Journal of Information Display
Vol. 12, No. 2, June 2011, 77–83

Taylor & Francis
Taylor & Francis Group

Flicker-reduced memory compression for a volume-zone liquid crystal display overdrive

Hisashi Sasaki[a*], Tatsuyuki Ishikawa[b], Yukio Ishikawa[b], Kazuhiro Ichikawa[b], and Nobuhiko Saitou[b]

[a] *Toshiba Corporation, Semiconductor Company, 580-1 Horikawa-cho, Saiwai-ku,/Kawasaki, Kanagawa, Japan*
[b] *Toshiba Microelectronics Corporation, 25-1 Ekimae-honcho, Kawasaki-ku/Kawasaki, Kanagawa, Japan*

(*Received 30 November 2010; Revised 14 January 2011; Accepted for publication 17 February 2011*)

The memory compression algorithm CCC-LCP (color-count-controlled local color palette) reduces flicker in the liquid crystal display (LCD) overdrive. Its compression ratio is 1:5.4 for 10-bit images, with a 33 dB PSNR peak signal-to-noise ratio and with five times flicker reduction compared with the block truncation coding. The authors' two-alternative forced choice subjective tests proposed two new soundness properties, the 'CMP harmlessness' and 'OD non-lost (or OD liveliness)', to clarify the functional interaction between the overdrive functionality OD and the compression functionality CMP. The tests verified that CCC-LCP is practically applicable (at a 1.2H viewing distance threshold) for 42" 37-ppi WXGA TVs.

Keywords: LCD overdrive; memory compression; flicker reduction; subjective evaluation; functional interaction

1. Introduction

Today, an overdrive is indispensable for liquid crystal display (LCD) TVs and LCD monitors to improve blurring of moving images. As its frame memory is too large for a timing controller, memory compression is the active technical area for overdrive practice. There are many image compression approaches such as BTC (block truncation coding) [1–4], Wavelet [5], DCT (discrete cosine transformation) [6], color quantization [7, 9] and DPCM (differential pulse code modulation) [8]. Each approach has drawbacks, though. Thus, in this study, a new image compression algorithm is proposed for a practical image-quality-first high-compression LCD overdrive.

In the first section of this paper, the proposed algorithm is described, and in the second section, the results of the performed simulation and subjective evaluation tests are presented. In the final section, the proposed algorithm is compared with other approaches.

2. The proposed compression method: local color palette

2.1. Algorithm

Before explaining in detail the proposed algorithm, its basic idea is described in Figure 1(a, b, c). Figure 1(a) shows that the proposed algorithm is processed blockwise, as, for instance, in a 4×4-pixel block. $N \times M$ is generally admissible, wherein N and M are positive integers. Practically, the authors' experience suggests that 4×4 is the best choice for the application in this study.

There are 16 pixels for a 4×4 block. The basic idea of the proposed algorithm is the color palette. Data compression is performed by selecting representative colors to approximate an original block. Figure 1(b) shows an example of the reduction of 16 colors to four colors. The four colors are called 'representative colors' (rep-colors), because they replace the original colors as the representatives of such colors. These four rep-colors constitute a color palette for a processing block. Thus, this approach is called 'local color palette', in contrast with the popular conventional color palette approach that has a single global color palette for a whole image.

Figure 1(c) shows a bitmap and rep-colors. The replacement of the original pixel color with the rep-color is implemented as a bitmap. For instance, the right upper pixel has the bitmap '1', which means that the rep-color '1' is used for the replacement. As there are $N \times M$ pixels for the $N \times M$ block , there are $N \times M$ bitmaps.

Figure 2 shows the three steps in the proposed algorithm, CCC-LCP (color-count-controlled local color palette). The first step is splitting, which classifies $4 \times 4 = 16$ pixels into four groups of partition clusters. The second step is the calculation of the rep-colors for each group. The third step is the quantization of the color values controlled by the color count. This color count is given by the second step, the calculation of the rep-colors. The following flags are set: 1 (true) for the existence of quantized data, and 0 (false) for the non-existence of quantized data. These flags are concatenated sequentially as color configurations.

