14th European Signal Processing Conference (EUSIPCO), September 4-8, 2006, Florence, Italy

AN ADAPTIVE ERROR CONCEALMENT MECHANISM FOR H.264/AVC ENCODED LOW-RESOLUTION VIDEO STREAMING

Olivia Nemethova, Ameen Al-Moghrabi and Markus Rupp

Institute of Communications and Radio-Frequency Engineering
Vienna University of Technology
Gusshausstrasse 25/389, 1040 Vienna Austria
{onemeth, mrupp}@nt.tuwien.ac.at

ABSTRACT

Low-rate video is widely used especially in mobile communications. H.264/AVC (advanced video coding) is well suited for the real-time error resilient transport over packet oriented networks. In real-time communications, lost packets at the receiver cannot be avoided. Therefore, it is essential to design efficient error concealment methods which allow to visually reduce the degradation caused by the missing information. Each method has its own quality of reconstruction. We implemented various efficient error concealment techniques and investigated their performance in different scenarios. As a result, we propose and evaluate an adaptive error concealment mechanism that accomplishes both - good performance and low complexity enabling the deployment for mobile video streaming applications. This mechanism selects suitable error concealment method according to the amount of instantaneous spatial and temporal information of the video sequence.

1. INTRODUCTION

For transmission of video over packet networks (both fixed and mobile), usually low resolutions of 176×144 pixel (QCIF) and 352×288 pixel (CIF) are used. In such small resolutions each pixel represents essential part of a picture. To avoid the effect of rather high overhead used by RTP/UDP/IP protocols, the encapsulated video part is usually large. Thus, the packet loss that is inevitable in real-time communications, leads to a considerable perceptual visual quality degradation. Error concealment methods aim to visually reduce such degradation. The larger the degraded area, the more difficult it is to conceal an error. To facilitate this, H.264/AVC [1] also offers several new error resilience features. One of them is the choice of the slicing type, which determines the size and the shape of the missing area after the erroneous reception; another example is the usage of redundant slices. However, H.264/AVC also brought some new challenges to the error concealment, most important of which is the spatial error propagation caused by the intra-frame prediction of its I frames. Many error concealment methods has been studied so far [2]. In this work, we focus on methods that can be applied to video streaming over mobile networks. Thus, we also have to consider the power and complexity limitation of a mobile terminal as well as a processing delay constraint. Simple temporal error concealment suitable for I frames if there was no scene change has been proposed in [3]. In [4] a frame interpolation method based on motion vector estimation for H.263 video streaming is proposed. Combination of methods, depending on the type of frame and movement is introduced in [5], using simple motion estimation method for the intra frame reconstruction. In [6] some improvements to H.264/AVC decoder [7] concealment are proposed and tested; a scene change detector, based on the generic H.264/AVC temporal activity detection is used to decide which method to call. Some more refinements to H.264/AVC spatial and temporal error concealments were proposed in [8]. The intention of this paper is to present a fully automatic low-complexity mechanism to choose the most appropriate error concealment method for a given situation. We implemented several improved spatial and temporal error concealment methods into the H.264/AVC [7] decoder and tested their performance for realistic streaming conditions. The results clearly demonstrate how important it is to choose the appropriate method according to the above mentioned conditions.

The paper is structured as follows: Sections 2 and 3 present spatial and temporal error concealment methods we implemented. Section 4 introduces the adaptive mechanism of the selection process. In Section 5 the implementation of the adaptive mechanism in H.264/AVC is discussed, experimental results are shown and evaluated. Section 6 contains conclusions and some final remarks.

2. SPATIAL ERROR CONCEALMENT

2.1 Weighted Averaging

The simplest and often used method is weighted averaging. Each pixel $p(i,j)$ of a missing macroblock is interpolated as a linear combination of the nearest pixels in the boundaries.

$$p(i,j) = \frac{d_R p_L + d_L p_R + d_B p_T + d_T p_B}{d_L + d_R + d_T + d_B}, \qquad (1)$$

where d_L, d_R, d_T, d_B are the distances between the interpolated pixel and the nearest pixel $p_L = p(i,0)$ in left, $p_R = p(i,N+1)$ in right, $p_T = p(0,j)$ in top and $p_B = p(N+1,j)$ in bottom boundary respectively as shown in Figure 1 left; N is the size of the macroblock. This method only performs well if the missing block is smooth, otherwise it produces visible artifacts.

