

動的メッシュ符号化技術の国際標準規格 V-DMC 対応 リアルタイムエンコーダの開発

Development of Real-time Encoder Compliant with Video-based Dynamic Mesh Coding

岸本 広輝^{*} 木谷 佳隆^{*}
Koki Kishimoto Yoshitaka Kidani

1. はじめに

写実的な三次元データ表現形式の一つとして動的メッシュが注目されており、メタバース等での応用が期待されている。しかし、動的メッシュはデータ量が膨大であるため、流通には高効率な符号化技術が不可欠である。これに対し、MPEG (Moving Picture Expert Group) は映像符号化のフレームワークを応用した映像ベースの動的メッシュ符号化方式 V-DMC (Video-based Dynamic Mesh Coding) [1], [2] を 2025 年 10 月の国際標準化に向けて開発している。V-DMC は、動的メッシュの幾何情報の大まかな構造と細部をそれぞれ基本メッシュと変位量で表現し、動的メッシュのテクスチャ情報と併せてデータ特性に応じて高効率に符号化する。しかし、MPEG が開発中の V-DMC テストモデル (TMM) [3] の符号化アルゴリズムは複雑であり、リアルタイムな処理速度が実現できていない。そこで本稿ではリアルタイムな処理速度を実現するための符号化アルゴリズムの並列化手法と簡素化手法を提案する。また、提案手法を実装したエンコーダを用いての処理速度と主観品質による提案手法の有効性評価結果を報告する。

2. V-DMC テストモデル (TMM)

開発した V-DMC 暫定仕様 [2] に対応のリアルタイムエンコーダは MPEG が開発している TMM [3] をベースにしている。TMM の符号化処理はエンコーダに入力される原メッシュの間引き処理（基本メッシュ生成処理）、変位量生成処理、テクスチャ生成処理、基本メッシュと変位量とテクスチャそれぞれの符号化処理から構成される。各処理を以下概説する。

原メッシュの間引き処理では Quadric Error Metric (QEM) [4] に基づいてメッシュの各辺の削除コストを算出し、コストの低い順に辺を削除して面数を減らす。変位量生成処理では、基本メッシュの面を再帰的に等分割して得られる細分割メッシュの各頂点に対する原メッシュの最近傍頂点の変位（すなわち変位量）を生成する。この細分割メッシュと変位量を符号化して伝送すれば、デコーダ側で原メッシュを近似する再構成メッシュが生成できる。テクスチャ生成処理では、原メッシュと再構成メッシュの面から頂点をサンプリングし、再構成メッシュの各サンプリング頂点に対する原メッシュの近傍サンプリング頂点を探索する。近傍サンプリング頂点の色情報から再構成メッシュのテクスチャを生成する。基本メッシュの符号化処理にはトポロジ符号化手法 [5] が、変位量とテクスチャの符号化処理には High Efficiency Video Coding (HEVC) [6] などの映像符号化手法がそれぞれ適用される。なお、変位量に映像符号化を適用するため、変位量の Wavelet 変換による係数算出、係数の二次元配置（画像情報に変換）も実施している。

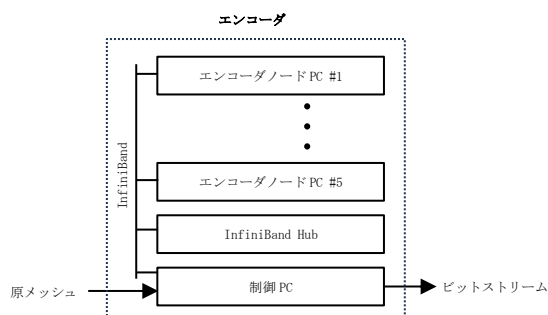


図1 リアルタイムエンコーダのハードウェア構成

3. リアルタイムエンコーダ

開発したリアルタイムエンコーダでは、TMM の符号化アルゴリズムの並列化手法と簡素化手法の導入によりリアルタイム処理を実現している。以降でリアルタイムエンコーダの構成と各手法を詳述する。簡素化手法については既発表の手法 [7] も含めて複数の手法で構成される。本稿では V-DMC の符号化処理において処理負荷が大きいテクスチャ生成処理と原メッシュの間引き処理を説明する。

3.1 リアルタイムエンコーダの構成と並列化手法

図1にリアルタイムエンコーダのハードウェア構成を示す。各エンコーダノード PC には映像符号化処理を担う GPU (RTX6000 または RTX4000) とそれ以外の符号化処理を担う CPU (Ryzen Threadripper 3990X) がそれぞれ1基ずつ搭載されている。制御 PC は GPU を搭載せず、CPU (Intel Core i9-13900K) のみを搭載している。ここで、TMM では変位量やテクスチャの映像符号化に CPU を使用しているが、本リアルタイムエンコーダではハードウェア HEVC エンコーダ (NVENC [8]) 対応の GPU を使用することで映像符号化の処理速度を向上させている。

