Video Coding by Texture Analysis and Synthesis Using Graph Cut

Yongbing Zhang[1], Xiangyang Ji[2], Debin Zhao[1], Wen Gao[1,2]

[1]Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, P.R. China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P. R.China
{ybzhang, xyji, dbzhao, wgao}@jdl.ac.cn

Abstract. A new approach to analyze and synthesize texture regions in video coding is presented, where texture blocks in video sequence are synthesized using graph cut technique. It first identifies the texture regions by video segmentation technique, and then calculates their motion vectors by motion vector (MV) scaling technique like temporal direct mode. After the correction of these MVs, texture regions are predicted from forward and/or backward reference frames by the corrected MVs. Furthermore, Overlapped Block Motion Compensation (OBMC) is applied to these texture regions to reduce block artifacts. Finally, the texture blocks are stitched together along optimal seams to reconstruct the current texture block using graph cuts. Experimental results show that the proposed method can achieve compared visual quality for texture regions with H.264/AVC, while spending fewer bits.

Keywords: texture, analysis and synthesis, video coding, graph cut, optimal seam

1 Introduction

Texture regions, such as grass, flower, sand, and cloud, appear in many video sequences. However, viewers are not sensitive to these texture regions which usually spend a lot of coding bits. In [1], it is assumed that for these highly textured regions, viewers perceive the semantic meaning of the displayed textures rather than the specific details. Thus it is not necessary to code these texture regions at the expense of high bit rate. In [2], a scheme of texture analyzer and synthesizer is presented. The aim of texture analyzer is to segment video frames and identify the texture regions in them. The texture synthesizer warps the identified texture regions by the warping parameters sent in the bit stream. The non-textured regions are encoded using traditional methods. Their method can save the bit rate up to 19.4% [2] without significant loss of visual performance. It achieved good results for rigid objects. However, it had to consider the neighborhood of the texture regions for non-rigid textures.

Analysis-synthesis-based codec has already been introduced for object-based video coding applications, e.g. see [3]. However, the purpose of the analyzer and synthesizer modules in this case is usually the identification and appropriate synthesis of moving objects, rather than texture regions. A similar wavelet-based analysis-synthesis for still image and video coding approach was introduced by Yoon and Adelson [4]. Whereas, the algorithm presented is optimized for still images.

Graph cut was introduced for image and video synthesis in [5]. In their approach textures are generated by copying input texture patches. It first searches for an appropriate location to place the patch, and then uses a graph cut technique to find the optimal region of the patch to paste in the output. This algorithm can generate textures perceptually similar to the example ones.

In this paper, the input sequences are classified into key frames (frames used as reference) and non-key frames (frames that are not used as reference). The non-key frames are first segmented and then the texture regions are synthesized using graph cut. As there is no bit for the MVs of the texture regions, the MVs of the texture regions are derived by direct mode. Due to the unreliable characteristic of the direct mode MVs, MV correction is required after the direct mode MVs are got. In most cases, translation mode is not suitable for texture regions, which present random distribution, thus OBMC [6] is applied when the MV correction is finished. After OBMC, block artifacts are greatly reduced. Texture regions processed by the aforementioned stages seem close to the natural ones. Next, each texture macroblock is divided into 4 overlapped patches. Each patch has three candidates, which are forward, backward, and bi-directional predictions. The best candidate patch is chosen according to the mismatch of pixels in the overlapped region between the new and old patches. Then graph cut is processed by finding an optimal seam in the overlapped regions, and the existing pixels in the old patches are maintained or updated on the different side of the seam. Textures processed by graph cut seem to be smoother, more natural and the spatial accuracy remain unchanged. So, it will be hard for viewers to find detail differences between synthesized and original texture regions.

The remainder of this paper is organized as follows. In Section 2 we introduce the MV derivation and MV correction in texture regions. In Section 3, we compensate the texture regions utilizing OBMC. And in Section 4 graph cut synthesis is presented. Experimental results are given in Section 5. Finally, conclusions are drawn in Section 6.

