修士論文

三次元点群の高速・高品質な可視化のための点密度を考慮したオクルージョン推定

A Study on Occlusion Estimation for Fast and High-Quality Rendering of 3D Point Clouds Considering Point Density

令和4年1月27日 提出

指導教員　小川 剛史 准教授

東京大学大学院
工学系研究科　電気系工学専攻
37-206522　　森島　正博

要旨

三次元点群は，バーチャル観光やテレプレゼンスなど，現実空間の視覚的な再現を目指すアプリケーションでよく利用される．このようなアプリケーションでは，レンダリングの品質の観点から，点群をもとにメッシュによる 3D モデルを作成することが一般的であるが，高負荷な事前計算や手作業での調節が必要な場合がある．点群を直接用いた高速・高品質なレンダリングを行うため，画像空間でオクルージョンが発生している点を推定する方法が提案されているが，表示する点群の密度に応じたパラメータ調整が必要である．そこで本稿では，画像空間における局所的な点の密度を考慮し，密度に応じた調整に依存しない推定手法について検討した．また，物体の境界付近における推定の誤りを改善するため，深度値の勾配を用いた推定手法の検討を行った．

Abstract

3D point clouds are often used in applications that aim to visually reproduce the real world, such as virtual tourism and telepresence. Many applications generate meshes from point clouds to render high-quality surfaces, but these approaches require costly computation in preprocessing or manual parameter tuning. To achieve fast and high-quality rendering by directly using point clouds, methods have been proposed to estimate the occluded points in screen space. However, it is necessary to adjust the parameter according to the density of the points. In this paper, we studied an estimation method that does not depend on adjustment according to the density by considering the local density of the points in screen space. Furthermore, in order to improve the estimation error near the object boundary, we investigate an estimation method using the gradient of the depth.

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究背景

点群は，物体表面を点の集合として表現したデータであり，フォトグラメトリに関する技術や 3D レーザースキャン技術の発達により容易に取得でき，実在する建物などを三次元的にキャプチャすることで物理的な移動を伴わない鑑賞を可能とするバーチャル観光 [2,3] や，三次元映像を用いて臨場感のある遠隔コミュニケーションを可能とするテレプレゼンスに関するアプリケーション [4,5] など，現実空間の視覚的な再現を目指すアプリケーションに利用されている．点群のように三次元的な形状の情報を持つデータを用いることで，6 自由度の視点移動が可能となり，特に近年ではバーチャルリアリティ（VR）の技術も取り入れられ，盛んに研究が行われている [1,6].

これらの応用では，レンダリング品質などの観点から，メッシュ表現による 3D モデルが一般的に利用される [2]．しかし，点群からメッシュを作成することは容易ではなく，細かい部分まで表現するために多数のメッシュが必要となり，その生成に高負荷な計算が必要となる場合や，手動でのパラメータ調整が必要となる場合がある [2,7–9].

そこで高負荷な計算や作業を省略するため，直接，点群をレンダリングに用いるアプローチが存在する [7,8,10,11]．点群を直接用いることにより，アプリケーションの作成を単純化することが可能となる．また，点群はメッシュに比べてデータ構造が単純であり，遠隔コミュニケーションなどリアルタイム性が重要なシステムに適しているという利点もある [12,13]．このアプローチにおける課題の一つはレンダリング品質であり，点群は点同士の連結に関する情報を持たないため，本来は物体によって遮蔽されて見えないはずの部分が点と点の間から透けて見え，適切にオクルージョンが反映されない問題がある [11]．現実空間の視覚的な再現を目的とするアプリケーションにおいて点群を直接用いるためには，没入感を損なわないようにレンダリング品質を改善する必要があり，また，インタラクティブな動作を可能とするために高速にレンダリングを行う必要がある．

点群のレンダリング品質の改善手法として，画像処理に基づく手法 [9,14] が挙げられる．これらは，遮蔽されている可能性が高い点を画像空間上で推定（オクルージョン推定）し

て除去することで適切なオクルージョンを再現し，補間処理を行うことで品質を改善する手法であり，VR などのインタラクティブなアプリケーションに向けて高速に動作する手法も提案されている [9].

しかし，これらの手法では，点群に含まれる点の密度が均一であるということが前提とされており [15]，点の密度によってオクルージョン推定に用いるパラメータを調整する必要があるという課題がある．よって，表示する点群を切り替えた際にパラメータの再調整が必要となる問題や，点の密度が不均一である場合に，パラメータ調整が困難になり品質が低下したり，視点変更時に再調整が必要となるといった問題が生じると考えられる．

1.2　研究目的

本研究の目的は，点群を直接用いて現実空間の視覚的な再現を目指すインタラクティブなアプリケーションに向けて，高速で高品質な点群のレンダリング手法を提案することである．また，遠隔コミュニケーションなどのリアルタイム性が重要であるアプリケーションに向け，事前処理が必要ない手法を目指す．この目的のために，前節で示した背景をふまえ，既存のレンダリング手法 [9] のオクルージョン推定処理に着目し，点の密度に応じたパラメータ調整を必要としないオクルージョン推定アルゴリズムの検討を行い，その有効性を検証する．また，レンダリング品質をさらに改善するため，物体の境界に着目したオクルージョン推定手法の検討と評価も行う．本研究では，提案手法の適用範囲を広げるため，既存研究 [9] と同様に基本的に各点の座標のみを用いるオクルージョン推定手法を検討した．

1.3　本研究の貢献

本研究では，画像空間における局所的な点の密度を考慮したオクルージョン推定を提案し，評価実験により，既存手法でパラメータ調整を行った結果と同程度，あるいは，それ以上の品質の結果が本手法によって得られる場合がある事を確認した．また，深度値の勾配を用いて物体の境界に着目したオクルージョン推定を行うことにより，物体の境界付近での推定誤りが改善し，レンダリング品質が改善される場合があることを確認した．

1.4　本論文の構成

第 2 章では，現実空間の再現を目指すアプリケーションにおいて点群を直接用いてレンダリングを行うシステムに関する研究や，点群のレンダリング手法を中心に関連研究について述べる．第 3 章では，本研究で検討した点の密度を考慮したオクルージョン推定手法と，物体の境界に着目したオクルージョン推定手法について詳細を述べる．第 4 章ではそれぞれの手法の有効性を検証する評価実験について述べ，第 5 章では評価実験で明らかになった問題を改善するための検討について述べる．最後に，第 6 章で結論と今後の課題を述べる．

第2章　関連研究

2.1　現実空間の再現を目指すアプリケーション

2.1.1　シーンの表現方法

現実空間の視覚的な再現は，近年の VR に関する研究の目標の一つとしても挙げられ，この目標が達成できれば，遠隔でのコミュニケーションや医療を可能にするテレプレゼンスの領域や，物理的な移動を伴わないバーチャル観光など，様々な応用が期待できる [1]．現在，様々な現実空間の表現方法が提案されており，画像ベースの手法と，形状ベースの手法に大別される [1]．画像ベースの手法は，主にカメラなどを用いて取得した現実空間の情報を直接用いて可視化する手法であり，視覚的に高品質な再現が可能な表現であると言える．画像ベースの手法の例として，360° 映像を用いたシステム [16,17] が挙げられる．360° 映像を用いることで，ユーザーは 3 軸方向に視点を回転することで空間を自由に見回すことができるが，視点の位置は固定されているため 3 軸方向の平行移動は難しいという課題がある．図 2.1 に示すように，3 軸方向の視点回転のみが可能となっている状態は three-degrees-of-freedom（3DoF）と呼ばれ，3 軸方向の回転に加えて 3 軸方向の平行移動が可能となっている状態は six-degrees-of-freedom（6DoF）と呼ばれる [1]．この呼び方に従うと，360° 映像などの画像ベースの手法を用いたシステムの視点移動は 3DoF であるものが多い．6DoF の視点移動が可能な画像ベースのシステム [18,19] も提案されているが，狭い範囲でのみ視点の平行移動が可能となっている．

一方，形状ベースの手法は，三次元点群や，メッシュとテクスチャを用いた 3D モデルなどを使用してシーンを表現する手法である．シーンの形状は，LIDAR を用いた 3D レーザースキャンや，複数の写真から Structure from Motion（SfM）を用いて三次元点群として取得可能である．現実空間を再現を目指すアプリケーションでは，レンダリング品質などの観点から，点群からメッシュによる 3D モデルを作成することが一般的である [2]．形状ベースの表現は，画像ベースの表現とは異なり三次元的な形状の情報を保持しているため，システム内で完全な 6DoF の視点移動が可能となる．画像ベースの手法に点群とメッシュを用いた形状ベースの手法を組み合わせ，広範囲での 6DoF の視点移動を可能とする

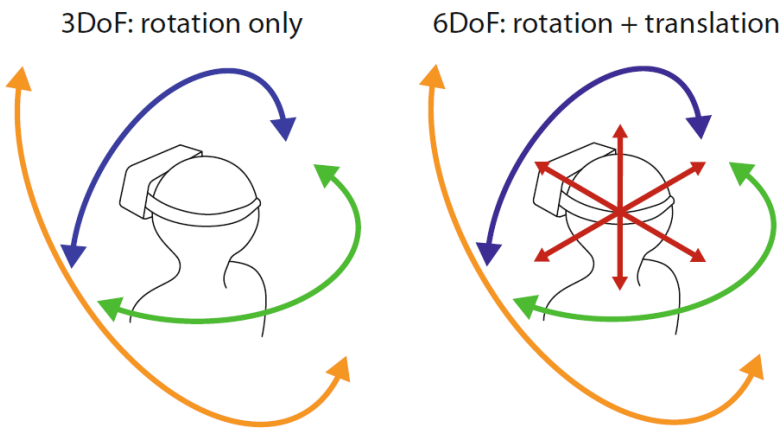


図 2.1　視点移動の自由度 [1]

手法も提案されている [20]．ただし，次の 2.1.2 節で述べるように，メッシュの作成には様々な課題がある．本研究では，6DoF の視点移動が可能でありメッシュの作成が必要ない，三次元点群を用いたシーンの表現に着目する．

2.1.2　点群を直接用いたアプリケーション

前述のように，現実空間を再現を目指すアプリケーションでは，点群からメッシュによる 3D モデルを作成することが一般的である．しかし，メッシュの作成という事前処理が必要となることに加え，メッシュの作成自体にも様々な課題がある．例えば，フォトリアリスティックな結果を得るためには大量のメッシュを生成する必要があるという課題 [7] や，メッシュの生成処理に時間がかかるという課題 [8, 9] が挙げられる．また，手動での調整作業が必要となる場合がある [2]．

そこで，いくつかの研究ではメッシュの作成を省略し，点群を直接用いてレンダリングを行うアプローチがとられている．例として，Head Mounted Display（HMD）を用いて VR 空間で点群を可視化するシステムが挙げられる [7, 10, 21]．Virtanen らは VR 空間で点群を移動させたり，点群に着色したりなど，インタラクティブなアプリケーションを想定したシステムを提案している [11]．また，点群の共有を容易にするために，Web ブラウザを用いた点群の可視化システム [8] も提案されている．

点群を直接用いることによりメッシュ作成の処理を省略できることに加え，点群は連結に関する情報を保持する必要が無く，メッシュに比べて単純なデータ構造であることも利点として挙げられる [13, 22]．このような利点は，リアルタイム性が重要なシステムで重要

であり，リアルタイムに取得されている点群を直接可視化するシステム [23, 24] や，点群を用いた三次元映像によるリアルタイムなコミュニケーションに関するシステム [12] が提案されている．また，点群を用いた三次元映像の効率的なストリーミングを可能とする手法 [25] も提案されている．

一方で，点群は点同士の連結に関する情報を持たないため，物体に遮蔽されているはずの箇所が点と点の間から透けて見えてしまうといったオクルージョンに関する問題など，レンダリング品質に関する課題がある [11]．本研究は，点群を直接用いてレンダリングを行うアプリケーションにおいて，レンダリング品質を改善することを目的としている．

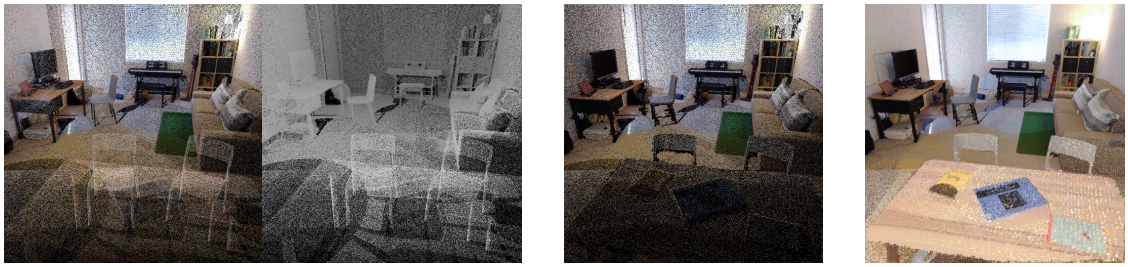
2.2　点群のレンダリング手法

2.2.1　高品質なレンダリング

前述の点群のレンダリング品質に関する課題に対し，これまでに様々なレンダリング品質の改善手法が提案されている．点群の高品質なレンダリング手法として，Splatting [26, 27] が挙げられる．Splatting は，例えば，点群に含まれるそれぞれの点を点として描画するのではなく，楕円体などの広がりを持つ形状としてブレンドを行うことでレンダリング品質を改善する手法である．しかし，それぞれの点に対応する法線や半径などの情報が必要であり，事前に高負荷で時間のかかる計算が必要となる場合がある [9, 11]．リアルタイムでの法線や半径の計算が可能な Splatting 手法 [24, 28] も提案されているが，近傍探索を用いており計算負荷が大きいという課題や [9]，近傍探索の近似が原因で法線が不正確になる場合があるという課題がある [24]．

事前の計算を必要としない手法として，点を 1 画素としてではなく，正方形や円などの大きさを持つ形状として描画するという手法が挙げられる．シンプルな手法ではあるが，点と点の間が補間され，オクルージョンもある程度再現することが可能である．Scheiblauer らは，点の距離と密度によって適切な点の大きさを決定する手法を提案している [29]．また，Schütz らは，正方形や円ではなく放物面を用いることによって，より高品質な結果を得る手法を提案している [30]．一方で，これらの手法では滑らかな表面を得ることが難しいなど，品質に関する課題がある [9]．

また，近年，点群のレンダリング結果の品質を機械学習を用いて改善する研究が行われている [1]．点群を入力，写真などの所望のレンダリング結果を正解としてモデルを学習させることによりフォトリアリスティックなレンダリング結果を得ることが可能であり，品質の改善において有効であることが示されている [31, 32]．また，色や光源を調整可能な機



(a) 各点を1画素として画面に投影した画像と深度マップ　(b) オクルージョン推定と除去　(c) レンダリング結果

図 2.2　画像処理に基づく点群のレンダリング品質の改善

械学習によるレンダリング手法 [33] も提案されている．一方で，シーンごとに学習が必要となる場合があるという課題や，大規模なネットワークを用いた計算を行う必要があり，高速なレンダリングは難しい場合があるといった課題がある [1].

