第11部

特集11 SoftwareDefinedMediaコンソーシアム

SDM WGメンバー

第１章　はじめに

Software Defined Media（SDM）コンソーシアムは、2014年に設立され、オブジェクト指向のディジタルメディアと、インターネット環境が前提の映像・音響空間を用いたビジネス創造を目指してきた。昨年までの報告書では、SDMのコンセプトやアーキテクチャ、試作プラットフォーム[129、130]、データ収録[130]、収録データベース設計[132、133、134]、SDMプラットホームを利用したアプリケーションであるSDM360^2[135、136]やLiVRation[137、138]、Web360^2[139]などを報告してきた。

本報告書では、2021年度に実施した活動を報告する。

第２章　ビル管理におけるBIMに基づくデジタルツイン基盤の研究

スマートビルディングとは空間とIoTを融合させることで、建物内の設備を一元管理し、エネルギー効率の最適化や快適性の向上などを目指すことである。我々はビルを対象にBIMやIoTを駆使したスマートビルディングの実証実験を行い、この実証実験では、事前に用意したBIMデータと静的センサなどによる各種データをもとに運搬ロボットの移動経路を動的に算出し、目的地への指令を出すなどの課題を解決することを試みた。課題として、人や什器などがオフィス内で常に変化しているため、事前に用意したマッピングデータのみでは、ロボットは新しく設置された障害物や人にぶつかってしまい、目的地にたどり着けないことが上げられる。これは配置が決まっていることが多い工場などと違い、インテリアのために什器が頻繁に置き換えることもあるオフィスなら

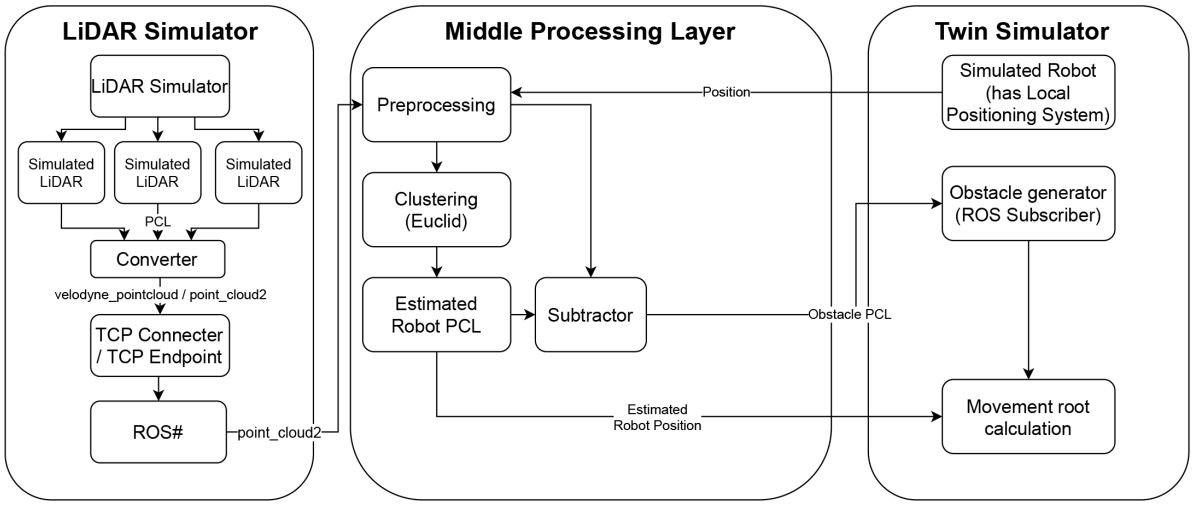

図1　行動計画システム概要図

ではの問題である。また、ロボットは、ロボット本体がSLAMを搭載している場合などには、移動しながら自身に搭載されたセンサを元に自分の中にローカルマップを生成するため、解決可能だが、データとしてはロボットからの情報に限られてしまい、実際には最短経路でない経路を選択してしまうことも考えられる。

そこで、本研究では、物理空間に事前に設置した複数のLiDARセンサの協調知覚により、本課題を解決することを目指した。これは、環境側にセンサを搭載することで、自律移動型ロボットの再マッピングの手間を省き、更にLiDARセンサを用いることで、高精度な測定を目指す。また、運搬ロボットは前面にしかLiDARセンサがついていないことが多く、他方面からの追突は予期していない場合があるため、環境側から認知することで、警告音を鳴らすといった対処が行えるといった利点が考えられる。

2.1　提案手法

図1にロボットの行動計画を行うシステムの概要図を、図2に本研究のデジタルツインの構成図を記載した。本研究では、提案手法として、対象のエリアに複数のセンサを設置し、それぞれのセンサのマッピング情報を集積してエリア全体のマッピングを行い、統括管理を行う方法を提唱する。これにより、ロボット単体でのマッピングと比べ、不明瞭な空間を排斥し、正しい最短経路を導きやすくなると考えられる。また、その際、LiDARセンサはある程度上方に設置することで障害物による対象物の遮蔽の影響を緩和する。

図1において、本手法を説明する。まず左のLiDAR Simulatorにて、複数のLiDARセンサをシミュレートし、複数の点群データの集合体を得る。次にその点群データ中間処理層にてクラスタリングする。ここで得られたクラスタの中からロボットと思われる物を判別し、その位置情報を中間処理層の一つの入力として扱う。右側のTwin Simulatorでは、現実世界のロボットのツインとなるオブジェクトを配置する。今回、実証実験にて使用したロボット（FRUTERA A-ROBOT-102）[*1]はある程度のノイズを含んだ自己位置推定ができており、その位置情報を活用できるものとする。この位置情報を中間処理層の二つ目の入力として使用する。最後にEuclid Clusteringと後述するロボットの推定方法により推定されたロボットの位置情報と、最初の点群の集合体からロボットと思われる点群クラスタを引いた障害物と思われる点群とを組み合わせ、ツイン上でロボットの移動経路を算出する。なお、この際、BIMデータの属性情報であるIfcSpatialZoneを使用し、ロボットの行動可能領域を事前に定めることが出来るといった機能も用い、BIMデータを活用する。

なお、ビルをテーマにした上記デジタルツインアプリケーションは、東京大学グリーンICTプロジェクトBIM基盤WG[*2]にて、研究を行った。我々は、主にデジタルツインアプリケーションの作成に従事し、利用したジオメトリデータはエヌ・ティ・ティ・コミュニケーションズ株式会社[*3]のスマートラボ、BIMデータの構築やバックエンドの作成、ロボットの調達などは本ワーキンググループに参加している各企業が行った。また、後の2.2.1項にて作成したAPIサーバは研究グループの一人と協力して作成した。

2.2　設計と実装

本アプリケーションは東大グリーンICTプロジェクトのBIM基盤WGの一つの実証実験としてNTTコミュニケーションズ田町ラボでの実験をベースとした開発を行って

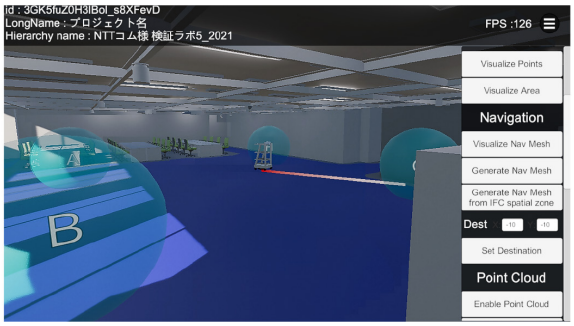

図2　デジタルツインアプリケーションのスクリーンショット

＊1　http://androbo.jp/business/r/frutera/
＊2　https://www.gutp.jp/activity/?id=204
＊3　https://www.ntt.com/index.html

いる。そのため、デジタルツインアプリケーションとしてロボットの制御以外にも照明、空調の制御、ビーコンの情報取得など様々な機構が作成されているが、本実験と関係性が薄い部分の説明は省略、又は省いている。シミュレーターを用いた本実験は、この様々なセンサの可視化及び制御機構を有したデジタルツインアプリケーション上で行う。

2.2.1　アプリケーション実装

今回、NTTコミュニケーションズ内での実証実験を行う際に、ロボット制御を行うAPIへのアクセスは、特定IPからのアクセスしか受容しないホワイトリスト制であった。そのため、この実証実験に向け、我々はゲームエンジンが行う全ての対外通信を総括するAPIサーバをAzureを用いて新しく作成した。具体的なアプリケーション構成を図3に示した。このサーバはゲームエンジンからはREST APIとして振る舞い、内部ではロボット制御のAPIに対してMQTTを用いて通信を行い、空調、照明状態の確認を行うAPIに対してはRESTを用いて通信を行う。また、本アプリケーションはゲームエンジンであるUnity3D[*4]を用いて作成した。

2.2.2　中間処理層

シミュレータはLiDARの挙動をシミュレートし、取得した点群データはROSトピック/velodyne_pointsにPublishされる。本項では、LiDARセンサから受け取った点群データを処理するためのROSパッケージを作成するために用いた手法を説明する。中間処理層では、ROS（Robot Operating System）[*5]と点群データを処理するために用いたライブラリとしてオープンソースにて開発されているPCL（Point Cloud Library）[*6]を用いた。

