# Chapter 3

# Verge Point Extraction and Image Surface Reconstruction

In the proposed image representation scheme, high curvature points on the image surfaces are adopted to represent an original image. In this chapter, the precise definition of verge point is introduced, and the procedures to obtain verge points are described in detail and analyzed comprehensively. In Chapter 3.1, we will introduce the procedure of extracting verge points on one-dimensional profiles. The procedure of extracting verge points on two-dimensional image surfaces are to be described in Chapter 3.2. Image surface reconstruction based on these extracted verge points is to be presented in Chapter 3.3 then.

## 3.1 Extracting Verge Points on One-Dimensional Profiles

As indicated in Chapter 1, we may imagine an intensity profile as an elastic string stretched by a few pulleys. Conceptually, these pulleys locate at highly curved places, and the radius of each pulley is inversely proportional to the local curvature of the profile. While the elastic string is stretched by pulleys, there exists stress between every adjacent pulley pair. Since the elastic string is fixed on both ends, we consider the stress

is large enough to resist gravity. Therefore, elastic string segments between two adjacent pulleys are considered to be straight line segments. The locations and sizes of these pulleys, together with the endpoints of the string, offer enough information to describe the outline of the profile. To detect and estimate these pulley positions and pulley sizes, we first calculate the curvature κ of the profile. If the profile is expressed as y = f(x), its curvature can be calculated as

$$\kappa(x) = \frac{f''(x)}{(1+(f'(x))^2)^{\frac{3}{2}}}, \tag{3.1}$$

where $f'(x)$ and $f''(x)$ denote the first and second derivatives of f(x), respectively [39]. A brief introduction to the derivation of (3.1) is available in Appendix A.

Then we calculate the local extremes (local maxima or local minima) of κ(x) since these pulleys are to be placed at highly curved places. Assume κ(x) reaches its local extremes at $x_{ex}$; that is, $\left.\frac{d\kappa(x)}{dx}\right|_{x=x_{ex}} = 0$. Then, $P_e \equiv (x_{ex}, f(x_{ex}))$ is chosen as the place where a pulley is tangentially placed on (or under) the profile, and the reciprocal of $|\kappa(x_{ex})|$ is used to approximate the radius r of the pulley. Since $P_e$ locates near the end points of the edges and smooth regions, we name $P_e$ as "verge point" through this paper.

At a verge point $P_e$, the first derivative $f'(x)$ at $x_{ex}$ can be used to estimate the tangential direction $\vec{e}_t$ at $P_e$ (see Fig. 3.1(a)). The pulley center can thus be located. Note that the sign of κ(x) at $x_{ex}$ indicates how the pulley bends the string around $P_e$. For a positive κ(x), the pulley bends down the string to form a convex arc; while for a negative κ(x), the pulley bends up the string to form a concave arc. Consequently, to

describe a pulley, three parameters are needed: pulley center, pulley radius, and pulley sign.

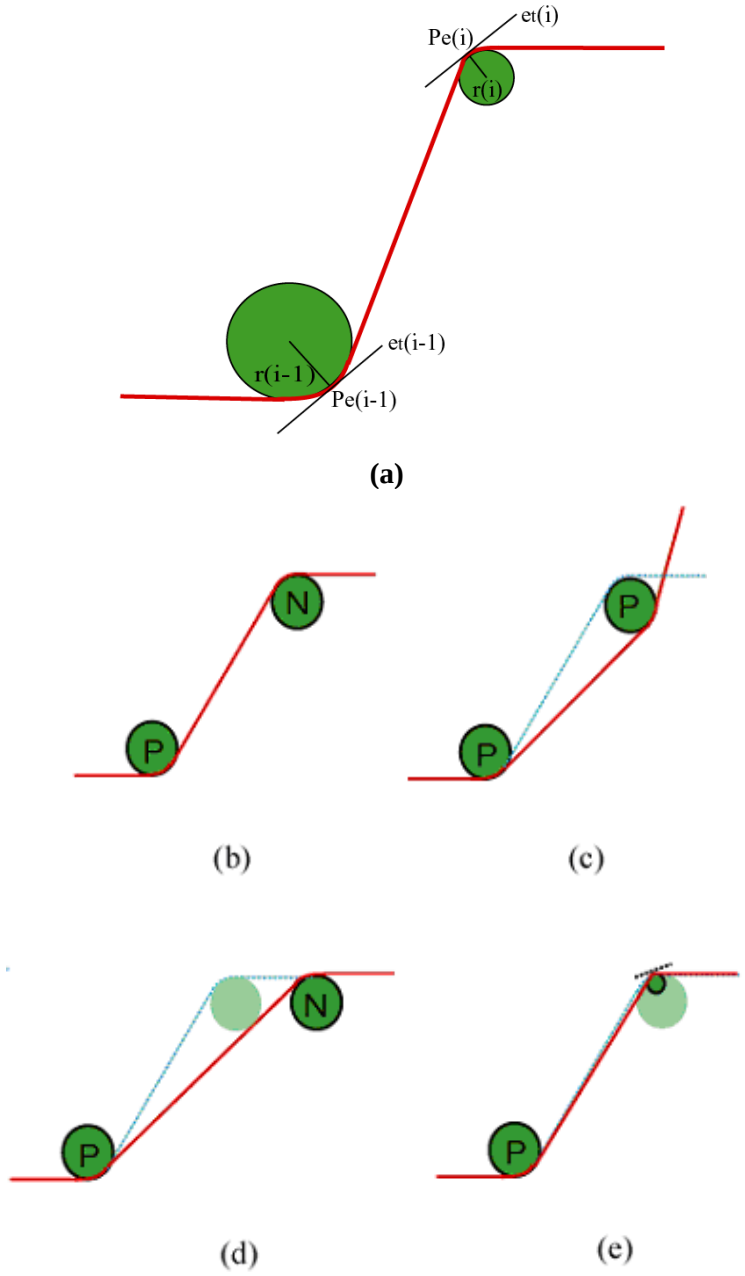



Fig. 3.1. Concept of verge points. (a) Illustration of $P_e$ and $\vec{e}_t$. (b) Reconstructed profile with correct pulley parameters. (c) Reconstructed profile with the sign of the right pulley being incorrectly recorded. (d) Reconstructed profile with the position of the right pulley being incorrectly recorded. (e) Reconstructed profile with the radius of the right pulley being incorrectly recorded.

With this pulley representation, the original profile can be easily reconstructed to a fairly accurate level. To achieve different degrees of resolutions, we can simply adjust the amount of pulleys accordingly: fewer pulleys for a lower resolution profile and more pulleys for a higher resolution profile. Moreover, any mistakenly placed pulley only changes the local shape of the profile.

To derive a more compact form to represent pulleys, an intuitive analysis is illustrated in Fig. 3.1 (b)-(e) to evaluate the impact of these three parameters. In Fig. 3.1(b), a profile reconstructed from two pulleys with opposite signs (a positive and a negative) is illustrated. Fig. 3.1(c)-(e) show 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 thesis, 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. Moreover, by shrinking the pulley to a unit circle (a point), the reconstruction process becomes much faster and easier. This is because we do not need to consider the reconstruction of the arc portion of the pulleys. The reconstruction of arc portion is time-consuming and complicated, especially for the two-dimensional case.

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. By using this verge point representation, the data amount required to represent the original profile is greatly reduced. This suggests the compactness of verge point representation, and also implies the potential of image compression using verge

point representation. The compactness of the verge point representation will be investigated in Chapter 4.

Once the major verge points of a profile are extracted, these points can be used to detect features, like 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 $\Delta L$ per unit length L is defined as strain. Assume the distance between two adjacent verge points is $W_d$ and the intensity difference between these two verge points is $H_d$. Then, the strain between two verge points can be estimated by

$$\varepsilon = \frac{|H_d|}{W_d}. \tag{3.2}$$

Both the strain parameter and the contrast $H_d$ can serve as a parameter to perform edge detection. This will be discussed further in Chapter 5.

Besides feature detection, image enhancement and editing can be achieved based on verge points. For example, to perform edge enhancement, we can move closer these verge pairs with opposite signs, as shown in Fig. 3.2(b). To perform contrast enhancement, we can move upward the verge points with negative sign while move downward the verge points with positive sign, as shown in Fig. 3.2(c). By moving the positions of the verge points, the shape of the elastic string is changed thereafter. This allows us to perform feature editing. This issue about feature editing is to be explored in Chapter 5.

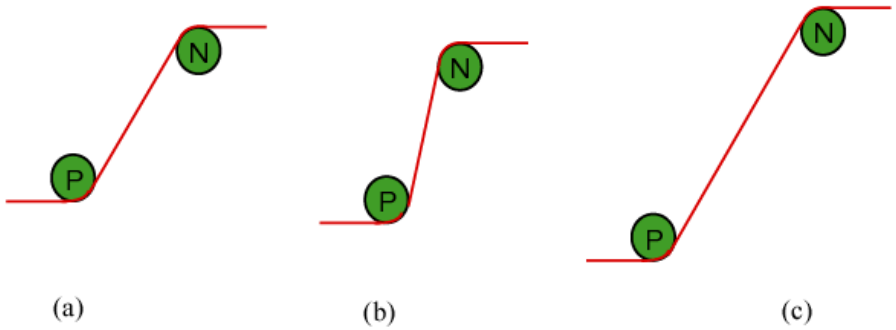



Fig. 3.2. Enhancement using verge points. (a) Original profile. (b) Edge enhancement. (c) Contrast enhancement.