2 MV Derivation and MV correction in texture regions

In order to achieve better visual performance, some preprocessing techniques are required. The first is to identify texture regions in a frame, which can be finished by image segmentation. In this paper, we utilize a similar segmentation method as [2]. As there is no bit for MVs of texture blocks in bitstream, the MVs of texture blocks are derived by direct mode [7]. As is shown in Fig. 1, for each texture block, the forward motion vector MV^{fw} and backward motion vector MV^{bw} are calculated as

$$MV_{b1}^{fw} = \frac{TD_B}{TD_D} \times MV_{B4} \quad (1)$$

$$MV_{bw}^{b1} = \frac{(TD_B - TD_D)}{TD_D} \times MV_{B4} \quad (2)$$

If the co-located block in the backward reference frame was coded in intra mode, we got the spatial direct MV to replace the corresponding temporal one. As the MVs derived by direct mode may be sometimes unreliable, the MVs of adjacent blocks are used to correct or smooth the unsatisfied MVs because of the spatial correlation as described in [8]. The aim of the MV correction is to smooth the isolated MVs. If an MV has a weak spatial correlativity with adjacent MVs, it is considered as an isolated MV. The isolated MVs make motions unreliable and cause block artifacts. So it is necessary to correct the isolated MVs by smoothing or median filtering using MVs of adjacent blocks.

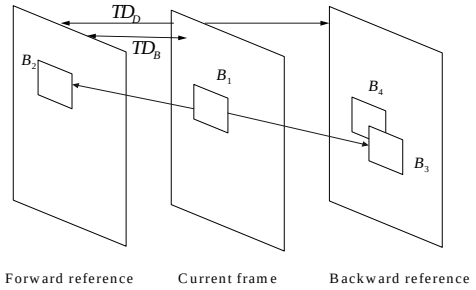


Fig.1 MV Derivation of current block

Let Block(*x*,*n*) be the block in frame *n* and *x*=*x*(*i*,*j*) denotes the spatial position at (*i*,*j*) in one frame and *n* is the frame number. Similarly, set MV(*x*,*n*) to be the MV at position *x* in frame *n*. To detect the isolated MV of Block(*x*,*n*), the distance (MVD) between MV(*x*,*n*) and its surroundings are calculated and the maximum MVD is selected as

$$MaxMVD(x,n) = \underset{x' \in s(x)}{Max} \left\| MV(x,n) - MV(x',n) \right\| \quad (3)$$

where s(*x*) denotes the set of adjacent blocks. x' is in the same frame as *x*. A threshold T by averaging the adjacent block MVs is computed to determine whether the MV(*x*,*n*) is isolated. If *MaxMVD(x,n)* is larger than T, MV(*x*,*n*) is isolated and should be corrected by MVs of adjacent blocks. In this paper, median filter is utilized for its good performance in removing noises. After MV correction, the MV of texture block becomes more reliable, and the block artifact is reduced.

3 Reduction of block artifacts by OBMC

Block artifacts reduced by just MV correction are not enough. Assign each block an MV is under the assumption that the corresponding block is undergoing translational movement. However, for texture regions, the pixel is unstructured and the movement is irregular. Thus, translational mode is not very suitable for texture regions. However, we find that OBMC is very suit for texture regions. As is shown in Fig.2, each pixel in the current block is predicted by a weighted average of several corresponding pixels in the reference frame. The corresponding pixels are determined by the MVs of the current block as well as adjacent blocks. The final pixel value is computed as

$$\Psi p(x) = \sum h_k(x)\Psi r(x + d_{m,k}), x \in B_m \quad (4)$$

where $h_k(x)$ is the weight of corresponding pixel in neighboring block B_m and should be inversely proportional to the distance between x and the center of $B_{m,k}$. For each 4x4 block, OBMC is used after MV correction is finished. The block artifacts are greatly reduced when OBMC is applied and the texture regions seem smoother and more natural.

