Fast Frame-rate Up-conversion of Depth Video via Video Coding *

Yanjie Li, Lifeng Sun
Information Science and Technology Department
Tsinghua University, Beijing, 100084
carol.lyj@gmail.com,sunlf@tsinghua.edu.cn

Tianfan Xue
Department of Information Engineering
The Chinese University of HongKong
xtf009@ie.cuhk.edu.hk

ABSTRACT

Recent development of depth sensors has facilitated the progress of 2D-plus-depth methods for 3D video representation, for which frame-rate up-conversion (FRUC) of depth video is a critical step. However, due to the computational cost of state-of-the-art FRUC methods, real time applications of 2D-plus-depth is still limited. In this paper, we present a method of speeding up the FRUC of the depth video by treating it as part of a video coding process, combined with a novel color-mapping algorithm is adopted to improve the quality of temporal upsampling. Experiments show that the proposed systems saves up to 99.5% of the frame interpolation time, while achieving virtually identical reconstructed depth video as state-of-the-art methods.

Categories and Subject Descriptors

I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms

Design, Performance

Keywords

Depth video, Frame-Rate Up-Conversion (FRUC), Video coding, Motion-Compensated Frame Interpolation (MCFI)

1. INTRODUCTION

Depth video representing the relative distance of objects to the video camera are useful in many multimedia applications, especially in 3DTV. The recent development of depth sensors for capturing depth maps has drawn great interests to 2D-plus-depth methods [4]. However, due to the computational and physical complexities, most methods for capturing depth video, such as stereoscopic and range sensors based systems, can only provide information at a low frame-rate, which severely limits the applications of 2D-plus-depth. Therefore, temporal upsampling of depth videos, also known as frame-rate up-conversion (FRUC), has become an emerging research area [1, 2, 3, 5].

State-of-the-art researches utilize motion-compensation assisted block-based frame interpolation (MCFI) [2, 5, 6] for FRUC of depth video. Conventional algorithms conduct motion estimation using the depth video itself, and are therefore unreliable because depth video lacks texture for conducting accurate motion estimation. Recently, Choi et al. exploited the idea of sharing motion information from 2D video data in [2], and significantly improves the interpolation results.

Although improvement of visual quality has been achieved by applying MCFI to FRUC, a major problem of all current MCFI based methods is the time cost, especially for high-resolution fast moving 2D videos, as exhaustive search of motion directions by calculating pairwise similarities between blocks is usually necessary.

In this paper, we propose a highly efficient FRUC system, by re-using motion information extracted from the 2D video that is available in many applications. At the encoding side, motion information of 2D blocks is estimated and stored in the bitstream, and can be used directly by the decoder, thereby by-passing the majority of the time consuming motion estimation during FRUC. As motion estimation is an integral part of most video coding algorithms, our proposed algorithm introduces virtually no additional cost. Experimental results demonstrate that our method reaches state-of-the-art performance and cuts 99.5% processing time of Choi et. al.'s method.

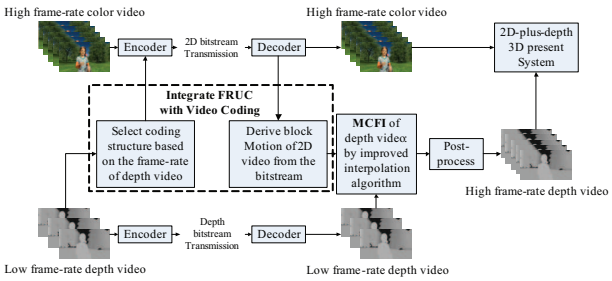

Figure 1: Proposed FRUC framework

Figure 1 illustrates the proposed algorithm on a high level. Although the H.264 AVC video coding standard was used in the experiments reported in this paper, other codecs are also applicable. On the encoder side, the coding structure selected for 2D video depends on the frame-rate of the depth

*Area chair: Pal Halvorsen

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.

video, so that the interpolated depth frames correspond to B-frames in 2D video. On the decoder side, motion information of 2D video derived from decoder is directly used for MCFI, which greatly reduces the time cost for motion estimation. In addition, we also propose an improved interpolation algorithm, namely the color-mapping algorithm, for MCFI, leading to both subjective and objective quality improvements.

