

# Compression Performances of Computer Vision Based Coding\*

Franck GALPIN<sup>†</sup>, Luce MORIN<sup>††</sup>, *Nonmembers*, and Koichiro DEGUCHI<sup>†</sup>, *Regular Member*

**SUMMARY** This paper presents new results in the field of very low bitrate coding and compression using 3D informations. Contrary to prior art in model-based coding where 3D models have to be known, the 3D models are automatically computed from the original video sequence. The camera parameters and the scene content are supposed unknown and the video sequence is processed on the fly. A stream of 3D models is then extracted and compressed, using adapted compression techniques. We finally show the results of the proposed compression scheme, and the efficiency of this approach.

**key words:** 3D model-based coding, compression, shape-from-motion.

## 1. Introduction

Instead of representing video sequence as a set of pixels, like in classical video coding algorithms, 3D model based coding aims at representing the video sequence with one or several textured 3D models and a set of camera/objects parameters. This topic has been addressed for many years, particularly in the field of video-conference where a 3D model of the human face is used to represent the video sequence of the speaker [1]. In this paper, we address the problem of the representation of static scenes viewed by a monocular moving camera. Both scene and camera are supposed unknown, means that we do not make any assumptions on scene contents nor on the camera parameters (both intrinsic and extrinsic).

The 3D-based coding has several advantages compared to image based coding: first it allows very low bitrate coding since the only informations to transmit are the camera parameters (position, orientation, focal length etc.), the 3D model of the scene (if it is unknown on the decoder side) and corresponding texture image mapped on this 3D model. Moreover, with such a coding, we can benefit of 3D features such as view-point transformation, illumination change with respect to geometry or synthetic 3D-model coding. However many works [2] [3] are based on known 3D-models, and few results are available concerning 3D-model based video compression with unknown 3D-models. On the other hand, computer vision techniques, and partic-

ularly shape-from-motion techniques allow to extract 3D-models of a static scene viewed by a moving monocular camera [4] [5] [6]. However, the shape-from-motion process still is a difficult problem without simplifying assumption on the scene or the camera parameters. Another solution is to take into account a very large amount of data along the video sequence in order to recover the whole 3D structure of the scene [7] [8]. These last method usually assumes a specific camera motion, as an exploring motion, in order to solve ambiguities in both scene structure and camera motion.

In this paper, we present an automatic scheme for 3D-model based video compression of static scene viewed by a moving monocular camera. A typical case is an outside walk using a simple cam-corder. Video coding framework brings some particular constraints: first we must consider that camera parameters (both extrinsic and intrinsic) and scene content are unknown. We further assume that the camera motion is not constrained to follow a specific path (like an exploring motion), however we assume that the camera motion is not degenerated (for example a pure rotation). We must also take into account that video coding requires an on-the-fly process: we must process only a small part of the video sequence, not waiting for all data to process the video sequence. We then have chosen to base our process on the extraction of a stream of 3D models, instead of a unique 3D model which requires too much informations in the case of video coding.

In the following, we first briefly describe the extraction of the 3D models and then pay more attention on the compression stage. Results of compression performance on real video sequences are finally shown and discussed.

## 2. 3D models stream generation

We first present the structure of the representation: each 3D model is extracted and used for a small portion of the video sequence called GOP (Group Of Picture). The GOPs overlap themselves, that is the last image of a GOP is the same as the first image of the next one. We call these images *Keyframes*. The keyframes are automatically selected adaptively to the video contents, based on several criteria depending on motion, percentage of outgoing points in images and 3D reconstruction stability [9].

<sup>†</sup>Graduate School of Information Sciences, Tohoku University, Aoba-campus 01, Sendai 980-8579, Japan  
Telephone: 022-217-7014; Fax: 022-217-7015.

<sup>††</sup>Irisa/INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France

\*This paper was presented at MVA2002, Nara, Japan.

For each GOP, a 3D model is automatically extracted on the fly. The principle of extraction is based on shape-from-motion method: we use a dense mesh-based motion estimator using multi-grid and multi-resolution approaches [11].

