Analysis of Block Matching Motion Estimation Algorithms

Graham Ball | V00732898 | ELEC 483 Final Project Report | Report Submitted:

Abstract

The performance of three Block-Matching Motion Estimation techniques are analyzed and compared. Using MATLAB, Block Matching Algorithms employing the conventional Mean Absolute difference for error measure are coded. The performance of algorithms is compared when using an exhaustive search and a 3-Step search [6] and when employing a multiresolution approach in the form of Hierarchical Matching algorithm. Two methods Hierarchical Matching algorithms are used. PSNR, execution time, predicted and error images are compared when a simple anchor and target frame are processed.

Introduction

Motion Estimation is a critical component used to achieve high compression of video sequences. Motion estimation also plays roles in computer vision techniques. A high diversity of applications has led to the development of multiple techniques and algorithms. Common techniques include Optical flow, Pixel-Based, Block-matching algorithms, Mesh-Based, Global and Regional-Based, and Multiresolution techniques [1].

The concept of motion estimation is used in video compression in the form of "inter prediction". Inter prediction is the use of previous frames to predict future frames and enables the encoder to code only the prediction error to the compressed video stream [1]. In this manner motion estimation reduces the redundancy of a video sequence in the time domain whereas DCT and quantization compression components reduce the sequence spatially [3]. Block matching motion estimation (BMME) techniques are used in several common video compression standards including MPEG-1, MPEG-2, MPEG-3, H.261, H.263, and H.264 [2]. BMME uses a simplistic approach to calculate motion vectors from base and target frames. Pixels within frames are processed into blocks. Search algorithms are employed to find the best matching block in a target from the anchor frame. Potential matches are commonly ranked using a sum of absolute differences (SAD) or a mean absolute difference calculation (MAD) [1][3].

The computational load of motion estimation algorithms (in particular BMME) is its greatest drawback, as such, many variations of the general algorithm have been developed to improve speed (usually at the cost of some accuracy) [4]. The computational load of matching mechanisms can be reduced by altering the method by which potential matches are evaluated (exploiting block features and attributes [2]) and/or by altering the search algorithms to reduce the number of potential matches (fractional accuracy searches and fast algorithm techniques [1]). In addition to the methods mentioned, BMME speeds can be further improved through the application of multiresolution algorithms such as the Hierarchical Block Matching Algorithm (HBMA).

Theory and Analysis of the solution

Block Matching Motion Estimation algorithms' greatest drawback is their computational load or the time it takes for them to execute. The algorithm must look at each block in the anchor image and evaluate individual pixels in every candidate blocks in the target frame, calculating a motion estimation vector when it finds a suitable match (smallest difference or error).

A multitude of block matching algorithms and variations exist. In order to analyze multiple algorithms a base or standard algorithm had to be selected against which to compare others. The Exhaustive block matching algorithm or full search algorithm assumes the simplest case, that motion in each block is constant (block-wise translational model [1]). This algorithm searches all candidates within a defined search window computing the error for each one and selecting the best fit, therefore, it provides the optimum result for the defined window [5].

Fast search algorithms, as their name implies, reduce the computational load of BMME, however, this comes at a cost. Fast BMAs don't search all candidates within the search range but instead attempt to predict the general location of the best matching block within the defined window and refine the search to this region, thus producing a sub optimal result.

The 3-Step search algorithm [6] is a simple algorithm that searches for the best match in a "coarse-to-fine" [5] pattern. As the name implies, the algorithm employs 3-steps in order to find the best matching candidate. In each step 8 points dispersed evenly around the search range are compared to the center point, upon selecting the best match the center is moved to said point, the search range is reduced, and the process repeated. The search range in the first step is usually equal to or slightly larger than the defined search window, the proceeding step is roughly half that of the initial and the search ends with 1-pel accuracy (Figure 1).

