リアルタイムでのPoint Cloud Data Mapの配信による自動運転の支援とその評価

水谷 将也[1]　塚田 学[1]　江崎 浩[1]　飯田 祐希[2]

概要：現在、自律型自動運転車の研究開発が盛んにされている中で、自己位置推定に使用される PCD マップは非常に容量が大きい問題や、刻一刻と変化する状況に対応しなければらないという課題などがある。本論文では、これらの問題を解決するために、エッジにキャッシュされた PCD マップをリアルタイムに配信するシステムを提案する。また、そのシステムを自動運転用のオープンソースソフトウェアである、Autoware を拡張し実装した。本論文では、この手法を検証するために、本郷キャンパスでの移動データを記録した ROSBAG を再生し、自己位置を推定することにより、エッジから PCD マップをダウンロードできるかどうかを検討した。その結果、エッジサーバを用いることにより、 PCD マップをダウンロードしながら自己位置確認を行うことができることが分かった。また、帯域幅を変えてダウンロード時間を測定し、同時に自己位置推定が正常に動作する帯域幅を調べた。その結果、 100 Mbps での PCD マップのダウンロード時間は最大 698 ms であり、 4G 通信でも動作可能であることが分かった。

1. はじめに

自動運転車や Advanced Driving Asistant System(ADAS) の開発が進められている中、自車に搭載されているセンサー群、コンピュータのみで自動運転を行う自律型自動運転には様々な問題が存在する。例えば、自車で検知できる範囲が距離的に制限されたり、死角が存在したりする問題であったり、自動運転車に搭載できる計算資源やストレージなどに限界があるという問題がある。

自車両が他の車両やエッジと通信することによって、これらの問題を解決するシステムを Cooperative Intelligent Transport System(CITS) と呼ぶ。[8]

自律型自動運転の開発は様々な組織で行われているが、オープンソースとして開発されているソフトウェアがいくつか存在する。その代表として Autoware Foundation による Autoware や、Baidu 社による apollo や、NVIDIA 社による NVIDIA Drive などがある。[11][1][2]

自律型自動運転のソフトウェアでは、自車両の位置を正確に把握することが重要になる。自己位置を計測する手法として、GPS に代表される GNSS(Global Navigation Satellite System) や SLAM(Simultaneous Localization and Mapping) などがある。[10][14] ここで、GNSS では数メートルの誤差が生じることが知られており、これは自動運転においては非常に致命的である。したがって、多くの自動運転ソフトウェアでは SLAM が使用されている。

SLAM は点群データと PCD マップ (Point Cloud Data Map) を使用して行われる。[4] 点群データは LIDAR という機械で、レーザを 360 度全方位に照射し、反射した点を計測することによって、周囲の 3 次元情報を点群としてデータ化したものである。PCD マップはこの点群データの集合体と言えるが、そのフォーマットは、メタデータを含んだ header とバイナリで表された点群データを含んだ body である。メタデータには各点が含んでいる情報やデータの型、サイズなどが記されており、body の部分は数字の列がバイナリで表されているが、header で定義されたメタデータをもとにその情報を読み込む。以上のようなフォーマットであるため、3 次元のすべての点を情報として持っているため、非常にデータ量が多い。また、将来的に長距離での自動運転を行うことを想定すると、経路上の全ての PCD マップを一台の車両で保存しておくのは現状の技術では難しい。また、道路や周りの形状は交通事故や工事などによって、刻一刻と変化するため、逐次マップの更新が必要になる。これらの理由から、車車間または車とエッジの通信によって PCD マップの送受信を行い、自律型自動運転を支援することは必要であると言える。

本論文では、この課題を解決するための先駆けとして、オープンソースソフトウェアである Autoware を拡張し、自車位置に合わせてエッジからマップをダウンロードする

[1] 東京大学大学院情報理工学系研究科
[2] 株式会社ティアフォー

実験を行った。また、マップをダウンロードする際にかかる遅延などを計測し、評価した。

本論文の構成は次のようになっている。第 2、3 章にて車両同士の通信に関する標準をはじめとした関連技術・関連研究について紹介し、第 4 章にて本研究における要件を設定し、手法を提案する。第 5 章にて提案手法の評価を行うためのフィールド実験の手法について説明し、第 6 章にて実験結果に対する評価を行う。第 7 章にて結論と今後の課題を述べる。

2. 関連技術

2.1 Autoware

自動運転のためのソフトウェアは多くあるが、本研究ではオープンソースのソフトウェアである Autoware を使用する。したがって本節では Autoware について説明する。Autoware による自動運転は図 1 のように行われる。まず、LIDAR やカメラなどのセンサーで周辺環境をセンシングする （Sensing)。次に、得られた情報から自己位置を推定すると同時に、画像認識などで物体を認識、予測する (Perception)。そして、どの経路を通るかや一時停止をするかなどの経路決定を行う（Decision)。最後に、車両の各部位をどのように動かすかを決定し、指示を出す（Planning, Actuation)。

