インターネットを介した車両の経路追従制御におけるディジタルツインコンピューティング実用化に向けた伝送遅延による遠隔走行制御特性劣化の改善に関する検討

防衛大学校理工学研究科後期課程

電子情報工学系専攻　情報通信工学教育研究分野

吉本　雄大

令和 5 年 3 月

目次

第 1 章　序論

1.1　研究背景

1.1.1　ディジタルツインコンピューティングへの注目

社会の IoT（Internet of Things）化が進む中で, ICT（Information and Communication Technology）を活用した新しいサービスの実現が期待されている[1, 2]．ICT の中でも，近年ディジタルツインという概念が注目を集めている．ディジタルツインとは，高度なセンシングにより現実空間の情報を収集してコンピュータ内で現実空間の状態を再現する技術体系，またはコンピュータ内の再現空間そのものの呼称である．ディジタルツインが適用されたシステムでは，人・物・環境についてのセンシング情報がネットワークを介して遠隔地のサーバに送信され，その中で現実の鏡合わせのようにディジタルツイン空間が再現される．集約されたセンシング情報と情報処理技術によって現実空間を高精度に模擬することで，現実空間の状態を遠隔地からリアルタイムに把握することや未来の状態を高精度に予測することが可能となる．この利点が既存のサービスの効率化および新しいサービスの実現に役立つと期待されている[3].

ディジタルツインは，2002 年にミシガン大学のマイケル・グリーブスによって初めて提唱された[4]．提唱時は工業製品のライフサイクルマネジメントの効率化を意図した概念であったが，現在は様々な産業分野への応用が期待されている．主な分野としては，製造業，ヘルスケア，スマートシティが挙げられる．製造業分野では，設計段階における製品機能の確認，生産ラインの最適化および監視，製品製造後の不具合発生の予測といった製品ライフサイクル全般へのディジタルツインの適用が期待されている[5, 6, 7]．実例として，いくつかの自動車会社が自社の製造設備・環境のディジタルツインを構成・運用することを検討している[8, 9]. ヘルスケア分野では，患者の身体をモデリングしたバイオディジタルツインと生体センシング情報を用いることで，遠隔医療における患者の現症の詳細な把握および将来的な病症発生の予測に基づいた治療計画の最適化を実現することが期待されている[10, 11]．スマートシティ分野では，多領域にわたってディジタルツインの応用が検討されている．具体例として，

街区内におけるモビリティ支援・管理，街区開発事業の効率化，食料・エネルギ消費のむだ削減，経済活動の統合的分析・予測等のアプリケーションを実現するためのプラットフォーム開発が期待されている[12, 13, 14].

ディジタルツインの応用に関する検討は，遠隔地からのシステムの監視や高度な予測に基づいた設計・計画の最適化を目的としたものが多い．一方で，車両・ロボットのリアルタイムな遠隔制御においてディジタルツインを活用する方法も研究されている．基礎的研究として，ロボットアームの遠隔制御においてディジタルツインを活用する方法が検討されている[15, 16]．車両については，IoT 社会における安全・円滑な自動運転を実現する方法として，車両の走行環境をディジタルツイン上で予測するシステムが有効であると提唱されている[17]．走行車のリアルタイム遠隔自動制御にディジタルツインを適用することで，多数の車両とそれらの周辺環境の状況を高度に再現する，または未来の走行状態を予測することが可能となる．これによって，ディジタルツインが適用された制御用サーバで一元的な走行車遠隔制御を実施することが可能になる．特に，ディジタルツイン上で制御対象の車両の状態を認識できるため，協調制御を効率的に実施できる利点が考えられる．

ディジタルツインを活用して安全な走行車遠隔制御を実施するためには，アプリケーションに要求される制御精度に応じて，ディジタルツインのモデリング精度を高める必要がある．走行環境を高度に再現するためには，各車両の走行状態，通路上の人・障害物，および路面や風雨等の環境についての情報が必要である．これらの情報は，各車両と通路付近に配備されたカメラ・センサおよび気象情報等を公開するサーバからインターネットを介して入手することができる．また，安定した遠隔自動制御のためには，ディジタルツインを備える制御サーバと走行車との間の通信品質を安定化させる仕組みが必要である．上記を踏まえて走行車遠隔制御システムを整えることで，以下を実現するアプリケーションを提供することが可能となると考えられる．

(1) サーバによる走行計画の作成・更新[18]および走行車の遠隔予測制御の一元的な実行

(2) 車両側からは見通し外となる空間の状態をディジタルツイン上で再現・予測することによる衝突事故等の回避[19]

(3) ディジタルツインをモニタリングすることによる走行状況の立体的把握および人間の判断による適切な緊急停止処置の実施

(4) 事故が実際に発生した際の過去のセンシング情報を用いた現場状況の再現および検証

以上から，より利便性の高い走行車遠隔自動制御を実現する手段として，ディジタルツインの活用方法を検討する価値があると考えられる．

1.1.2 走行車の遠隔自動制御

走行車の自動制御は多くの分野で実用化を期待されている技術である[20]. 比較的大型の車両に関しては，公道における普通自動車の安全な自律運転[21]，大陸横断配送におけるトラックの自律隊列走行[22]が検討例として挙げられる．小型搬送車に関する例としては，軍事用のワイヤレス多機能ロボット[23]，公共空間における自動車椅子サービス[24]が挙げられる．また，本格的な実用化に向けて自動運転車両やオフィス用品配送車両の実証実験が公道，企業のオフィス内等の環境下で行われている[25, 26].

自動運転車両の制御方法は大きく分けて 2 種類存在する． 1 つは各車両の自律分散制御方式である．この制御方法では，自己位置と周辺環境のセンシング，制御値の計算および駆動部の制御といった走行に必要な機能を全て各車両に実装する必要がある．そのため，車両 1 台あたりに必要な処理能力および製造・維持コストが高い．自動運転サービスで提供する車両の台数が多い程システム全体のコストは大きくなる．もう 1 つの制御方法は制御サーバと無線通信による遠隔制御方式である．制御サーバとしては，車両に搭載するコンピュータよりも遥かに高い演算能力と信頼性の高いブロードバンドネットワークを備えたサーバを用いることが期待できる．車体の制御機能は緊急停止等の安全上遠隔化できない機能を除いて制御サーバに集約され，各車両は制御サーバから送信される制御信号に従って走行する．現場の走行車の状態は車両自身または周辺の IoT 機器からサーバへ伝達する．これによって，各

車両の機能を簡素化し，1 台当たりのコストを低減することが可能である．多数の車両を活用する自動運転サービスの場合，走行車の台数が多い程システム全体のコストが自律分散制御方式よりも小さくなる．制御機能が制御サーバに集約されているため，システムの更新と保守管理を一元的に実施することが可能である．また，各車両の状態もサーバが一元的に認識できるため，サーバによって複数の走行車を遠隔協調制御することが可能である．以上の利点から，遠隔制御方式の場合，費用対効果と利便性が高い走行車遠隔自動制御システムを実現することが可能であると考えられる．より経済的な自動運転アプリケーションが様々な分野で実用化されると，社会全体で移動・配送に関わるコストや労働力をより効率化できる可能性がある．

遠隔制御においては通信ネットワークを介した制御の安定性を保証することが必要である．通信を介したフィードバック制御の制御安定性は伝送品質に大きく影響される[27]. インターネットや無線アクセスを介したパケット伝送品質が低い場合，つまり伝送遅延，遅延ジッタおよびパケットロス率が大きい場合，遠隔地の制御サーバでは端末を安全に制御できなくなる危険性が高い．通信ネットワークに要求される伝送品質は運用するサービスによって異なる[28]. 非常に精密な動作が求められる遠隔外科手術用ロボットシステムにおいては，外科手術におけるリスク回避の観点から通信伝送品質に対する要求条件は非常に厳しい．この場合，High Definition（HD）画質相当のデータを 10 ms 以下の遅延で安定して伝送できる通信回線を確保する必要があると示されている[29]. 無人走行車の遠隔制御に関しては，シミュレーションにより 100 ms を超える片道伝送遅延が生じる場合は車両に最短経路上を走行させることが困難になることが示されている[30]. 実際の走行車を用いた実験では，片道伝送遅延が 250 ms 以上である場合には目標走行経路を許容範囲内で走行することが不可能になることが確認された[30]. このように, 伝送遅延の長さは遠隔制御システムを実現する上で重要な課題である．

遠隔制御による走行車自動運転の実用化を達成するために様々な研究が行われている．例として，クラウドサーバと伝送遅延が小さいモバイルエッジサーバを協調利用する手法が検討されている[31]．また，第 5 世代モバイル通信ネットワーク（The fifth Generation Mobile

network : 5G）を用いた走行車遠隔制御システムの実現性が検討されている[32]．5G は超高信頼かつ低遅延の無線通信（Ultra-Reliable and Low Latency Communications : URLLC）を実現できる技術であり，無線通信において 1 ms 以下の伝送遅延を達成可能であることが示された[33]．URLLC とモバイルエッジサーバを活用することで，フィードバックループの遅延を 10 ms 程度で安定させた遠隔制御が実現できる可能性がある[34]．しかし，モバイルエッジサーバで遠隔制御を実施する場合，サーバ 1 台あたりのカバーエリアが狭いため，遠隔制御サービスの提供範囲の拡大に応じて多数のサーバを配置する必要がある．また，制御端末がカバーエリアを行き来する場合はサーバ間で端末情報を正確に取り扱う必要がある．そのため，サービス提供範囲の広さに応じたサーバの維持・運用コストの増加や制御システムの過度な複雑化といった問題点が考えられる．これらの問題を避けるためには，1 つのクラウドサーバで広範囲の端末制御を可能とする仕組みが必要である．クラウドサーバを用いる場合，モバイルエッジサーバと比べてより高度な演算リソースを活用できる，サーバの位置と自動制御サービスの利用場所に関する制約が大きく緩和される，遠隔制御に関する情報収集・処理を一元的に実行できるといった利点がある[35]．しかし，モバイルエッジサーバの場合よりも大きな伝送遅延が発生するため，遠隔制御の安定性を確保することが困難となる[36]．

1.2　研究目的

近年，様々な分野で小型無人車両の自動運転化が期待されている．具体例として，公共空間における自動車椅子[24]，街区や住宅地でのラストワンマイル配送[37]，オフィスビルや工場における物品配送・整理[26]が挙げられる．現在実用化されている自動制御車両は殆どが自律制御で走行するものである．一般的に，自律制御走行車は 1 台当たりの購入コストとシステム運用コストが高い．前節より，ディジタルツインを適用したクラウドサーバによる遠隔制御システムを実現化することができれば，走行車遠隔制御アプリケーションを広範囲かつ経済的に提供できるようになると考えられる．さらに，ディジタルツインを高度に活用することで，走行車のリアルタイム遠隔制御だけでなく，経路計画や維持管理等もクラウドサー

バでまとめて実施する統合的アプリケーションを実現することが可能になる．

小型走行車について現在主流である自律分散制御法方式からクラウドサーバによる遠隔集中制御化する利点について具体例を挙げる．自律制御走行の車両が高コストになる要因の 1 つとして，自己位置と周辺環境を認識するための SLAM (Simultaneous Localization and Mapping)[38]や Visual SLAM[39]の処理およびそれに基づく走行制御を各車両が個別に実行する必要があることが挙げられる．このような自律制御方式をディジタルツインを適用したクラウドサーバによる遠隔制御方式に置き換えることを考える．各車両は LiDAR 等センサに基づく数値またはカメラ画像をサーバにフィードバックとして送信し，サーバから送られる制御信号に従って走行する．フィードバックを受信するクラウドサーバは SLAM における地図情報を保有し，地図の構成・修正，各車両の位置情報の認識，および制御信号計算を実行する．このため，各車両が地図情報を保有し，端末間の直接的な通信または統合管理ソフトを介した情報の共有・更新をする必要がない．地図と各車両の位置に関する情報はディジタルツインを構成する一要素として活用可能である．車両からのフィードバック以外の情報も収集・分析することで走行空間をより高精度にディジタルツイン化することができる．現在または未来における歩行者の移動状態や通路の混雑等をディジタルツインで把握することで，衝突事故のリスクが低くかつ迅速に目標地点に到達する最適な経路をリアルタイムに探索することも可能である．このようなシステムの場合，各車両に要求される演算性能は自律制御方式の場合よりも大幅に低減できる．それに伴って，システム全体のコストも低減できる．なお，遠隔制御における安全上の観点から緊急停止機能は各車両に実装する必要がある．SLAM のためのセンサは変わらず車両に配備されるため，これは容易である．

ディジタルツインを適用したクラウドサーバによる小型車両遠隔制御システムを実現化するためには，いくつかの研究段階を経る必要があると考えられる．具体例を示すと，まず第 1 に，ディジタルツインを用いた基礎的な制御システムを提案するとともに走行車のリアルタイム遠隔制御におけるディジタルツインの有効性を示す．第 2 に，安全な遠隔走行制御を行うために必要なセンシング情報を分析する．第 3 に，走行車・通路等の周辺環境におけるカ

メラ・センサの最適な配備について検討するとともに，実用的な遠隔制御システムを設計・実証する．

本研究の目的は，図 1-1 に示すようなクラウドサーバ内のディジタルツインを用いた走行車遠隔制御システムを実現することである．特に，緊急停止機能を除き走行制御に関する機能の大部分をサーバに集約して，多数の小型走行車両を経済的に自動制御するシステムの実現を想定している．具体的なアプリケーションとして，屋内外における自律配送車両や監視用ロボットが挙げられる．小型走行車に着目した理由は，小型かつ比較的低速で走行するため，衝突等の安全上のリスクが比較的低く，クラウドサーバを用いた遠隔制御においても安全な遠隔制御を実現できる可能性が高いと考えたためである．本研究では，車両の経路追従走行にディジタルツインを適用することを前提とした遠隔走行制御システムを提案し，その有効性について検討した．この際，走行車として一般的である前輪によって走行方向を変化させる小型４輪走行車を制御対象として，シミュレーション及び実験を実施した．

指定された経路を追従させる制御機能は自動運転において基本的な制御方法である．通常のフィードバック制御で走行車の遠隔制御を実行する場合，指定経路上を安全に走らせるためには走行車と制御器との間の信号の伝送遅延を努めて小さくする必要がある．クラウドサーバを用いてフィードバック制御を実施する場合は，安全な経路追従走行が困難になる程の伝送遅延と遅延ジッタが発生する危険性が高い．目的とするシステムを実現化するためには，大きな伝送遅延とジッタが発生することを前提とした制御システムを考案する必要がある．本研究では，クラウドサーバにディジタルツインを適用した小型無人走行車の遠隔経路追従制御システムの実現化に向けて，ディジタルツインによる走行状態の予測結果を活用して遠隔走行制御における伝送遅延の影響を低減し，走行制御の精度及び安定性を向上させる方法を提案するとともに，その有効性を定量的に評価した．この際，エミュレータによって実際の通信における伝送遅延を模擬した．また，走行車を模擬する簡易的なディジタルツインを用いた．より高度かつ実用的なディジタルツインの適用に関しては，今後の課題である．

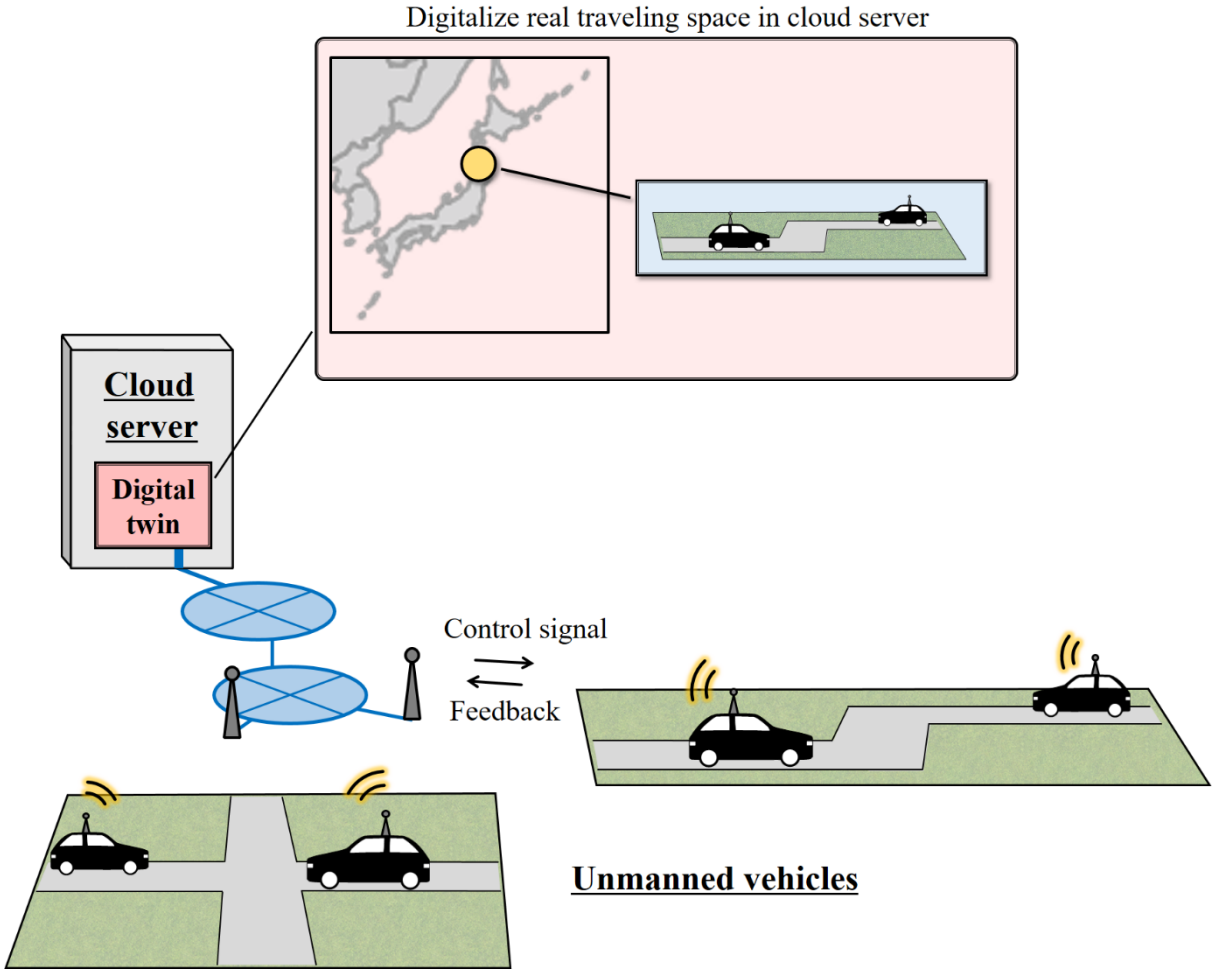


図 1-1　クラウドサーバを用いた無人走行車遠隔制御

1.3　研究概要

はじめに，クラウドサーバによる小型走行車遠隔制御において，ディジタルツインを活用して伝送遅延の影響を低減する制御方法を提案した．制御サーバには制御プログラムに加えて制御対象である走行車の状態を模擬するディジタルツインが適用される．走行車にはサーバから送信される信号の伝送遅延ジッタを吸収するジッタバッファが適用される．これらによって，制御サーバは一定時間後の走行車の状態をディジタルツインで予測して制御信号を作成・送信する．また，走行車は実際の位置と向きの情報を制御サーバに送信する．この情報は走行車に対するディジタルツインの予測誤差を修正し，予測結果を実際の状態に近づけるために使用される．

次に，上述の提案システムの有効性を評価するため，走行車の遠隔経路追従制御において簡易的なディジタルツインを適用した提案システムの場合とそうでない場合の制御特性をシミュレーションで比較検討した．クラウドサーバを制御サーバとすることを想定した伝送遅延を模擬するため，実際のインターネット上の遅延の測定結果に基づいて作成した遅延データセットをネットワークエミュレータで再生した．評価の結果，提案システムにより走行車の遠隔制御特性が大きく向上することが明らかになった．従って，ディジタルツインを適用したクラウドサーバによる走行車の遠隔経路追従制御が有効であることを確認できた．

次に，小型走行車の経路追従制御においてディジタルツインに求められる予測精度についてシミュレーションおよび小型ラジコン車を用いた実験によって検討した．ディジタルツインで走行空間を模擬する場合，車両自体の速度とステアリング角，および周辺環境の状態についてのセンシング情報が必要となる．ある時間における実際のセンシング情報とディジタルツイン上の予測結果に誤差がある場合，走行車遠隔制御の精度は低下する．ここでは，走行車の走行速度またはステアリング角の挙動についてディジタルツインと実際の車両との間で誤差が存在する場合に，遠隔経路追従制御の精度へ及ぼす影響を評価した．その結果，走行車の速度およびステアリング角の状態に関するディジタルツインの許容範囲を明らかにした．より安全な走行車遠隔制御のためには，走行速度よりもステアリング角の挙動をより正

確にディジタルツイン上で再現する必要があることも明らかにした．また，経路追従走行の目標経路の形状によっては，ディジタルツインに対して実際の車の速度が比較的遅い状態でも良好な遠隔走行制御を達成できることが示された．

最後に，提案システムにおける制御信号バッファリング時間の最適化についてシミュレーションにより検討した．インターネットと無線アクセスを介することでクラウドサーバから車両へ送られる制御信号のパケットには遅延ジッタが生じる．このジッタが大きい程走行車の遠隔経路追従制御の精度が低下する．そのため，ジッタバッファを車両側に実装することがより安定した走行を達成するために有効である．ジッタバッファは，パケットに適応的な遅延を付与する[40]ことで伝送遅延の変動を吸収するバッファである．インターネットを介した伝送遅延の変動特性は突発的かつ急激に変化する可能性がある．ジッタバッファを適用したシステムでは，システムの安定性を保障するために信号のバッファリング時間を最適化する仕組みが必要である．ここでは，提案システムにおいて制御信号の伝送遅延変動特性に対してジッタバッファの機能を動的に最適化するための仕組みを考案した．また，動的最適化を適用した場合とそうでない場合を比較し，走行車の遠隔経路追従走行の安定性がどの程度改善するか評価した．その結果，本検討で考案した動的最適化方法が有効であることを明らかにした．

上述の検討は，実際の伝送遅延状況を再現したケーススタディである．これによって，クラウドサーバによる小型走行車の遠隔経路追従制御において，走行車のディジタルツインを活用して伝送遅延の影響を低減する提案システムの有効性を確認した．また，伝送遅延が大きく不安定な状況を再現した中で，提案システムの制御特性を維持・向上させるために必要な要求条件について評価した．

1.4　論文構成

本論文は、本章序論から第 8 章までで構成されている．第 2 章以下の構成は次のとおりである．

・第 2 章　関連研究

車両・ロボット等の遠隔制御方式，フィードバックループ制御の安定化，ディジタルツインコンピューティング技術およびその応用についての先行研究等について述べるとともに，それらを踏まえた本研究の意義を説明する．

・第 3 章　クラウドサーバによる無人走行車の経路追従制御システム

クラウドサーバを用いた場合の小型車両走行のフィードバック制御の基本的な構成と伝送遅延による影響について述べる．

・第 4 章　無人走行車の経路追従走行システムへのディジタルツインの適用

第 3 章で述べたシステムを基として，クラウドサーバにディジタルツインコンピューティングを，走行車にジッタバッファを適用することで伝送遅延の影響を低減する遠隔制御システムを提案するとともに，その仕組みを説明する．

・第 5 章　無人走行車の遠隔経路追従制御におけるディジタルツインの有効性

1 つ目の検討として，簡易的なディジタルツインを用いてクラウドサーバによる走行車遠隔制御の精度を向上させる提案システムを構成し，その有効性をシミュレーションによって定量的に評価した結果を述べる．この章の内容は，後述の研究業績の一覧で示す論文[13]に相当する．

・第 6 章　クラウドサーバの遠隔走行制御におけるディジタルツインの予測誤差の影響

車両の遠隔制御において，現実の状態をディジタルツイン上で完璧に再現するのは非常に困難である．2 つ目の検討として，走行車の走行速度またはステアリング角の挙動に関するディジタルツインのモデリング誤差が走行車遠隔制御に及ぼす影響について，シミュレーションによって定量的に評価するとともに，実車による実験で確認した．それらの結果から，経路追従走行においてディジタルツインに必要な精度について述べる．この章は研究業績の論文[13]の一部と論文[15]に相当する．

・第 7 章　ジッタバッファにおける制御信号バッファリング時間の動的最適化

3 つ目の検討として，制御信号のバッファリング時間を制御信号の伝送遅延変動特性の変化に応じて動的に最適化する仕組みを考案するとともに，その有効性をシミュレーションによって定量的に評価した．その結果から，クラウドサーバによる小型走行車の遠隔制御におけるバッファリング時間動的最適化の有効性について述べる. この章は研究業績の論文[12]と[16]に相当する.

・第 8 章　総括

本研究内の 3 つの検討結果をまとめるとともに，ディジタルツインを適用したクラウドサーバによる小型車両の遠隔制御システムの実現化に関する今後の課題について述べる.

第 2 章　関連研究

2.1　緒言

本章では本研究の内容に関連する研究および技術動向について取り上げるとともに，本研究の位置づけについて述べる．2.2 節では，車両の遠隔制御方式に関する先行研究，実証実験等を述べる．2.3 節では，伝送遅延等のむだ時間を含むフィードバック制御を安定化させる方法についての先行研究を述べる．2.4 節では，ディジタルツインの提唱およびその活用に関する先行研究および実際の企業の取り組みについて述べる．2.5 節では，それら先行研究等を踏まえた本研究の目的を述べるとともに，研究目的を達成する上での本研究の位置づけについて述べる．

2.2　車両の遠隔制御方式

無線通信技術の発展に伴い，車両の遠隔制御についての研究が進められている．遠隔手動操縦は危険な環境における車両等機械の運転を安全化する，あるいは運転手の勤務形態をより柔軟化するための技術として注目されている．人間が車両を遠隔操縦する場合は，操縦動作に伴う制御信号の送信と映像等のフィードバック受信の双方に発生する伝送遅延によって正確な制御が困難になる．文献[41]の実験において，伝送遅延が大きい場合は車両を正確に遠隔手動操縦することが非常に困難であることが実証された．3GPP では，車両の遠隔手動操縦への要求仕様としてエンドツーエンド遅延は 5 ms 程度と定められている[42]．これを解決し得る技術として，無線通信区間において URLLC を実現する 5G モバイル通信ネットワーク技術[32, 33]が注目されている．5G を活用した車両等の遠隔手動操縦は様々な企業・研究機関で実現化に向けた実証実験・試験運用が実施されている[43, 44]．車両の自動制御はモビリティ社会における労働力やエネルギの効率化を達成する技術として様々な分野から注目されている．自動制御車両についての研究は，車両自体にカメラ・センサ，および制御器が実装された自律分散制御に関するものが多い. 具体例として, 普通自動車の公道における自律走行[21],

先頭車両を基準とした大型トラックの隊列走行[22]について検討が進められている. 大型車両だけでなく小型車両の自動制御化も期待されている．例として，公共空間における自動車椅子[24], 街区や住宅地でのラストワンマイル配送[37], オフィスビルや工場における物品配送・整理[26]等において自律分散制御方式の小型車両の実用化が検討されている.

より効果的に車両の自動制御を実現する方法として，クラウドサーバまたはエッジサーバを活用した遠隔制御が検討されている．図 2-1 にクラウドサーバとエッジサーバの概要を示す．クラウドサーバは，分散リソースを活用した高い演算性能やアプリケーション機能を通信ネットワークを介してエンドユーザに提供するクラウドコンピューティングの利用形態である[35]. クラウドサーバでインターネットを介して広範囲にサービスやインフラストラクチャを提供することで，ユーザがデバイスの種類や場所に関係なくデータへアクセスまたはアプリケーションを使用することを可能とする．サービス等の開発，更新および維持管理はサーバ側で一元的に実施される．以上の特徴から，クラウドサーバの活用には運用コストの低減，柔軟なサービス提供，およびスケーラビリティの向上[45]という利点がある．しかし，車両遠隔制御にクラウドサーバを活用する場合，制御が不安定になりやすいという問題点がある．クラウドサーバがインターネットを介して制御車両と通信する場合，高速かつ安定した通信を確保することは難しい．そのため，遠隔手動操縦の場合と同様に，伝送遅延によって正確な制御が困難になる可能性が高い．一方でエッジサーバは，エンドユーザとの間の伝送遅延を極力小さくすることを目的としたエッジコンピューティングの形態である[2]．エッジサーバはクラウドサーバよりも演算性能が低くサービスのカバーエリアが小さいが，ユーザに対するアプリケーションのリアルタイム応答速度を短くすることが可能である[46]. 特にモバイルエッジサーバはインターネットを介さずモバイルネットワークでサーバと端末との間の通信を行うため，低遅延かつ安定した通信を実現することが可能である．しかし，サービスのカバーエリアを広げるためには各ユーザの場所にサーバを分散配置しなければならない. 車両の遠隔制御の場合，車両が各エッジサーバのカバーエリアを行き来する際にユーザ情報がサーバ間で正しく処理されるようにシステムを構成しなければならない．また，どちらの

形態のサーバを車両遠隔制御に活用する場合でもダウンタイム[47], すなわち機器の故障, サイバー攻撃等に起因する動作停止への対策が必要である.

文献[31]の研究では, クラウドサーバまたはモバイルエッジサーバによる車両の遠隔制御について検討されている. 実験により, クラウドサーバを用いた場合は車両との間の往復伝送遅延が 150 ms である場合に正確な経路追従走行が困難になることが確認された. 一方で, ネットワーク上の往復伝送遅延に応じて制御をクラウドサーバとモバイルエッジサーバとの間で切り替える協調システムを構築した結果, モバイルエッジサーバのみで車両を制御する場合に近い制御精度を達成できた. 文献[48, 49]の研究では, 2 種類の制御サーバを活用した車両遠隔制御におけるサーバの配置について検討されている. 制御サーバの 1 つはモバイルエッジサーバであり, もう 1 つはクラウドまたはモバイル通信ネットワークの上層に配置されるエッジサーバである. 広範囲の車両の情報を収集・分析する役割を担うエッジサーバをモバイル通信ネットワークの基地局に近い位置に配置する場合, 伝送遅延の短さから車両の制御は安定するが制御下の車両台数は少ない. 一方でクラウド上に配置する場合, 制御車両台数は多いが伝送遅延が長くなる. シミュレーションによって, 伝送遅延と制御車両台数のトレードオフを解決するエッジサーバの配置について定量的な評価が得られた. 文献[50]では, 車両の遠隔協調制御システムに 5G を適用した場合の制御特性について検討されている. 5G 通信はモバイルエッジサーバと車両が接続するルータとの間に構成された. 実験によって, 5G 通信の場合は有線であった場合と比較して殆ど同じ精度の車両遠隔制御を達成できることが示された.

文献[51]では, 通信ネットワークの信頼性が不十分な状態で安定した車両遠隔制御を実現することを目的としてマルチパスを利用した遠隔制御方法が提案されている. 複数の独立した通信チャネルを遠隔制御の走行車に利用することは, 通信チャネルの途絶およびパケット伝送遅延の急激な増加への対策として有効であることが示されている.

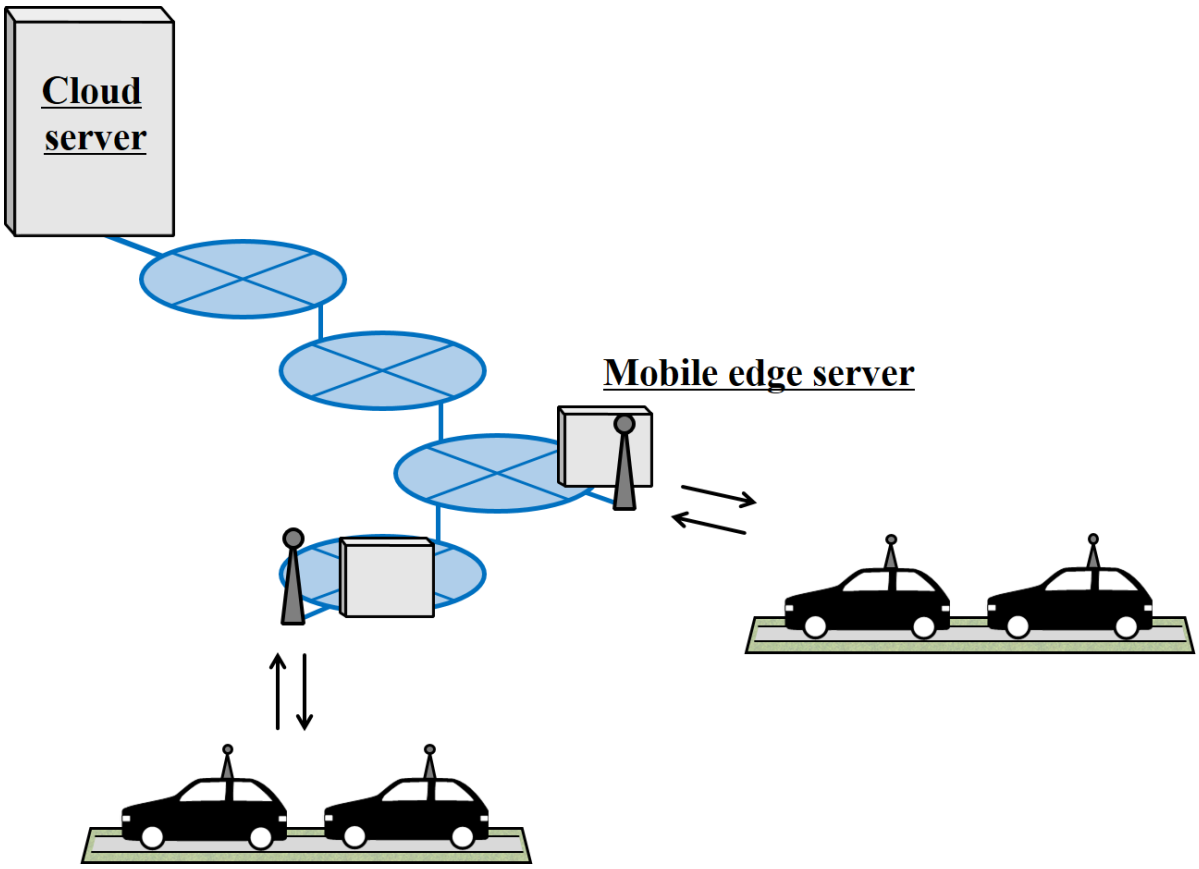


図 2-1　クラウドサーバとモバイルエッジサーバのネットワーク上の配置例

2.3　フィードバックループの安定化

一般的にフィードバック制御は制御器，駆動部，制御対象およびフィードバック要素で構成される[52]．図 2-2 にその基礎的な構成を示す．制御器は与えられた目標値を達成するための入力値を制御対象に与える．制御対象の内部では駆動部により入力値が操作量に変換され，対象が動作する．制御対象を測定することで得られた出力値はフィードバックされ，目標値と比較した偏差値が求められる．フィードバック制御では，その偏差を 0 に近づける，または 0 にすることを目標とする．制御器と制御対象との間の閉ループにおいて伝送遅延等の外乱が大きい場合には，安定した制御を達成することが困難となる．フィードバック制御においては，アプリケーションに要求される性能を安定して発揮し続けることが重要である。そのため，遅延等外乱による影響を克服する研究が行われている．

フィードバックループにおける伝送遅延は制御理論上でむだ時間と呼ばれる[53]. むだ時間を含む制御系を安定させる一般的な方法としてスミス予測器が挙げられる[54]. スミス予測器はむだ時間が無い状態における制御対象の出力値を予測するものである．制御器とスミス予測器との間で閉ループを構成することで，フィードバックループ中のむだ時間の影響を低減した制御を実現する．スミス予測器から発展した制御方法であるモデル予測制御は広い分野で活用されている．モデル予測制御では，入力値に対する出力値のインパルス応答モデル，ステップ応答モデル, カルマンフィルタ[55]を用いて分析した状態空間モデル等を応用することで，むだ時間が経過した後の制御対象の状態を予測する．実例として，制御ループが比較的遅い石油精製や化学プラントにおいてモデル予測制御が活用されている[56, 57]．またロボットの制御を安定化させる方法としても有効である[58].

