

Active Basis for Modeling, Learning and Recognizing Deformable Templates

Ying Nian Wu¹, Zhangzhang Si¹, Haifeng Gong^{1,2}, and Song-Chun Zhu^{1,2}

¹ Department of Statistics, University of California, Los Angeles

² Lotus Hill Research Institute, Ezhou, China

{ywu, zzsi, hfgong, sczhu}@stat.ucla.edu

Revised September 25, 2008

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 at a dense collection of locations and orientations. 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: Generative model; Object recognition; Shared sketch algorithm; Sum maps and max maps; Wavelet sparse coding.

This paper is submitted to the International Journal of Computer Vision (IJCV). A shorter version has been published in ICCV 2007 [21]. For reproducibility, the matlab/C codes and data used in this paper are available at www.stat.ucla.edu/~ywu/ActiveBasis.html

1 Introduction

Deformable template [25] is an important element in object recognition [5]. 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

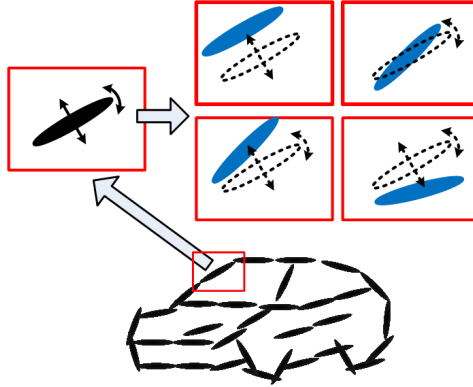


Figure 1: Active basis. Each basis element is illustrated by a thin ellipsoid at a certain location and orientation. The upper half shows the perturbation of one basis element. By shifting its location or orientation or both within a limited range, the basis element (illustrated by a black ellipsoid)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

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 Figure (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 Figure (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 Figure (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 Figure (2) are allowed to locally change their locations and orientations to code each observed image, as illustrated by the

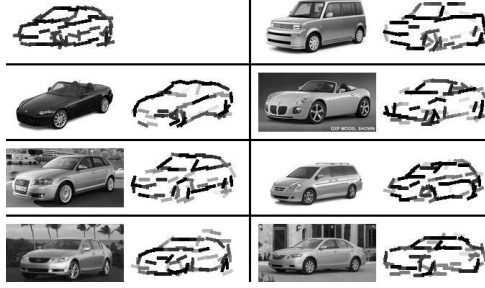


Figure 2: Active basis formed by 60 Gabor wavelet elements. The first block displays the 60 elements, where each element is represented by a bar. For each of the other 7 blocks, the left plot is the observed image, and the right plot displays the 60 Gabor wavelet elements resulting from locally shifting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

remaining 7 blocks of Figure (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.

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 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 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 is equivalent 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

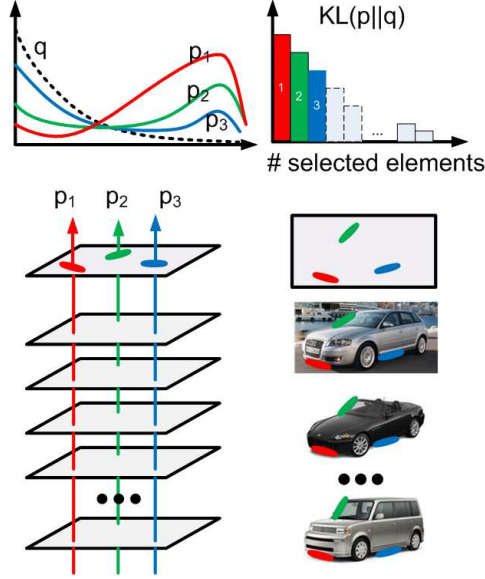


Figure 3: Shared sketch algorithm. A selected element (colored ellipsoid) is shared by all the training images. For each image, a perturbed version of the element seeks to sketch a local edge segment near the element by a local maximization operation. The elements of the active basis are selected sequentially 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.

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 Figure (2) is learned by the shared sketch algorithm. It is worth noting that for the last two examples in Figure (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. 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 Figure (4), the thin ellipsoids in the SUM1 maps illustrate the local filtering or summation operation. Then a

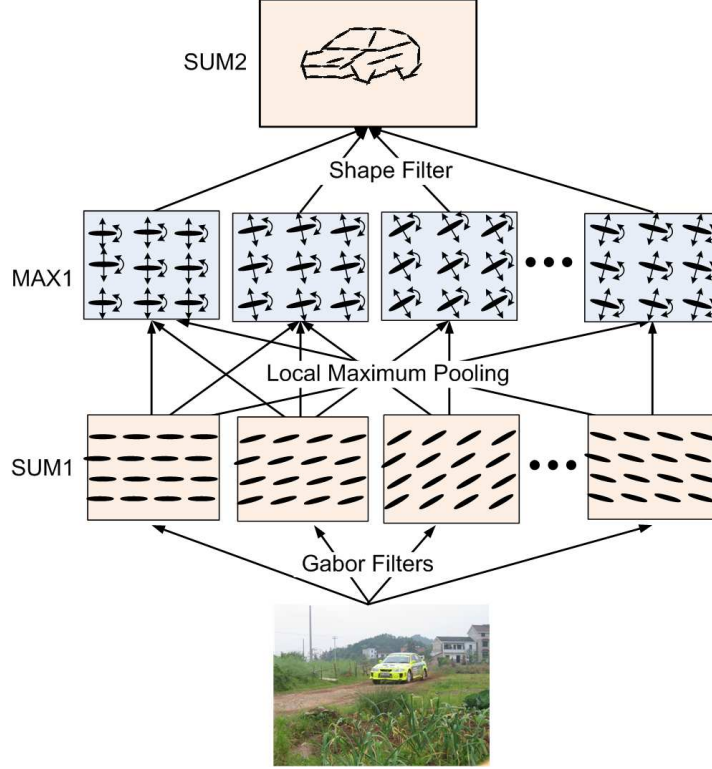


Figure 4: Sum-max maps. The SUM1 maps are obtained by convolving the input image with Gabor filters at all the locations and orientations. The ellipsoids in the SUM1 maps illustrate the local filtering or summation operation. The MAX1 maps are obtained by applying a local maximization operator to the SUM1 maps. The arrows in the MAX1 maps illustrate the perturbations over which the local maximization is taken. The SUM2 maps are 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.

layer of max maps, or MAX1 maps, is computed by applying a local maximization operator to the SUM1 maps. In Figure (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 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 Figure (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.

1.4 Review of literature

Our work cultivates key insights from three major theories on biological and computer vision, namely, Olshausen and Field’s theory of linear sparse coding using Gabor wavelets [14], Riesenhuber and Poggio’s theory on local maximum pooling of Gabor filter responses [15], and Viola and Jones’s theory of adaboost learning [6] using Harr wavelets as weak classifiers [17].

Our active basis model is based on Olshausen and Field’s linear representation. Inspired by Riesenhuber and Poggio’s operator, we add local perturbations to the basis elements. Motivated by Viola and Jones’s work, we apply the model to images of object shapes, so that the training images share the same set of selected elements, which form a common deformable template [25]. This connects Olshausen and Field’s work to shape models such as active contours [11] and active appearance model [1].

Our algorithmic architecture of sum-max maps is a variation on the theme of Riesenhuber and Poggio’s cortex-like hierarchical structure of simple and complex cells [15]. See also the recent work of Mutch and Lowe [13] for further improvement. Our architecture is essentially the same as Riesenhuber and Poggio’s structure up to the MAX1 maps. However, our operator for computing the SUM2 map for template matching is different from the template matching scheme of Riesenhuber and Poggio. Our operator is a local summation over highly selected locations and orientations. Moreover, the computation is not entirely a bottom-up process. After the bottom-up scoring process, a top-down retrieving process traces the locations and orientations of the perturbed elements, which form the deformed template that is matched to the input image. Furthermore, since the local summation is taken over a small number of selected locations and orientations, it may be more efficient than comparing all the intensities of the MAX1 maps.

Our shared sketch algorithm is similar to the learning scheme of Viola and Jones’s version of adaboost. However, the algorithm is guided by a generative model, where the selection of the basis elements is based on explaining away the image data instead of fitting the classification boundary. Our learning algorithm can be considered a parallel version of the matching pursuit algorithm [12] where we select the basis elements to simultaneously code all the training images. It can also be considered a variation of the projection pursuit algorithm for density estimation [7], where we add to it the local maximization and local inhibition operations of the matching pursuit algorithm. The exponential family model we adopt is related to the feature induction schemes of Della Pietra et al. [4] and Zhu et al. [30].

Our work is a continuation of our long term search for generative models and model-based

algorithms [30, 23, 27, 22, 9, 10], as well as our attempt to understand these models within a common information-theoretical framework [20]. The active basis model can be considered a revision of our previous model on textons [27]. It can also be viewed as an inhomogeneous version of the Markov random field model that we previously developed for textures [30], as we show in [20]. More important, the active basis model is a simplest instance of the and-or graph [29] in the compositional framework [8] that we have been studying. The and-or grammar naturally suggests that we can further compose multiple active bases to represent more articulate shapes.

2 Representation, Learning, and Inference

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

2.1 Gabor wavelets and sparse coding

A dictionary of Gabor wavelets. A Gabor function [3] is of the form

$$G(x, y) \propto \exp\{-(x/\sigma_x)^2 - (y/\sigma_y)^2/2\}e^{ix}.$$

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 good fit to the receptive fields of the simple cells in V1 [3].

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.

Operation. 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 2 pixels), and $\alpha \in \{a\pi/A, a = 0, \dots, A-1\}$ (e.g., $A = 15$).

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

$$\langle \mathbf{I}, B_{x,y,s,\alpha,\eta} \rangle = \sum_{x', y'} \mathbf{I}(x', y') B_{x,y,s,\alpha,\eta}(x', y').$$

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

$$|\langle \mathbf{I}, B_{x,y,s,\alpha} \rangle|^2 = \langle \mathbf{I}, B_{x,y,s,\alpha,0} \rangle^2 + \langle \mathbf{I}, B_{x,y,s,\alpha,1} \rangle^2.$$

In the computation of $\langle \mathbf{I}, B_{x,y,s,\alpha} \rangle$, $B_{x,y,s,\alpha}$ is a linear filtering operator, which can serve as edge detector or local spectral analyzer.

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

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

where $|D|$ is the number of pixels in \mathbf{I} , and A is the total number of orientations. For each input image \mathbf{I} , we normalize it to $\mathbf{I} \leftarrow \mathbf{I}/\sigma(s)$.

Representation. A deeper perspective is offered by the sparse coding theory of Olshausen and Field [14], where $B_{x,y,s,\alpha}$ serves as a representational element. Specifically, for an image \mathbf{I} , we can represent it by

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

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 $\{B_i, i = 1, \dots, n\}$ are selected from the dictionary. If the $\{B_i, i = 1, \dots, n\}$ 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} , we 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 small residual U . Of course, for different images, we usually select 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 [14] were able to learn from natural image patches a dictionary of wavelet elements that closely resemble the properties of the receptive fields of the simple cells in V1.

2.2 Representation: active basis model

The sparse coding model (2) is constructed for the whole ensemble of natural images, where for different \mathbf{I} , we may represent them with completely different wavelet elements $(B_i, i = 1, \dots, n)$ 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 $(B_i, i = 1, \dots, n)$, and these elements form a common template. However, when we use $(B_i, i = 1, \dots, n)$ 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 $\{\mathbf{I}_m, m = 1, \dots, M\}$ 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. It is possible to learn multiple templates at multiple scales and then combine them.

