Model-Based Video Coding Using Colour and Depth Cameras

David Sandberg
Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Per-Erik Forssén
Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Jens Ogniewski
Information Coding
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

***Abstract*—In this paper, we present a model-based video coding method that uses input from colour and depth cameras, such as the Microsoft Kinect. The model-based approach uses a 3D representation of the scene, enabling several other applications besides video playback. Some of these applications are stereoscopic viewing, object insertion for augmented reality and free viewpoint viewing.**

The video encoding step uses computer vision to estimate the camera motion. The scene geometry is represented by keyframes, which are encoded as *3D quads* using a quadtree, allowing good compression rates. Camera motion in-between keyframes is approximated to be linear. The relative camera positions at keyframes and the scene geometry are then compressed and transmitted to the decoder.

Our experiments demonstrate that the model-based approach delivers a high level of detail at competitively low bitrates.

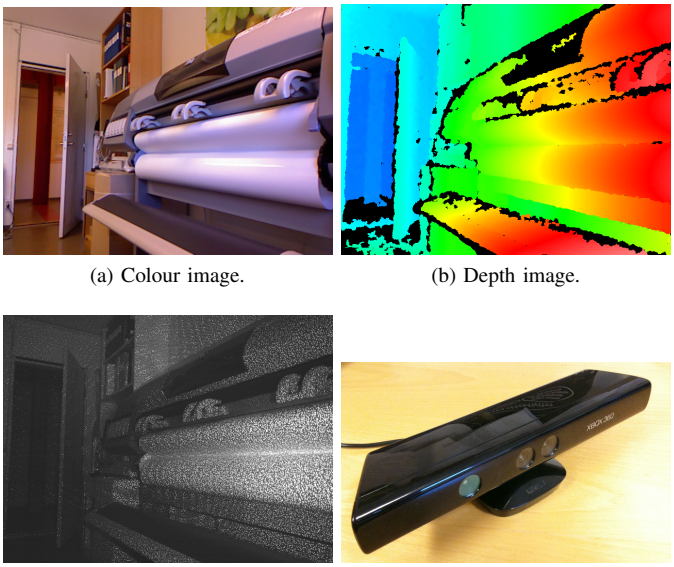
I. Introduction

Video coding using 3D models can reach very low bitrates compared to the traditional block-based approach. The reason for this is that textures need to be transmitted less often, as they are coded separately from the 3D structure and camera motion.

The difficulty with 3D model-based coding lies in extracting unknown 3D structure from the captured scene. Dense structure from motion for a single camera is not yet plug-and-play for arbitrary sequences. Recently, devices that deliver both colour and depth streams, such as the infrared structured light systems developed by Primesense [1] (Microsoft Kinect, and the Asus WAVI sensors are consumer products that use this technology), have made production of dense structure and motion video much easier. See Fig. 1 for an illustration of the Kinect sensor, and its output.

Previously, 3D video coding systems have required multiple sensors, either a colour camera and separate depth camera have been used [2], or multiple colour cameras. As the registration of the video streams require the relative poses of the sensors to be known, such systems must be calibrated before each video capture.

There are other advantages with model-based video coding besides offering low bitrates. The available 3D representation additionally allows for stereoscopic viewing of the video, by having the decoder render two sidewise offset streams. The 3D representation also allows objects to be inserted into the scene for augmented reality, and free viewpoint viewing is also possible [3], i.e. looking at the scene from different, freely selectable points of view.



(a) Colour image. (b) Depth image.

(c) Structured light. (d) Kinect sensor.

Fig. 1. Colour and depth image captured with the Kinect.

A. Related Work

Traditional video coders are *block based*. This means that each video frame is segmented into blocks, which are encoded independently, using still-image coding techniques. Already early video codecs offered the possibility to encode the difference of a block to another block instead of a direct encoding [4]. Coding the difference allows the coder to exploit redundancies between different images of the sequence. In second generation video codecs [5] this redundancy was further exploited, by introducing a 2D segmentation approach. However, the most advanced video codecs like H.264 [6] do not take full advantage of such advanced techniques – instead, they offer a more traditional block-based solution, albeit with the possibility of using different and much finer block sizes.

