2 are the bounds for the allowed activities (e.g., $b_1 = 6$ pixels, and $b_2 = \pi/15$).

In the above notation, the active basis $\mathbf{B} = (B_i, i = 1, \ldots, n)$ forms a deformable template. The deformed active basis is $\mathbf{B}_m = (B_{m,i}, i = 1, \ldots, n) \approx \mathbf{B}$. See Fig. 2 for an illustration.

It is important to distinguish between $\mathbf{B}$ and $\mathbf{B}_m$. $\mathbf{B}$ is the common "average" template shared by all the examples $\{\mathbf{I}_m\}$. $\mathbf{B}_m$ is the image specific template that only describes $\mathbf{I}_m$. $\mathbf{B}$ is learned from all the training images $\{\mathbf{I}_m\}$, and it can generalize to testing images, because the basis elements in $\mathbf{B}$ are active.

Because we fix the scale s in the representation (3) to (10), the linear superposition $\sum_{i=1}^{n} c_{m,i} B_{m,i}$ only explains the frequency band of $\mathbf{I}_m$ around the frequency $\omega = 1/s$, while leaving the remaining frequency components to the unexplained U_m. U_m can be further explained by templates at other scales.

2.3 Learning: Shared Sketch Algorithm

Given the set of training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$, the shared sketch algorithm sequentially selects B_i and perturbs it to $B_{m,i} \approx B_i$ to sketch each image $\mathbf{I}_m$. The basic idea is to select B_i so that its perturbed versions $\{B_{m,i}, m = 1, \ldots, M\}$ sketch as many edge segments as possible in the training images $\{\mathbf{I}_m\}$.

Shared sketch algorithm

Input: Training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$.

Output: Common template $\mathbf{B} = (B_i, i = 1, \ldots, n)$, and deformed template $\mathbf{B}_m = (B_{m,i}, i = 1, \ldots, n)$ that sketches $\mathbf{I}_m$ for $m = 1, \ldots, M$.

1. Convolution: For each $m = 1, \ldots, M$, and for each $B \in$ Dictionary, compute $[\mathbf{I}_m, B] = h(|\langle \mathbf{I}_m, B \rangle|^2)$. Set $i \leftarrow 1$.
2. Local maximization: For each putative candidate $B_i \in$ Dictionary, do the following: For each $m = 1, \ldots, M$, choose the optimal $B_{m,i}$ that maximizes $[\mathbf{I}_m, B_{m,i}]$ among all possible $B_{m,i} \approx B_i$.
3. Selection: Choose that particular candidate B_i whose corresponding $\sum_{m=1}^{M} [\mathbf{I}_m, B_{m,i}]$ achieves the maximum among all possible $B_i \in$ Dictionary. Record this B_i and retrieve the corresponding optimal $B_{m,i} \approx B_i$ for $m = 1, \ldots, M$.
4. Non-maximum suppression: For each $m = 1, \ldots, M$, if $[\mathbf{I}_m, B_{m,i}] > 0$, then for every $B \in$ Dictionary such that $\text{corr}(B, B_{m,i}) > \epsilon$, set $[\mathbf{I}_m, B] \leftarrow 0$.
5. Stop if $i = n$. Otherwise let $i \leftarrow i + 1$, and go back to 2.

In the above description, $h()$ is a monotone increasing (or non-decreasing) transformation that discounts large value of $|\langle \mathbf{I}_m, B \rangle|^2$. $[\mathbf{I}_m, \mathbf{B}]$ records the response of $\mathbf{B}$ to $\mathbf{I}_m$. It can change during the algorithm because of the non-maximum suppression.

For two Gabor elements B_1 and B_2, $\text{corr}(B_1, B_2) = \sum_{\eta_1=0}^{1} \sum_{\eta_2=0}^{1} \langle B_{1,\eta_1}, B_{2,\eta_2} \rangle^2$ measures their correlation or overlap in spatial and frequency domains. B_1 and B_2 are orthogonal as long as they do not overlap in either spatial domain or frequency domain. The non-maximum suppression step suppresses those B that overlap with the selected

$B_{m,i}$. ϵ (e.g., $\epsilon = .1$) is the tolerance of the overlap between selected basis elements in the deformed active basis.

See Fig. 3 for an illustration of the above algorithm.

Comparison with edge detection. The algorithm can be considered a parallel version of edge detection simultaneously applied to multiple images. For a putative B_i, the local maximization step seeks to sketch a local edge segment in image $\mathbf{I}_m$ by a perturbed version $B_{m,i} \approx B_i$. The selection step seeks to find B_i with the strongest overall response $\sum_{m=1}^{M}[\mathbf{I}_m, B_{m,i}]$, which pools the edge strengths from the training images around B_i. After B_i is selected, we retrieve the corresponding $B_{m,i}$, and let $B_{m,i}$ suppress or inhibit nearby overlapping Gabor elements B by setting the response $[\mathbf{I}_m, B] \leftarrow 0$. So for each image $\mathbf{I}_m$, the selected $(B_{m,i}, i = 1, \ldots, n)$ are approximately orthogonal to each other.

If $M = 1$ and if we forbid perturbations in locations and orientations by setting $b_1 = b_2 = 0$, then the algorithm reduces to usual edge detection.

For $M > 1$, the shared sketch algorithm seeks to accomplish the following two tasks: (1) Eliminating the background edges. (2) Averaging the foreground shapes.

Transformation of responses. To understand the transformation $h()$, let us consider a simplified discontinuous one: $h(r) = 1_{r>\xi}$, where ξ is a threshold for edge detection. More specifically, $h(r) = 1$ if $r > \xi$, and $h(r) = 0$ otherwise. Then $\sum_{m=1}^{M} h(r_{m,i})$ simply counts the number of detected edge segments in the training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$. That is, we select B_i and perturb it to $\{B_{m,i}\}$, so that $\{B_{m,i}\}$ sketch as many edge segments as possible.

In this article we entertain the following designs of continuous transformations. The learned templates are not very sensitive to the choice of the transformation.

(1) Sigmoid transformation. The transformation is characterized by a saturation level ξ (e.g., $\xi = 6$),

$$h(r) = \text{sigmoid}(r) = \xi\left[\frac{2}{1 + e^{-2r/\xi}} - 1\right], \tag{11}$$

which increases from 0 to ξ, and $h'(0) = 1$.

(2) Whitening transformation. Let $q(r)$ be the marginal distribution of $r = |\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2$ where $\mathbf{I}$ is a random sample from the ensemble of natural images. Let $F(t) = q(r > t)$, i.e., the probability that $r > t$ under $q(r)$. The nonlinear whitening transformation is

$$h(r) = \text{whiten}(r) = -\log F(r). \tag{12}$$

On top of the whitening normalization in Sect. 2.1, the nonlinear whitening transformation (12) makes the marginal distribution of $|\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2$ the same as that of the white noise.

(3) Thresholding transformation. A crude but simple approximation to whiten(r) is

$$h(r) = \text{threshold}(r) = \min(r, T), \tag{13}$$

where T is a threshold (e.g., $T = 16$).

Scoring template matching. Let $\mathbf{B} = (B_i, i = 1, \ldots, n)$ be the template. For each training image $\mathbf{I}_m$, the template matching is scored by

$$\text{MATCH}(\mathbf{I}_m, \mathbf{B}) = \sum_{i=1}^{n}(\lambda_i[\mathbf{I}_m, B_{m,i}] - \log Z(\lambda_i)). \tag{14}$$

λ_i can be calculated directly from $\sum_{m=1}^{M}[\mathbf{I}_m, B_{m,i}]$ in the selection step. $Z()$ is a non-linear function. This template matching score is actually a log-likelihood ratio for an exponential family model, and the weight vector $\Lambda = (\lambda_i, i = 1, \ldots, n)$ is estimated by maximum likelihood method. See the next section for details.

Active correlation. We can also use a linear score for template matching:

$$\text{MATCH}(\mathbf{I}_m, \mathbf{B}) = \sum_{i=1}^{n}\theta_i[\mathbf{I}_m, B_{m,i}], \tag{15}$$

where $h(r) = \text{whiten}(r)^{1/2}$ or $h(r) = \text{threshold}(r)^{1/2}$, and $\Theta = (\theta_i, i = 1, \ldots, n)$ is a unit vector, with $\|\Theta\|^2 = \sum_{i=1}^{n}\theta_i^2 = 1$. The elements are still selected by the shared sketch algorithm, with the aforementioned new definition of $h()$. To estimate Θ, we first calculate $\theta_i = \sum_{m=1}^{M}[\mathbf{I}_m, B_{m,i}]/M$, then we normalize $\Theta = (\theta_i, i = 1, \ldots, n)$ to be a unit vector.

The template matching score (15) can be considered the active correlation between the template $\mathbf{B}$ and the image $\mathbf{I}_m$, where $\mathbf{B}$ is deformed to $\mathbf{B}_m = (B_{m,i}, i = 1, \ldots, n)$ before the inner product is calculated. We may also consider (15) as the inner product between $\mathbf{I}_m$ and the vector $V = \sum_{i=1}^{n}\theta_i B_i$. V is an active vector because B_i can be perturbed to $B_{m,i}$ when we correlate V with $\mathbf{I}_m$. V can be considered an active average of the images $\{\mathbf{I}_m\}$.

2.4 Inference: Sum-Max Maps

After training the active basis model, specifically, after selecting $\mathbf{B} = (B_i = B_{x_i,y_i,s,\alpha_i}, i = 1, \ldots, n)$, and computing the weight vector $\Lambda = (\lambda_i, i = 1, \ldots, n)$ or $\Theta = (\theta_i, i = 1, \ldots, n)$, we can use the trained model to detect and then sketch the object in a testing image.

Let $\mathbf{I}$ be a testing image defined on a lattice D. Here we use the notation D to denote the lattice of $\mathbf{I}$ instead of the bounding box of the template $\mathbf{B}$, which is usually smaller than D. We assume that the bounding box of the template $\mathbf{B}$ is centered at origin $(x = 0, y = 0)$. We can scan the template over D, and at each position $(x, y) \in D$, we fit the active basis model to the image patch of $\mathbf{I}$ within the bounding box (or the scanning window) centered at (x, y), and calculate the template matching score according to (14) or (15).

Pseudo-code for inference algorithm

Input: Template $\mathbf{B} = (B_i = B_{x_i,y_i,s,\alpha_i}, i = 1, \ldots, n)$, $\Lambda = (\lambda_i, i = 1, \ldots, n)$, and testing image $\mathbf{I}$.

Output: Location $(\hat{x}, \hat{y})$ of the detected object, and the deformed template $(B_{\hat{x}_i,\hat{y}_i,s,\hat{\alpha}_i}, i = 1, \ldots, n)$ that sketches $\mathbf{I}$.

Up-1 For all $(x, y) \in D$, and for all α, compute the SUM1 maps:

$$\text{SUM1}(x, y, s, \alpha) = |\langle \mathbf{I}, B_{x,y,s,\alpha} \rangle|^2.$$

Up-2 For all $(x, y) \in D$, and for all α, compute the MAX1 maps:

$$\begin{aligned}&\text{MAX1}(x, y, s, \alpha)\\ &= \max_{\substack{d\in[-b_1,b_1]\\ \delta\in[-b_2,b_2]}} \text{SUM1}(x + d\cos\alpha,\\ &\quad y + d\sin\alpha, s, \alpha + \delta).\end{aligned} \tag{16}$$

Let $(\hat{d}, \hat{\delta})$ be the value of (d, δ) that achieves the maximum in (16). Let $\hat{x} = x + \hat{d}\cos\alpha$, $\hat{y} = y + \hat{d}\sin\alpha$, and $\hat{\alpha} = \alpha + \hat{\delta}$. Record $\text{TRACK1}(x, y, s, \alpha) = (\hat{x}, \hat{y}, \hat{\alpha})$.

Up-3 For all $(x, y) \in D$, compute the SUM2 map:

$$\begin{aligned}&\text{SUM2}(x, y)\\ &= \sum_{i=1}^{n} [\lambda_i h(\text{MAX1}(x + x_i, y + y_i, s, \alpha_i))\\ &\quad - \log Z(\lambda_i)].\end{aligned}$$

Up-4 Compute the MAX2 score: $\text{MAX2} = \max_{(x,y)\in D} \text{SUM2}(x, y)$.

Down-4 Retrieve $(\hat{x}, \hat{y})$ that achieves the maximum in the computation of Up-4.

Down-3 Retrieve $(\hat{x} + x_i, \hat{y} + y_i, \alpha_i)$ in the computation of $\text{MAX1}(x + x_i, y + y_i, s, \alpha_i)$ for $i = 1, \ldots, n$ in Up-3.

Down-2 Retrieve $(\hat{x}_i, \hat{y}_i, \hat{\alpha}_i) = \text{TRACK1}(\hat{x} + x_i, \hat{y} + y_i, s, \alpha_i)$, for $i = 1, \ldots, n$, where the TRACK1 maps are defined in Up-2.

Down-1 Retrieve the coefficients in the computation of $\text{SUM1}(\hat{x}_i, \hat{y}_i, s, \hat{\alpha}_i)$ for $i = 1, \ldots, n$ in Up-1.

Bottom-up detection and top-down sketching. The inference algorithm consists of two processes. The first process is a bottom-up detection process, which calculates SUM1, MAX1, SUM2, MAX2 scores consecutively. The following are the questions that these scores seek to answer:

SUM1 maps: Is there an edge segment at this location and orientation?

MAX1 maps: Is there an edge segment at a *nearby* location and orientation? Where is it?

SUM2 map: Is there a certain composition of edge segments that form the template at this location?

MAX2 score: Is there a certain composition within the whole image?

These maps are soft scores, not hard decisions. They are computed in a bottom-up process, SUM1 → MAX1 → SUM2 → MAX2.

This is to be followed by a top-down retrieving process, which retrieves the central location of the template and then retrieves the locations and orientations of the basis elements of the deformed template. The following are the questions to be answered:

Back to MAX2 score: If there is a template, where is it?

Back to SUM2 map: What are the locations and orientations of the elements of the template before deformation?

Back to MAX1 maps: What are the nearby locations and orientations that these elements are perturbed to?

Back to SUM1 maps: What are the coefficients of these perturbed elements?

The top-down retrieving process follows the sequence MAX2 → SUM2 → MAX1 → SUM1. The process deforms the template to sketch the observed image.

Shape filter. The SUM2 map in Up-3 scores template matching. The computation of SUM2 can be considered a shape filter for template matching. Like Gabor filters, it is also a local weighted summation operator. See Fig. 4 for an illustration. The shape filter in Up-3 has fixed $(x_i, y_i, \alpha_i, i = 1, \ldots, n)$. But it is computed on the MAX1 maps instead of SUM1 maps, so it is invariant to shape deformation.

For an input image, we can apply the above algorithm at multiple resolutions of the input image. Then we can choose the resolution that achieves the maximum MAX2 score as the optimal resolution.

Comparison with Riesenhuber and Poggio's cortex-like structure. The above sum-max structure is inspired by the cortex-like structure of Riesenhuber and Poggio (1999). The differences are as follows: (1) The TRACK1 maps are recorded in Up-2 step together with the MAX1 maps. The TRACK1 maps link the locations and orientations of the MAX1 maps back to the locations and orientations of the SUM1 maps where the local maxima are achieved. (2) A SUM2 operator is used for template matching. This operator is learned from training images. (3) A top-down sketching process is triggered after the bottom-up detection process. The top-down process is guided by the TRACK1 maps. (4) The selected wavelet elements in the deformed active basis inhibit nearby overlapping elements, especially in the learning stage.

2.5 Shared Sketch Algorithm Based on Sum-Max Maps

The shared sketch algorithm in Sect. 2.3 can be expressed more precisely in terms of the sum maps and max maps.

Pseudo-code for shared sketch algorithm

Input: Training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$.

Output: Template $\mathbf{B} = (B_i = B_{x_i, y_i, s, \alpha_i}, i = 1, \ldots, n)$, weight vector $\Lambda = (\lambda_i, i = 1, \ldots, n)$, and deformed template $\mathbf{B}_m = (B_{m,i} = B_{x_{m,i}, y_{m,i}, s, \alpha_{m,i}}, i = 1, \ldots, n)$ that sketches $\mathbf{I}_m$ for $m = 1, \ldots, M$.

1. Convolution: For each $m = 1, \ldots, M$, for all $(x, y) \in D$, and for all α, compute the SUM1 maps $\text{SUM1}_m(x, y, s, \alpha) = |\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle|^2$, in the same way as in the Up-1 step of the inference algorithm.
2. Local maximization: For each $m = 1, \ldots, M$, for all $(x, y) \in D$, and for all α, compute the MAX1 maps:

$$\text{MAX1}_m(x, y, s, \alpha) = \max_{\substack{d \in [-b_1, b_1] \\ \delta \in [-b_2, b_2]}} \text{SUM1}_m(x + d\cos\alpha, y + d\sin\alpha, s, \alpha + \delta), \quad (17)$$

   and record $\text{TRACK1}_m(x, y, s, \alpha)$, in the same way as in the Up-2 step of the inference algorithm.
   For each $m = 1, \ldots, M$, set $\text{SUM2}_m \leftarrow 0$. Set $i \leftarrow 1$.
3. Selection: Find (x_i, y_i, α_i) by maximizing $\sum_{m=1}^{M} h(\text{MAX1}_m(x, y, s, \alpha))$ over all (x, y, α). Compute λ_i from $\sum_{m=1}^{M} h(\text{MAX1}_m(x_i, y_i, s, \alpha_i))$. Update $\text{SUM2}_m \leftarrow \text{SUM2}_m + \lambda_i h(\text{MAX1}_m(x_i, y_i, s, \alpha_i)) - \log Z(\lambda_i)$ for each $m = 1, \ldots, M$.
4. Non-maximum suppression: Retrieve $(x_{m,i}, y_{m,i}, \alpha_{m,i}) = \text{TRACK1}_m(x_i, y_i, s, \alpha_i)$ for each $m = 1, \ldots, M$, similar to the Down-2 step of the inference algorithm.
   If $\text{MAX1}_m(x_i, y_i, s, \alpha_i) > 0$, then for all those (x, y, α) such that $\text{corr}(B_{x_{m,i}, y_{m,i}, s, \alpha_{m,i}}, B_{x,y,s,\alpha}) > \epsilon$, set $\text{SUM1}_m(x, y, s, \alpha) \leftarrow 0$.
   Re-compute the MAX1 maps according to (17).
5. Stop if $i = n$. Otherwise let $i \leftarrow i + 1$, and go back to Step 3.

The above algorithm can be easily mapped to computer code. The following are remarks on implementing it:

(1) In updating the SUM1 maps and the MAX1 maps in Step 4, we only need to update the parts of the maps that are affected.

(2) The correlation $\text{corr}(B_{x_{m,i}, y_{m,i}, s, \alpha_{m,i}}, B_{x,y,s,\alpha})$ in Step 4 only depends on $(x_{m,i} - x, y_{m,i} - y, \alpha_{m,i} - \alpha)$. We can store a correlation function $\text{corr}(\Delta x, \Delta y, \Delta\alpha) = \text{corr}(B_{x+\Delta x, y+\Delta y, s, \alpha+\Delta\alpha}, B_{x,y,s,\alpha})$ before running the algorithm.

Multiple alignment score. The SUM2_m score evaluates the matching of $\mathbf{I}_m$ to the learned template $\mathbf{B}$ according to (14). The total score $\sum_{m=1}^{M} \text{SUM2}_m$ measures the overall alignment of multiple training images. This multiple alignment score is very useful for unsupervised learning, where the objects in the training images are of unknown locations, scales, and categories. The alignment score $\sum_{m=1}^{M} \text{SUM2}_m$ is the criterion that determines these hidden variables.

We would like to point out a subtle difference between the computation of SUM2_m score in the learning algorithm and the computation of SUM2 map in the inference algorithm. In the learning algorithm, there is a non-maximum suppression step, where $B_{m,i}$ suppresses nearby overlapping elements. This is necessary for selecting the basis elements. In the inference algorithm, we omit this step for efficiency. This is because the elements selected by the learning algorithm are already well spaced due to the non-maximum suppression in learning, so there is not much need for non-maximum suppression in inference.

3 Theoretical Underpinning

This section presents theoretical underpinnings of the model and the algorithms presented in the previous section. Readers who are more interested in applications and experiments can jump to the next section.

3.1 Probability Distribution on Image Intensities

With multiple training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$ represented by (3) to (10), we can pool the probability distribution of $\{(c_{m,i}, i = 1, \ldots, n)\}$ as well as the distribution of $\{U_m\}$ over $m = 1, \ldots, M$. With these two distributions, we can obtain the distribution of $\mathbf{I}_m$, or more specifically, the distribution of $\mathbf{I}_m$ given $\mathbf{B}_m$, $p(\mathbf{I}_m \mid \mathbf{B}_m)$. With the probability density $p(\mathbf{I}_m \mid \mathbf{B}_m)$, both learning and inference can be based on maximizing the likelihood function.

