# PIXEL DECIMATION IN BLOCK MATCHING TECHNIQUES

Ing. Peter RADOCZI
Prof. Ing. Dušan LEVICKÝ, CSc.
Dept. of Electronics and Multimedia Communication
Technical University of Košice
Park Komenského 13, 040 01Košice
Slovak Republic

## Abstract

*Block motion estimation using full search algorithm is computationally extensive. Previously proposed fast algorithms reduce the computation cost by limiting the number of locations searched. In this paper we present algorithms for block motion estimation that produce similar performance to that full search algorithm. The algorithms are based on the pixel decimation.*

## Keywords

motion estimation, block matching, pixel decimation

## 1. Introduction

In digital television coding or videotelephoning, one attempts to minimise the temporal redundancy in a sequence of images so as to reduce the transmission data rate. Motion compensated interframe coding is a most efficient approach to realise this goal. In motion compensated interframe prediction, the frame to frame displacement of a moving object in scene is estimated, and the prediction is formed by displacing the previous frame elements.

It is well known that the success of motion compensated prediction depends on the ability to estimate the displacement of moving object. Several mathematical models have been proposed to estimate the motion vector fields in a sequence of pictures. They can be classified basically into two groups: the block matching motion estimation algorithms and recursive (gradient) motion estimation algorithms.

Among various criteria that can be used as a measure of the match between the two block, the mean squared error (MSE) and the mean absolute difference (MAD) are favoured towards normalised cross correlation function [1],[6]. If a maximum displacement of $\mathbf{d_m}$ pixel/frame (Fig.1) is allowed for both the horizontal and vertical directions, there are $(2d_m+1)^2$ locations for the best match to the present block.

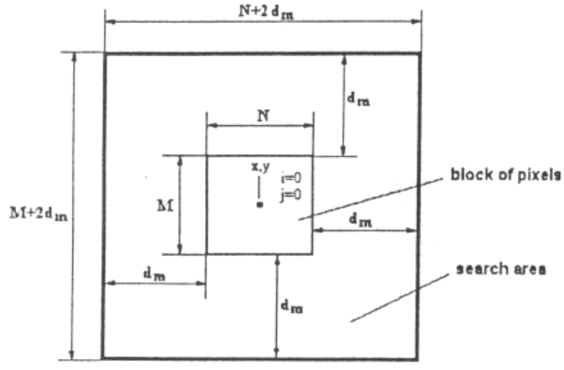


Fig.1 Depiction of the search area

The fundamental idea behind the block matching algorithms is that the motion estimation can be obtained by searching the position of the best match in the previous frame. The full search algorithm is straightforward to find the optimal match in a search area, but an enormous amount of computation is required. Therefore, most of efforts in the block matching methods are to seek an efficient search procedure to reduce the computational complexity but without an obvious effect on the estimation accuracy, such as some proposals like the 3-step search algorithm [2], the 2-D logarithmic search algorithm [3], the conjugate search algorithm [4], etc. Most of these fast algorithms are based on the assumption that the distortion criterion is quadrantmonotonous in the search area. However, this assumption is not always true.

## 2. Description of proposed pixel decimation techniques

When matching a block from the present frame to a block from the previous frame, the matching criterion is evaluated using every pixel of the block. Since block matching is based on the assumption that all pixels in a block move the same amount, a good estimate of the motion could be obtained, in principle, by using only a fraction of the pixels in a block. If few pixels are used, however, there will eventually be a reducing in a accuracy of the motion estimates. In this paper we present a pixel-decimation techniques using subsampling by factor of 2 and 4 that preserve the motion estimation accuracy.

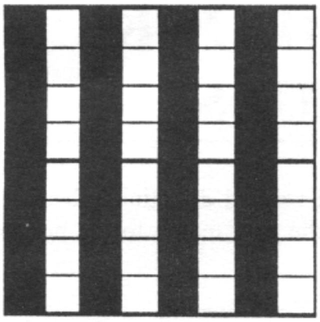

Fig. 2a Vertical pixel decimation

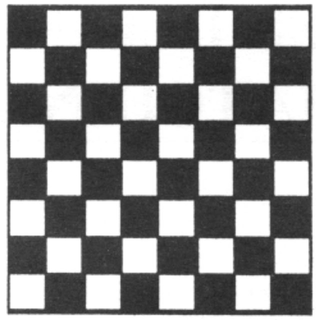

Fig. 2b Chess pixel decimation

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |
| a | b | a | b | a | b | a | b |
| c | d | c | d | c | d | c | d |