*Corresponding author. Email: hisashi3.sasaki@toshiba.co.jp

ISSN 1598-0316 print/ISSN 2158-1606 online
© 2011 The Korean Information Display Society
DOI: 10.1080/15980316.2011.567821
http://www.informaworld.com

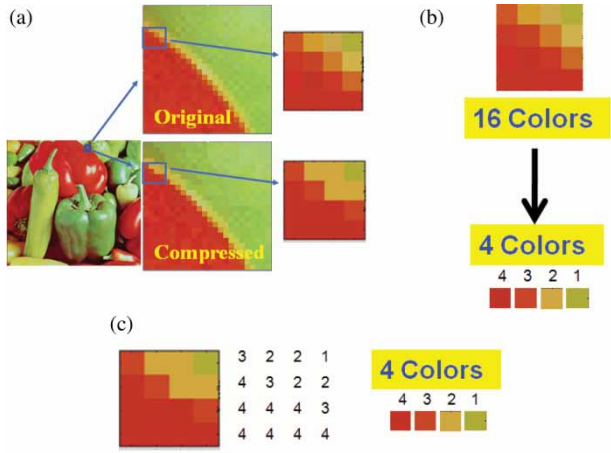

Figure 1. (a) Basic idea of block processing. (b) Basic idea of color data reduction in the local color palette. (c) Basic idea of the bitmap and rep-color for a local color palette.

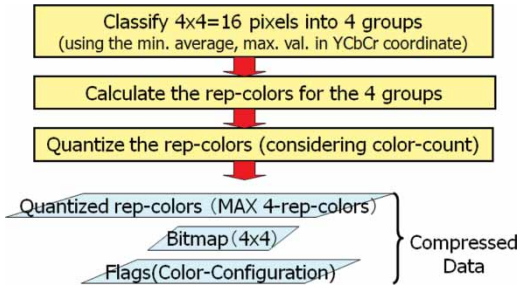

Figure 2. Encoding flowchart.

Next, more details of the steps of the proposed algorithm shall be described. The first step is splitting in the YCbCr color space, which consists of the following sub-steps: (1-1), (1-2), and (1-3). RGB may also be applicable as a color space. YCbCr generally yields better results, though.

(1-1) Calculate MIN (minimum), MAX (maximum), and AVE (average) for all the pixels in a block.

(1-2) Create four groups of partition clusters, as follows. Figure 3 shows that there are nine partition patterns. For instance, the left upper pattern means that there are two cutting planes that divide two regions (the lower region and the upper region in Y) into four final regions. Let y denote the Y coordinate for a given color pixel. Let MIN_Y, MAX_Y, and AVE_Y denote the MIN, MAX, and AVE in the Y coordinate, respectively. Then, the lower region is given by MIN_Y <= y < AVE_Y, and the upper region is given by AVE_Y < y <= MAX_Y. Furthermore, the second division is performed using each regional minimum, maximum, and average. Let MIN_lower_Y, MAX_lower_Y, and AVE_lower_Y be the minimum, maximum, and average for the lower region, respectively. Let MIN_upper_Y, MAX_upper_Y, and AVE_upper_Y be the minimum, maximum, and average for the upper region, respectively. The lower region is divided again by MIN_lower_Y, MAX_lower_Y, and AVE_lower_Y. The upper region is divided again by MIN_upper_Y, MAX_upper_Y, and AVE_upper_Y. The same notations are used for the Cb and Cr coordinates: for instance, MIN_Cb and MIN_lower_Cb for the Cb coordinate.

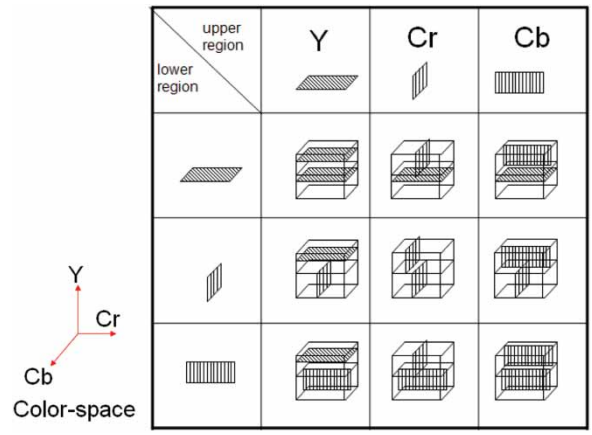

Figure 3. Partition pattern matrix.

