EXPLORATIONS ON ENGINEERING LETTERS (EEL)
VOLUME 1, ISSUE 1 (2016):PP.154-163
SANA ACADEMIC PRESS

An Efficient way of Compressing Motion Pictures in CCTV Videos

M.Ananthi[1], R. Latha [2]
[1]Research Scholar, St. Peter's University, Chennai.7
[2]Prof. & Head., Dept. of Computer Applications, St. Peter's University, Chennai
ananthivijay1@gmail.com

Abstract*— In recent years, many algorithms are proposed for reducing the computational cost and for providing good quality. In that our project, efficient way of compressing motion pictures helps in providing reduced computational cost and also with high quality. We make use of orthogonal diamond search algorithm which overcomes the previous algorithm. In this algorithm, we start with the orthogonal way of search and then ends up with diamond way of search. The existing work was impractical because of two reasons : 1) larger number of computation which was complex it compute 2) performance are analyzed interms of search points which was time consuming. Our project overcome these two drawbacks. The input sent will be the video and we encrypt it as images. Then the unwanted background are removed so that the capacity of storage can increase. We make use of orthogonal diamond search for computation to get a better result. This project results with an efficient, less time consuming and good quality video.*

Keywords: *Motion estimation, Block matching algorithm, Orthogonal diamond search, Diamond search*

I . INTRODUCTION

With the advent of the multimedia age and the spread of Internet, video storage on CD/DVD and streaming video has been gaining a lot of popularity. The ISO Moving Picture Experts Group (MPEG) video coding standards pertain towards compressed video storage on physical media like CD/DVD, where asthe International Telecommunications Union (ITU) addresses real-time point-to-point or multi-point communications over a network. The former has the advantage of having higher bandwidth for data transmission.

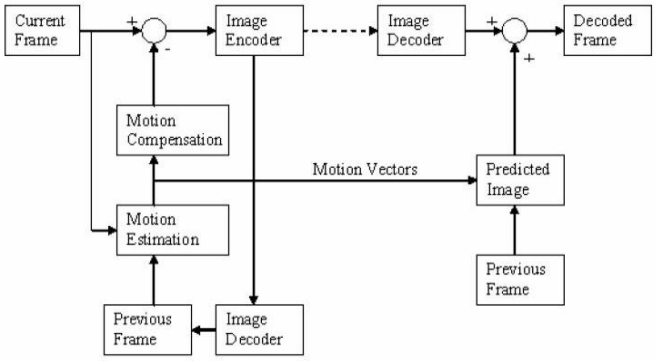

Figure 1.1 MPEG / H.26x video compression process flow

In either standard the basic flow of the entire compression decompression process is largely the same and is depicted in Fig. 1.1.The encoding side estimates the motion in the current frame with respect to a previous frame. A motion compensated image for the current frame is then created that is built of blocks of image from the previous frame. The motion vectors for blocks used for motion estimation are transmitted, as well as the difference of the compensated image with the current frame is also JPEG encoded and sent. The encoded image that is sent is then decoded at the encoder and used as a reference frame for the subsequent frames. The decoder reverses the process and creates a full frame. The whole idea behind motion estimation based video compression is to save on bits by sending JPEG encoded difference images which inherently have less energy and can be highly compressed as compared to sending a full frame that is JPEG encoded. Motion JPEG, where all frames are JPEG encoded, achieves anything between 10:1 to 15:1 compression ration, where as MPEG can achieve a compression ratio of 30:1 and is also useful at 100:1 ratio. It should be noted that the first frame is always sent full, and so are some other frames that might occur at some regular interval (like every 6th frame). The standards do not specify this and this might change with every video being sent based on the dynamics of the video

.

BLOCK MATCHING ALGORITHM

A Block Matching Algorithm is a way of locating matching macroblocks in a sequence of digital video frames for the purposes of motion estimation. The underlying supposition behind motion estimation is that the patterns corresponding to objects and background in a frame of video sequence move within the frame to form corresponding objects on the subsequent frame. This can be used to discover temporal redundancy in the video sequence, increasing the effectiveness of inter-frame video compression by defining the contents of a macroblock by reference to the contents of a known macroblock which is minimally different.

A block matching algorithm involves dividing the current frame of a video into macroblocks and comparing each of the macroblocks with a corresponding block and its adjacent neighbors in a nearby frame of the video (sometimes just the previous one). A vector is created that models the movement of a macroblock from one location to another. This movement, calculated for all the macroblocks comprising a frame, constitutes the motion estimated in a frame.

