Title	実空間での位置合わせを可能とするAR コンテンツオーサリング手法の提案
Author(s)	碓﨑，敬祐
Citation	
Issue Date	2019-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/15809
Rights	
Description	Supervisor:宮田　一乗，先端科学技術研究科，修士（知識科学）

修士論文

実空間での位置合わせを可能とする AR コンテンツオーサリング手法の提案

1710049　碓﨑　敬祐

主指導教員　宮田 一乗
審査委員主査　宮田 一乗
審査委員　永井 由佳里
西本 一志
由井薗 隆也

北陸先端科学技術大学院大学
先端科学技術研究科
(知識科学)

平成 31 年 2 月

目 次

図 目 次

1　はじめに

2016 年は VR 元年とよばれ 3DCG(3 次元コンピューターグラフィックス) の技術は一般的になり、これからさらに普及していくと予想 [1] されている。3DCG コンテンツはゲームエンジンの発展により個人でもハイクオリティな作品の制作が可能になった。そして、VR(Virtual Reality) は HMD(Head Mounted Display) の発展により、AR(Augmented Reality) は様々なセンサーやカメラを搭載したスマートフォンとの相性がよく、より手軽に体験が可能となった。AR に関しては Apple iOS 向けの ARKit [2] と Google android 向けの ARCore [3]、Unity [4] の vforia [8] といったフレームワークの発展が著しい。モーショントラッキングによるデバイスの位置推定、水平・垂直面の検出による机や地面、壁面を利用したオブジェクト配置、環境の照明状態の推定によるオブジェクトの明暗の表現などが可能になり利用者はよりリアルな体験を得られるようになっている。

一方で、図 1 に示す投資金額のグラフを比較 [9] しても AR 開発は VR 開発と違い普及しているとは言いがたい。その理由として、AR コンテンツの開発と利用が VR コンテンツに比べて困難であると考える。

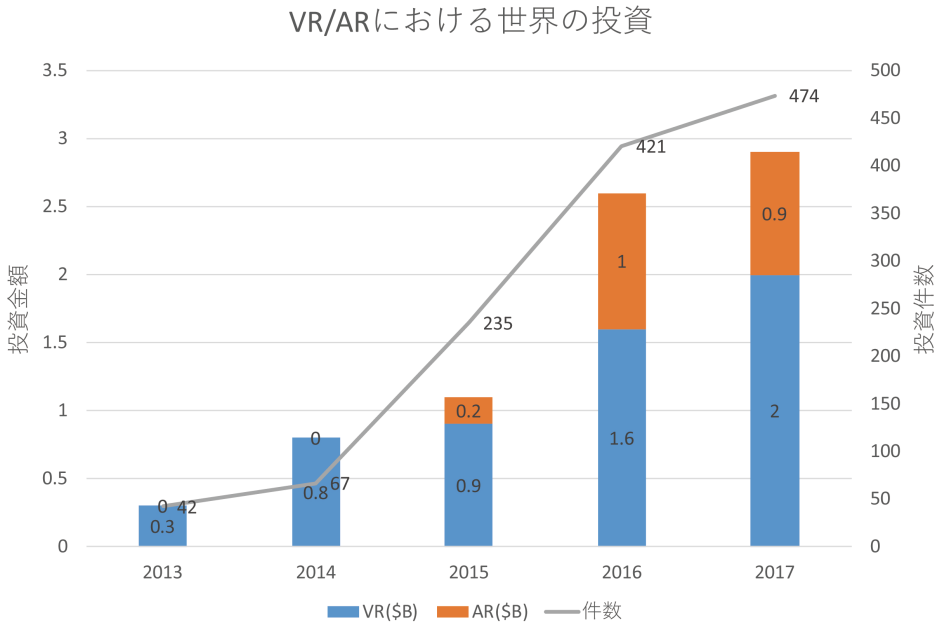


図 1: vrvca”VR/AR Global Investment Report & Outlook 2018”

VR にはない AR の最大の利点は実空間を利用できることである。しかしながら、実際に開発を行うと現状では図 2 に示すように基本的にマーカー以外何もない空間にオブジェクトを配置することで 3DCG 空間を作成していく事になる。これではコンテンツがどのような環境で利用されるのかわからないため実空間と作用するような複雑なコンテンツの制作を行うことは難しい。

図 2: Unity で Vuforia を用いた AR コンテンツの開発画面

AR コンテンツを利用する際には利用者がコンテンツの初期位置をマーカーを配置するなどして決定する必要がある。この方式は図 3 のような家具やキャラクターといった単体のオブジェクトを不特定な空間に配置して鑑賞する事には向いているが、空間を作ることには向いていない。また、実空間には広さの制限や家具などの障害物があり、VR のような大規模なコンテンツを体験することは難しい。

このような自由度から AR コンテンツの製作者は実利用される空間に制約を設けたり、イベント会場などの特定の実空間に整合したコンテンツを設計することは困難である。以上のような理由からオブジェクトを配置して鑑賞するような単純なコンテンツが主流になっている。

本研究では、コンテンツ作成の際に SLAM(Simultaneous Localization and Mapping) で作成した物の配置や壁の位置などの周辺環境の情報を記述した環境地図を使用する。コンテンツが利用される空間の把握が可能になり、特定空間において実空間を利用したコンテンツの制作を容易に

するARコンテンツのオーサリング手法を提案する。実空間に整合したコンテンツを作成できコンテンツ制作者の制作意図を反映したオブジェクトの配置が可能になることによって、インスタレーションなどのアート作品やイベント会場の演出などに使用が可能となる。

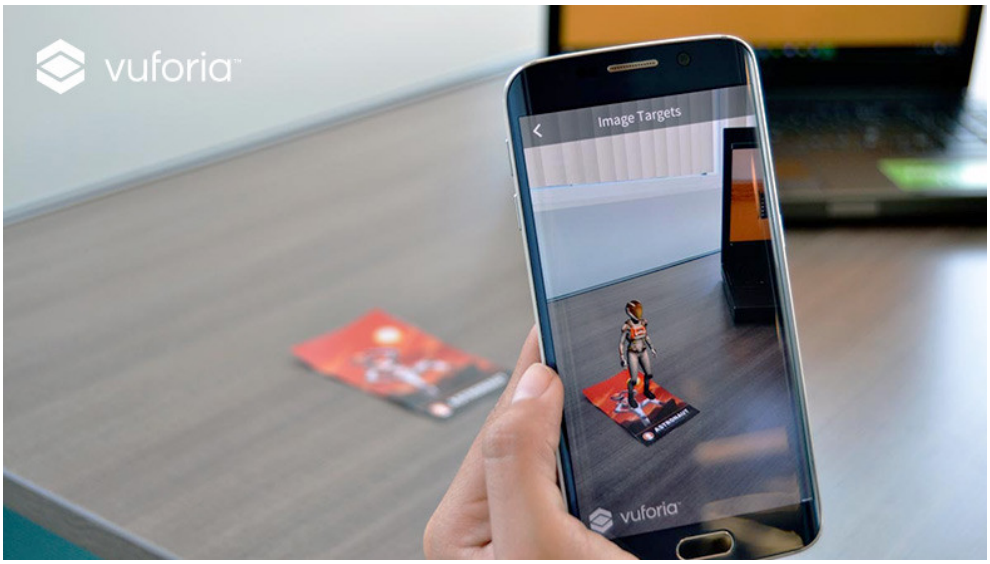


図 3: 図 2 の左端のマーカーを使用した AR コンテンツの例

2　関連事例

本章では AR コンテンツの制作環境と認識方法、いくつかの AR コンテンツの実例、実環境に合わせて 3D モデルを重畳表示する研究について述べ、本研究の位置づけを示す。

2.1　AR の制作環境

現状の AR コンテンツは Unity [4] や Unreal Engine [5] といったゲームエンジン、もしくは Android Studio [6] や Xcode [7] と言った IDE(Integrated Development Environment:統合開発環境) に、Unity に標準搭載されている Vuforia [8]、Apple の ARKit、Google の ARCore などのフレームワークを用いて作成される。

図 4 のようにゲームエンジンでは基本的に何もない空間にオブジェクトを配置することで 3DCG 空間を作成していく。フレームワークを用いることで前述した図 2 のようなマーカーなど AR の開発に必要なアセットが使えるようになる。図 5 のように IDE を使用する場合ではコードで開発するのでそもそも GUI 的な支援は行われていない。

