Limited Color Display for Compressed Image and Video

Soo-Chang Pei, *Fellow, IEEE*, Ching-Min Cheng, and Lung-Feng Ho

***Abstract*—Many display devices nowadays still allow a limited number of colors, called color palette, to be displayed simultaneously. Besides, images and videos in most World Wide Web databases are in compressed formats. Therefore, it becomes an important issue to retrieve a suitable color palette from compressed domain in order to have fast and faithful color reproduction for these devices. In this paper, the color-palette design methods for compressed images and videos are presented. The proposed approaches use the reduced, rather than the whole, image for the color palette design to avoid the heavy computation in image or video decompression. Also, for compressed videos, a shifting-window scheme is proposed to smooth out color variations in the change of color palette. In these methods, we extend the dependent scalar quantization algorithm of a single image to accomplish the color palette design. Experimental results show that output image quality of proposed methods is acceptable to human eyes. In addition, empirical results show that the proposed shifting-window scheme can reduce the main problem of displaying quantized image sequences, screen flicker.**

***Index Terms*—Color display, color palette, color quantization, compressed image, compressed video.**

I. Introduction

With THE prevalence of multimedia and the Internet, more and more digital images and videos are available for people to access. The digital image or video format is usually quantized with integer from 0 to 255 for each of three color components (e.g., red, green, blue). All possible combinations of three of these values gives 256^3 (or 16 million) distinct colors for full-color digital display. However, due to the costs of high speed video RAM, many current PCs and workstations generally have a single 8-bit frame buffer to allow only a limited number of colors, called a color palette, to be simultaneously displayed. If an acceptable output image quality is desired, it is necessary to develop a useful procedure, called color quantization, for designing the color palette.

Let the set of all points of input color image be I. The color value of I is represented as $\boldsymbol{I} = (I_1, I_2, I_3)$, where I_i is the associated color component value. Color-palette design, or the color quantization algorithm, is to partition the input colors into M disjoint sets, C_s, $1 \leq s \leq M$. M is the predetermined size of the color palette which is usually limited by the hardware constraints. In each color set C_s, there is a representative color $\boldsymbol{q}_s$. These representative colors constitute the color palette. This process of color quantization can be represented by

$$Q(\boldsymbol{I}) = \boldsymbol{q}_s, \qquad s \in [1, M] \tag{1}$$

where $Q(\boldsymbol{I})$ indicates the output of the color-quantization algorithm and $\{\boldsymbol{q}_s : s \in [1, M]\}$ is the set of available output colors in the color palette. Thus, the main issues in the color-palette design are how to find the representative color $\boldsymbol{q}_s$ and partition the colors into M color sets, C_s.

In the past, the color-palette design focused on uncompressed data. For one single image, several color-quantization algorithms have been proposed. Heckbert has proposed a median cut (MC) algorithm [1], where the color space is recursively divided into M rectangular regions with equal color occurrence and their centroids being representation colors. Braudaway [2] has presented an improved Popularity algorithm which reduces the frequency of neighbors of a selected color to prevent the color concentration in one neighborhood. Recently, the popular vector quantization (VQ) technique [8] is applied to the color-palette design, too. The colors in input image are used as training vectors in order to refine the initial rough palette. But VQ-type algorithms are usually computational intensive such that they are not suitable for real-time processing. Pei and Cheng have suggested a dependent scalar quantization (DSQ) algorithm [3], which exploits dependency of input colors and sequentially partitions the color space. The experimental results show that the DSQ can reduce the computation complexity and its output image quality is acceptable to human eyes. Also, some color-quantization researchers have focused on the processing of image sequences. Roytman and Gotsman [12] have proposed an algorithm for dynamic color quantization of image sequences, which quantizes each image independently to produce a different color palette for each image. Besides, they suggest the approach of color palette filling to prevent frequent switching of color palettes, which leads to the problem of screen flicker.

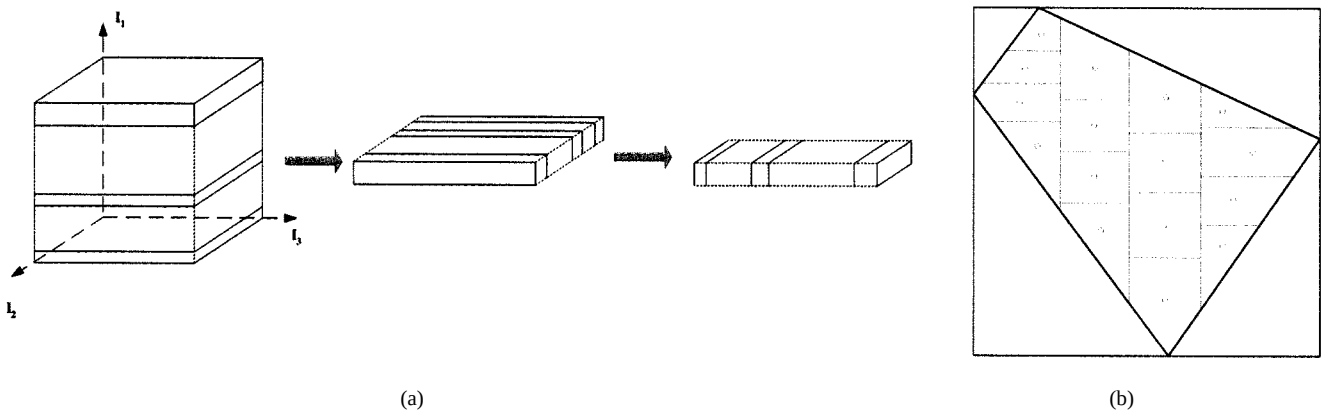
However, with the consideration of communication bandwidth and storage space, most image or video data nowadays comes in compressed format. Extra heavy computation of image or video decompression is required before any above-mentioned color quantization is employed. How to design the color palette directly from compressed data has thus become a significant issue. In this paper, we extend the DSQ to present some novel color-palette design methods for compressed images and videos. Concerning compressed images, the proposed method use the reduced image in compressed domain to design color palette. For compressed video, the

Manuscript received December 15, 1998; revised December 28, 1999. This work was supported by National Science Council, R.O.C., under Contract NSC 88-2213-E-002-047. This paper was recommended by Associate Editor R. Lancini.

