Vol.2017-CG-168 No.37
Vol.2017-DCC-17 No.37
Vol.2017-CVIM-209 No.37
2017/11/9

複数の RGB-D カメラによるリアルタイムモデリング

松本依里紗 [1, a)]　藤田悟 [2]　廣津登志夫 [2]

概要：近年の VR への関心の高まりにより，現実世界の物体のモデル化の需要が高まりつつある．現在，VR スタジオなどの体験型施設では，仮想的に作られた三次元空間に没入して移動するようなアプリケーションは多く見られるが，将来的には実空間の物体を仮想空間に再現して遠隔のコミュニケーションを支援するようなアプリケーションへの拡張が期待される．この際には対象となる動く物体を全方向からスキャニングした三次元モデルが必要となってくる．しかし，従来の手法では RGB-D カメラを動かしてモデルを生成するため，動体のモデル化ができなかった．また，多方向の RGB-D カメラから物体の表面情報を使ってモデルを生成する手法では，撮影のノイズによる欠落を回避することが困難であった．そこで，本研究では異なる視点の複数の RGB-D カメラから得られた深度情報をボリュームデータとして統合し，リアルタイムに欠損の少ない三次元モデルの生成する手法を提案する．ここで生成したモデルを連続して投影することにより，現実世界の動きを仮想空間でリアルタイムに表現することが可能となる．

1. はじめに

近年，Head Mounted Display(HMD)の普及に伴い VR コンテンツの需要が高まりつつある．現状，VR スタジオなどの体験型施設で提供されているコンテンツの多くは，コンピュータにより作られたサイバースペースを CG により描画したものとなっている．そのようなサイバースペース中に多数の現実感のある物体を配置するには，映画やゲームの CG でもよく使われているような実世界の物体をベースにした三次元モデルの生成技術により，より多くのリアルな三次元モデルを生成することが必要となる．また，将来的には，実世界の物体の三次元モデルをリアルタイムに取得しサイバースペースに投影することで，VR 技術に支援された遠隔コミュニケーションを提供するといった，より実生活に溶け込んだアプリケーションへ展開することを考えると，容易に広く普及させることができる三次元モデル生成技術が必要になる．

Microsoft が発売した RGB-D カメラの Kinect は安価に奥行き方向を含めた実空間の情報を取得できることから，多くの研究に用いられている．この Kinect を用いた Kinect Fusion では，Kinect を手で持ちながらリアルタイムにスキャン状況を確認できることから，専門知識のない一般ユーザでも容易に三次元スキャンを行うことができる．しかし，1 つの物体のスキャンに時間がかかるため，対象が静止物体に限られるという問題がある．したがって，人間のような動体のスキャニングを行う際に僅かでも動いてしまうと三次元モデル生成が失敗してしまう．

本研究では，複数の RGB-D カメラを用い対象物を撮影し，GPGPU を用いた高速データ処理により，リアルタイムに三次元モデル生成を行う．本論文の構成は以下の通りとなる．第 2 章では，本研究のベースとなる研究についてのべ，第 3 章で従来手法の再現実験の結果から問題点を示す．第 4 章では本手法のシステム構成と処理アルゴリズムについて述べる．第 5 章では本手法による速度測定を行った結果を示す．

2. 既存手法

Izadi ら[1][2]は，リアルタイムに三次元形状の復元を行う手法として Kinect Fusion を提案した．RGB-D カメラである Microsoft の Kinect と並列演算において性能を発揮する GPGPU 技術を組み合わせ，Kinect で取得した深度マップから，物体の三次元形状を取得する手法である．物体の全周を撮影するように Kinect を連続的に動かし様々な角度，位置の深度マップを得る．そして，空間を均一なグリッド上のボクセルボリュームと考えて，得られた深度マップの情報からボクセルデータを更新していく．ボクセルデータには近傍にある物体面との距離を多値で表現した符号付距離場が格納され，最後に Ray marching 法をボクセルボリュームに対して行い，表面推定をする．しかし，この手法では 1 つの三次元形状を復元するために手動での全周スキャンが必要であり，動体のスキャニングはできない．

Alexiadis ら[3]は複数台の RGD-D カメラを 4 方向に配置し，動体の三次元モデルをリアルタイムに生成する手法を提案している．ここでは，1 つの深度マップから 1 枚のメッシュを生成し，それぞれのメッシュを重ね合わせた後，重複部分のメッシュを除去，隣接する三角形を検出し，結合することで 1 つの三次元モデルの生成を行っている．この手法は，RGB-D カメラから取得した物体の表層の面的な情報だけを使用している．そのため，RGB-D カメラのノイズなどによって生じた欠落が微少であれば補間により補正することができるが，隣接する三角形が一定の範囲にないような大きな欠落については補正することができない．