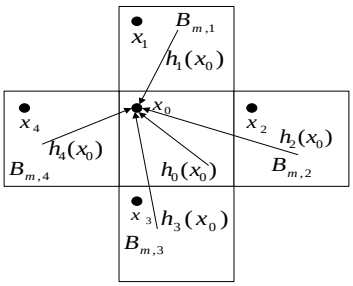


Fig. 2. OBMC with 4 neighborhood MVs

4 Texture synthesis using graph cut

In [5], texture is synthesized by copying irregularly shaped patches from the sample image into output image. The patch copying process is performed in two stages. Firstly, a candidate rectangular patch is selected by performing a comparison between the candidate patch and the patches already in the output image. Secondly, an optimal (irregularly shaped) portion of this rectangle is computed and only these pixels are copied into the output image. The portion to copy is determined by using a graph cut algorithm.

4.1 Selection of the best candidate patch

Combining with H.264, we divide each texture macroblock into 4 non-overlapped 8x8 blocks. As is shown in Fig.3, the patch size is set to be 12x12. Each patch contains two parts: an 8x8 block and its overlapped regions. So, each macroblock can be synthesized by stitching 4 patches together. As each texture block has forward and backward MVs, it has 3 candidate patches, i.e. forward, backward and bi-direction patches. In order to determine which candidate patch is chosen, we first compute the sum of square error (SSE) in the overlapped region， where *OV* means overlapped

$$Dif = \sum_{x \in OV} \left(I_N(x) - I_O(x) \right)^2 \tag{5}$$

region *A* and region *B*, $I_N(x)$ and $I_O(x)$ means the pixel value of position *x* in the new and old patches respectively. Using formula 5, we compute the forward, backward and bi-directional *Dif*, and find the minimum. Then, we set the direction of the chosen patch to be the corresponding direction of the minimum *Dif*. Once the direction is determined, we get the best candidate patch.

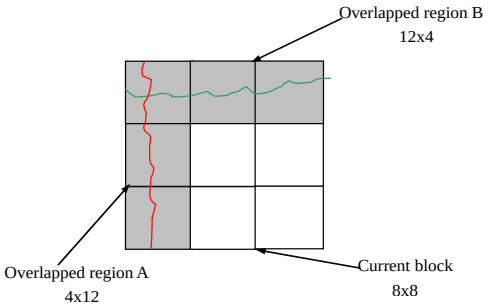


Fig. 3. a patch and its overlapped regions

4.2 Determination of the optimal seam

The heart of the patch based synthesis is to determine an optimal seam in the overlapped regions. When the optimal seam is determined, the pixels on the opposite side of the seam are processed differently. For example, in the overlapped region *A*, the pixels on the left of the optimal seam (red line) maintains the same, while the pixels on the right of the red line are updated by the pixels in the current patch. In the same way, the pixels above the green line are unchanged, whereas the pixels below the green line are replaced by the pixels in the current patch. It can be easily seen that the up left corner of the patch is processed twice, which is because it is contained in both region *A* and region *B*. Pixel values in the right and bottom part of the current 8x8 block may be changed in the following steps, and it is just the basic idea of the patch synthesis.

In order to find an optimal seam in the overlapped region, we have to choose a matching quality measure for pixels from the old and new patch. In this paper, we choose the simplest quality measure, the luminance difference between pixels in the overlapped regions. Let *s* be the pixel position in the overlapped region, and let *N*(*s*) and *O*(*s*) be the luminance at position *s* in the new and old patches, respectively. The matching quality cost *M* between pixels, which come from patches *N* and *O*, is defined to be

$$M\left(s,N,O\right)=\left(N\left(s\right)-O\left(s\right)\right)^2 \quad (6)$$

We then use this matching quality to solve the path finding problem. Considering the overlapped region *A* shown in Fig.3, from the top to the bottom, we find a minimum cost path, and record the minimum cost pixels in the path. For the pixels in the path and the pixels in the left of the path, pixel values remain the same, whereas pixels on the right of the path are replaced by the corresponding ones in the new patch.