# 3.2. Extracting Verge Points on Image Surfaces

## 3.2.1. Verge Points Extraction Based on Principal Curvature

The concept of verge points can be easily extended to represent 2-D images. Similar to the concept of using pulleys to represent intensity profiles, we can imagine the image surface as a plastic cloth stretched by 3-D pipes. Fig. 3.3(a) presents a synthetic image and Fig. 3.3(b) shows its image surface. As shown in Fig. 3.3(c), by placing pipes at highly curved places, the stretched rubber cloth can emulate the image surface. In differential geometry, these highly curved places happen at the positions where at least one of the two principal curvatures has a large enough magnitude [39].

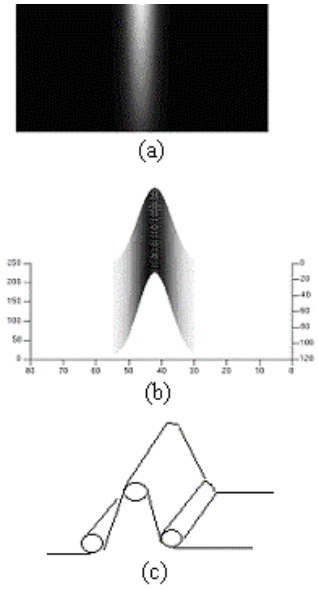



Fig. 3.3. Image surface emulation using a rubber cloth and pipes. (a) Synthetic image. (b) Corresponding image surface of (a). (c) Illustration of a rubber cloth stretched by pipes.

In differential geometry, the second fundamental form is usually used to measure the curvature at a point **p** on a surface S along a specific tangent vector $\vec{t}$. This second fundamental form is defined as

$$II_p(\vec{t}) = -\langle d\vec{N}_p(\vec{t}), \vec{t} \rangle, \tag{3.3}$$

where $\vec{N}_p$ denotes the unit normal vector at **p** and $d\vec{N}_p(\vec{t})$ denotes the differential of $\vec{N}_p$ along $\vec{t}$ [39]. If a curve on the surface is formulated as $\vec{\alpha}(s) = S(u(s), v(s))$, which is a map from a 2-D domain $(u(s), v(s))$ to the surface, then the linear map $d\vec{N}_p(\vec{t})$ can also be formulated as

$$dN_p \begin{pmatrix} \frac{du(s)}{ds} \\ \frac{dv(s)}{ds} \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{pmatrix} \frac{du(s)}{ds} \\ \frac{dv(s)}{ds} \end{pmatrix} \equiv \mathbf{A} \begin{pmatrix} \frac{du(s)}{ds} \\ \frac{dv(s)}{ds} \end{pmatrix}, \tag{3.4}$$

where the elements of matrix **A** can be obtained by using the first and second fundamental forms [39]. A brief introduction concerning the derivation of (3.4) is available in Appendix B.

For a point **p** on a surface S, once the elements in matrix **A** are identified, the normal curvature along any direction on the tangential plane can be calculated. The two eigenvectors of matrix **A** indicate the directions along which the surface around **p** has the maximum normal curvature and minimum normal curvature, with their eigenvalues $k_1$ and $k_2$ indicating the normal curvature values. These two eigenvalues can be used to detect highly curved places on a surface.

With the matrix A, Gaussian curvature K and mean curvature H, which are commonly used in differential geometry, can be calculated as

$$\begin{aligned} K &= \det(A) \\ H &= -\frac{1}{2}(a_{11} + a_{22}) \end{aligned} . \tag{3.5}$$

It can be proved that the principal curvatures $k_1$ and $k_2$ can be easily computed from K and H as

$$\begin{aligned} k_1 &= H + \sqrt{H^2 - K} \\ k_2 &= H - \sqrt{H^2 - K} \end{aligned} . \tag{3.6}$$

For an image surface in the form of (x, y, f(x,y)), the directions of these two principal curvatures can be deduced by calculating the eigenvalues and eigenvectors of the following matrix [39]:

$$\mathbf{A} = \frac{1}{(1 + f_x{}^2 + f_y{}^2)^{3/2}} \begin{bmatrix} -f_{xy} f_x f_y + f_{xx}(1 + f_y{}^2) & -f_{yy} f_x f_y + f_{xy}(1 + f_y{}^2) \\ -f_{xx} f_x f_y + f_{xy}(1 + f_x{}^2) & -f_{xy} f_x f_y + f_{yy}(1 + f_x{}^2) \end{bmatrix}. \tag{3.7}$$

In this thesis, we denote $k_1(x, y)$ as the eigenvalue with the larger magnitude and denote $\Lambda_1(x, y)$ 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 2nd-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 $G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp^{-(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2})}$ and the symbol "*" denotes the convolution operation. In this thesis, we choose $\sigma_x = \sigma_y = \sigma_m$. A larger $\sigma_m$ 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 (3.7) has a poor SNR performance. Fig. 3.4 illustrates such a phenomenon. Fig. 3.4(a) shows an image of step edge polluted by white Gaussian noise with $\sigma_n = 2$, and Fig. 3.4(b) shows the traversal profile of Fig. 3.4(a). Fig. 3.4(c) shows the estimated $|k_1|$ profile based on Matrix **A**. It appears that the curvature estimation is so noisy that the detection of verge points becomes extremely difficult.

In our approach, as used by Jong in [38], the Hessian matrix **H** is used for the estimation of principal curvatures. The Hessian **H** ignores all the first-order terms in **A** and is expressed as

$$\mathbf{H} = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}. \tag{3.8}$$

Similarly, the eigenvalues of **H** are used as the estimation of principal curvatures. Fig. 3.4(d) shows the estimated $|\hat{k}_1|$ profile based on **H**. It can be seen that the use of **H**, instead of **A**, offers a much improved signal-to-noise ratio in estimating principal curvatures.

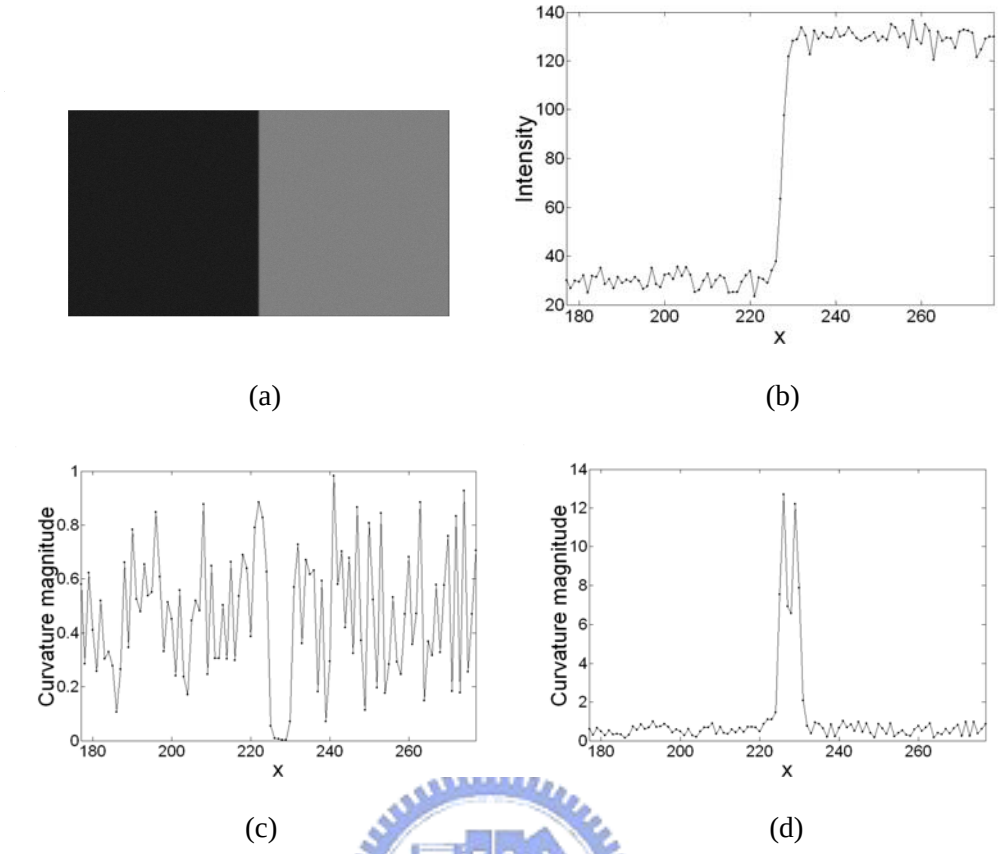



Fig. 3.4. Comparisons between **A** and **H**, in terms of SNR performance. (a) Step edge image polluted by white Gaussian noise with $\sigma_n = 2$. (b) A transversal profile of (a). (c) Computed curvature profile based on Matrix **A**. (d) Computed curvature profile based on Hessian **H**.

## 3.2.2. Quantitative Comparisons between Matrix **M** and Hessian **H**

To give a quantitative description of the phenomenon on principal curvature estimation, we compare **A** and **H** in terms of their SNR performance. Assume the principal curvatures estimated from **A** and **H** are denoted as $k$ and $\hat{k}$, respectively. Due to image noise, there are fluctuations in the estimations of $f_x$, $f_y$, $f_{xx}$, $f_{xy}$, and $f_{yy}$. In

this thesis, we denote these fluctuations as $\Delta f_x$, $\Delta f_y$, $\Delta f_{xx}$, $\Delta f_{xy}$, and $\Delta f_{yy}$. We first calculate the means and variances of these fluctuations.

