




Article

# Efficient Motion Estimation for Remotely Controlled Vehicles: A Novel Algorithm Leveraging User Interaction

Jakov Benjak , Daniel Hofman \*  and Hrvoje Mlinarić 

Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia; jakov.benjak@fer.hr (J.B.); hrvoje.mlinaric@fer.hr (H.M.)

\* Correspondence: daniel.hofman@fer.hr

**Abstract:** Unmanned Aerial Vehicles (UAVs) are increasingly being used in a variety of applications, including entertainment, surveillance, and delivery. However, the real-time Motion Estimation (ME) of UAVs is challenging due to the high speed and unpredictable movements of these vehicles. This paper presents a novel algorithm for optimizing ME for Remotely Controlled Vehicles (RCVs), with a particular focus on UAVs. The proposed algorithm, called Motion Dynamics Input Search (MDIS), incorporates information from vehicle motion dynamics estimation to enhance the accuracy and efficiency of ME. The MDIS algorithm addresses the challenges associated with real-time ME in RCVs by leveraging user input to guide the search for the most similar blocks in the previous video frame. Through extensive experimentation and evaluation, this study demonstrates the effectiveness of the proposed algorithm in improving ME performance for RCVs. The findings highlight the potential impact of user interaction and motion dynamics estimation in shaping the future of ME algorithms for RCVs and similar applications.

**Keywords:** high-efficiency video coding; custom motion estimation; video coding; data compression; remotely controlled vehicles; controller data



**Citation:** Benjak, J.; Hofman, D.; Mlinarić, H. Efficient Motion Estimation for Remotely Controlled Vehicles: A Novel Algorithm Leveraging User Interaction. *Appl. Sci.* **2024**, *14*, 7294. <https://doi.org/10.3390/app14167294>

Academic Editors: Mehmet Aydin, Yong Yue and Xiaohui Zhu

Received: 20 July 2024

Revised: 10 August 2024

Accepted: 13 August 2024

Published: 19 August 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

ME is a critical component in many video processing applications, such as video compression, object tracking, and video stabilization. ME works by predicting the movement of objects within video frames, allowing for more efficient data encoding and transmission. Common ME algorithms include Full Search (FS), Diamond Search (DS), Hexagonal Search (HS) and Test-Zone Search (TZS), each offering a different trade-off between computational complexity and accuracy. FS, while the most accurate, is computationally intensive, making it less suitable for real-time applications. DS, HS, and TZS, on the other hand, provide faster computations with slightly lower accuracy, making them more practical for applications requiring low latency.

In recent years, there has been growing interest in effective video compression for RCVs, such as drones, and UAVs [1]. This is due to the increasing popularity of UAVs for applications such as aerial photography, videography, military, and entertainment [2]. The entertainment sector has witnessed a significant boost with the rise of drone racing. This engaging sport involves pilots skillfully maneuvering UAVs through obstacle courses, aiming to complete the course in the quickest time possible. As drones can fly very fast when it comes to drone racing, it is crucial that the latency in video transmission from the drone to the pilot's first-person view (FPV) goggles is as minimal as possible. This is not only important for the pilot's performance, but also for the safety of the drone, as real-time video adaptation can help avoid collisions [3]. Statistics indicate a growing need for low-latency video processing in drones and other technologies. For instance, drone racing requires video transmission latencies below 150 milliseconds to ensure that pilots can react swiftly to their environment. Similarly, applications in military surveillance

and emergency response demand near-instantaneous video feedback to make timely and informed decisions.

To address this, our research focuses on reducing latency through video compression before network transmission, a challenging task under limited computing resources. Since ME is typically the most time-consuming part of video coding, we propose an innovative approach to accelerate it. This paper introduces a novel ME algorithm, Motion Dynamics Input Search (MDIS), which builds upon the foundation of the well-known DS algorithm while incorporating motion dynamics estimation that responds to real-time user interaction with the controller. Instead of searching all the points of the diamond for each block, MDIS only searches the selected locations based on the user input from the controller. Additionally, the search range is modified based on the location of the pixel in the image and the current drone movement type. While our research is centered on UAVs and FPV systems, the MDIS algorithm can be applied to any RCV system.

We evaluate MDIS on a drone-flying simulator which records user inputs for each frame. We also evaluate it on real 4K drone footage. By leveraging motion dynamics estimation in the ME process, our algorithm aims to enhance the efficiency, accuracy, and adaptability of ME for RCVs, with a primary focus on UAVs.

In summary, the main contributions of this paper are as follows:

- The development of the MDIS algorithm, which integrates motion dynamics estimation and user interaction to enhance ME efficiency.
- The comprehensive evaluation of MDIS using both simulated and real drone footage, demonstrating significant improvements in encoding speed and compression ratio without compromising video quality.
- The implementation and testing of MDIS within an open-source HEVC video encoder, showcasing its practical applicability and performance advantages.

## 2. Related Work

The problem of ME in general and for UAVs has been a topic of significant interest in recent years. A key focus has been on improving the efficiency and accuracy of block-matching ME algorithms.

The researchers of [4] proposed an All-Direction Search (ADS) pattern, which searches for the best block in all possible directions. For further improvement in search speed, a halfway stop technique was applied in the search process. The results showed that the proposed ADS algorithm outperformed other state-of-the-art and eminent ME algorithms. This work is particularly relevant as it presents a novel approach to block-matching that can potentially enhance the performance of the DS algorithm. Wang and Yang [5] proposed an improved fast ME algorithm for H.266/VVC, where they replaced the DS model with an HS model. They also increased the number of initial matching search points of the rotation HS model, which led to a significant reduction in the overall coding time. Hassan and Butt [6] studied the effects of a meta-heuristic algorithm on ME. They proposed the Firefly algorithm for ME, which saved a considerable amount of time with a comparable encoding efficiency. Multiple other ME optimizations have also been successfully implemented in [7–10].

In the realm of user interaction, Yu et al. [11] presented EyeRobot, a concept that enables dynamic viewpoints for telepresence using the intuitive control of the user's head motion. This approach could potentially be adapted for the MDIS ME algorithm. Yoo et al. [12] proposed a hybrid hand-gesture system that combines an inertial measurement unit (IMU)-based motion capture system and a vision-based gesture system to increase real-time performance. Their approach divided IMU-based commands and vision-based commands according to whether drone operation commands are continuously input.

The work by Varga et al. [13] on the validation of a Limit Ellipsis Controller (LEC) for rescue drones presents an innovative control algorithm designed for UAVs in emergency situations. In their study, the LEC was adapted for indoor environments, showcasing its potential to improve drone stability and maneuverability in semi-structured settings. The

integration of our algorithm could further improve their system by optimizing the ME in both the visual and thermal camera equipped on the UAV. Similarly, [14] illustrates the significant utility of drones in outdoor safety and rescue operations, particularly in expediting the search for missing persons. The real-time video feed provided by UAVs is essential for drone pilots to make timely and informed decisions. In these scenarios, it is critical that the video feed is both promptly available and of high quality. Our algorithm aligns well with these requirements by enhancing the overall encoding speed, thereby ensuring quicker end-to-end video transmission. Generally, our algorithm could be implemented to improve ME in various domains, like construction, agriculture, or mining [15–18].

