Dynamic Motion Estimation for Transcoding P Frames in H.264 to MPEG-2 Transcoders

Hari Kalva, *Senior Member*, IEEE and Phil Kunzelmann, *Student Member*, IEEE

Abstract — *The efficiency of H.264 video compression together with the wide use of MPEG-2 today necessitates transcoding between these two technologies. The combination of bandwidth efficient technology such as H.264 and the widely deployed MPEG-2 allows for innovative video delivery services that can exploit both these technologies. Using H.264 for video delivery and MPEG-2 for playback on the end user equipment fully leverages the benefits of these technologies. This architecture requires transcoding H.264 video delivered over networks to MPEG-2 format for playback on the end-user device. We present algorithms for transcoding H.264 to MPEG-2 and evaluate the performance of these algorithms. The performance is compared with a baseline transcoder with a full decoding stage and a full encoding stage. The proposed algorithms reduce the complexity by reducing the search space for motion estimation. The results show that the proposed algorithms reduce the motion estimation complexity by over 80% with a negligible loss in PSNR.*

Index Terms — H.264 transcoding, MPEG-2 transcoding, motion estimation, complexity.

I. INTRODUCTION

The MPEG-2 video coding standard is being used in virtually all broadcast video applications today including digital TV and DVD. The H.264 specification represents a significant advance in the state of video coding technology and provides equivalent video at an average of half the MPEG-2 bitrates [1]. The compression efficiency alone, however, is not sufficient to replace MPEG-2 with H.264 in the short term. The wide use of MPEG-2 represents an investment of billions of dollars in MPEG-2 infrastructure; overcoming this investment inertia requires time. In the near term MPEG-2 and H.264 will co-exist and transcoding between these formats is necessary. Transcoding MPEG-2 to H.264 is being actively studied and a number of papers have been published in this area [2-4]. The compression efficiency of H.264 and the wide availability of MPEG-2 make possible new video service architectures that can use H.264 for bandwidth efficient delivery and MPEG-2 for playback over the widely available end-user equipment. The H.264 coded video can be delivered over IP networks and can be transcoded at the head end of a Cable TV network for delivery as MPEG-2 video. Alternatively, the H.264 video is downloaded to a set-top-box or digital video recorder (DVR) where it is transcoded for playback in MPEG-2 on the existing digital TV or DVR. The placement of the transcoder depends on the application needs. Complexity reduction is essential to support faster transcoding either at the head end or at the user premises.

A transcoder essentially consists of a decoding state that decodes the input video either fully or partially and then encodes it to the target video format. The general approaches to complexity reduction in transcoding attempt to reuse the information gathered in the decoding stage to reduce the complexity of the encoding stage thereby reducing the complexity of the transcoding operation. Where there have been several efforts transcoding MPEG-2 to H.264, the problem of transcoding H.264 to MPEG-2 is not well studied. The present problem of transcoding H.264 to MPEG-2 is different and the methods developed for MPEG-2 to H.264 cannot be applied. The only work we have come across is the H.264 to MPEG-2 transcoder reported in [5]. This work directly derives MPEG-2 motion vectors from the H.264 motion vectors but results in PSNR loss of over 3 dB.

The key to reducing the complexity of the transcoder is reusing the motion vector and MB mode information to reduce the transcoding complexity. Reuse of H.264 motion information for MPEG-2 encoding, however, is problematic given the advanced features of H.264. Variable block-size motion compensation and increased pixel search accuracy in particular make a direct translation of H.264 motion vectors difficult and are prone to introducing error. An accurate translation becomes more difficult as the number of sub-blocks for a particular 16x16 MB increases. Each additional sub-block introduces the possibility a motion vector was found in a different direction. Using an average of these vectors to arrive at a single 16x16 vector becomes less reliable as the mean angle between the vectors increases. The key components of the proposed motion estimation algorithms are: 1) MB mode mapping, 2) Motion vector estimation 3) Dynamic search range and, 4) Dynamic window. These approaches require slightly more, but still negligible, computation than direct motion vector mapping proposed in [5] but the resulting drop in PSNR is negligible at less than 0.3 dB in the worst case. This paper presents and evaluates algorithms to exploit the motion information to reduce the complexity of the MPEG-2 encoding stage.