2.2 Directional Interpolation

To smooth along the edges we implemented a directional interpolation based method presented in [9]. First, we detect the edges using a horizontal and vertical Sobel mask:

$$\mathbf{S}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \mathbf{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \qquad (2)$$

The masks are applied on the luminance value $y_{i,j}$ of each pixel at the missing area boundaries as follows:

$$G_x(i,j) = \mathrm{vec}(\mathbf{S}_x)^T \mathrm{vec}(\mathbf{F}(i,j)), \qquad (3)$$

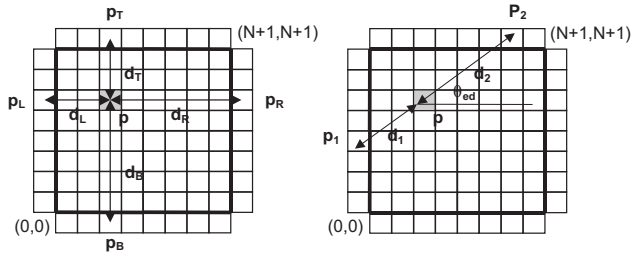

Figure 1: Spatial interpolation: weighted averaging (left), directional interpolation showed for the whole block (right).

$$G_y(i,j) = \mathrm{vec}(\mathbf{S}_y)^T \mathrm{vec}(\mathbf{F}(i,j)), \tag{4}$$

$\mathbf{F}(i,j)$ being the relevant part of a boundary:

$$\mathbf{F}(i,j) = \begin{bmatrix} y_{i-1,j-1} & y_{i-1,j} & y_{i-1,j+1} \\ y_{i,j-1} & y_{i,j} & y_{i,j+1} \\ y_{i+1,j-1} & y_{i+1,j} & y_{i+1,j+1} \end{bmatrix}. \tag{5}$$

Magnitude of the gradient $|G(i,j)|$ and its direction $\theta_g(i,j)$ can then be calculated for each pixel at the missing block boundary as follows:

$$|G(i,j)| = \sqrt{G_x^2(i,j) + G_y^2(i,j)}, \tag{6}$$

$$\theta_g(i,j) = \arctan\left(\frac{G_y(i,j)}{G_x(i,j)}\right). \tag{7}$$

Please, note that the slope $a(i,j)$ of the edge, perpendicular to the gradient direction θ_g, can be expressed as follows:

$$a(i,j) = \frac{G_y(i,j)}{G_x(i,j)} = \cot\left[\theta_g(i,j)\right]. \tag{8}$$

The dominant gradient direction within certain area A can be subsequently determined as a sum of all pixel gradients in A weighted by their magnitude:

$$\theta_{gd} = \frac{\sum_{\forall i,j \in A} \theta_g(i,j)|G(i,j)|}{\sum_{\forall i,j \in A} |G(i,j)|}. \tag{9}$$

We further prolong the detected edge with direction θ_{gd} (with corresponding edge slope $a_d = \cot\left[\theta_{gd}\right]$ and interpolate the block or its part in that direction by means of weighted averaging (Figure 1, right):

$$p(i,j) = \frac{1}{d_1 + d_2}\left[d_2 p_1 + d_1 p_2\right], \tag{10}$$

where p_1 and p_2 are the points at the boundaries from which the missing pixel is interpolated. They can be obtained as an intersection of the block boundaries with a line of slope a_d. Symbols d_1 and d_2 denote the distance of $p(i,j)$ from p_1 and p_2 respectively:

$$p_1 = p(i_1, j_1), \qquad p_2 = p(i_2, j_2), \tag{11}$$

$$i_1 = \max\left[i - i \cdot \frac{1}{a_d}; 0\right], \tag{12}$$

$$j_1 = \max\left[j - j \cdot a_d; 0\right], \tag{13}$$

$$i_2 = \min\left[i + i \cdot \frac{1}{a_d}; N+1\right], \tag{14}$$

$$j_2 = \min\left[j + j \cdot a_d; N+1\right], \tag{15}$$

$$d_1 = \sqrt{(i-i_1)^2 + (j-j_1)^2}, \tag{16}$$

$$d_2 = \sqrt{(i+i_2)^2 + (j+j_2)^2}. \tag{17}$$

The coordinates are rounded to integer values. To support more than one edge per macroblock, segmentation to areas belonging to all important entering edges is performed according to [9]. Subsequently, for each segment directional smoothing is performed. Segment based smoothing makes this method benefiting over the one used in [8], where the H.264/AVC generic spatial prediction type information from the neighbouring macroblocks is used to conceal the missing one, not supporting more than one edge.

3. TEMPORAL ERROR CONCEALMENT

If the residuals are lost but the motion vectors (MV) correctly received, the simplest is to decode the missing block by setting the missing residuals to zero. If the whole macroblock information got lost, the simplest so-called "copy-paste" method can be used. The missing block is replaced by spatially corresponding block from the previous frame. This only performs well for low-motion sequences. Better performance is provided by motion compensated interpolation methods described in the following.

3.1 Motion vector interpolation

If the motion vectors $\mathbf{mv}_D : D \in \{T,B,L,R\}$ of the top, bottom, left and right neighbouring macroblock are known, then the motion vector $\mathbf{mv}$ of a missing block may be easily approximated by an average of those:

$$\widehat{\mathbf{mv}} = \frac{1}{M}\sum_D \mathbf{mv}_D. \tag{18}$$

The missing block will be replaced by the block in the previous frame having the position indicated by $\widehat{\mathbf{mv}}$. Performance of this method is limited for lower resolutions as each macroblock can also contain parts of different objects moving in different directions. However, H.264/AVC supports multiple block sizes for motion compensation which can be used to refine the motion estimation [8]. We implement a refined motion vector interpolation: The missing macroblock (16×16) is first segmented in smaller blocks (8×8, 4×4 or 2×2). For each such block its MV is estimated as a weighted averaging of the motion vectors belonging to the nearest neighbouring blocks. If no ABT was used, the motion vectors on the boundary D will be the same for all the blocks. Let $\mathbf{mv}_D^{(k)}$ be the motion vector of the k-th block in the boundary D. The estimated motion vector $\widehat{\mathbf{mv}}^{(i,j)}$ of the block being on the position (i,j) within the missing macroblock can be written as follows:

$$\widehat{\mathbf{mv}}^{(i,j)} = \frac{d_R \mathbf{mv}_L^{(j)} + d_L \mathbf{mv}_R^{(j)} + d_T \mathbf{mv}_B^{(i)} + d_B \mathbf{mv}_T^{(i)}}{d_L + d_R + d_T + d_B}. \tag{19}$$

This method is directly applicable only for P and B frames. For I frames the motion vectors have to be estimated as described in the Sections 3.2 and 3.3.

3.2 Boundary matching

Let $\mathbf{B}$ be the area corresponding to a one pixel wide boundary of a missing block in the n-th frame $\mathbf{F}_n$. Motion vectors of the missing block as well as those of its neighbours are unknown (occurs mostly for I frames, but also in specific cases for P frames with some inserted I macroblocks). We are looking for the coordinates $[\hat{x},\hat{y}]$ of the best match to $\mathbf{B}$ within the search area $\mathbf{A}$ in the previous frame $\mathbf{F}_{n-1}$:

$$[\hat{x},\hat{y}] = \arg\min_{x,y\in\mathbf{A}} \sum_{i,j\in\mathbf{B}} |\mathbf{F}_{n-1}(x+i,y+j) - \mathbf{B}(i,j)|. \quad (20)$$

The sum of absolute differences (SAD) was chosen as a similarity metric for its low computational complexity. The size of $\mathbf{B}$ depends on the number of correctly received neighbours M, boundaries of which are used for matching. The area $\mathbf{A}$ is an important design parameter and should be chosen with respect to the amount of motion in the sequence. In our implementation $\mathbf{A}$ was a squared area of 9×9 pixels, with its center having the same position within $\mathbf{F}_{n-1}$ as $\mathbf{B}$ within $\mathbf{F}_n$. The macroblock sized area starting at the position $[\hat{x},\hat{y}]$ in $\mathbf{F}_{n-1}$ is taken to conceal the damaged macroblock in $\mathbf{F}_n$.

