エネルギー科学研究科

エネルギー社会・環境科学専攻修士論文

題目：　動的に変化する実世界小空間への没入を目的とした高精細立体映像提示システムの開発

指導教員：下田　宏　教授

氏名：山田　涼楓

提出年月日：令和7年2月5日（水）

論文要旨

題目：動的に変化する実世界小空間への没入を目的とした高精細立体映像提示システムの開発

エネルギー情報学分野（下田研究室），山田涼楓

要旨：

近年、エネルギー問題や防災対策についての議論は、地域社会の持続可能な発展や安全確保において重要な役割を担っている。これらの議論の場では、一般的に図表や写真などの補助資料が用いられるが、従来の紙媒体やPC画面などの2次元的なツールでは直感的な理解が困難である。これに対し、3次元的なツールとしてミニチュア模型を活用する手法も試みられているが、災害発生時の状況を具体的に模擬することは難しく、また、ミニチュアであるために現実の大きさで想像することが難しく、現実に即した議論を行う上では限界がある。

そこで本研究では、実在する小空間内を対象に高精細な自由視点画像をリアルタイムに生成する手法を実現することで、ミニチュアのジオラマで形成された、配置や構成を直接手で触って変更可能な動的に変化する実世界小空間への没入体験を可能とするシステムを開発した。実現した手法では、最初に、没入対象となる小空間を複数のRGB-Dカメラで撮影し、デプス画像とカラー画像を取得した。次に、取得した複数のデプス画像をノイズ除去処理を適用しながら融合し、高解像度な自由視点デプス画像を生成した。最後に、生成した自由視点高解像度デプス画像と取得したカラー画像を用いて自由視点画像の各ピクセルの色を決定し、自由視点カラー画像を生成した。その際、デプス情報を利用することで、被写体同士の重なりによるオクルージョンの影響を排除し、生成画像の色の誤りを改善した。加えて、被写体の形状に応じた色の参照を行うことで、生成画像の歪みを改善した。

その後、体験者の頭の動きに合わせて、両目用の自由視点画像を個別に生成してHMDの両眼に提示することで立体視を可能にし、動的に変化する実世界小空間への没入体験を実現した。システム体験時には、視点を被写体の特定の個所に近づけることで、被写体の細部を確認することができた。さらに、ミニチュア模型のレイアウトを変化させることで、体験者自身が環境を自由に作り変えながら体験することができた。

今後の課題としては、本システムを複数人で同時に体験できるように拡張することや、システム体験時の印象評価実験を実施し、本システムでの体験が体験者の心理に及ぼす影響について調査することなどが挙げられる。

目 次

図 目 次

表 目 次

第 1 章 序論

1.1 研究の背景

近年、エネルギー問題や防災対策についての議論は、地域社会の持続可能な発展や安全の確保等において重要な役割を担っている。これらの議論の場では、図表や写真などの補助資料を活用し、議論の対象の具体的な状況や課題についてのイメージの共有を図ることが一般的である。しかし、従来の紙媒体や PC 画面などの 2 次元的なツールを用いた補助資料 [1–5] では議論対象を直感的に理解することが困難な場合がある。また、これらのツールでは 3 次元的な操作が難しく、例えば、街の防災計画の立案や都市開発のシミュレーションにおいて、議論の進行に応じてその場で建物のレイアウトを直感的に操作しながら議論を行うことは困難である。

これに対し、3 次元的なツールとしてミニチュアサイズの模型を用いてイメージの共有を図る手法も試みられてきた [6–8]。この手法では、模型の作成に手間と時間を要するものの、議論対象の状況を視覚的に理解しやすく、また、議論の内容に応じて模型のレイアウトを柔軟に変更できる。この方法を用いれば、街の防災計画立案や都市開発のシミュレーションにおいて、街や都市の状況を俯瞰的に把握できる。また、議論内容に応じてその場で模型のレイアウトを適宜調整することで、それぞれの状況に応じた具体的な議論を進めることができる。一方で、この方法では、火災や水害などの災害発生時の状況を具体的に模擬することは難しい。また、ミニチュアであるために現実の大きさで想像することが難しく、議論対象の空間を現実の空間として体感することは困難であり、現実に即した議論を行う上では限界がある。

このような課題に対する新たなアプローチとして、近年、Augmented Reality（以下、AR）や Virtual Reality（以下、VR）の応用が注目されている。AR とは、現実の環境から視覚や聴覚、触覚などの知覚に与えられる情報を追加あるいは削減、変化させる技術 [9] であり、VR とは、現実ではないが実質的に現実のように感じられる環境を人工的に作り出す技術 [10] である。これらの技術は、低コストかつ、体験の際に高い臨場感を提供できる。加えて、火災や水害などの現実では体験が難しい災害も再現できる。AR を活用した体験では、デバイスの画面上で炎や水などのエフェクトを現実世界に重畳することで、模型だけでは再現が難しい災害時の様子を視覚的に模擬できる。ただ

し、模擬した環境内に入り込んだかのような没入型の体験を提供することは難しく、デバイスの画面を通して、作成した模型を俯瞰的に眺める形に限定される。一方で、VRを活用した体験では、災害時の環境を再現した仮想空間内に没入することで、あたかも現実世界で災害を体験しているかのような体感を得ることができる。ただし、VRでは、触覚デバイス等の技術的制約により、体験者が仮想空間内のオブジェクトを直感的に操作することが難しく、体験者の操作性に課題が残されている。

これらの課題を解決するアプローチとして、体験者が小型模型（ミニチュア）を自らの手で直接操作しながら、そのミニチュアの世界に入り込んだかのような没入体験を提供するシステムを実現することが考えられる。このようなシステムが実現すれば、模型のレイアウトを変更することで、体験環境をリアルタイムに作り変えながら体験することが可能となる。また、AR技術とCGエフェクトの活用により、体験時の臨場感を向上させることができる。さらに、議論の進行状況をログとして保存しておくことで、過去の議論の状況を再現したり、詳細に確認することも可能になる。

1.2　研究の目的

本研究では、複数のカメラと高速画像処理技術を用いて、ミニチュアのジオラマで形成された、配置や構成を直接手で触って変更可能な、動的に変化する実世界小空間への没入体験を可能とするシステムを開発することを目的とする。

このシステムが実現すれば、小空間内に配置したミニチュアのジオラマ内で地震や水害等を模擬的に発生させることで、リアルな災害体験が可能になる。加えて、設計者の用意した環境に制限されることなく、体験者自身がリアルタイムに環境を作り変えながら体験することも可能になる。また、本システムを、複数人で同じ空間に同時に没入できるように発展させれば、ミニチュア模型の配置をリアルタイムに操作しながら直感的な議論を行うことで、集団での共感や問題発見が促進可能な、新たな集団コミュニケーションの形を提供することも可能になる。

第 2 章 自由視点画像生成に関する既存手法

本研究では、動的に変化する実世界小空間への没入体験を可能とするシステムの開発を目的としている。このようなシステムを開発するためには、1.2 節で述べたように、小空間を複数のカメラで同時に撮影し、撮影した画像を用いて小空間内の高精細な自由視点画像をリアルタイムに生成する必要がある。

自由視点画像生成に関する既存手法には、被写体の 3 次元形状を計測し、復元した 3 次元モデルをレンダリングして自由視点画像を生成する手法と、被写体の 3 次元形状を用いずに画像群や光線情報を用いて直接自由視点画像を生成する手法がある [11, 12]。また、近年では、ニューラルネットワークを用いて新たな視点からの画像を生成する手法も提案されている。

本章では、以上で述べた、被写体の 3 次元形状を計測して自由視点画像を生成する手法、画像群から直接自由視点画像を生成する手法、およびニューラルネットワークを用いて自由視点画像を生成する手法に関する既存研究について詳述する。

2.1 被写体の 3 次元形状を計測して自由視点画像を生成する手法

被写体の 3 次元形状を計測して自由視点画像を生成する手法は、光を被写体に照射して 3 次元形状を計測する能動的な計測方法を用いたものと、通常の RGB カメラから得られる画像情報のみから 3 次元モデルの復元を行う受動的な計測方法を用いたものの 2 つの方法に大別される。

能動的な計測方法には、Time of Flight(以下、ToF)、スリット光投影法（光切断法

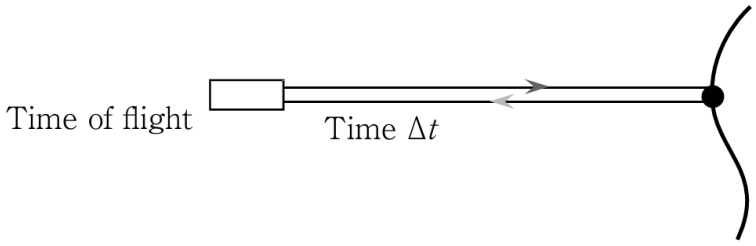


図 2.1: ToF による距離の計測方法 [13]

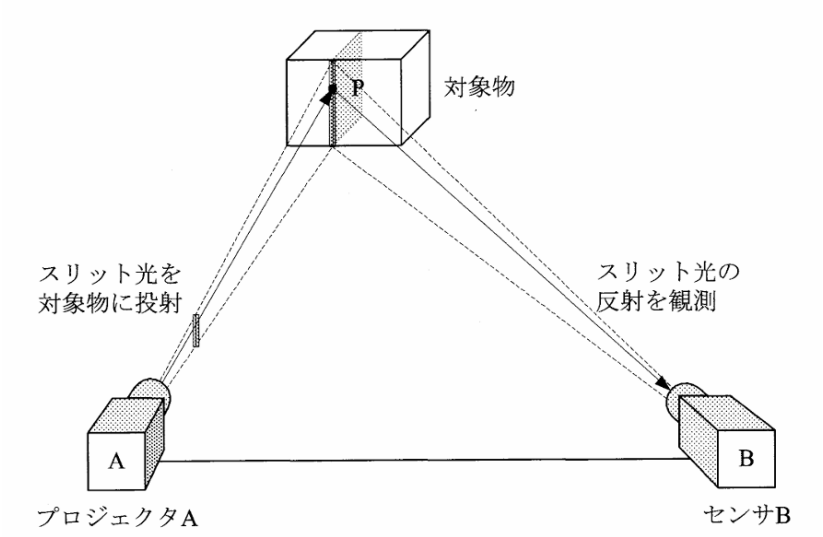


図 2.2: スリット光投影法による計測方法 [14]

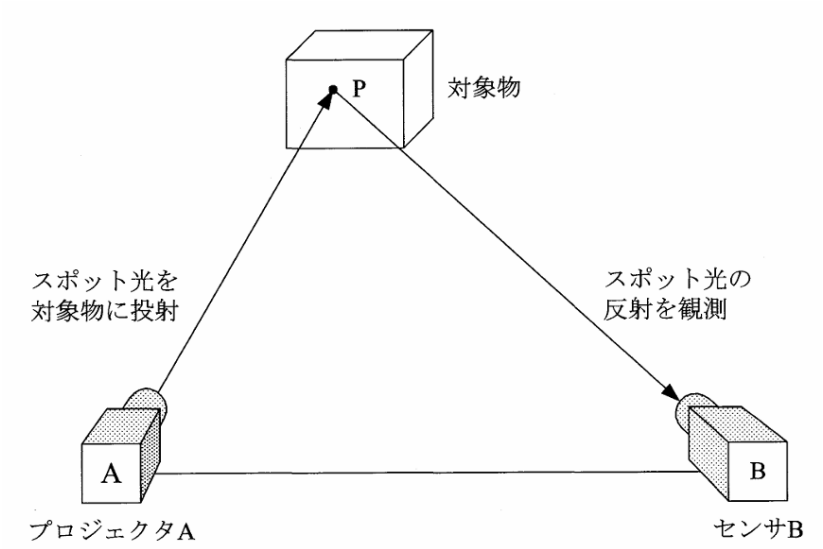


図 2.3: スポット光投影法による計測方法 [14]

とも呼ばれる)、スポット光投影法などがある [13,14]。 ToF では、図 2.1 のように、投影したレーザー光が被写体上で反射して戻るまでの時間（位相差）から距離を算出する。スリット光投影法では、図 2.2 のように、照射光パターンの歪みやずれから三角測量の原理で距離を算出する。スポット光投影法では、図 2.3 のように、投影先にできる光点位置のずれから三角測量の原理で距離を算出する。これらの手法の問題点として、ToF センサでは、高性能な ToF センサはコストが高いことや、極端に近いまたは遠い距離を計測する際は奥行方向の精度が低いこと、スリット光投影法では、ToF と共通する問題として複数の機器を同時に使用すると照射光が互いに干渉してしまうこと、ス

ポット光投影法では、得られる距離情報の解像度が低く、また精度も低いことなどが挙げられる。

これらの計測方法を用いて3次元形状の復元を行う方法には、KinectFusion[15]などがある。KinectFusionは、上述した計測方法で取得した被写体までの距離情報とTruncated Signed Distance Function（TSDF）を用いて、図2.4に示すように被写体の3次元形状を復元する。この手法では、処理対象の3次元空間を均等な大きさのボクセルに分割してグリッド毎にTSDFを計算し、ボクセル単位でノイズ除去を行うことでモデルの精度を向上させる[16]。この手法の問題点として、被写体の詳細な凹凸まで高精度に処理を行うには大量のメモリや計算が必要となることが挙げられる。その際、効率的なメモリ管理のために視点によってグリッドの解像度を動的に変化させることも考えられるが、KinectFusionでの適切なグリッドの解像度は、最終的に生成する自由視点画像の解像度や、自由視点の姿勢と被写体の位置関係に応じて変化する。また、自由視点の視野内の空間に対しても、視点に近い領域ほど高解像度のグリッドが求められ、遠い領域では低解像度のグリッドが適切となる。こうした関係は非常に複雑であるため、最適なグリッド解像度をリアルタイムで求めるのは困難である。

一方、受動的な計測方法を用いて3次元形状を復元する方法には、視体積交差法[17–19]やMulti View Stereo（以下、MVS）[20,21]などがある。 視体積交差法は、被写体のシルエットを用いて3次元形状を復元する方法である。具体的には、図2.5に示すように、各カメラの視点から見たシルエットを基に、物体が存在する可能性のある空間領域（視体積）を求め、空間を等間隔に分割したボクセルのうち視体積外となるボクセルを削除することで3次元形状を復元する。MVSは、あらかじめ求めたカメラの姿勢情報と画像群を用いて3次元形状を復元する方法である。具体的には、画像ごとの対応点から、ステレオ法を用いてその奥行きを推定し、図2.6に示すように3次元形状を復元する。これらの手法の問題点としては、視体積交差法では、原理的に凹んだ形状を復元

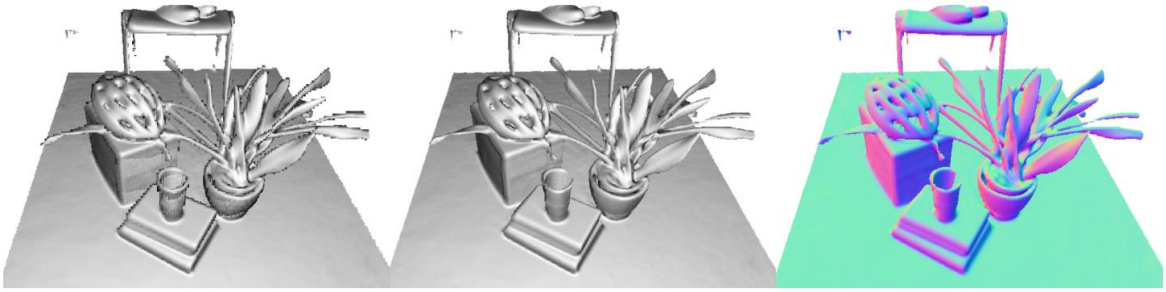


図 2.4: KinectFusionで復元した3次元モデルの例[15]

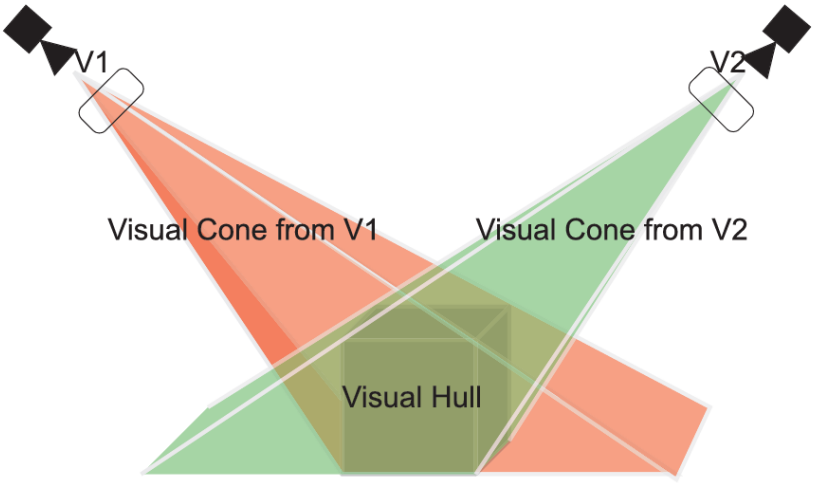


図 2.5: 視体積交差法の概念図 [19]

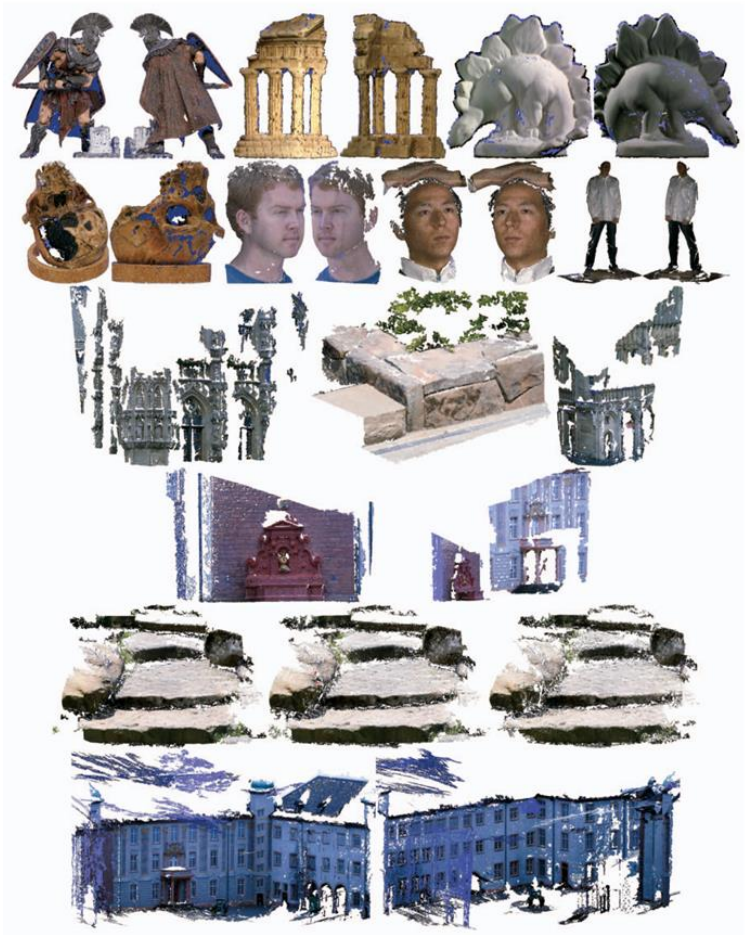


図 2.6: patch-based MVS で復元した 3 次元モデルの例 [21]

できないため詳細な形状の再現に不向きであること、MVS では、特徴的なテクスチャの無い領域では特徴点が認識できず欠損が生じてしまうことや、計算量が多く処理速度が遅いことなどが挙げられる。

2.2 画像群から直接自由視点画像を生成する手法

画像群から直接自由視点画像を生成する手法には、モーフィング[22]や、ライトフィールドレンダリング[23,24]などがある。モーフィングでは、2つの画像間で複数の対応点を設定し、それを基に、図2.7に示すように2つの画像の中間画像を生成する。ライトフィールドレンダリングでは、対象となる空間を通る光線情報を記録することで任意の視点における画像を生成する。これらの手法の問題点として、モーフィングでは元となる2つの画像の中間の画像しか高精度に生成できないこと、ライトフィールドレンダリングでは大量のデータを用いるため計算量が多く、処理速度が遅いことなどが挙げられる。

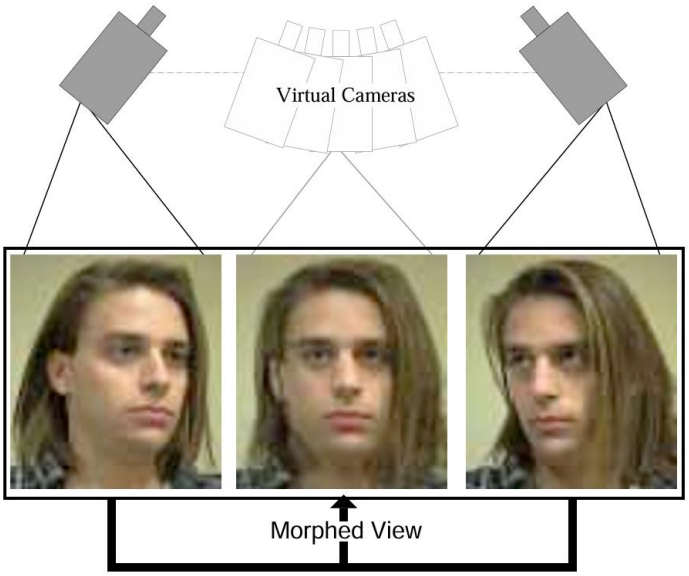


図 2.7: モーフィングによる生成画像の例[22]

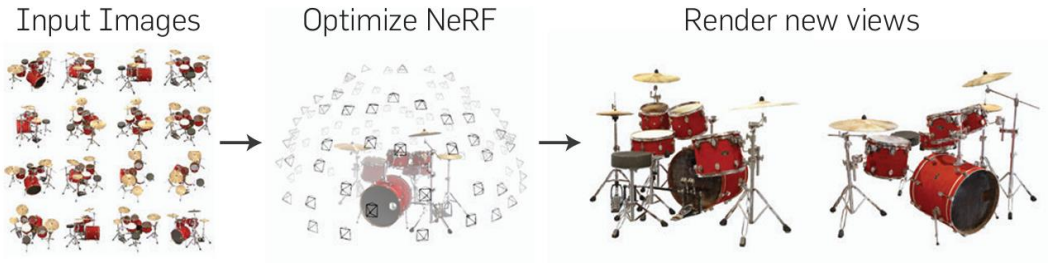


図 2.8: NeRF を用いて生成した自由視点画像の例 [25]

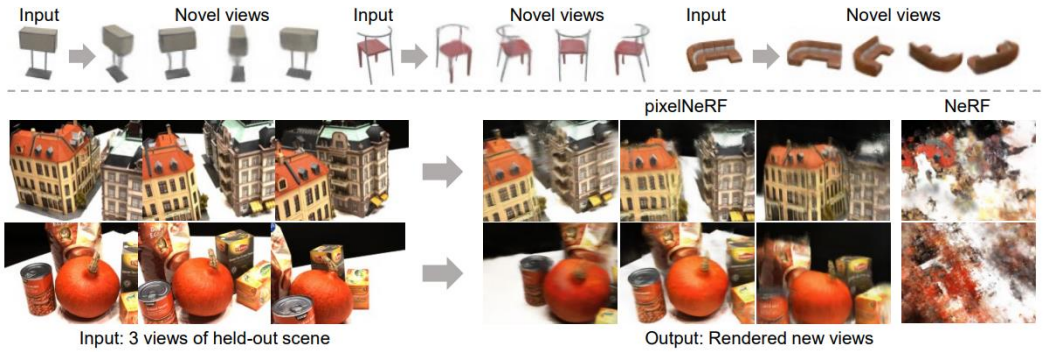


図 2.9: pixelNeRF を用いて生成した自由視点画像の例 [26]

2.3 ニューラルネットワークを用いて自由視点画像を生成する手法

近年、ニューラルネットワークを用いた自由視点画像生成手法として、Neural Radiance Fields（以下、NeRF）[25] とその拡張 [26–31] が注目されている。NeRF では、シーン全体を体積的な放射輝度として表現し、ニューラルネットワークを利用して各視点における RGB 値と体積密度を推定することで、図 2.8 に示すように、任意の視点からの高品質な画像を生成する。しかし、これらの手法は静的なシーンを対象としており、処理速度の問題により動的なシーンには対応できない。また、高品質な画像の生成には大量の入力画像が必要であり、入力画像の不足により生成される画像の精度が著しく低下する可能性があるなどの問題もある。これらの問題に対処するため、最近では、NeRF を拡張した手法が提案されている。例えば、pixelNeRF[26] は、図 2.9 に示すように、1 枚または少数の入力画像から画像を生成することができるが、生成画像の品質には課題が残る。IBRNet[27] では、従来の NeRF と比較して計算時間が大幅に短縮されたが、依然としてリアルタイムでの処理は難しく、また、高品質な画像の生成には高密度の

入力画像を必要とする。このように、ニューラルネットワークを用いた自由視点画像生成手法の問題点としては、計算コストが高く、動的なシーンに対する自由視点画像をリアルタイムに生成することが困難であることや、学習時に撮影されなかったものがシーンに入ると精度が低下すること、シーンが複雑な場合に不自然な画像が生成されリアリティが損なわれる可能性があることなどが挙げられる。加えて、これらの手法では、推測された生成画像が実際の環境を正しく表しているかも不確かである。

2.4 本研究の位置づけ

上述のように、自由視点画像生成の既存手法では、生成する自由視点画像の精度や処理速度等の問題があり、任意の視点からの実写性の高い画像をリアルタイムに生成することは困難である。特に、自由視点画像を生成する対象の被写体が小さい場合にはノイズや解像度の問題が顕著となり、高精細な映像をリアルタイムに生成することは難しい。

