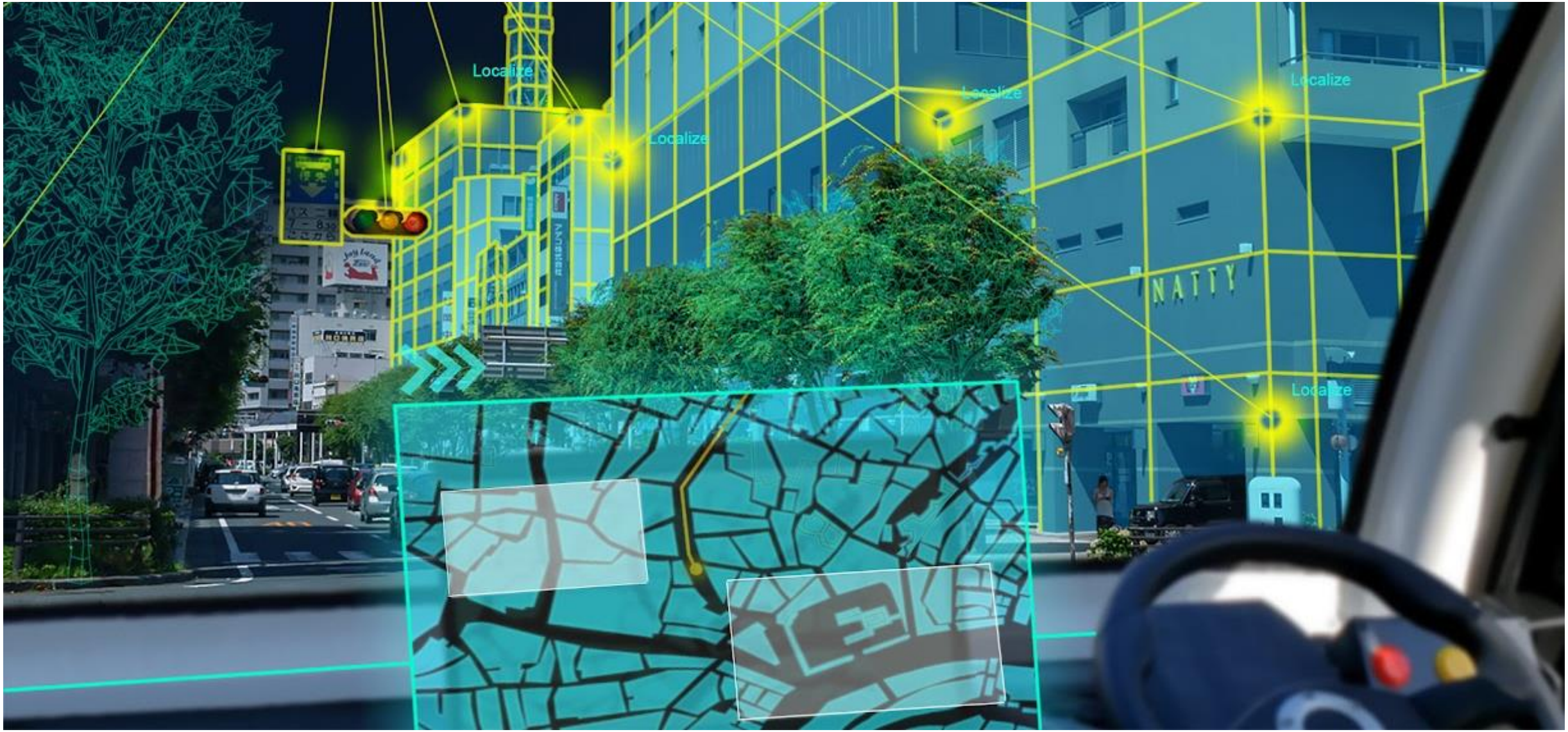
自動運転車両の自己位置推定におけるVPS(Visual Positioning System)活用Ver2 技術検証レポート

Technical Report for VPS Utilization in Localization of Autonomous Vehicles



目次

Ⅰ．実証概要

Ⅱ．実証技術の概要

Ⅲ．実証システム

Ⅳ．実証技術の検証

Ⅴ．成果と課題

全体概要（1/2）

ユースケース名	自動運転車両の自己位置推定におけるVPS(Visual Positioning System)活用Ver2
実施場所	静岡県　沼津市
目標・課題 ・創出価値	現在の自動運転システムにおける自己位置推定には、GNSS、LiDAR、速度計、ジャイロスコープ等の各種センサや、3Dベクトルデータ、3D点群データ等の様々なデータが活用されており、一定の実用水準にあるものの、使用されるLiDAR等の機器は高額なものが多く、また、必要となる高精度なデジタルマップの作成負荷も高いため、様々な用途に自動運転システムを導入しやすい状況には至っていない。安価、簡易、安全、スケーラブルな自動運転システムの普及のため、3D都市モデルと光学カメラ画像を組み合わせたVPS技術の確立が必要である。
ユースケース の概要	今回の実証では、2021年度に実施した自動運転車両の自己位置推定におけるVPS（Visual Positioning System）の活用に係るユースケースの検証結果や明らかとなった課題に基づき、3D都市モデルと、産業技術総合研究所から提供されているVPS「C*」(C-STAR)を活用した自己位置推定システムの実用化に向けた開発を実施する。カメラ画像から取得した情報と、3D都市モデルから生成されるデータの特徴点とを照らし合わせることにより、車両の自己位置を推定するシステムを開発・検証し、自動運転システムへの活用を見据えたフィージビリティスタディを行う。

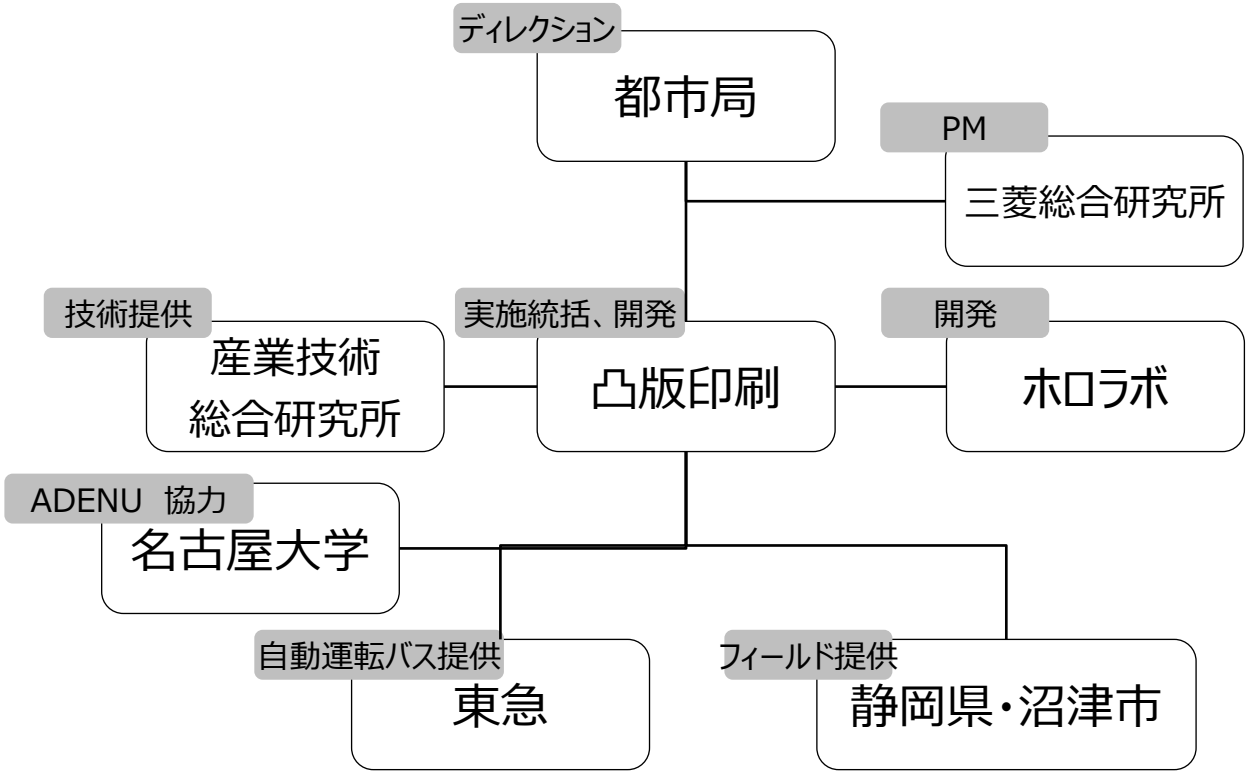
全体概要（2/2）

実証仮説	3D都市モデルを用い、廉価な光学カメラの画像から自己位置推定を可能とすることができれば、低コストで都市内交通などの様々なモビリティに自動運転機能を提供できる。一連の実証ではこれを目指した技術実証を行う。 2021年度に実施した技術実証では、3D都市モデルを活用したVPSによる自己位置推定の検証を行い、位置測位のアルゴリズムや3D都市モデルで用いられるテクスチャ情報のあり方等についての課題を抽出した。 今回の実証では、2021年度に抽出された課題を解決するため、テクスチャ情報だけに依存せず、建物等の地物・形状から生成される特徴も活かすことができる手法を用いて、3D都市モデルに最適化されたVPSの構築を目指す。
検証ポイント	利用するVPS技術としては、新たに産業技術総合研究所から提供されている「C*」を用いる。「C*」は、3Dデータをレンダリングした映像と実際のカメラ映像とをNID（Normal Information Distance）と呼ばれる画像同士の類似度を指標に照合して自己位置を推定する仕組みとなっており、必ずしも高精細なテクスチャを必要とせず、テクスチャ品質が高くない3D都市モデルでの最適化を目指すことが可能となる。具体的には、3D都市モデル（LOD3）を実際の光源環境に近い状態で高品質にレンダリングし、その映像と実際の車両に設置したカメラから取得した映像とを照合することで位置情報を推定する。本実証では、このC*を中心としたシステムを構築し、自動運転車に搭載したカメラ映像からの自己位置推定を行い、RTK-GNSSで取得した座標との差により、その位置精度を検証する。 また、今後の社会実装に向けて、VPSで自己位置推定された位置情報と自動運転統合ソフトウェア（ADENU）による位置情報を参照できるようにし、実際の自動運転に用いることができるかを検証する。

実施体制

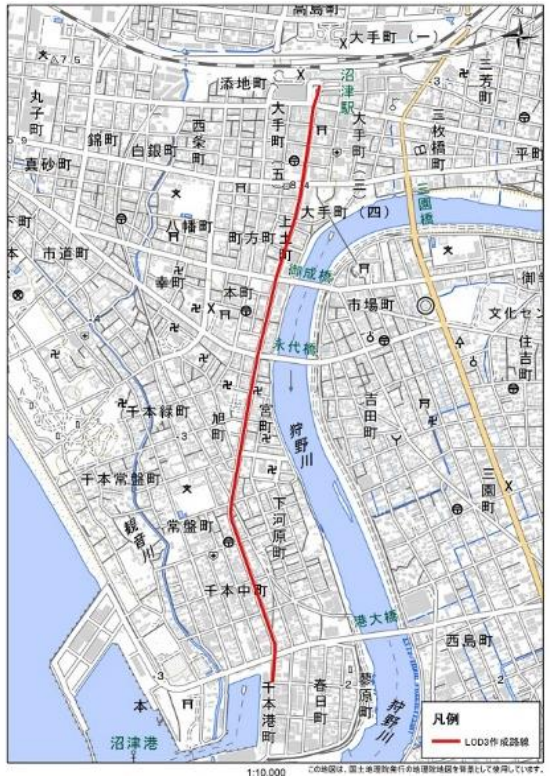
表　各主体の役割

主体	役割
凸版印刷株式会社	実施統括 アプリケーション開発
株式会社ホロラボ	システムインテグレート アプリケーション開発
国立研究開発法人 産業技術総合研究所	VPS技術提供
東急株式会社	自動運転バス車両の提供・運行
国立大学法人 東海国立大学機構 名古屋大学	自動運転ソフトウェア「ADENU」 連携協力
静岡県・沼津市	自動運転走行フィールド提供
三菱総合研究所	PM



実証エリア

静岡県　沼津市　沼津駅前から沼津港までの約2kmのルート



スケジュール

実施事項	令和4年										令和5年		
	3月	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. 企画検討・実証計画策定		■	■										
2. データ取得・整備・更新		■	■										
3. システム設計・開発			■	■	■	■	■	■	■	■			
3-1. 要件定義			◀	■	■	■	▶						
3-2. システム開発				◀	■	■	■	■	■	▶			
3-3. 現地検証					●		●		●				
4. ユースケース開発の実証											■		
5. 事業成果のとりまとめ												■	■

Ⅰ．実証概要

Ⅱ．実証技術の概要

Ⅲ．実証システム

Ⅳ．実証技術の検証

Ⅴ．成果と課題

活用技術｜一覧

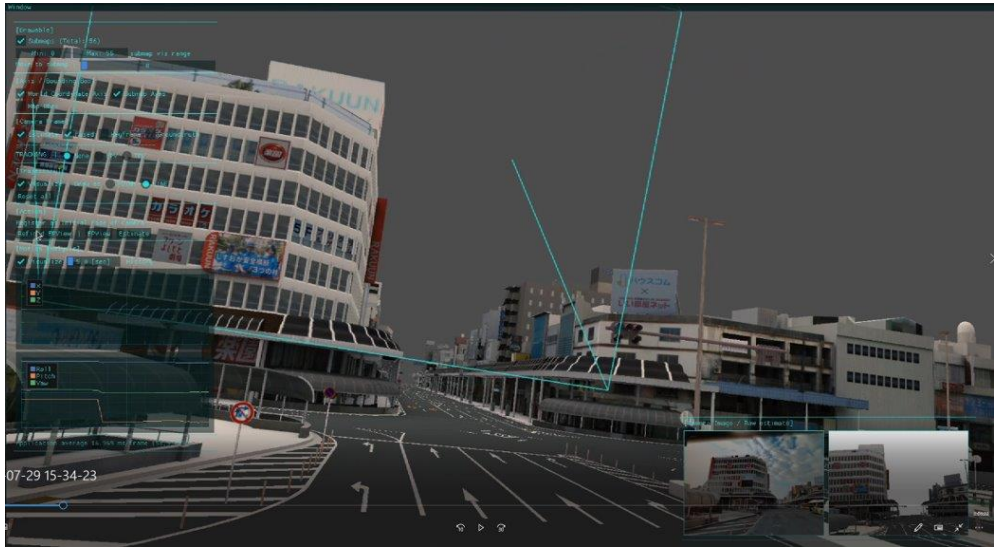
活用技術	内容
C*	産業技術総合研究所で研究・開発された単眼カメラによるVPSソフトウェア
ADENU	名古屋大学COIで研究・開発された自動運転車用のソフトウェアパッケージ

Ⅱ. 実証技術の概要 > 2. C*
C*の概要

概要

項目	内容
名称	C*(Cross-modal Simultaneous Tracking And Rendering、CSTAR)
概要	• 産総研が開発した画像同士の類似度を位置推定に利用する自己位置推定手法
主な機能	• 入力画像と3Dレンダリング画像を照合し推定位置を出力する • 既存手法のような特徴量3Dマップを必要としない • 写実的な3Dモデルを利用しての位置推定が可能 • 実時間処理が可能 • 廉価な単眼カメラで、センサ特性等によらない頑健な位置推定が可能
利用する機能	• 単眼カメラによる自己位置推定機能

C*動作画面



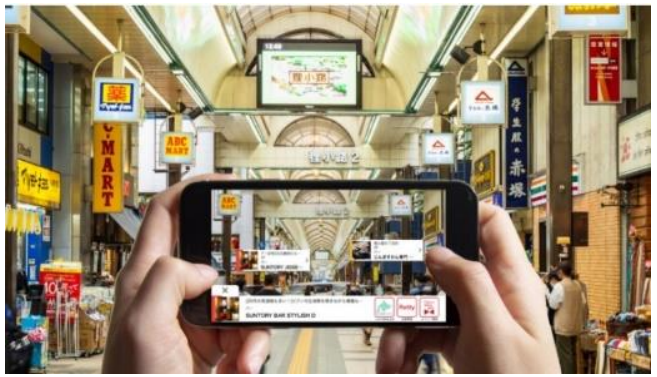
VPSの重要性

- まちづくりのデジタルトランスフォーメーションにおいては、我々が存在するフィジカル空間と仮想空間との間で、情報・データの収集・提供・フィードバックを行うことにより、新たな価値を生み出していくことが期待される。フィジカル空間をサイバー空間に再現し、相互に緊密に融合させる概念がデジタルツインであり、近年この基盤となるオープンデータとして、3D都市モデルが整備・公開されてきている。
- デジタルツインにおける情報取得や活用の際に重要となる要素のひとつが位置情報であり、さまざまな情報は位置情報に紐づけられることで利用価値が増すと考えられる。
- これまで測位情報の取得については、GNSS (Global Navigation Satellite System) を活用したサービス等が広く活用されている。一方、視覚情報を主として活用する技術がVPS (Visual Positioning System/Service)である。VPSは、画像に基づく測位技術であり、汎用的なRGBカメラがあれば稼働するため、スマートフォン等のさまざまなデバイスで利用可能であり、屋内外でも使用可能という特徴を持つ。

位置情報測位手法の分類

測位情報	使用装置	対象場所のインフラと事前情報	対応場所	姿勢推定
電波	衛星／受信機	必要に応じて地上基準局	野外	不可
	Wi-Fi/Bluetooth	AP式説が必要。地図生成が必要	室内	不可
赤外線	LiDAR	不要。地図生成が必要	屋内／屋外	可
画像	RGBカメラ	不要。地図生成が必要	屋内／屋外	可

VPSの活用事例（AR観光ガイド）



出所）PLATEAUウェブサイト（2023年3月4日閲覧）
https://www.mlit.go.jp/plateau/use-case/uc20-025/

VPSと自動運転

自動運転システムにおける自己位置推定については、これまで、SLAM（Simultaneous Localization and Mapping）やHDマップ（高精度3次元地図データ）等の技術・データを用いたシステムの開発・実証・実用化が進んでいる。これらの技術と、今回実証するVPSの特徴を比較すると以下のとおりである（一部仮説を含む）。

自動運転システムにおける自己位置推定技術の比較（一部仮説を含む）

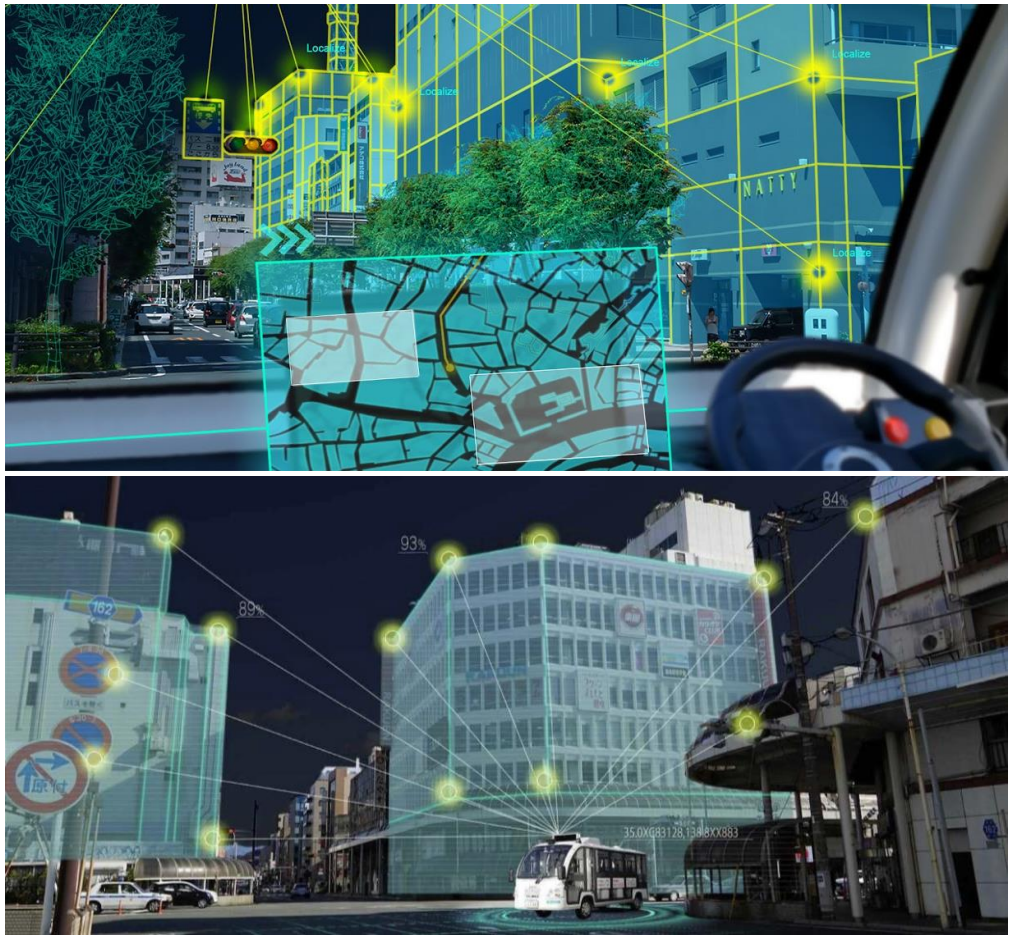
技術名	VPS×3D都市モデル（仮説）	SLAM	HDマップ
活用センサ	光学カメラ	LiDAR（Light Detection And Ranging）	ミリ波レーダー / カメラ / LiDAR、ジャイロ等（複数活用）
活用データ	3D都市モデル（LOD3）データ	3D点群データ	高精度3D地図データ（ベクトルデータ）
自己位置推定方法	光学カメラから取得した画像情報と3D都市モデルデータの特徴点とを比較し自己位置を推定。	LiDARで取得したデータを用いて地図データを作成し、その地図をもとに以降の走行時に自己位置を推定。	あらかじめ整備した3D地図データと車両のセンサーから取得したデータ等を比較し自己位置を推定（HDマップだけでなく他センサによる情報も統合的に活用して自己位置を推定）。
特徴・メリット	• SLAMやHDマップよりも安価に構築・運用できる可能性。	• 高精度3D地図が整備されていない区間でも自動運転車両を走行させることができる。	• ベクトル化、構造化、属性付与した地図のためデータ量は小さく、処理負荷は小さい。
課題・デメリット	• 夜間、大雨や雪など視界不明瞭時の運用は課題。	• 実走行前に、地図を生成するための準備走行が必要となる。 • データ量が大きいため、一定の処理負荷がかかる。	• 全国地図の整備・更新にコストがかかる。
適用場所	（主に）一般道	（主に）一般道	高速道路・自動車専用道路（一般道にも拡張中）
適用車両	（主に）バス等	（主に）バス等	（主に）一般車

VPSと3D都市モデル

既存のVPSでは、実写映像をもとに自己位置推定用マップを作成していた。しかし、この手法では現地で大量の映像を取得し処理する必要があることや、広範囲のデータを保守・管理することが困難であることなどから、Google社の提供するGeospatial APIを除き広範囲な都市規模でのVPSは存在していなかった。

前述のように、VPSはまちづくりのデジタルトランスフォーメーションやデジタルツインにおいて重要となる技術のひとつであり、これをオープンな環境で活用できるようにすることには大きな意義があると考えられる。3D都市モデルは、誰でも自由に活用できるオープンデータであり、VPSの自己位置推定のためのマップとして活用することが期待される。

このような背景から、本実証においては、3D都市モデルを活用したVPSの開発・検証を行った。



昨年度の実証成果と課題

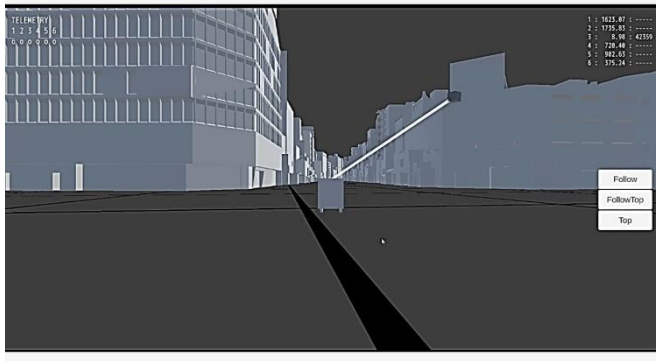
今回の実証に先立ち2021年度に実施した実証では、VPSとしてImmersal(https://immersal.com/)を用い、3D都市モデルを活用してマップデータを作成した。実証の結果としては、条件のよい場所での自己位置推定は可能であったが、実際のカメラ画像を使ったマップを利用した際と比較して自己位置推定に係る性能は劣ることが確認された。これは、Immersalが利用している技術が、画像の特徴点や特徴量をもとにしたものであり、主にテクスチャデータを使って位置合わせを行っているため、3D都市モデルが保有するテクスチャデータの精度・詳細度では十分な精度を出すことは困難だったためと考えられる。