Model-based or object-based approaches have been tried as well [7]. These exploit knowledge of the 3D structure of scene objects or the imaged scene. Theoretically, by knowing

the geometry of the scene, the video can be encoded with a minimal bitrate since the objects of a scene and their textures have to be encoded only once. The textures, which usually consume most of the bitrate in these videos, can then be encoded using a highly optimized still-image coder like JPEG2000 [8]. However, apart from a few specialised applications like video-telephony [9] these techniques are only seldom used. A more recent work in this area can be found in [10], where the geometry of static scenes are encoded as meshes.

For modern 3D and multi-view videos, several other approaches have been suggested, such as the encoding of several video streams with inter-stream prediction to exploit spatial redundancy [11], and the adding of a depth-image stream [12].

The system presented in this paper does 3D model-based coding, and is most similar to [10] and [3]. Differences compared to [10] and [3] are the use of a combined colour and depth camera in our method, where [10] uses structure from motion and [3] uses multiple colour cameras to capture the depth. Because of the differences in input data, it is difficult to make a fair comparison. Further differences are the representation of the geometry, where [10] use a mesh, whilst our method uses a quad-based approach similar to [3]. The advantage when using a quad-based representation is that the discontinuities in the 3D models do not have to be detected. Moreover, our method for extracting and compressing quads is different from [3] (differences will be highlighted in the related section).

B. Overview of the Proposed Method

The method presented in this paper is split into an encoding and decoding part. The encoding part receives data from colour and depth cameras. Pre-processing of camera data and relationship between the colour and depth cameras are explained in section II. The encoder selects keyframes with a variable number of frames between them. The camera motion between these keyframes is estimated (section III) and 3D models are fitted onto the keyframes (section IV). Camera motion and 3D models for each keyframe are compressed and transmitted to the decoder. The decoder reconstructs the video given these keyframes which is briefly explained in section V.

The results achieved using the proposed method are shown and compared to H.264 in section VI.

II. Colour and Depth Cameras

Combined colour and depth cameras have recently become popular, with the introduction of the Microsoft Kinect (see Fig. 1), and the Asus WAVI sensors. Both of these devices use structured light, and are based on a patented reference implementation by the company Primesense [1].

Full-frame depth sensors that use the time-of-flight principle are also available, and an alternative way to obtain a combined colour and depth camera is thus to place a colour camera next to a time-of-flight sensor [2].

In both structured light, and time-of-flight systems, the colour image sensor and the depth image sensor are separate units, and thus by necessity offset with respect to each other.

The relationship between the colour and depth cameras can be expressed using the pinhole camera model [13]. The origin of the coordinate system is set to the camera centre of the depth camera, yielding a camera matrix

$$\mathbf{C}_{\text{Depth}} = \mathbf{K}_{\text{Depth}}[\mathbf{R}_{\text{Depth}} \mid \mathbf{t}_{\text{Depth}}] = \mathbf{K}_{\text{Depth}}[\mathbf{I} \mid \mathbf{0}] \,. \tag{1}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{0}$ is the null vector and $\mathbf{K}_{\text{Depth}}$ is the depth camera's calibration matrix. The colour camera, with calibration matrix $\mathbf{K}_{\text{RGB}}$, is rotated $\mathbf{R}_{\text{RGB}}$ and translated $\mathbf{t}_{\text{RGB}}$ relative to the depth camera

$$\mathbf{C}_{\text{RGB}} = \mathbf{K}_{\text{RGB}}[\mathbf{R}_{\text{RGB}} \mid \mathbf{t}_{\text{RGB}}] \,. \tag{2}$$

A point $\mathbf{x}$ in the depth image with homogeneous coordinates $\mathbf{x}_h$, can be reprojected as a line $\mathcal{L}$ through the origin out into the world [13]

$$\mathcal{L} = \{\mathbf{x} : \mathbf{x} = \gamma \mathbf{K}_{\text{Depth}}^{-1} \mathbf{x}_h, \gamma \in \mathbb{R}\} \,. \tag{3}$$