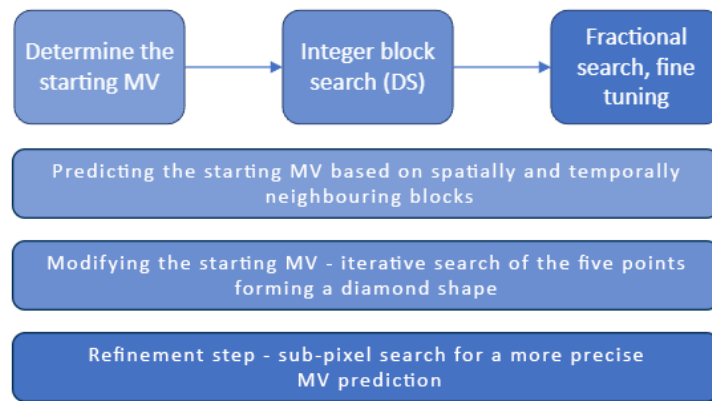
Classical forms of user interaction, such as joysticks, have been fundamental in RCV control, providing direct manipulation of the vehicle's movement. Modern systems continue to use these traditional controls while integrating advanced features like vibration feedback and ergonomic design to enhance user experience and control accuracy [19]. These systems are often complemented by templates or predefined paths that can guide the drone's motion, as seen in traditional remote-controlled aircraft systems. Semi-autonomous systems, where a human operator sets high-level goals or waypoints for the drone to follow, also play a crucial role. For example, recent advancements have shown the integration of intuitive control systems like handheld controllers with high-resolution displays and robust wireless communication, significantly enhancing operational efficiency and user interaction. Systems like those described by Yogi et al. [20] and Chen et al. [21], where operators set goal points and the UAV autonomously navigates to these points, demonstrate the balance between manual control and autonomy. Our algorithm could be applied in such systems, which require a video feed, and where drone movements are roughly known in advance. By knowing the estimated drone movements, a drone direction output file (discussed in further sections) could be generated and applied in ME.

Our research aims to address the challenge of reducing search locations in the ME process by leveraging user inputs from the controller. In previous work, we introduced a basic version of the MDIS algorithm, called User Input Search [22]. To our knowledge, no other research has optimized ME algorithms using controller input, making our approach unique and innovative. The main contributions are highlighted in the previous section.

### 3. Motion Dynamics Input Search Algorithm

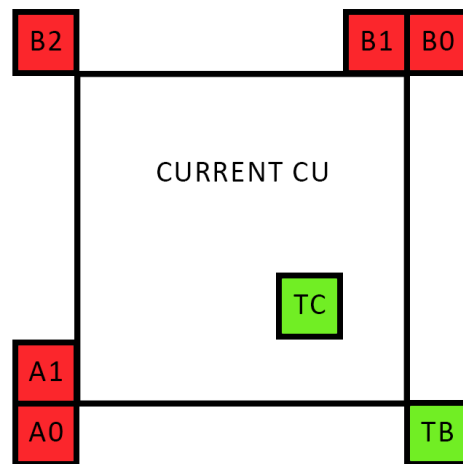
In order to implement the MDIS algorithm, an open-source High-Efficiency Video Coding (HEVC) video encoder, Kvazaar [23], was modified. Specifically, the DS algorithm, which is a part of Kvazaar's source code, was upgraded to incorporate MDIS. After conducting several preliminary tests, in which we encoded both simulator and real drone footage, we selected the DS algorithm that proved to be most suitable for our specific use case. Between DS, TZS, and HS, there was no clear winner in terms of encoding quality; the superior method varied depending on the specific content of the video being encoded, with the quality difference, measured in terms of PSNR, never exceeding 1.7%. In terms of compression ratio, TZS consistently demonstrated superior performance, with files encoded using TZS being on average 1.1% smaller than the smallest file produced by either DS or HS. Despite this, DS emerged as the fastest among the three, outperforming the TZS algorithm by an average of 56% and the HS algorithm by an average of 2.3%. DS's straightforward implementation in Kvazaar further solidified our choice. In summary, the DS and HS algorithms displayed comparable performance levels, whereas TZS appeared to prioritize a higher compression ratio while maintaining encoding quality, trading off increased encoding time.

Kvazaar's ME algorithm flow with the use of DS is depicted in Figure 1.



**Figure 1.** Flowchart depicting the widely recognized process flow of ME in the HEVC algorithm.

- The first step is to determine the starting motion vector (MV), which is done with Advanced Motion-Vector Prediction (AMVP). As described in [24], AMVP works by predicting the MV of a block in the current frame based on the MVs of spatially and temporally neighboring blocks. In Figure 2, the candidates A0, A1, B0, B1, and B2 represent spatial candidates and TB and TC represent temporal candidates. This prediction is then used as a starting point for the ME process, which can significantly reduce the search time and computational complexity. AMVP is particularly effective in scenarios where the motion in the video is relatively consistent or predictable.



**Figure 2.** HEVC AMVP scheme showing red rectangles as spatial candidates from the same frame and green rectangles as temporal candidates from different frames.

- The next step is finding the best MV, by modifying the starting MV. This process is carried out by conducting an iterative search of the five points forming a diamond shape. The center is then shifted to the point with the best match. This iteration continues until the best match is found at the center point, or the number of steps has exceeded the step limit. Searching for a single point refers to calculating the Sum of Absolute Differences (SAD) and comparing it with the current best SAD. The SAD is calculated as

$$SAD(A, B) = \sum_{i=1}^n \sum_{j=1}^m |A(i, j) - B(i, j)|, \tag{1}$$

where  $A$  and  $B$  are matrices representing pixel values, or simply blocks being compared, and  $n$  and  $m$  represent the dimensions of the blocks. If the calculated SAD is

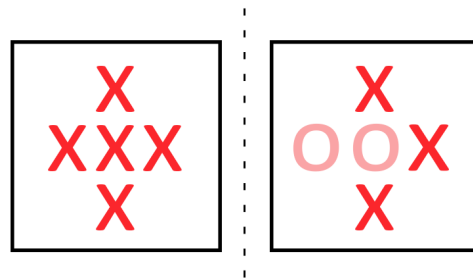


lower than the best (lowest) SAD, then the current point becomes the best match. This second step is done on integer precision, or in other words, on a larger scale.

- The last step is fine-tuning the MV, by performing a fractional search. A fractional search, also known as a sub-pixel search, is a refinement step in ME that allows for more precise MV prediction. After the best match has been found at an integer pixel position, a fractional search is conducted around this position at a sub-pixel level. This is typically achieved by using interpolation methods to create a high-resolution grid between the integer pixels. The search is then conducted on this grid to find the best match with even greater precision. This process can significantly improve the accuracy of the MV and the overall quality of the video compression.