事前計算を必要とせず，高速かつ高品質な点群のレンダリング手法として，画像処理に基づく手法が挙げられる．これらの手法は，点群を投影した画像に対して処理を行うことで品質を改善する手法である．図 2.2 に処理の例を示す．まず，点群の深度マップを用いてオクルージョン推定を行う．この時，判定対象の点の画像空間における近傍領域内の点が推定に用いられる．オクルージョン推定により，遮蔽されていると判定された点は除去される．次に，除去されずに残った点を膨張させて，点が存在しない画素を補間する処理が行われる．Bouchiba らは，Pintus らが提案した画像処理に基づく手法 [14] の計算量を改善した手法を提案している [9]．本研究では，画像処理に基づく Bouchiba らの手法 [9] に着目して，2.3 節で述べるようなオクルージョン推定処理の課題を改善するための検討を行う．

2.2.2　レンダリングの高速化

点群の高品質なレンダリングに加え，インタラクティブなアプリケーションにおいては高速なレンダリングも重要である．level-of-detail（LOD）は，例えば，視点の近くでは高密度の点群，遠くでは低密度の点群を表示するといった方法により，レンダリングする点の数を削減することで高速化を行う手法である [7]．Schütz らは，LOD における密度を連続的に変化させることにより，没入感を損なわないような LOD 手法を提案している [34]．また，1 フレームですべての点をレンダリングするのではなく，数フレームにわたって少しずつ描画することで高速化を目指す手法 [35, 36] も提案されている．さらに，空間をセ

ルなどに分割した上で各セルに点を格納し，オクルージョンによって見えないセルに含まれる点のレンダリングを省略する，occlusion culling も提案されている [25,37]．前述のような点の削減の他に，GPU による実装を工夫することで点群のレンダリングの高速化を行う手法も提案されている [38,39]．本研究は点群の高品質なレンダリングに着目しているため，本節で述べたような高速化処理の実装や評価は行っていないが，点群の投影を行った後の画像処理に基づく品質の改善を目指すため，これらの高速化手法も同時に適用できる可能性があると考えられる．

2.3　点群におけるオクルージョン推定手法

これまでに様々なオクルージョン推定手法が提案されている．簡単な手法の一つとして，点の法線を用いた手法が挙げられる．これは，各点 i の法線ベクトル $\boldsymbol{n}_i$ を計算し，$\boldsymbol{n}_i$ がカメラの方向を向いていれば点 i は見えていて，そうでなければ見えてないなどと判定する手法である．しかし，法線の計算には前処理が必要であることに加え [9]，同じ方向の法線を持つ点同士のオクルージョンを正しく判定することができない [40]．Katz らは三次元空間において点群を反転し凸包を求めることによるオクルージョン推定手法を提案しているが [41]，リアルタイムでの動作は難しい [14]．また，2.2.2 節で述べたような occlusion culling により，高速化と同時に，遮蔽されているはずの点が見えてしまう問題を緩和することが可能であるが，空間の分割単位ごとにオクルージョンを判定するため，大まかな判定のみが可能となっている [25].

リアルタイムな動作が可能であるオクルージョン推定手法として，画像処理に基づく手法がある [9,14]．Bouchiba らの研究 [9] では，Pinuts らの研究 [14] の計算量と補間処理の改善などが行われている．これらの手法は，事前計算の必要が無く，品質の観点からも有効であるとされている [9,11]．一方で，点群に含まれる点の密度によってオクルージョン推定に用いるパラメータを調整する必要があり，また，密度が均一であるということが前提となっている [15]．点群の取得方法によって密度が不均一になる場合があり [11,15,35]，また，文献 [11,31] で提案されているシステムのように，別々にキャプチャされた複数の点群を用いて一つのシーンを構成する場合に密度が不均一になる場合が考えられるため，点の密度が均一であるという仮定や，密度に応じたパラメータ調整に依存しないオクルージョン推定が可能であることが望ましいと考えられる．密度が不均一な場合も考慮したオクルージョン推定も提案されているが [15,42]，VR などのインタラクティブなアプリケーションへの応用は議論されておらず，文献 [9] と比較して高速な動作は難しいと考えられ

る．本研究では，画像処理に基づく手法 [9] に着目し，点の密度に応じたパラメータ調整を必要としないオクルージョン推定アルゴリズムの検討を行う．

2.4　Bouchiba らのオクルージョン推定手法

Bouchiba らはオクルージョン推定に加え，レンダリング品質を改善するための補間処理や，シェーディング処理を提案しているが，ここでは本手法を説明するのに必要なオクルージョン推定に関する事項を記す．文献 [9] では，すべての画素についてオクルージョン推定が行われる．その際に判定対象としている点の画像空間上での近傍の点が使用される．使用する近傍の点を高速化に選択するために，点の 3 次元位置を格納した層状のテクスチャが作成される．この処理について次の 2.4.1 節で詳しく述べる．

2.4.1　Pyramid Construction

まず，画素値として，投影される点のカメラを基準とした 3 次元的な位置 (x, y, z) を格納したテクスチャを準備する．その後，図 2.3 のように，2×2 ピクセルの範囲において，z 成分が最小の画素を残すようにダウンサンプリングして解像度を半分に落としたテクスチャを作成する．解像度が最大のテクスチャを 0 層目としてこの処理を繰り返すことにより，ピラミッド状の層を持ち，各画素に点の 3 次元位置が格納されたテクスチャが作成される．

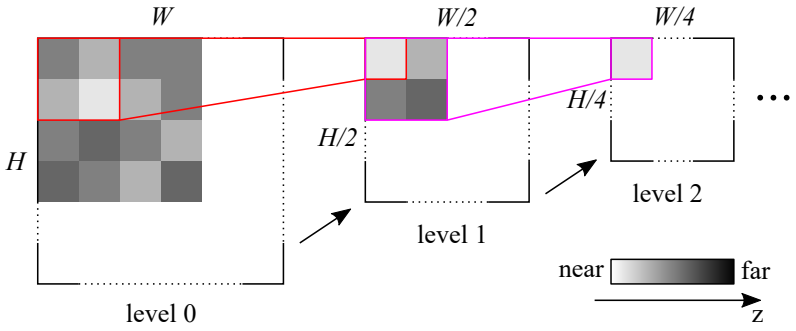


図 2.3　ダウンサンプリングの概要

2.4.2 Occlusion Operator

判定対象の点の 3 次元座標を $\boldsymbol{x}$，選択された近傍の点の 1 つの 3 次元座標を $\boldsymbol{y}$ として，式 (2.1) によって次の値（occlusion value）を計算する．

$$1 - \frac{\boldsymbol{y} - \boldsymbol{x}}{||\boldsymbol{y} - \boldsymbol{x}||} \cdot \frac{-\boldsymbol{y}}{||\boldsymbol{y}||} \tag{2.1}$$

この値は，点 $\boldsymbol{x}$ が点 $\boldsymbol{y}$ によって遮蔽されている度合いを表し，より遮蔽されている場合に小さい値となる．いくつかの近傍の点について式 (2.1) を計算し，得られた値の平均値がしきい値を下回っていれば遮蔽されているとして除去される．

式 (2.1) の計算に用いる近傍の点は，判定対象の点を中心とした画像空間における正方形の範囲（近傍領域）内に存在する点から選択される．詳細は省略するが，選択を高速化するため，近傍領域のサイズは，level 0 (3×3)，level 1 (6×6)， level 2 (12×12) のように段階的に決定され，一辺の長さは 2 のべき乗に比例するようになっている．

近傍領域のサイズは，近傍の点が含まれるために十分な大きさである必要がある一方で，大きすぎても推定誤りが起こりやすくなるため，適切に定める必要がある．文献 [9] では，画素 i の近傍領域サイズの一辺の長さ $L[i]$ は，近くの物体ほど大きく，遠くの物体ほど小さく表示されるのに合わせて，深度値 $z[i]$ を用いて，式 (2.2) のように決定される．

$$L[i] = \frac{p}{z[i]} \tag{2.2}$$

ただし，p は，解像度やカメラの視野角，点の密度などを含む，調整が必要なパラメータである．また，深度値 $z[i]$ には，解像度を落とした 4 層目のテクスチャの z 成分が使用される．最終的な近傍領域サイズの段階数 $l[i]$ は，$l[i] = \log_2 L[i]$ で決定される．

第3章　提案手法

3.1　点の密度を用いた近傍領域サイズの決定

2.4 節で述べたように，既存研究ではオクルージョン推定の際には判定対象の点の画像空間上での近傍の点が使用され，深度値から適切な近傍領域サイズを計算するために，点の密度に応じてパラメータ p を調整する必要がある．そこで本稿では，深度値ではなく点の密度を用いて適切な近傍領域サイズを計算することを提案する．具体的には，画像空間上での局所的な点の密度を計算し，近傍領域サイズを定める手法を検討する．本手法により，オクルージョン推定の際に点の密度に応じたパラメータ調整の必要がなくなる事，また，場所によって密度が異なる場合にそれぞれの場所で適切な近傍領域サイズが計算できるため，より高品質な結果が得られる事が期待される．

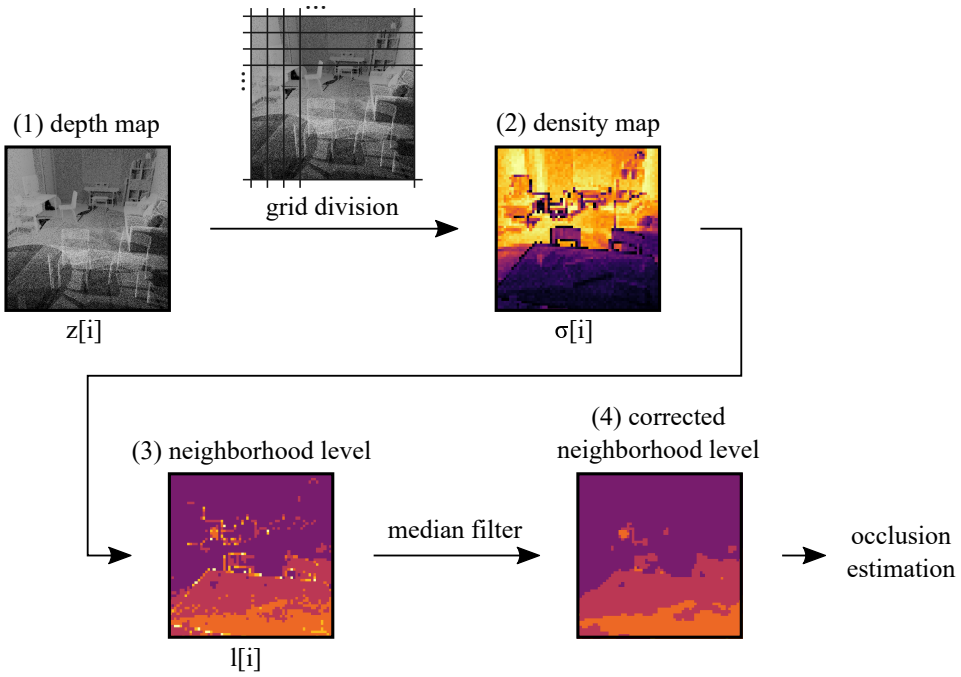


図 3.1　提案手法の概要

図 3.1 に提案手法の概要を示す．以降は，提案手法の詳細を二段階に分けて述べる．

3.1.1　画像空間での点の密度の算出

まず，点群を投影した深度マップを用いて点の密度を計算する．本稿では，画像全体をグリッド状に分割し，各グリッド内に存在する点を数えることで局所的な点の密度を算出した．この時のグリッドの大きさは，文献 [9] で近傍領域サイズの決定に 4 層目のテクスチャの深度値が用いられているのに従い，$2^4 = 16$ ピクセル四方とした．

また，ある点が遮蔽されていると判断するためには，その点を遮蔽している点が近傍領域に含まれる必要がある．したがって，近傍領域サイズの計算のためには，各グリッド内に存在する点の中で，特に視点に対して手前側に存在する点の密度が重要であると考えられる．そこで本稿では，各グリッドにおける深度値の最小値 $z_{\min}$ を基準として，以下の式 (3.1) を満たす点 i を数えてグリッドの面積 ($16^2 = 256$) で除算することで密度を計算した．

$$z[i] - z_{\min} < e \tag{3.1}$$

ただし，e は密度の計算に用いる点を深度値によって区別するためのしきい値であり，本稿では $e = 0.04$ (40 cm) とした．この方法により，例えば，図 3.2(b) にオレンジ色で示す点が密度の計算に用いられる．図 3.3 にすべての点を数えた場合と，式 (3.1) を満たす点のみを数えた場合の密度のマップを示す．すべての点を数えた場合，図 3.3(a) に示すように，テーブル部分などの奥行方向に物体が重なっている箇所では計算された密度が大きくなっている．一方，式 (3.1) の条件により手前の点のみを数えた場合，図 3.3(b) に示すようにテーブル部分の密度が低く計算されている．