The value in the depth image z for point $\mathbf{x}$, $z(\mathbf{x})$, then tells us where on the line the point is

$$\mathbf{y} = z(\mathbf{x}) \mathbf{K}_{\text{Depth}}^{-1} \mathbf{x}_h \,. \tag{4}$$

The point in the world $\mathbf{y}$, with the homogeneous coordinates $\mathbf{y}_h$, is then projected onto the colour camera

$$\mathbf{x}_{\text{RGB}} = \mathbf{C}_{\text{RGB}} \mathbf{y}_h \,. \tag{5}$$

Note that this transfer of the depth map to the colour image requires the camera matrices to be known. A calibration procedure that finds them is described in [14]. Furthermore, note that this calibration needs only to be performed once for each device, and that it is even stored on the device itself, and is accessible using the OpenNI drivers [15].

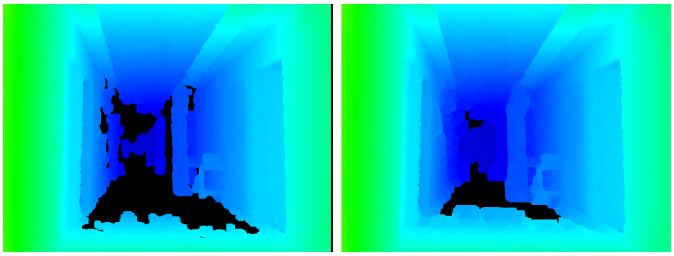
A. Depth Filtering

The depth image can contain a significant amount of missing data. Data may be missing in several circumstances, such as when the object is too close or far away. Other circumstances when data may be missing in are if surfaces are reflective, or when structured light cannot be projected and deciphered, due to the angle of the surface being too narrow relative to the camera. Since the infrared projector and camera are not in the same position, occlusion will also cause missing data.

To improve the motion estimation and 3D model, the depth image is filtered. The missing data is first interpolated with a 2D isotropic Gaussian filter

$$z(\mathbf{x}) = \frac{\sum\limits_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma_p^2}\right) z(\mathbf{x}_i) c(\mathbf{x}_i)}{\sum\limits_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma_p^2}\right) c(\mathbf{x}_i)} \tag{6}$$

where $\mathcal{N}(\mathbf{x})$ is the neighbourhood of point $\mathbf{x}$ and c is a mask that is 0 if the data is missing in that point, otherwise 1. A small $\mathcal{N}$ is preferred, so that only a small local neighbourhood is used when filling in missing data. Since the first iteration will not always fill in all missing data, the process is repeated



(a) Input depth image.

(b) After three iterations using a Gaussian filter.

(c) Gaussian filtered depth image.

(d) Bilateral filter applied to the Gaussian filtered image.

Fig. 2. Filling in depth data using Gaussian and bilateral filters.

until all missing data is filled in (Fig. 2c). Then, a bilateral filter [16] is used to smoothen out this filtered missing data, whilst still preserving the edges (Fig. 2d).

$$z(\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \exp\left(-\frac{(z(\mathbf{x}) - z(\mathbf{x}_i))^2}{2\sigma_d^2}\right) z(\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \exp\left(-\frac{(z(\mathbf{x}) - z(\mathbf{x}_i))^2}{2\sigma_d^2}\right)} \tag{7}$$

This operation is only run on missing data points, i.e. when $c(\mathbf{x}) = 0$.

III. Camera Motion Estimation

The rotation and translation between two keyframes are estimated by finding corresponding features in both keyframes' colour images. Features in an image are found by using the Harris corner detector [17]. A patch in an image represents a corner if both eigenvalues of the patch's structure tensor, $\mathbf{A}$, are large.

$$\mathbf{A}(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \nabla I(\mathbf{x}_i)(\nabla I(\mathbf{x}_i))^\top \tag{8}$$

Correspondences between features in two images are then estimated using the OpenCV [18] implementation of the Lucas-Kanade optical flow algorithm [19].

The 3D coordinate for each feature is determined by finding the 3D point whose projection in the colour image is the closest to the feature (Euclidean distance).

