Error Concealment Based on Semantic Prioritization with Hardware-Based Face Tracking

Jae-Beom Lee, Ju-Hyun Park, Hyuk-Jae Lee, and Woo-Chan Lee

With video compression standards such as MPEG-4, a transmission error happens in a video-packet basis, rather than in a macroblock basis. In this context, we propose a semantic error prioritization method that determines the size of a video packet based on the importance of its contents. A video packet length is made to be short for an important area such as a facial area in order to reduce the possibility of error accumulation. To facilitate the semantic error prioritization, an efficient hardware algorithm for face tracking is proposed. The increase of hardware complexity is minimal because a motion estimation engine is efficiently re-used for face tracking. Experimental results demonstrate that the facial area is well protected with the proposed scheme.

Keywords: Error concealment, semantic prioritization, design reuse, MPEG-4, face tracking

Manuscript received Feb. 24, 2004; revised Oct. 6, 2004.

Jae-Beom Lee (email: jbl@mamurian.com) and Ju-Hyun Park (email: pjhyun@mamurian.com) are with Mamurian Design, Seoul, Korea.

Hyuk-Jae Lee (phone: +82 2 880 1825, email: hjlee@ee.snu.ac.kr) and Woo-Chan Lee (email: woochan@snu.ac.kr) are with the School of Electrical Engineering & Computer Science, Seoul National University, Seoul, Korea.

I. Introduction

An IP-based architecture for 3G wireless systems promises to provide next-generation wireless multimedia services such as voice, high-speed data, Internet access, and audio and video streaming on an all IP networks [1]. However, wireless video has bandwidth, delay, and loss requirements that many existing mobile networks cannot provide. To guarantee such quality of service is a hard task in wireless communications because temporally high bit-error rates are inevitable during fading periods [2]-[7]. In fact, a low signal-to-noise ratio and/or signal-to-interface-plus-noise ratio are unavoidable to achieve high system capacity in the wireless access of personal communications systems. Since the transmission of information bits is packet-based, a high bit-error rate results in a high packet-loss ratio. Consequently, the use of error-resilient techniques is highly desirable [8].

Like any other compressed data, compressed video is highly sensitive to data loss. Data loss propagates within the sequence and may become very annoying to end users. Error-resilient schemes have been introduced to limit these impairments [3], [4], [9]-[13]. These schemes could be roughly classified into three categories: error-concealment techniques, error localization techniques, and unequal error protection techniques based on class-based packets. Traditionally, error-handling such as forward-error correction has been performed as a channel coding with communication levels, not with a source coding level. Such an approach has been justified by the fundamental principle of an information theory called the "separation principle." In other words, the best coding system can be obtained when the best source coding and the best channel coding systems are separately designed and combined. Under certain circumstances, however, channel coding techniques cannot be utilized fully. For example, enough redundancies cannot/need not be appended on multimedia bit-streams for very low bit rate wireless applications due to the following reasons. First, channel coding will simply increase the bit-stream too much to protect such a high error rate, thus making the generated bit-streams unfit for the given bandwidth. Second, multimedia streams are generally not "error-sensitive", but are "delay-sensitive"—people can still understand media in the presence of errors most of time, but they cannot endure much of a delay in the course of streaming. Under such multimedia communication scenarios, error-resilient techniques are preferred to a heavy forward error correction code protection overhead. Installation of error-resilient techniques would be more efficient with the source coding bit-stream as in MPEG-4 [14], [15].

MPEG-4 video provides bit-stream devices as error-resilient tools in the source coding level. It is a unique standard that considers error-resilient tools together with source coding functionality. Truly robust video coding requires a diversity of strategies. MPEG-4 video offers the following diverse characteristics about error protection. First, the object-based organization of MPEG-4 video potentially makes it easier to achieve a higher degree of error robustness due to the possibility of prioritizing each semantic object based on its relevance. Second, video object planes (VOPs) already offer a means of resynchronization to prevent the accumulation of errors, and it is possible for an encoder to offer an increased error resilience by placing resynchronization (re-sync) markers in the bit-stream with approximately constant spacing. Third, data partitioning provides a mechanism to increase error resilience by separating the normal motion and texture data of all macroblocks in a video packet and sending all of the motion data followed by a motion marker, followed by all the texture data. Fourth, reversible variable length codes (VLCs) offer a mechanism for a decoder to recover additional texture data in the presence of errors since the special design of reversible VLCs enables the decoding of code words in both the forward (normal) and reverse directions. Fifth, intra-refresh updates the intra coding of macroblocks based on a given rule (for example, a fixed number in each frame), so it can provide a refresh from the coding error [14]-[16].

Resynchronization tools, as the name implies, attempt to enable resynchronization between the decoder and the bit-stream after an error has been detected. Generally, the data between the synchronization point prior to the error and the first point where synchronization is re-established, are discarded. If the resynchronization approach is effective at localizing the amount of data discarded by the decoder, then other types of tools that recover data and/or conceal the effects of errors enhance the bit-stream quality further [14], [16].

In this paper, we propose a semantic error prioritization method to improve error resilience. The basic idea comes from the fact that a transmission error with the video compression standard such as MPEG-4 happens in a video packet (VP) basis, rather than in a macroblock basis. Thus, the proposed method determines the length of a VP based on the importance of its content. For example, if the content is important, such as a face, the length of the VP is short in order to contain only 1 macroblock. If the content does not contain significant information such as the background, the length of the VP is elongated to cover quite a large background area. Such a partition is to be of the same number of VPs for a frame so that the total number of VPs and their overheads are of the same efficiency with the periodic resynch markers typically used in MPEG-4 schemes. The only difference is that the length of the VPs is dependent on the importance of their contents.

