Video Coding with Cubic Spline Interpolation and Adaptive Motion Model Selection

Haricharan Lakshman, Heiko Schwarz and Thomas Wiegand
Image Processing Department
Fraunhofer Institute for Telecommunications -
Heinrich Hertz Institute
Einsteinufer 37, 10587 Berlin, Germany

***Abstract*—We propose a block based video coder with multiple motion models. The coder is designed such that each block can be predicted using the motion model with the best rate-distortion performance. In such a scheme, there is a significant impact on the interpolation, motion compensated prediction structure and the design of motion vector predictors, because neighboring blocks could have been predicted by different models. We employ cubic splines to perform interpolation and use the spline framework to estimate the translational shift and warping of blocks. We address the problem of adaptive model selection and enhance the motion vector predictors to account for multiple motion models. Bitrate savings up to 11.7% are observed for the tested sequences compared to an improved H.264/AVC reference.**

I. Introduction

Block based video coding with translational motion compensation (MC) is well established in international standards like H.264/AVC [1]. Moving from 2-parameter (translational) model to 4-parameter symmetric zoom-rotation model or the 6-parameter affine model has the potential to improve the video compression efficiency. An affine model was adopted as part of the global MC in MPEG4-Part 2 ASP. Some of the early proposals for the H.264/AVC standard were based on affine motion models [2], [3]. The proposal [3] had rate-distortion optimized quantization of the model parameters to simulate lower order Parametric Motion Model (PMM). The adoption of PMM into mainstream video coding has been rather slow due to two major reasons - difficulty in estimating the parameters and additional bits required to transmit more parameters. With the advances in semiconductor technology the issue of estimation complexity is becoming less important. The extra side information required to signal the additional motion parameters is a more fundamental problem.

The benefits from a higher order PMM can be viewed in two ways. From a motion estimation point of view, PMMs are able to more accurately represent complex motions that occur in a typical video sequence, whereas from an optimization point of view, the additional parameters provide extra degrees of freedom to reduce the rate-distortion cost. Although PMMs show gains [4] when applied on previous standards like H.261 and H.263, the advantage might be reduced in H.264/AVC because of the improved structure. In order to fully utilize H.264/AVC capabilities and harness the advantages of PMMs, we propose to include the PMM as a candidate for motion compensated prediction which competes with the traditional translational model for each block. This involves the signaling of the chosen prediction model and the extra motion parameters whenever applicable.

The motion vector field when using PMM can consist of locations in the reference picture that do not fall on integer pixels. In order to get the prediction signal, it is necessary to evaluate the reference picture at non-grid locations. To this end, discrete-to-continous mapping techniques [5] are utilized. In [3], the intermediate locations are evaluated by fitting a cubic convolution kernel to the reference pixels. The usage of a cubic convolution kernel, which is a fixed 4-tap filter, is justifiable in the context of H.263, but compared to a 6-tap filter in the H.264/AVC standard, a performance loss is observed. Hence we propose to use cubic splines for the interpolation and discrete-to-continous mapping. The spline coefficients are signal adaptive and are known to provide very good interpolation properties [5]. Therefore, even when PMMs are not used for MC, cubic spline is a good candidate for the interpolation of quarter-sample locations resulting from fractional shifts in translational model.

This paper is organized as follows. The overall framework of the video coder, in terms of the interaction of interpolation and MC module, is discussed in Sec. II. The computation of spline coefficients and the interpolation process is specified in Sec. III. In Sec. IV, we elaborate the optimized mode selection strategy and MV predictor design for the case of multiple motion models. In Sec. VII we provide simulation results comparing the proposal with a translational model based video coder. Finally, in Sec. VIII, we conclude the paper.

II. Framework

The design of a variable block size coder with multiple motion models involves many challenges. The estimation of higher order motion parameters for small blocks tend to become unstable and can be categorized as an overfitting problem in comparison to global motion estimation. The continuity of motion vector (MV) predictor which predicts current MVs using neighboring blocks' MVs is disrupted because different motion models could have got selected for the neighbor blocks. The quantization and entropy coding of motion parameters significantly affect the RD peformance. In this paper, we