通信ネットワークを介した遠隔制御においては制御信号とフィードバック信号の送受信にネットワークによる伝送遅延が生じる．これらの遅延もむだ時間の 1 種である．リアルタイムかつ高精度な遠隔制御を追求する場合，伝送遅延の変動特性を考慮した制御方法が必要がある．文献[59]では，低遅延状態と高遅延状態の 2 状態をマルコフ過程で確率遷移させる方式で実際の伝送遅延を予測する方法が提案された．文献[60]では，倒立振り子の遠隔制御におい

てモデル予測制御を適用する手法について検討されている．この研究では，制御信号の伝送遅延の変動をカルマンフィルタを用いて予測することで安定した制御を達成できることが示された．車両の遠隔制御に関しても，伝送遅延を予測した状態予測制御の有効性が検討されている．文献[61]では，ルータのキューイングメカニズムを考慮して入力値の伝送遅延を予測することで，小型車両の遠隔制御を安定化させる方法が検討されている．文献[62]では，状態予測制御とジッタバッファを活用した車両の遠隔自動制御システムが提案されている．ジッタバッファとは伝送遅延ジッタを吸収して制御遅延を一定化させるもので，通信を介したアプリケーションの動作を安定化させる効果がある[63]．この研究では，車両の走行状態を正確に予測できる場合，フィードバックループ内に 3,000 ms の片道伝送遅延が存在する場合でも良好な車両遠隔制御を達成できることが示された．

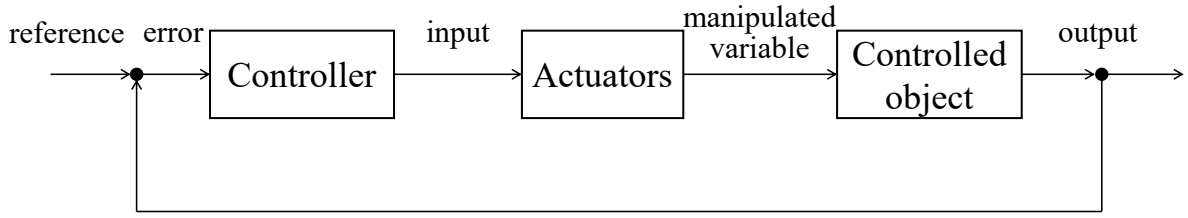


図 2-2　フィードバック制御系の基本的構成

2.4　ディジタルツインコンピューティングを活用したアプリケーション

ディジタルツインは，現実空間をセンシングした情報を収集してコンピュータ内で現実空間の状態を再現する技術体系，または再現された空間そのものの呼称である．ディジタルツインは 2002 年に，工業製品のライフサイクルマネジメントを効率化する概念として提唱された[4]．近年は様々な分野でディジタルツインの活用方法が検討されている．

文献[64]では，自動車産業において期待されるディジタルツインの活用方法および今後研究を要する基幹的技術について解説されている．ディジタルツインによる高度なシミュレーションによって，製品の設計段階における機能確認，製造工程の遠隔監視・制御，製造後の各ユニットの状態監視・劣化予測およびメンテナンス計画の立案を効率化することが期待されている．ディジタルツインを活用したこれらのアプリケーションを実用化するためには，制御対象の高度な 3D モデリング，センシング情報の収集・統合的分析，および現在または未来の状態のシミュレーションを実行する機能が必要である．一部の自動車製造･販売企業では，実際の製造工場をモデリングすることによる製造環境のディジタルツイン化や，世界中の自社生産車両の走行状態を収集・分析して最適なメンテナンス計画の立案・提案を実施する統合的プラットフォームの構築が検討されている[8, 9]．文献[7]では，ディジタルツインを用いた製造現場のオートメーション化および遠隔監視について検討するため，マニピュレータ，小型自律移動ロボットおよび簡易的なディジタルツインを構成するサーバ用 PC で構成される運搬システムが試作された．その試作システムでは，ディジタルツイン上で各移動ロボットや運搬物の位置，通路上の障害物等を含めた作業空間の状態が再現される．ディジタルツイン上の状態は，運搬すべき荷物の有無の判定および運搬を実行する移動ロボットの走行経路の指定を行うために参照される．実験により，そのシステムが良好に稼働することが確認された．文献[65]では，製造業における機械の遠隔制御・監視を実現するため，ディジタルツインを活用したシステムのアーキテクチャに関して基礎的な検討が行われている．この研究では，実際の機械の制御状態をディジタルツインで把握するシステムにおけるフィードバック情報の適切な活用方法について議論されている．

ヘルスケアでは，患者の臓器等のモデルデータと生体情報のセンシングによって構成したバイオディジタルツインの活用が検討されている[10]. バイオディジタルツインを利用することで，医者は患者を診療する際に体内の症状をより詳細に把握することが可能となる．バイオディジタルツインを利用した症状の把握は患者と直接対面しない遠隔診療を普及させる技術としても期待されている．さらに，バイオディジタルツインを用いて将来の疾患の発生を予測することで，発病の予防および長期的かつ最適な治療計画の立案が可能となる．また，外科手術においてもディジタルツインの活用が期待されている．文献[66]では，経過時間が長く精神的負担が大きい外科手術において医師を支援するための外科手術用ディジタルツインの実現化について検討されている．この文献では，手術の補助用途としてディジタルツインを活用したソフトウェアを開発することや手術中においてディジタルツイン精度を確保することの必要性が述べられている．

ディジタルツイン技術は新規開発された製品の実用性の確認や知識・技術への習熟を目的としたシミュレーションの手段として活用することができる．その場合，物体のモデリングおよび現実の物理的特性の再現をアプリケーションの要求条件以上の精度で実施する必要がある．文献[67]では工事用機械の遠隔操縦に関する試験を実施する際に，実際の機械の代用としてディジタルツイン上の仮想的な機械が用いられている．文献[68]では，ディジタルツイン技術を活用することで仮想空間上の安全教育・訓練を提供する事例について取り上げられている．

2.5　結言

ディジタルツインは既存のサービスの効率化または新しいサービスを実現する技術として近年注目されている．しかし，走行車のリアルタイム遠隔制御へ活用する研究はない．クラウドサーバによる走行車遠隔制御にディジタルツインを適用すると，クラウドサーバの処理性能と情報の一元的活用を背景として，遠隔走行制御に関する統合的アプリケーションを実現できる可能性がある．本研究では，クラウドサーバによる小型走行車の遠隔制御システム

の実現化を研究目的としている．その基礎的段階として，本研究ではディジタルツインの適用を前提としたクラウドサーバによる遠隔走行制御方法の提案およびその有効性の検討を実施した．本研究では小型走行車の走行環境を簡易的なディジタルツインで再現した．研究目的を実現するためには，本研究の成果を基として，実用的な精度の遠隔制御を行うために必要なディジタルツインの構成およびセンシング情報について検討する必要がある．また，走行空間を模擬するために配置するカメラ・センサを最適な配備方法についても検討する必要があると考えられる．

前述の通り，走行車の遠隔制御においては一般的に伝送遅延が大きい場合安定した制御は困難となる．先行研究では，実用的手段として伝送遅延が小さいモバイルエッジサーバを活用する方法が検討されている．しかし，より経済的かつ利便性の高い車両の遠隔自動制御サービスを実用化することを考慮すると，より広い範囲の走行車の情報を統合的に収集・処理できる方式の方が望ましい．即ち，クラウドサーバを用いた遠隔制御で走行車を安全に制御できる方法について検討する価値があると考えられる．インターネットを介した通信を前提とする場合，遠隔走行制御の安定性という観点から無視できない大きさの伝送遅延および遅延ジッタが発生する．先行研究により，伝送遅延の変動を予測または吸収する機能をシステムに実装することで走行車の遠隔制御が安定化することが示されている．本研究では，クラウドサーバを制御用サーバとして活用することを前提として，走行車の遠隔制御システムの提案および評価を実施した．提案したシステムは，伝送遅延の変動を考慮した予測制御を行うように考案されている．

第 3 章　クラウドサーバによる無人走行車の経路追従制御システム

3.1　緒言

本研究の最終的な目標は小型の無人走行車をクラウドサーバによって遠隔集中制御するシステムを実現化することである．そのため，本研究ではクラウドサーバによる走行車の遠隔経路追従制御において，ディジタルツインを活用して伝送遅延の影響を低減する方法を検討した．経路追従走行は走行車の自動制御における基本的な制御機能である．実用的な遠隔走行制御を行うためには，経路追従，障害回避および緊急停止といった複数の制御機能が必要となる．これらの機能のうち，障害回避および緊急停止機能は安全上の観点から各走行車に組み込まれるべきである．特にクラウドサーバを制御器として活用する場合は，通信ネットワークによる伝送遅延によって障害物判定および停止信号の伝送が大きく遅れることで衝突事故を回避することができなくなる危険性がある．一方で経路追従制御については，予測制御の適用等によって制御機能を遠隔化できる可能性があり，クラウドサーバにより実施する場合，制御下の車両および走行環境について統合的な情報処理を行うことで最適な経路探索を行うことも可能である．そのため，より効率的な走行車の経路追従制御が実現できる．

本研究ではクラウドサーバの制御に従って 1 台の走行車が目標経路上を走行するシステムを考案した．遠隔経路追従制御におけるディジタルツインを活用した制御方法の有効性を評価するため，一般的なフィードバック制御の場合とディジタルツインを活用した場合の 2 種類のシステムについて検討した．それらのシステムをシミュレータとして実装し，それぞれの経路追従制御の精度を定量的に評価した．システムの実装においては，プログラム言語として Java を使用した．本章の 3.2 節では，一般的なフィードバック制御，つまり，クラウドサーバが走行車からのフィードバック情報を直接的に参照して制御を行うシステムについての概要を解説する．3.3 節では，システムの各構成要素の具体的な機能についてシミュレータの構成とともに解説する．3.4 節では構成したシミュレータによる無人走行車の走行経路の図を用いて，遠隔経路追従制御における伝送遅延の影響を例示する．

表 3-1　システム構成要素の略称

略称	解説
CS	・クラウドサーバ（Cloud Server）を示す. ・本研究の走行車遠隔制御システムにおいては，車両の走行状態に基づいた制御信号の作成，送信を実施する.
UV	・小型の無人走行車（Unmanned Vehicle）を示す. ・CS からの制御信号に従い走行する.
NE	・ネットワークエミュレータ（Network Emulator）を示す. ・シミュレーションにおいて，インターネットおよび無線アクセスを介した CS と UV の間の通信における伝送遅延を模擬する.

なお，本文以降ではシステムおよびシミュレータ上の主な構成要素について表 3-1 に示す略称を用いて解説する.

3.2　制御システムの概要

ディジタルツインを活用して伝送遅延の影響を低減する走行車の遠隔経路追従制御システムを評価するため，まず一般的なフィードバック制御に基づいて CS が UV を遠隔制御するシステムを考案した．図 3-1 にそのシステムの概要を示す.

CS と UV はインターネットおよび無線アクセスを介して通信する．UV は CS からの制御信号に従って走行するとともに，一定周期ごとに自身の位置と車体の向きを検出し，走行状態の情報として CS へ送信する．CS は UV が目標経路の始点から終点へ到達するまでの間，以下の(1)から(4)に示す処理を順に実行する.

(1) UV の制御を始める前に目標とする経路を決定する．また，UV から送信された状態情報を受信し，制御開始時点の位置・向きを認識する．

(2) UV の遠隔制御を開始する．

(3) UV の現在の状態に基づき，UV が目標経路に沿って走行するための制御信号を計算・送信する．この制御信号には，車両の走行速度や前輪のステアリング角の入力値が含まれる．

(4) UV からの状態情報を受信後，(3)に戻る．

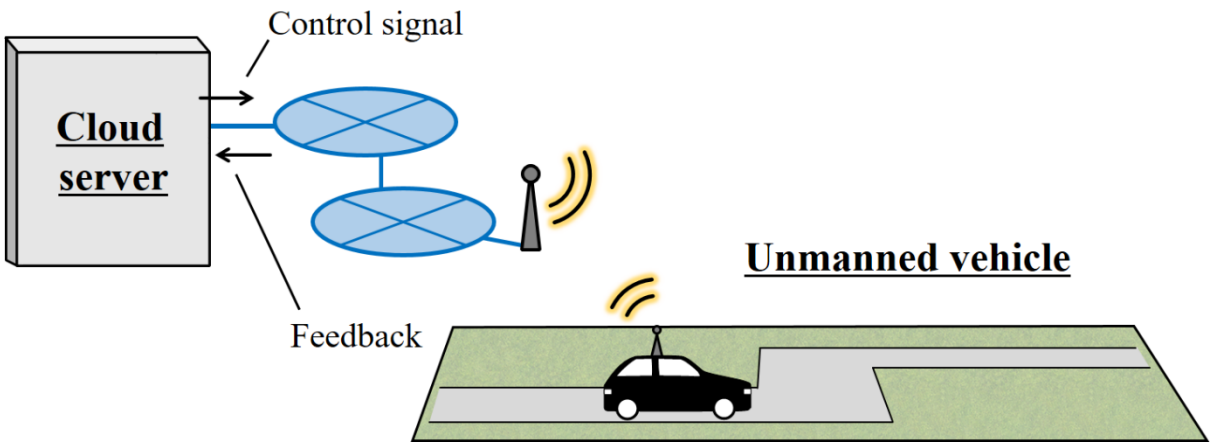


図 3-1　UV 遠隔制御システムの概要

3.3　システムのシミュレータ構成および各構成要素の機能

3.2 節に示したシステムにおける UV の遠隔制御特性を定量的に評価するために，シミュレータとしてシステムを実装した．シミュレータは CS プログラム，UV プログラムおよび NE プログラムを基幹として構成される．図 3-2 にその構成を示す．本研究では，通信ネットワークによる伝送遅延がある状態における UV 遠隔経路追従制御の精度を評価するためにシミュレータを用いた．本論では，目標とする走行経路に沿って UV が走行できた場合に CS が良好な遠隔経路追従制御を達成できたとみなす．本章で解説するシステムのシミュレータは第 5 章の検討で用いた．

以下の 3.3.1 項から 3.3.3 項で，それぞれのプログラムの機能について個別に解説するとともに，実装したシステムの具体的な振舞いについて解説する．

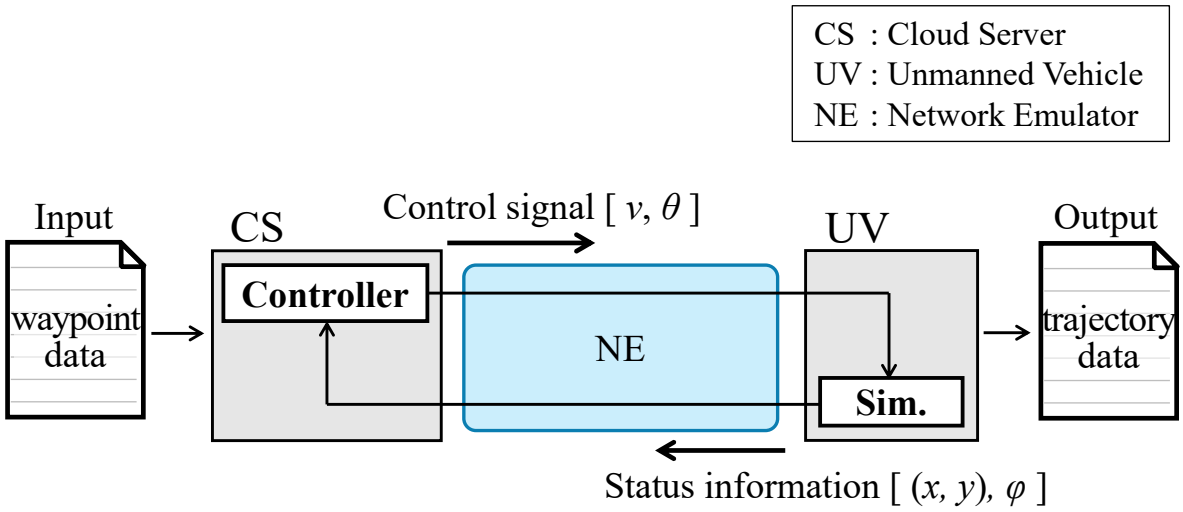


図 3-2　シミュレータの構成

3.3.1 UV

UV プログラムは，小型 4 輪走行車の走行を模擬する．本論では，実際の小型搬送車両を参考に軸距が 0.8 m の 4 輪走行車とした．シミュレーション中，この車両が仮想的な平面空間を走行するとともに，一定周期で自己位置を含めた走行状態の情報を CS へ送信する．

本研究では, CS を制御器として用いた場合の伝送遅延の影響に着目して, CS による UV の遠隔経路追従制御の特性を評価する．伝送遅延による UV の走行経路の劣化を定量的に評価するため，制御信号の伝送遅延が殆ど無い状態では UV が目標経路に沿って正確に走行できるシステムを想定した．具体的には，UV は CS からの制御信号に正確に追従できるものとして構成した．実際の走行車の経路追従制御では，制御信号に対する車両の機械的反応を制御するために様々な技術が存在する．例として，PID (Proportional Integral Derivative) 制御[69]が挙げられる．また，UV は自己位置をごく短時間かつ正確に測定できるものと想定した．位置測定の具体的な技術としては，カメラ画像分析や RTK-GPS (Real Time Kinematic - Global Positioning System) [70]が候補として挙げられる．実際の UV 遠隔制御システムにおいては，制御信号に対する制御方式および位置測定技術の精度はシステムの UV 制御精度に影響する．将来的にシステムの実用化を検討する場合，車体の制御方式および高精度かつ経済的な位置測定技術について評価する必要がある．これらは今後の研究課題とする．

以下に UV プログラムの具体的な機能を解説する. UV の初期状態は仮想平面上の位置を(0, 0)，車体の向きを y 軸方向と設定される．CS から送信された制御信号を受信すると，UV は走行時間 t (ms)における移動座標(x, y)を計算する．この計算は 1 ms ごとに実行される．制御信号には車両の走行速度 v (m/s)と前輪のステアリング角 θ (rad)が含まれる．なお，θ の値が正である場合，UV は車体の右側へ曲がる．移動座標の計算は，以下の①から②までの手順で実行される．

① 図 3-3 に示すように，車体を原点，車体の向きを Y 軸方向としたとしたローカル座標上の移動位置(X, Y)を計算する．計算式は以下の式(3.1)から(3.4)に示す．X と Y の単位は m である．ここで，r は UV の回転半径 (m)，δ は回転角 (rad)を示す．また，WB は

UV の軸距 (m)を示す．本研究では前述の通り WB = 0.8 m とした．式(3.2)と(3.4)内の $\frac{v}{1000}$ は 1 ms ごとの移動距離を示す．

$$X = \begin{cases} r(1-cos\delta) & (\theta \neq 0) \\ 0 & (\theta = 0) \end{cases} \tag{3.1}$$

$$Y = \begin{cases} rsin\delta & (\theta \neq 0) \\ \dfrac{v}{1000} & (\theta = 0) \end{cases} \tag{3.2}$$

$$r = \frac{WB}{sin\theta} \tag{3.3}$$

$$\delta = \frac{v}{r \times 1000} \tag{3.4}$$

② 図 3-4 に示すように，ローカル座標上の移動(X, Y)を走行時間 t におけるワールド座標上の移動位置(x_t, y_t)および車体の向き(φ_t)に変換する．計算式は以下の式(3.5)から(3.7)に示す．φ_t の単位は rad であり，y 軸方向を基準とした右回りを正として車体の向きを示す．

$$x_t = x_{t-1} + Xcos\varphi_{t-1} + Ysin\varphi_{t-1} \tag{3.5}$$

$$y_t = y_{t-1} + Ycos\varphi_{t-1} - Xsin\varphi_{t-1} \tag{3.6}$$

$$\varphi_t = \begin{cases} \varphi_{t-1} + \delta & (\theta \neq 0) \\ \varphi_{t-1} & (\theta = 0) \end{cases} \tag{3.7}$$

UV は一定周期で(x_t, y_t, φ_t)の値を状態情報として CS へ送信する．この情報は UDP (User Datagram Protocol) パケットとして送信される．本章で解説されるディジタルツインが適用されていない UV 遠隔制御システムでは，この周期を 10 ms とした．この値は制御信頼性[71]および通信トラフィック[72]の観点から走行車の制御周期として適切な値である．完成したシミュレータを予備試験として実際に動作させた際，CS と UV の間の通信における伝送遅延が無い場合に正確な UV 遠隔経路追従制御が達成できることを確認した．

UV は CS が遠隔制御を終了すると停止し，初期位置から最後の座標までの移動座標を csv 形式で出力する．第 5 章以降で述べる評価において，CS による UV の遠隔制御特性を評価するためにこの UV 走行経路のデータを用いた．

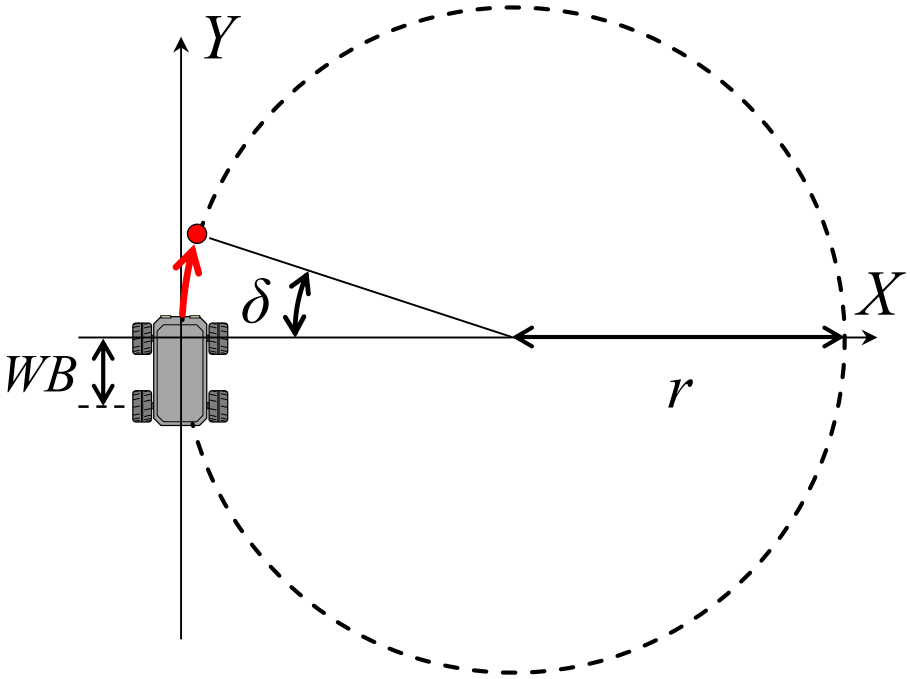


図 3-3　ローカル座標における UV の移動

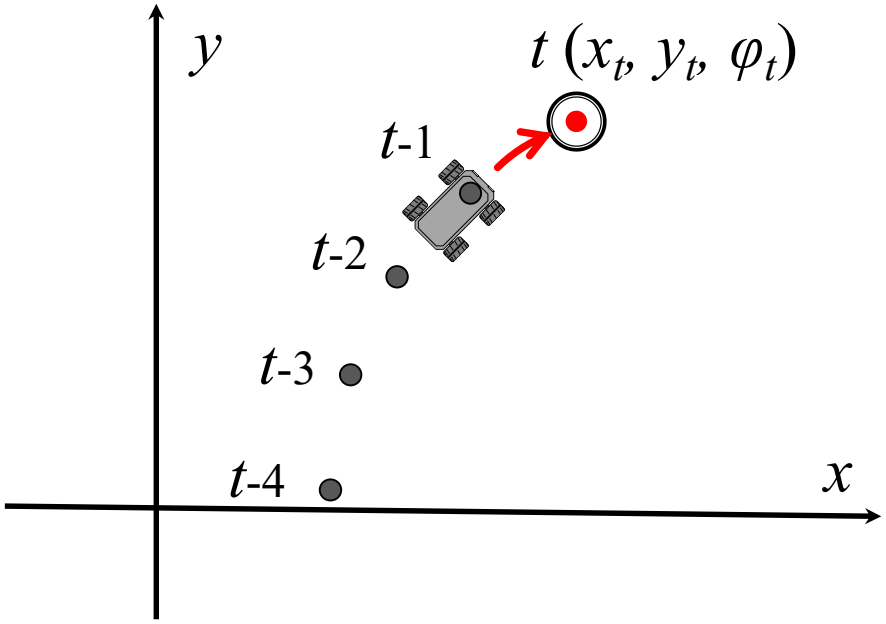


図 3-4　ワールド座標における UV の走行状態

3.3.2 CS

CS プログラムは UV に所定の目標経路に沿って走らせるように制御信号を送信する．CS は UV の制御開始前に目標とする走行経路の情報を与えられる．CS による遠隔制御によって，UV は走行開始位置から目標経路の終点まで走行する．以下に，CS による遠隔制御方法を具体的に解説する．UV の遠隔制御が始まる前に，CS には複数の WP (Way Point) で構成される目標経路が与えられる．各 WP は座標(x, y)の値で指定される．1 番目の WP (WP1) は UV が最初に向かう座標である．CS は UV が WP を順番に通過して最後の WP に到達するように制御信号を計算・送信する．UV が最後の WP に到達したことを認識すると，CS は遠隔制御を終了する．遠隔制御が始まると，CS は WP1 を最初の目標座標とする．そして以下の①から④に示す処理を UV が最後の WP に到達するまで実行し続ける．

① UV が送信した状態情報を受信し，制御対象である UV の位置と向きを認識する．

② 図 3-5 に示すように，目標座標とした WP に UV が到達しているか判断する．図中の WP_i と WP_{i-1} はそれぞれ現在の目標座標と前の目標座標を示す．$i=1$ である場合，$WP_{i-1} = (0, 0)$である．角度 α は UV から WP_i への方向角である．角度 β は WP_i，WP_{i-1} および UV の 3 点による内角である．長さ L は UV と WP_i との距離である．長さ JL は UV が目標座標に到達したと判断する長さである．本研究では，小型走行車が目標座標に十分に接近したと判断させるために，$JL = 0.2$ m と設定した．$L \leq JL$ である場合，CS は UV が目標座標に到達したと判断する．$L \leq JL$ かつ $\beta \geq \pi/2$ である場合，UV が目標座標に到達しなかったが，その横を通過したと判断する．以上の 2 条件のどちらかを満たした場合，CS は新しい目標座標としてWP_{i+1}を定める．

③ UV が目標座標に到達するための制御信号 v, θ を計算し，UDP パケットとして送信する．本研究で UV として具体的に想定する小型搬送車両は一般的に通路上を一定の巡航速度で走行するものである．そのため，本研究では v はシミュレーションごとに一定の値とした．θ は UV が目標座標に円周状の経路で到達するように式(3.8)によって計算される．$\alpha \neq 0$ である場合，円周経路の回転半径 R が式(3.9)によって計算される．

なお，α の単位は rad である．θ の取り得る最大角θ_{max}はシミュレーションごとに設定した．

$$\theta = \begin{cases} \min\left(\theta_{max}, arcsin\dfrac{WB}{R}\right) & (\alpha > 0) \\ 0 & (\alpha = 0) \\ \max\left(-\theta_{max}, arcsin\dfrac{WB}{R}\right) & (\alpha < 0) \end{cases} \tag{3.8}$$

$$R = \frac{L}{2sin\alpha} \tag{3.9}$$

④ ①に戻る．

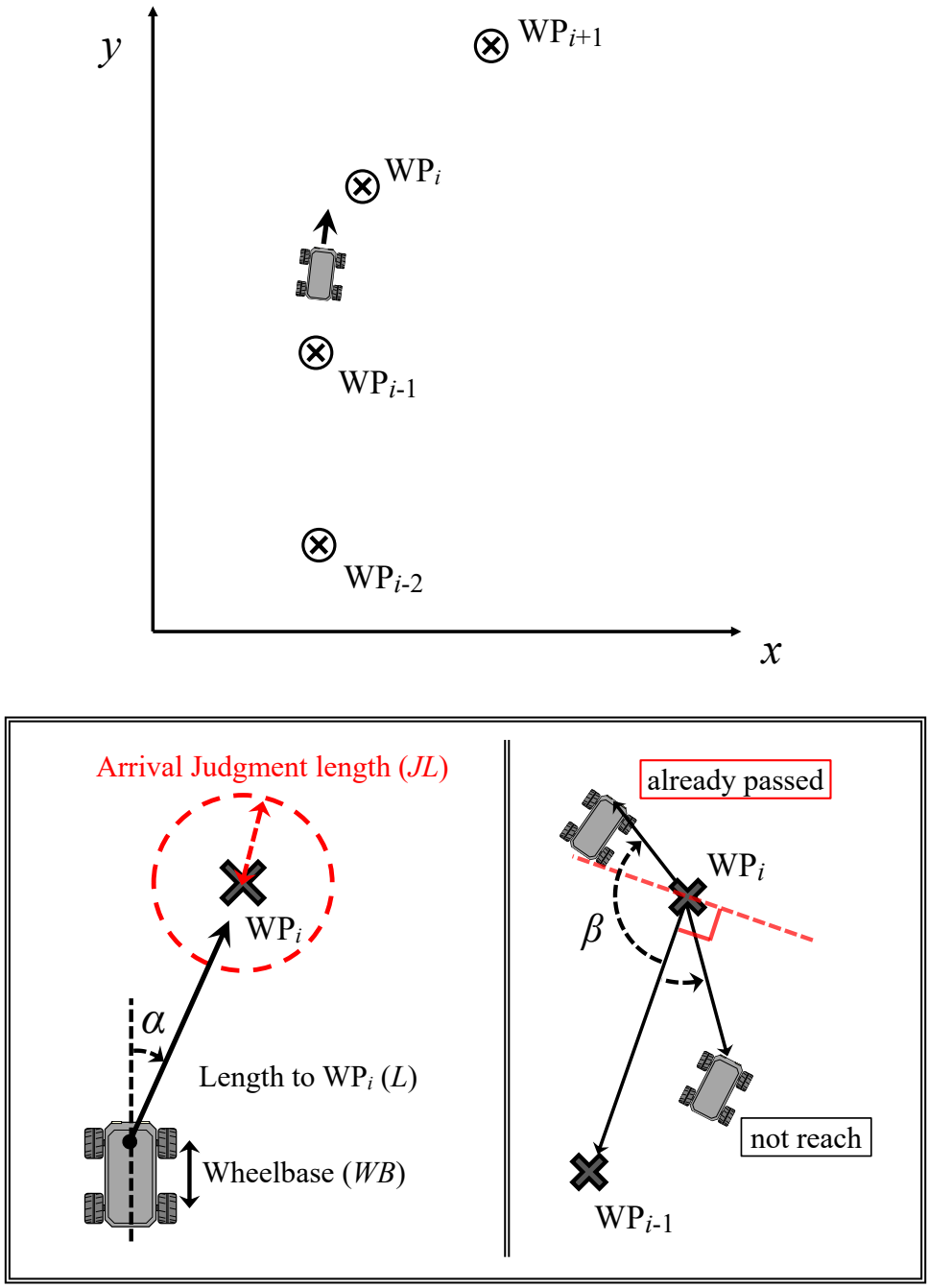


図 3-5　目標座標とする WP への到達判定

3.3.3 NE

シミュレーションにおいて CS と UV との間のパケット通信の伝送遅延を再現するために NE プログラムを作成した．NE にはシミュレーション開始前に所定の遅延データセットが入力される．遅延データセットはパケット伝送遅延時間の値（ms）が時系列的に配列されたものである．シミュレーション中，NE は CS から送られた制御信号のパケットを受信する．受信されたパケットは遅延データセット上の数値に基づいて NE 内で保持された後に，UV に送信される．UV からの状態情報のパケットも同様の処理を受ける．シミュレーション継続中に遅延データセット配列の最後に到達した場合，NE は配列の最初に戻り伝送遅延の再現を繰り返す．また，所定の条件を与えることで CS から UV および UV から CS への通信におけるパケットロスの発生も模擬することができる．第 5 章以降の検討において NE を活用した．それぞれの検討で再現させた伝送遅延の特性については各章で具体的に解説する．

3.4 UV の遠隔制御における伝送遅延の影響

3.3 節で解説した CS による UV 遠隔制御はヒューリスティックな手法と言える．完成した UV 遠隔制御シミュレータを用いて，CS による UV の遠隔制御特性を確認した．その結果，CS と UV との間の伝送遅延が無いまたは非常に小さい場合は，指定された目標経路にそって十分に正確な走行が達成できることを確認した．一方で，伝送遅延が大きくなるにつれて UV の制御精度が明確に劣化した．図 3-6 にその例を示す．図中の灰線は WP の配置によって決定される UV 経路追従走行の目標経路を示す．本章で解説したシステムおよびそのシミュレータは，UV の遠隔経路追従制御における伝送遅延の影響を定量的に評価する上で有効である．

CS と UV との間の伝送遅延は，制御理論におけるむだ時間の 1 種として UV のフィードバック制御を大きく乱す．経路追従制御においては，伝送遅延が大きい程 UV が指定された経路を逸脱せずに走行することが困難になる．過去の研究において，伝送遅延が大きい場合は小型走行車の遠隔制御が困難になることが確認されている[30, 31]．文献[30]では，小型走行

車のフィードバック制御において片道伝送遅延が 250 ms を超える場合，安全な走行制御が不可能となることが確認された．仮に UV の状態情報が 250 ms 遅れて伝送される場合，CS は 250 ms 前の UV の走行状態に基づいて制御信号を計算・送信する．さらに，制御信号も 250 ms 遅れて伝送されると，UV は 500 ms 前の状態に基づいた制御信号を受信して走行する．その結果，図 3-6 に示すように目標経路から大きく逸脱した走行結果となる．

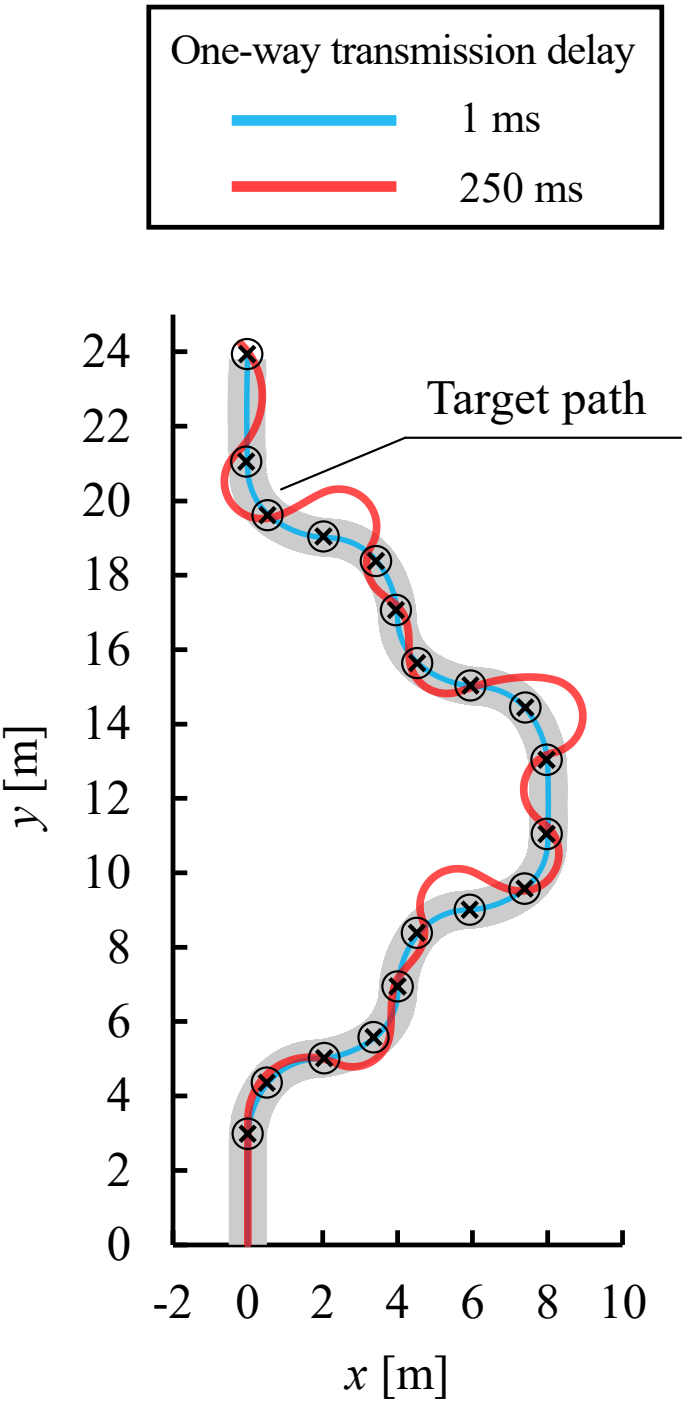


図 3-6　伝送遅延による UV の経路追従走行の劣化

3.5　結言

本章ではクラウドサーバが走行車からのフィードバック情報を直接的に参照して制御を行う, 一般的なフィードバック制御に基づくUV遠隔経路追従制御システムについて解説した. このシステムを実装したシミュレータでは伝送遅延が UV 遠隔制御の精度に及ぼす影響を評価することができる.

物品の自動配送のようなアプリケーションにおいては，安全上の観点から指定経路を逸脱しないようにUVを制御できる能力が必須である. 特に制御器としてCSを活用する場合，一般的なフィードバック制御では安全な遠隔制御が困難となる程の伝送遅延が発生することを考慮して制御システムを考案しなければならない. 第 4 章では，伝送遅延の影響を低減した UV 遠隔制御を達成するために，ディジタルツインをクラウドサーバに適用したシステムについての提案を述べる. 本章で解説したシステムは，第 5 章で解説する検討において第 4 章の提案システムの有効性を定量的に評価する比較対象として用いた.