これらの機能を実現するために、Autoware は ROS というミドルウェア上で実装されている。[3] ROS はロボットの制御を行うために最適化されたミドルウェアであり、分散処理を行うために、各プログラムをノードで実装し、各ノードはトピックで情報を交換する。

ここで、Localization（自己位置推定）というプロセスについて説明する。自己位置推定で必要な入力情報としては位置推定の開始位置を指定するための GNSS から得られる位置情報、LIDAR から得られる点群データ、PCD マップである。そして、LIDAR から得られる点群データと PCD マップを比較し、最も一致している確率の高い位置を自己位置として推定する。前章でも述べたが、ここで使用される PCD マップのデータ量が非常に多いことが、本研究の課題の要因である。

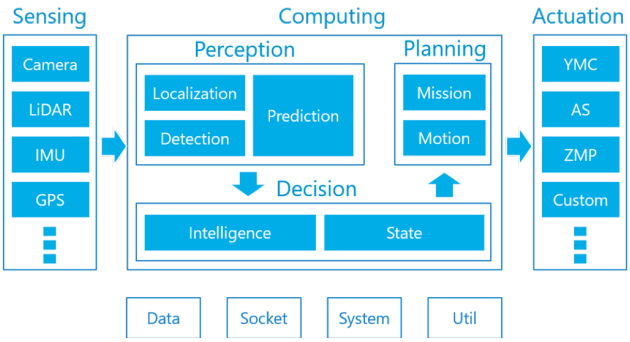


図 1: Process of self-driving in Autoware

2.2 座標系と PCD マップ

Autoware では多様な座標系を組み合わせることで空間を表現している。代表的なものとして、Autoware に入力されている地図データの中心を原点とする座標系、地図の座標系（並行直角座標系の原点を原点とする座標)、車両の座標系（後輪車軸の中身を原点とする座標)、LIDAR の取り付け位置を原点とする座標系などがある。したがって、周辺環境の情報を Autoware が認識するにはその都度変換が必要になる。

例えば、LIDAR が観測した点群データを地図データと照らし合わせたい場合には、LIDAR を原点とする座標系から、地図データの中心を原点とする座標系へと変換しなければならない。

ここで、Autoware では UTM 座標系を基調とした地理識別子による空間参照で全世界の座標を特定することができる MGRS を座標系として使用している。これは投影座標系に由来するため、緯度によって特定することができる範囲の面積が、約ではなく厳密な距離四方で特定できるという利点がある。また、地図データの中で、PCD マップはデータ量が膨大なため、正方形に区切られて保存され使用される。(図 2)

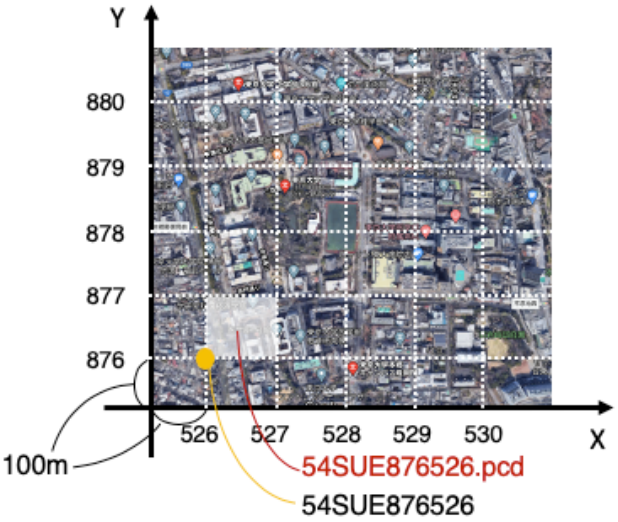


図 2: PCD map and MGRS coordinate system

2.3 通信メディアとエッジコンピューティング

CITS では車車間の通信 (V2V) や、車両とインフラ (エッジ) との通信 (V2I) などを含む、車両とすべてのものとの通信 (V2X) を使用する。これら通信に使用するメッセージフォーマットは European Telecomunications Standards Institute(ETSI) や International Organization for Standardization(ISO) によって、自車両の周囲の情報を伝搬するための Cooperative Awareness Message(CAM)

