International Journal of Scientific & Engineering Research, Volume 6, Issue 9, September-2015
ISSN 2229-5518

LOSSLESS HUFFMAN CODING FOR IMAGE COMPRESSION USING K-MEANS ALGORITHM WITH MOTION ESTIMATION TECHNIQUE VIA "3-LEVEL HBMA OVER EBMA" ON BLOCK SIZE OF 16 AND 64

Ali Tariq Bhatti[1], Dr. Jung H. Kim[2]
[1,2]Department of Electrical and Computer engineering
[1,2]North Carolina A&T State University, Greensboro NC USA
[1]atbhatti@aggies.ncat.edu, ali_tariq302@hotmail.com, alitariq.researcher.engineer@gmail.com
[2]kim@ncat.edu

Abstract— Now-a-days, image compression techniques are very common in a wide area of researches in terms of lossless and lossy compression. Image compression can also play an impact role in video or motion estimation techniques. Motion estimation (ME) is an essential part of any video processing system of determining motion vectors that are representing the transition or motion of objects in successive video frames as store in images. It is use to find its applications in two main areas: reduction of temporal redundancy in video coders and representation of true motion of objects in real-time video applications. In this research paper, the block size of 16 and 64 has been used to compute for part A and part B, so then the analysis of performance metrics. In part A, read an image of 256X256 in any jpg or gif format. Furthermore, Lossless Huffman using K-Means algorithm of block size of both 16 and 64 and choose codebook size of 50 in part A. In part B, record voice video of 128X128 with at least one moving object in any video format such as .wmv, .avi, etc. file with reasonable brightness and contrast using 30 frames/second and convert those video frames into images by RGB to intensity format, applying median filter by using selected motion estimation algorithm via MATLAB implementation. Then, in this research paper, 3-level HBMA algorithm over EBMA for two second voice video is used. Take the anchor and the tracked frames of 2 second voice video are each represented by a pyramid, and the EBMA or one of its variants is employed to estimate motion vectors of blocks in each level of the pyramid. Different frames of image for 3 different videos like 1st, middle, last frame, etc. are used as a original image and motion vector image during this algorithm. Quiver displays velocity vectors as arrows for motion estimation for the videos in that frame of image in this research paper. Moreover, display the original video, motion vector video, threshold video and their results. Also did analysis to compute 1st frame and last frame of the 2 second original video, and first frame and middle frame of the 2 second motion vector video by RGB to intensity format. Compute MSE between the original and the result. Therefore, other evaluation metrics used such as PSNR and MAD for 3-level HBMA over EBMA of block sizes of 16 and 64 via MATLAB implementation. Furthermore, investigate the analysis of performance metrics for part A and part B to be used in this research paper. Finally, there will be a change in MAD and MSE will be decreased when taking larger block size like of 64 in the context lossless Huffman coding for image compression using K-Means algorithm and Video estimation technique via HBMA over EBMA algorithm.

Index Terms— Huffman, K-Means, Motion Estimation, HBMA, EBMA

1 INTRODUCTION

1.1 Huffman coding:

Huffman coding is a lossless image compression technique applied using K-Means algorithm for 2 different scenarios in this research paper.

1.2 K-Means algorithm:

K-Means Algorithm is the Clustering algorithm that follows a simple way to classify a given data set through a certain number of clusters. The main idea behind K-Means Algorithm is to define 'K' centroids in K-Means algorithm, one for each cluster. These centroids should be placed in the best way,so they are much as possible far away from each other. One of the disadvantages of K-Means Algorithm is to ignore measurement errors, or uncertainty, associated with the data and it is also known as Error based Clustering.

Quantization is the process of limiting real numbers to discrete integer values. Vector quantization is a lossy compression technique based on block coding. It maps a vector to a codeword drawn from a predesigned codebook with the goal of minimizing distortion. K-Means is an unsupervised machine learning technique. The basic idea of the K-means cluster is to place N data points in an l-dimensional space into K clusters.

1.3 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. In order to evaluate the performance of video compression coding [9], in motion estimation, it is necessary to define a measure to compare the original video and the video after compressed.

The output of the motion-estimation algorithm comprises the motion vector for each block, and the pixel value differences between the blocks in the current frame and the "matched" blocks in the reference frame.

The major aspects of motion estimation techniques are to (i) reduced computational complexity (ii) Good visual quality in terms of representation of true motion (iii) High compression

ratio. In contrast, computational complexity of a motion estimation technique can be determined by three factors which are as:
(a) Search algorithm that decides the overall computational complexity and accuracy of motion estimation
(b) Search area is a area that the span of search window to find the best match
(c) Cost function is the function for different cost metrics to find the best match.

2 Block Diagram of 3-level HBMA

2.1 Main block diagram

This is the main block diagram used for block size of 16 and 64 to compute Part A and Part B and then the analysis of performance metrics in this research. However, Part A and Part B have been split with its own block diagram to do image block size of 16 and 64 in figure 1.

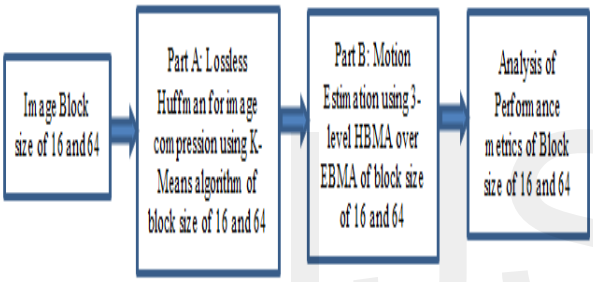


Figure 1: Main block diagram

2.2 Part A block diagram

(a) In Part (A) block diagram, read an image size of 256X256 in jpg or gif format.
(b) Then, choose the block size either 16 or 64 and also choose codebook size of 50 in one of the two scenarios.
(c) K-Means algorithm is applied based on block and codebook size for Huffman coding. Follow, steps of K-Means algorithms and the process of Huffman coding will be implemented using this algorithm.
(d) Lastly, compute the performance metrics as in figure 2.

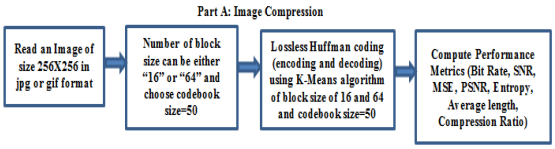


Figure 2 Part (A) Block Diagram

2.3 3-level HBMA Block Diagram explanation

- We acquire and record two second voice video with at least one moving object in any video format such as .wmv, .avi, etc file with reasonable brightness and contrast using 30 frames/second. (In this research paper, .avi format is used).
- Then, read this 2 second acquired video using "MATLAB".
- Convert the image from RGB to intensity format.
  - hcsc=vision.ColorSpaceConverter;
  - hcsc.Conversion = 'RGB to intensity';
  - image1 = step(hcsc, imresize(frame2im(mov1(1,kk)),[128 128]));
- Estimate the direction and speed of object motion from one video frame to another using 3-level Hierarchical Block Matching Algorithm (HBMA).
- Calculate the overall and running mean of the velocities.
- Create a 2-D median filter and filter out slight speckle noise
  - image1 = medfilt2(step(hcsc, imresize(frame2im(mov1(1,kk)),[128 128])));
- Appling 3 level hierarchical block matching algorithm, the anchor and the tracked frames are each represented by a pyramid, and the EBMA or one of its variants is employed to estimate motion vectors of blocks in each level of the pyramid.
- Display the original video, motion vector video, threshold video and the results
- Compute MSE between the original and the result.
- Using other evaluation metrics such as PSNR and MAD for 3-level HBMA.

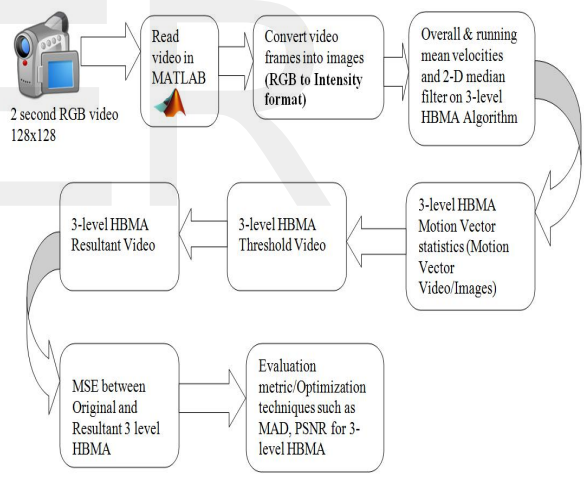


Figure 3 Part (B) Block Diagram

3 METHODOLOGY AND SELECTION OF DESIGN FACTORS

The methodology and design factors are discussed in two parts such as Part A and Part B based on main block diagram in figure 1.