第 4 章　無人走行車の経路追従走行システムへのディジタルツインの適用

4.1　緒言

ネットワーク上のサーバを制御器として用いる場合，UV の遠隔制御のために必要な情報を一元的に処理することができる．CS の場合，通信を介した伝送遅延が大きい代わりにより広範囲・多数の UV の制御を集約化することができる．CS に UV とその走行空間を模擬するためのディジタルツインを適用すると，多数の UV を効率的かつ低コストに制御するだけでなく，各 UV の位置や走行経路の環境等の情報を統合的に分析にすることで最適な経路探索を行うことができる．また，車体が故障する可能性を評価し，効率的な整備計画の立案または整備の必要性の通知を行うことができる．そのような統合的アプリケーションを実現化するために，ディジタルツインを活用した UV 遠隔制御方法を考案し，その有効性を定量的に評価する必要がある．

第 3 章で示したように，一般的なフィードバック制御によって遠隔経路追従走行を実施する場合，CS では安定した UV 走行制御を行うことができない．伝送遅延が無い場合に正確な UV 制御を達成できるシステムがあると仮定しても，制御機能を CS に移すと通信による伝送遅延の影響によって制御精度が大きく低下する．制御理論では，むだ時間が大きなシステムにおいて安定した制御を達成するためには予測制御が有効とされている．倒立振り子の遠隔制御では，状態予測制御を用いることで伝送遅延がある状態でも安定した振り子の制御を行うことができることが示されている[60]．本研究では，ディジタルツインを用いた予測制御によって CS が UV の遠隔経路追従走行を制御するシステムを提案する．そのシステムは，第 3 章で解説した制御システムをベースとして，CS と UV に機能を追加したものである．提案システムの考案の際，通信ネットワークによる伝送遅延とその変動が大きい場合でも安定した UV 遠隔制御を達成することを目標とした．また，そのシステムによる UV 制御特性を定量的に評価するため，シミュレータとしてそれぞれの機能を実装した．

本章の 4.2 節では，ディジタルツインを活用して伝送遅延の影響を低減する UV 遠隔制御

システムについての概要を解説する．4.3 節では，追加した構成要素の具体的な機能をシミュレータの構成とともに解説する．4.4 節では，伝送遅延に対する実装した提案システムの有効性についてまとめる．ただし，その有効性の検証・確認については以降の章で解説する．なお，第 3 章と同様に Java を使用してシステムを実装した．参考に，本論の付録として CS プログラムのソースコードを添付する．

4.2　制御システムの概要

図 4-1 にディジタルツインを適用した UV 遠隔制御システムの概要を示す．第 3 章で示した UV のフィードバック制御システムを基として，CS には UV の走行空間を模擬するためのディジタルツインを適用した．また，UV には制御信号の伝送遅延の変動を吸収するためのジッタバッファを適用した．以下にディジタルツインとジッタバッファが適用された UV 遠隔制御システムの仕組みを解説する．

ディジタルツインは制御対象である UV の挙動とその走行空間を模擬する．ディジタルツイン内の UV は実際に走行する UV をモデル化したものである．CS は実際の UV へ制御信号を送信する際に，ディジタルツイン内の UV モデルに対しても同じ制御信号を与える．その後，ディジタルツインを用いて制御信号を受信した UV の走行状態をシミュレーションによって予測する．新しい制御信号を計算するため，CS は UV からの状態情報の代わりにディジタルツイン上の予測結果を用いる．これにより CS は UV の状態予測制御を実行する．また，安定した周期で制御信号を計算・送信することができる．

ディジタルツインを用いた UV の予測制御において，実際の UV の挙動を完全に予測することは難しい．言い換えれば，実際の UV とディジタルツイン上の UV モデルとの間で走行状態の誤差が生じる．実際の UV は様々な影響を受けながら走行する．例えば，路面の状態や車体の整備不良によって，走行速度やステアリング角が制御信号とは異なる状態で走行する可能性がある．実際の UV 走行状態に対するディジタルツインの予測誤差を補正するため，CS は UV からの状態情報を活用する．この補正は状態情報を受信する度に行われる．

ディジタルツインの適用によって，UV の経路追従走行に関する CS の処理が変化する．以下の(1)から(5)にて CS が実行する処理を順に述べる．これらの処理は UV が終点へ到達するまで繰り返される．

(1) UV の制御を始める前に目標とする経路を決定する．また，実際の UV から送信された状態情報を受信し，ディジタルツイン上の UV モデルの初期位置・向きを設定する．
(2) UV の遠隔制御を開始する．
(3) ディジタルツイン上の UV モデルの状態に基づき，UV が目標経路に沿って走行するための制御信号を計算・送信する．
(4) (5)を実行する前に UV からの状態情報を受信した場合，実際の UV の状態に基づき，ディジタルツインによる UV の走行状態の予測結果を補正する．
(5) (3)から一定時間が経過した後，ディジタルツインを用いて制御信号を受信した後の UV の走行状態を予測する．その後(3)に戻る．

UV の遠隔制御において，通信ネットワークを介した伝送遅延の変動は制御周期の乱れを生じさせる直接的原因となる．仮に CS と UV との間の伝送遅延が常に一定である場合，CS の制御信号は一定の時間（D）が経過した後で UV に実行される．そのため，ディジタルツイン上で時間 D 後の UV の状態を予測すればよいので，安定した予測制御を達成するのは容易である．しかし，インターネットを介した通信においては制御信号の伝送遅延の変動は避けられない．また，その変動特性を予測することは困難である．受信する制御信号に遅延ジッタが生じると，UV は一定の周期で制御信号を受信・実行することができなくなるため，走行状態が不安定化する．また，制御信号が受信されるタイミングが不安定になることで，実際の UV と CS による予測との間の誤差がさらに大きくなる．遅延ジッタによるシステムの不安定化はネットワーク制御システムにおいて一般的な問題である．文献[63]より，ネットワーク制御システムの安定性を向上するには，適応的なバッファによって伝送遅延ジッタを小さくすることが有効であることが示されている．

ジッタバッファは UV が受信した制御信号の伝送遅延ジッタを吸収する．制御信号の伝送遅延は信号ごとに異なる．UV は受信した制御信号の伝送遅延が一定となるように，ジッタバッファ内で信号を保持する．例えば，ジッタバッファによって伝送遅延を 200 ms に均一化するようにシステムを定義したとする．この場合，制御信号が伝送遅延 10 ms，70 ms，150 ms で UV に伝送されると，ジッタバッファを介することで各信号は CS から送信された 200 ms 後に UV で実行される．各信号の伝送遅延は増加するが，伝送遅延変動の吸収によって CS による遠隔制御は安定化する．上記の遅延変動の吸収のためには，CS と UV との間でコンピュータ上の時刻が一致していることが望ましい．本研究では CS と UV の時刻が一致している前提で検討を行った．時刻同期の手段としては，高精度な時刻同期プロトコルの適用が考えられる．あるいは，ジッタバッファによる遅延変動吸収によって，CS と UV との間の時刻差の問題をある程度解消できる可能性がある．提案システムにおける時刻同期の必要性およびその方法については，今後の研究課題とする．

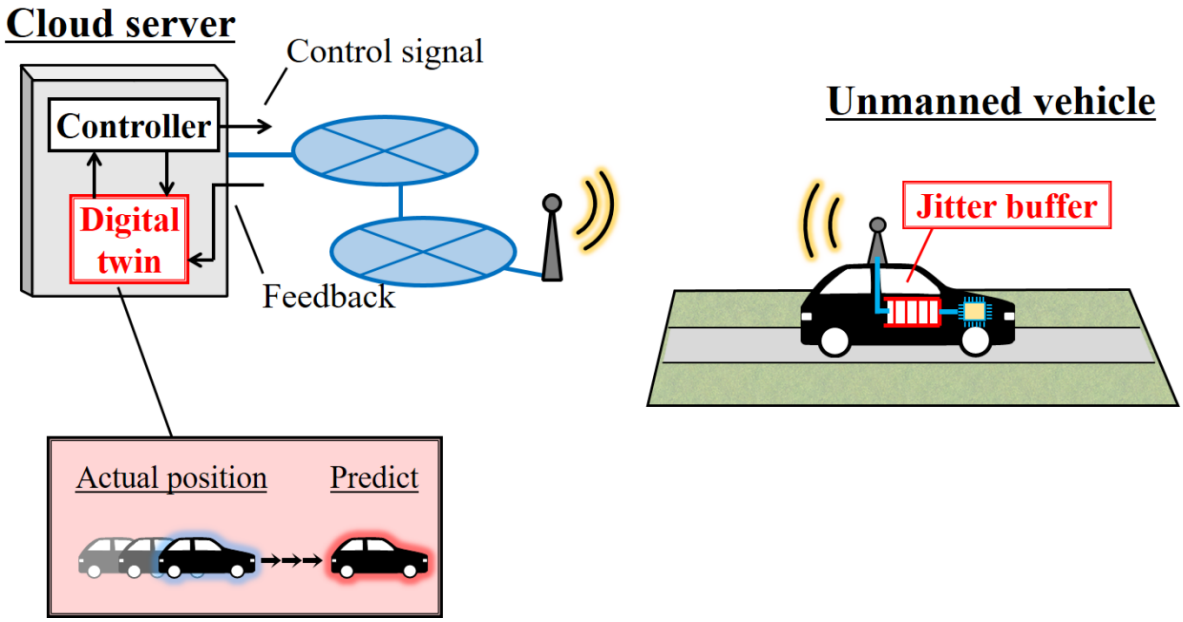
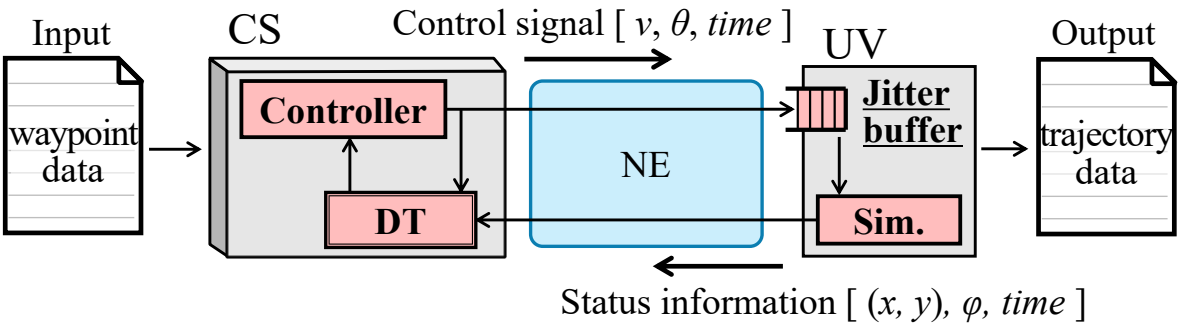


図 4-1　UV 遠隔制御システムへのディジタルツインとジッタバッファの適用

4.3　システムのシミュレータ構成およびディジタルツインとジッタバッファを活用した制御機能

4.2 節にディジタルツインを活用した UV 遠隔制御システムの概要を解説した．この提案システムにおける UV 制御特性を定量的に評価するために，図 4-1 の概要図で示したシステムをシミュレータとして実装した．このシミュレータは提案システムと同様に，第 3 章で示したものを基として機能を追加したものである．この際，簡易的なディジタルツインを C S に含まれる機能として実装した．NE プログラムによる伝送遅延再現，UV プログラムの移動位置計算，CS プログラムによる制御信号計算および目標到達判定の機能は第 3 章と同様である．図 4-2 にその構成と機能を示す．本章で解説するシステムのシミュレータは第 5 章から第 7 章で解説する各検討で用いられた．

以下の 4.3.1 項でディジタルツインを適用した CS による予測制御について，シミュレータに実装した機能を解説する．また，4.3.2 項で UV に適用されたジッタバッファによる遅延変動吸収について解説する．



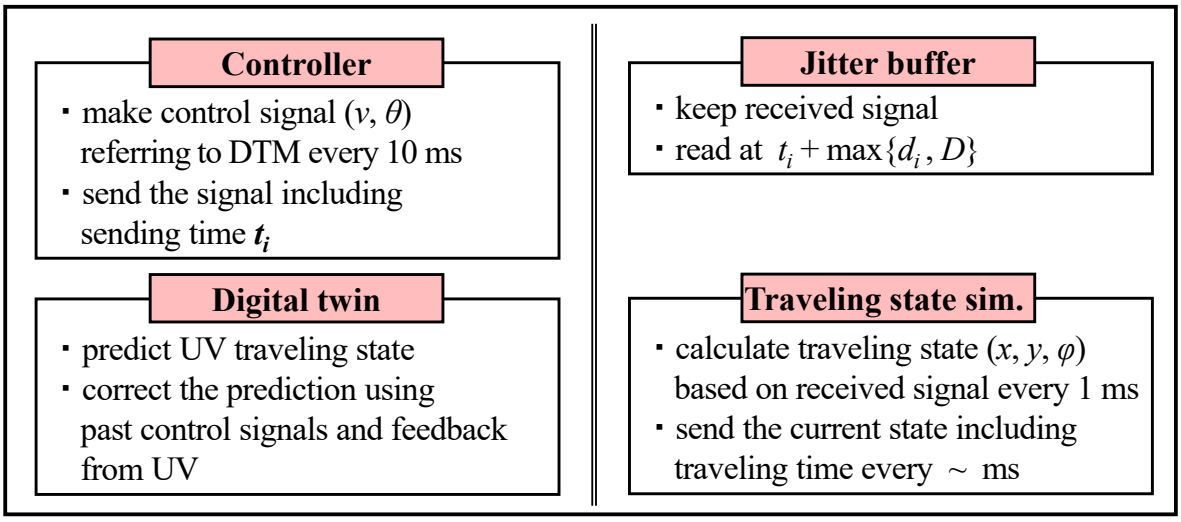


図 4-2　シミュレータの構成および機能

4.3.1 ディジタルツインを活用した予測制御

ディジタルツインとして，CS 内には UV の走行状態を模擬するためのプログラムを実装した．このプログラムは第 3 章で解説した UV プログラムの一部をコピーしたものに基づいて作成したものである．具体的には，制御信号に対する移動位置計算機能が同じである．本研究では，伝送遅延がある状態での UV の経路追従走行を評価するため，制御対象の UV の走行状態を模擬するシンプルなディジタルツインを実装した．CS プログラムは制御信号を作成する制御器とディジタルツインの 2 種類の機能を含む．

CS による UV の制御過程は第 3 章から以下のように変化する．UV の遠隔制御が始まる前に，ディジタルツインにおける UV モデルの状態(x', y', φ')は制御対象の UV の状態(x, y, φ)と同期される．この同期には UV からの状態情報が用いられる．遠隔制御中，CS の制御器はディジタルツイン上の UV モデルの状態を参照し，制御信号を計算・送信する．各制御信号には，走行速度 v とステアリング角 θ に加えて CS の送信時刻 t が含まれる．この時刻は後述するジッタバッファにおいて活用される．送信された制御信号は CS 内で記録される．制御信号の送信後，CS はディジタルツインを用いてその信号を受信した後の UV の状態を予測する．図 4-3 にその概要を示す．制御器は一定周期ごとに UV の走行状態の予測結果を参照し，制御信号を送信する．本研究ではその周期を 10 ms とした．この値は第 3 章で解説した制御システムにおける UV のフィードバック周期である．この時，CS はディジタルツイン上で 10 ms 経過した後の UV モデルの状態を参照する．これによって，CS において制御器とディジタルツインとの間で 10 ms 周期かつ伝送遅延がない状態のフィードバックループが形成される．その結果，CS は安定した周期で制御信号の計算・送信を実行することができる．

ディジタルツイン上の UV モデルを活用した走行状態の予測には，実際の UV の状態に対する誤差が生じる．UV を安全に遠隔制御するためには，この誤差を逐次修正して予測結果を実際の状態に近づける必要がある．UV から送られる状態情報はこの修正のために活用される．状態情報には UV が送信した時の位置・向きに加えて走行時間が含まれる．ここで走行時間とは，本提案システムにおいて UV が最初の制御信号を受け取って走行し始めてから経

過した時間を意味する．状態情報を受信すると，CS は図 4-3 に示すように実際の UV の状態をディジタルツイン上で再現する．図中の(x_j, y_j)，φ_j，t_j は UV の走行時間t_j における実際の UV の位置・向きを示す．また，図 4-4 に示すように CS が記録した制御信号の中で UV がt_j 以降に実行した信号を特定する．これらによって，CS は(x_j, y_j)から移動する UV の状態を再度予測する．この修正は UV からの状態情報を受信する度に実行される．ディジタルツインを適用した制御システムにおいて，UV による状態情報は UV の制御において直接的に参照される情報ではない．そのため，一般的フィードバック制御の場合ほど送信周期を短くする必要がない．シミュレータが完成した後，正確な経路追従走行を達成するために十分な送信周期を確認した．その結果，100 ms 程度の周期で十分であることがわかった．送信周期を 100 ms より短くした場合は，経路追従走行の特性は殆ど改善しなかった．本研究で実施した検討では，100 ms をこの提案システムにおける状態情報の送信周期の最低値とした．

上記のディジタルツインによる予測結果の修正は，実際の UV の位置・向きに応じて処理を行うものである．周期的に送信される状態情報は，実際の UV に対するディジタルツイン上の UV のモデリング精度を高める目的でも活用できる．例えば，制御信号に対する UV の実際の挙動をフィードバックすることで，ディジタルツイン上の UV モデルの特性をより実物に近づけることができる．このような修正機能は，第 6 章で解説する定量的な評価のために本研究では実装しなかった．将来的に提案システムの実現化に向けた検討を行う際には，ディジタルツイン上のモデルを動的に高精度化する方法について検討する必要がある．

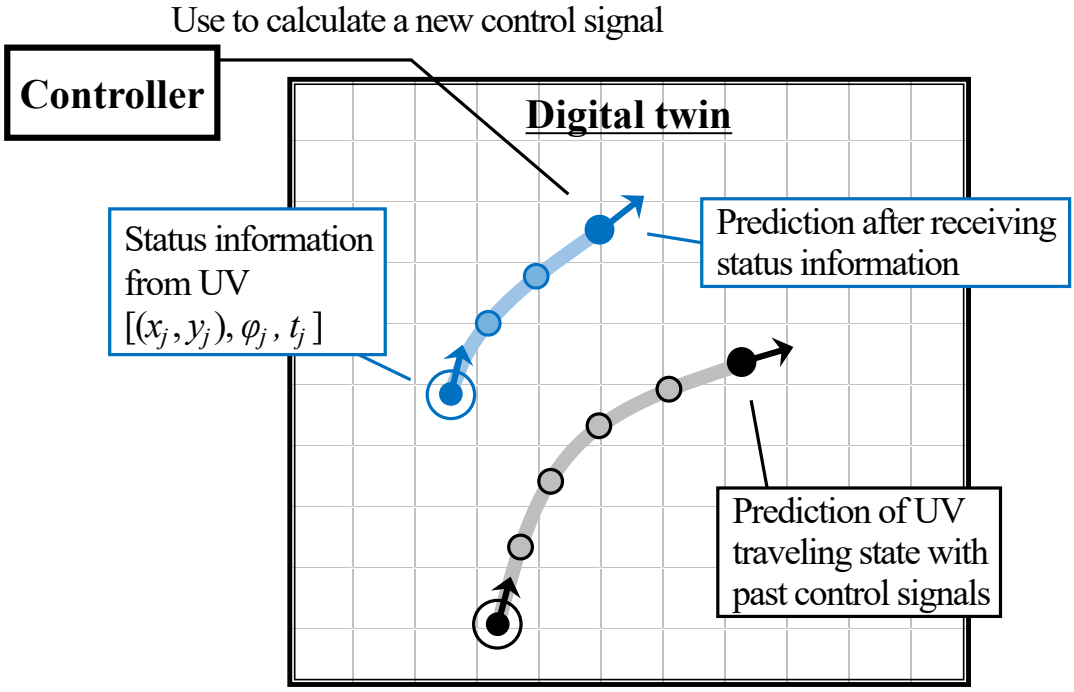


図 4-3　UV 走行状態の予測と状態情報を活用した補正

Control signals recorded in CS

	v [m/s]	θ [rad]	Traveling time [ms]
First signal	1.0	0.000	0
	1.0	0.001	10
	1.0	0.002	20
	⋮	⋮	⋮
	1.0	0.003	100
	⋮	⋮	⋮
	1.0	0.004	200
	⋮	⋮	⋮
Latest signal	1.0	0.005	500

Receiving status information (t_{j-1} = 100 ms)

Receiving status information (t_j = 200 ms)

図 4-4　UV 走行状態の予測および補正に用いる制御信号の例

4.3.2　ジッタバッファによる遅延変動吸収

ディジタルツインを適用したCSのUV遠隔制御を安定化させるため，図4-2に示すようにUV内にジッタバッファを実装した．ジッタバッファはCSからUVへ送られる制御信号の伝送遅延時間を増加させることで，その変動を吸収する．UVからCSへ送られる状態情報にはジッタバッファによる遅延変動吸収が必要ないため，CS側にはジッタバッファを実装しなかった．

図 4-5 にてジッタバッファによる制御信号の伝送遅延変動の吸収を図解する．図中のt_iはCSが制御信号 i を送信した時刻である．d_iは制御信号 i がUVに伝送された時の遅延時間である．D はジッタバッファを介した際の制御信号の最大伝送遅延時間を示すパラメータ値である．この値はシステムのパラメータ値としてUVだけでなくCSにも与えられる．制御信号 i はUVに伝送されると，UVの時刻が$t_i + \max(d_i, D)$となった時にUVに使用される．$d_i \leq D$ である時，信号 i はジッタバッファで伝送遅延時間が D となるまで保持される．$d_i > D$ である時，信号 i は即座に UV に使用される．これによって，各制御信号の伝送遅延は増加するが，伝送遅延の変動を吸収することで制御信号の遅延ジッタを0 msにする，もしくは大きく低減することができる．

D の値はCSと UV の間の伝送遅延の特性に応じて決定するのが望ましい．第5章以降で解説するシミュレーション評価では，様々な伝送遅延を再現して評価を実施した．D の値は再現した伝送遅延特性に応じて定めた．具体的な設定値については各章で述べる．

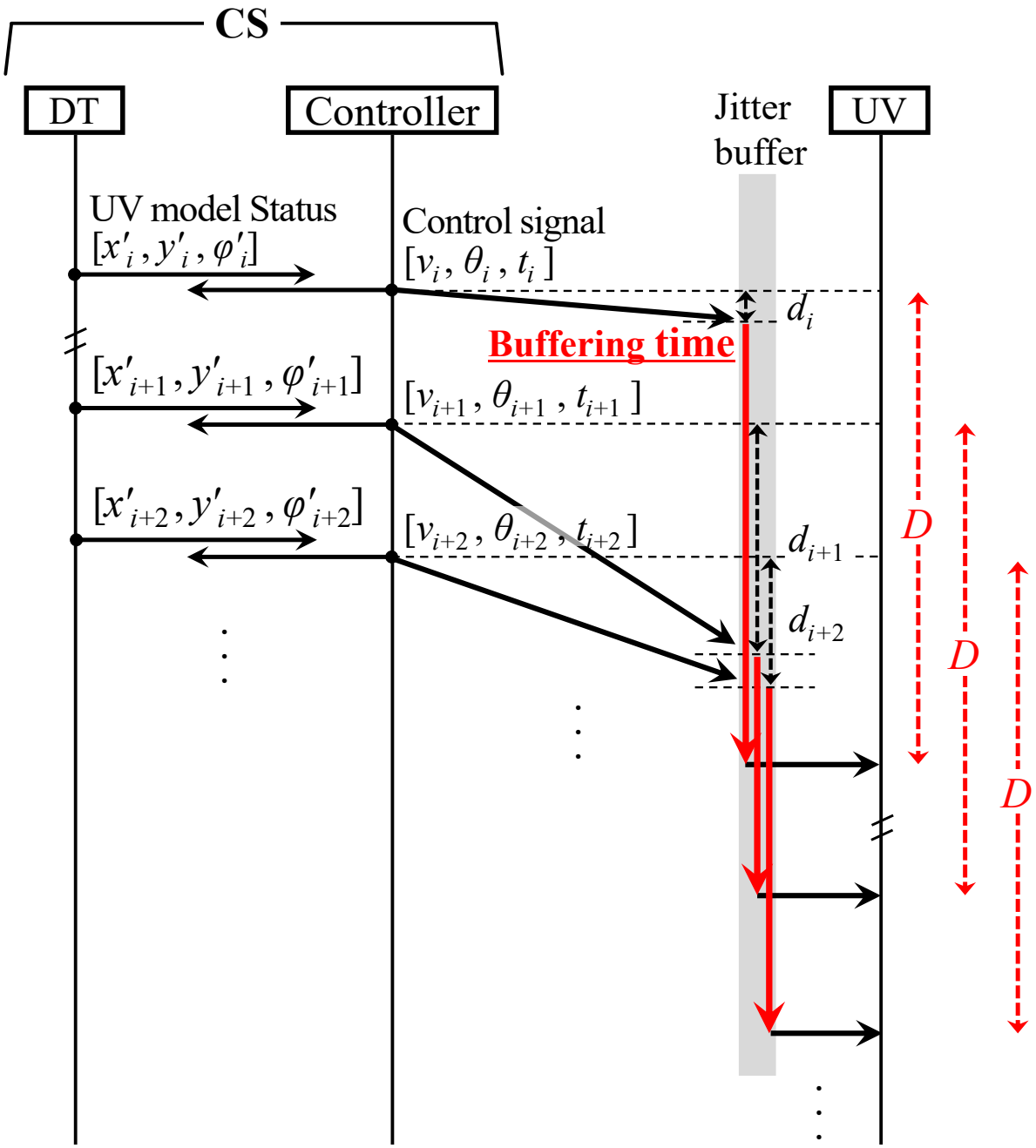


図 4-5　ジッタバッファを介した制御信号の伝送遅延

4.4　ディジタルツインおよびジッタバッファの有効性

小型搬送車両のような UV を遠隔制御するシステムにおいては，目標経路から逸脱しないように走行制御できる能力が安全上の観点から必須である．UV の経路追従走行の正確性や安全性において，CS を制御器として用いる場合，UV との間に無視できない大きさの伝送遅延が生じる可能性がある．さらに，通信ネットワークのトラフィック状況に応じて伝送遅延の変動が生じる．制御機能そのものが正確な UV 走行制御を実現できるものであっても，CS を制御器として活用する場合，伝送遅延およびその変動の影響によって制御精度が低下する．提案システムは一般的なフィードバック制御システムの場合よりも有効であると考えられる．即ち，上記の伝送遅延の影響を低減し，伝送遅延が存在しない状態に近い UV 走行制御特性を達成できる可能性がある．

4.5　結言

本章では，CS による UV の遠隔経路追従制御にディジタルツインを活用する提案システムについて解説した．一般的に，CS を用いた遠隔制御では伝送遅延が制御特性に与える影響が大きい．そのため，制御対象を安定して制御することは極めて困難である．走行車の経路追従走行でも同様の問題がある．提案システムは，ディジタルツインを適用した CS による UV 走行状態の予測およびジッタバッファによる制御信号の伝送遅延変動の吸収によって，UV の遠隔経路追従制御の精度および安定性を向上させることを目的として考案したものである．

UV の遠隔かつリアルタイムな制御におけるディジタルツインの有効性に関して定量的な評価はまだ行われていない．ディジタルツインを適用した CS による UV 遠隔制御アプリケーションの実現性を検討する基礎的段階として，本研究では提案システムの有効性を確認するとともに UV 遠隔制御の特性について検討した．後述する第 5 章から 7 章では提案システムを用いた検討内容について解説する．第 5 章では，まず提案システムの有効性を評価した結果を示す．この検討では UV の経路追従走行をシミュレーションによって比較することで，ディジタルツインを適用した場合のシステムの有効性を定量的に評価した．第 6 章では，制

御対象に対するディジタルツインの誤差の影響について評価した結果を示す．ディジタルツインを用いた遠隔制御では，実際の UV の走行状態に対してディジタルツイン上で予測される走行状態に誤差がある場合，UV 遠隔走行制御の精度は低下する．実際の UV は様々な要素から物理的影響を受けながら経路上を走行する．そのため，UV からのフィードバックを用いてディジタルツイン上の UV モデルをリアルタイムに最適化し続けたとしても，ディジタルツインの誤差を完全に解消することは困難である．この検討では，ディジタルツインの誤差が小型 UV の遠隔制御に及ぼす影響をシミュレーションによって定量的に評価した．また，実際の小型車両を用いた実験によってディジタルツインの誤差がある状態の経路追従走行特性を確認するとともに，ディジタルツインの誤差における許容範囲について評価した．第 7 章では，提案システムにおける制御信号の伝送遅延変動の吸収の最適化について評価した結果を示す．実際のインターネットを介した通信ではネットワーク利用者のアプリケーション使用状況等に応じて伝送遅延の変動特性が急激に変化する可能性がある．ジッタバッファによる制御信号の遅延変動吸収機能は，その変化に応じて動的に最適化されるのが望ましい．この検討では，その動的最適化を実現するメソッドを提案するとともに，シミュレーションによってその有効性を定量的に評価した．なお，第 6 章および 7 章で解説する検討では，それぞれの評価のために必要な機能を本章で解説した UV 遠隔制御システムに追加した．詳細は各章で述べる．

第 5 章　無人走行車の遠隔経路追従制御におけるディジタルツインの有効性

5.1　緒言

本章では，前述の第 4 章で提案したディジタルツインを活用した UV 遠隔制御システムの有効性を定量的に評価した結果について述べる．具体的には，CS と UV の間に伝送遅延がある状況を再現した状態で UV の遠隔経路追従制御のシミュレーションを実施し，UV が走行した経路を評価した．提案システムを評価するため，ディジタルツインとジッタバッファの有無による遠隔制御特性の変化を比較した．

本論の各検討では，CS が UV に指定された目標経路に沿って一定の巡航速度で走行する状況を想定した．安全な UV の遠隔経路追従制御のためには，所定の範囲から逸脱せずに最終到達地点まで走行させることができる制御能力が必須である．言い換えれば，基準とする走行経路からの UV 走行経路の乖離が所定の長さに収まる限り，良好な遠隔経路追従制御が達成できたと言える．CS の制御結果である UV 走行経路を評価するに当たり，目標とする走行経路に対するそれぞれの UV 走行経路の乖離の最大値に着目した．その値を評価指標として CS による UV の走行制御特性を定量的に評価した．

5.2 節では本章のシミュレーション評価の条件を述べる．5.3 節ではシミュレーションの結果である UV 走行経路を評価するために定義した指標について述べる．5.4 節では評価結果とその考察について述べる．

5.2　シミュレーション条件

本検討では，提案システムの有効性を評価するため，制御システムを変更しつつシミュレーションを実施した．5.2.1 項では，制御システムの種類を解説する．遠隔経路追従走行の条件として，UV に追従させる目標経路および CS と UV との間の通信ネットワークを介したパケットの伝送特性がある．5.2.2 項と 5.2.3 項でそれらについて解説する．また，システム上 CS と UV を動作させるために必要なパラメータ値がいくつか存在する．例として，UV の走

行速度，状態情報の送信周期，ジッタバッファによる最大バッファリング時間 D が挙げられる．5.2.4 項ではこれらの設定値についてまとめて述べる．これには既に第 3 章と 4 章で解説した内容も含まれる．

5.2.1 UV 遠隔制御システムの種類

第 4 章で解説した提案システムの有効性を評価するため，以下(i)から(iii)に示す 3 種類の制御システムにおける UV 遠隔制御特性を評価した．

(i) ディジタルツインなし

第 3 章で解説した一般的なフィードバックループ制御に基づく制御システム

(ii) ディジタルツインのみ適用

第 4 章で解説した提案システムのジッタバッファ機能を無効化したもの

(iii) ディジタルツインとジッタバッファを適用

第 4 章で解説した提案システム

5.2.2 目標経路

UV の遠隔経路追従走行における提案システムの有効性を評価する上では，遠隔走行制御がより困難な目標経路が条件として適切である．図 5-1 の黒線で示すスラローム形状の経路を目標経路として定義した．目標経路の選定に当たり，制御信号とフィードバック情報の伝送遅延の影響を大きく受ける経路が適切であると考えた．本章の検討の前の予備的な検証として，第 3 章で解説した UV 遠隔制御シミュレータを用いて様々な形状の経路における UV の遠隔走行制御を評価した．その結果，図に示すスラローム経路において UV の経路追従特性が最も悪かった．スラローム経路を正確に走行するためには走行方向を頻繁に切り替える必要がある．そのため，信号の伝送遅延による制御の不安定化が他の形状の経路よりも大き

い．このような経路においても CS が良好な UV の遠隔経路追従制御を達成できる場合，UV の遠隔制御システムとして有効であると判断できる．

図 5-1 に示すように，UV の経路追従走行の目標とするスラローム経路は 10 個の WP によって構成される．CS は UV が(0, 0)から出発し，WP1 から各 WP を順に通過しながら走行して WP10 に到達するまで遠隔制御を実行する．

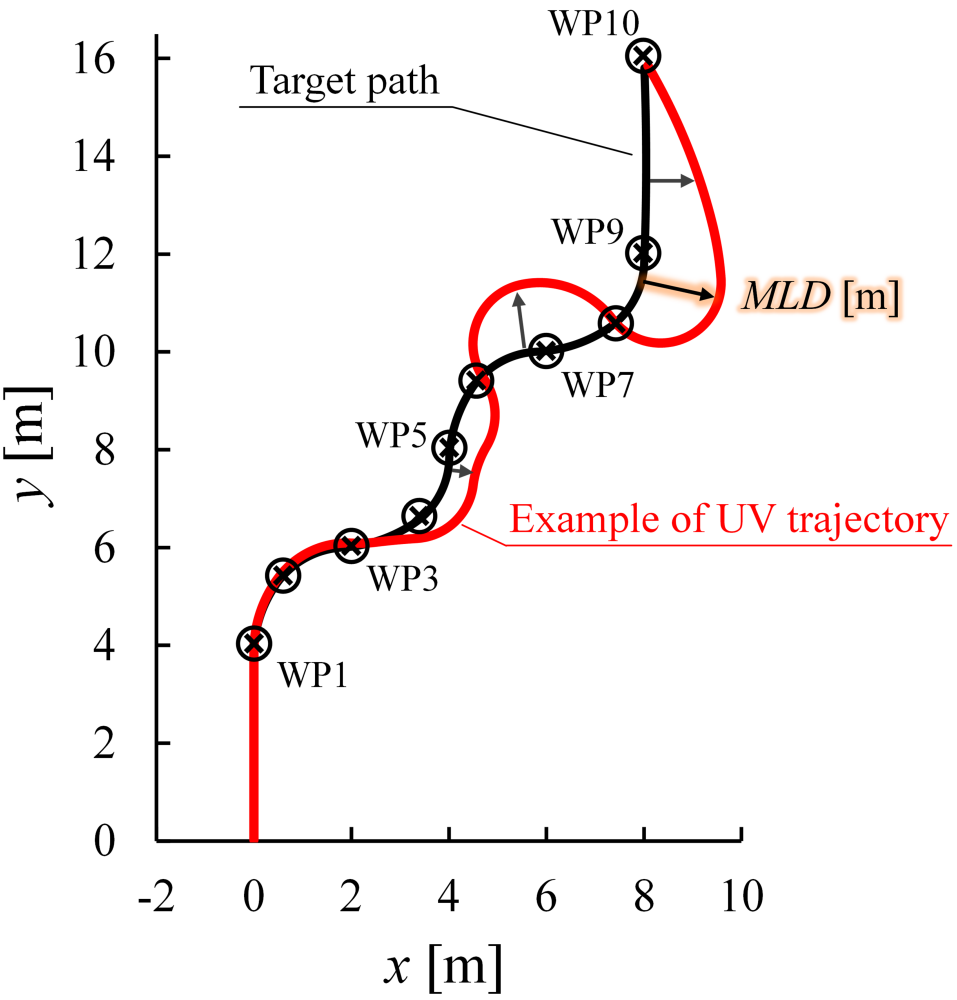


図 5-1　目標経路と評価指標 *MLD* の例

5.2.3　CS と UV との間の伝送特性

図 3-2 と 4-2 に示すように，構成したシミュレータでは NE によってインターネットと無線アクセスを介したパケットの伝送遅延が再現される．NE は所定の伝送遅延のデータセットによって時間変動する伝送遅延を再現する．また，所定の条件に従ってパケットロスを再現する．以下に伝送遅延とパケットロスについて個別に解説する．伝送遅延については，遅延データセットの作成方法とともに，NE で再現される伝送遅延の特性を解説する．

A．伝送遅延

通信ネットワークにおけるインターネット区間と無線アクセス区間の伝送遅延特性を個別に再現するように，NE を作成した．NE はそれぞれの区間の伝送遅延を別の伝送遅延データセットを用いて再現する．それぞれのデータセット上の伝送遅延値を合計することで，NE を通した伝送遅延として CS と UV との間のパケット伝送遅延を再現する．本研究では，シミュレーションにおいて実際の伝送遅延を十分に模擬する遅延データセットを用意した．

図 5-2 はインターネット区間の片道伝送遅延を図示する．これらはパレート分布を実際の伝送遅延の測定結果にフィッティングさせることで作成した遅延データセットである．先行研究[73]により，インターネット上の片道伝送遅延はパレート分布を用いることで十分に模擬できることが示された．パレート分布の確率密度関数の公式を以下の式(5.1)に示す．

$$f(x) = \frac{KA^K}{x^{K+1}} \quad (A > 0, K > 0, x > A) \tag{5.1}$$

2021 年 7 月に横須賀市と世界中の著名なクラウドサーバとの間の RTT (Round Trip Time) を測定した．RTT の値は 1 秒ごとに測定した．この測定を各サーバとの間でそれぞれ 6 時間の間継続した．それらの測定結果と式(5.1)に示すパレート分布の式を用いて，3 つの伝送遅延モデルを作成した．パレート分布の尺度パラメータ A と形状パラメータ K は以下の式(5.2)と(5.3)によってフィッティングした．ここで，d_i は測定結果の中の i 番目の RTT の半値を意味する．n は測定した RTT 値の総数を示す．mode は最頻値を意味する．

$$A = \mathrm{mode}\,(d_1, d_2, \ldots, d_n) \tag{5.2}$$

$$K = \frac{n}{\sum_{i=1}^{n} \log \frac{d_i}{A}} \tag{5.3}$$

測定結果にフィッティングされた式(5.1)は伝送遅延モデルの式として活用できる．式の左辺に 0 から *K*/*A* の間の一様乱数の値を代入すると，*x* の値が求められる．この値が片道伝送遅延のデータセットを形成する遅延の値である．伝送遅延のデータセットとして csv 形式で保存する際は，各値が整数となるように四捨五入した．

上記の方法によって作成した 3 つの伝送遅延モデルをそれぞれモデル A，B，C とする．伝送遅延モデル A は測定した中で最も伝送遅延が小さかった東京のサーバとの間の RTT 測定結果から作成された．モデル C は最も伝送遅延が大きかったフランクフルトのサーバとの間の結果から作成された．モデル B はモデル A と C の中間の大きさの伝送遅延を再現するものであり，サンフランシスコのサーバとの間の測定結果から作成された．これらの伝送遅延モデルを用いて，図 5-2 に示す 120 秒間の伝送遅延データセットを作成した．データセット内の各遅延値は 10 ms ごとに NE で再生される．表 5-1 に各伝送遅延モデルのパレート分布パラメータの値を示す．また，作成された伝送遅延データセットにおける遅延の平均値と最大値も示す．遅延の最小値はパレート分布の尺度パラメータ *A* と同じ値である．

無線アクセス区間は Wi-Fi か URLLC のどちらかを想定した．URLLC を想定する場合，その片道伝送遅延は 1 ms とした[74]．Wi-Fi の場合，伝送遅延のデータセットとして IEEE 802.11ac における 20 秒間の片道伝送遅延の測定結果を用いた．この規格は現在多くのオフィスで広く利用されているので，実用的評価という観点から Wi-Fi の規格として適した選択であると判断した．伝送遅延の測定は，Wi-Fi の通信において 300 Mbps のトラフィックをバックグラウンドの通信として付与した状況下で行った．図 5-3 は測定した伝送遅延とその最小値，最大値および平均値を図示する．この伝送遅延データセットもインターネット区間と同様に 10 ms ごとに NE で再生される．

シミュレーション時は，NE にはインターネット区間の 3 種類の伝送遅延のうち 1 つ，および無線アクセス区間の 2 種類の伝送遅延のうち 1 つが与えられる．それぞれのデータセットによる伝送遅延の合計値が CS と UV との間の伝送遅延となる．NE は CS または UV からのパケットを受信すると，2 つの伝送遅延データセットによって決められた時間が経過するまでパケットを保持した後に本来の送信先に送信する．言い換えれば，シミュレーションの条件として，NE は 6 種類の伝送遅延のうち 1 種類を再現すると言える．シミュレーション中には，CS と UV との間の上り通信と下り通信の双方に同じ伝送遅延を再現させた．同一条件のシミュレーションを繰り返す際，NE は前回の試行で再生し終えた所の続きから，伝送遅延の再現を再開する．

表 5-1　伝送遅延データセットのパラメータ

Name	Parameters of Pareto		Average	Max
	K	*A* (Min)		
Model A	2.5	3 ms	3.8 ms	52 ms
Model B	15	54 ms	57.1 ms	110 ms
Model C	30	120 ms	123.6 ms	169 ms

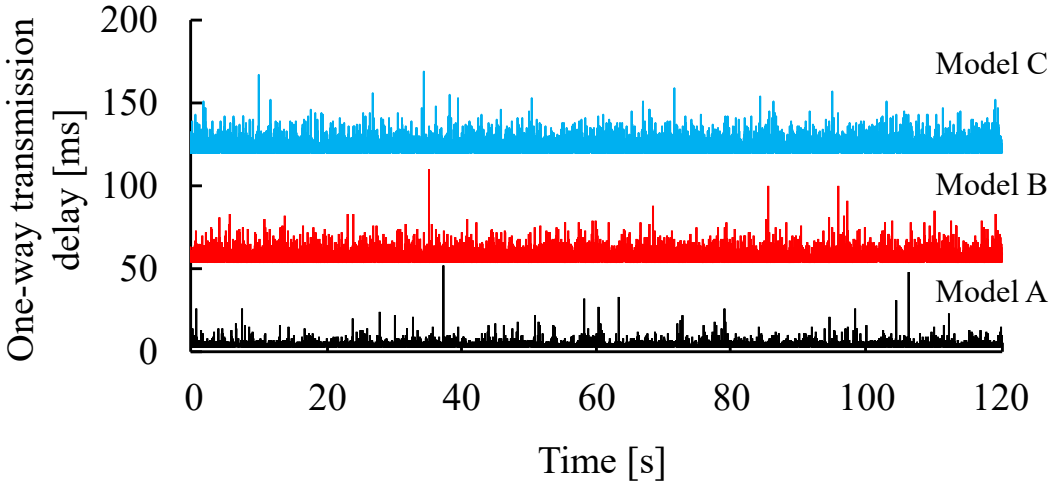


図 5-2　インターネット区間の片道パケット伝送遅延

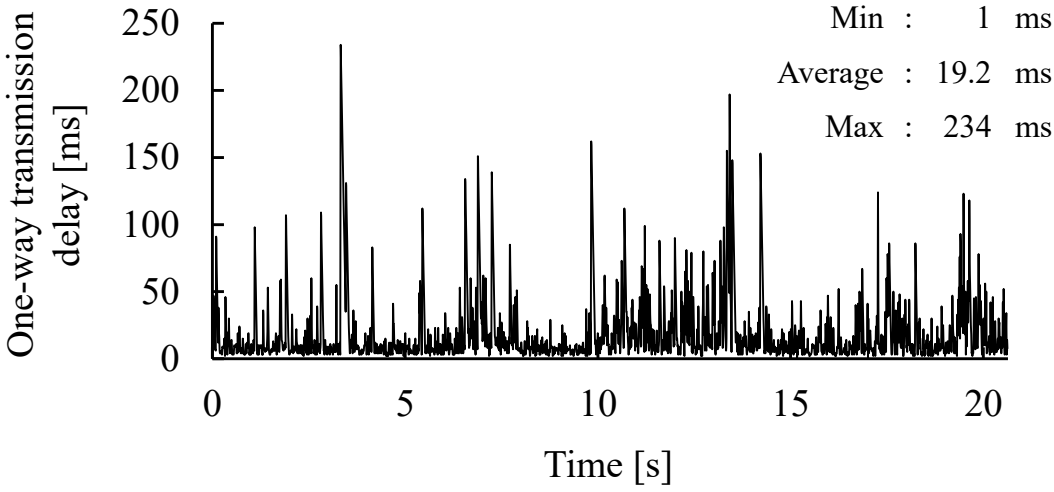


図 5-3　Wi-Fi 区間の片道パケット伝送遅延

B. パケットロス

NE では所定の条件によってパケットロスを発生させることができる. 実際のインターネットを介した通信では，パケットロスは様々な原因によって生じる．例として，通信設備におけるバッファオーバーフローが挙げられる．本検討では，提案システムにおいてロスするパケットの割合に対する UV 遠隔制御の基礎的特性を評価するために，以下のシンプルな方法によるランダムパケットロスとした.

