Learning-based Multiview Video Coding

Baochun Bai*, Li Cheng†, Cheng Lei*, Pierre Boulanger*, and Janelle Harms*
* Department of Computing Science, University of Alberta, Edmonton, Canada
Email: {baochun, clei, pierreb, harms}@cs.ualberta.ca
† Toyota Technological Institute at Chicago, 6045 S. Kenwood Ave. Chicago, Illinois, USA 60637
Email: {licheng}@tti-c.org

***Abstract*—In the past decade, machine learning techniques have made great progress. Inspired by the recent advancement on semi-supervised learning techniques, we propose a novel learning-based multiview video compression framework. Our scheme can efficiently compress the multiview video represented by multiview-video-plus-depth (MVD) format. We model the multiview video compression problem as a semi-supervised learning problem and design sophisticated mechanisms to achieve high compression efficiency. Our approach is significantly different from the traditional hybrid coding scheme such as H.264-based multiview video coding methods. The preliminary results show promising compression performance.**

***Index Terms*—joint multiview video coding, semi-supervised learning, multiview video plus depth format, 3D TV, H.264**

I. Introduction

Multiview video compression is crucial to the success of multiview video applications such as 3D TV and free viewpoint video (FVV). Joint multiview video coding (JMVC) that exploits correlation among multiview videos at the encoder is a popular multiview video compression framework. Research on JMVC initially focuses on how to efficiently compress multiview video texture data based on conventional block-based prediction and transform coding approach by taking advantage of both temporal and inter-view similarity among multiple video streams [1]. For example, the Joint Video Team (JVT) of MPEG and ITU-T is now developing a Joint Multiview Video Model (JMVM) which is based on the H.264 hybrid video coding standard [2]. Recently some researchers started developing methods to efficiently compress both multiview video texture and depth data [3], [4]. Multiview-video-plus-depth (MVD) format is a popular data representation format for 3D video applications. The MVD format makes it easy to synthesize virtual views in client machines.

In the past decade, machine learning techniques such as clustering and classification have made significant progress. Video coding such as vector quantization based methods have inherent connection with clustering techniques [5]. Cheng and Vishwanathan [6] recently propose a new method based on semi-supervised learning [7] to compress image and video chrominance data. Inspired by their work, we propose a learning-based multiview video coding (LMVC) scheme to compress multiview videos represented by the MVD format. The basic idea of our scheme is to model the multiview video compression problem as a semi-supervised learning problem. Depth data are compressed by H.264-based schemes. Texture data are compressed by semi-supervised learning algorithms.

II. Semi-Supervised Learning

Let $\mathcal{X}$ be the space of observations and $\mathcal{Y}$ the space of labels. $\mathcal{Y}$ is assumed to be a finite subset of $\mathbf{R}$. The semi-supervised learning [7] is formulated as follows: Given a sequence $\{(\mathbf{x_i}, y_i)\}_{i=1}^{m}$ of labeled observations drawn from $\mathcal{X} \times \mathcal{Y}$, $\{\mathbf{x_i}\}_{i=m+1}^{n}$ of unlabeled observations drawn from $\mathcal{X}$, and a loss function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \rightarrow \mathbf{R}$, learn a function $f \in H$ which minimizes the loss on the labeled observations and also generalizes well to unseen observations.

Graph-based semi-supervised learning methods construct an adjacency graph, $\mathcal{G}$, whose nodes are the observations, and edges encode nearest neighbor relationships. The semi-supervised learning problem is modeled as estimating a smooth function that respects neighborhood relations on the graph. [6] learns the function f by minimizing

$$J(f) = \frac{\lambda}{n^2}\|f\|_{\mathcal{G}}^2 + \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, f). \qquad (1)$$

where $\|f\|_{\mathcal{G}}^2 = \mathbf{f}^\top \bigtriangledown_{\mathcal{G}} \mathbf{f}$ denotes the regularizer and l is a loss function between the given label y_i and predicted label $f(x_i)$ for labeled examples.

Semi-supervised learning approaches can be classified into two categories: transductive learning and inductive learning. Transductive learning works on both labeled and unlabeled training data, but cannot handle unseen data. Inductive learning can handle unseen data and aim at generating a prediction function which is defined on the entire space.