(1-3) Select a partition pattern from the partition pattern matrix, as follows. Let the widths be defined as WIDTH_ <COORD>= MAX_ <COORD> −MIN_ <COORD> and WIDTH_ < REGION > _ < COORD >= MAX_ <REGION> _ <COORD> −MIN_ <REGION> _ <COORD>, wherein < REGION >= lower| upper and <COORD>= Y|Cb|Cr. It is experimentally known that the Y coordinate statistically yields good results for the first division, so the cutting-place direction was fixed at Y; that is, AVE_Y always determines the upper and lower regions. This fix is an option that helps simplify the algorithm. The second division is performed as follows. For instance, the lower region will be explained more because the same procedure is applied to the upper region. The max MAX_WIDTH is calculated from the three widths: WIDTH_lower_Y, WIDTH_lower_Cb, and WIDTH_lower_Cr. Then, the second-divide direction that was determined from the Y coordinate for MAX_WIDTH is WIDTH_lower_Y and for the Cb coordinate for MAX_WIDTH, WIDTH_lower_Cb, and from the Cr coordinate for MAX_WIDTH, WIDTH_lower_Cr. Thus, there are nine patterns, because there are three patterns for each of the lower and upper regions, respectively. Thus, the color space,

which classified 4x4 pixels into four groups of partition clusters, was split.

The second step is the calculation of the rep-colors for each group. This step consists of sub-steps (2-1) and (2-2).

(2-1) Calculate a rep-color or an average color of pixels for a group. Note that there is no pixel in the group for some cases. In these cases, there is a possibility that there are no data in the group, such as in the sample case, in which, the pixels are located just on the cutting plane: the pixels are forced to belong to either single region, so that one of the regions (= group) has no pixel. Such existence information is gathered as a flag. The flag is set as 1 (true) for existence and as 0 (false) for non-existence for a group.

(2-2) Count the number of calculated rep-colors. This count is called the 'rep-color count'. In fact, the count is calculated from the flags by adding their values as integers. For instance, $1 + 1 + 0 + 1 = 3$ is calculated when there are three existence flags (shown as the integers 1 in the equation) and a single non-existence flag (shown as 0). These flags constitute the 'color configuration' as the catenation of flags 1101.

The third step is the quantization of the rep-colors. It consists of the following three sub-steps: (3-1), (3-2), and (3-3).

(3-1) To attain a smaller data size, the rep-colors are adaptively quantized using the rep-color count. When the rep-color count is 1, 10 bit-depth quantization is adopted for a single rep-color in YCbCr. This is denoted as '10,10,10', wherein 10, 10, and 10 are sequentially catenated in the YCbCr order. When the rep-color count is 2, '8,8,8' is adopted. When the rep-color count is 3, '6,5,5' is adopted.

(3-2) When the rep-color count is 4, a differential mode flag, DIF-MODE: DIF-MODE == '1', must be adopted, which refers to the differential mode, and DIF-MODE == '0' refers to the normal node. When WIDTH_ < COORD > is larger than a specified threshold value, DIF-MODE must be set at '0'; otherwise, it must be set at '1'. The threshold values are experimentally determined in practice.

(3-3) For the normal mode, its bit depth is directly specified as '5,4,4', as in the previous step. For the differential mode, a 6-bit-depth base value and its difference value for each of the color components Y, Cb, and Cr were prepared. To represent four difference values, 3 bits were used to control the bit depth and 2 bit depths for the value itself. Thus, $6 + 2 \times 4 + 3 = 17$ bits are needed for a single color component.