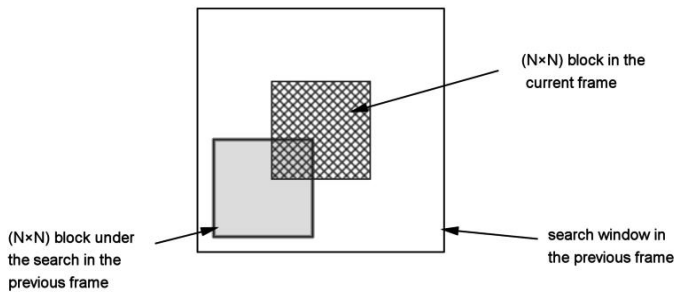

Figure 1.2 Block-matching algorithm

The search area for a good macroblock match is decided by the ‘search parameter’, p, where p is the number of pixels on all four sides of the corresponding macro-block in the previous frame. The search parameter is a measure of motion. The larger the value of p, larger is the potential motion and the possibility for finding a good match. A full search of all potential blocks however is a computationally expensive task. Typical inputs are a macroblock of size 16 pixels and a search area of p = 7 pixels.

1.1.1 Motivation

Motion estimation is the process of determining motion vectors that describe the transformation from one 2D image to another; usually from adjacent frames in a video sequence. The motion vectors may relate to the whole image (global motion estimation) or specific parts, such as rectangular blocks, arbitrary shaped patches or even per pixel. The motion vectors may be represented by a translational model or many other models that can approximate the motion of a real video camera, such as rotation and translation in all three dimensions and zoom. Applying the motion vectors to an image to predict the transformation to another image, on account of moving camera or object in the image is called motion compensation. The combination of motion estimation and motion compensation is a key part of video compression as used by MPEG 1, 2 and 4 as well as many other video codecs. Motion estimation based video compression helps in saving bits by sending encoded difference images which have inherently less entropy as opposed to sending a fully coded frame. However the most computationally expensive and resource extensive operation in the entire compression process is motion estimation. Hence, fast and computationally inexpensive algorithms for motion estimation is a need for video compression.

1.1.2 Evaluation Metrics

A metric for matching a macroblock with another block is based on a cost function. The most popular in terms of computational expense is:

Mean difference or Mean Absolute Difference (MAD) = $$\frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}|C_{ij}-R_{ij}|$$

Mean Squared Error (MSE) = $$\frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(C_{ij}-R_{ij})^2$$

where N is the size of the macro-block, and C_{ij} and R_{ij} are the pixels being compared in current macroblock and reference macroblock, respectively. The motion compensated image that is created using the motion vectors and macroblocks from the reference frame is characterized by Peak signal-to-noise ratio (PSNR),

$$\mathrm{PSNR} = 10\log_{10}\frac{(\text{peak to peak value of original data})^2}{\mathrm{MSE}}$$

II . MATERIAL METHODS

Block Matching algorithms have been researched since mid-1980's. Many algorithms have been developed, but only some of the most basic or commonly used have been described below.

1.2.1 Exhaustive Search

This algorithm calculates the cost function at each possible location in the search window. This leads to the best possible match of the macro-block in the reference frame with a block in another frame. The resulting motion compensated image has highest peak signal-to-noise ratio as compared to any other block matching algorithm. However this is the most computationally extensive block matching algorithm among all. A larger search window requires greater number of computations.

1.2.2 Three Step Search

It is one of the earliest fast block matching algorithm. It runs as follows:

1. Start with search location at center
2. Set step size 'S' = 4 and search parameter 'p' = 7
3. Search 8 locations +/- S pixels around location (0,0) and the location (0,0)
4. Pick among the 9 locations searched, the one with minimum cost function
5. Set the new search origin to the above picked location
6. Set the new step size as S = S/2
7. Repeat the search procedure until S = 1

The resulting location for S=1 is the one with minimum cost function and the macro block at this location is the best match.There is a reduction in computation by a factor of 9 in this algorithm. For p=7, while ES evaluates cost for 225 macro-blocks, TSS evaluates only for 25 macro blocks.

1.2.3 Four Step Search

Four Step Search is an improvement over TSS in terms of lower computational cost and better peak signal-to-noise ratio. Similar to NTSS, FSS also employs center biased searching and has a halfway stop provision.

