3D SCENE MODELING FOR DISTRIBUTED VIDEO CODING

Matthieu Maitre[*]

Beckman Institute
University of Illinois at Urbana-Champaign
maitre@uiuc.edu

Christine Guillemot, Luce Morin[†]

IRISA
Rennes, France
{christine.guillemot, luce.morin}@irisa.fr

ABSTRACT

The compression efficiency of Distributed Video-Coding (DVC) suffers from the necessity of transmitting a large number of key-frames which are intra-coded. This paper describes a new 3D model-based DVC approach which reduces the key-frame frequency. The decoder first recovers a 3D model from the key-frames. It then predicts the intermediate frames by projecting it onto 2D image planes and applying image-based rendering techniques. This paper also introduces a new quasi-DVC method relying on a limited point tracking at the encoder. It greatly improves the prediction PSNR, while only slightly increasing the encoder complexity. It also allows the encoder to adaptively select the key-frames based on the video motion-content.

Index Terms— Video coding, Stereo vision

1. INTRODUCTION

Distributed Video Coding (DVC) is a recent approach which is being investigated as a possible alternative to classical predictive coders for applications requiring low-complexity encoders as well as error resilience. By moving the motion-compensation stage to the decoder, it keeps the encoder complexity to a minimum and benefits from compression based on both spatial and temporal correlations. Moreover, since the frames are encoded independently, it avoids temporal propagation of errors.

Previous studies [1, 2] have shown the potential of DVC but also noted the compression gap remaining between DVC and predictive coding. It is in part due to the poor performance of block-based motion-compensation when applied to distant key-frames. Key-frames need to be sent sparsely because of their high bitrate cost. This makes the motion fields between them often too complex to be approximated by spatially blockwise-constant and temporally piecewise-constant motion fields. This also requires motion vectors to be searched inside large regions, which increases the likelihood of large errors.

This calls for the introduction of new motion models. Unlike predictive coding, DVC does not require motion-model parameters to be sent through the communication channel, thus allowing complex models without compression penalty. In this paper, we specialize DVC to videos of static scenes obtained from a unique moving camera, a type of video of particular importance to remote exploration by drones or remote virtual reality. We take advantage of techniques developed in the context of Structure-from-Motion (SfM) [3] and propose motion models based on 3D information. First, the decoder estimates the camera parameters and the 3D scene-model from the key-frames. Then, it linearly interpolates the camera parameters at intermediate times and projects the 3D model onto the associated image planes, giving motion fields between the intermediate frames and the key-frames. Finally, it predicts the intermediate frames using Image-Based Rendering (IBR) techniques [4].

However, our experimental results shall show that even if this approach greatly improves the quality of estimated motion-fields, its impact on the PSNR of the predicted frames is limited. The prediction is hindered by the interpolation of camera-parameters at intermediate times. Therefore, we propose to go beyond DVC with an approach called quasi-DVC (qDVC). The encoder shares some limited information between frames, under the form of point tracks. This allows the decoder to estimate the camera parameters at intermediate times, instead of interpolating them. Moreover, this only slightly increases the encoder complexity. Finally, this allows the encoder to adapt the key-frame frequency to the video motion-content, a feature not possible in the DVC framework.

This article presents both DVC and quasi-DVC approaches. Section 2 describes the encoders, Section 3 details the decoders and Section 4 presents our experimental results.

2. ENCODER

2.1. Distributed video encoding

The 3D-DVC encoder is identical to a 2D-DVC encoder, as shown in Figure 1. We consider the pixel-domain codec described in [5] which improves upon the approach proposed in [1]. The DVC encoder begins by splitting the input video-

[*]The first author performed the work while at the IRISA.

[†]This work has been partly funded by the European Commission in the context of the network of excellence SIMILAR and of the IST-Discover project.

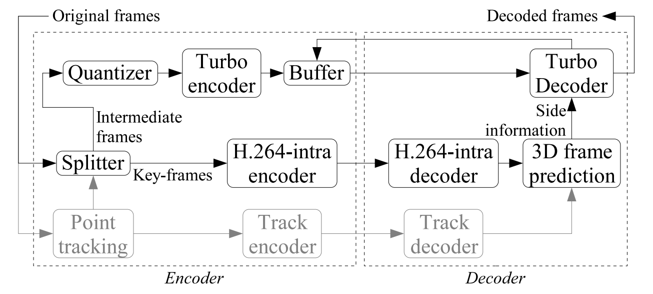

Fig. 1. Video codecs: 3D-DVC and 3D-qDVC. They differ by the point-track stream, only present in 3D-qDVC (in gray).

stream into key-frames and Wyner-Ziv (WZ) frames, using a constant key-frame frequency. It encodes the key-frames using a standard intra-encoder (H.264-intra in our case). It quantizes the WZ-frames into bit-planes, turbo-encodes them and transmits punctured parity-bits.