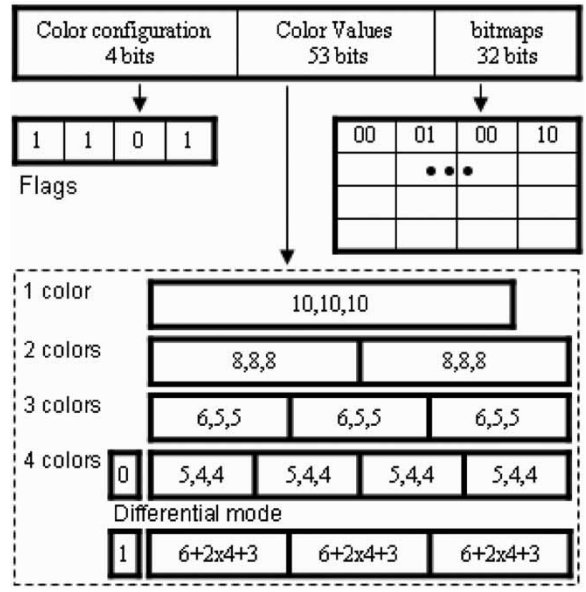

Figure 4. Compressed data.

The quantization was determined based on the rep-color count to keep the data size as comparable as possible to the maximum size. When the rep-color count is 1, its size is $10 + 10 + 10 = 30$ bits. When the rep-color count is 2, its size is $(8 + 8 + 8) \times 2 = 48$ bits. When the rep-color count is 3, its size is $(6 + 5 + 5) \times 3 = 48$ bits. When the rep-color count is 4 and the DIF-MODE='0', its size is $1 + (5 + 4 + 4) \times 3 = 53$ bits. When the rep-color count is 4 and the DIF-MODE='1', its size is $1 + (6 + 2 \times 4 + 3) \times 3 = 52$ bits. These sizes are bounded by the maximum size 53 bits, so that the fixed 53 bits are prepared in advance for these cases (see also 'color values' in Figure 4).

Figure 4 shows the authors' compressed data. The data's bit sequence has three parts: the color configuration, color values, and bitmaps. Note that the bitmaps differ from BTC. In this case, bitmaps identify a rep-color, whereas in BTC, bitmaps independently identify a rep value (representative value) for each color component. That is, the authors' color values constitute a color palette, which is adjectivally modified as 'local' by considering that a single global color palette is commonly used for a whole image in many conventional compressions. For this reason, the authors' approach was called the 'color-count-controlled local color palette'. This difference in the handling of the bitmap is the authors' algorithm key point, which reduces the compression flicker that is mentioned in the later section.

Decoding is fundamentally identical to BTC, except for the color-count control. Flag values are numerically added to count the rep-color in the stored bit sequence. Then, the color count identifies the color configuration. The color

Images→ ↓ Alg.	bicycle N5 A	cafe N2 A	orchid N6 A	fruits N3 A	portrait N1 A	average
CCC-LCP	33.0	28.1	37.8	33.5	32.5	33.0
hcFFD2	32.40	21.29	36.16	27.88	33.35	30.22

Figure 5. PSNR.

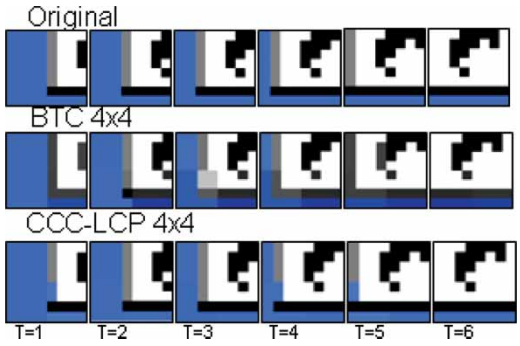

Figure 6. Compression flicker.

values are sequentially extracted from the color configuration and de-quantized. Finally, the image is reconstructed. The differential mode treats flat images to improve the color bits by making them deeper.

Original images have $(10 + 10 + 10) \times 4 \times 4 = 480$ bits. The authors' compressed data have $4 + 53 + 32 = 89$ bits. Thus, the compression ratio is $89 : 480 = 1 : 5.4$.

