Block-Matching-Based Implementation of Affine Motion Estimation for HEVC

Chihiro TSUTAKE† and Toshiyuki YOSHIDA†, Members

SUMMARY Many of affine motion compensation techniques proposed thus far employ least-square-based techniques in estimating affine parameters, which requires a hardware structure different from conventional block-matching-based one. This paper proposes a new affine motion estimation/compensation framework friendly to block-matching-based parameter estimation, and applies it to an HEVC encoder to demonstrate its coding efficiency and computation cost. To avoid a nest of search loops, a new affine motion model is first introduced by decomposing the conventional 4-parameter affine model into two 3-parameter ones. Then, a block-matching-based fast parameter estimation technique is proposed for the models. The experimental results given in this paper show that our approach is advantageous over conventional techniques.

doi:10.1587/transinf.2017EDP7201

1. Introduction

Current video coding standards employ a hybrid coding framework, in which spatio-temporal redundancy is removed by a combination of prediction and transformation techniques [1], [2]. The latest video coding standard H.264/HEVC (High Efficiency Video Coding) and its predecessor H.264/AVC (Advanced Video Coding) as well as classical video coding standards employ intra- and inter-coding techniques: in intra coding, spatial redundancy is mainly eliminated with a transformation technique, whereas temporal redundancy is removed by motion compensation (MC) followed by a spatial transformation in inter coding [3]–[5].

Typical video coding techniques standardized thus far utilize the translational motion model

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  a & b \\
  -b & a
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  v_x \\
  v_y
\end{bmatrix},
\]

(1)

where \([x', y']^T\) and \([x, y]^T\) represent the coordinates on the reference and target frames, respectively, and \([v_x, v_y]^T\) is a motion vector (MV). This model is sufficient in relatively small coding blocks because a wide range of motion, e.g., zoom and rotation, can be approximated by translational motion. As briefly reviewed in the next section, HEVC employs a recursive splitting of coding units (CUs): a larger prediction unit (PU) is partitioned into smaller one so that the motion in each PU can be approximated by the translational model.

Recently, much effort has been made for affine motion compensation (AMC) [6]–[9] to improve the motion model (1). References [6]–[8] utilize the 4-parameter model

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  a & b & 0 \\
  -b & a & 0
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} + \begin{bmatrix}
  v_x \\
  v_y \\
  1
\end{bmatrix},
\]

(2)

where \(a\) and \(b\) represent the affine parameters. While this model requires signaling of more motion parameters than the translational one, it can compensate for a wide range of motions such as zoom and rotation for larger PUs. An introduction of such a model thus relieves over-splitting of PUs, which in turn reduces a bit cost for PU splitting.

Another advantage of AMC has been efficiently exploited in Refs. [7], [8], which can improve spatio-temporal motion-parameter prediction (MPP). Reference [7] has proposed a Block-to-Block Translational Shift Compensation (BBTSC) technique based on a spatial correlation of affine parameters, where the advanced motion vector prediction (AMVP) technique in HEVC [3]–[5] has been improved by compensating for prediction vectors with affine parameters. Reference [8] has proposed a Translational Motion Vector Field (TMVF) technique to further improve the BBTSC technique. As discussed in Sect. 3.2, these techniques can improve the coding efficiency as a by-product of AMC.

Compared with the translational model comprising an MV alone, the affine motion model requires more parameters to be estimated, resulting in much larger computation cost. As reviewed in Sect. 3.1, Refs. [6]–[9] implemented fast affine parameter estimation techniques in HEVC using least-square-based minimization methods. Although the techniques are much faster than an exhaustive block matching (BM) technique, hardware implementation of such a technique requires modules totally different from conventional BM-based ones, e.g., a gradient calculator and a linear equation solver. Furthermore, the least-square-based approach requires initial parameters to be optimized: good and global initial guess is necessary in particular for \([v_x, v_y]^T\) in Eq. (1), which can only be obtained with a BM-based approach. The techniques in Refs. [6]–[8] thus require both of least-square-based and BM-based modules, which results in severe overhead in actually implementation.

Based on the discussion thus far, this paper proposes
an efficient BM-based AMC framework, and demonstrates that it can achieve the coding efficiency and computation cost comparable to those of the least-square-based approach. In this paper, a new affine motion model is first introduced by decomposing Eq. (2) into two 3-parameter models so that their parameters can be estimated with a BM-based approach, and then their fast estimation technique is proposed. Finally, the proposed techniques are implemented in an HEVC encoder to evaluate its coding efficiency and computation cost.

