An Analysis of Block-matching Algorithms for Efficient Motion Estimation

Tuna Temiz
Technische Universität Ilmenau
Ilmenau, Germany
tuna.temiz@tu-ilmenau.de

***Abstract*—Block Matching Algorithms are used as means of macroblock locating within a given series of digital video frames with the intention of motion estimation. There has been exhaustive research regarding these algorithms in the past thirty years, with many different algorithms developed as a result. As the number of available methods is virtually numerous, finding the most efficient one to use poses to be a constant issue. This paper analyzes and explains some of the most common Block Matching Algorithms with the intention of making the decision process easier regarding which algorithm to use based on their efficiency in various cases.**

Keywords—macroblock, full search, three step search, hierarchical search, efficiency, nearest neighbors search

I. Introduction

What very well may be the most intensive part of a video encoder in terms of computation is motion estimation. The required computational complexity is generally unsatisfactory in its initial state for use and therefore a certain reduction is a must, which remains among the most difficult issues for motion compensation. What is especially important in real-time video compression is this process, which is the most recent stage in video encoding. However, it is to be noted that certain applications can accomplish the compression offline in nonreal-time and therefore make the time spent on optimizing the compression more affordable. No additional motion compensation is required by the decoder, allowing a much better run-time compared to the encoder. Additionally, in various applications like the distribution of a video file, while there will be multiple decompressions, only one compression is necessary, which allows the said applications to meet the expense of a costly computation during the process of compressing.

Finding a certain point in the reference frame which fits the current macroblock most successfully and calculating the vector of motion between the two points remains a common method of accomplishing motion estimation. Therefore, discovering the best-fitting point is the focus in this process. An image is partitioned into certain rectangular blocks that do not overlap in the block-translation model, and it is commonly used for searching. By converting a similar point of source from the reference frame, the formation of each block in the predicted image is made possible. Turning or scaling of the block cannot be detected by this model. An example of the said models is the exhaustive search, sometimes referred to as the full search, which is a brute-force algorithm.

II. Exhaustive Search

In an exhaustive search (also called full search), each individual block in a given range is tested against the target block, which is the block it is defined to match. When it comes to the minimization of the Sum of Absolute Differences (SAD), this method of search is able to find the optimal motion vector each and every time as within the given search range the algorithm makes a comparison of all possible displacements. The pseudo code below is intended to show the algorithm in a C-like manner:

```
for ( int i = 0; i < 256; ++i ) {
  x[i] = pixel in current macroblock
  y[i] = pixel in a 16x16 block in reference frame
  for each k = 16x16 block in reference frame {
    D[k] = sum of distortion( x[i], y[i][k] )
  }
}

return k-th 16x16 block in reference frame that minimizes D
```

Figure 2-1. Pseudocode for Exhaustive Search [4]

Assuming the size of the search window is *2A+1* and the position of the current macroblock is its center *(0,0)*, lower-left and upper-right corners which correspond to the coordinates *(-A, -A)* and *(+A, +A)* will create a search window, generally in the shape of a rectangle. SAD of every location within the window will be evaluated by exhaustive search, and the complete number of points will equal to $(2A+1)^2$. Starting with the upper-left corner at *(-A, +A)* and going in what is called raster scan order, which is from left to right as well as top to bottom until there remains no unevaluated SADs at any position is a simple exhaustive search strategy. It should be noted that SAD positions are small just like the between frame movements of objects where a regular video sequence is concerned. The chances of discovering a minimum SAD close to the ends of the search window are slim. Instead, the action of simplifying an exhaustive search by choosing the center *(0,0)* as the starting point and going through different points in a spiral manner as shown in figure 2-2 can be considered. Evidently, there is no advantage to be gained by the evaluation of each SAD position. Instead, a tactic of premature termination can be used for a shortcut, which means the process of going through SAD's is ended when the current value is a larger one than the old minimum SAD value (particular attention must be paid here that the search is not affected as long as there are uncovered positions remaining in the window, and only the evaluation of the SAD in the position is ended). Through the use of this tactic, the chances of the search ending in a premature manner is quite likely while the pattern of the search expands further and further outwardly, which in turn results in a considerable reduction in the run-time of SAD calculation [3].