The paper is organized as follows. Section 2 gives detailed description of the proposed algorithm, with experimental results and analysis given in Section 3. Finally, Section 4 contains the conclusion.

2. PROPOSED FRAMEWORK

2.1 Integration FRUC with video coding

2.1.1 2D video coding structure selection

In performing 2D video coding, our algorithm selects a coding structure so that interpolated depth frames correspond to non-reference B-frames, whereas captured depth frames correspond to I-frames or P-frames in the 2D video. As an example, if we capture one depth frame for every three depth frames, we would choose the coding structure shown in Figure 2, so that captured (and therefore deemed more reliable) information from two temporally neighboring frames are used to represented an interpolated frame, leading to improved quality. Furthermore, the interpolated depth frames corresponding to non-reference B frames are not used for the construction of other frames, so as to reduce interpolation error propagation.

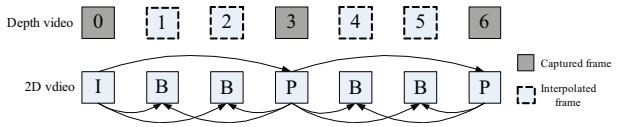

Figure 2: Example of coding structure selection

2.1.2 Motion information derivation from decoder

H.264/AVC standard divides a frame into blocks of different size and each block can be intra-predicted or inter-predicted. Block of size 4×4 is the smallest block used in H.264/AVC standard, and larger blocks can be viewed as multiple 4×4 blocks with the same prediction mode. In this paper, we choose block of size 4×4 as *basic-block* for interpolation, and use coordinate of its top-left pixel to tab a basic-block. For example, basic-block $B(x, y)$ denotes the 4×4 block with top-left pixel (x, y), and pixel $(i, j), i, j \in [0, 3]$ in $B(x, y)$ denotes the pixel $(x + i, y + j)$ in the frame.

The proposed framework divides the interpolated depth frame into basic-blocks and applies MCFI to them using the motion information of corresponding blocks in 2D map derived by H.264/AVC decoder.

Basic-blocks in the 2D B-frame can be inter-predicted or intra-predicted. The inter-predicted block is either an uni-direction predicted block which has a single pair of reference frame and motion vector, or a bi-direction predicted block which has two pairs of reference frame and motion vector, forward and backward. Block $B(x, y)$ with motion vector mv is matched to $B(x+mv_x, y+mv_y)$ in its reference frame. The intra-predicted block has no motion information between frames.

We interpolate the depth block corresponding to an inter-predicted block in 2D map by MCFI, while we fill the depth block corresponding to an intra-predicted block in the post processing described in Section 2.3.

2.2 MCFI for depth blocks corresponding to inter-predicted 2D blocks

For a depth block corresponding to an uni-direction predicted block in a 2D B-frame, we apply the improved interpolation algorithm described in Section 2.2.1, to fill it using a single pair of reference frame and motion vector.

For the depth block corresponding to a bi-direction predicted block in the 2D B-frame, we fill it by weighted average of the interpolation results of two pairs of reference frame and motion vetor. Let $B(x, y)$ denote the interpolated basic-block; $d(i, j)$ denotes pixel (i, j) in $B(x, y)$, $i, j \in [0, 3]$; and $d_f(i, j)$ and $d_b(i, j)$ denote pixel (i, j) in $B(x, y)$ filled using forward motion information and backward motion information respectively. Then $d(i, j)$ is filled as:

1) If $d_f(i, j)$ and $d_b(i, j)$ are not filled, $d(i, j)$ is not filled;
2) If $d_f(i, j)$ is filled and $d_b(i, j)$ not, $d(i, j)$ will be $d_f(i, j)$;
3) If $d_b(i, j)$ is filled and $d_f(i, j)$ not, $d(i, j)$ will be $d_b(i, j)$;
4) If both $d_f(i, j)$ and $d_b(i, j)$ are filled, the value of $d(i, j)$ will be

$$d(i, j) = \frac{T_b}{T_f + T_b} d_f(i, j) + \frac{T_f}{T_f + T_b} d_b(i, j) \quad (1)$$

where T_f and T_b denotes temporal distances from the interpolated depth frame to forward reference depth frame and backward reference depth frame, respectively.

