Chapter 4

Data Structures Based on Verge Point Representation

In Chapter 3, we present an architecture to represent images using verge points on image surfaces. In this chapter, we describe some data structures used for verge point representation. To improve the compactness of verge point representation, verge points are linked into verge curves under the guidance of the eigen-vectors of the Hessian matrix. To further condense representation data, verge curves are approximated by using B-spline control points. The procedures of verge curves linking and B-spline approximation are presented in Section 4.1 and Section 4.2. In Section 4.3, based on the verge point representation, we propose an image codec to perform image compression. Furthermore, in Section 4.4 and Section 4.5, we show that the proposed verge point representation offers great scalability and interactivity in image transmission. These properties are especially suitable for visual communications.

4.1. Verge Curves Linking

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

In differential geometry, the eigen-vector $\Lambda_2(x,y)$ indicates the direction along which a locally quadratic image surface bends least. For line edges or step edges, $\Lambda_2(x,y)$ points to the possible direction of the next edge element. Fig. 4.1(b) shows the directions of $\Lambda_2(x,y)$ near the shoulder of Lena in Fig 4.1(a). We can see from Fig. 4.1(b) that these eigen-vectors provide useful clues for verge curve linking. To link verge points into verge curves, a two-phase linking scheme similar to the hysteresis approach used in the Canny Operator [47] is adopted. In the first phase, verge points with $|\hat{k}_1|$ larger than T_k are to be linked first. Adjacent verge points at (x,y) and (x',y') are to be linked into curves if they satisfy the constraint on curvature sign S and the constraint on angle difference T_θ; that is, we test whether

$$S(x,y) \times S(x',y') > 0, \tag{4.1}$$

and

$$\|\Lambda_2(x',y') - \Lambda_2(x,y)\| < T_\theta. \tag{4.2}$$

The first constraint means these two verge points need to have the same sign of curvature. The second constraint reflects the requirement for curve shape. The use of a larger T_θ generates long, but sometimes curvy, verge curves. The use of a smaller T_θ generates smooth, but sometimes short, verge curves. The selection of T_θ influences the shape of linked curves, but has little impact on reconstruction quality. In our approach, we prefer smooth linking and set T_θ to be $\pi/8$ empirically. In the second phase of linking process, we extend the endpoints of linked curves to further link these candidates with $|\hat{k}_1|$ smaller than T_k. Here, the threshold T_k is to be determined as mentioned in Section 3.2.3. Similar to the first phase, we still apply the constraints of (4.1) and (4.2) for curve linking.

(a)

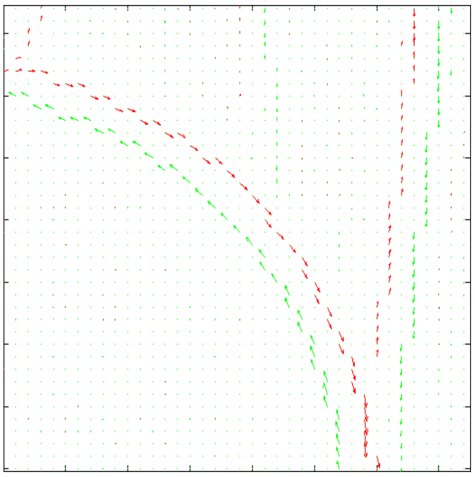

(b)

Fig. 4.1 (a) Original image. (b) Eigenvectors $\Lambda_2(x, y)$ around Lena's shoulder.

4.2. B-spline Curve Approximation

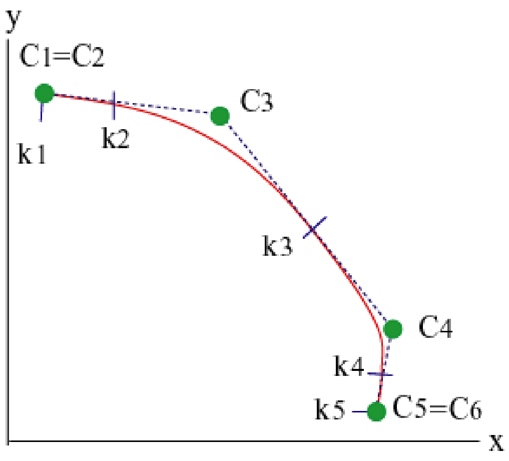
Since the linked verge curves are usually smooth, we can further compress these verge curves by using B-spline approximation [48][49]. With the B-spline approximation, each verge curve can be represented by a small number of control vertices. In addition, this approximation offers high flexibility to control and manipulate verge curves. In theory, a B-spline curve Y(t), with $t_{min} \leq t \leq t_{max}$, can be expressed as

$$Y(t) = \sum_{i=1}^{M} V_i B_{i,k}(t), \qquad 2 \leq k \leq M \quad . \tag{4.3}$$

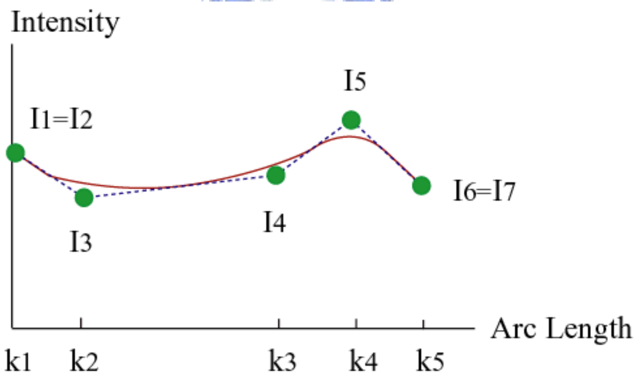
Here, $B_{i,k}(t)$ denotes a set of basis functions of order k, V_i's denote the control vertices, and M denotes the total number of control vertices [48][49]. A practical way to deduce $N_{i,k}(t)$ is to use the Cox-deBoor recursion formula, which is expressed as

$$B_{i,0}(t) = \begin{cases} 1 & if \quad t_i \le t < t_{i+1} \\ 0 & otherwise \end{cases}$$
$$B_{i,k}(t) = \frac{(t-t_i)B_{i,k-1}(t)}{t_{i+k}-t_i} + \frac{(t_{i+k+1}-t)B_{i+1,k-1}(t)}{t_{i+k+1}-t_{i+1}} . \tag{4.4}$$

Here, t_i's denote the elements in a knot vector, which are arranged in a monotonically increasing order. With the B-spline approximation, each verge curve can be represented by a small number of control points.



(a)



(b)

Fig. 4.2. B-spline approximation. (a) Shape component (C: shape control points, k: knots). (b) Intensity component (I: intensity control points, k: knots).

In our approach, each verge curve is decomposed into the shape component and the intensity component, as shown in Fig. 4.2(a) and (b). For the shape component, we adopt the shape coding algorithm proposed in [50]. In [50], a third order uniform

B-spline curve approximation is adopted, and the positions of control points are calculated using a weighted directed acyclic graph. To reconstruct the intensity value in a given verge curve, we use the same knot vectors used for shape coding and calculate the arc length of each knot. For the approximation of intensity component, the horizontal positions of the control points are set as the arc length of each knot. The vertical position of control points are then calculated using least square fitting. Assume a vertical position f(x) is sampled at N samples, x_1, x_2, …, x_N. To approximate the vertical position f(x) using B-spline approximation, we calculate the control vertices V_i's by minimizing the mean square error E between the approximated B-spline curve and the original curve:

$$E = \sum_{j=1}^{N} \{ \sum_{i=1}^{M} V_i B_{i,k}(t_i) - f(x_j) \}^2 \quad . \tag{4.5}$$

In the B-spline representation, these control points are used to represent verge curves. That is, the original list of N verge points is replaced by the list of M control points. Since M is usually much less than N, this representation provides an even more compact form for the original image. On the other hand, since the B-spline representation is only an approximation to the original verge curve, larger distortion is expected in the reconstructed image. However, this distortion can be properly restricted by setting an upper bound over approximation errors. To further clarify the essence of this algorithm, we illustrate in Fig. 4.3 the major phases of the proposed representation method. Fig. 4.3(a) shows the original image and Fig. 4.3(b) shows the corresponding image surface. Fig. 4.3(c) shows the verge curves extracted from the image surface. Fig. 4.3(d) shows the B-spline control vertices computed from the extracted verge curves. Either the verge curves or the B-spline control vertices can be used to represent the original image. The verge-curve representation is more accurate, while the control-vertex representation is more compact. Then, in Fig. 4.3(e), we show the reconstructed image surface based on the B-spline control vertices in Fig. 4.3(d). Finally, Fig. 4.3(f) shows the reconstructed image.