本研究では、RGB-D カメラと呼ばれる、カラー画像とデプス画像（ピクセル毎に被写体までの距離を示す画像）を同時に取得できるカメラを用いて、任意の視点から見たデプス画像（以下、自由視点デプス画像）を高解像度かつ高速に生成し、それを用いて、小空間に対する高精細な自由視点画像をリアルタイムに生成する手法を実現する。RGB-D カメラを用いることで、撮影する空間の 3 次元情報を高速に取得することが可能となるが、1 台の RGB-D カメラのみでは物陰となる領域が多く、小空間内の環境を抜け落ちなく撮影することは困難である。また、市販の RGB-D カメラでは解像度に限界があり、1 台のカメラ画像のみでは詳細な画像を生成することは困難である。そのため、複数のカメラを同時に用いる必要があるが、使用するカメラの台数の増加に伴いシステムでの処理の計算量も増えるため、如何にして効率的かつ並列処理可能なアルゴリズムとして実現するかが課題となる。加えて、RGB-D カメラで取得されるデプス画像にはノイズが多く含まれるという問題があり、如何にしてノイズを低減させるかも課題となる。これらの課題に対して、本研究では、3.5 節で述べるノイズ除去アルゴリズムにより、デプス画像に含まれるランダムノイズの除去を試みる。また、提案システムでの処理を並列化可能なアルゴリズムとして設計し、複数の PC を用いて処理を分散させるとともに、生成画像のピクセル毎に GPU を用いた並列処理を適用することで処理速度の向上を試みる。

第 3 章 高精細立体映像提示システムの設計

本章では、開発する高精細立体映像提示システムの概要について説明し、次に提案システムでの処理について説明する。

3.1 提案システムの概要

提案システムの概要を図 3.1 に示す。動的な（動きのある）小空間への没入体験を可能にするためには、没入対象となる小空間を複数のカメラで同時に撮影し、撮影した画像を用いて、自由視点画像をリアルタイムに生成する必要がある。本システムでは、数十 cm 四方程度のボックス内の空間を没入対象の小空間として設定し、ボックスの周囲に複数の RGB-D カメラを設置する。提案システムの体験の際は、体験者はトラッキング機能を備えた両眼式ヘッドマウントディスプレイ（以下、HMD）を装着し、体験者の頭の動きに応じてリアルタイムに右目用・左目用の自由視点画像を個別に生成して HMD 上に提示することで、両眼立体視をしながら動的に変化する小空間に没入することを可能にする。この時、HMD を装着する際や体験中に、HMD の姿勢が体験者の意図しない姿勢になってしまう場合がある。例えば、HMD の初期キャリブレーション時に HMD の向きがずれていたり、長時間の使用により体験者の姿勢の変化が生じたりすることで、正しい空間認識が困難になる場合がある。本システムでは、体験者の持つコントローラの操作により HMD の姿勢が適宜初期化され、視点を自然な姿勢にリセットすることができるようにする。

本システムでの処理は、事前準備の際に実行するオフライン処理と体験時に実行するオンライン処理に分けられる。オフライン処理では、キャリブレーションとして、使用するカメラの焦点距離などの内部パラメータとカメラ間の相対姿勢（本論文では，カメラの位置と方向を合わせて姿勢と呼ぶ）を求める。オンライン処理では、初めに、没入対象となるボックス内の空間を複数の RGB-D カメラで撮影し、空間内のデプス画像とカラー画像を取得する。そして、取得した複数のデプス画像をピクセルごとにノイズ除去処理を適用しながら融合することで、高解像度な自由視点デプス画像を生成する。その後、生成した自由視点デプス画像と取得したカラー画像を元に、各ピクセルが格納するデプス値を用いて、自由視点画像の各ピクセルの色を決定する。この際、

被写体同士の重なりによる生成画像の色の誤りを軽減するため、取得したデプス画像を用いて、オクルージョン（手前にある被写体が後ろにある被写体を隠している状態）を考慮した上でカラー情報を取得する。また、生成画像の歪みを低減させるため、被写体表面の正面から撮影した RGB-D カメラから優先的にカラー情報を取得する。以上の処理を両眼用にそれぞれ個別に実行し、HMD の左右の目に異なる映像を提示することで立体視を可能にする。

本システムでは、動的に変化する実世界小空間への没入体験を可能にするために、上記の処理をリアルタイムで実行する必要がある。したがって、本処理を並列化可能なアルゴリズムとして設計し、複数の PC を用いて処理を分散させるとともに、GPU を用いた並列処理による高速化を行った。

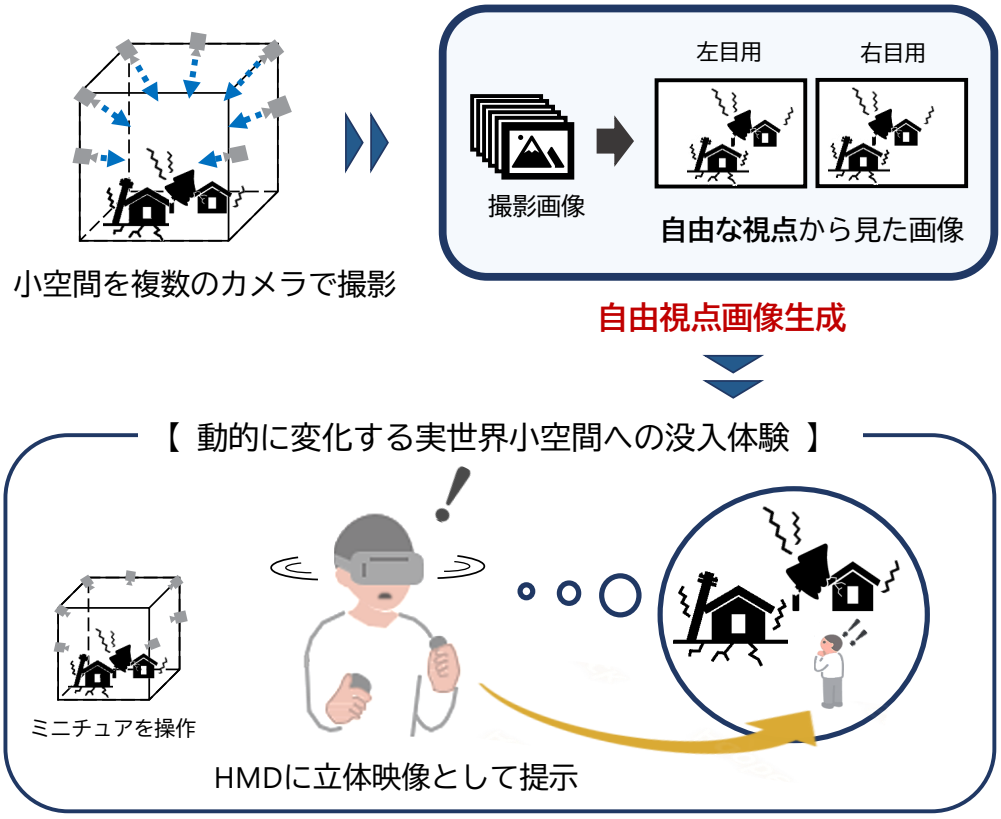


図 3.1: 提案システムの概要

3.2 座標系の定義

本提案システムでは、各カメラからのデプス画像の融合やカラー情報の取得のために、使用するカメラ間の座標系の相互変換が必要である。本節では、本研究で使用する座標系について定義する。図 3.2 および図 3.3 に本研究で扱う座標系を示す。N 個のカメラのうち i 番目のカメラの座標系 Ci（$i = 1, 2, ..., N$）は、カメラの焦点位置を原点、光軸方向（画像平面に垂直な方向）を z 軸の正の方向、画像の x,y 軸に平行な方向をそれぞれ x 軸、y 軸とする右手系の座標系とする。観察者の視点となる自由視点座標系 F は、カメラ座標系と同様に定義され、観察方向を z 軸の正の方向、観察者から見て鉛直下向きを y 軸の正の方向、右の方向を x 軸の正の方向とする。本研究では、カメラ座標系 Ci の内、事前に定めた 1 つのカメラ座標系を本システムでの世界座標系 W とし、その他のカメラおよび自由視点の世界座標系に対する相対姿勢を用いて、各カメラ座標系での座標を世界座標系基準のものに変換する。各カメラ座標系の世界座標系に対する同次変換行列 T_{ciw} は、回転行列を $\boldsymbol{R_{ci}}$、 並進ベクトルを $\boldsymbol{t_{ci}}$ とすると、

$$T_{ciw} = \begin{pmatrix} \boldsymbol{R_{ci}} & \boldsymbol{t_{ci}} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tag{3.1}$$

$$\boldsymbol{R_{ci}} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{3.2}$$

$$\boldsymbol{t_{ci}} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{3.3}$$

で表される。この時、各カメラ座標系での座標 $(x_{ci}, y_{ci}, z_{ci})^\top$ を世界座標系での座標に変換した値 $(x_w, y_w, z_w)^\top$ は、以下の式で導かれる。

$$\begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = T_{ciw} \begin{pmatrix} x_{ci} \\ y_{ci} \\ z_{ci} \\ 1 \end{pmatrix} \tag{3.4}$$

同様に、自由視点座標系の世界座標系に対する同次変換行列 T_{fw} は、回転行列を $\boldsymbol{R_f}$、

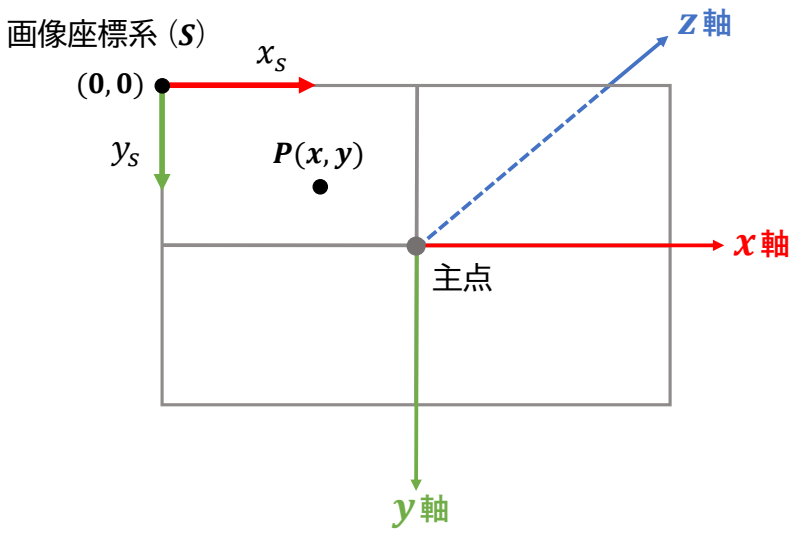


図 3.2: 画像軸および画像座標系の定義

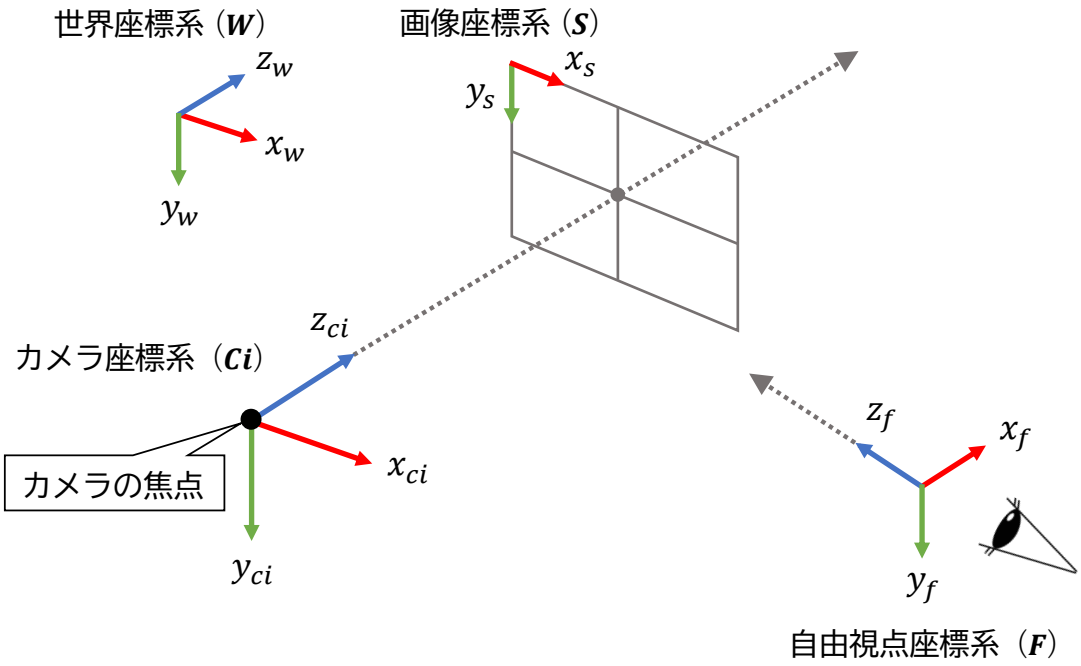


図 3.3: 本研究で使用する座標系

並進ベクトルを $\boldsymbol{t_f}$ とすると、

$$T_{fw} = \begin{pmatrix} \boldsymbol{R_f} & \boldsymbol{t_f} \\ \boldsymbol{0}^\top & 1 \end{pmatrix} \tag{3.5}$$

$$\boldsymbol{R_f} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{3.6}$$

$$\boldsymbol{t_f} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \tag{3.7}$$

で表される。この時、自由視点座標系での座標 $(x_f, y_f, z_f)^\top$ を世界座標系での座標に変換した値は、以下の式で導かれる。

$$\begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = T_{fw} \begin{pmatrix} x_f \\ y_f \\ z_f \\ 1 \end{pmatrix} \tag{3.8}$$

3.3 使用する RGB-D カメラの選定

本システムでは、体験中にボックス内の任意の位置へ視点を移動した際の映像を生成するため、小空間内の環境を様々な方向から同時に撮影する必要がある。そのため本研究では複数のカメラを同時に使用するが、2 章で述べたように、被写体に特徴が少ない場合、カラー画像のみを用いて自由視点画像を生成することは困難である。そこで、本システムでは、被写体の特徴が少ない場合でも体験を可能にするために、カラー画像に加えてデプス画像を取得可能な RGB-D カメラを使用する。RGB-D カメラは、2.1 節で述べた能動的な計測方法で被写体までの距離を計測する場合が多い。ToF 方式を採用した RGB-D カメラは解像度が高く、スリット光投影方式を採用した RGB-D カメラは距離精度が高いなどの利点があるが、これらの方式は複数の機器を同時に使用すると照射光が互いに干渉してしまい、計測精度が大幅に低下するという問題がある。これに対して、アクティブ IR ステレオ方式を採用した RGB-D カメラは、環境の視覚的特徴を増やすことを目的に赤外線を照射し、照射のパターン自体は利用しないため、複数の機器を同時に使用できる。以上の理由から、本研究では RGB-D カメラとしてア

クティブ IR ステレオ方式の RGB-D カメラを使用する。ただし、3.5 節で後述するように、アクティブ IR ステレオ方式の RGB-D カメラは取得したデプス画像にノイズが多く含まれるという問題があり、その対策が必要である。

3.4 カメラのキャリブレーション方法

カメラのパラメータには内部パラメータと外部パラメータがある。内部パラメータはカメラの焦点距離や歪み係数などのカメラの特性を示し、外部パラメータはカメラの姿勢である位置と方向を示す。このようなカメラのパラメータを事前に求める作業をキャリブレーションと呼ぶ。3.2 節で述べたように、本研究では、使用するカメラ間での座標系の相互変換を可能にするために、あらかじめ配置したすべてのカメラの外部パラメータを求めておく必要がある。また、カメラ画像の処理の際には各カメラの内部パラメータも求めておく必要がある。本研究では、Multical[32]、Multi-View Environment(MVE)[33]、および ORB SLAM3[34,35] の 3 つのソフトウェアを使用してキャリブレーションの方法を検討した。それぞれの具体的な方法の詳細については 4.2.1 項で述べる。

3.5 デプス画像のノイズ除去と自由視点デプス画像の生成

本研究では、後の処理としてオクルージョンへの対応や被写体表面形状に応じた処理を実行可能にするために、最初に複数の RGB-D カメラから取得したデプス画像を融合して自由視点デプス画像を生成する。本節では、複数のデプス画像をノイズ除去処理を適用して融合し、自由視点デプス画像を生成する処理について述べる。

3.5.1 アルゴリズムの概要

3.3 節で述べたように、本研究では、RGB-D カメラとしてアクティブ IR ステレオ方式を採用したものを使用するが、この方式では取得されるデプス情報にノイズが多いという問題がある。ノイズ除去の既存手法としては、3 次元空間を分割したボクセル単位でノイズ除去を行う KinectFusion などがあるが、2.1 節で述べたように、高精細な画像の生成には大量のメモリが必要となり、また、効率的なメモリ管理のための処理も複雑である。加えて、処理の中で一度被写体の 3 次元形状を推定するため、多視点からの映像生成には適している一方で、今回のような少数の視点からの映像の生成に使用するには無駄が多い。そこで、本研究では、村山らのノイズ除去手法 [36] をベースと

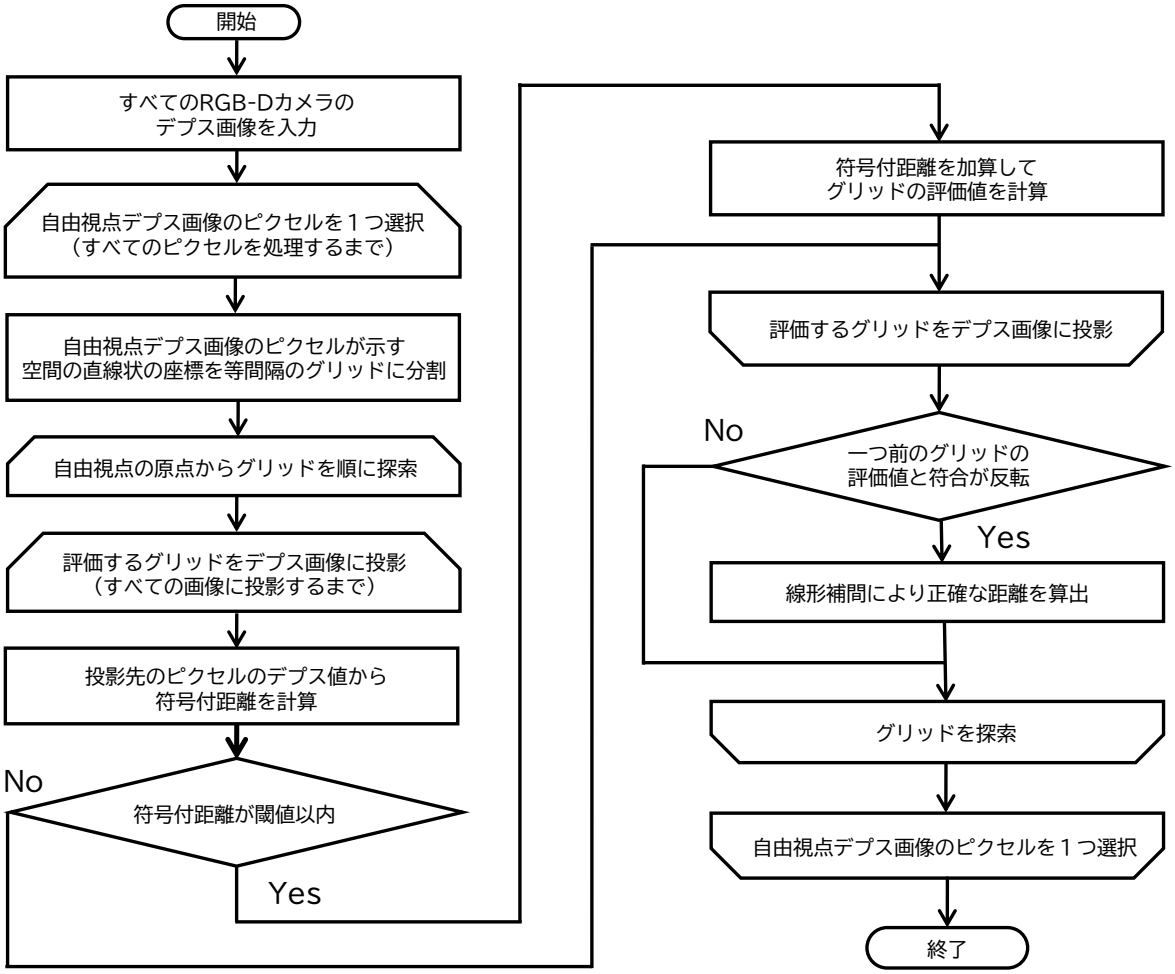


図 3.4: 自由視点デプス画像を生成する処理の流れ

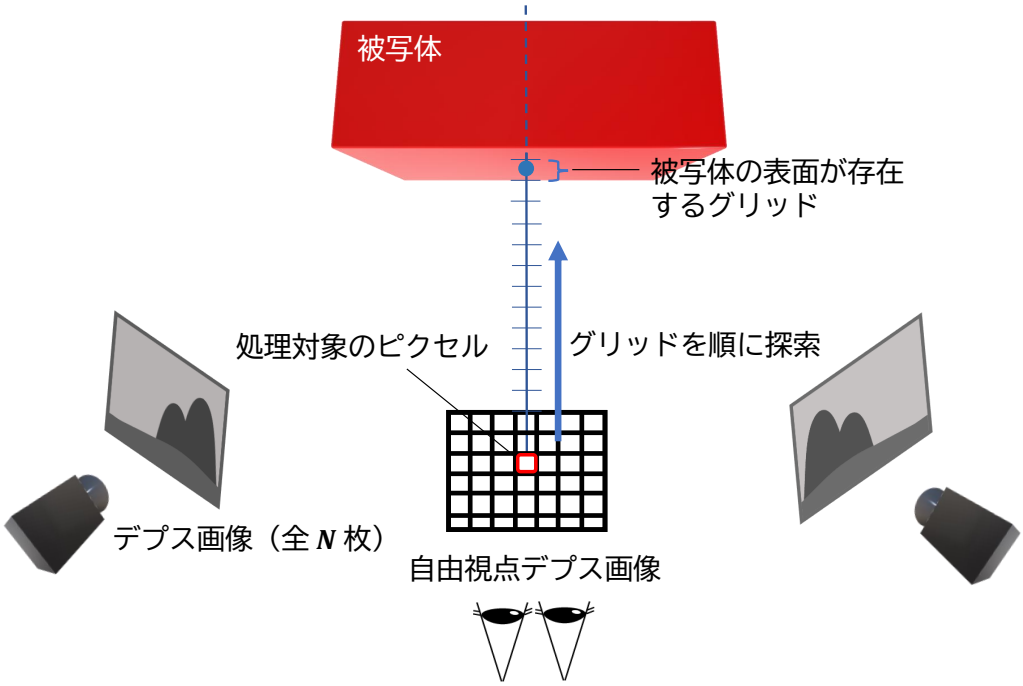


図 3.5: ノイズ除去アルゴリズムの概要

して、デプス画像のノイズ除去と自由視点の新たなデプス画像の生成を行う手法を実現する。村山らのノイズ除去処理では、環境を 1 台のカメラで異なる位置から連続して撮影し、撮影したデプス画像を用いて、撮影時と同じカメラ姿勢におけるノイズを除去したデプス画像を出力する。一方、本手法では、環境を複数の異なるカメラで同時に撮影し、撮影したデプス画像を用いて、ノイズを除去した新しい視点のデプス画像（自由視点デプス画像）を生成する。自由視点デプス画像を生成する処理の流れを図 3.4 に示す。はじめに、図 3.5 の青線のように、自由視点デプス画像の各ピクセルが写す空間の直線状の座標を等間隔のグリッド（本論文におけるグリッドとは、直線を等間隔に区切った座標を示す）に分割する。そして、自由視点の原点から順にグリッドを探索し、各グリッドに対して被写体の表面がそのグリッド内部に含まれるかどうかを評価することで被写体表面までの正確な距離を求め、正しいデプス値とする。

先述のように、KinectFusion では、高精細な画像生成のためのメモリ管理として、視点によってグリッドの解像度を動的に変化させる処理が考えられる。しかし、最適なグリッドの解像度は最終的に生成する自由視点画像の解像度や自由視点の姿勢と被写体の位置関係に応じて変化する。さらに、自由視点の視野内の空間に対しても、自由視点との距離に応じてグリッドの解像度をそれぞれ適切に調整する必要がある。これに対して、本手法では、生成画像のピクセル単位でノイズ除去およびデプス値の算出

を行う点が特徴であり、上述のような複雑な処理を必要としない。そのため、生成する自由視点デプス画像の詳細度が被写体とカメラの間の距離に依存せず、また、各カメラから得られる情報を有効に利用できる。加えて、ピクセル毎に独立した処理を行うため、GPU を用いた並列処理が可能である。

3.5.2 アルゴリズムの詳細

本処理では、自由視点デプス画像のすべてのピクセル毎に以下の処理を実行する。まず、図 3.6 のように、自由視点の原点から順にグリッドを探索し、あらかじめ求めた N 個のカメラの姿勢を用いて、評価するグリッドの 3 次元座標 $t = (x_t, y_t, z_t)$ をすべてのデプス画像（全 N 枚）に投影する。投影先の画像を取得したカメラ i（$i = 1, 2, ..., N$）の座標系基準のグリッド j の 3 次元座標 $\boldsymbol{g} = (g_x, g_y, g_z)$ は式 (3.9) から式 (3.11) を用いて求められる。

$$\begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix}_{i,j} = \boldsymbol{t}_i + l_j \hat{\boldsymbol{v}}_i = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}_i + \frac{l_j}{|\boldsymbol{v}_i|} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}_i \tag{3.9}$$

$$\boldsymbol{v}_i = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{pmatrix}_i = \boldsymbol{M}_i \times \hat{\boldsymbol{a}} = \boldsymbol{M}_i \times \frac{1}{|\boldsymbol{a}|} \begin{pmatrix} x_t - c_x \\ y_t - c_y \\ f_x \\ 0 \end{pmatrix} \tag{3.10}$$

$$\boldsymbol{M}_i = T_{ciw}^{-1} \times T_{fw} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}_i \tag{3.11}$$

