

# NeuralHashを用いたVersatile Video Codingにおける動き推定の計算負荷低減手法

松野 駿樹<sup>1,a)</sup> 新田 高庸<sup>1,b)</sup>

## 概要：

本稿では、NeuralHash (NH) を用いた Versatile Video Coding (VVC) における動き推定 (Motion Estimation, ME) の計算負荷低減手法を提案する。NH を用いた ME では、ME の評価指標として従来用いられる差分絶対値和 (Sum of Absolute Differences, SAD) の代わりに 96 ビットの NH 値を用いる。VVC 参照ソフトウェア (VTM) で使用される TZSearch において、ブロックサイズが  $128 \times 128$  のときの Raster Search (ストライドは 8 を使用) 時に提案手法を実施した結果、画質劣化は BD-rate (Bjontegaard Delta rate) で 0.13 % 以下と小さく、符号化特性への影響は限定的であった。また、NH キャッシュを実装することで、NH 値を毎回計算する場合と比べて約 2.5 倍の高速化を実現した (QP = 47 の場合)。さらに、将来のハードウェアでの実装を意識し、NH 計算と映像符号化処理を分離するため、NH 値の事前計算手法も提案する。事前計算手法では、参照画像に原画像を用いており、さらに、探索の中心点を事前計算した点に移動させているにもかかわらず、ストライドが 4 である条件で、BD-rate 0.10 % 以下であることを確認した。

## 1. はじめに

Versatile Video Coding (VVC)[1] は、2020 年に国際標準化された高効率な映像符号化方式であり、HEVC (High Efficiency Video Coding) [2] と比較して約 2 倍の圧縮効率を達成している。この高い圧縮効率により、インターネット動画配信や地上波放送のような帯域制約のある環境でも高品質な映像を提供できる。一方で、圧縮率向上のために符号化処理は複雑化しており、動き推定 (Motion Estimation, ME) の計算量が大幅に増加している。

ME は、フレーム間の動きを予測して時間的冗長性を削減する映像圧縮の中核処理である (図 1)。ME では、ブロック単位で画素の類似度を評価し、最適な動きベクトルを決定する。この類似度評価には一般に差分絶対値和 (Sum of Absolute Differences, SAD) が用いられる。しかし、ブロックサイズが大きくなるほど SAD の計算量は急増する。特に VVC は最大ブロックサイズが HEVC の  $64 \times 64$  画素から  $128 \times 128$  画素へと拡張されているため、従来手法の計算負荷はさらに高くなるため、処理負荷削減が求められている。

この課題に対して、NeuralHash (NH) を利用する手法

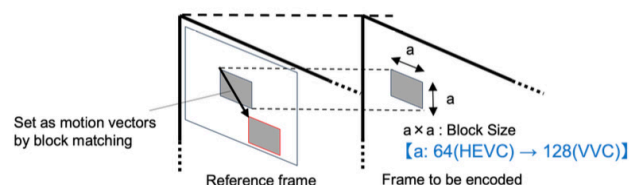


図 1 動き推定 [3].

が提案されている [3]。NH は画像を 96 ビットのハッシュ値に変換し、類似した画像どうしを高速に比較できる知覚ハッシュ手法である。SAD のように画素数に応じて計算量が増える処理とは異なり、NH はブロックごとの類似度をハッシュ値の比較で扱えるため、大きなブロックを用いる場合に特に有効であると考えられる。一方で、NH を用いた ME ではハッシュ値の生成に Deep Neural Network (DNN) 推論処理を伴うため、計算コストの評価と削減方法の検討が課題となる。

そこで本稿では、NH を用いた ME において、一度計算した NH 値を再利用することで処理負荷を削減する手法を提案し、その有効性を評価する。特に、符号化中に NH を逐次計算する構成と、GOP (Group of Pictures) 単位で NH キャッシュを事前生成する構成を実装し、符号化特性および処理時間の観点から比較評価することで、NH を用いた ME を実用的な高速化へ結び付けるための条件を明ら