そこで本研究では Kinect Fusion と同様の，ボリュームボ

1 法政大学大学院情報科学研究科
2 法政大学情報科学部
a) 16t0019@cis.k.hosei.ac.jp

Vol.2017-CG-168 No.37
Vol.2017-DCC-17 No.37
Vol.2017-CVIM-209 No.37
2017/11/9

クセルを用いた複数 Kinect による全周スキャンによって，欠落のない動体のリアルタイム三次元モデル生成を行う．

図 2　表面モデルの重ね合わせによって生じた欠落

3. 予備実験

まず，予備実験として Alexiadis らの手法を拡張して三次元モデルの取得を行った．ここでは，円周上に等間隔に配置する Kinect を 4 台から 6 台に増やし，より高品質の三次元モデルの取得を目指した．実験の結果を図 1 に示す．この実験では，Kinect の中心で動いている人物のスキャンをリアルタイムに行っている．Alexiadis らの手法と同様に，それぞれの Kinect から得た深度マップで表面モデルを生成し，それらを複数重ね合わせることで 1 つの三次元モデルとした．図 1 のように大きな欠落が生じることがあるが，このような欠落は Alexiadis らの手法でも補間することはできない．また，6 台の Kinect を用いると処理速度が 3.3fps 程度となり，リアルタイムの処理には及ばない．

4. システム構成

本研究では，異なる位置，角度から同時にスキャンを行うため，複数の Kinect を用いる．Kinect から得られるデータは 1920*1080 の RGB データと 512*424 の深度データであり，今回は表面形状推定に深度データを用いる．また，本研究で用いる Kinect V2 では単一の PC で複数台を扱うことができない．そこで，各 Kinect に PC を接続し，Kinect から得られた深度マップをサーバに送信することで複数台の Kinect を同時に扱えるようにする．また，サーバと PC 間はイーサネットを用いて接続，プロトコルはデータの欠落を避けるために TCP を用いる．図 2 のように，Kinect と接続した PC は深度マップを取得し，それをサーバに送信する．サーバでは，各 Kinect で取得された深度マップを統合し，GPGPU を用いてリアルタイムに三次元表面形状推定を行う．Kinect の台数を増やすと，より精緻なモデルが生成できると考えられるが，今回の実装環境では取得する深

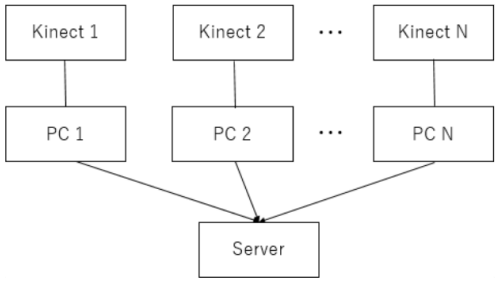

図 1 システムの環境

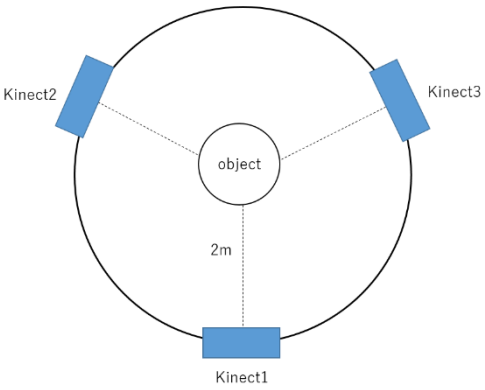

図 3　システム概要

度マップに重なりが生じる最小の構成として，図 3 のように半径 2m の円周上に等間隔に Kinect を配置し，円の中心に被写体を配置する形をとった．

まず，事前処理としてカメラのキャリブレーションを行う．次に各 Kinect から深度マップを取得，送信する．サーバでは，すべての深度マップの情報からボクセルデータの重みを更新，ボクセルデータを用いて Marching Cubes 法により表面形状推定を行う．また，ボクセルデータを Octree 構造で保存することによりデータのない不要な空間の形状推定を省くことができる．本手法による出力結果が図 4 となる．描画は OpenGL を用いる．以降で各手法について説明する．