To implement the semantic error prioritization method, a facial area should be detected. Face tracking, in general, requires a large amount of computation. This paper proposes a new face tracking technique that efficiently re-uses a motion estimation engine that is available for most modern multimedia processors. The addition of hardware complexity is minimal because the proposed technique simply re-uses existing hardware. Our experimental results are demonstrated under the scenario of an MPEG-4 wireless video.

The structure of this paper is as follows. The idea of the semantic error prioritization method is provided in section II. An efficient hardware re-use technique for face tracking is discussed in section III. The simulation results are presented in section IV. A discussion and concluding remarks follow in section V.

II. Semantic Error Prioritization

1. Resynchronization in MPEG-4

In Fig. 1, a typical video packet is depicted. A resynch marker is used to distinguish the start of a new video packet. This marker is distinguishable from all possible VLC code words as well as the VOP start code. Header information is also provided at the start of a video packet. Contained in this header is the information necessary to restart the decoding process and includes the macroblock address (number) of the first macroblock contained in this packet and the quantization parameter (*quant_scale*) necessary to decode the first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantization parameter allows the differential decoding process to be resynchronized. Following the *quant_scale* is the header extension code (HEC). As the name implies, HEC is a single bit used to indicate whether additional information will be available in the header. If the HEC is equal to "1", then additional information is available. This information includes the modulo time base, *vop_time_increment*, *vop_coding_type*, *intra_dc_vlc_thr*, *vop_fcode_forward*, and

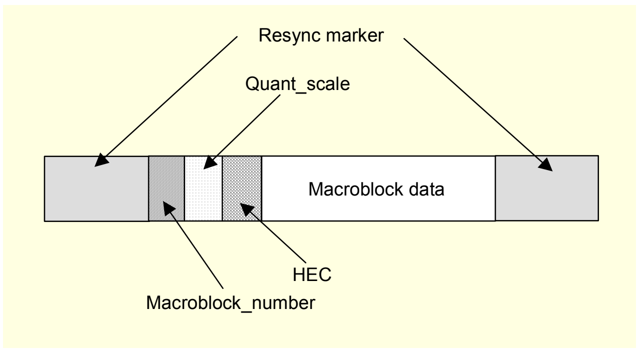

Fig. 1. Resync markers in MPEG-4 video streams.

vop_fcode_backward [14], [16].

The HEC allows each video packet to be decoded "independently." In other words, the necessary information to decode the VP is included in the header extension code field [14], [16].

If the VOP header information is corrupted by the transmission error, it can be corrected by the HEC information—the decoder can detect the error in the VOP header if the decoded information is inconsistent with its semantics. For example, because it is prohibited that the values of the *vop_fcode_forward* and *vop_fcode_backward* be set to "0," the decoder can detect the error in the *fcode* information. In such a case, the decoder can correct the value by using the HEC information of the next VP [14], [16].

When utilizing the error resilience tools within ISO/IEC 14496, some of the compression efficiency tools are modified. For example, all predictively encoded information must be confined within a video packet so as to prevent the propagation of errors. In other words, when predicting (i.e., AC/DC prediction and motion vector prediction) a video packet boundary is treated like a VOP boundary.

2. Semantic Error Prioritization

Since the data between the synchronization point prior to the error and the first point where synchronization is reestablished is typically abandoned, the number of macroblocks discarded is more than just one 8×8 pixel block or one 16×16 pixel block. Therefore, the assumption taken in many literatures that such a small block is only damaged does not conform with standard-based video compression. In addition, if we choose the length of a VP to include only 1 macroblock for an entire video, such an implementation is not realistic due to such a large overhead of VPs. To get around such difficulty, we propose "semantic error prioritization." In MPEG-4, the original idea of the VP approach adopted by ISO/IEC 14496 is based on providing periodic (i.e., almost equi-distant) re-synch markers throughout the bit stream. In other words, the lengths of the VPs are not

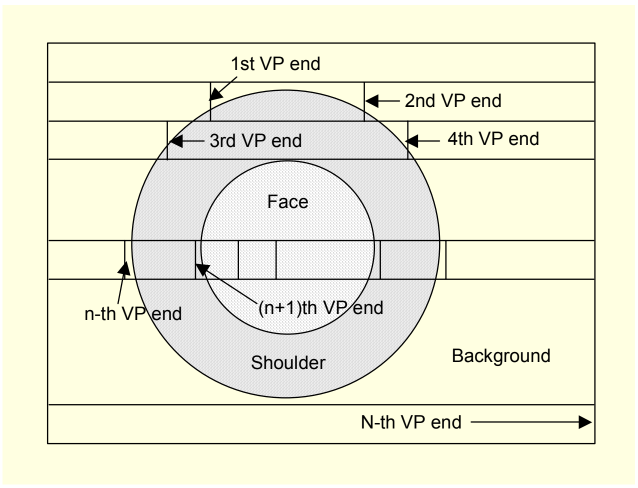

Fig. 2. Semantic construction in video packets.

based on the number of macroblocks, but instead on the number of bits contained in that packet. If the number of bits contained in the current VP exceeds a predetermined threshold, then a new VP is created at the start of the next macroblock. We change the typical use of re-synch markers in the paper to prioritize errors—we use different lengths of VPs for different areas of perceptual importance.

In the semantic error prioritization approach, a VP is characterizing the importance of its content with its length. In other words, if the data currently processed is important like a face, the length of the VP is short to contain only 1 macroblock. If the data currently processed does not contain significant information like a background, the length of the VP is elongated to cover quite a large background area. Such a partition is to be of the same number of VPs for a frame so that the total number of VPs and their overheads are of the same efficiency with the periodic re-synch markers typically used in MPEG-4 schemes. The only difference is that the length of the VPs is dependent on the importance of their contents.

