International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 9, September- 2014

Frame Rate Up-Conversion using Trilateral Filtering For Video Processing

[1]Dr. M. Anto Bennet,
[1]Professor, Department of ECE,
Veltech, Chennai-600062

[2] M. Manimaraboopathy, [3] R. Srinath
[4] P. Maragathavalli [5] S. Mekala
[2,3,4,5] Assistant Professors,
Department of ECE, Veltech, Chennai-600062

***Abstract*: - This paper demonstrates the enhancement of the visual quality of low frame rate video presented on liquid crystal display. By using two methods such as Frame rate up conversion algorithm and Trilateral filtering to minimize the difference between the adjacent frames or blocks, an effective FRUC algorithm partitions a large block into several sub-blocks of smaller size and estimates their motions between the frames. Motion estimation searches for the block which has the minimum difference (cost) with the processed block in terms of some block matching Algorithm. In the proposed FRUC method, the two predictions of a frame to be interpolated are generated through shifting its nearest neighbor frames in the previous and following directions with the motion vectors estimated between them. The initial interpolated frame and its pixel's reliability are subsequently estimated from these two predictions. Then apply a trilateral filter on the initial prediction to correct the unreliable pixels and to restore the missing pixels. Their Performance can be analyzed with measuring PSNR Parameter.**

Keywords: Trilateral Filter, Frame up Conversion, Bilateral Filter

I. INTRODUCTION

A.FRAME RATE UP CONVERSION

In video periodically skipping frames can reduce the video bit rate, the visual quality of the video, especially motion continuity, is also inevitably degraded. To have good video rendition with smooth motion, the player (decoder) needs to increase the temporal resolution of the video by using a Frame Rate Up-Conversion (FRUC) scheme. FRUC scheme is used to produce more frames so that the LCDs can display these frames faster. FRUC schemes are rather effective in reducing the LCD hold time and motion blur, and they are increasingly common in high quality LCD [7]televisions. There are two categories of FRUC algorithms. The first category, including such methods as frame repetition and temporal frame averaging, interpolates the middle frame in the temporal domain (the intermediate frame) without considering scene changes. These algorithms work well for static scenes, but they will produce jerkiness and blurring effects around moving objects. To address this problem, the second category, generally known as motion compensated interpolation (MCI), is proposed to predict the intermediate frame by using either linear or nonlinear interpolators to adapt to scene changes. Motion estimation is a key step of MCI. To obtain a higher temporal resolution, accurate motion estimation is required for smooth video rendition. Although optical flow estimation can provide video motion down to pixel or sub-pixel level, it is rarely used for MCI due to its high computational cost. Hence, most FRUC algorithms make use of some block matching algorithm, which is computationally less complex, to generate the video motion.

A. B.TRILATERAL FILTERING

Trilateral filtering is mainly used in de noising and also used in correcting the unreliable pixels, thus the missing values are obtained. The neighboring frames are highly correlated. The restore information from the reference frame and estimated frames are used obtain the missing values, thus the noise error are suppressed. The biasing estimation which has less accurate than its corresponding reference block. It is feasible to obtain the similar blocks in the frames of better estimate. Trilateral filter for it smoothes the interpolated [8,9,13] errors in both the spatial and temporal domains. The interpolated frame and the reference frames accurate pixels should contribute more for frame interpolation as well as restoration. It is therefore necessary to filter the interpolated frame step by step can be adaptively selected according to the available computation resources. It is heuristic that more. Trilateral filter it smoothes the interpolated error in both the spatial and temporal domains. The interpolated frame and the reference frames can be adaptively selected. According to the available computation resource is heuristic that more accurate pixels should contribute more for frame interpolation as well as restoration. It is necessary to filter the interpolated frame step by step. The trilateral filter is applied to exploit redundancy among the spatial and temporal neighboring blocks to remove the noise and errors of the initial estimation. The trilateral filter is applied to the last stage of FRUC algorithm. The trilateral filter is used to correct the unreliable pixels and to restore the missing pixels. The trilateral filter is applied for both spatial and temporal domains. Due to these properties, we selected the trilateral filtering to correct the unreliable pixels.

The trilateral filtering is not suitable for the frequency domain because of there is the complexity of restore the missing pixels. It is also remove the noise unwanted tool and error of the initial estimation of frame. Many smoothing filters have been introduced, varying from the simple mean and median filtering to more complex filters such as anisotropic filtering .These filters aim at smoothing the image to remove some form of noise.

The trilateral filter was introduced as a means to reduce impulse noise in images. The principles of the filter were based on the bilateral filter which is an edge-preserving Gaussian filter. The trilateral filter was extended to be a gradient-preserving filter, including the local image gradient (signal plane) into the filtering process. We present a new, single-pass nonlinear filter for edge-preserving smoothing and visual detail removal for N dimensional signals in computer graphics, image processing and computer vision applications. Unlike bilateral filters or anisotropic diffusion methods that smooth towards piecewise constant solutions, the trilateral filter provides stronger noise reduction and better outlier rejection in high-gradient regions, and it mimics the edge-limited smoothing behavior of shock-forming PDEs by region finding with a fast min-max stack. Yet the trilateral filter requires only one user-set parameter, filters an input signal in a single pass, and does not use an iterative solver as required by most PDE methods. Like the bilateral filter, the trilateral filter easily extends to N-dimensional signals, yet it also offers better performance for many visual applications including appearance-preserving contrast reduction problems for digital photography and de-noising polygonal meshes.