Assume we denote the signal part of an image as s(x,y) while the noise part as n(x,y). That is,

$$f(x,y) = s(x,y) + n(x,y)\,. \tag{3.9}$$

The estimation of $f_x$ can be expressed as

$$f(x,y) * \frac{\partial G(x,y)}{\partial x} = s(x,y) * \frac{\partial G(x,y)}{\partial x} + n(x,y) * \frac{\partial G(x,y)}{\partial x} \equiv s(x,y) * \frac{\partial G(x,y)}{\partial x} + \Delta f_x(x,y)\,. \tag{3.10}$$

Assume the noise n(x,y) is additive white Gaussian noise with variance $\sigma_n^2$. Then, the mean and variance of $\Delta f_x$ can be expressed as

$$E[\Delta f_x(x,y)] = E[n(x,y) * \frac{\partial G(x,y)}{\partial x}] = E[n(x,y)] * \frac{\partial G(x,y)}{\partial x} = 0 \ , \text{ and} \tag{3.11}$$

$$\begin{aligned}
&Var[\Delta f_x(x,y)] = E[(\Delta f_x(x,y))^2] \\
&= E[(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_x\sigma_y}(-\frac{p}{\sigma_x^2}) e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2})} n(x-p,y-q)dpdq)^2] \\
&= E[(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{p}{\sigma_x^2})(-\frac{\hat{p}}{\sigma_x^2}) e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2})} \times n(x-p,y-q)n(x-\hat{p},y-\hat{q})dpdqd\hat{p}d\hat{q})] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{p}{\sigma_x^2})(-\frac{\hat{p}}{\sigma_x^2}) e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2})} \times E[n(x-p,y-q)n(x-\hat{p},y-\hat{q})]dpdqd\hat{p}d\hat{q} \\
&= (\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{p}{\sigma_x^2})^2 e^{-(\frac{p^2}{\sigma_x^2}+\frac{q^2}{\sigma_y^2})} dpdq)\cdot\sigma_n^2 \\
&= \frac{1}{4\sqrt{\pi}\sigma_x^3}\frac{1}{2\sqrt{\pi}\sigma_y}\sigma_n^2
\end{aligned} \tag{3.12}$$

In this thesis, we choose the scale parameters to be equal; that is, $\sigma_x = \sigma_y = \sigma_m$. Hence, (3.12) can be expressed as

$$Var[\Delta f_x(x,y)] = \frac{1}{8\pi\sigma_m^4}\sigma_n^2\,. \tag{3.13}$$

Similarly, the mean of $\Delta f_y$, $\Delta f_{xx}$, $\Delta f_{xy}$, and $\Delta f_{yy}$ can be calculated to be

$$E[\Delta f_y(x,y)] = E[\Delta f_{xx}(x,y)] = E[\Delta f_{xy}(x,y)] = E[\Delta f_{yy}(x,y)] = 0\,. \tag{3.14}$$

Equivalently, the variance of $\Delta f_y$ is derived as

$$Var[\Delta f_y(x,y)] = \frac{1}{8\pi\sigma_m^4}\sigma_n^2, \tag{3.15}$$

and the variances of $\Delta f_{xx}$ and $\Delta f_{yy}$ can be derived as

$$\begin{aligned}
&Var[\Delta f_{xx}(x,y)] = E[(\Delta f_{xx}(x,y))^2] \\
&= E[(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{2\pi\sigma_x\sigma_y}(-\frac{1}{\sigma_x^2}-(\frac{p}{\sigma_x^2})^2)e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2})}n(x-p,y-q)dpdq)^2] \\
&= E[(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{1}{\sigma_x^2}-(\frac{p}{\sigma_x^2})^2)(-\frac{1}{\sigma_x^2}-(\frac{\hat{p}}{\sigma_x^2})^2)e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2})} \\
&\times n(x-p,y-q)n(x-\hat{p},y-\hat{q})dpdqd\hat{p}d\hat{q})] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{1}{\sigma_x^2}-(\frac{p}{\sigma_x^2})^2)(-\frac{1}{\sigma_x^2}-(\frac{\hat{p}}{\sigma_x^2})^2)e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2})} \\
&\times E[n(x-p,y-q)n(x-\hat{p},y-\hat{q})]dpdqd\hat{p}d\hat{q} \\
&= \frac{3}{16\pi\sigma_m^6}\sigma_n^2 = Var[\Delta f_{yy}(x,y)]
\end{aligned} \tag{3.16}$$

On the other hand, the variance of $\Delta f_{xy}$ can be derived as

$$\begin{aligned}
&Var[\Delta f_{xy}(x,y)] = E[(\Delta f_{xy}(x,y))^2] \\
&= E[(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{2\pi\sigma_x\sigma_y}(-\frac{p}{\sigma_x^2}-\frac{q}{\sigma_x^2})e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2})}n(x-p,y-q)dpdq)^2] \\
&= E[(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{p}{\sigma_x^2}-\frac{q}{\sigma_x^2})(-\frac{\hat{p}}{\sigma_x^2}-\frac{\hat{q}}{\sigma_x^2})e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2})} \\
&\times n(x-p,y-q)n(x-\hat{p},y-\hat{q})dpdqd\hat{p}d\hat{q})] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\frac{1}{2\pi\sigma_x\sigma_y})^2(-\frac{p}{\sigma_x^2}-\frac{q}{\sigma_x^2})(-\frac{\hat{p}}{\sigma_x^2}-\frac{\hat{q}}{\sigma_x^2})e^{-(\frac{p^2}{2\sigma_x^2}+\frac{q^2}{2\sigma_y^2}+\frac{\hat{p}^2}{2\sigma_x^2}+\frac{\hat{q}^2}{2\sigma_y^2})} \\
&\times E[n(x-p,y-q)n(x-\hat{p},y-\hat{q})]dpdqd\hat{p}d\hat{q} \\
&= \frac{1}{16\pi\sigma_m^6}\sigma_n^2
\end{aligned} \tag{3.17}$$

Then, without loss of generality, we discuss the case of a vertical step edge as shown in Fig. 3.4(a). This step edge is modeled as

$$s(x,y) = h_0 + h\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma_b{}^2}} e^{-\frac{\tau^2}{2\sigma_b{}^2}} d\tau \,. \tag{3.18}$$

After calculating all the derivatives, we have

$$s_y = \frac{d(h_0 + h\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma_b{}^2}} e^{-\frac{\tau^2}{2\sigma_b{}^2}} d\tau)}{dy} = 0\,, \tag{3.19}$$

$$s_{xy} = \frac{d^2(d(h_0 + h\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma_b{}^2}} e^{-\frac{\tau^2}{2\sigma_b{}^2}} d\tau))}{dxdy} = 0\,, \tag{3.20}$$

and

$$s_{yy} = \frac{d^2(d(h_0 + h\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma_b{}^2}} e^{-\frac{\tau^2}{2\sigma_b{}^2}} d\tau))}{dy^2} = 0\,, \tag{3.21}$$

Hence, we have

$$f_x = s_x + \Delta f_x\,, \tag{3.22}$$

$$f_y = \Delta f_y\,, \tag{3.23}$$

$$f_{xx} = s_{xx} + \Delta f_{xx}\,, \tag{3.24}$$

$$f_{xy} = \Delta f_{xy}\,, \tag{3.25}$$

and

$$f_{yy} = \Delta f_{yy}\,. \tag{3.26}$$

Since the image surfaces around edge sections and smooth regions are different, we discuss the feature detections using matrix **A** and Hessian **H** for edge regions (Case I) and for smooth regions (Case II). We first discuss noise interference in edge sections.

***Case I: 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

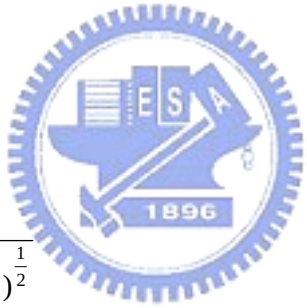
$$\mathbf{A} \approx \frac{1}{(1+f_x^{\,2})^{\frac{3}{2}}}\begin{bmatrix} f_{xx} & \Delta f_{xy} \\ \Delta f_{xy}(1+f_x^{\,2}) & \Delta f_{yy}(1+f_x^{\,2}) \end{bmatrix}, \tag{3.27}$$

and

$$\mathbf{H} \approx \begin{bmatrix} f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}. \tag{3.28}$$

In matrix **A**, since the diagonal term $\Delta f_{xy}$ $\Delta f_{xy}(1+f_x^{\,2})$ is small, we may approximate the curvature estimate of **A** as

$$\begin{aligned} &\textit{if } f_{xx} > \Delta f_{yy}(1+f_x^{\,2}) \\ &k \approx \frac{f_{xx}}{(1+f_x^{\,2})^{\frac{3}{2}}} \\ &\textit{else} \\ &k \approx \frac{\Delta f_{yy}(1+f_x^{\,2})}{(1+f_x^{\,2})^{\frac{3}{2}}} = \frac{\Delta f_{yy}}{(1+f_x^{\,2})^{\frac{1}{2}}} \end{aligned} \tag{3.29}$$



