International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

# Implementation of H.264 Video Codec for Block Matching Algorithms

**Vivek Sinha[1], Dr. K. S. Geetha[2]**

[1]Student of Master of Technology, Communication Systems, Department of ECE,
R.V. College of Engineering, Bengaluru, Karnataka, India

[2]Professor and Associate Dean, Department of ECE, R.V. College of Engineering, Bengaluru, Karnataka, India

**Abstract:** ***Digital video processing is a technique in which the input video sequence is compressed before being transmitted or stored. The H.264 Advanced Video Codec performs video processing with a good coding efficiency and robustness to network losses. Due to this reason it finds its application in low bit-rate Internet streaming as well as HDTV broadcast and Digital Cinema with nearly lossless coding. The H.264 codec uses block-based motion estimation techniques which reduces the time delay in determining the motion vectors. These motion estimation techniques are performed by different block matching algorithms, namely Exhaustive Search (ES), Diamond Search (DS) and Adaptive Rood Pattern Search (ARPS). The motion vectors obtained require more computation time for its implementation depending on the type of block matching algorithm used. The main objective of this paper is to validate the improved performance of the H.264 codec for various motion estimation algorithms under different motion video conditions.***

**Keywords**: H.264/AVC, Motion Estimation, Motion Compensation, Exhaustive Search, Diamond Search, Adaptive Rood Pattern Search

## 1. Introduction

The *H.264 / MPEG-4* Part 10 Advanced Video Coding [1] is the latest advancement seen in the field of video coding for international standards which also includes the H.261, MPEG-2 and H.263 standards. This standard has the highest coding efficiency commercially with low implementation complexity and cost. The main aim of the video coding standard is to enhance the video quality keeping in mind the bit budget constraint [2]. For this reason, the two parameters which define the quality for any given video, Peak Signal-to-Noise Ratio (*PSNR*) and Compression Ratio (*CR*) are regularly used in these coding techniques. The layout of H.264 encoder/decoder is depicted in Fig. 1.