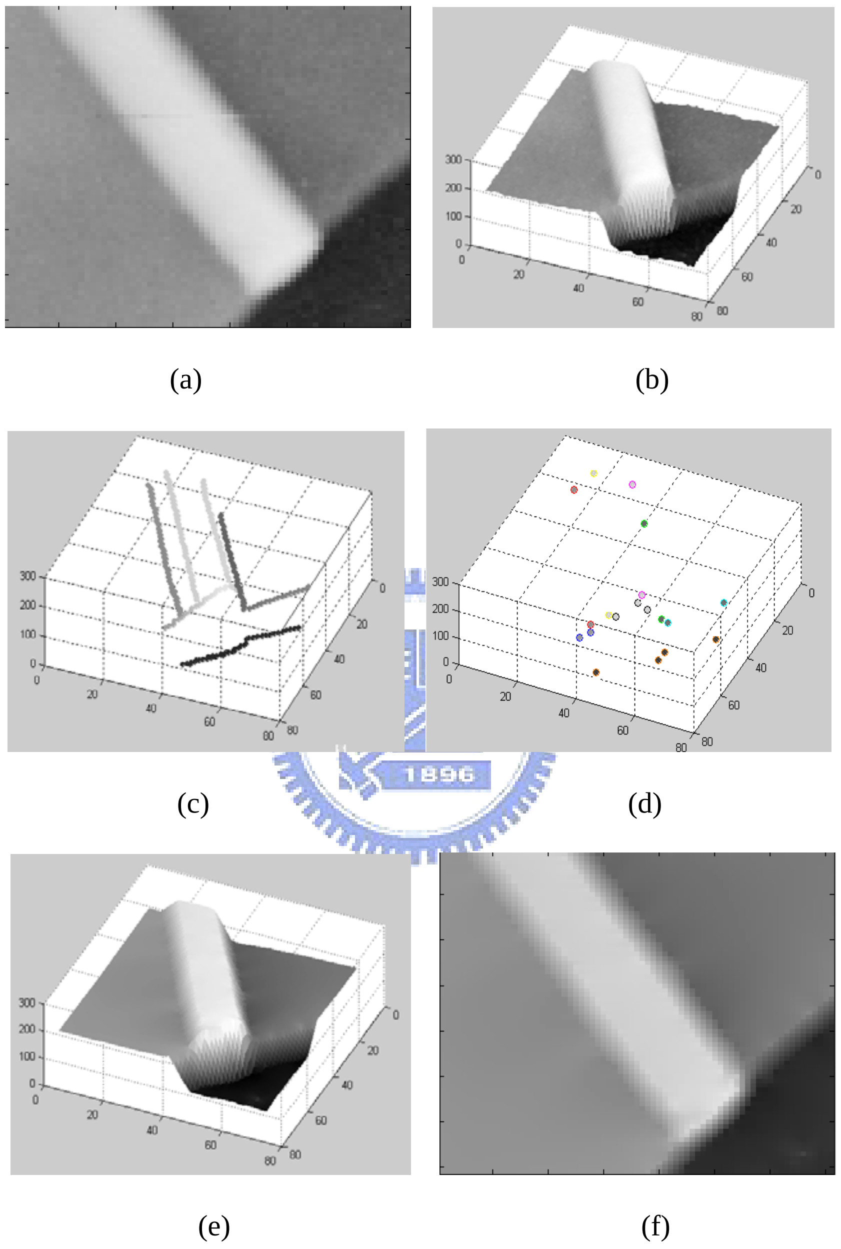
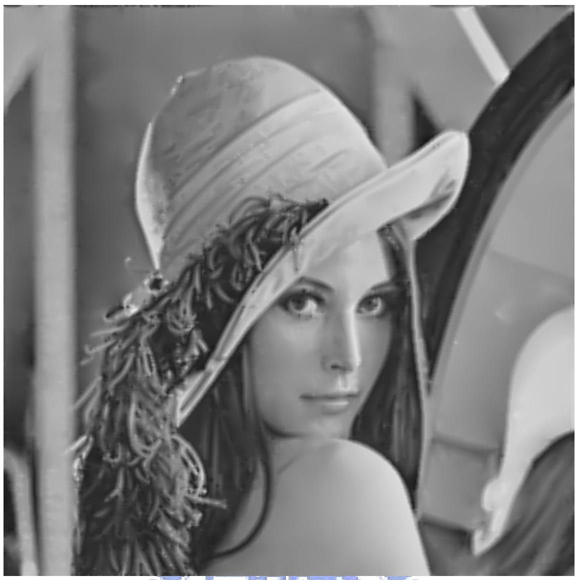
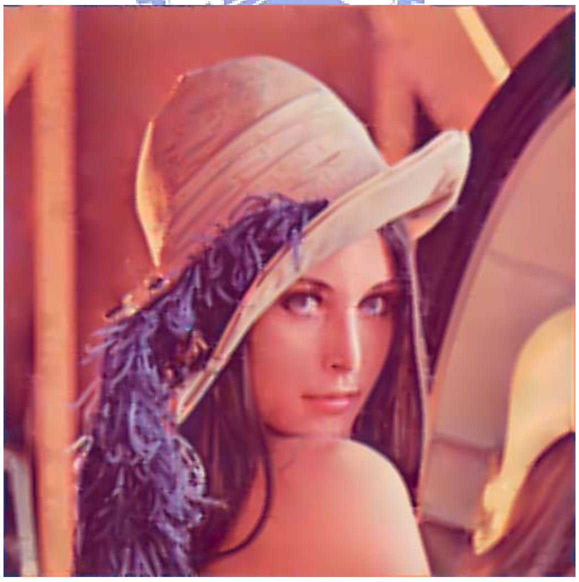


Fig. 4.3. (a) Original image. (b) Original image surface. (c) Extracted verge curves. (d) Control points using B-spline approximation. (e) Reconstructed image surface using (d). (f) Reconstructed image.

Fig. 4.4(a) and (b) show the reconstructed images based on the B-spline representation. These two images are visually similar to the original images. In addition, due to the affine invariant property of B-spline curve, this B-spline representation is expected to be very suitable for spatial image scaling.



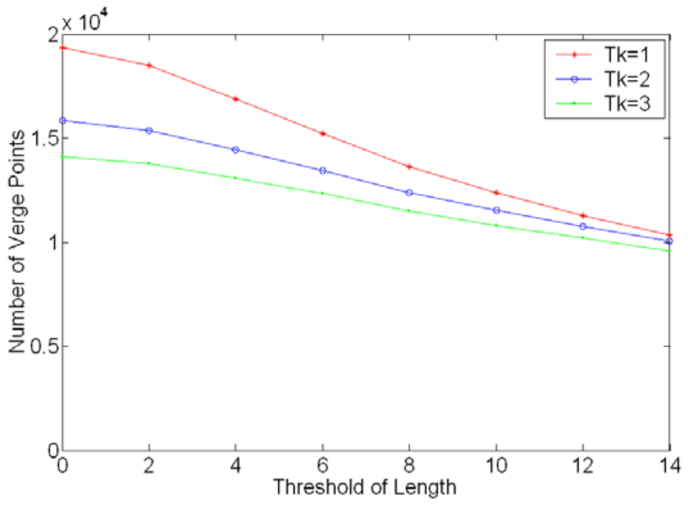
(a)



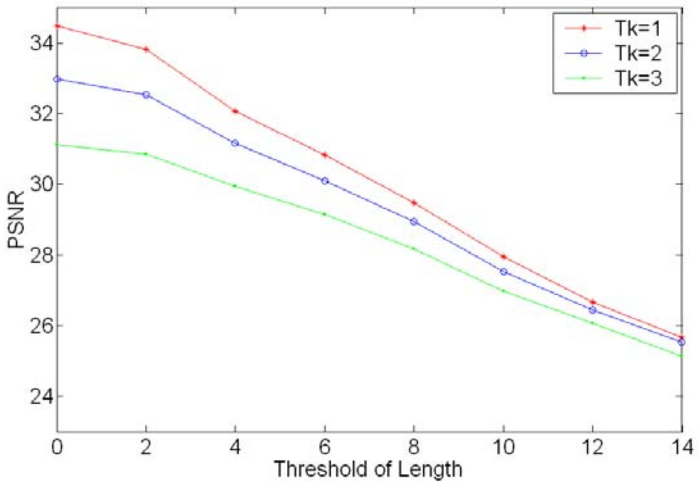
(b)

Fig. 4.4. (a) Reconstructed gray-level image using B-spline approximation. (2957 control points) (b) Reconstructed color image using B-spline approximation under the CIE L*a*b* color space. (L*: 3173 control points, a*: 2783 control points, b*: 2974 control points).

In image reconstruction, different levels of image quality can be achieved by placing different thresholds over the length of verge curve and the magnitude of curvature. Fig. 4.5 presents the PSNR and verge/control-point number with respect to T_{length}, the threshold of curve length, and T_k, the threshold of curvature magnitude. In this simulation, we choose $\sigma_m = 1$. It can be seen in Fig. 4.5(a)-(d) that different levels of image quality can be achieved by adjusting the values of T_{length} and T_k. In addition, while the number of verge points and control vertices drop quickly with an increase in T_{length}, the PSNR decreases with a less acute slope. Moreover, Fig 4.5(e) presents the ratio of the total pixel number over the number of verge points. Fig 4.5(f) presents the ratio of the total pixel number over the number of control points.

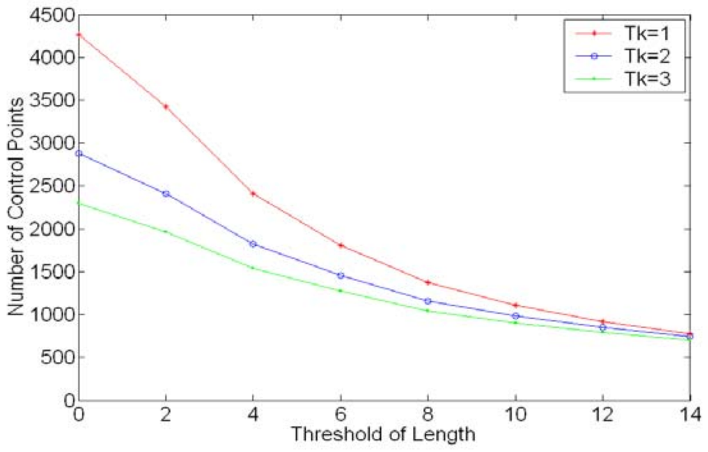


(a)



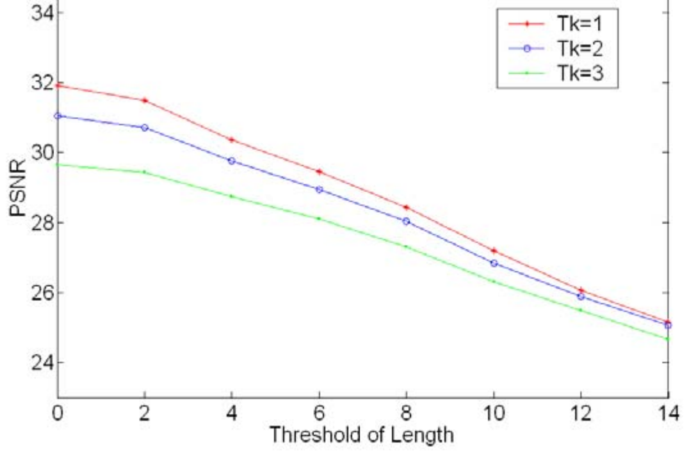
Tk=1
Tk=2
Tk=3
PSNR
Threshold of Length

(b)



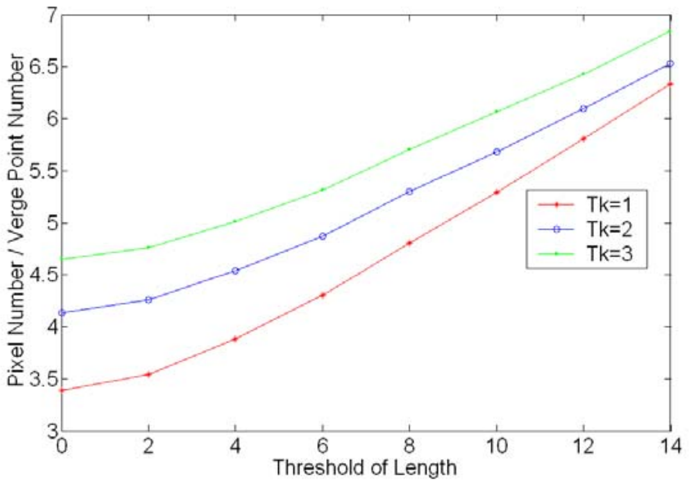
Tk=1
Tk=2
Tk=3
Number of Control Points
Threshold of Length

(c)

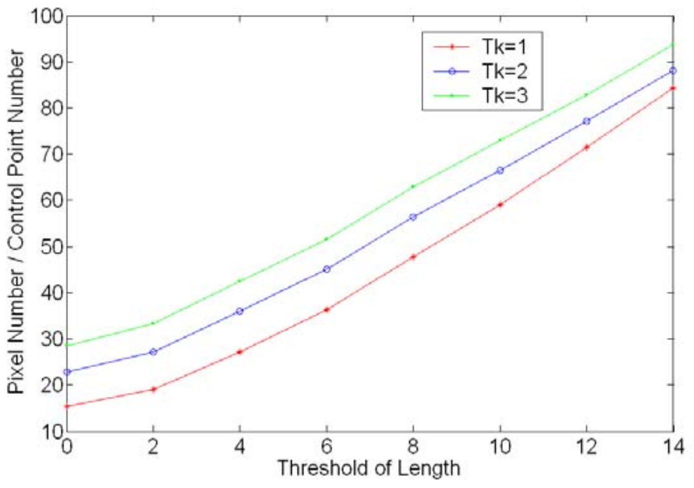


Tk=1
Tk=2
Tk=3
PSNR
Threshold of Length

(d)



(e)



(f)

Fig. 4.5. (a) Number of verge points versus the threshold of curve length. (b) PSNR of the reconstructed images versus the threshold of curve length (using verge-curve representation). (c) Number of control points versus the threshold of curve length. (d) PSNR of the reconstructed image versus the threshold of curve length (using B-spline control-point representation). (e) Ratio of the total pixel number over the number of verge points. (f) Ratio of the total pixel number over the number of control points.