The idea behind the scheme is the following: when the amount of missing information is small, typical image recovery works well—most of the recovery algorithms assume the size to be 8×8 or 16×16 pixels. When the amount of missing information is quite large, typical recovery does not work well. In addition, little information loss in a facial area does not damage the perception of human observers, and a huge information loss in the background area does not matter that much to human observers. Therefore, a perceptually important area is designed to the smallest partition, while a perceptually negligible area is partitioned to a relatively large unit. When an error hits randomly in the bit-stream, a corresponding VP is damaged. Under such a scheme, an important area will be damaged in the smallest unit, while a negligible area will be damaged in a relatively large unit. Doing so is to lead a restoration algorithm to a more efficient and workable one in the next post-processing stage. Figure 2 shows such partitions—in this paper, we have face, shoulders, and background areas, where different sizes of VPs are constructed. We also assume that the HEC mode is used in the paper.

3. Balance Equations

One of the most important tasks is to set the lengths of area-specific video packets for each region. We consider a model of three different areas—face, shoulders, and background in incoming video frames. Given a fixed number of video packets, the first task is to obtain the length of face video packets. The lengths of the background and shoulders are actually pre-fixed since to minimize the numbers of video packets for the areas is necessary as shown in Figure 2. Otherwise, 1-macroblock-sized video packets might not be enough to cover the entire face area with a pre-fixed number of total macroblocks. The total count of macroblocks for a frame is given as follows:

$$\begin{aligned} total\_cnt\_mb = cnt\_face\_mb \\ + cmt\_shoulder\_mb \\ + cmt\_background\_mb, \end{aligned} \tag{1}$$

where *total_cnt_mb*, *cnt_face_mb*, *cnt_shoulder_mb* and *cnt_background_mb* are the total number of macroblocks in a frame, the number of macroblocks in the face area, the number of macroblocks in the shoulders area, and the number of macroblocks in the background area, respectively. The total count of VPs for an entire frame satisfies the following equations:

$$\begin{aligned} total\_cnt\_vp = \frac{cnt\_face\_mb}{length\_face\_vp} \\ + \frac{cnt\_shoulder\_mb}{length\_shoulder\_vp} \\ + \frac{cnt\_background\_mb}{length\_background\_vp}, \end{aligned} \tag{2}$$

$$\begin{aligned} = \frac{cnt\_face\_mb}{length\_face\_vp} \\ + cnt\_shoulder\_vp \\ + cnt\_background\_vp, \end{aligned} \tag{3}$$

where *total_cnt_vp*, *length_face_vp*, *length_shoulder_VP*, and *length _background_vp* are the total number of VPs in a frame, the length of VPs in the face area, the length of VPs in the shoulders area and the length of VPs in the background area, respectively. Also, *cnt_shoulder_vp* and *cnt_background_vp* are the number of VPs in the shoulders area and the number of VPs in the background area. From the above equation, the length of face VPs can be obtained as follows:

$$\begin{aligned} &length\_face\_vp \\ &= \frac{cnt\_face\_mb}{total\_cnt\_vp - cnt\_shoulder\_vp - cnt\_background\_vp} \end{aligned} \tag{4}$$

4. Selection of Lengths of Video Packets

One potential problem is that sometimes 1-macroblock-sized VPs cannot be assigned for all facial areas as shown in Fig. 3. For example, if the total count of VPs is 33 and the number of VPs is not enough to assign 1-macroblock-sized VPs to all facial areas, some of the face VPs are given to VPs whose lengths are simply more than 1 macroblock. The algorithm to resolve this issue is practically important. The policy to select the length of the VP in the face area can be as follows. First, if *length_face_vp* = 1 from (4), 1 macroblock is taken as the length of the VP. Second, if *length_face_vp* > 1 from (4), the following equations hold:

$$m + \frac{n}{k} = cnt\_face\_vp\ , \tag{5}$$

$$m + n = cnt\_face\_mb\ , \tag{6}$$

Above two equations result in:

$$m = \frac{k(cnt\_face\_vp) - cnt\_face\_mb}{k-1}\ , \tag{7}$$

Starting from $k = 2$, m values are iteratively obtained. (i.e., $k = 2,3,\ldots$) When we get the first valid m, we stop the iteration. The valid m is between 2 and $m+n-2$. Third, if *length_face_vp* < 1 from (4), we select the length of the VP in the face area to be 1

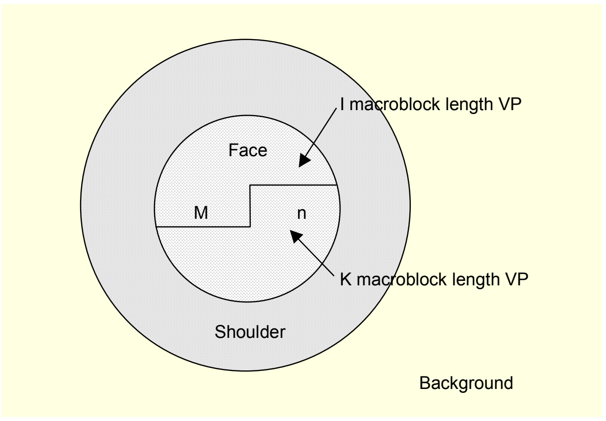

Fig. 3. The shape of VPs in a facial area.

macroblock. If the *total_cnt_vp* is not met, then the macroblocks in the shoulder area are constructed as 1 macroblock VPs from the upper-left corner. This continues until the overall number of VPs count is *total_cnt_vp* .

III. Face Tracking

To implement the semantic error prioritization technique, face tracking is essential. Face tracking is a technique to locate and track the human face and its features in a typical head-and-shoulders video sequence. Face tracking requires a large amount of computation and needs additional hardware dedicated to the computation. As a result, if a video processor performs face tracking, its cost is increased due to the increase of the chip area occupied by the additional hardware. In addition, the additional hardware may also increase the power consumption, which is a critical issue for wireless devices. Thus, the employment of face tracking can be beneficial only when the hardware cost and power consumption are not greatly increased by the additional hardware for this technique.

