Primal Sketch: Integrating Structure and Texture ⋆

Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu

Departments of Statistics and Computer Science
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

This article proposes a generative image model, which we call "primal sketch," following Marr's insight and terminology. This model combines two prominent classes of generative models, namely, sparse coding model and Markov random field model, for representing geometric structures and stochastic textures respectively. Specifically, the image lattice is divided into structure domain and texture domain. The sparse coding model is used to represent image intensities on the structure domain, where edge and ridge segments are modeled by image coding functions with explicit geometric and photometric parameters. The edge and ridge segments form a sketch graph, which is governed by a simple spatial prior model. The Markov random field model is used to summarize image intensities on the texture domain, where the texture patterns are characterized by feature statistics in the form of marginal histograms of responses from a set of linear filters. The Markov random fields in-paint the texture domain while interpolating the structure domain seamlessly. We propose a sketch pursuit algorithm for model fitting. We show a number of experiments on real images to demonstrate the model and the algorithm.

Key words: Sparse coding, Markov random fields, Image primitives, Sketch graphs, Lossy image coding

⋆ We thank Arthur Pece for pointing out the connection with vector quantization. We also thank him and an anonymous referee for detailed comments and suggestions that have greatly improved the presentation of the paper. We thank Alan Yuille, Zhuowen Tu, Feng Han, and Yizhou Wang for insightful discussions. The work is supported by NSF IIS-0222967.

Email address: cguo,sczhu,ywu@stat.ucla.edu (Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu).

1 Introduction

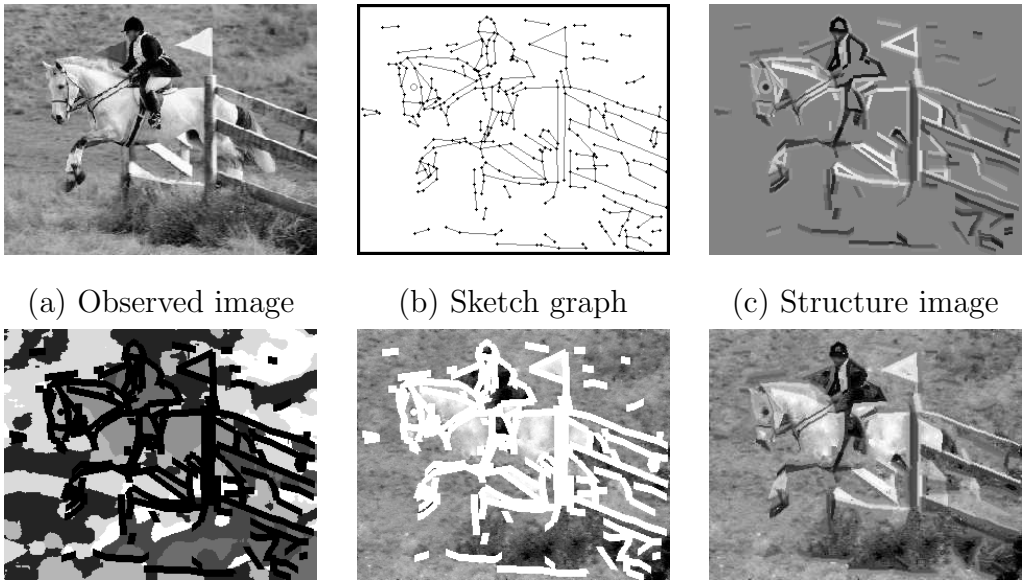
Geometric structures and stochastic textures are two ubiquitous classes of visual phenomena in natural scenes. Geometric structures appear simple, and can be represented by edges, ridges, and their compositions such as corners and junctions. Stochastic textures appear complex, and are often characterized by feature statistics. Despite their apparent distinctions, texture impressions are often caused by large number of object structures that are either too small or too distant relative to camera resolution. Moreover, as we change the viewing distance or camera resolution, the same group of objects may appear either as structures or textures. It is therefore desirable to integrate structures and textures in a common representational and computational framework.

In this article, we propose a generative model, which we call "primal sketch," following the insight and terminology of Marr [15]. The model combines two prominent classes of generative models. One is sparse coding model, for representing geometric structures. The other is Markov random field model, for characterizing stochastic textures.

Specifically, the image lattice is divided into structure domain and texture domain. The sparse coding model is used to represent image intensities on the structure domain, where the most common structures are boundaries of objects that are above a certain scale. Following Elder and Zucker [7], we model the image intensities of object boundaries by a small number of edge and ridge coding functions with explicit geometric and photometric parameters. For instance, an edge segment is modeled by an elongate step function convolved with a Gaussian kernel. A ridge segment is a composition of two parallel edge segments. These edge and ridge segments form a sketch graph, whose nodes are corners and junctions. The sketch graph is regulated by a simple spatial prior model. The form of our sparse coding model is similar to vector quantization by Gersho and Gray [9], where the coding functions serve as coding vectors.

The Markov random field model is used to summarize image intensities on the texture domain, where the texture patterns are characterized by feature statistics in the form of marginal histograms of responses from a set of linear filters. The Markov random fields in-paint the texture domain while interpolating the structure domain seamlessly.

Figure (1) shows an example. (a) is the observed image. (b) is the sketch graph, where each line segment represents an edge or ridge coding function. (c) is the reconstructed structure domain of the image using these edge and ridge coding functions. (d) is a segmentation of the remaining texture domain into a number of homogeneous texture regions, by clustering the local marginal histograms



(a) Observed image (b) Sketch graph (c) Structure image

(d) Texture regions (e) Synthesized textures (f) Synthesized image

Fig. 1. An example of the primal sketch model. (a) An observed image. (b) The sketch graph computed from the image. (c) The structure domain of the image. (d) The remaining texture domain is segmented into a number of homogeneous texture regions. (e) Synthesized textures on these regions. (f) The final synthesized image that integrates seamlessly the structure and texture parts.

of filter responses. Here different regions are represented by different shades. (e) displays the synthesized textures in the segmented regions. (f) is the final synthesized image by putting (c) and (e) together. Because the textures are synthesized with the structure domain as boundary conditions, the textures interpolate the structure domain seamlessly.

In our representation, the sparse coding model and the Markov random field model are intrinsically connected. The elongate and oriented linear filters, such as Gabor filters [6] or Difference of Gaussian filters [25], are used to detect edges or ridges at different frequencies. On the structure domain where edge and ridge segments are present, the filters have large responses along the edge or ridge directions, and the optimally tuned filters are highly connected and aligned across space and frequency. Such regularities and redundancies in filter responses can be accounted for by the image coding functions for edge and ridge segments. On the remaining texture domain where the filters fail to detect edges or ridges, the filter responses are weak and they are not well aligned over space or frequency. So we can pool the marginal histograms of filter responses to form statistical summaries of the image intensities. In that sense, texture statistics arise from recycling the responses of filters that fail to detect edges or ridges, and the sparse coding model and random field model represent two different schemes for grouping or combining filter responses.

The rest of the paper is organized as follows. Section 2 reviews sparse coding model and random field model to set the background, and then motivate our integrated modeling scheme. Section 3 presents the primal sketch representation. Section 4 describes the sketch pursuit algorithm. Section 5 shows a set of experiments on real images. Section 6 concludes with a discussion.

2 Background and motivation

Image modeling has been based on two alternative theories. The first theory originated from computational harmonic analysis. It represents an image deterministically by a linear superposition of wavelet functions. The second theory originated from statistical physics. It characterizes an image by a Markov random field, which describes the image in terms of a set of feature statistics.

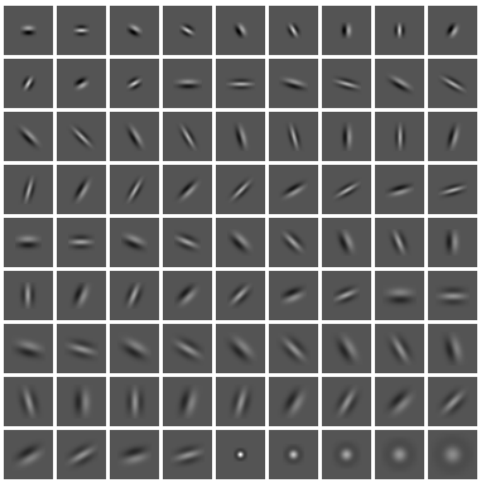


Fig. 2. A set of Difference of Gaussian filters, including elongate and oriented filters and isotropic filters.

Both theories involve a set of localized, oriented, elongated filters (as well as some isotropic filters) at different scales or frequencies. Two popular forms of the linear filters are Gabor functions [6] and Difference of Gaussian functions [25]. Such functions are localized in both spatial and frequency domains. In Figure (2), we plot a set of Difference of Gaussian functions. Gabor filters are similar in shape. Both forms of functions are biologically motivated by observations on the simple cells in primary visual cortex. While the Difference of Gaussian filters emphasize the spatial domain concept of gradient and Hessian, the Gabor filters emphasize the frequency domain concept of local Fourier contents.

