東京大学
情報理工学系研究科 電子情報学専攻
修士論文

デジタルツインのための 3D マップとネットワーク型認識

48-206402
青谷和真
Kazuma Aotani

指導教員　落合 秀也 准教授

2021 年 1 月

概要

現実世界を正確に再現し幅広く管理することができるデジタルツインは近年注目を集めている。しかし、デジタルツインの新たな作成には、対象となる建物の 3D データの取得や、詳細なメタデータの付与、さらには広くアクセス可能になることが求められるなど要求事項が多岐にわたる。そこで、3D スキャンにより取得した 3D データに対し、オントロジーを活用してメタデータを付与し、これらの情報を取得する API を作成した。また、デジタルツインの作成後にはその活用が重要となるが、その一つとして屋内でのロボットの走行などが挙げられる。他のセンサを活用し協調的な認識を行うことで効率的な経路計画が期待できるが、点群データは容量が大きく、ネットワークリソースを逼迫させるため、そのまま送信することは難しい。そこで、ネットワーク状況に応じて送信する点群のエリアを動的に変更することで点群のリアルタイムでの送信を試みた。結果として、ネットワーク帯域を狭めてもリアルタイムで点群を送信することに成功した。

Abstract

In recent years, digital twins, which can accurately copy and govern the real world, have gotten a lot of attention. The production of a new digital twin, on the other hand, necessitates a number of conditions, including the collecting of 3D data for the target building, the insertion of detailed metadata, and the requirement to make it widely accessible. Therefore, we made 3D model by using scanning, and add metadata to it with ontology. Moreover, we've designed an API that allows you to get 3D data from it. After developed, it's critical to use the digital twin , and one of the example is operating indoor robots. Using other sensors for cooperative recognition can result in more efficient route planning, but sending point cloud data is problematic as it is. Using extra sensors for cooperative recognition can result in more efficient route planning, but sending point cloud data is problematic since it is huge and puts a burden on network resources. As a result, we tried to send point clouds in real time by dynamically modifying the area of the point cloud to be broadcast based on network conditions. As a result, even though the network bandwidth was halved, we were able to broadcast the point cloud in real time.

Contents

第 1 章　序論 1

1.1 はじめに 1
1.2 目的 1
1.3 構成 2

第 2 章　デジタルツイン 3

2.1 定義 3
2.2 デジタルツインの構築 3
2.3 セマンティクスウェブ技術 3

第 3 章　3D マップの作成 7

3.1 実験 7
3.2 PLATEAU と屋内情報の活用 12
3.3 まとめ 12

第 4 章　3D データを活用した協調的認識 14

4.1 3D データによる協調的な環境認識 14
4.2 点群データの送信 15
4.3 ROS 16
4.4 ROS による自律走行 16

第 5 章　提案手法 18

5.1 協調的な環境認識 18
5.2 データの選択的送信 18
5.3 デジタルツインとの連携 19
5.4 実装 19

第 6 章　実験 21

6.1 実験概要 21
6.2 シミュレーション環境 21
6.3 結果 23

第 7 章　結論 28

7.1 まとめ 28
7.2 今後の課題 28

発表文献と研究活動 29

References 30

第 1 章

序論

1.1 はじめに

近年の IoT 化の進行などにより機器や設備の稼働状況，環境情報などをリアルタイムで収集することが可能となっており，これらの情報の可視化やシミュレーションを行うディジタルツインへの注目が高まっている．ディジタルツインは特に製造業における生産性の向上やスマートシティの実現といった分野での活用が期待される [1] [2].

ディジタルツインアプリケーションの作成には，対象の建物の 3 次元形状及びそれに付随する属性情報が用いられる．そこで，BIM(Building Information Modeling) と呼ばれる，建物の形状情報と属性情報を併せ持った建築情報モデルの使用が有効であるが，既存の建物には BIM が存在しない場合が多い．そのため，3D スキャンを用いて建物の形状情報を取得し，それに対し属性情報を付加し，BIM 化するというプロセスが必要となる．

ある建物のデータを使用したディジタルツインのユースケースは多数存在し，これらのデータは多様なドメインのデータと相互に参照し合うことでより価値を発揮する．そのため，建物に関するメタデータは一つの固有のシステムの中に取り組むのではなく，オープンデータとして汎用的に扱える必要がある．そこで，属性情報の付加の際には，オントロジーに従って記述する事で，建物に関する情報から LOD（Linked Open Data）を作成することが望ましい.

また，デジタルツインの作成後には，その活用として，例えば室内でロボットの操作する，ということが挙げられる．経路決定の際の環境認識には，Lidar などのセンサによる 3D データが用いられるが，こうしたデータは容量が大きくそのまま送信することは効率が悪く避けられがちである．一方で生のデータの送信は認識の精度を向上させる．リアルタイムでの操作を行うためには，LIDAR センサなどから取得した点群データとデジタルツインを照らし合わせ，適切な環境認識を行う必要があり，そのためには点群の効果的な送信が求められる.

1.2 目的

本論文の目的は 2 点ある．1 点目は，デジタルツインに求められる要件を実際に作成することを通して検討することである．2 点目は，デジタルツインの活用例として，室内でロボットを協調的に操作するために点群をリアルタイムに送信する手法について検討することである.

1.3 構成

本論文は次のように構成される．第 2 章では，デジタルツインの概念やそれを構成する技術，応用例などを紹介する．第 3 章では，現実の建物を対象として実際に 3D マップを作成した過程について述べる第 4 章では，ロボットの協調的な経路計画に関する技術について述べる．第 5 章では，提案手法を説明する．第 6 章では，シミュレーションにより提案手法が有効であることを示した．

第 2 章

デジタルツイン

2.1 定義

デジタルツインは，狭義では原子レベルから幾何学的なレベルまで，現実上の物体を完全に記述する一連の情報であると定義される [3]．これらは設計，生産，運用，保守に至るまでの製品のライフサイクルをシミュレーションすることに用いられる．一方で広義にはシステムの状態やプロセスのシミュレート，監視，制御などができる統合的システムであると定義されている．このデジタルツインは，物理的空間，デジタル空間，そして 2 つの空間をつなぐ情報処理層の 3 つから構成される．BIM 等から抽出した形状データや実空間から収集した時系列データを用いて，サイバー空間でシミュレーションやデータの可視化などを行うアプリケーションは「デジタルツイン」と呼ばれる．デジタルツインは元々製造業で先行してきた概念であり，シミュレーションによって現実と同様のプロダクトをサイバー空間上に作り，想定されるトラブルの検証などに利用されていた. 現在はデジタルツインは製造業に加え，ヘルスケア，スマートシティといった分野での活用が盛んである [4].

2.2 デジタルツインの構築

2.3 セマンティクスウェブ技術

2.3.1 オントロジ

オントロジは「概念 (Concept) と関係性 (Relation) の組を持ち，形式化されているためにコンピュータによる推 論 (Reasoning) が可能であるもので，特定のドメインに対して有用にデザインされている」と定義されており，以下のような特徴を持つ [5]．オントロジはセマンティックウェブにおいて中心的な役割を果たす技術である. 情報の一元的な表現が可能になり，ウェブ上でデータを共有したり処理したりすることが容易になる. また，オントロジの持つ関係性などのセマンティクスは，ウェブサイトの組織化やナビゲーションに有用であるため，検索の精度を向上させることができる. オントロジやデータモデルを記述するための技術仕様としては，RDF がある．

RDF/LOD

RDF (Resource Description Framework) は Web 上にあるリソースを記述するためのモデルで, W3C (World Wide Web Consortium) によって規格化された.RDF では，主語，述語，目的語の 3 つの要素でリソースの関係を表現する．これらのトリプルの集合を RDF のグラフと呼び，リソースの識別には URI (Uniform Resource Identifier) を用いる.RDF のデータを格納する専用データベースをトリプルストア

(Triplestore) と呼び，RDF のクエリ言語 SPARQL を用いてクエリを発行することができる.RDF を記述する方法として XML 形式を採用した RDF/XML や JSON 形式を採用した JSON-LD，簡潔で比較的可読性が高い Turtle，Turtle やその前身の Notation3 の派生であり簡潔性を排除 する代わりに解析性を向上させた N-Triples などがある.SPARQL クエリには Turtle 記法を用いることができる．

Tim Berners-Lee は Web 上にデータを公開する方針として以下の 4 条件をみたす Linked Data を提唱した. [6]

1. モノやコトに対する識別子に URI を使用する
2. 誰もが参照できるよう HTTP の URI を使用する
3. URI を参照した際には，RDF や SPARQL などを用いて有用な情報が提供される
4. 多くの情報が得られるよう外部へのリンクを含める

LOD(Linked Open Data) とはオープンライセンスのもとで公開され，無料での再利用が可能な Linked Data のことである.LOD および Linked Data は RDF を用いて作成することが可能であり，特に RDF データを，外部リンクを含みオープンライセンスのもとで公開することは LOD の作成を意味する．RDF データや LOD は，Tim Berners-Lee が提案するオープンデータのための 5 段階評価の枠組み “5-star Open Data”において，4 段階目，5 段階目と高い評価に位置づけられる. 再利用性の高さや処理に特定のアプリケーションを必要としないこと，URI を用いてリソースを一意に識別できることが利点に挙げられ，さらに，LOD では外部と参照し合うことでデータの価値が高まる．

2.3.2 3D モデルのオープンデータ化

街情報などの３ D データを公開し，利活用する例が国内外で見られる．スマートシティの実現など，3D データをより高度なデータマネジメントに応用するには，ジオメトリ情報だけでなく建物の名称や高さといった多様なメタデータを情報として保有している必要がある．

2.3.3 CityGML

街情報などの３ D データを公開し，利活用する例が国内外で見られる．スマートシティの実現など，3D データをより高度なデータマネジメントに応用するには，ジオメトリ情報だけでなく建物の名称や高さといった多様なメタデータを情報として保有している必要がある．このような要件を踏まえた日本全国の 3D 都市モデルの整備・活用・オープンデータ化プロジェクトとして，国土交通省が主導する PLATEU と呼ばれるものがある．このプロジェクトでは東京都 23 区をはじめとした複数都市の３ D モデルを公開しており，人流計測や混雑状況の可視化に用いられている．これらの都市モデルは CityGML という形式のファイルで表現されており，ジオメトリの情報に加え，構造物の属性情報が XML 形式で組み込まれている．また，CityGML のもう一つの特徴としてモデルの詳細度に応じてレベルが分かれており，これらを一元管理することができる [7]. 図 2.1 に CityGML ビューワーにより 3D マップを表示させている画面を示す.