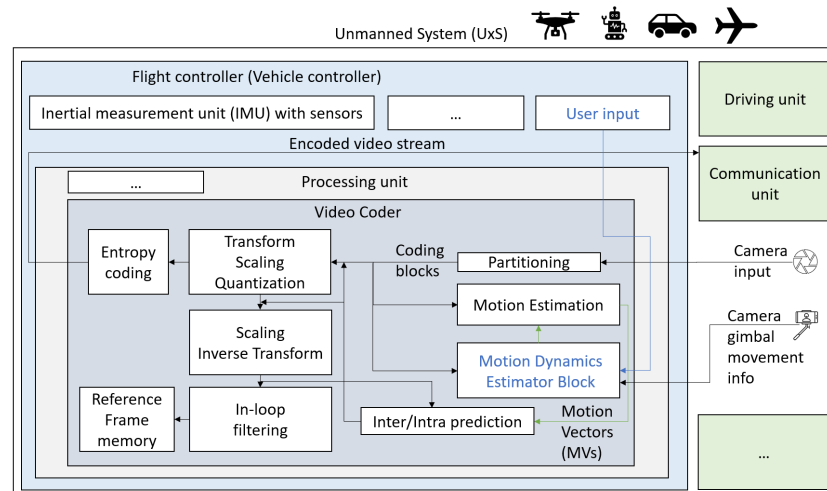
In MDIS, steps 1 and 3 are identical to those in the DS. The difference is in step 2, or the integer block search. The main idea behind the algorithm is to consider the motion dynamics estimation that is influenced by user inputs received from the controller, and also some other case-specific parameters. The information received from the controller includes which joysticks were pushed at which video frame, as well as the push intensities.

For instance, if the drone is directed to move upwards, then it makes sense to focus the search only on the upside direction. The same goes with every other direction. The rationale behind this focused approach is the predominantly global nature of the movement in drone-captured footage, especially evident in drone racing scenes. In such scenarios, the drone's motion effectively dictates the movement trajectory of nearly every pixel within the frame, rendering a comprehensive search across all potential diamond locations unnecessary. By knowing the drone's directional movement within the current frame, MDIS optimizes the ME process by selectively searching only relevant diamond locations, thereby reducing both ME and overall encoding times. Figure 3 illustrates the difference between searched locations in DS and MDIS algorithms—locations marked with a red 'X' are searched, while the locations marked with a faded red 'O' are not searched in MDIS. In this particular case, the drone appears to be moving to the right, as evidenced by the exclusion of the left and middle search locations. This is merely one example; search locations can be selectively filtered based on the type of movement being analyzed.



**Figure 3.** Comparative visualization of search patterns: DS algorithm blocks on the left, showcasing a traditional search approach, contrasted with MDIS algorithm blocks on the right, demonstrating the adaptive search based on user input and motion dynamics.

The Motion Dynamics Estimator Block in Figure 4 can be imagined as a 'black box' which estimates motion dynamics depending on several parameters, such as the mounting point of the camera (given as an offset from the origin of the vehicle's frame of reference), the default pose of its viewport and user inputs from the controller. Whether the camera is firmly fixed or mobile (attached to a gimbal) is also considered in the MDIS algorithm. The Motion Dynamics Estimator Block is a block that interpolates inside the standard video encoder and instructs the Motion Estimation block which locations will be searched. It is located in the processing unit of the Unmanned System (UxS).



**Figure 4.** Simplified diagram of a video coding framework incorporating the Motion Dynamics Estimator Block, illustrating its role in enhancing ME.

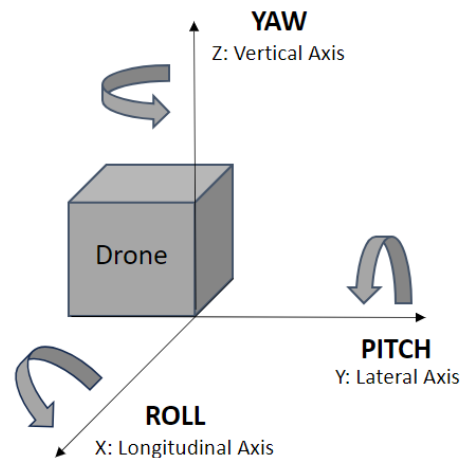
Given that tests in this study were performed on simulator footage as well as on real 4K drone sequences, the Motion Dynamics Estimator Block was customized to suit these particular scenarios.

To recapitulate, the semantic description of the MDIS algorithm is as follows:

- Determine the starting MV—Use the AMVP scheme.
- Collect data—Obtain data from the Motion Dynamics Estimator block (globally defined parameters mentioned earlier and user input from the controller).
- Process data—Analyze the collected data to determine the appropriate search model for the current frame (detailed in Sections 3.2 and 3.3).
- Perform integer-block search—Use the determined search model to conduct the integer-block search.
- Refine the MV—Apply fractional search to refine the MV.

### 3.1. Understanding Image Changes Based on Drone Movements

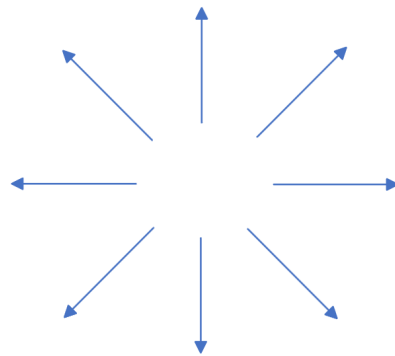
The movement of a vehicle in motion can be described by a combination of linear and angular motion in different dimensions. When it comes to drones, linear motion can be decomposed into velocity vectors, each parallel to one of the three axes (x, y, and z), while the angular motion can be decomposed into three rotational components, each around one of the three axes (Figure 5).



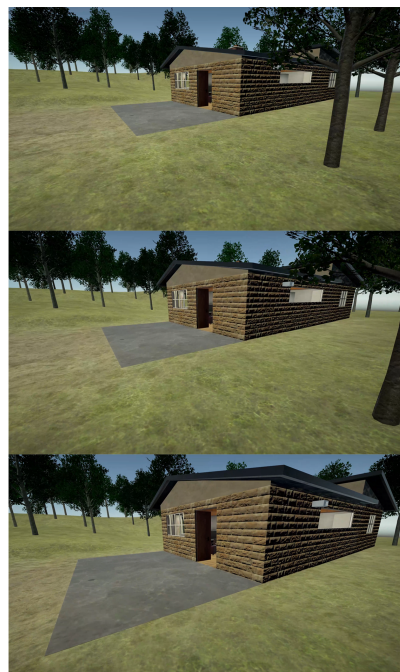
**Figure 5.** Diagram illustrating drone movement: axes denote the direction of linear motion, and arrows indicate rotational movements (roll, pitch, and yaw).

Each rotational component of an aerial vehicle has a specific name: the rotation around the Longitudinal Axis of the vehicle (axis X) is called “roll”, the rotation around the Lateral Axis of the vehicle (axis Y) is called “pitch”, while the rotation around the axis vertical to the vehicle (axis Z) is called “yaw”.

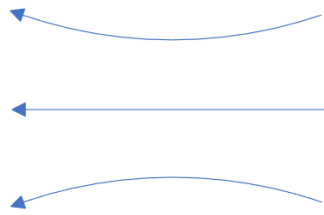
The most common movement in drone scenes is the forward movement, which requires the drone to slightly pitch. When the drone moves forward, pixels tend to spread away from the center of the image (Figure 6). The further away the pixel is from the center, the more it will expand (move toward the corner of the image) in the next frame. An example of a drone forward motion in a simulator can be seen in Figure 7. Similarly, when the drone moves backward, pixels compress toward the center of the image. The next movement type is the yaw rotation movement. When the drone rotates to the right, pixels tend to follow the movement scheme illustrated in Figure 8. Pixels located closer to the top or bottom of the image exhibit a more pronounced arc in their movement, while pixels located in the center of the image move in a straight line. Of course, when the drone is rotating to the left, the pixels move to the right, and vice versa for the right rotation. Additionally, the arc angle depends on the camera lens characteristics.



