EVALUATION OF FAST ALGORITHMS FOR MOTION ESTIMATION

Ionuţ Pirnog, Cristian Anghel, Andrei Alexandru Enescu, and Constantin Paleologu

Telecommunications Department, University Politehnica of Bucharest, Romania
{ionut, canghel, aenescu, pale}@comm.pub.ro

ABSTRACT

In this paper we present an evaluation of the fast algorithms used for motion estimation and compensation. The presented algorithms are classified in two categories. The first category contains the algorithms with fixed number of iterations, i.e., Three Step Search (TSS), New Three Step Search (NTSS), and Four Step Search (FSS). The second category includes motion estimation algorithms with variable number of iterations, i.e., Orthogonal Search (OS), Two Dimensional Logarithmic Search (TDLS), and Adaptive Rood Pattern Search (ARPS). It is proved that for the second category of algorithms the number of iterations depends on the dimension of the search window. The evaluation is done by comparing the peak signal-to-noise ratio (PSNR) of the compensated motion frame and the number of blocks that are used.

Keywords – Motion estimation, fast algorithms, fixed and variable iterations**.**

I. INTRODUCTION

Multimedia Information Retrieval (MIR) emerged as a branch of the Information Retrieval Domain simultaneously with the increasing interest in multimedia content analysis, characterization, and retrieval. Since by multimedia content we understand audio data, video data, textual data (and combinations of these), the Information Retrieval is defined for textual data; thus we can easily conclude that MIR means audio and video content analysis.

Motions estimation played a key role in video compression [1] and is more successfully used since the development of the Moving Pictures Experts Group (MPEG) standards. To enhance the "access to" and "retrieval of" multimedia content, MPEG has developed a standard called MPEG-7 that provides a rich set of standardized tools to describe multimedia content [2]. This set of tools includes motion descriptors that can be used for classification, indexing, comparison, and retrieval of video content [2]. For the video content compressed using the MPEG-4 standard, the motion information is contained in the form of motion vectors for the B and P frames and can be used directly for the extraction of the motion descriptors. If the video content does not include motion information, then the motion vectors can be extracted using one of the many fast motions estimation algorithms developed in the last decade.

The basic idea behind motion estimation is the block by block comparison of two consecutive frames, i.e., the current frame and the previous frame. The blocks are rectangular areas of a frame, having the dimensions chosen according to the application. Usually, the blocks are 8×8 or 16×16 pixels for compression; when high precision is needed the blocks can contain only one pixel. In this case, the motion vector has a dimension equal to the number of pixels, and is referred as "optical flow."

Fast motion estimation algorithms are used in order to minimize the computational complexity with little loss in the precision of the estimation. This is the case of video compression because the predicted frames P and B are based on a full frame I, the motion information, and the difference between the current frame and the motion compensated frame. These algorithms are simple but very efficient, so that they are extensively used in video compression.

The rest of the paper is organized as follows. In Section II we describe six of the most used motion estimation algorithms divided in two categories, i.e., fixed number of iterations and variable number of iterations. Section III contains the comparative experimental results obtained through simulations, using three different scenarios. In Section IV, we present the conclusions of the evaluation and future work.

II. BLOCK MATCHING ALGORITHMS

As stated in the first section, motion is a very important characteristic of video content and can be used for compression (e.g., MPEG-4) and retrieval (e.g., MPEG-7). Combining with image segmentation we can obtain camera motion information (using the motion vectors of the background) and object motion information (using the motion vectors of region labeled as objects).

The block matching algorithms used for motion estimation split the current frame into non overlapping blocks of size 8×8 or 16×16 pixels and, for every block, the corresponding block in the previous frame is found [3]. For a better understanding of the basic method for motion estimation, let us denote the current frame with $F_c(x,y)$, the previous frame with $F_p(x,y)$, and with $B^c_{m,j}(x,y)$ the block number j of the current frame. The parameter m is the dimension of the block and the pair (x,y) is the horizontal and vertical position of the block in the frame. In the following, in order to simplify the notation, we will omit the subscript m. For every block $B^c_j(x,y)$ of the current frame, we can define a search window $W^p_{l,j}(x,y)$ as an extension of the block in the previous frame $B^p_k(x,y)$, with the same position as the bloc in the current frame. The parameter l of the search window represents the dimension of the extension in all four directions. This means that the search window

 ISBN: 978-1-61208-123-6

$W_{l,j}^{p}(x,y)$ used for the motion estimation of a block $B_j^c(x,y)$ will have the dimension $(2l+m)\times(2l+m)$.

