Chapter 5

Image Features Detection and Image Surface Manipulation

In Chapter 3, we present verge-point extraction and image surface reconstruction. In Chapter 4, we demonstrate the diversified data structures of verge point representation. Now, in the first half of this chapter, we demonstrate that it is feasible to represent image features, such as points, edges, and corners, using verge point representation. We also proposed a multi-scale method to extract homogeneous regions on image surfaces. Then, in the second half of this chapter, different types of image surface manipulations based on verge point representation are presented. These manipulations can effectively achieve different visual functionalities, like image enhancement, color editing, and shape editing.

5.1 Image Features Detection

Feature detection plays an important role in image understanding, because a lot of information could be revealed by merely preserving these feature elements. Image features generally include points, ridges and valleys, step edges, and corners. Fig. 5.1(a) shows an original image, and the detected edge features are shown in Fig. 5.1(b). We can see form Fig. 5.1(b) that it provides a primitive sketch of Fig. 5.1(a). So far, a lot of research works concerning feature detection have already been conducted [59]-[67].

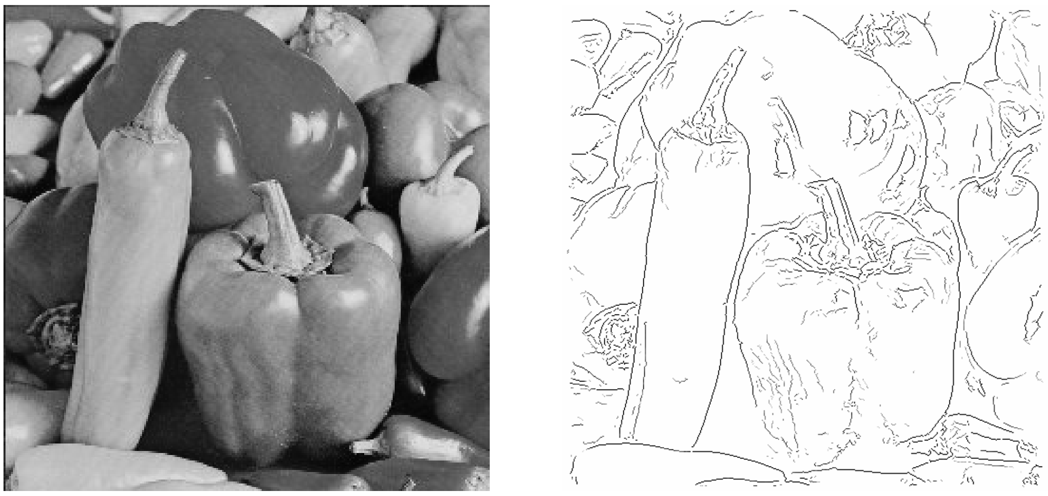
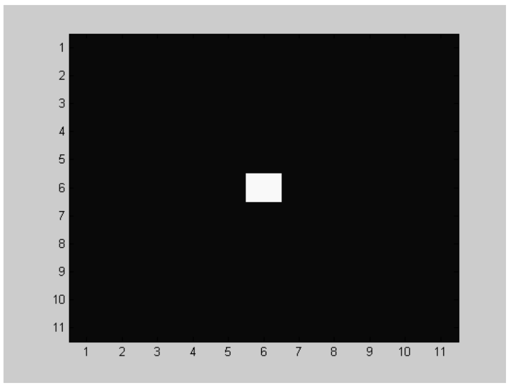


Fig. 5.1 (a) Original image. (b) Detected edges

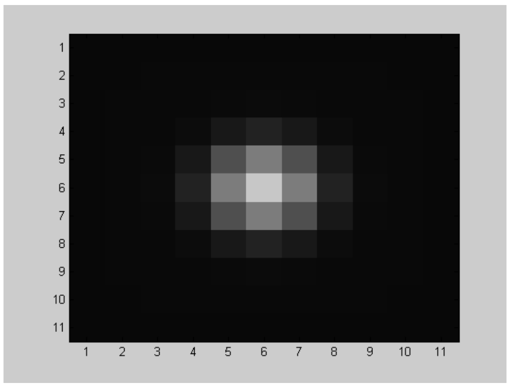
Here, we'll use the sign and magnitude of eigenvalues obtained from the Hessian tensor to detect various types of image features. The procedures of these image feature extractions are to be presented in the following sections.

5.1.1 Point Detection

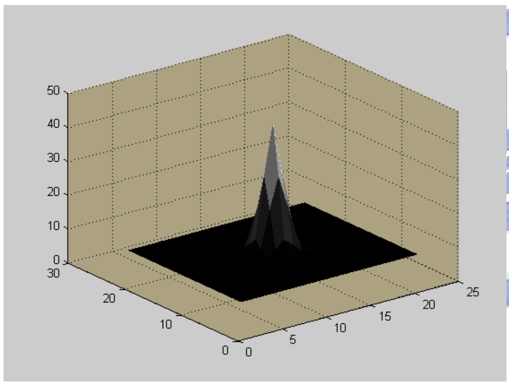
To detect points, we first smooth an image using Gaussian operator. Fig. 5.2(a) and (b) show a point image and the smoothed point image, where the scale parameter σ_m is set as 1. Then, Hessian tensor is applied to detecting verge points. For a point with higher intensity than its neighbors, the geometric shape is composed of a concave elliptic surface at the center and surrounded by convex hyperbolic surfaces. Therefore, the magnitudes of both eigenvalues, $k_1(x,y)$ and $k_2(x,y)$, are high at the center point and its neighbors. The signs of $k_1(x,y)$ and $k_2(x,y)$ at the center point are all negative, while the signs of the neighbors are one positive and one negative. In addition, if the point is isotropic, $k_1(x,y)$ equals to $k_2(x,y)$ at the center point. The obtained eigenvalues using Hessian H applied on Fig. 5.2(b) are shown in Fig. 5.2(d) and (e) respectively, where positive values are indicated in red while negative values are indicated in green. The point in yellow indicates that $k_1(x,y)$ equals $k_2(x,y)$. Based on this surface type, we can detect points by checking a structure with a concave elliptic point surrounded by concave hyperbolic points.



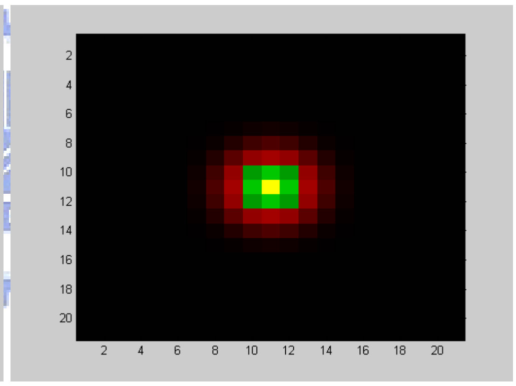
(a)



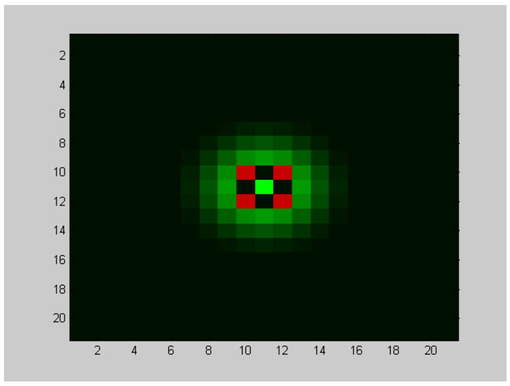
(b)



(c)



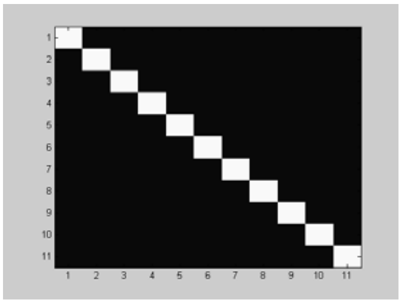
(d)



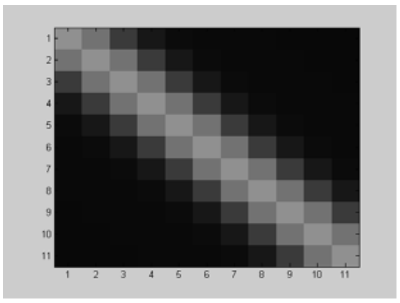
(e)

Fig. 5.2. Point detection (a) Original point image. (b) Point image after Gaussian smoothing. (c) Point image surface. (d) Larger eigenvalues, k1. (e) Smaller eigenvalues, k2.

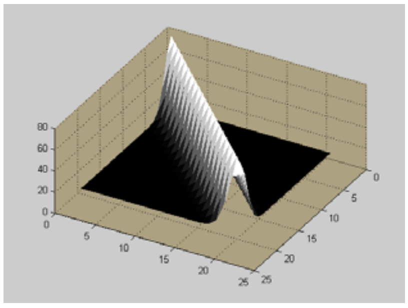
5.1.2 Line Edge Detection



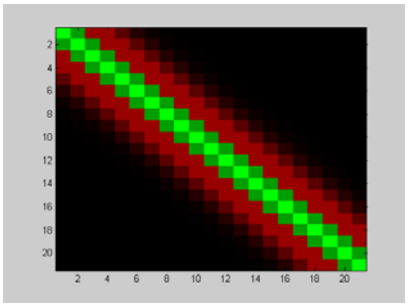
(a)



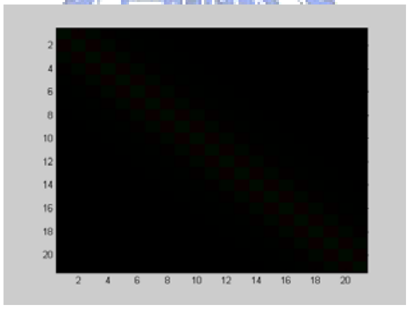
(b)



(c)



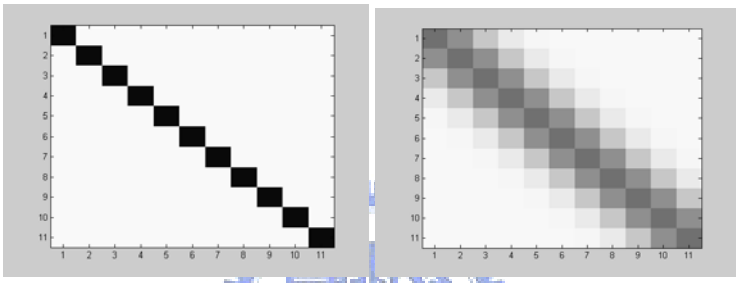
(d)



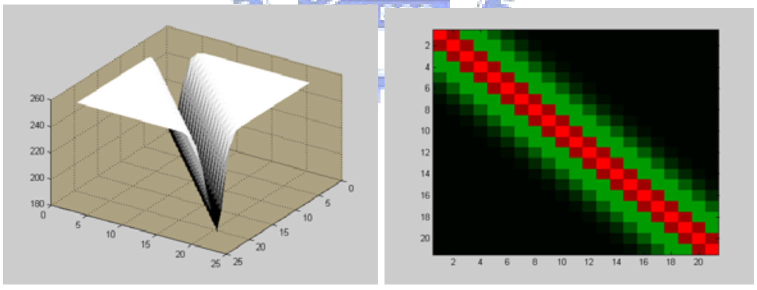
(e)

Fig. 5.3 Ridge Detection (a) Original ridge image. (b) Ridge image after Gaussian smoothing. (c) Ridge image surface. (d) Larger eigenvalues, k1. (e) Smaller eigenvalues, k2.