<sup>1</sup> 会津大学  
The University of Aizu  
<sup>a)</sup> m5291048@u-aizu.ac.jp  
<sup>b)</sup> koyo@u-aizu.ac.jp

かにする。

第2章では本稿で用いる NH を用いた ME について説明する。第3章では、NH を用いた ME の高速化手法において、符号化中に NH を逐次計算する構成の場合と、GOP 単位で NH を事前計算する場合とについて示す。これらの提案手法を第4章で評価し、第5章でまとめる。

## 2. NeuralHash を用いた動き推定

### 2.1 NeuralHash

NeuralHash (NH) は、Apple 社が開発した、ディープラーニングを用いた知覚ハッシュ技術である [4], [5]。入力画像に対して前処理と推論を行い、最終的に 96 ビットのハッシュ値を出力する。類似した画像は近いハッシュ値を生成する性質を利用することで、画像間の類似度を高速に比較できる。

図2に、評価映像として用いた Basketball の第1フレームと Cactus の第1フレームおよび第16フレームに対して NH で得られたハッシュ値の例を示す。特に Cactus の2フレームは類似度が高く、NH 値間の XOR (排他的論理和) ハミング距離も小さい。

### 2.2 NeuralHash を用いた動き推定手法

VTM における ME では、TZSearch と呼ばれる動き推定手法が用いられる [7]。TZSearch は Diamond Search と呼ばれる探索を基本とするが、それによりよい動きベクトルが見つからないときは、Raster Search と呼ばれる探索を実施する。Raster Search では、参照画像の特定の探索範囲内を一定間隔で調べる。この一定間隔の値を以後 iRaster と呼ぶ。

本稿では、NH の有効性を確認するため、128×128 画素のブロックサイズの Raster Search において、SAD の代わりに、NH 値を用いた評価を実施する。具体的には、符号化対象のブロックの NH 値と、参照画像内の参照候補ブロックの NH 値とを求め、それらの XOR の結果の 1 の数を評価値とする。評価値  $Cost_{NH}$  は次式で表せる。

$$Cost_{NH} = \sum_{i=1}^{96} [H_i \neq R_i] \quad (1)$$

ここで、 $H_i$  は符号化対象ブロックのハッシュ値の  $i$  ビット目、 $R_i$  は参照候補ブロックのハッシュ値の  $i$  ビット目を、それぞれ表す。 $Cost_{NH}$  が最小となる候補を、最も類似したブロックとして動きベクトルの候補に採用する。

図3に、TZSearch において NH 評価を適用する位置を示す。本図に示すように、本実装では Raster Search において、128×128 ブロックの場合に NH に基づく評価を適用し、それ以外の条件では既存の SAD ベースの評価へ切り替える。Raster Search で得られた候補に対しては、後段の2点探索やダイヤモンド探索など、従来の refinement

をそのまま適用する。

### 2.3 計算負荷

NH ではニューラルネットワークによる推論処理を伴うため、SAD とは異なる計算特性を持つ。SAD は画素ごとの差分の絶対値を総和する単純な演算であるのに対し、NH は固定サイズへの正規化と推論を通じてハッシュ値を生成する。そのため、ブロックサイズが変わっても処理の流れは大きく変わらない一方で、推論処理そのものは浮動小数点演算を多く含むため、計算コストは小さくない。

NH を用いた ME で最も計算量が多い処理は、NH 値算出のための推論である。本稿では、NH 値をキャッシュに保存して再利用したり、事前計算しておくことにより、推論回数を削減し、全体の処理時間を低減する。次章では、この考え方に基づく提案手法について述べる。

## 3. 提案手法

### 3.1 符号化中に NH を逐次計算する構成における高速化

提案手法では、輝度成分の 128×128 ブロックを対象に NH 値を計算し、一度得られた値をブロック位置と種別 (符号化対象または参照画像) を表すキーとともに保持する。同一位置かつ同一種別のブロックが再び現れた場合には、保持済みの値を再利用することで、NH の推論回数を削減する。