We first simplify the notation using matrices and vectors. $\mathbf{I}_m$ can be treated as a $|D| \times 1$ column vector, where $|D|$ is the number of pixels. $\mathbf{B} = (B_{i,0}, B_{i,1}, i = 1, \ldots, n)$ can be treated as a $|D| \times 2n$ matrix, where each $B_{i,\eta}$ $(\eta = 0, 1)$ is a $|D| \times 1$ vector. Each $\mathbf{B}_m$ can be treated as a $|D| \times 2n$ matrix in the same way. We can write $C = (c_{m,0}, c_{m,1}, i = 1, \ldots, n)'$ as a $2n \times 1$ vector. Thus in matrix notation, (3) becomes $\mathbf{I}_m = \mathbf{B}_m C_m + U_m$.

Linear decomposition. We assume that $\mathbf{B}_m C_m$ is the projection of $\mathbf{I}_m$ onto the subspace spanned by the column vectors of $\mathbf{B}_m$, so $C_m = (\mathbf{B}_m' \mathbf{B}_m)^{-1} \mathbf{B}_m' \mathbf{I}_m$. If $\mathbf{B}_m$ is orthogonal, then $C_m = \mathbf{B}_m' \mathbf{I}_m$. U_m resides in the $|D| - 2n$ dimensions that are orthogonal to the columns of $\mathbf{B}_m$. There is no loss of generality in such an assumption, because if U_m is not orthogonal to $\mathbf{B}_m$, we can always project U_m onto $\mathbf{B}_m$, and let $\mathbf{B}_m C_m$ absorb this projection. We can write $U_m = \bar{\mathbf{B}}_m \bar{C}_m$, where $\bar{\mathbf{B}}_m$ is a $|D| \times (|D| - 2n)$ matrix whose columns are

orthogonal to those of $\mathbf{B}_m$, and $\bar{C}_m$ is a $(|D| - 2n) \times 1$ vector. Thus, $\mathbf{I}_m = \mathbf{B}_m C_m + \bar{\mathbf{B}}_m \bar{C}_m$. There is a one-to-one linear mapping between $\mathbf{I}_m$ and $(C_m, \bar{C}_m)$. $\bar{\mathbf{B}}_m$ and $\bar{C}_m$ can be made implicit in statistical modeling.

Shape and texture. Now we are ready to specify the probability density $p(\mathbf{I}_m \mid \mathbf{B}_m)$. For the linear representation $\mathbf{I}_m = \mathbf{B}_m C_m + \bar{\mathbf{B}}_m \bar{C}_m$,

$$\begin{aligned} p(\mathbf{I}_m \mid \mathbf{B}_m) &= p(C_m, \bar{C}_m)|J_m| \\ &= p(C_m)p(\bar{C}_m \mid C_m)|J_m|, \end{aligned} \tag{18}$$

where $|J_m|$ is the absolute value of the determinant of the Jacobian matrix of the linear transformation from $\mathbf{I}_m$ to $(C_m, \bar{C}_m)$. $p(C_m)$ is the distribution of the coefficients for coding the foreground shape, and $p(\bar{C}_m \mid C_m)$ is the distribution of the residual background texture given the foreground coefficients. The distribution $p(\mathbf{I}_m \mid \mathbf{B}_m)$ is fully determined by $p(C_m)$ and $p(\bar{C}_m \mid C_m)$.

Let $q(\mathbf{I}_m)$ be a reference distribution. Similar to (18), we can write $q(\mathbf{I}_m) = q(C_m)q(\bar{C}_m \mid C_m)|J_m|$ with the same Jacobian J_m. We want to construct $p(\mathbf{I}_m \mid \mathbf{B}_m)$ by modifying $q(\mathbf{I}_m)$. Specifically, we assume that $p(\bar{C}_m \mid C_m) = q(\bar{C}_m \mid C_m)$, i.e., the conditional distribution of the residual background in $p(\mathbf{I}_m \mid \mathbf{B}_m)$ is assumed to be the same as that in $q(\mathbf{I}_m)$. Then

$$\begin{aligned} p(\mathbf{I}_m \mid \mathbf{B}_m) &= q(\mathbf{I}_m)\frac{p(C_m)}{q(C_m)} \\ &= q(\mathbf{I}_m)\frac{p(c_{m,1}, \ldots, c_{m,n})}{q(c_{m,1}, \ldots, c_{m,n})}, \end{aligned} \tag{19}$$

where we substitute $p(C_m)$ for $q(C_m)$ to construct a density $p(\mathbf{I}_m)$ from $q(\mathbf{I}_m)$.

The model (19) combines both texture and shape. $q(\mathbf{I}_m)$ models the background texture, and $\mathbf{B}_m$ and $p(C_m)$ model the foreground shape. The foreground shape pops out from the background texture, as modeled by the probability ratio $p(C_m)/q(C_m)$.

Density substitution and maximum entropy. The form (19) is a density substitution scheme that has been used in projection pursuit (Friedman 1987). It is also valid if C_m is a non-linear differentiable reduction of $\mathbf{I}_m$, or if C_m is discrete. Such a form enables us to build a probability model on image intensities instead of features. Such a generative model makes it possible to select the features by explaining away the image data. Model (19) can be justified by the maximum entropy principle (Pietra et al. 1997): $p(\mathbf{I}_m \mid \mathbf{B}_m)$ is the distribution that is closest to $q(\mathbf{I}_m)$ in terms of Kullback-Leibler divergence among all the distributions that share the same $p(C_m)$.

Choices of reference distribution. We assume $q(\mathbf{I}_m)$ to be stationary. The following are some choices of $q(\mathbf{I}_m)$:

(1) Gaussian white noise distribution. This is the distribution that is often assumed in linear additive model, and is implicitly assumed in the least squares criterion for model fitting. Under this reference distribution, $q(c_{m,1}, \ldots, c_{m,n})$ is multivariate Gaussian. If $(B_{m,i}, i = 1, \ldots, n)$ are orthogonal to each other, then $(c_{m,i}, i = 1, \ldots, n)$ are independent. We call this the orthogonal-independence property.

(2) Non-Gaussian marginal approximation to the distribution of natural image patches. This is the distribution $q(\mathbf{I}_m)$ that we shall use in this paper. In particular, we assume that the marginal distributions of $\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle$ are all the same as that in natural images. Such a marginal distribution is highly non-Gaussian, with a heavy tail that allows occasional strong edges. We also assume that $q(\mathbf{I}_m)$ inherits the orthogonal-independence property from Gaussian white noise. Such a distribution is the simplest modification of the Gaussian white noise distribution, and it provides a better model than the Gaussian white noise for the background U_m, by allowing strong edges in U_m.

Figure 5 shows the two natural images that we use for pooling the marginal distribution of Gabor filter responses. The left one is a rural scene, which has more textures. The right one is an urban scene, which is more regular.

Figure 6 displays the marginal histogram of sigmoid$(|\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle|^2)$ pooled over all (x, y, α) from the two natural images in Fig. 5. It is a long-tailed distribution. The small bump at the end is caused by the saturation of the sigmoid transformation. Different scales s produce very similar histograms.

Fig. 5 Two 768 × 1024 (height × length) natural images that are used to pool the marginal distribution of filter responses

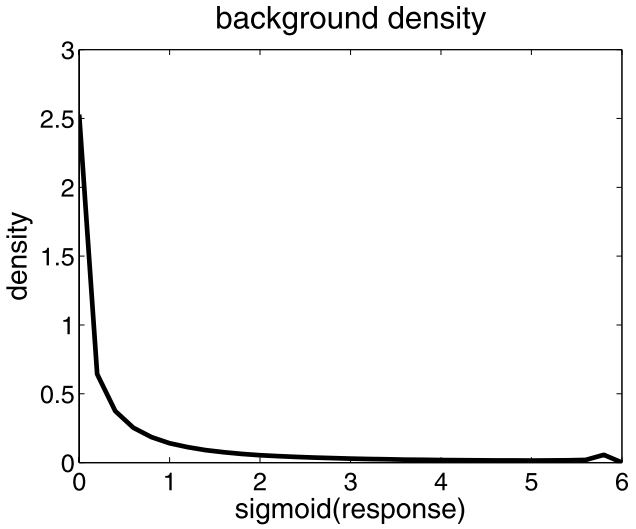


Fig. 6 The density of sigmoid($|\langle \mathbf{I}_m, B_{x,y,s,\alpha}\rangle|^2$) pooled over all (x, y, α) from the two natural images in Fig. 5

A more formal model for $q(\mathbf{I}_m)$ is the Markov random field model that Zhu and Mumford (1997) developed for natural images. For this model, the orthogonal-independence property is approximately true.

(3) The Markov random field distribution that matches the marginal distributions of filter responses of the observed image $\mathbf{I}_m$. Such a model has been developed by Zhu et al. (1997). The marginal distributions are pooled from the observed image $\mathbf{I}_m$ over $(x, y) \in D$, instead of the above two natural images. This model is related to the adaptive background to be discussed in Sect. 3.6.

Log-likelihood and Kullback-Leiber divergence. To learn $\mathbf{B}$ and $\{\mathbf{B}_m \approx \mathbf{B}, m = 1, \ldots, M\}$, we can maximize the average log-likelihood ratio

$$\frac{1}{M}\sum_{m=1}^{M} \log \frac{p(\mathbf{I}_m \mid \mathbf{B}_m)}{q(\mathbf{I}_m)} = \frac{1}{M}\sum_{m=1}^{M} \log \frac{p(c_{m,1}, \ldots, c_{m,n})}{q(c_{m,1}, \ldots, c_{m,n})}. \quad (20)$$

The average log-likelihood ratio converges to $\mathrm{KL}(p(C_m) \,\|\, q(C_m))$ as $M \to \infty$, provided that $p(C_m)$ can be consistently estimated from the training images. Here $\mathrm{KL}(p \,\|\, q)$ denotes the Kullback-Leibler divergence from p to q. In order to maximize the log-likelihood ratio, we want to choose $\mathbf{B}$ and deform it to $\{\mathbf{B}_m \approx \mathbf{B}\}$ to maximize $\mathrm{KL}(p(C_m) \,\|\, q(C_m))$, so that the maximum contrast is achieved between the foreground shape and the background texture. $\mathrm{KL}(p(C_m) \,\|\, q(C_m))$ also measures the coding gain achieved by coding C_m by $p(C_m)$ instead of $q(C_m)$, while continuing to code the residual background by $q(\bar{C}_m|C_m)$.

It is impossible to select $\mathbf{B}$ and $\{\mathbf{B}_m\}$ all at once. In the next subsection, we present an algorithm that sequentially pursues B_i and perturbs it to $\{B_{m,i}\}$.

3.2 Coupling Matching Pursuit with Projection Pursuit

In this subsection, we describe a shared matching pursuit process for selecting the basis elements $\mathbf{B} = (B_i, i = 1, \ldots, n)$. The process couples matching pursuit (Mallat and Zhang 1993) with projection pursuit (Friedman 1987). The matching pursuit is used to encode each training image. The projection pursuit is used to estimate the probability density of the image by pooling the coefficients produced by the matching pursuit.

The matching pursuit is a process that sequentially adds elements $B_{m,i}, i = 1, \ldots, n$ to improve the encoding of image $\mathbf{I}_m$. It has the following form:

1. For $m = 1, \ldots, M$, set $U_m \leftarrow \mathbf{I}_m$. Set $i \leftarrow 1$.
2. For $m = 1, \ldots, M$, choose $B_{m,i}$. Let $c_{m,i} = \langle U_m, B_{m,i}\rangle$.
3. For $m = 1, \ldots, M$, update $U_m \leftarrow U_m - c_{m,i} B_{m,i}$. Represent $\mathbf{I}_m = c_{m,1} B_{m,1} + \cdots + c_{m,i} B_{m,i} + U_m$.
4. If $i = n$, stop. Otherwise, set $i \leftarrow i + 1$, go back to Step 2.

We need to add the following three components to the above matching pursuit process.

(1) *The selection of $B_{m,i}$ given B_i*. The original matching pursuit algorithm selects $B_{m,i} = \arg\max_B |\langle U_m, B\rangle|^2$ in Step 2, where the maximization is over all $B \in$ Dictionary, so that $B_{m,i}$ achieves the best fit to the unexplained residual image U_m. In shared matching pursuit process, however, the $B_{m,i}$ are constrained to be perturbed versions of a commonly shared B_i. Therefore, for each putative B_i, we need to select $B_{m,i} = \arg\max_{B \approx B_i} |\langle U_m, B\rangle|^2$.

(2) *The updating of $p(\mathbf{I}_m)$*. After computing $c_{m,i} = \langle U_m, B_{m,i}\rangle$ in each iteration i, we can pool a distribution $p_i(c)$ from $\{c_{m,i}, m = 1, \ldots, M\}$. We can use such pooled densities $p_1(c), \ldots, p_n(c)$ to construct the density $p(\mathbf{I}_m)$.

Specifically, we update $p(\mathbf{I}_m)$ sequentially using projection pursuit. Let $p_0(\mathbf{I}_m) = q(\mathbf{I}_m)$, i.e., the distribution of background texture. At each iteration i, after selecting $\{B_{m,i}, m = 1, \ldots, M\}$, we need to update $p_{i-1}(\mathbf{I}_m)$ to $p_i(\mathbf{I}_m)$. We can apply the density substitution scheme of projection pursuit, and let

$$p_i(\mathbf{I}_m) = p_{i-1}(\mathbf{I}_m)\frac{p_i(c_{m,i})}{q_{i-1}(c_{m,i})}, \quad (21)$$

where $q_{i-1}(c)$ is the density of $c_{m,i} = \langle U_m, B_{m,i}\rangle$ under the current model $p_{i-1}(\mathbf{I}_m)$. This density substitution scheme is very similar to the model construction scheme of (19), except that we use $p_{i-1}(\mathbf{I}_m)$ as the current reference distribution, and we only substitute the density of $c_{m,i} = \langle U_m, B_{m,i}\rangle$ under $p_{i-1}(\mathbf{I}_m)$. $c_{m,i} = \langle U_m, B_{m,i}\rangle$ can also be written as $c_{m,i} = \langle \mathbf{I}_m, \tilde{B}_{m,i}\rangle$, where $\tilde{B}_{m,i}$ can be constructed from

$B_{m,1}, \ldots, B_{m,i-1}$ and $B_{m,i}$. So $p_i(\mathbf{I}_m)$ is a legitimate density function.

(3) *The selection of* B_i. We select B_i sequentially by the maximum likelihood principle. The increase in the average log-likelihood is

$$\frac{1}{M}\sum_{m=1}^{M} \log \frac{p_i(\mathbf{I}_m)}{p_{i-1}(\mathbf{I}_m)} = \frac{1}{M}\sum_{m=1}^{M} \log \frac{p_i(c_{m,i})}{q_{i-1}(c_{m,i})},$$

which converges to $\mathrm{KL}(p_i(c) \,\|\, q_{i-1}(c))$. So we want to select B_i that achieves the maximum $\mathrm{KL}(p_i(c) \,\|\, q_{i-1}(c))$. That is, $\mathrm{KL}(p_i(c) \,\|\, q_{i-1}(c))$ is the pursuit index that drives the selection of B_i. Intuitively, this means that we want to select B_i so that the distribution of the responses of the perturbed versions $\{B_{m,i} \approx B_i\}$ is most different from what is predicted by the current model $p_{i-1}(\mathbf{I}_m)$.

With the above components (1), (2), and (3) incorporated into the matching pursuit process, we will eventually reach the model $p(\mathbf{I}_m) = q(\mathbf{I}_m)\prod_{i=1}^{n} p_i(c_{m,i})/q_{i-1}(c_{m,i})$. This is an approximation to the model (19) in the previous subsection. See Fig. 3 for an illustration of the shared matching pursuit process.

The computational burden in the shared matching pursuit process lies in the computation of $q_{i-1}(c)$ in (21), which requires Monte Carlo sampling from $p_{i-1}(\mathbf{I}_m)$. If we have negative training images, we can re-weight these negative examples after each iteration, and use these re-weighted examples as samples from $p_{i-1}(\mathbf{I}_m)$.

3.3 Shared Sketch Algorithm

We can simplify the shared matching pursuit process into a shared sketch algorithm.

Non-maximum suppression. After selecting $B_{m,i}$ and computing $c_{m,i} = \langle U_m, B_{m,i}\rangle$, we need to update $U_m \leftarrow U_m - c_{m,i}B_{m,i}$, i.e., $B_{m,i}$ explains away part of U_m or $\mathbf{I}_m$. This can be considered a soft inhibition. If an element B has a high correlation with $B_{m,i}$, in other words, if B heavily overlaps with $B_{m,i}$ in both spatial domain and frequency domain, then such a redundant B can add little to further explaining $\mathbf{I}_m$, in that after updating $U_m \leftarrow U_m - c_{m,i}B_{m,i}$, $|\langle U_m, B\rangle|^2$ can be very small. Therefore, we may simply enforce that, for each $\mathbf{I}_m$, the selected elements of $\mathbf{B}_m = (B_{m,i}, i = 1, \ldots, n)$ do not overlap with each other, or the selected $(B_{m,i}, i = 1, \ldots, n)$ are orthogonal to each other. Then, after $B_{m,i}$ is selected, we let $B_{m,i}$ suppress any B that overlaps with $B_{m,i}$. For such non-overlapping $(B_{m,i}, i = 1, \ldots, n)$, $c_{m,i} = \langle U_m, B_{m,i}\rangle = \langle \mathbf{I}_m, B_{m,i}\rangle$. In practice, we allow small correlations between the elements $(B_{m,i}, i = 1, \ldots, n)$.

Such a hard inhibition has the advantage that it forces the selected elements to be well spaced and form a clean template.

Background density. Let the reference distribution $q(\mathbf{I}_m)$ be the non-Gaussian marginal approximation to the distribution of natural images, as explained in Sect. 3.1. Then $(c_{m,i}, i = 1, \ldots, n)$ are independent for orthogonal $(B_{m,i}, i = 1, \ldots, n)$, a property inherited from white noise. Therefore, $q_{i-1}(c) = q(c)$, which is the marginal distribution of $c_{m,i}$ under $q(\mathbf{I}_m)$. Because $q(\mathbf{I}_m)$ is stationary, $q(c)$ is the same for all $c_{m,i}, i = 1, \ldots, n$. Hence, the pursuit index is $\mathrm{KL}(p_i(c) \,\|\, q(c))$, where, again, $p_i(c)$ is the density pooled from $\{c_{m,i}, m = 1, \ldots, M\}$.

If we stop the process after n iterations, then the resulting model is

$$p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m)\prod_{i=1}^{n} \frac{p_i(c_{m,i})}{q(c_{m,i})}. \tag{22}$$

$q(c)$ can be pooled from natural images before we start the shared sketch algorithm. We do not need negative examples beyond $q(c)$. See Sect. 3.1 and Fig. 6.

3.4 Parametrization by Exponential Family Model

Parametric model. We can further simplify the Kullback-Leibler divergence by assuming the following exponential family model:

$$p(c; \lambda) = \frac{1}{Z(\lambda)} \exp\{\lambda h(r)\} q(c), \tag{23}$$

where $\lambda > 0$ is the parameter, $r = |c|^2$, and

$$Z(\lambda) = \int \exp\{\lambda h(r)\} q(c) dc = \mathrm{E}_q[\exp\{\lambda h(r)\}] \tag{24}$$

is the normalizing constant. $h(r)$ is an increasing function, so $p(c; \lambda)$ puts more probability than $q(c)$ on those c with large r. The above model can be justified by the maximum entropy principle (Pietra et al. 1997).

Let $p(r; \lambda)$ and $q(r)$ be the densities of $r = |c|^2$ under $p(c; \lambda)$ and $q(c)$ respectively, then $p(c; \lambda)/q(c) = p(r; \lambda)/q(r) = \exp\{\lambda h(r)\}/Z(\lambda)$.