3.3 Block matching

Better results can be obtained by searching the best match for the correctly received macroblocks $\mathbf{MB}_D : D \in \{T,B,L,R\}$, neighbouring the missing one on top, bottom, left or right side respectively:

$$[\hat{x},\hat{y}]_D = \arg\min_{x,y\in\mathbf{A}_D} \sum_{i,j\in\mathbf{MB}_D} |\mathbf{F}_{n-1}(x+i,y+j) - \mathbf{MB}_D(i,j)|, \quad (21)$$

$\mathbf{A}_D$ representing the search area for the best match of $\mathbf{MB}_D$, with its center spatially corresponding to the start of the missing macroblock. The final position of the best match is given by

$$\hat{x} = \frac{1}{M}\sum_D \hat{x}_D; \quad \hat{y} = \frac{1}{M}\sum_D \hat{y}_D. \quad (22)$$

The macroblock sized area starting at the position $[\hat{x},\hat{y}]$ in $\mathbf{F}_{n-1}$ is taken to conceal the damaged macroblock in $\mathbf{F}_n$. To reduce the necessary number of operations, only parts of the neighbouring macroblocks can be used for the MV search. We chose the size of 10×8 and $\mathbf{A}_D$ two times as large as the searched block. We achieve better results by searching the motion vectors for such blocks that are smaller than the whole macroblock. For the search of MVs belonging to a subblock, blocks of the size smaller or greater than the subbblock itself may be used. The MVs of blocks belonging to the missing macroblock can be interpolated using the estimated MVs obtained for the subblocks of the neighbours, using Equation (19).

4. ADAPTIVE CONCEALMENT MECHANISM

All introduced methods we implemented into the Joint Model [7] and tested using various QCIF video sequences, each 400 frames long. We set a fixed number of 500 bytes per slice, I frame each 2 seconds, QP = 28, one reference frame, no B frames, no rate distortion optimization.

4.1 Performance of individual methods

At low resolutions, a single MB contains a lot of information. Therefore, temporal interpolation provides in most cases a better and simpler basis for error concealment than spatial domain interpolation. In Figure 2 screenshots of a part of an I frame concealed by all of the previously described methods can be seen. Spatial domain interpolation shows the worst

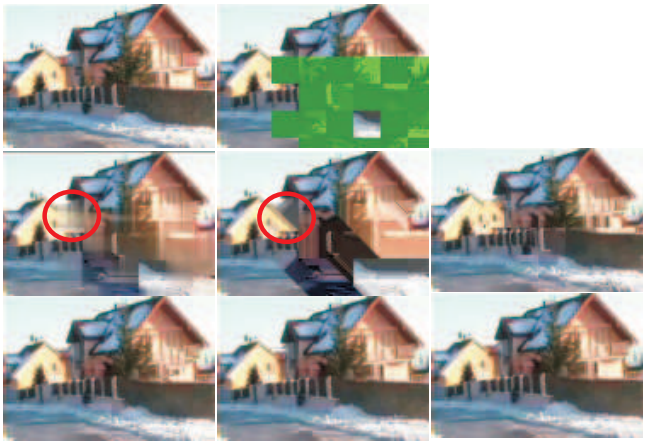

Figure 2: Screenshots of a part of an I frame in the "panorama" sequence (left to right): compressed original (Y-PSNR= 35.86dB), error pattern (Y-PSNR= 10.45dB), weighted averaging (Y-PSNR= 18.09dB), directional interpolation (Y-PSNR= 16.57dB), "copy-paste" (Y-PSNR= 22.76dB), boundary matching (Y-PSNR= 26.27dB), 8×8 block matching (Y-PSNR= 30.27dB), 2×2 block matching (Y-PSNR= 30.74dB).