2.1 Sparse coding model

Let $\mathbf{I}(x, y), (x, y) \in \Lambda$ be an image defined on an image lattice Λ. Let $\{B_i(x, y), i = 1, ..., N\}$ be a set of coding elements such as those depicted in Figure (2). We can represent image $\mathbf{I}$ by

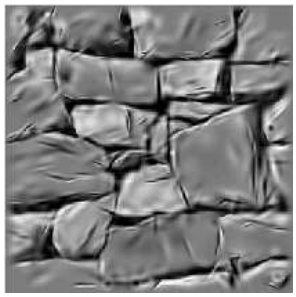
$$\mathbf{I}(x, y) = \sum_{i=1}^{N} c_i B_i(x, y) + \epsilon(x, y), \tag{1}$$

where c_i are the coefficients and ϵ is the residual image. This linear representation has been intensively studied in image coding and harmonic analysis. The interested reader is referred to the book of Mallat [13] and the references therein.

The dictionary of coding elements $\{B_i(x, y), i = 1, ..., N\}$ can be over-complete, in the sense that the number of coding functions N is greater than the number of pixels in image $\mathbf{I}$. The main principle in over-complete representation is sparsity. That is, the dictionary $\{B_i\}$ should be designed in such a way that for a typical natural image, we only need to choose a small number of elements to approximate it within a small error, i.e., only a small number of coefficients in $\{c_i\}$ of equation (1) need to be significantly different from 0. Using this principle, Olshausen and Field [18] learned an over-complete set of coding elements that resemble the Gabor functions. The notion of sparsity can also be expressed statistically by assuming that $\{c_i\}$ follow independent long-tail distributions. See the paper by Pece [19] for a thorough review of this topic.

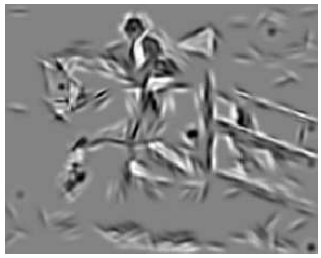
With a given over-complete dictionary of coding elements, the matching pursuit algorithm of Mallat and Zhang [14] can be employed to find a sparse representation of an image. The basic idea of this algorithm is to add one element at a time, and each time, we choose the element that results in the maximum reduction in the L_2-norm of the reconstruction error.

We did some experiments using linear sparse coding with a set of Gabor functions as well as isotropic Laplacian of Gaussian functions. Since these functions can be centered at every pixel, the dictionary of all the elements is highly over-complete. Figure (3) shows an example of sparse coding. (a) is an observed image of 128 $\times$ 128 pixels. (b) is the image reconstructed by 300 elements selected by the matching pursuit algorithm. Figure (4) shows a second example with a symbolic representation, where each selected elongate and oriented Gabor function is represented by a bar at the same location, with the same elongation and orientation. The isotropic elements are represented by circles.



(a) Observed 128 × 128 image (b) Reconstructed with 300 elements

Fig. 3. A sparse coding example: the representation is computed by matching pursuit.



(a) Observed image (b) Reconstructed (c) Symbolic sketch

Fig. 4. A sparse coding example computed by matching pursuit. (a) is the observed image. (b) is reconstructed with 300 elements. (c) is a symbolic representation where each selected elongate and oriented element is represented by a bar at the same location, with the same elongation and orientation. The isotropic elements are represented by circles.

The linear sparse coding model can capture the image structures such as object boundaries with a small number of elements. However, a small number of elements cannot represent textures very well, since textures are often very random and are of high complexity. If we force the linear additive model to represent textures, the representation will not be sparse.

Moreover, the linear additive model is still not adequate for representing structures. From the reconstructed image of Figure (4), we can see that a small number elements cannot capture the boundaries very well, and they do not line up into lines and curves. There are no concepts of corners and junctions either.

2.2 Markov random field model

We use $\{F_k, k = 1, ..., K\}$ to represent the set of Gabor or Difference of Gaussian filters, where k indexes the shape of the filter. The response of filter k at pixel (x, y) is denoted $[F_k * \mathbf{I}](x, y)$ or simply $F_k * \mathbf{I}(x, y)$. These operators

are generalized versions of the ubiquitous gradient operator $\nabla \mathbf{I}$ that is used in Canny edge detection [3] and variational/PDE approaches to image processing (see the recent book of Aubert and Kornprobst [1], and the references therein).

Zhu, Wu, and Mumford [26] proposed a Markov random field model [2] for textures that can be written in the form of the following Gibbs distribution:

$$p(\mathbf{I}) = \frac{1}{Z} \exp\{-\sum_{x,y} \sum_{k} \phi_k(F_k * \mathbf{I}(x,y))\}, \tag{2}$$

where $\phi_k()$ are one-dimensional functions, and Z is the normalizing constant depending on $\{\phi_k()\}$. $\phi_k()$ can be further parameterized as piecewise constant functions. Model (2) is an embellished version of the early ϕ-model of Geman and Graffigne [8], which can be used as a prior distribution for Bayesian image processing. The reader is refer to the book of Winkler [23] for a review of Markov random fields and their applications in image processing. The energy function in model (2) can also be viewed as a generalized form of the regularization terms commonly used in variational/PDE methods for controlling image smoothness [1].

The above model has an interesting connection to feature statistics. For each operator F_k, we collect all the responses over the whole image lattice Λ, so we have a sample $\{F_k * \mathbf{I}(x,y), \forall (x,y) \in \Lambda\}$. Then we summarize this sample by a histogram $h_k(\mathbf{I})$, where for each bin of this histogram, we calculate the proportion of the responses falling into this bin, regardless of the positions of these responses. Specifically, $h_k = \{h_{k,z}, \forall z\}$, and

$$h_{k,z} = \frac{1}{|\Lambda|} \sum_{(x,y) \in \Lambda} \delta(z; F_k * \mathbf{I}(x,y)), \tag{3}$$

where z indexes the histogram bins, and $\delta(z;x) = 1$ if x belongs to bin z, and $\delta(z;x) = 0$ otherwise.

We call these histograms the marginal histograms [10]. Let $h(\mathbf{I}) = (h_k(\mathbf{I}), k = 1, ..., K)$. Then let's consider the following image ensemble

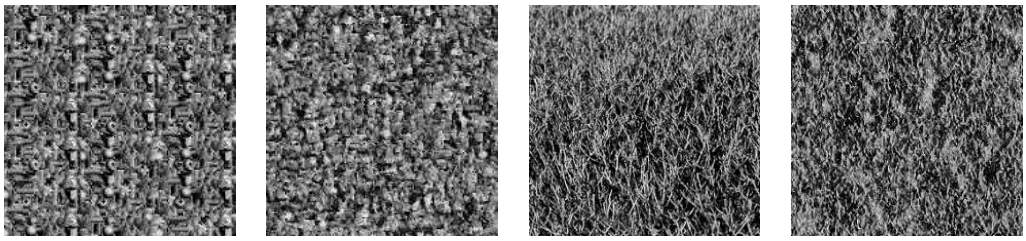
$$\Omega(h) = \{\mathbf{I} : h(\mathbf{I}) = h\} = \{\mathbf{I} : h_k(\mathbf{I}) = h_k, k = 1, ..., K\}, \tag{4}$$

that is, the set of images that produce the same histograms $h(h_1, ..., h_K)$. This ensemble is termed micro-canonical ensemble in statistical physics.

According to the fundamental result of "equivalence of ensembles" in statistical physics [5], the Markov random field model or the Gibbs distribution (2)

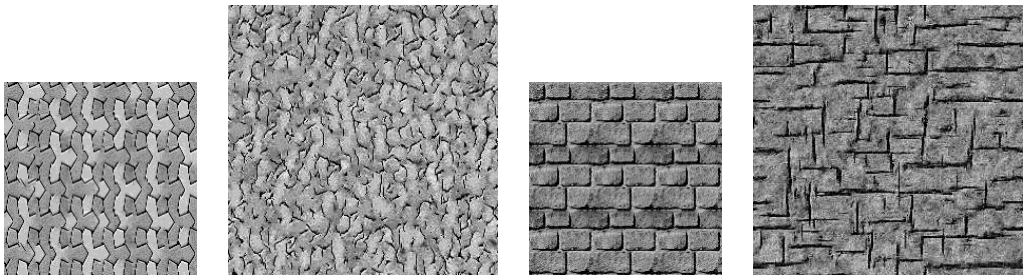
converges to the uniform distribution over the micro-canonical ensemble $\Omega(h)$ for some h, as the image lattice Λ goes to Z^2, i.e., the infinite lattice. Inversely, under the uniform distribution over the micro-canonical ensemble $\Omega(h)$, if we fix h and let $\Lambda \to Z^2$, then for any fixed local patch $\Lambda_0 \subset \Lambda$, the distribution of the image patch on Λ_0, i.e., $\mathbf{I}_{\Lambda_0}$, follows the Markov random field model (2) for some $\{\phi_k()\}$.

