Smooth side-match weighted vector quantiser with variable block size for image coding

S.-B. Yang

Abstract: Although side-match vector quantisation (SMVQ) reduces the bit rate, the quality of image coding using SMVQ generally degenerates as the grey level transition across the boundaries of neighbouring blocks increases or decreases. The author proposes a smooth side-match weighted method to yield a state codebook according to the smoothness of the grey levels between neighbouring blocks. When a block is encoded, a corresponding weight is assigned to each neighbouring block to represent its relative importance using the smooth side-match weighted method. This smooth side-match weighted vector quantisation (SSMWVQ) achieves a higher PSNR than SMVQ at the same bit rate. Also, each block can be pre-encoded in an image, allowing each encoded block to use all neighbouring blocks to yield the state codebook in SSMWVQ, rather than using only two neighbouring blocks, as in SMVQ. Moreover, SSMWVQ selects many high-detail blocks as basic blocks to enhance the coding quality, and merges many low-detail blocks into a larger one to reduce further the bit rate. Experimental results reveal that SSMWVQ has a higher PSNR and lower bit rate than other methods.

1 Introduction

Vector quantisation (VQ) [1, 2] is a useful technique for data compression and coding, in particular for image coding and speech coding. In vector quantisation, a codebook, which consists of a set of codewords must be found. Every datum that is in the form of a vector will be represented by a nearest codeword. Compression is thus achieved by representing the input vector using the codeword index. Reconstruction of the image is then by a simple table lookup technique with the necessary transformation.

Several low bit-rate methods for encoding images have been recently proposed, such as SPIHT [3] and HTSMVQ [4]. Finite-state vector quantisers (FSVQ) [5–7] have also been proposed for compressing speech and images at low bit rates. FSVQ provides efficient results because it exploits the correlation between neighbouring vectors to reduce each index size by contracting the codebook to be a smaller one, called a state codebook. FSVQ selects a smaller state codebook from a supercodebook by the current encoder's state. Thus, each input vector is encoded by using its own state codebook. Hence, FSVQ can achieve the better bit rate efficiency.

SMVQ was proposed in [8] and it is a class of FSVQ. Recently, several variants of SMVQ have been proposed in [9–13]. SMVQ tries to make the grey levels of the pixels right across the boundaries of the neighbouring blocks as similar as possible. That is, SMVQ selects from the supercodebook the codewords with small side-match distortion as the state codebook. However, if the grey level transition across the boundaries of the neighbouring blocks is increasing or decreasing, SMVQ may fail to select the codeword that fits the current encoded block best. The reason is that the side-match vector quantiser does not select the codewords to fit the changing style of the grey levels of pixels between the neighbouring blocks. In [12], the smooth side-match vector quantiser (SSMVQ) was proposed. In SSMVQ, the smooth side-match method replaces the conventional side-match method to encode the images, and then the coding quality is enhanced as the same bit rate. Furthermore, in SMVQ and SSMVQ, each encoded block uses two neighbouring blocks to calculate the side-match distortions for the codewords in the supercodebook. However, Fig. 1 shows an example to illustrate that the use of two neighbouring blocks is not sufficient to encode a block well. In Fig. 1, the central block uses the upper and left blocks as the neighbouring blocks; it cannot be encoded well because no edge information is available in the right and lower blocks and the prediction is very bad. Moreover, the SMVQ and SSMVQ are error-propagating vector quantisers. In [4], to make it possible to solve the error-propagating problem, more blocks are selected as the basic blocks while an image is encoded. These basic blocks are first coded by using ordinary VQ. However, the bit rate is increasing when more blocks are used as the basic blocks. In [13], the modified Viterbi algorithm is proposed for solving the error-propagating problem in 1-D SMVQ. In general, the 1-D SMVQ is not sufficient to encode the block well.

SSMWVQ is proposed in the paper and achieves better PSNR than SMVQ because the smooth side-match weighted method selects the state codebook according to the smoothness of the grey levels of pixels between the neighbouring blocks as SSMVQ does. In real images, the grey level transitions across the boundaries of neighbouring blocks are generally flat, increasing or decreasing. Also, in SSMWVQ, when a block is encoded, a corresponding weight is given to each of its neighbouring blocks to represent the relative importance among the neighbouring

IEE Proceedings online no. 20040731

doi: 10.1049/ip-vis:20040731

Paper first received 19th November 2003 and in revised form 7th April 2004

The author is with the Department of Computer Science and Information Engineering, Leader University, No. 188 Sec. 5 An-Chung Rd., Tainan City, Taiwan, Republic of China

E-mail: ysb@mail.leader.edu.tw

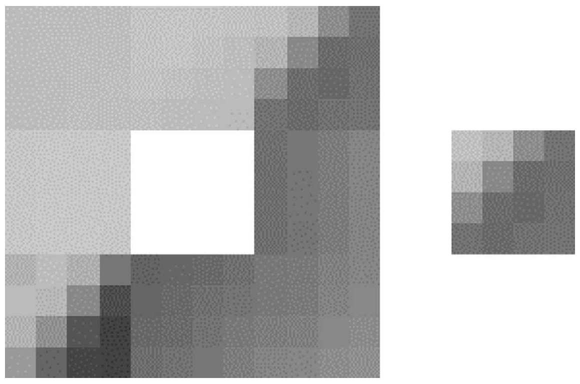


Fig. 1 *A block and its neighbouring blocks*

blocks. Thus, the coding quality by SSMWVQ is better than SSMVQ at the same bit rate. Furthermore, in SSMWVQ, all blocks can be pre-encoded in an image, and then each block can be re-encoded by using more than two neighbouring blocks to calculate the smooth side-match weighted distortions for all the codewords to yield the state codebook. Also, in SSMWVQ, a method for selecting blocks as basic blocks is proposed. Restated, the selection of the basic blocks in SSMWVQ differs according to the image. SSMWVQ selects high-detail blocks as the basic blocks in an image, thus improving the visual perception quality and reducing the error-propagating codes in the image. Furthermore, in SSMWVQ, many low-detail blocks can be merged into a single larger block to further reduce the bit rate.

2 Design of smooth side-match weighted method

The conventional side-match method is given in subsection 2.1, and the smooth side-match weighted method is described in subsection 2.2.