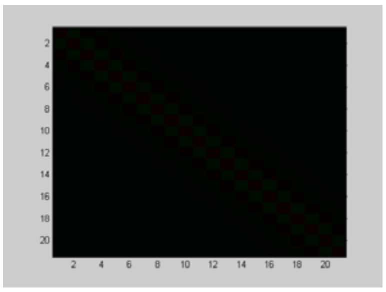
Line edges could be classified into ridges and valleys according to the intensity differences with respect to the surrounding background. Fig. 5.3(a) and (b) show a ridge image and the smoothed ridge image, where the scale parameter σ_m is set as 1. Then, the Hessian tensor is applied to detecting verge points. The image surface of a ridge is shown in Fig. 5.3(c). For a ridge, the geometric shape is featured by a segment of concave parabolic (negative) points clamped by two segments of convex parabolic (positive) points. Therefore, for a ridge, the magnitudes of the larger $k_1(x, y)$ are local extremes, while the magnitudes of $k_2(x, y)$ are near zero. The obtained eigenvalues, $k_1(x, y)$ and $k_2(x, y)$, using Hessian tensor are shown in Fig. 5.3(d) and (e) respectively, where positive values are indicated in red while negative values are indicated in green. Based on the surface types, we can detect ridges by detecting a structure with a concave parabolic point clamped by two concave parabolic points. Similarly, we can check a structure with a segment of convex parabolic points clamped by two segments of concave parabolic points to detect valleys (see Fig. 5.4).



(a) (b)



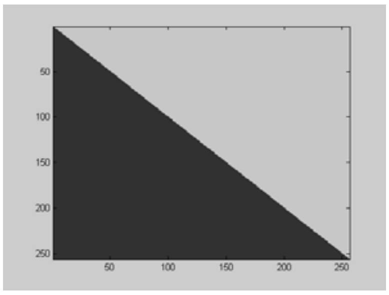
(c) (d)



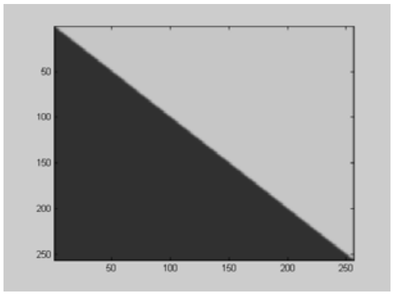
(e)

Fig. 5.4 Valley detection. (a) Original valley image. (b) Valley image after Gaussian smoothing. (c) Valley image surface. (d) Larger eigenvalues, k1. (e) Smaller eigenvalues, k2.

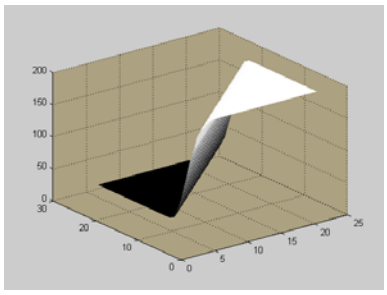
5.1.3 Step Edge Detection



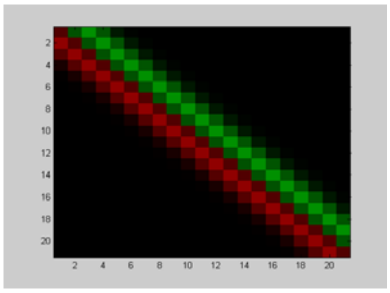
(a)



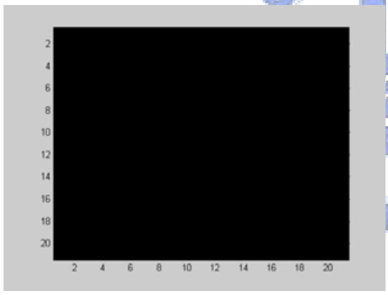
(b)



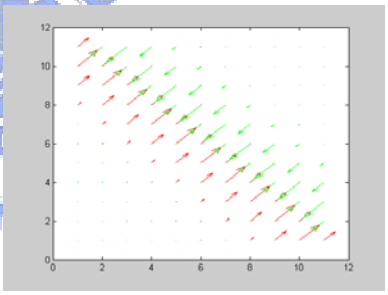
(c)



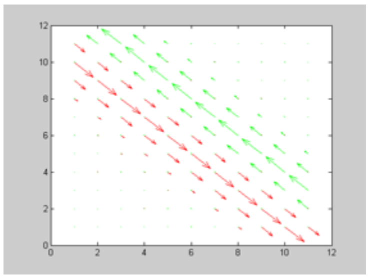
(d)



(e)



(f)

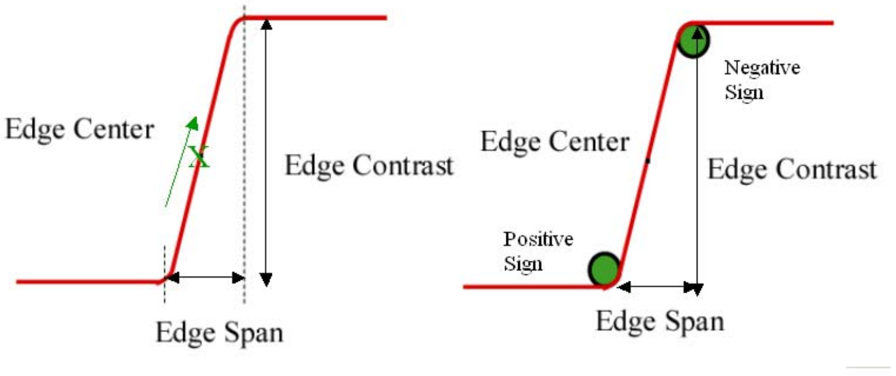


(g)

Fig. 5.5. Step edge detection. (a) Original step edge. (b) Step edge after Gaussian smoothing. (c) Step edge image surface. (d) Larger eigenvalues, k1. (e) Smaller eigenvalues, k2. (f) Eigenvector, $\Lambda_1(x,y)$. (g) Eigenvector, $\Lambda_2(x,y)$.

To detect step edges, we first analyze the image surface of step edges. Fig. 5.5(a) and (b) show a step edge and the smoothed step edge, where the scale parameter σ_m is set as 1. Similarly, Hessian tensor is applied to detecting verge points. The image surface of a step edge is shown in Fig. 5.5(c). For a step edge, the geometric shape of image surface is featured by a segment of concave parabolic (negative) points and a corresponding segment of convex parabolic (positive) points. Therefore, for a step edge, the magnitudes of larger $k_1(x, y)$ are local extremes, while the magnitudes of $k_2(x, y)$ are near zero. The obtained eigenvalues, $k_1(x, y)$ and $k_2(x, y)$, using Hessian tensor, are shown in Fig. 5.5(d) and (e) respectively, where positive values are indicated in red and negative values are indicated in green. We can detect step edge by detecting a structure with two parabolic segments of points with different sign. The eigenvector, $\Lambda_1(x, y)$ and $\Lambda_2(x, y)$, which can be used for verge curve linking, are shown in Fig. 5.5(f) and (g).

To detect edges, most edge operators aim at detecting the edge center and measuring the edge strength based on magnitude of derivatives (see Fig. 5.6(a), where the edge center is indicated by the symbol “x” and the edge strength is indicated by an arrowed line in green). Although these operators may work well for specific applications, some crucial information may have been lost if we only preserve the edge center and the magnitude of derivatives. Since there are infinite sets of contrast-span pair with the same slope, it would be difficult to estimate the edge contrast and edge span by merely using the magnitude of derivatives.



(a) (b)

Fig. 5.6. (a) Illustration of edge center, edge span, and edge contrast. (b) Concept of using verge points to detect edges.

In Fig. 5.6(b), we show the verge point representation. Based on verge point representation, the edge center, edge contrast, and edge span could be determined easily. With edge span and edge contrast, edge strength can be expressed either by edge contrast or by strain ε as indicated in Chapter 2. As indicated in [61][66], in many cases, edge contrast may well represent the perception of edge strength. Furthermore, we can enhance or soften an edge by altering the edge span. In Fig. 5.7, the edge formed by P6 and P7 could be removed because the contrast between the two points is relatively small, although the slope in that section is high. This may be helpful in combating noise. In the section formed by P9, P10, P11, and P12, P9-P11 are actually in the same concave trend. In this case, both P10 and P11 may be removed to form a single edge with larger contrast.

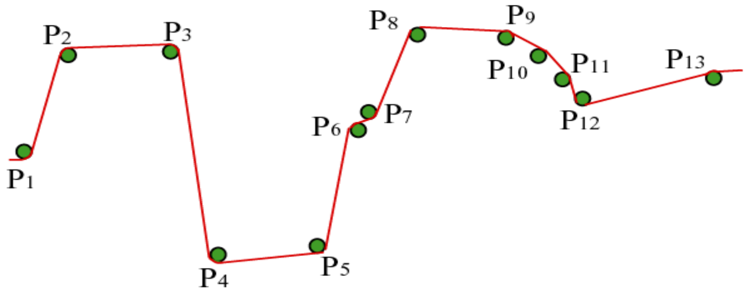
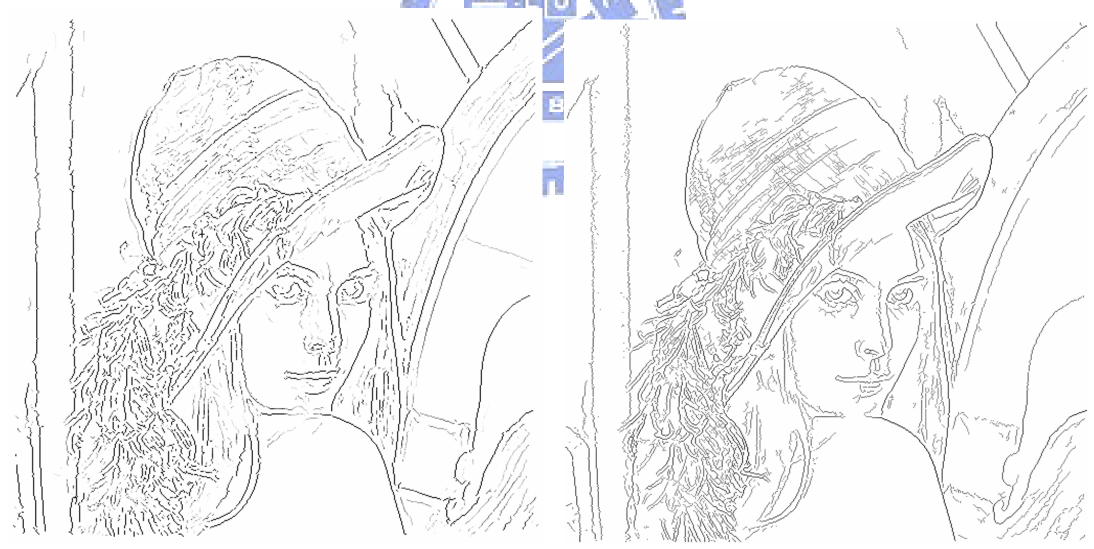


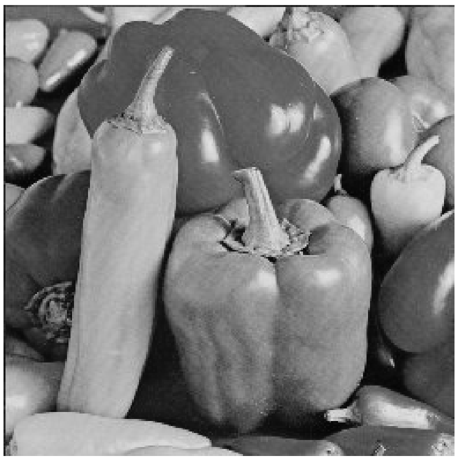
Fig. 5.7 Edge strength determination.