978-1-4244-7138-6/10/$26.00 ©2010 IEEE

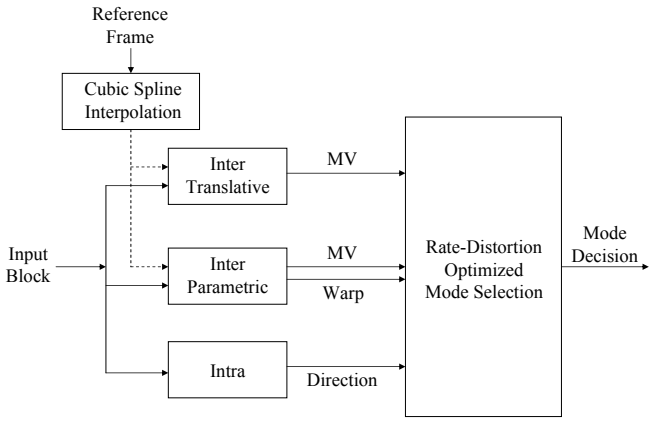


Fig. 1. Block diagram of the proposed system. In the first step, the reference frames are converted into spline coefficients. Next, for every input (macro) block, three possible predictions are computed: using translational motion model, parametric motion model and intra prediction. The mode that has the least RD cost is selected.

address these issues and also discuss the optimized selection of motion model for each block.

We use a quad-tree framework in which the block sizes are allowed to vary from 64×64 to 4×4. There has been considerable gain reported [6] when using block sizes higher than the traditional 16×16. The encoder starts with the biggest block size and evaluates the RD cost of all possbile prediction modes and chooses the mode with the smallest cost. The block is then split into four sub-blocks and the RD costs are evaluated for each sub-block. The encoder then compares the cost of the parent block with the sum of the costs of the sub-blocks to decide whether to retain the parent block or to split it. The sub-blocks can be further split recursively untill the minimum allowed block size. At the end, a quad-tree prediction structure is obtained with the chosen predciction mode for each node. Fig. 1 shows the RD optimized mode decision scheme for each block. The transmission of Inter-translative MV and Intra mode Direction is the same as in H.264/AVC. In case of Inter-parametric mode, extra 'Warp' parameters are signalled, described in detail in Sec. V.

III. Cubic Spline Interpolation

Splines are piecewise polynomials with pieces that are smoothly connected together. Within the family of polynomial splines, cubic splines are the most popular in applications because of their minimum curvature property [5]. B-splines are the basic building blocks for splines. Denoting the B-spline of degree 3 as $\beta^3(x)$ and the spline coefficients as $c(k)$, the cubic spline model for a particular location x in the signal s is given by,

$$s(x) = \sum_{k_1}^{k_1+3} c(k)\beta^3(x-k), \tag{1}$$

where $k_1 = \lceil x-2 \rceil$. Given the signal samples, the interpolation task is to determine the spline coefficients such that there is a perfect fit at integers locations. The computation of the

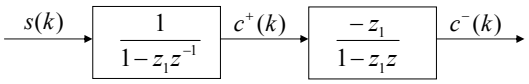


Fig. 2. Computation of spline coefficients. Input signal $s(k)$ passes through a causal filter (left-to-right direction) to produce intermediate result $c^+(k)$. An anti-causal filter operates on $c^+(k)$ (right-to-left direction) to produce the spline coefficients denoted as $c^-(k)$.

spline coefficients can be implemented very efficiently using a cascade of first order causal and anti-causal recursive filters [7] as shown in Fig. 2. The spline coefficient computation is done in a separable way i.e. 1D filtering is applied successively along the rows and columns of the reference image. Denoting $c^-(k) = c(k)/6$, the following recursive algorithm [7] is used to compute the spline coefficients,

$$\begin{aligned} c^+(k) &= s(k) + z_1 c^+(k-1), k = 1, \cdots, N-1 \\ c^-(k) &= z_1(c^-(k+1) - c^+(k)), k = N-2, \cdots, 0. \end{aligned} \tag{2}$$

In the interpolation step, for each location in the destination the corresponding set of points in the source image is determined and image value is computed by a 2D convolution.

Splines have an important property of smoothness. Splines of degree n are $(n-1)$ times continuously differentiable. This is a very useful feature not only for interpolation but also for motion estimation. For the estimation of parameters of PMMs (e.g. affine warping), gradients need to be computed at non-grid locations. This can be easily done using the already computed spline coefficients. Using the same spline model for these different aspects of interpolation and motion estimation makes the design consistent and robust compared the traditional technique of using 3 different filters (a low pass filter of half-pel interpolation, a bilinear filter for quarter-pel interpolation and a high pass filter for gradient computation).

IV. Parametric Motion model estimation

Parametric motion models provide better motion representation than a translational motion model. Examples of parametric models include 6-parameter affine, 4-parameter isotropic-zoom-rotation model etc. We perform the parameter estimation in two steps:

- Block matching to estimate translational motion of the block,
- Iterative estimation of warping parameters from the best block matched position.

