立体視デバイスを用いた大規模点群の高速描画

電気通信大学 ○石橋　寛史，齋藤　和人，増田　宏

Efficient Point-Based Stereoscopic Rendering for Supporting Maintenance Tasks

The University of Electro-Communications: Hiroshi Ishibashi, Kazuto Saito, Hiroshi Masuda

Point-cloud rendering systems based on stereoscopic devices are useful for recognizing the current situation of factories. However, in existing methods, details of point-clouds have to be omitted to achieve a sufficient rendering speed. In this research, we develop an efficient point rendering method for supporting maintenance tasks. In our method, we adjust the resolution of point-clouds to the one of stereoscopic devices for achieving high rendering performance. In addition, our system allows users to optionally increase the resolution at the user-specified points to investigate the details. We implemented our stereoscopic rendering method and evaluated the performance of our method.

Key words: Point- cloud, point-based rendering, stereoscopic rendering, head mounted display

1.　緒言

近年，老朽化が進んだインフラ設備や生産施設などの保全作業の効率化が重要な課題となっている．保守管理作業を効率化するため，これまでに，補修箇所の自動検出などの手法が検討されてきた．一方で現況の把握や検査においては，熟練作業員による目視作業も不可欠である．もし，現場の詳細な３次元情報が取得でき，立体視デバイスを用いて，これまでは現場で行なっていた現況の把握や検査が，現場に赴くことなくできるのであれば，保守作業の飛躍的な効率化が期待できる．

近年では，安価な立体視デバイスが開発されており，仮想環境の立体視が身近なものになっている．また，地上型レーザスキャナ(TLS)を用いれば，設備の高密度点群を短時間で得ることができる．そこで，本研究では，大規模な高密度点群を用いて，立体視デバイスによる保守点検作業を支援する手法について考える．

仮想環境上で設備の点検を行うにあたっては，高密度な点群が必要である．しかし大規模な点群はデータ量が大きく，描画のコストが非常に高い．一方で高速な描画を行うために点群を間引くと，点検を行うために十分な情報を得られなくなるという問題がある．これまでに，我々は，立体視デバイスの解像度に合わせて点群密度を調整することで，高速な点群立体視を実現する手法を提案した[1]．しかし，仮想空間上での移動に制限があり，また，点検業務に必要な詳細描画が行えないなどの問題があった．

そこで本研究では，これまでの手法を拡張し，高品質な大規模点群の描画と，ユーザが注視箇所をインタラクティブに詳細化できる手法について検討する．なお，本研究では，立体視デバイスとしてOculus Rift CV1とコントローラOculus Touch(図1)を用いて，提案手法の実装と評価を行う．

図 1 Oculus Rift CV1 と Oculus Touch

2.　検査業務に適した点群立体視

2.1.　検査業務で要求される描画機能

点検等の作業を考えた場合，目視検査は現状把握に有効な手段である．そのためには，立体視デバイスで仮想空間内を見渡して全体を把握することと，注目箇所をより詳細に観察できることが求められる．

そこで本研究では，立体視デバイスの解像度に合わせた「大局的描画」と，注視点を描画可能な最高密度で描画する「詳細描画」

図 2 デプスマップ

の二つを実現することを考える．大局的描画においては，立体視デバイスのコントーラで移動地点を指定すると，ユーザの視点がその位置に移動し，描画が更新される．また，詳細描画においては，ユーザが仮想空間中に注視範囲を指定すると，その範囲の点群を詳細化して描画する．詳細化した部分では，ユーザがその部分に近づいて見ると，細部まで再現された描画が得られる．

2.2.　大局的な点群描画

大規模な点群は描画コストが高く，全てを描画するとフレームレートが低下し，没入感やリアルタイム性を損なう． 一方で，立体視デバイスには解像度の上限があり，Oculus Rift CV1 の場合は片目あたり 1080×1200 ピクセル以上の点を描画する必要がない．また，両目から見えない領域の点は描画する必要がない．そのため，高密度な点群は，立体視デバイスの解像度に合わせて間引くことができ，また，不可視領域の点群は，描画対象から除外することができる．

