electronics

MDPI

Article

A Fast Gradient Iterative Affine Motion Estimation Algorithm Based on Edge Detection for Versatile Video Coding

Jingping Hong, Zhihong Dong *, Xue Zhang, Nannan Song and Peng Cao

College of Information Engineering, Beijing Institute of Graphic Communication, Beijing 102600, China; hongjingping2022@163.com (J.H.); zxfk9158@163.com (X.Z.); nancyyo1881@gmail.com (N.S.); pc@bigc.edu.cn (P.C.)

* Correspondence: dongzhihong@bigc.edu.cn

Abstract: In the Versatile Video Coding (VVC) standard, affine motion models have been applied to enhance the resolution of complex motion patterns. However, due to the high computational complexity involved in affine motion estimation, real-time video processing applications face significant challenges. This paper focuses on optimizing affine motion estimation algorithms in the VVC environment and proposes a fast gradient iterative algorithm based on edge detection for efficient computation. Firstly, we establish judging conditions during the construction of affine motion candidate lists to streamline the redundant judging process. Secondly, we employ the Canny edge detection method for gradient assessment in the affine motion estimation process, thereby enhancing the iteration speed of affine motion vectors. The experimentalresults show that the encoding time of the affine motion estimation algorithm is about 15–35% lower than the overall encoding time of the anchor algorithm encoder, the average encoding time of the affine motion estimation part of the inter-frame prediction part is reduced by 24.79%, and the peak signal-to-noise ratio (PSNR) is only reduced by 0.04.

Keywords: versatile video coding; inter-prediction; affine motion estimation; edge detection

Citation: Hong, J.; Dong, Z.; Zhang, X.; Song, N.; Cao, P. A Fast Gradient Iterative Affine Motion Estimation Algorithm Based on Edge Detection for Versatile Video Coding. *Electronics* **2023**, *12*, 3414. https://doi.org/10.3390/electronics12163414

Academic Editor: Stefanos Kollias

Received: 14 July 2023
Revised: 6 August 2023
Accepted: 9 August 2023
Published: 11 August 2023

1. Introduction

With the increasing amount of video data and the growing demand for high-quality video services, efficient video coding technology plays a crucial role in reducing bandwidth requirements and improving video compression performance. The H.266/VVC standard is the latest video coding standard developed by the Joint Video Expert Group (JVET) [1–3], aiming to provide significantly improved coding efficiency compared to previous standards and achieve a better coding performance than high-efficiency video coding (HEVC) [4–7]. The goal of the H.266/VVC standard is to provide higher compression rates under the same video quality. To achieve this goal, H.266/VVC adopts a series of innovative technologies, including the fast affine motion estimation algorithm (AME), advanced motion vector prediction (AMVP), prediction value correction based on the optical flow field, and inter-frame weighted prediction. For bidirectional prediction, decoder-side motion vector refinement (DMVR) [8], bidirectional optical flow (BDOF) [9,10], and affine motion compensation (AMC) [11–13] are employed at the decoding end to optimize the precision of the prediction, thereby enhancing its overall accuracy.

Affine motion estimation is a crucial step in video coding, serving to characterize inter-frame motion and facilitate differential frame encoding. Compared to previous motion estimation algorithms, affine motion models are more suitable for processing high-definition video content due to their ability to handle complex video scenes that involve the translation, rotation, and scaling of objects. However, traditional affine motion estimation algorithms suffer from high computational complexity and insufficient accuracy, which limits the efficiency and quality of video coding. VVC provides two types of affine motion estimation models in the affine motion estimation module, namely the four-parameter

affine model and the six-parameter affine model. In the affine motion estimation module, the structure of the current coding unit (CU) is 4 × 4 sub-blocks, and the motion vector of each sub-block can be obtained from the control point motion vector (CPMV) of the two affine motion models. Figure 1 shows these three transformations of the image. The three images (a–c) below describe the rotation, translation, and scaling of an image, where α denotes the rotation angle of the image, $(\Delta x, \Delta y)$ denotes the translation of the image, and s denotes the scaling.

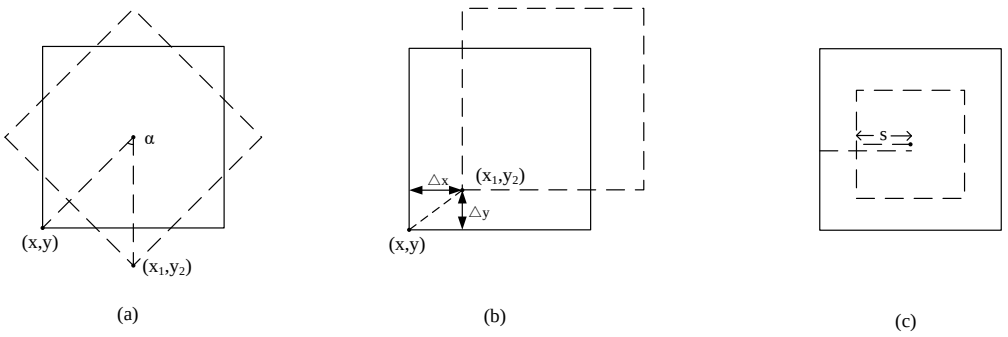


Figure 1. Three transformations of the image: (**a**) rotation, (**b**) translation, (**c**) zooming.

The affine motion model depicts the motion of an object or image in two-dimensional space under transformations such as translation, rotation, scaling, and misalignment. Affine transformations can be expressed as a combination of linear transformations and translations while preserving their affine properties. Through affine transformation, the image can be geometrically corrected, scaled, and aligned to improve the accuracy and stability of image processing and analysis. The affine transformation model of an image can be obtained through dimensionality reduction, utilizing the translation properties of Fourier transform and estimation theory for log-polar transformed images. Assuming that image $f_{p2}(x, y)$ is the result of a translation of the image $f_{p1}(x, y)$ by $(\Delta x, \Delta y)$, the relationship between the two images is shown in Formula (1), with the Fourier transform relationship as Formula (2) and the energy spectrum (amplitude spectrum) represented by Formula (3).