The rest of this paper is organized as follows. Sections 2 and 3 briefly review the HEVC standard and the HEVC test model (HM) implementation, and related works for AMC, respectively. Section 4 elaborates the proposed techniques. Section 5 gives experimental results of the proposed technique and comparisons with the least-square-based approach. Finally, Sect. 6 concludes this paper.

2. HEVC Inter Coding and Implementation in HM

2.1 HEVC Inter Coding [3]–[5]

HEVC employs a quadtree structure of coding blocks referred to as Coding Tree Units (CTUs), which are recursively divided into smaller blocks called CUs. Each leaf block of the tree is referred to as a PU, which is utilized as a unit of prediction. HEVC inter coding allows symmetric/asymmetric CU partitioning, in which a PU size can be selected from a pool of sizes ranging from $64 \times 64$ to $8 \times 4$ or $4 \times 8$, depending on texture/motion activity of each CU.

In HEVC, the syntax elements belonging to the slice segment data, e.g., transform coefficients, MVs, and so on, are entropy-coded by an arithmetic coding scheme referred to as Context-based Adaptive Binary Arithmetic Coding (CABAC) [10], [11], which comprises the following three building blocks:

1. **Binarization**: The syntax elements fed into CABAC are binary or nonbinary data. Each nonbinary-valued syntax element is first mapped into a unique binary sequence. In HEVC, one of the several binarization processes [3]–[5] is utilized depending on the syntax, and the binarized sequences are passed to the subsequent regular/bypass coding stage.

2. **Context Modeling**: The regular coding stage utilizes an individual probability model corresponding to each bit of binarized syntax elements, which is known as a context model (CM). Each CM preserves the probability of the most probable symbol (MPS) for each bit, and updates the probability depending on the bit fed into CABAC. The probability is represented by 64 states initialized in terms of the quantization parameter (QP) and prescribed value referred to as ‘initValue’. In case the probability of MPS is close to 0.5, the bypass coding is selected with the probability of MPS being fixed to 0.5.

3. **Arithmetic Coding**: An arithmetic coder converts a sequence of binarized syntax elements into a unique binary value between the interval 0 to 1 by recursively dividing the current interval depending on the probability of MPS, and outputs the binary value as a bitstream. In the regular coding stage, the corresponding CM is updated according to every input bit. The bypass coding stage, on the other hand, requires a lower computation cost than the regular one because the probability of MPS is fixed to 0.5 without a context modeling.

2.2 Implementation in HM

Although Rate-Distortion-based (RD) inter coding requires the optimal determinations of not only MVs but also PU sizes, an exhaustive search for all possible combinations of MVs and PU sizes leads to an unrealistic computation cost. This is the reason why a fast motion estimation (ME) technique has been implemented in the HM. Another problem in terms of the ME complexity is an evaluation of the RD cost function. The full RD cost function for ME/MC is defined as

$$J_{\text{mode}} = \text{SSE} + \lambda_{\text{mode}} B_{\text{mode}}.$$  \hfill (3)

where SSE represents the distortion in terms of the sum of squared error between the original and motion-compensated pixels, $B_{\text{mode}}$ is the total bit cost for encoding the current PU by CABAC, and $\lambda_{\text{mode}}$ is a Lagrange multiplier. Note that a straightforward evaluation of Eq. (3) leads to a very large computation cost.

To alleviate these problems, the HM employs a three-step MV search technique with simplified RD cost functions. In the first step, the optimal integer-pel MV is determined with the test-zone (TZ) search algorithm [12] by minimizing the simplified RD cost function

$$J_{\text{pred,sad}} = \text{SAD} + \lambda_{\text{pred}} B_{\text{pred}},$$ \hfill (4)

where SAD is the sum of absolute difference between the original and motion-compensated pixels, $B_{\text{pred}}$ is the bit cost predicted for encoding the MV of the current PU, and $\lambda_{\text{pred}}$ is a Lagrange multiplier. In the second and third steps, the optimal half- and quarter-pel MVs are obtained successively by searching 8 neighboring MV candidates centered at the MV found in the previous step. In these steps, the RD cost function

$$J_{\text{pred,sadt}} = \text{SATD} + \lambda_{\text{pred}} B_{\text{pred}}$$ \hfill (5)

is evaluated for each MV candidate, where SATD represents the sum of absolute Hadamard-transformed difference. In the HM, the three-step MV search technique is applied to each PU size in each CTU to find the optimal PU size and MV.

3. Review of Related Works

3.1 Least-Square-Based Affine Parameter Estimation

The simplest estimation technique for affine parameters
is a BM-based exhaustive search. Unfortunately, since such an approach requires an unrealistic computation cost, Refs. [6]–[9] commonly resort to an approach totally different from a BM-based technique, i.e., a least-square-based approach using Newton-Raphson technique.