Here, we demonstrate edge detection using the intensity slope (strain ε). For a verge point P with sign S on an image, we search along 8 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 pre-selected threshold, an edge is detected and the middle point of that pair is marked as an edge point. Fig. 5.9(a) shows the simulation result of this edge detector, with $T_k = 2$. As a comparison, edges detected by the well-known Canny Operator are shown in Fig. 5.9(b). Fig. 5.9(b) is 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 0.043 and 0.087, respectively. It can be seen that the performance of edge detection using Hessian **H** is comparable to that of Canny Operator. Another example for edge detection is shown in Fig. 5.10.

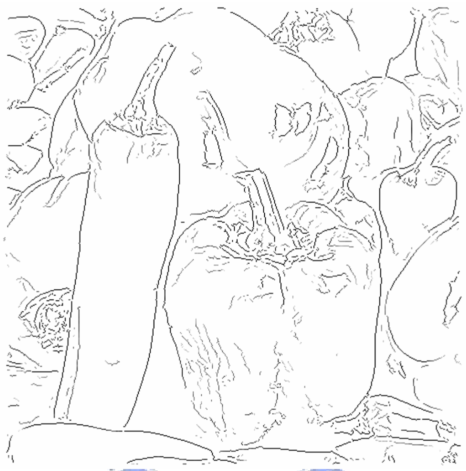


(a) (b)

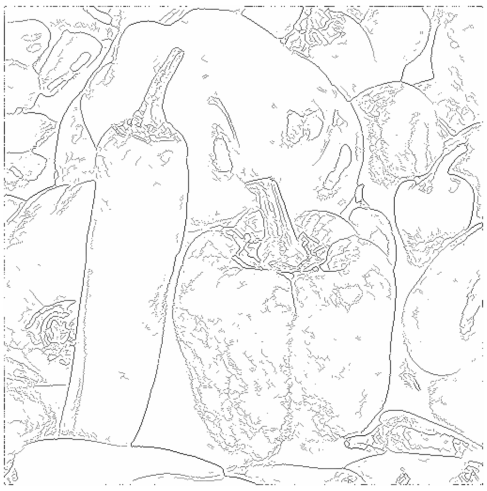
Fig. 5.9. (a) Edge detection based on verge points. (b) Edge detection based on the Canny Operator.



(a)



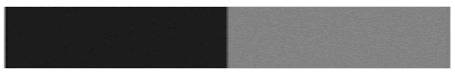
(b)



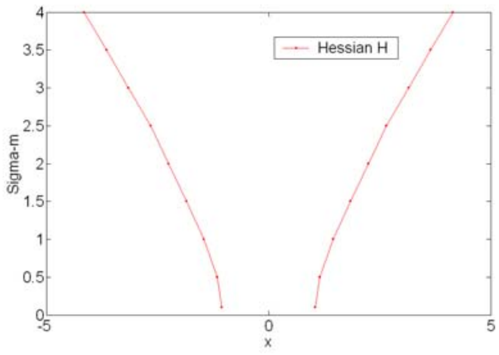
(c)

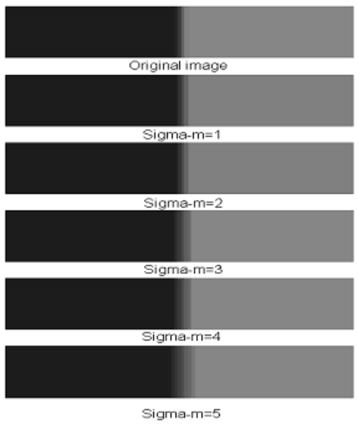
Fig. 5.10. (a) Original image. (b) Detected edges using the proposed method. (c) Detected edges using the Canny Operator.

Here, we also investigate the performance of Hessian **H** in edge detection under different scales. In Fig. 5.11(a), we show a step edge image. The edge center locates at x=0. In Fig. 5.11(b), we show the positions of the detected verge points for the step edge image shown in Fig. 5.11(a). The vertical axis represents the value of a scale parameter σ_m of the Gaussian smooth function. The horizontal axis represents the x axis. The reconstructed edge images using detected edge center, edge contrast, and edge span are shown in Fig. 5.11(c). It can be seen that the Hessian operator can detect the edge contrast and edge span correctly as long as σ_m doesn't grow too large. The detection of edge center is less influenced by the selection of σ_m, since we use the midpoint of the high-curvature pair to detect edge center.



(a)

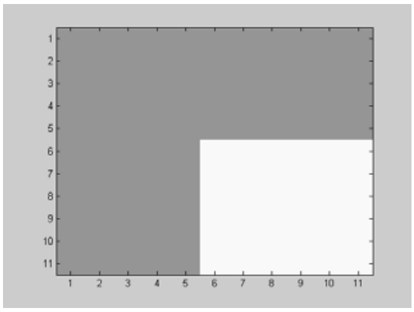




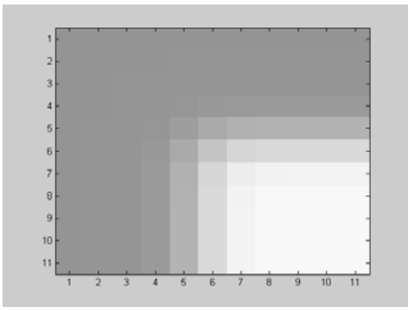
(b) (c)

Fig. 5.11. (a) Original edge image. (b) Positions of the detected high-curvature pair under different scales. (c) Reconstructed edge images using the detected edge center, edge contrast, and edge span for different scales.

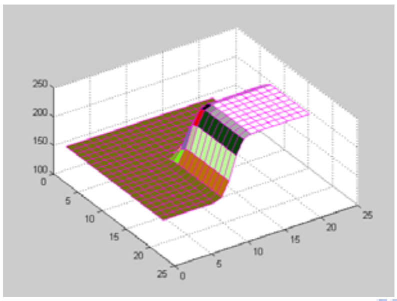
5.1.4 Corner Detection



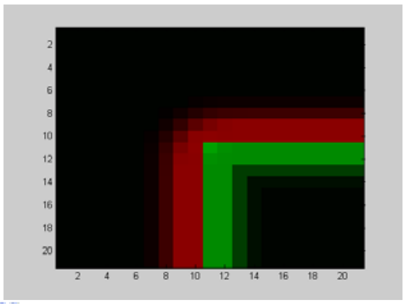
(a)



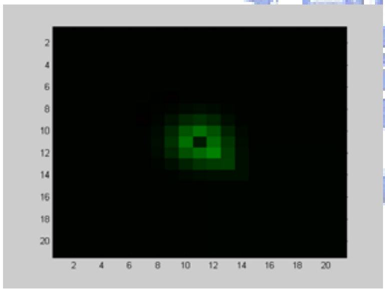
(b)



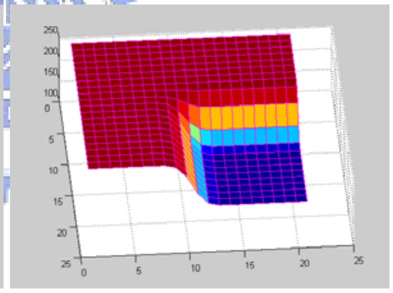
(c)



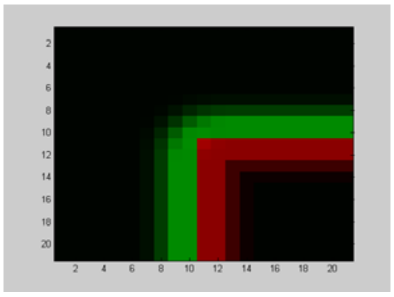
(d)



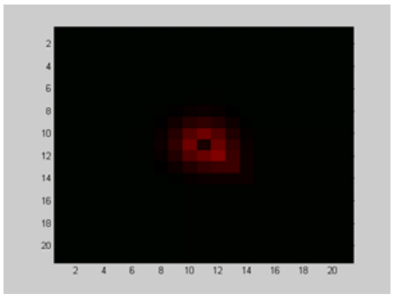
(e)



(f)



(g)



(h)

Fig. 5.12 Corner detection (a) A Corner image. (b) Corner image after Gaussian smoothing. (c) Image surface around corners. (d) Larger eigenvalues, k1. (e) Smaller eigenvalues, k2. (f) Another type of corner. (g) Larger eigenvalues, k1. (h) Smaller eigenvalues, k2.

To detect corners, we first analyze the image surface around corners. Fig. 5.12(a) and (b) show a corner and the smoothed corner image, where the scale parameter σ_m is set as 1. The image surface of a corner is shown in Fig. 5.12(c). For a corner shown in Fig. 5.12 (a), the geometric shape of image surface is featured by a pair, concave elliptic surface and a convex hyperbolic surface. For these corner pair, the magnitudes of k1 and k2 are both high. The obtained eigenvalues, $k_1(x,y)$ and $k_2(x,y)$, are shown in Fig. 5.12(d) and (e) respectively, where positive values are indicated in red and negative values are indicated in green. We use local extreme of the products of $k_1(x,y)$ and $k_2(x,y)$ (Gaussian curvature) to detect corner pair. Then, the midpoint of a corner pair is selected as the location of a corner. Another type of corner opposite to Fig. 5.12(a) in intensity is shown in Fig. 5.12(f), and the detected eigenvalues are shown in Fig. 5. 12(g) and (h).