$$f_{p2}(x, y) = f_{p1}(x - \Delta x, y - \Delta y) \tag{1}$$

$$F_2(\xi, \eta) = e^{-j2\pi(\xi\Delta x + \eta\Delta y)} F_1(\xi, \eta) \tag{2}$$

$$M_2(\xi, \eta) = M_1(\xi, \eta) \tag{3}$$

where M_2 and M_1 are the energy spectra of F_2 and F_1. Firstly, through spectral transformation, we convert the geometric transformations (rotation and scaling) of the image into frequency domain translations. This allows for simpler and more efficient operations in the frequency domain, where image rotation and scaling can be achieved by straightforwardly shifting frequency components, reducing the complexity of geometric transformation calculations. The statement implies that the amplitude spectrum of the image translation remains unchanged in the frequency domain, which can be utilized for dimensionality reduction estimation of image transformation. Assuming that the positional relationship of the image satisfies Formula (4), Formula (5) is the relationship expression between two images.

$$(x_1, y_1)' = sR(\alpha)(x, y)' + T \tag{4}$$

$$\begin{cases} x_1 = sx\cos\alpha + sy\sin\alpha + \Delta x \\ y_1 = -sx\sin\alpha + sy\cos\alpha + \Delta y \end{cases} \tag{5}$$

where $R(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$, $T = (\Delta x, \Delta y)'$, (x_1, y_1) represents the coordinates in the transformed image, (x, y) represents the coordinates in the original image, and s is the zooming factor, describing how the point (x, y) is stretched or shrunk in the horizontal and vertical directions after the rotation. If s is greater than 1, it indicates an enlargement; if s is between 0 and 1, it represents a reduction; if s is a negative value, it might involve a reflection. α is the rotation angle, describing the degree to which the point (x, y) is rotated around the origin. If α is a positive value, it indicates a counterclockwise rotation; if α is a negative value, it represents a clockwise rotation. The unit of angle is typically in radians, and $(\Delta x, \Delta y)$ are displacements in x and y directions. $mv^h_{(x,y)}$ and $mv^v_{(x,y)}$ are horizontal and vertical MVs determined by a, b, and $(\Delta x, \Delta y)$. Replacing $s\cos\alpha$ and $s\sin\alpha$ with a and b can obtain a concise form of MVs.

$$\begin{cases} mv^h_{(x,y)} = x_1 - x = ax + by + \Delta x \\ mv^v_{(x,y)} = y_1 - y = -bx + ay + \Delta y \end{cases} \tag{6}$$

The nonlinear mapping of the image is realized by bilinear interpolation. In this way, the rotation and scaling transformation of the image are reduced to the translation transformation, and the four-parameter motion model can be obtained.

The motion vectors of the sub-blocks based on the current coding block can be obtained by an affine motion model with four and six parameters of two control point motion vectors CPMV1 and CPMV2, or three control point motion vectors CPMV1, CPMV2, and CPMV3, as shown in Figure 2, where the x-coordinate and y-coordinate denote the horizontal and vertical components of the motion vector, (a) is a four-parameter affine motion model and (b) is a six-parameter affine motion model, and CPMV1, CPMV2, and CPMV3 are the motion vectors of the upper left corner control point, upper right corner motion vector, and lower left corner control point motion vectors of the current coding unit. The motion vectors of the sub-blocks centered on (x, y) in Figure 2 can be calculated by the following equation:

$$\begin{cases} mv^h_{(x,y)} = \frac{mv^h_2 - mv^h_1}{W-1}x - \frac{mv^v_2 - mv^v_1}{W-1}y + mv^h_1 \\ mv^v_{(x,y)} = \frac{mv^v_2 - mv^v_1}{W-1}x + \frac{mv^h_2 - mv^h_1}{W-1}y + mv^v_1 \end{cases} \tag{7}$$

where $mv^h_{(x,y)}$ and $mv^v_{(x,y)}$ represent the motion vectors in the horizontal and vertical directions of the current encoding sub-blocks (x, y), and W is the width of the current encoding unit; thus, $(W-1)$ is the distance between Control-point 1 and Control-point 2, and mv^h_2, mv^h_1, mv^v_2, and mv^v_1 are the MV components of Control-point 1 and Control-point 2. The results of the four parameters (a, b, T) are delivered only by mv_1 and mv_2. Through this method, the current encoding unit can be obtained from the motion vectors of 4×4 sub-blocks.

Formula (7) can also be rewritten in the form of a matrix:

$$mv(x, y) = A(x, y)MV^T_A \tag{8}$$

$$A(x,y) = \begin{bmatrix} (1-\frac{x}{W-1}) & \frac{x}{W-1} & \frac{y}{W-1} & -\frac{y}{W-1} \\ -\frac{y}{W-1} & \frac{y}{W-1} & (1-\frac{x}{W-1}) & \frac{x}{W-1} \end{bmatrix} \tag{9}$$

$$MV_A = \begin{bmatrix} mv_1^h & mv_2^h & mv_1^v & mv_2^v \end{bmatrix} \tag{10}$$

The six-parameter affine motion model has one more CPMV than the four-parameter affine model. The motion vectors of the current coding block of the six-parameter affine motion model are derived in a similar way to that of the four-parameter affine motion model. The formula for calculating the motion vector centered on the current block (x, y) is as follows:

$$\begin{cases} mv_{(x,y)}^h = \frac{mv_2^h - mv_1^h}{W-1}x + \frac{mv_3^v - mv_1^v}{H-1}y + mv_1^h \\ mv_{(x,y)}^v = \frac{mv_2^v - mv_1^v}{W-1}x + \frac{mv_3^h - mv_1^h}{H-1}y + mv_1^v \end{cases} \tag{11}$$

where mv_3^v and mv_3^h represent the motion vectors in the horizontal and vertical directions motion vectors, H is the height of the block, and $(H-1)$ is the distance between Control-point 1 and Control-point 3. This formula can also be rewritten in matrix form. The mv_{1x}, mv_{2x}, mv_{3x} in Figure 2 represent mv_1^h, mv_2^h, mv_3^h, and mv_{1y}, mv_{2y}, mv_{3y} in Figure 2 represent mv_1^v, mv_2^v, mv_3^v.

$$mv(x,y) = A(x,y)MV_A^T \tag{12}$$

$$A(x,y) = \begin{bmatrix} 1-\frac{x}{W-1}-\frac{y}{H-1} & \frac{x}{W-1} & \frac{y}{H-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-\frac{x}{W-1}-\frac{y}{H-1} & \frac{x}{W-1} & \frac{y}{H-1} \end{bmatrix} \tag{13}$$

$$MV_A = \begin{bmatrix} mv_1^h & mv_2^h & mv_3^h & mv_1^v & mv_2^v & mv_3^v \end{bmatrix} \tag{14}$$

