A Simple Compression Method Using Movion Vector of Video Encoder in the Distributed System

Yasuyuki Miura¹, Sho Nakane¹ and Shigeyoshi Watanabe¹
¹Shonan Institute of Technology, 1–1–25, Tsujido-nishi-kaigan, Fujisawa, Kanagawa 251–8511, Japan

Abstract—With the technological advancements of PCs, it has been possible to edit and encode video using software. Parallel processing has been possible with low-price PCs. We have developed a parallel encoding system for a cluster or Grid with multi-core PCs. Because the transmission between PCs may become bottlenecked if an uncompressed video image is the first sent, we have proposed a simple compression that can be transmitted via a LAN. In this paper, we presented a simple compression method and we proposed the improved method by using motion vector from video encoder. Also, the results of our performance evaluation were presented. As a result from some experimentation, maximum and average of the data amount were reduced by using motion vector.

Keywords: MPEG, Distributed Processing, Video Encoding

1. Introduction

With the technological advancements of PCs, it has been possible to edit and encode video using software. Parallel processing has been possible with low-price PCs. However, the software[1][2] encoding of video images still places a serious burden on PCs. It takes a long time to encode high-ratio, high-quality, and high-resolution movies. This is the burden of contents producers.

High-performance encoders with a DSP or a multi-core CPU have been developed[3]. However, it is very expensive to introduce them. Although low-cost hardware encoders are provided, the quality of encoded movies using them is not very high because of software and hardware limitations.

We have been developed a parallel encoding system for a cluster or Grid environment with multi-core PCs. In our system, the Grid is constructed of multiple PCs in a LAN setup and the video images are encoded in the system. The encoding process is as follows:

(1) Send the uncompressed image to the Encoder PC.
(2) Start the encoding process.
(3) Reply a bitstream to the producer’s terminal and confirm the encoded image.

For complex encoding that includes high-resolution P-frames and B-frames, high specification hardware is necessary. In this system, by separating the encoding process from the user’s terminal, real-time encoding can be achieved and the user can confirm the encoded movie during the encoding process.

2. Parallel Encoding System of Video Image

We are developing a parallel encoding system for video images. An outline of this system is shown in Fig. 1. In our system, the Grid environment is constructed of multiple PCs in a LAN setup and the video images are encoded in the system. The encoding process is as follows:

(1) Send the uncompressed image to the Encoder PC.
(2) Start the encoding process.
(3) Reply a bitstream to the producer’s terminal and confirm the encoded image.

For complex encoding that includes high-resolution P-frames and B-frames, high specification hardware is necessary. In this system, by separating the encoding process from the user’s terminal, real-time encoding can be achieved and the user can confirm the encoded movie during the encoding process.

3. Simple Encoding Method

3.1 Requirements

An uncompressed image must be sent in step (1) in the process shown in Fig. 1. When the uncompressed image is directly sent, it places a serious burden on the LAN. Therefore, we decided that a simple compression method that does not place any burden on PC is needed. To achieve this, the following requirements are necessary:

Frame Rate and Resolution

Since an old PC can be used as the terminal PC, we assume that it has lower specifications. In addition, the CPU load for the simple compression should be as low as possible, because the decoding process is simultaneously driven in the terminal...
PC. The standard home-use digital video camera has a $720 \times 480$ pixels resolution, and the resolution of a DVD movie is $640 \times 480$ pixels. So, the required resolution of this system is assumed to be more than $640 \times 480$ pixels, and the required frame rate is assumed to be more than 30 frames/s in this research.

Compression Ratio
In this system, a high ratio compression, such as that for an MPEG, is not necessary because a transmission via a 100Mbp-1Gbps LAN is assumed. However, since a bit-rate of $640 \times 480$ pixels and a 30 frames/s uncompressed video image requires a 220-Mbps bandwidth, the network capacity of 100BASE-T is not sufficient enough to transmit the uncompressed image. The transmission of an uncompressed image weighs on the communications of other traffic. So, a fraction of the compression is necessary. In addition, it is desirable to be able to change the compression ratio according to the network performance.

Degradation of Image Quality
In this system, the video image is decoded and encoded after transmission to the encoder PC. In general, repetition of the encoding and decoding causes image quality degradation. So, a lossless compression is desirable in the compression process of this system. If necessary, a slight degradation in quality is permitted. However, a process with a serious amount of quality degradation must be avoided.

