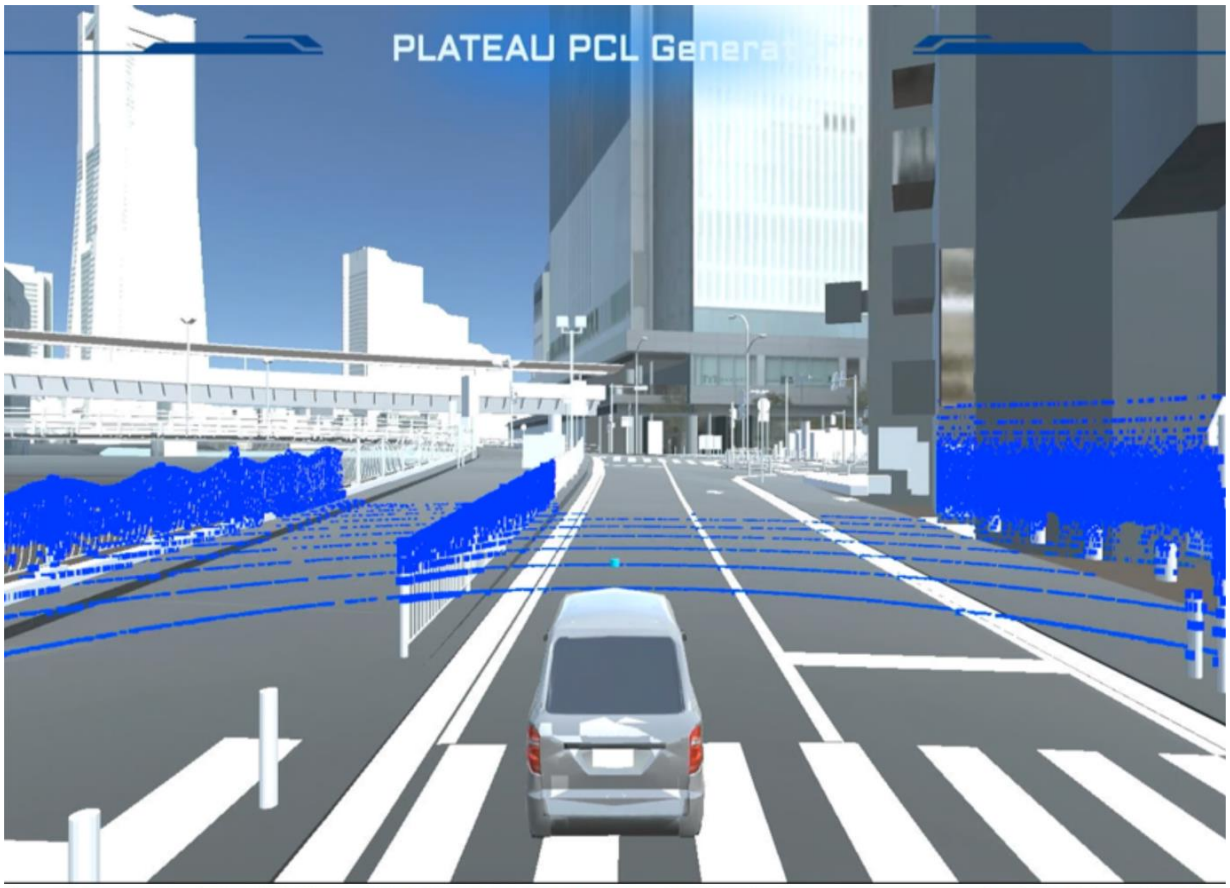
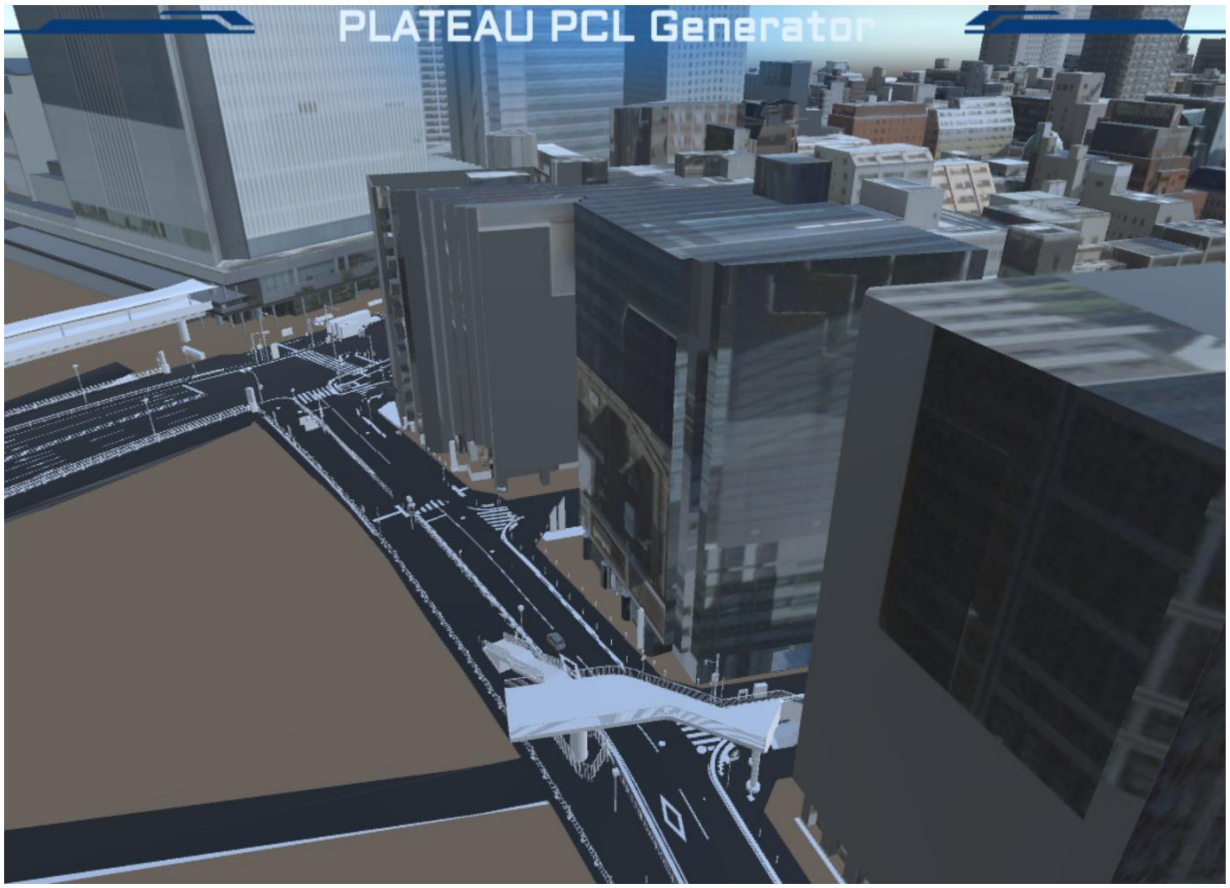
PLATEAU by MLIT

PLATEAU Technical Report

3D 都市モデル活用のための技術資料





3D 都市モデルと BIM を活用したモビリティ自律運航システム（車両） v2.0 技術検証レポート

series No. 87

Technical Report on Autonomous Mobility Operation System Utilizing 3D City Models and BIM Models (Part: AGV) v2.0

目次

1. ユースケースの概要

1-1. 現状と課題

資材運搬や物流等を担う搬送車両の自律走行システムとしては、GPS 測位による自己位置測位技術を用いることが一般的だが、ビルの間など受信環境が悪い場所での精度担保が難しく、公道での安全な自律走行の確立には至っていない。他方、LiDAR SLAM を用いた自己位置測位は高い精度を担保することが可能だが、点群マップの事前取得のコストやルート選定の硬直性が課題となっている。

そのため、2022 年度に実施した「3D 都市モデルと BIM を活用したモビリティ自律運行システム」では、3D 都市モデルを配置した仮想空間の中で仮想車両及び仮想 LiDAR を稼働させることで想定ルート上の点群データを取得し、これを用いて現実空間の自己位置測位を行うシステムを開発した。

このシステムを用いて、大阪市舞洲地区の公道で自律走行を行い、デジタルツインビュワーを用いたモニタリングを行った。結果として、3D 都市モデルから作成した点群マップで自己位置測位を行い、自律走行が可能であることを確認できた。特に、道路の両脇に特徴のある建築物がある場合、あるいは建築物が周りになくとも植樹・道路標識がある場合に自己位置測位の精度が向上することが確認できた。

一方で、現実空間と 3D 都市モデルがかい離がある場合（駐車場にある植え込み、立ち入り禁止のためのフェンス、スタジアムの柱が 3D 都市モデルにはない等）に自己位置測位の精度が下がり、自律走行が難しいことも明らかとなった。加えて、実世界と 3D 都市モデルのかい離がない場合でも、15km/h を超える走行速度帯におけるマッチングの精度及びスピードに課題があることが分かった。

なお、2022 年度の検証では LOD3 エリアで自動走行を行うことを前提としていた。しかし 2023 年度時点では、LOD3 が整備されているエリアが限られており、3D 都市モデルが一般的に整備されている LOD1 や LOD2 のエリアでもシステムを利用できることがスケーラビリティの課題となった。

1-2. 課題解決のアプローチ

昨年度の実証実験で課題となった、仮想空間と現実のかい離による精度低下や、15 ㎞/h 以上の速度に対応できなかったという課題に対して、今回の実証実験では、仮想点群生成プロセスの汎用化および自己位置測位の機能向上を目指した改修を行う。

自己位置測位は事前に作成された点群マップと、車両に取り付けられた 3D LiDAR の点群データとのマッチングにより算出される。自己位置推定の精度を向上させるためには、より現実と近しい点群マップを用意する必要がある。今回の実証実験では、実際の検証で使う車両や 3D LiDAR の取り付け位置を再現するために、車両の形状や 3D LiDAR の取り付ける位置を変更可能とする UI の開発を行う。3D LiDAR の取り付け位置だけでなく、車両が走行するコースも重要となるため、実際に走行するルートを再現するために、車両の走行ルートを設定できる機能を実装する。
また、上記システムは PLATEAU SDK for Unity を活用することで、3D 都市モデルを FBX 形式に変換して利用するプロセスを省き、より効率化されたフローを確立することで、スケーラビリティ向上を目指す。
そして、昨年度は公道の自動運転を行なったため 15km/h の速度で検証するという制約があり、車両の速度が自己位置測位に与える影響を把握することが出来なかった。今回の実証実験では車両の速度別で検証できる環境を用意し、車両の速度が自己位置測位に与える影響の検証を行う。

3D 都市モデルから生成した点群マップによる汎用的かつ高精度の自己位置測位手法を確立した本システムにより、運輸サービス事業者やゼネコンが安価且つ効率的な点群マップの生成と高精度な自己位置推定機能を使用可能になり、AGV などの無人搬送車両を活用した新たなモビリティサービスの社会実装を加速させることができる。

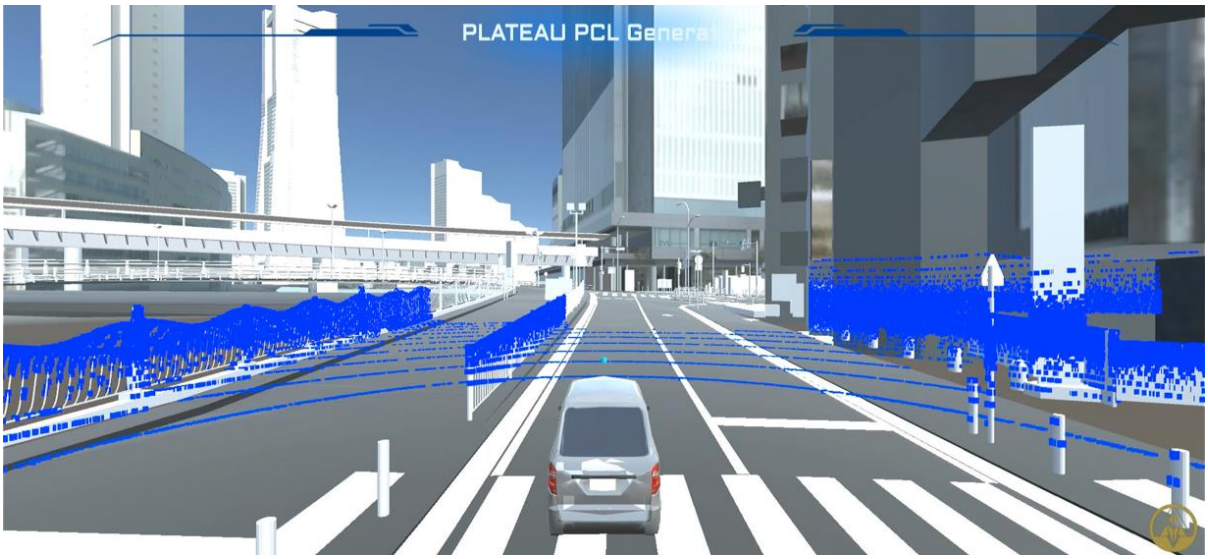


図 1- 1　PLATEAU PCL Generator にて仮想車両が走行している様子

図 1-2 Autoware にて自己位置の測定をしている様子　俯瞰ビュー

1-3. 創出価値

都市部等が抱える交通課題の一つとして、建設工事の資材運搬等による渋滞が挙げられる。交通インフラが滞ることで、工事の施工計画だけでなく、近隣住民の生活や都市機能への影響も及んでいる。この課題を解決するために、昨今、道路における搬送車両に最適化された自律走行が求められている。
現在、自律走行を行うにあたり、主に二つの手法がある。第一に GPS による自己位置測位技術である。一般的に精度の高い手法ではあるが、GPS の電波が受け取りにくい環境下である都市部や屋外/屋内の切り替わる場所等、受信環境に影響される欠点がある。第二に、点群マップと 3D LiDAR による自己位置測位技術がある。こちらも GPS 同様に高い精度を誇るが、点群マップの事前取得が必要であることや走行ルート選定が柔軟に行えない等、欠点がある。

本ユースケースでは、第二の手法の欠点を補うことに焦点を当てた。具体的には、ROS 等のオープンソースのモビリティ用ソフトウェアを活用し、3D 都市モデルと BIM を統合した地図データをマップとして、無人搬送車両（AGV）自律走行の一助となるオペレーションシステムを開発する。主な対象者は、ゼネコンや運輸サービス事業者とする。将来的には、3D 都市モデルと BIM モデルを統合したモビリティのオペレーションシステムにより、通常の自律走行運搬に加え、建設現場への「ラストワンマイル」へのアプローチも可能とする。これらのユーザーインタフェース開発と提供を行うことで、屋外/屋内のシームレスな移動を目指す。

1-4. 想定事業機会

表 1-1 想定事業機会

項目	内容
利用者	● ゼネコン ● 運輸サービス事業者
サービス仮説	● 自律走行の搬送車両による業務支援サービス ➢ 資材運搬物流などの省力化につながる自律走行システムによる業務支援サービスを提供 ● 自律走行の搬送車両の遠隔モニタリングサービス ➢ 自己位置測位とデジタルツインビューによる自律走行支援型遠隔車両の管制サービスの提供
提供価値	● GPS 測位のみでは安全な公道自律走行が担保できない搬送車両に対して 3D マップデータを活用した安全な自律走行システムを提供 ➢ 資材運搬や物流等を担う搬送車両の自律走行を可能とするため、LiDAR や GPS 等のセンサと 3D 都市モデルを利用したマップによる自己位置測位を組み合わせたシステム ● 建築・物流事業者が利用可能な、デジタルツインビューを組み合わせたユーザーインタフェースを開発し提供

2. 実証実験の概要

2-1. 実証仮説

- 今回の実証実験では、昨年度課題となった仮想点群生成の汎用性向上に向けて、3D 都市モデルから仮想 LiDAR を用いて点群マップを生成するシステムを開発する。仮想空間上の走行環境を現実世界に近づけるためのパラメータ設定や、取得した点群に対して最適なノイズフィルタ処理を実行することで、異なる都市特性や 3D 都市モデルの各 LOD 、現実世界を走行する車の速度にも対応した点群マップ生成システムを実現する。
- 本システムを用いた仮想点群生成プロセスについて、3D 都市モデルがあれば容易に点群マップの生成が可能な仕様にするため、汎用的なゲームエンジン上に仮想空間内でルートの生成や仮想 LiDAR を簡易的に利用しやすいインタフェースの開発を行なった。また LOD3 の整備エリアだけでなく、LOD2 の整備エリアでも、建物や道路の特徴次第では本システムを利用した自己位置測位が可能となる。

2-2. 実証フロー

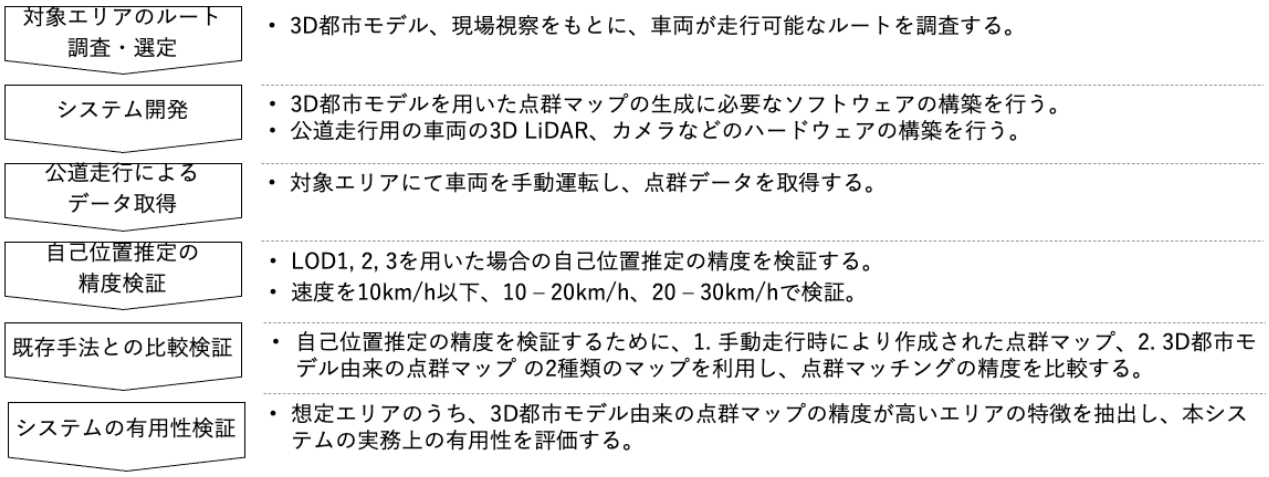


図 2-1 実証フロー

2-3. 検証ポイント

- 3D 都市モデルから仮想 LiDAR を用いて生成した点群マップを用いて、現実空間の都市特性や整備されている 3D 都市モデルにおける LOD 別の自己位置推定精度
  - 3D 都市モデルを元データとして Unity 上に取り込み、仮想車両の車幅・高さ・長さ及び LiDAR の周波数やステップ数などのパラメータの調整やノイズフィルタを使用することで、現実空間に近い点群マップを生成できることを確認する。
  - 仮想空間で作成した点群マップと、現実空間を走行する車両から取得した点群データをリアルタイムでマッチングさせ、高精度且つリアルタイムな自己位置測位ができることを確認する。
  - 上記 2 点に対して、都市の特性の違う「神奈川県横浜市 みなとみらい地区」と「大阪府大阪市 舞洲地区」で実施し、走行ルート内での建築物の地物の量や整備されている 3D 都市モデルの LOD 別における差分を抽出する。

上記 1 点の検証ポイントについては、【4 章：実証技術の検証】にて検証結果を記載

- 仮想点群生成プロセスの工数削減に対する本システムの有用性
  - 点群マップの事前取得に掛かる工数（人日）と 3D 都市モデル由来の点群マップを生成する工数を比較し、効率化に寄与していることを確認する。
  - また、システム利用が想定される事業者等に対しヒアリングを実施し、対象先の既存使用システムや手法と比較して、自己位置推定に必要な精度を担保しつつ、ユーザービリティが向上されているかを確認する。

上記 1 点の検証ポイントについては、【5 章：BtoB ビジネスでの有用性検証】にて検証結果を記載

2-4. 実施体制

表 2-1 実施体制

役割	主体	詳細
全体管理	国交省 都市局	プロジェクト全体ディレクション
	アクセンチュア	プロジェクト全体マネジメント
実施事業者	竹中工務店	ユースケース実証における全体管理 開発・実証　実行とりまとめ
	アダワープジャパン	車両対応システム開発・実証

2-5. 実証エリア

表 2-2 実証エリア

項目	内容
実証地	神奈川県横浜市中区　みなとみらい線馬車道駅周辺
面積	0.75 ㎢
マップ（対象エリアは赤枠内）	

表 2-3 実証エリア

項目	内容
実証地	大阪府大阪市此花区　舞洲地区、夢舞大橋周辺
面積	0.57 ㎢
マップ （対象エリアは赤枠内）	

2-6. スケジュール

表 2-4 スケジュール

実施事項	2023 年									2024 年		
	4 月	5 月	6 月	7 月	8 月	9 月	10 月	11 月	12 月	1 月	2 月	3 月
1. 定例会議	←	—	—	—	—	—	—	—	—	—	→	
2. 報告書作成	←	—	→	←	→					←	—	→
3. 車両データ取得						←	—	—	→			
4. 車両仕様検討	←	—	→									
5. 3D 都市モデル整備		←	→									
6. システム仕様策定・開発			←	—	→							
7. PLATEAU SDK for Unity 対応				←	→							
8. 自己位置測位精度割合向上						←	—	→				
9. 公道走行						←	—	—	→			

3. 実証システム

3-1. アーキテクチャ

3-1-1. システムアーキテクチャ

今回の実証実験では、仮想点群生成プロセスの汎用化を実現するため、3D 都市モデルから仮想 LiDAR を用いて点群マップを生成する「PLATEAU PCL Generator」を開発した。PLATEAU PCL Generator は点群マップを出力するにあたり、具体的には、「①3D 都市モデルの配置機能」「②ルート作成機能」「③車両パラメータ設定機能」「④LiDAR パラメータ設定機能」「⑤点群取得機能」の 5 つの機能を新規開発した。

① 3D 都市モデルの配置機能は、 CityGML 形式の 3D 都市モデルを読み込み、Unity シーン上に配置する。実装には PLATEAU SDK for Unity を用いた。
② ルート作成機能は、Unity シーン内の 3D 都市モデル空間を車両が走行するルートを設定する。実装には PLATEAU SDK for Unity で読み込んだ道路モデルをベースとして、道路モデル上に WayPoint のオブジェクトを配置し、その WayPoint オブジェクトを繋げたコースを仮想車両が走行するルートとして設定するプログラムを開発した。
③ 車両パラメータ設定機能は、仮想 LiDAR を搭載した車両を②で作成したルートで走行させることで走行ルート周辺の空間の三次元点群測量を行う。仮想空間上で車両を走行させることにより、現実空間における点群測量と同じ状況を再現することができる。車両の横幅や車高といった形状や走行速度、LiDAR 取付け位置等を設定することができる。実装にはシーン上に配置された車両オブジェクトを対象に、PLATEAU PCL Generator 上のダイアログに数値を入力することで、車幅、高さ、長さを設定する UI を開発した。
④ LiDAR パラメータ設定機能は、③の機能と連携し、仮想空間上の車両に搭載する LiDAR の照射飛距離や角度、周波数やステップ数を設定する。実装にはシーン上に配置された LiDAR オブジェクトを対象として、③の実装と同様に PLATEAU PCL Generator 上のダイアログに入力された数値を照射飛距離や角度、周波数やステップ数を設定できる UI を開発した。
⑤ 点群取得機能は、③及び④と連携し、LiDAR を搭載した車両が走行する際の点群の照射をシミュレートすることで、仮想空間上で点群が照射された建物モデル、道路モデル、都市設備モデル、植生モデルの点群を取得する。実装には LiDAR シミュレーションライブラリである Robotec GPU LiDAR を用いた。これらの機能により、昨年度の実証実験における課題であった仮想点群生成のプロセスを汎用化し、3D 都市モデルが整備されている都市であれば、仮想空間内のシミュレーション結果を基に PCD 形式で点群マップを簡易に出力することが可能になる。

PLATEAU PCL Generator で作成した点群マップは、現実空間を走行する車両から取得した点群データとリアルタイムでマッチングさせることで、自律走行オペレーションのための自己位置測位を行う。また、自律走行のオペレーションシステムとして Autoware の点群マッチング機能を使用する。データ処理のリアルタイム性向上のため、出力された点群マップを事前に Cloud Compare を用いて軽量化した上で Autoware に読み込みを行う。

自動走行用の自己位置測位システムとして、Autoware の機能である Autoware-sensing 及び Autoware-localization を用いた。まず、Autoware-sensing は LiDAR から取得した点群データを Autoware に取り込む役割を担う。具体的には、LiDAR から取得した点群データを取り込み、Autoware で処理するために ROS（Robot Operating System）メッセージに変換の上、Autoware-localization へ出力する。

次に、Autoware-localization は先述した Autoware-sensing から出力された点群データと、あらかじめ仮想空間上で生成した点群マップを突合するスキャンマッチングという手法を用いて自己位置測位を行う機能を担う。スキャンマッチングには、自律走行への活用を見据えてリアルタイム性が必要なことから、計算速度が速く、点群データのノイズに強い NDT スキャンマッチングを採用した。なお検証では、無人搬送車両を想定し、実車両を用いた走行では、トラック車両に全方位 LiDAR の OS1（Ouster 社製）を搭載することで点群データの取得を行っている。

3D 都市モデルの LOD レベルが自己位置測位に与える影響を調査するために、建物 LOD2、道路 LOD1 のエリアと建物 LOD3、道路 LOD3、都市設備 LOD3、植生 LOD3 のエリアの点群マップを出力し、みなとみらい地区で走行検証を行った。本システムでは PLATEAU SDK for Unity を用いており、3D 都市モデルをシーン上に読み込む際に 3D 都市モデルの LOD レベルを選択することが可能となる。

本システムのシステム・アーキテクチャは下記の通りである。

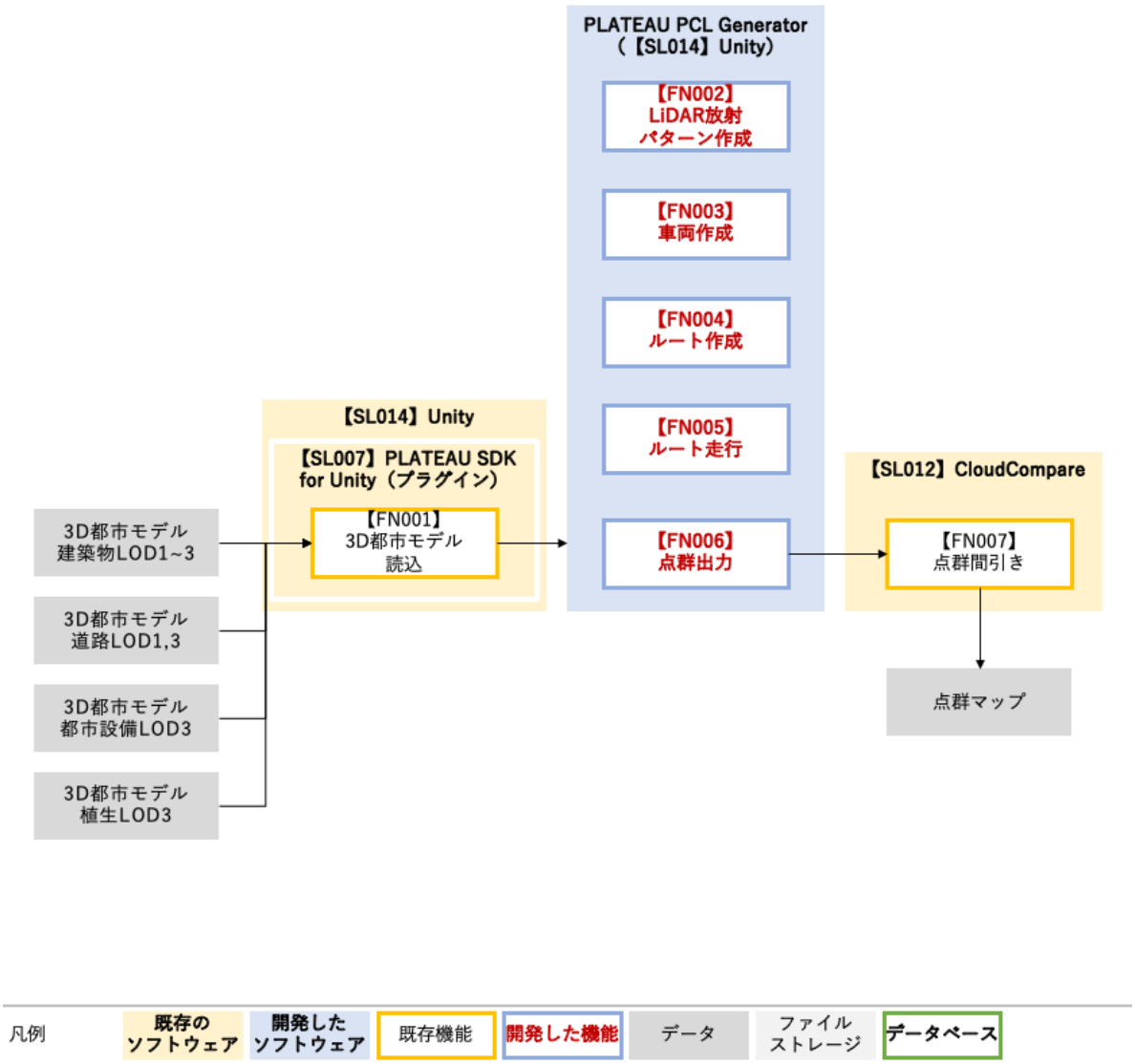


図 3-1 システムアーキテクチャ（点群マップ生成）

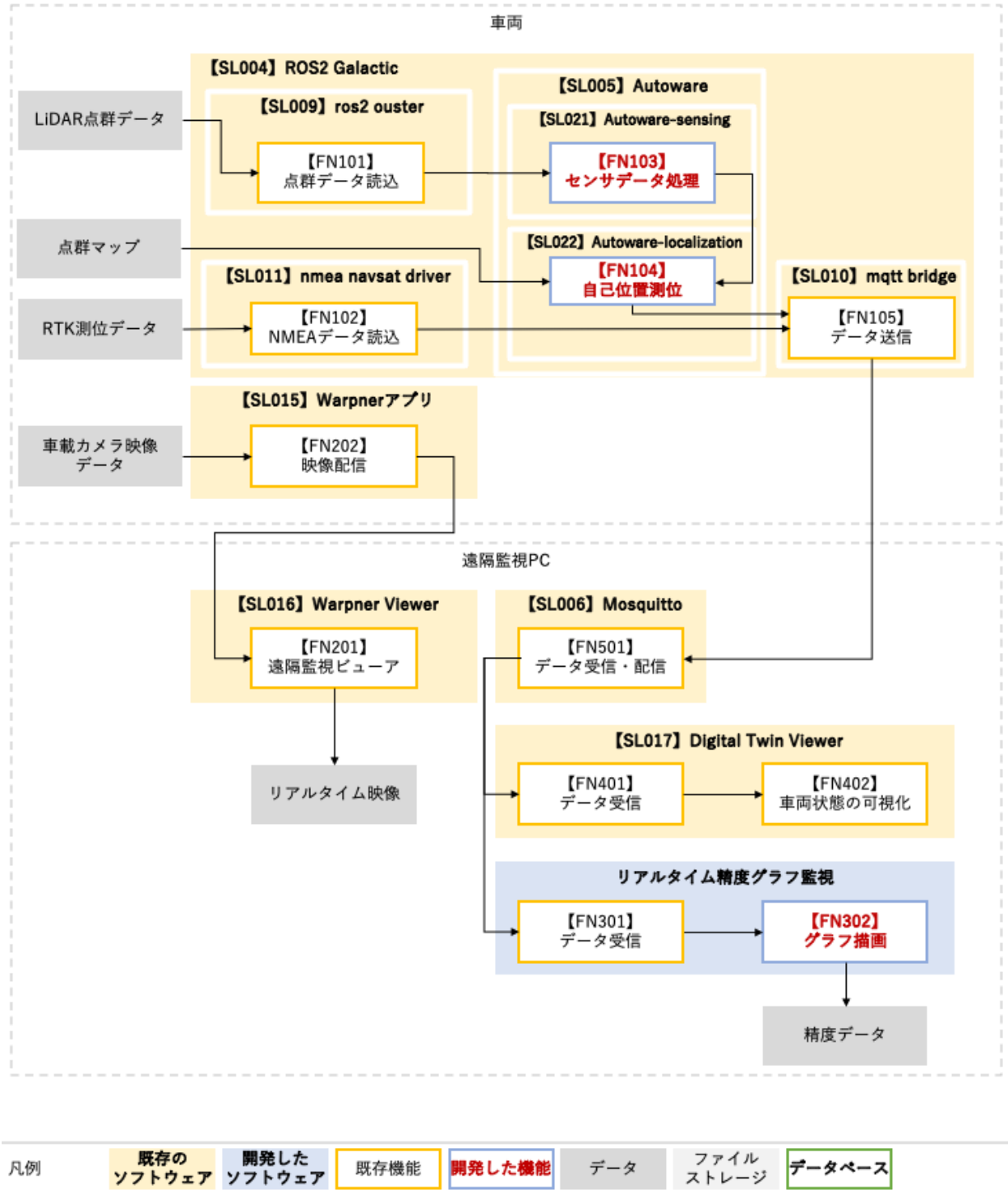


図 3-2 システムアーキテクチャ（車両自律運航）

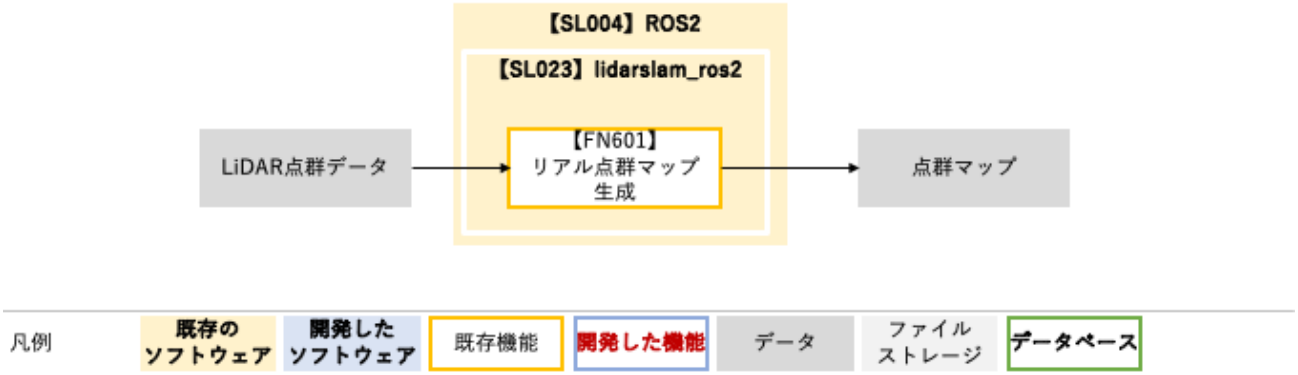


図 3-3 システムアーキテクチャ（リアルマップ生成）

3-1-2. データアーキテクチャ

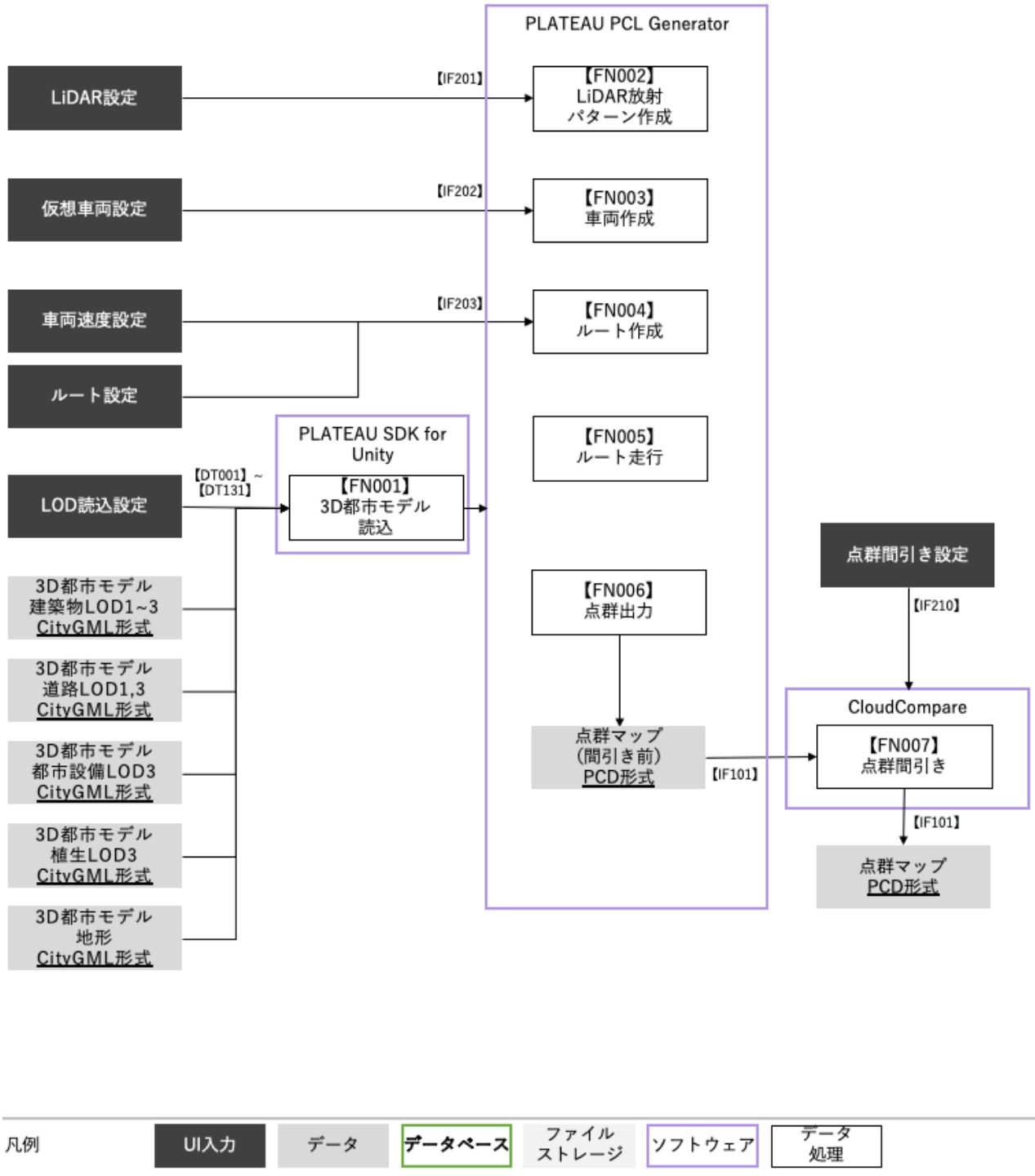


図 3-4 データアーキテクチャ（点群マップ生成）

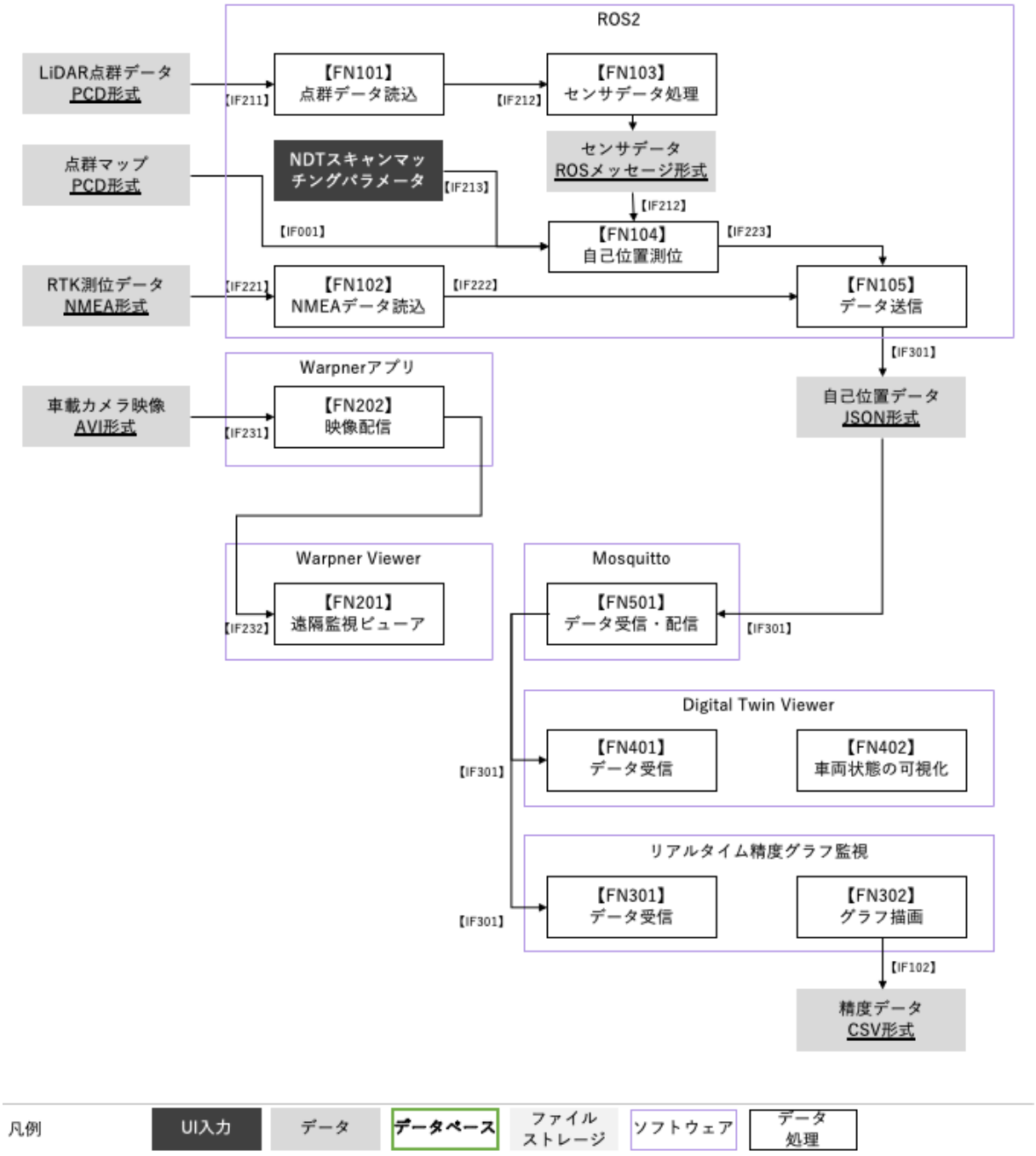


図 3-5 データアーキテクチャ（車両自己位置測位）

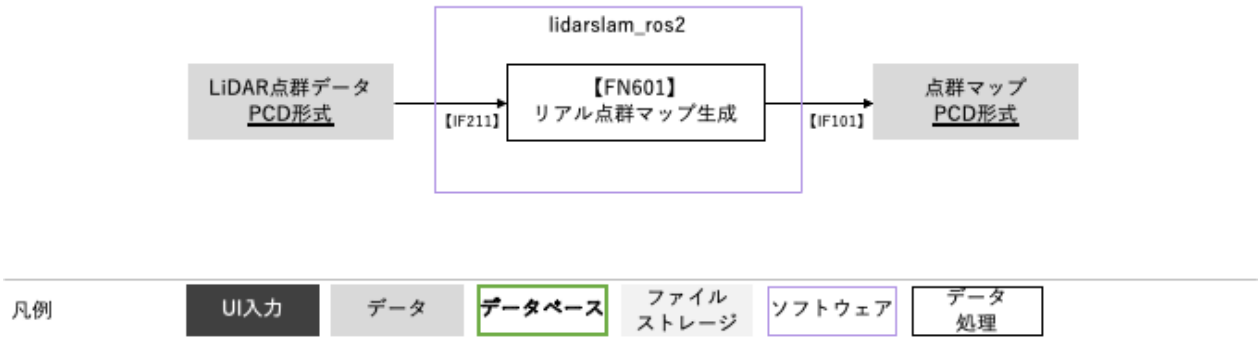


図 3-6 データアーキテクチャ（リアルマップ生成）

3-1-3. ハードウェアアーキテクチャ

3-1-3-a. 利用したハードウェア一覧

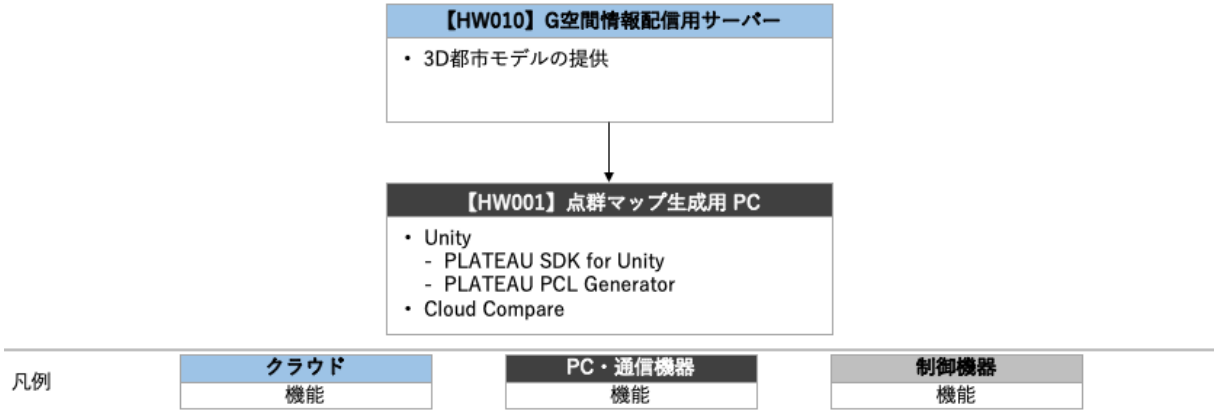


図 3-5 ハードウェアアーキテクチャ（点群マップ生成）

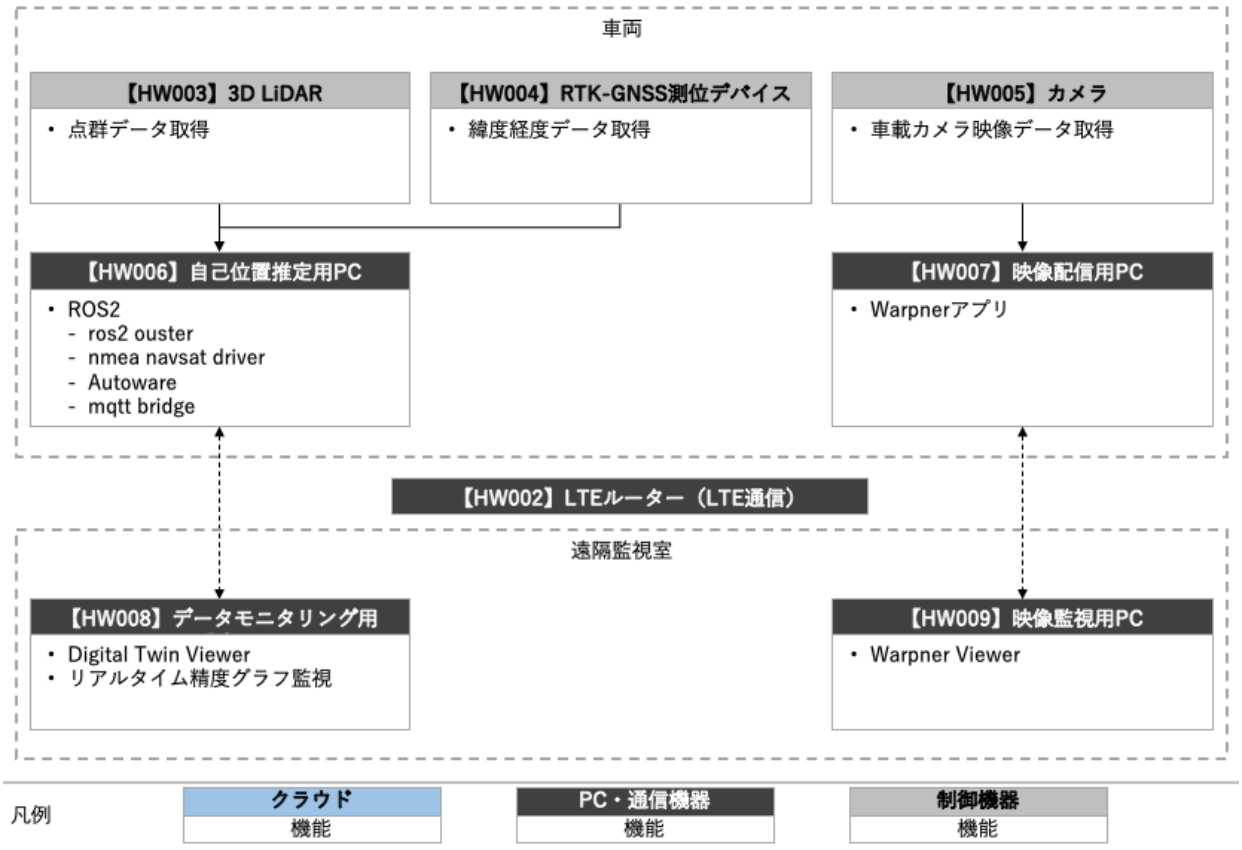


図 3-7 ハードウェアアーキテクチャ（車両自己位置測位）

表 3-1 利用したハードウェア一覧

ID	種別	品番	用途
HW001	点群マップ生成用 PC	GALLERIA XL7C-R45	● Unity ● PLATEAU SDK for Unity ● PLATEAU PCL Generator ● Cloud Compare
HW002	LTE ルーター	GLiNet Puli	● 外部システム連携のために車両に搭載する LTE ルーター
HW003	3D LiDAR	Ouster OS-1	● 自己位置測位の点群マッチングのために用いる 3D LiDAR
HW004	RTK-GNSS 測位デバイス	Ichimill	● 自己位置測位の点群マッチングのために用いる RTK-GNSS 測位デバイス
HW005	カメラ	CMS-V43BK	● 車両の遠隔監視のために用いるカメラ
HW006	自己位置推定用 PC	System76 Kudu	● 自己位置測位のソフトウェアを動かすための Ubuntu PC
HW007	映像配信用 PC	GMK Tech NAB5	● 映像を配信するための Ubuntu PC
HW008	データモニタリング用 PC	GALLERIA XL7C-R45	● Digital Twin Viewer ● リアルタイム精度グラフ監視
HW009	映像監視用 PC	GALLERIA XL7C-R45	● Warpner Viewer
HW010	G 空間情報配信用サーバー	-	● 3D 都市モデルの提供