提案手法の中核は、NH 値をブロック単位で保持するキャッシュである。キャッシュのキーは、フレーム番号を表す POC、画面上の座標  $(x, y)$ 、および符号化対象ブロックと参照候補ブロックを区別する種別から構成する。ここで、種別を区別するのは、同一フレーム番号、同一座標であっても、符号化対象ブロックは符号化前のデータであり、参照候補ブロックは符号化後のデータであるので、NH 値は異なるためである。NH キャッシュは符号化中、メモリ上で管理し、同一 GOP 内で再利用する。これにより、同じブロックが複数回現れても、NH 値を毎回推論し直す必要がなくなる。

提案手法では、NH 値を求める際にまず NH キャッシュからの取得を試みる。キャッシュに存在する場合は再利用し、存在しない場合は NH を計算してキャッシュへ追加する。取得した符号化対象ブロック、参照候補ブロックの2つの NH 値のハミング距離を計算し、最小距離の候補を採用する。図4に、提案手法の処理の流れを示す。

### 3.2 NH キャッシュ事前生成による高速化

NH を符号化中に逐次計算する構成では、符号化性能への影響は小さい一方で、推論処理が大きなオーバーヘッドとなる。そこでここで、符号化開始前に GOP 分の NH キャッシュをまとめて生成し、符号化時にはそのキャッシュを参照する構成を評価する。図5にその概要を示す。



(a) Basketball 第1フレーム  
4f986d7c687be12655fddf69



(b) Cactus 第1フレーム  
e2d96f9becfaa97063618037



(c) Cactus 第16フレーム  
e2d9679af8bae9f0626580b3

図2 NeuralHash によるハッシュ値の例.

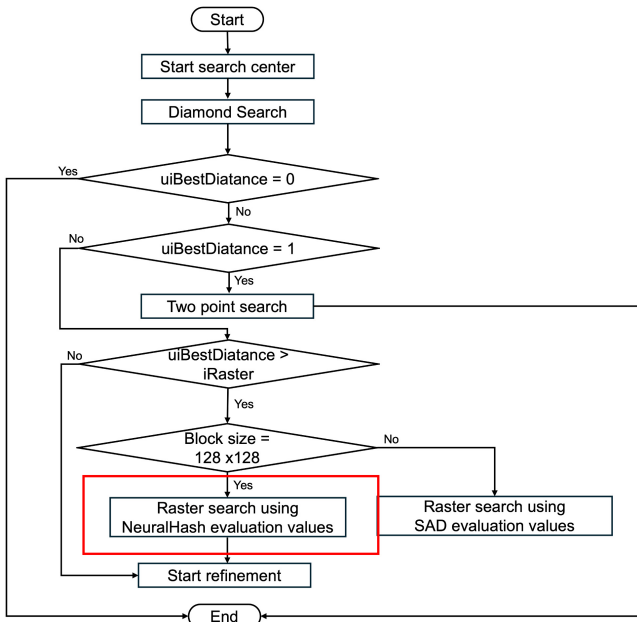


図3 TZSearch における NeuralHash 評価の適用位置.

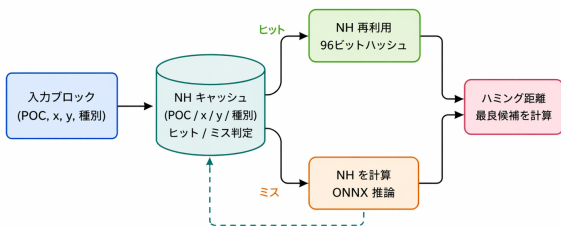


図4 NeuralHash キャッシュを用いた提案手法の概要.



図5 GOP 単位の NH キャッシュ事前生成の概要.