All the features' 3D coordinates belonging to one image are centred around their own origin. The rotation between two sets of corresponding features can then be estimated by solving the orthogonal Procrustes problem [20]. If $(3 \times n)$ matrix $\mathbf{A}$ contains the 3D points from the first image in its columns, and $(3 \times n)$ matrix $\mathbf{B}$ contains the corresponding 3D points from the second image, the orthogonal Procrustes algorithm finds the transformation, $\mathbf{R}$, that minimises

$$\epsilon = \|\mathbf{A} - \mathbf{R}\,\mathbf{B}\|^2. \tag{9}$$

This can be solved by using singular value decomposition

$$\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathtt{SVD}(\mathbf{B}\mathbf{A}^\top) \tag{10}$$

where rotation $\mathbf{R}$ is given by

$$\mathbf{R} = \mathbf{V}\mathbf{U}^\top. \tag{11}$$

Data from the depth camera can contain noisy points (outliers) that will reduce the quality of the transformation. RANSAC [21] is used to eliminate outliers and only using the less noisy points (inliers) when estimating the transformation.

The full transformation chain (in homogeneous coordinates) from 3D points in the second image to 3D points in the first image is then given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & \mathbf{t}_A \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{t}_B \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{12}$$

where $\mathbf{I}$ is a 3×3 identity matrix, $\mathbf{0}$ is a 3-element null vector, $\mathbf{t}_A$ and $\mathbf{t}_B$ are the translations used to centre the points of the first and second images around their origin.

IV. Scene Geometry Representation

The geometry used to represent the scene for each keyframe is 3D *quadrilaterals* (denoted quads in the following). Each 3D quad is made up of four vertices describing its world and texture coordinates.

A quadtree of a pre-defined depth is first fitted onto the keyframe's colour image. This quadtree is then tessellated to allow for large quads on large flat surfaces, and smaller quads where there are high details. For each quad in this quadtree, the 3D midpoint and average normal are calculated from the 3D points that are projected onto the quad. Given the quad midpoint $\mathbf{p}$ and its average normal $\hat{\mathbf{n}}$, the distance, D, from the origin to the plane of a quad can be calculated with the plane equation

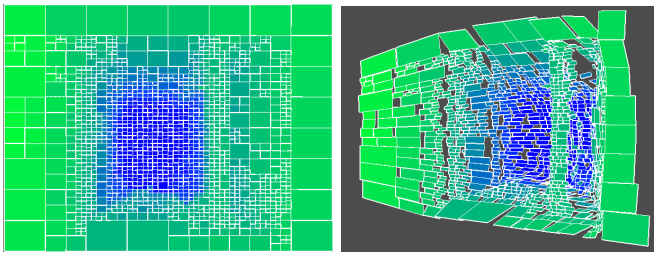
$$\mathbf{p} \cdot \hat{\mathbf{n}} + D = 0\,. \tag{13}$$

The distance from the plane is then measured for every 3D point that projects onto the plane's quad

$$N_\epsilon = \sum_{i=1}^{N} \begin{cases} 1, & |\mathbf{p}_i \cdot \hat{\mathbf{n}} + D| > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

If the number of points, N_ϵ, that are further away than ϵ from the plane is large, then the quad is split into four new quads. Otherwise the quad is marked for having the possibility to merge with its parent's other children. If all four children belonging to a parent are marked as being able to merge, then the plane normal in each one of the children is checked to see if they are close to being parallel.

$$\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \geq t_n, \;\; i, j \in \{1, 2, 3, 4\}, \; i \neq j \tag{15}$$



(a) The colour camera's point of view. (b) Seen from the side.

Fig. 3. Geometry generated from a quadtree seen from two different views. Frame is from the same video sequence as Fig. 2.

where t_n is a threshold close to 1. If all four normals are close to being parallel, then their parent is merged into a child. An example of a tessellated quadtree can be seen in Fig. 3.

The method presented in [3] fits a quadtree on the depth image which is in the same coordinate system as the colour image. Their quadtree is then tessellated if the depth difference between two pixels is above a specified threshold. Our method fits the quadtree on the colour image and tessellates it given the 3D points' distance from its plane.