リアルタイムエンコーダにはフレーム群 (GoF) 単位およびフレーム単位の並列化手法 [9] を導入している。具体的に、制御 PC は、InfiniBand を介して各エンコーダノード PC に原メッシュを GoF 単位で分配し、エンコーダノード PC 側では分配された GoF 単位の原メッシュを独立に処理する。さらに、各エンコーダノード PC は GoF 単位の原メッシュを符号化する際、フレーム単位で並列処理する。これにより、エンコーダ全体の GPU および CPU の使用率が向上し、原メッシュの1フレームあたりの符号化処理速度が向上する。GoF 並列数を n 、フレーム並列数を m とすると、全体の並列数は nm となり、理想的な条件下では nm 倍の処理速度向上が期待できる。

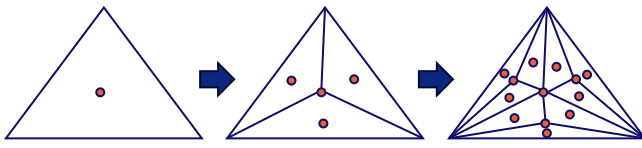


図2 重心サンプリング手順

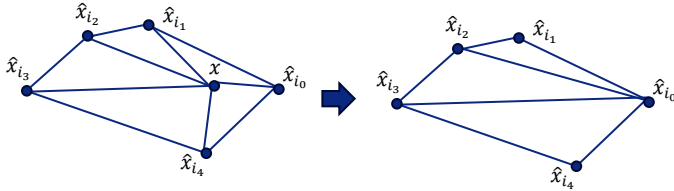


図3 ランダム間引き手順

3.2 符号化アルゴリズムの簡素化手法

テクスチャ生成処理と原メッシュの間引き処理の簡素化手法として、重心サンプリングを用いたテクスチャ生成手法と、ハイブリッド間引き処理手法を提案する。以降で各手法のアルゴリズムを TMM の手法と比較した上で詳述する。

TMM のテクスチャ生成では、初めに原メッシュと再構成メッシュそれぞれに対して、面ごとに法線を算出し、その法線に平行な光線を計算する。次に、算出した光線と面の交点をサンプリングし、再構成メッシュのサンプリング頂点の最近傍にある頂点の色情報をテクスチャの画素として計算する。しかし、サンプリング頂点の探索に必要な法線や交点の計算量は大きい。

これに対し、提案する重心サンプリングを用いたテクスチャ生成手法では、法線や交点を計算せずにサンプリング頂点を探索して、テクスチャが生成できる。図2に重心サンプリングの手順を示す。具体的には、重心サンプリングは任意の繰り返し回数 M に対して、以下の手順でサンプリング頂点を探索する。

1. 面の重心をサンプリング頂点とする。
2. 面の各頂点と重心を接続することで、新たな面を生成し、生成した面の重心をサンプリング頂点とする。
3. 2で生成した面に対して手順1, 2を M 回繰り返す。

TMM の間引き処理では、メッシュを構成するすべての辺に対して QEM を計算し、辺を削除することに削除対象の辺に接続する頂点を持つ各辺のコストを再計算する必要がある、計算量が大きい。

これに対し、提案するハイブリッド間引き処理手法では、TMM の間引き開始直後は面密度が高く、どの面を削減しても幾何情報の損失が少ない点に着目し、間引き開始直後はランダムに頂点を間引くことで面の数を削減する。具体的には、ランダム間引き率 $p[\%]$ 、QEM 間引き率 $q[\%]$ をそれぞれ設定し、原メッシュの面の数 N に対して、 $Np/100$ 個の面をランダム間引きで削減した後に、 $Nq/100$ 個の面を QEM で削減する。図3にランダム間引きの概要を示す。ランダム間引きは具体的には、以下のような手順で面を削減する。

1. 原メッシュからランダムに頂点 x を選択する。
2. 頂点 x から構成される辺のうち、最近傍の頂点 $\hat{x}$ を見つける。

表1 各データの情報

シーケンス	テクスチャ解像度	頂点数	面数	フレームレート [fps]
longdress	2048 x 2048	21,000	40,000	30
soldier	2048 x 2048	23,000	40,000	30
basketball_player	2048 x 2048	21,000	40,000	30
dancer	2048 x 2048	21,000	40,000	30
contemporary dancer	2880 x 2880	60,000	120,000	30

