Joint Tracking and Multiview Video Compression

Cha Zhang and Dinei Florêncio
Communication and Collaborations Systems Group
Microsoft Research, Redmond, WA, USA 98052
{chazhang,dinei}@microsoft.com

ABSTRACT

In immersive communication applications, knowing the user's viewing position can help improve the efficiency of multiview compression and streaming significantly, since often only a subset of the views are needed to synthesize the desired view(s). However, uncertainty regarding the viewer location can have negative impacts on the rendering quality. In this paper, we propose an algorithm to improve the robustness of view-dependent compression schemes by jointly performing user tracking and compression. A face tracker tracks the user's head location and sends the probability distribution of the face locations as one or many particles. The server then applies motion model to the particles and compresses the multiview video accordingly in order to improve the expected rendering quality of the viewer. Experimental results show significantly improved robustness against tracking errors.

Keywords: joint tracking and compression, condensation, view-dependent compression

1. INTRODUCTION

Advances in camera, display and networking technology have enabled a new set of applications for three dimensional (3D) scene communication, among which are 3D TV/free viewpoint TV (FTV) [1], tele-immersive environment [2], immersive teleconferencing [3], etc. In these applications, multiple video cameras are often used to simultaneously acquire the visual scene from different viewpoints. These videos are then transmitted to the remote end for rendering, providing the user an immersive experience.

Due to the high raw data rate of multiview video, multiview video compression is regarded as an essential piece of technology to enable 3D communication, and has attracted a lot of attention recently [4]. In particular, researchers have extensively studied predictive coding for multiview video compression, since there is apparently huge redundancy across the videos from different viewpoints. Another source of redundancy comes from viewpoint selection in interactive viewing scenarios, e.g., immersive tele-conferencing [6]. Figure 1 shows a two-party conferencing session between site A and B. At site A a set of cameras is used to capture the scene. The videos are compressed and sent to site B, which are then decompressed and rendered to the user in an interactive fashion. Depending on the type of display that is available at site B, one or a few virtual views need to be synthesized based on the viewer's head position at site B. To support such interactivity, the videos must be compressed and sent to site B with little delay. In addition, to support the motion parallax effects [6] at site B, it is imperative that the view be generated at the receiver side.

The above immersive tele-conferencing scenario naturally leads to compression and streaming schemes based on predicted viewer positions [7][8]. The idea is to have the user at site B send his/her current viewpoint/look direction and motion information to site A. Site A then compresses the multiview video accordingly to save bandwidth. In [7], all the views were first compressed with base layer multiview coding. The enhancement layers of the two views closest to the remote user's position were then added to the stream to improve their visual quality. In our previous work in [8], we let Site A render the virtual view (one or multiple) and generate weight maps for all the views. The weight maps indicate the quality requirement of each pixel or macroblock. If the weight is high, the pixel is used for rendering and needs to be encoded at high quality. Otherwise, the pixel can be heavily compressed. Site A then adaptively encode the multiview video based on these weight maps, and stream the video to site B for regular decompression and rendering. It was shown in [8] that if the prediction of the user's head position at site B is accurate, we can easily achieve more than 50% savings in bit rate at the same rendering quality.

In practice, however, it is a difficult task to predict future head positions accurately. The user's head movement is non-stationary, and prediction based on the tracking history can be error-prone. For example, in the interactive light field

streaming work by Ramanathan et al. [9], it is shown that view prediction can help improve performance, but this improvement is limited by the prediction accuracy. This coincide with our observation in [8], where we notice a 25 cm error in predicted viewer position can cause over 2dB drop in video reconstruction quality.