The active basis model is a composition of strokes that are perturbable:

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

$$\text{Perturbations : } B_{m,i} \approx B_i, \quad i = 1, \dots, n, \quad (4)$$

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

$$B_i = B_{x_i, y_i, s, \alpha_i}, \quad (5)$$

$$B_{m,i} = B_{x_{m,i}, y_{m,i}, s, \alpha_{m,i}}, \quad (6)$$

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

$$x_{m,i} = x_i + d_{m,i} \sin \alpha_i, \quad (7)$$

$$y_{m,i} = y_i + d_{m,i} \cos \alpha_i, \quad (8)$$

$$\alpha_{m,i} = \alpha_i + \delta_{m,i}, \quad (9)$$

$$d_{m,i} \in [-b_1, b_1], \quad \delta_{m,i} \in [-b_2, b_2]. \quad (10)$$

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

Because we fix the scale s in the linear 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 or resolutions.

2.3 Learning: shared sketch algorithm

Given the set of training images $\{\mathbf{I}_m, m = 1, \dots, M\}$, 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 those B_i so that its perturbed versions $\{B_{m,i}, m = 1, \dots, M\}$ sketch as many edge segments as possible in the training images $\{\mathbf{I}_m\}$.

Description of the shared sketch algorithm

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

Output : Template $\mathbf{B} = (B_i, i = 1, \dots, n)$, and deformed template $\mathbf{B}_m = (B_{m,i}, i = 1, \dots, n)$ that is matched to \mathbf{I}_m for $m = 1, \dots, M$.

1. Convolution: For each $m = 1, \dots, M$, and for each $B \in \text{Dictionary}$, compute $[\mathbf{I}_m, B] = h(|\langle \mathbf{I}_m, B \rangle|^2)$. Set $i \leftarrow 1$.
2. Local maximization: For each putative candidate $B_i \in \text{Dictionary}$, do the following: For each $m = 1, \dots, M$, 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 $B_i \in \text{Dictionary}$. Record this B_i and retrieve the corresponding optimal $B_{m,i} \approx B_i$ for $m = 1, \dots, M$.
4. Non-maximum suppression: For each $m = 1, \dots, M$, if $[\mathbf{I}_m, B_{m,i}] > 0$, then for every $B \in \text{Dictionary}$ 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 $|\langle \mathbf{I}_m, B \rangle|^2$. 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.

See Figure (3) for an illustration of the above algorithm. The algorithm can be considered a parallel version of edge detection. 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 $\sum_{m=1}^M [\mathbf{I}_m, B_{m,i}]$, 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 $[\mathbf{I}_m, B] \leftarrow 0$. So for each image \mathbf{I}_m , the selected $(B_{m,i}, i = 1, \dots, n)$ are approximately orthogonal to each other.

The algorithm can learn from a single training image. 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.

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 $r \geq \xi$, 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 $\{\mathbf{I}_m, m = 1, \dots, M\}$. 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., $\xi = 6$),

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

which increases from 0 to ξ .

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

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

(3) Thresholding transformation. A crude but simple approximation to $\text{whiten}(r)$ is

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

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

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

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

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 $\Lambda = (\lambda_i, i = 1, \dots, n)$ is estimated by maximum likelihood method. See the next section for details.

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

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

where $h(r) = \text{whiten}(r)^{1/2}$ or $h(r) = \text{threshold}(r)^{1/2}$, and $\Theta = (\theta_i, i = 1, \dots, n)$ 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 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 $\Theta = (\theta_i, i = 1, \dots, n)$ 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 , because \mathbf{B} is deformed to $\mathbf{B}_m = (B_{m,i}, i = 1, \dots, n)$ 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 .

2.4 Inference: sum-max maps

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

Let \mathbf{I} be a testing image, which is larger than the bounding box of the template \mathbf{B} . We assume that the bounding box of \mathbf{B} is centered at origin $(x = 0, y = 0)$. We can scan the template over image \mathbf{I} , and at each position (x, y) , we fit the active basis model to the image within the bounding box centered at (x, y) , and calculate the template matching score according to Equation (14) or (15).

The inference algorithm consists of two processes. The first process is a bottom-up scoring 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?

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 \rightarrow MAX1 \rightarrow SUM2 \rightarrow 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 order of MAX2 \rightarrow SUM2 \rightarrow MAX1 \rightarrow SUM1. The process detects and deforms the template to interpret the observed image.

Pseudo-code for inference algorithm

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

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

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

$$\text{SUM1}(x, y, s, \alpha) = |\langle \mathbf{I}, B_{x, y, s, \alpha} \rangle|^2. \quad (16)$$

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

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

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

$$\text{SUM2}(x, y) = \sum_{i=1}^n [\lambda_i h(\text{MAX1}(x + x_i, y + y_i, s, \alpha_i)) - \log Z(\lambda_i)]. \quad (18)$$

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

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 $i = 1, \dots, n$ in Up-3.

Down-2 Retrieve $\hat{x}_i, \hat{y}_i, \hat{\alpha}_i$ for $i = 1, \dots, n$, such that

$$\text{MAX1}(\hat{x} + x_i, \hat{y} + y_i, s, \alpha_i) = \text{SUM1}(\hat{x}_i, \hat{y}_i, s, \hat{\alpha}_i) \quad (19)$$

in the local maximization operation (17) of Up-2.

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

Then the Gabor wavelet elements $(B_{\hat{x}_i, \hat{y}_i, s, \hat{\alpha}_i}, i = 1, \dots, n)$ form the deformed template fitted to image **I**.

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 Figure (4) for an illustration. The shape filter in Up-3 has fixed $(x_i, y_i, \alpha_i, i = 1, \dots, n)$. 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.

2.5 Shared sketch algorithm based on sum-max maps

The shared sketch algorithm in Subsection (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 $\{\mathbf{I}_m, m = 1, \dots, M\}$.

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

1. Convolution: For each $m = 1, \dots, M$, for all $(x, y) \in D$, and for all α , compute the SUM1 maps:

$$\text{SUM1}_m(x, y, s, \alpha) = |\langle \mathbf{I}_m, B_{x, y, s, \alpha} \rangle|^2. \quad (20)$$

2. Local maximization: For each $m = 1, \dots, M$, for all $(x, y) \in D$, and for all α , compute the MAX1 maps:

$$\text{MAX1}_m(x, y, s, \alpha) = \max_{\substack{d \in [-b_1, b_1] \\ \delta \in [-b_2, b_2]}} \text{SUM1}_m(x + d \sin \alpha, y + d \cos \alpha, s, \alpha + \delta). \quad (21)$$

For each $m = 1, \dots, M$, 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) .

For each $m = 1, \dots, M$, retrieve $(x_{m,i}, y_{m,i}, \alpha_{m,i})$ so that

$$r_{m,i} = \text{MAX1}_m(x_i, y_i, s, \alpha_i) = \text{SUM1}_m(x_{m,i}, y_{m,i}, s, \alpha_{m,i}) \quad (22)$$

in the local maximization computation in (21). That is, we perturb $B_i = B_{x_i, y_i, s, \alpha_i}$ to $B_{m,i} = B_{x_{m,i}, y_{m,i}, s, \alpha_{m,i}}$ to fit \mathbf{I}_m .

Compute λ_i from $\sum_{m=1}^M h(\text{MAX1}_m(x, y, s, \alpha))$. For each $m = 1, \dots, M$, compute $\text{SUM2}_m \leftarrow \text{SUM2}_m + \lambda_i h(r_{m,i}) - \log Z(\lambda_i)$.

4. Non-maximum suppression: If $r_{m,i} > 0$, 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 (21).

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 some 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 $C(\Delta x, \Delta y, \Delta \alpha) = \text{corr}(B_{x+\Delta x, y+\Delta y, s, \alpha+\Delta \alpha}, B_{x, y, s, \alpha})$ before we run the algorithm.

(3) After selecting (x_i, y_i, α_i) , we need to go back to retrieve $(x_{m,i}, y_{m,i}, \alpha_{m,i})$. We can record this information for every (x, y, α) when performing the local maximization in (21), and then retrieve

the information for (x_i, y_i, α_i) . If we choose not to store this information beforehand, we can re-do the local maximization in (21) for (x_i, y_i, α_i) to retrieve $(x_{m,i}, y_{m,i}, \alpha_{m,i})$.

The SUM2_m score evaluates the matching of \mathbf{I}_m to the learned template \mathbf{B} according to Equation (14). The total score $\sum_{m=1}^M \text{SUM2}_m$ measures the overall alignment of all the training images. This 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 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 no much need for non-maximum suppression in inference. In this paper, we use the inference algorithm for detection. For classification, we use the learning algorithm where non-maximum suppression is applied. See Section (4) for details.

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 $\{\mathbf{I}_m, m = 1, \dots, M\}$ represented by (3) to (10), we can pool the probability distribution of $\{(c_{m,i}, i = 1, \dots, n)\}$ as well as the distribution of $\{U_m\}$ over $m = 1, \dots, M$. 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 , $p(\mathbf{I}_m | \mathbf{B}_m)$. The probability density $p(\mathbf{I}_m | \mathbf{B}_m)$ can be used for maximum likelihood learning of \mathbf{B} and $\{\mathbf{B}_m\}$ from training images. It can also be used for finding $\mathbf{B}_m \approx \mathbf{B}$ and scoring the template matching in recognition after \mathbf{B} is learned from training images.

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. $\mathbf{B} = (B_{i,0}, B_{i,1}, i = 1, \dots, n)$ 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 $C = (c_{m,0}, c_{m,1}, i = 1, \dots, n)'$ as a $2n \times 1$ vector. Thus in matrix notation, Equation (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 $p(\mathbf{I}_m | \mathbf{B}_m)$. For the linear representation $\mathbf{I}_m = \mathbf{B}_m C_m + \bar{\mathbf{B}}_m \bar{C}_m$,

$$p(\mathbf{I}_m | \mathbf{B}_m) = p(C_m, \bar{C}_m) |J_m| = p(C_m) p(\bar{C}_m | C_m) |J_m|, \quad (23)$$

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 $p(\bar{C}_m | C_m)$ is the distribution of the residual background given the foreground coefficients.

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

$$p(\mathbf{I}_m | \mathbf{B}_m) = q(\mathbf{I}_m) \frac{p(C_m)}{q(C_m)} = q(\mathbf{I}_m) \frac{p(c_{m,1}, \dots, c_{m,n})}{q(c_{m,1}, \dots, c_{m,n})}, \quad (24)$$

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

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

(1) 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, $q(c_{m,1}, \dots, c_{m,n})$ is multivariate Gaussian.

(2) The distribution of natural image patches. This is the distribution that we shall use in this paper. In particular, we make use of the marginal distribution of filter responses $\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle$ in natural images. It is a heavy tail distribution that allows occasional strong edges. We do not need to specify $q(\mathbf{I}_m)$ beyond this marginal distribution.

(3) The Markov random field distribution that matches the marginal distributions of filter responses $\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle$ in natural images. Such a Markov random field model has been developed by Zhu and Mumford [28]. It is a more explicit form of (2).

(4) 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, Wu, and Mumford [30]. The marginal distributions are pooled from the observed image \mathbf{I}_m over $(x, y) \in D$.

(5) Uniform distribution over the set of images that share certain marginal statistics pooled over image domain. Such a distribution can be related to the Markov random field model in (4) as shown in Wu, Zhu, and Liu [23].

The model (24) 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 $p(C_m)/q(C_m)$.

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

$$\frac{1}{M} \sum_{i=1}^M \log \frac{p(\mathbf{I}_m | \mathbf{B}_m)}{q(\mathbf{I}_m)} = \frac{1}{M} \sum_{i=1}^M \log \frac{p(c_{m,1}, \dots, c_{m,n})}{q(c_{m,1}, \dots, c_{m,n})}. \quad (25)$$

The average log-likelihood converges to $\text{KL}(p(C_m)||q(C_m))$ as $M \rightarrow \infty$, assuming that $p(C_m)$ can be consistently estimated from the training images. Here $\text{KL}(p||q)$ denotes the Kullback-Leibler divergence from p to q . In order to maximize the log-likelihood, we want to choose \mathbf{B} and deform it to $\{\mathbf{B}_m \approx \mathbf{B}\}$ to maximize $\text{KL}(p(C_m)||q(C_m))$, so that we achieve the maximum contrast between the foreground shape and the background texture. $\text{KL}(p(C_m)||q(C_m))$ 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 $\mathbf{B} = (B_i, i = 1, \dots, n)$. The process couples matching pursuit [12] with projection pursuit [7]. The shared sketch learning algorithm is an approximation to it.