Estimating p_i. We estimate $q(c)$ by pooling a histogram from natural images. See Fig. 6. We estimate $p_i(c)$ from $\{c_{m,i} = \langle \mathbf{I}_m, B_{m,i}\rangle, m = 1, \ldots, M\}$ by fitting the density $p(c; \lambda_i)$ to $\{c_{m,i}\}$. Specifically, let us define the mean parameter

$$\mu(\lambda) = \mathrm{E}_\lambda[h(r)] = \int h(r) \frac{1}{Z(\lambda)} \exp\{\lambda h(r)\} q(r) dr. \tag{25}$$

Figure 7 shows the function $\mu(\lambda)$. We estimate the parameter λ_i by solving the following estimating equation

$$\mu(\lambda_i) = \frac{1}{M}\sum_{m=1}^{M} h(r_{m,i}), \tag{26}$$

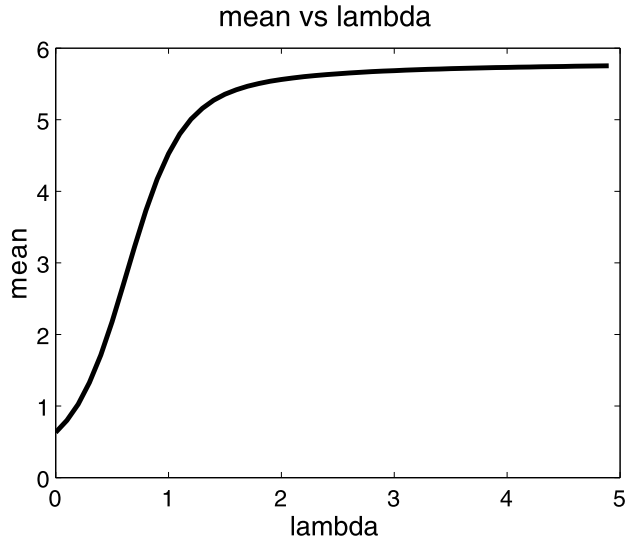


Fig. 7 The function $\mu(\lambda)$, where $h(r)$ is the sigmoid transformation

where $r_{m,i} = |c_{m,i}|^2$, so that $\hat{\lambda}_i = \mu^{-1}(\sum_{m=1}^{M} h(r_{m,i})/M)$. This is done by inverting the function in Fig. 7. $\hat{\lambda}_i$ is the maximum likelihood estimate that maximizes $\sum_{m=1}^{M} \log[p(c_{m,i}; \lambda_i)/q(c_{m,i})]$ over λ_i (Pietra et al. 1997). We estimate $p_i(c)$ by $p(c; \hat{\lambda}_i)$.

To avoid over-fitting, we impose an upper bound on λ (e.g., $\lambda < 5$). That is, in the rare case where no value of λ below the upper bound satisfies the estimating equation (26), we then let the estimated λ be this upper bound. The upper bound plays a role mostly in single image learning, which we shall discuss at the end of this subsection.

Both $\log Z(\lambda)$ in (24) and $\mu(\lambda)$ in (25) are one-dimensional monotone functions. We can store their values over a grid of λ values below the upper bound mentioned above, and use nearest neighbor linear interpolation for points in between. The solution to the estimating equation (26) can be efficiently obtained by looking up the stored monotone function $\mu(\lambda)$.

Selecting B_i. The average log-likelihood ratio

$$\frac{1}{M}\sum_{m=1}^{M} \log \frac{p(c_{m,i}; \hat{\lambda}_i)}{q(c_{m,i})} = \mathrm{KL}(p(c; \hat{\lambda}_i) \,\|\, q(c)). \tag{27}$$

It is an increasing function of $\sum_{m=1}^{M} h(r_{m,i})/M$. Therefore, we choose B_i and perturb it to $\{B_{m,i}\}$ by maximizing the pursuit index $\sum_{m=1}^{M} h(r_{m,i})$.

Perturbing B_i to $B_{m,i}$. $p(c; \lambda_i)/q(c)$ is a monotone increasing function of $r = |c|^2$. This justifies that, given B_i, we should perturb B_i to $B_{m,i}$ to maximize $|\langle \mathbf{I}_m, B_{m,i}\rangle|^2$, subject to the approximate non-overlapping constraint. Such $B_{m,i}$ is the maximum likelihood estimate given B_i.

Thus, the estimation of λ_i, the perturbation of B_i to $B_{m,i}$, and the selection of B_i all follow the maximum likelihood principle.

Template matching score. The resulting model is

$$\begin{aligned} &p(\mathbf{I}_m \mid \mathbf{B}_m) \\ &= q(\mathbf{I}_m)\prod_{i=1}^{n} \frac{p_i(c_{m,i})}{q(c_{m,i})} \\ &= q(\mathbf{I}_m)\prod_{i=1}^{n} \frac{1}{Z(\hat{\lambda}_i)} \exp\{\hat{\lambda}_i h(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2)\}. \end{aligned} \tag{28}$$

To score the template matching, we can compute the log-likelihood ratio

$$\begin{aligned} &\log \frac{p(\mathbf{I}_m \mid \mathbf{B}_m)}{q(\mathbf{I}_m)} \\ &= \sum_{i=1}^{n} [\hat{\lambda}_i h(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2) - \log Z(\hat{\lambda}_i)]. \end{aligned} \tag{29}$$

From a classification perspective, $(h(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2), i = 1, \ldots, n)$ are the features that tell apart the positive examples from p and the negative examples from q. See also Tu (2007) for a related model.

Single image learning. Because of the parametrization in the form of the exponential family model, we can learn the model from a single image. This enables us to initialize unsupervised learning by fitting the model to a single image. For single image learning, we set $b_1 = b_2 = 0$, i.e., we do not allow any activity. In that case, the estimated common template $\mathbf{B}$ is the same as the deformed template $\mathbf{B}_m$. However, after learning $\mathbf{B}$ in this way, we immediately re-set b_1 and b_2 to their normal values for detection purpose. The $\mathbf{B}$ with re-set (b_1, b_2) is an active basis that can generalize to other images. The upper bound imposed on λ_i helps avoid overfitting in single image learning. The current upper bound ($\lambda_i < 5$) still appears too large for single image learning, and can be further reduced.

3.5 Transformation and Normalization

The following are explanations why we use the sigmoid and whitening transformations for $h(r)$.

Sigmoid transformation. The saturation in the sigmoid transformation can be justified by mixture distributions.

Let $p_{\mathrm{on}}(r)$ be the density of $r = |\langle \mathbf{I}, B_{x,y,s,\alpha}\rangle|^2$ when the Gabor wavelet $B_{x,y,s,\alpha}$ is on an edge. Let $p_{\mathrm{off}}(r)$ be the density of r when the Gabor wavelet is off the edge. We assume that $p_{\mathrm{on}}(r)$ has a much longer tail than $p_{\mathrm{off}}(r)$. Let $q(r)$ and $p_i(r)$ be the densities of $r = |c|^2$ under $q(c)$ and $p_i(c)$ respectively. It is reasonable to assume that $q(r) = (1 - \rho_0)p_{\mathrm{off}}(r) + \rho_0 p_{\mathrm{on}}(r)$, and $p_i(r) = (1 - \rho_i)p_{\mathrm{off}}(r) + \rho_i p_{\mathrm{on}}(r)$. That is, both $q(r)$ and $p_i(r)$ are mixtures of the same on-distribution and off-distribution, with $\rho_i > \rho_0 > 0$. As $r \to \infty$, $\log[p_i(r)/q(r)] \to \log(\rho_i/\rho_0) >$

0, i.e., a positive constant. So we may assume the following log-linear model: $\log[p_i(c)/q(c)] = \log[p_i(r)/q(r)] = \lambda_i h(r) + \text{constant}$, where $\lambda_i > 0$, and $h(r)$ reaches a fixed saturation level as $r \to \infty$. This justifies the saturation in the sigmoid transformation.

Whitening transformation. The whitening transformation makes $q(\mathbf{I}_m)$ closer to the white noise distribution, which is a simpler null hypothesis. It also leads to explicit expressions of $\mu(\lambda)$ and $\log Z(\lambda)$.

Let $F(t) = q(r > t)$, i.e., the probability that $r > t$ under $q(r)$ or $q(\mathbf{I}_m)$. The reason we call $h(r) = -\log F(r)$ the whitening transformation is that $\Pr(h(r) > t) = \Pr(-\log F(r) > t) = \Pr(F(r) < e^{-t}) = e^{-t}$, i.e., $h(r)$ follows Exponential distribution with unit expectation. This is the distribution of r if $q(\mathbf{I}_m)$ is Gaussian white noise. This is because the local energy r is the sum of the squares of the Gabor sine response and Gabor cosine response, and both of them follow independent Normal distributions if $q(\mathbf{I}_m)$ is Gaussian white noise. So their sum of squares follows a χ^2_2 distribution, which is the Exponential distribution. The distribution has expectation 1 because we normalize the image to have unit $\sigma^2(s)$. See (1).

The whitening transformation changes a long-tailed distribution $q(r)$ to a short-tailed Exponential distribution. With the whitening transformation, under $p(c; \lambda)$ of (23), $h(r) \sim \text{Exp}(1 - \lambda)$, which is an Exponential distribution with $\mu(\lambda) = 1/(1 - \lambda)$. $Z(\lambda) = 1/(1 - \lambda)$. λ_i can be estimated by $\hat{\lambda}_i = 1 - M/\sum_{m=1}^{M} h(r_{m,i})$.

Normalization schemes. Before applying the transformation, we need to normalize the filter responses by dividing them by the average energy or power spectrum $\sigma^2(s)$ in (1). However, there can be various schemes for computing $\sigma^2(s)$, due to various choices of the domain D within which we pool the average. The following are some options: (1) Let D be the domain of the whole image. This is the simplest option. However, the bounding box of the training images may be much smaller than the image domain of the testing image, and that causes inconsistency in learning and testing. The following two options solve the inconsistency problem: (2) When we scan the template over the testing image, we normalize the filter responses by the average pooled within the scanning window of the template. (3) For both training and testing images, for each $B_{x,y,s,\alpha}$, we normalize its response by the average pooled within a local window centered at (x, y).

For the experiments in this paper, we have implemented option (2) for Experiments 2, 5a, and 5b in the detection stage. We use option (3) for Experiment 1.3. We use the simple option (1) in Experiments 3, 5c, 6, and 10, which are of illustrative nature.

3.6 Adaptive Texture Background

Marginal histograms. In model (28), $p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m) \prod_{i=1}^{n} p_i(c_{m,i})/q(c_{m,i})$, where $q(c)$ is the marginal distribution of filter responses pooled over the two natural images in Sect. 3.1. In scoring an image $\mathbf{I}_m$, the log-likelihood ratio score is computed by $\sum_{i=1}^{n} \log[p_i(c_{m,i})/q(c_{m,i})]$, according to (29). We can change the generic $q(c)$ to a background texture model fitted specifically to $\mathbf{I}_m$. Specifically, for each image $\mathbf{I}_m$, and for each orientation α, let $q_{m,\alpha}(c)$ be the marginal distribution (or histogram) pooled from $\{\langle \mathbf{I}_m, B_{x,y,s,\alpha}\rangle, \forall (x, y)\}$. Then we can score the image $\mathbf{I}_m$ by $\sum_{i=1}^{n} \log[p_i(c_{m,i}; \hat{\lambda}_i)/q_{m,\alpha_i}(c_{m,i})]$, where α_i is the orientation of $B_{m,i}$. See Fig. 8 for an illustration.

The marginal histogram $q_{m,\alpha}$ captures texture information in $\mathbf{I}_m$, and provides the adaptive image-specific background for scoring the template matching. Such spatially pooled histograms have been commonly used in literature. For instance, Zhu et al. (1997) developed a Markov random field model for textures based on such histograms. In fact, the above scheme amounts to assuming the model $p(\mathbf{I}_m \mid \mathbf{B}_m) = q_m(\mathbf{I}_m) \prod_{i=1}^{n} p_i(c_{m,i})/q_{m,\alpha_i}(c_{m,i})$, where $q_m(\mathbf{I}_m)$ is the Markov random field model (Zhu et al. 1997) fitted to $\mathbf{I}_m$ by matching to the marginal histograms of $\mathbf{I}_m$. Therefore, the active basis model for shape leads to a natural justification for the Markov random field model for texture.

Template matching score against adaptive background. Just like we can further parameterize $p_i(c)$ by the exponential family model $p(c; \lambda_i)$ as defined in (23), $q_{m,\alpha}$ can also be parameterized in the same form. Let

$$h_\alpha(\mathbf{I}_m) = \frac{1}{|D|} \sum_{(x,y)\in D} h(|\langle \mathbf{I}_m, B_{x,y,s,\alpha}\rangle|^2)$$

be the spatially pooled average at orientation α. We can fit a model $q_{m,\alpha}(c) = p(c; \lambda_{m,\alpha})$ to match $h_\alpha(\mathbf{I}_m)$. The max-

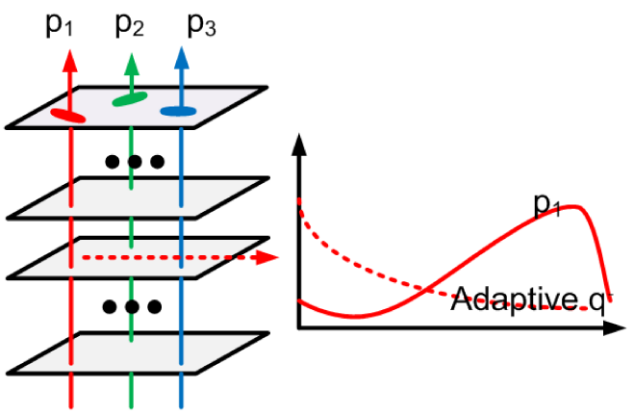


Fig. 8 For each image $\mathbf{I}_m$, at each orientation, an adaptive q is pooled from the Gabor filter responses at all the pixels in this image. Such adaptive q's capture texture information in image $\mathbf{I}_m$. Each p_i is paired with an adaptive q at the orientation that is the same as $B_{m,i}$

imum likelihood estimate $\hat{\lambda}_{m,\alpha} = \mu^{-1}(h_\alpha(\mathbf{I}_m))$. Then we compute the log-likelihood ratio score or SUM2 score by

$$\begin{aligned}
&\text{SUM2}_m \\
&= \sum_{i=1}^{n} \log \frac{p(c_{m,i}; \hat{\lambda}_i)}{p(c_{m,i}; \hat{\lambda}_{m,\alpha_i})} \\
&= \sum_{i=1}^{n} \{[\hat{\lambda}_i h(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2) - \log Z(\hat{\lambda}_i)] \\
&\quad - [\hat{\lambda}_{m,\alpha_i} h(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2) - \log Z(\hat{\lambda}_{m,\alpha_i})]\}, \qquad (30)
\end{aligned}$$

where α_i is the orientation of $B_{m,i}$. We can also let α_i be the orientation of B_i, which is what we did in Experiment 4. Experiments on classification suggest that the score (30) has a slight advantage over the original score (29).

3.7 Active Mean Vector and Active Correlation

We can replace the log-likelihood score $\log[p(\mathbf{I}_m \mid \mathbf{B}_m)/q(\mathbf{I}_m)]$ in (29) by the correlation between $\mathbf{I}_m$ and the vector $V_m = \sum_{i=1}^{n} \theta_i B_{m,i}$, which is defined as

$$\langle \mathbf{I}_m | V_m \rangle = \sum_{i=1}^{n} \theta_i \text{whiten}(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2)^{1/2}. \qquad (31)$$

We assume that $\mathbf{I}_m$ is normalized. The reason we use whitening transformation defined by (12) is that after such a transformation, the distribution of the natural images is closer to white noise. Geometrically, the white noise distribution is close to the uniform distribution over a high dimensional sphere. Image patches (after normalization and whitening transformation) from the same object category form a cluster on this sphere. Such a simple picture makes the concept of correlation geometrically meaningful. The correlation score (31) can be considered the length that $\mathbf{I}_m$ projects on V_m. In (31), we filter out the local phase information, because phase is irrelevant for shape. We call (31) the active correlation between $\mathbf{I}_m$ and the vector $V = \sum_{i=1}^{n} \theta_i B_i$, because we perturb V to V_m in order to best correlate it with $\mathbf{I}_m$.

For the training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$, we want to find the vector $V = \sum_{i=1}^{n} \theta_i B_i$ that best correlates with $\{\mathbf{I}_m, m = 1, \ldots, M\}$, by maximizing the sum of the active correlation scores:

$$\begin{aligned}
&\sum_{i=1}^{m} \langle \mathbf{I}_m \mid V_m \approx V \rangle \\
&= \sum_{i=1}^{n} \left[\theta_i \sum_{m=1}^{M} \text{whiten}(|\langle \mathbf{I}_m, B_{m,i}\rangle|^2)^{1/2} \right]. \qquad (32)
\end{aligned}$$

The algorithm for learning $\mathbf{B} = (B_i, i = 1, \ldots, n)$ and $\Theta = (\theta_i, i = 1, \ldots, n)$ is described in Sect. 2.3. The resulting $V = \sum_{i=1}^{n} \theta_i B_i$ can be consider the mean shape of $\{\mathbf{I}_m, m = 1, \ldots, M\}$. We call it the active mean vector. Geometrically, V points to the center of the cluster formed by $\{\mathbf{I}_m, m = 1, \ldots, M\}$. The active mean vector is a non-linear average that involves dimension reduction and perturbation of basis elements. It can be used in the K-mean clustering as we shall show in Sect. 6.1.

4 Supervised Learning, Detection, and Classification

This section applies the learning and inference algorithms to supervised learning, detection, and classification.

4.1 Learning with Given Bounding Boxes

In supervised learning, we assume that the training images are defined on the same image lattice which is the bounding box of the objects in these images.

In the experiments in this article, we hand pick the number of basis elements, n. In principle, it can be automatically determined by comparing $\sum_{m=1}^{M} h(r_{m,i})/M$ with the average of $h(\text{MAX1}(x, y, s, \alpha))$ in natural images or in the observed image $\mathbf{I}_m$, or equivalently, by enforcing a lower bound on the estimated λ_i, so that if λ_i is below this lower bound, we then stop the algorithm. As suggested by the experiments on classification, the choice of n is not critical.

We also hand pick the resize factor of the training images. Of course, in each experiment, the same resize factor is applied to all the training images.

Parameter values. The following are the parameter values that we use in all the experiments in this paper (unless otherwise stated). Length of Gabor wavelets = 17. In some experiments, we also combine templates of Gabor wavelets at different scales. (x, y) is sub-sampled every 2 pixels or 1 pixel. The sub-sampling rate for experiments in all the subsections of this Sect. 4 is 2. The orientation α takes $A = 15$ equally spaced angles in $[0, \pi]$. The orthogonality tolerance is $\epsilon = .1$. The threshold $T = 16$ in the threshold transformation (13). The saturation level $\xi = 6$ in the sigmoid transformation (11). The shift along the normal direction $d_{m,i} \in [-b_1, b_1] = [-6, 6]$ pixels. In some experiments, we also make this range smaller, such as $b_1 = 3$ or 2 pixels. The shift of orientation $\delta_{m,i} \in [-b_2, b_2] = \{-1, 0, 1\} \times \pi/15$.

Experiment 1. In Experiment 1.1, we take $h(r) = \text{threshold}(r)$, as defined by (13), so that there is no need to pool $q(r)$. This simple choice was used in our ICCV paper (Wu et al. 2007). We apply the shared sketch algorithm to a training set of $M = 37$ car images. The car images are 82×164 (height $\times$ length). Figure 9 displays the results, where $n = 60$. The first block displays the learned active basis $\mathbf{B} = \{B_i, i = 1, \ldots, n = 60\}$, where each B_i is represented symbolically by a bar at the same

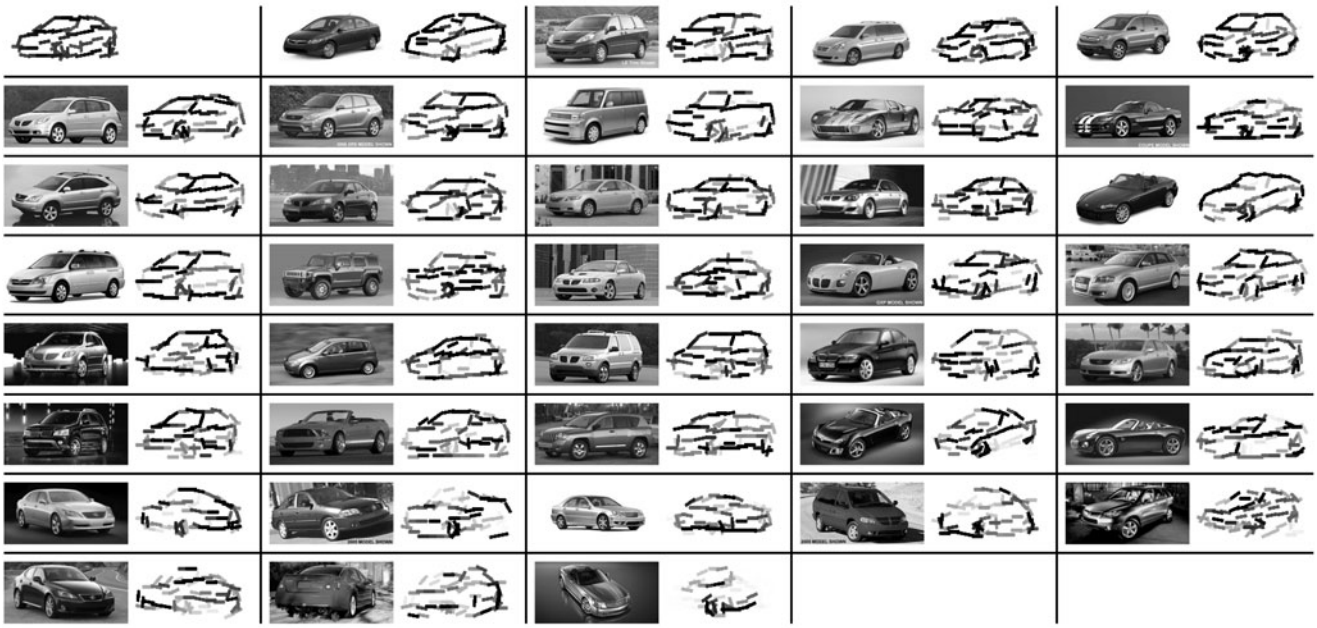


Fig. 9 Experiment 1.1. The 37 training images are 82 × 164 (height × length). The first block displays the learned active basis consisting of 60 elements. Each element is symbolized by a *bar*. The rest of the blocks display the observed images and the corresponding deformed active bases. The images are displayed in the descending order of the log-likelihood ratio, which scores the template matching

Fig. 10 Experiment 1.2. The 15 images are 179 × 112. Number of elements is 50. $h()$ is sigmoid transformation

location and with the same length and orientation as B_i. The intensity of the bar that symbolizes B_i is the average $\sum_{m=1}^{M} h(\text{MAX1}_m(x_i, y_i, s, \alpha_i))/M$. For the remaining M pairs of plots, the left plot shows $\mathbf{I}_m$, and the right plot shows $\mathbf{B}_m = (B_{m,i}, i = 1, \ldots, n)$. The intensity of the bar that symbolizes $B_{m,i}$ is the squared root of $h(|\langle \mathbf{I}_m, B_{m,i} \rangle|^2)$. These M examples are arranged in descending order by the SUM2_m scores output by the algorithm. We can see that all the examples with non-typical poses are in the lower end.