表2 符号化処理速度の比較

シーケンス	TMM	リアルタイムエンコーダ
longdress	0.013	52.9
soldier	0.011	57.2
basketball_player	0.013	67.2
dancer	0.013	64.0
contemporary dancer	0.015	32.5
average	0.013	54.8

3. 頂点 x を持つ面のうち、 x を $\hat{x}$ に置き換えることで面を再生成する。例えば、図3の間引き前のメッシュを構成する面 $(\hat{x}_{i_1}, \hat{x}_{i_2}, x)$ は、面 $(\hat{x}_{i_1}, \hat{x}_{i_2}, \hat{x}_{i_0})$ となる。
4. 手順3によって面の面積が0になる場合は、その面を削除する。例えば、図3の間引き前のメッシュを構成する面 $(\hat{x}_{i_1}, x, \hat{x}_{i_0})$ は、面 $(\hat{x}_{i_1}, \hat{x}_{i_0}, \hat{x}_{i_0})$ となり、面積が0となるため、面 $(\hat{x}_{i_1}, \hat{x}_{i_0}, \hat{x}_{i_0})$ は削除される。

4. 実験

4.1 提案手法の有効性評価

リアルタイムエンコーダに導入した提案手法の有効性を評価するため、多様な動的メッシュ素材に対する符号化処理速度を測定し、また、再構成メッシュの主観品質を TMM と比較した。使用した動的メッシュ素材は MPEG の V-DMC 向け共通テスト条件 (CTC) [10] で使用される素材 (longdress, soldier, basketball_player, dancer) と、自主撮影制作素材 contemporary dancer である。表1に各素材の諸元を示す。CTC 素材のテクスチャ解像度は 2048×2048 、頂点数は 20,000、面数は 40,000 である一方、自主撮影制作素材のテクスチャ解像度は 2880×2880 、頂点数は 60,000、面数は 120,000 であり、自主撮影制作素材の方が符号化処理のリアルタイム化実現難易度が高い。幾何情報とテクスチャ情報の総符号化ビットレートについては、CTC 素材と自主撮影制作素材ともにテクスチャ解像度が 4K 映像相当であるため、HEVC を用いた 4K 映像配信サービスで一般的な 10-15Mbps で符号化した。具体的には、CTC 素材と自主撮影制作素材の総符号化ビットレートがそれぞれ、10Mbps と 15Mbps になるような幾何情報とテクスチャ情報に対する量子化パラメータを TMM とリアルタイムエンコーダともに設定した。

表2は各メッシュ素材に対する TMM と開発したリアルタイムエンコーダの符号化処理速度を示している。リアルタイムエンコーダは、TMM と比較して、提案手法の導入により、符号化処理速度が平均 4,212 倍向上している。また、リアルタイムエンコーダの符号化処理速度は、すべてのメッシュ素材のフレームレート以上であり、リアルタイム処理を実現している。CTC 素材ではいずれの素材も 50fps 以上の符号化処理速度を実現している。これはリアルタイムエンコーダに導入した提案手法の汎用性能（異なる種類のメッシュ素材に対する汎用性）の高さを示している。自主撮影制作素材では、CTC 素材と比較して符号化処理速度が低い。これは、自主撮影制作素材が CTC 素材よりもテ



図4 再構成メッシュの比較

表3 各種条件下での符号化処理速度[fps]の評価

No.	ノード数	ハイブリッド間引き	重心サンプリング	符号化処理速度 [fps]
1	45	あり	あり	32.3
2	45	なし	あり	31.4
3	45	あり	なし	31.5
4	1	あり	あり	1.2
5	15	あり	あり	19.5
6	30	あり	あり	31.6

クスタ解像度が高いことと面数が多いために、テクスチャ生成処理と、間引き処理の処理量が大いことが要因として考えられる。

図4は、TMMとリアルタイムエンコーダによるCTC素材のdancerと自主撮影制作素材 contemporary dancerの再構成メッシュをそれぞれ正面方向から見た静止画像の比較である。各画像の比較により、TMMの再構成メッシュに対してリアルタイムエンコーダの再構成メッシュの品質低下が大部分の領域で視認されず、dancerのTシャツ下部の文字の精細さが失われる等の細部で品質低下が視認された。同様に、実験素材すべての再構成メッシュを動画で比較した結果、TMMに対してリアルタイムエンコーダの再構成メッシュの品質低下は大部分で観測されなかった。これによりリアルタイムエンコーダに導入した提案手法はTMMに対して符号化処理速度を向上しつつ、主観品質（すなわち符号化性能）をほとんど維持しているといえる。