3.1 Part A

3.1.1 Huffman coding using K-Means algorithm based on block size and codebook

Step 1: Read my image with it equal dimensions such as 256X256 in this research.

Step 2: Initialization of the size of block 'M' and size of codebook 'N' for two scenarios (block size of 16 and 64 and choose codebook size of 50).
Step 3: Huffman coding to quantizing K- Mean clustering for an image
There are 4 cases to use K-Mean Algorithm, which are as:
(a) Initialize a set of training vectors with any variable as 'X' and we need a codebook of size N as in this case.
(b) Second case is to randomly choose M dimensional or block vectors as the initial set of code words in the codebook.
(c) Third case is to search for nearest neighbor for each training vector. This will allow finding the codeword in the current codebook which seems to be closest in terms of spectral distance and assign that vector to the corresponding cell.
(d) Finally update the Centroid for the code word in each cell using the training vectors assigned to that cell. In this case 4, repeat case second and third again and again until the procedure converges or Average distance falls below a preset threshold.

3.1.2 The Quad tree decomposition

This approach divides a square image into four equal sized square blocks, and then tests each block to see if meets some criterion of homogeneity. If a block size meets the criterion it is not divided any further, and the test criterion is applied to those blocks. This process is repeated iteratively until each block meets the criterion.

3.1.3 Huffman Encoding

The Huffman encoding starts by constructing a list of all the alphabet symbols in descending order of their probabilities. It then constructs, from the bottom up, a binary tree with a symbol at every leaf. This is done in steps, where at each step two symbols with the smallest probabilities are selected, added to the top of the partial tree, deleted from the list, and replaced with an auxiliary symbol representing the two original symbols [22]. When the list is reduced to just one auxiliary symbol (representing the entire alphabet), the tree is complete. The tree is then traversed to determine the code words of the symbols.

3.1.4 Huffman Decoding

Before starting the compression of a data file, the encoder has to determine the codes. It does that based on the probabilities of frequencies of occurrence of the symbols. The probabilities or frequencies have to be written, as side information, on the output, so that any Huffman decoder will be able to decompress the data. This is easy, because the frequencies are integers and the probabilities can be written as scaled integers. It normally adds just a few hundred bytes to the output. It is also possible to write the variable-length codes themselves on the output, but this may be awkward, because the codes have different sizes. It is also possible to write the Huffman tree on the output [21], but this may require more space than just the frequencies. In any case, the decoder must know what is at the start of the compressed file, read it, and construct the Huffman tree for the alphabet. Only then can it read and decode the rest of its input. The algorithm for decoding is simple. Start at the root and read the first bit off the input (the compressed file). If it is zero, follow the bottom edge of the tree; if it is one, follow the top edge. Read the next bit and move another edge toward the leaves of the tree. When the decoder arrives at a leaf, it finds there the original, uncompressed symbol, and that code is emitted by the decoder. The process starts again at the root with the next bit.

3.2 Part B

3.2.1 Motion Estimation Algorithms:

The algorithms for finding motion vectors can be categorized into two methods such as pixel based methods known as "Direct" and feature based methods known as "Indirect". The direct methods are the following:

- Block Matching Algorithms (BMA):- One of the famous techniques for motion estimation is the Block Matching Algorithm (BMA). Block-matching motion estimation is an important part for any motion compensated video coding standards such as ISO/IEC MPEG and ITUT [15]. It reduces the temporal redundancy, which is found predominantly in any video sequence. The frame is divided into non-overlapped macro blocks (MBs) of size N×N that are then compared with corresponding macro block (MB) and its adjacent neighbors in the reference frame to create a vector that stipulates the movement of a macro block from one location to another in the reference frame [16], i.e. finding matching macro block of the same size N×N in the search area in the reference frame. It is a way of locating matching blocks in a sequence of digital video frames for the purposes of motion estimation. The goal of a block matching algorithm is to find a matching block from a frame i in some other frame j, which may appear before or after i. However, it can be used to discover temporal redundancy in the video sequence, increasing the effectiveness of inter frame video compression and television standards conversion. Hence, block matching algorithms make use of criteria to determine whether if a given block in frame j matches the search block in frame i.

(a)Phase correlation and Frequency domain methods.
(b)Pixel recursive algorithms

Recently, some search algorithms were proposed for fast motion estimation based on the assumption that most of the objects' motions are zero or relatively small [6-7]. Moreover, sequences of ordered images allow the estimation of motion as either instantaneous image velocities or discrete image displacements. Fortunately, this method tries to calculate the motion between two image frames which are taken at times *t* and $t+\Delta t$ at every voxel position. In fact, for a 2D dimensional case (3D or *n*-D cases are similar) a voxel at location (x,y,t) with intensity $I(x,y,t)$ will have moved by Δx, Δy, and Δt between the two image frames, and the following Image Constraint equation can be given as

$$I(x,y,t) = I(x+\Delta x, y+\Delta y, t+\Delta t) \quad (1)$$

If we assume the movement to be small, the image constraint at $I(x,y,t)$ with Taylor series can be developed to get:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \quad (2)$$

From these two equations it follows that:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y - \frac{\partial I}{\partial t}\Delta t = 0$$

Or

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} - \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \quad (3)$$

results in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \quad (4)$$

From equation (4), where V_x, V_y are the x and y components of the velocity or optical flow of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t) in the corresponding directions. I_x, I_y, and I_t can be written for the derivatives in the following way. Thus:

$$I_x V_x + I_y V_y = -I_t$$

Or

$$\nabla I^T \cdot \vec{V} - -I_t \quad (5)$$

Equation 5 is in two unknowns and cannot be solved as such. This phenomenon also called as Aperture Problem of the optical flow algorithms. Hence, finding the optical flow, another set of equations is needed which is to be given by some additional constraint.

Motion compensation prediction assumes that the current frame can be locally modeled as a translation of the frames in the previous time. In order to get motion compensation, the motion of the moving objects has to be estimated first is called as Motion Estimation (ME). ME module is usually the most computationally intensive part (50-80% of the entire system) in a video encoder [12-13]. During block matching, each target block of the current frame is compared with a previous frame in order to find the best matching block. Block-matching algorithms calculate the best match using the Mean Absolute Difference (MAD) or Sum of Absolute Differences (SAD) [4]. For direct methods several evaluation metrics or optimization techniques are used.

- Mean squared error (MSE)
- Sum of absolute differences (SAD)
- Mean absolute difference (MAD)
- Sum of squared errors (SSE)
- Sum of absolute transformed differences (SATD)
- Peak Signal to Noise Ratio (PSNR)

However, in this research paper, MSE, MAD, and PSNR is used as evaluation metric.

3.2.2 Exhaustive Block Matching Algorithm:

In this research paper, Hierarchical block matching algorithm is used over Exhaustive block matching algorithm, where the goal of motion estimation (ME) is to minimize the total bits used for coding the motion vectors (MVs) and the prediction errors, however; producing a motion-compensated prediction of a frame which is coded from a previous reference frame. All ME algorithms are based on temporal changes related to image intensities. At time t a point in an image (x, y) is moved to ($x + d_x$, $y + d_y$) at time $t + d_t$,. Therefore equation (6) is as

$$\psi(x+d_x, y+d_y, t+d_t) = \psi(x,y,t) \quad (6)$$

If we suppose that d_t is small, the equation 6 changes to

$$\frac{\partial \psi}{\partial x}v_x + \frac{\partial \psi}{\partial y}v_y + \frac{\partial \psi}{\partial t} = 0 \quad (7)$$

where (v_x, v_y) represents the velocity vector. When applying ME between two frames, $\psi(x, y, t_1)$ is considered to be the Anchor frame (Reference frame) and $\psi(x, y, t_2)$ is considered to be the Target frame shown in figure 6. The anchor frame can be either before or after the target frame in time but when $t_1 < t_2$, the problem is referred to as forward ME (which is implemented in this research) and when $t_1 > t_2$, it is referred to as backward ME.