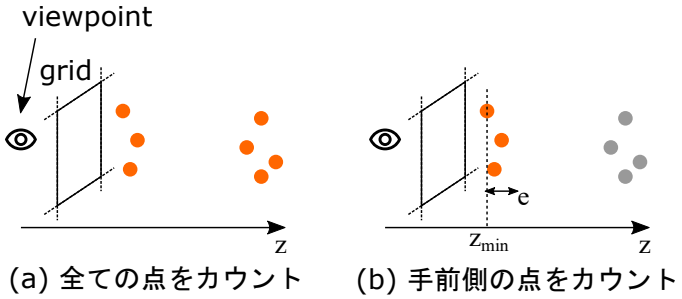


図 3.2　密度の計算方法

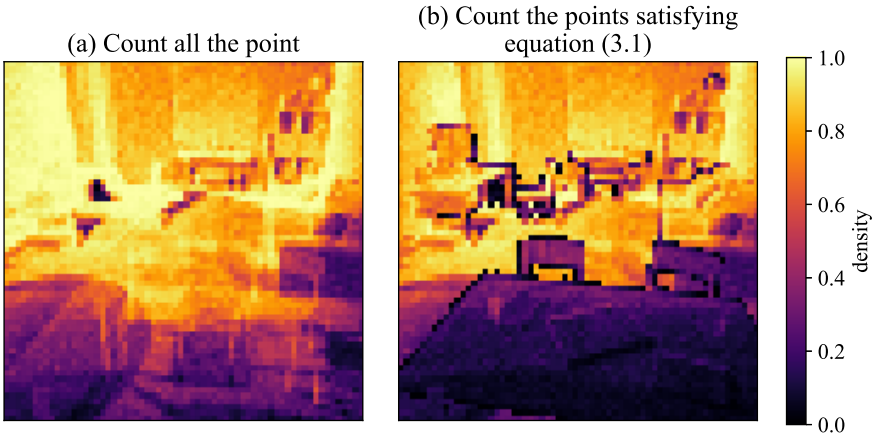


図 3.3　計算された密度 σ[i]

3.1.2　密度を用いた近傍領域サイズの計算

2.4 節で述べたように，既存手法では，近傍領域サイズの一辺の長さ $L[i]$ は深度値に反比例するように定められている．したがって，点群に含まれる点同士の間隔が一定であれば，透視投影後の画像空間において，近傍領域に含まれる点の数はおおよそ一定になると考えられる．本稿ではこのことに着目し，近傍領域に含まれる点の数が一定値 C になるように一辺の長さ $L[i]$ を求めることを検討した．具体的には，3.1.1 節で求めた，画素 i が含まれるグリッドの密度を $\sigma[i]$ として，$\sigma[i] \cdot L[i]^2 = C$ が成り立つように，パラメータ p' を用いて，式 (3.2) のように定めた．

$$L[i] = \frac{p'}{\sqrt{\sigma[i]}} \tag{3.2}$$

次の第 4 章の評価実験により p' の値を実験的に決定し，シーンによらず同じ値を用いた．

最終的な近傍領域サイズの段階 $l[i]$ は，文献 [9] と同じく $l[i] = \log_2 L[i]$ で決定することを検討したが，図 3.3(b) のように，特に物体の境界付近の密度が小さく計算される傾向があり，図 3.4(a) のように境界付近の近傍領域が大きなサイズになってしまう場合があった．そこで，$l[i]$ のマップに 3×3 のメディアンフィルタを適用することで，図 3.4(b) のように近傍領域サイズの補正を行った．

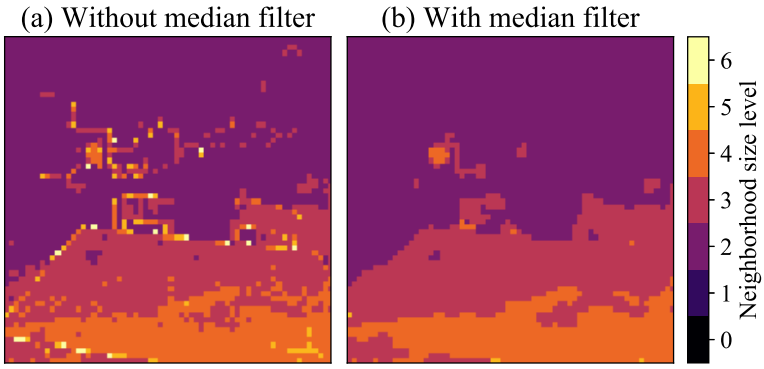


図 3.4　メディアンフィルタを用いた近傍領域サイズの補正

3.2　深度値の勾配を用いた近傍領域サイズの補正

深度値を用いて近傍領域サイズを決定する既存手法や，前述の点の密度を用いた近傍領域サイズの決定手法においては，オクルージョン推定において特に物体の境界付近で誤りが多く，レンダリングの際に過剰な点の除去が行われてしまう場合があった．原因の一つとして，着目している点が例えば図 3.5(a) のような平面の中央部にあるのか，それとも図 3.5(b) のようなエッジ部にあるのかなど，近傍領域内における点同士の配置特徴が考慮されていないことが挙げられる．そこで，点同士の配置特徴を考慮するため，深度値の勾配を用いて近傍領域サイズを補正することを検討した．以降は勾配の具体的な算出方法と，勾配の値を用いた近傍領域サイズの補正方法について記す．

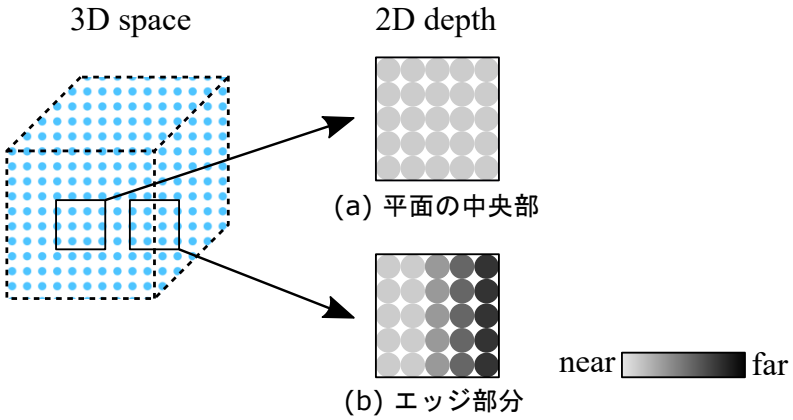


図 3.5　三次元空間上で立方体の表面に点が配置されている例

3.2.1　深度値の勾配の算出

まず，点群を投影した際の深度マップにソーベルフィルタを適用し，各画素 i おける勾配の大きさ $g[i]$ を算出する．ここで，点群を投影した際の深度マップは画像全体として深度の変化が激しく，直接フィルタを適用しても図 3.6(a) のようにエッジが抽出されなかった．そこで，2.4.1 節で述べたダウンサンプリング処理を何回か行うことによって平滑化された深度マップを用いて勾配の計算を行った．勾配の計算結果を図 3.6(b) に示す．ダウンサンプリングによってエッジが適切に得られるようになる一方，エッジの解像度は低くなるため，ダウンサンプリングは深度マップが平滑化されるために必要最低限の回数だけ行うべきだと考えられる．そこで，ダウンサンプリングの回数は，点が密集している箇所では少なく，点がまばらな箇所では多くなるように，補正前の近傍領域サイズの段階数 $l[i]$ に従った．

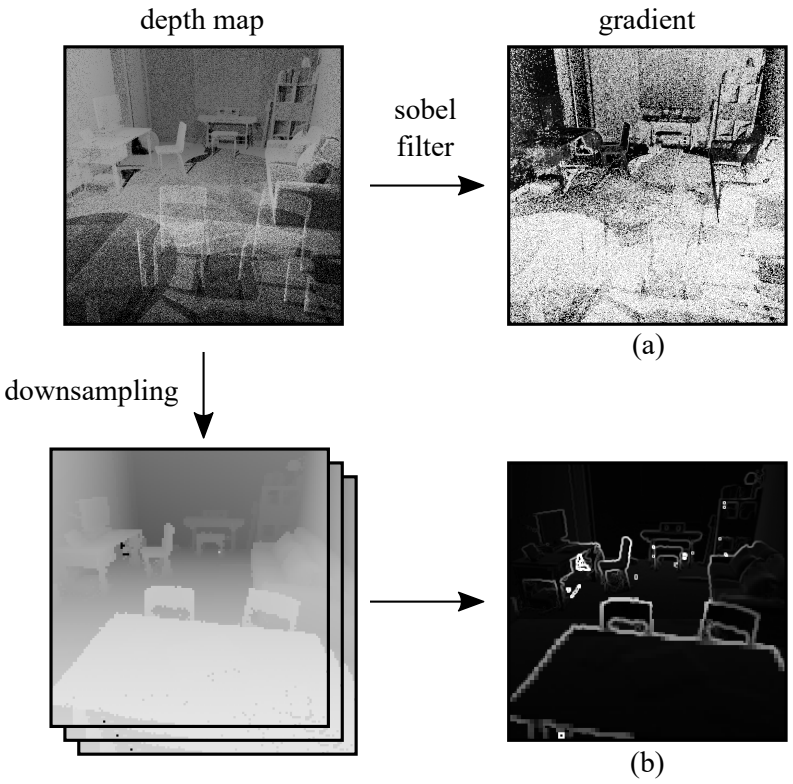


図 3.6　深度マップへのソーベルフィルタの適用結果

3.2.2　近傍領域サイズの補正

次に，算出した勾配 $g[i]$ 用いて，オクルージョン推定に用いる近傍領域サイズの補正を行った．物体の境界付近，つまり，勾配の値が大きく空間的な高周波成分が大きい箇所では，細かい箇所を重視するために近傍領域を小さく補正した方が正確な推定ができると考えられる．そこで，補正後の近傍領域サイズの段階数 $l'[i]$ を，式 (3.3) のように，勾配の大きさがしきい値 g_{th} 以上であれば，補正前の近傍領域サイズの段階数 $l[i]$ より 1 段階小さくなるように設定した．それ以外の場合は $l[i]$ に従った．

$$l'[i] = \begin{cases} l[i] - 1 & (g[i] > g_{\mathrm{th}}) \\ l[i] & (\mathrm{otherwise}) \end{cases} \tag{3.3}$$

第4章　評価実験

4.1　実験方法

4.1.1　使用した点群

本研究では，Park らの研究 [43] で使用された，LIDAR スキャナで屋内を測定した RGB カラーを含む点群[1]を使用した．使用した点群を MeshLab[2] [44] を用いて可視化した様子を図 4.1 に示す．点群の密度による影響を明確に評価するため，CloudCompare[3]の Subsample 機能により元の点群からサンプリングして作成した点群を用いて実験を行った．具体的には，元の点群から 1000 万点をランダムにサンプリングした高密度な点群と，その点群から 100 万点をランダムにサンプリングした低密度な点群を準備した．また，特定のオブジェクトのみを低密度とするなど，高密度な部分と低密度な部分が存在するように統合した点群も作成した．これらの点群と 2 種類の視点の組み合わせた合計 10 個のシーンで実験を行った．使用した視点と点群の統合の際に着目したオブジェクトを図 4.2，シーンの詳細を表 4.1 に示す．

図 4.1　使用した点群

[1] http://redwood-data.org/indoor_lidar_rgbd/
[2] https://www.meshlab.net/
[3] https://www.cloudcompare.org/

(A)　(B)

図 4.2　使用した視点 (A, B) と着目したオブジェクト

表 4.1　使用したシーン

シーン名	視点	密度に関する詳細
(A1)	(A)	高密度（1000 万点）
(A2)	(A)	低密度（100 万点）
(A3)	(A)	高密度（画像左半分）＋ 低密度（右半分）
(A4)	(A)	低密度（table）＋ 高密度（table 以外）
(A5)	(A)	低密度（desk）＋ 高密度（desk 以外）
(A6)	(A)	高密度（desk）＋ 低密度（desk 以外）
(B1)	(B)	高密度
(B2)	(B)	低密度
(B3)	(B)	低密度（desk）＋ 高密度（desk 以外）
(B4)	(B)	高密度（desk）＋ 低密度（desk 以外）

また，文献 [9] では，背景，つまり，点が投影されていない画素についてもオクルージョン推定を行っているが，本稿では屋内環境で取得された点群のみを実験で使用したため，既存手法，提案手法ともに，比較の際には背景のオクルージョン推定は省略し，背景の画素はすべて補間処理による上書きの対象とした．

4.1.2　レンダリング品質の評価指標

画像の品質の定量的な指標として，PSNR（Peak signal-to-noise ratio）を用いた．PSNR は，ある画像の品質を対応する高品質な基準画像との類似度として定義する指標で，値が大きいほど高品質であることを表す．画像の解像度を $m \times n$，基準画像を K とすると，画像 I の PSNR は，平均二乗誤差 MSE を用いて式 (4.1) のように定義される．

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(I(i,j) - K(i,j))^2$$
$$PSNR = 20\log_{10}\frac{MAX_I}{\sqrt{MSE}} \tag{4.1}$$

ただし，MAX_I はデータのとりうる最大値である．

本稿では，オクルージョン推定の結果を評価するために，深度マップに関する PSNR を算出した．PSNR は点群の描画品質の評価に用いられている例 [35] があり，また，深度マップの品質の評価に用いられている例 [45] がある．それらの研究を参考に，本研究でも品質の指標として PSNR を用いた．

4.1.3　メッシュと基準画像の作成

点群から作成したメッシュを用いて理想的なオクルージョン推定を行い，その結果を補間することで PSNR の計算に用いる基準画像を作成した．図 4.3 に基準画像の作成方法の概要を示す．可能な限り公平な基準画像を生成するため以下のようにメッシュを作成した．

まず，CloudCompare を用いてノイズ除去のための SOR（Statistical Outlier Removal）フィルタを点群に適用し，次に，MeshLab を用いて法線の計算を行った．メッシュの生成は，3D データ処理用のライブラリである Open3D [46] により，ball pivoting algorithm（BPA）[47] を適用することで行った．BPA で使用されるパラメータであるボール半径 ρ