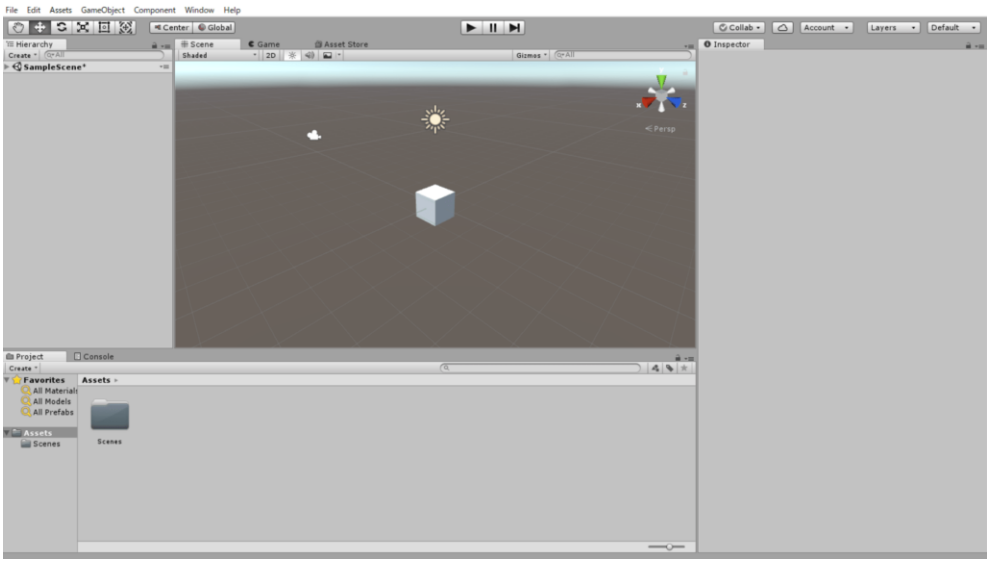


図 4: Unity での開発画面

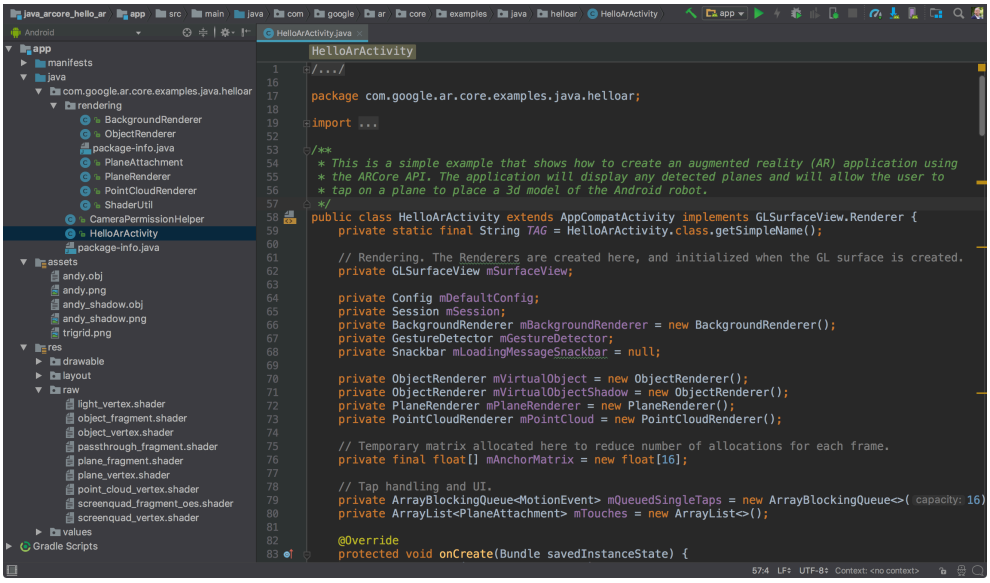


図 5: Android Studio で ARCore を用いた AR コンテンツの開発画面

2.2 認識方法

体験者はデバイスを用いて 3D オブジェクトの配置場所を認識し、表示する。AR オブジェクトを配置する場所の位置決定方法として以下の手法がある。

2.2.1 マーカー式

マーカーとなるパターン化した図形を決め、それと特徴が一致する場所に 3D オブジェクトを表示する方法である。

他の手法に比べて認識精度が高く、利用者が表示したい位置にオブジェクトを配置することが可能である。そのため一般的に AR として普及しているのはこのマーカー式である。しかし、マーカーを画角内に収める必要があり、本来 3D オブジェクトが表示されている場所であってもマーカーが画角内にない場合は表示できない欠点を持つ。

2.2.2　マーカーレス式

空間の特徴点を認識することでパターン化されたマーカーを使用することなく、3D オブジェクトを表示することができる。

画像特徴点を用いることでパターン化された図形だけではなく画像や物体をマーカーとして使用することができる。環境内にあるものを使用することができるのでより自然に AR を体験することができる。しかし、マーカー式と同じく画角内に特徴点を収める必要があるので同様の欠点を持つ。最近ではこの方式が一般的にマーカー式 AR と呼ばれることが多い。

従来のマーカーレス式 AR に代わり、近年は SLAM やモーションキャプチャなどを利用したカメラの位置推定を用いた AR の事例が、マーカーレス式 AR と呼ばれるようになっている。画像特徴点を認識することにより、空間認識を自己位置推定によってカメラの姿勢推定が行えることから、AR との相性が良い。この方式は特定の画像や立体によるマーカーを準備する必要がなく、また画角内にマーカーを収める必要もないため、マーカー式の場合画角に収まらないような大きな 3D オブジェクトでも表示が可能である。しかしながら、3D オブジェクトを表示する原点を体験者が決定する必要があり、制作者が意図した場所に 3D オブジェクトを配置できない。

2.2.3　ロケーションベース型

GPS などのセンサーを用いて得られた位置情報を使って 3D オブジェクトを表示する方法である。最近ではポケモン GO で話題になった。

GPS を使用する場合は世界規模で使用することが可能であるが、精度の問題があり詳細な位置合わせができず屋内ではさらに精度が悪くなる欠点がある。また、精度を向上させるにはセンサーを多数配置する必要がある。

2.3　コンテンツ

2.3.1　IKEA Place

IKEA Place [10] は IKEA が配信している家具配置アプリケーションである。iOS 版は ARKit を android 版は ARCore を使用して構築されている。

使い方を図 6 に示す。最初に家具を配置する面を取得する。次に配置したい家具を選択する。最後に表示された家具を任意の場所に配置する。

アプリケーションの利用者は家具を自室に配置した時どのような雰囲気であるかを知ることができる。一方、コンテンツを提供している IKEA 側はどのような室内で使用されるか事前に知ることができないので商品の宣伝のためにリアルスケールの AR モデルルームといったコンテンツを用いての商品の紹介などはできない。

(a) 選択画面　(b) 選択した画面　(c) 面の取得　(d) 家具の表示

図 6: IKEA Place の利用画面 [10]

2.3.2 Google Art & Culture

Google が配信している AR 美術館アプリケーション Google Art & Culture [11] の機能の 1 つである Pocket Gallery [12] はヨハネス・フェルメールの作品を一挙に鑑賞できるコンテンツである。IKEA Place と同じく iOS 版は ARKit を android 版は ARCore を使用して構築されている。

使い方を図 7 に示す。最初にオブジェクト配置可能な面を取得し、俯瞰した AR ギャラリーを配置する。その後、鑑賞したい部屋をタップすると、AR ギャラリーの中に入ることができる。

スマートフォンを動かすことで視点の移動は可能だが、移動はタップ操作で行われる。すなわち、ギャラリー内の移動をするために実空間を移動してもその動きは羽根井されない。AR を用いて 3DCG で制作した仮想空間に入り込む作品ではあるが、コンテンツ内の仮想空間と実空間の関連性がないので没入感は感じられない。

(a) 面の取得　(b) ギャラリーの配置　(c) ギャラリー内の様子

図 7: Pocket Gallery の利用画面 [11]

2.4　関連研究

現実空間の特定の場所に意図したオブジェクトを配置する方法に関して、以降で述べる。意図した場所にオブジェクトを配置できるようになることによって、仮想オブジェクトと実空間に関連性を持たせることが可能になる。

2.4.1　SLAM を使用した AR 可視化

力学現象の数値シミュレーション結果の可視化を行った池田 直旺らの研究 [13] では、SLAM 技術に基づく AR 可視化システムを用いた。事前に用意したシミュレーション結果モデルと実空間の位置合わせの初期位置合わせにマーカーを用いている。この方式は正確な初期位置を取得できるため、より正確な AR を実装できる。

