Authored by Yi Woeh Lim (33354715), Wai Hon Loke (33522812)

Motion Compensation

Introduction

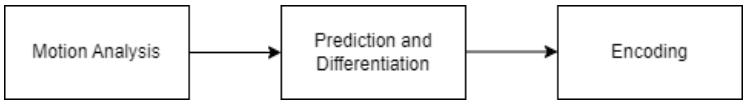

Figure 1: *High Level Block Diagram of Motion Compensation*
Source: Adapted from [1]

Motion compensation (MC) is an algorithm that predicts a frame in a video given previous frames or even future frames in more advanced MC [2]. MC exploits the redundancy between consecutive frames. MC coding is commonly split into three stages as shown in the above block diagram. The first stage, which is the motion analysis stage: consecutive frames are analysed to detect motion between frames. The current frame is "sliced" into blocks; motion vectors for each block are created with respect to the closest match in the previous frame [1]. The next stage would be prediction and differentiation. The current frame is predicted using previous frame and motion vectors [1]. However, prediction is inevitably prone to error, thus the prediction error or residual for each pixel is calculated to keep track of it [1]. The last stage involves encoding the current frame as the residual and motion vectors.

Methodology

Motion compensation performance was analysed in MATLAB using two consecutive frames extracted from a 4K 2160p video. For each block in the current frame, motion vectors were estimated using two different search strategies: exhaustive search and logarithmic search. A total of 4 different block sizes were evaluated for logarithmic search, following a $2^n \times 2^n$ configuration with n = 1, 2, 3 and 4 being implemented, whereas exhaustive search was investigated using only n = 2, 3, 4 due to immense computation and time resources needed to implement n = 1. Further analysis on the effect of different search ranges was carried out, ranging from p = 0 to p = 16 for linear search and $p = 2^n$ with n = 0 to 10, for logarithmic search. Note that $search\,range = (p + block\,size) \times (p + block\,size)$. In both cases, a search range of 0 (exhaustive) and 2^0 (logarithmic) represents the baseline case where no motion vectors were used in predicting the current frame. Processing complexity in the form of time complexity and root mean square error (RMSE) were used as the performance metric to evaluate the effectiveness and accuracy of each of these factors. To ease analysis and interpretation of the collected data, all the results were summarised and presented using 3D plots.

Result

From the results obtained (refer to Figure 2 below and Appendix A), increasing search range would significantly increase computation time needed for exhaustive search while the impact is less adverse for logarithmic search. This is mainly due to how the search range expands quadratically with every increase in p, with this being said, exhaustive search would scale quadratically as well, leading it to have a time complexity of $O(n^2)$. Whereas logarithmic search of time complexity $O(log(n))$ would not suffer much adverse effect when search range grows quadratically due to its efficient algorithm. By setting p to be constant, block size affects both exhaustive search and logarithmic search similarly. This is because block size determines the number of segments a frame is divided into, and the number of times the searching process is repeated. As such, increasing the block size will inevitably decrease the overall computational time as a result.

When examining the effect of manipulating each parameter on the accuracy of motion matching, an increase in block size increases the prediction error because a large group of pixels is being predicted as a whole using a single motion vector, effectively blurring pixel-level movement details. However, this overall effect is small when investigated in MATLAB because this phenomenon is highly frame dependent. Two consecutive frames with highly complex movement involving different small bodies moving in different directions will be heavily affected by the chosen block size whereas frames that consist of objects that move as a whole in a single direction will be least impacted. When the block size is kept constant, changing the search range turns out to have different effects depending on the search algorithm used. In the case of exhaustive search, increasing the search range will rapidly decrease the percentage

error due to more range being covered to find the optimal matching block. Nevertheless, as the search range increases, it is observed that the percentage error becomes saturated, which can be interpreted as an optimal matching block has been found. Therefore, increasing the search range will simply yield no additional improvement. For logarithmic search, increasing the search range will drop the percentage error rapidly at first until it reaches a global minimum, in which case the percentage error starts to increase. This observation is due to the nature in which the logarithmic search is carried out. In the first iteration, the algorithm will look at 9 different blocks with step size equal to the search range divided by 2, and the subsequent iterations will have their search center moved correspondingly to the previously found best matching block. As such, the chance of successfully finding the best match will be highly dependent on this first iteration that can reveal where the true matching block is. As a result, the search range must be carefully selected for logarithmic search depending on the movement present in the subsequent frames. Lastly, it is important to note that a motion compensation done without motion vectors yields the highest percentage error in predicting the new frame, highlighting the significance of considering motion vectors.

