

# Reversible Motion Compensation for Wireless Video

Jiong Sun and Haibo Li

Digital Media Lab, Umeå University, Sweden

*Abstract*— How to stop error propagation due to packet loss is one of the hardest problems with wireless video. In this paper, we have proposed 1) an ideal client-server communication mode for wireless video. The center idea is to free the server from the feedbacks coming from clients. and 2) a new motion compensation scheme: reversible motion compensation. With this scheme the video frames can be decoded either from the beginning frame or from the ending one. This is the magic of this scheme to stop error propagation. It is this scheme that makes it possible to achieve the ideal client-server communication mode. Experimental results based on long test video sequences show that this new scheme is very effective to stop error propagation. In our test a 3-dB improvement is obtained. Another benefit with this new scheme is that a compressed video can be smoothly decoded in two directions, forward and backward, just like the functions, FFW and FBW provided by VCRs.

*Index Terms*— wireless video, error propagation, motion compensation.

## I. INTRODUCTION

With the development of wireless mobile devices such as Smart phones, Multimedia phones, Pocket PCs, wireless video becomes one of the most attractive functions of such devices. A typical application scenario is shown in Fig.1 where the mobile client can download video clips from the remote server and display it on his mobile terminal. In this type of application the video server sends packets of a video clip to the base station, which are then forwarded to the mobile client. At the client's end, there is some kind of applications that can be used to display the received video clip.

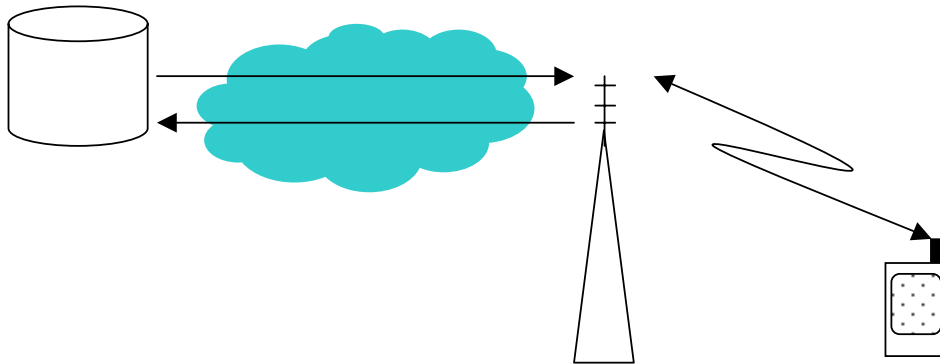


Fig.1 A typical application scenario of wireless video

Although wireless video services are highly desirable, however, it is very challenging to deliver a video to mobile clients. It is simply due to the fact that the mobile channels have very limited capacity and are very noisy. To transmit a video sequence through wireless channels, very high compression is necessarily needed to apply for. Today the almost all video compression standards are based on hybrid coding scheme as shown in Fig.2. and Fig.3. The reason why it is popular is mainly due to the fact that it removes spatial and temporal redundancy in a very efficient way and it provides short coding delay. Although high compression performance is achieved from its closed-loop structure,

from which the problem is generated also! If one carefully study the coding structure, it is not hard to find out that a decoder is embedded in the encoder. This tells us that an identical decoder has to be running in the encoder. There will be a disaster if the decoder loses synchronization with the encoder which happens very often.