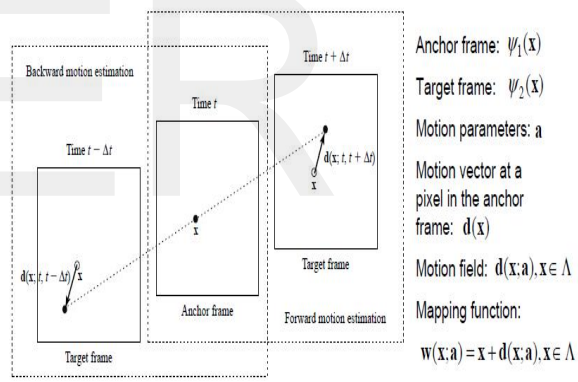


Figure 4: Illustration of backward and forward ME

To reduce complexity of ME, a fixed partition of the image domain is made into many small blocks. The motion variation within each block can be characterized by a simple model and each block's motion parameters can be independently estimated. This is known as block-based algorithm. The most popular criterion for Block-Matching Algorithm (BMA) is the sum of the differences between luminance values of every pair of corresponding points between the anchor frame ψ_1 and the target frame ψ_2. The goal is to minimize the displaced frame difference (DFD) error and this is done using the mean absolute difference (MAD) equation such as

$$E_{DFD}(a) = \sum_{\mathbf{x}\in\Lambda} \left|\psi_2(\mathbf{w}(\mathbf{x};\mathbf{a})) - \psi_1(\mathbf{x})\right| \quad \textbf{(8)}$$

" Λ " is the set of all pixels in ψ_1. When given a block in the anchor frame, the goal is to determine a matching block in the

target frame which minimizes the DFD error. The motion vector of a block is the displacement vector d_m between the spatial positions of the two blocks that changes from equation (8) to

$$E_m(\mathbf{d}_m) = \sum_{\mathbf{x} \in B_m} \left| \psi_2(\mathbf{x} + \mathbf{d}_m) - \psi_1(\mathbf{x}) \right| \quad \textbf{(9)}$$

where B_m is a block in the anchor frame. In this research paper, the Exhaustive BMA (EBMA) for three levels HBMA uses an exhaustive search to determine the d_m that minimizes this error. The search region of EBMA is usually symmetric with respect to the current block and the estimation accuracy is determined by the search step size. In our case, the step size is one pixel which is known as integer-pel accuracy search. When using integer-pel EBMA, it is noted that the overall computational load is independent of the block size and the total number of operations for a complete frame is $M^2(2R+1)^2$ where M and R are the image size (when image size is square) and search range respectively. Finally, I also analyzed using EBMA for my 3 level HBMA algorithm that this procedure is computationally expensive for larger images. An image may include fast-motion and slow-motion objects simultaneously. Recently, some search algorithms were proposed for fast motion estimation based on the assumption that most of the objects' motions are zero or relatively small [6][7].

Given a macro block in the anchor block B_m, the motion estimation is to determine a matching block B_m in the target frame such that the error D(s, t) between the two blocks is minimizes. The most straight forward method is the exhaustive block-matching algorithm (EBMA). The procedure of EBMA [5] is shown in figure 5; we can know that how EBMA search procedure applied for my 3 level HBMA research paper.

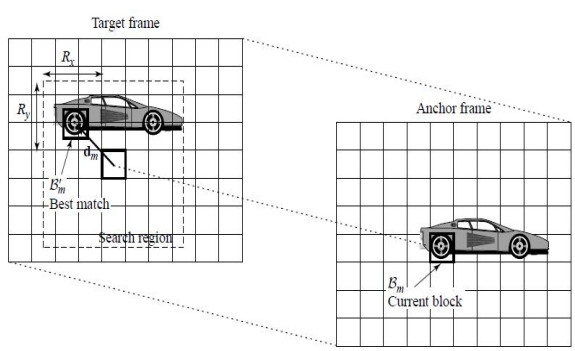


Figure 5: EBMA search procedure

3.2.2.1 Pros and Cons of EBMA:

(1)Blocking effect (discontinuity across block boundary) in the predicted image because the block-wise translation model is not accurate

(2) In EBMA, motion field somewhat chaotic because MVs are estimated independently from block to block. To fix, Mesh-based motion estimation to be used and imposing smoothness constraint explicitly

(3) Wrong MV in the flat region because motion is indeterminate when spatial gradient is near zero.

3.2.3 Hierarchical Block Matching Algorithm (HBMA):

Hierarchical estimation of the motion vector field also known as pyramid search is a widely applied approach to motion estimation. It offers low computational complexity and high efficiency, plus a large degree of flexibility in the trade-off between the two. The hierarchical algorithm is a newer, both fast and efficient approach. Methods in this category explore the correlation between different resolution levels of representation of the same frame. In hierarchical methods, the frame size at different resolution levels is identical to each other, while the MB size varies, with lower level having larger MB size. It is assumed that larger MB's MV in the lower level can be used as a good prediction of the MVs of the smaller MBs (covered by the corresponding larger MB) in the higher level. However, this assumption often mismatches the actual situation, and consequently could lead to poor performance [18]. The idea of HBMA or Multi-Resolution scheme is based on predicting an initial estimate at the coarse level and refining the estimate at the fine level. Usually two- or three-level hierarchical search is adopted. The search range at the fine level is much smaller than the original search range. More levels can save more computation, but the probability of being trapped in local minimum is higher because when the image is scaled down, the detailed textures will be lost. In fact, the Multi-Resolution technique has been regarded as one of the most efficient methods in BMA and is mostly adopted in applications with very large frames and search areas [17, 18]. However, since the characteristics of such a block generally depend on many factors, it is very difficult to model the blocks accurately using analytic methods [11]. In hierarchical estimation, both frames undergo a process of size and resolution reduction. Several levels are constructed, each containing the same image as the previous level, having both dimensions reduced by a certain factor (usually 2). The result is a pyramid, where the lowest level is the initial image, and each level above it is the same image at ¼ of its size as shown in figure 6.

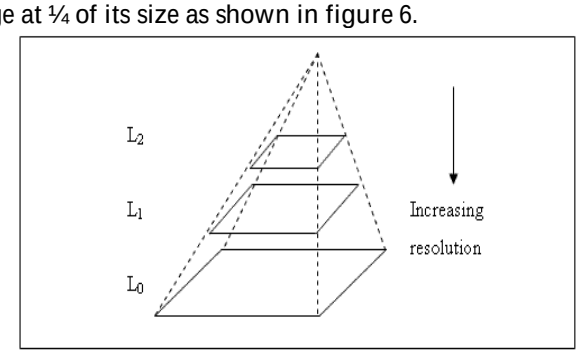


Figure 6: HBMA (pyramid) structure of an image

In order to create a lower resolution image from the initial one, two approaches can be used such as the mean intensity or sub-sampling.

$$g_L(p,q) = \left\lfloor \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} g_{L-1}(2p+u, 2q+v) \right\rfloor, 1 \le L \le 2 \quad (10)$$

where $g_L(p,q)$ is the pixel intensity of pixel (p,q) of the Lth level, and $g_0(p,q)$ denotes pixel (p,q) of the original image.

For the mean intensity approach, each block of (usually) four pixels is replaced by one, having their mean intensity. Sub-sampling is another popular approach for reducing an image's size. During sub-sampling, each block of pixels is replaced by only one of them (e.g. the upper leftmost). This approach yields lower quality results than the mean intensity method, but is significantly faster.

After the pyramid has been created for both images, the corresponding higher-level block is located, for each 0-level block of the first frame. A full search then takes place in the higher level of the second frame. This means that a search window is defined in the second frame, and, for each block in the first frame all candidate motion vectors are evaluated. This is achieved by comparing all the blocks in the search window to the block in the first frame whose vector is sought. Traditional regional matching algorithm cannot get satisfactory results in region of low texture [10] because the object function has multi-minimum in this region. Even some optimal methods such as adaptive windows, multiple windows and variable windows were proposed, it is still not practical because the calculation and selection of the window is too time-consuming. So the "foreground fattening" phenomenon [8] in the traditional local matching methods is avoided even that the window size is considerable large.