When it comes to the decision of a suitable range of search, there are various tactics to choose from. Multiple

things may factor in; however, the range of the search is most usually reliant on the degree of quality with which the fastest objects are wanted to be tracked, and the movement speed of said objects in the image sequence. Tracking an object which changes location greater than the height or the width between frames that follow each other is not feasible, if the frames of reference are adjacent as well. In the case of an object which has a movement speed of approximately half an image width in between two frames, tracking the said object may be logical, however within the context of television, this will be resulted in an object appearing and disappearing in a very short time, close to one tenths of a second, which is a speed greater than what is required. Assuming the dimensions of the video is around 512 × 512 which has a frame rate per second (fps) of 40, a motion corresponding to approximately 12 pixels per frame would need accommodation, if the intention is to track a certain object which requires around half a second to cross a frame. Doubling the range of search to 24 pixels will be required if the intention is to predict using not one but two frames in succession.

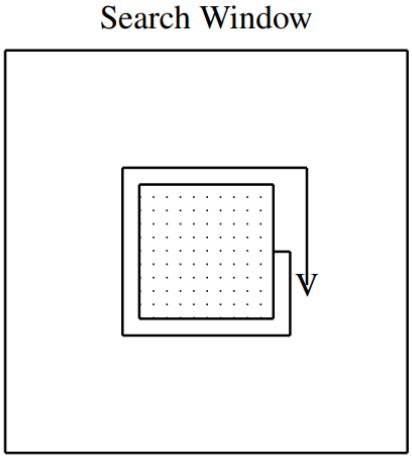

Figure 2-2. Full Search with Spiral Scan [5]

Additionally, the horizontal direction will have more, and swifter movements compared to the vertical direction in a real-life scenario. From cases of trial and error, it has been found out that searching the height half as much as the width yields the best and most effective results.

It can be said that exhaustive search is relatively simple in its implementation. Although, if something like the minimum SAD as the best match determination criteria is assumed, a truthful representation of the object that appears in two consecutive frames may not be possible. What makes a quality match is the least possible amount of errors of residual nature, with an end compression result that can be considered good. That being said, in the case of real movement not being accurately represented by the matches, correlation of the vectors of motion belonging to adjacent matching blocks will be poor, and as a result the encoding of the said vectors will not be efficient.

III. Fast Searches

Exhaustive search yielding the result that is most optimal is an already established fact, but from a computational point of view, it is undeniably demanding and intensive. This is where fast search algorithms come into play, and the said algorithms sacrifice the preciseness of the image prediction for a more efficient way of searching. Fast search algorithms use certain subsets of coordinates belonging to the window that is being searched for the assessment of the search criteria.

A. Three Step Search (TSS)

One popular fast search algorithm which is also considerably robust is the Three Step Search (TSS), which was first conceived by Koga et al in 1981 [1]. TSS was modified later on to become the N-Step Search (NSS), which calculates, in a particular coordinate subset, the SADs inside the search window. In order to find a certain pattern of search, it seeks the most suitable motion vectors. The search positions of this algorithm can be observed in Figure 3-1. For better clarity, assume that the center of the figure is also the point of origin (0,0). The first course of action is to choose a move which is of size b, as well as picking the origin as the starting point of the search. The said move b is equal to 4 in Figure 3-1. The coordinates of eight blocks which are b units far away from the origin and the origin itself has to be chosen for comparison in the first stage; first 9 coordinates have the value of ‘1’ in Figure 3-1. For the new center of the search, the coordinate that yields the smallest SAD has to be chosen, and in Figure 3-1, these are shown with a thicker outline. After this begins the next phase of the search and an additional 8 coordinates are defined which surround the updated center. In addition to this, the step size is also reduced to half, equaling 2 in this particular case. In Figure 3-1, the coordinates for this phase carry the value of ‘2’. The center of search is changed once more, based on the most suitable coordinate and the whole process is carried on in this manner until further division of the step size is no longer possible.