V. Compression and Decoding

For each keyframe the encoder transmits the relative camera movement to the previous keyframe, the 3D model and its texture. Instead of sending all the vertices for the 3D model, the quadtree structure and each quad's plane equation is transmitted. To further reduce the number of bits, the floating point numbers used to represent the plane equations are converted to integers and compressed using a fixed-length Huffman code. The 3D model's texture (the camera colour image) is compressed using JPEG2000 [8].

Compressing and transmitting the plane equation is another difference compared to [3]. In their method the depth of each vertex is compressed. Our method compresses the plane equation instead because the normal is less sensitive to quantization errors.

Given the quadtree structure and the plane equations, the decoder can reconstruct the keyframe's 3D model. Frames in-between the keyframes are predicted by interpolating the camera movement with SLERP [22] for the rotation and linear interpolation for the translation. The current and next keyframes' 3D models are projected onto the predicted frame's colour image in two separate images. These two images are then linearly blended together to create a smooth transition between keyframes.

VI. Performance Evaluation

The performance of the proposed method has been evaluated against the H.264 standard using the H.264/AVC JM reference software [23] at low bitrates. In the results shown here the target bitrate is set to 270 kbit/s.

Unlike H.264, our method contains information about the depth of the scene. To be able to get this information using H.264 a separate stream needs to be encoded. This extra stream would need to reduce the video's visual quality to maintain the same bitrate. However, an extra depth stream has not been encoded in these experiments.

(a) Frame #0. (b) Frame #120.

(c) Frame #220. (d) Frame #359.

Fig. 4. Frames from the test sequence.

There are several parameters that can be changed in the proposed method to impact the quality and size of the video. The parameters that have the largest impact are texture quality and the numbers of bits the plane equations are encoded with.

In the test sequence used, the camera moves along a corridor. The camera mostly moves along the z axis, and occasionally rotates. There is a significant amount of missing depth data in this sequence due to reflections and the length of the corridor. Selected frames from the uncompressed test sequence can be seen in Fig. 4.

Two different quality measures have been used to evaluate the results. Using the common *peak signal-to-noise ratio* (PSNR) shows that the proposed method peaks at each keyframe (as seen in Fig. 5), and has its lowest points between each keyframe. Putting keyframes closer together (having more keyframes) increases the PSNR between them, but also increases the total bitrate. The texture quality can be increased when using sparse keyframes (blue line), but has to be decreased when using more dense keyframes (green line), in order to obtain the same bitrate.

The PSNR error measure does not often correspond to how the human eye perceives the quality of an image. The *Structural SIMilarity* (SSIM) index [24] is a more appropriate method for evaluating visual quality perceived by the human eye. Rather than evaluating the error pixel-by-pixel as PSNR does, the SSIM looks at the structure of a pixel's neighbourhood to locally adjust the contrast and luminance. Fig. 6 shows the SSIM for the proposed method and H.264.

The errors for the two coding methods are very different. Details are lost when using H.264 at low bitrates. In Fig. 7 the visual quality of H.264 compared to our method can be seen. Our proposed method maintains the details, but has two

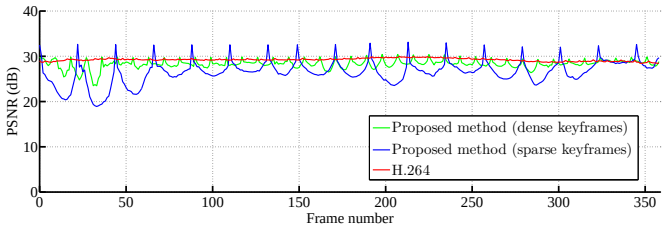


Fig. 5. Comparison of PSNR between the proposed method and H.264.