表3に、リアルタイムエンコーダの各提案手法の有効性を評価するために contemporary dance に対して、異なる6種類の実験設定で測定した結果を示す。表3のNo.1はリアルタイムエンコーダ設定と同じ設定である。

No.1およびNo.4～6の比較により、ノード数（同時に並列処理するGoF数）の増加による符号化処理速度の向上が確認できる。一方でノード数の増加による速度向上はノード数45で飽和している。これに対し、No.1～No.3の比較により、符号化アルゴリズムの簡素化手法による符号化処理速度の向上が確認できる。以上の比較により、提案するV-DMC符号化アルゴリズムの並列化手法と簡素化手法による処理速度の向上、すなわち有効性が確認できる。

4.2 実網上で動的メッシュライブ伝送実験

メタバース上で動的メッシュをライブ配信するユースケースを想定し、ライブ配信の一連の処理（動的メッシュの撮影、符号化・伝送、受信・再生）の処理速度を検証する実験環境を図5のように構築した。本実験に際して撮影設備からの原メッシュのリアルタイム入力機能と実IP網への符号化メッシュのIP配信機能をエンコーダに具備した。また、伝送された符号化メッシュの受信・再生機能をスマートフォンに実装した。符号化メッシュを配信するIP網には商用網（インターネットと5G網）を使用した。本実験では、受信側でフレームドロップなく再構成メッシュが長時間連続再生されることを確認した[11]。この実験結果

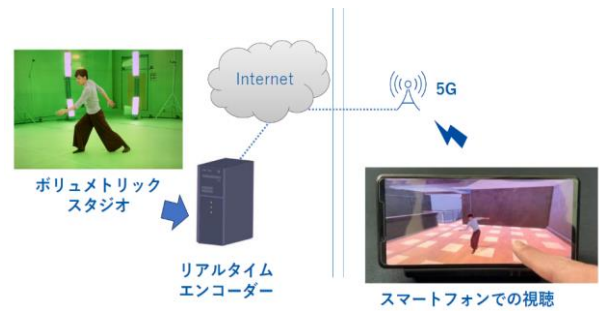


図5 実証実験環境

は、リアルタイムエンコーダを含めたV-DMCライブ配信システムが実際のユースケースにおいて十分に実用的であることを示している。

5. おわりに

本稿では、映像ベースの動的メッシュ符号化方式V-DMC対応のリアルタイムエンコーダを実現するV-DMC符号化アルゴリズムの簡素化手法と並列化手法を提案した。リアルタイムエンコーダはMPEGのテストモデルの再構成メッシュの主観品質を維持したまま符号化処理速度を向上させた。また、著者らは開発したエンコーダを用いて、世界ではじめて動的メッシュをV-DMCでライブ伝送するユースケースの実現可能性を立証した。今後の展望として、複数人物を含む動的メッシュのリアルタイムな符号化処理が実現できるアルゴリズムを検討したい。

謝辞

本研究成果の一部は、国立研究開発法人情報通信研究機構（NICT）の委託研究（JP012368C06801）により得られたものです。

参考文献

- [1] Z. Wenjie, et al. "Standardization Status of MPEG Video-based Dynamic Mesh Coding (V-DMC)." ICASSP 2025 IEEE International Conference on Acoustics, Speech and Signal Processing, 2025.
- [2] MPEG, "Text of ISO/IEC CD 23090-29 Video-based mesh coding." June 2024.
- [3] MPEG, "V-DMC codec description," July 2024.
- [4] G. Michael, and P. S. Heckbert. "Surface simplification using quadric error metrics." Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 1997.
- [5] R. Jarek. "Edgebreaker: Connectivity compression for triangle meshes." IEEE transactions on visualization and computer graphics, vol.5, no.1, p.47-61, 2002.
- [6] ITU-T, Recommendation "ITU-T H.265 | HEVC," April 2013.
- [7] 岸本広輝, 木谷佳隆, 河村圭. "V-DMCにおける高効率な非テクスチャ領域補間法に関する一検討," 画像符号化シンポジウム, 2024.
- [8] "NVIDIA Video Codec SDK", <https://developer.nvidia.com/video-codec-sdk> (2025) (accessed 2025.6.6)
- [9] 木谷佳隆, 海野恭平, 河村圭. "H.266 | VVC 対応 4K/8K リアルタイムコーデックの開発," 映像情報メディア学会誌, vol.78, no.1, p.115-123, 2024.
- [10] MPEG, "Common Test Conditions for V-DMC," April 2024.
- [11] KDDI総合研究所, "世界初、国際標準規格「V-DMC」に対応した3Dメッシュ映像のリアルタイム圧縮技術を開発," https://newsroom.kddi.com/news/detail/kddi_nr-421_3695.html (2025) (accessed 2025.6.6)