Vol.2016-DPS-167 No.19
Vol.2016-MBL-79 No.19
Vol.2016-ITS-65 No.19
2016/5/27

簡易ロボット遠隔ナビゲーションにおけるネットワーク遅延の影響評価

加藤 由花[1,a)]　田中 麻美子[1]

概要：本稿では，クラウド環境を利用した簡易ロボット遠隔ナビゲーションを対象に，ネットワーク遅延がナビゲーション結果に与える影響について考察する．ここでは，ロボットシミュレータを利用した遠隔ナビゲーション実験を行い，擬似的に遅延を挿入したネットワーク環境において，遠隔ナビゲーションが適切に行われない場合があることを明らかにする．そこから，ネットワーク遅延を考慮した制御モデルの必要性を導き，事前に予測しておいた遅延モデルに従い，観測を遅らせ，制御を早める仕組みを組み込んだ制御モデルを設計する．

1. はじめに

コンシューマ向け低価格ロボットの登場や，ロボットソフトウェアのコンポーネント化 [1][2] を契機に，近年，ロボティクスと ICT の融合が注目されるようになってきた．特に，クラウドロボティクス [3] 等，ネットワークを利用したロボットサービスに関する研究が活発に行われている．

我々もこれまで，クラウドロボティクスの実現に向け，遠隔地に存在する簡易型移動ロボットをインターネット経由で操作するサービスを対象に，クラウド型のサービス構築プラットフォーム [4] および遠隔ナビゲーション機能 [5] について研究を進めてきた．前者は，ソフトウェアプログラマによるロボットプログラム開発を支援するために，ロボットの現実世界での移動や状況変化をプラットフォーム内に隠蔽し，API の呼び出しとリスナによるイベント取得のみで開発を可能にする仕組みを実現したものである．後者は，このプラットフォーム内に隠蔽された遠隔ナビゲーション機能であり，実環境の忠実なモデルを仮想空間として事前に構築しておき，仮想空間と実空間のマッピングにより遠隔ナビゲーションを実現する仕組みである．限定的な環境での評価実験により手法の有効性が検証されているが，一方，クラウド環境での利用を前提としているにも関わらず，ネットワーク遅延の影響は考慮されていないという問題が残されていた．

本稿では，このネットワーク遅延の影響を考察するために，ロボットシミュレータを用いた遠隔ナビゲーション実験を行い，クラウド環境を利用したナビゲーションにおけるネットワーク遅延の影響を明らかにする．さらに，事前に予測しておいた遅延モデルを制御に組み込むことにより，ネットワーク遅延を考慮した遠隔ナビゲーション手法の設計を行う [6]．

一般に，ネットワークを介したロボットの遠隔操作は，移動のためのロボットの知能（計測・制御など）をどこに持たせるかにより 3 つの形態に分類できる．自律度の高いロボットと環境センシングを組み合わせた制御（実環境に知能を持たせる形態），オペレータによるリモコン型の制御（遠隔地のオペレータの知能を利用する形態），そしてクラウド上に知能を持たせる形態である．それぞれ，多様な環境への適用の難しさ，操作性の低さ，ネットワークを介した通信におけるネットワーク遅延・リソース不足等の問題が存在している．本稿で対象とする制御モデルでは，「オペレータ（人間）はネットワーク遅延が存在する環境であっても慣れにより遅延に順応していく」ことに着目し，ネットワーク遅延をモデルに組み込む．

2. サービス構築プラットフォーム

2.1 プラットフォームの概要

まず，本稿で対象とするサービス構築プラットフォーム [4] について説明する．これは，移動ロボットを利用した遠隔地の案内サービスの構築を実現するものであり，遠隔操作はインターネット経由で行う．構築対象となるシステムの構成を **図 1** に示す．システムは実ロボット，インターネット上のサーバ，ユーザがロボットを操作するための仮想空間から構成され，プラットフォームはこのサーバ

[1] 東京女子大学 大学院理学研究科
Graduate School of Science, Tokyo Woman's Christian University, Suginami, Tokyo 167–8585, Japan
[a)] yuka@lab.twcu.ac.jp

Vol.2016-DPS-167 No.19
Vol.2016-MBL-79 No.19
Vol.2016-ITS-65 No.19
2016/5/27

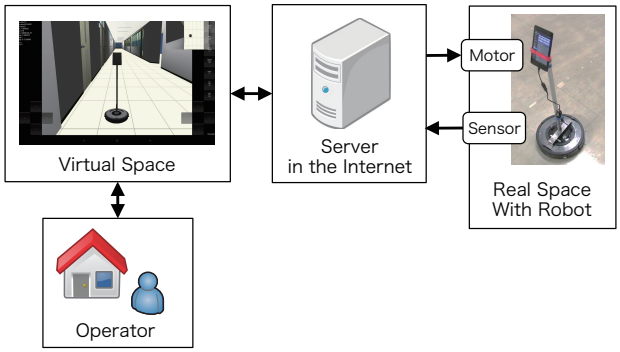

図 1 対象とする遠隔ナビゲーションシステムの構成.

