芸術科学会論文誌 Vol. 12, No. 2, pp. 82 – 91

分散軽量化手法の計算負荷を考慮した大規模ポリゴンモデルの領域分割手法

塩谷大樹† 松山克胤† 今野晃市† 徳山喜政††

†岩手大学大学院 工学研究科　††東京工芸大学 工学部

A New Large Scale Segmentation Algorithm for Decreasing Calculation Costs of Distributed Simplification

Hiroki Shioya † Katsutsugu Matsuyama † Kouichi Konno † Yoshimasa Tokuyama ††

† Graduate School of Engineering, Iwate University

†† Faculty of Engineering, Tokyo Polytechnic University

E-mail: shioya@lk.cis.iwate-u.ac.jp, {matsuyama, konno}@eecs.iwate-u.ac.jp, tokuyama@mega.t-kougei.ac.jp

概要

筆者らは，ＱＥＭ手法を高速に実行することができる，分散ＱＥＭ手法を提案している．分散ＱＥＭ手法は，データを分散させて，複数のＰＣで並列にＱＥＭ手法を適用することにより高速化を実現したものである．本研究は，既提案の分散ＱＥＭ手法の拡張を行うものであり，新たな領域分割手法の開発により，分散化効率の向上および適用可能なデータサイズの向上を行うものである．既提案手法では単純にサーバあたりのポリゴン数が均等になるように領域分割が行われているのに対し，本手法では，ポリゴンモデルの局所的な特徴が領域ごとに異なる事実に着目し，効率的な分散化のための領域分割を行う．また，既提案手法ではクライアントＰＣで領域分割を行っていたのに対し，各サーバで分散的にデータ読み込みを行うことで，より大規模なデータに対しても分散ＱＥＭ手法を適用可能にする．分散軽量化の処理時間は，領域分割処理と軽量化処理にかかる時間の合算である．本手法は従来の手法に比べて，領域分割処理にかかる時間は増大したが，軽量化処理にかかる時間については大幅な改善が行われ，結果として計算時間の短縮を行うことができた．

キーワード：領域分割, 分散ＱＥＭ手法, 並列処理, 大規模ポリゴン

1　はじめに

３次元計測機器から得られた３次元モデルを様々な場面で活用するとき，データ量の膨大な形状モデルを，実時間で処理することが求められている．３次元計測技術の進歩により，計測によって得られる３次元モデルのデータ量は，年を追うごとに巨大化し，形状モデリングや解析のための計算量，メモリ使用量も増加の一途をたどっている．

膨大な３次元モデルを１台のＰＣで扱うためには，多大なメモリを必要とする．しかし，通常の３２ビットＣＰＵを搭載するＰＣで扱える仮想メモリのメモリ空間は，最大で４ギガバイト程度であり，これが，１台のＰＣで扱える３次元モデルのデータ量の上限となる．したがって，１台のＰＣで膨大なデータ量のモデルを効率的に扱うためには，３次元モデルの軽量化技術が不可欠である．

軽量化手法の一つに，ＱＥＭ手法 [1] がある．ＱＥＭ手法は，頂点を縮退した後の詳細さと軽量化速度の間で最も良いバランスを持つアルゴリズムのひとつである．しかし，ＱＥＭ手法を，メモリ容量を超える大規模ポリゴンモデルに適用することがそもそも困難である．

筆者らは，ＱＥＭ手法を高速に実行することができる，分散ＱＥＭ手法を提案している [6]．これは，データを分散させて，複数のＰＣで並列にＱＥＭ手法を適用することにより高速化を実現したものである．本研究は，既提案の分散 QEM 手法 [6] の拡張を行うものであり，新たな領域分割手法の開発により，分散化効率の向上および適用可能なデータサイズの向上を行うものである．

2　関連研究

2.1　軽量化手法

ポリゴンモデルの軽量化手法はいくつか提案されている．なかでも，Garland と Hackbert が考案したＱＥＭ手法 [1] は，軽量化を実行した後の詳細さと軽量化スピー

ドの間で最も良いバランスを持つアルゴリズムの一つである．ＱＥＭ手法は，まず稜線の両端点を頂点ペアと定義し，全ての頂点ペアのコスト値を算出する．次に，最小コストの頂点ペアを削除し，１点に縮退した頂点で置換する．更に，置換された頂点のまわりで，コスト値を再計算する．以上のように，コスト値の算出と頂点ペアの削除を繰り返すことにより軽量化を行う手法である．図１にＱＥＭ手法の概念図を示す．頂点（V_1,V_2）の間の稜線を縮退することで，２つのポリゴンを削減できる．ＱＥＭ手法は，１ステップに１つの稜線しか削除しないため，処理速度が遅く，ＰＣクラスタによる分散化が難しい．

ＱＥＭ手法の拡張に関する研究は，時間効率を改良するものや，適応可能な対象を拡張するものなどがある．対象を拡張するものに，ポリゴンモデルの骨格に重みをつけ軽量化を行うことで，アニメーションに適した軽量化を行う手法 [2] が挙げられる．また，時間効率に関するものに，ポリゴンモデルをＢＳＰ木を用いて領域分割し，それぞれにＱＥＭを適用する手法 [3]，ボロノイ図によって稜線を単純化し，カメラの位置による重みをつけることで軽量化する手法 [4]，そして，頂点情報を画像情報に変換し，ＧＰＵを用いてリアルタイムに計算する手法 [5] が挙げられる．