**Figure 6.** Visual representation illustrating pixel displacement towards the edges during forward drone motion.



**Figure 7.** Forward drone motion example, highlighting the effect of pixels expanding from the center toward the image edges.



**Figure 8.** Visual representation of pixel trajectories during a drone’s right yaw rotation.

Pixel movements are much simpler when it comes to left–right and up–down drone motion. When the drone moves to the left it must perform a slight roll rotation first, and afterward, the pixels move straight to the right. If the drone is in an upward motion, the pixels move straight downwards. Hence, no complex search models were created for the roll and throttle movements. Simply, only the diamond locations that match the current direction of the drone motion are searched.

3.2. Search Model Creation

The MDIS model was designed with the described pixel movements in mind. The model creation starts by segmenting the image into nine equal regions. This is achieved by partitioning both the vertical and horizontal axes into three equal segments, resulting in a 3x3 grid. The next step is determining which diamond locations should be searched in each of the nine segments. We developed a few search model versions which varied in several parameters. For example, in one version of MDIS, the diamond shape is extended so that it also includes corner locations (Figure 9). Different versions also varied in parameters such as the weight in the weighted probability function, or the probability threshold, described later on.



**Figure 9.** Search locations in Diamond Search (left) and potential search locations in Motion Dynamics Input Search (right).

The general model design process will be described through the forward motion model. The search model for left–right motion is created analogously. Figure 10 shows the forward motion search model in MDIS. The whole box represents a single image, which is segmented into nine segments, as explained earlier. In each segment, a modified diamond shape is drawn, where locations marked with ‘X’ represent locations that should be searched, and locations marked with ‘0’ represent locations that should be skipped. In other words, depending on the block location in the image, some search locations will be included and some will be excluded.

For example, in the forward motion search model shown in Figure 10, blocks that belong in the center grid segment are searched for the most similar block in all possible directions. Similarly, blocks in the upper-left grid segment are only searched in the right and bottom directions. In this particular case, the diagonal search locations are also shown, which can be excluded in some search models.



**Figure 10.** Discrete forward motion search model, depicting selective search patterns based on block positions in an image.

We call this model the Discrete search model (DSM). While MDIS could be implemented following DSM, we decided to improve it by creating the so-called Continuous Search Model (CSM). Instead of simply deciding whether a certain location should be searched depending on the block location, the CSM calculates the search probability for each search location.

The search probability is calculated based on the block coordinates, and each search location uses a different function, inspired by the DSM. All the functions for each search location in a forward motion are displayed in Figure 11. Every function was crafted to emulate the DSM location-filtering approach, with an additional goal of functions having minimal computational complexity. The variable  $x$  denotes the  $x$ -coordinate of the upper left corner of the current block, while  $y$  denotes the  $y$ -coordinate of the same corner.  $w$  stands for the picture width, and  $h$  represents the picture height. Each grid segment refers to one search location. The small grayscale images inside each grid segment represent the 2D probability functions—the brighter the area, the higher the probability. These grayscale images correspond to images themselves, in terms of resolution. Let us look at the upper left search location for an example—the closer the block is to the bottom right corner of the image, the higher the probability that the upper left search location will be searched. This can be seen from the small grayscale image in the upper left grid segment—the closer the block is to the bottom right corner, the brighter it is. Search probabilities are calculated for each block independently. At the start of the MDIS algorithm, the probability threshold is defined (0.5 for instance), and each search location that has a search probability higher than the probability threshold will be searched.

$P(x,y) = \frac{x+y}{w+h}$	$P(x,y) = \frac{y}{h}$	$P(x,y) = \frac{ x-w +y}{w+h}$
$P(x,y) = \frac{x}{w}$	$P(x,y) = 1 - \frac{\sqrt{(x-\frac{w}{2})^2 + (y-\frac{h}{2})^2}}{\sqrt{(\frac{w}{2})^2 + (\frac{h}{2})^2}}$	$P(x,y) = 1 - \frac{x}{w}$
$P(x,y) = \frac{ y-h +x}{w+h}$	$P(x,y) = 1 - \frac{y}{h}$	$P(x,y) = 1 - \frac{x+y}{w+h}$

**Figure 11.** Set of probability functions determining search locations in forward motion, based on block position within the image.



Figure 12 is a snippet of MVs generated by the DS algorithm, in a case where the drone moved forward. This phenomenon consistently manifests whenever the drone advances, illustrating that MVs accurately adhere to the forward motion model outlined by MDIS. All the vectors point toward the center of the image. One very important observation, which was already discussed in the previous subsection, is that the further away the pixel is from the center, the more it will move towards the edge of the image. This is also considered in the search model. When searching for the most similar block, the search range is multiplied by the weighted probability function—the same probability function that was calculated earlier. This is a great benefit of the CSM in comparison with the DSM. Not only do the probabilities determine whether the search location should be considered at all, but they also dictate the search range. This can also be seen in Figure 12. Notice how blocks that are further away from the center have bigger MVs.

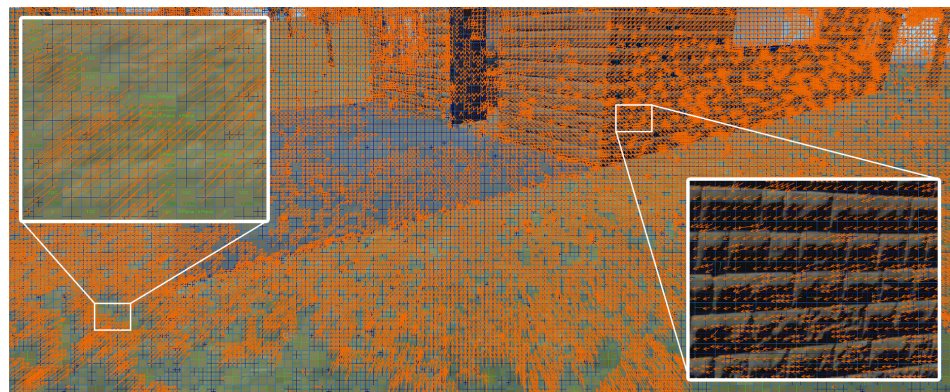


Figure 12. Visualization of motion vectors resulting from forward motion, showcasing directional pixel displacement.