The rest of the paper is organized as follows: Section 2 summarizes motion estimation in H.264 and MPEG-2, the

[1] This work was supported in part by a grant from RealNetworks Inc..

The authors are with the Multimedia Lab, Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA. Dr. Hari Kalva can be reached at hari@cse.fau.edu or https://www.cse.fau.edu/~hari

Manuscript received December 20, 2007

H.264 motion information reuse is discussed in Section 3. Results are discussed in Section 4 and conclusions are presented in Section 5.

II. MOTION ESTIMATION

Since motion estimation takes the most resources in the MPEG-2 encoding stage, our efforts to reduce the complexity of the transcoder focus on the motion estimation process. The motion compensation in MPEG-2 and H.264 differ substantially. The H.264 motion compensation is more sophisticated and this motion information has to be reused in the MPEG-2 encoding stage. The key problem here is selecting the MPEG-2 motion vector based on the H.264 motion information.

A. Motion Compensation in H.264

The motion compensation of macro blocks in H.264 uses variable block sizes and motion vectors with quarter-pixel resolution. A macro block can be coded as one 16x16 or two 16x8 or two 8x16 or four 8x8 blocks. Each 8x8 block can in turn be coded as one 8x8 or two 4x8 or two 8x4 or four 4x4 sub-blocks. As many as 16 different reference frames can be used and the actual number of reference frames is constrained by the buffer size in a particular profile and level. A B MB in H.264 has a generalized definition: a B MB is predicted from any two predictions as opposed to one forward and one backward prediction used in MPEG-2 video coding. Furthermore, a B picture can also be used as a reference picture. These motion compensation tools, though complex, contribute significant coding gains in H.264.

B. Motion Compensation in MPEG-2

Compared to H.264, the motion compensation in MPEG-2 is relatively simple. Each macroblock is coded as a 16x16 block and predictions are formed for 16x16 blocks (or 16x8 for field prediction). A P MB has one forward motion vector and a B MB can have one forward, or one backward, or a forward and a backward motion vectors. Furthermore, MPEG-2 does not allow the use of B frames as reference frames.

III. REUSING H.264 MOTION INFORMATION FOR TRANSCODING

A. Dynamic Range

As an initial application of transcoding methods that make best use of information gained during the decoding stage, a dynamic range may be implemented for an MPEG-2 ME search. Dynamic ranging adjusts ME search range parameters to the size of the H.264 MVs with the greatest magnitudes. In this way, MPEG-2 ME is provided with the largest possible set of candidate blocks indicated by H.264. In addition, dynamic ranging allows ME search space to be greatly reduced particularly where MBs with low magnitude MVs are encountered. Our prior work on complexity reduction using dynamic range is reported in [6]. The search range for motion estimation in MPEG-2 is adjusted for every MB based on the motion vectors of the MB in H.264. The range for X and Y direction is selected independently as depicted in Fig. 1.

B. Motion Vector Selection

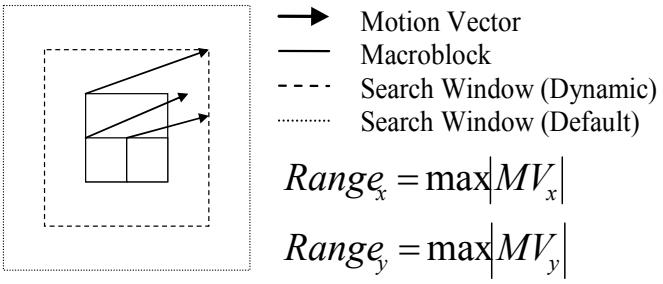


Fig. 1 Dynamic Range Selection

While using dynamic search rage reduces the search space for motion estimation, this is based on the longest H.264 motion vector and results in unnecessary computations. The complexity can be further reduced by limiting motion estimation to a region of the frame where motion occurred. We refer to this as the dominant motion direction and derive a single dominant motion vector using the H.264 motion information that could contain up to 16 motion vectors. The dominant motion vector is then used in determining the dynamic range.