2.1 The conventional side-match method

An FSVQ consists of two finite state machines, an FSVQ encoder and an FSVQ decoder. The FSVQ encoder uses the previously encoded vector to define a state $s \in S = \{s_i : i = 1, \ldots, M\}$, where S denotes the state space. Hence, the FSVQ can be defined as a mapping from $R^k \times S$ to a subset of a supercodebook C. For each state s_i, the state function selects N codewords as the state codebook. For encoding an input vector x_n, the FSVQ encoder finds the current state s_n and then searches the state codebook SC to find its corresponding codeword and then the code u_n is output. To decode a vector, the FSVQ decoder finds the same state s_n and then the state function selects N codewords in the supercodebook C as the state codebook SC. Finally, by using the code u_n the FSVQ decoder searches for the corresponding codeword $\hat{x}_n$ in the state codebook SC.

The SMVQ is a kind of FSVQ. Let x be the $m \times n$ block to be encoded. The state space S is defined as $S = \{u \times \ell : u$ is the codeword for some x's upper block and ℓ is the codeword for some x's left block$\}$. The vertical correlation of state u and the horizontal correlation of state ℓ define the state s of x. The vertical side-match distortion of a codeword y in the supercodebook is defined as:

$$D_u(y) = \sum_{i=1}^{n} (y_{1,i} - u_{m,i})^2 \qquad (1)$$

and the horizontal side-match distortion is defined as:

$$D_\ell(y) = \sum_{j=1}^{m} (y_{j,1} - \ell_{j,n})^2 \qquad (2)$$

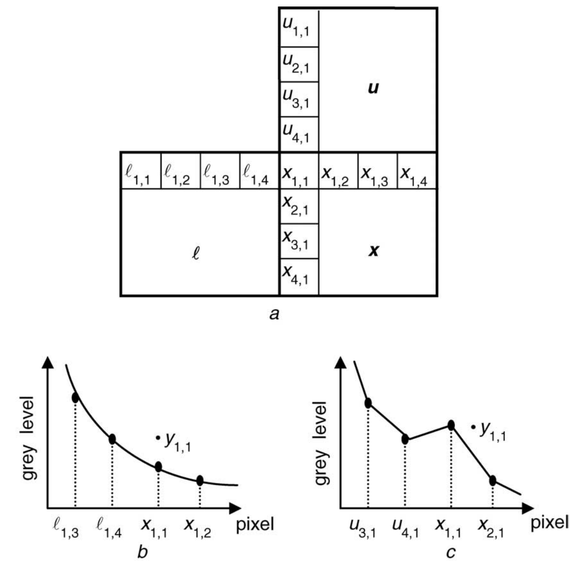
Then, the side-match distortion corresponding to the codeword y is defined as:

$$D(y) = D_u(y) + D_\ell(y) \qquad (3)$$

The selection algorithm selects N_s codewords that have smaller side-match distortions among the codewords in the supercodebook for state s. These N_s codewords constitute x's state codebook. SMVQ sorts the selected codewords in the state codebook according to the side-match distortions of codewords. Then, SMVQ selects the codeword nearest to x from the state codebook to replace x and the index of this codeword is regarded as the code of x.

2.2 The smooth side-match weighted method

SMVQ attempts to bring the grey levels of pixels right across the boundaries of neighbouring blocks as close as possible. In SMVQ, the side-match method selects a subset of the codewords with the smallest side-match distortions in the supercodebook to be the state codebook. However, if the grey levels of the pixels across the boundaries between neighbouring blocks are increasing or decreasing, then the SMVQ tends not to encode a block well. Figure 2*a* shows an example of encoded block x, its left block ℓ and upper block u. Figure 2*b* illustrates the grey levels of the pixels in the encoded block x and its left block ℓ. In choosing codeword y of the state codebook for block x, the side-match method selects the codeword y such that the grey levels of $y_{1,1}$ and $\ell_{1,4}$ are as close as possible. Furthermore, Fig. 2*b* reveals that the grey level of $y_{1,1}$ thus selected may significantly deviate from the grey level of $x_{1,1}$. In real images, changes in grey levels are generally almost smooth among neighbouring pixels, a property which is applied to define the smooth side-match weighted distortion. That is, if changes in grey levels are smooth among neighbouring blocks, the smooth side-match weighted distortion is more appropriate than the conventional side-match distortion. However, in real images, not all changes in grey levels among neighbouring blocks are smooth, especially when the grey levels of the blocks are complex. Figure 2*c* reveals that the changes in



$u_{1,1}$
$u_{2,1}$
$u_{3,1}$
$u_{4,1}$
u
$\ell_{1,1}$ $\ell_{1,2}$ $\ell_{1,3}$ $\ell_{1,4}$ $x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$
$x_{2,1}$
$x_{3,1}$
$x_{4,1}$
ℓ
x
a
grey level
$y_{1,1}$
$\ell_{1,3}$ $\ell_{1,4}$ $x_{1,1}$ $x_{1,2}$
pixel
b
grey level
$y_{1,1}$
$u_{3,1}$ $u_{4,1}$ $x_{1,1}$ $x_{2,1}$
pixel
c

Fig. 2 *The codeword y to encode the block x using its left block ℓ and the upper block u*

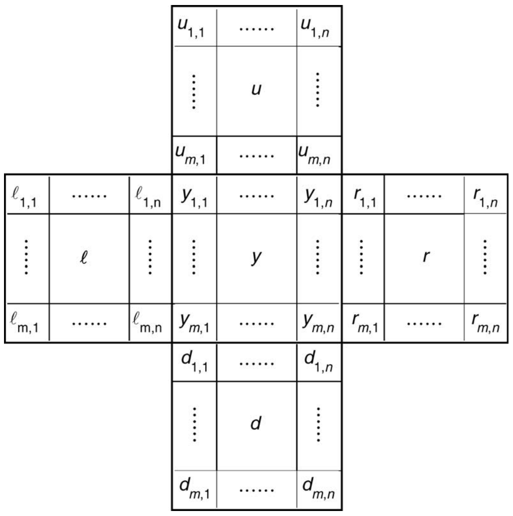


Fig. 3 *The codeword y and its neighbouring blocks*

grey levels among neighbouring pixels are not smoother than shown in Fig. 2*b*. That is, the neighbouring block ℓ is more appropriate than the neighbouring block u for use in the smooth side-match weighted method to predict the state codebook. A corresponding weight is thus assigned to each neighbouring block to present the importance of each neighbouring block in the smooth side-match weighted method. If the grey levels of the block are smooth, then the grey levels of its neighbouring blocks are usually also smooth. Restated, if a block is complex, then its neighbouring blocks will be even more complex. Therefore, in SSMWVQ, the corresponding weight for each neighbouring block depends on the smoothness of its grey levels. Figure 3 displays that the situation when four neighbouring blocks are used to calculate the smooth side-match weighted distortion of codeword y to encode block x. In Fig. 3, the size of all the blocks is $m \times n$. Meanwhile, the difference, $\mathrm{dif}(e,f)$, between the grey levels of pixels e and f is defined as follows:

$$\mathrm{dif}(e,f) = (\text{grey level of } e) - (\text{grey level of } f).$$

First, the smoothness for four neighbouring blocks is defined as:

$$W'_u = \sum_{j=1}^{n} \left| \frac{(\mathrm{dif}(u_{m-3,j}, u_{m-2,j}) + \mathrm{dif}(u_{m-1,j}, u_{m,j}))}{2} - \mathrm{dif}(u_{m-2,j}, u_{m-1,j}) \right| \quad (4)$$

$$W'_\ell = \sum_{i=1}^{m} \left| \frac{(\mathrm{dif}(\ell_{i,n-3}, \ell_{i,n-2}) + \mathrm{dif}(\ell_{i,n-1}, \ell_{i,n}))}{2} - \mathrm{dif}(\ell_{i,n-2}, \ell_{i,n-1}) \right| \quad (5)$$

$$W'_r = \sum_{i=1}^{m} \left| \frac{(\mathrm{dif}(r_{i,4}, r_{i,3}) + \mathrm{dif}(r_{i,2}, r_{i,1}))}{2} - \mathrm{dif}(r_{i,3}, r_{i,2}) \right| \quad (6)$$

$$W'_d = \sum_{j=1}^{n} \left| \frac{(\mathrm{dif}(d_{4,j}, d_{3,j}) + \mathrm{dif}(d_{2,j}, d_{1,j}))}{2} - \mathrm{dif}(d_{3,j}, d_{2,j}) \right| \quad (7)$$

Next, the corresponding weight for each neighbouring block is given as follows:

$$W_u = \frac{\frac{1}{W'_u}}{\frac{1}{W'_u} + \frac{1}{W'_r} + \frac{1}{W'_\ell} + \frac{1}{W'_d}} \quad (8)$$

$$W_r = \frac{\frac{1}{W'_r}}{\frac{1}{W'_u} + \frac{1}{W'_r} + \frac{1}{W'_\ell} + \frac{1}{W'_d}} \quad (9)$$

$$W_\ell = \frac{\frac{1}{W'_\ell}}{\frac{1}{W'_u} + \frac{1}{W'_r} + \frac{1}{W'_\ell} + \frac{1}{W'_d}} \quad (10)$$

$$W_d = \frac{\frac{1}{W'_d}}{\frac{1}{W'_u} + \frac{1}{W'_r} + \frac{1}{W'_\ell} + \frac{1}{W'_d}} \quad (11)$$

Then, the smooth side-match weighted distortions of four neighbouring blocks for codeword y are defined as:

$$D_u(y) = \sum_{j=1}^{n} \left| \frac{(\mathrm{dif}(u_{m-1,j}, u_{m,j}) + \mathrm{dif}(y_{1,j}, y_{2,j}))}{2} - \mathrm{dif}(u_{m,j}, y_{1,j}) \right| \quad (12)$$

$$D_\ell(y) = \sum_{i=1}^{m} \left| \frac{(\mathrm{dif}(\ell_{i,n-1}, \ell_{i,n}) + \mathrm{dif}(y_{i,1}, y_{i,2}))}{2} - \mathrm{dif}(\ell_{i,n}, y_{i,1}) \right| \quad (13)$$

$$D_r(y) = \sum_{j=1}^{m} \left| \frac{(\mathrm{dif}(r_{i,2}, r_{i,1}) + \mathrm{dif}(y_{i,n}, y_{i,n-1}))}{2} - \mathrm{dif}(r_{i,1}, y_{i,n}) \right| \quad (14)$$

$$D_d(y) = \sum_{j=1}^{n} \left| \frac{(\mathrm{dif}(d_{2,j}, d_{1,j}) + \mathrm{dif}(y_{m,j}, y_{m-1,j}))}{2} - \mathrm{dif}(d_{1,j}, y_{m,j}) \right| \quad (15)$$

Therefore, when the block x is encoded, the smooth side-match weighted distortion of the codeword y is defined as

$$D(y) = D_u(y)W_u + D_\ell(y)W_\ell + D_r(y)W_r + D_d(y)W_d \quad (16)$$

The selection algorithm selects N_s codewords from the supercodebook as x's state codebook. These codewords y_i have the smallest smooth side-match weighted distortions $D(y_i)$ for $i = 1, \ldots, N_s$. Also, SSMWVQ sorts the selected codewords in the state codebook according to the smooth side-match weighted distortions of codewords. Thus, SSMWVQ selects the codeword nearest to x from the state codebook to replace x and the index of this codeword is regarded as the code of x.

3 Design of SSMWVQ

To make it possible to test the performance of SSMWVQ, three types of SSMWVQ, namely SSMWVQ(1), SSMWVQ(2) and SSMWVQ(3), are proposed in this paper. SSMWVQ(1) is similar to the conventional SMVQ. In SSMWVQ(1), the smooth side-match weighted method is used to yield the state codebook rather than using the side-match method in SMVQ. In SSMWVQ(2), all blocks can be pre-encoded in an image, and then each block can use whole neighbouring blocks to yield the state codebook by using the smooth side-match weighted method. Both methods, SSMWVQ(1) and SSMWVQ(2), are used to encode blocks of a fixed size. In order to further reduce the bit rate and enhance the coding quality, SSMWVQ(3) is proposed to encode the blocks with variable sizes. We describe the designs of SSMWVQ(1), SSMWVQ(2) and SSMWVQ(3) in Sections 3.1, 3.2 and 3.3, respectively.