Immersalを活用したVPSでは、テクスチャの精度や詳細度をあげる対応が必要であり、これには多大なコストがかかるため、精度保証がされている3D都市モデルの幾何形状を直接活かした仕組みが構築できないか、という仮説が設定された。

また、Immersalは独占的な商用ソフトウェアであるため、アルゴリズムの詳細な仕組みの理解や内部パラメータの調整などが困難であり、3D都市モデルを活用する際の最適化や車載での実行などのユースケースに対しての適用が困難であった。

このような課題に対応するため、今年度の実証では、VPS技術として、国立研究開発法人産業技術総合研究所の研究開発成果である「C*」（Cross-modal Simultaneous Tracking And Rendering。CSTAR。本技術検証レポート内では「C*」と記載する。）を用いた検証を行うこととした。

3D都市モデルと現地映像との重畳イメージ



VPSによる自己位置推定状況の可視化プログラム

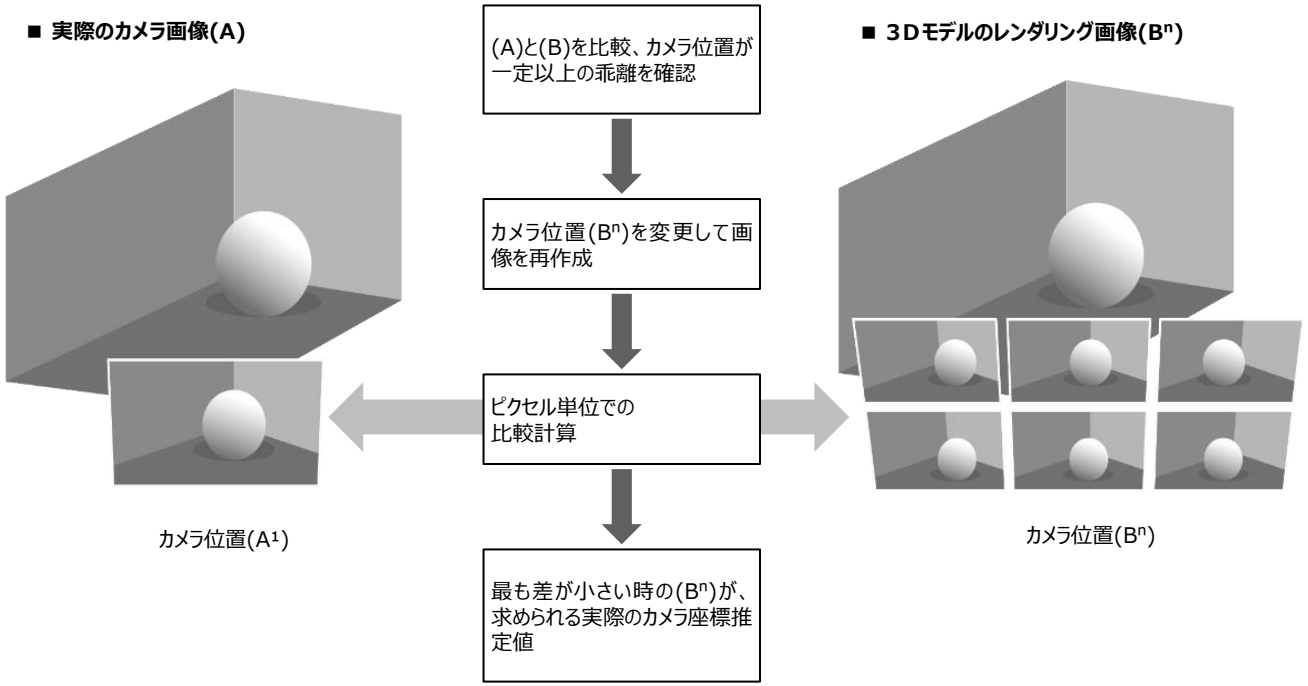
C*のアルゴリズム概要

C*は、国立研究開発法人産業技術総合研究所の大石氏を中心とするグループが開発した単眼カメラを用いたカメラ位置推定手法である。
ロボットにおける自己位置推定を主な目的として開発されているが、アルゴリズムとしては広範囲な応用が可能である。
C*の主要構成要素は以下3点である。

- **3Dモデルのレンダリングと実写画像との比較**
- **画像同士の類似度指標にNID（詳細後述）を活用**
- **最適化アルゴリズムにより自己位置の推定を実施**

また、C*はROS（Robot Operating System）、Linux、CUDAの環境で動作する。
C*は、カメラからの入力画像と3Dモデルをレンダリングした画像の類似度をもとにカメラ座標を推定するため、
既存のVPSの大半が使っている画像特徴点・特徴量を用いた方式と比べて以下の利点がある。

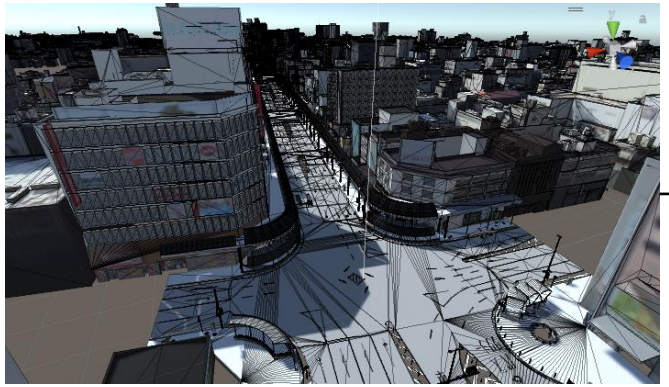
- **3Dモデルのみで自己位置推定が可能。**
- **点ではなく面で処理するためノイズやオクルージョンに強い。**
- **局所的な画像の特徴だけでなく、3Dモデルによる幾何学的形状を自己位置推定に活かすことができる。**



出所） S. Oishi, Y. Kawamata, M. Yokozuka, K. Koide, A. Banno and J. Miura, "C*: Cross-Modal Simultaneous Tracking and Rendering for 6-DoF Monocular Camera Localization Beyond Modalities," in IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5229-5236, Oct. 2020

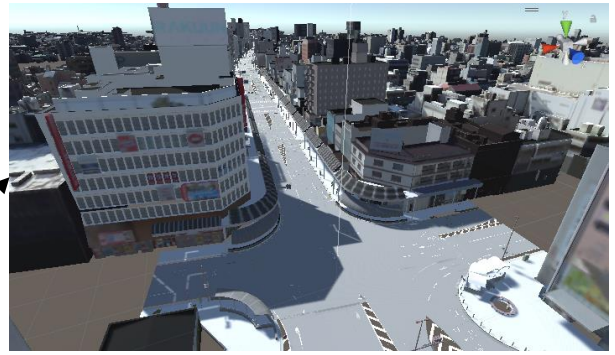
3Dモデルのレンダリング

C*では、レンダリングのためにOBJ形式の3Dモデルや、PLY形式の点群データを利用することができる。レンダリングを行う「Renderer」と呼ばれるC*のコンポーネントでは、通常のレンダリングと同様な色情報に加えて、その点が位置する三次元座標の情報をピクセルの情報として取得する。三次元座標情報は、後述の最適化アルゴリズム内で投影変換のために使われる。C*においてはRendererはOpenGLを使った独自のプログラムで実装されている。

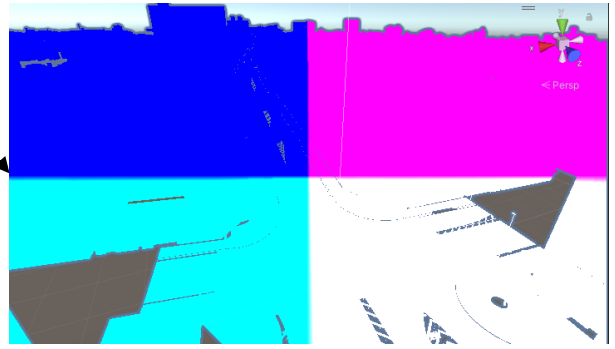


OBJファイルなどの3Dデータ
本実証では3D都市モデルから変換したものを使用

Renderer



色情報をレンダリングした画像



三次元座標をレンダリングした画像
各ピクセルにXYZの三次元座標の数値が格納されている

NID 画像どうしの類似度指標

NID(Normal Information Distance)は、C*では画像どうしの類似度の指標として使われている。以下の手法で計算される。

NID

Keyframe画像(3Dマップから生成される画像)をk、Current画像（カメラ画像）をtとして、NIDの計算は以下の通り。

$$NID = \frac{H(t,k) - I(t;k)}{H(t,k)}$$

$H(t,k)$と$I(t;k)$はそれぞれ、結合エントロピーと相互情報量で、計算は以下の通り。

結合エントロピー・相互情報量

$$H(t,k) = -\sum_{x,y} p_{tk}(x,y)\log_2\left(p_{tk}(x,y)\right)$$

$$I(t;k) = H(t) + H(k) - H(t,k)$$

$$H(t) = -\sum_{x} p_t(x)\log_2\left(p_t(x)\right) \qquad H(k) = -\sum_{y} p_k(y)\log_2\left(p_k(y)\right)$$

$p_{tk}(x,y)$は同時確率。$p_t(x)$, $p_k(y)$は周辺確率。

周辺確率

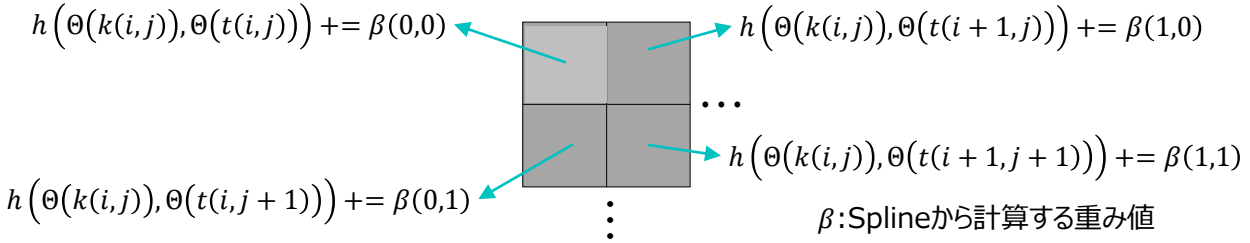
$$p_t(x) = \sum_{y} p_{tk}(x,y) \qquad p_k(y) = \sum_{x} p_{tk}(x,y)$$

同時確率

$p_{tk}(x,y)$計算のため、Keyframe画像とCurrent画像の画素値についての2次元ヒストグラム$h(x,y)$を作成する（xとyはそれぞれ、tとkのBinのインデックス番号）。
Current画像側は、Keyframe画像と一致する画素のほかに、近傍画素に対して、B-Splineに従う重みを加えている(図1)。

ヒストグラムから、同時確率$p_{tk}(x,y)$は、

$$p_{tk}(x,y) = \frac{h(x,y)}{\text{ピクセル数}}$$

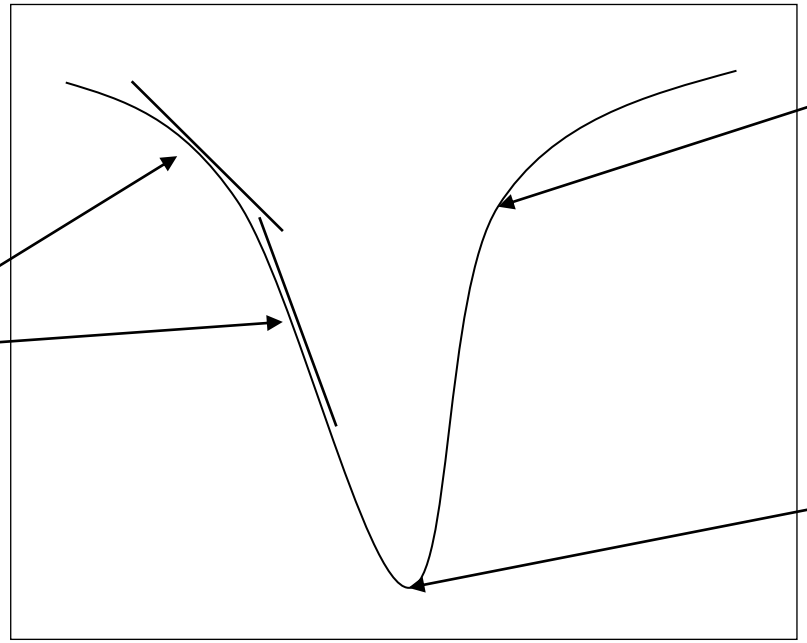


画素(i,j)におけるヒストグラムの計算

出所）S. Oishi, Y. Kawamata, M. Yokozuka, K. Koide, A. Banno and J. Miura, "C*: Cross-Modal Simultaneous Tracking and Rendering for 6-DoF Monocular Camera Localization Beyond Modalities," in IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5229-5236, Oct. 2020

最適化アルゴリズム

C*ではBFGS法(Broyden–Fletcher–Goldfarb–Shanno algorithm)という最適化アルゴリズムを利用して、自己の位置と回転を反復計算により求める。まず、位置と回転を入力するとそのカメラ姿勢でレンダリングした画像と実写画像とのNIDを出力する関数を作成し、この関数の微分を求めBFGS法を適用することにより、NIDの最小点（カメラ画像とレンダリング画像が最も類似している点）を探索し、解となる位置と回転を推定結果の自己位置とする仕組みとなっている。



位置と回転をパラメータとする画像同士のNIDの関数。実際には多次元の複雑な非線形関数となる。

関数の微分＝勾配を何回も計算しながらより値の小さな場所を探索する

関数の最小値＝実写画像とレンダリングした画像の差が最も小さい地点＝もっとも確からしい推定位置

S. Agarwal, K. Mierle, and Others, “Ceres solver,” http://ceres-solver.org (Accessed September, 15, 2019).

Ⅱ. 実証技術の概要 > 3. ADENU

ADENUの概要

概要

項目	内容
名称	ADENU
概要	• 名古屋大学が開発した自動運転パッケージ
主な機能	● 各種センサから情報を取得するための機能を提供 ● センサデータから交通環境の認識処理を実施 ● 地図データの管理、ダイナミックマップや信号情報のとの通信機能を提供 ● 自動走行に必要な行動判断と経路・制御計画を生成する機能を提供 ● センサや制御機器の異常を検知し、自動走行を安全に保つための機能を提供 ● 交通法規や走行時の制限事項などを設定する機能を提供し、自動走行機能をカスタマイズ ● 車両を制御するための指令値や車両側センサ情報等を車両側制御機器と通信する機能を提供
利用する機能	上記機能を統合した自動運転車両を実証に利用

ADENUロゴ

名古屋大学のADENU資料より抜粋

ADENUの概要

- 自動運転統合ソフトウェアは自動運転システムを設計する上で必要な機能を有しているソフトウェアであり、国内においては名古屋大学COI（Center of Innovation）が開発した自動運転ソフトウェアパッケージ「ADENU（Autonomous Drive Enabler by Nagoya University）」や、株式会社ティアフォーによるオープンソースソフトウェアである自動運転OS「Autowave」がある。
- 「ADENU」は自動運転車両などの自律走行に必要な7種類の機能（各種センサデータの取得、地図データの管理、地図とセンサを融合した走行環境理解、経路計画の生成、車両への制御指令機能等）を有しており、市街地でのMaaSや施設内の人や物の輸送などに利用可能なシステムである。
- 愛知県春日井市、豊田市、静岡県伊豆高原、下田市、沼津市、掛川市等でADENUを搭載したバス「ゆっくりミニバス」による実証が実施されている。

ADENUの7種類の機能

名称	概要	代表的な機能
Sensor	各種センサから情報を取得するための機能を提供	・カメラ画像の取得 ・LiDAR距離計測データの取得 ・GNSSによる位置データの取得 ・IMU情報の取得
Perception	センサデータから交通環境の認識処理を実施	・3次元点群地図による自己位置推定 ・画像処理による物標認識
Map	地図データの管理、ダイナミックマップや信号情報のとの通信機能を提供	・3次元点群地図の読み込み ・高精度地図の入出力 ・ダイナミックマップサーバとの通信 ・地図データ構造へのアクセス ・信号等インフラ側情報の取得
Autonomous	自動走行に必要な行動判断と経路・制御計画を生成する機能を提供	・障害物と地図とのデータ融合 ・交通規則に基づく行動判断 ・目的地までのルート検索 ・走行経路（速度・操舵計画）の生成
System monitor	センサや制御機器の異常を検知し、自動走行を安全に保つための機能を提供	・センサ系の通信監視 ・自己位置推定の安定化 ・異常値検出による自動停車
Setting	交通法規や走行時の制限事項などを設定する機能を提供し、自動走行機能をカスタマイズ	・走行規則の記述 ・センサパラメータ調整 ・速度等機能限界の設定
Control	車両を制御するための指令値や車両側センサ情報等を車両側制御機器と通信する機能を提供	・ヤマハ社製車両との通信制御 ・新明工業社製車両との通信制御 ・タジマEV社製車両との通信制御

出所）http://www.coi.nagoya-u.ac.jp/html/coiura/nu-coi_materials/material_ADENU_NU-COI.pdf

Ⅰ．実証概要

Ⅱ．実証技術の概要

Ⅲ．実証システム

Ⅳ．実証技術の検証

Ⅴ．成果と課題

実証フロー

今回の実証では自動運転車両に各種機材を搭載し、3D都市モデルを利用したVPSの自己位置推定の精度等を検証した。

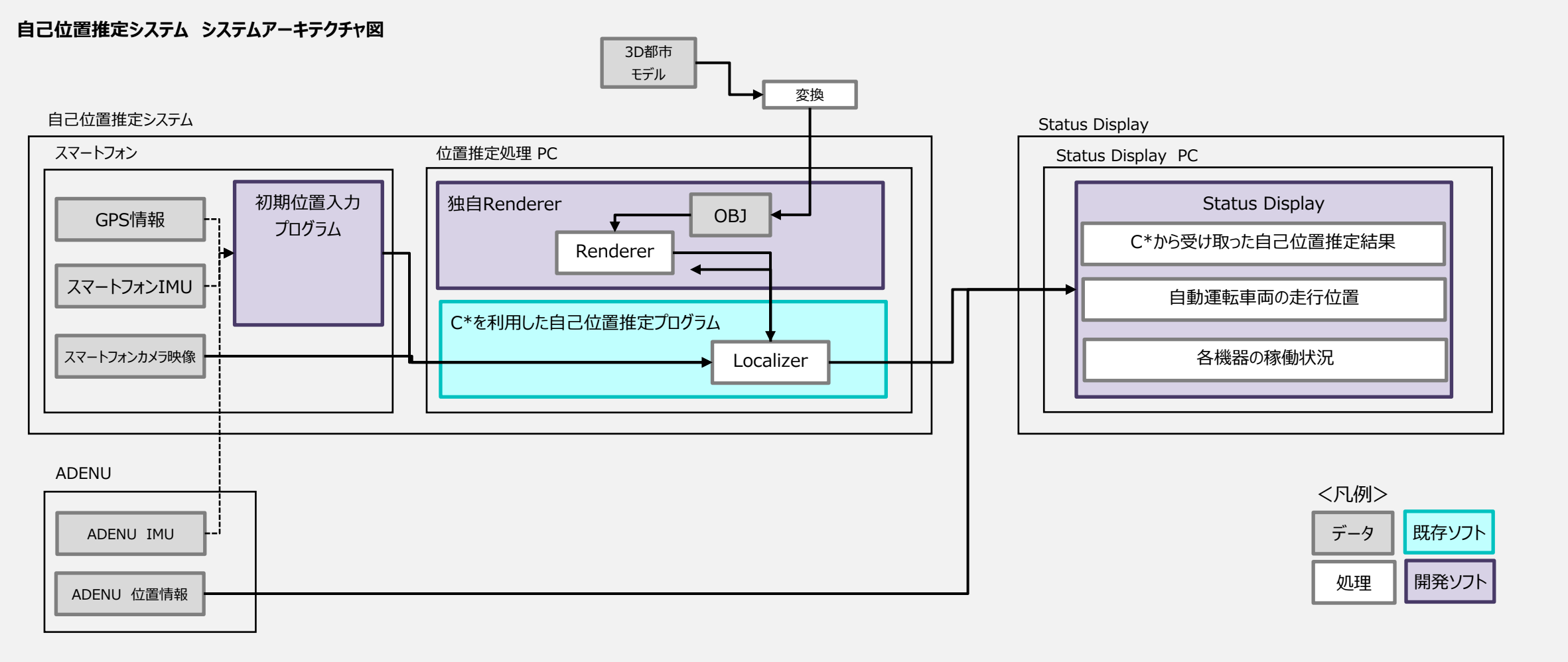
3D都市モデル入力	• 対象となる地域の建物、道路、都市設備、DEMの3Dモデル（LOD2またはLOD3）を、変換・軽量化してアプリに入力する。
セッティング	• 自動運転車両への機器設置、ネットワーク接続などを行う。
自動走行の実施	• 実証ルートを自動運転車両により走行し、以下の検証を行う。 - 自己位置推定の状況 - 各種条件（レンダリング設定、座標補間設定など）での走行データ取得 - ソフトウェアの動作確認
データ分析	• 取得したデータを分析し、位置推定精度やVPS動作の定性・定量分析を行う。

業務要件

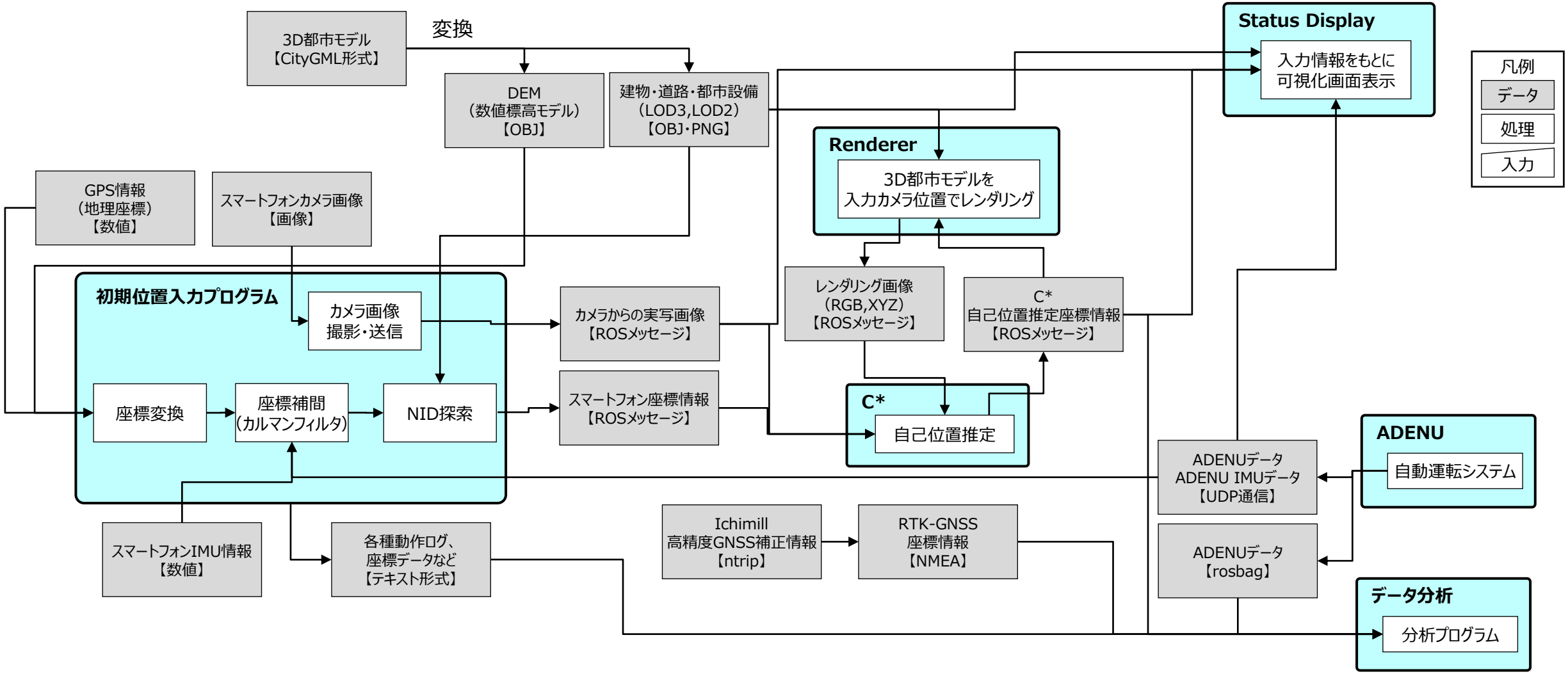
本実証は、将来的に自動運転のための自己位置推定を様々なモビリティに対して廉価に提供することを目的とする。以下に自動運転における自己位置推定で必要となる業務フローの中で、本実証の成果が想定する寄与を提示する