The least-square-based approach first defines the ME error $E$ for a target PU $B$ as

$$E = \sum_{x \in B} \left( f_{tgt}(x) - f_{ref}(Ax + v) \right)^2,$$

(6)

where $f_{tgt}$ and $f_{ref}$ are the target and reference frames, respectively, and

$$A = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}, \quad x = \begin{bmatrix} x \\ y \end{bmatrix}, \quad v = \begin{bmatrix} v_x \\ v_y \end{bmatrix}.$$

Then, $E$ is minimized with respect to $A$ and $v$ to find the optimal affine parameters. In Newton-Raphson technique, the locally optimal parameter vector

$$p = \left[ a, b, v_x, v_y \right]^T$$

(7)

is estimated iteratively by a linearization of Eq. (6) with respect to $p$. Actually, a set of linear equation

$$H(p) = g$$

(8)

is derived by linearizing Eq. (6), where $H$ represents Hessian matrix of $E$, and $g$ is the gradient vector of $E$. The $n$-th solution $p(n)$ is updated with

$$p(n+1) = p(n) + \Delta p$$

(9)

by solving Eq. (8) for $\Delta p$. Once the locally optimal affine parameters are obtained, they are then quantized and entropy-coded to form a bitstream together with the other syntax elements.

3.2 Improvement of Motion Parameter Prediction

In HEVC, the AMVP technique derives the prediction vector for the current MV from a pool of the spatio-temporally neighboring PUs, and stores MVs as reference candidates with their minimal grid size $16 \times 16$ [3]–[5]. An introduction of AMC can improve AMVP and MPP under the assumption that the affine parameters $a$ and $b$ in Eq. (2) are almost constant in the current and neighboring PUs.

The BBTSC technique in Ref. [7] compensates for prediction MVs by using the block-to-block distance and spatially neighboring $a$ and $b$ in Eq. (2). The TMVF technique in Ref. [8] further improves the BBTSC technique. It first reduces the minimal grid size for reference MVs, and then derives the reference values for $a$, $b$, and $[v_x, v_y]$ with Householder reflections and a singular value decomposition.

3.3 Comments for Refs. [7], [8]

References [7], [8] claim that an application of AMC together with the MPP improvement technique gives about $-14.1\%$ and $-20.7\%$ BD-Rates, respectively, for the Low-Delay-P structure.

Although Refs. [7], [8] allow low-cost affine parameter estimation by virtue of the least-square-based approach, these techniques require a totally different architecture in hardware implementation compared with a conventional BM-based one. The rest of this paper thus proposes a new affine motion estimation/compensation framework friendly to BM-based parameter estimation and shows that our technique gives almost equivalent performance for the least-square-based approach.

4. Proposed Affine Motion Estimation/Compensation Framework and Fast Parameter Estimation Technique

4.1 Simplification of Affine Motion Model

The actual motion model in AMC should be determined by considering the tradeoff between MC efficiency and an ME cost. Although the 4-parameter model (2) can compensate for a wide variety of motions, a BM-based brute-force search leads to a nest of loops for $a$, $b$, and $[v_x, v_y]^T$, resulting in an unrealistic computation cost. To avoid the difficulty, the 4-parameter model is decomposed into two 3-parameter ones:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 + s & 0 \\ 0 & 1 + s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix},$$

(10)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & -r \\ r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix},$$

(11)

where $s$ and $r$ (both are defined as non zero) represent zoom and rotation factors, respectively.

As explained later, one of the three models (1), (10), and (11), denoted as $M_T$, $M_Z$, and $M_R$, respectively, is selected actually based on an RD optimization. Although the 3-parameter models are inferior to the 4-parameter one in terms of MC efficiency alone, total RD efficiency of the 3-parameter models is almost equivalent to, or slightly higher than that of the 4-parameter model, which will be demonstrated in Sect. 5.2.

As the transformed coordinate $[x', y']^T$ leads to a real number in general, a pixel interpolation is required both in encoding and decoding. References [6]–[8] generate 1/16-pel-precision pixel values upon request in the least-square process. Since, however, the BM-based approach makes multiple accesses to the same interpolated pixels, the request-based interpolation approach is quite inefficient. Although an upsampling of entire reference frame can improve the efficiency, it requires a large frame memory. To avoid the difficulties, this paper interpolates the pixel value on a real-valued coordinate $[x', y']^T$ by applying a bilinear interpolation for quarter-pel-accuracy pixels interpolated and stored.

