A Complexity and Quality Evaluation of Block Based Motion Estimation Algorithms

S. Usama, M. Montaser, O. Ahmed

Motion estimation is a method, by which temporal redundancies are reduced, which is an important aspect of video compression algorithms. In this paper we present a comparison among some of the well-known block based motion estimation algorithms. A performance evaluation of these algorithms is proposed to decide the best algorithm from the point of view of complexity and quality for noise-free video sequences and also for noisy video sequences.

Keywords: motion estimation, block matching, quality evaluation, complexity evaluation.

1 Introduction

Interframe predictive coding is used to eliminate the large amount of temporal and spatial redundancy that exists in video sequences, and helps in compressing them. In conventional predictive coding the difference between the current frame and the predicted frame (based on the previous frame) is coded and transmitted.

The better the prediction, the smaller the error and hence the transmission bit rate. If a scene is still, then a good prediction for a particular pel in the current frame is the same pel as in the previous frame, and the error is zero. However, when there is motion in a sequence, then a pel on the same part of the moving object is a better prediction for the current pel. Use of knowledge about the displacement of an object in successive frames is called Motion Compensation. There are a large number of motion compensation algorithms for interframe predictive coding. In this study, however, we have focused on a single class of such algorithms, called Block Matching Algorithms. These algorithms estimate the amount of motion on a block-by-block basis, i.e. for each block in the current frame a block from the previous frame is found that is said to match this block, based on a certain criterion. There are a number of criteria to evaluate the "goodness" of the match, some of which are:

1. Pixel Difference Classification (PDC),
2. Mean Absolute Difference (MAD),
3. Mean Squared Difference (MSD).

Mean absolute difference (MAD) is the most commonly used cost function, since it does not need a multiplication operation. PDC counts the number of matching pixels between two blocks.

Mathematically these cost functions can be defined as:

$$\mathrm{MAD}(i,j)=\frac{1}{NM}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left|C(x+k,\ y+l)-R(x+i+k,\ y+j+l)\right|$$

$$\mathrm{MSD}(i,j)=\frac{1}{NM}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left[C(x+k,\ y+l)-R(x+i+k,\ y+j+l)\right]^2$$

$$\mathrm{PDC}(i,j)=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}T_{i,j}(k,l),$$

Where, for given threshold t,

$$T_{i,j}(k,l)=\begin{cases}1 & \text{if } \left|C(x+k,\ y+l)-R(x+i+k,\ y+j+l)\right|\le t\\ 0 & \text{otherwise,}\end{cases}$$

where, $R(x+i+k,\ y+j+l)$ and $C(x+k,\ y+l)$ are the reference frame's block and the current frame's block respectively and, the motion vector is defined by (i, j).

2 Search algorithms

In this section some famous algorithms are introduced, including FS, TSS, 4SS, OSA, OTS, CBOSA, CSA, MRVBS, and multi-resolution algorithms.

2.1 Full search algorithm (FS)

The simplest method to find the motion vector for each macro-block is to compute the certain cost function at each location in the search space. This is referred to as the full search algorithm. The cost function used in the full search algorithm is the mean absolute difference MAD. The best matching block is the reference block for which $\mathrm{MAD}(i, j)$ is minimized, thus the coordinates (i, j) define the motion vector. The main problem of the full search algorithm is the computation complexity, which can be estimated as follows [1]. For each motion vector there are $(2p+1)^2$ search locations. At each location (i, j) $N\times M$ pixels are computed. Each pixel comparison requires four operations, namely, a subtraction, an absolute-value calculation, one addition, and one division, if the cost of accessing pixels $C(x+k,\ y+l)$ and $R(x+i+k,\ y+j+l)$ is ignored. Thus the total complexity per macro-block is $(2p+1)^2\times NM\times 4$ operations. Then for frame resolution $I\times J$ and frame rate F frames per second, the overall complexity is defined as:

$$\begin{aligned}\text{complexity} &= \frac{IJF}{NM}(2p+1)^2\times MN\times 4\\ &= IJF(2p+1)^2\times 4 \quad \text{operation / sec.}\end{aligned}$$

For example for typical values for broadcast TV with $N=M=16$, $I=720$, $J=480$ and $F=30$ the motion estimation based on the full search algorithm requires 39.85 GOPS (Giga operations per second) for $p=15$, and 9.32 GOPS for $p=7$. This example shows that the full search algorithm is computationally expensive but guarantees finding the minimum MAE

value. Due to the high computational complexity of the full search, alternative search methods are desirable.

2.2 Three step search algorithm (TSS)

This algorithm [2] is simple and robust and also provides near optimal performance, so it has become very popular. It searches for the best motion vectors in a coarse to fine search pattern. It can compute displacement up to ±7 pixels. The algorithm may be described as follows:

Step 1: An initial step size is chosen. Eight blocks at a distance of step size from the center (around the center block) are picked for comparison.

Step 2: The step size is halved. The center is moved to the point with the minimum distortion.

Steps 1 and 2 are repeated until the step size is equal to 1. One problem that occurs with the Three Step Search is that it uses a uniformly allocated checking point pattern in the first step, which becomes inefficient for small motion estimation. For each motion vector there are (8×3+1) search locations. At each location (i, j) $N\times M$ pixels are computed. Each pixel comparison also requires three operations, a subtraction, an absolute-value calculation, and one addition, if the cost of accessing pixels $C(x+k, y+l)$ and $R(x+k+i, y+l+j)$ is ignored. Thus the total complexity per macro-block is $(8\times3+1)\times NM\times4$ operations. Then for frame resolution $I\times J$ and frame rate F frames per second, the overall complexity is defined as:

$$\text{Complexity} = \frac{IJF}{NM}25\times MNx\times 4 = IJF\times 100\,\text{operation/sec.}$$

For example if $I=720$ and $J=480$ and $F=30$ then the overall complexity is equal to 1036.8 MOPS, and this is just 2.6 % of the full search required operations.

2.3 Four step search (4SS)

This algorithm [3] is based on the real world image sequence's characteristic of center-biased motion. The algorithm starts with a nine-point comparison and then the other points for comparison are selected on the basis of the following algorithm:

Step 1: Start with a step size of 2. Pick nine points around the search window center. Calculate the distortion and find the point with the smallest distortion. If this point is found to be the center of the searching area, go to *step 4*, otherwise go to *step 2*.

Step 2: Move the center to the point with the smallest distortion. The step size is maintained at 2. The search pattern, however, depends on the position of the previous minimum distortion.
a) If the previous minimum point is located at the corner of the previous search area, five points are picked.
b) If the previous minimum distortion point is located at the middle of the horizontal or vertical axis of the previous search window, three additional checking points are picked. Locate the point with the minimum distortion. If this is at the center, go to *step 4*, otherwise go to *step 3*.

Step 3: The search pattern strategy is the same, however it will finally go to *step 4*.

Step 4: The step size is reduced to 1 and all nine points around the center of the search are examined. The computational complexity of the four step search is less than that of the three step search, while the performance in terms of quality is as good.

2.4 Multi-resolution algorithms

Spatial multiresolution video sequences provide video at multiple frame sizes, allowing extraction of only the resolution or bit rate required by the user. To illustrate the efficiency of multi-resolution based algorithms [1] in comparison with full-frame based algorithms, assume that the current frame and the reference frame are decomposed into two levels using a simple averaging filter (2×2) twice. Using the FS algorithm in the lowest resolution (level 2), the complexity of the algorithm is as follows.

Level 2:

assume the parameters for broadcast TV (720×480 at 30 frames per second). Then the picture size in level 2 is 180×120, macroblock size 4×4, and the number of macroblocks is equal to (180×120)/(4×4), equal to 1,350 at 30 frames/second. The searching window will be rescaled to $\left[\left\lceil -\frac{p}{4}\right\rceil, \left\lceil \frac{p}{4}\right\rceil\right]$. If p = 15 then the searching window in level 2 is [4, 4], thus the number of search locations is $(2\times4+1)^2 = 81$. The complexity for level 2 = 30×180×120×81×4 = 209,952 MOPS.

Level 1:

in this level the picture size = 360×240, and the macroblock size = 8×8. The number of macroblocks = 1,350 at 30 frames/second. In this level there is just a search for the best matching within the center resulting from level 2 and its eight neighbors, so the searching window is [−1, 1] and the number of searching locations = 9. The complexity for level 1 = 30×240×360×9×3 = 69.984 MOPS.