Figure 13 shows the DSM for the left drone motion. The CSM was designed as well and is shown in Figure 14. Again, all the functions were designed with the intention of minimizing computational complexity, hence some functions have sharp transitions in probability values, even though there might be a better probability function available. The model for the right drone motion is analogous. The function *local\_thresh* sets all the values within a specified region to 0. The function *thresh* assigns a value of 0 to all elements that fall below a certain threshold. The parameter *A* from function (2) is a two-dimensional area representing a part of an image, and the parameter *T* is a scalar value.

$$local\_thresh(x) = \begin{cases} 0 & \text{if } x \in A \\ x & \text{else} \end{cases} \tag{2}$$

$$thresh(x) = \begin{cases} 0 & \text{if } x < T \\ x & \text{if } x \geq T \end{cases} \tag{3}$$

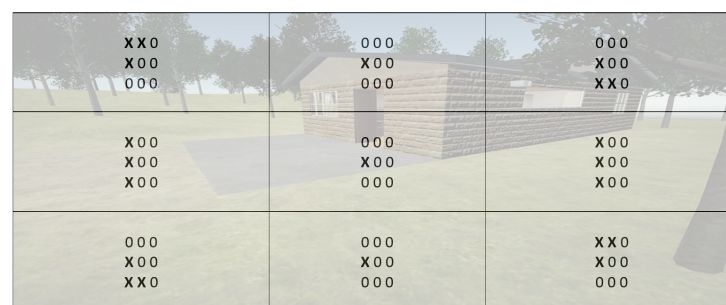
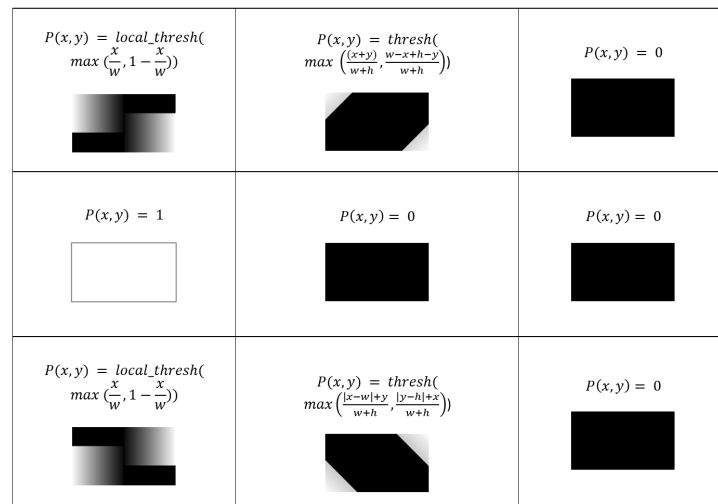


Figure 13. Discrete search model for leftward drone motion, illustrating search pattern adjustments based on an image block position.





**Figure 14.** Set of probability functions determining search locations in left motion, based on block position within the image.

### 3.3. Determining Search Directions Based on the Received User Input

Even though a system has been developed that allows users to try the MDIS algorithm in real time, the tests for this research were made using pre-recorded footage and accompanying log files which include user input. The input received from the user has the following format:

*Right Stick Vertical* – [3.319998] → – 0.3877063

In this example, Right Stick Vertical, which corresponds to the forward motion, was pushed with the intensity of –0.3877063 at the second 3.319998. Generally, the received input follows the following format:

*STICK* – [TIMESTAMP] → *INTENSITY*

The timestamp can easily be converted into the frame number, and the intensities range from –1 to 1. The idea is to use the received input to generate a textual file that contains numbers from ‘0’ to ‘8’, which correspond to all the possible movements, starting with ‘0’, which means undefined (perform a normal DS), ‘1’ for forward motion, ‘2’ for backward motion, ‘3’ for moving to the left, and so on (Table 1). The MDIS algorithm then reads those numbers and uses a specific search model depending on the read number.

**Table 1.** Mapping of movement codes to descriptions.

Movement Code	Description
0	Undefined (Normal Diamond Search)
1	Forward motion
2	Backward motion
3	Moving to the left
4	Moving to the right
5	Upward motion
6	Downward motion
7	Rotating to the left
8	Rotating to the right

The accuracy of the utilized motion dynamics estimation is ensured by relying on the same joystick input data already used for controlling the drone, which is handled and secured by the drone’s manufacturers. Since this information is integral to the drone’s

operation, it is inherently accurate and reliable. Even if an error occurs, it would not cause a significant issue because such errors are rare and because ME uses the AMVP scheme described earlier in this section, which is robust to rare disturbances. For example, we tested this by intentionally inserting a random value in the generated textual file at every 20th character location, and the impact on ME quality was negligible.

Transforming the received user input data into the desired formatted output presents a complex challenge. It is not sufficient to merely identify the joystick with the greatest push intensity at a given frame and assume that it determines the final search model. This approach would not yield optimal results. Consider the following scenario: a drone is stationary and begins to accelerate forward. Naturally, the highest intensity would be for the Right Stick Vertical. It might seem intuitive to immediately start using the forward motion search model. However, it is important to note that due to physical forces, the drone initially leans downward (pitch) as it starts to accelerate. This downward tilt can be leveraged by employing a combination of the downward and forward motion search models. Relying solely on the standard forward motion model during the several initial frames of drone acceleration would not yield optimal results. Similarly, when the drone starts moving left or right, it first slightly tilts to the side (roll).

While integrating additional sensor data such as radar or laser data could potentially enhance the precision of motion dynamics estimation, our primary focus is on leveraging joystick input data. This decision is based on the following factors:

- Data availability and reliability: joystick input data are readily available and reliably processed by the drone's control system. These data are directly tied to the user's commands and provide immediate feedback on the intended movements.
- Integration complexity: incorporating radar or laser data would introduce additional complexity into the system. These sensors require calibration, synchronization, and integration with the existing control and processing framework. This can be challenging and may necessitate significant changes to the hardware and software architecture.

In scenarios where user inputs are assisted by systems that modify them, such as stabilization systems, the input data may not directly map to the proper movement code. For instance, if a stabilization system smoothens abrupt joystick movements to prevent sudden jerks, the input data might be mapped incorrectly, because incorrect data would be used in movement code calculation. The MDIS algorithm should match the stabilization mechanism's adjustments and use these modified input data when calculating movement codes, as it more accurately represents the actual control inputs being applied to the drone.

Another important factor to consider is that our primary interest lies in the direction of acceleration, not just the drone's movement. This is due to the fact that MVs are refined based on a starting MV. For instance, if the drone is moving forward but begins to decelerate, the backward motion model should be employed, despite the drone's continued forward motion. The same goes with all other movement types, not just the forward/backward motion. Very small accelerations and decelerations are not problematic, because the fractional pixel search, which is unchanged in MDIS, can handle small changes in MVs. Specifically, this last case of small changes in acceleration refers to cases in which the user's fingers aren't completely steady, so the joystick push intensities can slightly vary from frame to frame.

The MDIS version tested in this article determined the final movement codes by following the following described principles:

- Consecutive frames condition—A joystick had to maintain the highest push intensity for several consecutive frames (eight in the tested MDIS version). Only after this consistency was maintained over several frames did the corresponding movement code start being written to the final direction output file.
- Acceleration threshold condition—To ensure we are capturing acceleration, the stick push intensity in the current frame had to differ from the previous frame by a specific threshold (e.g., 0.05). If any of these conditions were not met, a '0' was written to the final directions output file.

In the current MDIS implementation, only the nine search modes are implemented. The combinations of multiple models, such as forward and downward motion, explained earlier, are not yet implemented. Whenever it is unclear which search model should be used, a '0' is written in the textual movement file, which means a normal DS is used for that frame. This is an improvement that can be implemented in future research.

#### 4. Experimental Results

Over 70 test cases, encompassing all types of movements, were carried out using footage from a simulator and real drone footage. Pre-recorded data were used for verifying the algorithm to ensure consistent and repeatable testing conditions. Moreover, pre-recorded data are in the highest-quality raw format so that there are no coding artifacts in the input video. This allows us to thoroughly evaluate the algorithm's performance under controlled scenarios, using both simulator footage with precise user input data and real drone footage. By doing so, we can accurately measure and compare the performance of the MDIS algorithm against the DS algorithm without the variability that might arise from live data collection.