図 8 は構造物周りの三次元気流シミュレーション結果を実際の対象構造物に対して AR 可視化を行っている。

一方でマーカーの設置位置がズレてしまうと SLAM の初期位置がズレてしまうのでオブジェクトがズレて表示されてしまう欠点を持つ。

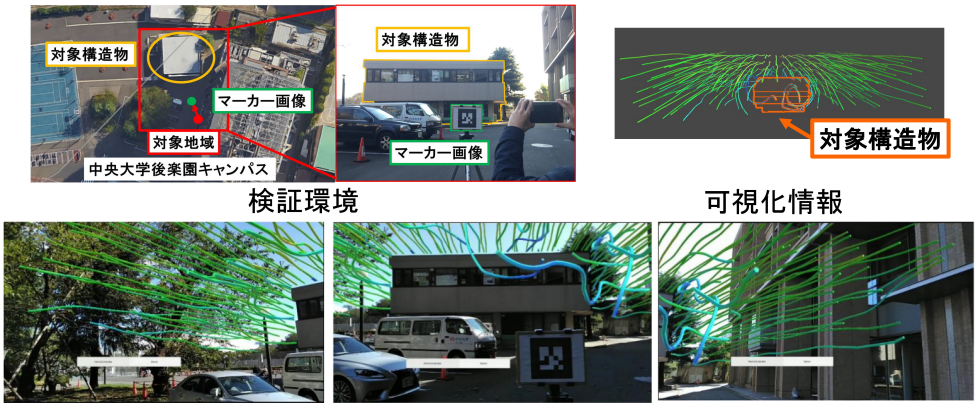
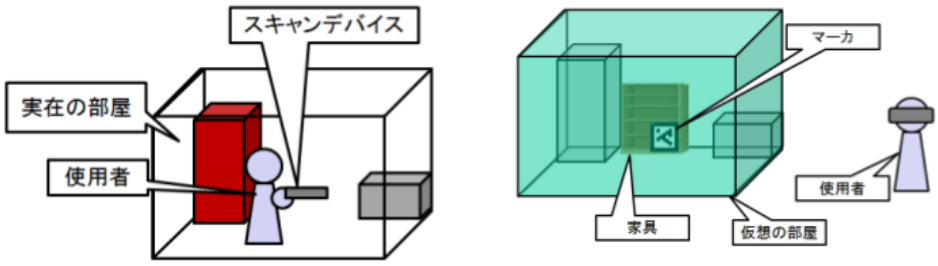


図 8: SLAM 技術に基づいた AR 可視化システム [13]

2.4.2　3 次元空間モデルの使用

家具に対して仮想の部屋を重畳する大寺らの研究 [14] では、事前に部屋のモデルを取得した後に家具に対して仮想の部屋の位置を固定する。従来の研究の主流は実空間に対して仮想の家具を重畳し表示することであったが、この研究では実物の家具に対する部屋環境の提示を行う。



(a) 実在の部屋空間のスキャニング　　(b) 仮想の部屋空間の提示

図 9: MR を用いた家具に対する部屋環境の可視化

2.5　本研究の位置づけ

近年、ゲームエンジンや AR 開発フレームワークの発展によって AR コンテンツは簡単に制作が可能になった。しかしながら現状の AR コンテ

ンツ開発では特定の実空間を利用することが困難であるために AR の利点を生かせず、小規模で単純なコンテンツになってしまっている。

一方で池田 直旺らの研究によって制作者と利用者が同一である場合はマーカーを用いることによって仮想オブジェクトを実空間の任意の位置に表示可能である。しかし、コンテンツの制作者と利用者が同一である場合は少なく、マーカーを用いるコンテンツの位置合わせ手法は適していない。

本研究ではコンテンツ制作者が指定した特定空間において実空間を利用したコンテンツの制作を容易にするオーサリング手法を提案する。

3　実空間での位置合わせを可能とするARコンテンツオーサリング手法の提案

本章では、本研究で設計したARコンテンツ制作システムについて説明する。はじめにシステムの概要として、システムの流れを述べる。次に使用するSLAMについて説明し、最後にシステムの各工程について詳述する。

3.1　システム概要

図10にシステムの流れを示す。本システムは最初にコンテンツ制作者がSLAMで環境地図を構築する。次にARシステムに環境地図および、表示するコンテンツの3Dモデルを読み込ませ位置合わせを行う。

利用者はシステムを用いて入力画像からSLAMで位置推定を行い、AR制作者が作成したARコンテンツを表示する。

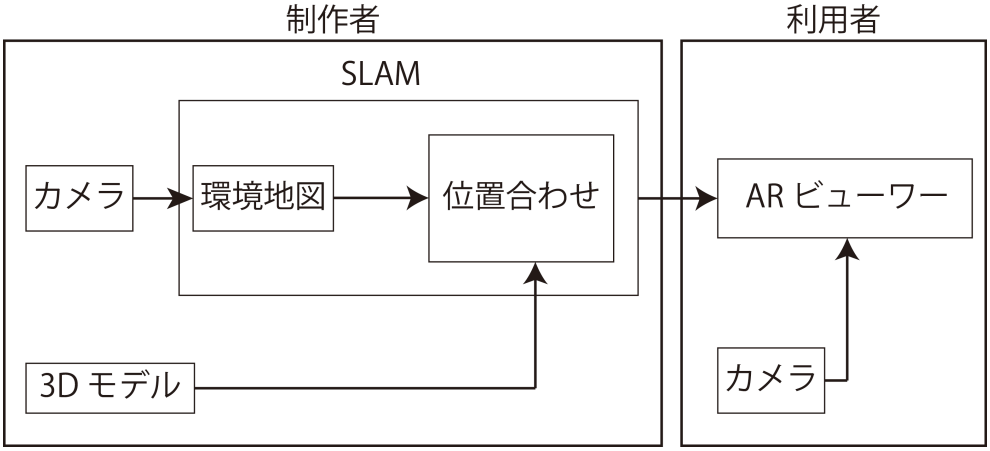


図 10: システム概要

3.2　SLAM

SLAM(Simultaneous Localization and Mapping) とは物の配置や壁の位置などの周辺環境の情報を記述した環境地図作成とカメラの自己位置推定を同時に行う技術である。

本システムはORB-SLAM2 [15] を使用して環境地図作成とカメラ位置の推定を行う。

3.2.1 ORB-SLAM2

ORB-SLAM2 は単眼カメラ、ステレオカメラ、RGB-D(カラー画像に加えて深度情報も取得できる) カメラを用いたリアルタイム SLAM ライブラリである。Tracking、Local mapping、Loop closing の 3 スレッドを並列実行し、同時に推定したマップポイントを使用した点群地図を作成する。

図 11 に ORB-SLAM2 の実行画面を示す。

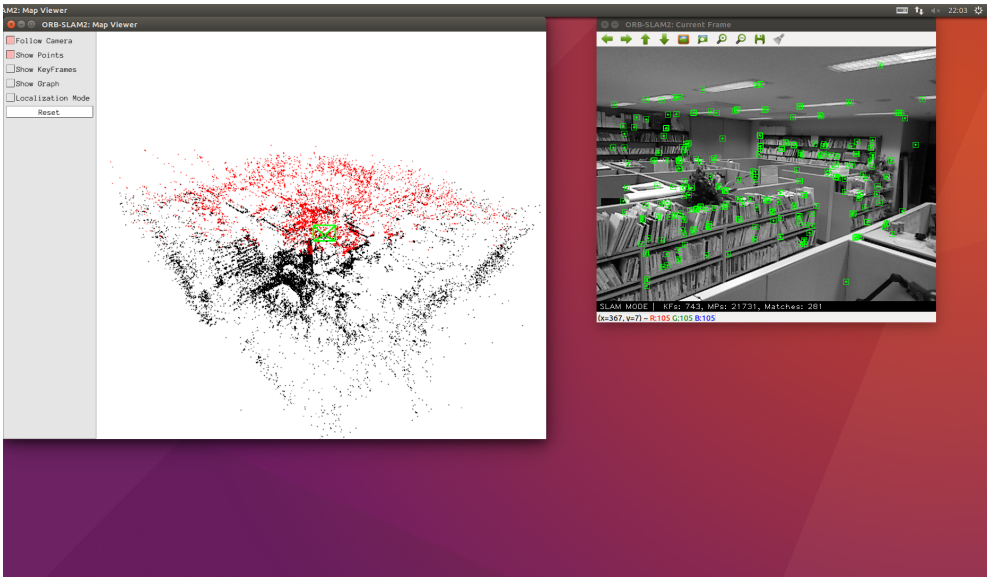


図 11: ORB-SLAM2

Tracking このスレッドでは図 11 の右のウィンドウから緑色の矩形に示すように ORB(Oriented FAST and Rotated BRIEF) [16] 特徴点の抽出を行い、カメラの位置と姿勢を推定する。入力フレームのうち以下の条件を満たしたものはキーフレームとして保存され、点群地図作成に使用される。

- カメラの位置と姿勢の推定をロストしていた状態から復帰後２０フレーム経過していること。

- 次に説明する Local Mapping が行われていない、または最後のキーフレーム決定から 20 フレーム以上経過していること。
- 現在のフレームが 50 点以上の特徴点を認識していること。
- キーフレームの重複を避けるためにフレームで認識される特徴点の一致率が 90%以下であること。

ORB とは基本的に FAST キーポイント検出器と BRIEF 特徴記述子を融合させたものである。

FAST(Features from Accelerated Segment Test) [17] はすべての画素に対してコーナー、非コーナーのラベルを与える。FAST におけるコーナーは以下のように決定される。画像中の pixel を 1 つ選び p とする。周囲の円上の 16pixel のうち連続した n 個以上の画素が p と比較して t 以上に明るいか暗い場合にコーナーとしてのラベルを与える。FAST によるコーナーのラベル付けの例を図 12 に示す。