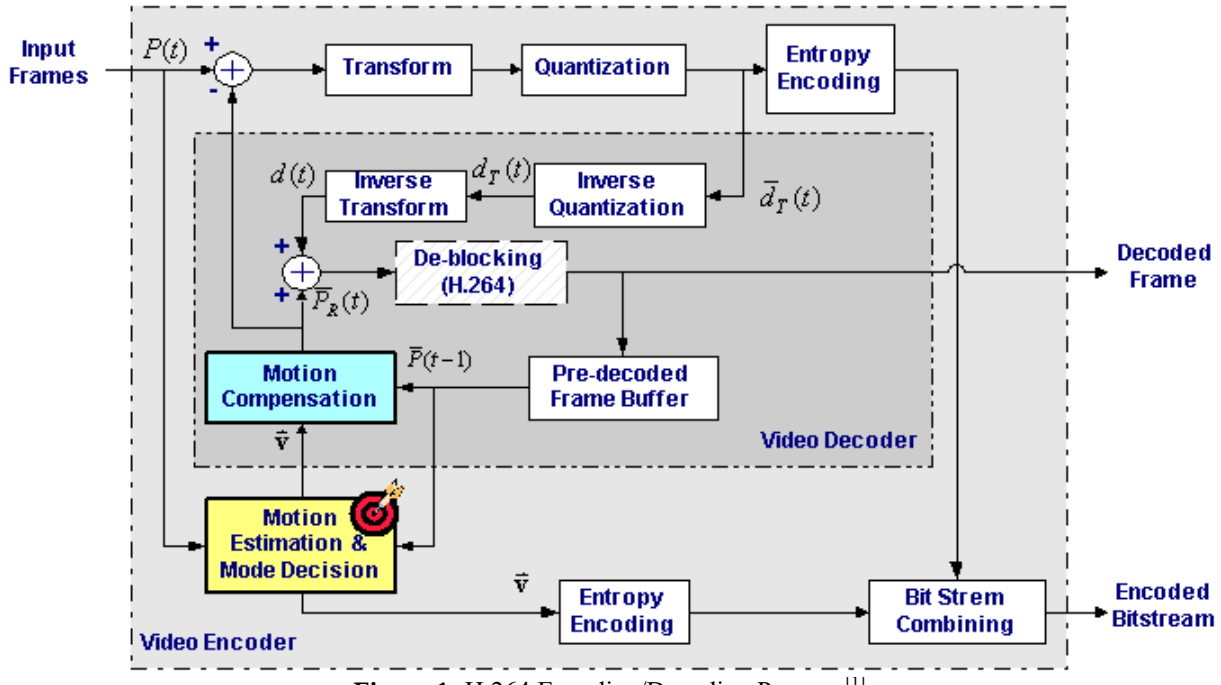



**Figure 1:** H.264 Encoding/Decoding Process [1]

Image data of a scalable video [3] that is fed into the encoder is divided into nonoverlapping blocks of variable sizes. Each block is called a macroblock (*MB*). The *MB* size chosen for this paper is 16x16 pixels. The block-based motion estimation technique [4] is used to determine the motion vectors (*MVs*) by obtaining the difference between a pre-stored reference frame and the current input frame. This difference is also known as displacement vector. The displacement vector can be computed using various Block Matching Motion Estimation Algorithms. These *MVs* can then be used to predict the candidate frame. Such a function is carried out by the motion compensation unit. Thus, if displacement is applied to the reference frame at different regions, the appearance of the resulting candidate frame is

shown by the motion compensation block. If the input frame if a reference frame, it is directly sent to the transformation and quantization block. Otherwise, transformation and quantization [5] is applied to the motion compensated image. Depending upon the type of block matching algorithm used, the compression ratio may vary from 5 to 60, thereby increasing the efficiency during real time applications. The *CAVLC* entropy encoder [6] is used for encoding the frame information to bitstreams.

In the decoding process, the decoder decodes a *MB*, rescales it and performs inverse transformation to obtain the residual *MB*. The prediction, which is generated by the decoder is same as the prediction generated by the encoder, is added to the residual *MB* and reconstructs the original *MB*.

The layout of this paper is as follows. Section 2 deals with the different block matching algorithms (*BMA*) which are implemented on the H.264 codec. Section 3 gives the mathematical representation of the *BMA*s discussed. Section 4 provides the result analysis and Section 5 concludes this paper.

## 2. Block Matching Motion Estimation Algorithms

### 2.1 Exhaustive Search (*ES*) Algorithm:

The most straight forward (*BMA*) is the exhaustive search algorithm [7]. In this algorithm, in order to find the best matching block, it exhaustively searches all the macro blocks with in the search window to find the best match. The implementation detail shows that though exhaustive search method is accurate but extremely computationally complex especially when applied to *HDTV* and *SHDTV* signals. Thus becomes main bottle neck in real time video applications.

### 2.2 Diamond Search (*DS*) Algorithm:

The Diamond Search Algorithm [8] performs motion estimation using two search patterns as shown in Fig. 2. Large Diamond Search Pattern (*LDSP*), the initial search pattern, consists of eight check-points which surrounds the ninth (centre) check-point. This gives the shape of a diamond (◊). The second search pattern, Small Diamond Search Pattern (*SDSP*) comprises of five check-points and has the shape of a smaller diamond.