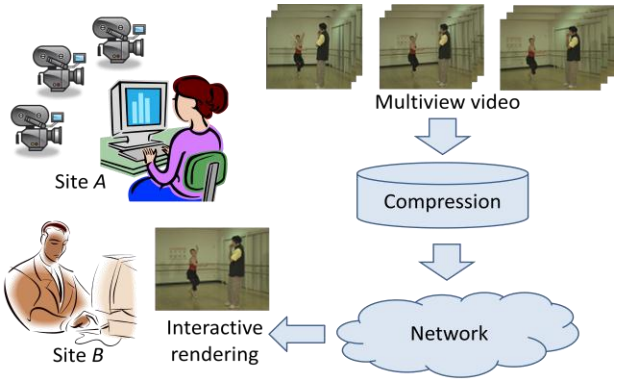

Figure 1 Immersive tele-conferencing scenario. The system could be symmetrical, i.e. site B could also sent its multiview video to site A.

In this paper, we propose a joint tracking and compression scheme to overcome the above issues. We take a statistical view for the head tracking problem, and adopt a condensation framework [10] for our problem. At any instance, the position and speed of the head is modeled with a non-parametric probability distribution. Upon receiving such a distribution at the sender (site A), it is first propagated according to the motion model, and a number of particles are sampled to represent the possible head positions in the future. The sender then synthesizes the virtual views at the sampled head positions and generates a set of weight maps. These weight maps are combined to improve the expected rendering quality at the receiver (site B). Due to the use of particles, the proposed scheme is more robust than predicting a single future head position for rendering, as shown in our experiments.

The rest of the paper is organized as follows. Condensation based tracking is briefly reviewed in Section 2. Joint tracking and compression is described in Section 3. Experimental results and conclusions are given in Section 4 and 5, respectively.

2. CONDENSATION BASED TRACKING

In this section we briefly review the condensation based framework for head position tracking. Let $\boldsymbol{x}_t$ be the status of the tracked object at time instance t; and let $\boldsymbol{Z}_t = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t\}$ represent the input video stream for tracking. The a posteriori probability of the object status at time t can be written as [10]:

$$p(\boldsymbol{x}_t|\boldsymbol{Z}_t) \propto p(\boldsymbol{z}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$$

where

$$p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1}) = \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$$

Here we assume that the object dynamics form a temporal Markov chain. Note in order to compute the a posteriori probability $p(\boldsymbol{x}_t|\boldsymbol{Z}_t)$, one needs to have the motion model $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, and the observation model $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$. In addition, the previous frame's a posteriori probability distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ must be kept during tracking.

In condensation based tracking, the previous a posteriori probability distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ is kept as a set of particles, each having a weight associated with it. The algorithm first resamples $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ as a new set of particles, which then go through three steps: drift, diffuse and measure. In the drift step, the particles deterministically move to a predicted new position based on the motion model. In the diffusion step, the particles go through Brownian motion in order to best represent $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$. In the measure step, the likelihood $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ are computed for each particle as weights of the resampled particles, and the resultant weighted particle set represents $p(\boldsymbol{x}_t|\boldsymbol{Z}_t)$.

Condensation based tracking is well known for its robustness against tracking errors. The a posteriori probability $p(\boldsymbol{x}_t|\boldsymbol{Z}_t)$ is maintained non-parametrically, and it can be used to keep multiple hypotheses of the object status in order to recover after the tracked object is temporarily occluded or lost. On the other hand, for immersive tele-conferencing applications, it is rare that the user's head position is occluded, and feature-based trackers such as that in [6] often outperforms typical condensation based trackers. Nevertheless, the framework of condensation can still be very useful in computing $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$, which is the prediction of future object status based on the previous observations.

Consider the extreme case that only a single particle is kept for $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$, e.g., $\boldsymbol{x}_{t-1}^0$. The particle $\boldsymbol{x}_{t-1}^0$ may simply be the tracking result of a feature-based tracker. Assume there is some uncertainty with the tracking result, hence $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ follows a Gaussian distribution:

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1}) = N(\boldsymbol{x}_{t-1}^0, \boldsymbol{\Sigma}_{t-1}^0)$$

where $N(\boldsymbol{m}, \boldsymbol{\Sigma})$ stands for Gaussian distribution with mean $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$. We also represent the motion model with a Gaussian distribution:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = N(f(\boldsymbol{x}_{t-1}), \boldsymbol{\Sigma}_f)$$

where $f(\boldsymbol{x}_{t-1})$ is a deterministic function to predict $\boldsymbol{x}_t$, and $\boldsymbol{\Sigma}_f$ is the prediction variance. If $f(\boldsymbol{x}_{t-1})$ is linear, the prediction distribution $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$ will also follow a Gaussian distribution. In practice, even if $f(\boldsymbol{x}_{t-1})$ is non-linear, we can still represent $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$ with particles. First, we resample $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ into *K* particles following the Gaussian distribution $N(\boldsymbol{x}_{t-1}^0, \boldsymbol{\Sigma}_{t-1}^0)$. Each particle then drifts according to the deterministic function $f(\boldsymbol{x}_{t-1})$, followed by a diffuse step that perturbs the prediction based on $N(f(\boldsymbol{x}_{t-1}), \boldsymbol{\Sigma}_f)$. We will use such a set of particles to perform joint tracking and multiview video compression, as explained in the next section.

3. JOINT TRACKING AND MULTIVIEW VIDEO COMPRESSION

Following the predicted viewer position based multiview video compression framework in [8], as shown in Figure 1, the user at site B first sends his/her current viewpoint/look direction and motion information to site A. Site A then renders the view and generates weight maps for all the video frames. These weight maps guide the video encoder to adaptively vary the quantization steps of the macroblocks in order to achieve better performance.

In [8], we assume there is an oracle that helps site A to perfectly predict the future position of the user at site B. This assumption is, certainly, unrealistic. In this paper, we explore a scheme that performs joint tracking and multiview video compression with the condensation framework. Namely, we let site B send its tracking status $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ to site A. Site A then computes $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$, which is used to control the encoder in order to achieve the best performance.

More specifically, as described in Section 2, at site A, we first resample $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ into *K* particles. These particles then drift and diffuse to represent $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$. Note since the latest observation $\boldsymbol{z}_t$ is not available at site A, $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$ is indeed the best prediction of the future status $\boldsymbol{x}_t$ at site A. We then render the virtual views at locations corresponding to the particles, and obtain *K* weight maps for each camera. These weight maps are averaged to obtain an average weight map for each camera, which is then used to control the quantization parameter QP for the macroblocks. The flowchart of the above process is summarized in Figure 2.

3.1 Weight map generation

Multiview video rendering belongs to the broad research field of image-based rendering [11], and has been studied extensively in the literature. In this paper, we limit ourselves to a particular form of multiview video – multi-stereoscopic video with depth maps. We assume we are given a number of video sequences captured from different viewpoints. Meanwhile, a single depth map is available to facilitate the virtual view rendering. An example data set is shown in Figure 3 [12]. Nevertheless, the joint tracking and multiview compression scheme proposed in this paper can be applicable to other multiview video formats.

We follow the same rendering algorithm in [8] to generate the *K* weight maps for each camera view. Briefly, for each particle representing $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$, we compute its corresponding virtual viewpoint based on the calibration parameters of the tracking camera. The to-be-rendered view is then split into light rays. For each light ray, we trace the light ray to the surface of the depth map, obtain the intersection, and reproject the intersection into nearby cameras. The intensity of the