We propose a new face tracking algorithm that does not significantly increase hardware cost by re-using existing hardware available in most video processors. The algorithm is designed such that most of the computation for face tracking is performed using a discrete cosine transform (DCT) engine as well as a motion estimation (ME) engine. Only a small part of computation is performed by a general-purpose RISC processor without demanding much computation power. Thus, the proposed algorithm can be effectively performed in a modern video processor without the increase of hardware cost and power consumption.

1. Face Tracking Algorithm

As shown in Fig. 4(a), the algorithm consists of two steps. The first step is to generate an edge-detected binary image that extracts the edges of an original image. The second step is to compare the edge-detected binary image with reference template images that contain the edge of a human face. The best-matched template image is used to locate the human face. To perform the first step, it is necessary to obtain a down-sampled image as shown in Fig. 4(b). Down sampling naturally involves pre-filtering. To reduce the computation time of pre-filtering, this paper uses transform domain filtering which can be performed by a DCT engine already existing in modern video processors. The down-sampling factors are chosen to fit the original video in the search window of the ME engine. In [17], the search window used in the hardware is assumed to be 31×31 pixels. For the input video in the CIF and QCIF formats, the down sampling factors are chosen to be 10 and 5, respectively. The purpose of choosing these specific factors is to make the final image size to be 34×28 pixels. With this size, the down sampled image fits in the search window by removing three row lines (the two uppermost lines and the one lowermost line). Once a down-sampled image is obtained, a Sobel operator is used to extract the edges.

After the edge-detected binary image is obtained, the next step is to compare the image with various reference images that contain a template for a human face. Figure 4(c) shows four templates. These templates contain an ellipse with the major axis lengths of the upper-half ellipse being 5, 6, 7, and 8. The sizes of the minor axis lengths of the upper-half ellipse are 3, 4, 5, and 6. For each of the ellipse templates the values of the boundary are set to 25, while the interior values are set to 200. The main reason to set the interior values as 200 is to identify eyes-nose-mouth features inside the face explained later in this section. The rest of the data in the search window is set to 0 because it should not confuse the template matching.

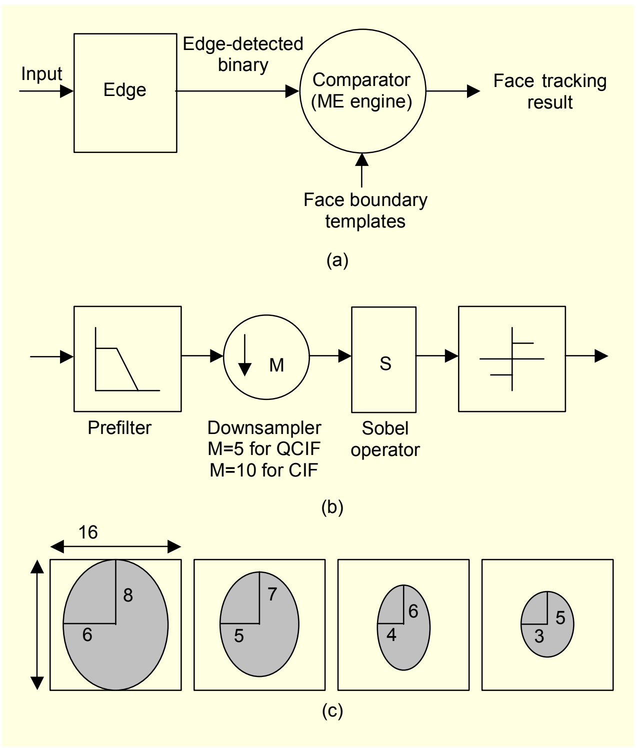

Fig. 4. Face tracking algorithm: (a) face tracking with the use of an ME engine, (b) generation of an edge-detected binary image, and (c) face boundary templates.

2. Mapping onto an MPEG-4 Processor

This section considers the mapping of the algorithm introduced in the previous subsection onto a real video processor, Vidan. Vidan is a video processor developed by Mamurian Design. It is evolved from the MoVa processor, an MPEG-4 video codec processor originally designed by ETRI (Electronics and Telecommunications Research Institute) in Korea. Vidan adds more features to Mova to support improved error resilience as well as various frame rates and image sizes. In addition, JPEG encoding and decoding are also supported. The processor includes hardware engines such as ME, DCT, VLD, VLC, and Deblocking that perform most of the MPEG-4 computation. I/O interfaces are also included to communicate with outside devices. With the dedicated hardware, the core processor can run at a very low clock frequency of 27 MHz, and some hardware components run even at a slower clock frequency of 13.5 MHz.

An ME engine is efficiently used to find the template that best matches the edge-detected binary image and locates the position of a face. Each template is given to the ME engine as the input and then compared with the down-sampled edge-detected binary image. The ME engine gives back the best-matched position. After all templates are compared, the best-matched template with the best-matched position is selected as the final result. The ME engine in the Vidan processor has a search window size of 48×48 pixels, while the template size is 16×16. Both the search window and the template are down sampled by a factor of two before the start of an ME operation so that the amount of computation time can be reduced. To fit the search window, the edge-detected binary image needs to be enlarged to 48×48 pixels by padding zeros to the pixels in the boundary area of the image. The down sampling inside the Vidan ME engine severely degrades the accuracy of face tracking, and consequently, the resulting average success rate drops to 72.4 percent. This is a severe drop-off compared to the average success rate of 95.5 percent when the search window and the template sizes are 31×31 and 16×16 pixels, respectively. Thus, a straightforward mapping of the algorithm in the previous subsection to the Vidan processor does not produce a satisfactory result.