3-1-3-b. 利用したハードウェア詳細

1)　【HW001】点群マップ生成用 PC：GALLERIA XL7C-R45

- 選定理由
  - 点群マップ作成処理に必要なメモリを持つため
  - 同様のスペックを持つ PC が監視用、遠隔操作用で合わせて 2 台必要なため
- 仕様・スペック
  - サイズ：縦 245mm×横 360mm×高さ(最薄部)20mm
  - OS：Windows 11
  - CPU：Intel®i7-13700H
  - メモリ：8GB
  - ストレージ：500GB
- イメージ

図 3-8 GALLERIA XL7C-R45

2)　【HW002】LTE ルーター：GLiNet Puli

- 選定理由
  - 今回の実証実験に十分な通信速度が担保可能であり、充電式バッテリー付属のため実証が長時間になった場合にも対応可能なため
- 仕様・スペック
  - サイズ：縦 120mm×横 74mm×高さ 28mm
  - CPU：QCA9531 @650MHz Soc
  - WiFi 速度：300Mbs(2.4Ghz)
  - Storage：NOR フラッシュ 16MB + NAND フラッシュ 128MB
- イメージ

図 3-9 GLiNet Puli

3) 【HW003】3D LiDAR：Ouster OS1-64

- 選定理由
  - 超広視野角
  - 軽量（450g）
- 仕様・スペック
  - 最大測定距離 200m
  - 垂直視野 45 度
  - ビーム本数 64 本の解像度
  - 2,621,440 ポイント/秒
- イメージ

図 3-10 3D LiDAR：Ouster OS1-64

4)　【HW004】RTK-GNSS 測位デバイス：Ichimill

- 選定理由
  - 点群マップ作成に必要なため
  - 従来の衛星測位に比べ、誤差が数センチメートルに抑えられるため
- 仕様・スペック
  - リアルタイムで誤差数センチメートルの測位が可能
  - 準天頂衛星 GNSS から GNSS 信号を受信して測位
- イメージ

図 3-11 Ichimill

5)　【HW005】カメラ：CMS-V43BK

- 選定理由
  - 走行中の周囲の様子撮影記録のため。
- 仕様・スペック
  - センサ：100 万画素 CMOS
  - ビデオ解像度：最大 1280×720
  - フォーカス：手動
  - レンズ：F1.75 f=2.5mm
  - 絞り、明るさ、コントラスト：自動調整
  - 重量：140g
  - サイズ：縦 115mm×横 25mm×高さ 33mm
- イメージ

図 3-12 CMS-V43BK

6)　【HW006】自己位置推定用 PC：System76 Kudu

- 選定理由
  - 自己位置測位のソフトウェアを動かすために Ubuntu PC が必要なため
- 仕様・スペック
  - サイズ：縦 258mm×横 361mm×高さ 29mm
  - OS：Ubuntu 20.04
  - CPU：AMD Ryzen 5900HX
  - メモリ：16GB
  - ストレージ：1TB
- イメージ

図 3-13 System76 Kudu

7) 【HW007】映像配信用 PC：GMK Tech NAB5

- 選定理由
  - 映像配信のソフトウェアを動かすために車載用の小型 PC が必要なため
- 仕様・スペック
  - サイズ：縦 138mm×横 137mm×高さ 99mm
  - OS：Ubuntu 20.04
  - CPU：Intel Celeron
  - メモリ：12GB
  - ストレージ：512GB
- イメージ

図 3-14 GMK Tech NAB5

8)　【HW008】データモニタリング用 PC：GALLERIA XL7C-R45

- 選定理由
  - 点群マップ作成処理に必要なメモリを持つため
  - 同様のスペックを持つ PC が監視用、遠隔操作用で合わせて 2 台必要なため
- 仕様・スペック
  - サイズ：縦 245mm×横 360mm×高さ(最薄部)20mm
  - OS：Windows 11
  - CPU：Intel®i7-13700H
  - メモリ：8GB
  - ストレージ：500GB
- イメージ

図 3-15 GALLERIA XL7C-R45

9)　【HW009】映像監視用 PC：GALLERIA XL7C-R45

- 選定理由
  - 点群マップ作成処理に必要なメモリを持つこと
  - 同様のスペックを持つ PC が監視用、遠隔操作用で合わせて 2 台必要なため。
- 仕様・スペック
  - サイズ：縦 245mm×横 360mm×高さ(最薄部)20mm
  - OS：Windows 11
  - CPU：Intel®i7-13700H
  - メモリ：8GB
  - ストレージ：500GB
- イメージ

図 3-16 GALLERIA XL7C-R45

10)　【HW010】G 空間情報配信用サーバー

- 選定理由
  - 点群マップ生成に必要なデータを配信する唯一のサーバーであるため
- 仕様・スペック
  - https://front.geospatial.jp/

3-2. システム機能

3-2-1. システム機能一覧

表 3-2 利用したシステム機能一覧

赤文字：新規開発

分類	ID	機能名	説明
点群マップ生成	FN001	3D 都市モデルを読み込み	PLATEAU Unity SDK を用いて 3D 都市モデルを仮想空間に読み込む
	FN002	LiDAR 放射パターン作成	仮想空間において点群を放射するための仮想 LiDAR を生成。飛距離や角度、ステップ数などのパラメータを設定
	FN003	車両作成	3D 都市モデル空間において走行するための車両（仮想車両）を生成。車両サイズ、LiDAR を取り付ける位置を設定
	FN004	ルート作成	3D 都市モデル空間において、仮想車両を走行させるためのルートを生成
	FN005	ルート走行	仮想空間内に設定した WayPoint を元に車両がルートを走行する機能
	FN006	点群出力	3D 都市モデル空間において、照射された点群データから点群マップを生成
	FN007	点群間引き	大量の点群データから必要な情報だけを取り出すために、一定の間隔で点をサンプリングし、密度を減らす
自己位置推定	FN101	点群データ読込	LiDAR の点群データを取り込む機能
	FN102	NMEA データ読込	RTK 測位デバイスから NMEA 形式のデータを取り込む機能
	FN103	センサデータ処理	車両に搭載した LiDAR、RTK 測位、IMU からセンサデータの情報を取得と処理を行う
	FN104	自己位置測位	3D LiDAR の点群データと点群マップをもとに点群マッチングを行い、自己位置測位処理を行う
	FN105	データ送信	自己位置測位のデータをデジタルツインビュー、精度グラフ監視へ送信する機能
遠隔監視	FN201	遠隔監視ビューア	遠隔監視者が車両の映像をリアルタイムで確認できるブラウザベースのアプリケーション
	FN202	映像配信	ウェブカメラから映像を取り込み、遠隔監視室へ映像を配信する機能
精度グラフ監視	FN301	データ受信	自己位置測位のデータを受信する機能
	FN302	グラフ描画	受信した自己位置測位データをリアルタイムでグラフに描画する機能

デジタルツインビュー	FN401	データ受信	自己位置測位のデータを受信する機能
	FN402	車両状態の可視化	車両の状態（位置、速度）をデジタルツインビューで可視化
データ配信	FN501	データ受信・配信	車両から自己位置測位のデータを受信し、遠隔監視室のデジタルツインビュー、精度グラフ監視へ送信する機能
リアルマップ生成	FN601	点群マップ生成	3D LiDAR を搭載した車両を走行し、点群マップを生成する機能

3-2-2. 利用したソフトウェア・ライブラリ

表 3-3 利用したソフトウェア・ライブラリ

赤文字：新規利用

ID	項目	内容
SL001	Ubuntu 20.04 LTE	● Autoware で自己位置測位を用いるための開発用 PC にインストールされている OS
SL002	Windows 11	● デジタルツインビュー、遠隔映像監視で用いる開発用 PC にインストールされている OS
SL003	Warpner 3.0.0	● 車両に搭載されたカメラ映像を低遅延で送受信するアプリケーション
SL004	ROS2 Galactic	● ロボットアプリケーション開発のためのミドルウェア
SL005	Autoware 1.11.0	● オープンソースの自動運転ソフトウェアで、センサデータの処理や制御アルゴリズムを提供し、自動車の自律的な走行を実現するためのプラットフォーム
SL006	Mosquitto 5.0	● ブリッシュ/サブスクライブ(Publish/Subscribe)型のメッセージングプロトコルである「MQTT」における、ブローカー（サーバ側）のミドルウェア
SL007	PLATEAU SDK for Unity 1.1.2	● PLATEAU の 3D 都市モデルを Unity で扱うためのライブラリ
SL008	Robotec GPU Lidar 0.16.1	● Unity シーン上で点群をシミュレートするためのライブラリ
SL009	ros2 ouster 0.5.0	● Ouster LiDAR を Autoware で読み込めるようにするための ROS2 ライブラリ
SL010	mqtt bridge 0.1.1	● Autoware から自己位置情報や車両情報を MQTT Broker へ送るための ROS2 ライブラリ
SL011	nmea navsat driver 0.6.1	● RTK-GNSS 測位デバイスから得られる NMEA 形式のデータを Autoware で読み込めるようにするための ROS2 ライブラリ
SL012	CloudCompare 2.13	● PointCloud データを読み込み、編集することができるオープンソースのソフトウェア
SL013	plotly 5.18.0	● Python ベースのグラフ描画ライブラリ

SL014	Unity 2021.3.9	● ゲームやアプリを開発するための統合開発環境
SL015	Warpner 2.1.1	● ウェブカメラの映像を取り込み、遠隔監視室に配信するアプリケーション
SL016	Warpner Viewer 3.0.1	● 車両に搭載したウェブカメラの映像をリアルタイムで受信するブラウザベースのアプリケーション
SL017	DigitalTwinViewer 1.0.0	● 車両に搭載した PC から、自己位置推定の情報を受信し、仮想空間内で表示する機能
SL018	paho-mqtt 2.0.0	● Autoware から自己位置情報や車両情報を MQTT Broker から受信するための Python ライブラリ
SL019	UnityEditor クラス	● Unity のエディタを拡張する機能。Unity 2021.3.9 のクラスの中に含まれる
SL020	DOTWeen 1.2	● Unity オブジェクトに移動系のアニメーションを付与するライブラリ
SL021	Autoware-sensing	● 3D LiDAR や GNSS、IMU のデータを受け取り、Autoware 内で利用するための処理を行うライブラリ。本システムでは 3D LiDAR のデータのみを利用
SL022	Autoware-localization	● Autoware-sensing から受け取ったセンサデータ、また点群マップを用いて自己位置測位を行うライブラリ
SL023	lidarslam_ros2 0.1.0	● NDT スキャンマッチングを利用した ROS2 の SLAM 用ライブラリ

3-2-3. 開発機能の詳細要件

開発機能の詳細要件を記す。なお、本業務において新規開発した要素（機能名）を赤字で示す。

1)　点群マップ生成

1. 【FN001】3D 都市モデル読込

- 機能概要
  - Unity シーン内に 3D 都市モデルを読み込む機能
  - PLATEAU SDK for Unity を用いて、CityGML 形式の 3D 都市モデルを Unity シーン内に出力
- フローチャート

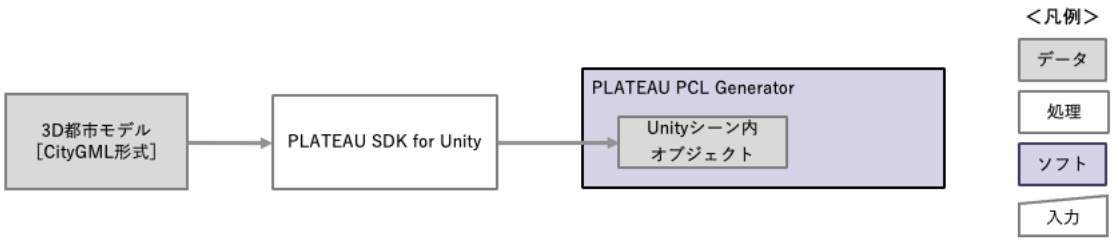


図 3-20 3D 都市モデル読込のフローチャート

- データ仕様
  - 入力
    - 3D 都市モデル読み込み設定（建築物、道路、都市設備、植生、地形）
      - 内容
        - Unity シーン内に 3D 都市モデルを読み込む
      - 形式
        - CityGML ファイル
      - データ詳細
        - 利用した 3D 都市モデル【DT001～DT131】を参照
  - 出力
    - なし
- 機能詳細
  - 3D 都市モデル読込
    - 処理内容
      - PLATEAU SDK for Unity を用いて、3D 都市モデルを読み込む
      - PLATEAU SDK for Unity で読み込む地域、LOD のレベル、テクスチャの有無、Mesh Colider の有無などのパラメータを設定
    - 利用するライブラリ

- PLATEAU SDK for Unity（ソフトウェア・ライブラリ【SL007】）

✧ 利用するアルゴリズム

- なし

2. 【FN002】LiDAR 放射パターン作成

- 機能概要
  - ➢ 仮想 LiDAR のパラメータを設定する機能
- フローチャート

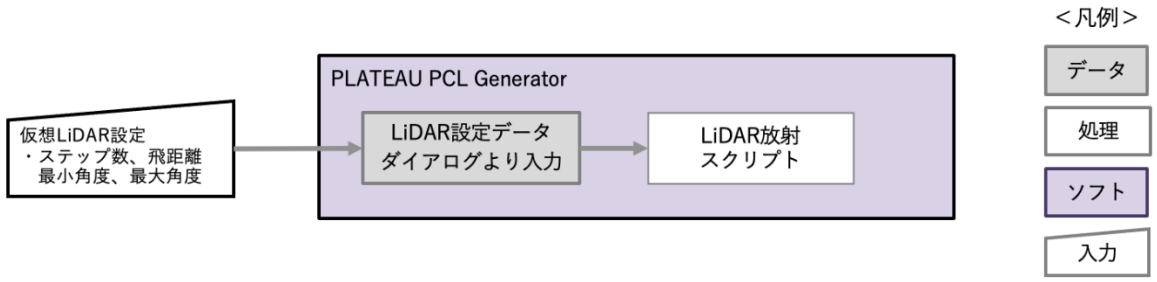


図 3-21 LiDAR 放射パターン作成のフローチャート

- データ仕様
  - ➢ 入力
    - ✧ 仮想 LiDAR 設定
      - 内容
        - ➢ 仮想 LiDAR の設定を Unity 上（PLATEAU PCL Generator ダイアログ）で設定
      - 形式
        - ➢ Unity シーン内ダイアログ上で値の入力
      - データ詳細
        - ➢ 内部連携インタフェース【IF201】を参照
  - ➢ 出力
    - ✧ なし
- 機能詳細
  - ➢ LiDAR 放射パターン作成
    - ✧ 処理内容
      - 仮想 LiDAR の各種パラメータを入力
      - Unity シーン内の LiDAR オブジェクトに形状、LiDAR 照射のパラメータを反映
    - ✧ 利用するライブラリ
      - Robotec GPU Lidar（ソフトウェア・ライブラリ【SL008】）

- ✧ 利用するアルゴリズム
  - なし

3. 【FN003】車両作成

- 機能概要
  - 仮想空間（3D 都市モデル空間）を走行する仮想車両を作成し、仮想 LiDAR 取り付け位置を調整する機能
- フローチャート

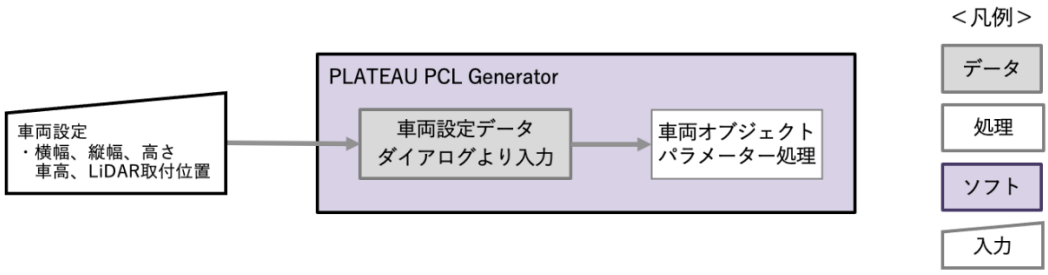


図 3-22 車両作成のフローチャート

- データ仕様
  - 入力
    - ✧ 車両作成
      - 内容
        - 仮想車両の設定を Unity 上（PLATEAU PCL Generator ダイアログ）で設定
      - 形式
        - Unity シーン内ダイアログ上で値の入力
      - データ詳細
        - 内部連携インタフェース【IF202】を参照
  - 出力
    - ✧ なし
- 機能詳細
  - 車両作成
    - ✧ 処理内容
      - 仮想車両の各種パラメータを設定するダイアログを表示
      - Unity シーン内の車両オブジェクト、LiDAR オブジェクトに形状や位置のパラメータを反映
      -
    - ✧ 利用するライブラリ
      - UnityEditor クラス（利用したソフトウェア・ライブラリ【SL019】）を参照
    - ✧ 利用するアルゴリズム

- なし

4. 【FN004】ルート作成

- 機能概要
  - 仮想空間（3D 都市モデル）内を仮想車両が走行する際のルートを生成する機能
- フローチャート

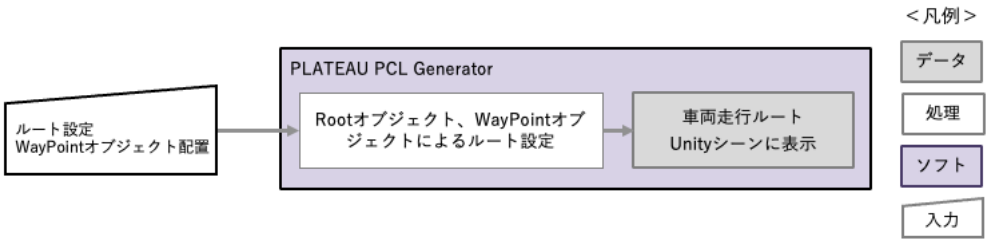


図 3-23 ルート作成のフローチャート

- データ仕様
  - 入力
    - WayPoint 設定
      - 内容
        - 仮想空間（3D 都市モデル）内を仮想車両が走行する際のルートを Unity シーン上で設定
        - Unity シーンを操作し、キーボード入力で WayPoint オブジェクトを配置（画面、入力イメージに関しては画面仕様詳細の【SC003】【SC004】を参照）
        - Root オブジェクトに対し、WayPoint 子オブジェクトを追加
      - 形式
        - Unity シーン内にてキーボードで入力
      - データ詳細
        - Unity シーン内にて Mesh Collider が設定された道路モデル上をマウスで選択、キーボード操作により WayPoint 子オブジェクトを作成
  - 出力
    - WayPoint オブジェクト
      - 内容
        - Unity シーン上に配置された WayPoint オブジェクト
        - Root オブジェクトを親オブジェクトとし、キーボード入力により道路モデル上に位置情報が付与された WayPoint 子オブジェクトを追加
      - 形式
        - Unity オブジェクト
      - データ詳細
        - 内部連携インタフェース【IF203】を参照

- 機能詳細
  - ルート作成
    - 処理内容
      - Unity シーン上に WayPoint オブジェクトを配置し、仮想車両が走行するルートを設定
    - 利用するライブラリ
      - UnityEditor クラス（利用したソフトウェア・ライブラリ【SL019】を参照）
      - DOTWeen（利用したソフトウェア・ライブラリ【SL020】）を参照）
    - 利用するアルゴリズム
      - なし

5. 【FN005】ルート走行

- 機能概要
  - 仮想空間内に設定した WayPoint を元に車両がルートを走行する機能
- フローチャート
  - 【FN006】に記載
- データ仕様
  - 入力
    - なし（【FN004】にてルート作成を入力）
  - 出力
    - なし（【FN005】にて点群マップ出力）
- 機能詳細
  - ルート走行
    - 処理内容
      - 【FN004】にて Root オブジェクトを親オブジェクトとした WayPoint 子オブジェクトをもとに仮想空間内のルートを走行
      - Unity シーンの Play ボタンを押すと、上記で設定したルートを車両が走行するアニメーションを開始（走行の様子は画面仕様詳細の【SC005】を参照）
    - 利用するライブラリ
      - DOTWeen（利用したソフトウェア・ライブラリ【SL020】）を参照）
    - 利用するアルゴリズム
      - なし

6. 【FN006】点群出力

- 機能概要
  - 仮想空間（3D 都市モデル空間）で、照射された点群データから点群マップを生成する機能
- フローチャート

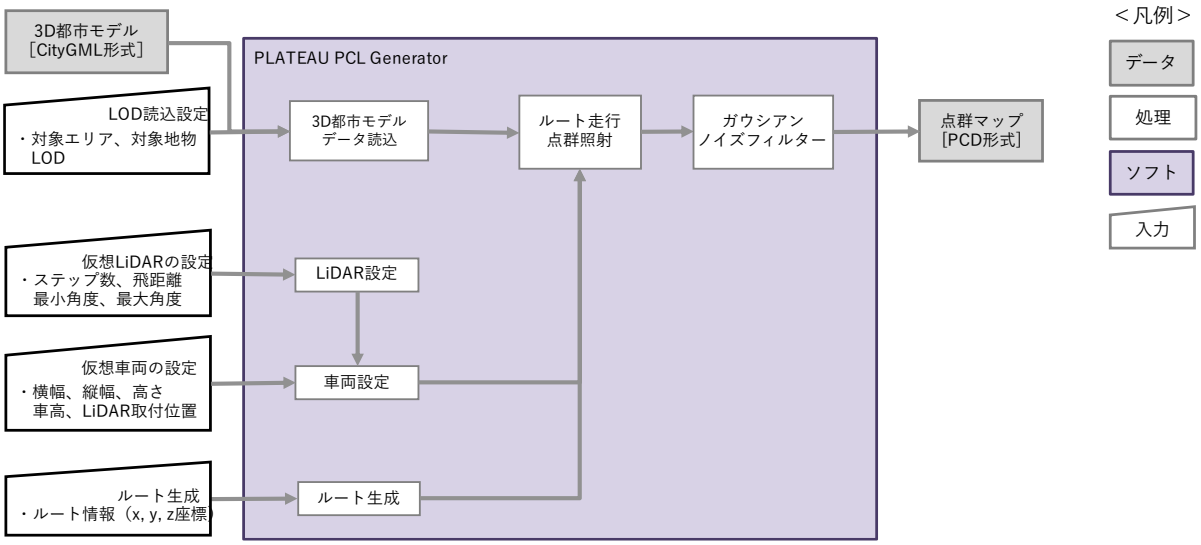


図 3-24 点群出力のフローチャート

- データ仕様
  - 入力
    - なし（システム機能【FN001】～【FN004】で入力）
  - 出力
    - 点群マップ
      - 内容
        - 仮想空間（3D 都市モデル空間）で、照射された点群データから生成された点群マップ
      - 形式
        - PCD 形式
      - データ詳細
        - ファイル出力インタフェース【IF101】を参照

- 機能詳細
  - 点群出力
    - 処理内容
      - 仮想 LiDAR、車両の各種パラメータを設定（システム機能【FN002】【FN003】）
      - Unity シーン上に Root オブジェクトを親オブジェクトとした、WayPoint 子オブジェクトを配置し、仮想車両が走行するルートを設定（システム機能【FN004】）
      - Play ボタンを押すことで、車両がルート走行を開始（システム機能【FN005】）

- 走行しながら点群を照射し、走行終了後、点群マップを出力

✧ 利用するライブラリ

- Robotec GPU Lidar（ソフトウェア・ライブラリ【SL008】）

✧ 利用するアルゴリズム

- ガウシアンノイズフィルタ（アルゴリズム【AL002】）

7. 【FN007】点群間引き

- 機能概要
  - CloudCompare を用いて、点群データのファイルサイズを減らすために、一定の間隔で点を間引き処理（サンプリング）し、密度を減らす
  - 間引き処理では、ランダムサンプリングを用い、点群サイズが 5GB（Autoware の読み込みで推奨される点群サイズ）以下になるように調整を行う
- フローチャート

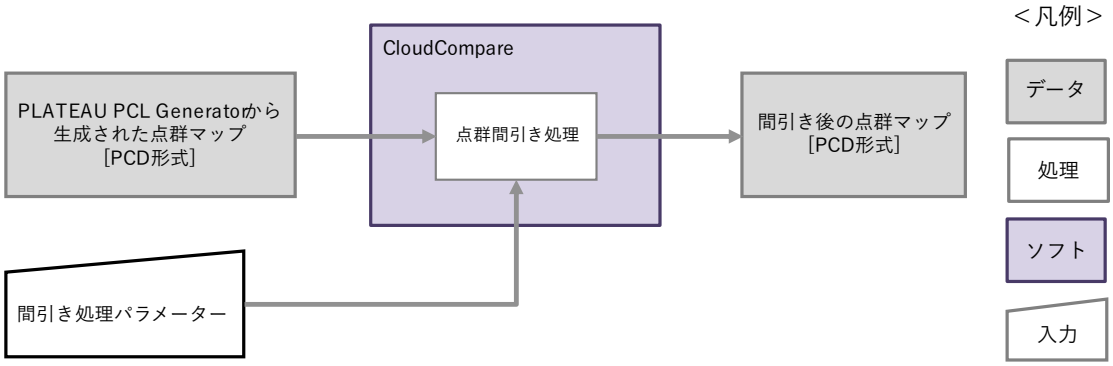


図 3-25 点群間引きのフローチャート

- データ仕様
  - 入力
    - ✧ 間引き処理パラメータ
      - 内容
        - PLATEAU PCL Generator から生成された点群マップを Cloud Compare で間引くためのパラメータを入力
      - 形式
        - CloudCompare の Cloud sub sampling ダイアログにて値を入力
      - データ詳細
        - 内部連携インタフェース【IF101】を参照
    - ✧
  - 出力
    - ✧ 間引き後の点群マップ
      - 内容
        - CloudCompare により、点群間引き処理された点群マップ
      - 形式

        - PCD 形式
    - データ詳細
        - ファイル出力インタフェース【IF101】を参照
- 機能詳細
    - 点群間引き
        - 処理内容
            - PLATEAU PCL Generator により出力された点群マップの間引き処理を行う
            - 間引き処理にはランダムサンプリングを用いる
            - 間引き処理後の点群サイズが 5GB 以下になるように、ランダムサンプリングの点群数のパラメータを調整する
            - 間引き処理後の点群マップは PCD 形式で出力される
        - 利用するライブラリ
            - CloudCompare（ソフトウェア・ライブラリ【SL012】を参照）
        - 利用するアルゴリズム
            - なし

8. 【FN101】点群データ読込

- 機能概要
    - LiDAR の点群データを取り込む機能
- データ仕様
    - 入力
        - LiDAR データ
            - 内容
                - 3D LiDAR から得られる点群データ
            - 形式
                - PCD 形式
            - データ詳細
                - 内部連携インタフェース【IF211】を参照
    - 出力
        - センサデータ（ROS メッセージ形式）
            - 内容
                - LiDAR から得られたセンサデータを ROS メッセージに変換し、Autoware の自己位置推定処理で使用する
            - 形式
                - ROS メッセージ
            - データ詳細
                - 内部連携インタフェース【IF212】を参照
- 機能詳細

- LiDAR の点群データを取り込む機能
  - 処理内容
    - 3D LiDAR から取得した PCD 形式の点群データを ROS メッセージに変換する機能
  - 利用するライブラリ
    - ros2-ouster（ソフトウェア・ライブラリ【SL009 を参照】
  - 利用するアルゴリズム
    - なし

9. 【FN102】NMEA データ読込

- 機能概要
  - RTK 測位デバイスから NMEA 形式のデータを取り込む機能
- データ仕様
  - 入力
    - NMEA データ
      - 内容
        - RTK 測位デバイスから得られるデータ
      - 形式
        - NMEA 形式
      - データ詳細
        - 内部連携インタフェース【IF221】を参照
  - 出力
    - NMEA データ（ROS メッセージ形式）
      - 内容
        - RTK 測位デバイスから得られた NMEA 形式のデータを Autoware の自己位置推定処理で使用するために ROS メッセージに変換
      - 形式
        - ROS メッセージ
      - データ詳細
        - 内部連携インタフェース【IF222】を参照
- 機能詳細
  - NMEA データ読込
    - 処理内容
      - RTK 測位デバイスから NMEA 形式のデータを受け取る
      - 取得した NMEA 形式のデータを ROS メッセージに変換
    - 利用するライブラリ
      - nmea_navsat_driver（ソフトウェア・ライブラリ【SL011】を参照）
    - 利用するアルゴリズム
      - なし

10. 【FN103】センサデータ処理

- 機能概要
  - 車両に搭載した LiDAR センサからセンサデータの取得と処理を行う
- フローチャート

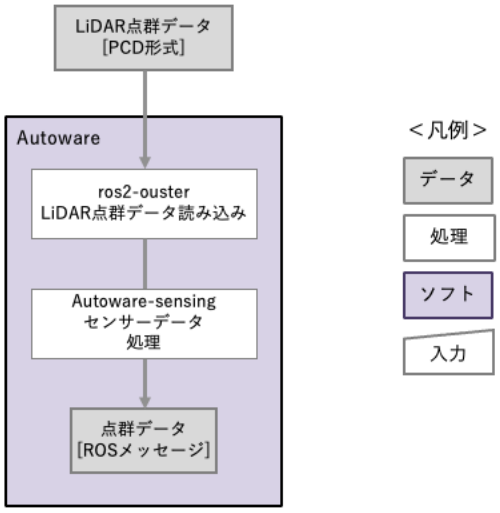


図 3-26 センサデータ処理のフローチャート

- データ仕様
  - 入力
    - LiDAR 点群データ
      - 内容
        - ros2 ouster から得られる、3D LiDAR から得られる点群データを ROS メッセージに変換したデータ
      - 形式
        - ROS メッセージ
      - データ詳細
        - 内部連携インタフェース【IF212】を参照
  - 出力
    - センサデータ（ROS メッセージ形式）
      - 内容
        - ros2 ouster から得られる点群データを元に、ノイズ除去を行い、ROS メッセージ（sensor_msgs/PointCloud2 形式）を出力。入力とデータは変わるが、データ形式は同一となる
      - 形式
        - ROS メッセージ
      - データ詳細
        - 内部連携インタフェース【IF212】を参照

- 機能詳細
  - センサデータ処理
    - 処理内容
      - LiDAR から取得したデータを ROS メッセージで受け取る
      - 車体に当たる点群の領域を選択し、点群データのマスキング処理を行う
      - 点群のノイズ除去を行う
    - 利用するライブラリ
      - Autoware-sensing（ソフトウェア・ライブラリ【SL021】を参照）
      - ros2-ouster（ソフトウェア・ライブラリ【SL009】を参照）
    - 利用するアルゴリズム
      - なし

11. 【FN104】自己位置測位

- 機能概要
  - 3D LiDAR の点群データと点群マップのマッチングを行い、自己位置測位を行う機能
- フローチャート

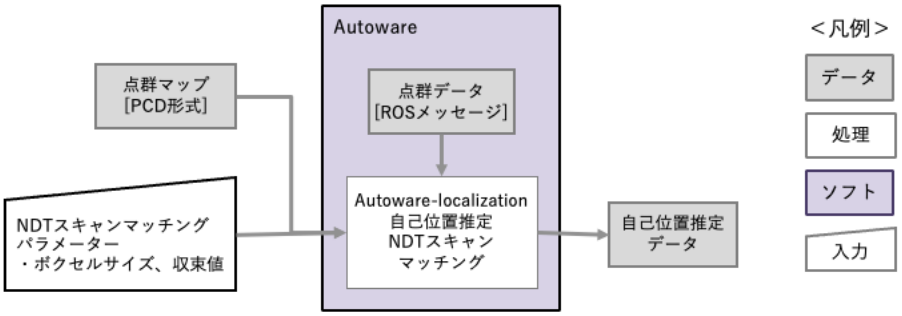


図 3-27 自己位置測位のフローチャート

- データ仕様
  - 入力
    - センサデータ
      - 内容
        - Autoware-sensing から出力された点群データの ROS メッセージ
      - 形式
        - ROS メッセージ
      - データ詳細
        - 内部連携インタフェース【IF212】を参照
    - 点群マップ

- 内容
  - PLATEAU PCL Generator から出力され、CloudCompare により間引き処理された点群マップ
- 形式
  - PCD 形式
- データ詳細
  - ファイル入力インタフェース【IF001】を参照

✧ NDT スキャンマッチングパラメータ

- 内容
  - Autoware-localization 内で処理される NDT スキャンマッチングのパラメータ
- 形式
  - YAML ファイルにて入力
- データ詳細
  - 内部連携インタフェース【IF213】を参照

➢ 出力

✧ 自己位置精度データ（ROS メッセージ形式）

- 内容
  - 自己位置測位の精度割合を表したデータ
- 形式
  - ROS メッセージ
- データ詳細
  - 内部連携インタフェース【IF205】を参照

- 機能詳細

➢ 自己位置測位

✧ 処理内容

- 点群マップと、LiDAR で取得した点群データとマッチングを行う

✧ 利用するライブラリ

- Autoware-localization（ソフトウェア・ライブラリ【SL022】を参照）

✧ 利用するアルゴリズム

- NDT スキャンマッチング（アルゴリズム【AL001】を参照）

12. 【FN105】データ送信

- 機能概要
  - 自己位置測位のデータをデジタルツインビュー、精度グラフ監視へ送信する機能
- データ仕様
  - 入力
    - 自己位置測位データ
      - 内容
        - 自己位置測位データの ROS メッセージ
      - 形式
        - ROS メッセージ
      - データ詳細
        - 内部連携インタフェース【IF223】を参照
    - MQTT メッセージ
      - 内容
        - 自己位置測位データが含まれている JSON 形式のメッセージ
      - 形式
        - JSON 形式
      - データ詳細
        - 内部連携インタフェース【IF301】を参照
- 機能詳細
  - データ送信
    - 処理内容
      - 自己位置測位のデータをデジタルツインビュー、精度グラフ監視へ送信する
    - 利用するライブラリ
      - mqtt_bridge（ソフトウェア・ライブラリ【SL010】を参照）
    - 利用するアルゴリズム
      - なし

13. 【FN201】映像配信

- 機能概要
  - ウェブカメラから映像を取り込み、遠隔監視室へ映像を配信する機能
- フローチャート
  - 【FN202】に記載
- データ仕様
  - 入力
    - カメラ映像
      - 内容

- 車載カメラから入力された映像
- 形式
  - UVC 形式
- データ詳細
  - 内部連携インタフェース【IF231】を参照

- 出力
  - リアルタイム映像
    - 内容
      - 車載カメラの映像をブラウザベースのアプリケーションへ配信
    - 形式
      - AV1 形式
    - データ詳細
      - 内部連携インタフェース【IF232】を参照

- 機能詳細
  - 映像配信
    - 処理内容
      - 車載に搭載したカメラ映像を LTE 経由で配信
    - 利用するライブラリ
      - Warpner（ソフトウェア・ライブラリ【SL003】を参照）
    - 利用するアルゴリズム
      - なし

14. 【FN202】遠隔監視ビューア

- 機能概要
  - 遠隔監視者が車両の映像をリアルタイムで確認するための機能
- フローチャート

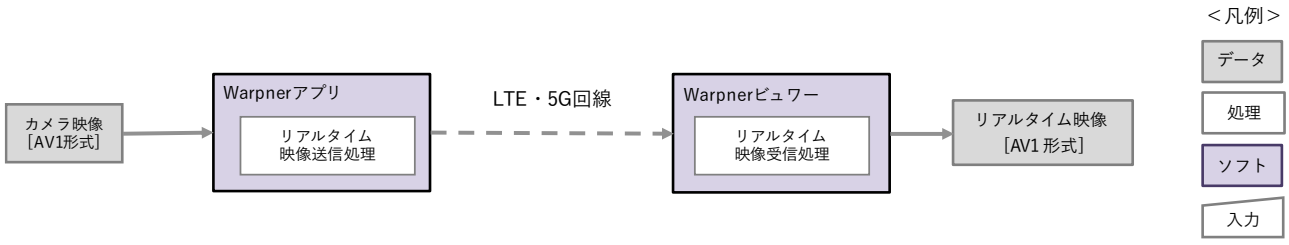


図 3-28 遠隔監視ビュワーのフローチャート

- データ仕様
  - 入力
    - カメラ映像
      - 内容

- 車載カメラから入力された映像
- 形式
  - AV1 形式
- データ詳細
  - 内部連携インタフェース【IF232】を参照

- 出力
  - なし

- 機能詳細
  - カメラ遠隔監視
    - 処理内容
      - 車載に搭載したカメラ映像を LTE 経由で送信し、ブラウザベースのアプリケーションで映像監視する
    - 利用するライブラリ
      - Warpner（ソフトウェア・ライブラリ【SL003】を参照）
    - 利用するアルゴリズム
      - なし

15. 【FN301】データ受信

- 機能概要
  - 自己位置測位のデータを受信する機能
- フローチャート
  - 【FN302】に記載
- データ仕様
  - 入力
    - JSON データ
      - 内容
        - Autoware から送信される JSON 形式の MQTT メッセージ
      - 形式
        - JSON 形式
      - データ詳細
        - 外部連携インタフェース【IF301】を参照
  - 出力
    - なし
- 機能詳細
  - データ受信
    - 処理内容
      - MQTT 形式の自己位置測位データを受信する機能
    - 利用するライブラリ

- paho-mqtt（ソフトウェア・ライブラリ【SL018】を参照）

✧ 利用するアルゴリズム

- なし

16. 【FN302】グラフ描画

- 機能概要
  - ➢ 自己位置測位の精度割合情報をリアルタイムで描画する機能
- フローチャート

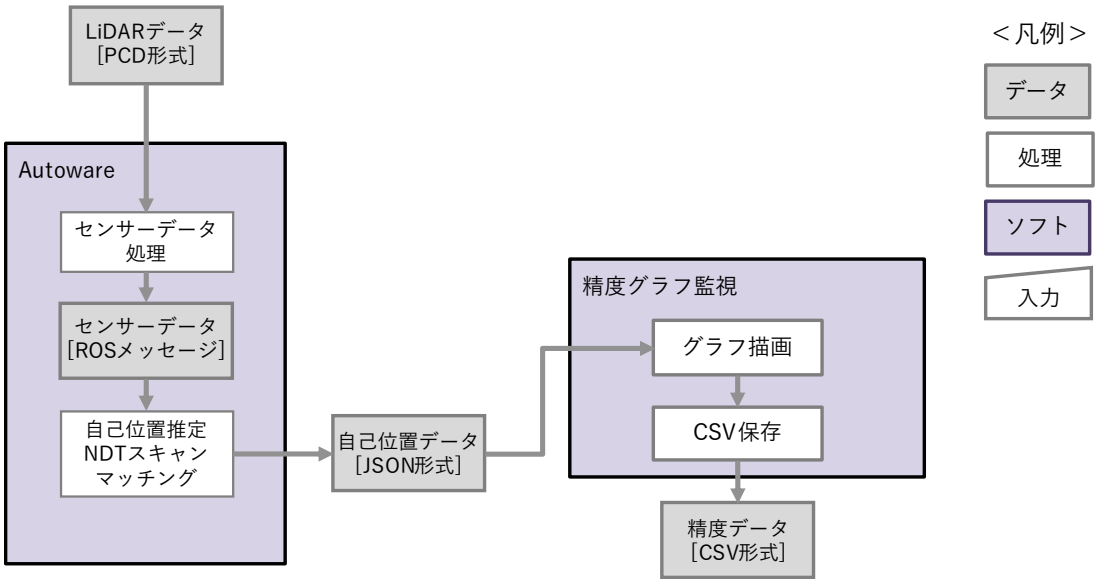


図 3-29 グラフ描画のフローチャート

- データ仕様
  - ➢ 入力
    - ✧ MQTT で受信する自己位置データ
      - 内容
        - ➢ MQTT で受信する JSON 形式の自己位置データ
      - 形式
        - ➢ JSON 形式
      - データ詳細
        - ➢ 外部連携インタフェース【IF301】を参照
    - ✧
  - ➢ 出力
    - ✧ 精度割合データ
      - 内容
        - ➢ 自己位置測位の精度割合データ
      - 形式
        - ➢ CSV 形式
      - データ詳細