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

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

We need to add the following three steps 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 $B \in \text{Dictionary}$, 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 $p_i(c)$ over $\{c_{m,i}, m = 1, \dots, M\}$. We can use such pooled densities $p_1(c), \dots, p_n(c)$ 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. After selecting $\{B_{m,i}, m = 1, \dots, M\}$, 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

$p_i(\mathbf{I}_m) = p_{i-1}(\mathbf{I}_m)p_i(c_{m,i})/q_{i-1}(c_{m,i})$, 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 Equation (24), except that we use $p_{i-1}(\mathbf{I}_m)$ as the current background, and we only replace 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 $B_{m,1}, \dots, B_{m,i-1}$ 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 $\sum_{m=1}^M \log[p_i(\mathbf{I}_m)/p_{i-1}(\mathbf{I}_m)]/M = \log[p_i(c_{m,i})/q_{i-1}(c_{m,i})]/M \rightarrow \text{KL}(p_i(c)||q_{i-1}(c))$. So we want to select B_i that achieves the maximum $\text{KL}(p_i(c)||q_{i-1}(c))$. That is, $\text{KL}(p_i(c)||q_{i-1}(c))$ 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 (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 (24) in the previous subsection. See Figure (3) for an illustration of the shared matching pursuit process.

The computational burden in the shared matching pursuit process lies in the computation of $q_{i-1}(c)$, which requires Monte Carlo sampling from $p_{i-1}(\mathbf{I}_m)$. If we have negative training images from $q(\mathbf{I}_m)$, 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 as an approximation

We can simplify the shared matching pursuit process into a shared sketch process with the following two approximations.

(1) *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 the 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 $\{B_{m,i}, i = 1, \dots, n\}$ do not overlap with each other, or the selected $\{B_{m,i}, i = 1, \dots, n\}$ 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 $(B_{m,i}, i = 1, \dots, n)$, $c_{m,i} = \langle U_m, B_{m,i} \rangle = \langle \mathbf{I}_m, B_{m,i} \rangle$. In practice, we allow small correlations between the elements $(B_{m,i}, i = 1, \dots, n)$.

(2) *Background density.* The current density $p_{i-1}(\mathbf{I}_m)$ results from sequentially updating the densities of $c_{m,1}, \dots, c_{m,i-1}$, starting from $q(\mathbf{I}_m)$. If $B_{m,i}$ has no overlap with $B_{m,1}, \dots, B_{m,i-1}$, then the distribution of $c_{m,i}$ under $p_{i-1}(\mathbf{I}_m)$, i.e., $q_{i-1}(c)$, can be approximated by $q(c)$, 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 $c_{m,i}, i = 1, \dots, n$. Therefore, the pursuit index is $\text{KL}(p_i(c)||q(c))$, where again, $p_i(c)$ is the density pooled over $\{c_{m,i}, m = 1, \dots, M\}$.

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

$$p(\mathbf{I}_m \mid \mathbf{B}_m) = q(\mathbf{I}_m) \prod_{i=1}^n \frac{p_i(c_{m,i})}{q(c_{m,i})}. \quad (26)$$

The pursued model (26) is an approximation to the model (24). $q(c)$ can be pooled from natural images before we start the shared sketch process. We do not need negative examples beyond $q(c)$.

3.4 Parametrization by exponential family model

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

$$p(c; \lambda) = \frac{1}{Z(\lambda)} \exp\{\lambda h(r)\} q(c), \quad (27)$$

where $\lambda > 0$ is the parameter, $r = |c|^2$, and $Z(\lambda) = \int \exp\{\lambda h(r)\} q(c) dc = \mathbb{E}_q[\exp\{\lambda h(r)\}]$ 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 , so model (27) is more encouraging towards large r . The above model can be justified by the maximum entropy principle [4, 30].

Let $p(r; \lambda)$ and $q(r)$ be the densities of $r = |c|^2$ under $p(c; \lambda)$ and $q(c)$ respectively, then $p(c; \lambda)/q(c) = p(r; \lambda)/q(r) = \exp\{\lambda h(r)\}/Z(\lambda)$. More specifically, let $c = (c_0, c_1)$, and let $\varphi = \arctan(c_1/c_0)$ be the local phase, then the conditional distributions of the local phase φ given the local energy r are the same under both $p(c; \lambda)$ and $q(c)$.

Estimating p_i . We estimate $q(c)$ by pooling a histogram from natural images. We estimate $p_i(c)$ from $\{c_{m,i} = \langle \mathbf{I}_m, B_{m,i} \rangle, m = 1, \dots, M\}$ by fitting the density $p(c; \lambda_i)$ to $\{c_{m,i}\}$. Specifically, let us define the mean parameter $\mu(\lambda) = \mathbb{E}_\lambda[h(r)] = \int h(r) \exp\{\lambda h(r)\} q(r) dr / Z(\lambda)$, we estimate λ_i by matching $\mu(\lambda_i) = \sum_{m=1}^M h(r_{m,i}) / M$, so that $\hat{\lambda}_i = \mu^{-1}(\sum_{m=1}^M h(r_{m,i}) / M)$. $\hat{\lambda}_i$ 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 [4]. Thus, we estimate $p_i(c)$ by $p(c; \hat{\lambda}_i)$.

Selecting B_i . The average log-likelihood $\sum_{m=1}^M \log[p(c_{m,i}; \hat{\lambda}_i) / q(c_{m,i})] / M = \text{KL}(p(c; \hat{\lambda}_i) \| q(c))$. 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. To score the template matching, we can compute the log-likelihood ratio

$$\begin{aligned} \log \frac{p(\mathbf{I}_m \mid \mathbf{B}_m)}{q(\mathbf{I}_m)} &= \log \prod_{i=1}^n \frac{p_i(c_{m,i}; \hat{\lambda}_i)}{q(c_{m,i})} \\ &= \sum_{i=1}^n \left[\hat{\lambda}_i h(r_{m,i}) - \log Z(\hat{\lambda}_i) \right], \end{aligned} \quad (28)$$

where $p(\mathbf{I}_m | \mathbf{B}_m)$ is the pursued model (26).

Both $\log Z(\lambda)$ and $\mu(\lambda)$ are one-dimensional functions. We can store their values over a grid of points, and use nearest neighbor linear interpolation for points in between.

The reader is referred to Wu et al. [20] for an information-theoretical perspective of this model, as well as its connection with Markov random fields [30] and adaboost [6]. The reader is also referred to Tu [16] on a generative model constructed from adaboost.

3.5 Transformation of filter responses

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 $p_{\text{on}}(r)$ 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 $p_{\text{off}}(r)$ be the density of r when the Gabor wavelet is off the edge. We assume that $p_{\text{on}}(r)$ has a much longer tail than $p_{\text{off}}(r)$. 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 $q(r) = (1 - \rho_0)p_{\text{off}}(r) + \rho_0 p_{\text{on}}(r)$, and $p_i(r) = (1 - \rho_i)p_{\text{off}}(r) + \rho_i p_{\text{on}}(r)$, that is, both $q(r)$ and $p_i(r)$ are mixtures of on-distribution and off-distribution, with $\rho_i > \rho_0 > 0$. As $r \rightarrow \infty$, $\log[p_i(r)/q(r)] \rightarrow \log(\rho_i/\rho_0) > 0$, 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 saturation level as $r \rightarrow \infty$. This justifies the choice of 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 Equation (1).

The whitening transformation changes a long tail distribution $q(r)$ to a short tail Exponential distribution. With the whitening transformation, under $p(c; \lambda)$ of (27), $h(r) \sim \text{Exp}(1 - \lambda)$, which is an Exponential distribution with $\mu(\lambda) = 1/(1 - \lambda_i)$ and $Z(\lambda) = 1/(1 - \lambda_i)$. λ_i can be estimated by $\hat{\lambda}_i = 1 - M / \sum_{m=1}^M h(r_{m,i})$.

3.6 Active mean vector and active correlation

We can replace the log-likelihood score $\log[p(\mathbf{I}_m | \mathbf{B}_m)/q(\mathbf{I}_m)]$ in Equation (28) 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

$$\langle \mathbf{I}_m | V_m \rangle = \sum_{i=1}^n \theta_i \text{whiten}(|\langle \mathbf{I}_m, B_{m,i} \rangle|^2)^{1/2}. \quad (29)$$

We assume that \mathbf{I}_m is normalized. The reason we use whitening transformation defined by Equation (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 (29) can be considered the length that \mathbf{I}_m projects on V_m . We also filter out the local phase information in (29), because phase is irrelevant for shape. We call (29) 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 with \mathbf{I}_m .

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

$$\sum_{i=1}^m \langle \mathbf{I}_m | V_m \approx V \rangle = \sum_{i=1}^n \left[\theta_i \sum_{m=1}^M \text{whiten}(|\langle \mathbf{I}_m, B_{m,i} \rangle|^2)^{1/2} \right]. \quad (30)$$

The algorithm for learning $\mathbf{B} = (B_i, i = 1, \dots, n)$ and $\Theta = (\theta_i, i = 1, \dots, n)$ is essentially the same as the shared sketch algorithm in Subsection (2.5). The resulting $V = \sum_{i=1}^n \theta_i B_i$ can be consider the mean shape of $\{\mathbf{I}_m, m = 1, \dots, M\}$. We call it the active mean vector. Geometrically, V points to the center of the cluster formed by $\{\mathbf{I}_m, m = 1, \dots, M\}$.

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 $h(\text{MAX}1_m(x, y, s, \alpha))$ in natural images or in the observed image \mathbf{I}_m . If the former is no much greater than the latter, we should stop the algorithm. 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 used in all the experiments in this paper (unless otherwise stated). Size of Gabor wavelets = 17×17 . (x, y) is sub-sampled every 2 pixels. The orientation α takes $A = 15$ equally spaced angles in $[0, \pi]$. The orthogonality tolerance is $\epsilon = .1$. The threshold is $T = 16$ 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. The shift of orientation $\delta_{m,i} \in [-b_2, b_2] = \{-1, 0, 1\} \times \pi/15$.

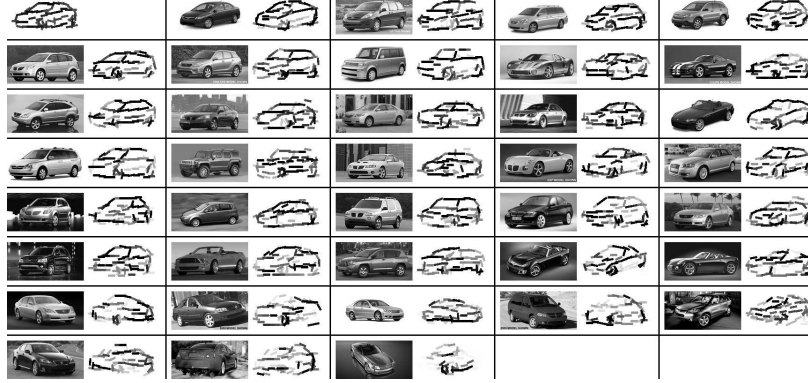


Figure 5: Experiment 1.1. The 37 training images are 82×164 . The first block displays the learned active basis consisting of 60 elements. Each element is symbolized by a bar. The rest of the blocks display the observed images and the corresponding deformed active bases. The images are displayed in the descending order of the log-likelihood ratio, which scores the template matching.