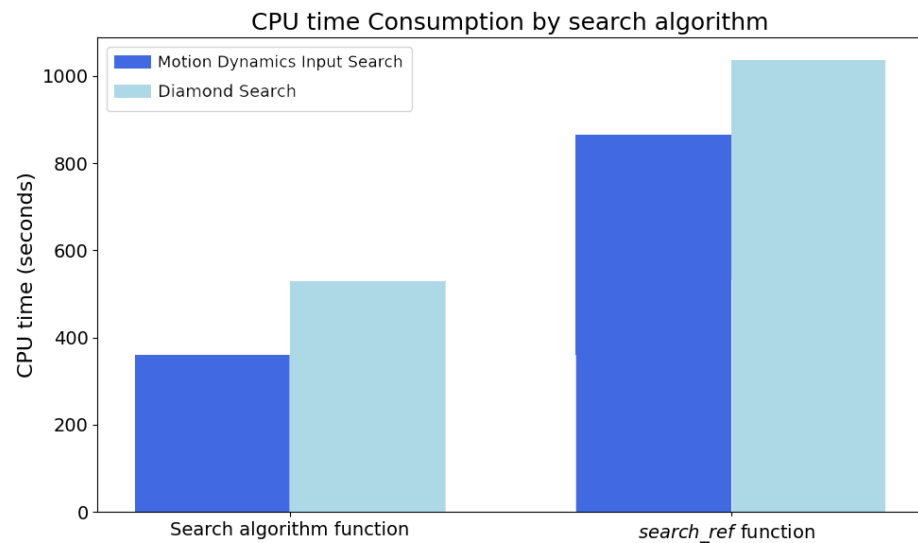
The simulator sequence under test spans 46 s and is presented in a 4K resolution at 60 frames per second. Real drone footage consists of 4 short sequences (4–20 s per sequence), also in 4K resolution, at 60 frames per second. The tests were performed on a 64-bit Windows personal computer with 16 GB of RAM, 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60 GHz processor (Intel, Santa Clara, CA, USA), and an NVIDIA GeForce GTX 1650 graphics card (NVIDIA, Santa Clara, CA, USA). The simulator footage was given particular attention because it provided precise user input data from the controller that corresponded to the footage. For the real drone footage, we manually created the textual user input files. This was done by individually inspecting each sequence and tracking the drone movements. Nevertheless, the approach of manually creating user input files for the real drone footage is also valid, because the data from the controller undergo the same processing as the approximated manual input, ensuring a consistent methodology across both types of footage.

In order to precisely measure the encoding times and times occupied by each function separately, Intel's VTune Profiler software version 2023.0.0 was used [25]. The Flame Graph analysis was used the most, since it shows exactly how much CPU time each function occupied. In order to optimize each search model independently, the simulator footage was segmented into several shorter sequences, each approximately 300 frames long. Each of these sequences was characterized by a single specific type of movement, namely left–right rotation, forwards–backwards motion, left–right motion, or upwards–downwards motion.

Several versions of MDIS were tested, and the following results refer to the best version, which will be explained in detail.

Figure 15 presents a comparison chart illustrating CPU time consumption, derived from Flame graph analysis, for encoding the entire simulator sequence using the MDIS and DS algorithms.

The bars on the left side of the chart represent the CPU time used exclusively for the search algorithm functions. In contrast, the bars on the right side illustrate the CPU time consumed by the broader *search\_ref* function, which includes not only the search functions but also other operations like early termination and selecting starting points, which are key elements of the whole ME process. This comparison shows that MDIS speeds up the overall ME process and does not affect the other ME process operations negatively. It is noteworthy that all other operations were executed within a similar time frame for both the DS and MDIS algorithms, confirming that MDIS is the primary factor in achieving faster search times. The DS algorithm accounted for 529 CPU seconds in total, which makes up 3.5% of the total encoding time. Conversely, the MDIS function accounted for 360 CPU seconds in total, representing 2.3% of the total encoding time. In a percentage, the MDIS algorithm is approximately 32% faster than the DS algorithm.



**Figure 15.** Chart showing CPU time consumption per functions for the entire simulator sequence encoding via MDIS and DS algorithms.

In addition to accelerating the encoding speed, the MDIS algorithm also maintains the same video quality as the DS algorithm, as evidenced by PSNR values which were consistently within a  $\pm 0.6\%$  range. Furthermore, MDIS offers a superior compression ratio. While the video encoded with DS amounts to 264 MB, the MDIS-encoded video is smaller, with a size of 257 MB.

Table 2 represents encoding data for real drone footage. Encoded video file size refers to the final video file size of the corresponding real drone sequence, expressed in MB. Search time refers to the CPU time consumed solely by the search function—by MDIS or DS. PSNR is calculated using FFmpeg [26] by comparing the final encoded video and the original, raw video. Even better results were achieved while testing on real drone footage, but it is worth noticing that real drone footage was filmed using only basic movements. No complex movements, such as combined forward motion and left–right rotation, were used. In real drone footage, MDIS was up to 4.25 times faster than the DS while preserving video quality and compression rate.

**Table 2.** Comparison of encoded file size, encoding time, and encoding quality for DS and MDIS in case of encoding 4 different real drone sequences.

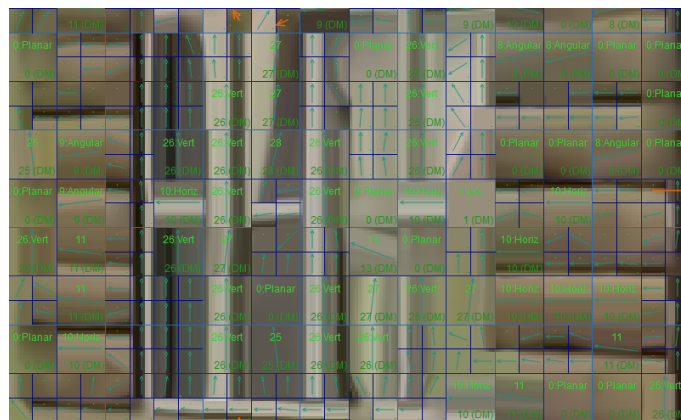
Algorithm	Sequence Number	Encoded Video File Size [MB]	Search Time [CPU s] (Less Is Better)	PSNR [dB] (More Is Better)
DS	1	10.2	1.53	45.36
	2	38.1	38.70	43.08
	3	19.7	1.11	43.68
	4	9.87	12.37	44.92
MDIS	1	10.2	0.36	45.44
	2	37.9	11.02	43.08
	3	19.7	0.35	43.69
	4	9.80	3.07	44.85

This version of MDIS excluded the extra corner search locations that were added in one of the MDIS versions (Figure 9). Including the corners resulted in a slightly smaller video file size, but at the cost of a longer encoding time. The reduction in file size was minimal (around 1 MB), while the MDIS execution time increased by approximately 15%. Therefore, the disadvantages of using the corner search locations outweigh the benefits, at least in this case of encoding the simulator footage. The search range weight factor used in