S. C. Pei and L. F. Ho are with Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, R.O.C. (e-mail: pei@cc.ee.ntu.edu.tw).

C. M. Cheng is with Telecommunication Lab, Chunghwa Telecommunication Co., Taiwan, R.O.C. (e-mail: cmc@ms.chttl.com.tw).

Publisher Item Identifier S 1051-8215(00)07557-1.



(a) (b)

Fig. 1. (a) Procedures of color space partition by the DSQ. (b) Partition of 2-D signal space by the DSQ.

proposed method extracts key color frames from compressed video data in order to alleviate the computation burden and storage usage. Besides, to display longer sequences without screen-flicker problem, we propose a shifting-window scheme.

This paper is organized as follows. In Section II, the DSQ algorithm is briefly introduced. In addition, we describe the proposed approach for compressed images, which uses the reduced image rather than the whole image for the color-palette design to avoid the heavy computation in image decompression. In Section III, the technique of extracting key color frames for compressed videos is presented and a shift-window scheme is proposed to solve the screen-flicker problem. Section IV reports the simulation results of proposed methods Finally, a conclusion is made in Section V.

II. Limited Color Display for Compressed Image

Concerning compressed image, JPEG [4] is the most widely accepted picture compression standard, which not only provides good quality, but also the compression ratio is high. When displaying any JPEG image on machines with limited color-display capability like most SUN workstations, the default color palette is usually not suitable. But the design of the color palette requires the original image, which is not available for JPEG images without decompression. The procedures of decompression also require lots of computations and much space for storage. Therefore, it is essential to design schemes which use few parts of decompression to extract the color palette.

In this section, we will explain two methods of color-palette design for compressed image, which are the extensions of the DSQ [3]. Before the descriptions of proposed methods, the DSQ is briefly introduced.

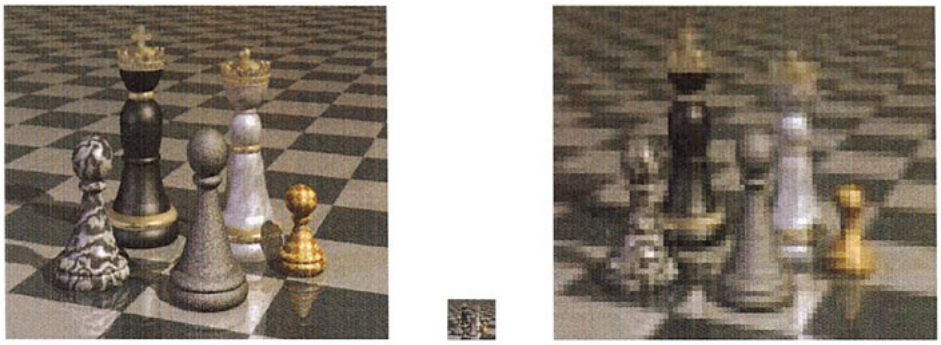
A. Review of DSQ

The DSQ is designed to quantize color images with both good quality and fast speed. As displayed in Fig. 1(a), the DSQ partitions color space of an image in a dependent way in order to fully utilize the correlations of color components of an image. The partition used is the binary moment preserving (MP) thresholding technique. The DSQ includes two stages: 1) bit allocation for different color components and 2) recursive binary MP thresholding. The purpose of bit allocation is to let every bit in the designed color palette provide as much information as possible. Usually, the wider distribution of color values in one color component, the more bits are needed to this component. To find suitable thresholds partitioning color space and representative colors of an image, the DSQ employs recursive binary MP thresholding. The principle is that for one specific color component I_i, the total pixels are divided into two classes using the binary MP principle. These two classes are then recursively divided into two subclasses until the number of bits B_i allocated to this color component is reached. The binary MP thresholding is based on the idea that after thresholding, the moments up to some order are kept the same.

Now we consider the case using binary MP thresholding to partition color component I_i into 2^{B_i} intervals. The range of the color component I_i, with boundary values 0 and $L-1$, is first partitioned into two intervals $[0, t_i(1)]$ and $[t_i(1), L-1]$, where $t_i(1)$ represents the threshold obtained by first binary MP thresholding. Then the interval $[0, t_i(1)]$ is partitioned into two subintervals $[0, t_i(2))$ and $[t_i(2), t_i(1)]$, where $t_i(2)$ is the threshold obtained by second binary MP thresholding. Similarly, $[t_i(1), L-1]$ is partitioned into two subintervals $[t_i(1), t_i(3)]$ and $[t_i(3), L-1]$ by third binary MP thresholding. If we continue the procedure recursively by using the binary MP thresholding $r_i = 2^{B_i} - 1$ times and order the resultant thresholds according to their values, the output thresholding levels $t_i(1), t_i(2), t_i(3), \ldots, t_i(r_i)$ will be determined, i.e., $t_i(l) \leq t_i(m)$ for $l \leq m$.

The steps to design a DSQ color palette are concluded as follows.

1) Use the bit-allocation procedure to find out the total thresholding levels in each color component.
2) Do the recursive binary MP thresholding on I_1 and calculate the threshold levels $t_1(l)$, $1 \leq l \leq r_1$.
3) Do the following steps $r_1 + 1$ times.
   a) Perform the recursive binary MP thresholding on I_2 in those pixels falling in the span $[t_1(l), t_1(l+1)]$ and calculate the associated thresholding levels $t_2(l, m)$, $l \leq m \leq r_2$.
   b) Do the following step $r_2 + 1$ times.
      Apply the recursive binary MP thresholding on I_3 in the pixels within the long bar limited by $[t_1(l), t_1(l+1)]$ and $[t_2(l, m), t_2(l, m+1)]$



(a) (b) (c)

Fig. 2. (a) Decoded JPEG image. (b) Its DC image. (c) Enlarged DC image.

and compute the associated thresholding levels $t_3(l, m, n), l \leq n \leq r_3$.

4) Assign the centroid of color points inside the cube formed by step 2 and 3 as a representative color.

The total number of binary MP thresholding needed in the DSQ is $r_1+(r_1+1)r_2+(r_1+1)(r_2+1)r_3 = M-1$. A covering of quantization in 2D signal space is illustrated in Fig. 1(b). As we can see, the DSQ will cluster color points appropriately and not waste any cell.