現在策定中の CityGML3.0 では，[8] では，CityGML の見直しが行われており，時系列データなど，動的な情報を記述できるようにするなど，新たな LOD の策定などに取り組んでいる

2.3.4 BIM

建物の形状情報と属性情報を持ったデータ形式として Building Information Modeling（BIM）が存在する．BIM は「建物の物理的・機能的特性をデジタルで表現したものであり，ライフサイクルで発生する何らかの意思決定のために，信頼性の高い情報を提供する共有知識資源である」と定義されている [9]. より具



Figure2.1: CESIUM により表示した CityGML

体的には，建物の 3 次元形状の情報だけでなく，部屋の名称や面積，材料といった属性情報を付加した建築情報モデルである．BIM は現在は主に建設プロセスの効率化や安全衛生面の管理，施設の管理といった分野に利用されている [10] [11]．特に近年は IoT との連携によりデジタルツインを実現し，より多くの分野での活用が期待されている [12].

一般に BIM の作成には Autodesk，GRAPHISOFT などが開発した専用のオーサリングツールを利用する．3D モデルのデータから BIM を作成する際は建物内のオブジェクトに対し手動でアノテーションしていくという手法を取ることが多い．しかし，こうした手動でのアノテーションはコストが大きいということや，ヒューマンエラーが起こり得るといった問題があるため，3 次元形状のデータから自動で BIM の作成をすることが求められる [13].

2.3.5 IFC ファイルから属性情報，構造情報の抽出

建築オブジェクトとそのプロパティを読み込む，及び書き込むための国際標準として，Industry Foundation Classes(IFC) ファイルが存在する．前節で述べた BIM もこの IFC 形式で出力される．BIM をデジタルツインアプリケーションに活用するためには建築オブジェクトの形状情報とメタデータを取得することが必要となる．IFC ファイルからこれらを抽出する API の作成を行なっている．

メタデータについては IFC 形式から BOT に従った記法に変換してグラフ DB に格納，形状情報については gltf や obj 形式に変換し PostGIS に格納することで，それぞれデジタルツインアプリケーションに対して提供しやすいように保持している．この一連の流れを図 2.2 に示す.

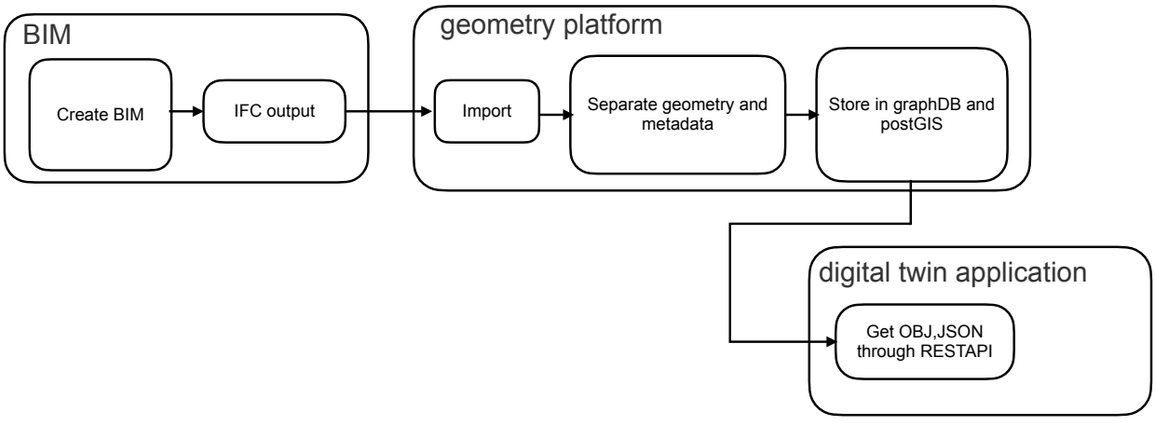


Figure2.2: System constitution for extracting metadata and 3D data from BIM

第 3 章

3D マップの作成

3.1 実験

デジタルツインアプリケーションの構築には，対象となる物体及び，それをデジタル空間上で再現するための 3D データ，メタデータが必要となる．3D データの取得には 2 章で述べたように BIM を用いることが多いが，BIM の作成には専門的な知識や時間，労力などのコストが要求される．新しく建設される建物に関しては建設プロセスの管理のために BIM が用意されていることが多いが，既存の建物の多くは BIM が用意されていないことがほとんどである．そこで，3D スキャンによる 3D データの取得をし，それに対してオントロジーに従った記法によりメタデータの付与をした．さらに，アプリケーションに対して情報を渡すための API を作成した．本章では，この各ステップに対する詳細を記す．

3.1.1 データ収集

３ D データの取得に関しては，フォトグラメトリックな方法，LIDAR などのセンサを用いた手法などがある．今回は３ D スキャンに MatterPort 社 [14] の提供するサービスを利用した．撮影に用いた機材は MatterPortPro2 で，赤外線による形状情報の取得，4K カメラによるテクスチャ情報の取得が可能になる．形状情報は点群，OBJ ファイルとして取得することができる．

撮影場所は東京大学本郷キャンパスの工学部 2 号館，エービー白河工業事業所，NTTCOM スマートシティラボの 3 箇所である．

東京大学キャンパスマップ

Matterport を用い, 東京大学工学部二号館の B1F から 13F までを撮影した．ただし,4F,10F 以外の階では階段からエレベーターホールまでの短い区間のみを撮影した．4F では, それらに加えて講義室や実験室を撮影し,10F では, 研究室を撮影した．4F の撮影には約 5 時間 (撮影回数:350 回),10F の撮影には約 2 時間 (撮影回数:120 回) を要した．撮影した講義室のデータの一部を図 3.1 に示す．これらの 3D モデルを用いて,3D マップを構築していく. 撮影に使用した Matterport Pro2 というカメラでは, 一度に 3 5m の範囲を撮影できるが, 撮影範囲が広くなると，徐々に誤差が蓄積されていき, 構造的に整合性が取れなくなる場合がある．図 3.1 では本来閉路となるはずの廊下が，ループを構築できないという問題が発生している．

エービー白河事業所マップ

BIM の活用分野の一つにファクトリーオートメーションが挙げられる．本節では, 工場内の情報の可視化等に活用するため作成した 3D モデルについて説明する．撮影を行なった場所はエービー白河事業所で, ここで



Figure3.1: 工学部 2 号館の点群データ．閉路の始点と終点の位置がずれている．

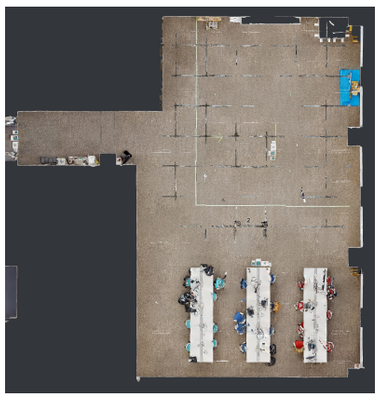


Figure3.2: NTT コミュニケーションのスマートラボの点群データ

はプラスチック金型制作などが行われている．工場内には大型の機械などが多く，オクルージョンが多く発生したため，撮影には多くの地点を要した．撮影に要した時間は 4 時間半程で，139 地点での撮影情報を用いた．に工場を上から見た画像を示す．また，工場内には環境センサや Lidar センサが複数箇所に設置されており，温度，湿度，照度，騒音や機械の電力といった情報を取得することが出来る．

NTTCOM スマートシティラボのマップ

東大グリーン ICT プロジェクト（GUTP）[*1] にてディジタルツインアプリケーション開発のためのワークフローの検討および実証実験が進められている．ディジタルツインアプリケーション開発の最初の段階として，今回 NTTCOM スマートシティラボにて撮影を行なった．この撮影では，Matterport Pro2 に加え，レーザーによるスキャンを行う BLK360 も用いられた．両者で撮影をしえられた点群の比較では，主に密度に差があり，物体の形状などは BLK360 の方が精度が上であった．例えば，床に貼られたテープが，BLK360 では一直線となっていたが，Matterport Pro2 では途中で途切れるなどしていた．この様子を図 3.3 に示す． 一方で，今回撮影を行なった規模では，部屋全体の精度としてはそれほど大きな差は出なかった．この撮影に要した時間は 1 時間ほどで，撮影地点は 48 箇所であった．

[*1] https://www.gutp.jp/

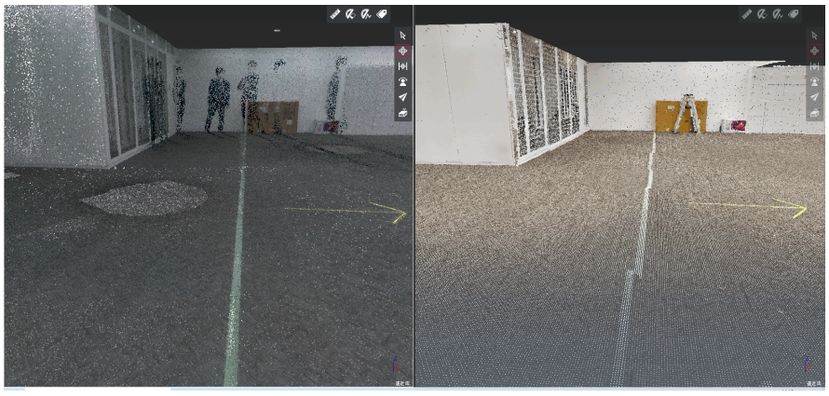


Figure3.3: Matterportpro2 と BLK360 で撮影した点群データの比較．左側が BLK360 によるもので右側が Matterportpro2 によるもの．点の密度や精度に差があり，Matterportpro2 によるものは床の上の線に途切れが生じるなどした．