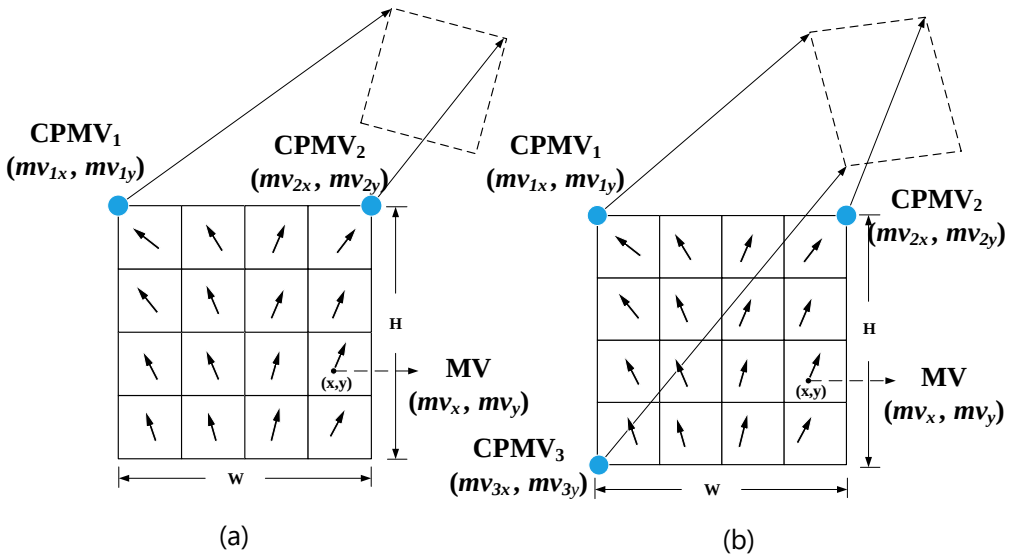


Figure 2. Affine motion model: (**a**) four-parameter affine model, (**b**) six-parameter affine model.

Introducing an affine motion estimation model in VVC can effectively describe complex video content and improve the performance of the encoder. However, the computational complexity of AME is high, and the motion estimation algorithm in the inter-prediction module takes up a long encoding time [14]. To achieve better encoding gain while reducing its computational complexity, many researchers have attempted to reduce its computational complexity in traditional affine motion estimation modules [15–21]. However, there is still little work in the inter-frame prediction affine motion estimation module of VVC and there is still a large optimization space.

This paper proposes a fast gradient iterative affine motion estimation algorithm based on edge detection that can effectively accelerate the affine motion estimation process of inter-frame prediction and achieve the goal of shortening the overall encoding time of the encoder. This method consists of two steps. The first process is to set judging conditions in the candidate set of affine motion vectors and achieve the goal of skipping redundant judging when the candidate set meets the conditions. The second process is to use the Canny edge detection operator in the affine motion estimation to obtain the gradient, thereby accelerating the gradient iteration. While ensuring the coding performance and image quality, it accelerates the time for the affine motion estimation part of inter-frame prediction and reduces the overall encoding time of the encoder.

The arrangement of the entire article is as follows: Section 2 reviews the relevant research achievements and progress, Section 3 provides a detailed introduction to the fast affine motion estimation algorithm proposed in this paper, Section 4 provides experimental analysis and results, and Section 5 provides the conclusion of this paper.

2. Related Work

Modern multimedia applications have increasingly high requirements for video encoders, requiring both high coding efficiency and low computational complexity to ensure low latency and high transmission speed in real-time applications. To meet this demand, researchers are committed to designing efficient and low-complexity video encoders. Although there has been some research work on reducing the computational complexity of VVC encoders, most of the research has focused on accelerating early decision making in the partitioning process [15–21]. Reference [15] proposed a fast partitioning algorithm for intra and inter-frame encoding. For intra-frame encoding, the Canny edge detection algorithm is used to extract the features of image encoding and the features are used to determine whether to skip vertical or horizontal partitioning, achieving the goal of early termination. For inter-frame encoding, the three-frame difference method is used to determine whether an object is a moving target. Reference [16] proposes a fast texture-based CU partitioning method that evaluates the complexity of the current CU to determine whether to skip subsequent partitioning. At the same time, an improved Canny operator is used to extract edge information to exclude horizontal or vertical partitioning patterns. Reference [17] analyzes the probability of affine motion estimation mode in bidirectional prediction, explores the mutual exclusion between skip mode and affine mode, and proposes a VVC fast affine motion estimation mode based on near coding information. Reference [18] studied a fast motion estimation algorithm for the early termination of partial blocks in the CU, using skip mode for the CU that does not require affine changes. In reference [19], Zhao et al. extracted the standard deviation and edge ratio to accelerate the division of the CU. The CU split information and the time position of the encoded frame are used for low-complexity encoders [20]. Reference [21] checks whether the optimal prediction mode of the current encoding block is skip mode. If it is, it skips the entire affine motion estimation process and checks the direction of the optimal prediction. The detection results determine whether to reduce the size of the reference sequence, thereby reducing computational complexity. Reference [22] proposes an adaptive affine four-parameter and six-parameter encoding architecture where the encoder can adaptively select between two affine motion models. Reference [23] proposes an affine motion estimation model that iteratively searches for affine motion vectors and a method for constructing an affine advanced motion vector prediction candidate (AAMVP) list, which has been adopted by the H.266/VVC standard. Reference [14] proposes an affine motion compensation based on feature matching that can further improve the efficiency of video coding. Reference [24] carries out affine motion estimation through block division and predicts each pixel using a reference coordinate system to achieve the purpose of predicting affine transformation. Reference [25] proposes an affine motion estimation scheme that does not require additional alternating segmentation and estimation, described by applying a segmented function of the parameter field, and derives a specific splitting optimization scheme at close range. Reference [26] proposes a method of

using rate-distortion theory and displacement estimation error to determine the minimum bit rate required for the information transmission of prediction error in the coding process. Reference [27] proposes a method for solving the problem of relative pose estimation by using the affine transformation between feature points. Reference [28] proposes a motion compensation scheme for three-zone segmentation. Based on segmentation information, three motion compensation regions are divided, namely the edge region, foreground region, and background region. By using the information from these three regions, the accuracy and encoding efficiency of motion compensation are improved. Reference [29] proposes a method of edge video compression texture synthesis based on a generative adversarial network to obtain the most authentic texture information. Reference [30] proposes an affine parameter model that utilizes matching algorithms to discover and extract feature point pairs from edges within consecutive frames and selects the optimal set of three sets of point pairs to describe global motion. Reference [31] proposes linear applications of traditional intra-prediction modes based on a pattern correlation processing sequence, region-based template matching prediction methods, and neural-network-based intra-prediction modes. Reference [32] proposes a context-based inter-mode judging method that skips affine modes by determining whether radial motion estimation is performed during the rate-distortion optimization process of the optimal CU mode decision. Reference [33] adds momentum parameters to accelerate the iterative process based on the symmetry of the affine motion estimation iterative process.