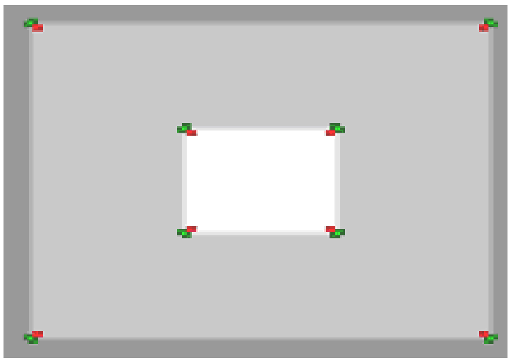
In Fig. 5.13(a), we show the simulation result for a synthetic image. Points with positive $k_1(x,y) \times k_2(x,y)$ are indicated in red, while points with negative product are indicated in green. We mark these local extremes of $k_1(x,y) \times k_2(x,y)$ as the corner pairs. Then, the mid-point of a corner pair is chosen to be a corner point. A simulation result for corner detection is shown in Fig. 5.13(b). As a comparison, the corner detection performed by the popular Harris Operator (indicated as **R**) [65] is shown in Fig. 5.13(c). The Harris operator **R** is defined as

$$R=\det(\mathbf{C}_{str})-k_{str} \times (\mathrm{trace}(\mathbf{C}_{str}))^2, \tag{5.1}$$

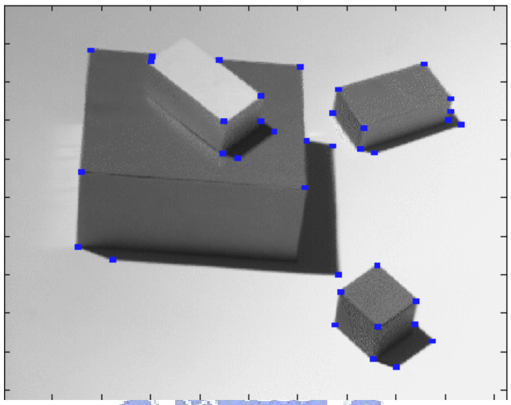
where k_{str} is a thresholding parameter and C is a local structure matrix. The matrix C is defined as

$$C=\begin{bmatrix} \left(\frac{\partial f(x,y)}{\partial x}\right)^2 & \frac{\partial f(x,y)}{\partial x}\times\frac{\partial f(x,y)}{\partial y} \\ \frac{\partial f(x,y)}{\partial x}\times\frac{\partial f(x,y)}{\partial y} & \left(\frac{\partial f(x,y)}{\partial y}\right)^2 \end{bmatrix}. \tag{5.1}$$

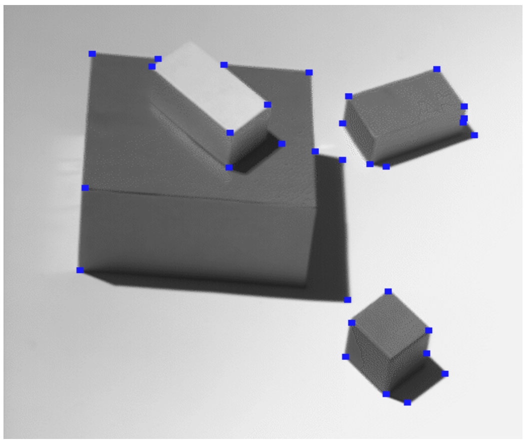
We can see that the performance of corner detection using Hessian **H** is comparable to that of Harris Operator.



(a)



(b)



(c)

Fig. 5.13. (a) Concept of corner detection. (b) Detected corners using Hessian **H**. (c) Detected corners using Harris operator.

In addition, since the intensity values of the two sides of the corner are identified, the proposed operator may distinguish two corners with the same gradient but with different intensity values or different properties. In Fig. 5.14, we use the signs of $k_1(x,y)\times k_2(x,y)$ around the two sides of corners and the values of skin color to distinguish fingertips from the corners between two fingers.

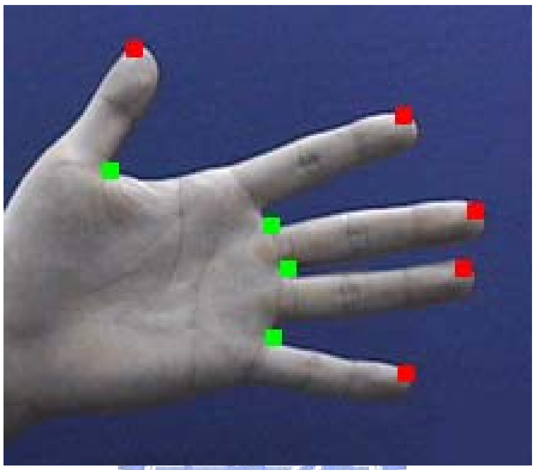


Fig. 5.14. Fingertips detection.

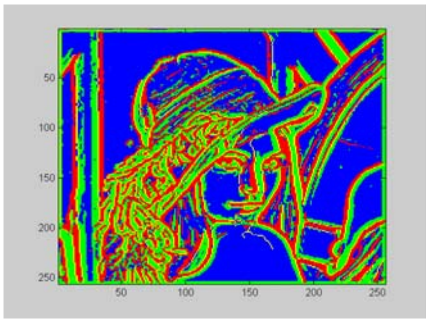
5.1.5 Homogeneous Region Detection

In this thesis, we propose a multi-scale method for region detection. For an image **IM**, assume the sets of edge, high curvature surfaces, and smooth regions are indicated by **DE**, **HP**, and **RG**, respectively. Clearly,

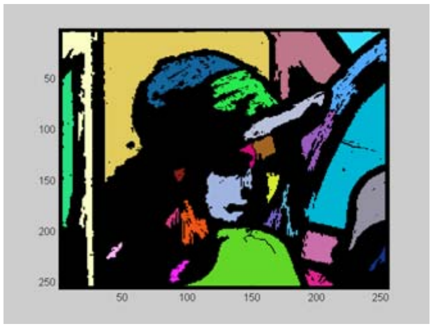
$$DE \cup HP \cup RG = IM \tag{5.2}$$

As indicated previously, edges can be determined by the aid of verge points. High curvature surfaces can be identified by inspecting the eigenvalues of Hessian tensor. Intuitively, smooth regions, which map to planer regions in image surface, can be obtained by connecting the points whose $k_1(x,y)$ and $k_2(x,y)$ are small. However, due to noise interference, directly pixel linking may mistakenly connect two different regions. In addition, due to the size of the applied operator, smaller region may be neglected. Fig. 5.15(a) shows an original image. Fig. 5.15(b) shows the surface feature map, where the scale parameter σ_m is set as 2. Positive and negative high curvature points are indicated in red and green, and edges are indicated in yellow. The applied curvature threshold is determined using (2.33). We can see in Fig. 5. 15(b) that small regions are neglected due to the operator size. Therefore, in this thesis, we proposed a 3-level multi-scale method to determine smooth regions. The scale parameters are selected as maxScale, maxScale/2, and maxScale /4 for each level, where maxScale is set as 2. The allowed minimum region sizes for each level are set as 4*minSize, 2*minSize, and minSize, where minSize is set as 5 in the simulation. For each level, to avoid mistakenly region linking caused by leakage, the histograms of regions in **SR1**, **SR2**, and **SR3**, are check if there exist multiple peaks. If multiple peaks exist in a region, we half the threshold in the region to separate that region into multiple sub-regions. Then, we perform inter-level region test to eliminate the overlapped region in lower level. We use the inter-level mapping between SR1 (higher level) and SR2 (lower level) as an example. For a region sr(t) detected in SR2, we check if it overlaps with regions detected in SR1. If overlap occurs, we consider sr(t) to be covered by at least one existing region in SR1. Therefore, we remove the label of sr(t), and set the domain of sr(t) as “occupied”. If no overlap exists, we consider sr(t) as a newly detected region. Similar rules are applied between SR2 and SR3. The raw detected regions for the three levels, **SR1**, **SR2**, and **SR3**, using the above parameters, are shown in Fig. 5.15(c)-(e). The detected smooth regions after inter-level region mapping are shown in Fig. 5.15(f). Then, we preserve edges and remove the high curvature surfaces to obtain object segments. A simple region growing method is applied to filling the unlabelled pixels originally belonging to high-curvature surfaces. The final detected smooth regions are shown in Fig. 5. 15(g). Another example is shown in Fig. 5.16.

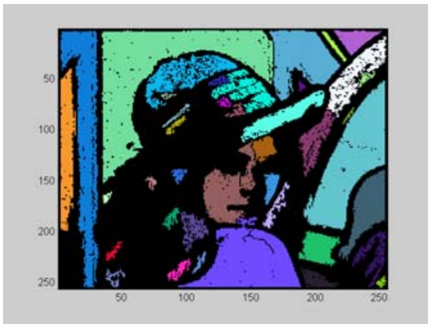
(a)



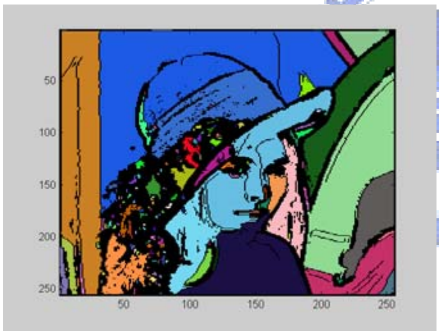
(b)



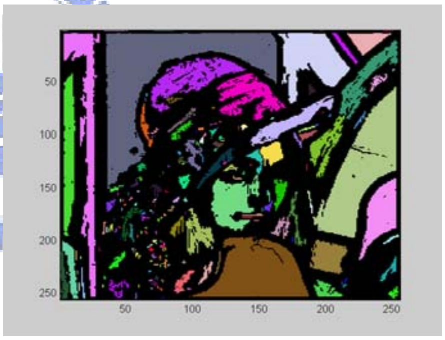
(c)



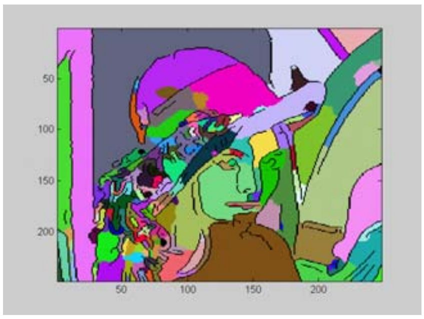
(d)



(e)

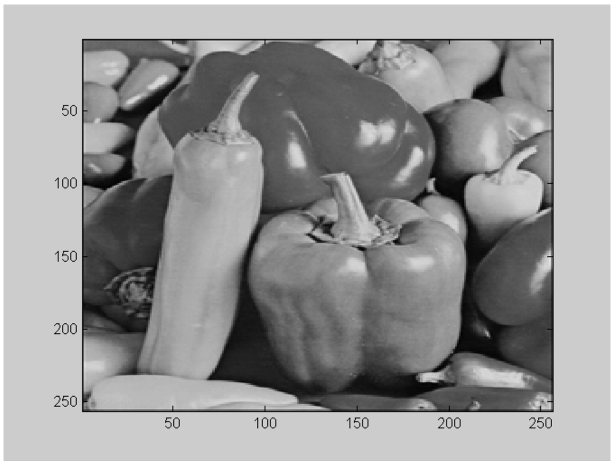


(f)

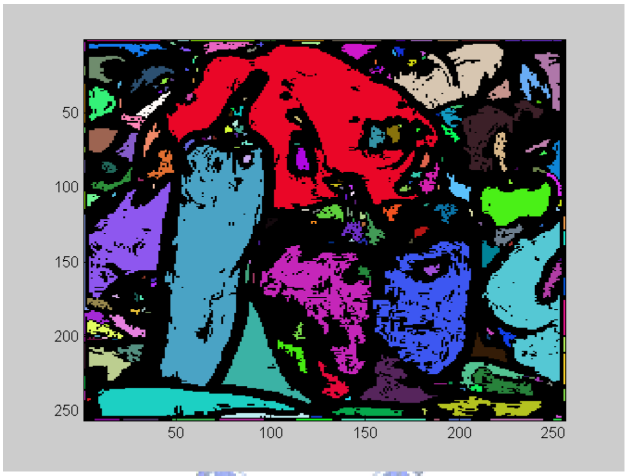


(g)

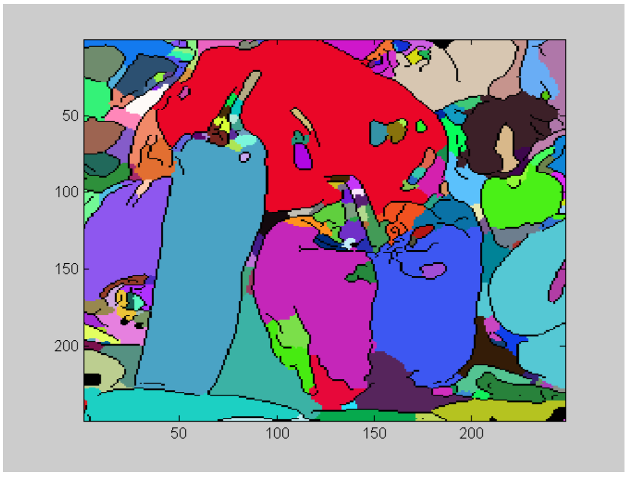
Fig. 5.15. (a) Original images. (b) Feature map. (c) First level region map. (d) Second level region map. (e) Third level region map. (f) Multi-level region map. (g) Final region segmentation.