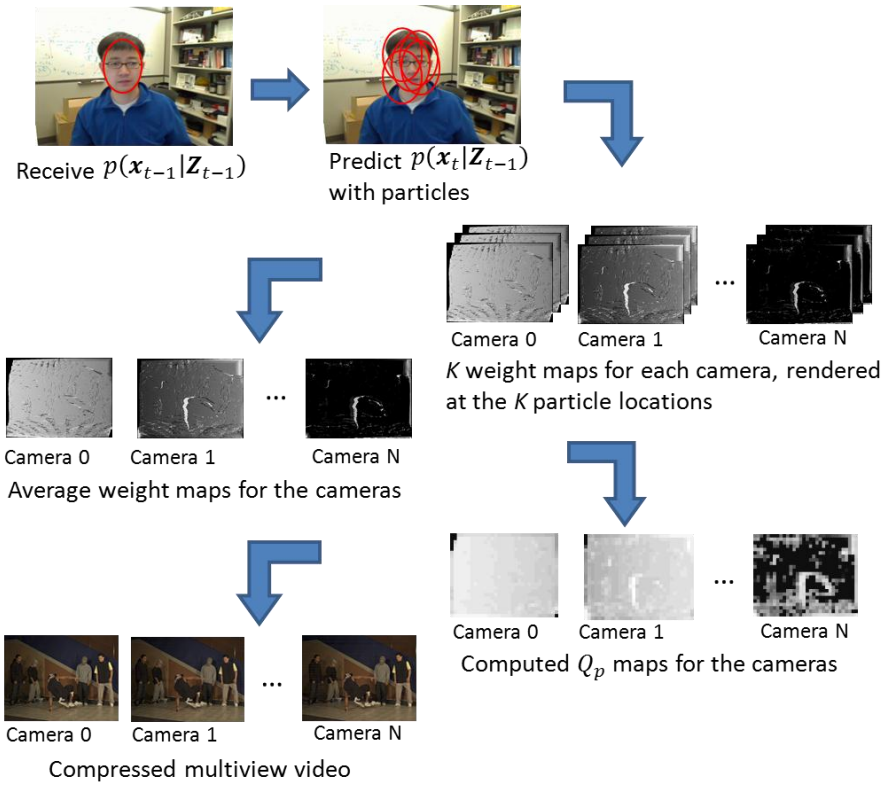

Figure 2 Flowchart of joint tracking and multiview compression.

light ray is thus the weighted average of the projected light rays in the nearby cameras. The weight can be determined by many factors [13]. In the simplest form, we can use the angular difference between the light ray to-be-rendered and the light ray being projected, assuming the capturing cameras are at roughly the same distance to the scene objects. Occlusions in the rendered view are explicitly considering during this process. That is, if a vertex is occluded in a nearby view, its weight for that camera will be set to zero.

The above process generates K weight maps for each camera rendered at the K particle locations, i.e., $W_i^1, \ldots, W_i^k, \ldots, W_i^K$, where i is the camera index, and $k = 1, \ldots, K$ is the particle index. These K weight maps are then averaged to obtain a single weight map for each camera. In order to improve the robustness of the system with respect to the motion model errors, we take the maximum value of weights for each pixel as the final weight map:

$$\bar{w}_{ij} = \max_k w_{ij}^k,$$

where w_{ij}^k is the value of the j^{th} pixel in weight map W_i^k.

3.2 Adaptive multiview video compression

We compress each of the camera views using a H.264 based codec and the weight maps described above. More specifically, the weights are L2 averaged within each 16x16 macroblock, and the quantization step for that macroblock is obtained by computing [8][1]:

$$QP_{im} = BaseQP - 6\log_2 \sqrt{\frac{1}{256}\sum_{j \in MB_m} \bar{w}_{ij}^2},$$

[1] There was a typo in [8]. The sign in the middle should be "-" instead of "+".

