Representing Images Using Points on Image Surfaces

Sheng-Jyh Wang, *Member, IEEE*, Lun-Chia Kuo, Hsin-Haw Jong, and Zong-Han Wu

***Abstract*—This paper presents a new approach to represent an image by "verge points," which are defined as high-curvature points on the image surface. This representation offers a compact and reversible way to preserve the essence of the original image. Various applications, such as compression, edge detection, image enhancement, and image editing, can be achieved based on this representation. In this paper, the whole procedure for verge point representation is presented. Based on these verge points, image reconstruction can be easily achieved via iterative linear interpolation. These extracted verge points with compatible properties are further linked into verge curves to offer more compact representation. Progressive representation is also developed based on a multiscale extraction scheme. Some potential applications are then presented to demonstrate the versatility of this representation.**

***Index Terms*—B-spline approximation, differential geometry, edge detection, image compression, image editing, image enhancement, image representation, image surface.**

I. Introduction

TRADITIONALLY, the raw data of an image are often recorded in array form, with each array element representing the achromatic and/or chromatic information at a corresponding image pixel. In many applications, a direct use of this form may face two major drawbacks: bulkiness and inefficiency. The first drawback comes from the large number of pixels involved in an image, while the second drawback comes from the fact that the spatial features, like boundaries and smooth regions, are not explicitly specified. To find a more effective way to represent images, plenty of methods have been proposed in the literature. For example, boundary-based methods, like chain code and signature [1]–[3], have been proposed to describe an image in terms of object boundaries. Region-based methods, like quadtree decomposition [4], [5], have been proposed to describe an image in terms of smooth regions. Transform-based methods, like discrete cosine transform (DCT) and discrete Fourier transform (DFT) [6]–[8], have been used to describe an image in terms of its transform coefficients. Multiresolution methods, like Gaussian pyramid and wavelet decomposition, have been proposed to describe an image in hierarchical forms [9]–[11]. Each representation has its strong points and its suitable applications. In recent years, new methods are still emerging, trying to offer new ways to effectively represent images. For example, in [12], an image is represented in an edge-based approach by parametrically modeling relevant image surface variations. An approximation of the original image can be reconstructed under the framework of regularization theory. In [13], singular fractal components are used to represent images. In [14], an image is represented by Gaussian Markov random field parameters in a multiresolution way. In [15], the components obtained by multiscale differential operators are used to represent images. In [16], multiscale edges, accompanied with a suitable wavelet model, are used to decompose an image. In [17], a set of block pattern models that satisfy certain image variation constraints are used to represent images.

In this paper, we propose a new image representation, which is based on "verge points" on image surface. With this representation, image data can be greatly reduced. A visually similar approximation of the original image can be reconstructed from these verge points via iterative linear interpolation. This representation can also be applied to various applications, like image compression, edge detection, image enhancement, and image editing. In this paper, the basic concept of verge points is introduced in Section II first. Then, the extraction of verge points is described in Section III. In Section IV, we discuss the reconstruction of image using verge points. We also discuss the B-spline representation that may further condense the data. In Section V, we present a few potential applications of this representation, like image compression, edge detection, image enhancement, and image editing. Finally, in Section VI, we conclude this paper.

II. Concept of Verge Points

We first discuss the concept of verge points from the viewpoint of a one-dimensional (1-D) intensity profile. Fig. 1(a) shows an intensity profile extracted from a real image. Fig. 1(b) shows the edge points extracted from this profile. Traditionally, these edge points are used as the major features of the profile. Based merely on these edge points, however, the reconstruction of the original image is not an easy task. In [12], several edge parameters, like contrast, width, and edge center, are recorded and the image reconstruction is achieved via a regularization algorithm. In [16], a multiscale edge representation is adopted and the image reconstruction is achieved via an iterative projection algorithm. In this paper, we propose the use of some other features that can effectively represent an intensity profile. Here, we imagine this intensity profile as an elastic string stretched by a few pulleys, as shown in Fig. 1(c). Conceptually, these pulleys locate at highly curved places, and the radius of each pulley is inversely proportional to the local curvature of the profile. The location and size of these pulleys offer sufficient information to describe the outline of the profile. As shown in Fig. 1(d), both step edges and ridges can be well represented in this manner.

Manuscript received August 20, 2003; revised August 8, 2004. This work was supported by the National Science Council of the Republic of China under Grant NSC-91-2219-E-009-042. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ivan W. Selesnick.

S.-J. Wang and L.-C. Kuo are with the Department and Institute of Electronics Engineering, National Chiao-Tung University (NCTU), Hsinchu 30050, Taiwan, R.O.C. (e-mail: shengjyh@cc.nctu.edu.tw; u8911835@cc.nctu.edu.tw).

H.-H. Jong is with eXtreme Graphics Innovation Corporation, Hsinchu 300, Taiwan, R.O.C. (e-mail: hh_chung@xgitech.com).

Z.-H. Wu is with Sunplus Technology, Hsinchu 300, Taiwan, R.O.C. (e-mail: wzh@sunplus.com.tw).

Digital Object Identifier 10.1109/TIP.2005.851693

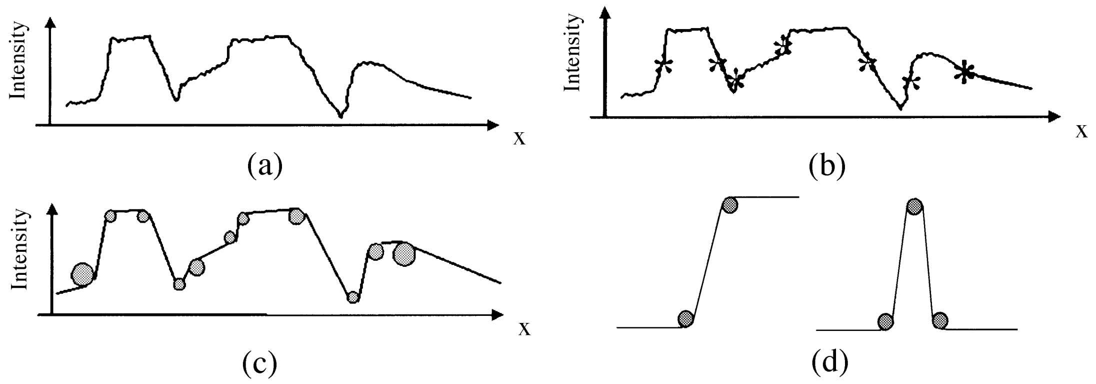


Fig. 1. Concept of verge points. (a) Original profile. (b) Edge points on the profile (marked as "*" signs). (c) Profile emulation using an elastic string and a few pulleys (verge points). (d) Verge point representation for (left) step edges and (right) ridges.

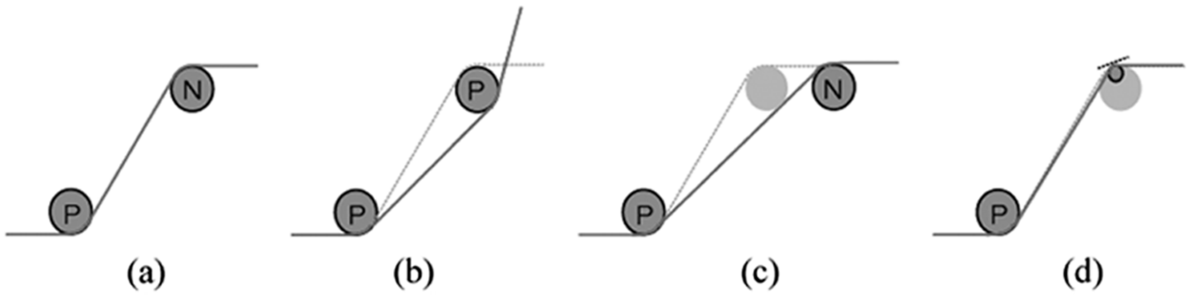


Fig. 2. (a) Original pulleys. (b) Reconstructed profile with the sign of the right pulley being incorrectly recorded. (c) Reconstructed profile with the position of the right pulley being incorrectly recorded. (d) Reconstructed profile with the radius of the right pulley being incorrectly recorded.

Intuitively, to describe a pulley, three parameters are needed: pulley position, pulley radius, and pulley sign. The definition of pulley sign is based on the sign of profile curvature at that pulley. A positive pulley bends down the string to form a convex arc; while a negative pulley bends up the string to form a concave arc. To derive a more compact form to represent pulleys, an intuitive analysis is illustrated in Fig. 2 to evaluate the impact of these three parameters. In Fig. 2(a), a profile reconstructed from two pulleys with opposite signs (a positive and a negative) is illustrated. Fig. 2(b)–(d) shows the reconstructed profiles when one parameter of the right pulley is mistakenly recorded. It can be seen that both pulley sign and pulley position are crucial for profile representation, while the size of pulley is less critical for profile reconstruction as long as the position of the tangential point can be accurately located. Based on this observation, we may shrink pulleys down to their tangential points. In this paper, these tangential points are called the "verge points" of the profile. With this reduction, the number of pulley parameters is reduced from three to two. Actually, the sign of verge point is no longer needed in profile reconstruction. However, for the sake of image analysis and image enhancement, we still preserve this sign information.

Once the major verge points of a profile are extracted, these points can be used to detect edges. An edge on the profile is a place where the imaginary elastic string is heavily deformed. The edge strength can be defined as the deformation per unit length of the stretched string. In material mechanics, the deformation $\Delta \mathrm{L}$ per unit length L is defined as strain. Assume the distance between two adjacent verge points is $\mathrm{W_d}$ and the intensity difference between these two verge points is $\mathrm{H_d}$. Then, the strain between two verge points can be estimated by $\varepsilon = (|H_d|)/(W_d)$. This strain parameter can serve as a parameter to perform edge detection.

III. Extracting and Linking of Verge Points

A. Extraction of Verge Points

The concept of verge points can be extended to represent two-dimemsional (2-D) images. Since the curvature information on an image surface is orientation-dependent, we may not simply extend the pulleys from 2-D circles to three-dimemsional (3-D) balls. Here, we treat the image surface as a plastic cloth stretched by 3-D pipes. By placing pipes at high-curvature points, the stretched rubber cloth emulates the image surface.

