Improved fast block matching algorithm in feature domain

Yiu-Hung Fok and Oscar C. Au

Department of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
Email: eeau@uxmail.ust.hk

Abstract

A fast block matching algorithm in the feature domain was proposed by Fok and Au with a computation reduction factor of $N/2$ for a search block size of $N \times N$. Although the algorithm can achieve close-to-optimal result, it requires a large amount of memory to store the features. This paper presents three improved fast block matching algorithms in the integral projections feature domain which can also reduce the computation significantly but with a considerably lower memory requirement. With a search block size of $N \times N$, two of our algorithms retain a computation reduction factor of $N/2$ while the other one can achieve a computation reduction factor of N . The three algorithms can achieve close-to-optimal performance in mean absolute difference(MAD) sense.

Keywords: block matching algorithm, feature domain, integral projections, motion estimation

1. INTRODUCTION

The raw data rate of video sequences is usually very large. Video compression is needed to reduce the video data rate for transmission and storage purposes. The most popular video compression standards include CCITT H.261 and MPEG, both of which use block-based motion compensation to reduce the temporal redundancy in image sequences. Brute force (exhaustive search) motion estimation is optimal in the mean absolute difference (MAD) sense but also requires prohibitively large amount of computation. As a result, different fast sub-optimal block matching algorithms had been proposed.¹⁻⁵ A fast block matching algorithm in the feature domain (FBMA) was proposed by Fok and Au⁶ achieving a computation reduction factor of $N/2$ when compared with the conventional exhaustive search (CES) algorithm. Here the basic search blocks are of size $N \times N$ and the location of a block is given by the location of the pixel at the top-left corner of that block. This algorithm first calculates the Horizontal Integral Projections(HIP) and the Vertical Integral Projections(VIP) of the current frame k . The HIP and the VIP at the (x,y) th location of the k -th frame are defined in equation (1) and (2) respectively. They are:

$$HIP_k(x, y) = \sum_{n=0}^{N-1} f_k(x+n, y) \quad (1)$$

$$VIP_k(x, y) = \sum_{n=0}^{N-1} f_k(x, y+n) \quad (2)$$

where $f_k(x,y)$ is the pixel value at the (x,y) th location of frame k .

The best R search locations are found according to a distance measure which was defined as Mean Absolute Difference(MAD) in the HIP and the VIP feature domains given in equation (3).

$$MAD_f(i, j) = \sum_{n=0}^{N-1} |HIP_k(x, y+n) - HIP_{k-1}(i, j+n)| + |VIP_k(x+n, y) - VIP_{k-1}(i+n, j)| \quad (3)$$

where (i,j) is the search location in the search area of the previous frame and (x,y) is the location of the block in the current frame.

In order to improve the matching result, these R locations are re-examined using another distance measure, the MAD in the pixel domain given in equation (4). The one with the least MAD is declared the best match. This step is called Re-examination.⁶

$$MAD_p(i,j) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} |f_k(x+m, y+n) - f_{k-1}(i+m, j+n)| \quad (4)$$

This algorithm achieves a computation reduction factor of $N/2$ for a block size of $N \times N$. However, this algorithm requires a large amount of memory to store the HIP and the VIP for a large frame.

In this paper, we propose several improvements to FBMA resulting in better performance and lower memory requirement while retaining the computation reduction factor of $N/2$.

2. ALGORITHMS

There are three improved fast block matching algorithms in the feature domain. They are based on the idea of using the alternating features when calculating the cost function in the feature domains. They are:

1. The Alternating Integral Projections Search(AIPS).
2. The Alternating Half Integral Projections Search(AHIPS).
3. The Alternating Reduced Half Integral Projections Search(ARHIPS).

2.1. Alternating Integral Projections Search(AIPS)

With frame size $Q \times P$ and block size $N \times N$, we transform both the even rows and the even columns of the previous frame from the pixel domain to the HIP and VIP domains. For an $N \times N$ block in the current frame, we calculate the HIP and the VIP in the following pattern.