2.2.1 Improved interpolation algorithm

As shown in Figure 3, D and F denote the interpolated depth frame and its corresponding 2D frame. Each basic block B in D has a co-located basic block B' in F. B' is matched to a block B'_R in its reference 2D frame F_R using its motion vector mv. Then we can get block B_R, co-located with B'_R, in the depth frame D_R corresponding to F_R.

Traditional interpolation algorithm just copies B_R to B. However, because shape of boundaries may change and since block motion vectors, generated in the encoder for compression purpose, may not absolutely correlate with real object motion, copying the block directly will cause distortions especially around the boundaries of objects. The distortions affect the quality of rendered 3D video seriously.

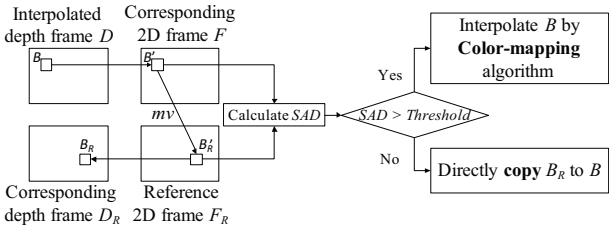

Figure 3: Improved interpolation algorithm

To solve such problem, this paper proposes an improved interpolation algorithm shown in Figure 3.

SAD in Figure 3 is calculated as:

$$SAD = \sum_{i=0}^{3} \sum_{j=0}^{3} \left|\left| c_{B'}(i, j) - c_{B'_R}(i, j) \right|\right| \quad (2)$$

where $c_{B'}(i, j)$ denotes the color value of pixel (i, j) in B' and $c_{B'_R}(i, j)$ denotes the color value of pixel (i, j) in B'_R.

Color-mapping algorithm is a novel interpolation algorithm proposed by us, which is more correct but consumes more time than block copying. Whether block copying or color-mapping algorithm is finally used depends on the similarity between B' and B'_R. When the blocks match well, we simply copy B_R to B. Otherwise, which means the motion information is not reliable or the shape of objects changes, color-mapping algorithm is selected. The threshold is chosen by considering the requirement of interpolation. Large threshold is chosen if high efficiency is required, while small value is selected if high quality is desired. In later experiments, we choose 15 because it promises satisfactory quality under ideal speed.

Basic idea of color-mapping is to map each color with a depth value locally, and interpolate each pixel according to its color in the corresponding 2D map. The process is described in Algorithm 1 and shown in Figure 4.

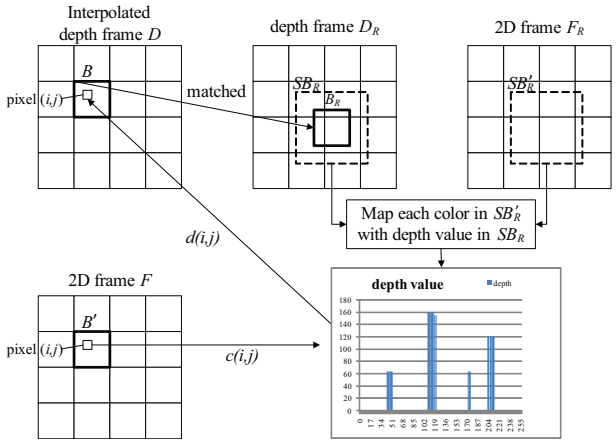

Figure 4: Process of color-mapping

The algorithm expands basic-block to super-block because 4×4 is too small to provide reliable color-depth mapping.

It is easy to understand that false motion vector will lead to larger color error and the pixel will not be filled because $weightSum = 0$. And the algorithm can also make clearer edges because pixels are filled based on color values in F.

2.3 Post processing

After MCFI, there are still pixels not filled because their color have no depth value mapped or belong to intra-predicted blocks. We fill such pixel as weighted average value of its neighboring pixels filled at the previous step. The weight depends on the distance and the color difference in 2D map between the interpolated pixel and the already filled pixel.

3. EXPERIMENTS AND RESULTS

To demonstrate the effectiveness of the proposed framework, we up-converse different sequences under different low frame-rates by using several FRUC approaches.

The testing sequences are provided by MSRA (1024×768) and Philips 3DTV website (960×540). The original frame rate of each sequence is 30 fps, and is temporally downsample to 15, 10 and 7.5 fps. The characters of the sequences are shown in Table 1.