(a)



(b)



(c)

Fig. 5.16. (a) Original images. (b) Multi-level region map. (c) Final region segmentation.

5.2 Image Surface Manipulation

In this section, different types of image surface manipulations based on verge point representation are presented. These manipulations can effectively achieve different visual functionalities, like image enhancement, color editing, and shape editing.

It is well-known that a benefit of digital imaging lies on the property that the content of an image can be edited into a more “suitable” form based on various task requirements. To edit or enhance an image, several methods have been proposed [68]-[73]. In [68], a deformable contour model is proposed to match the object contour by energy minimization. In [69], a sketch editing framework is proposed to decompose handwritten figures into visual structures for ease of manipulation. In [70], an interactive tool named “Intelligent Scissors” is developed for image segmentation and composition. In the proposed method, an optimal path between two points assigned by users is calculated suing dynamic programming. In [71], to enhance an image, a new adaptive contrast enhancement algorithm is proposed to reduce noise over-enhancement and ringing artifacts. In [72], an adaptive unshap masing is proposed to enahnce an image based on the observation that areas with medium details may need to be enahnced more than those areas with very low or high dynamics. In [73], an edge-based method is proposed to edit an image. An image is decomposed into modeled edges with parameters, such as edge center, edge contrast, and scales.

The term “suitable”, however, is quite subjective. It is difficult to have a single and universal criterion for “suitability” that holds for various types of applications. For instance, for a blurred image, we might aim at the enhancement of sharpness; while for an image lack of observable details, we might aim at the enhancement of contrast. Moreover, if we are interested in specific features, like the intensity or shape of an object, only a portion of the image needs to be amended while the rest part of the image should remain unchanged.

Since it is difficult to anticipate in advance how an image should be modified, a flexible architecture for image editing is needed. In this section, based on verge point representation, we proposed a flexible architecture to manipulate image surfaces for image editing. The architecture features in three aspects. First, an image is decomposed into the skeleton component and the residual component. By tuning different weightings of these two components, different effects could be manifested. Second, unlike the pixel-based scheme used in traditional image editing software, the color or grayscale can be edited globally or locally. Third, the B-spline approximation allows users to change the shapes easily.

5.2.1 Problem Formulation

In (3.5), an image F(D) is expressed as $F(D) = \sum\sum_{(\mu,\eta)\in D} f(\mu,\eta)\delta(x-\mu, y-\eta)$, where δ indicates the 2-D Dirac impulse function, and $D=\{(x,y) \mid 1\le x\le x_{max}, 1\le y\le y_{max}\}$. To develop a flexible architecture for image editing, we divide the image F(D) into a skeleton component and a residual component; that is

$$F(D) = F_{skeleton}(D) + F_{residual}(D). \qquad (5.3)$$

We first illustrate the concept of the proposed approach using a one-dimensional image profile. The skeleton component can be obtained by finding the verge points in the profile first, and then apply linear interpolation over these verge points. This process can be done using the method proposed in Chapter 3. As shown in Fig. 5.17(a), the interpolated profile does catch the outline of the original profile. Hence, we name the interpolated profile as skeleton component. After having obtained the skeleton component, the residual component is defined as the difference between the original profile and the skeleton component. The residual component represents the properties of the profile, such as coarseness or smoothness. This decomposition allows us to control the skeleton component and the residual component separately. For example, we may intend to increase the contrast between a human face and the background without enhancing the texture of the face. This goal can be achieved by moving up and down the verge points to stretch the contrast of the skeleton component, while keeping the residual component unchanged (see Fig. 5.17(b)). Similarly, a softened image can be obtained by reducing the contrast of the skeleton component. The image sharpness can be improved by moving skeleton pairs (two verge points with different signs) toward each other. In addition, by adjusting the amount of the residual component, different visual effects, such as "smoother" or "coarser", can be obtained. Furthermore, since verge points divide the profile into several segments, changing the position of a verge point would only influence adjacent segments. This allows local controls of the image contents.

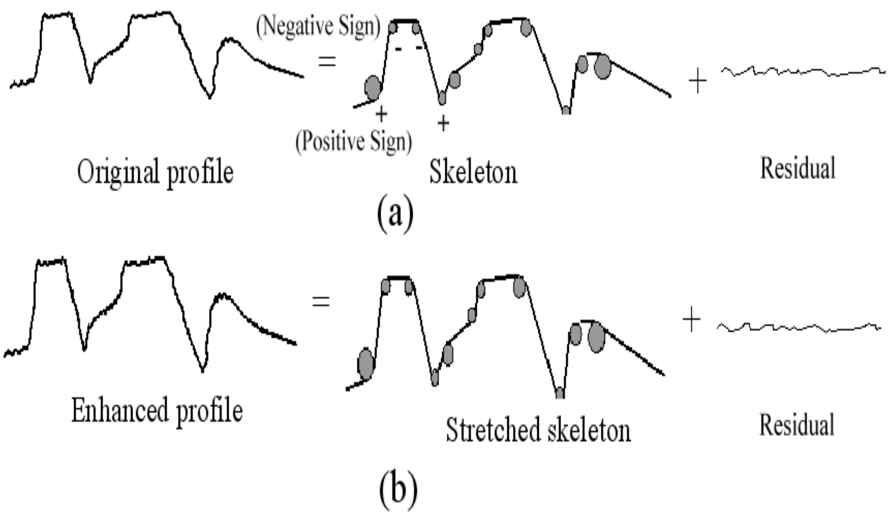


Fig. 5.17. Concept of image decomposition.

The curvature threshold T_k used in the extraction of verge points can be used to determine how many details should be captured in the skeleton component. Fig. 5.18(b) shows the extracted verge points of the image in Fig. 5.18(a), where positive verge points are indicated in red while negative verge points are indicated in green.

Assume the set of the positions of extracted verge points is expressed as Ω; that is,

$$\Omega=\{(x_{sp},y_{sp})\mid (x_{sp},y_{sp})\in \text{verge points}\}. \qquad (5.4)$$

Once these verge points are given, the skeleton component $F_{skeleton}(D)$ can be generated using interpolation techniques as described in Chapter 2, that is,

$$F_{skeleton}(\Omega;D) \equiv \Gamma\{\sum\sum_{(\mu,\eta)\in\Omega} f(\mu,\eta)\delta(x-\mu,y-\eta);D\}\ , \qquad (5.5)$$

where Γ{} denotes the interpolation process employed over the set of verge points to generate an image with domain D. Fig. 5.18(c) shows the skeleton component interpolated using the verge points in Fig. 5.18(b). Fig. 5.18(d) presents the residual component. To adjust image contrast, we adopt two parameters, Δ_{sp} and α, in the following equation:

$$\tilde{F}(D)=\Gamma\{\sum\sum_{(\mu,\eta)\in\Omega}[f(\mu,\eta)+\Delta_{sp}(\mu,\eta)]\delta(x-\mu,y-\eta);D\}\ +\alpha\times F_{residual}(D). \qquad (5.5)$$

In (5.5), $\tilde{F}(D)$ denotes the processed image. $\Delta_{sp}(\mu,\eta)$ indicates the amount of adjustment over the intensity value of a verge point at (μ,η). The parameter α indicates the proportion of residual component that is to be added into $\tilde{F}(D)$.

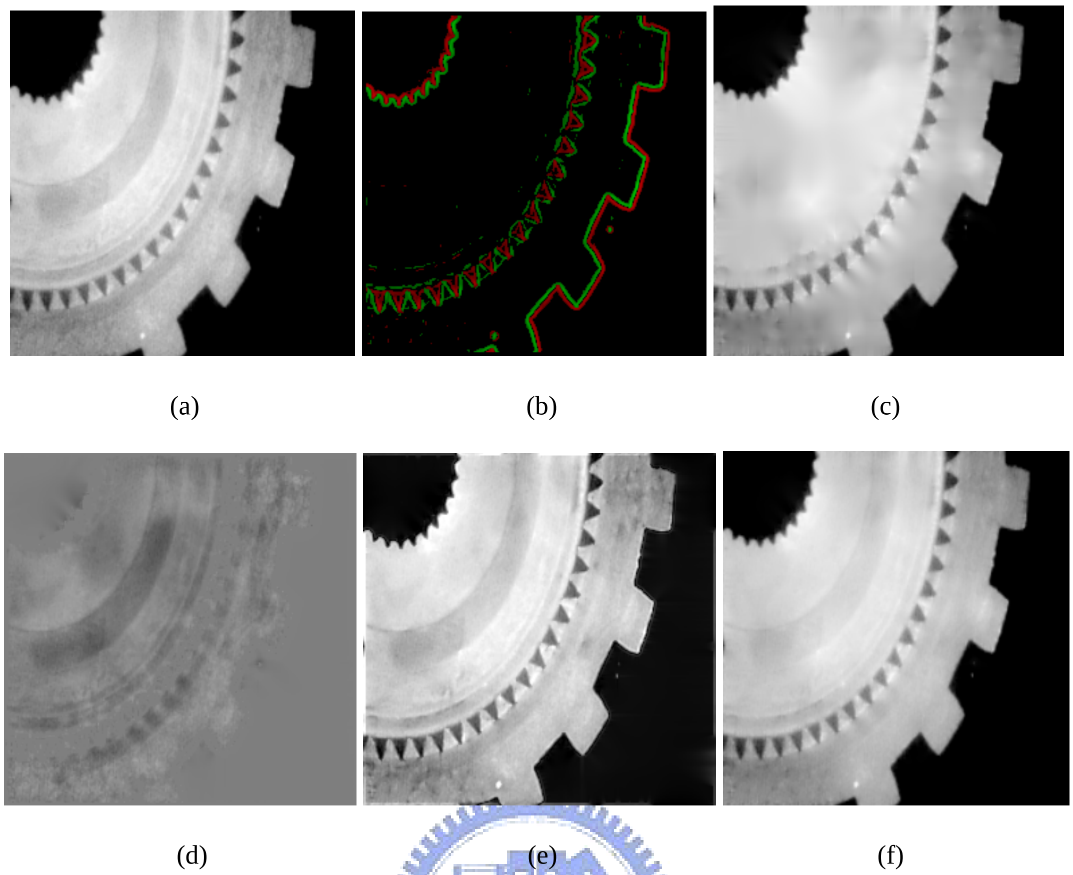
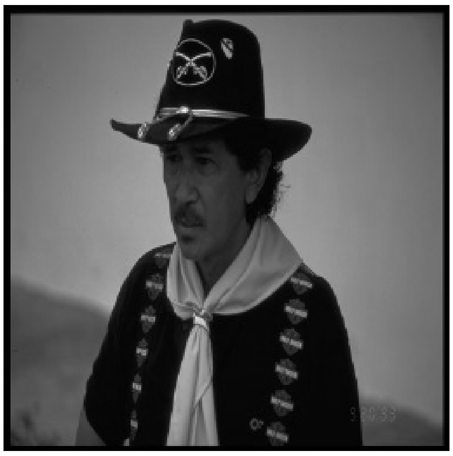