上に構築される．プラットフォームの設計指針は以下のとおりである．

- プラットフォームの利用者は，ICT 分野のソフトウェアプログラマとし，実空間での操作はプラットフォーム内に隠蔽する
- 遠隔ナビゲーションに適したユーザインタフェースを提供する．地理的な前提知識を持たないユーザでも，ロボットを目的地までナビゲートできるものとする
- 大学のキャンパスやショッピングモール等，一般的な環境でのシステム構築を支援する

1 番めについては，実空間のロボット操作に対する API を規定することにより，ICT 分野のソフトウェアプログラマにとって一般的な Web アプリケーション開発と同様の手法で，案内サービスの開発を実現している．2 番めについては，仮想空間上での移動，探索等により直感的な操作を実現している．3 番めについては，可能な限りロボット単体のセンサで環境を認識した上で，この部分をプラットフォームの機能として実装し，隠蔽している．

プラットフォームの構成を **図 2** に示す．インターネット経由でロボットとの通信を行うための通信ライブラリ層，案内サービスを実現するためのプラットフォーム機能群，各々のサービスに相当するユーザアプリケーション，操作者とのフロントエンドである Web サーバから構成され，ソフトウェアプログラマは，プラットフォームが API として提供する機能群を利用し，ユーザアプリケーションを開発することになる．

2.2 プラットフォームの機能

プラットフォーム機能群は，案内サービスに要求される機能を API としてユーザアプリケーションに提供する「案内サービスモジュール」，プラットフォームに接続されるロボットとの間でデータをやりとりする「ロボット API モジュール」，プラットフォーム内で両者の処理の調整を行う「調整モジュール」の 3 つのモジュールから構成される．案内サービスモジュールには，サービスのためのイベント取得機能，カメラ画像取得機能等も含まれるが，本稿ではこの中のナビゲーション機能を考察対象とする．

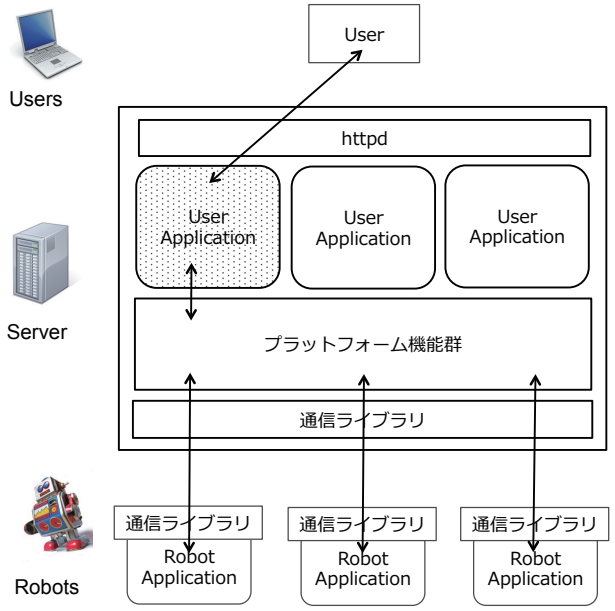

図 2 プラットフォームの構成.

3. 遠隔ナビゲーション機能

3.1 機能の概要

対象プラットフォームでは，ユーザの操作は，仮想空間として構築された GUI を用いて行われる．そのためナビゲーション機能は，仮想空間，実ロボット，サーバの 3 つの部分から構成されると考えられ，事前に構築しておいた仮想空間を利用し，遠隔地に存在するロボットを操作することになる．対象とするロボットは低コストの簡易型ロボットを想定し，実環境におけるロボット制御で必要な位置推定，経路計画などの処理は全てクラウド上（サーバ上）に実装する．

サーバ上に実装される機能は，i) 仮想空間での移動命令を実ロボットの移動命令に変換する機能，ii) 実ロボットから観測データを収集する機能，iii) 収集した観測データから実ロボットの状態を推定する機能，iv) 仮想ロボットと実ロボットの状態のズレを補正する機能の 4 つである．遠隔ナビゲーションは，仮想空間と実空間でのナビゲーションを，サーバ上でマッピングすることにより実現する [5].

3.2 仮想空間でのナビゲーション

まず，仮想空間でのナビゲーションを考える．仮想ロボットの状態を，時刻 t におけるロボットの二次元座標 (x_t^*, y_t^*) と姿勢 θ_t^* を用いて $r_t^* = (x_t^*, y_t^*, \theta_t^*)^T$ と表わし，仮想ロボットの移動を，移動速度 v_t^* と角速度 ω_t^* を用いて $a_t^* = (v_t^*, \omega_t^*)^T$ と表わすことにする．すると，仮想空間では a_t^* が入力として与えられることになるが，仮想ロボットの状態に関する情報は完全に観測可能であるため，これが r_t^* に反映され，仮想ロボットの移動が実現する．

