User Input Search - Custom Motion Estimation Algorithm Optimized for UAVs

J. Benjak, D. Hofman
Faculty of Electrical Engineering and Computing / Department of Control and Computer Engineering, Zagreb, Croatia
jakov.benjak@fer.hr, daniel.hofman@fer.hr

***Abstract*—Unmanned aerial vehicle technology is growing rapidly as it finds its application in the various industries, including military & defense, agriculture, logistics, transportation, healthcare, entertainment and many others. One of the fastest growing industries including drones is the entertainment industry. More specifically, First Person View (FPV) piloting has become a popular sport which attracts huge masses. In FPV systems, the pilot wears FPV goggles and controls the drone using a controller, and the drone transmits video data to the goggles in real time. Since drones usually fly very quickly, the video quality in terms of resolution, compression artefacts and end-to-end delay must be maintained. This paper explores the idea of optimizing the Motion Estimation (ME) algorithm of existing High Efficiency Video Coding (HEVC) algorithms by utilizing user input from the controller. For example, if the drone is directed to hover to the left, then it would make sense to search for the most similar blocks in the previous video frame only on the left side of the referent block. For the purpose of this research, a HEVC video coder was customized to receive additional input besides the video data – the user input from the controller, or in other words, drone movement directions. We call this ME algorithm User Input Search (UIS). Our UIS algorithm is compared with the standard ME algorithms and its efficiency is tested.**

Keywords*—*High Efficiency Video Coding, Custom Motion Estimation, Video Coding, Data Compression

I. INTRODUCTION

Unmanned aerial vehicles (UAVs), commonly known as drones, have witnessed a massive growth in their application in various industries [1]. The entertainment industry, in particular, has embraced the use of drones with First Person View (FPV) piloting systems. In FPV systems, the pilot controls the drone using a controller and receives real-time video data through FPV goggles. Such systems are commonly used in drone races and film making. Drone racing is a popular sport in which pilots control UAVs to compete in races through obstacle courses, and the goal is to complete the course as fast as possible, without crashing. However, such systems have their limitations which are yet to be solved. Due to the rapid movement of drones, maintaining high video quality, including resolution, low end-to-end delay and hiding compression artifacts is challenging.

To address this issue, this paper presents a novel approach to optimizing the Motion Estimation (ME) algorithms in existing High Efficiency Video Coding (HEVC) video coders. Our User Input Search (UIS) algorithm involves incorporating user input from the controller to improve the efficiency of the ME process. The main idea is to search for the most similar blocks only in the direction of the drone movement. For example, if the drone hovers to the left, UIS would only search for the most similar blocks on the current block's left side in the previous frame. UIS results in less searches, which means less operations in the encoding process. This optimization should maintain the video quality, but increase the encoding speed, or in other words, reduce latency.

This paper tests this idea only on the drone footage, but the idea can be expanded to other machines as well. For example, land vehicles like remotely controlled trucks, machines for remote medical procedures, remote security cameras, etc. Basically, any kind of remotely controlled device which transmits video. Another yet to be researched topic is using this idea on mobile phone cameras, by using the information from mobile phone's accelerometer in order to estimate device motion.

The UIS algorithm was implemented on an open-source High Efficiency Video Coding (HEVC) encoder and tested on several drone sequences. All sequences were filmed by the author's associates. At the end of this research, a performance comparison of our UIS algorithm and the standard ME algorithms is given.

II. RELATED WORK

Because ME can be the most power and time consuming part of the entire coding process, a lot of researchers tried optimizing it. Authors in [2] developed a low-complexity, adaptive fractional-pixel ME skipped algorithm for the HEVC encoder. Their algorithm reduced ME encoding time by an average of 63.22%, with encoding efficiency maintained. A method for optimizing the number of pixels in the current search window has been proposed in [3]. Their proposed method can save 56% of computations in comparison with the Test Zone Search (TZS) scheme - the default HEVC search scheme, with negligible decrease of coding quality. TZSearch algorithm improvements and other optimized ME algorithms have also been successfully implemented in [4], [5], [6]. Some more recent works implement artificial intelligence into ME algorithms in order to improve them. One example of optimizing an existing HEVC ME algorithm has been proposed in [7].