\(^{†}\)A negative value of BD-Rate indicates that the target encoder outperforms the anchor one.

\(^{††}\)A linearized version of the rotation matrix.
with the set of low pass filters in HEVC. More specifically, a luma (chroma) value is calculated by a bilinear interpolation for the surrounding four pixels obtained by the 4-times (8-times) interpolation regulated in the HEVC standard.

Although an affine motion model generally gives higher MC performance over the translational one, its gain is often canceled with a bit cost required for additional affine parameters, which suggests that AMC is advantageous in larger PUs. The affine motion models are thus applied only to the largest 64 × 64 PUs.

4.2 Block-Matching-Based Fast Affine Parameter Estimation and Motion Model Selection

To avoid a nest of parameter loops, the proposed algorithm decomposes the simultaneous search of \((s, v)\) into an individual search of \(s\) and \(v\), based on an iterative greedy algorithm\(^1\). The procedure of the proposed technique is listed as follows, where \(s^{(k)}\) and \(v^{(k)}\) denote \(s\) and \(v\) in the \(k\)-th iteration, respectively, and \(N\) and \(\Delta\) represent the number of search points (candidates) and search step size for \(s\), respectively.

1. Let \(k = 0\).
2. **Initialization of MV:** The quarter-pel initial MV \(v^{(0)}\) is estimated with \(s^{(0)} = 0\) as in the HM.
3. **Initialization of \(s\):** The initial value \(s^{(0)}\) is estimated by the following three-step search with the MV fixed to \(v^{(0)}\). The first step obtains the optimal \(s\) in a step size of \(4 \cdot \Delta\), i.e., by searching every 4 point of the \(N\) candidates. Then, the second (third) step compares the neighboring 2 points centered at the previous candidate with a step size of \(2 \cdot \Delta\) (\(\Delta\)). If \(s^{(0)} = 0\), the iteration is terminated.
4. **Refinement of MV:** To find \(v^{(k+1)}\), this step searches the quarter-pel nine MV candidates centered at \(v^{(k)}\) with \(s^{(k)}\) fixed. If \(v^{(k+1)} = v^{(k)}\), the iteration is terminated.
5. **Refinement of \(s\):** \(s^{(k+1)}\) is estimated by searching the 3 points \(s^{(k)} - \Delta, s^{(k)}, \) and \(s^{(k)} + \Delta,\) with \(v^{(k+1)}\) fixed. If \(s^{(k+1)} = s^{(k)}\), the iteration is terminated.
6. \(k\) is incremented, and if \(k\) is equal to the maximum number of iteration \(K\), the iteration is terminated. Otherwise, the steps 4 and 5 are repeated.

The maximum number of iteration \(K\) is determined in Sect. 5.1. In each step, the RD-optimal \(s\) and \(v\) are determined: in step 2, Eqs. (4) and (5) are utilized as in the HM while a modified version of Eq. (5) is applied in steps 3–6 with the bit amount of \(s\) or \(r\) included in the rate \(B_{\text{pred}}\). Since a precise evaluation for the bit amount of \(s\) or \(r\) is computationally intensive, the most pessimistic case of the bit amount is included in the bitrate \(B_{\text{pred}}\) of Eq. (5), which will be elaborated at the end of Sect. 4.3.2.

The RD-optimal motion model is finally determined by estimating the parameters for the models (1), (10), and (11), and by comparing the corresponding RD costs.

4.3 Number of Search Points \(N\), Search Step Size \(\Delta\), and Entropy Coding for \(s\) and \(r\)

4.3.1 Number of Search Points \(N\) and Search Step Size \(\Delta\)

Since the impacts of \(s\) and \(r\) on MC efficiency are equivalent, an identical number of search points \(N\) and search step size \(\Delta\) can be given for \(s\) and \(r\). In this paper, \(N = 16\) and \(\Delta = 1/256\) have been selected based on the experiments in Appendix A, which quantizes \(s\) and \(r\) into

\[
\begin{align*}
& s_i = i \cdot \Delta \\
& r_i = i \cdot \Delta
\end{align*}
\]

(12)

for the quantization index \(i\).

4.3.2 Entropy Coding

As the parameters \(s\) and \(r\) are spatially correlated over neighboring PUs, an introduction of a differential coding can improve coding efficiency. In encoding \(s\) and \(r\), the similar technique for MVs in HEVC is utilized: the differential coding is selected if the current PU and one of candidate reference PUs [3]–[5] have the same MC mode. Otherwise, a non-differential coding is applied.