In differential geometry, these high-curvature points happen at the positions where at least one of the two principal curvatures has a large enough magnitude [18]. For an image surface in the form of $(\mathrm{x}, \mathrm{y}, \mathrm{f}(\mathrm{x}, \mathrm{y}))$, the directions of these two principal curvatures can be deduced by calculating the eigenvalues and eigenvectors of (1), shown at the bottom of the page [18]. In this paper, we denote $k_1(x,y)$ as the eigenvalue with the larger magnitude and de-

$$\mathbf{A} = \frac{1}{\left(1+f_x^2+f_y^2\right)^{3/2}} \begin{bmatrix} -f_{xy}f_xf_y + f_{xx}(1+f_y^2) & -f_{yy}f_xf_y + f_{xy}(1+f_y^2) \\ -f_{xx}f_xf_y + f_{xy}(1+f_x^2) & -f_{xy}f_xf_y + f_{yy}(1+f_x^2) \end{bmatrix} \tag{1}$$

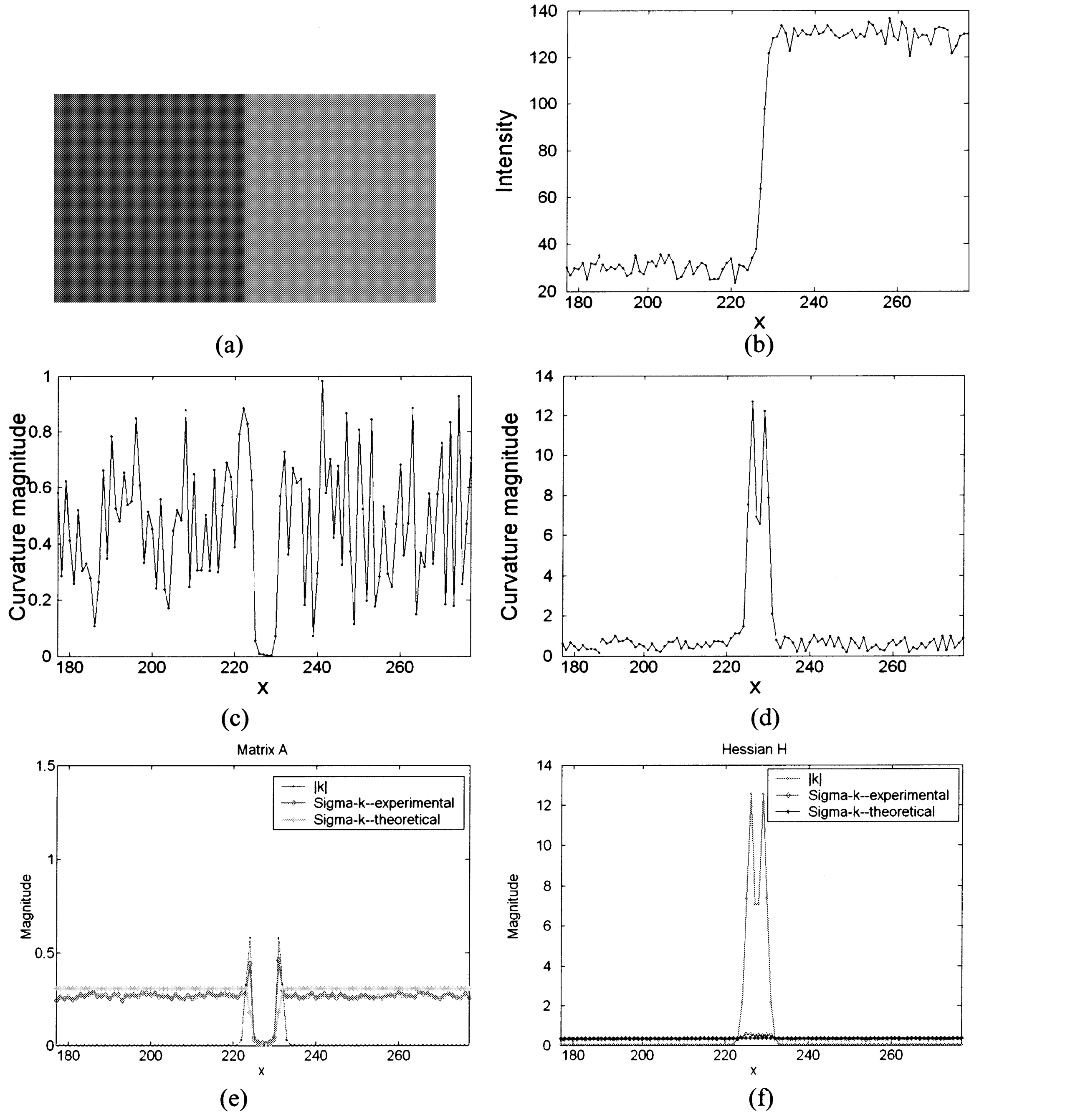


Fig. 3. Comparisons between A and H, in terms of SNR performance. (a) Step-edge image polluted by white Gaussian noise with $\sigma_n = 2$. (b) A transversal profile of (a). (c) Computed curvature profile based on Matrix A. (d) Computed curvature profile based on Hessian H. (e) Expected $|k|$ profile without noise interference (cyan) and the standard deviation of $|\Delta k|$ (pink and yellow), based on Matrix A. (f) Expected $|\hat{k}|$ profile without noise interference (red) and the standard deviation of $|\Delta \hat{k}|$ (green and blue), based on Hessian H (remark: $\sigma_m = 1$ for this simulation).

note $\Lambda_1(x,y)$ as the corresponding eigenvector of $k_1(x,y)$. On the contrary, $k_2(x,y)$ is the eigenvalue with the smaller magnitude and $\Lambda_2(x,y)$ is the eigenvector for $k_2(x,y)$. Here, f_x and f_y denote the first derivatives of f(x, y), while f_{xx}, f_{yy}, and f_{xy} denote the second-order derivatives. To suppress noise, we incorporate the Gaussian smoothing operation into these differentiation operators. That is, these derivatives are calculated as

$$f_x \equiv \frac{\partial(f(x,y) * G(x,y))}{\partial x} = f(x,y) * \frac{\partial G(x,y)}{\partial x}$$
$$f_{xx} \equiv \frac{\partial^2(f(x,y) * G(x,y))}{\partial x^2} = f(x,y) * \frac{\partial^2 G(x,y)}{\partial x^2}$$

and so on. Here, G(x,y) denotes the Gaussian smoothing function

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}$$

and the symbol "*" denotes the convolution operation. In this paper, we choose $\sigma_\mathrm{x} = \sigma_\mathrm{y} = \sigma_\mathrm{m}$. A larger σ_m produces better noise suppression but, at the same time, distorts the signal more, and vice versa.

In practice, however, the calculation of principal curvatures based on (1) has a poor SNR performance. Fig. 3 illustrates such a phenomenon. Fig. 3(a) shows an image of step edge polluted by white Gaussian noise with $\sigma_n = 2$ and Fig. 3(b) shows the traversal profile of Fig. 3(a). Fig. 3(c) shows the estimated $|k_1|$ profile based on Matrix $\mathbf{A}$. It appears that the curvature estimation is so noisy that the detection of verge points becomes extremely difficult.

In our approach, the Hessian matrix $\mathbf{H}$ is used for the estimation of principal curvatures. The Hessian $\mathbf{H}$ ignores all the first-order terms in $\mathbf{A}$ and is expressed as

$$\mathbf{H} = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}. \tag{2}$$

Similarly, the eigenvalues of $\mathbf{H}$ are used as the estimation of principal curvatures. Fig. 3(d) shows the estimated $|\hat{k}_1|$ profile based on $\mathbf{H}$. It can be seen that the use of $\mathbf{H}$, instead of $\mathbf{A}$, offers a much improved signal-to-noise ratio (SNR) in estimating principal curvatures. To give a quantitative description of this phenomenon, we compare $\mathbf{A}$ and $\mathbf{H}$ in terms of their SNR performance. Without loss of generality, we discuss the case of a vertical step edge as shown in Fig. 3(a). Assume the principal curvatures estimated from $\mathbf{A}$ and $\mathbf{H}$ are denoted as k and $\hat{k}$, respectively. As indicated in Appendix B, $k \approx (f_{xx})/((1+(f_x)^2)^{3/2})$ and $\hat{k} \approx f_{xx}$. Due to the existence of f_x in the denominator of k, the peak value of k is usually much smaller than that of $\hat{k}$. On the other hand, due to image noise, there are fluctuations in the estimations of k and $\hat{k}$. It is proved in Appendix B that

$$\mathrm{Var}(\Delta k) \approx \frac{1}{(1+(f_x)^2)^3}\mathrm{Var}(\Delta f_{xx}) + \frac{9(f_{xx}f_x)^2}{(1+(f_x)^2)^5}\mathrm{Var}(\Delta f_x)$$

and $\mathrm{Var}(\Delta\hat{k}) \approx \mathrm{Var}(\Delta f_{xx})$ around the edge region; while

$$\mathrm{Var}(|\Delta k|) \approx \mathrm{Var}(|\Delta\hat{k}|) = \frac{\sigma_n^2}{4\sigma_m^6}\left\{\frac{1}{\pi} - \frac{1}{\pi^2} - \frac{1}{8}\right\}$$

around smooth regions. With $|\hat{\mathrm{k}}|_{\mathrm{peak}} \gg |\mathrm{k}|_{\mathrm{peak}}$ and $\mathrm{var}(|\Delta\hat{k}|) \approx \mathrm{var}(|\Delta k|)$, the Hessian $\mathbf{H}$ does provide a better SNR performance than $\mathbf{A}$. All these formulae have been verified via computer simulations. Fig. 3(e) and (f) illustrates the computed $|k_1|$ profile and $|\hat{k}_1|$ profile, together with the standard deviations of $|\Delta k|$ and $|\Delta\hat{k}|$, for this step image in Fig. 3(a). These two figures verify that the SNR performance of $\mathbf{H}$ is superior to that of $\mathbf{A}$.

In addition to Matrix $\mathbf{A}$ and Hessian $\mathbf{H}$, there exist some other methods for curvature estimation. For example, in [19], a parameterized curvilinear model is used and curvatures are estimated by minimizing the residual energy measured along the model gradient. In [20], an n-dimensional estimator is proposed to estimate curvatures in images of higher dimensions. Hence, the curvature estimation method adopted in this paper can be considered as an independent module and can be replaced by alternative curvature estimators.