Level 0:

picture size = 720×490, and macroblock size =16×16. The number of macroblocks = 1,350 at 30 frames/second. In this level also there is just a search for the best matching within the center resulting from level 1 and its eight neighbors, so the searching window is [-1,1] and the number of searching locations = 9. The complexity for level 0 = 30×480×720×9×4 = 279,936 MOPS. Then the total complexity of this algorithm = 373.248+93.312+209.952 = 676.512 MOPS. This is a significant reduction over 29.98 GOPS that is needed for the FS algorithm.

From the complexity point of view the multi-resolution search algorithm is very efficient; however, such a method requires increased storage due to the need to keep pictures at different resolutions. Also, because the search starts at the lowest resolution small objects may be completely eliminated and thus fail to be tracked. On the other hand the creation of low-resolution pictures provides some immunity to noise.

2.5 Wavelet based algorithms [5]

An efficient multi-resolution tool is the wavelet transform, so we review a robust algorithms based on the wavelet trans-

formation, MRVBS (Multi-resolution Variable Block Size) algorithm. It is based on a central search process in three layers, namely, layer 2, layer 1, and layer 0 (the original frame). MAD is used as a cost function. The main steps are described as follows: first the current frame and the previous frame are decomposed into two layers of the wavelet domain.

Step 1: in layer 2, the central search process is applied on the low band i.e., searching for the best matching within the nine neighboring blocks to get an initial motion vector. The block size used in this step is 4×4, and the estimated motion vector is used as the new center for the central search process for the details.

Step 2: the estimated motion vector in the previous step is rescaled and used as the new center for the three highest bands in layer 1 with block size 8×8.

Step 3: from the estimated motion vectors in step 2, the median values are chosen to be rescaled into layer 0 and then used as a new center to estimate the final motion vector by using block size 16×16.

The computational cost of this algorithm without the wavelet complexity is $(36*p_2+27*p_1+9*p_0)$, where p_2, p_1, and p_0 are the block size in layer 2, layer 1, and layer 0, respectively.

2.6 Two dimensional logarithmic search (TDL)

This algorithm was introduced by Jain & Jain [6]. Although this algorithm requires more steps than the Three Step Search, it can be more accurate, especially when the search window is large. The algorithm can be described as follows:

Step 1: Choose an initial step size as 2^J. Look at the block at the center of the search and the four blocks at a distance s from this on the *X* and *Y*-axes. (The five positions form a + sign).

Step 2: If the position of the best match is at the center, halve the step size. If, however, one of the other four points is the best match, then it becomes a new center, and step 1 is repeated.

Step 3: When the step size becomes 1, all the nine blocks around the center are chosen for the search and the best among them is picked as the required block. Many variations of this algorithm exist, and they differ mainly in the way in which the step size is changed. Some people argue that the step size should be halved at every stage. Some people believe that the step size should also be halved if an edge of the search space is reached.

2.7 Orthogonal search algorithm (OSA)

This algorithm was introduced by Puri [7] and it is a hybrid of the Three Step Search and the Two Dimensional Logarithmic Search. It has a vertical stage followed by a horizontal stage for the search for the optimal block. Then the algorithm may be described as follows:

Step 1: Pick a step size (usually half the maximum displacement in the search window). Take two points at a distance of the step size in the horizontal direction from the center of the search window and locate (among these) the point of minimum distortion. Move the center to this point.

Step 2: Take two points at a distance of the step size from the center in the vertical direction and find the point with the minimum distortion.

Step 3: If it is greater than one; halve the step size and repeat steps 1 and 2, otherwise, halt.

2.8 Center-biased orthogonal search algorithm (CBOSA)

The CBOSA algorithm [8] for finding small motion is described below. The CBOSA algorithm is a modification of the orthogonal search algorithm (OSA), which is reviewed in section (2.7). The OSA algorithm has faster convergence, fewer checking points and fewer searching steps. However, the performance of OSA in terms of MSE is much lower than that of 3SS and other fast BMAs. This is because the OSA algorithm does not make use of the center-biased motion vector distribution characteristics of the real world video sequence. In order to tackle this drawback, the CBOSA algorithm uses a smaller step size in the first step so as to increase the probability of catching the global minimum point.

For the maximum motion displacement of ±7 in both the horizontal and vertical directions, the CBOSA algorithm uses three horizontal checking points with a step size of 2 in the first step (*Step 1–H*). If the minimum BDM (block distortion measure) is at the center, it jumps to the vertical step (*Step 1-V*). Otherwise, one more checking point is searched in the horizontal direction).

This extra step is to make sure that the algorithm can cover the whole search window even using a small step size of 2 in the first step. Using the minimum BDM point found in *Step 1–H*, *Step 1–V* uses the same searching strategy as *Step 1–H* to search in the vertical direction. Then the algorithm jumps to *Step 2–H* and *Step 2–V*, respectively. These two steps use three checking points also with a step size of 2 in the horizontal and vertical directions, respectively.

After *Step 2–V*, the algorithm jumps to *Step 3–H* and *Step 3–V*, respectively. *Step 3–H* and *Step 3–V* use three checking points with the step size reduced to 1 in the horizontal and vertical directions, respectively.

Thus, the number of checking points required by the CBOSA algorithm varies from $(3+2+2+2)=9$ to $(3+1+2+1+2+2+2+2)=15$. The worse case computational requirement is just 2 checking points more than that of OSA, but it is 10 checking points fewer than for 3SS.

2.9 One at a time algorithm (OTS) [9]

This is a simple, but effective way of trying to find a point with the optimal block. During the horizontal stage, the point on the horizontal direction with the minimum distortion is found. Then, starting with this point, the minimum distortion in the vertical direction is found. The algorithm may be described as follows:

Step 1: Pick three points about the center of the search window (horizontal).

Step 2: If the smallest distortion is for the center point, start the vertical stage, otherwise look at the next point in the horizontal direction closer to the point with

the smallest distortion (from the previous stage). Continue looking in that direction till you find the point with the smallest distortion. (Going in the same direction, the point next to it must have a larger distortion).

Step 3: Repeat the above, but taking points in the vertical direction about the point that have the smallest distortion in the horizontal direction.

This search algorithm requires very little time; however the quality of the match is not very good.

2.10 Cross search algorithm (CSA)

This algorithm was introduced by M. Ghanbari [10]. The basic idea in this algorithm is still a logarithmic step search. However, the main difference between this and the logarithmic search method presented above is that the search locations picked are the end points of an "*x*" rather than a "+". The algorithm may be described as follows:

Step 1: The center block is compared with the current block and if the distortion is less than a certain threshold, the algorithm stops.

Step 2: Pick the first set of points in the shape of an "*x*" around the center. (The step size picked is usually half the maximum displacement). Move the center to the point of minimum distortion.

Step 3: If the step size is greater than 1, halve it and repeat step 2, otherwise go to step 4.

Step 4: If in the final stage the point of minimum distortion is the bottom left or the top right point, then evaluate the distortion at 4 more points around it with a search area of a "+". If, however, the point of minimum distortion is the top left or bottom right point, evaluate the distortion at 4 more points around it in the shape of an "*x*".

3 Performance comparison of the motion estimation algorithms

In this section we will introduce a comparison between some of the most efficient algorithms from different points of view. The well known FS, 4SS, 3SS, OSA, CBOSA, OTS, and CSA algorithms compared from the PSNR point of view as well as the are complexity point of view. The comparison is performed for noise-free sequences as well as noisy sequences with different SNR.

3.1 Complexity point of view

FS algorithm: the FS algorithm searches for the best matching within a large window $[-p:p]\times[-p:p]$. This means that it searches for the best matching within a $(2p+1)2$ block. Thus for the simplest cost function MAE three operations are performed, namely, one addition, one absolute computation, and one for subtraction, then the total operation number for just one block matching is equal to $4NM(2p+1)^2$, where N and M are the block size, and the total operation number per frame is $4IJ(2p+1)^2$ where I, and J are the frame size. This is too many operations, and requires very high speed processors.

TSS algorithm: the TSS algorithm searches for the best matching within $[-p:p]\times[-p:p]$ window. Here p is equal to 7, but only blocks in this window with a certain step are checked. The total number of checked blocks is 25. This means that the total operation per frame is 75 (*IJ*), so it requires just 2.6 % of the operations required for the FS algorithm (with $p = 15$). Note that data access is not taken into consideration.