MV Selection methods provide a translation for multiple motion vectors to a single motion vector. Three methods for this translation have been tested experimentally: Selection by Mean, Coordinate and Angle. MV Selection applies only to Dynamic Windowing algorithms (which apply the selected MV to center the search window).

MV Mean

The most straightforward method for MV selection is to take a simple average. For mean MV selection, a vector average is taken for a set of H.264 MVs to derive a single 16x16 MV. As the number of MVs in H.264 increases the MV derived using the mean metric deviated more from the actual MPEG-2 MV. Using the MV mean thus doesn't work well for the general case.

MV Voting by Position

Simulations show that a simple mean can potentially introduce a significant drop in PSNR. As the differences between sub-block MV magnitudes and positions increases the derived motion vector becomes less reliable.

The potential for an unreliable average is the motivation for MV voting. The purpose of this algorithm is to select only the H.264 MV most similar to other MVs. This method is

intended to select only the MV most likely to produce a reliable 16x16 MB match. MV voting by position is a simple algorithm that selects a MV from a set by lowest absolute difference from other members.

$$MV = \min\left\|MV_{selected}\right| - \left|MV_n\right\|, n = 1...N$$

MV Voting by Angle

It is also possible that a set of MVs may provide a better indication of the general direction of motion. MV voting by angle is an algorithm that discards magnitude for selection by directional difference only. Similar to MV voting by position, MV voting by angle uses the tan2 function (to avoid tangent periodicity) to select a set member by lowest absolute difference.

$$MV = \min(\sum_{n=1}^{N}\left|\tan(\frac{MVY_{selected}}{MVX_{selected}}) - \tan(\frac{MVY_n}{MVX_n})\right|), n = 1..N$$

C. Dynamic Window

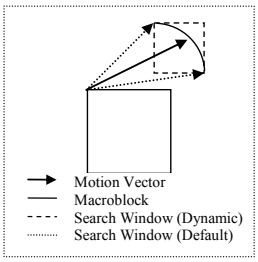
A dynamic range allows for a significant reduction of ME search space through adjustment of search window dimensions. A further reduction in candidate block space is possible by adjusting search window position as well. The 'fixed' windowing algorithms proposed use MV selection methods to center the MPEG-2 ME search window on a small set of candidates indicated by the selected MV. This window around a candidate MV is usually referred to as the refinement window. The ideal case for this algorithm is one where the refinement window may be reduced to a single pel and the selected H.264 MV is effectively reused. Simulations, however, show direct H.264 MV reuse to be problematic for retention signal quality. It follows that as the size of the ME refinement window increases, and so the number of potential candidates for a match, image quality for a fixed bitrate will increase. For this reason, simulations for 1-pel, 3-pel and 9-pel fixed refinement windows are examined for comparative PSNR.

Dynamic Refinement Window

The Dynamic Windowing algorithm presented, using a fixed size refinement window, is a fairly simple approach and does not work well with smaller windows. Using a larger window could increase unnecessary computation. A more informed approach, however, is one that considers the orientation of MVs in selecting the refinement window size. Through an application of refinement window sizing by angular variance, however, it is possible to reduce the complexity further. This proposed algorithm first uses an MV selection method to derive a single vector. The variance is then computed for the angle of this vector and other members of its MB. The variance in angle is then used to derive a chord from points above and below the selected motion vector. The endpoints of this chord are then used to determine the dimensions of the search window.

IV. Results and Discussion

Simulations run for the following experiments use JM 10.2 H.264 and MPEG-2 reference software from the MPEG



$$\theta = \arctan 2(MV_y, MV_x)$$

$$\sigma_\theta^2 = \frac{\sum_{n=1}^{N}(\theta_n - \theta_{selected})^2}{N}$$

$$\rho = \sqrt{MV_x^2 + MV_y^2}$$

$$X_{range} = \left\|\cos(\theta - \sigma_\theta^2) \bullet \rho\right| - MV_x\left|\right.$$

$$Y_{range} = \left\|\sin(\theta - \sigma_\theta^2) \bullet \rho\right| - MV_x\left|\right.$$