To obtain the verge points of an image $\mathrm{f(x,y)}$, we check at each pixel the value of $|\hat{k}_1(x,y)|$. If $|\hat{k}_1(x,y)|$ is a local maximum, that pixel is marked as a candidate of verge points. Due to image noise, plenty of candidates are actually false alarm and we need a threshold $\mathrm{T_k}$ to remove them. The use of a larger $\mathrm{T_k}$ suppresses false alarms but, at the same time, suppresses the detection of many image details. On the contrary, the use of a smaller $\mathrm{T_k}$ allows more tiny features to be detected but, at the same time, creates more false alarms. In our approach, we choose this threshold based on the statistical distribution of false alarms. Over smooth regions, $\hat{k}_1(x,y)$ is presumed to be zero. Due to image noise, $\hat{k}_1(x,y)$ actually fluctuates around zero. Once $|\hat{k}_1(x,y)|$ exceeds $\mathrm{T_k}$, a candidate of verge point is mistakenly detected. Based on the formulae deduced in Appendix B, we have

$$E[|\hat{k}_1|] = \frac{1}{2}\left(\frac{1}{2\sqrt{2}} + \frac{1}{\pi}\right)\frac{\sigma_n}{\sigma_m^3} \tag{3}$$

and

$$\mathrm{Var}[|\hat{k}_1|] \equiv \sigma_{k_1}^2 = (\frac{1}{\pi} - \frac{1}{\pi^2} - \frac{1}{8})\frac{\sigma_n^2}{4\sigma_m^6}. \tag{4}$$

Hence, once the noise power σ_n^2 in the image can be estimated, $\mathrm{T_k}$ can be chosen to be $E[|\hat{k}_1|] + m\sigma_{k_1}$. Here, m is to be determined by the user. A typical choice of m is 2 or 3. Moreover, to estimate σ_n, there exist several noise estimation methods in the literature [21]–[23]. In this paper, we adopt the method proposed in [21]. In [21], an image is assumed to contain Gaussian distributed noise and has sufficient background area. Under these assumptions, the fluctuations in the gradient components f_x and f_y follow the Gaussian distribution and are closely related to σ_n. Moreover, the pdf (probability density function) for the magnitude of the gradient (f_x, f_y) follows a Rayleigh-like distribution. By detecting the peak of the Rayleigh pdf, the fluctuations of gradient components can be estimated and σ_n can, thus, be deduced. For the 256×256 Lena image used in this paper, σ_n is estimated to be 2.1. If we choose $\sigma_\mathrm{m} = 1$, then the threshold is set to be $\mathrm{T_k} = 2.0$ if $\mathrm{m} = 2$, or $\mathrm{T_k} = 2.6$ if $\mathrm{m} = 3$. Fig. 4(a) shows a 256×256 Lena image. Fig. 4(b) shows the $\hat{k}_1(x,y)$ calculated from $\mathbf{H}$, with positive values in red and negative values in green. Fig. 4(c) shows the directions of $\Lambda_2(x,y)$. Fig. 4(d) shows the extracted verge points after thresholding. In this simulation, we choose $\sigma_\mathrm{m} = 1$ and $\mathrm{T_k} = 2.0$.

B. Linking of Verge Points

Once verge points are extracted from a 2-D image, these verge points can be further linked into "verge curves." These verge curves not only reflect the shapes of objects in an image, but also allow an easier manipulation of the image surface. In addition, it costs less to represent a linked curve than to represent a whole set of verge points.

To link verge points into verge curves, a two-phase linking scheme similar to the hysteresis approach used in the Canny Operator [24] is adopted. In the first phase, verge points with $|\hat{k}_1|$ larger than $\mathrm{T_k}$ are to be linked first. Adjacent verge points at $\mathrm{(x,y)}$ and $\mathrm{(x',y')}$ are linked into curves if they satisfy the constraints on curvature sign S and angle difference T_θ; that is, we test whether

$$S(x,y) \times S(x',y') > 0$$

and

$$\|\Lambda_2(x',y') - \Lambda_2(x,y)\| < T_\theta. \tag{5}$$

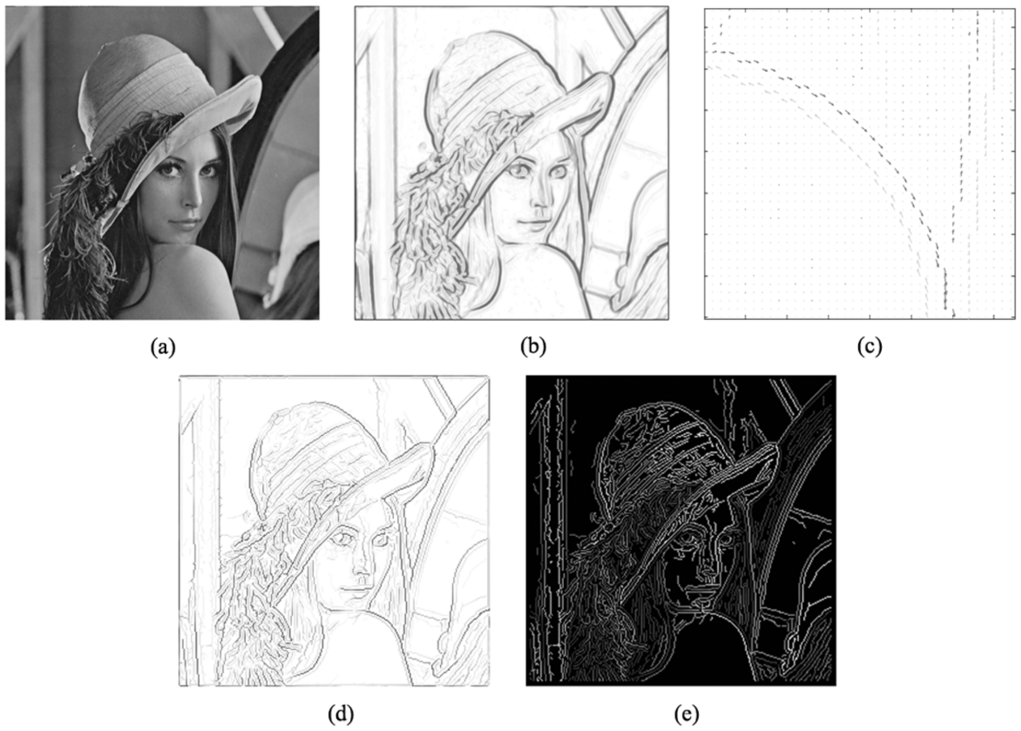


Fig. 4. (a) Original image (256 by 256). (b) $\hat{k}_1(x, y)$ calculated from $\mathbf{H}$, with positive curvatures in red and negative curvatures in green. (c) Eigenvectors $\Lambda_2(x, y)$ around Lena's shoulder. (d) Extracted verge points. (e) Verge curves represented in terms of intensity values.

The first constraint means these two verge points need to have the same sign of curvature. The second constraint reflects the requirement for curve shape. The use of a larger T_θ generates long, but sometimes curvy, verge curves. The use of a smaller T_θ generates smooth, but sometimes short, verge curves. The selection of T_θ influences the shape of linked curves, but has little impact on reconstruction quality. In our approach, we prefer smooth linking and set T_θ to be $\pi/8$ empirically. In the second phase of linking process, we extend the endpoints of linked curves to further link these candidates with $|\hat{k}_1|$ smaller than T_k. Similar to the first phase, we still apply the constraints of (5) for curve linking.

The verge curves finally extracted are shown in Fig. 4(e). In Fig. 4(e), these linked verge curves are represented in terms of their intensity values. It can be seen that Fig. 4(e) offers a compact way to represent the original image. As will be shown in the next section, a reconstruction of Fig. 4(a) based on 4(e) can be achieved via simple iterative linear interpolation.

IV. Image Reconstruction and Data Representation

A. Image Reconstruction

After extracting verge curves, these linked curves can be imagined as 3-D pipes in the $(x, y, f(x, y))$ space. These 3-D pipes record the critical positions of the image surface, while the nonverge-curve parts of the image surface can be imagined as smooth patches stretched by these 3-D pipes. Although there may exist several ways to interpolate the nonverge-curve parts of the image surface, we aim for a simple interpolation approach for the reconstruction of image surface based on these extracted verge points. In Fig. 5(a), we show the reconstructed image by linearly interpolating the intensity values of the verge points in Fig. 4(e) to fill in the nonverge-curve parts of the image. Here, the interpolated intensity value at a target pixel is computed as

$$I_r = \frac{\sum w_i I_i}{\sum w_i}, \quad \text{where } w_i = 1/d_i. \tag{6}$$

In (6), I_r denotes the interpolated intensity value at the target pixel, I_i denotes the intensity value at p_i, which is the nearest verge point searched along one of the four directions (up, down, left, and right), and w_is denote the weightings that are inversely proportional to the distance between p_i and the target pixel. This linear interpolation is fast in computation but may generate a less smooth image.

To achieve a smooth reconstruction, the iterative interpolation scheme proposed by Itoh is adopted [25]. With Itoh's approach, the pixels next to these verge points are linearly interpolated first, using the same equation in (6). These newly interpolated pixels are then used as reference pixels to interpolate their neighboring pixels. The same procedure continues until no further pixel needs to be processed. The interpolation results of the first few iterations are shown in Fig. 5(b)–(f), and the final result is shown in Fig. 5(g). Compared with the straightforward linear interpolation method, the iterative linear method offers smoother reconstruction and is less sensitive to the missing or adding of verge points. The difference between the original image and the iteratively reconstructed image is shown in Fig. 5(h). Most differences appear around edges or lines. Moreover, even though there is a large difference over the upper arm of Lena, the produced distortion is not visually apparent as long as the original image is not to be placed side by side with the reconstructed image.