2.2. Quasi-distributed video encoding

The limitations of this purely DVC approach led us to consider an alternative quasi-DVC solution, in which a limited point-tracking is added to the encoder, as shown in Figure 1. Point-tracks offer two major benefits. First, they allow the decoder to estimate the camera parameters at intermediate times, instead of interpolating them. This greatly reduces the reprojection errors. Second, they enable the encoder to dynamically adapt the key-frame frequency: a new key-frame is only sent when the length of the longest track or the number of lost tracks exceed some thresholds. However, point-tracking also introduces overheads on the encoder complexity and the bitrate, which must be kept to a minimum.

We propose to attain these goals using a modified Kanade-Lucas-Tomasi (KLT) tracker [6]. First, feature points are detected on key-frames using the Harris-Stephen corner detector [7]. These points can be very sparse since the decoder only needs them to recover 11 camera parameters. Points are then tracked between consecutive frames by looking for similar intensity neighborhoods, which are assumed to follow a translational motion model. Points are robustly tracked with sub-pixel accuracy by minimizing the Sum of Squared Differences (SSD) between neighborhoods using the Levenberg-Marquardt approach [8]. The bitrate overhead is reduced by using DPCM-encoding.

The minimization relies on image derivatives. It is therefore limited to small motions. The range of tolerated motions is increased through a conditional multiscale approach. A basic coarse-to-fine scheme would introduce a systematic complexity overhead and miss corners too weak to be detected at coarser resolutions. Instead, the tracker first performs the SSD minimization at the finest resolution and only falls back to coarser resolutions at tracks for which it failed.

```
foreach pair of consecutive key-frames do
    Detect and match feature points
    Robustly estimate the fundamental matrix $\mathbf{F}$
    Obtain the projection matrices $\mathbf{P}_0$ and $\mathbf{P}_1$
    Triangulate the projective depths $\lambda$
    Propagate correspondences along edges
    Interpolate or estimate the projection matrices $\mathbf{P}_t$
    Interpolate the WZ-frames from the key-frames
```

Algorithm 1: Scene modeling and frame prediction

3. DECODER

3.1. Overview

As shown in Figure 1 and Algorithm 1, the decoder starts by decoding the key-frames. Then, it predicts the WZ-frames using motion interpolation. Finally, it corrects this prediction using the parity bits from the encoder and turbo-decoding. The 3D-DVC decoder differs from previous 2D-DVC decoders by its frame prediction stage. The 3D-qDVC decoder also takes advantage of point tracks to estimate the camera parameters associated with the WZ-frames, instead of interpolating them.

3.2. Scene modeling

Frame prediction takes a pair of consecutive key-frames as an input. Without loss of generality, we assume that the first key-frame was taken at time $t = 0$ and the second key-frame at time $t = 1$. Feature-points on each key-frame are denoted by respectively x_0 and x_1. A correspondence between key-frames is denoted by $(\mathsf{x}_0, \mathsf{x}_1)$. Points are assumed to be in homogeneous coordinates. A correspondence is said to be valid when it stems from the projection of a unique 3D point X onto the image planes. That is $\exists \mathsf{X}$ s.t. $\mathsf{x}_0 \sim \mathbf{P}_0\mathsf{X}, \mathsf{x}_1 \sim \mathbf{P}_1\mathsf{X}$ where $\mathbf{P}_0$ and $\mathbf{P}_1$ are the projection matrices associated with each key-frame and '$\sim$' denotes an equality up-to-scale. This condition is equivalent to $\mathsf{x}_1^t\mathbf{F}\mathsf{x}_0 = 0$ where $\mathbf{F}$ is the so-called fundamental matrix. More details can be found in [3].

Like at the encoder, feature-points are detected at the decoder using the Harris-Stephen corner detector. However, the number of feature-points allowed this time is much greater to obtain as many 3D points as possible. Also, the detector has now to cope with quantization noise.

Feature points are then matched across key-frames to obtain correspondences. A cascade of tests removes erroneous correspondences. Tests are ordered by increasing complexity so that the most complex ones handle the least correspondences. At first, correspondences between all points are considered. A first test removes correspondences whose motions are too large. A second test compares the intensity histograms of feature-point neighborhoods and removes those with poor chi-square statistics [8]. A third test proceeds similarly using the SSD as a criterion, performing a local optimization of feature-point locations to obtain meaningful SSD values. A fourth test enforces that each point belongs to at most one correspondence. Finally, a fifth test removes correspondences which are not compatible with the epipolar geometry found by robustly estimating the fundamental matrix $\mathbf{F}$.