4. Traditional Method of Video Compression

The most major standards of video coding are MPEG-1[5], MPEG-2[6], MPEG-4[7], and H.264-AVC[8]. These methods need the large amount of processing time for DCT or Motion Prediction. JPEG[9] is the most famous coding standard for still image. JPEG is also using DCT as MPEG. So, JPEG is not suitable for this method as MPEG.

PNG[10] is lossless encoding method for still image. This method spends the large amount of processing for LZ77, and a high CPU load is needed for efficient compression using filtering. For example, coding time of a frame for PNG is shown in Fig. 2. Fig. 2 is the result of encoding of $640 \times 480$ pixels image using LibPNG. Coding time is different with compression levels. The highest compression level 9 takes the time for 1-2 seconds, and the minimum compression level 1 takes the time about 100 ms. In PNG, since many coding time is needed for LZ77 and filtering, it is unsuitable for high-speed compression.

5. Simple Compression

5.1 The Data Flow of Simple Compression

In this method, based on MPEG-2[6] encoding, some blocks are reduced. This reduction is necessary because motion estimation and DCT are a heavy load on a CPU. On the other hand, to use the similarity of neighboring frames, all the frames are divided into I-frames without inter-frame prediction and P-frames with inter-frame prediction.

The frame sequence of the proposed method is shown in Fig. 3. As shown in Fig. 3, some P-frames are arranged after an I-frame. Each P-frame refers to the previous I-frame. The "Threshold Processing" is also processed if necessary.

The data flow of each frame is shown in Fig. 4. Our method is divided into "RGB→YUV transform", "Simple Difference", "RLE", and "Huffman Encoding". All of them are processed in P-frames. Only the "RGB→YUV transform", "RLE", and "Huffman Encoding" are processed in I-frames. The "Threshold Processing" is also processed if necessary.
5.2 RGB→YUV transform

An uncompressed video image of RGB format is transformed to YUV format. The YUV format is used for many standard video encoding. Since an RGB→YUV transformation places only a light load on the CPU and the number of data can be reduced by reducing the U element and the V element, the image data is transformed to the YUV format in the first step of our method.

5.3 Simple Difference

Inter-frame prediction is used in our method to use the similarity of neighboring frames. We use the simple difference between the P-frame and the previous I-frame because the detection of a motion vector puts a large burden on the CPU load.

"Simple Difference" is driven in the P-frames only. Let \( I_{py}(x,y), I_{pu}(x,y), I_{pv}(x,y) \) be the \( y, u, v \) element pixel values of the P-frame at \((x,y)\), and let \( I_{ry}(x,y), I_{ru}(x,y), I_{rv}(x,y) \) be the \( y, u, v \) element pixel values of the previous I-frame at \((x,y)\). The pixel values after "Simple Difference" \( I_{dy}(x,y), I_{du}(x,y), I_{dv}(x,y) \) are as follows:

\[
\begin{align*}
I_{dy}(x,y) &= I_{py}(x,y) - I_{ry}(x,y) \\
I_{du}(x,y) &= I_{pu}(x,y) - I_{ru}(x,y) \\
I_{dv}(x,y) &= I_{pv}(x,y) - I_{rv}(x,y)
\end{align*}
\]

5.4 RLE

Run length encoding (RLE)[11] is a kind of lossless encoding. Some methods are proposed to deal with the problem of the number of encoded data greatly increasing when a normal RLE is executed for non-consecutive values. RLE, SRLE, Pack Bits, and RLE-\(m\) are some of the more well-known improvement methods. Since our past research has proved that Pack Bits is suitable for our method, Pack Bits is used in our method. In our method, the pixel values after "Simple Difference" \( I_{dy}(x,y), I_{du}(x,y), \) and \( I_{dv}(x,y) \) are scanned horizontally, and RLE byte streams \( RLE_{y}, RLE_{u}, \) and \( RLE_{v} \) are generated.

5.5 Huffman Encoding

The Huffman encoding is a kind of the entropy encoding. In the Huffman encoding, a Huffman tree is constructed from the data line, and a bit string is allocated to the data based on the Huffman tree. Huffman encoding is inferior in compression ratio in comparison with arithmetic coding[14], but is suitable for real-time processing because there is less computational complexity.

There are two types of encoding for Huffman encoding[13], static Huffman encoding[12] and dynamic Huffman encoding. The former is the method for assigning a bit string after scanning and generating the Huffman tree. The latter is the method for assigning bits while generating the Huffman tree. The latter can generate a bit stream without finishing a scan of the whole data. However, because the CPU load is large, static Huffman encoding is used in our method. In our method, static Huffman encoding is processed to \( RLE_{y}, RLE_{u}, \) and \( RLE_{v} \), and bit streams \( b_{y}, b_{u}, \) and \( b_{v} \) are generated.