Fig. 2. Dynamic refinement window derivation

Software Simulation Group. To remove exogenous factors, intra (spatial) modes were disabled for all encoded sequences. Each video sequence used was encoded with the following configuration: 150 frames, 720x480 resolution, YUV 4:2:0 chroma format, GOP size of 15 with I and P frames only. The H.264 input was encoded with QP of 22 (giving a PSNR of ~43 dB) and the MPEG-2 output was generated for seven different bitrates from 6 Mbps to 12 Mbps. PSNR was calculated from the decoded H.264 and reconstructed MPEG-2 YUV frames. Transcoder complexity was measured as total time spent for the MPEG-2 encoding process only (as this was the only variable). Elapsed time was measured in milliseconds and all sequences were encoded using a DELL Optiplex GX620 with a 3.0 GHz Intel Pentium D and 1 GB of RAM.

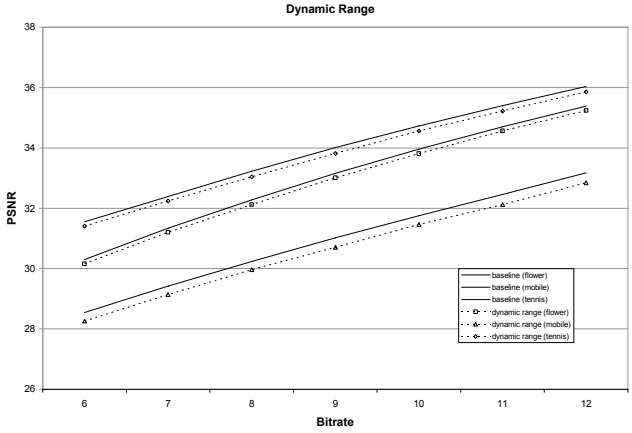
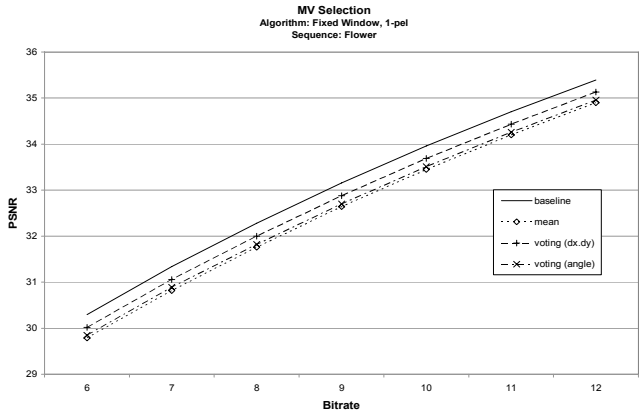
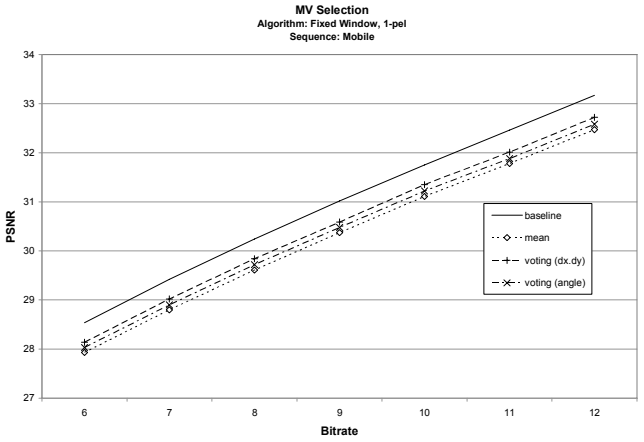


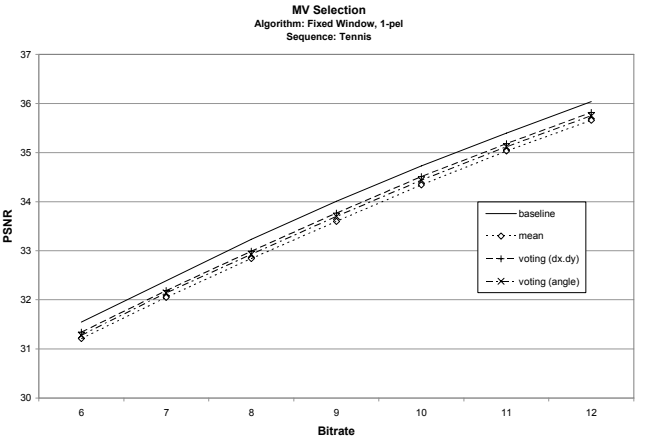
Fig. 3. RD performance of dynamic motion search range



