IEEE Int. Conference Neural Networks & Signal Processing
Zhenjiang, China, June 8~10, 2008

# A LOSSLESS CODING SCHEME FOR ENCODING COLOR-INDEXED VIDEO SEQUENCES

*Yuk-Hee Chan*

Centre for Signal Processing
Dept of Electronic and Information Engineering
The Hong Kong Polytechnic University
Email: enyhchan@polyu.edu.hk

## ABSTRACT

A new lossless video compression scheme for coding sequences of color-indexed frames is proposed. This scheme handles the frames sequentially. It reduces both the zero-order entropy and the energy of the index map of a specific frame by turning the index map into a new index map each element of which serves as an index to a pixel-dependently reordered version of the palette. Accordingly, a matching coding scheme can be exploited to encode the new index map effectively. A technique similar to the one used in animated GIF is also used to remove the inter-frame redundancy. Simulation results show that the proposed method is superior (in terms of compression ratio) to other state-of-art lossless compression algorithms in coding color-indexed video sequences.

***Index Terms*** — Lossless coding, video coding, color indexed images, color indexed sequences, palette reordering

## 1. INTRODUCTION

For historic and some other reasons, there are a lot of digital video sequences in multimedia applications which are made up with color-indexed image frames. As a well-known example, sequences in animated GIF format are still widely used nowadays to provide prompt decoding and guaranteed image quality in web applications. Unlike conventional full-color video sequences, these video sequences are palette-based. Each frame of them is represented with a color index map each element of which serves as an index to select a color from a predefined set of colors called palette to represent the color of a pixel in the frame[1]. Since two completely different colors can be of similar index values in a palette, compressing such a sequence must be done losslessly.

Accordingly, current lossy video compression standards such as H.264 are not suitable for handling sequences of color-indexed frames. Though there are some successful proposals for lossless video coding[2], they are basically proposed for coding full-color video sequences and hence are not able to handle the concerned sequences effectively and efficiently either.

A new lossless compression scheme for coding sequences of color-indexed frames is proposed in this paper. This scheme handles a video sequence frame by frame. In general, the coding of a frame can be decomposed into two stages. In stage 1, it raster scans the frame and, on the fly, reassigns indices to palette colors pixel by pixel adaptively based on (i) the predicted color of the current pixel and (ii) an updated statistical study of the processed image pixels. As a result, it is able to produce an index map of very low zero-order entropy and little energy. In stage 2, the resultant index map is further decomposed into a number of holey binary bit planes and then each of them is encoded with a context-based binary arithmetic coder. Simulation results show that the performance of the proposed compression scheme outperforms the other state-of-art lossless compression schemes in terms of output bit rate.

## 2. PIXEL-WISE PALETTE REORDERING

This section presents the method used in stage 1 to reassign indices to palette colors pixel by pixel adaptively. Since the index assignment of the palette is pixel-dependent, the reordering method is referred to as pixel-wise palette reordering method so as to discriminate it from the conventional palette reordering methods[3] in which a fixed palette is designed for all pixels to share.

Let the current input color-indexed image frame be $\mathbf{X}$ and the associated palette be $\Omega=\{\vec{c}_k \,|k=0,1\ldots N\text{-}1\}$, where $N$ is the size of the palette and $\vec{c}_k$ is the $k^{\text{th}}$ palette color in $\Omega$. Without lose of generality, we assume all $\vec{c}_k$ in $\Omega$ are sorted according to their luminance and $\vec{c}_0$ (i.e. $k$=0) is the one of the minimum luminance. Note that this criterion can be easily satisfied through an initialization process. This sorted palette is used as a reference palette in the codec.

Based on the index map of $\mathbf{X}$, a full-color image can be constructed with palette $\Omega$. The image is raster scanned and processed. For each pixel, the intensity values of its three color components are individually predicted with their own corresponding color planes by using a MED predictor.