本方式の狙いは、NH の計算タイミングを符号化ループの内側から外側へ移すことである。将来的に NH を用いた ME をハードウェアで実装する場合、NH 計算回路と ME 回路との処理を並列に実行可能となるためである。本方式

では、符号化開始前に原画像から GOP 分の NH キャッシュを生成し、符号化時はそのキャッシュを参照する。NH の計算はフレームおよびブロックごとに独立しているため、事前計算として並列化することが可能となる。

ただし、ここで以下の2つの点に注意が必要である。まず第一に、本来は参照画像を用いて NH を計算すべきであるが、参照画像は符号化中に生成される局所復号画像であるため、事前生成では利用することができない。そこで、本方式では局所復号画像の代わりに原画像を用いることとしている。そのため、NH キャッシュにおける種別は不要となる。

次に、任意の座標での NH 値が必要となる可能性があるが、それらあらゆる NH 値を求めるには処理量が膨大になるため、事前生成は  $x$  座標、 $y$  座標ともに、iRaster を法として 0 となる点を左上とするブロックのみについて計算している。Raster Search の探索の中心がこのような点ではない場合、もっとも近い事前生成された NH 値で代用している。これらにより、探索精度は劣化するが、その後 refinement 処理があることから、符号化性能に与える影響は必ずしも大きくないことが予想される。実際の影響は次節で明らかにする。

## 4. 評価

従来の SAD ベースの ME を用いる VTM (以下、VTM<sub>original</sub>) と、1) NH を逐次計算する構成において NH キャッシュを用いた手法を実装した VTM (VTM<sub>NH</sub>)、および、2) GOP 単位の NH キャッシュ事前生成手法を利用した VTM (VTM<sub>NH2</sub>) との2実装とを比較し、NH 値の ME への有効性、NH 値の再利用による処理時間に与える影響、事前計算による符号化性能への影響、などを評価する。

### 4.1 VTM<sub>NH</sub> の実験

実験には VTM 23.0 を用いた。実験機は MacBook Pro (13-inch, 2020, Four Thunderbolt 3 ports) であり、2 GHz クアッドコア Intel Core i5, Intel Iris Plus Graphics 1536 MB, 16 GB 3733 MHz LPDDR4X を搭載してい

表 1 VTM\_NH の RD 結果.

シーケンス	QP	VTM_original		VTM_NH	
		Rate [kbps]	Y-PSNR [dB]	Rate [kbps]	Y-PSNR [dB]
Basketball	27	4911.07	40.39	4915.27	40.39
	32	2678.33	38.56	2682.07	38.56
	37	1500.15	36.38	1499.10	36.37
	42	879.11	33.99	879.53	33.98
	47	510.89	31.30	510.75	31.29
Cactus	27	5575.03	37.28	5571.30	37.28
	32	2910.25	35.67	2913.42	35.67
	37	1535.38	33.69	1536.12	33.68
	42	812.40	31.55	812.51	31.55
	47	416.70	29.26	417.37	29.25
CatRobot	27	11583.37	39.73	11577.69	39.73
	32	6264.36	38.64	6264.35	38.64
	37	3516.32	37.09	3519.77	37.09
	42	2011.74	35.12	2014.47	35.13
	47	1113.21	32.75	1115.93	32.75

る。符号化設定は RA (Random Access) を基にし、量子化パラメータ (Quantization Parameter, QP) は 27, 32, 37, 42, 47, iRaster は 8 とした。

入力シーケンスには、ハイビジョン・システム評価用標準動画第 2 版 [6] に含まれる Basketball (1920×1080, 10 bit, 60 fps), および映像符号化標準化活動で用いられる Cactus (1920×1080, 8 bit, 50 fps), CatRobot (3840×2160, 10 bit, 60 fps) を用いた。いずれも先頭 33 フレームを対象とし、それぞれのシーケンスを VTM\_original と提案手法である VTM\_NH, VTM\_NH2 の 2 実装でそれぞれ符号化した。

図 6 に、Basketball, Cactus, CatRobot の RD (Rate-Distortion) 曲線を示す。Y-PSNR (輝度信号の Peak Signal to Noise Ratio) 基準の BD-rate は、Basketball で 0.13 %, Cactus で 0.08 %, CatRobot で 0.02 % であった。いずれのシーケンスでも RD 曲線はほぼ重なっており、NH が ME の評価値として有用であることを表している。参考として、YUV-PSNR (輝度信号および色差信号の加重平均 PSNR) 基準の BD-rate は、Basketball で 0.09 %, Cactus で 0.05 %, CatRobot で 0.01 % であった。