The algorithm runs as follows:

1. Start with search location at center
2. Set step size 'S' = 2, (irrespective of search parameter 'p')
3. Search 8 locations +/- S pixels around location (0,0) as shown in figure
4. Pick among the 9 locations searched, the one with minimum cost function
5. If the minimum weight is found at center for search window:

a. Set the new search origin as shown in figure 7(d)

b. Set the new step size as S = S/2 = 1

c. Repeat the search procedure from steps 3 to 4

d. Select location with the least weight as motion vector

6. If the minimum weight is found at one of the 8 locations other than the center:

a. Set the new origin to this location

b. Fix the step size as S = 2

c. Repeat the search procedure from steps 3 to 4. Depending on location of new origin, search through 5 locations or 3 locations

d. Select the location with the least weight

e. If the least weight location is at the center of new window go to step 5, else go to step 6

1.3.1 Diamond Search

Diamond Search (DS) algorithm uses a diamond search point pattern and the algorithm runs exactly the same as 4SS. However, there is no limit on the number of steps that the algorithm can take.

Two different types of fixed patterns are used for search,

- Large Diamond Search Pattern (LDSP)
- Small Diamond Search Pattern (SDSP)

The algorithm runs as follows:

- LDSP:

a. Start with search location at center

b. Set step size 'S' = 2

c. Search 8 locations pixels (X,Y) such that (|X|+|Y|=S) around location (0,0) using a diamond search point pattern

d. Pick among the 9 locations searched, the one with minimum cost function

e. If the minimum weight is found at center for search window, go to SDSP step

f. If the minimum weight is found at one of the 8 locations other than the center, set the new origin to this location

g. Repeat LDSP

- SDSP:

a. Set the new search origin

b. Set the new step size as S = S/2 = 1

c. Repeat the search procedure to find location with least weight

d. Select location with the least weight as motion vector

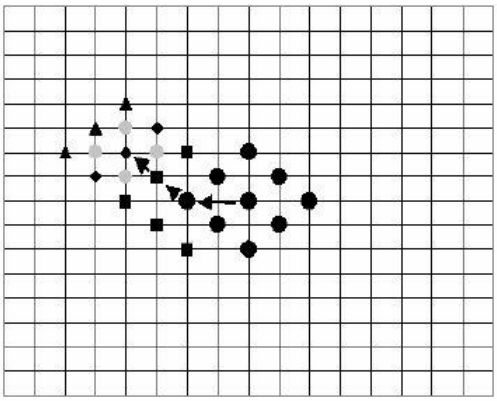

Figure 1.3 Diamond Search procedure

This algorithm finds the global minimum very accurately as the search pattern is neither too big nor too small. Diamond Search algorithm has a peak signal-to-noise ratio close to that of Exhaustive Search with significantly less computational expense.

1.3.2 Orthogonal-Diamond Search

Orthogonal Search algorithm (ODS) employs the orthogonal shape and Small Diamond Search Pattern (SDSP) for its search steps. Although the orthogonal search algorithm (OSA) has only 13 search points per block ME, it uses uniformly allocated search patterns in its first few steps which are inefficient for catching small MV of stationary or quasi-stationary blocks. Therefore, to improve compensation error, center-biased search point pattern is used.

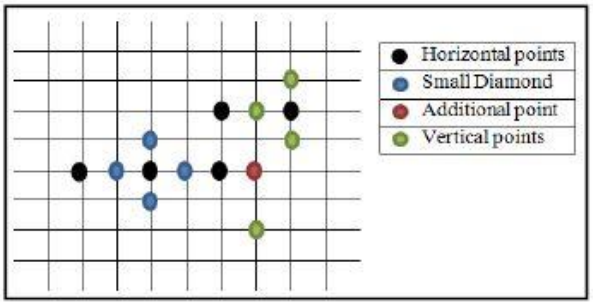

Figure 1.4 The proposed Orthogonal Diamond search pattern

III. DISCUSSION

3.1 BLOCK MATCHING ALGORITHM

BMA is best described as the technique that estimates the amount of motion on a block by block basis. Block matching finds the best block from the previous frame to reconstruct an area of the current frame by dividing each coding frame into non-overlapping blocks with size of N-by-M pixels. The macro blocks are then compared with corresponding block and its adjacent neighbors in the previous frame within a search area of size (N + 2q x M + 2q). The general idea is shown in Fig. 3.1 where q is the maximum displacement allowed, A is the search window in the previous frame, B is the block in the current frame and C is the block in the previous frame.