To encode \(s\) or \(r\) as well as the selected MC mode, the syntax elements in Tables 1 (a) and (b) have been introduced. The selected MC mode \(M_T, M_Z, \) or \(M_R\) is signaled by \(C_1\) and \(C_2\). The quantization index in Eq. (12) is described by the index \(|i|\) and its sign \(S_s\) or \(S_r\) in the non-differential coding, where the signs of \(s\) and \(r\) have been separated to store each CM independently in CABAC. On the other hand in the differential coding, the difference of \(s\) or \(r\) is represented by \(\Delta i\) with its sign included. Note that no additional symbol is necessary to indicate the differential/non-differential mode because the mode can be uniquely determined in decoding by examining candidate reference PUs.

In this paper, based on the experiments in Appendix A, \(C_1, C_2, S_s,\) and \(S_r\) are encoded with the regular mode in CABAC while \(\Delta i\) and \(|i|\) are represented by a truncated unary code and encoded with the bypass mode in CABAC.

### Table 1 Syntax elements and semantics for \(s\) and \(r\).

| Symbol | Semantics | 0 | 1 |
|--------|-----------|---|---|
| \(C_1\) | affine model flag | translation \(M_T\) | affine \(M_T\) or \(M_P\) |
| \(C_2\) | zoom/rotation flag | zoom \(M_Z\) (10) | rotation \(M_G\) (11) |
| \(S_s\) | sign of \(s\) | negative \(s\) | positive \(s\) |
| \(S_r\) | sign of \(r\) | negative \(r\) | positive \(r\) |

| Symbol | Semantics | Range |
|--------|-----------|-------|
| \(\Delta i\) | difference of quantization index \(i\) | \([-16, 16]\) |
| \(|i|\) | absolute value of quantization index \(i\) | \([1, 8]\) |

\(^1\)This section elaborates the search algorithm for the model (10) for simplicity: the same algorithm can be applied to the model (11).
every QP, the CM of \( C_1 \) is initialized with ‘InitValue’ = 157, and \( C_2, S_s, S_r \) with ‘initValue’ = 154 [3]–[5].

In the fast affine parameter estimation technique in Sect. 4.2, the bit amount of \( s \) or \( r \) is ideally included in \( B_{\text{pred}} \) by encoding it in the same way as above. Since, however, this is computationally intensive, the most pessimistic bit amount is evaluated by virtually encoding \( C_1, C_2, S_s, S_r \) with the bypass mode and the corresponding bit amount is added to \( B_{\text{pred}} \) together with that for \( \Delta t \) or \( |l| \).

5. Experimental Results

To compare our technique with a least-square-based approach, the proposed fast affine parameter estimation technique was implemented in the HM 14.1 encoder together with the entropy coding scheme for the technique was included in the encoder because it can greatly improve the coding efficiency with a negligible computation cost.

In this section, the test sequences listed in Table 2 were encoded in the Low-Delay-P mode as in Ref. [7]. With the original HM 14.1 encoder as an anchor, the computation complexity and coding efficiency were evaluated, respectively, in terms of the BD-Rate [%] (Luma-Y) with QP = 22, 27, 32, and 37 and the reduction ratio of the encoding time

\[
\Delta T = \frac{T_{\text{larger}} - T_{\text{HM14.1}}}{T_{\text{HM14.1}}} [%].
\]

Table 2 Test sequences utilized in this paper. “Z”, “R”, and “T” in the third column denote Zoom, Rotation, and Translation, respectively.

| Test Seq. | Resolution | Dominant Motion |
|-----------|------------|-----------------|
| 1. RotatingDisk | 352 × 240 | R + T |
| 2. Cactus | 1920 × 1080 | R + T |
| 3. Drone1 | 1920 × 1080 | Z + R + T |
| 4. Drone2 | 1920 × 1080 | Z + R + T |
| 5. Station2 | 1920 × 1080 | Z + T |
| 6. BQSquare | 416 × 240 | Z + T |
| 7. FungusZoom | 416 × 240 | Z |
| 8. InToTree | 1280 × 720 | Z |
| 9. Kimono1 | 1920 × 1080 | T |
| 10. ParkScene | 1920 × 1080 | T |

The fast affine parameter estimation technique in Sect. 4.2 is first optimized with respect to the maximum number of iteration \( K \), and then the proposed technique is compared with Ref. [7].

5.1 Optimization for Maximum Number of Iteration \( K \)

The test sequences 1–8 were encoded by the proposed technique with \( K \) varied from 0 to 4. Figure 1 shows the BD-Rate and \( \Delta T \) averaged over 8 sequences for each \( K \). It can be observed from Fig. 1 that \( \Delta T \) linearly increases for a variation of \( K \), while the BD-Rate almost saturates at \( K = 1 \). This paper employs \( K = 1 \) by taking the computation complexity and coding efficiency into account.