In order to avoid the performance degradation by the down sampling inside the ME engine, this paper proposes to enlarge the input image and template sizes by 64×64 and 32×32 pixels, respectively. After down sampling, the window sizes become 32×32 and 16×16 pixels, respectively. These sizes are best considering the trade-off between the computation amount and performance as shown by simulation results in the next section. However, these window sizes are too large for the ME engine to handle. Therefore, they must be decomposed to fit in the ME engine input size. The search window needs to be decomposed into four 48×48 pixel subblocks as shown in Fig. 5. Note that the decomposed blocks of the search window are not disjoint blocks. The template is also decomposed into four disjoint 16×16 pixel subblocks. The upper-left reference subblock and search subwindow are given as the inputs to the ME engine. These blocks are shaded in Fig. 5. Once the ME operation is complete, the upper-right reference subblock and search subwindow are also given to the ME engine. Four ME operations are required to perform the complete ME operation.

Consider the modification of the ME engine to perform four ME operations. The ME engine of the Vidan processor is made of source-pixel-based linear arrays as shown in Fig. 6 [17]. Two data buses, s1 and s2, are used to provide the search window while one bus, t, is used to input the template. At each clock cycle, each processing element (PE) computes the absolute difference of two input pixels, accumulates it to the value received from the left PE, and passes the result to the right PE. The rightmost PE (PE7) generates the final result accumulating the absolute differences between all the pixels. Then, the result is given to the comparator unit to generate the motion vector. To run the proposed mapping technique, the ME engine needs to run four times. When the first ME operation is finished, the rightmost PE should not feed the result to the comparator. Instead, the result is fed back to the leftmost PE (PE0) so that it can be accumulated to the result of the second ME operation. For the second and third ME operations, the results of PE7 are also fed back to PE0 to be accumulated

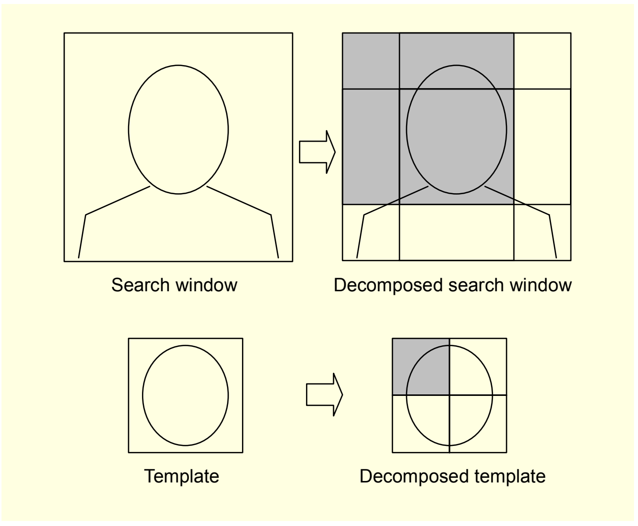

Fig. 5. Decomposed search window and template.

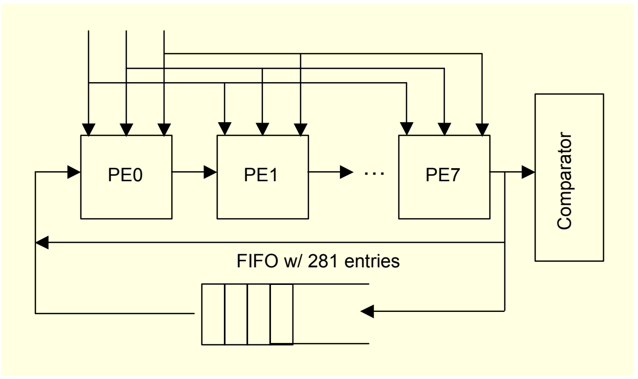

Fig. 6. Modified source-pixel-based array architecture for the proposed motion estimation.

Only for the last ME operation, the comparator unit operates and generates the final motion vector.

The feedback path has a first in, first out FIFO that stores the intermediate result and provides the value to PE0 at the right time. Recall that the Vidan ME engine has a search window of 48 x 48 pixels and a template of 32×32. After the search window and template are down sampled inside the ME engine, their sizes become 24×24 and 16×16 pixels, respectively. Thus, the ME engine computes the absolute difference for 289 (17×17) positions. For all these positions, the intermediate results are computed and then stored to be accumulated by the next ME operation. Among these results, eight are stored in the PEs. The remaining 281 entries are stored outside the PEs. Thus, a FIFO with 281 entries is used as shown in Fig. 6. In this way, the mapping technique is implemented with minimal additional hardware. The structure of PEs as well as their interconnections is not changed. The additional hardware components are a 281-entry FIFO as well as a circuit for controlling the comparator unit and the FIFO.

IV. Experimental Result

Data compression takes advantage of variable length codes (VLCs) that correspond to the probabilities of their symbols. Each symbol has its own length—so, when two symbols are concatenated, the second symbol can be interpreted only after the successful parsing of the first symbol. One dangerous problem about VLCs is that it is difficult to find the right boundary under the bit errors due to the "variable length" characteristics. In other words, once a wrong boundary is detected, all decoded symbols after that point are not expected to be correct. Many times, decoders just stop parsing since they cannot even find even codes. Therefore, re-synchronization is necessary to fresh-start the parsing of the right codes after a decoder finds any error in the parsed bitstream. This concept has been imbedded in MPEG-4 as "re-synch markers." To this end, a VP is defined as a data stream between two adjacent re-synch markers. Furthermore, there are advanced tools developed in MPEG-4 to recover more symbols from even a damaged VP—if we are lucky, quite many VLCs can be recovered from a damaged VP. Those advanced tools include the RVLC tool that makes a reverse decoding possible even in a damaged VP. Of course, we do not get any other symbol recovered for the worst case. In this paper, we assume the worst case scenario where we only find the synchronization point for further decoding without any other symbol being recovered.

