Fast Search for Long-Term Memory Motion-Compensated Prediction

Thomas Wiegand[1], Bo Lincoln[2], and Bernd Girod[1]

[1]Telecommunications Laboratory
University of Erlangen-Nuremberg
Cauerstr. 7/NT, D-91058 Erlangen, Germany
wiegand|girod@nt.e-technik.uni-erlangen.de

[2]Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
bosse@stanford.edu

Abstract

Long-term memory motion-compensated prediction extends the spatial displacement utilized in block-based hybrid video coding by a frame reference parameter permitting the use of many previously decoded pictures. This extension of the motion search range significantly increases the motion-compensated prediction gain. However, the amount of motion search related computation is significantly increased by the new approach as well. Based on the triangle inequality, we investigate a modified motion search order especially suited for the long-term memory approach. The concept of half-pel refinement is incorporated into the new search ordering method. A hierarchy of triangle inequalities provides an additional speed-up at the cost of memory. It is demonstrated that lossy methods based on the hierarchy of triangle inequalities give additional speed-ups at small losses of prediction gain. At very minor losses in prediction gain, for the sequences Foreman, Mother-Daughter and Stefan a reduction in computation time by factors of 11.2, 8.1, and 3.6, respectively, is reported, when searching over 50 frames.

1 Introduction

The improved rate-distortion performance of long-term memory motion-compensating prediction has been shown by integrating long-term memory prediction into an H.263 codec [1, 2]. The bit-rate savings achieved by using 50 instead of 1 frame in our rate-distortion optimized H.263-based video codec are 23 % for the sequence *Foreman* and 17 % for the sequence *Mother-Daughter* [1]. The codec utilizes a sliding window of past, decoded frames. Hence, if the long-term memory buffer comprises M frames, a decoded frame is searched M times in the block motion estimation of M successive frames. Therefore, any pre-computation associated with to a decoded frame in the long-term memory can be re-used M times to speed up motion estimation.

Several methods are known to speed-up motion estimation that are based on mathematical inequalities [3, 4]. These inequalities, e.g., the triangle inequality, give a lower bound on the norm of the difference between vectors. In block matching, the search criteria very often used for the distortion are the sum of the absolute differences (SAD) or the sum of the squared differences (SSD) between the motion-compensated prediction $c[x,y]$ and the original signal $s[x,y]$. By incorporating the triangle inequality into the sums for SAD and SSD, we obtain

$$D(s,c) = \sum_{[x,y]\in\mathcal{B}} \left|s[x,y] - c[x,y]\right|^p \geq \hat{D}(s,c) = \left|\left(\sum_{[x,y]\in\mathcal{B}} \left|s[x,y]\right|^p\right)^{\frac{1}{p}} - \left(\sum_{[x,y]\in\mathcal{B}} \left|c[x,y]\right|^p\right)^{\frac{1}{p}}\right|^p, \quad (1)$$

with $p = 1$ for SAD and $p = 2$ for SSD. Note that for $p = 2$, the inequality used in [4] differs from (1). Empirically, we have found little difference between those inequalities. For some blocks, the inequality used in [4] provides a more accurate bound whereas for other blocks the triangle inequality performs better. The set $\mathcal{B}$ comprises the sampling positions of the blocks considered, e.g., a block of 16×16 samples.

Assume D_{min} to be the smallest distortion value previously computed in the block motion search. Then, the distortion $D(s,c)$ of another block c in our search range is guaranteed to exceed D_{min} if the lower bound of $D(s,c)$ exceeds D_{min}. More precisely, reject block c if

$$\hat{D}(s,c) \geq D_{min}. \quad (2)$$

The special structure of the motion estimation problem permits a fast method to compute the norm values of all blocks $c[x,y]$ in the previously decoded frames [3].

0-8186-8821-1/98 $10.00 © 1998 IEEE

AMAZON-1018
8,204,134

2 Search Order

A small value for D_{min} determined in the beginning of the search leads to the rejection of many other blocks later and thus reduces computation. Hence, the order in which the blocks in the search range are tested has a great impact on the computation time. For example, given the Huffman code tables for the motion vectors as prior information about our search space, the search ordering should follow increasing bit-rate for the motion vectors. This way, we maximize the probability to find a good match in the search at the beginning. A good approximation of these probabilities is a search spiral, as the one used in the test model for the H.263 standard [5]. Hence, for long-term memory motion search, a spiral ordering for each frame can be used when searching over the M frames in the long-term memory buffer.