Figures 10–13 display more examples, where the results are obtained by the same algorithm. Different choices of $h()$ and different normalization schemes produce similar results.

Negative experience in Experiment 1. This experiment requires that the training images are roughly aligned and the objects are in the same pose. If this is not the case, our method cannot learn clean templates. Also, our method does not do well on objects with strong textures, such as zebras, leopards, tigers, giraffes, etc. The learning algorithm tends to sketch edges in textures.

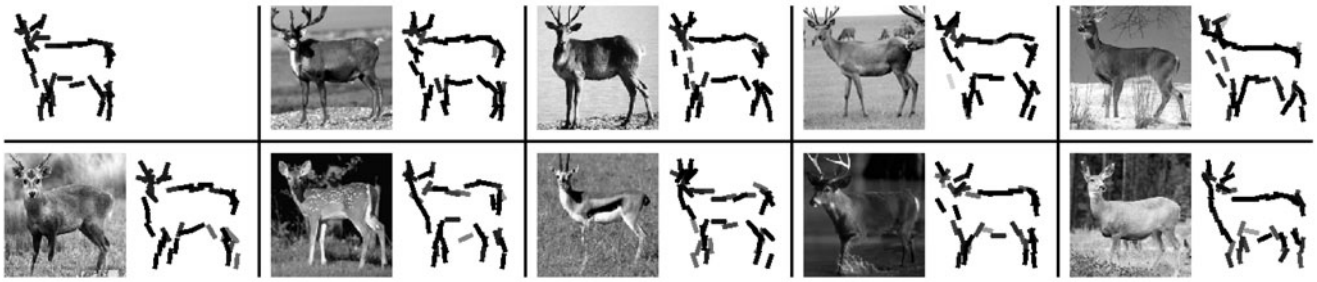


Fig. 11 Experiment 1.3. The 9 images are 122×120. Number of elements is 40. $h()$ is sigmoid transformation. The filter responses are normalized locally within a 20×20 window. See Sect. 3.5 for a discussion of normalization schemes

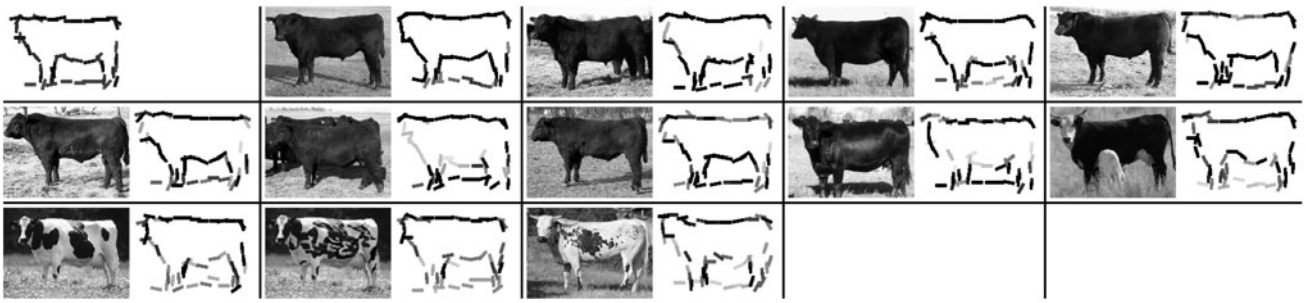
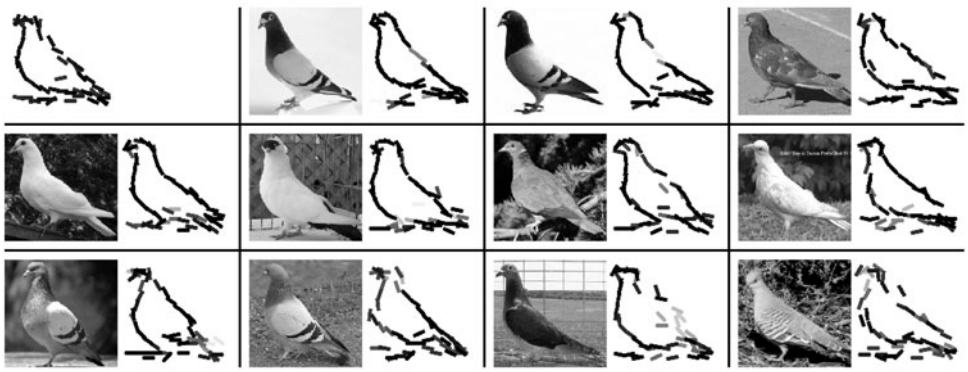


Fig. 12 Experiment 1.4. The 12 images are 120×167. Number of elements is 50. $h()$ is threshold transformation

Fig. 13 Experiment 1.5. The 11 images are 133×140. Number of elements is 50. $h()$ is sigmoid transformation



In Sect. 5, we shall show that our method can be extended to learning from non-aligned images. In Sect. 6, we shall show that our method can be used to find clusters in training images.

4.2 Detection by Inference Algorithm

This section studies the detection task using the inference algorithm based on sum-max maps.

Experiment 2. In Experiment 2.1, we learn the template from training images in Experiment 1.1, with $h()$ being the sigmoid transformation. Figure 14(a) displays the learned template. The bounding box is 82×164. Then we use the learned template to detect the car in the testing image, which is shown in Fig. 14(b). We run the inference algorithm over 15 resolutions of the testing image, from 50×67 to 751×1001. Figure 14(c) displays the superposed sketch of $(B_{\hat{x}_i,\hat{y}_i,s,\hat{\alpha}_i}, i = 1, \ldots, n = 60)$ at the optimal resolution.

In the inference algorithm, the filter responses are normalized by the average response within the 82×164 sliding window of the template. To handle flat regions in sky and ground where the average responses are very small, we enforce a lower bound on the averages, which is 1% of the maximal average within the testing image. When scanning the template over the image, we may allow the template to be partially outside the image. We only need to set the filter responses of those elements that are outside the image to be 0.

Figure 15 displays the MAX2 scores over the 15 resolutions, as well as the SUM2 map at the optimal resolution that achieves the maximum MAX2 score over these 15 resolutions.

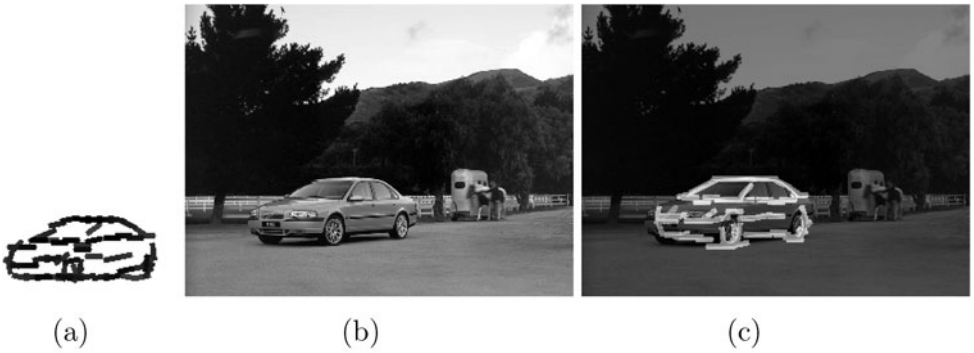


Fig. 14 Experiment 2.1. (**a**) Template learned from training images in Experiment 1.1. $h()$ is the sigmoid transformation. Size of template is 82×164. (**b**) Testing image. The inference algorithm is run over 15 resolutions, from 50×67 to 751×1001. (**c**) Superposed with sketch of the 60 elements of the deformed active basis at the optimal resolution and location

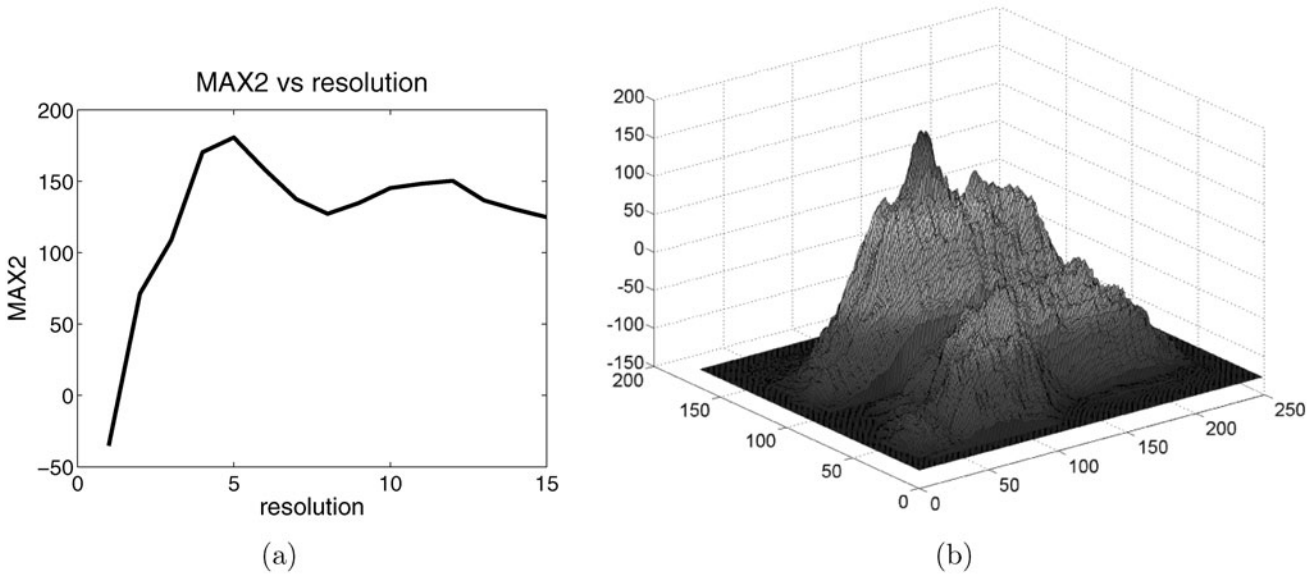


Fig. 15 Experiment 2.1. (**a**) MAX2 scores at resolutions 1 to 15. (**b**) SUM2 map at the optimal resolution

Fig. 16 Experiment 2.2. Testing image

Figure 16 displays the observed image in Experiment 2.2. The deformable template is learned in Experiment 1.2. The bounding box is 179×112. We run the inference algorithm on 10 resolutions of the testing image, from 150×110 to 286×190.

Figure 17 displays the superposed sketch at each of the 10 resolutions. Figure 18 displays the MAX2 scores over the 10 resolutions. There are two peaks, corresponding to the two human figures in the testing image.

In Experiment 2.3, we learn templates using Gabor wavelets of 5 different scales from the training images in Experiment 1.3, and then combine them for detection. The lengths of the Gabor wavelets at these 5 scales are 17, 25, 33, 39, 49 respectively. Figure 19 displays the 5 templates. The number of elements at the lowest scale is 40. The numbers of elements at other scales are inverse proportional to the corresponding scales. The filter responses are normalized within the whole templates.

For each template, we apply the inference algorithm over 15 resolutions of the testing image, which is shown in Fig. 20. We then combine these 5 templates by summing over their SUM2 maps. The MAX2 score is computed from this combined SUM2 map.

Fig. 17 Experiment 2.2. Superposed sketch of 50 elements of the deformed active basis at each of the 10 resolutions, from 150×110 to 286×190. The bounding box is 179×112

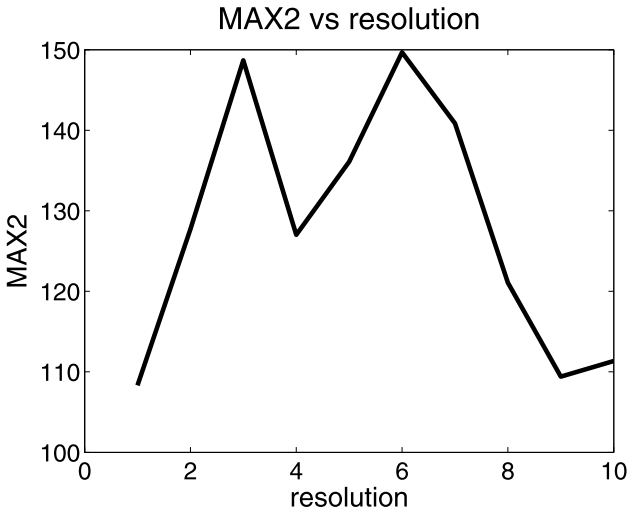


Fig. 18 Experiment 2.2. MAX2 scores at resolutions 1 to 10

Fig. 19 Experiment 2.3. Learned templates using Gabor wavelets of lengths 17, 25, 33, 39, 49 respectively

Fig. 20 Experiment 2.3. Testing image. For each template, we run the inference algorithm over 15 resolutions, from 110×140 to 341×434

Figure 21 displays the superposed templates of the 5 scales, at the detected location and resolution of the testing image.

Figure 22(a) displays the MAX2 scores over the 15 resolutions. (b) displays the combined SUM2 map at the optimal resolution. The combined SUM2 map is the sum of the SUM2 maps of the 5 templates.

Computationally, applying a larger Gabor filter to an image is the same as applying a smaller Gabor filter to a lower resolution of the same image, although the former may have more numerical precision. In Experiment 2.3, we have not eliminated such a computational redundancy. We use multi-scale Gabor wavelets and meanwhile we also search over multiple resolutions of the testing image.

Negative experience in Experiment 2. Our method can sometimes be distracted by cluttered edges or strong edges in the background. One may need to incorporate local appearance variables such as textures and smoothness into the model.

4.3 Geometric Transformation of Template

Given a template $\mathbf{B} = (B_i = B_{x_i, y_i, s, \alpha_i}, i = 1, \ldots, n)$, we can transform this template by dilation, rotation, and changing the aspect ratio. This amounts to simple transformations of $(x_i, y_i, \alpha_i, i = 1, \ldots, n)$.

Fig. 21 Experiment 2.3. Superposed with templates of 5 scales, at detected resolution and location

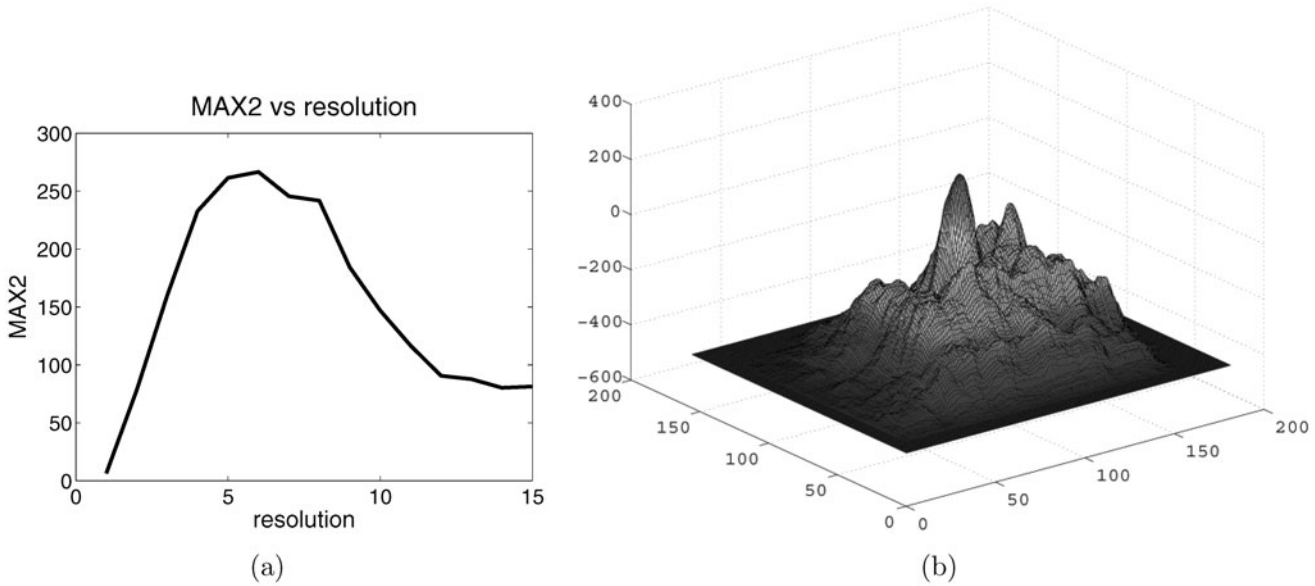


Fig. 22 Experiment 2.3. (**a**) MAX2 scores at resolutions 1 to 15. (**b**) Combined SUM2 map at the optimal resolution

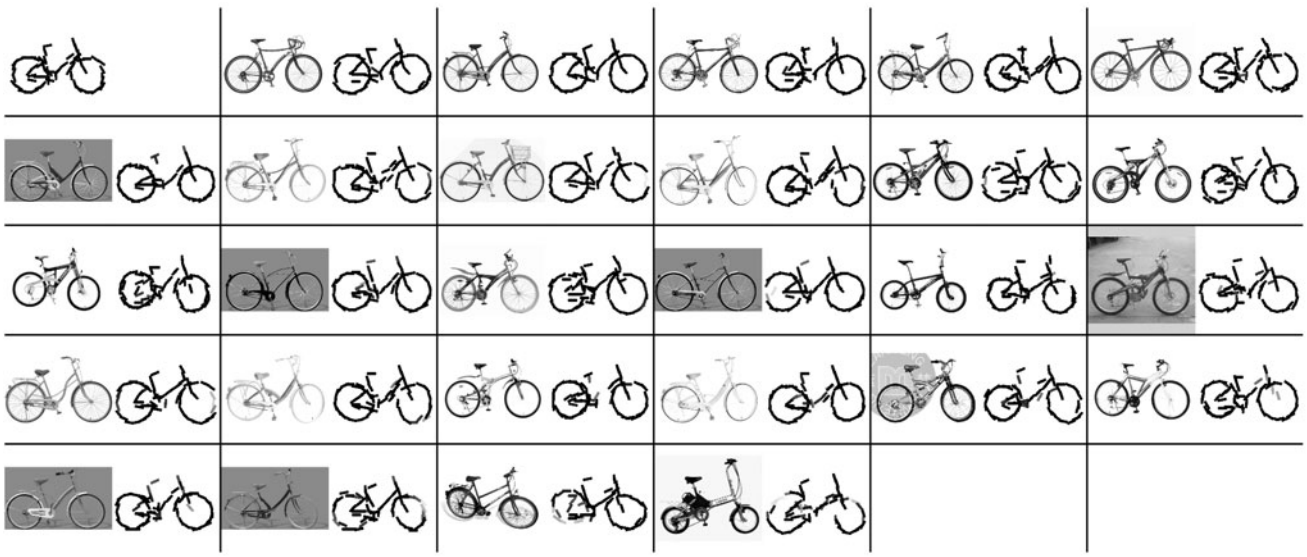


Fig. 23 Experiment 3.1. The 27 images are 180 × 180. Number of elements is 60

Experiment 3. Figure 23 displays the bike template learned from 27 images, using the active basis model with sigmoid transformation.