3.1 Design of SSMWVQ(1)

The difference between SSMWVQ(1) and conventional SMVQ is the method used to select the codewords from the supercodebook as the state codebook. When an image is encoded in SSMWVQ(1), it is divided into 4×4 blocks. The diagonal blocks, those from the left-upper block to the right-lower block in an image, are used as the basic blocks and they are first encoded using the ordinary codebook. Since the diagonal blocks are used as the basic blocks, the image is divided into two regions, the lower triangular region and the upper triangular region. When a non-basic block $B_{i,j}$ is encoded in SSMWVQ(1), block $B_{i,j}$ uses two neighbouring blocks to calculate the smooth side-match weighted distortions, as in (16), for all codewords in the supercodebook. Meanwhile, the smooth side-match weighted method selects the codewords with the smaller smooth side-match weighted distortions from the supercodebook as the $B_{i,j}$'s state codebook. SSMWVQ(1) then sorts the selected codewords in the state codebook according to the smooth side-match weighted distortions of the codewords. Thus, SSMWVQ(1) selects the codeword nearest to $B_{i,j}$ from the state codebook to replace $B_{i,j}$, and the index of this codeword is then regarded as the code of $B_{i,j}$. However, in SSMWVQ(1), each encoded block has just two neighbouring blocks that need to be used in the smooth side-match weighted method, like the SMVQ and SSMVQ. In SSMWVQ(1), W_ℓ and W_d are both set to zero in (16) for the blocks in the lower triangular region, while for the blocks in the upper triangular region, W_u and W_r are set to zero.

3.2 Design of SSMWVQ(2)

In SSMWVQ(2), there are two stages to encode an image. In the first stage of SSMWVQ(2), the diagonal blocks are selected as the basic blocks and they are encoded by the ordinary codebook as SSMWVQ(1). Then, each non-basic block can use two neighbouring blocks to calculate the smooth side-match weighted distortions for all the codewords, and then the codeword with the smallest smooth side-match weighted distortion in the state codebook is regarded the encoding result in the first stage. In general, the codeword with the smallest smooth side-match weighted distortion in the state codebook is called the best codeword. Thus, after the first stage of SSMWVQ(2), each non-basic block is encoded by its best codeword.

Let the codeword $C_{i,j}$ indicate the encoding result of block $B_{i,j}$ in the first stage of SSMWVQ(2). In the second stage of SSMWVQ(2), each non-basic block will be encoded once again by using the smooth side-match weighted method. When the non-basic block $B_{i,j}$ is encoded again in the second stage, $B_{i,j}$ uses the pre-encoded codewords, $C_{i,j+1}$, $C_{i-1,j}$, $C_{i,j-1}$ and $C_{i+1,j}$, as the neighbouring blocks to calculate the smooth side-match weighted distortions in (16) for all the codewords in the supercodebook. Since all the neighbouring blocks of the block $B_{i,j}$ are pre-encoded in the first stage, the block $B_{i,j}$ can use all the neighbouring blocks in the smooth side-match weighted method rather than two neighbouring blocks in SSMWVQ(1). Therefore, SSMWVQ(2) can select better codewords as the state codebook for the block $B_{i,j}$ than SSMWVQ(1) selected. Similarly, SSMWVQ(2) sorts the selected codewords in the state codebook according to the smooth side-match weighted distortions of codewords. Thus, the codeword nearest to $B_{i,j}$ is selected from the state codebook to replace $B_{i,j}$ and the index of this codeword is regarded as the code of $B_{i,j}$. The coding quality of SSMWVQ(2) is better than that of SSMWVQ(1) as the same bit rate.

3.3 Design of SSMWVQ(3)

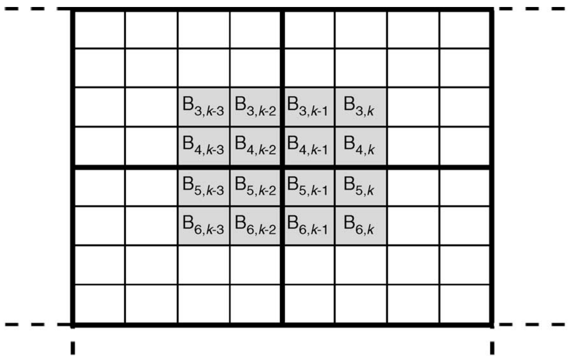
To further enhance coding quality, SSMWVQ(3) selects many high-detail blocks as the basic blocks to encode the image, thus improving coding quality and reducing the error-propagating codes in an image. Furthermore, SSMWVQ(3) can merge many smaller low-detail blocks into a single larger one to further reduce the bit rate. SSMWVQ(3) needs three codebooks to encode the image. The first codebook is the same as that used in SSMWVQ(1) and SSMWVQ(2). This codebook is used to encode blocks sized 4×4 and the other two codebooks encode low-detail 8×8 and 16×16 blocks, respectively. Meanwhile, in SSMWVQ(3), the first codebook is used to encode high-detail and low-detail 4×4 blocks, and this codebook is larger than the other two 8×8 and 16×16 codebooks. Also, SSMWVQ(3) encoded the image in two stages and they are described as follows.

In the first stage of SSMWVQ(3), when an image is encoded in SSMWVQ(3), the image is divided into 4×4 blocks. The diagonal blocks, those from the left-upper block to the right-lower block in an image, are encoded by the ordinary codebook as with SSMWVQ(2). Then, each non-basic block can be encoded by its best codeword. Furthermore, to enhance the coding quality and the visual perception quality, many high-detail blocks are selected as the basic blocks in SSMWVQ(3). The diagonal blocks can be positioned before encoding the images. However, the high-detail blocks are positioned differently according to the image. This work describes below how high-detail blocks of the encoded image are selected as basic blocks. First, all the codewords are sorted in the codebook according to the variance of each codeword. Let the codebook contain N codewords, $y_1, y_2, \ldots, y_N$. The variance $\mathrm{var}(y_i)$ then indicates the variance of codeword y_i, and the codewords can be sorted according to their individual variance, as follows:

$$\mathrm{var}(y_1) \geqq \mathrm{var}(y_2) \geqq \cdots\cdots \geqq \mathrm{var}(y_k) \geqq \cdots\cdots \geqq \mathrm{var}(y_N) \tag{17}$$

After each block is encoded by its best codeword in the first stage of SSMWVQ(3), the high-detail blocks can be decided according to the encoded code of each block. For example, if a block is encoded by the best codeword y_1, then this block indicates it has high variance as the codeword y_1 and thus this block is regarded as a high-detail block. Moreover, if a block is encoded by the best codeword y_N, then it indicates this block has low variance as the codeword y_N and thus this block is regarded as a low-detail block. SSMWVQ(3) selects many high-detail blocks as the basic blocks. Since the variances of the codewords, $\mathrm{var}(y_i)$ for $1 \leqq i \leqq N$, can be pre-calculated and pre-sorted in the codebook, we need not to calculate the variances of all the blocks in the first stage.