4.3 Image Compression Based on Verge Point Representation

To transmit image data through fixed-rate channels or variable-rate network bandwidth, different image coding approaches have been proposed [51]-[58]. A generic image codec may include the following components.

(a) Feature extraction or transformation: Detect features to in an image, or transform an images into different domain.

(b) Quantization: Quantize original data to strike balance between rate and distortion.

(c) Entropy Coding: Reduce redundancy in symbols after quantization.

Fig. 4.6 shows the block diagram of JPEG encoder. An original is level shifted first, and then DCT transformations are applied for each blocks. The DCT coefficients are quantized using quantization tables. The produced symbols in each block are zigzag scanned and compressed using entropy coding techniques (including run-length coding and variable-length coding).

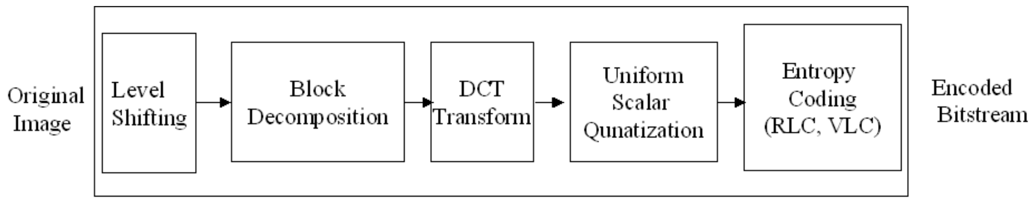


Fig. 4.6. Block diagram of JPEG encoder.

One crucial concern for image compression is to reduce redundancy in the raw image. In this section, we discuss the coding issues based on verge point representation. In addition to coding efficiency, other image coding concerns, such as scalability and interactivity, are also important. In the past decades, the infrastructure and technology of the internet are rapidly developed. In network environment, scalable image coding is useful for combating the fluctuation of the allowed bandwidth. Moreover, for network users, since objects in an image provide more information than the original pixel array, it would be helpful if the objects in an image can be conveniently identified and selected. Based on the verge point representation, we present the data structures for an image coding scheme featured in two aspects, scalability and interactivity. To cope with the fluctuation of network bandwidth, the proposed codec offers different modes of scalability, including spatial scalability, SNR scalability, and shape scalability. To increase the interactivity between the coded data and the network users, the proposed codec provides convenient mechanism to perform region of interest selection and transmission without performing image segmentation. The overall structure of the proposed codec is shown in Fig. 4.7. In the first phase, the three-dimensional verge curves Ω_v are extracted in the image surface, and approximated using a set of B-spline control points Ω_b. By arranging the Ω_b in different hierarchical forms, scalable transmission and region of interest transmission could be achieved. To support interactive transmission, scaling and bitplane shifting factors are incorporated to create different visual effects at the decoder side. Three parameters, curvature threshold T_k, shape distortion allowance A_d, and intensity quantizer Q_i, are included to strike balance between visual quality and required bandwidth.

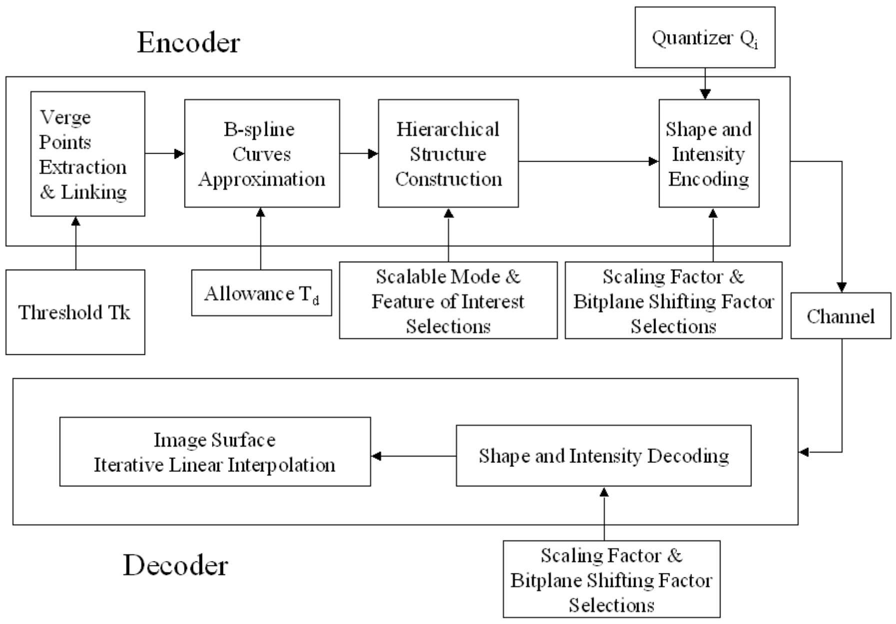

Encoder
Quantizer Q_i
Verge Points Extraction & Linking
B-spline Curves Approximation
Hierarchical Structure Construction
Shape and Intensity Encoding
Threshold Tk
Allowance T_d
Scalable Mode & Feature of Interest Selections
Scaling Factor & Bitplane Shifting Factor Selections
Channel
Image Surface Iterative Linear Interpolation
Shape and Intensity Decoding
Decoder
Scaling Factor & Bitplane Shifting Factor Selections

Fig. 4.7. Block diagram of the proposed image codec.

To describe the proposed image codec, in Section 4.3, the basic data structure for the proposed image codec is presented. The scalability and interactivity of the proposed image codec is introduced in Section 4.4 and Section 4.5, respectively.

4.3.1 Data Arrangement for the Proposed Image Codec

As mentioned above, these extracted verge curves offer an effective way to represent images. A straightforward approach to record these verge curves is a hierarchical data structure as illustrated in Fig. 4.8 At the top layer, the verge curves for the L* component image, a* component image, and b* component image are stored separately. Take the set of L* verge curves as an example; the header information, the total number of verge curves in the L* component image, is recorded first, followed by the sequence of verge curves. For each verge curve, the curvature sign and the total number of linked verge points in this verge curve are recorded first. Then the list of verge points in this verge curve is recorded. For each verge point, its (x, y, f(x,y)) coordinates are recorded as X, Y, and F, respectively.

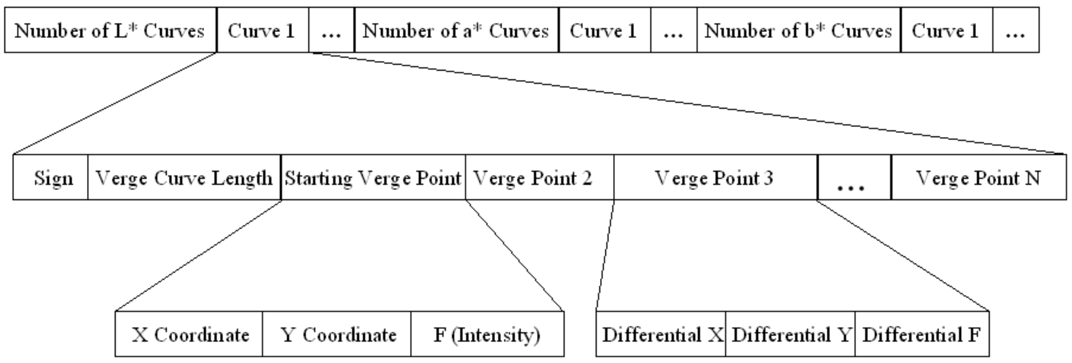


Fig. 4.8. Data structure for the storage of verge curves.

To record these control points, we use a similar hierarchical data structure for verge points, except replacing the verge points with the control points in the middle layer of the hierarchy (see Fig. 4.9).

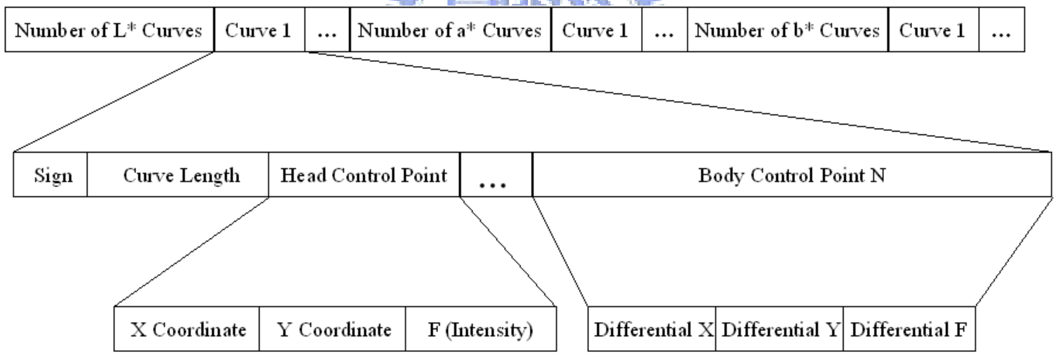


Fig. 4.9. Basic data structure for B-spline representation.

4.3.2 Parameters Selections

To strike the balance between visual quality and required bandwidth, three parameters, T_k, A_d, and Q_i, are included in the proposed scheme. These parameters can be either specified by users or adaptively determined.