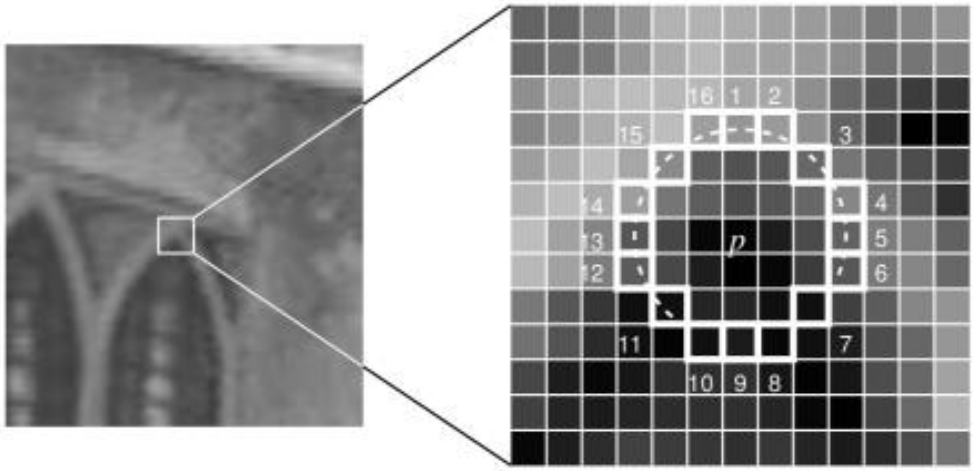


図 12: FAST によるラベル付けの例 [17]

BRIEF (Binary Robust Independent Elementary Features) [18] は特徴量記述子を使うことなく 2 点の輝度差で二値特徴量を生成する特徴記述手法である。

Local mapping　このスレッドでは新しくキーフレームが挿入された場合に点群地図の生成と更新が行われる。新しいキーフレームの特徴点を

追加し、これまでのキーフレームで保存された特徴点と一致するものが少ないものは削除する。現在のキーフレームと隣接するキーフレームのMapPointsとカメラ姿勢をLocal BA(Bundle Adjustment)でカメラの位置と姿勢の推定を最適化する。このBAを行う際の複雑さを抑え、同じ環境で長時間操作を可能にするために重複したキーフレームは随時削除される。

BA(Bundle Adjustment)とは、3次元シーンを複数のカメラで撮影した画像から、シーンの3次元形状と各カメラの撮影位置や内部パラメータをすべて同時に、カメラの透視投影モデルに合致するように定めることである[19]。

Loop closing　このスレッドでは新しいキーフレームごとに以前地図を生成された部分を再訪したかどうかの検索を行い、ループを検出した場合にはglobal BAで最適化を行う。

マップの更新が行われなくなると地図全体の誤差を少なくするために、保存されたキーフレームの特徴点全てを使用したFULL BAを行い最適化を行う。

3.2.2　ORB-SLAM2のAR機能

ORB-SLAM2にはAR機能のデモを図13に示す。このデモはARビューワー内でキューブを重畳表示できる機能を持つ。緑の点は取得された特徴点である。キューブの配置場所の決定方法を以下に示す。まずSLAMで取得した点群を頂点とし、面を取得する。続いてx-zに並行な面に対して法線ベクトルyの原点にキューブの底面の中心を置く。これによって面に対して置かれたようにキューブを配置することが可能なデモである。

このシステムで配置されるオブジェクトのモデル座標の原点はワールド座標系で指定されているのではなく、姿勢推定されたカメラ座標の原点との相対座標で位置が決定されている。カメラが移動すると推定された原点の位置もワールド座標系に対して変化してしまうので、そのままではカメラの移動とともに配置されたキューブも移動してしまう。カメラ座標はSLAMで推定され、モデル座標はカメラ座標に対して座標返還を行うことで求める。

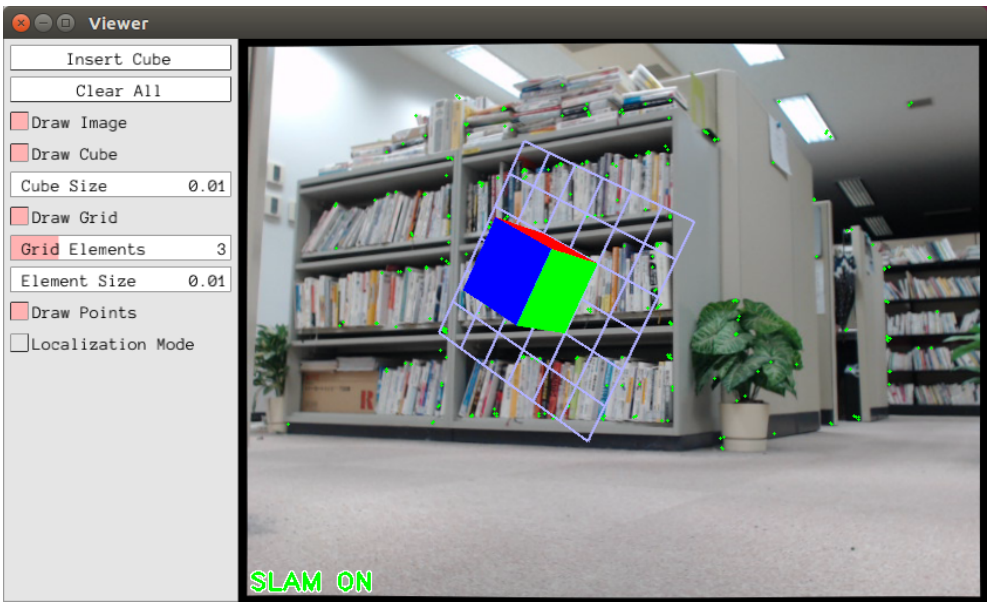


図 13: ORB-SLAM2 に実装されている AR デモの画面

3.3 提案手法

システムの各工程を以下に示す。最初に環境地図の作成を行い、次にモデルの位置調整を行う。

3.3.1 環境地図の作成

ORB-SLAM2 を用いた図 14 に示す空間の環境地図の作成を図 15 に示す。図 15 に示された中心付近の緑色の四角錐はカメラの推定位置を示し、赤い点が現在 SLAM システムで認識されているとされている特徴点、黒い点がそれ以外である。また、a、b はそれぞれ図図 14 で示された画像の撮影位置である。単眼カメラを用いて作成する環境地図にはシステムの特性上、後述するスケール誤差などが生じてしまうためより正確な環境地図が作成できる RGB-D カメラを用いる。環境地図のワールド座標の原点は ORB-SLAM2 の Tracking で最初のキーフレームが取得された場所とする。

(a) 左側から撮影

(b) 右側から撮影

図 14: 環境地図の作成を行う実空間

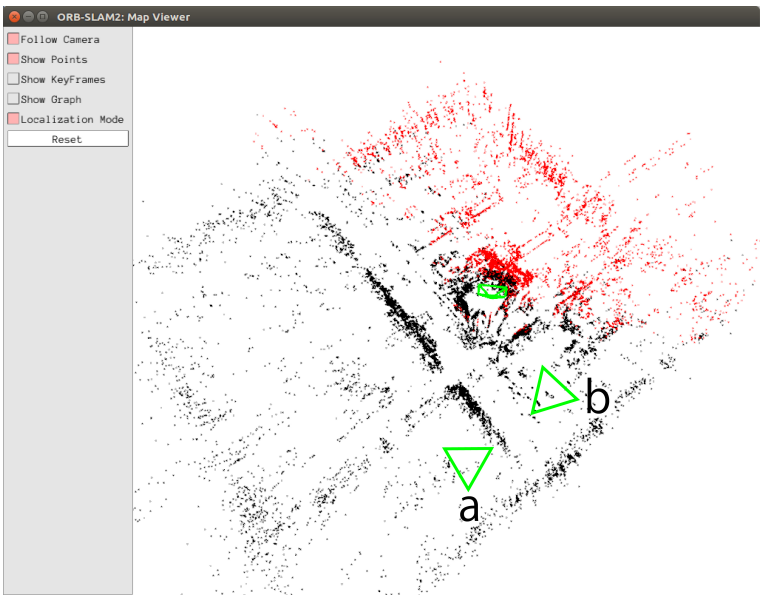
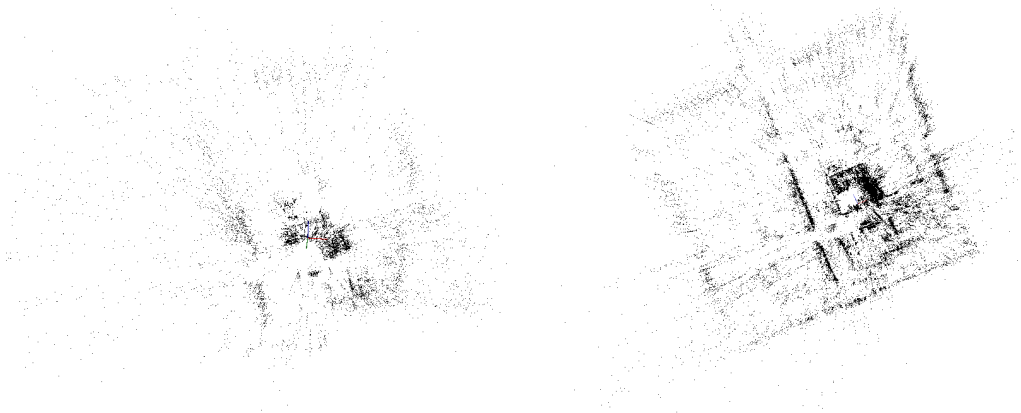