A common problem that arises in hierarchical search is that an erroneous match in the higher level is almost always propagated to the lower levels, and is bound to result in an erroneous motion vector. This is not a rare case at all, since the low resolution of the highest level often leads to ambiguity. That is, a number of candidate blocks will have similar Block Distance Measure (BDM) values, and the one being actually matched will be slightly better than the others in the higher level, but not necessarily better at level 0. To prevent this, it is possible to propagate more than one vector down to level 1. If a number of blocks with similar BDMs appear at the highest level, all of them can be refined for every lower level and the best one selected among them. The increase in computational cost is small compared to the increase in the quality of the results.

3.2.3.1 HBMA Implementation:

Using EBMA to derive motion vectors requires of an extremely large computations. In addition, the estimated block motion vectors often lead to a chaotic motion field. In 3 level hierarchical block matching algorithm research paper, the anchor and the tracked frames are each represented by a pyramid, and the EBMA or one of its variants is employed to estimate motion vectors of blocks in each level of the pyramid. Figure 8 as going from level2 pass through the low pass filter and down sampling by factor 2 and then do motion estimation. However, it searches for motion vector of block successfully in level0. By looking figure 6, it illustrates the process when the spatial resolution is reduced by half both horizontally and vertically at each increasing level of pyramid. In fact, we assume that the same block size is used at different levels, so that the number of blocks is reduced by half in each dimension as well. HBMA implementation steps can be explained from the below figure 6 and figure 7:

- Figure 6 shows two video frames of size 32 X 32, in which a gray block in the anchor frame is moved by a displacement of (13,11).
- The block size used at each level is 4 X 4, and a search step size is 1 pixel.
- Starting from level 1, for block (0,0), the motion vector is found to be $d_{1,0,0} = d_1 = (3,3)$.
- When going to level 2, for block (0,1), it is initially assigned the motion vector $d_{2,0,1} = u(d_{1,0,0}) = 2d_1 = (6,6)$.
- Starting with this initial motion vector, the correction vector is found to be $q_2 = (1,-1)$, leading to the final estimated MV $(d_{2,0,1}) = d_2 = (7,5)$.
- Finally at level 3, block (1,2) is initially assigned a motion vector of $d_{3,1,2}=u(d_{2,0,1})=2d_2=(14,10)$.
- With a correction vector of $q_3 = (-1,1)$, the final estimate is $d_{3,1,2} = d_3 = (13,11)$.

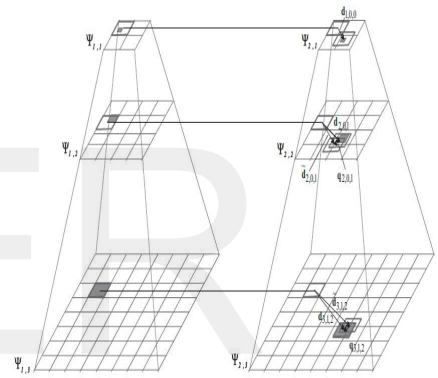


Figure 7: Illustration of Hierarchical Block Matching Algorithm

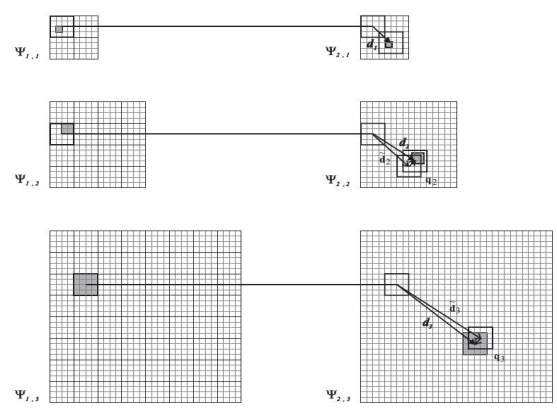


Figure 8: Example of using a 3 - level HBMA block based motion estimation

The number of operations involved in the HBMA depends on the search range at each level. If the desired search range is R in the finest resolution, with a L- level pyramid, and frame size of M X M, then the number of operations required is approximately,

$$1/3*4^{-(L-2)}*4M^2R^2 \quad . (11)$$

3.2.3.2 Application:

(1) The hierarchical motion field estimation offers high quality results, at a very low computational cost, and the large number of parameters to be specified, (block size, number of levels, scaling factor).
(2) Many variations make it one of the most widely used algorithm for a wide range of applications besides motion estimation, including video coding and stereo vision.

3.2.3.3 Computational Requirement of HBMA

The computation requirement of HBMA is based on such as
(1)Assumption such as image size of MXM, block size of NXN at every level "L" and search range.
(2) Operation counts for EBMA of image size "M", block size "N" and search range "R" and # of operations such as $M^2=(2R+1)^2$
(3) Saving factors such as $3*4^{(L-2)}=3$ when L=2; 12 when (L=3)

3.2.4 Quiver

A quiver plot displays velocity vectors as arrows with components (u,v) at the points (x,y). For example, the first vector is defined by components u(1),v(1) and is displayed at the point x(1),y(1). In MATLAB, quiver(x,y,u,v) plots vectors as arrows at the coordinates specified in each corresponding pair of elements in x and y.

3.3 Performance Metrics for lossless Huffman coding using K-Means algorithm

There are following performance metrics used for image compression of original and reconstructed image such as

(a) Bit Rate:
Bit Rate is defined as

$$\text{Bit Rate} = \frac{\text{Number of bits to index a vector}}{\text{Number of samples in a vector}}$$

$$\text{Bit Rate} = \frac{\log_2 N}{(M*M)} \quad (12)$$

The units for Bit Rate is bits/pixel.

(b) Compression Ratio:
Compression Ratio is defined as:

$$\text{Compression Ratio} = \frac{\text{Original Bit Rate}}{\text{New Bit Rate}} \quad (13)$$

Compression Ratio is Unit-less.

(c) SNR:
SNR (Signal-To-Noise Ratio) is defined as

$$\text{SNR} = \frac{10\log_{10}*(\sum^{n}_{i=0} X_i^2)}{(\sum^{n}_{i=0} (Y_i - X_i)^2)} \quad (14)$$

(d) MSE:
The Mean Square Error (MSE) is the error metric used to compare image quality. The MSE represents the cumulative squared error between the reconstructed(Y_i) and the original image(X_i). MSE is a risk function corresponding to the expected value of the squared error loss or quadratic loss. Mean Square Error (MSE) is given by

$$\text{MSE} = \frac{\text{sum}(\text{sum}((Y_i - X_i)^2))}{(M * N)} \quad (15)$$

(e) PSNR
Peak Signal-to-Noise Ratio short as PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its MSE representation. PSNR is usually expressed in terms of the logarithmic decibel scale.

$$\text{PSNR} = 10*\log_{10}\frac{(256^2)}{\text{MSE}} \quad (16)$$

(f) Entropy
Entropy can be defined as the average number of binary symbols needed to encode the output of the source. So, entropy is

$$H(\mathcal{S}) = -\sum p_i \log_2 p_i, \quad (17)$$

(g)Average Length
Average Length is the summation of each probability multiplied by number of bits in the code-word. The code-word for each symbol is obtained traversing the binary tree from its root to the leaf corresponding to the symbol. Symbols with the highest frequencies end up at the top of the tree, and result in the shortest codes [23]. The average length of the code is given by the average of the product of probability of the symbol and number of bits used to encode it. More information can found in [24] [25].

$$\text{Average length}=L= \sum P(a_i)\, n(a_i) \quad (18)$$

4 Experimental Procedure and Results:

Huffman coding using K-Means algorithm in order to compute block size of 16 and 64 in this section. The original image is as