CS または UV からのパケットを受信すると，NE は所定のパケットロスの確率に基づきそのパケットを破棄するか判断する．例えば，パケットロス率を 3%とした場合，NE はパケットを受信する度に 3%の確率でパケットを破棄する. 破棄されなかったパケットは前述の通り NE を通して送信される．パケットロス率は，シミュレーション開始前に NE に与えられる．パケットロスを発生させる場合，CS と UV との間の双方向の通信に同じパケットロス率を与える．後述の 5.4 節では，パケットロス率は当初 0%に設定して評価を実施した.

5.2.4 CS と UV のパラメータ

CS によって指定される UV の走行速度 v はシミュレーションごとに 0.5 m/s から 2.0 m/s の間の一定値とした．速度の範囲は実際に工場等で利用されている小型の自律搬送ロボットを参考にして設定した．また，CS による UV のステアリング角 θ の最大値は 0.7 rad とした.

UV が状態情報を送信する周期は，ディジタルツインが適用された提案システムでは 100 ms とした．ディジタルツインが適用されていない制御システムでは 10 ms とした.

第 4 章で解説した提案システムでは UV にジッタバッファが適用されている．ジッタバッファによる制御信号パケットの伝送遅延変動の吸収に関わるパラメータである D は以下のように設定した. d_m をシミュレーション時に NE が再現するパケット伝送遅延の最小値とする．各シミュレーションごとに $D = d_m + 200$ ms とした．これにより，提案システムでは殆どの制御信号の伝送遅延を一定とすることができる.

5.3　評価指標 *MLD*

物品の配送のようなアプリケーションにおいては，安全上の観点から指定経路を逸脱しないようにUVを制御できる能力が必須である．本論では，UVの経路追従走行の結果を評価するためにUVが走行した経路が目標とする経路から逸脱した長さに着目した．

本研究の遠隔制御システムでは，UV の走行経路は遠隔制御の開始位置(0, 0)から停止位置までに移動した座標(x, y)を 1 ms ごとにcsv形式で記録したデータとして出力される．UVの経路追従走行の結果を定量的に評価するため，目標経路を基準としたときの UV 走行経路の乖離の最大値（Maximum Lateral Deviation）を評価指標とした．以降はその略称である *MLD* を指標として呼称する．

5.2.2 項の図 5-1 に *MLD* の例を示す．*MLD* は 2 種類の UV の走行経路を比較することで求められる．まず，図中の黒線で示される目標経路と同一となる UV 走行経路を求めて比較の基準とする．この基準経路は，第 3 章の UV 遠隔制御シミュレータで伝送遅延がない，かつUVの走行速度が非常に遅い場合の走行結果を求めることで得られる．本研究では，走行速度を 0.1 m/s として基準経路を求めた．この時，図に示す目標経路と殆ど同じスラローム状の走行経路のデータが得られた．図中の赤線は伝送遅延が大きい場合の UV 走行結果の一例である．以下(1)から(4)に示す処理を順に行うことで，2 つの走行結果を比較した時の *MLD* の値を求める．

(1) シミュレーションによってUV走行経路データを得る．このデータ内の各座標値を(x_i, y_i)とする．基準経路データの各座標値は(p_j, q_j)とする．

(2) 図 5-4 に示すように，UV 走行経路データ内の i 番目の座標(x_i, y_i)における基準経路からの横方向乖離（LD_i）の長さを求める．この長さはUV走行経路上の座標(x_i, y_i)から基準経路に垂線を引いた時の長さに相当する．基準経路上のある座標(p_j, q_j)において座標(x_i, y_i)に対して隣接し，かつ走行方向に対する垂線を引けるものがある場合，その長さをLD_iの値とする．そうでない場合，3 点(x_i, y_i)，(p_j, q_j)，(p_{j+1}, q_{j+1})を結んだ 3 角

形の内角が全て π/2 rad 未満となる j を探索する．この時，点(x_i, y_i)を頂点とした 3 角形の高さがLD_i の値に相当する．

(3) (2)を UV 走行経路データの全ての座標において実施する．

(4) LD_i の最大値をその UV 走行経路データにおける MLD の値とする．

MLD の値が小さい程，CS による UV の遠隔経路追従制御が正確であったことを示す．遠隔制御に要求される制御精度はアプリケーションごとに異なる．本章では，提案システムの有効性を判断するための基準として，$MLD \leq 0.1$ m を制御精度の許容範囲とした．予備的な検証を通して，本検討の条件ではこの許容範囲を満たす場合に UV 走行経路が目標経路と殆ど同じとなることを確認した．

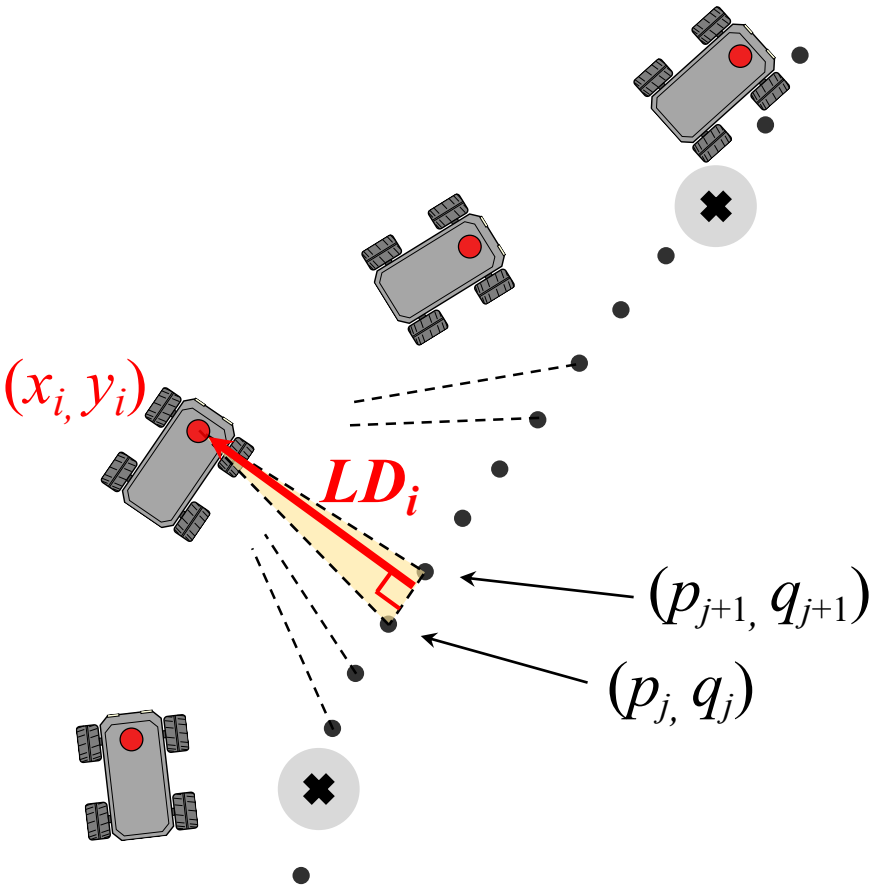


図 5-4　UV 走行経路データの各座標における *LD* の長さ

5.4　評価結果

条件ごとに UV の遠隔制御のシミュレーションを 100 回実施した．シミュレーションの結果から求められた *MLD* の値を以下に提示するグラフにまとめる．各グラフでは，点は 100 回のシミュレーション結果の中央値を示す．また，誤差範囲は第一四分位数から第三四分位数までのばらつきを示す．

図 5-5 に CS と UV との間の伝送遅延特性を変えた場合の *MLD* の変化を示す．パケットロス率は 0%とした．横軸は UV の走行速度を示す．縦軸は *MLD* の値を示す．黒，赤，青の破線はそれぞれ 5.2.1 項で述べた制御システム(i)，(ii)，(iii)を示す．また，各グラフの上部に伝送遅延を再現するために NE に与えた伝送遅延データセットの種類を示す．また，参考として図 5-6 にシミュレーション結果である UV 走行経路の一部を示す．図中の灰線は 5.2.2 項で示した目標経路を示す. 黒線はシステム(i)の場合の UV 走行経路である．この時, $MLD = 0.354$ m であった．青線はシステム(iii)の場合の UV 走行経路であり，$MLD = 0.0103$ m であった．これらは，伝送遅延モデル C と Wi-Fi を組み合わせた伝送遅延および $v = 1.0$ m/s という条件のシミュレーション結果の中央値に近い結果である．

伝送遅延特性が同じ場合の各制御システムによる *MLD* の変化を比較する. インターネット区間が遅延が小さい伝送遅延モデル A であった時，無線アクセス区間に URLLC を用いた場合はどのシステムでも走行速度 2.0 m/s で $MLD \leq 0.1$ m を達成できた．言い換えれば，どのシステムも正確な UV 走行制御が達成できたと評価できる．Wi-Fi を用いた場合，システム(i)は速度が 1.5 m/s 以上となると $MLD > 0.1$ m となり，正確な制御ができなかった．一方で，システム(ii)と(iii)はどちらも速度が 2.0 m/s でも正確な制御が達成できた．さらに，システム(iii)は(ii)と比べて *MLD* の中央値およびばらつきが小さいので，より正確かつ安定した走行制御を達成できたと言える．インターネット区間がモデル B であった時, URLLC を用いた場合システム(i)は速度 2.0 m/s では正確な制御ができなかったが，システム(ii)と(iii)は達成できた．Wi-Fi を用いた場合，システム(i)は速度が 1.5 m/s では正確な制御ができなかった．また，速度が 1.0 m/s の場合でも正確な制御が達成できないことが多かった．一方で，システム(ii)と(iii)は

正確な制御が達成できた．インターネット区間が遅延が大きいモデル C であった時，システム(i)は無線アクセス区間の種類に関わらず，速度が 0.5 m/s である場合のみ正確な制御を達成できた．システム(ii)と(iii)は Wi-Fi の場合でも正確な制御を達成できた．

システム(i)と他のシステムの結果を比較すると，ディジタルツインの適用は UV の遠隔経路追従制御において制御精度の改善に非常に有効であることがわかる．その有効性は，通信による伝送遅延が大きい程顕著であることが確認された．また，システム(ii)と(iii)の結果を比較すると，ジッタバッファの適用は UV の遠隔制御の精度の改善に有効であることが確認された．伝送遅延の組合せ(a)，(c)，(e)のシミュレーション結果はシステム(ii)と(iii)の間で差が殆どない．一方で伝送遅延(b)，(d)，(f)では，システム(iii)はより正確かつ安定した UV 遠隔制御を達成できた．ジッタバッファによる遅延変動の有効性は通信による伝送遅延の変動が大きい程顕著であることが確認できる．CS を用いた遠隔制御では，伝送遅延の平均値や変動が大きい通信状態となる可能性を考慮する必要がある．図 5-5 に示す結果から，提案システムは CS による UV の経路追従制御において有効であると考えられる．本検討では 5.2.2 項に示すスラローム状の経路を目標経路として設定した．この経路では伝送遅延が存在する状態では UV 経路追従制御が非常に困難となる．この経路で良好な UV 遠隔制御を達成できることを確認できたので，提案システムは他の形状の経路上での UV 遠隔制御においても有効であると考えらえる．

次に，追加検討として提案システムにおいてパケットロスが制御結果に及ぼす影響を評価する．図 5-7 に通信ネットワーク上でランダムパケットロスが生じた場合の *MLD* の変化を示す．横軸はパケットロス率を示す．破線の色は走行速度の違いを示す．この評価の条件として，伝送遅延はインターネット区間をモデル C，無線アクセス区間を Wi-Fi とした．これは全 6 種類の伝送遅延の組合せの中でもっとも伝送遅延の変動が大きいものである．

図 5-7 から，提案システムは通信ネットワークによるパケットロス率が 30%の場合，走行速度が 2.0 m/s であっても $MLD \leq 0.1$ m を達成し，正確な制御が可能であることを示した．パケットロス率をさらに増加させると *MLD* の値とそのばらつきが増加し，正確な制御を達成で

きる速度の上限が低下することが確認できる．また，比較的低速で走行する場合は，パケットロス率が大きい場合でも良好な走行制御を達成できることが確認できる．本検討では，提案システムは通信によるパケットロス率が 30%以下である場合に正確な UV 経路追従走行を達成できると言える．これは 5.2.2 項で示すスラローム形状の経路における結果である．他の形状の経路の場合，許容パケットロス率はより大きくなると考えられる．

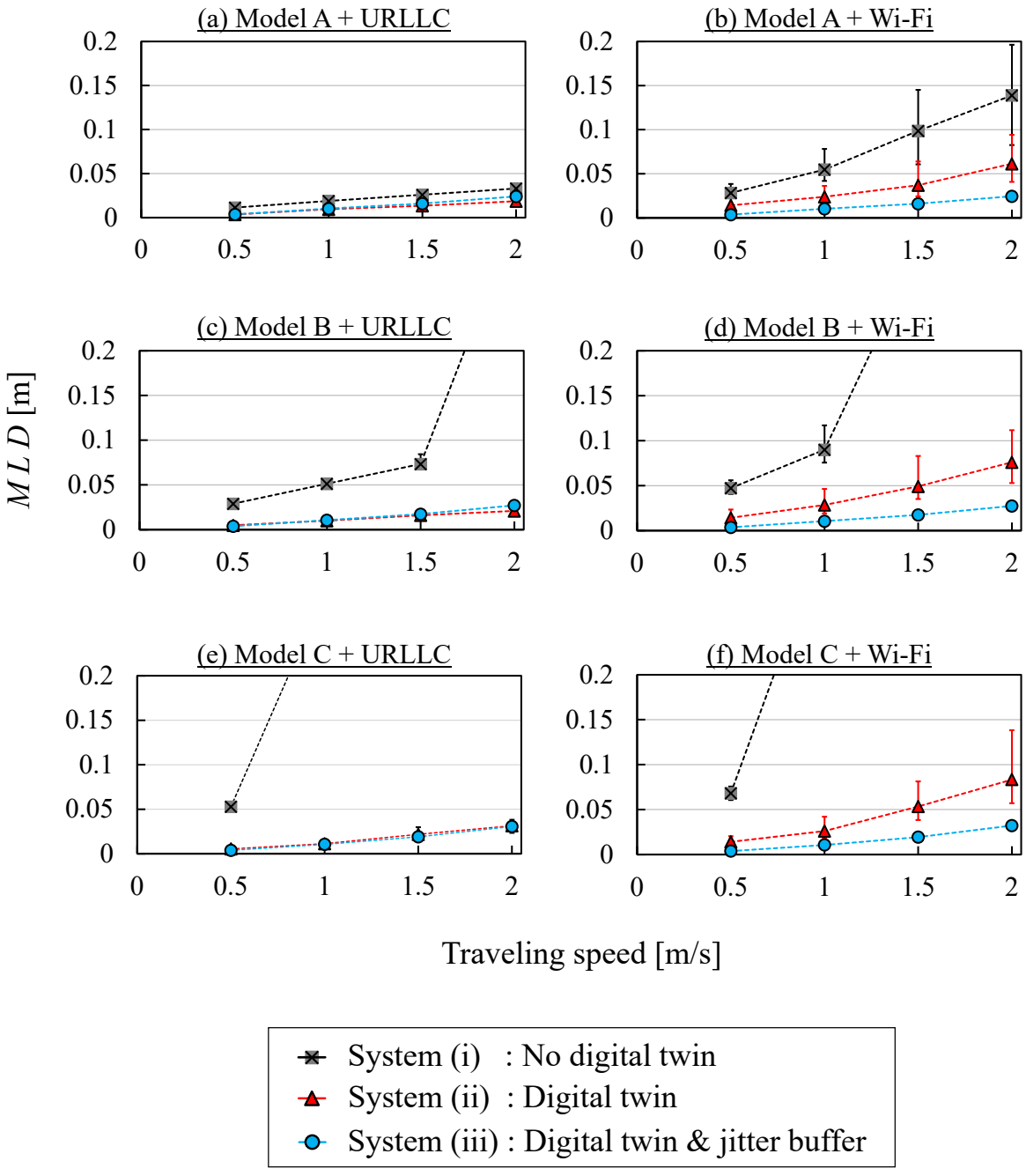
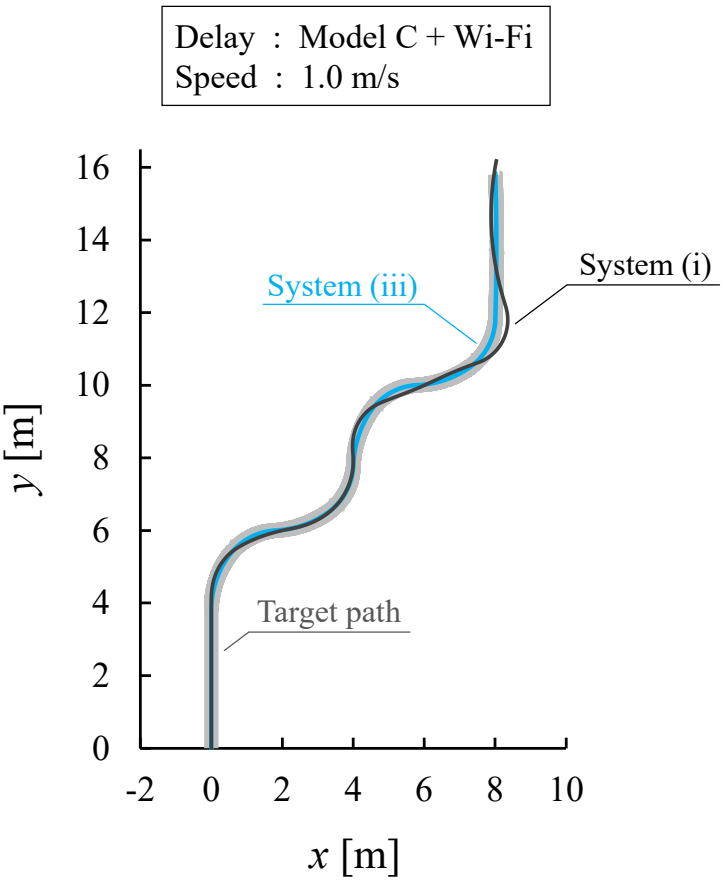
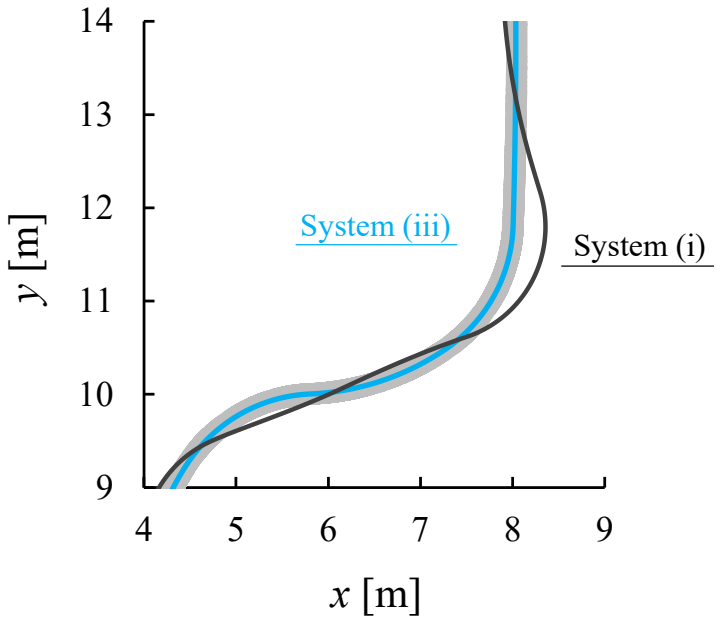


図 5-5　各伝送遅延特性における *MLD* の変化



(a)　UV 走行経路の全体



(b)　UV 走行経路の一部

図 5-6　伝送遅延モデル C，Wi-Fi および走行速度 1.0 m/s の場合の UV 走行経路

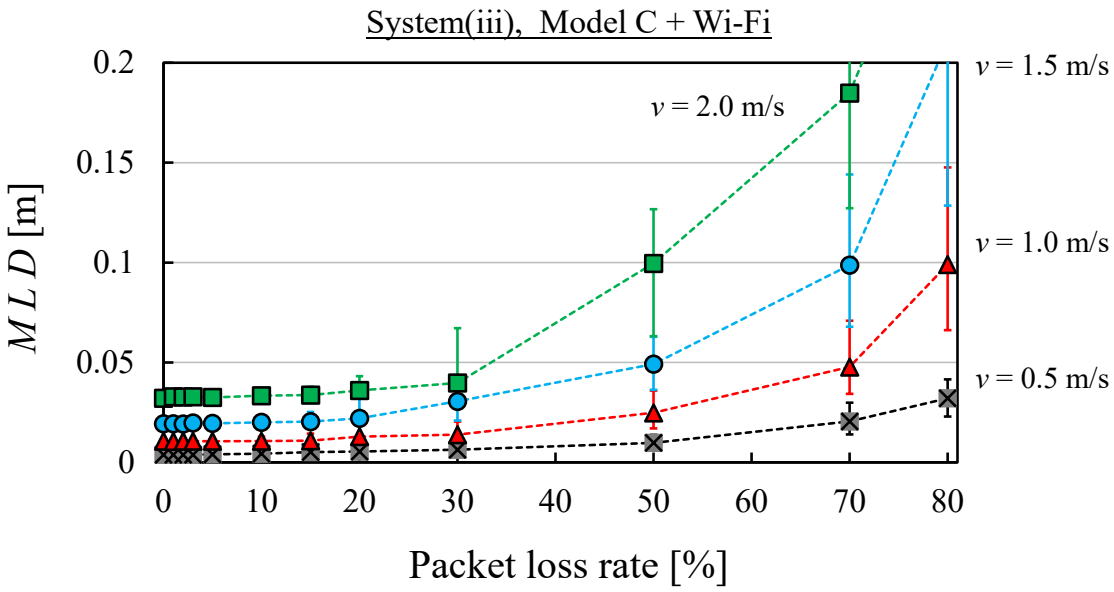


図 5-7　提案システムの UV 遠隔制御に対するパケットロス率の影響

5.5　結言

走行車を遠隔制御する時，通信ネットワークによる伝送遅延とその変動が制御の精度や安定性を劣化させる．伝送遅延がない場合に正確な制御を実現できる制御機能であっても，伝送遅延による影響は避けられない．小型 UV の遠隔経路追従制御における伝送遅延の問題を解決するため，CS による UV の経路追従制御において，ディジタルツインを活用してその制御精度を向上させるシステムを考案した．本検討では，その提案システムの有効性を定量的に評価した．

シミュレーション評価を通して，提案システムについて以下を確認した．

(1) UV の走行状態を模擬するディジタルツインを CS に適用することで，通信による伝送遅延が大きい場合であっても正確な UV 経路追従制御が可能となる．

(2) (1)に加えて，制御信号の伝送遅延変動を吸収するジッタバッファを UV に適用することで，通信による伝送遅延ジッタが大きい場合であっても正確な UV 遠隔制御が可能となる．

上記の通り，提案システムによる CS の UV 経路追従制御が有効であることが確認された．しかしディジタルツインを活用した UV の遠隔制御について，検討すべき要素は多いと考えられる．本検討では，シミュレーションでは CS と UV は独立したプログラムとして動作させたため，ディジタルツインで完全な予測を行うことはできなかった．しかし，プログラムの仕組みから，CS に適用されたディジタルツインは制御対象である UV が制御信号を受信した後の走行状態をある程度正確に模擬することできる状態であったと考えられる．現実的に考えると，ディジタルツイン上の予測には実際の UV の状態に対する誤差が存在するはずである．UV の遠隔制御においてディジタルツインの誤差がどの程度影響を及ぼすかを検討することで，より実用的な観点から提案システムを評価することが可能であると考えられる．また，本検討ではジッタバッファの遅延変動吸収に関するパラメータ D を固定値としてシミュレーション開始前に設定した．実際の通信による伝送遅延を想定すると，パラメータ D は遠

隔制御時の通信ネットワークの状態に適した値とするべきである．より現実的な制御システムとするためには，提案システムのジッタバッファに関する機能を改善する必要があると考えられる．

第 6 章　クラウドサーバの遠隔走行制御におけるディジタルツインの予測誤差の影響

6.1　緒言

ディジタルツインとは現実の物体および環境のモデルを用いてコンピュータ上で空間を高度に再現する技術またはシステムの概念である．ディジタルツインを活用すると，CS は UV の走行状態を予測することができる．高度なディジタルツインによって車体内部の機械的特性や路面等周辺環境の状態を再現することで，非常に正確な予測が可能となると考えられる．本研究の提案システムでは CS によって制御信号が計算・送信されるため，走行状態の予測に加えて実際の車体の位置・向きの情報に基づいた再予測を実行できる．

ディジタルツインによる予測の正確性は提案システムにおける UV の走行制御の精度および安定性に影響すると考えらえる．ディジタルツインが実際の UV とその走行環境を完全に模擬できている場合，UV の走行状態を非常に正確に予測することが可能である．しかし，一般的に考えて現実の現象をコンピュータで完全に予測することは困難である．実際には，UV は車体内部の機械的制約や周辺環境から様々な影響を受けながら走行する．実際の UV とディジタルツイン上の UV モデルが同じ制御信号を受信する場合でも，両者の走行速度やステアリング角等の挙動は完全には一致しない．これによってディジタルツイン上の予測結果に誤差が生じる．例えば，CS がある制御信号を送った時，ディジタルツイン上の UV モデルではステアリング角が 0.3 rad と模擬したとする．これに対して実際の UV のステアリング角が 0.25 rad であった場合，その角度の差に応じて，ディジタルツイン上で予測した UV モデルの位置・向きが実際の UV と異なるものになる．

ディジタルツインの予測の正確性は，現実の物体をモデリングする精度に影響されると考えられる．UV の場合，モデリングの精度は走行制御に関する過去のデータを統計的に分析・活用することで改善できる可能性がある．しかし，リアルタイムな走行制御において発生する誤差を完全に解消することは困難である．ディジタルツインを活用した UV の遠隔制御について検討を進める上では，モデリングの誤差またはそれによって生じるディジタルツイン

の予測結果の誤差が走行制御特性に及ぼす影響を定量的に評価することが必要である．

本章では，UV の走行状態に関するディジタルツインの予測誤差が CS による UV の遠隔経路追従制御に及ぼす影響について検討した結果について解説する．検討内容はシミュレーションと実際の小型車両を用いた実験の 2 種類である．シミュレーション評価では，第 5 章に続きディジタルツインを適用された UV 遠隔制御システムのシミュレータを用いた．この評価では，制御信号の値を UV が実行する際に所定の補正比率を掛けるように UV プログラムを調整した．これによって，ディジタルツインによる走行状態の予測に定量的な誤差が発生する状態とした．次に，提案システムによる制御特性を実際に評価するため，UV として小型ラジコン車を用いることで，ディジタルツインを活用した遠隔走行制御システムを試験的に構成した．これによって，実際の走行車の制御においてディジタルツインの誤差が及ぼす影響を評価した．

6.2 節では，シミュレーション評価のために行った UV プログラムの調整および評価結果について述べる．6.3 節では，実際の小型ラジコン車を用いた実験について述べる．6.4 節では，それぞれの検討の結果をまとめる．なお，遠隔制御の結果である UV 走行経路を評価するため，第 5 章と同様に指標 *MLD* を用いた．

6.2　シミュレーション評価

本検討では，提案システムにおいてディジタルツイン上で予測誤差が生じる状態を想定し，正確な UV 経路追従走行を達成するための条件を定量的に評価する．シミュレーションは第 4 章で解説したシミュレータを用いて実施した．各シミュレーションでは，制御対象である UV と CS 内のディジタルツイン上の UV モデルとの間で，走行速度またはステアリング角の状態に差がある状態を模擬させた．

6.2.1 項では UV プログラムに追加した機能について述べる．6.2.2 項ではシミュレーション条件について述べる．ここで述べる条件は第 5 章のものと共通した項目が多い．6.2.3 項ではシミュレーション評価の結果を解説する．

6.2.1　UV プログラムの調整

第 3 章で解説した UV プログラムを，受信した制御信号 v, θ のどちらかの値に対して補正比率を掛けるように調整した．図 6-1 にその概要図を示す．どちらの値に比率を掛けるかはシミュレーションごとに変更した．v または θ に掛ける比率を本章では *MER* (Modeling Error ratio) と呼称する．CS 内のディジタルツインでは，制御信号の値に従って走行状態(x, y, φ)を予測する．そのため，UV 内で制御信号の値に掛ける *MER* の比率は，UV とディジタルツインとの間の走行パラメータの比率に等しい．これによって，*MER* の値に応じた定量的な予測誤差を発生させることができる．以下の式(6.1)と(6.2)に，UV プログラムを調整したことによる UV とディジタルツイン上の UV モデルとの間の v, θ の関係を示す．v_{uv}，θ_{uv} はそれぞれ制御信号を実行した UV の走行速度とステアリング角の値を示す．v_{dt}，θ_{dt} はそれぞれディジタルツイン上の UV モデルの値を示す．*MER* が 1 より大きい場合，UV はディジタルツイン上の UV モデルよりも速い速度, または大きなステアリング角で走行することを意味する. *MER* が 1 より小さい場合はその逆である．

$$v_{uv} = v_{dt} \times MER_v \tag{6.1}$$

$$\theta_{uv} = \theta_{dt} \times MER_\theta \tag{6.2}$$

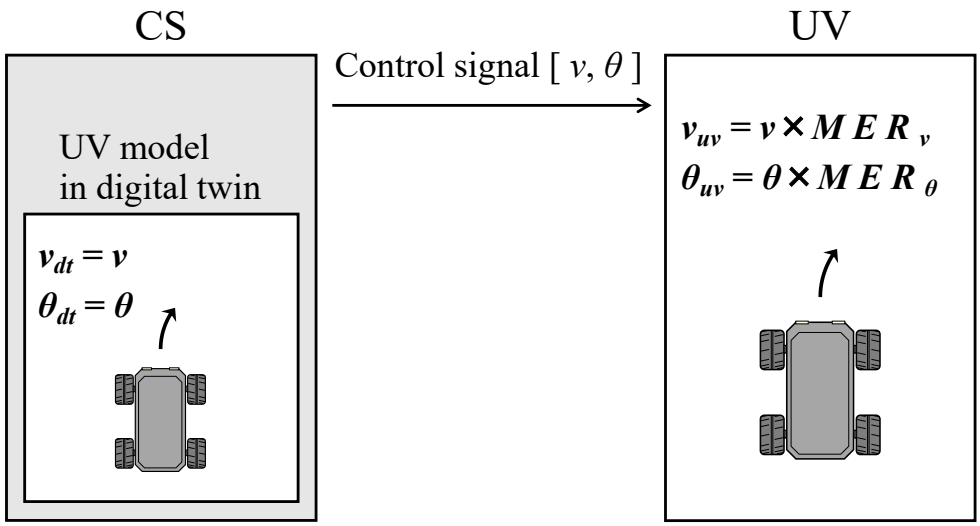


図 6-1　制御信号の値に対する補正比率 *MER* の適用

6.2.2　シミュレーション条件

MER の大きさはシミュレーションごとに 0.8 から 1.2 の範囲から一定の値を与えた．UV の経路追従走行における目標経路は第 5 章と同様のものとした．同じ目標経路を利用するので，走行結果の正確性を評価する基準も同様に $MLD \leq 0.1$ m とした．NE によって再現する伝送遅延は，第 5 章における伝送遅延モデル C と Wi-Fi の組合せとした．パケットロスの発生は，APNIC (Asia Pacific Network Information Centre) が公開した統計[75]を参照して，ランダムパケットロス率を 3%とした．UV の走行速度は第 5 章と同様に 0.5 m/s から 2.0 m/s の間で変化させた．UV による状態情報の送信周期は本検討では 100 ms から 1,000 ms の間で変化させた．ディジタルツインの予測結果の誤差は状態情報を用いて修正される．そのため，UV とディジタルツインとの間の誤差について考慮する場合，この送信周期の長さが UV 走行制御の正確性に大きく影響する可能性がある．ジッタバッファのパラメータである D は第 5 章と同様に再現される遅延の最小値より 200 ms 大きな値とした．本検討では，上記の伝送遅延データセットの組合せに対して，$D = 324$ ms となった．

6.2.3　評価結果

v または θ に誤差がある場合の CS による UV 遠隔経路追従制御について評価するため，条件ごとに UV の遠隔制御のシミュレーションを 100 回実施した．状態情報の送信周期は 100 ms とした．シミュレーション結果は以下に示す各グラフにまとめる．グラフの各点および誤差範囲は第 5 章と同様にシミュレーション結果の中央値および第一四分位数から第三四分位数までのばらつきを示す．

図 6-2 と図 6-3 にそれぞれMER_v またはMER_θ を変化させた場合の *MLD* の結果を図示する．図 6-2 の場合は$MER_\theta = 1$ とした．図 6-3 の場合は$MER_v = 1$ とした．それぞれの横軸は *MER* の大きさを示す．$MER = 1$ の場合は，v または θ について UV と CS 内のディジタルツイン上の UV モデルの挙動が一致していたことを意味する．縦軸は *MLD* を示す．各線の色の違いはそれぞれ制御信号によって伝送される v の値を示す．

図 6-2 から，MER_v が 1 から増減するにつれて *MLD* の値は増加した．言い換えれば，UV の走行速度についてのディジタルツインの誤差が大きい程 UV の制御精度が劣化することが確認できた．*MLD* の増加の度合いは走行速度が速い程大きかった．走行速度が 2.0 m/s であった場合，$0.9 \leq MER_v \leq 1.1$ の時に $MLD \leq 0.1$ m を達成できた．図 6-3 から，走行速度の場合と同様に，UV のステアリング角についてのディジタルツインの誤差が大きい程 UV の制御精度が劣化することが確認できた．走行速度が 2.0 m/s であった場合，$0.9 \leq MER_\theta \leq 1.1$ の時に $MLD \leq 0.1$ m を達成できた．走行速度の場合と比較すると，$MER_\theta > 1$ における *MLD* の増加は同程度であった．一方で，$MER_\theta < 1$ においては *MLD* の増加はより大きかった．$MER_\theta < 1$ は，UV がディジタルツインの UV モデルの状態よりも小さいステアリング角で走行したことを意味する．この状態では，UV は走行経路上で曲がる際に目標経路の外側に逸れて走るため，他の場合と比べて目標経路上から UV が大きく逸脱する危険性が高いと考えられる．そのため，ディジタルツインを活用した UV の遠隔走行制御では，ステアリング角の挙動について UV モデルの精度を高める必要があることが確認できたと言える．

図 6-2 と図 6-3 に示すシミュレーション評価において，走行速度 2.0 m/s 以下で UV を制御する場合ディジタルツインの誤差の許容範囲は $0.9 \leq MER \leq 1.1$ であることがわかる．この許容範囲は，UV からの状態情報の送信周期を 100 ms とした時の結果である．状態情報はディジタルツインの予測結果を修正するために用いられる．実際の UV や走行環境の状態に対してディジタルツイン上で再現したモデルに大きな誤差が存在する場合，その送信周期は遠隔走行制御の精度に大きく影響すると予想される．