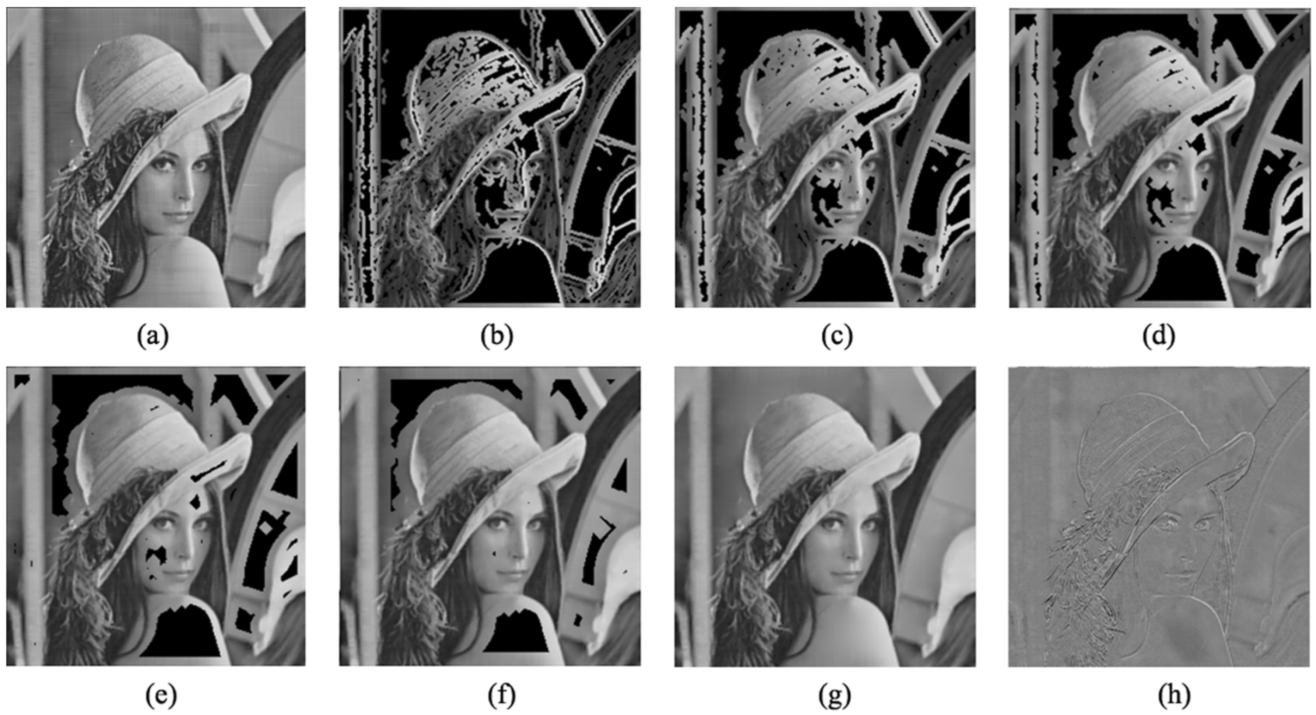


Fig. 5. (a) Reconstructed image using direct linear interpolation. (b)–(f) Reconstructed images in the first few iterations of the iterative linear interpolation. (g) Final result using iterative linear interpolation. (h) The difference image between the original image and the reconstructed image in (g).

Conceptually, the selections of neighboring verge points for interpolation should depend on the direction of verge curve. A more reasonable way is to perform the interpolation process along the direction perpendicular to the verge curve. However, this type of interpolation will be cumbersome and technically difficult. Fortunately, according to our observations, neighboring verge points with the same curvature sign tend to have similar intensity values. With the property, different selections of neighboring verge pairs have less impact over the quality of the reconstruction results. Moreover, with the use of the iterative scheme, the reconstruction quality becomes even less sensitive to the selection of verge pairs. Hence, in our approach, we simply choose the vertical direction and horizontal direction as the directions for the interpolation process.

The intensity values of these verge points can be further quantized without producing significant impact on the quality of reconstructed images. In Fig. 6, we quantize the intensity values at these verge points into 2, 4, 8, 16, 32, 64, 128, and 256 levels, respectively. The eight reconstructed images based on the quantized intensity values are shown in Fig. 6(a). The relation between the quantization level and peak signal-to-noise ratio (PSNR) of these reconstructed images are shown in Fig. 6(b). Here, PSNR is defined as

$$\mathrm{PSNR} = 20 \times \log_{10}\left(\frac{I_{\max}}{\sqrt{\mathrm{MSE}}}\right) \tag{7}$$

with $\mathrm{I}_{\max}$ denoting the maximally allowed intensity value and MSE denoting the mean square error between the original image and the reconstructed image. For an 8-bit gray-level image, $\mathrm{I}_{\max} = 255$. In Fig. 6(b), it appears the PSNR remains steady if the quantization levels are larger than 16. Moreover, even though the 2-level quantization produces a fairly poor PSNR, the reconstructed Lena image still offers a rich description of the original image.

B. Reconstruction for Color Images

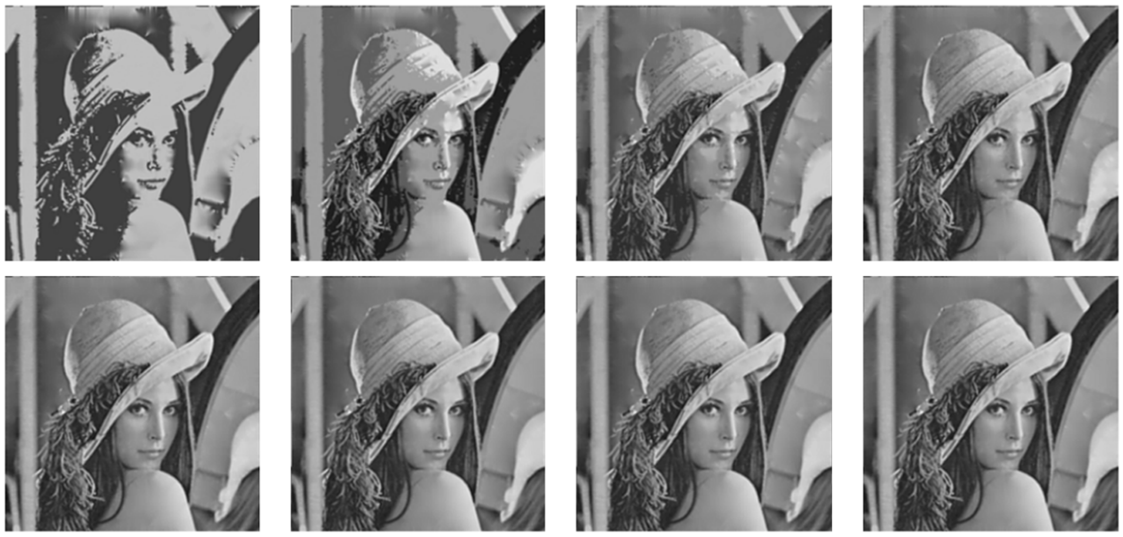
The concept of verge points is also applicable to color images. To extract verge points and verge curves from a color image, the original color image is first decomposed into three component images. Then, the image surface of each component image is processed separately. Oppositely, to reconstruct the original image from image verges, the image surface of each component image is reconstructed first and then all three reconstructed image surfaces are combined together to obtain the final color image.

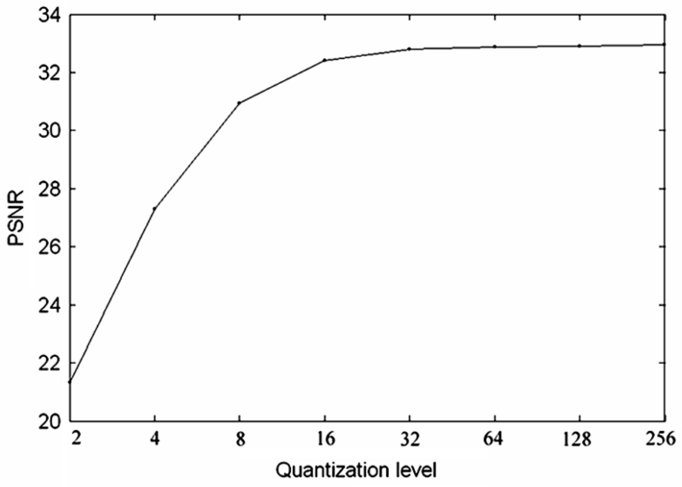
In this paper, we choose CIE $\mathrm{L^*a^*b^*}$ as the color space to demonstrate the feasibility of color image reconstruction. In this color space, $\mathrm{L^*}$ represents the achromatic component, while $\mathrm{a^*}$ and $\mathrm{b^*}$ represent the chromatic components. The conversion between RGB and $\mathrm{CIEL^*a^*b^*}$ can be found in color related books, like [26]. Fig. 7(a) shows the original color image. This color image is decomposed into $\mathrm{L^*, a^*}$, and $\mathrm{b^*}$ component images first. Then, verge points are extracted for each component image. Fig. 7(b) shows the reconstructed color image using all the verge curves extracted from these three component images. It can be seen that a visually similar reconstruction of the original color image is achievable.

C. B-Spline Curve Representation

As mentioned above, these extracted verge curves offer an effective way to represent images. Since the linked verge curves are usually smooth, we can further compress these verge curves by using B-spline approximation [27], [28]. With the B-spline approximation, each verge curve can be represented by a small number of control points.

In our approach, each verge curve is decomposed into the shape component and intensity component, as shown in


(a)



(b)

Fig. 6. (a) Comparison of reconstructed images. The intensity values at the verge points are quantized into 2, 4, 8, 16, 32, 64, 128, and 256 levels, respectively (from upper left to lower right). (b) The PSNR value of these reconstructed images.

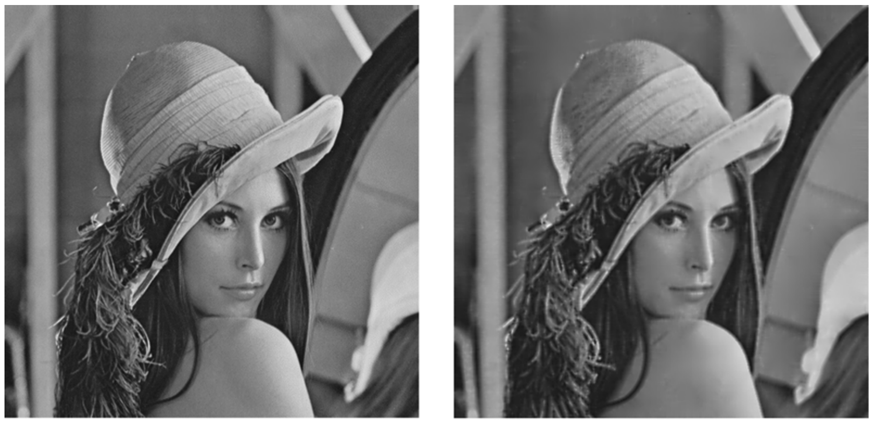

(a) (b)

Fig. 7. (a) Original color image (256 × 256). (b) Reconstructed color image under CIE $L^*a^*b^*$ color space, using 16 812 verge points for L^*, 12 537 verge points for a^*, and 14 416 verge points for b^*.