しかし，文献 [2] から文献 [5] では，単一のプロセスで処理を行うため，１台のＰＣでは扱えないような，大きなサイズのポリゴンについての軽量化を行うことは難しい．本研究では，１台のＰＣで扱えないような大規模ポリゴンモデルを軽量化するため，既提案手法である分散ＱＥＭ手法 [6] を効率よく動作させるための大規模ポリゴンモデルの領域分割手法を提案する．まず，本研究で用いられる分散ＱＥＭ手法を次節で説明する．

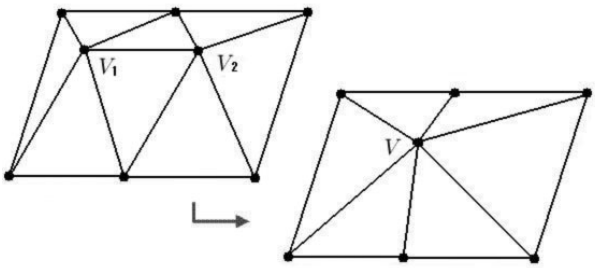

図 1: QEM の概念図

2.1.1 分散ＱＥＭ手法

文献 [6] による分散ＱＥＭ手法は，ＰＣクラスタを利用してポリゴンモデルの軽量化を行う手法である．ＱＥＭ手法では，モデルの局所的な幾何形状に基づいて各稜線のコストを計算するので，ポリゴンを各サーバに分散した場合でも各稜線のコストは分散前と変わらない．分散ＱＥＭ手法は，各サーバで計算したコストを集約し，ユーザが指定した削除割合から軽量化終了のための閾値を各サーバへ指示することで, 各サーバが並列に軽量化する手法である．これにより各サーバでは，ポリゴン数に依存せずに独立して軽量化を実行することが可能となる．

分散ＱＥＭ手法では，軽量化処理を２つのパスに分けて行う．１パス目の処理では領域分割で生じる境界線上の頂点を端点とする稜線を，削除対象外とみなして計算を行い，結果を統合化する．その後２パス目で，境界線上の頂点を端点とする稜線を含めた，全稜線を削除対象として軽量化を行う．

統合化とは，各サーバが行った１パス目の軽量化処理後のポリゴンデータを１つにまとめることである．１パス目の軽量化処理では，境界線上の頂点を端点とする稜線を削除対象外としているので，１つにまとめたときに各領域の連続性を保持したまま統合化することが可能である．統合化処理後，２パス目の処理として，クライアント１台で軽量化処理を行うことで領域間の連続性を保持したまま全体の軽量化を行うことが可能になる．図２はポリゴンモデルを軽量化した際の，１パス目の結果と２パス目の結果である．元データの色が同色の領域が各サーバに割り当てられる．

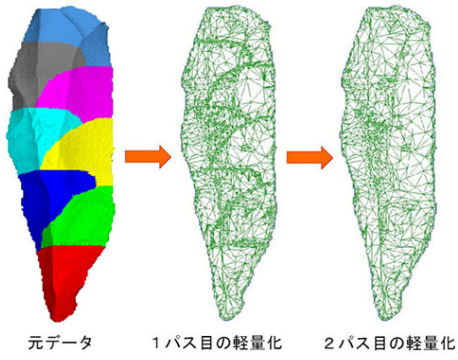

図 2: ２パスによる軽量化

文献 [6] の分散ＱＥＭ手法は，単純にサーバあたりのポリゴン数が均等になるように領域分割を行なっている．しかし，実際にはポリゴンモデルの局所的な特徴は領域ごとに異なるために，単純な分割では，効率的な分散化は行われない．また，文献 [6] では，クライアントＰＣで領域分割を計算しているために，適用可能なデータサイズがクライアントＰＣのメモリ容量に制限されている．分散ＱＥＭ手法を活用するためには効率的な領域分割手法の開発が必要である．

2.2 領域分割手法

本研究は分散ＱＥＭ手法の分散化効率を向上させるための新たな領域分割手法を開発するものである．ポリゴンモデルの領域分割手法はいくつか提案されている．以下に，代表的な領域分割手法を挙げる．

文献 [7] では，３次元モデルにＱＥＭ手法を適用して，軽量化したポリゴンモデルから元のモデルの特徴線を抽出し，特徴線に基づいて元のポリゴンモデルを領域分割する．

文献 [8] では，入力された３次元モデルを二つの領域に分割し，分割された領域についても再帰的に分割を行う手法が提案されている．分割を行うかどうかの判定は，測地距離と角距離を用いて行われている

文献 [9] では，３次元モデルを構成する各ポリゴンの中心から，モデルの内側に向かう線を放射し，他のポリゴンとの交差点と放射した点との距離に基づいて領域分割する．具体的には，一つのポリゴンから複数の線を放射し，得られた全ての距離の平均をＳＤＦ値（Signed Distance Function）とする．ＳＤＦ値を全てのポリゴンについて計算し，ＳＤＦ値が同じ領域をグループ化し，領域分割を行う．

文献 [7] から [9] の手法は，１台のＰＣ上で領域分割を行うものであり，大規模なポリゴンに適用できない．そこで本研究では，サーバへの負荷を効果的に分散させるために新たな領域分割手法を開発する．