- ファイル出力インタフェース【IF102】を参照

- 機能詳細
  - グラフ描画
    - 処理内容
      - 自己位置測位の精度割合データをグラフに表示する
    - 利用するライブラリ
      - plotly（ソフトウェア・ライブラリ【SL013】を参照）
    - 利用するアルゴリズム
      - なし

17. 【FN401】データ受信

- 機能概要
  - 自己位置測位のデータを受信する機能
- フローチャート
  - 【FN402】に記載
- データ仕様
  - 入力
    - JSON データ
      - 内容
        - Autoware から送信される JSON 形式の MQTT メッセージ
      - 形式
        - JSON 形式
      - データ詳細
        - 外部連携インタフェース【IF301】を参照
  - 出力
    - なし
- 機能詳細
  - データ受信
    - 処理内容
      - MQTT 形式の自己位置測位データを受信する機能
    - 利用するライブラリ
      - paho-mqtt（ソフトウェア・ライブラリ【SL018】を参照）
    - 利用するアルゴリズム
      - なし

18. 【FN402】車両状態の可視化

- 機能概要

- 車両の状態をデジタルツインビューで可視化する機能

- フローチャート

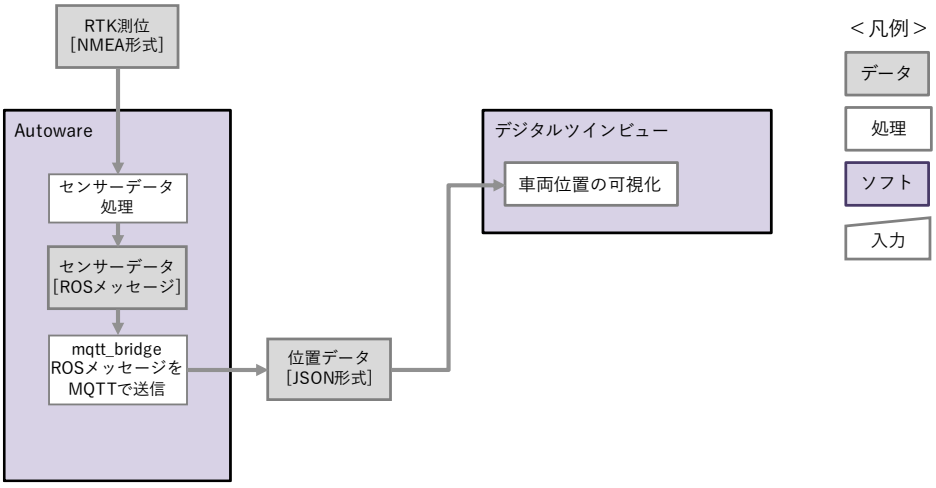


図 3-30 車両状態の可視化のフローチャート

- データ仕様
  - 入力
    - 位置データ
      - 内容
        - RTK-GNSS 測位デバイスのデータを ROS メッセージに変換後、MQTT を用いてデジタルツインビューへ送信するために JSON に変換し出力
      - 形式
        - JSON 形式
      - データ詳細
        - 外部連携インタフェース【IF301】を参照
  - 出力
    - なし

- 機能詳細
  - 車両状態の可視化
    - 処理内容
      - RTK 測位デバイスから得られた位置情報を基にデジタルツインビューに表示
    - 利用するライブラリ
      - mqtt_bridge（ソフトウェア・ライブラリ【SL010】を参照）
    - 利用するアルゴリズム
      - なし

19. 【FN501】データ受信・配信

- 機能概要

- 車両から自己位置測位のデータを受信し、遠隔監視室のデジタルツインビュー、精度グラフ監視へ送信する機能

- データ仕様
  - 入力
    - 自己位置測位データ
      - 内容
        - Autoware から出力された自己位置測位データを JSON 形式に変換し MQTT メッセージで配信
      - 形式
        - JSON 形式
      - データ詳細
        - 外部連携インタフェース【IF301】を参照
    -
  - 出力
    - 自己位置測位データ
      - 内容
        - 自己位置測位データを精度監視グラフ、デジタルツインビューに MQTT メッセージで配信
      - 形式
        - JSON 形式
      - データ詳細
        - 外部連携インタフェース【IF301】を参照

- 機能詳細
  - データ受信・配信
    - 処理内容
      - MQTT メッセージを受信、配信するためのサーバー側のミドルウェア
    - 利用するライブラリ
      - Mosquitto（ソフトウェア・ライブラリ【SL006】を参照）
    - 利用するアルゴリズム
      - なし

20. 【FN601】リアルマップ生成

- 機能概要
  - 3D LiDAR の点群データから PCD 形式の点群マップを生成する機能
  - 3D LiDAR を車両に搭載し車両を走行、点群の照射を行うことで点群マップを生成
  - PCLG 点群マップと精度を比較するために、リアルマップを生成
- フローチャート

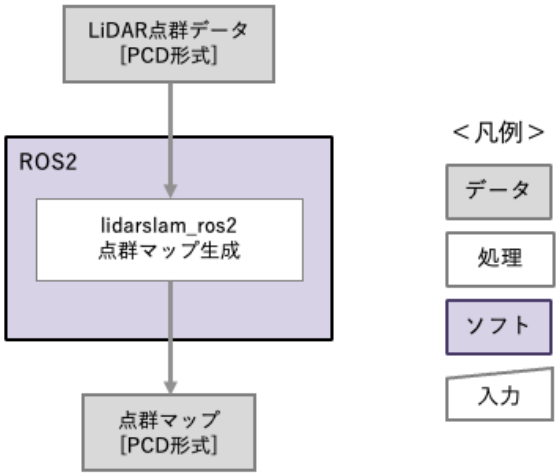


図 3-31 リアルマップ生成のフローチャート

- データ仕様
  - 入力
    - LiDAR 点群データ
      - 内容
        - 3D LiDAR から得られる点群データ
      - 形式
        - PCD 形式
      - データ詳細
        - 内部連携インタフェース【IF211】を参照
  - 出力
    - 点群マップ
      - 内容
        - lidarslam_ros2 の SLAM（自己位置推定、環境地図作成）により生成された点群マップ
      - 形式
        - PCD 形式
      - データ詳細
        - ファイル出力インタフェース【IF101】を参照
- 機能詳細
  - 点群マップ生成
    - 処理内容
      - 3D LiDAR から受け取ったデータを元に NDT スキャンマッチングで自己位置測位を行い、車両を走行させることで環境マップを生成
    - 利用するライブラリ
      - lidarslam_ros2（ソフトウェア・ライブラリ【SL023】を参照）

✧ 利用するアルゴリズム

- NDT スキャンマッチング（アルゴリズム【AL001】を参照）

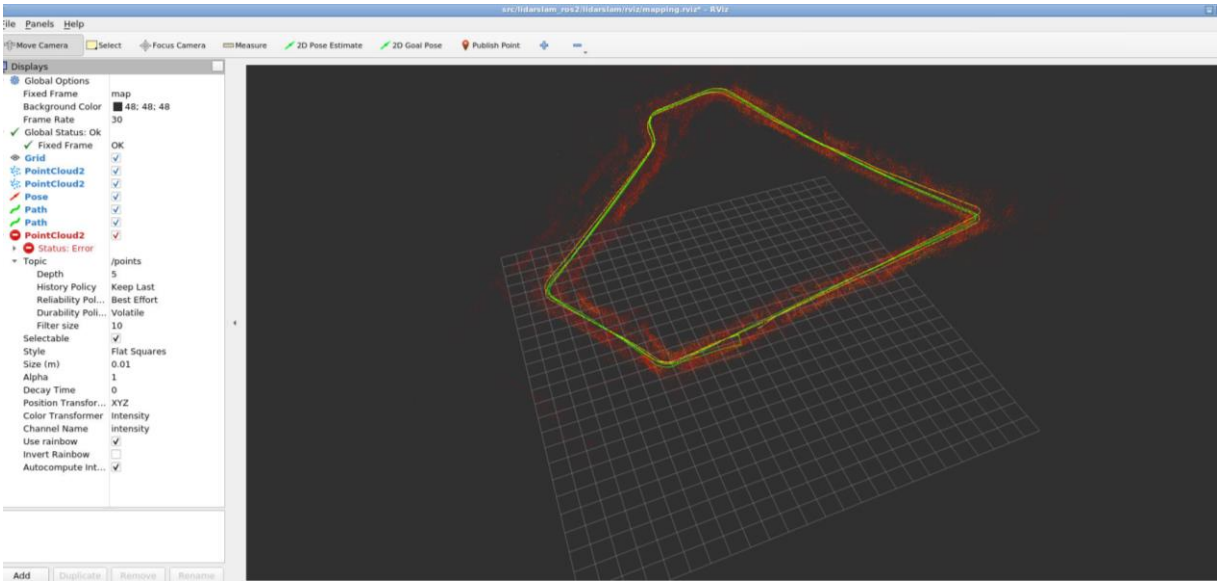


図 3-32 lidarslam_ros2 用いて点群マップを生成する様子

3-3. アルゴリズム

3-3-1. 利用したアルゴリズム

表 3-10 利用したアルゴリズム一覧

ID	アルゴリズムを利用した機能	名称	説明	選定理由
AL001	FN104, FN601	NDT スキャンマッチング	NDT スキャンマッチングは、レーザースキャンデータの点群を、複数の正規分布（ガウス分布）で近似し、 事前に入力した点群マップとのマッチングを区切られた単位毎に行う。これにより、効率的なマッチングが可能となり、車両の姿勢（位置と方向）を推定する	● ノイズに対する堅牢性、計算量の効率性
AL002	FN006	ガウシアンノイズフィルタ	点群内の各点に対し、点を中心とした周囲の点の空間的な分布をガウス分布で近似し、新しい位置を決定する。その処理を反復することでノイズによる位置のばらつきが減少し、点群データが滑らかになる	● 点群の平滑化 ● 計算の効率性

1) 【AL001】NDT スキャンマッチング

- 概要
  - NDT スキャンマッチングとは、点群マップをボクセル（ボリュームとピクセルを組み合わせた 3 次元空間を表現するための最小単位）ごとに区切り正規分布を計算し、3D LiDAR から得られた点群データとマッチングを行う手法。
- 処理フロー
  - アルゴリズムの流れとして、1. 点群マップを一定の大きさの格子状に分割。2. 格子内の平均と分散を算出。 3. 3DLiDAR による入力点群に対応する要素を求める。 4. 評価値を計算する。 5. ニュートン法による入力点群の座標値の更新。 6. 収束するまで、上記の工程を繰り返す。という処理フローとなる。
- NDT スキャンマッチングと ICP スキャンマッチングとの比較
  - スキャンマッチングには、NDT スキャンマッチングの他に ICP スキャンマッチングという手法がある。ICP スキャンマッチングは地図データ郡群とセンサデータ群の 2 つの点群間の最近傍点を対応点とし、対応関係にある点間の距離計が最短になる箇所をマッチングポイントとする方法である。NDT スキャンマッチングと ICP スキャンマッチングを比較した表を下記に示す。

表 3-11 NDT スキャンマッチング、ICP スキャンマッチング比較表

	NDT スキャンマッチング	ICP スキャンマッチング
概要	環境地図をボクセルごとに区切り正規分布を計算し、3 次元 LiDAR から得られた点群データとマッチングを行う手法	地図データ郡群とセンサデータ群の 2 つの点群間の最近傍点を対応点とする。対応関係にある点間の距離計が最短になる箇所をマッチングポイントとする方法
計算速度	高速 点群データが多い場合でも有用 点群データを正規分布に基づいたヒストグラムに変換することで、高速な位置姿勢推定を実現できる。 リアルタイム性の要求が高いロボットや自動運転車などのアプリケーションに適任	高速 但し、点群データが多いと低速になる
ノイズ	強い 点群データに含まれるノイズや不良点に対しても比較的頑健な処理を行うことが可能 実際のセンサから取得した点群データにも適用可能	弱い 不良点があると計算時間が長くなる
計算量	多い 正規分布を用いたフィッティングにより大域的な最適解を見つけ易く、異なる観測角度から得られる点群データのマッチングにも有効 計算量が多いためマシンパワーが必要となる	少ない 点と点のマッチングにより、計算量は低いが、点群量や不良点が増加するにつれて計算量が増大
精度割合	高い 但し、初期値に依存 地図データと比較して、点群データのマッチングの誤差を減らすことができる。 高精度な位置姿勢推定を実現可能	高い 但し、不良点があると下がる
選考理由	車両の自己位置測位によるスキャンマッチングでは、一般的に点群マップのサイズが大きくなるという特徴がある。よって大きな点群マップの場合でも計算速度が早い NDT スキャンマッチングが適している。但し計算量は増えるので、相応のマシンパワーをもった PC を利用する必要がある	

- イメージ

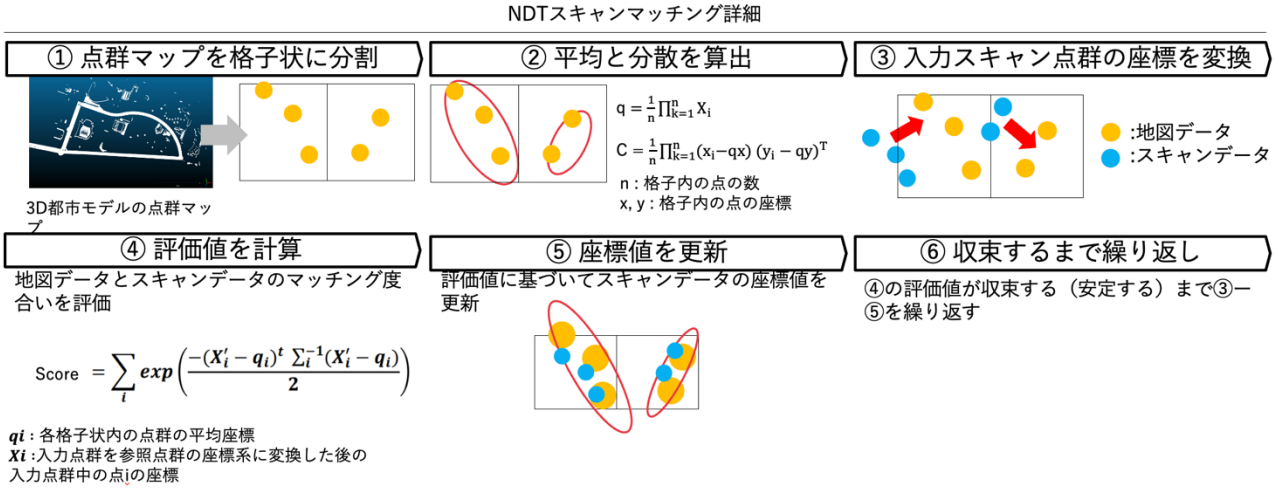


図 3-40 スキャンマッチングのイメージ

- NDT スキャンマッチングの実装
  - 本システムでは、Autoware-localization に実装されている NDT スキャンマッチングを利用した自己位置推定を行う。また、自己位置測位処理（システム機能【FN104】）にて NDT スキャンマッチングを計算する。下記にそのフローを記述する
  - 1. PCD 形式の点群マップを読み込む
  - 2. ROS メッセージに変換された点群データを Autoware-sensing から受け取る
  - 3. Autoware-localization に NDT スキャンマッチングパラメータ（内部連携インタフェース【IF213】）を入力し、自己位置測位を計算する
- NDT スキャンマッチングパラメータ
  - NDT スキャンマッチングパラメータ（内部連携インタフェース【IF213】）の各項目に関して概要を記述する
  - trans_epsilion：NDT スキャンマッチングにおいて変換の収束を判断するための許容誤差を指定するパラメータ。スキャンマッチングで得られる連続した姿勢変換の差の許容誤差を表す。trans_epsilon の値が小さいほど NDT スキャンマッチングは精度の高い解に収束するが、計算コストが増加する。逆に trans_epsilon の値が小さい場合、NDT スキャンマッチングは早く収束するが、精度が低下する可能性がある
  - step_size：線形探索において使用される最大ステップ長を指定するパラメータ。NDT スキャンマッチングでは点群マップと 3D LiDAR から入力された点群データを比較して車両の姿勢を最適化する。その際にボクセル化された点群マップと 3D LiDAR の点群データを比較し、現在の位置から最適な方向に進む際に一度に進むことが出来る最大の距離を制限する。step_size が大きいほどアルゴリズムは遠くまで探索を進めてしまい、収束に時間がかかる。逆に小さすぎると収束までに多くの反復が必要となる
  - voxel_grid_resolution：ボクセルグリッドの解像度。NDT スキャンマッチングにおいて使用される点群マップをグリッド上に分割する際の解像度を指定するパラメータ。NDT スキャンマッチングで

は 3D LiDAR から得られる点群データを処理して点群マップと照合する際に、データを効率的に処理するために空間をグリッド上に分割する。voxel_grid_resolution が高いほど空間が細かく分割され、精度の高いマッチングが可能となるが、計算コストが増加する。voxel_grid_resolution が低い場合、計算コストは減少するが、精度も下がってしまう

- max_iterations：NDT スキャンマッチングにおいてアルゴリズムが最適な解に収束するまでの最大反復回数を指定するパラメータ。このパラメータはアルゴリズムが収束するまで繰り返し処理を行う回数を制御する。mac_iterations の値が小さいとアルゴリズムが収束する前に反復が終了する可能性があり、逆に大きすぎると計算時間が長くなるが、収束することが保証される
- num_threads：プログラムが並列処理を行う際に使用するスレッドの数を指定するパラメータ

2) 【AL002】ガウシアンノイズフィルタ

- 概要
  - ガウシアンノイズフィルタ（又はガウスフィルタ）は、主にデジタル画像をスムージング（平滑化）するためによく使われる手法の一つであるが、今回はガウシアンノイズフィルタを仮想 LiDAR のシミュレーターに応用する。仮想 LiDAR から得られる点群データには、1. 到達点による角度のノイズ、2. 距離によるノイズの 2 つのノイズが挙げられる。ガウシアンノイズフィルタは、上記のノイズを平滑化するために、各点群データをガウス分布に基づき点群位置を修正する。
- ガウス関数による重み計算
  - ガウシアンノイズフィルタは、以下の数式で重みを計算することができる。

$$G(i,j) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

  -
  - i, j：対象となる点群の中心からの距離
  - σ^：標準偏差
  - σ^2：分散
- 到達点による角度のノイズに関して
  - 下記に到達点による放射角度のガウシアンノイズフィルタ適用の例を示す。青色の矢印で示した光線がノイズフィルタ適用前の光線となっている。仮想 LiDAR から放射される光線は、LiDAR の回転によりモデルに到達するが、その値は一定とは限らない（物体の移動や、回転により）。よってそれらの光線を平滑化した光線が赤色の矢印で示す。

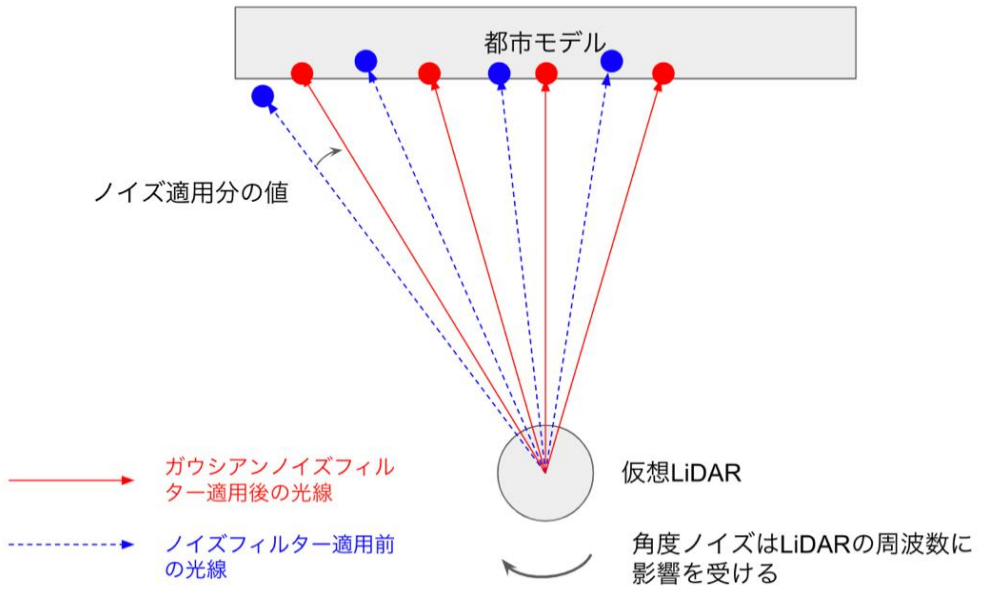


図 3-41 仮想 LiDAR の角度ノイズに関して

- 距離によるノイズに関して

- 下記に距離によるガウシアンノイズフィルタ適用の例を示す。青色の矢印で示した光線がノイズフィルタ適用前の光線となっている。仮想 LiDAR から放射される光線は、光線の飛行距離の計算により、その飛距離が算出される（time-of-flight）。その飛距離は、移動や回転により一定とは限らないため、平滑化するためにガウシアンノイズフィルタを適用した光線を赤色の矢印で示す。

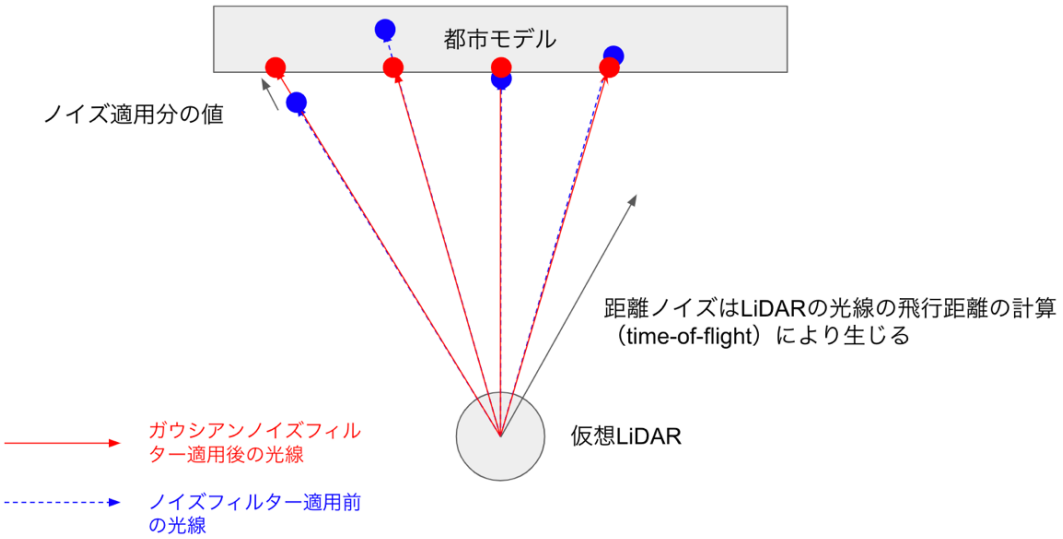


図 3-42 仮想 LiDAR の距離ノイズに関して

3-4. データインタフェース

3-4-1. ファイル入力インタフェース

1) 【IF001】Autoware-localization に入力する点群マップ

Autoware-localization では PCD 形式の点群マップを入力する。PLATEAU PCL Generator から出力された点群マップは CloudCompare を利用して間引き処理を行い、PCD 形式のマップとして出力される（ファイル出力インタフェース【IF101】を参照）。Autoware-localization では自己位置測位の環境マップとして、PCD 形式のマップを読み込む。

- 本インタフェースを利用した機能
  - 【FN104】

表 3-20 点群マップのデータ（PCD 形式）

名称	説明	データ型
X	Point の X 座標を示す	浮動点小数
Y	Point の Y 座標を示す	浮動点小数
Z	Point の Z 座標を示す	浮動点小数
Intencity	Point の強度を示す	整数

3-4-2. ファイル出力インタフェース

1) 【IF101】PLATEAU PCL Generator から出力される点群マップ

本システムでは PLATEAU PCL Generator で点群マップを出力(システム機能【FN006】)、また Cloud Compare を用いて間引き処理を行なった後に点群マップを出力（システム機能【FN007】）し、合計二回の点群マップ出力を行う。点群マップは PCD 形式で出力される。

- 本インタフェースを利用した機能【FN006】【FN007】
  - 【FN006】【FN007】【FN601】

表 3-21 点群マップのデータ（PCD 形式）

名称	説明	データ型
X	Point の X 座標を示す	浮動点小数
Y	Point の Y 座標を示す	浮動点小数
Z	Point の Z 座標を示す	浮動点小数
Intencity	Point の強度を示す	整数

2) 【IF102】スキャンマッチング精度 CSV ファイル

遠隔監視室では車両から送信されるスキャンマッチング精度データを受信し、下記の表に示す CSV ファイルにて出力を行う。CSV ファイルにはタイムスタンプ情報、位置情報、スキャンマッチングの精度が出力される。

- 本インタフェースを利用した機能
  - 【FN302】

表 3-22 スキャンマッチング精度 CSV ファイル

名称	説明	データ型
sec	タイムスタンプ	整数
latitude	緯度	浮動点小数
longitude	経度	浮動点小数
transformation_probability	スキャンマッチングの精度を示す。地図データと LiDAR スキャンデータのマッチング度合いを示す指標	浮動点小数

3-4-3. 内部連携インタフェース

1) 【IF201】仮想 LiDAR 設定インタフェース

Unity のダイアログ上にて、仮想 LiDAR の設定を行う。ダイアログ上で入力が必要となる値を下記の表に示す。

- 本インタフェースを利用した機能
  - 【FN002】

表 3-23 仮想 LiDAR 設定のダイアログの入力値

名称	説明	単位	データ型
周波数	LiDAR の周波数	Hz	整数
水平ステップ数	LiDAR の水平ステップ数	-	整数
最小水平角度	LiDAR の最小水平角度	度	整数
最大水平角度	LiDAR の最大水平角度	度	整数
飛距離	LiDAR から放射される照射の飛距離	m	整数
ノイズフィルタ有無	照射された LiDAR に対してノイズフィルタをかけるか	-	真偽値

2) 【IF202】仮想車両設定インタフェース

Unity ダイアログ上にて、仮想車両の設定を行う。ダイアログ上で入力が必要となる値を下記の表に示す。

- 本インタフェースを利用した機能
  - 【FN003】

表 3-24 仮想車両設定のデータ形式

名称	説明	単位	データ型
全長	車両の全長	m	浮動点小数
車幅	車両の車幅	m	浮動点小数
車高	車両の車高	m	浮動点小数
仮想 LiDAR 長さ	LiDAR の長さ	m	浮動点小数
仮想 LiDAR 高さ	LiDAR の高さ	m	浮動点小数
仮想 LiDAR 横幅	LiDAR の横幅	m	浮動点小数
取り付け位置 X	車両の中心を原点とした LiDAR 取り付け位置の X 座標	m	浮動点小数
取り付け位置 Y	車高を原点とした LiDAR 取り付け位置の Y 座標	m	浮動点小数
取り付け位置 Z	車両の中心を原点とした LiDAR 取り付け位置の Z 座標	m	浮動点小数

3) 【IF203】ルート作成インタフェース

ルート作成では Unity シーンを操作し、マウスとキーボード入力にて WayPoint オブジェクトを道路モデル上に追加する。その際に WayPoint オブジェクトは Root オブジェクトの子オブジェクトとなり、WayPoint を追加した分、Root オブジェクトの子オブジェクトが増える。WayPoint 子オブジェクトは追加した順番が Root 親オブジェクトに保持される。

- 本インタフェースを利用した機能
  - 【FN004】

表 3-25 ルート作成のデータ形式

名称	説明	単位	データ型
Root 親オブジェクト	WayPoint 子オブジェクトを持つ親オブジェクト	-	Unity オブジェクト
WayPoint 子オブジェクト	Root 親オブジェクトの子オブジェクト	-	Unity オブジェクト
WayPoint X 座標	X 座標	m	浮動点小数
WayPoint Y 座標	Y 座標	m	浮動点小数
WayPoint Z 座標	Z 座標	m	浮動点小数

4) 【IF210】CloudCompare 点群間引処理パラメータ

CloudCompare を用いて、PLATEAU PCL Generator で出力した点群マップの点群間引き処理を行う。Autoware では 5GB の点群マップの利用が推奨されているため、ランダムサンプリングを用いて点群マップのサイズが 5GB 以下になるように調整を行う。

- 本インタフェースを利用した機能
  - 【FN007】

表 3-26 CloudCompare 点群間引処理ダイアログの入力値

名称	説明	データ型
method	点群サンプリングのメソッドを選択	文字列
remaining points	ランダムサンプリング後の点群数を入力	整数

5) 【IF211】3D LiDAR から出力される PCD 形式のデータ

3D LiDAR からは下記に示した PCD 形式のデータを受け取る。自己位置測位では ros2 ouster（ソフトウェア・ライブラリ【SL009】を参照）を用いて PointCloud2 形式の ROS メッセージへ変換する。

- 本インタフェースを利用した機能
  - 【FN101】【FN601】

表 3-27 3D LiDAR から出力される点群データ（PCD 形式）

名称	説明	データ型
X	Point の X 座標を示す	浮動点小数
Y	Point の Y 座標を示す	浮動点小数
Z	Point の Z 座標を示す	浮動点小数
Intencity	Point の強度を示す	整数

6) 【IF212】ros2 ouster から出力される点群データ

点群データの ROS メッセージ形式を下記の表に示す。本システムでは 3D LiDAR から取得したデータを ros2 ouster を用いて ROS メッセージに変換を行なった。また Autoware-sensing では、点群のノイズ除去を行なった上で下記の ROS メッセージを出力する。

- 本インタフェースを利用した機能
  - 【FN101】【FN103】

表 3-28 点群データの ROS メッセージ（sensor_msgs/PointCloud2 形式）

名称	説明	ROS データ型
header	センサデータ取得時刻やフレーム ID などのヘッダー情報	Header
height	点群データの 2 次元構造を示す次元の行数	uint32
width	点群データの 2 次元構造を示す次元の列数	uint32
fields	点群データ構造のレイアウトを記述	PointField[]
is_bigendian	データがビッグインディアンか否か	bool
point_step	点群データの長さ	uint32
row_step	点群データの行数	uint32
data	実際に得られる点群のデータ	uint8[]
is_dense	点群の密度を示す。 全ての点群データが有効な場合は真を返す。	bool

7) 【IF213】NDT スキャンマッチングパラメータ

Autoware-localization 内で処理される NDT スキャンマッチングのデータ形式を下記に示す。（項目の詳細に関してはアルゴリズム【AL001】を参照）

- 本インタフェースを利用した機能
  - 【FN104】

表 3-29 NDT スキャンマッチングパラメータのデータ形式

名称	説明	データ型
trans_epsilon	NDT スキャンマッチングにおいて変換の収束を判断するための許容誤差を指定するパラメータ	浮動点小数
step_size	線形探索において使用される最大ステップ長を指定するパラメータ	浮動点小数
voxel_grid_resolution	ボクセルのグリッドの解像度	浮動点小数
max_iterations	最大反復回数を指定するパラメータ	整数
num_threads	並列計算に使用されるスレッド数	整数

8) 【IF221】NMEA データ

RTK 測位デバイスから出力される NMEA データの形式を下記の表に示す。

- 本インタフェースを利用した機能
  - 【FN102】

表 3-30 NMEA データ形式

名称	説明	データ型
データタイプ	GxGGA の形式で表されたメッセージ ID	文字列
時刻	UTC 時刻	浮動点小数
緯度	緯度情報	浮動点小数
南北	緯度の南北を表す	文字列
経度	経度情報	浮動点小数
東西	経度の東西を表す	文字列

9) 【IF222】位置情報 ROS メッセージ

RTK 測位デバイスから NMEA データを受け取り、nmea navsat driver にて ROS メッセージに変換した際のデータ形式。

- 本インタフェースを利用した機能
  - 【FN102】

表 3-17 位置情報 ROS メッセージのデータ形式（sensor_msgs/NavSatFix 形式）

名称	説明	ROS データ型
header	センサデータ取得時刻やフレーム ID などのヘッダー情報	Header
latitude	緯度	float64
longitude	経度	float64
altitude	高度	float64
status	衛星の FIX 情報	NavSatFix

10) 【IF223】Autoware-localization から出力される自己位置測位情報

Autoware-localization から出力される ROS メッセージのデータ形式を下記の表に示す

- 本インタフェースを利用した機能
  - 【FN105】

表 3-18 Autoware-localization から出力される自己位置測位情報のデータ形式

名称	説明	ROS データ型
ndt_pose	推定姿勢情報	geometry_msgs::msg::PoseStamped
ndt_pose_with_covariance	分散付きの推定姿勢情報	geometry_msgs::msg::PoseWithCovarianceStamped
diagnostics	自己位置測位の状態	diagnostic_msgs::msg::DiagnosticArray
exe_time_ms	スキャンマッチングの実行時間	tier4_debug_msgs::msg::Float32Stamped
transform_probability	スキャンマッチングのスコア	tier4_debug_msgs::msg::Float32Stamped
iteration_num	スキャンマッチングの反復回数	tier4_debug_msgs::msg::Int32Stamped

11）【IF231】カメラ読み込み

UVC 形式のカメラを用いて遠隔監視室から映像の監視を行う。

- 本インタフェースを利用した機能
  - 【FN201】

表 3-19 映像配信のデータ形式

名称	データ型
カメラ読み込み	UVC1.0

12）【IF232】映像配信

カメラから読み込んだ映像は AV1 形式に圧縮して遠隔監視ビュワーに送信する。

- 本インタフェースを利用した機能
  - 【FN201】【FN202】

表 3-20 映像配信のデータ形式

名称	データ型
映像圧縮方式	AV1

3-4-4. 外部連携インタフェース

1) 【IF301】MQTT メッセージ送受信

- インタフェースの概要
  - 自己位置測位の精度割合データをリアルタイム精度監視グラフに表示するために、MQTT を用いてデータを送信する
- 本インタフェースを利用した機能
  - 【FN105】【FN301】【FN401】【FN501】
- プロトコル
  - MQTT
- MQTT パス
  - MQTT Broker の URL
- メソッド
  - Publish, Subscribe
- ペイロード
  - MQTT のペイロードを下記に示す。

表 3-21 MQTT のペイロード

名称	説明	データ型
time_stamp	タイムスタンプ	整数
latitude	緯度	浮動点小数
longitude	経度	浮動点小数
transform_probability	スキャンマッチングのスコア	浮動点小数

3-5. 実証に用いたデータ

3-5-1. 活用したデータ一覧

1） 利用した 3D 都市モデル 横浜市

- 年度：2022 年度
- 都市名：横浜市
- ファイル名：14100_yokohama-shi_2022_citygml_1_op
- メッシュ番号：53391530、53391531、53391540, 53391541（インデックスマップで黄色囲いの箇所）

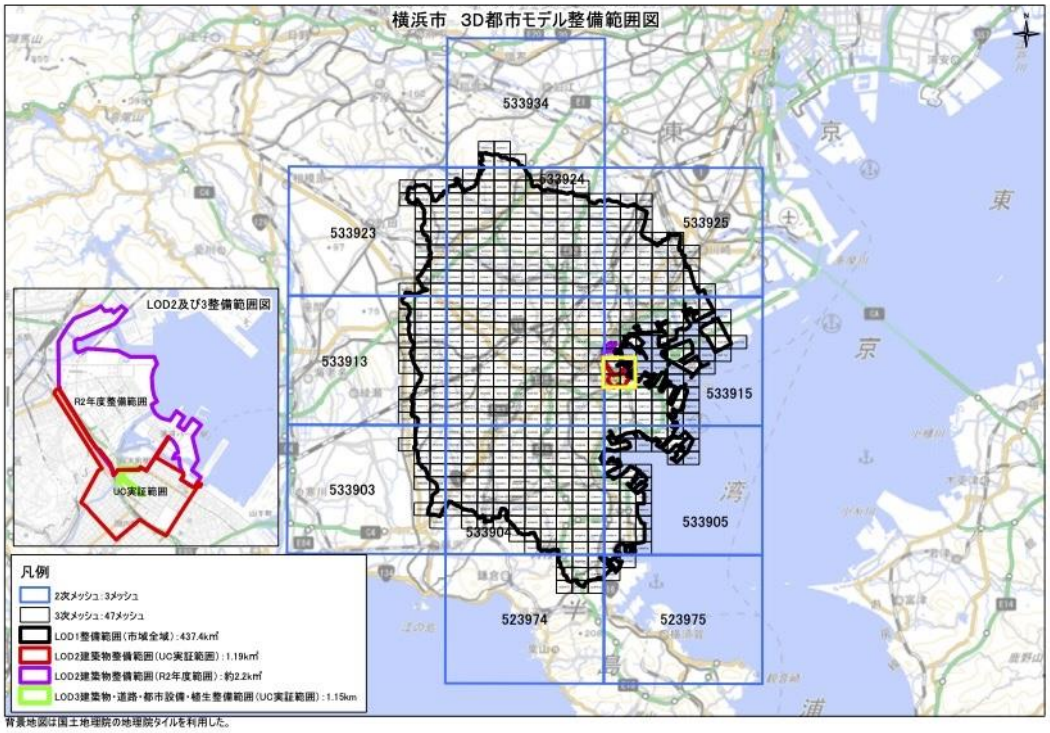


図 3-50 インデックスマップ（横浜市）

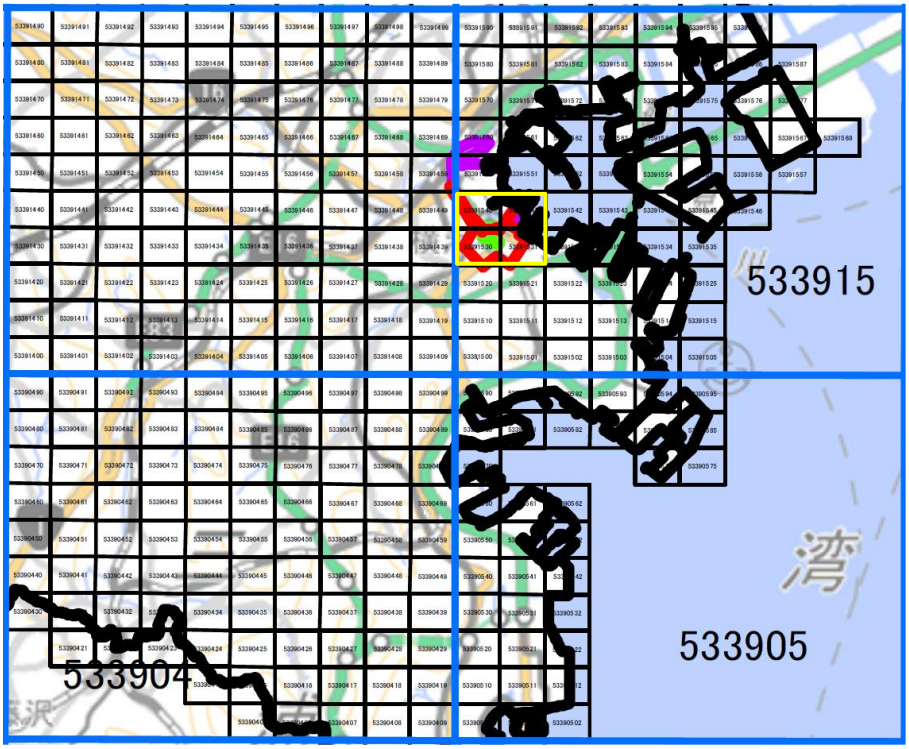


図 3-51 インデックスマップ　拡大（横浜市）

表 3-2 利用した 3D 都市モデル

地物	地物型	属性区分	ID	属性名	内容	データを利用した機能（ID）
建築物 LOD1 LOD2 LOD3	bldg:Building	空間属性	DT001	bldg:lod1Solid	LOD1 空間モデル	FN001
			DT002	bldg:lod2Solid	LOD2.0 建築物モデル	FN001
			DT003	bldg:lod3Solid	LOD3.0 建築物モデル	FN001
道路 LOD1・ LOD2	tran: Road	空間属性	DT011	tran:lod1MultiSurface	LOD1 道路モデル	FN001
			DT012	tran:lod3MultiSurface	LOD3.0 道路モデル	FN001
都市設備 LOD3	frn:CityFurniture	空間属性	DT021	frn:lod3Geometry	LOD3.0 都市設備モデル	FN001
植生 LOD3	veg:SolitaryVegetationObject	空間属性	DT031	veg:lod3Geometry	LOD3 植生モデル	FN001
地形	dem	空間属性	DT041	dem:ReliefFeature	地形モデル	FN001

2） 利用した 3D 都市モデル 大阪市

- 年度：2022 年度
- 都市名：大阪市
- ファイル名：27100_osaka-shi_2022_citygml_1
- メッシュ番号：52350300、52350301、52350302、51357390、51357391、53157392（インデックスマップで黄色囲いの箇所）

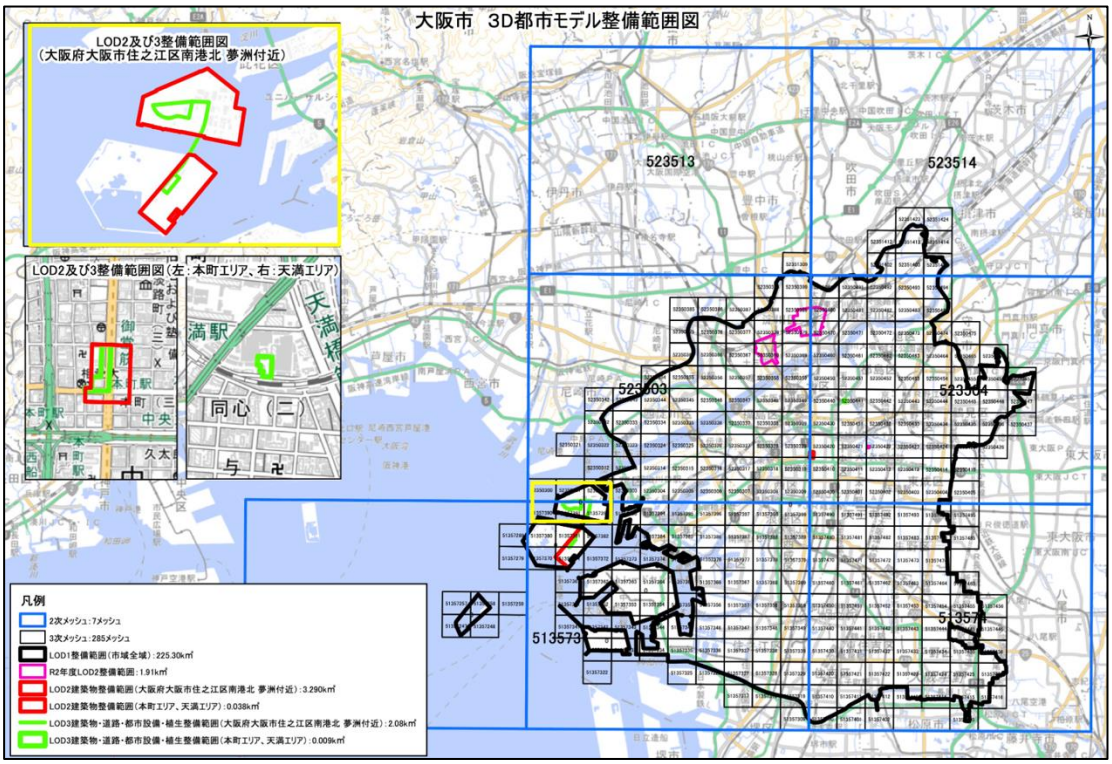


図 3-52 インデックスマップ（大阪市）

1. データ一覧

表 3-3 利用したその他データ（一覧）

地物	地物型	属性区分	ID	属性名	内容	データを利用した機能（ID）
建築物 LOD1 LOD2	bldg:Building	空間属性	DT101	bldg:lod1Solid	LOD1 空間モデル	FN001
			DT102	bldg:lod2Solid	LOD2.0 建築物モデル	FN001
道路 LOD1・LOD2	tran: Road	空間属性	DT111	tran:lod3MultiSurface	LOD3.0 道路モデル	FN001
都市設備 LOD3	frn:CityFurniture	空間属性	DT121	frn:lod3Geometry	LOD3.0 都市設備モデル	FN001
植生 LOD3	veg:SolitaryVegetationObject	空間属性	DT131	veg:lod3Geometry	LOD3 植生モデル	FN001