Figures 7 through 9 show the simulation results about semantic error prioritization. Figure 7 compares the proposed method with a typical MPEG-4 VP method when a face error happens. Figures 8 and 9 show results of the proposed method when shoulder and background errors occur. In Figure 7, the error hits the same place in the bitstream in both (a) and (b). A typical method uses VPs of relatively uniform length over the entire frame, while the proposed method uses a face and shoulder model with different VP sizes in a frame. As aforementioned, the size of VPs for a face is 1 macroblock, while the size of VPs for shoulders is the length of the shoulders in the fitting model. For the background, the size of the VPs can be full length since it does not contain important information. Consequently, an error position is interpreted semantically (i.e., face, shoulder and background) in the decoder as shown in Figs. 7 through 9. For face areas, the size of the damaged areas in the proposed method is smaller than those of

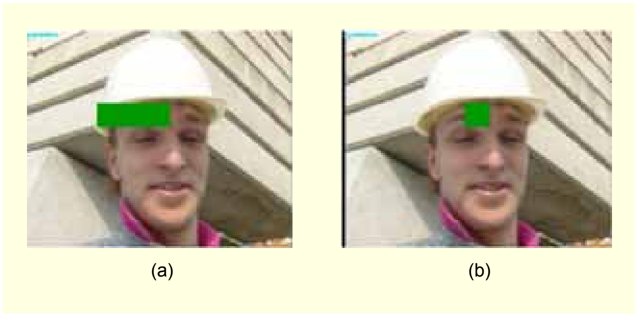

Fig. 7. A typical result of a face error in a frame: (a) uniform error in typical MPEG-4 VP and (b) semantic error prioritization.

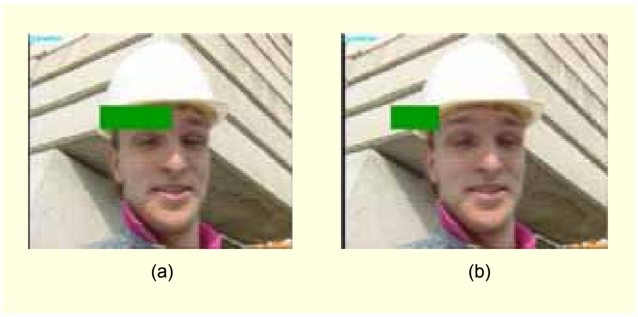

Fig. 8. A typical result of a shoulder error in a frame: (a) uniform error in typical MPEG-4 VP and (b) semantic error prioritization.

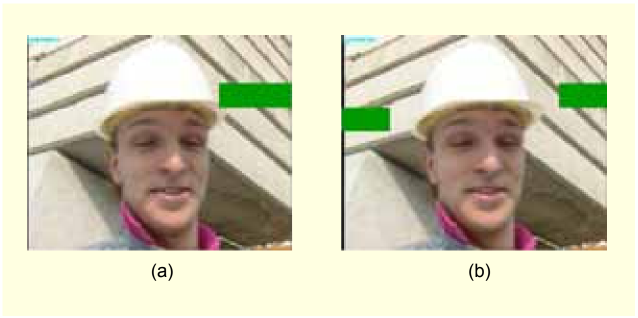

Fig. 9. A typical result of a background error in a frame: (a) uniform error in typical MPEG-4 VP and (b) semantic error prioritization.

a typical method while, for background areas, the size of the damaged areas in the proposed method is larger than that of a typical method—thus making the number of VPs in a frame be the same between the two methods.

Figures 10 through 12 show the simulation results of the semantic error prioritization under multiple errors in the bit-stream. Figure 10 compares the proposed method with a typical MPEG-4 VP method when two errors happen, while Fig. 11 shows the results of a typical method and the proposed method when three errors happen in the bit-stream. In Fig. 12, the results of ten errors have been shown. From the simulation results, the face area is relatively well protected from bit-stream damage, while the background area is not protected at all. This is the effect of the proposed method due to different VP sizes with different semantic areas. The lengths of VPs in the face area are the smallest, while those in the background area are the largest. The lengths of VPs in the shoulder area are in-between. In other words, damage to the face area is suppressed maximally, while damage to the background area is suppressed minimally. Overall, visual perception is definitely improved with the proposed method. Even with the worst case scenario in Fig. 12, we can still see some face area that contains the most important information. Therefore, the proposed method allows a more comfortable image quality.

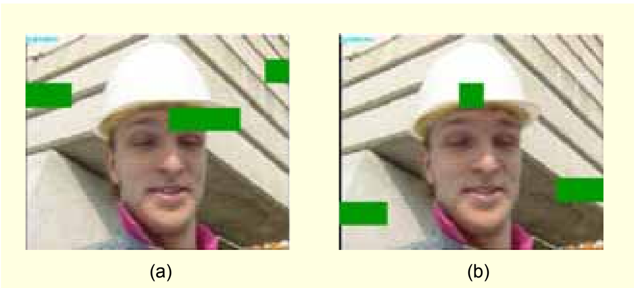

Fig. 10. A typical result of 2 random errors in a frame: (a) uniform error in typical MPEG-4 VP and (b) semantic error prioritization.

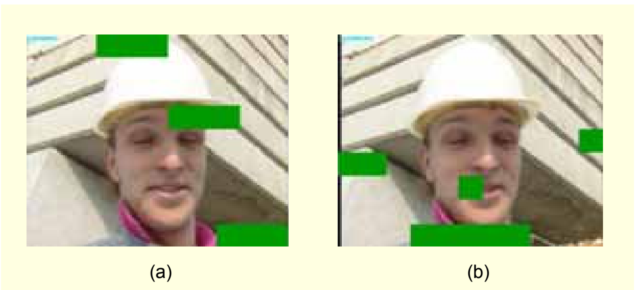

Fig. 11. A typical result of 3 random errors in a frame: (a) uniform error in typical MPEG-4 VP and (b) semantic error prioritization.