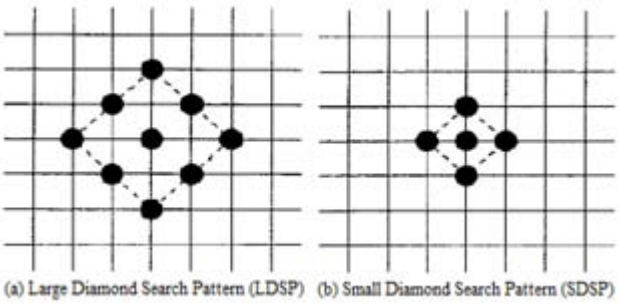



**Figure 2:** Two Search Patterns for DS Algorithm [8]

The steps of the *DS* algorithm is:

**Step 1)** The search window consists of the *LDSP* placed at its centre and the minimum block distortion (*MBD*) position is calculated through its nine check-points. If the *MBD* is found to be at the centre of the search pattern, go to Step 3, else go to Step 2.

**Step 2)** The centre of the *LDSP* is re-positioned at the *MBD* position found in the previous step and Step 1 is repeated until the *MBD* is obtained at the centre of the *LDSP*. Once obtained, go to Step 3.

**Step 3)** *SDSP* takes over from *LDSP*. The *MBD* obtained is placed at the centre of *SDSP* and repetitive iterations are performed until the local minimum is obtained at the centre of *SDSP*. Finally, this position gives the motion vectors of the best matching block.

The search of the *MBD* is restricted to the boundary of the search window. Any point outside the search boundary is truncated. Computation of the *MBD* position is a repetitive process and each iteration leads us to the accurate position value. The two-step procedure provides a faster to determine the motion vectors and thus reduce the computation time. The adaptive rood pattern search algorithm further helps in reducing the computation time as it uses a much flexible approach to determine the global minimum.

### 2.3 Adaptive Rood Pattern Search (ARPS) Algorithm

The Adaptive Rood Pattern [9] carries out the estimation where the search pattern used are of two types: the first is the adaptive rood pattern (*ARP*) which consists of a rood shape "+" whose arms can be varied as per the requirement. This determines the pattern size. Thus, depending on the value of the predicted motion behaviour, the *ARP* armlength is adjusted accordingly for each macroblock. One of the important features of the *ARP* is that it is used at the beginning of every macroblock search, that too only once. This is done to provide the best starting point in the search window from where a local search can be performed and the global minimum can be obtained by checking minimum number of check-points. It is depicted in Fig. 3.

For MBs which lie on the leftmost part of the frame, arm length is taken as 2 pixels (Γ=2), since it gives a robust performance. Otherwise, the arm-length is calculated using the formula given in Equation 1:

$$\Gamma = \text{Round}\,|\overrightarrow{MV}_{predicted}|$$
$$= \text{Round}\left(\sqrt{\overrightarrow{MV}^2_{predicted}(a) + \overrightarrow{MV}^2_{predicted}(b)}\right) \quad (1)$$

where $MV_{predicted}(a)$ is the horizontal component of the predicted *MV* and $MV_{predicted}(b)$is the vertical component of the predicted *MV*. In calculating Γ, two square operations and one square-root operation is performed, which increases the complexity during hardware implementation. To reduce this complexity, an alternate method is employed where the maximum of two direction components is the size of the rood arm as given by Equation 2, i.e.

$$\Gamma = \text{Max}\,\{|MV_{predicted}(a)|, |MV_{predicted}(b)|\} \quad (2)$$

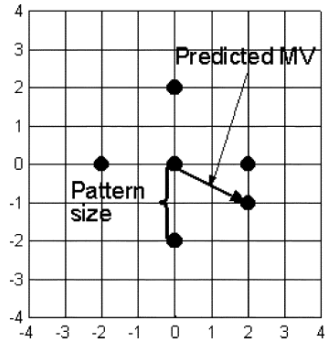



**Figure 3:** Adaptive Rood Pattern [9]

The second is the unit-size rood pattern which has a fixed arm length of one pel and it performs the final search of the algorithm. This fixed search is repeated until the local minimum is obtained at the centre of the search pattern. This is shown in Fig. 4.