本研究では以下の手順で大局的な点群描画を行う．

まず，岡本らの手法[1]で，図 2 に示すようなデプスマップを作成する．デプスマップとは，視点に最も近い点のみを抽出した可視点の集合のことであり，その解像度は，立体視デバイスの解像度に合わせて設定される．ここでは，横軸を方位角，縦軸を仰角とした二次元座標上に点群を投影し，同一のピクセルに複数の点が存在する場合は，視点からの距離が最も近いものを選択する．この際，処理の高速化のために，視点から十分離れた地点でスキャンされた点群はデプスマップ作成の処理から除外する．また，立体視デバイスから視線方向の情報を取得し，視野角の外にある点群を描画対象から除外する．視線の極座標ベクトルがわかれば，視野の範囲を決定することで計算が可能である．図 3 に，実際に描画を行った様子を示す．

ユーザが指定した地点に高速に移動するために，前処理として，DTM (Digital Terrain Model) を作成しておく．地表面上に格子を生成し，格子に含まれる点群でz座標が最も小さい値を保持する．移動地点は，指定した地点の 3 次元座標の地表面とする．

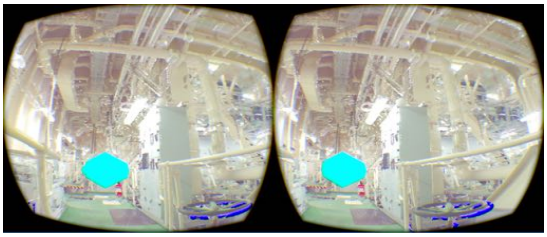


図 3 描画点群

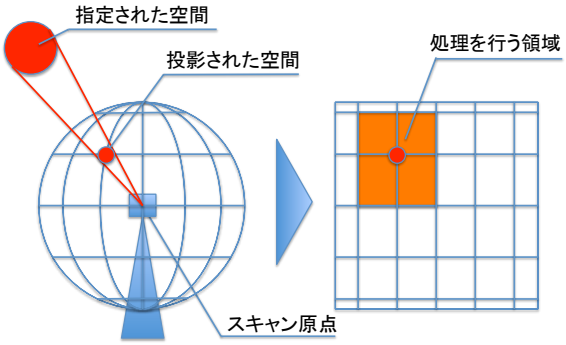
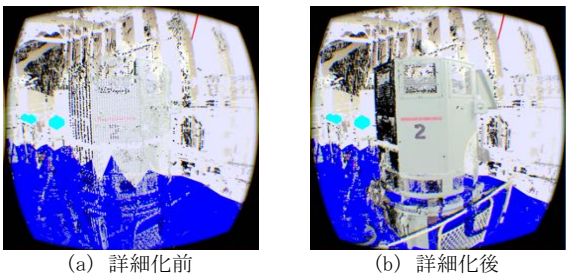


図 4 詳細化処理を行う点群の選択



(a) 詳細化前　(b) 詳細化後

図 5 詳細化前と詳細化後の点群

ユーザが指定した地点に移動する際に，移動先の視点を中心としたデプスマップを再構成する．これにより，大きな移動をしても描画の品質が低下しない移動手段が実現できる．大規模点群においてはすべての点群を RAM 上に保持することが難しいため，移動のたびに HDD 等のストレージから点群を読み込み，デプスマップを更新する．

2.3. 詳細点群の描画

ここでは，ユーザが指定した注視範囲について，その範囲の点群を抽出して描画する仕組みについて考える．

Oculus Rift は頭部の移動にカメラが追従することが可能なため，特定の物体を注視するような点検に有効である．しかし大域的描画はデプスマップを作成した初期の視点位置から見える点群のみを描画しているため，中心位置から見ることのできない物体の側面や裏面を見ようとすると，点が存在しないという問題がある．そのため，ユーザが注視範囲を指定し，その部分についてのみ詳細な点群を描画することで，十分な描画速度である 90 fps を維持したまま詳細な点検を行うことを考える．