	従来の業務フロー	本システムが目指す業務フロー
①自動運転用マップ作成	• MMS搭載のLiDARによるスキャンなど、高コストな高精度マップの作成が必要	• すでに整備されている3D都市モデルを低コストで変換して活用
②自動運転システム	• LiDAR、RTK-GNSSなど、高価な機材を利用した自動運転システム	• スマートフォンなどのコモディティ化した機材を用いた廉価なシステム
③自動運転による走行	• 各種センサの情報を統合し、大規模な高精度マップを利用する、高負荷なシステム。 • 高度な処理能力を持つプロセッサが必要。	• スマートフォン程度の性能の小型コンピュータの活用などで、小型の個人向けモビリティや配送ロボットなどにも活用できるポータビリティ。
④その他	-	• 3D都市モデルから自動運転用マップを作成しているため、3D都市モデルとの連携（更新の反映、3D都市モデルへのFB）が可能

システムアーキテクチャ全体図



データアーキテクチャ全体図



システム機能

機能名	説明
1:C*を利用した自己位置推定機能	• 単眼カメラを用いて自己位置推定を行う。 • 本実証の中心であり、カメラ画像と初期位置の入力から自己位置推定を行い、推定した座標を出力する。 • 本機能を利用して自動運転車両の位置を取得する。
2:独自Renderer機能	• サイズの大きな3D都市モデルの高速なレンダリングや光源設定等レンダリング諸条件の変更が可能なレンダリング機能。 • 指定した位置からの3D都市モデルレンダリング画像と深度画像を出力する。 • C*の自己位置推定のための画像入力源となる。
3:初期位置入力機能	• C*に入力するカメラ画像と初期位置を送出する機能。 • スマートフォンで動作し、スマートフォンのカメラやGPS、IMUを利用する。
4:Status Display	• C*によるシステムの稼働状況や推定結果の座標、ADENU推定位置をまとめて表示するプログラム。 • C*とはROS#ライブラリとrosbridgeを介してネットワーク接続する。 • ADENUからは、UDP通信によりデータを受信する。

Ⅲ. 実証システム ＞ 4. システム機能
1:C*を利用した自己位置推定機能

PLATEAU by MLIT

機能仕様

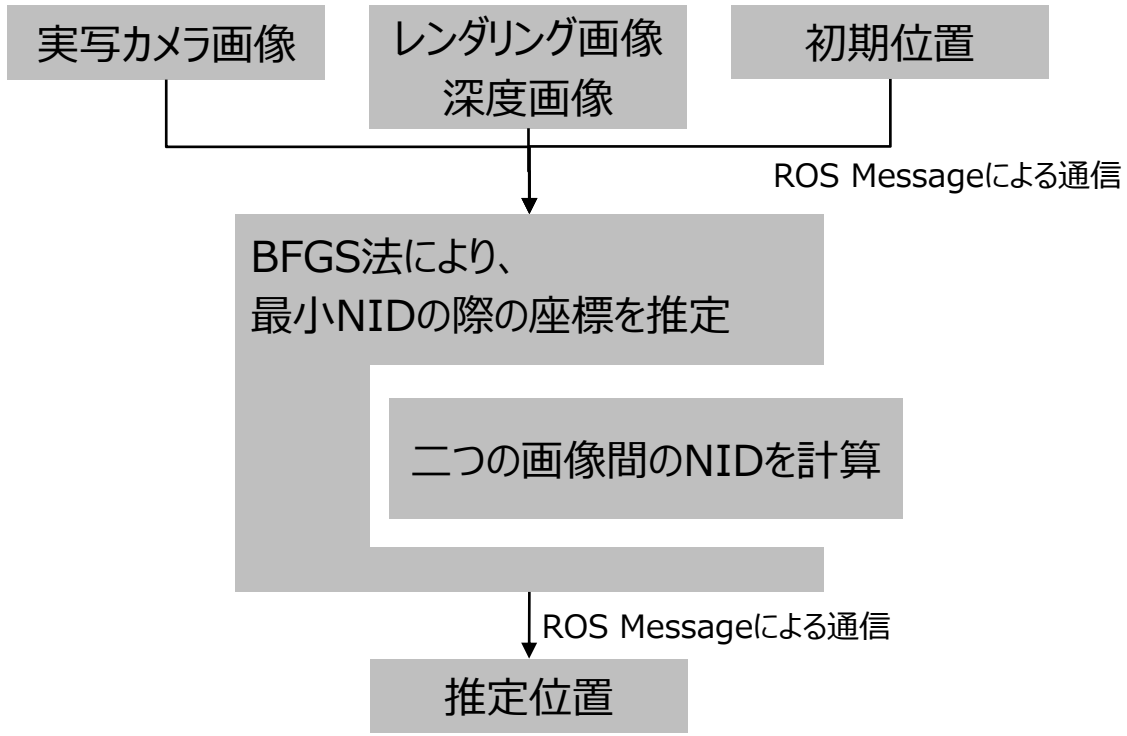
項目	詳細
機能名	1:C*を利用した自己位置推定機能
機能概要	産総研提供のC*のLocalizerと呼ばれる自己位置推定機能により、実写カメラ画像とレンダリング画像から自己位置の推定を行う。
入力データ仕様	実写カメラ画像（3:初期位置入力機能から送信されるJPEG圧縮された画像をROS Message形式で受信） 3D都市モデルレンダリング画像、深度画像（2:独自Renderer機能から送信されるJPEG圧縮された画像をROS Message形式で受信） 初期位置座標（3:初期位置入力機能から送信されるX,Y,Z＋回転の座標値をROS Message形式で受信）
出力データ仕様	自己位置推定結果の座標値（4:Status Display機能に対して、X,Y,Z＋回転の座標値をROS Message形式で送信）
利用するアルゴリズム	BFGS法、NID

利用ライブラリ・依存ソフトウェア

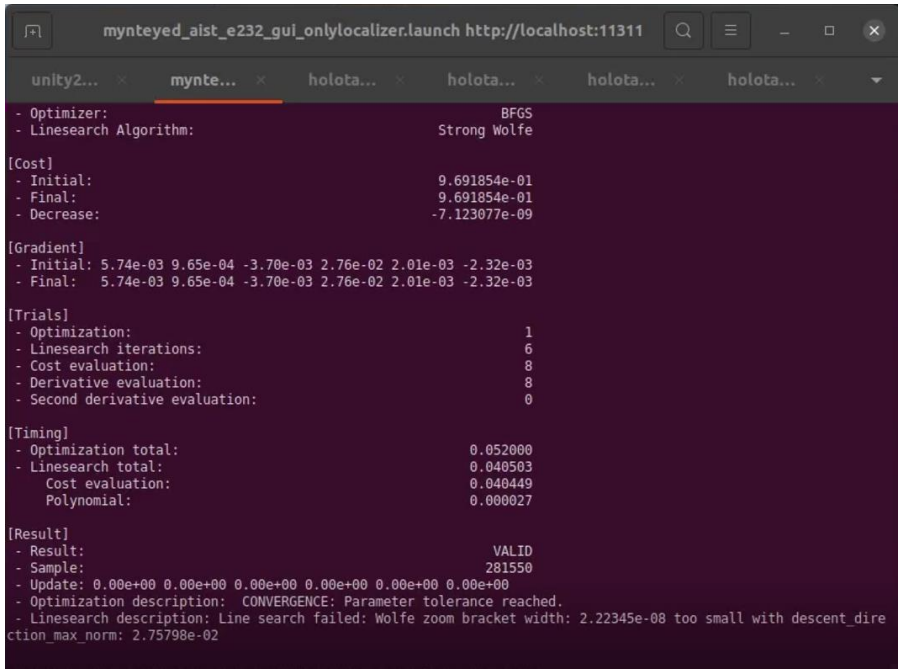
利用ライブラリ	用途
ROS noetic	Robot Operating System。メッセージのルーティングを基盤としたロボット用のOS。各機能間の連携・結合のため利用。
CUDA	NVIDIAのGPUを利用した高速並列計算のためのライブラリ。NID計算などの計算負荷の高い処理の高速化に利用
Eigen	線形代数のためのC++テンプレートライブラリ。ベクトル、行列の基本的な計算などに利用。
OpenCV	コンピュータビジョンのライブラリ、画像の処理で利用。
ceres	最適化問題のソルバー。BFGS法を実行するのに利用。
Linux	オペレーティングシステム。Ubuntu 20.04を利用。

Ⅲ. 実証システム ＞ 4. システム機能
1:C*を利用した自己位置推定機能

処理フロー



画面イメージ



CUIで動作するプログラムのためテキスト画面を提示

Ⅲ. 実証システム ＞ 4. システム機能
2:独自Renderer機能

機能仕様

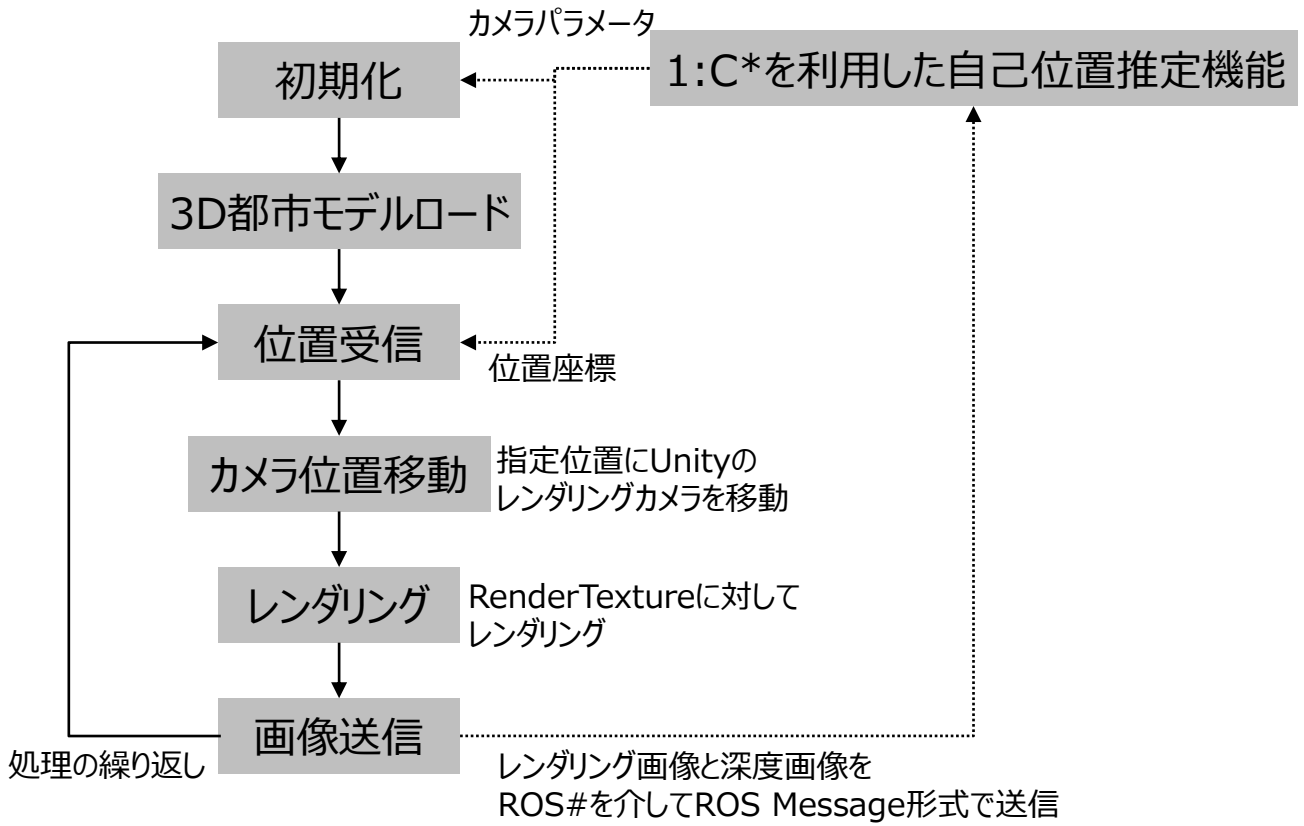
項目	詳細
機能名	2:独自Renderer機能
機能概要	自己位置推定機能から指定された位置で3D都市モデルのレンダリング画像と深度画像をレンダリングし、送出する。
入力データ仕様	カメラパラメータ（起動時に1回実行される。設定ファイルで指定するか、1:C*を利用した自己位置推定機能からROS Message形式で受信） 位置座標（1:C*を利用した自己位置推定機能からX,Y,Z＋回転の座標値をROS Message形式で受信） 軽量化された3D都市モデル。プログラムビルド時に組み込み。
出力データ仕様	3D都市モデルレンダリング画像・深度画像（JPEG圧縮された画像をROS Message形式で送信）
利用するアルゴリズム	-

利用ライブラリ・依存ソフトウェア

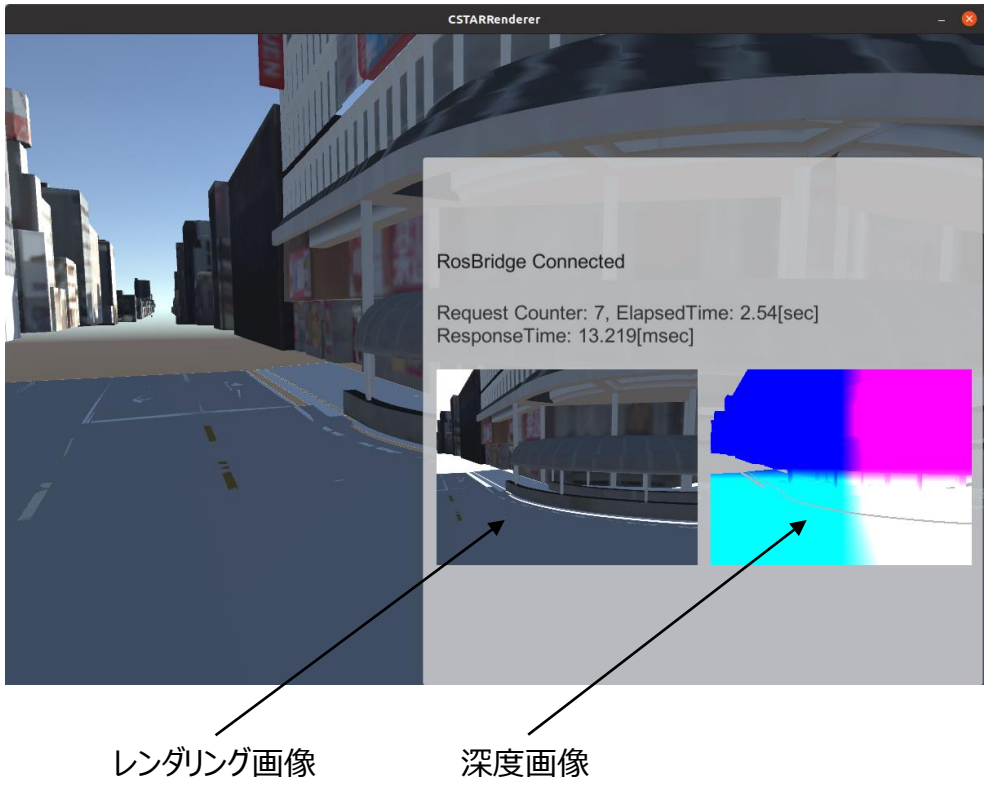
利用ライブラリ	用途
Unity	3Dゲームエンジン。高速で精細な3Dレンダリングが可能。独自シェーダーを用いて、RGB画像とXYZ画像をレンダリングするために利用。
ROS#	C#で実装されたROSのネットワークと接続するためのライブラリ。1:C*を利用した自己位置推定機能と接続するために利用。
rosbridge	Websocket上のJSONメッセージとROSネットワークのROS Message形式とを接続するサーバー。ROS＃はこのサーバーを介してROSネットワークと接続する。

Ⅲ. 実証システム ＞ 4. システム機能
2:独自Renderer機能

処理フロー



画面イメージ



2:独自Renderer機能

レンダリング仕様

2:独自Renderer機能では、指定された位置からのレンダリング画像と深度画像を生成する。

レンダリング画像は、UnityのStandardシェーダーを用いた一般的なレンダリングだが、深度画像は専用のシェーダーを用いて、レンダリング結果の各ピクセルのRGB値にピクセルの色ではなくピクセルの座標を格納する。

頂点シェーダーでは、各頂点にワールド座標の座標値を格納する。

フラグメントシェーダーでは、各頂点のワールド座標がGPUにより自動的に線形補間されて入力されるので、そのX,Y,Zの値をそれぞれ、R,G,Bの出力に格納する。

シェーダープログラム

```
v2f vert (appdata_base v)
{
  v2f o;
  float3 n = UnityObjectToWorldNormal(v.normal);
  o.uv = UnityObjectToClipPos (v.vertex);
  o.worldPos = mul(UNITY_MATRIX_MV, v.vertex).xyz;
  return o;
}
float4 frag (v2f i) : SV_Target
{
  return float4 (i.worldPos.x, -i.worldPos.y, -i.worldPos.z, 1.0);
}
```

Ⅲ. 実証システム ＞ 4. システム機能
3:初期位置入力機能

機能仕様

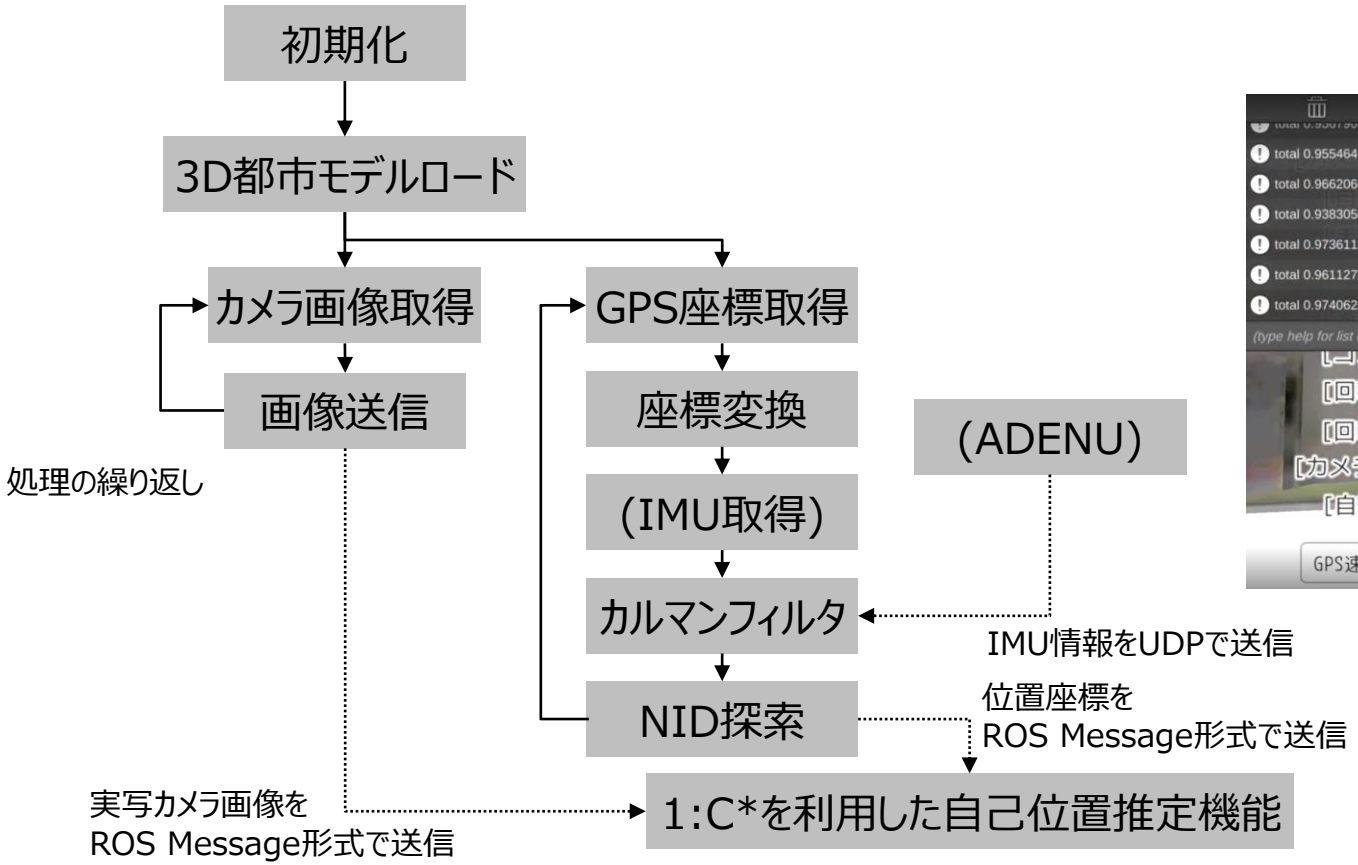
項目	詳細
機能名	3:初期位置入力機能
機能概要	1:C*を利用した自己位置推定機能に入力するカメラ画像と初期位置を送出する機能。スマートフォンのカメラ映像とGPSで取得した初期位置座標を送信する。
入力データ仕様	GPS座標（WGS84の緯度、経度、楕円体高を取得） カメラ画像（Unityの機能から取得） 3D都市モデル（高さ算出のため地形モデル(DEM)を利用。NID探索のためにLOD3の建物、道路、都市設備を利用。OBJに変換してビルド時に組み込み） スマートフォンIMU情報（Unityの機能から取得） ADENUのIMU情報（UDP通信で受信）
出力データ仕様	実写カメラ画像（JPEG圧縮された画像をROS Message形式で1:C*を利用した自己位置推定機能に送信） 初期位置座標（X,Y,Z＋回転の座標値をROS Message形式で1:C*を利用した自己位置推定機能に送信）
利用するアルゴリズム	カルマンフィルタ、NID探索

利用ライブラリ

利用ライブラリ	用途
Unity	3Dゲームエンジン。高速で精細な3Dレンダリングが可能。スマートフォン端末のセンサー取得のAPIもあるため、実装の基盤として利用。
ROS#	C#で実装されたROSのネットワークと接続するためのライブラリ。1:C*を利用した自己位置推定機能と接続するために利用。
rosbridge	Websocket上のJSONメッセージとROSネットワークのROS Message形式とを接続するサーバー。ROS＃はこのサーバーを介してROSネットワークと接続する。
Native Toolkit	Unity用のライブラリ。スマートフォンのセンサなどを扱う機能を提供する。GPSの座標取得に利用。

3:初期位置入力機能

動作フロー



画面イメージ

4:Status Display

機能仕様

項目	詳細
機能名	4:Status Display
機能概要	走行検証中、システムの稼働状況や推定結果をリアルタイムに確認できるように、各機能から出力される情報を表示する機能。
入力データ仕様	3D都市モデル（表示のためにLOD3の建物、道路、都市設備を利用。OBJに変換してビルド時に組み込み） ADENUの情報（ADENUからUDP通信で受信） 実写カメラ画像（3:初期位置入力機能からJPEG圧縮された画像をROS Message形式で受信） 自己位置推定結果の座標値（1:C*を利用した自己位置推定機能から、X,Y,Z＋回転の座標値をROS Message形式で受信）
出力データ仕様	現在の各種ステータスなどを画面に表示する
利用するアルゴリズム	-