Overall, most of the current research work is focused on the skip judging of the affine motion estimation module. However, there have not been many improvements and optimizations to the architecture of the affine motion itself. In H.266/VVC, the affine motion estimation algorithm obtains the optimal radiative motion vector through gradient iteration. However, in the current research, the gradient iteration method adopts the traditional traversal algorithm to obtain gradient information through traversing images. This approach is not suitable for scenarios where there is a large amount of affine motion in high-definition video, resulting in the high computational complexity and complexity of the affine motion estimation module itself not being well addressed. This is what current research work needs to achieve.

3. Materials and Methods

Affine motion estimation is located in the inter-prediction module of the H.266/VVC encoder, which uses a method similar to the inter-prediction motion estimation in H.265/HEVC to search for motion vectors. The affine motion estimation of the VVC encoder first uses affine advanced motion vector prediction technology (AAMVP) to obtain the starting candidate list of affine motion vectors for the current encoding block. Then, a set of optimal candidate running vectors is selected as the starting search points in the list, and the optimal motion vector combination for the current encoding block is determined through iterative search.

3.1. Affine Advanced Motion Vector Prediction

The AAMVP technology is used in the inter-frame prediction of VVC to construct a candidate list of starting vector groups while using judging conditions to select the optimal set of vector combinations as the starting position for the iterative search. In VVC, the candidate length of AAMVP is defined as two, and the candidate list is established for each predicted image. Each list only contains unidirectional motion information. Figure 3 is the AAMVP candidate list build diagram. The encoder first checks the inheritance of available information adjacent to the current encoding unit in the order of bottom left, bottom, top right, and top left. If affine motion estimation mode is used in adjacent blocks, the affine information of adjacent encoded blocks is directly inherited. If the candidate set of the previous operation is not filled, adjacent blocks at the motion vectors of the three control points of the current encoding block are checked separately, and the motion vectors are combined using the first nonaffine motion mode translation motion vector at each CPMV.

If it is still not satisfied, the time-domain translation motion vector and zero-value MV are combined to fill the candidate list. It can be seen that the candidate list of AAMVP adopts five steps to construct motion vector combinations, namely: spatial adjacent affine mode CU inheritance; translation construction of adjacent CU in airspace; translated MV filling of adjacent CU in airspace; time domain translation MV filling; zero-value MV padding.

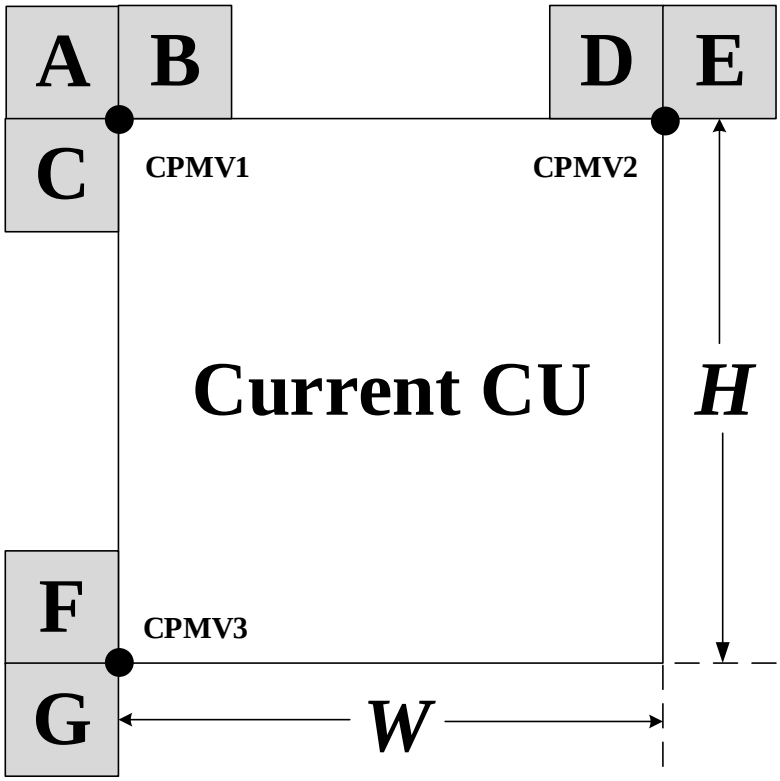


Figure 3. Affine advanced motion vector prediction list.

In the process of constructing the candidate list for AAMVP, the adjacent blocks of adjacent affine modes in the spatial domain need to meet three conditions: first, they must be in inter-frame encoding mode, then in affine encoding mode, and, finally, the reference image must be the same as the current CU reference image. The translation construction of the adjacent CU in the same spatial domain also needs to meet three conditions: first, the inter-frame encoding mode, then the nonaffine encoding mode, and, finally, the reference image must be the same as the reference image of the current CU. We define the first condition as *Condition*_1. The second condition is *Condition*_2. When the conditions are not met, the judging of the current neighboring block is skipped in advance, and there is no need to perform other complex condition calculations and judging. At the same time, in the process of constructing the candidate set, after each step, a judging is made on whether the candidate set is filled. If the candidate set has already been filled in the current step, the subsequent judging steps are skipped, which can achieve the early termination of the candidate list construction process and reduce the computational burden of the encoder in this step. Figure 4 below is the optimization flowchart of AMMVP.