B. Design of Color Palette from the DC Image

The step which needs most computations in decoding a JPEG image is inverse discrete cosine transform (IDCT). If we design the color palette for a JPEG image in an uncompressed domain, we must handle this computation-intensive step, IDCT. Moreover, in doing it this way, the data size of the uncompressed image is much larger than that of the original compressed image. These drawbacks become more serious when several JPEG images desire limited-color displays in the multi-tasking window system like Window95 or XWindow. To overcome these troublesome problems, we must find a method which uses a minimum of JPEG decompression procedures to extract the suitable color palette in the compressed domain.

The proposed method exploits the DC image rather than full-resolution image for finding color palette. The DC image is the image consisting of DC coefficients of all 8×8 blocks in the image. The most important advantage of doing this is the low computation complexity, as compared to fully decoding the image. IDCT is not longer necessary and even the zigzag scanning or run-length decoding can be skipped for extracting the DC image. Although the fine spatial details of the image are lost in the DC image, the most important feature required in designing the color palette, the color characteristics of the image, is kept. As shown in Fig. 2, it is obvious that the DC image reserves the most significant colors in the original image, which are needed for acceptable limited-color display. The DC image can also be utilized in other compressed domain applications. Nakajima [10] has used the DC image performing scene-change detection of video data.

The procedures to extract the DC image of the JPEG image are shown in Fig. 3. The extracted DC image is then processed by the DSQ to get the color palette. The obtained color palette can be stored in the header of the JPEG image. The insertion of

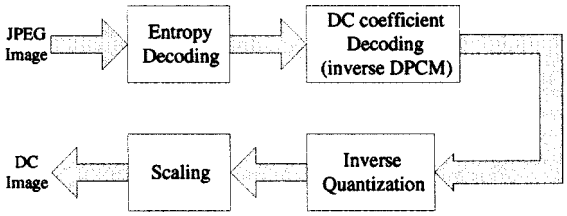


Fig. 3. Extraction of the DC image from a JPEG image.

color palette in the header of JPEG image indeed increases the output bit-rate. However, this increment is not significant. For example, we consider the following case of compressed image:

1) the original color image with size 512×512 being compressed with ratio $12:1$;
2) the design of color palette generating 256 representative colors and the bit allocation via the DSQ algorithm to three color components being (2, 3, 3).

Then, the header of JPEG image should allocate $256 \times 3 = 768$ bytes for representative colors and $2^2 + 2^2 \times 2^3 + 2^2 \times 2^3 \times 2^3 = 292$ bytes for the partition boundary generated by the DSQ algorithm. This allocation will increase the bit rate as $(768 + 292) * 4/(512 \times 512) = 0.016$.

On receiving the JPEG image, the receiver can decide whether it is using the color palette for better limited-color display quality or not. If the color palette extracted by the proposed scheme is selected, we utilize the boundaries of color cubes partitioned by the DSQ to obtain the displayed color of each pixel of the decoded JPEG image. The search of the associated representative color for each pixel is based on the well-organized tree structure of color space partition by the DSQ. It takes at most $(r_1+1)+(r_2+1)+(r_3+1)$ comparisons. Finally, scaling is needed because the DC coefficient $F(0, 0)$ is for an 8×8 block, which is eight times larger than the real DC value.

C. Extension to the AC Image

If the image size of the JPEG image is too small, the extracted DC image might not contain enough color information in the original image. Also, in some applications, the original image is too large to be analyzed efficiently. It is desirable to analyze a small-size image rather than the original image of full resolution. Thus, the DC image is not suitable for the color-palette

(a) (b) (c)

Fig. 4. (a) Decoded JPEG image. (b) Its enlarged 1DC + 3AC image. (c) Its enlarged 1DC + 2AC image.

design. In these situations, we observed from several experiments that the AC image, which is constructed by DC and three AC coefficients ($F(0, 1)$, $F(1, 0)$, and $F(1, 1)$) of all 8×8 blocks in the JPEG image, can provide enough color information. This image is designated as the 1DC + 3AC image in this paper. These coefficients are selected because they can generate the AC image easily. The inverse two-by-two DCT is given by

$$\begin{cases} f'(0, 0) = \frac{1}{2}[F(0, 0) + F(0, 1) + F(1, 0) + F(1, 1)] \\ f'(0, 1) = \frac{1}{2}[F(0, 0) - F(0, 1) + F(1, 0) - F(1, 1)] \\ f'(1, 0) = \frac{1}{2}[F(0, 0) + F(0, 1) - F(1, 0) - F(1, 1)] \\ f'(1, 1) = \frac{1}{2}[F(0, 0) - F(0, 1) - F(1, 0) + F(1, 1)] \end{cases} \tag{2}$$

where $f'(i, j)$ with $i = 0, 1$ and $j = 0, 1$ are the values of the pixels of each block in the 1DC + 3AC image.

But one noticeable thing is that the AC coefficient $F(1, 1)$ is usually zero or much smaller than the other three coefficients. Therefore, in the 1DC + 3AC image, the influence of the coefficient $F(1, 1)$ could be ignored in most situations. In this way, we choose the 1DC + 2AC image rather than 1DC + 3AC image for designing the color palette. In getting the 1DC + 2AC image, (2) is still applicable, with $F(1, 1) = 0$. One advantage of using 1DC + 2AC image is that the needed run-length decoding is fewer than that in 1DC + 3AC image. One example of 1DC + 3AC and 1DC + 2DC image is shown in Fig. 4. Comparing Fig. 2 with Fig. 4, it is very obvious that more details are kept in the 1DC + 3AC and 1DC + 2AC images. Furthermore, the 1DC + 2AC image is almost the same as the 1DC + 3AC image. Hence, the 1DC + 2AC image is more preferable when the image size is too small or more details in the image are needed.

Compared with the extraction of the DC image, the computations of getting the 1DC + 3AC image or 1DC + 2AC image in (2) are very simple, too. They only involve additions and shifts, which are most desirable in either software or hardware implementations. The only price paid is getting those AC coefficients. The AC coefficients could be acquired by several run-length decoding steps which can be done by using a look-up table. The procedures of extracting the 1DC + 3AC image or 1DC + 2AC image are shown in Fig. 5.