If a partial observation of the current block is available, the motion vector probability may differ significantly from these long-term statistics that were used to derive the Huffman codes. Much better guesses for the probability of finding a good match in the search space are given by the values of the block norms. More precisely, a block $c[x, y]$ that has a norm similar to the block $s[x, y]$ is very likely to be a good match. Hence, we propose to conduct the motion search in the order of the absolute differences between the norm of $s[x, y]$ and the norms of the blocks in the search range $c[x, y]$. The *norm-ordered search* stops if (2) is violated. Thus, the algorithm does not even have to test those positions which cannot yield distortion values lower than the minimum previously found.

3 Half-Pel Refinement

When searching in spiral order over the M frames in the long-term memory buffer, the method we choose for half-pel refinement in [1] is the following: for each frame, find the best motion vector by full search on integer-pel positions using the spiral ordering followed by half-pel refinement, and determine the final motion vector by choosing among the motion vectors found for each frame.

Using the norm-ordered search, the half-pel refinement procedure needs to be modified since we are no longer searching in all the frames in the long-term memory. Instead, we keep the best N integer-pel vectors found throughout the norm-ordered search and conduct the half-pel refinement procedure for these candidates. The final motion vector is determined by choosing among these half-pel refined candidates.

In Table 1, several approaches to motion search using the triangle inequality are compared for three test sequences: *Foreman*, *Mother-Daughter*, and *Stefan*. These sequences were chosen so as to represent extreme cases among the publicly available test sequences.

The experiment conducted to produce the results in Table 1 as well as all following results is to predict blocks of size 16×16 in frames 200 to 299 of those sequences using M past original frames that are sampled at 10 Hz. The motion search finds the corresponding block in the M past frames in a spatial search range of $[-15..15] \times [-15..15]$ that minimizes the SSD. When computing the SSD of the 16×16 block, after each line of 16 samples, the distortion computed so far is compared against D_{min} and the current block is rejected if D_{min} is exceeded. The experiments are conducted on a 300 MHz Sun UltraSPARC, single processor, 1GB RAM, Solaris 2.5. No VIS instructions are used. Note that the results highly depend on the machine used.

M	1		10		50	
	PSNR	T	PSNR	T	PSNR	T
Foreman						
F	27.84	0.34	29.12	4.21	29.68	23.74
S	27.84	0.26	29.12	3.02	29.68	17.29
N_M	27.85	0.26	29.15	1.90	29.72	9.94
N_V	27.85	0.26	29.13	1.54	29.71	6.72
Mother-Daughter						
F	40.46	0.20	41.07	2.27	41.27	15.66
S	40.46	0.13	41.07	1.49	41.27	11.28
N_M	40.46	0.14	41.09	1.51	41.27	9.57
N_V	40.46	0.14	41.07	1.19	41.27	5.86
Stefan						
F	20.37	0.47	21.16	5.71	22.16	30.86
S	20.37	0.46	21.16	5.45	22.16	28.83
N_M	20.39	0.57	21.19	5.34	22.20	25.96
N_V	20.39	0.57	21.18	4.74	22.18	21.12

Table 1: Comparison of spiral-ordered full search, spiral-ordered search using the triangle inequality, and norm-ordered search while varying memory size over 1, 10, 50 and the number of positions for half-pel refinement. The abbreviations are: M: memory, F: spiral-ordered full search, S: spiral-ordered search using the triangle inequality, N_M: norm-ordered search using the triangle inequality when half-pel refining the M best integer-pel vectors, N_V: norm-ordered search using the triangle inequality when half-pel refining a variable number of best integer-pel vectors, where the numbers are $N = 1$ for $M = 1$, $N = 5$ for $M = 10$, and $N = 10$ for $M = 50$. The PSNR values in dB represent the prediction error variance. Higher values indicate smaller prediction errors. T stands for the computation time in seconds per frame.

In Table 1, full search (F), is compared against three search methods using the triangle inequality. These methods are spiral-ordered search (S), and two norm-ordered search methods (N_M and N_V). Note, without loss in prediction efficiency (PSNR), the overall computation time (T) is reduced by a factor of up to 3.5 for the *Foreman* sequence for $M = 50$ when comparing the results for search methods F and N_V. However, there is no benefit in using our method for $M = 1$ frame, indicating, that in our current implementation, the norm-ordering produces too much overhead.