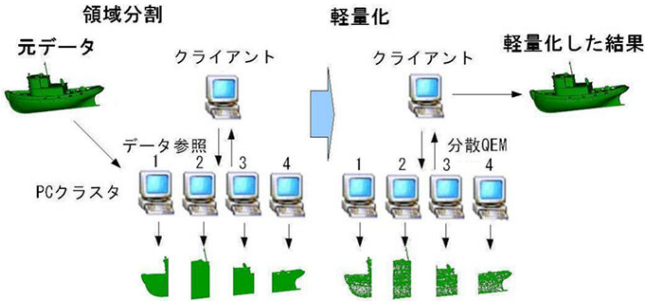

図 3: 分散軽量化システム

なお，手順１から６のうち，本論文で新たに提案するのは手順１から４である．以下の節では手順１から４について詳細を記述する．また，手順5,6 は文献 [6] を直接的に用いるものであるが，手順４と５の接続のために閾値の設定などの改良を行っている．本提案システムは領域分割のパートと軽量化のパートとに大別できる．領域分割は手順１から４で，軽量化は手順５から６で行うものである．このうち，手順２と手順５はサーバが独立に処理を行う部分であり，それ以外の手順はクライアントとサーバが協調して処理を行う．以下で処理の詳細を説明する．本研究では，クライアント・サーバ間の通信に関してはＤＣＯＭ [10] を利用する．ＤＣＯＭは Windows OS の標準機能のため，通信ソフトを新たにインストールする必要がないことが，利点である．

3 提案手法

本章では，分散ＱＥＭ [6] を含む軽量化システム全体の説明および，本論文で提案する領域分割手法について記述する．本研究の分散軽量化システムは，図３に示すように，入力されたポリゴンモデルを領域分割し，各サーバで分散ＱＥＭを実行する．分散ＱＥＭの実行終了後，クライアントでポリゴンを統合し，２パス目の軽量化を行う．本システムの処理手順は以下のようである．

1. ポリゴンモデル読み込み
2. 頂点のコスト値を計算
3. コスト値に基づいた頂点の割り当て
4. 分散ＱＥＭのための削除割合の計算
5. 各サーバで，与えられた閾値を削除割合として分散ＱＥＭ手法を実行
6. 分散ＱＥＭ手法による軽量化結果を統合化し，ユーザが指定した削除割合になるまで，クライアントでＱＥＭ手法を実行

3.1 ポリゴンモデルの読み込み

文献 [6] では，クライアントＰＣで領域分割を計算している．文献 [6] の領域分割は，入力ポリゴンデータを全てクライアントＰＣで読み込む必要があり，したがって，適用可能なデータサイズがクライアントＰＣのメモリ容量に制限されている．本手法では，各サーバが分散的にデータ読み込みを行い領域分割することによって，より大規模なデータに対しても分散ＱＥＭ手法を適用可能にする．

クライアントは，サーバ台数で頂点数を按分して，各サーバに読み込む大域的頂点ＩＤの範囲を決定する．また，サーバは指定された範囲の頂点のデータを読み込み，図４に示すように局所的頂点ＩＤを割り当てて保存する．例えばポリゴンの頂点数が１００個で，サーバ数が２台の場合，サーバ１に０から４９までの大域的ＩＤを読み込ませ，サーバ２には５０から９９までの大域的ＩＤを読み込ませる．各サーバは０から４９までの局所的ＩＤで頂点を管理する．

クライアントは，大域的頂点ＩＤから，指定するサーバと局所的頂点ＩＤのペアを算出する．これにより，どのサーバに必要な頂点データが入っているか把握するこ

とができるため，クライアントは，全ての頂点データを参照することができる．また，クライアントはＩＤのみをメモリに保存して，領域分割処理を行っているので使用するメモリが少なくて済む．

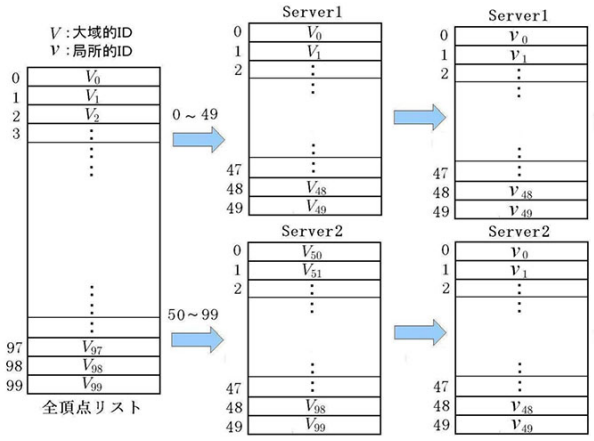

図 4: 頂点データ輪切りの概念図

3.2 頂点のコスト値計算

文献 [6] では単純にサーバあたりのポリゴン数が均等になるように領域分割が行われている．実際には，ポリゴンモデルの局所的な特徴が領域ごとに異なるために，ポリゴン数均等の分散化では，負荷分散は均等にならない場合が多い．本節では効率的な分散化のための，コスト値に基づく領域分割手法を説明する．ＱＥＭ手法 [1] では，各稜線に対してコスト値（QC（Qadric error metrics Cost）値）を算出して縮退する稜線を決定しているが，ここでは，各頂点に対するコスト値として，VC（Vertex Cost）値，および SC（Segmentation Cost）値を定義して，SC 値を用いて領域を計算する．各頂点の VC 値は，頂点を共有する各稜線の QC 値の平均である．VC 値に基づいて領域分割の指標に用いる値が SC 値である．図 5 はそれぞれの値の関係を示したものである．VC 値および SC 値の計算は式（1），式 (2) で行う．