The corresponding block represents the best block obtained by comparing the block from the current frame, B_j^c, with all the overlapping blocks from the search window, $B_k^p, k=\overline{1,N}$, where N represents the number of blocks in the search window.

The corresponding block or the best block is found using the minimization of a cost function defined as the mean square error (MSE) or as the mean absolute error (MAE) between the current block in the current frame and the current block in the search window, i.e.,

$$MSE\left(B_j^c, B_k^p\right) = \left(B_j^c - B_k^p\right)^2 = \\ = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(P_c(i,j) - P_p(i,j)\right)^2 \quad (1)$$

$$MAE\left(B_j^c, B_k^p\right) = \left|B_j^c - B_k^p\right| = \\ = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left|P_c(i,j) - P_p(i,j)\right| \quad (2)$$

where $P_c(i,j)$ and $P_p(i,j)$ represents the pixels in position (i,j) from the current and previous frames. The values of the parameters m and l determines the precision and the computational complexity of the motion estimation. If the motion of an object in a video is wide, then, for a good estimation, it is necessary a large search window, but this means that the process will be computationally expensive. This algorithm is known as the Full Search (FS) algorithm because it searches the best block from all the blocks in the search window [4]. The precision of this algorithm is very good, if the window is large enough to include the amplitude of the motion, but because it uses all the blocks in the search window it is also computationally expensive.

The class of fast block matching algorithms for motion estimation was developed with the goal of lowering the computational time without causing a high loss in precision. This is done by using for comparison only a small number of blocks in the search window. The developed fast algorithms use the same principle but with a different block selection scheme.

A. Three Step Search Algorithm

The Three Step Search (TSS) is the first fast algorithm that was developed and, as its name suggests, it uses three steps to determine the best block. The three steps are as follows [4]:

1. The current block, $B_j^c(x,y)$, is compared to the centre block in the search window, $B_j^p(x,y)$, and 8 blocks located at a distance S from the centre block. The cost function for all the blocks is computed and the best block for this step is determined as the block with the minimum cost.
2. The determined best block is selected as the new centre, the distance S is halved, and step 1 is repeated.
3. When the distance is 1, the best block is determined as the block with the minimum cost.

The values of the motion vector for the current block are obtained as de difference between the horizontal and vertical positions of the best block and the positions of the current block. Usually, the initial distance is set to $S=4$ and the search window parameter is set to $l=7$. These values where experimentally determined but lead to good enough results for estimating motion with low amplitude. Larger values of these parameters may lead, in some cases, to a better estimation, but definitely lead to an increase of the computational cost. Compared to the FS algorithm, the TSS algorithm has a nine times lower computational cost.

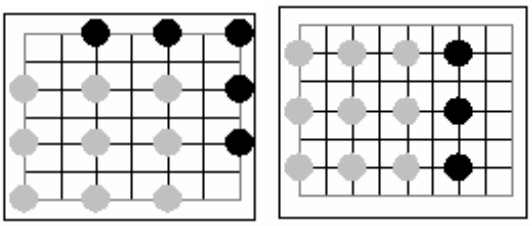

Figure 1. The neighboring blocks selected if the best block is one of the blocks with $S=1$.

B. New Three Search Algorithm

The improvement introduced by the New Three Step Search (NTSS) is a better estimation of the motion with low amplitude [5]. This is done by evaluating in the first step another 8 blocks located at a distance $S=1$ from the centre block. The best block from these initial 17 blocks is determined based on the cost functions; depending on the positions of the best block we have three situations:

1. If the best block is the one in the centre of the search window, then the algorithm stops.
2. If the best block is one of the blocks located at a distance $S=4$, then the TSS algorithm is used.
3. If the best block is one of the blocks located at a distance $S=1$, then its neighbors are compared with the current block and the best block is determined as the block with the minimum cost function.

To decrease the number of blocks compared and to eliminate the re-evaluation of some blocks, the neighbors selected in the last case depends on the position of the best block as shown in Figure 1.

C. Four Step Search Algorithm

The Four Step Search (FSS) is performed in four steps [6]:

1. The centre block and the eight block al distance $S=2$ are evaluated. If the block with the minimum cost is the centre block, then the algorithm jumps to step 4. Else it goes to step 2.
2. The block with the minimum cost is selected as the centre block and the neighboring blocks at distance

 ISBN: 978-1-61208-123-6

$S = 2$ are evaluated. The selection of the neighboring blocks is similar to NTSS.