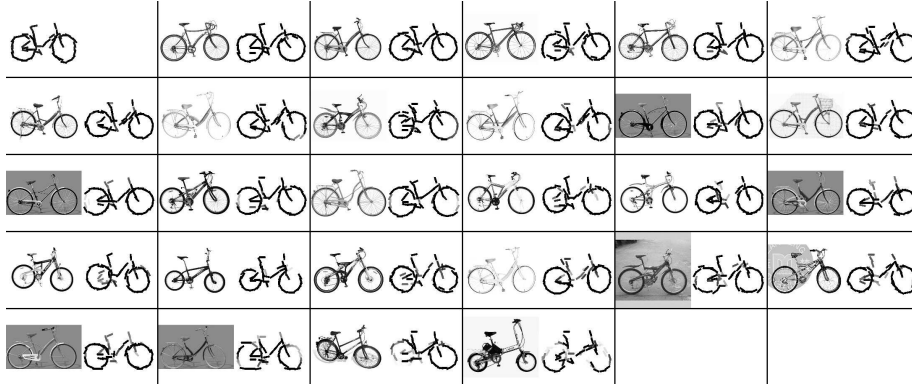


Figure 6: Experiment 1.2. The 27 images are 180×180 . Number of elements is 60.

Experiment 1. In this experiment, we take $h(r) = \text{threshold}(r)$, as defined by Equation (13), so that there is no need to pool $q(r)$. Other designs of $h(r)$ produce similar results.

In Experiment 1.1, we apply the shared sketch algorithm to a training set of $M = 37$ car images. The car images are 82×164 . Figure (6) displays the results, where $n = 60$. The first block displays the learned active basis $\mathbf{B} = \{B_i, i = 1, \dots, n = 60\}$, where each B_i is represented symbolically by a bar at the same 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 $\mathbf{B}_m = (B_{m,i}, i = 1, \dots, n)$. 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.



Figure 7: Experiment 1.3. The 15 images are 179×112 . Number of elements is 50.

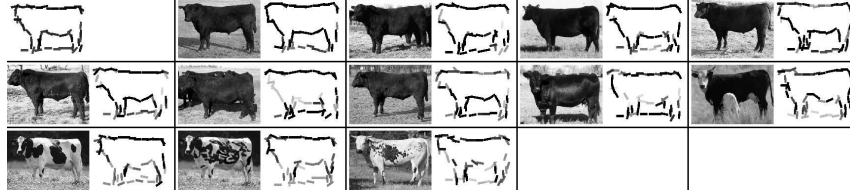


Figure 8: Experiment 1.4. The 12 images are 120×167 . Number of elements is 50.

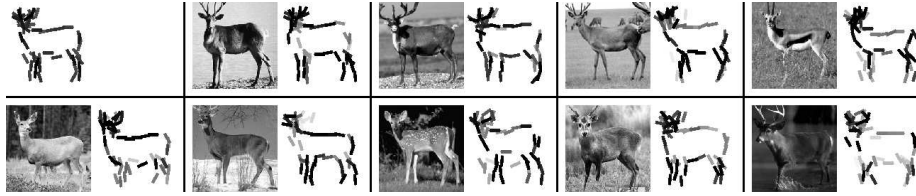


Figure 9: Experiment 1.5. The 9 images are 122×120 . Number of elements is 50.

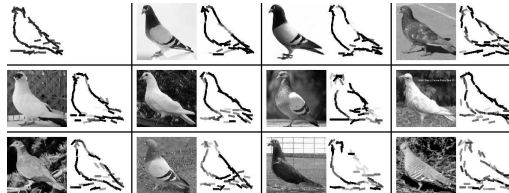


Figure 10: Experiment 1.6. The 11 images are 133×140 . Number of elements is 50.

Figures (6) - (10) display more examples, where the results are obtained by the same algorithm.

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.

Later we shall show that our method can be extended so that we can learn from non-aligned images and 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.



Figure 11: Experiment 2.1. (a) Testing image. The recognition algorithm is run on 10 resolutions, from 270×360 to 432×536 . (b) Superposed with sketch of the 60 elements of the deformed active basis at the optimal resolution 378×504 . The bounding box of the template is 82×164 .

Experiment 2. Figure (11.a) displays the observed image in Experiment 2.1. The deformable template is learned in Experiment 1.1. The bounding box is 82×164 . We run the recognition algorithm on 10 resolutions of the testing image, from 270×360 to 432×536 . Figure (11.b) displays the superposed sketch of $(B_{\hat{x}_i, \hat{y}_i, s, \hat{\alpha}_i}, i = 1, \dots, n = 60)$ at the optimal resolution 378×504 . Again, we let $h(r) = \text{threshold}(r)$, the same as in Experiment 1.

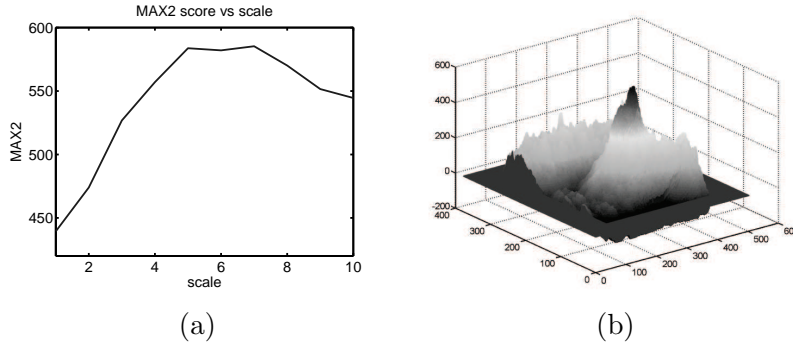


Figure 12: Experiment 2.1. (a) MAX2 scores at resolutions 1 to 10. (b) SUM2 map at the optimal resolution 7.

Figure (12.a) displays the MAX2 scores at the 10 resolutions. The maximum is achieved at the

7th resolution. Figure (12.b) displays the SUM2 map at this optimal resolution. Recall that SUM2 is the log of the likelihood ratio. If we use the likelihood ratio, then the value at the maximum of the SUM2 map far exceeds the values of other pixels.

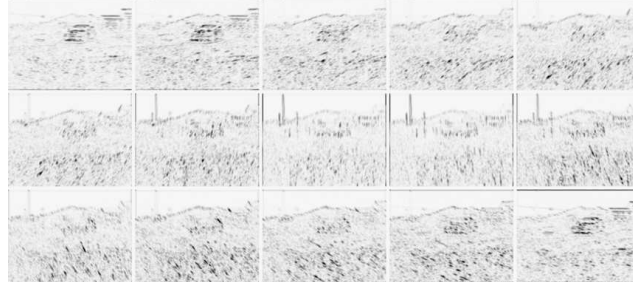


Figure 13: Experiment 2.1. Negative square root of SUM1 maps at the optimal resolution. There are 15 orientations.

Figure (13) displays $-\text{SUM1}(x, y, s, \alpha)^{1/2}$. There are 15 orientations, so there are 15 SUM1 maps.

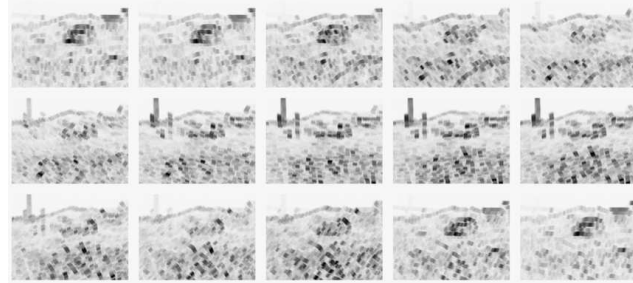


Figure 14: Experiment 2.1. Negative square root of MAX1 maps at the optimal resolution. There are 15 orientations.

Figure (14) displays $-\text{MAX1}(x, y, s, \alpha)^{1/2}$. There are 15 MAX1 maps. These MAX1 maps are blurred versions of the SUM1 maps. They themselves are not very meaningful unless they are combined into the SUM2 map using the active basis model learned by Experiment 1.

Figure (15.a) displays the observed image in Experiment 2.2. The deformable template is learned in Experiment 1.2. The bounding box is 180×180 . We run the recognition algorithm on 10 resolutions of the testing image, from 320×240 to 500×375 . Figure (15.b) displays the superposed sketch of $(B_{\hat{x}_i, \hat{y}_i, s, \hat{\alpha}_i}, i = 1, \dots, n = 60)$ at the optimal resolution 400×300 . Because the pose of the bike in the testing image is not entirely the same as the poses in the training images, the sketched bike in Figure (15.b) misses part of the front wheel.

Figure (16) displays the observed image in Experiment 2.3. The deformable template is learned in Experiment 1.3. The bounding box is 179×112 . We run the recognition algorithm on 10 resolutions of the testing image, from 192×220 to 322×368 .

Figure (17) displays the superposed sketch at each of the 10 resolutions.

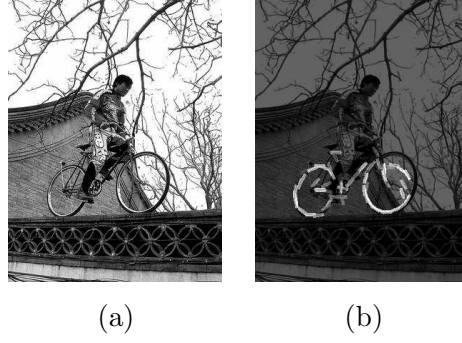


Figure 15: Experiment 2.2. (a) Testing image. The algorithm is run on 10 resolutions, from 320×240 to 500×375 . (b) Superposed with sketch of 60 elements of the deformed active basis at the optimal resolution 400×300 . The bounding box of the template is 180×180 .



Figure 16: Experiment 2.3. Testing image.



Figure 17: Experiment 2.3. Superposed sketch of 50 elements of the deformed active basis at each of the 10 resolutions, from 192×220 to 322×368 . The bounding box is 179×112 .

Negative experience in Experiment 2. Our method can sometimes be distracted by cluttered edges in the background. We may need to combine templates at multiple resolutions to overcome this problem.

4.3 Classification by learning algorithm

In this section, we evaluate our method on classification tasks and compare the ROC curves.

Scoring testing images. We learn the active basis $\mathbf{B} = (B_i, i = 1, \dots, n)$ and estimate $\Lambda = (\lambda_i, i = 1, \dots, n)$ (or $\Theta = (\theta_i, i = 1, \dots, n)$ for active correlation) from the training images. Then for

each testing image \mathbf{I}_m , we can compute its score SUM2_m according to Equation (28). This can be done using the same algorithm as the shared sketch algorithm in Subsection (2.5), except that we remove the selection of (x_i, y_i, α_i) and the computation of λ_i in the selection step, because B_i and λ_i are already available. We obtain the ROC curves based on SUM2_m scores.



Figure 18: Experiment 3.1. Results obtained by fitting the active basis model with sigmoid transformation. The saturation level is 6. The 43 images are 127×85 . The number of basis elements 40.

Experiment 3. Figure (18) displays the 43 positive training images in Experiment 3.1. It also displays the corresponding $\mathbf{B}_m = (B_{m,i}, i = 1, \dots, n)$, $n = 40$. The results are obtained with $h(r) = \text{sigmoid}(r)$. The saturation level $\xi = 6$. The $q(r)$ is pooled from 157 negative training images. The bounding boxes of the training images and testing images are all given. The sizes of the training and testing image patches are all 127×85 .

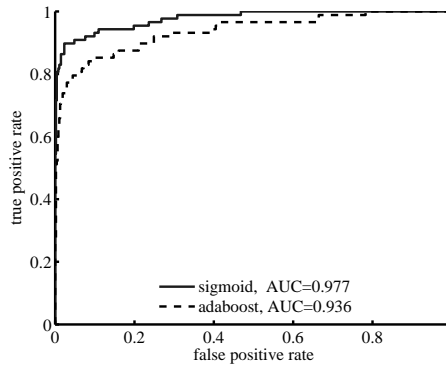


Figure 19: Experiment 3.1. The solid curve is ROC of active basis model with sigmoid transformation. The dashed curve is ROC of the adaboost method where the weak classifiers are based on thresholding the filter responses in SUM1 maps.

We then test on a separate data set with 88 positives and 474 negatives. The solid curve of Figure (19) is the ROC curve for the active basis model with sigmoid transformation. The AUC

(area under curve) is .977.