3-5-2. 生成・変換したデータ

表 3-4 生成・変換したデータ

ID	システムに入力するデータ（データ形式）	用途	処理内容	データ処理ソフトウェア	活用データ（データ形式）	データを利用した機能 (ID)
DT201	3D 都市モデル（CityGML 形式）	点群マッチングのベースとなる点群マップを生成する	● 3D 都市モデル（CityGML）を PLATEAU PCL Generator に読み込み、点群マップを生成する	PLATEAU PCL Generator	PCD 形式	FN005

3-6. ユーザーインタフェース

3-6-1. 画面一覧

1） PLATEAU PCL Generator 画面

表 3-5 PLATEAU PCL Generator 画面一覧

ID	連携（ID）	画面名	画面説明	画面を表示した機能（ID）
SC001		メイン画面	● Unity シーンのメイン画面	
SC002	005	LiDAR、車両設定ダイアログ	● 仮想 LiDAR､仮想車両のパラメータを設定できるダイアログ	FN002, 003
SC003	005	ルート名設定ダイアログ	● ルート名を設定できるダイアログ	FN004
SC004	005	WayPoint 設定画面	● Unity シーン上で、ルートを設定するための WayPoint を設定する画面	FN005
SC005		仮想走行画面	● 仮想車両が仮想空間（3D 都市モデル）内を走行する画面	FN005

2） 遠隔監視室画面

表 3-6 遠隔監視室画面一覧

ID	連携（ID）	画面名	画面説明	画面を表示した機能（ID）
SC101	―	デジタルツインビューア	● 3D 都市モデルを利用し、車両の位置を 3D で表示するデジタルツインビューア	FN105
SC102	―	遠隔監視ビューア	● 車両に搭載されたカメラの映像を監視するためのビューア	FN103
SC103	―	精度割合グラフ描画	● 自己位置測位の精度割合の結果をグラフで表示	FN104

3-6-2. 画面遷移図

1） PLATEAU PCL Generator 画面

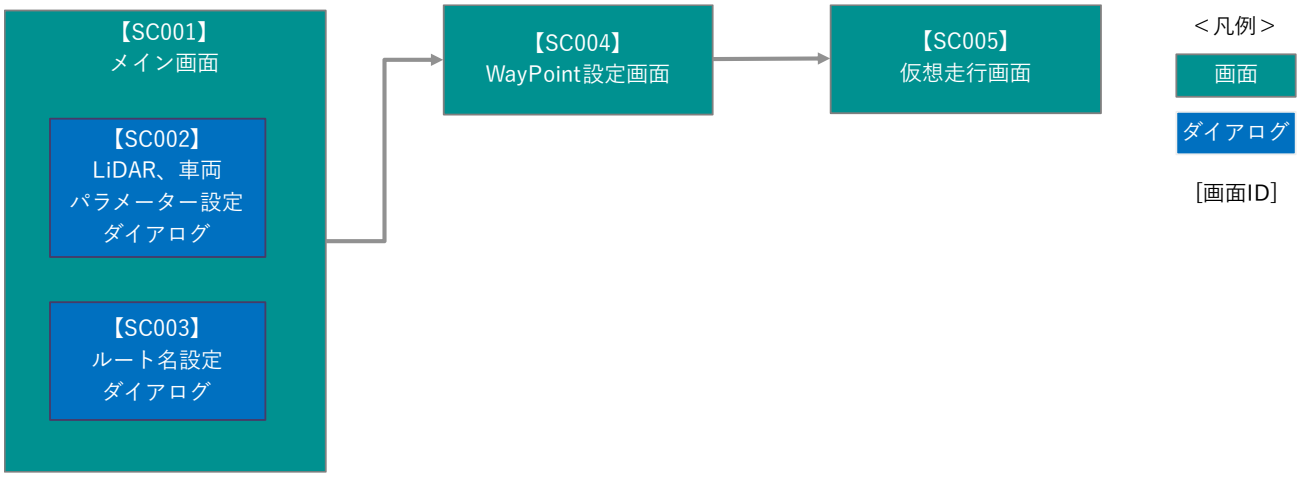


図 3-60 PLATEAU PCL Generator の画面遷移図

3-6-3. 各画面仕様詳細

1） PLATEAU PCL Generator 画面

1. 【SC001】メイン画面

- 画面の目的・概要
  - PLATEAU SDK for Unity を用いて 3D 都市モデルを配置したメイン画面
  - 画面上の PLATEAU PCL Generator から、各種設定が可能
- 画面イメージ

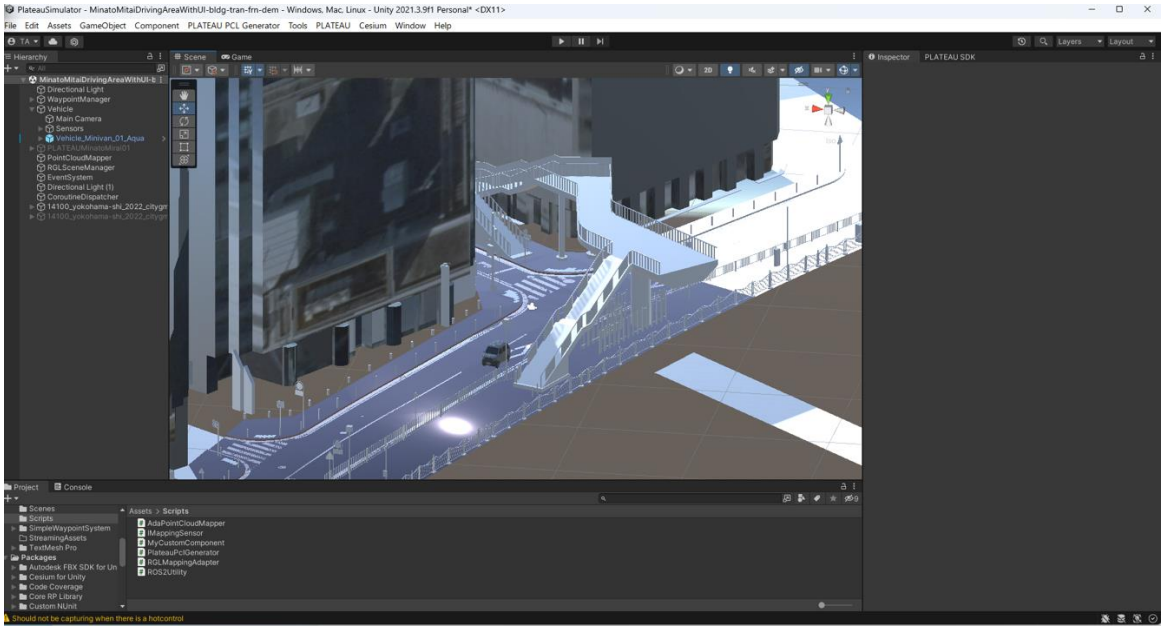


図 3-61 メイン画面のイメージ

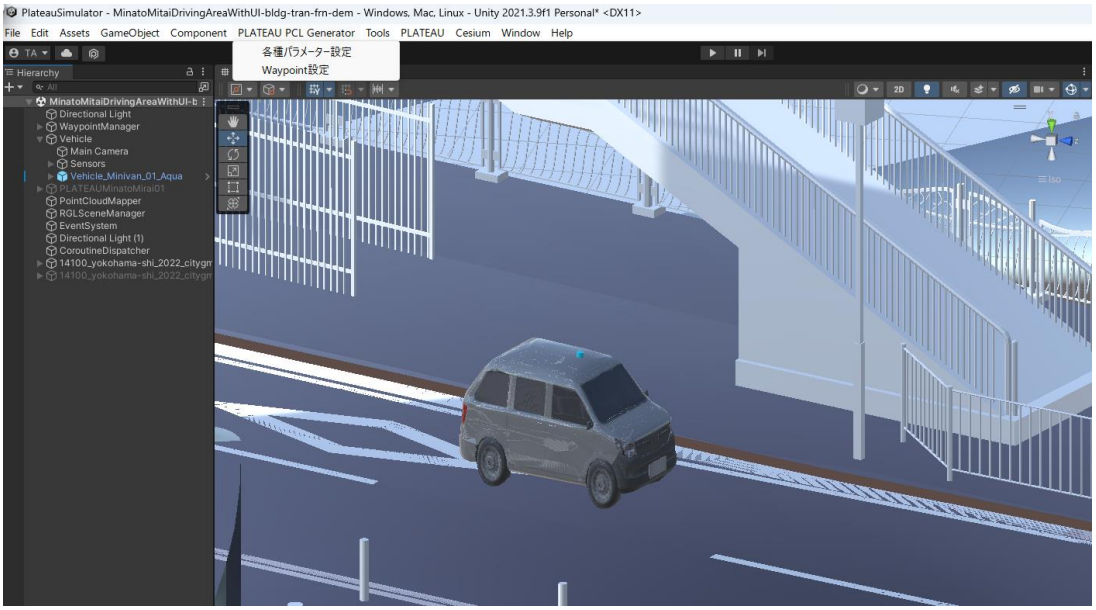


図 3-62 PLATEAU PCL Generator メニューバー

2. 【SC002】LiDAR、車両設定ダイアログ

- 画面の目的・概要
  - Unity のメニューから簡易的に LiDAR や車両の設定をすることが出来る
  - 仮想 LiDAR の各種パラメータを設定する（システム機能【FN002】を参照）
  - 仮想車両の各種パラメータを設定する（システム機能【FN003】を参照）
- 画面イメージ

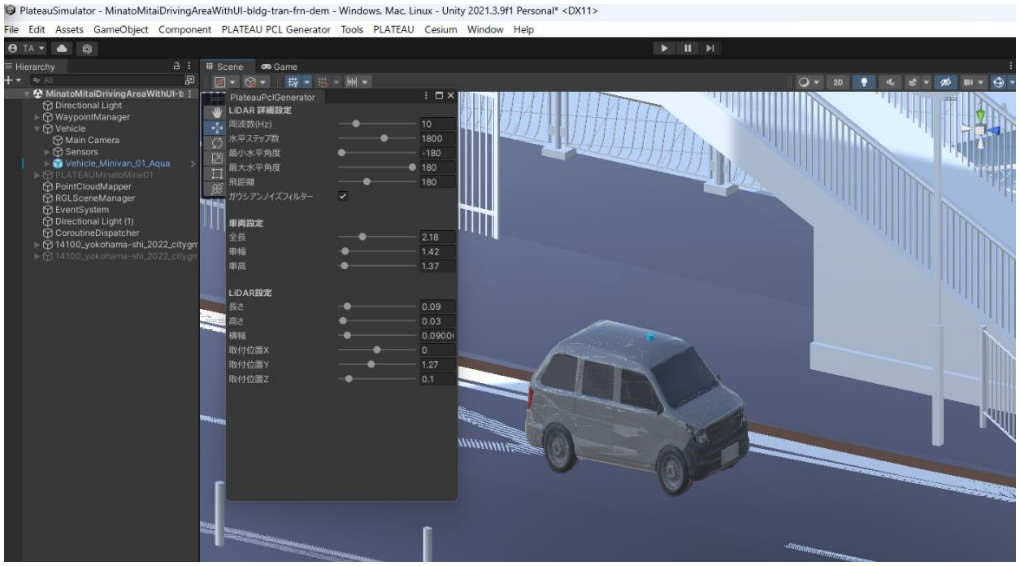


図 3-63 仮想 LiDAR、仮想車両設定ダイアログのイメージ

3. 【SC003】ルート名設定ダイアログ

- 画面の目的・概要
  - Unity のメニューから簡易的にルート設定をすることが出来る
  - ルート名を設定、パス作成開始ボタンで、WayPoint を設定することが可能
  - パスを設定ボタンで、ルートを設定することが可能
- 画面イメージ

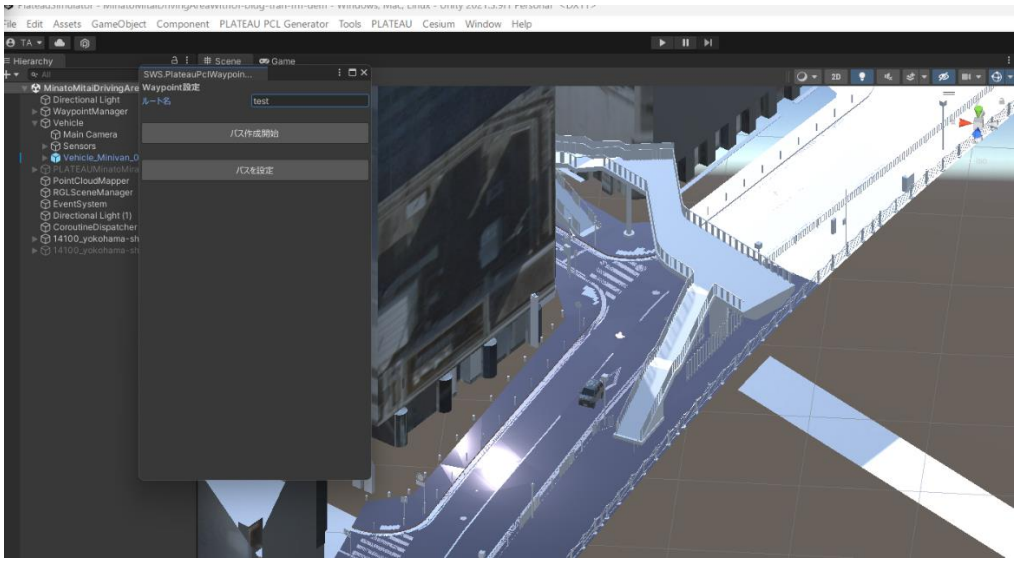


図 3-64 ルート名設定ダイアログのイメージ

4. 【SC004】WayPoint 設定画面

- 画面の目的・概要
  - Unity のシーン上で簡易的に WayPoint の設定をすることが出来る
  - Unity シーンを動かしながら、キーボードで P ボタンを入力することで、WayPoint を設定可能
- 画面イメージ

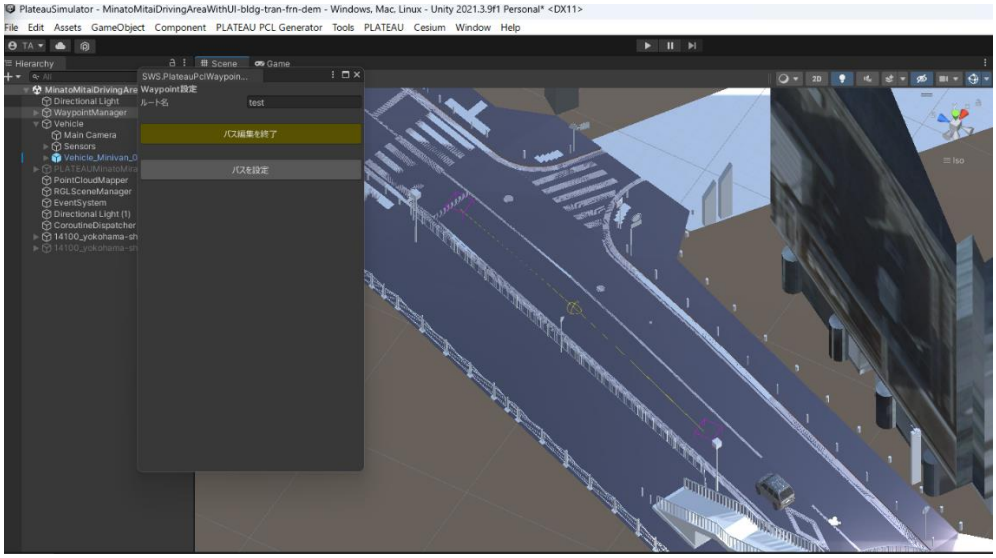


図 3-65 WayPoint 設定のイメージ

5. 【SC005】仮想走行画面

- 画面の目的・概要
  - Unity のシーンから簡易的に車両の走行を開始することが出来る
  - Play ボタンを押すと、ルートを走行開始
- 画面イメージ

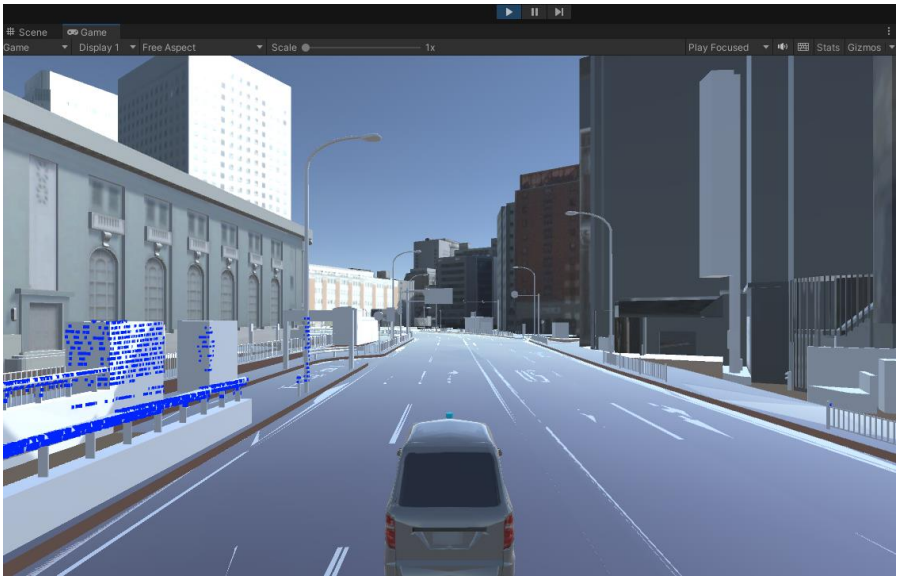


図 3-66 仮想走行画面のイメージ

2） 遠隔監視画面

1. 【SC101】デジタルツインビュワー

- 画面の目的・概要
  - 3D 都市モデルを利用し、車両の位置をデジタルツインビューで表示することで車両の走行状態を 3 次元的に可視化することが出来る
- 画面イメージ

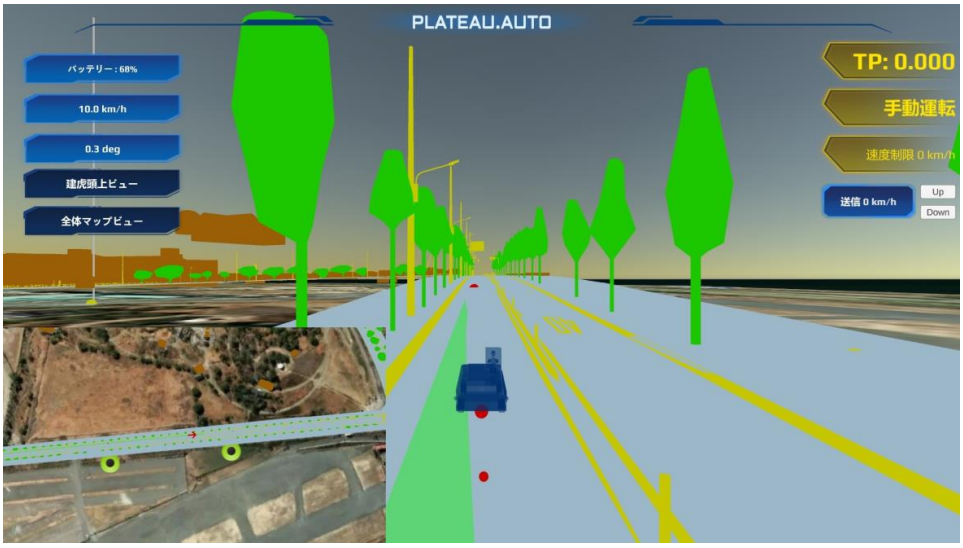


図 3-67 デジタルツインビュー画面のイメージ

2. 【SC102】遠隔監視ビュワー

- 画面の目的・概要
  - 車両に搭載したカメラ映像を、リアルタイムに確認することが出来る
- 画面イメージ

図 3-68 遠隔監視ビュワー画面のイメージ

3. 【SC103】精度割合グラフ描画

- 画面の目的・概要
  - 点群マッチングの精度割合をリアルタイムに表示することが出来る
  - 横軸は時間（ｓ）、縦軸はマッチング精度割合を示す
- 画面イメージ