Motion is defined as a displacement of an object over a period of time and measured in two consecutive frames of a video image. Therefore, motion vector (MV) is best defined as the displacement between the current block and previous block which is found by the best match based on certain matching cost function.

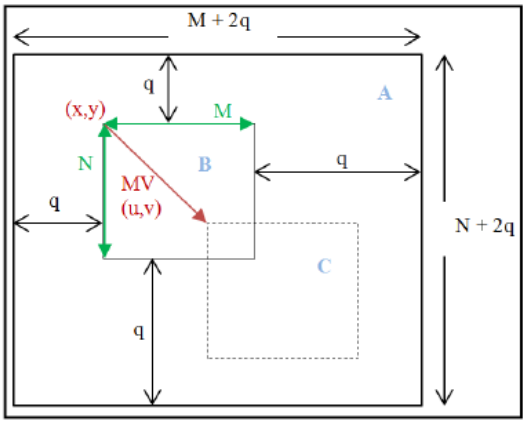

Figure 3.1 Current and previous frames (N x M)

3.1.1 Matching Cost Function

There are several matching cost functions that can be usedfor this purpose, such as mean squared error (MSE), sum of absolute difference (SAD), mean squared difference (MSD) and mean absolute different (MAD). One of the major factors affecting the ME algorithm complexity and its performance is the cost function. MSD and MAD are the two well-known cost functions due to its simplicity where the distortion or the matching error between the block must be minimized to obtain the best match. The functions are briefly explained as follows;

Mean Squared Difference (MSD)

$$MSD(i,j) = \frac{1}{NM}\sum_{k}\sum_{l}[C_f(k,l) - R_{f-1}(k+i,l+j)]^2$$

Mean Absolute Difference (MAD)

$$MAD(i,j) = \frac{1}{NM}\sum_{k}\sum_{l}|C_f(k,l) - R_{f-1}(k+i,l+j)|$$

3.1.2 Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise ratio (PSNR) is used to determine the quality of the compressed images. The higher the PSNR value, the better the quality of the compensated images. The PSNR equation for a gray scale image is defined as follows:

$$\boldsymbol{PSNR} = 10\ log_{10}\left(\frac{(Peak\ to\ peak\ value\ of\ original\ data)^2}{MSD}\right)$$

$$= 10\ log_{10}\left(\frac{255^2}{\frac{1}{NM}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{ij} - \hat{x}_{ij})^2}\right)$$

Where (NM) is the dimension of the frame in pixels while $X_{ij}$and x_{ij} are the luminance components of the original and thereconstructed image, respectively, at the spatial location (i,j).

3.2 MOTION ESTIMATION

Motion estimation is the process of determining motion vectors that describe the transformation from one 2D image to another; usually from adjacent frames in a video sequence. It is an ill-posed problem as the motion is in three dimensions but the images are a projection of the 3D scene onto a 2D plane. The motion vectors may relate to the whole image (global motion estimation) or specific parts, such as rectangular blocks, arbitrary shaped patches or even per pixel. The motion vectors may be represented by a translational model or many other models that can approximate the motion of a real video camera, such as rotation and translation in all three dimensions and zoom. Successive video frames may contain the same objects (still or moving). Motion estimation examines the movement of objects in an image sequence to try to obtain vectors representing the estimated motion. Motion compensation uses the knowledge of object motion so obtained to achieve data compression. In interframe coding, motion estimation and compensation have become powerful techniques to eliminate the temporal redundancy due to high correlation between consecutive frames. In real video scenes, motion can be a complex combination of translation and rotation. Such motion is difficult to estimate and may require large amounts of processing. However, translational motion is easily estimated and has been used successfully for motion compensated coding.

3.2.1Motion compensation

Motion compensation is an algorithmic technique used to predict a frame in a video, given the previous and/or future frames by accounting for motion of the camera and/or objects in the video. It is

employed in the encoding of video data for video compression, for example in the generation of MPEG-2 files. Motion compensation describes a picture in terms of the transformation of a reference picture to the current picture. The reference picture may be previous in time or even from the future. When images can be accurately synthesised from previously transmitted/stored images, the compression efficiency can be improved.

3.2.1.1Motion Compensation in MPEG

In MPEG, images are predicted from previous frames (P frames) or bidirectional from previous and future frames (B frames). B frames are more complex because the image sequence must be transmitted/stored out of order so that the future frame is available to generate the B frames. After predicting frames using motion compensation, the coder finds the error (residual) which is then compressed and transmitted.