The authors' FPGA implementation requires 25 KG. This is small compared with other types of signal processing in the TCON (timing controller) of LCD TVs. The processing time of this algorithm is shorter than the frame time that is required by the overdrive itself. This means that the authors' algorithm is small enough.

2.2. *PSNR Analysis*

Figure 5 shows a PSNR comparison between the authors' simulation results and those of FFD2 [1]. The authors' PSNR is 3 dB higher than that of FFD2.

2.3. *MAXDIF Analysis*

Figure 6 shows a zoomed comparison of the original, BTC, and CCC-LCP images generated by the right-to-left movement of a still image, at 1 pixel per frame. It can be easily seen that BTC has a more serious fluctuation than CCC-LCP. The lower left corner of the 'shortcut mark of the icon' is critical for a flicker, especially for $T = 3$ and 4. The lower part directly below the icon is also critical: it is much darker than the original, and its area is widely connected. This flicker essentially originates from the movement of the processing block boundary as a side-effect of compression. Thus, such flicker is called the 'compression flicker'.

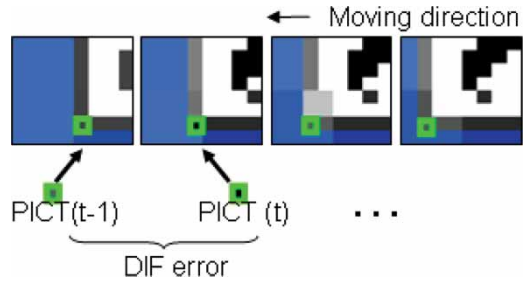

Figure 7. Definition of DIF.

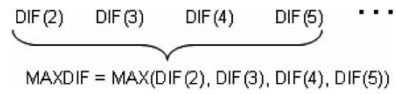

Figure 8. Definition of MAXDIF.

To formulate such compression flicker quantitatively, two quantities, DIF and MAXDIF, and its histogram are used. As a flicker is a color fluctuation in a moving pixel, the color difference plays an important role. Let PICT $(t - 1)$ and PICT (t) denote two identical compressed pixels for two successive frames. The quantity 'DIF error' is defined as a triplet of component-wise (RGB) absolute differences for the two pixel instances (Figure 7).

MAXDIF is defined as a triplet of maximum values for a given DIF sequence. Figure 8 illustrates the 4-length DIF sequence. Practically, processing a block size would produce an equal sequence length, because block-size translation exactly reproduces the encoding situation.

Figure 9 shows two MAXDIF error images for BTC and CCC-LCP. It can be easily seen that BTC has more serious flicker than CCC-LCP, and that its icon edges are critical. The MAXDIF value is truncated to 255 as the maximum. Its moving direction is the upper left, and its original image is shown in Figure 12.

Finally, Figure 10 shows a reduction of the compression flicker via a histogram of MAXDIF. To check how many pixels contribute to a flicker, the threshold value should be introduced. That is, this threshold gives the cumulative count, the pixel of which has a MAXDIF value above the threshold. For instance, let the threshold be 32. Then, the cumulative counts are as follows: 4208, 3373 and 5324 for BTC images and 616, 545 and 1359 for CCC-LCP for RGB. These counts estimate 5.6 times' improvement ($4208/616 = 6.8$, $3373/545 = 6.2$, and $5324/1359 = 3.9$).

3. Subjective Evaluation

3.1. *Experimental Setup for Subjective Evaluation*

Subjective tests are prerequisites to the practical application of TV. The 2AFC (two-alternative forced choice) test was used. The authors' two alternatives were forced to choose a response to two synchronously moving images displayed

Figure 9. MAXDIF error images.