(a)

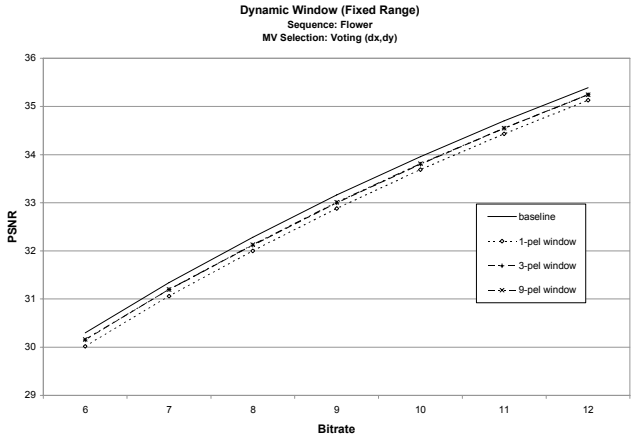


(b)

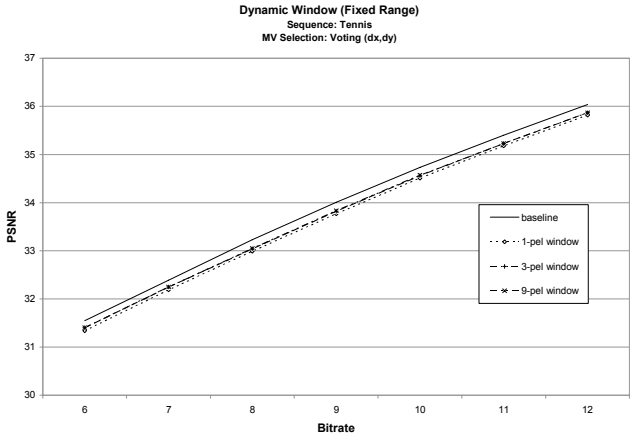


(c)

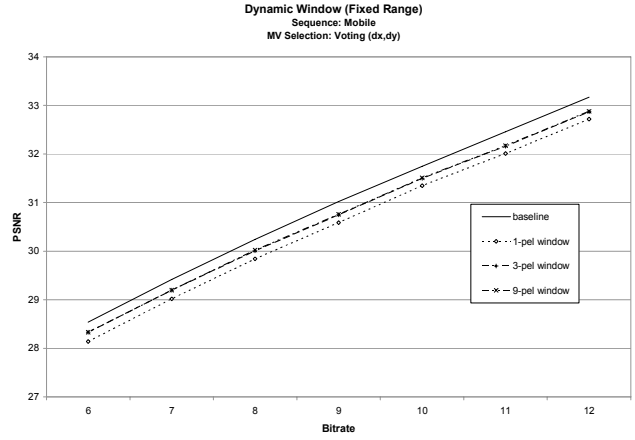
Fig. 4. Comparison of MV selection methods for (a) Flower (b) Tennis and (c) Mobile sequences



(a)



(b)



(c)

Fig. 5. Fixed MV refinement window performance (a) for (a) Flower (b) Tennis and (c) Mobile sequences

Figure 3 shows the results of performing motion estimation using dynamic range. The motion search range is determined based on the largest x and y components of the H.264 motion vectors. The RD performance is close to baseline but still suffers a loss is PSNR of close to 0.2dB on average. The loss in PSNR is due to the differences in the motion estimation in MPEG-2 and H.264 that result in very different motion vectors. The loss in PSNR is due to smaller MVs selected in H.264 due to the variable block sizes used for motion estimation. When motion vectors are long, the search range is usually close to the full search of 16 used in the baseline