Fig. 8(a) and (b). For the shape component, we adopt the shape coding algorithm proposed in [29]. In [29], a third-order uniform B-spline curve approximation is adopted, and the positions of control points are calculated using a weighted directed acyclic

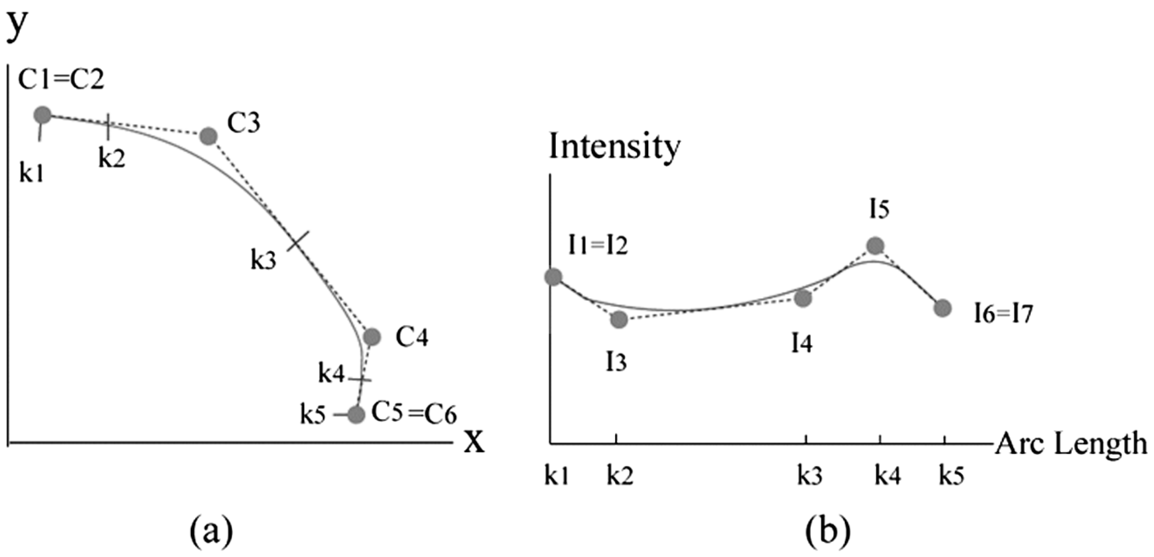


Fig. 8. B-spline approximation. (a) Shape component (C: shape control points; k: knots). (b) Intensity component (I: intensity control points; k: knots).

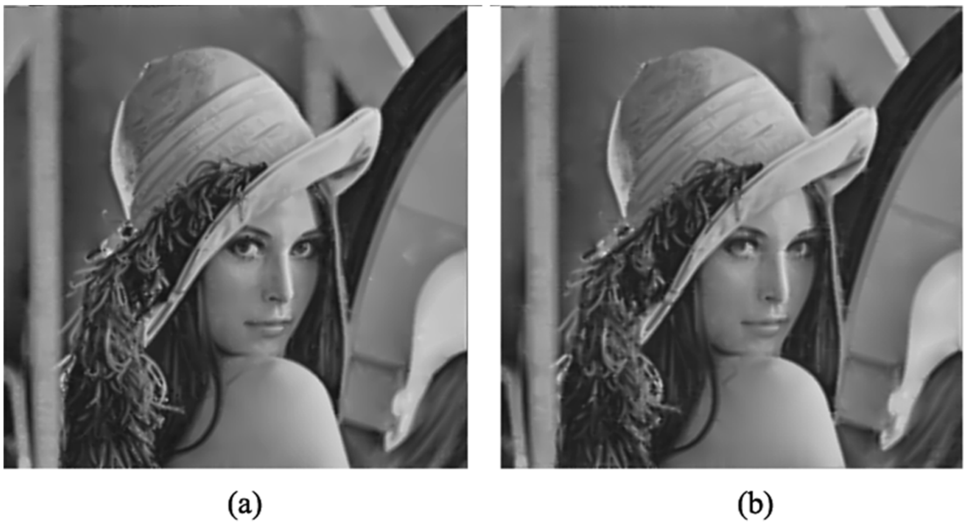


Fig. 9. (a) Reconstructed gray-level image using B-spline approximation (2957 control points). (b) Reconstructed color image using B-spline approximation under the CIE $L^*a^*b^*$ color space. (L^*: 3173 control points, a^*: 2783 control points, and b^*: 2974 control points).

graph. To reconstruct the intensity value in a given verge curve, we use the same knot vectors used for shape coding and calculate the arc length of each knots. For the approximation of intensity component, the horizontal positions of the control points are set as the arc length of each knot. The vertical position of control points are then calculated using least square fitting.

In the B-spline representation, these control points are used to represent verge curves. That is, the original list of N verge points is replaced by the list of M control points. Since M is usually much less than N, this representation provides an even more compact form for the original image. On the other hand, since the B-spline representation is only an approximation to the original verge curve, larger distortion is expected in the reconstructed image. However, this distortion can be properly restricted by setting an upper bound over approximation errors. Fig. 9(a) and (b) shows the reconstructed images based on the B-spline representation. These two images are still visually similar to the original images. In addition, due to the affine invariant property of B-spline curve, this B-spline representation is expected to be very suitable for spatial image scaling.

As mentioned above, both verge curve representation and B-spline representation offer effective ways to represent images. A straightforward approach to record verge curves is a hierarchical data structure as illustrated in Fig. 10. At the top layer, the verge curves for the L^* component image, a^* component image, and b^* component image are stored separately. Take the set of L^* verge curves as an example; the total number of verge curves is recorded first, followed by the sequence of verge curves. For each verge curve, the curvature sign and the total number of linked verge points are recorded first. Then, for each verge point, its $(x, y, f(x, y))$ coordinates are recorded as X, Y, and F, respectively. To achieve better compactness, only the first verge point of each verge curve is coded with the original value, while the remaining verge points are coded in a differential manner. For B-spline curve representation, a similar data structure can be adopted, with these verge points being replaced by control points.

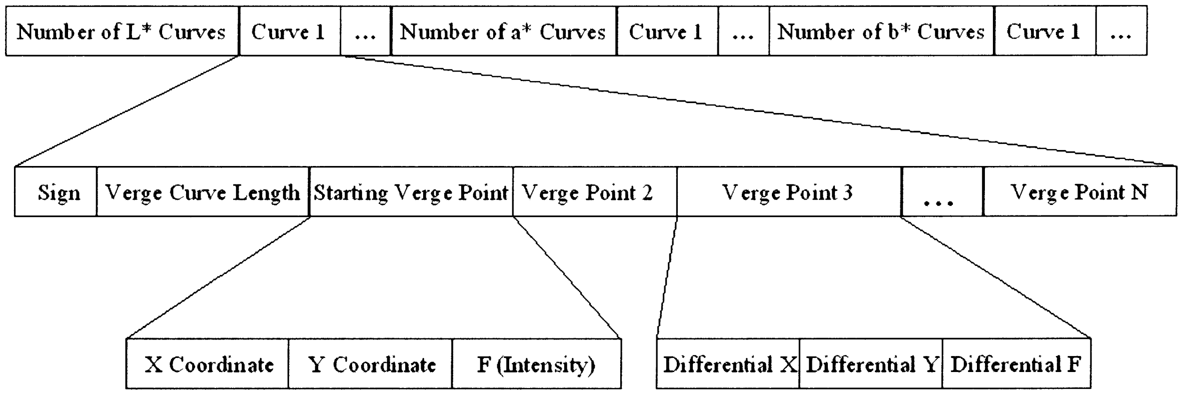


Fig. 10. Data structure for the storage of verge curves.

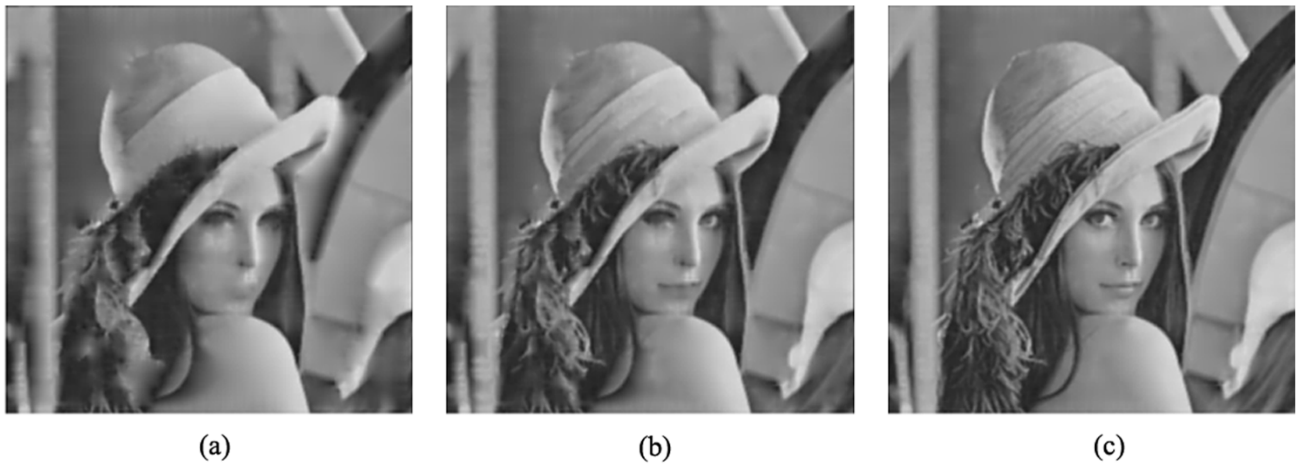


Fig. 11. Progressive image reconstruction. (a) Using "top-layer" verge curves only; totally, 8219 verge points. (b) Using "top-layer" and "middle-layer" verge curves; totally, 11 523 verge points. (c) Using all verge curves; totally, 15 926 verge points.

D. Progressive Image Representation

To represent verge curves in a progressive way, a multiscale approach proposed in [30] is adopted. Here, we further modified that method to support progressive transmission. In the modified approach, the original image is decomposed into a three-layer Gaussian pyramid, with $N/4*N/4$ pixels in the top layer, $N/2*N/2$ pixels in the middle layer, and $N \times N$ pixels in the bottom layer. The feature extraction and linking processes are first performed over all three layers separately. Then, the interlayer mappings between every pair of adjacent layers are examined in a top-down order, based on a curve-based strategy. Take the interlayer mapping between the top layer and the middle layer as an example. Given a verge curve C_k on the top layer, all its 8-connection neighboring pixels are labeled with the curvature sign of that verge curve. These labeled pixels are upsampled by a factor of two to form corresponding areas $R(C_k)$ in the middle layer. Then, in the middle layer, all these verge points locating within $R(C_k)$ are examined to see whether they possess the same curvature sign as C_k. Those verge points with the same sign are considered to be mapped from C_k. If half verge points of a curve in the middle layer are mapped from C_k, that curve is considered to be mapped from C_k and is tagged as a "top-layer" curve. On the other hand, if no verge curve is identified within $R(C_k)$, the feature detection and linking procedures are applied over $R(C_k)$ again with a larger mask sigma (e.g., $2\sigma_m$). Once new verge curves are extracted within $R(C_k)$, the procedure mentioned above is repeated again to tag these newly generated curves. After having identified all "top-layer" curves in the middle layer, the remaining curves are tagged as "middle-layer" curves. That is, all the curves in the middle layer are classified into "top-layer" curves and "middle-layer" curves.