Vol.2016-DPS-167 No.19
Vol.2016-MBL-79 No.19
Vol.2016-ITS-65 No.19
2016/5/27

3.3 実空間でのナビゲーション

次に，実空間でのナビゲーションを考える．実ロボットの状態を，時刻 t におけるロボットの二次元座標 (x_t, y_t) と姿勢 θ_t を用いて $r_t = (x_t, y_t, \theta_t)^T$ と表わし，実ロボットの移動を，移動速度 v_t と角速度 ω_t を用いて $a_t = (v_t, \omega_t)^T$ と表わすことにする．また，実環境における制御の集合を $a_{1:t} = \{a_1, ..., a_t\}$，観測の集合を $z_{1:t} = \{z_1, ..., z_t\}$ とする．

実ロボットは，不確実性の伴う実環境内を，サーバから送信される移動コマンド（a_t^* をサーバ上で実ロボット用コマンドに変換したもの）に従って移動するが，実ロボットの真の状態を直接観測することはできない．そのため，移動誤差を考慮して，移動後の状態を推定することになる．今，サーバからの命令としての移動を $\hat{a}_t = (\hat{v}_t, \hat{\omega}_t)^T$，$\epsilon_t$ を移動の誤差（正規分布等で与える）とすると，$a_t = \hat{a}_t + \epsilon_t = (v_t, \omega_t)^T$ と書け，この a_t を使って r_t は以下の式で推定される．

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} v_t cos\theta_{t-1} \\ v_t sin\theta_{t-1} \\ \omega_t \end{pmatrix} \triangle t \qquad (1)$$

これが，確率分布として定義される遷移モデル，$p(r_t \mid r_{t-1}, a_t)$ になる．ここで，$\triangle t$ は離散化した時刻の 1 ステップに相当する微小時間である．

3.4 マッピング機能

仮想ロボットは理想的な環境内を誤差なしに移動するが，実ロボットは不確実な環境内を移動するため，適切な補正を行わない限り，r_t と r_t^* のずれは拡大する．この補正を行うのがマッピング機能である．本システムでは，実空間のモデルとして仮想空間を事前に構築することから，これを位置推定のための地図として利用することを考える．ここでは，ランドマーク m_j の集合として地図 m を定義し，m_j の位置を $(m_{j,x}, m_{j,y})^T$ と表わす．そして，状態が r_t のとき z_t が観測される確率を，確率分布 $p(z_t \mid r_t, m)$ として与えておく（センシングの精度を正規分布等で与える）．これが，ここでの観測モデルになる．

この条件の下で，$z_{1:t}$，$a_{1:t}$，m から r_t を推定する．本稿では，実ロボットの初期状態は既知であり，状態の更新は再帰的に行われると仮定する．このとき，状態 r_t の確率分布は，

$$p(r_t|z_{1:t}, a_{1:t}, m) = \alpha\ p(z_t|r_t, m) \qquad (2)$$
$$\times \int p(r_t|r_{t-1}, a_t)\ p(r_{t-1}|z_{1:t-1}, a_{1:t-1}, m)\ dr_{t-1}$$

となる．ここでベイズ理論とマルコフ性の仮定を用いた．その結果，r_t が逐次的に推定されることになる．

次に，これを r_t^* と比較し，差分を解消するための移動コマンドを生成し，実ロボットを移動させる．制御の振動を防ぐために，この補正は 1 回のみ行う．r_t と r_t^* の差が解消しない場合には，仮想ロボットの位置を更新する．

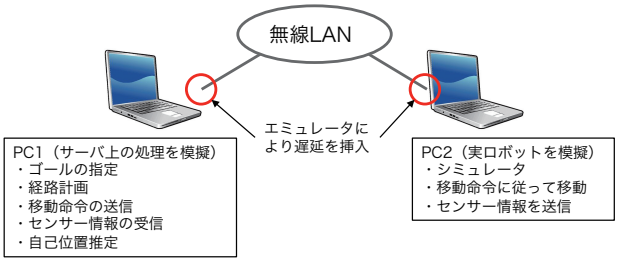

図 3 実験の環境．

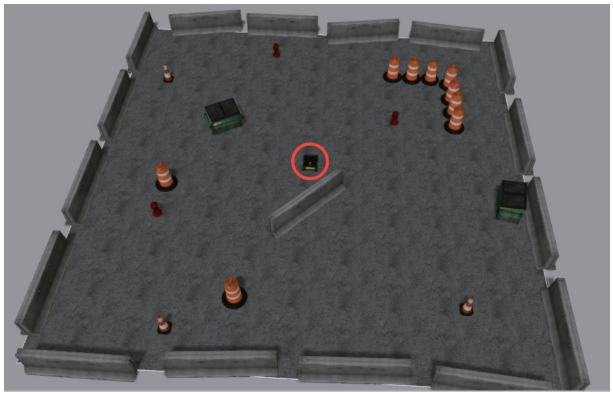

図 4 シミュレータの画面．

4. 実験

クラウド環境を用いた遠隔ナビゲーションにおけるネットワーク遅延の影響を調査するために，ロボットシミュレータを用いた遠隔ナビゲーション実験を行う．ここでは，前章で述べた実ロボットと仮想ロボット間の状態のずれを調べる．