Table 1: Total Time Summary for 3 pel refinement window

Dynamic Window / 3-pel / Voting (dx,dy) (Total Time)										
Bitrate (Mbps)	d Time (%)							Avg. Gain	Max. Δ PSNR	Min. Δ PSNR
	6	7	8	9	10	11	12			
flower	65.29%	63.58%	63.39%	63.20%	63.03%	62.92%	62.73%	**63.45%**	-0.16	-0.14
hook	60.60%	58.70%	58.53%	58.44%	58.29%	58.16%	58.00%	**58.67%**	-0.05	-0.02
mobile	66.65%	65.17%	64.94%	64.82%	64.71%	64.47%	64.33%	**65.01%**	-0.30	-0.21
tennis	82.80%	66.06%	65.90%	65.79%	65.67%	65.53%	65.41%	**68.17%**	-0.19	-0.14

Table 2: Motion Estimation Time Summary for Dynamic refinement window

Dynamic Window / 3-pel / Voting (dx,dy) (ME Time)										
Bitrate (Mbps)	d Time (%)							Avg. Gain	Max. Δ PSNR	Min. Δ PSNR
	6	7	8	9	10	11	12			
flower	89.38%	88.88%	89.00%	89.33%	89.42%	88.97%	89.07%	**89.15%**	-0.16	-0.14
hook	83.26%	83.18%	82.66%	82.85%	83.72%	83.43%	83.24%	**83.19%**	-0.05	-0.02
mobile	88.89%	88.88%	89.14%	88.93%	88.65%	88.87%	89.12%	**88.93%**	-0.30	-0.21
tennis	88.41%	81.95%	81.55%	81.50%	81.74%	81.50%	81.57%	**82.60%**	-0.19	-0.14

Table 3: Total Time Summary for Dynamic refinement window

Dynamic Window / Range by Angle Variance / Voting (dx,dy) (Total Time)										
Bitrate (Mbps)	d Time (%)							Avg. Gain	Max. Δ PSNR	Min. Δ PSNR
	6	7	8	9	10	11	12			
flower	64.96%	63.26%	63.02%	62.88%	62.67%	62.58%	62.40%	**63.11%**	-0.16	-0.14
hook	58.96%	57.02%	56.88%	56.75%	56.62%	56.48%	56.31%	**57.00%**	-0.05	-0.02
mobile	66.28%	64.79%	64.52%	64.39%	64.27%	64.07%	63.98%	**64.62%**	-0.30	-0.21
tennis	82.48%	65.46%	65.35%	65.24%	65.13%	64.99%	64.86%	**67.64%**	-0.19	-0.14

encoder and the best match found would be close to the baseline transcoder. The computational savings due to the dynamic range can be improved using the dynamic window approach.

Figure 4.a,b,c shows the performance of three MV selection methods discussed in Section 3. MV Selection by coordinate voting showed the best PSNR for all the sequences tested. The plots shown are for a 1-pel refinement window; i.e., the motion vector selected is refined in a motion search window of radius 1 pel. The difference between the baseline and voting by coordinate is at most 0.3 dB with a refinement window of 1. The PSNR difference between selection methods was negligible as the refinement window increased in size. The PSNR difference with baseline also reduces with increased refinement window. Refinement windows of 3, 6, and 9 were tested; the plots in these cases are not show due to space constraints. The MV selection by voting was adopted for MV selection in all other experiments.

Figure 5 shows the RD curves for varying refinement window. In this experiment, the MV selection based on voting is used and the motion vector is refined in a fixed window of 1 or 3 or 9 pixels. The results show that the RD performance improves with the refinement window size and the results hold across all tested sequences. The MV selection based on voting give a very good estimate of MPEG-2 MV and there is virtually no difference between a 3 pel and a 9-pel window.

Figure 6 shows the RD curves for the dynamic refinement window determined using the variance of the MV angle as described in Section 3. The results show that the performance is very close to the baseline transcoder. The MV selection here is based on voting.

The performance summary is shown in the tables 1-3 for the total time and motion estimation time. The proposed transcoder reduces the motion estimation time by over 80% and the total time by over 65%. Compared with the fixed window approach, the time complexity results are similar to that of 3 pixel refinement window and PSNR is about 0.2 dB better.