The camera intrinsic parameters are estimated using a self-calibration algorithm or fixed to approximated values. Accurate intrinsic parameters are not required since we design our representation to be robust to inaccurate intrinsic parameters. The extrinsic parameters are computed using classical calibration methods and an adapted bundle adjustment method [9]. The dense motion field from the first to the last image of the GOP and camera parameters for these two images allow to reconstruct a dense depth map of the first image of the GOP. Figure 1 shows an example of depth map extracted from the *Street* video sequence (see results section).



**Fig. 1** An example of a depth map (bottom) extract from the *Street* video sequence. The top figure is the corresponding image in the sequence.

A uniform triangular mesh is then applied on each first image of each GOPs which gives a 3D model and its associated texture image (the first image of the GOP). Camera extrinsic parameters are then retrieved for each image of the video sequence using a pose estimation algorithm. The original video sequence reconstruction is then realized by the projection of each 3D textured

models. A 3D fading is also used to smoothly switch from one GOP to the next one [9]. One must notice that the decoding stage uses classical 3D projection algorithms and is then realized in real time.

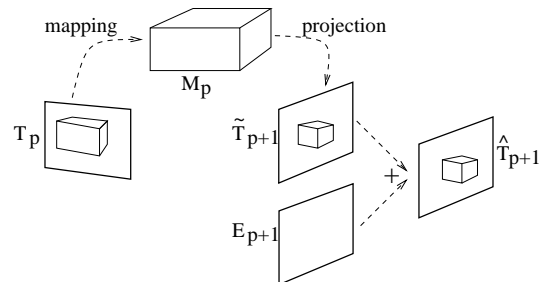
### 3. Stream compression

Once the original video sequence has been encoded as a 3D model stream, we want to compress it. The compression of the different parts of the representation (3D model, texture images and camera parameters) use adapted compression techniques.

#### 3.1 Texture images

We use a structure similar to the MPEG format one for low bitrate (IPP), that is: one *Intra* (I) image at the beginning of the video stream followed by *Predicted* (P) images. *Intra* images do not depend of other images, whereas *Predicted* images depend on the previously decoded image.

The first image of the sequence (*Intra* type) is compressed using a method similar to JPEG2000 standard [10], that is: a wavelet transformation, an adaptive quantification (using *EBCOT*) and an entropic coding.



**Fig. 2** Principle of predictive coding method of the texture images.

#### 3.2 3D models

Each 3D model can be seen as a 3D model extruded from the uniformly meshed depth map. The 3D models stream, that is the decimated depth map, is also compressed using a wavelet transformation. However, we use an adapted quantification law: the depth (from the camera position to the 3D vertex of the 3D mesh) of each point of the 3D model is quantified using a normalized inverse law:

Equations of back-projection for a 3D point  $M = (x, y, z)$  are given by:

$$\begin{aligned} \tilde{m} &= K_t(I_3|0_3)\tilde{M} \\ \tilde{m}' &= K_{t'}(R|t)\tilde{M} \end{aligned} \quad (1)$$

$$d(m) = \|m - m'\| \quad (2)$$

We assume that:

- intrinsic parameters are fixed for a given GOP,
- pixel motion induced by the camera rotation is insignificant compared to the one induced by the camera translation.

We then obtain:

$$m = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ z \end{pmatrix} \quad (3)$$

with  $t = (t_x, t_y, t_z)$  :

$$m' = \begin{pmatrix} \frac{x+t_x}{z+t_z} \\ \frac{y+t_y}{z+t_z} \\ z+t_z \end{pmatrix} \quad (4)$$

That is, for a pure translation:

$$d(m) \simeq \sqrt{\left(\frac{x}{z} - \frac{x+t_x}{z+t_z}\right)^2 + \left(\frac{y}{z} - \frac{y+t_y}{z+t_z}\right)^2} \quad (5)$$

or:

$$d^2(m) \simeq \frac{(x.t_z - z.t_x)^2 + (y.t_z - z.t_y)^2}{z^2.(z+t_z)^2} \quad (6)$$

$$d^2(\alpha) \simeq \frac{(u_x.t_z - u_z.t_x)^2 + (u_y.t_z - u_z.t_y)^2}{u_z^2.(\alpha.u_z + t_z)^2} \quad (7)$$

for  $\alpha.u_z \gg t_z$ , which is a reasonable hypothesis for the scenes we deal with:

$$d^2(\alpha) \simeq \frac{1}{\alpha^2} \cdot \frac{(u_x.t_z - u_z.t_x)^2 + (u_y.t_z - u_z.t_y)^2}{u_z^4} \quad (8)$$

that is  $d(\alpha)$  is proportional to  $1/\alpha$ . It simply implies that the more a point is far from the optical center, the less the projection error is.