Figure 24 shows three examples of detection. We transform the template into a collection of templates at different scales, orientations, and aspect ratios. After that, we use these templates to detect the object by the inference algorithm, using the sum-max maps. We do not need to try multiple resolutions, because we already scale the template. Finally, we choose the transformed template that gives the best

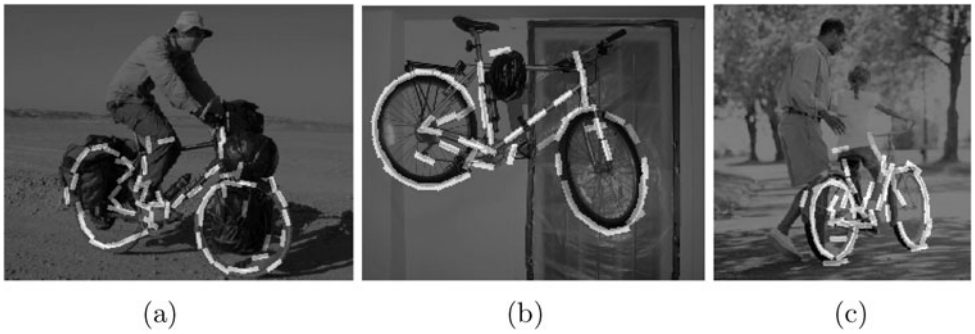


Fig. 24 Experiment 3.1. (**a**) The image size is 252 × 320. The scale factor is 1.4. The rotation is $1 \times \pi/15$. The aspect factor is 0.9. (**b**) The image size is 200 × 250. The scale factor is 1.4. The rotation is $1 \times \pi/15$. The aspect factor is 1. (**c**) The image size is 248 × 232. The scale factor is 1.2. The rotation is $-1 \times \pi/15$. The aspect factor is 0.6

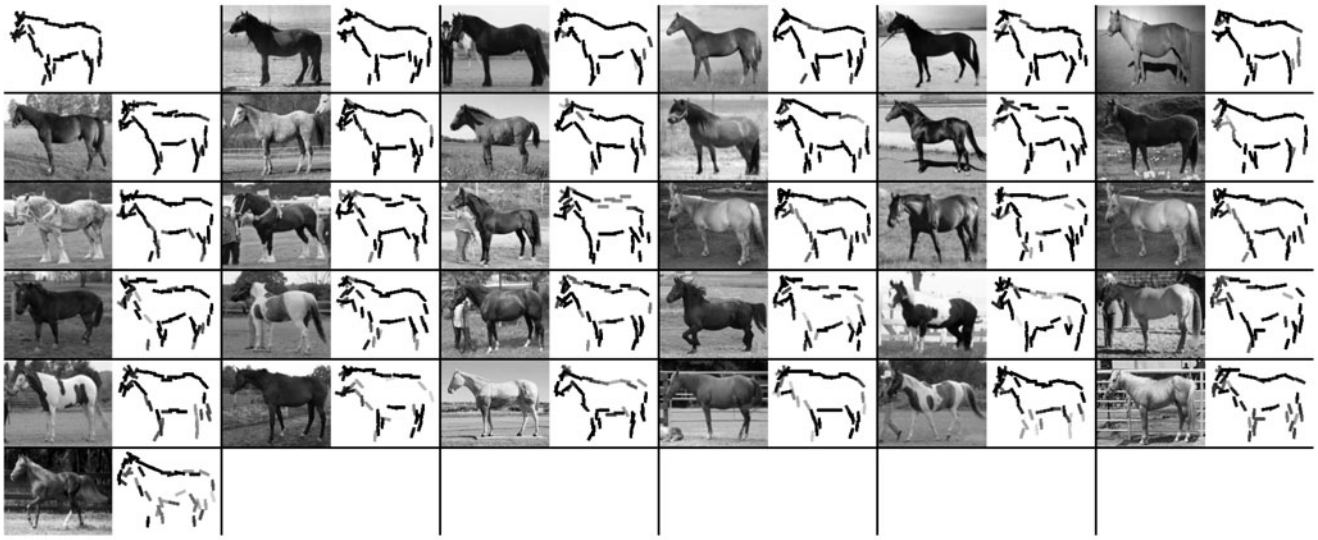


Fig. 25 Experiment 3.2. The 30 images are 120 × 150. Number of elements is 40

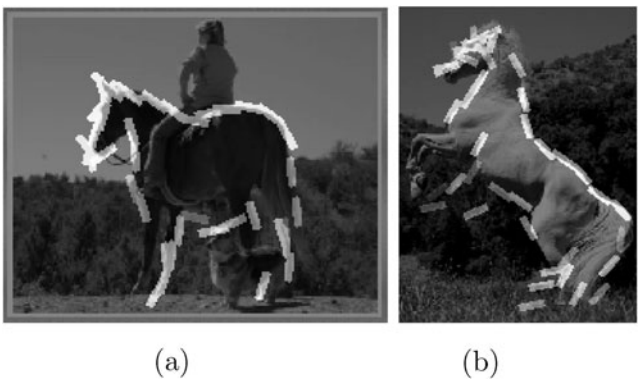


Fig. 26 Experiment 3.2. (**a**) The image size is 166 × 202. The scale factor is 1.2. The rotation is 0. The aspect factor is 0.8. (**b**) The image size is 192 × 144. The scale factor is 1. The rotation is $4 \times \pi/15$. The aspect factor is 1.4

match in terms of the MAX2 score, and superpose the deformed template on the input image.

Figures 25 and 26 show another example with the horse template.

Negative experience in Experiment 3. We encountered some difficulty with the bicycle template. When the viewing distance is close, the size of one wheel can be larger than the size of the other wheel, so a single scale factor does not give a very good fit. An additional difficulty is caused by the fact that the frontal wheel may turn to a different direction than the back wheel.

The above difficulty suggests that we should better split the bicycle template into two part-templates, and allow each part-template to have its own geometric transformation. We shall explore the composition of multiple part-templates in Sect. 8.

4.4 Classification

In this section, we evaluate our method on classification tasks and compare it with adaboost (Freund and Schapire 1997; Viola and Jones 2004) and PCA in terms of the areas under the ROC curves, or the AUC scores.

We learn the active basis $\mathbf{B} = (B_i, i = 1, \ldots, n)$ and estimate $\Lambda = (\lambda_i, i = 1, \ldots, n)$ from the training images. Then for each testing image $\mathbf{I}_m$, we compute its score SUM2_m

according to (29) or (30). The latter scores the template against the adaptive background. The testing step is accomplished by the inference algorithm. We fit the active basis model with sigmoid transformation. The parameter values are taken to be their default values specified in Sect. 4.1. The sigmoid transformation outperforms whitening and threshold transformations.

Experiment 4. We conduct cross validation experiments with 131 positive images of heads and shoulders, and 600 + negative images. The image size is 85 × 127. In total, there are 5 repetitions × 3 methods × 5 numbers of positive training examples (5, 10, 20, 40, 80). The number of negative training examples is kept at 160. We pool $q(r)$ from the negative training images for learning the active basis. The learning does not require negative images beyond this one-dimensional marginal histogram.

Figure 27 plots the AUC scores against the numbers of positive training examples. The vertical bars represent the 90% confidence intervals estimated from the 5 repetitions based on t-statistics. The number of basis elements in active basis is 40.

The adaboost features are obtained by thresholding the MAX1 maps, i.e., $1_{\text{MAX1}(x,y,s,\alpha)>c}$ or $1_{\text{MAX1}(x,y,s,\alpha)<c}$. For each (x, y, α), an optimal threshold c is searched over a grid of 50 equally spaced points from the minimum to the maximum of $\{\text{MAX1}_m(x, y, s, \alpha)\}$. The number of adaboost features is 80. In conducting this experiment, we noticed an issue in our previous implementation of adaboost (Wu et al. 2007), where the threshold for each basis element is pre-trained on the training examples with the uniform weights, and the adaboost only selects the basis elements. While this might not be unfair because the $h()$ function in active basis learning is fixed beforehand, in the current implementation, the thresholds are trained during the adaboost iterations on re-weighted examples, and the adaboost is started from balanced uniform weights, i.e., the total weights of positives and negatives are both $1/2$.

For PCA, we first normalize each image to have marginal mean 0 and marginal variance 1. Then we estimate the mean image and the principal components from all the positive training images. In testing, we fit the learned mean image and the principal components to each testing image, and score the image by the squared norm of the residual image. As for the number of principal components, we use the first two components, which gives good performance among different choices. We may let the number of components increase along with the increase of the sample size, but there seems to be not much hope that PCA can be competitive with the other two methods. One needs to extend it to active appearance model (Cootes et al. 2001) to explicitly account for shape deformations in order to achieve competitive performance.

Number of basis elements. Figure 28 plots the AUC scores of active basis with adaptive background over the numbers of basis elements, where the numbers of training examples are 5 and 40 respectively. The optimal performance is attained around 30 elements. The performance does not change much if we continue to increase the number of elements.

Figure 29 shows the active basis template and adaboost template sketched by the first 30 elements, as well as the mean image and the two principal components. They are all learned from the same 40 positive training images during one repetition of the cross validation experiment. For adaboost templates, most features are of the type $1_{\text{MAX1}(x,y,s,\alpha)>c}$.

As a further illustration, Fig. 30 displays the active basis learning results from the first 5 positive images.

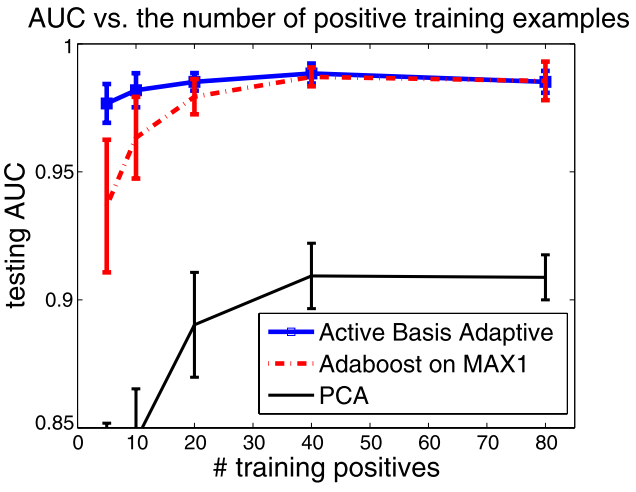


Fig. 27 Experiment 4.1. AUC scores over the number of positive training examples for active basis with adaptive background, adaboost, and PCA. The vertical bars are 90% confidence intervals

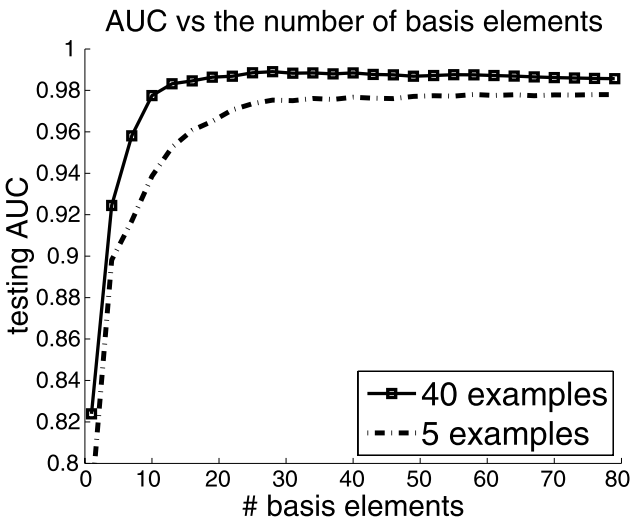


Fig. 28 Experiment 4.1. AUC scores of active basis with adaptive background versus the number of elements

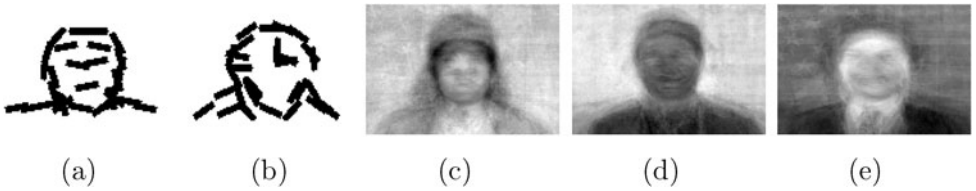


Fig. 29 Experiment 4.1. Learned from the same training set with 40 positive examples. (**a**) Active basis template (the first 30 basis elements). (**b**) Adaboost template (the first 30 elements). (**c**) Mean positive image. (**d**) and (**e**) The first two principal components

Fig. 30 Experiment 4.1. Learning active basis from the first 5 training images. The images are 85×127. Number of elements is 30

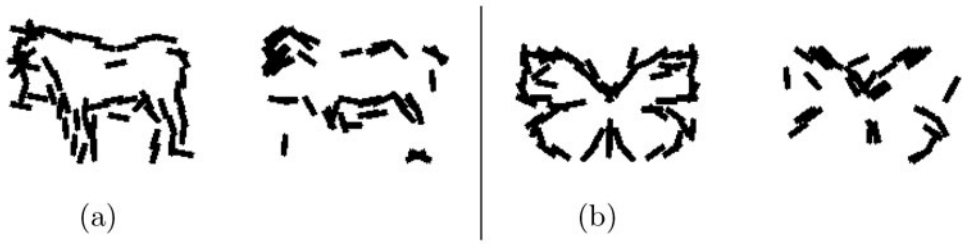


Fig. 31 Experiments 4.2 and 4.3. Active basis template (*left*) and adaboost template (*right*), with 60 elements, learned from the same training set with 20 positive examples. (**a**) Horses data set. (**b**) Butterflies data set

Fig. 32 Experiment 4.3. Learning active basis from the first 9 training images. The images are 100×150. Number of elements is 50

We conduct the same experiments on a horse data set and a butterfly date set. Figure 31 displays the active basis templates and adaboost templates learned from the same training sets. Figure 32 displays the active basis learned from the first 9 positive images of butterflies.

Experiment 4 suggests that the active basis model is comparable to adaboost in classification when the sample size is relatively small, despite the fact that it maximizes the log-likelihood ratio instead of the classification margin. The active basis model does not need negative examples beyond pooling a marginal histogram. In fact, the selection of the basis elements $\mathbf{B} = (B_i, i = 1, \ldots, n)$ does not require negative examples at all. The marginal histogram is only used for estimating the weight vector $\Lambda = (\lambda_i, i = 1, \ldots, n)$.

For generative model to stay competitive with the discriminative approach when the sample size is very large, we may need to represent the positive training set by a mixture of multiple prototypes, as in Amit and Trouve (2007). See also Sect. 6.2 on learning prototype templates.

We still do not understand the relationship between generative and discriminative learning, either empirically or theoretically. The generative learning maximizes the likelihood ratio, and tends to focus on typical positive examples inside the classification boundary. The discriminative learning maximizes the class probability or margin, and tends to focus on marginal examples that are close to classification boundary. In the pursuit of basis elements, the generative learning does not re-weight positive examples, and the inhibition between basis elements is carried out through residual images. In the discriminative learning, the inhibition is done by re-weighting the training examples. It is unclear what the effect of the above-mentioned differences is on the learned templates. It is even more unclear how these differences play out in unsupervised learning, which involves inferring latent variables in training examples, or finding structures in the training set.

In what follows, we shall present some experiments which suggest that it is possible to learn the active basis model in the situations that are not fully supervised.

5 Learning from Non-aligned Images

In this section, we study the problem of learning from images where the objects are of unknown locations and scales.

5.1 Multiple Image Alignment

For the training image patches $\{\mathbf{I}_m, m = 1, \ldots, M\}$ defined on the same bounding box, such as those in the previous section, we can define the multiple alignment score (as compared to pairwise alignment) by

$$\text{ALIGN}(\mathbf{I}_m, m = 1, \ldots, M) = \sum_{m=1}^{M} \text{MATCH}(\mathbf{I}_m, \mathbf{B}), \quad (33)$$

where $\mathbf{B}$ is the template learned from the image patches, and $\text{MATCH}(\mathbf{I}_m, \mathbf{B})$ is the template matching score defined by either (14) for log-likelihood or (15) for active correlation. The computation is carried out by the shared sketch algorithm in Sect. 2.5, and $\text{ALIGN}(\mathbf{I}_m, m = 1, \ldots, M) = \sum_{m=1}^{M} \text{SUM2}_m$, where the SUM2_m scores are output by the algorithm.

When the training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$ are of different sizes, and the objects appear at different locations in the training images, we need to infer the unknown locations. Let $\text{box}(x, y)$ be the rectangular bounding box of the template centered at (x, y). For an image $\mathbf{I}$, let $\mathbf{I}[\text{box}(x, y)]$ be the image patch cropped from the image $\mathbf{I}$ within $\text{box}(x, y)$. We want to maximize the alignment score

$$\text{ALIGN}(\mathbf{I}_m[\text{box}(x_m, y_m)], m = 1, \ldots, M) \quad (34)$$

over $\{(x_m, y_m), m = 1, \ldots, M\}$, where (x_m, y_m) is the unknown location of the bounding box in $\mathbf{I}_m$.

The alignment score can be maximized by a greedy algorithm that iterates the following two steps:

(1) Supervised learning: Given $\{(x_m, y_m), m = 1, \ldots, M\}$, estimate $(\mathbf{B}, \Lambda)$ from $\{\mathbf{I}_m[\text{box}(x_m, y_m)], m = 1, \ldots, M\}$ using the shared sketch algorithm in Sect. 2.5.

(2) Detection: Given $(\mathbf{B}, \Lambda)$, estimate (x_m, y_m) from each $\mathbf{I}_m$ using the inference algorithm in Sect. 2.4. (x_m, y_m) achieves the maximum of the SUM2 map.

Experiment 5a. In this experiment, we initialize the algorithm by specifying the bounding box for the first training image. Then we estimate $(\mathbf{B}, \Lambda)$ from this single image patch. In learning from the single image patch, we set $b_1 = b_2 = 0$, that is, we do not allow the elements B_i to perturb. After that, we re-set b_1 and b_2 to their default values, and iterate Step (2) and Step (1) described above. In Experiment 5, with the exception of the horses example, the default value for b_1, i.e., the allowed range of displacement in location, is 3 pixels, and the sub-sampling rate is 1 pixel. For horses example, $b_1 = 6$ pixels, and the sub-sampling rate is 2 pixels. The default value for b_2, i.e., the allowed range of displacement in orientation, is $\pi/15$, as before. The reason we make b_1 smaller than in the previous experiments is that with the adjustment of the overall locations and scales of the objects, better alignment is expected to be achieved.

In Step (2), we search over 9 different resolutions, from 0.8 to 1.2 times the input image size (enlarging the range to 0.6 to 1.4 can still result in meaningful templates). We crop $\mathbf{I}_m[\text{box}(x_m, y_m)]$ from the optimal resolution.

We run the algorithm for 5 iterations. Figure 33 displays some examples.

We can also learn templates at different scales in the step of supervised learning, and combine them in the detection step. Figure 34 displays two examples of multi-scale templates.

Experiment 5b. This is a repetition of Experiment 5a, except that we do not assume that the bounding box of the object in the first image is given. We simply start from the template learned from the whole image of the first training image. In other words, we assume that the whole image lattice of the first image is the bounding box of the template. Figure 35 displays two examples. In each example, the first template is learned from the first image, and the template serves as the initialization of the algorithm. The second template is produced after 5 iterations of the algorithm used in Experiment 5a.

Negative experience in Experiments 5a and 5b. When there are cluttered edges in the background, the detection step may fail to locate the objects. When the objects have large deformations or pose changes, the learned template may not be clean, and may fail to sketch the objects in the training images correctly. In Experiments 5b, if the objects do not occupy significant portions of the training images, our method may fail to establish correct alignment.

5.2 Learning Part-templates

The algorithm in Experiment 5a can be used to learn part-templates from training images.

Experiment 5c. We start the learning algorithm from a large number of patches cropped from the training images, and for each starting patch, we learn a template using the same iterative algorithm as in Experiment 5a. The number of iterations is 3. Here we use active correlation (see Sect. 3.7) instead of the log-likelihood for learning and detection.

Then we select the first K templates with the highest alignment scores. We did not perform spatial inhibition between the part-templates. After that, we double the sizes of the input images, and use the same procedure to learn part-templates at a higher resolution.

Figure 36 displays the top three part-templates learned from three car images. Because of the large deformations in these three cars, it is impossible to learn a common template for the whole cars, but it is still possible to learn meaningful part templates that correspond to frontal, middle and rear parts of the cars.