や緊急情報を伝搬するための Decentralized Environmental Notification Message(DENM) などが標準化されている。[7][5] V2V、V2I は非常に高速に移動する物体同士の通信であり、これらの通信方式は従来のネットワークで使用されるような、基地局や固定網を介した通信ではなく、P2P の通信で実現する Vehicular Adhoc Network(VANET) によって実現される。VANET を実現する通信規格としては、5.8GHz 帯を使用する Dedicated Short Range Communications(DSRC) や最近では 5G など携帯網による D2D 通信がある。[12] DSRC や 5G による通信は伝送速度が従来の規格や、インターネットを介した通信に対して高速である。(表 1) この利点を活かすために、計算処理をインターネット越しに分散することで高速なサービスを提供するクラウドサービスを基地局や WiFi のアクセスポイントの近く (エッジ) にサーバを設置し、サービスを提供するエッジコンピューティングの研究が多くされている。

表 1: Specifications of communication standard [6]

通信規格	無線帯域	伝送速度
DSRC	10MHz	3 ～ 27Mbps
5G	100MHz	1 ～ 10Gbps(実効値 30Mbps)

3. 関連研究

エッジを使用してマップなどの容量の大きなデータを効率的に配信する研究は多くなされている。

Kim ら [13] はエッジにあるストレージサーバの容量が限られているというシナリオでクラウドにあるデータを効率よく配信するアルゴリズムを提案した。この研究では、車両の経路とデータリクエストをすべてクラウドが把握できるという条件で、データをエッジの AP にプリフェッチすることで、データの到達確率を向上させることを目的とした。ここでは、エッジのストレージ容量や、通信帯域などを制約条件として、データの到達確率の目的関数を最大化するようなパラメータを求める。更に、各アクセスポイントのデータの到達率などが、予めわかっておらず、MAB ベースの学習によって把握していく場合についても検証を行った。これらのアルゴリズムをシミュレーションによって検証実験することで、多項式時間で準最適解を見つけることが可能であることを示した。

Gangadharan ら [9] はクラウドとエッジを利用した最適なマップの配信を提案した。この論文では、エッジや車両のデータをすべてクラウドに集約し、解析をし、その結果をもとにクラウドが指示を出すシナリオを想定している。ここでは、データのメモリ量、データの到達時間、エッジサーバのリソース、車の密度、帯域などを変数として、すべての制約を方程式で表し、すべてのエッジの帯域最大使用率を最小にすることを目的関数として定義することで、通信の最適化を行う実験を行った。実験は Matlab と IBM ILOG CPLEX を使用したシミュレーションによって行われ、使用する通信帯域の最適化を行う上で、車両の動きを考慮することの有用性をしめした。

Zhang ら [15] はエッジにアクセスポイントがあり、車はエッジを経由してデータを取得するシナリオを想定し、通信の負荷を最適化する手法を提案した。従来の手法は通信を最適化するために、帯域や車両密度などのデータを中央サーバに集約し、最適な解を導き、その結果をエッジや車両に指示する必要があるという課題があった。しかし、この研究では、通信プロトコルとして、IP の代わりに、Named Data Network(NDN) を使用することで、自律的にデータをエッジにキャッシュすることが通信時間、通信帯域を小さくすることに役立つことを示した。また、NDN のメッセージとして、データをリクエストするパケット (Interest) とデータを返信するパケット (Data) に加えて、データをプリフェッチするパケットを新たに提案し、これによってデータの取得の遅延を更に小さくすることができることを示した。

実用的なネットワーク環境を構築した上で、実際の自動運転に用いられるソフトウェア (Autoware) を使用して、地図ダウンロードの検証をした研究はこれまでになかった。

4. 本研究の目的

4.1 PCD マップに関わるシステムの課題

将来的には自動運転は長距離で行うことが想像される。また、実際に公道を走る距離が長くなればなるほど、交通事故や工事による車線規制などが多くなることが想定される。ここで、自己位置推定の際には Autoware では PCD マップを使用するが、現状のシステムでは、走行経路のデータ全てを車両内に保存しておく必要がある。また、PCD マップの更新機能も存在しない。このような仮定のもとでの問題点として以下の二つが存在する。

(1) 長距離にわたる PCD マップを全て車両に保存しておくことが困難であること。
(2) 車線規制などによって道路周辺の環境が変化した場合に対応できないこと。