Because depth video is not directly used for display, we measure the quality of depth video by calculating the PSNR between the rendered free-viewpoint video at the position of nearby viewpoint and the standard free-viewpoint video rendered by the original high frame-rate depth video.

Algorithm 1 Color-mapping

Input: Interpolated basic-block B in depth frame D; matched basic-block B_R in depth frame D_R; 2D frame F and F_R corresponding to D and D_R respectively; a Gaussian filter kernel g.
Initialization: Set $m[c]$ records whether color c has a mapped depth value and set $dv[c]$ saves the depth value mapped to it

1. Expand B_R to super-block SB_R of size 8×8
2. Find super-block SB'_R co-located to SB_R in F_R
3. **for** color value $c \in [0, 255]$
4. **if** at least one pixel in SB'_R have value c
5. $m[c] \leftarrow 1$
6. $dv[c] \leftarrow$ average depth value of all the pixels in SB_R correspond to the pixels having value c in SB'_R
7. **else**
8. $m[c] \leftarrow 0$ and $dv[c] \leftarrow 0$
9. **endfor**
10. Find B', which is the co-located basic-block of B in F
11. **for** each pixel (i, j) in B
12. Find color value $c(i, j)$ for pixel (i, j) in B'
13. Define Ω as the searching color range centered over $c(i, j)$
14. $depthSum = \sum_{c \in \Omega} dv(c)m(c)g(||c - c(i, j)||)$
15. $weightSum = \sum_{c \in \Omega} m(c)g(||c - c(i, j)||)$
16. **if** $weightSum \neq 0$
17. $d(i, j) = \frac{depthSum}{weightSum}$
18. **else**
19. Do not fill $d(i, j)$
20. **endfor**

Table 1: Test Sequences

Sequence	*Video description*
Girl	Low motion for both 2D and depth video
Ballet	High motion for a few objects in 2D and depth video
Football	High motion for 2D video, low motion for depth video
Frisbee	High motion for both 2D and depth video

3.1 Comparison with other approaches

This paper compares performance of the proposed framework with other two FRUC approaches: 1) MCFI using 2D motion information derived by full-search block matching algorithm (FBMA) as in [2], abbreviated as MCFI-F; 2) conventional MCFI using motion information of depth video in [6], which is abbreviated as MCFI-D.

PSNR of the rendered video and average time cost to dispose per frame are used to evaluate the three FRUC approaches. The experiment is done on depth video of 15 fps by CPU of 2.56GHz without any speedup by hardware or parallelism. The results are listed in Table 2.

It is obvious that our framework has the highest efficiency among the three approaches. Compared with MCFI-F, it saves up to 99.5% running time per frame. For sequences with high motion in 2D video, the advantage is notable. Our framework achieves satisfactory rendering quality, which is comparable with MCFI-F. Especially, for sequence *Football* and *Frisbee*, our framework outperforms MCFI-F obviously. This is because high motion in the 2D video of these sequences makes the motion estimation in MCFI-F unreliable and time-consuming.

Although speed of MCFI-D on some sequences is accept-

Table 2: Performance Comparison

Sequence	*PSNR Time*	*Approaches*		
		ours	MCFI-F	MCFI-D
Girl	*PSNR*	26.82	26.84	26.44
	ms/f	68	1989	104
Ballet	*PSNR*	28.01	28.16	26.74
	ms/f	53	4151	241
Football	*PSNR*	26.94	26.69	26.40
	ms/f	117	13313	76
Frisbee	*PSNR*	22.86	22.88	20.96
	ms/f	88	18950	302

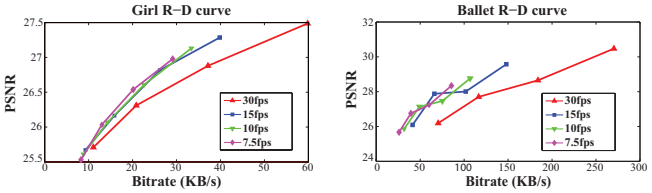

Figure 5: R-D curve under different up-conversion rate for *Girl* and *Ballet*

able, distortion of its rendered free-viewpoint video is serious. This is mainly because the motion estimated from depth video is unreliable. For the depth video with a large motion, MCFI-D is also time consuming.