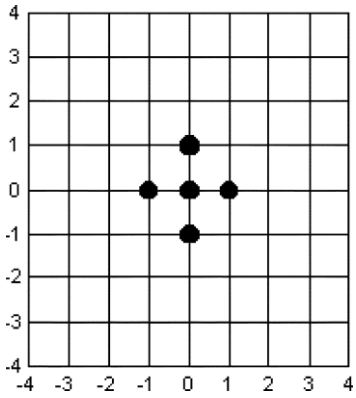


**Figure 4:** Fixed Pattern Search - Refined Local Search [9]

The steps of the *ARPS* algorithm is:

**Step 1)** Computing the matching error (sum of absolute differences, $SAD_{centre}$) between the current block and the block at the same location in the reference frame

if $SAD_{centre} < T$ (threshold)
$\overrightarrow{MV_{target}} = [0\ 0]$
Stop;
else
if the current block is a left-most boundry block,
$\Gamma = 2$;
else
$\Gamma = \text{Max}\{|MV_{predicted}(a)|, |MV_{predicted}(b)|\}$
Go to Step 2.

**Step 2)** The centre position of the search window is to be aligned with the centre of *ARP* and verify the four points along with the position of the predicted MV to obtain the current *MME*.

**Step 3)** Place the centre of the *URP* at the acquired *MME* point in the last step and note the checkpoints. Redo until the *MME* is obtained at the centre of the *URP*. Once obtained, the *MV* value is the final value.

## 3. Mathematical Equations

The motion compensation for the block matching algorithms is given by Equation 3:

dy = motionVect(1,mbCount) row index
dx = motionVect(2,mbCount) column index (3)

where each macroblock is allocated a motion vector (motionVect) which have integer values in both horizontal and vertical directions and mbCount is the the total number of macroblocks into which an image frame is divided. Equation 4 gives the two dimensional (2-D) discrete cosine transform [10]:

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos\frac{(2j+1)\pi y}{2N} \cos\frac{(2i+1)\pi x}{2N} \quad (4)$$

where $Y_{xy}$ is the input coefficient of the image sample and $X_{ij}$ is the output coefficient of the image sample. Similarly, the inverse 2-D discrete cosine transform is given by Equation 5:

$$Y_{xy} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_i C_j X_{ij} \cos\frac{(2y+1)\pi j}{2N} \cos\frac{(2x+1)\pi i}{2N} \quad (5)$$

where, *M* and *N* are the rows and columns of the image pixels respectively.

The peak signal-to-noise ratio is given by Equation 7 as:

$$PSNR = 20.\log_{10}(MAX) - 10.\log_{10}(MSE) \quad (7)$$

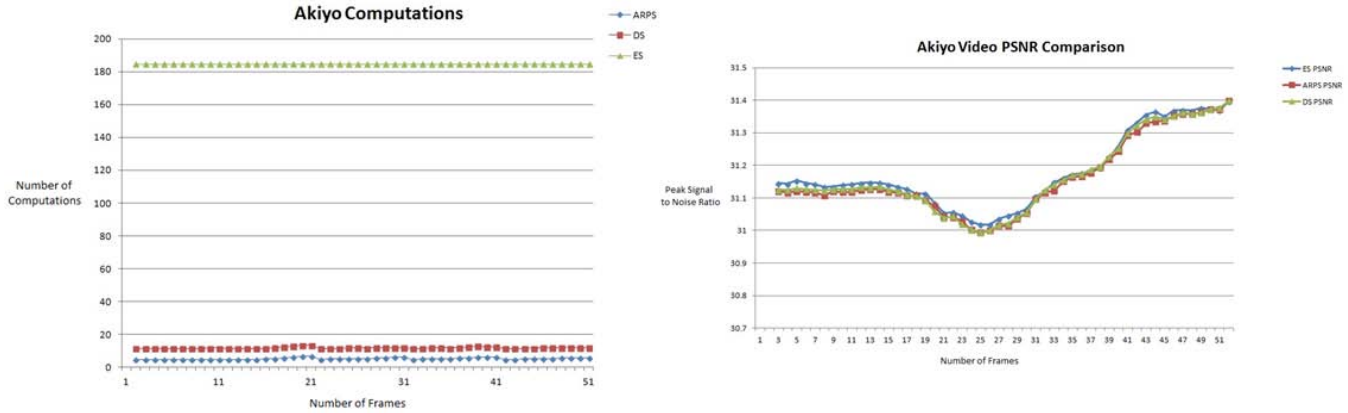
## 4. Simulation Results

The test video sequences considered in this work have the following specifications which are described in Table 1. While the test sequences considered are fast moving videos as well as slow moving videos, one of the videos consists of an underwater sequence. The effect of these parameters vastly affects the quality of the video reconstructed. The performance of the H.264 codec improves as the motion of the objects within the video reduces. This is due to the fact that smaller motion variation leads to much accurate determination of the motion vectors when compared to the video where motion is high. Ultimately, this results in a relatively better PSNR value which has been verified in this section.

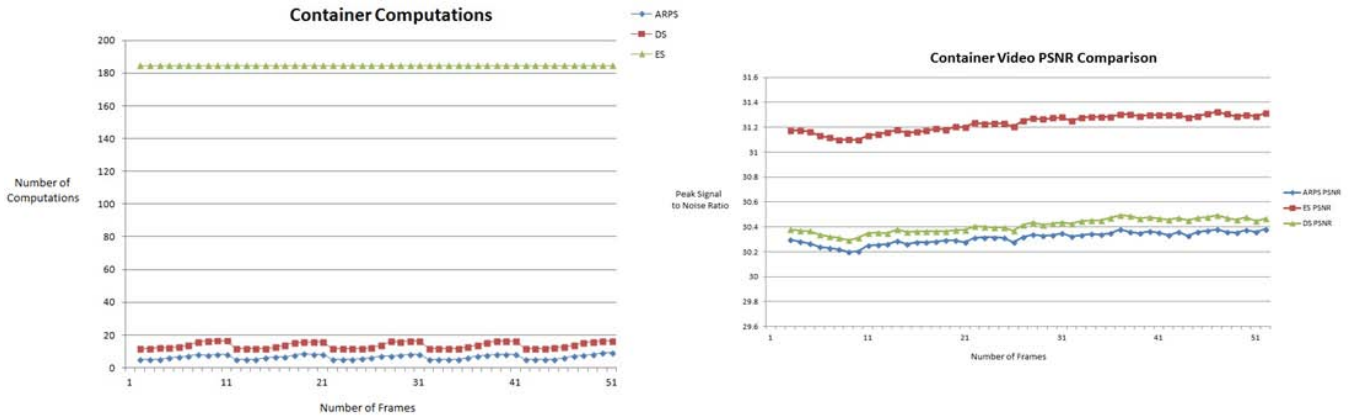
**Table 1:** Specification of Test Video Sequences

| Test Video | Video Format | Number of Frames |
|---|---|---|
| Akiyo | QCIF | 50 |
| Container | QCIF | 50 |
| News | QCIF | 50 |
| Underwater | QCIF | 50 |