4SS algorithm: in 4SS certain conditions are inserted for jumping between steps to overcome computation overlap. The total number of checked blocks varies between the maximum value (27) and the minimum value (17). On an average it requires 22 blocks to be checked. This means that the total operation per frame is 66 (*IJ*), and it requires just 2.289 % of the operations required for FS the algorithm.

TDL algorithm: in this algorithm for a maximum displacement of ±7 it requires checking points varying from $(5+8) = 13$ to $(5+4+8) = 17$ checking points.

OSA algorithm: for maximum displacement of ±7 the OSA algorithm requires $(3+2+2+2+2+2) = 13$ checking points.

CBOSA algorithm: the number of checking points required by the CBOSA algorithm varies from $(3+2+2+2) = 9$ to $(3+1+2+1+2+2+2+2) = 15$. It is very fast compared with the FSS, FS, or TSS algorithms, but it requires 2 more checking points than for the OSA algorithm.

CSA algorithm: for a maximum displacement of ±7 the CSA algorithm requires $(5+4+4) = 13$ checking points it can be formulated in a general form as $5+4*\log_2 W$, where W is the initial step size. For example, it is chosen to equal 4 for a maximum displacement of ±7.

OTS algorithm: the OTS algorithm is very attractive from the computation point of view. The number of checking points required by the OTS algorithm varies from $(3+2) = 5$ to $(3+1+1+1+1+1+1+2+1+1+1+1+1+1) = 17$, and the number of checking points may take the values 5, 6, 7, 8, …17. An advantage of this algorithm is that it adds only one checking point at a time till reaching the minimum distortion.

3.2 Quality point of view

In this section we introduce the simulation results for a comparison between some well known algorithms. In the simulation we used two different techniques to search for the best matching blocks:

1. Searching within non-overlapped blocks in the search area, as in Fig. 1.
2. Searching within overlapped blocks, as in Fig. 2.

It is clear from Figs. 1, 2, that the difference between the overlapped and non overlapped block technique is that the displacement in the overlapped blocks is in the pixels while in the non overlapped blocks it is an integer number of the block size. Thus for the same complexity the searching window is $(2*p+1)^2*N^2$, and $(2*p+1)^2$ pixels for non overlapped and overlapped techniques, respectively. These two searching windows give the same searching points, and consequently the same complexity.

The comparison between different algorithms in this section will indicate the effect of three major factors in motion estimation algorithms.

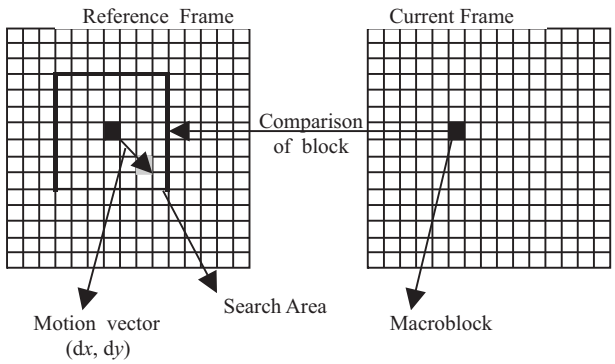

Fig. 1: Searching within non-overlapped blocks

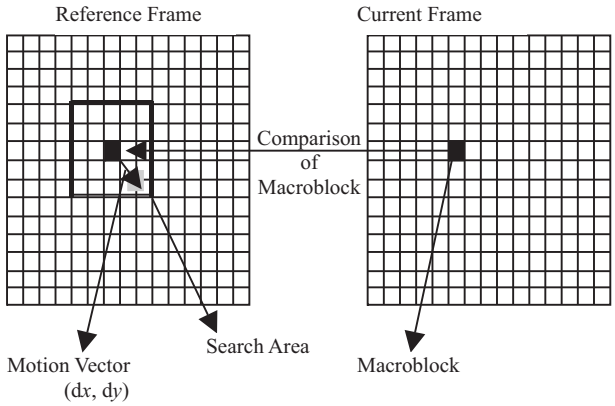

Fig. 2: Searching within overlapped blocks

1) The cost function.
2) The block size.
3) The addition of external noise. The effect of these factors is simulated with almost all the motion types.

3.2.1 The effect of the cost function

The cost function is one of the major factors that affect the complexity of the motion estimation algorithm and consequently its performance. In this section two well known and widely used cost functions are compared from the point of view of complexity and also the effect on the performance of different algorithms:

Mean Square Difference (MSD): to execute an MSD equation four operations have to be performed, namely; one subtraction, one addition, one squaring operation (multiplication), and one division. These operations are performed in addition to the data accessing.

Mean Absolute Difference (MAD): MAD also requires four operations; one subtraction, one absolute value computation, one addition, and one division, plus data accessing.

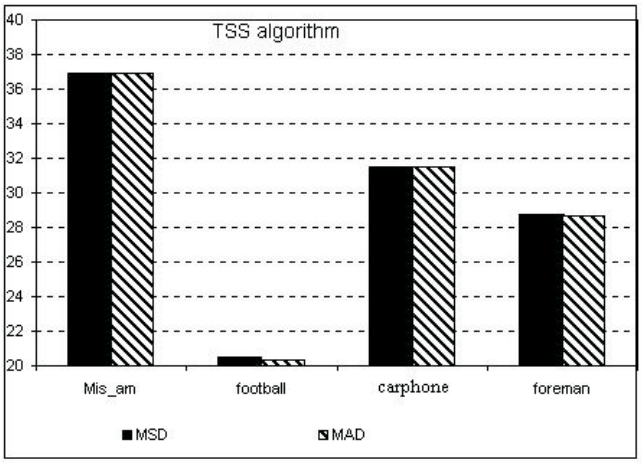

Fig. 3: The effect of the cost function

Table 1: The effect of the cost on different algorithms using the overlapped block technique

	Algorithm	Claire	Akiyo	Mot_daug	Salesman	Grandma	Suzie	Container	Mis_Am	Football	Carfone	Foreman	Silent	News
MSD	FS	43.5474	44.4123	37.2656	39.0550	44.3305	36.8690	43.7964	37.1411	23.5908	33.8072	32.6524	36.3689	38.6385
	3SS	43.5448	44.4123	37.1765	39.1357	44.3257	36.7711	43.7967	37.1357	22.5886	33.6144	32.3541	36.2465	38.5294
	4SS	43.3002	44.2467	36.0448	38.5753	44.2794	35.2129	43.7950	36.9039	21.4003	31.7651	29.3743	35.2058	38.2778
	TDL	43.5448	44.4123	37.1115	39.1318	44.3270	36.3258	43.7966	37.0261	21.9381	33.5907	32.1399	36.2873	38.6219
	OSA	43.5396	44.3946	37.0775	39.0355	44.3205	36.5008	43.7967	37.1225	21.9206	33.4743	32.1316	36.1810	38.6103
	CBOSA	43.5394	44.3946	37.0626	39.0436	44.3205	36.4833	43.7967	37.1220	21.7396	33.4637	32.1477	36.1252	38.6236
	OTS	43.5368	44.3963	37.0518	39.0167	44.3151	36.0907	43.7965	37.0188	21.5874	33.5764	31.9536	36.0010	38.5794
	CSA	43.5214	44.4123	36.9050	38.9088	44.3191	36.1068	43.7966	37.0215	21.8452	33.2189	31.7770	36.0568	38.5700
MAD	FS	43.5145	44.5667	37.1630	39.0550	44.3245	36.8167	43.7964	37.1225	22.9285	33.7505	32.6306	36.1862	38.5197
	3SS	43.5123	44.3828	37.0821	39.0204	44.3198	36.6795	43.7964	37.1067	22.5110	33.5387	32.2222	36.1050	8.5105
	4SS	42.8956	44.0538	36.5073	38.6163	44.2609	35.8949	43.7964	37.0504	21.3811	32.5986	30.5635	35.7104	8.0756
	TDL	43.5123	44.3828	37.0242	39.0184	44.3169	36.2371	43.7963	37.0004	21.8857	33.5071	32.0957	36.0065	8.5076
	OSA	43.4960	44.3812	36.9955	38.9326	44.3095	36.4073	43.7963	37.0951	21.8874	33.4039	31.9828	35.9547	8.5076
	CBOSA	43.4959	44.3812	36.9720	38.9380	44.3110	36.4230	43.7963	37.0946	21.6624	33.3958	32.1301	35.8209	8.5257
	OTS	43.4798	44.3828	36.9568	38.9377	44.3065	36.0126	43.7963	36.9939	21.4116	33.5067	31.9247	35.7376	8.4877
	CSA	43.4879	44.3828	36.8263	38.8235	44.3119	36.0175	43.7963	36.9952	21.8158	33.1682	31.7464	35.8129	8.4893

