EXPERIMENTAL STUDY ON SCAN ORDER AND MOTION COMPENSATION IN LOSSLESS VIDEO CODING

Stefan A. Kramatsch[1,2], Herbert Stögner[2], and Andreas Uhl[2,3]

[2]School of Telematics & Network Engineering, Carinthia Tech Institute, Austria

[3]Department of Scientific Computing, Salzburg University, Austria

Abstract: *Video data is not necessarily interpreted as a sequence of frames ordered in time. We investigate different strategies how to scan and order the data with respect to their respective suitedness for lossless video compression. Video coding schemes with low time and space complexity competitive to more complex techniques may be derived from our results.*

Keywords: *lossless video coding, medical imaging, block matching, scan order.*

1 INTRODUCTION

The majority of lossy video coding algorithms is based on motion compensated hybrid coding techniques like MPEG-1, MPEG-2, MPEG-4, and H.261 – H.264 [1, 2]. These video codecs exploit the temporal redundancies present in video data (i.e. the similarity of frames over time) by applying algorithms for motion estamation and compensation. In most cases, block matching techniques [3] are used. The "remaining" residual frames are encoded similar to still images thereby exploiting spatial correlations still present in the data.

Medical imaging is the main application field for lossless video coding. In these applications, most techniques employ lossless image coding techniques like lossless JPEG, JPEG-LS, or lossless JPEG 2000 on a per-frame basis. Of course, temporal redundancy is ignored in such schemes which results in limited compression performance. On the other hand, three dimensional transforms (e.g. 3D DWT [4]) are used to decorrelate the video data but these techniques suffer from high complexity and high memory requirements. Motion compensation techniques as employed in lossy schemes are hardly used in lossless environments.

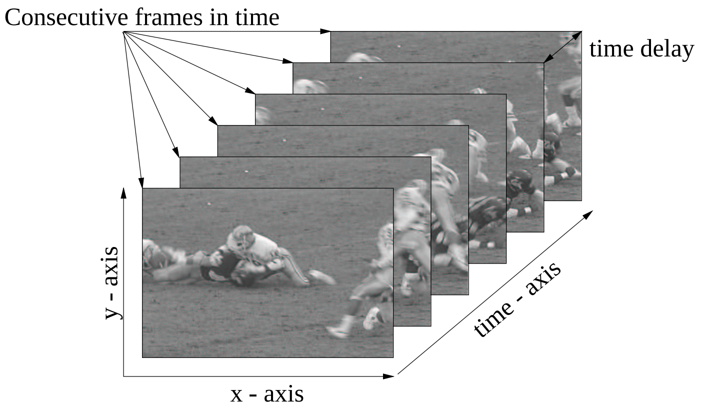
In this work, we view and process video data in a different manner as compared to classical approaches. We interpret a video as a 3D block of data which can be viewed and processed in any arbitrary order, in particular we do not consider the temporal direction as being necessarily of special nature. In section 2, we discuss and visualize various techniques how to scan and process video data. Section 3 provides corresponding experimental compression results using JPEG 2000 and generic one-dimensional lossless coding algorithms in highly non-standard fashion. Section 4 concludes this paper.

2 LOSSLESS VIDEO CODING

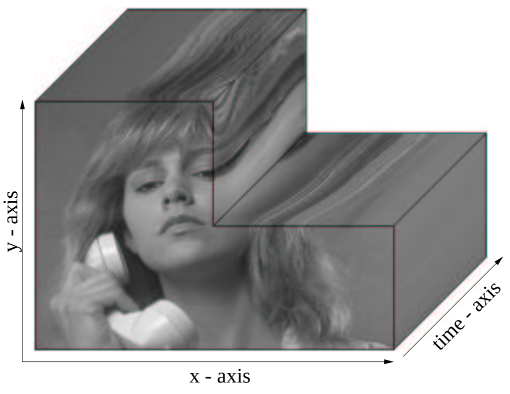
The classical view of video data is depicted in Fig. 1.a. The video consists of a set of spatial frames which are temporally ordered still images. As can be seen from the figure, these frames are very similar, even in the presence of strong motion. The aim of inter-frame coding techniques is to exploit this similarity, either by employing block-matching motion compensation among frames (motion compensated hybrid coding) or by applying a transform