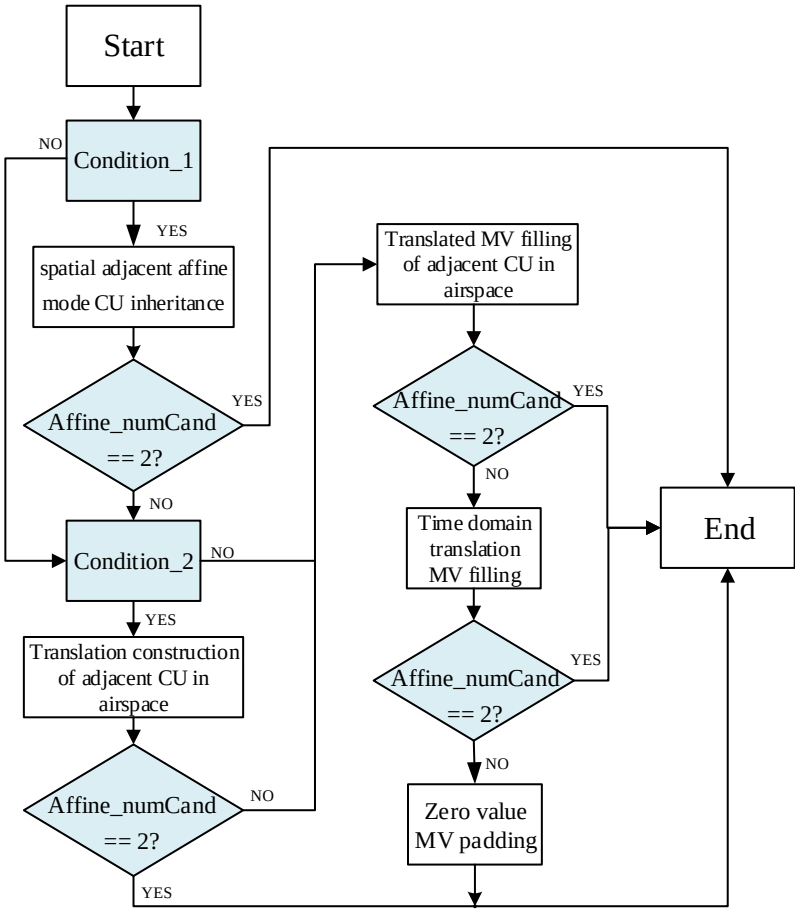


Figure 4. Affine advanced motion vector prediction condition.

3.2. The Iterative Search of Affine Motion Vectors

In the VVC standard, the encoder uses the AAMVP technique from the previous step to obtain the optimal affine motion vector combination as the starting search motion vector group and obtains the optimal affine motion vector combination for the current encoding block through iterative search. Fast affine motion estimation needs to calculate a set of optimal affine motion vectors, usually two or three, so the VVC encoder uses mean squared error (MSE) as the matching criterion. The formula definition for MSE is as follows:

$$MSE = \frac{1}{w \times h} \sum_{(x,y) \in Cur} \left| P_{cur}(x,y) - P_{ref}((x,y) + mv(x,y)) \right|^2 \tag{15}$$

where w and h are the width and height of the current encoding block, $P_{cur}(x,y)$ is the image where the current encoding block is located, and $P_{ref}(x,y)$ is the reference image for the current encoding block.

Define the change in motion vector after the ith iteration as d^i_{MV}. The expression of the motion vector at the ith iteration can be defined as follows:

$$mv^i_{(x,y)} = A(x,y)((mv^{i-1}_{(x,y)})^T + (d^i_{MV})^T) = mv^{i-1}_{(x,y)} + A(x,y)(d^i_{MV})^T \tag{16}$$

where (x,y) represents the position of the current encoding block, and the change in the motion vector d^i_{MV} is a row matrix. Now, its transpose is given as follows:

$$(d^i_{MV})^T = \begin{pmatrix} d^h_{MV1} \\ d^h_{MV2} \\ d^v_{MV1} \\ d^v_{MV2} \end{pmatrix} = \begin{pmatrix} mv^{ih}_1 - mv^{(i-1)h}_1 \\ mv^{ih}_2 - mv^{(i-1)h}_2 \\ mv^{iv}_1 - mv^{(i-1)v}_1 \\ mv^{iv}_2 - mv^{(i-1)v}_2 \end{pmatrix} \tag{17}$$

After i iterations, the pixel values of the current reference point can be obtained as follows:

$$P_{ref}((x,y) + mv_{(x,y)}) = P_{ref}((x_{i-1}, y_{i-1}) + A(x,y)(d^i_{MV})^T) \tag{18}$$

where (x_{i-1}, y_{i-1}) is the position of the matching block during the previous iteration search, and Taylor polynomial expansion is performed on Formula (13) while ignoring higher-order polynomials to obtain Formula (14):

$$P_{\text{ref}}((x_{i-1}, y_{i-1}) + A(x,y)(d^i_{\text{MV}})^T) \approx P_{\text{ref}}(x_{i-1}, y_{i-1}) + P'_{\text{ref}}(x_{i-1}, y_{i-1})A(x,y)(d^i_{\text{MV}})^T \tag{19}$$

To minimize the value of MSE during the iteration process, the pixel value $P_{ref}(x,y)$ of the reference point needs to be as close as possible to the original pixel value $P_{cur}(x,y)$. Set the relative gradient of the relative change d^i_{MV} of the motion vector to zero. If the value is zero during the iteration process, it indicates that the current reference pixel value is closest to the original pixel value and is the best-matched result. In the encoder model of VVC, the Sobel operator is used to convolute the pixel matrix to obtain the gradient. The formula is as follows:

$$\begin{cases} g_h = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} P_{i-1,j-1} & P_{i-1,j} & P_{i-1,j+1} \\ P_{i,j-1} & P_{i,j} & P_{i,j+1} \\ P_{i+1,j-1} & P_{i+1,j} & P_{i+1,j+1} \end{pmatrix} \\ g_v = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} P_{i-1,j-1} & P_{i-1,j} & P_{i-1,j+1} \\ P_{i,j-1} & P_{i,j} & P_{i,j+1} \\ P_{i+1,j-1} & P_{i+1,j} & P_{i+1,j+1} \end{pmatrix} \end{cases} \tag{20}$$

Considering the complexity of high-definition video content, using the Sobel operator to traverse images to obtain gradients greatly increases computational complexity, and the encoder has a high time consumption. This article uses the Canny edge detection algorithm to optimize the operation of traversing images. In affine motion models, the Canny edge detection algorithm is more efficient than simply using the Sobel algorithm to obtain the gradient changes of motion vector groups. However, the computational complexity of the Canny edge detection algorithm is higher. Considering that the affine motion estimation model is only used in the inter-prediction module of the VVC encoder, the affine motion model is only used when it meets specific conditions. Therefore, when the encoder chooses to use the affine motion mode, the Canny edge detection algorithm is activated. At the same time, when performing affine motion estimation, the Canny algorithm performs global gradient calculation during the processing of the first frame image and then makes corresponding marks. When processing subsequent images, it detects whether the current encoding block has already been calculated as a gradient in the first frame image. If it has already been calculated, the gradient value is directly read from the cache without the need for global calculations. The purpose of this operation is to skip the image area with an unchanged background and focus the image processing on the changing area, which is the calculation of motion vectors. Algorithm 1 is the C++ Pseudocode proposed for gradient calculation. The specific steps of Algorithm 1 are as follows:

- Obtain the original data of the image, initialize the variable, cache, and mark the image with calculated edges.