Similarly, the mapping process is applied between the middle layer and the bottom layer to classify verge curves in the bottom layer into "top-layer" curves, "middle-layer" curves, and "bottom-layer" curves. After having classified all verge curves into these three different classes, different levels of image quality can be achieved by arranging the order of transmission, with "top-layer" curves first while "bottom-layer" curves last. With this arrangement, the capability of progressive reconstruction can be achieved as shown in Fig. 11. In Fig. 11(a), only "top-layer" verge curves are used to reconstruct the Lena image. This reconstructed image catches a gross outline of Lena, but lacks plenty of details. As the verge curves tagged to "middle-layer" curves and "bottom-layer" curves are transmitted, more and more details are revealed, as shown in Fig. 11(b) and (c), respectively. Furthermore, for verge curves of the same class, curve contrast may also be used as an indicator to determine the order of transmission, with large-contrast curves first while small-contrast curves last. With

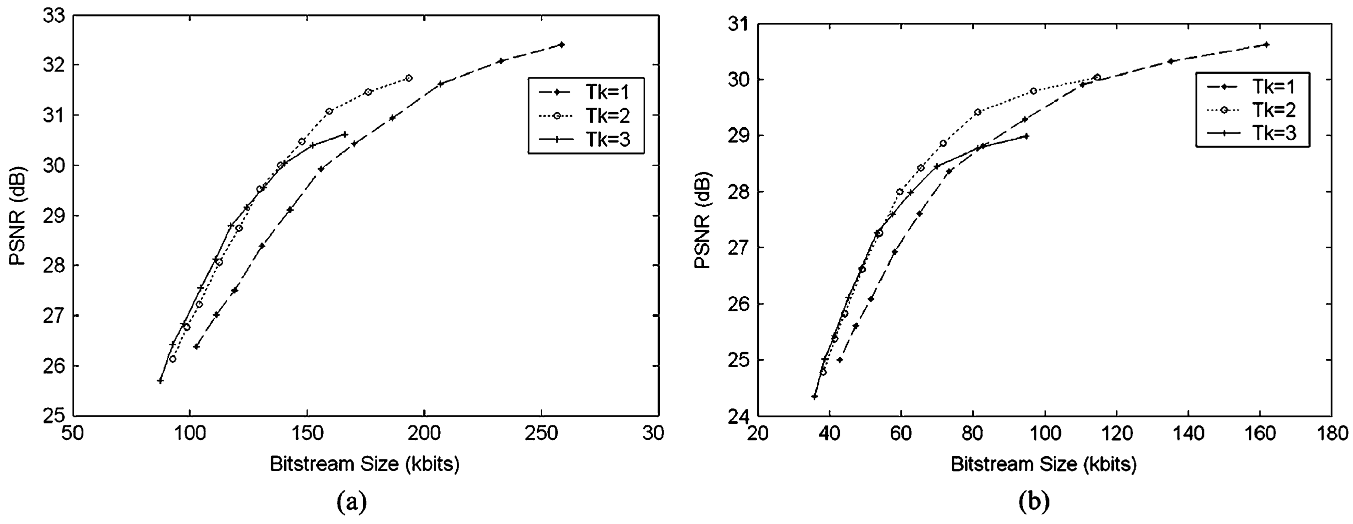


Fig. 12. (a) Relation between storage requirement and PSNR for different Tks, based on the verge curve representation (test image: 512 × 512 Lena image). (b) Relation between storage requirement and PSNR for different Tks, based on the B-spline curve representation (test image: 512 × 512 Lena image).

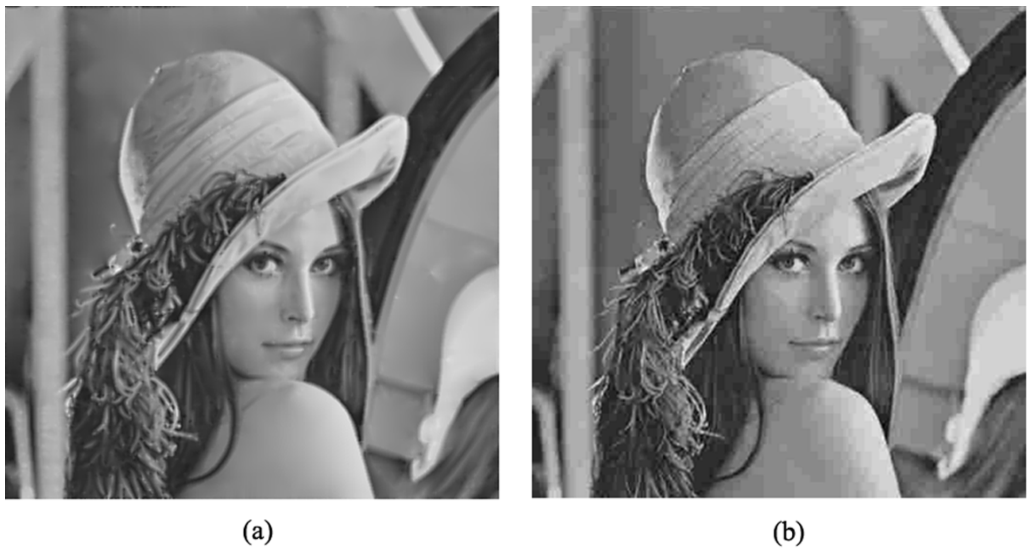


Fig. 13. Illustrations of two different compression algorithms. (a) Reconstructed image using the proposed method; compression rate $= 31.12$, PSNR $= 28.67$. (b) Reconstructed image using JPEG compression, compression rate $= 31.08$, PSNR $= 30.65$.

curve tag and curve contrast as the indicators, different levels of image quality can be achieved at different bitstream sizes. In Fig. 12(a), we show the PSNR of the reconstructed image with respect to the required bitstream size at different selections of curvature threshold $\mathrm{T_k}$. In Fig. 12(b), we further show the relation between PSNR and the required bitstream size for the case of B-spline curve representation.

V. Potential Applications

As mentioned in previous sections, the verge curves extracted from an image, or the B-spline control points computed from the verge curves, offer a new way for image representation. The fact of being able to reconstruct a visually similar image indicates that these verge curves must have kept the essence of the original image. Hence, these verge curves may not only be used for image representation, but also be used to manipulate or analyze the shape of image surface directly. In this section, we mention some potential usages of verge points/curves in various applications.

A. Feature-Based Image Compression

As mentioned in Section IV, 3-D verge pipes are decomposed into shape components and intensity components. The control points of the shape components can be differentially encoded using a modified chain code and a run-length coding [29]. In addition, as mentioned in Section IV, the image visual quality only drops slightly when the intensity values at the verge points are quantized down to 5 bits. After quantization, the dynamic range of differential coding is greatly reduced. Hence, based on B-spline representation, we have further developed an efficient compression algorithm. The reconstructed image using this compression scheme is shown in Fig. 13(a) with PSNR of 28.67 dB and compression rate of 31.12. As a comparison, the image compressed by JPEG is shown in Fig. 13(b) with PSNR of 30.65 dB and compression rate of 31.08. Even though there is a 2-dB difference in PSNR, the visual quality of Fig. 13(a) is quite similar to that of Fig. 13(b).

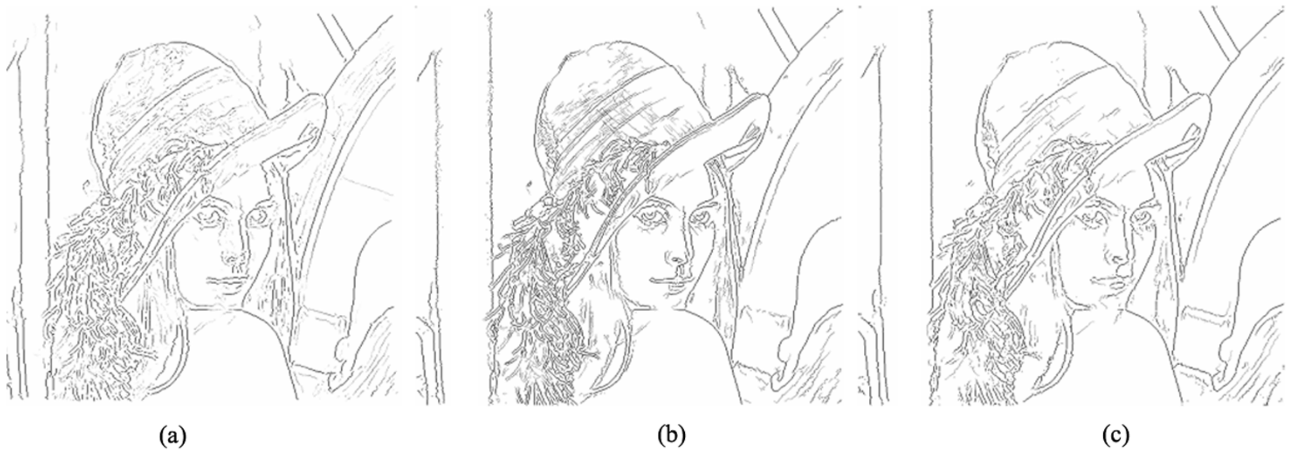


Fig. 14. (a) Edge detection based on verge points. (b) Edge detection based on the Canny Operator, where σ_{m} is set as 1 and the low- and high-hysteresis thresholds are set as 0.0375 and 0.0938, respectively. (c) Edge detection based on the Canny Operator, where σ_{m} is set as 2 and the low- and high-hysteresis thresholds are set as 0.0375 and 0.0938, respectively. The gray-level value represents the edge strength. The averaged edge strengths of these three figures are adjusted to be equal.