表 1 に、各 QP におけるビットレートと Y-PSNR を示す。VTM\_original と VTM\_NH の値は近く、RD 曲線がほぼ重なっていることと整合する。

表 2 に、各 QP における処理時間を示す。VTM\_NH については総エンコード時間、NH 計算時間、および NH 計算時間を差し引いた NH 除外後の総時間を示す。また、VTM\_NH2 については、GOP の事前キャッシュ生成時間を含めない条件でのエンコード時間を示す。

加えて、NH キャッシュ自体の効果を確認するため、キャッシュを用いずに NH 値を逐次計算する構成との比較も行った。QP = 47 の条件では、NH キャッシュを用いることで、NH 値を毎回計算する場合と比べて約 2.5 倍の高速化が得られた。ただし、表 2 に示すように、VTM\_NH の総処理時間は VTM original より長く、NH 推論処理そのものが依然として主要なオーバーヘッドである。

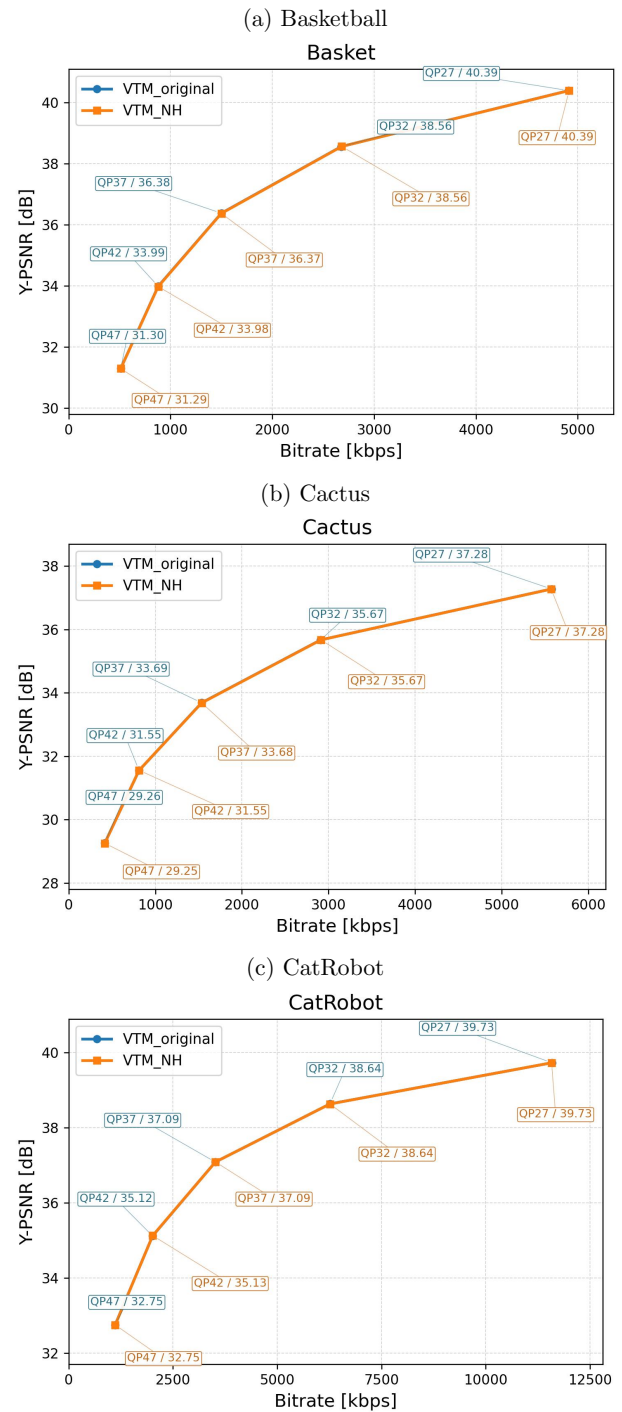


図 6 各シーケンスにおける RD カーブ

## 4.2 VTM\_NH2 の実験

VTM\_NH2 の実験では、NH が事前計算された座標と探索点のずれがなるべく少なくなるように、iRaster は 4 とした。VTM\_original と VTM\_NH2 を用い、前節と同様の実験を実施した。なお、処理時間は GOP の事前キャッシュ生成に要する時間を含めずに比較した。これにより、事前計算された NH が符号化品質と処理時間に与える影響を確認する。

図 7 に RD 曲線を、表 3 に BD-rate を、表 2 に処理時間を示す。



表 2 各手法の処理時間. – は未評価を表す.