The displacement of a location $\mathbf{x} = [x, y]^T$ resulting from a parametric motion model $\mathbf{B}(\mathbf{x})$ with parameter vector $\mathbf{a}$ can be represented as,

$$\mathbf{d}(\mathbf{x}) = \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}. \tag{3}$$

For instance, the displacement due to an affine motion is,

$$\begin{bmatrix} d_x(x,y) \\ d_y(x,y) \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}. \tag{4}$$

Let s denote the original frame and r the reference frame, then the motion estimation error can be expressed as,

$$\begin{aligned} E(\mathbf{x}) &= s(\mathbf{x}) - r(\mathbf{x} + \mathbf{d}(\mathbf{x})) \\ &= s(\mathbf{x}) - r(\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}). \end{aligned} \tag{5}$$

The task of motion estimation is to find the optimal vector $\mathbf{a}$ such that the motion estimation error for the block is minimized. To this end, Gauss-Newton method for minimization of non-linear equation is utilized. It is an iterative procedure in which each iteration performs a linearization of the problem around the current solution and solves the simplified linear system.

We start by performing a block matching of the current block with the reference picture. The resulting motion vector is utilized to initialize the translational component of the parameter vector $\mathbf{a}$ and the higher order components of $\mathbf{a}$ are set to 0. In each iteration, a vector $\mathbf{\Delta a}_i$ is estimated and is used to update the solution,

$$\mathbf{a} = \mathbf{a}_i + \mathbf{\Delta a}_i \tag{6}$$

Assuming the update vector $\mathbf{\Delta a}_i$ is small, the linearization of the reference picture model can be performed as,

$$\begin{aligned} r(\mathbf{x} + \mathbf{d}(\mathbf{x})) &= r(\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}) \\ &= r(\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}_i + \mathbf{B}(\mathbf{x}) \cdot \mathbf{\Delta a}_i) \\ &\approx r(\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}_i) + \\ &\quad \frac{\partial r}{\partial \mathbf{x}}(\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}_i)\mathbf{B}(\mathbf{x})\mathbf{\Delta a}_i. \end{aligned} \tag{7}$$

Denoting $\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}_i$ as $\mathbf{x}_i$ and $\frac{\partial r}{\partial \mathbf{x}}(\mathbf{x} + \mathbf{B}(\mathbf{x}) \cdot \mathbf{a}_i)\mathbf{B}(\mathbf{x})$ as $\mathbf{g}(\mathbf{x}_i)$, we get $r(\mathbf{x} + \mathbf{d}(\mathbf{x})) = r(\mathbf{x}_i) + \mathbf{g}(\mathbf{x}_i)\mathbf{\Delta a}_i$. The motion estimation error can now be written as,

$$\begin{aligned} E(\mathbf{x}) &\approx s(\mathbf{x}) - r(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_i)\mathbf{\Delta a}_i \\ &= u(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_i)\mathbf{\Delta a}_i, \end{aligned} \tag{8}$$

where, the difference $s(\mathbf{x}) - r(\mathbf{x}_i)$ is denoted by $u(\mathbf{x}_i)$. Considering a set of pixels $\mathbf{x} \in \mathcal{A}$ to represent a block in a video frame, the motion estimation error energy becomes,

$$J = \sum_{\mathbf{x} \in \mathcal{A}} (u(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_i)\mathbf{\Delta a}_i)^2. \tag{9}$$

The error energy in Eq. 9 can be minimized by the standard least-squares minimization, resulting in the optimum update vector $\mathbf{\Delta a}_i$. The new motion parameters are computed according to Eq. 6 and the entire procedure is repeated till maximum number of iterations is reached. It has to be noted that Gauss-Newton procedure might not converge in certain cases (especially for small blocks), therefore after each updation based on the linearized model, the actual error energy is computed. Whenever the error energy increases compared to previously computed parameters, the iteration is stopped and the last good parameters are used as final motion parameters.

V. Signalling of Motion parameters

Efficient encoding of motion parameters is critical for the overall performance of MC prediction. With the increased number of motion parameters, the signalling has to be very efficient in order to compete with the translational mode. It involves,

- Representation of motion paramters so that they can be easily encoded,
- Prediction of the parameters using information from blocks in the neighborhood,
- Entropy coding of the parameters.