[1]This artificial name represents a group of students working on this paper in the framework of the Multimedia 1 lab in winterterm 2004/2005: Agnes Gruber, Alexander Krapesch, Stefan Matschitsch, Thomas Mayerdorfer, Stefan Miedl, Stefan Moser, Martin Tschinder, and Stefan Zorn-Pauli.

stage in the temporal direction as well (3D techniques). This latter 3D interpretation of video is shown in Fig. 1.b where the frames are accumulated to form a three dimensional (3D) data block of visual data.



(a) Sequence of frames



(b) 3D block

Fig. 1 Different views of video data.

This block may be cut along different directions as shown in the figure providing interesting views of the visual and temporal content of the video. In the following, we consider the video as being given in this accumulated 3D manner.

2.1 Non-standard Frame Structure

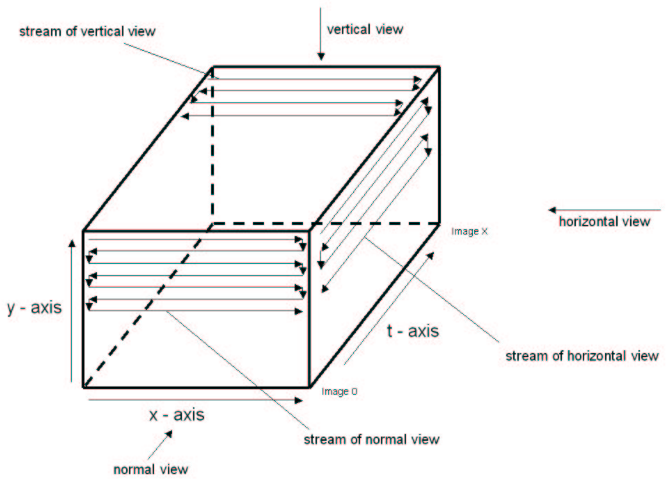
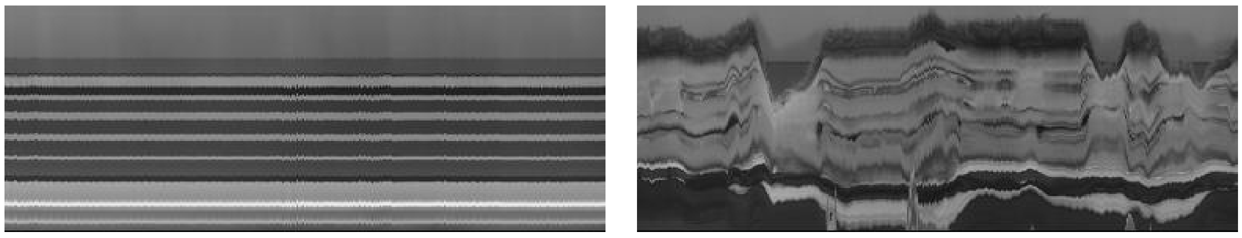


Fig. 2 Different scans through video data.