4.1 実験の環境

実験は，無線 LAN（同一セグメント）に接続された 2 台の PC（PC1，PC2）を用いて行う．実験の環境を **図 3** に示す．PC2 では実ロボットを模擬したロボットシミュレータを実行する．シミュレータ内のロボットは，外部からの移動命令に従いシミュレータ環境内を移動し，環境内での観測結果をセンサー情報として出力する．PC1 ではサーバ上の機能を模擬する．ここでは，地図上で指定したゴールに従って経路計画を行い，その結果を移動命令としてシミュレータ内のロボットに送信する．また，ロボットからセンサー情報を受信し，自己位置推定を行う．クラウド環境は，ネットワークエミュレータを用いて出力パケットに遅延を挿入することにより模擬する．

表 1 利用する ROS パッケージ

パッケージ名	用途
move_base	経路計画とナビゲーション
amcl	自己位置推定
map_server	地図の管理
gmapping	SLAM による地図の生成

ロボットシミュレータには Gazebo[*1]を利用し，四輪自律走行ロボット Husky[*2]のモデルを利用する．シミュレータの画面を **図 4** に示す．画面中央の赤丸で囲った物体がロボットである．このロボットが環境内を移動する．センサー情報としては，Laser Lange Finder（LRF）およびオドメトリ情報が取得できる．サーバ側でのナビゲーション機能の実装には ROS[*3]パッケージを利用し，可視化ツール rviz[*4]を用いてロボットの持つ地図情報と受信したセンサー情報を表示する．利用する ROS パッケージの一覧を **表 1** に示す．この中で，amcl パッケージは，与えられた地図，遷移モデル，センサー情報を使って自己位置推定を行うパッケージであり，パーティクルフィルタにより実装されている．オドメトリから移動距離と方向を推定し，LRF による距離観測の結果により推定した位置を補正する．ネットワーク遅延の挿入には Linux の tc コマンドを利用する．これにより，PC1 と PC2 の無線 LAN インタフェースからの出力パケットに遅延を挿入する．遅延は，平均値と分散を指定した正規分布として与える．

4.2 実験の内容

実験の開始前に，環境地図を作成する．これには gmapping パッケージを利用し，外部コマンドによりシミュレータ内のロボットを環境内で移動させることにより地図を取得する．実験では，PC1 上の可視化ツールによりナビゲーションのゴールを指定することによりロボットを移動させる．ゴールの指定は 4 段階で行い，1 つのゴールに到着した時点で次のゴールを指定するという作業を繰り返し，ロボットを最終的なゴールまでナビゲートする．具体的には，図 4 に示されるロボットの位置を初期位置とし，右上の障害物の裏，左下の角，初期位置の壁に対して裏側，初期位置の順にゴールを指定する．そして，初期位置に戻った時点での，サーバ上での推定位置とシミュレータから受信した LRF の情報を比較する．

以上の実験を，平均遅延時間 μ とその分散 σ，制御および観測の間隔 Δt を以下のように変化させて実施する．なお，ここに示す遅延時間は片道の遅延時間であり，PC1, PC2 それぞれの無線 LAN インタフェースに同様に設定を行うものである．また，$\mu = \sigma = 0$ の場合でも，LAN 間での転送時間はかかることに注意が必要である．

- $\mu = 0$ ms, $\sigma = 0$ ms, $\Delta\, t = 100$ ms（デフォルト値）
- $\mu = 100$ ms, $\sigma = 10$ ms, $\Delta\, t = 100$ ms
- $\mu = 500$ ms, $\sigma = 50$ ms, $\Delta\, t = 100$ ms
- $\mu = 500$ ms, $\sigma = 50$ ms, $\Delta\, t = 33$ ms
- $\mu = 500$ ms, $\sigma = 50$ ms, $\Delta\, t = 1000$ ms

[*1] http://gazebosim.org/
[*2] http://www.clearpathrobotics.com/
[*3] http://www.ros.org/
[*4] http://wiki.ros.org/rviz/

4.3 実験の結果

実験の結果を **図 5**〜 **図 9** に示す．可視化ツールでは，事前に作成した自己位置推定のための地図（PC1 上のロボットモデルが認識している環境）が表示される．ここでは，障害物はコストマップとして表示され，障害物が存在するエリア（深緑），車輪型ロボットの移動（回転等）により衝突の可能性のあるエリア（薄緑），障害物に近づきすぎることを避けるためのバッファとなるエリア（薄紫）の 3 段階が色分けされて表示される．コストマップには，ロボットが持っている地図をベースに定義される大域的なコストマップのほか，受信したセンサー情報に従い定義される局所的なコストマップが存在し，地図上の水色と濃い青のエリアはこれに相当する．センシングの結果観測される障害物の位置（今回は LRF の観測結果）は黄色の線で表示されている．ロボットの経路計画は，これらのコストマップに従って行われる．ここでは，中央付近の黒い物体がロボットであり，細い水色の矢印の集合がゴールに到達するまでのロボットのオドメトリである（ロボットの軌跡を示す）．いずれの結果も，4 段階で指定したゴールへの移動が終了した時点での状態が表示されており，自己位置が正しく推定されている場合は大域的コストマップと局所的コストマップ，センサー情報が一致する．