4.1 位置合わせ

本研究では複数の RGB-D カメラを用いる為，キャリブレーションが必要となる．リアルタイム性を重視している為，事前処理としてキャリブレーションを行う．ます，適度に深度差のある剛体を，撮影空間である円内の中央に配置する．この物体を異なる位置，角度で設置した Kinect で撮影し，深度マップを取得する．すべての深度マップが矛盾なく元の剛体を再現できるように重ね合わせることができれば，個々のカメラの位置を推定できたと言える．

実際には複数の 3 つ以上の深度マップを同時に調整することは困難であるため，隣接する 2 つの深度マップで順に位置合わせを行う．この際に，ICP(Iterative Closest Point)ア

Vol.2017-CG-168 No.37
Vol.2017-DCC-17 No.37
Vol.2017-CVIM-209 No.37
2017/11/9

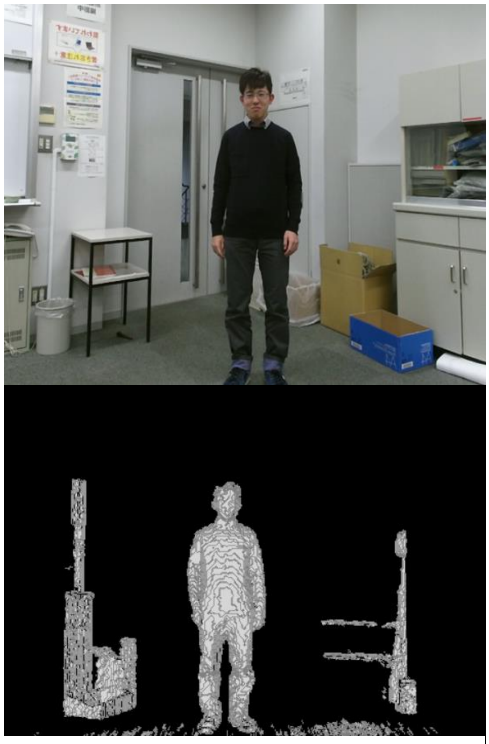

図 4　システムの出力結果

ルゴリズム[4]を用いる．対応関係が未知な 2 つの点群をマッチングさせ，反復的に比較していくことにより，位置合わせを行う手法である．しかし，一周分の位置合わせを終えた時に，最初と最後の深度マップが正しく重ね合わせることができない．これは，1 つずつの位置合わせの誤差が積算されるためである．そこで，2 台間の位置合わせの後，全てのカメラ位置を微調整する．

Kinect をN台用いるとき，隣接するカメラ座標への変換行列をP$_n$とすると，$P_{1,\cdots,}P_N$の変換行列を掛け合わせると理論上は単位行列になるが，誤差の積算により差異が生じる．積算誤差行列はすべての変換行列の積の逆行列で表すことができる．その誤差行列の回転移動の角度を求め，N 分割し，全ての変換行列に掛けることで積算誤差の補間を行う．平行移動部分は要素を N 分割する．誤差行列の移動方向は基準となったカメラ座標系となるため，全ての変換行列に同じ誤差行列を用いることはできない．そこで，それぞれの Kinect を起点として周回の積算誤差を算出し，それらの値を用いてそれぞれの補間行列を求める．n 番目の Kinect を起点とした N 個の変換行列の積の逆行列M_nは式(1)となる．

$$M_n = (P_n P_{n+1} \cdots P_N P_1 \cdots P_{n-1})^{-1} \quad (1)$$

X 軸回りの回転行列を$\boldsymbol{R}_x(\theta_n)$，Y 軸回りの回転行列を$\boldsymbol{R}_y(\varphi_n)$，Z 軸回りの回転行列を$\boldsymbol{R}_z(\omega_n)$とする．式(2)より$\theta_n$，$\varphi_n$，$\omega_n$を求め，さらに式(3)より積算誤差補間行列$C_n$を求める．

$$\boldsymbol{R}_x(\theta_n)\boldsymbol{R}_y(\varphi_n)\boldsymbol{R}_z(\omega_n) = \begin{pmatrix} M_{n_{11}} & M_{n_{12}} & M_{n_{13}} \\ M_{n_{21}} & M_{n_{22}} & M_{n_{23}} \\ M_{n_{31}} & M_{n_{32}} & M_{n_{33}} \end{pmatrix} \quad (2)$$

$$C_n = \boldsymbol{R}_x\left(\frac{\theta_n}{N}\right)\boldsymbol{R}_y\left(\frac{\varphi_n}{N}\right)\boldsymbol{R}_z\left(\frac{\omega_n}{N}\right)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{M_{n_{41}}}{N} & \frac{M_{n_{42}}}{N} & \frac{M_{n_{43}}}{N} & 1 \end{pmatrix} \quad (3)$$