5 Experimental results

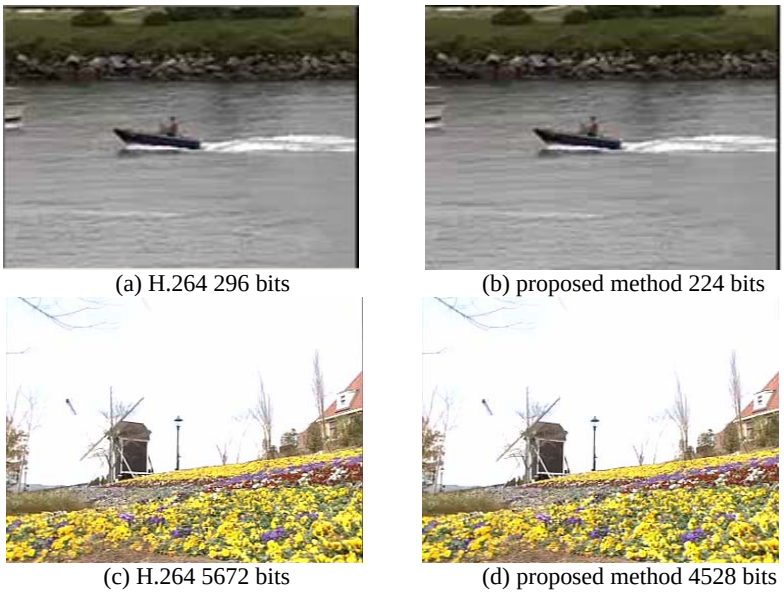
In this paper, we synthesized two well known sequences: *flowergarden* and *coastguard* with 30fps. *Flowergarden* contains rigid textures and *coastguard* contains non-rigid textures. All the experiments are implemented based on the H.264/AVC reference software JM98. Fig.4 shows results of the traditional methods in H.264 and proposed method used for texture regions in sequence *coastguard* and *flowergarden*. From Fig.4, we can see that the proposed method achieves similar visual performance as H.264, while saving the bits above 20%.

Table. 1 shows that our proposed method outperforms the skip mode in H.264 for texture regions at the same bit rate. The maximum PSNR gain is up to 0.34dB for sequence *coastguard* in CIF format compared with the skip mode. And the maximum PSNR gain is up to 0.25dB for sequence *flowergarden* in QCIF format. As texture regions only take up a small fraction of the whole frame, and the proposed method is only applied to texture regions, the PSNR gain is relatively large compared with the whole frame. The detailed information about the bit saving of the proposed method compared with H.264 is shown in Table 2. The maximum saving of bits can be up to 36.29% compared with H.264 for sequence *coastguard* in CIF format. The maximum saving of bits can be up to 21.83% compared with H.264 for sequence *flowergarden* in CIF format.

Fig.5 shows the rate-distortion curves of the proposed method for sequences *coastguard* and *flowergarden* compared with the skip mode. It can be easily seen that the proposed method can significantly improve the coding efficiency compared with the skip mode when spending the same bits.

6 Conclusions

In this paper, we propose a video texture analysis and synthesis approach using graph cut. It is composed of three steps. Firstly, it segments a frame into texture macroblocks and non-texture macroblocks. Secondly, it carries on some preprocessing for texture macroblocks, which include MV derivation, MV correction and OBMC. Thirdly, optimal texture candidates are determined and stitched together. Experimental results show that the proposed method achieves better object and visual performances than that when encoded by skip mode for texture regions. Furthermore, the proposed method achieves similar visual performance compared with H.264, while saving bit rate up to 36% at most. Besides, the proposed method is suitable for both rigid and non-rigid textures, which approves its robustness



(a) H.264 296 bits (b) proposed method 224 bits

(c) H.264 5672 bits (d) proposed method 4528 bits

Fig. 4. Visual comparisons between different methods. (a),(b),(c) and (d) illustrate the results for H.264 encoded and the proposed for sequences *coastguard*(QCIF) and *flowergarden*(CIF)