図 15: ORB-SLAM2 で得られた環境地図

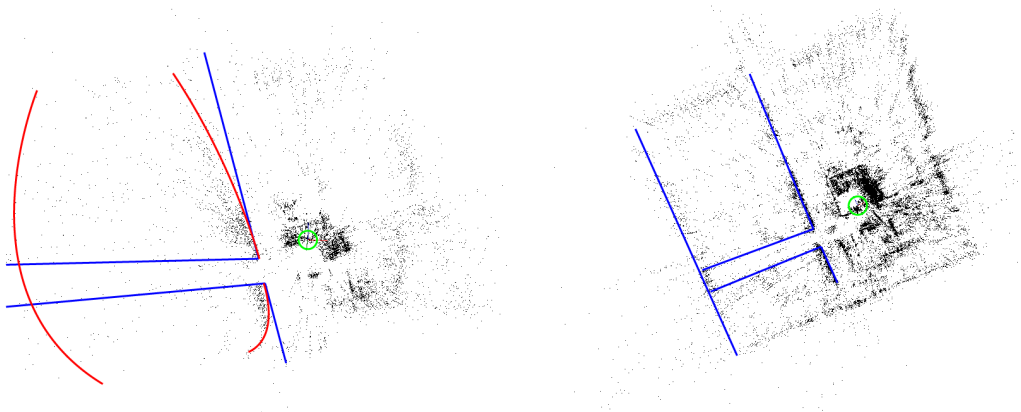
単眼カメラと RGB-D カメラで得られた環境マップを図 16 に示す。この事例では単眼カメラは Logicool HD Pro Webcam C920r、RGB-D カメラは Microsoft Kinect for Xbox 360 を使用した。単眼カメラで得られた環境地図は誤差が蓄積されることによって RGB-D カメラで得られた環境地図に比べて遠くの点の精度が低くなっている。これによって部屋空間の大きさが RGB-D カメラで得られた環境地図に比べて大きくなっており、スケールに誤差が発生している。図 17 では正しい壁面を青線で、単

眼カメラで得られた壁面の大まかな位置を赤線で、環境地図の原点を緑円で示している。



(a) 単眼カメラ　　(b) RGB-D カメラ

図 16: 取得した環境地図の比較



(a) 単眼カメラ　　(b) RGB-D カメラ

図 17: 誤差の比較

3.3.2　3D モデルの位置合わせ

提案システムの AR 編集機能は図 18 に示すように環境地図によるオブジェクトの表示位置の指定、AR カメラによるリアルタイム視覚化機能、

画像の入力画面を持つ。左上のウィンドウが環境地図、右上のウィンドウが AR ビューワー、右下のウィンドウが入力画像と検出された特徴点を表示している。環境地図、AR カメラの描画は OpenGL [20] で行われ、入力画像は OpenCV [21] を用いて ORB 特徴量を得る。

システムを起動すると入力された環境地図と 3D オブジェクトが表示される。3D オブジェクトのモデル座標の原点は環境地図の原点に配置される。そこから平行移動と回転移動の操作で任意の位置に移動させる。

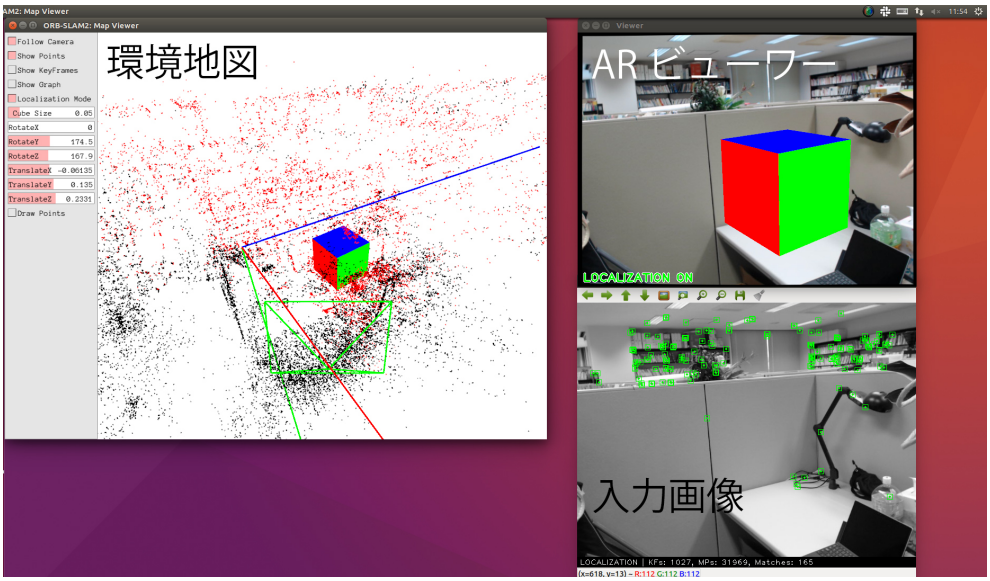
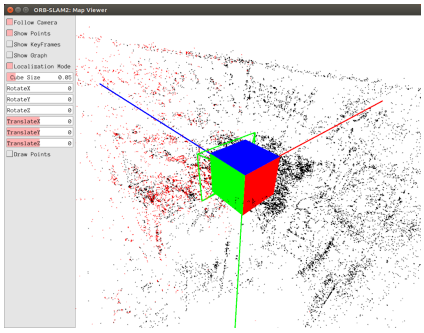
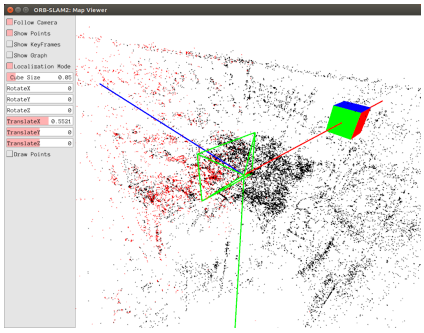


図 18: システムの編集画面

平行移動　ユーザーは x、y、z 軸方向にオブジェクトを移動させるメニューバー上のそれぞれ TranslateX、TranslateY、TranslateZ を用いてオブジェクトを表示させたい位置まで移動させる。TranslateX を図 19 に、TranslateY を図 20 に、TranslateZ を図 21 に示す。図中の座標軸はそれぞれ x 軸が赤、y 軸が緑、z 軸が青である。