The curvature estimate of **H** may be approximated as

$$\hat{k} \approx f_{xx}. \tag{3.30}$$

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

$$\begin{aligned} &\textit{if } f_{xx} > \Delta f_{yy}(1+f_x^{\,2}) \\ &\Delta k \approx \frac{\Delta f_{xx}}{(1+f_x^{\,2})^{\frac{3}{2}}} - \frac{3 f_{xx} f_x}{(1+f_x^{\,2})^{\frac{5}{2}}}\Delta f_x \\ &\textit{else} \\ &\Delta k \approx \frac{\Delta f_{yy}(1+f_x^{\,2})}{(1+f_x^{\,2})^{\frac{3}{2}}} = \frac{\Delta f_{yy}}{(1+f_x^{\,2})^{\frac{1}{2}}} \end{aligned} \tag{3.31}$$

and

$$\Delta\hat{k} \approx \Delta f_{xx} \,. \tag{3.32}$$

The variance of $\Delta k$ can thus be calculated as

$$\begin{aligned} &if \;\; f_{xx} > \Delta f_{yy}(1+ f_x{}^2) \\ &Var(\Delta k) \approx \frac{1}{(1+ f_x{}^2)^3} Var(\Delta f_{xx}) - 2\frac{3 f_{xx} f_x}{(1+ f_x{}^2)^4} E(\Delta f_{xx} \Delta f_x) + \frac{9 f_{xx}{}^2 f_x{}^2}{(1+ f_x{}^2)^5} Var(\Delta f_x) \\ &else \\ &Var(\Delta k) \approx \frac{1}{(1+ f_x{}^2)^{\frac{1}{2}}} Var(\Delta f_{yy}) \end{aligned} \tag{3.33}$$

$E[\Delta f_{xx}(x,y)\Delta f_x(x,y)]$ can be obtained as

$$\begin{aligned} &E[\Delta f_{xx}(x,y)\Delta f_x(x,y)] \\ &= E[(n(x,y) * \frac{\partial^2 G(x,y)}{\partial x^2})(n(x,y) * \frac{\partial G(x,y)}{\partial x})] \\ &= E[n(x,y)] * E[n(x,y)] * \frac{\partial^2 G(x,y)}{\partial x^2} * \frac{\partial G(x,y)}{\partial x} \\ &= 0 \end{aligned} \tag{3.34}$$

Thus, we have

$$\begin{aligned} &if \;\; f_{xx} > \Delta f_{yy}(1+ f_x{}^2) \\ &Var(\Delta k) \approx \frac{1}{(1+ f_x{}^2)^3} Var(\Delta f_{xx}) + \frac{9 f_{xx}{}^2 f_x{}^2}{(1+ f_x{}^2)^5} Var(\Delta f_x) \\ &else \\ &Var(\Delta k) \approx \frac{1}{(1+ f_x{}^2)^{\frac{1}{2}}} Var(\Delta f_{yy}) \end{aligned} \tag{3.35}$$

On the other hand, we can deduce that

$$Var(\Delta\hat{k}) \approx Var(\Delta f_{xx}). \tag{3.36}$$

Now let us discuss another case, which is about noise interference in smooth regions.

***Case II: Over smooth regions.***

Around smooth regions, $f_{xx} \approx 0$ and $f_x \approx 0$. In this case, we have

$$\mathbf{A} \approx \frac{1}{(1+(\Delta f_x)^2+(\Delta f_y)^2)^{3/2}} \begin{bmatrix} \Delta f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}, \tag{3.37}$$

and

$$\mathbf{H} \approx \begin{bmatrix} \Delta f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}. \tag{3.38}$$

If $\Delta f_x << 1$ and $\Delta f_y << 1$, then the curvature values estimated from **A** and H would be approximately the same. Therefore, $Var(|\Delta k|) \approx Var(|\Delta \hat{k}|)$.

Let $|\Delta \hat{k}_1|$ and $|\Delta \hat{k}_2|$ be the magnitudes of principal curvatures estimated from $\begin{bmatrix} \Delta f_{xx} & \Delta f_{xy} \\ \Delta f_{xy} & \Delta f_{yy} \end{bmatrix}$ and $|\Delta \hat{k}_1| \geq |\Delta \hat{k}_2|$. We have

$$|\Delta \hat{k}_1| = \frac{|\Delta f_{xx} + \Delta f_{yy}| + \sqrt{(\Delta f_{xx} + \Delta f_{yy})^2 - 4(\Delta f_{xx}\Delta f_{yy} - \Delta f_{xy}^2)}}{2}. \tag{3.39}$$

Here, we define

$$\Delta q \equiv |\Delta f_{xx} + \Delta f_{yy}|, \tag{3.40}$$

and

$$\Delta r \equiv \sqrt{(\Delta f_{xx} + \Delta f_{yy}) - 4(\Delta f_{xx}\Delta f_{yy} - f_{xy}^2)}. \tag{3.41}$$

After a long deduction, it can be proved that the pdf (probability distribution function) of $\Delta q$ follows

$$f_{\Delta q}(\Delta q) = \begin{cases} \dfrac{2}{\sqrt{2\pi\sigma_{\Delta p}^2}} e^{-\frac{\Delta q^2}{2\sigma_{\Delta p}^2}} & \Delta q \geq 0 \\ 0 & \Delta q < 0 \end{cases} \quad \text{with} \quad \sigma_{\Delta p}^2 = \frac{\sigma_n^2}{2\pi\sigma_m^6}; \tag{3.42}$$

while the pdf of $\Delta r$ follows

$$f_{\Delta r}(\Delta r)=\begin{cases}\dfrac{\Delta r}{\sigma_s}e^{-\frac{(\Delta r)^2}{2\sigma_s^{\,2}}} & \Delta r\ge 0\\ 0 & \Delta r<0\end{cases}\qquad \text{with } \sigma_s^2=\frac{\sigma_n^2}{4\pi\sigma_m^6}. \tag{3.43}$$

After combining (3.38)-(3.42), we finally have

$$E[|\Delta\hat{k}_1|]=E[\frac{\Delta q+\Delta r}{2}]=\frac{1}{2}(\frac{1}{2\sqrt{2}}+\frac{1}{\pi})\frac{1}{\sigma_m^{\,3}}\sigma_n, \tag{3.44}$$

and

$$\begin{aligned}Var[|\Delta\hat{k}_1|]&=(\frac{1}{\sqrt{2}\pi}+\frac{1}{\pi})\frac{\sigma_n^{\,2}}{4\sigma_m^{\,6}}-\{\frac{1}{2}(\frac{1}{2\sqrt{2}}+\frac{1}{\pi})\frac{1}{\sigma_m^{\,3}}\sigma_n\}^2\\&=\frac{\sigma_n^{\,2}}{4\sigma_m^{\,6}}\{\frac{1}{\pi}-\frac{1}{\pi^2}-\frac{1}{8}\}\end{aligned}. \tag{3.45}$$

As indicated in the previous deduction, $k\approx\dfrac{f_{xx}}{(1+(f_x)^2)^{\frac{3}{2}}}$ or $\dfrac{\Delta f_{yy}}{(1+f_x^{\,2})^{\frac{1}{2}}}$ and $\hat{k}\approx f_{xx}$.

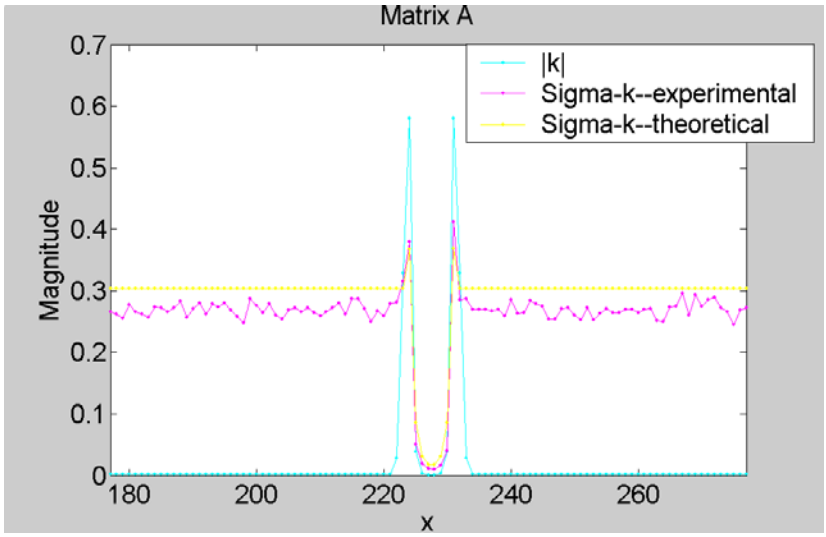
Due to the existence of $f_x$ in the denominator of $k$, the peak value of $k$ is usually much smaller than that of $\hat{k}$. On the other hand, due to image noise, there are fluctuations in the estimations of $k$ and $\hat{k}$. It is proved in the previous deduction that

$Var(\Delta k)\approx\dfrac{1}{(1+(f_x)^2)^3}Var(\Delta f_{xx})+\dfrac{9(f_{xx}f_x)^2}{(1+(f_x)^2)^5}Var(\Delta f_x)$ or $\dfrac{1}{(1+f_x^{\,2})^{\frac{1}{2}}}Var(\Delta f_{yy})$ and