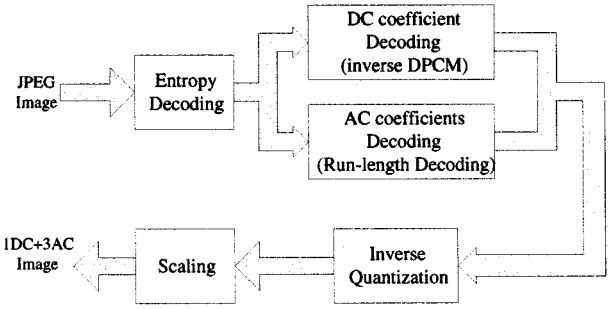


Fig. 5. Extraction of the 1DC + 3AC or 1DC + 2AC image from a JPEG image.

III. Limited Color Display for Compressed Video

Now more and more video clips are able to be accessed in some World Wide Web databases. However, with the limit of bandwidth of internet, those video clips are usually compressed by using MPEG [6], [7] or H.26x [5] format. In this condition, how to analyze the video clips to design a suitable color palette for those machines with limited color display becomes important. In this section, we will introduce methods of color-palette design for MPEG compressed video.

Similar to JPEG images, the low-resolution DC images of MPEG video, called DC sequences, are first extracted for color-palette design. Additionally, to avoid huge computation costs, we will only apply the DSQ to those DC images, called key color frames, which contain the major color information of the entire DC sequence, for obtaining the color palette of MPEG video. From the observation of DC sequences, we noticed that color information inside a frame is unchanged for most of the sequence except key color frames, which consists of color changes in the sequence. Thus, detection of color changes is essential for finding key color frames. The detection of key color frames is based on two steps and introduced in the following subsection:

1) detection of potential key color frames;
2) detection of key color frames and extraction of color palette.

Besides, if only one fixed color palette is used, the degradation would get worse and worse as the sequence length gets longer. To overcome this problem, a multiple color-palette scheme called shifting-window scheme is proposed to display compressed video with good quality even when the sequence length is getting longer.

A. Extraction of DC Sequence

We will use MPEG-1 video as the example to illustrate the process of DC sequence extraction. In MPEG-1, the DC coefficient of the DCT block in an I frame is the average of pixels in the block. The DC image for an I frame can be easily formed by getting the DC coefficient in every block. However, the challenge exists in extracting DC images from P or B frames, which use motion compensation to exploit the temporal redundancy. A general case for a P frame has been proposed by Yeo and Liu [11] and is shown in Fig. 6.

Here, B_{ref} is the block with motion vector (mvx, mvy) pointing to the current block of interest and B_1, B_2, B_3 and B_4 are the four neighboring blocks which derive the reference block B_{ref}. We can express the DC component of DCT coefficients of B_{ref}, denoted as DCT(B_{ref}), in the following equation:

$$(\text{DCT}(B_{\text{ref}}))_{00} = \sum_{i=1}^{4} \sum_{m=0}^{7} \sum_{l=0}^{7} w_{ml}^{i} (\text{DCT}(B_i))_{ml} \quad (3)$$

where w_{ml}^{i} are weighting factors related to the motion vector.

This precise calculation of DCT coefficients is time-consuming. Since we are only interested in the DC component, the first-order approximation is used rather than precise calculation

$$(\text{DCT}(B_{\text{ref}}))_{00} = \sum_{i=1}^{4} \frac{w_i h_i}{64} (\text{DCT}(B_i))_{00} \quad (4)$$

where w_i and h_i are the overlapping width and height of B_{ref} in block B_i. It can be shown that $w_i h_i/64$ corresponds to w_{00}^{i} in (3), and this is why it is called a first-order approximation. Although the approximation error will accumulate, the error is acceptable in most cases because the GOP size is usually small and the error will reset to zero at every I frame. The other advantage of this approximation is that it requires only the motion-vector information and the DC values in the reference frames. This approximation approach can also be applied to a B frame where two reference frames might be needed. The problem of half-pixel-wise prediction can be solved by using the average of larger blocks depending on the motion vector. The 9×9 block is needed if half-pixel-wise prediction occurs in both x and y direction. The 8×9 block is used for half-pixel accuracy only in the x direction and the 9×8 block is for y direction only.

B. Key Color-Frame Detection for Compressed Video

After the DC sequence is derived from an MPEG video, we use its DC images for key color-frame detection. To represent color information of a frame, the hue component of HSV space [13] is adopted in this paper. The hue component has been widely accepted as a good candidate of representing color difference. Using this representation, we compare the normalized difference of hue histogram between consecutive frames in order to detect boundaries between consecutive color changes. The principle behind this is that two frames having a unchanging background and objects will show little difference in their corresponding hue histograms. The normalization procedure is utilized for reducing the impact of noise imposing

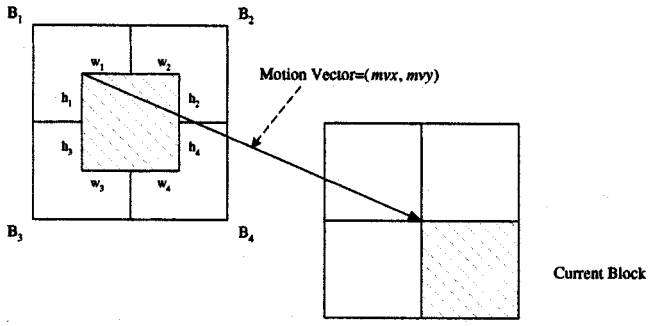


Fig. 6. Current block, motion vector, and reference block.

on the hue histograms of consecutive frames. The normalized hue difference between the hue histograms of the lth and $(l-1)$th frame, S_l, is given by the following equations:

$$S_l = \sum_{j=1}^{L} \text{NHD}_{l,\,l-1}(j) \quad (5)$$

where

$$\text{NHD}_{l,\,l-1}(j) = \begin{cases} H_l^3(j), & \text{for } H_{l-1}(j) = 0 \\ \left| \dfrac{H_l(j) - H_{l-1}(j)}{\min(H_l(j),\, H_{l-1}(j)) + 1} \right|, & \text{otherwise} \end{cases} \quad (6)$$

with $H_l(j)$ and $H_{l-1}(j)$ being hue histograms of the two consecutive frames, respectively, and L of (5) being the number of hue component bins in comparison. In (6), we choose the minimum value of $H_l(j)$ and $H_{l-1}(j)$ to normalize the hue difference. The condition $H_{l-1}(j) = 0$ is set to reflect the situation when pixels with the certain hue value j exists in frame l, but not in frame $l-1$.