978-1-4244-2311-8/08/$25.00 ©2008 IEEE.

MED is used in JPEG-LS and the details of its operation can be found in [4].

Suppose the prediction results of the red, the green and the blue color components of the current pixel $(i,j)$ are, respectively, $r(i,j)$, $g(i,j)$ and $b(i,j)$. In vector form, the prediction result of pixel $(i,j)$ is $\vec{v}(i,j)=(r(i,j),g(i,j),b(i,j))$. $\vec{v}(i,j)$ is then color quantized with palette $\Omega$. In practice, the quantization result could be different from the original color of pixel $(i,j)$. Without lose of generality, we assume that the quantization result and the original color of pixel $(i,j)$ are, respectively, the $p^{\text{th}}$ and the $r^{\text{th}}$ palette colors in $\Omega$. For the purpose of reference, they are denoted as $\vec{c}_p$ and $\vec{c}_r$ respectively.

In the proposed scheme, statistics about the frequency of the occurrence of specific events are collected when processing the image such that the scheme can learn from the experience to improve its reordering performance adaptively. Specifically, we build five simple probability models and then merge them to form a combined probability model. Five separate tables are constructed for building these five probability models.

The first table $\{T_d(m,n)|m,n=0,1\ldots N\text{-}1\}$ is referred to as *discrepancy frequency table* (DF-Table) as its entry $T_d(m,n)$ records the occurrences of encountering a pixel whose quantized predicted color and real color are, respectively, $\vec{c}_m$ and $\vec{c}_n$ up to the moment when pixel $(i,j)$ is processed. With this table, the conditional probability that $\vec{c}_k$ is the real color of pixel $(i,j)$ given that $\vec{c}_p$ is the quantized predicted color of the pixel can be estimated as

$$P_p(k\mid p)=T_d(p,k)/\Sigma_{u=0}^{N-1}T_d(p,u) \quad \text{for } k=0,1\ldots N\text{-}1 \quad (1).$$

All $T_d(m,n)$ values are initialized to 1 at the beginning of coding a frame and the DF-table is updated after a pixel is processed.

The other four tables, which are denoted as $\{T_N(m,n)|m,n=0,1\ldots N\text{-}1\}$, $\{T_W(m,n)|m,n=0,1\ldots N\text{-}1\}$, $\{T_{NW}(m,n)|m,n=0,1\ldots N\text{-}1\}$ and $\{T_{NE}(m,n)|m,n=0,1\ldots N\text{-}1\}$, are constructed to reflect how likely that a particular color in $\Omega$ is the real color of a pixel when the color of one of the four processed connected neighbors (i.e. the northern, the western, the northwestern and the northeastern) of the pixel is given. For example, $\{T_N(m,n)|m,n=0,1\ldots N\text{-}1\}$ records the occurrences of encountering a pixel whose northern neighbor's real color is $\vec{c}_m$ while its own real color is $\vec{c}_n$ up to the moment when pixel $(i,j)$ is processed. Entry values of all these tables are also initialized to 1 at the beginning of coding a frame and all tables are updated after a pixel is processed.

Without lose of generality, here we assume that the real colors of pixels $(i,j\text{-}1)$, $(i\text{-}1,j\text{-}1)$, $(i\text{-}1,j)$ and $(i\text{-}1,j\text{+}1)$ are, respectively, the $r_1^{\text{th}}$, the $r_2^{\text{th}}$, the $r_3^{\text{th}}$ and the $r_4^{\text{th}}$ palette colors in $\Omega$. Based on the four tables, four additional context-based probability models can be derived as

$$P_{c1}(k\mid r_1)=T_W(r_1,k)/\Sigma_{u=0}^{N-1}T_W(r_1,u)$$
$$P_{c2}(k\mid r_2)=T_{NW}(r_2,k)/\Sigma_{u=0}^{N-1}T_{NW}(r_2,u)$$
$$P_{c3}(k\mid r_3)=T_N(r_3,k)/\Sigma_{u=0}^{N-1}T_N(r_3,u)$$
$$P_{c4}(k\mid r_4)=T_{NE}(r_4,k)/\Sigma_{u=0}^{N-1}T_{NE}(r_4,u)$$
$$\text{for } k=0,1\ldots N\text{-}1 \quad (2).$$

Each of them tells how likely that $\vec{c}_k$ is the real color of pixel $(i,j)$ when the real color of a particular neighbor of pixel $(i,j)$ is given.