In the H.264/AVC standard, where a translational motion model is used, a significant amount of bit-rate reduction in MV signalling is obtained by MV prediction. A median filter operates on the MVs of the causal neighborhood producing a MV predictor for the current block. The MV difference is computed and is coded using arithmetic codes. In order to introduce the option of multiple motion models for each block, it is necessary to take care of the continuity of MV predictors.

1) Motion Parameter Mapping: Before performing prediction, we map the motion parameters to a suitable space because directly using the estimated parameters is not efficient for signalling. Here, we choose the affine motion model as an example of PMM and consider a block of size $M \times N$. A direct representation of $a_0, a_1, \cdots, a_5$ suffers from the fact that a_2, a_5 (refer Eq. 4) do not really represent the translational displacement of the entire block, but that of the top-left pixel in that block. The solution proposed in [4] is to orthonormalize $a_0, a_1, \cdots, a_5$ before quantization. Although, orthonormalization would make the parameters more robust to quantization effects, it cannot be directly used in the median predictor because translational and affine blocks would have different MV quantization steps. Hence, we propose to extract the translational component of the affine motion field by orthogonalization and quantize this component using the same step size as regular MVs (e.g. quarter-pel precision). The warping parameters a_0, a_1, a_3, a_4 are used to compute the relative displacement of the top-left $(0, 0)$ and top-right $(M-1, 0)$ pixels of the considered block with respect to the centroid. Such a mapping to corner motion vector (CMV) enables us to utilize the framework of MV coding also for the warping parameters. The entire process of mapping can be expressed in matrix notation as,

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} \frac{M-1}{2} & \frac{N-1}{2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{M-1}{2} & \frac{N-1}{2} & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ M-1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & M-1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}$$

2) Motion Parameter Coding: The mapped parameters $\mathbf{d}_{cen} = (c_0, c_1)$ contains the translational shift of the centroid of the block and has a high variance. As discussed earlier, $\mathbf{d}_{cen}$ is treated as a regular translational MV and predicted using MVs from neighboring blocks before quantization. The motion of centroid is subtracted from the displacement of top-left and top-right corners resulting in $\mathbf{d}_{left} = (c_2, c_3)$ and $\mathbf{d}_{right} = (c_4, c_5)$. This leads to lower variance of warp

Coding option	Value
Block sizes	64×64 to 4×4
RD optimization	ON
No. ref. pictures	4
Deblocking filter	ON
No. of frames	100

TABLE I
CONFIGURATION SETTINGS FOR REFERENCE AND TEST ENCODERS

parameters which are then quantized to the precision of regular MVs. The quantized differences are coded using the CABAC framework [8] of H.264/AVC, in which they are binarized and entropy coded using an arithmetic coder.

VI. ADAPTIVE SELECTION OF MOTION MODEL

The operational control of the video encoder is an important problem in video coding. Typical video sequences contain varying content and motion, necessitating the selection between different coding options for different parts of the image. We employ the popular Lagrangian approach [9] for choosing the best coding mode suited for the current source signal.

In our design, each block can be predicted as Intra, Inter-translative or Inter-parametric as mentioned earlier. For each block, the coding mode with associated parameters is optimized given the decisions made for prior coded blocks. For each candidate prediction mode, the best residual coding mode is determined and the reconstruction signal is computed. The Lagrangian mode decision for a block proceeds by minimizing

$$J_{mode} = D_{REC} + \lambda_{mode} \cdot R_{REC}, \tag{10}$$

where, λ_{mode} is the lagrangian parameter for the minimization. The distortion D_{REC} is measured as the sum of squared differences of the reconstructed signal and the original signal. The rate R_{REC} is the total rate for signalling the motion parameters, mode information and transform coefficients of prediction residuals.

VII. SIMULATION SETUP AND RESULTS

We evaluate the performace of the proposed system using 12 video sequences from the VCEG/MPEG database using the IPPP coding structure. The reference software used for evaluating our algorithm is based on the H.264/AVC standard and is on an average 10-15 % better than the JM17.0 software due to larger blocks and encoder optimizations. We integrated cubic spline interpolation and affine motion model into the reference software and report the additional gain on top of the improved reference software. We use 100 frames from each of the sequences to execute the tests. The number of reference pictures used for MC prediction is set to 4. The encoder settings used for the tests are summarized in Tab. I. The codec is executed at four different quantization parameter settings, namely $QP = 22, 27, 32, 37$. In the IPPP structure, the first picture is coded as Intra using the user set quantization step QP and the subsequent pictures are coded as P pictures at a step of $QP + 1$, with the option of Inter/Intra modes.