ここで、l_j は自由視点の原点からグリッド j までの距離、T_{ciw} はカメラ i の座標系の世界座標系に対する同次変換行列、T_{fw} は自由視点座標系の世界座標系に対する同次変換行列である。また、$\boldsymbol{v}_i$ は自由視点の原点から処理対象ピクセルへのベクトル（カメラ i の座標系基準）、$\boldsymbol{a}$ は自由視点の原点から処理対象ピクセルへのベクトル（自由視点座標系基準）、f_x、f_y はカメラの焦点距離（単位はピクセル）、c_x、c_y は主点（単位はピクセル）である。

次に、評価グリッドを i 番目のデプス画像へ投影した際の座標 (x, y) は式 (3.12) およ

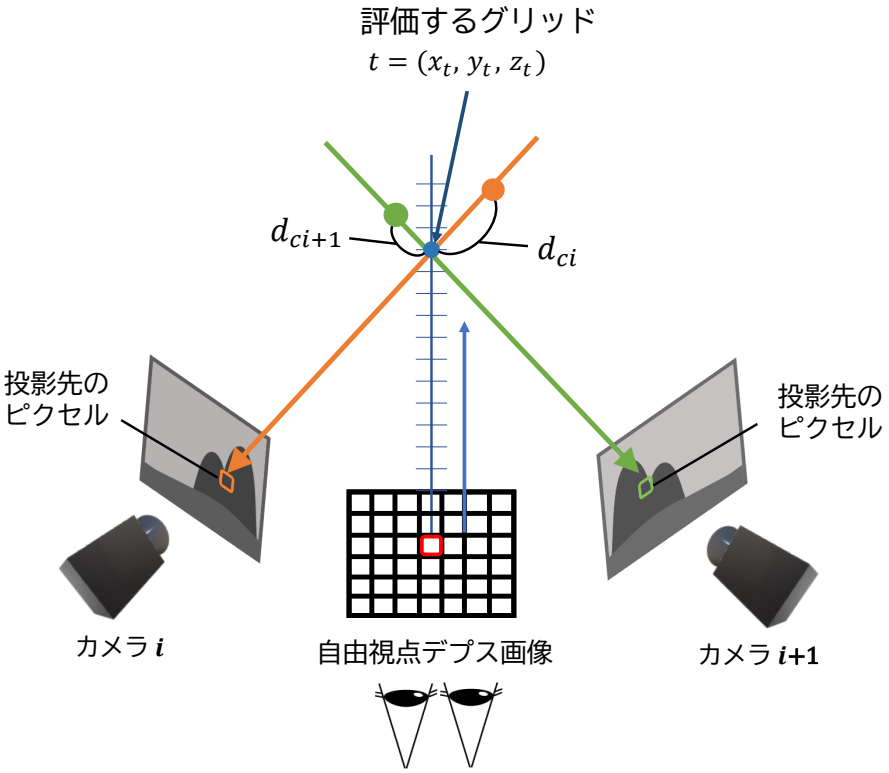


図 3.6: グリッドの評価値を求める処理

び (3.13) を用いて求められる。

$$x = c_x + f_x(g_x/g_z) \tag{3.12}$$

$$y = c_y + f_y(g_y/g_z) \tag{3.13}$$

この時、投影先の座標が画像内に存在しない場合は、その画像は次の処理に使用しない。

その後、投影先のピクセルのデプス値 z_{ci}（単位は mm）から、式 (3.14) によりそのピクセルが映す被写体上の点のカメラ i の座標系基準の 3 次元座標 $\boldsymbol{p}_{ci}$ を求める。

$$\boldsymbol{p}_{ci} = \begin{pmatrix} p_x^{ci} \\ p_y^{ci} \\ p_z^{ci} \end{pmatrix} = \begin{pmatrix} \frac{(x-c_x)z_{ci}}{f_x} \\ \frac{(y-c_y)z_{ci}}{f_y} \\ z_{ci} \end{pmatrix} \tag{3.14}$$

この 3 次元座標 $\boldsymbol{p}_{ci}$ とグリッドの 3 次元座標 $\boldsymbol{g}$ との符号付距離 d_{ci} を式 (3.15) により求める。この時、投影したピクセルが映す被写体上の 3 次元座標がグリッドよりもカメラ側にあれば符号付距離の符号は負、逆側にあれば正とする。この符号付距離の絶対値が閾値以下となる全ての距離の平均値を計算し、そのグリッドの評価値とする。

$$d_{ci} = p_z^{ci} - g_z^{ci} \tag{3.15}$$

被写体表面の推定に関しては、図 3.7 に示すように、評価値の符号が反転する部分に被写体の表面があるものとし、その座標を探索する。具体的にはまず、隣接するグリッド間で評価値の符号が反転する箇所を探索する。その後、評価値が 0 となる被写体表面の位置までの正確な距離 z を式 (3.16) に示す線形補間を用いて算出し、処理対象ピクセルの新しいデプス値とする。

$$z = z_k + \frac{A_k}{A_k - A_{k+1}} \times (z_{k+1} - z_k) \tag{3.16}$$

ここで、A_k と A_{k+1} は手前から奥に向けて隣接する k、$k+1$ 番目の 2 つのグリッドの評価値、z_k と z_{k+1} は自由視点の原点からグリッドまでの距離である。これらの処理により、各 RGB-D カメラで撮影したデプス画像を用いて、ランダムノイズを除去した自由視点デプス画像を生成する。本処理により RGB-D カメラ単体を用いて小空間内を撮影した場合よりも高解像度かつ高密度なデプス画像を生成することが可能になり、オクルージョンの考慮などの後の処理の精度を高められるだけでなく、より高密度な自由視点カラー画像を生成することが可能になる。

本アルゴリズムは、カメラの解像度やカメラと自由視点の姿勢の遠近に関わらず、カ

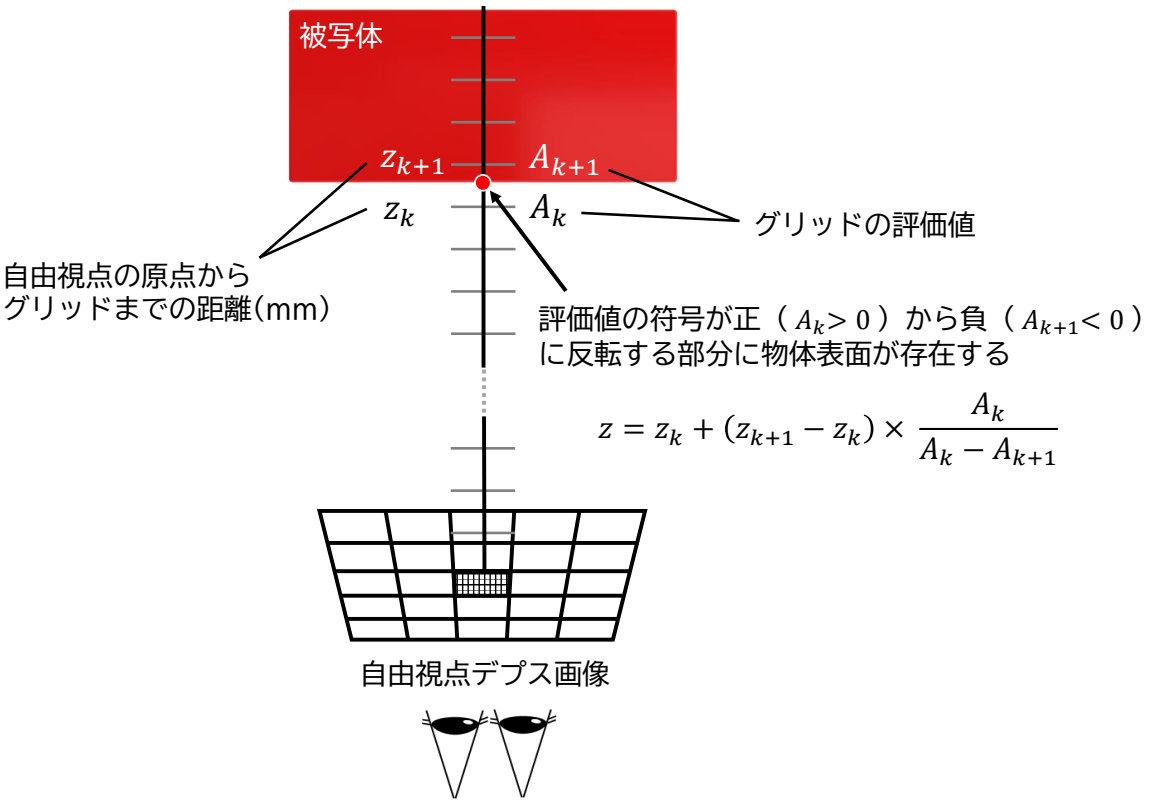


図 3.7: グリッドの 3 次元座標と評価値から距離を求める処理

メラの視野内に入ったすべてのグリッドに対してデプス値を求められるアルゴリズム（実在しないピクセルのデプス値を求めることも可能）であり、デプス画像の超解像処理に相当する。

3.6 自由視点カラー画像の生成

本提案システムでは、3.5 節で生成した自由視点デプス画像を基に、自由視点デプス画像の各ピクセルが撮影している被写体の位置を求め、複数のカラー画像の中から適切なピクセルのカラー情報を取得することで自由視点カラー画像を生成する。自由視点カラー画像を生成する処理の流れを図 3.8 に示す。はじめに、3.6.1 項で基本的なカラー情報の取得方法について説明するが、この手法のみでは、カメラから見て被写体の前に別の被写体が存在する場合、すなわちオクルージョンが発生する場合に、誤ったカラー情報が取得されてしまう場合がある。また、被写体の表面形状によっては、事前に行ったカメラのキャリブレーションの精度が不十分であることなどが原因で、生成した画像に歪みが生じる場合もある。これらの問題への対策として、3.6.2 項ではオクルージョンへの対応方法、3.6.3 項では被写体表面の法線方向を考慮したカラー情報の取得方法についてそれぞれ述べる。

3.6.1 カラー情報の取得

被写体に対するカラー情報の基本的な取得方法を図 3.9 に示す。最初に、生成した自由視点デプス画像の各ピクセルが格納しているデプス値から、式 (3.17) によりそのピクセルが映す被写体上の点の自由視点座標系基準の 3 次元座標 $\boldsymbol{p}_f$ を求める。

$$\boldsymbol{p}_f = \begin{pmatrix} x_f \\ y_f \\ z_f \end{pmatrix} = \begin{pmatrix} \frac{(x-c_x)z_f}{f_x} \\ \frac{(y-c_y)z_f}{f_y} \\ z_f \end{pmatrix} \tag{3.17}$$

ここで、x, y は処理対象ピクセルの座標、f_x、f_y はカメラの焦点距離（単位はピクセル）、c_x、c_y は主点（単位はピクセル）、z_f は処理対象ピクセルが格納しているデプス値（単位は mm）である。

次に、3.4 節で述べたキャリブレーションにより求めた各カメラの姿勢を用いて、式 (3.18) および (3.19) により自由視点座標系基準の被写体上の点の 3 次元座標 $\boldsymbol{p}_f$ を各カ

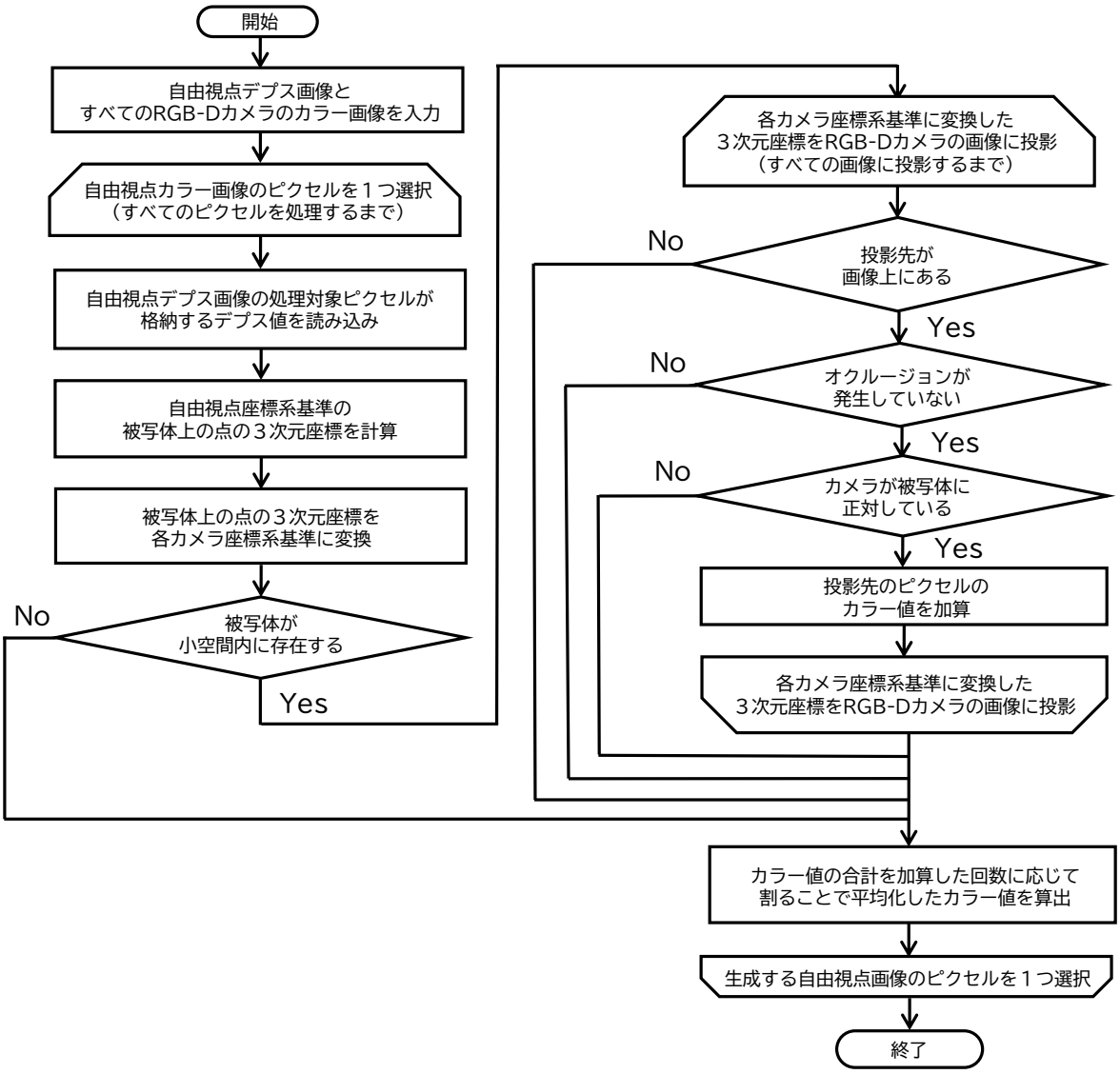


図 3.8: 自由視点カラー画像を生成する処理の流れ

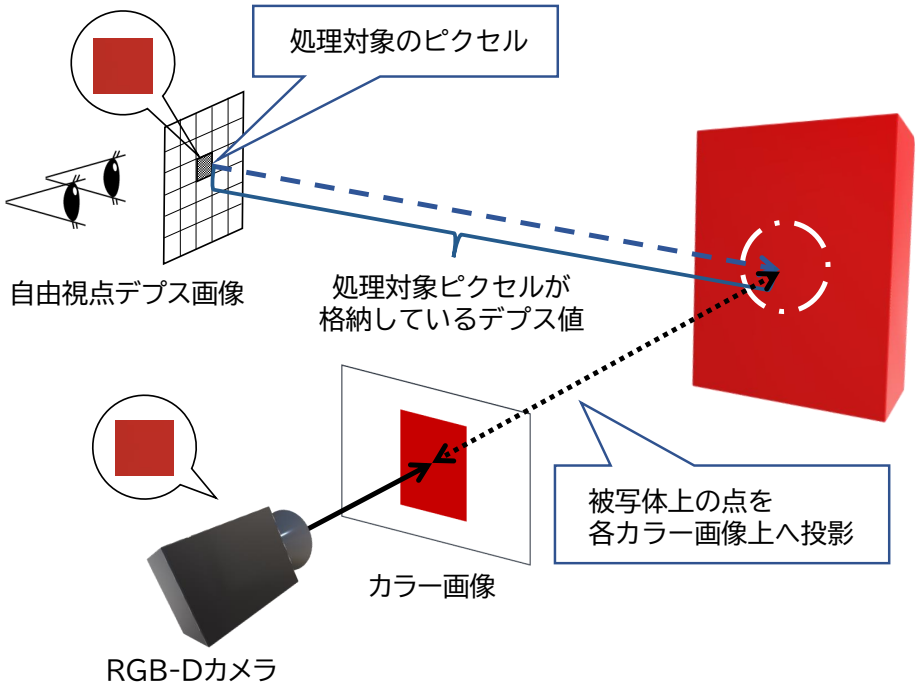


図 3.9: 被写体に対するカラー情報の基本的な取得方法

メラ座標系基準の 3 次元座標 $\boldsymbol{p}_{ci} = (p_x^{ci}, p_y^{ci}, p_z^{ci})$ に変換する。

$$\boldsymbol{p}_{ci} = \begin{pmatrix} p_x^{ci} \\ p_y^{ci} \\ p_z^{ci} \\ 1 \end{pmatrix} = \boldsymbol{M}_{ci} \times \begin{pmatrix} x_f \\ y_f \\ z_f \\ 1 \end{pmatrix} \tag{3.18}$$

$$\boldsymbol{M}_{ci} = T_{ciw}^{-1} \times T_{fw} \tag{3.19}$$

本システムは小空間への没入体験の実現を目的とするため、各カメラで撮影される空間のカラー情報のうち、必要となる情報は小空間内のカラー情報のみである。したがって、式 (3.18) および (3.19) により求めた各カメラ座標系基準の被写体上の点の 3 次元座標 $\boldsymbol{p}_i$ を基に、没入対象となる小空間内に存在する被写体に対してのみカラー情報を取得する処理とする。

その後、変換した各カメラ座標系基準の 3 次元座標を各カラー画像に投影し、投影先のピクセルのカラー値（R（赤）、G（緑）、B（青）の 3 原色で表された色の値）を求める。変換した各カメラ座標系基準の 3 次元座標を i 番目のカラー画像へ投影した際

の座標 (x, y) は式 (3.20) および (3.21) を用いて求められる。

$$x = c_x + f_x(p_x^{ci}/p_z^{ci}) \tag{3.20}$$

$$y = c_y + f_y(p_y^{ci}/p_z^{ci}) \tag{3.21}$$

この時、投影先の座標が画像内に存在しない場合は、その画像は処理に使用しない。

最後に、投影先の各ピクセルのカラー値を用いて、処理対象ピクセルのカラー値を算出する。その際、より正確なカラー値を求めるために、投影先の各ピクセルのカラー値から外れ値を排除する方法や、投影先の座標がピクセルの中心に最も近くなる画像のカラー値のみを使用する方法などが考えられる。しかし、本システムでは、使用するカメラがボックスの周囲に様々な姿勢で取り付けられているため、環境光の影響により、取得されるカラー画像の色味がカメラごとに異なる。そのため、外れ値の排除は適切な判断が難しく、また、ピクセルごとに異なるカメラの画像を参照する方法では、画像ごとの色味の違いが影響し、最終的な自由視点画像の色が不自然になる場合がある。そこで本手法では、カラー値の合計値を加算した回数に応じて割ることで平均化したカラー値を算出し、処理対象ピクセルのカラー値とする。

3.6.2 オクルージョンへの対応

本システムを使用する際には、没入対象となる小空間内に複数の被写体や凹凸のある複雑な形状の被写体が存在する状況が想定される。その際、3.6 節で述べたように、ある視点から見た場合に複数の被写体が重なり、手前の被写体が後ろに存在する被写体を隠してしまうオクルージョンが発生することがある。オクルージョンが発生している例を図 3.10 に示す。この状況下で、3.6.1 項で述べたカラー情報の取得方法を用いた場合を考える。カラー情報を取得しようとしている自由視点デプス画像の処理対象ピクセルが格納しているデプス値 z_f から求めた自由視点座標系基準の点の 3 次元座標 $\boldsymbol{p}_f$ は図 3.10 の赤い被写体上の点であり、処理対象ピクセルに対して本来取得されるべきカラー情報は赤色を示すカラー情報である。一方、自由視点座標系基準からカメラ座標系基準に変換した被写体上の点の 3 次元座標 $\boldsymbol{p}_{ci}$ をカラー画像上へ投影した場合、投影先のカラー画像のピクセルの持つカラー情報は青色を示すカラー情報となる。これは、カラー情報の取得先となるカメラから見ると、手前に存在する青い被写体が後ろに存在する赤い被写体を隠してしまっているためである。したがって、3.6.1 項で述べたカラー情報の取得方法をそのまま用いた場合、本来赤色を示すカラー情報が取得

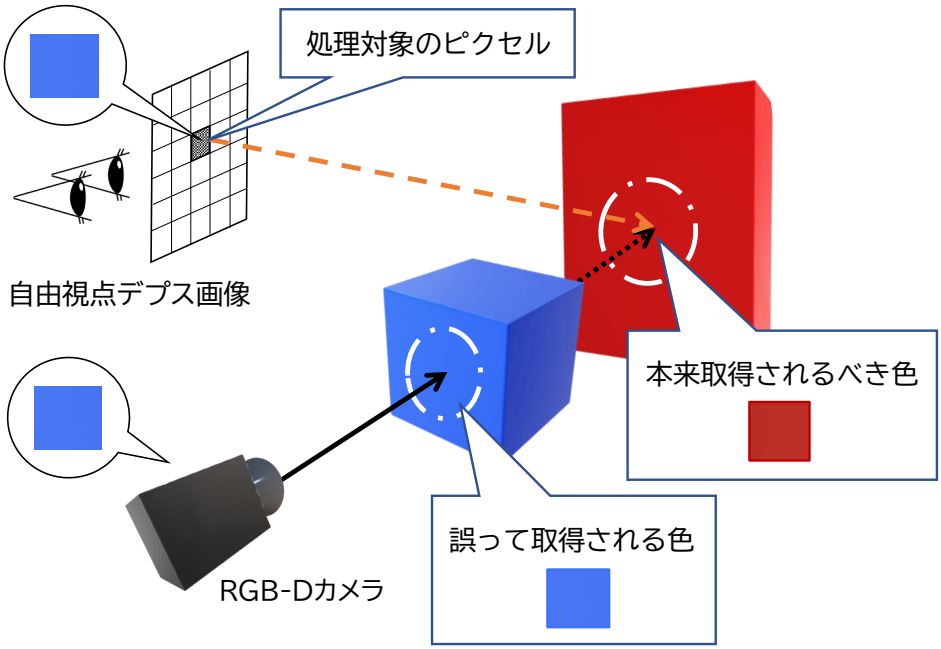


図 3.10: オクルージョンが原因で誤ったカラー情報が取得される例

されるべきピクセルに対して実際に取得されるのは青色を示すカラー情報となってしまう。

このような状況を避けるために、本研究では、取得したデプス画像を用いてオクルージョンへの対応を行った。オクルージョンへの対応方法を図 3.11 に示す。最初に、式 (3.20) および (3.21) を用いて求めた、カラー画像への投影先の座標 (x, y) を基に、投影先のカラー画像を取得したカメラのデプス画像の (x, y) の座標のピクセルが格納しているデプス値を求め、d_1 とする。次に、式 (3.18) および (3.19) を用いて求めた、各カメラ座標系基準の被写体上の点の 3 次元座標 $\boldsymbol{p}_{ci}$ の z 座標の値 p_z^{ci} を d_2 とする。この時、以下の式を満たす場合にはカラー値を加算しない。

$$d_1 < d_2 - d_{th} \tag{3.22}$$

ここで、d_{th} はカラー値の加算の可否の判断に用いる閾値である。つまり、取得したデプス画像から取得されたデプス値の値 d_1 が、各カメラ座標系基準に変換した点の z 座標の値 d_2 と比べて閾値 d_{th} よりも小さい場合にはオクルージョンが発生しているとみなし、カラー値を加算しないことにする。この処理により、オクルージョンの発生に伴う誤ったカラー値の影響を排除し、生成画像の色の誤りの軽減を試みる。

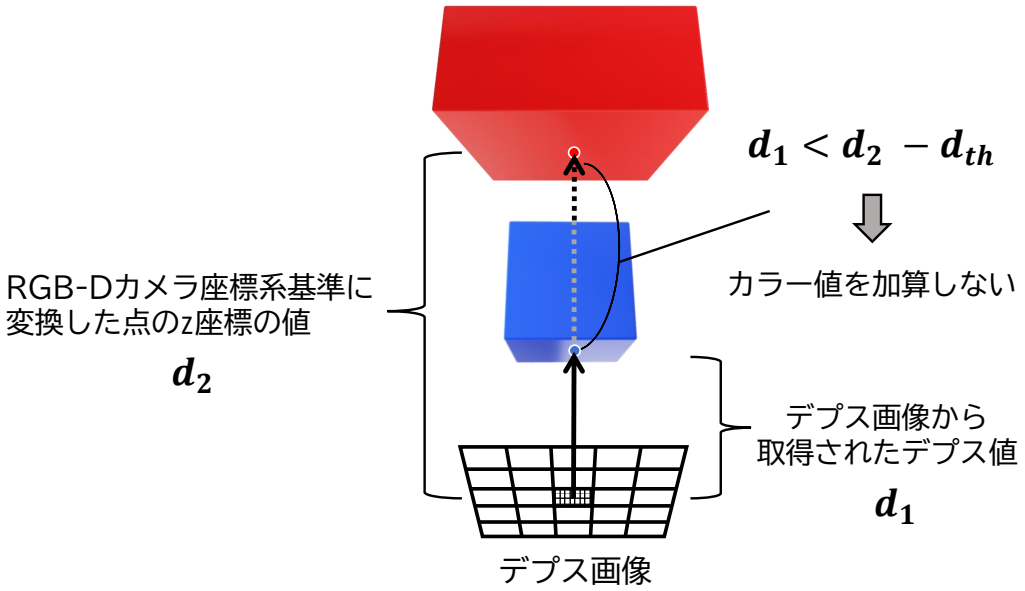


図 3.11: オクルージョンへの対応

3.6.3 被写体表面の法線方向の考慮

本提案システムで使用される被写体としては、様々な表面形状を持つものが想定される。その際、カメラの姿勢と被写体表面の法線方向との関係によっては適切なカラー値を取得することが困難となる場合がある。被写体表面の法線方向の考慮が必要となる例を図 3.12 に示す。被写体表面がカメラに正対している場合、すなわち、カメラの撮影方向と被写体表面の法線方向のなす角の大きさが大きい場合は取得したカラー情報に誤りが生じにくい。一方、被写体表面がカメラに正対していない場合、すなわち、カメラの撮影方向と被写体表面の法線方向のなす角の大きさが小さい場合は 3.4 節で述べたキャリブレーションの精度が不十分であることなどが原因で、取得したカラー情報に誤りが生じやすい。ここで、取得したカラー情報に誤りが生じやすいとは、各カラー画像上へ投影した際、投影先のずれがわずかであっても取得されるカラー情報に影響が出やすいことを示す。具体的には、図 3.12 に示す状態において、カラー画像上への投影先がわずかでもずれた場合、取得されるカラー情報が大きく異なることになり、誤ったカラー情報の取得により生成する自由視点画像にひずみが生じたように見えてしまう原因となってしまう。