Table 2: The effect of the cost on different algorithms using the non overlapped block technique

	Algorithm	Claire	Akiyo	Mot_daug	Salesman	Grandma	Suzie	Container	Mis_Am	Football	Carfone	Foreman	Silent	News
MSD	FS	42.7768	44.0277	35.8226	38.0863	44.2413	34.6745	43.7960	36.9393	21.1449	31.5224	28.7398	35.1039	37.9532
	3SS	42.7768	44.0277	35.7407	38.0771	44.2413	34.6135	43.7960	36.9288	20.5063	31.5180	28.7127	34.9695	37.9305
	4SS	42.7768	44.0277	35.5863	38.0771	44.2326	34.5888	43.7960	36.6986	20.1529	31.5064	28.7053	34.7968	37.9298
	TDL	42.7768	44.0277	35.6900	38.0771	44.2413	34.5729	43.7960	36.9077	20.2925	31.5153	28.6511	34.8341	37.9305
	OSA	42.7768	44.0277	35.7155	38.0771	44.2413	34.6088	43.7960	36.9194	20.2273	31.5157	28.6499	34.8917	37.9305
	CBOSA	42.7768	44.0277	35.7126	38.0771	44.2413	34.5949	43.7960	36.9098	20.0801	31.5156	28.6494	34.8287	37.9305
	OTS	42.7768	44.0277	35.6906	38.0771	44.2412	34.5775	43.7960	36.9034	19.9627	31.5152	28.6464	34.7360	37.9305
	CSA	42.7768	44.0277	35.6892	38.0771	44.2413	34.5731	43.7960	36.9076	20.1619	31.5153	28.6465	34.8313	37.9305
MAD	FS	42.7768	44.0277	35.7964	38.0802	44.2409	34.6410	43.7960	36.9267	20.8890	31.5072	28.7184	34.9766	37.9268
	3SS	42.7768	44.0277	35.7332	38.0771	44.2409	34.5961	43.7960	36.9157	20.3351	31.5032	28.7028	34.8675	37.9268
	4SS	42.7768	44.0277	35.7075	38.0771	44.2402	34.5646	43.7960	36.8977	20.0559	31.4973	28.6327	34.7843	37.9268
	TDL	42.7768	44.0277	35.6866	38.0771	44.2409	34.5559	43.7960	36.9036	20.1449	31.5023	28.6475	34.7554	37.9268
	OSA	42.7768	44.0277	35.7074	38.0771	44.2408	34.5793	43.7960	36.9111	20.0946	31.5026	28.6447	34.8347	37.9268
	CBOSA	42.7768	44.0277	35.7085	38.0771	44.2408	34.5720	43.7960	36.9003	19.9489	31.5026	28.6447	34.7537	37.9268
	OTS	42.7768	44.0277	35.6891	38.0771	44.2408	34.5551	43.7960	36.8982	19.8432	31.5022	28.6431	34.6913	37.9267
	CSA	42.7768	44.0277	35.6859	38.0771	44.2409	34.5561	43.7960	36.9036	20.0297	31.5023	28.6431	34.7388	37.9268

It is clear that MAD is simpler than MSD, because an absolute evaluation operation rather is required, than the squaring operation. MAD is therefore preferable to MSD from the complexity point of view.

Tables 1, 2 present a comparison between MSD and MAD cost functions for different algorithms and with different noise-free video sequences from the PSNR point of view, using overlapped blocks and non-overlapped blocks, respectively. Fig. 3 shows an example of the effect of the cost function. In this example the TSS algorithm is used, with a constant block size for the two cases (16×16), and using the overlapped blocks technique. This example illustrates that at *first* MSD achieves better quality than MAD (higher PSNR), but the improvement in the PSNR is small in comparison with the increase in complexity. We can therefore conclude that MAD is better than MSD, when complexity and quality are traded off.

3.2.2 The effect of adding external noise

Video sequences are usually not pure. Some noise almost always corrupts the sequences. Noise may come from the camera (this is called camera noise), or it may be from the transmission lines. The algorithm is therefore required to be robust against the addition of noise. In this section the robust-

Table 3: The effect of the adding gaussian noise with SNR = 25 db on different algorithms using the overlapped block technique.

	Algorithm	Claire	Akiyo	Mot_daug	Salesman	Grandma	Suzie	Container	Mis_Am	Football	Carfone	Foreman	Silent	News
MSD	FS	29.4134	29.6722	27.9724	31.3667	32.6476	28.3379	26.5496	29.2625	22.2051	27.5143	25.1267	26.8225	29.9344
	3SS	29.3249	29.6530	27.9331	31.3790	32.6174	28.2959	26.5087	29.2079	21.4108	27.4111	25.0125	26.8069	29.9308
	4SS	29.2215	29.5168	27.6073	31.0839	32.5855	27.7661	26.4546	29.1125	20.4853	26.7187	24.0649	26.5515	29.7430
	TDL	29.2646	29.6208	27.8652	31.3600	32.5782	28.0768	26.4690	29.1601	20.9107	27.3920	24.9583	26.7881	29.9217
	OSA	29.2977	29.2772	27.8948	31.3532	32.5899	28.1825	26.4952	29.1804	20.8884	27.3524	24.9576	26.7506	29.9178
	CBOSA	29.2772	29.6383	27.8735	31.3498	32.5825	28.1837	26.4729	29.1624	20.7383	27.3422	24.9522	26.7485	29.9188
	OTS	29.2282	29.6234	27.8558	31.3407	32.5574	27.9863	26.4604	29.1183	20.5936	27.3513	24.9028	26.7119	29.9016
	CSA	29.2768	29.6400	27.8504	31.3164	32.5828	28.0142	26.4755	29.1508	20.8304	27.2488	24.8637	26.7336	29.9008
MAD	FS	29.3731	29.6440	27.9357	31.3662	32.6318	28.3129	26.5330	29.2356	21.7096	27.4818	25.1092	26.8004	29.9233
	3SS	29.2870	29.6345	27.9031	31.3654	32.5891	28.2548	26.4859	29.1750	21.3939	27.3836	24.9587	26.7773	29.9067
	4SS	29.1674	29.5069	27.5440	31.0566	32.5605	27.7385	26.4079	29.0629	20.4072	26.6927	24.0404	26.5357	29.7871
	TDL	29.2655	29.6242	27.8732	31.3542	32.5752	28.0672	26.4666	29.1400	20.8855	27.3786	24.9412	26.7557	29.8866
	OSA	29.2553	29.6275	27.8682	31.3245	32.5648	28.1567	26.4643	29.1464	20.8823	27.3007	24.9019	26.7331	29.8984
	CBOSA	29.2482	29.6196	27.8515	31.3331	32.5674	28.1686	26.4541	29.1291	20.7114	27.3091	24.9375	26.7147	29.8993
	OTS	29.1998	29.6136	27.8311	31.3347	32.5497	27.9681	26.4319	29.0930	20.4992	27.3310	24.8916	26.6859	29.8788
	CSA	29.2475	29.6129	27.8241	31.3043	32.5765	27.9798	26.4536	29.1222	20.8277	27.2225	24.8538	26.7089	29.8867

Table 4: The effect of the adding gaussian noise with SNR = 20 dB on different algorithms using the overlapped block technique