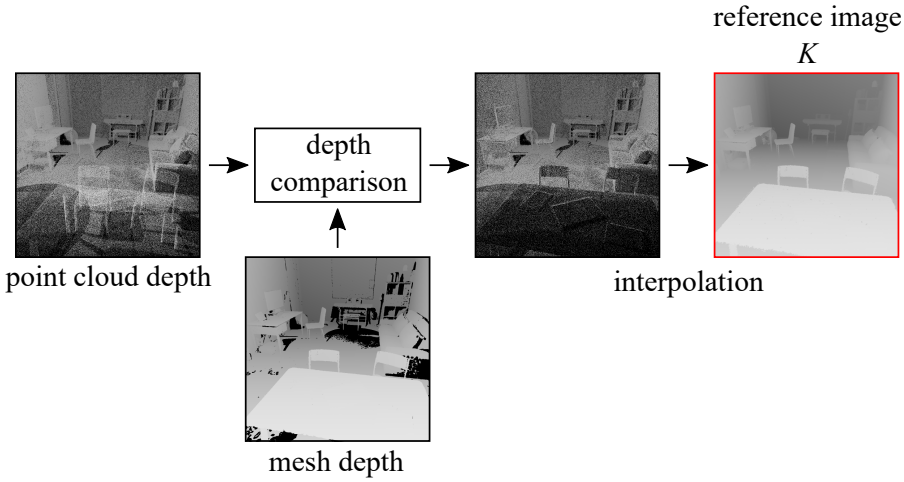


図 4.3　基準画像の作成方法の概要

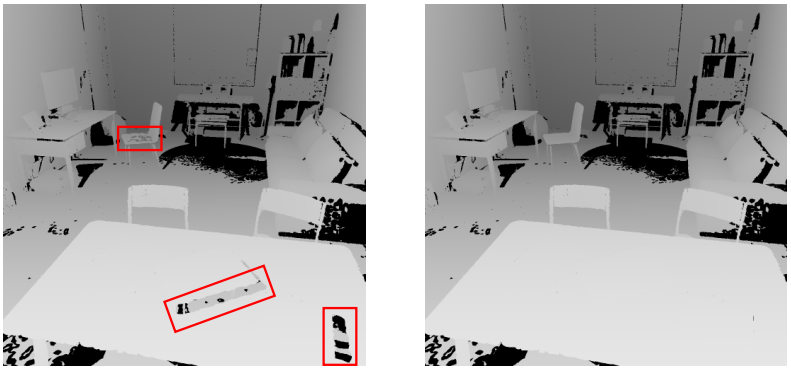


図 4.4　メッシュの修正例

は適切に設定する必要があり，本研究では ρ を変化させて複数のメッシュを生成し，適切であると考えられる ρ の値を主観で 4 個選択した．その後，それぞれの値で生成されるメッシュを統合して，最終的なメッシュデータとした．また，処理時間の関係から，全体を格子状に分割し，それぞれの部分についてメッシュの生成を行った．分割の際に，格子よりも少し大きく切り出すことで隣接する部分に重なりを生じさせ，後述する正解データの作成に影響が出ないようにした．

パラメータ調整だけでは対応できない部分は，Blender[4]を用いて手作業によるメッシュの修正を行った．修正の際は，明らかに不必要なメッシュの削除と，既存の頂点に対して不足しているメッシュの追加のみを行い，頂点の移動や追加，削除は行っていない．メッシュの修正例を図 4.4 に示す．修正を行った箇所の例を赤枠で示した．

次に，オクルージョン推定の精度を判定するための正解データを構築したメッシュから作成した．各点 i について，深度値 $z_p[i]$ とメッシュをレンダリングした画像の点 i に対応する画素の深度値 $z_m[i]$ を比較し，$z_p[i] - z_m[i] > \epsilon$ であれば点 i は遮蔽されている点とし，そうでなければ遮蔽されていない点としてラベル付けした．なお定数 ϵ は 0.005 とした．各視点における基準画像は，高密度な点群を投影した深度マップに対してオクルージョン推定の正解に従って除去処理を行った結果を，推定時と同様に補間することで作成した．

4.1.4　フレームレートの評価

シーン (A1) において 1000 回の描画を実行するのに要する時間を計測し，1 秒間あたりの描画の実行回数（フレームレート）の平均値を算出した．

[4]https://www.blender.org/

4.1.5　実験環境

実装は Unity 2020.1.14f1 を用い，GPU による並列処理にはコンピュートシェーダを用いた．レンダリングの解像度は 1024×1024 と設定し，実行は 3.40 GHz Intel Core i7-6700 CPU，NVIDIA GeForce GTX 1660 Ti を用いた環境で行った．

4.2　密度を用いた近傍領域サイズの決定手法の評価

4.2.1　結果

既存手法 [9] と 3.1 節で述べた点の密度によって近傍領域サイズを決定する提案手法について，各シーンでパラメータ p, p' を変化させたときの PSNR を図 4.5 に示す．ただし，横軸のスケールを揃えて比較するため，各手法のシーン (A1) における PSNR のピーク位置 $(= p_0,\ p'_0)$ を用いて，それぞれ $\hat{p} = p/p_0,\ \hat{p}' = p'/p'_0$ で正規化した値を横軸として設定した．また，PSNR は各シーン，手法における最小値が青，最大値が赤となるようにカラーマッピングして示した．既存手法 [9] では，異なるシーン間でピーク位置が異なる場合があるが，本手法ではピークの位置はどのシーンでも $\hat{p}' = 1$ 付近になっている．例として，高密度なシーン (A1) と低密度なシーン (A2) におけるレンダリング結果を図 4.6 に示す．既存手法では，$\hat{p} = 1$ の時にシーン (A1) の品質が最大となる一方で，シーン (A2) ではテーブル部分が透け，品質が低下している．また，$\hat{p} = 3.125$ 付近ではシーン (A2) のテーブル部分の品質は改善され，画像全体の品質も最大となるが，シーン (A1) では赤枠で示す箇所にノイズが生じている．一方，提案手法では，密度の異なる二つのシーンに対して同じ値のパラメータ $\hat{p}'$ を用いているが，それぞれのシーンで高品質な結果が得られている．図 4.5 の結果を参考に各シーンの品質を確認し，本手法で用いるパラメータの値を $\hat{p}' = 1.2$ と設定した．

次に，PSNR の比較を行った結果を図 4.7 に示す．比較の際の基準は，各シーンに本手法（$\hat{p}' = 1.2$）を適用したときの PSNR の値とした．図 4.7 に基準としたパラメータの値をバツ印として示した．また，横軸は図 4.5 と同様に正規化されたパラメータである．比較結果は，文献 [48] などで用いられている，PSNR が 0.2 dB 程度異なると主観的にも違いが分かるという目安に従い，より高品質（基準より 0.2 dB 以上大きい），同程度の品質（基準との差の絶対値が 0.2 dB 以内），より低品質（基準より 0.2 dB 以上小さい）の 3 種類に分類して示した．シーン (A2)-(A6)，(B4) では，既存手法でパラメータ調整を行って

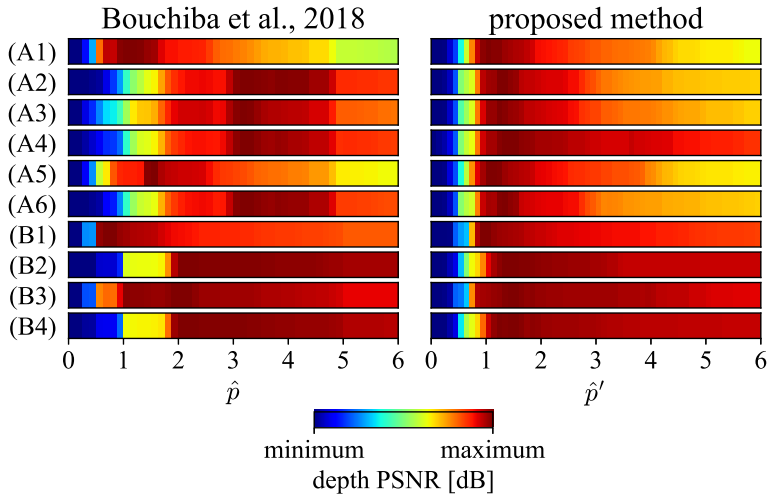


図 4.5　各手法，シーンにおける PSNR

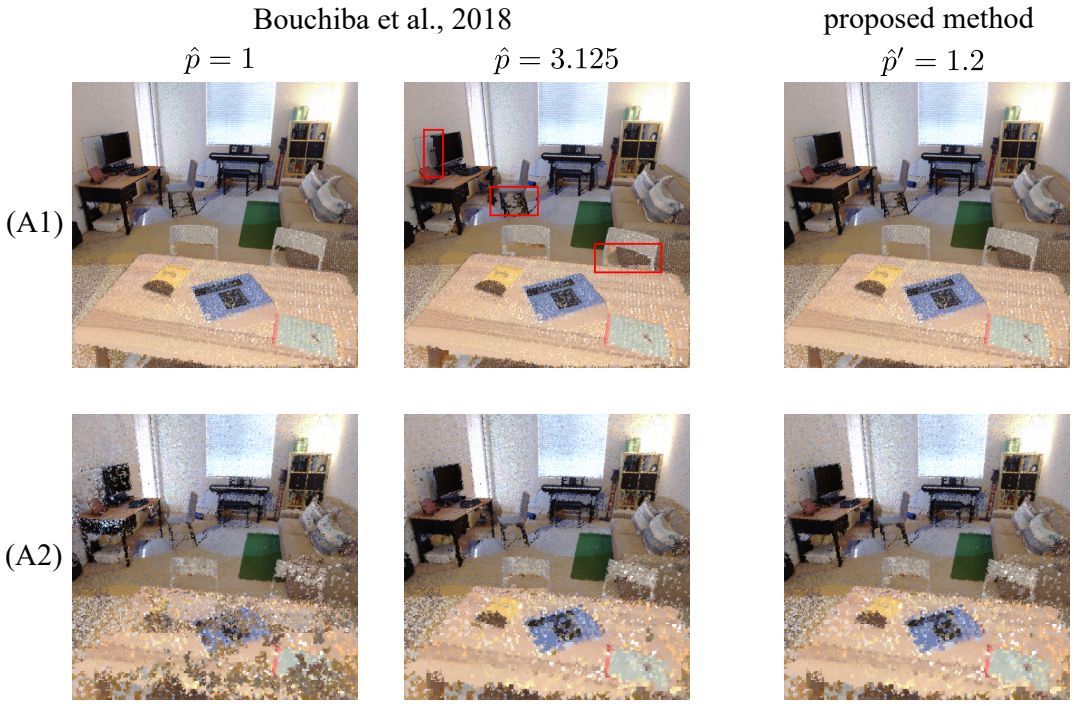


図 4.6　シーン (A1)，(A2) におけるレンダリング結果

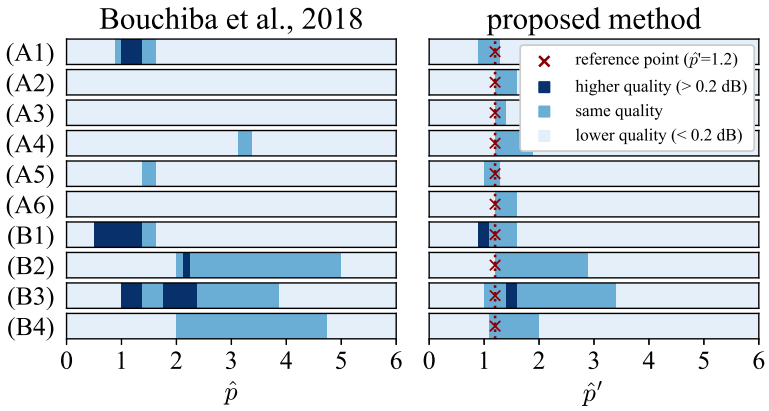


図 4.7　各手法，シーンにおける PSNR の比較

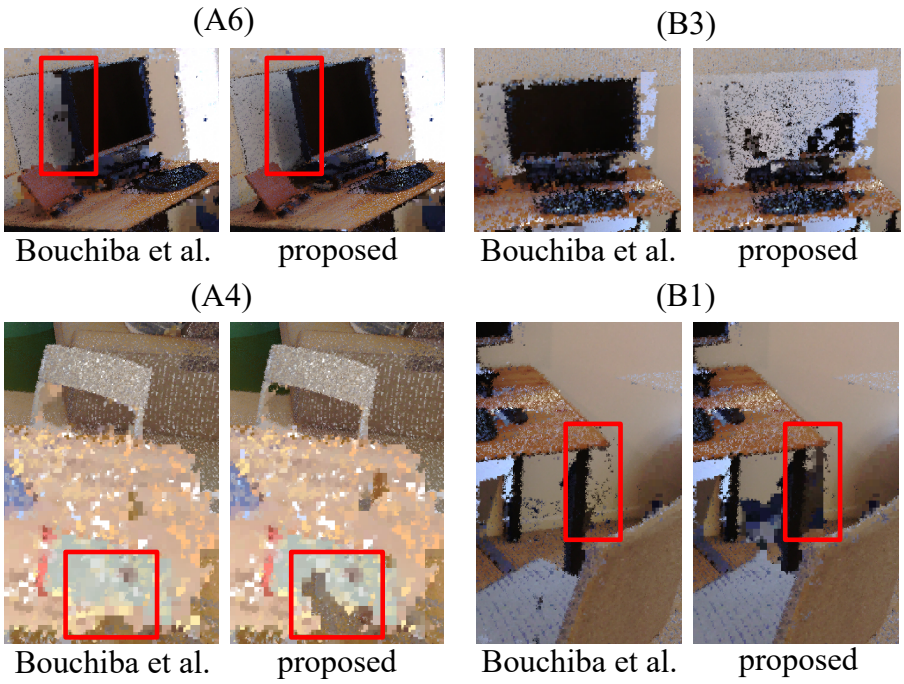


図 4.8　レンダリング結果の比較

PSNR を最大化した場合と同程度，あるいは，より高品質な結果が，パラメータを固定した提案手法で得られている．

図 4.8 に既存手法で PSNR が最大となるレンダリング結果と，提案手法によるレンダリング結果の一部を示す．提案手法の方が高品質な結果が得られるシーン (A6) では，赤枠で示す箇所に着目すると，提案手法の結果では物体の境界付近のブロックノイズが少ないよ

図 4.9　シーン (B3) におけるしきい値 e のレンダリング結果への影響

うに見える．これは，特に点の密度が不均一な場合，深度を用いるよりも密度を用いた方が密度がより適切な近傍領域サイズが求まる場合があるためだと考えられる．

一方で，図 4.7 を見ると，シーン (A1)，(B1)-(B3) のように，調整を行った既存手法よりも提案手法の方が低品質となる場合もある．例えばシーン (B3) では，図 4.8(B3) のように，提案手法でディスプレイが透けていることが分かる．原因の一つとして，密度の計算時にしきい値 e による手前側の点の区別が適切にできておらず，ディスプレイ後方の壁部分の点も手前の点として数えられ，密度が正しく計算できていないことが考えられる．図 4.9 にしきい値 e を $e = 0.01$ (10 cm) に設定した場合のレンダリング結果を示す．ディスプレイ部分が透けて見えている状態が改善されており，これは，しきい値 e が小さくなったことによってディスプレイ部分の点のみが適切に密度の計算に使用されたためだと考えられる．したがって，本手法で用いたしきい値 e の適切な値は，シーンによって異なる場合があると考えられる．また，シーン (A4) では，図 4.8(A4) に赤枠で示すように，提案手法ではテーブルの一部に穴が開いている．原因の一つとして，密度を計算する際のグリッドのサイズが不十分で，テーブル部分の点が含まれないグリッドが存在し，密度が正しく計算できない場合があることが考えられる．またシーン (B1) では，図 4.8(B1) に赤枠で示すように，提案手法で机の脚の周りに黒いノイズが目立っている．点群を確認すると，図 4.10 のように，画面上のノイズの位置には点群自体のノイズと思われる点が存在しており，このようなノイズの点が密度の計算に影響し，提案手法の品質低下につながったと考えられる．