This quantification allows to compress the 3D model with a linear reprojection error regarding to the quantification step. The mesh size also allows to adapt the bitrate of the compressed video sequence. We typically fix the mesh size to 8 for low bitrate and 12 for very low bitrate. The quantization step is chosen such as the projection error is less than 0.5 pixel.

### 3.3 Compressed stream generation

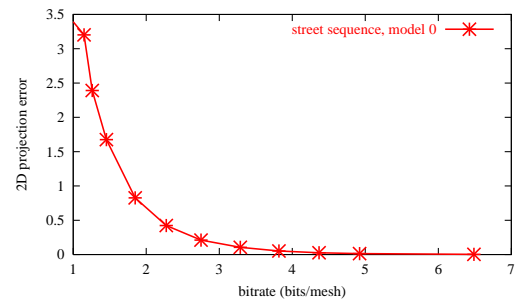
The representation is effectively compressed, allowing to compute the real coding cost. We can accurately choose the target bitrate  $R_c$  thanks to *EBCOT* coding of texture images. The coding cost  $R_{text}$  of the current texture image is simply chosen as:

$$R_{text} = \frac{R_c.(t_{n+1} - t_n)}{fr} - (R_{cam} + R_M) \quad (9)$$

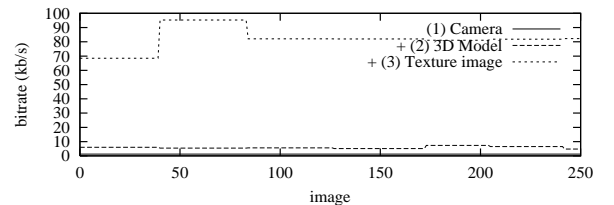
with  $fr$  the framerate of the video sequence,  $R_{cam}$  the coding cost of the camera parameters (100 bits per *Keyframe* images and 50 bits for other intermediate images),  $R_M$  the coding cost of the 3D model and  $t_n$  the time at GOP  $n$ .

### 3.4 Representation properties

Figure 3 shows a typical result on the first 3D model extracted from a real video sequence (see figures 5 and 6). We plot 2D reprojection error of the 3D model, compared to the reprojection without compression, depending on the coding cost per mesh in bits. We see that the 3D model cost is between 2 and 3 bits per mesh for a reprojection error less than 0.5 pixel, which is very low compared to the texture cost. This is confirmed with figure 4: we can see that the texture image cost remains the major coding cost in the proposed representation.



**Fig. 3** Typical bitrate obtained with the adapted quantification.



**Fig. 4** Coding cost of each part of the representation: (1) Camera cost, (2) adding the 3D model cost, (3) adding the texture image cost.

One must notice that the proposed representation intrinsically allows 3D manipulations like stereo-visualization or scene manipulation [9].

## 4. Results

The compression scheme performance is first compared to state of the art low bitrate encoder (H26L) and we show new results in very low bitrate coding area.

On figure (8), we show a result with the *Street* video sequence<sup>†</sup>: format is CIF 4:2:0 at 25Hz, global motion is a translation along z-axis, neither camera parameters nor scene content are known. The internal

<sup>†</sup>video sequence from Thomson Multimedia and FTRD

camera parameters are fixed to typical values and focal length is fixed to an approximate value. The video sequence is compressed with H26L video encoder <sup>††</sup> at 82kb/s which is the minimum bitrate with this coder. The sequence is also encoded with the proposed method denoted *Rec3D* with the same parameters. The figure (8) shows *PSNR* score along the sequence: *Texture* curve denotes the *PSNR* of the compressed texture images mapped on the successive 3D models. The *Image* curve denotes the *PSNR* of the reconstructed sequence with the *Rec3D* method and *H26L* with the H26L encoder. The curves show that *PSNR* score are similar for both methods, and better for *Rec3D* method when the 3D model is refresh (peak on the second curve). On figures (5-7), we compare visual quality of reconstructed video sequence: the images clearly show that *Rec3D* provides better visual quality. For such a bitrate, H26L method introduces classical block artifacts and texture degradation whereas *Rec3D* provides better texture images resulting in good visual quality of the reconstructed video sequence.