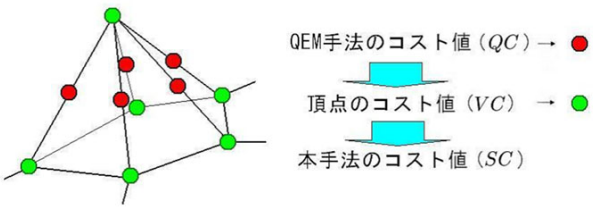

図 5: コスト値概念図

$$VC(v_i) = \frac{\sum_{E \in Edge(v_i)} \mathbf{v}_a{}^T Q \mathbf{v}_a}{E_{num}} \tag{1}$$

$$SC(v_i) = MaxCost - VC(v_i) \tag{2}$$

ここで，$E \in Edge(v_i)$ は，SC 値を求める頂点 v_i を共有している全ての稜線，E_{num} は頂点 v_i を共有している稜線の数を表す．また，$MaxCost$ は，$VC(v_i)$ の最大値を表し，$\mathbf{v}_a$ は E の中点の座標値である．また，Q は式 (3) に示すような４×４行列で，式 (4) の K_p の総和である．

$$Q = \sum_{P \in plane(v_i)} K_p \tag{3}$$

$$K_p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \tag{4}$$

式 (1) における Q は，VC 値を求める頂点 v_i の Q を Q_i，v_i を端点とする稜線のもう一方の頂点 v_j の Q を Q_j としたとき，Q=Q_i+Q_j で表される [1]．

ここで，$P \in plane(v_i)$ は，v_i に接触している全ての平面を表している．行列 K_p の要素は，頂点 v_i を共有するポリゴンがのる平面の方程式 (5) から得られる．ただし，$a^2 + b^2 + c^2 = 1$ とする．

$$ax + by + cz + d = 0 \tag{5}$$

ＱＥＭ手法では，最も QC 値が低い稜線を縮退させる．VC 値は QC 値に基づき算出した値であり，本研究では，低い VC 値を持つ頂点ほど，隣接する稜線が縮退される可能性が高いと考える．そして，低い VC 値を持つ頂点ほど，軽量化処理の計算対象となる確率が高く，計算負荷が高い頂点であるとみなす．式 (2) に示すように，SC 値は $MaxCost$ から VC 値を引いた値であり，高い SC 値を持つ頂点ほど計算負荷が高い頂点であると考える．

3.1 節で述べたように，各サーバに読み込んだ頂点データを元に，各サーバが独立に SC 値を計算する.SC 値を計算するために，割り当てられた頂点データ以外の頂点データが必要になる場合は，随時ポリゴンモデルから追加していく. 具体的な処理を図 6 を用いて説明する．図 6 は，すべての頂点のデータが 2 つのサーバに分割されて読み込まれている様子を示している．サーバ１に頂点 V_0,V_1 のデータが読み込まれ，サーバ２に頂点 V_2 のデー

タが読み込まれている．ここで，頂点 V_0 における SC 値を計算する場合に，頂点 V_1,V_2 が必要になるが，頂点 V_1 はサーバ１に存在するが，頂点 V_2 は，サーバ１に存在しない．このような場合には，頂点 V_2 のデータをサーバ１へ追加読み込みした後，頂点 V_0 における SC 値を計算する．上記の処理は，サーバで独立に計算できるため，少ない計算オーバーヘッドで頂点の SC 値を算出することが可能である．

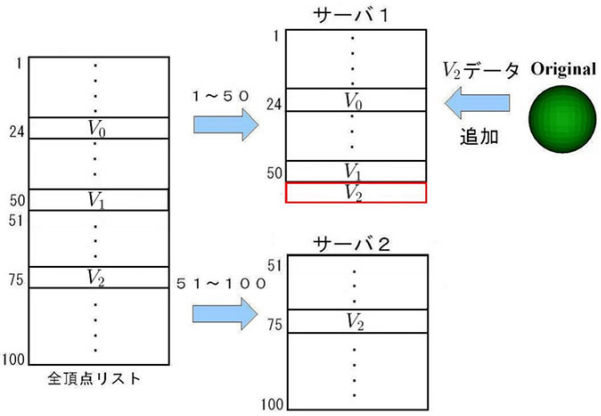

図 6: 頂点追加の例

3.3 コスト値に基づいた頂点の割り当て

3.2 節で説明した方法で算出した SC 値を用いて，削除される可能性の高い頂点を端点とする稜線数を均一にすることで，分散ＱＥＭ手法実行時の各サーバ負荷を均一にするように，各領域に頂点を割り当てる．具体的な手順を以下に記述する．