- By traversing each pixel of the image, calculate the gradient and error at each pixel position. If isCannyComputed is false, it indicates that the Canny edge image needs to be recalculated for the first time; otherwise, skip.
- Traverse the image and repeat the calculation.

Algorithm 1: xAffineMotionEstimation.

Input : origBuf: Original buffer of pixel values;
predBuf: Predicted buffer of pixel values;
bufStride: Stride of the original buffer;
predBufStride; Stride of the predicted buffer;
Output: piError : Array of prediction errors;
pdDerivateX(Y): Array of horizontal(vertical) derivatives;

```
Initialize variables:;
pOrg = origBuf.Y().buf; pPred = predBuf.Y().buf; isCannyComputed = false;
for j from 0 to height − 1 do
    for i from 0 to width − 1 do
        if i > 0 and i < width − 1 and j > 0 and j < height − 1 then
            if isCannyComputed is false then
                Compute Canny edge image (predMat, cannyEdges);
                Set isCannyComputed to true;
            end
            Calculate horizontal gradient using Canny edges;
            Calculate vertical gradient using Canny edges;
        end
        Set pdDerivateX[i + j × width] to dx;
        Set pdDerivateY[i + j × width] to dy;
        Calculate prediction error: piError[i + j × width] = pOrg[i] − pPred[i];
    end
    Update pointers: pOrg = pOrg + bufStride, pPred = pPred + predBufStride;
end
if isCannyComputed is false then
    Compute gradient using Canny edges;
    Update motion vectors using gradient;
    Set isCannyComputed to true;
end
```

4. Experiments and Results Analysis

4.1. Simulation Setup

In order to analyze and evaluate the performance of the algorithm proposed in this paper, the official testing software VTM10.0 of H.266/VVC was used as the anchor for testing, and the algorithm proposed in this article was implemented using JVET Common Test Condition (CTC) [34] configuration. The compiling environment is VS 2019, and Microsoft Windows 10 64-bit Bitwise operation operating system was adopted. The configuration file in the experiment uses low latency P-frames, and the quantization parameters (QP) are 22, 27, 32, 37. Table 1 shows the experimental environment parameters.

Table 1. The environments and conditions of simulation.

Items	Descriptions
Software	VTM-10.0
Configuration file	encoder_lowdelay_P_vtm.cfg
Number of frames to be coded	30
Quantization parameter	22, 27, 32, 37
Search range	64
CU size/depth	64/4
Sampling of luminance to chrominance	4:2:0

4.2. Performance and Analysis

At the same time, the evaluation index uses the Bjøntegard delta bitrate (BDBR) [35] to measure the encoding performance of the proposed algorithm and the original algorithm on bitrate. A negative number of data indicates that the proposed algorithm can save the corresponding data volume, while a negative number indicates an increase in data volume and poor performance. In addition, the Bjøntegard delta peak signal-to-noise rate (BD-PSNR) was used to evaluate the performance index of the proposed algorithm and the original algorithm in encoding image quality. A positive value represents an enhancement of the processed image quality, while a negative value indicates significant distortion and poor performance compared to the original image. Table 2 provides parameter information for the test sequence.

Table 2. Detailed characteristics of the experimental video sequences.

Sequences	Size	Bit-Depth	Frame Rate
BasketballDrive	1920 × 1080	8	50
Cactus	1920 × 1080	10	50
FourPeople	1280 × 720	8	60
KristenAndSara	1280 × 720	8	60
BasketballDrill	832 × 480	8	50
PartyScene	832 × 480	8	50
RaceHorses	416 × 240	8	30
BQSquare	416 × 240	8	60
BasketballPass	416 × 240	8	50

Firstly, VTM10.0 and the proposed algorithm are compared. In order to compare the impact of the proposed algorithm and the algorithm in VTM10.0 on encoder encoding time, a formula is defined to calculate the average time of each algorithm:

$$EncT_{all,aff} = \frac{1}{4} \sum_{QP_i \in \{37,32,27,22\}} \left(\frac{T_{org(QP_i)} - T_{pro(QP_i)}}{T_{org(QP_i)}} \times 100\% \right) \quad (21)$$

where $EncT_{all,aff}$ represents the overall encoding time of the encoder or the affine motion estimation encoding time. Due to the corresponding changes in the algorithm testing time based on changes in QP, the method of taking the average value is adopted for evaluation and measurement. Table 3 shows the experimental results of the algorithm proposed in this article. In the improved algorithm, the overall encoding time of the encoder was saved by 6.22%, and the encoding time in the affine motion estimation module was reduced by an average of 24.79%. Among them, the reduction in encoding time for sequences BasketballPass, BQSquare, and KristenAndSara affine motion estimation was all over 30%, indicating that the improved algorithm has a good optimization effect on processing video sequences with a large amount of affine motion, effectively reducing the encoding time of the encoder. On the contrary, the bitrate of the Basketball Drive video sequence increased too significantly, indicating a large amount of data when processing certain high-definition videos.

Table 3. The proposed method compared to the original VVC experimental results.

Sequences	BDBR/%	BD-PSNR/dB	EncTall/%	EncTaff/%
BasketballPass	0.34	−0.063	11.12	31.80
BQSquare	0.84	−0.115	9.05	32.11
RaceHorses	0.83	−0.037	8.23	23.59
PartyScene	0.93	−0.040	6.92	18.96
BasketballDrill	0.50	−0.019	3.27	15.39
KristenAndSara	1.06	−0.028	4.81	32.98
FourPeople	0.39	−0.018	4.33	27.13
Cactus	0.51	−0.012	4.36	20.47
BasketballDrive	1.50	−0.030	3.89	20.66
Average	0.76	−0.040	6.22	24.79

In order to further compare the performance of the algorithm proposed in this article, we will conduct a comparative analysis between the algorithm proposed in this article and the current research methods for related work. As shown in Table 4, compared with Ren et al. [33], the algorithm proposed in this paper has a better effect in reducing the overall encoding time of VVC encoders, with little loss in BDPSNR and little increase in bitrate. The method proposed by Ren et al. [33]. effectively reduces the computational complexity of affine motion estimation, but when the CU adopts affine mode, the process of affine motion estimation still needs to be executed, although momentum parameters are added to accelerate the iteration process.