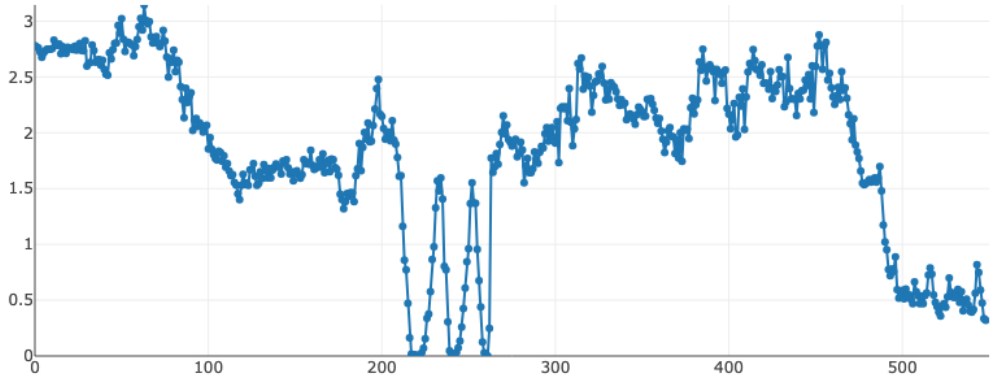


図 3-69 精度割合グラフ描画のイメージ

3-7. 実証システムの利用手順

3-7-1. 実証システムの利用フロー

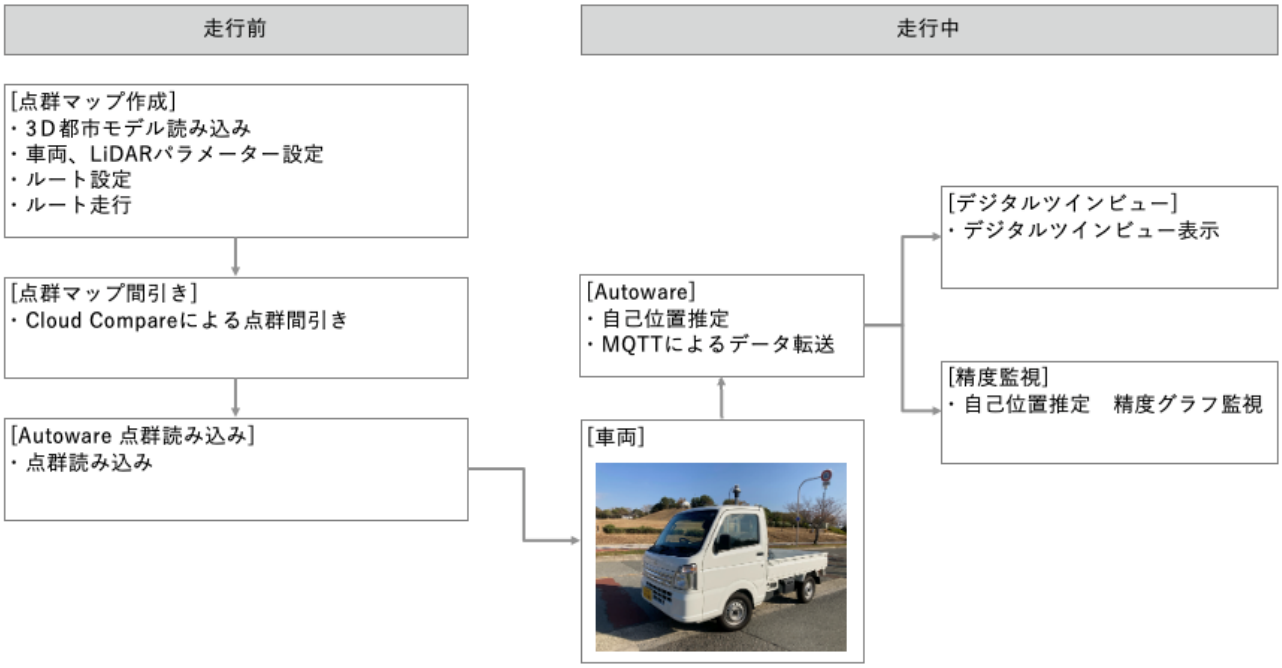


図 3-70 システムの利用フロー

3-7-2. 各画面操作方法

1） 3D 都市モデル　読み込み

- PLATEAU SDK for Unity を利用して、3D 都市モデルを読み込む

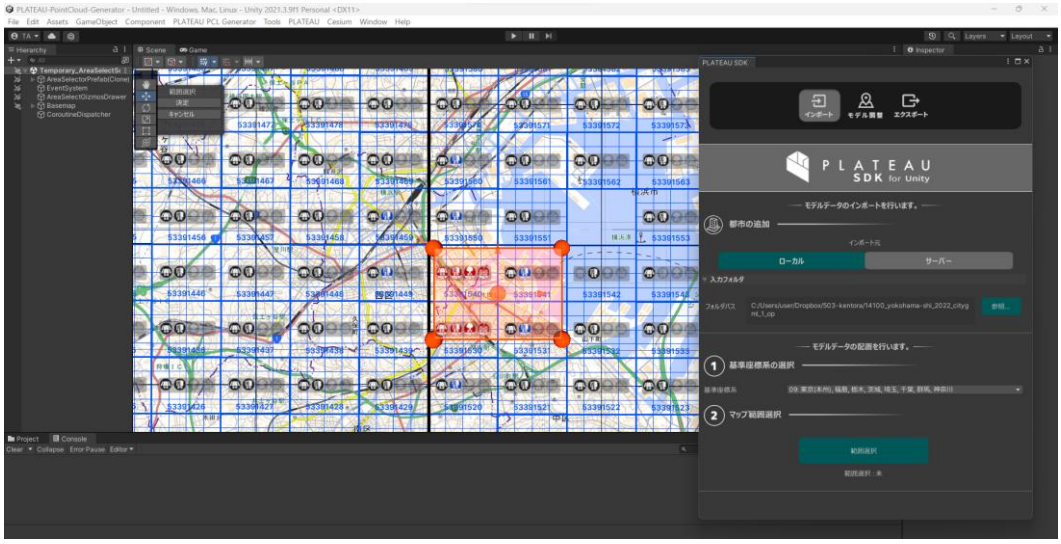


図 3-71 PLATEAU SDK for Unity 読み込み画面

- 地物を読み込む際は、「Mesh Collider をセットする」チェックボックスをオンにする

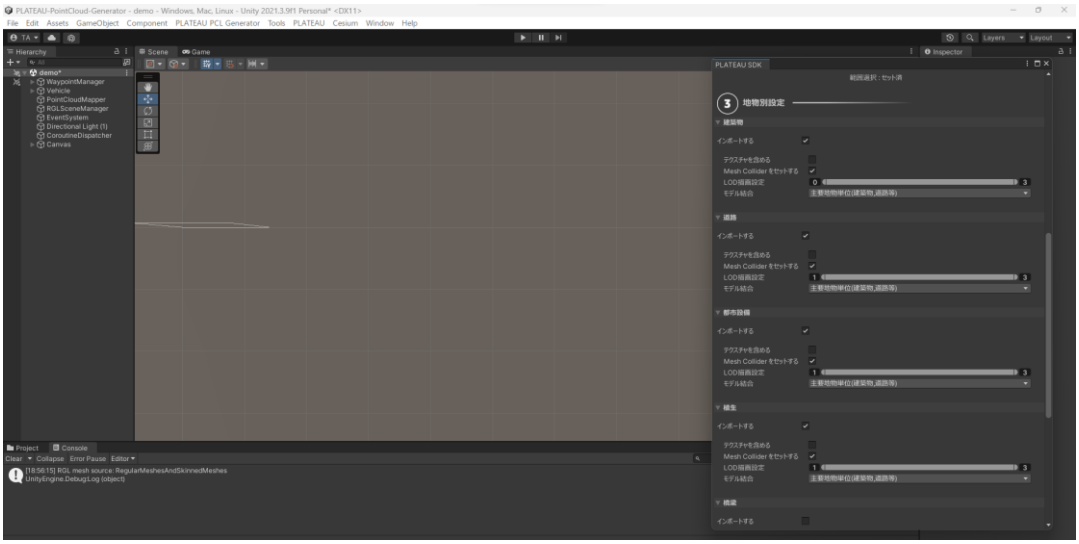


図 3-72 PLATEAU SDK for Unity 地物別設定画面

- 3D 都市モデルをシーンに配置後の画面

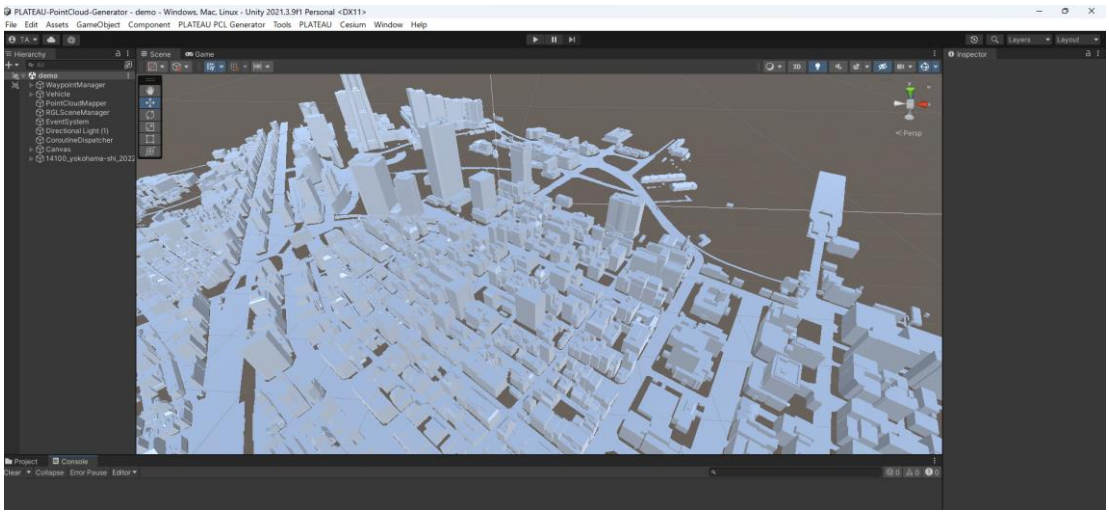


図 3-73 PLATEAU SDK for Unity 読み込み画面

2) 車両、LiDAR パラメータ設定

- Unity メニューバーから、「PLATEAU PCL Generator」ボタンをクリック
- 「各種パラメータ設定」ボタンを押す

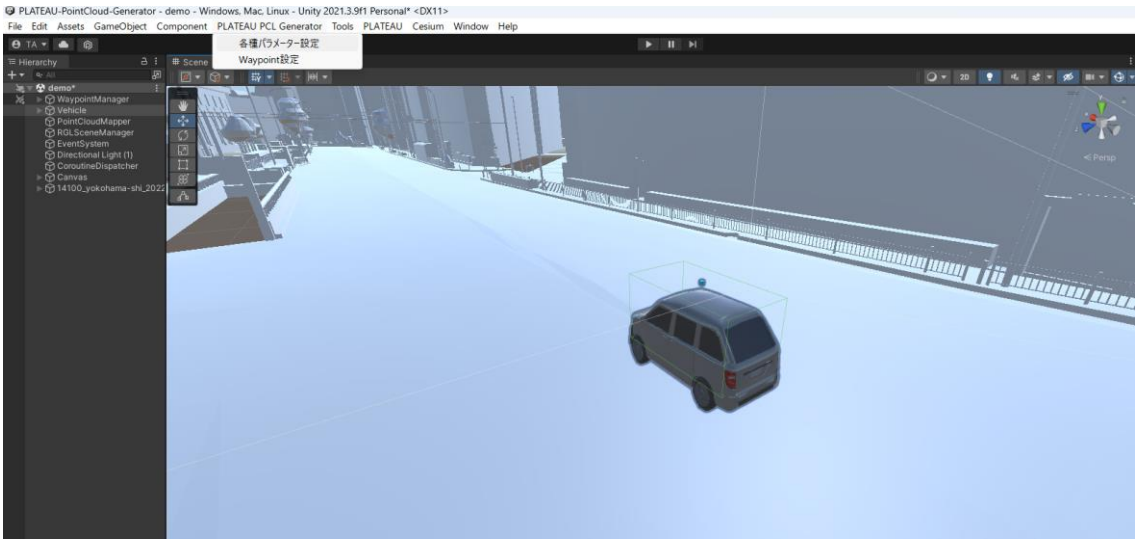


図 3-74 PLATEAU PCL Generator メニューバー

- パラメータ設定ウィンドウより、車両、LiDAR のパラメータを設定する

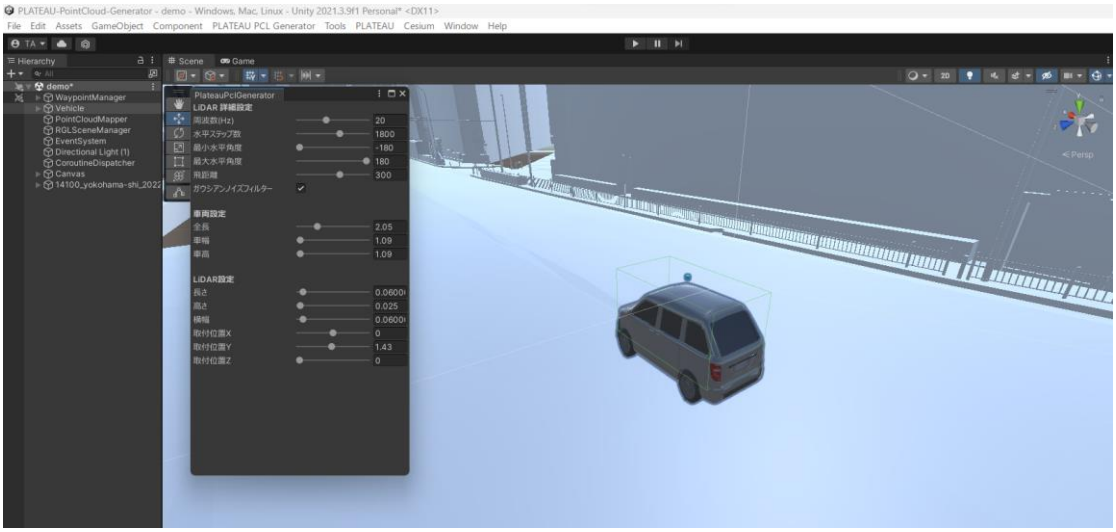


図 3-75 車両、LiDAR パラメータ設定ウィンドウ

3) ルート設定

- Unity メニューバーから、「PLATEAU PCL Generator」ボタンをクリック
- 「Waypoint 設定」ボタンを押す

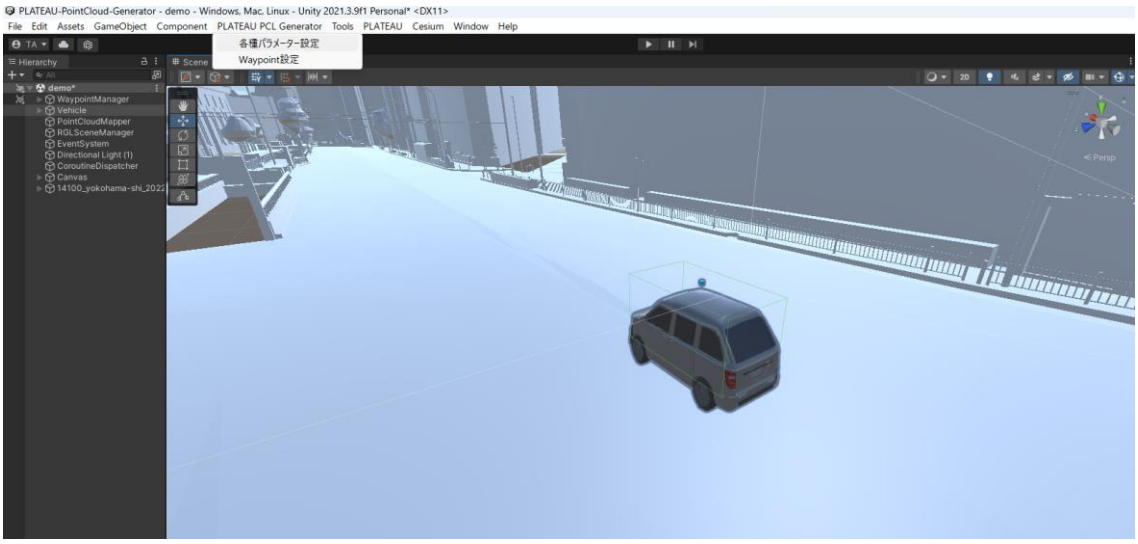


図 3-76 PLATEAU PCL Generator メニューバー

- Waypoint 設定ウィンドウが開く
- ルート名を入力
- 「パス作成開始」ボタンを押す

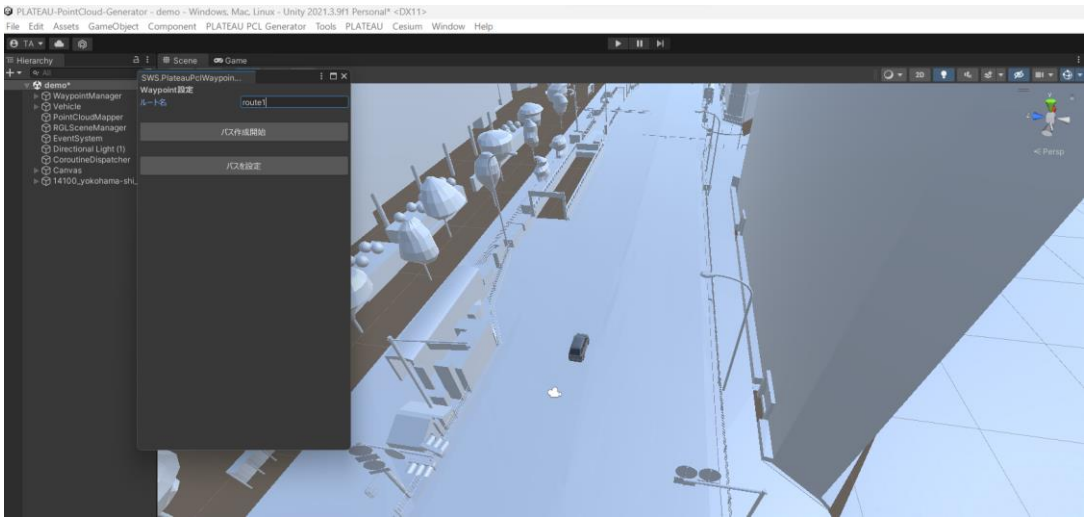


図 3-77 Waypoint 設定ウィンドウ

- シーン上で P キーを押すことで、Waypoint オブジェクトを配置する
- 配置後、「パス編集を終了ボタン」を押す

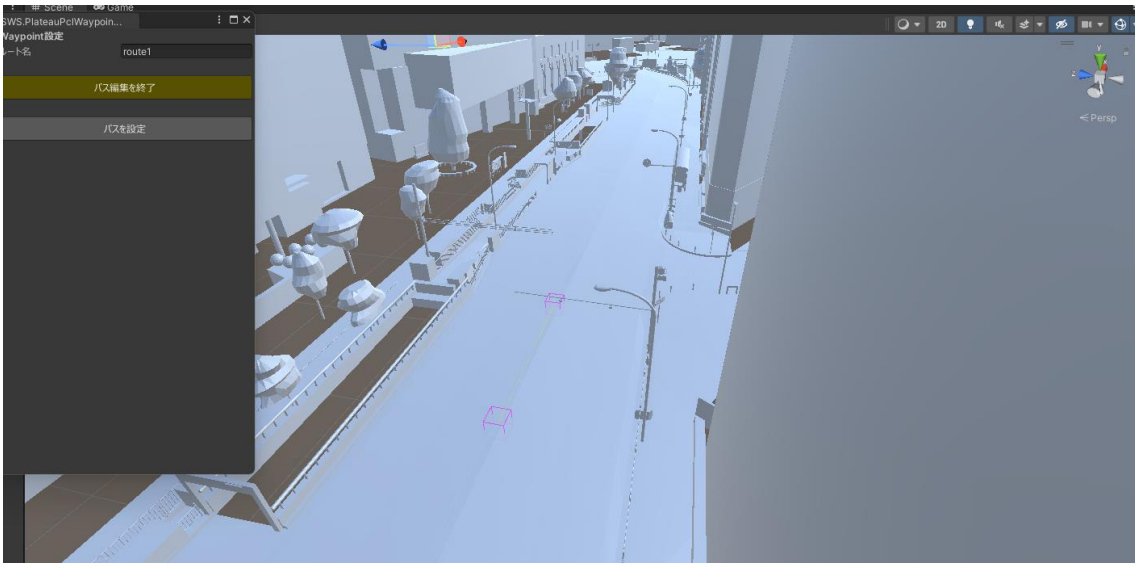


図 3-78 Waypoint 配置

- 「パスを設定」を押し、車両の走行ルートを設定

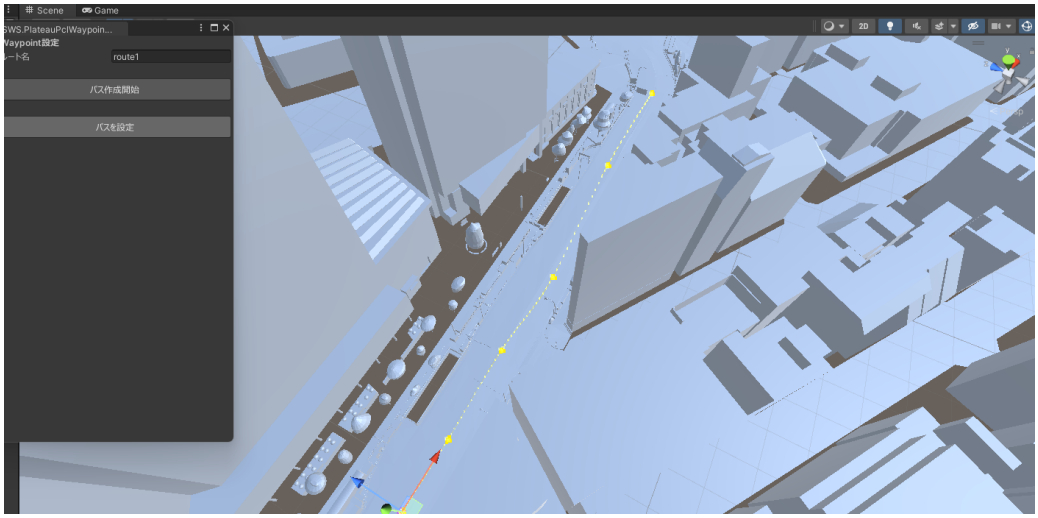


図 3-79 Waypoint 設定完了

4) ルート走行

- Unity シーン内の Play ボタンを押すと、車両が走行を開始

図 3-80 Unity シーン再生

- 走行終了後、Play ボタンを再度押すと点群マップを出力

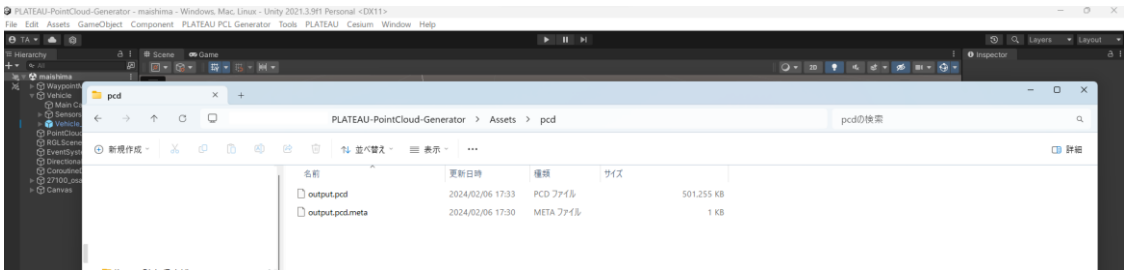


図 3-81 点群マップ出力

5） CloudCompare による点群マップ間引き

- 出力された点群マップを CloudCompare で読み込み

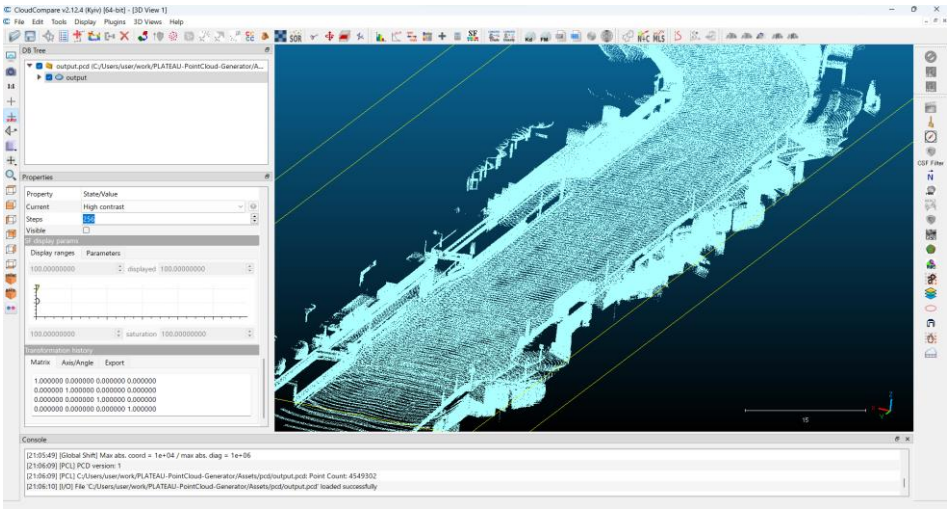


図 3-82 CloudCompare で点群マップ読み込み

- サンプリング画面を開き、ランダムサンプリングを実行
- 点群マップのサイズが 5GB 以内になるように調整

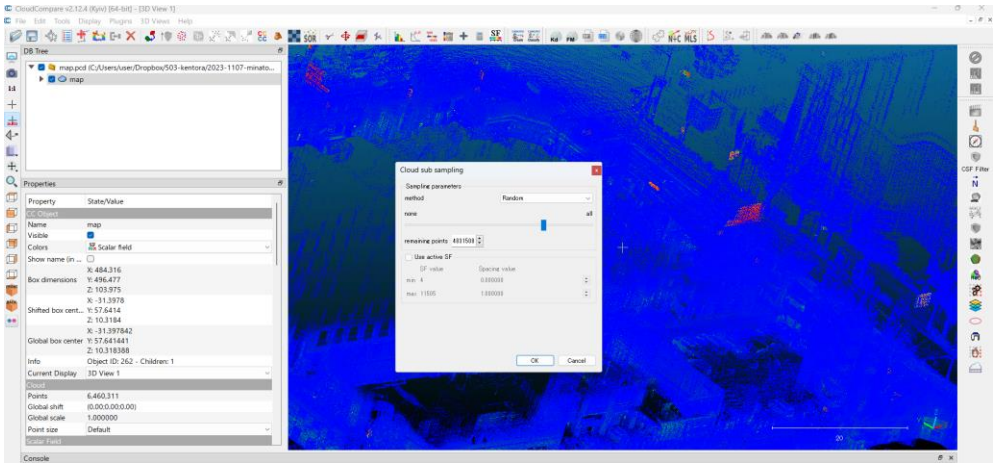


図 3-83 CloudCompare で間引き処理

6） Autoware 点群読み込み

- 間引き処理された点群マップを Autoware で読み込み

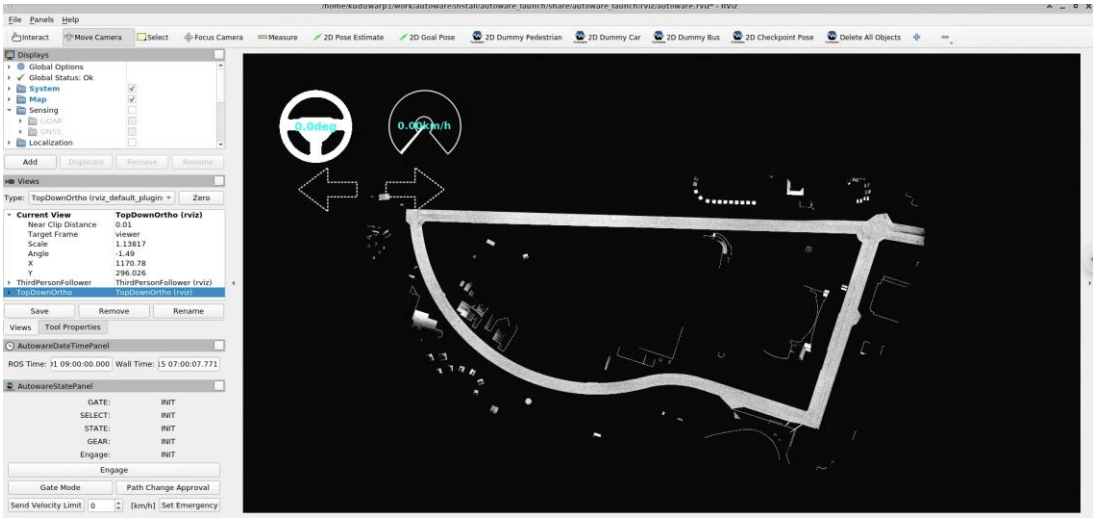


図 3-84 Autoware 点群マップ読み込み

7） Autoware 自己位置推定

- Autoware を用いて、自己位置推定処理を行う

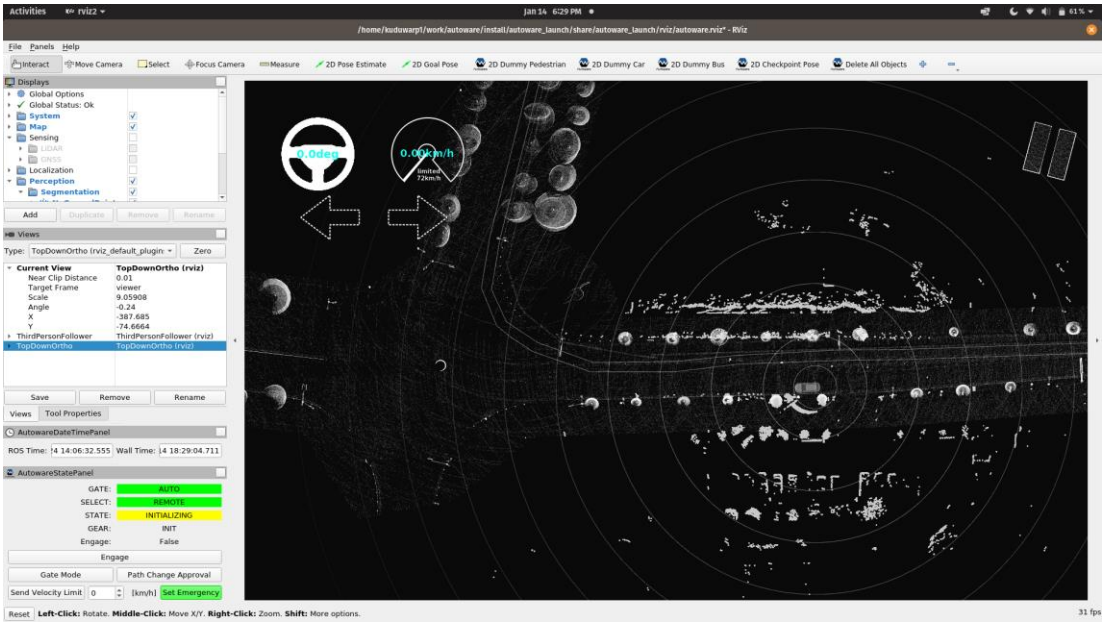


図 3-85 Autoware 自己位置推定

8） Autoware 自己位置推定

- 読み込まれた点群マップと、車載した LiDAR の点群データを用いて、自己位置推定処理を行う

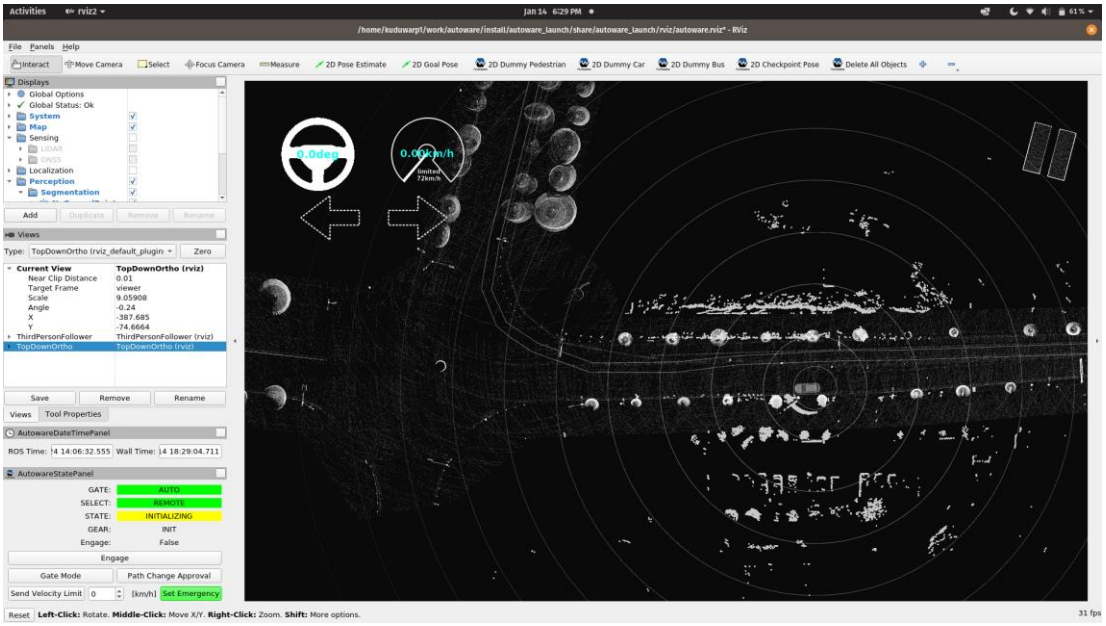


図 3-86 Autoware 自己位置推定

9) デジタルツインビュー

- デジタルツインビューにて、車両の現在位置を確認する

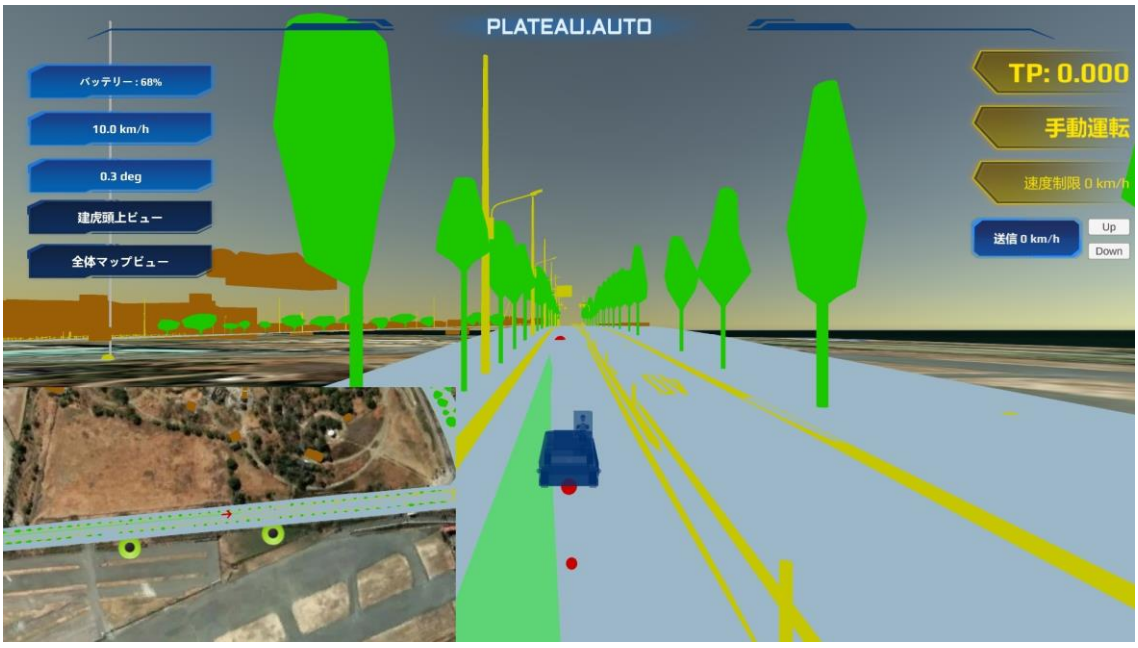


図 3-87 デジタルツインビュー画面

10) 自己位置推定　精度グラフ監視

- 精度グラフ監視スクリプトを実行し、自己位置推定の精度を監視する

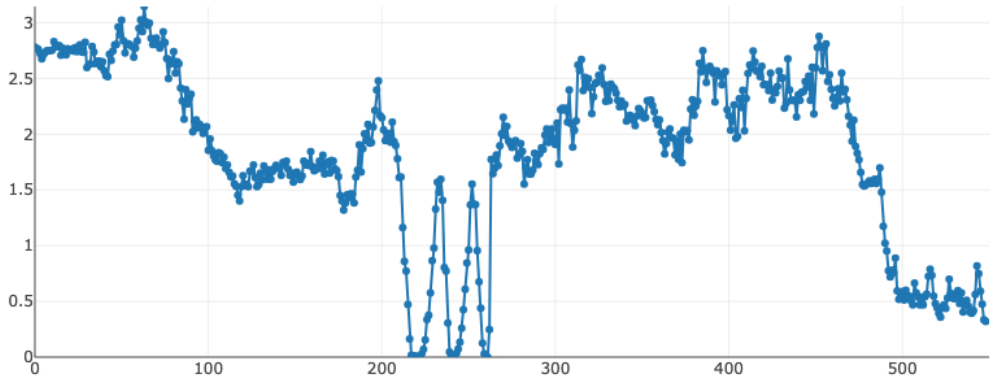


図 3-88 自己位置推定　精度監視グラフ

4. 実証技術の検証

4-1. 自己位置推定の精度割合検証　神奈川県横浜市　みなとみらい地区

- 神奈川県横浜市みなとみらい地区で行った自己位置推定の精度割合の検証結果を下記にまとめる。

4-1-1. 検証目的

- 神奈川県横浜市みなとみらい地区において、PLATEAU PCL Generator（以下「PCLG」という。）から出力された点群マップを用いた自己位置推定の精度割合と、実際に車両を走行させて取得したリアルマップの自己位置推定の精度割合の比較を行う。
- LOD 別の有用性検証を行うために、PCLG からは下記の 3 つの点群マップを出力し、リアルマップとの精度割合の比較を行う。
  1. 建築物 LOD1、道路 LOD1
  2. 建築物 LOD2、道路 LOD1
  3. 建築物 LOD3、道路 LOD3、都市設備 LOD3
- 速度別の有用性検証を行うために、同一のマップを用いて最大速度 20km/h と 30km/h で走行し、リアルマップとの精度割合の比較を行う。

4-1-2. KPI

表 4-1 KPI 一覧

No.	評価指標・KPI	目標値	目標値の設定理由	検証方法サマリー
1	点群マッチングの精度割合 建築物 LOD1、道路 LOD1	30%	建築物が多い方が精度割合は増加すると考えられる	事前取得点群マップとの精度割合データとの比較を基に精度割合を算出
2	点群マッチングの精度割合 建築物 LOD2、道路 LOD1	30%	建築物 LOD2 は建築物 LOD1 と比較しても外壁の差異がないため、精度割合は変わらないと考えられる	同上
3	点群マッチングの精度割合 建築物 LOD3、道路 LOD3、都市設備 LOD3	70%	建築物が多いエリアでは、特徴点が多く、自己位置推定の精度割合は増加すると予測される	同上
4	速度別の比較	-	（走行速度によるマッチングの精度割合の推移を比較するのみのため、目標値なし）	最大速度 20 ㎞/h と 30 ㎞/h のマッチング精度割合を経過比較

図 4-1 実証実験の様子（みなとみらい地区）遠隔監視　室内

図 4-2 実証実験の様子（みなとみらい地区）遠隔監視　室内

図 4-3 実証実験の様子（みなとみらい地区）車両と走行道路

図 4-4 実証実験の様子（みなとみらい地区）車両の俯瞰写真

図 4-5 実証実験の様子（舞洲地区）遠隔監視　室内

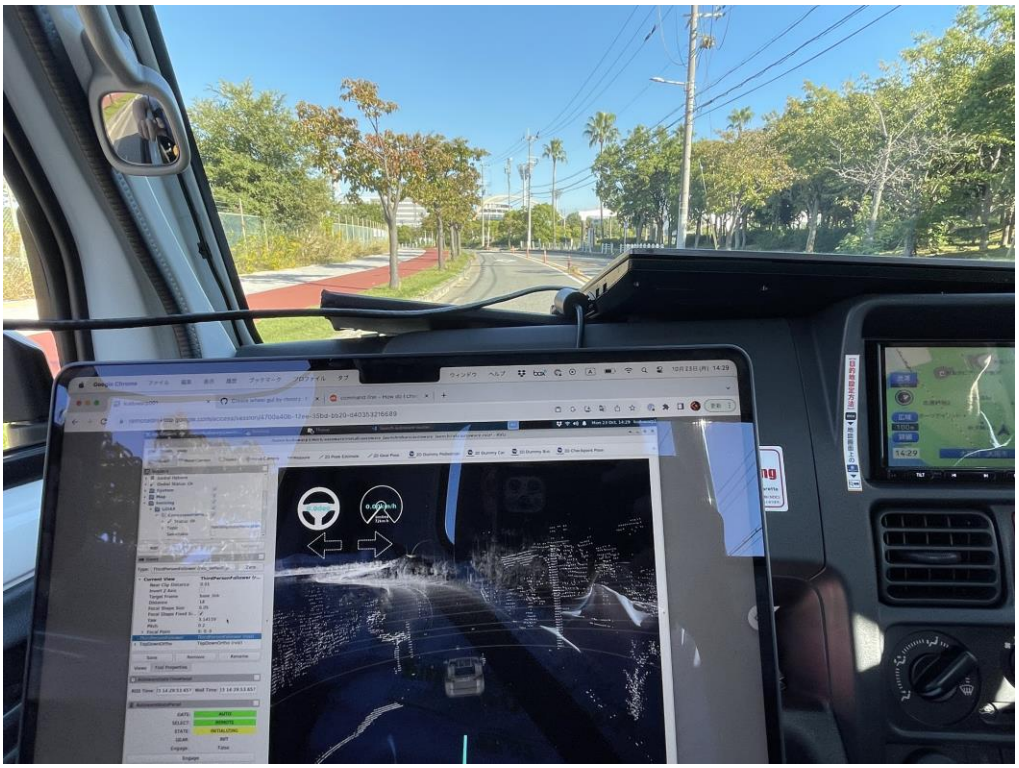


図 4-6 実証実験の様子（舞洲地区）走行中の車内

図 4-7 実証実験の様子（舞洲地区）車両と走行道路

図 4-8 実証実験の様子（舞洲地区）車両と走行道路

4-1-3. 検証方法と検証シナリオ

1） 点群マッチングの精度割合

リアルマップを用いた点群マッチングの精度と、PCLG マップを用いた場合の点群マッチングの精度の比較を行う。

分母：リアルマップを用いた際の点群マッチングの精度

分子：PCLG マップを用いた際の点群マッチングの精度

$$\text{マッチングの精度割合[\%]} = \frac{\text{PCLG マップによる点群マッチングの精度（TP 値）}}{\text{リアルマップによる点群マッチングの精度（TP 値）}} \times 100$$

- TP（Transformation Probability、変換確率）：NDT スキャンマッチングのスコアを表す。スキャンデータと地図データの間のマッチングの度合いを示す指標

2） 元浜町通り～関内大通り周辺走行ルート

元浜町通りから関内大通りを通過し、走行開始地点に戻るルートを下記に示す。

図 4-10 元浜町通りから本町通り周辺の走行ルート

3) 馬車道通り～本町通り走行ルート

馬車道通りから入船通りを通過し、本町通りを経由して走行開始地点に戻るルートを下記に示す。

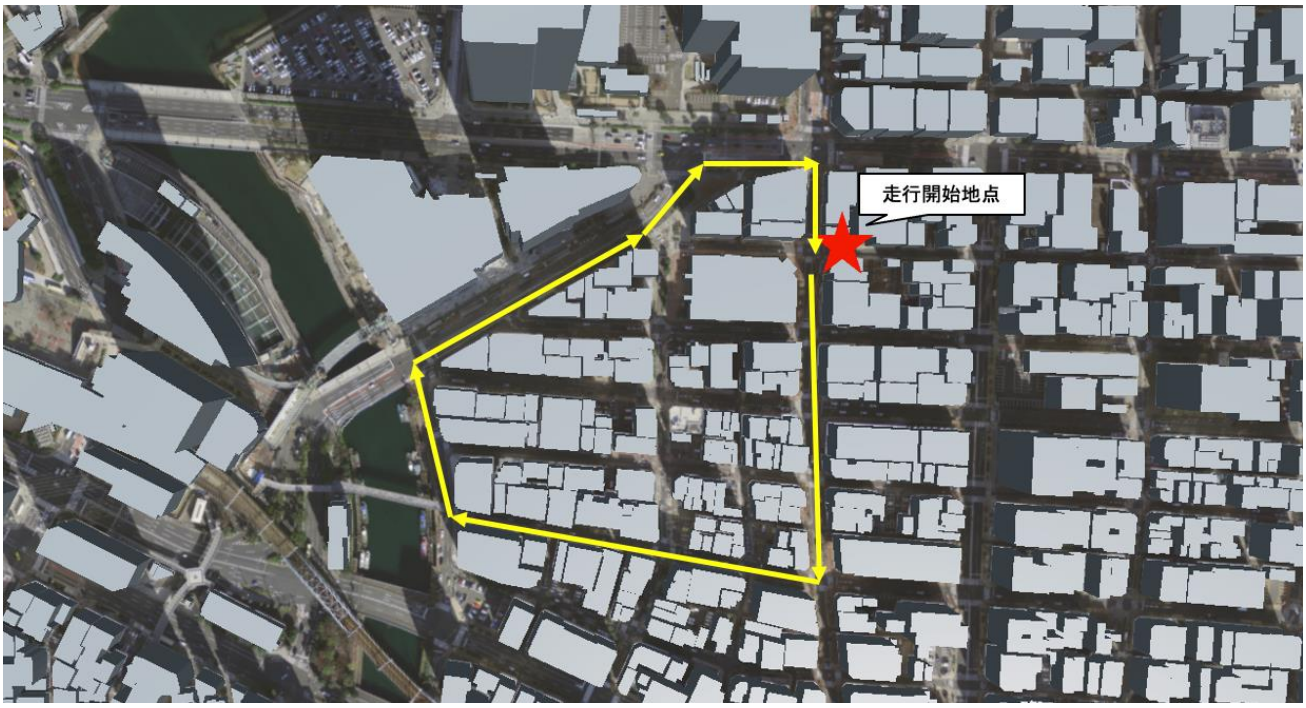


図 4-11 馬車道通りから本町通り周辺の走行ルート

4) みなとみらい地区　リアルマップ

みなとみらい地区では「元浜町通り～関内大通り周辺走行ルート」「馬車道通り～本町通り走行ルート」において、検証に利用するハードウェア構成と同じ機材を用いて、点群マップの事前取得を行った。

図 4-12 リアルマップ取得と PCLG マップ精度検証の両方に利用した車両

リアルマップ生成（システム機能【FN601】）を用いてみなとみらい地区の点群マップを生成した。

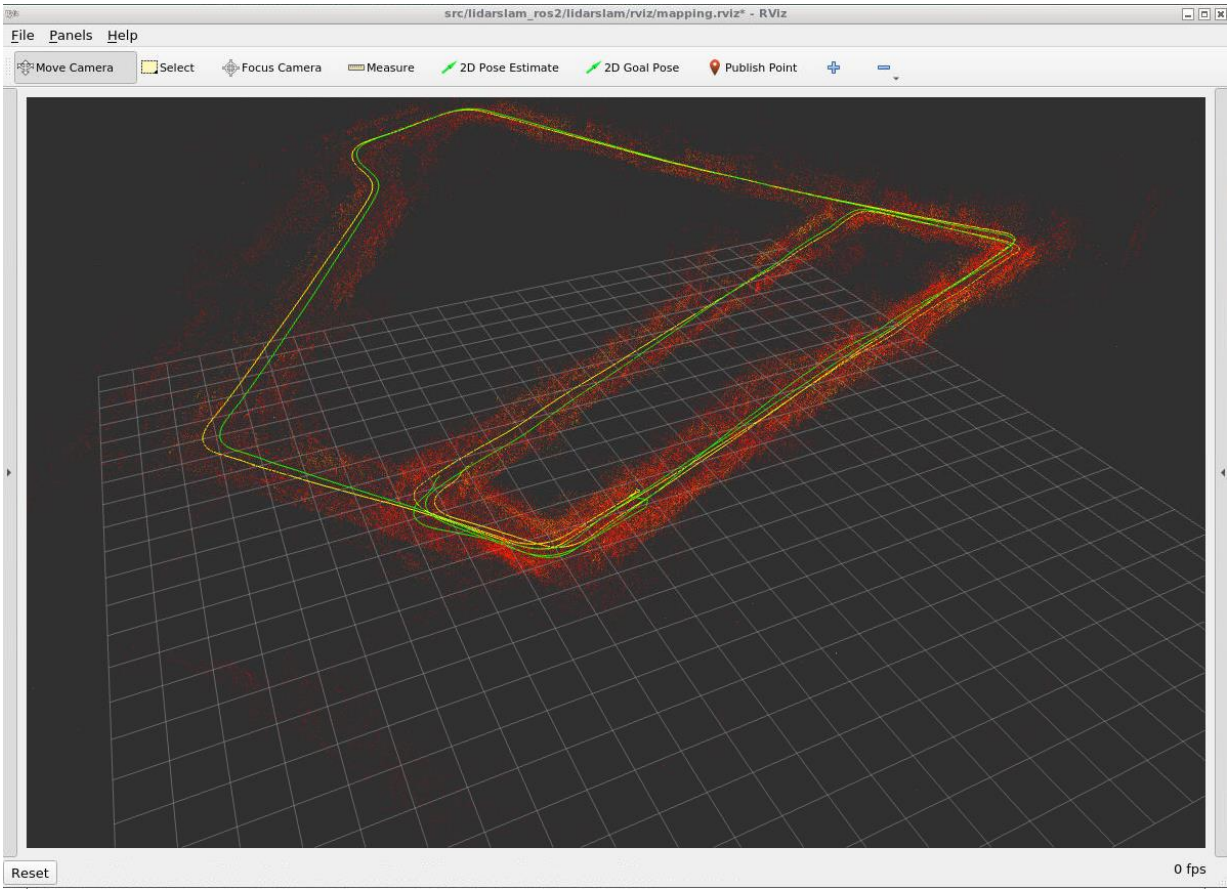


図 4-13 みなとみらい地区のリアルマップ取得の様子

元浜町通りから関内大通りを通過し、元浜町通りに戻るルートでのリアルマップを下記に示す。

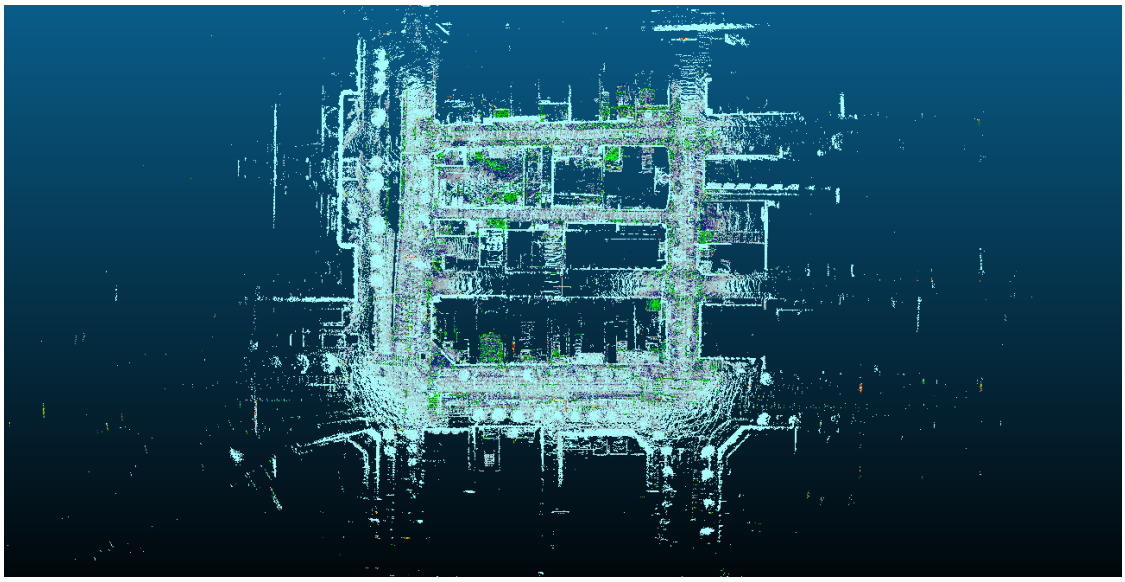


図 4-14 元浜町通り～関内大通り周辺走行ルート　リアルマップ

馬車道通りから本町通りを通過し、馬車道通りに戻るルートでのリアルマップを下記に示す。

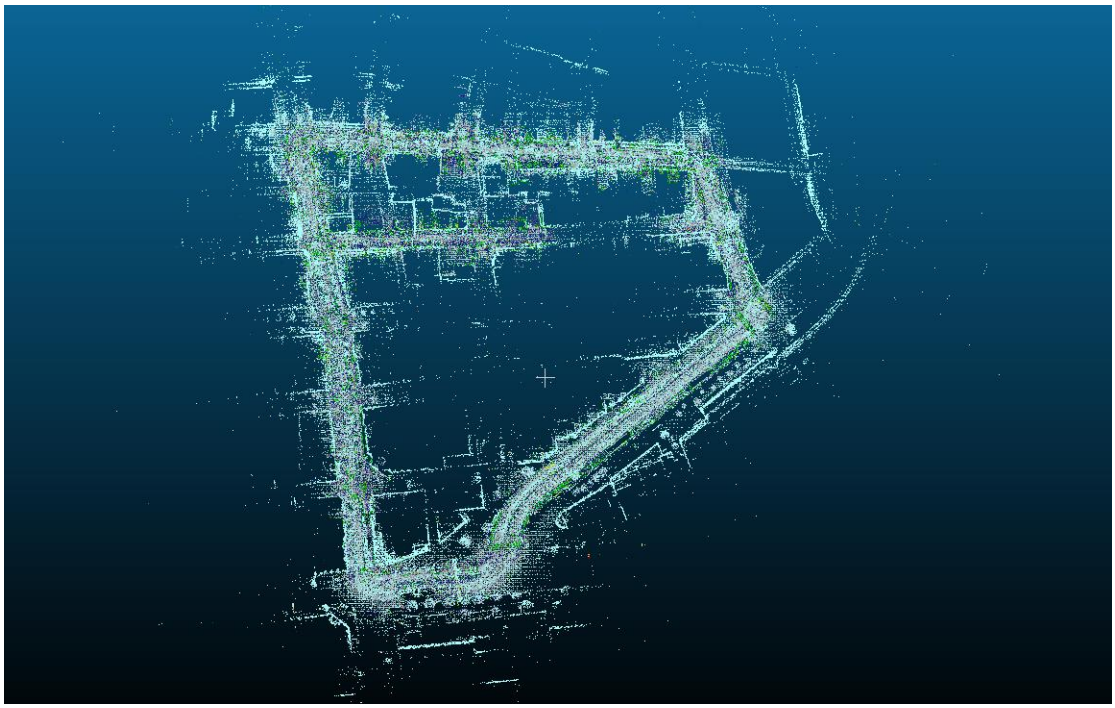


図 4-15 馬車道通り～本町通り走行ルート　リアルマップ

5） みなとみらい地区　PCLG マップ

PCLG を用いて出力した点群マップを下記に示す。本町通りから馬車道通りに走行するルートでは、道路 LOD3 から道路 LOD1 となるルートがあり、道路の高さが変わってしまうため、道路 LOD1 に高さを付与したモデルを利用してルートを設定し、点群マップを出力した。

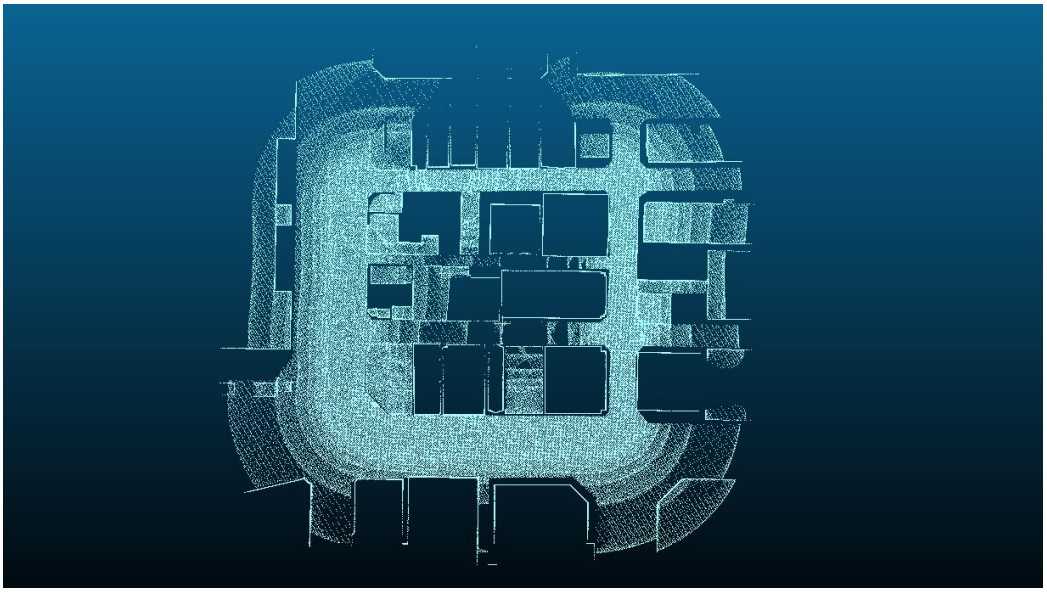


図 4-16 元浜町通り～関内大通り周辺走行ルート　PCLG マップ

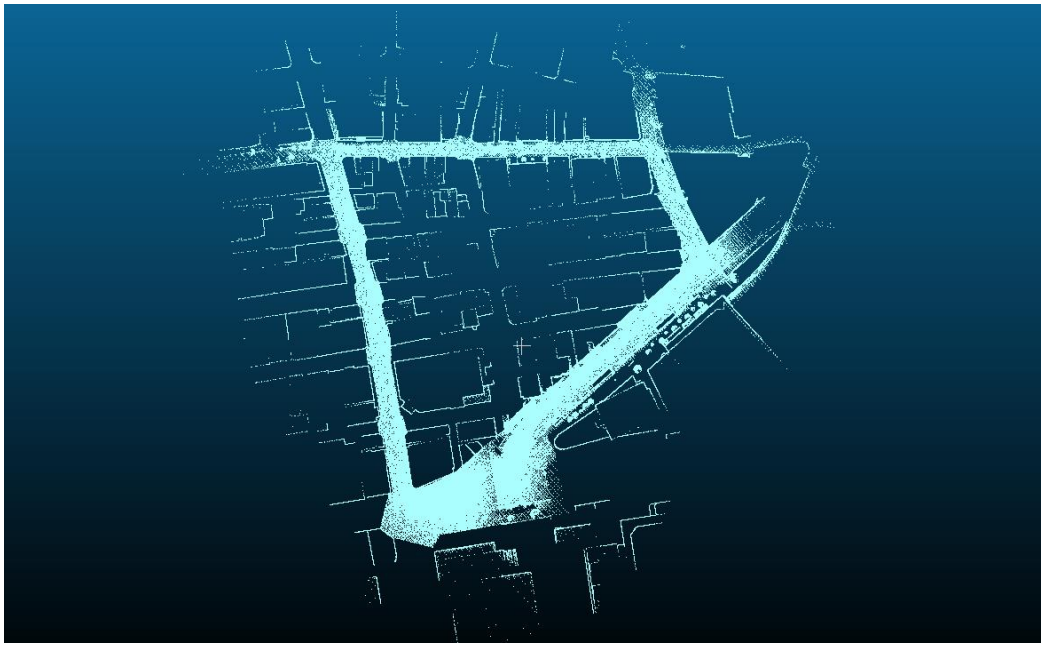


図 4-17 馬車道通り～本町通り走行ルート　PCLG マップ

下記に馬車道通りから本町通りへ走行する際の道路 LOD3 から道路 LOD1 となる箇所を示す。交差点付近の灰色で示したエリアが道路 LOD3 のモデルであり、緑で示したのが道路 LOD1 のモデルとなる。対して赤で示したのが高さを付与した道路 LOD1 のモデルである。

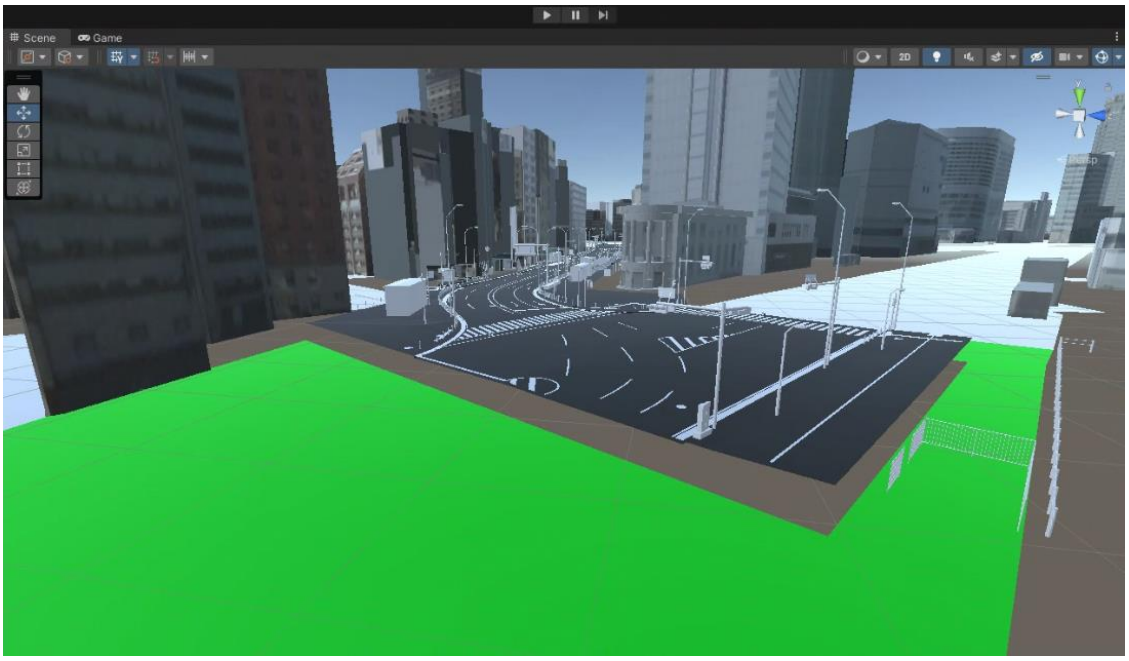


図 4-18 馬車道通り～本町通り走行ルート　PCLG Unity シーンビュー道路 LOD1 高さ付与無し

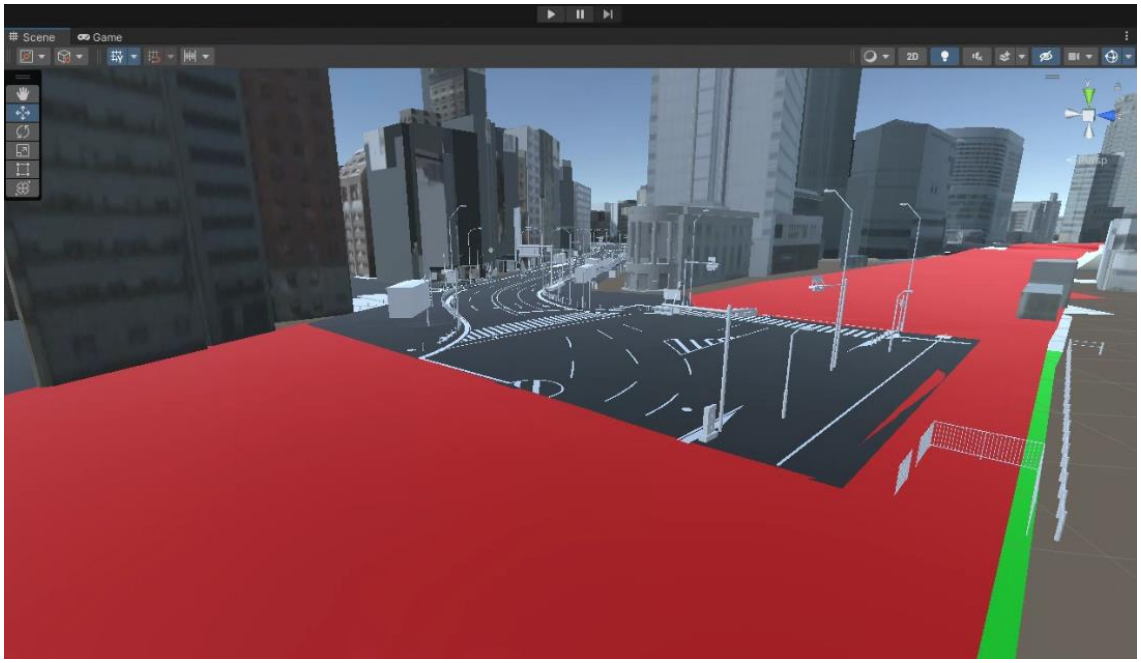


図 4-19 馬車道通り～本町通り走行ルート　PCLG Unity シーンビュー道路 LOD1 高さ付与有り

6） 道路 LOD1 モデルへの高さ付与に関して

道路 LOD1 の高さ付与は、下記の手順で行なった。

1. dem モデルの平滑化処理。ポリゴンメッシュ分割により、既存の dem モデルの平滑化処理を行なった。
2. 道路 LOD1 モデルのレイキャスト処理。道路 LOD1 モデルから平滑化処理を行なった dem モデルに対しレイキャストを行い、ヒットしたポイントに道路 LOD1 モデルの頂点を移動した。