利用ライブラリ

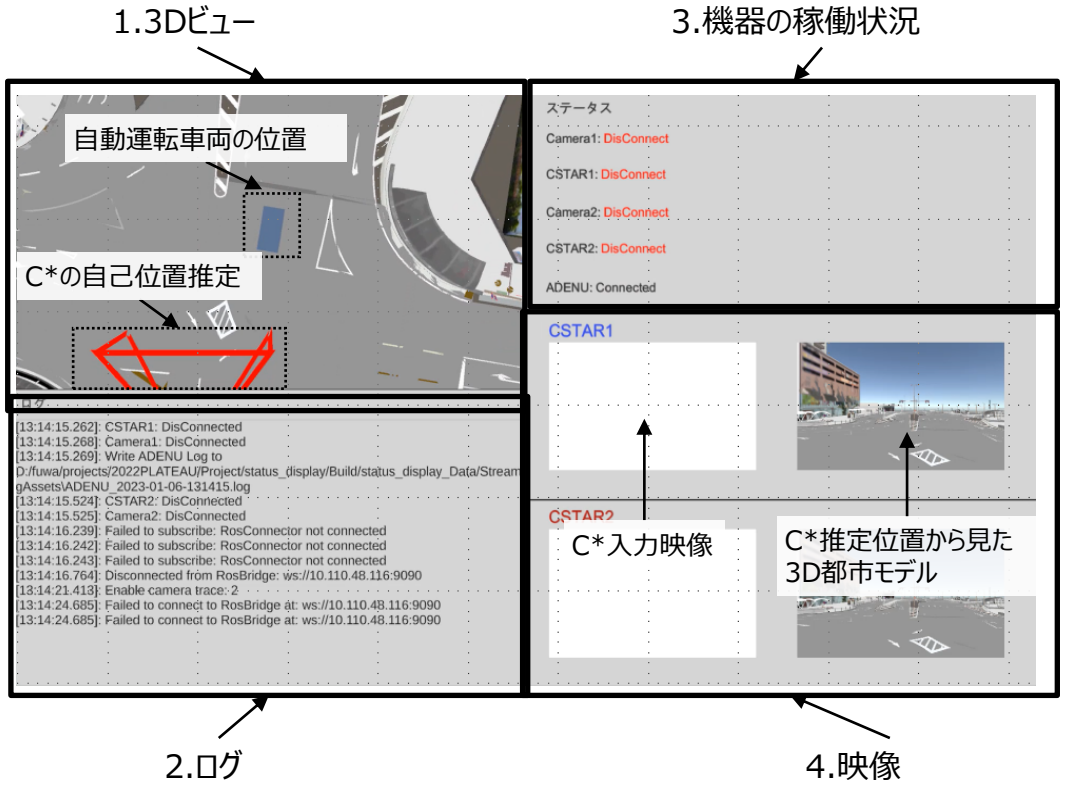
利用ライブラリ	用途
Unity	3Dゲームエンジン。高速で精細な3Dレンダリングが可能。実装の基盤として利用。
ROS#	C#で実装されたROSのネットワークと接続するためのライブラリ。1:C*を利用した自己位置推定機能と接続するために利用。
rosbridge	Websocket上のJSONメッセージとROSネットワークのROS Message形式とを接続するサーバー。ROS＃はこのサーバーを介してROSネットワークと接続する。

4:Status Display

Status Displayの機能詳細

項目	詳細
3Dビュー	C*の自己位置推定と自動運転車両の位置を、3D都市モデル上に表示する。
ログ	Status Displayアプリのログを表示する。
機器の稼働状況	スマートフォン、位置処理推定PC、ADENUの稼働状況を表示する。
映像	C*に入力された映像と、C*の推定位置から見た3D都市モデルを表示する。

Status Displayの画面



アルゴリズム

機能名	説明
1:C*を利用した自己位置推定プログラム	• 画像の類似度判定にNIDアルゴリズムを利用 • 最適化アルゴリズムのパラメータ調整
2:独自Renderer	• Rendererの詳細実装 • 3D都市モデルの軽量化
3:初期位置入力プログラム	• スマートフォンGPS・DEMからの位置取得と座標変換 • カルマンフィルタによる座標補間 • NID探索による初期位置の精度向上
4:Status Display	• C*やADENUからデータ受信

1:C*を利用した自己位置推定プログラム

C*初期検証と課題

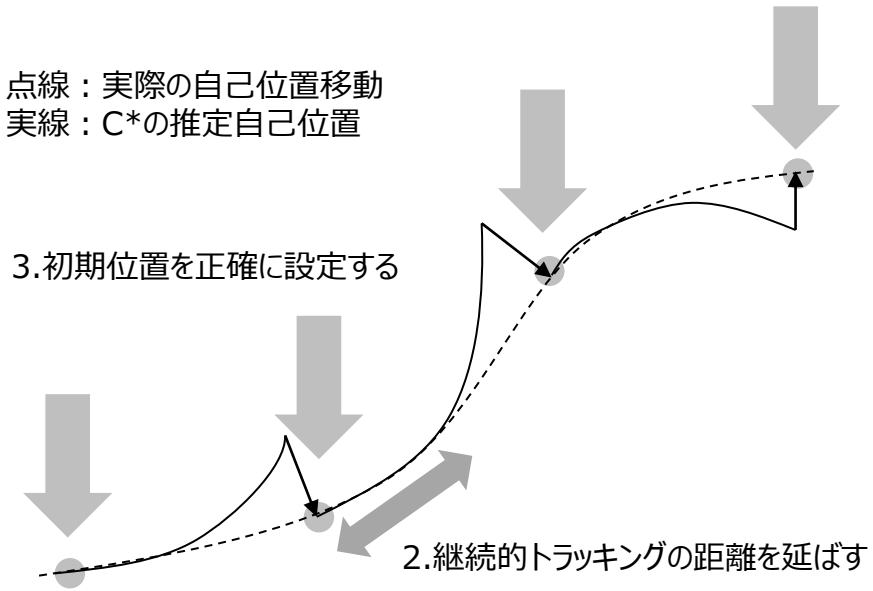
本実証の最初期に、産業技術総合研究所提供のバイナリ版C*についての動作確認と基本的な検証を実施した。検証では産業技術総合研究所から提供されたシステムの3Dモデルのみを3D都市モデルに変更し、実証対象地域の沼津駅前商店街にて動作させた。その結果、自己位置推定自体は動作したが、以降のシステム構築にあたっての課題として、大きく以下3点を抽出した。

1. **C*で使用している画像類似度指標であるNIDは実写画像と3D都市モデルレンダリング画像に対して有効か**
2. **継続的トラッキングが困難である**
3. **初期位置入力機能の欠如**

1.は、C*の3D都市モデルに対する適用可能性の問題である。

2.は、右図の模式図でC*の推定自己位置がなるべく長時間実際の位置に沿うように、トラッキングの持続時間を延ばすための施策の検討を要するという課題である。

3.は、そもそもC*の初期実装では手動や設定ファイルに書かれた初期位置を用いているため、その機能自体の開発、および、より精度の高い初期位置計算手法の検討が必要であるという課題である。



1:C*を利用した自己位置推定プログラム

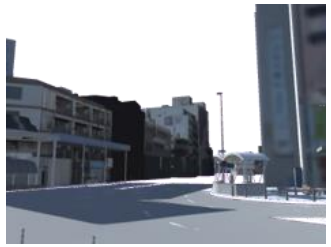
1. NIDの有効性

産業技術総合研究所から提供されたデモデータ（レーザースキャナによる詳細度の高い3Dモデルを使用）での実行時と比較して、3D都市モデルを使用した実行時には自己位置推定が安定しない傾向が見られた（すぐにトラッキングが失敗し、推定座標が大きくずれるなど）。

この結果から実写に近い3Dモデルと比較して、3D都市モデルに対してNIDは有効な画像類似度指標となっているかの確認が必要と考え、実際に画像どうしのNIDを計算する単体のプログラムを作成し、実写画像とUnityで作成した3D都市モデルのレンダリング画像とのNIDについて、テクスチャ解像度、照明条件、比較場所を様々に変えて計算し検証した。

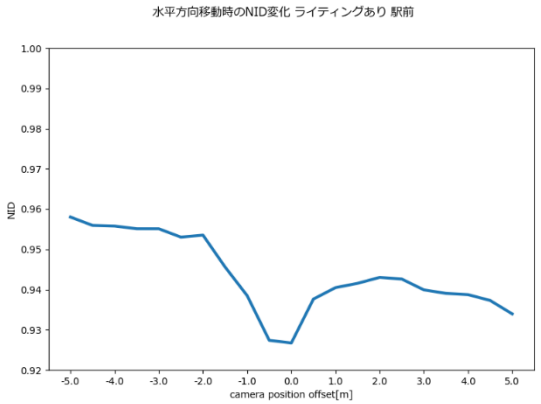
この結果、各条件にて、実写視点位置付近でNIDが極小となることが確認できた。

NID計算対象の実写画像

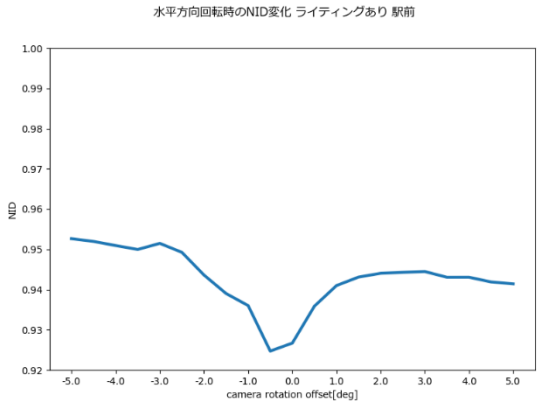


レンダリング画像

レンダリング条件、カメラ位置を変えてNIDを計算、グラフ化



水平方向に3D空間のカメラを移動させたときのNID



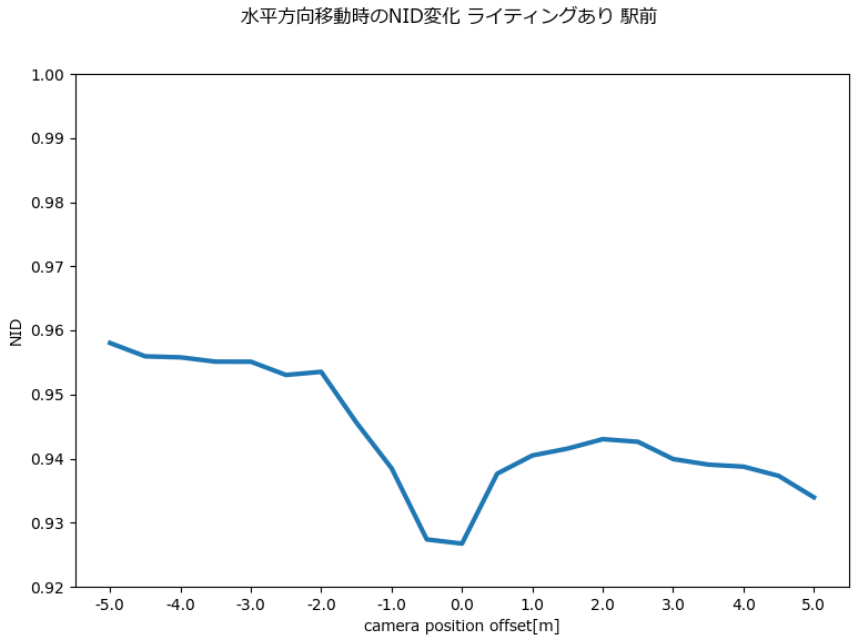
左右方向に3D空間のカメラを回転させたときのNID

Ⅲ. 実証システム ＞ 5. アルゴリズム
1:C*を利用した自己位置推定プログラム

1. NIDの有効性

実写画像とレンダリング画像のNIDを計算し、実写カメラ位置付近でNID極小値となったことから、NIDは実写画像と3D都市モデルのレンダリング画像で有効であることがわかった。

一方で、NIDが落ち込む範囲が狭く、この範囲から探索を始めないと真値に到達できないことから、初期値が真値に近い位置にないと推定が困難であると考えられることがわかった。これは初期位置設定の精度の重要性につながる点である。



水平方向の位置の変位でのNID変換のグラフ

この場合、-2.0m以下や+2.0m以上で、NIDの増減が反転しており、もし推定の初期位置が-2.0～2.0の範囲外だと、正しい位置を推定できない可能性が高い。

1:C*を利用した自己位置推定プログラム

システムの開発・実装の方針

前述のとおり、自己位置推定におけるNIDの有効性を検証できたため、残る2つの課題に対して以下の検討、実装を行った。

- **2. 継続的トラッキングの課題**
  - C*のパラメータ調整
  - C*用独自Renderer実装
  - 3D都市モデルの3Dデータ軽量化

- **3. 初期位置入力の課題**
  - スマートフォンGPSによる初期位置入力
  - 3D都市モデルのDEMからのカメラ高さの計算
  - センサフュージョンによる座標補間
  - NID計算の高速化と初期位置探索

1:C*を利用した自己位置推定プログラム

継続的トラッキングの課題-C*のパラメータ調整

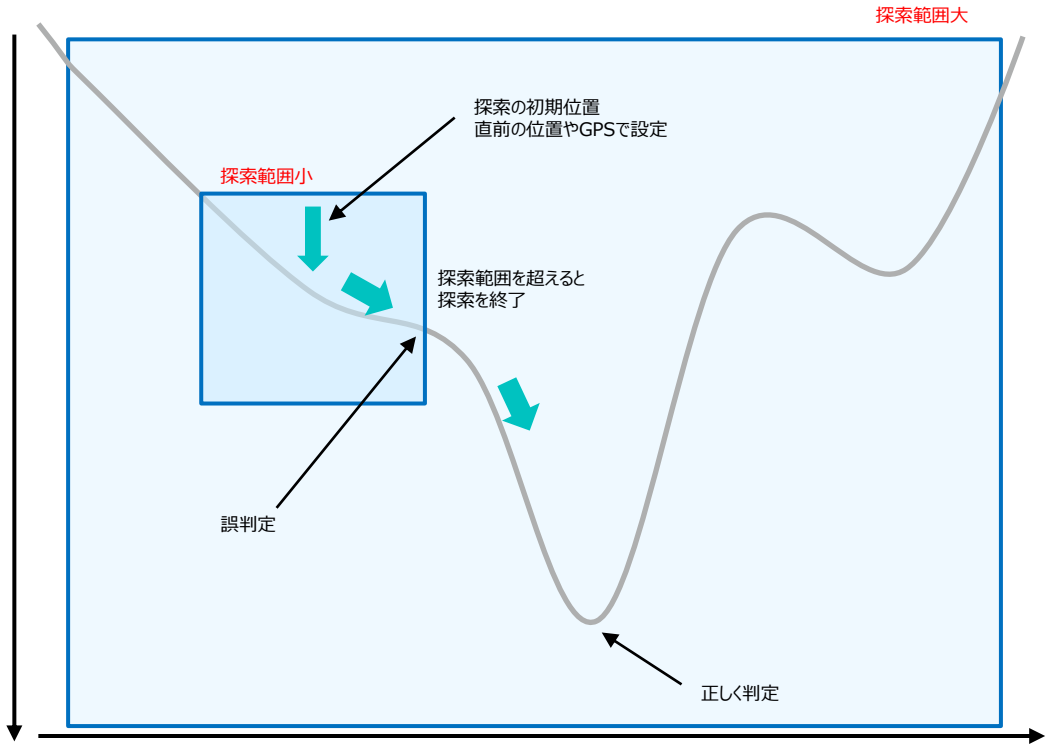
C*のパラメータ調整では、linesearch_trans_boundaryとlinesearch_rot_boundaryの2パラメータを既定の設定より大きくすることで、自己位置の探索範囲を拡大し、トラッキングが失敗することを抑制した。

これらのパラメータは、C*内部での自己位置を探索する際の範囲上限として設定されており、初期値からの変位がこの設定値を超えた段階で探索終了となるものである。

このため、値が小さいと局所的な最小値に落ち込みやすく、範囲外に解があった場合に正しく探索できない。

一方で値を大きくすると探索範囲が増え、処理負荷が増大する。

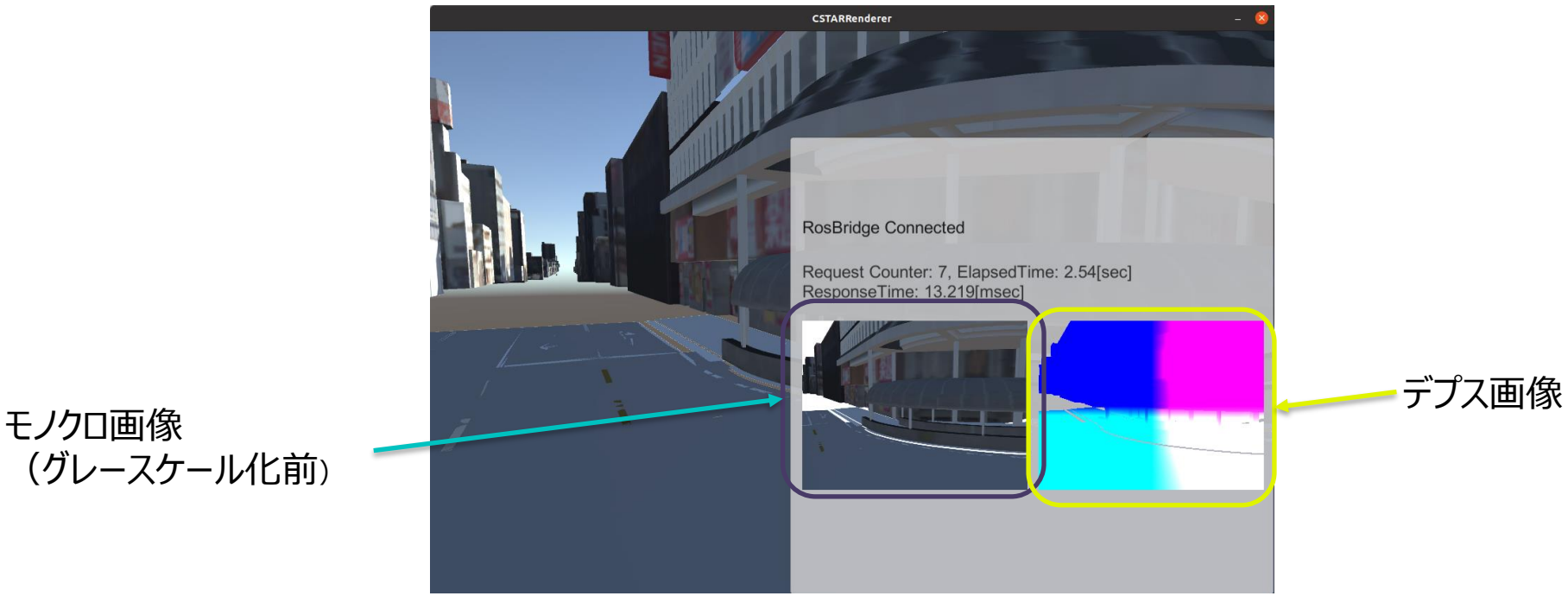
今回いくつかの設定値を検証し、linesearch_trans_boundary=15という設定を採用した。



2:独自Renderer

継続的トラッキングの課題-C*用独自Rendererの実装

C*のオリジナルのRendererの実装はOpenGLを使った簡単な実装のため、サイズの大きなデータの表示で実用的な速度でのレンダリングができず、表現力の観点でも細かい設定やリアルなレンダリングができないため、3Dレンダリングに特化しているUnityで独自実装を行った。C*との接続には、ROS#ライブラリを用い、rosbridge経由で接続する構成とした。

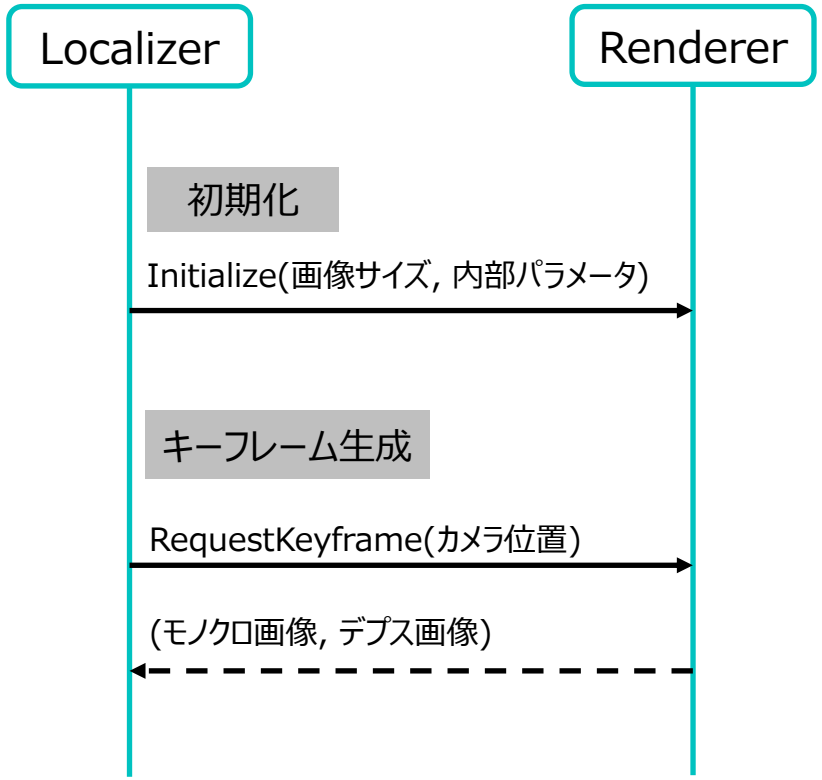


独自Renderer実行画面

2:独自Renderer

継続的トラッキングの課題-C*用独自Rendererの実装

- Rendererの外部化にあたって、提供されたC*のソースコードを調査し、LocalizerとRendererの連携手順を確認した。
- LocalizerとRendererはクライアントとサーバの関係になっており、また通信方法としてROSのサービス機能を使用している。
- LocalizerとRendererの連携部分の動作は以下のとおりである。
  - 初期化
    1. C*起動時に、LocalizerからRendererへ画像サイズやカメラの内部パラメータを送信
    2. Rendererはそれらのパラメータを記録
  - キーフレーム生成
    1. ローカライズ処理中、Localizerでキーフレームの更新が必要になったタイミングで、Rendererに対してキーフレーム描画を要求
    2. Rendererは、カメラ位置から以下の2つの画像を生成
       - モデルをモノクロでレンダリングした画像（モノクロ画像）
       - カメラからの距離を表した、デプス画像
    3. Rendererは、Localizerへ2つの画像を送信
- これらの情報から、Rendererの外部化を行うには、上記Rendererのサービスを模倣すればよいことがわかった。



LocalizerとRendererのシーケンス図

Ⅲ. 実証システム > 5. アルゴリズム
2:独自Renderer

継続的トラッキングの課題- C*用独自Rendererの実装

- 調査結果をもとに、独自Rendererを制作した。
- 独自RendererはUnityで作成した。理由としては、Unityエディタにより、光源設定などのレンダリング諸条件の変更が容易であり、またサイズの大きいモデルも高速にレンダリングできるためである。
- **独自Rendererの機能は以下の通り**
  - **レンダリング用のカメラの設定**
    - Localizerから受け取るカメラの内部パラメータから、レンダリング用のカメラの設定を行う(右図1参照)。
  - **キーフレーム要求**
    - モノクロ画像とデプス画像を生成する。モノクロ画像は、Unityのスタンダードマテリアルを割り当てたモデルに対してレンダリング後、グレースケール化する。デプス画像は各ピクセルの位置を出力するシェーダを使ってモデルをレンダリングする。
  - **Localizerとの通信**
    - Unityは直接ROSを使用することができないため、ROSの通信を仲介するROS#[1] を使用し、Rosbridge[2]を経由して、C*のLocalizerとROSメッセージをやり取りすることとした(右図2参照)。

```
// camera: Unityのカメラ
// Fx, Fy: 焦点距離
// Cx, Cy: カメラ中心
// Width, Height: 画像サイズ
var sizeX = camera.focalLength * Width / Fx;
var sizeY = camera.focalLength * Height / Fy;
var shiftX = (Cx - Width * 0.5f) / Width;
var shiftY = (Cy - Height * 0.5f) / Height;
camera.sensorSize = new Vector2(sizeX, sizeY);
camera.lensShift = new Vector2(shiftX, shiftY);
```