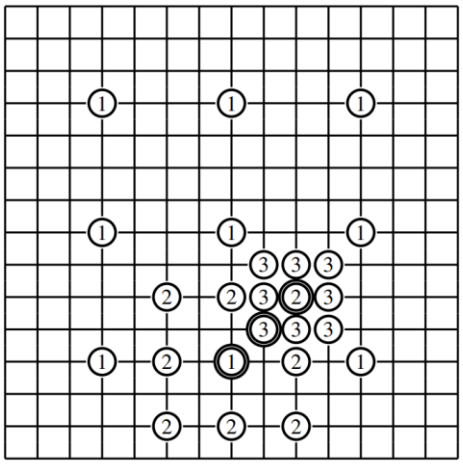

Figure 3-1. Three Step Search (TSS) [5]

B. *Hierarchical Search*

In order to minimize the amount of computations needed in the coarser parts, what is called a coarse-to-fine kind of approach is used for hierarchical search. Among many different values of pixels that are next to each other, the value that has been sampled from a specific set of coordinates has a possibility of being the average at a level that can be defined as coarse, which means that a bigger block likely contains a lot of different pixel coordinates but a lesser amount of search coordinates. In order to initially guess the movement or the motion, the hierarchical search algorithm seeks a bigger block as a beginning move and with employing blocks of a smaller size to search areas of a smaller size this process can be refined with success, provided that every individual block is properly filtered. Since the overall inclination of the movement is well defined first with the usage of bigger blocks that have been filtered, this method's chances of calculating the actual motion vector is better [2]. In addition to this, with the usage of smaller blocks, more reliable measurements are possible. It is a well-established fact that obtaining vectors of motion by employing hierarchical search will yield a much lower entropy value. What this means is the bit numbers required for the encoding process will be less.

With the usage of subsampling, various pyramidal visuals can be created by using the mean pyramid method, which makes a good example in this case. Beginning from coarser parts or in other words higher levels and moving towards lower level ones the vector of motion can be guessed by employing this method. If low pass filters are used to create the visual pyramids, the noise can be lessened in the coarser parts. Additionally, in order to create pyramidal visuals with more than one level, a basic averaging can be employed with great efficiency. As an example, assume the value of the grey level belonging to the position (x, y) is $g_L(x, y)$ which itself belongs to the L-th level and the initial visual belonging to the level 0 is represented by $g_0(x, y)$. In a window that does not overlap which belongs to the subsequent lower level, the four values of pixels have to be averaged in order to acquire the value of the pixel belonging to a certain level, which is shown in the following equation [4]:

$$g_L(x,y) = \lfloor \frac{1}{4} \sum_{i=0}^{1} \sum_{j=0}^{1} g_{L-1}(2x+i, 2y+j) \rfloor$$

The floor function here is x, which is denoted by $[x]$ in this equation. A single pixel belonging to level 2 will equal a block of size of 2×2 at level 1 and a block size of 4×4 at level 0, assuming the pyramid is comprised of three levels in total. Following this logic, at level L > 0, a block of size $16/2^L \times 16/2^L$ will be created at a level 0 block of size 16×16. By employing the minimal SAD principal, the visuals can be searched beginning from level 2 following the creation of the mean pyramid. In order to do this the vector of motion which bears the minimum SAD belonging to that level as the coarse vector of motion is selected. The vector of motion that belonged to the level with the higher value is then sent to the adjacent lower level which happens to be level 1 in this instance. What is done afterwards is the vector of motion that was acquired is used to lead the betterment process belonging to that level. The estimation of movement method is repeated once more into level 0. It is highly probable that the exact values will be seen at various points, as depending on tinier blocks of a relative nature, SAD's are calculated at the level with the highest value. This poses an obvious obstacle, and in order to overcome this, the usage of multiple candidates belonging to the highest level (which corresponds to level 2 in this case) is required. At the second level, a certain amount of vectors of motion are pushed into the lower level. In order to pinpoint the least amount of change coordinate to be the center if search at level 0, exhaustive search inside a small window which surrounds the possible candidates is employed. The locations of search belonging to the algorithm can be seen in Figure 3-2. What is selected as the centers of search for windows that belong to the next level are the three points that match the best, which are in level 2 and level 1 of the figure respectively. The window in which the best match is searched can be seen at level 0.