Therefore, for a relatively large image lattice, if we want to fit the Markov random field model (2) to an observed image, and synthesize images from the fitted model, we can directly estimate the marginal histograms of filter responses h from the observed image, and simulate images by randomly sampling from the image ensemble $\Omega(h)$. This can be accomplished using simulated annealing algorithm. See Wu, Zhu and Liu [24] for more details.



(a) (b) (c) (d)

Fig. 5. Random field and marginal histograms: (a) and (c) are observed images. (b) and (d) are "reconstructed" by matching marginal histograms of filter responses.



(a) (b) (c) (d)

Fig. 6. Random field and marginal histograms: (a) and (c) are observed images. (b) and (d) are "reconstructed" by matching marginal histograms of filter responses.

Experiments show that this model is quite effective in representing stochastic textures. See Figure (5) for two examples. However, as is evident in Figure (6), the random field models are ineffective in representing large image structures such as long edges.

2.3 Edge and ridge model

To understand the connection between the sparse coding model and the Markov random field model, we may consider the most common image structures: edges and ridges. Elder and Zucker [7] represent an edge segment as an elongate step function convolved with a Gaussian kernel. A ridge is a composition of two parallel edges. See Figure (7) for an illustration.

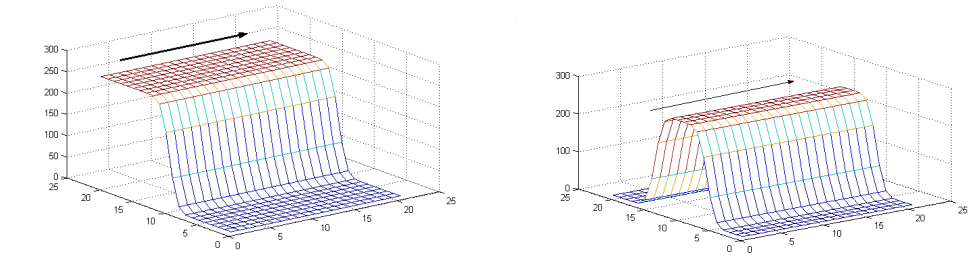


Fig. 7. An edge is modeled by a step function convolved with a Gaussian kernel. A ridge is modeled by double edges.

Consider, for example, a long step edge between two flat regions. Such an image structure evokes strong responses from Gabor filters (or Difference of Gaussian filters) across space and frequency (or scale). At each frequency or scale, the locations and orientations of the edge points can be detected by finding the optimally tuned elements in the same manner as Canny edge detection [3]. The orientations of these optimally tuned elements are highly aligned across space and frequency. That is, the linear filter responses from such an edge structure form a very regular pattern.

In our representation scheme, after detecting the locations and orientations of the edge points using linear filters, we model the intensities of the edge segment explicitly by fitting the model of Elder and Zucker [7]. This is the domain where sparse coding model applies.

One can also use the optimally tuned Gabor elements to code the edge structure, but one would need quite a number of Gabor elements. This is because a Gabor element is localized in both spatial and frequency domains, whereas a long step edge between two flat regions spans large ranges in both spatial and frequency domains. In order to capture the sparsity of the edge structure, we must further model the highly regular pattern of these Gabor elements and their coefficients. In this article, we choose not to do that, but to fit the parametric functions of edges and ridges directly.

On the part of the image where no edges or ridges are present, the filter responses are usually weak and are not well aligned in spatial or frequency domains. In that case, we can pool the marginal histograms of filter responses

to summarize the texture patterns. This is the domain where the Markov random field model applies.

3 Primal sketch model

According to the model, the image is generated as a mosaic as follows: the image lattice Λ is divided into a structure domain Λ_{str} and a textured domain Λ_{tex}. The image intensities on the structure domain are represented by a set of coding functions for edges and ridges. The image intensities on the texture domain are characterized by Markov random fields that interpolate the structure domain of the image.

3.1 Structure domain

The model for the structure domain of the image is

$$\mathbf{I}(x,y) = \sum_{i=1}^{n} B(x,y|\theta_i) + \epsilon(x,y), \quad (x,y) \in \Lambda_{\text{str}}, \quad i = 1, ..., n. \tag{5}$$

The coding functions $B(x,y|\theta_i)$ are used to represent edge and ridge segments (as well as blobs) in the image, where θ_i are geometric and photometric parameters of these coding functions. Let $\Lambda_{\text{str},i}$ be the set of pixels coded by $B(x,y \mid \theta_i)$. They do not overlap each other except over the small number of pixels where they join each other to form corners and junctions. Therefore, $B(x,y|\theta_i)$ is similar to coding vectors in vector quantization.

An edge segment is modeled by a 2D function that is constant along the edge, and has a profile across the edge. Specifically,

$$B(x,y \mid \theta) = f(-(x-u)\sin\alpha + (y-v)\cos\alpha), \tag{6}$$

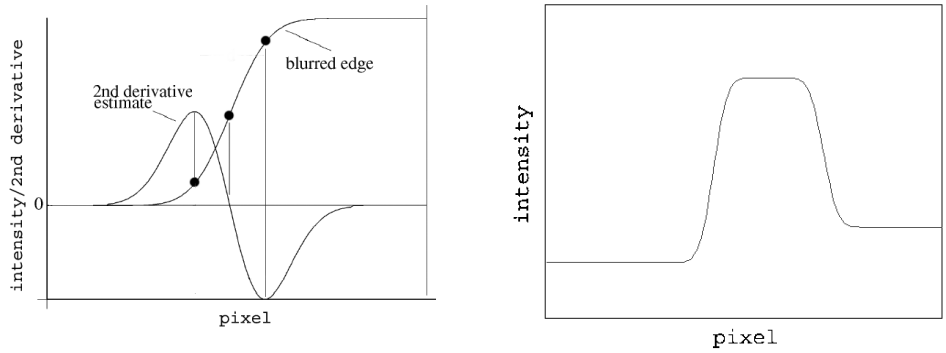
where

$$\begin{aligned} -l < (x-u)\cos\alpha + (y-v)\sin\alpha \le l, \\ -w \le -(x-u)\sin\alpha + (y-v)\cos\alpha \le w. \end{aligned}$$

That is, the function $B(x,y \mid \theta)$ is supported on a rectangle centered at (u,v), with length $2l+1$, width $2w+1$, and orientation α.

For the profile function $f()$, let $f_0(x) = -1/2$ for $x < 0$ and $f_0(x) = 1/2$ for $x \geq 0$, and let $g_s()$ be a Gaussian function of standard deviation s. Then $f() = a + bf_0() * g_s()$. This is the model proposed by Elder and Zucker [7]. The convolution with Gaussian kernel is used to model the blurred transition of intensity values across the edge, caused by the three dimensional shape of the underlying physical structure, as well as the resolution and focus of the camera. As proposed by Elder and Zucker [7], the parameter s can be determined by the distance between the two extrema of the second derivative $f''()$. See Figure (8.a) for an illustration.

Thus in the coding function $B(x, y \mid \theta)$ for an edge segment, $\theta = (t, u, v, \alpha, l, w, s, a, b)$, namely, type (which is edge in this case), center, orientation, length, width, sharpness, average intensity, intensity jump. θ captures geometric and photometric aspects of an edge explicitly, and the coding function is non-linear in θ.



(a) Edge profile (b) Ridge profile

Fig. 8. (a) An edge profile and its second derivative. The blurring scale is determined by the distance between the extrema of the second derivative. (b) A ridge profile is a composition of two edge profiles.

A ridge segment has the same functional form as (6), where the profile $f()$ is a composition of two edge profiles. See Figure (8.b) for an illustration, where there are three flat regions. The profile of a multi-ridge is a composition of two or more ridge profiles. A blob function is modeled by rotating an edge profile, more specifically, $B(x, y \mid \theta) = f(\sqrt{(x-u)^2 + (y-v)^2} - r)$, where $(x-u)^2 + (y-v)^2 \leq R^2$, and again $f() = a + bf_0() * g_s()$ being a step edge convolved with a Gaussian kernel. This function is supported on a disk area centered at (u, v) with radius R. The transition of intensity occur at the circle of radius $r < R$.

The corners and junctions are important structures in images. They are modeled as compositions of edge or ridge functions. When a number of such coding functions join to form a corner or a junction, the image intensities of the small

number of overlapping pixels are modeled as averages of these coding functions. The end point of a ridge is modeled by a half blob.

See Figure (9) for a sample of local structure elements, which are the coding functions and their combinations. There are eight types of elements: blobs, end points, edges, ridges, multi-ridges, corners, junctions and crosses. Figure (9.a) shows the symbolic representations of these elements. Figure (9.b) displays the image patches of these elements.