Figure 1: Alternate IP Pattern for a block in current frame.

At each location in the search area, we calculate the cost function using the alternating features, i.e., either the white or the shaded HIP and VIP in Figure 1. If the block is in an even row of the search area (even j), we use the white HIP to calculate the MAD in the feature domain. i.e.

$$MAD_{even}^{row}(i,j) = \sum_{n=0}^{N/2-1} |HIP_k(x, y+2n) - HIP_{k-1}(i, j+2n)| \quad (5)$$

If the block is in an odd row of the search area (odd j), we use the shaded HIP to calculate the MAD in the feature domain. i.e.

$$MAD_{odd}^{row}(i, j) = \sum_{n=0}^{N/2-1} |HIP_k(x, y + 2n + 1) - HIP_{k-1}(i, j + 2n + 1)| \quad (6)$$

Similarly, if the block is in an even column of the search area (even i), we use the white VIP to calculate the MAD in the feature domain. i.e.

$$MAD_{even}^{col}(i, j) = \sum_{n=0}^{N/2-1} |VIP_k(x + 2n, y) - VIP_{k-1}(i + 2n, j)| \quad (7)$$

If the block is in an odd column of the search area (odd i), we use the shaded VIP to calculate the MAD in the feature domain. i.e.

$$MAD_{odd}^{col}(i, j) = \sum_{n=0}^{N/2-1} |VIP_k(x + 2n + 1, y) - VIP_{k-1}(i + 2n + 1, j)| \quad (8)$$

The cost in the feature domain for this block is defined as the sum of the two MAD described above. Therefore, there are four possible combinations of costs depending on the location of the block in the search area:

$$MAD_f(i, j) = \begin{cases} MAD_{even}^{row}(i, j) + MAD_{even}^{col}(i, j) & \text{for even } i, \text{ even } j \\ MAD_{even}^{row}(i, j) + MAD_{odd}^{col}(i, j) & \text{for odd } i, \text{ even } j \\ MAD_{odd}^{row}(i, j) + MAD_{even}^{col}(i, j) & \text{for even } i, \text{ odd } j \\ MAD_{odd}^{row}(i, j) + MAD_{odd}^{col}(i, j) & \text{for odd } i, \text{ odd } j \end{cases} \quad (9)$$

Finally, we re-examine the R best candidates by calculating the MAD in the pixel domain defined in equation (4) and find the best match.

2.2. Alternating Half Integral Projections Search(AHIPS)

In this case, we define the Half Horizontal Integral Projection(HHIP) as the sum of the pixel values in only one half of a row, i.e.

$$HHIP_k(x, y) = \sum_{n=0}^{N/2-1} f_k(x + n, y) \quad (10)$$

where $f_k(x, y)$ is the pixel value at the (x, y) th location of the frame k .

Similarly, we define the Half Vertical Integral Projection(HVIP) as the sum of the pixel values in only one half of a column, i.e.

$$HVIP_k(x, y) = \sum_{n=0}^{N/2-1} f_k(x, y + n) \quad (11)$$

To represent a block, according to its location, we use N white or shaded HHIP and N white or shaded HVIP as shown in Figure 2.

For an $Q \times P$ frame, with block size $N \times N$, we first transform both the even rows and the even columns of the previous frame from the pixel domain to the HHIP and the HVIP domains. For an $N \times N$ block in the current frame, we calculate the HHIP and the HVIP in the following pattern.

Figure 2: Alternating Half IP Pattern for a block in current frame.