None of the above mentioned researches considered utilizing user input from any sort of input device. To the best of the authors knowledge, there are no published researches which discuss and implement custom ME algorithms which leverage user input from the controller.

III. USER INPUT SEARCH IMPLEMENTATION

For the purpose of this research, an open source HEVC video encoder Kvazaar [8] was upgraded to include our custom ME algorithm, UIS. Kvazaar source code includes a few different ME algorithms, such as TZS, Diamond Search (DS) and Hexagon Search (HS). UIS was implemented by tweaking the DS algorithm. DS aims to find the best prediction block from the previous frame for each block in the current frame by searching for the closest match. The search pattern forms a diamond shape where the center of the diamond is initially set to the location dependant on the search step, and the four points on the diamond correspond to candidate blocks. The four points are points above and below the center, and to the right and to the left of the center. The algorithm checks the cost of each candidate block in the diamond and moves the center of the diamond to the candidate with the lowest cost. The cost function is usually the Sum of Absolute Differences (SAD) or the Mean Square Error (MSE). SAD simply sums up all the absolute values of pixels differences between the current block and the candidate block. MSE squares each pixel difference and performs the mean operation afterwards. This process continues until no better candidate can be found or a stopping criterion is met (e.g., a maximum number of search steps). The best motion vector found during the search process is stored in the variable and returned as the result of the algorithm.

UIS, on the other hand, works similarly to the DS, but checks only for one candidate in each step, in the direction of the drone movement. For example, if the drone moves upwards, the DS would calculate the cost for all four candidates, in every direction, while the UIS would only calculate the cost for the upward candidate. This would be repeated until the best match is in the center, or the maximum number of search steps is reached. However, the maximum number of search steps was removed while performing tests.

Test were performed on four sequences, filmed using a DJI ZENMUSE X5S camera on a DJI Inspire 2 drone. Camera output was a set of .DNG frames, which were later losslessly compiled into a .yuv video using FFMpeg [9]. In each sequence, the drone moved or rotated straight in one direction, either upwards, left or right. Each sequence was different in terms of spatial complexity, meaning that obstacles - such as houses, bushes, trees, etc., were at different distances and generally the number of such obstacles was different in each sequence. Figure 1 and Figure 2 display sequences which were used for testing. Each sequence was exactly 240 frames long. During encoding, only I and P frames were used, because in real-time encoding, B frames do not make sense. However, some test were made using B frames as well, to show the UIS true advantages which can be used when coding any kind of drone footage which had a controlled movement, and which doesn't require ultra-low latency real-time encoding.

By inspecting the encoded video in the Elecard StreamEye software [10], it was very clearly seen that most of the motion vectors were indeed following the direction of the drone movement, when using the DS algorithm. This was the case with all the tested sequences. Of course, the results were the same when using the UIS algorithm. An example is shown in the Figure 3, and it can be seen that the generated motion vectors are very similar, which is exactly the result we were hoping for.

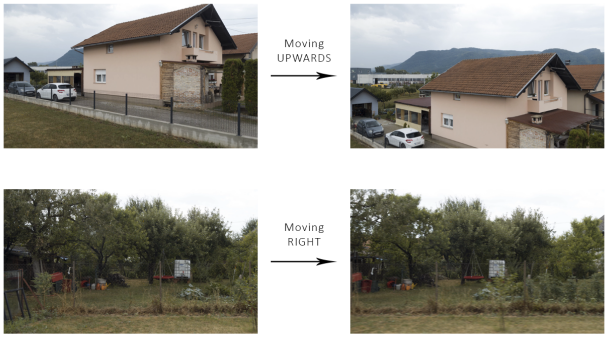

Fig. 1: Drone movement in the 1st and the 2nd sequence

Fig. 2: Drone movement in the 3rd and the 4th sequence

IV. EXPERIMENTAL RESULTS

All the test were performed on a 64-bit Windows personal computer with 16GB of RAM, 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz processor and an NVIDIA GeForce GTX 1650 graphics card. While performing the tests, no other processes were purposely started on the computer. Since some processes can start unnoticed during the encoding process, in order to do the UIS performance evaluation more precisely, we used the Intel VTune Profiler software [11]. To check exactly how