ここでは、図4のようにセンサを環境に設置し、そこにロボットを配置した際に、中間処理層にて行う各処理を実行した結果を記載していく。なお、センサの位置は固定し、点群の組み合わせを実行する際に、正しく組み合わせを行えるように全てのセンサの位置情報は事前に分かっているものとする。

■点群の組み合わせ処理

各LiDARセンサはROSトピック/velodyne_pointsに点群データをアップロードするが、LiDARセンサごとに情報を分離するために、ヘッダー情報に含まれるframe_idをvelodyne_x（xはナンバリング）ように変更する。

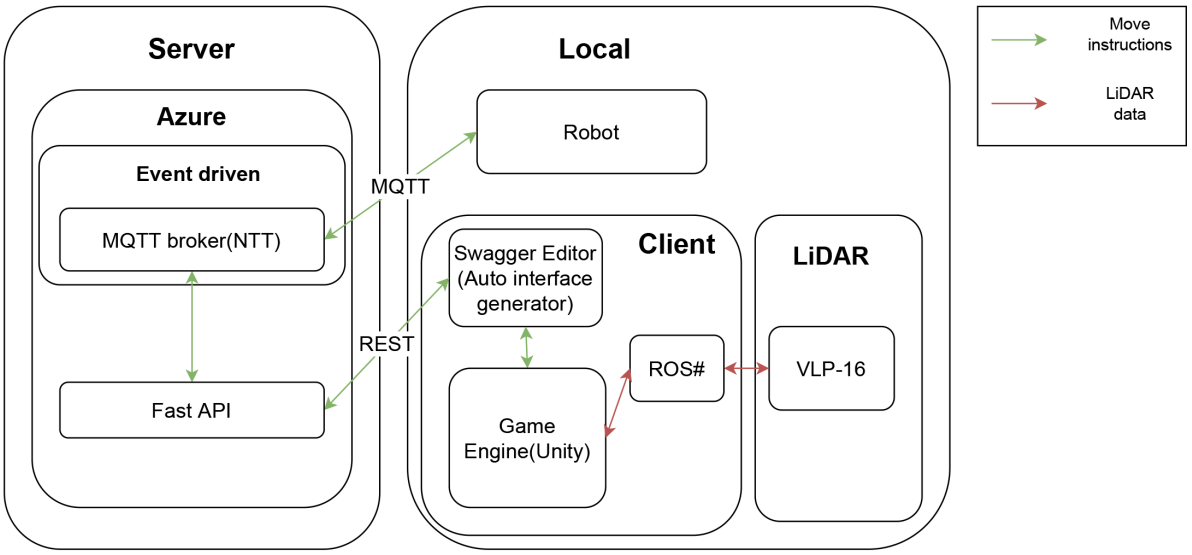

図3　アプリケーション構成

＊4　https://unity.com/
＊5　https://www.ros.org/
＊6　https://pointclouds.org/

次に各センサの座標を元に、それぞれの点群に対し、座標変換を行う。最後に一つの点群データとして組み合わせを行う。図5は、以上の操作を行った際の点群データの可視化である。

■点群の軽量化

点群の組み合わせ処理を行った点群データは一つのセンサが取得する点数とセンサの数の掛け合わせを行った点数であるため、点群数の削減を行うことで処理の高速化を図る。

ここでは、Voxel Grid Downsamplingを行う。これは、入力した点群データに対し、空間上に三次元の立方体(ボクセル)を配置し、各ボクセルに含まれる点群をそれらの重心で近似するといった処理である。これにより、入力点群の特徴を抑えたままダウンサンプリングを行っている。組み合わせ処理を行った点群データに対し、ボクセルサイズを5[*cm*]に設定し、実行したところ、約228,000点数の点群が約18,000点数まで削減された。

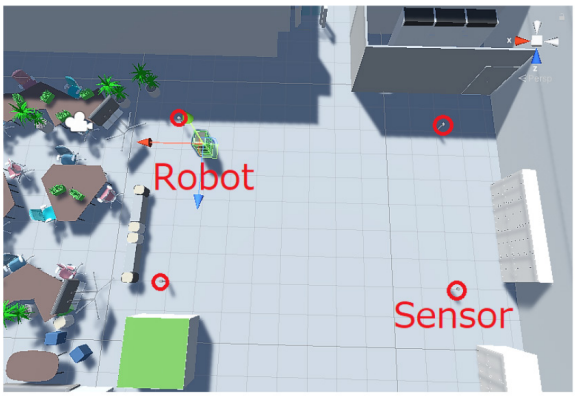

図4　シミュレータ上のロボット、センサの配置

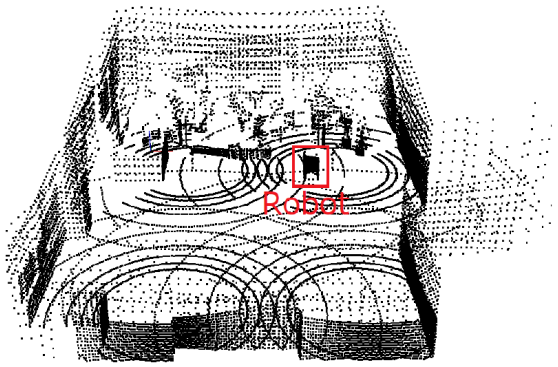

図5　VLP-16シミュレータによって得られた点群

■床面の点群の削除

与えられた点群の中から一定以上の大きさの平面を検出する手法として、Plane Model Segmentation[140]という手法が存在する。この手法では、入力モデル(ここでは平面)と距離の閾値を指定し、RANSAC[141]を用いて、入力点群の中から入力モデルと近しい平面を抽出する。

しかし、今回はロボットは段差など高低が発生する物体を登らず、移動する位置座標の高さは常に一定であるため、床を含む一定以下の高さの点群を削除するといった処理を行っている。図6は点群の軽量化を行った入力点群に対し、この処理を行った結果を可視化したものである。

■点群のクラスタリング

ここでは、点群の特徴を踏まえ、クラスタリングを行い、点群を各物体に分離する。ここではEuclid Clustering Extraction[142]を用いる。Euclid Clustering Extractionとは、オクツリーデータ構造を用い、各三次元ボックス内のデータをkd-tree構造を用いて最近傍探索を行い、各クラスタを抽出する。

図7は、床面の点群の削除した後の点群にEuclid Clusteringを実行し、各クラスタごとに色を割り当てた際の様子である。図の真ん中には、対象のロボットが茶色

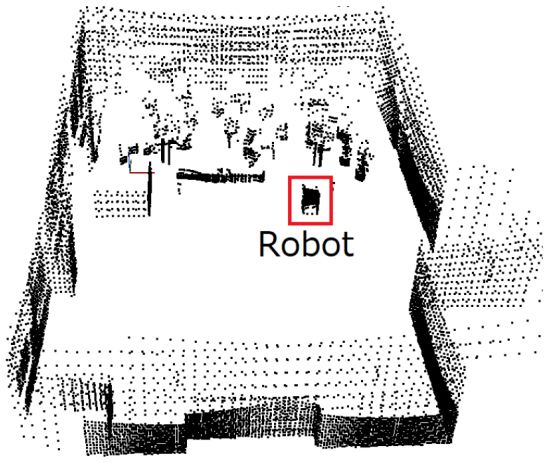

図6　床面の点群の削除を実行後に得られた点群

のバウンディングボックスと共に黒く塗りつぶされている(赤色の四角で囲まれている物体)。

■ロボットの座標の推定
ここでは、抽出した各クラスタからどのクラスタがロボットであるかを特定する。ここでは、前提条件として、ロボットの位置情報が利用できるため、経路探索を行う際のロボットの初期座標を用いる。初期座標近辺に存在し、ロボットの特徴(今回は大きさを使用した)と合致するクラスタを抽出し、そのクラスタがロボットであると想定する。

以上の操作から推定したロボットの位置はROSトピック/processed_poseとして、PoseStamped型でPublishしている。また、障害物情報は推定したロボットの点群を削除したうえで、ROSトピック/processed_pc2として、PointCloud2型でPublishしている。

なお、ここまでの処理にかかる実行時間は、平均約192.43[*ms*]となった。

2.2.3　経路探索

経路探索には、Unity3Dの機能の一つであるNavigation System[*7]を用いる。この機能では、仮想空間上のジオメトリデータを元に、移動可能領域をNavMeshとして算出し、その領域内でA*アルゴリズムを用い、経路探索を行うことができる。本研究では、VLP-16(及びそのシミュレータ)から取得し、中間処理を施した点群情報をそのままジオメトリ上に反映させ、障害物としてNavigation Systemに認識させている。

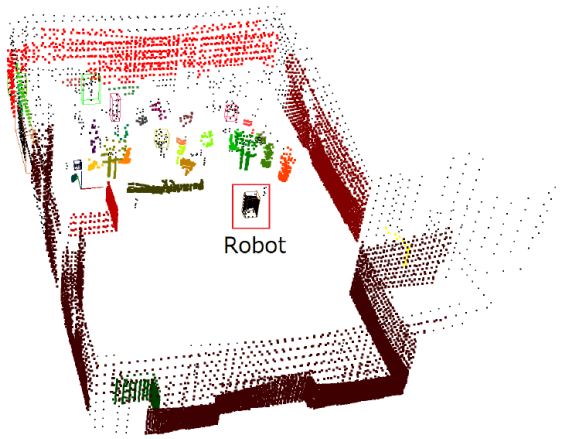

図7　Euclid Clustering Extraction実行後

2.2.4　運搬ロボットについて

本実証実験で用いた運搬ロボットはアンドロボティクス株式会社のFRUTERA A-ROBOT-102である。以降、後の2.3節にて行うシミュレーションでは本ロボットの特徴を踏まえ、独自に条件を設定した仮想ロボットを用いる。