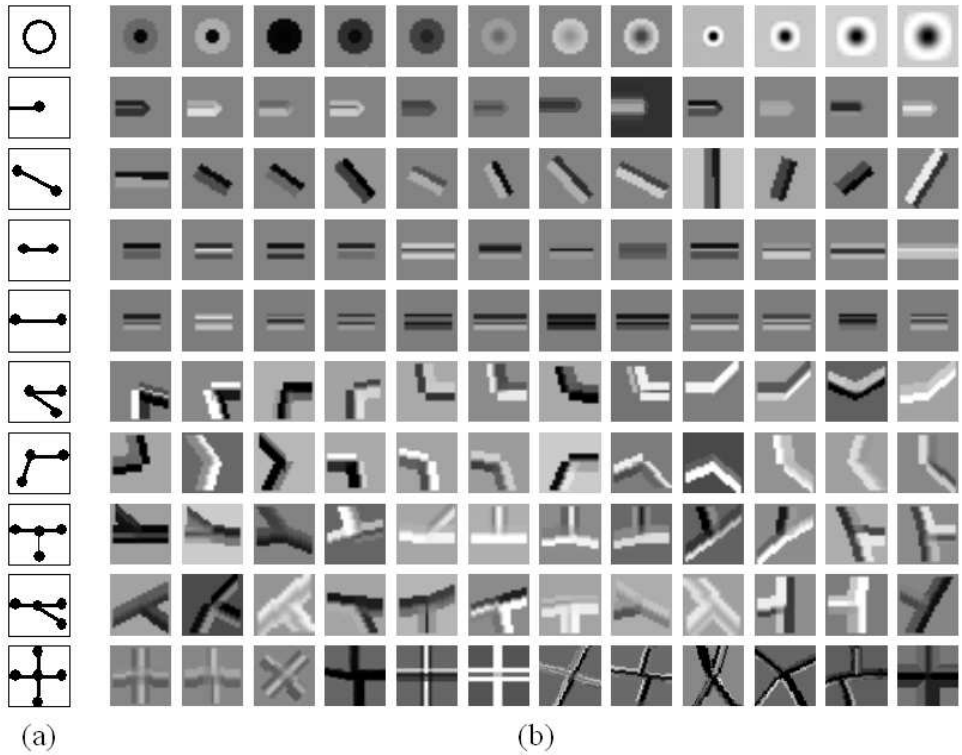


Fig. 9. A collection of local structure elements produced by our model. There are eight types of elements: blobs, end points, edges, ridges, multi-ridges, corners, junctions and crosses. (a) The symbolic representation. (b) The photometric representation.

Let $S_{\text{str}} = (\theta_i, i = 1, ..., n)$ be the sketch graph formed by these coding functions. The graph has a set of nodes or vertices $V = \cup_{d=0}^{4} V_d$, where V_d is the set of nodes with degree d, i.e., the nodes with d arms. For instance, a blob node has degree 0, an end point has degree 1, a corner has degree 2, a T-junction has degree 3, and a cross has degree 4. We do not allow nodes with more than 4 arms. S_{str} is regularized by a simple spatial prior model :

$$p(S_{\text{str}}) \propto \exp\{-\sum_{d=0}^{4} \lambda_d |V_d|\}, \tag{7}$$

where $|V_d|$ is the number of nodes with d arms. The prior probability or the

energy term $\gamma_{\rm str}(S_{\rm str}) = \sum_{d=0}^{4} \lambda_d |V_d|$ penalizes free end points by setting $\lambda_{\rm str}$ at a large value. We shall give concrete parameter values in Section (4).

3.2 Texture domain

The texture domain $\Lambda_{\rm tex}$ is segmented into m regions of homogenous texture patterns, $\Lambda_{\rm tex} = \cup_{j=1}^{m} \Lambda_{{\rm tex},j}$. Within each region j, we pool the marginal histograms of the responses from the K filters, $h_j = (h_{j,k}, k = 1, ..., K)$, where

$$h_{j,k,z} = \frac{1}{|\Lambda_{{\rm tex},j}|} \sum_{(x,y) \in \Lambda_{{\rm tex},j}} \delta(z; F_k * \mathbf{I}(x,y)), \tag{8}$$

where z indexes the histogram bins, and $\delta(z; x) = 1$ if x belongs to bin z, and $\delta(z; x) = 0$ otherwise.

According to the previous section, this is equivalent to a Markov random field model for each texture region:

$$p(\mathbf{I}_{\Lambda_{{\rm tex},j}}) \propto \exp\{- \sum_{(x,y) \in \Lambda_{{\rm tex},j}} \sum_{k=1}^{K} \phi_{j,k}(F_k * \mathbf{I}(x,y))\}. \tag{9}$$

These Markov random fields have the structure domain as boundary conditions, because when we apply filters F_k on the pixels in $\Lambda_{\rm tex}$, these filters may also cover some pixels in $\Lambda_{\rm str}$. These Markov random fields in-paint the texture domain $\Lambda_{\rm tex}$ while interpolating the structure domain $\Lambda_{\rm str}$, and the in-painting is guided by the marginal histograms of linear filters within each region. This point of view is closely related to the in-painting work of Chan and Shen [4].

Let $S_{\rm tex} = (\Lambda_{{\rm tex},j}, j = 1, ..., m)$ denotes the segmentation of the texture domain. $S_{\rm tex}$ follows a prior model $p(S_{\rm tex}) \propto \exp\{-\gamma_{\rm tex}(S_{\rm tex})\}$, for instance, $\gamma_{\rm tex}(S_{\rm tex}) = \rho m$ to penalize the number of regions. One may use more sophisticated models for segmentation (see, e.g., Tu, Chen, Yuille and Zhu [22]).

3.3 Integrated model

Formally, we can integrate the structure model (5) and the texture model (9) into a probability distribution. Our inspiration for such an integration comes from the model of Mumford and Shah [17]. In their method, the prior model for the noiseless image can be written as

$$p(\mathbf{I}, S) = \frac{1}{Z} \exp\{- \sum_{(x,y)\in\Lambda/S} \lambda |\nabla \mathbf{I}(x,y)|^2 - \gamma |S|\}, \tag{10}$$

where S is a set of pixels of discontinuity that correspond to the boundaries of objects, and $|S|$ is the number of pixels in S. In model (10), S is the structure domain of the image, and the remaining part is the texture domain.

Our model can be viewed as an extension of the Mumford-Shah model. Let $S = (S_{\text{str}}, S_{\text{tex}})$, we have

$$\begin{aligned} p(\mathbf{I}, S) = \frac{1}{Z} \exp\{ &- \sum_{i=1}^{n} \sum_{(x,y)\in\Lambda_{\text{str},i}} \frac{1}{2\sigma^2} (\mathbf{I}(x,y) - B_i(x,y \mid \theta_i))^2 - \gamma_{\text{str}}(S_{\text{str}}) \\ &- \sum_{j=1}^{m} \sum_{(x,y)\in\Lambda_{\text{tex},j}} \sum_{k=1}^{K} \phi_{j,k}(F_k * \mathbf{I}(x,y)) - \gamma_{\text{tex}}(S_{\text{tex}})\}. \end{aligned} \tag{11}$$

Compared to Mumford-Shah model, model (11) is more sophisticated in both structure part and texture part.

4 Sketch pursuit algorithm

This section details the sketch pursuit algorithm for computing the sketch graph and clustering the textures. We first present the objective function that is used in the later phases of the algorithm after the initialization phase.

4.1 The objective function

The logarithm of $p(\mathbf{I}, S)$ defined in (11) should be the objective function to maximize for inferring S. But computationally it is too demanding. Therefore, we construct an approximated objective function by assuming that the texture part can be approximated by Gaussian model with slowly varying means.

To derive the approximated objective function, let's first consider the following simple scenario first. Suppose we want to test whether a coding function $B(x, y \mid \theta)$ is present in image $\mathbf{I}$, we can consider the following hypothesis testing problem:

$$\begin{aligned} &H_0 : \mathbf{I}(x,y) = \mu + N(0, \sigma^2) \\ &H_1 : \mathbf{I}(x,y) = B(x, y \mid \theta) + N(0, \sigma^2), \ (x,y) \in \Lambda_1, \end{aligned}$$

where Λ_1 is the set of pixels covered by $B(x, y \mid \theta)$. μ in the null hypothesis H_0 is estimated as the average of the pixel values in Λ_1.

Define

$$\Delta L(B) = \sum_{(x,y)\in\Lambda_1} \left[(\mathbf{I}(x,y) - \mu)^2 - (\mathbf{I}(x,y) - B(x,y \mid \theta))^2 \right]. \tag{12}$$

Then the log likelihood ratio score for the above hypothesis testing is $\Delta L(B)/2\sigma^2$.