II. EXISTING METHOD

Filtering is perhaps the most fundamental operation of image processing and computer vision. In the broadest sense of the term "filtering", the value of the filtered image at a given location is a function of the values of the input image in a small neighborhood of the same location. For example, Gaussian low-pass filtering computes a weighted average of pixel values in the neighborhood, in which the weights decrease with distance from the neighborhood center. Although formal and quantitative explanations of this weight fall-off can be given, the intuition is that images typically vary slowly over space, so near pixels [11,12,15] are likely to have similar values, and it is therefore appropriate to average them together. The noise values that corrupt these nearby pixels are mutually less correlated than the signal values, so noise is averaged away while the assumption of slow spatial variations fails at edges, which are consequently blurred by linear low-pass filtering. Many efforts have been devoted to reducing this undesired effect. Bilateral filtering is a simple, non-iterative scheme for edge-preserving smoothing. The basic idea underlying bilateral filtering is to do in the range of an image what traditional filters do in its domain. Two pixels can be close to one another, that is, occupy nearby spatial location, or they can be similar to one another, that is, have nearby values, possibly in a perceptually meaningful fashion.

Fig 1

Fig 2

Figure 1 and 2 shows the potential of bilateral filtering for the removal of texture. The picture "simplification" illustrated by figure 2 can be useful for data reduction without loss of overall shape features in applications such as image transmission, picture editing and manipulation, image description for retrieval.

In statistics, a moving average, also called rolling average, rolling mean or running average, is a type of finite impulse response filter used to analyze a set of data points by creating a series of averages of different subsets of the full data set. The median filter is well-known .However, if a user wishes to predefine a set of feature types to remove or retain, the median filter does not necessarily satisfy the requirements. A more general filter, called the Weighted Median Filter, of which the median [1-3] filter is a special case, is described. It enables filters to be designed with a wide variety of properties. The question of finding the number of distinct ways a class of filters can act is considered and solved for some classes.

Frame rate up conversion (FRUC) methods that employ motion have been proven to provide better image quality compared to non motion-based methods. While motion-based method improves the quality of interpolation, artifacts are introduced in the presence of incorrect motion vectors. In this paper, we study the design problem of optimal temporal interpolation filter for motion compensated FRUC (MC-FRUC). The optimal filter [4,6] is obtained by minimizing the prediction error variance between the original frame and the interpolated frame. In FRUC applications, the original frame that is skipped is not available at the decoder, so models for the power spectral density of the original signal and prediction error are used to formulate the problem.

The closed-form solution for the filter is obtained by Lagrange multipliers and statistical motion vector error modeling. The effect of motion vector errors on resulting optimal filters and prediction error is analyzed. The performance of the optimal filter is compared to non adaptive temporal averaging filters by using two different motion vector reliability measures. The results confirm that to improve the quality of temporal interpolation in MC, the interpolation filter should be designed based on the reliability of motion vectors and the statistics of the MC prediction error.

The best-known order –statistic filter is the median filter, which, as its name implies, replaces the value of a pixel by the median of the intensity levels in the neighborhood of that pixel. The value of the pixel is included in the computation of the median. Median filters are quite popular because, for certain types of random noise, they provide excellent noise reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise. The median filter yields excellent results for images corrupted by this type of noise [16-20].

III. PROPOSED SYSTEM

The adjacent reference frames are shifted with the estimated motion vectors to generate the interpolated frame. For scenes with objects which are moving smoothly, the forward and backward predicted motion vectors should be coherent, and the interpolated frames are identical. Unfortunately, this may not always be the case in practice and the estimated motion vector field may overlap with each other or not fully covered whole scene. This phenomenon is especially obvious if unidirectional motion estimation is used for FRUC. Because the forward and backward motion vectors are not always identical, the interpolated frames may be different. Which pixels in the previous and backward interpolated frames are more reliable is an open problem to be investigated is shown in fig 3.