シーケンス	QP	VTM_NH (iRaster = 8)				VTM_NH2 (iRaster = 4)	
		Orig. [s]	NH [s]	NH time [s]	NH excl. [s]	Orig. [s]	NH2 [s]
Basketball	27	7594.00	42093.69	30872.98	11220.71	7545.18	5482.26
	32	4778.69	24384.33	20273.41	4110.92	4925.90	3425.89
	37	3070.11	25064.25	22306.31	2757.93	3385.76	2298.39
	42	1883.24	24791.24	23037.81	1753.43	2135.15	1414.37
	47	1111.05	24278.58	23201.72	1076.86	1266.00	854.05
Cactus	27	8711.87	35129.36	24014.90	11114.46	8927.52	6241.82
	32	5265.42	31006.83	24214.33	6792.50	5631.44	3948.07
	37	3202.58	28558.81	24459.81	4099.00	3469.68	2449.74
	42	1850.73	28272.42	25871.22	2401.20	2059.64	1427.09
	47	1066.36	27667.33	26264.87	1402.46	1081.50	805.35
CatRobot	27	16719.78	123087.10	101309.84	21777.26	–	–
	32	10932.62	178806.39	155705.64	23100.76	–	–
	37	6069.64	115262.01	106464.86	8797.15	–	–
	42	3590.70	114386.27	108094.75	6291.52	–	–
	47	2106.95	113146.38	109311.53	3834.85	–	–

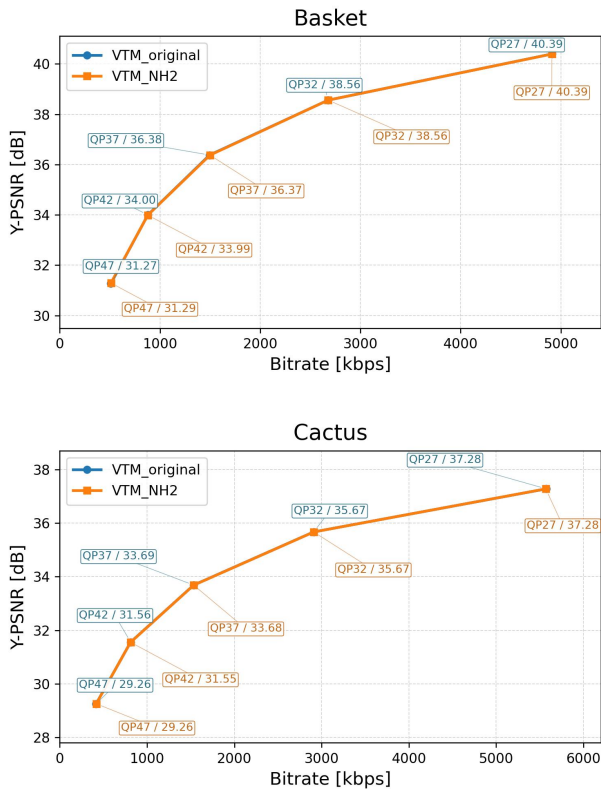


図 7 VTM\_original と VTM\_NH2 の RD 曲線.

表 3 VTM\_NH2 の BD-rate.