このような状況を避けるために、本研究では 3.6.2 項で述べたオクルージョンへの対応に加え、カメラの撮影方向と被写体表面の法線方向との関係も考慮した。先述のように、カメラの撮影方向と被写体表面の法線方向のなす角の大きさが小さい場合、適

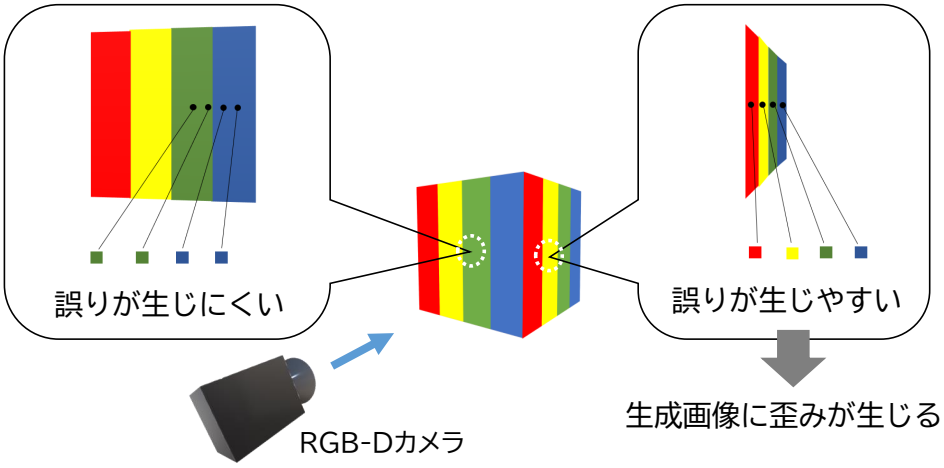


図 3.12: 被写体表面の法線方向の考慮が必要となる例

切なカラー情報を取得することが困難になる。そこで、カメラの撮影方向ベクトルと被写体表面の法線ベクトルを計算し、2 つのベクトルのなす角の大きさが一定値以下である場合にはカラー値を加算しないことにする。図 3.13 に 2 つのベクトルとそのなす角の定義を、図 3.14 および図 3.15 にカメラの撮影方向と被写体表面の法線方向を基にカラー値の加算の可否を判断する方法を示す。図 3.14 では、カメラの撮影方向ベクトルと被写体表面の法線ベクトルの間のなす角の大きさが一定値よりも大きいため、カラー値を加算する。一方、図 3.15 では、カメラの撮影方向ベクトルと被写体表面の法線ベクトルの間のなす角の大きさが一定値以下であるため、適切なカラー情報の取得が困難であるとみなしカラー値を加算しない。

2 つのベクトルのなす角の大きさを判断する方法として、各カメラ座標系において、カメラの単位方向ベクトル（カメラの撮影方向ベクトルを正規化したもの）と被写体表面の単位法線ベクトルの内積を求めて利用する方法が考えられる。すなわち、図 3.13 に示すように、カメラの単位方向ベクトルを $\boldsymbol{m}$、被写体表面の単位法線ベクトルを $\boldsymbol{n}$、2 つのベクトルのなす角の大きさを θ としたとき、2 つのベクトルの内積は以下の式で表される。

$$\boldsymbol{m} \cdot \boldsymbol{n} = \cos\theta \tag{3.23}$$

ここで、予め閾値とする $\cos\theta$ の値を求めておき、この値とベクトルの内積の値を比較する方法が考えられるが、カメラの単位方向ベクトルは $\boldsymbol{m} = (0, 0, 1)$ であるから、被写体表面の単位法線ベクトルを $\boldsymbol{n} = (a, b, c)$ としたときの内積は以下の式で表される。

$$\boldsymbol{m} \cdot \boldsymbol{n} = c \tag{3.24}$$

式 (3.23) と式 (3.24) より、カメラの単位方向ベクトルと被写体表面の単位法線ベクトルのなす角の大きさ θ と被写体表面の単位法線ベクトルの z 座標値である c との関係は $\cos\theta = c$ と表せるため、被写体表面の法線方向を考慮したカラー値の加算の可否の条件として、被写体表面の単位法線ベクトルの z 座標値による判断が可能となる。したがって、2 つのベクトルのなす角の大きさが一定値以下である場合、つまり被写体表面の単位法線ベクトルの z 座標値である c が閾値以上である場合にはカラー値を加算しないことにする。この処理により、カメラの姿勢と被写体表面の法線方向の関係に伴う誤ったカラー値の影響を排除し、生成画像の歪みの低減を試みる。

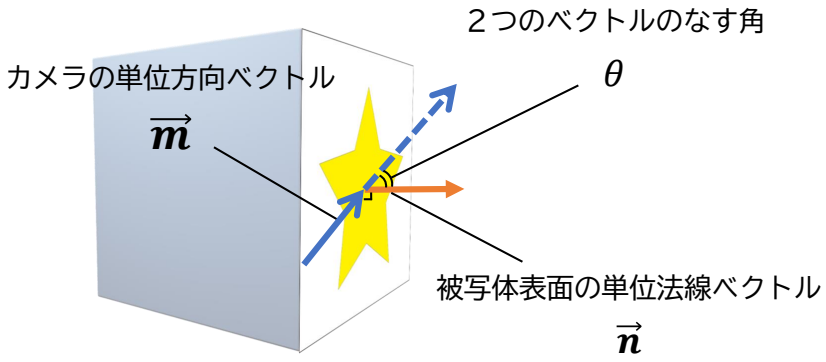


図 3.13: 2 つのベクトルとそのなす角の定義

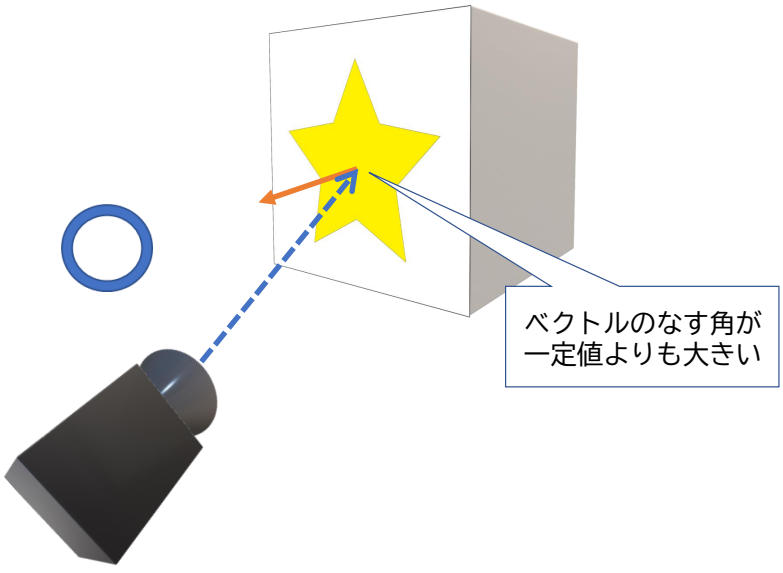


図 3.14: カメラの撮影方向と被写体表面の法線方向を基にカラー値の加算の可否を判断する方法（カラー値を加算する場合）

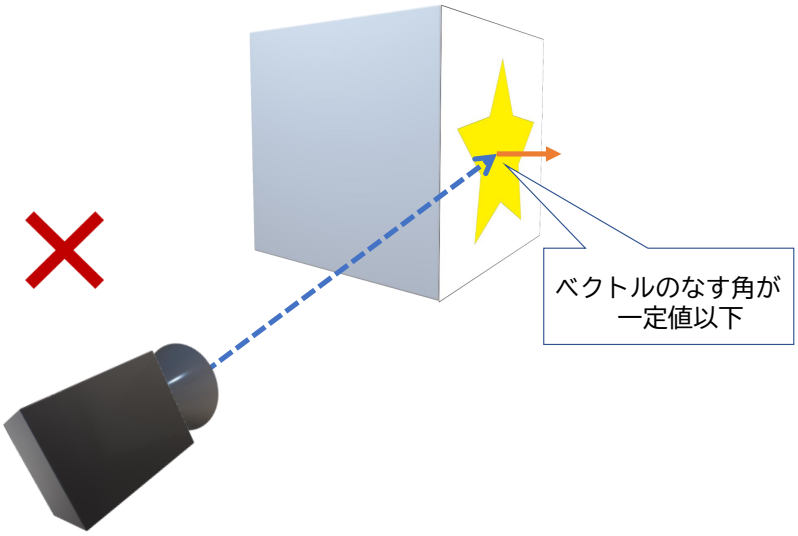


図 3.15: カメラの撮影方向と被写体表面の法線方向を基にカラー値の加算の可否を判断する方法（カラー値を加算しない場合）

第 4 章 高精細立体映像提示システムの実装

4.1 ハードウェアの実装

提案システムのハードウェア構成を図 4.1 に、実際に使用したボックスの外観とカメラの配置をそれぞれ図 4.2 および図 4.3 に示す。本研究で没入対象とする小空間には、0.9m 四方のボックス内空間を設定した。ボックスは 0.9m の L 型アングル材を使用して作成した。ボックス内撮影用のカメラには、3.3 節で述べたように、アクティブ IR ステレオ方式の Intel RealSense Depth Camera D435[37] を 12 台使用し、カメラの固定具は 3D プリンタの MakerBot Replicater+を使用して作成した。各カメラの配置について、本システムでは、システムの体験時に小空間内の任意の位置に視点を移動可能にするために、小空間内の環境を様々な方向から抜け落ちなく撮影する必要がある。そのため、ボックスの周囲に満遍なくカメラを配置する方が望ましいが、一方で、システムの体験時には複数人でボックス内の被写体を操作しながら体験することを想定するため、ボックス内部への体験者のアクセスのしやすさも重要となる。そこで、本研究では、図 4.2 のように、ボックス側面のうちの 2 面に 6 台ずつのカメラを取り付け、残りの 2 面からボックス内部へアクセスする配置とした。加えて、小空間に対する撮影の抜け落ちを減らすため、カメラの取り付け場所としては、図 4.3 の左に示すように、ボックス内を様々な視点から撮影できるように配置し、取り付け角度については、図 4.3 の中央および右に示すように、ボックス内環境がカメラの画角内に収まるよう調整した。使用するボックスのサイズについては、使用した RealSense の最小深度距離（デプス情報を取得できる最短の距離）が 0.1〜0.2m であるため、ボックスのサイズが小さすぎる場合、カメラと被写体の距離が近づくことでデプス画像の取得が困難になる。加えて、ボックス内部への体験者のアクセスも制限されてしまう可能性がある。以上の理由から、本研究では、ボックスのサイズを 0.9m 四方に設定した。

3.1 節で述べたカメラのキャリブレーションでは、追加で 1 台のカメラとシングルボードコンピュータを搭載した杖型デバイスを作成して画像を撮影した。使用した杖型デバイスの外観を図 4.4 に示す。キャリブレーション用のカメラには Intel RealSense Depth Camera D435 を、シングルボードコンピュータには Raspberry Pi 4 Model B（以下、RaspberryPi）を使用した。杖型デバイスを作成した理由や個々のハードの使用

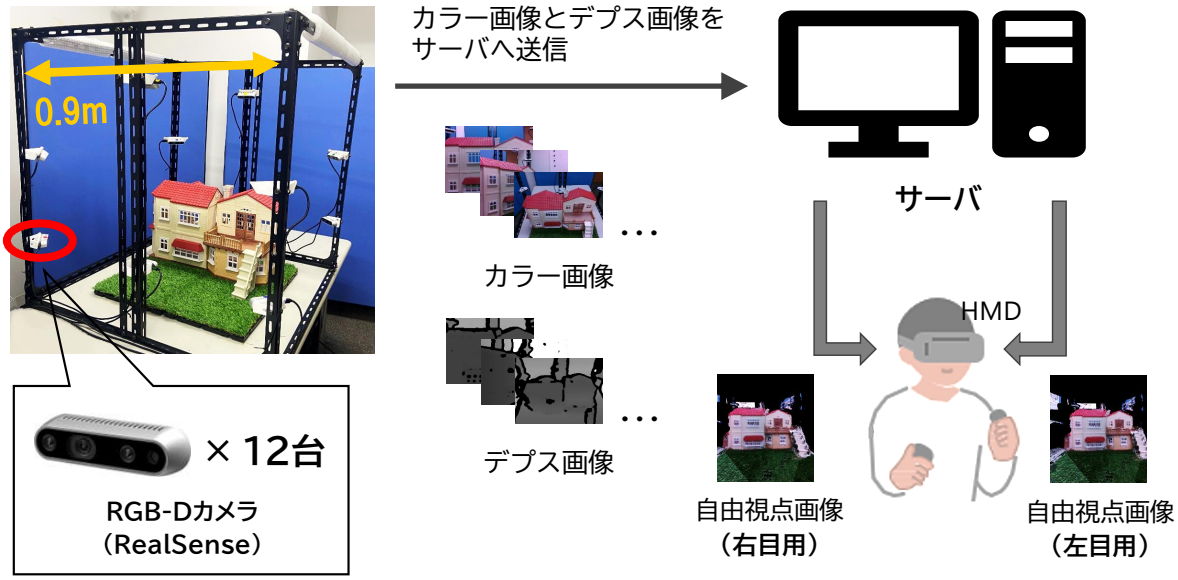


図 4.1: システムの構成図

図 4.2: システムで使用したボックスの外観

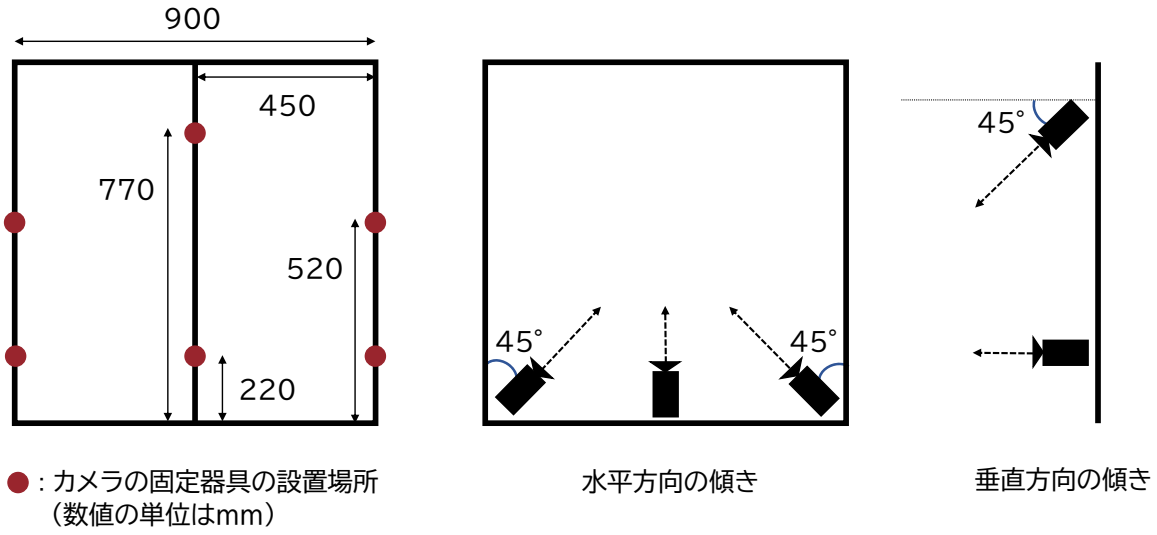


図 4.3: RGB-D カメラの設置位置

理由等については 4.2.1 項で述べる。 HMD には、装着したまま自由視点画像のみが見えるモードと外界を見ることができるモードを切り替え可能な Meta Quest 社の Meta Quest3 を使用した。図 4.4〜4.5 および表 4.1 にボックス内撮影用カメラとキャリブレーション用カメラの外観および仕様を、表 4.2 に使用したシングルボードコンピュータの仕様を、図 4.6 および表 4.3 に使用した HMD とコントローラの外観および HMD の仕様を示す。表 4.1 に示したボックス内撮影用カメラの設定について、フレームレートに関しては、本システムでの処理が可能なフレームレートが約 10〜11fps であることが後の評価で分かっており、その値より小さい値かつ RealSense で設定可能な値として 6fps に設定した。また、カラー画像およびデプス画像の解像度に関して、RealSense では、カラー画像は最大 1920 × 1080 ピクセル、デプス画像は最大 1280 × 720 ピクセルまで設定可能であるが、解像度を上げると画像処理の負荷も高くなり、処理速度が低下する。本研究では、4.2.1 項で後述するように、カメラのキャリブレーションの精度に限界があり、撮影する画像の解像度を上げても生成画像の精度に大きな向上は見られなかった。そこで、処理速度の向上を優先し、カラー画像およびデプス画像の解像度を 640 × 480 ピクセルに設定した。本システムでは、処理速度を向上さるために、RealSense を用いてカラー画像とデプス画像を撮影してサーバ側の PC に送信するクライアント用 PC と、受け取った画像を用いて両目用の自由視点画像をそれぞれ個別に生成して HMD 上に提示するサーバ用 PC に処理を分散させた。さらに、12 台のカメラの画像を処理するには 1 台の PC では負荷が高かったため、クライアント用 PC を 2 台用意し、表 4.4 に示す仕様の USB 拡張カードを使用してそれぞれ 6 台ずつのカメラの

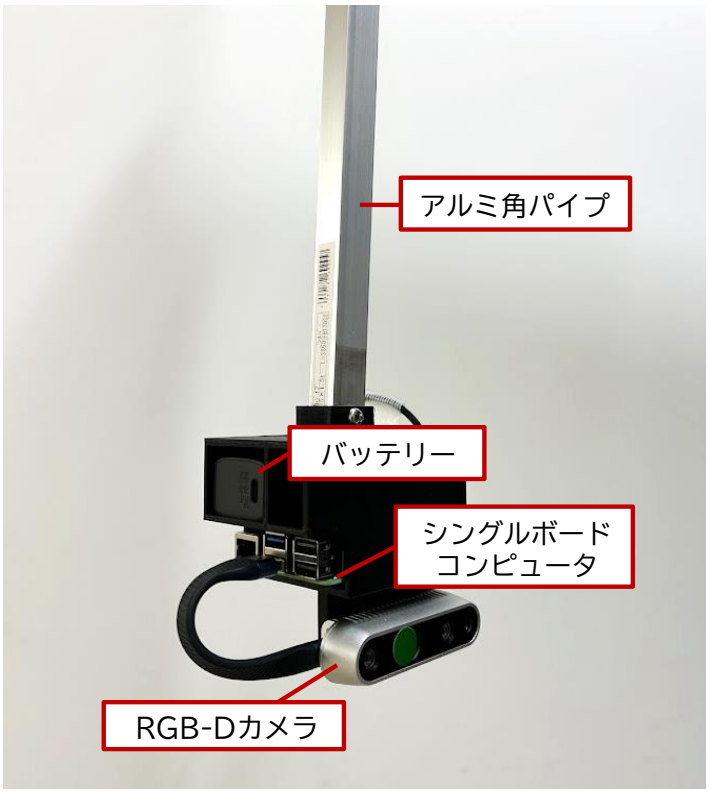


図 4.4: キャリブレーション用画像の撮影に使用した杖型デバイスの外観

図 4.5: システムで使用した RGB-D カメラの外観

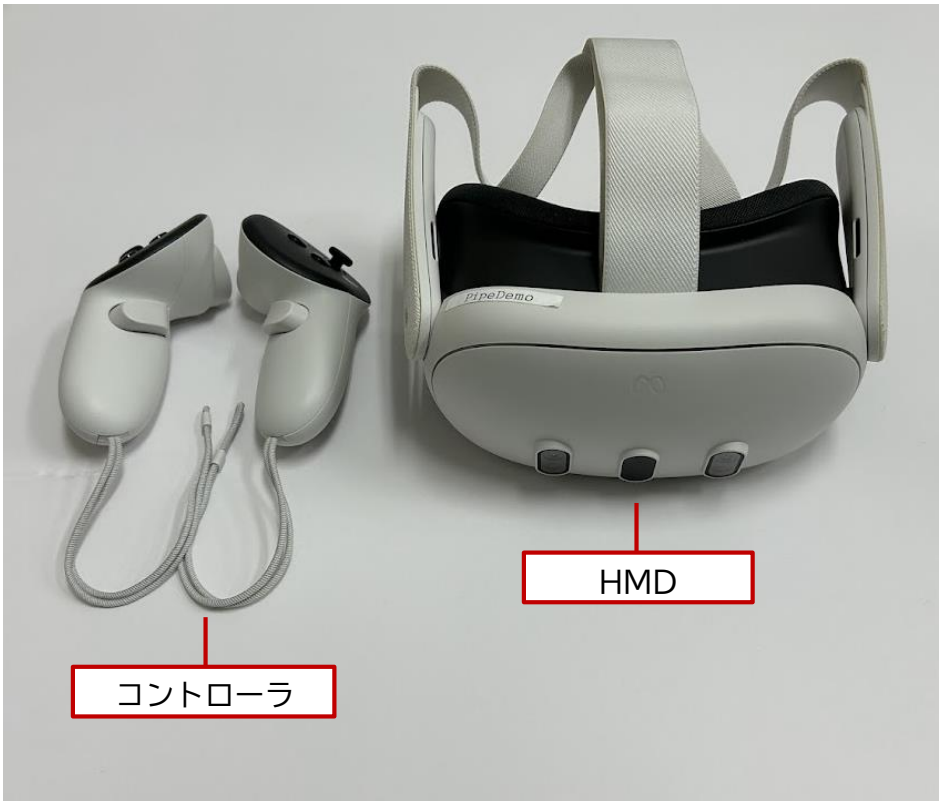


図 4.6: システムで使用した HMD とコントローラの外観

画像を処理した。クライアント用 PC2 台それぞれの仕様を表 4.5 および表 4.6 に、サーバ用 PC1 台の仕様を表 4.7 に示す。加えて、PC 間での画像の送受信の際の通信速度を向上させるため、表 4.8 および表 4.9 に示す仕様の LAN カードおよび表 4.10 に示す仕様のスイッチングハブを使用した。

また、1 章で述べたように、本システムを用いた体験として、地震や水害等の災害体験や街の防災計画立案等を想定しているため、これらの体験の実現に使用する被写体として家の模型を用意した。加えて、システムの性能（生成画像の解像度や、オクルージョンおよび被写体形状への対応）を評価するために使用する大き目の被写体と小さ目の被写体として、2 体のロボットを用意した。使用した家の模型と 2 体のロボットの外観および仕様を図 4.7〜4.8 および表 4.11 に示す。

表 4.1: システムで使用した RGB-D カメラの仕様

製品名	Intel RealSense Depth Camera D435					
	ボックス内撮影用カメラ					
	color			depth		
	解像度	fps	FOV	解像度	fps	FOV
1 台目	640 × 480	6	55.61° × 43.18°	640 × 480	6	79.97° × 64.34°
2 台目	640 × 480	6	55.11° × 42.74°	640 × 480	6	80.12° × 64.47°
3 台目	640 × 480	6	54.86° × 42.51°	640 × 480	6	79.43° × 63.84°
4 台目	640 × 480	6	54.96° × 42.62°	640 × 480	6	79.34° × 63.76°
5 台目	640 × 480	6	55.07° × 42.67°	640 × 480	6	79.68° × 64.07°
6 台目	640 × 480	6	54.96° × 42.62°	640 × 480	6	79.71° × 64.10°
7 台目	640 × 480	6	55.60° × 43.17°	640 × 480	6	79.72° × 64.11°
8 台目	640 × 480	6	55.02° × 42.74°	640 × 480	6	79.84° × 64.22°
9 台目	640 × 480	6	55.63° × 43.18°	640 × 480	6	79.97° × 64.34°
10 台目	640 × 480	6	55.29° × 42.96°	640 × 480	6	79.58° × 63.98°
11 台目	640 × 480	6	55.55° × 43.21°	640 × 480	6	79.81° × 64.20°
12 台目	640 × 480	6	55.39° × 43.05°	640 × 480	6	79.64° × 64.04°
	キャリブレーション用カメラ					
	color			depth		
	解像度	fps	FOV	解像度	fps	FOV
	640 × 480	60	54.90° × 42.57°	640 × 480	60	79.65° × 64.04°

表 4.2: システムで使用したシングルボードコンピュータの仕様

製品名	Raspberry Pi 4 Model B
CPU	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
メモリ	8GB

表 4.3: システムで使用した HMD の仕様

製品名	Meta Quest3
解像度	2064 × 2208 (片目あたり)
FOV	110° × 96°
CPU	Snapdragon XR2 Gen2
RAM	8GB

表 4.4: クライアント用 PC に使用した USB 拡張カードの仕様

製品名	Inateck 16Gbps PCIe-USB 3.2 Gen 2
ポート数	USB Type-C 3.2 Gen 2 ポート × 2
	USB Type-A 3.2 Gen 2 ポート × 6
伝送速度	最大 16Gbps

表 4.5: クライアント用 PC の仕様 (1 台目)

CPU	Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
GPU	Intel(R) UHD Graphics 630
メモリ	32.0GB
OS	Windows 10 Pro 64bit

表 4.6: クライアント用 PC の仕様 (2 台目)

CPU	Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
GPU	Intel(R) UHD Graphics 630
メモリ	32.0GB
OS	Windows 10 Pro 64bit