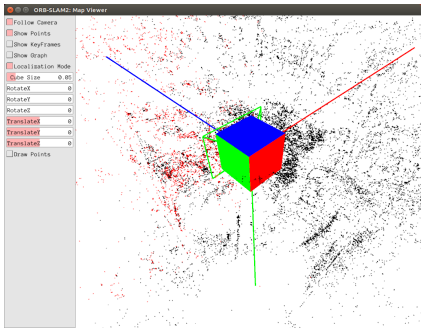


(a) 移動前

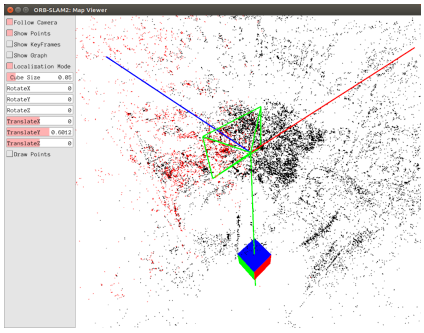


(b) 移動後

図 19: x 軸方向の平行移動

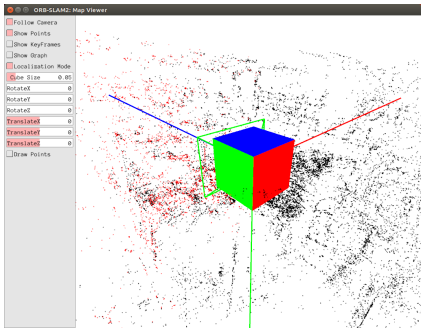


(a) 移動前

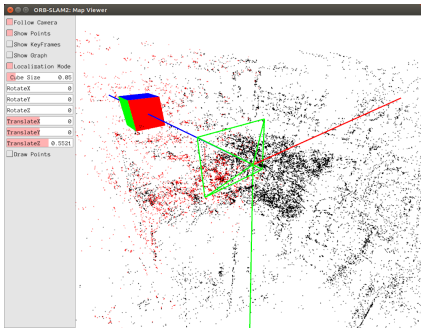


(b) 移動後

図 20: y 軸方向の平行移動



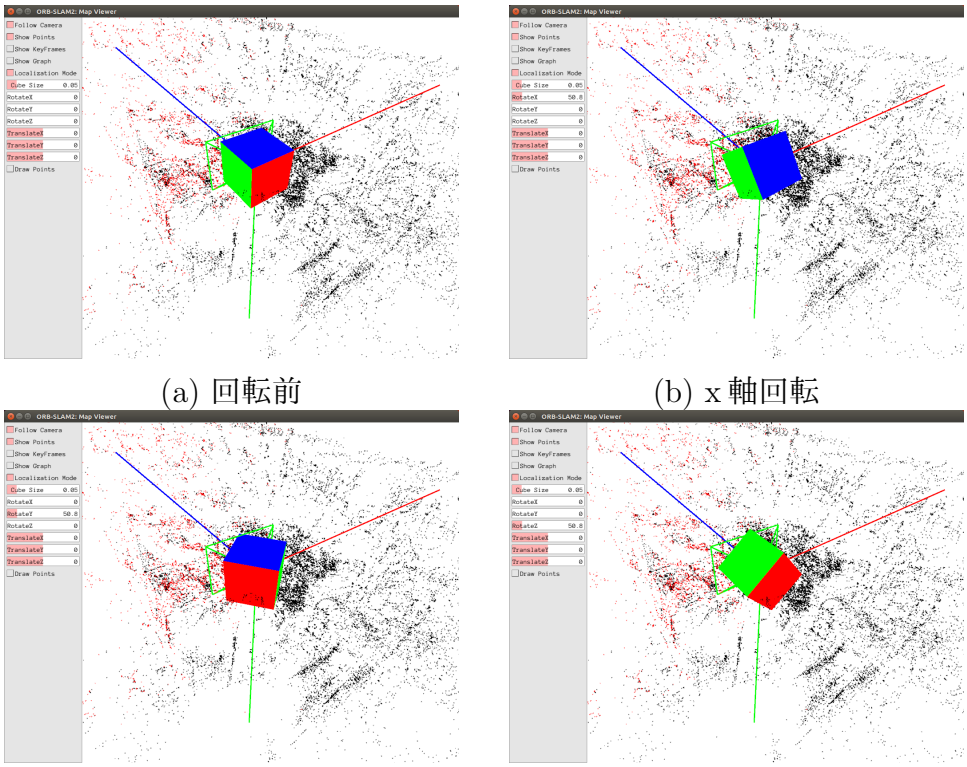
(a) 移動前



(b) 移動後

図 21: z 軸方向の平行移動

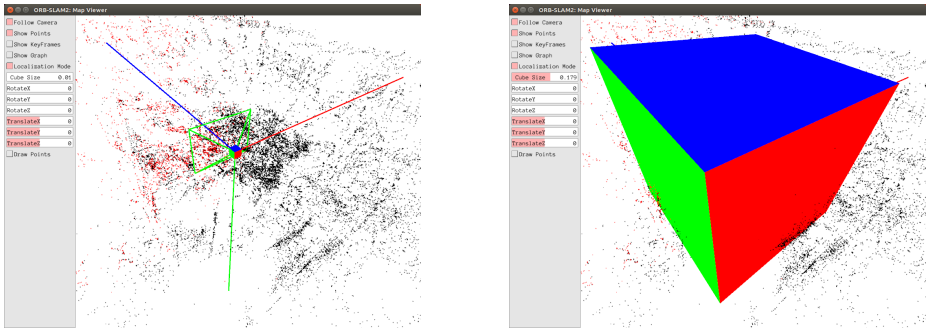
回転移動　ユーザーはx、y、z軸方向にオブジェクトを回転移動させるメニューバー上のそれぞれRotateX、RotateY、RotateZを用いてオブジェクトを表示させたい角度に調整する。オブジェクトの回転移動はモデル座標系の原点を中心に行われる。回転移動を図22に示す。図中の座標軸はそれぞれx軸が赤、y軸が緑、z軸が青である。



(a) 回転前　(b) x軸回転　(c) y軸回転　(d) z軸回転

図 22: 回転移動

拡大縮小　ユーザーはサイズ変更が可能なメニューバーにあるCube Sizeでオブジェクトの拡大縮小を行いを環境地図に合わせる。オブジェクトの拡大縮小はモデル座標系の原点を中心に行われる。拡大縮小を図23に示す。



a 縮小　　b 拡大

図 23: スケール変更

以上の行程を繰り返すことで調整を行っていく。

3.4 AR ビューワー

AR ビューワーを図 24 に示す。操作は一切必要ないので配置したオブジェクトにカメラを向けると、その場所にオブジェクトが描画される。このように、実際にどのように重畳表示されているか確認が行える。

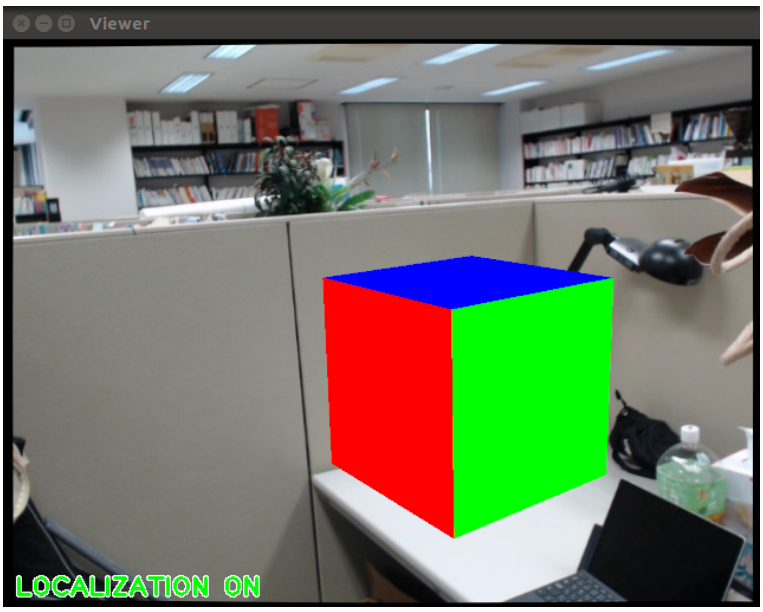


図 24: AR ビューワー

4　事例

本章では提案手法を用いた屋内と屋外の事例を紹介する。

4.1　屋内

図 14 に示す室内空間において図 25 に示すデスクに 3D オブジェクトを配置する。

図 25: デスク

4.1.1　実行環境

実行環境を以下に示す。

OS	Ubuntu 16.04 LTS
CPU	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz 3.60GHz
RAM	16.0 GB
RGB-D カメラ	Microsoft Kinect for Xbox 360
web カメラ	Logicool HD Pro Webcam C920r

環境地図作成の入力を RGB-D カメラで行い、オブジェクトの位置調整完了後の AR の確認は web カメラで行った。

4.1.2 環境地図作成

図 15 に示した環境地図を使用する。

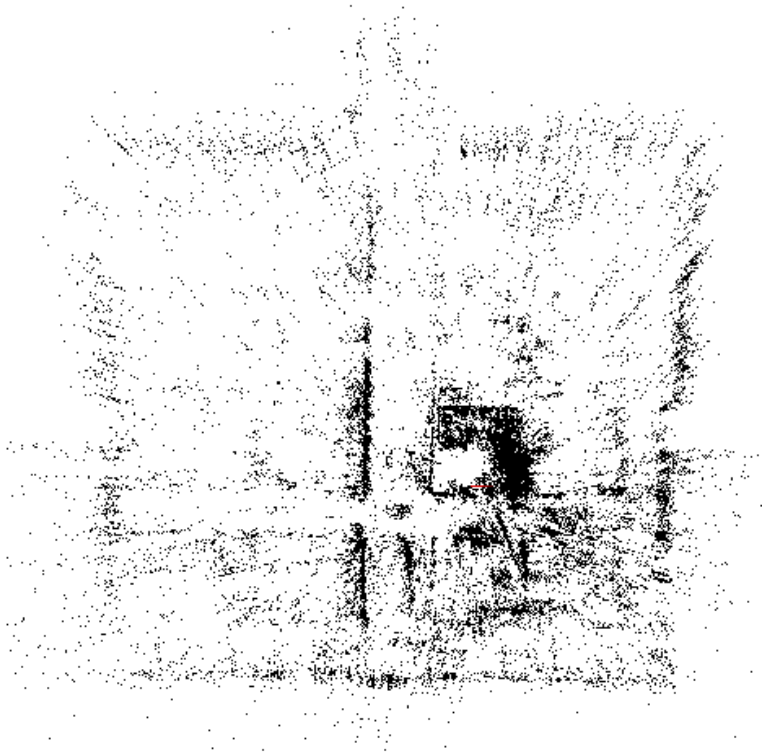


図 26: 環境地図

4.1.3 結果

環境地図での配置を図 28 で、AR ビュワーを図 27 で示す。点群の環境地図と実空間の位置関係をわかりやすくするために手動で補助線を加えた画像を図 29 に示す。