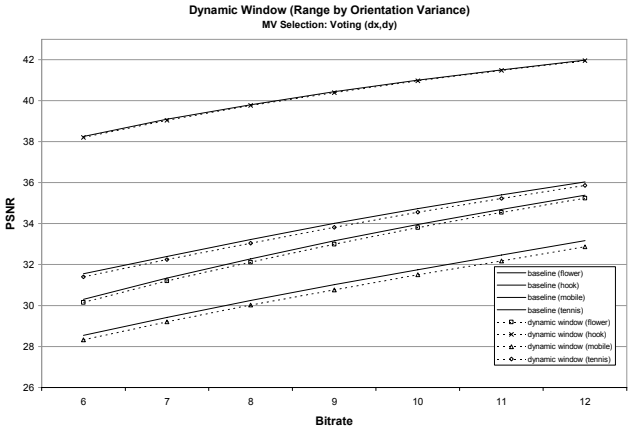


Fig. 6. Dynamic MV refinement window performance

V. CONCLUSION

The H.264 to MPEG-2 transcoding makes possible new types of services that can leverage the efficiency of H.264 and the wide availability of MPEG-2 infrastructure. This paper presents tools for P frame transcoding in H.264 to MPEG-2 transcoders. The proposed approaches exploit the motion information gathered in the H.264 decoding stage to first derive a dominant motion vector and then refine the motion vector. A dynamic refinement widow was developed as fixed refinement windows cannot perform well for a general case. The results show that the proposed approaches improve the transcoder performance by reducing the motion estimation time by over 80% and total MPEG-2 encoding time by over 65% resulting in a PSNR drop of less than 0.3 dB.

REFERENCES

[1] T. Wiegand, G.J. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Transactions on Circuits and Systems for Video Technology, vol.13, no.7pp. 560- 576, July 2003.

[2] H. Kalva and B. Petljanski, "Exploiting the directional features in MPEG-2 for H.264 intra transcoding," IEEE Transactions on Consumer Electronics, vol.52, no.2, May 2006, pp. 706- 711.

[3] J. Xin, A. Vetro, S. Sekiguchi, and K. Sugimoto, "MPEG-2 to H.264/AVC Transcoding for Efficient Storage of Broadcast Video Bitstreams," International Conference on Consumer Electronics, 2006. pp. 417- 418, Jan. 2006.

[4] G. Fernandez-Escribano, H. Kalva, P. Cuenca, and L. Orozco-Barbosa, "Very Low Complexity MPEG-2 to H.264 Transcoding Using Machine Learning," Proceedings of the ACM Multimedia 2007, Santa Barbara, CA, October 2006, pp. 931-940.

[5] J. Chu et. al., "H.264/MPEG-2 Transcoding based on Personal Video Recorder Platform," Proceedings of the Ninth International Symposium on Consumer Electronics pp. 438–440, June 2005.

[6] P. Kunzelmann and H. Kalva, "Reduced complexity MPEG_2 to H.264 transcoder," Proceedings of the International Conference on Consumer Electronics (ICCE 2007), Las Vegas, January, 2007.

Hari Kalva (M'92-SM'05) Dr. Kalva joined the Department of Computer Science and Engineering at Florida Atlantic University as an Assistant Professor in August 2003. He was a co-founder and the Vice President of Engineering of Flavor Software, a New York company founded in 1999, that developed MPEG-4 based solutions for the media and entertainment industry. Dr. Kalva is an expert on digital audio-visual communications systems with over 12 years of experience in multimedia research, development, and standardization. He has made key contributions to the MPEG-4 Systems standard and also contributed to the DAVIC standards development. His research interests include pervasive media delivery, content adaptation, video compression, and communication. He has over 70 published papers and seven patents (10 pending) to his credit. He is the author of one book and co-author of five book-chapters.

Dr. Kalva received a Ph.D. and an M.Phil. in Electrical Engineering from Columbia University in 2000 and 1999 respectively. He received an M.S. in Computer Engineering from Florida Atlantic University in 1994, and a B. Tech. in Electronics and Communications Engineering from N.B.K.R. Institute of Science and Technology, S.V. University, Tirupati, India in 1991.

Phil Kunzelmann (M'06) Phil Kunzelmann is a recent M.Sc. graduate of the Dept. of Computer Science and Engineering at Florida Atlantic University. His recent research has focused on reduced complexity H.264 to MPEG-2 and MPEG-2 to H.264 transcoding.