	Algorithm	Claire	Akiyo	Mot_daug	Salesman	Grandma	Suzie	Container	Mis_Am	Football	Carfone	Foreman	Silent	News
MSD	FS	24.6454	24.9011	23.5174	27.0233	27.9679	23.9352	21.7355	24.8762	20.1625	23.4123	20.7877	22.3660	25.6711
	3SS	24.5211	24.8633	23.4701	27.0220	27.9273	23.8821	21.6488	24.7843	19.6216	23.3344	20.7060	22.3418	25.6469
	4SS	24.4304	24.7429	23.2883	26.8762	27.8972	23.5996	21.5808	24.6799	18.9677	22.9831	20.2567	22.1961	25.5331
	TDL	24.1756	24.6807	23.1540	26.8660	27.7889	23.2628	21.4326	24.5156	17.1585	22.8037	20.0480	22.0807	25.4713
	OSA	24.4690	24.8423	23.4395	27.0024	27.9053	23.8089	21.6154	24.7411	19.2658	23.2817	20.6498	22.3179	25.6331
	CBOSA	24.4520	24.8295	23.4238	27.0083	27.8949	23.8033	21.6038	24.7352	19.1476	23.2705	20.6522	22.3054	25.6251
	OTS	24.3842	24.8056	23.3890	27.0009	27.8650	23.6878	21.5592	24.6723	19.0078	23.2578	20.6144	22.2789	25.6066
	CSA	24.4600	24.8181	23.4024	26.9950	27.8948	23.7152	21.6043	24.7266	19.2243	23.2294	20.6068	22.3022	25.6183
MAD	FS	24.5962	24.8687	23.4848	27.0217	27.9538	23.9111	21.7139	24.8364	19.8365	23.3842	20.7733	22.3443	25.6414
	3SS	24.4744	24.8321	23.4439	27.0071	27.9037	23.8458	21.6215	24.7473	19.5976	23.2996	20.6707	22.3190	25.6275
	4SS	24.3854	24.7030	23.2590	26.8540	27.8720	23.5583	21.5363	24.6361	18.9205	22.9048	20.2166	22.1685	25.4961
	TDL	24.4459	24.8089	23.4104	27.0036	27.8813	23.7428	21.5894	24.7060	19.2689	23.2678	20.6338	22.3028	25.6163
	OSA	24.4389	24.8117	23.4037	26.9944	27.8869	23.7785	21.5824	24.7152	19.2500	23.2569	20.6096	22.2989	25.6077
	CBOSA	24.4205	24.8077	23.3936	26.9983	27.8828	23.7750	21.5729	24.7009	19.1366	23.2532	20.6168	22.2807	25.6074
	OTS	24.3610	24.7839	23.3691	26.9929	27.8573	23.6601	21.5393	24.6434	18.9239	23.2314	20.5835	22.2549	25.5929
	CSA	24.4250	24.7997	23.3828	26.9821	27.8781	23.6838	21.5767	24.6980	19.2166	23.2077	20.5909	22.2828	25.6116

Table 5: The effect of adding gaussian noise with SNR = 25 dB on different algorithms using the non overlapped block technique

	Algorithm	Claire	Akiyo	Mot_daug	Salesman	Grandma	Suzie	Container	Mis_Am	Football	Carefone	Foreman	Silent	News
MSD	FS	29.0930	29.5213	27.5724	31.0762	32.5427	27.4645	26.4322	29.0839	20.2670	26.5460	23.8262	26.5191	29.6984
	3SS	29.0657	29.5130	27.5307	31.0702	32.5258	27.4107	26.4250	29.0498	19.7371	26.5291	23.8154	26.4794	29.6919
	4SS	29.0476	29.5110	27.2323	31.0675	32.3083	27.3863	26.0877	27.8835	19.4417	26.0194	23.7917	26.4159	29.6272
	TDL	29.0479	29.5177	27.5052	31.0746	32.5254	27.3728	26.4076	29.0410	19.5542	26.5222	23.7758	26.4352	29.6916
	OSA	29.0564	29.5128	27.5225	31.0722	32.5274	27.4079	26.4179	29.0408	19.5026	26.5223	23.7677	26.4488	29.6938
	CBOSA	29.0541	29.5216	27.5128	31.0729	32.5258	27.4006	26.4194	29.0499	19.3747	26.5331	23.7709	26.4401	29.6972
	OTS	29.0441	29.5246	27.4973	31.0678	32.5225	27.3789	26.4079	29.0296	19.2696	26.5294	23.7708	26.4047	29.6887
	CSA	29.0503	29.5131	27.4994	31.0653	32.5167	27.3825	26.4123	29.0487	19.4416	26.5222	23.7655	26.4304	29.6934
MAD	FS	29.0773	29.5214	27.5656	31.0738	32.5388	27.4433	26.4189	29.0693	20.1119	26.5412	23.8153	26.5097	29.6909
	3SS	29.0538	29.5187	27.5209	31.0715	32.5279	27.4051	26.4028	29.0461	19.6215	26.5306	23.8017	26.4575	29.6833
	4SS	29.0437	29.5177	27.2766	31.0724	32.3003	27.4051	26.0138	28.2736	19.4122	25.8462	23.7570	26.4090	29.5894
	TDL	29.0399	29.5130	27.4993	31.0715	32.5218	27.3738	26.4098	29.0398	19.4583	26.5241	23.7739	26.4226	29.6978
	OSA	29.0470	29.5115	27.5164	31.0717	32.5138	27.3904	26.4032	29.0375	19.4047	26.5297	23.7771	26.4457	29.6932
	CBOSA	29.0399	29.5137	27.5109	31.0646	32.5212	27.3772	26.4126	29.0339	19.2773	26.5231	23.7681	26.4237	29.6886
	OTS	29.0414	29.5129	27.4869	31.0737	32.5192	27.3670	26.3943	29.0230	19.1884	26.5269	23.7751	26.3996	29.6806
	CSA	29.0347	29.5138	27.5039	31.0686	32.5153	27.3698	26.4041	29.0275	19.3566	26.5172	23.7731	26.4216	29.6870

ness of some algorithms is tested. Here we used white Gaussian noise, with SNR of 25 db and 20 db. The results are shown in tables 3, 4, 5, and 6. Also, an example to indicate the relation between noise (SNR) and quality (PSNR) is shown in Fig. 4. In this example FS, TSS, TDL, and 4SS are used to compare their robustness to noise. Another example is shown in that figure to indicate the robustness of one algorithm (TSS chosen) with a different video sequence. These two examples show that as the noise increases the quality decreases, and the FS algorithm is the best even with addition of heavy noise. The TSS algorithm is the second best algorithm for noisy seqences.

3.2.3 The Effect of Block Size

The choice of macroblock size or simply block size (NxM) is the result of tradeoffs among three conflicting requirements. Specifically,

1. Small values for *N* and *M* (from four to eight) are preferable, since the smoothness constraint would be easily met at this resolution;

Table 6: The effect of adding gaussian noise with SNR = 20 dB on different algorithms using the non overlapped block technique

	Algorithm	Claire	Akiyo	Mot_daug	Salesman	Grandma	Suzie	Container	Mis_Am	Football	Carefone	Foreman	Silent	News
MSD	FS	24.3748	24.7095	23.2762	26.8718	27.8667	23.4065	21.5700	24.6805	18.8333	22.8580	20.0903	22.1782	25.5085
	3SS	24.3222	24.7058	23.2421	26.8693	27.8518	23.3709	21.5412	24.6382	18.4370	22.8450	20.0787	22.1503	25.5035
	4SS	24.2718	24.6826	23.0947	26.8716	27.7363	23.3344	21.2836	23.6099	18.2048	22.3892	20.0444	22.1158	25.3588
	TDL	24.2923	24.6975	23.2296	26.8707	27.8417	23.3347	21.5069	24.6061	18.2926	22.8312	20.0591	22.1394	25.4902
	OSA	24.2986	24.7051	23.2385	26.8700	27.8328	23.3491	21.5340	24.6263	18.2527	22.8380	20.0638	22.1335	25.5038
	CBOSA	24.2952	24.7053	23.2393	26.8694	27.8379	23.3401	21.5068	24.6128	18.1633	22.8321	20.0698	22.1425	25.4971
	OTS	24.2769	24.7001	23.2245	26.8663	27.8272	23.3278	21.5048	24.5963	18.0727	22.8316	20.0630	22.1216	25.4871
	CSA	24.2765	24.6929	23.2226	26.8746	27.8400	23.3329	21.4946	24.6023	18.2132	22.8342	20.0600	22.1262	25.5020
MAD	FS	24.3433	24.6944	23.2724	26.8684	27.8606	23.3890	21.5432	24.6478	18.7474	22.8499	20.0904	22.1633	25.5024
	3SS	24.3007	24.6900	23.2391	26.8659	27.8464	23.3545	21.5169	24.6232	18.3712	22.8315	20.0823	22.1440	25.4979
	4SS	24.2659	24.6835	23.1040	26.8711	27.7022	23.3391	21.1987	23.3878	18.1935	22.3078	20.0516	22.1182	25.3493
	TDL	24.2692	24.6956	23.2205	26.8683	27.8424	23.3278	21.5026	24.5962	18.2356	22.8331	20.0582	22.1255	25.4899
	OSA	24.2892	24.6942	23.2295	26.8631	27.8355	23.3441	21.5178	24.6081	18.1976	22.8319	20.0643	22.1415	25.4919
	CBOSA	24.2815	24.6993	23.2415	26.8700	27.8335	23.3378	21.5102	24.5968	18.1053	22.8273	20.0666	22.1297	25.4968
	OTS	24.2642	24.6943	23.2235	26.8684	27.8189	23.3240	21.4948	24.5876	18.0279	22.8249	20.0606	22.1158	25.4806
	CSA	24.2611	24.6924	23.2219	26.8695	27.8274	23.3234	21.5016	24.5919	18.1615	22.8293	20.0672	22.1322	25.4872