また，本稿では提案手法のパラメータを $\hat{p}' = 1.2$ としたが，シーン (B1)，(B3) ではより高品質な結果が得られるパラメータ $\hat{p}'$ が存在する．よって，ピークの位置は厳密には揃っ

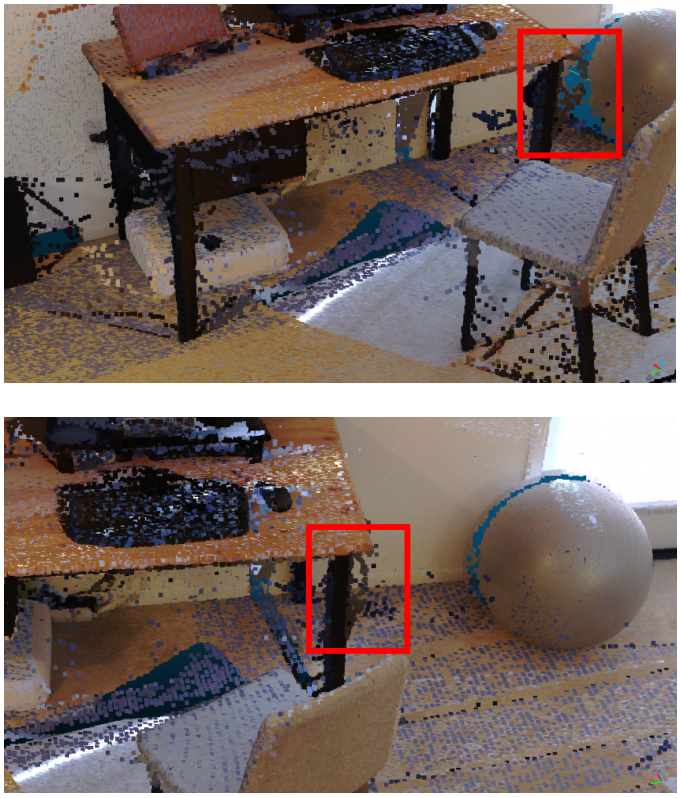


図 4.10　CloudCompare を用いて各点を正方形として点群を可視化した様子

ておらず，原因としては，上記のような品質を低下させる要因によってシーンによって少しずつピーク位置が異なっているということが考えられる．

シーン (A1) におけるフレームレートは，既存手法（$\hat{p} = 1$）は 209 FPS（Frames per second）であり，本手法は 187 FPS であった．

4.2.2　考察

前節の評価実験の結果より，既存手法でパラメータ調整を行った結果と同程度，あるいは，それ以上の品質の結果が本手法によって得られる場合がある事が確認された．よって，点の密度を考慮することで密度に応じたパラメータ調整に依存せずにレンダリング品質の改善を行うことが可能となり，本手法の有効性が示された．

一方，しきい値 e の適切な値がシーンによって異なる場合や，グリッドのサイズが不十分な場合があるなど，特定のシーンではレンダリング品質に改善の余地がある．したがって，本手法によって品質が低下してしまう条件の調査や，密度計算の改善手法の検討，ノイ

ズへの対策手法の検討が必要であると考えられる．また，本研究では実験に使用するシーンの全体的な品質を確認してパラメータ $\hat{p}'$ を決定したが，未知のシーンに適用した際の評価や，より多様なシーンでレンダリング品質の改善を可能とするため，最適なパラメータに自動調整するアプローチの検討も必要だと考えられる．

また，本手法のフレームレートは既存手法のものよりも低下するが，より高性能なハードウェアによるフレームレートの改善などが可能であることを考慮すると，十分に実用的な値であると考えられる．ただし，密度計算の近似や並列化など，アルゴリズムの工夫によるフレームレートの改善の余地はあると考えられる．

4.3　深度値の勾配を用いた近傍領域サイズの補正手法の評価

4.3.1　深度値を用いて決定した近傍領域サイズの補正結果

シーン (A1) において，既存手法 [9] を用いて深度値から計算した近傍領域サイズの段階数 $l[i]$ を，3.2 節で述べた提案手法により補正した場合について実験を行った．図 4.11 に勾配しきい値 g_{th} 変化させた時の PSNR を示す．$g_{\mathrm{th}} = 0.05$ 付近において，本手法の PSNR の方が既存手法よりも大きくなっており，勾配しきい値を適切に設定することで，より高品質な結果が得られると考えられる．

図 4.12 にクルージョン推定の結果を示す．図 4.12 の赤い点は誤ったオクルージョン推定がされた点を示す．赤枠で示した箇所に着目すると，既存手法で目立っている物体の境

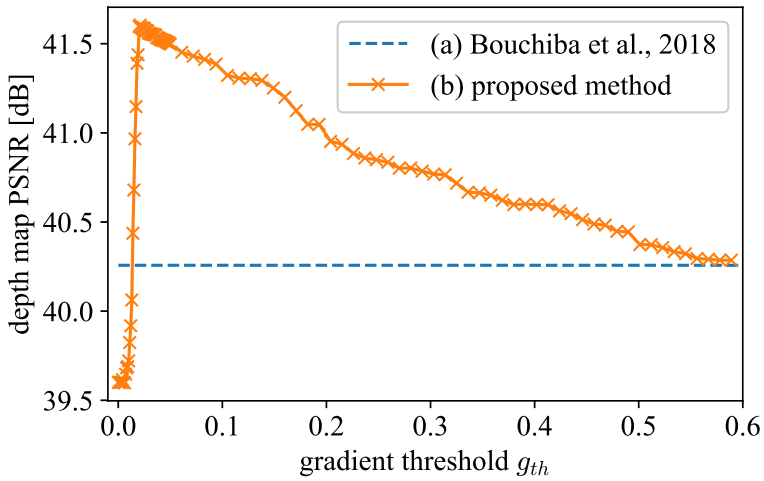
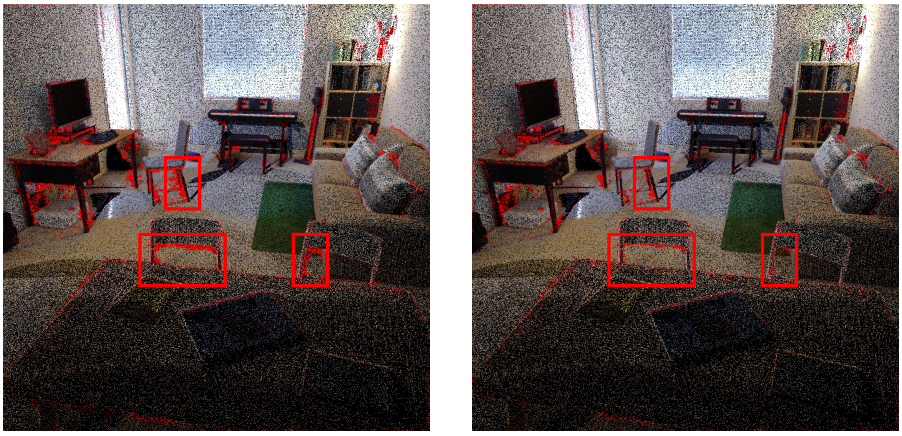


図 4.11　勾配しきい値 g_{th} と PSNR の関係



Bouchiba et al., 2018　　proposed method (g_{th} = 0.05)

図 4.12　除去処理の結果

Bouchiba et al., 2018　　proposed method (g_{th} = 0.05)

図 4.13　補間処理の結果

界付近での誤りが，本手法で改善されていることが分かる．図 4.13 に示す補間処理後の結果でも，推定の誤りが原因だと考えられる不自然なノイズが減少していることが分かる．

また，補正前の既存手法と補正を適用した際のオクルージョン推定の不正解数の割合を表 4.2 に示す．False Occluded は見えているにも関わらず遮蔽されていると推定する誤り，False Visible は，遮蔽されているにも関わらず見えていると推定する誤りである．提案手法では，過剰な除去の原因となる False Occluded を低減できている．一方で，その減少幅より小さい幅ではあるが，False Visible が増加している．

また，フレームレートは既存手法が 209 FPS，提案手法が 203 FPS であり，高速な動作

表 4.2　オクルージョン推定の不正解の割合

	既存手法 [9]	補正あり ($g_{\mathrm{th}} = 0.05$)
False Occluded	3.07 %	2.18 %
False Visible	0.22 %	0.25 %

が可能であるといえる．以上より，深度値の勾配を用いた提案手法を用いて，既存手法で深度値から計算した近傍領域サイズを補正することでレンダリング結果の品質を改善できる場合が確認できた．

4.3.2　密度を用いて決定した近傍領域サイズの補正結果

3.1 節で述べた提案手法を用いて点の密度から計算した近傍領域サイズの段階数 $l[i]$ を，3.2 節で述べた提案手法を用いて補正した場合について実験を行った．実験には高密度な点群を用いたシーン (A1)，(B1)，低密度な点群を用いたシーン (A2)，(B2) の 4 つのシーンを用いた．図 4.14 に各シーンについて勾配しきい値 g_{th} を変化させた時の PSNR を示す．高密度なシーン (A1)，(B1) の場合は，4.3.1 節と同様に，勾配しきい値 g_{th} を適切に設定することで補正前より高品質な結果が得られる．図 4.15 にクルージョン推定の結果を示す．図 4.15 の赤い点は誤ったオクルージョン推定がされた点を示す．4.3.1 節の結果と同様に，赤枠で示す箇所に着目すると，補正前に目立っている物体の境界付近での誤りが補正により改善されていることが分かる．図 4.16 に示す補間処理後の結果でも，推定の誤りが原因だと考えられる不自然なノイズが減少していることが分かる．以上より，密度を用いて計算した近傍領域サイズを補正することでレンダリング結果の品質を改善できる場合が確認できた．一方で，図 4.14 に示すように，低密度なシーン (A2)，(B2) の場合はどのような値の勾配しきい値でも品質の改善には至っていない．

また，それぞれのシーンおいて両手法を適用した際のオクルージョン推定の不正解数の割合を表 4.3-4.6 に示す．4.3.1 節と同様に，False Occluded は見えているにも関わらず遮蔽されていると推定する誤り，False Visible は，遮蔽されているにも関わらず見えていると推定する誤りである．ただし，勾配しきい値 g_{th} は，図 4.17 に示すように，各シーンにおいて適切に物体の境界が抽出される値に設定した．高密度なシーンであるシーン (A1)，(B1) では，それぞれ表 4.3，表 4.4 のように，勾配を用いた補正によって過剰な除去の原因となる False Occluded が減少している．一方で，その減少幅より小さい幅ではあるが，False Visible が増加している．図 4.16(B1) のように，シーン (B1) において補正を適用し

た場合にディスプレイ部分に透けて見える点が増える原因は，増加した False Visible であると考えられる．

一方で低密度なシーンであるシーン (A2), (B2) では，それぞれ表 4.5，表 4.6 のように，高密度なシーンの場合と比較して補正前の False Visible が多い．補正によって False Occluded は低減されているが，False Visible の増分も大きく，全体として高密度なシーンの場合よりも効果は小さかったと考えられる．