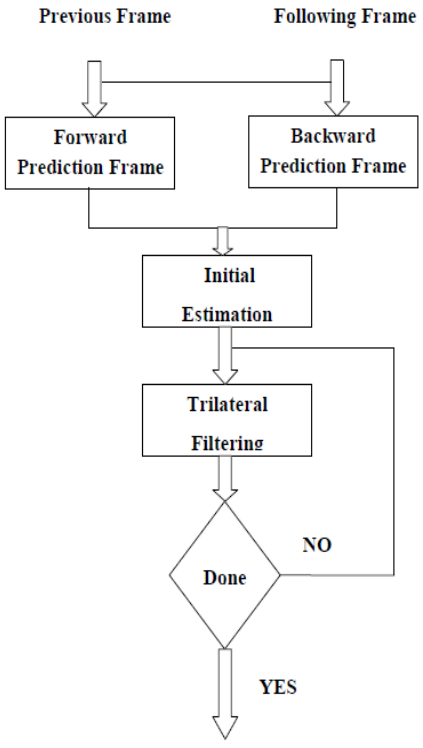

Fig 3 Block Diagram of Frame Rate Up-Conversion Using Trilateral Filtering

A standard movie, which is also known as motion picture, can be defined as a sequence of several scenes. A scene is then defined as a sequence of several seconds of motion recorded without interruption. A scene usually has at least three seconds. A movie in the cinema is shown as a sequence of still pictures, at a rate of 24 frames per second. Similarly, a TV broadcast consists of a transmission of 30 frames per second (NTSC, and some flavors of PAL, such as PAL-M), 25 frames per second (PAL, SECAM) or anything from 5 to 30 frames per second for typical videos in the Internet. The name motion picture comes from the fact that a video, once encoded, is nothing but a sequence of still pictures that are shown at a reasonably high frequency. That gives the viewer the illusion that it is in fact a continuous animation. Each frame is shown for one small fraction of a second, more precisely 1/ k seconds, where k is the number of frames per second. Coming back to the definition of a scene, where the frames are captured without interruption, one can expect consecutive frames to be quite similar to one another, as very little time is allowed until the next frame is to be captured. With all this in mind we can finally conclude that each scene is composed of at least 3 × k frames (since a scene is at least 3 seconds long). In the NTSC case, for example, that means that a movie is composed of a sequence of various segments (scenes) each of which has at least 90 frames similar to one another.

The sequence of five consecutive frames in a l0fps-video stream of the standard “mother and daughter” reference video (presents other reference
Videos). Changes from frame 1 to frame 2 is almost negligible; only changes in frames 3 through 5 are in fact noticeable due to the hand movement. Do notice, however, that while the hand of the mother moves, the rest of each frame is quite similar to its previous frame. Such similarity between neighbor frames is known as “temporal redundancy”, while the similarity of pixels in a given frame is known as “spatial redundancy”. Motion estimation is a technique that is often used to exploit the temporal redundancy described above. Before going further with details on motion estimation we need to describe briefly how a video sequence is organized. As mentioned earlier a video is composed of a number of pictures. Each picture is composed of a number of pixels or pels (picture elements). A video frame has its pixels grouped in 8×8 blocks. The blocks are then grouped in macroblocks(MB), which are composed of 4 luminance blocks each (plus equivalent chrominance blocks). Macroblocks are then organized in “groups of blocks” (GOBs) which are grouped in pictures (or in layers and then pictures). Pictures are further grouped in scenes, as described above, and we can consider scenes grouped as movies. Motion estimation is often performed in the macro block domain. For simplicity’ sake we’ll refer to the macro blocks as blocks, but we shall remember that most often the macro block domain is the one in use for motion estimation.

For motion estimation the idea is that one block b of a current frame C is sought for in a previous (or future)

frame R. If a block of pixels which is similar enough to block b is found in R then instead of transmitting the whole block just a "motion vector" is transmitted. A couple of neighbor frames (A and B), being the currently encoded frame while A is a previous reference frame. The block that contains the top-left portion of the star if sought for in frame A can be found shifted as much as denoted by the arrow which represents the motion vector. All is summarized in D. So, instead of encoding the pixels of the current blocks of the frame B one can simply send the motion vector. Ideally, a given macro block would be sought for in the whole reference frame; however, due to the computational complexity of the motion estimation stage the search is usually limited to a pre-defined region around the macro block. Most often such region includes 15 or 7 pixels to all four directions in a given reference frame. The search region is often denoted by the interval [-p, p] meaning that it includes p pixels in all directions.

C. Two-Stage Video Compression Model

The video compression model is a two-stage procedure. The first procedure consists of taking advantage of the temporal redundancy followed by a procedure similar to that used for lossy image compression which aims at exploring the spatial redundancy.. In the temporal redundancy exploitation stage (A) we have motion estimation of the current frame (**C**) using the reference frame (**R**). The first stage produces both a set of motion vectors (**i, j**) as well as difference macro blocks (**C-R**). The difference macro blocks then go through the second stage which exploits spatial redundancy. One may notice that the difference frame has usually very high spatial redundancy due to the fact that it only stores information of difference of motion estimated macro blocks as well as macro blocks where a good match is not found in the reference frame(s). The matching criteria are detailed in the next section. The reference to C-R is a didactic reference to what in fact would be each macro block of C minus the equivalent match block in R. It is also worth mentioning that in the image shown as C-R was applied the negative so that pixels whose difference is zero show up white rather than black, which would be tougher to visualize.

D. Matching Criteria

Let x, y denote the location of the current macro block. The pixels of the current macro block can then be denoted by C(x+k, y+l) while the pixels in the reference frame can be denoted as R(x+i, y+j). We can now define a cost function based on the Mean Absolute Error (MAE) or Mean Absolute Difference (MAD) as where M , N denotes the size of the macro block, -p = i = p and – p = j = p , with [-p,p] being the search region as described above. The matching block will be R (x+i,y+j) for which MAE is minimized, henceforth i, j defines the motion vector.

It is important to notice that the MAE is not the only option for matching criteria, as one can use Mean Square Error and other expressions as well. MAE is, however, often selected due to its computational simplicity. It is also worth mentioning that when the minimum MAE has a value higher than a given threshold the block is said "not found". That means that the motion estimation failed and that block is to be encoded without exploiting temporal redundancy with regard to the R reference frame. We can observe this phenomena when there is a scene change or a new object is inserted in the scene between the reference frame and the current frame, or yet if motion goes beyond the search area [-p,p]. Motion estimation doesn't have to be performed in a single reference frame. In some cases bi-directional motion estimation is performed. That means that other than looking for a macro block in a previous reference frame R, the block is also sought for in a reference frame F in the future. That is very useful for the case where a new object is inserted into the scene, as it'll be found in a future reference frame F even tough not being present in a previous reference frame R. More generally, multiple frames in the past and future can be used by a motion estimator, but more often we either use a single frame in the past or one in the past and one in the future, as described herein.