The camera parameters are estimated using self-calibration. The interpolation of camera parameters requires the 3D space to be euclidean. However, this is only possible when the camera motion between key-frames is generic enough, a condition rarely met in practice. Instead, we settle for quasi-euclidean self-calibration. The projection matrices can be written as $\mathbf{P}_0 = [\mathbf{I}\,0]\,, \mathbf{P}_1 = [\mathbf{R}\,\mathsf{t}]$ where $\mathbf{I}$ is the identity matrix, $\mathbf{R}$ a matrix and t a vector. These quantities are related to the fundamental matrix by $\mathsf{t} \in \ker(\mathbf{F})$ and $\mathbf{R} = [\mathsf{t}]_\times \mathbf{F} - \mathsf{t}\ \mathsf{a}^t$, where a is a vector and $[.]_\times$ denotes the cross-product operator. Assuming small camera rotations and slowly varying intrinsic parameters between key-frames, the vector a is found by minimizing $\left\| [\mathsf{t}]_\times \mathbf{F} - \mathsf{t}\ \mathsf{a}^t - \mathbf{I} \right\|$.

A cloud of 3D points is recovered by computing a pair of projective depths $\{\lambda_0, \lambda_1\}$ from each correspondence. These scalars are solutions of the equation $\lambda_1 \mathsf{x}_1 = \lambda_0 \mathbf{R} \mathsf{x}_0 + \mathsf{t}$ and are related to the underlying 3D point by $\mathsf{X} = [\lambda_0 \mathsf{x}_0^t\ 1]^t$.

Projective depths are only known at corners. Therefore, interpolation is required to obtain dense motion fields. Such an approximation is particularly harmful to the prediction PSNR in edge regions. Fortunately, the intersection of edges and epipolar lines gives points which, like corners, can be matched to obtain more correspondences.

Edge-points are detected in the first key-frame using the Canny edge-detector [3]. Correspondences are propagated by matching edge-points close to previously matched points. For a given edge-point, matching consists in a SSD-based full search along a portion of the associated epipolar line, followed by sub-pixel refinement around the best candidate via golden search [8]. The full-search domain is a small window centered around the location that the matching point would have if it followed the same motion as the one of its nearest correspondence.

3.3. WZ-Frame interpolation

Camera parameters need to be known at intermediate times to be able to project the 3D model onto intermediate image planes and obtain motion fields. The 3D-DVC decoder linearly interpolates them from the ones associated with the key-frames. On the other hand, the 3D-qDVC decoder estimates them from the point tracks. Using the locations of these tracks at $t = 0$ and $t = 1$, it computes the associated 3D points X, as described in the previous section. Then, it obtains the projection matrices $\mathbf{P}_t$ at each intermediate time t by solving the set of equations $\mathsf{x}_t \sim \mathbf{P}_t \mathsf{X}$. The estimation approach, unlike the interpolation one, does not require the 3D space to be truly euclidean and does not assume a constant camera-motion.

The projection of the 3D points onto the image planes gives motion vectors from the WZ-frames to the key-frames at corners and edges. They need to be interpolated to obtain dense motion fields. We present two interpolation schemes which differ by their assumptions on smoothness. One relies on block-matching under the epipolar geometry constraint and the other on the fitting of a mesh onto the 3D point cloud. The latter is more resistant to erroneous correspondences but tends to over-smooth depth discontinuities.

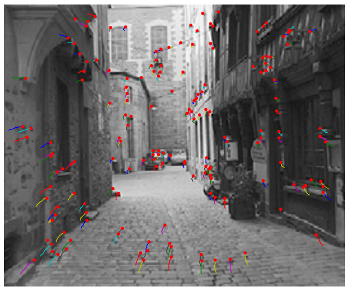
Fig. 2. Tracking (red dots: corners in first key-frame, multi-color curves: tracks in following WZ-frames).

The epipolar block-matching scheme divides WZ-frames into blocks and searches for pairs of blocks with low SSD on the key-frames. This one-dimensional search is performed along epipolar lines in one of the key-frame, the locations in the other key-frame resulting from trifocal transfer [3]. As for the propagation along edges, full-search domains are small windows centered around locations determined by the nearest correspondences and sub-pixel accuracy is attained using golden search. Finally, each pair of blocks is linearly blended based on time to predict the WZ-frames.

The mesh fitting scheme estimates for each WZ-frame an elevation mesh made of regular triangles. The projective depths associated with the mesh vertices are determined by energy minimization. The energy is defined as the depth distance between the mesh and the 3D points along with a Tikhonov regularization. Correspondences which lead to inverted triangles or large depth errors are removed. This mesh is projected onto the key-frames, which are then warped using 2D texture mapping and blended to predict the WZ-frame.