図1 レンダリング用カメラの設定[3]

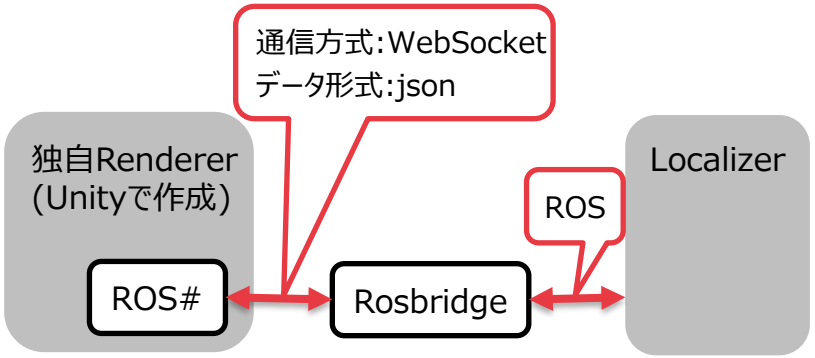


図2 RendererとLocalizerのデータ通信

[1] https://assetstore.unity.com/packages/tools/physics/ros-107085
[2] http://wiki.ros.org/rosbridge_suite
[3] https://answers.unity.com/questions/814701/how-to-simulate-unity-pinhole-camera-from-its-intr.html

2:独自Renderer

3D都市モデルの3Dデータ軽量化

C*では、処理速度が間に合わないとトラッキングが継続しないため、各コンポーネントの処理速度を向上させる必要がある。特に、Rendererによる3Dモデルのレンダリングでは、3D都市モデルの大容量のデータを適切かつ高速にレンダリングする必要がある。

3D都市モデルのLOD3データは、建物ごとにテクスチャが用意されているが、VPSには必要のない領域も多い。また、走行実証ではLOD3とLOD2の両方のモデルを扱うため、かなり大きな3Dモデルを使用することになる。

このため、 Rendererで使用する3D都市モデルに対して、Unity上のSimplygonプラグインにより、小範囲での分割、および、分割単位でのテクスチャのアトラス化、メッシュの結合の処理を行った。

3D都市モデルの分割（色分け表示）

3:初期位置入力プログラム

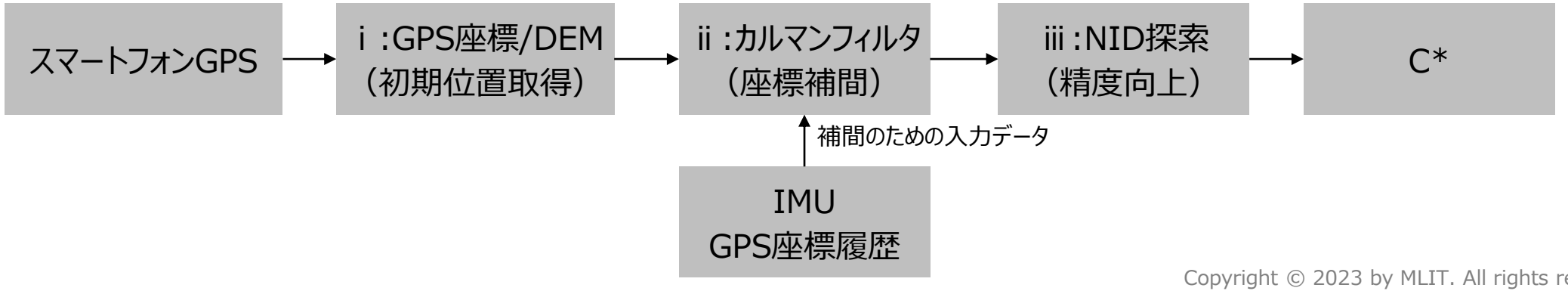
初期のC*の構成では、一度C*が自己位置推定に失敗し座標が大きく外れてしまうと、それを復帰させるための仕組みが存在しない。この結果、正常な状態に戻すためには、都度人力で初期位置の座標を設定する必要があった。実際に自動運転車両に搭載して検証するためには、自動的に初期位置を設定する機能を開発する必要がある。また、「C*初期検証と課題-1．NIDの有効性」の検証の結果、初期位置を真値から2mよりも小さい範囲に設定することで、自己位置推定が正しく行われる可能性が高まることが分かったため、より真値に近い位置を初期位置として設定するためのアルゴリズムを開発する。

このため、位置精度が十分ではなく、データ取得も低頻度なスマートフォン搭載GPSの座標を段階的に補間し、C*の座標系に変換して初期位置として出力するための以下の３つの機能を実装した。（詳細は次ページ以降）

ⅰスマートフォンGPSと3D都市モデルの地形モデル（DEM）を使用して三次元座標を出力する機能、

ⅱカルマンフィルタによりGPS座標を補間して出力する機能

ⅲNID探索によりカルマンフィルタの座標の精度をさらに高めて出力する機能

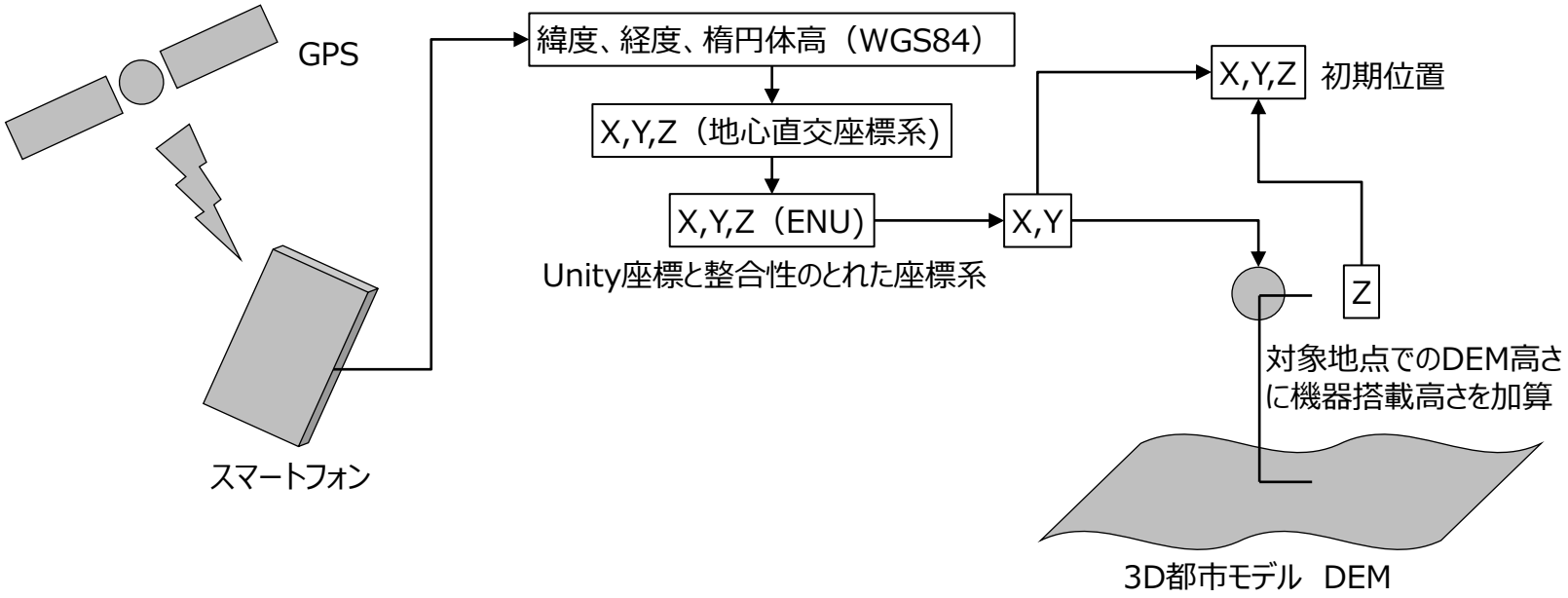


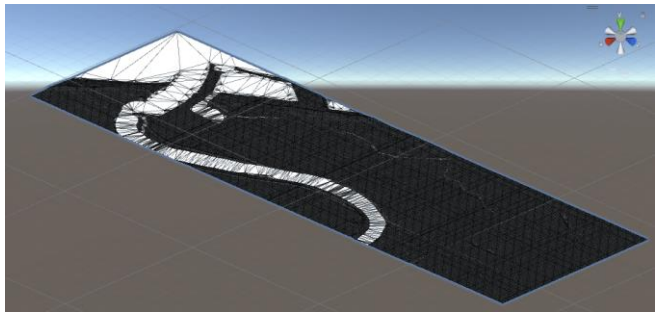
3:初期位置入力プログラム

ｉ:GPS座標/DEM　初期位置入力機能詳細　スマートフォンGPSとDEMによる座標取得

初期位置入力元として、スマートフォン搭載のGPSを用いた。スマートフォンのGPSは座標をWGS84の緯度経度と楕円体高で出力するため、地心直交座標系（ECEF）に変換し、さらに検証地域内の原点を緯度経度楕円体高で指定し回転処理をしてENU座標系に変換した。（次項で詳細記載）

スマートフォンGPS座標の垂直方向の精度が低いため、より正確な高さを取得するために水平方向の座標値の地点の3D都市モデルのDEMからスマートフォン搭載高さを加算し初期位置の高さとした。





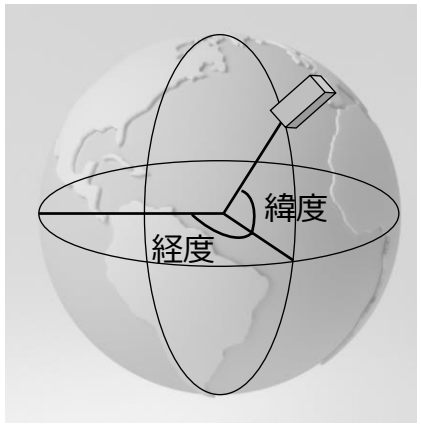
実際に利用した対象地域の3D都市モデルのDEM

Ⅲ. 実証システム ＞ 5. アルゴリズム
3:初期位置入力プログラム

ｉ:GPS座標/DEM　座標変換詳細

スマートフォンのGPS座標の緯度、経度、楕円体高を地球の中心を原点とする三次元直交座標に変換し、さらにUnity上の座標系の方向に合わせるための移動（原点の移動）、回転操作を行った。

平面直角座標への変換のような投影変換による歪みが極力生じないように、このような変換を採用している。



スマートフォンのGPSからの座標
緯度、経度、楕円体高（WGS84）

$$e^2 = f(2-f)$$

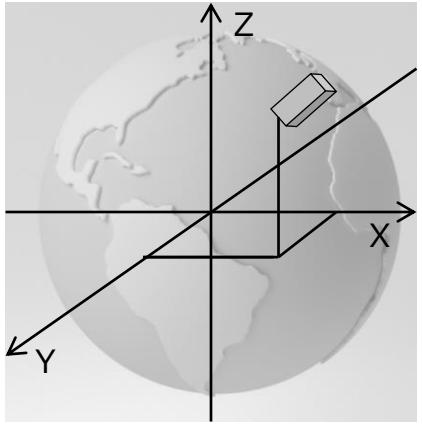
$$N = \frac{a}{\sqrt{1-e^2 sin^2 B}}$$

$$X = (N+H)cosBcosL$$

$$Y = (N+H)cosBsinL$$

$$Z = (N(1-e^2)+H)sinB$$

ただし、緯度B、経度L、楕円体高H、長半径a、扁平率fであり、GRS80楕円体のパラメータ（a=6378137m,1/f=298.257222101）を使用する。
参考：世界測地系と座標変換(飛田 幹男)、国土地理院Webページ



地心直交座標系（ECEF）
X、Y、Zの座標としてあらわす

$$\theta = -(L)$$

$$X' = xcos\theta - ysin\theta$$

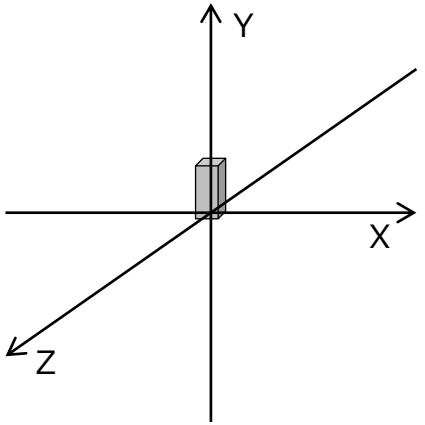
$$Y = xsin\theta + ycos\theta$$

$$\varphi = 90 - B$$

$$X = X'cos\varphi - zsin\varphi$$

$$Z = X'sin\varphi + zcos\varphi$$

ただし、原点の緯度B、経度Lとし、x,y,zをECEFの座標を原点座標でオフセットしたものとする。



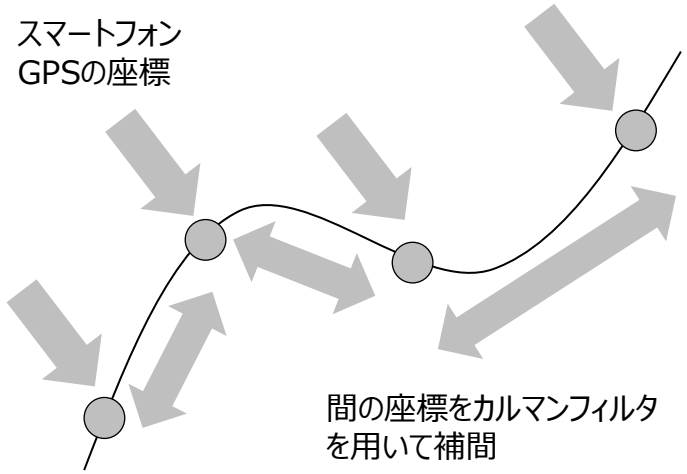
Unityに合わせた座標系（ENU）
実際の上方向を
Y軸（Unityの上方向）と合わせる

Ⅲ. 実証システム ＞ 5. アルゴリズム
3:初期位置入力プログラム

ⅱ:カルマンフィルタ　センサフュージョンによる座標補間

スマートフォンのGPSによる位置取得頻度は、C*の計算より低頻度なため、位置を補間するためカルマンフィルタを用いた。カルマンフィルタは、離散的な誤差のある観測値から時間変化する量を推定するためのアルゴリズムであり、ロケットの軌道推定にはじまり、ロボットや自動運転での位置の補間などに活用されている実績のある仕組みである。

カルマンフィルタでは、現在の座標とセンサの観測値を用いて、次の座標を予測し、実際に次の座標が得られたときに予測値との差分を用いて予測パラメータを調整する過程を繰り返す。これにより、精度の高い座標補間が可能になる。



予測 [編集]

$$\hat{\boldsymbol{x}}_{k|k-1} = F_k \hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{u}_k \quad \text{（今の時刻の予測推定値）}$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^{\mathrm{T}} + G_k Q_k G_k^{\mathrm{T}} \quad \text{（今の時刻の予測誤差行列）}$$

更新 [編集]

$$\boldsymbol{e}_k = \boldsymbol{z}_k - H_k \hat{\boldsymbol{x}}_{k|k-1} \quad \text{（観測残差、innovation）}$$
$$S_k = R_k + H_k P_{k|k-1} H_k^{\mathrm{T}} \quad \text{（観測残差の共分散）}$$
$$K_k = P_{k|k-1} H_k^{\mathrm{T}} S_k^{-1} \quad \text{（最適 カルマンゲイン）}$$
$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k \boldsymbol{e}_k \quad \text{（更新された状態の推定値）}$$
$$P_{k|k} = (\mathrm{I} - K_k H_k) P_{k|k-1} \quad \text{（更新された誤差の共分散）}$$

カルマンフィルタの数式
カルマンフィルター - Wikipedia

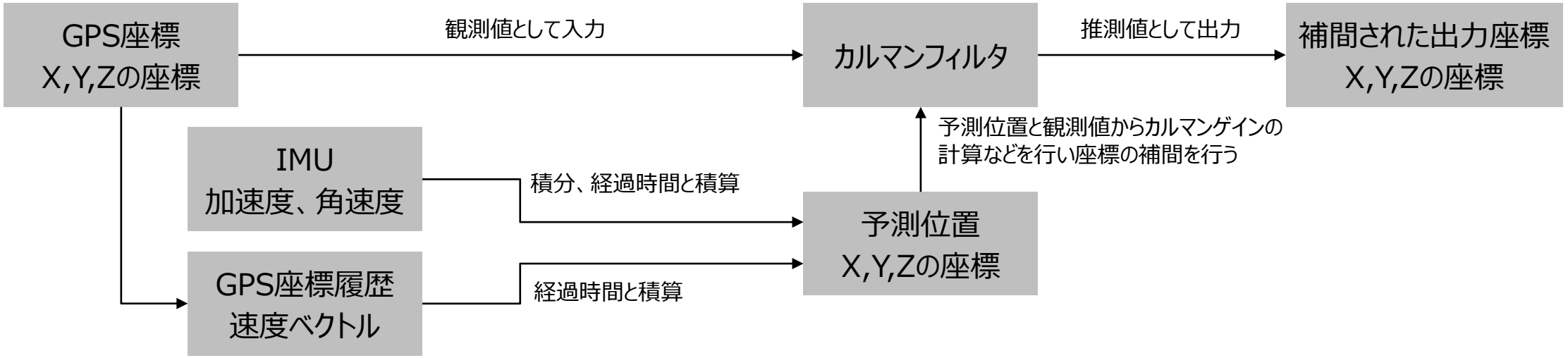
3:初期位置入力プログラム

ⅱ:カルマンフィルタ　センサフュージョンによる座標補間

実証ではカルマンフィルタの入力として、スマートフォンのIMUもしくはADENUのIMUからの加速度角速度の情報を用いる実装と、スマートフォンのGPSの過去の座標履歴から速度を算出し用いる実装を検証した。これらの実装は選択可能な形で初期位置入力プログラム内に組み込まれている。

IMUを用いる実装では、スマートフォンかADENUのIMUから、加速度、角速度を取得する。加速度の積分は速度となり、二階積分で位置となるので、これと角速度の積分を用いて予測位置を計算する。

GPSの過去の座標履歴の実装では、直近の複数のGPS座標を保持しておき、その差分と経過時間から速度ベクトルを計算し、直前の座標に速度ベクトルと経過時間の積を加算することで予測位置を計算する。

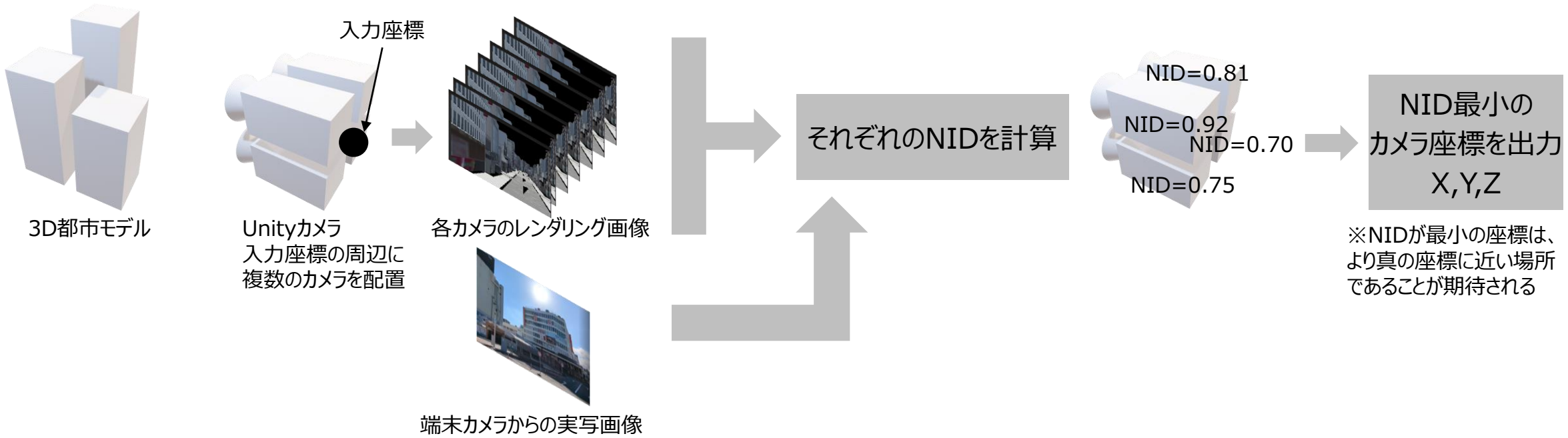


Ⅲ. 実証システム ＞ 5. アルゴリズム
3:初期位置入力プログラム

ⅲ :NID探索　NID計算の高速化と初期位置探索

カルマンフィルタにより補間された座標からさらに精度をあげるために、C*で使用しているNIDの計算を活用する。カルマンフィルタから与えられた座標近傍のNIDを複数計算し、最小のものを探索してその位置をC*への入力に利用する仕組みを実装した。この実装はより真値に近い座標をC*の前段で探索することを意図した、本ユースケースの独自実装である。

本実装はUnityで実装されており、カルマンフィルタから入力された座標の近傍に144個(3×4×12で配置)のカメラを0.5m間隔で配置する。各カメラのレンダリング画像と実写画像とのNIDを計算し、その中で最小のNIDになったカメラの座標を出力する



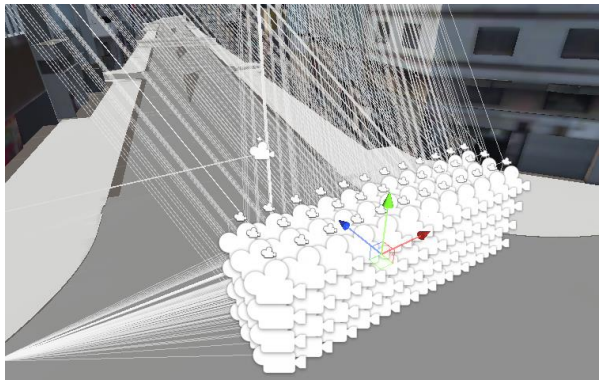
3:初期位置入力プログラム

ⅲ :NID探索　NID計算の高速化と初期位置探索

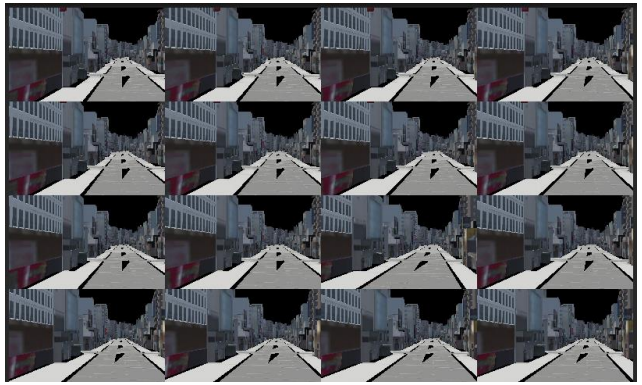
大量の画像を取得してそれぞれのNIDを計算する必要があるため、GPUを用いて計算を並列化するComputeShaderというUnityの機能を活用する実装を行い高速化した。

まず、Unityに複数のカメラを配置しレンダリング結果をまとめて一つのテクスチャに書き出すように設定した。このテクスチャを実写のカメラ画像と共にComputeShaderで実装したプログラムに入力すると、GPUのSharedMemoryに確保した配列に、レンダリング画像と実写画像の輝度の相関ヒストグラムが計算され、Parallel Reductionアルゴリズムを用いて高速化された合計計算ルーチンでNID計算を行う。また、この際各スレッドの競合が起きないように計算順序の工夫と、InterlockedAddを用いた排他ロックを行う。さらに、本機能は大まかな絞り込みを高速で行う必要があるので、C*本体で行っているサブピクセル補間を行わない実装とした。