4.3.2.1 Curvature Threshold T_k

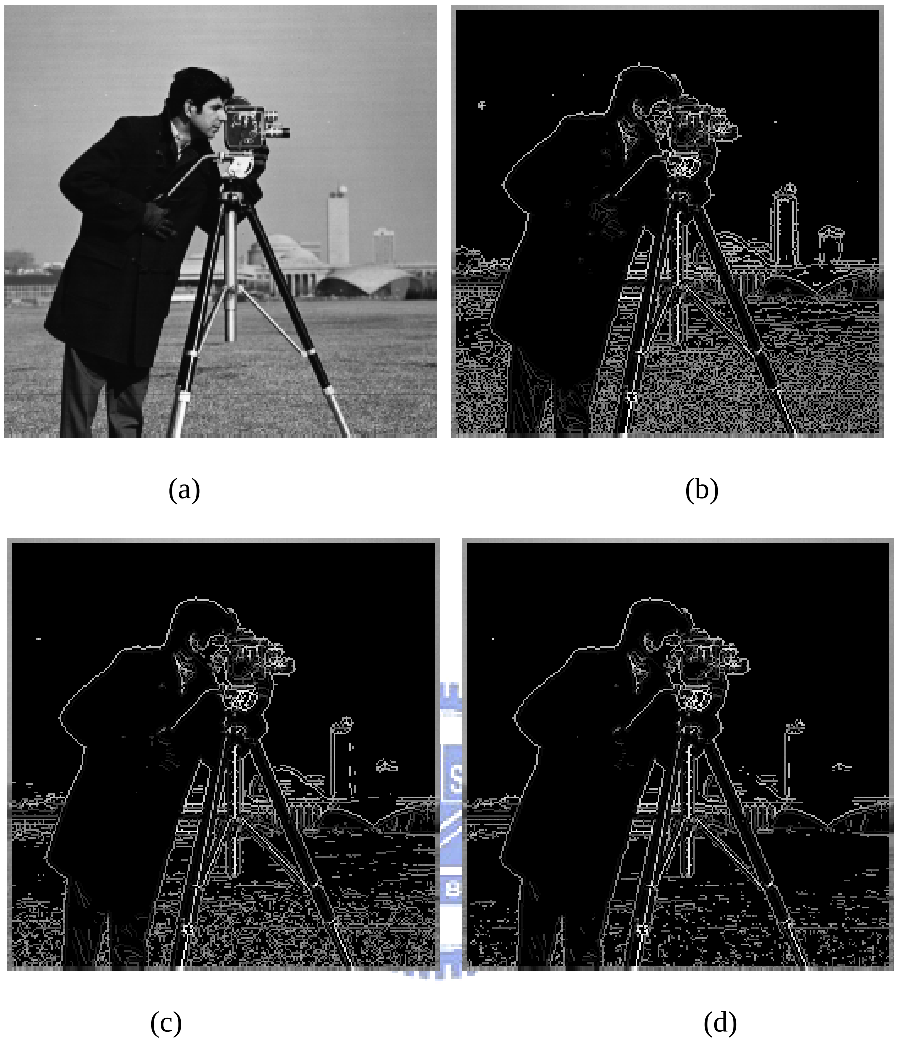


Fig. 4.10. (a) Original image. (b) Extracted verge points with T_k=3. (c) Extracted verge points with T_k=6. (d) Extracted verge points with T_k=9.

In Chapter 3, the curvature threshold T_k for image representation is determined based on noise variance in an image. To deal with image compression, an additional constraint, allowed bandwidth, is considered. Therefore, we discuss the selections of T_k for image coding from the viewpoint of rate-distortion. Here, the curvature threshold T_k, determines how many details would be transmitted to the decoder side. If the allowed bandwidth is wide, a smaller T_k is adopted to capture more details. On the contrary, if only a narrow bandwidth is allowed, a larger T_k needs to be chosen to reduce data amount. Fig. 4.10 (b)-(d) show the extracted verge points with using different T_k's over Fig. 4.10(a). The required verge point numbers and the MSE values between the reconstructed images and the original images are listed in Table 4.1.

Table 4.1 The number of required verge points and the MSE under different T_k's.

	T_k=3	T_k=6	T_k=9
Required Verge Points	12473	9318	7103
Mean Squre Error	47.14	85.84	137.51

If the allowed bandwidth cannot be predicted in advance, we calculate the default T_k based on rate-distortion relation. Let **QS** indicate the allowed curvature set. To determine T_k, we apply the elements in **QS** in an ascending order to obtain the relation between the number of verge points R_e and the mean square error D_e of the reconstructed image. The number of verge points generated by using element qs_{min} is indicated by R_{max}, and the mean square error generated by qs_{max} is indicated by D_{max}. R_e and D_e are normalized by R_{max} and D_{max}, respectively. This relation is used to estimate the rate-distortion curve. Fig. 4.11 shows the estimated rate-distortion curve of the test image "peppers". We can see in Fig. 4.11 that on the left of the knee point, the mean square error D_e increases drastically. On the other hand, on the right of the knee point, it takes a wider rate range to reduce a certain amount of distortion. Therefore, we choose the knee point to be the place where the rate-distortion curve bends most in the estimated R-D diagram. The curvature value which produces the knee point is considered as the default curvature quantizer T_k^{def}. That is, T_k^{def} is defined as

$$T_k \overset{def}{=} \arg\max_{j \in QS} \frac{\left| \frac{d^2 D_e}{dR_e^{\,2}}(j) \right|}{\left(1 + \left(\frac{dD}{dR}(j)\right)^2\right)^{3/2}}, \qquad (4.6)$$

where j indicates a point j on the estimated R-D curve.

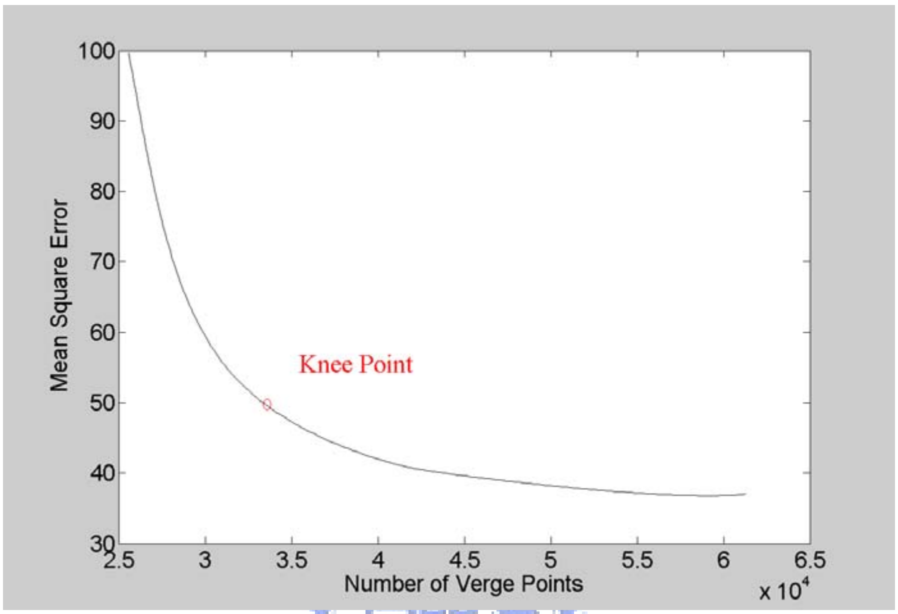


Fig. 4.11. Estimated rate-distortion curve of the image “peppers”.

4.3.2.2. Shape Distortion Allowance A_d

The shape distortion allowance A_d determines the maximally allowed position distortion between the verge curves and the approximated B-spline curves. This parameter trades off between rate and position accuracy. If a larger A_d is applied, the required number of B-spline control points becomes smaller, while the deviation between a verge curve and its approximated B-spline curve becomes larger. On the contrary, using a smaller A_d produces smaller position deviation with the penalty of larger required bandwidth.

The parameter A_d can be specified globally or locally. Table 4.2 shows the relation between the maximum allowed position distortion A_d and the number of required control points for the 256x256 image “lena”. Different features in an image may have different visual impacts. Curves with higher contrast and longer length may provide more apparent visual impact, such as the contour of a highlighted human face with a dark background. On the contrary, curves with smaller contrast and shorter length may cause less visual impact, such as the vague textures on a cloth. Fig. 12(a) and (b) show the reconstructed images with A_d=1 and A_d=3. We can see that the distortion in the face contour is much more apparent than the distortion in the texture of the hat. In addition, an unstructured verge curve, such as long cracks on walls, may allow a larger A_d. For a verge curve, the degree of structure may be detected from the difference between adjacent eigen-vectors Λ_2's.

Table 4.2 Relation between allowed shape distortion Ad and required control points.

	A_d=1	A_d =2	A_d =3
Required Control Points	2473	2299	2093

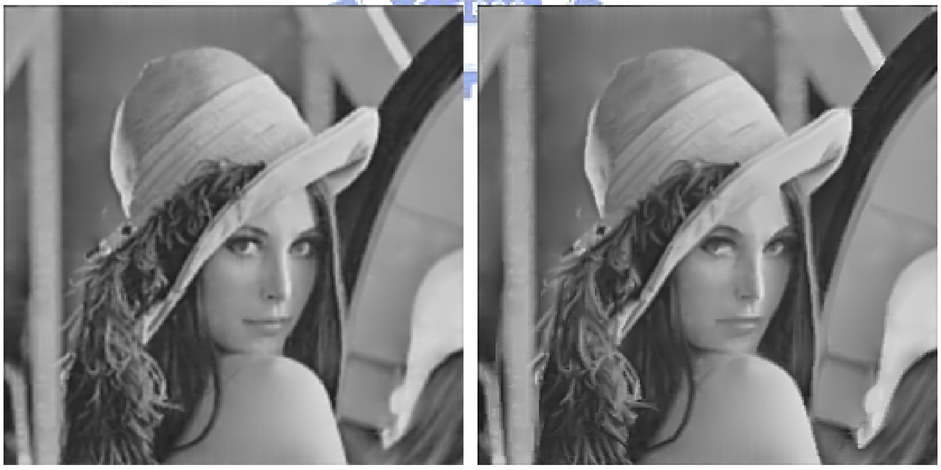


Fig. 4.12. (a) Allowed position distortion A_d =1. (b) Allowed position distortion A_d =3.

One possible selection to adaptively determine A_d is based on the accumulated contrast AC that is defined as the sum of all the curvature magnitudes along a curve. For the j-th curve in Ω_v, its corresponding A_d is defined as

$$A_d(j) = (\frac{AC_{\max}}{AC(j)})^{\alpha} \times A_{d\,const}, \tag{4.7}$$

where AC_{max} indicates the maximum accumulated contrast in Ω_v . The weighting factor α is set as 0.3 empirically, and A_{dconst} is the global distortion parameter.