3. If the best block is the centre block, then the algorithm goes to step 4, else step 2 is repeated.
4. The distance is set to $S = 1$ and the centre block and its eight neighbors are evaluated. The best block is the block with the minimum cost.

The minimum number of evaluated blocks is the same as in the case of the NTSS algorithm, but the maximum number decreases.

D. Two Dimensional Logarithmic Search Algorithm

The Two Dimensional Logarithmic Search (TDLS) algorithm is similar to the TSS algorithm, but verifies the centre block and only 4 blocks located at distance S on the horizontal and vertical axes [6]. The value of S is not fixed, as in the case of TSS and FSS algorithms, and can be chosen depending of the dimension of the search window. The steps required by this algorithm are:

1. The value of the distance parameter S is set. The centre block and the 4 blocks at distance S are evaluated, and the block with the lowest cost is selected.
2. If the selected block is the centre block, then the distance S is halved. If one of the other blocks is selected, then this block is set as the new centre and step 1 is repeated.
3. When the distance becomes equal to one, the centre block and all its neighbors are evaluated. The block with the lowest cost is the best block.

It is not always clear that the TDLS algorithm obtains better results than the other presented algorithms. But the fact that the initial value of S is not imposed can be very helpful in case of a motion with large amplitude.

E. Orthogonal Search Algorithm

The Orthogonal Search (OS) is a combination of the TSS and TDLS algorithms [3]. The algorithm involves the following steps:

1. The initial distance is chosen as half of de maximum distance of the search window. The centre block and two blocks on the horizontal axis are evaluated. The block with the minimum cost is set as the new centre.
2. The centre block and two blocks on the vertical axis are evaluated and the new centre is selected.
3. If the distance parameter S is bigger than one, then the distance is halved and the steps 1 and 2 are repeated. Else, the last centre block is the best block.

The computational cost for the OS algorithm is smaller as compared to the TDLS algorithm, but the precision decreases.

F. Adaptive Rood Pattern Search Algorithm

The Adaptive Rood Pattern Search (ARPS) algorithm uses the motion information of the neighboring block in the left. This is helpful if the current block and its neighbor on the left belong to the same object in the frame; in this case, their motion is similar [6]. The steps of the ARPS algorithm are:

Table 1. PSNR for video "Motion" with $m = 16$ and $l = 7$.

Algorithm	1-2	2-3	3-4	4-5
FS	34,06	31,93	30,28	29,27
TSS	33,98	31,81	30,15	29,25
NTSS	33,94	31,8	30,15	29,25
FSS	31,73	30,2	28,93	28,05
OS	33,86	31,44	29,56	28,41
TDLS	32,77	30,81	29,46	28,57
ARPS	33,67	31,72	30,01	29,07
5-6	**6-7**	**7-8**	**8-9**	**9-10**
30,43	29,47	29,92	27,72	27,89
30,27	29,33	29,75	27,55	27,81
30,21	29,33	29,79	27,6	27,83
28,88	28,07	28,78	27,05	27,04
29,26	27,66	28,51	25,82	27,04
29,43	28,44	28,99	27,14	27,18
30,01	29,1	29,47	27,48	27,6

1. The centre block, the block indicated by the motion vector of the neighbor, and four blocks are evaluated. The four blocks are selected on the horizontal and vertical axes at a distance S, chosen as the maximum value between the absolute values of the motion vector.
2. The block with the minimum cost is selected as the new centre block, the distance is set to 1, and the centre block together with its four axis neighbors are evaluated.
3. If the block with the minimum cost is in the centre, then the algorithm stops; consequently, this is the best block. Else, step 2 is repeated.

For the blocks in the first column, there are no left neighbors, so that the distance S is set to 2.

The major advantage of this algorithm is that after the first step the search is moved to the area where the best block is, without going through intermediary steps. Consequently, the computational cost is smaller than all the other algorithms.

III. EXPERIMENTAL RESULTS

The goal of this evaluation was to determine the performances of the fast block motion estimation algorithms, the parameters that determine the efficiency of the motion estimation, and if these algorithms can be improved.

In order to evaluate the performances, we have implemented the algorithms in Matlab using two videos for estimation of the motion. The first observation we have made based on our simulations is that the classification of the algorithms based on the efficiency and computational cost does not depend on the video selected.