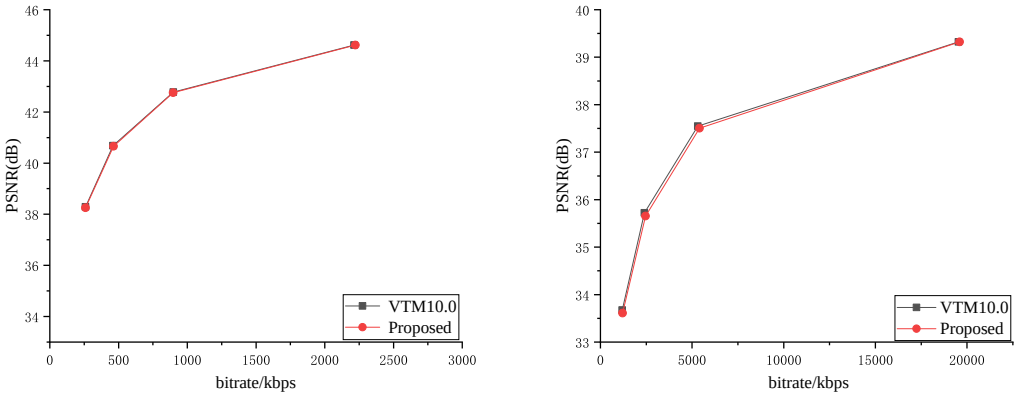
Table 4. The proposed method compared to the state-of-the-art experimental results.

Sequence Name	Ren et al. [33]		Proposed	
	BDBR/%	SavTall/%	BDBR/%	SavTall/%
BasketballDrive	0.08	5.00	1.50	3.89
Cactus	0.11	6.00	0.51	4.36
BasketballDrill	0.06	3.00	0.50	3.27
PartyScene	0.26	4.00	0.93	6.92
RaceHorses	0.08	5.00	0.83	8.23
BasketballPass	0.08	2.00	0.34	11.12
Average	0.11	4.16	0.77	6.30

At the same time, Figure 5 shows the reconstructed and original frames of the video sequence processed by the algorithm in this paper. From the subjective naked eye observation, the distortion of the image is almost invisible. In summary, the algorithm proposed in this article can shorten the encoding time while ensuring video quality and bitrate. In order to more intuitively observe the compression and distortion degree of the proposed algorithm on video images, Figure 6 shows the RD curves of the test sequences KristenAndSara and Cactus. From the RD curve, it can be seen that the method proposed in this paper coincides with the algorithm curve in the official testing sequence VTM10.0 of VVC. This means that the algorithm proposed in this article greatly shortens the encoding time required by the encoder while maintaining an almost constant video quality and bit rate.

(**a**) (**b**)

Figure 5. Video sequence: (**a**) original frame; (**b**) reconstructed frame.



(**a**) (**b**)

Figure 6. RD curves: (**a**) KristenAndSara RD curve; (**b**) Cactus RD curve.

5. Conclusions

Due to the complexity of high-definition video content and the addition of multiple optimization algorithms to the encoder, the processing of affine motion estimation in VVC inter-frame prediction needs to be optimized. To address the above issues, this paper proposes a fast affine motion algorithm based on edge detection to accelerate the encoder's processing of affine motion patterns. By adding pre-judging conditions in the process of constructing affine candidate lists, unnecessary judging steps are skipped to achieve acceleration. In the iterative search process of affine motion vectors, the Canny edge detection algorithm is used to accelerate the iterative gradient, making full use of the readily calculated image gradient, avoiding repeated calculations, and making the calculation process more efficient. The experimental results show that BDBR only increases by 0.76% and BDPSNR only loses 0.04 db. Compared with the anchor algorithm, the overall coding time of the proposed algorithm is reduced by 6.22%, and the coding time of the affine motion estimation part of the inter-frame prediction part is reduced by 24.79%. In future research work, we will focus on the overall model of affine motion estimation, which can be combined with hardware to achieve the goal of saving more encoding time.