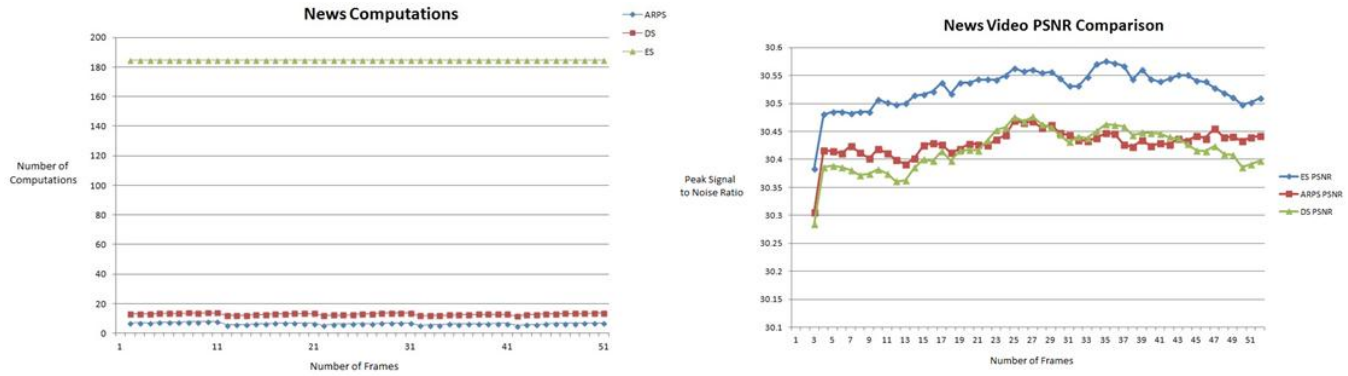
Using the above parameters, these different test video sequences were implemented on the *H.264 AVC/MPEG-4* Part 10 by applying various block matching algorithm. The result analysis is done using Fig. 5.



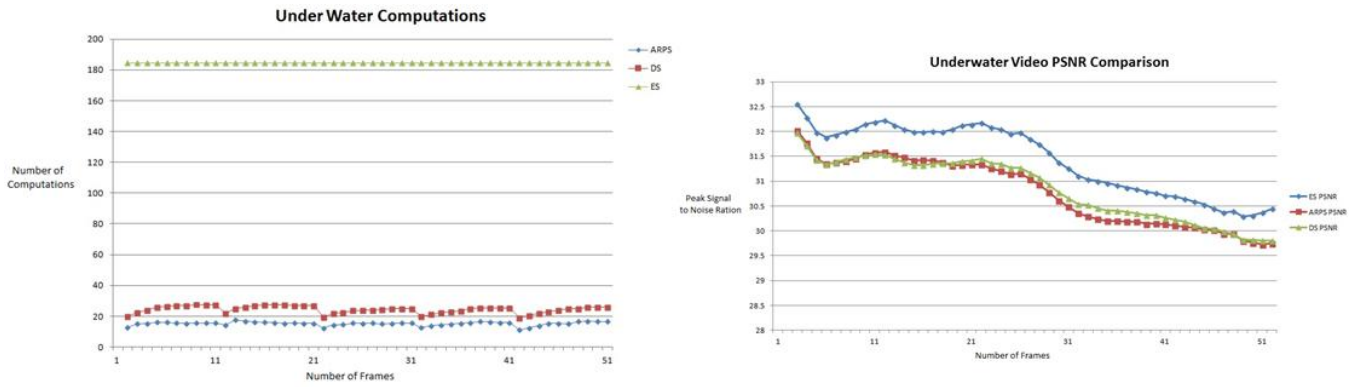

(a) Akiyo Video




(b) Container Video




(c) News Video




(d) Underwater Video

**Figure 5:** Result Analysis of various Test Video Sequences

Various test video sequences were implemented onto the H.264 video codec using the exhaustive search, diamond search and adaptive rood pattern search algorithms. The total number of computations to obtain the motion vectors were calculated and compared for each of the algorithms. The *PSNR* of video sequence was then calculated and compared for the three algorithms. This is depicted in Fig. 5. It can be observed that the exhaustive search undertakes maximum number of computations as compared to the *DS* and *ARPS*. This concludes that *ES* cannot be used for real-time applications. It can also be observed that the *PSNR* is best for *ARPS* when compared to the *ES* and *DS*. Thus, it can be said that *ARPS* is the best algorithm for motion estimation and detection as it takes minimum number of computations while maintaining a desired *PSNR*.

It can therefore be concluded that *ARPS* is the best approach for any type of video scenario. It can be used for real-time applications as it provides a good *PSNR* with the least number of computations. Table 2 below summarizes the overall implementations of this project. From the table it can be observed that the compression ratio obtained from ARPS is, by far, the most effective choice for transmission of information sequences.