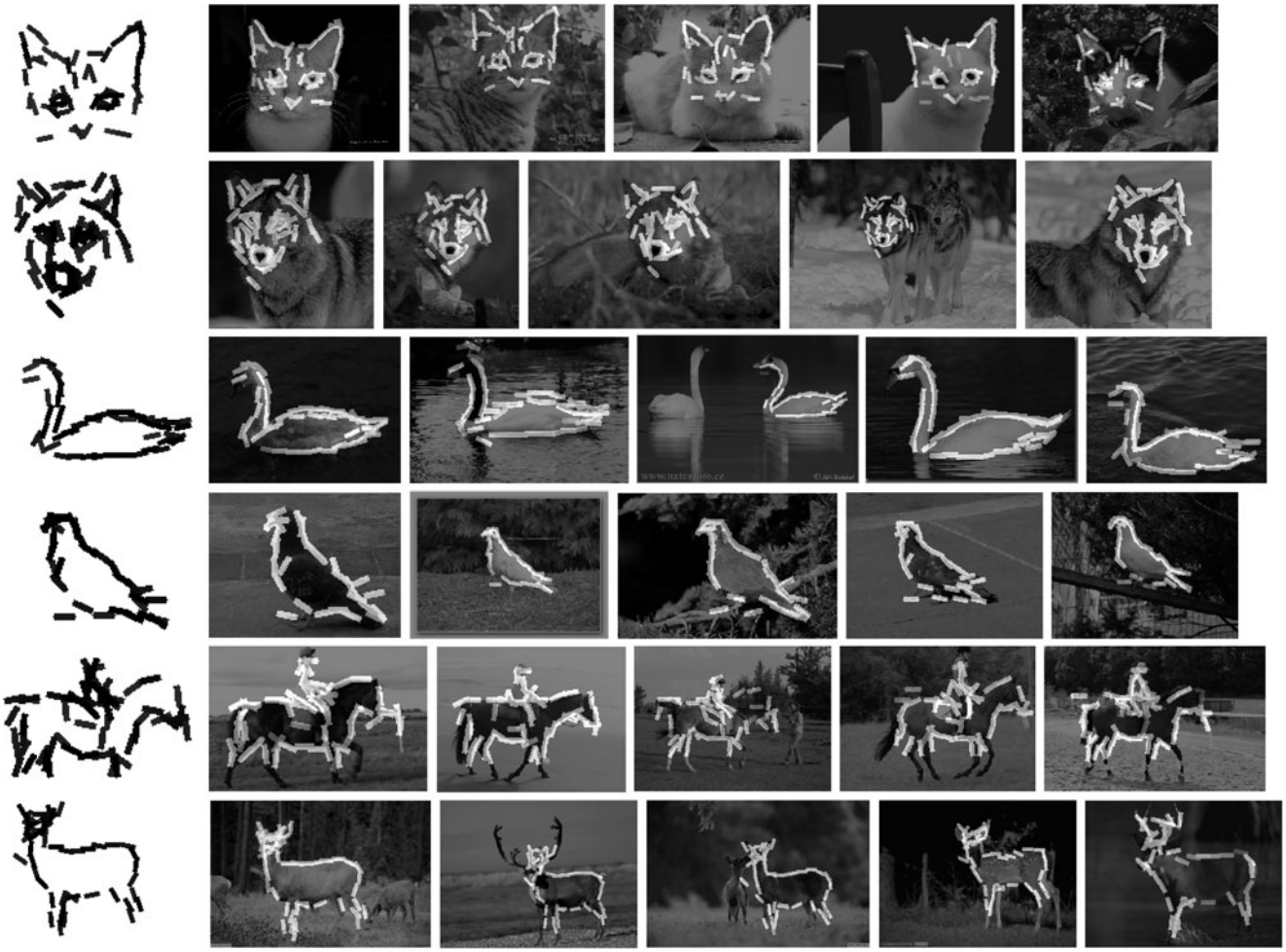


Fig. 33 Experiment 5a. The bounding box in the first image is given. The number of iteration is 5. With the exception of horses example, the allowed displacement in location is up to 3 pixels, and the sub-sampling rate is 1 pixel. For the horses example, the allowed displacement in location is up to 6 pixels, and the sub-sampling rate is 2 pixels. (1) Cats: The size of the bounding box is 136 × 140. The number of elements in the active basis is 60. (2) Wolves: The bounding box is 117 × 117. Number of elements is 60. (3) Swans: The bounding box is 129 × 178. Number of elements is 50. (4) Pigeons: The bounding box is 103 × 129. Number of elements is 30. (5) Horses: The bounding box is 103 × 158. Number of elements is 60. (6) Deers: The bounding box is 143 × 149. Number of elements is 50

Fig. 34 Experiment 5a. Multi-scale templates. The lengths of the Gabor wavelets are 17, 25, 33, 39 respectively. Cat: Number of elements at the lowest scale is 60. The numbers of elements are inverse proportional to the scales. Swan: Number of elements at the lowest scale is 50

Figure 37 displays the top two part-templates learned from these three images after we resize these images by a factor of 2.

Negative experience in Experiment 5c. When the part-template is small relative to the whole objects, the method often fails to establish correct correspondence among the images.

The above difficulty suggests that we should add constraints for more reliable learning of the parts. If the bounding boxes are given as in Experiment 1, we can restrict the ranges of movements of parts in the training images, so that in the detection step, we do not need to search over the whole images. If the bounding boxes are not given, we might simultaneously learn multiple parts while restricting their relative positions. We leave it to future investigations.

5.3 Learning Moving Template from Motion Sequence

Our method can also be used to learn a moving deformable template from a video sequence. Let $(\mathbf{I}_t, t = 1, \ldots, M)$ be

Fig. 35 Experiment 5b. In each example, the first template is the starting template. The second template is learned after 5 iterations. The number of elements of the active basis is 30 in the left example, and 60 in the right example

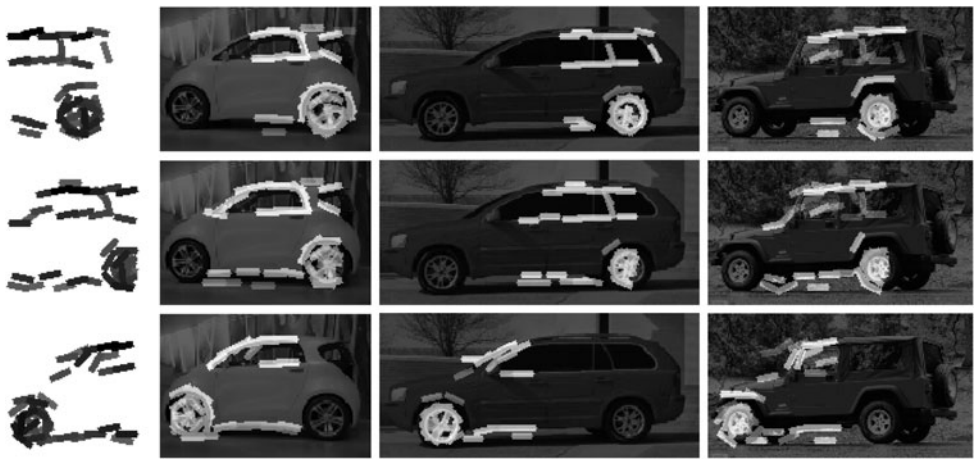


Fig. 36 Experiment 5c. The top three part-templates. The size of the bounding box is 100×100. The number of elements is 40. The allowed activity in location is up to 3 pixels. The allowed activity in orientation is up to $\pi/15$, as usual. The number of iterations is 3

Fig. 37 Experiment 5c. The top two part-templates learned after the sizes of the input images are doubled. The parameters are the same as in Fig. 36

a sequence of frames of an object shape that is moving at a speed $v = (v_x, v_y)$. We can estimate v and learn a template of the object shape simultaneously.

At the true speed $v = (v_x, v_y)$, let $\mathbf{J}_t^{(v)}(x, y) = \mathbf{I}_t(x + v_x t, y + v_y t)$, i.e., for frame t, we shift the image lattice back by vt, then the object shapes in $\{\mathbf{J}_t^{(v)}, t = 1, \ldots, M\}$ will be well aligned. If we apply the shared sketch algorithm to $\{\mathbf{J}_t^{(v)}\}$, we shall learn a clean template that has a high alignment score. We can try all possible v, and choose the v that achieves the maximum alignment score, i.e., we maximize

$$\text{ALIGN}(\mathbf{J}_t^{(v)}, t = 1, \ldots, M) \tag{35}$$

over v. This is actually a simpler problem than learning from non-aligned images.

In our experiment, we use active correlation (see Sect. 3.7) to evaluate the alignment score (35). Before computing this score, we need to perform background subtraction. First, we compute the background SUM1: $\text{SUM1}_0(x, y, s, \alpha) = \sum_{t=1}^{M} \text{SUM1}_t(x, y, s, \alpha)/M$. Then we modify $\text{SUM1}_t(x, y, s, \alpha) \leftarrow [\text{SUM1}_t(x, y, s, \alpha) - \text{SUM1}_0(x, y, s, \alpha)]_+$, where $[r]_+ = r$ if $r > 0$ and $[r]_+ = 0$ otherwise. For each v, we compute the alignment score of the background subtracted SUM1 maps using the shared sketch algorithm.

Experiment 6. We learn the moving template from a sequence of 19 frames of size 204×258. The image sequence is cropped from the PETS 2006 benchmark data (Ferryman 2006). We try 5 different directions v_x/v_y, and at each direction, we try 7 different speeds. Figure 38 displays the alignment scores at different speeds of the optimal direction.

Figure 39 displays the learned template and the superposed sketch for each frame at the optimal speed and direction.

Figure 40 displays another example.

6 Clustering and Local Learning

In this section, we study the problem of clustering, where we need to learn multiple templates from the training set, which is a mixture of different poses or different categories.

Unlike conventional clustering problem, we not only need to separate the examples into different clusters, but we also need to learn the active basis for each cluster, i.e., find the dimensions that characterize each cluster. These two tasks can be naturally integrated, and they actually depend on each other.

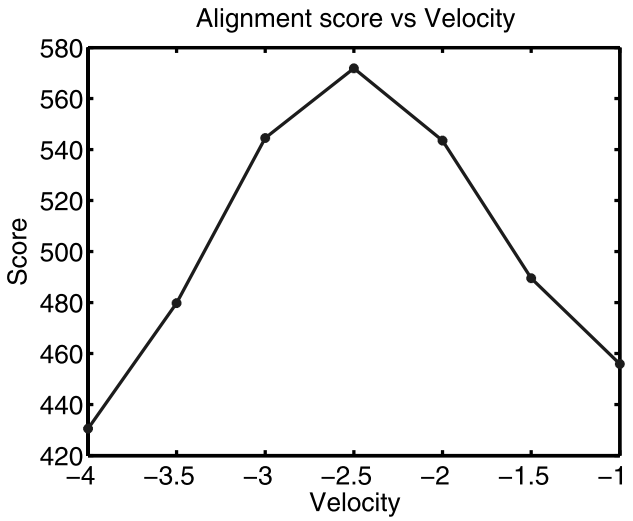


Fig. 38 Experiment 6.1. Alignment scores at different speeds of the optimal direction

6.1 EM and K-mean

Mixture model and EM. Suppose there are K clusters, and each cluster k can be described by an active basis model $\mathbf{B}^{(k)} = (B_i^{(k)}, i = 1, \ldots, n)$ and $\Lambda^{(k)} = (\lambda_i^{(k)}, i = 1, \ldots, n)$. Let $\rho^{(k)}$ be the probability that a training image $\mathbf{I}_m$ comes from cluster k, $k = 1, \ldots, K$. So $\mathbf{I}_m \sim \sum_{k=1}^{K} \rho^{(k)} p^{(k)}(\mathbf{I}_m \mid \mathbf{B}_m^{(k)})$, i.e., a mixture distribution, where each $p^{(k)}(\mathbf{I}_m \mid \mathbf{B}_m^{(k)})$ is modeled as in Sect. 3.1.

We can learn $\{\rho^{(k)}, \mathbf{B}^{(k)}, \Lambda^{(k)}, k = 1, \ldots, K\}$ by the EM algorithm (Dempster et al. 1977). For each image $\mathbf{I}_m$, we define $(z_m^{(k)}, k = 1, \ldots, K)$ as an indictor vector, where $z_m^{(k)} = 1$ if $\mathbf{I}_m$ comes from cluster k, otherwise $z_m^{(k)} = 0$.

E-step. For each $m = 1, \ldots, M$ and $k = 1, \ldots, K$, we impute

$$z_m^{(k)} = \frac{\rho^{(k)} \exp\{\mathrm{SUM2}_m^{(k)}\}}{\sum_{k=1}^{K} \rho^{(k)} \exp\{\mathrm{SUM2}_m^{(k)}\}}.$$

This is a soft classification based on the current models of the clusters, where each $z_m^{(k)}$ becomes a fraction. The $\mathrm{SUM2}_m^{(k)}$ scores are obtained in the M-step.

M-step. For each $k = 1, \ldots, K$, we learn $\mathbf{B}^{(k)}$ and $\Lambda^{(k)}$ according to the shared sketch algorithm in Sect. 2.5. We only need to make the following changes to the original version of the learning algorithm.

(1) In Step 3, find (x_i, y_i, α_i) by maximizing $\sum_{m=1}^{M} z_m^{(k)} h(\mathrm{MAX1}_m(x, y, s, \alpha))$, which is a weighted sum.

(2) In Step 3, compute $\hat{\lambda}_i$ by

$$\hat{\lambda}_i = \mu^{-1}\left(\frac{\sum_{m=1}^{M} h(r_{m,i}) z_m^{(k)}}{\sum_{m=1}^{M} z_m^{(k)}}\right), \tag{36}$$

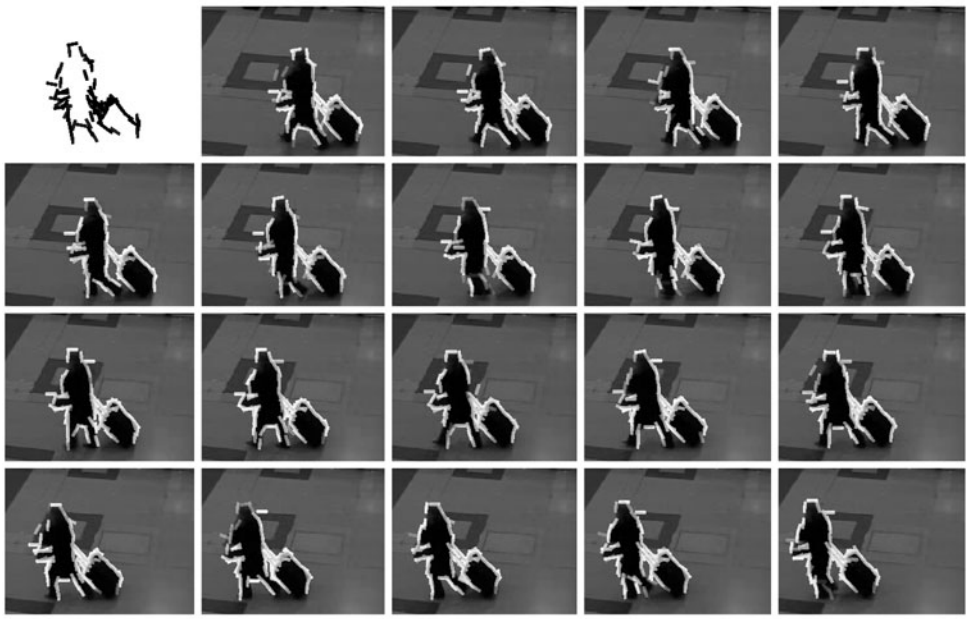


Fig. 39 Experiment 6.1. Learned template and superposed sketch for each frame at the optimal speed and direction. There are 19 frames of size 204 × 258, cropped from PETS 2006 benchmark data. Number of elements is 70

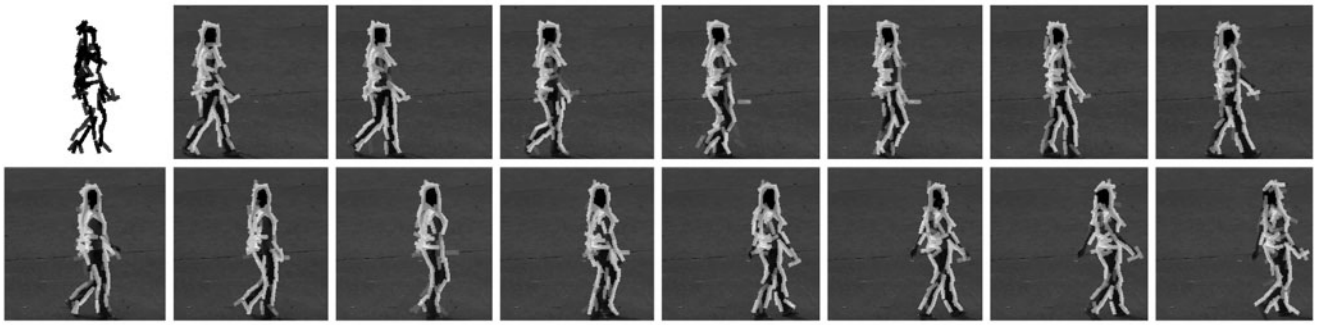


Fig. 40 Experiment 6.2. Learned template and superposed sketches at the optimal speed. The image frames are 180 × 186. Number of elements is 80

Fig. 41 Experiment 7.1. Learned templates from a mixture of 106 training images in Experiment 4. Image size is 120 × 150. Number of elements in each template is 40. Number of iteration is 4. (**a**) EM. (**b**) K-mean

that is, we match $\mu(\lambda_i)$ to the weighted average.

(3) At the end of the algorithm, attach a superscript (k) to the resulting SUM2_m and $\mathbf{B}$. $\text{SUM2}_m^{(k)}$ can then be used in the E-step.

We initialize the algorithm by randomly generating $\{z_m^{(k)}\}$, and then iterate the M-step and the E-step. We stop the algorithm after a few iterations. Then we classify $\mathbf{I}_m$ to the cluster k_* that maximizes $z_m^{(k)}$ over all $k = 1, \ldots, K$.

Experiment 7. Figure 41(a) displays the templates $\mathbf{B}^{(k)}$, $k = 1, 2, 3$, learned from the mixture of three subsets of positive images in Experiment 4. The EM algorithm can easily separate the three clusters.

Figure 42(a) displays the learned templates $\mathbf{B}^{(k)}, k = 1, 2$ from a mixture of images of horses facing different directions. The EM algorithm separates the two clusters.

K-mean clustering. Our K-mean clustering scheme is different than the conventional ones. The mean vector is not a simple average. It is obtained by the shared sketch algorithm. The distance is not simple Euclidean distance, but is defined in terms of active correlation.

We can pose the clustering problem as the following alignment problem: Find $\{(z_m^{(k)}, k = 1, \ldots, K), m = 1, \ldots, M\}$ to maximize

$$\sum_{k=1}^{K} \text{ALIGN}\big(\mathbf{I}_m, z_m^{(k)} = 1\big), \tag{37}$$

where $\text{ALIGN}(\mathbf{I}_m, z_m^{(k)} = 1)$ is the alignment score of the k-th cluster. See Sect. 5.1 for the definition of the alignment

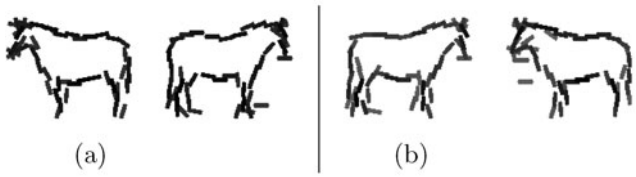


Fig. 42 Experiment 7.2. Learned templates for 57 images of horses facing two different directions. Image size is 120 × 150. Number of elements in each template is 50. (**a**) EM with 4 iterations. (**b**) K-mean with 8 iterations

score. Here $z_m^{(k)}$ are $0/1$ variables instead of fractions. The computation of the alignment score by the shared sketch algorithm also produces the template $\mathbf{B}^{(k)}$ for the k-th cluster. If we use active correlation to learn the template for each cluster and score the alignment within each cluster, then the learned $(\mathbf{B}^{(k)}, \Theta^{(k)})$ gives us an active mean vector $V^{(k)} = \sum_{i=1}^{n} \theta_i^{(k)} B_i^{(k)}$ for each cluster. The mean vector $V^{(k)}$ points to the center of the k-th cluster. The K-mean algorithm is a greedy scheme that maximizes (37), and it iterates the following two steps:

(1) Given $\{(z_m^{(k)}, k = 1, \ldots, K), m = 1, \ldots, M\}$, estimate the mean vector $(\mathbf{B}^{(k)}, \Theta^{(k)})$ from $\{\mathbf{I}_m, z_m^{(k)} = 1\}$ for each $k = 1, \ldots, K$.

(2) Given $\{\mathbf{B}^{(k)}, \Theta^{(k)}, k = 1, \ldots, K\}$, classify each image $\mathbf{I}_m$ to a cluster k_* that maximizes $\langle \mathbf{I}_m \mid V_m^{(k)} \rangle$ (see (31)) over all $k = 1, \ldots, K$, where $V_m^{(k)}$ is the deformed version of $V^{(k)}$ for fitting image $\mathbf{I}_m$ (see Sect. 3.7). Set $z_m^{(k_*)} = 1$, and set $z_m^{(k)} = 0$ for $k \neq k_*$.

The implementation of this K-mean algorithm is similar to the EM algorithm. We only need to make the following modifications:

(1) Change the E-step: let $z_m^{(k_*)} = 1$ if k_* achieves the maximum of $\text{SUM2}_m^{(k)}$ among all $k = 1, \ldots, K$, and set the rest of $z_m^{(k)}$ to 0.

(2) Change the M-step: for each $k = 1, \ldots, K$, compute SUM2_m and estimate $\mathbf{B}$ and Θ for each cluster k using the shared sketch algorithm that maximizes the active correlation (see (32)).

Figures 41(b) and 42(b) display the learned templates $\mathbf{B}^{(k)}, k = 1, \ldots, K$ using K-mean algorithm. We initialize the algorithm with random $\{z_m^{(k)}\}$.