最終的に各画像と実写カメラ画像のNIDが計算される。この中の最小値になったカメラの座標をC*への最終的な入力値とする実装となっている。



Unity上に実際に大量のカメラを配置したところ



レンダリング結果をテクスチャにまとめ、
一括で計算することで高速化

```
for (int i = 0; i < 8;i++) {
  for (int j = 0;j < 8;j++) {
    float4 ref = Reference[gid.xy * 8 + id.xy * uint2(320, 240) + int2(i,j)];
    float4 cur = Current[gid.xy * 8 + int2(i, j)];
    if (ref.w != 0 || cur.w != 0) {
      float refMono = 0.299 * ref.x + 0.587 * ref.y + 0.114 * ref.z;
      float curMono = 0.299 * cur.x + 0.587 * cur.y + 0.114 * cur.z;
      uint refIdx = uint(refMono * 255) / 16;
      uint curIdx = uint(curMono * 255) / 16;
      hist[(id.x * 4 + id.y) * 16 * 16 + refIdx * 16 + curIdx] += 1;
    }
  }
}
```

NID計算のComputeShaderプログラムの一部抜粋
二画像の輝度の相関ヒストグラムの計算部分

4:Status Display

Status DisplayはUnityで実装した。

Status Displayに必要な情報を各機器から取得した。

- C*の位置推定結果とスマートフォンカメラ映像：

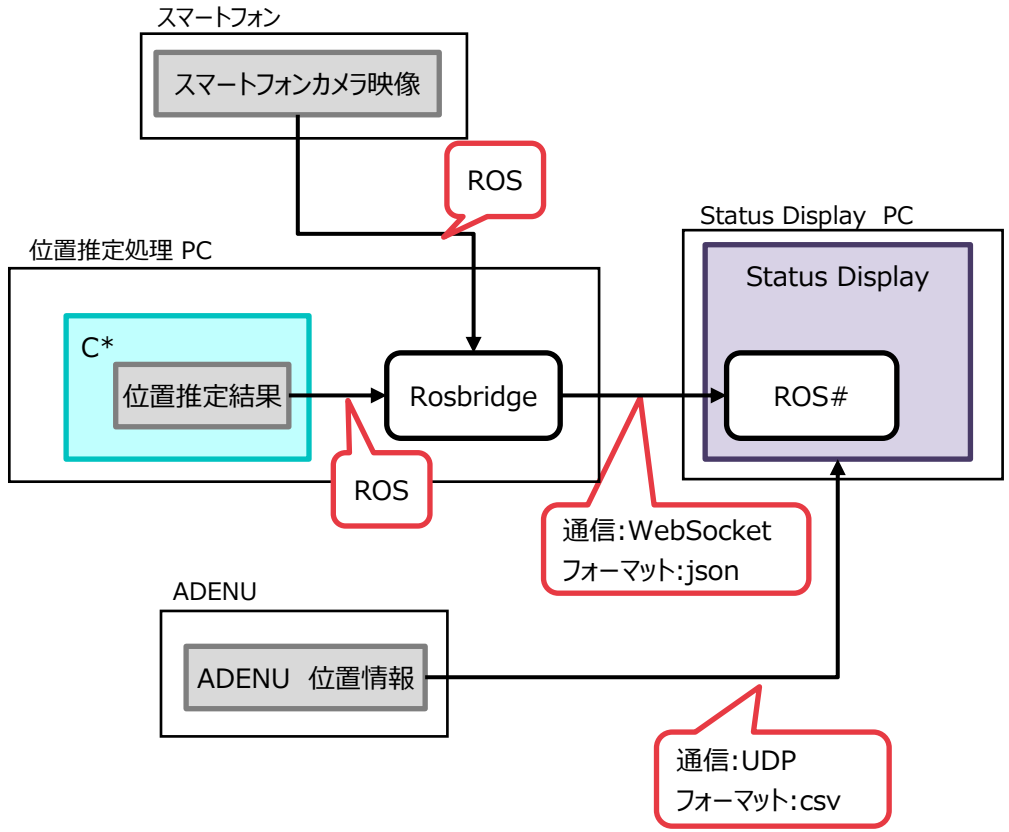
  ROS#ライブラリとRosbridgeを介して取得した。

- ADENUの位置情報：

  位置が記述されたcsvデータをUDP通信で取得した。

Status Displayの各機能の実装内容

機能	実装内容
1. 3Dビュー	C*から受信した位置推定結果と、ADENUから受信した位置情報を、3D都市モデルとともに表示するようにした。
2. ログ	機器の通信状況や、アプリエラーなどのログを表示するようにした。
3. 機器の稼働状況	各機器から一定期間データを受信していなければ、その旨を表示するようにした。
4. 映像	C*から受信した推定位置から3D都市モデルをレンダリングした映像と、スマートフォンカメラ映像を並べて表示するようにした。



各機器からのデータ取得

①活用データ｜3D都市モデル一覧

地物	地物型	属性区分	属性名	内容
建築物LOD2	bldg:Building	空間属性	bldg:lod2Solid	建築物のLOD2の立体
	app:Appearance	空間属性	app:imageURI など	建築物のLOD2テクスチャ
建築物LOD3	bldg:Building	空間属性	bldg:lod3Solid	建築物のLOD3の立体
	app:Appearance	空間属性	app:imageURI など	建築物のLOD3テクスチャ
道路LOD3	tran:Road	空間属性	tran:lod3MultiSruface	道路のLOD3立体
都市設備LOD3	frn:CityFurniture	空間属性	frn:lod3Geometry	都市設備のLOD3立体
	app:Appearance	空間属性	app:imageURI など	都市設備のLOD3テクスチャ
地形LOD2	dem:TINRelief	空間属性	dem:tin	地形のLOD2立体

①活用データ｜その他の活用データ一覧

活用データ	内容	データ形式	出所
スマートフォンGPS	スマートフォンのGPSで取得した現在地座標	地理座標の数値	端末内蔵GPS
スマートフォンIMU	スマートフォンのセンサで取得した加速度角速度	数値	端末内蔵センサ
RTK-GNSS	RTK-GNSSで取得した現在地座標（データ分析用）	NMEA形式	RTK-GNSS受信機
ADENUデータ	ADENUで認識している自己位置の座標など（表示・データ分析用）	CSV、ROSBAG	ADENU
ADENU IMUデータ	ADENUで取得した加速度と角速度	CSV形式データをUDPで受信	ADENU

スマートフォンGPS座標・IMUデータ

スマートフォンのGPSで受信した地理座標をUnityでライブラリ（Native Toolkit）により取得した。Unityの標準の方法では座標がfloat型で得られ精度が足りないため、double型で得られるライブラリを使用した。

取得した緯度経度は、地心直交座標系（ECEF）に変換し、さらに検証地域内の原点を緯度、経度、楕円体高で指定し回転処理をしてENU座標系に変換した。

スマートフォンのIMUデータは、UnityのInputクラスの加速度、ジャイロ値取得機能により、取得、利用した。

Native Toolkit | Integration | Unity Asset Store

RTK-GNSS座標

分析の際、実際の位置を精度よく取得してVPSの精度評価をするために高精度のRTK-GNSS受信機で地理座標を記録した。

GNSS受信機は、Droggerの2周波GNSS受信機「RWP」パッケージを使用した。

RTKのための補正情報として、ソフトバンク社のIchimill高精度測位サービスを利用した。

Bluetoothでスマートフォンに接続し、座標の記録やIchimillの補正情報（ntrip）の送信を行った。

使用したRTK-GNSS受信機

ADENUデータ/ADENU IMUデータ

自動運転システムADENUから、リアルタイムにUDP通信を介して時刻、座標、緯度、経度、速度、姿勢、加速度、角速度のデータを受信し、Status Display、初期位置入力プログラムで利用した。

データフォーマットは、CSV文字列で、以下のサンプルのようなデータが秒間10数回ほどADENUから送信される。

特に本実証システムを稼働する自動運転車両に搭載されているMovella Technologies社（旧Xsens Technologies社）のMTi-100シリーズIMUからの情報（加速度・角速度）は、初期位置入力プログラムのカルマンフィルタの入力値として利用した。

また、別途rosbag形式で記録したADENUの稼働情報を分析のために受領した。

自動運転車に搭載されているIMU

Time(sec),Time(nsec),X,Y,Z,Lat,Lon,Speed(m/s),Pose(w),Pose(x),Pose(y),Pose(z),acc_x(m/s2)acc_x(m/s2),acc_y(m/s2),acc_z(m/s2),roll(rad/s),pitch(rad/s),yaw(rad/s)
1671591521,615161356,-99524.011,32777.805,9.203,35.1023662,138.8595421,0,0.88023,0,0,-0.47454,1.15651,-0.09351,9.73925,0.00066,0.00164,0.0017
1671591521,715155609,-99524.011,32777.805,9.203,35.1023662,138.8595421,0,0.88023,0,0,-0.47454,1.1631,-0.09721,9.712,0.00501,-0.00391,-0.00249
1671591521,815136096,-99524.011,32777.805,9.203,35.1023662,138.8595421,0,0.88023,0,0,-0.47454,1.15462,-0.09643,9.729,0.00207,0.00359,0.00271

ADENUから送られてくるデータのサンプル

②データ処理｜一覧

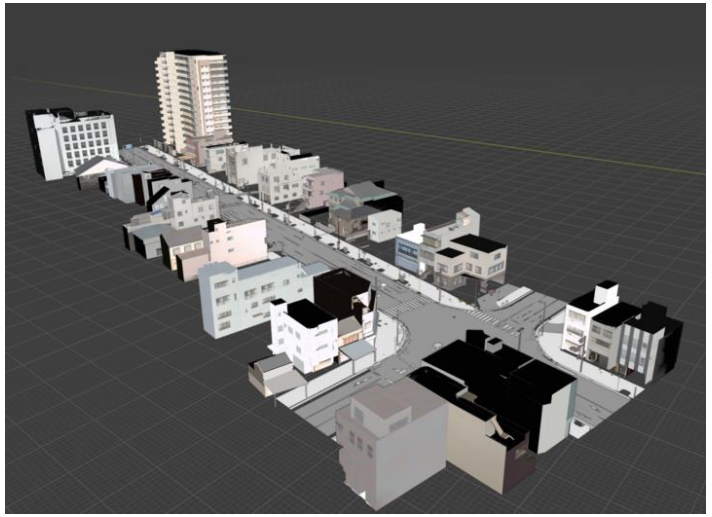
システムに入力するデータ（データ形式）	用途	処理内容	データ処理ソフトウェア	活用データ（データ形式）
3Dモデルデータ（OBJ形式）	C*用のマップデータ	• テクスチャ付き建築物LOD3モデルのCityGML形式から座標変換とOBJ形式への変換	Unity＋独自ソフトウェア	3D都市モデル（CityGML形式）
3Dモデルデータ（OBJ形式）	C*用のマップデータ	• テクスチャ付き建築物LOD2モデルのCityGML形式から座標変換とOBJ形式への変換	Unity＋独自ソフトウェア	3D都市モデル（CityGML形式）

3Dモデルデータの生成

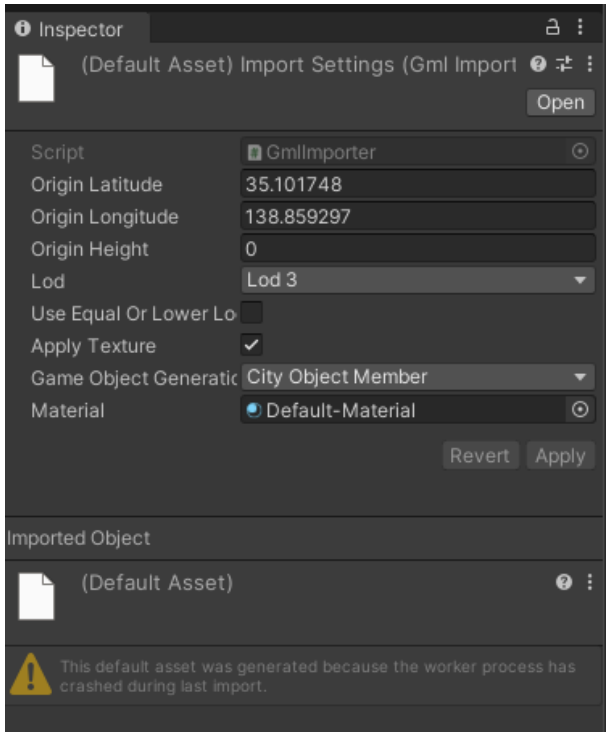
3D都市モデルのCityGML形式データは、そのままではレンダリングできないため、ホロラボ社で独自に開発したインポートプログラムを利用してUnity上に読み込んだ。この際、JGD2011の経緯度をECEF→ENU座標系に変換し、Unityの座標系と合わせた。

また、テクスチャも読み込み、適切に3Dモデルに適用した。

次に、「3D都市モデルの3Dデータ軽量化」で記載したように、Simplygonプラグインを用いて軽量化したものを、再度OBJ形式で出力し、C*のオリジナルのRendererや、独自Rendererに読み込ませた。



変換し表示した3D都市モデル



CityGMLインポーターの画面

③出力データ｜一覧

出力データ	内容	データ形式
推定座標	C*が自己位置推定した結果の座標	ROS Message形式
データ分析結果	C*の位置精度の分析結果、各座標のプロット	CSV、画像ファイル
各プログラムのログ	C*、初期位置入力プログラム、RTK-GNSS受信座標などのデータ	CSV、テキストファイル

C*推定座標の出力

C*が自己位置推定した座標はROSのメッセージ形式で出力される。この出力はファイルに記録されると同時にネットワーク経由でStatus Displayに表示される。

ファイルに記録された座標は後段のデータ処理に利用する。

Status Displayでの表示

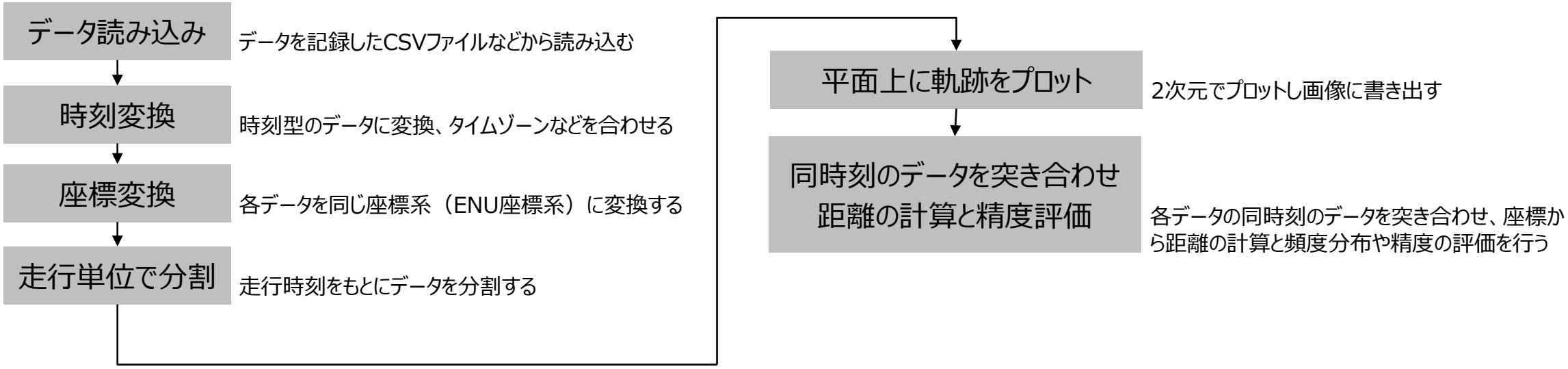
```
transforms:
  -
    header:
      seq: 1
      stamp:
        secs: 1673936574
        nsecs: 139109135
      frame_id: "world"
    child_frame_id: "mynteyed"
    transform:
      translation:
        x: -0.20457549911373066
        y: -1.479062993569643
        z: 0.2117887067670789
      rotation:
        x: 0.0988035102613292
        y: 0.7126039018847931
        z: -0.690085701825234
        w: -0.07883698063080606
---
```

ROS Message 形式での座標出力例

データ分析結果

実証実施後のデータ分析は主にGoogle ColaboratoryのPython環境で行った。NumpyやPandasなど大きなデータを高速に扱うことのできるツールを活用している。

データ処理は下図のようなフローとなっている。



システムテスト結果

試験項目	確認内容	結果
独自Rendererへの3Dモデル読み込みと動作	独自Rendererに変換した3Dモデルを読み込み、表示でき、ROSメッセージの送信ができる	OK
各コンポーネントの接続	初期位置入力プログラム、C*、独自Renderer、Status Displayの各コンポーネントが正しく接続され、データを送受信できる	OK
ログの記録	各コンポーネントでログが記録されている	OK
初期位置入力プログラムの動作	GPS取得、各種計算処理が行われ、初期位置が約5秒ごとにROSメッセージで送信される	OK
C*の動作	入力された初期位置と画像群から自己位置推定ができる	OK

Ⅰ．実証概要

Ⅱ．実証技術の概要

Ⅲ．実証システム

Ⅳ．実証技術の検証

Ⅴ．成果と課題

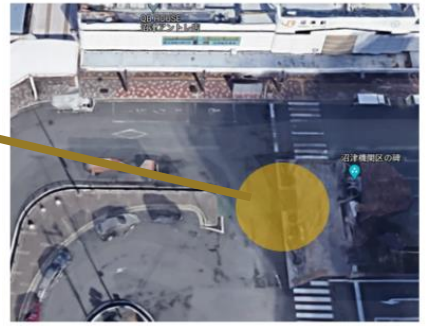
Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
①検証内容｜実証詳細

目的	3D都市モデルを活用したVPSによる自動運転車両の自動運転モード下での自己位置推定
実施期間	2023年1月16日（月）～17日（火）
実施場所	静岡県　沼津市（沼津駅前～沼津港）
主な参加者	国土交通省都市局、三菱総合研究所、凸版印刷、ホロラボ、東急
実施内容	1.　3D都市モデルを3次元モデルとして活用したVPSによる自動運転車両、自動運転下での自己位置推定 2.　Status DisplayによるVPS自己位置推定結果のリアルタイム表示、自動運転車両自己位置の重畳表示
気象条件	•　1月16日（12時～16時）：降水量0.0mm、平均気温10.5度、天候雨→曇り、日照時間0h •　1月17日（12時～16時）：降水量0.0mm、平均気温9.2度、天候曇り→晴れ、日照時間2.6h
評価方法	1.　自己位置推定結果とRTK-GNSSによる座標位置比較による評価（実施内容1.より） 2.　自己位置推定結果とADENU座標位置比較による評価（実施内容1.より） 3.　実証システムによる自己位置推定結果の定性的な確認による評価（実施内容2.より）

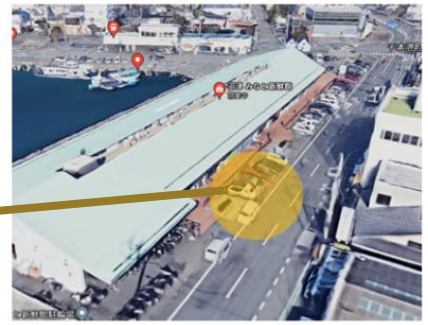
Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
①検証内容｜運行経路・運行スケジュール

運行経路

沼津駅～沼津港（約2.2ｋｍ）



沼津駅



沼津港

運行スケジュール

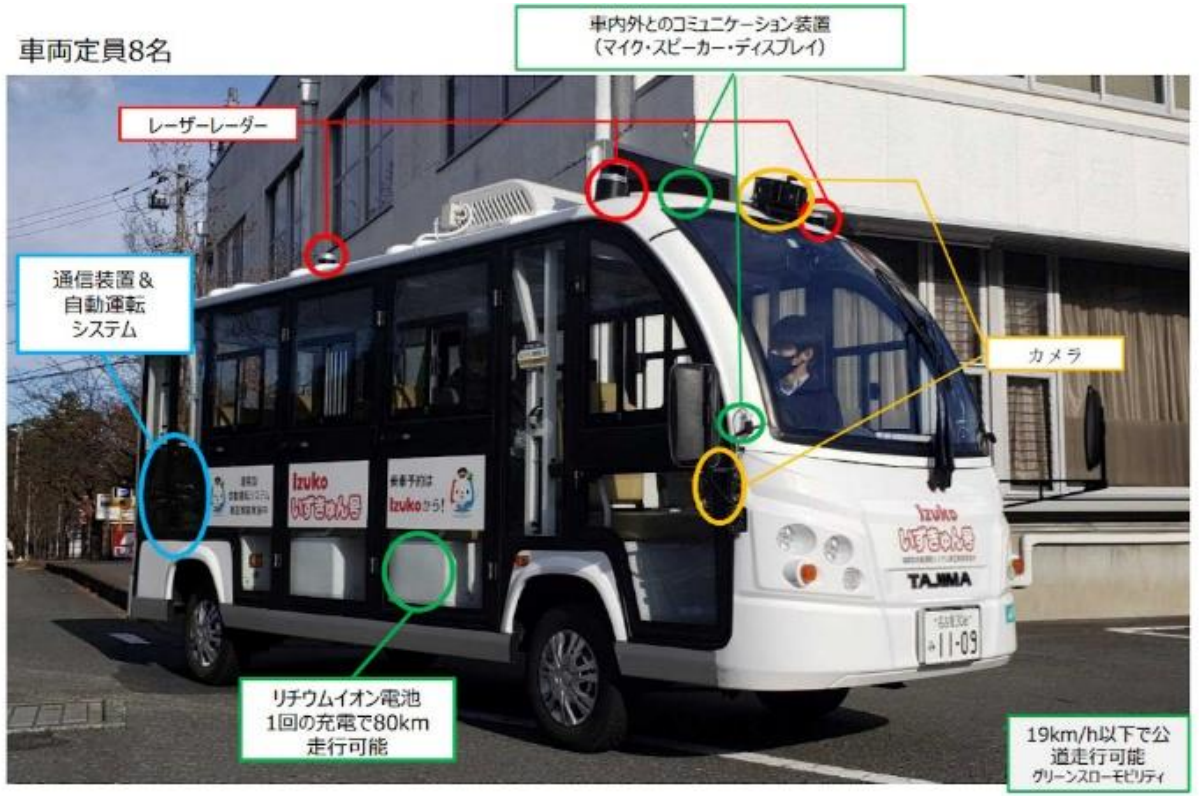
	1月16日(月)		1月17日(火)	
	沼津駅→沼津港	沼津港→沼津駅	沼津駅→沼津港	沼津港→沼津駅
08:00				
09:00	・機材セッティング ・ソフトの動作確認 (9：00～)		・機材セッティング	
10:00			Day2-1-1	Day2-1-2
11:00			Day2-2-1	Day2-2-2
12:00	昼食		昼食	
13:00	Day1-1-1	Day1-1-2	Day2-3-1	Day2-3-2
14:00	Day1-2-1	Day1-2-2	Day2-4-1	Day2-4-2
15:00	Day1-3-1	Day1-3-2	Day2-5-1	Day2-5-2
16:00	・機材撤収		・機材撤収	
17:00				
18:00				