また，フレームレートは補正を適用しない場合が 187 FPS，補正を行った場合が 185 FPS であった．

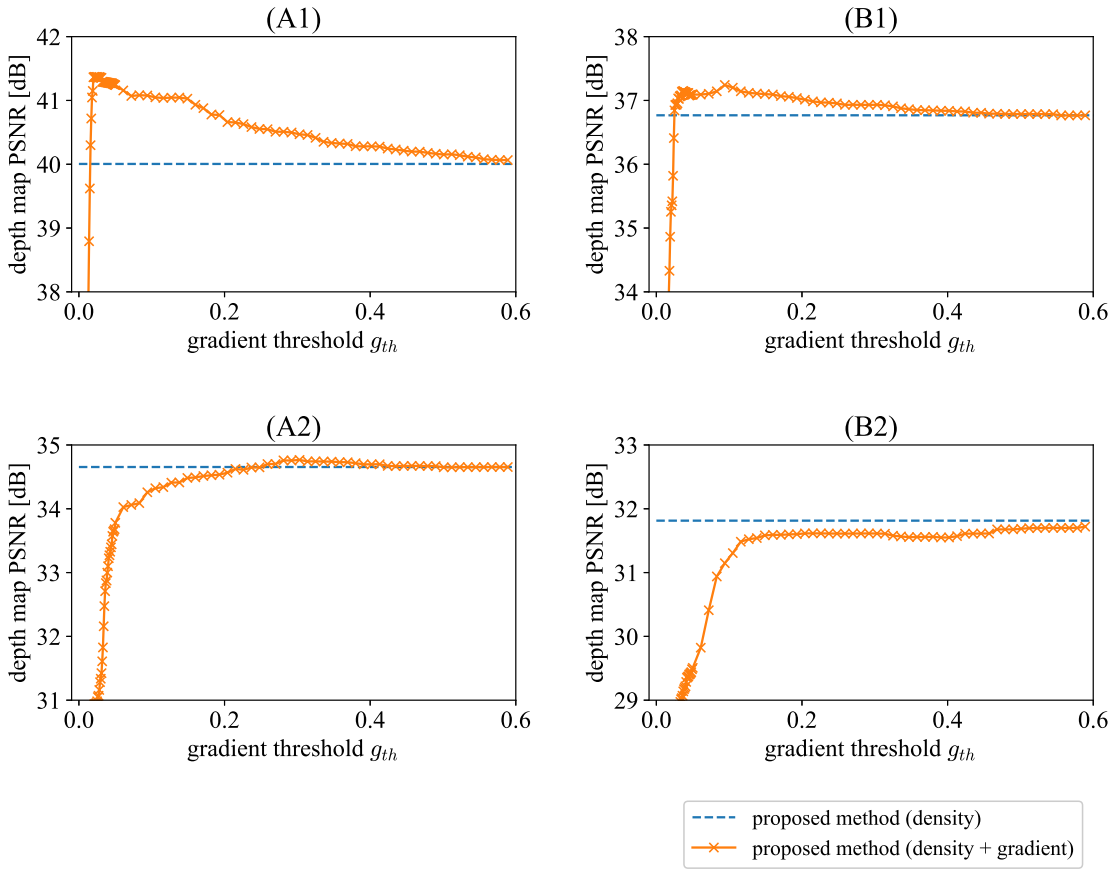


図 4.14　勾配しきい値 g_{th} と PSNR の関係

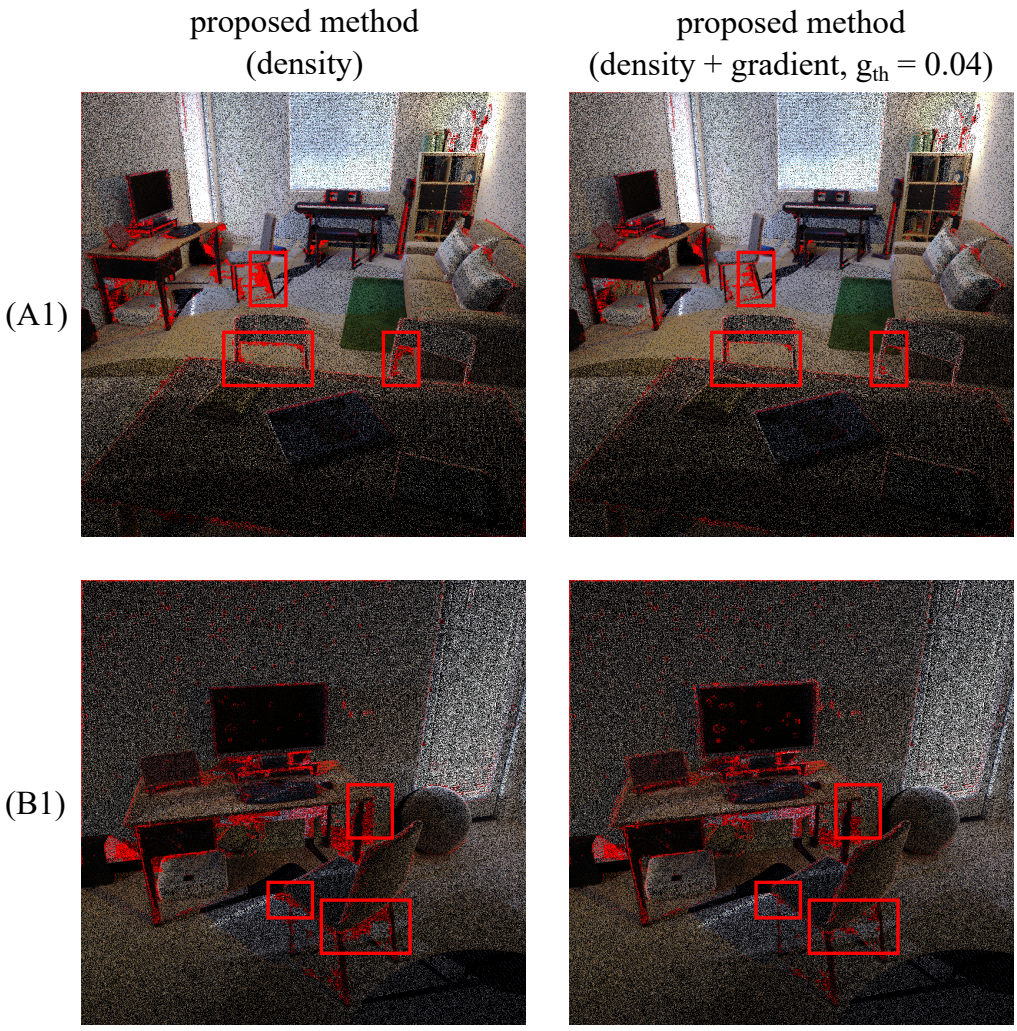


図 4.15　除去処理の結果

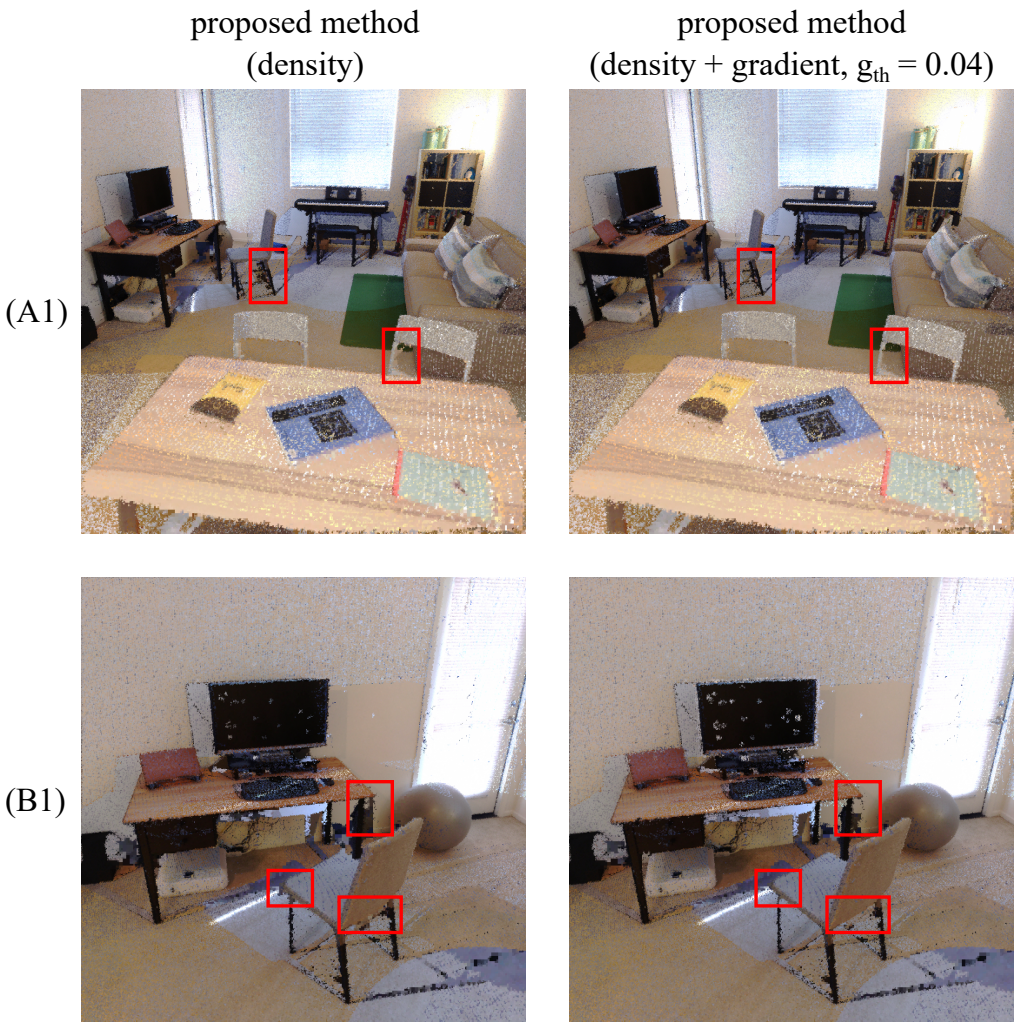


図 4.16　補間処理の結果

表 4.3　シーン (A1) におけるオクルージョン推定の不正解の割合

	補正なし	補正あり ($g_{\mathrm{th}} = 0.04$)
False Occluded	3.16 %	2.18 %
False Visible	0.23 %	0.27 %

表 4.4　シーン (B1) におけるオクルージョン推定の不正解の割合

	補正なし	補正あり ($g_{\mathrm{th}} = 0.04$)
False Occluded	3.45 %	2.58 %
False Visible	0.7 %	1.09 %

表 4.5　シーン (A2) におけるオクルージョン推定の不正解の割合

	補正なし	補正あり ($g_{\mathrm{th}} = 0.1$)
False Occluded	2.94 %	1.87 %
False Visible	1.71 %	2.27 %

表 4.6　シーン (B2) におけるオクルージョン推定の不正解の割合

	補正なし	補正あり ($g_{\mathrm{th}} = 0.1$)
False Occluded	2.15 %	1.64 %
False Visible	3.01 %	3.71 %

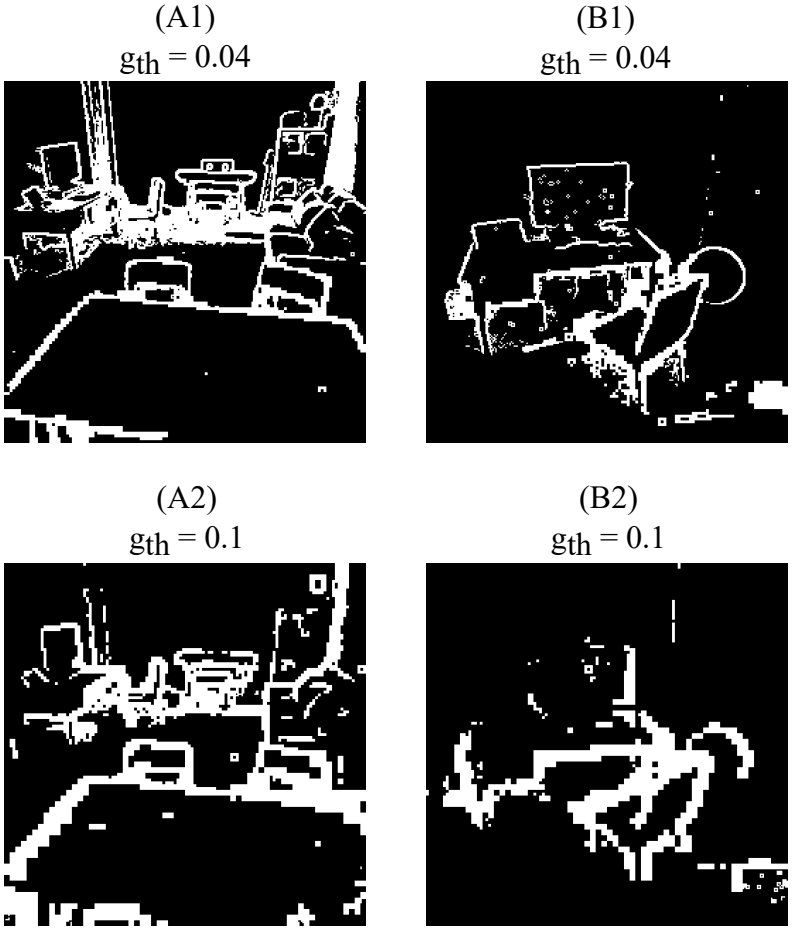


図 4.17　物体境界の抽出結果

4.3.3　考察

高密度なシーン (A1)，(B1) においては，実験によって確認された PSNR の増加と False Occluded の減少より，オクルージョン推定による過剰な除去が補正によって改善し，結果として補間処理結果の品質が改善されたと言える．補正によって False Occluded が減少する一方で False Visible が増加したが，減少分と比較すれば小さな増加であり，全体としての品質は改善されたと考えられる．また，勾配の計算はフレームレートに大きく影響しなかったことから，十分高速な手法であると言える．

一方で低密度なシーン (A2)，(B2) では補正前の False Visible の割合が高密度なシーンの場合よりも多く，補正適用後の False Visible の増分も大きかった．これは，点が全体として少ないため，オクルージョン推定の際に見えていると推定されやすい傾向があり，本節で提案した近傍領域サイズを小さくする補正によって，見えていると推定される点がさらに増えたためだと考えられる．また，補正の効果が小さくなった要因として，図 4.17 に示すように，点の密度が低い場合には抽出される物体境界の解像度も低くなることも考えられる．したがって，本節の勾配を用いた補正手法は，点の密度が十分に大きいときに有効であると考えられ，補正が有効であるための具体的な条件の調査が必要であると考えられる．

また，図 4.11，図 4.14 を見ると，勾配しきい値 g_{th} が 0 付近だと補正前の PSNR を下回り，0.6 付近まで大きくすると補正前とほぼ同等になっている．これは，勾配しきい値を小さくするほど，境界以外も含めた多くの箇所の近傍領域が小さくなるため，全体的に推定が不正確になるためだと考えられる．また，勾配しきい値を大きくするほど，補正される箇所が少なくなり補正前と等価になるため，PSNR が同等になっていると考えられる．つまり，本手法で品質の改善を行うためには，勾配しきい値を適切に設定する必要がある．密度が十分に大きい場合でも，PSNR が最大となる勾配しきい値はシーン (A1)，(B1) で異なるため，より多様なシーンでの適用を可能とするために，しきい値の定め方の検討や，最適なしきい値が変化する原因の調査が必要であると考えられる．

第5章　提案手法の改良のための検討

5.1　オブジェクト ID を用いた密度の算出

第 3 章で述べた提案手法では，オクルージョン推定に各点の座標のみを用いていた．ここで，文献 [11,31] で提案されているシステムのように，例えば，オブジェクトごとに別々に点群のキャプチャを行い，システム上でそれらの点群を用いて一つのシーンを構成する場合を考えると，各点がどのオブジェクトに属するかという情報を利用できるため，オクルージョン推定がより簡単になると考えられる．本研究では，各点がどのオブジェクトに属するかという情報を，オブジェクト ID と呼ぶ．本節では，各点の座標に加えてオブジェクト ID が利用できる場合を想定し，オブジェクト ID を用いたオクルージョン推定手法を検討した．