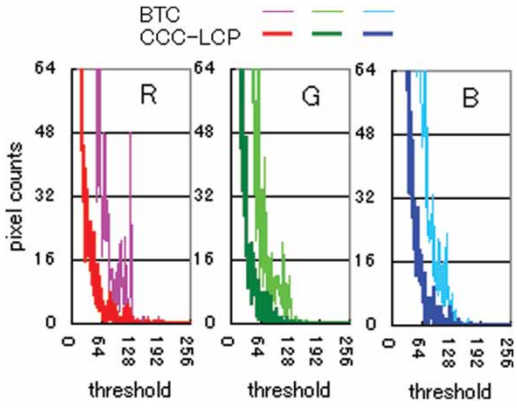

Figure 10. MAXDIF error histogram.

side by side on a TV set. As the two displayed images were randomly instantiated from the two seed images that were to be tested, there were four possible cases: two cases of the identical images and two cases of the different images. The examinee was asked to check exclusively whether or not the two displayed images were identical. There was a solo examinee, Sasaki, who had naked visual acuity scores of 20/20 (right) and 20/25 (left). If his correct response probability was high, it is known that the he really detected the difference between the two seed images. Figure 11 shows the authors' experimental setting.

Figure 12 shows the test image 'PC desktop' as an artificial image, and Figure 13 shows the test image 'Church' as a natural image. Many check points are indicated in these images. In Figure 12, icons may sometimes cause flickering and letters may sometimes cause

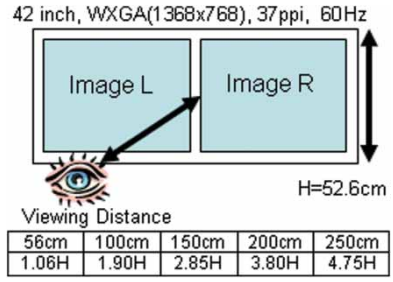

Figure 11. The authors' experimental setting for 2AFC.

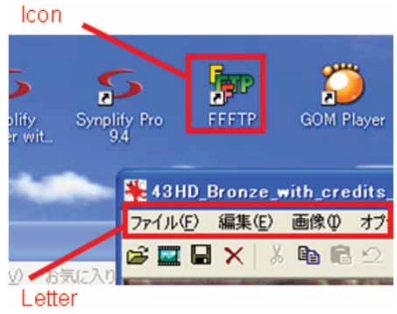

Figure 12. Artificial test image 'PC desktop'.

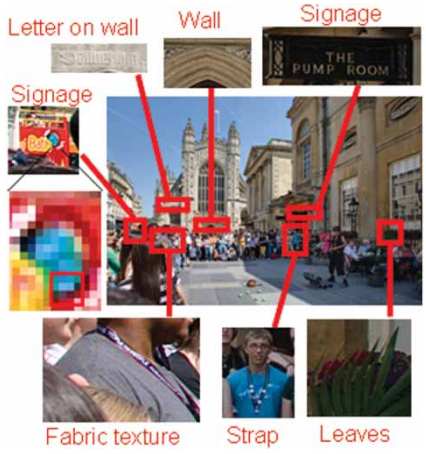

Figure 13. Natural test image 'Church'.

blurring. In Figure 13, check points may cause flickering or color-value-conversion error.

3.2. *Soundness of the Subjective Evaluation of the LCD Overdrive*

Before explaining the subjective tests, the labeling convention that specifies the test conditions will be defined. The label 'OD' means the overdrive functionality and the label 'CMP' means the compression functionality. When the OD label has the suffix N (No), the overdrive is set to inactive (OFF). When its suffix is Y (Yes), the overdrive is set to active (ON). When its suffix is ONOFF, two settings are toggled to test the overdrive functionality. The same convention is used for

the CMP label. Thus, the authors' test is a concatenation of the two labeling notations OD_ < OD_value > _CMP_ < CMP_value >, wherein < OD_value >::= N|Y|ONOFF and < CMP_value >::= N|Y|ONOFF.

Next, the 'level of functional interaction' for the two functions of overdrive and compression will be discussed. 'Primary level' means that the function OD or CMP is independently tested without any interaction. 'Interactive level' means that the functional interaction is tested. The interactive test is the main concern in this study, and the primary test is merely informative.

The interactive tests check two soundness properties: the CMP harmless (to OD) property and the OD non-lost (by CMP) property. The CMP harmless test is labeled 'OD_Y_CMP_ONOFF'. The OD non-lost test is labeled 'OD_ONOFF_CMP_Y'.