1. 各サーバで計算された，頂点の SC 値をクライアントに渡す．
2. クライアントは，集めた SC 値を降順でソートする．
3. ソート後，Floor(N*r) 番目の SC 値を，基準 SC 値として保持する．ここで，N はモデルの頂点数，r はユーザが指定した削除割合である．本手法では，基準 SC 値を，稜線が削除されるかどうかを分つ擬似的な閾値と考える．
4. 入力モデルのバウンディングボックスを生成する．生成されたバウンディングボックスの各辺の長さを比較して，最大長さとなる辺の座標軸を求める．
5. 最大長さとなる辺の座標軸上で，座標値が最小となる頂点を探索し，領域分割の開始点とする．
6. 開始点から，座標軸の正方向に昇順に探索を行い，領域の拡張を行う．領域ごとに変数 QV と QL を定義し，初期値を 0 とする．QV は，削除される可能性の高い頂点数を表す．各頂点の SC 値が基準 SC 値以上の場合，QV に 1 を加算し，その頂点を共有している稜線の数を QL に加算する．ただし，各頂点の SC 値が基準 SC 値未満のときは，その頂点はカウントしない．
7. QL 値が式 (6) を満たす場合に，領域の拡張を終了する．ただし，AL は式 (7) で表され，基準 SC 値以上の SC 値を持つ頂点を共有している稜線数である．

$$QL > \frac{AL}{\text{領域数}} \tag{6}$$

$$AL = \sum QL \tag{7}$$

手順 4 で，最も長い座標軸を求めた理由は，境界線上の頂点数をできるだけ少なくし，軽量化２パス目の入力となるポリゴンモデルのデータ量を抑えるためである．手順 6 で算出した QL は，基準 SC 値以上の頂点を共有する稜線，すなわち，削除される確率の高い稜線の数を擬似的に計算したものであり，QL を均等にすることで負荷分散の均等化を図るものである．

領域を 2 分割した例を図 7 に示す．図 7 では x 軸が最大長さとなる辺の座標軸であり，x 軸上の座標値が最小である頂点が開始点となる．図 8 は，あるポリゴンを４分割した例である．

3.4 分散ＱＥＭのための削除割合の計算

3.3 節では，分散負荷を均等にするための，各サーバの頂点の割り当てについて述べた．分散ＱＥＭ手法 [6] では，ＱＥＭ処理を独立に行うが，各サーバに対し削除割合を指定する必要がある．本研究では，3.3 節の手法で計算した QV を，各サーバで削除する頂点数とみなし，削除割合を算出した後，分散ＱＥＭ手法 [6] を適用して軽量化を行う．図 8 の V は各サーバが持つ頂点数，F はポリゴン数，QV は削除する頂点数であり，削除割合は QV/V である．

4 実験結果

提案システムをＰＣクラスタ上に実装し，領域分割および軽量化を実行する．図 9 に示すように，ＰＣクラスタは，8 台のＰＣをギガビットネットワークで接続したものを使用する．各ＰＣは，ＣＰＵ: Pentium4 3.0GHz, メモリ: 1.0GB を搭載している．サーバを 8 台使用す

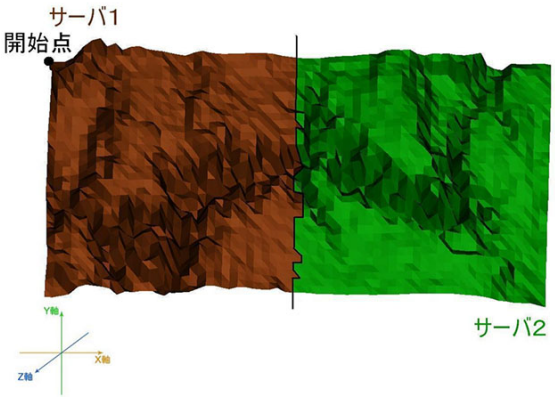

図 7: 領域 2 分割

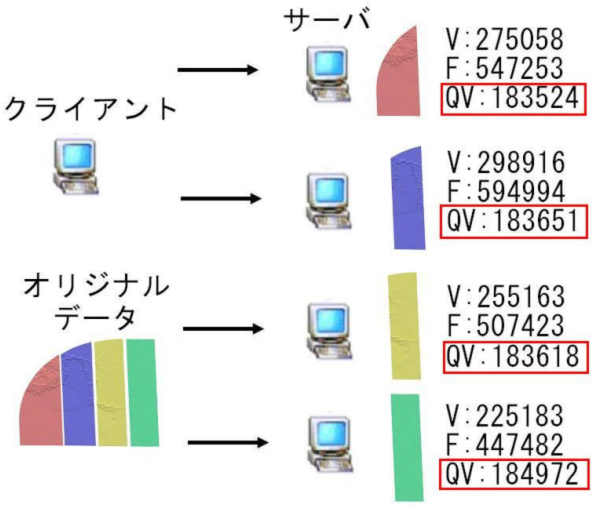

図 8: 領域 4 分割

るので，以下の実験では，入力ポリゴンモデルを 8 分割する．

まず，軽量化の処理時間を計測する．本手法の評価のために，頂点数を各サーバで均等になるように分割する「頂点数均等法」と，本提案システムの処理時間を計測する．頂点数均等法は，文献 [6] の領域分割処理を各サーバが行うことに相当する．実験に使用したモデルを表 1 に，実験に使用した一部のモデルの軽量化前のデータを図 10 に，実験に使用した一部のモデルの軽量化後のデータを図 11 に示す．

表 2 は，2 つのモデル Land および GCleft についての領域分割および軽量化の処理時間を示す．システム全体の処理時間は，領域分割処理と軽量化処理に要した時間の合算である．Land の計算時間がわずかな改善であったのに対し，GCleft では，領域分割にかかる時間は増大したが，軽量化処理にかかる時間が大幅に改善されており，結果として計算時間の短縮を行うことができた．