In the second stage of SSMWVQ(3), many smaller low-detail 4×4 blocks will be merged to further reduce the bit rate. In SSMWVQ(3), the low-detail blocks can also be identified according to the code encoded for each block in the first stage. That is, if the code of the block exceeds the threshold $m (m < N)$, then the variance of this block is low and it can be considered a low-detail block. Figure 4 illustrates an example. In Fig. 4, the grey blocks are the



$B_{3,k\text{-}3}$ $B_{3,k\text{-}2}$ $B_{3,k\text{-}1}$ $B_{3,k}$
$B_{4,k\text{-}3}$ $B_{4,k\text{-}2}$ $B_{4,k\text{-}1}$ $B_{4,k}$
$B_{5,k\text{-}3}$ $B_{5,k\text{-}2}$ $B_{5,k\text{-}1}$ $B_{5,k}$
$B_{6,k\text{-}3}$ $B_{6,k\text{-}2}$ $B_{6,k\text{-}1}$ $B_{6,k}$

Fig. 4 *Example of the low-detail blocks in an image*

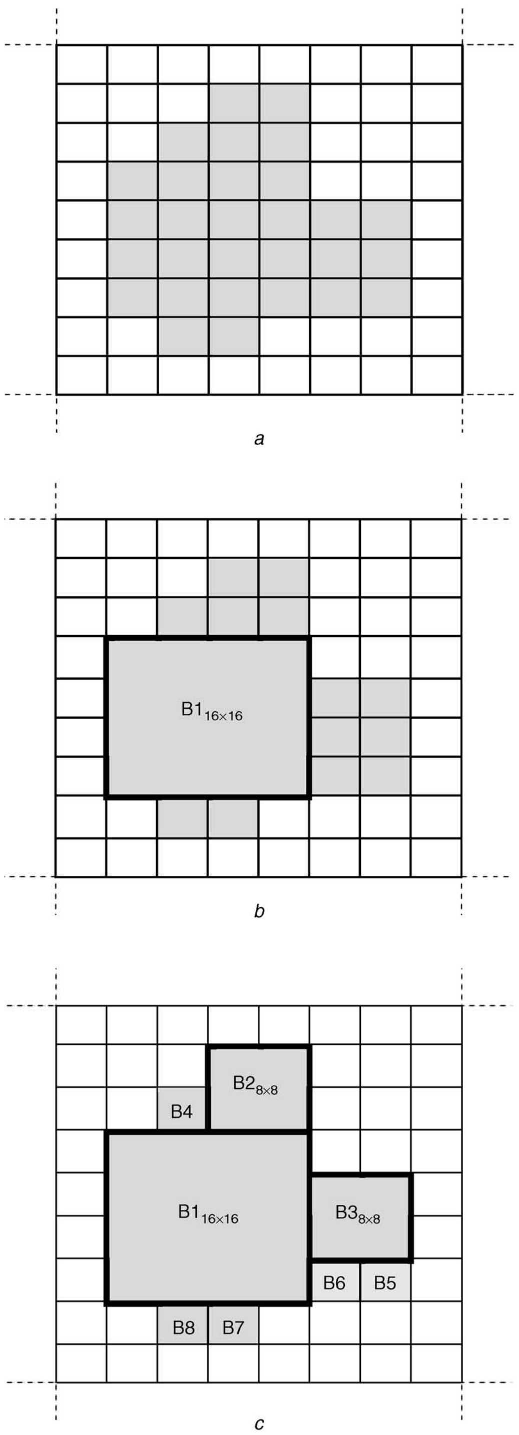


Fig. 5 *Example for merging many low-detail blocks into a larger one in SSMWVQ(3)*

low-detail 4×4 blocks. In [14], the encoded image is first divided into blocks of size 16×16, and then whether each block is low-detail or not is determined.

Low-detail 16×16 blocks are directly encoded by the codebook for the blocks of size 16×16, while other high-detail 16×16 blocks are further divided into four 8×8 blocks. However, although a low-detail 16×16 block exists in Fig. 4, the conventional segmentation method cannot encode the 16×16 block. The conventional segmentation method fails because the low-detail 16×16 block is first divided into four low-detail 8×8 blocks, each of which can then be encoded. Furthermore, after dividing the larger block into smaller ones, the segmentation codes, such as quadtree codes [10, 12, 15], must be sent to the decoder, increasing the bit rate for the encoded image. In the second stage of SSMWVQ(3), SSMWVQ(3) tries to merge many non-basic and low-detail 4×4 blocks into a 16×16 block without increasing the bit rate. First, each low-detail 4×4 block is considered, whether or not its neighbours are low-detail blocks. Figure 4 shows that block $B_{3,k}$ is a low-detail block and so are its neighbours, meaning that these blocks, $B_{i,j}$ for $3 \le i \le 6$ and $k-3 \le j \le k$, can be merged into a single 16×16 block, and then be encoded by the codebook for low-detail 16×16 blocks by using the smooth side-match weighted method. Secondly, after the 16×16 blocks are encoded, each low-detail 4×4 block that has not been encoded seeks neighbouring low-detail blocks to merge into a single 8×8 block. Similarly, each 8×8 block can be encoded by using the codebook for low-detail 8×8 blocks. Finally, the only remaining un-encoded blocks are those with a size of 4×4, and these are encoded by the codebook for 4×4 blocks by using the smooth side-match weighted method, thus obtaining the codes for the remaining blocks. Figure 5*a* shows an example for the low-detail blocks. In Fig. 5*b*, the low-detail 16×16 block is found and then encoded. As Fig. 5*c* illustrates, two low-detail 8×8 blocks then require encoding. Finally, the remaining blocks, with a size of 4×4, are encoded, as shown in Fig. 5*c*.