In our approach, when assigning a new color index to a palette color, probability models $P_p(k\mid p)$, $P_{c1}(k\mid r_1)$, $P_{c2}(k\mid r_2)$, $P_{c3}(k\mid r_3)$ and $P_{c4}(k\mid r_4)$ are merged to form a single probability model to determine the likeliness that the concerned palette color is the real color of the pixel currently processed. In particular, the combined probability model is given as

$$P(k)=L(k)/\Sigma_{u=0}^{N-1}L(u) \quad \text{for } k=0,1\ldots N\text{-}1 \quad (3),$$

where

$$L(k)=w_0T_d(p,k)+w_1T_W(r_1,k)+w_2T_{NW}(r_2,k)+w_3T_N(r_3,k)+w_4T_{NE}(r_4,k)$$
$$\text{for } k=0,1\ldots N\text{-}1 \quad (4).$$

The larger the value of $L(k)$, the more likely that $\vec{c}_k$ is the real color of pixel $(i,j)$.

Weighting factors $w_0$, $w_1$, $w_2$, $w_3$ and $w_4$ are used to weight the contribution of the five probability models according to their significance in the merging. They can be determined with a training process with a set of training images. We found that $\{w_0,w_1,w_2,w_3,w_4\}=\{4,2,1,2,1\}$ was a good choice in our simulation study.

It is possible that some neighbors of pixel $(i,j)$ are out of the image boundary. In such a case, corresponding items are removed from eqn. (4) when $L(k)$ is computed. For example, when one processes a pixel on the left boundary of an image, the western and the northwestern neighbors are missing and hence eqn. (4) becomes $L(k)=w_3T_N(r_3,k)+w_4T_{NE}(r_4,k)+w_0T_d(p,k)$.

Once $L(k)$ is determined for pixel $(i,j)$, the colors in palette $\Omega$ are adaptively reordered based on $L(k)$. In particular, $\vec{c}_k$'s are sorted according to the values of $L(k)$ for $k=0,1\ldots N\text{-}1$ in descending order. If there exist two different colors $\vec{c}_l$ and $\vec{c}_d$ such that $L(l)=L(d)$, $\vec{c}_l$ and $\vec{c}_d$ will be sorted according to their Euclidean distances to $\vec{c}_p$. The closer one is put in front of the other. If they are still not distinguishable, their order will be determined by their ranking in reference palette $\Omega$.

The position of $\vec{c}_r$ in the newly reordered queue can be used as an index to the queue and is used to represent the pixel in the output of the reordering method. Note the reordered queue forms a transient version of palette $\Omega$. After processing this pixel, $T_d(p,r)$, $T_W(r_1,r)$, $T_{NW}(r_2,r)$, $T_N(r_3,r)$

and $T_{NE}(r_4,r)$ are incremented by 1 to update the frequency count of corresponding events.

The algorithm proceeds to process next pixel in the scanning path by repeating the same procedures until all pixels are processed. The final result is a new index map each element of which serves as an index to a transient version of palette Ω.

In the decoder, to decode a pixel, the same process is carried out to determine the same transient version of palette Ω. As soon as the index for the pixel is received, it can be used to fetch the corresponding color in the transient version of palette Ω to reconstruct the full-color image frame directly.

The order of the algorithm's complexity is $O(NS_i)$ for a frame, where $S_i$ is the image size in terms of number of pixels and $N$ is the number of colors in the palette.