Fig. 3a Decimation by Liu and Zaccarin

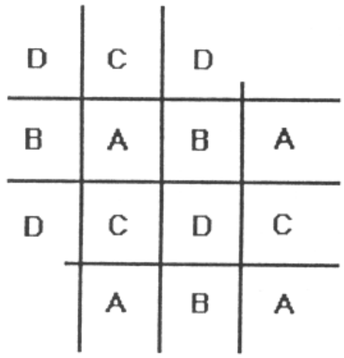


Fig. 3b Location of patterns over the search area

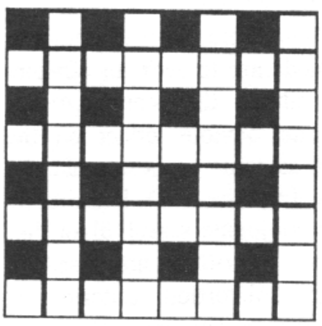

Fig. 4a Orthogonal selection of pixels

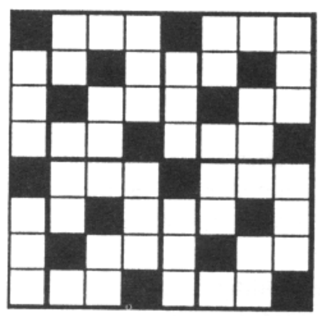

Fig. 4b Nonorthogonal selection of pixels

Fig. 5a. Talk.raw

5b. Goose.raw

5c. Walk.raw

Tab. 1a Talk.raw

| dm | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| all pixels | 26.1939 | 32.3711 | 32.4845 | 32.5496 | 32.5946 | 32.6149 | 32.6221 | 32.6343 | 32.6472 | 32.6628 |
| Pd2 vert | 25.9821 | 32.1891 | 32.2972 | 32.3229 | 32.3937 | 32.4184 | 32.4143 | 32.3959 | 32.4014 | 32.3972 |
| Pd2 chess | 26.1528 | 32.2948 | 32.3887 | 32.4370 | 32.4446 | 32.4890 | 32.4976 | 32.5099 | 32.5145 | 32.5334 |
| Pd2 adapt | 26.1498 | 32.3078 | 32.1443 | 31.9724 | 32.0533 | 32.0026 | 31.9842 | 31.9479 | 31.9378 | 31.9177 |
| Pd4 orth | 25.7129 | 31.9604 | 31.9450 | 31.8236 | 31.8222 | 31.8906 | 31.8628 | 31.7190 | 31.7079 | 31.6562 |
| Pd4 adapt | 26.0788 | 31.6607 | 30.9249 | 31.1203 | 30.8011 | 31.1792 | 30.3394 | 30.0245 | 29.8782 | 29.6742 |
| Pd4 LiuZacc | 25.2645 | 31.4290 | 31.3758 | 31.1832 | 31.0705 | 31.0824 | 31.0530 | 31.0089 | 31.0466 | 31.0530 |
| Pd4 nonorth | 25.8942 | 32.0946 | 32.1551 | 32.0811 | 32.1111 | 32.1110 | 32.1051 | 32.0927 | 32.0113 | 32.1111 |

Tab. 1b Goose.raw

| dm | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| all pixels | 28.3451 | 29.2390 | 29.9412 | 30.5145 | 31.0322 | 31.7265 | 32.2298 | 32.5542 | 32.7581 | 32.8865 |
| Pd2 vert | 28.2755 | 29.1253 | 29.7855 | 30.3312 | 30.8417 | 31.4667 | 31.9342 | 32.2467 | 32.4176 | 32.5514 |
| Pd2 chess | 28.3067 | 29.1768 | 29.8771 | 30.4317 | 30.9360 | 31.6219 | 32.1036 | 32.4212 | 32.6177 | 32.7361 |
| Pd2 adapt | 28.2000 | 28.9521 | 29.5421 | 29.9915 | 30.4903 | 31.1431 | 31.5829 | 31.8304 | 31.9694 | 32.0552 |
| Pd4 orth | 28.1598 | 28.9312 | 29.4766 | 29.9426 | 30.3720 | 30.9047 | 31.3543 | 31.6462 | 31.8228 | 31.9087 |
| Pd4 adapt | 28.0749 | 28.4670 | 28.8296 | 29.1297 | 29.2147 | 29.9695 | 30.3199 | 30.4970 | 30.5211 | 30.5796 |
| Pd4 LiuZacc | 28.0272 | 28.7179 | 29.3037 | 29.7755 | 30.2640 | 30.8211 | 31.2364 | 31.5416 | 31.6676 | 31.7733 |
| Pd4 nonorth | 28.1403 | 28.9635 | 29.5837 | 30.1024 | 30.5371 | 31.2242 | 31.6533 | 31.8975 | 32.0619 | 32.1710 |