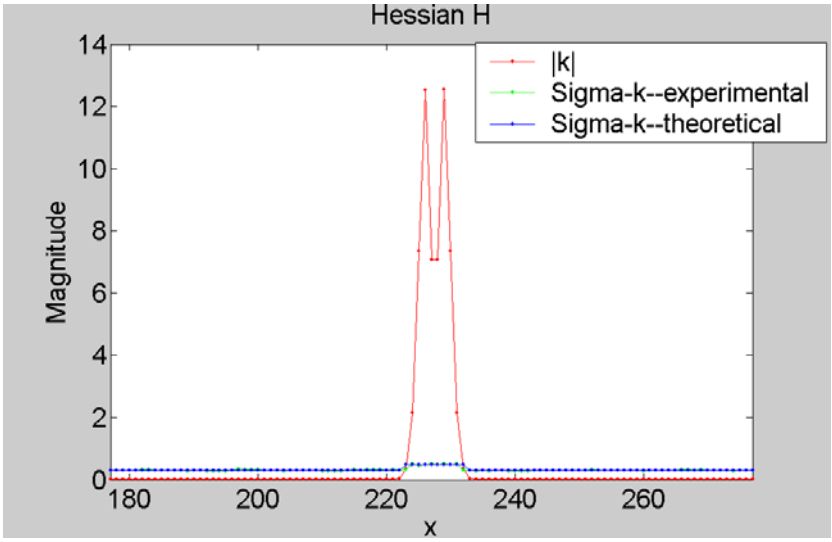
$Var(\Delta\hat{k})\approx Var(\Delta f_{xx})$ around the edge region; while $Var(|\Delta k|)\approx Var(|\Delta\hat{k}|)=\dfrac{\sigma_n^{\,2}}{4\sigma_m^{\,6}}\{\frac{1}{\pi}-\frac{1}{\pi^2}-\frac{1}{8}\}$ around smooth regions. With $|\hat{k}|_{peak}>>|k|_{peak}$ and $var(|\Delta\hat{k}|)\approx var(|\Delta k|)$, the Hessian **H** does provide a better SNR performance than **A**.

All these formulae have been further verified via computer simulations. Fig. 3.4(e) and (f) illustrate 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. 3.4(a). Fig. 3.4(e) shows

expected $|k|$ profile without noise interference (cyan) and the standard deviation of experimental and theoretical $|\Delta k|$ (pink and yellow), based on Matrix **A**. Fig. 3.4(f) shows expected $|\hat{k}|$ profile without noise interference (red) and the experimental and theoretical standard deviation of $|\Delta \hat{k}|$ (green and blue), based on Hessian **H**. In this simulation, $\sigma_m = 1$. We can see from Fig. 3.5(a) and 3.5(b) that the produced noise variances using matrix **A** and Hessian **H** are similar, while the signal component using **H** is much higher than using **A**. Experimental results with different parameters are shown in Fig. 3.6 and Fig. 3.7. These figures verify that the SNR performance of **H** is superior to that of **A**.

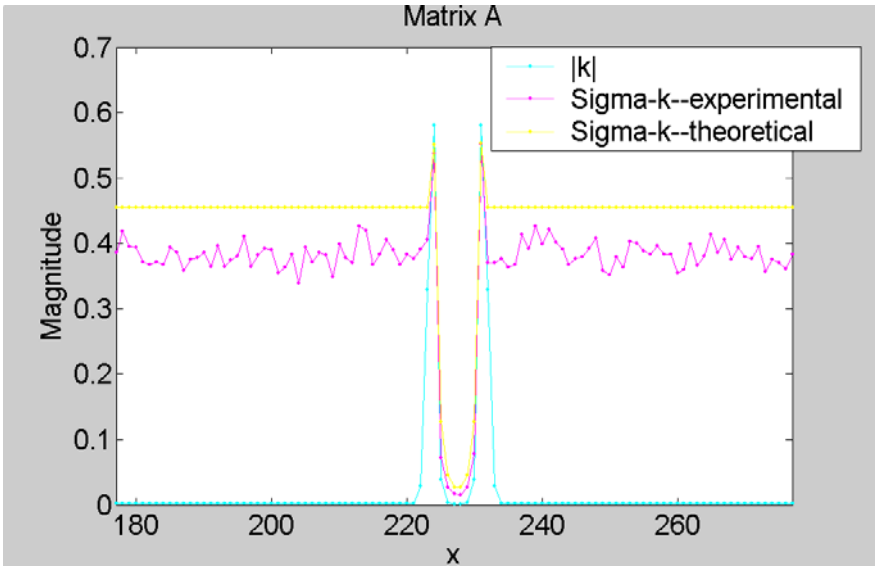



(a)

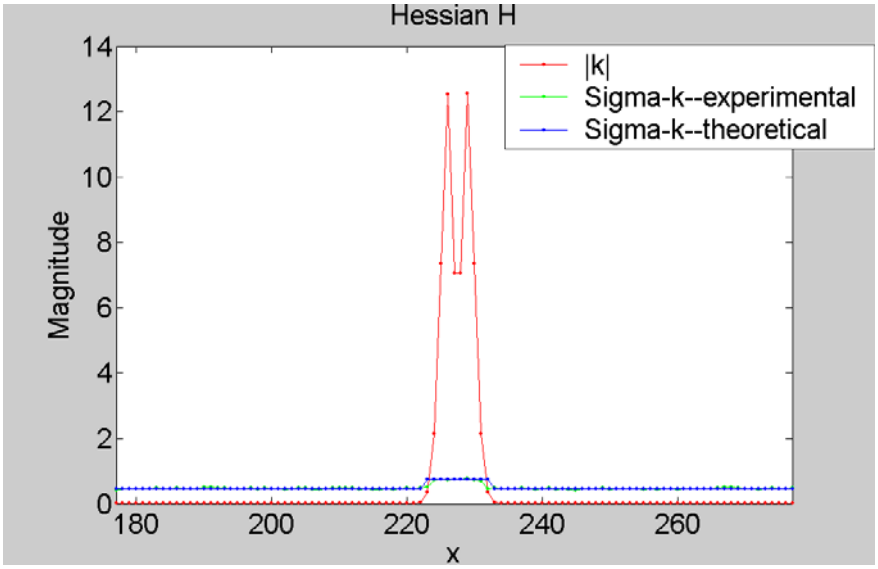



(b)

Fig. 3.5. (a) Expected $|k|$ profile without noise interference (cyan) and the standard deviation of $|\Delta k|$ (pink and yellow), based on Matrix **A**. (b) Expected $|\hat{k}|$ profile without noise interference (red) and the standard deviation of $|\Delta\hat{k}|$ (green and blue), based on Hessian **H**. (Remark: $\sigma_m = 1, \sigma_n = 2$ for this simulation).

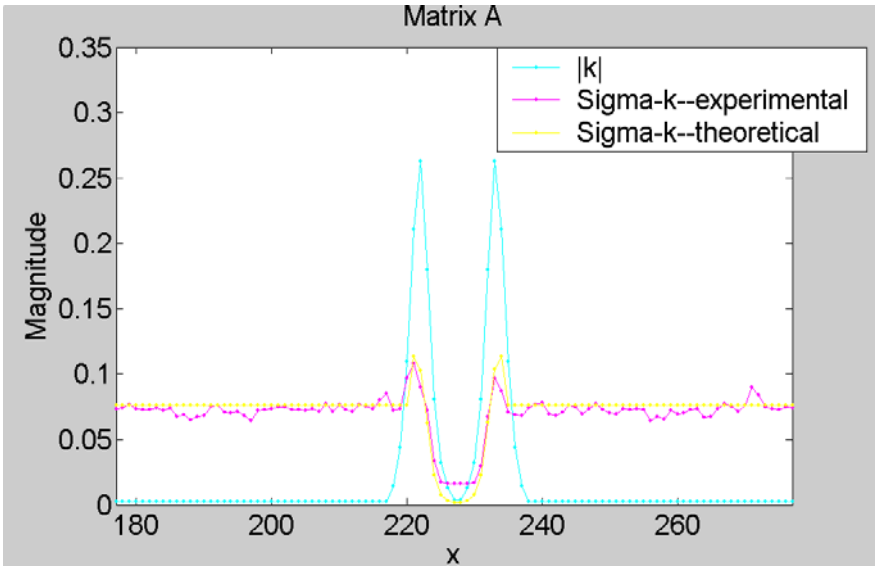



(a)

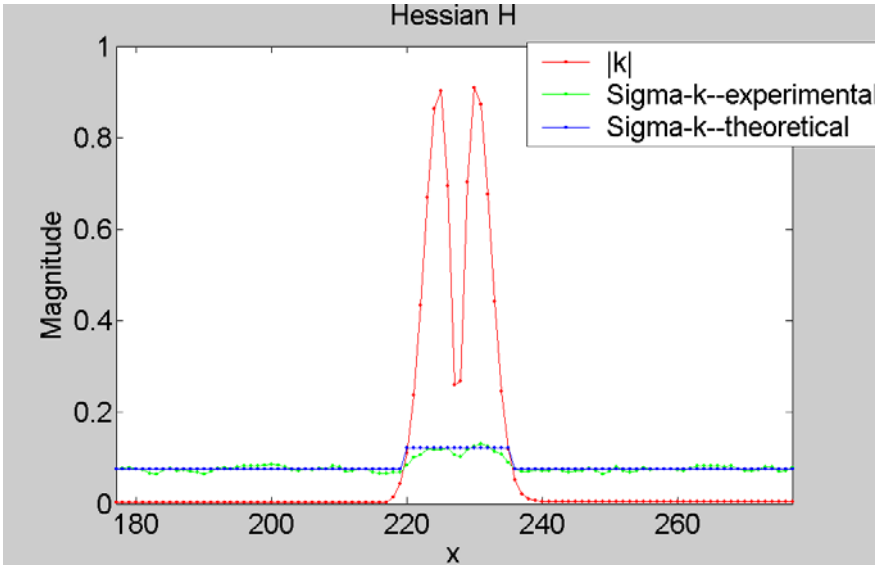



(b)

Fig. 3.6. (a) Expected $|k|$ profile without noise interference (cyan) and the standard deviation of $|\Delta k|$ (pink and yellow), based on Matrix **A**. (b) Expected $|\hat{k}|$ profile without noise interference (red) and the standard deviation of $|\Delta\hat{k}|$ (green and blue), based on Hessian **H**. (Remark: $\sigma_m = 1, \sigma_n = 3$ for this simulation).