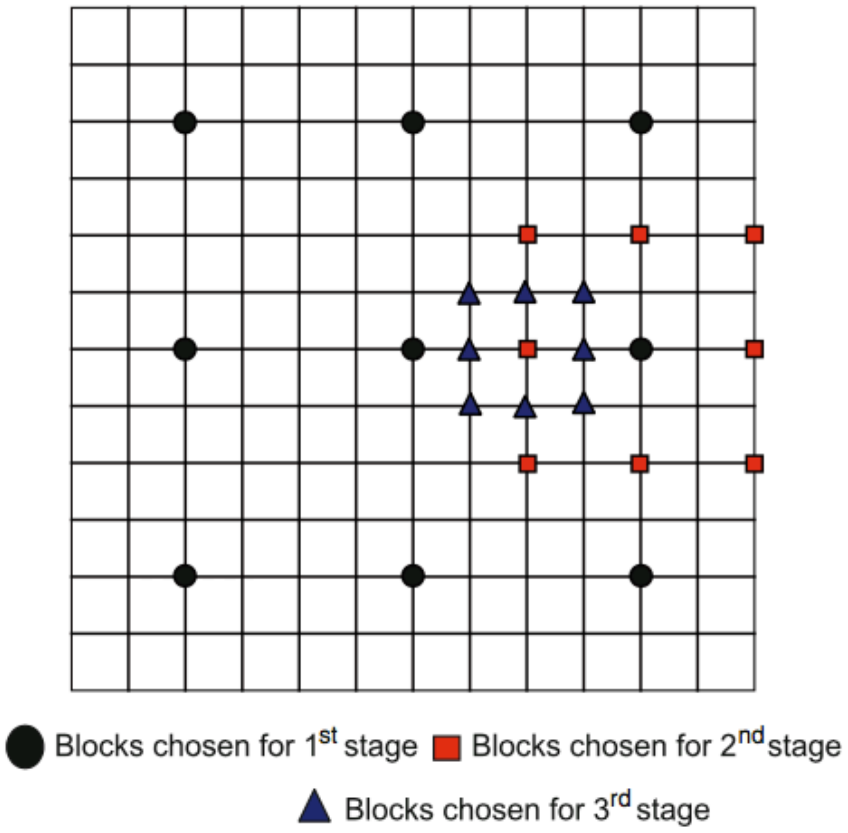

Figure 1 – Search process for a 3-Step search algorithm as described by Chakrabarti et. al [5].

A multiresolution algorithm such as the HBMA can be applied alongside the EBMA or any fast algorithm to further reduce the computational load of the BMME. Similar to the 3-Step the HBMA uses a "coarse-to-fine" approach to find the best matching candidate block. Unlike the 3-step algorithm or other fast BMAs the HBMA changes the resolution of the anchor and target frames and achieves the motion vector by employing a Gaussian pyramid (Figure 2). The base level of the pyramid(s) ($\psi_{1,3}$ $\psi_{2,3}$ in Figure 2) is the original frame(s) and each proceeding level is half the resolution of that which precedes it. The search starts with the highest level and a search range of $R/2^{L-1}$, where R is the desired search range [1]. A motion vector is returned at each level, each level uses the previous levels motion vector as a center point for the search range. As the resolution increases the motion vectors are refined.

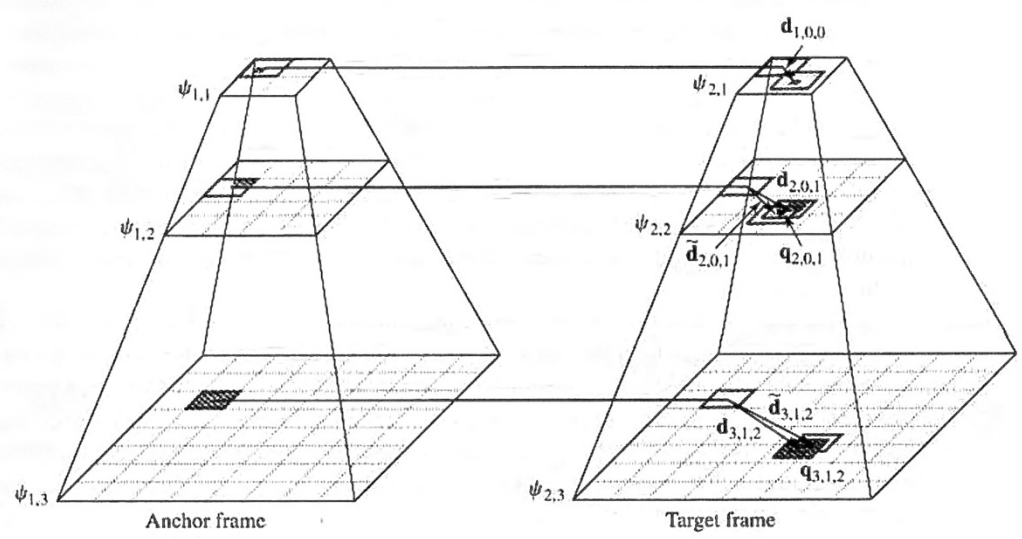

Figure 2 – Illustration of the HBMA from Wang et. al [1].