Tab. 1c Walk.raw

| dm | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| all pixels | 18.5237 | 19.5741 | 20.5289 | 21.5248 | 22.7091 | 24.0174 | 25.1052 | 25.6863 | 25.9127 | 26.0837 |
| Pd2 vert | 18.4803 | 19.4809 | 20.3840 | 21.2540 | 22.4390 | 23.7301 | 24.7744 | 25.2378 | 25.4252 | 25.5554 |
| Pd2 chess | 18.5005 | 19.5352 | 20.4586 | 21.4317 | 22.6249 | 23.9148 | 24.9714 | 25.5368 | 25.7443 | 25.8804 |
| Pd2 adapt | 18.4041 | 19.3469 | 20.1385 | 20.9746 | 22.0670 | 23.0724 | 24.1250 | 24.9438 | 24.9187 | 25.0069 |
| Pd4 orth | 18.3936 | 19.3865 | 20.1658 | 21.0632 | 22.1836 | 23.4324 | 24.3698 | 24.8569 | 24.9488 | 25.1319 |
| Pd4 adapt | 18.2909 | 19.1338 | 19.7713 | 20.3531 | 21.0927 | 21.9992 | 22.5741 | 23.1403 | 23.2753 | 23.4165 |
| Pd4 LiuZacc | 18.3260 | 19.3138 | 20.1734 | 20.9752 | 22.1068 | 23.2589 | 24.3001 | 24.7842 | 24.9501 | 25.0907 |
| Pd4 nonorth | 18.3878 | 19.3583 | 20.2334 | 21.1815 | 22.2714 | 23.4441 | 24.4832 | 25.0204 | 25.1929 | 25.3280 |

Tab. 2

| dm=6 | TALK.RAW | | | GOOSE.RAW | | | WALK.RAW | | |
|---|---|---|---|---|---|---|---|---|---|
| | all pixels | pd2 chess | pd4 nonorth | all pixels | pd2 chess | pd4 nonorth | all pixels | pd2 chess | pd4 nonorth |
| FS | 32,6149 | 32,4890 | 32,1110 | 31,7265 | 31,6219 | 31,2242 | 24,0174 | 23,9148 | 23,4441 |
| TSS | 29,2641 | 28,7812 | 28,3827 | 31,1929 | 31,0839 | 30,5857 | 22,8256 | 22,6166 | 22,2376 |
| TDL | 29,1875 | 28,8662 | 28,7429 | 30,8291 | 30,2727 | 29,7230 | 22,0251 | 21,4800 | 20,1219 |
| CDS | 27,2533 | 27,5109 | 26,6515 | 30,5186 | 30,2362 | 29,7045 | 21,5003 | 21,2421 | 20,1284 |

The simplest way, how to subsampling a square block of pixel is using only odd (even) rows (columns), as a shown on the Fig. 2a. In this section we present another simple technique that called chess-subsampling. Fig. 2b shows a block of 8 x 8 pixels with chess subsampling technique. These two proposed methods are compared together with an adaptive subsampling technique. This technique is based on the evaluation of values in a different block of pixels after simply interframe prediction and 32 pixels with maximum values are selected. For each block of pixels in the frame they can situated in different positions.

Liu and Zaccarin [5] presented pixel-decimation technique by factor 4. Fig. 3a shows a block of 8 x 8 pixels with each pixel labelled **a**, **b**, **c** or **d**. They call pattern **A** the subsampling pattern that consist of all the **a** pixels. Similarly, patterns **B**, **C** and **D** are the subsampling patterns that consist of all the **b**, **c** and **d** pixels, respectively. If pattern **A** is used at the location (x,y), then it is also used at locations (x+2i,y+2j) for i,j integers within the search area (Fig. 3b). Pattern **B** is used at location (x+2i,y+1+2j), pattern **C** at (x+1+2i,y+2j), and pattern **D** at (x+1+2i,y+1+2j). For each of the subsampling pattern is obtained a motion vector that minimises the MAD over the locations where that pattern is used. Then is computed the match for each of four vectors, but using all pixels in the block. The one that has minimum MAD among the four is selected as the motion vector for the block.

On the other hand we present much simpler techniques by factor 4 that above, again. On the Fig. 4 we can see the graphic depiction of the orthogonal selection of pixels. This method outgoing from pixel-decimation technique described above, but used only pattern **A** for estimation the motion vector over the all search area. The second one is a nonorthogonal selection of pixels in the block. This approach is based on the assumption that better estimation of motion vector is achieved, when we used pixels from each rows and each column. Unlike the orthogonal pixel-decimation by factor 4, where are not used any pixels in even rows and even columns, in the nonorthogonal pixel-decimation are used some pixels from each rows and each columns. The process an adaptive pixel decimation by factor 4 is the same that adaptive pixel-decimation by factor 2, but only 16 pixels are selected for estimation the motion vector.