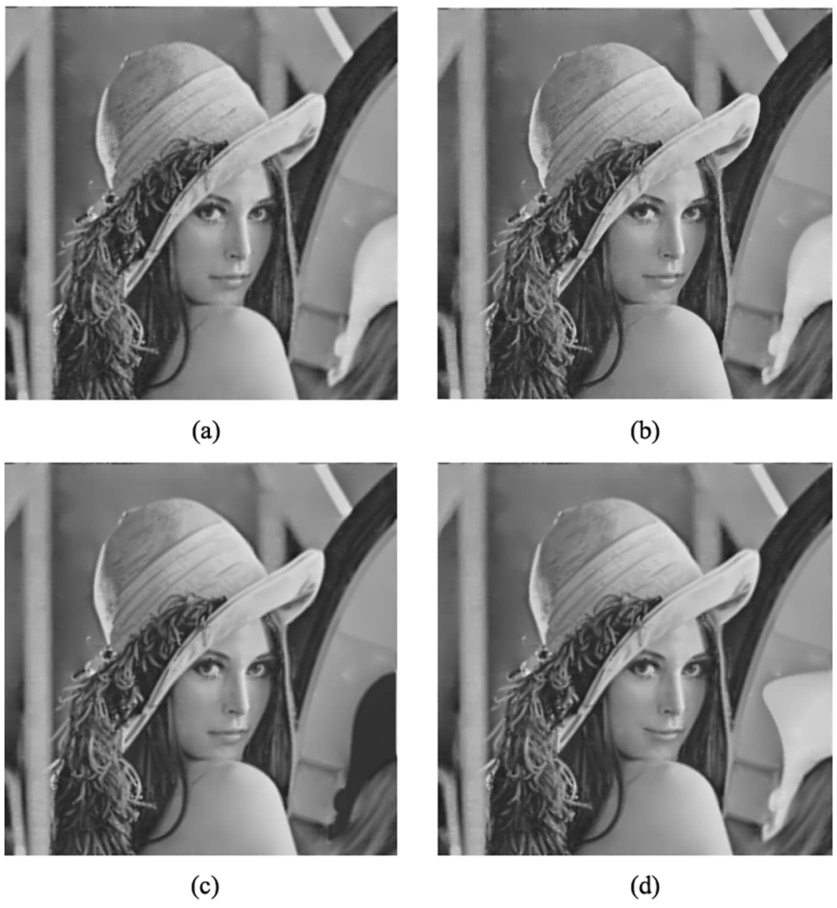


Fig. 15. Illustrations of image manipulations using verge points. (a) Reconstructed image with the original contrast. (b) Edge-enhanced image. (c) Image edited by changing the gray level of a region at the lower right corner. (d) Image edited by changing the shape of mouth and the shape of a region at the lower right corner.

B. Edge Detection

As mentioned in Section II, we can use the intensity slope (strain ε) between adjacent verge points to detect edges on a 1-D profile. Similarly, for a 2-D image, we can also use verge points to detect edges. For example, for a verge point P with sign S on an image, we search along eight different directions for adjacent verge points with opposite sign. We measure the strain ε for each possible pairs and select the pair with the maximum strain. If the intensity slope of the selected pair is above a preselected threshold, an edge is detected and the middle point of that pair is marked as an edge point. Fig. 14(a) shows the simulation result of this primitive edge detector, where $\mathrm{T_k} = 2$ and $\sigma_{\mathrm{m}} = 2$. As a comparison, edges detected by the well-known

Canny Operator are shown in Fig. 14(b) and Fig. 14(c), with the scale parameter being set as 1 and 2, respectively. These results are obtained using the Canny detector tool offered in Matlab 6.0. In this simulation, the low- and high-hysteresis thresholds of the Canny Operator are automatically set as 0.0375 and 0.0938, respectively. It can be seen that, even with such a primitive operator, the performance of edge detection is comparable to that of Canny Operator.

C. Feature-Based Image Enhancement and Editing

These verge points can also be used for contrast enhancement and sharpness enhancement. For an intensity boundary in an image, two verge curves with opposite sign are detected along the boundary curve. For these two verge curves, their intensity difference determines the contrast of the boundary, while their intensity slope determines the sharpness. Hence, to enhance contrast, we may simply enlarge the intensity difference between these two verge curves. To soften contrast, we may shorten the intensity difference between these two verge curves. To enhance sharpness, we may shorten the spatial distance between adjacent verge curves of opposite sign. In Fig. 15(a) and (b), we show the reconstructed Lena image and the edge-enhanced image, respectively.

Since verge curves preserve the shapes and contrast of objects, an interactive image editing can also be performed by manipulating the positions and altitudes of verge curves. For example, the removal of an object in an image can be done by simply removing the verge curves of that object. To adjust the gray level or color of an object, we can adjust the altitudes of the corresponding verge curves. In addition, image warping can be performed by adjusting the positions of B-spline control points. In Fig. 15(c) and (d), we demonstrate gray-level adjustment and shape editing over the Lena image, respectively.

VI. Conclusion

This paper proposes a new approach to represent images. This representation is inspired by emulating an image surface with a rubber cloth stretched by 3-D pipes. The whole procedure is developed with the aid of differential geometry to extract verge points from image surfaces. The extracted verge points with compatible properties are linked into verge curves. These extracted verge points offer an effective way to extract spatial features, like edges, of the original image. In addition, the extracted verge curves can be further compressed using B-spline approximation. Based on the extracted verge curves or the computed B-spline control vertices, the original image can be well reconstructed. Potential applications, such as compression, edge detection, image enhancement, and image editing, are also briefly presented. These simulation results have demonstrated the versatility of this proposed representation.

Appendix A

Due to image noise, there are fluctuations in the estimations of f_x, f_y, f_{xx}, f_{xy}, and f_{yy}. In this paper, we denote these fluctuations as $\Delta f_x, \Delta f_y, \Delta f_{xx}, \Delta f_{xy}$, and Δf_{yy}. This section is to calculate the means and variances of these fluctuations.

If we denote the signal part of an image as $\mathrm{s(x,y)}$ while the noise part as $\mathrm{n(x,y)}$, the estimation of f_x can be expressed as

$$\begin{aligned} f(x,y) * \frac{\partial G(x,y)}{\partial x} &= s(x,y) * \frac{\partial G(x,y)}{\partial x} \\ &\quad + n(x,y) * \frac{\partial G(x,y)}{\partial x} \\ &\equiv s(x,y) * \frac{\partial G(x,y)}{\partial x} + \Delta f_x(x,y). \end{aligned}$$

Assume the noise $\mathrm{n(x,y)}$ is additive white Gaussian noise with variance σ_n^2. Then, we have

$$\begin{aligned} &E[\Delta f_x(x,y)] \\ &\quad = E\left[n(x,y) * \frac{\partial G(x,y)}{\partial x}\right] \\ &\quad = E[n(x,y)] * \frac{\partial G(x,y)}{\partial x} = 0 \end{aligned} \tag{8}$$

and

$$\begin{aligned} &\mathrm{Var}[\Delta f_x(x,y)] \\ &\quad = E[(\Delta f_x(x,y))^2] \quad = E\left[\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_x\sigma_y}\left(-\frac{p}{\sigma_x^2}\right)\right.\right. \\ &\qquad \left.\left.\times e^{-\left(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}\right)} n(x-p,y-q)dpdq\right)^2\right] \\ &\quad = E\left[\left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\frac{1}{2\pi\sigma_x\sigma_y}\right)^2\left(-\frac{p}{\sigma_x^2}\right)\right.\right. \\ &\qquad \times\left(-\frac{\hat{p}}{\sigma_x^2}\right) e^{-\left(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2}\right)} \\ &\qquad \times n(x-p,y-q)n(x-\hat{p},y-\hat{q})dpdqd\hat{p}d\hat{q})] \\ &\quad = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\frac{1}{2\pi\sigma_x\sigma_y}\right)^2 \\ &\qquad \times\left(-\frac{p}{\sigma_x^2}\right)\left(-\frac{\hat{p}}{\sigma_x^2}\right) e^{-\left(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2}\right)} \\ &\qquad \times E[n(x-p,y-q)n(x-\hat{p},y-\hat{q})]dpdqd\hat{p}d\hat{q} \\ &\quad = \left(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\frac{1}{2\pi\sigma_x\sigma_y}\right)^2\right. \\ &\qquad \left.\times\left(-\frac{p}{\sigma_x^2}\right)^2 e^{-\left(\frac{p^2}{\sigma_x^2}+\frac{q^2}{\sigma_y^2}\right)} dpdq\right)\cdot\sigma_n^2 \\ &\quad = \frac{1}{4\sqrt{\pi}\sigma_x^3}\frac{1}{2\sqrt{\pi}\sigma_y}\sigma_n^2. \end{aligned} \tag{9}$$

In this paper, we choose $\sigma_x = \sigma_y = \sigma_m$. Hence, $\mathrm{Var}[\Delta f_x(x,y)] = (1/8\pi\sigma_m^4)\sigma_n^2$.

Similarly, we can deduce that $E[\Delta f_y(x,y)] = E[\Delta f_{xx}(x,y)] = E[\Delta f_{xy}(x,y)] = E[\Delta f_{yy}(x,y)] = 0$

$$\mathrm{Var}[\Delta f_y(x,y)] = \frac{1}{8\pi\sigma_m^4}\sigma_n^2 \tag{10}$$

$$\begin{aligned}\mathrm{Var}[\Delta f_{xx}(x,y)] &= \frac{3}{16\pi\sigma_m^6}\sigma_n^2 \\ &= \mathrm{Var}[\Delta f_{yy}(x,y)]\end{aligned} \tag{11}$$

and

$$\mathrm{Var}[\Delta f_{xy}(x,y)] = \frac{1}{16\pi\sigma_m^6}\sigma_n^2. \tag{12}$$

$$|\Delta\hat{k}_1| = \frac{|\Delta f_{xx} + \Delta f_{yy}| + \sqrt{(\Delta f_{xx} + \Delta f_{yy})^2 - 4(\Delta f_{xx}\Delta f_{yy} - \Delta f_{xy}^2)}}{2} \tag{17}$$

APPENDIX B

To give a quantitative comparison between the SNR performance of A and H, we assume the input image to be a vertical step edge as shown in Fig. 3(a), without the loss of generality. This step edge is modeled as

$$s(x,y) = h_0 + h \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma_b^2}} e^{-\frac{\tau^2}{2\sigma_b^2}}\, d\tau. \tag{13}$$

After calculating all the derivatives, we have $s_y = s_{xy} = s_{yy} = 0$. Hence, $f_x = s_x + \Delta f_x$, $f_y = \Delta f_y$, $f_{xx} = s_{xx} + \Delta f_{xx}$, $f_{xy} = \Delta f_{xy}$, and $f_{yy} = \Delta f_{yy}$.

Case A: In the Neighborhood of the Edge: In the neighborhood of the edge, s_x and s_{xx} do not vanish. In this case, we may approximate A and H as

$$\mathbf{A} \approx \frac{1}{(1+f_x^2)^{\frac{3}{2}}} \begin{bmatrix} f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}$$

and

$$\mathbf{H} \approx \begin{bmatrix} f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}.$$

After calculating the eigenvalues of $\mathbf{A}$ and $\mathbf{H}$, we have k $\approx (f_{xx})/((1+(f_x)^2)^{3/2})$ and $\hat{k} \approx f_{xx}$.