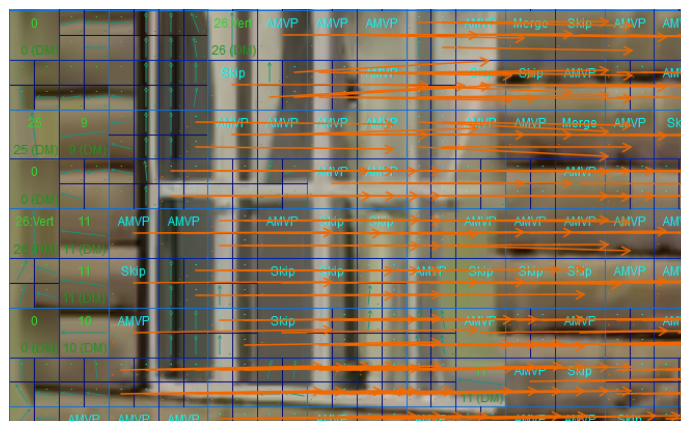
this version was 2. This refers to the search range that is modified based on the calculated search probabilities. This would mean that a block that had a search probability of 1.0 had the search range doubled. It is worth noting that the process of calculating the search probabilities has a negligible impact on the encoding time.

We found it very interesting how MDIS resulted in smaller video file sizes. Hence, a detailed inspection was performed using VQAnalyzer software version 7.2.0.73697 [27]. Oftentimes, MDIS guides the ME process in the right direction. In DS, there can be cases where blocks ‘accidentally’ find the most similar blocks on search locations that do not align with the drone’s movement. For instance, during the first iteration of DS, the center might shift to the left search location even though the drone is moving right. While the left search location may indeed be the best match in the first iteration, DS might not find better blocks in subsequent iterations. In contrast, MDIS, which might not find the best match in the first iteration, often identifies much better blocks in later iterations.

This discrepancy can also impact neighboring blocks due to the process of selecting the initial MV. At times, DS might conclude that the best match is in the left search location, causing neighboring blocks to start their search from that point, which can lead to ‘dead-ends’. Although this can also occur with MDIS, it happens less frequently due to the enforced search direction, which is grounded in theoretical principles. Figures 16 and 17 show this effect—they are snippets of the same video frame, but one is from the video encoded with DS, and one is from the video encoded with MDIS. There are much fewer MVs in the case of DS, which makes the DS frame much bigger. In this case, the frame encoded with DS takes 2.27 MB, while the frame encoded with MDIS takes 1.95 MB.



**Figure 16.** Frame snippet encoded using the DS algorithm, illustrating an area where no MVs were found in the ME process.



**Figure 17.** Identical frame snippet as depicted in Figure 16, highlighting numerous MVs (in orange color) detected during the ME process.

A similar situation where MDIS identifies more MVs than DS arises due to the extended search range facilitated by search probabilities. With DS, blocks located near the edges of frames often end up without assigned MVs. However, MDIS more frequently succeeds in finding MVs for such blocks due to its extended search range capability.

The final search directions file was generated using a Python version 3.9.13 script. The script is designed to process the input files and generate an output file containing the final search directions for the MDIS algorithm. The four input files follow the scheme explained earlier, and each file corresponds to one joystick movement. The script checks which joystick had the highest push intensity at that moment, and writes only that information in a separate file. Later, that newly created file is processed in the following way. If the push intensity is below 0.1, write a '0' to the final directions file. Otherwise, compare the intensity and movement category with the previous value, and follow the logic described earlier in this research. For example, if the movement category is forward motion, and high acceleration is detected, then still write a '0' in the output file because the drone is currently leaning forward, and the classical forward motion model can not yet be applied. Only once the high acceleration stops, start applying the forward motion model, or in other words, start writing the corresponding number to the output file.

## 5. Conclusions and Future Work

This research has presented a novel ME algorithm, MDIS, designed to enhance video compression for RCVs, with a particular focus on UAVs. The MDIS algorithm builds upon the well-established DS algorithm, incorporating information from vehicle motion dynamics estimation and real-time user interaction data from the controller to guide the search process. This approach has been shown to significantly improve the efficiency and accuracy of ME, reducing the computational complexity and latency of video transmission, which is of paramount importance in applications such as drone racing, FPV systems, construction, agriculture, or mining. Overall, the MDIS algorithm's improvements in video encoding speed and quality are crucial across these diverse applications, enhancing operational efficiency, safety, and decision-making processes.

The MDIS algorithm has been thoroughly tested using both simulator and real drone footage. The results have demonstrated that MDIS can reduce the encoding time by approximately 32% compared to the DS algorithm in the best-case scenario while maintaining the same video quality. Furthermore, MDIS has been shown to provide a superior compression ratio, resulting in smaller video file sizes. This is a significant achievement, as it allows for more an efficient use of network bandwidth and storage resources. This also means MDIS reduces overall end-to-end video transmission latency and can increase the final frame rate of the displayed video, enhancing the pilot's user experience.

This research has also highlighted the importance of considering the direction of acceleration, rather than just the direction of movement, in the ME process. This insight has been incorporated into the MDIS algorithm, allowing it to more accurately predict the MVs and thus improve the efficiency of the search process.

MDIS presents a significant enhancement over DS through its improved search range modification, achieved by incorporating search probabilities. This enhancement enables the MDIS algorithm to identify a greater number of MVs compared to DS. In DS, blocks situated near the frame edges frequently lack assigned MVs. In contrast, MDIS demonstrates a higher success rate in obtaining MVs for such blocks, thanks to its expanded search range capability.

Despite these promising results, there are several potential avenues for further research and improvement. One such area is the implementation of combined search models in MDIS. Currently, MDIS uses a single search model based on the dominant movement type. However, in certain scenarios, such as when the drone is accelerating or moving forward and rotating at the same time, a combination of different search models may yield better results. Implementing this feature in MDIS could further improve its performance.



Another area for future work is the refinement of the process for determining final search directions based on user input. The current approach, while effective, could potentially be improved by incorporating more sophisticated machine learning techniques to better predict the final search directions.

Moreover, the process of early termination, which was not mentioned in this research before, could be modified to follow the logic of MDIS. The purpose of early termination in ME is to optimize computational efficiency. Instead of exhaustively searching through all possible MVs, the encoder may terminate the search early if it determines that further exploration is unlikely to yield a significantly better result. This is based on heuristics and thresholds calculated during the ME process. Early termination takes up a lot of CPU time because it performs an HS in a small area, similar to the DS, but at sub-pixel-level precision. The HS search locations could also be modified to follow the same logic depending on the user input.

In conclusion, the MDIS algorithm represents a significant step forward in the field of video compression for RCVs. By intelligently leveraging user input data, MDIS has the potential to greatly enhance the performance and user experience of applications such as drone racing and FPV systems. As such, it provides a promising foundation for future research and development in this exciting area.

**Author Contributions:** Conceptualization, D.H.; methodology, J.B. and D.H.; software, J.B.; validation, J.B. and D.H.; formal analysis, J.B. and D.H.; investigation, J.B.; resources, D.H.; data curation, J.B.; writing—original draft preparation, J.B., D.H. and H.M.; writing—review and editing, J.B., D.H. and H.M.; visualization, J.B. and D.H.; supervision, D.H.; project administration, D.H.; funding acquisition, D.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work has been supported in part by the project KK.01.2.1.02.0054 Development of ultra-low latency video signal transmission device and NPOO.C3.2.R3-I1.01.0430 Reducing energy consumption and latency during video compression on drones by knowing the direction of movement, financed by the EU from the European Regional Development Fund.