For comparison, we also display the ROC curve for the adaboost method. The weak classifiers are obtained by thresholding the filter responses $\text{SUM1}_m(x, y, s, \alpha)$ computed in Step 1(a) of the shared sketch algorithm in Subsection (2.5). The threshold for each weak classifier is selected from a dense grid of points. We select 80 weak classifiers using the adaboost method. The dashed curve of Figure (19) is the ROC curve for the adaboost method. The AUC is .936.

We also try another type of weak classifiers, which are based on thresholding the local maxima of filter responses, i.e., $\text{MAX1}_m(x, y, s, \alpha)$ obtained in Step 1(b) of the shared sketch algorithm. We select 40 weak classifiers using the adaboost method. The AUC is .943. If we increase the number of weak classifiers, the AUC starts to decrease, suggesting that the classifier starts to overfit.

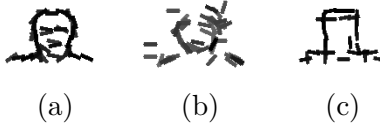


Figure 20: Experiment 3.1. (a) The template learned by the active basis model. (b) Learned by the adaboost method. (c) Learned from tiny training images.

Figure (20.a) displays the template learned by the active basis model. Figure (20.b) displays the template learned by the adaboost method, where a weak classifier is illustrated by a bar that symbolizes the corresponding Gabor wavelet.

We also try the active basis model with transholding transformation. The thresholding level is 16. The AUC is .941. The reason that it does not do as well as sigmoid transformation may be due to the fact that the thresholding level is too high, so that the strong edges in the negative examples contribute too much to the log-likelihood ratio score SUM2_m . We then try the active basis model with whitening transformation with $q(r)$ or $F(r)$ pooled from negative training images. The AUC is .965.

We also try active correlation. The AUC for thresholding transformation is .971. The AUC for whitening transformation is .974. The improvement over log-likelihood may be due to the fact that we take squared root of $\text{whiten}(r)$ in active correlation. This discounts strong edges in negative examples.

Combining templates at multiple resolutions. A low saturation level in the sigmoid model is helpful to discount strong edges in the background or in the negative examples. But this may cause another problem: a large number of weak edges in the cluttered background may collectively produce high SUM2 score. One way to alleviate this problem is to combine templates at multiple resolutions, because the weak clutter edges cannot survive long in the scale space [19], so they will not produce high SUM2 score at lower resolution.

In addition to the template displayed in Figure (18), we learn another template using Gabor wavelets at a scale twice as large as the original scale. We select $n = 20$ basis elements at this scale. Then in testing, we use the sum of the SUM2_m scores obtained at the two scales. This leads to an

AUC = .982.

Adaptive texture background. In scoring a testing image \mathbf{I}_m , the SUM2_m is computed by $\sum_{i=1}^n \log[p_i(c_{m,i}; \hat{\lambda}_i)/q(c_{m,i})]$, according to Equation (28), where $q(c)$ is pooled from natural images or negative training images. We can change $q(c)$ to a background texture model fitted specifically to \mathbf{I}_m . Specifically, for each image \mathbf{I}_m , and for each orientation α , we can fit a model $q_m(c; \lambda_\alpha)$ to $\{\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle, \forall (x, y) \in D\}$. The distribution $q_m(c; \lambda_\alpha)$ is again the exponential family model of the form specified by Equation (27). The maximum likelihood estimate $\hat{\lambda}_\alpha = \mu^{-1}(\sum_{(x,y) \in D} h(|\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle|^2)/|D|)$. Then we compute the SUM2_m score by

$$\begin{aligned} \text{SUM2}_m &= \sum_{i=1}^n \log[p_i(c_{m,i}; \hat{\lambda}_i)/q_m(c_{m,i}; \hat{\lambda}_{\alpha_i})] \\ &= \sum_{i=1}^n [(\hat{\lambda}_i h(|\langle \mathbf{I}_m, B_{m,i} \rangle|^2) - \log Z(\hat{\lambda}_i)) \\ &\quad - (\hat{\lambda}_{\alpha_i} h(|\langle \mathbf{I}_m, B_{m,i} \rangle|^2) - \log Z(\hat{\lambda}_{\alpha_i}))], \end{aligned} \quad (31)$$

where α_i is the orientation of $B_{m,i}$. That is, we score the template matching against the adaptive texture background summarized by $q_m(c; \hat{\lambda}_{\alpha_i})$. The marginal distributions pooled over $\{\langle \mathbf{I}_m, B_{x,y,s,\alpha} \rangle, \forall (x, y) \in D\}$ for different α and s have been used by Zhu, Wu, and Mumford [30] for modeling textures. This new SUM2_m score leads to an AUC = .983.

Learning from single training image. We can also train the sigmoid model on a single positive training image. We use the 13th image in Figure (18). It has a clean background. We set $b_1 = b_2 = 0$ in learning, that is, we do not allow the B_i to perturb. Then we set b_1 and b_2 at their default values for testing. The AUC is .926.

Learning from tiny images. We can also train the model on tiny images. We zoom out the positive training images by a factor of 10, so the image size is 12×8 . In order to learn from such tiny images, we then zoom in these 12×8 images by a factor of 10, so the image size becomes 120×80 . Then we apply the shared sketch algorithm on these 120×80 images. The learned template is shown in Figure (20.c). Then we test on the testing images, which also go through the same zooming out and zooming in operations. The AUC is .957.

We also did another experiment on horse images. Figure (21) displays the learning results of Experiment 3.2. There are 31 positive training examples. They are 150×120 . The number of basis elements is $n = 40$. There are 249 positive testing images. We use the same sets of negative training and testing images as in Experiment 3.1 (with resizing to match the size of the positive examples). The AUC scores are as follows: sigmoid = .987, whiten = .982, whiten with active correlation = .984, threshold = .981, threshold with active correlation = .985, combining two templates = .988, adaptive texture background = .989, single image learning = .980, tiny image learning = .956, adaboost method by thresholding the SUM1 maps = .937, adaboost method by thresholding the MAX1 maps = .957.

We did a third experiment on butterfly images. Figure (22) displays the learning results of Experiment 3.3. There are 33 positive training examples. They are 150×100 . The number of basis elements is $n = 40$. There are 191 positive testing images. We use the same sets of negative

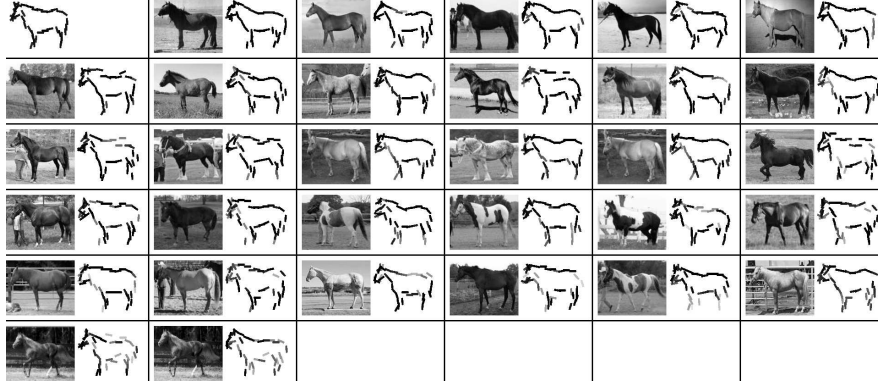


Figure 21: Experiment 3.2. The 31 images are 150×120 . The number of basis elements is 40.

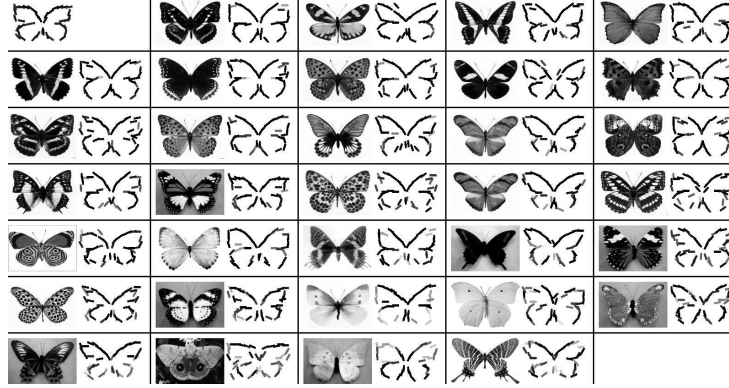


Figure 22: Experiment 3.3. The 33 images are 150×100 . The number of basis elements is 40.

training and testing images as in Experiment 3.1. The AUC for sigmoid model is .999. The AUC for adaboost by thresholding MAX1 maps is .994.

Experiment 3 focuses on the situation where the number of positive training images is relatively small. Compared to adaboost, the active basis model applies non-maximum suppression instead of re-weighting before selecting the next basis element. The active basis model does not require negative examples (except pooling a marginal histogram from natural images). Moreover, the generative model can be conveniently used for unsupervised learning as we shall show in the following sections.

4.4 Geometric transformation of template

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

Figure (23) shows three examples. The bicycle template is learned in Experiment 1, where we use the sigmoid transformation. We then transform it into a collection of templates as different

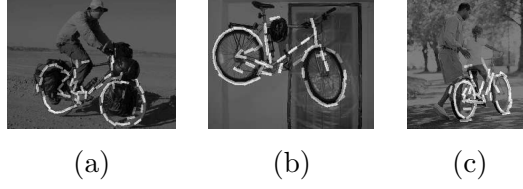


Figure 23: Experiment 4.1. (a) The number of elements is 60. The image size is 252×320 . The scale factor is 1.4. The rotation is $1 \times \pi/15$. The aspect factor is 0.9. (b) The image size is 200×250 . The scale factor is 1.4. The rotation is $1 \times \pi/15$. The aspect factor is 1. (c) The image size is 248×232 . The scale factor is 1.2. The rotation is $-1 \times \pi/15$. The aspect factor is 0.6.

scales, orientations, and aspect ratios. After that, we use these templates to detect the objects by template matching, as in Experiment 2. Finally, we choose the transformed template that gives the best match, and superpose it on the input image.

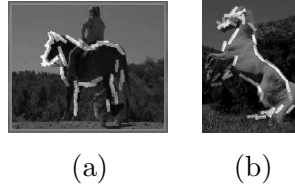


Figure 24: Experiment 4.2. (a) The number of elements is 40. The image size is 138×168 . The scale factor is 1. The rotation is $0 \times \pi/15$. The aspect factor is 0.8. (b) The image size is 192×144 . The scale factor is 1. The rotation is $4 \times \pi/15$. The aspect factor is 1.3.

Figure (24) shows another example with the horse template, learned in Experiment 3.

Negative experience in Experiment 4. 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 very good fit. Also, 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 each part-template has its own geometric transformation. We shall study the composition of multiple part-templates later.

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 Image alignment

For the training image patches $\{\mathbf{I}_m, m = 1, \dots, M\}$ defined on the same bounding box, such as those in the previous section, we can define their overall alignment by

$$\text{ALIGN}(\mathbf{I}_m, m = 1, \dots, M) = \sum_{m=1}^M \text{MATCH}(\mathbf{I}_m, \mathbf{B}), \quad (32)$$

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 Subsection (2.5), and $\text{ALIGN}(\mathbf{I}_m, m = 1, \dots, M) = \sum_{m=1}^M \text{SUM2}_m$, where the SUM2_m scores are output by the algorithm.

When the training images $\{\mathbf{I}_m, m = 1, \dots, M\}$ 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

$$\text{ALIGN}(\mathbf{I}_m[\text{box}(x_m, y_m)], m = 1, \dots, M), \quad (33)$$

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 $\{(x_m, y_m), m = 1, \dots, M\}$, estimate (\mathbf{B}, Λ) from $\{\mathbf{I}_m[\text{box}(x_m, y_m)], m = 1, \dots, M\}$ using the shared sketch algorithm in Subsection (2.5).
- (2) Detection: Given (\mathbf{B}, Λ) , estimate (x_m, y_m) from each \mathbf{I}_m using the inference algorithm in Subsection (2.4). (x_m, y_m) achieves the maximum of the SUM2 map.



Figure 25: Experiment 5a.1. The bounding box of the first image is given. The size of the bounding box is 136×140 . The number of elements in the active basis is 60.