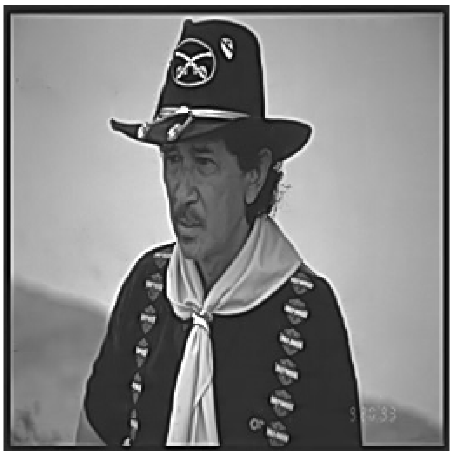
Fig. 5.18. (a) Original image. (b) Extracted verge points. (c) Skeleton component with threshold T_k = 6. (d) Residual component of Fig. 4.13(c). (e)Enhancing skeleton component only. (f) Changing residual component only; α is set as 0.5.

5.2.2 Editing Images in Intensity Domain

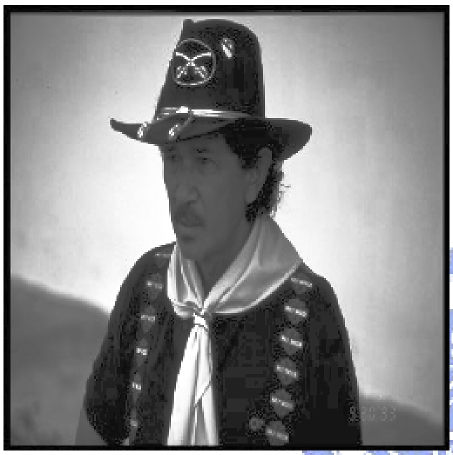
The sign of Δ_{sp} is determined by the sign of principal curvature at (μ,η), and the magnitude of Δ_{sp} can be either a constant or a function of $|k_1(\mu,\eta)|$. By choosing different α's and Δ_{sp}'s, different effects are manifested in the adjusted image $\tilde{F}(D)$. For example, by setting α=1 and choosing a Δ_{sp} function to stretch verge points along the intensity axis, an image with enhanced skeleton and unchanged residual is obtained. As shown in Fig. 5.18(e), $\Delta_{sp}(\mu,\eta)$ is set as $2*k_1(\mu,\eta)$ and α = 1. On the other hand, if we choose $\Delta_{sp}(\mu,\eta)$ =0 for every verge point and set $\alpha<1$, an image with unchanged skeleton component but lighter texture (smoother surface) is obtained. By setting $\alpha>1$, on the contrary, produces an image with heavier residual component (coarser surface). In Fig. 5.18(f), $\Delta_{sp}(\mu,\eta)$ is set to be 0 and $\alpha = 0.5$. It can be seen that different types of contrast adjustment can be achieved under this proposed scheme.

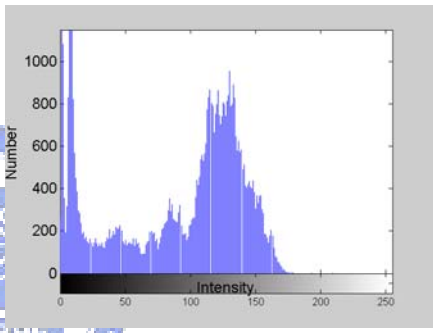
Next, we demonstrate how to achieve global contrast enhancement based on this decomposition scheme. To efficiently utilize the available intensity range, the mean of an image can be shifted to the middle point of the allowed range first. Then, the aforementioned contrast stretching process can be applied. Fig. 5.19(a) shows an original image, Fig. 5.19(b) shows the enhanced image using the proposed method, and Fig. 5.19(c) shows the enhanced image using histogram equalization. The corresponding histograms are shown in Fig. 5.19(d)-(f). Compared Fig. 5.19(b) with Fig. 5.19(c), the proposed method produced a more natural and clear image. In addition, with histogram equalization, the histogram of the enhanced image is quite different from the original histogram; while with the proposed method, the histogram of the enhanced image looks similar to the original histogram. Another example is shown in Fig. 5.20.



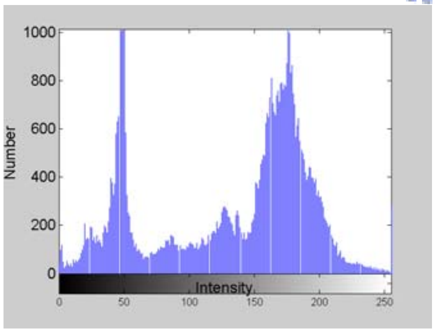


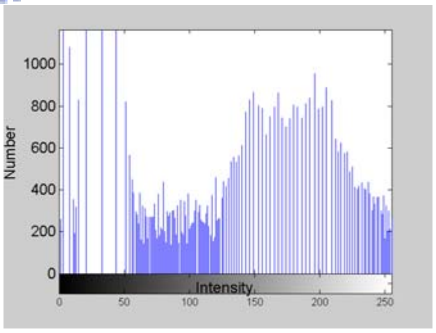
(a) (b)





(c) (d)





(e) (f)

Fig. 5.19. (a) Original image. (b) Enhanced image using proposed method. (c) Enhanced image using histogram equalization. (d)-(f) Histograms of (a)-(c) respectively.

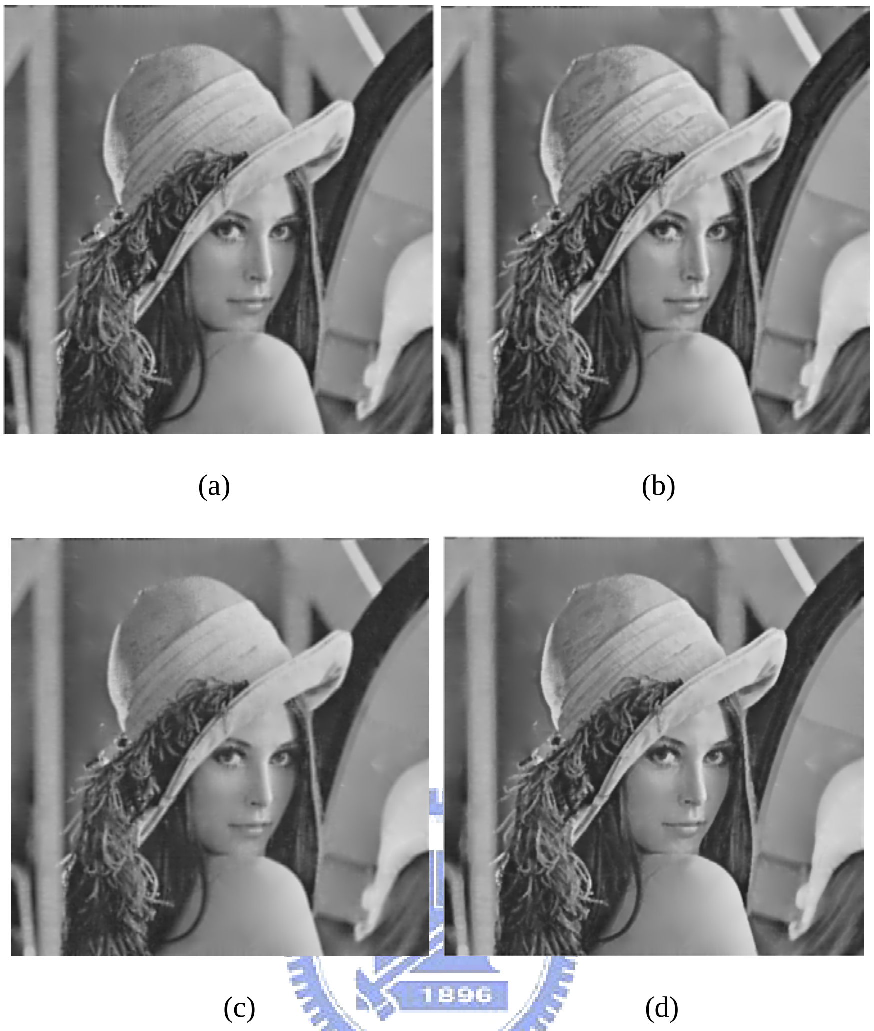


Fig. 5.20. Image contrast and sharpness altering. (a) Image with original contrast. (b) Image with enhanced contrast. (c) Image with softened contrast. (d) Edge enhancement.

The concepts of these proposed methods are applicable to color images. To extract verge points from a color image, the original color image is first converted into an appropriate color space. Here, we choose the CIE LCH color space, which is relatively uniform and convenient, for adjustment. To convert a color image from RGB space to CIE LCH space, we convert the RGB space to the CIELAB space using (3.51) and (3.52). After having converted RGB space into the LAB space, the C and H components of the LCH space map to the radius r and the orientation θ of the (a,b) coordinates in the a-b plane. For a color image, we can change the altitude of skeleton curves to modify the brightness, hue, or saturation of the image. An example of color image enhancement, in which the L component is enhanced, is shown in Fig. 5.21(b).

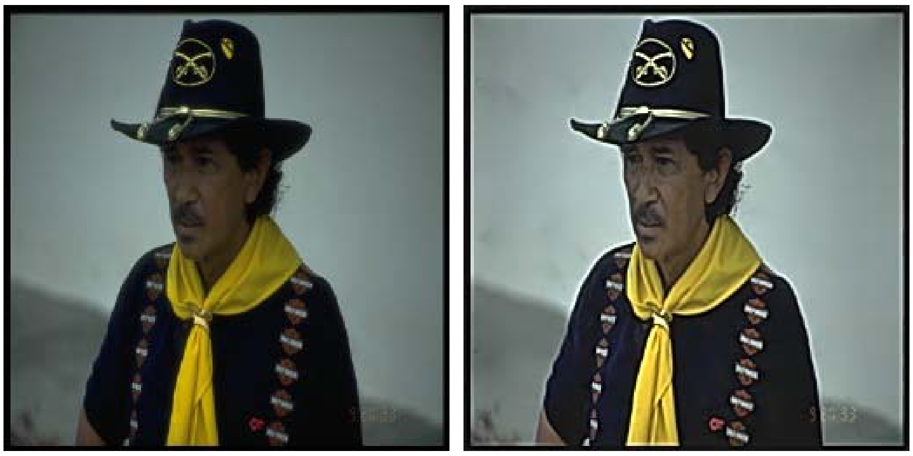

(a) (b)

Fig. 5.21 (a) Original image. (b) Enhanced color image by enhancing the L component only.