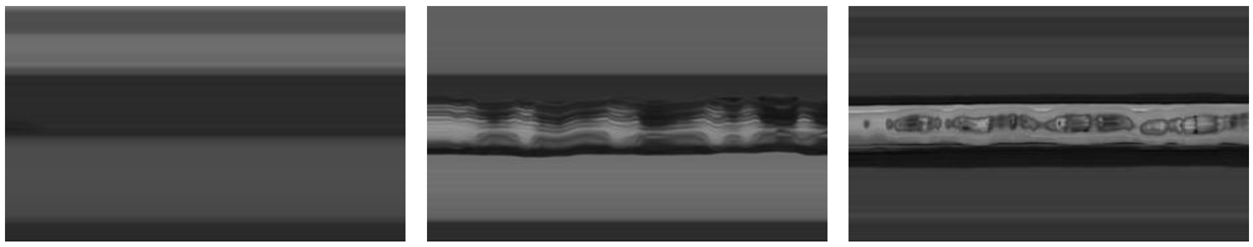
Given the 3D block of video data, the classical way to view or process it is orthogonal to the spatial plane in temporal direction (labelled as "normal view" in Fig. 2). However, different ways to view or process the data are possible. The "horizontal view" takes a sideview of the video data as shown in Fig. 2. Using this approach we result in "horizontal frames" which have the original image height and the number of frames as their respective spatial dimensions. Stepping through the video composed of horizontal frames, the adjacent frames correspond to spatially adjacent slices consisting of temporally translated columns of the original frames. Therefore, the first horizontal frame is just the side-plane of the 3D video cube. In Fig. 3 we visualize two types of horizontal frames: the third horizontal frame (Fig. 3.a) is close to the edge of the video and consists mostly of background columns which do hardly change over time which results in almost straight lines of the border of the regions. On the other hand, the horizontal frame 095 consists of columns at the center of the original frames which frequently change in time. Fig. 3.b displays this frame which shows the movements of parts of the head giving an interesting visual impression.



(a) horizontal frame 003 (b) horizontal frame 095

Fig. 3 "Carphone" video: 176 x 144 x 383 pixels

The "vertical view" looks at the video data from above as shown in Fig. 2. Using this approach we result in "vertical frames" which have the original image width and the number of frames as their respective spatial dimensions. Stepping through the video composed of vertical frames, the adjacent frames correspond to spatially adjacent slices consisting of temporally translated lines of the original frames. Therefore, the first vertical frame is just the upper plane of the 3D video cube.



(a) vertical frame 017 (b) vertical frame 043 (c) vertical frame 084

Fig. 4 "Akiyo" video: 176 x 144 x 300 pixels

Whereas the first frame consists of lines close to the upper edge of the original frames (Fig. 4.a) which do not change over time which results in an extremely smooth frame, the second (Fig. 4.b) and third (Fig. 4.c) frames consist of lines from the head and neck area, respectively which change in a rapid manner due to movements of the speaker. Therefore, only the side areas of these two frames are as smooth as shown in Fig. 4.a since only in these areas the lines stem from non-moving background areas.

2.2 Block-Matching Displacement Compensation

In block-matching motion compensation [3], the scene (i.e. video frame) is classically divided into non-overlapping "block" regions. For estimating the motion, each block in the current frame is compared against the blocks in the search area in the reference frame and the motion vector corresponding to the best match is returned. The "best" match of the blocks is identified to be that match giving the minimum mean square error (MSE) of all blocks in search area. The block giving the minimal error is stored describing the prediction in term of a motion vectors which describes the displacement of the block between current and reference frame. The frame consisting of the predictions for all blocks in the current frames (the predicted frame) is subtracted from the current frame resulting in the residual frame, which has to be compressed.

The idea of block-matching motion compensation may be generalized to block-matching displacement compensation. Differences among frames do not necessarily need to be caused by motion in general. For example, when applying block-matching to the horizontal and vertical frames defined in the previous section, the differences among frames are due to spatial displacement of the frames in either horizontal or vertical direction.

Fig. 5 visualizes residual frames when block matching is applied to the respective vertical frames of the video. Fig. 5.a reflects the motion of the upper parts of the salesmans' head, whereas Fig. 5.b shows the movements of the hands in the central part of the video. As can be seen, block matching does no good job on the fast moving hands.

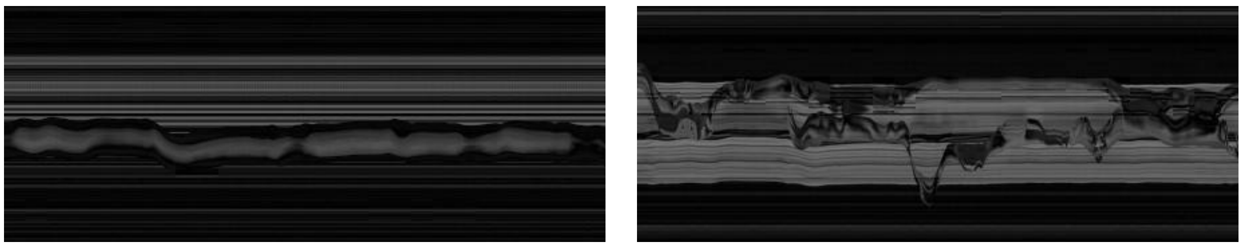

(a) vertical frame 028 (b) vertical frame 070

Fig. 5 "Salesman" video with vertical block-matching.