シーケンス	BD-rate [%]
Basketball	0.10
Cactus	0.06

表 2 では、QP が高くなるにつれて処理時間が短くなる傾向が確認できる。また、GOP の事前キャッシュ生成時間を含めない条件では、VTM\_NH2 のエンコード時間は VTM\_original より短い。したがって、本結果は、事前計算を除いた符号化ループ内の処理時間において、NH 再利用が負荷低減に寄与する可能性を示している。

表 3 の BD-rate は Basketball で 0.10 %、Cactus で 0.06 % であり、NH の再利用自体は符号化特性を大きく変えないことが分かる。

### 4.3 考察

VTM\_NH の評価では、3 シーケンスとも RD 曲線は VTM original とほぼ重なり、Y-PSNR 基準の BD-rate は 0.13 % 以下であった。このことから、本評価条件の範囲では、128 × 128 ブロックの Raster Search において SAD の代わりに NH 値のハミング距離を用いても、符号化性能への影響は小さいと考えられる。

一方で、VTM\_NH の総処理時間は VTM original より長く、NH 値算出のための推論処理が大きなオーバーヘッドとなっている。したがって、NH を用いた ME を高速化へ結び付けるには、NH 値の比較処理そのものよりも、NH 計算を符号化ループ内から切り離す構成が重要である。

VTM\_NH2 では、GOP 単位で事前生成した NH キャッシュを用いることで、事前キャッシュ生成時間を含めない条件では VTM original より短いエンコード時間となった。また、Basketball および Cactus の BD-rate は 0.10 % 以下であり、符号化性能への影響も小さかった。この結果は、NH 計算を前処理として符号化ループ外へ移す構成が、符号化中の処理負荷低減に有効である可能性を示している。ただし、本評価では事前キャッシュ生成時間を処理時間を含めていないため、今後は事前生成時間を含めた総処理時間で評価する必要がある。

### 5. 結論

本研究では、NeuralHash (NH) を用いた VVC の動き推定において、計算済みのハッシュ値をキャッシュして再利用する計算負荷低減手法を提案し、VTM へ実装して評価した。Basketball、Cactus、CatRobot の先頭 33 フレームを用いた評価では、VTM\_original と VTM\_NH の RD 曲

線はほぼ重なり、Y-PSNR 基準の BD-rate も Basketball で 0.13 %, Cactus で 0.08 %, CatRobot で 0.02 % と小さかった。したがって、本評価条件の範囲では、NH の導入は符号化特性を大きく損なわないことが確認された。

一方で、エンコード中に NH を逐次計算する VTM\_NH では、NH の推論処理が主要なオーバーヘッドとなり、総処理時間は VTM\_original より長くなった。別条件評価として、GOP 単位の事前キャッシュ生成を実装した VTM\_NH2 についても評価した。iRaster 4 条件での BD-rate は Basketball で 0.10 %, Cactus で 0.06 % にとどまり、事前キャッシュ生成時間を含めない条件では、エンコード時間は VTM\_original より短くなった。したがって、本評価条件の範囲では、NH を用いた ME を高速化へ結び付けるには、符号化ループ内から NH 計算を切り離す構成が有効である可能性が示唆された。

また、この構成は NH 生成部を前処理として独立化しやすく、比較処理もビット演算主体で構成できるため、ハードウェア実装を見据えた処理構造との親和性が高いと考えられる。今後は、事前キャッシュ生成時間を含めた総処理時間の評価と、キャッシュ生成処理の軽量化が必要である。

## 参考文献

- [1] Benjamin, et al., “Overview of the Versatile Video Coding (VVC) Standard and its Applications,” IEEE Trans. on Computer, vol. 31, no. 10, pp. 3736–3764, October 2021.
- [2] Xufeng Li, et al., “Fast Motion Estimation Methods for HEVC,” IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, August 2014.
- [3] Ryota Igarashi, “Motion Estimation Using NeuralHash for VVC Hardware Implementation,” Graduation thesis, The University of Aizu, 2024.
- [4] “CSAM Detection Technical Summary,” [https://www.apple.com/child-safety/pdf/CSAM\\_Detection\\_Technical\\_Summary.pdf](https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf)
- [5] Asuhariet Ygvar, “AppleNeuralHash2ONNX,” <https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX>
- [6] ITE/ARIB Hi-Vision Test Sequence 2nd Edition, <https://www.ite.or.jp/contents/chart/manual-rev1.3.pdf>
- [7] Z. Pan, Y. Zhang, S. Kwong, X. Wang, and L. Xu, “Early termination for TZSearch in HEVC Motion Estimation,” 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2013.