4 Experimental results

The experiments conducted herein employ five images, each 512×512, as training images. The LBG algorithm [16] is

Table 1: Coding qualities of three methods at bit rate 0.191 bit/pixel

	PSNR (dB)		
Images	SMVQ	SSMVQ	SSMWVQ(1)
Lena	29.51	30.92	31.45
F-16	29.82	31.42	32.18
Boats	29.32	30.62	31.01
Peppers	30.22	31.89	32.34

Table 2: Coding qualities of three methods at bit rate 0.191 bit/pixel by using two stages to encode an image

	PSNR (dB)					
	SMVQ		SSMVQ		SSMWVQ(2)	
Images	First	Second	First	Second	First	Second
Lena	25.31	32.28	25.52	33.53	26.21	34.33
F-16	25.59	32.44	26.12	33.98	27.05	35.01
Boats	25.01	32.12	25.33	33.32	25.89	34.72
Peppers	25.98	32.96	26.42	34.41	27.18	35.05

Table 3: PSNR by SSMWVQ(3)

	PSNR (dB)			
Blocks	Lena	Boats	Peppers	F-16
Low-detail blocks 16 × 16	35.31	35.33	35.41	35.44
Low-detail blocks 8 × 8	34.98	34.89	35.22	35.25
The other blocks 4 × 4	33.53	33.65	34.22	34.25
Overall	34.80	34.84	35.15	35.18

Table 4: Bit rate by SSMWVQ(3)

		Bit rate (bit/pixel)			
Blocks		Lena	Boats	Peppers	F-16
Basic blocks	diagonal	0.00488	0.00488	0.00488	0.00488
	high-detail	0.00488	0.00488	0.00488	0.00488
Non-basic	16 × 16	0.00077	0.00062	0.00116	0.00057
blocks	8 × 8	0.00618	0.00624	0.00772	0.00658
	4 × 4	0.12983	0.13833	0.11129	0.12828
Overall		0.14654	0.15495	0.12993	0.14699

used to design three codebooks, for the 4 × 4 blocks, and the low-detail 8 × 8 and 16 × 16 blocks. The codebooks for 4 × 4, 8 × 8 and 16 × 16 blocks are 1024, 256 and 256, respectively. The testing images not included in the training set are used herein to test coding performance. To verify the efficiency of the smooth side-match weighted method, SSMWVQ(1) is compared to the SMVQ and SSMVQ methods. The SSMVQ method indicates that the smooth side-match method is used to yield the state codebook, while the smooth side-match method is denoted as the smooth side-match weighted method without using the corresponding weights for the neighbouring blocks. That is, all the weights of neighbouring blocks are set to equality. To enable comparison of the performance of these three methods, the same codebook and diagonal blocks are used as the basic blocks for image encoding. Table 1 lists the coding qualities of these three methods when the size of the supercodebook is 1024 and that of the state codebook is 8. In Table 1, the coding quality of SSMWVQ(1) exceeds that of the other two methods given the same bit rate. This phenomenon occurs because the smooth side-match weighted method selects better codewords as the state codebook than the other two methods selected.

The SSMWVQ(2) consists of two stages to encode an image. In the first stage of SSMWVQ(2), each non-basic