4.2 PCD マップ配信システムの要件

以上の問題点を解決するために、以下の要件が挙げられる。

4.2.0.1 PCD マップは車両に全て保存するのではなく、エッジやクラウドなどからデータをダウンロードできること

PCD マップをすべて保存することができないという 1 つ目の問題点を解決するために必要である。

4.2.0.2 PCD マップ配信の遅延を小さくすること

PCD マップは自己位置推定に関わるため、自動運転の

根幹に関わる部分である。したがって、PCD マップの配信を想定したときに、PCD マップの正確性を担保する上で、この要件が必要である。

4.2.0.3 PCD マップはリアルタイムに近い更新頻度であること

道路周辺の環境が変化したときに対応できないという 2 つ目の問題点を解決するために必要である。

5. PCD マップ配信システム

5.1 PCD マップ配信システムの概要

本研究では以上の要件を満たすようなシステムを提案する。まず、本システムの構成を図 3 に示す。図のようにシステム内には、クラウド、エッジ、車両の三種類のノードがある。クラウドは、PCD マップの管理を行うための HTTP サーバである PCD マップレジストリ、PCD マップを保存しておくためのストレージ、保存先のファイルパスを保存しておくためのデータベースで構成されている。エッジは、クラウドに PCD ファイルの更新を確認する同期サーバ、PCD マップを保存しておくためのストレージで構成されている。車両には、自動運転のソフトウェアである Autoware と、PCD マップレジストリに http リクエストを送信するためのクライアントで構成されている。

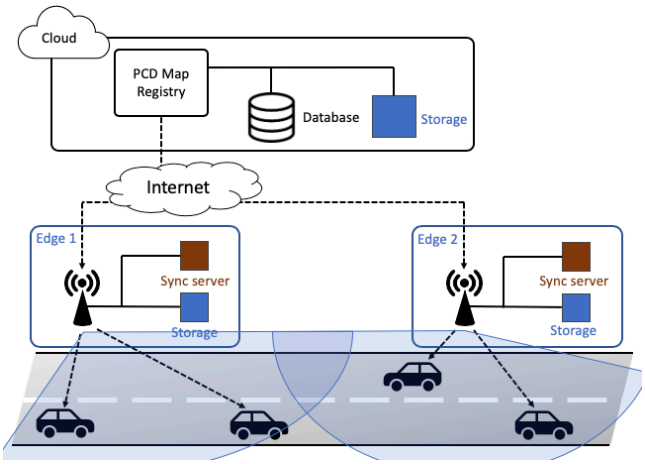


図 3: System configuration

次に、図 4 でシステムのフローを示す。

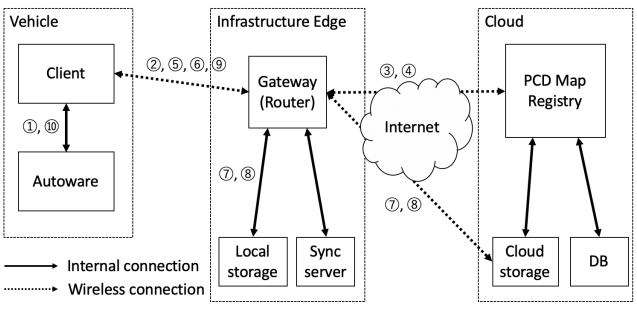


図 4: Process of system

図のように、車両とクラウドサーバの間にエッジを配置し、Map の配信を行う。PCD マップのダウンロードの流れは以下の通りである。

地図の更新確認自車位置の変化を検知し、HTTP クライアントにファイルをリクエスト

地図の検索HTTP クライアントから PCD マップレジストリに PCD マップのファイルパスをリクエスト

最新地図の発見マップレジストリはデータベースを参照しエッジストレージにマップがあればそのファイルパスを、なければクラウドのファイルパスをクライアントに送信

地図の取得クライアントから該当するストレージにマップをリクエストし、ストレージからクライアントにマップを送信

このシステム構成にすることで、要件 1 の「PCD マップは車両に全て保存するのではなく、エッジやクラウドなどからデータをダウンロードできること」が達成される。

5.2 同期サーバ

本システムが段階的に普及していく中で、エッジネットワークが新たに設置される場合、既存のアーキテクチャを壊さないことが必要となってくる。そのため、エッジネットワークに PCD マップをキャッシュする仕組みは push 型とする。なので、エッジネットワークはグラーバルな ip アドレスによって、一意に判別可能である必要がないため、プライベートなネットワークとする。また、エッジネットワークは管理するエリアを予め決めておく。更に、エッジネットワークには定期的に PCD マップレジストリの更新を確認する同期サーバを設ける。具体的な同期の手順は以下の通りである。

地図のバージョン履歴の検索同期サーバからクラウドの PCD マップレジストリに管轄エリアのマップのファイルパスをリクエストする

最新地図の発見、地図の取得更新がある場合は該当のファイルパスにアクセスし、PCD マップをダウンロードする

地図の DB 管理更新があった場合は、PCD マップレジストリに新たな PCD マップのファイルパスを登録する

同期サーバでは以上の流れを定期的に行う。

6. 実装

6.1 車両側での処理

車両側のシステムの詳細なアーキテクチャを図 5 に示す。図では Autoware のノードとトピックの中で関係のある部分のみを抜粋し、エッジストレージやクラウドストレージとの関係を示している。