$\mu = \sigma = 0$ の場合（図 5），環境地図と LRF の観測結果は良く合っており，ロボットの移動に伴う誤差の蓄積が，観測結果により適切に補正されていることがわかる．遅延の大きくない環境であれば，ネットワークを介したナビゲーションを適切に行うことができると考えられる．$\mu = 100$, $\sigma = 10$ の場合（**図 6**）も，適切なナビゲーションが行われている．これは，制御周期が 100 ms であるため，1 周期程度の遅れは解消されるためと考えられる．ただし，観測遅れの影響から，ロボットの移動速度は 図 5 に比べて遅くなっており，移動速度が早い場合には影響が大きくなる可能性がある．$\mu = 500$, $\sigma = 50$ の場合（**図 7**）は，中央の壁の観測結果（LRF のデータ）が 2 重になっており，観測結果にずれが発生している．周囲の壁の観測結果にも遅れが生じており，サーバ上のロボットモデルが認識している自己位置と，観測から推測される位置にずれが生じていることがわかる．

次に，この影響度合いを，遅延時間を 500 ms に固定し，制御，観測の周期を変えることで検証する．まず，$\Delta\, t = 33$ の場合（**図 8**），環境地図と観測結果は大きくずれており，適切な自己位置推定が行えない，つまり適切なナビゲーションが行えていないことがわかる．センセー情報が大域的なコストマップと大きくずれてしまったことから，全体的に局所的なコストが定義されてしまっている．$\Delta\, t$ は小さい方がリアルタイム性が向上すると考えられるが，ずれに対する感度も上がる．さらに，今回の実験では，ロボットの移動速度が極端に遅くなる現象が観測され，ロボット

Vol.2016-DPS-167 No.19
Vol.2016-MBL-79 No.19
Vol.2016-ITS-65 No.19
2016/5/27

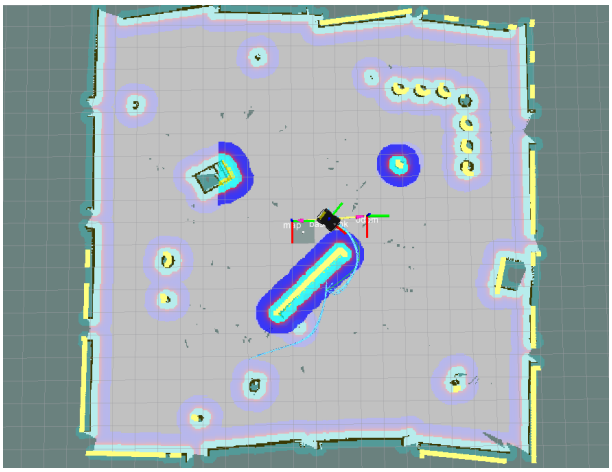

図 5 実験の結果（$\mu = 0,\ \sigma = 0,\ \Delta\, t = 100$）.

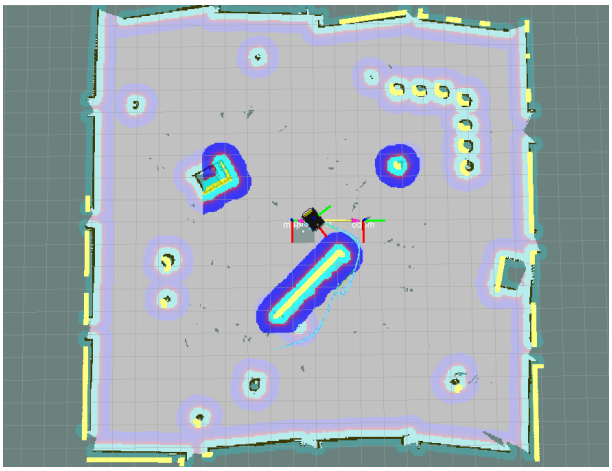

図 6 実験の結果（$\mu = 100,\ \sigma = 10,\ \Delta\, t = 100$）.

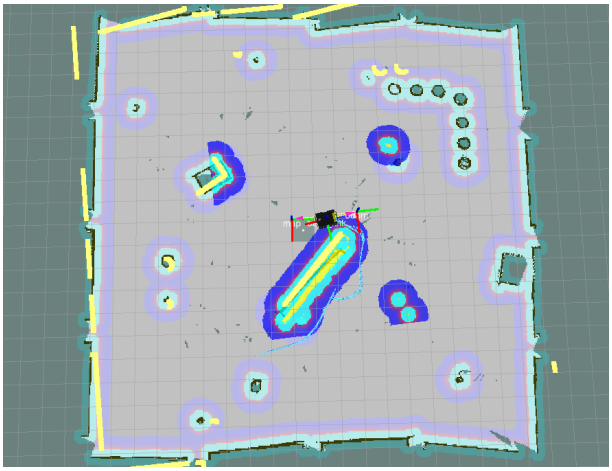

図 7 実験の結果（$\mu = 500,\ \sigma = 50,\ \Delta\, t = 100$）.