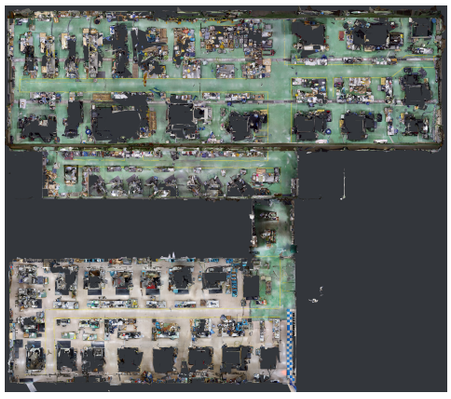


Figure3.4: エービー白河工場の点群データ

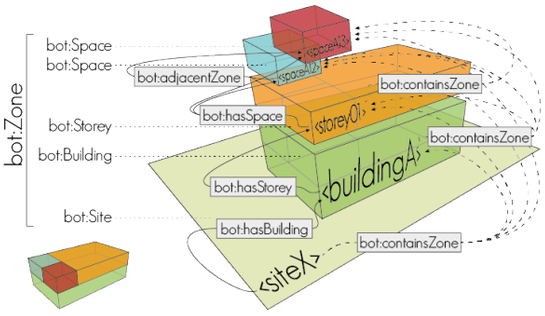


Figure3.5: 区域内のクラス間の関係性 [15]

3.1.2 BOT によるメタデータ付与

建物に関する情報として必要なものとして，構造情報が挙げられる．これを記述するために，オントロジーとして BOT [15] を利用している．

BOT では，図 3.5 に示すように，区域から始まり，階層的に建物，階数，空間を記述する．また，各部屋に対しては，その部屋の名称，固有の ID，設置されている場合はセンサの ID がメタデータとして付与されている．

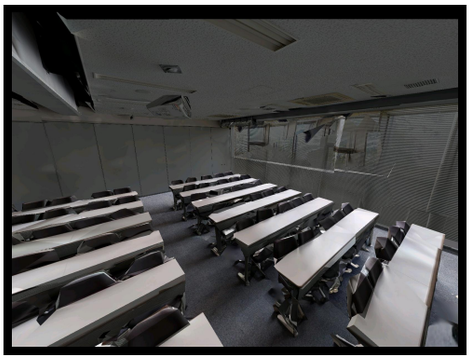


Figure3.6: 本節で述べた API から取得した自由視点からの二次元画像

3.1.3 API 設計

3.1.1 で収集したデータを用いて，建物の形状情報，属性情報をアプリケーションに対して提供する API を，Python の FastAPI で実装した．API の構成としては，建物に関するメタデータ，3D データ，時系列データを与えるものを用意している．メタデータについては，引数を指定しない場合，その建物の名称，所在地，その建物に含まれる階数や部屋についての情報を JSON 形式で取得することができる．部屋の ID をパラメータとして指定した場合には，その部屋内の情報として，部屋固有の ID，部屋の名称，3D データの格納先のパス，センサなどが設置されている場合はそのセンサの名称と MQTT ブローカーの IP アドレスを取得できる．また，外部アプリケーションとの連携等を考慮し，3D データ上の任意の視点からの画像を生成し提供するということを行なっている．この機能の実装例がなかったため，今回 2 通りの実装を行い，その評価を行なった．

3D マップから 2D 画像への変換

前節で記述したメタデータ及び部屋の画像を取得することのできる API を Docker 内に FastAPI を用いて作成した．3D マップからの画像の取得は python を用いた手法と Unity による手法の 2 種類の手法を用いて実装を行なった．Python を用いた実装では,VM 上に Xvfb による仮想ディスプレイを作成し,pygame を用いて obj ファイルを読み込み, 指定された視点からの仮想ディスプレイのキャプチャを行なっている．視点の向き, 視点の位置, 部屋は POST リクエストにより指定している．

Unity を用いた実装では,FastAPI を通した POST リクエストから Unity へアクセスする．POST リクエストには, 上部から撮影するか斜部から撮影するかといった情報やどの部屋を撮影するかといった情報を含んでいる．POST リクエストを受け取った後は,Xvfb を用いて仮想ディスプレイを作成し, ビルド済みの Unity アプリケーションデータを用いて 3D モデルを配置し, キャプチャを行い画像ファイルを生成する．最後にその画像ファイルを Base64 形式でユーザに返却するといった構成になっている．図 3.6 に，この API を通じて取得した部屋の画像を示す．

API 評価

Python で作成した 2 次元画像と Unity で作成した 2 次元画像を図 3.7 に示す．Python による実装の方が部屋の様子が鮮明に写っているが，Unity の場合は約 4.3 秒，Python の場合は約 13.6 秒と応答時間に大きく差が出た．上記で設計した API のうち，特に応答時間が長くサーバーに負荷のかかる 2D 画像の生成機能について，応答時間を評価した．一つの部屋の obj ファイルは 4〜16MB 程度であり，これを読み込み画

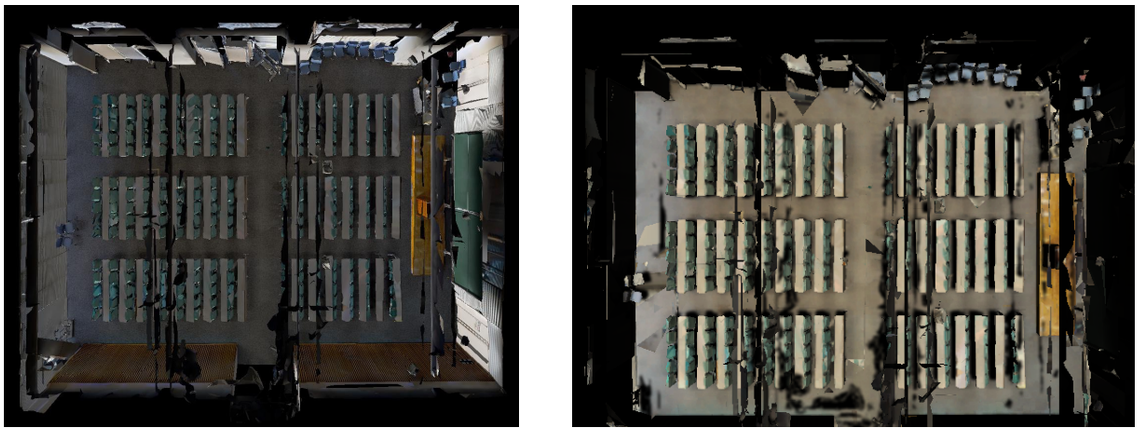


Figure3.7: Python（左）,Unity（右）によって生成した部屋の 2 次元画像

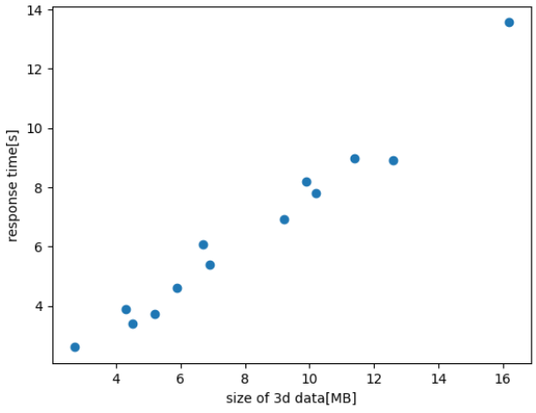


Figure3.8: 3D データのファイルサイズと API の応答時間の関係をプロットした散布図

像を生成するまでの時間はおおよそ 3D データのサイズに比例した．図 3.8 にファイルサイズと応答時間の関係を表したグラフを示す．応答時間は Python により実装した API で計測し，10 回の平均を取った．また，ネットワーク環境等による影響をなくすためローカルホスト上にサーバーを立ててアクセスを行なった．

真上から見た画像や，横から見た画像など，利用頻度が高いと思われる画像について，キャッシュをしておくことで応答時間は $1.5 * 10^{-2}$ 秒程度まで削減することができた．また，同時に 100 件のリクエストを投げるとサーバーが処理落ちするため，RabbitMQ による非同期処理を実装している．

特に Python での実装に関して，実用的な API とするためにはより応答時間を短縮することが求められる．応答時間は主に 3D データのサイズに比例するため，例えばポリゴン数を削減することは有効である．一方でポリゴン数を削減しすぎると画像の詳細どの低下に繋がるため，最適なポリゴン数の模索が求められる．また，今回用意した 3 D データは 3 D スキャンにより取得してきたものであり，頂点の数が BIM などによる 3 D モデルよりも多くなってしまう傾向がある．そこで，点群から自動で BIM を作成する際に用いられる平面検出など [16] を用いることで，3D モデルをより簡単な形に置き換えることでファイルサイズの削減につながることが予想される．

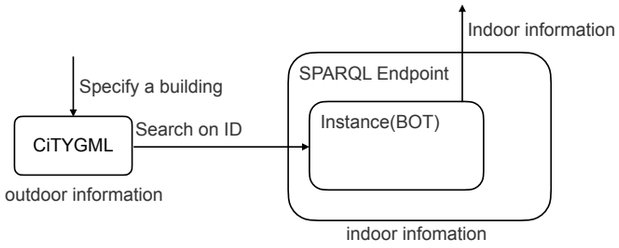


Figure3.9: 屋内情報と CItyGML を組み合わせたビューワーの構成図

3.2 PLATEAU と屋内情報の活用

PLATEAU で公開されているデータは街の外観を捉えた 3D モデル及びそれに付随するメタデータである．メタデータには，住所，高さ，面積などに加え浸水ランクなどのデータが含まれており，XML 形式で記述されている．一方で，前章で作成した 3D マップは建物内部の 3D データとメタデータを含んでいる．これらの屋内外情報を繋ぐことによって，より詳細度の高い 3D マップの作成が可能になる．また，人流データなど，これまで屋外での情報と屋内での情報が分断されていたものをつなぐことで，データの利活用が可能になる．