# 3. Experimental results

The presented block matching techniques with pixel decimation by factor 2 and 4 were tested on the two following frame by sequences in the format **raw** ( 256x256, 8 bpp ) Fig. 5a,b,c.

Degree of reduction interframe redundancy was evaluated by PSNR

$$PSNR = 10\log_{10}\left\{\frac{(255)^2}{\frac{1}{256*256}\sum_{i=1}^{256}\sum_{j=1}^{256}\left(X_{i,j}-X^c_{i,j}\right)^2}\right\} \quad [dB],$$

Where : $X_{i,j}$ are values of pixels into the current frame $X^c_{i,j}$ are values of pixels into the previous frame after motion compensation.

Performances of proposed pixel-decimation procedures is right to compare with the simple interframe prediction:

- $PSNR_{Talk}$ = 20.9775 dB,
- $PSNR_{Hus}$ = 27.1827 dB,
- $PSNR_{Walk}$ = 17.4803 dB.

In the Tab. 1a,b,c is the evaluation of proposed kinds of the pixel decimation with full search algorithm.

On the Tab. 1 we can see that the best results from all three types of sequences given pixel decimation with chess selection of matching pixels by factor 2 a the pixel decimation with nonorthogonal selection of matching pixels by factor 4, respectively. Therefore, these two methods of pixel decimation have been tested with some fast search algorithms (Tab. 2), such as three step search algorithm (TSS), two-dimensional logarithmic search algorithm (TDL)a conjugate direction search (CDS).

# 4. Conclusion

A number of fast algorithms for block motion estimation have been previously proposed. These algorithms reduce the number of computations by limiting the number of locations searched. They rely on a monotonically increasing MSE around the location of the optimal motion vector to iteratively determine that location. In a number of cases, though, the MSE surface has several local minima in which these algorithms can be trapped.

In proposed techniques we presented approach based on the reduction of the computation complexity in sense using only fraction of pixels in the block The main aim of this paper have been to present and experimentally tested of basic pixel decimation approaches in block motion estimation technique. We present some techniques to estimate a motion vector for each block of pixels by using only a fraction of the pixels in the block.

Tab. 1 shows the performance of pixel decimation techniques by factor 2 and 4. The performance of full search algorithm without any subsampling (full pixels) is also shown. It is seen that the proposed approach has PSNR very close to the full search using no subsampling. In addition, by using pixel decimation with chess selection of matching pixels by factor 2 a the pixel decimation with nonorthogonal selection of matching pixels by factor 4, respectively, we obtained the best performance from all proposed methods of pixel decimation. Therefore, these two methods of pixel decimation have been tested with some fast search algorithms.

# References

[1] H. G. Musmann , P. Pirsch, H.-J. Grallert : Advances in Picture Coding , Proceedings of the IEEE, Vol.73, No.4, April 1985, pp.523-548

[2] Koga, T., Iinuma, K., Hirano, A., Iijima, Y., Ishiguro, T. : Motion-Compensated Interframe Coding for Video Conferencing, Proc. Nat. Telecommun. Conf., 1981 New Orleans, LA, pp. G5.3.1-G5.3.5.

[3] R. Srinivasan, K.R.Rao : Predictive Coding Based on Efficient Motion Estimation, IEEE Transaction on Communications, Vol. COM-33, No. 8, August 1985, pp.888-896.

[4] J. R. Jain, A. K. Jain : Displacement measurement and its application in interframe image coding, IEEE Trans. Commun., Vol. COM-29, No. 12, December 1981, pp. 1799-1808.

[5] B. Liu, A. Zaccarin : New Fast algorithms for the Estimation of Block Motion Vectors, IEEE Transactions on Circuits and System for Video Technology, Vol. 3, No. 2, April 1993, pp. 148-157.

[6] Radoczi, P., Levický, D. : Comparison function of distortion for block matching algorithms, Zborník Nové smery v spracovaní signálov V, Liptovský Mikuláš 2000, s. 147-152.

# About authors ...

**Peter RADOCZI** was born in 1976 in Košice, Slovakia. He received Ing. (MSc.) degree at the Faculty of Electrical Engineering and Informatics of Technical University of Košice in 1999. In this time he is PhD. student at the Department of Electronics and Multimedia Communications.

**Dušan LEVICKÝ** ( Prof., Ing., CSc. ) was born in Slanec, Slovak Republic, in 1948. He received the Ing. ( MSc. ) degree and CSc. ( PhD. ) degree from the Technical University in Košice and he is now professor at the Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics in Košice. His research interest includes digital image processing, multimedia communications and cryptography.