Conclusion

In a nutshell, through investigation carried out in MATLAB, it is revealed that exhaustive search as the most fundamental block matching algorithm performed best in terms of accuracy in finding the best matching block due to its nature of examining every possible block inside a given search radius. At the same time, it is the most computational expense when carrying out the given task. On the other hand, logarithmic search becomes a more realistic approach due to its improved algorithm to reduce the computational time. However, parameters such as search range and block size must be chosen carefully through motion analysis to give the best results when using logarithmic search. Given the significance of bitrate efficiency especially in telecommunications, further improvements to motion compensation can be achieved through advanced algorithms such as variable block size that allows a flexible trade-off between prediction error and bit size, leading to a better overall compression performance.

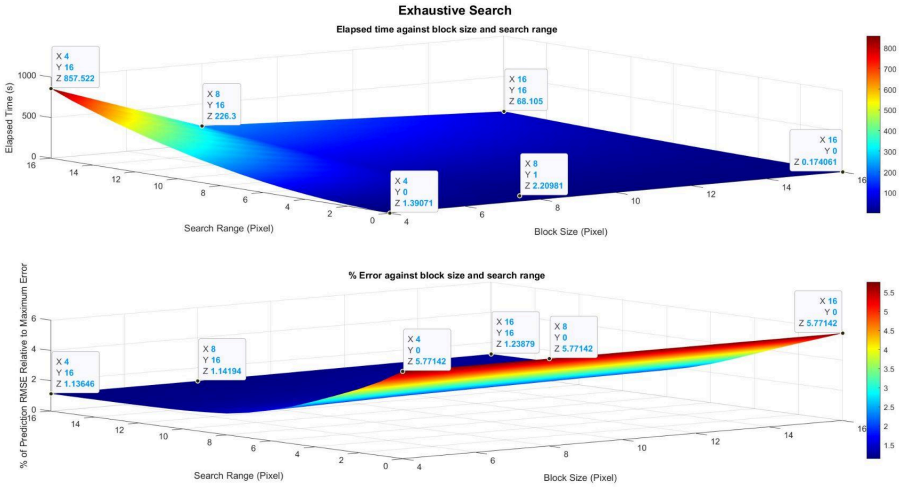

Figure 2: *3D subplots of exhaustive search performance*

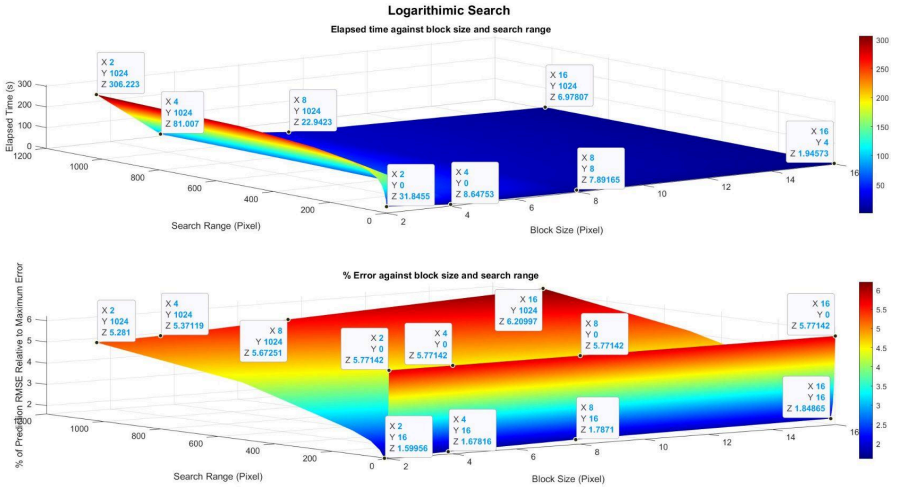

Figure 3: *3D subplots of logarithm search performance*

Authored by Yi Woeh Lim (33354715), Wai Hon Loke (33522812)

References

[1]
Y.-Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering*. CRC Press, 2019.
[2]
Wikipedia Contributors, “Motion Compensation,” *Wikipedia*, Apr. 20, 2025. https://en.wikipedia.org/wiki/Motion_compensation# (accessed May 18, 2025).