の軌跡も迂回しがちになっている．より現実的な環境では大きな誤差になると考えられる．一方，$\Delta\, t = 1000$ の場合（図 9）は，観測の遅れは存在するが，$\Delta\, t$ を大きくすることでずれの影響を削減することができている．ただし，この場合もロボットの移動速度は非常に遅く，観測結果を待ちながら制御命令を出していることがわかる．

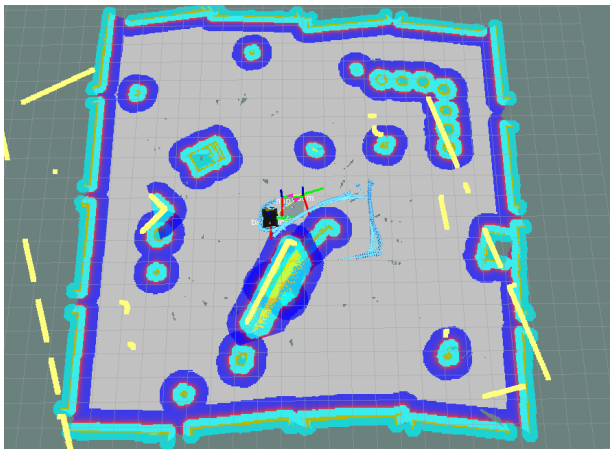

図 8 実験の結果（$\mu = 500,\ \sigma = 50,\ \Delta\, t = 33$）.

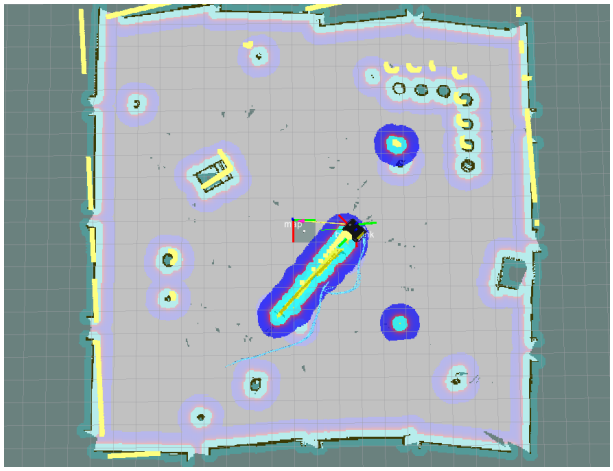

図 9 実験の結果（$\mu = 500,\ \sigma = 50,\ \Delta\, t = 1000$）.

以上より，今回の実験条件では，1 秒程度のネットワーク遅延が発生する場合，適切なナビゲーションが行えない可能性があることがわかった．遅延時間と制御の精度との関係については今後検証を進めていく必要があるが，ロボットの移動速度，制御周期に依存することから，クラウド環境で想定される数 10〜数 100 ms 程度の遅延時間であっても，大きな影響が出る可能性がある．

5. 制御モデルの拡張

以上の結果より，本章では，位置推定モデルにネットワーク遅延を組み込むことを考える．3 章で説明したモデルでは，正確なランドマークは存在するが，観測はサーバ上で行うため，実際のランドマークの位置と観測時点での推定位置にずれが生じる．また，移動命令もサーバから送出されるため，送出時点と移動時点で時間のずれが生じる．この対応方法を考える．遠隔操作を行うオペレータは，操作と観測の遅延に慣れると言われているが，これはこの補正を暗黙のうちに行っており，それまで経験した観測と制御のずれを考慮し，観測を遅らせ，制御を早めていると考えられる．本モデルではこれを利用する．