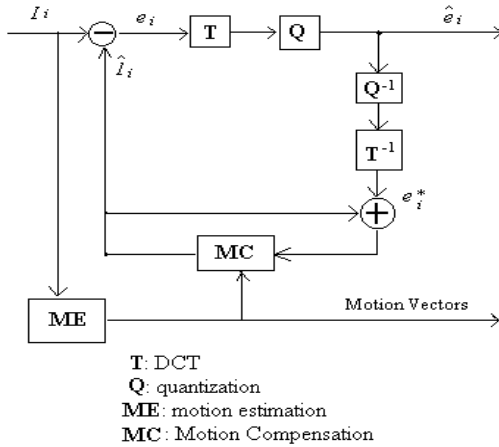


Fig.2. Hybrid Coding Scheme: Encoder

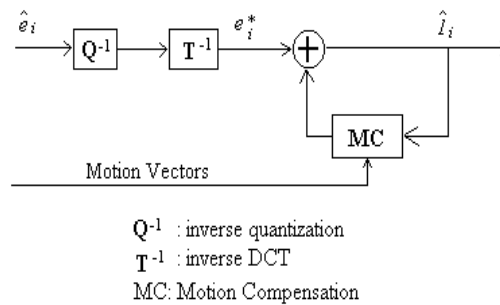


Fig.3. Hybrid Coding Scheme: Decoder

Now let us see what will happen if there is package loss. Assume for a typical low bit rate video, a good strategy to pack compressed bit stream is “one picture, one packet”[1]. When a package is lost, that is, a frame is lost, to be able to continue decoding following frames, some kind of error-concealing techniques have to be used to recover the lost frame. The popular way is to use the previous frame to substitute the lost one. Since the successive frames look very similar, the frame difference is not visually noticeable. Unfortunately, the slight frame difference will lead to error propagation, that is, when an error occurs in one frame, it propagates rapidly in the following frames. This is illustrated in Fig.4. The error propagation is naturally because of the recursive structure of hybrid coding scheme. Therefore, from its inherent structure, the hybrid coding scheme is not suitable for wireless video purpose!

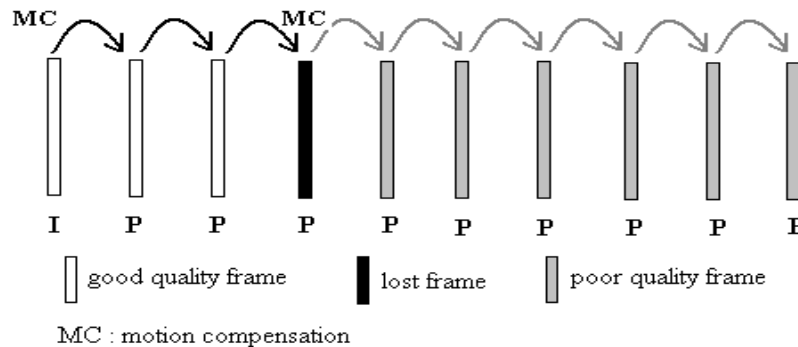


Fig.4. Packet loss leads to error propagation.

Another solution to handle the data loss is to employ reliable transportation protocol: TCP/IP or UDP with feedback. With such protocols, a request for re-sending is sent from a client to the server when the client detects a lost package. Without regarding time delay a virtual error-free channel can be assumed at the application level for the video coding. The major drawback with this way to handle error is that 1) there will be considerable time delay for a client to get a packet; 2) the server will be busy coping with feedback requests from clients; 3) wireless channel will cause a lot of packet loss, which dramatically increases Internet traffic due to the re-sending requests.

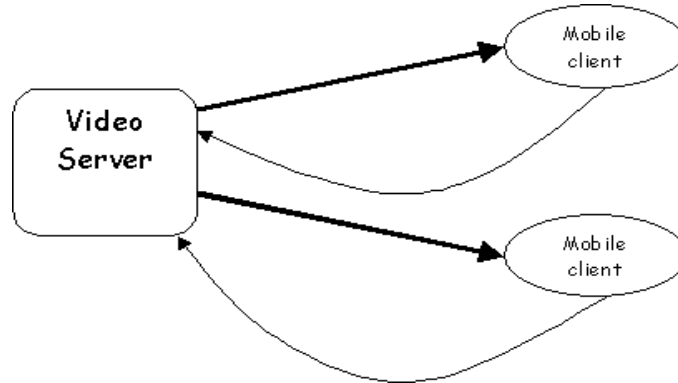


Fig.5 Video server gets heavy load due to feedbacks from clients

One solution to this problem is indirect TCP, proposed by Bakne and Badrinath[2]. It split the TCP connection into two parts. The first connection goes from the video server to the base station, while the base station simply copies packets between the connections in both directions.

In this paper, we propose an ideal client-server communication mode for wireless video that is shown in Fig.6. The basic idea is to free the server from the feedbacks coming from clients. The philosophy is simple: the server is doing its best to send out all information; as for its clients, how much information can be obtained or how good the quality of the reconstructed video can be is only up to the receiving and processing strategies adopted by individual clients.