performance. The directional interpolation method should not be used if only two neighbours are available [9]. The circles in the Figure 2 mark the position of the first missing macroblock being replaced by weighted averaging and directional interpolation. In this case directional interpolation nicely prolongs the edges, however not completely correct as the information from the bottom and right macroblocks is missing. Spatial error propagation even worsens the situation in later macroblocks. The "panorama" sequence had a frame rate decimated by four, resulting in higher motion. The "copy-paste" method does not perform well. It causes freezing of the concealed picture parts resulting in overall jerkiness of the video. Better perceptual results we obtain using boundary matching, but having a closer look, there are still some small artifacts left. The block matching algorithm leads to the best results. Perceptually 8×8 block matching performs already sufficiently.

4.2 Decision tree

The efficiency of a particular error concealment method depends strongly on the instantaneous spatial and temporal features of the concealed video sequence frame, in particular:

- presence of a scene cut,
- amount and type of movement,
- amount of spatial information,
- error size, shape and position,
- type of the frame and the information available.

For the decision we chose the presence of a scene cut or a fast change to decide between the temporal and spatial methods; number of neighbours and spatial character to decide between directional interpolation and weighted averaging; type of the frame to decide between the MV estimation method.

Following is the pseudo-code describing the process of the error concealment method selection:

```
if scene-change
   if clear-edges AND enough-neighbours
      method = directional-interpolation
   else
      method = weighted-averaging
else
   if I-frame
      method = block-matching
   else
      if MV-correct
         method = decode-without-residuals
      else
         method = MV-interpolation
```

In [8] it is proposed to detect the scene changes using the number of inserted I macroblocks. However, an H.264/AVC encoder can be set not to insert I macroblocks at all or to limit their number [7]. To make the method independent of the encoder settings we use a dynamic threshold technique according to [10]. In this method SAD is calculated for each frame: $\mathrm{SAD}(n) = \sum_{i=1}^{N}\sum_{j=1}^{M}|\mathbf{F}_n(i,j) - \mathbf{F}_{n-1}(i,j)|$, excluding the missing parts. The instantaneous value of threshold is calculated as a linear combination of the instantaneous SAD value, mean and variance of SAD calculated over the last 20 frames. This avoids detecting the scenes with higher amount of movement as scene cuts or fast scene changes (`scene-change`). The SAD calculation for P frames reduces to adding up the residuals. The edges (`clear-edges`) we detect as described in Section 2.2. For directional interpolation we require at least three correctly received neighbouring macroblocks (`enough-neighbours`). If one of the neighbouring blocks to the missing one in an P frame is an I macroblock, we also use MV interpolation error concealment. However, MV of the I macroblock we estimate before using the block matching principle as described in Section 3.3.

5. EXPERIMENTAL RESULTS

To evaluate the quality of reconstruction of a frame $\mathbf{F}$ we use the peak to signal-to-noise ratio of its YUV color space luminance component (Y-PSNR):

$$\text{Y-PSNR} = 10 \cdot \log_{10} \frac{255^2}{\text{Y-MSE}} [dB], \qquad (23)$$

Y-MSE being mean square error for luminance defined as

$$\text{Y-MSE} = \frac{1}{M \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{M} [\mathbf{F}(i,j) - \mathbf{F}_o(i,j)]^2, \qquad (24)$$

where $N \times M$ is the size of the frame and $\mathbf{F}_o$ is luminance in the original frame (uncompressed and not degraded). As a reference method we used the simple combination of weighted averaging for I frames and "copy-paste" for P and B frames (present in the JM) as it is widely used. However, the provided results (Y-PSNR) can be easily compared to any other of known methods, the most relevant of which seems to be [8]. In Table 1 the description of the sequences used for experiments can be seen. The error-free Y-PSNR is the Y-PSNR averaged over all frames of the particular sequence compressed and decoded without any errors. In Table 2 and 3 the Y-PSNR of the proposed method against the reference method is shown for the original and reduced frame rate (FR) respectively. Frame rate reduction leads to increase of the temporal activity and may result in using of different error concealment mechanisms. For these experiments we considered packet loss at IP layer with probability $p = 7\%$ resulting in the loss of the whole slice (no data partitioning). The average Y-PSNR is calculated over all sequence frames and averaged over 15 different packet loss realizations. To give a hint about the impact of an error, we also present the average size of an I and P slice in macroblocks. The biggest improvement we obtain for the "panorama" sequence, containing linear movement only. The improvement is even higher for the reduced frame rate, as the movement remains still linear and MV estimation can be performed appropriately. The improvement for other sequences is lower with reduced frame rate, because if there is a non-linear movement, the MV estimation is not sufficient to conceal if we do not know the residuals. The smaller the frame rate, the bigger the shift