5.2 Comparison with Least-Square-Based Approach

The proposed technique is first compared with Ref. [7] to demonstrate the advantages of the 3-parameter MC models and the BM-based affine parameter estimation technique. To confirm the efficiency of the BM-based technique alone, our technique is compared with one in which the BM-based technique is replaced with a least-square approach, denoted as “3-parameter least square”. All the test sequences in Table 2 were encoded by the three encoders, and BD-Rates and \( \Delta T \) were evaluated for each sequence, which is summarized in Table 3.

From Table 3, the proposed technique outperforms

Table 3 Comparisons of coding efficiency and computation complexity in terms of BD-Rate and \( \Delta T \).

| Test Seq. | Proposed | Ref. [7] (our impl.) | 3-parameter least square |
|-----------|----------|----------------------|-------------------------|
|           | BD-Rate [%] | \( \Delta T \) [%] | BD-Rate [%] | \( \Delta T \) [%] | BD-Rate [%] | \( \Delta T \) [%] |
| 1. RotatingDisk | −3.54 | 23.65 | −3.31 | 23.06 | −3.34 | 27.35 |
| 2. Cactus | −6.30 | 16.93 | −6.72 | 17.67 | −6.29 | 20.02 |
| 3. Drone1 | −2.36 | 20.90 | −2.35 | 19.00 | −2.23 | 27.84 |
| 4. Drone2 | −6.47 | 20.65 | −6.34 | 19.21 | −6.35 | 28.22 |
| 5. Station2 | −43.63 | 19.67 | −44.89 | 20.38 | −43.94 | 29.13 |
| 6. BQSquare | −1.93 | 22.97 | −1.54 | 24.60 | −1.85 | 30.05 |
| 7. FungusZoom | −24.74 | 29.14 | −23.55 | 28.99 | −24.48 | 33.07 |
| 8. InToTree | −8.08 | 21.55 | −8.17 | 21.62 | −7.45 | 32.28 |
| 9. Kimono | −0.30 | 13.46 | 0.13 | 14.84 | 0.10 | 20.87 |
| 10. ParkScene | −0.02 | 19.02 | 0.16 | 18.87 | 0.07 | 27.84 |
| Average | −9.74 | 20.79 | −9.46 | 20.82 | −9.58 | 28.27 |

Fig. 1 Averaged BD-Rate and \( \Delta T \) with varied \( K \).
Ref. [7] in terms of both criteria in average, which shows validity for both of the 3-parameter models and BM-based approach. The superiority of the 3-parameter models can be attributed to their low-overhead feature compared with the 4-parameter model. It is also verified from Table 3 that an introduction of AMC hardly affects the coding efficiency on sequences undergoing translational motion alone.

In view of the results comparing with the 3-parameter least-square approach, our technique is equivalent to or slightly superior to the least-square-based one in terms of the BD-Rate and ΔT, which illustrates that the affine parameters can be efficiently estimated with our approach as far as the 3-parameter models are employed.

6. Conclusions

AMC has been recognized as one of the promising techniques that can compensate for incompleteness of the translational model. Its intensive complexity, however, has been impeding an actual application to video encoders. Although least-square-based AMC techniques have thus been proposed in literature, a BM-based technique is still desirable in hardware implementation because of its simpleness.

This paper has thus proposed a BM-based AMC framework and implemented it in HEVC. To derive the new affine motion model friendly to BM-based parameter estimation, the conventional 4-parameter model was first decomposed into two 3-parameter ones, and then the BM-based fast parameter search technique was proposed. The search technique together with the encoding scheme for the affine parameters were implemented in the HM 14.1 encoder, and its efficiency and computation cost were compared with a least-square-based approach.

As a result, it has been confirmed that the proposed technique gives higher coding efficiency with less computational complexity, which opens up a new prospect in a realization of AMC with a BM-based technique.

Although the computation cost has been drastically reduced with the proposed technique, the coding efficiency has been slightly degraded compared with the full search technique. It is thus necessary to improve the proposed ME technique so that it can achieve almost the same efficiency as the full search, which is left for future work.