4. EXPERIMENTAL RESULTS

The codecs were evaluated on several sequences. We present here the results on the 50 first frames of the street sequence, a CIF sequence at 30fps. The camera is mostly moving forward, with some slight rotations. Figure 2 displays its first frame, along with the points-tracks between the first two key-frames. Note their sparseness. The bitrate overhead introduced by point-tracking is .01b/sample. The encoder-complexity overhead is negligible compared to predictive 3D video-encoding [9], and 35 times smaller than basic 2D block-matching with an integer search-range of equivalent size. The

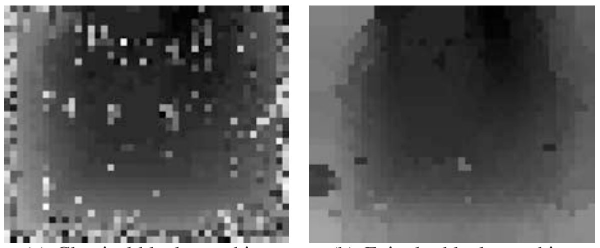

Fig. 3. Norm of the block motion-fields between the first two key-frames (same intensity scaling)..

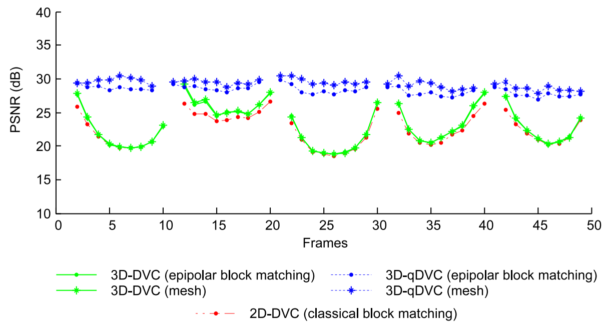

Fig. 4. PSNR of interpolated WZ-frames using lossless key-frames.

adaptative key-frame frequency induced by point-tracks is approximately one key-frame every 10 frames. To allow comparison, the key-frame frequencies of 2D-DVC, 3D-DVC and H.264-inter were set to the same value.

Figure 3 compares the motion-fields obtained with classical block-matching and with the proposed epipolar block matching. The latter is qualitatively superior, the number and size of errors being much smaller. Figure 4 compares the PSNR of the WZ-frame interpolation by 2D-DVC, 3D-DVC and 3D-qDVC using lossless key-frames. The qualitative improvement of motion-fields has no significant effect. However, point-tracking increases the PSNR by up to 10dB. Figure 5 compares the rate-distortion performances of H.264-intra, H.264-inter (IP...PI), the 2D-DVC Discover codec [5] and 3D-qDVC. Our codec outperforms both 2D-DVC and H.264-intra, and approaches H.264-inter at lower bitrates.

5. CONCLUSION

In this paper we proposed new DVC methods based on 3D reconstruction. We showed that the epipolar geometry helps improving the quality of motion-fields. Moreover, adding point-tracking to the encoder significantly increases the prediction PSNR and allows adaptive key-frame frequencies, while only introducing limited overheads. Future work shall consider the non-i.i.d nature of prediction errors to improve turbo-decoding.

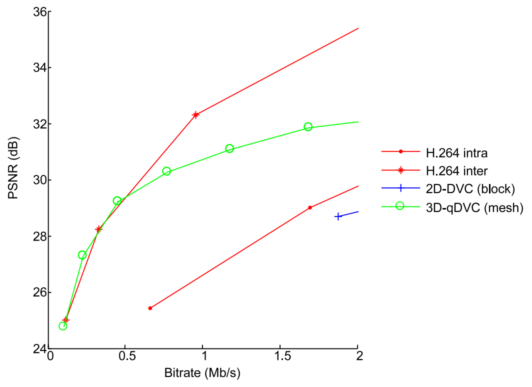

Fig. 5. Rate distortion for H.264-intra, H.264-inter, 2D-DVC and 3D-qDVC, using lossy key-frames.

6. ACKNOWLEDGMENT

We are thankful to both the IST development team and the Discover software team for providing the 2D-DVC codec.

7. REFERENCES

[1] B. Girod, A.M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. of the IEEE*, vol. 93, no. 1, pp. 71–83, January 2005.

[2] R. Purit and K. Ramchandran, "PRISM: A new robust video coding architecture based on distributed compression principles," in *Proc. Allerton Conf.*, 2002.

[3] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, August 2002.

[4] H.Y. Shum and S.B. Kang, "A review of image-based rendering techniques," in *Proc. SPIE Conf. on Visual Com. and Im. Proc.*, 2000.

[5] J. Ascenso, C. Brites, and F. Pereira, "Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding," in *EURASIP Conf. on SIPMCS*, 2005.

[6] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: a unifying framework," *IJCV*, vol. 56, no. 3, pp. 221–255, 2004.

[7] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vision Conf.*, 1988, pp. 147–151.

[8] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C : The Art of Scientific Computing*, Cambridge University Press, 1993.

[9] R. Balter, P. Gioia, and L. Morin, "Time evolving 3D model representation for scalable video coding," in *Proc. ICIP*, 2005.