III. Learning-based Multiview Video Coding

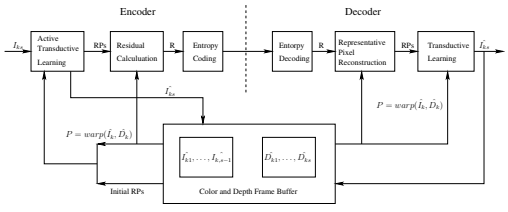


Fig. 1. The Architecture of LMVC

The proposed learning-based multiview video coding scheme aims at efficiently compressing multiview videos represented with MVD format. The basic idea is to first compress the depth map by the H.264-based schemes and then use an active semi-supervised learning method to compress texture

data. LMVC compresses the texture image by choosing a small number of representative pixels (RP) at the encoder and uses the semi-supervised learning to reconstruct the original image based on the RPs at the decoder. Fig. 1 gives an illustration of the architecture of the proposed learning-based multiview video coding scheme. A texture image includes both luminance and chrominance frames. The generic architecture of LMVC shown in Fig. 1 can be used to compress both luminance and chrominance frames. However, there is a slight difference to compress luminance frames and chrominance frames. To compress a luminance frame, LMVC needs to first obtain an approximate frame through temporal/inter-view prediction. The approximate frame aims at constructing the adjacency graph used in the graph-based semi-supervised learning algorithm. To compress the chrominance frames, LMVC uses the luminance frames to construct the adjacency graph as in [6]. Depth map is used in LMVC to help generate approximate frames and reuse the representative pixels. Though LMVC can be used to compress both key pictures and non-key pictures [8], we focus on studying its performance on key pictures. Namely, only inter-view prediction is used to generate approximate frames and decides RP reuse. Specifically, we concentrate on the inter-view prediction structure similar to the prediction structure for key pictures such as KS-IPP in [8].

Since the eventual bit rate is proportional to the total number of representative pixels transmitted to the decoder, the challenge of LMVC is how to choose the minimum number of representative pixels to reach the target PSNR. Formally, let $\{I_{ij} = \langle Y_{ij}, U_{ij}, V_{ij} \rangle\}$, $i = 1, 2, \cdots, z$ and $j = 1, 2, \cdots, u$ denote a multiview video sequence where Y is luminance frames, U and V are chrominance frames, u is the number of video sequences captured by u different cameras, and z is the number of images in each video sequence. Let $\{D_{ij}\}$, $i = 1, 2, \cdots, z$ and $j = 1, 2, \cdots, u$ denote the disparity map for frame i in video stream j. For a specific image k in the video stream s, to compress the luminance frame, LMVC needs to first get the approximate frame $Y^p_{ks} = warp(\hat{Y_{k1}}, \cdots, \hat{Y_{k,s-1}}, \hat{D_{k1}}, \cdots, \hat{D_{k,s}})$, where $\hat{Y_{kj}}$ and $\hat{D_{kj}}, j = 1, \cdots, s-1$ are reconstructed luminance and depth frames in the video stream j. Y^p_{ks} is used in the semi-supervised learning algorithm to construct the adjacency graph. To compress the chrominance frames, the reconstructed luminance frame $\hat{Y_{ks}}$ is then used to construct the adjacency graph. In this paper, we are interested in using the semi-supervised learning approach as illustrated in Fig 1 to compress $I_{ks}, s = 1, 2, \cdots, u$. Given a target distortion level denoted by the target PSNR, SNR_t, and the realized PSNR, SNR_r, the goal of our algorithms is to try to achieve $|SNR_r - SNR_t| \leq \epsilon$, where ϵ is a preset threshold value, by minimizing the total bit rate. We will elaborate on the algorithm details in the following sections.

A. Transductive Learning

Similar to [6], we use graph-based semi-supervised learning to choose representative pixels (RPs). The learning objective is trying to minimize the generalized risk denoted by Equation 1. However, unlike the inductive learning approach in [6], we use the transductive learning approach since inductive learning is computationally expensive to inverse an $n \times n$ dense matrix to solve the semi-supervised learning problem. In our current implementation, the transductive learning actually minimizes the following objective functions:

$$\sum_{i=1}^{n} \left(f(\mathbf{x_i}) - \sum_{i \sim j} w_{ij} f(\mathbf{x_j}) \right)^2 + \sum_{i=1}^{m} \delta(f((x_i), y_i). \quad (2)$$

where w_{ij} is the weight between the pixel $\mathbf{x_i}$ and its neighbor pixels $\mathbf{x_j}$ and $\delta(f(\mathbf{x_i}), y_i)$ is a loss function which is 0 if $f((x_i) = y_i$ and ∞ otherwise. Minimizing Equation 2 actually is an optimization problem with quadratic cost function and linear constraints. It can be solved efficiently by yielding a large sparse system of linear equations [9].