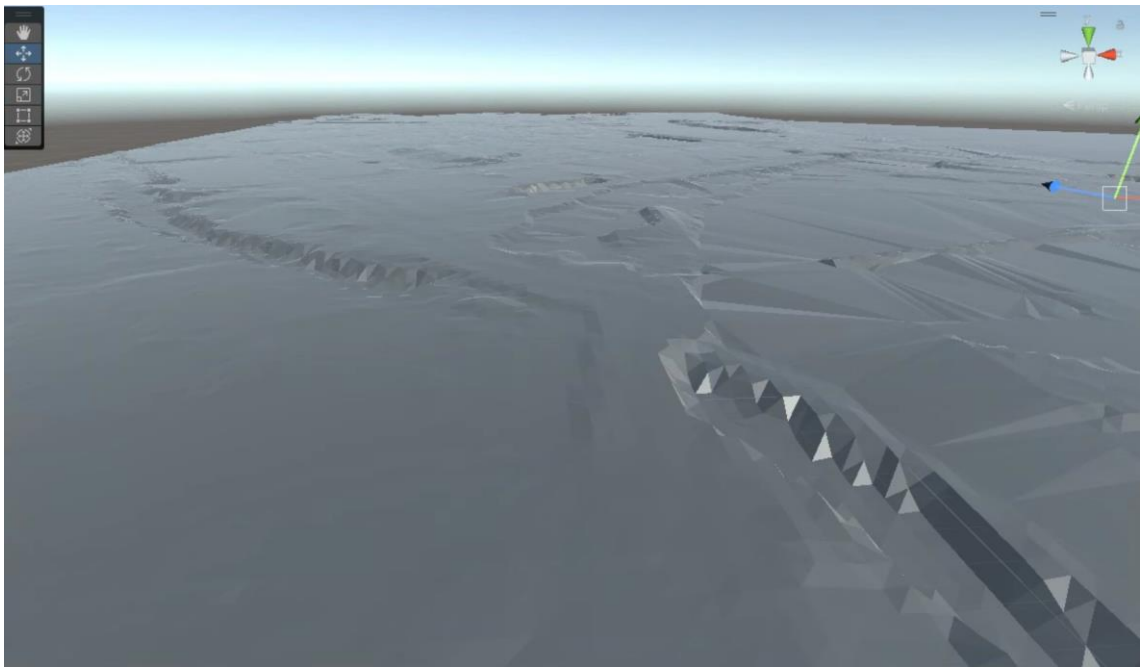


図 4-20 平滑化処理前の dem モデル

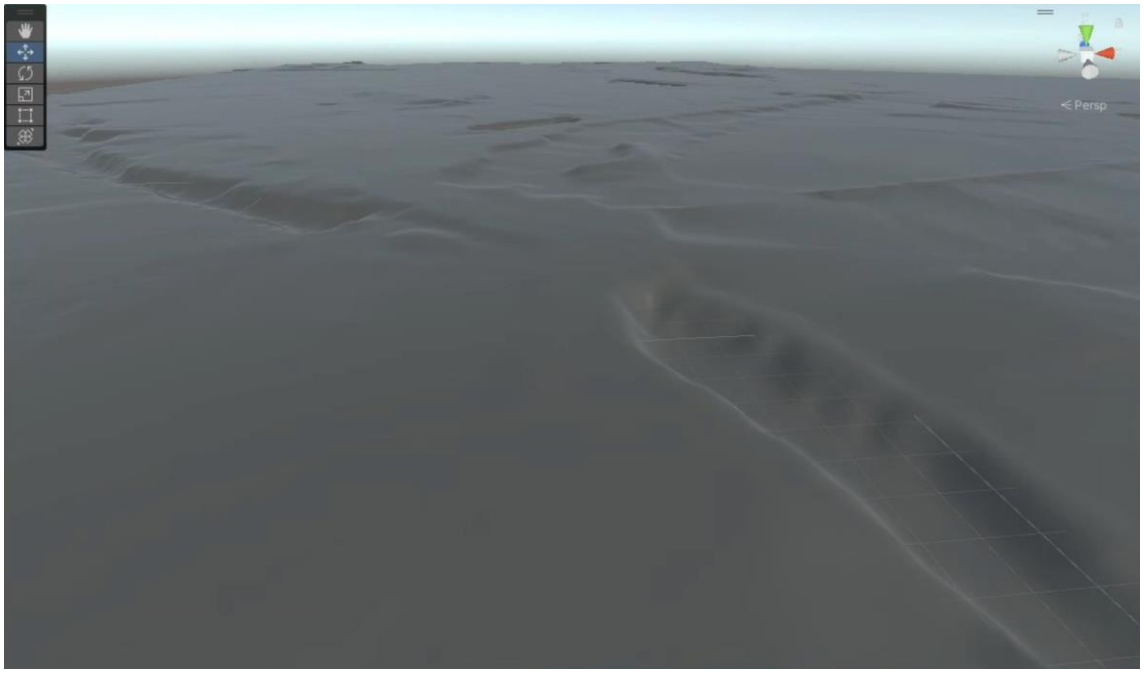


図 4-21 平滑化処理後の dem モデル

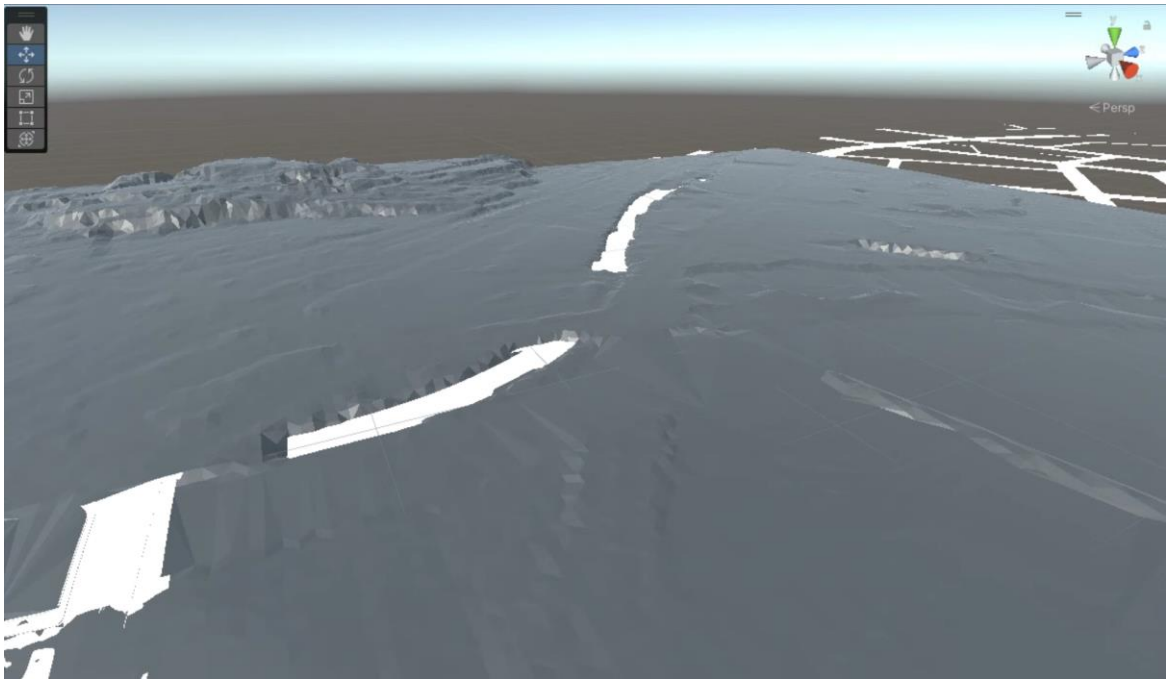


図 4-22 レイキャスト処理前の道路 LOD1 モデル　dem モデルに道路モデルが埋まる

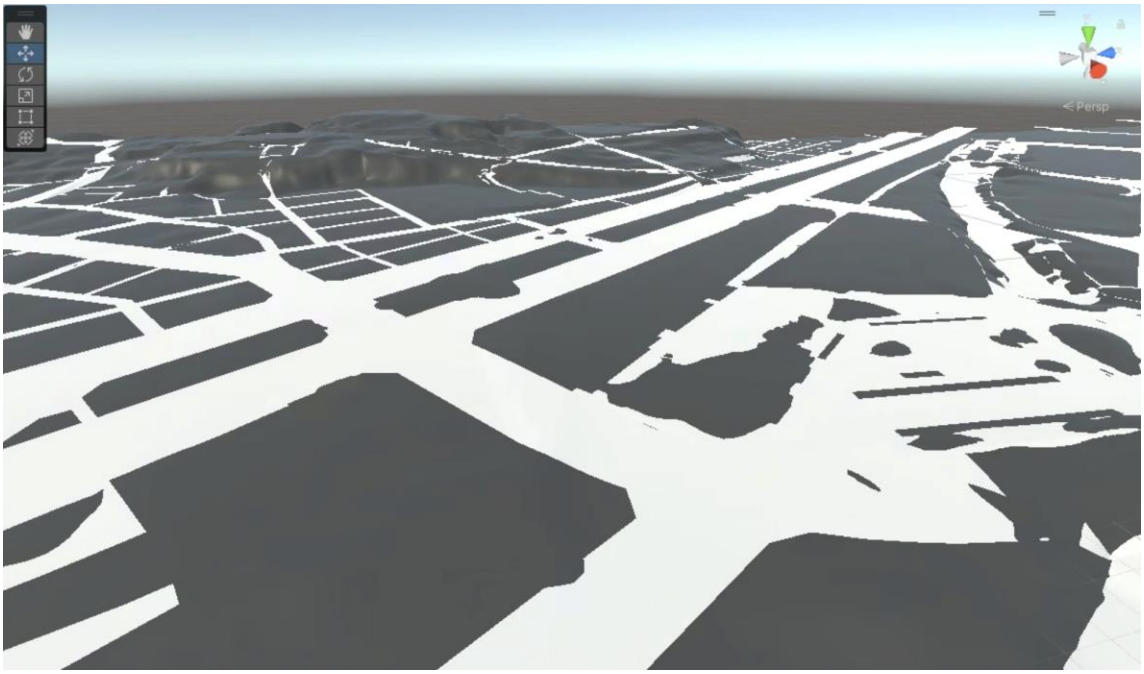


図 4-23 レイキャスト処理後の道路 LOD1 モデル dem モデルに対して高さを合わせる

7） 速度別の評価

馬車道通りから本町通りのエリアを時速 20km と時速 30km で走行した際の精度比較を行う。

4-1-4. 検証結果

1） 元浜町通り～関内大通り周辺走行ルート（建築物 LOD1、道路 LOD1 エリア）

みなとみらい地区の元浜町通りから本町通りへのルートでは、走行を開始した路地のエリアでは精度割合が50%を超えたが、大通りに出ると精度割合は 30%以下と大きく減少した。

精度が減少した要因として、万国橋通りでは交通量が多く車に遮られてしまうため、反対側の建築物まで点群センサが届かないこと、またリアルマップでは中央分離帯の特徴点を捉えているのに対し、3D 都市モデルでは中央分離帯が表現されていなかったことにより、自己位置推定の正確性を欠いたと考えられる。。

なお、本町通りから関内大通りの路地に入ると、自己位置水底の精度割合が増加した。増加した要因は、道路の両側の建築物の特徴点を認識したためである。

元浜町通りから関内大通りの走行ルートを下記に示す。

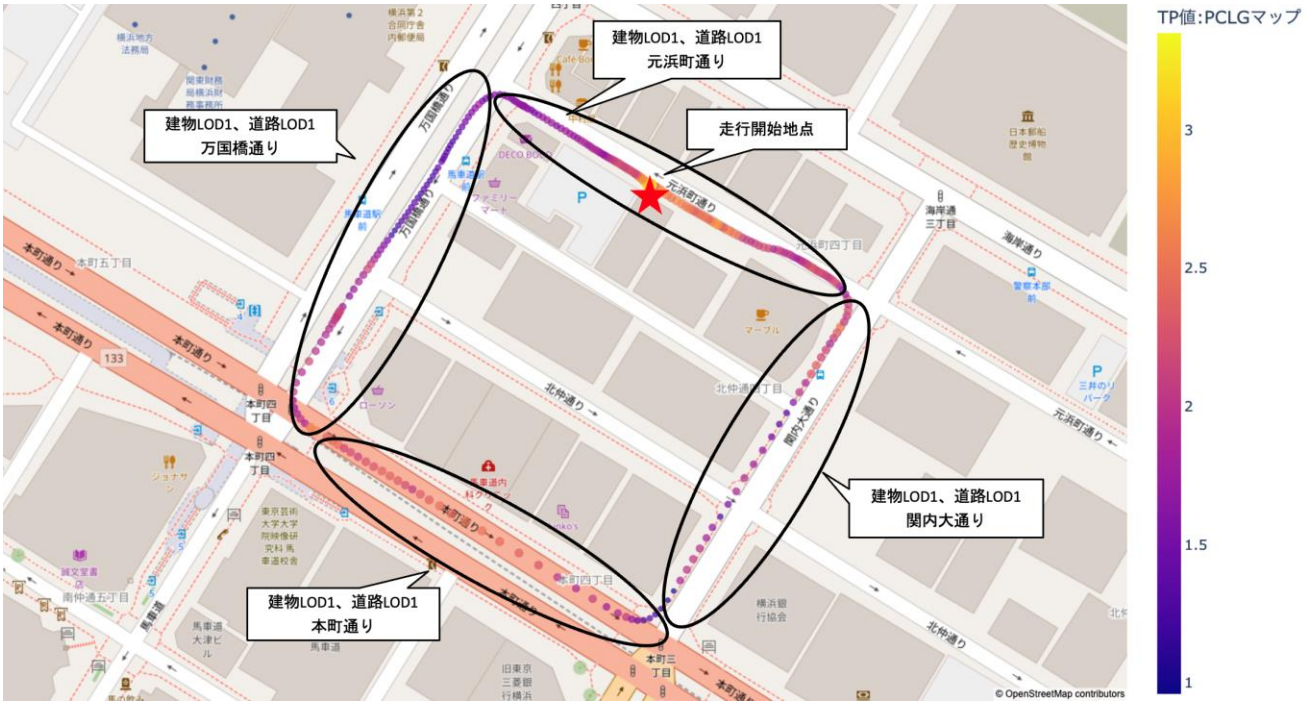


図 4-30 元浜町通りエリアの走行軌跡と 3D 都市モデル整備状況

元浜町通りから本町通りを通過し、関内大通りを走行した点群マッチングの精度割合の推移を下記の図の赤線で示す。青色がリアルマップを用いた TP 値の推移を、緑色が PCLG マップを用いた TP 値の推移を、赤色でマッチング精度割合を示している。

元浜町通りでは走行開始から 70 秒ほどは精度割合が 50%を超えたが、70 秒後に万国橋通りに出ると精度割合は急激に下がった。

要因として、大通りに出た際にリアルマップでは中央分離帯の特徴点を捉えているのに対し、3D 都市モデルでは中央分離帯が表現されておらず、現実と 3D 都市モデルのかい離により自己位置推定の精度が低下したと考えられる。

また、本町通りから関内大通りに入る 270 秒辺りからマッチング精度割合が上昇に転じるのは、3D 都市モデルの建築物によって路地の通りの特徴点が取得しやすいことが要因として挙げられる。

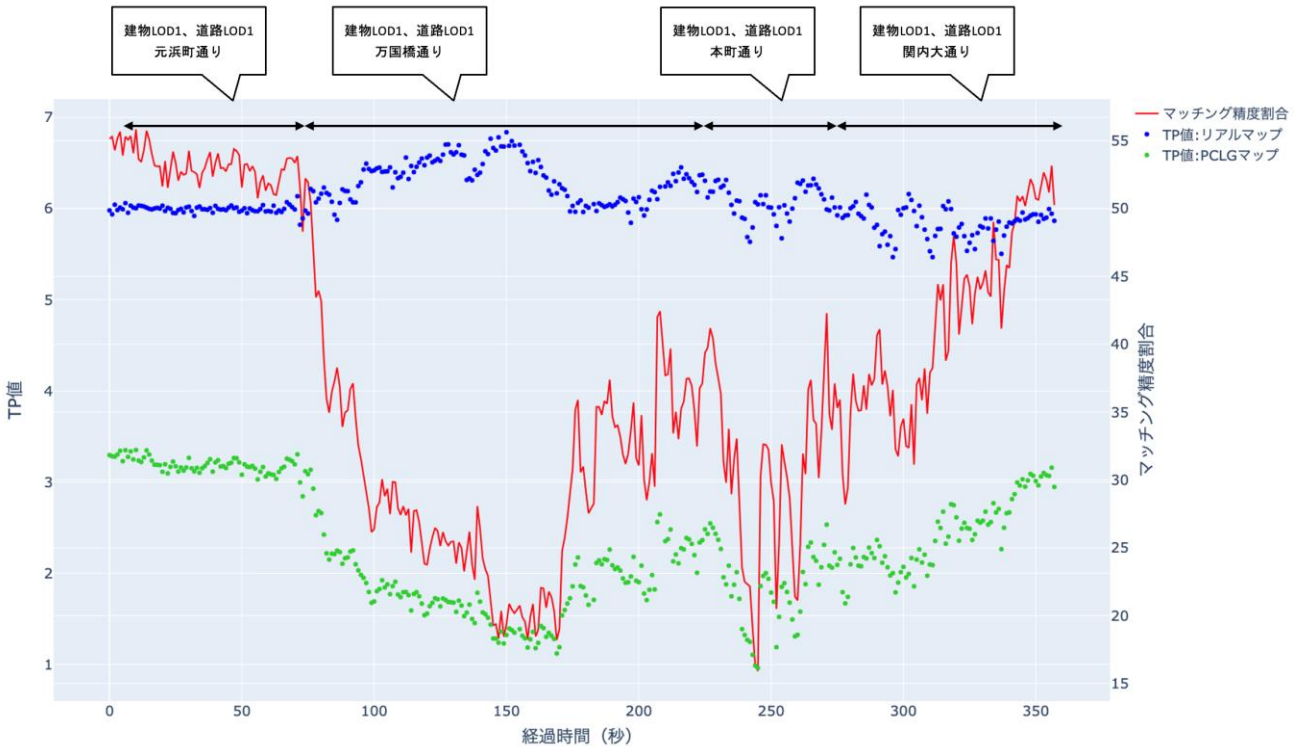


図 4-31 元浜町通りエリア　マッチング精度割合の推移

下記の 2 つの図は、元浜町通りの現地の様子と PCLG 上での 3D 都市モデルの整備状況を比較したものである。元浜町通りは建築物モデルがあり、開始直後に、特徴点をつかみやすかった。

図 4-32 元浜町通り　現地の様子

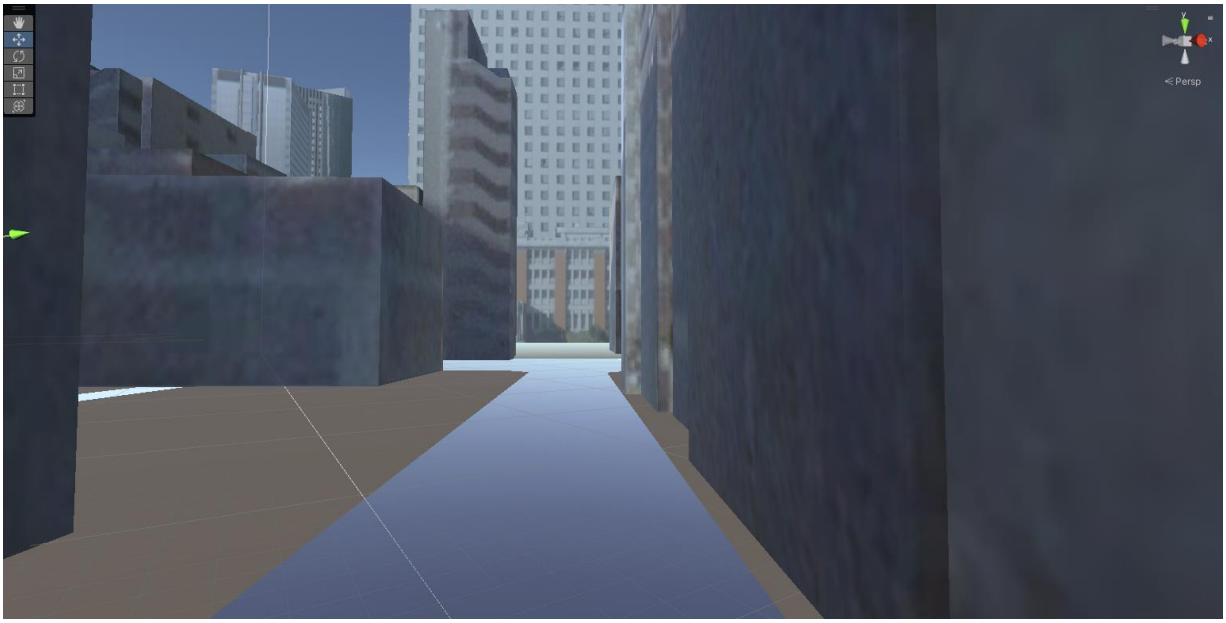


図 4-33 元浜町通り　3D 都市モデル整備状況

下記の 2 つの図は、万国橋通りの現地の様子と PCLG 上での 3D 都市モデルの整備状況を比較したものである。右側の路地エリアでは一般の走行車が少なく、建築物モデルの表現があり、精度を高く保つことができた。一方、大通りは、片側 2 車線、計 4 車線となっている。大通りでは、交通量が多く、バスやトラックなどの大型車に LiDAR が遮られることが多かった。また、車線から反対側の建築物まで距離があり、中央分離帯を特徴として捉えることができなかった。これらの要因により、PCLG マップでは大通りに出ると自己位置推定の精度が減少する傾向となった。

図 4-34 万国橋通り　現地の様子

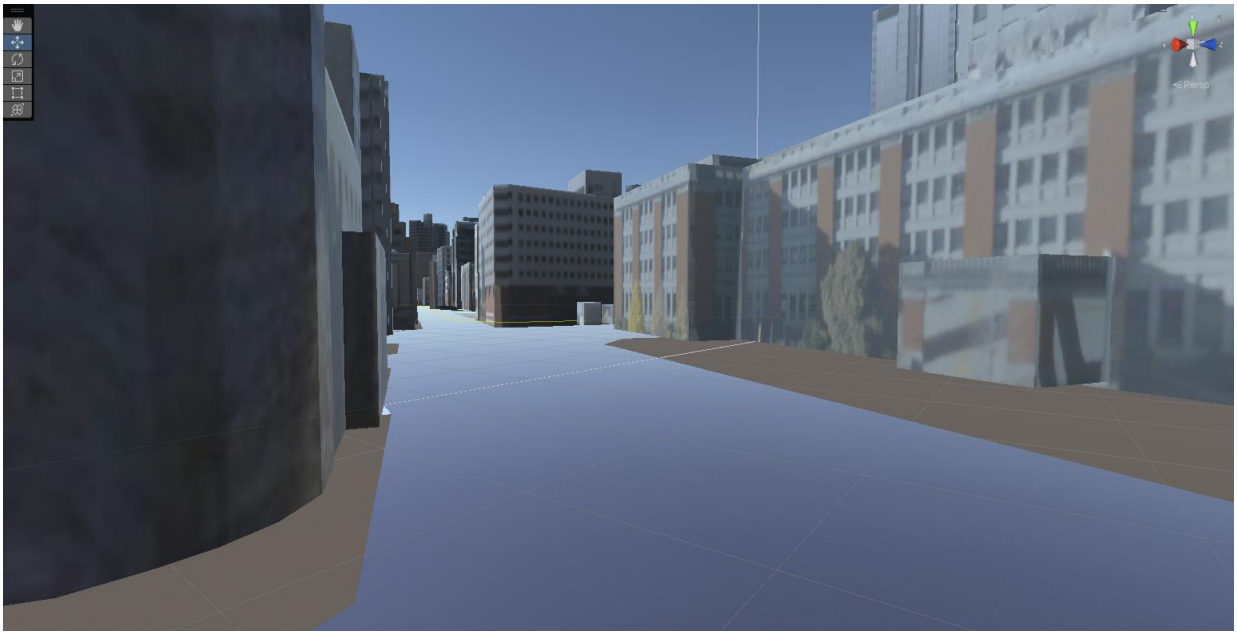


図 4-35 万国橋通り　3D 都市モデル整備状況

本町通りの現地の様子と PCLG 上での 3D 都市モデルの整備状況を下記の図に示す。片側 2 車線となり、交通量も多く、精度割合は 15%から 30%の値の推移となった。

図 4-36 本町通り 現地の様子

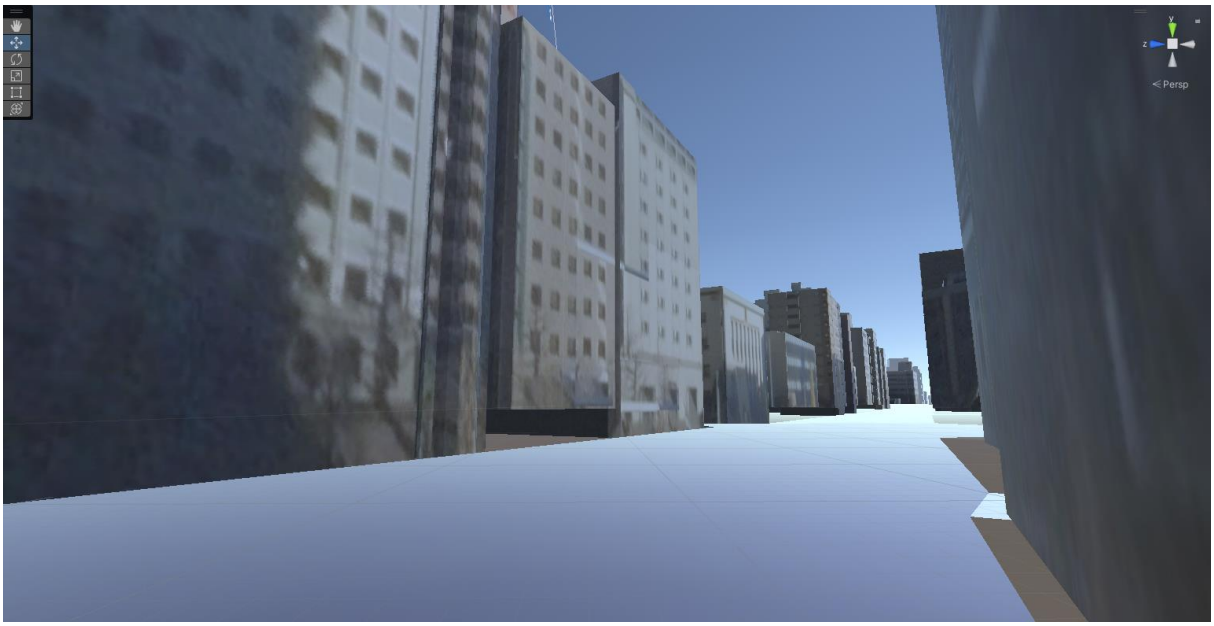


図 4-37 本町通り 3D 都市モデル整備状況

下記の 2 つの図は、関内大通りの現地の様子と PCLG 上での 3D 都市モデルの整備状況を比較したものである。路上駐車の車が多いが、道幅が狭くなったことで建築物の特徴点が捉えやすくなった。点群マッチングの精度割合は 30%から 50%で推移した。

図 4-38 関内大通り 現地の様子

図 4-39 関内大通り 3D 都市モデル整備状況

2） 馬車道通り（建築物 LOD2、道路 LOD1 エリア）
3） 本町通り走行ルート（建築物 LOD3、道路 LOD3、都市設備 LOD3）
※本ルートは継続して走行することで、マッチング精度を比較した。同章で以下に記載

みなとみらい地区の馬車道通り入り口が走行開始地点である。ここから入船通りへ向かうルートは、建築物 LOD2, 道路 LOD1 の整備された路地である。ここでのマッチング精度割合は 30%から 50%ほどであった。
馬車道通りから入船通りへ向かうルートは、建築物 LOD3、道路 LOD3、都市設備 LOD3 が整備された路地である。マッチング精度割合は増加し、40%から 70％の結果となった。
入船通りを抜けて本町通りへ向かうルートは、右側のみ建築物がある。そのため、左側から地物の特徴を捉えることが難しかった。マッチング精度割合は減少し、45%から 50%の推移となった。その後、本町通りに出ると、精度割合はやや減少し、40%辺りの推移状況となった。
本町通りを直進し、走行終了地点へ向かうルートは、建築物 LOD3、道路 LOD3、都市設備 LOD3 から建築物 LOD2、道路 LOD1 のエリアに変化する。エリアを境に、精度割合は一気に減少し、ほぼ 0%に近い数値となった。

馬車道通りから本町通りの走行ルートを下記に示す。

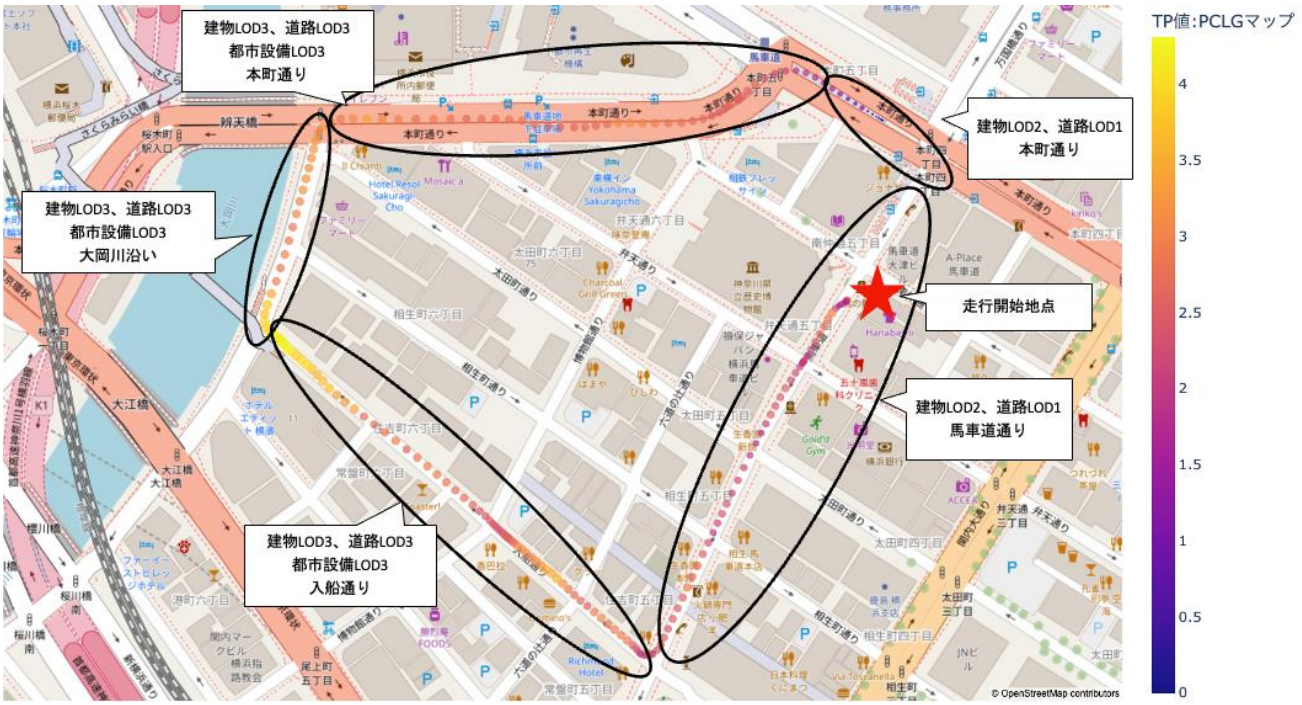


図 4-40 馬車道通り～本町通り　走行ルートと 3D 都市モデル整備状況

馬車道通りから入船通りを通過し、本町通りを走行した点群マッチングの精度割合の推移を下記の図の赤線で示す。青色がリアルマップを用いた TP 値の推移を、緑色が PCLG マップを用いた TP 値の推移を、赤色でマッチング精度割合を示している。
走行開始時から 60 秒ほどは馬車道通りのルート、60 秒から 140 秒ほどは入船通り前半のルート、140 秒ほど

で入船通り後半のルート、170 秒から大岡川沿いのルート、240 秒付近で本町通りを進み、走行終了するルートとなっている。

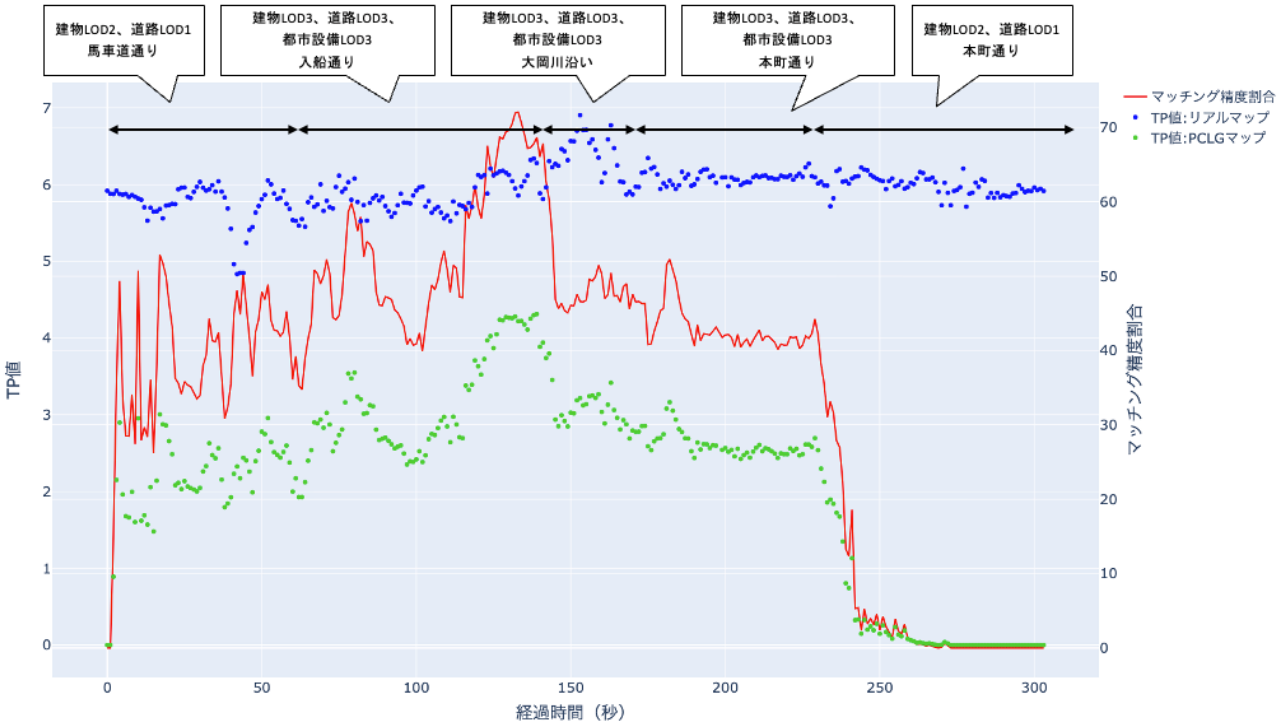


図 4-41 馬車道通りから入船通り　マッチング精度割合の推移

下記の 2 つの図は、馬車道通りの現地の様子と PCLG 上での 3D 都市モデルの整備状況を比較したものである。この通りでは建築物 LOD2 と道路 LOD1 が整備されている。都市設備は整備されていないが、両側の建築物の特徴点を捉えられているため、マッチング精度割合は 30%を超えているものと思われる。

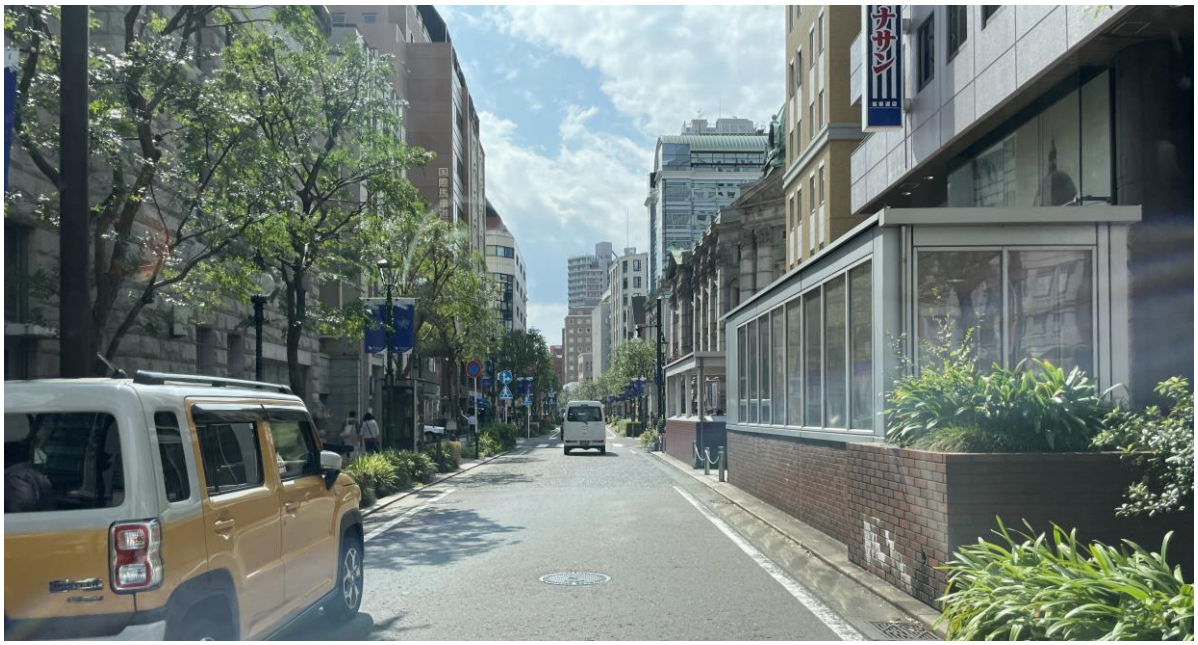


図 4-42 馬車道通り　現地の様子

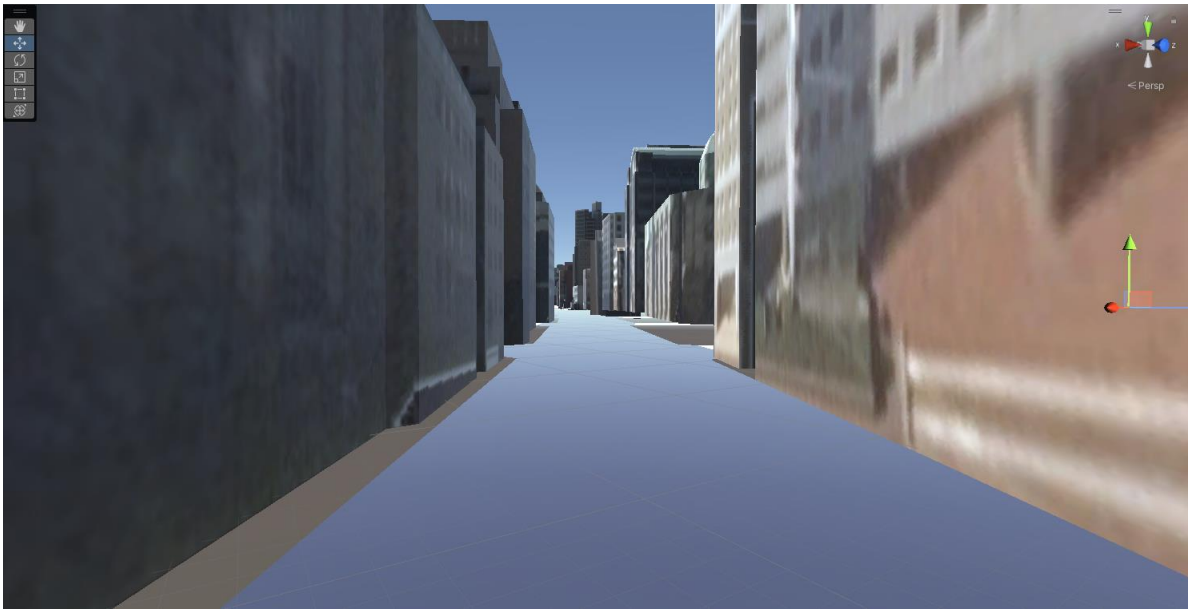


図 4-43 馬車道通り　3D 都市モデル整備状況

下記の 2 つの図は、入船通り前半の現地の様子と PCLG 上での 3D 都市モデルの整備状況を比較したものである。このエリアでは建築物 LOD3、道路 LOD3、都市設備 LOD3 が整備されている。特に標識などの都市設備が忠実に再現されているため、マッチング精度割合は 40%を越す推移となった。

図 4-44　入船通り　現地の様子

図 4-45　入船通り　3D 都市モデルの整備状況

下記の 2 つの図は、入船通り後半の現地の様子と PCLG 上での 3D 都市モデルの整備状況を比較したものである。このエリアがみなとみらい地区で最も精度割合が高かった。3D 都市モデルと現実世界とのかい離が少なく、歩道橋を特徴点として捉えやすかったため、自己位置推定の精度割合が 72%まで増加した。

図 4-46　入船通りから大岡川沿いへ向かうルート　現地の様子

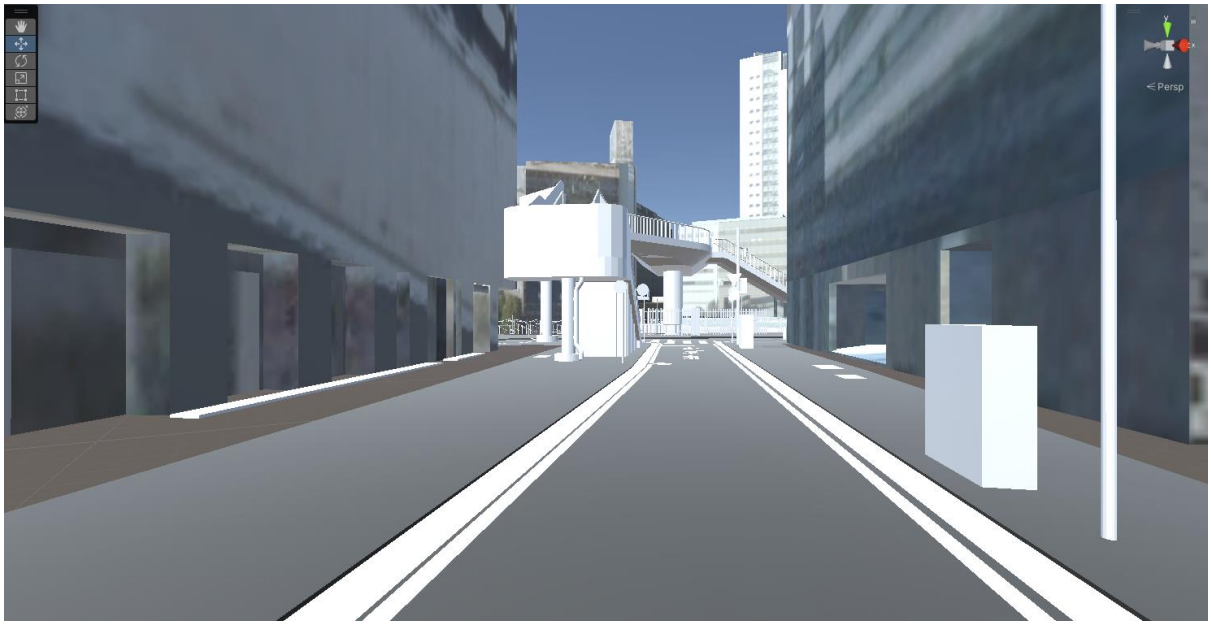


図 4-47　入船通りから本町通りへ向かうルート　3D 都市モデル整備状況

下記の 2 つの図は、大岡川沿いのルートの現地の様子と、PCLG 上での 3D 都市モデル整備状況を比較したものである。川沿いのため、右側のみ建築物があり、左側に建築物がなかった。特徴点が減ってしまい、精度割合は 45%から 50%の推移となった。

図 4-48　大岡川沿い　現地の様子

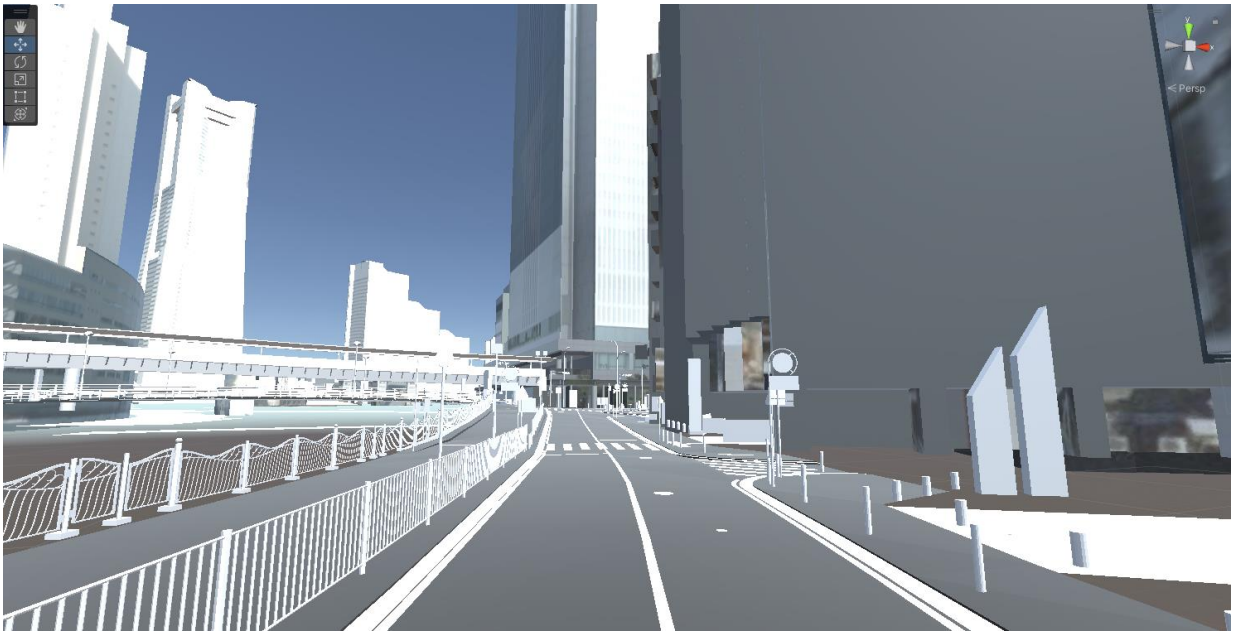


図 4-49　大岡川沿い　3D 都市モデル整備状況

下記の 2 つの図は、本町通りの現地の様子と 3D 都市モデル整備状況を比較したものである。建築物 LOD3、道路 LOD3、都市設備 LOD3 と地物が充実してはいるが、大通りで交通量が多く、点群が遮られることが多かったため、マッチング精度割合は 40%から 45%の間の推移となった。

図 4-50　本町通り　現地の様子

図 4-51　本町通り　3D 都市モデル整備状況

下記の 2 つの図は、本町五丁目交差点付近の現地の様子と 3D 都市モデルの整備状況を下記で比較したものである。交差点の右折前は建築物 LOD3、道路 LOD3、都市設備 LOD3 が整備されているが、右折後は建築物 LOD2、道路 LOD1 のみとなる。
右折開始直後からマッチング精度割合は急激に減少し、右折を終えると 0%台の値となる。片側 3 車線で交通量が非常に多い、都市設備のモデルがないといったことが原因である。