表 3 はモデル Land の，表 4 はモデル GCleft の，サーバごとの軽量化実行時間を示す．表中の赤字は最大の実行時間，すなわち，軽量化処理の実行時間である．表 3 では最大の実行時間がわずかに改善されている．表 4 の GCleft では分散の偏りが均等化され，2.6 倍程度の高速化を行うことができた．表 3 と表 4 は，どちらも計算負荷となる領域がサーバ 7 に局在している．頂点数均等法と比較して，本手法では計算負荷の分散を行うことができたといえる．

図 9: PC クラスタ

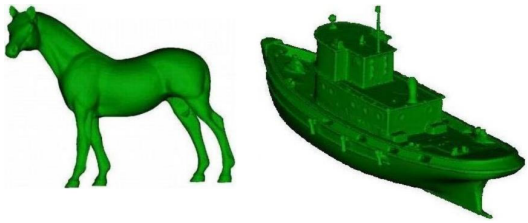

図 10: 実験に使用したモデル:軽量化前（一部）

表 5 は，サーバ台数を変更した場合の領域分割および軽量化の実行時間であり，図 12 と図 13 はそれぞれ，モデル Boat および Land の実行時間のグラフである．サーバ台数が増加するに従って処理時間が減少している．モデル Land の分割結果を図 14 に，軽量化結果を図 15，図 16，図 17 に示す．図 15，図 16 および図 17 は，削除割合が異なるものである．データは地形を表すものである．凹凸の変化量が多い部分を保つように軽量化が行われていることが確認できる．

5 検討

5.1 実行時間

実験結果から，本提案システムは頂点数均等法と比較して，計算時間の短縮をおこなうことができたといえる．これは，負荷分散を均等化することで，分散化効率の向上ができたことを示している．頂点数均等法に比べ，本

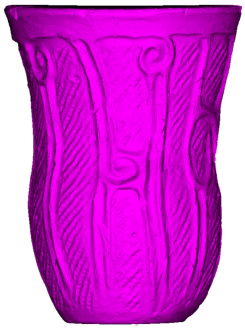

図 11: 実験に使用したモデル:軽量化後（一部）

手法の領域分割の実行時間は増大している．これは，領域の分割にコスト値の計算が含まれているためである．領域分割に比べて軽量化の計算の方が，全体の計算に占める割合が多く，軽量化の高速化を図る本手法のアプローチは有効なものであるといえる．

図 12 と図 13 はそれぞれ，モデル Boat および Land の実行時間のグラフである．それぞれ，領域分割および軽量化に要した時間を示しており，各サーバの実行時間の最小値を青，最大値を赤で示している．図 12 のモデル Boat の場合は最大値と最小値との差が少なく，負荷分散が効果的に行えていることを示している．一方，図 13 のモデル Land の場合，サーバ台数が増えるにしたがい，最大値と最小値との差がみられ，負荷分散についてはまだ改善の余地があるといえる．

5.2 データサイズ

本提案システムでは，各サーバのメモリ空間の合計が上限となる．なお，クライアントＰＣが保持するデータはポリゴンＩＤと SC 値のリストであり，このリストもメモリ空間内に収まる必要があるが，リストのデータ量が小さいために問題とはならない．

表 5 の「×」は，メモリが不足していたために実行ができなかったことを示している．モデル GCleft および GCright は，モデル GrandCanyon を 2 つに分割したものである．GCleft および GCright の軽量化はサーバを 8 台使用することにより実行可能となる．また，GrandCanyon はサーバ 8 台では軽量化を行うことができなかったが，GCleft および GCright の軽量化結果を統合することにより，間接的な軽量化を行うことができた．モデル GrandCanyon の間接的な軽量化の結果を図 18，図 19 に示す．GrandCanyon は地形を表すデータである．GCleft と GCright はそれぞれ独立して処理が行われており，分割の境界部分については軽量化が行われていない．

表 1 と表 5 から，モデル Land（頂点数 100 万，ポリゴン数 200 万程度）は 2 台のサーバで，また，モデル GCleft と GCright（それぞれ，頂点数 400 万，ポリゴン数 800 万程度）は 8 台のサーバで，領域分割および軽量化処理を行うことができることがわかる．これらモデルの頂点数とポリゴン数をサーバ台数で割ることによって，サーバ 1 台あたり，頂点数 50 万，ポリゴン数 100 万程度であれば計算可能であると概算できる．また，本論文で使用したサーバ PC は，1.0GB のメモリを搭載している．メモリを 2 倍にすると，扱える頂点数とポリゴン数も 2 倍になると仮定すると，計算可能な頂点数とポリゴン数は，それぞれ，50 万× MS，100 万× MS と見積もることができる．ここで，MS は，メモリ容量 [GB] × サーバ台数である．例えば，8.0GB のメモリを搭載しているサーバを 16 台使用した場合，計算可能なポリゴン数は，1.28 億程度であると見積もることができる．

モデル Doki は，GCleft や GCright と比べて少ないポリゴン数のモデルであるが，2 台および 4 台のサーバ環境では軽量化に失敗し，また，8 台のサーバ環境でも効率的な計算を行えていない．これは，モデル Doki に，共有している稜線数が極端に多い頂点が含まれていることが原因であると考えられる．このようなモデルは本手法のコスト値の計算に大きな影響を与える可能性があり，領域分割の偏りを発生させる要因となる．結果的に，負荷の分散化効率の劣化や，計算の失敗などが発生したと考えられる．