Table 1. PSNR comparison between the proposed method and the skip mode

Video Sequence	Format	Average PSNR gain			
		QP=30	QP=32	QP=34	QP=36
flower	CIF	0.2363db	0.2757db	0.1847db	0.1589db
	QCIF	0.2514db	0.1167db	0.0926db	0.0736db
coastguard	CIF	0.2186db	0.241db	0.343db	0.2353db
	QCIF	0.1563db	0.1557db	0.1649db	0.085db

Table 2. Bit rate comparison between the proposed method and H.264

Video Sequence	Format	Average bits saving			
		QP=30	QP=32	QP=34	QP=36
flower	CIF	21.83%	18.06%	20.02%	11.54%
	QCIF	10.45%	9.3%	3.53%	6.48%
coastguard	CIF	36.29%	34.97%	29.45%	19.94%
	QCIF	15.55%	21.91%	28.6%	2.1%

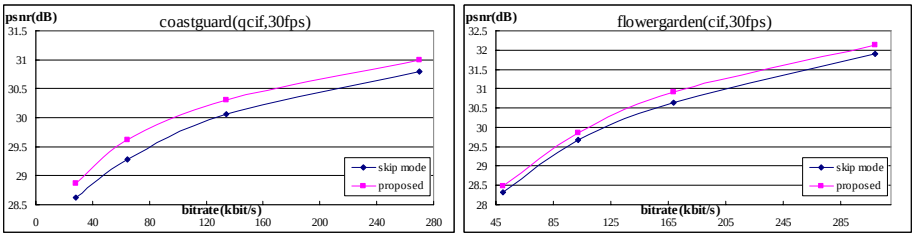


Fig. 5 Rate-distortion curves for skip mode and proposed method for texture regions in sequences *coastguard* and *flowergarden***.**

References

1. P.Ndjiki-Nya, T. Hinz, A. Smolic, and T. Wiegand.: A Generic Automatic Content-based Approach for Improved H.264/AVC Video Coding. ICIP 2005, Genoa, Italy, September 2005
2. P.Ndjiki-Nya, T. Wiegand: Video Coding using texture analysis and synthesis. PCS 2003, Saint-Malo, France, April 2003
3. M.Wollborn Prototype Prediction for Clour Update in Object-Based Analysis-Synthesis Coding. IEEE Trans. Circuits Syst. Video Technol., vol.4, no.3,pp.236-245, June. 1994
4. S.-Y. Yoon and E.H. Adelson. Subband texture-synthesis for image coding. Proceedings of SPIE, Human vision and Electronic Imaging III (1998) 489-497
5. V.Kwatra, A.Schödl, I.Essa, G. Turk, A. Bobick: Graphcut Textures: Image and Video Synthesis using Graph Cuts. ACM Transactions on Graphics, SIGGRAPH 2003
6. T. Kuo, and C.-C. Jay Kuo. Fast Overlapped Block Motion Compensation with Checkerboard Block Partitioning. IEEE Trans. Circuits Syst. Video Technol, vol.8, no.6, pp 705-712, October, 1998
7. A.M.Tourapis, Feng.Wu, Shipeng Li. direct mode coding for bipredictive slices in the H.264 Standard. IEEE Trans. Circuits Syst. Video Technol, vol.15, no.1, pp 119-126, Jan. 2005
8. H. Sasai, S. Kondo and S.Kadono. frame-rate up-conversion using reliable analysis of transmitted motion information. Proc. of IEEE Conference on Acoustics, Speech, and Signal processing, vol.5, pp 257-260, May, 2004
9. J.Zhang, L.Sun, S.Yang.: Position Prediction Motion-compensated interpolation for frame rate up conversation using temporal modeling. ICIP 2005
10. Jiefu Zhai, Keman Yu, Jiang Li, Shipeng Li.: A Low Complexity Motion Compensated Frame Interpolation Method. ISCAS 2005, May 2005
11. A.Schödl, R. Szeliski, David H. Salesin, and Irfan Essa. Video Textures. Proceedings of the 27th annual conference on computer graphics and interactive techniques.(2000) 489-498