Experiment 5a: In this experiment, we initialize the algorithm by specifying the bounding box for the first training image. Then we estimate (\mathbf{B}, Λ) 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 reset b_1 and b_2 to their default values, and iterate Step (2) and Step (1) described above.

In Step (2), we search over 9 different resolutions (from 0.8 to 1.2 times the input image size). We crop $\mathbf{I}_m[\text{box}(x_m, y_m)]$ from the optimal resolution.

We run the algorithm for three iterations. Figures (25) - (30) display some examples.



Figure 26: Experiment 5a.2. The bounding box is 116×116 . Number of elements is 50.



Figure 27: Experiment 5a.3. The bounding box is 115×161 . Number of elements is 50.



Figure 28: Experiment 5a.4. The bounding box is 94×99 . Number of elements is 30.



Figure 29: Experiment 5a.5. The bounding box is 94×138 . Number of elements is 60.



Figure 30: Experiment 5a.6. The bounding box is 134×148 . Number of elements is 60.

Experiment 5b. This experiment is about pairwise alignment, that is, we align two images so that they overlap on the common parts. We learn a template from the first image with no activity and without any given bounding box. Then we restore the activity and scan the learned template on the second image using the inference algorithm as in Experiment 2. When the template is partially out of the bound of the second image, we set the responses of those elements that are out of the bound to 0. We scan the template over 7 resolutions of the second image, from .7 to 1.3 times the size of the input image. At the optimal location and resolution, we deform the template and sketch the second image. In this experiment, we use active correlation for learning and inference.

Figure (31) displays the pairwise alignment. (a) shows the first image and the template learned from this image. (b) shows the second image, and the sketch of this image by the learned template, where the intensities of the elements are proportional to the square roots of the corresponding responses from the second image. Figure (32) displays the alignment where we switch the order of the two images.



Figure 31: Experiment 5b.1. The number of elements is 50.

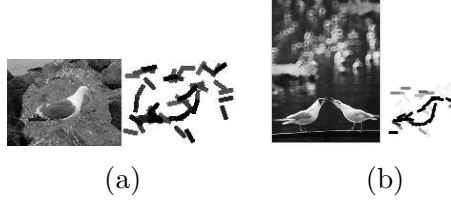


Figure 32: Experiment 5b.2. The number of elements is 50.

Figure (33) displays the result of aligning the first frame of a vide sequence to the 15th frame of the same sequence. The image sequence is cropped from the PETS 2006 benchmark data and is to be used again in Experiment 6.



Figure 33: Experiment 5b.3. The number of elements is 80.

Experiment 5c. 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 example. Figure (34) 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 3 iterations of the algorithm used in Experiment 5a. As in the pairwise alignment, we allow the template to be partially out of the bounds of the images in the detection step.

Negative experience in Experiments 5a-c. 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 and 5c, if the objects do not occupy significant portions of the training images, our method may fail to establish correct alignment.



Figure 34: Experiment 5c. In each example, the first template is the starting template. The second template is learned after 3 iterations. The number of elements of the active basis is 60 in the left example, and 50 in the right example.

5.2 Learning part-templates

The algorithm in Experiment 5a can be used to learn part-templates from training images. In Experiment 5d, we start from a large number of patches cropped from the training images, and for each starting patch, we learn a templates using the same algorithm as in Experiment 5a. Here we use active correlation 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.

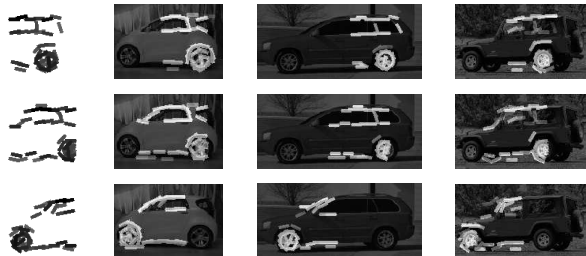


Figure 35: Experiment 5d. The top three part-templates. The size of the bounding box is 100×100 . The number of elements is 40. The allowed activity in location is up to 3 pixels. The allowed activity in orientation is up to $\pi/15$.

Figure (35) 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.

Figure (36) 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 5d. 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



Figure 36: Experiment 5d. The top two part-templates learned after the sizes of the input images are doubled. The parameters are the same as in Figure (35)

whole images. If the bounding boxes are not given as in Experiment 5a, we can simultaneously learn multiple parts while restricting their relative positions, and this is very much like a recursion of Experiment 5a or 5c.

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 $(\mathbf{I}_t, t = 1, \dots, M)$ be 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 $\{\mathbf{J}_t^{(v)}, t = 1, \dots, M\}$ 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

$$\text{ALIGN}(\mathbf{J}_t^{(v)}, t = 1, \dots, M) \quad (34)$$

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

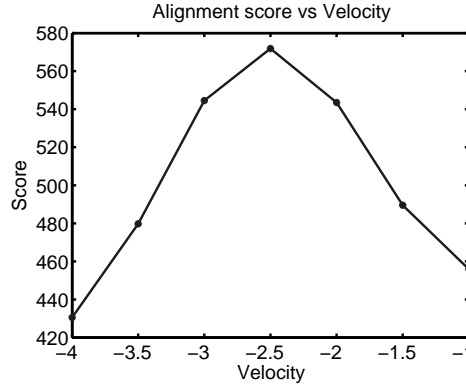


Figure 37: Experiment 6.1. Alignment scores at different speeds of the optimal direction.

In our experiment, we use active correlation to evaluate the alignment score (34). Before computing this score, we need to do 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 using the background subtracted SUM1 maps.

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. We try 5 different directions v_x/v_y , and at each direction, we try 7 different speeds. Figure (37) displays the alignment scores at different speeds of the optimal direction.

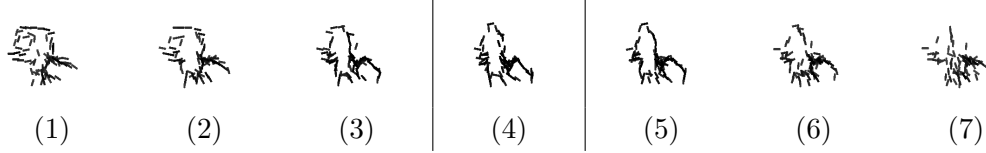


Figure 38: Experiment 6.1. Learned templates at different speeds. (4) is the one at the optimal speed. There are 19 frames of size 204×258 . Number of elements is 70.

Figure (38) displays the templates learned at different speeds of the optimal direction. Template (4) is learned at the optimal speed. The number of basis elements is 70.

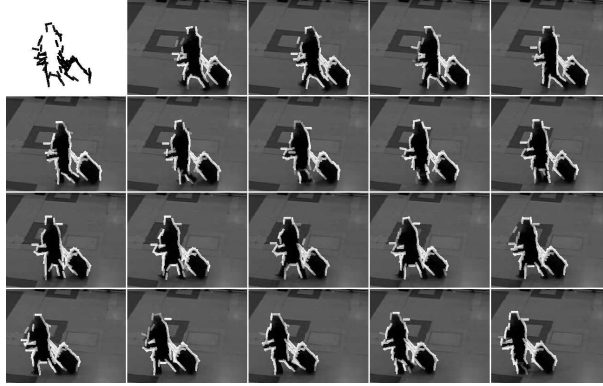


Figure 39: Experiment 6.1. Learned template and superposed sketch for each frame at the optimal speed and direction.

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

Figure (40) displays results for another example.

The learned templates can be used for tracking. Compared to fully unsupervised learning, it provides a more reliable way to learn templates and their parts.

6 Clustering and Local Learning

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

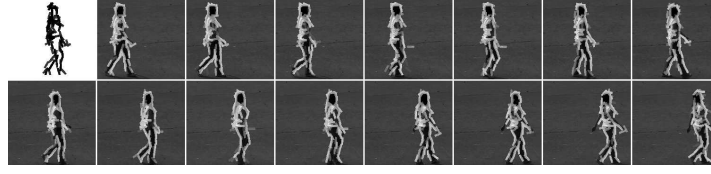


Figure 40: Experiment 6.2. Learned template and superposed sketches at the optimal speed. The image frames are 180×186 . Number of elements is 80.

6.1 EM and K-mean

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

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

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

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

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

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

(1) In Step 2(a), we find (x_i, y_i, α_i) by maximizing $\sum_{m=1}^M z_m^{(k)} h(\text{MAX1}(x, y, s, \alpha))$, which is a weighted sum.

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

$$\hat{\lambda}_i = \mu^{-1} \left(\frac{\sum_{m=1}^M h(r_{m,i}) z_m^{(k)}}{\sum_{m=1}^M z_m^{(k)}} \right), \quad (35)$$

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

(3) After we finish the algorithm, we attach a superscript (k) to the resulting SUM2_m and \mathbf{B} .

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 $k = 1, \dots, K$.

Experiment 7. Figure (41.a) displays the learned templates $\mathbf{B}^{(k)}, k = 1, \dots, 3$ from the mixture of the three sets of positive training images in Experiment 3. The image size is 120×150 . For the head and shoulder images and the butterfly images, we resize them to be 120×150 . The number

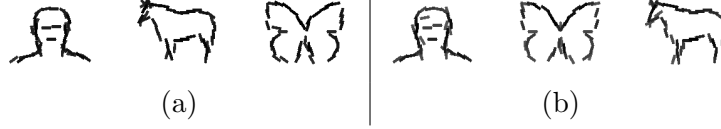


Figure 41: Experiment 7.1. Learned templates from the mixture of the three sets of positive training images in Experiment 3. Image size is 120×150 . Number of elements in each template is 40. Number of iteration is 4. (a) EM. (b) K-mean.

of elements in each $\mathbf{B}^{(k)}$ is 40. We run the EM algorithm for 4 iterations. The EM algorithm can easily separate the three clusters. Only one error is made, where a horse image is classified as a head-shoulder image.

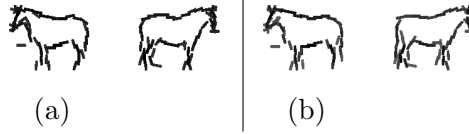


Figure 42: Experiment 7.2. Learned templates for 58 images of horses facing two different directions. Image size is 120×150 . Number of elements in each template is 50. (a) EM with 4 iterations. (b) K-mean with 8 iterations.

Figure (41.a) displays the learned templates $\mathbf{B}^{(k)}$, $k = 1, 2$ from a mixture of 58 images of horses facing different directions. The image size is 120×150 . The number of elements in each $\mathbf{B}^{(k)}$ is 50. We run the EM algorithm for 4 iterations. The EM algorithm easily separates the two clusters.

K-mean clustering. We can pose the clustering problem as the following alignment problem: find $\{(z_m^{(k)}, k = 1, \dots, K), m = 1, \dots, M\}$ to maximize

$$\sum_{k=1}^K \text{ALIGN}(\mathbf{I}_m, z_m^{(k)} = 1), \quad (36)$$

where $\text{ALIGN}\{\mathbf{I}_m, z_m^{(k)} = 1\}$ is the alignment score of the k -th cluster. The computation of the alignment score 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 multiple 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. This leads to a K-mean algorithm. It is a greedy algorithm that maximizes (36), and it iterates the following two steps:

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

(2) Given $\{\mathbf{B}^{(k)}, \theta^{(k)}, k = 1, \dots, K\}$, classify each image \mathbf{I}_m to a cluster k_* that maximizes $\langle \mathbf{I}_m | V_m^{(k)} \rangle$ over all $k = 1, \dots, K$. 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: we let $z_m^{(k_*)} = 1$ if k_* achieves the maximum of $\text{SUM2}_m^{(k)}$ among all $k = 1, \dots, K$, and we set the the rest of $z_m^{(k)}$ to be 0.

(2) Change the M-step: for each $k = 1, \dots, K$, we compute SUM2_m and estimate \mathbf{B} and Θ for each cluster k using the shared sketch algorithm that maximizes the active correlation. The algorithm is described in Section (3.6).