E. Algorithms

The motion estimation is the most computational intensive procedure for standard video compression. To seek a match for a macro block in a [-p, p] search region leads to (2 p+ 1) 2 search locations, each of which requires M × N pixels to be compared, where M, N give the size of the source macro block. Considering F the number of reference frames being considered in the matching process, such procedure reaches a total of (2 p + 1) 2 × MN × F executions of the Mean Absolute Error expression:

$$|C(x + k, y + l) – R(x + i + k, y + j + l)|, \quad (1)$$

This involves 3 operations per pixel. Considering a standard 16×16 macro block with p =15 and F=l in a VGA-sized video stream (640×480 pixels) we have 40×30 = 1200 macro blocks per frame. That leads to a grand total of 778.41M operations for a single frame. In a 30-fps video stream that would entail 23.3523G operations per second for motion estimation alone.

If bi-directional motion estimation is in place we'd reach 46.7046G operations per second the scheme described herein is refereed to as exhaustive search. If a video needs not to be processed in real time it may be the best option as the exhaustive search ensures that we'll find the best match for every macroblock. In some cases, however, we won't be able to afford an exhaustive search. Some algorithms have been developed aiming at finding a suboptimal match in much less time than the exhaustive search.

One example of algorithm that provides a sub-optimal result in much less operations is the two-dimensional logarithm, which resembles binary search. The basic idea is to divide the search region in two areas at each step: the first is inside of the [- p /2, p /2] area, whilst the latter is outside of such. We select nine positions: (0,0) as well as the four corners and four medium points of the limit area (left – marked as 1). The MAE is then calculated for each

of the eight perimeter values and the smaller MAE is selected as the starting point for the next pass. We then go on selecting a [- p /4, p /4] region around the selected position and repeat the process until we search in a one-pixel region around the previously selected pixel The last MAE selected will be the ending point of the motion vector.

F. Sub-pixel Motion Estimation

Some video encoders allow the motion estimation to go one step further than the whole-pixel-domain motion estimation, allowing sub-pixel motion estimation. Half-pixel motion estimation, for instance, is found in video codecs such as H263. Sub-pixel motion estimation consists of creating imaginary pixels between existing pixels as some sort of average of the neighbor pixels. That extra level of pixels would allow for a motion vector which stops in between real pixels. That allows better matching in the event that an object didn't move in the whole-pixel domain. The (right) the four extra positions between pixels, which are to be interpolated and also considered in an extra last-step in the motion estimation process.

G. Detection of Camera Motion, Motion Detection and Object Tracking

Motion Estimation and the resulting motion vectors can be used for a number of scene analyses through which it is possible to detect what happens in the video sequence. For instance, let us consider a video scene where the camera is panning right. If we consider a couple of consecutive frames we can verify that as the camera pans right the objects in the scene move leftwards until they eventually disappear at the left edge of the video frame. Considering macro blocks which compose the frame we can then deduct that most of the motion vectors should be pointing leftwards, as that's where the objects are moving to in the video sequence. If there is no motion in the scene but the camera is moving, the motion vectors will be basically aligned and with similar intensity and direction than neighbor vectors. For some scene where we have both motions in the scene and the camera, we would also have a large number of motion vectors pointing in the opposite direction to where the camera is moving, at least in the macroblocks that compose the background. We can, hence, analyze the orientation of the motion vectors of a given frame (or a given set of consecutive frames) to identify camera movements. Scene change, i.e. an indoor scene change to an outdoor scenario for instance, can be detected through the failure to estimate motion from one frame to the next, as typically no matches should be easily found.

When we put all the camera movement/scene change detection together, we can build a comprehensive set of procedures for classification/analysis of video sequences based purely in motion estimation, as well as its resulting motion vectors. Going one step further, if we have a scene which doesn't change much, for instance consider a surveillance camera which captures a safe lock inside a bank, the motion estimation should overall find excellent matches right at the same position where a given macro block was located in the previous frames. If someone approaches the safe there will be a group of macro blocks which will have some motion vectors pointing in the opposite direction as that individual moves. That could be used to trigger a security alarm. In order to prevent the alarm from being trigged if a small animal (a rat perhaps) moves around in the room, one can model the system to only trigger the alarm if the amount of macro blocks with movement detected is larger than a given threshold which would be typical of a human being. Going yet another step further, in the surveillance application described above, the macro blocks which compose the movement of an object in the room delimits the format of such object. Analyzing such format may allow the identification of the object.

H. 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.

It is possible to shift a block by a non-integer number of pixels, which is called sub pixel precision. The in-between pixels are generated by interpolating neighboring pixels. Commonly, half-pixel or quarter pixel precision (Qpel, used by H.264 and MPEG-4/ASP) is used. The computational expense of sub-pixel precision is much higher due to the extra processing required for interpolation and on the encoder side, a much greater number of potential source blocks to be evaluated.