In order to infer the sketch graph $S_{\text{str}} = \{B_i, i = 1, ..., n\}$ from the image $\mathbf{I}$, we choose to maximize the following objective function

$$L(S_{\text{str}}) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} \Delta L(B_i) - \gamma_{\text{str}}(S_{\text{str}}) = \frac{1}{2\sigma^2} \left[\sum_{i=1}^{n} \Delta L(B_i) - \tilde{\gamma}_{\text{str}}(S_{\text{str}}) \right] \tag{13}$$

where $\tilde{\gamma}_{\text{str}}(S_{\text{str}}) = 2\sigma^2 \gamma_{\text{str}}(S_{\text{str}})$. In practice, we only need to specify $\tilde{\gamma}_{\text{str}}(S_{\text{str}})$ a priori, without explicitly specifying σ^2, for maximizing $L(S_{\text{str}})$.

The objective function (13) can be justified as the regularized log-likelihood of the following working model:

$$\begin{aligned} &\mathbf{I}(x,y) = B_i(x, y \mid \theta_i) + N(0, \sigma^2),\ (x,y) \in \Lambda_{\text{str},i},\ i = 1, ..., n, \\ &\mathbf{I}(x,y) = \mu(x,y) + N(0, \sigma^2),\ (x,y) \in \Lambda_{\text{tex}}. \end{aligned} \tag{14}$$

This working model is a simplified version of the integrated model presented in the previous section, where the texture part Λ_{tex} is modeled by Gaussian distribution with slowly varying mean $\mu(x, y)$. Specifically, let l_0 be the log-likelihood of the following background model:

$$\mathbf{I}(x,y) = \mu(x,y) + N(0, \sigma^2), (x,y) \in \Lambda,$$

then the log-likelihood of the working model (14) is $l_0 + \sum_{i=1}^{n} \Delta L(B_i)$, assuming that for $(x, y) \in \Lambda_{\text{str},i}$, i.e., pixels covered by B_i, $\mu(x, y)$ in the background model are all equal to the average of the pixel values in $\Lambda_{\text{str},i}$. Therefore $L(S_{\text{str}})$ in (13) is the regularized log-likelihood of the working model (14). Similar objective functions in the form of likelihood ratios have been previous constructed by Pece [20] and references therein.

The reason we choose to use the working model (14) is mainly due to its computational simplicity. It also fits our logic that texture statistics arise by pooling the filter responses where we fail to detect and fit edge or ridge functions. We currently have no proof that maximizing (13) will also maximizes the log of (11). We shall explore this issue in further work.

The sketch pursuit algorithm is a greedy algorithm for achieving a local maximum of the object function (13). It consists of the following phases. *Phase 0*: an edge and ridge detector based on linear filters is run to give an initialization for the sketch graph. *Phase 1*: a greedy algorithm is used to determine the sketch graph but without using the spatial prior model. *Phase 2*: a greedy algorithm based on a set of graph operators is used to edit the sketch graph to achieve good spatial organization as required by the spatial prior model. *Phase 3*: the remaining portion of the image is segmented into homogeneous texture regions by clustering the local marginal histograms of filter responses. The inference algorithm yields two outputs. 1) a sketch graph for the image, with edge and ridge segments, as well as corners and junctions. 2) a parameterized representation of the image which allows the image to be re-synthesized and to be encoded efficiently.

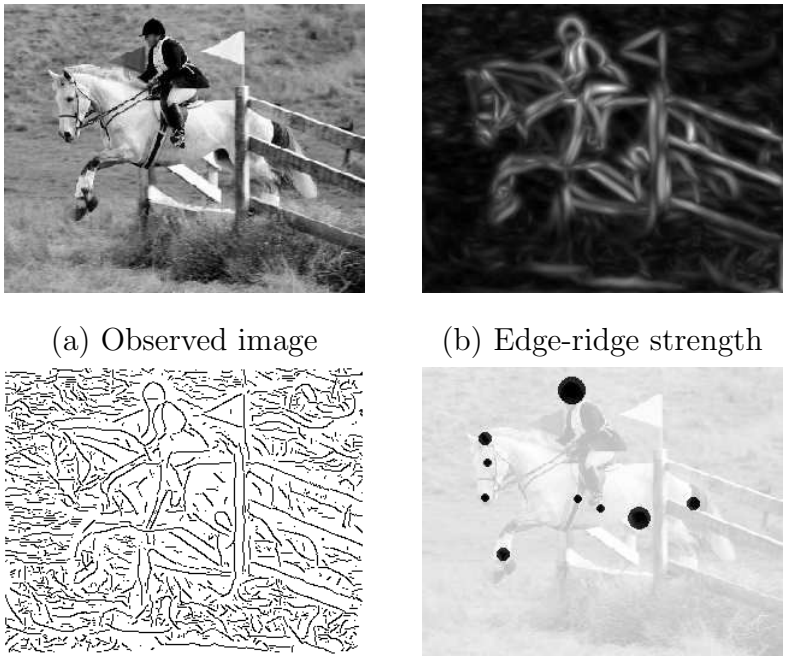
4.2 Phase 0: detect edges, ridges, and blobs

The detection method we use is essentially an extension of Canny edge detector [3]. At each pixel, we compute the sum of the squares of $\mathrm{D}^1\mathrm{G}$ and $\mathrm{D}^2\mathrm{G}$ responses for each scale and orientation, where $\mathrm{D}^1\mathrm{G}$ and $\mathrm{D}^2\mathrm{G}$ denote the Difference of Gaussian filters for computing the first derivatives and second derivatives respectively. One can also replace $\mathrm{D}^1\mathrm{G}$ and $\mathrm{D}^2\mathrm{G}$ filters by Gabor sine and Gabor cosine filters respectively. We normalize the filters to have mean 0 and L_1 norm 1. In our implementation, we use 3-5 scales and 18 orientations. The maximum of the combined responses over these scales and orientations is considered the edge-ridge strength at that pixel, and the orientation of the corresponding filters is considered the orientation of this pixel. See Figure (10) for an example. (a) is the observed image. (b) is the map of the edge-ridge strengths. Using the non-maxima suppression method in Canny edge detector, a pixel is considered a detected edge-ridge point if its edge-ridge strength is above a threshold, and achieves local maximum among the pixels on the norm direction of the edge-ridge orientation of this pixel. Figure (10.c) is a binary map that displays the detected edge-ridge points.

The blob strength at a pixel is measured by the maximum response of the isotropic filters over a number of scales. We use thresholding and non-maxima suppression to get the detected blobs, as shown in Figure (10.d).

4.3 Phase 1: sequentially add coding functions

Phase 1 follows Phase 0. We assume that the image has already been normalized to have mean 0 and variance 1. From the edge-ridge map, we find the edge-ridge point of the maximum strength over the whole image. From



(a) Observed image (b) Edge-ridge strength

(c) Detected edge-ridge points (d) Detected blobs

Fig. 10. Sketch pursuit phase 0: detecting edges, ridges, and blobs using derivative filters. For the observed image (a), a set of derivative filters are applied to get the edge-ridge strength (b). The detected edge-ridge points (c) are computed by non-maxima suppression. (d) shows the detected blobs.

this point, we connect the edge-ridge points along the orientation of this current point to form a line segment. We do this until we cannot extend the line segment any further. This procedure is similar to the maximal alignment of Moisan, Desolneux, and Morel [16].

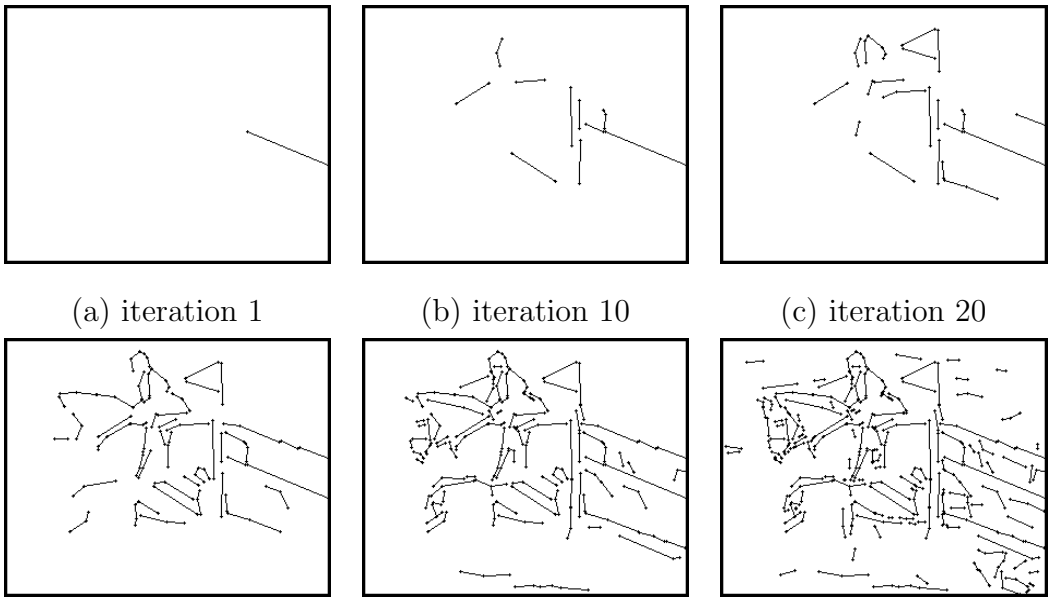
After that, we compute the profile of this line segment by averaging the image intensities on the cross-sections of the line segment. Then we decide the extrema of the second derivative of the profile (see Figure 8 for the illustration of the extrema of the second derivative). The width of the profile is determined so that there are two pixels beyond the extrema on the two sides of the profile. We start from the profile of 7 pixel width, and then increase the width until the above criterion is met. By doing so, we only model the intensity transition across the edge or ridge, while leaving the flat regions on the two sides to be captured by texture models. Then we fit an edge or ridge function B_1, and compute $\Delta L(B_1)$. If $\Delta L(B_1) > \epsilon$, we then accept B_1. We choose $\epsilon = 5$ (for image normalized to mean 0 and variance 1) in our implementation.