Figure (41.b) and Figure (42.b) display the learned templates $\mathbf{B}^{(k)}$, $k = 1, \dots, K$ 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.

6.2 Local learning of representative templates

The EM and the K-mean clustering methods assume that the number of clusters is given. They also assume that the clusters should be distinct from each other. The results of the algorithms seem to be severely dependent on initialization. To address these problems, we develop a local learning scheme by modifying the K-mean method in the previous subsection. In this scheme, we learn a local representative template around each input image, by recruiting a small number of nearest neighbors. With the local templates and their nearest neighbors, we can then perform trimming and merging for clustering.

In Experiment 8, we learn local representatives in an ensemble of 123 images of animal heads. Around each image, we learn a template from this image and its K nearest neighbors, using an iterative scheme based on the active correlation. Specifically, we first learn a template for this single image where no activity is allowed. Then using the learned template, we find K nearest neighbors by active correlation, where activity is restored. After that we re-learn the template from this image and its K nearest neighbors. The weight for each of the K nearest neighbors is 1, whereas the weight for this image is ρK . In our experiment, $K = 5$, and $\rho = 1/3$. Then again, we find the K nearest neighbors, and iterate. After learning all the templates, we trim them to satisfy the constraint that the neighbors of the remaining templates should not overlap (this may be too aggressive). This leaves 16 templates.

Figure (43) shows the 16 templates obtained by local learning. They are ordered by the alignment scores computed from the K nearest neighbors.

Figure (44) shows the top four templates. For each template, we also show its five nearest neighbors and their sketches.

Figure (45) shows another four templates. Although these templates are less clear and clean



Figure 43: Experiment 8. The 16 representative templates locally learned by active correlation. They are ordered by the total alignment scores. 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 $\pi/15$.

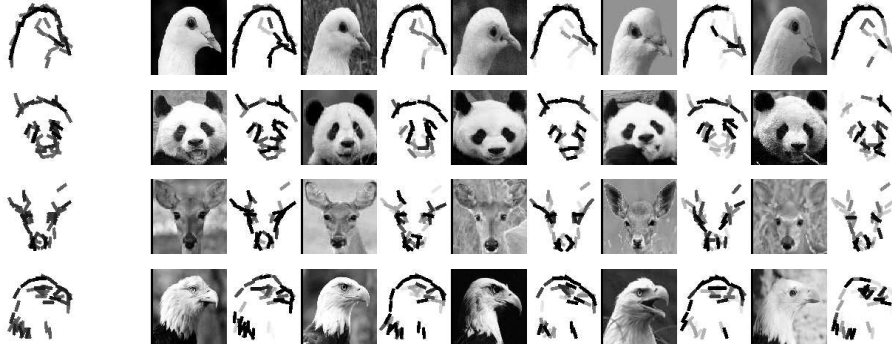


Figure 44: Experiment 8. The top four templates and their neighbors.

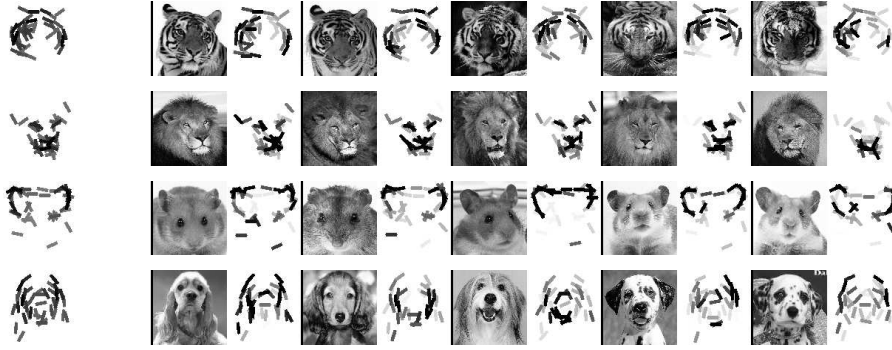


Figure 45: Experiment 8. Another four templates and their neighbors.

than the top four templates, they still manage to recruit nearest neighbors that are from identical categories.

We can then use these 16 templates to initialize the K-mean clustering. The results are meaningful. With local learning, we can also perform hierarchical clustering by merging clusters.

Such a local learning scheme identifies local dimensions in the image ensemble. It also furnishes us with a local metric measured by active correlation. It is unclear whether we could use support vector machine [2] to select the positive as well as negative templates, and use these support deformable templates for classification.

We have not had much experience with local learning. We tried it on digits, and it produced meaningful results, although there is some mixing between different categories, such as “4” or “9”.

7 Synthesis by Multi-scale Gabors and DoGs

In most of the experiments so far, we only analyze the images at a single scale. This is sometimes not enough for detection and classification, where we should learn templates at multiple scales and combine the template matching scores. For instance, Figure (46) displays the template learned from the deer images, where the basis elements are twice the size of the elements in Experiment 1. We can combine these two templates for detection and classification.

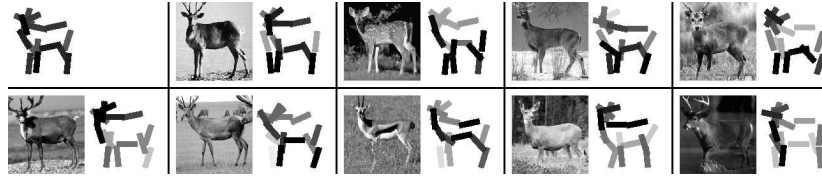


Figure 46: The size of the Gabor wavelet elements are twice the size of the elements in Experiment 1. The number of elements is 15.

In computer vision, researchers often distinguish between edges and regions. Actually, these are two relative concepts in the frequency domain. While edges can be captured by high frequency Gabor wavelets, the regions 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.

In Experiment 9, we select the wavelet elements of active basis from a dictionary of Gabor wavelets and Difference of Gaussian (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 record their responses, we use matching pursuit [12] 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 project coefficients.

Figure (47) displays the select Gabor and DoG elements. The Gabor elements are illustrated by bars at different sizes, and 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 (48) displays the reconstructed images. The DoG elements are necessary to account for the large regional contrasts.

Figures (49) - (52) display more examples. Ideally, the large Gabor elements gauge the breadths of the edges, while the small Gabor elements gauge the sharpness of the edges. The large DoG



Figure 47: 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.



Figure 48: 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 .

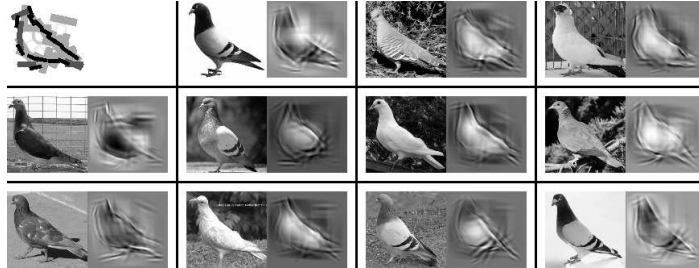


Figure 49: Experiment 9.2. The first block displays all the 50 selected Gabor and DoG elements. The remaining blocks display the original 95×100 images and the corresponding reconstructed images.

elements gauge the sizes of the regions, which are to be contoured by the Gabor elements. It is unclear whether we could use large Gabor and DoG elements to substitute the region concepts, so that we do not have to do image segmentation.

8 Composing Multiple Part-Templates

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



Figure 50: Experiment 9.4. 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.

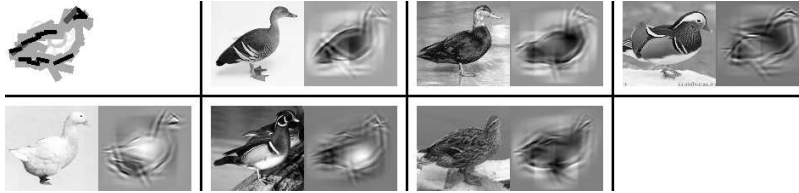


Figure 51: Experiment 9.5. The first block displays all the 40 selected Gabor and DoG elements. The remaining blocks display the original 100×110 images and the corresponding reconstructed images.



Figure 52: Experiment 9.7. 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.

8.1 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.

Figure (53.a) displays an image of size 330×496 . Figure (53.b) displays the superposed sketch. The template is learned in Experiment 1.2 from the bicycle images. See Figure (6). We split the bicycle template in Figure (6) horizontally into two part-templates. The bounding box for the



Figure 53: 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.

part-template of the front wheel is 112×126 . The bounding box for the part-template of the back wheel is 86×76 . We allow the two part-templates to locally shift horizontally, so these two part-templates make up a multi-basis. We then fit this multi-basis to the tandem bike in Figure (53.a) and obtain the sketch in Figure (53.b).

Given the two part-templates, the inference can be accomplished by alternating the sum maps and max maps as illustrated by Figure (54). Here we have two SUM2 maps, one for each part-template. On top of each SUM2 map, there is also a MAX2 maps. 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 retrieves the location of the whole template, and then retrieves the locations of the part-templates, and finally the elements of the part-templates.

Inference by recursive sum-max maps

Up-1 For $j = 1, 2$, compute $\text{SUM2}(x, y, j)$ using the inference algorithm of Subsection (2.4).

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

$$\text{MAX2}(x, y, j) = \max_{\substack{-b_x \leq \Delta x \leq b_x \\ -b_y \leq \Delta y \leq b_y}} \text{SUM2}(x + \Delta x, y + \Delta y, j), j = 1, 2. \quad (37)$$

Up-3 For all (x, y) and $j = 1, 2$, compute

$$\text{SUM3}(x, y) = \sum_{j=1}^2 \text{MAX2}(x + x_j, y + y_j). \quad (38)$$

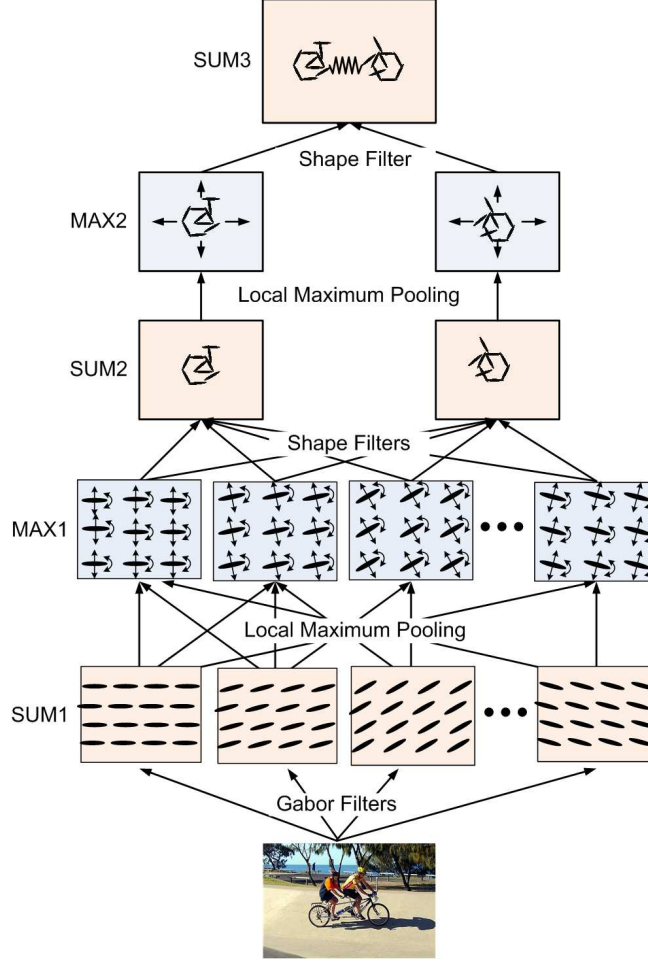


Figure 54: Sum-max maps. A SUM2 map is computed for each part-template. For each SUM2 map, a MAX2 map is computed by applying a local maximization operator to the SUM2 map. Then a SUM3 map is computed by summing over the two MAX2 maps. The SUM3 map scores the template matching, where the template consists of two part-templates that are allowed to locally shift their locations.