**Table 2:** Comparison between Different Test Sequences for PSNR, Compression Ratio and number of Computations

| Parameter | BMAs | Akiyo | News | Container | Under Water |
|---|---|---|---|---|---|
| PSNR | ES | 31.1785 | 30.5270 | 31.2316 | 31.2861 |
| | DS | 31.1668 | 30.4180 | 30.6954 | 30.6954 |
| | ARPS | 31.1620 | 30.4291 | 30.3122 | 30.6184 |
| Compression Ratio | ES | 5.3566 | 5.7246 | 5.9772 | 5.6531 |
| | DE | 22.3294 | 22.6426 | 22.9780 | 22.6327 |
| | ARPS | 61.2210 | 61.4667 | 61.5211 | 61.2161 |
| Number of Computation | ES | 188.1481 | 188.1481 | 188.1481 | 188.1481 |
| | DS | 11.6017 | 11.7980 | 11.6708 | 14.4380 |
| | ARPS | 4.9727 | 5.1867 | 5.0656 | 7.9199 |

## 5. Conclusion

Transmission of information in a low bitrate environment is always a challenging task. In this project, several test video sequences were taken and were implemented on the H.264/MPEG-4 Part 10 AVC. The PSNR obtained for these sequences clearly shows that the Exhaustive Search maintains a relatively high PSNR value when compared to the Diamond Search and the Adaptive Rood Pattern Search algorithms. The tabulation of the compression ratios show that the ARPS algorithm provides a maximum compression while the ES provides the least compression. When the computational speeds of these algorithms were computed, the ARPS took the least time as compared to DS. ES, which performs pixel-by-pixel search, took the maximum time for computation. Overall, the ARPS is, by far, the most effective block matching algorithm which provides the maximum compression ratio with minimum computation time while maintaining a desirable PSNR.

## References

[1] Iain E.Richardson, "The H.264 Advanced Video Compression Standards", *John Wiley and Sons, Ltd.*, 2003, pp. 76-231.
[2] C. E.Shannon, “A mathematical theory of comm nication,” *Bell Syst. Tech. J., volume 27,* July–October 1948, pp. 379–422, 623–654.
[3] Detlev Marpe and Heiko Schwarz,"Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", *Li-Weiley Publications*, Jul. 2007, pp. 365-431.
[4] R. J.Clarke, "Transform Coding of Images", *ISBN:0121757315, Acdemic Press,* 1985, pp. 234-254.
[5] Loukil, H. " An efficient hardware architecture design for H.264/AVC INTRA 4×4 algorithm" *Laboratory of Electronics & Inf. Technology, Sfax National Engineering School*, December 2008, pp. 173 - 176.
[6] W.Badawy and C.A.Rahman, “CAVLC Encoder Design for Real Time Mobile Video Applications”, *Circuits and Systems II: Express Briefs, IEEE Transactions, Volume 54, Issue 10*, October 2007, pp. 873-876.
[7] S.Zh , “Fast Motion Estimation Algorithms For Video Coding”, *M.S. thesis, School Elect. Electron. Enineering.*, Nanyang TechNumber University of Singapore, 1998., pp. 212-326.
[8] Kai-Kuang Ma and Shan Zhu, " A New Diamond Search Algortthm for Fast Block-Matching Motion Estimation", *IEEE Transactions on Image Proccesing, Volume 9, Number 2,* February 2000, pp. 287-290.
[9] Kai-Kuang Ma Senior Member, IEEE and Yao Nie, Student Member, IEEE, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation", *IEEE Transactions on Image Processing, Volume 11, Number 12*, December 2002, pp. 1442-1449.

## Author Profile

**Vivek Sinha** received his B.E. degree from Nagarjuna College of Engineering and Technology, Visvesvaraya Technological University in 2012. He is currently pursuing his M. Tech. from R.V. College of Engineering, Visvesvaraya Technological University (2013-2015).