Similar to luminance change of the DC sequence, we have observed that color change is also a local activity which involves details regarding several neighboring frames. For scene-change analysis, Yeo and Liu [9] have suggested setting the threshold of luminance change in order to match the local activity. We adopt this approach to detect color changes in this paper and choose a sliding-window thresholding technique proposed by Yeo and Liu [9] to avoid false alarms which might occur in camera operations or object changes. In this technique, $2n-1$ frames with $2n-2$ hue differences are examined in a local range. Inside this local window, a color change from $(l-1)$th to lth image occurs if the following conditions are satisfied.

1) The difference is the maximum with the window, i.e., $S_j \le S_l$, $j = l-n+1, \ldots, l-1, l+1, \ldots, l+n-1$.
2) S_l is also m times of the second maximum in this window.

After examination of each window, the window is shifted one frame to prepare the next examination until the whole sequence is processed. In criterion 1, the parameter n is set to be smaller than the minimum duration between two color changes, but large enough to avoid false alarms. This is because as the window size gets smaller, the threshold is closer to be a global approach which is unfavorable for color-change detection. If we set $n = 30$ for

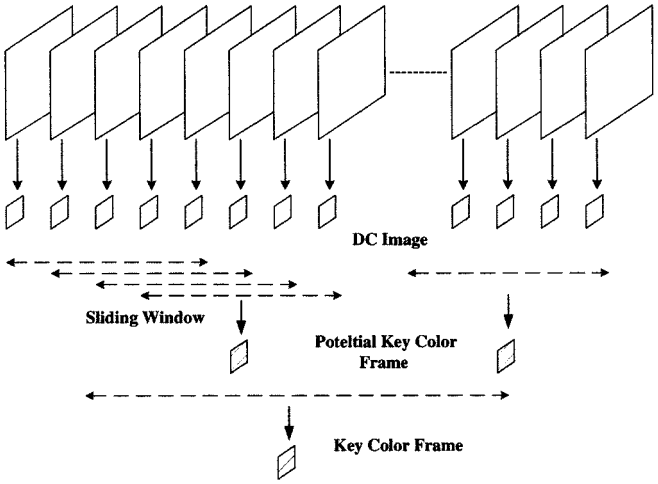


Fig. 7. Detection of key color frame in compressed domain.

a 30 frames/s video sequence, it means that there cannot be two color changes within a second. The parameter m in criterion 2 is imposed to guard against some camera operations such as fast panning or zooming. For these operations, the hue differences S_l would maintain consecutive peaks across several frames. From experimental results, we understand that the design of m depends on the tradeoff between increasing the detection rate and decreasing the false alarm rate. It has been found that the values of m varies from 2.0 to 4.0 give good results.

Through the above sliding-window thresholding scheme, the detected frames are called potential key color frames. From experimental results, we observed that there are redundant false-alarmed frames, which do not contain significant color information, inside these potential key color frames. Then, we adopt a coarse-to-fine strategy to eliminate those false-alarmed frames. In this strategy, these potential key color frames is processed one more time by the sliding-window thresholding scheme. After this examination, the detected frames are desired key color frames which are then used by the DSQ for the extraction of color palettes. We illustrate the proposed scheme of detecting key color frames for compressed video in Fig. 7.

To do limited-color display with the color palette extracted by the proposed scheme, we employ the same procedures as for the compressed image. Using the well-organized boundaries of color cubes partitioned by the DSQ, pixels of each image of decoded MPEG sequence are mapped to their associated representative colors.

C. Shifting-Window Scheme

If only one fixed color palette is assigned to the whole sequence, the performance of color quantization is getting worse as the sequence length gets longer. On the other hand, if a color palette is designed for every key color frame, a serious visual artifact, screen flicker, may occur when the color palette is changed for these key color frames [12]. This problem happens because there is a sudden change of images colors. When the frame buffer of the display contains an image, a new color palette which belongs to the next image is already active. This phenomenon of screen flicker is sharp and unpleasant to human eyes.

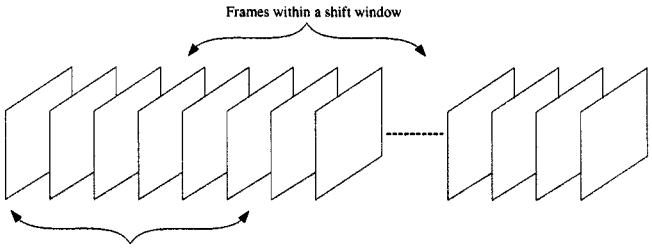


Fig. 8. Proposed shifting-window scheme for color quantization of MPEG videos.

TABLE I
APSNR OF DECODED JPEG IMAGE, DECODED IMAGE QUANTIZED BY PROPOSED METHOD USING THE DC IMAGE AND DECODED IMAGE QUANTIZED BY THE ISQ

Image (size)	decoded image(dB)	proposed method using the DC image(dB)	decoded image using the ISQ(dB)
Boats(512x512)	34.8770	32.2286	26.0690
Jet(512x512)	33.0118	30.5679	25.4180
Lena(512x512)	33.0205	31.3977	24.7092
Toys(720x480)	35.7203	32.6753	25.7581

TABLE II
APSNR OF DECODED JPEG IMAGE, DECODED IMAGE QUANTIZED BY PROPOSED METHODS USING THE DC IMAGE, THE 1DC + 3AC IMAGE AND THE 1DC + 2AC IMAGE

Image (size)	decoded image(dB)	proposed methods using		
		DC image	1DC+3AC image	1DC+2AC image
Lena(128x128)	31.5709	28.3087	30.3191	30.4965
Lena(256x256)	32.8243	30.6361	31.5175	31.4630
Boats(240x160)	32.4880	29.7418	30.8629	30.8170
Chess(320x240)	33.0556	28.9017	30.9755	30.5808

To solve the screen-flicker problem, we still use the DSQ to design color palettes for the sequence, but a color palette is designed for each fixed-length shifting-window in the sequence. The procedures of color-palette design in the video sequence of each window are the same as mentioned above. The key color frames in each shifting window are detected and applied to the DSQ for the color-palette design as depicted in Fig. 8. These windows contains overlapping frames, which cause the color distribution to not vary too much from window to window even if the color change occurs. As a result, the DSQ can generate smoothly varying color palettes for the sequence if the bit-allocation procedure is fixed. In addition, since every entry of the color palette of the processed window would not differ significantly from that of the next window, screen flicker is greatly reduced.