Sequence	Resolution	% Delta Bitrate [10]	
		Spline	Spline+Affine
BasketballPass	416×240	-0.05	-2.03
BlowingBubbles	416×240	-2.28	-3.93
BQSquare	416×240	-10.01	-11.76
RaceHorses	416×240	0.29	-1.84
BasketballDrill	832×480	-1.53	-2.59
BQMall	832×480	-1.19	-2.21
PartyScene	832×480	-5.04	-5.98
RaceHorses	832×480	0.17	-1.07
BQTerrace	1920×1080	-0.59	-1.25
BasketballDrive	1920×1080	0.76	-0.22
Cactus	1920×1080	0.60	-2.94
Wisley	1920×1080	-1.02	-2.64

TABLE II
BITRATE SAVINGS USING IPPP CODING STRUCTURE. THE BITRATE REDUCTION DUE TO CUBIC SPLINE IN COMPARISION TO 6-TAP H.264/AVC INTERPOLATION FILTER IS SHOWN UNDER COLUMN 'SPLINE'. THE OVERALL BITRATE REDUCTION BY INCLUDING AFFINE MOTION MODEL IS SHOWN UNDER COLUMN 'SPLINE+AFFINE'.

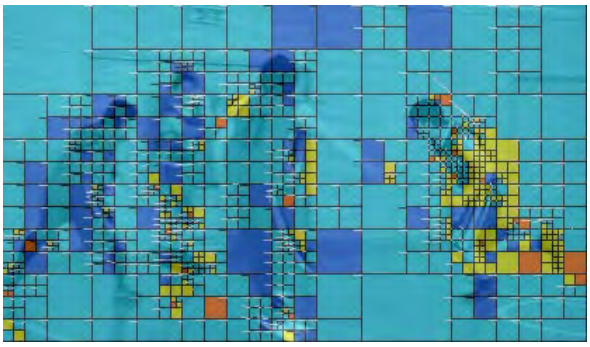
The Inter mode includes Inter-Translative (IT) and Affine Inter-Parametric (AIP) prediction modes. Quarter-pel motion compensation is performed, i.e. the translational MVs in case of IT & AIP and the corner MVs in case of AIP are quantized to quarter-pel precision. Within inter blocks, a mode flag is signalled, using the CABAC entropy coding scheme, which indicates whether the block is coded in IT or AIP mode. The reference codec and the test codec are used to generate 4 RD points each and the performance improvement is measured in terms of Bjøntegaard Delta Bit Rate (BDBR) metric [10].

The test results are shown in Tab. II. There is a significant overall bitrate saving provided by the proposed approach. In particular, the cubic spline interpolation outperforms the existing 6-tap interpolation in H.264 for most of the sequences. For some sequences ('RaceHorses', 'BasketballDrive' & 'Cactus') there is a very small bitrate increase due to cubic spline interpolation. However, the cubic splines provide a good framework for Affine prediction. When cubic splines are used along with Affine prediction, bitrate savings are observed for all the sequences.

In Fig. 3(a), a frame from a video sequence along with the mode decisions (Fig. 3(b)) are depicted. It is a sequence with fast motion, where there is a relatively small shift in the background but the subjects are moving quickly. From the quad-tree splitting shown in Fig. 3(b), it can be noticed that the background is coded with large translational blocks (Fig. 3(b), cyan blocks), whereas the moving subjects are coded with smaller blocks. Also, many blocks in the fast moving region are coded using the affine motion model (Fig. 3(b), blue blocks). During the motion of the subjects, some areas get disoccluded and are coded as intra blocks (Fig. 3(b), yellow & orange blocks).

In Fig. 4, RD curves for two sequences are shown. The RD curves of the reference 6-tap interpolation and cubic spline interpolation are depicted in red and green, respectively. The final result when using both cubic spline interpolation and affine motion model is depicted in blue.

(a) Frame no. 75 of sequence 'Basketball Pass'