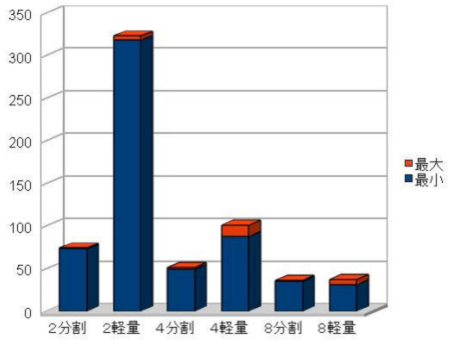

図 12: Boat:実験結果（秒）

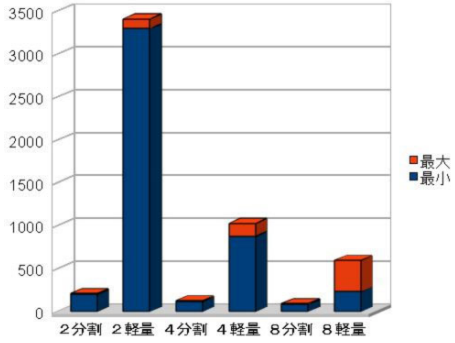

図 13: Land:実験結果（秒）

表 1: 実験に使用したポリゴンモデル

モデル	頂点数	ポリゴン数
Horse	48,485	96,966
Boat	388,164	776,368
Land	1,051,093	2,097,152
Doki	2,277,128	4,554,240
GCleft	4,245,060	8,481,840
GCright	4,151,782	8,295,376
Grand Canyon	8,394,753	16,777,216

表 2: 領域分割全体の実行時間（秒）

モデル	手法	分割	軽量	全体	差
Land	頂点数均等法	145	625	770	-26
	本手法	143	599	742	
GCleft	頂点数均等法	645	8,536	9,181	-4,950
	本手法	946	3,285	4231	

表 3: Land の軽量化実行時間（秒）

Land	サーバ 1	サーバ 2	サーバ 3	サーバ 4	サーバ 5	サーバ 6	サーバ 7	サーバ 8
頂点数均等法	236	239	234	223	245	288	625	330
本手法	258	254	235	246	236	291	599	308

表 4: GCleft の軽量化実行時間（秒）

GCleft	サーバ 1	サーバ 2	サーバ 3	サーバ 4	サーバ 5	サーバ 6	サーバ 7	サーバ 8
頂点数均等法	2,366	2,556	2,561	2,695	2,547	2,946	8,536	3,927
本手法	2,421	3,108	3,038	3,088	2,883	2,928	3,285	2,955

表 5: 実験まとめ:クラスタの実行時間最小/最大（秒）

ポリゴンモデル	サーバ 2 台		サーバ 4 台		サーバ 8 台	
	領域分割	軽量化	領域分割	軽量化	領域分割	軽量化
Horse	9/9	11/11	6/6	4/5	4/4	2/3
Boat	73/74	318/323	49/51	88/101	35/36	31/37
Land	201/211	3291/3406	121/129	879/1028	85/93	235/599
Doki	×	×	×	×	1262/1860	780/917
GCleft	1072/1297	×	559/603	×	421/437	2421/3285
GCright	938/942	×	517/539	×	362/393	2729/4686
GrandCanyon	×	×	×	×	755/809	×

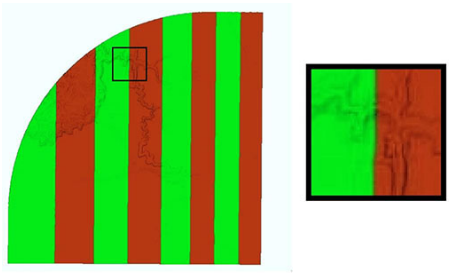

図 14: Land:分割結果

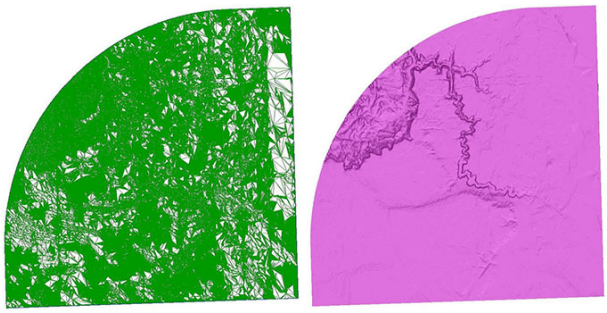

図 15: Land:1 パス７０%軽量化結果．頂点数: 331,092, ポリゴン数: 630,380.

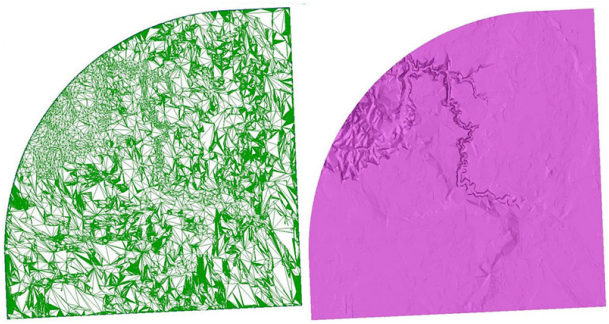

図 16: Land:2 パス９０%軽量化結果．頂点数: 51,092, ポリゴン数: 63,984.