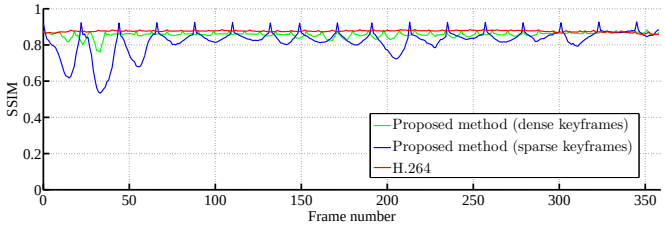
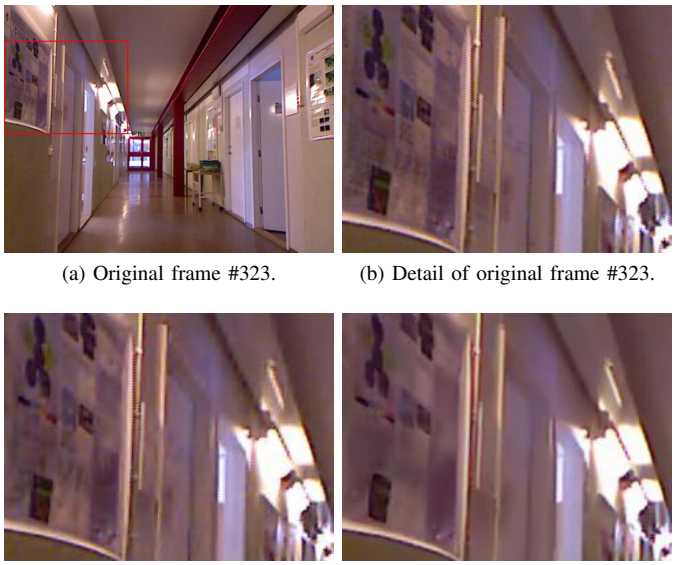


Fig. 6. Comparison of SSIM between the proposed method and H.264.



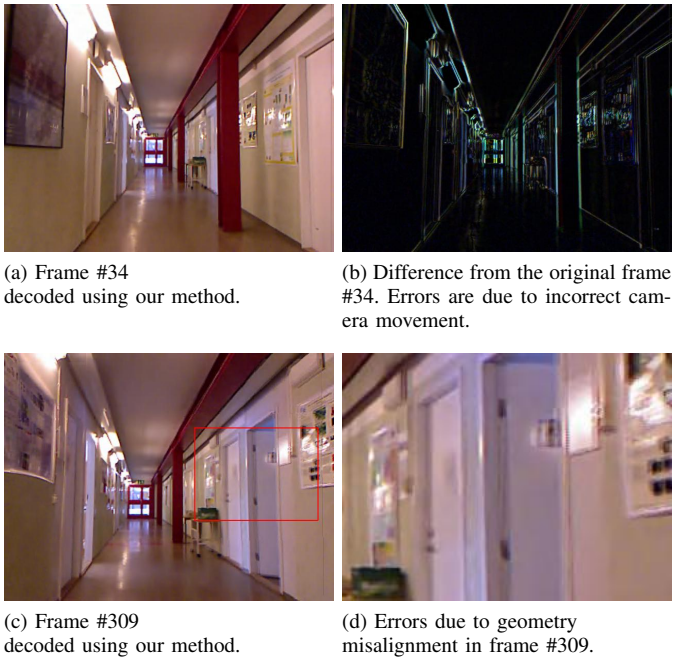
(a) Original frame #323. (b) Detail of original frame #323.

(c) Detail of frame #323 decoded using our method. (d) Detail of frame #323 decoded using H.264.

Fig. 7. Visual quality comparison between our method and H.264.

other types of artifacts. The first three dips in the PSNR curve (Fig. 5) are due to errors in the camera movement. For example, frame #34 that has the lowest PSNR of all the frames looks good (Fig. 8a), but the movement is incorrect which reduces its PSNR value. The difference from the original image can be seen in Fig. 8b. Later in the sequence the errors are caused by imperfections in the geometry. Because of this misalignment, the blending generates "ghosting effects" as seen in Fig. 8d.

The bigger drops in PSNR for the sparse keyframes setting in Figs. 5 and 6 are mainly caused by small camera pose errors introduced by the linear interpolation of the camera motion.



(a) Frame #34 decoded using our method. (b) Difference from the original frame #34. Errors are due to incorrect camera movement.

(c) Frame #309 decoded using our method. (d) Errors due to geometry misalignment in frame #309.

Fig. 8. Two types of errors when using the proposed method.

VII. Concluding Remarks

We have presented a method for compressing video using a model-based approach given input from colour and depth cameras. Our method encodes both the colour and depth streams, but it has been compared to H.264 which encodes only the colour stream.