2.3　実験

今回使用する仮想ロボットのベースとなるA-ROBOT-102はロボットは自身の先端下部に装着しているLiDARセンサを用いて、事前にマッピングを行う(SLAMを行っていると思われる)。

そしてそのマッピングしたエリアの中からいくつかの目的地とホームポジションを指定する。移動の際にはそれらの目的地を指定し、ロボットがそこまでの最短経路を内部で求め、移動を開始する。その際に、障害物(人など)に接近した場合、正面のLiDARセンサが物体を捉えることで、障害物に接近したと判断し、一時動作を停止する。また、本ロボットはオドメトリを利用した自己位置推定を行っていると思われ、車輪が空転した場合などを除き、少量のノイズを含みつつ現在地をユーザに伝達可能である。

以上をまとめ、仮想ロボットの条件として、以下の項目を設定した。

- 一定のノイズを含む自己位置(平面位置、角度)をアプリケーションに伝達可能
- 障害物に対して距離0.5[*m*]以内には接近不可
- 最大速度を1.2[*m/s*]、加速度を0.3[m/s^2]と仮定
- 座標を指定し、その場所に直線的に移動することが出来ると仮定

上記の条件の内、最大速度については、実証実験で用い

＊7　https://docs.unity3d.com/2022.1/Documentation/Manual/Navigation.html

ている運搬ロボットの仕様として記載されていたものを使用した。実証実験で用いている運搬ロボットの移動機能は、事前に設定した座標に向かい移動するものであり、上記条件に設定したような自在に座標を指定する柔軟性はない。しかし現在、数多くのロボットは方向と移動量や移動時間を指定し、移動するといった操作が可能である。そのため、本実験では、仮想ロボットは自由に移動できるという仮定の下シミュレーションを実施する。ただし、実証実験で用いている運搬ロボットと同様に座標指定を行い、そこに対し直線的に移動するものとする。

シミュレータからロボットや障害物を正しく認知し、ロボットの行動計画を行えるかをテストするため、以下の実験を行った。図8には本実験にて仮想ロボットが移動を開始する移動開始地点と目的地を示している。移動開始地点は図8上にて、緑色の立方体で表されており、目的地は赤色の立方体で表されている。

複数のLiDARは中間処理層にLiDARが取得した全ての点群を送信し、中間処理層にてロボットや周辺の障害物などを分離、認知する。それをデジタルツインアプリケーションに送信し、デジタルツインアプリケーションがロボットの行動計画を行い、経路をロボットに伝送する。その時の様子を図9に示した。この図ではロボットのツインが与えられた情報を元に赤い経路を生成している様子が示されている。この図でロボットのモデルは推定された位置情報に位置するように配置されている。また、この図にて黒い点の集合体が障害物として認知された点群であり、床に生成された青いエリアはロボットが移動可能(障害物から0.5m]離れている場所)なエリアである。

図10には、デジタルツインアプリケーションによって生成された行動計画を遂行したシミュレータ上でのロボットが映っている。図の右上には本ロボットがデジタルツインアプリケーションから送信された移動命令(座標情報)が表示されている。

2.4　評価

シミュレータ上の実際の仮想ロボットの位置と中間処理を終え、提案手法により予測された仮想ロボットの位置の誤差平均を計測する。ここで計測は仮想ロボットがスタートしてからゴールするまで行い、毎フレーム(今回の実験では60fps)情報を取得するものとする。ここで、図8に表示されている仮想空間上にて、左方向がX軸正、下方向がZ軸正である。

なお、ここで各軸毎の平均誤差zはロボットの座標の真の値をxk (ここで、kはシミュレーション開始時から終了時までに取得したデータ(n個)を順番に並べた際の添え字)とし、予測されたロボットの座標をyとすると

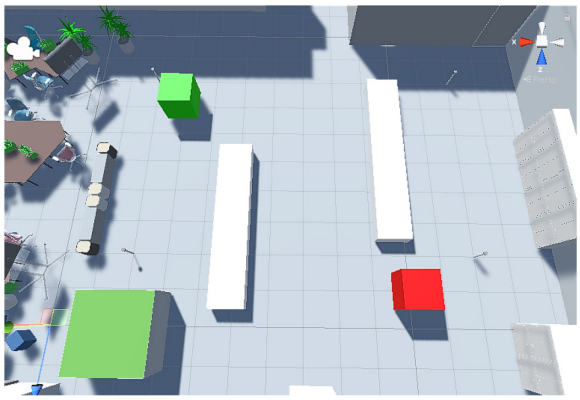

図8　シミュレータ上でのスタート(緑)とゴール(赤)と障害物の配置

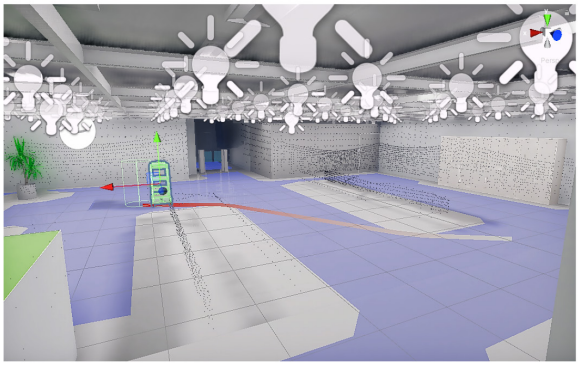

図9　生成された行動計画

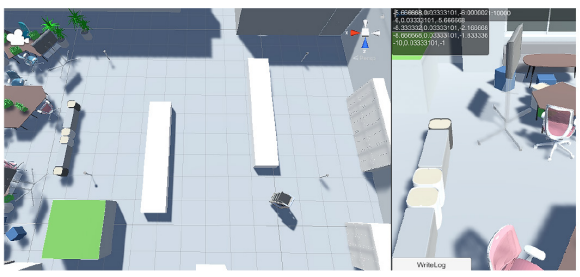

図10　生成された行動計画

$$z = \frac{\sum_{k=1}^{n} |x_k - y_k|}{n}$$

としてあらわされ、標準偏差は上記平均誤差zの標準偏差を取得するものとする。

最後にX軸、Z軸双方を考慮した誤差はX軸とZ軸の差分のユークリッド距離を利用する。

表1に本実験を実施した実行環境を、表2に計測されたデータを示した。表2を見ると、全体的に200[*mm*]ほどのずれが生じており、誤差が大きいと考えられる。ロボットの横幅は約460[*mm*]であるため、ロボットの半分ほどのずれが生じている。また、X軸方向の標準偏差は527[*mm*]、全体の標準偏差は273.43[*mm*]とかなり大きい。これは、今回の走行経路において、障害物がセンサからロボットを遮蔽している場所を考慮すると、ロボットを追跡できるセンサが限られているからであると思われる。

この結果を踏まえると、今回の手法を用いたプログラムの実行速度が遅く、その影響がこの結果に表されていると考えられる。

そのため、新たにシミュレーション中に各フレームごとの実行時間を測定し、その時間だけ予測値の遷移をずらし、それらと真値の差分を計測した。表3に、新たに取得したデータを示した。

これを参照すると表2と比較し、かなり誤差が減っていることがわかる。全体誤差を参照すると約85[*mm*]のずれが存在し、全体の標準偏差を参照すると93[*mm*]ほどの値が確認できるが、これは流動体などの接近時にある程度のマージンを保持し、接近時には停止する設定を持つ運搬ロボットにおいて、500[*mm*]ほどのマージンを担保することが多いため、それと比較すると少ないものではないかと思われる。

この結果から、本手法では高精度のLiDARを活用し、運搬ロボットの動的な行動計画に対応することができる可能性を示した。しかし、上記結果の通り、この手法を現実的な物とするには、特に大きい影響を誇る実行時間の削減を目指さなければいけないということがわかった。

第３章　生産管理のためのデジタルツインアプリケーション

デジタルツインのユースケースの一つとして、工場での使用が考えられる。工場では工作機械や人の流れ、作業

表1　検証システムの構成

	OS	CPU
PC	Windows 10 Pro	Intel ® Core™ i9-10850K CPU 3.60GHz

	GPU	メモリ	ソフトウェア
PC	NVIDIA GeForce RTX 3080	32.0 GB	Unity2020.3.18f1

表2　平均誤差

	平均誤差 [mm]	標準偏差 [mm]
X	240.65	527.00
Z	201.33	190.34
全体	328.21	273.43

表3　平均誤差(時間調整後)

	平均誤差 [mm]	標準偏差 [mm]
X	67.02	125.56
Z	44.10	76.91
全体	85.01	93.26

効率など様々な要素が可視化され、管理されることが望まれるが、現在、様々な企業がスマートファクトリを推進しているように、実際にデジタルツインを構築し、活用している工場はまだ多くなく、普及段階である。ここで、スマートファクトリーのためのデジタルツインを考えた際、工場の効率化のために何を重点的に可視化、フィードバックするかによって効率であるか否かが変わってくる。