隣接する Kinect から n 番目の Kinect への新たな変換行列$P_n^{'}$を式(4)より求める．

$$P_n^{'} = P_n C_n \quad (4)$$

$P_n^{'}$を用いることで各カメラ座標系からグローバル座標系への変換が可能となる．

4.2 ボクセルデータの更新

描画する空間を均一なボクセルボリュームとし，各 Kinect から取得した深度マップを用いてボクセルデータの値の更新を行う．初期値はすべて 0 である．値の更新には符号付距離場を用いる．カメラ視点から各深度座標までの距離をd_p，ボクセルまでの距離をd_vとし，その距離の差をボクセルデータに与え，更新する．ボクセルデータの値は，生成する表面モデルまでの距離を表しており，値が小さいほど表面に近い．また，符号はボクセルが表面の内側と外側どちらに含まれるかを表しており，符号の境界に表面が形成される．値を更新するボクセルは図 5 のようにカメラ視線上にあるボクセルのみであり，深度座標から一定の範囲で打ち切ったものである．その為，表面から離れている何もない空間については，0 のまま変化しない．

同時刻に取得したすべての Kinect から得た深度マップでボクセルデータの値を決定する．ボクセルデータの値は次のフレームには引き継がずにリセットすることで，物体の動作に影響されず毎フレームごとに異なるモデルが生成される．

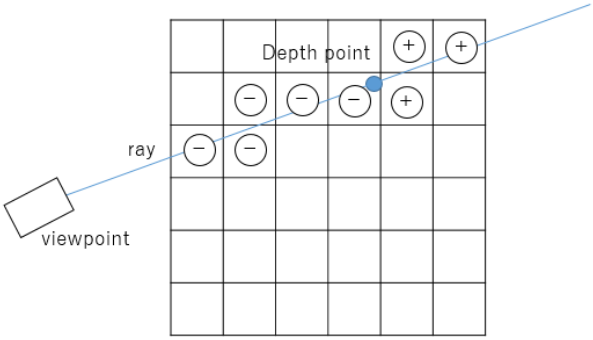

図 5　符号付距離場によるボクセルデータの更新

Vol.2017-CG-168 No.37
Vol.2017-DCC-17 No.37
Vol.2017-CVIM-209 No.37
2017/11/9

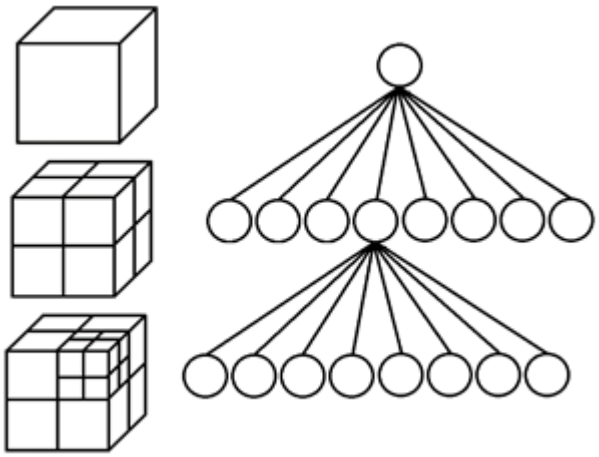

図 6　Octree 構造

4.3 表面形状の推定

4.2 節で求めたボクセルデータの値から表面モデルを構成するポリゴンを求める．たがいに隣接する 8 つのボクセルデータについて考えていき，これらの 8 つの値の符号から Marching Cubes 法[4]によってポリゴン群を割り当てる．Marching Cubes 法は 8 つの頂点の符号から三角形ポリゴン群を割り当てる．割り当てられる三角形のパターンは 256 通りであり，対称性を考慮すると 15 通りとなる．三角形の頂点は，頂点間の辺上に配置されるが，ボクセルデータの値により，三角形の頂点を変動させることで滑らかなモデルとなる．

4.4 Octree を用いた高速化

4.3 節では Marching Cubes によりポリゴン群を求めたが，ボクセル数が膨大なため計算コストがかかる．例えば，ボクセルボリュームのグリッド 1 辺の大きさが 512 の場合，1 億個以上のノードにアクセスする必要があり，リアルタイムでの処理は困難である．しかし，すべてのボクセルが面を持つのではなく，何もない空間，物体の内部でありポリゴンを描画する必要のない空間が存在する．それらのボクセルに対して処理を行うことは不要であるため，Octree 構造を用いて削減を行う(図 6). Octree は子を 8 個持つ 8 分木の木構造であり，X・Y・Z 軸のそれぞれの方向に 2 分割されている．何もない空間については浅い木の高さ，情報量が多い空間については深い木の高さとなる．