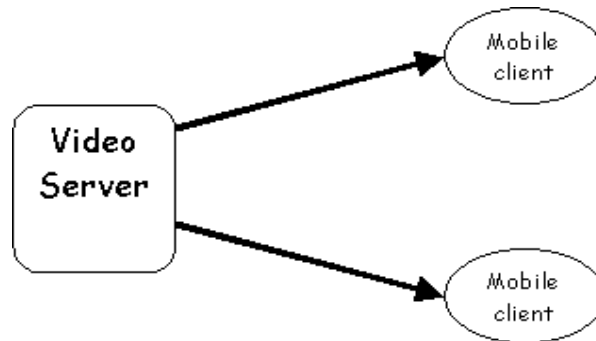


Fig. 6 Video server doesn't get any feedbacks from clients.

Since there is no way to prevent packet loss, a client has to find a smart way to limit the error propagation. From the analysis above we learn that it is very difficult to use the hybrid coding scheme to achieve this mode. In this paper we present a new coding scheme, with which the effect caused by the packet loss can be reduced to minimum. The idea is to invent a bi-directionally decodable video, just as what is shown in Fig.7. If there is a frame is lost the following frames can backwardly decoded from the end frame. In this way the error due to packet loss will not be able to propagate! We will have a detailed discussion on how to implement such a new coding scheme.

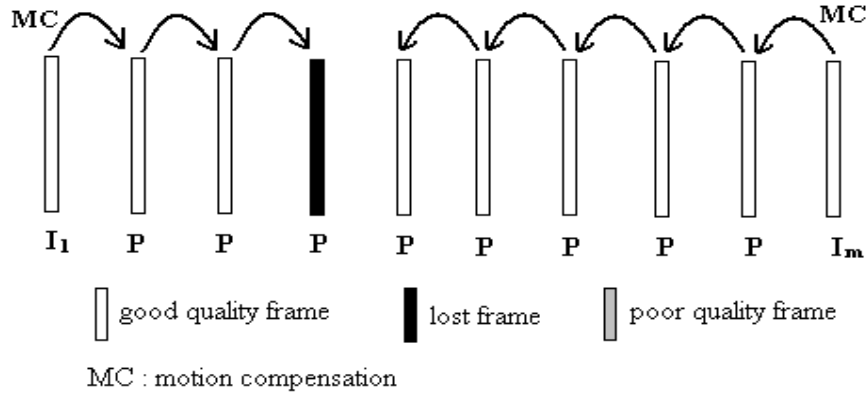


Fig.7. Reversible MC scheme

Please note that B-frame, known in MPEG and H26x, can also be used for this purpose but the performance of using B frames is not good simply due to that the temporal redundancy between successive frames is not exploited!

## II. A NEW MOTION COMPENSATED CODING SCHEME

### A. How to do motion compensation?

Before we introduce our new motion compensation scheme, let us take a look at how traditional motion compensation works. It aligns the previous frame  $I_{i-1}$  to the input frame  $I_i$  as  $I_i^*$ ,

$$I_i^* = MC(I_{i-1}) \tag{1}$$

Where  $MC(.)$  is the motion compensation. Fig.8 shows the frame alignment of traditional motion compensation.