我々は、上記の課題に対して、電力消費量の管理や物理空間の工場の稼働状況の可視化という観点から、生産管理のためのデジタルツインアプリケーション(図12)、及び工場に設置された工作機械やIoTデバイスなどの情報を集積しアプリケーションに提供するためのAPIを作成した[143]。

本章では、本デジタルツインアプリケーションの概要、評価として実施した利用評価アンケートについて、報告する。

3.1 提案手法

本システムでは、3Dビューワを搭載し、静的なジオメトリデータ上に工作機械の電力消費量から推定した稼働状況やパトランプによる異常状態、IoTデバイスに搭載されたセンサによって取得された温度情報や騒音情報などの環境状態の投影を行うことでリアルタイムに工場の状況を直感的に認識できるようにしている。ここで、静的なジオメトリデータは、ジオメトリデータ上の工作機械の位置などのメタ情報を付加することで、BIMのサブセットとして利用する。また本研究では工場の効率化のために、電力状況について着目した。工場内の工作機械の電力消費量、稼働状況、パトランプに関する情報などを集積し、可視化するシステムを提案する。工作機械から蓄積したデータは、電力消費量を棒グラフを、稼働状況をガントチャートを作成することで、過去のデータから工作機械の利用状況を可視化する。過去のデータについては、年、月、日といった時間に関する基準やエリアやグループなどと言った工作機械の所属するドメインによってまとめられており、データはこれらの項目によってカスタマイズされたものを表示する。そして、利用者の作業効率への影響という観点から、工場に設置した工作機械やIoTデバイスの情報を反映する速度を速めることで、工場内の情報共有や緊急性の高い事象を逃さないようにリアルタイム性を高めることを目指した。

今回実装したアプリケーションの構成を図11にアプリ

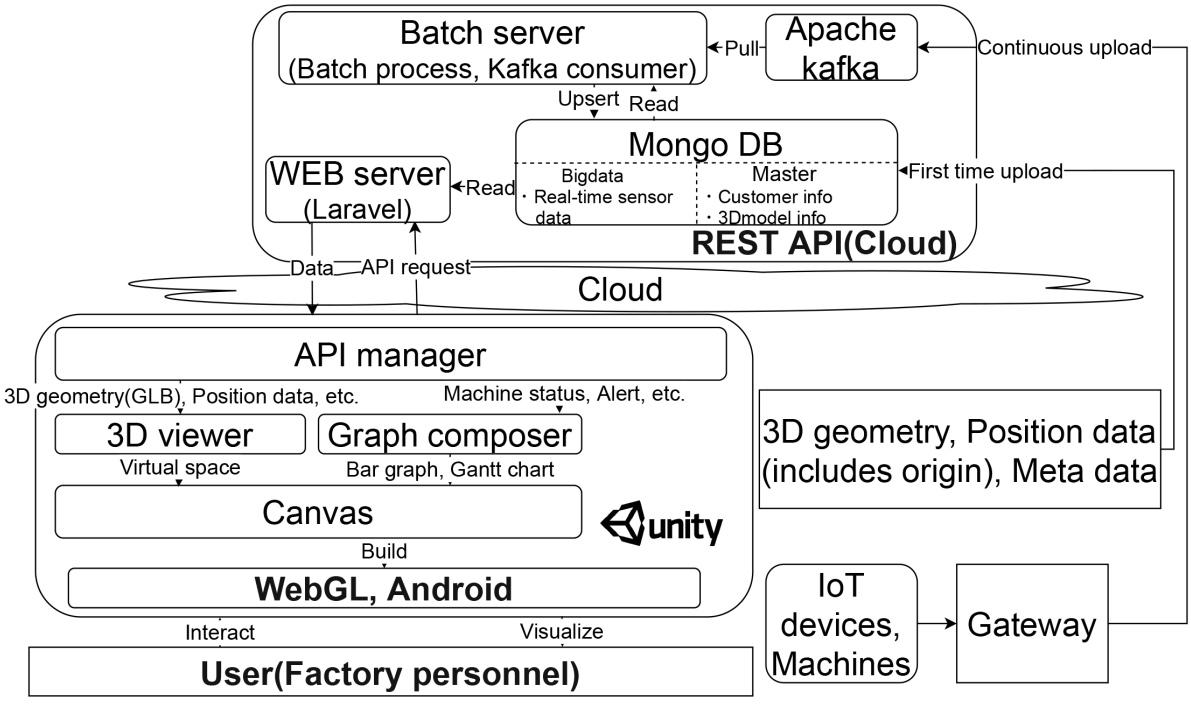

図11　アプリケーション構成

ケーションのスクリーンショットを図12に示した。本実装では、バックエンドとフロントエンドに分けている。バックエンドは、ジオメトリデータを格納し、IoTゲートウェイから取得した工作機械やIoTデバイスからの情報をリアルタイムに取得、集積、カスタマイズを行うクラウドサーバである。フロントエンドでは、ゲームエンジンであるUnity3Dを用いて、ジオメトリデータを用いたサイバー空間内でのシミュレーションや3Dでの可視化、各種グラフの表示を行う。

なお、工場をテーマにした上記デジタルツインアプリケーションでは、株式会社シムックスイニシアティブ*8との共同研究という形で研究を行った。具体的には、ジオメトリデータを作成する工場の選定、バックエンドの作成、アプリケーションデザインや一部機能の実装補助などを株式会社シムックスイニシアティブが行った。

3.2　設計と実装

3.2.1　アプリケーション

本アプリケーションの構築にはゲームエンジンであるUnity3Dを利用した。利用するデータは全て、後述するクラウドサーバから取得し、ここでは、3Dビューワ内での可視化、グラフ作成のためのデータとして利用している。なお、取得しているデータは以下のとおりである。

- ジオメトリデータ
- IoTデバイスからのセンシングデータ(温度、湿度、照度、騒音)
- LiDARセンサからの点群データ
- 工作機械のパトランプ
- 推定された工作機械の状態
- 工作機械の消費電力
- 顧客情報

また、IoTデバイスや工作機械ではNTPとの時刻同期機能を利用しており、データを取得するタイミングにて正確なタイムスタンプを保存しており、アプリケーション側にてデータを利用する際にはこれらも利用可能である。なお、基本的にデータ改変を行うことでCPUリソースを使わないために、集積されたデータのカスタマイズについては後述するクラウドサーバ上で行っている。図12では、3Dビューワにて工作機械の稼働状況、電力消費量などを閲覧している。工作機械の稼働状況は、対応した色を割り当てており、利用者が一目で工場全体の工作機械の稼働状況を把握できるようにしている。また、パトランプに関する情報も取得しており、図12の画面上部にあるタブを稼働から警告に変更することで、工場内全ての工作機械のパトランプの情報を取得し、画面に反映する。また、図12に表示されているように、工作機械にマウスカーソルを合わせることで、工作機械に関する詳細情報(工作機械の名称、機械種別、稼働状況、消費電力、パトランプの状態)をポップアップにて表示する。工場内はいくつかのエリアに分けることが可能で、広い工場の中で目的のエリアだけを中止したい場合は、図13に示されているようにエリア別に3Dビューワーを利用することができる。また、エリアでのグループ分け以外にも機械種別ごとにグループ分けを行うことも出来る。これらのグループ分けは利用者が事前に決めることになる。

図12　アプリケーションのスクリーンショット

図13　エリア別の強調表示機能

＊8　https://www.cimx-initiative.com/

工場の3Dジオメトリには、テキスチャが付与されている場合、そのテキスチャも表示されるが、その場合、工作機械を示すオブジェクトが見にくい場合がある。そのため、図14に示しているように、3Dジオメトリに割り当てられているテキスチャを単色かつ透過したものに変更する機能がある。これにより、視覚的に工作機械が捉えやすくなっている。そして、図14では、IoTデバイスから取得した環境情報をヒートマップとして3Dジオメトリの上にオーバーレイする機能も搭載しており、環境情報を数値だけでなく、可視化することを行っている。図15では、工作機械の稼働状況をガントチャートを用いて時間別に表示している。図16では、工作機械の電力消費量を稼働状況による積み上げ棒グラフにて表示している。ここでは、3Dビューワーをリアルタイムに情報を把握するための機能として、グラフは過去や現在の履歴データから工場の状態変化を示すための機能である。

3.3　評価

今回、このアプリケーションやクラウドサーバを含めた全体のシステムを評価するにあたって、主観評価としてユーザアンケートを実施した。この節では、本アンケートを用いた定性的な評価を行う。なお、今回のユーザアンケートで使用したアプリケーションはブラウザで利用するWebGL版とした。このアンケートはリッカート尺度に基づき、アプリケーションの内容について、8つの指標を用意し、それぞれ**悪い**から**良い**までの五段階評価で回答してもらった。

3.3.1　質問項目

アンケートでは、リッカート尺度に基づき、以下8つの質問に回答してもらった。

Q1 3Dモデルの再現度に満足がいきましたか？
Q2 実際の工場（3Dモデル）に配置された工作機械と同じ場所にオブジェクトは存在したように感じましたか？
Q3 現在の工場の稼働状態を把握することができましたか？
Q4 過去の工場の稼働状態を把握することができましたか？
Q5 操作は簡単にできましたか？
Q6 3Dによる可視化は直感的でしたか？
Q7 グラフ機能による、稼働状況の可視化はわかりやすいですか？
Q8 グラフ機能は工場業務に役に立ちそうですか？