On figures (9-12), we show similar results on the *Stairway* video sequence. The format of this sequence is CIF 4:2:0 at 25Hz, global motion is a translation along x-axis, neither camera parameters nor scene content are known. The internal camera parameters are fixed to typical values and focal length is fixed to an approximate value. One must notice that this video sequence is very shaky. The video sequence is compressed with the H26L video encoder at 125kb/s which is the minimum bitrate with this coder, and the proposed coder. The figure (12) shows that the *PSNR* of H26L coder is better than the one of *Rec3D* coder. However, the *PSNR* of the texture images and figures (9-11) show that the visual quality of the reconstructed sequence still is better with the *Rec3D* coder.

Moreover, the proposed method allows very low bitrate coding (up to 16kb/s for CIF, 25Hz format) which are not reachable by classical image based coders. The figures (13-15) shows results on the *Street* video sequence for a target bitrate of 16kb/s. The figure (15) shows the *PSNR* score along the video sequence: *Texture* curve shows that *PSNR* score remains good for such a bitrate. The figures (13) and (14) show an image extracted from the reconstructed video sequence: we see that global visual quality is quite good, despite compression artifacts. One must notice that such a bitrate is considered as very low for CIF/25Hz format video sequence.

## 5. Conclusion

We show results on real video sequences and evaluate performance of vision computer based approach for video compression. The results clearly show that such

an approach is very efficient for low bitrate coding and also show new results for very low bitrate coding. We show that the proposed scheme allows better performance than classical scheme like H26L for static scene video sequences. However, quality still is difficult to evaluate for such a coding approach: the method is intrinsically not based on a *pixel coding* approach, making classical quality measures (such as the *PSNR* score) unsuitable. Moreover, we plan to extend the proposed scheme to scene with moving objects, making the approach more general for video coding purpose.

## Acknowledgments

This work is supported by the Japan Society for the Promotion of Science (JSPS) through JSPS Postdoctoral Fellowship for Foreign Researchers.

## References

- [1] F. Prêteux and M. Malciu, "Model-based head tracking and 3d pose estimation," in *Visual Conference on Image Processing*, San Jose, California, 1998, pp. 94–110.
- [2] B. Girod and al., "3d image models and compression - synthetic hybrid or natural fit?," in *Proc. ICIP*, Oct. 1999.
- [3] G. Martinez, "Joint position estimation for object-based analysis-synthesis coding," in *Proc. VCIP*, June 2000.
- [4] M. Pollefeys, R. Koch, M. Vergauwen, B. Deknuydt, and L. Van Gool, "three-dimensional scene reconstruction from images," in *proceedings SPIE Electronic Imaging, Three-Dimensional Image Capture and Applications III*, 2000.
- [5] R. Kochand M. Pollefeys and L. Van Gool, "Realistic 3d scene modeling from uncalibrated image sequences," in *Proc. ICIP*, Oct. 1999.
- [6] M. Magnor and B. Girod, "Model-based coding of multi-viewpoint imagery," in *Proc. VCIP*, June 2000.
- [7] Andrew W. Fitzgibbon and Andrew Zisserman, "Automatic camera recovery for closed or open image sequences," in *ECCV (1)*, 1998, pp. 311–326.
- [8] D. Nistèr, "Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors," in *Proc. ECCV*, Dublin, Ireland, 2000, pp. 649–663.
- [9] Franck Galpin and Luce Morin, "Sliding adjustment for 3d video representation," *Eurasip Journal on Applied Signal Processing, special issue on Signal Processing for 3D Imaging and Virtual Reality*, vol.10, Oct. 2002.
- [10] Information Technology, "Jpeg2000 image coding system, iso/iec fdis 15444-1," 2000.
- [11] S. Pateux, G. Marquant, and D. Chavira-Martinez. Object mosaicking via meshes and crack-lines technique. application to low bit-rate video coding. In *Proceedings of Picture Coding Symposium 2001*, pages 417–420, Seoul, Korea, April 2001.