Assume $s_{xx} \gg \Delta f_{xx}$ and $s_x \gg \Delta f_x$. Then the fluctuations of k and $\hat{k}$ can be approximated as

$$\Delta k \approx \frac{1}{(1+f_x^2)^{\frac{3}{2}}}\Delta f_{xx} - \frac{3 f_{xx} f_x}{(1+f_x^2)^{\frac{5}{2}}}\Delta f_x$$

and

$$\Delta\hat{k} \approx \Delta f_{xx}.$$

The variance of Δk can, thus, be calculated as

$$\mathrm{Var}(\Delta k) = \frac{1}{(1+f_x^2)^3}\mathrm{Var}\,(\Delta f_{xx}) - 2\frac{3 f_{xx} f_x}{(1+f_x^2)^4} E(\Delta f_{xx}\Delta f_x) + \frac{9 f_{xx}^2 f_x^2}{(1+f_x^2)^5}\mathrm{Var}(\Delta f_x). \tag{14}$$

It can be proved that $E(\Delta f_{xx}\Delta f_x) = 0$. Thus, we have

$$Var(\Delta k) = \frac{1}{(1+f_x^2)^3}\mathrm{Var}(\Delta f_{xx}) + \frac{9 f_{xx}^2 f_x^2}{(1+f_x^2)^5}\mathrm{Var}(\Delta f_x). \tag{15}$$

On the other hand, we can deduce that

$$\mathrm{Var}(\Delta\hat{k}) = Var(\Delta f_{xx}). \tag{16}$$

Case B: Over Smooth Regions: Around smooth regions, $f_{xx} \approx 0$ and $f_x \approx 0$. In that case, we have

$$\mathbf{H} \approx \begin{bmatrix} \Delta f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}$$

and

$$\mathbf{A} \approx \frac{1}{(1+(\Delta f_x)^2+(\Delta f_y)^2)^{3/2}} \begin{bmatrix} \Delta f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}.$$

If $\Delta f_x \ll 1$ and $\Delta f_y \ll 1$, then the curvature values estimated from H and A would be approximately the same.

Let $|\Delta\hat{k}_1|$ and $|\Delta\hat{k}_2|$ be the principal curvature estimated from $[\begin{smallmatrix} \Delta f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{smallmatrix}]$ and $|\Delta\hat{k}_1| \geq |\Delta\hat{k}_2|$. We have (17), shown at the top of the page. Define $\Delta q \equiv |\Delta f_{xx} + \Delta f_{yy}|$ and $\Delta r \equiv \sqrt{(\Delta f_{xx} + \Delta f_{yy}) - 4(\Delta f_{xx}\Delta f_{yy} - f_{xy}^2)}$. After a long deduction, it can be proved that the pdf (probability distribution function) of Δq follows

$$f_{\Delta q}(\Delta q) = \begin{cases} \frac{2}{\sqrt{2\pi\sigma_{\Delta p}^2}} e^{-\frac{\Delta q^2}{2\sigma_{\Delta p}^2}} & \Delta q \geq 0 \\ 0 & \Delta q < 0 \end{cases} \quad \text{with } \sigma_{\Delta p}^2 = \frac{\sigma_n^2}{2\pi\sigma_m^6} \tag{18}$$

while the pdf of Δr follows

$$f_{\Delta r}(\Delta r) = \begin{cases} \frac{\Delta r}{\sigma_s} e^{-\frac{(\Delta r)^2}{2\sigma_s^2}} & \Delta r \geq 0 \\ 0 & \Delta r < 0 \end{cases} \quad \text{with } \sigma_s^2 = \frac{\sigma_n^2}{4\pi\sigma_m^6}. \tag{19}$$

After combining (17) –(19), we finally have

$$E[|\Delta\hat{k}_1|] = E\left[\frac{\Delta q + \Delta r}{2}\right] = \frac{1}{2}\left(\frac{1}{2\sqrt{2}} + \frac{1}{\pi}\right)\frac{1}{\sigma_m^3}\sigma_n \tag{20}$$

and

$$\mathrm{Var}[|\Delta\hat{k}_1|] = \left(\frac{1}{\sqrt{2}\pi} + \frac{1}{\pi}\right)\frac{\sigma_n^2}{4\sigma_m^6} - \left\{\frac{1}{2}\left(\frac{1}{2\sqrt{2}} + \frac{1}{\pi}\right)\frac{1}{\sigma_m^3}\sigma_n\right\}^2 = \frac{\sigma_n^2}{4\sigma_m^6}\left\{\frac{1}{\pi} - \frac{1}{\pi^2} - \frac{1}{8}\right\}. \tag{21}$$

ACKNOWLEDGMENT

The authors would like to thank the associate editor, Dr. I. Selesnick, and the anonymous reviewers of this paper for their helpful comments.

REFERENCES

[1] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260–268, 1961.
[2] W. H. Tsai and S. S. Yu, "Attributed string matching with merging for shape recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 4, pp. 453–462, Apr. 1985.
[3] D. Vernon, "Two-dimensional object recognition using partial contours," *Image Vis. Comput.*, vol. 5, no. 1, pp. 21–27, 1987.
[4] H. Samet, "The quadtree and related hierarchical data structures," in *ACM Comput. Surv.*, vol. 16, 1984, pp. 187–260.
[5] V. Khanna, P. Gupta, and C. J. Hwang, "Maintenance of connected components in quadtree-based image representation," in *Proc. Int. Conf. Information Technology: Coding and Computing*, Las Vegas, NV, 2001, pp. 647–651.

[6] N. Ahmed, T. Natarjan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[7] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. New York: Springer-Verlag, 1975.

[8] E. L. Hall, *Computer Image Processing and Recognition*. New York: Academic, 1979.

[9] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact code," *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 337–345, Apr. 1983.

[10] A. Rosenfeld, *Multiresolution Image Processing and Analysis*. New York: Springer-Verlag, 1984.

[11] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 4, pp. 674–693, Jul. 1989.

[12] P. J. L. van Beek, "Edge-based image representation and coding," Ph.D. dissertation, Delft Univ. Technol., Delft, The Netherlands, 1995.

[13] A. Turiel and A. del Pozo, "Reconstructing images from their most singular fractal manifold," *IEEE Trans. Image Process.*, vol. 10, no. 4, pp. 345–350, Apr. 2002.

[14] S. Lakshmanan, A. K. Jain, and Y. Zhong, "Multi-resolution image representation using Markov random fields," in *IEEE Int. Conf. Image Processing*, vol. 1, 1994, pp. 855–860.

[15] J. Y. P. Wang, "Image representations using multiscale differential operators," *IEEE Trans. Image Process.*, vol. 8, no. 12, pp. 1757–1771, Dec. 1999.

[16] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 7, pp. 710–732, Jul. 1992.

[17] Y. Wang and S. K. Mitra, "Image representation using block pattern models and its image processing applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 321–336, Apr. 1993.

[18] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*. Upper Saddle River, NJ: Prentice-Hall, 1976.

[19] J. van de Weijer, L. J. van Vliet, P. W. Verbeek, and M. van Ginkel, "Curvature estimation in oriented patterns using curvilinear models applied to gradient vector fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 1035–1042, Sep. 2001.

[20] B. Rieger and L. J. van Vliet, "Curvature of n-dimensional space curves in grey-value images," *IEEE Trans. Image Process.*, vol. 11, no. 7, pp. 738–745, Jul. 2002.

[21] R. Bracho and A. C. Sanderson, "Segmentation of images based on intensity gradient information," in *Proc. IEEE Comput. Soc. Conf. Computer Vision and Pattern Recognition*, San Francisco, CA, 1985, pp. 341–347.

[22] P. Meer, J.-M. Jolion, and A. Rosenfeld, "A fast parallel algorithm for blind estimation of noise variance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 2, pp. 216–223, Feb. 1990.

[23] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," *Proc. Inst. Elect. Eng.*, vol. 146, no. 2, pp. 80–84, Apr. 1999.

[24] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Apr. 1986.

[25] Y. Itoh, "An edge-oriented progressive image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 2, pp. 135–142, Apr. 1996.

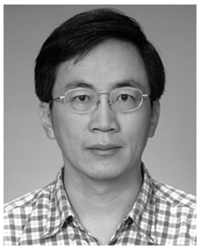
[26] R. S. Berns, *Billmeyer and Saltzman's Principles of Color Technology*. New York: Wiley, 2000.

[27] C. De Boor, "On calculation with B-splines," *J. Approx. Theory*, vol. 6, pp. 50–62, 1972.

[28] P. Lancaster and K. Salkauskas, *Curve and Surface Fitting: An Introduction*. New York: Academic, 1986.

[29] F. W. Meier, G. M. Schuster, and A. K. Katsaggelos, "A mathematical model for shape coding with B-splines," *Signal Process.: Image Commun.*, vol. 15, pp. 685–701, May 2000.

[30] H. H. Jong, Z. H. Wu, and S. J. Wang, "The extraction and linking of verge points in an image," in *Proce. IEEE Int. Conf. Image Processing*, Kobe, Japan, Oct. 1999, pp. 696–699.

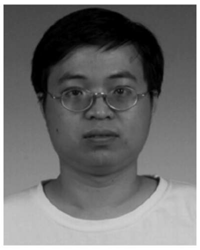


Sheng-Jyh Wang (M'95) received the B.S. degree in electronics engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., in 1984, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1990 and 1995, respectively.

He is currently an Associate Professor with the Department of Electronics Engineering, NCTU. His research interests are in the areas of image processing, video processing, and image analysis.

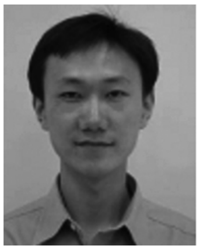
Lun-Chia Kuo was born in Taichung, Taiwan, R.O.C., in 1972. He received the B.S. and M.S. degrees from the National Taiwan University (NTU), Taipei, in 1994 and 1996, respectively. He is currently pursuing the Ph.D. degree in electronics engineering at the National Chiao-Tung University, Hsinchu, Taiwan.

He joined VICTOR Taichung Machinery in 1996 and the Computer and Communication Research Laboratories, ITRI, in 1999. His research interests include image representation and video communication.



Hsin-Haw Jong was born in Taichung, Taiwan, R.O.C., in 1974. He received the B.S. degree and the M.S. in electronics engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1995 and 1998, respectively.

In 1998, he joined the Silicon Intergrated System Corporation as a Design Engineer for GPU chips. He is currently the Section Manager of the GPU design of the eXtreme Graphics Innovation Corporation.



Zong-Han Wu was born in Taipei, Taiwan, R.O.C., in 1973. He received the M.S. degree in electronics engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1998.

His research interests include video/image compression and MPEG4/H.264 CODEC IC design.