I.Variableblock-size motion compensation

Variable block-size motion compensation (VBSMC) is the use of BMC with the ability for the encoder to dynamically select the size of the blocks. When coding video, the use of larger blocks can reduce the number of bits needed to represent the motion vectors, while the use of smaller blocks can result in a smaller amount of prediction residual information to encode. Older designs such as H.261 and MPEG-1 video typically use a fixed block size, while newer ones such as H.263, MPEG-4 Part 2, H.264/MPEG-4 AVC, and VC- 1 give the encoder the ability to dynamically choose what block size will be used to represent the motion.

J.Overlapped block motion compensation

Overlapped block motion compensation (OBMC) is a good solution to these problems because it not only increases prediction accuracy but also avoids blocking artifacts. When using OBMC, blocks are typically twice as big in each dimension and overlap quadrant-wise with all 8 neighboring blocks. Thus, each pixel belongs to 4 blocks. In such a scheme, there are 4 predictions for each pixel which are summed up to a weighted mean. For this purpose, blocks are associated with a window function that has the property that the sum of 4 overlapped windows is equal to 1 everywhere. Studies of methods for reducing the complexity of OBMC have shown that the contribution to the window function is smallest for the diagonally-adjacent block. Reducing the weight for this contribution to zero and increasing the other weights by an equal amount leads to a substantial reduction in complexity without a large penalty in quality. In such a scheme, each pixel then belongs to 3 blocks rather than 4, and rather than using 8 neighboring blocks, only 4 are used for each block to be compensated. Such a scheme is found in the H.263 Annex F Advanced Prediction mode.

1V.PROPOSED ALGORITHM (BLOCK MATCHING ALGORITHM)

K. Three Step Search

Three Step Search (TSS) is one of the first non full search algorithm which use three steps. It is mainly used for real time video compression with low bit rate video application such as video conferencing and videophone.

Fig. 4 illustrates an example of TSS algorithm. The search starts with a step size equal to or slightly larger than half of the maximum search range. In each step, nine search points are compared. They consist of the central point of the square search and eight search points located on the search area boundaries as shown in Fig. 4.1. The step size is reduced by half after each step, and the search ends with the step size of one pel. This search proceeds by moving the search area center to the best matching point in the previous step. We need three search steps for a maximum search range between 8 and 15 pels.

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
6									2		2		2
5													
4			1				1		2		1		2
3								3	3	3			
2								3	2	3	2		2
1								3	3	3			
0			1				0				1		
-1													
-2													
-3													
-4			1				1				1		
-5													
-6													

Fig 4: TSS Algorithm

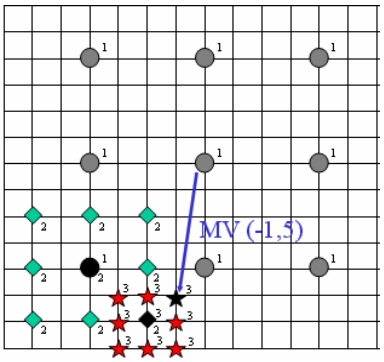

Fig 4.1:TSS Algorithm

L. DIAMOND SEARCH

The DS adopts two diamond-shaped search patterns [5], illustrated in Fig.4.2: large diamond search pattern (LDSP) with nine search points and small diamond search pattern (SDSP) with five search points. The LDSP is repeated until that it reaches the edge of the search window, or a new minimum matching distortion point occurs at the center of LDSP. The search pattern is then switched to SDSP, which is used to refine the search algorithm.

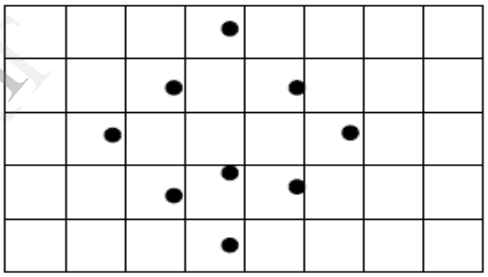

Fig 4.2: Diamond Search

M.CROSS DIAMOND SEARCH

Like the SDS, the Cross Diamond Search (CDS) is a fast motion search algorithm which uses one search pattern

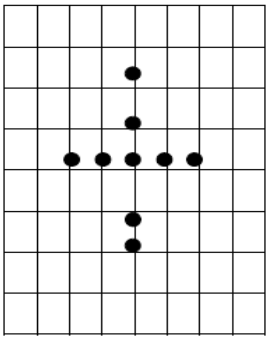

Fig 4.3: Example of CDS

This form is adapted for the fasts motions, but for the slow sequences SDS can give betters results. This algorithm can also be considered as derived from DS, the only difference it allows to improve research on the

horizontal and vertical component of the motion. Fig.4.3 illustrates an example of CDS path.

N.VARYING DIAMOND SEARCH ALGORITHM

The last search we implemented is Varying Diamond Search. This borrows ideas from the three step search, but instead of using eight positions it uses four.The option of a three step or a four step is allowed in this search as well. Figure 4.4 illustrates what a varying diamond search will look like when it uses three steps. The bold positions are those with the lowest cost, and the lower the number, the earlier it is searched.