3.2.1 実装

CityGML 形式で与えられた東京都 23 区の 3D マップと前章で用意した工学部 2 号館の 3D データを用いることで屋内外の情報を繋いだビューワーを作成した．ビューワーのシステム構成を図 3.9 に示す．CityGML の読み込み及び可視化には CESIUM という 3D ビューワーエンジンを活用しており，建物情報及びメタデータの取得を行なっている．街全体のビューワーから任意の建物を指定すると，その建物固有の ID から，SPARQL エンドポイントに対して検索を行い，オントロジーで記述されたインスタンスを特定することができる．

3.2.2 課題点

屋内に関する情報はオントロジーに従った記述により RDF 形式で記述されているが，屋外情報に関しては CityGML で与えられており，それぞれの情報を一括で管理することが現状では出来ていない．また，CityGML の形式で与えられたファイルを読み込むには，CESIUM のような専用のツールを用いることが必要とされるため，自由な開発を阻害してしまっている現状にある．そのため，**??**節で触れたような，メタデータや形状データを取り出し提供する API などが重要になってくると考えられる．

3.3 まとめ

本稿では，デジタルツインアプリケーション作成の枠組みとして，建造物の 3D データ取得からメタデータの付与，建物に関する情報をアプリケーションに提供する API の作成を行なった．また，この屋内の情報と，オープンデータとして提供されている 3D マップを組み合わせることで，屋内外のデータを繋いだより詳細な 3D マップビューワーを作成した．

今後の課題としては，今回作成した API の応答時間をより早くするための 3D データの処理や，この API を利用して現実世界へのフィードバックまでを含めたデジタルツインアプリケーションの作成することで実用

性の評価を行うことや，3D スキャンを通じて獲得した 3D データと BIM を同様に扱うためのデータ処理などが挙げられる．また，CityGML のデータをより扱いやすくするための枠組みの導入などが必要となる．

第 4 章

3D データを活用した協調的認識

本章では，3D データを用いた協調的な環境認識，また，3D データを効率的に送信するための圧縮技術，ストリーミング技術について紹介を行う．

4.1　3D データによる協調的な環境認識

点群データはレーザースキャンや LIDAR センサによって獲得されることが一般的である．これらのスキャンによって獲得される点群データの容量は膨大であり，ネットワーク帯域を逼迫させ得るため，効率よく送信することが必要となる．そのための代表的な手法として，点群圧縮が挙げられる．主に以下の 3 つの手法が挙げられる．

自動運転において，各車両は，カメラや LiDAR などのセンサによる情報を組み合わせることで周囲の環境認識を行なっている．また，これらのデータは，車両間や，路側機と共有することでより安全な運転を実現させることができる．その際，センサデータの共有は，データの処理レベルによって以下の 3 つに分類される．

- 生データベース

前処理のされていない生のデータを共有する

- 特徴量ベース

生データから特定の特徴を抽出し，それらを送信する

- オブジェクトベース

各センサから得られたデータから環境認識を行い，障害物のリストを生成し，それらを送信する

これらの 3 つの内，生データの送信はデータ容量が大きく，ネットワークリソースを逼迫させるため避けられる傾向にある．一方で，生の点群データを送信することは，例えば各センサ単体では認識できなかったオブジェクトを他のセンサからのデータと統合することで認識できるようになる，といったメリットがある．

4.1.1　特徴量ベース

Feature based Cooperative perception(F-Cooper) [17] では，自車のセンサに加え周囲の車のセンサのデータを融合させることで認識精度の向上を行なった．生のデータを送信するのではなく，特徴マップを互いに共有することで送信データの削減をしている．

4.1.2 オブジェクトベース

オブジェクトベースの手法では，各センサーで取得された三次元データから物体検出を行い，検出された物体のリストのみを送信するという方法である．物体のバウンディングボックスの位置のみを送信するため，データ送信の効率が良くなるというメリットがある．

4.1.3 生データベース

[18] では，複数のセンサによる点群データを融合する早期融合方式と，複数のセンサから独立して推定された境界ボックスを融合する後期融合方式の 2 種類の方式の性能評価を行なった．T 字路とラウンドアバウトの 2 つのシチュエーションで実験を行い，早期融合方式では，通信帯域幅をより多く使用しながらも，後期融合方式と比べてを大きく上回ることが示されている．また，単一のセンサのみを用いた場合では 30% であったのに対し，協調認識では 95% 以上の対象物が認識できている．[19] では，生の点群データと動的オブジェクト，速度ベクトルを選択的に送信している．[20] では，送信データを点群，レーザースキャンによるデータなどを使用し，それぞれの送信の遅延を通信手段ごとに計測している．

4.2 点群データの送信

生の点群データは容量が大きく，ネットワークの帯域幅を逼迫させる恐れがあるため上記のように何らかの処理を加え別の形式に変換してから送信されることが多い [20]．しかし，より低次元の情報を活用することで物体検知の精度向上が見られた例などもある．そのため，点群をリアルタイムに送信すことにも意味は大きく，点群の圧縮なども盛んに研究されている．点群の圧縮ではオクトリー構造を活用した可逆的な圧縮や，機械学習を用いた圧縮などが行われている．点群は探索木をもとに圧縮する方法，2 次元に投影する方法，3 次元のまま処理をする方法の 3 種類に大別されている [21]．しかし，データの欠損が生じることや，3 次元データをそのまま扱うことの難しさなどが報告されている．

4.2.1 動的な点群のストリーミング

このように 3 次元データをリアルタイムに送信することは非常に困難あるが，3 次元データのリアルタイムなストリーミングが求められる例として，高い没入感が求められる VR のアプリケーションなどが挙げられる．これらを再生するデバイスはモバイル機器であり，制限された帯域幅でいかに高画質の点群を効率的に送信するかが課題となる．センサなどの低次元な CPU でリアルタイムで点群を圧縮することは難しく，上記の点群の圧縮に加えて様々な送信データの削減が行われてる．その一つが動的に送信する点群データを変更することである．

[22] では，生の点群データを複数のレベルの密度でサブサンプリングしたものをサーバーに用意し，クライアント側のデバイスからの HTTP リクエストに従って最適な密度の点群データを送信するアーキテクチャが考案されている．人間の視覚特性に基づいたクライアントのデバイスが HTTP リクエスト，レスポンス通信により密度の異なる点群データを動的に受信することが可能になっている．

これらの多くの動的ストリーミングの手法に共通しているのはクライアントの要求に応じ，点群の密度を動的に変更していること，3 次元形状の中でも重要度の高い箇所とそうではない箇所に分け，重要度に応じて点群の密度を変更しているということである [23] [24].



Figure4.1: ROS におけるノードとトピックの関係

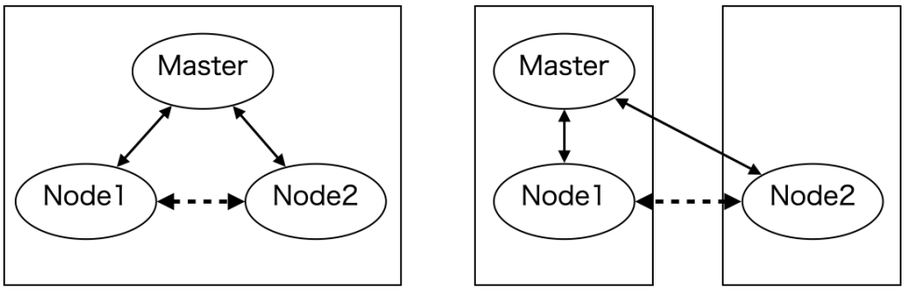


Figure4.2: 複数の PC を跨いだ ROS の実行

4.3 ROS

ロボット開発のためのオープンソースのミドルウェアフレームワークとして，ROS [25] と呼ばれるものがある．ROS は，オープンソースのミ ドルウェアフレームワークで，Linux OS 上で動作する．ROS の実装の基本的な概念として，マスター，ノード，メッセージ，トピック，サービス がある. マスターは，ROS のノード同士をつなげる役割を果たす。ノードは，計算を行うプロセスである.ROS のノード一つ一つは単純な処理を行い，複数のノードを組み合わせることでロボット全体の複雑な処理を実現している．ノード同士はパブリッシュ/サブスクライブ・モデルのメッセージのやりとりを行う. ノードとトピックの関係は図 4.1 のように表される. この図は，ノード 1 がトピック 1 にメッセージをパブリッシュし，ノード 2 がトピック 1 をサブスクライブするこ とで，ノード 1 がパブリッシュしたメッセージを受け取ることを示す．また，図 4.1 のように，ノードはまず最初にマスターに接続して，そのノードがパブ リッシュやサブスクライブしているノードの情報を得る. そして，その後，ノード同士で通信を始める. このような特徴から，図 4.2 のように，ROS は複数 PC 間にまたがって実行することができる．また，ROS には可視化用の rviz やシミュレーション用の gazebo が用意されている．

4.4 ROS による自律走行

ROS では自律走行のためのパッケージとして navigation が用意されている．図 4.3 に navigation の構成を示す.