CMP harmless test: The authors expected non-detection of the compressed image against its original under the overdrive condition. Therefore, the two displayed images should be an overdriven image with a compressed memory and an overdriven image with an original memory. In other words, the compression functionality does not harm the overdrive functionality. Thus, this test will be called 'CMP harmless to OD'.

OD non-lost test: In addition to the CMP harmless test, the second test, the OD non-lost test, was performed to check if the overdrive functionality would not be lost under memory compression. The authors expected to detect an overdriven image against a non-overdriven image. Therefore, the two displayed images should be an overdriven image with a compressed memory and a non-overdriven image with a compressed memory. In other words, as it may be said that the overdrive functionality is alive, it may be called the 'OD liveliness property'.

The third additional test, OD_N_CMP_ONOFF, is somewhat controversial (obviously, excessive for the overdrive itself). It directly checks the difference between the original and compressed images. This test provides the information needed to recognize where and how much the compressed image has deteriorated. These deteriorated pixels are check candidates for other tests.

The last test, OD_ONOFF_CMP_N, is a common native test, which checks the overdrive functionality itself.

3.3. Test Results of the Subjective Evaluation

The CMP harmless tests (notated as 'OD_Y_CMP_ONOFF') must first be discussed. Figure 14 shows the 1.2H detection distance for the artificial image, which is smaller than half of the 3H distance. That is, 1.2H is short enough for the HDTV criterion. Figure 15 shows no detection of the natural image.

Next, the OD non-lost tests (notated as 'OD_ONOFF_CMP_Y') are discussed. Both Figures 14 and 15 show the soundness of the overdrive, that is, its functionality is not lost due to compression.

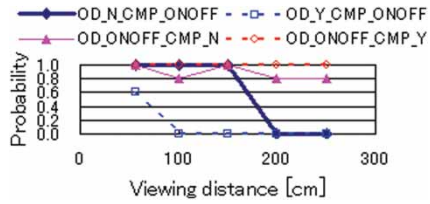

Figure 14. Detection probability for the 'PC desktop'.

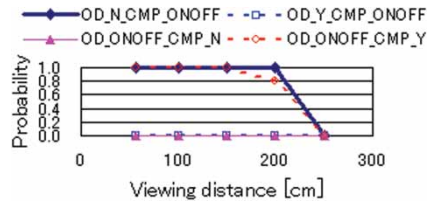

Figure 15. Detection probability for the 'Church'.

4. Discussions

4.1. Discussions on compression

4.1.1. *Comparisons with other competitive approaches*

Variable-length approaches such as DCT [6] and Wavelet [5] have the fatal problem of unbalanced error (Figure 16). This is because of their quantization, in that the first prediction phase adopts a pessimistic estimate of the code length, which causes the visibly coarse quantization step.

Block compression generally has a compression flicker. Many BTC derivatives [1–4] aim at a higher compression ratio. Higher quality is required for a practical overdrive, however, rather than a higher compression ratio. That is, the authors' trial used the image-quality-first approach.

Texture compression (TC; e.g., S3TC) is famous as high-quality compression for texture mapping. (Figure 17 shows that CCC-LCP is comparable with others in the PSNR.) Generally, however, TC is not applicable to real-time encoding. Only the GPU-based approach [9] provides real-time encoding, but it requires a GPU, a very large hardware for peripheral devices.

DPCM [8] has good PSNR, and no compression flickering. Its compression ratio is poor, though.

4.1.2. *Basis for the work*

This approach is inspired by two works: LCQ (local color quantization) [10] and BDA (bit-depth analysis) [11]. LCQ has K-means optimization, which means LCQ is impractical

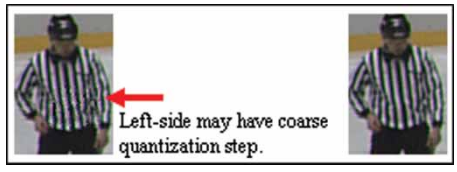

Figure 16. Unbalanced error problem.

