プレイヤパフォーマンス低下を抑制する オンラインゲーム向け遅延補償に関する一研究

本生 崇人　川崎 慈英[1]　藤橋 卓也[1]　猿渡 俊介[1]　渡辺 尚[1]

概要：有線ネットワークおよび無線ネットワークのブロードバンド化にともなって，ネットワークを介して複数のプレイヤと共通のオンラインゲームを協力プレイ・対戦プレイする需要が高まっている．一方で，複数プレイヤが共通のオンラインゲームを協力プレイ・対戦プレイする場合，パケット損失や輻輳などに起因するネットワーク遅延の増加が各プレイヤのパフォーマンスに悪影響をもたらす．特に，シューティングゲームに代表される非同期型ゲームでは，ネットワーク遅延に起因してプレイヤ端末間で発生するゲーム情報の差異がプレイヤによる攻撃行動・回避行動に影響を及ぼす．ネットワーク遅延に起因するプレイヤパフォーマンスへの悪影響を軽減することを目的として，本稿では各プレイヤ端末上で動作するオンラインシューティングゲーム向け遅延補償手法を提案する．提案手法では過去のゲーム情報を用いてプレイヤ端末間で生じるゲーム情報の差異を補償する．具体的には，過去のキャラクタ位置情報を元にした線形補間に基づく遅延補償手法，過去のゲーム画面を元にした深層強化学習に基づく遅延補償手法をそれぞれ提案してプレイヤ端末上で欠落した相手プレイヤの位置情報・操作情報を補償する．ネットワーク遅延，人間による操作を考慮した実験評価から，線形補間および深層強化学習を用いた提案手法それぞれがネットワーク遅延に起因するプレイヤパフォーマンスへの悪影響を低減できることを確認した．

1. はじめに

ブロードバンドネットワークの普及にともなって，ネットワークを介して複数のプレイヤが共通のゲームをリアルタイムで協力・対戦プレイするオンラインゲームへの需要が増加している．例えば，近年では e-sports として Tom Clancy's Rainbow Six Siege や スプラトゥーンなどのオンラインシューティングゲームの大会が数日・数ヶ月ごとに開催され，新たなエンターテイメント・プロスポーツとして世界中で脚光を浴びている．

一般的にオンラインゲームはネットワークを介してサーバに複数プレイヤが接続するクライアント・サーバ型と，サーバを介さずにプレイヤ同士が接続するピアツーピア型に分けられる [1]．クライアント・サーバ型のオンラインゲームでは，各プレイヤ端末から受信したゲーム情報を元にサーバが各種処理を実施するとともに，他プレイヤのゲーム情報をサーバから各プレイヤ端末に伝送する．例えば，クライアント・サーバ型のオンラインシューティングゲームでは，各プレイヤ端末から受信した自身のキャラクタ位置，キャラクタが発射した銃弾位置を元にしてサーバが命中判定を実施するとともに，他プレイヤのキャラクタ位置と銃弾の位置をサーバから各プレイヤ端末に伝送する．

このとき，サーバと各プレイヤ端末間のネットワークでは，パケット損失や輻輳などにともなって遅延がたびたび増大する [2]．ネットワーク遅延の増大はキャラクタ位置情報や銃弾の位置情報などを含んだゲーム情報をサーバや他プレイヤ端末に伝送する際の遅れにつながるため，自身が保持するゲーム情報とサーバ・他プレイヤ端末が保持する自身のゲーム情報との間で差異を招く．ネットワーク遅延に起因するゲーム情報の差異を軽減する一方法として，ネットワーク遅延が大きいプレイヤ端末からのゲーム情報を受信するまでゲームの進行を停止する方法が挙げられる．プレイヤのターンが順番に回ってくるゲームなど，ゲーム情報を端末間で同期するためにゲーム進行を一時停止できるゲーム (同期型ゲーム) では本手法を用いてゲーム情報の差異を補償することができる．しかしながら，シューティングゲームなど，リアルタイム性が高くゲーム情報の同期のためにゲーム進行を停止できないゲーム (非同期型ゲーム) では，サーバは各プレイヤのゲーム情報を同期しないまま，決まった周期で各種処理を実施する．このとき，サーバ上に残されたゲーム情報の差異はプレイヤが意図しない攻撃行動・回避行動の失敗を招く [3]．