References

[1] Y. Sakai and T. Yoshida, Image Information Coding, Ohmsha, Tokyo, 2001 (in Japanese).
[2] T. Yoshida, T. Suzuki, and T. Hiroaki, Image Coding, Corona Publishing, Tokyo, 2008 (in Japanese).
[3] T. Suzuki, S. Takamura, and T. Chujoh ed., H.265/HEVC Textbook, Impress Japan, Tokyo, 2013 (in Japanese).
[4] M. Wien, High Efficiency Video Coding, Springer, 2015.
[5] G.J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” IEEE Trans. Circuits Syst. Video Technol., vol.22, no.12, pp.1649–1668, Dec. 2012.
[6] C. Heithausen and J.H. Vorwerk, “Motion compensation with higher order motion model for HEVC,” IEEE International Conference on Acoustic Speech and Signal Processing, pp.1438–1442, April 2015.
[7] C. Heithausen, M. Blüser, M. Wien, and J.-R. Ohm, “Improved higher order motion compensation in HEVC with block-to-block translational shift compensation,” IEEE International Conference on Acoustic Speech and Signal Processing, pp.2008–2012, Sept. 2016.
[8] C. Heithausen, M. Meyer, M. Blüser, and J.-R. Ohm, “Temporal prediction of motion parameters with interchangeable model,” Data Compression Conference, pp.400–409, April 2017.
[9] L. Li, H. Li, D. Liu, Z. Li, H. Yang, L. Sixin, H. Chen, and F. Wu, “An efficient four-parameter affine motion model for video coding,” IEEE Trans. Circuits Syst. Video Technol., pp.1–14, April 2017.
[10] D. Marpe, H. Schwarz, and T. Wiegand, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,” IEEE Trans. Circuits Syst. Video Technol., vol.13, no.7, pp.620–636, July 2003.
[11] V. Sze and M. Budagavi, “High throughput CABAC entropy coding in HEVC,” IEEE Trans. Circuits Syst. Video Technol., vol.22, no.12, pp.1778–1791, Oct. 2012.
[12] N. Purnachand, L.N. Alves, and A. Navarro, “Fast motion estimation algorithm for HEVC,” IEEE Second International Conference on Consumer Electronics, pp.34–37, Sept. 2012.
[13] R. Gallager and D.V. Voorhis, “Optimal source coded for geometrically distributed integer alphabets,” IEEE Trans. Inf. Theory, vol.21, no.2, pp.228–230, March 1975.
[14] N. Merhav, G. Seroussi, and M.J. Weinberger, “Optimal prefix codes for sources with two-sided geometric distributions,” IEEE Trans. Inf. Theory, vol.46, no.1, pp.121–135, Jan. 2000.

Appendix A: Determination of Parameters and Coding Scheme for the Proposed Encoder

A realization of the proposed encoder requires a determination of the number of search points N and search step size Δ for s and r, and their entropy coding scheme.

A simultaneous and the RD-optimal determination of N and Δ together with their entropy coding scheme is actually impossible because of its computation complexity. Therefore, in Appendix A.1, the candidate values of N and Δ are first determined so that the distortion D can be minimized in terms of the BD-PSNR with the bitrate R for s and r neglected. Then in Appendix A.2, the entropy coding strategy for s and r is investigated based on their statistics. Finally in Appendix A.3, the RD-optimal N and Δ are determined with the bitrate R taken into account using the entropy coding scheme.

Since the fast search technique proposed in Sect.4.2 can find near-optimal affine parameters, that may affect the statistics to be obtained in this appendix. A brute-force ME technique, referred to as “full search”, is thus employed in this appendix, where the HM three-step MV search is nested in the search of s or r.

A.1 Candidates of N and Δ

To evaluate the distortion D in terms of the BD-PSNR with N and Δ varied, the test sequences 1–8 in Table 2 were encoded by the proposed encoder employing the full search. Table A-1 shows the BD-PSNR averaged over the 8 se-
sequences for each \( N \) and \( \Delta \).

From the viewpoint of the coding efficiency and computation cost, \( N \) and \( \Delta \) that maximizes the BD-PSNR with the least \( N \) is optimal. Since the BD-PSNR saturates at \((N, \Delta) = (32, 1/256)\) in Table A-1, the combination together with its neighbors \((16, 1/256)\) and \((16, 1/128)\) were selected as the candidates for \( N \) and \( \Delta \).

### A.2 Entropy Coding Strategy for \( s \) and \( r \)

\( s \) and \( r \) are entropy-coded based on the syntax elements \( C_1, C_2, S_s, S_r, || \), and \( \Delta \) as defined in Sect. 4.3.2. This appendix discusses the entropy coding strategy for \( s \) and \( r \) based on their statistics. Let \( P_{C_1}(x) \), \( P_{C_2}(x) \), \( P_{S_s}(x) \), and \( P_{S_r}(x) \) \((x = 0 \text{ or } 1)\) denote the probability distributions for the binary elements \( C_1, C_2, S_s, \) and \( S_r \). To evaluate the probability distributions, the test sequences 1–8 in Table 2 were encoded by the proposed encoder with the bitrate of \( s \) and \( r \) neglected for each of the candidate \((N, \Delta) = (32, 1/256), (16, 1/256), \) and \((16, 1/128)\) and \( \text{QP} = 22, 27, 32, \) and 37. The probability distributions obtained for each sequence are averaged over the 3 candidates and 4 QPs, which are listed in Table A-2.