図 6-4 に状態情報の送信周期を増加させた場合の *MLD* の変化を図示する．このシミュレーション評価ではディジタルツインの誤差として $MER_\theta = 0.9$ とした．横軸は状態情報の送信周期を示す．送信周期が長い程 *MLD* は増加した．その増加の度合いは走行速度が速い程大きい．走行速度が 2.0 m/s であった場合，送信周期が 100 ms の時のみ $MLD \leq 0.1$ m を達成できた．一方で，走行速度 0.5 m/s の場合は，送信周期を 1,000 ms まで増加させても $MLD \leq 0.1$ m を達成できた．提案システムにおいては，ディジタルツインの誤差が比較的大きい状態で遠

隔制御を実施する場合，状態情報の送信周期と UV の走行速度との間にはトレードオフの関係が成り立つことが確認できる．ただし，比較的遅い走行速度でよいアプリケーションであれば，制御端末の送信周期が長くても十分に良好な遠隔走行制御が実現できる．

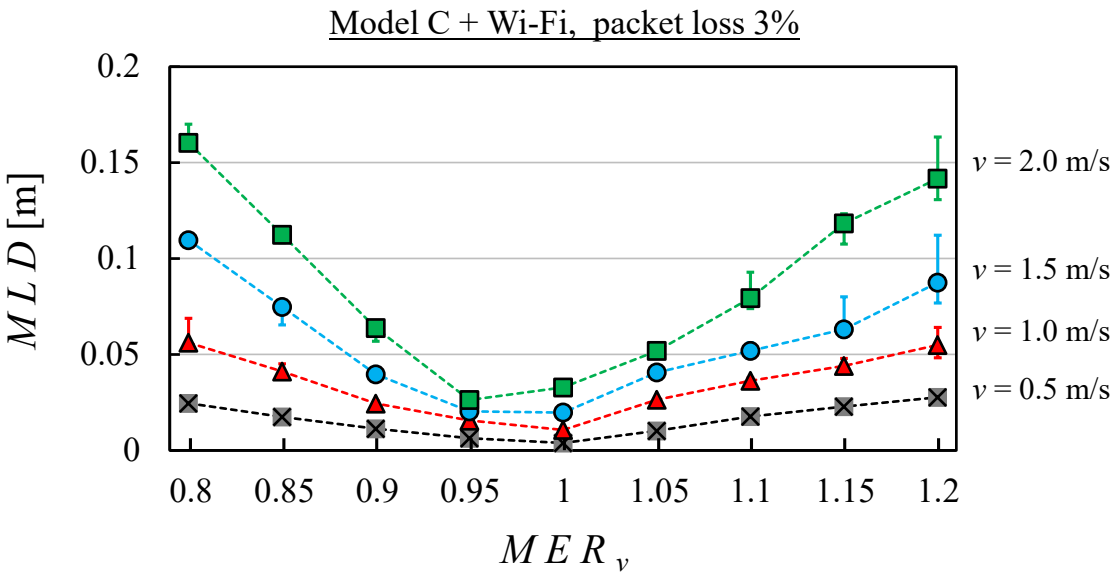


図 6-2　MER_v による *MLD* の変化

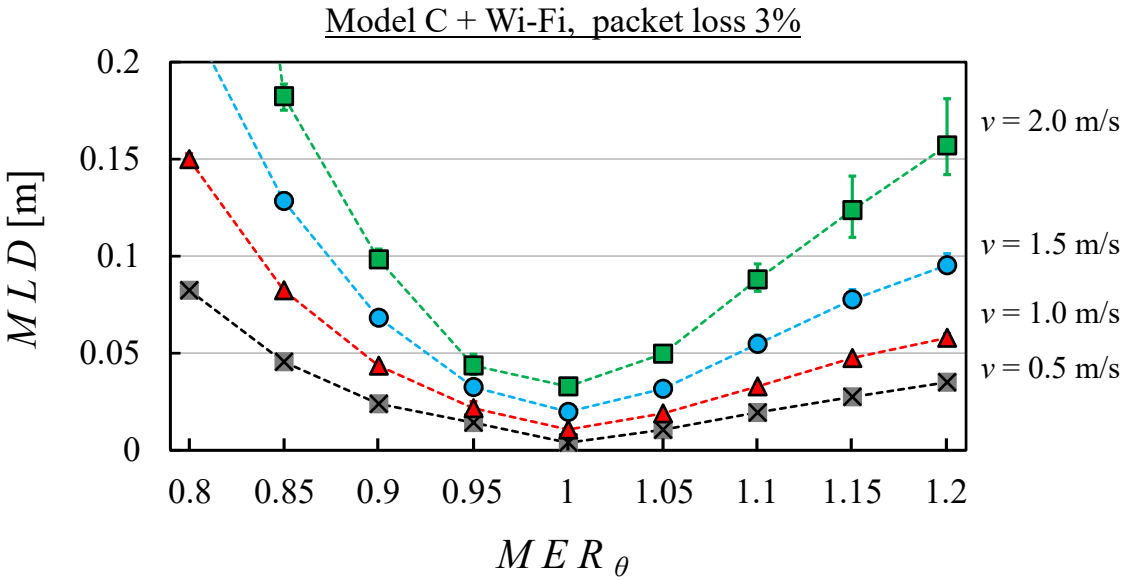


図 6-3　MER_θ による *MLD* の変化

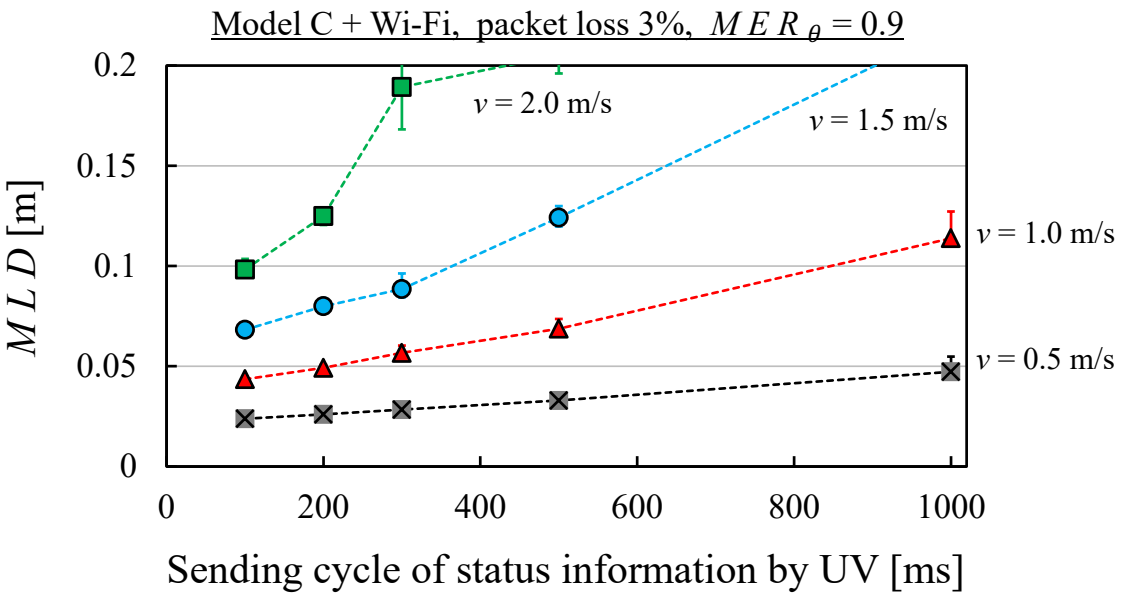


図 6-4　状態情報の送信周期による *MLD* の変化

6.3　小型ラジコン車を用いた実験

6.2 節では提案システムにおいて正確な UV 経路追従走行を達成するために必要な条件をシミュレーションによって評価した．これは仮想的な UV を用いた定量的評価である．実際の走行車では，様々な物理的影響によってシミュレーションとは異なる遠隔制御特性となる可能性がある．提案システムの制御特性を実際に確認するためには，実際の走行車を用いた実験が必要である．本検討では実際の小型の 4 輪ラジコン車を用いた試験的な制御システムを構成した．実験を通して提案システムの有効性を実際に確認するとともに，ディジタルツインの誤差が走行制御特性に及ぼす影響について実践的な評価結果を得た．この実験では，ラジコン車とディジタルツイン上の走行車のモデルとの間の誤差として，実際の走行速度とディジタルツイン上の走行速度との誤差に着目した．

6.3.1 項では実験用システムの構成を解説する．6.3.2 項では実験の条件について述べる．6.3.3 項では実験結果を解説する．

6.3.1　システム構成

図 6-5 に実験のために構成したシステムを図示する．システムは PC，Wi-Fi アクセスポイント，カメラ，および小型ラジコン車で構成される．図 6-6 に実験用 UV とした用いた小型ラジコン車の外観を示す．本実験では，後述の通り NE による伝送遅延を一定としたため，ジッタバッファ機能の実装は省略した．また，信号の送受信は UDP パケットを用いて実行させた．

PC 内には Java で構築した 3 つのプログラムを実装した．CS プログラムは第 4 章で解説したシミュレータと同様に制御器とディジタルツインの機能を含む．これらの具体的な処理も第 4 章で解説したものと同様である．ディジタルツインには，実験における制御対象であるラジコン車を模したモデルを適用した．走行車の状態情報は，PC に接続されたカメラを用いて特定される．図中で Position detector と記載される位置特定プログラムはカメラの映像からラジコン車の位置(x, y)と向き φ を特定する．ラジコン車の現在の状態情報$(x, y, \varphi, time)$は一定周期で CS プログラムに送信される．この時の各状態情報は，遠隔制御中はディジタルツイン

の予測結果の修正に用いられる．また，各状態情報を時系列的に記録して実験中のラジコン車の走行経路を示すデータとして保存させることで，UV の遠隔経路追従走行の特性を評価した．制御信号と状態情報のパケットには NE プログラムによって伝送遅延が付与される．付与される伝送遅延は後述の実験条件で示す．

Wi-Fi アクセスポイントとして，Aterm WG2600HP (NEC 製)を使用した．制御信号は IEEE802.11ac でラジコン車に送信される．ラジコン車として TAM58547 (タミヤ製)を使用した．このラジコン車は全幅約 28 cm，全長約 39 cm，軸距約 20 cm の小型車両である．この車両に制御信号を受信するためにタブレット PC を搭載した．受信された信号はタブレット PC を介して走行速度のためのモータと前輪のステアリング角のためのサーボモータを制御する Arduino ユニットに伝送される．

CS プログラムは WP によって指定される経路を走行するようにラジコン車を遠隔制御する．構成したシステムの制御特性を確認するため，NE による伝送遅延および状態情報の送信周期を変化させた時のラジコン車の走行経路を評価した．経路追従走行の目標経路として，後述する実験条件と同様の経路を用いた．評価指標として *MLD* を用いた．図 6-7 に NE による片道伝送遅延を変化させた場合の結果を示す．この時の伝送遅延は固定値とした．また，図 6-8 に状態情報の送信周期を変化させた場合の結果を示す．各グラフの赤線は本検討の実験用システムの結果である．黒線は実験用システムでディジタルツインを活用せず，状態情報を用いた直接的フィードバック制御を実施させた場合の結果である．破線は $MLD = 0.5$ m を示す．これは本システムにおいて経路追従制御を達成できたと判断する指標である．図 6-7 より，ディジタルツインを活用しなかった場合は伝送遅延が 100 ms で $MLD > 0.5$ m となった．200 ms 以上では，走行途中に目標経路から大きく逸脱して経路の終点に到達できなくなったため，経路追従走行が不可能な状態と判断して *MLD* を計算しなかった．ディジタルツインを活用した場合は, 伝送遅延を 500 ms まで増加させても経路追従走行を達成できた. また，図 6-8 よりディジタルツインを活用した場合は状態情報の送信周期を 500 ms まで増加させても経路追従走行を達成できた．以上から，本検討の実験用システムにおいてディジタルツイ

ンを活用したラジコン車の走行制御が十分に動作することを確認した.

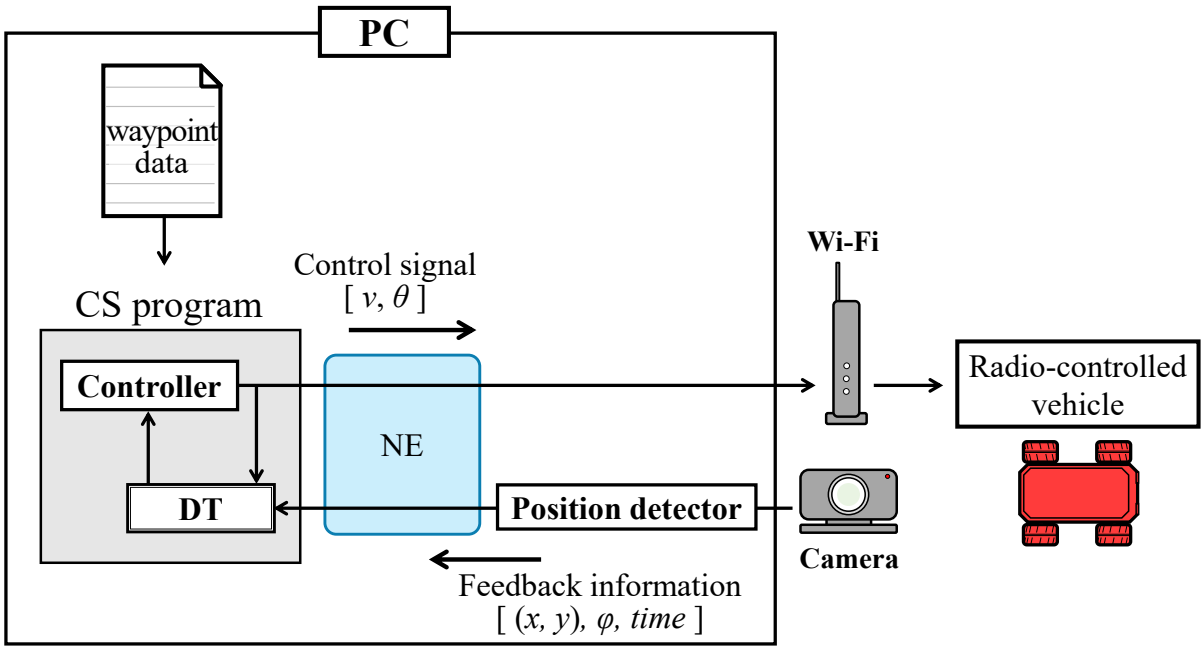


図 6-5　実験用 UV 遠隔制御システムの構成

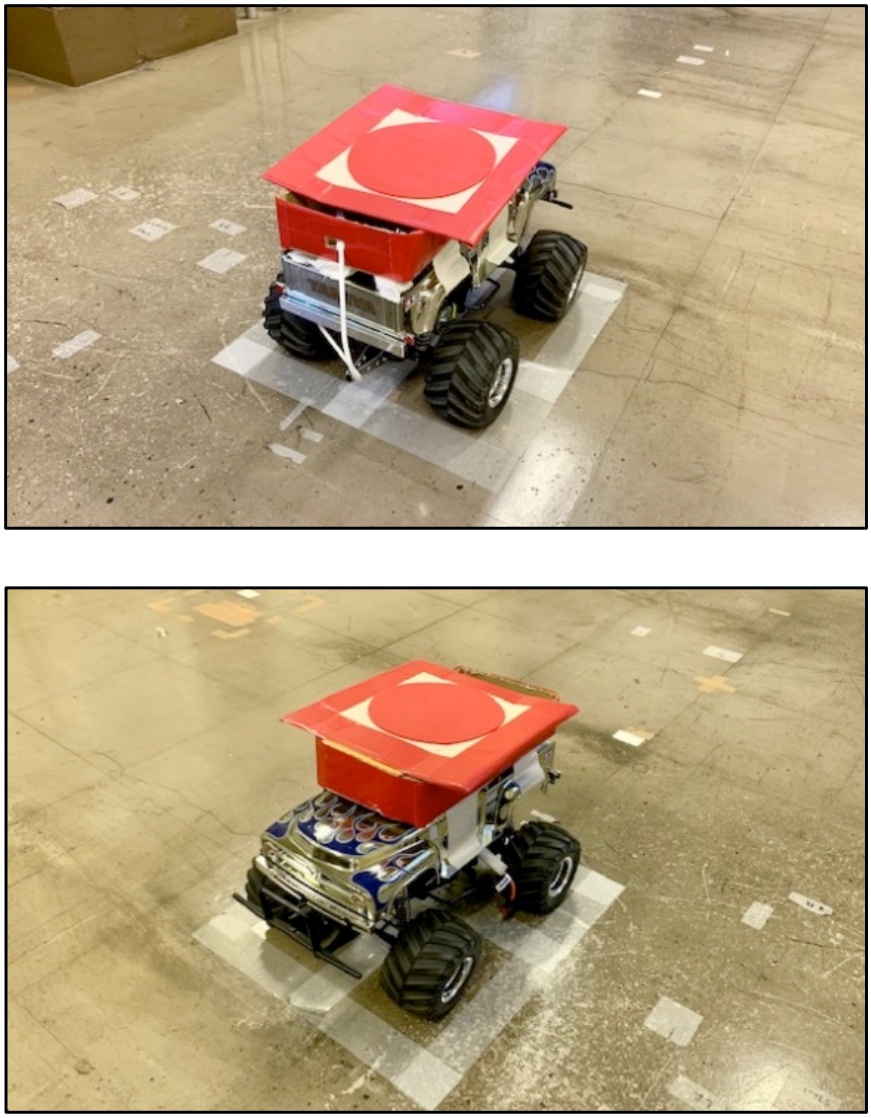


図 6-6　実験に用いた小型ラジコン車

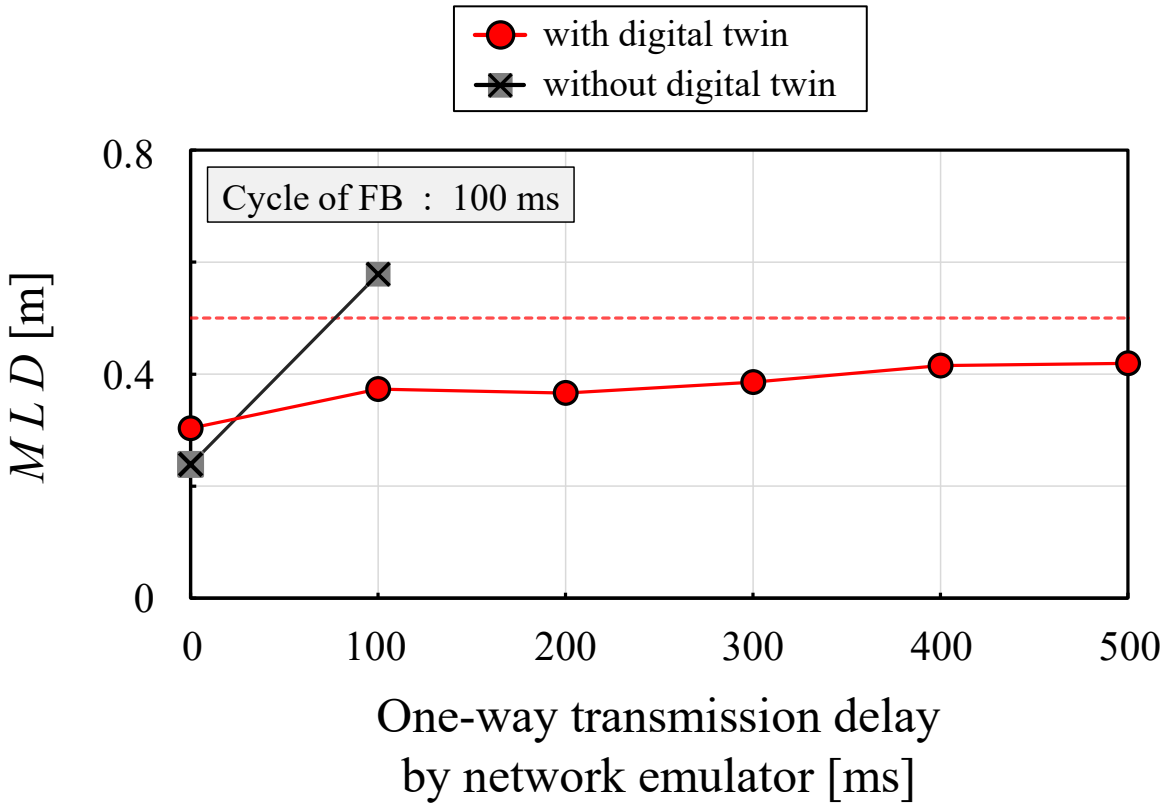


図 6-7　伝送遅延を変化させた場合の遠隔走行制御結果

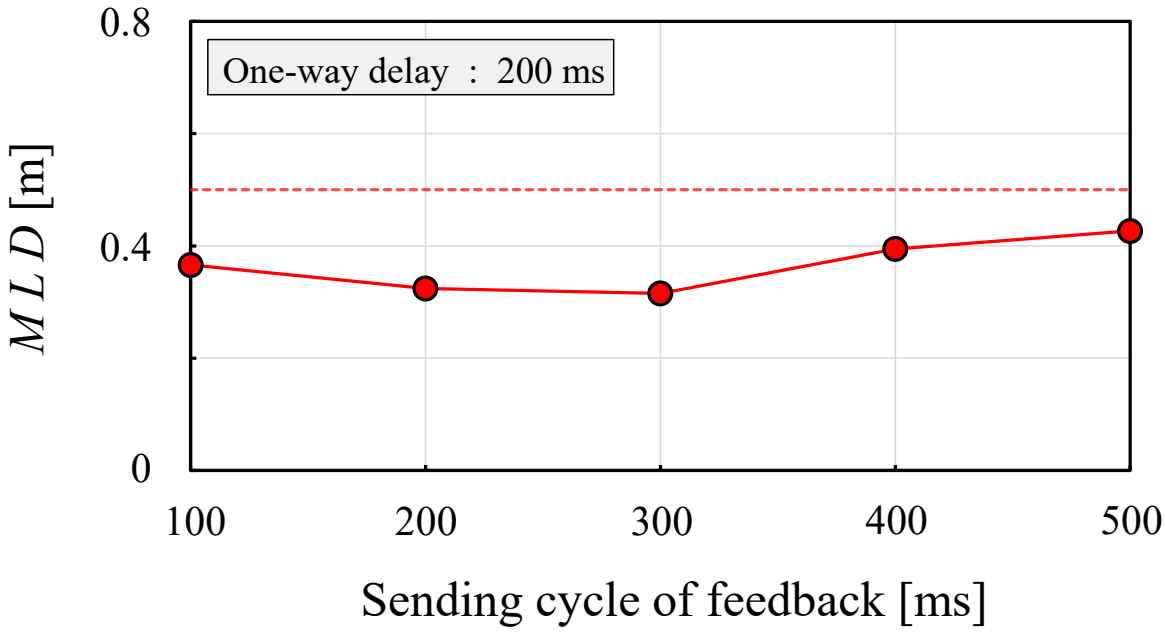


図 6-8　状態情報の送信周期を変化させた場合の遠隔走行制御結果

6.3.2　実験条件

図 6-9 に経路追従走行の目標経路を図示する．WP はラジコン車は始点(-1, 2)から走行し，8 の字を描く経路で 1 周して同じ座標へ戻るように配置される．参考として，図 6-10 に実際の実験環境における WP の配置を図示する．図 6-9 内の灰線はラジコン車の追従および *MLD* 計算のための目標経路を示す．この経路は第 5 章と同様にシミュレータによって求められた，CS による理想的な走行経路である．赤線はラジコン車の走行経路の例である．ラジコン車が目標経路に十分に追従できたか判断する指標として，$MLD \leq 0.5$ m とした．

NE は上り通信および下り通信の双方で 200 ms の片道伝送遅延を再現させた．また，ラジコン車の状態情報は 100 ms の周期で送信させた．

CS プログラムが送信する制御信号の値 v, θ はラジコン車の特性に合わせて調整した．v は 0.55 m/s，$|\theta|$の最大値は 0.52 rad とした．ラジコン車の実際の走行速度は制御信号の値と完全に等しいとは限らない．ディジタルツインに対する実際のラジコン車の走行速度の比を第 5 章と同様にMER_v で表すとする．予備実験により，ディジタルツイン上の走行車モデルが制御信号 v の値に従って走行する場合，本実験のラジコン車においては $0.91 \leq MER_v \leq 1.09$ となることを確認した．ここで，ラジコン車の走行速度はその走行経路から算出した．MER_v がより大きい値である場合にラジコン車の遠隔制御へ与える影響を確認するため，実験ごとにディジタルツイン側の走行速度を変化させた．これによって，MER_v の範囲を $0.56 \leq MER_v \leq 1.55$ まで拡大した．

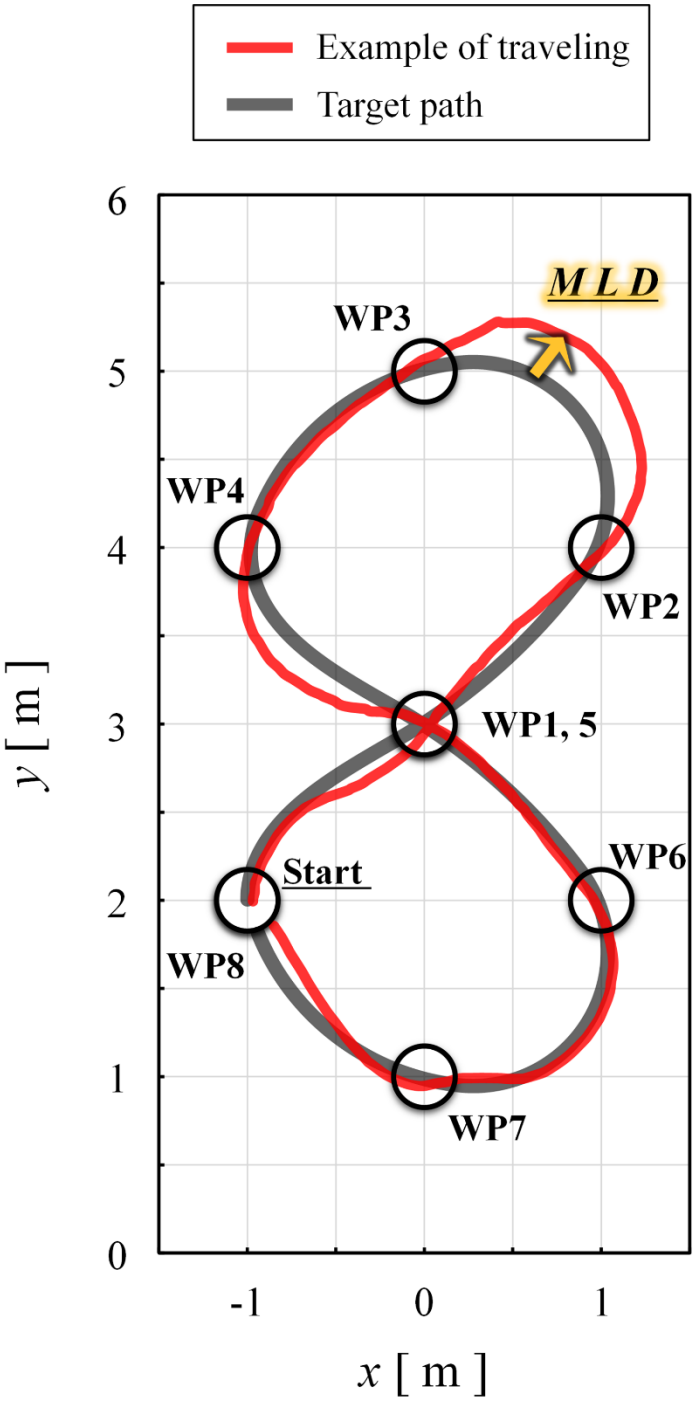

Example of traveling
Target path
MLD
WP3
WP4
WP2
WP1, 5
Start
WP6
WP8
WP7
6
5
4
3
2
1
0
-1
0
1
y [m]
x [m]

図 6-9　目標経路

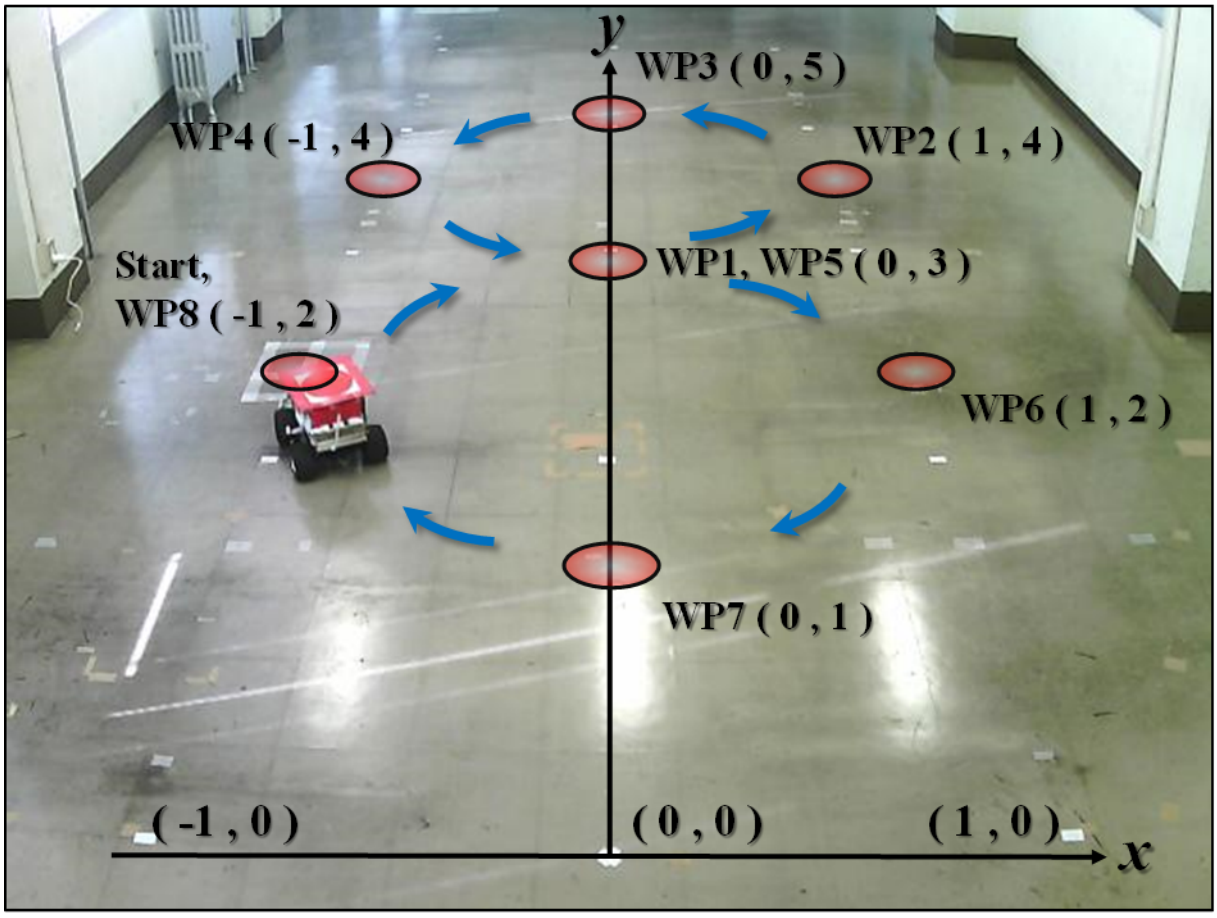


図 6-10　実験環境と WP 配置

6.3.3　実験結果

図 6-11 にMER_vの大きさに対する *MLD* の変化を図示する．横軸と縦軸はそれぞれMER_vと各実験のラジコン車の走行経路から求めた *MLD* を示す．赤点は各実験結果を示す．実験ごとにラジコン車の走行速度が微妙に変化したため，MER_vの値も実験ごとに異なる．黒い破線は *MLD* = 0.239 m を示す．これは本試験システムによる最良の経路追従走行結果で，NE による伝送遅延が無い状態で達成された．赤い破線は良好な経路追従走行が達成できたと判断する基準である *MLD* = 0.5 m を示す．$0.6 \leq MER_v \leq 1.1$ の間では，*MLD* ≤ 0.5 m を達成できた．この範囲は本実験におけるMER_vの許容範囲と言える．MER_vが 1.1 より大きくなるにつれて，*MLD* > 0.5 m となる結果が徐々に増加した．MER_vが 0.6 より小さい場合には，*MLD* > 0.5 m となる結果が 2 回生じた．そのうち 1 回は *MLD* = 1.58 m と目標経路からの乖離が非常に大きくなった．

これらの結果から，8 の字状の経路上を走行する状況では，提案システムにおける実際の小型走行車の速度がディジタルツイン上の走行車のモデルよりも遅い場合でも良好な経路追従走行を維持しやすいという特性がわかった．この理由として，6.2 節のスラローム経路と比べて，8 の字経路は車両の走行方向があまり頻繁に変化しないということが影響していると考えられる．8 の字の円周部分を走行している間は車体の走行方向が変わらないため，実際の走行位置が CS の予測よりも後方であることで最適でない θ の値が計算・送信された場合でも目標経路から大きく外れない．そのため，CS におけるフィードバックを活用した予測結果の補正によって，目標経路から逸脱しない制御精度を維持できると考えられる．8 の字よりも走行方向が変化することが多い形状の経路では，速度に関するディジタルツインモデルの許容誤差の下限値は本実験の 0.6 よりも 1 により近い比率になると考えられる．また，実際の走行速度の方が速い場合でも，その誤差比が 1.1 程度の小ささに収まる限り，小型走行車の経路追従走行に殆ど影響を与えない．この上限の値は 6.2 節のシミュレーション結果と共通している．8 の字経路の中央のような車体の走行方向が変わる WP を通過する際には CS は θ の値を大きく変えて送信するが，実際の UV は予測位置よりも前方に進んだ状態でその信号を受

信する．そのため，目標経路から外れてしまった後で走行方向を変更することになる．この点がシミュレーションの場合と共通している．一方で，走行車の速度が一定水準より遅い場合は走行結果が著しく悪化する場合がある．この理由として，実際の走行速度が遅すぎる場合，実際の走行位置と CS による予測結果との乖離が大きくなるため，車両が到達する位置次第で，適切な θ 値と予測による θ 値との絶対値の差が非常に大きくなることが考えられる．

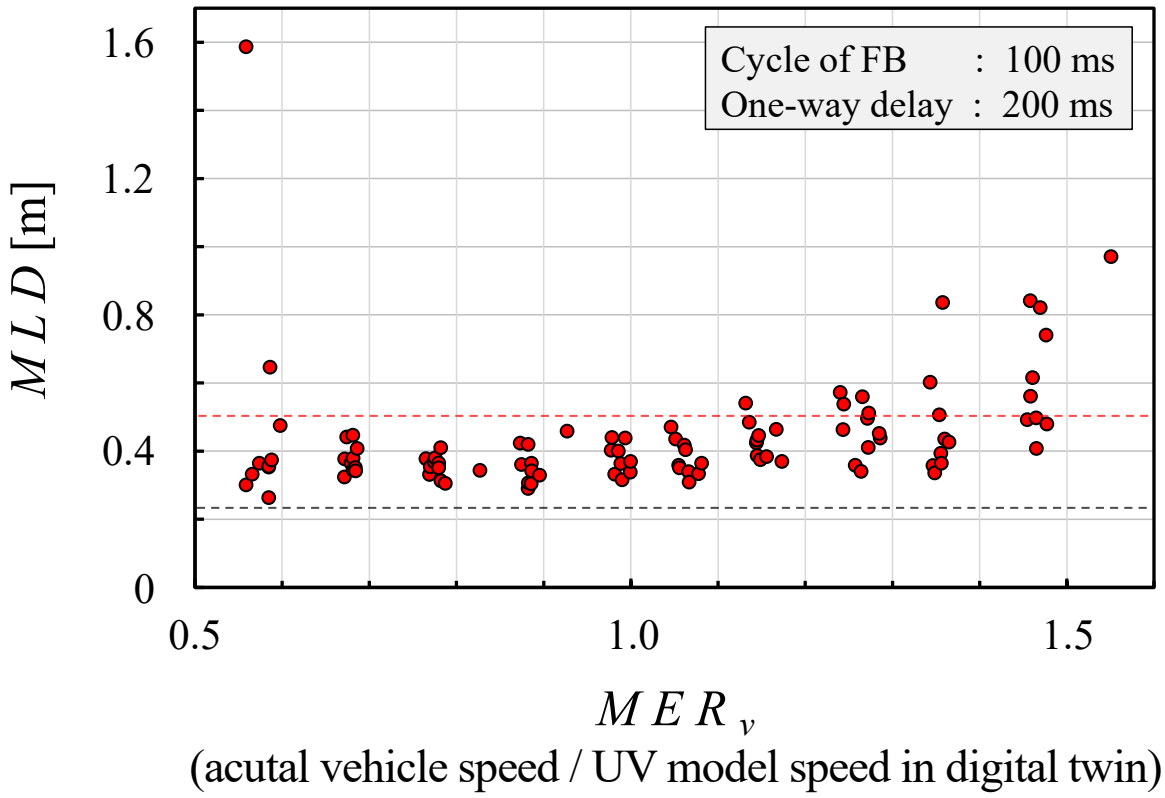


図 6-11　走行速度のディジタルツイン誤差に対するラジコン車の *MLD*

6.4　結言

提案システムにおける遠隔経路追従走行において，ディジタルツインの予測と実際の UV の状態との誤差はなるべく小さい方が良い．目標とするべき誤差の小ささを判断するためには，経路追従走行における許容誤差について定量的に評価することが必要である．

小型の 4 輪走行車に関するディジタルツインの誤差が CS による遠隔経路追従制御に及ぼす影響について，シミュレーション評価および実験によって評価した．伝送遅延が無い状態では正確な UV 経路追従制御が可能となるシステムを想定し，遠隔経路追従走行が困難となる状況の制御特性をシミュレーションによって評価した結果，以下のことがわかった．

(1) 制御対象の UV とディジタルツイン上の UV モデルとの間の走行速度の誤差比が 0.9 から 1.1 の範囲内であれば，正確な経路追従走行が可能となる．

(2) ステアリング角の誤差比についても，誤差比は 0.9 から 1.1 の範囲内で正確な経路追従走行が可能となる．ただし，ステアリング角の誤差は走行速度よりも UV 走行経路により大きな影響を与える．ステアリング角がディジタルツインのモデルよりも UV の方が小さい場合，UV は目標経路からより大きく乖離して走行する危険性が高い．