Images→ ↓ Alg.	lena	mandril	lorikeet	kodim 1	kodim 2	kodim 3	kodim 4	kodim 5	average
CCC-LCP	35.8	28.0	33.8	33.0	37.2	37.9	36.8	32.2	34.3
GPU	33.6	27.3	32.6	31.7	34.3	35.5	34.9	30.2	32.5
ATI	36.0	28.7	34.4	34.8	36.9	38.5	38.0	32.8	35.0
NVIDIA	35.2	27.8	33.1	34.6	36.5	38.0	37.6	32.5	34.4

Figure 17. PSNR comparison with TC [9].

for real-time encoding. Thus, the quantization step was re-created for real-time encoding, its rep-color counts were increased, and the color-count control that BDA suggested was added to enhance the image quality.

4.1.3. *Target products*

The dominant screen sizes in production are 30–40". The authors' compression is suitable for these volume-zone products. Full high definition has a comparative pixel resolution of 37 ppi for the larger screen size of 60". This is also the authors' target.

4.2. Discussions on the Subjective Evaluation

MPRT (moving picture response time) measurement is a popular method of getting the optimal OD-LUT (overdrive lookup table). This measurement is also used as a test that is objectively conducted under the very critical worst conditions to manifest blurring caused by boundary change in a large moving box. It is good for use as an exact instrumental test, but it is far from the everyday living environment, in which, the running texture (ticker) is critical to blurring. The authors' subjective test supports such everyday life tests. Furthermore, as their test has the viewing distance as a parameter, its results yield an informative evaluation. Mostly in everyday life, the 3H distance is enough as the average viewing distance, and therefore, the authors' compression is all right for everyday life. When one comes closer to a TV set, he or she can detect more details. Such closer watching is bad for the health, however, and is too maniacal and has excessive specifications from an engineer's point of view, which is different from ordinary people's viewpoint.

The authors' soundness test is a new approach, because it divides the whole-functionality check into more detailed functionality checks, that is, the CMP harmless and OD non-lost tests, to clarify the functional interaction. The conventional study did not provide a clear conceptual distinction.

5. Conclusion

Two new technologies are proposed in this paper: CCC-LCP compression and the authors' 2AFC soundness subjective tests, and the CMP harmless and OD non-lost properties. Based on these, it was verified that CCC-LCP reduces compression flicker in a high-quality, high-compression LCD overdrive. Its compression ratio is 1:5.4 for 10 bits, its PSNR is 33 dB, and its compression flicker is reduced 5 times compared with that of BTC. The authors' 2AFC tests showed that the threshold distance was 1.2 H for the 42" (37-ppi) WXGA, which is good enough for the everyday living environment.

References

[1] J. Someya, A. Nahase, N. Okuda, K. Nakanishi and H. Sugiura, *SID Tech. Dig.* 464 (2008).
[2] J. Wang, K. Min and J. Chong, *ITC-CSCC Tech. Dig.* 1629 (2008).
[3] J.-W. Han, M.-C. Hwang, S.-G. Kim, T.-H. You, and S.-J. Ko, *IEEE Trans. Consum. Electron.* **54-4**, 1839 (2008).
[4] H. Pan, X. Feng, and S. Daly, in *IDW Tech. Dig.* 1981 (2006).
[5] S.K. Lee, D. Yoo, S. Kang, and Y.H. Kim, in *SID Tech. Dig.* 468 (2008).
[6] R.H.M. Wubben, G.J. Hekstra, and H.A.W. Schmeitz, in *SID Tech. Dig.* 1348 (2004).
[7] S.-J. Koo, C.-G. Kim, J.-K. An, M.-H. Park, and S.-D. Yeo, in *SID Tech. Dig.* 474 (2005).
[8] S.-J. Lee, S.-H. Lee, and D.-H. Kim, US Patent Application Publication US2008/0131087A1, (2008).
[9] O. Alexanderson and C. Gurell, *Compressing Dynamically Generated Textures on the GPU* PhD thesis, Lund University, 2006.
[10] G. Qiu, in *CIC Tech. Dig.* 205 (1998).
[11] H. Sasaki, in *IDW Tech. Dig.* 1409 (2008).