各設問の意図として、まず設問Q1、Q2では、映像の再現性について問いた。現実の工場のツインとして、仮想空間上に工場のジオメトリデータと共に各種工作機械、デバイスのオブジェクトを反映させているため、そこに違和感を感じたかを問いている。

設問Q1では、Matterportを用い作成したジオメトリデー

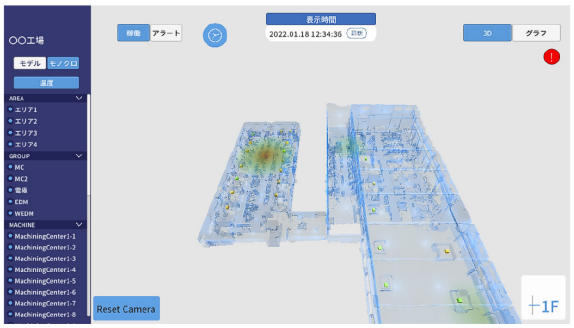

図14　テキスチャ透過、ヒートマップ表示機能

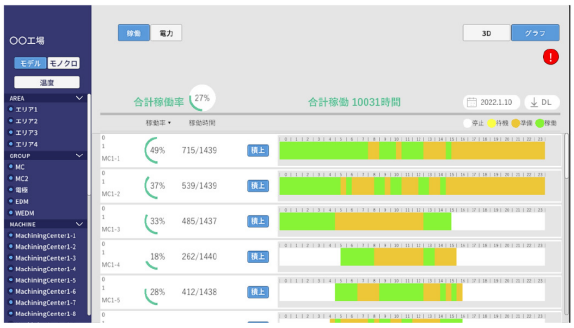

図15　工作機械の稼働状況を示したガントチャート

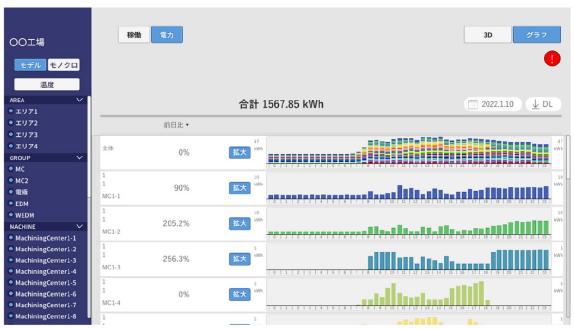

図16　工作機械の電力消費量を示した棒グラフ

タに違和感を感じるか、設問Q2では、工作機械やデバイスを示す立方体上のオブジェクトの位置がジオメトリデータ上の工作機械の位置からずれていないか、違和感を感じないかを問いている。設問Q3、Q4では、指定した操作を正しく行い、本アプリケーションの目的である現在、過去の工場の稼働状況の把握を行うことが出来たかを問いている。

設問Q5では、稼働状況把握などの目的を達成するための操作において、その容易さを問いている。設問Q6では、本アンケートの主目的である、三次元空間上に工場のツインを再現し、可視化したことについて、その効果を問いている。設問Q7、Q8では、グラフ機能について問いている。設問Q7では、グラフによる稼働状況の把握のしやすさ、設問Q8では、その利用可能性について問いている。

また、アンケートでは、リッカート尺度に基づいた8つの質問の他に、以下の補足情報についての回答項目を用意した。

- 回答者の性別
- 回答者の年齢
- 工場業務に関する専門性
- 本アプリケーションを使用するにあたって使用したデバイスの種類(名称)
- 本アプリケーションを使用するにあたって使用したデバイスのOS名
- 本アプリケーションを使用するにあたって使用したブラウザ名
- アプリケーションを使用した際のネットワーク環境
- 本アプリケーション内にてサンプルとして使用している工場へ行ったことがあるかどうか
- 感想・要望
- 不具合報告

3.3.2　アンケートの回答方法、操作手順の指定

回答者には、本アプリケーションがブラウザ上で操作できるURLとアンケートを回答するためのウェブフォーム(Google Formsを利用)のURLを提供した。なお、アンケートの説明書きの中にアプリケーションの操作方法を説明したプレゼンテーション資料や同じく操作方法を説明した動画ファイルへを参照するためのURLを記載したため、回答者は必要に応じてこれらの事前知識を得ることが出来た。ここでは、回答者に実行してもらう操作として、3D空間の閲覧操作、各工作機械の稼働状態の表示、過去の工作機械の状況を知れるタイムラプス機能の使用、各種グラフの表示を指定した。

3.3.3　アンケート実施時の状況

アンケートでは、共同研究先である株式会社シムックスイニシアティブの中島が登壇した寺小屋というセミナーの参加者に対して実施した。ここで、参加者は約50名であった。アンケートの実施までの具体的な内容としては、中島が2021年11月10日に開催されたセミナーにて、アンケートの説明を行い、先述した本アプリケーションがブラウザ上で操作できるURLとアンケートを回答するためのウェブフォームのURLを提供した。アンケートの回答期限は2021年11月10日から2021年12月1日までとした。回答者は期間中、自由に上記URLを参照でき、自由にアプリケーションを触れるものとした。また期間中、アプリケーションを用意しているサーバも同時に複数人がアクセスして良いように増強を行ってからアンケートを実施した。

3.3.4　アンケート結果

約50名の参加者の内、回答者は男性22名、女性1名であった。年齢構成としては、20代 2名、30代 3名、40代 8名、50代 8名、60以上 2名であった。また、工場業務の専門性に対する解答として専門家 6名、非専門家 15名、無回答 2名であった。参加者は工場関係者のみならず、ビジネス的な観点を持ち参加する人が多かったと思われる。

アンケートの調査結果を図17に示す。グラフの横軸は1から5までの5段階の回答比率を表し、合計値は100%である。中間評価である3の回答比率の中心を横軸の原点0%に置く。横軸の正の方向には高評価である4と5を、負の方向には低評価である1と2を置いている。これは、高評価の回答数が低評価の回答数より多い場合、グラフが正の方向に偏り、その逆の場合、負の方向に偏ることを示す。

図17を参照すると、アンケート結果は殆どの設問で概ね

高評価を得られた。また、アンケートの感想や要望の項にて受け取った意見を確認すると、稼働状況の可視化がリアルであったといった意見や温度マップといった環境に関する情報を三次元空間上に投影することは分かりやすいといった意見があった。これは、本手法にて作成した三次元空間上に工場の稼働状況を再現するといった手法が一定の効能を持つことを示していると言える。

しかし、設問Q5については、他の設問と違い、低評価の方が顕著である。これは、操作性の項目であるが、これはアンケートに用意した感想・要望の項を確認すると、操作が難しい、望んだ工作機械に対して思うようにズームすることが出来なかったといった意見があった。これは、本アプリケーションでは、工場の中心点(全体のメッシュデータの端を計算し、その中点を求める)を中心として回転する設計になっており、上下左右回転といった自由移動を限定しているためにあると想定される。

被験者の他の意見として、生産業務において3Dである必要性は感じなかったと言った意見があった。実際、各企業が展開している工場監視ツールやIoTデバイス管理ツールは二次元上にマッピングされた監視ツールが多い。ただ、本意見の続きとして、新入社員などが工場管理を行う場合や設備数が多く、設備番号で直感的にどこの設備だと分からない場合には有効かも知れないといった意見や温度マップなど環境に関することについては3Dマップのメリットを感じたといった意見に続けており、本アプリケーションの監視だけではない目的というものが提示された。

本アプリケーションは、現在、温度マップの表示のみ行っており、湿度マップや騒音マップなどは作成していない。そのため、今後はそういった3D空間をフルに活用することが出来るヒートマップ機能の拡張や3D空間を活用した新たな機能の開発が求められる。また、対象とした工場内にて、立体的に監視対象のオブジェクトが配置されている場合、例えば、スペースを有効活用するべく、立体的に荷物を置いていることが多い、倉庫や物置などについては3Dとしての恩恵が強いのではないかと思われる。