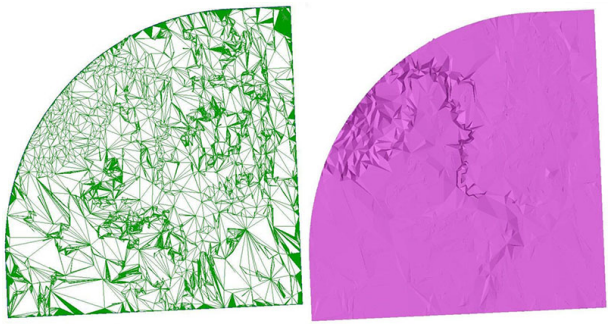

図 17: Land:2 パス９７%軽量化結果．頂点数: 31,092, ポリゴン数: 22,890.

6　まとめ

既提案の分散 QEM 手法に対し，新たに領域分割手法を開発し，分散化効率の向上を行った．既提案手法では単純にサーバあたりのポリゴン数が均等になるように領域分割が行われているのに対し，ポリゴンモデルの局所的な特徴が領域ごとに異なる事実に着目し，効率的な分散化のための領域分割を行った．本手法は従来の手法に比べて，領域分割処理にかかる時間は増大しているが，軽量化処理の大幅な改善が行われており，結果として計算時間の短縮を行うことができた．今後，効果的なコスト値計算手法の検討を行うなどの，負荷のさらなる均等化を目指して研究を推進する．

参考文献

[1] Michael Garland and Paul S.Heckbert: “ Surface Simplification Using Quadric Error Metrics ”, Proc. SIGGRAPH ’97, pp.209-216, 1997.

[2] Eric Landreneau,Scitt Schacfer: ”Simplification of Articulated Meshes”, EUROGRAPHICS 2009.

[3] Eric Shaffer, Michael Garland: “ Efficient Adaptive Simplification of Massive Meshes ”,VIS’01 proceedings of the conference on Visualization ’01, 2001.

[4] Ling Yang,Liqiang Zhang,Jingtao Ma,Zhizhong Kang,Lixin Zhang,Jonathan Li:“ Efficient Simplification of Large Vector Maps Rendered onto 3D Landscapes ”, IEEE Computer Society March/April 2011 ,Vol. 31 No. 2,pp. 14-23.

[5] Christopher DeCoro, Natalya Tatarchuk: “ Real-time Mesh Simplification Using the GPU ”,Proc. of the 2007 Symposium on Interactive 3D Graphics and Games, April 2007.

[6] 吉田安男，今野晃市，徳山喜政: “ PC クラスタ環境のための 3 次元モデルの軽量化手法の分散化 ”, 芸術科学会論文誌, Vol.7, No.3, pp.113–123, 2008.

[7] Zhang Liyan, Liu Shenglan, Wu Xi, Zhou Laishui: “ Segmentation and Parametrization of Arbitrary Polygon Meshes ”, GMP2004, pp.143-152, 2004.

[8] Sagi Katz and Ayellet Tal: “ Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts ”, Proc.SIGGRAPH 2003, pp.954-961, 2003.

[9] Lior Shapira, Ariel Shamir, Daniel Cohen-Or: “ Consistent Mesh Partitioning and Skeletonisation

using the Shape Diameter Function ”, The Visual Computer, Vol.24, No.04, pp.249-259, 2008.

[10] Frank E: “ Redmond III, Dcom： Microsoft Distributed Component Object Model, IDG Books Worldwide, Inc., 1998.

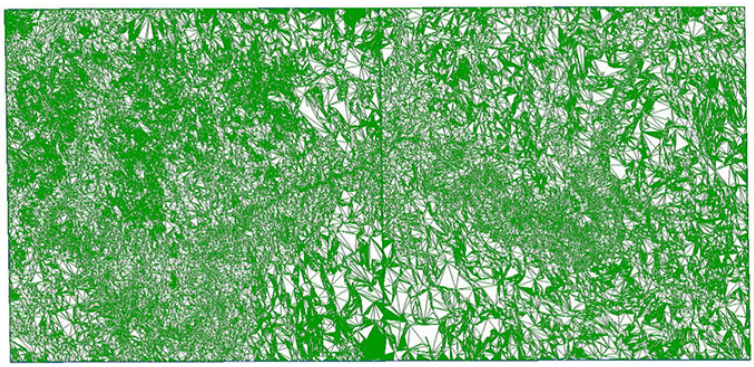

図 18: Grand Canyon 軽量化：頂点と稜線表示

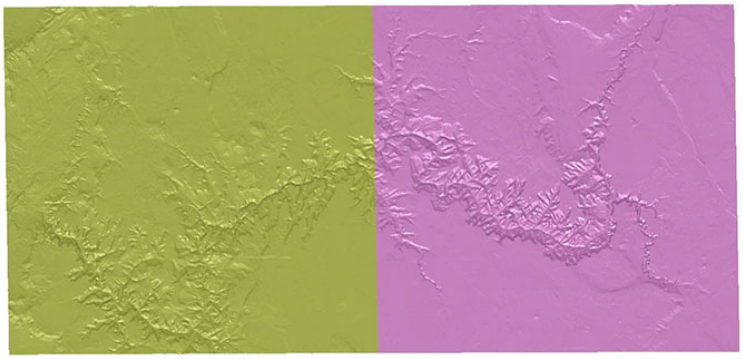

図 19: Grand Canyon 軽量化：面表示