Algorithm Tradeoffs

The tradeoffs of each BMME technique must be considered when selecting the best fit for an application. For instance, if a video sequence is being compressed during a video conference call EBMA may yield the optimal result but video lag may be so significant that the conversation is impacted. General tradeoffs of the three techniques are summarized in Table 1.

Table 1 – Summary of general tradeoffs of selected Block-Matching Algorithms

Technique	PROS	CONS
EBMA	• Ease of Implementation • Optimum result	• Computational load
3-Step	• Ease of Implementation • Reduced computational load	• Sub optimal result
EBMA with HBMA	• Reduced computational load	• Sub optimal result • Difficult to implement

EBMA computational load

For an MxM image and block of size N, and a search range of R the computational load (every MAD calculation) is represented by formula 1.1 [1].

$M^2(2R+1)^2$ (1.1)

Note that the number of operations is independent of the block size N.

Sample Calculation: Using a search range of 8 and an image of size 32x32, Cost = 295 936 Operations

3-Step search algorithm computational load

For an MxM image and block of size N, and a search range of R the computational load (every MAD calculation) is represented by formula 1.2.

$M^2(8+7+7)$ (1.2)

Note that the number of operations is now independent of the block size and the search range as each block when using the 3-Step algorithm has 8 candidate blocks at the first step and 8-1 candidates for the second and third.

Sample Calculation: Using a search range of 8 and an image of size 32x32, Cost = 22 528 Operations

EBMA with HBMA computational load

When implemented with HBMA the number of operations for an EBMA is now dependent on the number of levels L [1].

$\sum_{l=1}^{L} M_l^2(2R_l+1)^2$ (1.3)

Sample Calculation: Using a search window of 8 and an image of size 32x32 we assign the following search ranges and image resolutions:

$M_1 = 32, R_1 = \frac{8}{4} = 2$

$M_2 = 16, R_2 = \frac{8}{4} = 2$

$M_3 = 8, R_3 = \frac{8}{4} = 2$

Cost = (25 600) + (6 400) + (1600) = 33 600 Operations

(The above cost is not entirely representative of the true functionality of the algorithm. This will be further explained in the discussion section of this paper)

Implementation

All algorithms were constructed using MATLAB 2017. Initial tests were conducted using the anchor and target frames train01.tif and train02.tif.

Searching Algorithms Implementation

EBMA

The Exhaustive Block Matching algorithm was constructed according to the following [1] pseudocode:

```
For i =1:N:height-N,
        For j=1:N:width-N % for every block
                MAD_min =1000000; mvx = 0; mvy =0;
                For k = -R:1:R
                        For l = -R:1:R % for every candidate in the search range
                                MAD =  = sum(sum(abs(anchor_frame((i:i+N-1,j:j+N-1))- ….
                                …. target_frame(i+k:i+k+N-1, j+l:j+l+N-1))));
                                % Calculate the MAD for the candidate
                                if MAD<MAD_min
                                        MAD_min = MAD; dy = k; dx = l;
                                End;
                        End;
                End;
                Predicted_frame(i:i+N-1, j:j+N-1) = target_frame(i+dy:i+dy+N-1, j+dx:j+dx+N-1);
                Iblk=floor((i-1)/N+1); jblk = floor((j-1)/N+1); % Block index
        Mvx(iblk,jblk)=dx; mvy(iblk,jblk)=dy;
        End;
End;
```

3-Step search

The 3-Step Search BMA was constructed according to the following pseudocode:

```
For Step =1:3,
        R = R/2^Step
        If Step == 3, R=1; % Ensure that the last step is always 1-pel accuracy
        For i =1:N:height-N,
                For j=1:N:width-N % for every block
                        MAD_min =1000000; mvx = 0; mvy =0;
                        For k = -R:R:R
                                For l = -R:R:R % for 8-1 candidate in the search range
        If k==0 & l==0 & Step !=1, end;
        % Don't compare center point on steps 2 and 3
                                        else
                                        MAD =  = sum(sum(abs(anchor_frame((i:i+N-1,j:j+N-1))-
                                        …. target_frame(i+k:i+k+N-1, j+l:j+l+N-1))));
                                        % Calculate the MAD for the candidate
                                        if MAD<MAD_min
                                                MAD_min = MAD; dy = k; dx = l;
                                        End;
                                End;
                        End;
                        Predicted_frame(i:i+N-1, j:j+N-1) = ….
        target_frame(i+dy:i+dy+N-1, j+dx:j+dx+N-1);
                        Iblk=floor((i-1)/N+1); jblk = floor((j-1)/N+1); % Block index
        Mvx(iblk,jblk)=dx; mvy(iblk,jblk)=dy;
                End;
        End;
End;
```