4.3.2.3 Intensity Qunatization Parameter Q_i

The quantization parameter Q_i controls the intensity accuracy between the original images and the decoded images. In the proposed codec, because the intensity values of the smooth regions are obtained using iterative linear interpolation, the impact of intensity quantization would be less significant as long as Q_i is not too large. To obtain Qi adaptively for different images, we increase Qi step by step, and use the quantized verge curves in Ω_v to reconstruct images. For a control point with intensity level F, the quantized level RF using Qi is defined as

$$RF = 2^{Q_i} \times \left\lfloor \frac{F}{2^{Q_i}} \right\rfloor + \left\lfloor 2^{Q_i - 1} \right\rfloor. \tag{4.8}$$

Fig. 4.13 shows the relation between the quantization factor Q_i and the PSNR value using the quantized verge curves in $\Omega_{v.}$ After passing the knee point in Fig. 4.13, the quality of an image starts to drop drastically. Therefore, we search for the knee point in the diagram to obtain Q_i^{def} . Here, Q_i^{def} is estimated using

$$Q_i^{def} = \arg \max_{j \in [0 \ldots BN-1]} \frac{|PSNR^{+}(j)\text{-}PSNR^{-}(j)|}{PSNR^{+}(j) + PSNR^{-}(j)}, \tag{4.9}$$

where BN indicates the number of intensity bitplanes. $PSNR^{+}$(j) and $PSNR^{-}$(j) indicate the magnitudes of the PSNR differences on the right-hand side and the left-hand side of bitplane j.

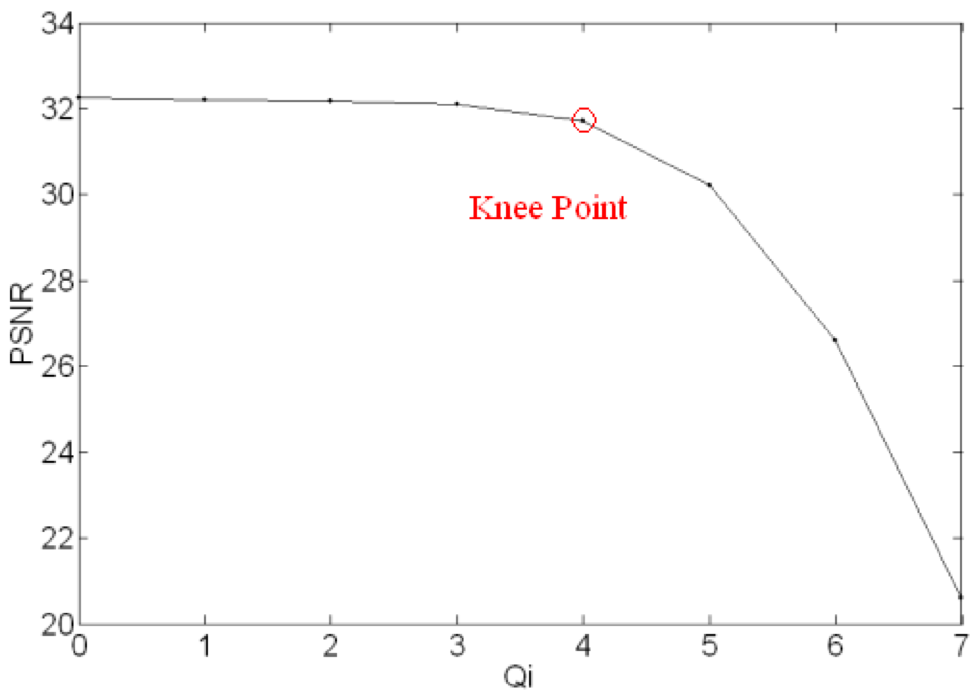
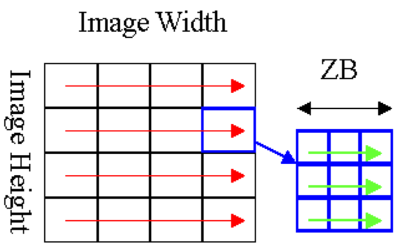
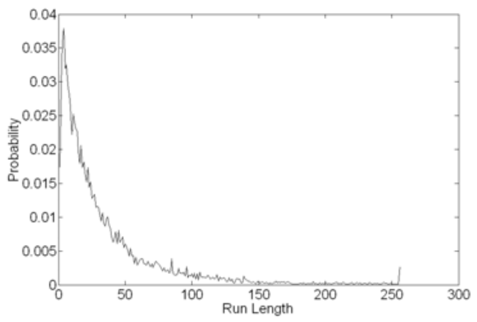


Fig. 4.13. Adaptive Q_i determination for the image “peppers”.

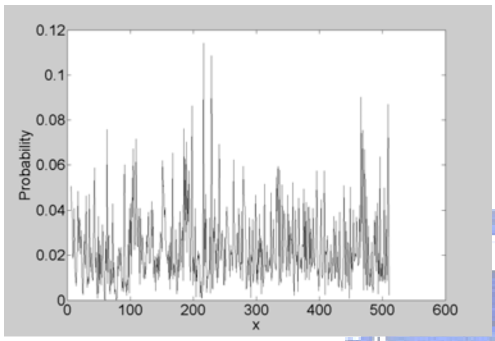
4.3.3. Entropy Coding

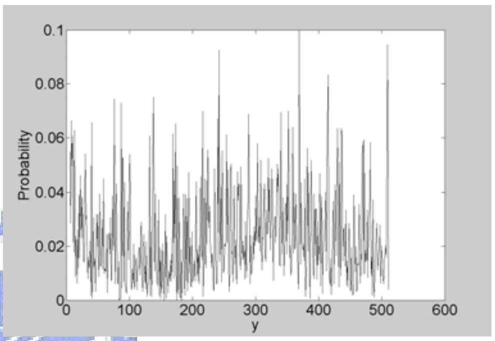
As indicated previously, for each verge curve in B-spline form, we code the starting control point and the body control points separately. To scan the starting points in an image, a zigzag scan may not be needed since the probabilities where the starting points occur are approximately equal. Therefore, the starting control points are encoded using a two-level sequential scan as show in Fig. 4.14(a), where ZB indicates the dimension of a basic scan cell. IW and IL indicate the image width and length. We perform inter-cell sequential scan for the $\frac{IW \times IL}{ZB^2}$ scan cells. In each cell, an intra-cell sequential scan is used to find the run length of two nearby starting points. The statistics of the run length of three 512x512 images, “lena”, “fruit” and “peppers”, is shown in Fig. 4.14 (b), where ZB is set as 16 empirically. Compared with the approach which directly records the positions of x and y (see Fig. 4.14 (c) and Fig. 4.14(d), the two-level zigzag scan combines the two variables (x,y) into one (run length), and reduces the number of required symbols.





(a) (b)





(c) (d)

Fig. 4.14. (a) Two-level sequential scan. (b) Statistics of run length. (c) Statistics of position x. (d) Statistics of position y

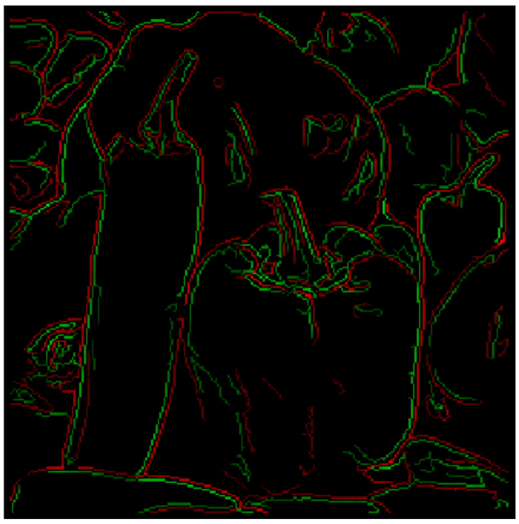
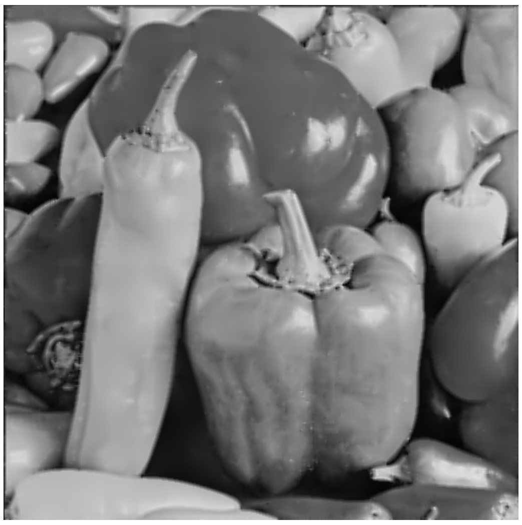
To encode the body control points, we adopt the scheme proposed in [50] with some modifications to improve the coding efficiency. In [50], the distance between two body control points is encoded using polar coordinate, radius R and angle θ. The angle θ is coded using 3 bits to indicate the eight possible directions. The maximum R of the code table used in [50] is 15. This is inefficient in coding straight lines, especially for long ones. As shown in Table 4.3, we reduce the maximum radius to 14 and use the codeword "111111" to indicate a straight line. The absolute position of the tail point of the line is followed immediately after the codeword. To check whether a curve is a straight line, we draw a line from the head point to the tail point of each curve. If the distortion between the curve and the locus of this line is less than Q_d, the curve is considered as a straight line. This curve is coded using linear mode.

For intensity component, the quantized intensity value of the starting point is encoded using its absolute value, while the intensity values of the body control points are coded differentially.

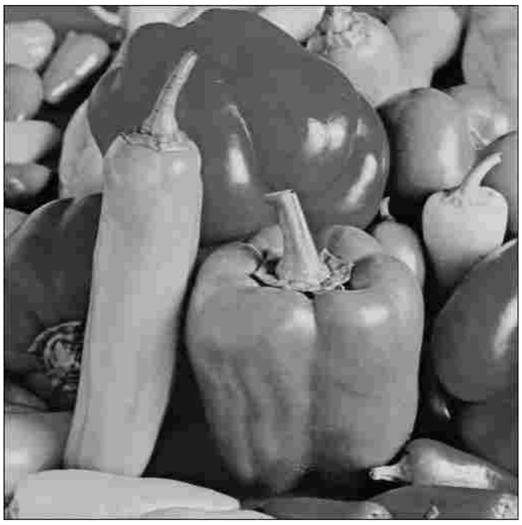
Table 4.3. Code words for body control points.