本研究では”new_points_map_loader”というノードを新たに実装した。処理の流れを図 6 に示

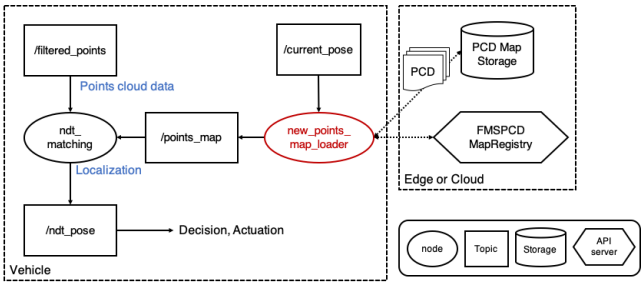


図 5: Detailed system architecture on the vehicle side

す。”new_points_map_loader”では、車両の現在の座標を示す”/current_pose”というトピックを Subscribe している。そして、”/current_pose”に合わせて、PCD マップレジストリに現在の座標周辺の PCD マップのファイルパスをリクエストし、受け取ったマップのファイルパスを元に PCD マップをリクエストする。最後に、受け取ったマップを”/points_map”というトピックに Publish している。

ここで、”/current_pose”は Autoware 内の自車両の後輪の中心を原点とした座標系で自己位置を表している。そのため、”new_points_map_loader”は自己位置を MGRS 座標系に変換する。そして、自己位置が存在するグリッドの周囲のグリッド (本研究では $5 \times 5 = 25$) のマップも含めてマップをリクエストする。走行中も常に”/current_pose”を Subscribe しており、自己位置のグリッドが変化すると車両内に保存されていないマップをリクエストする。以上のアルゴリズムによって自己位置に合わせて周辺のマップを Autoware にロードできる。

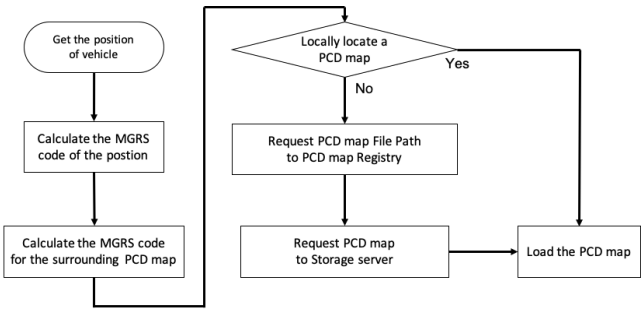


図 6: Process on new_points_map_loader

7. リアルタイムで PCD マップをダウンロードする実験

7.1 実験環境

本実験は無線通信を使用したリアルタイムでのマップのダウンロードの評価を行うことが目的であり、予め収録しておいた本郷キャンパス内の走行データを再生し、実験を行った。本システムを評価するために、図 7 に示すような実験環境を設けた。クラウドサーバとしては AWS の EC2 インスタンス（バージニア北部リージョン）と s3 を使用している。エッジにはプラベートなネットワークを作成し、ストレージサーバを PC で動かした。ストレージサーバとしては minio を使用した。車両として一台の PC をルータに無線接続し、中で Autoware を動かし、走行データを再生した。

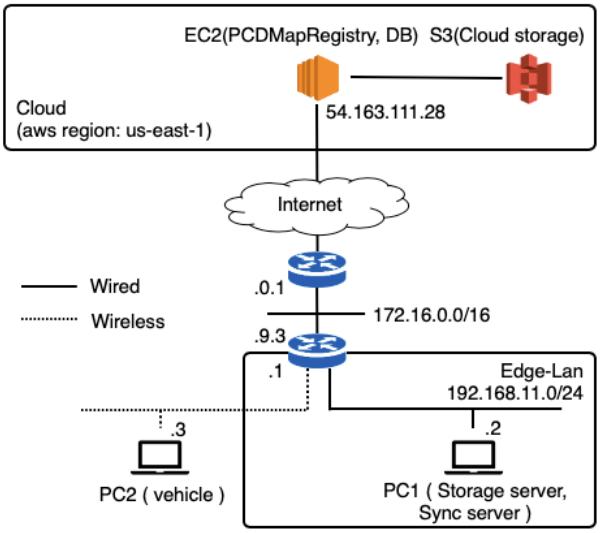


図 7: experiment environment

7.2 通信の帯域幅

本実験で使用する通信部分のスペックを表 2 に示す。

また、iperf を使用して通信の帯域幅を確認した。まず、PC1 を wifi ではなく、Ethernet 1000Base-T でルータと接続し、PC1 に iperf server を立て、PC2 から iperf のテストを行った。次に、PC1 を WiFi IEEE802.11ac でルータと接続し、同様のテストを行った。以上の結果から、IEEE 802.11ac の帯域幅が 638Mbps であることがわかる。また、PC1 からクラウド上の EC2 に立てた iperf server に同様のテストを行った結果、592Kbps であった。