Sequence name	Y-PSNR err.-free [dB]	Content description
foreman	35.24	dynamic talking head
akiyo	38.38	static talking head
mobile	32.98	zoom,pan,spatial details
panorama	35.14	village houses pan
soccer	34.14	soccer game (wide-angle)
squash	37.56	static cam, squash game
videoclip	35.95	diverse scenes

Table 1: Description of the test sequences.

Sequence name	Size of I/P slice [MB]	Ref. method Y-PSNR [dB]	Prop. method Y-PSNR [dB]
foreman	13/60	26.63	32.55
akiyo	17/99	31.51	37.46
mobile	5/20	22.89	29.90
panorama	11/53	25.69	32.68
soccer	14/43	27.18	31.42
squash	21/79	31.82	35.34
videoclip	11/38	25.25	29.17

Table 2: Results for various test sequences (PSNR averaged over 15 realizations of packet loss), $p = 7\%$, FR$= 30$ fps.

Sequence name	Size of I/P slice [MB]	Ref. method Y-PSNR [dB]	Prop. method Y-PSNR [dB]
foreman	13/38	26.09	30.40
akiyo	17/99	31.79	36.48
mobile	5/15	21.06	26.85
panorama	11/40	22.36	30.14
soccer	14/29	26.68	30.47
squash	21/50	30.21	34.56
videoclip	11/15	24.21	27.52

Table 3: Results for various test sequences (PSNR averaged over 15 realizations of packet loss), $p = 7\%$, FR decimated by four.

of the moving object from frame to frame. The ”squash” sequence shows good performance already for the reference method as only the players are shifted, the background is static. The size of players was small compared to the size of the frame (approximately 10% of the frame width and 40% of the frame height), but not as small as in the ”soccer” sequence, so still the proposed method was able to compensate the movement. Figure 3 shows Y-PSNR over the first 200 frames of the ”videoclip” sequence for the proposed and reference method. The sequence contains several scene changes (5 being cuts and 4 being transitions) leading to the use of the spatial error concealment in our method. The improvement

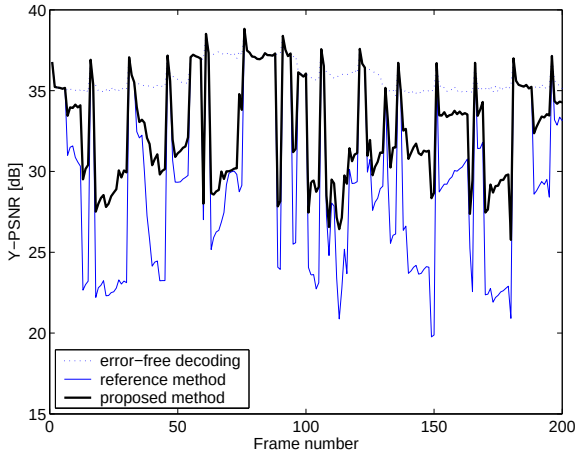

Figure 3: PSNR over the frame number for the first 200 frames of ”videoclip” sequence with $p = 7\%$, FR$=$ 30 fps.

that can be achieved in the case of a scene cut is limited by the performance of spatial error concealment, which does not perform well if the concealed area is large and the resolution small. Slow transitions are not recognized as scene changes due to our dynamic threshold causing the algorithm to use temporal concealment and achieving better performance in such cases. In Figure 4 Y-PSNR over packet loss for the ”soccer” sequence is shown. In this sequence, there is non-