At each location in the search area, we calculate the cost function using the alternating features, i.e., either the white or the shaded HHIP and HVIP. If the block is in an even row of the search area (even j), we use the white HHIP to calculate the MAD in the feature domain. i.e.

$$MAD_{even}^{row}(i, j) = \sum_{n=0}^{N/2-1} \{ |HHIP_k(x, y + 2n) - HHIP_{k-1}(i, j + 2n)| + |HHIP_k(x + N/2, y + 2n) - HHIP_{k-1}(i + N/2, j + 2n)| \} \quad (12)$$

If the block is in an odd row of the search area (odd j), we use the shaded HHIP to calculate the MAD in the feature domain. i.e.

$$MAD_{odd}^{row}(i, j) = \sum_{n=0}^{N/2-1} \{ |HHIP_k(x, y + 2n + 1) - HHIP_{k-1}(i, j + 2n + 1)| + |HHIP_k(x + N/2, y + 2n + 1) - HHIP_{k-1}(i + N/2, j + 2n + 1)| \} \quad (13)$$

Similarly, if the block is in an even column of the search area (even i), we use the white HVIP to calculate the MAD in the feature domain. i.e.

$$MAD_{even}^{col}(i, j) = \sum_{n=0}^{N/2-1} \{ |HVIP_k(x + 2n, y) - HVIP_{k-1}(i + 2n, j)| + |HVIP_k(x + 2n, y + N/2) - HVIP_{k-1}(i + 2n, j + N/2)| \} \quad (14)$$

Algorithm	Additional Memory Requirement	Computation Complexity
CES	0	$PQ(2W+1)^2$
FBMA	$[(P-N+1)Q + (Q-N+1)P]*4$	$2PQ(2W+1)^2/N$
AIPS	$(P-N+1)Q + (Q-N+1)P$	$PQ(2W+1)^2/N$
AHIPS	$(P-N/2+1)Q + (Q-N/2+1)P$	$2PQ(2W+1)^2/N$
ARHIPS	$[(P-N/2+1)Q + (Q-N/2+1)P]/2$	$2PQ(2W+1)^2/N$

Table 1: Memory and computation requirement of different algorithms

In terms of computation complexity, FBMA, AHIPS and ARHIPS retain the computation reduction factor of $N/2$. Notice that, AIPS achieves a computation reduction factor of N compared with CES.

For AIPS and AHIPS, the memory requirements are reduced by a factor of four when compared to FBMA⁶, because only the HIP for the even rows and the VIP for the even columns of the previous frame are stored for calculating the cost function in the feature domain. The HIP and the VIP of the current frame are not stored and they are calculated during searching of the current block. The price paid for this is a slight increase of computation which is less than $2PQ$. A further reduction by a factor of two can be achieved by ARHIPS, since FBMA, AIPS and AHIPS use two bytes for storage of each integral projection while ARHIPS uses only one byte to store an integral projection. Obviously, with fewer bits or lower resolution of the features, ARHIPS should not be as good as AHIPS.

In fact, with careful design of algorithms, we only need to store no more than $2Q(W+N)$ features for the previous frame. It is because when the search location moves down by one row the integral projections for that row are no longer needed. The memory used to store these integral projections can then be used to store the new integral projections for the row at the bottom of the new search area. It is illustrated as Figure 3.

Figure 3: Smart use of memory.

4. SIMULATION RESULTS

Our algorithms were simulated using 200 frames of the "Football" and the "Tennis" sequences. Each frame contained 352×240 pixels quantized uniformly to 8 bits. Only the luminance component was considered. The size of the block was 16×16 . The maximum displacement in the search space was ± 16 pixels in both the horizontal and the vertical directions. For re-examination in each algorithm, we used five best candidates in the feature domains.