3 EXPERIMENTS

Our aim is to exploit the alternative perspectives of video data as discussed in the last section for lossless video compression. We use the following testvideos (and give their spatial and temporal resolutions): Akiyo (176 x 144 x 300), Carphone (176 x 144 x 383), Claire (176 x 144 x 494), Football (720 x 486 x 60), Foreman (176 x 144 x 49), Grandma (176 x 144 x 871), Mobile (720 x 576 x 40), Mother and Daughter (176 x 144 x 962), and Salesman (176 x 144 x 449). Five out of these nine videos exhibit rather low motion content (Akiyo, Claire, Grandma, Mother and Daughter, and Salesman), the remaining four videos are characterized by a large amount of movements (Carphone, Football, Foreman, Mobile).

3.1 Experimental Settings

We use two different ways to compress the data: the first way is to compress the data frame-by frame using JPEG 2000 (the JJ2000 JAVA implementation[2] is used for this purpose). The second way employs generic one-dimensional compression engines. For that purpose, the video data is scanned as shown in Fig. 2 on a frame basis – when a frame is finished, the scan of the subsequent frame is started at the pixel position identical to the last pixel scanned thereby reversing the scan order in each frame but guaranteeing high correlation of the data. The resulting stream of pixel values is than compressed using publicly available implementations of Huffman Coding[3], Runlength Encoding[4], and Arithmetic Coding[5].

[2]http://jj2000.epfl.ch/
[3]http://www.cjkware.com/wamckee/huffman.zip
[4]http://www.datacompression.info/RLE.shtml
[5]http://michael.dipperstein.com/arithmetic/index.html

These techniques may be applied directly to the temporal (i.e. “normal” in Table 6), horizontal, and vertical frames of the videos as they are generated. As a second option, we apply block-matching (bm) displacement compensation to temporal, horizontal, and vertical frames and compress the reference (first) frame and all subsequent residual frames (which means that block-matching is applied relative to the first frame for all remaining frames). Block-matching uses 16×16 pixels blocks and a search area of 32×32 pixels as search area in the reference frame.

3.2 Results

The following Table lists compression ratios for JPEG 2000 and arithmetic compression. Note that Huffman coding gives the same results as arithmetic coding reduced only by a small constant and therefore no values are shown. Runlength coding gives very poor compression results in general (a majority of results is below 1.0). The general trend is similar to arithmetic coding as well and we therefore refrain from listing results explicitely. Also note that due to the adaptiveness of Huffman and arithmetic coding the streams resulting from the normal, horizontal, and vertical view (without block-matching applied) result in the same compression ratios. Therefore only values for the normal view are given.

Fig. 6 Compression ratios for JPEG 2000 and arithmetic compression.

mode \\ video	Akiyo	Carp.	Claire	Foot.	Fore.	Grand.	Mob.	Moth.	Sale.
JPEG 2000									
normal	2,182	1,877	2,727	1,747	1,632	1,855	1,632	1,865	1,643
vertical	6,283	2,122	3,908	1,647	1,726	3,109	1,560	2,769	3,206
horizontal	6,409	2,046	4,080	1,433	1,644	3,081	1,518	2,811	3,054
normal bm	3,712	1,755	2,749	1,594	1,609	2,308	1,606	1,921	2,253
vertical bm	6,106	1,864	2,715	1,670	1,357	2,780	1,508	2,477	2,632
horizontal bm	5,969	1,917	4,083	1,291	1,582	2,683	1,371	2,474	2,628
Arithmetic coding									
normal	1,118	1,095	1,263	1,138	1,094	1,175	1,121	1,144	1,184
normal bm	3,117	1,541	2,391	1,321	1,569	2,424	1,510	1,906	2,205
vertical bm	1,185	1,204	1,405	1,275	1,173	1,413	1,121	1,290	1,299
horizontal bm	1,278	1,100	1,279	1,279	1,086	1,317	1,299	1,250	1,337