表 4.7: サーバ用 PC の仕様

CPU	Intel(R) Core(TM) i7-14700K
GPU	NVIDIA GeForce RTX 4090
メモリ	64.0GB
OS	Windows10 Pro 64bit

表 4.8: クライアント用 PC に使用した LAN カードの仕様

製品名	GBE2.5i-PCIE
ポート数	RJ-45 外部 1 ポート
通信規格	最大 2.5GBase-T

表 4.9: サーバ用 PC に使用した LAN カードの仕様

製品名	GBE2.5i-PCIE
ポート数	1 ポート（AUTO-MDIX 対応）
伝送速度（規格値）	10Gbps(最大 10GBase-T)

表 4.10: システムで使用したスイッチングハブの仕様

製品名	LXW-10G2/2G4
ポート数	6 ポート
伝送速度（規格値）	10Gbps(最大 10GBase-T)

図 4.7: 被写体として用いた家の模型の外観

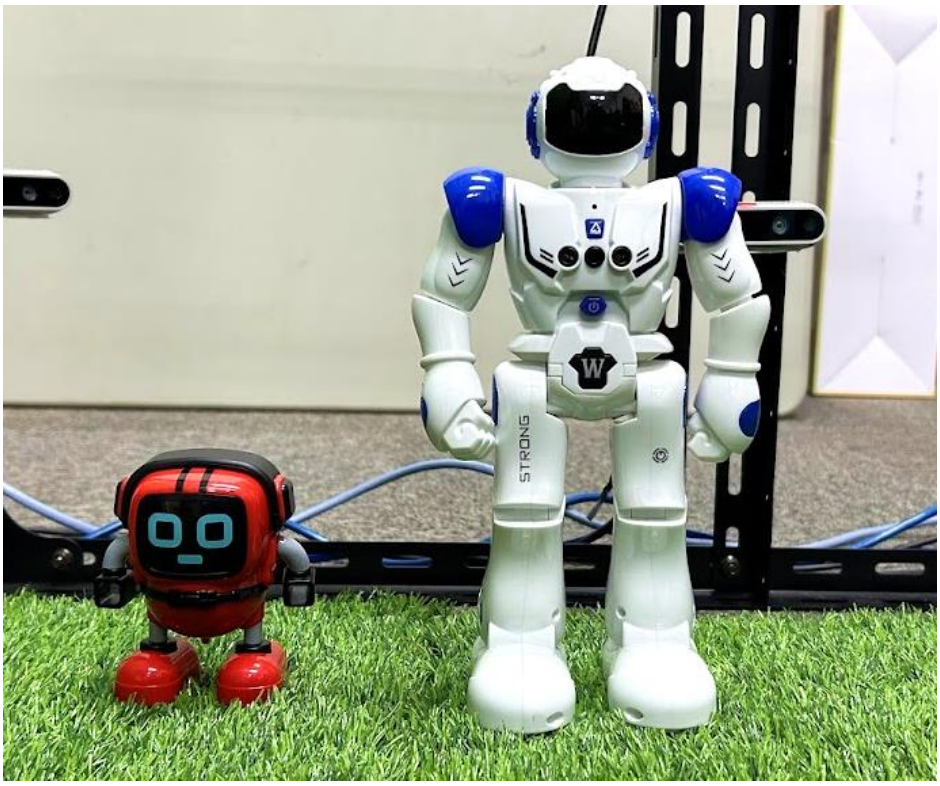


図 4.8: 被写体として用いたロボットの外観

表 4.11: 被写体として用いた家の模型とロボットの仕様

家の模型	Sylvanian Families ドールハウス 220 × 385 × 585 mm
白色のロボット	AUGYMERMULTIFUNCTIONALROBOTHT9930-1 147 × 82 × 253 mm
赤色のロボット	urban field ジャイロバトルロボット 303002-1 80 × 45 × 96 mm

4.2　ソフトウェアの実装

提案システムの処理は、3.1 節で述べたように、使用するカメラ間の相対姿勢を求めるオフライン処理と、両目用の自由視点画像を生成して HMD 上に立体映像として提示するオンライン処理に分けられる。オフライン処理ではカメラのキャリブレーションを実施する。オンライン処理では、RealSense を用いたカラー画像とデプス画像の撮影および送受信と、両目用の自由視点画像の生成および HMD 上への提示を行う。この際、4.1 節で述べたように、これらの処理を 2 台のクライアント用 PC と 1 台のサーバ用 PC の、合計 3 台の PC に分散させることで処理速度を向上させた。それぞれのクライアント用 PC では、カラー画像とデプス画像の撮影および送信のためのアプリケーション（以下，キャプチャアプリ）を実行する。サーバ用 PC では、撮影されたカラー画像とデプス画像を受信し、3 章で述べた手法により、右目用と左目用の自由視点画像をそれぞれ個別に生成して HMD 上に提示するアプリケーション（以下、体験アプリ）を実行する。以下では、カメラのキャリブレーション、キャプチャアプリ、および体験アプリのそれぞれの処理の実装について詳述する。

4.2.1　カメラのキャリブレーション

本研究で提案するシステムでの体験の質を向上させるには、小空間の周囲に配置したカメラ間の相対的な姿勢（外部パラメータ）とカメラの焦点距離やレンズ歪（内部パラメータ）を高精度に取得する必要がある。そのため、本研究では、複数のキャリブレーションを検討し、可能な限り高精度にキャリブレーションを実施することを試みた。具体的には、3.4 節で述べたように、Multical、MVE、および ORB SLAM3 の 3 つのソフトウェアを使用してキャリブレーションを実施する方法を検討した。Multical は、マーカを用いて複数のカメラの外部パラメータおよび内部パラメータを推定するソフトウェアである。MVE は、複数の視点からの画像を用いて 3 次元モデルを再構成するソフトウェアである。ORB SLAM3 は、1 台のカメラで連続して撮影された画像を用いて、各画像を取得した際のカメラ姿勢を推定するソフトウェアである。以下では、それぞれのソフトウェアを使用してキャリブレーションを実施する具体的な方法について詳述する。

4.2.1.1 Multical を用いたキャリブレーション方法

Multical は、形状やテクスチャが既知のマーカを複数のカメラで様々な角度から同時に撮影し、各カメラの外部パラメータと内部パラメータを推定するソフトウェアである。本研究での Multical を用いたキャリブレーションでは、使用するカメラの内部パラメータと外部パラメータを同時に推定した。本キャリブレーション方法の検討時には、図 4.9 に示すように、デプス画像を取得するための RGB-D カメラと、高解像度のカラー画像を取得するための RGB カメラの 2 種類のカメラを使用し、これらのカメラで取得した画像を提案システムに使用した。RGB-D カメラに加えて RGB カメラも使用した理由は、RGB カメラで取得した高解像度のカラー画像を用いることで、生成する自由視点画像の実写性を高めることを試みたためである。本処理では、最初に Multical を用いて各カメラの大まかな内部パラメータと外部パラメータを取得した。しかし、Multical のみでは十分な精度が得られなかったため、Multical で得られた結果を初期値として使用し、独自のキャリブレーションを追加で実施することで、カメラの内部パラメータと外部パラメータの最適化を行った。

Multical を用いたキャリブレーションでは、最初に、図 4.10 に示したパターンの異なる 2 枚のマーカを A3 サイズに印刷して図 4.11 のように 350mm × 350mm × 450mm のサイズの直方体のアクリルボックスに貼り付け、このボックスを動かしながら、複数のカメラで様々な角度からマーカを撮影した。この際、普通紙を用いてマーカを印刷

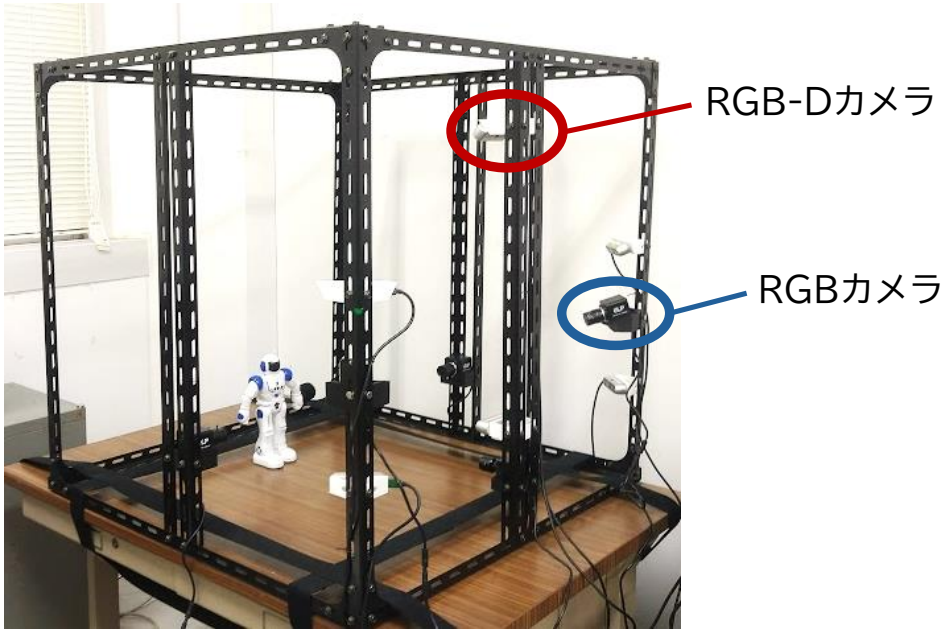


図 4.9: Multical を用いたキャリブレーション方法検討時のシステム環境

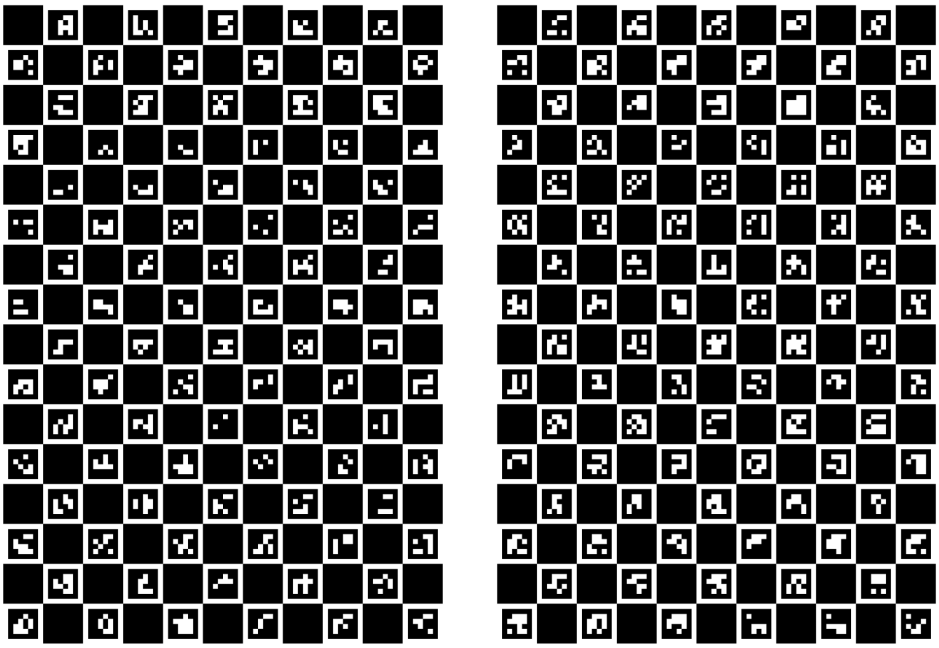


図 4.10: Multical を用いたキャリブレーションで使用した 2 枚のマーカ

図 4.11: Multical を用いたキャリブレーションに使用したアクリルボックス

RGB-Dカメラのカラー画像

RGBカメラの画像

図 4.12: RGB-D カメラおよび RGB カメラで撮影したキャリブレーション用画像の例

したところ、光の反射の影響を受けて一部のマーカが正しく認識されなかった。そこで、光の反射の影響を防ぐため不織布を用いて印刷したところ、湿気により不織布にシワが生じてしまい、マーカの認識に支障をきたした。そのため、本研究では、印刷紙として耐水紙を使用し、光の反射と湿気の影響を抑えた。追加で用いた RGB カメラには市販の USB カメラ（ELP-USB8MP02G-BFV-JP）を使用し、取得したカラー画像の解像度は 1280 × 960 ピクセルであった。撮影した画像の枚数は各 RGB-D カメラおよび RGB カメラあたりそれぞれ 109 枚であった。図 4.12 に RGB-D カメラおよび RGB カメラで撮影したキャリブレーション用画像の例を示す。

追加で実施したキャリブレーションでは、ディスプレイ上に表示されたランダムなドットパターンを同時に撮影した複数の画像を用いてカメラのパラメータを最適化した。本処理では、撮影対象空間内にディスプレイを配置し、後の画像処理を容易にするために撮影環境を暗くした上でディスプレイ上のランダムな 2 次元位置に 1 つのドットを表示し、このドットを各カメラで同時に撮影した画像を複数用意した。この時、実際にディスプレイに表示したドットのディスプレイ上での 2 次元位置も記録した。はじめに、ランダムなドットの初期 3 次元位置と撮影した画像上で認識されたドットの 2 次元位置から再投影誤差を計算し、再投影誤差が小さくなるようにドットの推定 3 次元位置を更新した。この時、最適化の処理で使用するカメラのパラメータの初期値に

表 4.12: ドットパターンの表示に使用したディスプレイの仕様

製品名	Diamondcrysta RDT234WLM
モニタサイズ	23 型（58.4cm）
解像度	1920 × 1080
画素ピッチ	0.265mm
視野角	左右 170°、上下 160°（コントラスト比 10）
コントラスト比	8000:1（CRO 非動作時 1000:1）

は Multical を使用して求めたカメラのパラメータを使用し、ドットの初期 3 次元位置はカメラ前方のランダムな 3 次元位置とした。次に、外部パラメータとドットの推定 3 次元位置を同時に最適化し、その後、内部パラメータである焦点距離を最適化した。最後に、すべてのパラメータに対して再度最適化処理を行うことで、内部パラメータと外部パラメータのそれぞれについて最適化された値を求め、本研究で用いるカメラのパラメータとした。ドットパターンの表示に使用したディスプレイの仕様を表 4.12 に示す。撮影したドット画像の枚数は各 RGB-D カメラおよび RGB カメラあたりそれぞれ 71 枚であった。ディスプレイに表示したドットの例を図 4.13 に、RGB-D カメラおよび RGB カメラで撮影したドットの画像の例を図 4.14 に示す。

Multical を用いたキャリブレーションおよびドットパターンを用いた最適化処理の結果を用いて自由視点画像を生成したところ、先述のように、Multical で得られたキャリブレーション結果のみでは十分な精度が得られなかった。また、ドットパターンを用いた最適化処理を追加で実施することにより精度は向上したものの、十分な精度には達しなかった。この要因としては、提案システムにおけるカメラ配置では共通のマーカを複数のカメラで同時に撮影することが難しく、Multical での処理の際に共通で認識されたマーカの数が不十分であったことが挙げられる。加えて、追加で使用した RGB カメラの画像の歪みが大きかったことも要因として挙げられる。また、使用するカメラの台数に関して、コストや処理時間の増加の観点からは使用するカメラの台数は少ない方が好ましい。そのため、以降では、提案システムにおいて、RGB-D カメラのみを用いてカラー画像とデプス画像を取得する方針とした。

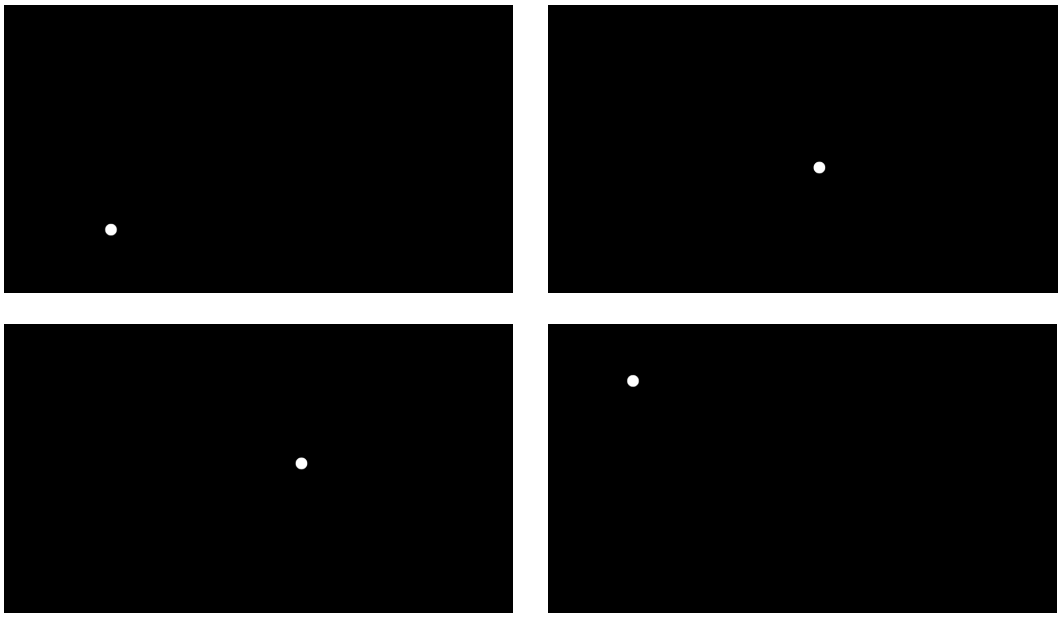
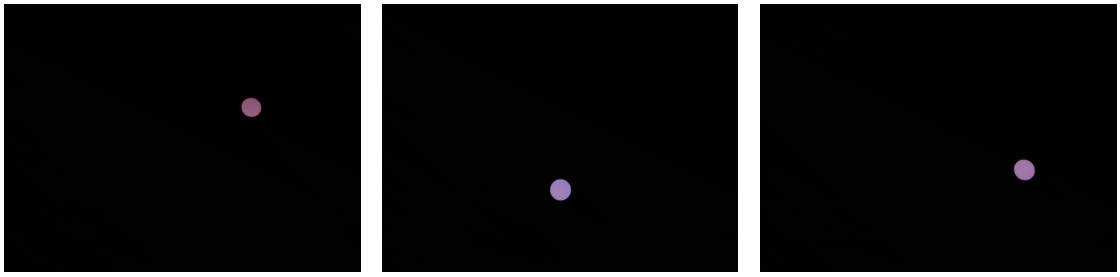
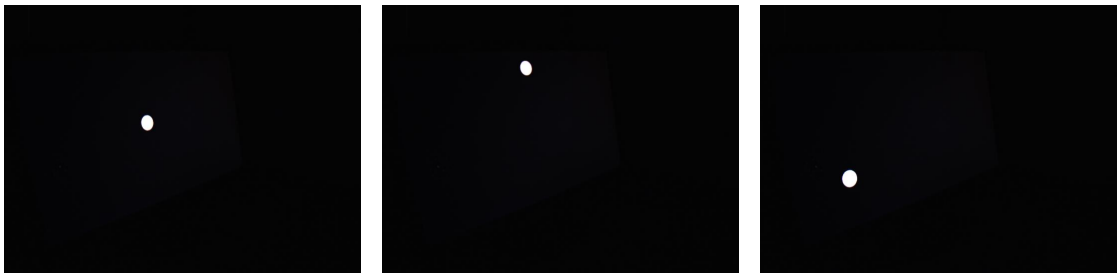


図 4.13: ディスプレイに表示したドットパターンの例



RGB-Dカメラで撮影したドットの画像



RGBカメラで撮影したドットの画像

図 4.14: RGB-D カメラおよび RGB カメラで撮影したドットの画像の例

4.2.1.2　MVE を用いたキャリブレーション方法

Multical を用いたキャリブレーションではカメラの内部パラメータと外部パラメータを同時に推定したが、パラメータを推定する際、これらのパラメータが相互に誤差を吸収し合うことで最終的な精度が低下する可能性がある。そこで、MVE を用いたキャリブレーションでは、内部パラメータにはカメラに書き込まれた工場出荷値を使用し、その値に基づき、MVE を用いて外部パラメータのみを推定した。

MVE は複数の視点からの画像を用いて 3 次元モデルを再構成するソフトウェアであり、Structure-from-Motion（以下、SfM）によって複数の画像間で特徴点を認識してカメラのパラメータを推定したのちに、2.1 節で述べた MVS によって 3 次元モデルを再構成する。MVE を用いたキャリブレーションでは、MVE 内で実行される SfM によって外部パラメータを取得した。その際、1 回の試行では十分な精度が得られなかったため、その後独自の手法により精度の向上を図った。

本処理では、最初に、図 4.15 に示すような、ポスターを使った特徴の多い静的な環境を作成し、その環境を対象として、カラー画像とデプス画像のペアを複数撮影した。その際、撮影されたデプス画像の中に、精度の低いデプス画像が存在した。そこで、事前に On-Chip Calibration[38] による自動キャリブレーションを実施してカメラのパラメータを補正し、デプス画像の精度を向上させた。SfM では複数の画像間で特徴点を

図 4.15: MVE を用いたキャリブレーション用の画像の撮影に使用した環境

検出するが、提案システムで使用する 12 台のカメラは図 4.2 および図 4.3 のように設置されており、これらのボックスに設置されたカメラ（以下、固定カメラ）で撮影した画像（以下、固定カメラ画像）のみでは、画像間で共通する特徴点の数が不足することなどが原因で十分なキャリブレーション精度が得られなかった。そこで、固定カメラ画像に加えて、移動可能な追加のカメラ（以下、手持ちカメラ）を 1 台使用して、環境を移動しながら連続して撮影した画像（以下、連続画像）を複数用意し、固定カメラ画像と連続画像の両方の画像を用いて SfM を実行した。その際、本キャリブレーション環境では、図 4.15 に示したようにボックス側面が覆われており、ボックス内部へはボックス上面のみからアクセスが可能であった。そこで、手持ちカメラを図 4.4 のように棒の先に取り付け、図 4.16 に示すようにボックス内の環境を撮影した。手持ちカメラの動きを制限しないよう手持ちカメラは無線で接続し、手ぶれの影響を抑えるため、手持ちカメラのフレームレートは 60fps に設定した。カメラのキャリブレーション処理はオフラインで行われるため、キャリブレーション用の画像をリアルタイムに取得する必要はない。そこで、撮影した画像を一旦、図 4.4 および表 4.2 に示す RasberryPi のメモリに格納し、撮影後にサーバ用 PC に転送して連続画像として利用した。図 4.17 に手持ちカメラで撮影した連続画像の例を示す。SfM ではカラー画像のみを元情報としてカメラ姿勢を推定するため、得られるカメラ間の姿勢は相対的なものであり、スケール情報は含まれない。そこで、MVE で使用したすべてのカラー画像とペアになるデプス画像を使用してスケール値を推定し、SfM で得られた結果と合わせて使用した。スケール値の推定では、最初に、各カメラのデプス画像から得られる値を使用して、各カメラ座標系基準の被写体上の点の 3 次元座標を計算し、各カメラ画像に投影した。その後、ペアとなるカラー画像のカラー値と投影先のカラー画像のカラー値から再投影誤差（色差）を計算し、再投影誤差が最も小さくなるようにスケール値を最適化した。なお、MVE では、使用する画像を同じカメラで撮影した（内部パラメータを共通化した上で最適化）か、異なるカメラで撮影した（カメラ毎に異なる内部パラメータを持つとして最適化）かを個別に指定することが困難であったため、SfM およびスケール値の推定に使用するカメラの内部パラメータには、手持ちカメラの内部パラメータ（カメラに書き込まれた工場出荷値）を使用した。手持ちカメラで撮影した連続画像の枚数はカラー画像とデプス画像それぞれ 186 枚、固定カメラで撮影した固定カメラ画像の枚数はカラー画像とデプス画像それぞれ 12 枚であった。

先述のように、MVE を用いたキャリブレーションでは、1 回の試行のみでは十分な精度が得られなかった。SfM では、最初に 2 枚の画像を初期画像ペアとして定め、この

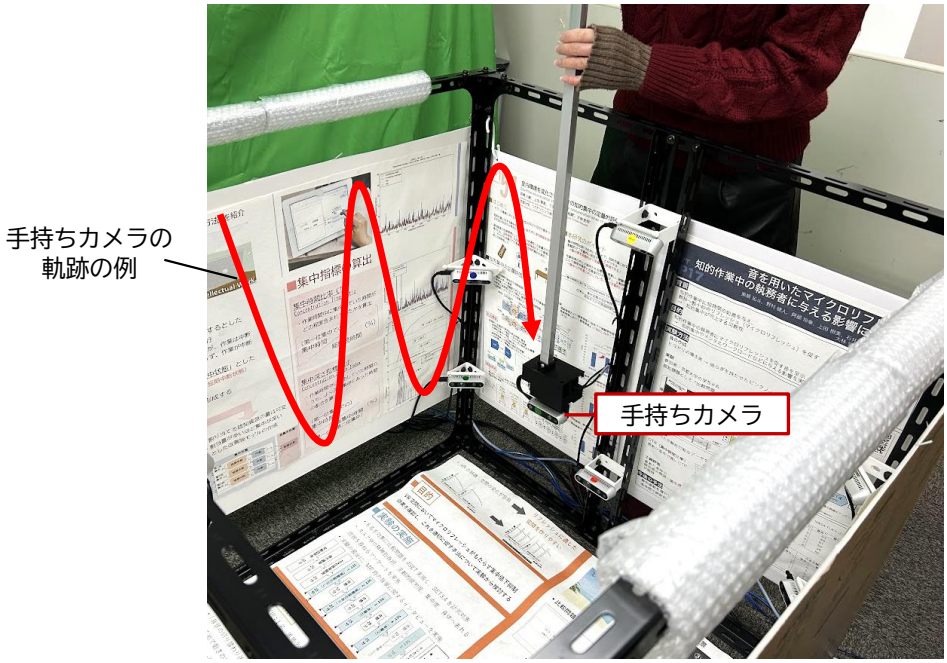
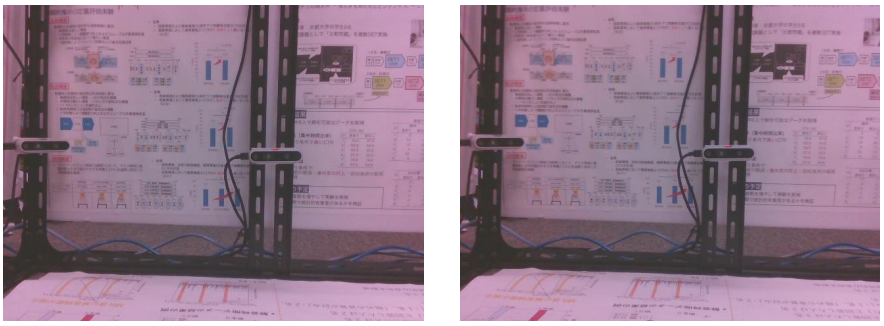
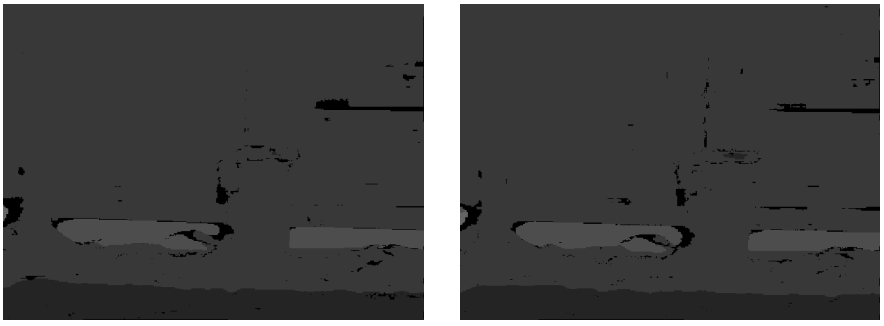