4 Multiple Triangle Inequalities

Assume a partitioning of $\mathcal{B}$ into subsets $\mathcal{B}_n$, such that

$$\mathcal{B} = \bigcup_n \mathcal{B}_n, \qquad \text{and} \qquad \bigcap_n \mathcal{B}_n = \emptyset. \tag{3}$$

The triangle inequality (1) holds for all possible subsets $\mathcal{B}_n$. Rewriting the formula for $D(s,c)$ we obtain

$$\sum_{[x,y]\in\mathcal{B}} \left|s[x,y]-c[x,y]\right|^p = \sum_n \sum_{[x,y]\in\mathcal{B}_n} \left|s[x,y]-c[x,y]\right|^p \tag{4}$$

and applying the triangle inequality for each $\mathcal{B}_n$ yields

$$D(s,c) = \sum_{[x,y]\in\mathcal{B}} \left|s[x,y]-c[x,y]\right|^p \geq$$

$$\sum_n \left| \left(\sum_{[x,y]\in\mathcal{B}_n} \left|s[x,y]\right|^p \right)^{\frac{1}{p}} - \left(\sum_{[x,y]\in\mathcal{B}_n} \left|c[x,y]\right|^p \right)^{\frac{1}{p}} \right|^p \tag{5}$$

Note that (5) is a tighter lower bound than (1), however, requires more computation. Hence, at this point we can trade-off the tightness of the lower bound against computational complexity.

An important issue within this context is the best choice of the partitions $\mathcal{B}_n$. In [4], it is proposed to partition a square 16×16 block in two different ways. The first partitioning produces 16 subsets $\mathcal{B}_n$ each being one of 16 lines containing 16 samples. The second partitioning consists of 16 subsets $\mathcal{B}_n$, each being one of 16 columns containing 16 samples.

Note that the H.263 video coding standard permits blocks of size 16×16 and blocks of size 8×8 in the advanced prediction mode. Hence, we follow the approach proposed in [6] where a 16×16 block is decomposed into 4 different partitions. The 16×16 block is partitioned into 1 set of 16×16 samples, into 4 subsets of 8×8 samples, into 16 subsets of 4×4 samples and, into 64 subsets of size 2×2 samples. The various (subset) triangle inequalities are successively applied in the order of the computation time to evaluate them, i.e., first the 16×16 triangle inequality is checked, then the inequalities relating to blocks of size 8×8, 4×4, and 2×2 samples are computed using (5).

Table 2 presents the comparison of the norm-ordered search (N_V in Tab. 1) to the approach of additional comparisons on the 8×8, 4×4 and 2×2 block level (H_4). Again, for $M = 1$ frames, the methods proposed are not beneficial. However, as we search over more frames such as $M = 50$, a speed-up factor of 1.4 against the norm-ordered search (N_V) and 5.0 against full search (method F in Tab. 1) is achieved for the sequence *Foreman*.

M	1		10		50	
	PSNR	T	PSNR	T	PSNR	T
Foreman						
N_V	27.85	0.26	29.13	1.54	29.71	6.72
H_4	27.85	0.22	29.13	1.11	29.71	4.73
Mother-Daughter						
N_V	40.46	0.14	41.07	1.19	41.27	5.86
H_4	40.46	0.15	41.07	1.05	41.27	5.00
Stefan						
N_V	20.39	0.57	21.18	4.74	22.18	21.12
H_4	20.39	0.57	21.18	4.59	22.18	20.91

Table 2: Comparison of the norm-ordered search (N_V in Tab. 1) to the approach of additional comparisons on the 8×8, 4×4 and 2×2 block level (H_4). The experimental settings and abbreviations are the same as for Tab. 1.

5 Lossy Search Methods Using the Triangle Inequality

In many applications the computation time available is very often not sufficient to conduct a full block motion search. Hence, the computation time needs to be reduced, which, in general, results in a reduced prediction gain. Two ideas have mainly been emphasized in the context of lossy search methods:

1. sub-sampling of the block for which distortion is measured and
2. reduction of the search range.

Item 1 reduces the computation time to check a particular block by evaluating the distortion criterion on a smaller amount of samples. When using a hierarchy of triangle inequalities, the sub-sampling can be incorporated by stopping at a certain level in the hierarchy. We adapt the level in the hierarchy by measuring the amount of activity in the block for which the motion search is conducted. The activity is measured as the sum of the absolute differences of neighboring samples. Then, if the mean activity measure is below a

certain threshold, we measure distortion only on the 2×2 block level of the triangle inequality hierarchy and never evaluate the distortion on the sample level. We found that if the samples on average change in the range of $1-2$ intensity values, the search can be conducted on the 2×2 block level of the triangle inequality hierarchy with very minor losses. Note that this lossy method only affects the integer-pel search and that we left the half-pel search unchanged for all experiments.