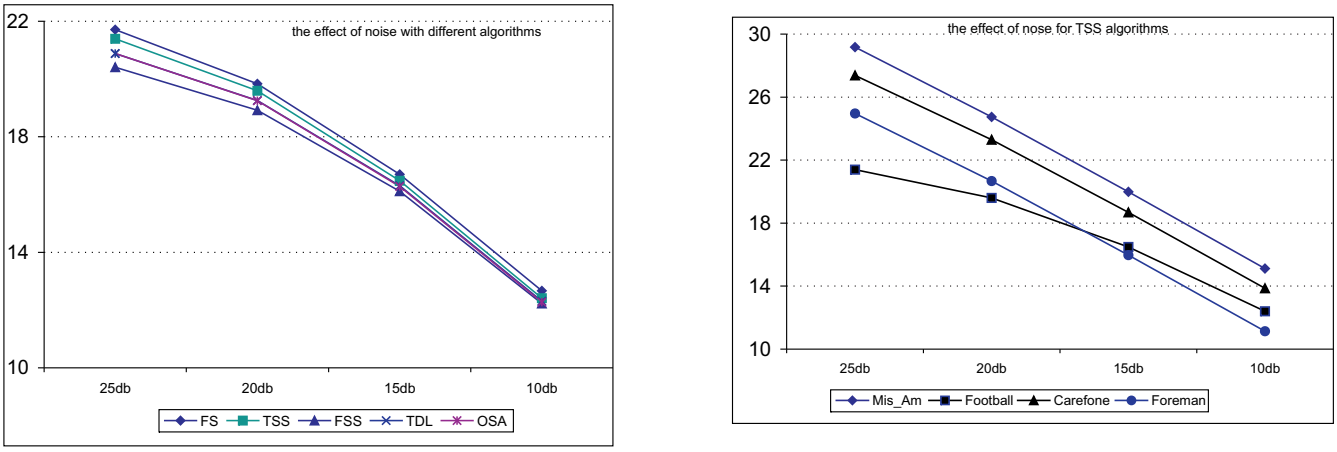

Fig. 4: The effect of noise on different algorithms, with different sequences

2. Small values for N and M reduce the reliability of the motion vector, since few pixels participate in the matching process;
3. Fast algorithms for finding motion vectors are more efficient for larger values of N and M.

In this section we will show the effect of the block size on the performance of the algorithms. In the simulation, different block sizes (4×4, 8×8, and 16×16) are compared using both the overlapped block and non-overlapped block techniques. MAD is used as a cost function, and the searching win-

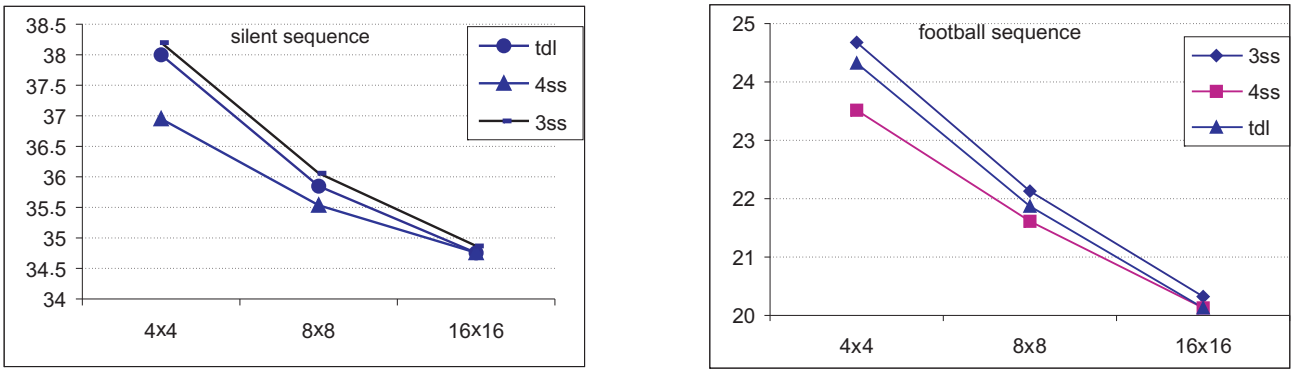

Fig. 5: Two examples showing the effect of block size

Table 7: The effect of block size on different algorithms using the non overlapped block technique

The cost function is the Mean Absolute Difference (MAD) Noise-free seguences									
Video sequence	Block size	FS	3SS	4SS	TDL	OSA	CBOSA	OTS	CSA
Claire	16×16	42.7768	42.7768	42.7768	42.7768	42.7768	42.7768	42.7768	42.7768
	8×8	42.7894	42.7862	42.7770	42.7846	42.7860	42.7839	42.7817	42.7837
	4×4	43.1684	43.0100	42.9610	42.9835	42.9487	42.9406	42.8924	42.9281
Akiyo	16×16	44.0277	44.0277	44.0277	44.0277	44.0277	44.0277	44.0277	44.0277
	8×8	44.0497	44.0407	44.0349	44.0397	44.0397	44.0373	44.0392	44.0407
	4×4	44.5296	44.3550	44.2828	44.3275	44.2827	44.2437	44.1994	44.2406
Mot_daug	16×16	35.7964	35.7332	35.7075	35.6866	35.7074	35.7085	35.6891	35.6859
	8×8	36.2719	36.0978	35.9943	35.9936	36.0011	35.9716	35.8743	35.9166
	4×4	37.5963	36.9544	36.6822	36.7944	36.6327	36.5454	36.4824	36.6330
Salesman	16×16	38.0802	38.0771	38.0771	38.0771	38.0771	38.0771	38.0771	38.0771
	8×8	38.3956	38.2350	38.1451	38.1958	38.1753	38.1675	38.1539	38.1871
	4×4	39.7085	39.1389	38.9262	39.0782	38.9602	38.8460	38.9358	38.9625
Grandma	16×16	44.2409	44.2409	44.2402	44.2409	44.2408	44.2408	44.2408	44.2409
	8×8	44.2705	44.2624	44.2528	44.2613	44.2591	44.2574	44.2568	44.2576
	4×4	44.5451	44.4325	44.3738	44.4160	44.3980	44.3544	44.3622	44.3844
Suzie	16×16	34.6410	34.5961	34.5646	34.5559	34.5793	34.5720	34.5551	34.5561
	8×8	35.1784	35.0081	34.9003	34.9428	34.9279	34.8953	34.8585	34.9217
	4×4	36.4271	35.8496	35.6005	35.7433	35.5897	35.4816	35.5631	35.6111
Mis_Am	16×16	36.9267	36.9157	36.8977	36.9036	36.9111	36.9003	36.8982	36.9036
	8×8	37.2996	37.2038	37.1216	37.1297	37.1608	37.1419	37.0462	37.0897
	4×4	38.5524	38.1191	37.8946	37.9207	37.9483	37.8733	37.6502	37.7892
Container	16×16	43.7960	43.7960	43.7960	43.7960	43.7960	43.7960	43.7960	43.7960
	8×8	43.7961	43.7961	43.7961	43.7961	43.7961	43.7961	43.7961	43.7961
	4×4	43.8102	43.8064	43.8036	43.8057	38.9602	43.8054	43.8042	43.8049
Carfone	16×16	31.5072	31.5032	31.4973	31.5023	31.5026	31.5026	31.5022	31.5023
	8×8	31.9189	31.8412	31.6262	31.8163	31.8280	31.8095	31.7954	31.7886
	4×4	33.5332	32.8910	32.4287	32.7960	32.7217	32.6610	32.6191	32.5616
Foreman	16×16	28.7184	28.7028	28.6327	28.6475	28.6447	28.6447	28.6431	28.6431
	8×8	29.4459	29.0493	28.8682	28.8742	28.7888	28.7704	28.7340	28.7767
	4×4	31.4843	30.3725	29.7362	29.9669	29.5708	29.4712	29.4314	29.6674
Silent	16×16	34.9766	34.8675	34.7843	34.7554	34.8347	34.7537	34.6913	34.7388
	8×8	36.5615	36.0605	35.7335	35.8490	35.7905	35.6099	35.5770	35.7755
	4×4	39.3660	38.1995	37.7022	37.9981	37.6317	.3660	37.3521	37.6430
News	16×16	37.9268	37.9268	37.9268	37.9268	37.9268	37.9268	37.9267	37.9268
	8×8	38.2139	38.1805	38.1177	38.1706	38.1607	38.1646	38.1432	38.1534
	4×4	39.6129	39.1762	39.0289	39.1427	39.0072	38.9403	38.8082	38.8152
Football	16×16	20.8890	20.3351	20.0559	20.1449	20.0946	19.9489	19.8432	20.0297
	8×8	23.1485	22.1201	21.7585	21.8682	21.6609	21.4343	21.3743	21.6571
	4×4	26.5200	24.6827	24.1832	24.3096	23.7828	23.3710	23.2316	23.8813