The results show that the proposed method can achieve a video with higher detail level compared to H.264. Even though the proposed method does not reach the same average PSNR as H.264, one has to consider that the encoded data stream also contains depth information about the scene. This information can be used for other applications such as stereoscopic viewing of the video, object insertion for augmented reality and free viewpoint viewing.

We have also shown that the artifacts in block-based and model-based coding are different. In block-based coding details in texture are lost, whereas in model-based coding the texture retains more detail, but is geometrically distorted. Because both PSNR and SSIM measure the error pixel-by-pixel, geometric distortions are treated as more severe.

Several details of the proposed method can be improved upon. The proposed method uses a computer vision based approach to estimate the camera motion. A large reduction of the PSNR is because of an incorrect camera trajectory. This is partly due to the linear approximation of the camera motion, but a more sophisticated method for estimating the camera motion could also lead to improvements.

The currently used extrapolation of missing data in the depth stream is quite crude. Better results could possibly be obtained by using locally planar assumptions in the extrapolation.

Representing the geometry using quads instead of a mesh removes the problem of finding discontinuities and allows for a

compact data representation. To further improve the proposed method, it would be of interest to allow for non-static scenes. Giving each quad a motion vector would allow objects to move within the scene.

What is shown in this paper is that although model-based coding has its shortcomings, it also has great potential, especially with the growing prevalence of 3D applications.

ACKNOWLEDGMENT

Prof. Robert Forchheimer (from Linköping University) is gratefully acknowledged for fruitful discussions. This work is supported by the Swedish Research Council through a grant for the project *Embodied Visual Object Recognition*, and by Linköping University.

REFERENCES

[1] Primesense reference platform
http://www.primesense.com/files/FMF_2.PDF.

[2] M. N. Do, Q. H. Nguyen, H. T. Nguyen, D. Kubacki, and S. J. Patel, "Immersive visual communication," in *IEEE Signal Processing Magazine*, January 2011, pp. 58–66.

[3] T. Colleu, S. Pateux, L. Morin, and C. Labit, "A polygon soup representation for multiview coding," *J. Vis. Comun. Image Represent.*, vol. 21, pp. 561–576, July 2010.

[4] T. Sikora, "MPEG digital video-coding standards," *IEEE Signal Processing Magazine*, vol. 14, no. 5, pp. 82–100, 1997.

[5] L. Torres and M. Kunt, *Video Coding: The Second Generation Approach*. Kluwer Academic, 1996, ISBN: 0792396804.

[6] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with h.264/avc: tools, performance, and complexity," *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, 2004.

[7] D. Pearson, "Developments in model-based video coding," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 892–906, June 1995.

[8] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *Consumer Electronics, IEEE Transactions on*, vol. 46, no. 4, pp. 1103–1127, November 2000.

[9] I. S. Pandzic and R. Forchheimer, *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. Wiley, 2002, ISBN: 0470844655.

[10] F. Galpin, R. Balter, L. Morin, and K. Deguchi, "3D Models Coding and Morphing for Efficient Video Compression," in *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR'2004*, Washington DC, USA, June 2004, pp. 334–341.

[11] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 11, pp. 1461–1473, November 2007.

[12] C. Fehn, "A 3D-TV system based on video plus depth information," in *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, vol. 2, November 2003, pp. 1529–1533.

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, ISBN: 0521540518.

[14] M. Wallenberg, M. Felsberg, P.-E. Forssén, and B. Dellen, "Channel coding for joint colour and depth segmentation," in *Proceedings of DAGM'11*, Frankfurt, Germany, 2011.

[15] Primesense Open NI API
http://www.openni.org/.

[16] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, 1998, pp. 839–846.

[17] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[18] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," Intel Corporation., Tech. Rep., 2000.

[19] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI81*, 1981, pp. 674–679.

[20] P. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, March 1966.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[22] K. Shoemake, "Animating rotation with quaternion curves," in *Int. Conf. on CGIT*, 1985, pp. 245–254.

[23] K. Sühring, "H.264/AVC JM reference software," http://iphome.hhi.de/suehring/tml/.

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.