IV. EXPERIMENTAL RESULTS

To illustrate the performance of proposed color-quantization methods for compressed image, four different RGB color images ("Boats," "Jet," "Lena," and "Toys") are tested. These images are first compressed by JPEG baseline compression. Then the color palette of 256 colors is designed by the DSQ from the

Fig. 9. "Lena" image. (a) Original image. (b) Decided JPEG image. (c) Decoded image quantized by proposed method using the DC image. (d) Decided image quantized by the ISQ.

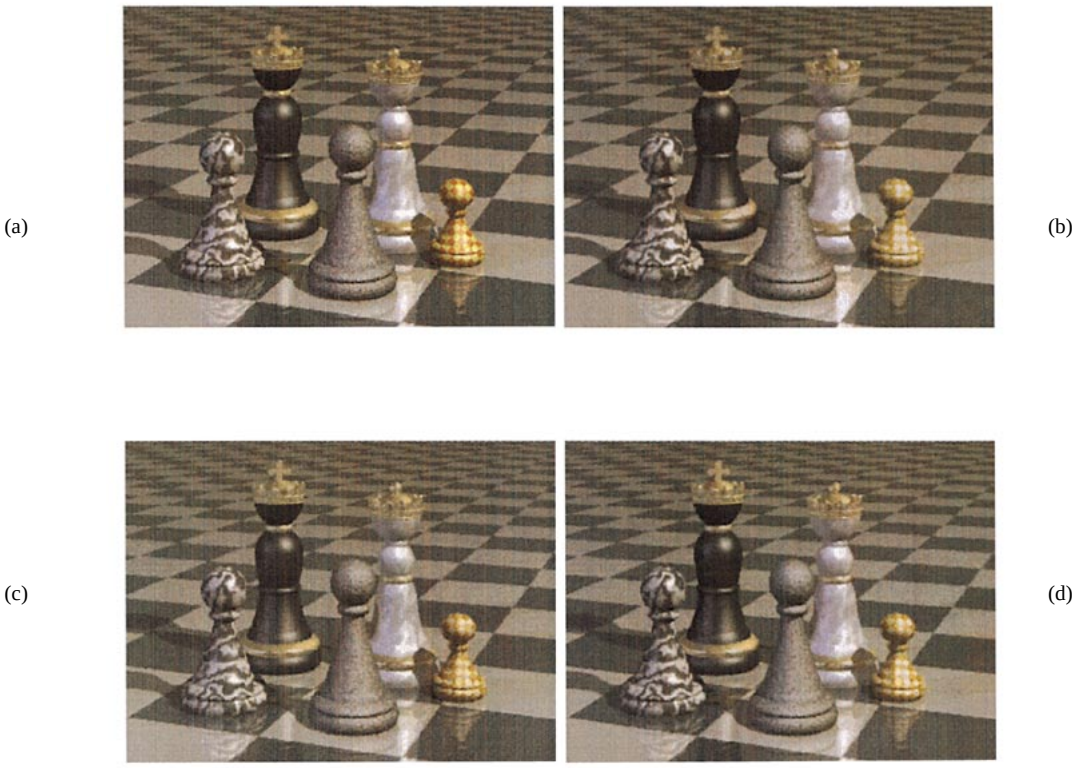


Fig. 10. "Chess" image (320×240). (a) Original image. Decoded images quantized by proposed methods using (b) the DC image, (c) the 1AC + 3AC image, and (d) the 1DC + 2AC image.

extracted the DC image, DC+ 2AC image, or DC + 3AC image. To evaluate the performance, the average peak signal-to-noise-ratio (APSNR) is used. The APSNR criterion is given by

$$\text{APSNR} = 10\log\frac{3 * N * 255^2}{\text{TSE}} \tag{7}$$

where TSE is the total square error of three color components between the original and quantized images. The results of proposed method using the DC image are shown in Table I. For comparison, the APSNRs of the decoded JPEG image and its color-quantized image by the independent scalar quantization (ISQ) [14], which subdivides each color component individu-

(a) (b) (c) (d) (e)

Fig. 11. The DC images of detected key color frames. (a) Frame 0. (b) Frame 3. (c) Frame 90. (d) Frame 195. (e) Frame 210.

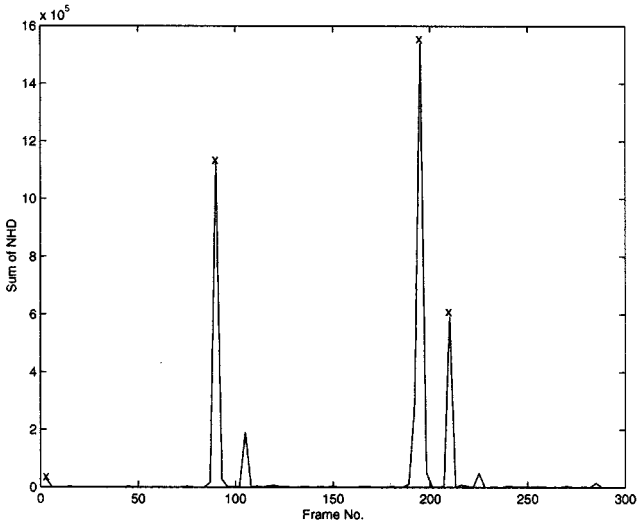


Fig. 12. Sum of hue difference plot of "News": S_l versus l.

ally with a fixed number of levels, are also included in Table I. The resultant images for "Lena" are displayed in Fig. 9. It is noticed that the performance of the proposed method is above 30 dB on average and better than that of using the ISQ. Although some other better color-quantization methods except the ISQ can be applied to the decoded JPEG image, they usually need extra heavy computation to obtain the color palette. This would slow the speed of limited-color displaying process. For instance, if the DSQ is applied to the decoded JPEG image "Lena" with size 512×512, the APSNR value of color-quantized image is 32 dB. However, the computation time to obtain the color palette is 12 s when using a SUN SPARC20 workstation. This computation time is larger than those of proposed schemes, which are 1.3, 2.3, and 2.4 s for using the DC image, the DC + 2AC image, and the DC + 3AC image, respectively, when the same image and workstation are tested. On the other hand, since the proposed schemes has extracted the color palette from the compressed domain in advance, the displaying process of the decoded image is fast.