## 3. CODING OF REINDEXED INDEX MAPS

The processing results of stage 1 can be directly encoded with entropy coding technique. However, to compress the index map further, we propose to use a dedicated bit-plane coding scheme instead. In this scheme, a reindexed index map is decomposed into $N$-1 bit planes as follows.

$$B_k(i,j) = \begin{cases} 1 & if \;\; I(i,j) > k \\ 0 & if \;\; I(i,j) = k \\ \text{don't care} & if \;\; I(i,j) < k \end{cases} \quad \text{for } k=0,1,\ldots N\text{-}2 \qquad (5)$$

where $B_k(i,j)$ is the $(i,j)^{th}$ element of the $k^{th}$ bit plane and $I(i,j)$ is the new index value of pixel $(i,j)$. For reference, this approach of bit-plane separation is referred to as *value-based bit-plane separation* (VBS).

Starting from $k$=0, bit planes are gradually constructed as $k$ increases. Once a bit plane is defined, its bits are raster scanned and encoded with context-based entropy coding. In other words, bit planes of lower $k$ values are encoded first. If $B_k(i,j)$ is a don't care bit, there must be $B_l(i,j)$=0 for a particular $l<k$. Accordingly, the coding of a don't care $B_k(i,j)$ bit can be skipped as bit plane $l$ was encoded and the fact of $I(i,j)=l$ is known to the decoder.

As most indices in the output index maps are of values identical or close to zero and the distribution of the index values can be modeled with an exponential function, the number of don't care bits found in a particular bit plane increases significantly as $k$ increases.

Each holey binary bit-plane $B_k$ is encoded with a context-based binary arithmetic coder. Fig. 1 shows the general form of the context templates used in the proposed coding scheme. It is possible that the binary context of $B_k(i,j)$ contains some don't care pixel bits when $B_k(i,j)$ is encoded with context-based entropy coding. In that case, the don't care bits can be filled with either 0 or 1. In our realization, it is filled with 0 for simplicity.

As the value of $k$ gets larger, there are more don't care bits in bit plane $B_k$ and, accordingly, the context template covers more don't care bits when it moves around. To achieve better performance and avoid context dilution problem, a context template of smaller size is used to encode a bit plane of larger $k$. As shown in Fig. 1, the pixel positions of the context template are numbered. Instead of using all template locations, only first $L$ positions are used, where $L$ is a function defined as

$$L = \lceil 9 - \log_2(k+1) \rceil \qquad \text{for } k=0,1,\ldots N\text{-}2 \qquad (6).$$

In the suggested binary arithmetic coder, the forgetting factor α and the biasing constant Δ for updating the conditional probability for having bit '1', $P(1|context)$, are, respectively, 0.985 and 0.006. In particular, the probability for having bit '1' when the context is binary pattern $i$ again is updated by

$$P(1 \mid context = binary\ pattern\ i) = (t_i + \Delta)/(s_i + 2\Delta) \qquad \text{for } i=0,1\ldots 4095 \qquad (7)$$

where $t_i$ and $s_i$ are updated whenever a context of binary pattern $i$ is encountered using

$$t_i := \begin{cases} \alpha\, t_i + 1 & if\ the\ current\ pixel\ value\ is\ 1 \\ \alpha\, t_i & else \end{cases} \qquad (8)$$

and

$$s_i := 1 + \alpha\, s_i \qquad (9)$$

The initial values of $t_i$ and $s_i$ are, respectively, 1 and 2 for all context patterns $i$. The initialization is done at the beginning of coding a frame.

| | | | | |
|---|---|---|---|---|
| 11 | 8 | 6 | 9 | 12 |
| 7 | 3 | 2 | 4 | 10 |
| 5 | 1 | X | | |

X : the pixel of interest

Fig. 1 The context template used in bit plane coding

## 4. SIMULATION RESULTS

Simulations were carried out to evaluate the performance of the proposed coding scheme. Five testing video sequences were encoded with different lossless video coding methods including motion JPEG-LS, motion JPEG2000, animated GIF and our proposed method. The five testing sequences are available at [5-9] and they are originally compressed in animated GIF format. They were decompressed and decomposed into sequences of frames before being compressed with the evaluated coding methods. Fig. 2 shows the last frames of the five sequences for reference. The index planes were encoded when motion JPEG-LS and motion JPEG2000 were used.