(3) ステアリング角の誤差比が 0.9 であっても，UV の走行速度が比較的遅い場合は状態情報の送信周期が 500 ms 以上の長さであっても十分な精度の遠隔経路追従走行を達成できる．

実際に提案システムにおけるディジタルツインの誤差の影響を確認するため，小型ラジコン車を用いた実験を実施した結果，以下のことが確認された．

(4) 伝送遅延またはフィードバック周期が大きい場合でも，ディジタルツインを活用した予測制御によって良好な遠隔経路追従走行を達成できた．

(5) 8 の字型の目標経路において, 小型ラジコン車とディジタルツインの走行車モデルとの間の走行速度の誤差比が 0.6 から 1.1 の間の場合に，良好な遠隔経路追従走行を達成できた．

シミュレーション評価と実験では，制御対象の UV とディジタルツイン上の UV モデルとの間の走行速度の誤差について検討した点が共通する．しかし 2 つの検討では目標経路の形状が異なる．シミュレーション評価ではスラロームの経路であった．実験では，実験環境上の制約から 8 の字の経路とした．この差によって，走行速度の誤差についての許容範囲の下限に差が生じたが，上限は概ね一致している．遠隔経路追従走行の難しさという観点から評価すると，シミュレーションの結果の方がより厳密であると考えられる．シミュレーション評価の結果を基準として実験による結果を考慮すると，目標経路の形状によって UV 遠隔走行制御における走行速度の誤差の許容範囲の下限が変化する可能性が考えられる．経路の形状に関わらず安全な制御を維持したい場合，UV とディジタルツイン上の UV モデルとの間の誤差は 0.9 から 1.1 の比率に収まるように UV モデルを調整すべきである．

UV の挙動をディジタルツインで予測する場合，リアルタイムなフィードバック情報だけでなく過去の情報を活用することで，ディジタルツインモデルの誤差を小さくすることが可能であると考えられる．これは UV だけでなく，路面の状態や歩行者等の制御対象以外の動体についても同様である．ディジタルツインを活用した遠隔制御システムにおけるモデルの動的な高精度化については別途研究を進める必要がある．

第 7 章　ジッタバッファにおける制御信号バッファリング時間の動的最適化

7.1　緒言

CS と端末との間のパケット通信では，ネットワーク上の様々な要因によって伝送遅延のジッタが発生する．提案システムにおいて，ジッタバッファは制御信号の伝送遅延ジッタを吸収することで伝送遅延を一定に保つ役割を持つ．制御信号の伝送遅延の安定化は UV における制御周期の安定化および CS における予測制御の安定化という利点をもたらす．

ジッタバッファによって制御信号を適切にバッファリングすることは，UV の遠隔経路追従制御の精度において重要な要素である．図 7-1 にジッタバッファを介した伝送遅延の例を示す．図中の D はジッタバッファで保持された信号の固定的伝送遅延の値である．青線が示すように，伝送遅延変動に対してジッタバッファによるバッファリング時間が十分に長い場合は，その変動を十分に吸収できる．しかし長すぎる場合，本来の伝送遅延よりも大幅に長い伝送遅延となる．ディジタルツインを活用した予測制御を実行する提案システムにおいて，CS は時間 D 先の UV の状態を予測する．ジッタバッファを介した固定遅延 D が長すぎる場合，予測誤差の増加をもたらす可能性がある．例えば，ステアリング角の状態についてディジタルツインの UV モデルに誤差がある場合，D の長さに応じてディジタルツイン上の予測と実際の UV の状態との間の誤差が大きくなる．一方で，赤線が示すようにバッファリング時間が短すぎる場合，ジッタバッファは伝送遅延変動を十分に吸収できない．そのため，制御周期の乱れによって UV の走行状態が不安定になる．このように，バッファリング時間と予測制御の精度・安定性の間にはトレードオフ関係が成り立つ．

上記の問題を解決するため，提案システムには伝送遅延の変動に応じてジッタバッファを最適化する仕組みが必要である．CS を用いた遠隔制御では，突発的に伝送遅延特性が変化する可能性が考えられるため，リアルタイムな伝送遅延特性に応じて動的に最適化する方法が良いと考えられる．そのため，ジッタバッファにおけるバッファリング時間を動的に調整する仕組みを考案し，第 4 章で解説した提案システムに適用した．この仕組みには動的調整に

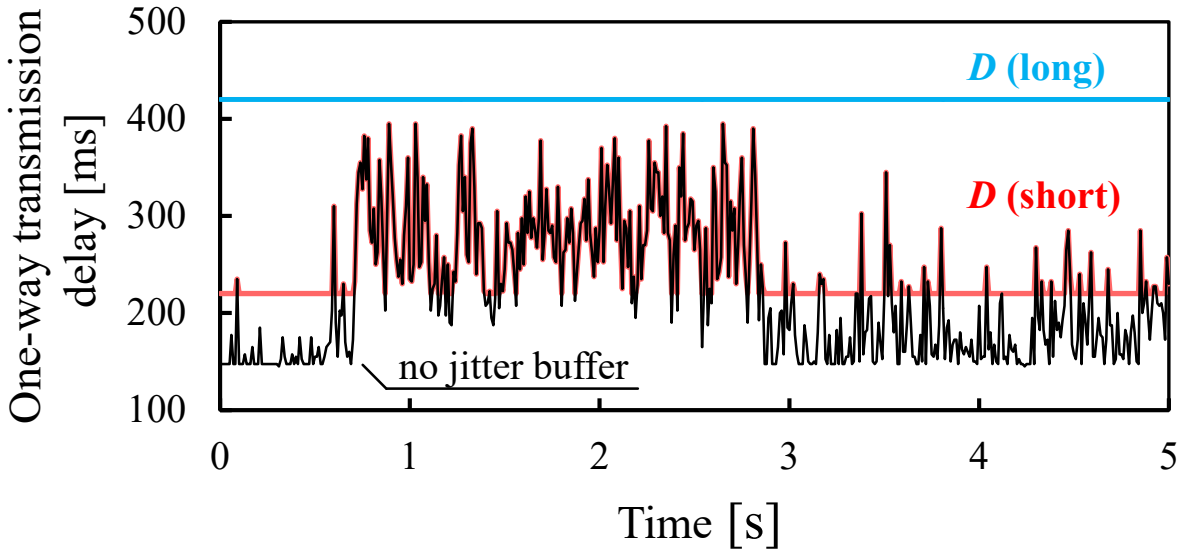


図 7-1　ジッタバッファを介した制御信号の伝送遅延変動

関する 2 つのパラメータが含まれる．これらのパラメータを適切に設定することで，UV 遠隔経路追従制御におけるバッファリング時間の動的最適化を達成できる．本章ではバッファリング時間の動的最適化の仕組み，およびシミュレーションによって UV 遠隔経路追従制御における動的最適化の有効性を定量的に評価した結果について解説する．この評価では，UV の遠隔制御中に伝送遅延の変動特性が急激に変化する状況を想定した．

7.2 節では，ジッタバッファによる制御信号バッファリング時間の長さが UV 遠隔経路追従制御に及ぼす影響を確認した結果について簡易的に述べる．7.3 節では，本検討で考案した制御信号バッファリング時間の動的調整の仕組みを解説する．7.4 節では，シミュレーション条件を述べる．7.5 節では，シミュレーション評価の結果を述べる．7.6 節にて本章の内容をまとめる．なお，本章でも UV 走行経路の評価指標として *MLD* を用いた．

7.2　UV の遠隔経路追従制御に対する制御信号バッファリング時間の影響

第 4 章で解説したシミュレータを用いて，ジッタバッファの制御信号バッファリング時間 *D* を変化させた場合の *MLD* の変化を評価した．経路追従走行の目標経路を図 7-2 に示す．CS

と UV との間の伝送遅延は，5.2.3 項で解説したインターネット区間の伝送遅延モデル A，B，C をそれぞれ Wi-Fi と組み合わせた 3 種類のうち 1 つを NE で再現した．パケットロスは 0%とした．*MLD* の変化をより明確化するために UV の走行速度 v は 2.0 m/s とした．ステアリング角 θ の最大値は 0.52 rad とした．UV の状態情報の送信周期は 100 ms とした．また，ディジタルツイン上の UV モデルの誤差は設定せず評価を実施した．

図 7-3 にシミュレーション結果を示す．各線はそれぞれインターネット区間の伝送遅延モデルを示す. 各点は条件ごとにシミュレーションを 10 回実施した時の *MLD* の平均値を示す. 図 7-3(a)は横軸をジッタバッファのパラメータ *D* としたグラフである．図 7-3(b)は(a)と同じ結果を横軸を別の指標に置き換えて図示したグラフである．*BPR* (Buffered Packet Rate) は全制御信号パケットのうち CS から UV に到達した時の伝送遅延が *D* より小さいパケットの割合を示す．言い換えれば，ジッタバッファによって伝送遅延を一定にされたパケットの割合である．それぞれの *BPR* の値は *D* から求めた．図 7-3(a)の各線の横軸の最大値は伝送遅延モデル A，B，C の順にそれぞれ 254 ms，296 ms，363 ms である．これらは，CS と UV との間の片道伝送遅延の最大値である．この時，ジッタバッファによって全ての制御信号が一定の伝送遅延として UV に使用された．これは，図(b)において *BPR* = 100%の結果に相当する．

図 7-3(a)より，*D* が低下するにつれて *MLD* が増加したことがわかる．3 つの線に共通する傾向として，*MLD* の増加が小さい区間と大きい区間に分かれる．*MLD* の増加が小さい限り，*D* はなるべく小さい方が良い. 図 7-3(b)より，*BPR* が 100%から 96%に低下する範囲では *MLD* の変化が小さい．その際，*MLD* の値は伝送遅延の種類ごとに 1.9，1.7，1.0 cm 増加した．ディジタルツイン上の UV モデルに実際の UV に対する誤差が存在する場合，良好な UV 走行制御を達成できる *BPR* の範囲はこのシミュレーション結果とは異なる可能性が高い．また，制御信号のエンコーディング/デコーディング方式にも影響される可能性が考えらえる．しかし，図 7-3 の結果から *BPR* が UV 遠隔走行制御においてジッタバッファによる制御信号バッファリング時間を最適化するための指標として有用であると考えられる．

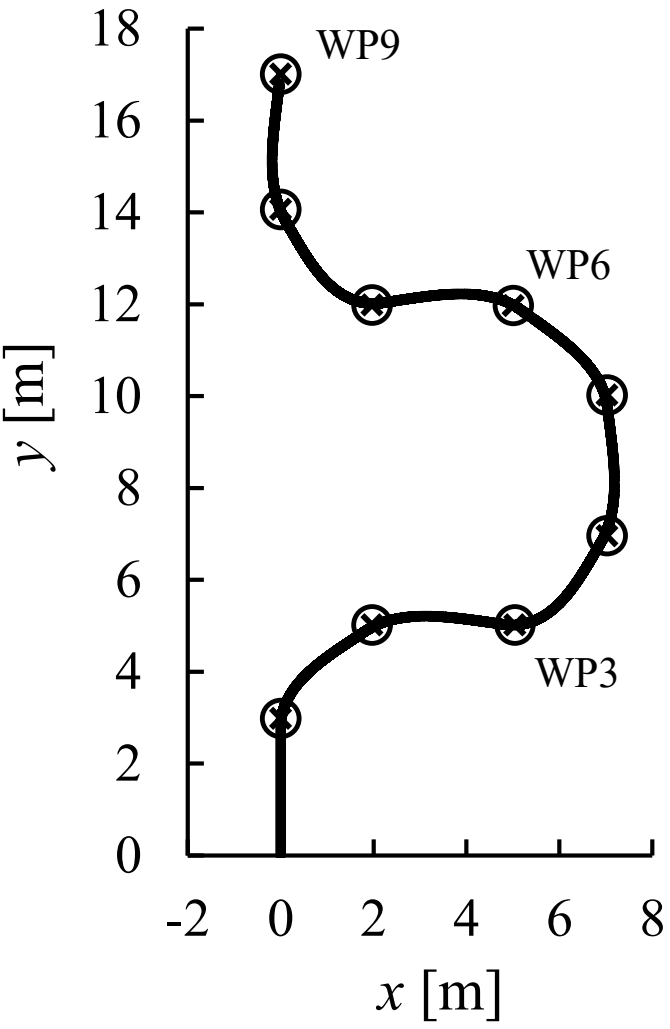
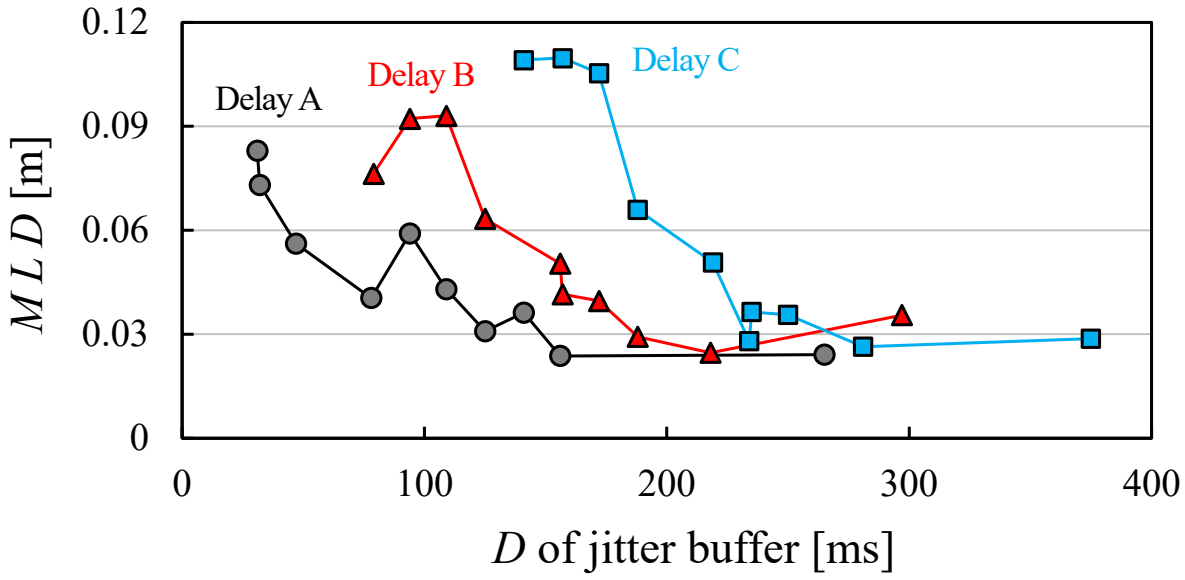
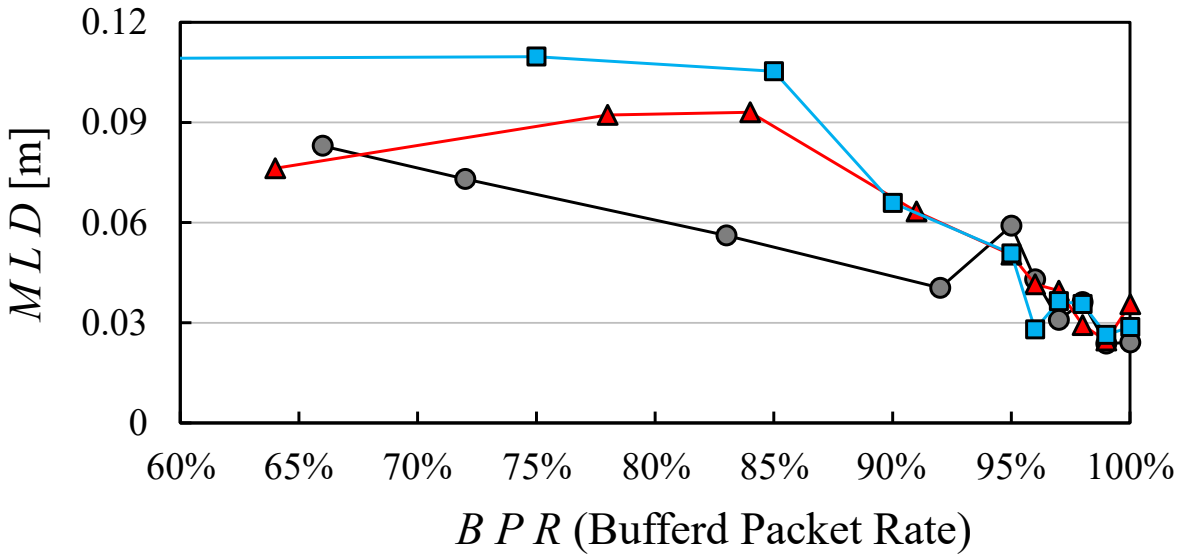


図 7-2　目標経路



(a)　横軸 ： バッファリング時間 *D*



(b)　横軸 ： ジッタバッファで伝送遅延を固定化された制御信号パケットの割合

図 7-3　ジッタバッファのバッファリング時間 *D* に対する *MLD* の変化

7.3　制御信号バッファリング時間の動的調整の仕組み

7.2 節の結果を参考に，ディジタルツインを適用した CS による UV 遠隔経路追従制御における制御信号バッファリング時間を動的に調整する仕組みを考案した．図 7-4 にその仕組みの概要を図示する．また，図 7-5 にバッファリング時間の動的調整が適用された場合の制御信号の伝送について図示する．

UV は伝送遅延変動を吸収するために受信した制御信号をジッタバッファで保持する．その際，各制御信号の伝送遅延を CS の送信時刻と UV の受信時刻から計算する．算出された実際の伝送遅延の値はその時点から数秒前までの間のものが UV 内に記録される．この時間間隔を本章では *TRD* (Time to Record Transmission Delay) と呼称する．これは本章で考案した動的調整方法におけるパラメータの 1 つである．UV は一定周期で状態情報を送信する．この時, 図 7-4 に示すように UV は伝送遅延の記録から算出した *D* の要求値を状態情報に含める．*D* の計算方法は以下の(1)から(2)に示す.

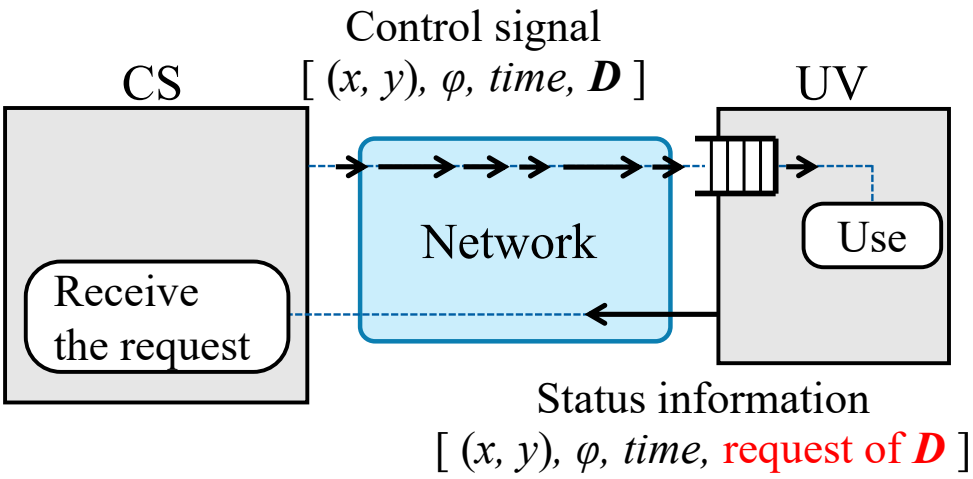
(1) 制御信号の伝送遅延の記録から，遅延値を昇順で並び替えた累積分布を作成する．
(2) 分布における所定の累積相対度数 *CRF* に該当する遅延値を特定する．この値を *D* の要求値とする．

上記の(2)の *CRF* は 7.2 節の *BPR* と等しい意味を持つ．*BPR* もこの仕組みにおけるパラメータの 1 つであり，所定の値が与えられる．状態情報を用いることで，ジッタバッファによる制御信号の *BPR* が所定の割合を維持するようにシステムにおける *D* の値を変更し続ける．CS は実際の伝送遅延変動に対応した *D* の値を認識するとともに，制御信号の中にジッタバッファで保持される最大バッファリング時間 *D* を含めて送信する．

上記の仕組みによって，UV の遠隔制御中に制御信号のバッファリング時間が動的に調整され続ける．また，過去に送信した制御信号が UV に使用されるタイミングを概ね把握できるため，CS はディジタルツインを活用した走行状態の予測をより適切に実行できる．図 7-5 は制御信号の伝送遅延の変化を例示する．ある時刻t_i に CS が送信した制御信号 *i* がD_1 という

値を含むとする．送信後，CS では信号 i が時刻$t_i + D_1$に UV で使用されると想定した予測が実行される．UV では，ジッタバッファが時刻$t_i + D_1$に制御信号 i を UV の駆動部へ転送する．また，UV は本来の伝送遅延d_iを記録する．UV は自身の現在の状態だけでなく，過去数秒間の d の値から算出した D の要求値を含めて状態情報を送信する．状態情報を受信すると，CS はD_1をD_2に変更する．

以上に解説した制御信号バッファリング時間の動的調整機能を，第 4 章で解説したシミュレータにおける CS および UV プログラムに実装した．この機能は *BPR* と *TRD* という 2 つのパラメータに所定の値を設定することで動作する．このシミュレータを活用して，制御信号バッファリング時間の動的最適化の有効性を定量的に評価した．



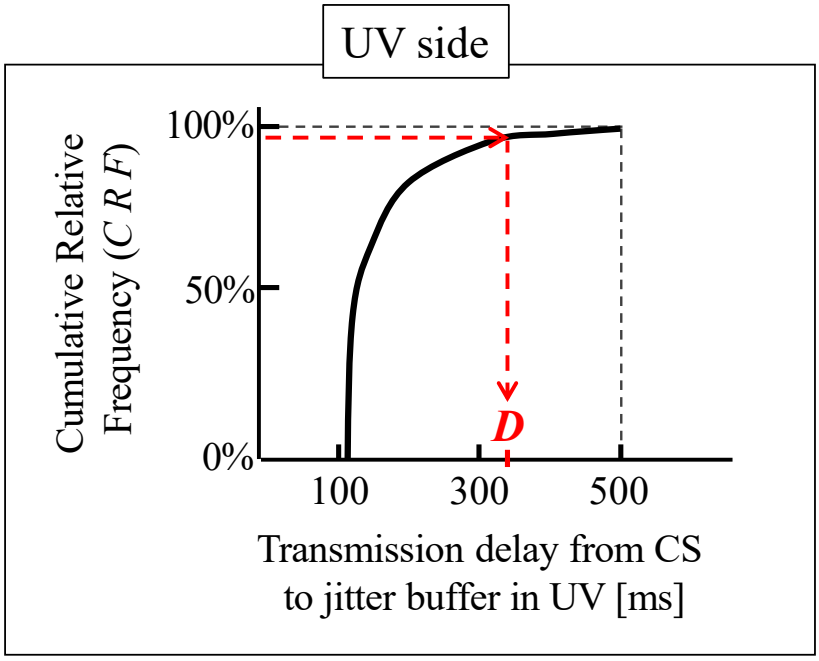


図 7-4　UV からの状態情報を活用した D の値の動的調整

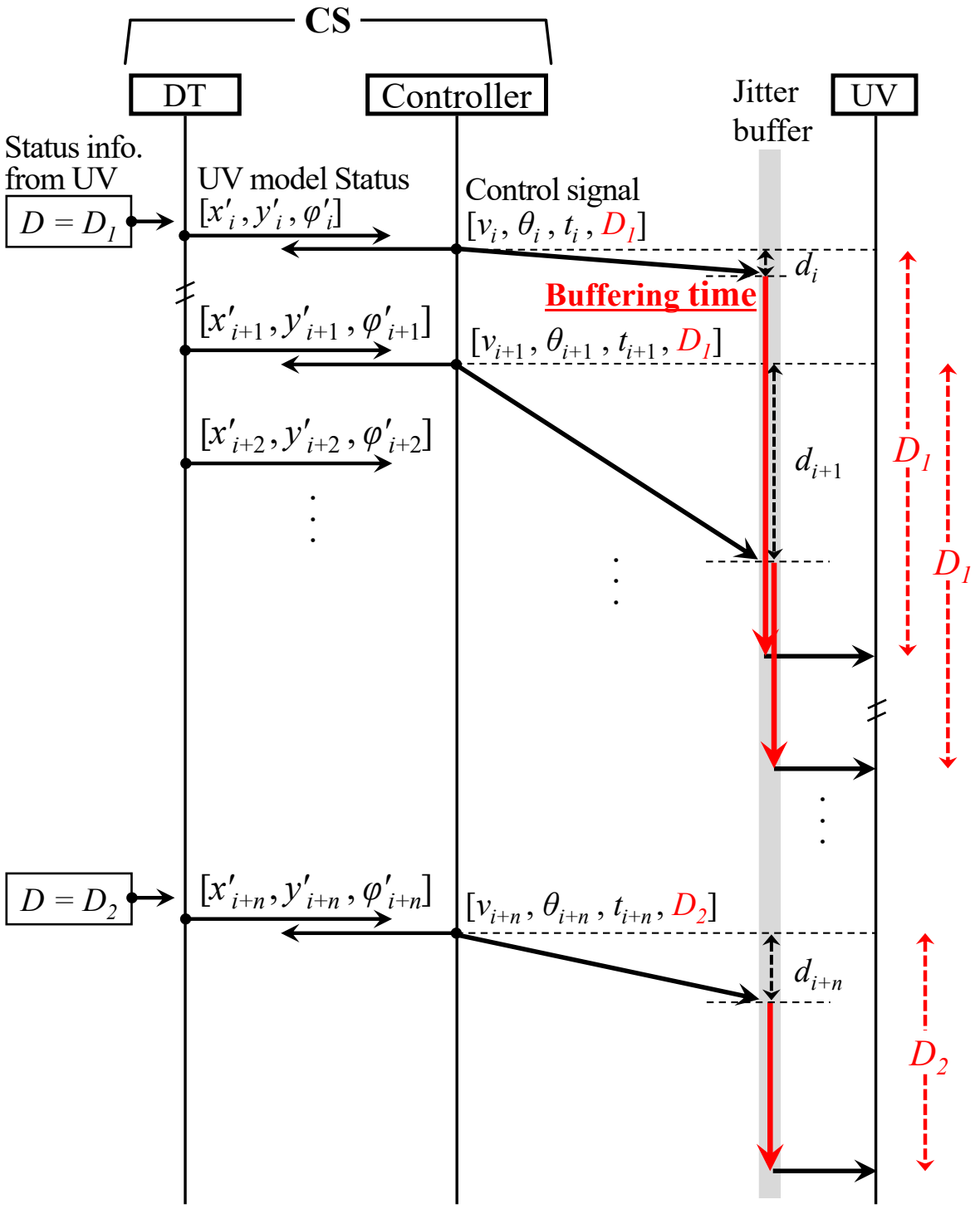


図 7-5　D の動的調整を適用した場合の制御信号バッファリング時間

7.4　シミュレーション条件

7.3 節で解説した動的調整機能のパラメータを変更した時の UV の走行特性を定量的に評価するシミュレーションを実施し, その結果から UV 遠隔経路追従制御におけるパラメータ *BPR* と *TRD* の最適値を求めた．そのパラメータの組合せを，提案システムにおいて制御信号バッファリング時間の動的最適化が達成できる状態とみなした．また，動的最適化が適用された場合とそうでない場合の UV 走行特性を定量的に評価した．

7.4.1 項では，本検討の経路追従走行における目標経路を示す．7.4.2 では，NE で再現した伝送遅延およびパケットロスについて解説する．7.4.3 では，CS と UV のパラメータ値について述べる．ここでは，本章の動的調整機能のパラメータである *BPR* と *TRD* を UV 側のパラメータの一部としてまとめて述べる．

7.4.1　目標経路

第 5 章の検討と同様に，スラローム状の経路が目標経路として適すると考慮した．図 7-6 に本検討の目標経路を示す．これは第 5 章の検討の目標経路と同じくスラローム形状であるがより長い経路である．後述の 7.4.2 項で述べるように，本検討ではインターネット区間の伝送遅延の変動特性が急激に変化する状況を模擬した評価を行った．本評価の条件では，UV が図 7-6 に示す経路を始点から終点まで走る間，伝送遅延変動特性の変化は 3~4 回発生する．

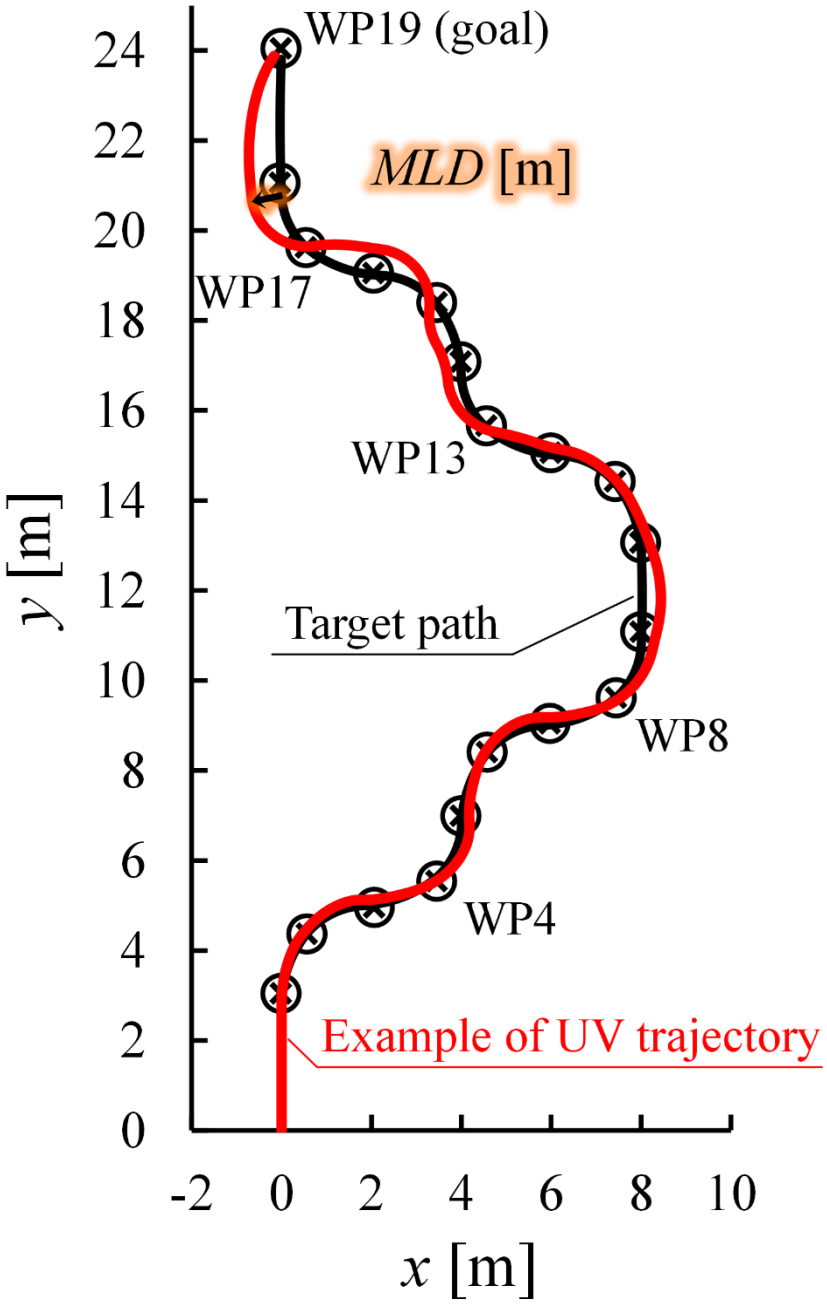

WP19 (goal)
MLD [m]
WP17
WP13
WP8
WP4
Target path
Example of UV trajectory
y [m]
x [m]
24
22
20
18
16
14
12
10
8
6
4
2
0
-2
0
2
4
6
8
10

図 7-6　目標経路

7.4.2　CS と UV との間の伝送遅延

以下に伝送遅延とパケットロスについて個別に解説する．

A．伝送遅延

図 7-7 と図 7-8 にそれぞれ NE におけるインターネット区間と無線アクセス区間の伝送遅延データセットを示す．インターネット区間の伝送遅延データセットは 120 秒分であり，図 7-7 はその一部を示す. 無線アクセス区間のデータセットは第 5 章と同様の Wi-Fi の実測値である．これらの組合せによって，伝送遅延とその変動が大きい状態の通信状況を再現できる．伝送遅延は CS と UV との間の上り通信および下り通信の双方に同様のものを発生させた．シミュレーション中に発生させた片道伝送遅延の最大値は 430 ms であった．

以下に，インターネット区間の伝送遅延データセットを作成した方法を解説する．第 5 章で実際のクラウドサーバとの間の RTT を測定したことを述べた．本検討では，その結果の中で最も伝送遅延が大きかったフランクフルトのサーバとの測定結果に注目した．RTT の測定結果を約 10 秒の時間間隔で分割し，それぞれの測定結果を RTT の変動特性に応じて 2 つのグループに分類した．グループ S には RTT の変動が小さかった測定結果を，グループ L には変動が大きかった測定結果を分けた．グループ S の結果はサーバとの通信品質が良かった場合の伝送遅延特性を示す．グループ L は通信品質がグループ S の状態から劣化した場合の伝送遅延特性を示す．

2 つのグループの RTT 測定結果をそれぞれパレート分布のフィッティングに用いた．その方法は第 5 章の式(5.1)から(5.3)に示す方法と同様である．その結果，グループ S の測定結果から伝送遅延モデル S を，グループ L の結果から伝送遅延モデル L を作成した．伝送遅延モデル S，L におけるパレート分布のパラメータ，伝送遅延の最小値および最大値を表 7-1 に示す．これら 2 つの伝送遅延モデルを切り替えながら伝送遅延のデータセットの値を生成することで，図 7-7 に示す伝送遅延データセットを生成した．モデルの切り替えは 10 秒ごとに行った．

表 7-1　伝送遅延モデルの特性

Name	Parameters of Pareto		Max
	K	*A* (Min)	
Model S	48	120 ms	142 ms
Model L	16.6	140 ms	203 ms

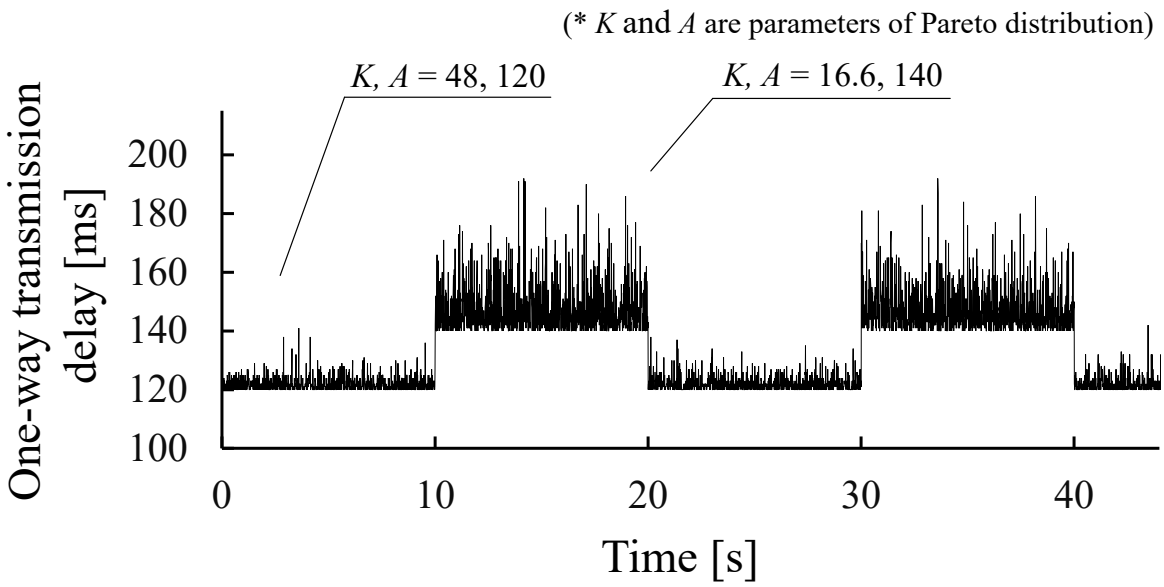


図 7-7　インターネット区間の片道パケット伝送遅延

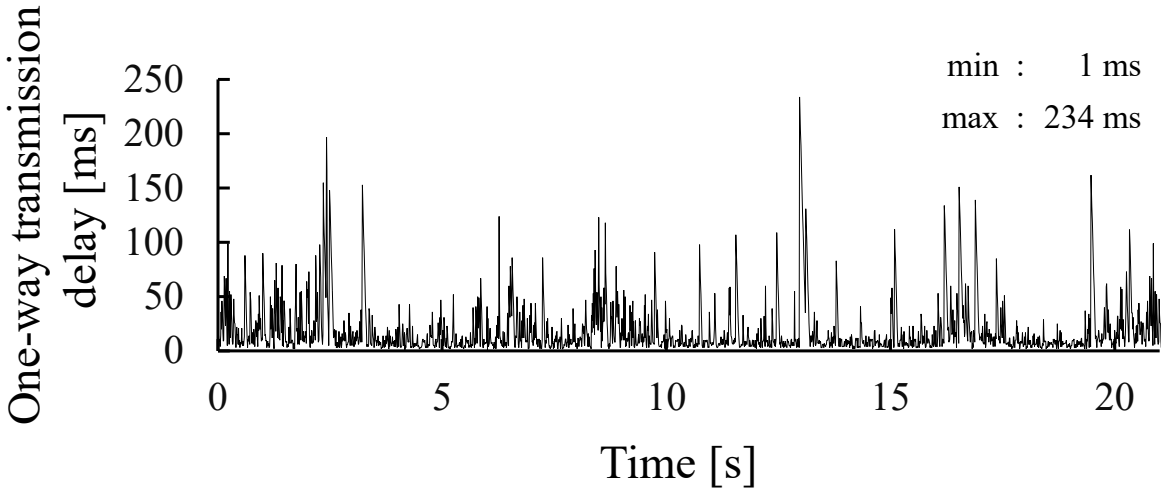


図 7-8　Wi-Fi 区間の片道パケット伝送遅延

この伝送遅延データセットを用いると，CS と UV との間の伝送遅延特性が頻繁かつ急激に変化する状況を再現できる．UV が 7.4.1 項に示す経路に沿うように走行する間，伝送遅延特性の変化が 3~4 回発生する．一般的に，このような伝送特性の変化は UV の遠隔走行制御の安定性を著しく劣化させる．本検討で再現させる伝送遅延は，本章で考案した動的調整機能による UV 制御特性の改善効果を評価する上で適切なものであると判断した．