(a)




(b)

Fig. 3.7. (a) Expected $|k|$ profile without noise interference (cyan) and the standard deviation of $|\Delta k|$ (pink and yellow), based on Matrix **A**. (b) Expected $|\hat{k}|$ profile without noise interference (red) and the standard deviation of $|\Delta\hat{k}|$ (green and blue), based on Hessian **H**. (Remark: $\sigma_m = 2$ , $\sigma_n = 3$ for this simulation).

To understand more about SNR comparisons between curvature matrix A and Hessian tensor H, we perform experiments using different parameters. The experimental results are shown in Table. 3.1, where $\sigma_m$ indicates the standard deviation of the Gaussian smoothing operator, $\sigma_n$ indicates the standard deviation of AWGN noise, and h indicates the edge contrast.

**Table 3.1 SNR comparisons between A and H.**

| Parameter | Region | SNR Hessian tensor H | SNR Matrix A | Ratio SNR(H)/SNR(A) |
|---|---|---|---|---|
| h=50,$\sigma_m$=1, $\sigma_n$=1 | Edge region | 25.64 | 2.32 | 11.03 |
| | Smooth region | 41.30 | 4.51 | 9.14 |
| h=150,$\sigma_m$=1, $\sigma_n$=1 | Edge region | 76.94 | 4.09 | 18.18 |
| | Smooth region | 123.95 | 2.52 | 49.00 |
| h=250,$\sigma_m$=1, $\sigma_n$=1 | Edge region | 128.24 | 7.53 | 17.01 |
| | Smooth region | 206.60 | 1.17 | 176.22 |
| h=50,$\sigma_m$=1, $\sigma_n$=2 | Edge region | 12.82 | 2.70 | 4.74 |
| | Smooth region | 20.65 | 2.25 | 9.14 |
| h=150,$\sigma_m$=1, $\sigma_n$=2 | Edge region | 38.47 | 1.50 | 25.61 |
| | Smooth region | 61.97 | 1.26 | 49.00 |
| h=250,$\sigma_m$=1, $\sigma_n$=2 | Edge region | 64.12 | 3.52 | 18.20 |
| | Smooth region | 103.30 | 0.58 | 176.22 |

The previous discussion is based on the assumption that noise is additive white Gaussian noise. For color noise or non-Gaussian-distributed noise, the derivations for fluctuation mean and fluctuation variance are more complicated. However, we can still analyze the effects caused by noise based on the previous discussion. On the other hand, for impulse noise, such as salt-and-pepper noise, the interference influence is difficult

to analyze. In this case, it would be difficult to distinguish impulse noise from point-like signals. Nevertheless, we may use median filters to remove impulse noise beforehand, and then perform the previous analyses.

In addition to Matrix **A** and Hessian **H**, there exist some other methods for curvature estimation. For example, in [40], a parameterized curvilinear model is used and curvatures are estimated by minimizing the residual energy measured along the model gradient. In [41], an n-dimensional estimator is proposed to estimate curvatures in images of higher dimensions. Hence, the curvature estimation method adopted in this thesis can be considered as an independent module and can be replaced by alternative curvature estimators.

### 3.2.3. Curvature Threshold Determination

To obtain the verge points of an image f(x,y), we check at each pixel the value of $\left|\hat{k}_1(x,y)\right|$. If $\left|\hat{k}_1(x,y)\right|$ 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 $T_k$ to remove them. The use of a larger $T_k$ suppresses more false alarms, but at the same time suppresses the detection of many image details. On the contrary, the use of a smaller $T_k$ 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. The detail is to be described as follows.

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 $\left|\hat{k}_1(x,y)\right|$ exceeds $T_k$, a candidate of verge point is mistakenly detected. Based on the formulae deduced in Section 3.2.2, we have

$E[|\hat{k}_1|] = \frac{1}{2}(\frac{1}{2\sqrt{2}} + \frac{1}{\pi})\frac{\sigma_n}{\sigma_m^3}$ and $Var[|\hat{k}_1|] \equiv \sigma_{k_1}^2 = (\frac{1}{\pi} - \frac{1}{\pi^2} - \frac{1}{8})\frac{\sigma_n^2}{4\sigma_m^6}$. Hence, once the noise power $\sigma_n^2$ in the image can be estimated, $T_k$ can be chosen to be

$$T_k = E[|\hat{k}_1|] + m\sigma_{k_1} . \tag{3.46}$$

Here, m is to be determined by the user. A typical choice of m is 2 or 3.

To estimate $\sigma_n$, there exist several noise estimation methods in the literature [42]-[44]. There are three common used estimation methods.

(a) Eliminate the structure is an image. Then used the filtered image to estimate noise. For example, Smoothen an image using an average or medium filter first, and subtract the smoothed image from the original image.

(b) Use a noise distribution model.

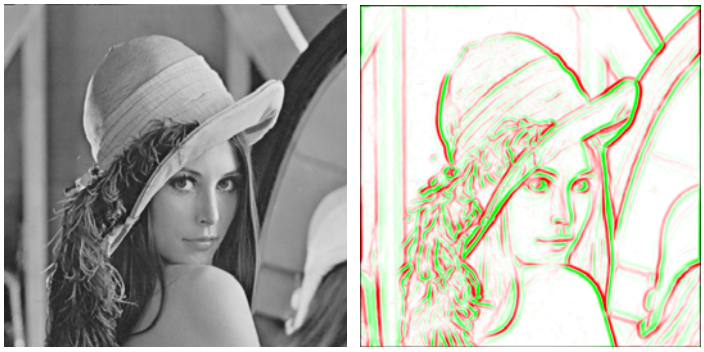
(c) Identify regions with less structures. Calculate the $\sigma_n$ of these smooth regions.

In [42], 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 $\sigma_n$. 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 $\sigma_n$ can thus be deduced. In [43], the authors proposed a method to tessellate an image into blocks, and use the hierarchy of the tessellation to form a variance pyramid. Retaining the 4 smallest variances in each hierarchy, a variance sequence of estimate is calculated based on a slippage test to remove outliers. The variance of noise is obtained by detecting the place where the deviation sequence $\alpha_d(l)$ become significant negative. The deviation sequence $\alpha_d(l)$ is defined as

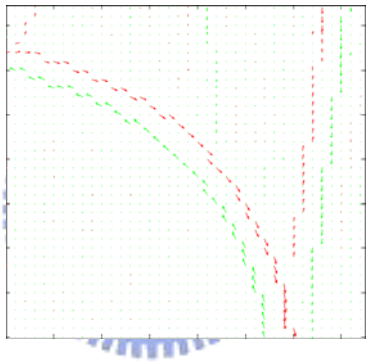
$$\alpha_d(l) = \frac{v_d(l-1)}{v_d(l)} - \beta_d(l), \quad l = 3,4,...n, \tag{3.47}$$

where $\beta_d(l)$ indicates the ration lower bound defined as

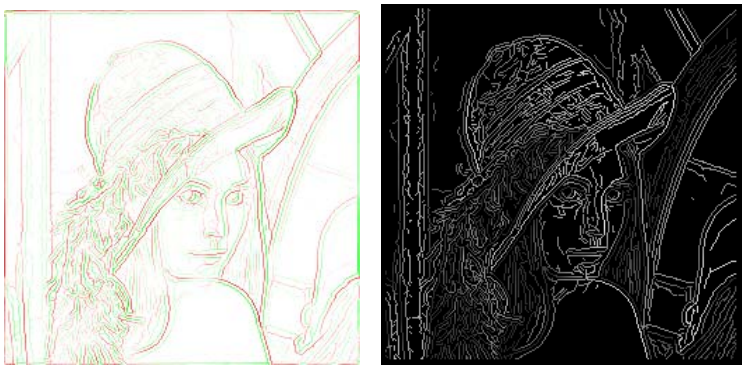
$$\beta_d(l) = 1 - 0.1 \times 2^{-l+6}. \tag{3.48}$$



(a) (b)



(c)



(d) (e)

Fig. 3.8. (a) Original image (256 by 256). (b) $\hat{k}_1(x,y)$ calculated from **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 points represented in terms of intensity values.