運行経路を時速19km/h以下で自動走行を実施。
往路と復路は運行スケジュールに基づき走行を実施。

Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
①検証内容｜使用機材（自動走行運転車両）

自動走行運転車両

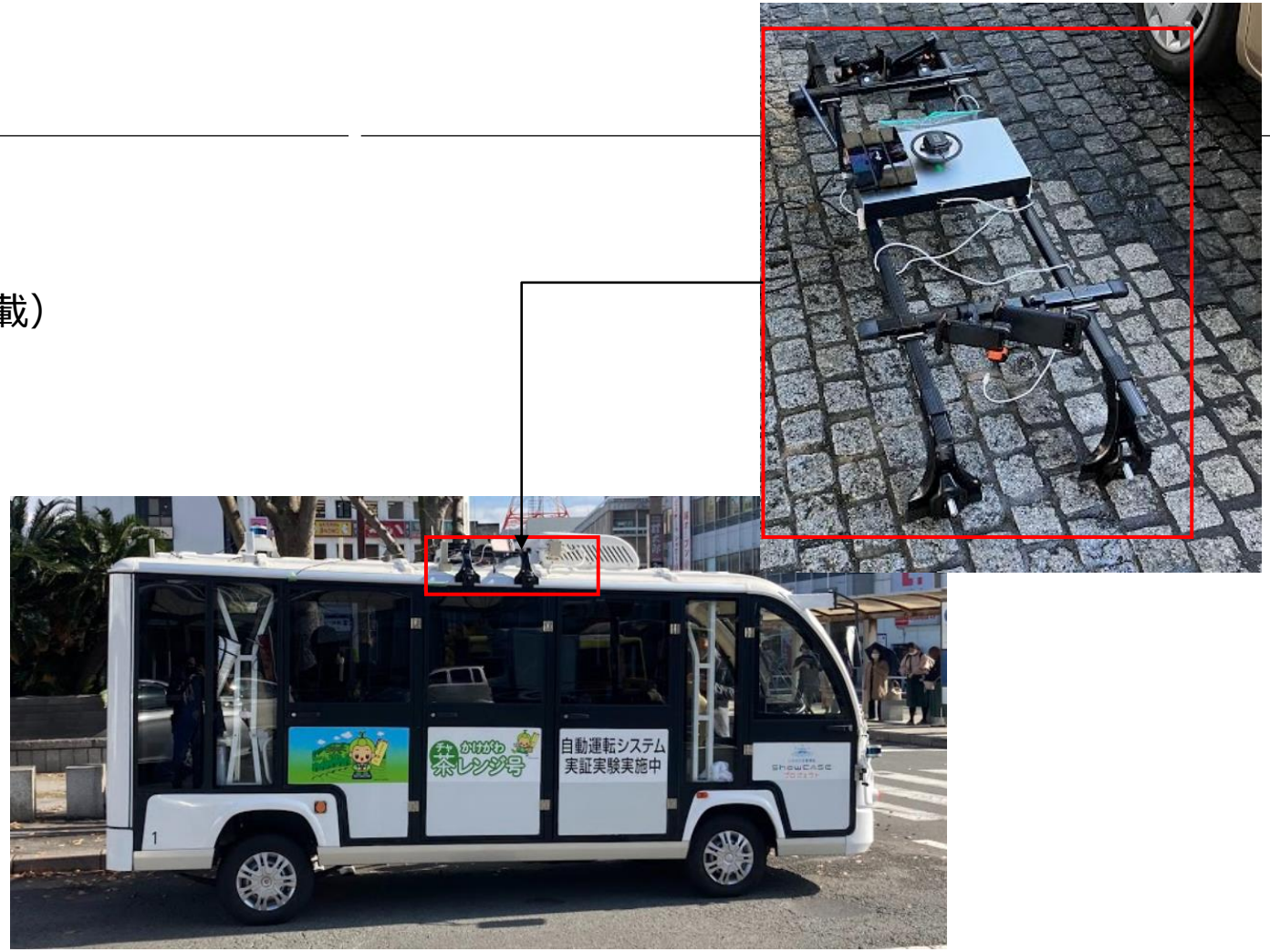
車両仕様
(ア)使用車両：TAJIMA-NAO-8J
(イ)車両保有・提供・運行者：東急株式会社
(ウ)乗車定員：8人
(エ)最高速度：19km/h
(オ)構造：
・全長4.9m、全幅1.5m、全高2.3m
・電動（EV）車
・車両重量約1650kg
・前方、後方どちらへも進行が可能
・自動走行運転レベル2GPS座標補間



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
①検証内容｜使用機材（VPS関連）

使用車両

VPS関連機材
- 車載治具
- スマートフォン2台（車載治具に搭載）
- GNSS（車載治具に搭載）
- 位置推定処理PC2台
- ステータス表示PC
- LAN HUB



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
①検証内容｜検証項目

3D都市モデルを活用したVPSによる自動運転車両の自動運転モード下での自己位置推定を実施した。

大項目	小項目	検証内容
VPSシステム検証	• 自動運転車両による自動運転走行下でのVPSシステムによる自己位置推定とRTK-GNSSとの比較検証。 • Status DisplayによるVPS初期位置ローカライズ、継続トラッキングの定性的検証。	

	VPS初期位置	• 各種VPS入力初期位置補間による検証
		  • 慣性計測装置（スマートフォン　IMU、ADENU　IMU）によるVPS入力初期位置の補間による結果検証
		  • スマートフォンGPS座標間計算補間によるVPS入力初期位置の補間による結果検証
		  • NID探索によるVPS入力初期位置補間による結果検証

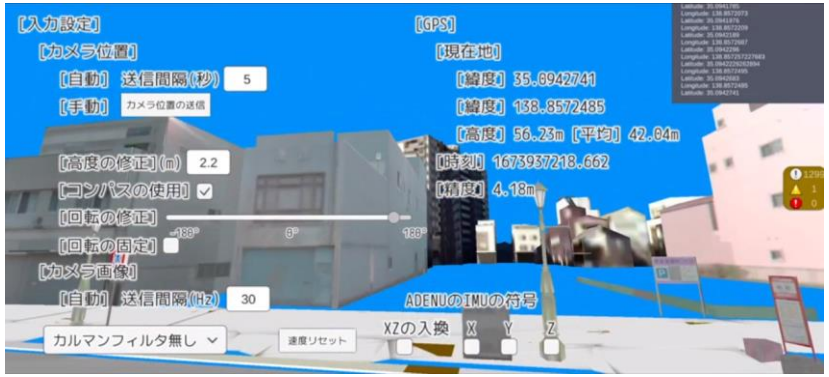
	VPSローカライズ	• ローカライズ精度の検証
	継続トラッキング	• 継続的なトラッキング精度の検証

Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜現地実証の状況

自動運転車両・自動走行の様子




実証中VPSシステム・Status Displayの様子



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜検証実施スケジュール

検証実施スケジュール

日付	実施時刻	実証名	位置処理推定PC種別	スマホ端末種別	使用モデル	カメラ向き計画	C*Param	カメラ入力	高さ・方位計画	GPS座標履歴補完	IMU計画	NID計画	光源計画	天気	天気（実際）
1月16日	14:00	Day1-1-1	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	なし	なし	なし	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	なし	なし	なし	Unlit	オリジナル(Unlit)	曇り
	14:20	Day1-1-2	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	なし	スマホIMU	なし	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	なし	スマホIMU	なし	Unlit	オリジナル(Unlit)	曇り
	14:50	Day1-2-1	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	なし	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	なし	Unlit	オリジナル(Unlit)	曇り
	15:10	Day1-2-2	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	曇り
	15:35	Day1-3-1	①	A	一部非表示モデル	左斜め前	30, 0.2	5秒	固定	あり	なし	なし	Unlit	オリジナル(Unlit)	曇り
			②	B	一部非表示モデル	右斜め前	30, 0.2	5秒	固定	あり	なし	なし	Unlit	オリジナル(Unlit)	曇り
	未実施	Day1-3-2	①	A											
			②	B											

日付	実施時刻	実証名	位置処理推定PC種別	スマホ端末種別	使用モデル	カメラ向き計画	C*Param	カメラ入力	高さ・方位計画	GPS座標履歴補完	IMU計画	NID計画	光源計画	天気	天気（実際）
1月17日	10:53	Day2-1-1	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	なし	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	ADENU IMU	なし	Unlit	オリジナル(Unlit)	曇り
	11:30	Day2-1-2	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	要確認	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	要確認	Unlit	オリジナル(Unlit)	曇り
	12:06	Day2-2-1	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	要確認	Unlit	オリジナル(Unlit)	曇り
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	要確認	Unlit	オリジナル(Unlit)	曇り
	未実施	Day2-2-2	①	A											
			②	B											
	14:20	Day2-3-1	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
	14:37	Day2-3-2	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
	15:25	Day2-4-1	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
	15:50	Day2-4-2	①	A	LOD3+LOD2（軽量化）	左斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
			②	B	LOD3+LOD2（軽量化）	右斜め前	30, 0.2	5秒	固定	あり	なし	あり	Unlit	オリジナル(Unlit)	晴れ
	未実施	Day2-5-1	①	A											
			②	B											
		Day2-5-2	①	A											
			②	B											

Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜自動運転車両による自動運転走行下でのVPSシステムによる自己位置推定

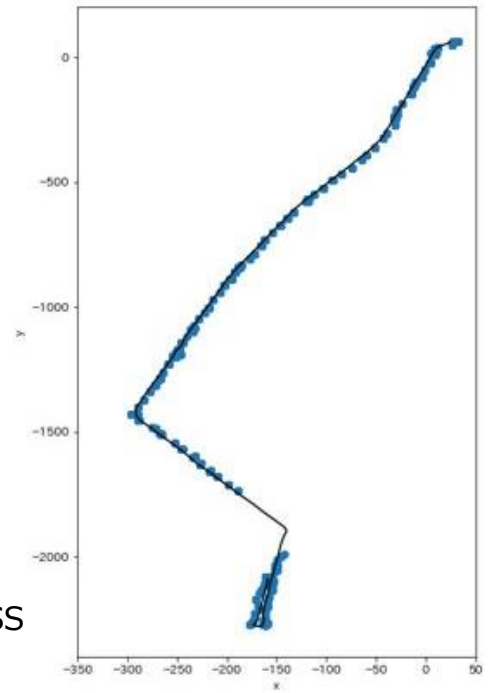
検証の結果得られた自己位置推定結果について、RTK-GNSSから取得した座標とC*を利用した自己位置推定プログラムによる座標とをグラフにプロットし（左の2図）、それぞれの座標がどの程度重なるかを確認した（本検証ではRTK-GNSS座標を真値と仮定）。

この結果、RTK-GNSSの軌跡に近い座標で自己位置推定されていることが確認できた。

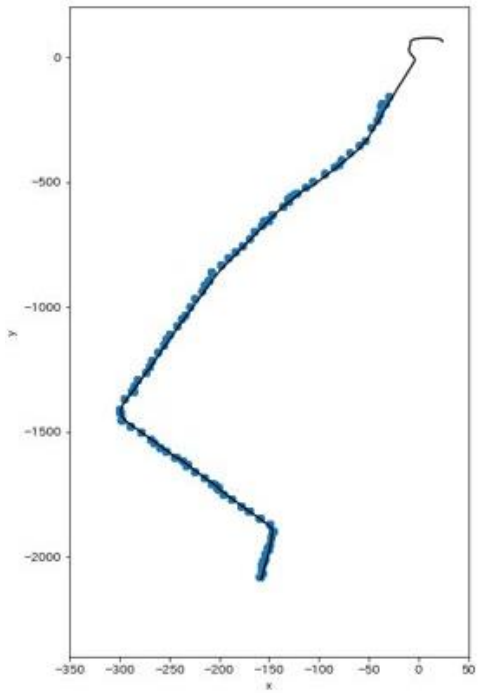
また、昨年度の結果（右図）と比べても、より近い点を推定できている。

実証2日目（1月17日）3回目（DAY2-3-1、DAY2-3-２)データにおける
RTK-GNSS取得座標、C*を利用した自己位置推定プログラム座標の比較と、昨年度の結果との比較

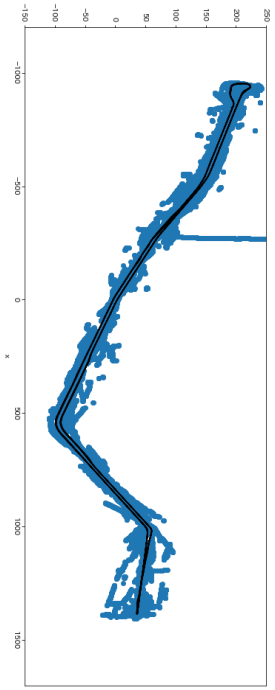
DAY2-3-1（沼津駅～沼津港）
端末A（進行方向右側）



DAY2-3-2（沼津港～沼津駅）
端末A（進行方向右側）



2021年度ベストスコアデータ（参考）



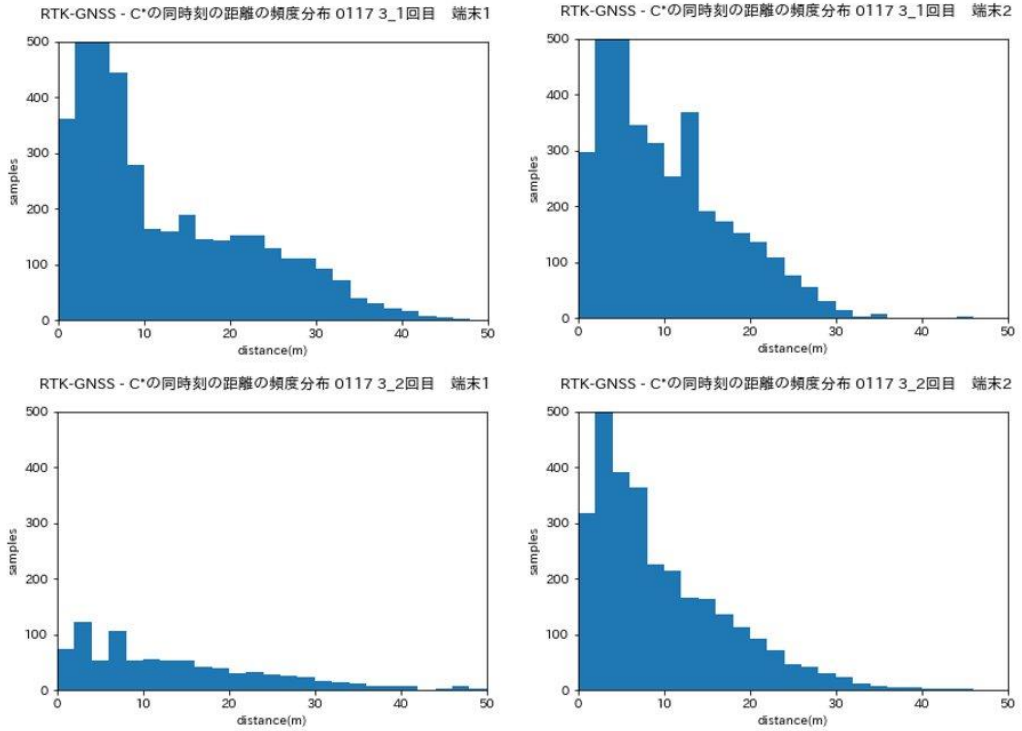
RTK-GNSS
C*

Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜自動運転車両による自動運転走行下でのVPSシステムによる自己位置推定

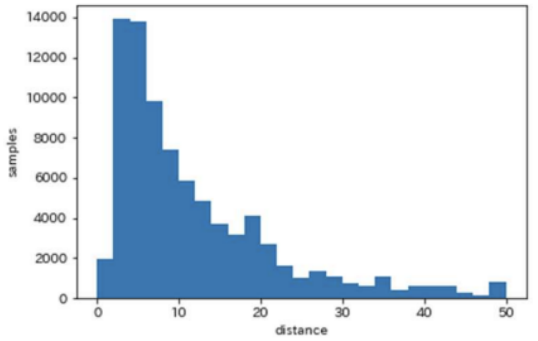
検証結果について、今回の実証で真値とみなすRTK-GNSS座標とC*を利用した自己位置推定プログラムによる自己位置推定座標との間にどの程度の差異が生じているかを、座標間の距離の頻度分布により検証した。
その結果2m～6mあたりにピークがあり、また0m~2mにも多くのサンプルが分布していることが見え、さらに昨年度の結果と比較すると、より真値に近い位置に多くのサンプルが分布する結果となっている。

実証2日目（1月17日） 4回目（DAY2-4-1)データにおけるRTK-GNSS取得座標、C*利用自己位置推定座標の比較

- 各回のRTK-GNSSによる座標とC*を利用した自己位置推定プログラムによる自己位置推定による座標間の距離の頻度分布
- 実証2日目（1月17日）の3回目（実証名DAY2-3-1、DAY2-3-２）データを抽出
- 縦軸＝頻度（samples数）
- 横軸＝RTK-GNSSとC*の同時刻での座標距離差分（m）

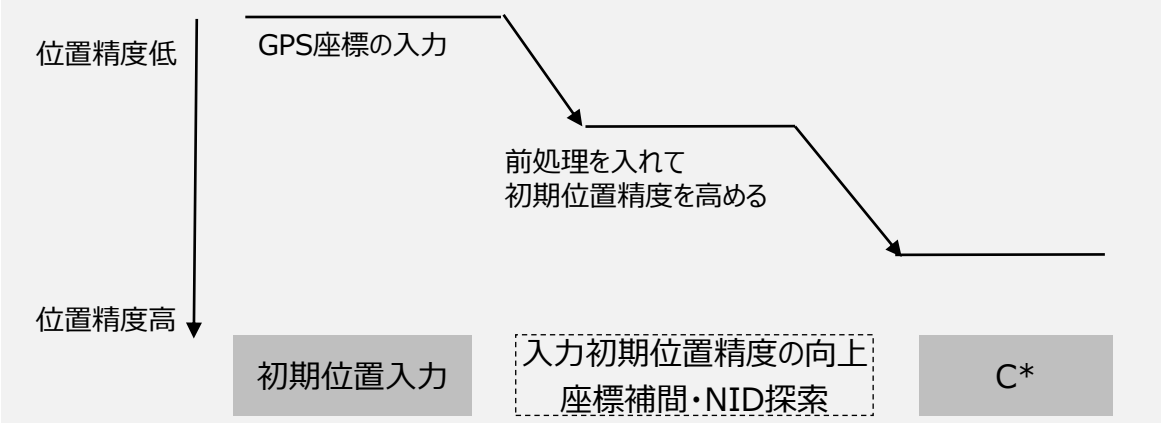


2021年度ベストスコアデータ（参考）



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜システム各段階での自己位置推定への寄与の検証

ここまでの結果で、システム全体では少なくとも2021年度の検証結果より精度の高い自己位置推定ができていることが確認できた。
しかし、今回はVPS単体ではなく、複数の機能を組み合わせており、初期位置入力で用いるスマートフォンGPS、C*による位置推定やその前段で行っているカルマンフィルタ、NID探索のそれぞれの段階の精度への寄与や挙動を確認する必要がある。
そのため、各段階での座標のプロット、RTK-GNSSとの差分集計等を行い、各機能の寄与度とその挙動を次項以降で検証した。

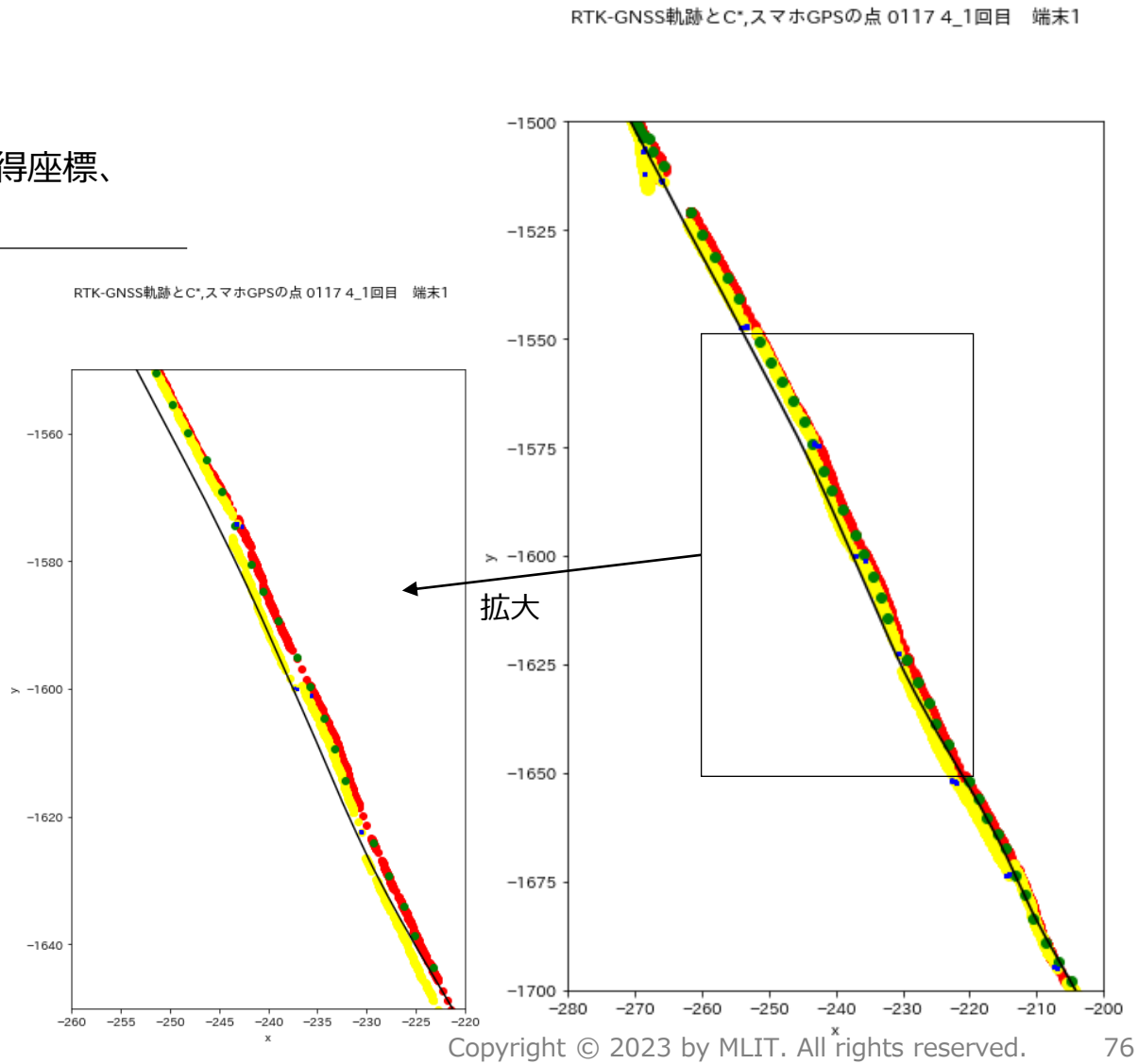


本システムの機能組み合わせと精度向上の模式図

Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜システム各段階での自己位置推定への寄与の検証

実証2日目（1月17日） 4回目（DAY2-4-1)データにおけるRTK-GNSS取得座標、
各段階での座標、VPS自己位置推定座標の比較

- 各回のRTK-GNSS、ⅰ:GPS座標/DEM、ⅱ:カルマンフィルタ、ⅲ:NID探索、VPS自己位置推定、RTK-GNSSによる座標を比較。
- 実証2日目（1月17日）の4回目（実証名DAY2-4-1）データを抽出。
  - RTK-GNSSによる座標＝黒線
  - ⅰ:GPS座標/DEMによる座標＝緑点
  - ⅱ:カルマンフィルタによる座標＝赤点
  - ⅲ:NID探索による座標＝黄点
  - C*を利用した自己位置推定プログラムによる自己位置推定座標＝青点
- ⅰ:GPS座標/DEMによる座標の間隔が広い状態からⅱ:カルマンフィルタで連続的な座標に補間され、さらにⅲ:NID探索でRTK-GNSSに近い位置になり、VPSによる自己位置推定が行われている様子が見られる。