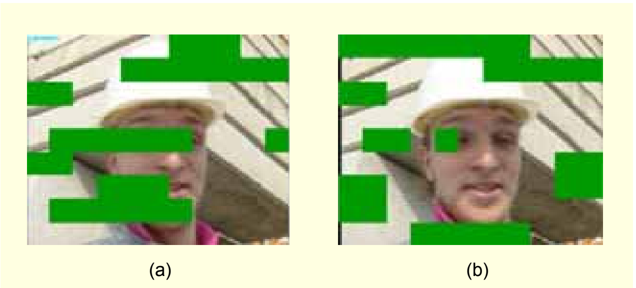

Fig. 12. A typical result of 10 random errors in a frame: (a) uniform error in typical MPEG-4 VP and (b) semantic error prioritization.

In Fig. 13, the simulation results are given to show the average success rate of face tracking depending on the window size. The size of the template varies from 8×8 to 40×40 pixels. For the search window size, the appropriate size corresponding to each template size is chosen. For example, $(8\times8, 22\times18)$ in the horizontal access shows that the template size is 8×8 pixels and the corresponding search window size is 22×18. Four image sequences, Akiyo, Claire, Grandma, and Missam are used for the simulation. The figure shows that the success rate increases as the window size increases. However, one cannot simply choose the largest possible window size because the amount of computation increases in proportion to N^4 for the template size of $N\times N$. Note in the figure that the success rate approaches near one hundred percent as the window size is $(16\times16, 27\times33)$. Thus, these simulation results lead to the conclusion that the window size of $(16\times16, 27\times33)$ is large enough to achieve a near optimal success rate without too much increase of computation. Note that the mapping technique developed in section III is tuned for this window size. This is because down sampling of $(32\times32, 64\times64)$ inside the Vidan ME engine results in $(16\times16, 32\times32)$ which is close to $(16\times16, 27\times33)$. Thus, the window size used in section III is the best window size considering the trade-off between the computation amount and performance.

V. Conclusions

In this paper, we devised a semantic error prioritization method to enhance the visual perception of human vision systems under data-loss scenarios. Typically, a human vision system is used to locate faces and track them in head-and-shoulders video sequences. Human observers are very sensitive to face information, while they are not so to background information. To take advantage of this aspect, we devised a way to vary the size of VPs in different semantic areas to improve visual quality even under data loss. Experimental results are demonstrated under the scenario of an MPEG-4

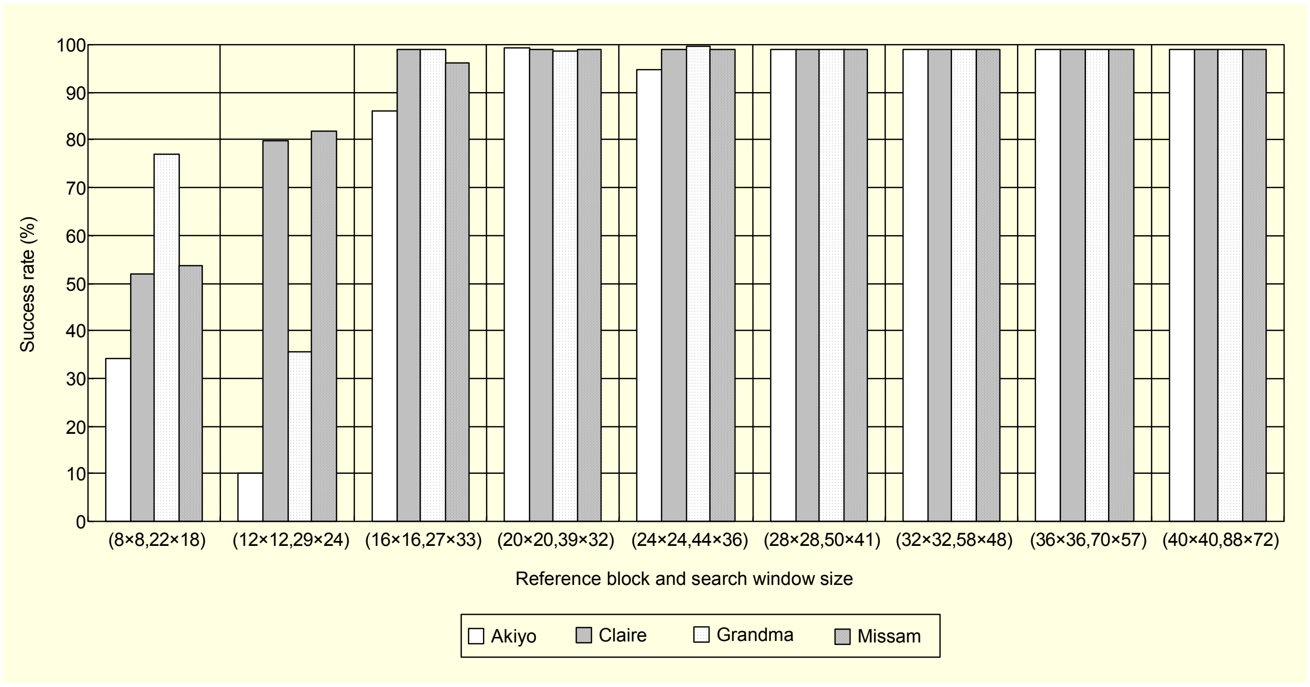

Fig. 13. Success rate vs. the template size.

wireless video and show that a facial area is relatively well protected from bit-stream damage, while a background area is not protected at all. This algorithm can be applied based on not only a pure software form but also a hardware reuse technique with the ME engine, as explained in the corresponding section. The increase of a hardware area is minimal because the existing ME engine is reused with minimal modification. An interesting future research topic is to extend the proposed algorithm to track multiple faces in a single frame. The effects of brightness changes of human faces may also be studied in the future.