Figure 4 and 5 show the MAD and the Peak Signal-To-Noise Ratio (PSNR) between the 80th to 110th estimated frames and the original frames of the "Football" sequence. Also, Figure 6 and 7 show the MAD and the PSNR between the 100th to 130th estimated frames and the original frames of the "Tennis" sequence. Assuming the range of the pixel value is

Algorithm	Additional Memory Requirement	Computation Complexity
CES	0	$PQ(2W+1)^2$
FBMA	$[(P-N+1)Q + (Q-N+1)P]*4$	$2PQ(2W+1)^2/N$
AIPS	$(P-N+1)Q + (Q-N+1)P$	$PQ(2W+1)^2/N$
AHIPS	$(P-N/2+1)Q + (Q-N/2+1)P$	$2PQ(2W+1)^2/N$
ARHIPS	$[(P-N/2+1)Q + (Q-N/2+1)P]/2$	$2PQ(2W+1)^2/N$

Table 1: Memory and computation requirement of different algorithms

In terms of computation complexity, FBMA, AHIPS and ARHIPS retain the computation reduction factor of $N/2$. Notice that, AIPS achieves a computation reduction factor of N compared with CES.

For AIPS and AHIPS, the memory requirements are reduced by a factor of four when compared to FBMA⁶, because only the HIP for the even rows and the VIP for the even columns of the previous frame are stored for calculating the cost function in the feature domain. The HIP and the VIP of the current frame are not stored and they are calculated during searching of the current block. The price paid for this is a slight increase of computation which is less than $2PQ$. A further reduction by a factor of two can be achieved by ARHIPS, since FBMA, AIPS and AHIPS use two bytes for storage of each integral projection while ARHIPS uses only one byte to store an integral projection. Obviously, with fewer bits or lower resolution of the features, ARHIPS should not be as good as AHIPS.

In fact, with careful design of algorithms, we only need to store no more than $2Q(W+N)$ features for the previous frame. It is because when the search location moves down by one row the integral projections for that row are no longer needed. The memory used to store these integral projections can then be used to store the new integral projections for the row at the bottom of the new search area. It is illustrated as Figure 3.

Figure 3: Smart use of memory.

4. SIMULATION RESULTS

Our algorithms were simulated using 200 frames of the "Football" and the "Tennis" sequences. Each frame contained 352×240 pixels quantized uniformly to 8 bits. Only the luminance component was considered. The size of the block was 16×16 . The maximum displacement in the search space was ± 16 pixels in both the horizontal and the vertical directions. For re-examination in each algorithm, we used five best candidates in the feature domains.

Figure 4 and 5 show the MAD and the Peak Signal-To-Noise Ratio (PSNR) between the 80th to 110th estimated frames and the original frames of the "Football" sequence. Also, Figure 6 and 7 show the MAD and the PSNR between the 100th to 130th estimated frames and the original frames of the "Tennis" sequence. Assuming the range of the pixel value is

from 0 to 255, PSNR is given by

$$PSNR = -10 \log_{10} \frac{\sum_{x=0}^P \sum_{y=0}^Q [f(x,y) - \tilde{f}(x,y)]^2}{Q \times P \times 255^2} \quad (19)$$

where $f(x,y)$ and $\tilde{f}(x,y)$ are the original frame and the estimated frame of size $Q \times P$ respectively.

These figures show the general profiles of the performance of different algorithms. As expected, the CES has the smallest MAD along all the algorithms considered. In the figures, the dotted lines with crosses and the dashed lines correspond to ARHIPS and AHIPS respectively. Both the MAD and the PSNR of ARHIPS are very closed to those of AHIPS, and effectively they are on top of each other. For the "Football" and the "Tennis" sequences, the MAD of FBMA is on the average 2.5% and 1.8% larger than that of CES⁶ respectively. The following table shows the percentages of MAD of different algorithms on the average larger than that of the CES for the both sequences.

Algorithm	"Football" sequence	"Tennis" sequence
FBMA	2.5%	1.8%
AIPS	3.6%	3.02%
AHIPS	1.33%	1.07%
ARHIPS	1.34%	1.08%

Table 2: Percentages of MAD of different algorithms on the average lager than that of CES

The following table shows the percentages of PSNR of different algorithms on the average smaller than that of the CES for the both sequences.