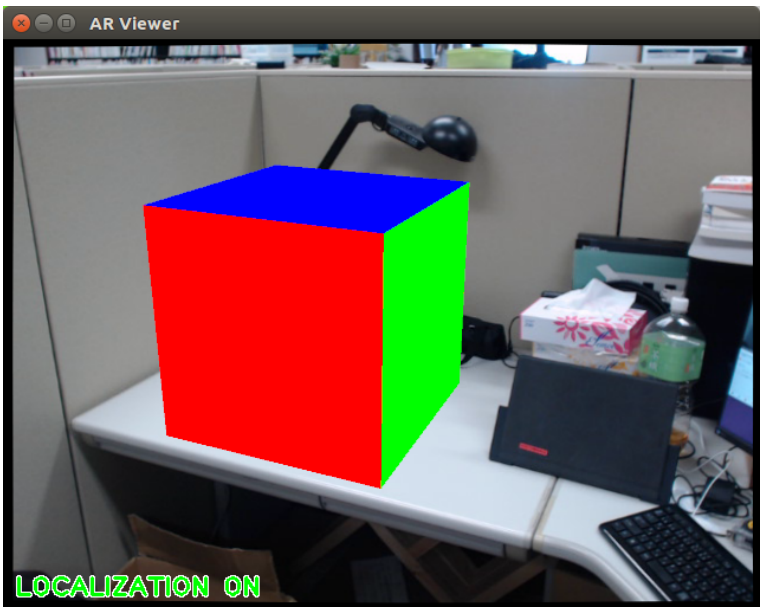


図 27: AR ビューワーでの配置

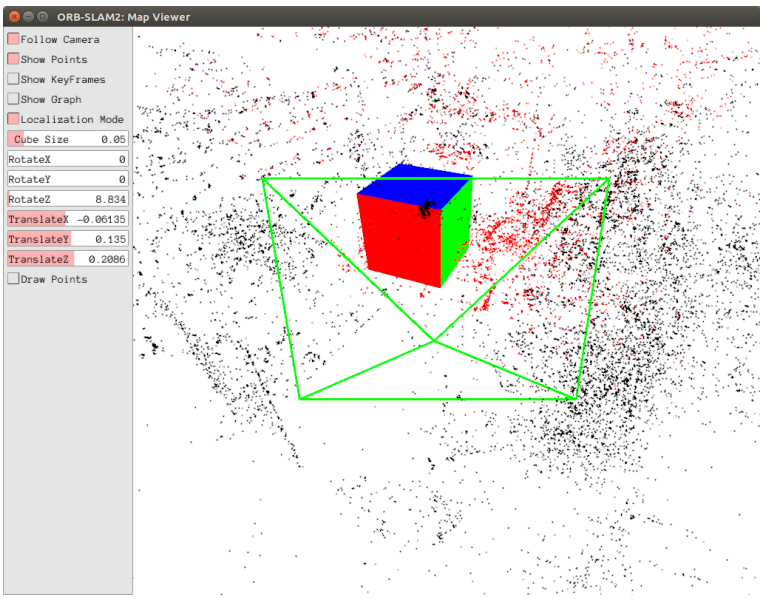


図 28: 環境地図での配置

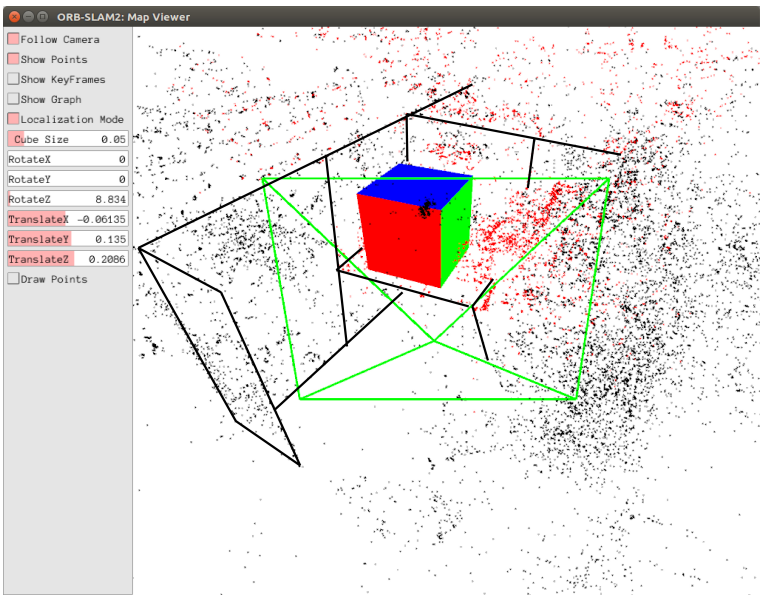


図 29: 補助線を加えた環境地図

4.2 屋外

ステレオカメラを用いたデータセット [23] を使用して屋外の AR 構築を行う。

4.2.1 実行環境

実行環境を以下に示す。

OS	Ubuntu 16.04 LTS
CPU	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz 3.60GHz
RAM	16.0 GB
データセット	KITTI odometry data set (grayscale) 00

環境地図作成の入力はステレオカメラの KITTI データセット 00 で行った。オブジェクトの位置調整完了後の AR 確認のための入力は、ステレオカメラのデータセットから左側の画像のみを用いて単眼カメラによる映像を作成して行った。