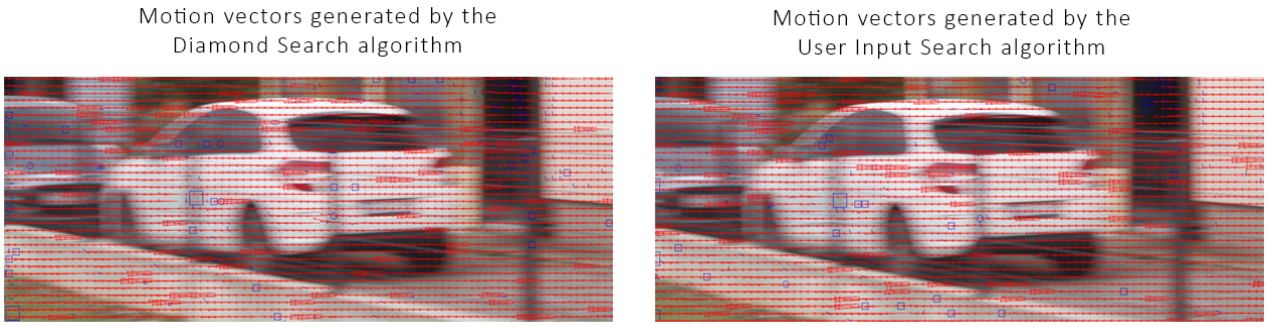

Fig. 3: Motion vectors generated by the DS and the UIS algorithms

much processor time each function had used, we used the VTune's Hotspots analysis and the Flame Graph (Figure 4). Using the Flame Graph tool, we were able to extract only the *diamond_search* and the *user_input_search* functions which perform the DS and the UIS algorithms. Table I represents crucial data in form of a table. Search time refers only to the CPU time that was taken by the *diamond_search* and the *user_input_search* functions. The obtained data shows that the goal was accomplished and that the UIS algorithm indeed improves the overall encoding process. The encoded file sizes were in a very close range, with the biggest difference being in the first sequence (10.2MB for DS, and 10.3MB for UIS). PSNR values were the same in all cases, but the search time was reduced significantly.

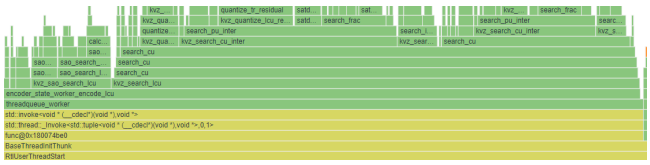

Fig. 4: VTune's Flame Graph from the Hotspots analysis

An noteworthy point is that the search time does not occupy a significant portion of the total encoding process time. The biggest percentage of the total encoding time was in case of the second sequence, where the search function took 1.4% of the total encoding time when using DS, and 0.4% when using UIS. This observation is visualised in Figure 5 and Figure 6, where the search function is highlighted with the pink color. The main reason behind the small percentage is because all sequences were encoded using the Kvazaar's default Group Of Pictures (GOP) structure, which is *lp-g4d3t1*, a low-delay p-frame only GOP. In such structure, each frame references only one frame, the previous frame. Figure 7 and Figure 8 display a Flame Graph of the second sequence, encoded using GOP *8: B-frame pyramid of length 8*. In such structure, each frame references up to four frames, which results in much more total searches, but in better compression as well. When using this structure, UIS shows it's real power, because the search function took 11.9% of the total encoding time when using DS, and only 2.7% when using UIS. In this case, UIS shortened the total encoding time by roughly 6.5%, while preserving video quality (in terms of PSNR), and increasing the encoded video file size by only 0.9%. In some cases, other than saving time, UIS resulted with video files of less size than the files generated using the DS algorithm.