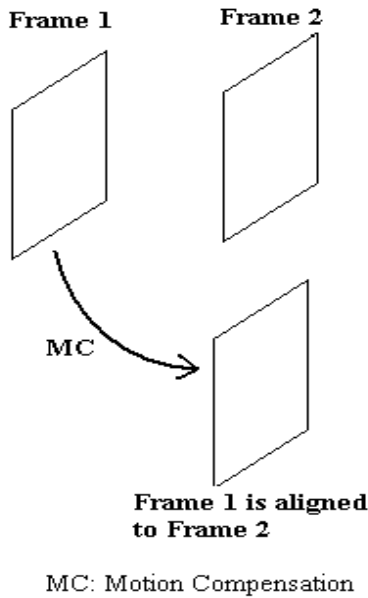


Fig.8. Frame alignment in traditional MC

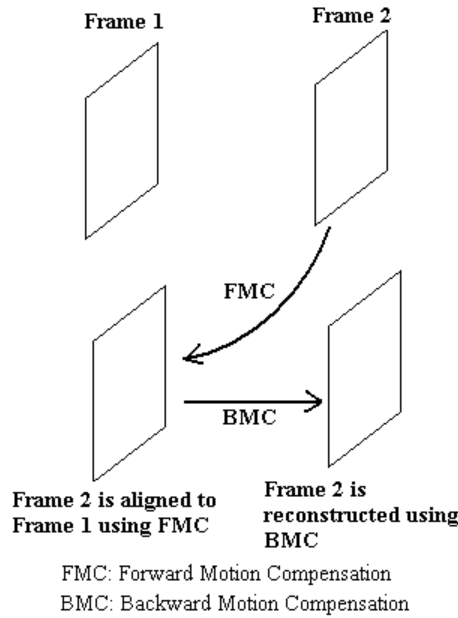


Fig.9. Frame alignment imitating human eyes

There's another way to implement motion compensation, that is, according to human eye's function. If we take a look at the human eye movement, we will see that human eye always tries to minimize the difference between the incoming image and the existing image in order to achieve visual perception. Eye movements try to align the input visual information to the received one, which means that eye movements perform motion compensation on the input image

while motion compensation is performed on the previous image in the conventional image coding concept. Fig.9 shows the frame alignment used in motion compensation imitating human eye movement.

Let  $\hat{I}_{i-1}$  stand for the previous frame and  $I_i$  for the input one. To imitate the eye movement function, the input frame  $I_i$  is aligned to the previous frame  $\hat{I}_{i-1}$ :

$$I_{i-1}^* = F(I_i) \quad (2)$$

where  $F$  is the forward motion compensation function.

The residual signal  $e_{i-1}$  between them is sent to the receiver:

$$e_{i-1} = I_{i-1}^* - \hat{I}_{i-1} \quad (3)$$

At the receiver end, the input frame can be reconstructed by a forward motion compensation function  $F$ :

$$\hat{I}_i = B(I_{i-1}^*) = B(\hat{I}_{i-1} + e_{i-1}) \quad (4)$$

In order to recover the input frame  $I_i$  the forward motion compensation must satisfy

$$(FoB)(I_i) = I_i \quad (5)$$

### B. The proposed motion compensated coding scheme

Fig.10 and Fig.11 show the encoder and the decoder.

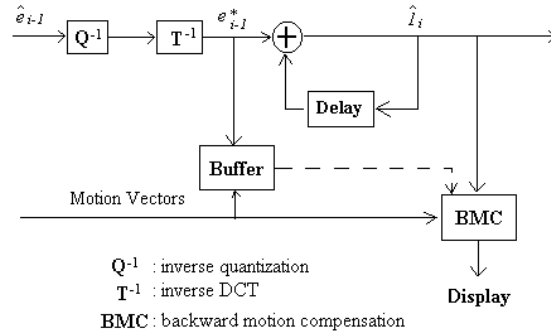
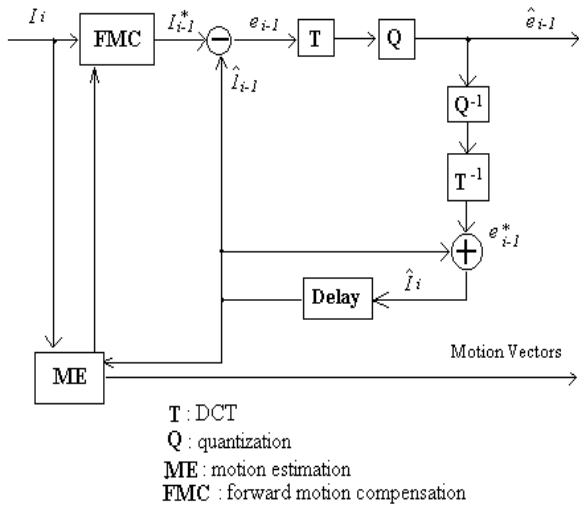


FIG.10. New motion compensated coding: Encoder FIG.11. NEW MOTION COMPENSATED CODING: DECODER