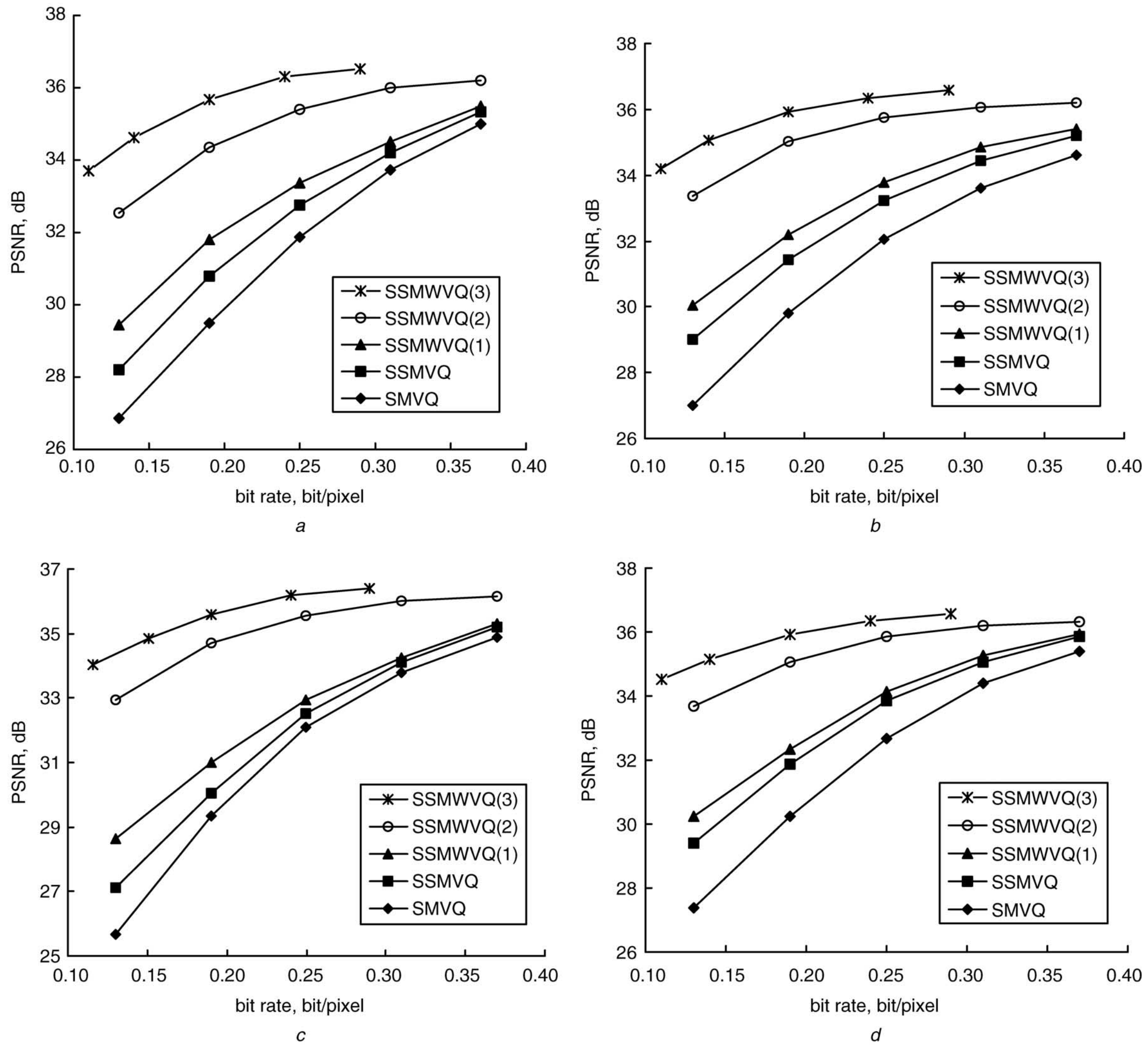


Fig. 6 *Comparisons of the various methods*

a 'Lena' image
b 'F-16' image
c 'Boats' image
d 'Peppers' image

block is encoded by the best codeword. Then, each non-basic block can be encoded one again by using more than two neighbouring blocks to yield the state codebook. However, the first stage of SSMWVQ(2) can also be applied to the conventional SMVQ and its variants. In other words, the SMVQ also consists of two stages as SSMWVQ(2) when encoding an image. Table 2 employs the comparison of coding qualities between the SSMWVQ(2) and the other methods. In Table 2, the supercodebook size is 1024 and the state codebook size is set to 8.

In SSMWVQ(3), besides using the diagonal blocks as the basic blocks, 128 high-detail blocks are selected as the basic blocks after the first stage of SSMWVQ(3). Thus, the total number of basic blocks used to encode the image is 256 $(128\,(\text{diagonal}) + 128\,(\text{high-detail}) = 256)$. In the second stage of SSMWVQ(3), the low-detail blocks can be decided according to the codes of the blocks encoded in the first stage. That is, if the code of block exceeds the threshold $m = 4N/5$, where N denotes as the supercodebook size 1024, then this block is considered a low-detail block. If the threshold m is high, the number of low-detail blocks in the encoded image is decreasing, and otherwise it is increasing. Tables 3 and 4 list the PSNR and bit rate for the testing images encoded by SSMWVQ(3), respectively. In Table 3, the state codebook sizes for the 4×4, 8×8 and 16×16 blocks are set as 8, 4 and 4, respectively. The use of the smaller size of state codebook to encode the low-detail blocks is intended to reduce the bit rate in SSMWVQ(3).

Figure 6 compares the coding qualities of the SSMWVQ and other methods. The state codebook sizes are set as 4, 8, 16, 32, and 64 for the 4×4 blocks, and 2, 4, 8, 16, and 32 for the 8×8 and 16×16 low-detail blocks, respectively. In Fig. 6, the SSMWVQ(3) achieves better coding quality than the other methods given the same bit rate. Meanwhile, in SSMWVQ(3), the low-detail blocks are as large as possible and are encoded by a smaller sized state codebook, further reducing the bit rate for the low-detail blocks. Also, the selection of many high-detail blocks as basic blocks and the use of larger state codebooks further enhances coding quality. Additionally, since many high-detail blocks are selected as the basic blocks in SSMWVQ(3), the image encoded by SSMWVQ(3) is higher quality than with the other methods. Figure 7*a* shows the coding quality of the 'Lena' image produced using only the first stage of SSMWVQ(3). Although the coding quality is not very good, the required bit rate is extremely small. In Fig. 7*a*, the bit rate only consists of the codes of the diagonal blocks and

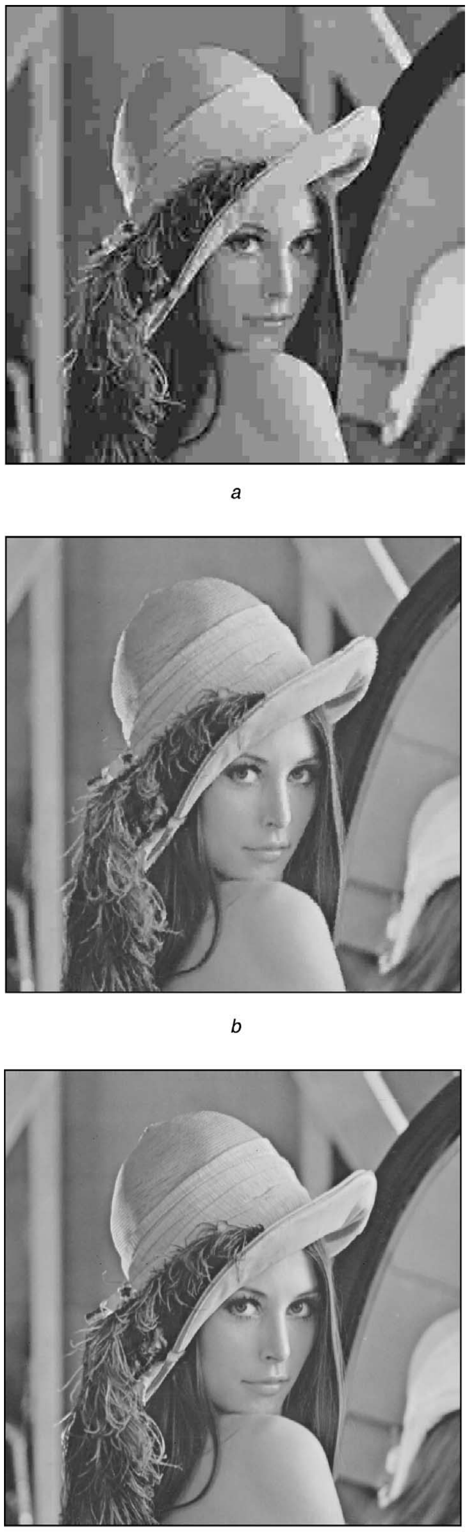


Fig. 7 *'Lena' image encoded by SSMWVQ(3)*
a First stage of SSMWVQ(3) (22.42 dB, 0.00996 bit/pixel)
b Second stage of SSMWVQ(3) (34.80 dB, 0.14242 bit/pixel)
c Original image