We also did a third experiment where we mix the positive training examples of head-shoulder images and negative training examples. The EM and K-mean algorithms can still separate out many of the positive training examples, although they also mistakenly include some negative examples into the positive cluster.

Negative experience in Experiment 7. When the object shapes of different categories are not very different, our method often fails to distinguish them if we start from random clustering.

The above difficulty is not caused by the model or the EM or K-mean iteration, but mainly by the fact that random clustering gives poor initialization. We address this issue in the next subsection.

6.2 Local Learning of Prototype Templates

To address the problem of initializing EM or K-mean, we develop a local learning scheme. The word "local" means being local in the high dimensional image space. It does not mean being local in the two-dimensional image lattice. Here the measure of locality or similarity depends on the models to be learned locally from similar examples. This naturally suggests an iterative procedure that iterates between learning the local model from similar examples and identifying similar examples based on the learned local model.

Local learning algorithm

Input: Training images $\{\mathbf{I}_m, m = 1, \ldots, M\}$.

Output: A prototype template $(\mathbf{B}^{(m)}, \Lambda^{(m)})$ around each image $\mathbf{I}_m$.

1. Initialize template $(\mathbf{B}^{(m)}, \Lambda^{(m)})$ by learning from the single image $\mathbf{I}_m$ using the shared sketch algorithm, with $b_1 = b_2 = 0$, i.e., no activity is allowed. Then restore b_1 and b_2 to their normal values (e.g., $b_1 = 2$ pixels, $b_2 = \pi/15$).
2. Use $(\mathbf{B}^{(m)}, \Lambda^{(m)})$ to score all the images, using the inference algorithm based on the sum-max maps. Find the K (e.g., $K = 5$) images with the highest SUM2 scores.
3. Re-learn $(\mathbf{B}^{(m)}, \Lambda^{(m)})$ from the K images identified in Step 2, using the shared sketch algorithm.
4. Go back to Step 2, and stop after t iterations (e.g., $t = 3$).

In the above algorithm, Step 2 can be very fast, because it only involves a linear combination of a small number of MAX1 scores for each image. Step 3 can also be fast because learning is done on a small number of nearest neighbors. A more localized implementation is to enforce that $\mathbf{I}_m$ must be among the K neighbors, and $\mathbf{I}_m$ may even receive a higher weight than other neighbors.

In local learning, we reduce the range of allowed activity, in order to get tighter clusters. Specifically, we set $b_1 = 2$ pixels, instead of 6 pixels as in Experiment 1.

Experiment 8. In Experiment 8.1, we learn local prototypes from a training set of 123 images of animal heads, where $K = 5$. After learning all the 123 templates, we trim them to satisfy the constraint that the K nearest neighbors of the remaining templates should not overlap (this may be too aggressive). This leaves 15 exemplar templates.

Figure 43 shows the 15 templates. They are ordered by the alignment scores computed from their respective K nearest neighbors.

Fig. 43 Experiment 8.1. The 15 locally learned prototypes. They are ordered by the alignment scores computed from their respective nearest neighbors. Image size is 100×100. Number of elements is 40. Number of iterations is 3 for learning each template. The allowed activity of location is up to 2 pixels. The allowed activity of orientation is up to $\pi/15$

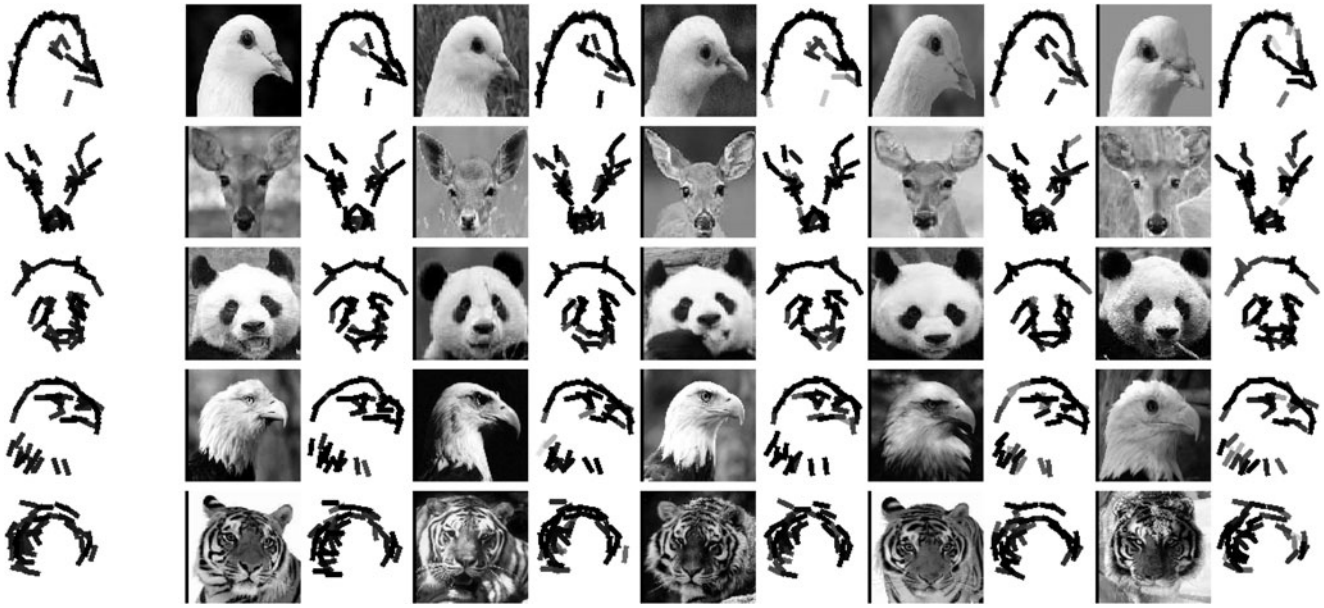


Fig. 44 Experiment 8.1. The top 5 templates and their neighbors

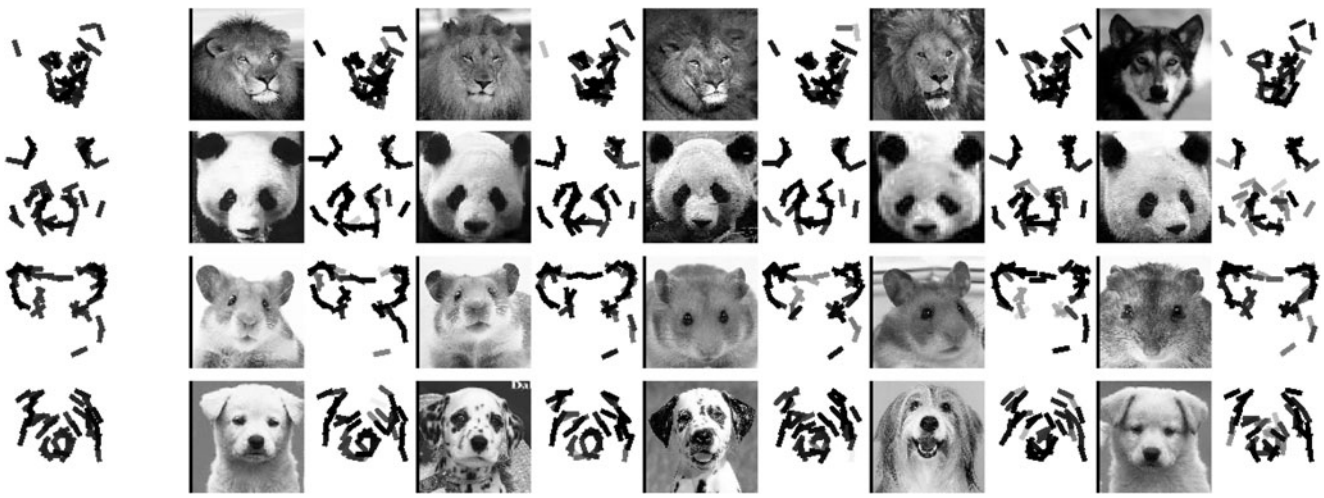


Fig. 45 Experiment 8.1. Some other templates and their neighbors

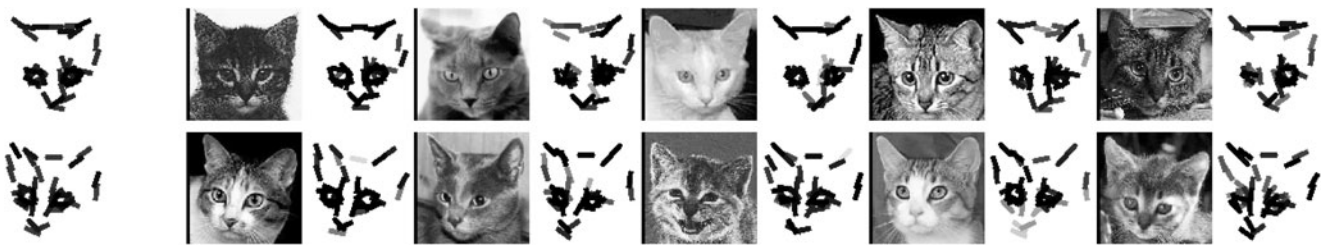


Fig. 46 Experiment 8.1. Templates of cats at two slightly different poses

Figure 44 shows the top 5 templates and their nearest neighbors.

Figure 45 shows another 4 templates and their nearest neighbors.

In Experiment 8.1, we pool $q(r)$ from negative images in Experiment 4. Pooling $q(r)$ from the two natural images in Sect. 3.1 leads to slightly different result. For other experiments in this paper, the two $q(r)$ lead to essentially the same results.

The locally learned templates may represent distinct object categories, but they may also represent different poses of the same object category. Figure 46 shows two templates of cats of slightly different poses.

In Experiment 8.2, we mix the images in Experiments 1.3 and 1.4, and the first 12 images in Experiment 3.2. The image length is 120. All the images share the same central horizontal line. The number of elements is 60. All the other parameters are the same as Experiment 8.1. The local learning algorithm returns 3 prototypes after trimming. Figure 47 displays the three prototypes. Figure 48 displays the three prototypes and their nearest neighbors.

In Experiment 8.3, we apply the same algorithm to 200 images of handwritten digits from the MNIST data set (LeCun et al. 1998), where we take the first 20 images for each digit. We obtain 21 locally learned templates, as shown in Fig. 49.

Fig. 49 Experiment 8.3. The 21 locally learned templates. Number of images is 200. The images are resized to 60×60. Original images are taken from MNIST data set. Number of elements is 15

In Experiment 8.4, we perform local learning on 912 images of horses. Some of the images are taken from the Weizmann data set (Borenstein and Ullman 2002) and the INRIA data set (Ferrari et al. 2007). The number of nearest neighbors $K = 20$. We sequentially select the templates $\{\mathbf{B}^{(l)}, l = 1, \ldots, L\}$ by maximizing a truncated log-likelihood score $\sum_{m=1}^{M} \text{truncate}(\max_l \text{SUM2}_m^{(l)}, T)$, where $\text{SUM2}_m^{(l)}$ is the template matching score of $\mathbf{I}_m$ matched to template $\mathbf{B}^{(l)}$, and $\max_l \text{SUM2}_m^{(l)}$ is the score of $\mathbf{I}_m$ matched to its closest template. $\text{truncate}(s, T) = s$ if $s > T$, and $\text{truncate}(s, T) = 0$ otherwise. That is, we only count the template matching scores where the images and the templates are good matches, in the hope that the selected templates and the images matched to them form the cores of the clusters. In this experiment, we let the number of elements be 40, and we set the threshold $T = 80$. Figure 50 shows the first 20 templates sequentially selected by maximizing the truncated log-likelihood score.

We are still unclear what is the principled way of selecting the locally learned templates. The selected templates can be used to initialize the EM algorithm for fitting the mixture model (we may also need to merge some of the similar clusters). It remains to be seen whether the mixture model fit this way can help the classification task or not.

Fig. 47 Experiment 8.2. The 3 representative templates locally learned. They are ordered by the total alignment scores. The image length is 120. All the images share the same central horizontal line. Number of elements is 60. All the other parameters are the same as Experiment 8.1

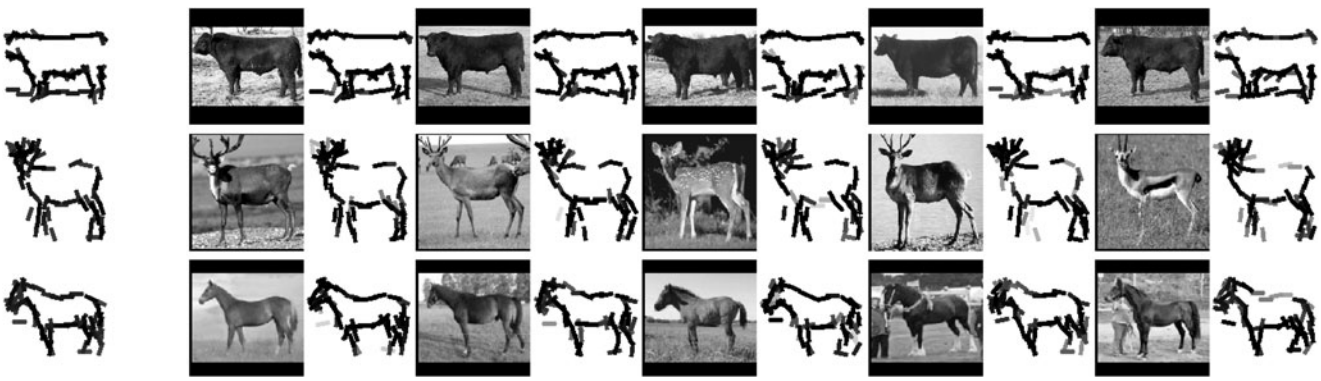


Fig. 48 Experiment 8.2. The 3 templates and their corresponding 5 nearest neighbors

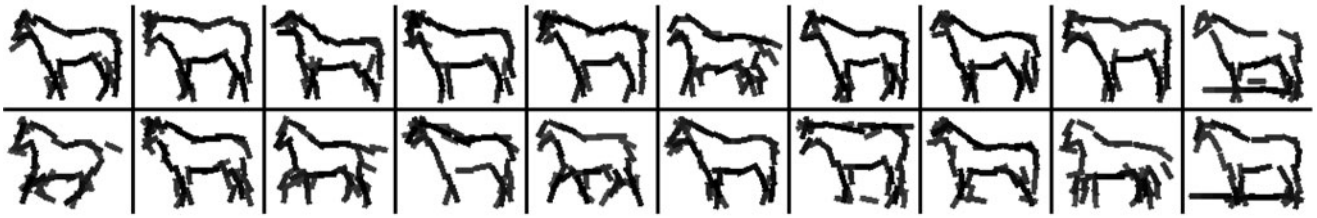


Fig. 50 Experiment 8.4. The 20 locally learned templates sequentially selected by maximizing a truncated log-likelihood score. The number of nearest neighbors is 20. The 912 images are 84 × 105. The number of elements is 40. Other parameters are the same as in Experiment 8.1

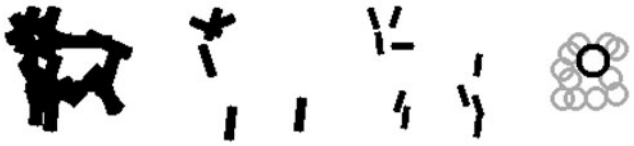


Fig. 51 Experiment 9.1. The selected Gabor elements (illustrated by *bars*) at 3 different scales and the selected DoG elements (illustrated by *circles*, and *larger circles* are darker than smaller ones). The lengths of the Gabor elements are 35, 25, and 17 pixels respectively. The sizes of the DoG elements are 77 and 55 respectively. The allowed activity of location is 4 pixels for both Gabor and DoG elements

7 Synthesis by Multi-scale Gabors and DoGs

Edges and regions can be considered two relative concepts in the frequency domain. While edges can be captured by high frequency Gabor wavelets, the regional contrasts can be encoded by low frequency wavelets, including the difference of Gaussian (DoG) wavelets. To account for both edges and regions, we need to combine Gabor and DoG wavelet elements at multiple frequency bands.

We select the wavelet elements of active basis from a dictionary of Gabor and DoG wavelets at different scales. We use the same shared sketch algorithm with sigmoid pursuit index, except that we normalize the filter responses by marginal variance. After selecting the elements and recording their responses, we use matching pursuit (Mallat and Zhang 1993) to reconstruct the images. We need to use matching pursuit for reconstruction because the selected elements are only approximately orthogonal to each other, so the projection coefficients and the reconstruction coefficients are slightly different. The matching pursuit algorithm computes the reconstruction coefficients from the projection coefficients.

Experiment 9. In Experiment 9.1, we use the same training images as in Experiment 1.3, except that we resize these images to make them smaller. Figure 51 displays the selected Gabor and DoG elements. The Gabor elements are illustrated by bars of different sizes. The DoG elements are illustrated by circles. The radius of a circle is about half of that of the blob represented by the corresponding DoG elements. Larger circles are darker than smaller ones.

Figure 52 displays the reconstructed images. The DoG elements are necessary to account for the large regional contrasts.

Figure 53 displays the reconstructed images with 150 wavelet elements. The reconstructed images have more details than those in Fig. 52. Alongside each reconstructed image, the corresponding residual image is also displayed. One can still recognize the objects from the residual images, suggesting that the model only explains away parts of the images.

Figures 54–57 display more examples. Ideally, the large Gabor and DoG elements gauge the breadths of the edges, while the small Gabor elements gauge the sharpness of the edges. The very large DoG elements may gauge the sizes of the regions, which are to be contoured by the Gabor elements.

Despite the fact that DoG elements can account for the regional intensity contrast, we still need to add local appearance variables to represent the local smoothness or textures in the interiors of the regions.

8 Composing Multiple Part-Templates

For articulate objects, we need to represent them as compositions of part-templates at different locations and resolutions.

Recursive active basis and recursive sum-max maps. An active basis is a composition of multiple Gabor wavelet elements, where each element is allowed to shift its location and orientation. We can further compose multiple active bases, where each active basis serves as a part-template that is allowed to change its overall location, orientation and scale. We call such a structure a "recursive active basis," which is a template that consists of multiple part-templates. The following experiment illustrates the basic idea.

Experiment 10. Figure 58(a) displays an observed image of size 330 × 496 of Experiment 10.1. Figure 58(b) displays the superposed sketch. The template is learned in Experiment 3.1 from the bicycle images. See Fig. 23. We split the bicycle template in Fig. 23 horizontally into two part-templates. The bounding box for the part-template of the

Fig. 52 Experiment 9.1. The first block displays all the 50 selected Gabor and DoG elements. The smaller Gabors are illustrated by *darker bars*. The remaining blocks display the original images and the corresponding reconstructed images. The image size is 102×100

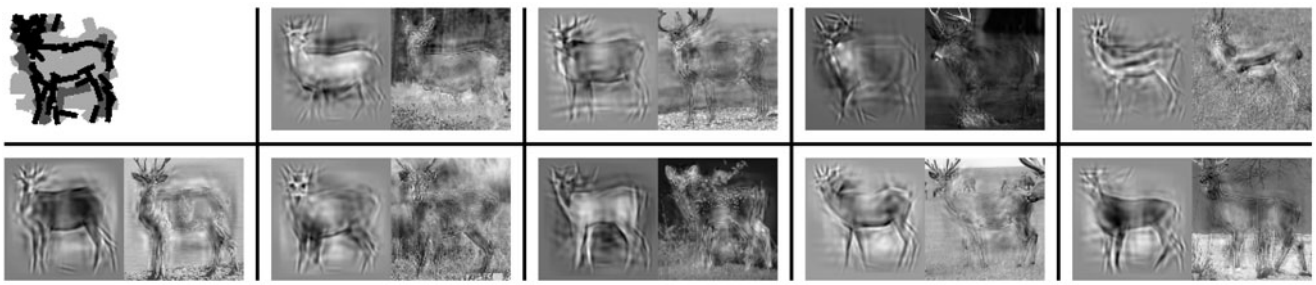


Fig. 53 Experiment 9.1. The first block displays all the 150 selected Gabor and DoG elements. The remaining blocks display the reconstructed images and the corresponding residual images

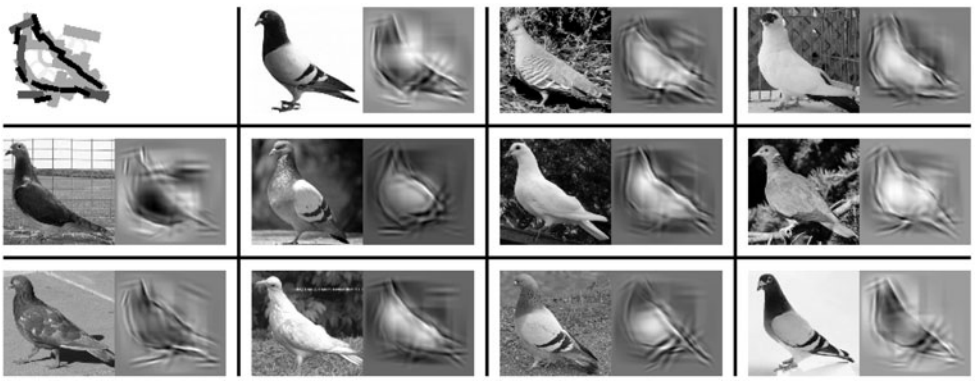


Fig. 54 Experiment 9.2. The first block displays all the 50 selected Gabor and DoG elements. The remaining blocks display the original 95×100 images and the corresponding reconstructed images

Fig. 55 Experiment 9.3. The first block displays all the 40 selected Gabor and DoG elements. The remaining blocks display the 100×70 original images and the corresponding reconstructed images

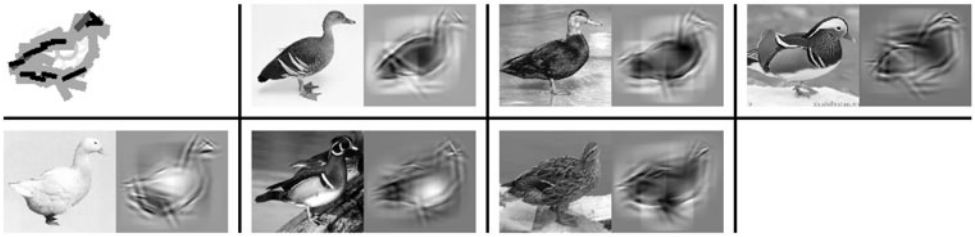


Fig. 56 Experiment 9.4. The first block displays all the 40 selected Gabor and DoG elements. The remaining blocks display the original 100×110 images and the corresponding reconstructed images

Fig. 57 Experiment 9.5. The first block displays all the 50 selected Gabor and DoG elements. The remaining blocks display the original 100×100 images and the corresponding reconstructed images

Fig. 58 Experiment 10.1. (**a**) Input image of 330×496. (**b**) Superposed sketch. The bounding box of the front wheel is 112×126. The bounding box of the back wheel is 86×76. The total number of elements is 60

front wheel is 112×126, and the bounding box for the part-template of the back wheel is 86×76 (we give some extra margin to the bounding box of each part-template at the splitting point). We allow the two part-templates to locally shift horizontally, so these two part-templates make up a recursive active basis. We then fit the recursive template to the tandem bike in Fig. 58(a) and obtain the sketch in Fig. 58(b).