We continue the procedure as follows. From each end of the accepted coding function B_1, we search the connected and aligned points in the edge-ridge map to form a line segment. We then fit the coding function B_2, and compute $\Delta L(B_2)$ to decide whether to accept it or not, by comparing it to ϵ. By re-

peating the above procedure, we obtain a chain of connected coding functions to form a curve.

If no more coding functions can be added to the ends of this curve, we then choose the edge-ridge point with the maximum strength among all the edge-ridge points that are not covered by existing coding functions, and start a new curve. We repeat this process until no more curve can be started.

Figure (11) shows the results of sketch pursuit phase 1 on the horse-riding image. The sketch graphs are obtained after 1, 10, 20, 50, 100, and 180 steps.



(a) iteration 1 (b) iteration 10 (c) iteration 20

(d) iteration 50 (e) iteration 100 (f) iteration 180

Fig. 11. Sketch pursuit phase 1: sequentially add coding functions. We show the sketch graph after iterations 1, 10, 20, 50, 100, and 180.

4.4 Phase 2: fix corners and junctions

After getting the initial sketch graph in phase 1, we use a collection of graph operators to edit the sketch graph to fix the corners and junctions. This phase is accomplished by a greedy algorithm that seeks to find a local mode of the objective function

$$L(S_{\text{str}}) = \sum_{i=1}^{n} \Delta L(B_i) - \sum_{d=0}^{4} \lambda_d |V_d|, \tag{15}$$

where $|V_d|$ is the number of nodes with d arms in the sketch graph S_{str}. In our experiments, we choose $\lambda_0 = 1$, $\lambda_1 = 5$, $\lambda_2 = 2$, $\lambda_3 = 3$,$\lambda_4 = 4$.

The reversible graph operators we use to edit the sketch graph are summarized and explained in Figure (12).

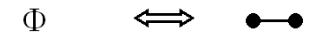
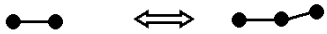
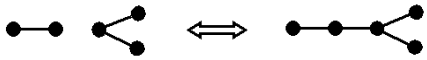
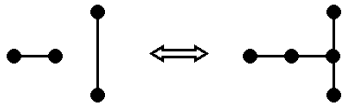
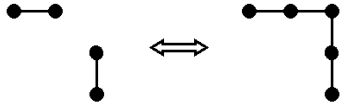
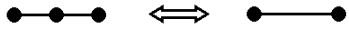
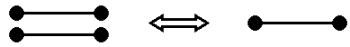
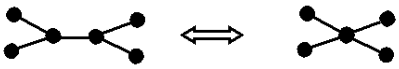
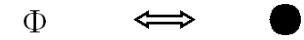
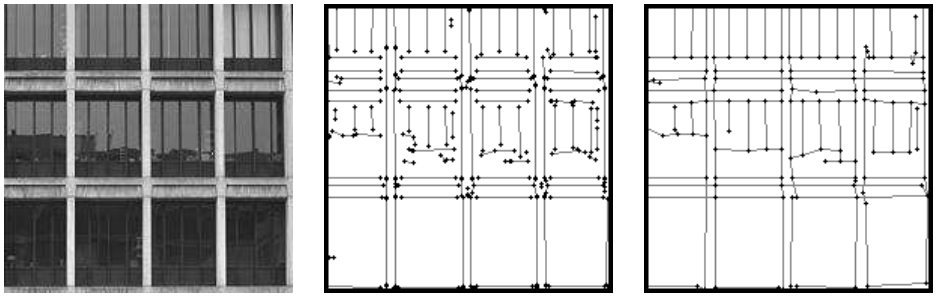
operators	graph change	illustration
G_1, G_1'	create/delete	
G_2, G_2'	grow/shrink	
G_3, G_3'	connect/disconnect	
G_4, G_4'	extend to touch/remove to disconnect	
G_5, G_5'	extend to join/remove to disconnect	
G_6, G_6'	combine/break	
G_7, G_7'	combine/split	
G_8, G_8'	merge/split	
G_9, G_9'	create/remove a blob	
G_{10}, G_{10}'	switch between stroke(s) and a blob	

Fig. 12. The ten pairs of reversible graph operators used in the sketch pursuit process.

Figure (13) shows an example of the sketch pursuit process. (a) is an input image. (b) is the sketch graph after phase 1. (c) is the sketch graph after phase 2. Comparing the sketch graphs in (b) and (c), we can see that some of the corners and junctions are missed in phase 1, but they are recovered in phase 2.

An example of applying graph operators is show in Figure (14). The ten pairs of reversible graph operators change the topological property of the sketch graph.

The computational strategy in this phase is as follows. As illustrated in Figure (14), from a current sketch graph, we randomly choose a local subgraph. All of the ten pairs of graph operators are attempted for editing that local subgraph. We examine all the new subgraphs that can be produced after 3-5



(a) (b) (c)

Fig. 13. Sketch pursuit. (a) Observed image. (b) Sketch graph after phase 1. (c) Sketch graph after phase 2.

steps of graph operations (usually around 5 to 20 new subgraphs). For each new subgraph, we compute the increase in $L(S_{\text{str}})$ in (15) if we change the current subgraph to this new subgraph. Currently we adopt a greedy method, where we choose the new subgraph that gives us the maximum increase in $L(S_{\text{str}})$. This process is repeated until no modification can be accepted.

4.5 Phase 3: texture clustering and synthesis

After Phase 2 is finished, we have the lattice Λ_{tex} for textures. We use k-mean clustering method to divide Λ_{tex} into homogeneous texture regions. We first compute the histograms of the derivative filters within a local window (e.g. 7×7 pixels). For example, if we use 7 filters and if 7 bins are used for each histogram, then totally we have a 49-dimensional feature vector for each pixel. We then cluster these feature vectors into different regions. In our current implementation, we use a small number (5-7) of small derivative filters for characterizing textures. After clustering Λ_{tex} into texture regions, we then synthesize textures in these regions by matching the corresponding histograms of filter responses.

5 Experiments

5.1 Primal sketch representation of real images

Figures (15), (16), and (17) show more experiments of the sketch pursuit process. It appears that the primal sketch representation captures both the structural and textural aspects of the image.