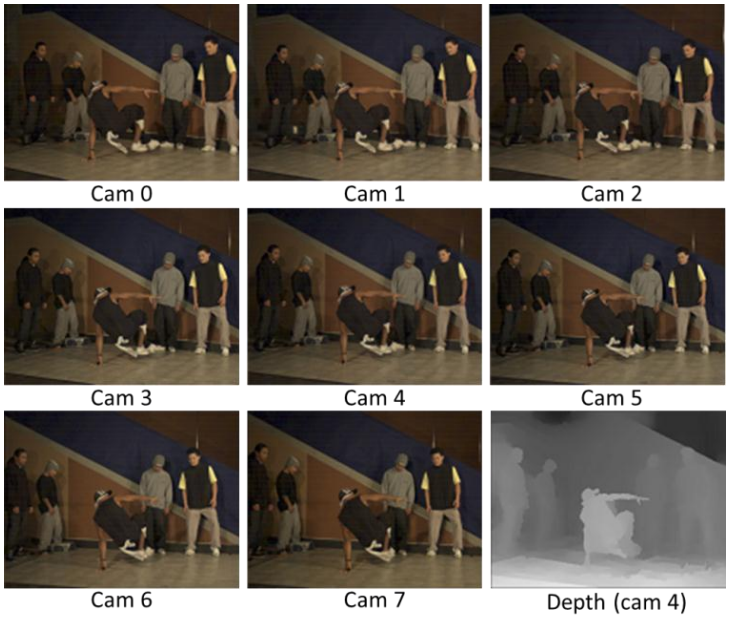

Figure 3 The *breakdancer* multiview data set [12].

where MB_m is the m^{th} macroblock, QP_{im} is the quantization step of the m^{th} macroblock in the i^{th} camera view. $BaseQP$ is a parameter that can be used to adjust the rate of the compressed bitstream. The QP_{im}'s are then used on a modified H.264 codec, where the quantization step varies from macroblock to macroblock, but no side information about the QP_{im} used is included, since these weight maps can be obtained at the decoder by reproducing them following the same procedure (the particle generation may involve random number generations, though site A and site B can always agree upon a common seed during the setup stage).

4. EXPERIMENTAL RESULTS

We captured a real-world face movement sequence to simulate an immersive teleconferencing experience. The sequence is tracked with a feature-based tracker as in [6], which we assume is perfect. We assume the server site knows the previous face position $\boldsymbol{x}_{t-1}$, but does not possess the current position $\boldsymbol{x}_t$. We experimented with two deterministic motion models:

$$f_1(\boldsymbol{x}_{t-1}) = \boldsymbol{x}_{t-1},$$

which uses the previous face position as the current face position, and

$$f_2(\boldsymbol{x}_{t-1}) = \boldsymbol{x}_{t-1} + \dot{\boldsymbol{x}}_{t-1},$$

where $\dot{\boldsymbol{x}}_{t-1} = \boldsymbol{x}_{t-1} - \boldsymbol{x}_{t-2}$ is the face movement speed. In the following, we refer the approach adopting $f_1(\boldsymbol{x}_{t-1})$ as *using static motion model*, and that adopting $f_2(\boldsymbol{x}_{t-1})$ as *using constant velocity model*.

Figure 4 shows the face position prediction error using the two models in the unit of pixels, where each pixel corresponds to about 0.25 cm in the user's physical world, or maps to about 2.25 cm in the world coordinate of the *breakdancer* sequence. It can be seen that the user stayed relatively still through frame 60, and then moved significantly between frame 60 and 100, where both schemes have large prediction errors.

We compare six approaches to encode the breakdancer sequences. The first approach directly encodes the sequences with the H.264/AVC software JM 16.2 [14]. For simplicity, all 8 sequences are encoded separately, as in all the other tested approaches. In the second approach, we assume the server magically know the current face position $\boldsymbol{x}_t$, which corresponds to the case in [8]. In the third and fourth methods, we assume only the previous face position $\boldsymbol{x}_{t-1}$ is

available, and the deterministic functions $f_1(\boldsymbol{x}_{t-1})$ and $f_2(\boldsymbol{x}_{t-1})$ are used to predict the current position $\boldsymbol{x}_t$. The predicted positions are directly used to encode the sequence, which is similar to [8] except that these predicted positions have errors (as was shown in Figure 4). In the last two methods, we follow the proposed scheme outlined in Section 3, where previous face position distribution $p(\boldsymbol{x}_{t-1}|\boldsymbol{Z}_{t-1})$ is first resampled with 20 particles. Each particle then follows the motion model $f_1(\boldsymbol{x}_{t-1})$ and $f_2(\boldsymbol{x}_{t-1})$ and is diffused to represent $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$. The average weight maps are generated as Section 3.1 and the sequences are compressed as in Section 3.2.