As mentioned in Chapter 4, the extracted verge points can be further linked into skeleton curves (verge curves) under the guidance of eigenvectors in **H.** This allows the users to adjust local features conveniently. The linked skeleton curves offer more information than verge points. The linked skeleton curves offer more information than verge points. As shown in Fig. 4.16, these skeleton curves with length longer than 50 pixels are extracted from Fig. 5.22.

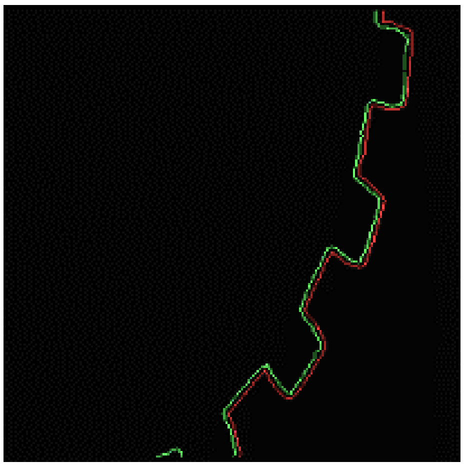


Fig. 5.22. Verge points extracted from Fig. 5.18(a) with length threshold = 50 pixels.

Based on the length of these linked skeleton curves, the skeleton component of an image may also be further decomposed into a hierarchy form. Assume, after linking, M skeleton curves are extracted. Each skeleton curve is actually a subset of Ω. We denote these M skeleton curves as $\Psi_1, \Psi_2, \ldots, \Psi_M$. Now, for each skeleton curve, we may apply different manipulations. Hence, Equation (5.5) can be further modified into a curve-dependent form as expressed in (5.6).

$$\tilde{F}(D) = \Gamma\{\{\sum_{i=1}^{M}[\sum\sum_{(\mu,\eta)\in\Psi_i}[f(\mu,\eta)+\Delta^i_{sp}(\mu,\eta)]\delta(x-\mu, y-\eta)]\}; D\} + \alpha \times F_{residual}(D). \tag{5.6}$$

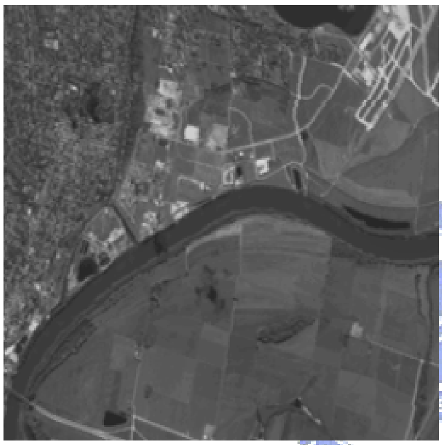
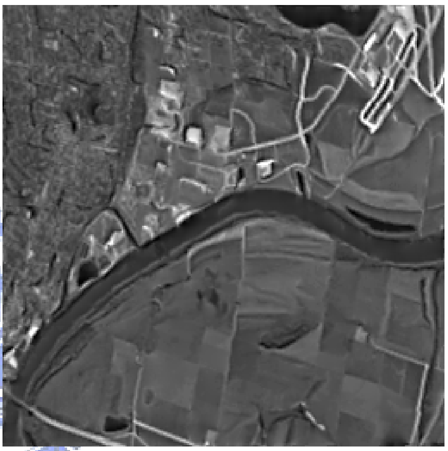
Here, we denote $\Delta^i_{sp}(\mu,\eta)$ as the amount of adjustment over the intensity value of a verge point on the ith curve. $\Delta^i_{sp}(\mu,\eta)$ may depend on the intensity at (μ, η), the average contrast of the ith curve, and the length of the ith curve. Since different processings may be needed for different features in an image, we can employ different strategies to satisfy different demands. For instance, if a user wants to enhance features with small contrast only, a threshold can be set over the curve contrast to pick up low-contrast curves. When a user hopes to highlight the objects with longer boundary, a large value of $\Delta^i_{sp}(\mu,\eta)$ can be applied over these curves with their curve length L longer than a pre-selected threshold. The proposed scheme provides a flexible scheme for various types of image enhancement methods. In comparison, it would be more troublesome to achieve this flexibility if we use some traditional image enhancement tools, like histogram equalization or the unsharp masking method. In Fig. 5.23, we show some examples of feature-based manipulations. Fig. 5.23(a) shows an original image. In this example, we set $\Delta^i_{sp}(\mu,\eta)$ as

$$\Delta^{\mathrm{i}}_{\mathrm{sp}}(\mu,\eta) = \begin{cases} 0, & \text{if } \mathrm{L(i)} < 20 \\ \sqrt{\dfrac{L(i)}{L_{\max}}} \times k_1(\mu,\eta), & \text{otherwise} \end{cases}, \tag{5.7}$$

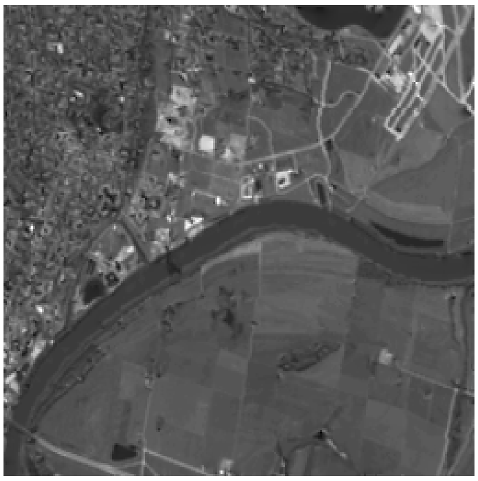
where L(i) denotes the length of the ith skeleton curve. The enhanced image based on (5.7) is shown in Fig. 5.23(b). It can be seen from this picture that features with longer boundaries are especially enhanced. In Fig. 5.23(c), on the other hand, we set $\Delta_{sp}^{i}(\mu,\eta)$ as

$$\Delta_{\mathrm{sp}}^{\mathrm{i}}(\mu,\eta)=\begin{cases}0 & \text{if } \mathrm{L(i)}>5\\ 2\times k_1(\mu,\eta), & \text{otherwise}\end{cases}. \tag{5.8}$$

It can be seen that features with shorter boundaries are enhanced.

(a) (b)



(c)

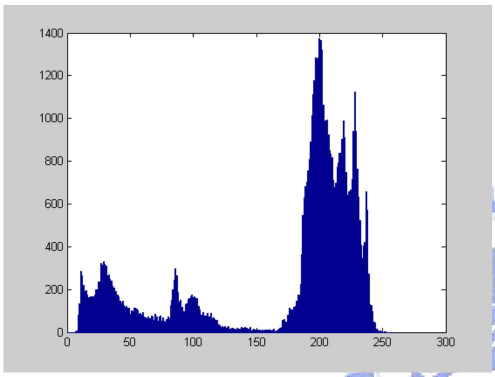
Fig. 5.23. (a) Original image. (b) Enhancing long features only. (c) Enhancing short features only.

The proposed image editing tool is also suitable for interactive backlight compensation. Fig. 5.24(a) shows an original image. Due to strong lightening, the man in the picture is in dark appearance. Fig. 5.24(b) shows the processed image of Fig. 24(a) after histogram equalization, and the corresponding histograms are shown in Fig. 5.24(c) and (d). We can see that histogram equalization does little help in enhancing this image. Global contrast adjustment may fail either, since raising the intensity level or increasing the contrast may lead to the undesired change of the background. The dilemma may be solved using the proposed method. The boundary skeletons curves around the man are grouped by interactively selecting these skeletons curves (see Fig. 5.24(e)). Then, only the skeleton curves inside the selected boundary (see Fig. 5.24(f)) are shifted higher along the intensity axis. Fig. 5.24(g) shows the final result, where the intensity shift is set to be 70. We can see from this experiment that the proposed method is quite flexible in editing objects in an image.

Fig. 5.25(a) shows a color image. In Fig. 5.25(b), we modify the hue component of some skeleton curves to change the chromatic information of the image content in the upper-right corner. In comparison, it would be difficult to achieve this task using pixel-based processing or a block-based processing.

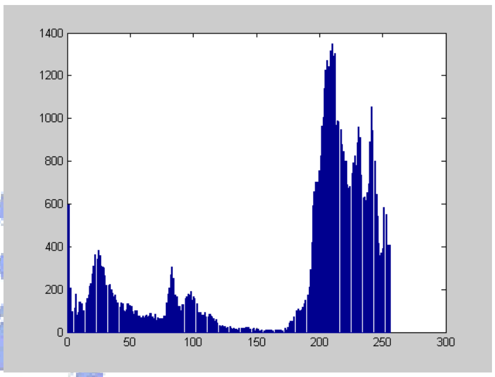
(a)

(b)



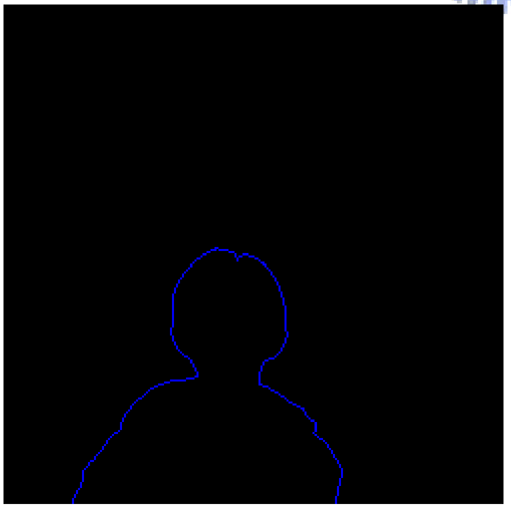
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300

(c)

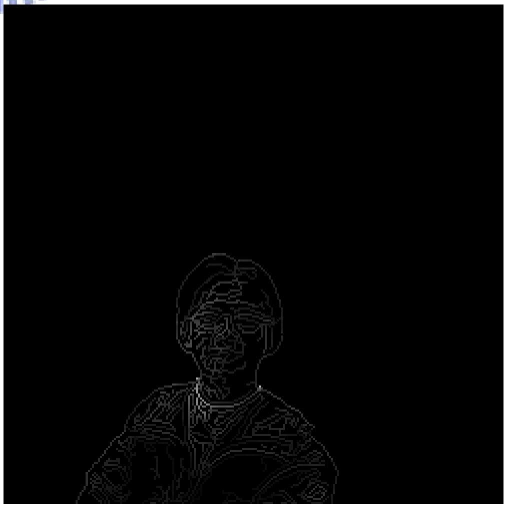


1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300

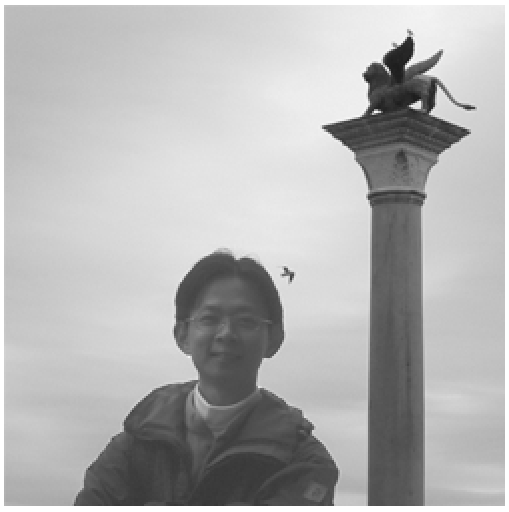
(d)