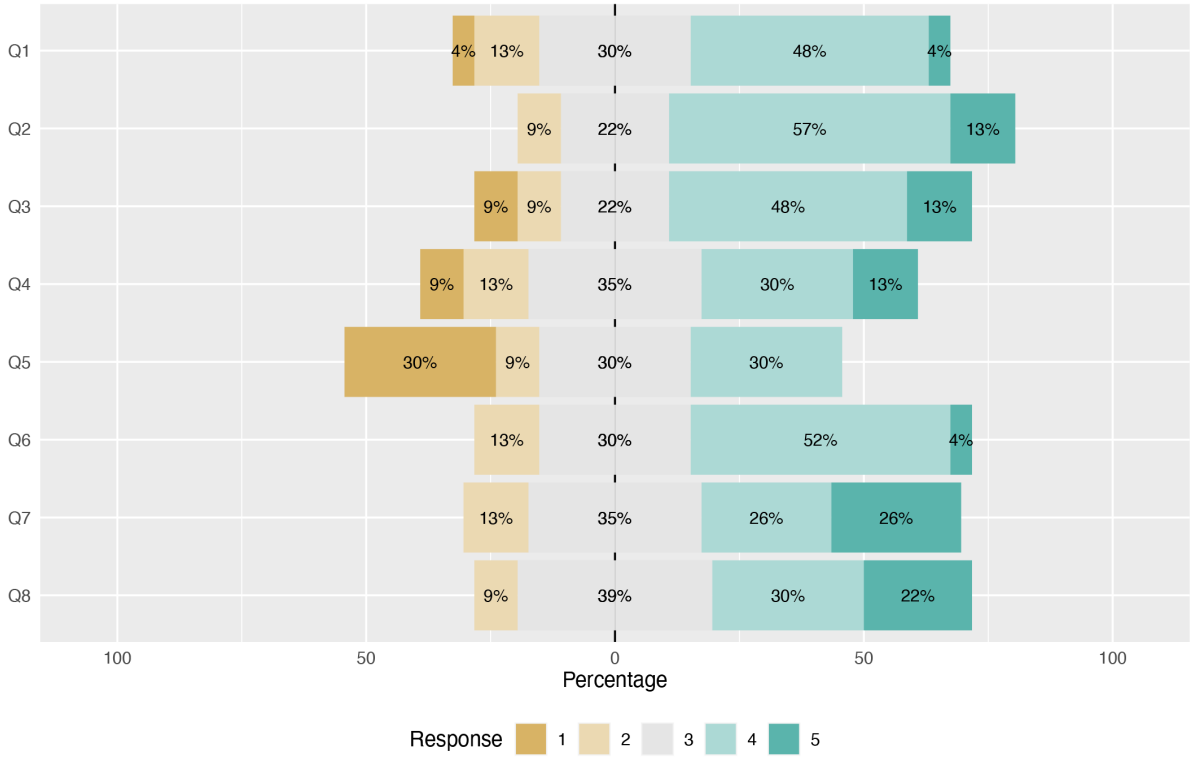

図17　アンケート結果のリッカート尺度

3.4 要求性能評価

3.4.1 ネットワーク性能

本アプリケーションを実行するにあたって本アプリケーションを継続的に利用するために必要なネットワーク性能を評価する。

本アプリケーション本体のデータは約9[*MB*]であり、これは初回起動時にダウンロードされたのち、Unity3Dの機能によりその大部分がブラウザ上にキャッシングさせることが出来る。また、本アプリケーションでは、ジオメトリデータはキャッシングを利用し、初回起動時以外には通信を圧迫しないため、ジオメトリデータも本アプリケーションを継続的に利用するためには、重視されない。そのため、ここでは、アプリケーション内でのユーザの操作によってリアルタイムに発生するサーバとの通信データを見ていく。

まず、工作機械やIoTデバイスなどからのセンサ値を取得する操作を行った場合にサーバへ通信を行うが、これは一つ当たり約800[*B*]であり、極めて軽量である。

次に、本アプリケーションが定期的に行う通信として、工作機械全体の状態把握(三次元空間上にて工作機械を表す立方体の色分けや警告通知に使用)がある。これは、全ての工作機械の情報を一度のリクエストで取得するが、データ容量は約5.5[*KB*]であり、これもまた軽量であると言える。他にも様々な通信(ログイン情報や利用者情報など)があるが、これらは全て1リクエスト当たり10[*KB*]未満である。

次にグラフ機能を使用した際に発生する、全ての工作機械の詳細データを取得した際のデータ容量を見ていく。グラフ機能は、日、月、年ごとに各工作機械の情報を閲覧できるが、これらは事前にサーバ側で計算されたデータを表示しているため、そのデータ容量は約33[*KB*]と小さい。

最後に、タイムラプス機能を使用した際に発生する、全ての工作機械の詳細データを取得した際のデータ容量を見ていく。まず、タイムラプス機能では、連続した日付と時間を指定し、過去の状態を閲覧することが出来る機能である。この機能では、指定した期間内の1分置きのデータを取得するため、通信時に発生するデータ通信量が非常に大きい。そのデータ容量は期間を1日に設定した際に、約3.6*MB*である。この機能では、何日間でも指定できるため、その期間を伸ばしていくたびに約3.6*MB*の通信量が1日ごとに発生する。

以上から、本アプリケーションを継続的に利用する場合、タイムラプス機能が特にボトルネックとなり、例えば1か月間のデータを一気にタイムラプス機能で見ようとした際には、約108[*MB*]ほどの通信を行うため、データダウンロードに10秒ほど待ったとしても約86.4[*Mbps*]の高速通信を要求されてしまうという結果となった。

なお、過去の発表[143]では、LiDARセンサからの点群の仮想空間上へのマッピングも行っていたが、点群データは容量が大きく、帯域を圧迫してしまうため、今回のアプリケーションでは機能を削除した。

第４章　WebRTCを用いた宅内環境間P2Pリアルタイム通信性能計測

4.1 WebRTC計測概要

新型コロナウイルスが世界で流行し、多くの人々がリモートワークをするようになり、遠隔コラボレーションをする機会が増えた。例えば、かつて対面で行っていた会議は、ZoomなどのWeb会議アプリを使って、遠隔で行う頻度が増えた。しかし、音楽の合奏などの、高品質な同期が必要な通信では、Zoomのようなサーバー・クライアント通信を使ったWeb会議アプリでは遅延が大きく、満足のいく活動ができない。この問題に対応するため、宅内ネットワーク同士をP2Pで結ぶことで、遅延を減らすアプリケーションが登場している。例えば、YamahaのSyncroomは、宅内ネットワーク同士をP2Pでメッシュ接続することで、最大五人でリモート合奏をすることができるアプリケーションである。しかし、このアプリケーションはネットワークの性能によってはうまく動かないことも多く、広く普及するに至っていない。普及に先駆けて、宅内ネットワーク間の性能計測を広く行い、実態を把握することが重要である。しかし、宅内ネットワー

ク間の性能計測を行っている先行研究はほとんどなく、計測する手法すら確立されていない。そこで、本研究ではWebRTCを用いて宅内環境を結ぶP2P性能を容易に計測する手法を提案する。WebRTCとはWeb Real-Time Communicationsの略で、ブラウザやモバイルアプリに簡潔なAPIを通じてリアルタイム通信を提供するオープンソースの技術である。

本研究の貢献は、専用ハードや専用ソフトを使うことなく、WebRTCにより宅内ネットワーク間の性能を簡単に計測するツールを開発し、実際のインターネットで計測した結果を報告することである。

4.2 要求事項

本研究で提案する手法の機能要件は以下の5つである。

計測の容易さ

RIPE AtlasやXbox 360を使用した計測は、専用ハードウェアに依存するため、多様な環境下での計測が困難である。専用ソフトウェアを用いた計測は専用ハードウェアを用いた計測よりも要件は緩いが、それでもインストール作業が必要となってくるため、依然として一定の困難がある。専用ハードウェアや専用ソフトウェアをインストールすることなく、多くのPCにすでにインストールされているソフトウェア上で計測できれば、上述した問題を解消でき、だれでも簡単に計測に参加できる。

P2P networkネットワークの分析

YamahaのSyncroomのようなアプリケーションは、複数の宅内環境をメッシュで接続することで多人数のコラボレーションを実現している。実際の利用環境と近い状態で計測するためには、多ノードをメッシュで接続した環境での計測が必要である。また、一般的な宅内環境ではNATを用いてインターネットに接続しているため、NAT環境での計測に対応する必要がある。

直感的な可視化

多ノードフルメッシュ接続の統計情報は、生データのままでは直感的に評価できない。計測をP2P通信性能面での問題解決に役立てるという本来の趣旨に結び付けるためには、データ可視化手法が不可欠である。今回は、メッシュ全体のP2P接続のRTTの傾向について直感的に把握し、問題解決へとつなげやすい可視化を目指した。

資源の制約

旧式のノートパソコンの性能でもでも動作するようなツールを作れば、可用性が高まり、より多様な宅内環境での計測に対応できる。またRaspberry Piのようなシングルボードコンピュータで動作するようなツールを作れば、宅内環境のクライアントとは別に、静的に常時接続をし続けるクライアントを配置して計測に参加させることが可能となる。本研究では可能な限り多様な環境で計測したり、Raspberry Piでも動作するような軽い計測ツールを作成することを目指した。

計測の粒度

本研究では、60ms以下の遅延を計測することを目指す。本研究では、合奏などの用途を想定したP2Pの性能評価を行うことを目指している。[144]では、問題なく合奏を行うことが可能なネットワーク遅延は30ms程度とされていることから、上記の目標を達成するには片道30ms、往復60ms程度の遅延を計測することが可能な計測手段を採用する必要がある。

これらの機能要件を満たすことができれば、従来よりも多様な環境下で、専用のハードウェアやソフトウェアを導入することなく、ユーザー主体でP2Pメッシュ接続環境下でRTTを計測をできる。

4.3 提案手法

本研究では、WebRTCを用いた宅内環境間P2P接続におけるRTTを計測する手法を提案する。WebRTCはNATを越えたP2P通信をサポートしており、かつgetStats ()というメソッドを使用すればRTTなどの統計情報も取得することができる。また、多くのPCにインストールされているGoogle Chromeなどのブラウザがあれば使用可能である。したがって、WebRTCを使った計測を行うことで、本研究が目指すEase of Measurementと Comprehensive analysis of P2P networkを実現することが可能である。