dow used is 15×15 (searching area parameter p=7). The simulation results are shown in Tables 7 and 8. Two different examples indicating the block size effect are shown in Fig. 5. These two examples show that the PSNR decreases as the block size increases. The increase in PSNR is at the cost of increasing the computation time.

Table 8: The effect of the block size on different algorithms using the overlapped block technique

The cost function is the Mean Absolute Difference (MAD) Noise-free sequences									
Video sequence	Block size	FS	3SS	4SS	TDL	OSA	CBOSA	OTS	CSA
Claire	16×16	43.5145	43.5123	42.8956	43.5123	43.4960	43.4959	43.4798	43.4879
	8×8	43.8594	43.7666	43.0371	43.7712	43.4959	43.4944	43.4718	43.6813
	4×4	44.5517	43.9486	43.4459	44.0145	43.7641	43.7465	43.7740	43.8218
Akiyo	16×16	44.5667	44.3828	44.0538	44.3828	44.3812	44.3812	44.3828	44.3828
	8×8	44.8593	44.7638	44.1545	44.7828	44.6570	44.6578	44.6489	44.7087
	4×4	45.6166	45.1495	44.7576	45.2207	45.0221	45.0259	45.1116	45.1231
Mot_daug	16×16	37.1630	37.0821	36.5073	37.0242	36.9955	36.9720	36.9565	36.8263
	8×8	38.3390	37.7442	37.0398	37.6741	37.4961	37.5502	37.5607	37.3572
	4×4	39.2502	38.4231	37.7508	38.4135	37.9643	38.0876	38.2214	38.0042
Salesman	16×16	39.0550	39.0204	38.6163	39.0184	38.9326	38.9380	38.9377	38.8235
	8×8	40.0834	39.7301	39.2691	39.7364	39.5043	39.5443	39.4983	39.5222
	4×4	41.3464	40.6167	40.1940	40.6732	40.2996	40.3844	40.4402	40.2478
Grandma	16×16	44.3245	44.3198	44.2609	44.3169	44.3095	44.3110	44.3076	44.3119
	8×8	44.5781	44.5037	44.3273	44.4985	44.4782	44.4742	44.4634	44.4731
	4×4	45.0130	44.8618	44.5958	44.8534	44.8045	44.8041	44.7732	44.7762
Suzie	16×16	36.8167	36.6795	35.8949	36.2371	36.4073	36.4230	36.0087	36.0175
	8×8	37.8662	37.3184	36.3338	37.0996	36.8292	36.8412	36.7084	36.6589
	4×4	38.9663	37.9522	37.0740	37.8821	37.3160	37.3391	37.4694	37.2577
Mis_Am	16×16	37.1225	37.1067	37.0504	37.0004	37.0951	37.0946	36.9958	36.9952
	8×8	37.8509	37.5961	37.4185	37.4205	37.5131	37.4977	37.3306	37.3833
	4×4	39.0408	38.7229	38.4166	38.4621	38.4970	38.4544	38.1839	38.3203
Container	16×16	43.7964	43.7964	43.7964	43.7963	43.7963	43.7963	43.7964	43.7963
	8×8	43.8018	43.7997	43.7995	43.7994	43.7994	43.7995	43.7992	43.7994
	4×4	43.8399	43.8288	43.8164	43.8288	43.8282	43.8285	43.8245	43.8235
Carfone	16×16	33.7505	33.5387	32.5986	33.5071	33.4039	33.3958	33.5072	33.1682
	8×8	35.1353	34.2742	33.2570	34.2545	34.0278	34.0320	34.2366	33.5551
	4×4	36.3276	35.2296	34.2488	35.2128	34.7692	34.8781	35.1781	34.1129
Foreman	16×16	32.6306	32.2222	30.5635	32.0957	31.9828	32.1301	31.9249	31.7464
	8×8	34.1180	32.9344	31.4437	32.7939	32.4283	32.6552	32.6965	32.2877
	4×4	35.4050	33.4195	32.4833	33.3044	32.7338	33.1495	33.4824	32.6653
Silent	16×16	36.1862	36.1050	35.7104	36.0065	35.9547	35.8209	35.7348	35.8129
	8×8	38.9408	38.0916	37.3871	37.9064	37.5271	37.4855	37.2798	37.5107
	4×4	40.9393	39.8626	38.8732	39.6441	38.8536	38.8269	38.6559	39.0480
News	16×16	38.5197	38.5105	38.0756	38.5076	38.5076	38.5257	38.4919	38.4893
	8×8	39.7669	39.3590	38.5155	39.3298	39.2583	39.3243	39.3236	39.1938
	4×4	41.2435	40.3439	39.6528	40.3543	40.1416	40.1696	40.4380	39.9729
Football	16×16	22.9285	22.5110	22.5599	21.8857	21.8874	21.6624	21.6624	21.8158
	8×8	26.3718	24.1567	22.8188	23.3926	23.1560	22.9056	22.7096	23.2567
	4×4	27.7078	25.8479	23.9034	24.9480	24.2826	23.7844	23.6967	24.8451

Fig. 6: (a) the original 2nd frame, (b) the original 5th frame, (c) the original 8th frame, (d) reconstructed frame 2 using the FS algorithm, (e) reconstructed frame 5 using the FS algorithm, (f) reconstructed frame 8 using the FS algorithm, (g) reconstructed frame 2 using the TSS algorithm, (h) reconstructed frame 5 using the TSS algorithm, (i) reconstructed frame 8 using the TSS algorithm

3.3 Visual results

In this section the reconstructed frames will be presented with the use of FS and TSS algorithms and with MSD as a cost function and for the overlapped technique. For comparison we used three video sequences, specifically;

Claire video sequence: this represents a head and shoulder sequence, and has just one moving object with slow motion.

Mother & Daughter sequence: this also represents a head, and shoulder sequence, but it has just two moving objects with slow motion.

Football sequence: this represents a multi-object sequence with fast motion.

For these three sequences the reconstructed frame is shown, the comparison is performed by estimating frame number $n+k$ from a reference frame n. For each sequence three cases are performed with $k=1$, $k=4$, and $k=7$. This is shown in Fig. 6 and Fig. 7. The PSNR corresponding to these cases is shown in Table 9. The simulation results show that the quality of the reconstructed frame decreases as the number of skipped frames increases (k). Appearing and disappearing of objects during the sequence also decreases the quality of the reconstructed frames.