A graph denoting the adjacency relationship between all labeled and unlabeled pixels is first built based on the per-pixel feature map that includes the pixel value information of a 3×3 local window around each pixel in the predicted frame P_{ks}. Similar to [6], [9], a 4-nearest neighbor adjacency graph is constructed in the feature space.

B. Representative Pixel Selection

The key challenge of the LMVC scheme is how to choose as few representative pixels (RPs) as possible to achieve the target PSNR. Our basic strategy is to group pixels into clusters and choose representative pixels from each pixel cluster and reuse as many representative pixels in the neighbor frames as possible to reduce total bits used to encode the representative pixels. We elaborate on the details of the algorithm as follows. The algorithm first encodes I_{k1} by using H.264 I frame. Denote the reconstructed frame as $\hat{I_{k1}}$. We use a mean-shift based segmentation algorithm [10] to process the frame $\hat{I_{k1}}$ and obtain an over-segmented image. A deterministic algorithm is used to choose q RPs from each segment and total m_{k1} RPs are selected. Represent each RP, $\{RP^{k1}_i\}, i = 1, \cdots, m_{k1}$, as $([x^{k1}_{iv}, x^{k1}_{iu}], \hat{v^{k1}_i})$, where $[x^{k1}_{iv}, x^{k1}_{iu}]$ is the coordinates of RP^{k1}_i and $\hat{v^{k1}_i}$ is the quantized value of RP^{k1}_i in the frame I_{k1}. To encode the frame I_{k2}, each RP in the frame I_{k1}, RP^{k1}_i, is warped into the frame I_{k2} to get a RP, RP^{k2}_j. Namely, $RP^{k2}_j = ([x^{k2}_{jv}, x^{k2}_{ju}], \hat{v^{k2}_j}) = warp(RP^{k1}_i, \hat{D_{k1}})$. Note that $\hat{v^{k2}_j} = \hat{v^{k1}_i}$. If the coordinate of RP^{k2}_j is within the boundary of the frame I_{k2}, RP^{k2}_j is used as a representative pixel. If the signal to noise ratio (SNR), $SNR^{k2}_{p_j}$, for RP^{k2}_j is greater than the target PSNR, SNR_t, namely, $SNR^{k2}_{p_j} \geq SNR_t$, the predicted pixel value, $\hat{v^{k2}_j}$, of RP^{k2}_j is kept as the quantized pixel value. Otherwise, $\hat{v^{k2}_j}$ is assigned a new quantized value which guarantees that its signal to noise ratio is greater than the target PSNR. A reuse flag, f^{k2}_p, is set for each predicted RP to indicate whether the predicted pixel value is used as the pixel label. In this way, we ensure that every predicted RP has correct label and this helps the transductive learning algorithm find right labels for unlabeled pixels. By reusing

the RPs, total m_{k2}^p RPs can be used as initial RPs to train the semi-supervised learner for the frame I_{k2}. If the semi-supervised learning algorithm cannot learn a good model based on the predicted RPs to make the predicted frame reach the target PSNR, the predicted pixel value of every unlabeled pixel based on the current model is evaluated and pixels in high error areas are identified and clustered. The algorithm then chooses q representative pixels from each cluster and assigns those RPs correct labels which ensure that their signal to noise ratios are greater than the target PSNR. The newly chosen RPs are added into the labeled pixel set and the semi-supervised learner starts to train a new model. The process repeats until the realized PSNR, SNR_r, is within a threshold, ϵ, of the target PSNR, SNR_t. Finally, m_{k2}^n new RPs are selected. Therefore, total $m_{k2} = m_{k2}^p + m_{k2}^n$ RPs are needed to compress the frame I_{k2}. To encode the frame I_{k3}, m_{k2} RPs in the frame I_{k2} is first warped to m_{k3}^p RPs in the frame I_{k3}. The above process goes on until u frames are all encoded.

C. Residual Calculation

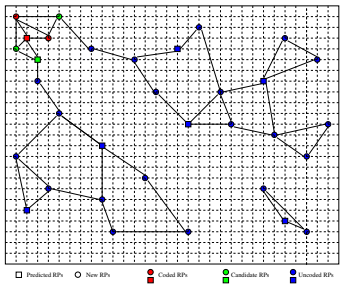


Fig. 2. Neighbor Graph of RPs and Residual Calculation Process

After choosing the right number of representative pixels to reach the target PSNR, residuals need to be computed so that the entropy encoding can achieve a high compression efficiency. Given a frame I_{kj} and its predicted frame P_{kj}, the straightforward way to calculate the residual is to directly deduct the corresponding pixel value, p_i^{kj}, in P_{kj} from the quantized pixel value of a RP, $\hat{v_i^{kj}}$. Namely, $R_i^{kj} = \hat{v_i^{kj}} - p_i^{kj}$. However, the method might not be able to obtain good compression performance since the best prediction for each RP might be its neighbor RPs in the frame I_{kj}. In this section, we propose a neighbor-graph based residual calculation scheme.