An important observation is that reversible motion compensation pairs, forward and backward ones, are used in the encoder and decoder, respectively. In the encoder, the first frame  $I_1$  is regarded as the I (Initial) frame. This frame is compressed without motion compensation. The second frame  $I_2$  will be aligned to the previous frame  $I_1$  using a reversible motion compensation operation described in next section. The aligned frame is denoted as  $I_1^*$ . Then we have the difference  $e_1 = I_1^* - I_1$ , whose compressed version  $e_1^*$  will be sent to the decoder. With  $e_1^*$  we can get  $\hat{I}_2 = I_1 + e_1^*$ , which will be regarded as the reconstructed version of the 'previous' frame ' $I_2$ ' in the next loop of

encoding when the third frame  $I_3$  comes. Repeat this process, we have  $\widehat{I}_3 = \widehat{I}_2 + e_2^* = I_1 + e_1^* + e_2^*$ . So that for the  $n$ th frame, we have

$$\widehat{I}_n = I_1 + \sum_{i=1}^{n-1} e_i^* \quad n > 1 \quad (6)$$

One important thing is: in the end of a video segment, or we can call a group of picture (GOP), we send  $\sum_{i=1}^{n-1} e_i^*$  instead of one single  $e_i^*$ . We will see the reason why doing this in the following.

In the decoder, we can use

$$\widehat{I}_i = \widehat{I}_{i-1} + e_{i-1}^* \quad i > 2 \quad (7)$$

to reconstruct frame  $\widehat{I}_i$ . Only a backward motion compensation is needed. Then we get  $I_i'$ . There is a buffer in the decoder used for storing motion vectors and  $e_i^*$ . From (7), we have

$$\widehat{I}_{i-1} = \widehat{I}_i - e_{i-1}^* \quad i > 2 \quad (8)$$

which means that we can reconstruct frame  $\widehat{I}_{i-1}$  from frame  $\widehat{I}_i$ . After a backward motion compensation, we get  $I_{i-1}'$ .

### C. Reversible motion compensation

In the new motion compensation scheme, both forward and backward motion compensation operations are used. As pointed out before, to be able to recover the original frame, the forward and backward motion compensation should be reversible, as in (5).

Such a motion compensation can be achieved with a triangular mesh [4]. Fig. 12, illustrates how such a motion compensation scheme works

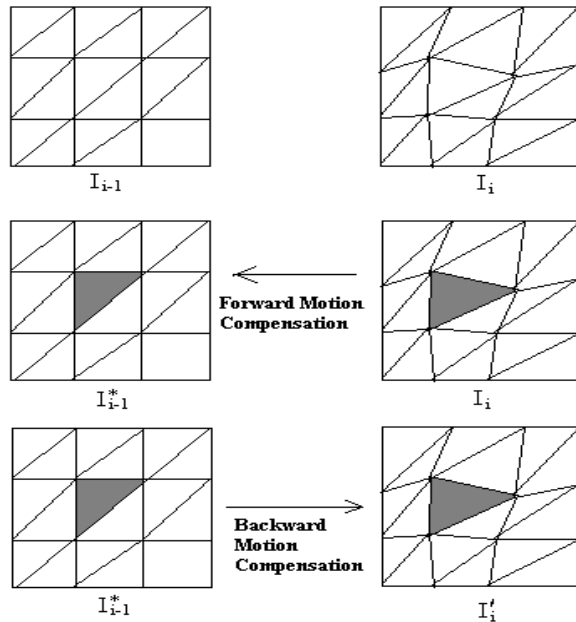


Fig. 12. Motion alignment compensation scheme

Steps to do a reversible motion compensation: First divide the previous image into regular triangle meshes. Second, align the input image  $I_i$  to the previous image  $I_{i-1}$ , that is, find the corresponding triangle mesh in the input image. Third, use the geometric transform to map the input image into its aligned version  $I_{i-1}^*$ . Finally, use an inverse geometric transform to retrieve the input image from  $I_{i-1}^*$ , get  $I_i'$ .

Fig. 13 shows some experiment results of two images using the triangle-based motion compensation scheme. More detail information about this implementation can be found in [4].