Then we chose four other small-sized images to demonstrate the performance of other proposed methods. The empirical results are shown in Table II. Normally, the performance of using the 1DC + 2AC image is better than that of using the DC image. For the 128×128 "Lena" image, because the DC image is only 16×16, which is exactly the size of the color palette, so the performance gain with the 1DC + 2AC image is very obvious. But as the image size gets larger, the gain becomes less clear. The small difference between the 1DC + 3AC and the 1DC + 2AC images for color-palette design can be seen in these results. Furthermore, we show the original and decoded JPEG images with color quantization for "Chess" in Fig. 10 to exhibit the above observations.

To evaluate the performance of the proposed method for compressed videos, a test sequence "News," which has 300 YUV frames with size 352×240, is used. The sequence consists of three main video segments which are news conference of a president candidate, a TV news reporter and news of a tropical storm. There are special effects of dissolving, fading in and fading out existed within each transition of two segments. This test sequence is first compressed by MPEG-1 compression. Then the proposed scheme is applied to design a color palette of 256 colors on the YUV color space. The color components were quantized in the order of Y(Luminance) first, then U and V last. The number of bits in each color component after bit allocation in the DSQ are 4(Y), 2(U), and 2(V).

The detected key color frames from the extracted DC sequence are frame 0, 3, 90, 195, and 210 which are shown in Fig. 11. The parameters of criterion 1 and 2 used to find a color change are set to be 7 and 2, respectively. The sum of hue difference in the first step of scheme of detecting key color frames is plotted in Fig. 12. As we can see, these frames indeed represent significant color difference. Frame 210 is detected because the yellow caption appears. Fig. 13 shows the selected original frames 30, 120, and 220 and their decoded MPEG-1 frames quantized by the proposed scheme with the color palette designed from above key color frames. We plot in Fig. 14 the PSNR distribution of luminance component of the decoded MPEG-1 sequence and the decoded sequence quantized by the proposed scheme. In this figure, the average PSNR of the decoded MPEG-1 sequence is 33.9805 dB and that of the proposed scheme is 32.4647 dB, which shows only about 1.5 dB lost on average in the color quantization. When we analyze Fig. 14, it is noticed that some peaks or intra-frames among frame 90 and frame 210 have about 5 dB lost between PSNR values of the decoded MPEG-1 sequence and the proposed scheme. Normally, decoded intra-frames of the MPEG-1 sequence are good quality, since no motion estimation is involved. For these intra-frames, it shows that the matching of colors of the decoded MPEG-1 frame with those of the original frame is better than that of the representative colors obtained from the proposed scheme. However, since the PSNR values of the proposed scheme for these frames are around 35 dB, we do not perceive significant degradation of picture quality in experiments. The color palette obtained by the proposed scheme is shown in Fig. 15(a). Concerning the computation time, the proposed method took about 8.8 s to extract a color palette for the test sequence when using a SUN SPARC20 workstation.

Fig. 13. Original and quantized frames of "News." (a), (c), and (e) are original frame 30, 120, and 220, respectively, and (b), (d), and (f) correspond to quantized frame 30, 120, and 220, respectively.

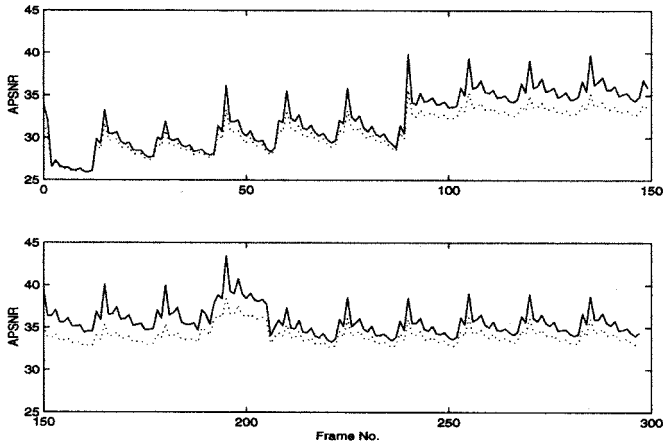


Fig. 14. PSNR in the sequence "News." The solid curve is for the decoded MPEG-1 sequence and the dotted curve corresponds to the proposed scheme.

We also executed the shifting-window scheme with the size of shifting-window being 150 and the size of overlapping frames between neighboring windows being 75 on the test sequence. This configuration results in three shifting windows to cover the test sequence. The designed color palettes for frames 0–149, 75–225, and 150–299 are shown in Fig. 15(b)–(d). As we can see, the color palette of frames 150–299 contains more dark colors, which appear in the video clip of the tropical storm. The gradual changes can be observed among these color palettes. When the corresponding quantized sequence is played back with frames 0–74 using palette of Fig. 15(b), frames 75–224 using palette of Fig. 15(c), and frames 225–299 using palette of Fig. 15(d), we have seen that screen-flicker phenomenon is insipid and acceptable to human eyes.