Since many of the probabilities in Table A-2 are biased to 0 or 1, \( C_1, C_2, S_r, \) and \( S_r \) are encoded with the regular mode in the CABAC in our entropy coding strategy, where \( C_1 \) is initialized with ‘initValue’ = 157 while the others are with ‘initValue’ = 154.

In the same way, the probability distributions for the absolute value of quantization index \(|\bar{I}|\) and difference of quantization index \(\Delta i\) were obtained for each candidate \((N, \Delta)\), as depicted in Figs. A-1 and A-2. Each of the probability distributions in Figs. A-1 and A-2 can be approximated with one-sided and two-sided geometric distributions

\[ P_o(x) = \left(1 - \theta_0\right) \theta_0^{x-1} \quad (x = 1, 2, \ldots, N/2) \quad (A-1) \]

and

\[ P_i(x) = \frac{1 - \theta_i}{1 + \theta_i} \theta_i^{|x|} \quad (x = -N, 0, \ldots, N), \quad (A-2) \]

respectively. This paper employs truncated Golomb-Rice code for \(|\bar{I}|\) and truncated Golomb code for \(\Delta i\) because they are optimal for symbols with geometric distributions. Actually, the optimal code [13], [14] can be derived from the parameters \(\theta_0\) and \(\theta_i\) in Eqs. (A-1) and (A-2), which can be obtained by least square approximations for Figs. A-1 and A-2 as shown in the dashed lines. These codes are finally fed into the bypass mode in the CABAC.

### A.3 Determination of \((N, \Delta)\) and Detailed Entropy Coding Scheme

Finally, the optimal combination \((N, \Delta)\) is determined in the
3 candidates by evaluating each of the BD-Rate and ΔT with the bitrate of s and r incorporated by designing the optimal codes based on the discussion in Appendix A.2. Table A·3 shows the BD-Rate (QP = 22, 27, 32, and 37) and ΔT for each candidate (N, Δ). By taking both of the BD-Rate and ΔT into account, (N, Δ) = (16, 1/256) has been selected as the optimal combination.

Least square approximations for Figs. A·1 and A·2 with (N, Δ) = (16, 1/256) give θo = 0.54 and θt = 0.42, from which the optimal codes for |i| and Δi can be constructed. Actually in this case, these codes lead to the truncated unary code [13], [14].

| Test Seq. | N = 32, Δ = 1/256 | N = 16, Δ = 1/256 | N = 16, Δ = 1/128 |
|-----------|-------------------|-------------------|-------------------|
| 1. RotatingDisk | -3.79 5121 | -3.30 3897 | -3.10 4043 |
| 2. Cactus | -6.80 4822 | -6.52 3304 | -6.00 3224 |
| 3. Drone1 | -2.13 5943 | -2.57 4281 | -2.03 4052 |
| 4. Drone2 | -6.07 5840 | -6.30 4176 | -4.34 3955 |
| 5. Station2 | -45.50 5980 | -43.94 4053 | -44.49 4176 |
| 6. BQSquare | -1.96 4486 | -1.97 3430 | -0.95 3027 |
| 7. FungusZoom | -25.56 5677 | -25.21 4538 | -20.76 3999 |
| 8. InToTree | -7.13 5741 | -7.97 4069 | -5.79 3855 |
| Average | -12.44 5451 | -12.22 3969 | -10.93 3791 |

Chihiro Tsutake received the B.E. and M.E. from University of Fukui, Japan, in 2015, 2017. He is currently a doctor course student of Graduate school of Engineering, University of Fukui, where he is doing a research on video coding.

Toshiyuki Yoshida received the B.E., M.E., and D.E. all in electronics from Tokyo Institute of Technology, Japan, in 1986, 1988, 1991. He was an associate professor in Tokyo Institute of Technology in 1994. He has moved to University of Fukui in 2003, where he is currently a professor in Dept. of Information Science. His research interests covers the image/video and signal processing and 3-D imaging. He is a co-author of several books, e.g., “Image Coding” (in Japanese, Corona Publishing) and “Image Information Coding” (in Japanese, Ohmsha).