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
6													
5													
4							1						
3													
2			2										
1													
0	2		1		2		0				1		
-1			3										
-2		3	2	3									
-3			3										
-4							1						
-5													
-6													

Fig4.4 : Varying Diamond Search

A best match is found by calculating the cost at each location. The spot with the lowest cost is chosen as the best match.

O.STEPS FOR CALCULATING SAD FOR FULL SEARCH ME ALGORITHM

Subtract 4*4 current image from 4*4 reference image and all the subtracted values which gives SAD(Sum of Absolute Difference).Hold the current block and move the reference block by pixel by pixel is shown in fig 4.5.

	-4	-3	-2	-1	0	1	2	3	4
4	74	75	76	77	78	79	80	81	50
3	73	44	45	46	47	48	49	26	51
2	72	43	22	23	24	25	10	27	52
1	71	42	21	8	9	2	11	28	53
0	70	41	20	7	1	3	12	29	54
-1	69	40	19	6	5	4	13	30	55
-2	68	39	18	17	16	15	14	31	56
-3	67	38	37	36	35	34	33	32	57
-4	66	65	64	63	62	61	60	59	58

Fig 4.5: FSS ME Algorithm

P.2D-LOG SEARCH ALGORITHM

To determine the distance from the center in which the search is started d=floor (2*log(s-1)) is Computed, where‘s’ is the search Size. The cost is then calculated at the center (0) and four other positions (1) surrounding it. The Position with least cost (1) is chosen and‘d’ remains the same. Next, the cost is computed at the three surrounding positions (2) and if their cost is not lower than the center then d is halved. If d is an odd number, 1 is subtracted from it and then it is halved is shown in fig 4.6.

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
6													
5													
4													
3							1						
2													
1													
0				1			0			1			
-1													
-2							3	4					
-3				2		3	1	3	4	2			
-4							3	4					
-5													
-6							2						

Fig 4.6: 2D-LOG SEARCH

Q.FAST MOTION ESTIMATION ALGORITHM

In this method, calculate the cost of each position in a diamond shape. The size of the diamond grows or shrinks based on the search Size. The below figure 4.7 shows when the search size of 4 is used. Also the cost for the center is calculated before this search starts and if no cost in the fast search is lower than the center, the center is chosen

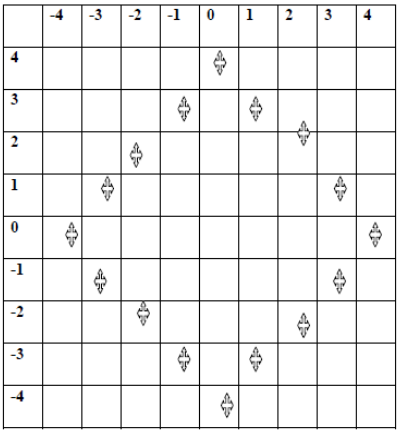

Fig 4.7: Fast motion estimation

V.PERFORMANCE ANALYSIS

S.NO	ALGORITHM NAME	TIME IN SEC
1	FULL SEARCH ME	12.04
2	CROSS SEARCH(FAST MOTION TECHNIQUE)	4.39
3	DIAMOND SEARCH	6.95
4	LOG SEARCH	12.28
5	THREE STEP SEARCH	7.782

Table 1.Performance & Analysis of PSNR Parameter

The phrase peak signal-to-noise ratio, often abbreviated 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 representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. The PSNR is most commonly used as a measure of quality of reconstruction of lossy compression codecs (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs it is used as an approximation to human perception of reconstruction quality, therefore in some cases one reconstruction may appear to be closer to the original than another, even though it has a lower PSNR (a higher PSNR would normally indicate that the reconstruction is of higher quality). One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content.

The most traditional ways of evaluating quality of digital video processing system (e.g. video codec like DivX, Xvid) are calculation of the signal-to-noise ratio (SNR) and peak signal-to-noise ratio (PSNR) between the original video signal and signal passed through this system. PSNR is the most widely used objective video quality metric. However, PSNR values do not perfectly correlate with a perceived visual quality due to the non-linear behavior of the human visual system.

The quality of the proposed algorithm evaluated in terms of PSNR. The proposed algorithm improves the quality of the interpolated video. The quality measure PSNR is computed between original frame and interpolated frame. The PSNR curves of the proposed FRUC algorithm are compared with those bench mark algorithm. The proposed algorithm obtained notable PSNR improvement from the best bench mark algorithm. The proposed algorithm achieves above 3.2dB improvement, which mainly comes from the noise suppression of the proposed trilateral filter for complicated and fast moving scenes. The proposed algorithm obtains 3-3.5dB improvement which exceeds the 2dB improvement of the bilateral FRUC method. Compared with existing FRUC algorithm the proposed algorithm achieve more than 3dB PSNR improvement is shown in table.1