In this paper, we implement both methods proposed in [42][43]. For the 256×256 Lena image used in this thesis, $\sigma_n$ is estimated to be 2.1 using [42] and 2.3 using [43]. Here, we set $\sigma_n$ to be the average; that is, $\sigma_n$ =2.2. If we choose $\sigma_m$ = 1, then the threshold is set to be $T_k$ = 2.0 if m = 2, or $T_k$ = 2.6 if m = 3. Fig. 3.8(a) shows a 256×256 Lena image. Fig. 3.8(b) shows the $\hat{k}_1(x, y)$ calculated from **H**, with positive values in red and negative values in green. Fig. 3.8(c) shows the directions of $\Lambda_2(x, y)$ over the shoulder of Lena. Fig. 3.8(d) shows the extracted verge points after thresholding. Fig. 3.8(e) shows the extracted verge points in term of their intensities. In this simulation, we choose $\sigma_m$ = 1 and $T_k$ = 2.0.

Besides the use of a global curvature threshold as aforementioned, the curvature threshold may also be determined adaptively. For areas with larger signal strength, the curvature threshold can be set to be a larger value to suppress more noise interference. On the contrary, for areas with weaker signal strength, the curvature threshold could be set to a smaller value to prevent excessive missing detection. To set threshold adaptively, the constant m used in (3.46) may be adjusted accordingly. Moreover, the area signal strength may be estimated at each pixel by checking the mean and standard deviation in the neighborhood of that pixel.

# 3.3. Image Surface Reconstruction

## 3.3.1. Surface Reconstruction for Gray-Scale Images

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 non-verge-curve parts of the image surface can be imagined as smooth patches stretched by these 3-D pipes. Assume the set of the positions of

extracted verge points is expressed as $\Omega_v$. To reconstruct the original image surface $S_r$ based on $\Omega_v$, one needs to find a mapping function W which yields

$$S_r=W(\Omega_v). \tag{3.49}$$

There may exist different mapping functions to interpolate the non-verge-curve parts of the image surface. In this thesis, we aim for a simple interpolation approach for the reconstruction of image surface based on these extracted verge points. Here, we reconstruct image surface based on the emulation of pipes and rubber cloth mentioned previously. For an element on the stretched cloth, the nearby surface would be influence by pipes which surround this element. Meanwhile, the influence on the given element from a neighboring pipe is inversely proportional to the distance between the element and the pipe. The interpolated intensity value at a target pixel may thus be computed as:

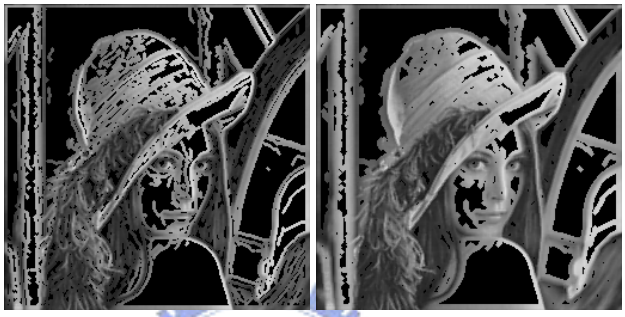
$$I_r = \frac{\sum w_i \mathrm{I}_i}{\sum w_i}, \quad \text{where } w_i = 1/d_i \ . \tag{3.50}$$

In (3.49), $I_r$ 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) to estimate the influence of surrounding pipes; and $w_i$'s denote the weightings that are inversely proportional to the distance between $p_i$ and the target pixel. In Fig. 3.9(a), we show the reconstructed image by linearly interpolating the intensity values of the verge points in Fig. 3.8(e) to fill in the non-verge-curve parts of the image. This linear interpolation is fast in computation but may generate a less smooth image.

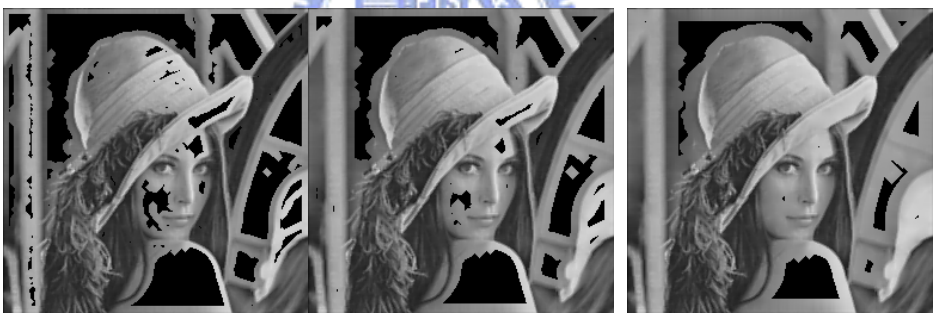
To achieve a smooth reconstruction, the iterative interpolation scheme proposed by Itoh is adopted [45]. With Itoh's approach, the pixels next to these verge points are linearly interpolated first, using the same equation in (3.50). These newly interpolated

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. 3.9(b)-(f), and the final result is shown in Fig. 3.9(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. 3.9(h). Most differences appear around edges or lines. Since human vision is less sensitive to high-frequency change (see the contrast sensitivity function plotted in Fig. 3.9(i)), these differences may have less impact on visual quality. 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. The image surface comparisons of different parts for the original image and the reconstructed image are shown in Fig. 3.9 (j)-(m). Another example of gray-scale image reconstruction is shown in Fig. 3.10.

(a)

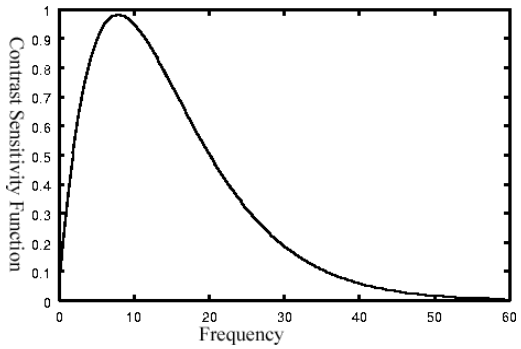


(b) (c)

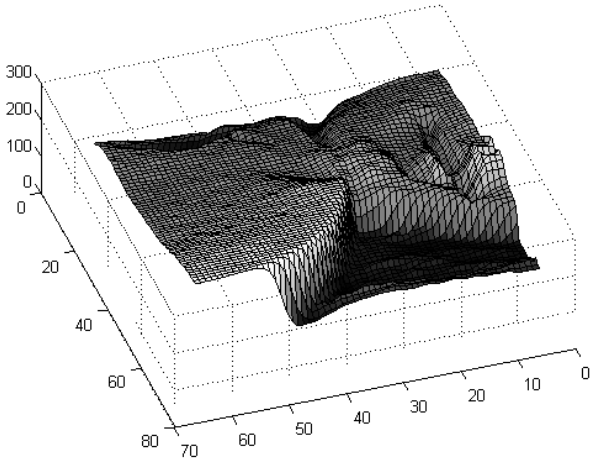


(d) (e) (f)

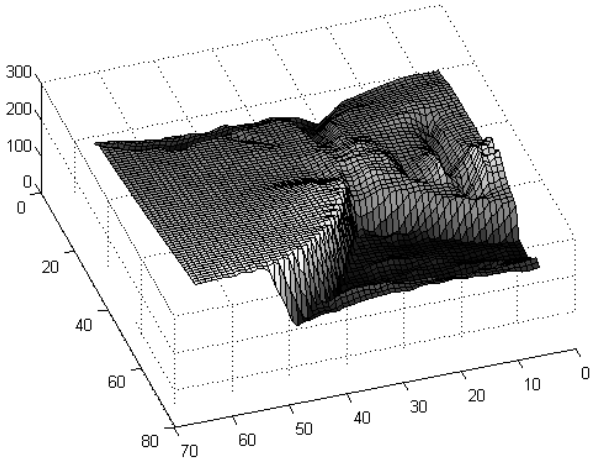
(g) (h)


Contrast Sensitivity Function
Frequency
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60

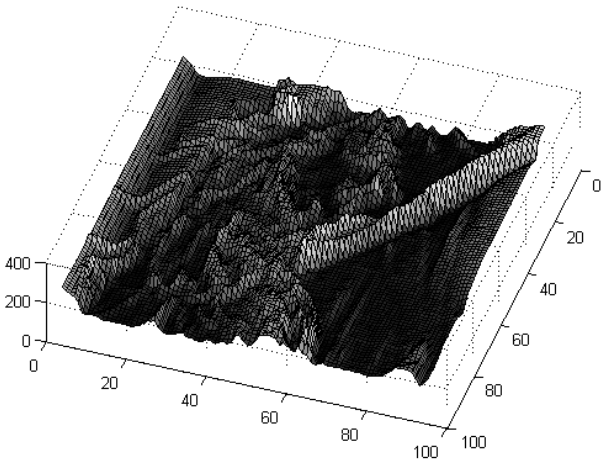
(i)


300
200
100
0
0
20
40
60
80
70
60
50
40
30
20
10
0

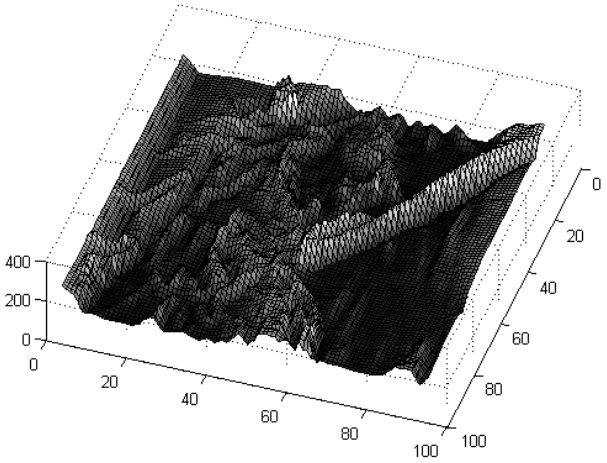
(j)