Given the two part-templates, the inference can be accomplished by alternating the sum maps and max maps as illustrated by Fig. 59. Here we have two SUM2 maps, one for each part-template. On top of each SUM2 map, there is also a MAX2 map. Then on top of the two MAX2 maps, a SUM3 maps is computed. After that a MAX3 score is obtained. These scores are computed by a bottom-up process, and they answer the following questions:

SUM2 maps: Is there a part-template at this location?

MAX2 maps: Is there a part-template at a *nearby* location?

SUM3 map: Is there a certain composition of part-templates that form the whole template at this location?

MAX3 score: Is there a composite template within the whole image?

If there is such a composite template, then a top-down process first retrieves the location of the whole template, then retrieves the locations of the part-templates, and finally retrieves the elements of the part-templates.

Inference by recursive sum-max maps

Input: Part-templates $(\mathbf{B}^{(j)}, \Lambda^{(j)})$, $j = 1, 2$. Their central locations (x_j, y_j) for $j = 1, 2$ in the composite whole template. Testing image $\mathbf{I}$.

Output: Detected location $(\hat{x}, \hat{y})$ of the whole template in the testing image $\mathbf{I}$, as well as the detected locations $(\hat{x}_j, \hat{y}_j)$ of the part-templates for $j = 1, 2$.

Up-1 For $j = 1, 2$, compute SUM2(x, y, j) using the inference algorithm of Sect. 2.4.

Up-2 For all (x, y) and $j = 1, 2$, compute

$$\begin{aligned} &\text{MAX2}(x, y, j) \\ &= \max_{\substack{-b_x \le \Delta x \le b_x \\ -b_y \le \Delta y \le b_y}} \text{SUM2}(x + \Delta x, y + \Delta y, j), \\ &j = 1, 2. \end{aligned} \tag{38}$$

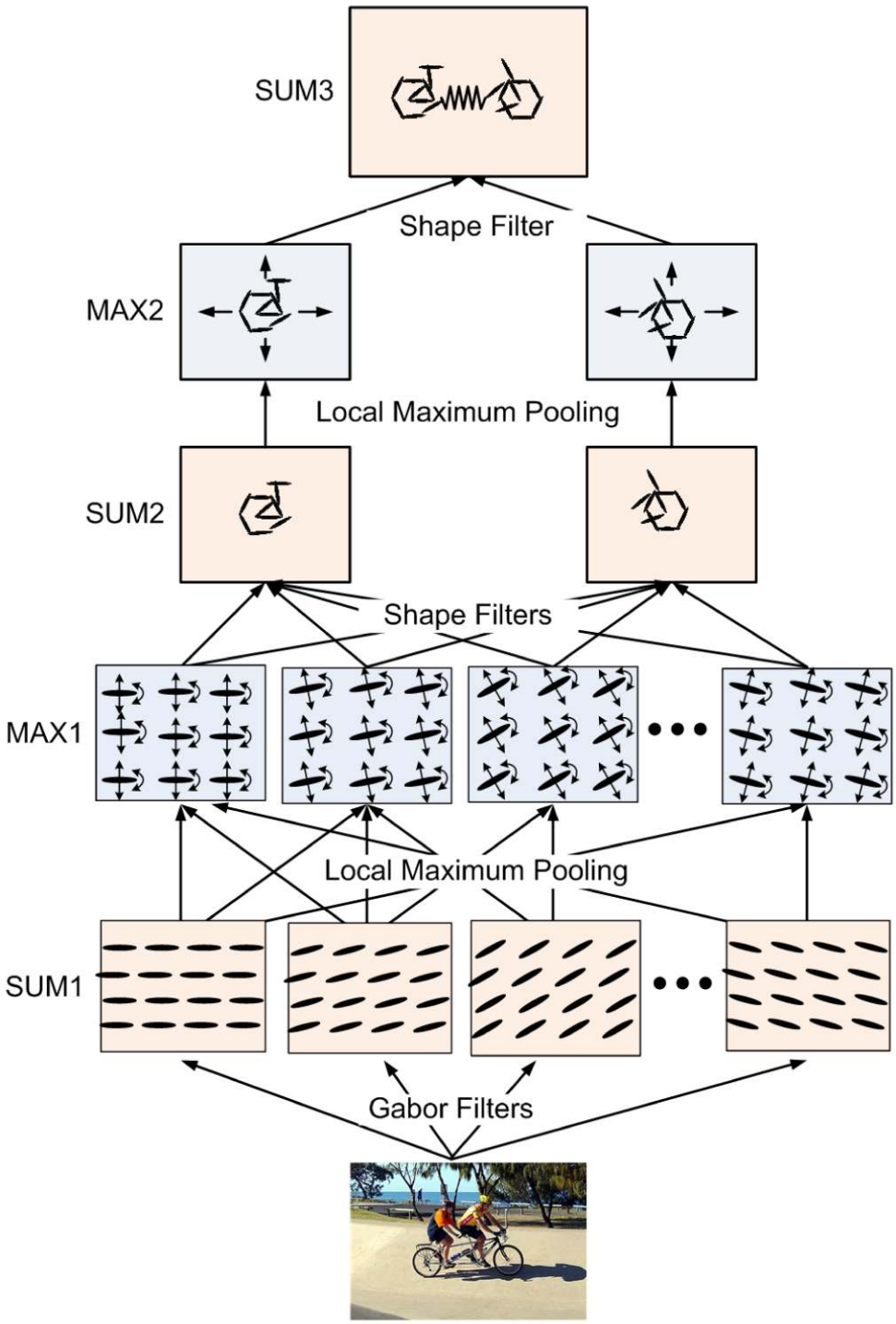


Fig. 59 Recursive sum-max maps. A SUM2 map is computed for each part-template. For each SUM2 map, a MAX2 map is computed by applying a local maximization operator to the SUM2 map. Then a SUM3 map is computed by summing over the two MAX2 maps. The SUM3 map scores the template matching, where the template consists of two part-templates that are allowed to locally shift their locations

Up-3 For all (x, y), compute

$$\mathrm{SUM3}(x, y) = \sum_{j=1}^{2} \mathrm{MAX2}(x + x_j, y + y_j, j).$$

Up-4 Compute $\mathrm{MAX3} = \max_{x,y} \mathrm{SUM3}(x, y)$.

Down-4 Retrieve $(\hat{x}, \hat{y})$ that achieves the maximum in Up-4.

Down-3 Retrieve $(\hat{x} + x_j, \hat{y} + y_j)$ in Up-3 for $j = 1, 2$.

Down-2 Retrieve $(\hat{x}_j, \hat{y}_j)$ so that

$$\mathrm{MAX2}(\hat{x} + x_j, \hat{y} + y_j, j) = \mathrm{SUM2}(\hat{x}_j, \hat{y}_j, j),$$

in the local maximization operation (38) in Up-2.

Down-1 Retrieve the perturbed elements of j-th part-template for $j = 1, 2$, as described in the inference algorithm of Sect. 2.4.

The retrieval in Down-2 step can be implemented by storing the TRACK2 maps in Up-2 step.

In Experiment 10.1, in Step Up-2, we take $b_x = 20$ pixels and $b_y = 4$ pixels. Let (x_1, y_1) and (x_2, y_2) be the central positions of the bounding boxes for the two part-templates in the original template learned from regular bicycles. Assume $x_1 < x_2$, we let $x_1 \leftarrow x_1 - b_x$, $x_2 \leftarrow x_2 + b_x$, and let the two part-templates shift around the new centers (x_1, y_1) and (x_2, y_2).

The MAX3 score in Up-4 measures the template matching, or the alignments of the two part-templates to the images. This MAX3 score can be used to decide where we should split the original bicycle template. Specifically, we can try different splitting points, and for each splitting point, we compute the MAX3 score. Figure 60 displays the MAX3 scores for 10 different splitting points. The result shown in Fig. 58(b) is obtained at the splitting point that achieves the maximum MAX3 score.

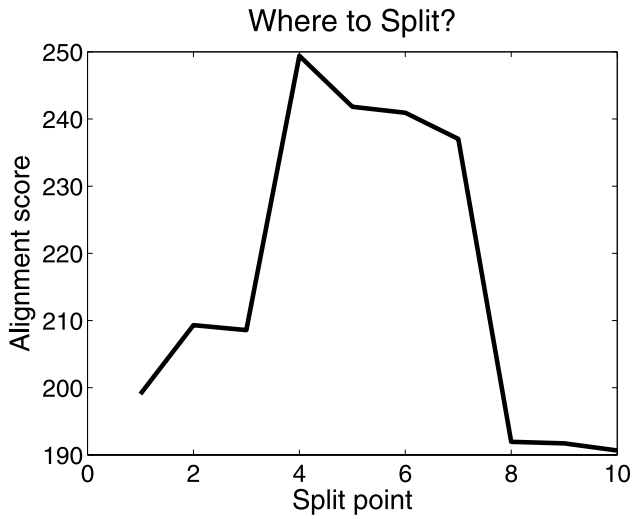


Fig. 60 Experiment 10.1. MAX3 scores for different splitting points

The recursive active basis can be considered a constellation model (Weber et al. 2000) whose constituent components are active bases. The MAX2 and SUM3 maps may have been commonly used in part-based models. Thanks to the work of Riesenhuber and Poggio (1999), we are able to extend the SUM and MAX operations down to the image intensities.

Account for large deformations. The recursive active basis and recursive sum-max maps can account for the existence of parts, as illustrated in Experiment 10.1. They can also be used to deal with large deformations.

Figure 61(a) displays the image of horse that we used in Experiment 3.2, where we change the aspect ratio of the horse template to fit this image. From a 2D point of view, this amounts to a large deformation that cannot be handled by a single-layer active basis model. We can use the same method in Experiment 10.1 to split the original horse template into two part-templates, and allow these two part-templates to move relative to each other. Figure 61(b) displays the result of fitting the recursive active basis at the optimal splitting point. As a comparison, Fig. 61(c) displays the result using the original template. The original template does not fit the head and the rear parts of the horse very well.

Experiment 10 on composing part-templates and Experiment 5c on learning part-templates are only illustrative and very preliminary. There is still a long way to go to develop a simple and robust scheme to learn multi-scale and multi-layer recursive active basis.

9 Discussion

The proposed approach is very simple for the vision tasks studied in this article. The model is not much more complex than a wavelet expansion, except that local perturbations are added to the wavelet elements. The learning algorithm is not much more complex than edge detection, except that it is performed simultaneously on multiple images. The

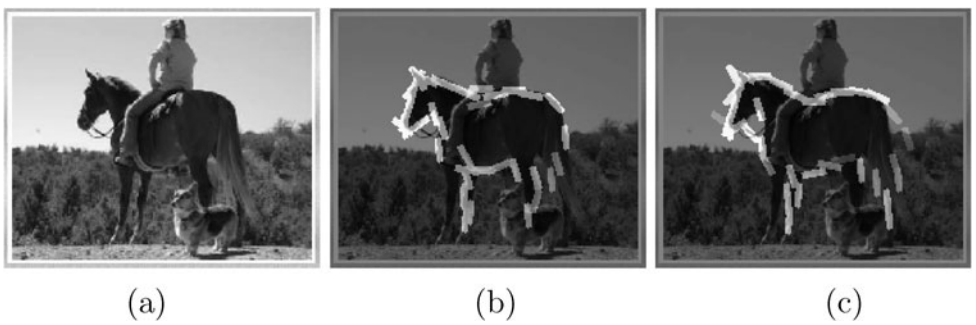


Fig. 61 Experiment 10.2. (**a**) The observed image of 166 × 202. (**b**) Superposed with sketch where the horse template is split horizontally into two part-templates. The total number of elements is 40. The left part-template is 116 × 76. The right part-template is 104 × 92. These two part-templates are allowed to move horizontally up to 10 pixels in each direction. (**c**) Superposed with sketch using the original horse template. In other words, the two part-templates are not allowed to move relative to each other

inference algorithm only involves two consecutive filtering operations on top of Gabor filtering. One is a local max filtering and the other is a local sum filtering.

We play with the active basis model in a variety of experiments. These experiments are merely illustrative and explorative. Far more empirical experiences are needed to better understand the limitations and inadequacies of the model and to improve it.

In retrospect, we find the following three principles relevant and helpful.

9.1 Sparsity

Olshausen and Field (1996) propose this principle for understanding V1 simple cells, where a typical natural image can be represented by a linear superposition of a small number of Gabor-like wavelet elements at different scales, locations, and orientations, plus a small residual. The reason for such a sparse representation is that edges are prominent and frequently occurring structures in natural images.

The active basis model can be considered a further step in sparse coding. In Olshausen-Field representation, each image is encoded by a sparse set of localized and oriented wavelet elements of various scales. This effects a key transition in representation, from raw image intensities to a geometric representation in terms of locations, orientations, and scales of the wavelet elements. We can further encode this geometric representation by a small number of templates, each being a composition of locations, orientations, and scales. The reason for such a sparser representation is that those templates are prominent and frequently occurring structures in natural images. In Olshausen-Field representation, we need to allow for small residuals in image intensities. Similarly, in this geometrical representation, we need to allow for small residuals in locations, orientations, and scales. Such small residuals become the perturbations or activities of the elements of the active basis model, so that the templates are deformable.

9.2 Compositionality

Geman et al. (2002) propose this principle for vision. If we want a compositional representation of image intensities and if we insist on linear representation for simplicity, then it is natural to adopt wavelet representation because the wavelet elements are localized in both spatial and frequency domains. The active basis model follows such a compositional scheme.

Zhu and Mumford (2006) investigate the and-or graph as a recursive compositional scheme for vision, where "and" accounts for compositions of constituent elements, while "or" accounts for variations in the constituent elements. The active basis model is a simplest form of an and-or graph, where "and" means composition of wavelet elements, and "or" means variations in the locations and orientations of the elements. The and-or graph is a grammar that can be applied recursively. The recursive active basis follows such a grammar.

The recursive architecture of sum-max maps is a variation on the theme of Riesenhuber and Poggio's cortex-like structure (Riesenhuber and Poggio 1999). The sum-max maps form a natural hierarchical structure for parsing an image according to the and-or grammar. The sum maps score the and-compositions, and the max maps account for the or-variations. After bottom-up scoring for detection and classification, the top-down retrieving produces the parsing of the image. See also the recent work of Zhu et al. (2008) on a recursive compositional scheme.

9.3 Invariance

Riesenhuber and Poggio (1999) propose this principle for V1 complex cells. While the V1 simple cells capture the essence of the image intensities via Olshausen-Field sparse coding, the local maximization operation of the V1 complex cells filters out shape deformations, and makes the subsequent processing invariant to shape deformations. Of course, invariance here is only approximate.

The Riesenhuber-Poggio scheme compares intensities of the MAX1 maps directly for template matching. We modify their template matching scheme by a weighted sum of the MAX1 intensities at highly selected locations and orientations. If the locations and orientations of the selected wavelet elements are at the centers of the local perturbations that cause shape deformation, then hopefully, the intensities of the MAX1 maps of these highly selected locations and orientations are more invariant (and more indicative of the object shapes) than the intensities of other locations and orientations.

Reproducibility

All the experimental results reported in this paper can be reproduced by the Matlab and mex-C code that we have posted on the webpage http://www.stat.ucla.edu/~ywu/ActiveBasis.html.

Acknowledgements We thank the reviewer for the helpful comments and suggestions. We thank Chuck Fleming for earlier collaboration on some of the experiments. We thank Alan Yuille, Stefano Soatto, Tai Sing Lee, Zhuowen Tu, and Leo Zhu for discussions. The work is supported by NSF-DMS 0707055, NSF-IIS 0713652, ONR N00014-05-01-0543, Air Force grant FA 9550-08-1-0489, and Keck foundation. We acknowledge the use of data sets provided by the Lotus Hill Institute, which is supported by a Chinese National 863 grant 2006AA01Z121 and an NSFC grant 60728203.

References

Amit, Y., & Trouve, A. (2007). Pop: Patchwork of parts models for object recognition. *International Journal of Computer Vision*, *75*, 267–282.

Borenstein, E., & Ullman, S. (2002). Class-specific, top-down segmentation. In *Proceedings of European conference on computer vision*.

Cootes, T. F., Edwards, G. J., & Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*, 681–685.

Daugman, J. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of Optical Society of America*, *2*, 1160–1169.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, *39*, 1–38.

Ferrari, V., Jurie, F., & Schmid, C. (2007). Accurate object detection with deformable shape models learnt from images. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

Ferryman, J. M. (2006). In *Proceedings of ninth IEEE international workshop on performance evaluation of tracking and surveillance (PETS 2006)*.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Friedman, J. H. (1987). Exploratory projection pursuit. *Journal of the American Statistical Association*, *82*, 249–266.

Geman, S., Potter, D. F., & Chi, Z. (2002). Composition systems. *Quarterly of Applied Mathematics*, *60*, 707–736.

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: active contour models. *International Journal of Computer Vision*, *1*, 321–331.

Lades, M., Vorbrggen, J. C., Buhmann, J., Lange, J., von der Malsburg, C., Wrtz, R. P., & Konen, W. (1993). Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, *42*, 300–311.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

Mallat, S., & Zhang, Z. (1993). Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, *41*, 3397–3415.

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607–609.

Pietra, S. D., Pietra, V. D., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 380–393.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*, 1019–1025.

Tu, Z. (2007). Learning generative models via discriminative approaches. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

Ullman, S. (1996). *High-level vision: object recognition and visual cognition*. Cambridge: MIT Press.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*, 137–154.

Weber, M., Welling, M., & Perona, P. (2000). Towards automatic discovery of object categories. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

Wu, Y. N., Shi, Z., Fleming, C., & Zhu, S. C. (2007). Deformable template as active basis. In *Proceedings of international conference on computer vision*.

Wu, Y. N., Guo, C., & Zhu, S. C. (2008). From information scaling of natural images to regimes of statistical models. *Quarterly of Applied Mathematics*, *66*, 81–122.

Yuille, A. L., Hallinan, P. W., & Cohen, D. S. (1992). Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, *8*, 99–111.

Zhu, L., Lin, C., Huang, H., Chen, Y., & Yuille, A. (2008). Unsupervised structure learning: hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *Proceedings of European conference on computer vision*.

Zhu, S. C., & Mumford, D. B. (1997). Prior learning and Gibbs reaction-diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 1236–1250.

Zhu, S. C., & Mumford, D. B. (2006). A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, *2*, 259–362.

Zhu, S. C., Wu, Y. N., & Mumford, D. B. (1997). Minimax entropy principle and its applications to texture modeling. *Neural Computation*, *9*, 1627–1660.

Zhu, S. C., Guo, C. E., Wang, Y. Z., & Xu, Z. J. (2005). What are textons? *International Journal of Computer Vision*, *62*, 121–143.