Multi Resolution Algorithms

A hierarchical block matching algorithm was applied employing both of the search algorithms described above. Multiresolution algorithms are beneficial in this way, they are capable of increasing the speed of Motion estimation algorithms regardless of what type of search algorithm

or match distortion calculation is being used. One downside of the hierarchical algorithm is the ease of implementation. The approach taken required alteration of the search algorithms from their original format. Inputs for previous motion vectors were required in order to shift the search range for each level of the pyramid.

The produced HBMA algorithms used pyramids with 3 levels of resolution. Creation of the pyramid itself was done by trial and error, through this approach 3 methods of pyramid creation were explored.

Method 1: Step-size adjustment

In the first method, the anchor and target frame are not adjusted, instead the step-size of the search algorithms were adjusted as well as the search range. By setting the step-size to 2^(L-l), L being the total number of levels and l being the current level, the search algorithms stepped through the candidates by multiples of 2. For example, if 3 levels were used the first search at l=1 (top level of the pyramid) would use a step-size of 4. The resolution of anchor and target images were therefore reduced spatially. This method proved difficult to implement with the 3-Step search algorithm as its implementation also relied on an adjustment of the step-size.

Method 2: Block-processed mean

The second method processed the anchor and target frames into lower resolutions by taking the mean of blocks of pixels. This method therefore required more computations than the first method but was much easier to implement and allowed for a seamless integration with the 3-Step search algorithm relative to the first.

Method 3: Built in Gaussian pyramid function

The final method investigated was by far the easiest as it made use of MATLABs built in function. Using the function ‘gaussPyramid()’, similarly to method 2 the anchor and target frames were processed into lower resolutions before implementing the search algorithms. The results of the third method are not included in this report but are comparable to the method 2 results.

Test Cases

Set target and anchor frames (Figure 3) were used to achieve a consistent standard of comparison for motion estimation algorithms. The predicted error image of the individual algorithms predicted

were calculated as the difference between predicted and anchor frames, algorithm PSNR values were then calculated using formula 2.1.

$$PSNR = 10log10(255 * 255/mean(mean(error\ image^2))) \quad (2.1)$$

The computational load of the algorithms was assumed to relate to the algorithms' execution times. The execution times of the individual algorithms were calculated using the built-in MATLAB stopwatch functions, 'tic' and 'toc'.

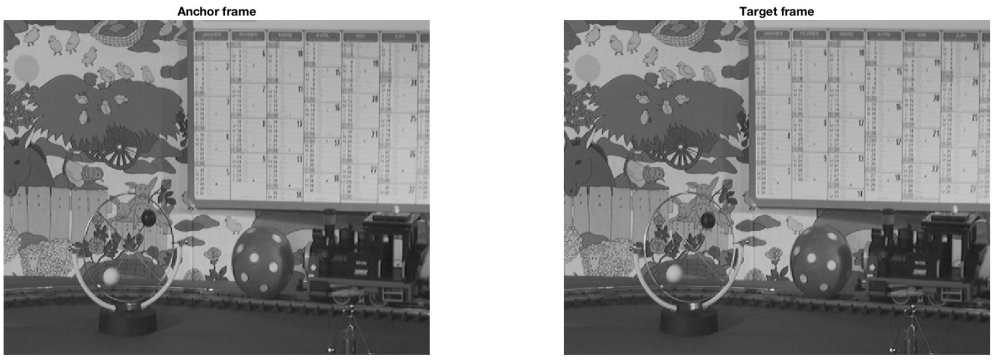

Figure 3 –Standard anchor (left) and target (right) frames used to compare motion estimation algorithms.