300
200
100
0
0
20
40
60
80
70
60
50
40
30
20
10
0

(k)

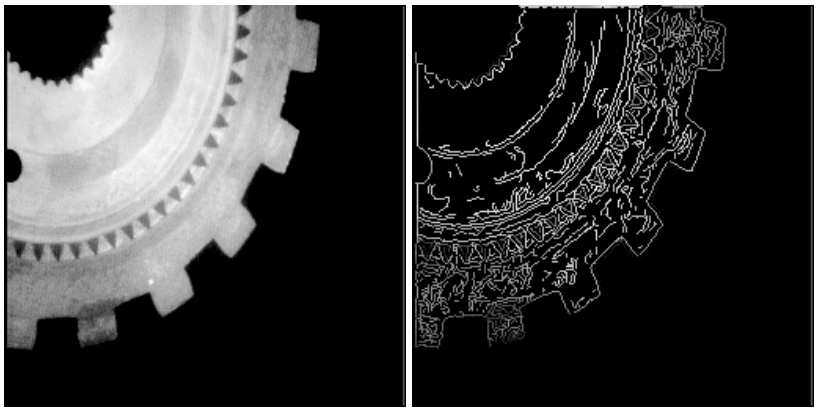



(l)




(m)

Fig. 3.9. (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). (i) Contrast sensitivity function of human visual system. (j) Original image surface around shoulder. (k) Reconstructed image surface around shoulder. (l) Original image surface around hair. (m) Reconstructed image surface around hair.



(a) (b)

(c)

Fig. 3.10. (a) Original image. (b) Extracted verge points. (c) Reconstructed image.

Several methods, such as linear interpolation, bi-linear interpolation, and spline interpolation, have been widely used to perform image interpolation based on uniform-grid samplings. If the bi-linear interpolation, which can be written in the form of ax+by+cxy+d, is adopted to perform image reconstruction from verge points, we have to find 4 nearby verge points to calculate these four parameters. For a higher-order interpolation scheme, we need more verge points to find the associated parameters. Since verge points are irregularly distributed, it would become very difficult to perform these types of reconstruction schemes. In this thesis, we aim at a simple, low complexity way to interpolate non-verge points based on irregularly distributed verge points. Therefore, it seems the iterative linear interpolation is a reasonable and efficient way to produce smooth and low-complexity interpolation.

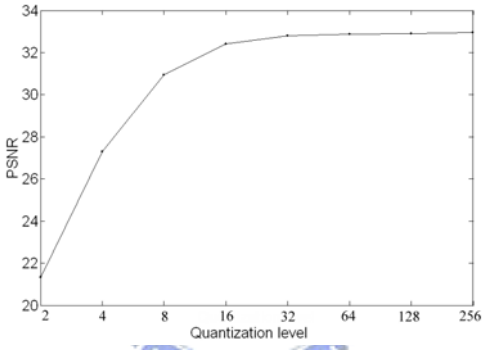
Conceptually, even for the simple linear interpolation case, the selection of neighboring verge points for interpolation should depend on the direction of verge curve. A more accurate way is to perform the interpolation process along the direction perpendicular to the verge curve, that is, according to the orientation of the corresponding planar surface. 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 required execution time for each process incorporated in the extraction of verge points and the reconstruction of image surface is listed in Table 3.2.

**Table 3.2** (Test images: Lena, Fruit, Peppers.)

| Execution Time / Procedure | | Lena | Fruit | Peppers |
|---|---|---|---|---|
| Derivatives calculation ($f_{xx}$, $f_{yy}$, $f_{xy}$) | | 0.1s | 0.1s | 0.1s |
| Eigen-values and eigen-vectors calculation | | 0.1s | 0.1s | 0.1s |
| Curvature threshold | Noise distribution model | 0.2s | 0.2s | 0.2s |
| | Smooth region identification | 3.7s | 3.4s | 3.5s |
| Reconstruction | Direct linear interpolation | 0.55s | 0.48s | 0.51s |
| | Iterative linear interpolation | 0.83s | 0.79s | 0.80s |

(a)




(b)

Fig. 3.11. (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.

The intensity values of these verge points can be further quantized without producing significant impact on the quality of reconstructed images. In Fig. 3.11, 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. 3.11(a). The relation between the quantization level and PSNR of these reconstructed images are shown in Fig. 3.11(b). For an 8-bit gray-level image, $I_{max}$ = 255. As shown in Fig. 3.11(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.

### 3.3.2. Surface 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 L*a*b* as the color space to demonstrate the feasibility of color image reconstruction. In this color space, L* represents the achromatic component, while a* and b* represent the chromatic components. To convert a color image from RGB space to CIE L*a*b* space, we have

$$
\begin{aligned}
L^* &= \begin{cases} 116\left(\sqrt[3]{\frac{Y}{Y_0}}\right) - 16 & \text{for } \frac{Y}{Yn} > 0.008856 \\ 903.3\left(\frac{Y}{Yn}\right) & \text{otherwise} \end{cases} \\
a^* &= 500\ \left( f\left(\frac{X}{Xn}\right) - f\left(\frac{Y}{Yn}\right) \right) \\
b^* &= 200\ \left( f\left(\frac{Y}{Yn}\right) - f\left(\frac{Z}{Zn}\right) \right) \\
\text{where}\quad f(t) &= \begin{cases} \sqrt[3]{t} & \text{for } t > 0.008856 \\ 7.787 * t + \frac{16}{116} & \text{otherwise} \end{cases}
\end{aligned}
\qquad . \qquad (3.51)
$$

Here,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412 & 0.358 & 0.189 \\ 0.213 & 0.715 & 0.072 \\ 0.019 & 0.119 & 0.950 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{3.52}$$

and ($X_n$, $Y_n$, $Z_n$) is the XYZ values of reference white [46]. Fig. 3.12(a) shows the original color image. This color image is decomposed into L*, a*, and b* component images first. Then, verge points are extracted for each component image. Fig. 3.12(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. Another example of color image reconstruction is shown in Fig. 3.13.

(a) (b)

Fig. 3.12. (a) Original color image (256×256). (b) Reconstructed color image under CIE L*a*b* color space, using 16812 verge points for L*, 12537 verge points for a*, and 14416 verge points for b*.

(a)

(b)

Fig. 3.13. (a) Original color image (256×256). (b) Reconstructed color image under CIE L*a*b* color space, using 15723 verge points for L*, 11435 verge points for a*, and 13429 verge points for b*.

Choosing L*a*b* color space benefits the verge point representation in two aspects. First, L*a*b* is a uniform color space. Therefore, we do not need to perform extra processing while determining the thresholds for color components. For example, in RGB color space, a MacAdam ellipsis in the G component is larger than that in R and B components. In this case, we may need different treatments on different chroma components to avoid unbalanced color difference. Second, L*a*b* separates

luminance from chroma. This separation helps us to deal with luminance component and chroma components separately. Since human vision is less sensitive to color change than luminance change, we may use less verge points for color components. An example is shown in Fig. 3.14. It can be seen that no apparent defects appear as we reduce the number of verge points for the a* component image and b* component image.

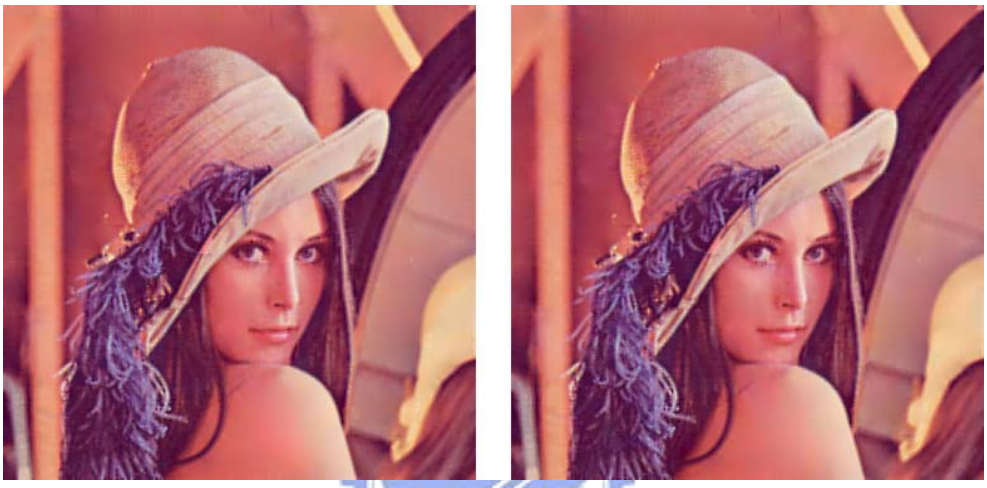


Fig. 3.14. (a) Reconstructed color image under CIE L*a*b* color space, using 34613 verge points for L*, 39961 verge points for a*, and 28452 verge points for b*. (b) Reconstructed color image under CIE L*a*b* color space, using 34613 verge points for L*, 28442 verge points for a*, and 19875 verge points for b*.