5.6 Improvement of compress ratio by using threshold processing

Since there are cases where the method described in the previous section is insufficient for compressibility depending on the communication capacity of the LAN, it is desirable for it to be able to change the compression ratio according to the performance of the network. Some noise is included in an image. The compressibility can be improved by removing the noise, because it causes degradation in the encoding efficiency. The simplest (it does not add to the CPU load) technique is used to improve the compressibility in our method. Therefore, we use the threshold processing for P-frame method.

In our method, the pixel values after "Threshold Processing" \( I_{ty}(x,y), I_{tu}(x,y), \) and \( I_{tv}(x,y) \) are generated from \( I_{dy}(x,y), I_{du}(x,y), \) and \( I_{dv}(x,y) \) respectively by threshold processing.

5.7 Compression Ratio and Processing Time

The amount of encoded data in the I-frame and the P-frame of a YUV (4,4,4) format is shown in Fig.5. Two bars of PNG of Fig.5 is the minimum and maximum compression level respectively. Five movies of this experiment have the following feature.
From the above results, the following points are shown.

1) Since the amount of uncompressed data is 900 Kbytes, the compression ratio of the proposed method becomes 20% - 40% with the P-frame, and 35% - 40% with the I-frame. These values are almost the same as the PNG format. However, in the high-speed LAN environment such as Gigabit Ether, this ratio of compression seems not to be a problem. Moreover, since the parallelization using multi-cores of information of the frame is reduced by simple difference, it seems to be possible to add a more complicated encoding method by the further improvement in the speed.

2) The Movie1, 3, and 5 had the high compression ratio of the P-frame compared with the I-frame. On the other hand, the difference to the compression ratio of the I-frame and the P-frame with the movie2 and movie4 was not seen. In video with few motions, this seems to be because the amount of information of the frame is reduced by simple difference.

Table 1: Compression Time of Proposed Method

<table>
<thead>
<tr>
<th>YUV + Simple Difference</th>
<th>RLE Huffman Encoding</th>
<th>Others</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>174.8ms</td>
<td>25.9m</td>
<td>26.9ms</td>
<td>8.2ms</td>
</tr>
</tbody>
</table>

6. The Use of Motion Vector

In the system mentioned in section II, since the motion vector is generated when the encoding process is done at Encoder PC, it is sent to and used in the terminal. Concretely, difference processing which used the motion vector from Encoder PC is done instead of "Simple Difference". However, in order to perform difference processing of next frame based on the returned motion vector, it is necessary to predict the motion vector for next frame.

For example, it is assumed that frames of the video image are arranged as one I-frame, n - 1 B-frames, one P-frame, n - 1 B-frames, one P-frame... as shown in Fig.6. In such case, encoding of the first I frame (frame-0) is done at first. Secondly, encoding of the next P frame (frame-n) is done. In this time, the motion vector between the frame-0 and the frame-n is generated. By using this motion vector, the motion vectors of succession frames are predicted and difference processing in the simple compression of them is done. In this case, the B frame from the frame 1 to the frame-n - 1 is sent firstly. Next, frames are sent in order the P frame of frame-2n, the B frames from frame-n + 1 to frame-2n - 1... Then, the motion vectors of these frames are predicted.

As a method for predicting the motion vector of frame-k based on the motion vector between the frame-0 and frame-n, the following two predictor can be considered.

The example of our method is shown in Fig.7. At first, When Frame-n (The second encoding frame of MPEG) is encoded, the motion vector of Frame-n is generated. Then the motion vector of Frame-n is sent to terminal, and it is used in the Simple Compression of next encoding frames (Frame-1,2,3...n - 1). At the terminal, by using motion vector, Simple Compression is done.
I    B   B   B   B
Fig. 6: Frames of MPEG encoding

\[ \mathbf{P} \rightarrow \mathbf{B} \rightarrow \mathbf{B} \rightarrow \mathbf{B} \]

By using motion vector, Simple Compression is done.

1) When Frame-\( n \) is encoded, the motion vector of Frame-\( n \) is generated.
2) The motion vector of Frame-\( n \) is sent to terminal, and it is used in the Simple Compression of Frame-1,2,3...

1) When Frame-n is encoded, the motion vector of Frame-n is generated.
2) The motion vector of Frame-n is sent to terminal, and it is used in the Simple Compression of Frame-1,2,3...

3) By using motion vector, Simple Compression is done.