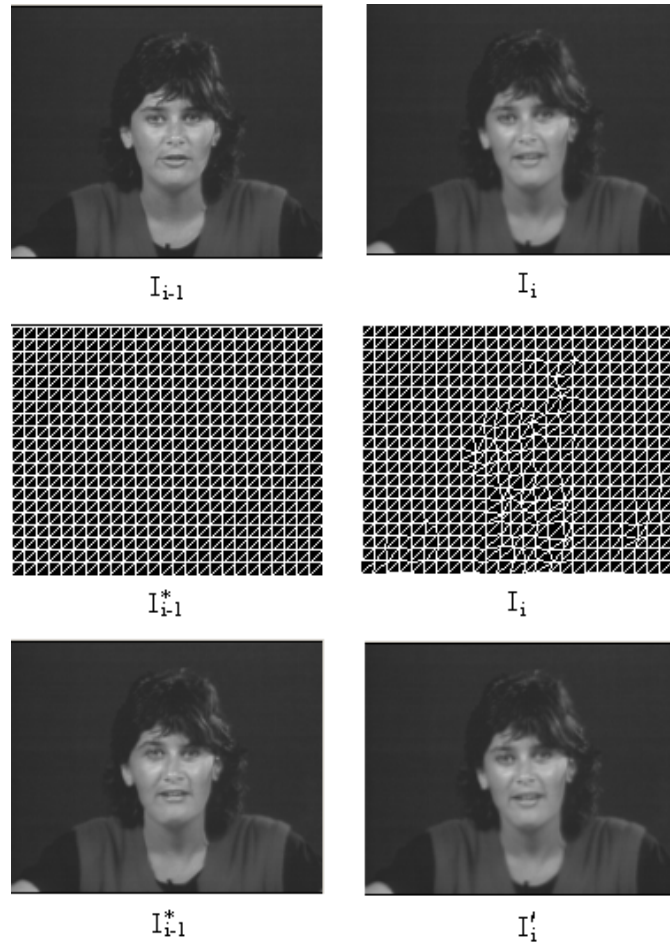


Fig.13. Experiment results of two images using triangle-based motion compensation scheme.

#### D. Displacement Estimation of a triangle mesh

In this section we briefly introduce how to compute the deformation of a triangular mesh between two successive frames. The displacement of vertices of a triangle can be used to represent the optical flow field of the triangle. When we are applying a triangle mesh, we should consider the common vertices shared by adjacent triangles. As in [4], if we take the triangle mesh ABCDE in Fig. 14 into account, we will have

$$\begin{bmatrix} u_{11} \\ u_{12} \\ \cdot \\ u_{5m} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & 0 & 0 \\ p_{21} & p_{22} & p_{23} & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ p_{m1} & p_{m2} & 0 & 0 & p_{m3} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \\ u_d \\ u_e \end{bmatrix} \quad (9)$$

where  $u_{ij}$  is the optical flow of a point inside the triangle mesh. This formula tells us that if we know the displacements of vertices, the optical flow of each point within the triangle can be computed. Now we are concerned that if we know the optical flow field of a triangle how can we compute the displacement of the vertices. This can be done with a Least-squared method.

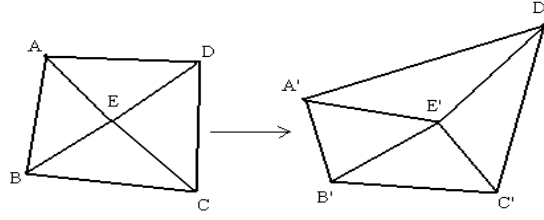


Fig.14. Moving triangle mesh

Let  $L$  denote the middle matrix in (9),  $u_i$  be the left vector, we have:

$$u_i = Lu_a \quad (10)$$

Further more we have:

$$\begin{aligned} u_a &= (L^T L)^{-1} L^T u_i \\ &= Gb \end{aligned} \quad (11)$$

where  $G = (L^T L)^{-1}$  and  $b = L^T u_i$

Formula (11) shows how to recover the displacement of vertices from optical flow field. Please note that  $G$  can be pre-computed and stored! Therefore, the complexity of this method is very low.

#### E. Implementation

We have used OpenGL[9] to implement the texture mapping part.