Vol.2016-DPS-167 No.19
Vol.2016-MBL-79 No.19
Vol.2016-ITS-65 No.19
2016/5/27

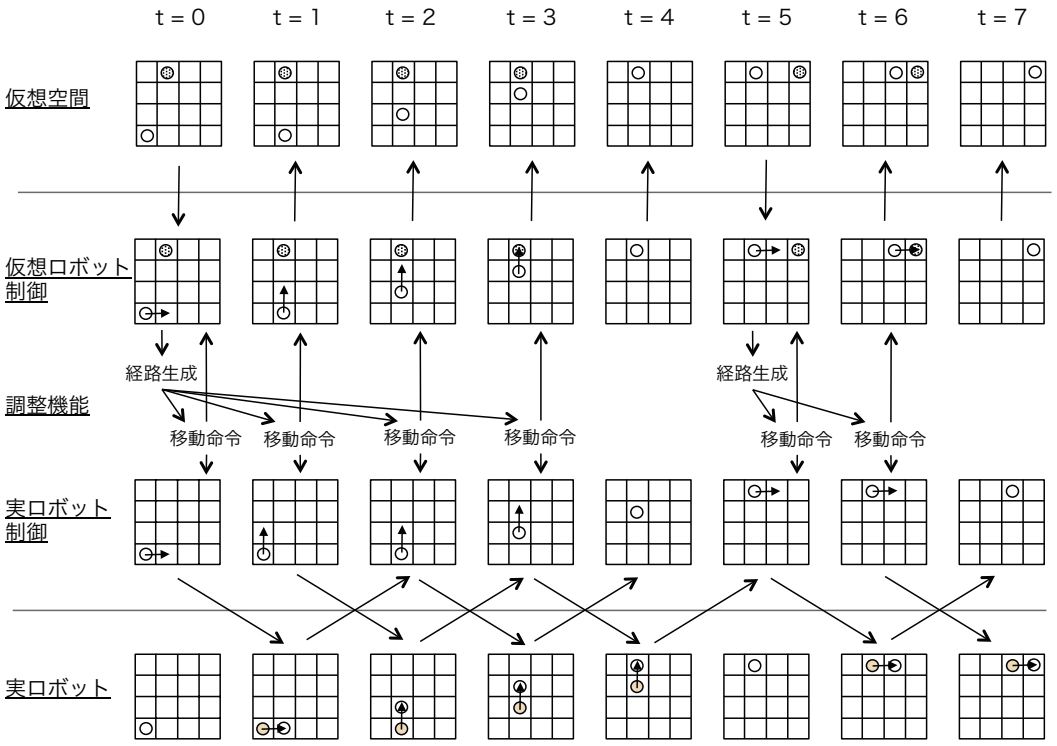

図 10 ネットワーク遅延の影響.

5.1 ナビゲーションにおける遅延の影響

まず，ネットワーク遅延の影響を **図 10** を用いて説明する．ここでは，ロボットが格子状に区切られた各セルを上下左右一マスづつ移動するモデルを例として取り上げる．図の上から，ユーザインタフェースに相当する仮想空間，それに対応したサーバ上の仮想ロボットモデル，仮想空間と実空間の調整機能，サーバ上の実ロボットモデル，実空間に存在する実ロボット，それぞれの位置（白丸）と移動命令（矢印）を表わす．網掛けの丸はユーザが指定するゴール地点を示す．最上部に記された $t = 0$ から $t = 7$ は時間ステップを表し，本モデルでは，時間は 1 ステップづつ離散的に推移すると仮定する．簡単のために，移動命令，観測ともに不確実性は存在せず，ロボットは移動命令に相当する移動を確実に行い，誤差のない正しい観測値が必ず得られると仮定する．また，ネットワーク遅延はサーバとロボット間にのみ存在し，片道 1 ステップ分の時間が遅延として必ず挿入されるとする．

今，仮想空間で $(0,0)$（左下のセルを $(0,0)$ とし，一つ右のセルを $(1,0)$，一つ上のセルを $(0,1)$ のように座標を割り振っておく）の位置に仮想ロボットが存在し，ユーザが仮想空間上で $(1,3)$ の位置にゴールを設定したとする．すると，現在位置とゴールを受け取ったサーバ上の仮想ロボットモデルは，調整機能に依頼しゴールまでの移動経路を生成する．この例では，右上上上という 4 ステップから成る経路が生成されている．この命令はサーバ上の実ロボットモデルと仮想ロボットモデル双方に送信される．仮想ロボットモデルは移動命令に従ってそのまま移動するので，$t = 1$ の時点で $(1,0)$ に移動する．一方，サーバ上の実空間ロボットモデルは，移動命令を実空間のロボットに送信し，センサー情報をロボットから受信することで移動を完了する．移動命令は $t = 1$ で実ロボットに送られ，その移動結果の観測データは $t = 2$ で実ロボットモデルが受信する．そのため，実ロボットモデルは $t = 2$ で移動することになる．このように，移動命令の遅れと観測の遅れが発生し，仮想ロボットモデルと実ロボットモデルの位置にはずれが生じる．現実世界では，移動命令，観測ともに不確実性が存在し，さらにネットワーク遅延（観測，制御のずれ）にも不確実性が加わることから，より大きな差異が発生すると考えられる．

本モデルではこの状態を模擬するために，ping コマンド等を利用し，対象となるネットワークの遅延時間の平均値と分散をあらかじめ計測しておき，これを 3 章で定義した遷移モデルと観測モデルに組み込む．なお，ここでは平常時のネットワーク環境を前提とし，バースト的な性能劣化は考慮しない．

5.2 遷移モデル

平均遅延時間に相当する，数ステップ前の移動命令から r_t を推定する．このとき，遅延時間は正規分布に従うと仮定し，各ステップごとの生起確率に応じた重み付き a_t の総和を，新たな a_t として式 (1) に代入する．t を連続時間と仮定すると，平均遅延時間を μ，分散を σ^2 としたとき，

$$a_t = \frac{1}{\sqrt{2\pi}\sigma} \int \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot a_x \, dx \qquad (3)$$

となる．ここで a_x は，時刻 x における移動命令である．実際には時刻は離散化して実装する．

5.3 観測モデル

平均遅延時間に相当する，数ステップ後の観測結果から r_t の事後確率を推定する．重み付けについては遷移モデルと同様に考え，重み付けされた新たな z_t により，$p(z_t \mid r_t, m)$ を決定する．式 (3) と同様に，以下で求める．

$$z_t = \frac{1}{\sqrt{2\pi}\sigma} \int \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \cdot z_x dx \qquad (4)$$