There are a number of limitations in our model. First, the relationship between

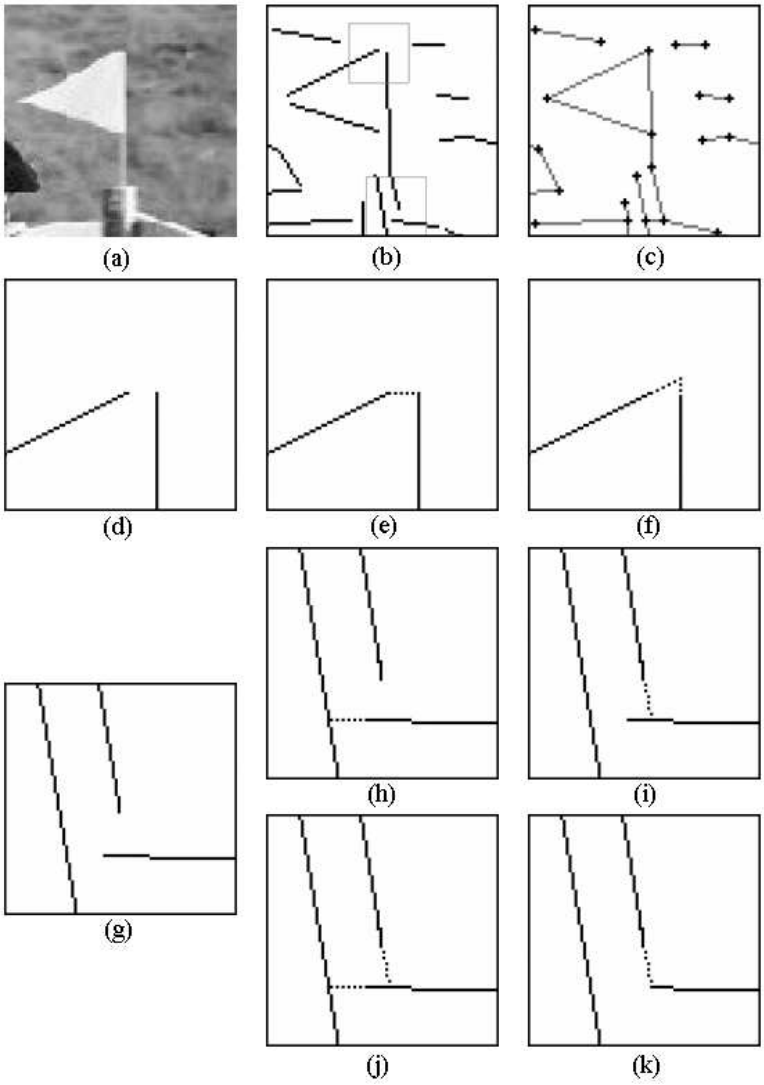
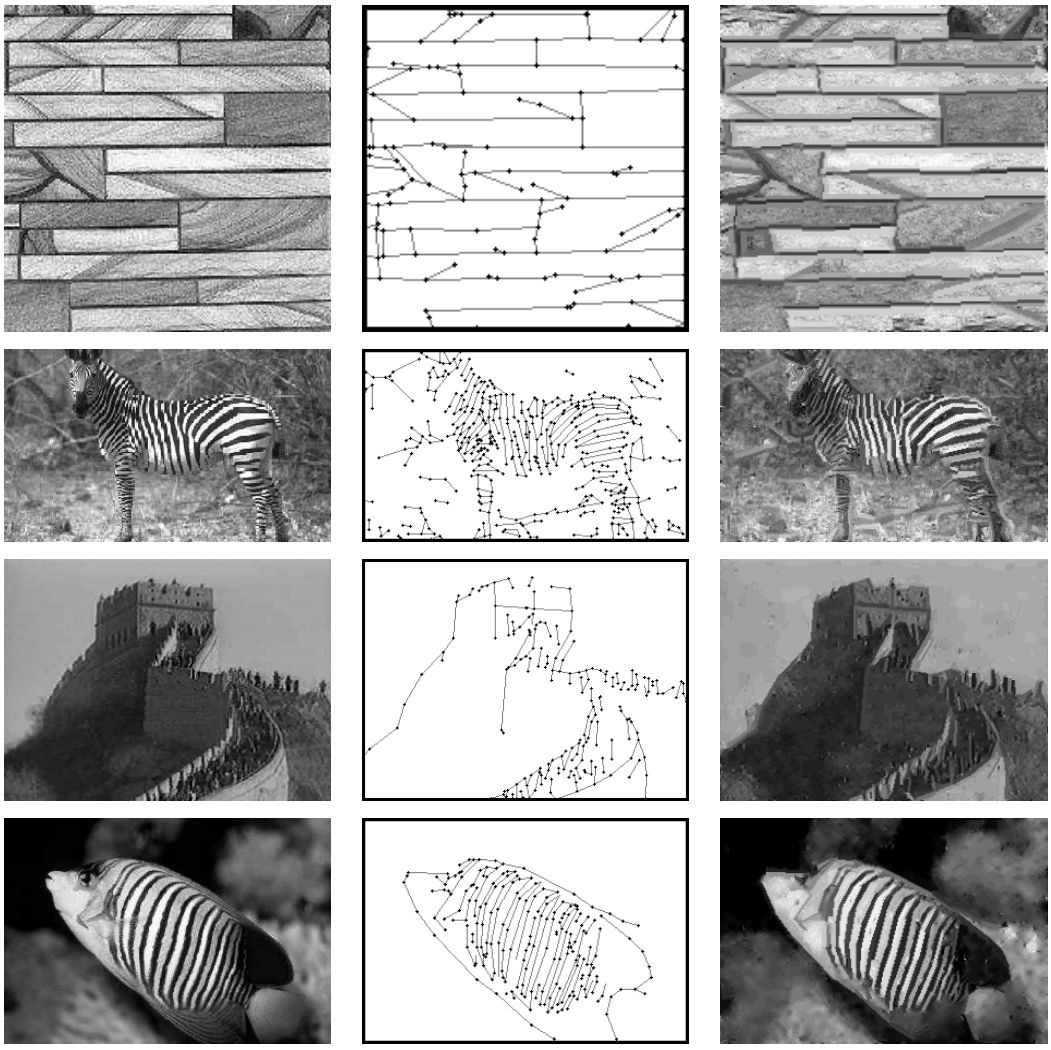


Fig. 14. An example of applying graph operators. (a) A local image patch from the horse-riding image. (b) Sketch graph after sketch pursuit phase 1. (c) Sketch graph after sketch pursuit phase 2. (d) Zoom-in view of the upper rectangle in (b). (e) Applying graph operator G_3 – connecting two vertices in (d). (f) Applying graph operator G_5 – extending two strokes to join each other. (g) Zoom-in view of the lower rectangle in (b). (h) Applying graph operator G_4 – extending one stroke to touch another stroke. (i) Applying graph operator G_4 to another stroke. (j) Combining (h) and (i). (k) Applying graph operator G_4 – extending one stroke, and applying G_1' – removing one stroke.

the sketch graph and the segmentation of the texture part is not modeled. Second, the model does not include structures caused by lighting and shading. Third, the model only assumes piecewise homogeneous textures. It cannot account for textures on slanted surfaces.



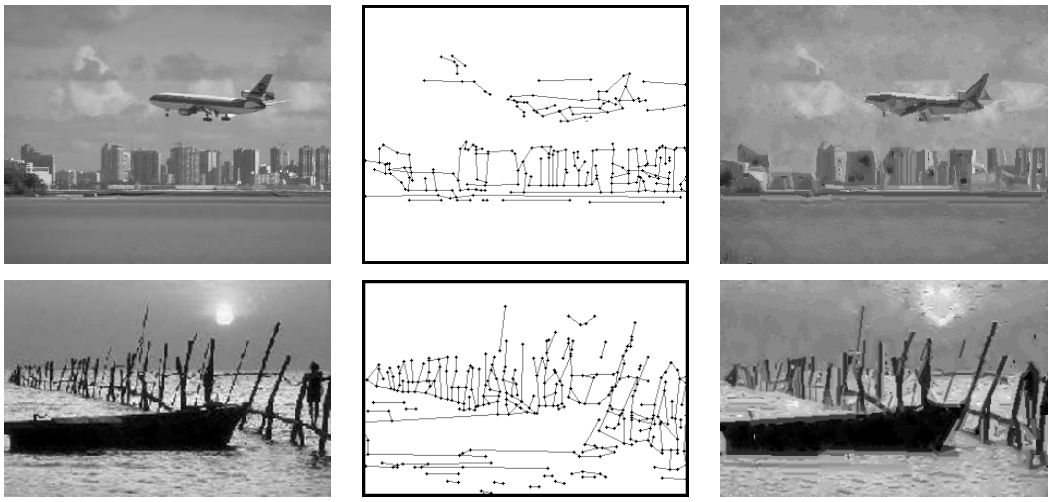
(a) (b) (c)

Fig. 15. More results of the primal sketch model. (a) Observed image. (b) Sketch graph. (c) Reconstructed image from the fitted primal sketch model.

5.2 Lossy image coding

The primal sketch model provides a lossy image coding scheme, in the sense that the texture regions are coded by histograms of filter responses, and the exact pixel values in texture regions are not coded. Table (1) counts the number of parameters for describing the primal sketch model in the experiment shown in Figure (1).