3.2.1.2 Block motion compensation

In block motion compensation (BMC), the frames are partitioned in blocks of pixels (e.g. macroblocks of 16×16 pixels in MPEG). Each block is predicted from a block of equal size in the reference frame. The blocks are not transformed in any way apart from being shifted to the position of the predicted block. This shift is represented by a motion vector. To exploit the redundancy between neighboring block vectors, (e.g. for a single moving object covered by multiple blocks) it is common to encode only the difference between the current and previous motion vector in the bit-stream. The result of this differencing process is mathematically equivalent to a global motion compensation capable of panning. Further down the encoding pipeline, an entropy coder will take advantage of the resulting statistical distribution of the motion vectors around the zero vector to reduce the output size.

3.2.2 Mean squared error

In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator measures the average of the squares of the errors or deviations, that is, the difference between the estimator and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. The difference occurs because of randomness or because the estimator doesn't account for information that could produce a more accurate estimate. The MSE is the second moment (about the origin) of the error, and thus incorporates both the variance of the estimator and its bias. For an unbiased estimator, the MSE is the variance of the estimator. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated. In an analogy to standard deviation, taking the square root of MSE yields the root-mean-square error or root-mean-square deviation (RMSE or RMSD), which has the same units as the quantity being estimated; for an unbiased estimator, the RMSE is the square root of the variance, known as the standard deviation.

3.2.3 Sum of Absolute Difference

In digital image processing, the sum of absolute differences (SAD) is a measure of the similarity between image blocks. It is calculated by taking the absolute difference between each pixel in the original block and the corresponding pixel in the block being used for comparison. These differences are summed to create a simple metric of block similarity, the L1 norm of the difference image or Manhattan distance between two image blocks. The sum of absolute differences may be used for a variety of purposes, such as object recognition, the generation of disparity maps for stereo images, and motion estimation for video compression.

It is noted that the add-and-drop and switching functionalities of intermediate BV-WXC result in inter path crosstalk. Furthermore, the filtering effect of BV-WSS causes power penalty for the edge subcarriers of each spectrum. Hence, along a path where intermediate BV-WXCs are cascaded, these penalties are accumulated which lead to OSNR degradation or equivalently TR reduction.

4.1 DIAMOND SEARCH (DS)

DS algorithm is exactly the same as 4SS, but the search point pattern is changed from a square to a diamond, and there is no limit on the number of steps that the algorithm can take. DS uses two different types of fixed patterns, one is Large Diamond Search Pattern (LDSP) and the other is Small Diamond Search Pattern (SDSP). These two patterns and the DS procedure are illustrated. Just like in FSS, the first step uses LDSP and if the least weight is at the center location we jump to fourth step. The consequent steps, except the last step, are also similar and use LDSP, but the number of points where cost function is checked are either 3 or 5 and are illustrated in second and third steps of procedure. The last step uses SDSP around the new search origin and the location with the least weight is the best match. As the search pattern is neither too small nor too big and the fact that there is no limit to the number of steps, this algorithm can find global minimum very accurately. The end result should see a PSNR close to that of ES while computational expense should be significantly less.

4.2 ORTHOGONAL DIAMOND SEARCH

Although the orthogonal search algorithm (OSA) has only 13 search points per block ME, it uses uniformly allocated search patterns in its first few steps which are inefficient for catching small MV of stationary or quasi-stationary blocks. Therefore, to improve compensation error, center-biased search point pattern is used [19]. The proposed ODS algorithm steps are summarized as follows:

Step 1: The first three points initiate the orthogonal search pattern where the center point is centered at the origin of the search window. All points are tested to find the minimum cost function (MCF) point. If the MCFpoint is found at the center, proceed to Step 2.Otherwise, the small diamond search pattern (SDSP) is employed to the center point and one additional point is added and proceeds to Step 2.

Step 2: Two vertical search points with step size of 2 are searched for MCF point and proceed to Step 3.

Step 3: The step size is reduced to 1 point for the two horizontal search points and then proceeds to Step 4.

Step 4: The search pattern and step size is the same as Step 3 but in vertical direction and the MV is found in this step is the final MV.