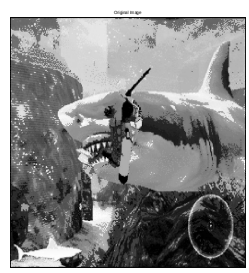


Figure 9: Original image

Size of Block: 16 and Size of Codebook: 50

Figure 10: Compressed image

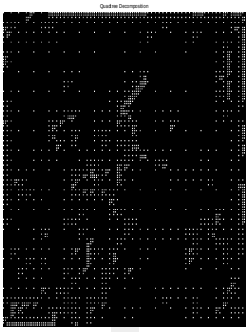


Figure 11: Quadtree Decomposition

Figure 12: Decompressed image

Size of Block: 64 and Size of Codebook: 50

Figure 13: Compressed image

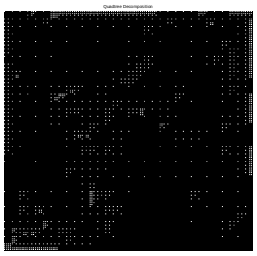


Figure 14: Quadtree Decomposition

Figure 15: Decompressed image

The videos used in this research paper are shark.avi, car.avi, and wildlife2.avi. We can choose different levels of hierarchy. In this method, the precision of our estimation is increased in a hierarchical manner by first estimating the motion vectors for the lowest precision and in the next hierarchical level improve our estimation by computation of the motion vectors with the precision of this level and adding the estimated motion vectors to the vectors obtained from the last level. At the decoder, the received motion vector and the residues will be utilized to find the corresponding block from figures A-F and uses the residues to reconstruct the block [19][20]. From the highest level we make our computations with half-pel accuracy in order to increase the precision. For shark.avi and car.avi, there are a total of 61 frames in two second video. The experiment is done for both shark.avi and car.avi. For wildlife2.avi, there are 69 frames approximately to

two seconds. Here is my display of my Shark,avi 2 second video in figure 16.

(a) Shark.avi video

Figure 16: Shark.avi video for 2 second

Applying 3-level HBMA algorithm for shark.avi video passing through the low pass filter and down-sampling by factor 2, and then applying motion estimation block of vectors using EBMA. At the 3 level, it find its search as shown in figure 17.

Figure 17: HBMA for shark.avi when No of blocks=8

HBMA motion estimation with R = 1, No of blocks = 8, Levels = 3, (A) - (C) - Anchor frame (D) - (F) Target Frame as shown in figure 17 for shark.avi video

Figure 18 shows the Target frame and the estimated frame. The difference/error between these two frames to be shown; when number of blocks is 8.

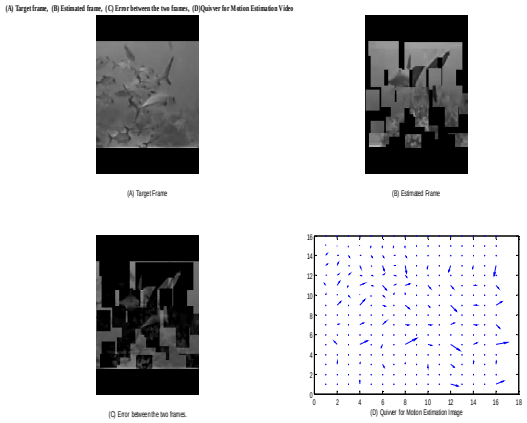


Figure 18: Target and Estimated frame when No of blocks=8

HBMA motion estimation with R = 1, Nb = 16, Levels = 3, (A) - (C) - Anchor frame (D) - (F) Target Frame as shown in figure 19.

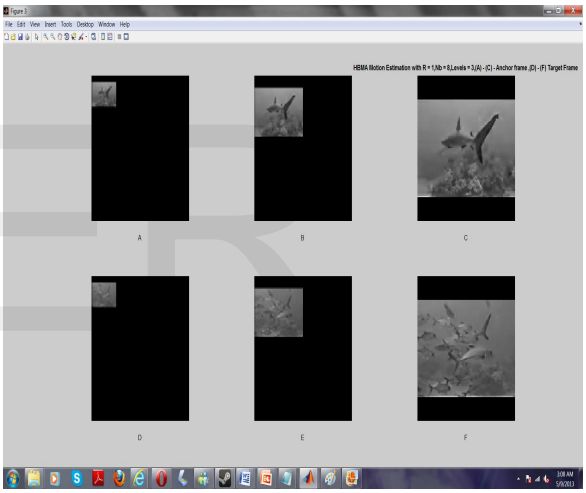


Figure 19: HBMA for shark.avi when No of blocks=16

Figure 20 shows the target frame, estimated frame and the error for a block size of 16.

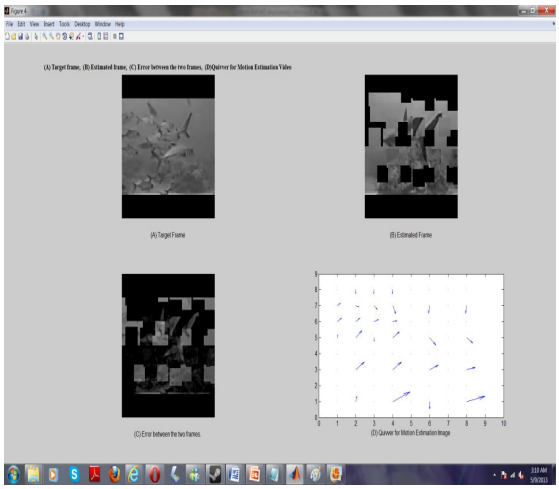


Figure 20: Target and Estimated frame when No of blocks=16

HBMA motion estimation with R = 1, Nb =64, Levels = 3, (A) - (C) - Anchor frame (D) - (F) Target Frame as shown in figure 21.

Figure 21: HBMA for shark.avi when No of blocks=64

Figure 22 shows the target frame, estimated frame and the error for a block size of 64.

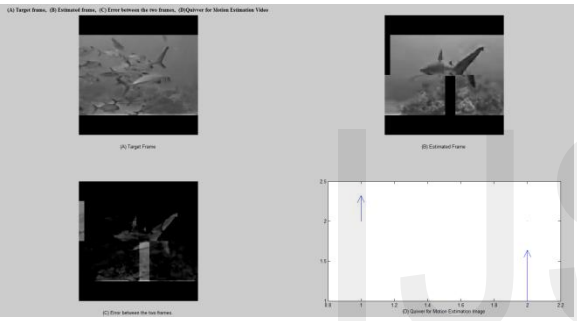


Figure 22: Target and Estimated frame when No of blocks=64

2nd video such as car.avi

HBMA motion estimation with R = 1, No of blocks = 8, Levels = 3, (A) - (C) - Anchor frame (D) - (F) Target Frame as shown in figure 23 for car.avi video

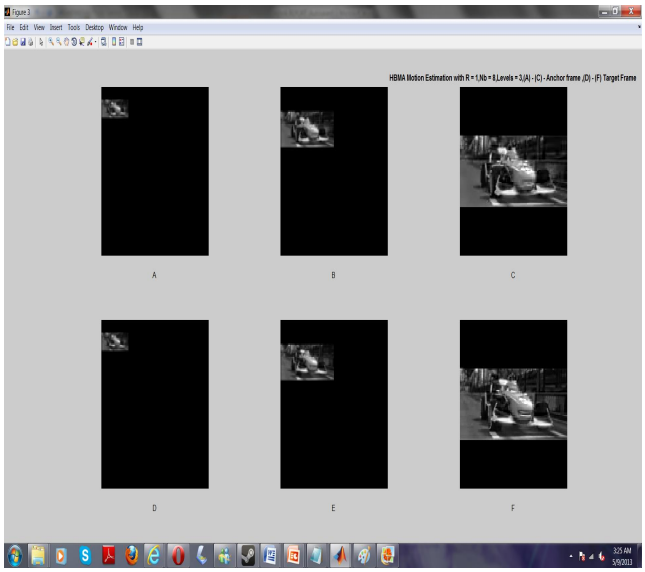


Figure 23: HBMA for car.avi when No of blocks=8

Figure 24 shows the Target frame and the estimated frame. The difference/error between these two frames are shown when number of blocks is 8.