図 4.16: MVE を用いたキャリブレーションにおける連続画像の撮影方法



カラー画像



デプス画像

図 4.17: 手持ちカメラで撮影した連続画像の例

初期画像ペアに対して特徴点マッチングを行った結果を基に後続の処理を進める。初期画像ペアとして選ばれる画像は試行ごとに異なるため、試行によって得られるキャリブレーション結果の値にばらつきが生じていた。この結果のばらつきを確認したところ、一部の試行では得られた結果の値に外れ値が含まれていた。そのため、追加の処理として、使用する画像の中から 2 枚の画像を選択するすべての組み合わせに対して、それらを初期画像ペアとして SfM およびスケール値の推定を実行し、それらすべての結果の収束値を求めることでスケール値を最適化した。

これらの結果を用いて自由視点画像を生成したところ、追加の処理による精度の向上が確認され、Multical を用いた結果と比較しても精度が大幅に向上したが、生成画像にはわずかな色のずれが残った。そこで、スケール値を手作業で微修正したところ、生成画像の色のずれが改善された。MVE を用いたキャリブレーション結果は提案システムで使用するのに十分な精度であったが、追加で実施するスケール値の最適化処理には数日を要し、さらに得られた結果を手作業で微修正する必要があるため、時間的コストには課題が残った。

4.2.1.3　ORB SLAM3 を用いたキャリブレーション方法

ORB SLAM3 は 1 台のカメラで連続して撮影されたカラー画像とデプス画像のペアを順番に処理して画像間の特徴点マッチングを行い、各画像を取得した際のカメラの外部パラメータをスケールも含めて推定する SLAM ソフトウェアである。提案システムでは複数のカメラを使用するため、厳密にはカメラごとに内部パラメータが異なるが、内部パラメータが同じと仮定した上で最適化対象から除外し、スケールも含めて外部パラメータのみを推定した方が処理が安定し、最終的に得られる結果の精度が向上する可能性があると考えた。そこで、ORB SLAM3 を用いたキャリブレーションでは、内部パラメータにはカメラに書き込まれた工場出荷値を使用し、その値に基づいて外部パラメータを推定した。

本処理では、MVE を用いたキャリブレーションと同様に、図 4.15 に示したようなポスターを使った特徴の多い静的な環境を作成し、その環境を対象として、カラー画像とデプス画像のペアを複数撮影した。また、撮影に先立ち、On-Chip Calibration による自動キャリブレーションを実施した。先述のように、ORB SLAM3 では 1 台のカメラで連続して撮影された画像のペアを使用することが想定されている。しかし、提案システムで使用する 12 台のカメラは図 4.2 および図 4.3 に示したように各々離れた場所に設置されているため、固定カメラで撮影した画像を連続した画像とみなすことは難

しく、固定カメラ画像のみでは十分なキャリブレーション精度が得られなかった。そこで、MVE を用いたキャリブレーションと同様に、手持ちカメラを図 4.4 のように棒の先に取り付けて連続画像を複数撮影し、処理に使用した。その際、固定カメラ画像と連続画像を合わせた画像列を連続した画像列として処理できる必要があるため、図 4.18 に示すように、手持ちカメラを各固定カメラに近づけながらボックス内の環境を連続して滑らかに撮影した。これにより、図 4.19 に示すように、連続画像内に各固定カメラ画像に類似した画像が含まれることとなり、固定カメラ画像に類似した画像同士の間に各固定カメラ画像を挿入することで、固定カメラ画像と連続画像を合わせた画像列を新たな連続した画像列とみなすことができる。この画像列を ORB SLAM3 の処理に使用し、すべての固定カメラの外部パラメータを取得した。手持ちカメラで撮影した連続画像の枚数はカラー画像とデプス画像それぞれ 3881 枚、固定カメラで撮影した固定カメラ画像の枚数はカラー画像とデプス画像それぞれ 12 枚であった。

連続画像の中に固定カメラ画像を挿入する際は、ORB アルゴリズムを用いて、各固定カメラのカラー画像と全ての連続画像との間で特徴点マッチングを行った。そして、図 4.20 に示すように、共通する特徴点の数が最も多い画像を各固定カメラ画像に最も類似した画像とみなし、その画像の後ろに各固定カメラ画像を挿入した。

ORB SLAM3 を用いたキャリブレーション結果を用いて自由視点画像を生成したところ、MVE の結果と同様に、提案システムで使用するのに十分な精度であることが確認された。先述のように、MVE を用いたキャリブレーションは時間的コストに課題が残るため、本研究では、最終的に ORB SLAM3 を用いたキャリブレーション結果を使用した。得られたキャリブレーション結果は json ファイルとして保存し、体験アプリの起動時にファイルを読み込んでカメラパラメータとして使用した。

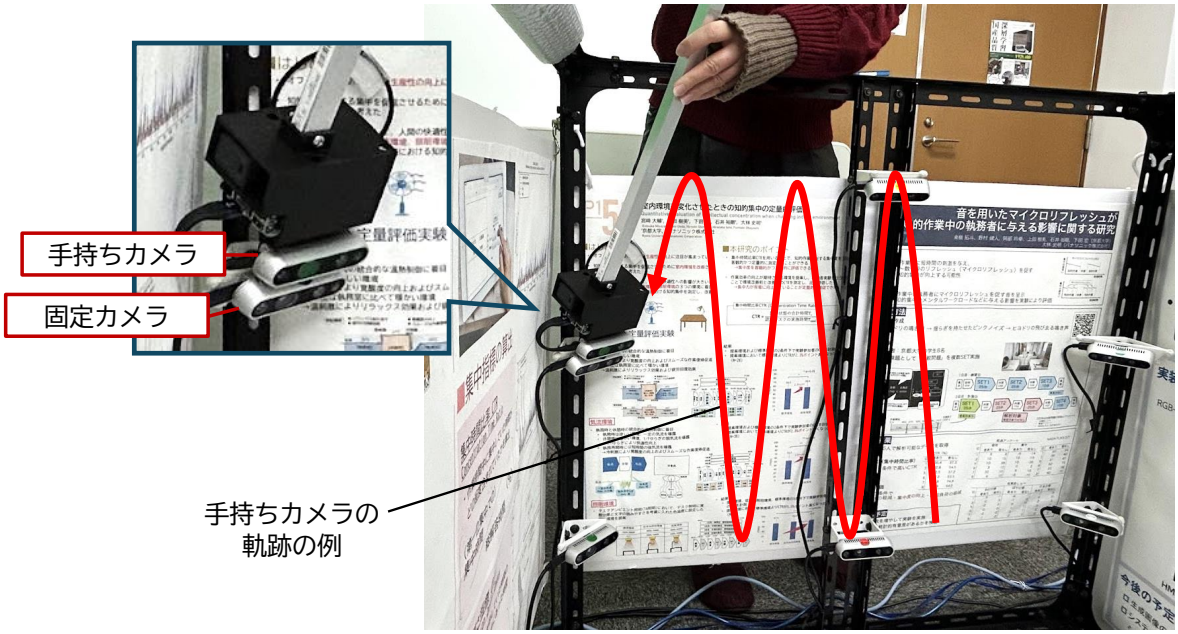


図 4.18: ORB SLAM3 を用いたキャリブレーションにおける連続画像の撮影方法

図 4.19: 各固定カメラ画像に最も類似した連続画像の例

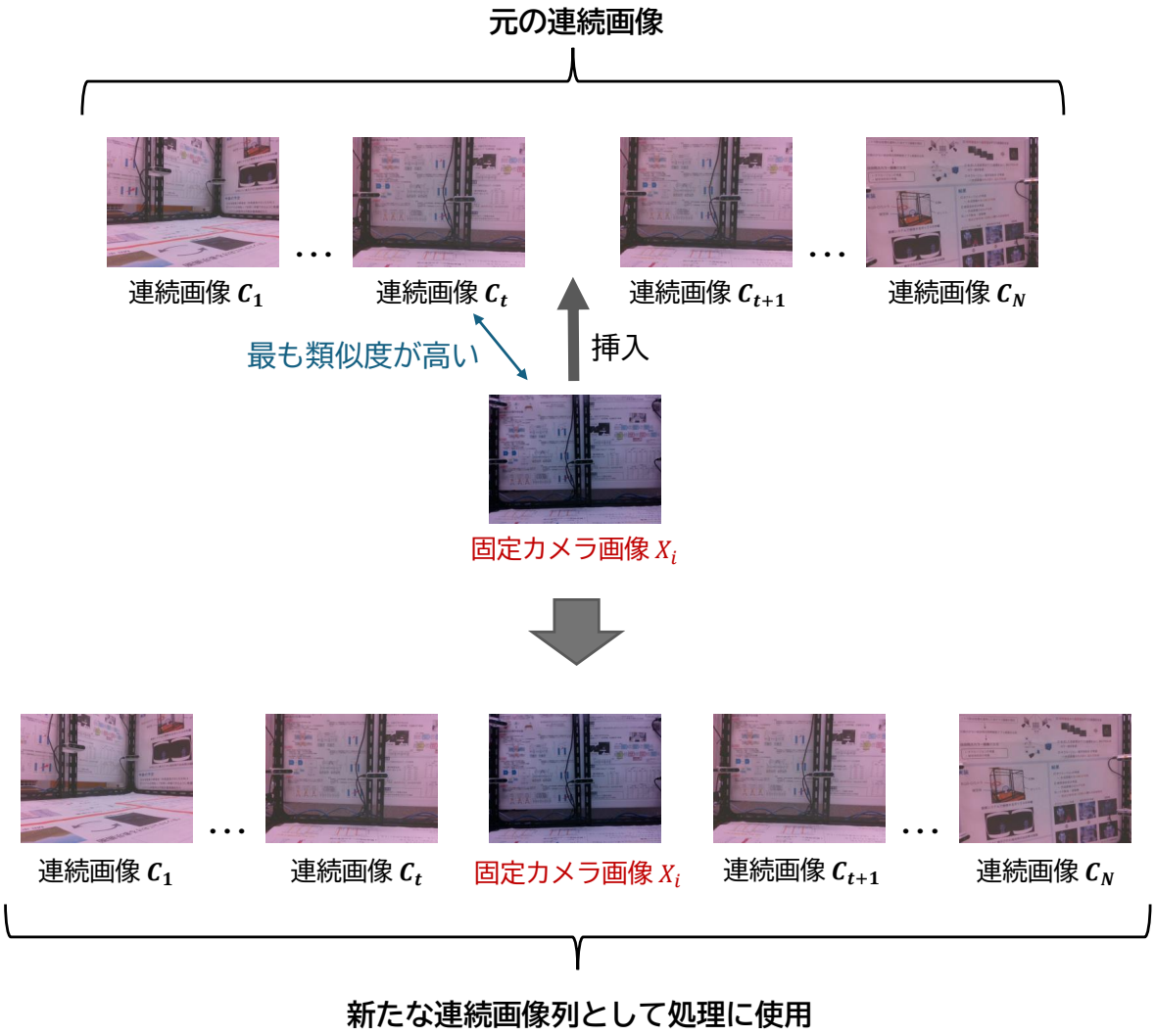


図 4.20: 連続画像の中に固定カメラ画像を挿入する処理の例（カラー画像）

4.2.2 カラー画像とデプス画像の撮影および送信のためのアプリケーション

キャプチャアプリはMicrosoft 社の VisualStudio2019 を用いて開発し、開発言語にはC++を用いた。本アプリでは、6 台の RealSense を用いてカラー画像とデプス画像を撮影し、撮影した画像を TCP 通信を用いてサーバ用 PC に送信する。その際、カメラごとに独立したスレッドを作成し、それぞれのスレッドで以下の処理を並列に実行した。

1. TCP 接続の確立: 指定した IP アドレスとポート番号に TCP 接続を確立
2. 画像の取得: RealSense からカラー画像とデプス画像を取得
3. 画像の送信: 取得した画像を TCP 通信でサーバ用 PC に送信

ここで、IP アドレスにはサーバ用 PC の IP アドレスを指定し、ポート番号は連続した 6 つのポート番号を 6 台のカメラそれぞれに 1 つずつ割り当てた。カラー画像とデプス画像の取得には Intel RealSense SDK 2.0[39] を、通信には boost(Ver 1.79.0)[40] に含まれている asio を用いた。

4.2.3 両目用の自由視点画像の生成と HMD 上への提示のためのアプリケーション

体験アプリはキャプチャアプリと同様に Microsoft 社の VisualStudio2019 を用いて開発し、開発言語にはC++を用いた。本アプリでは、最初に 2 台のクライアント用 PC から、それぞれ 6 台の RealSense で撮影したカラー画像とデプス画像を受信する。その際、カメラごとに独立したスレッドを作成し、それぞれのスレッドで以下の処理を並列に実行した。

1. TCP 接続の確立: 指定したポート番号に TCP 接続を確立
2. 画像の取得: クライアント用 PC から送信されたカラー画像とデプス画像を取得

ここで、ポート番号にはキャプチャアプリで指定したポート番号と同じポート番号を指定した。

その後、受信したカラー画像とデプス画像を用いて、体験者の頭の動きに応じた両目用の自由視点画像を生成する。体験者の頭の動きは体験者が装着した HMD の姿勢を読み込むことで取得し、この姿勢をシステムで生成する自由視点画像の姿勢とした。

HMD の姿勢の読み込みには、Unity techinologies 社が提供する統合開発環境の Unity（Ver.2022.3.16f1）[41] および Unity Asset Store から入手可能な Meta Quest デバイス向けの VR/AR アプリ開発キット Meta XR Interaction SDK（Ver.60.0.1）[42] を使用した。読み込んだ HMD の姿勢は、Unity で使用する左手座標系からシステムで使用する右手座標系に変換したのち、UDP 通信でサーバ用 PC に送信し、キャプチャアプリ内で読み込んで使用した。加えて、3.1 節で述べたように、HMD を装着する際や体験中に、HMD の姿勢が体験者の意図しない姿勢になってしまう場合があるため、視点を自然な姿勢にリセットすることができるように、体験者が図 4.6 に示したコントローラのボタンを押した際に HMD の姿勢を初期化する設定とした。

自由視点画像の生成では、デプス画像のノイズ除去および自由視点デプス画像の生成に GPU（CUDA）を用いた並列処理を、自由視点カラー画像の生成に OpenMP を用いた並列処理を適用することで、これらの処理をピクセル毎に並列化可能なアルゴリズムとして実装した。本アルゴリズムを実装する際、3.5 節や 3.6.2 項および 3.6.3 項で述べたように、いくつかの閾値を設定する必要がある。本研究では、デプス画像のノイズ除去と自由視点デプス画像の生成の際の符号付距離の絶対値の上限の閾値を 40mm、オクルージョンへの対応の際の閾値 d_{th} を 5mm にそれぞれ設定した。また、被写体表面の法線方向の考慮の際には、被写体表面の単位法線ベクトルの z 座標値である c に対する閾値を-0.4 とした。すなわち、2 つのベクトルのなす角の大きさにの閾値は約 114° となる。HMD 上への提示のため、生成する自由視点画像の解像度は 516 × 552 ピクセルとした。

最後に、生成した両目用の自由視点画像を HMD 上に提示する。HMD 上への画像提示には Unity（Ver.2022.3.16f1）および Meta XR Interaction SDK（Ver.60.0.1）を使用し、両目用の自由視点画像を HMD の左右の目に提示した。HMD とサーバ用 PC との接続には Meta Quest Link および Air Link を使用し、無線で接続した。HMD と PC を有線ではなく無線で接続した理由は、有線での接続の場合、システムの体験中に HMD につながれた接続ケーブルにより体験者の動きが制限される可能性があるためである。

生成した両目用の自由視点画像を HMD 上に提示した例を図 4.21 に示す。本システムでは、自由視点画像の各ピクセルのデプス値と自由視点および各カメラの姿勢が取得できているため、3.6.1 項で述べたように、自由視点画像に写る被写体のうち没入対象となる小空間内に存在する被写体に対してのみカラー情報を取得する処理が可能である。この背景処理を適用した場合、図 4.22 のように、小空間内に存在する被写体以外についてはカラー情報が取得されず、背景部分が黒色に表示される。これにより、小空間

外の物が体験者の視界に映り込むことで没入感が損なわれることを防ぐことができる。システム全体での処理速度は両目用の自由視点画像 1 フレームあたり約 90〜100ms、すなわち約 10〜11fps であった。

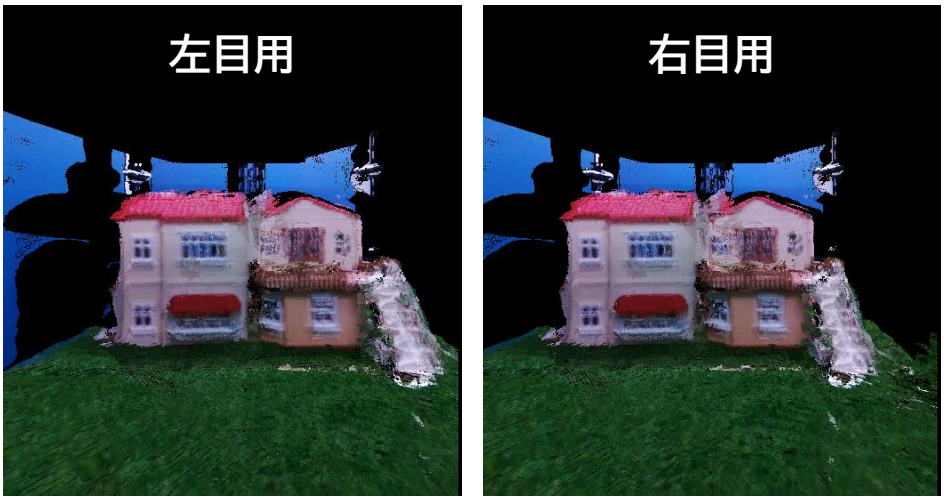


図 4.21: HMD 上に提示された両目用の自由視点画像の例（背景処理前）

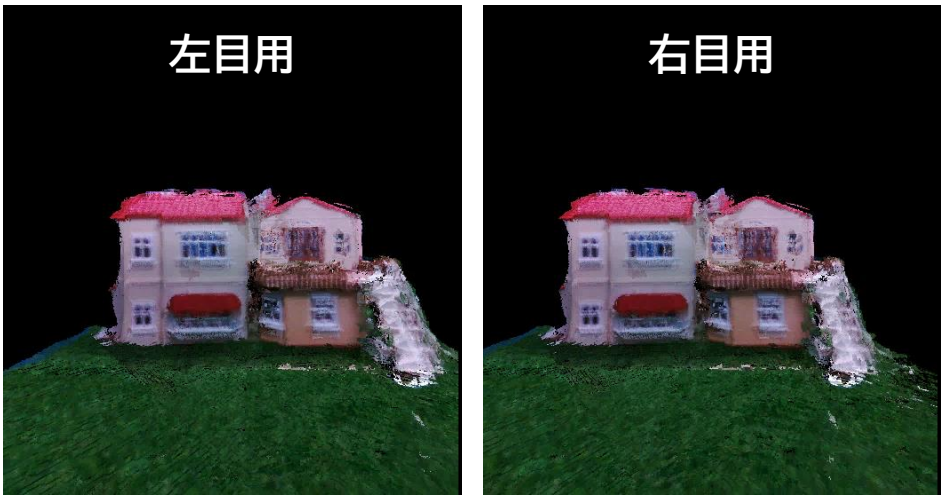


図 4.22: HMD 上に提示された両目用の自由視点画像の例（背景処理後）

第 5 章 高精細立体映像提示システムの評価

本研究では、3.5 節で述べた手法を用いて、ランダムノイズを除去した高解像度な自由視点デプス画像の生成を試みた。また、3.6.2 項および 3.6.3 項で述べた手法を用いて、自由視点カラー画像を生成する際に、各カラー画像から適切なカラー情報を取得することで、生成画像の色の誤りや歪みの改善を試みた。本章では、これらの画質改善手法の有効性の評価として、3.5 節で述べたデプス画像のノイズ除去と超解像、3.6.2 項で述べたオクルージョンへの対応、および 3.6.3 項で述べた被写体表面の法線方向の考慮の各処理に対して、それぞれの効果を評価する方法を述べたのち、その結果と考察を述べる。その後、システムの体験例について述べる。

5.1 デプス画像のノイズ除去と超解像の有効性の確認

本評価では、自由視点が被写体に近接した場合の生成画像の精度に着目し、デプス画像のノイズ除去と超解像の有効性を確認する。

5.1.1 評価方法

デプス画像のノイズ除去と超解像の有効性の評価では、動的な環境を対象としたリアルタイム処理が可能な既存手法として、RGB-D カメラから取得したカラー画像とデプス画像を直接利用して作成した色付き点群をもとに生成した自由視点画像（以下、点群画像）と、提案システムで生成した自由視点画像を比較する。その際、被写体の特定の個所に近接した視点を複数設定し、それぞれの視点における生成画像を比較することで、自由視点が被写体に近接した際にも高解像度な画像を生成可能であることを確認する。

点群画像の生成では、はじめに、各 RGB-D カメラを用いてカラー画像とデプス画像を取得する。その後、取得したデプス画像の各ピクセルが格納しているデプス値から、式 (5.1) によりそのピクセルが映す被写体上の点の各カメラ座標系基準の 3 次元座

標 $\boldsymbol{p}_{ci} = (p_x^{ci}, p_y^{ci}, p_z^{ci})$ を求める。

$$\boldsymbol{p}_{ci} = \begin{pmatrix} p_x^{ci} \\ p_y^{ci} \\ p_z^{ci} \end{pmatrix} = \begin{pmatrix} \frac{(x-c_x)z_{ci}}{f_x} \\ \frac{(y-c_y)z_{ci}}{f_y} \\ z_{ci} \end{pmatrix} \tag{5.1}$$

ここで、x, y は処理対象ピクセルの座標、f_x、f_y はカメラの焦点距離（単位はピクセル）、c_x、c_y は主点（単位はピクセル）、z_{ci} は処理対象ピクセルが格納しているデプス値（単位は mm）である。

次に、3.4 節で求めたカメラの姿勢を用いて、式 (5.2) および (5.3) により各カメラ座標系基準の被写体上の点の 3 次元座標 $\boldsymbol{p}_{ci}$ を世界座標系基準の 3 次元座標 $\boldsymbol{p}_w$ に変換する。

$$\boldsymbol{p}_w = \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \boldsymbol{T}_{ciw} \times \begin{pmatrix} p_x^{ci} \\ p_y^{ci} \\ p_z^{ci} \\ 1 \end{pmatrix} \tag{5.2}$$

ここで、T_{ciw} はカメラ i の座標系の世界座標系に対する同次変換行列である。

最後に、カラー画像のデプス画像と同じ座標のピクセルから色情報を取得し、小空間内の環境の色付き点群を生成する。そして、得られた点群を、Point Cloud Library[43]（以下、PCL）に付属する PCLVisualizer を用いて点群画像として表示する。

本処理には表 4.1 に示した 12 台の RGB-D カメラおよび表 4.7 に示した PC を使用した。点群画像をレンダリングする際には提案システムにおける自由視点と同じ外部パラメータおよび内部パラメータを指定することで、提案システムと同じ姿勢かつ解像度の点群画像を生成した。点群画像の背景色は、提案システムで生成した自由視点画像の背景色と同じ黒色に設定した。描画する点のサイズについて、サイズを小さくすることで被写体の細部に対する再現性は向上するが、得られる点群の密度は低くなる。一方、点のサイズを大きくすることで得られる点群の密度は高くなるが、取得したデプス画像のノイズの影響も大きくなり、生成画像の精度の低下につながる。本処理では、描画する点のサイズが 2 ピクセルの場合、3 ピクセルの場合、および 5 ピクセルの場合のそれぞれについて点群画像を生成し、提案システムで生成した自由視点画像との比較を行った。実際に使用した環境を図 5.1 に示す。

図 5.1: デプス画像のノイズ除去と超解像の有効性を確認するために使用した環境

5.1.2 結果と考察

点群を用いて生成した自由視点画像と提案システムで生成した自由視点画像との比較を図 5.2 および図 5.3 に示す。本研究では、没入対象となる空間を 90cm 四方の小空間に設定しているため、システム体験時に体験者の視点が被写体の特定の個所に近づいた場合に、体験者の視点と被写体表面との距離が数十 cm 以下になることが想定される。本システムを用いて没入体験を可能にするためには、このように体験者の視点と被写体表面との距離が近距離となる場合でも被写体の細部が視認可能である必要がある。点群画像では、描画する点のサイズが 2 ピクセルおよび 3 ピクセルの場合、RGB-D カメラ単体でのデプス画像の解像度が不十分であることが原因で、自由視点における視野内に入る点群の密度が被写体と視点間の距離に対して十分ではないために、自由視点が被写体に近づくほど被写体の視認性が低下している。さらに、図 5.3 の右端の画像が示すように、点群が疎であることで点同士の隙間から背景が見えており、不自然な生成結果となっている。また、描画する点のサイズが 5 ピクセルの場合には、自由視点の視野内の点群の密度が高くなることで上述の問題に関してはある程度の改善が見られる一方で、点のサイズを大きくすることで RGB-D カメラで取得したデプス画像に含まれるノイズの影響も大きくなり、部分的に生成画像の精度が低下していること