まず，立体視デバイスのコントーラを用いて，移動地点の指定と注視箇所の指定を行う．移動地点を指定すると，ユーザの視点はその地点に移動し，描画される点群が更新される．

次に，注視箇所を指定するときには，注視する場所と範囲を指定する．ここでは，注視箇所は，空間中の球によって表現するものとし，ユーザは注視地点をコントーラで指定した後，画面上に描画される球を操作して半径を決定する．

ここで指定された球内に含まれる点群の解像度を高くして，詳細化された描画を行う．指定された空間内の点群が複数のスキャンによって計測されている場合，それらから球内の点群を選び出し，詳細度を高める．しかしながら，すべての点群を RAM 上に保持することは困難であり，また，膨大な点群に関して球の内外判定を行うことは大きな処理コストを必要とする．そこで，構造化された点群を外部記憶装置に保持し，必要な部分のみにアクセスすることで効率的な判定を行う．

ここでは，TLS 点群が，横軸を方位角，縦軸を仰角とした 2 次元配列状に格納されていることを利用して，処理を高速化する．スキャンごとに点群を粗い格子状に区切って格納し，各格子点に周囲 4 つの矩形情報を持たせる．ユーザによって選択された空間をスキャナの持つ 2 次元配列上に投影し，選択された空間に含まれる格子点を抽出する．抽出された格子点の周りの 4 つの配列要素に含まれる点群に対してのみ RAM 上に読み込んで内外判定を行うことで，詳細化する点群を選定する(図 4)．

3. 評価実験・考察

本研究では，ユーザがコントローラで注視箇所を指定することで，その範囲を詳細化する．ここでは，球内のすべての点群を読み込むこととした．本手法では，選択球の半径が小さいうちは 90 fps 以上の描画性能が実現できるが，半径が大きくなるに従って描画すべき点の個数が大きくなるため，描画性能が低下していくことを確認した．

そこで，描画処理をする点群数と fps との関係を検証した．立体視デバイスは Oculus Rift CV1，コントローラには Oculus Touch を用いた．実行環境は，CPU: Intel Core i7-6700K，GPU: GeForce GTX 1080，RAM: 64.0GB で，使用した言語は C++である．

描画すべき点の個数を徐々に増加させていって，fps を測定した結果を図 6 に示す．我々の検証実験では，描画点群数が 2000 万点程度までは，Oculus Rift CV1 の最高値である 90 fps が実現できたが，その範囲を超えると fps は急速に低下した．

このことから，詳細点群を描画する場合には，総描画点数が 上限である 2000 万点付近を超えないように設定する必要があることが明らかになった．すなわち，上限の点数を N_{max}，大局的描画の点数を N とするとき，詳細点群を $N_{max}- N$ 個以下まで間引く必要がある．なお，本手法では，間引く事によって不自然なパターンがでないように，乱数を用いて間引いている．

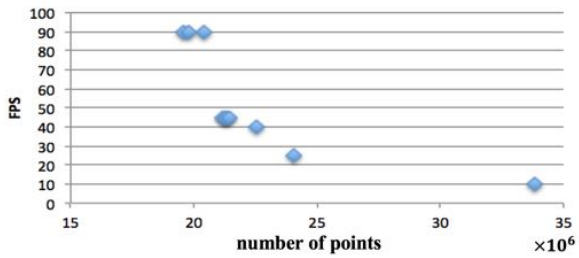


図 6 描画点数と描画速度の関係

4. まとめ

リアルタイム性を損なわない大規模点群の高速描画を実現した．また，選択された範囲の点群を高速に抽出し，詳細に描画する手法を示した．また，本手法を実装して評価を行い，高速描画できる上限の点数を明らかにした．

今後は，詳細化範囲の自動抽出や，階層的な点群の詳細化手法を検討していく．また，点群のセグメンテーションを行い，距離に応じて点の描画サイズを変えていく手法についても検討する．

参考文献

[1]岡本大樹，丹羽健，増田宏:立体視に適した大規模点群のレンダリング(第 2 報)，精密工学会春季講演会 2015