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

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

$$\text{MAX2}(\hat{x} + x_j, \hat{y} + y_j, j) = \text{SUM2}(\hat{x}_j, \hat{y}_j, j), \quad (39)$$

in the local maximization operation (37) in Up-2.

Down-1 Retrieve the perturbed elements of j -th part-template as described the inference algorithm of Subsection (2.4).

In our experiment, 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) .

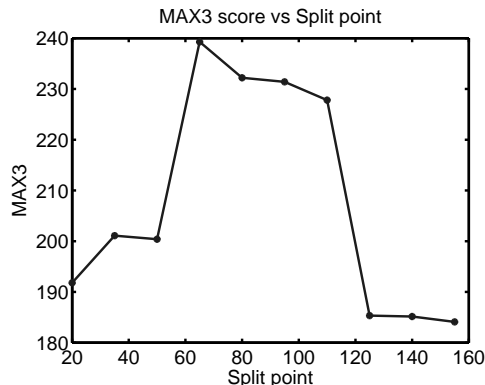


Figure 55: Experiment 10.1. MAX3 scores for different splitting points.

The MAX3 score in Step 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 (55) displays the MAX3 scores for 10 different splitting points. The result shown in Figure (53.b) is obtained at the splitting point the achieves the maximum MAX3 score.

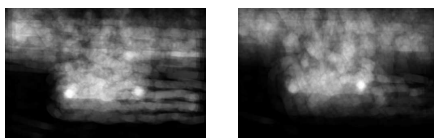


Figure 56: Experiment 10.1. The two SUM2 maps for the two part-templates at the optimal splitting point.

Figure (56) displays the two SUM2 maps for the two part-templates at the optimal splitting point.

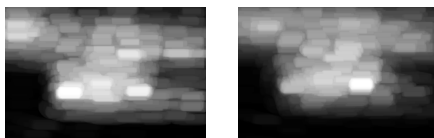


Figure 57: Experiment 10.1. The two MAX2 maps for the two part-templates at the optimal splitting point.

Figure (57) displays the two MAX2 maps for the two part-templates at the optimal splitting point.

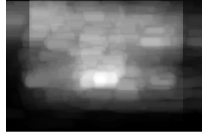


Figure 58: SUM3 map at the optimal splitting point.

Figure (58) displays the SUM3 map at the optimal splitting point.

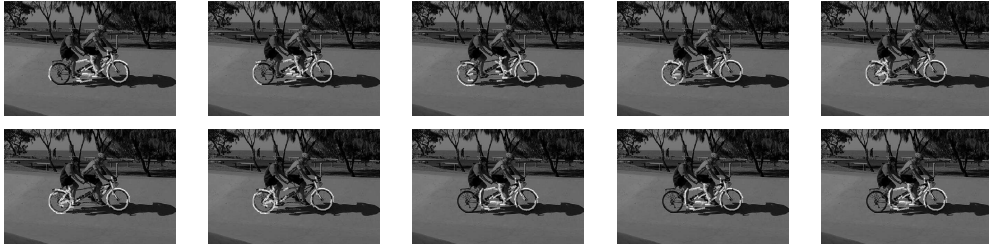


Figure 59: Experiment 10.1. Superposed sketches obtained by all the splitting points.

Figure (59) displays superposed sketches obtained at all the splitting points.

In this experiment, we set the response of a basis element to 0 if it is outside the boundary of the image. So the SUM2 maps are of the same size as the original image. If the center of a part-template is outside the image boundary, we set the responses of all its basis elements to 0. So the SUM3 map is also of the same size as the original image.

The recursive active basis can be considered a constellation model [18] whose constituent components are active basis. The MAX2 and SUM2 maps may have been commonly used in part-based models. Thanks to the work of Riesenhuber and Poggio [15], we are able to extend the SUM and MAX operations down to the image intensities.

8.2 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 (60.a) displays the image of horse that we used in Experiment 4, 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. However, we still 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 (60.b) displays the result of fitting the recursive active basis at the optimal splitting point. As a comparison, Figure (60.c) displays the result using the original template. The original template does not fit the rear part of the horse very well.

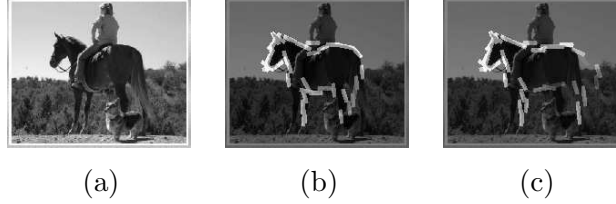


Figure 60: Experiment 10.2. (a) The observed image. (b) Superposed with sketch where the horse template is split horizontally into two part-templates. These two part-templates are allowed to move horizontally up to 10 pixels in each direction. (c) Superposed with sketch using the original horse template. In other words, the two part-templates are not allowed to move relative to each other.

Correlated activities. At this point, we would like to discuss the modeling of the perturbations or activities of the basis elements in the active basis model. For simplicity, we assume that the perturbations are independent and uniformly distributed within a small range. This leads to the simple inference algorithm based on the sum-max maps. The computational complexity is linear in the size of the image.

The recursive active basis can be considered a scheme to account for correlated activities, where those elements that belong to the same part-template share a common overall movement, in addition to their perturbations relative to the overall location of the part-template.

The method of Experiment 5 may be used to learn part-templates. We shall report our work on this issue elsewhere.

9 Discussion

We conclude this article with some comments on our method and a discussion of several related principles.

9.1 Simplicity

We argue that our method cannot be simpler for the vision tasks that we have studied in this article. The model is no much more complex than a wavelet expansion, except that we add local perturbations to the wavelet elements. The learning algorithm is no much more complex than edge detection, except that we perform it in a parallel fashion on multiple images. The inference algorithm only involves two consecutive operations on top of Gabor filtering. One is a local max filtering and the other is a local sum filtering.

For unsupervised learning, such as learning from non-aligned images and local learning of representative templates, we exploit the fact that our method can learn from single images, and the iterative learning procedures only involve an extra step of detection or classification in addition to the step of supervised learning. It remains to be seen how reliable it is to initialize from single

image learning.

The learning and inference algorithms are very easy to program. There are not many sensitive tuning parameters, although we did tune a couple of parameters such as the image resize factor and the number of elements. The data and codes for reproducing the experimental results in this paper are available from the reproducibility page. We only include some of the positive results in this paper. To avoid partial reporting, we did summarize our negative experiences in the experiments, although we did not document them. Because of the simplicity of the method, when things go wrong, it is often fairly easy to see where is the problem.

Our work is a simple logical consequence if we assume the validity of Olshausen-Field theory for V1 simple cells [14], and the Riesenhuber-Poggio theory for V1 complex cells [15]. We, of course, are not in a position to claim any neuroscientific validity of the SUM2 maps in our method. Our work is also a simple re-tooling of the texton model of Zhu, et al. [27].

In retrospect, we find that the following three principles provide useful insights.

9.2 Sparsity

Olshausen and Field [14] proposed 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 image. 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 sparsification. The reason is as follows. In Olshausen-Field representation, each image is encoded by a set of locations, orientations, and scales of wavelet elements. We can further encode these locations, orientations, and scales by an even smaller 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 template representation, we need to allow for small residuals in locations, orientations, and scales. Such small residuals become the perturbations in the active basis model, so that the templates are deformable.

9.3 Compositionality

S. Geman et al. [8] proposed 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 [29] investigated 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 sum-max maps is a variation of the cortex-like structure proposed by Riesenhuber and Poggio [15]. It is 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 L. Zhu et al. [26] on a recursive compositional scheme.

9.4 Invariance

Riesenhuber and Poggio [15] proposed 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 modified 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.

Acknowledgement

We thank Chuck Fleming for earlier collaboration on some of the experiments. We thank Alan Yuille, Stefano Soatto, Zhuowen Tu, and Leo Zhu for discussions. The work is supported by NSF-DMS 0707055, NSF-IIS 0713652, ONR N00014-05-01-0543, and Keck foundation. We acknowledge the use of data sets [24] provided by the Lotus Hill Institute, which is supported by a Chinese National 863 grant 2006AA01Z121 and an NSFC grant 60728203.

References

- [1] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 681-685, 2001.
- [2] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, 20, 273-297, 1995.
- [3] J. Daugman, "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, 1985.
- [4] S. Della Pietra, V. Della Pietra, and J. Lafferty, "Inducing features of random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 380-393, 1997.

- [5] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, C-22, 67-92, 1973.
- [6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, 55, 119-139, 1997.
- [7] J. H. Friedman, "Exploratory projection pursuit," *Journal of the American Statistical Association*, 82, 249-266, 1987.
- [8] S. Geman, D. F. Potter, and Z. Chi, "Composition system," *Quarterly of Applied Math*, 60, 707-736, 2002.
- [9] C. Guo, S. C. Zhu, and Y. N. Wu, "Towards a mathematical theory of primal sketch and sketchability," *International Conference on Computer Vision*, 2, 1228-1235, 2003.
- [10] C. Guo, S. C. Zhu, and Y. N. Wu, "Primal sketch: integrating structure and texture," *Computer Vision and Image Understanding*, 106, 5-19, 2007.
- [11] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, 1, 321-331, 1988.
- [12] S. Mallat and Z. Zhang, "Matching pursuit in a time-frequency dictionary," *IEEE Transactions on Signal Processing*, 41, 3397-415, 1993.
- [13] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," *Proceedings of Computer Vision and Pattern Recognition*, 2006.
- [14] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, 381, 607-609, 1996.
- [15] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, 2, 1019-1025, 1999.
- [16] Z. Tu, "Learning generative models via discriminative approaches," *Proceedings of IEEE Computer Vision and Pattern Recognition*, 2007.
- [17] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, 57, 137-154, 2004.
- [18] M. Weber, M. Welling and P. Perona, "Towards automatic discovery of object categories", *Proceedings of Computer Vision and Pattern Recognition*, 2000.
- [19] A. Witkin, "Scale-space filtering," *Proceedings of International Joint Conference on Artificial Intelligence*, 1983.
- [20] Y. N. Wu, C. Guo, S. C. Zhu, "From information scaling of natural images to regimes of statistical models," *Quarterly of Applied Math*, 66, 81-122, 2008.
- [21] Y. N. Wu, Z. Si, C. Flemming, and S. C. Zhu, "Active basis as deformable templates," *Proceedings of International Conference on Computer Vision*, 2007.
- [22] Y. N. Wu, S. C. Zhu, and C. Guo, "Statistical modeling of texture sketch," *Proceedings of European Conference of Computer Vision*, 2002.
- [23] Y. N. Wu, S.C. Zhu, and X. Liu, "Equivalence of Julesz ensemble and FRAME models," *International Journal of Computer Vision*, 38, 245-261, 2000.

- [24] Z. Yao, X. Yang, and S. C. Zhu, “Introduction to a large scale general purpose groundtruth database: methodology, annotation tools, and benchmarks,” *6th International Conference on EMMCVPR*, 2007.
- [25] A. L. Yuille, P. W. Hallinan, and D. S. Cohen, “Feature extraction from faces using deformable templates,” *International Journal of Computer Vision*, 8, 99-111, 1992.
- [26] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille, “Unsupervised structure learning: hierarchical recursive composition, suspicious coincidence and competitive exclusion,” *Proceedings of European Conference of Computer Vision*, 2008.
- [27] S. C. Zhu, C. E. Guo, Y. Z. Wang, and Z. J. Xu, “What are textons,” *International Journal of Computer Vision*, 62, 121-143, 2005.
- [28] S. C. Zhu and D. B. Mumford, “Prior learning and Gibbs reaction-diffusion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1236-1250, 1997.
- [29] S. C. Zhu and D. B. Mumford, “A stochastic grammar of images,” *Foundations and Trends in Computer Graphics and Vision*, 2, 259-362, 2006.
- [30] S. C. Zhu, Y. N. Wu, and D. B. Mumford, “Minimax entropy principle and its applications in texture modeling,” *Neural Computation*, 9, 1627-1660, 1997.