Given m_{kj} RPs and their coordinates and quantized pixel value, our algorithm first constructs a t-nearest neighbor graph based on the Manhattan distance between coordinates of different RPs. Namely, each RP in the graph has at least t neighbors and its degree is greater than or equal to t. Fig. 2 shows a 2-nearest neighbor graph. Each RP maintains an ascending order list of its neighbors based on the Manhattan distance and coordinates. The neighbors are first ordered by Manhattan distance. When two neighbors have the same Manhattan distance, the neighbor with smaller coordinates are in front of the one with larger coordinates. Based on the neighbor graph, the algorithm can decide a unique encoding order to calculate the residual of each RP after the first coded RP is determined. Now the RP with the smallest coordinate is first coded. The residual of the first coded RP is always calculated by deducting its corresponding pixel value in the predicted frame P_{kj}.

The algorithm maintains three set of RPs: coded RP set, *SC*, candidate RP set, *SN*, uncoded RP set, *SU*. *SC* includes the RPs whose residual is already calculated. *SN* includes the RPs from which the next coded RP will be selected. *SU* is composed of RPs whose residuals are not yet computed. For each RP, the algorithm also maintains a variable, f_m, to indicate from which RP its residual is calculated. For a t-nearest neighbor graph, f_m has $t+1$ values to indicate $t+1$ possible mode. $f_m = 0$ means that the residual is calculated from the corresponding pixel in the predicted frame P_{kj}. $f_m = 1, \cdots, w, \cdots, t$ means that the residual is computed from the wth-nearest neighbor RP. Once the residual of the first RP is coded, it is added into the set *SC*. Then all its uncoded neighbor RPs are added into the set *SN*. The next coded RP is chosen from the set *SN* and has the property that its shortest Manhattan distance with one of RPs in the set *SC* is smallest among all the RPs in the set *SN*. When two or more RPs have the same shortest Manhattan distance from the RPs in the set *SC*, the RP with the smallest coordinate is chosen to be coded next. The chosen RP calculates its residual with all its neighbor RPs in the coded set *SC* and then selects the residual whose absolute value is the smallest as its final residual and sets f_m to indicate the neighbor RP that leads to the smallest residual. The process continues until the residuals of all the RPs are calculated or the set *SN* becomes empty. When *SN* is empty, it means that the neighbor graph is disconnected as the case shown in Fig. 2 and then the RP with the smallest coordinate in the uncoded set *SU* is chosen to be the next coded RP. Experiments show that the neighbor graph approach has a better compression efficiency than the straightforward one.

D. Entropy Coding

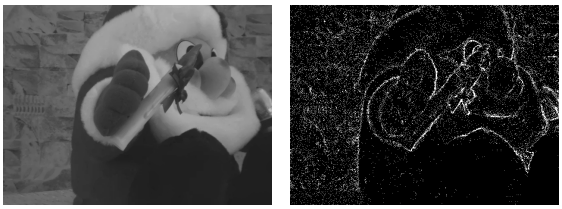
After calculating the residual for all RPs, the coordinates and residuals of RPs are encoded by using arithmetic coding. Coordinates are coded first. Only coordinates of new RPs in the frame $\{I_{kj}\}, j = 2, \cdots, u$ are coded. The coordinates of RPs in the frame I_{k1} that are chosen by the deterministic RP selection algorithm need not be coded since the decoder can use the same algorithm to find them. The reuse flag f_p, the prediction mode f_m, and the residuals of unreused RPs and new RPs are then coded. Note that the coding order of residuals must follow the same order from which residuals are computed. A zero-order word-based arithmetic coding algorithm [11] is now used in our implementation. The decoding process just follows the inverse steps shown in Fig. 1 and uses similar algorithms to reconstruct the original image.

IV. Experiments

We use the 624×464 "Santa Claus" multiview video sequence to study the performance of our schemes. It has 9 views. We evaluate the performance of LMVC on compressing

luminance frames and chrominance frames and compare its performance with JMVM.