Algorithm	"Football" sequence	"Tennis" sequence
FBMA	0.74%	0.53%
AIPS	1.13%	0.84%
AHIPS	0.41%	0.21%
ARHIPS	0.41%	0.21%

Table 3: Percentages of PSNR of different algorithms on the average smaller than that of CES

Notice that, in 34 out of 200 frames of the "Football" sequence, the PSNR of ARHIPS can even be larger than or equal to those of the CES. While in 56 out of 200 frames of the "Tennis" sequence, the PSNR of ARHIPS were actually larger than or equal to those of the CES.

Figure 8 shows the estimated 82th frames of AIPS, AHIPS, ARHIPS and the conventional exhaustive search of "Football" sequence. Figure 9 shows the estimated 82th frames of AIPS, AHIPS, ARHIPS and the conventional exhaustive search of the "Tennis" sequence. Both frames are with lots of motion. In terms of subjective image quality, the estimated frames of AIPS, AHIPS and ARHIPS are very closed to and sometimes better than those of the CES.

5. CONCLUSION

In this paper, we present three fast motion estimation algorithms in the feature domain: AIPS, AHIPS and ARHIPS. The computation requirements of these algorithms are approximately equal to FBMA. The memory requirements for these three algorithms are greatly reduced when compared with FBMA. In fact, with careful design of algorithms, the memory requirement can be further reduced. The conventional exhaustive search algorithm is optimal with respect to MAD, but simulation shows that our algorithms can achieve close-to-optimal MAD performance. AHIPS and ARHIPS are found to perform better than AIPS with respect to MAD and PSNR. On the other hand, with a block size of $N \times N$, AIPS achieves a computation reduction by a factor of N while AHIPS and ARHIPS achieve a computation reduction factor of $N/2$. This provides a trade off between the image quality and the computational complexity.

6. ACKNOWLEDGMENTS

This project is supported in part by HKTIIT grant #HKTIIT 92/93.001 and in part by RGC grant #HKUST 195/93E.

7. REFERENCES

1. J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. Commun., vol. COM-29, December 1981, pp. 1799-1808.
2. J. S. Kim and R. H. Park, "A fast feature-based block matching algorithm using integral projections," IEEE Journal on Selected Areas in Communications, vol. 10, no. 5, June 1992, pp. 968-979.
3. Q. Wang and R. J. Clarke, "Motion estimation and compensation for image sequence coding," Signal Processing: Image Communication, vol. 4, 1992, pp. 167-174.
4. S. Kappagantula and K. R. Rao, "Motion Compensated Interframe Image Prediction," IEEE Trans. Commun., vol. COM-33, no. 9, September 1985, pp. 1011-1015.
5. T. Koga, K. Iinuma, A. Hirano, Iilima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in Proc. Nat. Telecommun. Conf., New Orleans, L. A., Nov. 29-Dec. 3, 1981, pp. G5.3.1-5.3.5.
6. Y. H. Fok and O. Au, "A Fast Block Matching Algorithm in Feature Domain," Proc. of IEEE Workshop on Visual Signal Processing and Communications, Melbourne, 21-22 September 1993, pp. 199-202.

Figure 4: MAD for Football Sequence using different algorithms.

Figure 5: PSNR for Football sequence using different algorithms.

Figure 6: MAD for Tennis sequence using different algorithms.

Figure 7: PSNR for Tennis sequence using different algorithms.

CES	AIPS
ARHIPS	AHIPS

Figure 8: The Estimated 82th frames for Football sequence using different algorithms.
 Top-left: CES. Top-right: AIPS. Bottom-left: ARHIPS. Bottom-right: AHIPS.

CES	AIPS
ARHIPS	AHIPS

Figure 9: The Estimated 82th frames for Tennis sequence using different algorithms.
 Top-left: CES. Top-right: AIPS. Bottom-left: ARHIPS. Bottom-right: AHIPS.