### III. SIMULATION AND RESULTS

In our simulation we made the following assumptions: the video frames are divided into several segments. Each segment has 30 frames. In each segment, the first frame and the last frame are coded as I frames, the others are P frames. Only one P frame will be lost. The test video sequence is 'Miss America', *miss\_am.qcif*.

The following experiments are carried out:

#### 1) No Packet Loss:

We measure the PSNR of the new scheme and the conventional block-based motion compensation scheme with different quantization steps. Fig.15 and Fig.16 show the results. From the figure we can see that the conventional scheme outperforms the new scheme a bit. It is not surprising since we send two I frames to increase the error resilience. The loss in performance is the price we paid for robustness.

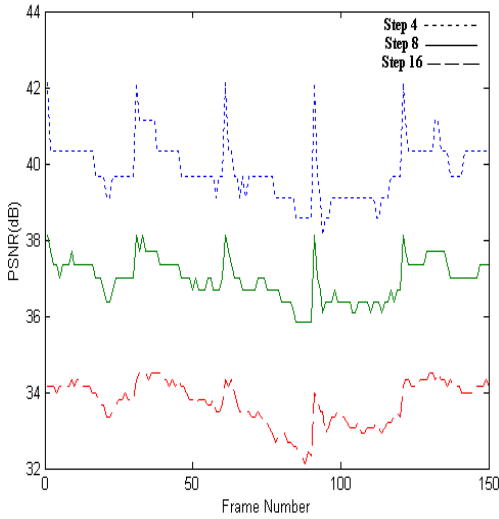


Fig.15

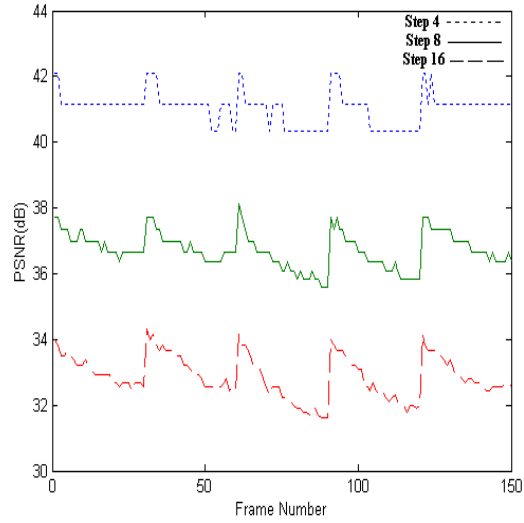


Fig.16

Fig.15. PSNR of the reconstructed frame when quantization steps are 4, 8, 16, respectively, using the new coding scheme. Fig. 16. PSNR of the reconstructed frame when quantization steps are 4, 8, 16, respectively, using the conventional block-based motion compensation method. Block size is  $8 \times 8$ .

2) *Packet Loss:*

Let's assume one frame in a segment is lost. If the previous frame is used to substitute the lost frame, the PSNR will drop rapidly due to error propagation when the hybrid coding scheme is employed.

By using the new coding scheme we can get much better performance. For the lost frame, we use the previous frame to substitute. For the rest of the frames in this segment, we can reconstruct them from the last frame in this segment.

For example, if the 6<sup>th</sup> frame is lost. In the decoder, we have the first frame  $I_1$ ,  $\sum_{i=1}^{29} e_i^*$ ,  $e_i^*$  ( $i=1,2,3,4,5,7,\dots,28$ ) and motion vectors for each frame except the 6<sup>th</sup>. According to (6), we can recover  $\hat{I}_i$  ( $i=29,28,\dots,7$ ) in a backwardly recursive way .

Fig.17 and Fig.18 show performance comparisons of two motion compensation schemes when a frame in a segment is lost.