When coding the sequences with the proposed method, two different schemes were separately evaluated. In the 1st scheme all frames were separately encoded with the approach presented in Sections 2 and 3. In the 2nd scheme, except the first frame which is encoded with the approach presented in Sections 2 and 3 directly, all other frames are

encoded as follows. For each frame, all pixels whose colors are identical to the previous frame are assigned a new color index which means transparent. The palette and the index map are then updated accordingly. After processing all frames, their updated color index maps are encoded with the 1st coding scheme as if they were typical frames. This scheme allows one to exploit the temporal correlation to remove the inter-frame redundancy as animated GIF does. For reference purpose, the 1st and the 2nd schemes are, respectively, referred to as Ours-noT and Ours-T.

Whatever scheme is used, the coding of a frame can be decomposed into two stages and the proposed pixel-wise palette reordering method is exploited in the first stage. Fig. 3 shows the contribution of this palette reordering method to the coding performance. Specifically, Fig. 3a shows the 34th frame of sequence *Shoe_Shine*, Fig. 3b shows a typical index map which is associated with a palette whose colors are sorted by luminance, and Fig. 3c is the processing result of Fig. 3b. For the sake of simplicity, no inter-frame correlation was used to obtain Fig. 3c. In other words, the palette reordering method was directly applied to the index plane shown in Fig. 3b to obtain the output shown in Fig. 3c. One can see that both the energy and the entropy of the output index plane is significantly reduced. In particular, the entropy values of the original and the output index planes of the frame (i.e. Figs. 3a and 3c) are, respectively, 5.358 bpp and 3.437 bpp.

Fig. 4 shows the entropy reduction in the index planes of different frames when Ours-noT and Ours-T are used to encode sequences *Shoe_Shine* and *Gymnast*. The stage 2 presented in Section 3 is skipped to obtain the result shown in Fig. 4 as the focus is on the performance of the proposed pixel-wise palette reordering method here. Scaled versions of the two testing sequences are provided in Fig. 5 and Fig. 6 as a reference for one to interpret its performance in different situations.

In general, Ours-T makes use of the inter-frame correlation to remove the redundancy and hence it can improve the compression performance when the video sequence is a synthetic sequence. Its superiority over Ours-noT can be verified by comparing the performance of the two schemes in Fig. 4a. However, for natural sequences in which halftoning is exploited during their production to improve their picture quality, Ours-T does not work effectively and sometimes may even lower the compression performance as shown in Fig. 4b.

Table 1 lists the compression performance achieved by various coding methods in terms of bits per pixel (bpp). Unlike Fig. 4 which just shows the coding performance of stage 1 (i.e. pixel-wise palette reordering), columns Ours-noT and Ours-T in Table 1 list the overall compression performance contributed by both stages 1 and 2. Besides, the bit overhead for coding all header information including the palette is already taken into account. On average, when Ours-noT (Ours-T) is used, Stage 2 can further reduce 16% (11%) of the bits required for coding the output of stage 1 directly with entropy coding. The proposed schemes are superior to the other evaluated methods.

As shown in Table 1, Ours-T performs better in coding synthetic sequences but poorer in coding halftoned natural sequences. A hybrid scheme which selects Ours-T or Ours-noT to handle a video sequence based on its nature as animated GIF does can be used. Column Ours-H in Table 1 shows the performance of such a hybrid scheme.