First we discuss JPEG 2000 based compression. For low motion video significantly higher compression ratios are obtained for horizontal and vertical frames as compared to normal frames, block-matching applied to normal frames improves the result of normal frames but the results stay below those of horizontal and vertical frames. Finally, block-matching applied to horizontal and vertical frames improves normal blockmatching but does not reach either the results of pure vertical and horizontal frame coding. For high motion video the situation changes significantly: for vertical and horizontal frame coding the compression ratios are not clearly improved and sometimes even decreased. Block-matching applied to any type of frames decreases the compression result.

In the case of arithmetic coding we notice that normal block-matching improves the result for all videos, the improvements are more significant for low motion video. Block-matching applied to vertical and horizontal frames still improves the classical “normal frame” result, but with lesser significance as compared to classical block-matching.

Based on the visual impression displayed in Figs. 3.a and 4 the better compression behaviour for horizontal and vertical frames is expected due to the high degree of smoothness present in these images. On the other hand, frames with highly non-stationary content like that shown in Fig. 3.b suggest that the compression ratios might vary a lot in the set of herizontal or vertical frames.

This is confirmed by the result shown in Fig. 7 where we plot the compression ratio on a frame basis. Whereas the values for normal frames are almost constant horizontal and vertical frames exhibit varying values with maxima at the first and last frames, respectively. This result corresponds well to the visual impression where we notice smooth frames at the edges of the data (note that this result employs sligthly different experimental settings but the results do carry over to our case).

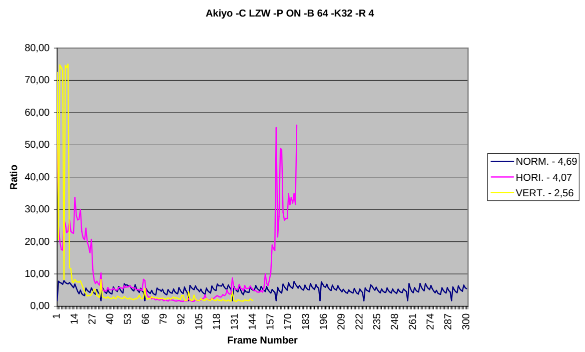


Fig. 7 Compression ratios of the different frame types of the Akiyo testvideo.

4 CONCLUSION

We have found that alternative interpretation and scan order leads to improved frame based (2D) compression results without the application of cost-intensive block-matching algorithms. This means that we are able to exploit temporal redundancies more efficiently with respect to compression and less expensive with respect to computational demand as compared to block-matching using horizontal and vertical frames. Of course, our approach is much less demanding as compared to the use of 3D transforms and allows the use of standardized lossless still image codecs. For a 1D (stream-based) setting as discussed, block matching still gives superior results in terms of compression ratio.

ACKNOWLEDGEMENTS

We thank Albert Meixner and Alexander Meschtscherjakov for providing the result displayed in Fig. 7.

REFERENCES

[1] B. Haskell, A. Puri, and A. Netravali, *Digital video: an introduction to MPEG-2.* Digital Multimedia Standards Series, Chapman & Hall, 1997.

[2] I. E. G. Richardson, *H.264 and MPEG-4 video compression: video coding for next generation multimedia.* Wiley & Sons, 2003.

[3] B. Furht, J. Greenberg, and R. Westwater, *Motion estimation algorithms for video compression.* Norwell, MA, USA, and Dordrecht, The Netherlands: Kluwer Academic Publishers Group, 1997.

[4] S. Cho, D. Kim, and W. A. Pearlman, Lossless compression of volumetric medical images with improved 3-D SPIHT algorithm, *Journal of Digital Imaging*, vol. 17, no. 1, pp. 57–63, 2004.