We have stated in the first section that the results are presented for three scenarios. These scenarios were obtained by varying the values of the search window parameter and the block size parameter.

1. First scenario

In the first scenario, we set the blocks to 16×16 pixels and the search window parameter to 7. The results for the video "Motion" with 10 frames are shown in Table 1 and Figure 2.

 ISBN: 978-1-61208-123-6

Table 2. Number of blocks verified.

Algorithm	Nb
FS	255
TSS	25
NTSS	19
FSS	18
OS	13
TDLS	18
ARPS	11

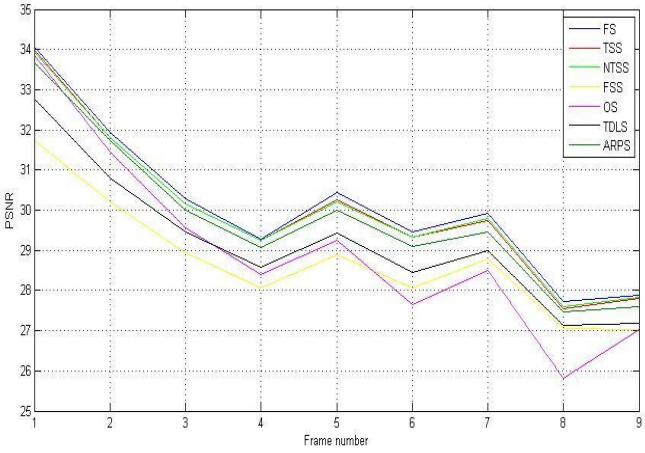

Figure 2. Motion estimation results for video "Motion" with $m = 16$ and $l = 7$.

The results in Table 1 show that the variation of the peak signal-to-noise ratio (PSNR) is not very high. The difference between these fast block matching algorithms is results in terms of the computational cost. This cost is evaluated through the number of blocks (Nb) in the verified search window for every block in the current frame. The values of the number of verified blocks are show in Table 2.

2. Second scenario

In the second scenario we set the blocks to 8×8 pixels and the search window parameter to 7. The results for the same video, "Motion", are shown in Table 3 and Figure 3.

As it can be seen, the PSNR slightly increases if we use 8×8 blocks but the increase in the computational cost is more significant. So, if the precision of estimation is not imposed it is recommended to use 16×16 blocks to obtain a smaller computational cost.

3. Third scenario

In the third scenario we set the blocks to 16×16 pixels and the search window parameter to 14 and 30. This scenario is based on the results from the first two scenarios and all of our simulations. We observed that even if we decrease the dimensions of the block, the estimation is not perfect. This happens because the motion amplitude in the videos we used is bigger than the search window. In Table 4 are presented the comparative results of the PSNR for these two values of the search window parameter and the results from the second scenario. The results are shown for frames 4 and 5 of the video "Motion."

Table 3. PSNR for video "Motion" with $m = 8$ and $l = 7$.

Algorithm	1-2	2-3	3-4	4-5
FS	34,61	32,53	31,28	30,78
TSS	34,19	32,24	30,96	30,49
NTSS	34,16	32,23	30,94	30,49
FSS	32,03	30,44	29,65	29,11
OS	34,09	31,68	29,91	28,88
TDLS	33,11	31,27	30,16	29,3
ARPS	33,27	31,76	30,46	30,02
5-6	**6-7**	**7-8**	**8-9**	**9-10**
31,91	31,21	31,94	28,71	29,06
31,57	30,85	31,54	28,46	28,9
31,54	30,92	31,69	28,56	28,97
29,83	29,21	29,87	27,82	27,82
29,97	28,78	30,03	26,53	27,76
30,29	29,83	30,49	27,9	28,09
30,75	30	30,97	28,29	28,59

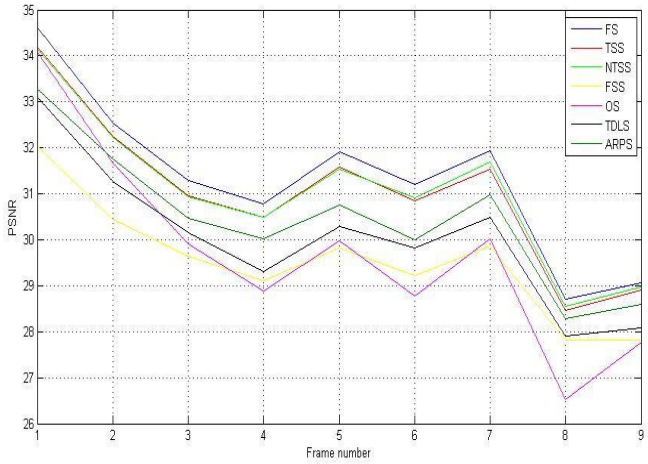

Figure 3.Motion estimation results for video "Motion" with $m = 8$ and $l = 7$.