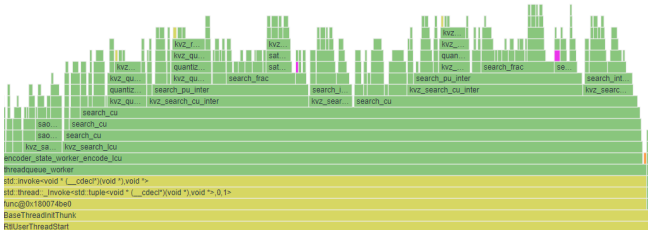

Fig. 5: DS algorithm portion of the total encoding process time (pink color)

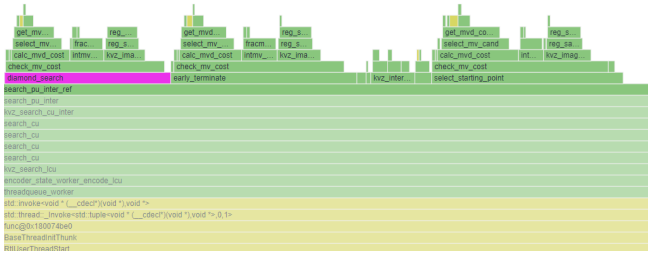

Fig. 6: Figure 5. zoomed-in

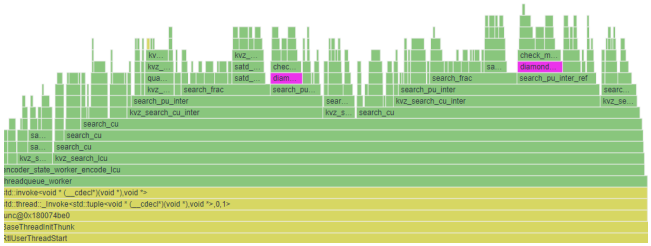

Fig. 7: DS algorithm portion of the total encoding process time (pink color), with using B frames

Algorithm	Sequence number	Encoded video file size [MB]	Search time [CPU s] (less is better)	PSNR [dB] (more is better)
DS	1	10.2	1.53	45.36
	2	38.1	38.70	43.08
	3	19.7	1.11	43.68
	4	9.87	12.37	44.92
UIS	1	10.3	0.37	45.36
	2	38.1	11.32	43.08
	3	19.7	0.39	43.68
	4	9.89	3.58	44.92

TABLE I: Comparison of encoded file size, encoding time and encoded file PSNR for DS and UIS algorithms

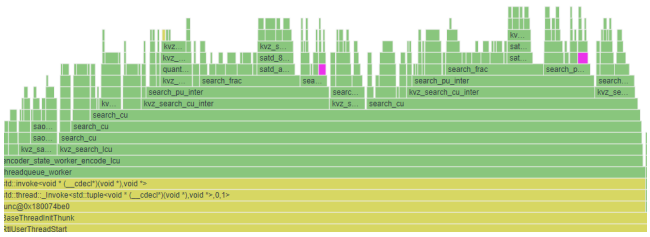

Fig. 8: UIS algorithm portion of the total encoding process time (pink color), with using B frames

V. CONCLUSIONS AND FUTURE WORK

This paper presents an optimization of the ME algorithm in the HEVC encoder Kvazaar by utilizing user input from the controller. An additional ME algorithm was implemented as part of the Kvazaar encoder, which we called UIS. Our results show that the UIS algorithm is more efficient compared to the standard DS algorithm. Video compression rate and quality is preserved, but the encoding speed is increased. When using the default Kvazaar's GOP structure, ME process was up to 4.14 times faster, and when using the GOP structure *8*, which includes B-type frames, the ME process was 4.72 times faster.