(a) 左側

(b) 右側

図 30: ステレオカメラデータセットの最初のフレーム

4.2.2 環境地図作成

KITTI odometry data set (grayscale) 00 の入力を図 31 に、使用して作成した環境地図を図 32 に示す。

図 31: 入力フレーム

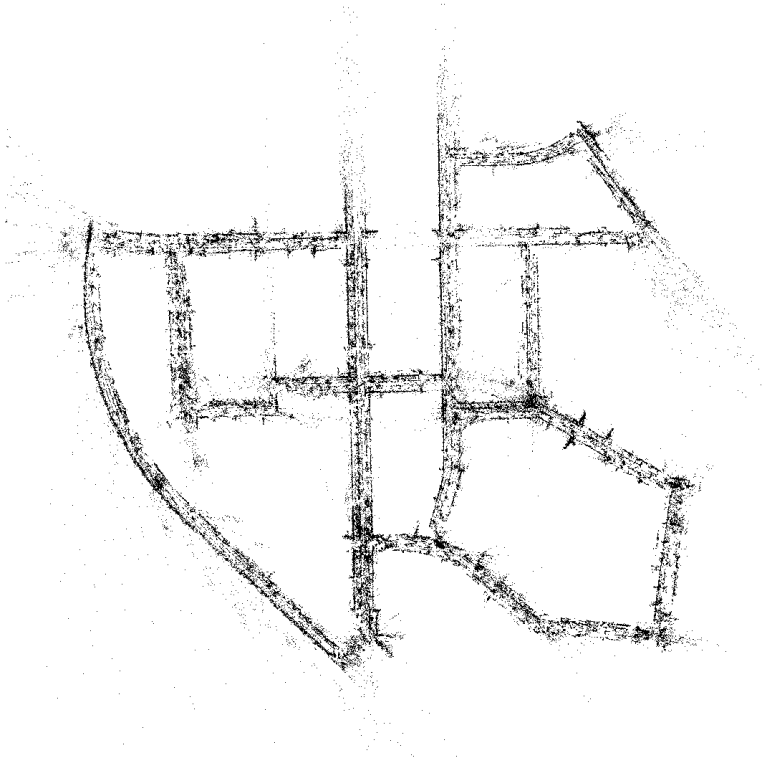


図 32: KITTI odometry data set (grayscale) 00

4.2.3 結果

環境地図での配置を図 34 で、AR ビュワーを図 33 で示す。環境地図と実空間の対応を示すために手動で補助線を加えた画像を図 35 に示す。

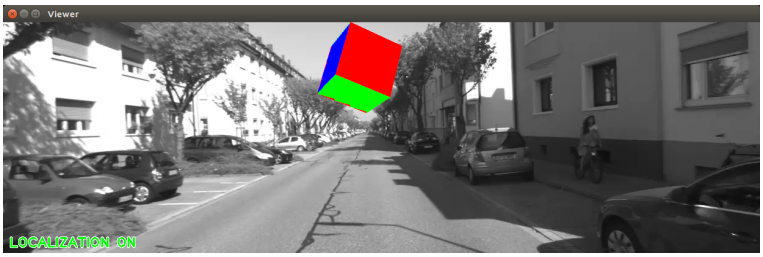


図 33: AR ビューワーでの配置

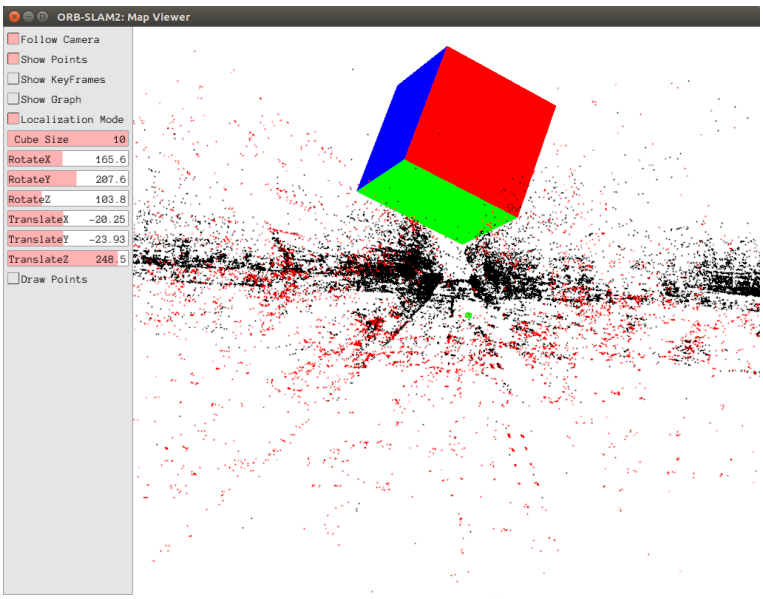


図 34: 環境地図での配置

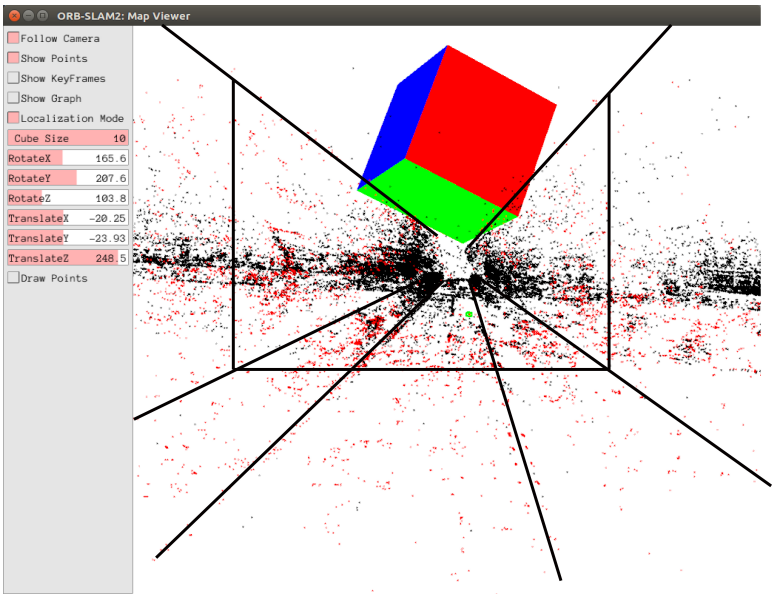


図 35: 補助線を加えた環境地図

4.3 考察

以上の事例より、本システムでは環境地図上に配置したオブジェクトを実空間の同じ場所に表示することができた。最初の環境地図作成時に精度の高い手法を用いることでスケール誤差に悩まされずにオブジェクトを意図した場所に配置ができた。本システムは最初に精度の高い環境地図が作れるかどうかが重要であると考える。

また、AR ビューワーの利用には一切の操作が必要ないことからコンテンツの操作は制作者のみで行われる。そのため、どの様な環境で利用されるのか、どの様に利用されるか、を制限しやすく複雑なコンテンツの開発が可能になると考える。

5　結論

本章では提案手法について述べ、今後の課題と展望を示す。

5.1　まとめ

本研究では環境地図を用いることで 3DCG オブジェクトを意図した位置に配置する手法を提案した。

本手法は、最初に SLAM を使用して環境地図を作成する。環境地図は 3DCG オブジェクト配置の際に空間を把握するためと、AR の際にカメラの位置と姿勢の推定を行うために使用される。次に環境地図を利用して 3DCG オブジェクトを意図した位置に配置する。

事例では実際に意図した場所に 3DCG オブジェクトの配置が可能であること。データセットなどの画像入力を用意することによって自分が現地に赴き、環境地図を作成しなくてもこのシステムを使用して 3DCG オブジェクトの配置が可能であることが確認できた。

5.2　課題と展望

実際にコンテンツを体験するにはスマートフォン等手軽に持ち運びができるデバイスが望ましいが、本システムは強力な CPU でなければリアルタイムで実行することができないため、デスクトップ PC で実装している。

環境地図作成の際、単眼カメラだけでもこのシステムは使用すること可能であるが、より精度の高い環境地図を得るため単眼カメラではなく屋内の事例では RGB-D カメラを、屋外の事例ではステレオカメラのデータセットを使用した。これらは一般的な機材ではなく、本システムの使用条件を厳しくしている。

ORB-SLAM2 で得られる環境地図は点群であるため、入力画像と AR ビューワーを確認しながらでないとそれ単体では環境の材等を把握することはできない。環境地図だけでは正確な配置は困難である。

今後の展望として、現在はオクルージョン表現ができていないため、意図した場所に配置できても本来見えない場所からでも見えてしまう問題がある。環境地図によって壁などの環境内の障害物の把握はできているので、それを利用してオクルージョン表現が可能であると考える。

謝辞

本研究を進めるにあたって宮田 一乗先生と謝　浩然先生には別分野から進学してきた自分に対して様々な助言をしていただきました。修士論文はギリギリまで時間がかかりましたが最後まで面倒を見ていただいたおかげでどうにか形にすることができました。本当に感謝しています、ありがとうございました。

昨年度宮田研究室所属の助教であった浦 正広先生には研究計画書や就職活動においてお世話になりました。どうにか自分が最初からしたかったことを研究にすることができました。

同期の皆は研究の相談から普段の生活まで、さまざまなことで協力できる友人になりました。楽しい院生生活を送れました。ありがとうございました。そしてこれからもよろしく。

最後に、父と母に感謝を。ここまで本当にいろいろなことがあり、その度に助けてもらいました。自分がやりたいと思ったことをやらせてもらいました。悔いのない学生生活でした、本当にありがとうございました。

参考文献

[1] IDC　Japan 株式会社"2018 年第 1 四半期　AR/VR ヘッドセットの国内／世界市場出荷実績と 2022 年までの予測を発表" https://www.idcjapan.co.jp/Press/Current/20180710Apr.html

[2] ARKit https://developer.apple.com/jp/arkit/

[3] ARCore https://developers.google.com/ar/

[4] Unity https://unity3d.com/jp

[5] Unreal Engine 4 https://www.unrealengine.com/ja/what-is-unreal-engine-4

[6] Android Studio https://developer.android.com/studio/?hl=ja

[7] Xcode https://developer.apple.com/jp/xcode/

[8] Vuforia https://unity3d.com/jp/partners/vuforia

[9] vrvca "VR/AR Global Investment Report & Outlook 2018"
https://static1.squarespace.com/static/575e5cd62b8ddeb3fba63f79/t/5aa963d4e2c483ff9d60f3c9/1521050602030/VRVCA_Global+Investment+Report+2018_vF+%28EN%29.pdf

[10] IKEA Place https://m.ikea.com/jp/ja/pages/campaigns2018/ikeaplace/

[11] Google Art & Culture https://artsandculture.google.com/?hl=ja

[12] Pocket Gallery https://www.blog.google/outreach-initiatives/arts-culture/art-gallery-your-pocket-see-vermeers-paintings-augmented-reality/

[13] 池田 直旺, 花立 麻衣子, 樫山 和男, 車谷 麻緒, 吉永 崇, 前田 勇司 "SLAM 技術に基づく空間情報を用いた AR 可視化システムの構築とその適用性の検討", 土木学会論文集 F3(土木情報学),2017

[14] 大寺 賢, 古矢 真之介, 岡田 謙一 “MR を用いた家具に対する部屋環境の可視化”, 研究報告デジタルコンテンツクリエーション（DCC）,1-6,2015

[15] Raúl Mur-Artal and Juan D. Tardós. “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras.”,IEEE TRANSACTIONS ON ROBOTICS, VOL. 33, NO. 5, OCTOBER 2017

[16] Ethan Rublee ; Vincent Rabaud ; Kurt Konolige ; Gary Bradski. “ORB: An efficient alternative to SIFT or SURF’,’,IEEE,2012

[17] Edward Rosten and Tom Drummond,”Machine learning for high-speed corner detection”,ECCV 2006: Computer Vision – ECCV 2006 pp 430-443

[18] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua,”BRIEF: Binary Robust Independent Elementary Features”,ECCV 2010: Computer Vision – ECCV 2010 pp 778-792

[19] 金谷 健一, 菅谷 保之, 金澤 靖 (2016)『3 次元コンピュータビジョン計算ハンドブック』森北出版.

[20] OpenGL https://www.opengl.org/

[21] OpenCV https://opencv.org/

[22] OpenGL 策定委員会, デーブ・シュライナー, メイソン・ウー, ジャッキー・ニーダー, トム・デイビス (2006)『OpenGL プログラミングガイド 原著第 5 版 OpenGL バージョン 2, オフィシャルガイド』松田晃一訳, ピアソン・エデュケーション.

[23] The KITTI Vision Benchmark Suite.http://www.cvlibs.net/datasets/kitti/