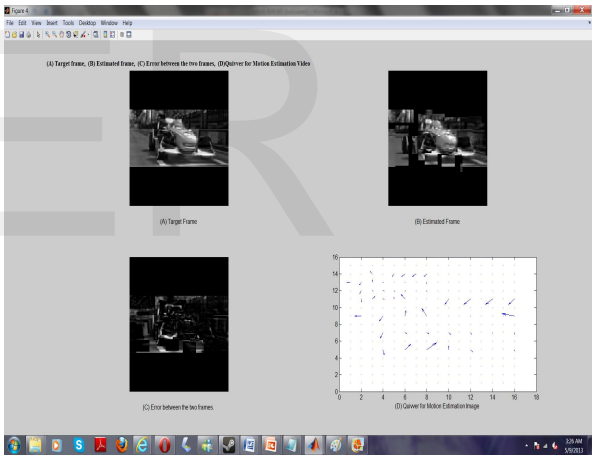


Figure 24: Target and Estimated frame when No of blocks=8

HBMA motion estimation with R = 1, No of blocks = 16, Levels = 3, (A) - (C) - Anchor frame (D) - (F) Target Frame as shown in figure 25 for car.avi video

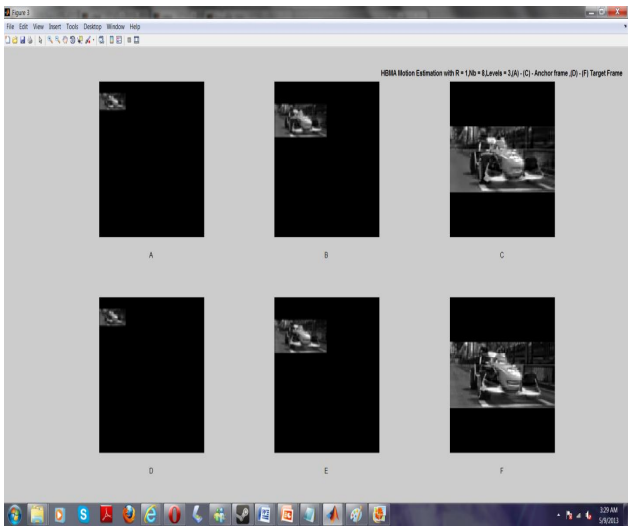


Figure 25: HBMA for car.avi when No of blocks=16

Figure 26 shows the Target frame and the estimated frame. The difference/ error between these two frames are also shown when number of blocks is 16.

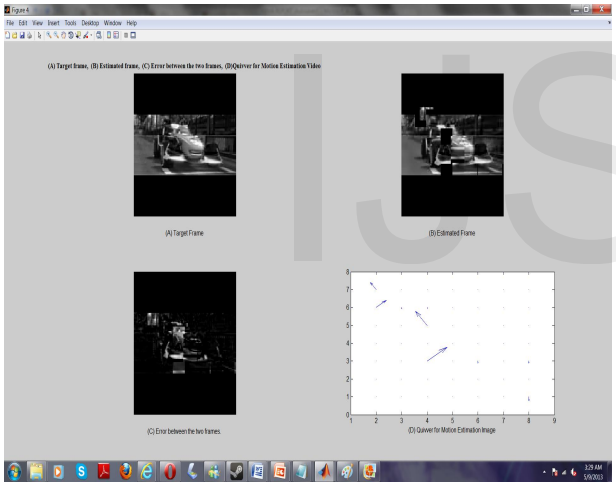


Figure 26: Target and Estimated frame when No of blocks=16

<u>Wildlife.avi 20th frame motion vector is as</u>

Figure 27 shows the motion vectors in blue arrow for the object moving in the motion vector video for 20th frame.

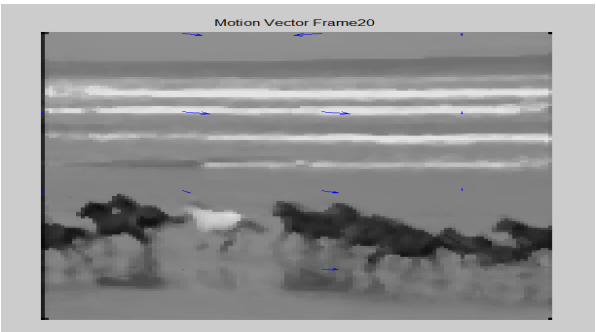


Figure 27: Wildlife2.avi motion vector for 20th frame

Figure 28 shows the original video for 20th frame.

Figure 28: Wildlife2.avi original video frame 20

Figure 29 shows the original video for 30th frame.

Figure 29: Wildlife2.avi original video frame 30

5 Technical Analysis

5.1 1st and last frame for original image and 1st and middle frame for motion vector image of 2 different 2 second videos(shark.avi and car.avi)

<u>(a)shark.avi video for 2 second</u>

Figure 30 and Figure 31: RGB to Intensity format for 1st and last frame (61) of Original video

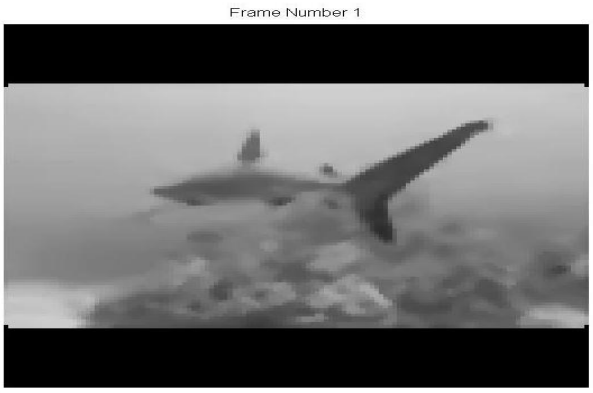


Figure 30: RGB to Intensity format for 1st frame of Original video

Figure 31: RGB to Intensity format for last frame (61) of Original video

Figure 32 and Figure 33: for 1st and middle frames (31) of blue arrows of object moving in motion vector video

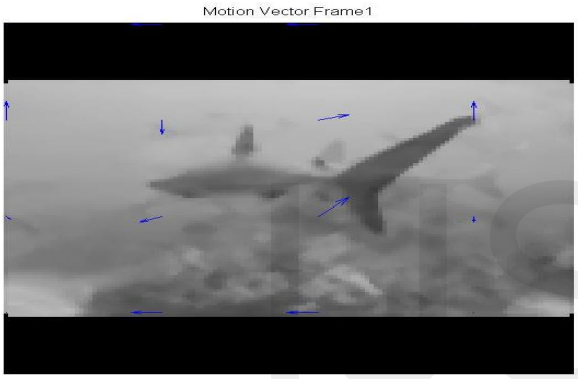


Figure 32: 1st frame of motion vector video

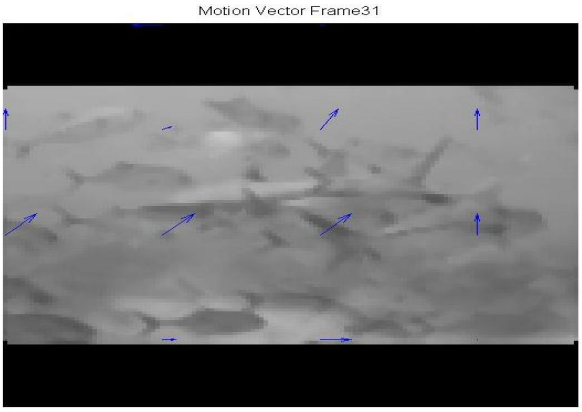


Figure 33: Middle frames (31) of motion vector video

(b)car.avi for 2 second video

Figure 34 and Figure 35: RGB to Intensity format for 1st and last frame (61) of Original video

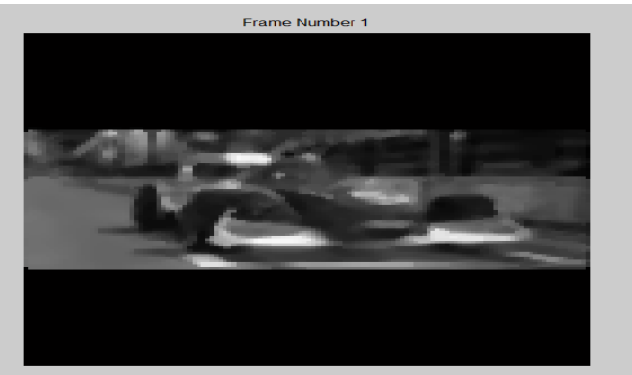


Figure 34: RGB to Intensity format for 1st frame of Original video

Figure 35: RGB to Intensity format for last frame(61) of Original video

Figure 36 and Figure 37: 1st and middle frame (31) of blue arrows of object moving motion vector video for car.avi