4.4.1 自己位置推定

ROS の navigation パッケージにおいては，自己位置推定は amcl と呼ばれるノードで行われる．amcl では，ロボット位置の確率分布を，サンプリングした複数のロボットのパーティクルにより近似する．逐次的

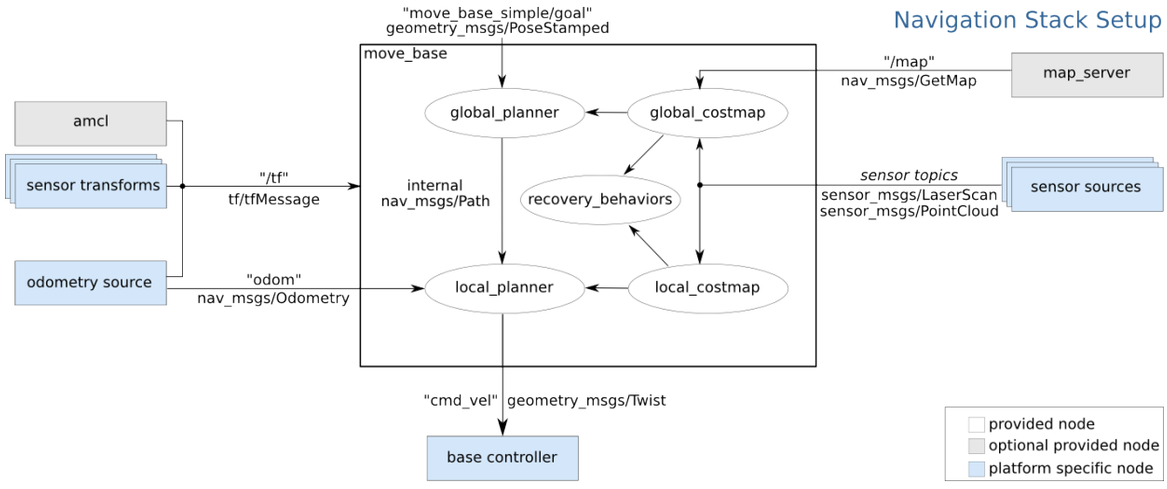


Figure4.3: ROS における navigaton の構成 [25]

に時刻 t における事後確率を更新していくことによって正しい位置に近づいていく．

4.4.2 経路計画

ROS の navigation パッケージに含まれるノードの move_base では，経路決定に Global DWA が用いられる．大域的なパスを求め，そこから局所的なパスが決定される．経路計画はレーザースキャナなどの入力情報をもとに作成される占有格子地図をもとに行われる．また，占有格子地図から，ロボットと障害物の距離に応じて重みづけをしたローカル，グローバルの 2 種類の二次元コストマップが作成される．これらの情報から，大域的なパスはダイクストラ法により最短かつ障害物を避けられるように決定される．

局所的なパスに関しては，ローカルコストマップを元に，ロボットの速度，加速度や姿勢を考慮しつつ障害物を避けるようなパスが求められる．

第 5 章

提案手法

本研究では，センサ群のデータを活用し，室内において協調的な環境認識を行い，ロボットを操作するための手法を提案する．

5.1 協調的な環境認識

自動運転の分野では，協調的な認識が一般である．一方で，ロボットの移動の場合，ロボット自体に装着された Lidar センサやカメラの情報から SLAM 等を活用して運行させることが多い．しかし，デジタルツインという状況下では，建物自体の詳細な 3D マップやその建物内に設置されたセンサの情報などが全て既知の情報として保持していることが期待される．そこで，ロボットに設置されたセンサだけではなく，部屋自体に設置されたセンサ群からの情報を活用して協調的に環境認識を行いロボットの経路計画を行うことを提案する．このようなネットワーク型の認識を行うことには次のようなメリットが考えられる．それは，より広域の環境認識を行うことができるため，ロバスト性の高い走行が可能になる，ということである．ロボット自体のセンサだけでは死角が生じやすく，そこから飛び出してきた障害物と衝突するということが想定される．しかし，死角となるエリアを部屋に設置したセンサからの情報を活用することで事前にその位置の障害物を検知し，避けるように動くことができる．

5.2 データの選択的送信

3D データをそのまま送信するだけではネットワークを逼迫させ，リアルタイムでの操作が困難となる．そこで，ロボットの経路決定に特に重要となるエリアの点群データを優先的に送信することで，動的にネットワークリソースを管理することができる．点群のデータ量をさくげんするためには 4 章で述べたように，点群の圧縮などが有効である．一方で，点群の圧縮の場合では，エンコーディングやデコーディングにかかる時間や，データの欠損などの点において課題が残る．そこで，本論文では送信する点群データの容量を削減するために，送信データの範囲を限定することを提案する．

4 章で述べたように，ROS における経路計画では，マップから計算された大域的なパスと，ロボット周辺の環境から計算された局所的なパスが用いられる．室内全域のデータをセンサで取得することができても，コストマップの範囲外のエリアのデータの重要度はコストマップ内のものと比べると劣る．また，部屋自体に設置されているセンサは有線で高速にサーバー側へ送信することができるが，部屋を動くロボットにはより帯域の小さな無線でデータのやりとりをする必要がある．そのため，常にネットワークリソースを監視し，トラフィックの管理をすることが求められる．そこで，本研究では，現在のネットワークの帯域をパラメータとし，センサから得られたデータの範囲を限定するシステムを実装する．

5.3 デジタルツインとの連携

2 章でデジタルツインの構築を行なったが，室内の場合環境が時間とともに変化することが予想される．例えば，倉庫内において荷物の出し入れを行うような環境では，荷物を積み上げることでロボットが通ることのできる経路が事前に作成したマップと異なってしまうことが起こりうる．そのため，デジタルツイン内でロボットを走行させる場合においても，リアルタイムに環境認識を行うことが必要になる．つまり，ロボット側に未知な障害物が部屋に存在し，経路を動的に変更することが求められる場合が想定される．そのような場合には一度作成した地図を更新する必要があり，SLAM を行う時など，ロボット側がより詳細な点群 3D データを必要とする事態が想定される．

5.4 実装

ROS において実装したシステムの概要を図 6.1 に示す．ロボットの経路計画及び自己位置推定には 4 章で述べた ROS の navigation パッケージを利用した．各ノードの説明を以下に示す．なお，navigation パッケージと共通するノード等は省略している．

Sensors ノードからは室内の三次元形状を取得し，点群データ (sensor_msgs::PointCloud2 形式) として送信している．また，センサはロボットの死角を補い，室内全域の様子を取得できるように複数設置されている．

network_state では，2 台の PC 間でのネットワーク帯域を監視するノードである．iperf コマンドを用いて，PC1 をサーバー，PC2 をクライアントとしてネットワーク帯域を測定し，その結果を/network_state として送信する．このノードは 10 秒に 1 回呼び出される．

area_filter ではネットワーク状況に応じて送信する点群データの範囲を動的に変更して切り取る処理を行う．具体的には，amcl ノードから送信されたロボットの現在地を中心として，ネットワーク帯域の値から求めた長さを縦及び横とする直方体の範囲内にある点群データを選択して送信する．

voxel_filter では，area_filter から送信された点群データを間引くことによって送信データの容量を小さくしている．このデータを move_base の local_costmap,global_costmap に送信することで，コストマップの更新をしている．

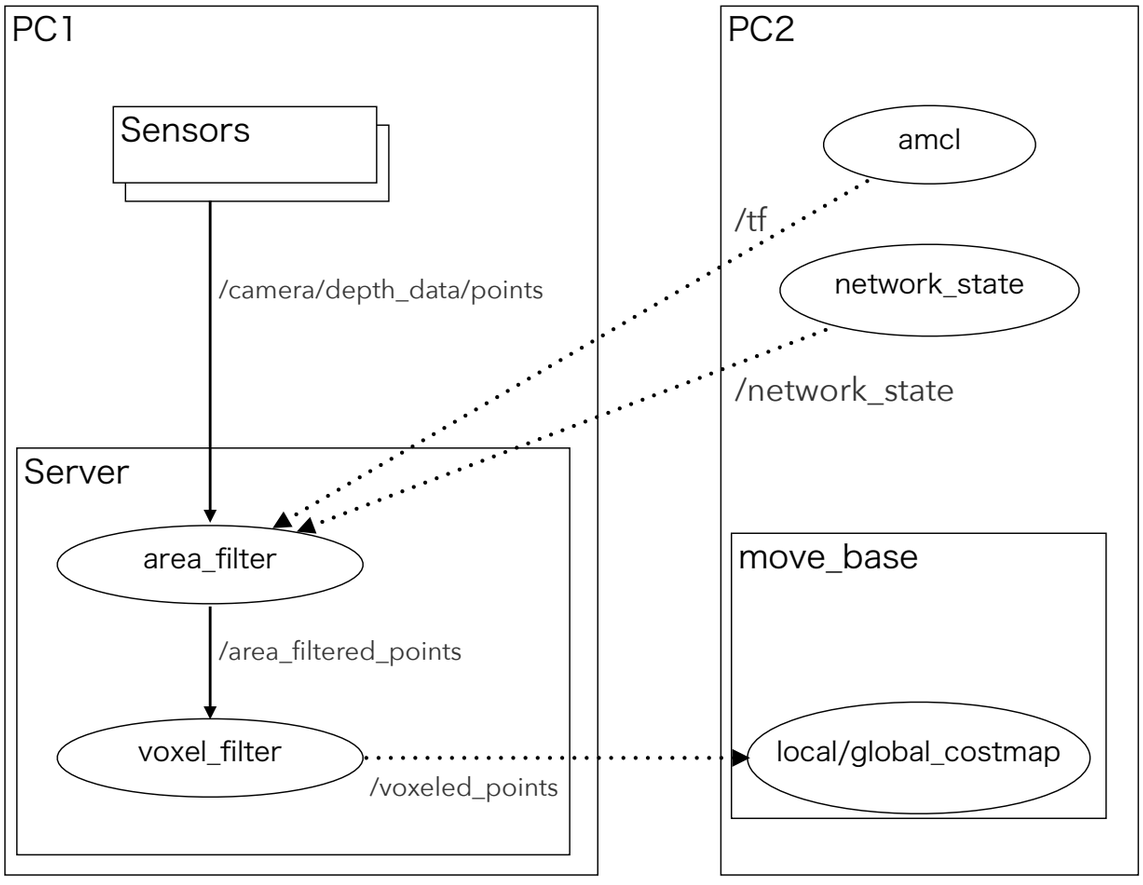


Figure5.1: ROS を用いて実装した提案手法のシステム構成

第 6 章

実験

本章では、5 章で述べた提案手法を実装しシミュレーションを行なった過程とその結果を示す。

6.1 実験概要

本実験では、5 章で述べた提案手法の評価を行うことを目的とする。具体的には、協調的認識を行うことでロボットの経路計画がより効率的になること、ネットワークリソースに応じて送信するデータを変更することで効率的にデータを送信できることを確認することを目指す。