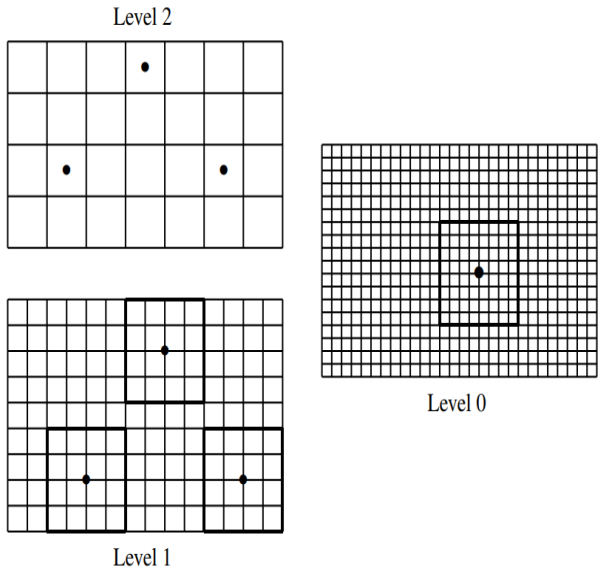

Figure 3-2. Search Locations for Hierarchical Search [4]

C. *Nearest Neighbours Search (NSS)*

Since it is not possible to say with certainty that the function of contortion rises continuously in an ordered manner in the case of a displacement further from the universal lower boundary potential, Three Step Search, Hierarchical Search as well as various additional non-optimal algorithms can be stuck in the regional lower boundary, which is a problem that needs addressing [4], [5]. The reason this is so is that the performance of the compression could be reduced drastically because the accuracy with which the algorithm makes an estimation will be seriously hampered, especially in contrast to exhaustive search. By introducing a means of forecast into one of the fast search methods, the

possibility of the aforementioned problem occurring can be prevented to a great extent. With the usage of formerly embedded vectors of motion which are used to show macroblocks of either a spatial or temporal nature, the present vector of motion can be forecasted. Since the forecasted vector of motion will most likely be within a closer vicinity compared to the vector (0,0) to the universal lower boundary potential, the process of confining the beginning point of the search in this manner will make it unlikely to get stuck in the regional lower boundary. It is a rather straightforward process from then on to find the most fitting vector of motion by applying a search on an area that can be considered small, assuming that the forecasted vector of motion is precise.

In order to reach a level of preciseness comparable to that of Exhaustive search, albeit with a considerably lesser complexity in terms of computational power required, Nearest Neighbours Search (NSS) employs a heavily regional motif of search in addition to the vector of motion forecast tactic that was described previously. In doing so, this algorithm makes an authentic vector of motion forecast which is able to pinpoint likely disturbances in a sequence of video images possible [4]. The difference between the present vector of motion and the forecasted vector of motion is passed on. The algorithm forecasts a vector of motion by the formerly embedded vectors of motion which makes this possible. The difference is minimized by the Nearest Neighbours Search which uses this occurrence to its advantage by prioritizing the vectors of motion which are in the vicinity of the forecasted vector of motion. An example of search coordinates belonging to the algorithm can be seen in Figure 3-3, where the SAD belonging to the initial point of the search is analyzed (and afterwards given the value of '0'). Afterwards the SAD belonging to the coordinates of the forecasted vector of motion and the SADs belonging to the perimeter (which are given the value of '1') are processed. The coordinate set that yields the minimum SAD value is chosen (which are emphasized in the figure) and turns into the point of origin for another search (which are given the value of '2' in the figure) which keeps running, unless the sum of absolute differences at '0' is the minimum value, in which case the process will be stopped. The ultimate coordinate point chosen by the algorithm is an emphasized '3' while the point of origin for the upcoming process is emphasized as '2'.