## 5. CONCLUSIONS

A lossless coding scheme for encoding color-quantized video sequences is proposed in this paper. This scheme handles the sequences frame by frame. Based on the nature of an input sequence, this scheme determines whether it should introduce a transparent color to a specific frame so as to remove the inter-frame redundancy. Accordingly, it adjusts the color index map and the palette if necessary.

The coding scheme then reorders the palette of the adjusted frame, or the original frame if no adjustment is required, adaptively with the method proposed in Section 2 to reshape the statistical properties of the index map. Effectively, it results in a new index map each element of which serves as an index to a pixel-dependently reordered version of the palette. This resultant index map contains very low zero-order entropy and energy.

The coding scheme finally decomposes the resultant index map into bit-planes and then encodes them with a context-based binary arithmetic coder. Simulation results reveal that the coding performance of the proposed coding scheme is better than other state-of-art lossless image compression methods when coding color-indexed images.

## 6. ACKNOWLEDGEMENT

This work was supported by Centre for Signal Processing, The Hong Kong Polytechnic University (Grants G-U286 and 1-BB9T). The author would also like to thank K. H. Chung for providing part of the simulation results.

## REFERENCES

[1] M.T. Orchard and C.A. Bouman, "Color quantization of images," IEEE Trans. On signal Processing, Dec 1991, Vol.39, No.12, pp.2677-2690.

[2] Y. Li and K. Sayood, "Lossless video sequence compression using adaptive prediction," IEEE Trans. On Image Processing, Vol.16, No.4, April 2007, pp.997-1007.

[3] A.J. Pinho and A. J. R. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images," IEEE Trans. Image Processing, Vol.13, No.11, Nov 2004, pp.1411-1418.

[4] ISO/IEC JTC 1/SC 29/WG 1 JPEG LS image coding system, ISO Working Document ISO/IEC JTC1/SC29/WG1 N399 – WD14495, July 1996.

[5] http://www.ifa.hawaii.edu/~joshw/Astrophotos/2006_01_05/NMSkies_Animation3.gif

[6] http://en.wikipedia.org/wiki/Image:BananaShoeShine.gif

[7] http://en.wikipedia.org/wiki/Image:Rotating_earth_%28large%29.gif
[8] http://www.nrmsc.usgs.gov/images/glacier_animation_slow.gif
[9] http://www.hyperfun.org/App/ASP/gymnast_anim.gif