V. Conclusion

In this paper, we have proposed color-quantization methods for compressed image and video, which are based on block-wise 2D-DCT operations. The proposed methods adopt the DSQ, which compromises complexity and quality, to design the color palette. For the compressed image (mainly JPEG image), the proposed method can use the DC or 1DC + 2AC image to design the color palette. The main advantage of this method is the low computational load as compared to the traditional methods which need to decode the whole image. As for the compressed video (basically MPEG video), the same technique for the compressed image is adopted. We design the color palette

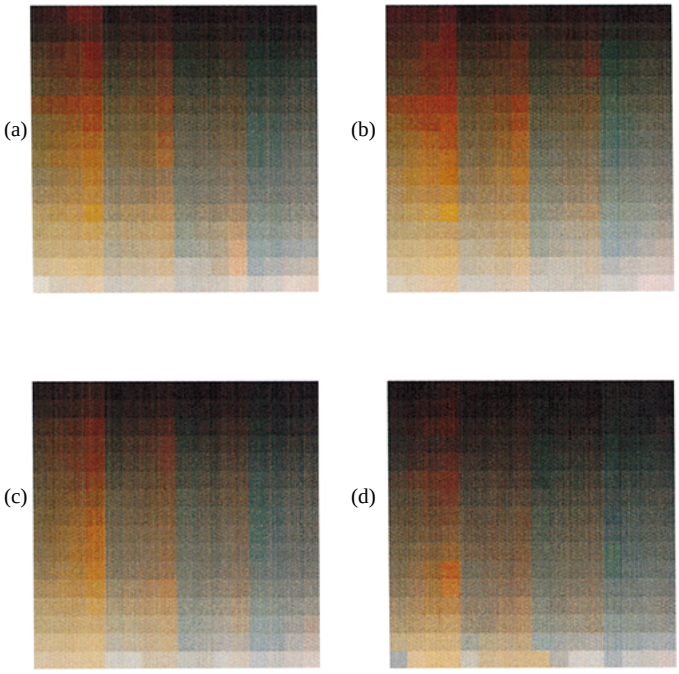


Fig. 15. (a) Color palette for the sequence "News" when only one color palette is used. (b)–(d) Color palettes for the same sequence for the shifting-windows scheme.

from key color frames of the DC sequence rather than the whole sequence in order to reduce the computation complexity. The decoded images quantized by the proposed methods for compressed image and video are acceptable to human eyes. Finally, a shifting-window scheme to solve screen-flicker problem is proposed. As shown by experimental results, we understand that this scheme can reduce the screen-flicker problem.

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers who made many useful comments. Their help is gratefully appreciated.

REFERENCES

[1] P. Heckbert, "Color image quantization for frame buffer display," *Comput. Graph.*, vol. 16, no. 3, pp. 297–397, July 1982.
[2] G. Braudaway, "A procedure for optimum choice of a small number of colors from a large color palette for color imaging," in *Proc. Electronic Imaging '87*, San Francisco, CA, 1987.
[3] S. C. Pei and C. M. Cheng, "Dependent scalar quantization of color images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 124–139, Apr. 1995.
[4] G. K. Wallace, "The JPEG still picture compression standard," *Comm. ACM*, vol. 34, no. 4, pp. 30–44, Apr. 1991.
[5] "Video Codec for Audiovisual Services at Px64 kbits,", Draft Revision of the CCITT recommendation H.261, WP XV/1 Rep. Part II, Dec. 1989.
[6] "MPEG Video Committee Draft,", ISO-IEC/JTC1/SC29/WE11/MPEG 90/176, Dec. 1990.
[7] D. Le Gall, "MPEG: A video compression standard for multimedia application," *Comm. ACM*, vol. 34, no. 4, pp. 46–58, Apr. 1991.
[8] R. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, pp. 4–29, Apr. 1984.
[9] B. L. Yeo and B. Liu, "Rapid scene analysis on compressed videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 533–544, Dec. 1995.
[10] Y. Nakajima, "A video browsing using fast scene cut detection for an efficient networked video database access," *IEICE Trans. Inform. and Syst.*, vol. E77-D, no. 12, pp. 1355–1364, Dec. 1994.
[11] B. L. Yeo and B. Liu, "On the extraction of DC sequences from MPEG compressed video," in *Proc. Int. Conf. on Image Processing*, vol. 2, Oct. 1995, pp. 260–263.
[12] E. Roytman and C. Gotsman, "Dynamic color quantization of video sequences," *IEEE Trans. Visual. Comput. Graphics*, vol. 1, pp. 274–286, Sept. 1995.
[13] W. K. Pratt, *Dig. Image Processing*. New York: Wiley, 1991.
[14] J. D. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes, *Comput. Graphics: Principles and Practice*. Reading, MA: Addison-Wesley, 1990.

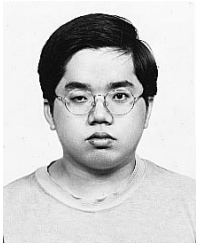
Soo-Chang Pei (S'71–M'86–SM'89–F'00) was born in Soo-Auo, Taiwan, in 1949. He received the B.S.E.E. degree from the National Taiwan University, Taipei, Taiwan, in 1970, and the M.S.E.E. and Ph.D. degrees from the University of California at Santa Barbara in 1972 and 1975, respectively.

He was an Engineering Officer in the Chinese Navy Shipyard from 1970 to 1971. From 1971 to 1975, he was a Research Assistant at the University of California at Santa Barbara. He was Professor and Chairman in the Electrical Engineering Department for both Tatung Institute of Technology and National Taiwan University during 1981–1983 and 1995–1998, respectively. Presently, he is the Professor of the Electrical Engineering Department, National Taiwan University. His research interests include digital signal processing, image processing, optical information processing, and laser holography.

Dr. Pei is a member of Eta Kappa Nu and the Optical Society of America.

Ching-Min Cheng was born in Taipei, Taiwan, in 1959. He received the B.S.E.E. degree from the National College of Marine Science and Technology, Keelung, Taiwan, in 1982, and the M.S.E.E. degree from the University of California at San Diego in 1986. In 1996, he received the Ph.D. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan.

From 1983 to 1984, he was an Engineering Officer in the Chinese Airforce Anti-Aircraft Corps. From 1986 to Aug. 1989, he served as a Patent Examiner in the National Bureau of Standards. Since September 1989, he has been with Telecommunication Labs, Ministry of Communications, Taiwan, as a Research Engineer. His research interests includes digital signal processing, video compression, and multimedia communication.



Lung-Feng Ho was born in Keelung, Taiwan, in 1974. He received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1996 and 1998, respectively.

He is currently serving in the Army. His research interests include image and video processing.