Future work includes modifying other ME algorithms besides DS, like HS or TZS. UIS algorithm could be upgraded in terms of number of search directions. Currently it offers only four directions, but could easily be expanded to also include diagonal directions. Besides hovering and rotation, UIS could be expanded to consider drone's forward and backward movement. Directions are not the only input the encoder receives from the user, it also receives the information regarding drone movement speed. Movement speed could be included in the UIS as well, to affect the search distance parameter. When the drone moves faster, it is expected that similar blocks will be a little more further in comparison when the drone moves slower. Of course, this is just a general rule which would have to be tested, because obstacles can always be either far or close to the drone. UIS could be implemented on the x264 encoder as well, since for some cases, it showed decent results in comparison with HEVC [12]. It could also be integrated in a system which offloads video encoding energy to the decoder side, performing ME only on the video G (green) component, instead of Y [13].

In summary, this research presents a promising solution to enhance video encoding speed controlled-camera video systems, offering ample potential for future research and growth.

VI. ACKNOWLEDGEMENT

This work has been supported in part by the project KK.01.2.1.02.0054 Razvoj uređaja za prijenos video signala ultra niske latencije (Development of ultra low latency video signal transmission device), financed by the EU from the European Regional Development Fund.

REFERENCES

[1] N. S. Labib, M. R. Brust, G. Danoy, and P. Bouvry, "The rise of drones in internet of things: A survey on the evolution, prospects and challenges of unmanned aerial vehicles," *IEEE Access*, vol. 9, pp. 115 466–115 487, 2021.

[2] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive fractional-pixel motion estimation skipped algorithm for efficient hevc motion estimation," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 14, 1 2018.

[3] L. Gao, S. Dong, W. Wang, R. Wang, and W. Gao, "A novel integer-pixel motion estimation algorithm based on quadratic prediction," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2015-December, pp. 2810–2814, 12 2015.

[4] Z. Pan, Y. Zhang, S. Kwong, X. Wang, and L. Xu, "Early termination for tzsearch in hevc motion estimation," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 1389–1393, 10 2013.

[5] H. Kibeya, F. Belghith, H. Loukil, M. A. B. Ayed, and N. Masmoudi, "Tzsearch pattern search improvement for hevc motion estimation modules," *2014 1st International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2014*, pp. 95–99, 2014.

[6] P. Nalluri, L. N. Alves, and A. Navarro, "Complexity reduction methods for fast motion estimation in hevc," *Signal Processing: Image Communication*, vol. 39, pp. 280–292, 11 2015.

[7] E. M. Ibrahim, E. Badry, A. M. Abdelsalam, I. L. Abdalla, M. Sayed, and H. Shalaby, "Neural networks based fractional pixel motion estimation for hevc," *Proceedings - 2018 IEEE International Symposium on Multimedia, ISM 2018*, pp. 110–113, 1 2019.

[8] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: Open-source hevc/h.265 encoder," *MM 2016 - Proceedings of the 2016 ACM Multimedia Conference*, pp. 1179–1182, 10 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2964284.2973796

[9] "FFmpeg," https://www.ffmpeg.org/, accessed: 2023-02-04.

[10] "Video analysis software, mpeg analyzer for video quality tests automation, batch execution | elecard: Video compression guru," https://www.elecard.com/products/video-analysis/streameye, accessed: 2023-02-04.

[11] "Fix performance bottlenecks with intel® vtune™ profiler," https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html#gs.ohbu13, accessed: 2023-02-04.

[12] J. Benjak, D. Hofman, J. Knezović, and M. Žagar, "Performance comparison of h.264 and h.265 encoders in a 4k fpv drone piloting system," *Applied Sciences 2022, Vol. 12, Page 6386*, vol. 12, p. 6386, 6 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/13/6386/htmhttps://www.mdpi.com/2076-3417/12/13/6386

[13] D. Hofman and J. Benjak, "Offloading video encoding energy consumption to the decoder," *2022 7th International Conference on Smart and Sustainable Technologies, SpliTech 2022*, 2022.