図 4-52　本町五丁目交差点付近　現地の様子

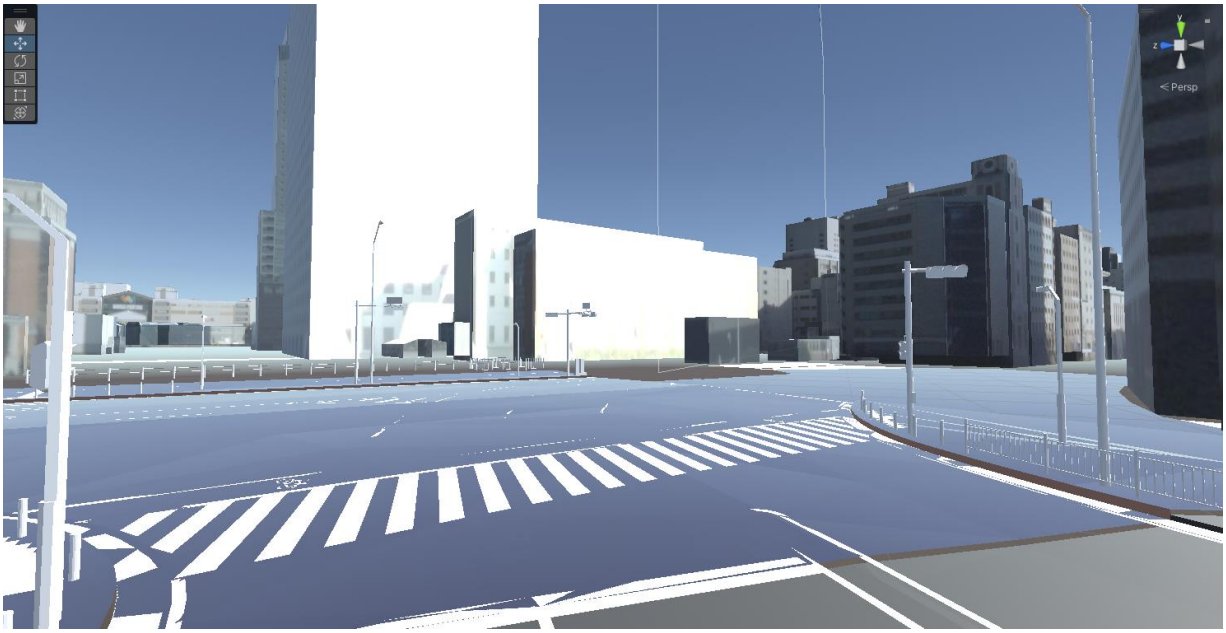


図 4-53　本町五丁目交差点付近　3D 都市モデル整備状況

表 4-2 みなとみらいエリア　検証結果

黄セル：KPI 達成	青セル：KPI 未達

検証内容	評価指標・KPI	目標値	結果		示唆
			項目	評価値	
建築物 LOD1、道路 LOD1	マッチング精度割合	30%以上	建築物密集地	50～55%	● 建築物が密集している路地では車両の両側の建築物モデルの特徴点が取りやすく、精度割合は KPI を達成した ● 本町通りは片側 2 車線、計 4 車線の大通りであり、道が広く走行車線側のみの片側の建築物の特徴点しか捉えられなかった。原因は、交通量が多く建築物までの点群が遮られたこと、中央分離帯を特徴点として捉えてしまうことが挙げられた
			大通り	15～30%	
建築物 LOD2、道路 LOD1 エリア	マッチング精度割合	30%以上	建築物密集地	30～50%	● 建築物が密集している路地では、建築物 LOD1 のエリアと同じく、両側の建築物モデルの特徴点を捉えることができ、点群マッチングの精度割合は増加した ● 大通りでは、建築物 LOD1 のエリアと同じく、精度割合が減少した
			大通り	0～30%	
建築物 LOD3、道路 LOD3、都市設備 LOD3	マッチング精度割合	70%以上	建築物密集地	40～60%	● 建築物密集値では 40%～60%と、建築物 LOD2 のエリアよりも精度割合が増加した ● 入船通り沿いエリアでは、本検証で一番高い精度を示した。要因として、地物と現実空間に相違があまりないことが挙げられた ● 大通りになると精度は減少した
			入船通り沿い	70～72%	
			大通り	40～50%	

4） 速度別の評価

下記の 2 つの図は、馬車道通りから本町通りまでのルートを最大時速 25km と時速 30km で走行した際の点群マッチングの精度割合の推移をそれぞれ示したものである。

結果として、最大時速 20km と 30km で走行した際の計測結果に、大きな差異は見受けられず、同じようなグラフの推移となった。時速 30km 以下では速度差による精度割合に差はないと言える。

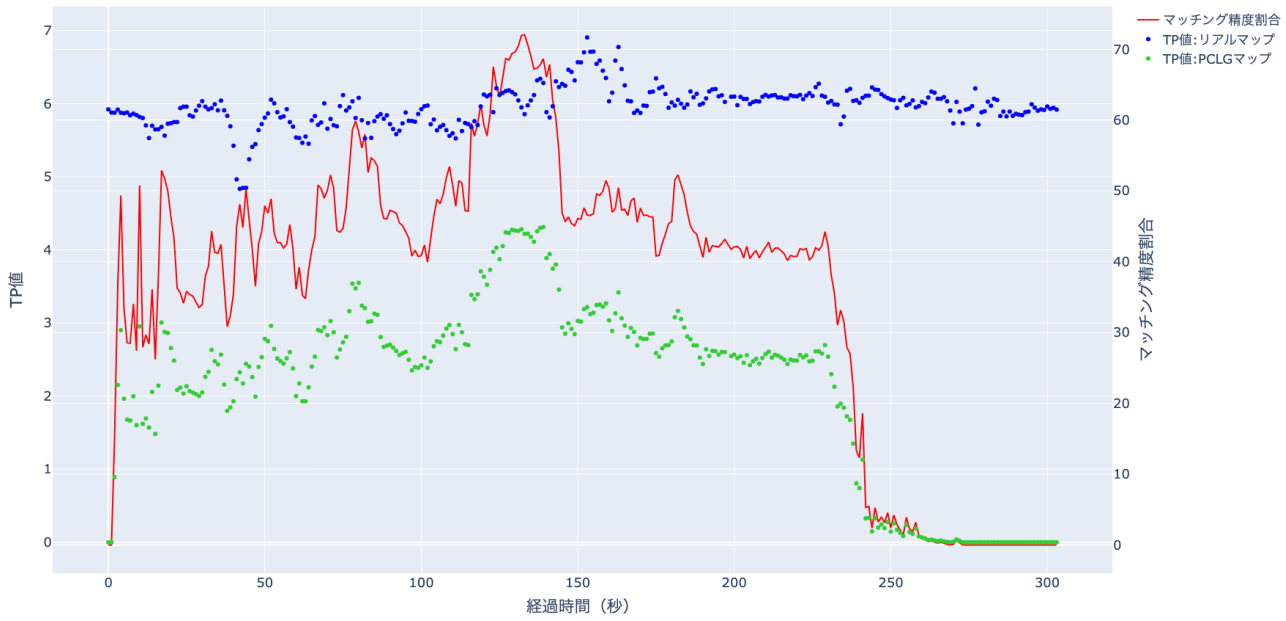


図 4-60　最大時速 20km で走行した際の推移

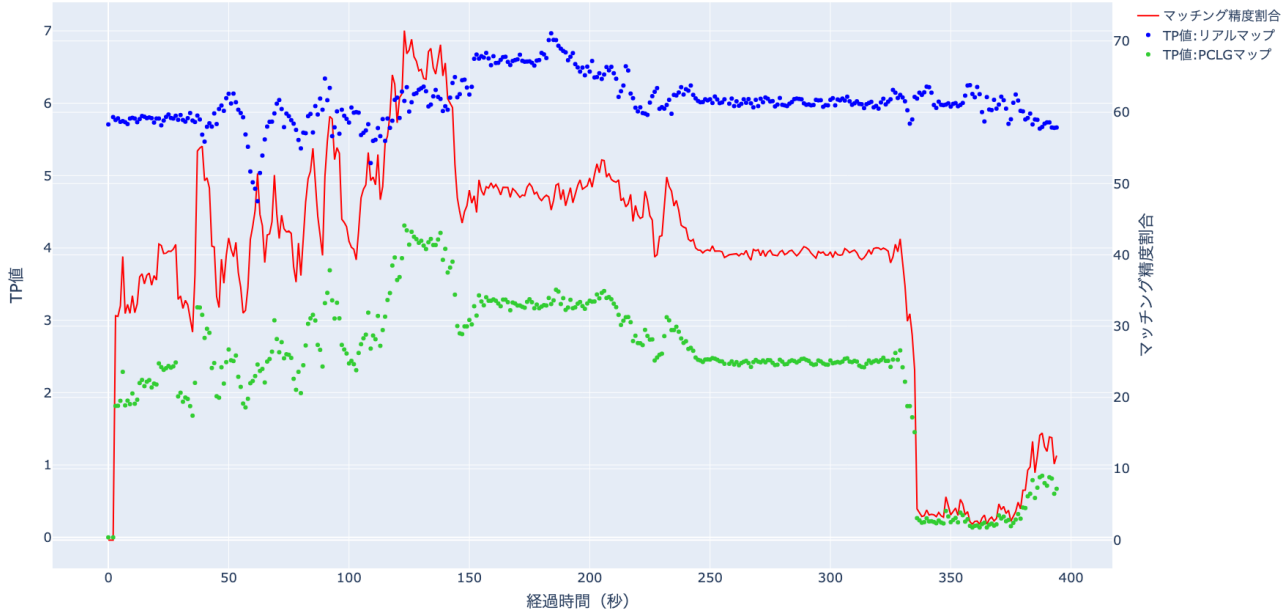


図 4-61　最大時速 30km で走行した際の推移

4-2. 自己位置推定の精度割合検証　大阪府大阪市　舞洲地区

- 大阪府大阪市舞洲地区で行った自己位置推定の精度割合の検証結果を下記にまとめる。

4-2-1. 検証目的

- 大阪府大阪市　舞洲地区において、PLATEAU PCL Generator から出力された点群マップを用いた自己位置推定の精度割合と、実際に車両を走行させて取得したリアルマップの自己位置推定の精度割合の比較を行う。
- 舞洲地区では特徴の異なる下記の 2 つの点群マップを出力し、リアルマップとの精度割合の比較を行う。
  1. ロッジ舞洲コース（建築物 LOD2、道路 LOD3、都市設備 LOD3、植生 LOD3）
  2. 夢舞大橋コース（道路 LOD3、都市設備 LOD3）
- 舞洲地区では交通量が多いみなとみらい地区に比べて交通量が少なく速度別の検証が行いやすいため、最大速度 15km/h と 30km/h の速度別の走行検証を行う。

4-2-2. KPI

表 4-3 KPI 一覧

No.	評価指標・KPI	目標値	目標値の設定理由	検証方法サマリー
1	点群マッチングの精度割合 建築物 LOD2、道路 LOD3、都市設備 LOD3、植生 LOD3	50%	舞洲エリアは建築物の地物が少なく、みなとみらいエリアに比べ精度が低下すると考えられる	事前取得点群マップとの精度割合データとの比較を基に割合を算出
2	点群マッチングの精度割合 道路 LOD3、都市設備 LOD3	30%	夢舞大橋では交通量が多く、精度割合はみなとみらいエリアと同等になると考えられる	同上
3	速度別の比較	-	（走行速度によるマッチングの精度割合の推移を比較するのみのため、目標値なし）	最大速度 15 ㎞/h と 30 ㎞/h のマッチング精度割合を経過比較

4-2-3. 検証方法と検証シナリオ

1) 点群マッチングの精度割合

【4-1-2.KPI】と同じく、点群マッチングの精度割合の KPI を評価対象とする。

2) ロッジ舞洲（舞洲二丁目）～舞洲緑地前走行ルート

ロッジ舞洲（舞洲二丁目）から舞洲緑地前を通過する走行ルートを下記に示す。

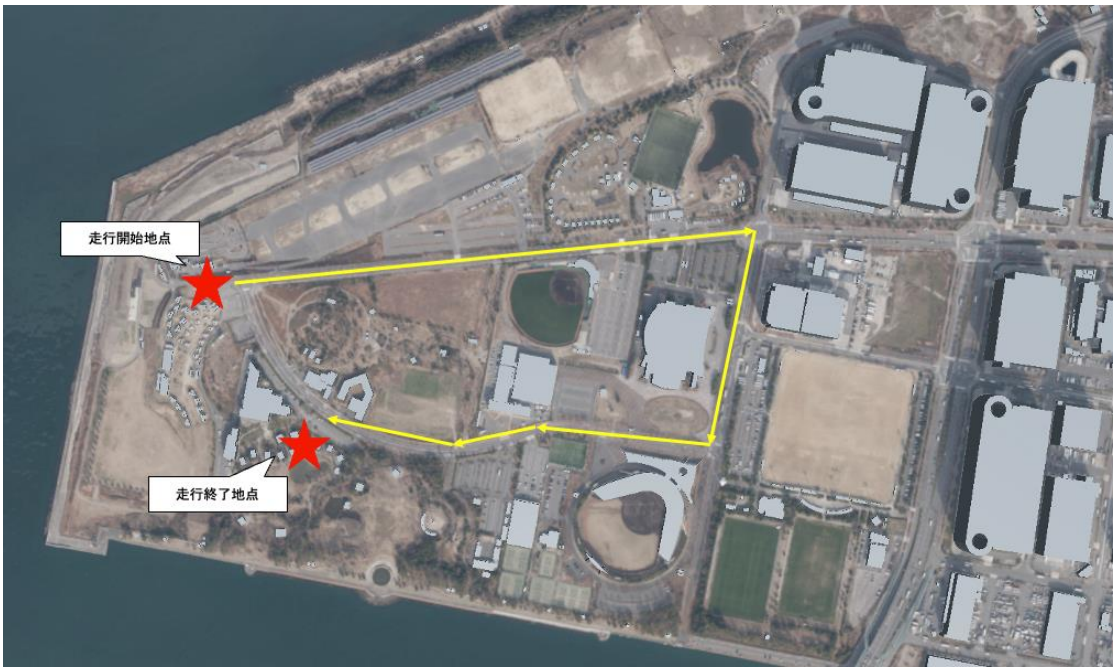


図 4-70　舞洲エリア　走行ルート

3） 夢舞大橋走行ルート

夢舞大橋の手前から夢舞大橋を通過する走行ルートを下記に示す

図 4-71　夢舞大橋　走行ルート

4） 舞洲地区　リアルマップ

舞洲地区では「ロッジ舞洲（舞洲二丁目）～舞洲緑地前走行ルート」「夢舞大橋ルート」において、検証に利用するハードウェア構成と同じ機材を用いて、点群マップの生成を行った。

図 4-72　検証用車両　リアルマップを取得するために夢舞大橋を走行する様子

みなとみらい地区と同様に、リアルマップ生成（システム機能【FN601】）を用いて舞洲地区の点群マップを生成した。

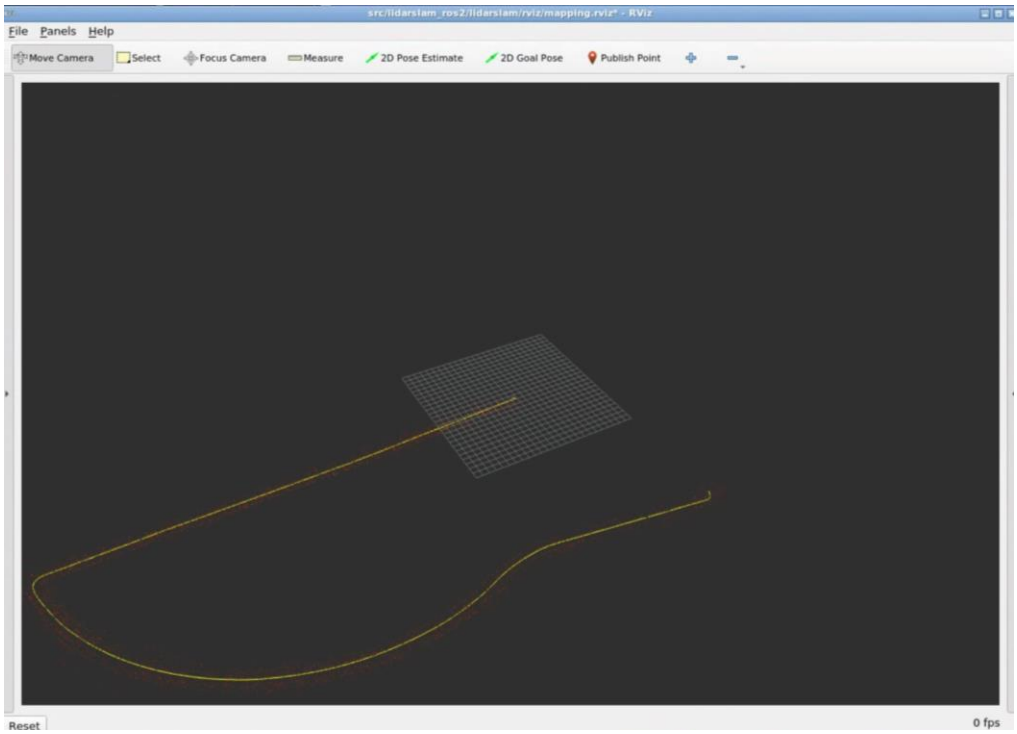


図 4-73　舞洲地区　リアルマップ取得の様子

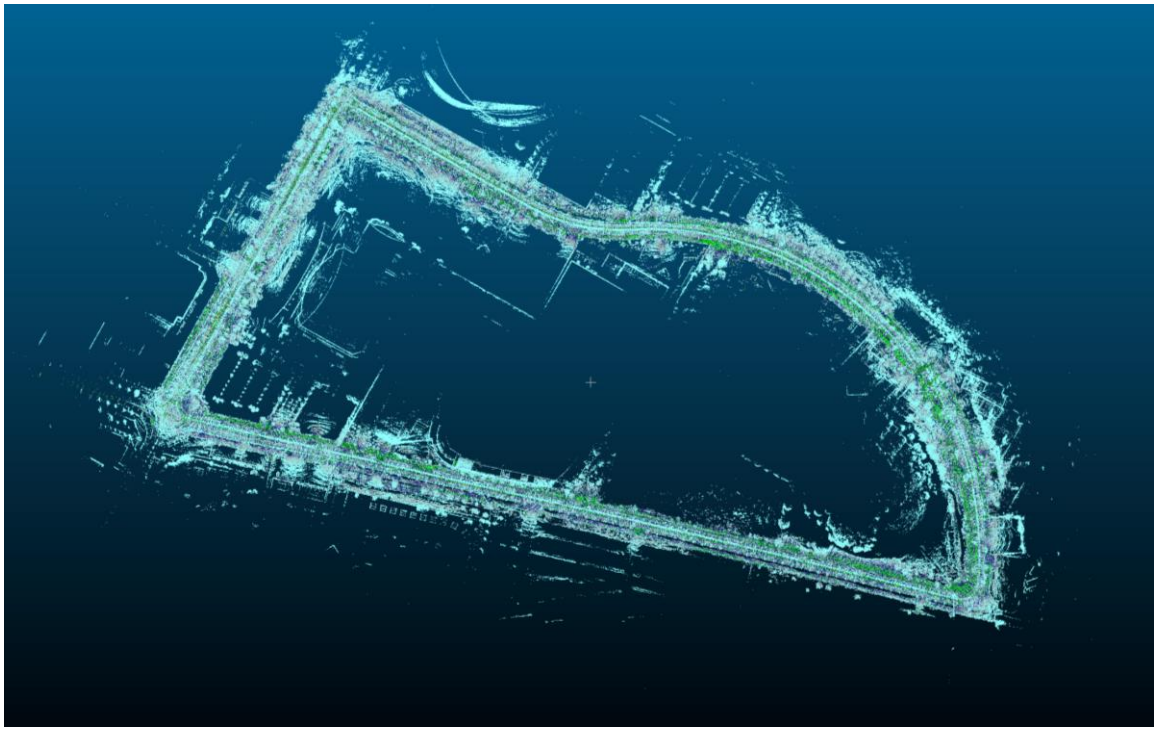


図 4-74　ロッジ舞洲～舞洲緑地前走行ルート　リアルマップ

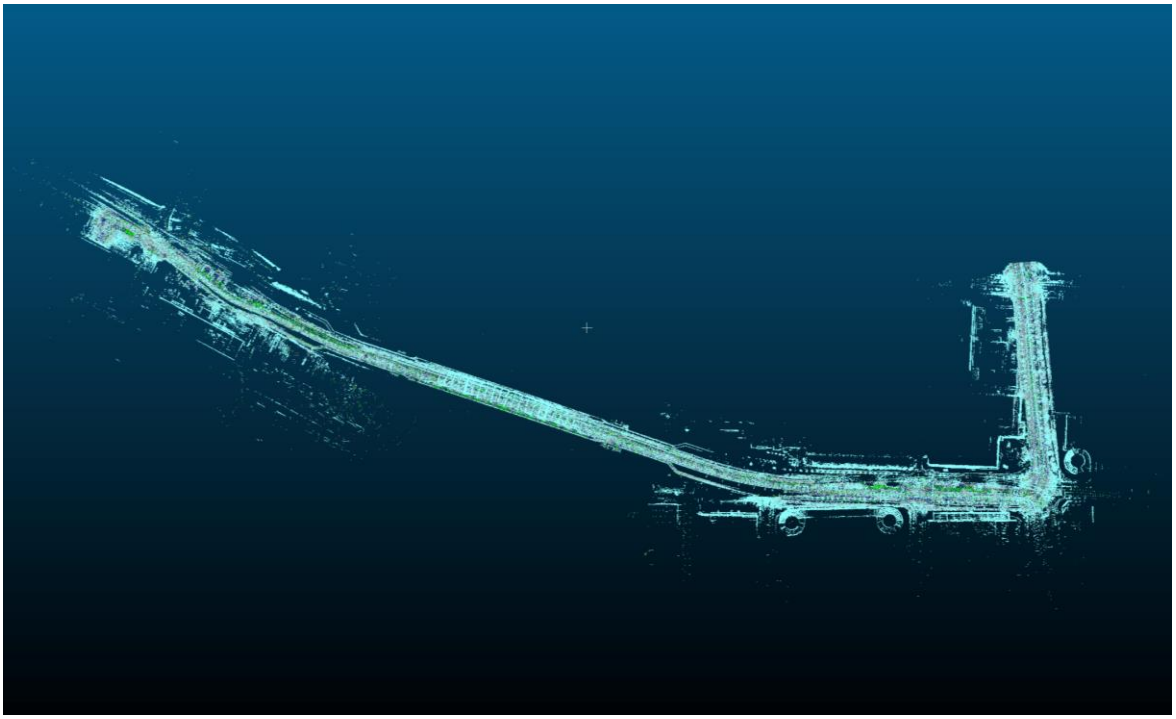


図 4-75　夢舞大橋走行ルート　リアルマップ

5） 舞洲地区　PCLG マップ

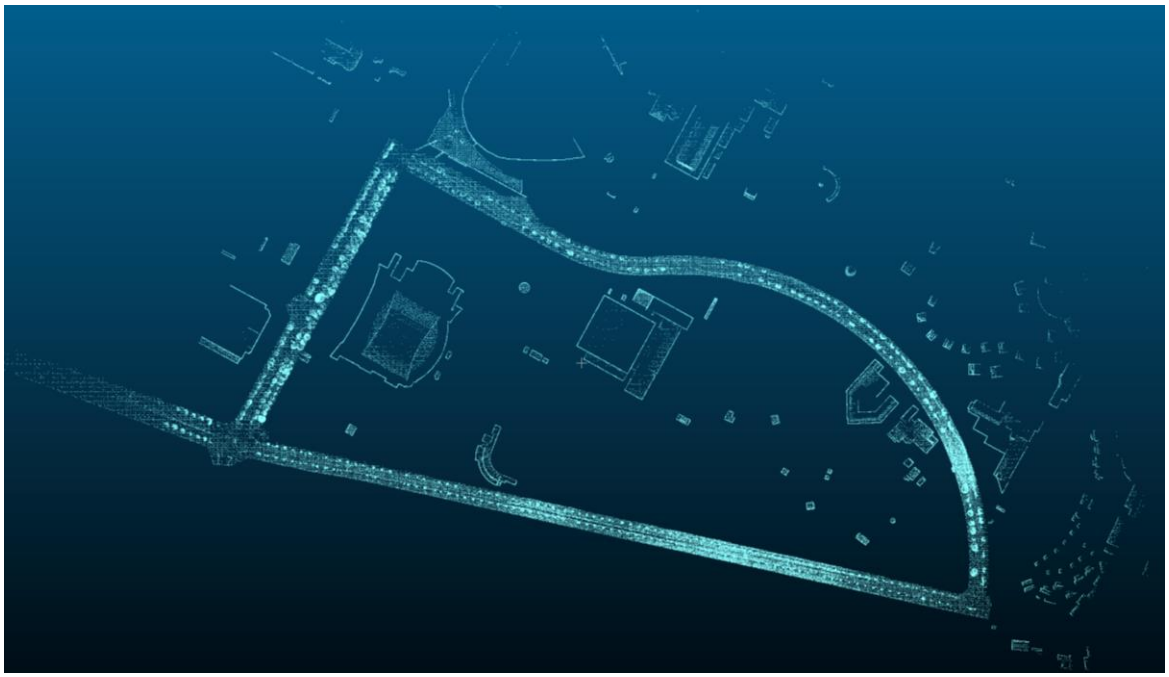


図 4-76　ロッジ舞洲～舞洲緑地前走行ルート　PCLG マップ

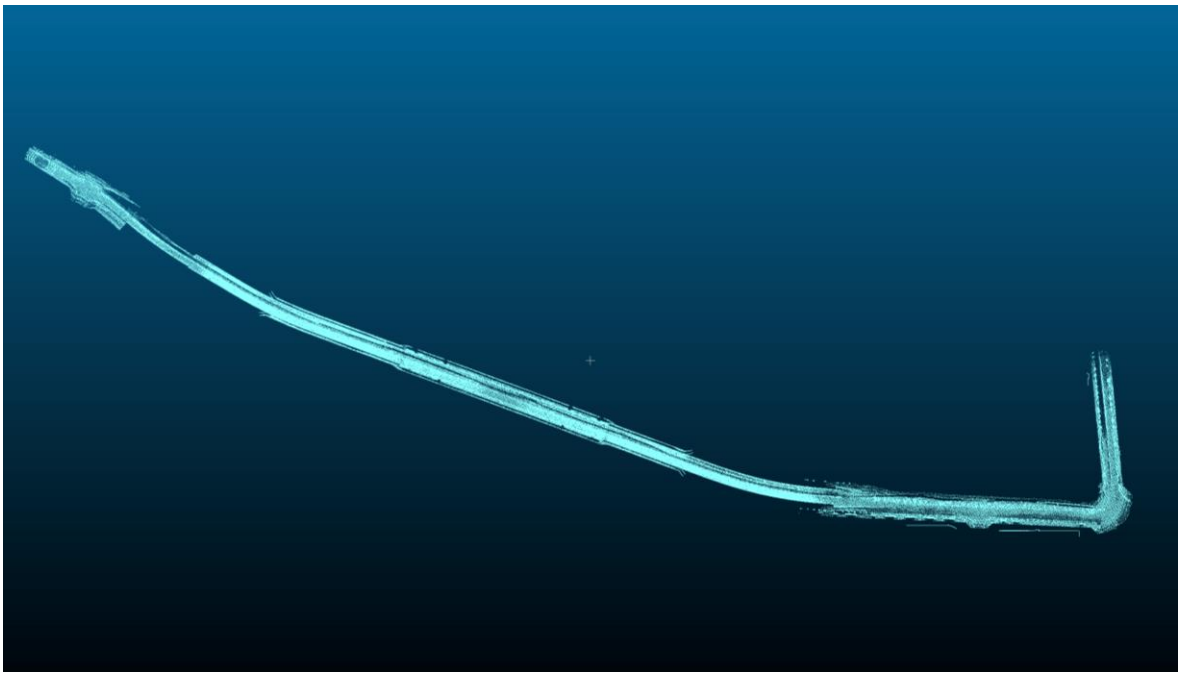


図 4-77　夢舞大橋走行ルート　PCLG マップ

6） 速度別評価

ロッジ舞洲から舞洲緑地前のエリアを時速 15km と時速 30km で走行した際の精度割合の比較を行う。

4-2-4. 検証結果

1) ロッジ舞洲（舞洲二丁目）～舞洲緑地前走行ルート（建築物 LOD2、道路 LOD3、都市設備 LOD3、植生 LOD3 エリア）

本検証では舞洲交差点付近の点群マッチング精度割合が最も高く、舞洲緑地前のカーブのエリアの精度が最も低くなった。

ロッジ舞洲（舞洲二丁目）から舞洲緑地の走行ルートを下記に示す。

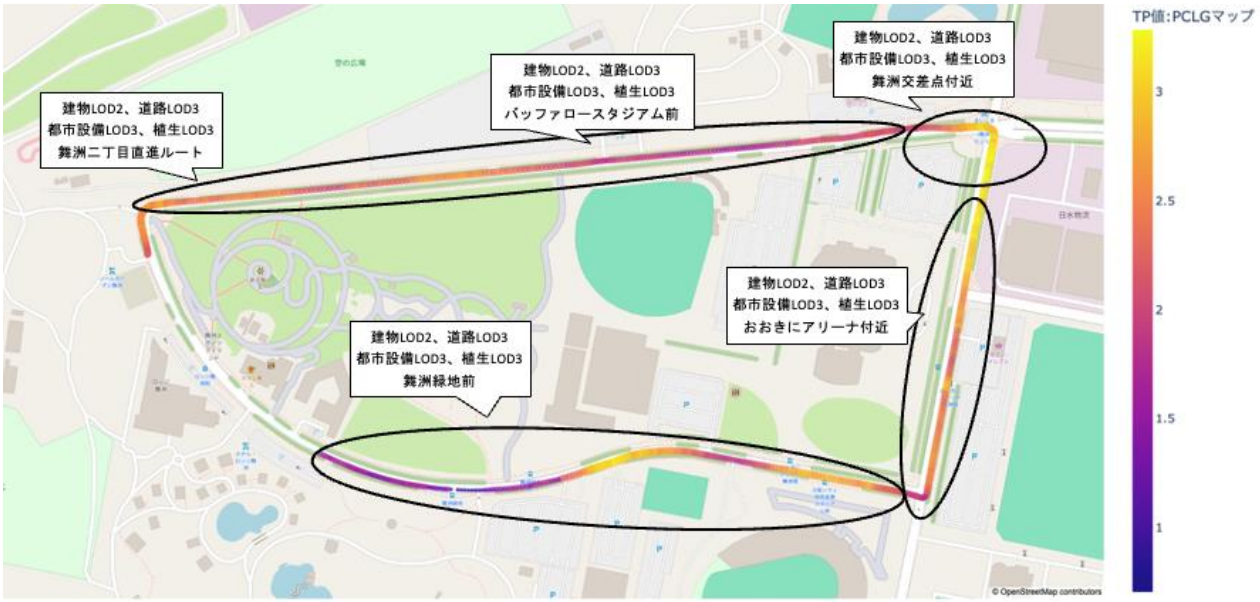


図 4-80　ロッジ舞洲（舞州二丁目）～舞洲緑地前　走行ルートと 3D 都市モデル整備状況

舞洲二丁目から舞洲緑地を走行した点群マッチングの精度割合の推移を下記の図の赤線で示す。青色がリアルマップを用いた TP 値の推移を、緑色が PCLG マップを用いた TP 値の推移を、赤色でマッチング精度割合を示している。

走行開始から 100 秒辺りまでの舞洲二丁目直進ルートでは精度割合が 40%から 50%で推移した。150 秒付近を境に点群マッチングの精度割合が 70%を超えているが、これはリアルマップの精度が急激に下がっていることが要因として挙げられる。

350 秒ほどから舞洲交差点付近を走行し、400 秒で精度割合が 60%を超えた。

その後おおきにアリーナ付近に向かって精度割合は緩く減少し、30%から 55%の間を推移した。

走行終了地点付近の舞洲緑地前では、精度割合が 30%以下に減少した。

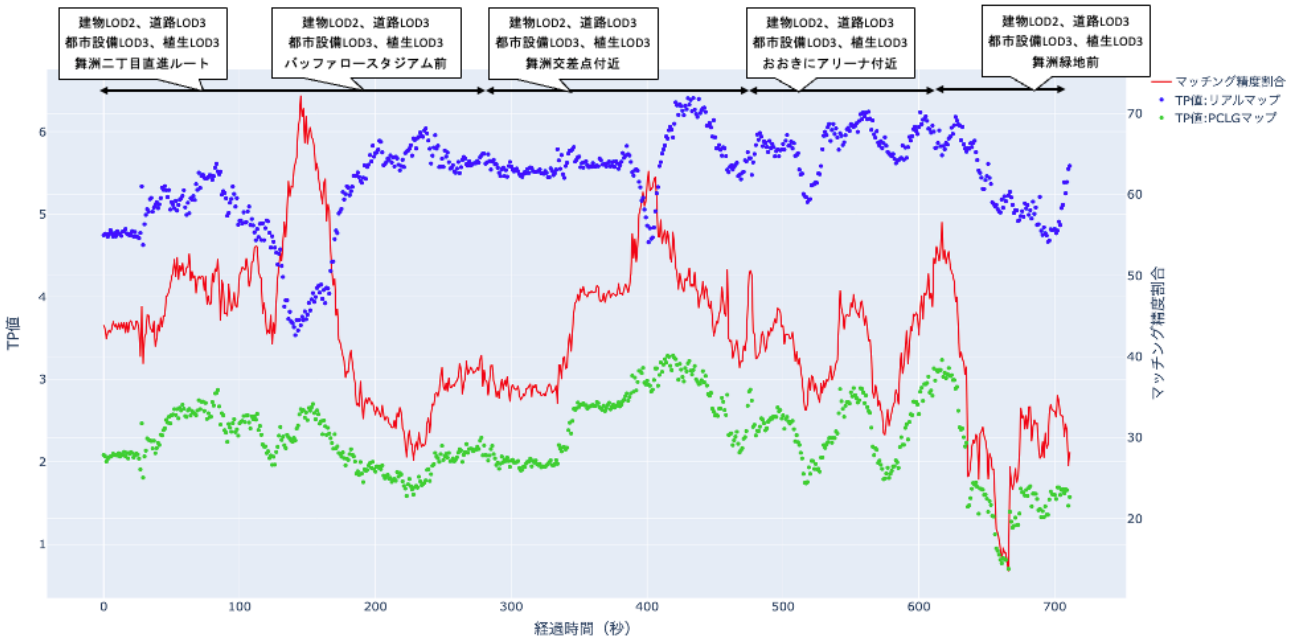


図 4-81 舞州エリア　マッチング精度割合の推移

下記の 2 つの図は、舞洲二丁目の直進ルートと PCLG 上での 3D 都市モデルの整備状況を比較したものである。舞洲二丁目直進ルートの前半では、建築物モデルがないものの、都市設備 LOD3、植生 LOD3、道路 LOD3 のモデルで点群マッチングの精度割合が 40%を超えた。
但し、舞洲二丁目直進ルートの後半、バッファロースタジアム前から舞洲交差点までは、精度割合が 30%から 40%で推移した。

図 4-82 舞洲二丁目直進ルート　現地の様子

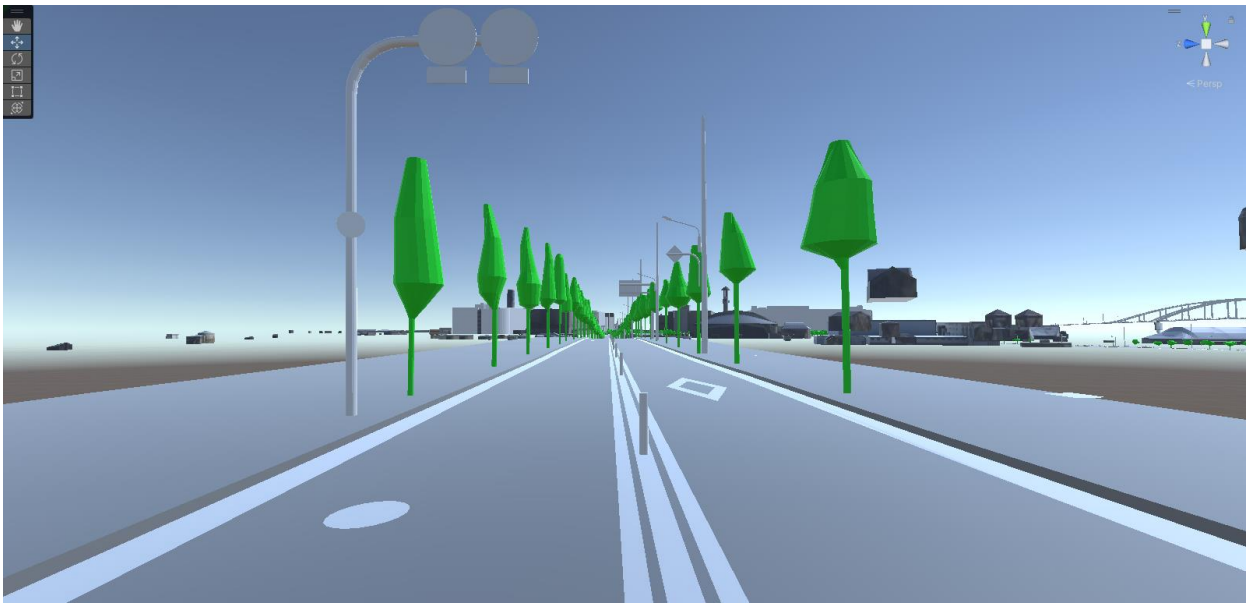


図 4-83　舞洲二丁目直進ルート　3D 都市モデル整備状況

下記の 2 つの図は、舞洲交差点付近と PCLG 上での 3D 都市モデルの整備状況を比較したものである。舞洲交差点付近の点群マッチング精度割合は 60%を超えた。精度増加の要因は、先ほどの舞洲二丁目直進ルートと比べると、道の両端にあるガードレール、ガードパイプの特徴点を捉えることができたためである。

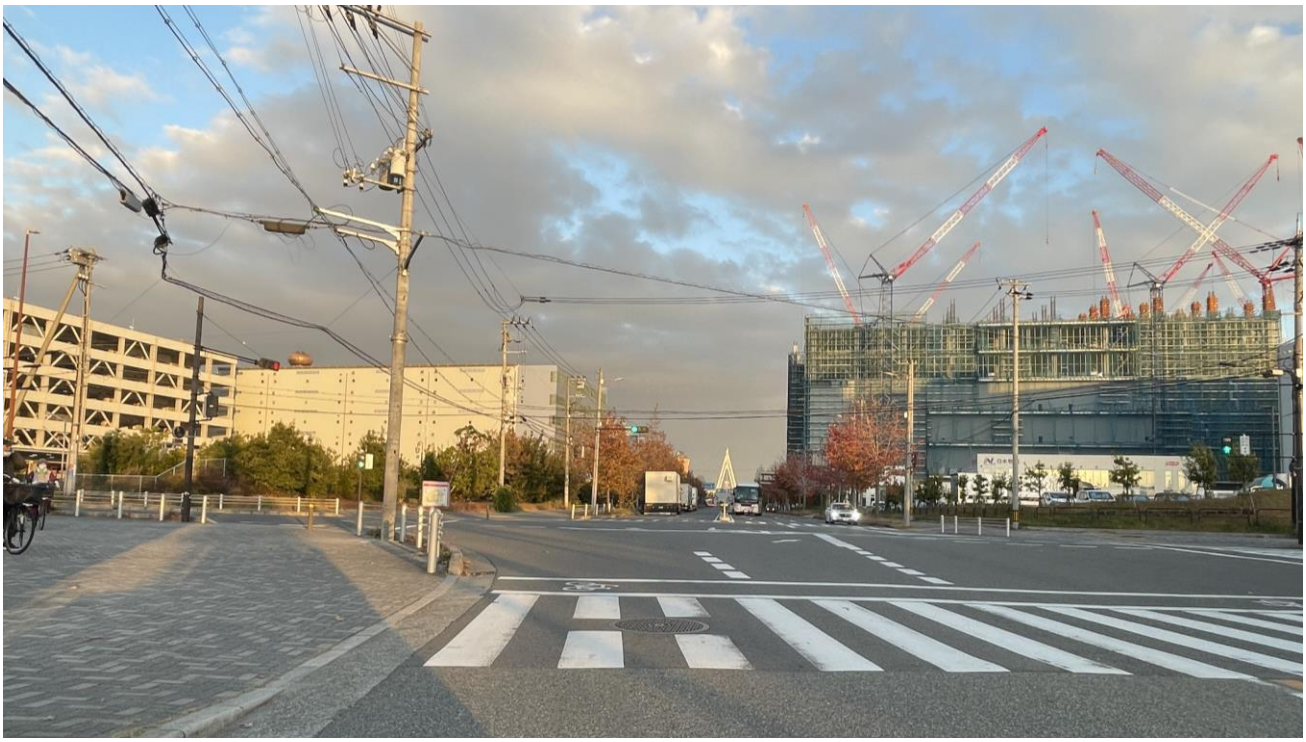


図 4-84 舞洲交差点付近　現地の様子

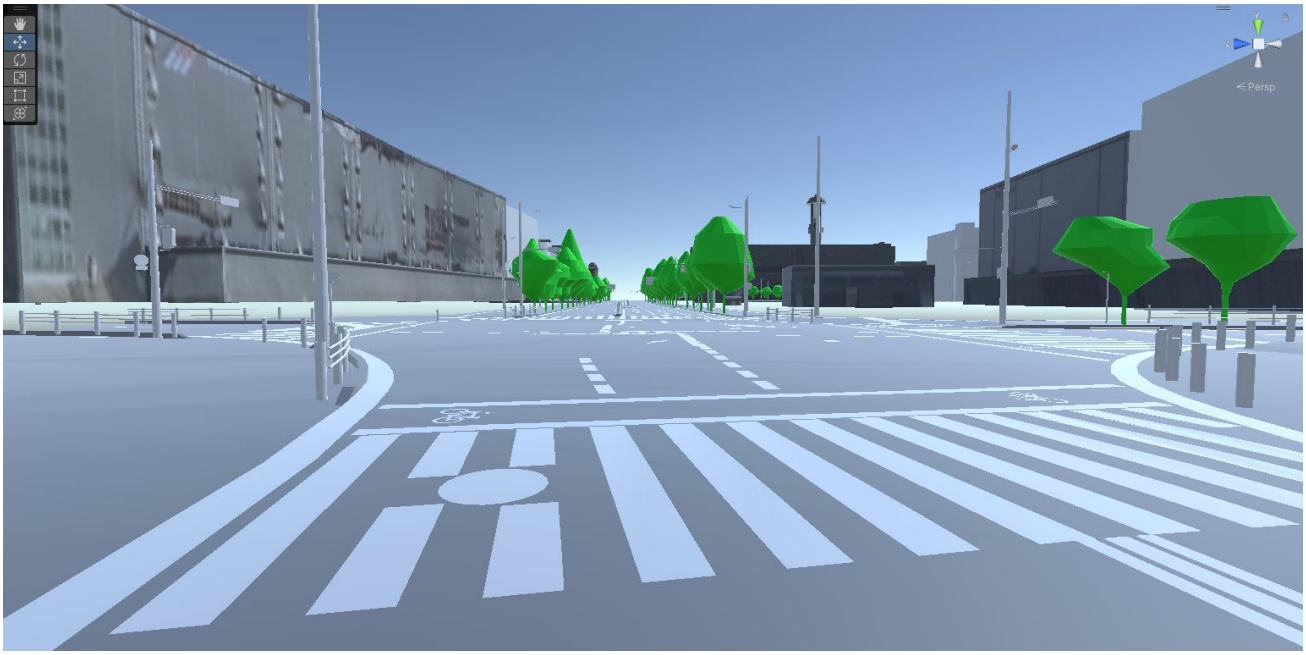


図 4-85　舞洲交差点付近　3D 都市モデル整備状況

下記の 2 つの図は、おおきにアリーナ付近と PCLG 上での 3D 都市モデルの整備状況を比較したものである。このルートの点群マッチング精度割合も 38%から 50%と、本検証において、高めで推移した。但し、このルートでは、路上駐車が多く、また下記の図でも見受けられるように左側の樹木が多くあり、その特徴点がつかみきれかった。そのため、先ほどの交差点付近より精度割合が減少した。