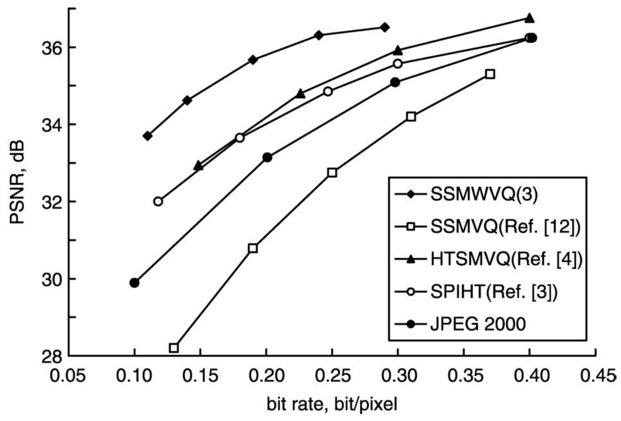


Fig. 8 *Comparisons among PSNR and bit rate of 'Lena' image coded by SSMWVQ(3) and the other methods*

Table 5: Times to code 'Lena' image by SSMWVQ(3) and JPEG 2000

Methods	Encoding time (s)	Decoding time (s)
SSMWVQ(3)	8.188	7.632
JPEG 2000	2.543	2.032

forty high-detail blocks. After the first stage of SSMWVQ(3) is finished, the second stage of SSMWVQ(3) continues to encode the 'Lena' image, as shown in Fig. 7*b*. Furthermore, Fig. 8 shows the quality of the 'Lena' image by the SSMWVQ(3) and other methods. In Fig. 8, SSMWVQ(3) outperforms the other methods in most cases. Table 5 lists the times required to code the 'Lena' image using SSMWVQ(3) and JPEG2000.

A personal computer (P-IV 2 GHz) was used to perform this experiment. Although the coding time required by SSMWVQ(3) exceeds that of JPEG2000, the quality of the 'Lena' image coded by SSMWVQ(3) exceeds that of the image coded by JPEG2000.

5 Conclusions

This paper has demonstrated the feasibility of applying the smooth side-match weighted method to select codewords from the supercodebook as the state codebook to encode the blocks in an image. In the smooth side-match weighted method, the corresponding weight for each neighbouring block depends on the smoothness of the neighbouring block. That is, if the grey levels of the neighbouring block are smoother, then in the smooth side-match weighted method its corresponding weight is enhanced; otherwise the corresponding weight is weakened. Furthermore, three types of SSMWVQs are proposed to encode the images. In SSMWVQ(3), each block is pre-encoded in the first stage, and then the high-detail blocks are selected as the basic blocks based on the encoded code of each block. Thus, the spread of the high-detail blocks differs from the various images. In the second stage of SSMWVQ(3), each block can be encoded by using all of the neighbouring blocks to calculate the smooth side-match weighted distortion for each codeword and thus create the state codebook, which is better than that obtained by the SMVQ and SSMVQ. Moreover, many low-detail blocks can be merged into a single larger one without requiring extra bit rate, thus further reducing the total bit rate in SSMWVQ(3). As indicated by the experimental results, SSMWVQ(3) achieves higher coding quality, a lower bit rate and better visual perception quality than conventional methods.

6 References

1 Gersho, A., and Gray, R.M.: 'Vector quantization and signal compression' (Kluwer, Boston, MA, USA, 1992)

2 Nasrabadi, N.M., and King, R.A.: 'Image coding using vector quantization: A review', *IEEE Trans. Commun.*, 1988, **36**, pp. 957–971

3 Said, A., and Pearlman, W.A.: 'New, fast, and efficient image codec based set partition in hierarchical trees', *IEEE Trans. Circuits Syst. Video Technol.*, 1996, **6**, pp. 243–249

4 Wei, H.C., Tsai, P.C., and Wang, J.S.: 'Three-sided side match finite-state vector quantization', *IEEE Trans. Circuits Syst. Video Technol.*, 2000, **10**, (1), pp. 51–58

5 Dunham, M.O., and Gray, R.M.: 'An algorithm for the design of label-transition finite-state vector quantizers', *IEEE Trans. Commun.*, 1985, **33**, (1), pp. 83–89

6 Foster, J., Gray, R.M., and Dunham, M.O.: 'Finite-state vector quantization for waveform coding', *IEEE Trans. Inf. Theory*, 1985, **31**, (3), pp. 348–359

7 Tsai, J.C., Hsieh, C.H., and Hsu, T.C.: 'A new dynamic finite-state vector quantization algorithm for image compression', *IEEE Trans. Image Process.*, 2000, **9**, (11), pp. 1825–1836

8 Kim, T.: 'Side match and overlap match vector quantizers for images', *IEEE Trans. Image Process.*, 1992, **1**, (2), pp. 170–185

9 Chang, R.F., and Chen, W.T.: 'Image coding using variable-rate side-match finite-state vector quantization', *IEEE Trans. Image Process.*, 1993, **2**, (1), pp. 104–108

10 Chang, R.F., and Chen, W.M.: 'Adaptive edge-based side-match finite-state classified vector quantization with quadtree map', *IEEE Trans. Image Process.*, 1996, **5**, (2), pp. 378–383

11 Chen, T.S., and Chang, C.C.: 'A new image coding algorithm using variable-rate side-match finite-state vector quantization', *IEEE Trans. Image Process.*, 1997, **6**, (8), pp. 1185–1187

12 Yang, S.B., and Tseng, L.Y.: 'Smooth side-match classified vector quantizer with variable block size', *IEEE Trans. Image Process.*, 2001, **10**, (5), pp. 677–685

13 Chung, J.K., and Lin, C.S.: 'Viterbi-based algorithm for side-match vector quantization over noisy channels', *IEEE Trans. Commun.*, 1996, **44**, (11), pp. 1455–1465

14 Vaisey, J., and Gersho, A.: 'Image compression with variable block size segmentation', *IEEE Trans. Signal Process.*, 1992, **40**, (8), pp. 2040–2060

15 Samet, H.: 'The quadtree and related hierarchical data structures', *ACM Comput. Surv.*, 1984, **16**, pp. 188–216

16 Linde, Y., Buzo, A., and Gray, R.M.: 'An algorithm for vector quantizer design', *IEEE Trans. Commun.*, 1980, **28**, (1), pp. 84–95