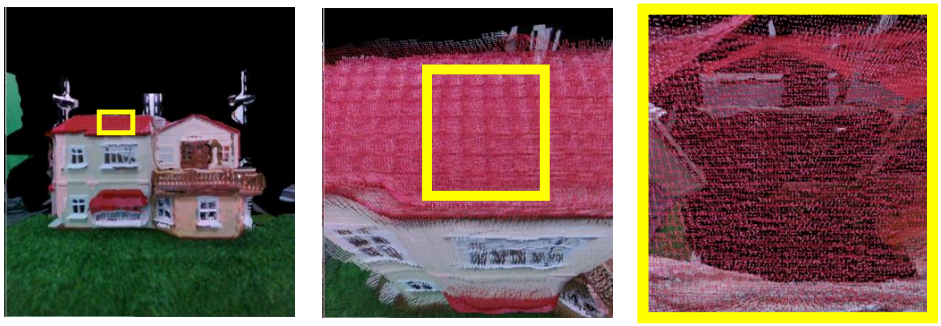
点群画像（点のサイズ：２ピクセル）

点群画像（点のサイズ：３ピクセル）

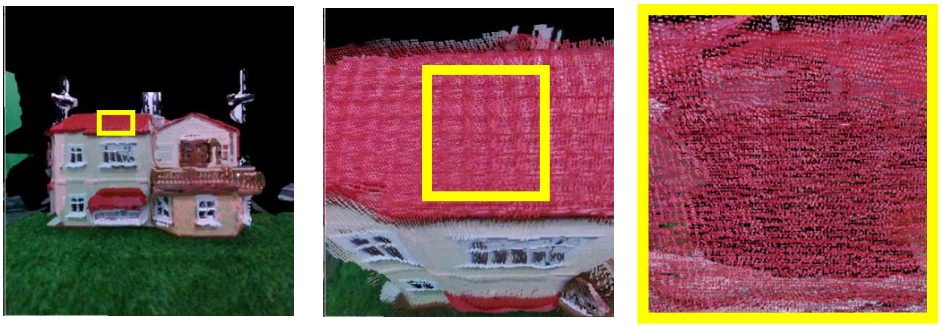
点群画像（点のサイズ：５ピクセル）

提案システムで生成した自由視点画像

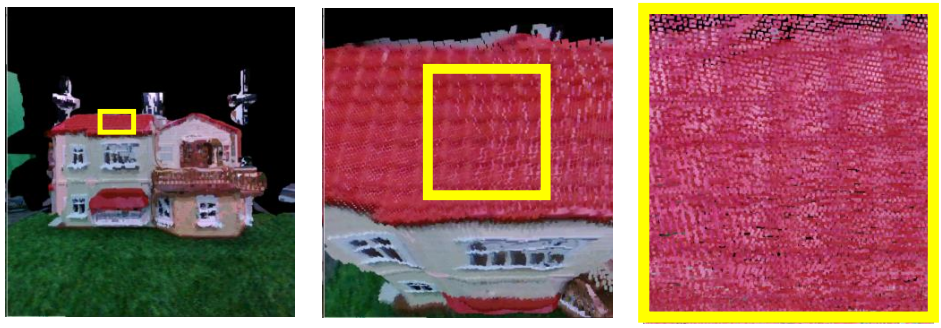
図 5.2: 点群を用いて生成した自由視点画像と提案システムで生成した自由視点画像との比較



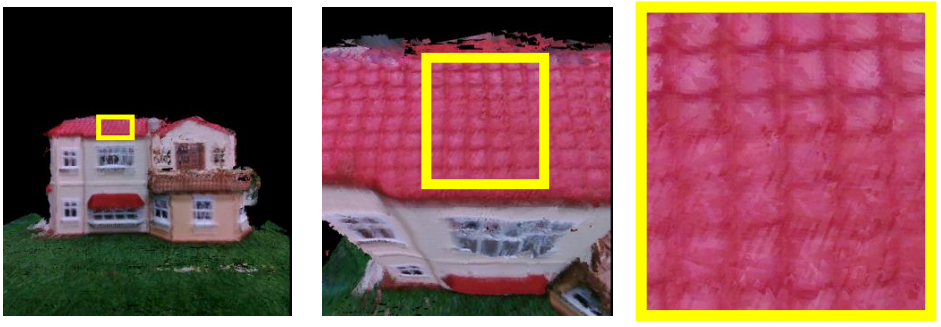
点群画像（点のサイズ：２ピクセル）



点群画像（点のサイズ：３ピクセル）



点群画像（点のサイズ：５ピクセル）



提案システムで生成した自由視点画像

図 5.3: 点群を用いて生成した自由視点画像と提案システムで生成した自由視点画像との比較（屋根に視点を近づけた場合）

が分かる。

これに対して、提案システムで生成した自由視点画像では、デプス画像のノイズ除去により RGB-D カメラ単体でデプス画像を取得した場合よりも高解像度かつ高密度なデプス画像を生成することができるため、図 5.2 および図 5.3 に示すように、点群画像と比較して、窓枠の輪郭などの精度が高いことが分かる。加えて、デプス画像の超解像の効果により、図 5.2 および図 5.3 に示すように、被写体に近づいた際にも欠損が生じることなく、点群画像に比べて被写体の詳細が確認できる。特に、被写体である家の模型に近い視点からの生成画像に注目すると、点群画像と比較して、家の窓枠や屋根の瓦の部分などの細部も詳細に確認することができる。

このように、提案システムで生成した自由視点画像では、点群画像と比較して高精度な自由視点画像を生成可能であり、また、自由視点が被写体に近づいた際にも優れた視認性を有しており、デプス画像のノイズ除去と超解像の有効性が確認できる。

5.2 オクルージョンへの対応の有効性の確認

本評価では、オクルージョン発生時の生成画像の精度に着目し、オクルージョンへの対応の有効性を確認する。

5.2.1 評価方法

オクルージョンへの対応の有効性の確認では、図 3.10 に示したようなオクルージョンが発生する環境を対象に撮影する。はじめに、オクルージョンが発生するカメラに関して、オクルージョン領域のカラー情報を適切に排除できているかを確認する。その際、RGB-D カメラ 12 台で取得したデプス情報を基に、オクルージョンが発生している 1 台のカメラのカラー画像のみを用いて画像を生成し、オクルージョンを考慮しない場合と考慮する場合について、複数の視点から見た結果を比較する。次に、オクルージョンが発生していないカメラを使用することで生成画像の精度が向上しているかを確認する。その際、RGB-D カメラ 12 台で取得したデプス情報を基に、オクルージョンが発生しているカメラと発生していないカメラの合計 2 台のカメラのカラー画像から自由視点カラー画像を生成し、オクルージョンを考慮しない場合と考慮する場合について、複数の視点から見た結果を比較する。

実際に使用した環境を図 5.4 に、使用する 2 台のカメラのカラー画像を図 5.5 に示す。本研究では、ボックスの周囲に複数のカメラを配置することで小空間に対する撮影の

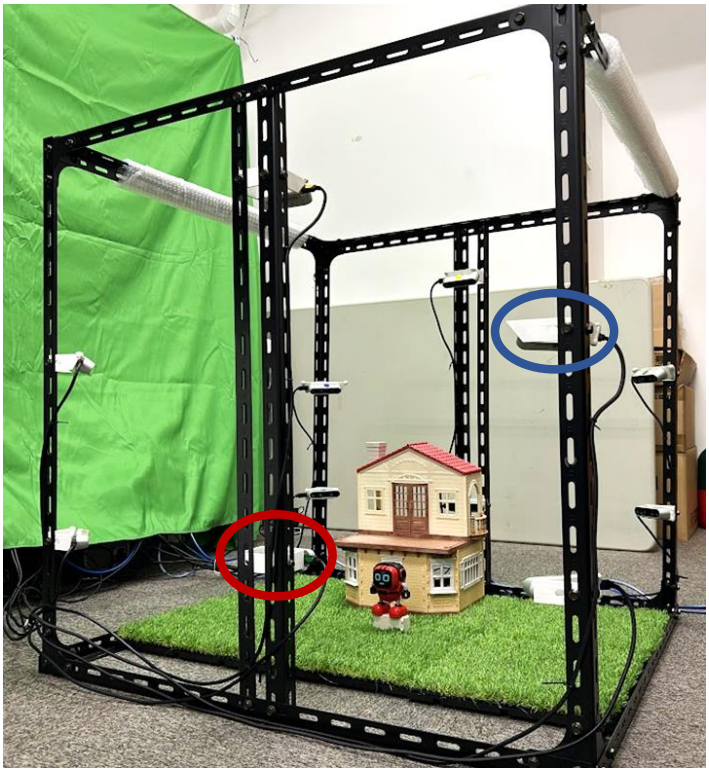


図 5.4: オクルージョンへの対応の有効性を確認するために使用した環境

オクルージョンが発生している
カメラの画像

オクルージョンが発生していない
カメラの画像

図 5.5: オクルージョンへの対応の有効性の確認に使用した各 RGB-D カメラのカラー画像

抜け落ちを減らすことができるが、図 5.4 に示すようなオクルージョンの発生時には、3.6.2 項で述べたように、オクルージョンの発生するカメラのカラー情報を適切に処理しなければ誤った画像が生成されてしまう可能性がある。

5.2.2 結果と考察

オクルージョンへの対応の有無による生成画像の比較を図 5.6 および図 5.7 に示す。はじめに、オクルージョンが発生している 1 台のカメラのみからカラー情報を取得した場合の、複数の視点から見た画像を比較した結果が図 5.6 である。 オクルージョンを考慮しない場合には、図 5.6 の左に示すように、家の模型の色が描画されるべき部分に対して赤色のロボットの色が誤って描画されている。これは、オクルージョンの発生により取得された誤ったカラー情報が使用されたためである。一方で、オクルージョンを考慮した場合には、図 5.6 の右に示すように、赤色のロボットの色が誤って描画されていた領域の色が描画されておらず、オクルージョン領域のカラー情報が適切に排除されていることが分かる。また、RGB-D カメラはカラー画像とデプス画像を同時に取得できるが、原理的に焦点位置を同一にすることはできないため、カラー画像で撮影できていてもデプス画像では撮影できていない領域が、特に被写体の淵近傍で生じることがある（同一カメラ内でのオクルージョンの発生)。そのような場合でも、本手法を用いれば、誤ったカラー情報の使用を避けることができることを示している。

次に、オクルージョンが発生しているカメラと発生していないカメラの合計 2 台のカメラからカラー情報を取得した場合の、複数の視点から見た画像を比較した結果が図 5.7 である。本研究では、3.6.1 項で述べたように、自由視点画像の各ピクセルの色を決定する際、各カメラで取得したカラー値の平均値を使用している。そのため、オクルージョンを考慮しない場合には、オクルージョンが発生しているカメラの誤ったカラー値も含めた平均値が生成画像の各ピクセルの色として使用され、その結果として、図 5.7 の左に示すように、生成した画像の色に誤りが生じてしまう。一方で、オクルージョンを考慮した場合には、図 5.7 の右に示すように、オクルージョンが発生しているカメラのカラー情報が適切に排除され、オクルージョンが発生していないカメラのみから正しくカラー情報を取得するようになり、生成した画像の色が改善していることが分かる。

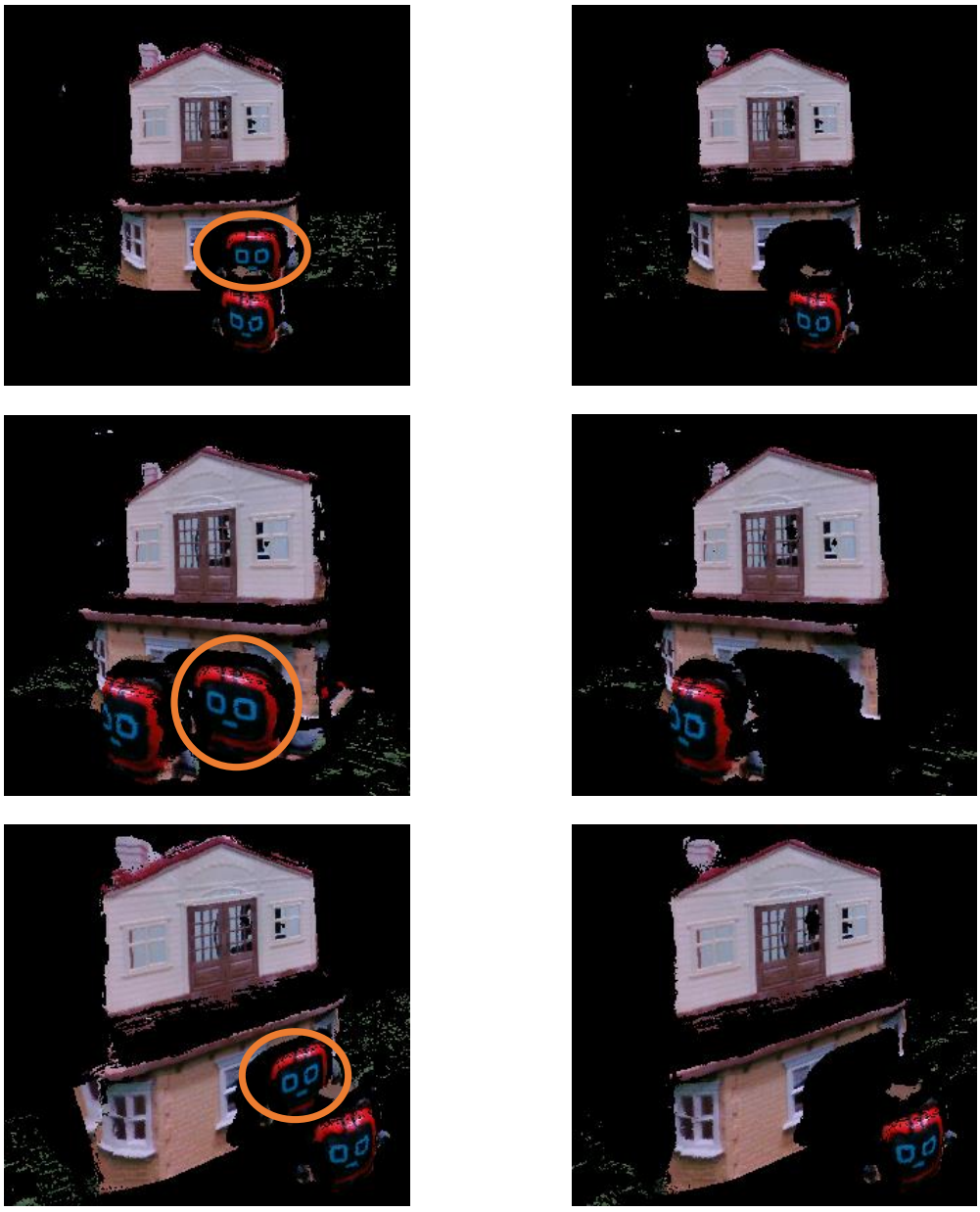


図 5.6: オクルージョンへの対応の有無による生成画像の比較（カメラ 1 台）

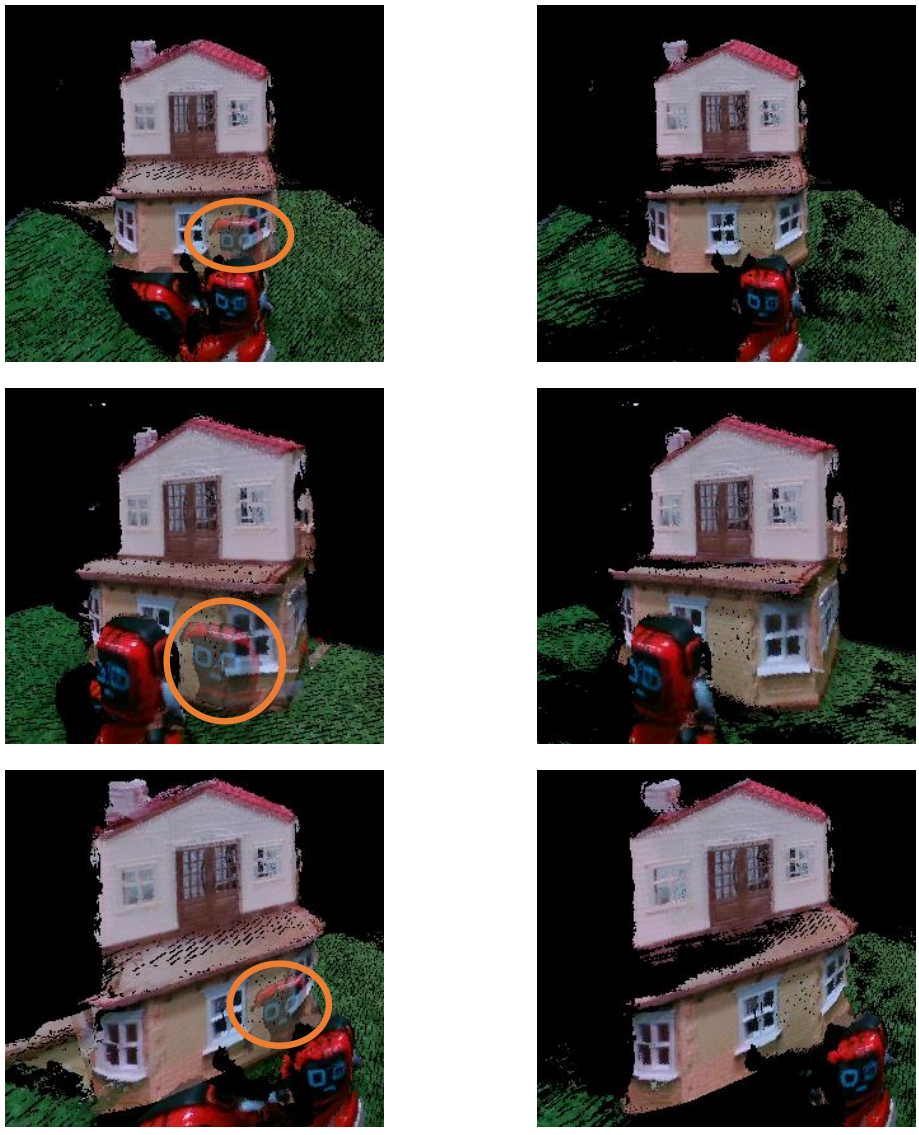


図 5.7: オクルージョンへの対応の有無による生成画像の比較（カメラ 2 台）

5.3　被写体表面の法線方向の考慮の有効性の確認

本評価では、被写体が複雑な表面形状を有する場合の生成画像の精度に着目し、被写体表面の法線方向の考慮の有効性を確認する。

5.3.1　評価方法

被写体表面の法線方向の考慮に対する有効性の評価では、複雑な表面形状を持つ被写体を用いる。はじめに、被写体に正対していないカメラに関して、被写体表面のうちカメラに正対していない領域のカラー情報を適切に排除できているかを確認する。その際、RGB-D カメラ 12 台で取得したデプス情報を基に、被写体に正対していない 1 台のカメラのカラー画像のみを用いて画像を生成し、被写体表面の法線方向を考慮しない場合と考慮する場合について、複数の視点から見た結果を比較する。

次に、被写体に正対するカメラを使用することで生成画像の精度が向上しているかを確認する。その際、RGB-D カメラ 12 台で取得したデプス情報を基に、被写体に正対していないカメラと正対するカメラの合計 2 台のカメラで取得したカラー画像から自由視点カラー画像を生成し、被写体表面の法線方向を考慮しない場合と考慮する場合について、複数の視点から見た結果を比較する。

実際に使用した環境を図 5.8 および図 5.9 に、使用する 2 台のカメラのカラー画像を図 5.10 に示す。本研究では、小空間内の被写体を複数の方向から同時に撮影することで撮影の抜け落ちを減らすことができるが、被写体の位置によっては図 5.8 および図 5.9 のように被写体に正対しないカメラが存在する場合がある。そのような場合、3.6.3 項で述べたように、被写体に正対しないカメラのカラー情報を適切に処理しなければ生成画像の精度が低下してしまう可能性がある。

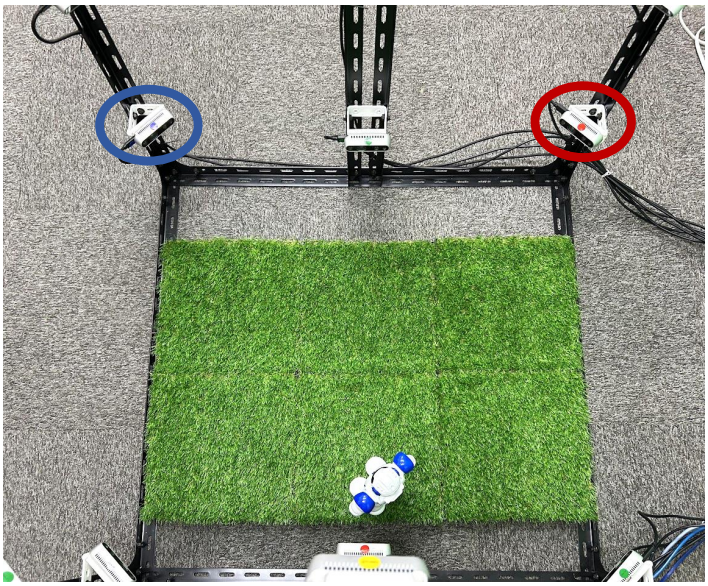
5.3.2　結果と考察

被写体表面の法線方向の考慮の有無による生成画像の比較を図 5.11 および図 5.12 に示す。はじめに、被写体に正対していない 1 台のカメラのみからカラー情報を取得した場合の、複数の視点から見た画像を比較した結果が図 5.11 である。被写体表面の法線方向を考慮しない場合には、図 5.11 の左に示すように、生成した画像の一部に歪みが生じている。これは、被写体に正対していないカメラはキャリブレーションの精度が不十分である影響を受けやすく、被写体に正対していないカメラからの誤りを含んだカラー情報が使用されたためである。一方で、図 5.11 の右に示すように、被写体表

：被写体に正対して**いない**カメラ（１台目）

：被写体に正対**する**カメラ（２台目）

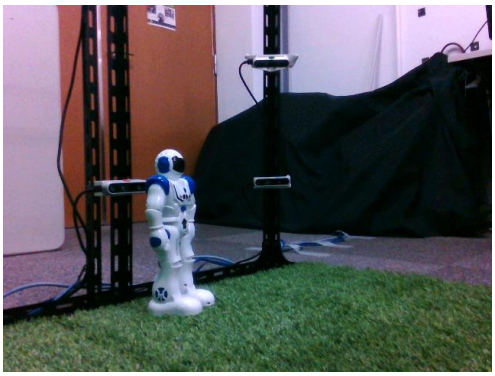
図 5.8: 被写体表面の法線方向の考慮の有効性を確認するために使用した環境



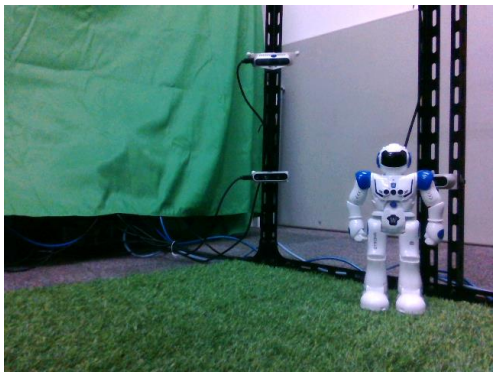
：被写体に正対して**いない**カメラ（１台目）

：被写体に正対**する**カメラ（２台目）

図 5.9: 被写体表面の法線方向の考慮の有効性を確認するために使用した環境（上から見た図）



被写体に正対して**いない**カメラの画像



被写体に正対**する**カメラの画像

図 5.10: 被写体表面の法線方向の考慮の有効性の確認に使用した各 RGB-D カメラのカラー画像

面の法線方向を考慮した場合には、生成画像の歪みが生じていた領域の色が描画されておらず、被写体に正対していないカメラのカラー情報が適切に排除されていることが分かる。

次に、被写体に正対していないカメラと正対するカメラの合計 2 台のカメラからカラー情報を取得した場合の、複数の視点から見た画像を比較した結果が図 5.12 である。

先述のように、本研究では、自由視点画像の各ピクセルの色を決定する際、各カメラで取得したカラー値の平均値を使用している。そのため、被写体表面の法線方向を考慮しない場合には、被写体に正対していないカメラの誤ったカラー値も含めた平均値が生成画像の各ピクセルの色として使用され、その結果として、図 5.12 の左に示すように、生成した画像にずれが生じてしまう。一方で、被写体表面の法線方向を考慮した場合には、図 5.12 の右に示すように、被写体に正対していないカメラのカラー情報が適切に排除され、被写体に正対するカメラのみから正しくカラー情報を取得するようになり、生成した画像のずれが改善していることが分かる。