本研究のデザインを図18に示す。本研究では、各宅内ネットワークにあるPCからGoogle Chromeを用いて計測用

Webサイトにアクセスを行い、JavaScriptの計測プログラムをクライアント上で実行する。計測プログラムの動作は以下の三段階に大別される。

- Signaling
- P2P Connect
- Collect Data and Visualize

Signalingでは、まずNATを超えたP2P接続に必要なSession Description Protocol（SDP）を、計測サイトに接続している他のクライアントとサーバーを通じて交換する。Signalingが完了すると、P2P Connectが行われる。P2P Connectでは、WebRTCによるNATを超えたP2P接続が開始され、設定されたサイズ・間隔でデータを送信する。P2P Connectが行われている間、Collect Data and Visualizeを行う。Collect Data and Visualizeでは、WebRTCの機能でP2P接続のRTTを取得し続ける。取得したRTTは、フロントエンドでリアルタイムに可視化されるほか、データ収集サーバーにもPOSTされ、データベースで永続化される。データベースで永続化されたデータに対し、計測終了後にヒートマップによる可視化を行う。これによって、本研究のReqquirementであるIntuitive Visualizationを目指す。

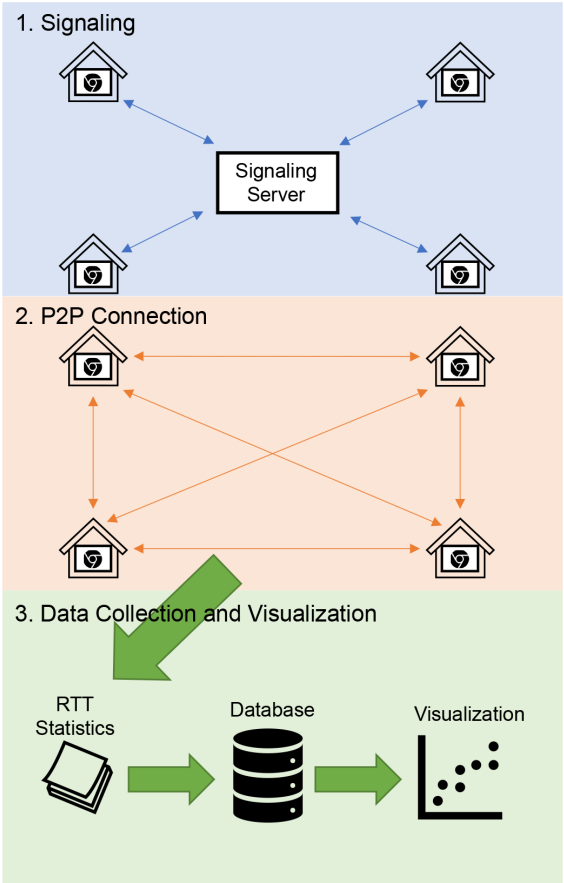

図18　設計

4.4　実装

本研究の実装を図19に示す。本手法の計測プログラムはクライアントのブラウザ上で実行されるJavaScriptファイルである。まず計測URLにアクセスすると、Signalingが行われる。Signalingでは、シグナリングサーバーとWebSocketを用いた通信が確立される。WebSocket通信が確立されると、クライアントはSDPをシグナリングサーバーを介して同じく計測サイトにアクセスしている他のクライアントと交換する。Signalingが完了した後、WebサイトのConnnectボタンを押すと、P2P Connectが行われ、他のクライアントとのWebRTCによるNATを越えたP2P接続が確立される。

そして、ある条件のデータサイズ・送信頻度で、他のクライアントにデータ送信が開始される。この条件は時刻によって変化する。P2P Connectが行われている間、Collect Data and Visualizeが行われる。Collect Data and Visualizeでは、1秒間隔でRTT・現在のUnixTime・データ送信条件が取得される。この統計情報はChart.js用いてクライアント上でリアルタイムに可視化されるほか、JSON形式でデータ収集サーバーにPOSTされる。POSTされるJSONに格納されるデータは

- Date: ブラウザ上で取得された日付[*ms*]
- yourIsp: ノードが接続している回線のISP
- candidatePair_RTT: WebRTCによるP2P接続のRTT[*ms*]
- currentSendBytes: 実験条件（送信バイト数）
- currentSendTime: 実験条件（送信間隔）

である。サーバーはFlaskによるAPIサーバーとSQLiteによるデータベースから成り、POSTされたJSONはAPIサーバーを通じてデータベースに登録され、永続化される。サーバー上のデータベースで永続化された統計情報に対し、計測終了後にサーバー上で分析・可視化が行われる。

1つのクライアントが取得可能な統計情報は、自分と他のクライアントとのP2P接続の情報のみだが、サーバーでそれらを収集することにより、サーバー上でフルメッシュでデータ分析を行うことが可能になる。

サーバーに統計情報を送信する際に、WebRTC接続のRTT以外に、計測開始時にipinfo.ioにリクエストを送信して得られたグローバルIPアドレスやISP等の情報も送信する。これによって、IPアドレスやISPの差異とRTTに関する議論なども行うことが可能となる。

サーバー上での可視化にはヒートマップや有向グラフ等の手法を用いた。これにより、メッシュの統計情報を直感的に理解することをめざした。

4.5 WebRTC計測値の信頼性評価

WebRTCの通信プロトコルは[145]で「SCTP over DTLS over UDP」と規定されており、これはRTTを計測する従来の手法、例えばPingとは大きく異なっている。そこで、本研究ではWebRTCのRTT計測値と、Ping計測値を比較し、WebRTC計測値にどのような特徴が見られるかを確認する予備実験を行った。

実験環境は表4に示す。tcコマンドで0ms、10ms、100msの定常遅延を発生した際の結果は表5となった。tcコマンドで発生させた遅延のオーダーに関わらず、WebRTCの方が平均で1-3ms計測値が大きくなるという結果であった。これはWebRTCの処理遅延であると考えられる。

また、tcコマンドで30msの定常遅延、30msのjitterを発生させた場合のboxplotを図20に示す。jitterを発生させた場合では、すべての四分位点においてWebRTCの方が0-3ms計測値が大きくなるという結果で、これもWebRTCの処理遅延によるものであると考えられる。

表4　WebRTC Ping比較実験

使用端末	from Ubuntu 18.04 PC to Raspberry Pi 4(2GB)
接続	1 ホップで Ethernet 接続
Web ブラウザ	Google Chrome(Ubuntu) Chromium(Raspberry Pi)
計測内容	tc コマンドの人工遅延を WebRTC と Ping で計測
計測回数	WebRTC,Ping で 1 秒おきに 500 回計測値を取得

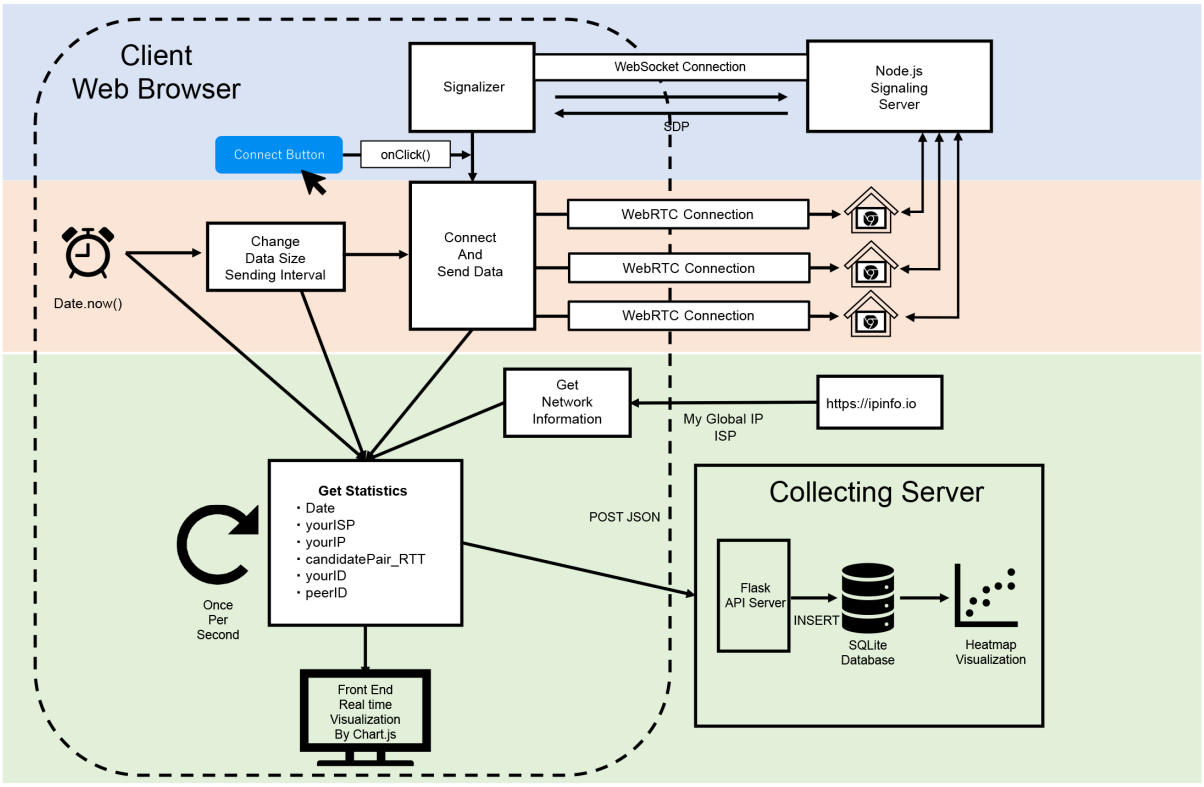

図19　実装

これらの結果から、WebRTCによる計測はPingと比較して定常的な処理遅延はあるものの、[146]で主張されている合奏を行う上での限界遅延50msを計測するのに十分な計測値を得ることが可能であると考えられる。これによって、本研究のRequirementのGranularity of measurementは、WebRTCを用いた計測によって満たされていることが分かった。