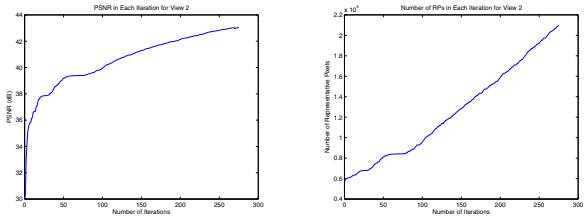
LMVC is first used to compress the luminance frames. The Y component of "Santa Claus" sequences are compressed. Fig. 3 shows decoded Y frame and distribution of RPs. It shows that RP selection algorithm works well and the algorithm could automatically choose RPs along boundaries and around occlusion regions where there are sharp pixel intensity change and great prediction error.



(a) Decoded Y Frame (b) RPs

Fig. 3. Compressing Luminance Frames of "Santa Claus" by LMVC

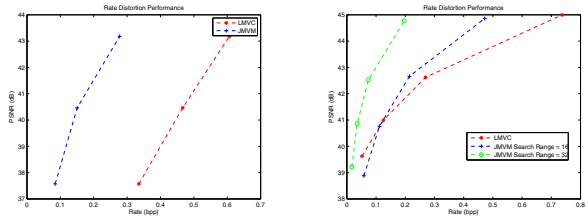
Fig. 4 shows the evolution of PSNR and the number of RPs during the course of training for view 2. It has similar characteristics to the results in [6]. Namely, initially it takes a relatively small number of RPs to quickly improve the realized PSNR, however, it takes a large number of RPs in the later iteration but with only moderate improvement of signal to noise ratio. Similar phenomena happen when LMVC is used to compress the chrominance frames. We omit the figures due to space limitation.



(a) PSNR Evolution (b) Number of RPs Evolution

Fig. 4. PSNR and Number of RPs Evolution Example

The rate-distortion performance of LMVC on compressing luminance frames and chrominance frames as shown in Fig. 5 is compared with JMVM. It shows that there is still a large gap between LMVC and JMVM on compressing luminance frames. The reason for the poor performance of LMVC on luminance frames might be because the predicted frame based on frames in the neighbor views is not good enough for LMVC to construct a good adjacency graph to predict the unknown pixel values based on a small number of RPs. When compressing chrominance frames, LMVC has a large performance gap with JMVM when the target PSNR is high



(a) Luminance Frames (b) Chrominance Frames

Fig. 5. Rate-Distortion Performance of LMVC on Luminance and Chrominance Frames of "Santa Claus" Sequence

and can achieve a small and comparable performance when the target PSNR is low.

V. Conclusions

In this paper, we propose a novel learning-based multiview video compression framework. LMVC intends to efficiently compress multiview video represented by the MVD format. We model the multiview video compression as a semi-supervised learning problem and find ways to solve it efficiently. The initial results show that our scheme still has a significant gap with JMVM when it is used to compress luminance frames. In the case of compressing chrominance frames, there is also a large gap with JMVM when the target PSNR is high. LMVC can achieve a comparable performance with JMVM when the target PSNR is relatively low.

References

[1] A. Smolic, K. Mueller, N. Stefanoski, J. Ostermann, A. Gotchev, G. B. Akar, G. Triatafyllidis, and A. Koz, "Coding algorithms for 3dtv — - a survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1606–1621, November 2007.

[2] A. Vetro, Y. Su, H. Kimata, and A. Smolic, "Joint multiview video model jmvm 2.0," in *ITU-T and ISO/IEC Joint Video Team, Document JVT-U207*, HangZhou, China, 2006.

[3] S.-T. Na, K.-J. Oh, and Y.-S. Ho, "Joint coding of multiview video and corresponding depth map," in *Proc. of IEEE International Conference on Image Processing (ICIP 2008)*, San Diego, CA, USA, 2008.

[4] P. Merkle, A. Smolic, K. Mller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proc. of IEEE International Conference on Image Processing (ICIP 2007)*, San Antonio, TX, USA, 2007.

[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[6] L. Cheng and S.V.N. Vishwanathan, "Learning to compress images and videos," in *Proc. of the 24th Annual Conference on Machine Learning (ICML 2007)*, Corvallis, Oregon, USA, 2007.

[7] X. Zhu, "Semi-supervised learning literature survey," in *Technical Report 1530, Computer Sciences, University of Wiscobnsin-Madison*, 2005.

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

[9] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *Proc. of ACM Annual Computer Graphics Conference (SIGGRAPH 2004)*, Los Angels, CA, 2004, pp. 689–694.

[10] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[11] A. Moffat, R. Neal, and I.H. Witten, "Arithmetic coding revisted," *ACM Transactions on Information Systems*, vol. 16, no. 3, pp. 256–294, 1998.