Item 2, the reduction of the search space, is realized by early termination of the norm-ordered search. For that (2) is modified to

$$\hat{D}(s,c) \cdot K \geq D_{min}, \tag{6}$$

where K is adapted to a value $K \geq 1$. A heuristic to adapt to the variance of the norm histogram in the search range is to set $K = 150 \cdot l/L$, where l is the number of searched positions checked so far, and L is the total number of positions in the search range. The number 150 has been obtained empirically.

M	1		10		50	
	PSNR	T	PSNR	T	PSNR	T
Foreman						
H_4	27.85	0.22	29.13	1.11	29.71	4.73
N_1	27.85	0.22	29.13	1.09	29.70	4.54
N_2	27.61	0.15	28.99	0.63	29.62	2.19
N_{12}	27.61	0.15	28.99	0.61	29.62	2.11
Mother-Daughter						
H_4	40.46	0.15	41.07	1.05	41.27	5.00
N_1	40.45	0.14	41.03	0.86	41.26	4.10
N_2	40.43	0.13	41.04	0.60	41.27	2.31
N_{12}	40.42	0.12	41.02	0.52	41.26	1.92
Stefan						
H_4	20.39	0.57	21.18	4.59	22.18	20.91
N_1	20.39	0.57	21.18	4.56	22.18	20.39
N_2	20.28	0.28	21.11	2.02	22.14	8.51
N_{12}	20.28	0.28	21.11	2.00	22.14	8.38

Table 3: Comparison of the two lossy methods related to item 1 (N_1), item 2 (N_2), and their combination (N_{12}). These three lossy search methods are compared to the values relating to method H_4 in Tab. 2, i.e., the fastest non-lossy method. Otherwise the settings are the same as for Tables 1 and 2.

Table 3 shows a comparison of the two lossy methods and the fastest non-lossy method. When applying the lossy method related to N_1, we only obtain significant gains for *Mother-Daughter*. The results for method N_2 show significant gains for all sequences, while the combination of the lossy methods N_{12} additionally improves for all sequences at very moderate PSNR losses. For the sequence *Foreman*, when searching over 50 frames, at a PSNR loss of 0.09 dB (for method N_{12}) a speed-up of 2.2 is achieved against the fastest non-lossy method (H_4) and against full search (F) our method is 11.2 times faster.

6 Final Remarks

We also apply our fast search method when incorporating a rate constraint into the minimization criterion for the motion estimation [1, 7]. As in [7], the computation time decreases with increasingly weighted rate constraint. Also, when using distorted instead of original frames as reference frames we observe only a small increase in computation time which we also observe for the spiral search. At the cost of increased memory requirements, the methods described in this paper reduce computation time for long-term memory motion search. At minor losses in PSNR, for the sequences *Foreman*, *Mother-Daughter* and *Stefan*, a reduction in computation time by factors of 11.2, 8.1, and 3.6, respectively, is reported, when searching over 50 frames.

7 Acknowledgment

The authors would like to thank Michael Todd Malkin for his work in the area of half-pel refinement. Furthermore, they wish to thank Klaus Stuhlmüller, Barry Andrews, and Paul Ning for useful discussions.

References

[1] T. Wiegand, X. Zhang, and B. Girod, "Motion-Compensating Long-Term Memory Prediction", in *Proc. ICIP*, Santa Barbara, USA, Oct. 1997.

[2] T. Wiegand, X. Zhang, and B. Girod, "Long-Term Memory Motion-Compensated Prediction", *IEEE TR-CSVT*, to appear Sep./Oct. 1997.

[3] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation", *IEEE TR-IP*, vol. 4, no. 1, pp. 105–107, Jan. 1995.

[4] Y.-C. Lin and S.-C. Tai, "Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression", *IEEE TR-COM*, vol. 45, no. 5, pp. 527–531, May 1997.

[5] Telenor Research, "TMN (H.263) Encoder/Decoder, Version 2.0", Download via anonymous ftp to `bonde.nta.no`, June 1997.

[6] C.-H. Lee and L.-H. Chen, "A Fast Search Algorithm for Vector Quantization Using Mean Pyramids of Codewords", *IEEE TR-COM*, vol. 43, no. 2/3/4, pp. 604–612, Feb./Mar./Apr. 1995.

[7] M. Coban and R. M. Mersereau, "Computationally Efficient Exhaustive Search Algorithm for Rate-Constrained Motion Estimation", in *Proc. ICIP*, Santa Barbara, USA, Oct. 1997.