5. 実験

提案手法の評価のために，三次元モデルの描画速度の測定を行う．描画する物体の大きさにより Octree によるボクセル削減の効果が異なってくる．また，より精度の高いモデルを生成するには多くのカメラから様々な角度でスキャンする必要がある．しかし，3 章で述べたように複数の

図 7　実験に用いた 3 種類の被写体

Kinect V2 を同時に利用するには取得したデータの送信処理が必要となる. そこで, 大きさが異なる被写体での実験, 接続台数を変えての実験の 2 通りを行う．接続台数の実験では，半径 2m の円周上に等間隔に 3 台の Kinect を設置して行う．被写体は円の中心に配置する．使用するハードウェアは CPU が Intel Core i7-4710MQ 2.5GHz, GPU が NVIDIA GeForce GTX 850M である．また，ボリュームボクセルの 1 辺の大きさは 256 とする．安定した結果を得るため，評価においては，被写体はすべて静止物としたが，処理自体は各フレームに対して行った．

表 1　被写体によるフレームレート

	ポリゴン数	フレームレート [fps]
物体 1	6,238,944	12.4
物体 2	8,155,596	11.6
物体 3	10,637,053	9.8

表 2　接続台数によるフレームレート

接続台数	フレームレート[fps]
1 台	12.4
2 台	7.0
3 台	5.8

5.1 描画面積の違いによる速度測定

図 7 の丸椅子，椅子の上に置いたぬいぐるみ，ホワイトボードの 3 種類の大きさが異なる物体の三次元モデル生成を行い，速度比較を行った．以降はそれぞれ物体 1，物体 2，物体 3 と呼称する．表 1 の結果のように，ポリゴン数の増加に伴い描画速度であるフレームレートは低下した．物体 1 と物体 3 ではポリゴン数に 1.7 倍の差があるが，フレームレートの低下率は 0.8 であった．

5.2 接続台数による速度測定

使用する Kinect の台数を 1 台~3 台として，フレームレート測定を行った．それぞれの Kinect の位置，角度は異なるため，被写体の異なる面を撮影している．結果を表 2 に

Vol.2017-CG-168 No.37
Vol.2017-DCC-17 No.37
Vol.2017-CVIM-209 No.37
2017/11/9

示す. 接続台数が 1 台と 2 台では, フレームレートが 5.4fps 低下したが，2 台と 3 台の差は 1.2fps と大幅な低下はなかった.

5.3 考察

接続台数による速度低下の要因は 2 つ考えられる．1 つ目はボクセルデータの更新処理速度である．ボクセルデータの更新は接続した Kinect の台数回行う必要があるが，以降の処理は接続台数の影響をほぼ受けない．今後，接続台数を増やすために更新処理の高速化が求められる．2 つ目に描画面積の増加が挙げられる．それぞれのカメラで異なる角度から撮影しているため，接続台数が増えると重複しない面の描画が増える．この問題については，接続台数が増えるごとに重複面も増えるため，大きな影響はないと考えられる.

6. まとめ

本論文では複数の RGB-D カメラを用いて，リアルタイムの三次元モデル生成の手法を提案した.

今後は精度向上のために接続台数を増やしていく．しかし，接続台数の増加による描画速度の低下が確認されたため，ボリュームデータの更新処理の高速化が必要となる．また，よりリアルな三次元モデル生成の為，RGB データを用いたテクスチャマップの生成を検討している.

参考文献

[1] Newcombe, Richard A., et al. "KinectFusion: Real-time dense surface mapping and tracking." Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on. IEEE, 2011.
[2] Izadi, Shahram, et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera." Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011.
[3] Alexiadis, Dimitrios S., Dimitrios Zarpalas, and Petros Daras. "Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras." IEEE Transactions on Multimedia 15.2 (2013): 339-358.
[4] Besl, Paul J., and Neil D. McKay. "A method for registration of 3-D shapes." IEEE Transactions on pattern analysis and machine intelligence 14.2 (1992): 239-256.
[5] Lorensen, William E., and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." ACM siggraph computer graphics. Vol. 21. No. 4. ACM, 1987.