Table 9: The PSNR performance for two algorithms

	Claire			Football		
Ref. →Cur.	1→2	1→5	1→8	1→2	1→5	1→8
FS	40.9697	33.9463	31.5458	25.2897	20.8944	19.7531
TSS	40.9697	33.7515	31.3894	23.7676	19.5222	18.3951

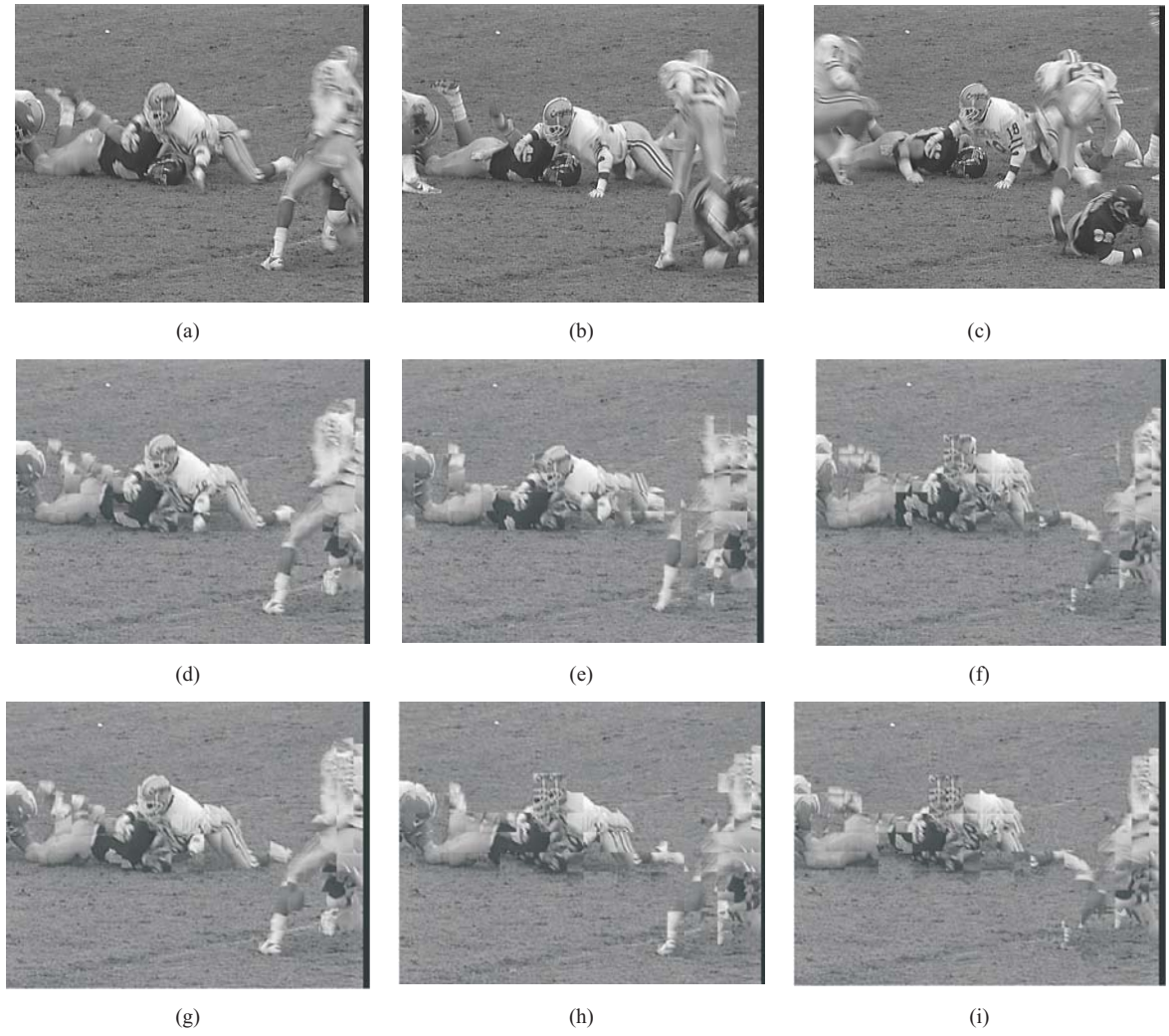

Fig. 7: (a) the original 2nd frame, (b) the original 5th frame, (c) the original 8th frame, (d) reconstructed frame 2 using TSS algorithm, (e) reconstructed frame 5 using the FS algorithm, (f) reconstructed frame 8 using the FS algorithm, (g) reconstructed frame 2 using the TSS algorithm, (h) reconstructed frame 5 using the TSS algorithm, (i) reconstructed frame 8 using the TSS algorithm

4 Conclusion

From the simulation results we can conclude that:

- There are two techniques for searching for the best matching, namely; 1) searching within non-overlapped blocks, 2) searching within overlapped blocks.
- A comparison between these two techniques was performed using the same searching algorithm, the same block size, the same cost function, and with the same complexity, i.e. searching points. The simulation indicates that searching within the overlapped blocks is the better from the quality point of view.
- The full search algorithm is the best algorithm from the quality point of view, but from the computation time (complexity) point of view it is the worst.
- The TSS algorithm is the best algorithm from the trade off quality – complexity point of view.
- The block size is one of the effective factors in the motion estimation algorithms. Small block size (such as 4×4 and 8×8) results in good quality, but reduces the reliability of the motion vector, since few pixels participate in the matching process. On the other hand, large block sizes (such as 16×16) are preferable for fast algorithms.
- The cost function affects the complexity of the searching algorithm. A comparison between MAD and MSD cost functions indicates that MSD achieves greater quality than MAD at the cost of increasing the complexity. MAD is preferable, since the difference in quality is very small.
- The addition of white Gaussian noise affects the direction of the motion vectors; consequently the reconstructed frame has less quality.

References

[1] Vasuev Bhaskaran, Konstantinos Konstantinides: "Image and Video Compression Standards Algorithms and Architectures." Second edition, Kluwer Academic Publishers.

[2] Koga, T., Linuma, K., Hirano, A., Lijima, Y., Ishiguro, T.: "Motion Compensated Interframe Coding for Video Conferencing." In proc. Nat: Telecomm. Conf. New Orleans, LA, p. G5.3.1–5.3.5., Nov. 29–Dec.3, 1981.

[3] Lai-Man Po, Wang-Chung Ma: "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," IEEE Trans. Circuits Syst. Video Technol., Vol. **6**, (Jun. 1996), No. 3, p. 313–317.

[5] Usama, S., Simak, B.: "A Low-complexity Wavelet Based Algorithm for Inter-frame Image Prediction." In: *Acta Polytechnica*. Vol. **42**, (2002), No. 2/2002, p. 30–34. ISSN 1210-2709.

[6] Jain, J. R., Jain, A. K.: "Displacement Measurement and its Application in Interframe Image Coding." *IEEE Transactions on Communications*, Volume COM-29, Number 12, p. 1799–1808, Dec. 1981.

[7] Puri, A., Hang, H.-M., Schilling, D. L.: "An Efficient Block-matching Algorithm for Motion Compensated Coding." Proceedings IEEE ICASSP (International Conference on Acoustics, Speech and Signal Processing), 1987 p. 25.4.1–24.4.4.

[8] Po, L. M., Ma, W. C.: "A New Center-biased Search Algorithm for Block Motion Estimation." Proceedings ICIP 95, Vol. I, Dec. 1995, p. 410–413.

[8] Rao, K. R., Hwang, J. J.: "Techniques & Standards For Image, Video & Audio Coding." Brazil, Rio de Janeiro, Prentice Hall, 1996.

[9] Srinivasan, R., Rao, K. R.: "Predictive Coding Based on Efficient Motion Estimation." *IEEE Trans. On Circuits and Systems in Video Technology*, Vol. Com-33, No. 8, Aug. 1985, p. 888–896.

[10] Ghanbari, M.: "The Cross-search Algorithm for Motion Estimation." *IEEE Transactions on Communications*, Volume 38, July 1990, No. 7, p. 950–953.

Dr. Sayed Usama
phone: +20101346626
e-mail: usama@acc.aun.edu.eg,
usama_s_1999@yahoo.com

Faculty of Engineering,
Assiut University
Assiut, Egypt

Dr. Montaser Mohamed

Faculty of Engineering
South Valley University
Aswan, Egypt

Eng. Osama Ahmed
e-mail: omer.osama@gmail.com
osama_12000@yahoo.com

Faculty of Engineering
South Valley University
Aswan, Egypt