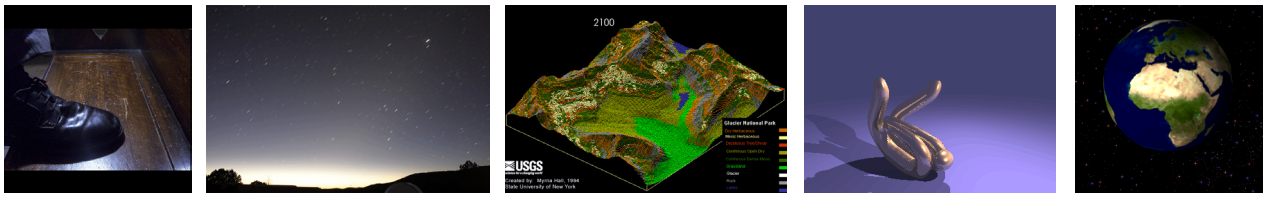
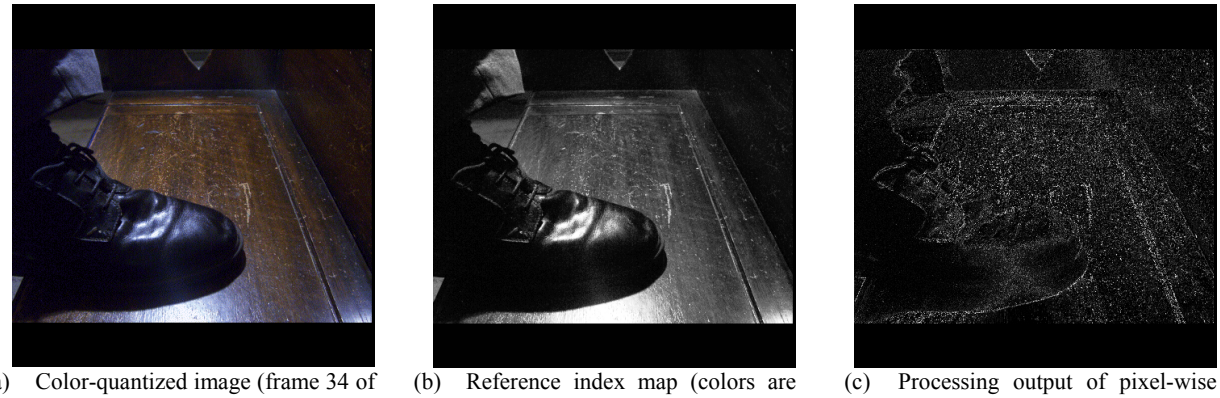
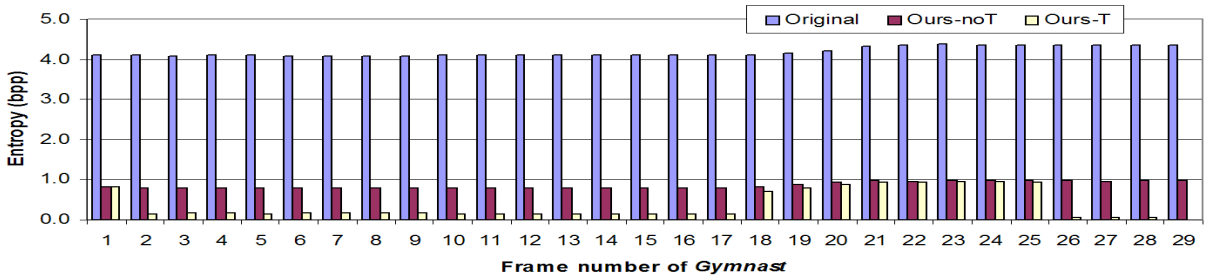



Fig. 2 Last frames of the five testing video sequences (scaled for display)

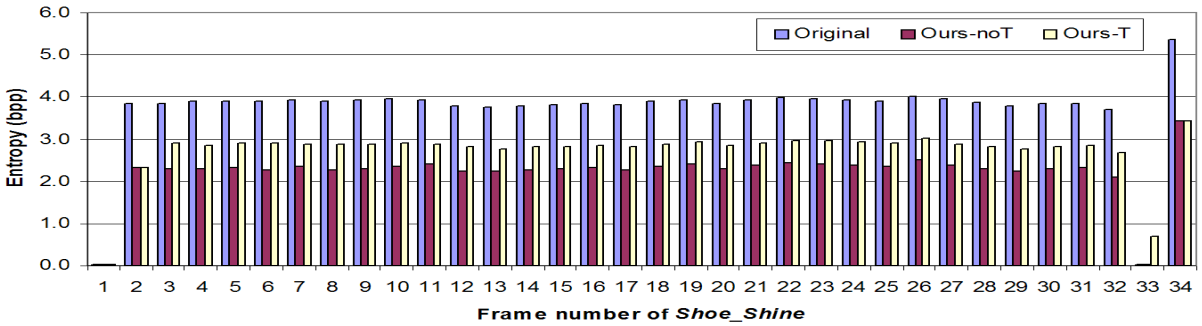


(a) Color-quantized image (frame 34 of *Shoe_Shine*) (b) Reference index map (colors are sorted by luminance) (c) Processing output of pixel-wise palette reordering

Fig. 3 Processing result of the proposed pixel-wise reordering method




(a) Synthetic sequence: *Gymnast*




(b) Halftoned natural sequence: *Shoe_Shine*

Fig. 4 Reduction in zero-order entropy of an index map when the proposed palette reordering method is applied