3.2 Compression by temporal down-sampling

We further compress depth video at the frame rate of 15, 10 and 7.5fps respectively and up-converse them by our framework. We also compress it at the original frame rate. For each condition, we set QP parameters from 24 to 36 in step 4. Figure 5 shows the Rate-Rendering Distortion (R-D) result of *Girl* and *Ballet*. Curves of low frame-rate are above the curves of original frame-rate, which demonstrates that we can compress depth video by temporal down-sampling it and up-conversing it using the proposed framework. The lines show that it can save up to approximately 37% bitrates of depth video while achieving the same rendering quality. We can also find that our framework is robust to large up-conversion scale. Moreover, compression under 7.5fps performs the best under small bitrates.

3.3 Subjective improvement

The visual quality of rendered images also improves when the proposed framework is exploited. Figure 6 shows the comparison between the rendered free-viewpoint samples of different compression conditions for depth video under approximate the same bitrates 68.33KB/s. It can be found that the proposed framework provides more accurate motion estimation than MCFI-D (see the arm of the lady). Moreover, compared with compression at the original 30fps, compressing low frame-rate depth video and using the proposed framework can provide clearer edges of objects (see the head of the man).

Figure 7 shows the effect of the novel color-mapping algorithm. From magnified local parts, we can see that blocks with false motion information are also interpolated correctly and boundaries are much smoother than block copying.

4. CONCLUSION

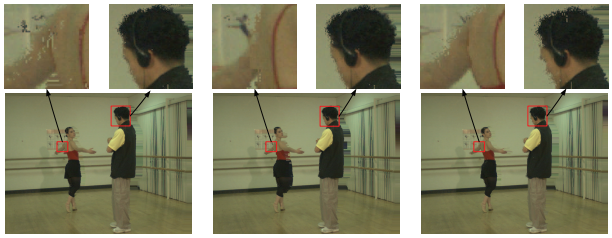

Figure 6: Visual quality of rendered frame of *Ballet*

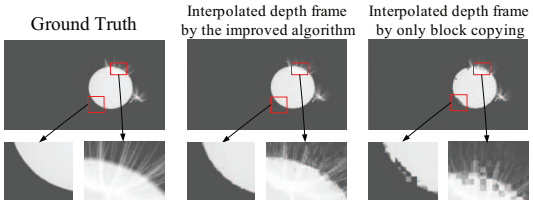

Figure 7: Effect of color-mapping algorithm

This paper speeds up the MCFI based FRUC of depth video by viewing it as part of the video coding process. This assumption is natural as video coding is commonly used in most video applications. The framework selects the coding structure for 2D video, which maps the interpolated depth frames to B frames in 2D video. And at the decoder side, motion information of corresponding 2D video are directly derived as motion estimation of depth video for MCFI, so that time cost for the motion estimation is significantly reduced. The framework also proposes a novel color-mapping interpolation algorithm, which enhances quality of interpolated depth frames.

5. ACKNOWLEDGEMENT

This work is supported by the Development Plan of China (973) under Grant No.2011CB302206, the National Natural Science Foundation of China under Grant No.60833009, and the National High-Tech Research and Development Plan of China (863) under Grant No.2009AA01Z328.

6. REFERENCES

[1] R. Cheng and K. Nahrstedt. Empirical study of 3D video source coding for autostereoscopic displays. In *Proceedings of the 15th international conference on Multimedia*, pages 573–576. ACM, 2007.

[2] J. Choi, D. Min, B. Ham, and K. Sohn. Spatial and temporal up-conversion technique for depth video. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3525–3528. IEEE, 2009.

[3] E. Ekmekcioglu, S. Worrall, and A. Kondoz. A temporal subsampling approach for multiview depth map compression. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(8):1209–1213, 2009.

[4] C. Fehn et al. Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV. In *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, volume 5291, pages 93–104. San Jose, CA, USA, 2004.

[5] H. Wang, C. Huang, and J. Yang. Depth maps interpolation from existing pairs of keyframes and depth maps for 3D video generation. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 3248–3251. IEEE, 2010.

[6] Y. Yang, Y. Tung, and J. Wu. Quality enhancement of frame rate up-converted video by adaptive frame skip and reliable motion extraction. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(12):1700–1713, 2007.