クライアント・サーバ型のオンラインシューティングゲームにおいてネットワーク遅延に起因する影響を軽減す

[1] 大阪大学大学院情報科学研究科

るために，サーバ側でゲーム情報の差異を補償する方法が提案されている [4], [5]．本手法では，サーバが各プレイヤのキャラクタに対して銃弾が命中したか判定するとき，最新のキャラクタ位置情報ではなく，ネットワーク遅延量に相当する過去のキャラクタ位置を命中判定に利用する．サーバ側での遅延補償手法はサーバ上で生じるゲーム情報の差異を補償してプレイヤパフォーマンスの低下を抑制することができる．一方で，各プレイヤ端末が保持する相手プレイヤのゲーム情報と相手プレイヤの実際のゲーム情報との間には差異が含まれたままとなる．この差異は “shot behind corners” などの問題を招く．具体的には，実際の相手プレイヤはキャラクタを操作しているにも関わらず，ネットワーク遅延の影響でプレイヤ端末上ではキャラクタが停止しているかのように表示され，命中判定にも停止時のキャラクタ位置をサーバが参照する．このとき，双方のプレイヤは本来のパフォーマンスを発揮することはできない．

本稿では，クライアント・サーバ型のオンラインシューティングゲームにおいてネットワーク遅延に起因するプレイヤパフォーマンスへの影響を軽減するため，各プレイヤ端末上で動作する遅延補償手法を提案する．提案手法では過去のゲーム情報を利用することで，ネットワーク遅延によって欠落した相手プレイヤのゲーム情報を補償する．具体的には，過去に受信した相手プレイヤのキャラクタ位置情報を入力とした線形補間から現在のキャラクタ位置情報を推定する遅延補償手法，プレイヤ端末上に表示された過去のゲーム画面を入力とした深層強化学習から相手プレイヤによる次の操作を推定する遅延補償手法を提案する．各提案手法を用いて補償した相手プレイヤのゲーム情報をゲーム画面に反映するとともに，各プレイヤはゲーム画面上の相手プレイヤのキャラクタに対して攻撃することでネットワーク遅延によるパフォーマンスへの影響を低減する．

提案手法による効果を評価するために，本研究ではクライアント・サーバ型の対戦型シューティングゲームシミュレータを開発した．各プレイヤがランダムにキャラクタを操作する場合，各プレイヤが銃弾を避けるようにキャラクタを操作する場合を想定したシミュレーション評価から，線形補間を用いた遅延補償手法，深層強化学習に用いた遅延補償手法がプレイヤの操作方法に応じてパフォーマンス低下を抑制できることがわかった．また，ネットワーク遅延を与えた実験評価および人間による操作ログを用いた実験評価から，提案手法がネットワーク遅延によるパフォーマンスへの影響を軽減可能であることがわかった．

本稿の構成は以下の通りである．2 節では本研究で想定するオンラインシューティングゲームモデルおよび提案する遅延補償手法について述べる．3 節で各提案手法がオンラインシューティングゲームにおけるプレイヤパフォーマンス低下を抑制できるかシミュレーション評価を通して議論する．4 節ではネットワーク遅延を与えた実験評価およびユーザによる操作ログを用いた実験評価を通して提案手法の効果をさらに議論する．5 節では関連研究について述べ，6 節においてまとめとする．