---

<sup>††</sup>TML coder v.6



**Fig. 5** Street video sequence at 82kb/s: image 100 reconstructed by H26L coder.



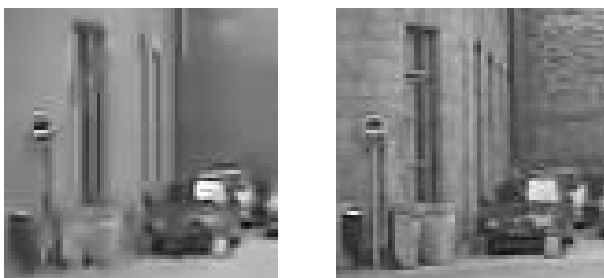
**Fig. 9** Stairway video sequence at 125kb/s: image 67 reconstructed by H26L coder.



**Fig. 6** Street video sequence at 82kb/s: image 100 reconstructed by Rec3D coder.



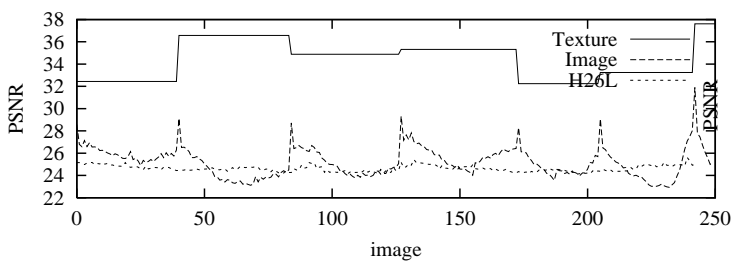
**Fig. 10** Stairway video sequence at 125kb/s: image 67 reconstructed by Rec3D coder.



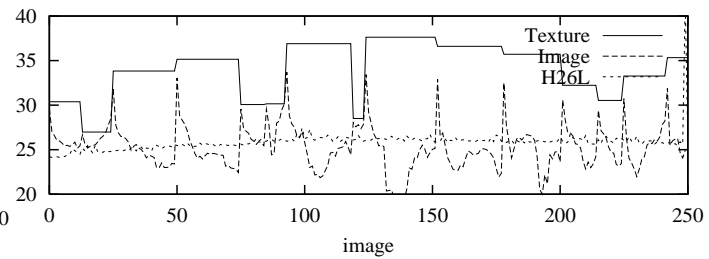
**Fig. 7** Street video sequence at 82kb/s: zoom on image 100 reconstructed by H26L (left) and Rec3D (right) coder.



**Fig. 11** Stairway video sequence at 125kb/s: zoom on image 67 reconstructed by H26L (left) and Rec3D (right) coder.



**Fig. 8** PSNR of Street video sequence at 82kb/s: comparison between H26L and Rec3d method.



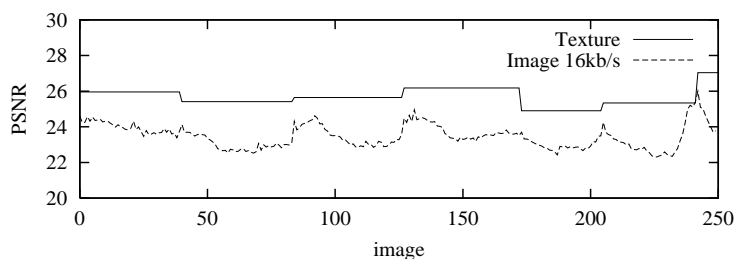
**Fig. 12** PSNR of Stairway video sequence at 125kb/s: comparison between H26L and Rec3d method.



**Fig. 13** *Street* video sequence at 16kb/s: image 100 reconstructed by Rec3D coder.



**Fig. 14** *Street* video sequence at 16kb/s: zoom on image 100 reconstructed by Rec3D coder.



**Fig. 15** *PSNR* of *Street* video sequence at 16kb/s compressed with Rec3D method.