Examples

EBMA

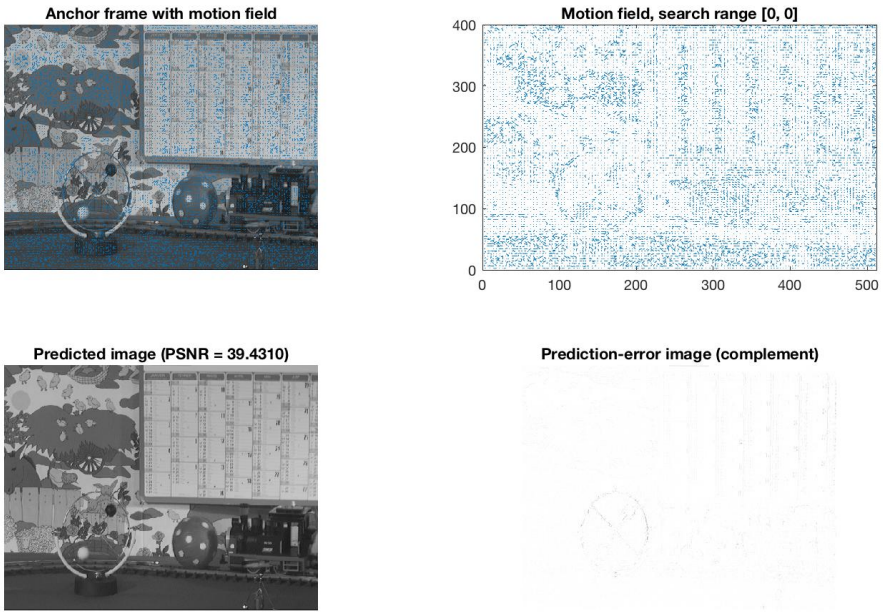

Figure 4 – Exhaustive Block Matching Algorithm results with search range = 8x8 and block size = 4x4. (Top Left) Anchor frame with overlaid motion field (Top Right) Motion field (Bottom Left) Predicted image produced with PSNR of 39.4310 (Bottom Right) Complemented prediction-error image.

3-Step search

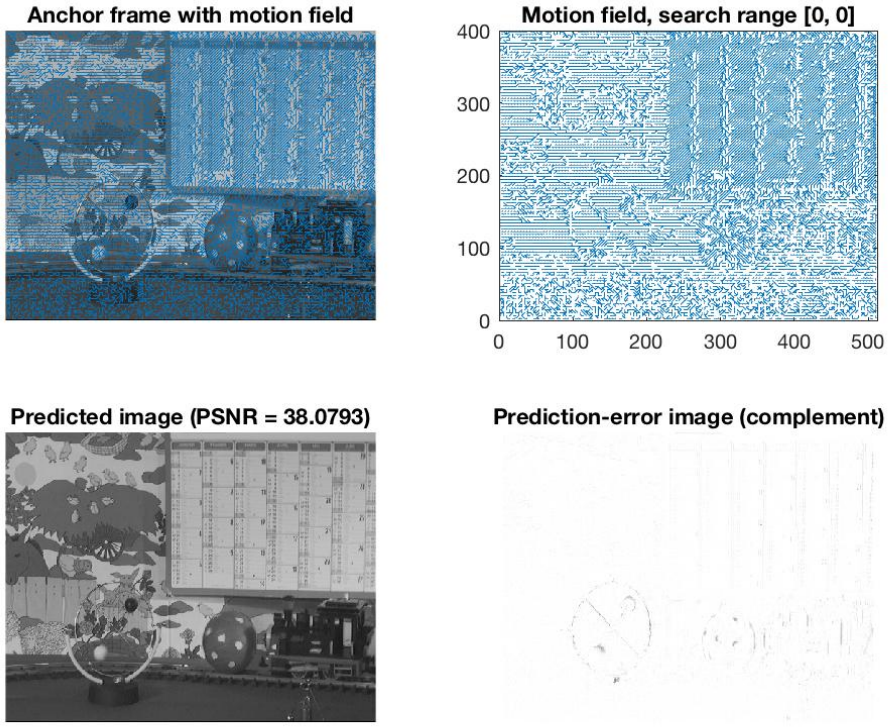

Figure 5 -3-Step Search Block Matching Algorithm results with search range = 8x8 and block size = 4x4. (Top Left) Anchor frame with overlaid motion field (Top Right) Motion field (Bottom Left) Predicted image produced with PSNR of 38.0793 (Bottom Right) Complemented prediction-error image