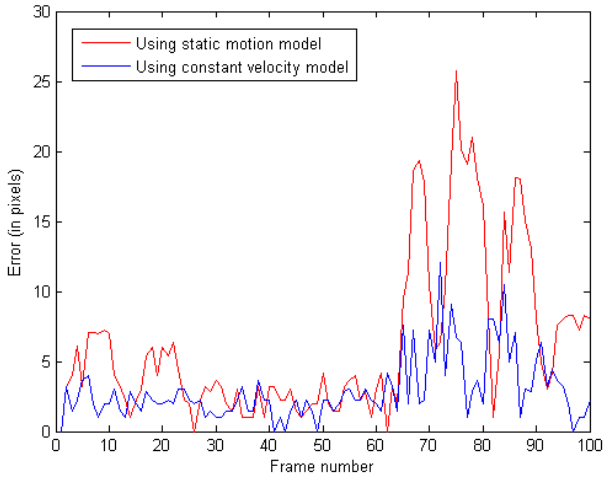

Figure 4 Face position prediction error using previous and predicted positions.

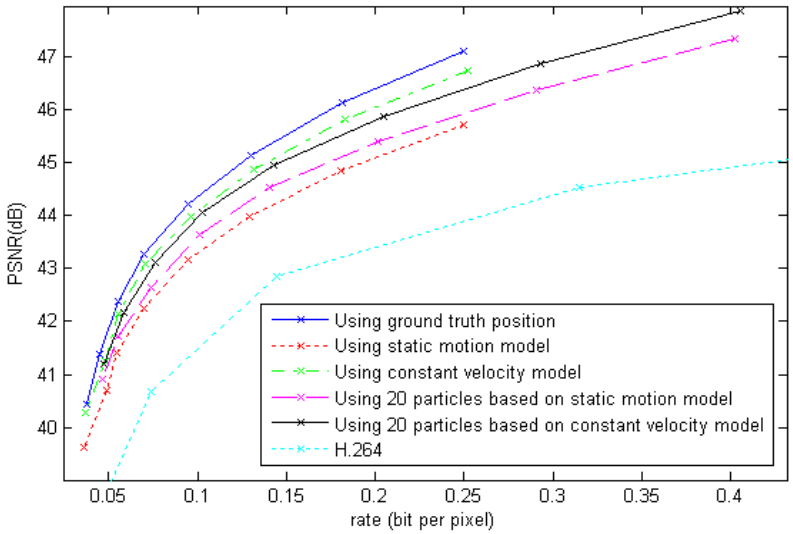

Figure 5 Average PSNR of the synthesized images at the client site for various schemes.

Figure 5 shows the rendering quality of the synthesized images at the client site. Note all schemes based on predicted viewer positions outperform the standard H.264 scheme significantly. Using the ground truth position gives the highest PSNR, though this scheme is impractical. In the rest four approaches, using constant velocity model appears to have the best performance, while using static motion model has the worst performance. Using 20 particles based on the static motion model performs better than not using particles, while using 20 particles based on the constant velocity model performs slightly worse.

However, the average PSNR only reveals part of the story. In Figure 6 we plot the frame-by-frame PSNR of various schemes at around 0.07 bit per pixel. Note the particle based approaches generally have a much more stable PSNR over the whole sequence. This demonstrates that our particle filter based joint tracking and compression scheme are more robust to errors in predicted face positions. In Table 1 we list the variance of PSNR over the whole sequence. It can be seen that with particle based schemes, the variances can be reduced for over 50% when compared with the corresponding approaches without using particles.