シナリオとしては、デジタルツイン環境でロボットを動かすことを想定し、事前にマップが与えられているものとする。このマップ上に、未知の障害物を設置することで、既存のマップでは通ることのできた経路を塞ぎ、今回の提案手法による経路計画で事前に行き止まりを検知することができるのかを検証する。また、ネットワーク帯域を制限した上で点群の送信を行い、提案手法による送信データの制限が限られたリソース内で有効に働くことを確かめる。さらに補足的に、点群データを送信する範囲を広げることが有効に働くのか確認する。

6.2 シミュレーション環境

サーバー側とロボット側の通信を実現するために、PC2 台を用いて実験を行なった。シミュレーションには gazebo を使用した。実験に使用した PC のスペックを表 6.1,6.2 に示す。gazebo ノードの起動や点群の処理といった重い処理はサーバーに相当する PC1 で行なった。エージェントとして動かすロボットは TURTLEBOT3 の waffle を利用した。waffle の画像を図**??**に示す。waffle にも 360° の Laser scanner が設置されており、この Laser scanner の入力のみを用いて経路計画を行う場合と提案手法による経路計画の比較を行う。

	PC1
CPU	Intel Core i7-10875H
Memory	32GB
GPU	RTX 2070
OS	Ubuntu 18.04
ROS version	Melodic

Table6.1: 実験に使用したサーバー側の PC のスペック

	PC2
CPU	AMD Ryzen 7 5700U
Memory	16GB
OS	Ubuntu 20.04
ROS version	Noetic

Table6.2: 実験に使用したロボット側の PC のスペック



Figure6.1: TURTLEBOT3(waffle)[*1]

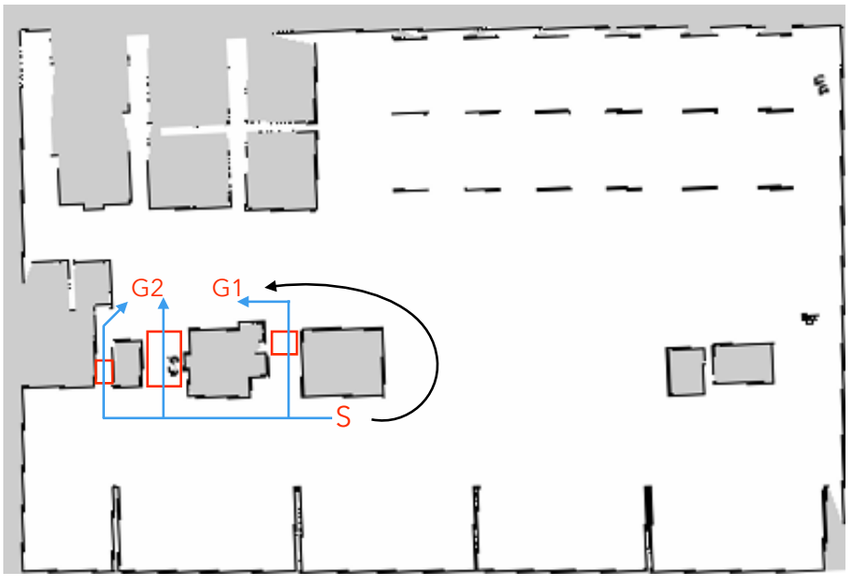


Figure6.2: 事前マップとして用意した画像。赤枠部分に障害物を追加している。

ロボットを走行させる 3 D ワールド及びマップを図 6.2 及び図 6.3 に示す。図 6.2 の S と記している箇所がスタート地点であり、G1 を目的地とする経路 1、G2 を目的地とする経路 2 を使って実験を行う。事前に与えたマップから生成されるパスが通れないかつスタート地点からは死角となっており認識できない箇所に障害物を設置している。

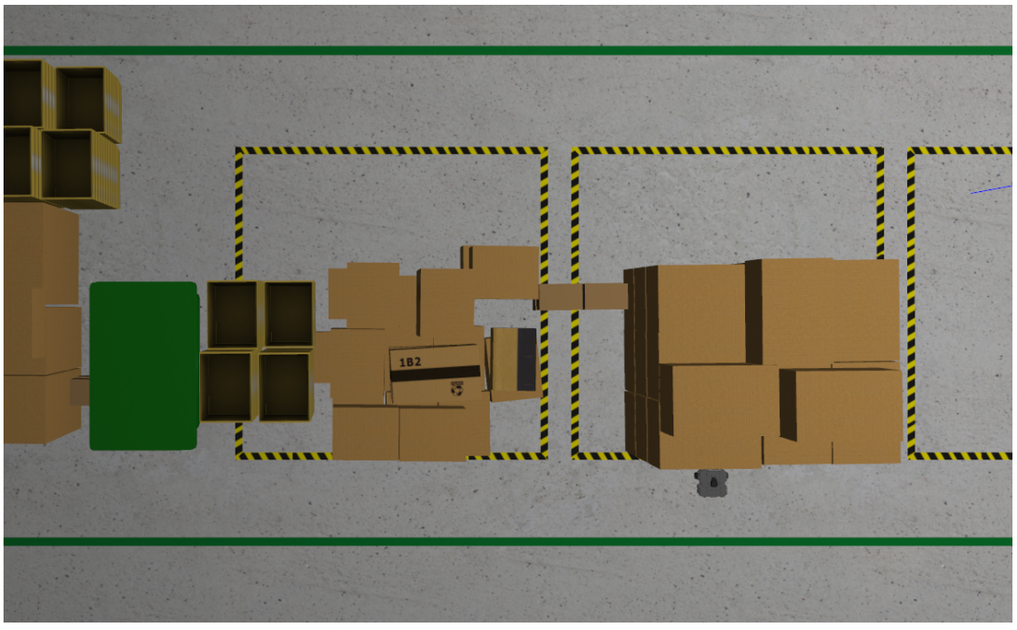


Figure6.3: 図 6.2 の 3 D マップ

6.3 結果

6.3.1 センサによる協調的認識

実際に経路計画を行う前にセンサからの入力によって死角にある障害物を他のセンサからの入力を用いて協調的に認識できることを確かめた。図**??**のワールドで赤枠内に kinect センサを設置している。図 6.6 ではロボットを中心にセンサの入力からバウンディングボックスの範囲内の点群のみを切り取って受信している様子を示している。ここで切り取られたデータが move_base 内のグローバル、ローカルの両コストマップに与えられる。その結果として、図 6.7 に示すように、ロボットからは死角の位置にある障害物がコストマップ上に反映されている。

6.3.2 提案手法と従来手法の比較

本提案手法による協調的な認識を活用することでより効率的な経路計画がなされていることを確認する。協調的な認識によりロボットは死角の障害物を認識できるようになり、将来行き止まりとなる経路などをより早く避けることが期待できる。

経路 1 における目的地までの到着時間及び走行距離を、ロボットのセンサのみを用いて経路計画を行なった場合と提案手法による経路計画を行なった場合の結果を比較した表を 6.3 に示す。(以下、ロボットに取り付けられたレーザースキャナからの入力のみを用いて経路計画を行う手法を従来手法と表す。) 測定方法としては、目的地を送信するノードを起動してから実際に目的地に辿り着くまでの時間を 10 回計測した。また、走行経路については、各フレームごとのロボットの現在地を/odom トピックから受信し、それらをスタートからゴールまで足し合わせることで計算した。なお、ネットワーク帯域は十分大きく設定している。また、実際にロボットが走行したパスを 2 次元マップ上に表した結果を図 6.8 に示す。経路は 10 回の走行のうち最も距離が平均値に近かったものを選択している。それぞれの挙動として、提案手法では、マップ上に新しく設置した障害物をスタート地点から動くことなく認識しているため初めから最短距離を進んでいることが読み取