References

[1] 3GPP2 Specification, Requirements for a 3G Network Based on Internet Protocol with Support for TIA/EIA-41 Interoperability, Oct. 2000.

[2] T. Stockhammer, H. Jenkac, and C. Weib, “Feedback and Error Protection Strategies for Wireless Progressive Video Transmission,” *IEEE Trans. Circuits Syst. Video Technol.,* June 2002, pp. 465-482.

[3] Y. Wang and S. Lin, “Error-Resilient Video Coding Using Multiple Description Motion Compensation,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 12, no. 6, June 2002, pp. 438-452.

[4] J. Zhang, J. F. Arnold, and M. R. Frater, “A Cell-Loss Concealment Technique for MPEG-2 Coded Video,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 10, no. 4, June 2000, pp. 659-665.

[5] N. Farber, B. Girod, and J. Villasenor, “Extensions of ITU-T Recommendation H.324 for Error Resilient Video Transmission,” *IEEE Communications Magazine,* June 1998, pp. 120-128.

[6] Raj Talluri, “Error-Resilient Video Coding in the ISO MPEG-4 Standard,” *IEEE Communications Magazine,* June 1998, pp. 112-119.

[7] L. Hanzo, C. H.Wong, and P. Cherriman, “Channel-Adaptive Wideband Wireless Video Telephony,” *IEEE Signal Processing Magazine,* July 2000, pp. 10-30.

[8] S. Zhao, Z. Xiong, and X.Wang. “Joint Error Control and Power Allocation for Video Transmission over CDMA Networks with Multiuser Detection,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 12, no. 6, June 2002, pp. 425-437.

[9] D.S. Turaga and T. Chen, “Model-Based Error Concealment for Wireless Video,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 12, no. 6, June 2002, pp. 483-495.

[10] F. Frossard and O. Verscheure, “AMISP: A Complete Content-Based MPEG-2 Error-Resilient Scheme,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 11, no. 9, Sept. 2001, pp. 989-998.

[11] S. Tsekeridou and I. Pitas, “MPEG-2 Error Concealment Based on Block-Matching Principles,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 10, no. 4. June 2000, pp. 646-658.

[12] P. Chang and T. Lee, “Precise and Fast Error Tracking for Error-Resilient Transmission of H.263 Video,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 10, no. 4, June 2000, pp. 600-607.

[13] W.J. Chen and J. Hwang, “The CBERC: A Content-Based Error-Resilient Coding Technique for Packet Video Communications,” *IEEE Trans. Circuits Syst. Video Technol.,* vol. 11, no. 8, Aug.

2001, pp. 974-980.

[14] A. Puri and A. Eleftheriadis, "MPEG-4: An Object-Based Multimedia Coding Standard Supporting Mobile Applications," *Mobile Networks and Applications,* vol. 3, Mar. 1998, pp. 5-32.

[15] Touradj Ebrahimi, "MPEG-4 Video Verification Model : A video Encoding/Decoding Algorithm Based on Content Representation," *Signal Processing: Image Communication,* vol. 3, no. 1, June 1997, pp. 26-40.

[16] Motion Picture Expert Group, ISO/IEC JTC1/SC29/WG11, Working Draft 3.0 of ISO/IEC 14496-2, MPEG96/N1643, Bristol, Apr. 1997.

[17] Vidan-Low Power/High Performance Single Chip MPEG-4 Video/JPEG Still Image Multi-functional Codec, http://www.mamurian.com.

[18] N. Brady, "MPEG-4 Standardized Methods for the Compression of Arbitrarily Shaped Video Objects," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 9, no. 8, Dec. 1999, pp. 1170-1189.

[19] D. Zhang and Z. Wang, "Image Information Restoration Based on Long-Range Correlation," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 12, no. 5, May 2002, pp. 331-341.

[20] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards-Algorithms and Architectures,* 2ed., Kluwer Academic Publishers, 1997.

Jae-Beom Lee has been a video architect at Intel Corporation in the Silicon Engineering Division, Consumer Electronics Groups in Oregon, USA since 2003. Prior to that, he worked as a Director of Engineering at Mamurian Design, an MPEG-4 video chipset company in Seoul, Korea, from 2001 to 2003. He served as a Principal Engineer and Codec Group Manager at SolidStreaming, a NY-based startup in Manhattan, in 2001. He worked as a Senior Engineer for C-Cube Microsystems from 1999 to 2001. He received the PhD degree from the Department of Engineering at Columbia University, New York City, USA.

Ju-Hyun Park received the BE, ME, and PhD degrees from Chonnam National University in 1993, 1995, and 1999. From 1999 to 2002, he was with ETRI in Daejeon, Korea, where he worked as a research scientist in the VLSI Architecture Team. Since 2002, he has been engaged with Mamurian Design Inc. (http://www.mamurian.com), in Seoul, Korea, and has been involved in the research and development of SoC multimedia silicon chips. His research interests include the design of SoC for multimedia applications.

Hyuk-Jae Lee is an Associate Professor at the School of Electrical Engineering and Computer Science in Seoul National University. Previously, he worked as a Senior Component Design Engineer at the Server and Workstation Chipset Division of Intel and as an Assistant Professor in the Computer Science Department of Louisiana Tech University. He received the PhD degree from the School of Electrical and Computer Engineering of Purdue University, West Lafayette, Indiana in 1996 and the BS and MS degrees from the Department of Electronics Engineering of Seoul National University in 1987 and 1989. His current research interest is the design of SoC for multimedia applications.

Woo-Chan Lee is an MS candidate at the School of Electrical Engineering and Computer Science of Seoul National University. He received the BS degree from the School of Electrical Engineering of Seoul National University in 2002. His current research interest are the error concealment algorithm and the design of SoC for multimedia applications.