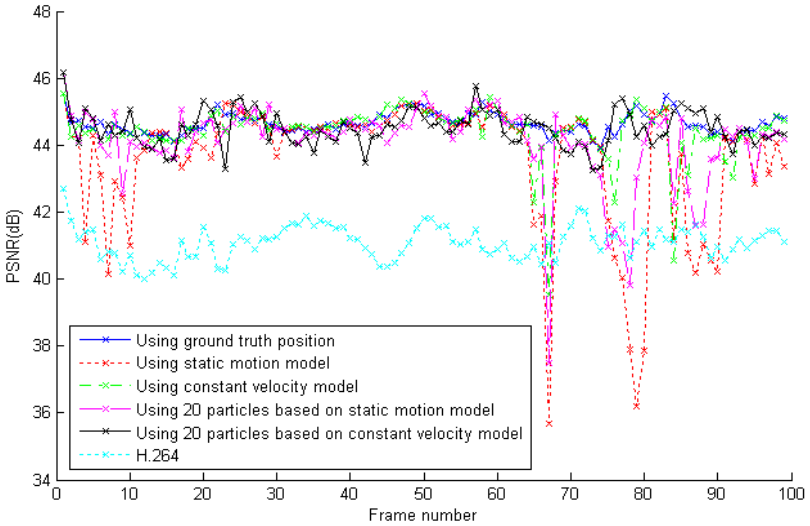

Figure 6 Frame-by-frame PSNR of the synthesized images at the client site for various schemes.

Table 1 Variance of PSNR for various schemes.

Approach	**Variance of PSNR (dB)**
Using ground truth position	0.097
Using static motion model	3.819
Using constant velocity model	0.693
Using 20 particles based on static motion model	1.477
Using 20 particles based on constant velocity model	0.285
H.264	0.246

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel framework for joint tracking and multiview video compression. The idea was to introduce particles to better represent the predicted face position distribution $p(\boldsymbol{x}_t|\boldsymbol{Z}_{t-1})$, and compute compression quality of the pixels by synthesizing multiple virtual views at the server site based on the particles. Such a scheme can accommodate possible errors in the predicted face position, which can be more robust than when only a single predicted position is used.

Immersive teleconferencing creates a new exciting research venue for multiview video processing, compression and streaming. While significant improvement has already been shown with our simple approach, there are many future research opportunities. In particular, exploring the interview redundancy may further improve the coding efficiency. Joint source/channel coding can also be helpful if there is interference in the transmission paths.

REFERENCES

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multi-view imaging and 3dtv," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10–21, 2007.

[2] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy, "Viewcast: View dissemination and management for multi-party 3d tele-immersive environments," in *ACM Multimedia*, 2007.

[3] H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. Goss, W. Culbertson, and T. Malzbender, "The coliseum immersive teleconferencing system," Tech. Rep., HP Labs, 2002.

[4] M. Flierl and B. Girod, "Multiview video compression," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 66–76, 2007.

[5] C. Zhang and J. Li, "Interactive browsing of 3D environment over the internet," in *Proc. SPIE VCIP*, 2001.

[6] C. Zhang, Z. Yin and D. Florencio, "Improving depth perception with motion parallax and its application in teleconferencing", *MMSP*, 2009.

[7] E. Kurutepe, R. Civanlar and M. Tekalp, "Client-driven selective streaming of multiview video for interactive 3DTV", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol. 17, No. 11, pp. 1558–1565, Nov. 2007.

[8] D. Florêncio and C. Zhang, "Multiview video compression and streaming based on predicted viewer position", *ICASSP* 2009.

[9] P. Ramanathan, M. Kalman and B. Girod, "Rate distortion optimized interactive light field streaming", *IEEE Trans. On Multimedia*, vol. 9, no. 4, pp. 813–825, June 2007.

[10] M. Isard and A. Blake, "CONDENSATION – conditional density propagation for visual tracking", *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[11] H.-Y. Shum, S.-C. Chan and S. B. Kang, *Image-Based Rendering*, Springer, 2006.

[12] C. L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM SIGGRAPH*, 2004.

[13] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, "Unstructured lumigraph rendering," *ACM SIGGRAPH*, 2001.

[14] http://iphome.hhi.de/suehring/tml/