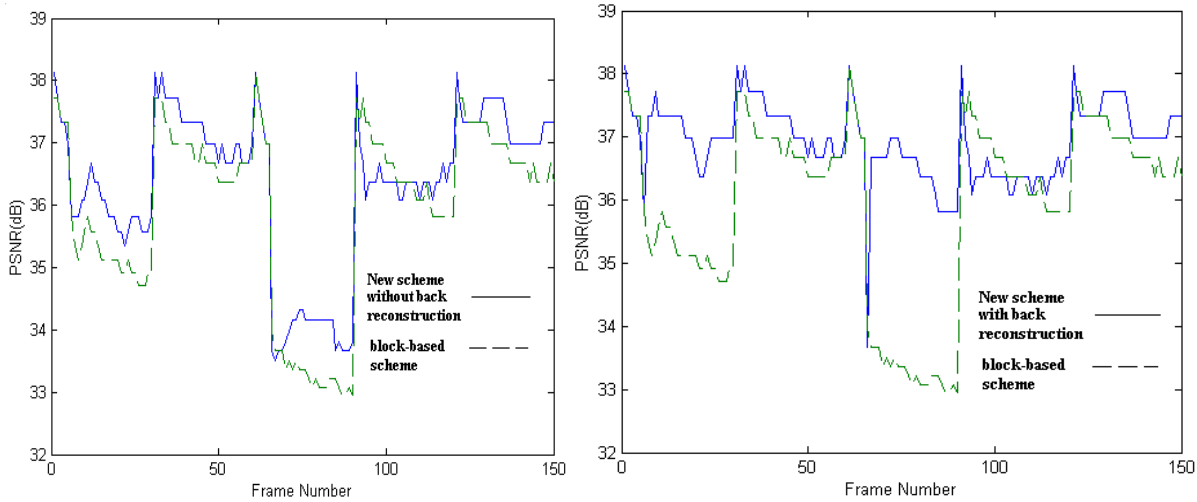


Fig.17 The PSNR of the reconstructed frames with quantization step is 8 when the 6<sup>th</sup> and the 66<sup>th</sup> frames are lost if we use the previous frame to substitute the lost one in the new scheme and the conventional block-based motion compensation scheme. The block size is 8×8.

Fig. 18 The PSNR of the reconstructed frames with quantization step is 8 when the 6<sup>th</sup> and the 66<sup>th</sup> frames are lost using the new scheme with using the conventional block-based motion compensation scheme. The block size is 8×8.

Important observations from Fig.17 and 18 is that 1) error concealing technique is not a good way to stop error propagation caused by packet loss, 2) reversible motion compensation is a very effective way to stop error propagation!

#### IV. CONCLUSION

In this paper, we have proposed 1) an ideal client-server communication mode for wireless video. The center idea is to free the server from the feedbacks coming from clients. and 2) a new motion compensation scheme: reversible motion compensation, which is the key to achieve the ideal client-server communication mode. Experimental results based on long test video sequences show that this new scheme is very effective to stop error propagation. In our test a 3-dB improvement is obtained. Another benefit with this new scheme, which is not addressed in this paper is that a compressed video can be smoothly decoded in two directions, forward and backward, just like the functions, FFW and FBW provided by VCRs.

#### REFERENCES

- [1] S. Wenger, and G. Côté, "Using RFC2429 and H.263+ at low to medium bit-rates for low-latency applications" Packet Video Workshop '99, New York, Apr.1999
- [2] A. S. Tanenbaum, "Computer Networks, third edition", Prentice Hall, Inc., 1996
- [3] K.Kamikura, O.Kawi, and K.Matsuda, "Principal devices and hardware volume estimation for moving picture decoder for digital storage media," SPIE/VCIP, Lausanne, Switzerland, vol. 1360, pp.1551-1559, Oct. 1990.
- [4] Haibo Li, "Low Bitrate Image Sequence Coding," Linköping Studies in Science and Technology. Dissertations. No. 318, pp. 185-209, Linköping, Sweden, 1993.
- [5] A. M. Tekalp, "Digital Video Processing," Prentice Hall, 1995.
- [6] L. Torres, M. Kunt, "Video Coding: the Second Generation Approach", pp. 403-408, 1996
- [7] S. Baker, I Matthews, "Equivalence and Efficiency of Image Alignment Algorithms", in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2001,
- [8] J. J. Hwang, K.R.Rao, "Techniques and Standards for Image, Video, and Audio Coding", Prentice Hall PTR, NJ, USA, 1996
- [9] M.Segal, K.Akeley, "The OpenGL Graphics System: A Specification (Version 1.3)