図 4-86 おおきにアリーナ付近　現地の様子

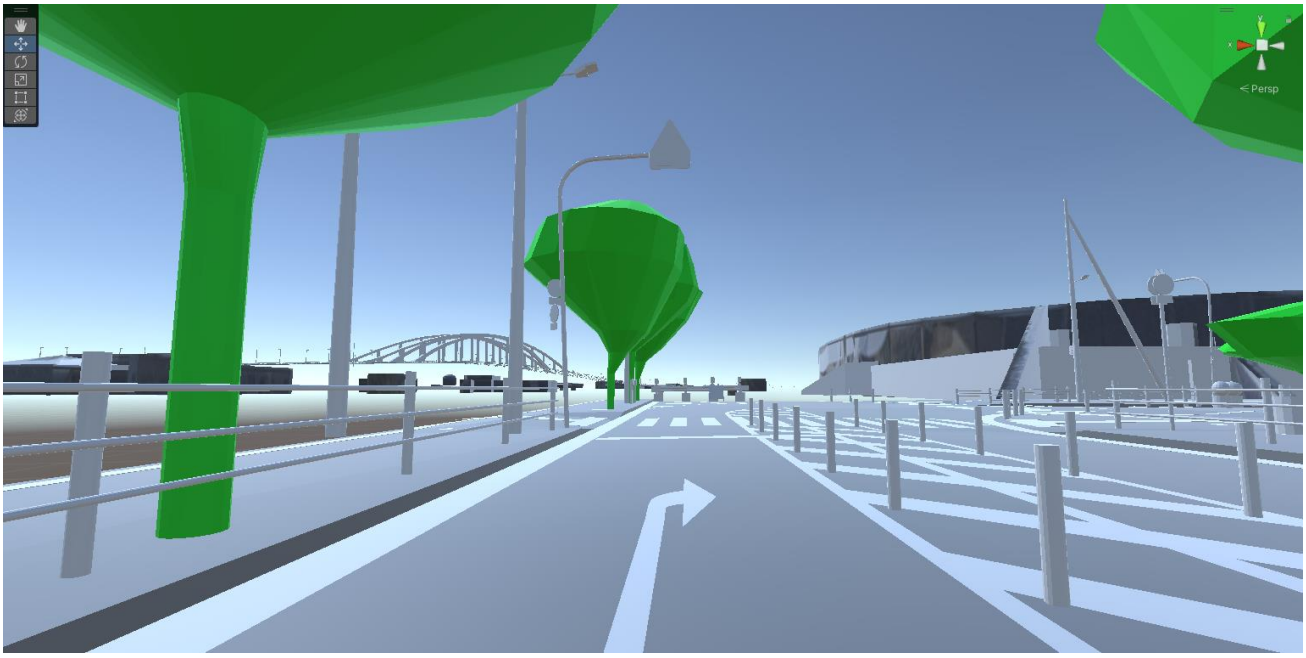


図 4-87 おおきにアリーナ付近　3D 都市モデル整備状況

下記の 2 つの図は、舞洲緑地付近と PCLG 上での 3D 都市モデルの整備状況を比較したものである。このルートでは 3D 都市モデルの地物が明確に少なく、またカーブのエリアで特徴点を外し、精度割合は 30%を下回った。

図 4-88 舞洲緑地付近　現地の様子

図 4-89 舞洲緑地付近　3D 都市モデル整備状況

2） 夢舞大橋　走行エリア（道路 LOD3、都市設備 LOD3 エリア）

夢舞大橋手前から、夢舞大橋を通過した走行軌跡を下記の図に示す。走行前半の夢舞大橋前は交通量が多く、車線も多いため、自己位置推定の精度割合は非常に低かった。夢舞大橋の途中から精度割合が増加した。

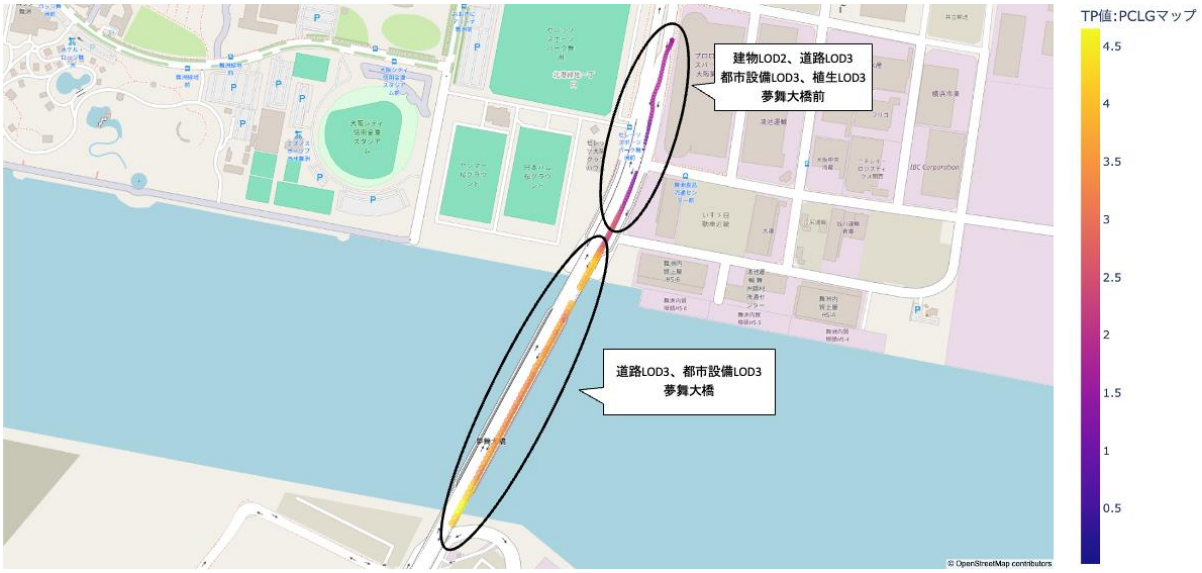


図 4-90 夢舞大橋　走行ルートと 3D 都市モデル整備状況

下記の図では、点群マッチングの精度割合を赤い線で示す。夢舞大橋前では精度割合が 50%から 0%まで減少した。しかし、橋の中央付近を境に、特徴点をつかみやすくなり、不安定ながらも 50%を超え、最終的には 70%を超える精度割合を示した。但し、全体的に車の交通量も多く、橋の反対側の特徴点がつかみづらい傾向があり、安定しない結果となった。

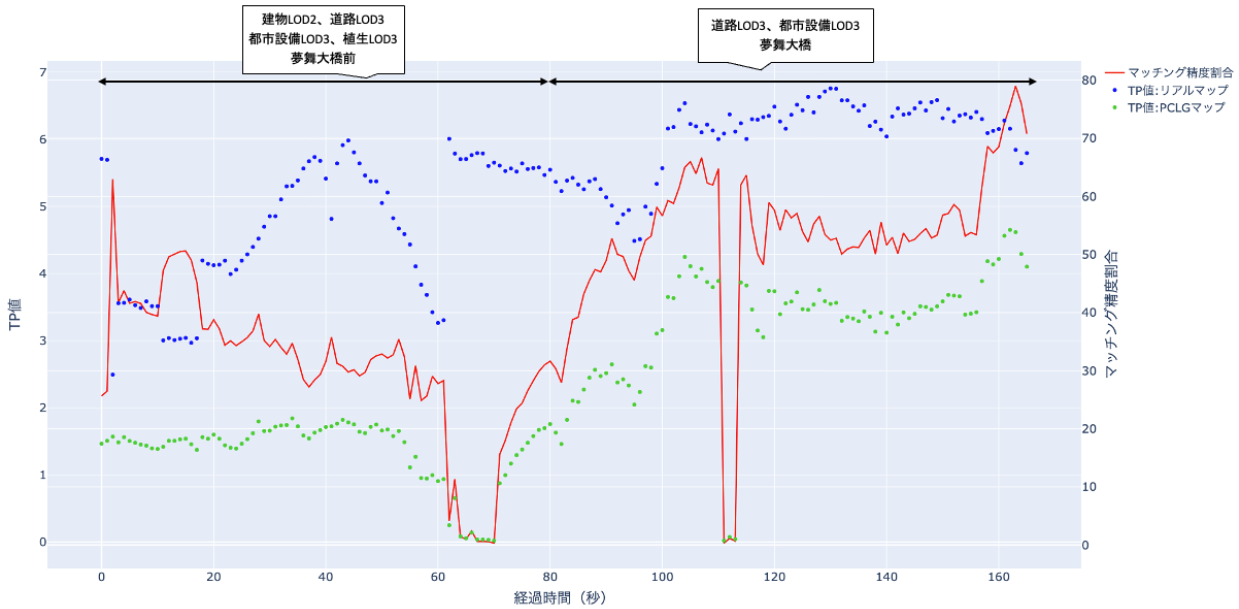


図 4-91 夢舞大橋　マッチング精度割合の推移

下記の 2 つの図は、夢舞大橋手前の現地の写真と PCLG 上での 3D 都市モデルの整備状況を比較したものである。片側 4 車線と広い道路では、道路の両側の特徴点を捉えられず、精度割合は 50%から 0%まで減少した。

図 4-92 夢舞大橋手前　現地の様子

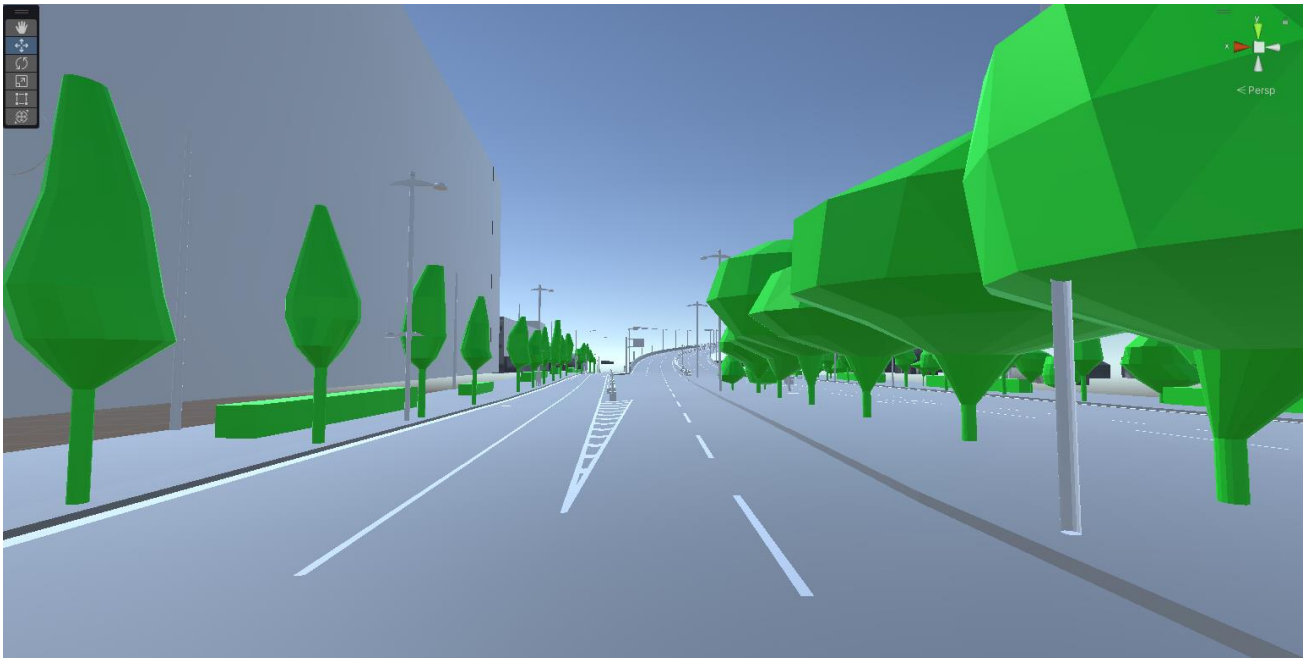


図 4-93 夢舞大橋手前　3D 都市モデルの整備状況

下記の図は、夢舞大橋に進入した際の現地の写真と PCLG 上での 3D 都市モデルの整備状況を比較したものである。3D 都市モデルが整備されたときには、片側 2 車線となり、一車線はガードレールが設置された状態だった。しかし、工事が進み、現地での検証時には片側 3 車線となっていた。3D と現実が、かい離した状態では自己位置推定の精度割合が 0%より上がらなかった。そこで、Unity 上で、ガードレールの情報を 3D で削除したところ、3D と現実の齟齬が少なくなり、現実点群マップの作成を行うことができた。

図 4-94 夢舞大橋　現地の様子

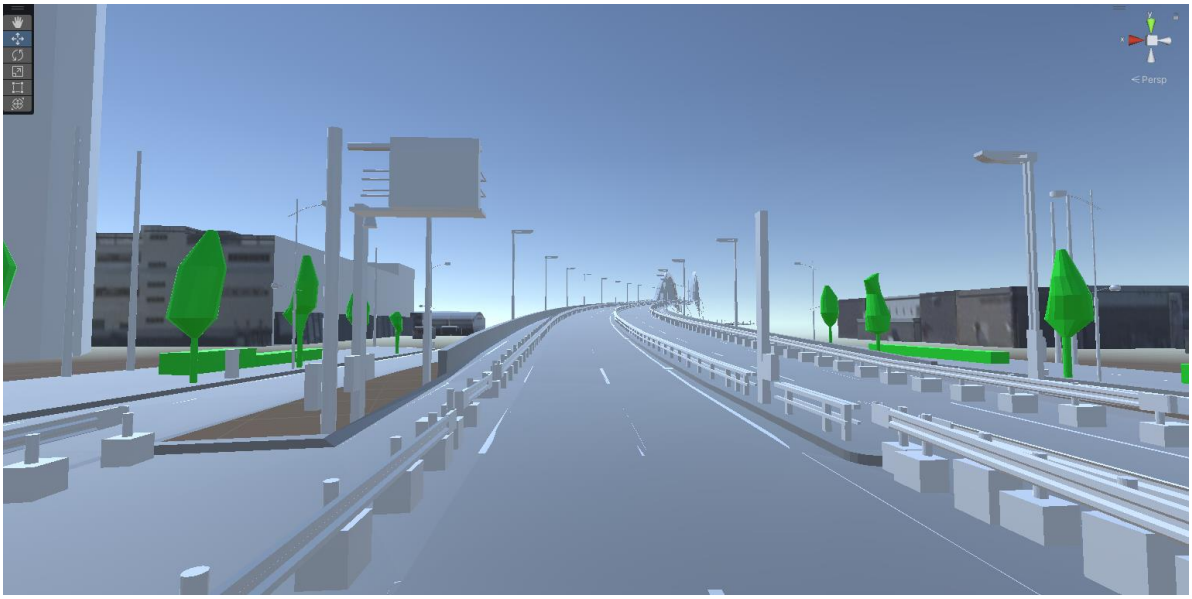


図 4-95 夢舞大橋　3D 都市モデル整備状況

下記の図に示される通り、3D 都市モデルの都市設備 LOD3 には左側の車線にガードレールの都市設備が配置されていた。この状態で検証を行うと、自己位置推定ができなかった（点群マッチングの精度割合が 0%）。Unity 上でガードレールを削除したことで、30%台の精度割合の値を得ることができた。

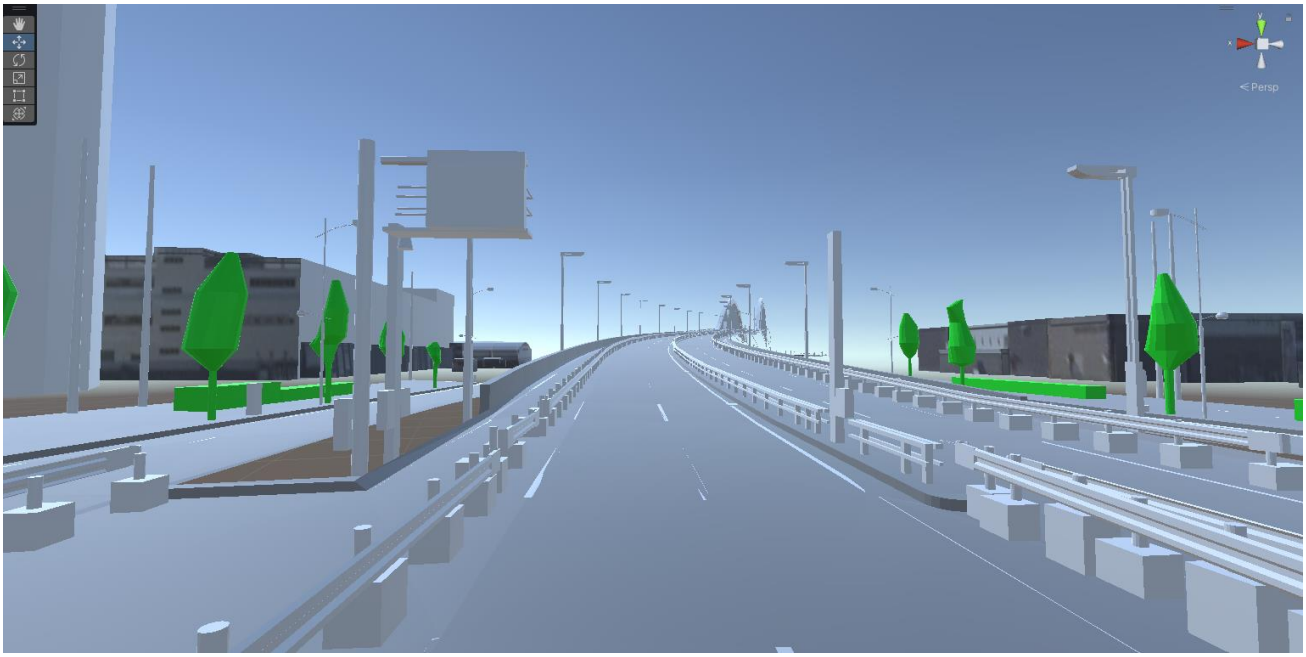


図 4-96 夢舞大橋　3D 都市モデル　ガードレール削除前

図 4-97 夢舞大橋　3D 都市モデル　ガードレール削除後

下記の 2 つの図は、夢舞大橋の中央付近の現地の様子と、PCLG 上での 3D 都市モデルの整備状況を比較したものである。ここでの精度割合は 50%から 80%と非常に高い値を示した。写真から見ても分かるとおり、現実の地物と 3D 都市モデルの表現が近しく、橋梁の特徴点を捉えることができたためである。

図 4-98 夢舞大橋中央付近　現地の様子

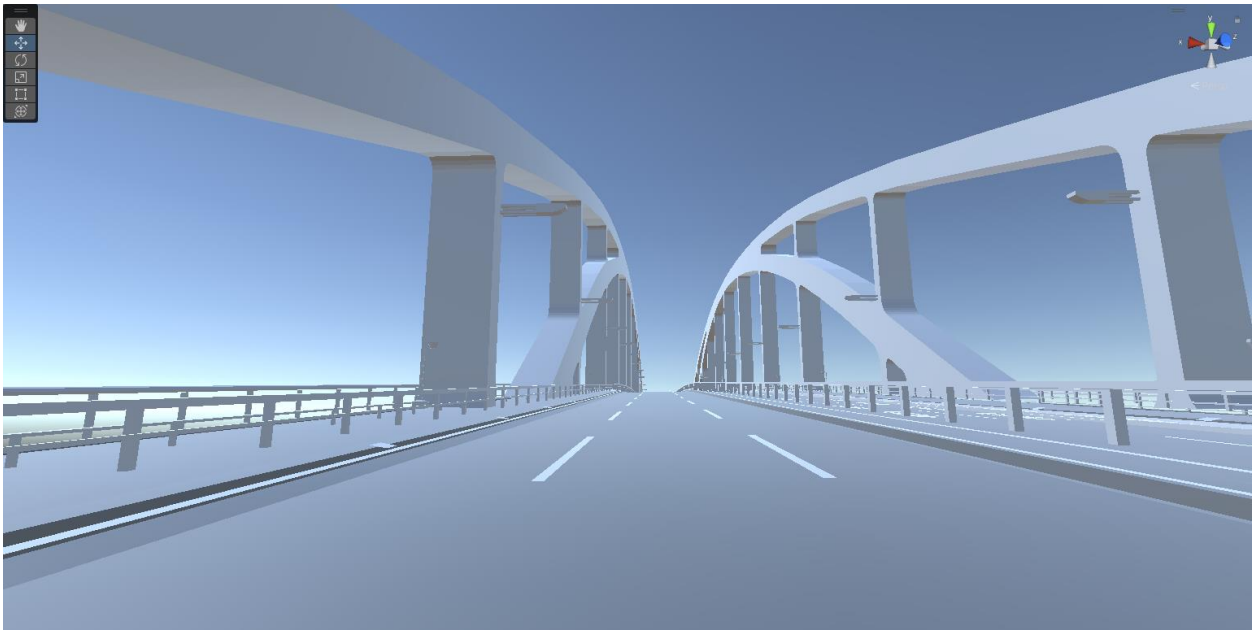


図 4-99 夢舞大橋中央付近　3D 都市モデル整備状況

表 4-4 舞洲エリア　検証結果

黄セル：KPI 達成	青セル：KPI 未達

検証内容	評価指標・KPI	目標値	結果		示唆
			項目	評価値	
建築物 LOD3、道路 LOD3、都市設備 LOD3、植生 LOD3	マッチング精度割合	50%以上	建築物少、直進コース	40～50%	● 建築物が少ないエリアでも、都市設備と植生モデルの特徴点から、40%以上の精度割合を得ることができたが、KPI としている 50%を安定して超えることができなかった ● 交差点のエリアでは、ガードレール、ガードポールの特徴点をつかみ、50%以上の精度割合を得ることができた ● 建築物が少なく、カーブするエリアでは特徴点を外したため、精度割合が減少した
			建築物少、交差点	50～60%	
			建築物少、カーブ	10～30%	
道路 LOD3、都市設備 LOD3	マッチング精度割合	30%以上	夢舞大橋手前	0～40%	● 夢舞大橋手前は片側 4 車線の道路だった。特徴点をつかめず、KPI としている 50%には至らなかった ● 橋の中央付近以降は地物が正確に表現されており、50%以上の精度割合を得ることができた
			夢舞大橋中央付近	50～80%	

3） 速度別評価

下記の二つの図は、ロッジ舞洲から舞洲緑地前までのルートを最大時速 15km と最大時速 30km で走行した際の点群マッチングの精度割合の推移を比較したものである。横軸は走行の経過時間である。最大時速 15km で走行した場合は 710 秒を要した。最大時速 30km で走行した場合は 250 秒の時間を要した。

点群マッチング割合の推移比較では、最大時速 15km と最大時速 30km で走行した際には大きな差異は見受けられず、双方とも同じようなグラフの推移を示した。時速 30km 以下では速度差による精度には差が見られなかった。

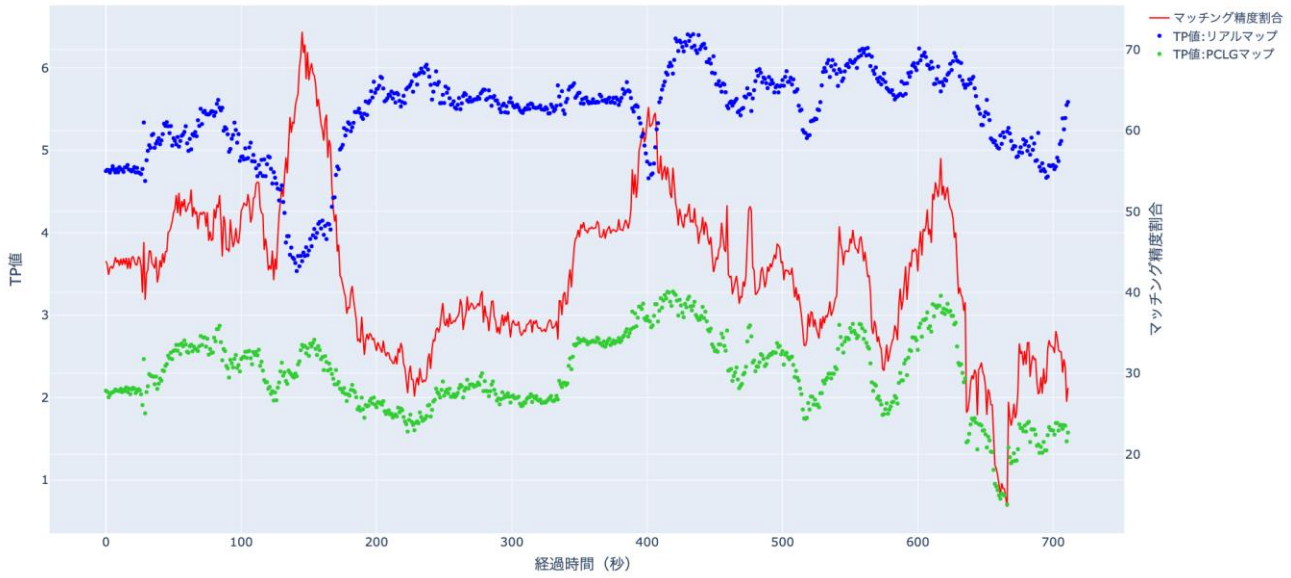


図 4-100　最大時速 15km で走行した際の推移

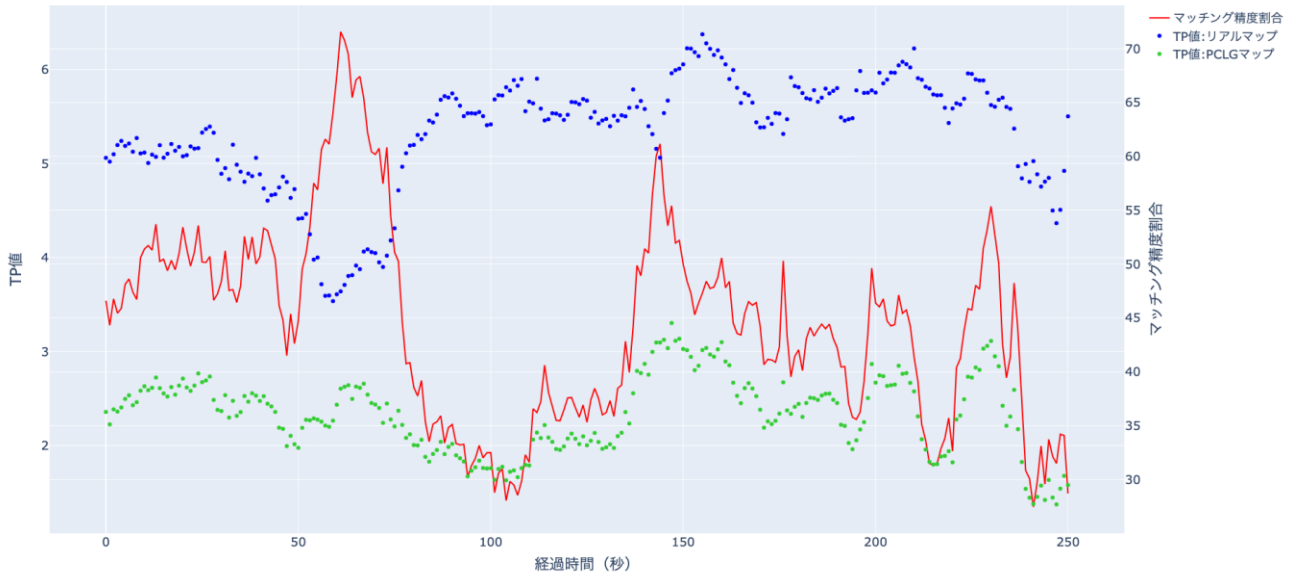


図 4-101　最大時速 30km で走行した際の推移

5. BtoB ビジネスでの有用性検証

5-1. 検証目的

実証仮説に基づき、以下の検証目的を設定する。

【検証仮説（再掲）】

- 今回の実証実験では、昨年度課題となった仮想点群生成の汎用性向上に向けて、3D 都市モデルから仮想 LiDAR を用いて点群マップを生成するシステムを開発する。仮想空間上の走行環境を現実世界に近づけるためのパラメータ設定や、取得した点群に対して最適なノイズフィルタ処理を実行することで、異なる都市特性や 3D 都市モデルの各 LOD 、現実世界を走行する車の速度にも対応した点群マップ生成システムを実現する。
- 本システムを用いた仮想点群生成プロセスについて、3D 都市モデルがあれば容易に点群マップの生成が可能な仕様にするため、汎用的なゲームエンジン上に仮想空間内でルートの生成や仮想 LiDAR を簡易的に利用しやすいインタフェースの開発を行なった。また LOD3 の整備エリアだけでなく、LOD2 の整備エリアでも、建物や道路の特徴次第では本システムを利用した自己位置測位が可能となる。

主に以下の点について、BtoB ビジネスに向けた有用性検証を行った。

- システムの有用性検証
  - PLATEAU PCL Generator に対するフィードバック
  - 実際に車両を走行させて取得した点群マップに対するフィードバック
  - PLATECU PCL Generator から出力された点群マップを用いた自己位置推定の精度割合に関するフィードバック

5-2. 検証方法

検証方法としては、被験者に対してデモンストレーションを取り入れたヒアリング・アンケートを実施した。（ヒアリング・アンケートの項目については「5-4.ヒアリング・アンケートの詳細」において記載）

事業者向けヒアリングの実施方法

- 会場：自社の会議室など
- 機材：体験・デモ用に以下のスペックの社用 PC を用意する。
  - PLATEAU PCL Generator 用 Windows PC
  - 走行データ再生用　Ubuntu PC
  - アンケート取得用 PC（Web ブラウザを利用）

5-3. 被験者

本実証実験では、運輸サービス事業者やゼネコン、道路事業者、自動運転ソフトウェア関連会社のうち、点群マップを用いたメンテナンスや自動運転を行う事業者をターゲットとしている。
本実証実験では、これらのユーザー該当する以下の方々にヒアリングを行い、本システムの価値を検証する。

表 5-1 被験者リスト

分類	具体名称	部署	役職	担当業務	人数
事業者/ユーザー	測量会社	-	-	環境地図作成	1 名
	道路管理会社	-	-	測量、図面作成	1 名

※ヒアリング対象先の希望により、会社名及びヒアリングの様子は非公開
※実証で用いたシステムの画面については、「4 章：4-2. 自己位置推定の精度割合検証　大阪府大阪市　舞洲地区」を参照

5-4. ヒアリング・アンケートの詳細

表 5-2 検証項目と評価方法

検証観点	No.	検証項目	定量評価	定性評価
PLATEAU PCL Generator に対するフィードバック	1	● PCLG のシステム全体の感想、出力に関するフィードバック	なし	アンケートの各設問に記入欄を設定
点群マップの精度割合に対するフィードバック	2	● 点群マップに対するフィードバック		
PCLG 点群マップの精度割合に対するフィードバック	3	● PCLG 点群を用いたみなとみらい、舞洲エリアの自己位置推定の精度割合に関するフィードバック		

5-5. 検証結果

5-5-1. ヒアリング結果

本実証実験では、想定事業者に PLATEAU PCL Generator や走行検証のフィードバックを得た。出力として PCD 形式で出力できると、自動運転以外にもメンテナンスやシミュレーションに利用可能というメリットや、Unity 上でオブジェクトを削除、追加できることが利便性につながるという知見を得た。
走行の検証に関しては、都市部のみなとみらいエリア及び、郊外の舞洲エリアの共通的な課題として、3D 都市モデルの更新頻度が挙げられた。特に舞洲地区は開発が現在も進んでいた。具体的には、夢洲大橋では現実では 3 車線だが、3D 都市モデルでは 2 車線となっていた。道路情報の更新をユーザー各自ができること、加えて、車や人の交通量情報を入力できるとより汎用性が広がるのではないか、という情報共有ができた。

- 1) PLATEAU PCL Generator に対するフィードバック

  - 仮想空間の中で車両を走行させて、点群マップを得ることができる。それが PCD 形式で出力される点は使いやすい。
  - 地物を Unity 上で削除でき、オブジェクトを追加できることは強みとなる。
  - 建築物の壁の厚みが点群マップで綺麗に出力できている。事前取得の点群マップだと、壁の厚みが問題になることが多い。
  - より現実世界との差異をなくすために、車や人の交通量、移動量をシナリオで入力できたらいい。
  - PCD 形式の点群マップは、Autoware 以外でのソフトウェアでも用いられている。そのため、Autoware と密に連携するよりも、PCD 形式の点群マップを出力することに重きを置いた方が市場性は高まると思われる。

- 2）点群マップに対するフィードバック

  - 点群マップを用いた自己位置推定の精度割合は高いことは評価できる。一方、出発地点から一周して戻ってくるとマップがずれる（ループクロージング処理）や、壁の厚みが正確に出ないなどの問題が挙げられる。
  - 3 車線以上ある大通りでは、建築物の特徴点を取りやすいように、基本的に左側を走行した方が良い。
  - 3 車線の真ん中を走行していると、周囲の車に遮られて点群マッチングの精度割合が落ちてしまう。
  - LiDAR を用いた点群マッチングの自己位置推定は、時速 40km が限界だと思われる。

- 3) PCLG 点群マップの精度割合に対するフィードバック

  - みなとみらい地区では路地だと精度割合が増加し、大通りだと精度割合が減少するなど、自己位置

推定の精度割合が道幅に依存する、という興味深いデータが得られた。

- 都市設備 LOD3 があるルートで精度割合が増加する要因として、建築物の外壁と都市設備によって、進行方向の自己位置推定が取得しやすいため、精度割合が増加につながっていると考えられる。
- 点群マッチングによる自動運転では、いかに点群マップを更新頻度が上げられるかどうかが課題になると思われる。

6. 成果と課題

6-1. 本実証で得られた成果

6-1-1. 3D 都市モデルの技術面での優位性

本実証を通じて、以下に示す 3D 都市モデルの技術面での優位性が明らかとなった。

表 6-1 3D 都市モデルの技術面での優位性

大項目	小項目	3D 都市モデルの技術面での優位性
システム・機能	PLATEAU PCL Generator	● 本検証で、建築物 LOD1 のエリアでも点群マップの有用性は確認できた。3D 都市モデルの整備範囲で利用可能というカバレッジの優位性がある ● 都市部では都市設備が、郊外地域では都市設備と植生のモデルが必要となる。地域に応じて Unity 上でオブジェクトの追加・削除をすることで、ユーザー自身でカスタマイズが可能になる

6-1-2. 3D 都市モデルのビジネス面での優位性

本実証を通じて、以下に示す 3D 都市モデルのビジネス面での優位性が明らかとなった。

表 6-2 3D 都市モデルのビジネス面での優位性

大項目	小項目	3D 都市モデルの技術面での優位性
工数	点群マップ生成	● 通常、点群マップを作成するのに、10 日ほどの工数が掛かる。しかし、3D 都市モデルを利用することで、工数を著しく短縮し、半日ほどで該当エリアの点群マップを PCD 形式で出力できる
コスト	点群マップ生成	● 本来、点群マップを用意するのに必要なコストは 100 万円以上かかる。しかし、3D 都市モデルというオープンデータを活用することで、コストを大幅に抑えた検証が可能になる

6-2. 実証実験で得られた課題と解決策

表 6-3 実証実験で得られた課題と解決策

大項目	小項目	実証実験で得られた課題	課題に対する対応策
システム（機能）	みなとみらい地区 LOD1 エリア	● 片側 1 車線の道路であれば建築物 LOD1、道路 LOD1 のモデルだけでも自己位置測位は可能であったが、自律走行のマップとして用いるには精度が低かった	● 建物 LOD1、道路 LOD1 の地域でも、都市設備のモデルがあると精度の向上が見込める。但し都市設備 LOD1 だと外形に囲まれた面となり、特徴点が少なく精度が下がる。そのため都市設備 LOD2 の簡略化した立体が必要になる
	みなとみらい地区 LOD3 エリア	● 道路 LOD3、都市設備 LOD3 が整備されているエリアの自己位置測位の精度は高く、自律走行のマップの精度を満たした ● LOD3 から LOD1 となるエリアでは、道路の高さが変わり、都市設備がなくなった。その地点を境に自己位置測位の精度が下がり、自己位置をロストした。上記の課題は道路 LOD モデルに高さを付与しても結果は変わらなかった。該当ルートは片側 3 車線の大通りだった。交通量も多いため点群が片側の建物の外壁にしか届かないケースが増加し、マッチング精度が減少した	● 道路モデルに関しては高さを付与したモデルを利用することで道路の高さ問題を解消することができた。但し検証エリアは海抜が低かったため、海抜が高い地域や起伏がある地域など、より多くのサンプルが必要となる ● 都市設備に関しては前項でも述べた通り、自己位置測位の精度を向上させるためには必須の要素である ● 上記の道路、都市設備の課題を解決できても、大通りでは点群が届かず、点群マッチング自体が有用でない例外のケースが存在する。そのためにも RTK 測位やカメラを使った別の自己位置測位の技術との組み合わせが必要になる
	舞洲地区 ロッジ舞洲 LOD3	● 建物モデルが少なくとも、道路 LOD3、都市設備 LOD3 のモデルで自己位置測位は可能だった。但しカーブや路上駐車などで精度が下がる	● カーブのエリアでは道路と都市設備のモデルだけでは不十分であるため、建物がある場合は建物モデルで補うか、他のセンサを利用した補正が必要となる ● 路上駐車があるルートでの走行は例外のケースとして扱い、車線変更の機能や他のセンサとの補正が必要となる
	舞洲地区	● 夢舞大橋での橋の入り口では点群マ	● 地物が少ないエリアでは前段で述べ

	夢舞大橋 LOD3	ッチングの対象となる地物が少なく、精度向上は難しかった。橋の中腹から橋梁が捉えられると、精度は大幅に向上した	た通り点群マッチングに加え、他の自己位置測位の技術との組み合わせが必要となる
	夢舞大橋 現実とのかい離	● 舞洲エリアの夢舞大橋を走行した際に、現実の地物と 3D 都市モデルの整備状況がかい離異したエリアがあり、Unity 上でモデルの修正が必要だった ● オブジェクトの削除のみの対応は問題ないが、オブジェクトを追加する場合は開発の工数が発生してしまう	● Unity 上のモデルを容易に更新する策として、追加されるようなオブジェクトの種類をあらかじめ用意し、簡単に配置できるよう改善する
	精度割合増加のための、都市設備の必要性	● 本検証で都市設備の有用性が明確になったが、3D 都市モデルのエリアの多くで都市設備が整備されていない	● 上記の課題にも記載したとおり、ある程度の都市設備を手動で追加できる仕組みが必要である
システム（UI・UX）	走行車両シミュレーション	● 都市部の場合、大通りでの走行車両の多さが精度に影響する	● PLATEAU PCL Generator に他の走行車両もシミュレートした上で点群マップを生成し、事前に精度が低く出力される箇所を予想できるようにする
サービス	シミュレーターとしての活用	● PCLG は点群マップを出力するだけの機能に留まり、走行自体は現地で行う必要がある	● PLATEAU PCL Generator を点群マップ出力に加え、Autoware と連携させることで、シミュレーターとして利用できるようにする
価格	ビジネスモデル	● 本検証で開発した PLATEAU PCL Generator はオープンソースで公開する。追加機能で収益を見込む	● オープンソースで公開する根幹の機能は無償で、サポートや、カスタマイズ機能は有償サポートといった付加価値でのビジネスモデルの構築が必要となる
販路	マーケティング	● 本検証で開発したシステムは、Unity が使えるエンジニアをターゲットに開発した。GitHub 上での集客が主である。	● YouTube やプレスリリースなどで本システムをアピールし、オープンソースとしての知名度を上げることで、販路の拡大が期待できる
販促	ターゲティング	● 本検証で開発したシステムは、オープンソースで公開されているが、エンジニア向けの情報しか公開されて	● 販路の課題にも挙げた通り、オープンソースとしての知名度を上げ、実績を積み上げることでより販促効果

		いない	にもつなげられる

6-3. 今後の展望

都市部における大規模な再開発が活性化し、自律走行が可能な無人搬送車両活用への期待が高まる一方、GPS 測位、LiDAR 共に精度面や効率面で汎用的なサービスとしては未だ課題が累積している。これらの課題解決策として、今回の実証では昨年度開発した車両用の自律走行システムをベースに、汎用性と可用性の向上に向けた点群生成プロセスの精緻化や仮想空間内での点群マップ取得プロセスの汎用化に向けた改修を行った。実用化に向けて、新たに抽出した課題を迅速に解決し、より高精度な運搬サービスを生み出すと共に、本システムの更なる高機能化を志向する。

今年度新規に開発した PLATEAU PCL Generator は 、3D 都市モデルが整備されている地域であれば、仮想空間内で自律運航に必要条件となる高精度な点群マップを作成することを可能にした。本システムによって昨年度課題となった 15km/h を超える車両速度でのマッチング精度については良好な結果が得られ、また、今年度は都市部のような建築物の密集するエリアでの検証も行い、汎用性の高さを確認できた。
課題としては、抽出された 3D 都市モデルと現実空間のかい離の影響、建築物数が少ないエリアでより速度を上げた場合（40 ㎞/h）の精度割合増加の要因、仮想空間内で取り込んだ 3D 都市モデルに対する事前処理や自己位置推定の計算時間短縮などが挙げられた。これらの課題に対処するために、引き続き検証を行い、今後も改善を図る。近い将来、3D 都市モデルのみから生成した点群マップを活用し、一層の進化を遂げた高精度かつ低コストな自律走行システムとしての社会実装を目指す。

7. 用語集

A)　アルファベット順

表 7-1 用語集（アルファベット順）

No.	用語	説明
1	3DLiDAR	● LiDAR は Light Detection And Ranging の略 ● レーザーを使用して周囲の環境を三次元でスキャンし、物体の位置や形状を正確に把握するセンサシステム
2	3D 都市モデル	都市の建築物、道路、地形などを三次元で表現したデジタルモデル
3	AGV	● Automated guided vehicle の略 ● 産業用途で多く使用される自動運転車で、人間が運転操作を行わなくとも自動で走行できる搬送車
4	AV1 形式	● 高効率なビデオ圧縮。特にインターネットストリーミングや高解像度メディアのために設計されている。
5	BIM	● Building Information Modeling の略 ● 建築物の物理的および機能的特性をデジタルモデル化するプロセスで、設計、建設、運用管理を効率化する。
6	CloudCompare	● ３D ポイントクラウドと３D メッシュデータを処理、比較、分析するためのオープンソースのソフトウェア
7	CSV	● Comma Separated Values の略 ● データをコンマで区切って格納する、シンプルで広く使用されるテキストベースのフォーマット
8	GNSS	● Global Navigation Satellite System の略 ● 地球を周回する衛星からの信号を使用して、地上の受信機の正確な位置情報を提供する全球的な衛星測位システム
9	IMU	● Inertial Measurement Unit の略 ● 慣性計測装置のことで、３次元の慣性運動（直行３軸方向の並進運動及び回転運動）を検出する
10	NDT スキャンマッチング	● Normal Distributions Transform の略
11	PCD 形式	● Point Cloud Data の略
12	PCLG マップ	● PLATEAU PCL Generator を使用して作成されたマップ
13	ROS	● Robot Operating System の略 ● ロボット開発において、重要な役割を果たすオープンソースのロボット制御ソフトウェアを意味する
14	RTK	● Real Time Kinematic の略

		● 固定局と移動局の 2 つの受信機を利用し、リアルタイムに 2 点間で情報をやりとりすることで、高精度での測位を可能にする手法
15	SDK	● Software Development Kit の略 ● 特定のソフトウェアパッケージ、ソフトウェアフレームワーク、ハードウェアプラットフォーム、コンピュータシステム、オペレーティングシステムなどの開発をサポートするためのツールとドキュメントの集合
16	Warpner	● WebRTC（音声やビデオ、データなどを Web ブラウザ間でリアルタイムにやりとりするための通信規格）を利用した映像伝送システム
17	Waypoint	● 経路上の地点情報 ● 自立走行を行う際に、現在地から次のポイント（座標）を設定したポイント

B)　五十音順

表 7-2 用語集（五十音順）

No.	用語	説明
1	ガウシアンノイズフィルタ	● 画像や信号からランダムな変動（ガウシアンノイズ）を平滑化するために、ガウス関数を基にした重み付け平均を用いる処理技術
2	ガウス分布	● 正規分布とも呼ばれ、自然現象や社会科学データなどに広く見られる、平均値を中心に左右対称なベル型の確率分布
3	自己位置測位	● ある場所にいる自分自身の位置を推定する技術 ● GPS を使う手法やカメラ映像を使う手法、LiDAR の超音波センサを使用する手法などが存在する
4	事前取得マップ	● 検証ルートを事前に走行し、LiDAR により取得した点群マップ
5	自律運行	● 人間の直接的な操作を必要とせず、センサ、AI、アルゴリズムなどを用いて自己判断で移動やタスクを行うシステムや車両の機能
6	デジタルツインビュワー	● 車両の位置や速度をリアルタイムで遠隔から監視できる、3D都市モデルを利用したビュワー
7	ボクセル	● Volume と Pixel の合成語で、三次元空間におけるデータの最小単位を表す立方体の要素
8	ラストワンマイル	● 現在は、物流・交通業界において多く用いられ、「顧客にモノ・サービスが到達する最後の 1 マイル（1.6km 程度）の区間・接点」を指す ● 元々は通信業界に用いられていた用語で、「生活者や企業に対し、通信接続を提供する最後の区間・接点」を意味していた

以上

3D 都市モデルと BIM を活用したモビリティ自律運航システム（車両）v2.0 技術検証レポート

2024 年 3 月　発行
委託者：国土交通省　都市局
受託者：株式会社竹中工務店/アダワープジャパン株式会社