**Data Availability Statement:** The raw data supporting the conclusions of this article will be made available by the authors on request.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Guo, J.; Li, X.; Lv, Z.; Yang, Y.; Li, L. Design of Real-time Video Transmission System for Drone Reliability. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *790*, 012004. [[CrossRef](#)]
2. Khan, N.A.; Brohi, S.N.; Jhanjhi, N. UAV's Applications, Architecture, Security Issues and Attack Scenarios: A Survey. *Lect. Notes Netw. Syst.* **2020**, *118*, 753–760. [[CrossRef](#)]
3. Singh, S.; Lee, H.W.; Tran, T.X.; Zhou, Y.; Sichitiu, M.L.; Guvenc, I.; Bhuyan, A. FPV Video Adaptation for UAV Collision Avoidance. *IEEE Open J. Commun. Soc.* **2021**, *2*, 2095–2110. [[CrossRef](#)]
4. Paramkusam, A.V.; Darimireddy, N.K.; Sridhar, B.; Siripurapu, S. All Directional Search Motion Estimation Algorithm. *Electronics* **2022**, *11*, 3736. [[CrossRef](#)]
5. Wang, J.; Yang, J. An Improved Fast Motion Estimation Algorithm in H.266/VVC. *J. Innov. Soc. Sci. Res.* **2022**. [[CrossRef](#)] [[PubMed](#)]
6. Hassan, K.H.; Butt, S.A. Motion Estimation in HEVC/H.265: Metaheuristic Approach to Improve the Efficiency. *Eng. Proc.* **2021**, *12*, 59. [[CrossRef](#)]
7. Gao, L.; Dong, S.; Wang, W.; Wang, R.; Gao, W. A novel integer-pixel motion estimation algorithm based on quadratic prediction. In Proceedings of the 2015 International Conference on Image Processing, ICIP, Quebec City, QC, Canada, 27–30 September 2015; pp. 2810–2814. [[CrossRef](#)]
8. Kibeya, H.; Belghith, F.; Loukil, H.; Ayed, M.A.B.; Masmoudi, N. TZSearch pattern search improvement for HEVC motion estimation modules. In Proceedings of the 2014 1st International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2014, Sousse, Tunisia, 17–19 March 2014; pp. 95–99. [[CrossRef](#)]
9. Pan, Z.; Lei, J.; Zhang, Y.; Wang, F.L. Adaptive fractional-pixel motion estimation skipped algorithm for efficient HEVC motion estimation. In *ACM Transactions on Multimedia Computing, Communications and Applications*; Association for Computing Machinery: New York, NY, USA, 2018; Volume 14. [[CrossRef](#)]
10. Nalluri, P.; Alves, L.N.; Navarro, A. Complexity reduction methods for fast motion estimation in HEVC. *Signal Process. Image Commun.* **2015**, *39*, 280–292. [[CrossRef](#)]

11. Yu, K.; Wegele, T.; Ostler, D.; Wilhelm, D.; Feußner, H. EyeRobot: Enabling telemedicine using a robot arm and a head-mounted display. *Curr. Dir. Biomed. Eng.* **2020**, *6*, 20200019. [[CrossRef](#)]
12. Yoo, M.; Na, Y.; Jo, K.; Song, H.; Kim, G.; Yun, J.; Kim, S.; Moon, C. Motion Estimation and Hand Gesture Recognition-Based Human—UAV Interaction Approach in Real Time. *Sensors* **2022**, *22*, 2513. [[CrossRef](#)] [[PubMed](#)]
13. Varga, B.; Doer, C.; Trommer, G.F.; Hohmann, S. Validation of a Limit Ellipsis Controller for Rescue Drones. In Proceedings of the SACI 2022—IEEE 16th International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, 25–28 May 2022; pp. 55–60. [[CrossRef](#)]
14. Tuśnio, N.; Wróblewski, W. The Efficiency of Drones Usage for Safety and Rescue Operations in an Open Area: A Case from Poland. *Sustainability* **2021**, *14*, 327. [[CrossRef](#)]
15. Umar, T. Applications of drones for safety inspection in the Gulf Cooperation Council construction. *Eng. Constr. Archit. Manag.* **2021**, *28*, 2337–2360. [[CrossRef](#)]
16. Nwaogu, J.M.; Yang, Y.; Chan, A.P.; Chi, H.L. Application of drones in the architecture, engineering, and construction (AEC) industry. *Autom. Constr.* **2023**, *150*, 104827. [[CrossRef](#)]
17. Debangshi, U. Drones—Applications in Agriculture. *Chron. Bioresour. Manag.* **2021**, *5*, 115–120.
18. Padró, J.C.; Cardozo, J.; Montero, P.; Ruiz-Carulla, R.; Alcañiz, J.M.; Serra, D.; Carabassa, V. Drone-Based Identification of Erosive Processes in Open-Pit Mining Restored Areas. *Land* **2022**, *11*, 212. [[CrossRef](#)]
19. ATLAS UAS. Available online: <https://www.atlasuas.com/news/atlas-ecosystem-and-small-uav-technologies-2023-results> (accessed on 6 August 2024).
20. Yogi, N.; Kumar, N. Control Systems for Unmanned Aerial Vehicles: Advancement and Challenges. *Lect. Notes Mech. Eng.* **2024**, *162*, 107–118. [[CrossRef](#)]
21. Chen, Y.; Li, N.; Zeng, W.; Wu, Y. Curved Path Following Control for a Small Fixed-Wing UAV with Parameters Adaptation. *Appl. Sci.* **2022**, *12*, 4187. [[CrossRef](#)]
22. Benjak, J.; Hofman, D. User Input Search—Custom Motion Estimation Algorithm Optimized for UAVs. In Proceedings of the 2023 46th ICT and Electronics Convention, MIPRO 2023, Opatija, Croatia, 22–26 May 2023; pp. 506–510. [[CrossRef](#)]
23. Viitanen, M.; Koivula, A.; Lemmetti, A.; Ylä-Outinen, A.; Vanne, J.; Hämäläinen, T.D. Kvazaar: Open-source HEVC/H.265 encoder. In Proceedings of the MM 2016—Proceedings of the 2016 ACM Multimedia Conference, Amsterdam, The Netherlands, 15–19 October 2016; pp. 1179–1182. [[CrossRef](#)]
24. Yu, Q.; Zhao, L.; Ma, S. Parallel AMVP candidate list construction for HEVC. In Proceedings of the 2012 IEEE Visual Communications and Image Processing, VCIP 2012, San Diego, CA, USA, 27–30 November 2012. [[CrossRef](#)]
25. Fix Performance Bottlenecks with Intel® VTune™ Profiler. Available online: <https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html#gs.ohbu13> (accessed on 19 July 2023).
26. FFmpeg. Available online: <https://ffmpeg.org/> (accessed on 2 August 2023).
27. Video Analyzer and Streaming Tester Software—VQ Analyzer. Available online: <https://vicuesoft.com/vq-analyzer/> (accessed on 19 July 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.