B．パケットロス

より現実的にパケットロスを再現するため，ITU-T 勧告 G.191 で規定されている”Discrete Gilbert Elliot Channel Model“を用いてパケットロスを発生させるように NE を調整した[76, 77]．図 7-9 にそのモデルを図示する．通信品質が良好な場合のパケットロス率は 0%，通信品質が悪い場合は 50%とした．総パケットロス率 r は APNIC の統計[75]を参照して 3%とした．パケットロスはバーストロスかランダムロスのどちらかの特性で再現させた．バーストロスの場合，バースト性指標 $b = 0.8$ と設定した．ランダムロスの場合は $b = 0.2$ とした．

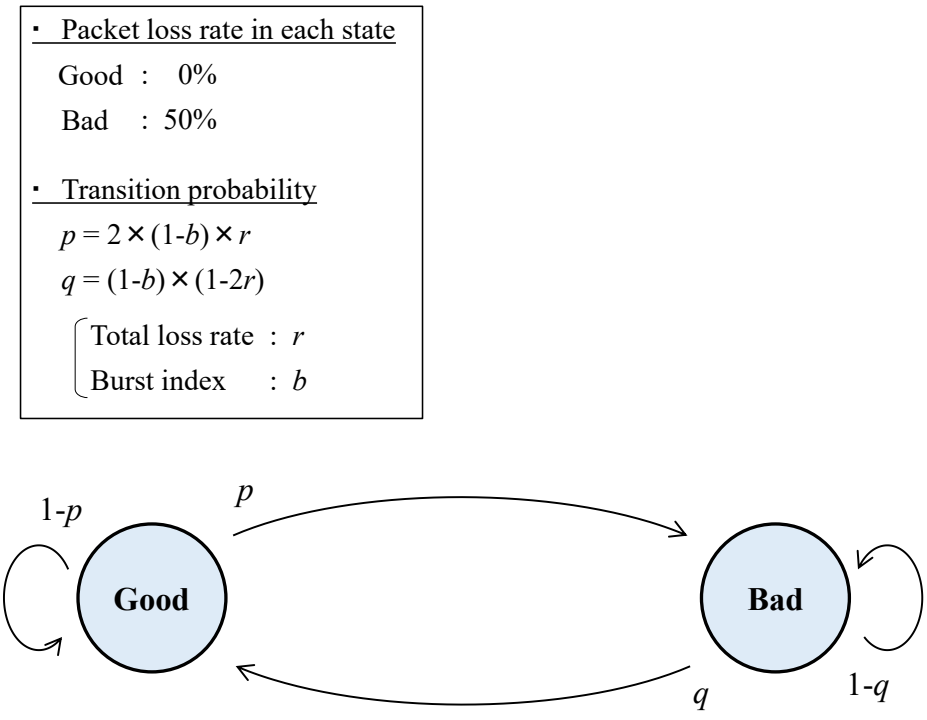


図 7-9 “Discrete Gilbert Elliot Channel Model”の概要

7.4.3　CS と UV のパラメータ

UV の走行速度 v は，屋内環境で小型 UV が安全に走行できる巡航速度を考慮して 1.0 m/s とした[78, 79]．ステアリング角 θ の最大値は 0.52 rad とした．状態情報の送信周期は 100 ms から 500 ms の間で設定した．第 6 章のシミュレーション結果から，CS による予測制御において制御対象の UV のステアリング角がディジタルツイン上の UV モデルよりも小さく動く場合に，より大きな予測誤差が生じやすいことが明らかになった．比較的大きな予測誤差を生じさせるため，本検討ではMER_θ = 0.8 を UV に適用した．この値は，小型ラジコン車を用いた予備実験から求めた．ラジコン車を様々なステアリング角で走行させた結果，MER_θ の最大値は 0.802 となった．この時，制御信号 θ = 0.20 rad に対してラジコン車のステアリング角 θ_{uv} = 0.1604 rad であった．

制御信号バッファリング時間を動的に調整するためのパラメータである *BPR* は 90%から 100%の間で設定した. 図 7-10 に *TRD* = 3 s とした時の *BPR* による *D* の値の変化を例示する．*BPR* が 90%より低い場合，*D* の変化は小さい．*BPR* が非常に低い場合，ジッタバッファによる伝送遅延変動の吸収は殆ど機能しない．これらの理由から *BPR* の設定範囲を決定した．直近の伝送遅延変動を参照するためのパラメータ *TRD* は 1 秒から 5 秒の間で設定した．

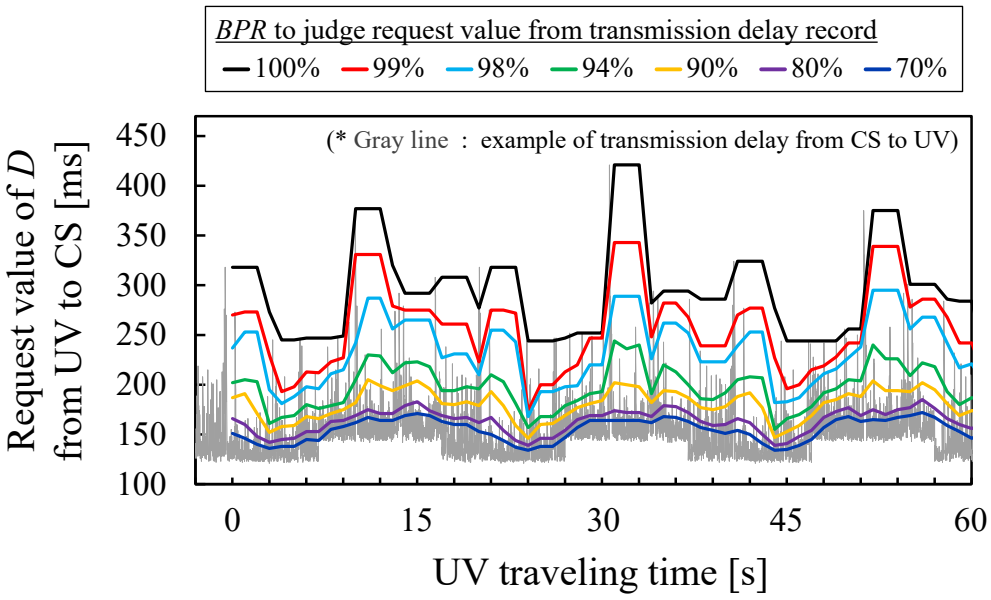


図 7-10　伝送遅延変動と *BPR* による *D* 要求値の変化

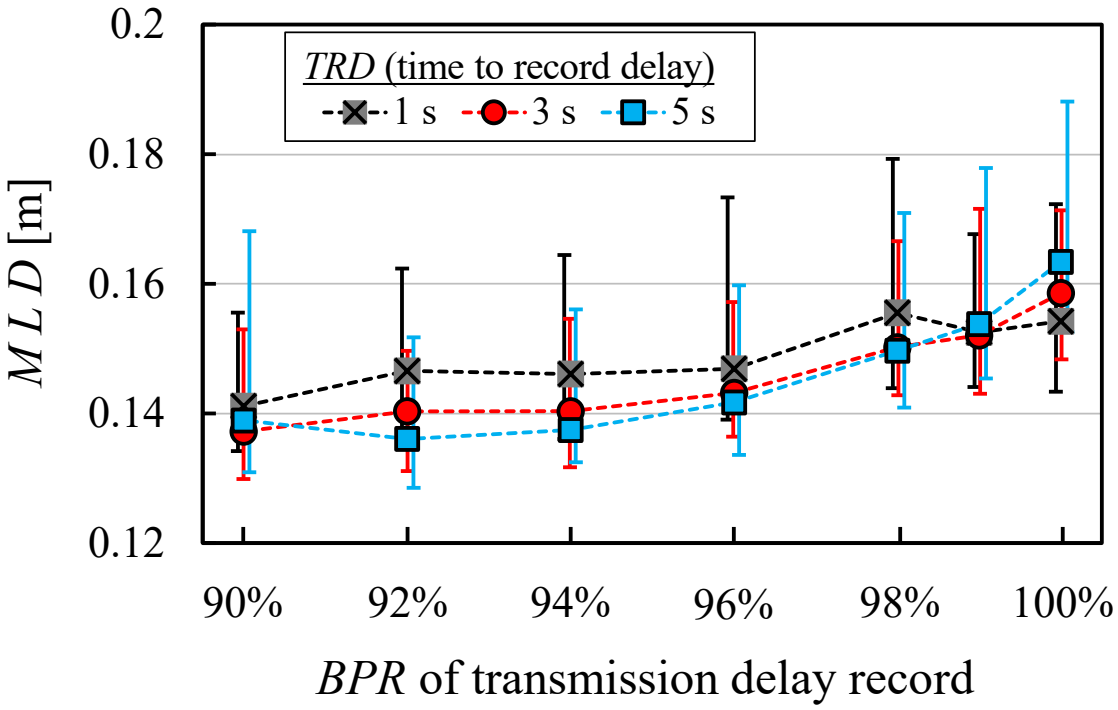
7.5　評価結果

提案システムを用いて 2 種類のシミュレーション評価を実施した．まず，考案した動的調整機能が UV 制御結果に及ぼす影響を定量的に評価するとともに，最も良い UV 遠隔経路追従走行を達成した時のパラメータの組合せを特定した．それらの最適なパラメータが，UV 遠隔経路追従制御を実施する提案システムにおける制御信号バッファリング時間を最適化できる状態であると考えた．次に，動的最適化の有無による UV 制御結果の差を定量的に評価した．シミュレーションは条件ごとに 100 回実施した．以下の図 7-11 と図 7-13 では，各点は条件ごとの結果の中央値を，誤差範囲は第一四分位数から第三四分位数までのばらつきを示す．

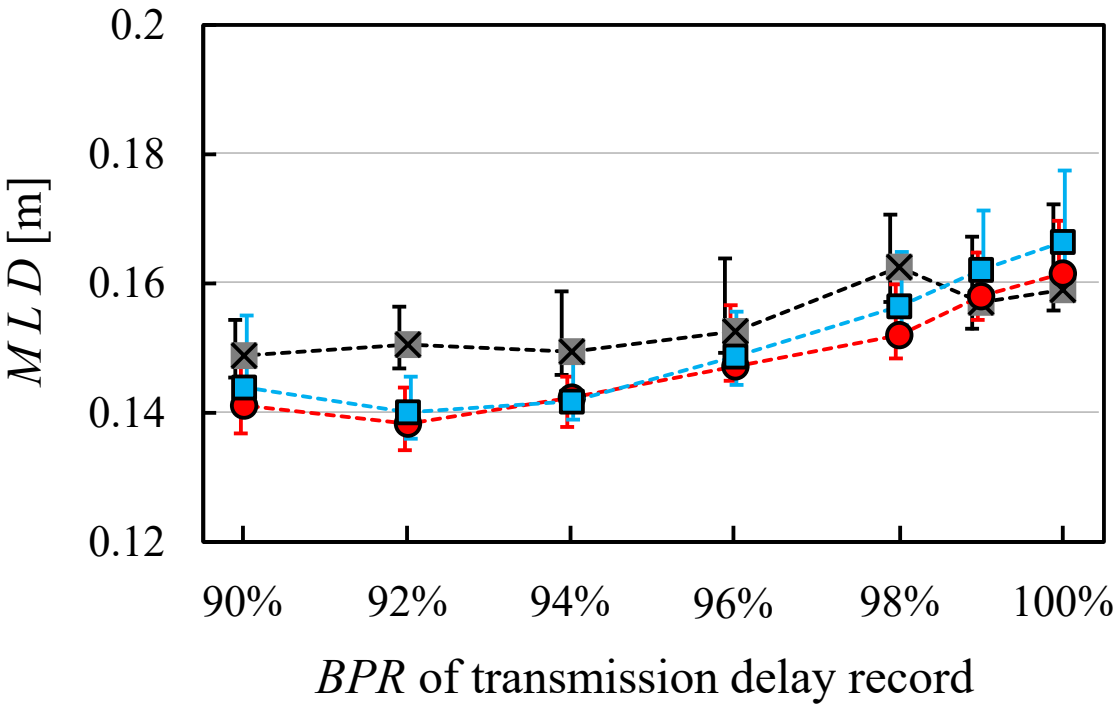
図 7-11 にパラメータ *BPR* と *TRD* を変化させた場合の *MLD* の変化を示す．図の(a)と(b)はそれぞれ NE でバーストパケットロスまたはランダムパケットロスを再現した時の結果である．横軸と縦軸はそれぞれ *BPR* と *MLD* を示す．それぞれの破線は *TRD* の違いを示す．状態情報の送信周期は 300 ms に設定した．図 7-11(a)と(b)を比較すると，*MLD* の中央値にあまり差はない．しかし，ばらつきは(a)の方が明確に大きい．図 7-12 にバーストパケットロスの場合の *MLD* の最大値を示す．

バーストパケットロスの結果に注目する．*BPR* = 100%の時，*MLD* の中央値と最大値は比較的大きかった．それぞれの破線において，*MLD* の最大値は *BPR* を 100%から 92%に下げた時に最も小さくなった．さらに，*TRD* = 3 s の時の最大値が最も小さかった．*MLD* のばらつきは *BPR* = 92%かつ *TRD* = 3 s の時に最も小さかった．この時，中央値も比較的小さい値となった．以上から，本検討では *BPR* = 92%かつ *TRD* = 3 s が提案システムによる UV 遠隔経路追従走行における制御信号バッファリング時間の動的最適化を達成する最適なパラメータであったと判断した．この場合，*D* の要求値の時系列変化は概ね図 7-10 の緑と橙の線の中間となった．なお，ランダムパケットロスの場合このパラメータの時に *MLD* の中央値が最小となった．

上記の結果に基づいて，バッファリング時間動的最適化を適用した際の UV 経路追従制御結果の変化を評価した．図 7-13 と図 7-14 にその結果を示す．これは NE にバーストパケットロスを再現させた場合の結果である．横軸は状態情報の送信周期を示す．状態情報は UV の



(a)　バーストパケットロスの場合



(b)　ランダムパケットロスの場合

図 7-11　パラメータ *BPR* と *TRD* による *MLD* の変化

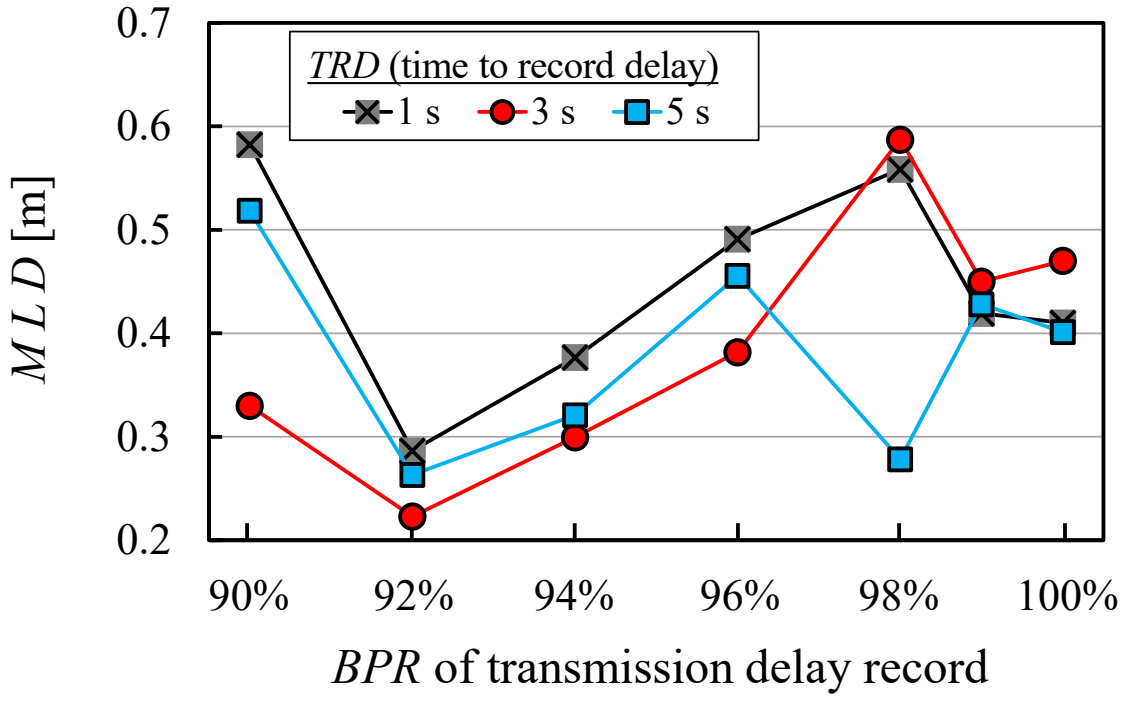


図 7-12　バーストパケットロスの場合の *MLD* の最大値

現在の走行状態に基づいてディジタルツイン上の UV の走行状態の予測結果を修正するために活用される．バッファリング時間動的最適化の機能を適用されたシステムにおいては，各制御信号のバッファリング時間に直接的に関係する *D* の値を更新するためにも必要な情報である．この送信周期は CS による UV 遠隔制御特性に大きく影響する．図中のそれぞれの破線は動的最適化機能の適用の有無を示す. 青い破線は動的最適化機能が適用された, つまり *BPR* = 92%と *TRD* = 3 s が設定された場合の結果を示す．黒と赤の破線は動的最適化機能が適用されず，*D* の値として固定値が与えられた場合の結果である．*D* = 430 ms の場合，制御信号の伝送遅延ジッタは完全に吸収され，全ての制御信号が一定の伝送遅延として UV に使用された．*D* = 192 ms は，UV が始点から終点まで走行する間の制御信号の伝送遅延の中で，*BPR* = 92%に相当する値である．本検討の目標経路を UV に 1 m/s で走行させると，走行時間は約 34 秒間である．その時間間隔における *BPR* = 92%の遅延値を求めるため，UV 遠隔制御シミュレーションを実施する前に NE で 34 秒分の片道伝送遅延を再現した．192 ms はその再現を 100 回繰り返した時の中央値かつ最頻値である．

グラフから, 状態情報の送信周期が増加するにつれて *MLD* の値も増加したことが確認できる．バッファリング時間の動的最適化が適用された場合の結果を *D* が固定値 430 ms であった場合と比較する．*MLD* の中央値，ばらつきおよび最大値は，動的最適化が適用された場合の方が明確に小さかった．*MLD* のばらつきの縮小および最大値の低下は送信周期が長い程明確になった．発生し得る伝送遅延の最大値を *D* とした場合，CS による走行状態の予測誤差もその大きさに応じて大きくなりやすい．遅延の状況に応じて D を最適化することがその予測誤差を小さくするため，CS による遠隔制御の精度が改善したと考えられる．*D* が固定値 192 ms であった場合と比較すると，*MLD* の中央値にはあまり差がなかった．一方で，ばらつきと最大値は動的最適化が適用された場合の方が小さかった．言い換えれば, UV 遠隔経路追従制御の精度が向上したと言える．特に，状態情報の送信周期が長い程制御精度の改善が顕著であった．*D* = 192 ms の場合の制御精度の不安定さは，走行時間約 34 秒で BPR = 92%に相当する固定値を *D* とした場合，インターネット区間の通信品質が良好な状態から悪化した時に制

御信号の伝送遅延変動をジッタバッファで十分に吸収できなくなったことが原因と考えらえる．

以上の定量的評価から，リアルタイムな伝送遅延変動に応じた制御信号バッファリング時間の動的最適化はディジタルツインを適用した CS による UV 遠隔経路追従制御において有効であることが明らかになった．本検討の条件として，伝送遅延の変動特性が急激かつ大きく変化する通信状況を想定した．通信状態安定しており伝送遅延変動が小さい場合は動的最適化による UV 制御特性の改善効果は小さい．この場合，ジッタバッファによる最大バッファリング時間 D の値を遠隔走行制御中に頻繁に調整する必要はない．伝送遅延の変動特性が不安定である場合, 動的最適化によって UV 制御特性が改善する. 本章で考案した仕組みは, 大きな伝送遅延が頻繁に生じるような通信ネットワークにおける UV の遠隔自律走行制御の実現に資すると考えられる．

制御信号バッファリング時間の動的最適化は経路追従走行だけでなく CS を活用した他の種類の遠隔制御アプリケーションにも有用である可能性がある．しかし，制御精度や安定性に関する要求条件はアプリケーション毎に異なる．アプリケーションの種類や規模に応じた動的最適化機能の定量的評価は別途実施する必要がある．

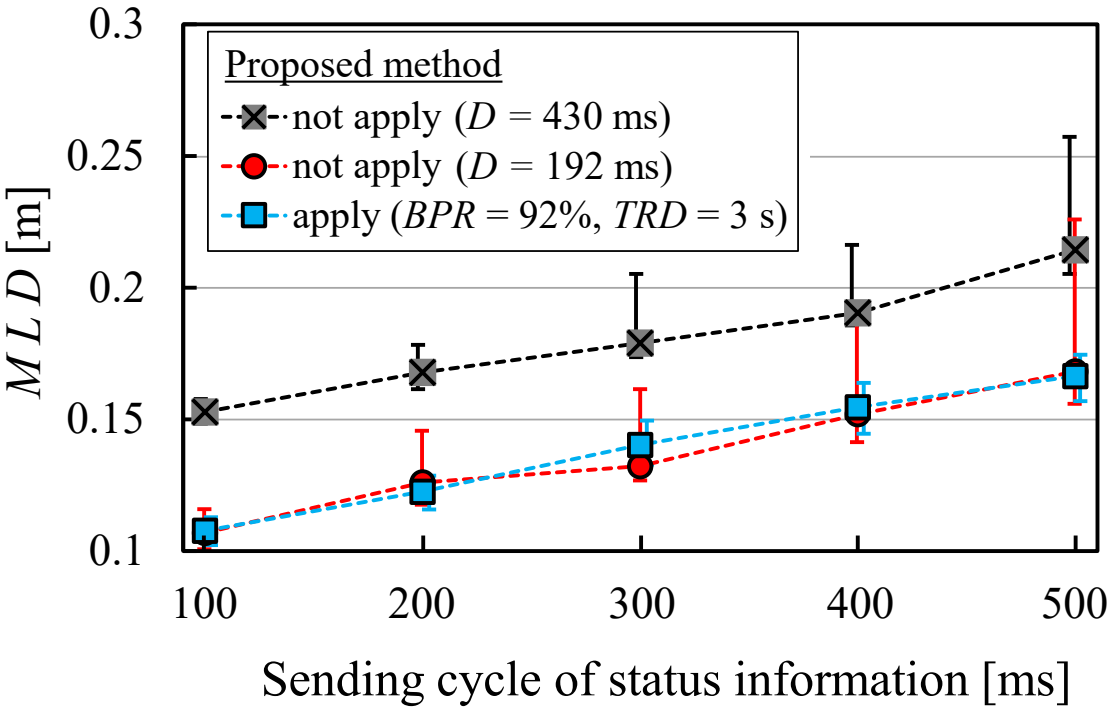


図 7-13　バッファリング時間の動的最適化の適用による *MLD* の変化

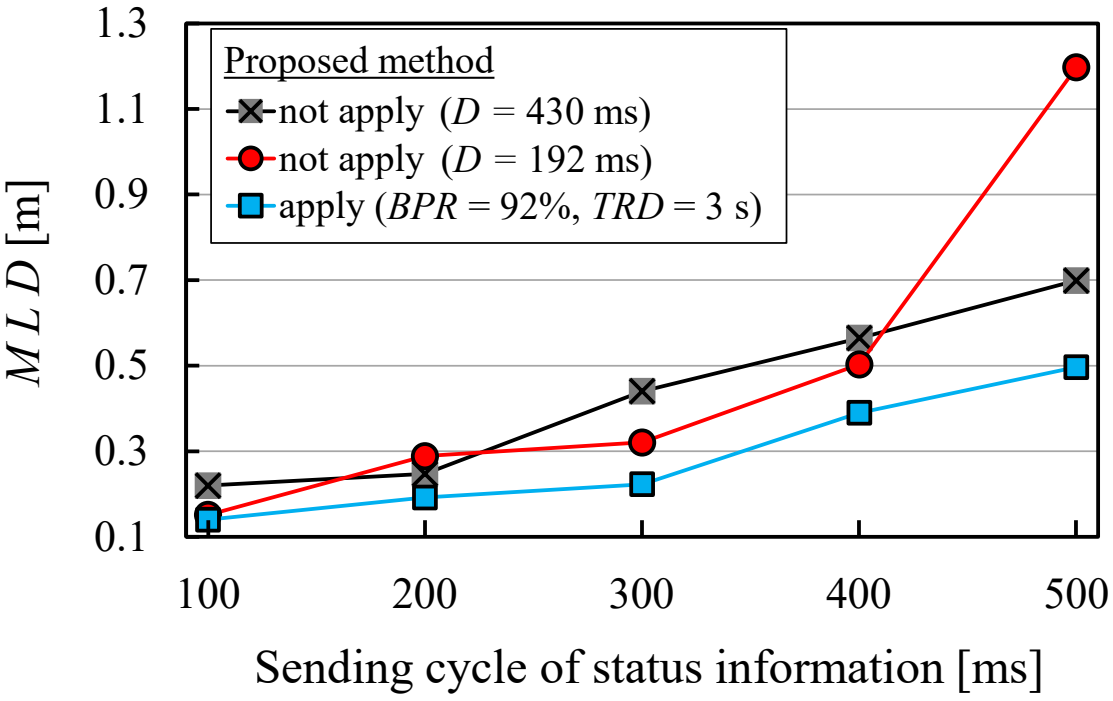


図 7-14　*MLD* の最大値

7.6　結言

CS による遠隔自律制御の制御精度および安定性は通信ネットワークの品質に大きく影響を受ける．インターネットを介した通信では伝送遅延とそのジッタが発生することは避けられない．また，突発的に大きな伝送遅延が発生することを予測することは困難である．

ディジタルツインを適用した CS による UV の遠隔経路追従制御についての第 5 章の検討結果から，ジッタバッファを UV に適用することで伝送遅延変動の影響を低減して UV の遠隔制御特性を改善できることが明らかになった．ジッタバッファによる制御信号のバッファリング時間をリアルタイムな伝送遅延特性に応じて最適化することは UV 遠隔制御をさらに安定化させる上で重要な要素である．本章では過去数秒間の制御信号の伝送遅延の値を用いてバッファリング時間を動的に調整する仕組みを考案した．また，その機能を第 4 章で提案した UV 遠隔経路追従制御のシステムに実装するとともに，制御信号バッファリング時間の動的最適化の有効性を定量的に評価した．その動的最適化はパラメータ *BPR* と *TRD* に最適値を与えることで達成される．シミュレーション評価の結果，以下を確認した．

(1) *BPR* = 92%，*TRD* = 3 s の時，考案した動的最適化機能によって CS による UV 遠隔経路追従制御における制御信号バッファリング時間を最適化できる．

(2) バッファリング時間の動的最適化を適用することで，バッファリング時間を一定値とする場合よりも UV 遠隔制御の精度が大きく改善する．

(3) (2)の効果は UV のフィードバック周期が長い程大きい．

上記の結果は UV の走行速度を 1 m/s とした時の結果である．これは小型搬送車の巡航速度として妥当な速さである．1 m/s より遅い場合でも同じ結果が得られると考えられる．より大きな車両では，巡航速度はより速くなると考えられる．走行速度が 1 m/s よりもっと速い場合，制御システムには制御信号のバッファリング時間をより緻密に制御することが求められると予想される．本章で実装したシステムで運用可能な走行速度の上限については別途研究で評価する必要がある．

第 8 章　総括

走行車とその周辺環境を模擬するディジタルツインをクラウドサーバ（CS）に適用することで，CS の処理能力を活用した走行状態の高度な再現・予測が可能となる．また，これを活用した適切な制御方法を用いることで，無人走行車（UV）のリアルタイム遠隔協調制御を主機能とした利便性の高いアプリケーションを実現できる可能性がある．小型搬送車両のような UV を多数かつ広範囲にわたり遠隔制御する場合，CS による遠隔集中制御方式の方が自律分散制御方式より経済的・効率的であると考えられる．

本研究は CS による小型 UV の遠隔制御システムの実現化を目的としている．前提として，緊急停止機能を除く主要な走行制御機能を CS に集約することを想定している．物品配送のようなアプリケーションにおいては，指定経路を逸脱しないように UV を制御できる能力は必須である．一般的に，通信ネットワークを介した走行車遠隔制御では伝送遅延によって経路追従特性が劣化する．CS を制御器として活用する場合は伝送遅延とその変動が大きくなる危険性が高いため，信頼性が高く，かつ経済的な UV 遠隔制御システムを実現することが困難となる．この問題を解決するため，ディジタルツインを活用して伝送遅延の影響を低減し，UV 遠隔経路追従制御を安定化させる仕組みを考案した．本論では CS による UV 遠隔制御システムにこの仕組みを適用し，これを提案システムとした．また，提案システムの有効性を評価するために UV 経路追従特性についていくつかの検討を実施した．この際，実際の通信における伝送遅延を模擬し，遅延とその変動が大きい状態でも提案システムによる遠隔走行制御が有効であるか評価した．

8.1 節に第 5 章から第 7 章で解説した各検討の結果をまとめる．8.2 節に本研究の目的を達成する上での本論の意義と，今後検討する必要がある課題について述べる．本研究が通信ネットワークを活用した無人機の遠隔制御技術の発展に寄与することを祈念して，以上を本論文の総括とする．

8.1 各検討のまとめ

提案システムはCSとUVとの間のフィードバック制御を基として，UVの走行環境を模擬するディジタルツインと伝送遅延ジッタを吸収するジッタバッファが適用されたものである．CSはディジタルツイン上のUVモデルの状態を参照することで，安定した周期で制御信号の計算・送信を実行する．UVが送信するフィードバック情報は，ディジタルツインによる予測結果を修正するために活用される．ジッタバッファは制御信号の伝送遅延の変動を吸収することで，UVにおける制御周期の安定化およびCSによるUV走行状態の予測精度の向上をもたらす．

簡易的なディジタルツインとジッタバッファを用いてUV遠隔経路追従走行のシミュレータを作成し，CSによるUV制御特性の改善効果を定量的に評価した．その結果，ディジタルツインの適用によって，通信ネットワークによる制御信号の伝送遅延が大きい場合であっても目標経路を正確に追従できるようになることを確認した．また，ジッタバッファの適用によって伝送遅延ジッタが大きい場合の経路追従走行の精度が大きく向上することを確認した．

ディジタルツインを活用した遠隔制御では，実際の制御対象に対してディジタルツイン上のモデルが持つモデリング上の誤差が制御精度に影響する．この誤差がUVの経路追従走行に及ぼす影響を定量的に評価するため，走行速度またはステアリング角の動作についてUVとディジタルツインとの間で一定の誤差が生じる状態を模擬したシミュレーションを実施した．また，小型ラジコン車を用いて提案システムを試験的に実現化するとともに，ディジタルツインの誤差の影響を実際に評価した．それらの結果から，小型UVの走行制御においては，走行速度およびステアリング角についてUVとディジタルツイン上のUVモデルとの間の誤差を0.9から1.1の間の比率とするべきであることを確認した．また，実際のUVのステアリング角の方が小さい場合にUVの経路追従走行特性が劣化しやすいことを確認した．言い換えれば，ディジタルツインを用いた走行車遠隔制御ではステアリング角の動作をより緻密にモデリングする必要がある．

CSによる遠隔制御においては，常に安定した通信状態を維持できるとは限らないため，遠

隔制御中にパケット伝送特性が不安定になる可能性を考慮した制御システムが必要である．提案システムの場合，ジッタバッファによる制御信号バッファリング時間を実際の通信状況に適応させる仕組みが必要である．そのバッファリング時間を動的に調整する仕組みを考案するとともに，UV 遠隔経路追従走行においてバッファリング時間を最適化できるパラメータの組合せ（$BPR = 92\%, TRD = 3$ s）を特定した．また，通信ネットワークによる伝送遅延特性に応じてバッファリング時間を動的に最適化する場合と固定的なバッファリング時間の場合の UV 制御特性を比較して，動的最適化の有効性を評価した．シミュレーション評価の結果，考案した動的最適化機能によって UV の経路追従走行の精度が大きく改善することを確認した．

これらの検討は，実際の伝送遅延を模擬したケーススタディである．それぞれの検討から，再現した伝送遅延状況においてディジタルツインを活用した制御方法を適用することで，CS による UV 遠隔走行制御を安定化できることを確認できた．また，より安全な遠隔制御のため，制御対象の小型 UV をディジタルツイン上で再現する際に必要となるモデリング精度を明らかにした．以上の結果は，小型の自律搬送車両の実用的巡航速度において成立する．ディジタルツインとジッタバッファを活用する提案システムは小型 UV 以外の遠隔制御アプリケーションにおいても有効である可能性がある．システム上の各パラメータへの要求条件はアプリケーションの種類に応じて異なると考えらえる

8.2 研究目的に対する本論の学術的意義および今後の課題

ディジタルツインは既存のサービスの効率化または新しいサービスを実現する技術として近年注目されている．しかし，走行車のリアルタイム遠隔制御へ活用する研究事例は見られない．ディジタルツインコンピューティングを適用された CS による遠隔走行制御が可能となれば，潤沢な演算リソースによる集約的情報処理によって高度な自律配送サービスを提供できる可能性がある．具体的には，UV のリアルタイム制御だけでなく，効率的な車両整備計画，交通上のリスクを高度に模擬した経路探索等，複数の機能を統合的に運用できる可能性

がある．

本研究の目的は，CS による小型走行車の遠隔制御システムの実現化である．本論はそれに向けた基礎的段階の成果と言える．本論では 1 台の UV を遠隔制御するシステムを実装するとともに，実際の伝送遅延を再現したケーススタディによって提案システムの有効性を検証した．提案システムについてより実用的に研究を進める上では，実際の伝送遅延はより多様である点に留意する必要がある．目的とするシステムを実現化するためには，本研究の結果に基づき更に研究を進める必要がある．以下に今後必要と考えられる研究内容を列挙する．

(1) 実際の通信環境を用いて提案した UV 遠隔制御システムの制御特性を確認する．

(2) 複数の UV を協調制御するシステムを構成し，その有効性を評価する．

(3) UV とその走行空間を高度に再現したディジタルツインを構成する手法, および安全な UV 遠隔走行制御を行うために必要なセンシング情報の種類を追究する．

(4) ディジタルツインを構成・修正するために車両と走行環境に配備するカメラ・センサの最適な配備方法を考案・実証する．

謝辞

本論文は，筆者が令和 2 年 4 月から令和 5 年 3 月までの 3 年間，防衛大学校 理工学研究科 後期課程 電子情報工学系専攻に在学していた間に実施した研究成果をまとめたものです．在学中，指導教官として同校 電気情報学群 通信工学科の葉玉 寿弥 教授よりご指導ご鞭撻をいただきました．

本校の電子情報工学系専攻に入学する以前，筆者は神戸大学大学院 工学研究科 建築学専攻で博士前期課程まで修了していました．葉玉教授には，研究分野を変更して博士後期課程に進んだ筆者への指導を快諾していただいた上，研究活動全般にわたり終始懇切丁寧なご指導を賜りました．ここに深謝の意を表します．

同学科 中村 僚兵 准教授には，実験およびシミュレーション方法についてのご教授，および本論文執筆への適切なご助言を賜りました．心より感謝申し上げます．

在学期間中，学内での生活および研究活動のため身近な立場から様々なご支援，ご協力を頂きました同学科 後藤 啓次 教授，陸上自衛隊 福嶋 匡謙 1 等陸尉，小川 拳史 1 等陸尉，渡辺 大郎 2 等陸尉，皆川 昌樹 2 等陸尉，黒崎 将史 2 等陸尉，海上自衛隊 川口 大貴 2 等海尉，防衛大学校本科 65 期 坂元 央歩 学生，山本 あすか 学生，66 期 横溝 丈二 学生，鈴木 凱仁 学生，67 期 串間 亜華音 学生，寺田 浩輝 学生に心からお礼申し上げます．

最後に，3 年間研究活動に専念する機会を与えてくださいました防衛省 陸上自衛隊 各位，並びに防衛大学校 教務部 各位に心より感謝申し上げます．

参考文献

[1] 総務省, “令和 4 年版情報通信白書　第 2 部　第 4 章　第 6 節　国内外におけるサービス・アプリケーションの動向,” pp.80-88, July 2022.

[2] C.H. Hsu, S. Wang, Y. Zhang, and A. Kobusinska, “Mobile edge computing,” Wireless Communications and Mobile Computing, vol.2018, article ID 7291954, June 2018.

[3] M. Grieves and J. Vickers, “Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems,” Transdisciplinary perspectives on complex systems, F. J. Kahlen, S. Flumerfelt and A. Alves, pp.85-113, Springer, Berlin, Aug. 2016.

[4] M. Grieves, “PLM - Beyond lean manufacturing,” Manufacturing Engineering, vol.130, no.3, p.23, Mar. 2003.

[5] K. Y. H. Lim, P. Zheng, C. H. Chen and L. Huang, “A digital twin-enhanced system for engineering product family design and optimization,” Journal of Manufacturing Systems, vol.57, pp.82-93, Aug. 2020.

[6] A. M. Gutiérres, J. D. Gonzáles, R. F. Guillén, P. Verde, R. Álvarez and H. Perez, “Digital twin for automatic transportation in industry 4.0, ” Sensors, vol.21, no.10, pp.33-44, May 2021.

[7] C. Asavasirikulkij, C. Mathong, T. Sinthumonkolchai, R. Chancharoen and W. Asdornwised, “A study of digital twin and its communication protocol in factory automation cell,” 2021 International Conference on Emerging Technologies for Communication (ICETC 2021), D4-2, Dec. 2021.

[8] S. Mamella and H. Causegic, “A porshe digital twin: driven by data streaming & NoSQL,” Porsche AG, https://medium.com/next-level-german-engineering/a-porsche-digital-twin-driven-by-

data-streaming-nosql-d92083771ffd, Sep. 2021.

[9] J. Friedrich, “All BMW group vehicle plants to be digitalized using 3D laser scanning by early 2023,” BMW group, https://www.press.bmwgroup.com/global/article/detail/T0400833EN/all-bmw-group-vehicle-plants-to-be-digitalised-using-3d-laser-scanning-by-early-2023?language=en, June 2022.

[10] H. Nakashima, K. Hayashi and H. Gotoh, “Data-driven medical and health support created using bio-digital twin,” NTT Technical Review, vol.19, no.7, pp.17-22, July 2021.