Radius	Code Word	Rate	Radius	Code Word	Rate
1	00	2 bits	8	11000	5 bits
2	010	3 bits	9	11001	5 bits
3	011	3 bits	10	11010	5 bits
4	1000	4 bits	11	11011	5 bits
5	1001	4 bits	12	11100	5 bits
6	1010	4 bits	13	11101	5 bits
7	1011	4 bits	14	11110	5 bits
			Linear	11111	5 bits

To evaluate the coding performance of the proposed codec, we compare the proposed method with JPEG. Fig. 4.15(b) shows the decoded image using the B-spline curves in Fig. 4.15(a) with PSNR of 28.83dB and compression rate of 30.86, where the Q_k, Q_d and Q_i are set as 2,1,and 4, respectively. As a comparison, the image compressed by JPEG is shown in Fig. 4.15(c) with PSNR of 30.29dB and compression rate of 30.79. Although there is a 1.5dB difference in PSNR, the visual quality of Fig. 4.15(b) is comparable to that of Fig. 4.15(c). Another example is shown in Fig. 4.16.

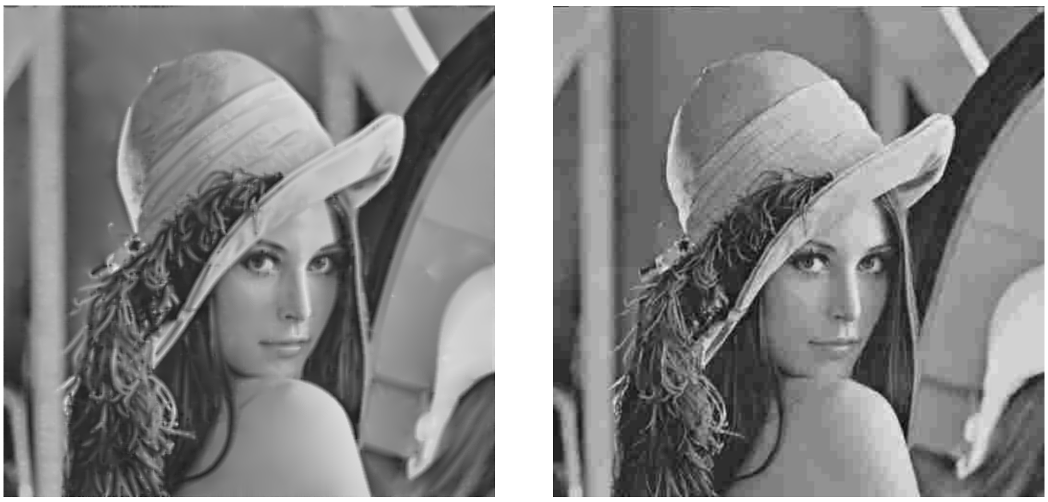



(a) (b)



(c)

Fig. 4.15. (a) Approximated B-spline curves, where curves with positive and negative curvature values are indicated in red and green respectively. (b) Decoded Image using the proposed method. (c) Decoded Image using JPEG.



(a) (b)

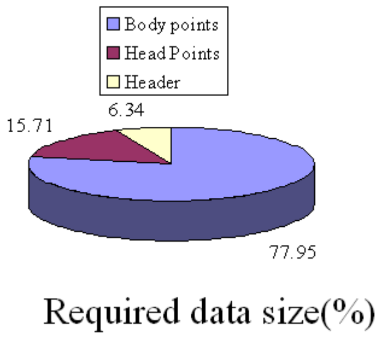
Fig. 4.16. Illustrations of two different compression algorithms. (a) Reconstructed image using the proposed method; compression rate=31.12, PSNR=28.67. (b) Reconstructed image using JPEG compression, compression rate=31.08, PSNR=30.65.

Some comparison results between control point codec and JPEG are listed in Table. 4.4. Although the PSNR gaps between the proposed codec and JPEG vary from 0.4dB to 2dB, the subjective image quality is comparable. We can also see in Table 4.4 that most resources are spent in the encoding of body points. In addition, the required execution time for encoding a bitstream and the coded size are listed in Table. 4.5, where SI and JP indicate the size and the moving step for the sliding window in the allocation of B-spline control points. With a larger JP, less execution time is needed, while more data are required to encode control points. A smaller JP produces more compact data but with longer execution time. In the method "NR", we search the best path in the initial sliding window, and search legitimate control points iteratively in rest sliding windows based on previously obtained results. For image decoding, the required execution time is 3.2 seconds for "Lena", 3.1 seconds for "Fruit", and 3.1 seconds for "Peppers".

Table 4.4.

	Proposed codec		JPEG	
	CR	PSNR	CR	PSNR
Lena	31.12	28.67	31.08	30.65
Fruit	30.57	28.28	30.45	30.31
Peppers	30.86	28.83	30.79	30.29
House	58.47	33.86	58.39	34.25

(a)



(b)

(a) Experimental results. (b) Resource consumption.

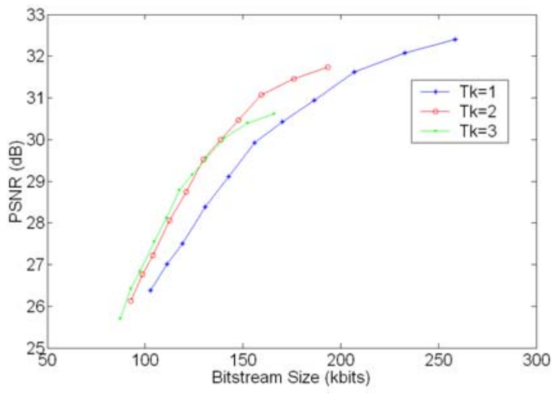
Table 4.5. Average required execution time and encoded bitstream size.

Test images: Lena, Fruit, Peppers.

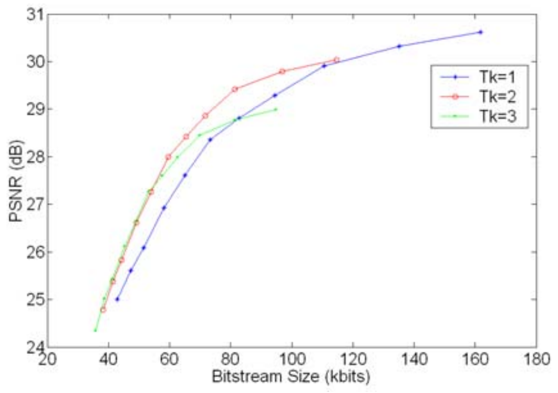
Encoding methods	Lena		Fruit		Peppers	
	Execution time	Bitstream size	Execution time	Bitstream size	Execution time	Bitstream size
NR	27s	91.6k bits	29s	93.9k bits	29s	92.9k bits
SI=16, JP=12	113s	79.5k bits	120s	81.8k bits	118s	81.1k bits
SI=16, JP=8	266s	68.9k bits	289s	69.9k bits	277s	68.9k bits
SI=16, JP=4	841s	67.8k bits	889s	69.2k bits	857s	68.5k bits

We also compare the coding performance of verge point representation with the coding performance of B-spline representation. In Fig. 4.17(a), we show the relation between the PSNR of the reconstructed image with respect to the required bitstream size at different selections of curvature quantizer $Q_{k,}$ where the transmission priority of linked curves are arranged using curve length in the descending order. In Fig. 4.17(b), we show the relation between PSNR and the required bitstream size for the case of

B-spline curve representation. We can see from the result that the B-spline representation offers higher compactness with a sacrifice of slightly lower objective visual quality.



(a)



(b)

Fig. 4.17. (a) The relation between storage requirement and PSNR for different Tk's, based on the verge curve representation. (Test image: 512x512 Lena image.) (b) The relation between storage requirement and PSNR for different Tk's, based on the B-spline curve representation. (Test image: 512x512 Lena image.)

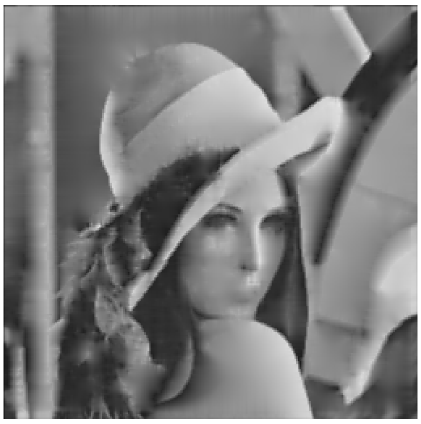
4.4. Scalable Image Transmission

Scalable coding is useful in transmitting images through network with fluctuated bandwidth. In this section, we present the scalability for the proposed image codec based on verge point representation. By rearranging the B-spline control points, the proposed method could offer diversified scalabilities, including SNR scalability, spatial scalability, and shape scalability.

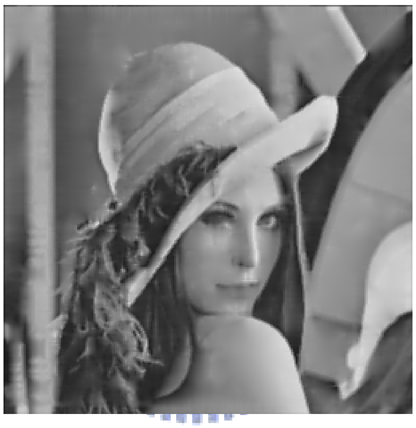
4.4.1 SNR scalability

In the proposed method, SNR scalability can be achieved in three ways, bitplane domain, scale domain, and hybrid way. In the first scenario, we divide the intensity value of the control points into bitplanes. MSB bitplanes are placed at the top of the encoded bitstream, while LSB bitplanes are placed at the bottom. For bitplane domain SNR scalability, transmitted images evolve from high contrast images to image with rich colors. The visual effects of transmitting different bitplanes have been shown in Fig. 3.11. In the second method, SNR scalability can be performed in a coarse-scale to fine-scale manner. In the proposed approach, the original image is decomposed into a 3-layer Gaussian pyramid, with N/4*N/4 pixels in the top layer, N/2*N/2 pixels in the middle layer, and N*N pixels in the bottom layer. The feature extraction and linking processes are first performed over all three layers separately. Then, the inter-layer mappings between every pair of adjacent layers are examined in a top-down order, based on a curve-based strategy. Take the inter-layer mapping between the top layer and the middle layer as an example. Given a verge curve C_k on the top layer, all its 8-connection neighboring pixels are labeled with the curvature sign of that verge curve. These labeled pixels are upsampled by a factor of two to form corresponding areas $R(C_k)$ in the middle layer. Then, in the middle layer, all these verge points locating within R(C_k) are examined to see whether they possess the same curvature sign as C_k. Those verge points with the same sign are considered to be mapped from C_k. If half verge points of a curve in the middle layer are mapped from C_k, that curve is considered to be mapped from C_k and is tagged as a “top-layer” curve. On the other hand, if no verge curve is identified within R(C_k), the feature detection and linking procedures are applied over R(C_k) again with a larger mask sigma (e.g. $2\sigma_m$). Once new verge curves are extracted within R(C_k), the procedure mentioned above is repeated again to tag these newly generated curves. After having identified all “top-layer” curves in the middle layer, the remaining curves are tagged as “middle-layer” curves. That is, all the curves in the middle layer are classified into “top-layer” curves and “middle-layer” curves.