Fig. 7: The Example of the Use of Motion Vector

- **Predictor A:**
  It assumes that the object or the frame itself is moving as linear uniform motion. Then the predicted motion vector of frame-\( k \) \( \vec{v}_{k,\text{prev}\#} \) is as follows:
  \[ \vec{v}_{k,\text{prev}\#} = \frac{k}{n} \times \vec{v}_{n} \]  

- **Predictor B:**
  The motion search using block matching algorithm may not reflect a motion correctly. Then, assuming that it does not have change in a motion with each frame, the predicted motion vector of frame-\( k \) \( \vec{v}_{k,\text{prev}\#} \) is assumed as follows:
  \[ \vec{v}_{k,\text{prev}\#} = \vec{v}_{n} \]  

In our method, the difference error of both the predictor A and the predictor B are calculated for every macroblock, and results with fewer errors are chosen for every macroblock. Thereby, since the amount of information decreases compared with simple difference, it is expected that a compression ratio will improve.

### 7. Performance Evaluation

#### 7.1 Experimental Method

To evaluate the performance of the proposed method, 30 types of movies were encoded by using the simple compression. The experimental movies are used from standard images by the Institute of Image Information and Television Engineers (ITE).

These movies are 24 bit full-color images at 360 × 240 pixels. For these movies, P-frame encoding (the encoding that is include for the simple difference) are processed and evaluated.

In this experiment, we compared the amount of data of simple difference and the proposed method using the motion vector between the frame-0 and frame-1 from MPEG-4 encoder. In this experiment, we experimented by using I-frame and P-frame. Only the first frame (frame-0) is I-frame, and from frame-1 to frame-3 are P-frame respectively. Among them, simple compression by the method shown in section 4 was performed at frame-2 and the frame-3, and the amount of compressed data was compared.

#### 7.2 Experimental Results

The results of our compression algorithm of 30 movies without and with motion vector are shown in Fig. 8 and Fig. 9. The result of frame-2 is shown in Fig. 8, and the result of frame-3 is shown in Fig. 9. In both of the compression for frame-2 and frame-3, the difference with frame-0 is used. Because the distance from frame-0 to frame-3 is longer than the distance to frame-2, the amount of compressed data for frame-3 becomes larger than frame-2. As shown in those figures, the amounts of compressed data become from 40 Kbytes to 140 Kbytes. The amounts of data of uncompressed data for those images are about 225 Kbytes. So the compression ratio becomes about from 18% to 62%. Also, 31.1% maximum and 9.0% average of the data amount were reduced by using motion vector.

The picture which has much (11SIF) and little (17SIF) influence by use of a motion vector of those images are shown in Fig.10 and Fig.11. In Fig.10, the whole image flows from right to left. And in Fig.11, the tree is shaking by the wind. The picture which moves smoothly as Fig.10 has much influence by using motion vector. The amount of compressed data of each frame with and without motion vector are shown in Fig.12 and Fig.13. However, since “With MV” in Frame-1 uses the motion vector between Frame-0 and Frame-1, it cannot realize simple compression with motion vector in fact. The values of Frame-1 in those figures are only shown as reference values.

In both figures, the amount of compressed data becomes large as distance with the Frame-0 becomes long. This is because the difference in a picture becomes large as frame distance becomes long. Also, any frame shows the same tendency in the following point, and it turns out that this technique is effective regardless of frame distance.

- In 11SIF, the amount of compressed data is sharply reduced by use of motion vector.
- In 17SIF, there are few effects by use of motion vector, and the difference in the amount of the data of the both method is not seen.
8. Conclusion

In this paper, we presented a simple compression method that can be transmitted via a LAN and we proposed the improved method by using motion vector from video encoder. In this method, difference processing using the motion vector from encoder PC is performed instead of simple difference. Also, the results of our performance evaluation were presented.

As a result from some experimentation, 31.1% maximum and 9.0% average of the data amount were reduced by using motion vector. It is thought that it requires some processing time by execution of this method. But by the reduction of the unnecessary processing such as RGB-YUV conversion and the use of the parallel processing of CPU, etc, it seems that real time processing becomes possible.

As the future work, to evaluate the processing time of proposed method is remained.

References
[1] Windows Media Encoder:
http://www.microsoft.com/japan/windows/windowsmedia
[3] Fixstars CodecSys:
http://www.fixstars.com/products/codecsys/
Fig. 10: Experimental Image (11SIF)

Fig. 11: Experimental Image (17SIF)

Fig. 12: Experimental Result (11SIF)

Fig. 13: Experimental Result (17SIF)