4.6 実験

このツールを使用して、実際に宅内環境を結ぶ計測実験を行った。実験参加者には、実際に各家庭・組織のLAN環境内にあるノートPCからGoogle Chromeで計測サイトにアクセスしてもらい、計測を実施した。結果を取得することができた計測ノード数は10で、GeoIPによる地理情報では図21に示した通り、東京に7、神奈川に1、千葉に1、岐阜に1ノードという分布となった。計測を行うにあたり、WebRTCのdataChannelでは毎秒100byteずつデータを送信し続け、1秒おきにRTTを取得し、サーバーへの送信・収集を行った。

表5　WebRTC計測値とPing計測値比較

tc command latency	0ms		10ms		100ms	
method	WebRTC	Ping	WebRTC	Ping	WebRTC	Ping
Average RTT[ms]	2.5	0.284	11.686	10.329	101.806	100.343

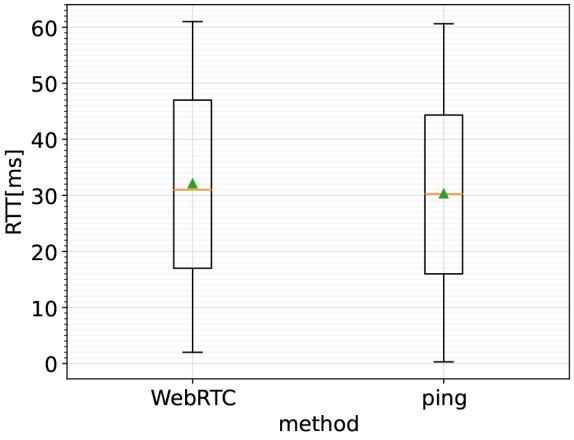

図20　jitter

メッシュ全体から取得されたRTT値群の分布は図22となった。これはRTT値群を10msごとの区間で分けている。0ms-50msの区間では10^3から10^4個の計測値が収集されているが、50ms-100msの区間で減少し、100ms以上の区間では10^2個程度の計測値しか収集されていない。

また、各P2PのPeerConnectionごとに得られたRTT値群の99％点をヒートマップで図示すると図23のような結果となった。このヒートマップの数字は各計測ノードに対応しており、図21の地理的関係とも対応している。なお、50ms以上の計測値は全て赤で表示しているが、そのようなピア数は29個となった。

4.7 計測結果の検討

図23では、1番のノードは東京に所在しているが、同じ東京のノード5番と10番との間に50ms以上の遅延が発生している。他方で、岐阜に所在している6番のノードは東京のノードと50ms未満の遅延で接続されているケースがある。このようなことから、宅内環境間を結ぶP2Pの遅延は、地理的な位置関係よりも宅内環境の問題に依存している可能性が高いと考えられる。これは、東京-岐阜

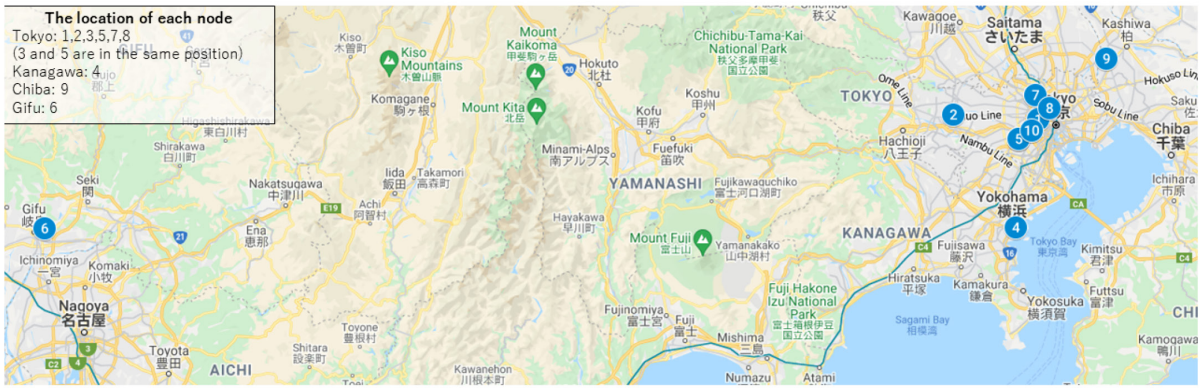

図21　location

間の270kmの距離が、光速で1ms程度しか要さず、光速による遅延が相対的に問題にならないということから考慮しても妥当な結果であると考えられる。

[146]で示されている、ネットワーク上で問題なく計測を行うことが可能な50ms未満の遅延のピアはRTTを取得できた全77ピア中48ピアという結果に終わっており、現在のインターネット環境では必ずしも多くの人がリモート合奏のような高い同期が必要な活動を行うことができるとは言えない状況であると考えられる。また、50ms未満という最低限の要件は達成していても、高いQoEを実現するにはさらに厳しい条件が必要であり、実際に満足のいく活動を行うことができる環境は更に限られてくると考えられる。

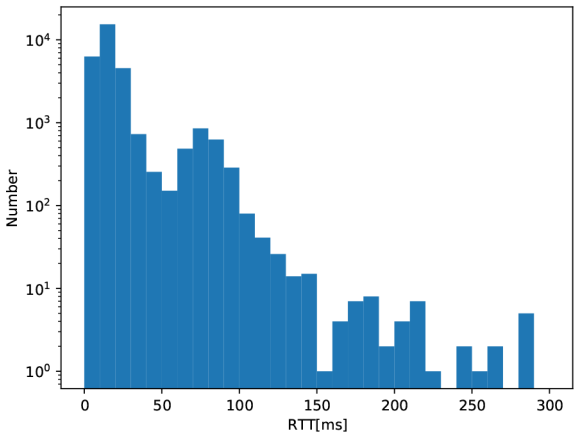

図22　Histogram

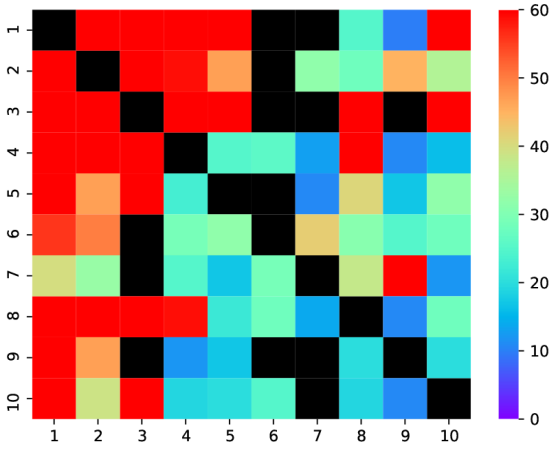

図23　PeerConnection RTT 99% Point[ms]

本研究ではフルメッシュによる接続を目指していたが、実際には全フルメッシュのピア数90通りのうち77ピアしかRTTの計測値を収集できておらず、これはWebRTCによるP2P接続の確立を行うことができなかったものと考えられる。より高い確度で安定的に計測値を取得するには、現在の宅内環境間を直接接続する単純な実装ではなく、TURNサーバーを中間に建てるような実装や、計測用のダミークライアントをあらかじめ配置しておく等の対処が有効であると考えられる。

4.8　結論

本研究では宅内環境間P2P接続のRTTを広範に、かつ簡単に、プラットフォームに依存せず計測できる手法として、WebRTCを用いたブラウザベースで動作する宅内環境間P2P計測ツールを提案・実装した。また、本ツールを使用して、実際に10個のノードに対してメッシュでP2P接続を行いRTTを取得し、可視化を行った。結果として、合奏のような高い同期が必要な活動を行うことが可能なピアは非常に限定されるということが分かった。今後の課題としては、現在のような遠隔環境間を直接P2Pで接続するだけの単純な実装ではなく、TURNやダミークライアント等をネットワーク上に配置したうえで計測を行うことや、地理的に大きく離れたピア接続のRTTを取得することにより、地理的距離がRTTにもたらす影響を検討することなどが挙げられる。

第５章　まとめ

本報告書では、2014年より開始したSDMコンソーシアムで進めて来た、視聴空間サービスのソフトウェア制御による研究で、2021年度に行なった活動について報告した。

SDMコンソーシアムでは、ソフトウェア処理による視聴空間の制御、映像音声を制御するネットワーク機器、インタラクティブ・ユーザインターフェイス、SDMプラットホームを利用したコンテンツ作成など、共同研究活動に参加するパートナーを募集しています。

ご協力をよろしくお願いいたします。