(e)

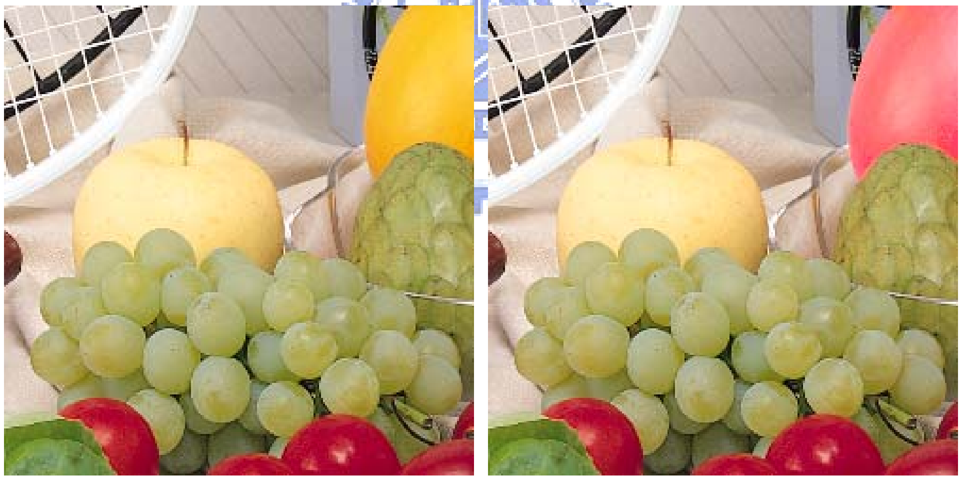


(f)



(g)

Fig. 5.24. (a) Original image. (b) Image of (a) after histogram equalization. (c) Histogram of (a). (d) Histogram of (b). (e) Interactive skeleton curves grouping. (f) Selected skeleton curves. (g) Image editing using the proposed method.



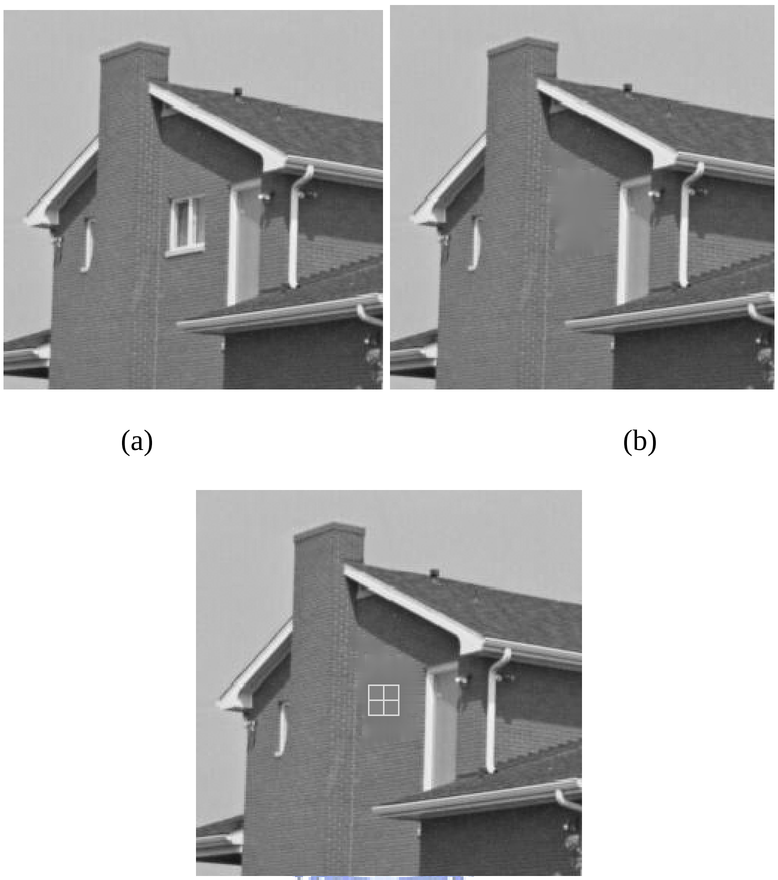
(a) (b)

Fig. 5.25. (a) Original Image. (b) Interactive editing.

5.2.3 Editing Images in Shape Domain

Since skeleton curves map to specific features of objects, shapes in an image could be edited by manipulating these skeleton curves. In Fig. 5.26(b), we show the removal of a window in Fig. 5.26(a) by eliminating the corresponding skeleton curves of the window. In Fig. 5.26(c), on the contrary, by adding some extra skeleton curves on Fig. 5.26(b), a new window is created.

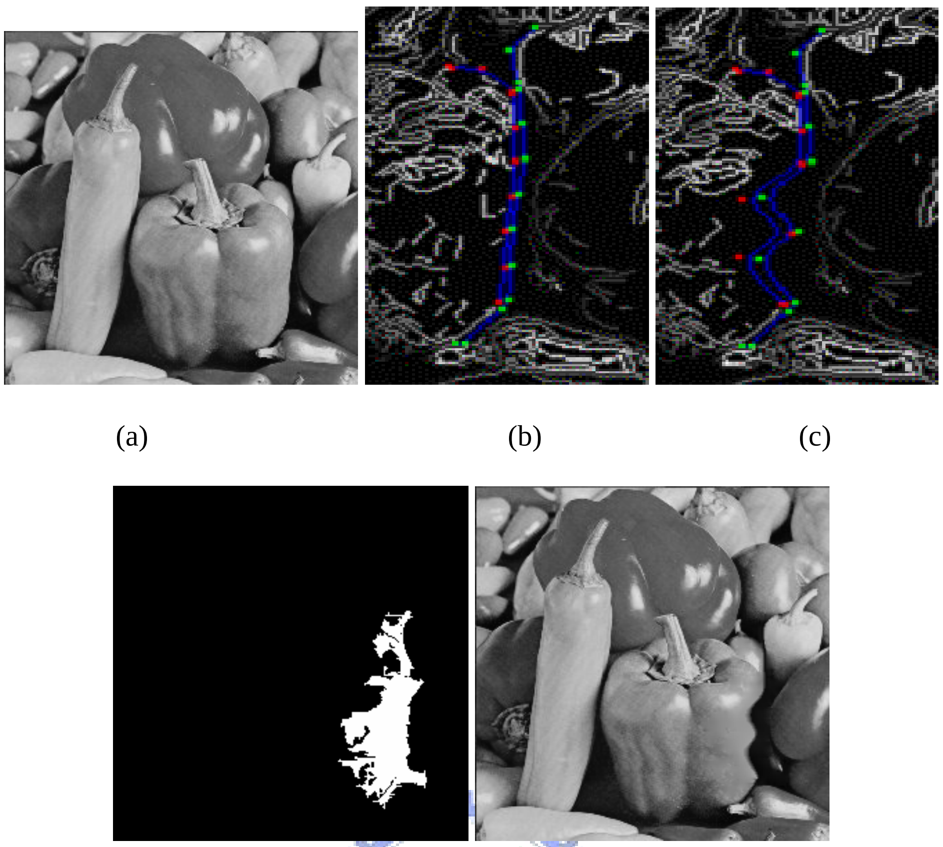
As indicated in Chapter 4, we use B-spline control points to approximate verge curves. This approximation further strengthens the flexibility of the proposed architecture. By shifting the positions of the control points in x-y plane, the shape of the selected object is changed. After this modification, however, the residual component cannot be added back directly since it does not match with the modified skeleton component. To solve this problem, we record for every pixel the supporting skeleton curves of that pixel. The supporting skeleton curves are defined as these skeleton curves that have ever participated in the linear interpolation process of this pixel. Once the shape of a curve is modified, all those pixels whose supporting skeleton curves contain this modified curve are marked as pixels of influence zone. The residual information in the influence zone is discarded in the reconstruction process to avoid mismatch in the modified image. The experimental results of shape editing are shown in Fig. 5.27. Fig. 5.27(a) shows an original image. In Fig. 5.27(b) and 5.27(c), the original control points and the modified control points are demonstrated. Fig. 5.27(d) presents the influence zones due to modification of some control points. Finally, in Fig. 5.27(e) the image with modified shape is shown. Compared with [73], the proposed method may provide wider flexibility in shape editing since the curves are approximated using B-spline representation. In addition, the rotation and scaling of an object approximated by B-spline curves are feasible due to the affined invariant property of B-spline curves. Fig. 5.28 shows another example.



(a) (b)

(c)

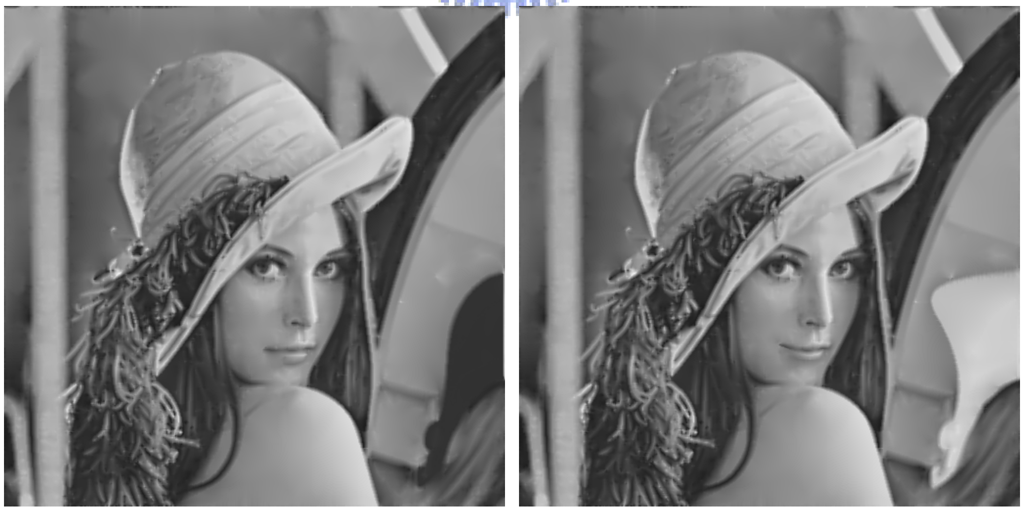
Fig. 5.26. (a) Original image. (b) Example of feature removals. (c) Example of feature creations.



(a) (b) (c)

(d) (e)

Fig. 5.27. Interactive shape editing. (a) Original image. (b) Original control points. (c) Modified control points. (d) Influenced zone. (e) Modified image.



(a) (b)

Fig. 5.28. Illustrations of image manipulations using verge points. (a) Image editing by changing the gray level of a region at the lower-right corner. (b) Image editing by changing the shape of mouth and the shape of a region at the lower-right corner.