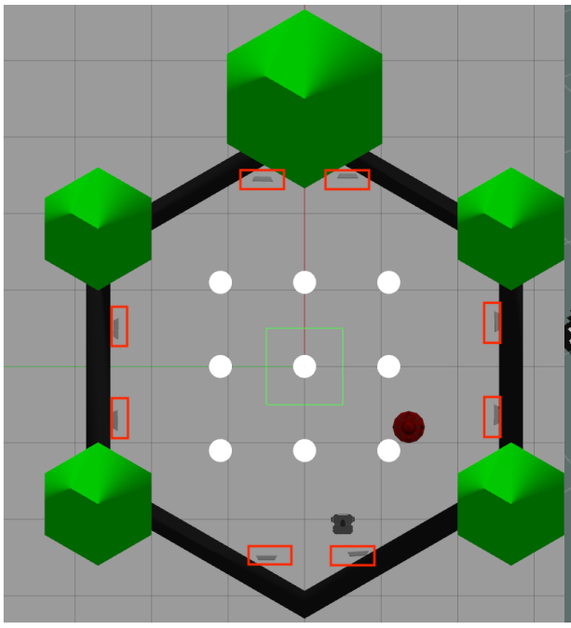


Figure6.4: gazebo で表示させたシミュレーションの図。赤枠内に Lidar センサを設置している。

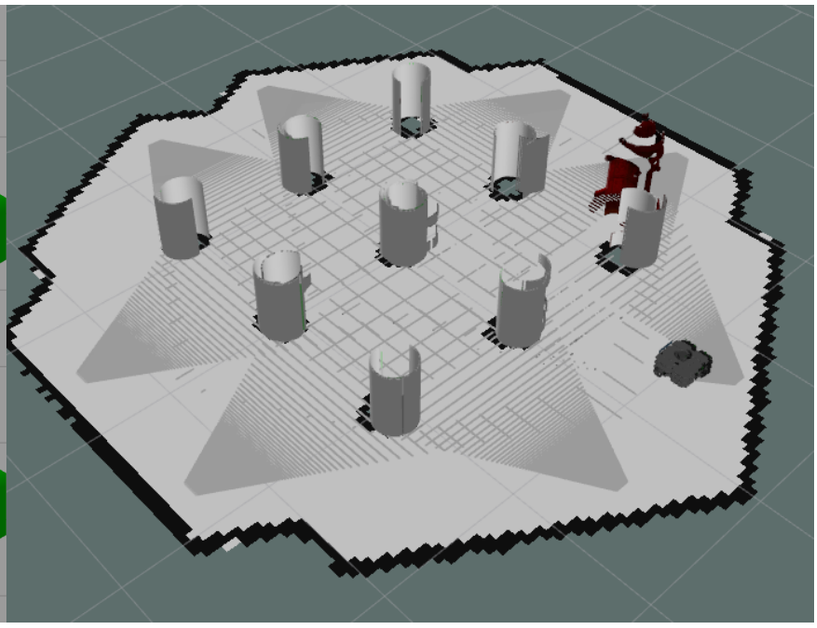


Figure6.5: センサから取得された生の点群データを表示した図

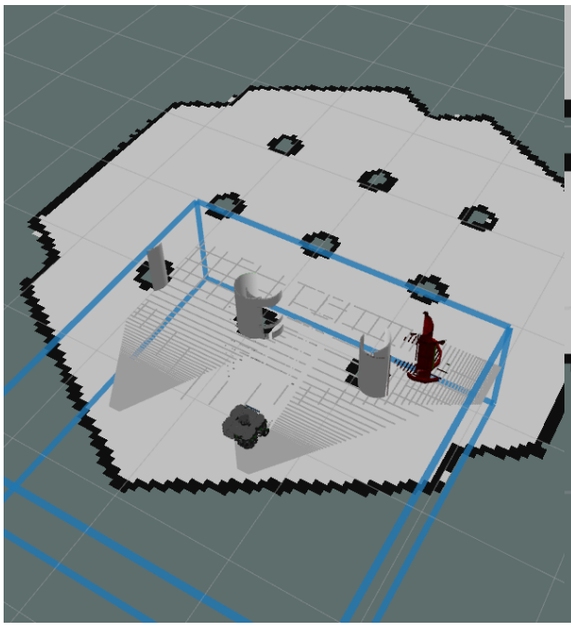


Figure6.6: 選択的に送信されたロボット周辺の点群データ。青枠内の点群が送信されている。

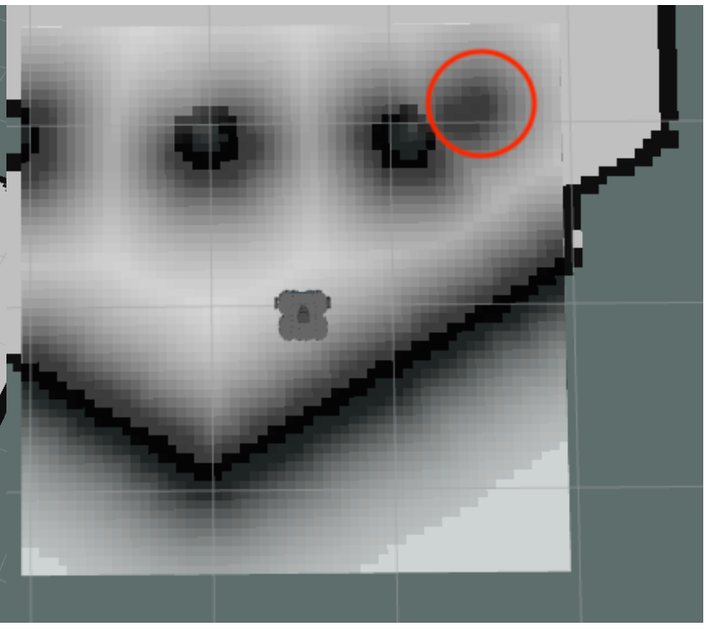


Figure6.7: ローカルコストマップ。ロボット側からは死角にある障害物が認識されている。

れる。一方で、従来手法では、スタート地点から事前マップ上から計算された最短経路の方向へ進み、障害物が自身のセンサで認識出来る位置まで動いた後に正しい経路に進んでいることが分かる。

次に経路 2 においても同様に従来手法と提案手法の比較を行なった。また、ネットワークリソースが十分にある環境と逼迫している環境を想定し、前者を提案手法 1、後者を提案手法 2 とした。この経路 2 の実験においては、点群を切り取るサイズを固定し、提案手法 1 では $10 \times 10 \times 2m^3$ の範囲を、提案手法では $3 \times 3 \times 2m^3$

Table6.3: 経路 1 での実験結果

	従来手法	提案手法
所要時間 (s)	55.2	50.3
走行距離 (m)	3.24	2.71

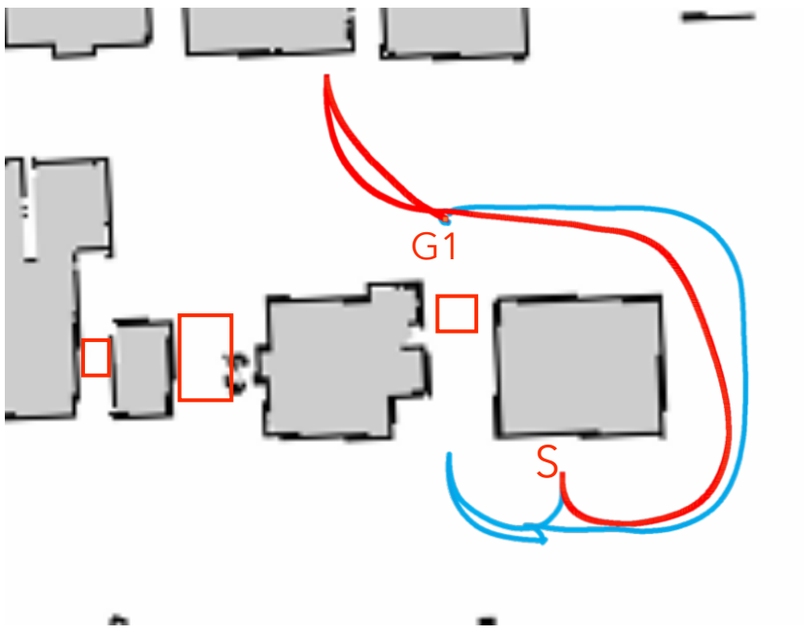


Figure6.8: 経路 1 での提案手法とロボットのセンサのみによる手法での実際の経路。青色が提案手法、赤色がロボットのセンサのみによるもの。

とした。この条件での結果を表 6.4 及び図 6.9 に示す。

Table6.4: 経路 2 での実験結果

	従来手法	提案手法 1	提案手法 2
所要時間 (s)	114.0	59.2	115.6
走行距離	6.33	3.78	6.21

経路 2 では、スタート地点から現在のパスが行き止まりに行き着くことが判明するまでの距離が経路 1 と比べて長くなっている分、最短の経路との差が大きくなっていることが分かる。また、この経路においては、提案手法 2 が従来手法より多くの時間がかかっている。バウンディングボックスの大きさが限られていたことにより障害物を検知するまでの走行経路が従来手法と変わらないことが原因であった。

以上の二つの経路での実験からロボットの死角を十分に補う点群データが与えられれば、協調的な認識によってより効率的な経路計画ができることが確認できた。

6.3.3 提案手法と点群を全て配信した場合の比較

次に、ネットワークリソースに応じて動的に送信する点群データを限定することによって様々な条件下でもデータを配送できることを確認する。ネットワーク帯域を制限した状態で提案手法及び点群を全て送信した場合にロボット側が何フレーム点群を受信することができているのか実験を行なった。ネットワーク帯域は