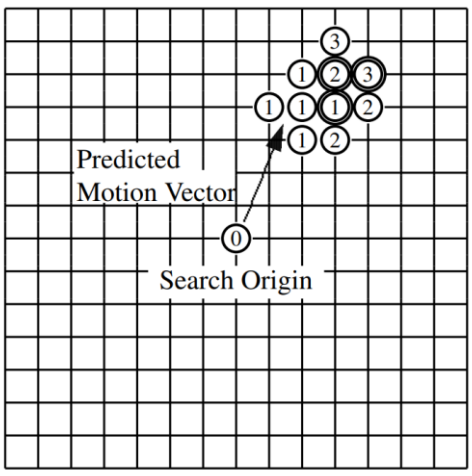

Figure 3-3. Nearest Neighbours Search (NSS) [5]

CONCLUSION

This paper serves as a survey study for efficient motion estimation, in which several block-matching algorithms were explained and analyzed. That being said, there are many more fast search algorithms that were not covered in detail by this paper, like Four Step Search which is an improvement compared to the already covered Three Step Search in terms of its computational demand and its emanated noise amount when comparing peak signals [6]. After an analysis of their method of function, predicting the efficiency of the algorithms turns into a rather straightforward task. It is safe to assume based on existing data that the Hierarchical Search is capable of providing a level of efficiency comparable to that of the Exhaustive Search, while at the same time being less demanding to execute in many cases. It is also worth noting that Hierarchical Search is more feasible to execute using hardware that was customized compared to other algorithms. While currently outshined by newer methods, the Three Step Search is the oldest algorithm for fast block matching and remains a reliable one to this day. However, there is no universal algorithm that fits all situations perfectly, and no algorithm will be reliable for every single case. The efficiency of a certain algorithm is best decided by making a comparison between its efficiency and that of Exhaustive Search's, which is the most efficient as well as the most demanding of all algorithms. As the size of the operation grows, the required computing power will grow as well.

One direction where this paper can expand upon is being more inclusive in terms of algorithms analyzed and compared. This will allow for a more reliable comparison to be made. Something that may be analyzed in the future is the bit stream as what is being transmitted boils down to bit streams at the end, and it will also make algorithm analysis process more reliable. It is also possible to try and combine various algorithms previously described and observe whether additional efficiency can be achieved without sacrificing further computational power.

REFERENCES

[1] T. Koga, K. linuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in Proc. Nat. Telecommun. Conf., 1981

[2] Kwon Moon Nam, Joon-Seek Kim, Rae-Hong Park "A Fast Hierarchical Motion Vector Estimation Algorithm Using Mean Pyramid" IEEE Transactions on Circuits and Systems for Video technology, vol. 5, August 1995.

[3] Th. Zahariadis, D. Kalivas "A Spiral Search Algorithm For Fast Estimation Of Block Motion Vectors" Signal Processing VIII, theories and applications. Proceedings of the EUSIPCO 96. Eighth European Signal Processing Conference

[4] F. June, An Introduction to Video Compression in C/C++,Createspace, 2010.

[5] F. June, An Introduction to 3D Computer Graphics, Stereoscopic Image, and Animation in OpenGL and C/C++, Createspace, 2011.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[6] Lai-Man Po, Wing-Chung Ma "A Novel Four Step Search Algorithm For Fast Block Motion Estimation" IEEE Transactions on Circuits and Systems for Video Technology, June 1996.