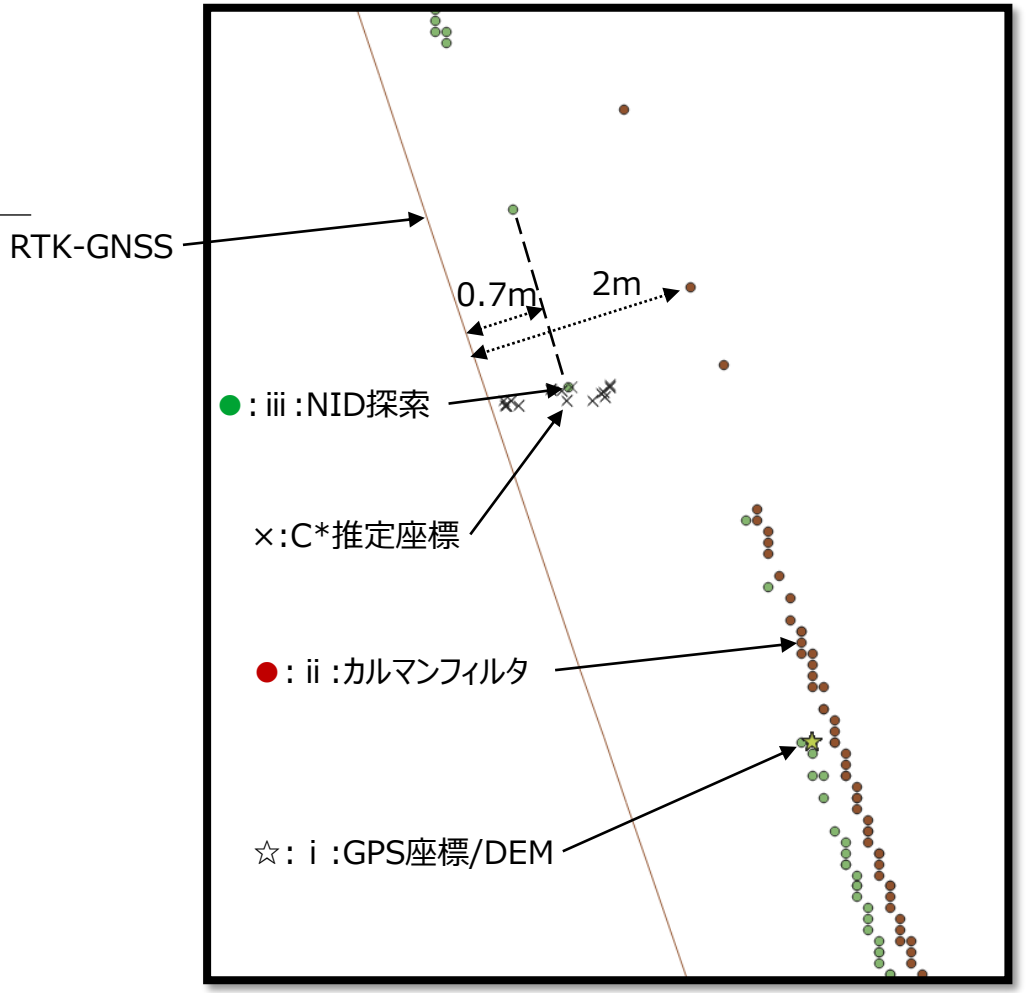


Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜システム各段階での自己位置推定への寄与の検証

実証2日目（1月17日） 4回目（DAY2-4-1)データにおけるRTK-GNSS取得座標、
各段階での座標、VPS自己位置推定座標の比較

- 前項より、さらに詳細に見るため、QGISでデータをプロットしたもの
- 各凡例はグラフの説明参照
- 前項同様に実証2日目（1月17日）の4回目（実証名DAY2-4-1）データを抽出。
- ⅰ:GPS座標/DEMとⅱ:カルマンフィルタがRTK-GNSSに対して2m離れているのに対して、ⅲ:NID探索により0.7mになり、さらにC*がより精度の高い座標を探索している挙動が見える。
- 最終的に、C*とRTK-GNSSの最近値は0.13mになっている。
- 実際に、段階を追って精度が上がり、C*による自己位置推定につながっていることが示せた。

QGISによる初期位置入力→C*の各段階詳細プロット



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜システム各段階での自己位置推定への寄与の検証

前項では、成功している箇所を詳細に確認したが、全体ではどうかを統計的に検証する。以下はⅰ:GPS座標/DEMとⅱ:カルマンフィルタ・ⅲ:NID探索それぞれの座標の距離の頻度分布を示したものである。

実証2日目（1月17日）の4回目（DAY2-4-1)データにおける各段階での座標の比較

- 実証2日目（1月17日）の4回目（実証名DAY2-4-2、DAY2-4-２)データを抽出。
- 図1
  - 縦軸＝頻度（samples数）
  - 横軸＝ ⅰ:GPS座標/DEMとⅱ:カルマンフィルタの座標間の距離（m）
  - 前項のプロットより、 ⅱ:カルマンフィルタの座標はⅰ:GPS座標/DEMから連続的に離れる挙動を示す。距離が離れるにしたがって減少するヒストグラムは全般的に同様の挙動となっていることを表している。
- 図2
  - 縦軸＝頻度（samples数）
  - 横軸＝ ⅰ:GPS座標/DEMとⅲ:NID探索の座標間の距離（m）
  - 前項のプロットより、 ⅲ:NID探索の座標はⅰ:GPS座標/DEMから一定距離をあけて分布すると考えられる。ヒストグラムは、1.5~2mにピークがあり、数mの範囲で探索が行われていると考えられる。

図1

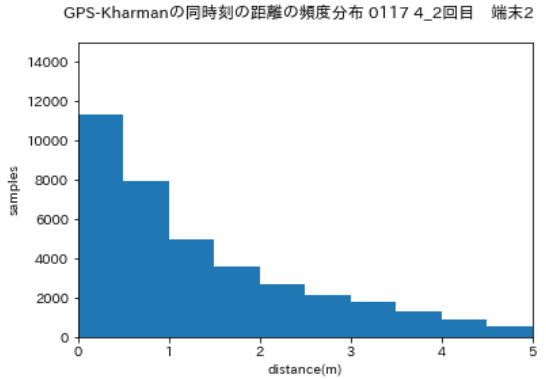
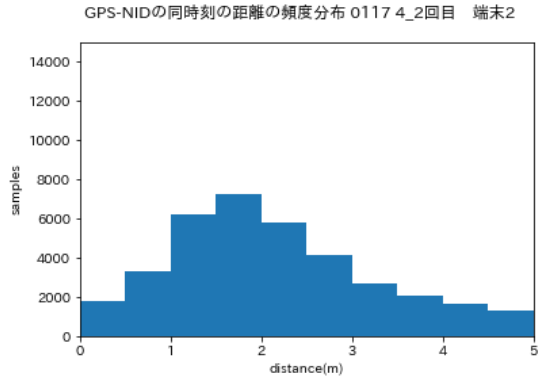


図2



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜システム各段階での自己位置推定への寄与の検証

一方で、各段階で得られた座標が目的とするRTK-GNSSに近くなっているかを検証する。以下はRTK-GNSSと各段階の座標の距離のヒストグラムである。

実証2日目（1月17日）4回目（DAY2-4-1)同時刻データにおけるRTK-GNSS取得座標、各段階での座標の比較

- 実証2日目（1月17日）の4回目（実証名DAY2-4-2)データを抽出。
- 図3
  - 縦軸＝頻度（samples数）
  - 横軸＝RTK-GNSSとⅰ:GPS座標/DEMの座標間の距離（m）
  - RTK-GNSSとⅰ:GPS座標/DEMのピークは2.5~3mとなっており、このピークが各段階でどう移行するかに注目する。
- 図4
  - 縦軸＝頻度（samples数）
  - 横軸＝RTK-GNSSとⅱ:カルマンフィルタの座標間の距離（m）
  - RTK-GNSSとⅱ:カルマンフィルタの座標のピークは2~2.5mであり、RTK-GNSSに近づいている。これは全般的に補間でより良い座標が計算されているといえる。
- 図5
  - 縦軸＝頻度（samples数）
  - 横軸＝RTK-GNSSとⅲ:NID探索の座標間の距離（m）
  - RTK-GNSSとⅲ:NID探索の座標のピークは図4同様2~2.5mであるが、その分布が狭まりピークが高くなっている。これは、 ⅱ:カルマンフィルタと比べてより多くの点でより良い座標値が得られているといえる。

図3

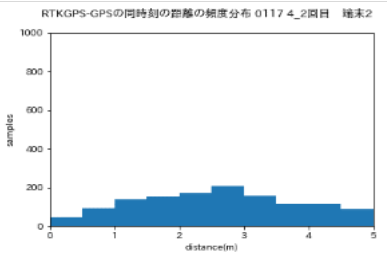


図4

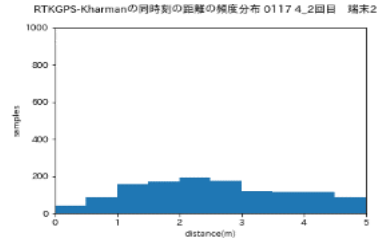
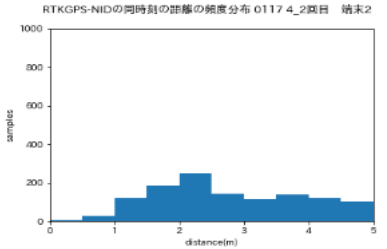


図5



Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜システム各段階での自己位置推定への寄与の検証

さらに、同時刻のRTK-GNSSの座標からの距離の統計値を計算することで前項の結果を補足した。
平均値が小さく、標準偏差が小さくなることが、よりよい精度となっていることを踏まえると、d1（端末1）の方ではⅰ:GPS座標/DEM＜ⅱ:カルマンフィルタ＜ⅲ:NID探索＜初期位置ローカライズ座標の順に精度がよくなっている傾向が見てとれる。一方でd2（端末2）の方では逆転しているところもあり、現状では検証方法に課題が残る。

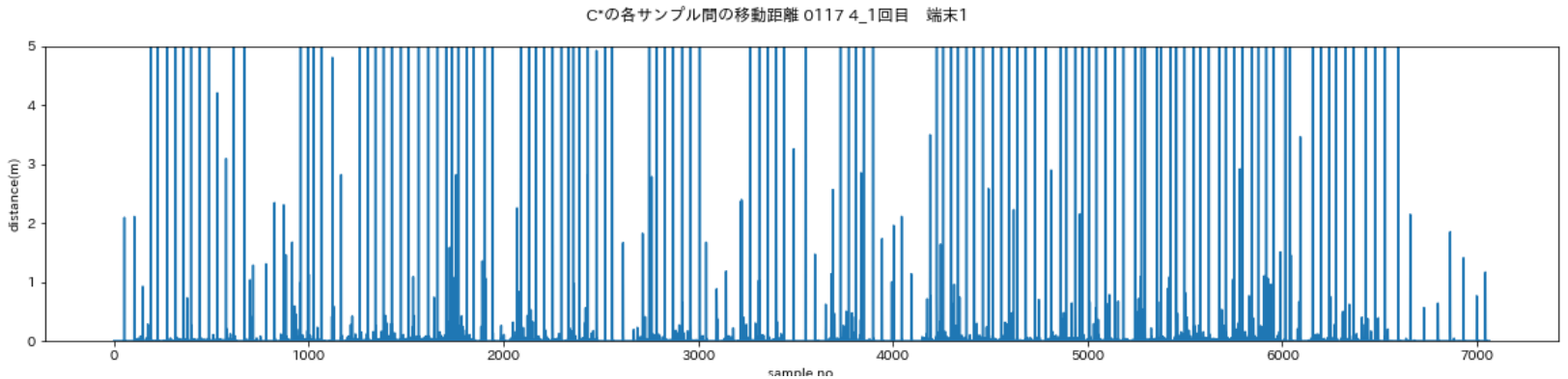
各段階での座標計算結果の統計

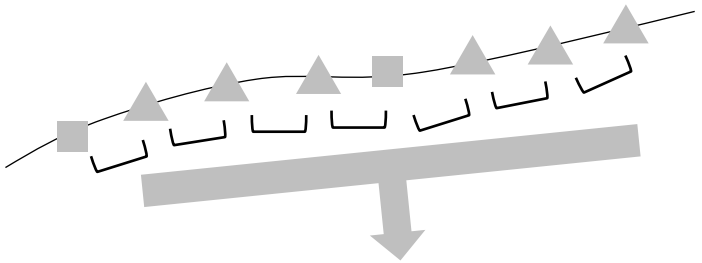
	ⅰ:GPS座標/DEM			ⅱ:カルマンフィルタ			ⅲ:NID探索			初期位置ローカライズ座標		
	平均	標準偏差	中央値	平均	標準偏差	中央値	平均	標準偏差	中央値	平均	標準偏差	中央値
3-1 d1	5.985014	3.308819	5.527408	5.693624	2.877413	5.553503	5.258590	2.472942	5.129856	5.221997	2.679027	5.216944
3-2 d1	5.871838	2.887265	5.734475	5.462620	2.518754	5.213191	5.313374	2.264871	5.177097	5.056870	2.541985	4.556960
4-1 d1	6.139308	3.100145	6.063271	6.068350	2.762919	6.141609	5.782633	2.486233	6.068102	5.908078	2.454739	6.316253
4-2 d1	5.273987	2.629943	4.616576	4.832465	2.534162	4.185533	4.855914	2.271428	4.475325	4.732642	2.249423	4.270153
3-1 d2	3.112367	1.873474	2.719741	3.391407	1.792896	2.931569	3.917129	1.942299	3.558813	4.428137	2.181135	4.029121
3-2 d2	2.738751	1.719861	2.400928	3.535214	1.795273	3.332391	3.923107	2.002883	3.470973	4.216306	2.099837	3.801031
4-1 d2	3.034225	2.465794	2.513118	3.400103	2.092743	2.936514	3.590005	2.039837	3.297732	4.180958	2.186581	3.818329
4-2 d2	3.611862	2.445497	3.021960	3.543334	2.293163	2.937286	3.881447	2.259395	3.320449			

Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜継続的なトラッキングの検証

ここまでで、初期位置入力の仕組みの効果が検証できたが、実証時のStatus Displayで観察した挙動では、初期位置入力時の位置推定以降の、継続的トラッキングは効果的に動作していない様子であった。ここではC*の継続的なトラッキング動作を検証する。まずC*への初期位置送信は5秒間隔で行っている。C*が出力する座標を前後で差分をとって時系列でプロットしたものが以下のグラフである。（C*(t)-C*(t-1)を縦軸とし、横軸tでプロットしたもの）
グラフでは、ほぼ5秒間隔で大きな値が発生しており、初期位置入力時に大きく位置が動く様子がわかる。
継続的なトラッキングが正常に動作していれば、このグラフは小さい値で平坦で滑らかな直線に近くなると想定されることから、C*の継続的なトラッキングが成功していないと言える。

C*の前後の座標の距離の図



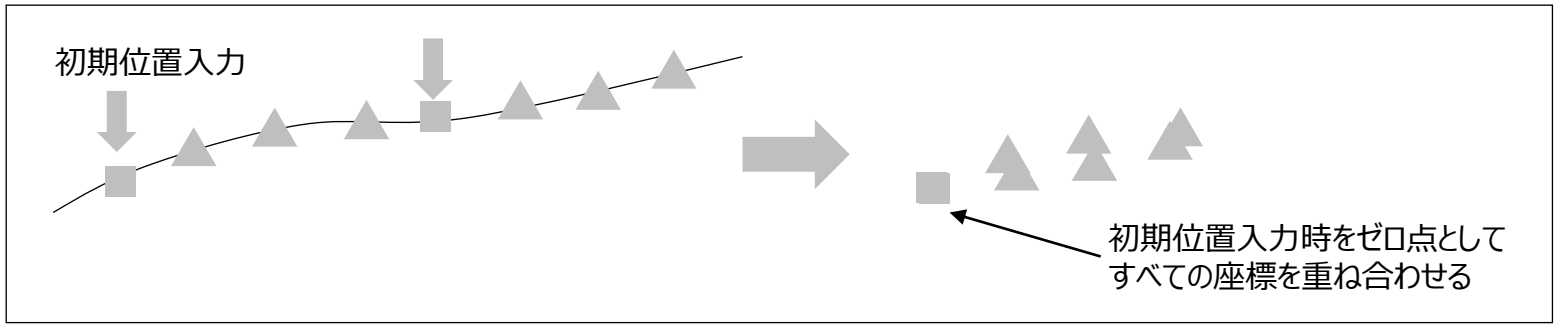


この各点の座標の差分の距離をプロット

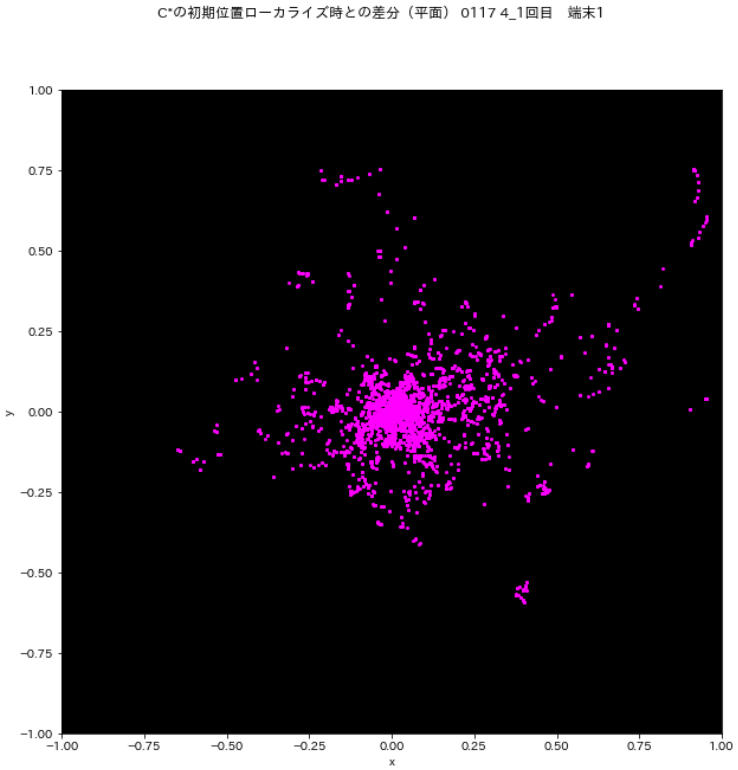
Ⅳ. 実証技術の検証 ＞ 1.自動運転車両による自己位置推定の実証
②検証結果｜継続的なトラッキングの検証

前項の検証を座標の分布を使って詳細を確認する。
右図に初期位置入力時の座標をゼロ点として、その後の座標がどう動いたかをプロットした。（下図説明参照）
継続的なトラッキングが成功していれば、下図のように点の分布に偏りが生じるはずだが、右図ではほぼ原点から等方向に分布しており、継続的なトラッキングは成功していないと言える。

図　プロットの方法



C*自己位置推定座標の初期位置からの変位



Ⅳ. 実証技術の検証 ＞ 2.政策面の検証
①検証内容

「1.自動運転車両による自己位置推定の実証」での実証と同時に、以下の政策面の検証を実施した。

1. 運用の容易性
   - 実際の運航に際して、始点・終点での自動運転開始・終了や、走行過程で必要な操作の汎用性
   - システムを設置する際の構成、工数
2. 運用コスト
   - 構成機材のコスト比較
   - データ変換のコスト

Ⅳ. 実証技術の検証 ＞ 2.政策面の検証
②検証結果

1．運用の容易性

運用の容易性について、本実証で以下の項目が検証できた。

項目	本システムの想定する運用に対しての検証結果
発車時の初回位置合わせ	スマートフォンGPSにより自動設定できることが実証
走行途中での操作	スマートフォンGPSにより自動設定できることが実証
システム設置構成、工数	スマートフォン,PC,ネットワーク装置の構成で動作 →アルゴリズムの一部をスマートフォン上で動作させることができたため、将来的には、スマートフォン単体でも動作できる可能性

Ⅳ. 実証技術の検証 ＞ 2.政策面の検証
②検証結果

2．運用コスト

運用コストについて、本実証で以下の項目が検証できた。

項目	本システムのコスト
構成機材のコスト	スマートフォン,PC,ネットワーク装置の構成で動作することを実証 →アルゴリズムの一部をスマートフォン上で動作させることができたため、将来的には、スマートフォン単体でも動作できる可能性があり、さらにコストを抑えることが可能となる
データ変換のコスト	3D都市モデルを一般的なPCで数時間で処理することで自己位置推定用のマップに変換できたことから、高精度マップを作成するよりはるかに低いコストで利用可能

Ⅰ．実証概要

Ⅱ．実証技術の概要

Ⅲ．実証システム

Ⅳ．実証技術の検証

Ⅴ．成果と課題

Ⅴ. 成果と課題 ＞ 1. 今年度の実証で得られた成果
①3D都市モデルによる技術面での優位性｜サマリ

項目	想定される技術面での優位性
3D都市モデルを活用したVPSマップ	VPSに必要なマップを個別に作成する必要がなく、オープンデータである3D都市モデル(建物モデルLOD2～3）を活用することで直接VPS用のマップとして利用することが可能であり、3D都市モデルが整備されているエリアであればVPSによる自己位置推定システムの活用が可能である。
地理座標の取得	地理座標情報を持つ3D都市モデルを直接VPS用のマップとすることで、VPSの出力座標として地理座標を利用できる。
技術・性能の標準化	様々な技術的選択肢があるなかで、今回実証した3D都市モデルを活用した「PLATEAU-VPS」を利用することで、同じ条件であれば同じ結果・性能を見込むことができる。

②3D都市モデルによる政策面での優位性｜サマリ

項目	想定される政策面での優位性
公共交通手段への適用	3D都市モデルと廉価な機材とにより高精度な自己位置推定による自動運転が可能になれば、低コストで自動運転機能を多数の低速モビリティなどに導入でき、都市部での利便性の向上や、過疎地域での住民の足としての公共交通の提供につながる。
汎用性	先進技術を共通基盤化することにより、従来では時間的・費用的なコストがかかっていた技術について、現実的なコストで導入することができる。
データの独立性	公共的資源である3D都市モデルをつかうことで、営利企業等にデータを独占されることがない。
サービスの継続性	データがオープンであり、継続的に更新されていくことから、サービスの継続性がより担保される。

今後の取り組みに向けた課題

項目	活用にあたっての課題
走行時の自己位置推定、継続トラッキング	車両が停止していたり手動で低速走行（10km/h前後）しているときは、より多くローカライズしているが、自動走行運転下（約19km/h）においてローカライズができていなかったり、遅れが発生している点から、より高速かつ高頻度なローカライズへの対応が必要である。
3D都市モデルの活用	3D都市モデルをVPS用のマップとして活用することができた一方、レンダリングにおけるライティング、テクスチャの解像度、モデルの有無による検証結果が適切な類似度であったにも関わらず、ローカライズしない地点があった。 3D都市モデルをVPS用の入力画像として使用することを前提とし、3D都市モデルと現実世界の差異を確認し、この課題を解決することが必要である。
適用モビリティの検討	電動車椅子、自動運搬ロボットなど、自動運転車両より低速なモビリティ等への適用から、現時点での課題解決を段階的に進めていくことが考えられる。

用語集

用語		内容
ア行	IMU	加速度角速度を取得するセンサを統合し、三次元の慣性運動を検出するモジュール
	RTK-GNSS	Real Time Kinematic-Global Navigation Satellite Systemの略で、GPSなど衛星を用いた「汎地球測位航法衛星システム」による測位と、地上に設置した「基準局」からの位置情報データを組み合わせることによて、高い精度の測位を実現する技術
	ADENU	Autonomous Drive Enabler by Nagoya Universityの略で、名古屋大学で 開発された自動運転車用のソフトウェアパッケージ。自動運転などの自動走行に必要 となる機能（各種センサデータの取得、地図データ管理、地図とセンサを融合した 走行状況の把握等）が体系的に含まれたソフトウェア
	NID	Normal Information Distanceの略。画像間の類似度指標の一つ
	RMSエラー	Root Mean Square Error の略で、 既知の位置と、デジタイズされた位置間の差異の尺度
カ行	カルマンフィルター	誤差のある観測値を用いて、動的システムの状態を推定し制御するためのアルゴリズム
サ行	SLAM	Simultaneous Localization and Mappingの略で、自己位置推定と環境地図 の作成を同時に行う」技術
タ行	DEM	Digital Elevation Model、数値標高モデル、地表面の地形のデジタル表現
	トラッキング	各時刻の自己位置を逐次的に推定すること
ラ行	Localizer	端末のカメラなどで撮影した画像から、あらかじめ作成しておいた三次元マップデータと 対照してカメラ行列・各種パラメータなどを計算し、自己位置を推定するアプリケーショ ン処理。画像処理により特徴点を計算し対照させる手法が多い。
	ROS	Robot Operating Systemの略。ロボット・アプリケーション作成を支援するライブラリとツール
	Renderer	3Dデータから2次元画像を生成するアプリケーション処理

自動運転車両の自己位置推定精度の向上及び有効性の検証
技術検証レポート

令和5年3月　発行
委託者：国土交通省 都市局 都市政策課
受託者：凸版印刷株式会社（株式会社ホロラボ）

本報告書は、凸版印刷株式会社が国土交通省との間で締結した業務委託契約書に基づき作成したものです。受託者の作業は、本報告書に記載された特定の手続や分析に限定されており、令和5年3月までに入手した情報にのみ基づいて実施しております。従って、令和5年4月以降に環境や状況の変化があったとしても、本報告書に記載されている内容には反映されておりません。