A . RESULT

The block size is fixed at 16x16 and the block matching is conducted within the 15x15 search window size and the frame distance between predicted frame and original frame is set to be 1 for consistent comparison with previous research works. The maximum displacement allowed is ±7 horizontally and vertically. The matching cost function used is MAD instead of MSD in the procedure as it does not require a multiplication operation, hence reducing the block matching computational requirement.

IV. CONCLUSION

Orthogonal Diamond Search (ODS) algorithm is being proposed for the fast BMA motion estimation. Based on the experimental results, ODS performs better in terms of search points compared to the

DiamondSearchalgorithms for all different types of motion contents. However, in terms of PSNR performance, ODS maintains a close performance when comparing between the other BMAs algorithms.

REFERENCES

1. P, S. Ian, G. Bala and B. George, 'A Study on BlockMatching Algorithms for Motion Estimation.',International Journal on Computer Science & Engineering,vol 3, iss 1, 2011.
2. K. Rijkse, 'H. 263: video coding for low-bit-ratecommunication', Communications Magazine, IEEE, vol 34,iss 12, pp. 42--45, 1996.
3. S. Vetrivel, K. Suba and G. Athisha, 'An overview of h. 26xseries and its applications', International Journal ofEngineering Science and Technology, vol 2, iss 9, pp.4622--4631, 2010.
4. J. de Oliveria, 'A Java H. 263 Decoder Implementation',Electrical and Computer Engineering Department,University of Ottawa, 1997.470
5. T. Koga, 'Motion-compensated interframe coding for videoconferencing', pp. 5--3, 1981.
6. R. Li, B. Zeng and M. Liou, 'A new three-step searchalgorithm for block motion estimation', Circuits andSystems for Video Technology, IEEE Transactions on, vol4, iss 4, pp. 438--442, 1994.
7. L. Po and W. Ma, 'A novel four-step search algorithm forfast block motion estimation', Circuits and Systems forVideo Technology, IEEE Transactions on, vol 6, iss 3, pp.313--317, 1996.
8. S. Zhu and K. Ma, 'A new diamond search algorithm forfast block-matching motion estimation', Image Processing,IEEE Transactions on, vol 9, iss 2, pp. 287--290, 2000.
9. R. Manap, S. Ranjit, A. Basari and B. Ahmad,'Performance Analysis of Hexagon-Diamond SearchAlgorithm for Motion Estimation', Proc. InternationalConf. on Computer Engineering and Technology, vol 3, pp.3--155, 2010.
10. H. Jia and L. Zhang, 'A new cross diamond searchalgorithm for block motion estimation', Proc. IEEEInternational Conf. on Acoustics, Speech and SignalProcessing, vol 3, p. --357, 2004.
11. C. Cheung and L. Po, 'Novel cross-diamond-hexagonalsearch algorithms for fast block motion estimation',Multimedia, IEEE Transactions on, vol 7, iss 1, pp. 16--22,2005.
12. S. Soongsathitanon, W. Woo and S. Dlay, 'Fast searchalgorithms for video coding using orthogonal logarithmicsearch algorithm', Consumer Electronics, IEEETransactions on, vol 51, iss 2, pp. 552--559, 2005.
13. P, S. Ian, G. Bala and J. Anitha, 'Enhanced modifiedorthogonal search for motion estimation', Recent Advancesin Intelligent Computational Systems (RAICS), 2011 IEEE,pp. 907--910, 2011.
14. R. Srinivasan and K. Rao, 'Predictive coding based onefficient motion estimation', Communications, IEEETransactions on, vol 33, iss 8, pp. 888--896, 1985.
15. K. Takaya, 'Detection of Moving Object in Video Scene -MPEG like Motion Vector vs. Optical Flow', in The FirstInternational Workshop on Video Processing for Security(VP4S-06) - Held Jointly with CRV?06, Quebec.
16. S. Metkar and S. Talbar, Motion Estimation Techniques forDigital Video Coding, 1st ed. New Delhi: Springer, 2013,pp. 13-31, 53-59.
17. S. Usama, M. Montaser and O. Ahmed, 'A complexity andquality evaluation of block based motion estimationalgorithms', ActaPolytechnica, vol 45, iss 1, 2005.
18. A. Barjatya, 'Block matching algorithms for motionestimation', IEEE Transactions Evolution Computation, vol8, iss 3, pp. 225--239, 2004.
19. L. Po and C. Cheung, 'A new center-biased orthogonalsearch algorithm for fast block motion estimation', vol 2,pp. 874--877, 1996.