5.1.1　方法

本研究では，オブジェクト ID を用いて 3.1.1 節で述べた点の密度の計算の改善を行うことを検討した．3.1.1 節で述べた方法では，視点に対して手前側に存在する点の密度を計算するため，各グリッド内における最小の深度値 $z_{\min}$ を基準として，定数 e を用いて，深度値 $z[i]$ が，式 (5.1) を満たす点 i を数えることで密度を計算した．

$$z[i] - z_{\min} < e \tag{5.1}$$

ここで，オブジェクト ID が利用できれば，手前側に存在するオブジェクトに属する点のみを数えることで，手前側の点の密度をより正確に計算できると考えられる．本研究では，各点 i に対してオブジェクト ID $\mathrm{id}[i]$ が定義され，同じオブジェクトに属する点には同じ ID が与えられ，異なるオブジェクトに属する点には異なる ID が与えられていることを想定した．このようなオブジェクト ID $\mathrm{id}[i]$ を用いて，各グリッド内における深度値が最小の点のオブジェクト ID を $\mathrm{id}_{\mathrm{zmin}}$ として，式 (5.1) を満たしていて，かつ，以下の式 (5.2) も満たす点 i を数えて密度を算出することを検討した．

$$\mathrm{id}[i] = \mathrm{id}_{\mathrm{zmin}} \tag{5.2}$$

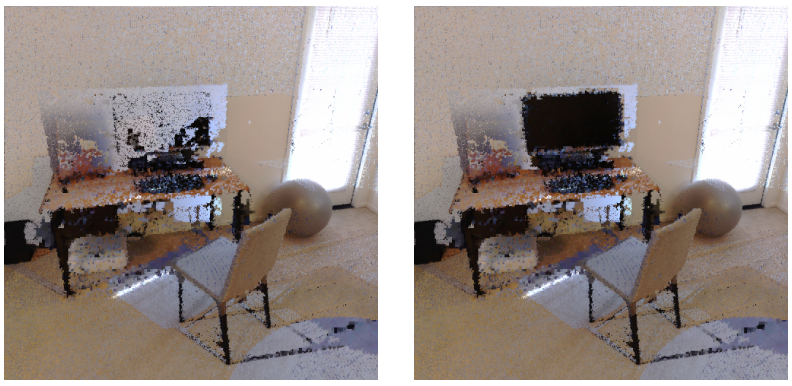
5.1.2　実験

第 4 章で述べたように，シーン (B4) では提案手法を適用するとレンダリング品質が低下する．本節では，オブジェクト ID を用いた密度計算によってシーン (B4) におけるレンダリング品質が改善されることを確認する実験を行った．実験のため，まず，CloudCompare を用いて，使用した点群について，ディスプレイの部分とそれ以外の部分に手動でセグメンテーションを行った．次に，セグメンテーションを行ったディスプレイ部分の点群と，それ以外の部分の点群にそれぞれ別のオブジェクト ID を与え，オブジェクト ID を用いた手法を適用し，レンダリング品質の評価を行った．また，第 4 章と同じく，$e = 0.04$ とした．

5.1.3　結果

レンダリング結果を図 5.1 に示す．オブジェクト ID を使用しない場合，ディスプレイ部分が透けてしまうのに対して，オブジェクト ID を使用した場合，透けることなくレンダリングされている．また，PSNR は 35.73 dB から 36.28 dB に改善している．

また，シーン (B4) におけるフレームレートは，オブジェクト ID を使用しない場合は 220 FPS，オブジェクト ID を使用した場合は 129 FPS であった．フレームレートが低下した原因の一つとして，実装の簡略化のため，通常のレンダリング処理に加えて，各点のオブ



(a) Without Object ID (35.73 dB)　(b) With Object ID (36.28 dB)

図 5.1　オブジェクト ID を用いた手法によるレンダリング結果と PSNR

ジェクト ID を投影したマップの取得するためにレンダリング処理を再度行っていることが考えられる．

5.1.4　考察

オブジェクト ID が利用できるという限定的な状況ではあるが，オブジェクト ID を使用することでレンダリング品質の改善が可能であることが確認できた．レンダリング品質が改善した要因として，本節で検討した手法により，オブジェクト ID を用いることで密度の計算に用いる手前側の点が適切に選択できるようになったことが考えられる．また，ディスプレイ部分以外の点に関しては異なるオブジェクトに対しても全て同一のオブジェクト ID を与えているが，レンダリング品質が低下するようなことはない．これは，ディスプレイ部分以外の点については同一の ID を持つため，常に式 (5.2) が満たされ，第 3 章で述べた提案手法と同じ処理が行われるためであると考えられる．よって，本節で検討した手法は，一部においてのみオブジェクト ID を用いた密度計算が可能な場合でも適用可能な手法であると考えられる．また，本節ではレンダリング結果を確認することを目的として簡略化した実装を行ったが，実際のアプリケーションで使用するためには，実装を工夫してフレームレートを改善する必要があると考えられる．

5.2　深度値のヒストグラムを用いた密度の算出

第 3 章で述べた密度の計算手法では，密度の計算に用いる点を区別するためのしきい値 e を小さく設定すると，図 5.5(B3)，(A1) のように，レンダリング結果にノイズが発生する場合があった．比較のため，ノイズが発生する範囲をそれぞれのレンダリング結果に赤枠で示す．本節ではこの問題を解決することを目指し，密度計算の改善手法の検討を行う．本節では，5.1 節のようなオブジェクト ID などは用いず，既存研究と同様に各点の座標のみを用いる手法を検討した．

5.2.1　深度分布の分析

改善手法の検討のため，シーン (B3) における結果に着目し，点を投影した画像について，いくつかのグリッド内の背景を含む $16 \times 16 = 256$ 点の深度値のヒストグラムを作成し分析を行った．図 5.2 に着目したグリッドの位置，図 5.3 に各グリッドのヒストグラムを示す．5.1 節でセグメンテーションしたデータを用いて，ディスプレイ部分の点をオレン

ジ色のバーで示した．また，赤い点線は，それぞれしきい値 e が 10 cm, 40 cm の場合に視点に対して手前側の点と判定される範囲を示し，この範囲内に含まれる点が密度の計算に用いられる．図 5.2(1) のディスプレイ部分のグリッドに着目すると，図 5.3(1) のように，$e = 40$ cm の場合は，密度の計算に用いられる点にディスプレイ部分ではない点が含まれている．このため，密度が大きく見積もられ近傍領域のサイズが小さくなり，正確なオクルージョン推定ができないことでディスプレイ部分が透けて見えると考えられる．一方で，$e = 10$ cm の場合には，ディスプレイ部分の点のみが密度の計算に用いられる点となっており，透けて見える部分も改善されている．

また，図 5.2(2) のように椅子の周りのノイズに着目すると，$e = 10$ cm の場合には，少数の深度値が小さい点が存在することにより，大部分の点が密度の計算に使用されなくなっている．このため，計算される密度が不正確になりオクルージョン推定の誤りに繋がり，レンダリング結果にノイズが生じていると考えられる．レンダリング結果のノイズの原因となる少数の深度値が小さい点は，点群自体に含まれるノイズであると考えられる．

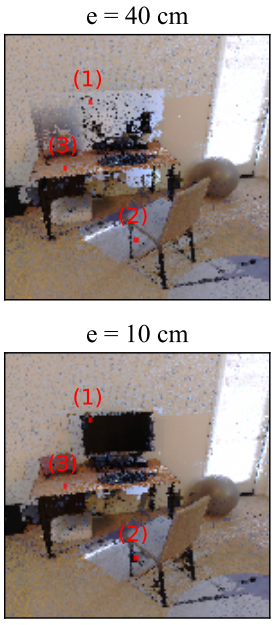


図 5.2　着目したグリッド位置

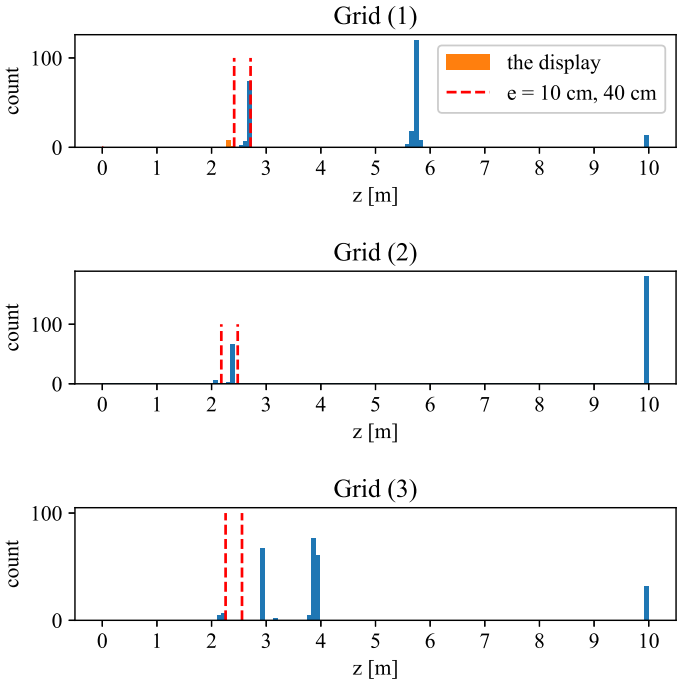


図 5.3　各グリッドにおける深度値のヒストグラム

5.2.2　方法

以上の分析により，深度値のヒストグラムを用いることで，少数の深度値の小さい点をノイズとして無視するような処理が可能であると考えられる．また，深度値のヒストグラムを観察すると，図 5.3(3) のように，密度の計算に考慮すべき手前側の点とそれ以外の点の間には度数がゼロの区間が存在する場合が多いことが分かった．よって，ヒストグラム上で度数がゼロの区間を検出することで，しきい値 e を用いることなく手前側の点の判定が可能となると考えられる．そこで本節では，各グリッドについて，深度値のヒストグラムを作成し，そのヒストグラムを深度値の小さい区間から順に見て，度数がしきい値 n_{th} より大きい連続する区間を見つけ，その区間の度数の合計から密度を算出するという手法を検討した．例えば，図 5.4 のようなヒストグラムが与えられた場合，灰色で示す区間はノイズとして無視され，オレンジ色で示す区間に含まれる点が視点に対して手前側の点と判定され，密度の計算に用いられる．

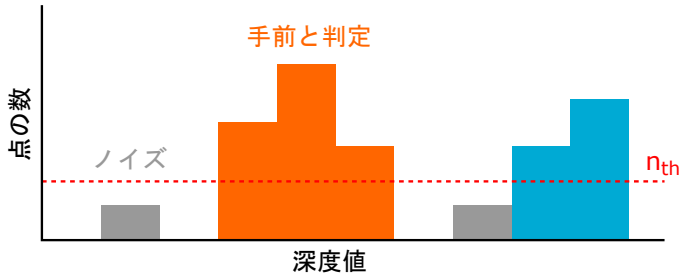


図 5.4　深度値のヒストグラムを用いた密度計算に用いる点の選択

5.2.3　結果

図 5.5 に各シーンにおけるレンダリング結果と PSNR を示す．ただし，ヒストグラムの区間の幅は 5 cm，$n_{\mathrm{th}} = 3$ とした．シーン (B3) に着目すると，e を小さくした場合よりも本節で検討したヒストグラムを用いた手法の方が PSNR の値が大きく，レンダリング結果でもノイズが目立たないように見える．またシーン (A1) では，e を小さくするとノイズの影響で PSNR が低下する一方で，ヒストグラムを用いた手法では PSNR が低下しない．しかし，テーブル部分を低密度に設定したシーン (A4) ではレンダリング品質が大きく低下している．これは，テーブル部分の点の密度が低いことに加え，視点の近くに存在している

ため，画像空間上での点の密度も低くなっており，誤ってノイズであると判定されているためだと考えられる．また，シーン (A1) において，第 3 章で述べたしきい値を用いた手法のフレームレートは 186 FPS，本節で検討したヒストグラムを用いた手法のフレームレートは 183 FPS であった．

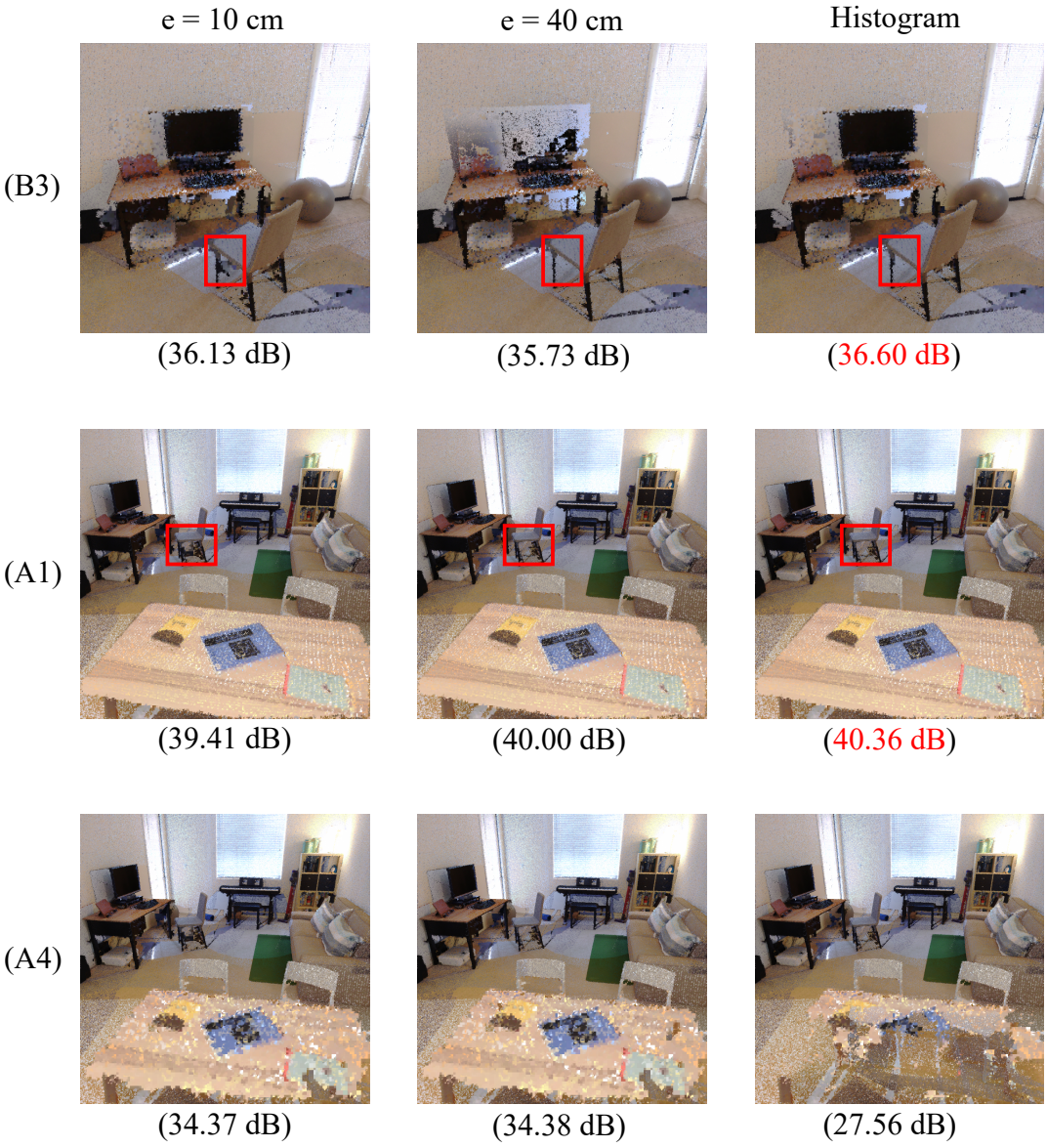


図 5.5　しきい値 e の調整結果とヒストグラムを用いた手法のレンダリング結果

5.2.4　考察

深度値のヒストグラムを用いて密度を計算することにより，しきい値 e を調整した時よりも高品質なレンダリング結果が得られる場合があることを確認し，本節で検討した手法が有効であるケースが確認できた．また，フレームレートはしきい値を用いて判定する場合とほぼ変わらず，高速な動作が可能であった．一方で，特に点の密度が小さいときに品質が大きく低下してしまう場合があることも確認され，第 3 章で述べた手法と同様に，特定のシーンでは品質が低下するという課題がある．本節で検討した手法は，ノイズは低密度であることを仮定してしきい値 n_{th} を用いた判定を行っているため，ノイズでない部分の点の密度は十分に大きい必要があると考えられる．品質の改善が可能となる条件をより明確にするためには，点の密度やノイズの量といった条件を定量的に評価していく必要があると考えられる．