EBMA with HBMA

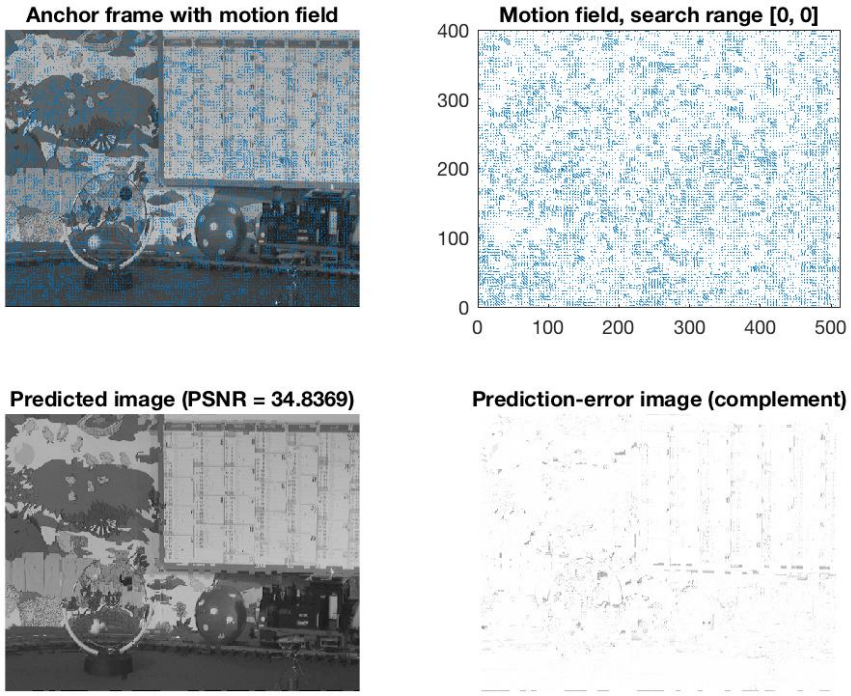

Figure 6 –Multiresolution Exhaustive Block Matching Algorithm results using Hierarchical Block Matching (method 1), search range = 8x8, and block size = 4x4. (Top Left) Anchor frame with overlaid motion field (Top Right) Motion field (Bottom Left) Predicted image produced with PSNR of 34.8369 (Bottom Right) Complemented prediction-error image.

EBMA with HBMA: Method 2

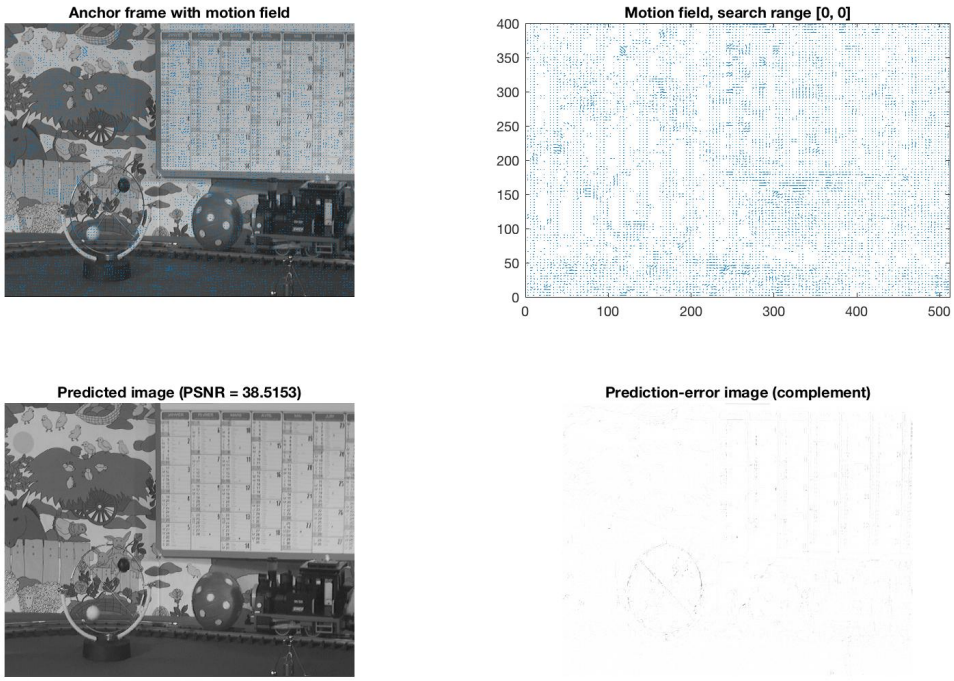

Figure 7 - Multiresolution Exhaustive Block Matching Algorithm results using Hierarchical Block Matching (method 2), search range = 8x8, and block size = 4x4. (Top Left) Anchor frame with overlaid motion field (Top Right) Motion field (Bottom Left) Predicted image produced with PSNR of 38.5153 (Bottom Right) Complemented prediction-error image.