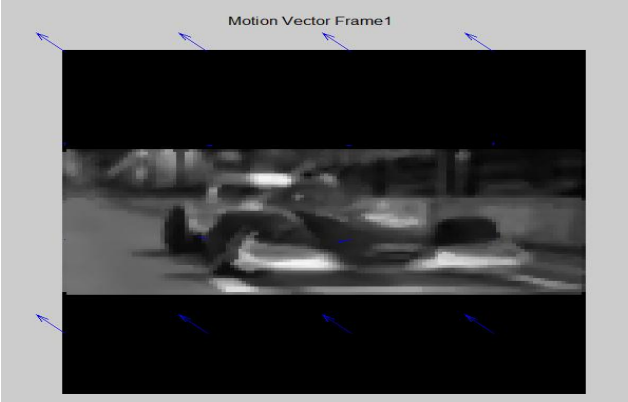


Figure 36: 1st frame of motion vector video for car.avi

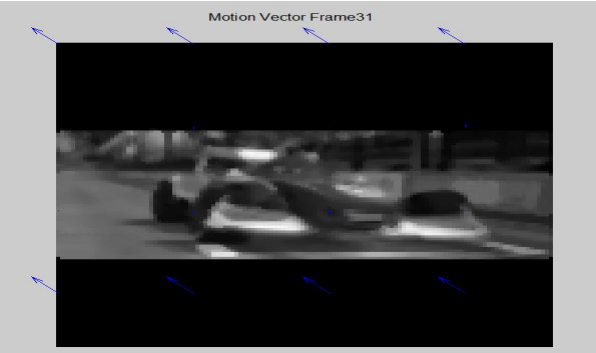


Figure 37: Middle frame(31) of motion vector video for car.avi

5.2 Original video, Motion vector video, Threshold video and Resultant for 3 different 2 second videos

Figure 38, 39, 40, and 41 for shark.avi original, motion vector, threshold and resultant video.

Figure 38 shows the Original Video image for shark.avi using the HBMA over EBMA.

Figure 38: Original Video image

In figure 39, "Yellow" dot indicates the object moving in the motion vector video image.

Figure 39: Motion Vector Video image

In figure 40, shows the blacks spot as a threshold that tells how much threshold it receiving from motion vector video of figure 39.

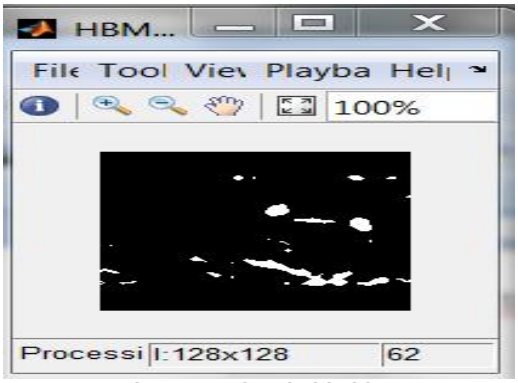


Figure 40: Threshold Video

Figure 41 shows the resultant video image of 0(zero) objects moving in it.

Figure 41: Resultant Video

Figure 42, 43, 44, and 45: Original, motion vector, threshold and resultant video for car.avi

Figure 42 shows the Original Video image for car.avi using the HBMA over EBMA and in figure 43, "Yellow" dot indicates the object moving in the motion vector video image

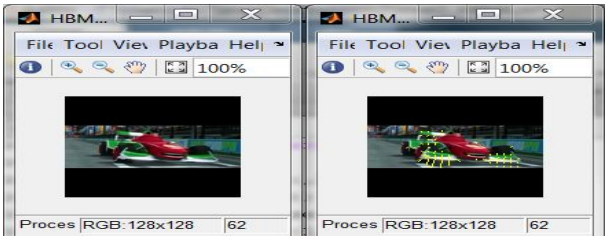
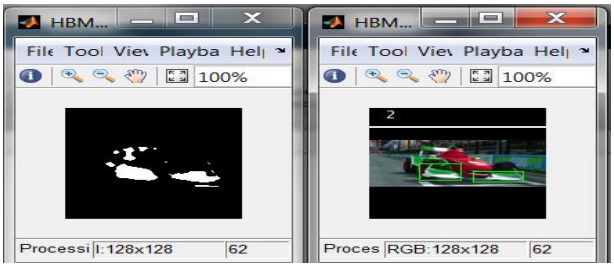


Figure 42 and 43: Original Video (left side), Motion Vector video (right side)

In figure 44, shows the blacks spot as a threshold that tells how much threshold it receiving from motion vector video of figure 43. Figure 45 shows the resultant video image of 4(four) objects moving in it



ISSN 2229-5518

Figure 44 and 45: Threshold Video (left side) and Resultant Video (right side)

5.3 20th and 30th frame for Wildlife2.avi to computer Original, Motion Vector, Threshold, and Resultant

Figure 46(a) represents the Original video image for 20th and 30th frame using the HBMA over EBMA.

Figure 46(b) represents the Motion Vector video image for 20th and 30th frame. "Yellow" dot indicates the object moving in the motion vector video image

Figure 46(c) represents the Threshold video image for 20th and 30th frame. It shows the blacks spot as a threshold that tells how much threshold it receiving from motion vector video of figure 35(b).

Figure 46(d) represents the Resultant video image for 20th and 30th frame. Therefore, it shows the resultant video image of 1(one) objects moving in it.

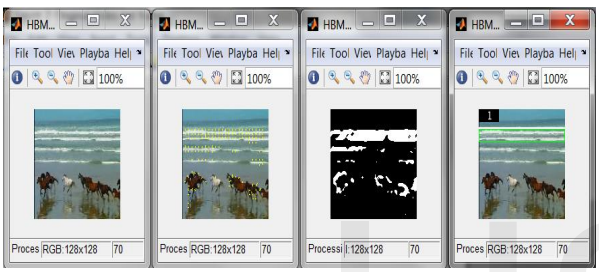


Figure 46(a) Original video, (b) Motion vector video, (c) Threshold video, and (d) Resultant video of Wildlife2.avi for 20th and 30th frame

6 Implementation for Evaluation metrics used for 3 level HBMA algorithm and Lossless Huffman coding using K-Means algorithm

First table represent the results for Motion estimation using HBMA over EBMA.

(a) Peak Signal to Noise ratio (PSNR): In order to reduce computational complexity and try to achieve the same PSNR fast searching algorithms have been developed. [4, 12]. It is an expression for the ratio between the maximum possible value power of a signal and the power of distorting noise that affects the quality of its representation.

PSNR = 10*real(log10(n*n/mse)); where 'n' is the peak value possible of any pixel in the images.

(b) MSE (Mean Squared Error) : It is arguably the most important criterion used to evaluate the performance of a predictor or an estimator.

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}\|f(i,j) - g(i,j)\|^2 \qquad (19)$$

f represents the matrix data of our original image 'g' represents the matrix data of our degraded image m represents the numbers of rows of pixels of the images and i represents the index of that row n represents the number of columns of pixels of the image and j represents the index of that column.

Therefore, Mean Absolute Difference is used in this research paper

(c) MAD (Mean Absolute Difference): The mean difference is a measure of statistical dispersion equal to the average absolute difference of two independent values drawn from a probability distribution. A related statistic is the relative mean difference, which is the mean difference divided by the arithmetic mean.