第6章　結論

6.1　結論

本研究では，点群の高速かつ高品質なレンダリング手法において，点の密度に応じたパラメータ調整を行うことなくレンダリング品質の改善を可能とするため，画像空間における局所的な点の密度を考慮したオクルージョン推定を提案し，有効性を評価した．評価実験では，既存手法でパラメータ調整を行った結果と同程度，あるいは，それ以上の品質の結果が本手法によって得られる場合があることが確認され，本手法の有効性が示された．さらに，物体境界でのオクルージョン推定の誤りを改善するため，深度値の勾配を考慮した推定手法を提案した．評価実験から，物体境界に着目することでオクルージョン推定の誤りが低減し，レンダリング品質が改善される場合があることが確認された．また，画像空間における局所的な点の密度や深度値の勾配の計算のためにフレームレートは低下するもののインタラクティブな動作は可能であり，点群の高速かつ高品質なレンダリング手法として本手法が有効であることが確認された．

6.2　今後の課題

評価実験により，特定のシーンでは本手法によってレンダリング品質が低下する場合が確認された．また，本研究で提案した点の密度の計算に用いるパラメータや，勾配を考慮する際のパラメータなどの最適な値がシーンによって異なる場合があることも確認された．したがって，本手法によって点の密度に応じたパラメータ調整に関する問題は緩和された一方で，一部のパラメータに関してはシーンに合わせた調整が必要であると考えられる．また，物体境界に着目した手法では，点の密度が十分に大きければ品質が改善される傾向があることは確認できたが，その密度の条件などの調査は不十分である．よって，調整が必要なパラメータの具体的な定め方の検討や，本手法が有効である場合と有効でない場合の具体的な条件の調査が必要であると考えられる．また，本研究ではシーンによらず同じ値のパラメータを使用する手法を検討し，実験に用いるシーンの全体的な品質を確認した

上でパラメータの値を決定したが，より多様なシーンで品質の改善を可能とするため，未知のシーンに対する評価や．最適なパラメータを自動で選択するような手法の検討も必要であると考えられる．

また，本研究では，点の密度による影響を明確に評価するため，ランダムに点をサンプリングすることによって密度を変更した点群を用いて評価実験を行った．密度の異なる点群に対する本手法の有効性は確認されたが，実際の点群に対する評価は不十分であるといえる．したがって，点群の取得方法や処理方法が点の密度にどのように影響するのかということなどを調査し，それを考慮した評価や改善手法の検討が必要であると考えられる．

さらに，本研究ではレンダリング品質に着目した検討を行ったため，高速化については十分な検討がされておらず，例えば密度の計算を並列化したり，近似したりすることによる高速化の余地があると考えられる．また，負荷の大きい計算を数フレームにわたって処理することによる高速化も考えられる．このような検討により，本手法によるフレームレートの低下が改善できる可能性があると考えられる．

謝辞

本研究を進めるにあたり，修士課程の二年間を通して支えていただいた多くの方々に感謝を申し上げます．

特に，指導教員である小川剛史先生には大変お世話になりました．新型コロナウイルス感染症の流行により，入学当初から例年の研究生活とは異なる部分が多かったと思います．しかし，毎週のミーティングでいただいた有意義で的確なアドバイスや，論文の添削など懇切丁寧なご指導のおかげで，リモートが中心の研究生活でありながら不自由なく研究を進めることができました．改めて，深く感謝申し上げます．

また，研究室の先輩，同期，後輩の皆様にはミーティングで様々なご意見をいただきました．大変感謝しております．先輩や同期の皆様の研究に関しては，被験者として実験に参加させていただく機会がたくさんありましたが，私自身の研究についても考えを深める機会となりました．また，雑談などは良い息抜きとなり，研究生活の大きな支えとなりました．

最後になりますが，小川先生，小川研究室の皆様をはじめ，家族や友人たちの支えもあって研究を進めることができました．私生活も含め，修士課程二年間にわたり私を支えてくださったすべての方々に深い感謝を申し上げます．ありがとうございました．

発表文献

国内会議 (査読なし)

1. 森島 正博, 小川 剛史. “三次元点群の高速・高品質な可視化のためのオクルージョン推定に関する一検討”, VR 学研報, Vol. 26, No. CS-1, CSVC2021-1, pp. 1-6 (Feb. 2021).

2. 森島 正博, 小川 剛史. “三次元点群の高速・高品質な可視化のための密度を考慮したオクルージョン推定”, 情処研報, Vol. 2022-DCC-030, No. 3 (Jan. 2022).

参考文献

[1] Christian Richardt, James Tompkin, and Gordon Wetzstein. Capture, reconstruction, and representation of the visual real world for virtual reality. In *Real VR–Immersive Digital Reality*, pp. 3–32. Springer, 2020.

[2] Florent Poux, Quentin Valembois, Christian Mattes, Leif Kobbelt, and Roland Billen. Initial user-centered design of a virtual reality heritage system: Applications for digital tourism. *Remote Sensing*, Vol. 12, No. 16, p. 2583, 2020.

[3] Tudor Caciora, Grigore Vasile Herman, Alexandru Ilieș, Ștefan Baias, Dorina Camelia Ilieș, Ioana Josan, and Nicolaie Hodor. The use of virtual reality to promote sustainable tourism: A case study of wooden churches historical monuments from romania. *Remote Sensing*, Vol. 13, No. 9, p. 1758, 2021.

[4] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th annual symposium on user interface software and technology*, pp. 741–754, 2016.

[5] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. Lookingood: Enhancing performance capture with real-time neural re-rendering. *ACM Trans. Graph.*, Vol. 37, No. 6, pp. 1–14, 2018.

[6] Kevin Ponto and Ross Tredinnick. High-resolution interactive immersive renderings of real-world environments. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 825–826, 2020.

[7] Daniele Bonatto, Ségolène Rogge, Arnaud Schenkel, Rudy Ercek, and Gauthier Lafruit. Explorations for real-time point cloud rendering of natural scenes in virtual reality. In *2016 International Conference on 3D Imaging (IC3D)*, pp. 1–7, 2016.

[8] Sören Discher, Rico Richter, and Jürgen Döllner. Concepts and techniques for web-based visualization and processing of massive 3d point clouds with semantics. *Graphical Models*, Vol. 104, p. 101036, 2019.

[9] Hassan Bouchiba, Jean-Emmanuel Deschaud, and Francois Goulette. Raw point cloud deferred shading through screen space pyramidal operators. In *Proceedings of the 39th Annual European Association for Computer Graphics Conference: Short Papers*, pp. 25–28, 2018.

[10] Patric Schmitz, Timothy Blut, Christian Mattes, and Leif Kobbelt. High-fidelity point-based rendering of large-scale 3-d scan datasets. *IEEE computer graphics and applications*, Vol. 40, No. 3, pp. 19–31, 2020.

[11] Juho-Pekka Virtanen, Sylvie Daniel, Tuomas Turppa, Lingli Zhu, Arttu Julin, Hannu Hyyppä, and Juha Hyyppä. Interactive dense point clouds in a game engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 163, pp. 375–389, 2020.

[12] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 27, No. 4, pp. 828–842, 2017.

[13] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou, Robert A. Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 9, No. 1, pp. 133–148, 2018.

[14] Ruggero Pintus, Enrico Gobbetti, and Marco Agus. Real-time rendering of massive unstructured raw point clouds using screen-space operators. In *Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage*, pp. 105–112, 2011.

[15] Pierre Biasutti, Aurélie Bugeau, Jean-François Aujol, and Mathieu Brédif. Visibility estimation in point clouds with variable density. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pp. 27–35, 2019.

[16] Taehyun Rhee, Lohit Petikam, Benjamin Allen, and Andrew Chalmers. Mr360: Mixed reality rendering for 360 panoramic videos. *IEEE transactions on visualization and computer graphics*, Vol. 23, No. 4, pp. 1379–1388, 2017.

[17] Stephen Thompson, Andrew Chalmers, and Taehyun Rhee. Real-time mixed reality rendering for underwater 360° videos. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 74–82, 2019.

[18] Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. 6-dof vr videos with a single 360-camera. In *2017 IEEE Virtual Reality (VR)*, pp. 37–44, 2017.

[19] Ryan S Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Transactions on Graphics (TOG)*, Vol. 37, No. 6, pp. 1–15, 2018.

[20] Hochul Cho, Jangyoon Kim, and Woontack Woo. Novel view synthesis with multiple 360 images for large-scale 6-dof virtual reality system. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 880–881, 2019.

[21] Sören Discher, Leon Masopust, Sebastian Schulz, Rico Richter, and Jürgen Döllner. A point-based and image-based multi-pass rendering technique for visualizing massive 3d point clouds in vr environments. *Journal of WSCG*, Vol. 26, No. 2, pp. 76–84, 2018.

[22] Emin Zerman, Cagri Ozcinar, Pan Gao, and Aljosa Smolic. Textured mesh vs coloured point cloud: A subjective study for volumetric video compression. In *2020*

Twelfth International Conference on Quality of Multimedia Experience (QoMEX), pp. 1–6, 2020.

[23] Gerd Bruder, Frank Steinicke, and Andreas Nüchter. Poster: Immersive point cloud virtual environments. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 161–162, 2014.

[24] Hansjörg Hofer, Florian Seitner, and Margrit Gelautz. An end-to-end system for real-time dynamic point cloud visualization. In *2018 International Conference on 3D Immersion (IC3D)*, pp. 1–8, 2018.

[25] Bo Han, Yu Liu, and Feng Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–13, 2020.

[26] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 371–378, 2001.

[27] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pp. 17–141, 2005.

[28] Reinhold Preiner, Stefan Jeschke, and Michael Wimmer. Auto splats: Dynamic point cloud visualization on the gpu. In *EGPGV*, pp. 139–148, 2012.

[29] Claus Scheiblauer and Michael Wimmer. Out-of-core selection and editing of huge point clouds. *Computers & Graphics*, Vol. 35, No. 2, pp. 342–351, 2011.

[30] Markus Schütz and Michael Wimmer. High-quality point-based rendering using fast single-pass interpolation. In *2015 Digital Heritage*, Vol. 1, pp. 369–372, 2015.

[31] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision – ECCV 2020*, pp. 696–712, 2020.

[32] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7830–7839, 2020.

[33] Gal Metzer, Rana Hanocka, Raja Giryes, Niloy J Mitra, and Daniel Cohen-Or. Z2p: Instant rendering of point clouds. *arXiv preprint arXiv:2105.14548*, 2021.

[34] Markus Schütz, Katharina Krösl, and Michael Wimmer. Real-time continuous level of detail rendering of point clouds. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 103–110, 2019.

[35] Ross Tredinnick, Markus Broecker, and Kevin Ponto. Progressive feedback point cloud rendering for virtual reality display. In *2016 IEEE Virtual Reality (VR)*, pp. 301–302, 2016.

[36] Markus Schütz, Gottfried Mandlburger, Johannes Otepka, and Michael Wimmer. Progressive real-time rendering of one billion points without hierarchical acceleration structures. *Computer Graphics Forum*, Vol. 39, No. 2, pp. 51–64, 2020.

[37] Ross Tredinnick, Markus Broecker, and Kevin Ponto. Experiencing interior environments: New approaches for the immersive display of large-scale point cloud data. In *2015 IEEE Virtual Reality (VR)*, pp. 297–298, 2015.

[38] Markus Schütz and Michael Wimmer. Rendering point clouds with compute shaders. In *SIGGRAPH Asia 2019 Posters*, pp. 1–2. 2019.

[39] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Rendering point clouds with compute shaders and vertex order optimization. *arXiv preprint arXiv:2104.07526*, 2021.

[40] B Alsadik, M Gerke, and G Vosselman. Visibility analysis of point cloud in close range photogrammetry. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 2, No. 5, pp. 9–16, 2014.

[41] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*, pp. 24–es. 2007.

[42] Wanning Zhang, Fuqiang Zhou, Lin Wang, and Pengfei Sun. Region growing based on 2-d–3-d mutual projections for visible point cloud segmentation. *IEEE Transactions on Instrumentation and Measurement*, Vol. 70, pp. 1–13, 2021.

[43] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 143–152, 2017.

[44] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, pp. 129–136, 2008.

[45] Josselin Gautier, Olivier Le Meur, and Christine Guillemot. Efficient depth map compression based on lossless edge coding and diffusion. In *2012 Picture Coding Symposium*, pp. 81–84, 2012.

[46] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[47] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, Vol. 5, No. 4, pp. 349–359, 1999.

[48] Yutaro Iwamoto, Xian-Hua Han, Tomoko Tateyama, Motonori Ohashi, So Sasatani, and Yen-Wei Chen. Gradient based edge preserving interpolation and its application to super-resolution. *IEEJ Transactions on Electronics, Information and Systems*, Vol. 131, No. 11, pp. 1901–1906, 2011.