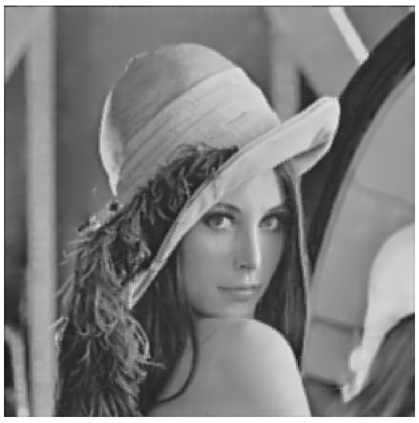
Similarly, the mapping process is applied between the middle layer and the bottom layer to classify verge curves in the bottom layer into “top-layer” curves, “middle-layer” curves, and “bottom-layer” curves. After having classified all verge curves into these three different classes, different levels of image quality can be achieved by arranging the order of transmission, with “top-layer” curves first while “bottom-layer” curves last. With this arrangement, the capability of progressive reconstruction can be achieved as shown in Fig. 4.18. In Fig. 4.18(a), only “top-layer” verge curves are used to reconstruct the Lena image. This reconstructed image catches a gross outline of Lena, but lacks plenty of details. As the verge curves tagged to “middle-layer” curves and “bottom-layer” curves are transmitted, more and more details are revealed, as shown in Fig. 4.18(b) and (c), respectively. Furthermore, for verge curves of the same class, curve contrast may also be used as an indicator to determine the order of transmission, with large-contrast curves first while small-contrast curves last. With curve tag and curve contrast as the indicators, different levels of image quality can be achieved at different bitstream sizes.



(a)



(b)



(c)

Fig. 4.18. Progressive image reconstruction. (a) Using “top-layer” verge curves only; totally 8219 verge points. (b) Using “top-layer” and “middle-layer” verge curves; totally 11523 verge points. (c) Using all verge curves; totally 15926 verge points.

Furthermore, the SNR scalability can be achieved by combining the bitplane scheme and scale scheme. The data structure for the proposed SNR scalability is shown in Fig. 4.19. For B-spline curves in the coarse scale CS, the shape component and the first b_1 bitplanes of the intensity component are transmitted first. After that, the bitplanes from b_1+1 to b_2 for B-spline curves in the coarse scale CS are followed. Then, for B-spline curve in the medium scale MS, shape component and the first b_2 bitplanes of the intensity component are transmitted. Similar rules apply in the fine scale FS. Fig. 4.20(a)-(c) show the examples of SNR scalable transmission, where b_1, b_2, and b_3 are set as 1, 3, and 4 respectively.

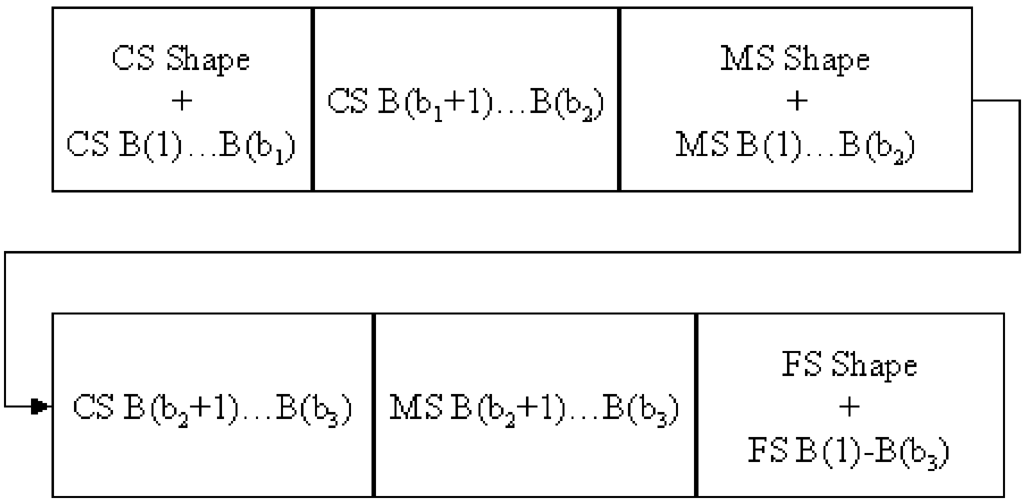
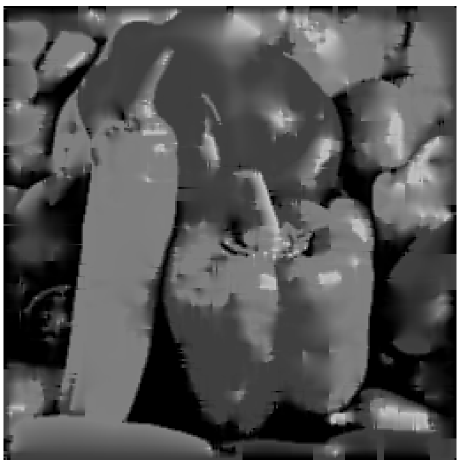
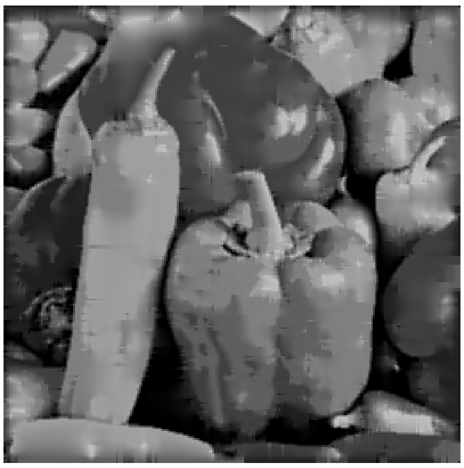


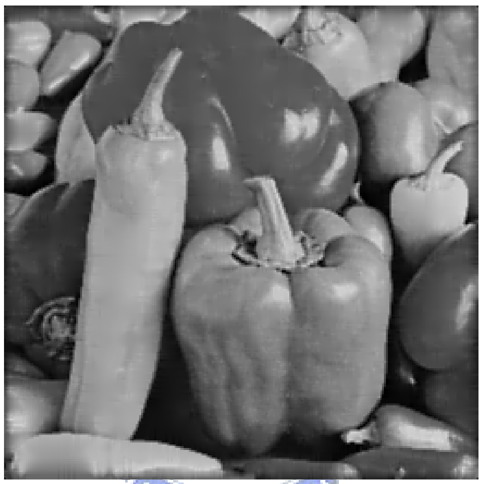
Fig. 4.19. Data arrangement for SNR scalable transmission.



(a)



(b)



(c)

Fig. 4.20. SNR scalable transmission of the image “peppers”. (a) (CS,b_1). (b) (CS+MS,b_2). (c) (CS+MS+FS,b_3).

4.4.2 Spatial Scalability

We use the affine invariant property of B-spline curves to perform spatial scalability. Given two resolutions, R1 (lower resolution) and R2 (higher resolution), we extract verge curves in R1 and R2 separately, and then approximate these curves using B-spline control points. The control points obtained in R1 are stored in layer R1. We scale up the shape component of the control points in R1 with a factor R2/R1, reconstruct these scaled curves, and match the corresponding curves in R2. The ratio

R2/R1 could be a rational factor. After curve mapping, for the unmapped curves in R2, we store the control points in layer R2. Fig. 4.21 shows the procedures to obtain highest resolution R3 from the three-layer image hierarchy at the decoder side. The experimental results using the proposed spatial scalable method are shown in Fig. 4.22(a)-(c). The image resolutions of Fig. 4.22(a)-(c) are 256x256, 370x370, and 512x512 respectively.

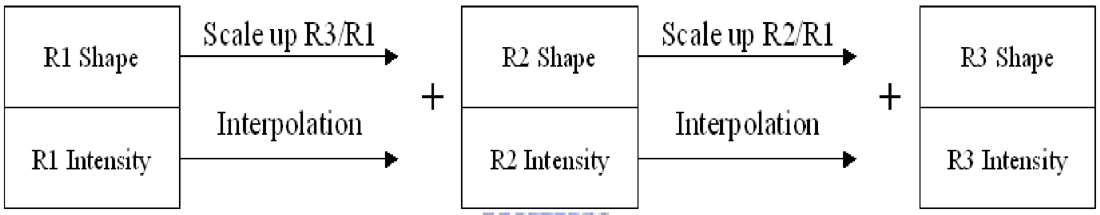
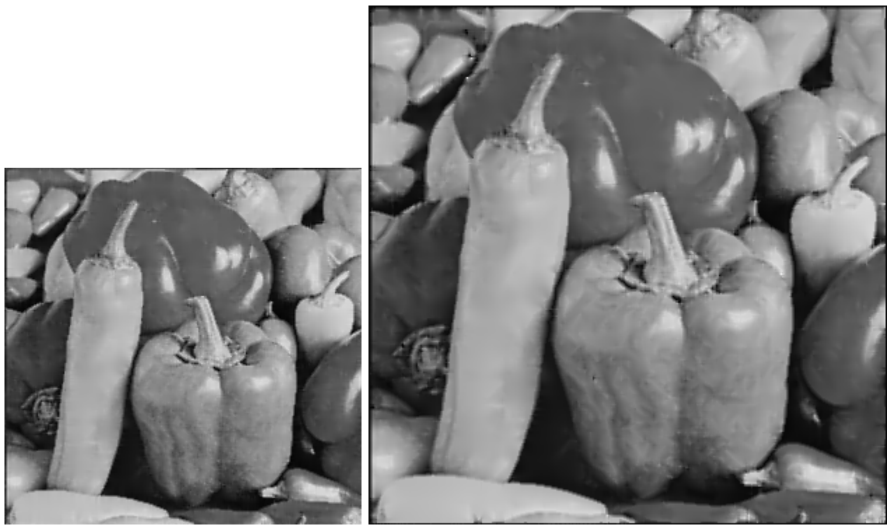
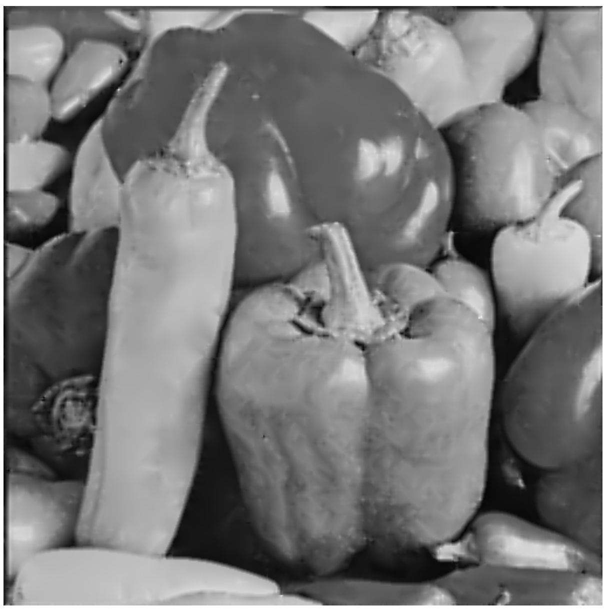


Fig. 4.21. Decoding process for the proposed spatial scalability scheme.