7.3 実験内容

まず、図 8 に示すような経路を走行したデータ（rosbag）を PC2 上で動作する Autoware で再生する。ROSBAG は走行中に lidar によって取得した点群データなどを Autoware が処理し、publish した topic のうち選択したものを保存したものである。したがって、ROSBAG を再生することによって、擬似的に車両を走らせて実験を行う。本実験では、ROSBAG を再生するとともに、新しく実装した”/new_points_map_loader”というノードを動かしてエッジストレージにある PCD マップをダウンロードしながら走行できるかの実験を行った。なお、自車両が位置する PCD マップに隣接する $5 \times 5 = 25$ 個のマップをダウンロードする設定で行った。

表 2: Communication method within the edge

場所	通信方式	帯域幅
PC1・無線ルータ間	WiFi IEEE 802.11ac	638Mbps
PC2・無線ルータ間	Ethernet 1000Base-T	942Mbps
PC1・EC2	-	592Kbps

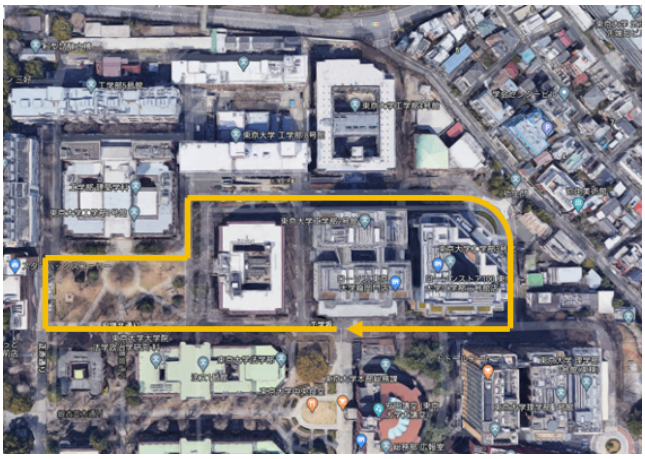
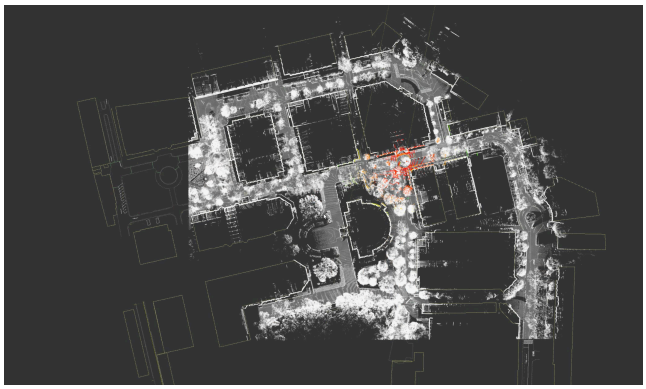
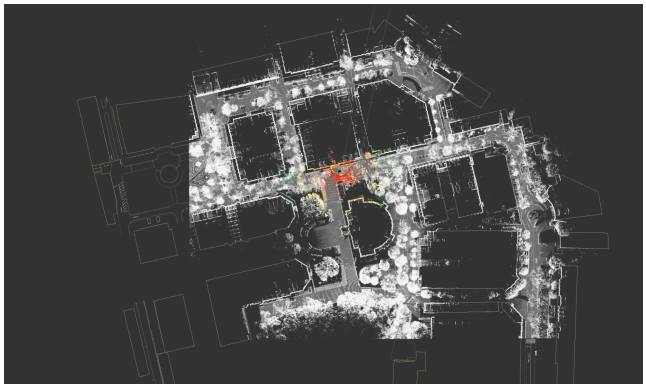


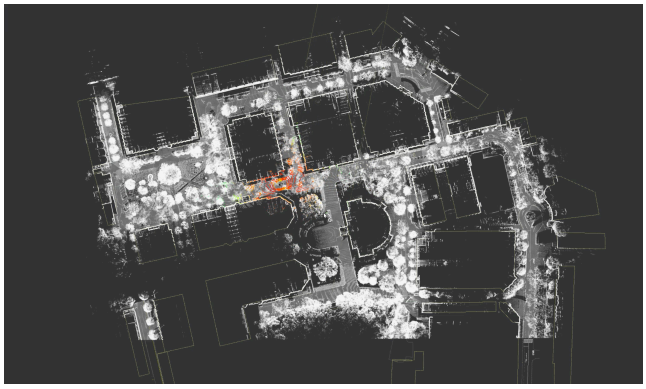
図 8: Driving route



(a) Stopped at the initial position



(b) State after a while after starting



(c) State where the PCD map has changed

図 9: Rviz 上の表示

7.4 実験結果

図 9 は実際に走行した結果を Rviz で表示した結果である。白い点群がロードしている PCD マップ、白や黄色の線が地物情報をあらわしたベクターマップと呼ばれるものである。(a) から順に初期位置で停止している状態、発進してしばらくした状態、自車両の位置する PCD マップが切り替わったときを表している。この結果から、本実験環境では、走行しながら WiFi を使用してエッジストレージからリアルタイムで自己位置に合わした、25 個の PCD マップをダウンロードすることが可能であることがわかった。