Table 4. Comparative results of the PSNR for different search window parameters.

Algorithm	$l = 7$	$l = 14$	$l = 30$
TSS	29,25	29,25	29,25
NTSS	29,25	29,25	29,25
FSS	28,05	28,05	28,05
OS	28,41	28,41	28,41
TDLS	28,57	31,3	31,89
ARPS	29,07	30,81	31,04

The results show that the fast algorithms with fixed value of the search parameter S have the same precision even if we increase the dimensions of the search window. This was a predictable result because the search parameter of these does not depend on the search window dimension, so that the number of steps will be constant.

Significant improvement of the PSNR appears for the TDLS and ARPS algorithms, i.e., two of the algorithms that have variable search parameter and variable number of steps.

 ISBN: 978-1-61208-123-6

Table 5. Comparative results of the number of blocks verified for different search window parameters.

Algorithm	$l = 7$	$l = 14$	$l = 30$
TSS	25	25	25
NTSS	25	25	25
FSS	18	18	18
OS	13	13	13
TDLS	18	19	20
ARPS	12	14	14

As we specified in Section II, the value of the search parameter of the TDLS algorithm is not fixed and the number of steps depends on the search window. The comparative results for different dimensions of the search window are obtained for the same value of the search parameter $S = 4$.

We can see that the value of the PSNR for both the ARPS and TDLS algorithms in the case of $l = 30$ (Table 5) is higher than the PSNR for the same frames, 4 and 5, of the FS algorithm in the case of $l = 7$.

The computational complexity of the algorithms is evaluated in terms of the mean number of blocks verified for every frame. Since the evaluation of the algorithms was done using Matlab it is difficult to express the computational complexity in terms of the number of arithmetical operations. Concerning the number of blocks verified, the FS algorithm has a number of 255 blocks in the search window for every block in the current frame, and the ARPS and TDLS algorithms have blocks verified, even if we increase the search window.

IV. CONCLUSIONS AND FUTURE WORK

By evaluating a representative number of fast block matching algorithms for motion estimation, the following conclusions can be outlined.

1. There are two classes of fast algorithms for motion estimation: i) fixed search parameter and constant number of steps, and ii) variable search parameter and variable number of steps.
2. Although the differences between algorithms are not very high (in terms of both PSNR and Nb), in the case of the same search parameters and search window size, there is no room for improvement for the algorithms in the first class.
3. The dimensions of the search window determine the precision of motion estimation for the algorithms in the second class; as it was shown in Section III, the improvements brought by increasing the window size lead to a value of the PSNR higher that the value obtained for the FS algorithm.

In future work we intend to combine the two classes, meaning that we will see how the algorithms in the first class behave for different search parameters and window sizes. Another track we intend to follow is lowering the number of block verified, for a large search window, through different search schemes.

ACKNOWLEDGEMENT

This work was supported by the UEFISCSU Romania under Grant PN-II-RU-TE no. 7/05.08.2010.

V. REFERENCES

[1] Z. Chen, "Efficient block matching algorithm for motion estimation," *International Journal of Signal Processing*, vol. 5, no. 2, pp. 133–137, 2009.

[2] ISO/MPEG N4358, "Text of ISO/IEC Final Draft International Standard 15938-3 Information Technology - Multimedia Content Description Interface - Part 3 Visual," MPEG Video Group, Sydney, July 2001.

[3] A. Barjatya, "Block matching algorithms for motion estimation," Final Project Paper 2004.

[4] Y. C. Lin and S. C. Tai, "Fast full-search block-matching algorithm for motion-compensated video compression," *IEEE Trans. Communications*, vol. 45, no. 5, pp. 527–531, May 1997.

[5] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[6] S. Jamkar, S. Belhe, S. Dravid, and M. S. Sutaone, "A comparison of block-matching search algorithms in motion estimation," in *Proc. 15th International Conference on Computer Communication*, pp. 730–739, Mumbai, India, 2002.

 ISBN: 978-1-61208-123-6