3-Step search with HBMA: Method 2

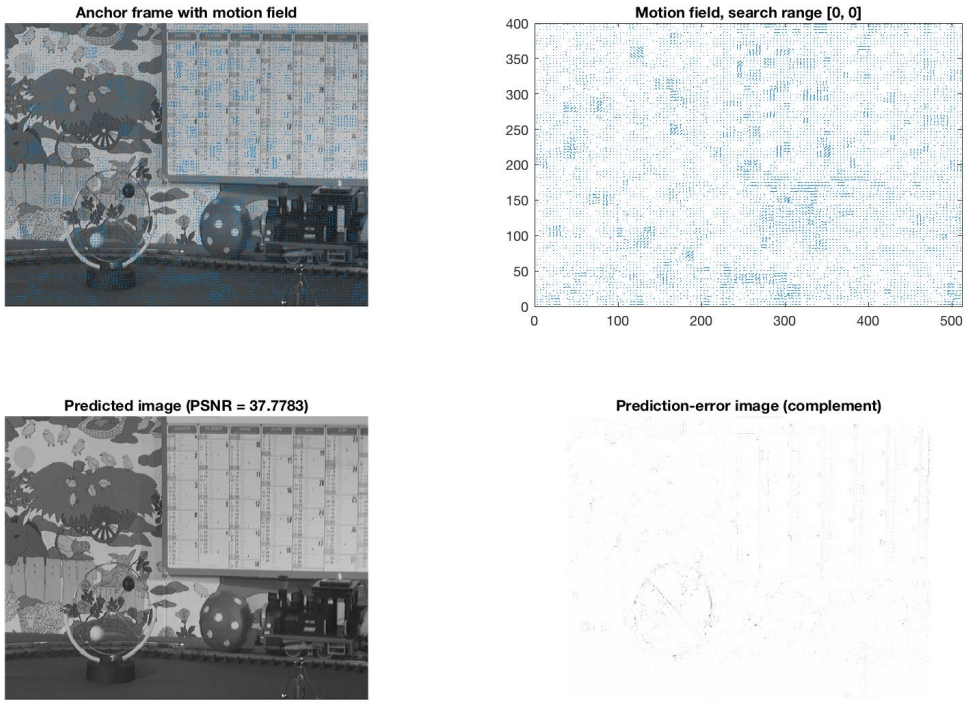

Figure 8 - Multiresolution 3-Step Search Block Matching Algorithm results using Hierarchical Block Matching (method 2), search range = 8x8, block size = 4x4, and 3 Levels. (Top Left) Anchor frame with overlaid motion field (Top Right) Motion field (Bottom Left) Predicted image produced with PSNR of 37.7783 (Bottom Right) Complemented prediction-error image.

Discussion

Table 2 – Summarized PSNR and execution time results for all implemented algorithms

Algorithm	EBMA	3-Step BMA	HBMA with EBMA (method 1)	HBMA with e EBMA (method 2)	HBMA with 3-Step BMA (method 2)
PSNR	39.431	38.079	34.837	38.515	37.778
Run Time (sec)	10.217	1.09	1.852	7.873	7.348

The results of 5 BMA implementations are presented in Figures 4-8 using an 8x8 search range and a 4x4 block-size. The algorithms' PSNR and execution times are summarized in Table 2. In terms of predicted image accuracy, the EBMA algorithm proved superior to all others with a PSNR value

of 39.431, however, the EBMA execution time was significantly longer. The 3-Step algorithm [6] produced the fastest result with a time of 1.08 seconds. These results coincide with the theories presented previously in this report.

The combination of search algorithms with multiresolution algorithms yielded varied results. Combining the EBMA with the HBMA implemented with method 1 (HBMA-1) (Figure 6) resulted in a drastically improved execution time (1.852) but the quality of the predicted image was the worst of all of the algorithms that were implemented (PSNR = 34.837). The EBMA using the HBMA implemented with method 2 (HBMA-2) (Figure 7) produced a predicted image of much higher quality but an execution time between that of the EBMA with and without HBMA-1. The increased run time is representative of the time it took to process the anchor and target frames into lower resolutions.

Implementing the 3-Step Search algorithm with HBMA-2 had a higher execution time and produced a lesser quality predicted frame then the implementation of the search algorithm on its own. The time difference can be easily explained as the number of operations used by the 3-Step search BMA is independent of the search region (proof in “theory and analysis” section of this report). The decreased quality of the image is related to problem resolved during trouble shooting while creating the algorithms.