(a) (b)



(c)

Fig. 4.22 Spatial scalable transmission for the image “peppers”. (a) 256x256. (b) 370x370. (c) 512x512.

4.4.3 Shape Scalability

Shape scalability is useful for users to retrieve the images they demand efficiently. For example, users may be interested in images which contain a specific number of objects. Given a B-spline curve with MB control points, C_1, C_2,…,C_{MB}, the shape scalability can be achieved by downsampling these control points. Fig. 4.23. shows an example of three-layer shape hierarchy, S1, S2, and S3. The head and tail control points must be recorded in S1. Except head and tail control points, the control point with index j is classified following the rules:

$$\begin{aligned}&\text{if } j \bmod 4=1,\ C_j \in S1\\ &\text{else if } j \bmod 4=3,\ C_j \in S2 \qquad (4.10)\\ &\text{else } C_j \in S3.\end{aligned}$$

The experimental results using the proposed shape scalability scheme are shown in Fig. 4.24(a)-(c).

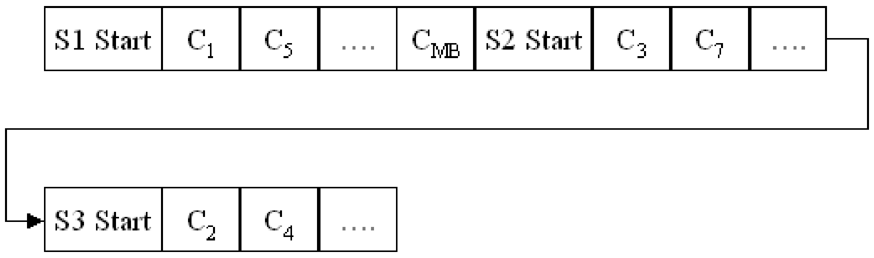
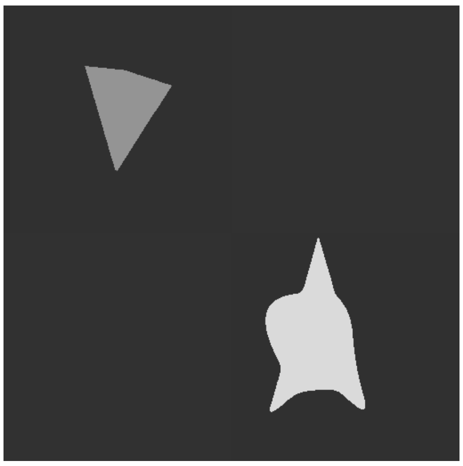
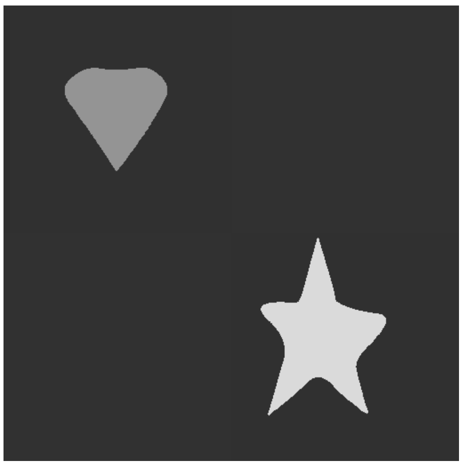
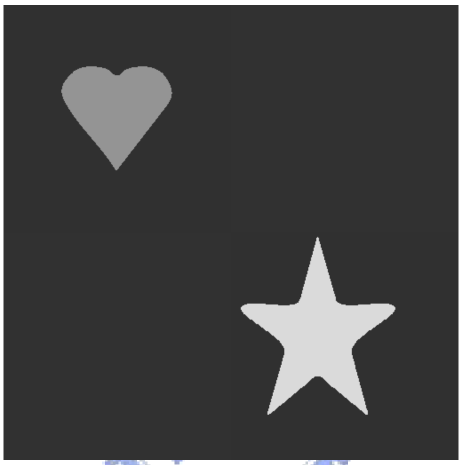


Fig. 4.23. Data arrangement for shape scalability.

(a) (b)



(c)

Fig. 4.24. Examples of shape scalability. (a) S1. (b) S1+S2. (c) S1+S2+S3.

4.5 Interactive Image Transmission

In the proposed method, it is convenient to select and transmit “region of interest” since we code an image based on the features. Instead of selecting specific regions, we select features of interest, which are approximated by B-spline control points, and reconstruct the region information by interpolation. To perform interactive selection and transmission, we divide the B-spline curve space Ω_b into two categories,

$$\Omega_b = \Omega_b^{\text{Normal Mode}} + \Omega_b^{\text{Feature of Interest}}. \tag{4.11}$$

By this decomposition, different effects can be presented by manipulating the curves in $\Omega_b^{\text{Feature of Intereset}}$. For example, to change the priority of the selected features, we can use

$$\Omega_b' = \Omega_b^{\text{Normal Mode}} + \text{BitShift}(\Omega_b^{\text{Feature of Interest}}, \text{BN}) \tag{4.12}$$

The priority of a feature is increased with a positive BN, and decreased with a negative BN. Fig. 4.25(a) shows a decoded image with BN=0. In Fig. 4.25(b), selected features are indicated in magenta. Fig. 4.25(c) shows the decoded image using Fig. 4.25(b), where the parameter BN is set as 6.

In addition, the resolution of the selected features can be changed by

$$\Omega_b' = \Omega_b^{\text{Normal Mode}} + \varphi(\Omega_b^{\text{Feature of Interest}}, \text{SF}), \tag{4.13}$$

where φ(.) represents the shape scaling function similar to the concept of the spatial scalability. The parameter SF indicates the scaling factor. Fig. 4.25(d) shows the scaled B-spline curves (indicated in yellow), and Fig. 4.25(e) shows the decoded image using Fig. 4.25(d). In this example, we set SF as 1.4. This scaling and bitplane shifting procedures can also be applied at the decoder side. This further increases the interactivity of the proposed coding scheme.

In fact, with the aid of B-spline representation, the color and shapes of objects can be conveniently modified by end-users at the decoder sides. The details of objects editing will be presented in Chapter 5.

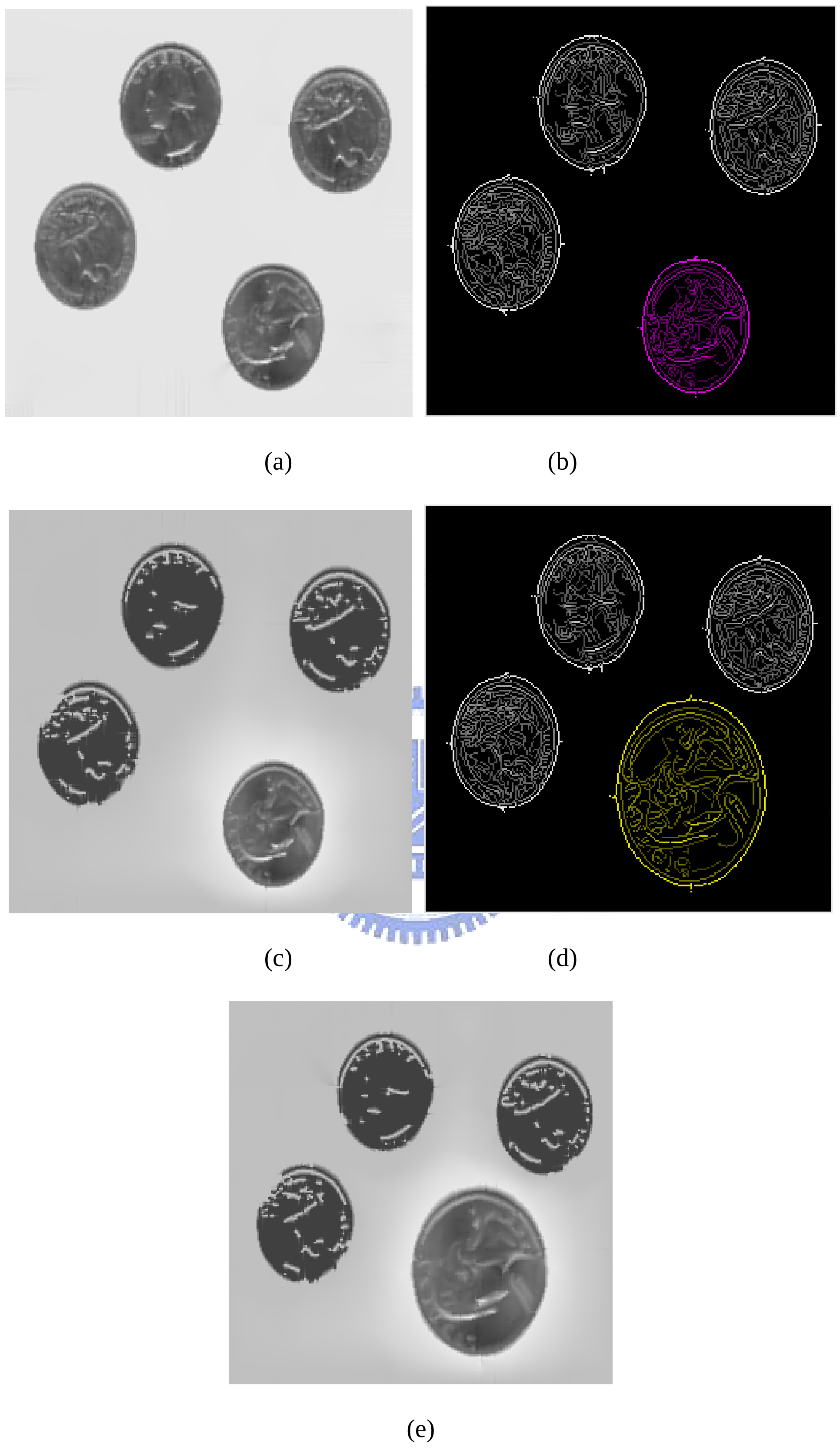


Fig. 4.25. Examples of feature-of-interest selection and transmission. (a) Original decoded image. (b) Features selected for bitplane shifting. (c) Decoded image with bitplane shifting. (d) Features selected for resolution shifting. (e) Decoded image using (d).