The observed image has 300×240 pixels, of which $18,185$ pixels (around 25%) are considered by our model as belonging to structure domain. The sketch graph has 152 vertices and 275 coding functions that are coded by $1,421$ bytes. The texture pixels are summarized (instead of being exactly coded)



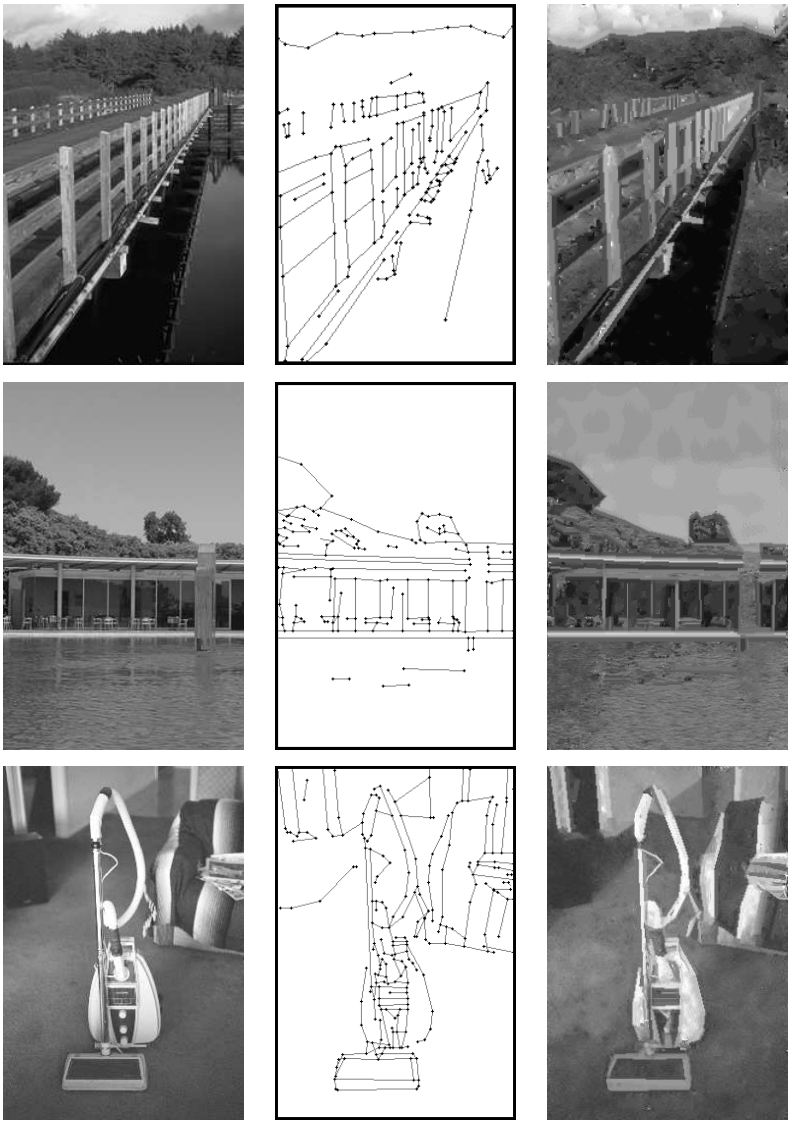
(a) (b) (c)

Fig. 16. More results of the primal sketch model. (a) Observed image. (b) Sketch graph. (c) Reconstructed image from the primal sketch model.

	coding description	coding length (bits)
Sketch graph		
	vertices 152	152×2×9=2,736
	strokes (275)	275×2×4.7=2,585
Sketch image	profiles (275)	275×(2.4×8+1.4×2)=6,050
Total for structure domain	18,185 pixels	11,371
Region boundaries	3659 pixels	3659×3 = 10,977
Texture regions and histograms		7×5×13×4.5 = 2,048
Total for texture domain	41,815 pixels	13,025
Total for the whole image	72,000 pixels	3,049 bytes, 0.04 byte/pixel

Table 1
The coding length of the primal sketch model in the experiment shown in Figure 1.

by 455 parameters, with 5 filters for 7 texture regions and each pools a 1D histogram of filter responses into 13 bins. Together with the codes for the region boundaries, the total coding length for the textures is $1,628$ bytes. The total coding length for the synthesized image in Figure (1.f) is $3,049$ bytes or 0.04 byte per pixel. For lossless (error $= 0$) JPEG 2000 coding, the compression rate is 0.685 byte per pixel. Of course, we are not making a direct comparison here, since our method is lossy coding. For this type of lossy coding, it is hard to quantify the perception error. We shall leave this issue to future study.



(a) (b) (c)

Fig. 17. More results of the primal sketch model. (a) Observed image. (b) Sketch graph. (c) Reconstructed image from the primal sketch model.

6 Discussion

The modeling of textures and structures has long been studied in vision. Julesz [11] first proposed a texture theory and conjectured that a texture is a set of images sharing some common statistics on some features related to human perception. Later he switched to a texton theory [12] and identified bars, edges, corners, terminators as textons. Marr [15] summarized Julesz's theories and other experimental results and proposed primal sketch as a symbolic representation of the raw image. In our model, the sparse coding model can be viewed as a mathematical formulation of Julesz's textons or Marr's tokens.

The Markov random field model can be viewed as a mathematical formulation of Julesz's first conjecture on feature statistics. Marr argued that the primal sketch representation should be parsimonious and sufficient to reconstruct the original image without much perceivable distortion. This is reflected in our model by the fact that it is generative, and it can be checked by synthesizing images from fitted models to see if the synthesized images are visually similar to the original image.

The filters we used are biologically motivated by neurological observations of V1. Our model can be considered as a scheme for processing information after linear filtering, where the strong and highly aligned responses are combined into coding functions with explicit geometric and photometric parameters, and the weak responses that are not aligned are grouped into marginal histograms as texture statistics. The edge and ridge coding functions may also account for the joint distributions of filter responses studied by Portilla and Simoncelli [21].

In comparison with Canny edge detection [3], our representation is generative, and is more complete, in the sense that we have not only a sketch graph for the structure domain, but also texture patterns on the texture domain.

References

[1] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing*, Springer, 2002.

[2] J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems (with discussion)", *J. Royal Statist. Soc., series B*, vol.36. pp. 192-236, 1974.

[3] J. Canny. "A computational approach to edge detection", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8**:679–698, 1986.

[4] T. Chan and J. Shen, "Local inpainting model and TV inpainting", *SIAM J. of Appl. Math*, 62:3, 1019-43, 2001.

[5] D. Chandler, *Introduction to Modern Statistical Mechanics*, Oxford University Press, 1987.

[6] 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.

[7] J.H. Elder and S. W. Zucker, "Local scale control for edge detection and blur estimation", *IEEE Trans. PAMI*, vol. 20, no. 7, 699-716, 1998.

[8] S. Geman and C. Graffigne, "Markov random field image models and their applications to computer vision", *Proceedings of the International Congress of Mathematicians*, Vol. 1, 2, 1496-1517, 1987.

[9] A. Gersho and R. M. Gray, *Vector Quantization and Signal Processing.* Boston, MA: Kluwer, 1992.

[10] D. J. Heeger and J. R. Bergen, "Pyramid Based Texture Analysis/Synthesis", *Computer Graphics Proc.*, pp. 229-238, 1995.

[11] B. Julesz, "Visual pattern discrimination", *IRE Transactions on Information Theory*, IT-8, pp. 84-92, 1962.

[12] B. Julesz, "Textons, the elements of texture perception and their interactions", *Nature*, Vol. 290, pp. 91-97, 1981.

[13] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

[14] S. Mallat and Z. Zhang, "Matching Pursuit in a Time-Frequency Dictionary", *IEEE Sig. Proc.*, 41, 3397-415, 1993.

[15] D. Marr, *Vision,* W. H. Freeman and Company, 1982.

[16] L. Moisan, A. Desolneux, and J.-M. Morel, "Meaningful Alignments", *Int'l J. Computer Vision*, vol. 40, no. 1, pp. 7-23, 2000.

[17] D. Mumford and J. Shah, "Optimal Approx. by Piecewise Smooth Functions and Assoc. Variational Prob.", *Comm. in Pure and Appl. Math*, 42(5), 577-685, 1989.

[18] B. A. Olshausen and D. J. Field, "Emergence of Simple-cell Receptive Field Properties by Learning a Sparse Code for Natural Images", *Nature*, Vol. 381, pp. 607-609, 1996.

[19] A. Pece, "The Problem of Sparse Image Coding", *Journal of Mathematical Imaging and Vision*, vol. 17(2), pp. 89-108, 2002.

[20] A. Pece, "Contour Tracking Based on Marginalized Likelihood Ratios", *Image and Vision Computing*, 2005.

[21] J. Portilla and E.P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients", *Int'l Journal of Computer Vision*, 40(1):49-71, October, 2000.

[22] Z. Tu, X. Chen, A. Yuille, and S.-C Zhu, "Image Parsing: Unifying Segmentation, Detection, and Object Recognition", *Int'l J. Computer Vision*, 2005.

[23] G. Winkler, *Image Analysis, Random Fields, and Dynamic Monte Carlo Methods*, Springer, 1995.

[24] Y. N. Wu, S. C. Zhu, and X. W. Liu, "Equivalence of Julesz Ensemble and FRAME Models", *Int'l Journal of Computer Vision*, 38(3):245–261, 2000.

[25] R. A. Young, "The Gaussian Derivative Model for Spatial Vision: I. Retinal Mechanism", *Spatial Vision*, 2(4), 273-293, 1987.

[26] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax Entropy Principle and Its Applications in Texture Modeling", *Neural Computation*, 9(8), 1627-1660, 1997.