The Hierarchical Block Matching Algorithm as described by Wang et. al [1] used a fixed search range of R/(2^L-1) for all levels of the resolution (Gaussian) pyramid. During the initial testing of the algorithm significant distortion throughout the entire predicted frame was noted when using this fixed search range (Figure 9). It was discovered that this was the result of the inability of the algorithm to correct for errors made in the initial prediction of motion vectors in the lowest resolution level. For example, if a group of pixels within a block showed no movement but were grouped into a block at a lower resolution that had an average movement of more than half of the search range the pixels with no movement would not be able to return to their initial locations. This error was fixed using an adaptive search range that grew with each level of resolution (Figure 10). In the case of the 3-Step search algorithm the adaptive search range was not quite significant enough to account for the larger steps made.

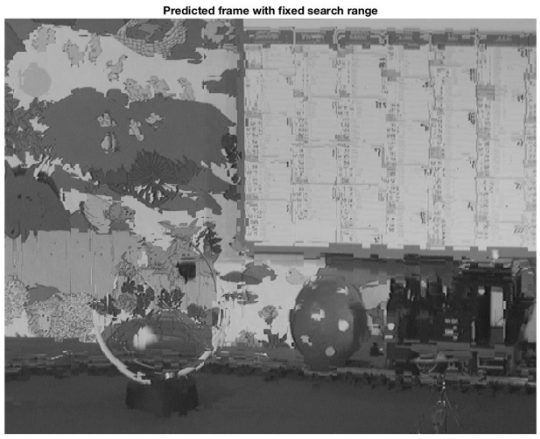

Figure 9 – Multiresolution Exhaustive Block Matching Algorithm results using Hierarchical Block Matching (method 1) using a fixed search range at all resolutions.

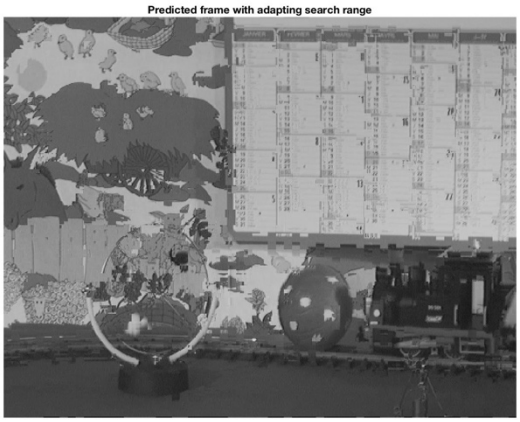

Figure 10 - Multiresolution Exhaustive Block Matching Algorithm results using Hierarchical Block Matching (method 1) using an adaptive search range.

Conclusion

Three Block-Matching Motion Estimation techniques were implemented in a manner of methods using MATLAB. PSNR, computing times, error images, motion vector fields, and the perceived quality of predicted frames produced were compared. The simple yet efficient 3-Step Search yielded the shortest execution time (1.09 seconds) and the EBMA produced the most accurate

predicted image (PSNR = 39.431). The combination of multiresolution techniques produced mixed results dependent of the search algorithm being used. Further investigation of the combination of search algorithms and multiresolution techniques should be investigated with optimization analysis in order to produce the most efficient and effective motion estimation algorithms for a given application.

References

[1] Y. Wang, J. Ostermann and Y. Zhang, *Video processing and communications*, 1st ed., Taipei: Pearson Education Taiwan, 2007

[2] B. Xiong and C. Zhu, "Efficient Block Matching Motion Estimation Using Multilevel Intra- and Inter-Subblock Features- Subblock-Based SATD", *IEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 1039-1043, July 2009

[3] X. Q. Gao, C. J. Duanmu, C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation", *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 501-504, March 2000

[4] P. Agathoklis, *ELEC 483 Digital Video Processing – Lecture slides*, University of Victoria, 2017

[5] I. Chakrabarti, *Motion Estimation for video coding*, 1st ed. Springer International PU, 2016

[6] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, "Motion-compensated interframe coding for video conferencing", *Procedings of Natural Telecommunication Conference*, pp. C9.6.1-C9.6.5, 1981

Appendix A

The accompanying MATLAB code to be submitted with this report can be accessed on the authors personal GitHub account: https://github.com/g-rambo/MATLAB_BMMA