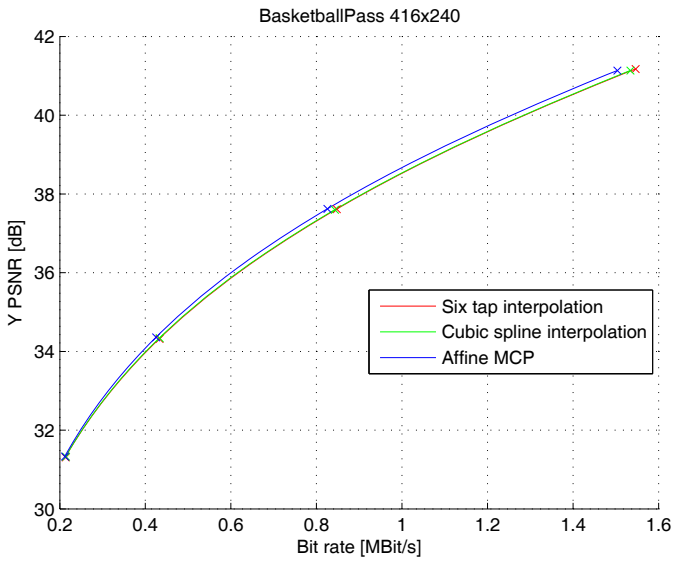


(b) Mode decisions for the frame shown above. Different colors are used to show different modes. Cyan Blocks: Inter-Translative; Blue Blocks: Inter-Parametric; Yellow & Orange Blocks: Intra

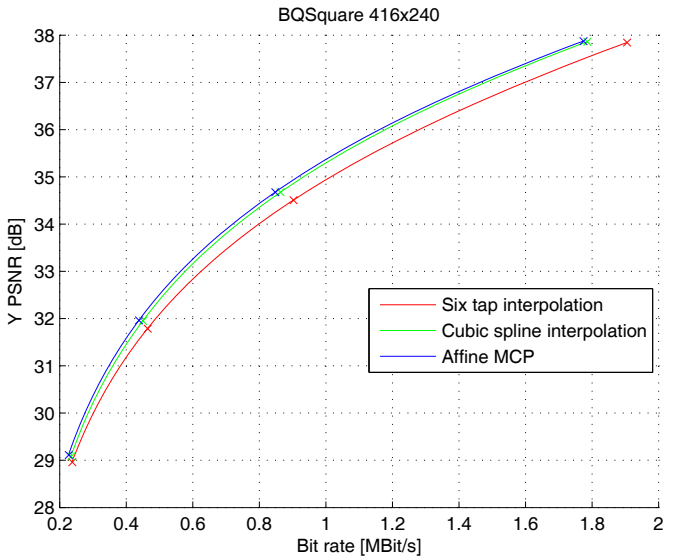
Fig. 3. Figure showing (a) original frame and (b) prediction mode decisions.

VIII. Conclusion

We presented a video coding system with improved interpolation and motion compensation capabilities. The fixed interpolation filter in the H.264/AVC standard is replaced by a spline based framework, which provides a discrete-to-continuous mapping inside integer pixel areas. This mapping is useful for estimation of translational shift and warping parameters for video blocks. The decision whether to transmit warping parameters depends on the RD performance of the different modes. The performance of the proposed algorithm is evaluated on MPEG sequences and bitrate savings up to 11.7% are observed. Future work involves the experimentation with other discrete-to-continous mapping techniques, improving coding of warping parameters and handling aliasing/noise in video sequences.



(a) RD curve 'Basketball Pass' IPPP



(b) RD curve 'BQSquare' GOP8

Fig. 4. Rate-distortion curves for IPPP and GOP8 structures.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, Jul. 2003.

[2] G. Heising, D. Marpe, H. L. Cycon, and A. P. Petukhov, "Wavelet-based Very Low Bit-Rate Video Coding Using Image Warping and Overlapped Block Motion Compensation," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 148, no. 2, pp. 93-101, Apr. 2001.

[3] Nokia MVC Codec. *ITU-T Proposal*, Feb. 2000. http://ftp3.itu.ch/av-arch/video-site/0002_Gen/

[4] M. Karczewicz, J. Nieweglowski and P. Haavisto, "Video coding using motion compensation with polynomial motion vector fields," *Signal Processing: Image Commun.*, vol. 10, pp. 63-91, 1997.

[5] M. Unser, "Splines: A Perfect Fit for Signal and Image Processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22-38, Nov. 1999.

[6] P. Chen, Y. Ye, M. Karczewicz, "Video coding using extended block sizes", *VCEG-AJ23*, San Diego, USA, 8-10 Oct. 2008.

[7] M. Unser, A. Aldroubi, M. Eden, "B-Spline Signal Processing: Part II - Efficient Design and Applications," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 834-848, Feb. 1993.

[8] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620-636, Jul. 2003.

[9] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-Constrained Coder Control and Comparison of Video Coding Standards", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688-703, Jul 2003.

[10] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," ITU-T VCEG Meeting, Austin, Texas, USA, Document VCEG-M33, Tech. Report, Apr. 2001.