5.4 離散化

実装時には時間を離散化する必要がある．ここでは，平均時間に相当するステップ数を k，分散に相当するステップ数を l とし，r_t を，$\{a_{t-k-l}, a_{t-k-l+1}, ..., a_{t-k}, ..., a_{t-k+l}\}$ と，$\{z_{t+k-l}, z_{t+k-l+1}, ..., z_{t+k+l}\}$ を用いて推定する．ここで，$k = \lfloor \mu/\triangle t \rfloor$，$l = \lfloor 2\sigma/\triangle t \rfloor$ である．この結果，式 (2) の右辺は，$p(r_t \mid z_{1:t+k+l}, a_{1:t-k+l}, m)$ に置き換えられ，これを逐次推定していくことになる．

6. 関連研究

ネットワークロボット，クラウドロボティクスに関する研究は盛んに行われており，例えば，ロボット用クラウド環境である RoboEarth [7] や，リソース利用のスケジューリング問題に取り組んだ研究 [8] などがある．多くの研究は，クラウド環境への機能のオフローディングを考察したものであり，ネットワークを介した制御についてはほとんど考えられていない．ネットワーク品質を考慮した研究としては，ネットワーク転送のモデル化を行っている研究 [9]，遅延を考慮したフィードバック制御 [10]，QoS（Quality of Service）モニタリングを利用した制御 [11] などが提案されている．しかし，観測を基にネットワークリソースのスケジューリングを行う手法がほとんどで，制御モデルへの遅延の組み込み，学習機構等は考慮されていない．

7. まとめ

本稿では，クラウド環境を利用した簡易ロボット遠隔ナビゲーションを対象に，ネットワーク遅延がナビゲーション結果に与える影響について考察した．ここでは，遠隔操作時の遅延に慣れるオペレータの行動を模擬し，事前に予測しておいた遅延モデルに従い，観測を遅らせ，制御を早める仕組みを組み込んだ制御モデルを検討した．また，ロボットシミュレータを利用した遠隔ナビゲーション実験を行い，擬似的に遅延を挿入したネットワーク環境では，ナビゲーションが適切に行われない場合があることを明らかにし，遅延を考慮した制御モデルの必要性を述べた．

今回，単純な遅延モデルを用いたが，各種パラメータの与え方，ステップ幅の決め方等，環境に適応したモデルの学習が必要であると考えられる．今後，学習機構を組み込んだモデルの構築，およびシミュレーション実験による手法の有効性検証を行っていく予定である．

謝辞 本研究の一部は，JSPS 科研費 15K00137, 26330299 の助成を受けたものである．

参考文献

[1] Quigley, M., Conley, K. and Gerkey, B.: ROS: an open-source Robot Operating System, *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics 2009* (2009).

[2] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T. and Yoon, W.: RT-Middleware: Distributed Component Middleware for RT (Robot Technology), *IEEE/RSJ International Conference on Intelligent Robots and Systems 2005 (IROS 2005)*, pp. 3933–3938 (2005).

[3] Arumugam, R., Enti, R., Bingbing, L., Xiaojun, W., Baskaran, K., king, F., Kumar, A., Meng, K. and KitVikas, G.: DAvinCi: A Cloud Computing Framework for Service Robots, *IEEE International Conference on Robotics 2010* (2010).

[4] 高橋雅彦，土屋陽介，成田雅彦，加藤由花：移動ロボットを利用した案内サービスの構築を支援するプラットフォーム環境，情報処理学会論文誌， Vol. 55, No. 2, pp. 1234–1245 (2014).

[5] Kato, Y.: A Remote Navigation System for a Simple Tele-presence Robot with Virtual Reality, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS2015)*, pp. 4524–4529 (2015).

[6] 加藤由花：テレプレゼンスロボットのための遠隔ナビゲーション手法，人工知能学会全国大会，pp. 1I4–NFC–02b–3 (2016).

[7] Riazuelo, L. et al.: RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach, *IEEE Trans. on Automation Science and Engineering*, Vol. 12, No. 2, pp. 432–443 (2015).

[8] Wang, L., Liu, M. and Meng, M. Q.-H.: Real-Time Multisensor Data Retrieval for Cloud Robotics Systems, *IEEE Trans. on Automation Science and Engineering*, Vol. 12, No. 2, pp. 507–518 (2015).

[9] Salmeron-Garcia, J., Inigo-Blasco, P., del Rio, F. D. and Cagigas-Muniz, D.: A Tradeoff Analysis of a Cloud-Based Robot Navigation Assistant Using Stereo Image Processing, *IEEE Trans. on Automation Science and Engineering*, Vol. 12, No. 2, pp. 444–454 (2015).

[10] Penizzotto, F., Slawinski, E., Salinax, L. R. and Mut, V. A.: Human-centered control scheme for delayed bilateral teleoperation of mobile robots, *Advanced Robotics*, Vol. 29, No. 19, pp. 1253–1268 (2015).

[11] Blumenthal, S., Hochgeschwender, N., Prassler, E., Voos, H. and Bruyninckx, H.: An Approach for a Distributed World Model with QoS-based Perception Algorithm Adaptation, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS2015)*, pp. 1806–1811 (2015).