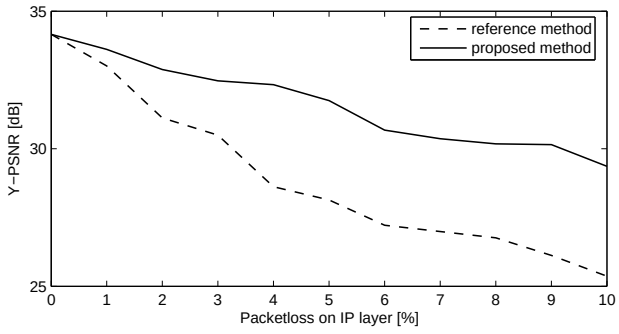

Figure 4: PSNR over the uniformly distributed packet loss probability p for the sequence ”soccer” with frame rate reduced by four.

linear movement (players appearance is changing from frame to frame). The improvement is rather high, compared for example to about 2% improvement of [8] achieved using similar assumptions. All individual methods only use simple operations that are performed also in an encoder and decoder (best match search, SAD, edge detection) and thus the complexity remains acceptable for streaming applications.

6. CONCLUSIONS

In this research we proposed an error concealment scheme adapting to the instantaneous sequence characteristics and the error pattern. The proposed method is based on a decision tree that takes into account scene change presence, type of the frame and error pattern. Unless there is no scene change, we use methods based on temporal interpolation as they perform considerably better than spatial interpolation, especially for low resolutions. We tested our method against the method used in today’s codecs (recommended for H.264/AVC). Performance of temporal error concealment is limited for fast moving videos. To investigate the impact of the speed of movement, we tested our method on seven different video sequences with original frame rate of 30 fps as well as for the frame rate decimated by four. The results show Y-PSNR improvement (compared to the typical simple methods) ranging from 3.5dB up to 7dB depending on the sequence, frame rate and packet loss probability. The error concealment algorithm also takes into account the perceptual video quality, preserving the smooth transitions between the blocks and prolonging of edges whenever possible.

REFERENCES

[1] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, ”Overview of the H.264/AVC Video Coding Standard”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, July 2003.

[2] M.T. Sun, A.R. Reibman, ”Compressed Video over Networks”, *Signal Processing and Communications Series*, Marcel Dekker Inc., New York, pp. 217-250, 2001.

[3] Y.H. Jung, Y. Kim, Y. Choe, ”Robust error concealment algorithm using iterative weighted boundary matching criterion,” *Proc. of IEEE Int. Conf. on Image Processing*, vol. 3, pp. 384-387, Sept. 2000.

[4] S. Belfiore, M. Grangetto, G. Olmo, ”An error concealment algorithm for streaming video,” *Proc. of IEEE Int. Conf. on Image Processing*, vol. 3, pp. 649-52, Sept. 2003.

[5] L. Su, Y. Zhang, W. Gao et. al., ”Improved Error Concealment Algorithms Based on H.264/AVC Non-normative Decoder,” *Proc. of IEEE Int. Conf. on Multimedia and Expo*, Jun 2004.

[6] G. Gennari, G.A. Mian, L. Celetto,” A H.264 Decoder Robust to Transmission Errors”, *Proc. of EURASIP EUSIPCO*, vol. 1, pp. 114-120, Vienna, Austria, Sep. 2004.

[7] H.264/AVC Software Coordination, ”Joint Model Software”, ver. 7.3, available in http://iphome.hhi.de/suehring/tml/.

[8] Y. Xu, Y. Zhou, ”H.264 Video Communication Based Refined Error Concealment Schemes,” *IEEE Trans. on Consumer Electronics*, vol. 50, no. 4, pp. 1135-1141, Nov. 2004.

[9] O. Nemethova, A. Al-Moghrabi, M. Rupp, ”Flexible Error Concealment for H.264 Based on Directional Interpolation,” *Proc. of IEEE WirelessCom*, Maui, Hawaii, June 2005.

[10] A. Dimou, O. Nemethova, M. Rupp, ”Scene Change Detection for H.264 Using Dynamic Threshold Techniques,” *Proc. of 5th EURASIP Conf. on Speech and Image Processing*, Smolenice, Slovakia, June 2005.