8. 帯域幅とダウンロードにかかる時間の影響

本実験では前章の構成でマップのダウンロードにかかる時間を測定した。実験においては本郷キャンパス全域のマップを使用し、それらをエッジストレージからダウンロードするのにかかる時間をそれぞれ求めた。その結果を図 10 に示す。また、クラウドサーバからダウンロードするのにかかる時間を計測した結果を図 11 に示す。

エッジストレージからダウンロードするのにかかる時間は平均で 109ms、最大で 282ms かかることがわかった。この結果から本郷キャンパス内の pcd マップであれば、最大でも 290ms 以内に抑えることができることがわかり、例えば、周囲 25 このマップすべてをダウンロードする際にかかる時間は最大で、$25 \times 0.290s = 7.25s$ であると言える。ここで、車両が時速 20km/h で走行すると考えると、車両の位置が変化し、新たに PCD マップをダウンロードしないといけなくなるまでにかかる時間は、$0.1km \div 20km/h \times 60 \times 60 = 18s$ であるため、マップのダウンロードは可能である。一方で、車両が時速 60km/h で走行すると考えると、車両の位置が変化し、新たに PCD マップをダウンロードしないといけなくなるまでにかかる時間は、$0.1km \div 60km/h \times 60 \times 60 = 6s$ であるため、最悪の場合を考えると、マップのダウンロードは可能ではない。

また、クラウドからダウンロードするのにかかる時間は

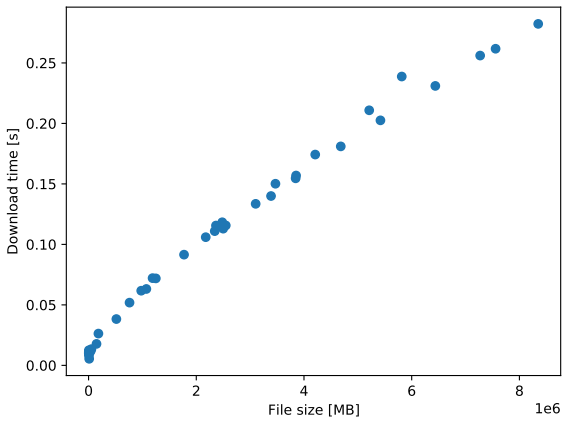


図 10: Time it takes to download maps from PC1 to PC2

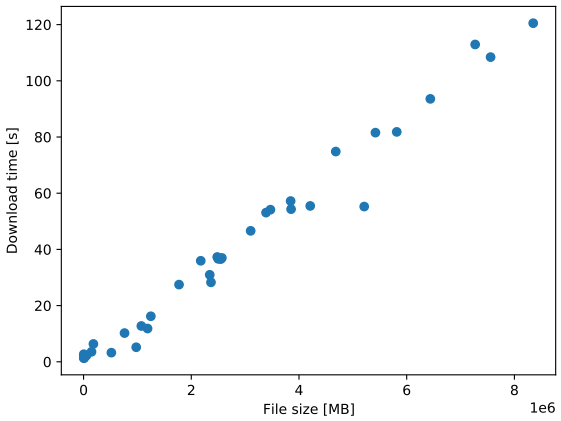


図 11: Time it takes to download maps from S3 to PC2

平均 37.0s で、最大で 121s かかることがわかった。以上の結果から、エッジからダウンロードする時間はクラウドからダウンロードする時間の平均 7.25 倍も小さくなっていることがわかる。以上から低速で移動する車両では PCD マップをダウンロードしながら、自己位置推定をすることが可能であることがわかった。

次に PC2 からデータを送信する際の帯域幅を変化させて PCD マップをダウンロードする実験を行った。図より、例えば、30Mbps のときは、PCD マップのダウンロードにかかる時間は平均で 715ms であり、25 個の PCD マップをダウンロードすることを考えると、$25 \times 0.715s = 17.875s$ である。現行の 4G では、最大で 30Mbps 以上の速度が出るため、4G でも本システムの運用が可能であることがわかる。[6]