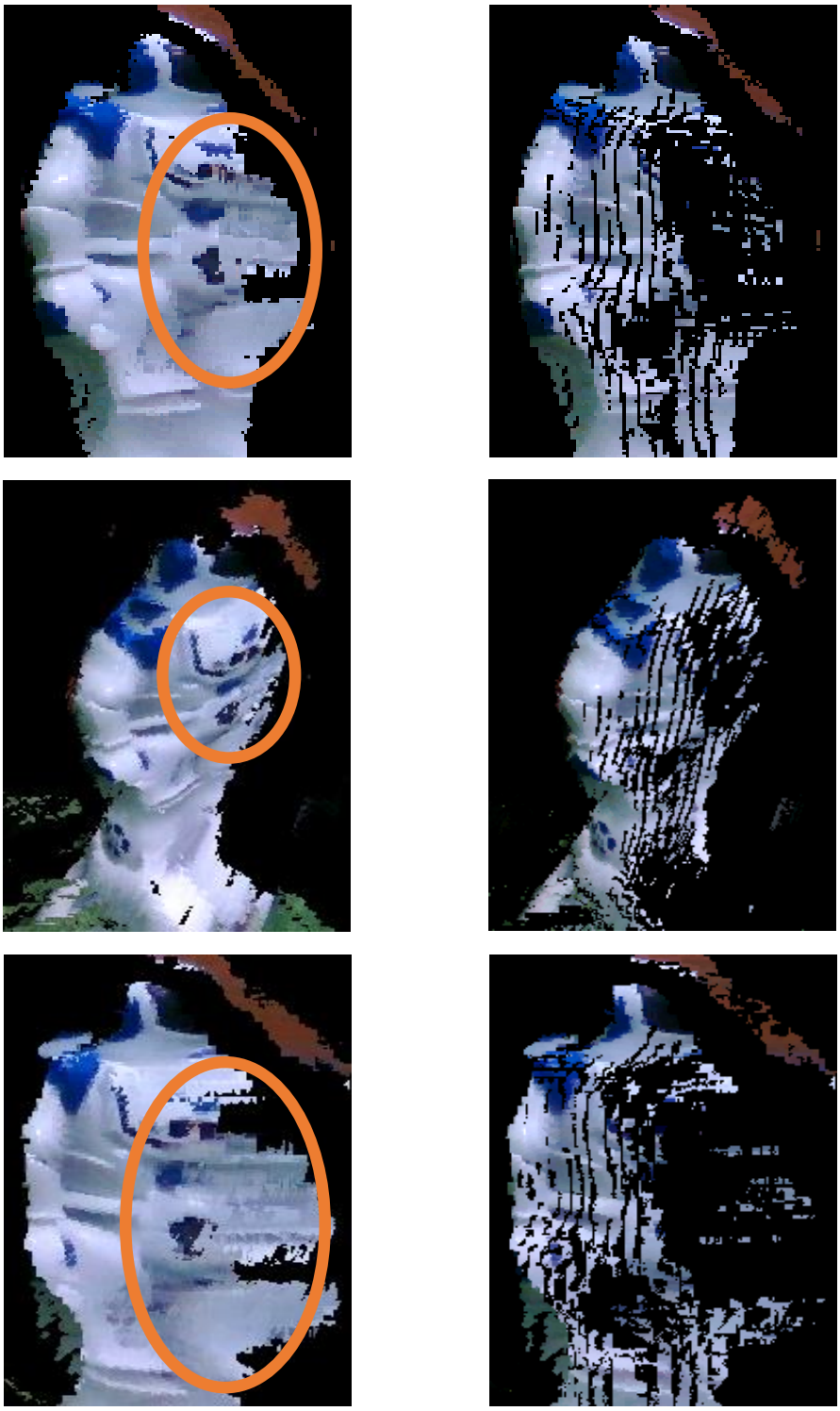


図 5.11: 被写体表面の法線方向の考慮の有無による生成画像の比較（カメラ 1 台）

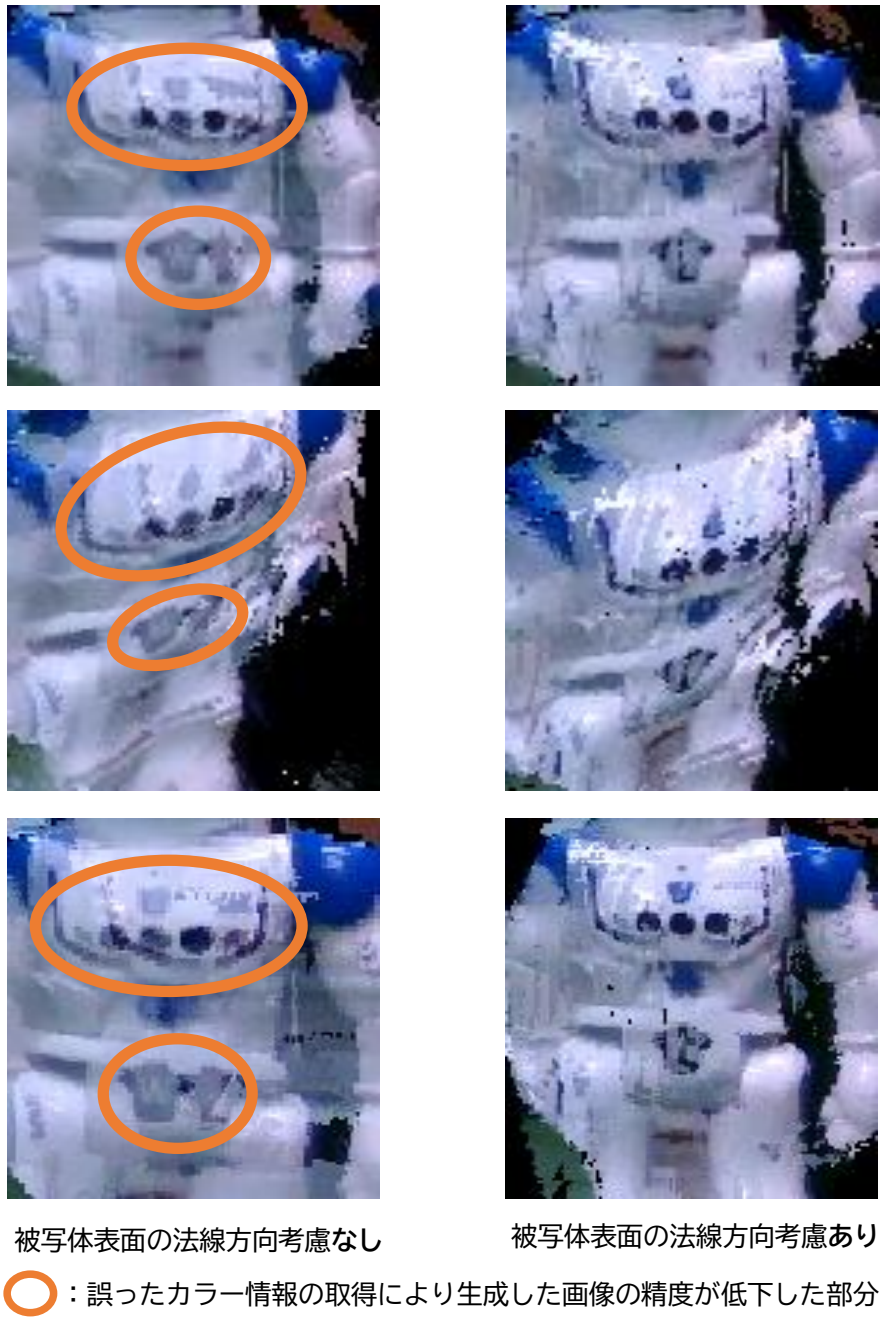


図 5.12: 被写体表面の法線方向の考慮の有無による生成画像の比較（カメラ 2 台）

5.4　システムの体験例

図 5.13 にシステム体験時の様子と、体験者が HMD で見る映像（右目用）の例を示す。体験者は、4.1 節で述べた、自由視点画像のみが見えるモードと外界を見ることができるモードを切り替え可能な HMD を装着し、被写体操作時には外界を見ることができるモードを、小空間への没入時には自由視点画像のみが見えるモードを使用した。被写体操作時は、図 5.13 に示すように、カメラの取り付けられていないボックス側面から内部へアクセスした。

体験の際、図 5.14 に示すように、HMD を被った体験者が頭を動かすと、その動きに連動した両目用の自由視点画像がリアルタイムに生成され、立体映像として HMD 上に提示された。その際、視点を被写体の特定の個所に近づけることで、被写体の細部を確認することができた。また、図 5.15 に示すように、体験者は複数のミニチュア模型のレイアウトを変化させ、体験環境を自由に作り変えながら体験することができた。

システム体験時の様子　　体験者がHMDで見る映像

図 5.13: システム体験時の様子と体験者が HMD で見る映像（右目用）の例

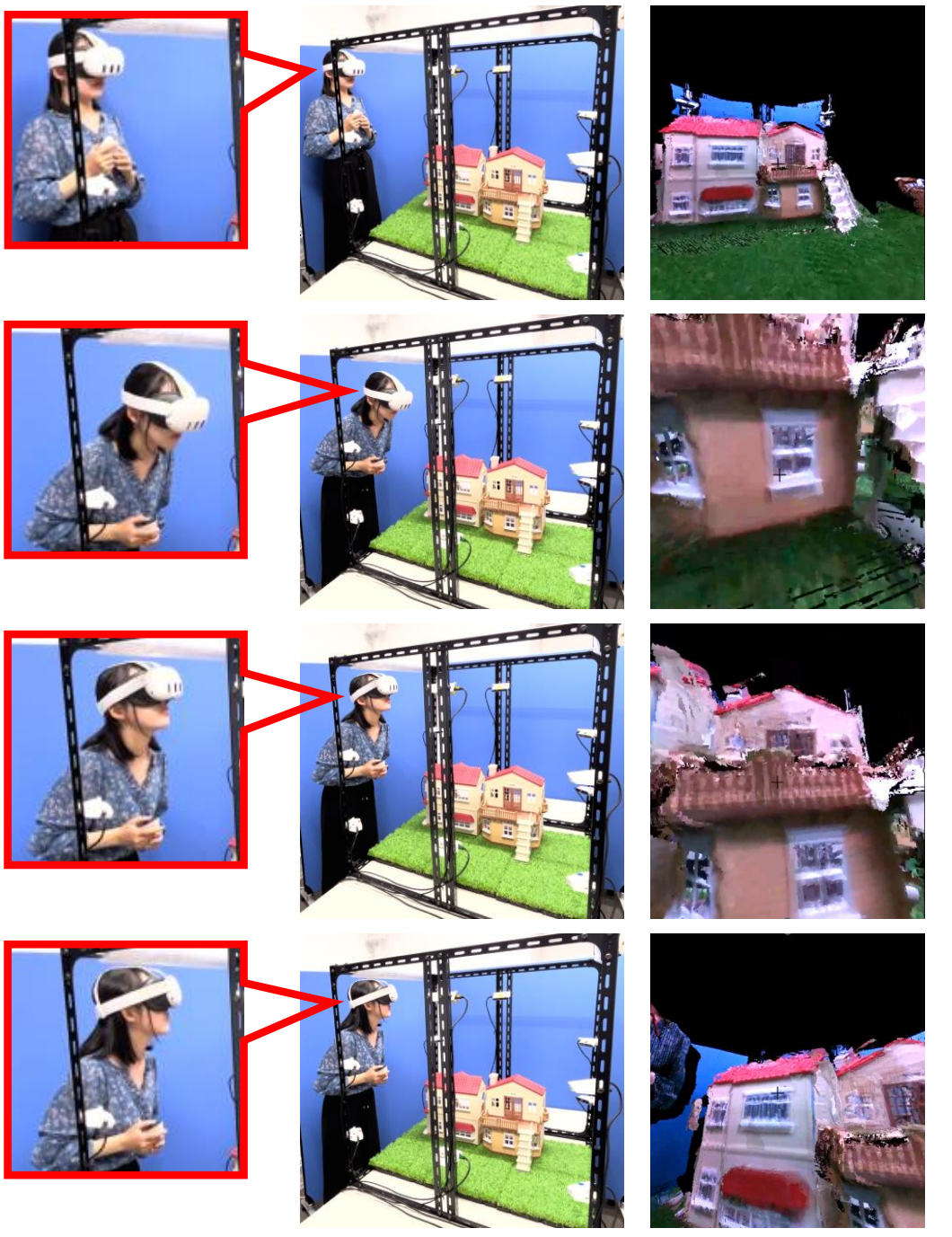
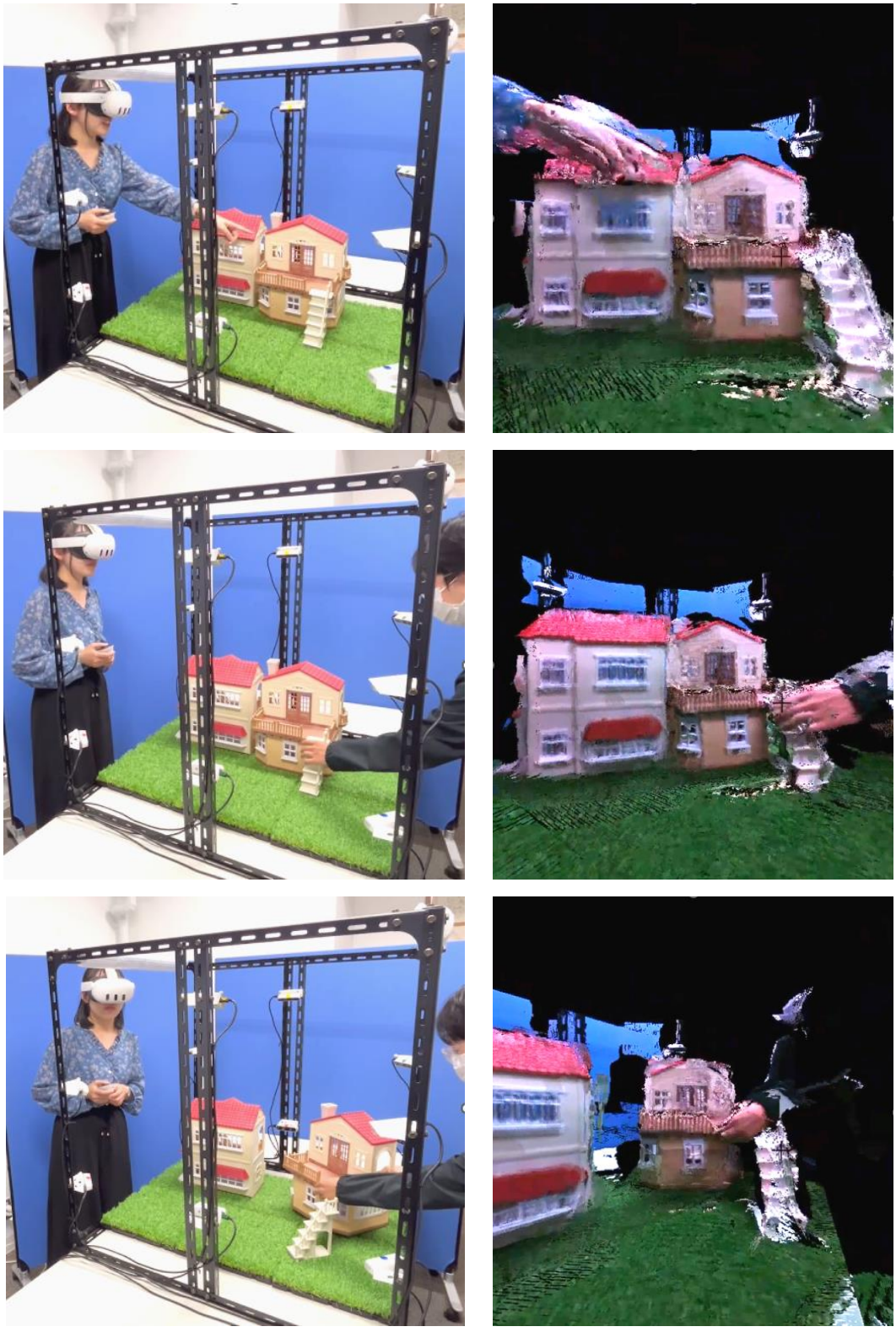


図 5.14: 被写体に視点を近づけている様子



システム体験時の様子　　体験者がHMDで見る映像

図 5.15: 環境を作り変えながら体験している様子

第 6 章 結論

本研究では、小空間に対する高精細な自由視点画像をリアルタイムに生成する手法を実現し、動的に変化する実世界小空間への没入体験を可能とするシステムを開発した。また、実現した手法を、生成画像の精度の観点から評価した。

自由視点画像の生成では、最初に、没入対象となる小空間を複数の位置に配置した RGB-D カメラで撮影し、デプス画像とカラー画像を取得した。次に、取得した複数のデプス画像をノイズ除去処理を適用しながら融合することで、高解像度な自由視点デプス画像を生成した。その後、生成した自由視点デプス画像と取得したカラー画像を用いて自由視点画像の各ピクセルの色を決定し、自由視点カラー画像を生成した。その際、生成画像の色の誤りを改善するため、取得したデプス画像を用いてオクルージョンを考慮した上でカラー情報を取得した。また、生成画像の歪みを低減させるため、被写体表面の正面から撮影したカメラから優先的にカラー情報を取得した。加えて、右目用と左目用の自由視点画像をそれぞれ個別に生成して立体視を可能にし、生成した両目用の画像を体験者の頭の動きに合わせて HMD 上に提示することで、実世界小空間への没入体験を実現した。

実装したシステムを用いて自由視点画像を生成した結果、オクルージョンや被写体表面の法線方向を考慮することで生成画像の色の誤りや歪みが改善しており、オクルージョンへの対応および被写体表面の法線方向の考慮の有効性が確認された。また、提案システムでは、視点が被写体に近接した際にも高解像度な自由視点画像を生成可能であり、被写体の細部に対する優れた視認性を有していたことから、デプス画像のノイズ除去と超解像の有効性が確認された。

実際にシステムを体験したところ、HMD を被った体験者が頭を動かすと、その頭の動きに対応した自由視点画像がリアルタイムに生成され、立体映像として HMD 上に提示された。提示される映像のフレームレートは約 10〜11fps であった。体験時には、視点を被写体の特定の個所に近づけることで、被写体の細部を確認することができた。また、複数のミニチュア模型のレイアウトを変化させることで、体験者自身が環境を自由に作り変えながら体験することができた。

今後の課題としては、キャリブレーション精度の向上により、生成する自由視点画像の精度をさらに向上させる必要がある。本研究では、4.2.1 項で述べたように、Multical、

MVE、および ORB SLAM3 の 3 つのソフトウェアを用いてキャリブレーション方法を検討したが、これに加えて新たなキャリブレーション方法を導入することで、さらなる精度向上が見込まれる。また、システムを複数人で同時に体験できるように拡張することや、システム体験時の印象評価実験を実施し、本システムでの体験が体験者の心理に及ぼす影響について調査することなども今後の課題として挙げられる。

謝 辞

本研究を進めるにあたり、研究会や論文執筆などで多くのご指導やご助言を頂きました下田宏教授に心より感謝申し上げます。

研究の進め方やミーティング、論文執筆等、研究活動全般に関する様々なご指導とお力添えを頂きました石井裕剛准教授には感謝の念に堪えません。

研究内容に関する助言だけでなく、日常の些細な相談事まで、数多くのご助言を頂きました上田樹美助教に深く感謝申し上げます。

事務手続きに加え、様々な相談にも乗って頂き、昼食時など日頃から談笑の相手になってくださった、普照郁美秘書に深く感謝申し上げます。

論文執筆の際、添削などのサポートをして頂きました修士 1 回生の田中君、荒木君に深くお礼申し上げます。研究会で貴重なご意見を頂き、また、研究室での活動全般において親睦を深めて頂き、研究室生活を楽しく有意義なものにして下さった博士課程の Orchida さん、竹内さん、修士 2 回生の小野君、佐々君、二神君、松岡君、Sun 君、修士 1 回生の阿部さん、尾籠君、松永君、工学部電気電子工学科 4 回生の尾崎君、神田君、宮本君、山﨑君に深く感謝申し上げます。

最後に、これまでの日常生活において支えとなってくださった家族や友人たちにも、この場を借りて深く感謝申し上げます。

参 考 文 献

[1] Microsoft PowerPoint, https://www.microsoft.com/ja-jp/microsoft-365/powerpoint/ (Accessed on 2/1/2025).

[2] Microsoft Whiteboard, https://www.microsoft.com/ja-jp/microsoft-365/microsoft-whiteboard/digital-whiteboard-app/ (Accessed on 2/1/2025).

[3] Adobe Acrobat, https://www.adobe.com/jp/acrobat.html/ (Accessed on 2/1/2025).

[4] Google Slides, https://workspace.google.com/intl/ja/products/slides/ (Accessed on 2/1/2025).

[5] Lucidchart, https://www.lucidchart.com/pages/ja/ (Accessed on 2/1/2025).

[6] 下山田 隆, 清野 聡子：豪雨災害体験の図化と模型制作による中学生の防災意識の伸長, 土木学会論文集 B1(水工学), **76**(2), pp.I_499–I_504 (2020).

[7] 戸田 圭一, 石垣 泰輔, 安田 誠宏, 馬場 康之, 中島 隆介：ジオラマタイプのミニチュア模型を用いた水防災教育の実践, 京都大学防災研究所年報, (60), pp.692–700 (2016).

[8] 野口 宇宙, 福崎 竜之輔, 安東 弘泰：ミニチュア模型を用いた自律運転車群に対する局所的経路選択制御によるエネルギー効率への影響の考察, 自動制御連合講演会講演論文集, **65**, pp.592–595 (2022).

[9] IT 用語辞典, https://e-words.jp/w/AR.html (Accessed on 2/1/2025).

[10] IT 用語辞典, https://e-words.jp/w/VR.html (Accessed on 2/1/2025).

[11] Ya-mei Feng, Dong-xiao Li, Kai Luo, Ming Zhang: Asymmetric bidirectional view synthesis for free viewpoint and three-dimensional video, IEEE Transactions on Consumer Electronics, **55**(4), pp.2349–2355 (2009).

[12] Siren Bato, Yoshinori Dobashi, Tsuyoshi Yamamoto: Image-Based Rendering Using Unstructured Image Set, 2009 Second International Workshop on Computer Science and Engineering, pp.288–292 (2009).

[13] 藤吉 弘亘, 梅田 和昇, 山本 和彦：画像入力・計測技術の新展開, 精密工学会誌, **75**(2), pp.228–232 (2009).

[14] 松岡 龍治：画像による 3 次元計測, 日本写真学会誌, **67**(5), pp.448–456 (2004).

[15] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, Andrew Fitzgibbon: KinectFusion: real-time dense surface mapping and tracking, 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pp.127–136 (2011).

[16] Brian Curless, Marc Levoy: A volumetric method for building complex models from range images, Proceedings of The 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp.303–312 (1996).

[17] A. Laurentini: How far 3D shapes can be understood from 2D silhouettes, IEEE Transactions on Pattern Analysis and Machine Intelligence, **17**(2), pp.188–195 (1995).

[18] 海老澤 一生, 東海林 健二, 外山 史, 森 博志：視体積交差におけるカメラ配置の最適化, 情報科学技術フォーラム講演論文集, pp.151–154 (2012).

[19] ウ 小軍, 延原 章平, 和田 俊和, 松山 隆司：多視点映像からの実時間 3 次元形状復元とその高精度化, 技術報告 2(2001-CVIM-131) (2002).

[20] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms, 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), pp.519–528 (2006).

[21] Yasutaka Furukawa, Jean Ponce: Accurate, Dense, and Robust Multiview Stereopsis, IEEE Transactions on Pattern Analysis and Machine Intelligence, **32**(8), pp.1362–1376 (2010).

[22] S.M. Seitz, C.R. Dyer: Physically-valid view synthesis by image interpolation, Proceedings IEEE Workshop on Representation of Visual Scenes (In Conjunction with ICCV'95), pp.18–25 (1995).

[23] Marc Levoy, Pat Hanrahan: Light field rendering, Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp.31–42 (1996).

[24] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, Marc Levoy: High performance imaging using large camera arrays, ACM Trans. Graph., **24**(3), pp.765–776 (2005).

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng: NeRF: representing scenes as neural radiance fields for view synthesis, Commun. ACM, **65**(1), pp.99–106 (2021).

[26] Alex Yu, Vickie Ye, Matthew Tancik, Angjoo Kanazawa: pixelNeRF: Neural Radiance Fields from One or Few Images, 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.4576–4585 (2021).

[27] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, Thomas Funkhouser: IBRNet: Learning Multi-View Image-Based Rendering, 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.4688–4697 (2021).

[28] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, Hao Su: MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo, 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp.14104–14113 (2021).

[29] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, Noha Radwan: RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.5470–5480 (2022).

[30] Julian Chibane, Aayush Bansal, Verica Lazova, Gerard Pons-Moll: Stereo Radiance Fields (SRF): Learning View Synthesis for Sparse Views of Novel Scenes, 2021

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.7907–7916 (2021).

[31] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, Daniel Duckworth: NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.7206–7215 (2021).

[32] Multical, https://github.com/oliver-batchelor/multical/ (Accessed on 2/1/2025).

[33] MVE, https://github.com/simonfuhrmann/mve/ (Accessed on 2/1/2025).

[34] ORB SLAM3, https://github.com/UZ-SLAMLab/ORB_SLAM3/ (Accessed on 2/1/2025).

[35] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, Juan D. Tardós: ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM, IEEE Transactions on Robotics, **37**(6), pp.1874–1890 (2021).

[36] Masahiro MURAYAMA, Toyohiro HIGASHIYAMA, Yuki HARAZONO, Hirotake ISHII, Hiroshi SHIMODA, Shinobu OKIDO, Yasuyoshi TARUTA: Depth Image Noise Reduction and Super-Resolution by Pixel-Wise Multi-Frame Fusion, IEICE Transactions on Information and Systems, **E105.D**(6), pp.1211–1224 (2022).

[37] Intel RealSense Depth Camera D435, https://www.intelrealsense.com/depth-camera-d435/ (Accessed on 2/1/2025).

[38] Brand new feature for Intel RealSense Stereo Depth Cameras-Self Calibration, https://www.intelrealsense.com/self-calibration-for-depth-cameras/ (Accessed on 2/1/2025).

[39] Intel RealSense SDK, https://github.com/IntelRealSense/librealsense/ (Accessed on 2/1/2025).

[40] Boost C++ Libraries, https://www.boost.org/ (Accessed on 2/1/2025).

[41] Unity Technologies, https://unity.com/ (Accessed on 2/1/2025).

[42] Meta XR Interaction SDK, https://assetstore.unity.com/packages/tools/integration/meta-xr-interaction-sdk-265014/ (Accessed on 2/1/2025).

[43] Point Cloud Library, https://pointclouds.org/ (Accessed on 2/1/2025).