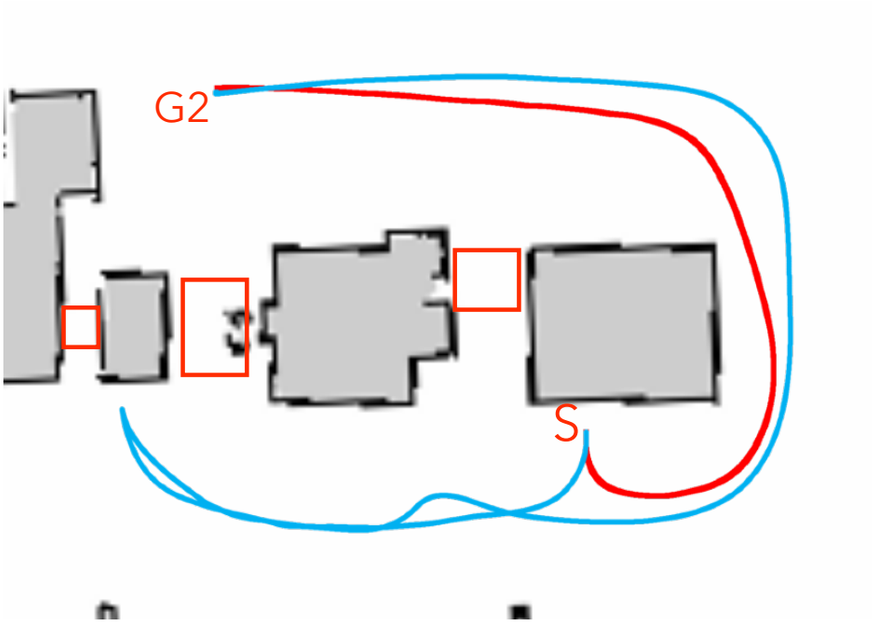


Figure6.9: 経路 2 での提案手法と従来手法での実際の経路。青色が提案手法、赤色が従来手法によるもの。

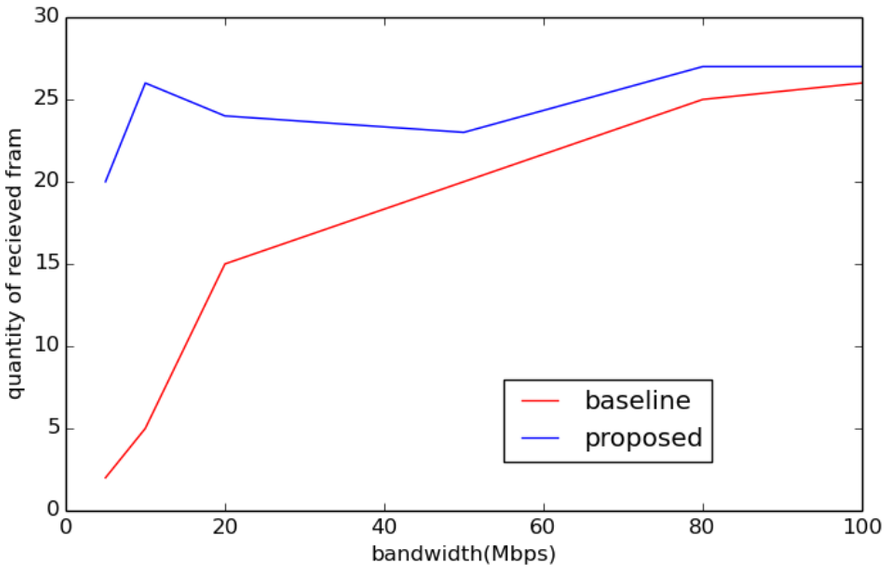


Figure6.10: ネットワークの帯域を制限した時に受け取った点群フレームの数

tc コマンドを用いて 5mbps,10mbps,20mbps,50mbps,80mbps,100mbps として各帯域の場合において、filternetwork_state ノードから 1 秒に 3 回の頻度でロボットの周辺の点群を送信した。それぞれの帯域において 10 秒間で受け取った点群データの総数を従来手法と全ての点群を送信した場合で比較したグラフを図 6.10 に示す。ネットワークリソースが十分にある状況ではそのまま点群を送信しても十分に受け取ることができるが、帯域が制限された場合に違いが現れた。提案手法では帯域を 5mbps パケットロスをほとんど起こすことなく送信できているが、点群を全て送信した場合、90% 以上のデータを送信することができていない。5mbps の場合、そのまま点群を送信するだけでは点群データが 5 秒に一度しか届いていないため、ロボットは各時点においてその 5 秒前のデータをもとに経路計画を行なっていることが分かる。今回のシミュ

レーション環境では静止した障害物のみを設置したため数秒前のデータをもとに動作しても問題はなかったが、実際の室内での運行を考えると、作業者が歩行している状況などが考えられるため、この遅延の影響はより大きくなる。

第 7 章

結論

7.1 まとめ

本論文では，デジタルツインアプリケーション作成の枠組みとして，建造物の 3D データ取得からメタデータの付与，建物に関する情報をアプリケーションに提供する API の作成を行なった．また，この屋内の情報と，オープンデータとして提供されている 3D マップを組み合わせることで，屋内外のデータを繋いだより詳細な 3D マップビューワーを作成した．また，デジタルツインを活用し室内でロボットを動かすために，センサの情報を用いた協調的な認識及び点群を効率的に配送するためのシステムを提案した．その結果として，様々な帯域に対してリアルタイムで点群を送信することに成功した．

7.2 今後の課題

3D モデルの作成については，今回作成した API の応答時間をより早くするための 3D データの処理や，この API を利用して現実世界へのフィードバックまでを含めたデジタルツインアプリケーションの作成することで実用性の評価を行うことや，3D スキャンを通じて獲得した 3D データと BIM を同様に扱うためのデータ処理などが挙げられる．また，CityGML のデータをより扱いやすくするための枠組みの導入などが必要となる．

また，協調的な経路計画については，大規模な点群データを効率的に送信するためのアルゴリズムを考案した．しかし，点群の効率的配送には点群圧縮の技術が不可欠である．また，圧縮以外にもサブサンプリングを行うなど，点群のデータ構造に着目した容量の削減の方法などもある．今後はこうした手法も取り入れ複合的な手法でリアルタイム性を向上させることが課題として挙げられる．また，協調的な認識が特に重要となるのは動く障害物など，死角から飛び出してくる特に必要となるのはよりリアルタイムでの経路計画においてはロボットの死角となる箇所を補うことが大事となるが，本論文における手法は静的な障害物を対象とし，ロボットの周辺の点群を優先的に送信するという手法をとった．しかしロボットの向きや経路などから例えば後方のエリアの重要度は前方よりも低くなるなど，点群の切り取り方には工夫の余地がある．また，今回の手法が動く障害物に対してどれだけ有効であるか検証を行う必要がある．

発表文献と研究活動

(1) 青谷和真、川村地平、塚田学、落合秀也、江崎浩、デジタルツイン基盤とその 3D データ活用．マルチメディア、分散、協調とモバイル (DICOMO2021) シンポジウム、オンライン,2021．

References

[1] A Fuller, Z Fan, C Day, and C Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, Vol. 8, pp. 108952–108971, 2020.

[2] Chiara Cimino, Elisa Negri, and Luca Fumagalli. Review of digital twin applications in manufacturing. *Comput. Ind.*, Vol. 113, p. 103130, December 2019.

[3] Yu Zheng, Sen Yang, and Huanchong Cheng. An application framework of digital twin and its case study. *J. Ambient Intell. Humaniz. Comput.*, Vol. 10, No. 3, pp. 1141–1153, March 2019.

[4] T Ruohomäki, E Airaksinen, P Huuska, O Kesäniemi, M Martikka, and J Suomisto. Smart city platform enabling digital twin. In *2018 International Conference on Intelligent Systems (IS)*, pp. 155–161. ieeexplore.ieee.org, September 2018.

[5] Ayesha Ameen, Khaleel Ur Rahman Khan, and B Padmaja Rani. Extracting knowledge from ontology using jena for semantic web. In *International Conference for Convergence for Technology-2014*, pp. 1–5. ieeexplore.ieee.org, April 2014.

[6] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pp. 205–227. IGI Global, 2011.

[7] Thomas H Kolbe, Gerhard Gröger, and Lutz Plümer. CityGML: Interoperable access to 3D city models. In Peter van Oosterom, Siyka Zlatanova, and Elfriede M Fendel, editors, *Geo-information for Disaster Management*, pp. 883–899. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[8] Tatjana Kutzner, Kanishk Chaturvedi, and Thomas H Kolbe. CityGML 3.0: New functions open up new applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, Vol. 88, No. 1, pp. 43–61, February 2020.

[9] Markus Koenig Christian KochJakob Beetz Andr ´ e Borrmann. *Building Information Modeling :Technology Foundations and Industry Practice*. Springer, 2018.

[10] Shu Tang, Dennis R Shelden, Charles M Eastman, Pardis Pishdad-Bozorgi, and Xinghua Gao. A review of building information modeling (BIM) and the internet of things (IoT) devices integration: Present status and future trends. *Autom. Constr.*, Vol. 101, pp. 127–139, May 2019.

[11] Alex Bradley, Haijiang Li, Robert Lark, and Simon Dunn. BIM for infrastructure: An overall review and constructor perspective. *Autom. Constr.*, Vol. 71, pp. 139–152, November 2016.

[12] Kendrik Yan Hong Lim, Pai Zheng, and Chun-Hsien Chen. A state-of-the-art survey of

digital twin: techniques, engineering product lifecycle management and business innovation perspectives. *J. Intell. Manuf.*, Vol. 31, No. 6, pp. 1313–1337, August 2020.

[13] Chao Wang, Yong K Cho, and Changwan Kim. Automatic BIM component extraction from point clouds of existing buildings for sustainability applications. *Autom. Constr.*, Vol. 56, pp. 1–13, August 2015.

[14] Matterport. `https://matterport.com`.

[15] Mads Holten Rasmussen, Maxime Lefrançois, Georg Ferdinand Schneider, and Pieter Pauwels. BOT: The building topology ontology of the W3C linked building data group. *Semant. Web*, pp. 1–19, October 2020.

[16] Qian Wang, Yi Tan, and Zhongya Mei. Computational methods of acquisition and processing of 3D point cloud data for construction applications. *Arch. Comput. Methods Eng.*, Vol. 27, No. 2, pp. 479–499, April 2020.

[17] Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3D point clouds. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 514–524. ieeexplore.ieee.org, July 2019.

[18] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative perception for 3D object detection in driving scenarios using infrastructure sensors. *IEEE Trans. Intell. Transp. Syst.*, pp. 1–13, 2020.

[19] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. AVR: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, pp. 81–95, New York, NY, USA, June 2018. Association for Computing Machinery.

[20] S Kim, B Qin, Z J Chong, X Shen, W Liu, M H Ang, E Frazzoli, and D Rus. Multivehicle cooperative driving using cooperative perception: Design and experimental validation. *IEEE Trans. Intell. Transp. Syst.*, Vol. 16, No. 2, pp. 663–680, April 2015.

[21] Chao Cao, Marius Preda, and Titus Zaharia. 3D point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*, Web3D '19, pp. 1–9, New York, NY, USA, July 2019. Association for Computing Machinery.

[22] Mohammad Hosseini and Christian Timmerer. Dynamic adaptive point cloud streaming. In *Proceedings of the 23rd Packet Video Workshop*, PV '18, pp. 25–30, New York, NY, USA, June 2018. Association for Computing Machinery.

[23] Feng Qian, Bo Han, Jarrell Pair, and Vijay Gopalakrishnan. Toward practical volumetric video streaming on commodity smartphones. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, HotMobile '19, pp. 135–140, New York, NY, USA, February 2019. Association for Computing Machinery.

[24] Jounsup Park, Philip A Chou, and Jenq-Neng Hwang. Rate-Utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 9, No. 1, pp. 149–162, March 2019.

[25] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

謝辞

本研究ならびに本論文の執筆にあたり, 塚田学准教授と落合秀也准教授，江崎浩教授には熱心なご指導, ご鞭撻を賜りました. 厚く御礼申し上げます．落合秀也准教授は指導教官として，合宿や発表練習の際に的確な助言をいただいたりと，大変お世話になりました．塚田学准教授には，研究で行き詰まった際にアドバイスをいただいたり，ミーティングの機会を設けていただいたりと，修士 2 年間での生活を大きく支えていただきました．江崎浩教授にも発表練習の際など優しい言葉をかけていただき，非常に心の支えとなりました．また，SDM の会合やワーキンググループなどでお世話になった方々，研究室の同期や先輩後輩，秘書の方も含め深く感謝しております．最後に，これまでの私の生活を支えて頂いた家族・友人にも心から感謝致します..