次に、PC2 の帯域幅を変化させて、自己位置推定を正常に行うことができるか否かを実験した。その結果、PC2 の帯域幅が 7Mbps までは正常に自己位置推定ができていたが、6Mbps で正常に自己位置推定ができなくなることがわかった。本実験で使用した走行データは平均 $13.1km/h$、

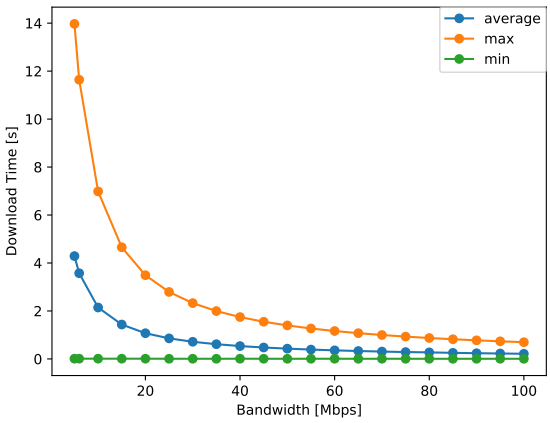


図 12: Time it takes to download the PCD map when the PC2 bandwidth is changed.

最大 $27.3km/h$ である。

9. まとめ

本稿では車々間通信に使用される通信メディアについて説明し、DSRC や 5G の利点を活かしたエッジコンピューティングについて説明した。更に、エッジストレージを使用して効率的にデータ量の大きなものをダウンロードする研究について紹介した。また、自動運転に必要な自己位置推定で使用する PCD マップの問題点について説明した。それを解決するためのシステムを提案した。また、Autoware をどのように拡張したかなど、実装についても説明した。本稿では、この手法を検証するために、本郷キャンパス内の走行データを収録した ROSBAG を再生し、自己位置推定をしながら、PCD マップをエッジからダウンロードすることができるかの検証を行った。その結果から、エッジストレージを使用することで、PCD マップをダウンロードしながら、自己位置推定を行うことが可能であることがわかった。また、帯域幅を変化させてダウンロード時間を計測し、同時に自己位置推定が正常に動作する帯域幅を調べた。その結果、100Mbps のときの PCD マップのダウンロード時間は最大でも 698ms であることがわかり、4G による通信でも、本システムが動作することを示した。今後の展望としては、車両が多くなった際の動作の検証と、PCD マップのアップロードを行うシステムについて検討したい。

参考文献

[1] Apollo. `http://apollo.auto/`. Accessed: 2020-4-25.
[2] Autonomous car development platform from NVIDIA DRIVE AGX. `https://www.nvidia.com/en-us/self-driving-cars/drive-platform/`. Accessed: 2020-4-25.
[3] Documentation - ROS wiki. `http://wiki.ros.org/Documentation`. Accessed: 2020-4-25.

[4] PCL - point cloud library (PCL). http://pointclouds.org/. Accessed: 2020-4-24.

[5] *ETSI EN 302 637-3 v1.2.2(2014-11), Intelligent Transport Systems(ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service.* 2014. http://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.02.02_60/en_30263703v010202p.pdf.

[6] Minjin Baek, Donggi Jeong, Dongho Choi, and Sangsun Lee. Vehicle trajectory prediction and collision warning via fusion of multisensors and wireless vehicular communications. *Sensors*, Vol. 20, No. 1, January 2020.

[7] ETSI. EN 302 637-2 - V1.3.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service.

[8] ETSI. TR 102 962 - V1.1.1 - Intelligent Transport Systems (ITS); Framework for Public Mobile Networks in Cooperative ITS (C-ITS). 2012.

[9] D Gangadharan, O Sokolsky, I Lee, B Kim, C Lin, and S Shiraishi. Bandwidth optimal Data/Service delivery for connected vehicles via edges. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 106–113, July 2018.

[10] Daniel garcia yarnoz. International committee on global navigation satellite systems (ICG). Accessed: 2020-4-24.

[11] S Kato, S Tokunaga, Y Maruyama, S Maeda, M Hirabayashi, Y Kitsukawa, A Monrroy, T Ando, Y Fujii, and T Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 287–296. ieeexplore.ieee.org, April 2018.

[12] J. B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, Vol. 99, No. 7, pp. 1162–1182, 2011.

[13] R Kim, H Lim, and B Krishnamachari. Prefetching-Based data dissemination in vehicular cloud systems. *IEEE Trans. Veh. Technol.*, Vol. 65, No. 1, pp. 292–306, January 2016.

[14] John J Leonard and Hugh F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IROS*, Vol. 3, pp. 1442–1447, 1991.

[15] Zhiyi Zhang, Tianxiang Li, John Dellaverson, and Lixia Zhang. RapidVFetch: Rapid downloading of named data via distributed V2V communication.