VI.SIMULATION RESULTS

CROSS SEARCH SIMULATION

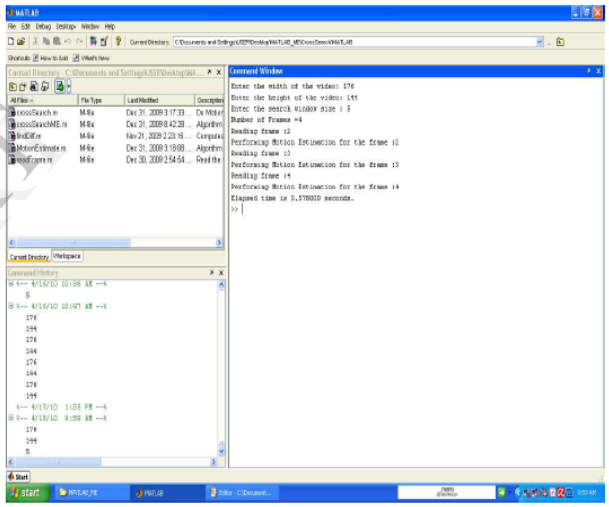

Fig:5 Cross Diamond Search

DIAMOND SEARCH SIMULATION

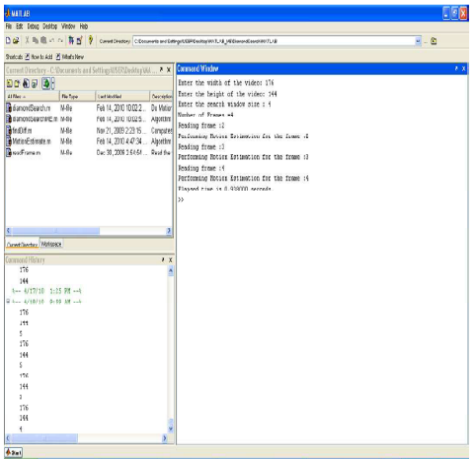

Fig 6.Cross Diamond Search

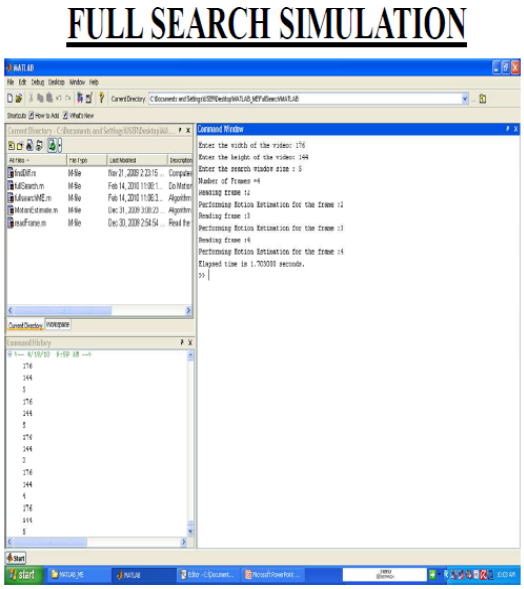

Fig 7.Full Diamond Search

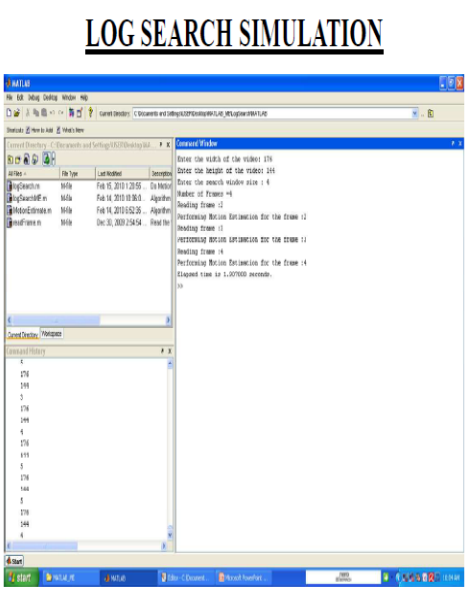

Fig 9.LogCross Diamond Search

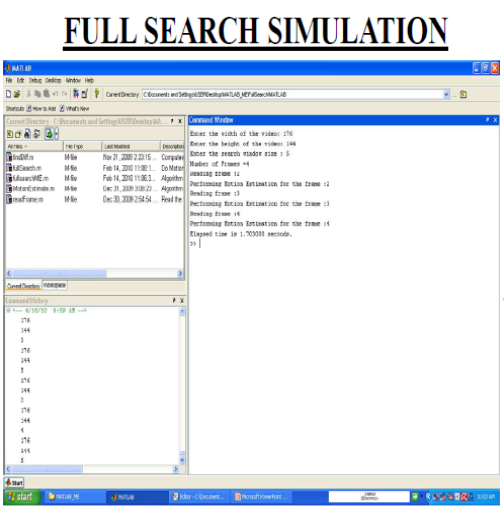

Fig 8.Three Steps Cross Diamond Search

Fig10. Bilateral Filter Input

Fig 11.Bilateral Filter Output

Fig 12.Trilateral Filter Output

Output was simulated using MATLAB shown in fig (5-11). A video clip contains a series of still images (frames). Hence, to compress a video clips we can directly apply still image compression on each of these frames. In fact, this is the technique used by the ―motion JPEG standard. Although modern vide-compression algorithms go beyond still-image compression techniques and take into account similarities between frames to achieve further compression, they also employ the techniques used in still-image compression Figure 10 shows the input of the video image the video image is a gray image or color image. It is of default size 576*768. For our convenience, the input image is resized to 512*512. The Figure 11 shows the Output image, the output image is compressed to the input image. The file size is decreased to the input image the size is about approximately 30,000.The Figure 12 shows that Trilateral filter, A Trilateral filter is a video filter applied to blocks in decoded video to improve visual quality and prediction performance by smoothing the sharp edges which can form between macro block.