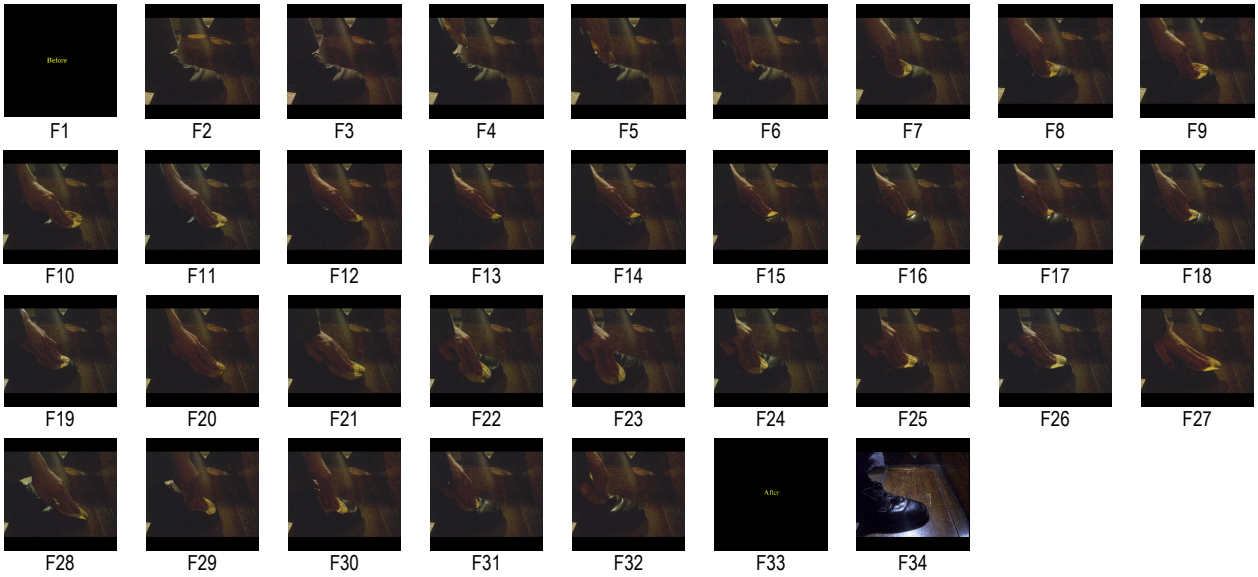



Fig. 5 Sequences *Shoe_Shine* (Scaled for display)


Fig. 6 Sequences *Gymnast* (Scaled for display. Frame 58-*i* = Frame *i* for $0<i<29$)

| Testing sequence | No. of frames | Frame size | Lossless coding methods for coding color-indexed video sequences | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Animated GIF | Motion JPEG-LS | Motion JPEG2000 | Ours-noT | Ours-T | Ours-H |
| Shoe_Shine | 34 | 450x450 | 2.8775 | 3.4051 | 3.7335 | 1.9771 | 2.4115 | 1.9771 |
| NWSkies | 11 | 800x531 | 3.4131 | 7.0081 | 7.7617 | 2.7525 | 3.4405 | 2.7525 |
| Glacier | 28 | 750x500 | 0.9645 | 1.7803 | 1.9632 | 0.8793 | 0.3519 | 0.3519 |
| Gymnast | 57 | 400x300 | 0.3654 | 1.0066 | 1.5829 | 0.6721 | 0.3429 | 0.3429 |
| Rotating_earth | 44 | 400x400 | 1.6242 | 1.9333 | 3.1451 | 0.9820 | 0.8571 | 0.8571 |
| **Average** | | | **1.8489** | **3.0267** | **3.6373** | **1.4526** | **1.4808** | **1.2563** |

Table 1. Performance of different lossless video compression methods for coding color-indexed video sequences in terms of bpp