[11] H. Hassani, X. Huang and S. MacFeely, “Impactful digital twin in the healthcare revolution,” Big Data and Cognitive Computing, vol.6, no.3, 83, Aug. 2022.

[12] C. Yamamoto, I. Shake, S. Fukada and S. Ueno, “Data-driven and optimized smart cities using urban DTC,” NTT Technical Review, vol.19, no.1, pp.45-51, Jan. 2021.

[13] L. Deren, Y. Wenbo and S. Zhenfeng, “Smart city based on digital twins,” Computational Urban Science, vol.1, no.4, pp.1-11, Mar. 2021.

[14] Government of Singapore, “Virtual singapore,” National Research Foundation, https://www.nrf.gov.sg/programmes/virtual-singapore, Feb. 2021.

[15] P. Isto, T. Heikkilä, A. Mämmelä, M. Uitto, T. Seppälä and J. M. Ahola, “5G based machine remote operation development utilizing digital twin,” Open Engineering, vol.10, no.1, pp.265-272, May 2020.

[16] H. Laaki, Y. Miche and K. Tammi, “Prototyping a ditital twin for real time remote control over mobile networks: application of remote surgery,” IEEE Access, vol.7, pp.20325-20336, Feb. 2019.

[17] 山中 直明, “自動運転におけるデジタルツイン活用,” ビヨンド 5G が描く未来, pp.148-150, 慶應義塾大学出版会株式会社, Tokyo, Jan. 2022.

[18] D. Bertsimas, P. Jaillet and S. Martin, “Online vehicle routing: the edge of optimization in large-scale applications,” Operations Research, vol.67, no.1, pp.143-162, Jan. 2019.

[19] Q. Chen, X. Song, H. Yamada and R. Shibasaki, “Learning deep representation from big and heterogeneous data for traffic accident inference,” Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16), pp.338-344, Phoenix, U.S.A, Feb. 2016.

[20] 高度情報通信ネットワーク社会推進戦略本部（IT 総合戦略本部）, “デジタル時代の新たな IT 政策大綱,” 首相官邸, https://warp.ndl.go.jp/info:ndljp/pid/12187388/www.kantei.go.jp/jp/singi/it2/kettei/pdf/20190607/siryou1.pdf, June 2019.

[21] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J.Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling and S. Thrun, “Towards fully autonomous driving: system and algorithms,” 2011 IEEE Intelligent Vehicles Symposium (IV), pp.163-168, Baden-Baden, Germany, June 2011.

[22] M. Mikami, K. Moto, K. Serizawa and H. Yoshino, “Field trial of dynamic mode switching for 5G new radio sidelink communications towards application to truck platooning,” IEICE Transactions on Communications, vol.E104-B, no.9, pp.1035-1045, Sep. 2021.

[23] S. Disha, J. Ashwini, G. Darshana and S. Poonam, “Wireless multifunctional robot for military applications,” International Research Journal of Engineering and Technology (IRJET), vol.5, no.3, pp.3932-3933, Dec. 2018.

[24] M. Burhanpurkar, M. Labbe, C. Guan, F. Michaud and J. Kelly, ”Cheap or robust? The

practical realization of self-driving wheelchair technology," 2017 International Conference on Rehabilitation Robotics (ICORR), pp.1079-1086, London, July 2017.

[25] Waymo LLC, "Introducing the Waymo driver to New York city's streets," https://blog.waymo.com/2021/11/introducing-waymo-driver-to-new-york.html, Nov. 2021.

[26] 山田 航也, "デリバリーロボットがオフィスの社員に直接配達　「東京ミッドタウン八重洲」が配達/清掃/運搬ロボットサービスの導入を発表," Robot start inc., https://robotstart.info/2022/04/22/office-building-delivery-robot.html, Apr. 2022.

[27] R. Marriswamy and H.R. Ravikumar, "Packet loss in wireless networks," International Journal of Engineering Research & Technology (IJERT), vol.3, no.1, pp.2881-2884, Jan. 2014.

[28] 都丸 敬介, "わかりやすい情報通信ネットワーク - 都丸敬介の情報通信講座," ソフトリサーチセンター, Tokyo, Feb. 2003.

[29] 大森 繁, 滝沢 賢一, "手術用ロボットシステムから見た無線通信技術への期待," 電子情報通信学会通信ソサイエティマガジン, vol.5, no.1, pp.31-35, Sep. 2011.

[30] 葉玉 寿弥, 後藤 和正, 皆川 昌樹, 長崎 兼大, 中村 僚兵, "通信網経由遠隔リアルタイム制御に対する伝送品質の影響度の実験的評価," 電子情報通信学会技術研究報告, vol.116, no.467, CS2016-95, pp.111-116, Feb. 2017.

[31] K. Sasaki, N. Suzuki, S. Makido and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," 55th Annual Conference of the Society of Instrument and Control Engineers (SICE), pp.1122-1127, Tsukuba, Japan, Sep. 2016.

[32] V. Balasubramanian, S. Otoum, M. Aloqaily, I. Al Ridhawi, and Y. Jararweh, "Low-

latency vehicular edge: A vehicular infrastructure model for 5G," Simulation Modelling Practice and Theory, vol.98, article.101968, pp.1-17, Jan. 2020.

[33] M. Iwabuchi, A. Benjebbour, Y. Kishiyama, G. Ren, C. Tang, T. Tian, L. Gu, Y. Cui and T. Takada, "5G experimental trials for ultra-reliable and low latency communications using new frame structure," IEICE Transactions on Communications, vol.E102-B, no.2, pp.381-390, Feb. 2019.

[34] D. Lee, M. Chung, W. Nam, K. Kim, W. A. Kim, J.J. Lee, S. Choi, H.K. Hong and E.S. Vandris, "Case study of scaled-up SKT 5G MEC reference architecture," Intel White paper - Communications Service Providers Muliti-Access Edge Computing, https://www.intel.co.jp/content/dam/www/public/us/en/documents/case-studies/sk-telecom-5g-mec-reference-architecture-paper.pdf, June 2020.

[35] Y. Jadeja and K. Modi, "Cloud computing - concepts, architecture and challenges," 2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET), pp.877-880, Nagercoil, India, Mar. 2012.

[36] 牧戸 知史, 佐々木 健吾, 中尾 彰宏, "5G 時代の自動車のコネクティビティ～リアルタイムサービスの実現に向けた期待～," 電子情報通信学会技術研究報告, vol.118, no.12, RCS2018-12, pp.59-63, Apr. 2018.

[37] S. Sindi and R. Woodman, "Autonomous goods vehicles for last-mile delivery: evaluation of impact and barriers," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1-6, Rhodes, Greece, Sep. 2020.

[38] 山田 寿夫, 中塚 正之, 甲藤 二郎, "センサーネットワークにおける SLAM を用いた位置推定実験," 電子情報通信学会技術研究報告, vol.107, no.525, IN2007-213, pp.325-330, Mar. 2008.

[39] 後村 胤樹, 大川 猛, 大津 金光, 馬場 敬信, 横田 隆史, “Visual SLAM ソフトウェア高速化検討のための処理時間分析,” 情報処理学会第 80 回全国大会, 7H-06, pp.1.125-1.126, Mar. 2018.

[40] L. Repele, R. Muradore, D. Quaglia and P. Fiorini, “Improving performance of networked control systems by using adaptive buffering,” IEEE Transactions on Industrial Electronics, vol.61, no.9, pp.4847-4856, Sep. 2014.

[41] Y. Sato, S. Kashihara and T. Ogishi, “Evaluation of latency effect in operability of remote driving,” IEICE Technical Report, vol.120, no.89, RCS2020-76, pp.105-110, July 2020.

[42] 3GPP TR 38.824, “Study on physical layer enhancements for NR ultra-reliable and low latency case (URLLC),” Aug. 2019.

[43] KDDI, “Japan’s first domestic 5G-enabled multi-vehicle autonomous driving experiment to be implemented on public roads,” https://news.kddi.com/kddi/corporate/english/newsrelease/2019/02/05/3650.html, Feb. 2019.

[44] Kyocera, “ローカル 5G を活用した無人化施工の実証実験を開始”, https://www.kyocera.co.jp/newsroom/news/2022/001949.html, Aug. 2022.

[45] M. M. Falatah and O. A. Batarfi, “Cloud scalability considerations,” International Journal of Computer Science & Engineering Survey (IJCSES), vol.5, no.4, pp.37-47, Aug. 2014.

[46] S. Xiao, C. Liu, Kenli Li and Keqin Li, “System delay optimization for mobile edge computing,” Future Generation Computer Systems, vol. 109, pp.17-28, Aug. 2020.

[47] J. Bogojeska, I. Giurgiu, G. Stark and D. Wiesmann, “IBM predictive analytics reduces server downtime,” INFORMS Journal on Applied Analytics, vol.51, no.1, pp.63-75, Feb. 2021.

[48].K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, “Layered vehicle control system coordinated between multiple edge servers,” R&D Review of Toyota CRDL, vol.49, no.1, pp.49-57, Jan. 2018.

[49]佐々木 健吾, 牧戸 知史, 中尾 彰宏, “協調運転実現のための多層エッジサーバによる車両制御システムインターネット遅延計測と計測結果を用いた車両制御評価,” 電子情報通信学会技術研究報告, vol.117, no.459, NS2017-234, pp.375-380, Mar. 2018.

[50]リュウ ケイコウ, 岸山 祥久, 佐々木 健吾, 中尾彰宏, “5G と MEC を活用した協調運転システムの提案,” 電子情報通信学会技術研究報告, vol.119, no.92, NS2019-41, pp.35-40, Jan. 2019.

[51]K. Goto, K. Okuda, R. Nakamura, and H. Hadama, “Experimental evaluation of a path switching function to avoid delay spikes in wireless LANs,” The 19th Asia-Pacific Network Operations and Management Symposium (APNOMS 2017), pp.267-270, Seoul, Korea, Sep. 2017.

[52]背戸 一登, 渡辺 亨, “フィードバック制御とは,” フィードバック制御の基礎と応用, pp.1-8, コロナ社, 東京都, Feb. 2013.

[53]東 剛人, 藤田 政之, “コンピュータネットワークにおけるむだ時間補償,“ 計測と制御, vol.45, no.10, pp.893-898, Oct. 2006.

[54]O.J. M. Smith, “Closer control of loops with dead time,” Chemical Engineering Progress, vol.53, no.5, pp.217-219, May 1957.

[55]足立 修一, 丸田 一郎, “カルマンフィルタの基礎,” 東京電機大学出版局, Tokyo, Oct. 2012.

[56] B. Pyakillya and S. Kladiev, “Identification of chemical reactor plant’s mathematical model,” MATEC Web of Conferences, vol.37, no.9, pp.1-3, Tomsk, Russia, Dec. 2015.

[57] J.M. Maciejowski, “Predictive control with constraints,” International Journal of Adaptive Control and Signal Processing, vol.17, no.3, pp.261-262, Apr. 2003.

[58] M. Velasco-Villa, B. del-Muro-Cuellar and A. Alvarez-Aguirre, “Smith-predictor compensator for a delayed omnidirectional mobile robot,” 2007 Mediterranean Conference on Control and Automation, T30-027, Athens, Greece, July 2007.

[59] S.Yasuda and H.Yoshida, “Prediction of round trip delay for wireless networks by a two-state model,”2018 IEEE Wireless Communications and Networking Conference (WCNC), pp.1-5, Barcelona, Spain, Apr. 2018.

[60] H. Yoshida, T. Kumagai and K. Satoda, “Dynamic state-predictive control for a remote control system with large delay fluctuation,”2018 IEEE International Conference on Consumer Electronics (ICCE), pp.1-6, Las Vegas, U.S.A, Jan. 2018.

[61] T. Ueno and T. Azuma, “A design of state predictive LQI controllers for a networked control system,” 2009 ICCAS-SICE, pp.2192-2195, Fukuoka, Japan, Aug. 2009.

[62] Y. Nagai, T. Watanabe, T. Sato, R. Nakamura and H. Hadama, “An evaluation of a state-predictive controller and a jitter buffer for remote controlled autonomous vehicles via the Internet,” The 10th International Conference on ICT Convergence (ICTC 2019), pp.444-449, Jeju lsland, Korea, Oct. 2019.

[63] L. Repele, R. Muradore, D. Quaglia, and P. Fiorini, “Improving performance of networked control systems by using adaptive buffering,” IEEE Transactions on Industrial Electronics, vol.61, no.9, pp.4847-4856, Sep. 2014.

[64] D. Piromalis and A. Kantaros, “Digital twins in the automotive industry: the road toward physical-digital convergence,” Applied System Innovation, vol.5, no.4, 65, July 2022.

[65] F. Akbarian, E. Fitzgerald and M. Kihl, “Synchronization in digital twins for industrial control systems,” 16th Swedish National Computer Networking Workshop (SNCNW 2020), pp. 1-4, June 2020.

[66] P. Riedel, M. Riesner, K. Wendt and U. Aßmann, “Data-driven digital twins in surgery utilizing augmented reality and cachine learning,” 2022 IEEE International Conference on Communications Workshops (ICC Workshops), pp.580-585, Seoul, Korea, May 2022.

[67] P. Isto, T. Heikkilä, A. Mämmelä, M. Uitto, T. Seppälä and J. M. Ahola, “5G based machine remote operation development utilizing digital twin,” Open Engineering, vol.10, no.1, pp.265-272, May 2020.

[68] T. Kaarlela, S. Pieskä and T. Pitkäaho, “Digital twin and virtual reality for safety training,” 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), pp.000115-000120, Mariehamn, Finland, Sep. 2020.

[69] K. H. Ang, G. Chong and Y. Li, “PID control system analysis, design, and technology,” IEEE Transactions on Control Systems Technology, vol.13, no.4, pp.559-576, July 2005.

[70] 佐田 達典, 江上 翔悟, 村山 盛行, “RTK-GPS による移動体測位の特性に関する基礎的研究,” 土木情報利用技術論文集, vol.17, no.4, pp.195-202, Nov. 2008.

[71] K. Ingemann, A. Nuchter, J. Hertzberg and H. Surmann, “About the control of high speed mobile indoor robots,” Proceedings of the 2nd European Conference on Mobile Robotics (ECMR ’05), ISBN 88-89177-187, pp.218-223, Ancona, Italy, Sep. 2005.

[72] 安田 真也, 吉田 裕志, “遠隔制御の応答性を改善するための制御周期の動的調整

法,” 電子情報通信学会技術研究報告, vol.117, no.460, IN2017-105, pp.93-98, Mar. 2018.

[73] K. Fujimoto, S. Ata and M. Murata, “Statistical analysis of packet delays in the Internet and its application to playout control for streaming applications,” IEICE Transactions on Communications, vol.E84-B, no.6, pp.1504-1512, June 2001.

[74] 3GPP TS 23.501 Version 15.3.0, “System architecture for the 5G system; Stage 2,” Sep. 2018.

[75] G. Huston, “Measuring the impact of DNS Flag Day 2020,” APNIC, https://blog.apnic.net/2020/12/14/measuring-the-impact-of-dns-flag-day-2020/, Dec. 2020.

[76] ITU-T Rec. G.191, “Software tools for speech and audio coding standardization,” Jan. 2019.

[77] TTC 標準 JJ-201.01 (第 5 版), “IP 電話の通話品質評価法,” p.43, Aug. 2008.

[78] R. Murai, T. Sakai, H. Kawano, S. Kinoshita, H. Uematsu and Y. Kitano, “Recognition of 3D dynamic environments and obstacle avoidance for in-hospital autonomous delivery robot,” Panasonic Technical Journal, vol.58, no.4, pp.51–57, Jan. 2013.

[79] Panasonic Group, “Panasonic autonomous delivery robots - HOSPI - aid hospital operations at changi general hospital,” https://news.panasonic.com/global/topics/4923, July 2015.

付録（CS プログラムのソースコード）

参考資料として，本論の各検討で活用した CS プログラムのソースコードを添付する．

```
// ------ [ Original class ] ------
import subsrc.Input;
import subsrc.Sen;
import subsrc.Rec;
import subsrc.Pos_Cal;
import subsrc.checkFilename;
import subsrc.UV_DTM_status;       //    for UV-status in digital twin
import subsrc.Targets;       //    set target-coordinates of path tracking, and set current target

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketAddress;
import java.net.SocketException;
import java.net.SocketTimeoutException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.Timer;
import java.util.TimerTask;



public class CS {
      // ------ [ Initialize (reset some of them by program to overwrite parameters for UV remote control) ] -------
         final static double pai = Math.PI;
         public static int sen_interval = 10;
         public static double v = 1.0;       //    cruising speed of UV [m/s]
         static double basedirect_x = 0;       //    UV-direction is based Y-axis ( right: + ,    left: - )
         static double basedirect_y = 1;       //      -> represent direction of Y-axis by vector components (direct_x, y)
         public static double accel = 1;       //    accel [m/s2] (not used in this simulation)
         public static double WB = 0.8;
         public static double tg2uv_j = 0.2;       //    condition(1) : judge whether “UV has reached to Target WP”
         public static double overLine_j = pai/2;       //    condition(2) : “US passed through the WP”
         static double alfamax = Math.PI / 4 ;
         public static double theta_max;
         static double max_in_asin;
         // ---------------------------------------------------------------------------------------------------------

         //------ [save name of calculation data] ------
```

```
static PrintWriter pw;
static String folder = "../result_UV/" ;
static String name1 = "#outputCS";
static String name_ext = ".csv";

// ------ [for stop remote control] ------
static boolean start = false;       //    true : start remote-control ,    false : stop
static int goal = 0;       //    goal = 1 (when UV reaches final target WP)

// ------ [overwrite during remote control] ------
static double[] UV_pos_fb = new double[3];       //    UV-status (x, y, fai)
//static double UV_v = 1;       //    (not used in this simulation)
static int UV_rseqNum = 0;       //    UV-traveling-time by feedback info.
static boolean new_fb = false;       //    recognize receiving new feedback from UV
static double[] cont_sig = new double[2];       //    control-signal [v,theta]
static long nowtime;
static int travelTime = 0;       //    UV-traveling-time in prediction by digital twin

// ------ [for communication] ------
static DatagramSocket recSocket;       //    socket for UDP in CS
static DatagramSocket cont_senSocket;
static int cont_recPort = 0;       //    rewrite whether NE is used or not
static int packetlength = 4 + 8*3 + 4;       //    length of UDP for control signal
static int JB_fb = 0;       //    request value D from UV (max-buffering-time of receiving signal)
public static int NEdown_nothere = 1;       //    Network Emulator for downstream = on(1) / off(0)
public static int NEup_nothere = 1;       //    for upstream
public static int debug = 0;       //    bug-check-mode = 0(invalid), 1, 2, ...


public static void main(String[] args) throws Exception {
        new CS();
        //    When repeating simulation many times, run CS() to loop in another class
}

public CS() throws Exception {
        // ------ [Step1 : initialize before control] ------
        new Input();       //    IPaddress, port-number, etc...
        new Input_UV("CS");       //    overwrite variables for remote control

        switch(NEdown_nothere) {
        case 0:       //    not use NE for reproducing transmission delay
                cont_recPort = Input.UV_recPort[0];
                break;
        case 1:
                cont_recPort = Input.NE_down_recPort;
                break;
        }
```

```
Targets target_UV1 = new Targets("../settingData/Targets.csv");       //    set target-WP & traveling mode
target_UV1.newTarget();       //    if using multiple UVs, set another instance

//     ( save remote control process )
checkFilename cf = new checkFilename();
String outputCS = cf.check(folder, name1, name_ext);
        if(debug != 0){
                FileWriter fw;
                fw = new FileWriter(outputCS,true);
                pw = new PrintWriter(new BufferedWriter(fw));
                pw.printf("CS-seq" + "," + "signal-used" + "," + "x" + "," + "y" + "," + "fai for Y" + "," + "signal-v" + "," +
                        "signal-theta" + "," + "UV_rseqNum" + "," + "r_x" + "," + "r_y");
                pw.println();
        }
        //pw.close();


// ------- [Step2 : check whether the other systems stand by or not] ------
int c;
if (debug > 0){
        System.out.println("    Check other programs.    Ready? [Enter]");
        c = System.in.read();
        System.in.skip(256);
} else {
        c = 3;
        while(c > 0){
                System.out.print(c+".");
                c-- ;
                Thread.sleep(1000);       //    countdown
        }
        System.out.println();
}

cont_senSocket = new DatagramSocket(Input.CS_cont_senPort);
recSocket = new DatagramSocket(Input.CS_recPort);
recSocket.setSoTimeout(3000);       //    timeout of Socket

//     (communication test / setting UV first position & first D-value)
System.out.println( "----- Step : test of delay ( CS --> UV ) -----" );
packetSen_test();       //    continue sending packet until receiving UV-reply ( UV-pos & request of D)
System.out.println("    -> Max signal-buffering-time in UV = " + JB_fb);
Thread.sleep(1000);       //    breaktime


// ------ [Step3 : main process for remote control] ------
System.out.println( "----- Step : Remote Control -----" );
Timer timer1 = new Timer();
TimerTask task1 = new TimerTask(){
```

```
//    In this inner class, simulate vehicles-motion every a certain time
//    if you want to control more vehicles, make new instance of class<UV_DTM_status>
UV_DTM_status UV1 = new UV_DTM_status(WB, UV_pos_fb);
//    calculate a certain vehicle motion & save signal(v,theta) to DTM-Log
Pos_Cal PosCal1 = new Pos_Cal();
//    temporarily keep the calculated UV_pos (x, y, fai)
double[] pos = new double[3];

// (counter)
final int Log_interval = 10;        //    save control-signal-data every 10 ms
int seq = -sen_interval;        //    it is run-time of CS-prediction
int JB_travel = JB_fb ;
final int first_JB = JB_fb ;        //    adjust time-scale in CS to UV-traveling-time

// (previous info.)
double[] pre_cont_sig = new double[2];        //    previous-cont_sig[v,theta]
int pre_travelTime = - Log_interval;        //    update when travelTime is over it
int ins_time;

// (to re-calculate UV_DTM_status)
int log_row_fx;        //    0 ~ log_length-1
int log_row_new;
final int log_length = UV1.signal_log.length;        //    array-length for control signals sent to UV1

// (for debug)
long time_debug = 0;


public void run(){
        if(start){
                //    [1.    update counter]
                seq += sen_interval;
                travelTime = seq + JB_travel - first_JB ;

                //    [2.    check travelTime is over the previous travelTime]
                if (travelTime > pre_travelTime){
                        nowtime = System.currentTimeMillis();
                        //    (1)    new prediction of UV_pos, using "previous" control-signal
                        pos = PosCal1.Go(UV1.UV_pos, pre_cont_sig[0] * (travelTime - pre_travelTime) / 1000,
                                        pre_cont_sig[1], UV1.WB);
                        //    (2)    new cont_sig(v,theta), using new UV_pos
                        makeSignal(target_UV1, pos, UV1.WB);        //    instanced-class(target, DTM)
                        UV1.UV_pos = pos;
                        //    (3)    send the signal
                        packetSen(seq, JB_travel);
                        //    (4)    save the signal into cont-sig-Log in DTM-class
                        ins_time = pre_travelTime + Log_interval;
                        while(true){
```

```
            if (ins_time >= travelTime){
                UV1.save2sigLog(ins_time/Log_interval, cont_sig);
                break ;
            }else{
                UV1.save2sigLog(ins_time/Log_interval, pre_cont_sig);
            }
            ins_time += Log_interval ;
        }

        // (if debug-mode, save the signal to CSV)
        if(debug != 0){
            pw.printf(seq + "," + travelTime + "," + pos[0] + "," + pos[1] + "," + pos[2] + "," +
                    cont_sig[0] + "," + cont_sig[1] + "," + UV_rseqNum + "," + UV_pos_fb[0] +
                    "," + UV_pos_fb[1]);
            pw.println();
        }
        //   (5)   update previous-time
        pre_travelTime = travelTime;
        pre_cont_sig = cont_sig;
    }

    //   [3.   Correct digitaltwin if CS got Feedback from UV ?]
    if (new_fb == true){
        new_fb = false;
        JB_travel = JB_fb;      // update D referring to UV's request
        // -- [start] --
        pos = UV_pos_fb;
        log_row_new = UV1.Log_row;      //   Latest signal in signal-Log
        log_row_fx = (UV_rseqNum / Log_interval) % log_length;
        // row in the Log corresponding to UV-real-pos

        if(debug > 1){
            System.out.println(" [now-DTM(seq)] " + String.format("%.3f",UV1.UV_pos[0]) + "," +
                    String.format("%.3f",UV1.UV_pos[1]) + "," +
                    String.format("%.3f",UV1.UV_pos[2]) + " (" + travelTime );
            System.out.println("   <UV-FB(seq)> " + String.format("%.3f",UV_pos_fb[0]) + "," +
                    String.format("%.3f",UV_pos_fb[1]) + "," +
                    String.format("%.3f",UV_pos_fb[2]) +" ("+ UV_rseqNum );
            System.out.print("      --> use signal : [" + log_row_fx +"]"+
                    UV1.signal_log[log_row_fx][0]+","+UV1.signal_log[log_row_fx][1]);
            time_debug = System.currentTimeMillis();   //start time
        }

        while(true){
            // DTM-signal-Log --> get v & theta
            pos = PosCal1.Go(pos, UV1.signal_log[log_row_fx][0] * Log_interval / 1000,
                    UV1.signal_log[log_row_fx][1], UV1.WB);
            log_row_fx ++ ;
```

```
                        if (log_row_fx == log_length){
                            log_row_fx = 0;
                        }
                        if (log_row_fx == log_row_new){
                            UV1.UV_pos = pos;
                            break;      //  finish, latest signal is used on step-2
                        }
                    }

                    if(debug > 1){
                        time_debug = System.currentTimeMillis() - time_debug; // end time
                        System.out.println("   -- [" + (log_row_fx - 1) +"]" + UV1.signal_log[log_row_fx - 1][0]
                                +","+UV1.signal_log[log_row_fx - 1][1]);
                        System.out.println(" [fixed-DTM   ] " + String.format("%.3f",UV1.UV_pos[0]) + "," +
                                String.format("%.3f",UV1.UV_pos[1]) + "," +
                                String.format("%.3f",UV1.UV_pos[2]) );
                        System.out.println("   ---     fixTime(ms):"+time_debug+"     ---");
                    }
                }       // end step-3

                //  [4.   finish the control when UV reached last target WP]
                if( goal == 1 || travelTime < 0 ){
                    if(debug != 0){
                        pw.close();
                    }
                    if (travelTime < 0){
                        System.out.println(" [Caution] Limit of int-value");
                    }else {
                        System.out.println(" --> DTM goal");
                    }
                    timer1.cancel();
                    timer1.purge();
                }
            }
        }
};


// (start)
start_stop();
// (activate TimerTask)
timer1.scheduleAtFixedRate(task1, 0, sen_interval);
// (activate packet receiver)
packetRec();

// ------ [close this program] ------
System.out.println(" --> packetRec Fin.");
recSocket.close();      //  close each socket
```

```
        cont_senSocket.close();
        System.out.println("-----    CS Fin.    -----");
}



//    method ; start or stop
static void start_stop(){
        if(start == false) {
                // reset
                cont_sig[0] = 0;
                goal = 0;
                new_fb = false;
                start = true;
                System.out.println("[Control start]");
        } else if (start == true) {
                start = false;
                System.out.println("[Control stop]");
        }
}


//    method ; calculation of signal[v,theta] for UV-traveling
static void makeSignal(Targets Tg, double[] pos, double WB_in){
        // argument (instance of class<Targets>,    position[x,y,fai],    wheelbase of vehicle)
        while(true){
                //    [1.    positional relation]
                double tg2uv = Math.sqrt( Math.pow( (Tg.now_tg[0] - pos[0]), 2 ) + Math.pow( (Tg.now_tg[1] - pos[1]), 2 ) );
                double x_wp2uv = pos[0] - Tg.now_tg[0];
                double y_wp2uv = pos[1] - Tg.now_tg[1];
                double L_wp2uv = Math.sqrt( Math.pow(x_wp2uv, 2) + Math.pow(y_wp2uv, 2) ) ;
                //    overLine    =    angle between two vector ([WP(i)->UV], [WP(i) -> WP(i-1)])
                double overLine = Math.acos( ( x_wp2uv * Tg.Vec_new2past[0] + y_wp2uv * Tg.Vec_new2past[1] )
                                                / ( L_wp2uv * Tg.Vec_new2past[2] ) );

                //    [2.    judge whether UV is in arrival-range ]
                if (tg2uv > tg2uv_j && overLine < overLine_j) {
                        //    (not reach WP area. So make signal to reach it)
                        //    alfa is angle between 2 point ( UV(x,y) -> targetWP(x,y) )
                        double alfa = Math.acos( (basedirect_x*(Tg.now_tg[0] - pos[0]) + basedirect_y*(Tg.now_tg[1]- pos[1]))
                                                / tg2uv);

                        //    (Judge alfa is right or left against UV-basedirect (right = +,    left = -) )
                        if (( basedirect_x*(Tg.now_tg[1] - pos[1]) - (Tg.now_tg[0] - pos[0])*basedirect_y ) > 0)
                                alfa = -alfa;
                        alfa = alfa - pos[2];        // it is angle ( alfa - UV-traveling-direction ),
                        if(alfa > pai){
```

```
                alfa -= 2*pai;      //  -180° < alfa < 180°
            }else if(alfa < -pai){
                alfa += 2*pai;
            }

            //  (get steering-angle and traveling-speed of vehicle)
            double in_asin;     //  value for UV-front-wheel-angle (rad)
            if(Tg.now_driveMode.equals("circle")){
                //  case : drive to target-pos by circular-route
                in_asin =  2*WB_in*Math.sin(alfa)/tg2uv ;
                if (in_asin > max_in_asin){
                    cont_sig[1] = theta_max;
                }else if (in_asin < -max_in_asin){
                    cont_sig[1] = -theta_max;
                }else{
                    cont_sig[1] = Math.asin( in_asin );
                }
            }else{
                //  case : drive to target-pos by line-route
                in_asin =  2*WB_in*Math.sin(alfa)/(tg2uv/2);
                if (alfa >= alfamax || in_asin > max_in_asin) {
                    cont_sig[1] = theta_max;
                }else if (alfa <= -alfamax || in_asin < -max_in_asin) {
                    cont_sig[1] = -theta_max;
                }else {
                    cont_sig[1] = Math.asin( in_asin );
                }
            }
            //  (in this simulation, speed is fixed every test)
            cont_sig[0] = v;
            if (goal == 1){
                cont_sig[0] = 0;
            }
            break;      //  finish this method

        } else {
            System.out.println("     > arrive(ms) : "+travelTime);
            if (Tg.i == Tg.tNum) {
                //  (Judging UV reached last WP, CS finishes the control process)
                start_stop();
                goal = 1;
            }
            Tg.newTarget();     //  shift target (x,y) to next one
        }
    }   //  end while loop  -->  run packetSen
}

//  method ; receiver for UV-feedback info.
```

```
static void packetRec(){
	byte[] getData = new byte[60];
	Rec Rec_cs = new Rec(recSocket, getData.length);

	while(goal == 0){
		try{
			try{
				getData = Rec_cs.Get();
			}catch(SocketTimeoutException xx){
				continue;	//	wait 3000ms which is set as timeout of socket
			}

			if(Rec_cs.opp_port == Input.UV_senPort || Rec.opp_port == Input.NE_up_senPort){
				UV_rseqNum = ByteBuffer.wrap( Arrays.copyOfRange(getData, 0, 4) ).getInt();

				//	(get feedback data for recalculation in predictive control)
				UV_pos_fb[0] = ByteBuffer.wrap( Arrays.copyOfRange(getData, 4, 12) ).getDouble();
				UV_pos_fb[1] = ByteBuffer.wrap( Arrays.copyOfRange(getData, 12, 20) ).getDouble();
				UV_pos_fb[2] = ByteBuffer.wrap( Arrays.copyOfRange(getData, 20, 28) ).getDouble();
				JB_fb = ByteBuffer.wrap( Arrays.copyOfRange(getData, 28, 32) ).getInt();

				//	(check)
				if (UV_rseqNum > travelTime){
					System.out.println(" [Caution!] UVfeedback["+ UV_rseqNum + "] exceeds Latest_DTM[" +
							travelTime + "].");
					continue;
				}
				if (UV_rseqNum > 0){
					new_fb = true;
				} else {
					//	UV_rseqNum == -1 for “test packet” before starting remote control
					System.out.println(" -> Get UV-Feedback before remote-control");
					break;
				}
			}
		}catch(IOException e){
			e.printStackTrace();
		}
	}
}


//	method ; make packet including control signal, and send it
static void packetSen(int sendseq, int JB_order){
	//	(send control-signal / test-signal before the remote control)
	ByteBuffer sbuf = ByteBuffer.allocate(packetlength);
	sbuf.putInt(sendseq);	// 4byte
	sbuf.putLong(nowtime);	// 8 byte
```

```
                sbuf.putDouble(cont_sig[0]);    // 8
                sbuf.putDouble(cont_sig[1]);    // 8
                sbuf.putInt(JB_order);    // 4

                byte[] cont_byteArray = sbuf.array();
                try{
                        new Sen(cont_senSocket, Input.UV_address[0], cont_recPort, cont_byteArray);
                }catch(IOException e){
                        e.printStackTrace();
                }
                cont_byteArray = null;
                sbuf = null;
        }


        //    method ; get UV-pos & D-request-value before remote-control
        //    (use once only)
        static void packetSen_test(){
                //    (for getting first UV-position and deciding value of JB_fb (D) )
                cont_sig[0] = 0;
                cont_sig[1] = 0;
                JB_fb = 0;
                test_send testSen = new test_send();
                testSen.start();
                packetRec();        //    break when CS gets reply-packet from UV
                testSen.interrupt();
                testSen = null;
        }
}



// -------- class to get UV-pos & D-request-value before remote-control ---------- //
//          (use once only)
class test_send extends Thread{
        //    (used in method "packetSen_test" in main process)
        this.isActive = false;

        //@Override
        public void run(){
                System.out.print("    -> ... ");

                while(true){
                        try{
                                CS.nowtime = System.currentTimeMillis();
                                try{
                                        CS.packetSen(-1, 0);        //    For test-packet, sendseq = -1
                                }catch(Exception e){
```

```
                        e.printStackTrace();
                    }
                    Thread.sleep(CS.sen_interval);
                }catch(InterruptedException ie){
                    //    this is called by thread.interrupt();
                    break;
                }
            }
        }


}
```

研究業績

[1] 佐藤 逸人, 森本 政之, 吉本 雄大, “残響音を付加した音源の物理特性とラウドネスの関係,” 日本音響学会建築音響研究会, Chiba, Mar. 2015.

[2] 吉本 雄大, 佐藤 逸人, “残響音を付加した音声の自己相関関数がラウドネスに与える影響,” 日本建築学会 2015 年度近畿支部研究発表会, Osaka, June 2015.

[3] 吉本 雄大, 佐藤 逸人, “残響音を付加した音声の物理特性とラウドネスの関係,” 日本音響学会 2015 年秋季研究発表会, Fukushima, Sep. 2015.

[4] 佐藤 逸人, 吉本 雄大, “残響音付加音声の時間変動特性とラウドネスの関係,” 日本音響学会建築音響研究会, Osaka, Oct. 2017.

[5] 吉本 雄大, 渡辺 大郎, 中村 僚兵, 葉玉 寿弥, “伝送遅延のある無人走行車遠隔制御における状態予測器の効果”, 2020 年電子情報通信学会ソサイエティ大会, B-11-27, pp.122, Sep. 2020.

[6] T. Watanabe, Y. Yoshimoto, R. Nakamura and H. Hadama, “A study on backup controller placed on a cloud server for remote-controlled unmanned vehicles,” 2020 International Conference on Emerging Tecnologies for Communications (ICETC 2020), E1-2, Dec. 2020.

[7] 吉本 雄大, 中村 僚兵, 葉玉 寿弥, “状態予測制御器を適用した車両遠隔制御において制御信号バッファリング時間が走行制御特性に及ぼす影響,” 電子情報通信学会技術研究報告, vol.121, no.14, CS2021-11, pp. 43-48, May 2021.

[8] 皆川 昌樹, 吉本 雄大, 中村 僚兵, 葉玉 寿弥, “無人走行車の状態予測制御の動的最適化機能の提案,” 2021 年電子情報通信学会ソサイエティ大会, B-11-11, pp.136, Sep. 2021.

[9] 吉本 雄大, 皆川 昌樹, 中村 僚兵, 葉玉 寿弥, “状態予測制御車両における最適なパケットバッファリング時間,” 2021 年電子情報通信学会ソサイエティ大会, B-11-12, pp.137, Sep. 2021.

[10] Y. Yoshimoto, M. Minagawa, R. Nakamura and H. Hadama, “Effect of buffering time on tracking control accuracy in remote vehicle control with digital twin computing,” 2021 International Conference on Emerging Tecnologies for Communications (ICETC 2021), P3-4, Dec. 2021.

[11] 皆川 昌樹, 吉本 雄大, 中村 僚兵, 葉玉 寿弥, “ディジタルツインを用いた遠隔走行車制御におけるフィードバック周期の影響の実験的評価,” 電子情報通信学会技術研究報告, vol.122, no.14, CS2022-2, pp.7-11, May 2022.

[12] Y. Yoshimoto, M. Minagawa, R. Nakamura and H. Hadama, “Buffering time optimization for path tracking accuracy in remote vehicle control with digital twin computing,” IEICE Communications Express, vol.11, no.8, pp.515-520, Aug. 2022.

[13] Y. Yoshimoto, T. Watanabe, R. Nakamura and H. Hadama, “Effectiveness of ditital twin computing on path tracking control of unmanned vehicle by cloud server,” IEICE Transactions on Communications, vol.E105-B, no.11, pp.1424-1433, Nov. 2022.

[14] M. Minagawa, Y. Yoshimoto, R. Nakamura and H. Hadama, “An Experimental study on modeling accuracy of digital twin for cloud-based remote vehicle path tracking control,” 2022 International Conference on Emerging Tecnologies for Communications (ICETC 2022), S5-6, Tokyo, Nov. 2022. (Received Best Short Paper Award)

[15] Y. Yoshimoto, M. Minagawa, R. Nakamura and H. Hadama, “An experimental study on digital twin computing for cloud-based remote path tracking control of unmanned vehicle,” IEICE Communications Express, vol.11, no.12, pp.877-882, Dec. 2022.

[16] Y. Yoshimoto, M. Minagawa, R. Nakamura and H. Hadama, "Adaptive buffering time optimiztion for path tracking control of unmanned vehicle by cloud server with digital twin," IEICE Transactions on Communications, vol.E106-B, no.7, pp.603-613, July 2023.