VII. CONCLUSION

The Presented FRUC algorithm using trilateral filtering has simple structure and reduce the computational cost. The compared with existing FRUC algorithms, the proposed algorithm achieves more than 3dB PSNR improvement. It also utilizes the unidirectional motion estimation, rather than bilateral motion estimation and it does not required multistage block motion estimation or block decomposition, which is with high computational cost. The proposed FRUC is mainly utilized to minimize the complexity which occurs in complex structures. Moreover the proposed work has maximum results in all video images. Finally, comparing the time complexity, the full spiral search has a time complexity of 1.5 where as the time required for proposed method is 0.5.Hence the future time complexity can be avoided by optimizing the filtering operations.

VIII. REFERENCES

[1] Ci Wang, Lei Zhang, Yuwen He, and Yap-Peng Tan, "*Frame Rate Up-Conversion Using Trilateral Filtering*", IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 6, Jun 2010.

[2] Chalidabhongse J. and Kuo C. C. J., "*Fast motion vector estimation using multiresolution-spatio-temporal correlations*," IEEE Trans. Circuits Syst. Video Technol., vol. 7, no. 3, pp. 477–488, Jun. 1997.

[3] Ghanbari M, "*The cross-search algorithm for motion estimation*,"IEEE Trans Commun., vol. 38, no. 7, pp. 950–953, Jul. 1990.

[4] Hsieh C. H. , Lu P. C., Shyn J. S., and Lu E. H, "*Motion estimation algorithm using interblock correlation*," Electron. Lett., vol. 26, no. 5, pp. 276–277, Mar. 1990.

[5] Huang E. H, Chen C.-Y, Tsai C.-H., Shen C.-F., and Chen L.-G.,"*Survey on block matching motion estimation algorithms and architectureswith new results*," J.VLSI Signal Process., vol. 42, no. 3, pp.297–320, Mar. 2006.

[6] Jain J. R. and Jain A. K., "*Displacement Measurement and Its Applicationin Interframe Image Coding*," IEEE Trans. Commun., vol. 29, no.12, pp. 1799–1808, Dec. 1981.

[7] Koga T. Iinuma K, Hirano A, Iijima Y, and Ishiguro T, "*Motion compensated interframe coding for video conferencing*," in Proc Nat.Telecommun. Conf., 1981, pp. G5.3.1–G5.3.5.

[8] Lee C.-H. and L.-H. Chen L.-H., "*A fast motion estimation algorithm based on the block sum pyramid*," IEEE Trans. Image Process., vol. 6, no. 11, pp. 1587–1591, Nov.1997.

[9] Li W.and Salari E, "*Successive elimination algorithm for motion estimation*," IEEE Trans. Image Process., vol. 4, no. 1, pp. 105–107, Jan. 1995.

[10] Lim K.P, Sullivan G, and Wiegand T, "*Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods*," Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG Doc. JVT-X101, Jul. 2007.

[11] Limb J.O, and Murphy J. A., "*H.264/AVC*" Reference Software ver. 11.1, Joint Video Team (JVT) of ISO MPEG & ITU-T VCEG [Online]. Available: http://iphom hhi.de/suehring/tml/, Aug 2006

[12] Liu B, and Zaccarin A, "*New fast algorithms for the estimation of block motion vectors*," IEEE Trans. Circuits Syst. Video Technol., vol. 3, no. 2, pp. 148–157, Apr.1993.

[13] Memon N.D.and K. Sayood K., "*Lossless compression of video sequences*," IEEE Trans. Commun., vol. 44, no. 10, pp. 1340–1345, Oct. 199

[14] Po L.M., and Ma W. C., "*A novel four-step search algorithm for fast block motion estimation*," IEEE Trans. Circuits Syst. Video Technol., vol. 6,no. 3, pp. 313–317, Jun. 1996.

[15] Tham J. Y., Ranganath S., Ranganath M., and Kassim A. A., "*A novelunrestricted center-biased diamond search algorithm for block motion estimation*," IEEE Trans Circuits Syst. Video Technol., vol. 8, no. 4, pp. 369–377, Apr. 1998.

[16] Tzovaras D., M. G. Strintzis M. G., and Sahinolou H., "*Evaluation of multiresolution block matching techniques for motion and disparity estimation*," Signal Process.: Image Commun., vol. 6, no. 1, pp. 59–67, Mar. 1994.

[17] Yang M., Cui H., and Tang K., "*Efficient tree structured motion estimation using successive elimination*," Proc. IEEE Vis., Image, Signal Process., vol. 151, no. 5, pp. 369–377, 2004.

[18] Zaccarin A.and Liu B., "*Fast algorithms for block motion estimation*," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 1992, vol. 3, pp. 449–452.

[19] Zhang M.F., Hu J, and Zhang L.M., "*Lossless video compression using combination of temporal and spatial prediction*," in Proc. IEEE Int. Conf. Neural Netw. Signal Process., Dec. 2003, vol. 2, pp. 1193–1196.

[20] Zhu S., and Ma K.K., "*A new diamond search algorithm for fast block matching motion estimation*," in Proc. IEEE Int. Conf. Inf., Commun., Signal Process., 1997, pp. 292–29.