Author Contributions: Z.D. conceived the idea and J.H. conducted the analyses and writing. X.Z., N.S. and P.C. contributed to the writing and revisions. P.C. provided funding support. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the General Project of National Natural Science Foundation of China under Grant 61972042 and in part by the Discipline Construction Project of Beijing Institute of Graphic Communication under Grant 21090123009.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no known competing financial interest or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Bross, B.; Chen, J.; Liu, S. Versatile Video Coding (Draft 1). In Proceedings of the 10th JVET Meeting, San Diego, CA, USA, 10–20 April 2018.
2. Bross, B.; Chen, J.; Ohm, J.R.; Sullivan, G.J.; Wang, Y.K. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). *Proc. IEEE* **2021**, *109*, 1463–1493. [CrossRef]
3. Hamidouche, W.; Biatek, T.; Abdoli, M.; François, E.; Pescador, F.; Radosavljević, M.; Menard, D.; Raulet, M. Versatile video coding standard: A review from coding tools to consumers deployment. *IEEE Consum. Electron. Mag.* **2022**, *11*, 10–24. [CrossRef]
4. Sidaty, N.; Hamidouche, W.; Déforges, O.; Philippe, P.; Fournier, J. Compression performance of the versatile video coding: HD and UHD visual quality monitoring. In Proceedings of the 2019 Picture Coding Symposium (PCS), Ningbo, China, 12–15 November 2019; pp. 1–5.
5. Li, X.; Chuang, H.; Chen, J.; Karczewicz, M.; Zhang, L.; Zhao, X.; Said, A. Multi-type-tree, document JVET-D0117. In Proceedings of the 4th JVET Meeting, Chengdu, China, 17–21 October 2016.
6. Schwarz, H.; Nguyen, T.; Marpe, D.; Wiegand, T. Hybrid video coding with trellis-coded quantization. In Proceedings of the 2019 Data Compression Conference (DCC), Snowbird, UT, USA, 26–29 March 2019; pp. 182–191.
7. Zhao, X.; Chen, J.; Karczewicz, M.; Said, A.; Seregin, V. Joint separable and non-separable transforms for next-generation video coding. *IEEE Trans. Image Process.* **2018**, *27*, 2514–2525. [CrossRef] [PubMed]
8. Sethuraman, S. CE9: Results of dmvr related tests CE9. 2.1 and CE9. 2.2. *Jt. Video Expert. Team (JVET) ITU-T SG* **2019**, *16*, 9–18.
9. Xiu, X.; He, Y.; Ye, Y.; Luo, J. Complexity Reduction and Bit-Width Control for Bi-Directional Optical Flow. U.S. Patent 11,470,308, 11 October 2022.
10. Kato, Y.; Toma, T.; Abe, K. Simplification of BDOF, document JVET-O0304. In Proceedings of the 15th JVET Meeting, Gothenburg, Sweden, 1–9 October 2019; pp. 3–12.
11. Lin, S.; Chen, H.; Zhang, H.; Maxim, S.; Yang, H.; Zhou, J. Affine transform prediction for next generation video coding, document COM16-C1016. In Proceedings of the Huawei Technologies, International Organisation for Standardisation Organisation Internationale De Normalisation ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, ISO/IEC JTC1/SC29/WG11 MPEG2015/m37525, Geneva, Switzerland, 1 January 2017.
12. Chen, J.; Karczewicz, M.; Huang, Y.W.; Choi, K.; Ohm, J.R.; Sullivan, G.J. The joint exploration model (JEM) for video compression with capability beyond HEVC. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 1208–1225. [CrossRef]
13. Zhao, X.; Seregin, V.; Said, A.; Zhang, K.; Egilmez, H.E.; Karczewicz, M. Low-complexity intra prediction refinements for video coding. In Proceedings of the 2018 Picture Coding Symposium (PCS), San Francisco, CA, USA, 24–27 June 2018; pp. 139–143.
14. Zhang, X.; Ma, S.; Wang, S.; Zhang, X.; Sun, H.; Gao, W. A joint compression scheme of video feature descriptors and visual content. *IEEE Trans. Image Process.* **2016**, *26*, 633–647. [CrossRef]
15. Tang, N.; Cao, J.; Liang, F.; Wang, J.; Liu, H.; Wang, X.; Du, X. Fast CTU partition decision algorithm for VVC intra and inter coding. In Proceedings of the 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Bangkok, Thailand, 11–14 November 2019; pp. 361–364.
16. Zhang, Q.; Zhao, Y.; Jiang, B.; Huang, L.; Wei, T. Fast CU partition decision method based on texture characteristics for H. 266/VVC. *IEEE Access* **2020**, *8*, 203516–203524. [CrossRef]
17. Li, X.; He, J.; Li, Q.; Chen, X. An Adjacency Encoding Information-Based Fast Affine Motion Estimation Method for Versatile Video Coding. *Electronics* **2022**, *11*, 3429. [CrossRef]
18. Guan, X.; Sun, X. VVC fast ME algorithm based on spatial texture features and time correlation. In Proceedings of the 2021 International Conference on Digital Society and Intelligent Systems (DSInS), Chengdu, China, 3–4 December 2021; pp. 371–377.
19. Zhao, J.; Wu, A.; Zhang, Q. SVM-based fast CU partition decision algorithm for VVC intra coding. *Electronics* **2022**, *11*, 2147. [CrossRef]
20. Khan, S.N.; Muhammad, N.; Farwa, S.; Saba, T.; Khattak, S.; Mahmood, Z. Early Cu depth decision and reference picture selection for low complexity Mv-Hevc. *Symmetry* **2019**, *11*, 454. [CrossRef]
21. Park, S.H.; Kang, J.W. Fast affine motion estimation for versatile video coding (VVC) encoding. *IEEE Access* **2019**, *7*, 158075–158084. [CrossRef]
22. Zhang, K.; Chen, Y.W.; Zhang, L.; Chien, W.J.; Karczewicz, M. An improved framework of affine motion compensation in video coding. *IEEE Trans. Image Process.* **2018**, *28*, 1456–1469. [CrossRef] [PubMed]
23. Li, L.; Li, H.; Liu, D.; Li, Z.; Yang, H.; Lin, S.; Chen, H.; Wu, F. An efficient four-parameter affine motion model for video coding. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 1934–1948. [CrossRef]
24. Kordasiewicz, R.C.; Gallant, M.D.; Shirani, S. Affine motion prediction based on translational motion vectors. *IEEE Trans. Circuits Syst. Video Technol.* **2007**, *17*, 1388–1394. [CrossRef]
25. Fortun, D.; Storath, M.; Rickert, D.; Weinmann, A.; Unser, M. Fast piecewise-affine motion estimation without segmentation. *IEEE Trans. Image Process.* **2018**, *27*, 5612–5624. [CrossRef]
26. Meuel, H.; Ostermann, J. Analysis of affine motion-compensated prediction in video coding. *IEEE Trans. Image Process.* **2020**, *29*, 7359–7374. [CrossRef]
27. Guan, B.; Zhao, J.; Li, Z.; Sun, F.; Fraundorfer, F. Relative pose estimation with a single affine correspondence. *IEEE Trans. Cybern.* **2021**, *52*, 10111–10122. [CrossRef]

28. Wang, Z.; Wang, S.; Zhang, X.; Wang, S.; Ma, S. Three-zone segmentation-based motion compensation for video compression. *IEEE Trans. Image Process.* **2019**, *28*, 5091–5104. [CrossRef]
29. Zhu, C.; Xu, J.; Feng, D.; Xie, R.; Song, L. Edge-based video compression texture synthesis using generative adversarial network. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 7061–7076. [CrossRef]
30. Huang, J.C.; Hsieh, W.S. Automatic feature-based global motion estimation in video sequences. *IEEE Trans. Consum. Electron.* **2004**, *50*, 911–915. [CrossRef]
31. Pfaff, J.; Schwarz, H.; Marpe, D.; Bross, B.; De-Luxán-Hernández, S.; Helle, P.; Helmrich, C.R.; Hinz, T.; Lim, W.Q.; Ma, J.; et al. Video compression using generalized binary partitioning, trellis coded quantization, perceptually optimized encoding, and advanced prediction and transform coding. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 1281–1295. [CrossRef]
32. Jung, S.; Jun, D. Context-based inter mode decision method for fast affine prediction in versatile video coding. *Electronics* **2021**, *10*, 1243. [CrossRef]
33. Ren, W.; He, W.; Cui, Y. An improved fast affine motion estimation based on edge detection algorithm for VVC. *Symmetry* **2020**, *12*, 1143. [CrossRef]
34. Bossen, F.; Boyce, J.; Li, X.; Seregin, V.; Sühring, K. JVET common test conditions and software reference configurations for SDR video. In Proceedings of the Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 10th Meeting, San Diego, CA, USA, 10–20 April 2018; Volume 16, pp. 19–27.
35. Gisle, B. Improvements of the BD-PSNR model. In Proceedings of the ITUT SG16/Q6, 34th VCEG Meeting, Berlin, Germany, 16–18 July 2008.