For a population of size *n*, with a sequence of values y_i, i = 1 to n:

$$MD = \frac{1}{n(n-1)}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n}|y_i - y_j|. \qquad (20)$$

Motion Estimation for 3 level HBMA over EBMA using Shark.avi using 1st and last frame based on block size of 8, 16, and 64

Block size=8	Mean Absolute Difference(MAD) 67.0300	PSNR reference to target 28.8901	MSE 83.9587	PSNR target to estimation 28.9307	MSE 83.1792
Block size=16	Mean Absolute Difference(MAD) 26.5281	PSNR reference to target 28.8901	MSE 83.9587	PSNR target to estimation 29.2811	MSE 76.7313
Block size=64	Mean Absolute Difference(MAD) 15.2358	PSNR reference to target 28.8901	MSE 83.9587	PSNR target to estimation 31.6606	MSE 44.363

Table 1

Huffman coding using K-Mean algorithm based on block size of 16 and 64

Block size=16 & 64 and Codebook size=50

Block size=16	Rate 0.3527	CR 22.6795	SNR 16.0649	Entropy 1.13	Average length 91.26	Time taken for compression=21.527 Time taken for Decompression=27.664590	Huffman coding Compression ratio 9.874527	MSE 39689.66	PSNR 2.18	Huffman Coding PSNR 18.360426
Block size=64	Rate 0.0882	CR 90.7181	SNR 14.0187	Entropy 1.19	Average length 67.01	Time taken for compression=10.915122 seconds Time taken for Decompression=16.511938	Huffman coding compression ratio 15.215718	MSE 15828.21	PSNR 6.17	Huffman Coding PSNR 17.297536

Table 2

7 Future Scope

Tremendous scope of research is going in the field of image compression and motion estimation. In future, more parameters may be added to extend the analysis of Image compression for the images using Lossless Huffman coding for K-Means algorithm in the context of motion estimation.

8 Conclusion

Motion estimation plays a very important role in motion compensated coding algorithms. It's also the most time consuming operation in the codec system. Many research works on block-based motion estimation algorithms were conducted to reduce the computational cost in three ways: 1) fast search by reduction of the number of candidate blocks for matching [1][2]; 2) fast algorithm by reduction of the computational complexity of the matching criteria [3][6]; 3) fast algorithm by block motion field subsampling. To improve motion estimation search time, there has been a tremendous contribution by various researchers and experts for the past two decades to refine block-matching algorithms [14].

The original video, motion vector video, threshold video and their results shows better observations in this research paper. Original image and Motion vectors image for 3 different videos like 1st, middle, last frame, etc. of image are used via MATLAB implementation during this algorithm. 1st frame and last frame of the 2 second original video, and first frame and middle frame of the 2 second motion vector video by RGB to intensity format were computed. MSE and other evaluation metrics used such as PSNR and MAD for 3-level HBMA over EBMA in this paper.

However, from the figures 41, figure 45, and 46(d), it indicates that car.avi shows 2 objects moving as compare to Shark.avi and Wildlife2.avi. In fact, analyzed from 3 tables such car.vi shows less MSE as compare to Shark.vi and Wildlife2.avi.

In this research paper, the analysis of lossless Huffman coding for image compression using K-Means algorithm along with HBMA over EBMA video estimation is used for block size of 16 and 64. In our analysis, there is not much variation in PSNR when the block size is changed from 8, 16, and 64. But there is a considerable change in the Mean Absolute Difference when the block size is varied. More accurate estimation is obtained when the block size is increased. Furthermore, MSE get decreased, when we take block size of 64 instead of 16 in the context of Lossless Huffman code using K-Means algorithm and Video estimation HBMA over EBMA algorithm.

Acknowledgment:

I want to thanks Dr. Jung H. Kim for his support and contribution along with his ideas in this research.

References:

[1] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation," *IEEE Trans. Image Processing,* vol. 4, pp. 105-107, Jan. 1995.

[2] C.-H. Lee and L.-H. Chen, "A Fast Motion Estimation Algorithm Based on the Block Sum Pyramid," *IEEE Trans. Image Processing,* vol. 6, pp. 1587-1591, Nov. 1997.

[3] B. Zeng, R. Li, and M. L. Liou, "Optimization of fast block motion estimation algorithms," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 7, pp. 833-844, Dec.1997.

[4] K. Sayood, *Introduction to Data Compression*, third ed.: Morgan Kaufmann, 2006.

[5] Yao Wand, Jorn Ostermann and Ya-Qin Zhang, "*Video Processing and Communications*", Prentice Hall, 2007.

[6] L.-K. Liu and E. Feig, "A block-based gradient decent search algorithm for Block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 6, pp. 833-844, Aug. 1996.

[7] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 6, pp. 3133-316, Jun. 1996.

[8] Kuk-Jin Yoon, In So Kweon, "Adaptive Support-Weight Approach for Correspondence Search", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 28, no. 4, pp. 650-656, 2006.

[9] Yun Q.Shi and Huifang Sun, "*Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*", CRC press, 2000.

[10] Du Yehui, Gu Jia, "Research on Stereo Matching Algorithms", In Proceeding of Conf. International Federation for Medical and Biological Engineering, pp. 521-524,2010.

[11] F. Dufaux and F. Moscheni: 'Motion Estimation Technique for digital TV : A Review and a new contribution', Proc, IEEE, vol. 83, pp. 858-876, 1995.

[12] S. R. and R. K.R., "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun,* vol. 33, pp. 888–896, 1985.

[13] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen and L.-G. Chen, "Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results," *The Journal of VLSI Signal Processing,* vol. 42, pp. 297-320, 2006.

[14] A. Puri, H. M. Hang and D. Schilling, "An efficient block-matching algorithm for motion-compensated coding," *IEEE International Conference on Acoustics, Speech, and Signal Processing,,* vol. 12, pp. 1063-1066, Apr 1987 1987.

[15] M. Ezhilarasan and P. Thambidurai, "Simplified Block Matching Algorithm for Fast Motion Estimation in Video Compression " *Journal of Computer Science,* vol. 4, pp. 282-289, 2008.

[16] [14] A. Barjatya, "Block Matching Algorithms for Motion Estimation," DIP 6620 Spring 2004.

[17] C. Cai, H. Zeng and S. Mitra, "Fast motion estimation for H.264," *Signal Processing: Image Communication,* 2009.

[18] B. C. Song and J. B. Ra, "A hierarchical block matching algorithm using partial distortion measure," *Visual Communications and Image Processing '98 (Proceedings Volume),* vol. 3309, pp. 88-95, 1998

[19] I. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*: Wiley, 2003.

[20] D.-Y. Huang, "A XviD-based Video Codec for Computer Animation," MSC, 2005.

[21] H.B.Kekre, Tanuja K Sarode, Sanjay R Sange(2011) " Image reconstruction using Fast Inverse Halftone & Huffman coding Technique", IJCA,volume 27-No 6, pp.34-40.

[22] Manoj Aggarwal and Ajai Narayan (2000) "Efficient Huffman Decoding", IEEE Trans, pp.936-939.

[23]http://www.webopedia.com/TERM/H/Huffman_compression.html

[24] Gupta, K., Verma, R.L. and Sanawer Alam, Md. (2013) Lossless Medical Image Compression Using Predictive Coding and Integer Wavele Transform based on Minimum Entropy

Criteriat. International Journal of Application or Innovation in Engineering & Management (IJAIEM), 2, 98-106.

[25] Mishra, K., Verma, R.L., Alam, S. and Vikram, H. (2013) Hybrid Image Compression Technique using Huffman Coding Algorithm. International Journal of Research in Electronics & Communication Technology, 1, 37-45.

BIOGRAPHIES

Ali Tariq Bhatti received his Associate degree in Information System Security (Highest Honors) from Rockingham Community College, NC USA, B.Sc. in Software engineering (Honors) from UET Taxila, Pakistan, M.Sc in Electrical engineering (Honors) from North Carolina A&T State University, NC USA, and currently pursuing PhD in Electrical engineering from North Carolina A&T State University. Working as a researcher in campus and working off-campus too. His area of interests and current research includes Coding Algorithm, Networking Security, Mobile Telecommunication, Biosensors, Genetic Algorithm, Swarm Algorithm, Health, Bioinformatics, Systems Biology, Control system, Power, Software development, Software Quality Assurance, Communication, and Signal Processing. For more information, contact Ali Tariq Bhatti alitariq.researcher.engineer@gmail.com.

Dr. Jung H. Kim is a professor in Electrical & Computer engineering department from North Carolina A&T State University. His research interests includes Signal Processing, Image Analysis and Processing, Pattern Recognition, Computer Vision, Digital and Data Communications, Video Transmission and Wireless Communications.