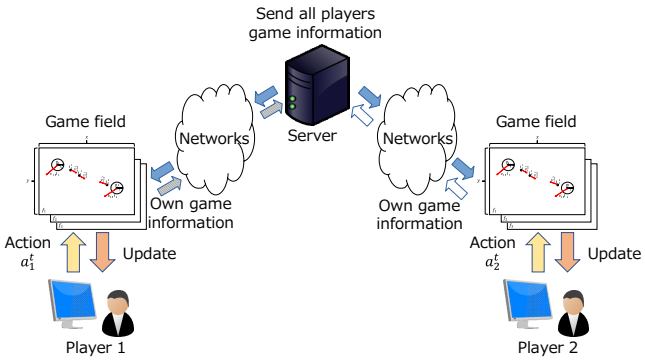
2. オンラインシューティングゲームを対象とした遅延補償手法

本稿では，クライアント・サーバ型のオンラインシューティングゲームを対象としてネットワーク遅延に起因するプレイヤパフォーマンスへの悪影響を軽減する 2 つの遅延補償手法を提案する．各提案手法では過去のゲーム情報から相手プレイヤのゲーム情報を推定することで，ネットワーク遅延下にある各プレイヤが本来のパフォーマンスを発揮できることを目指す．

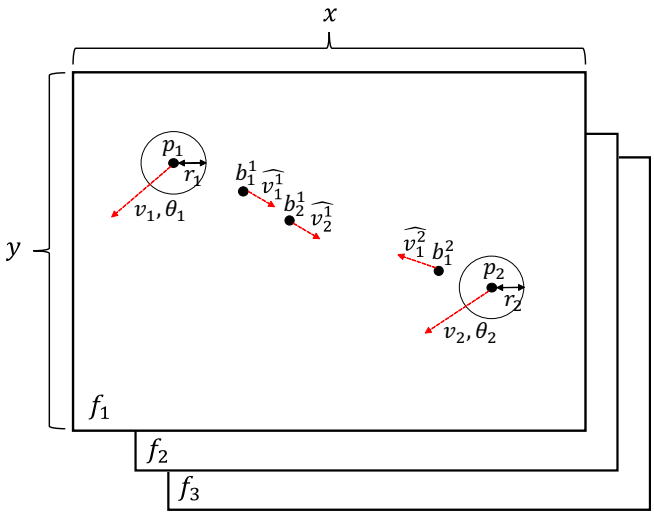
2.1 想定するオンラインシューティングゲームモデル

図 1 に本稿で想定するシューティングゲームモデルを示す．ゲームの種類はクライアント・サーバ型で非同期型の対戦型シューティングゲーム，プレイヤ数は 2 人である．図 1 (a) に各プレイヤ端末・サーバ間で情報を同期する流れの全体像を示す．各プレイヤ端末はゲームフィールド上の自身のキャラクタの位置情報と発射した銃弾の位置情報を 1 秒に 60 回サーバに送信する．サーバは 1 秒に 60 回，受信した各プレイヤ端末の位置情報と銃弾の位置情報を元にして各プレイヤに対する銃弾の命中を判定したあと，各プレイヤ端末に命中判定結果とすべてのプレイヤの位置情報を送信する．各プレイヤ端末は自身が保持する各プレイヤのキャラクタ位置にしたがってゲームフィールドを $1/f$ 秒ごとに更新して表示する．ここで，f (フレーム/秒) はゲームフィールドの更新頻度とする．各プレイヤ端末 i はプレイヤ端末上に表示された t 番目のゲームフィールド f_t の状況を元にして操作 a_i^t を入力してキャラクタを操作する．各プレイヤ端末・サーバ間のネットワークには上り方向・下り方向の遅延が発生する．各プレイヤ端末はサーバからデータを受信した際，現在の時間と受信したデータに含まれる送信時間との差を計算することで自身とサーバ間に発生している下り方向のネットワーク遅延を計測する．

図 1 (b) に本稿で想定するシューティングゲームフィールドを示す．ゲームフィールドは $x \times y$ 画素の四角形で各プレイヤ端末の画面上に表示される．プレイヤ i が操作するキャラクタは t 番目のゲームフィールド表示時に端末が保持する各キャラクタ位置 $\boldsymbol{p_i^t} = \{x_i^t, y_i^t\}$ を中心点とした半径 r_i の円でフィールド上に表示される．自身のキャラクタは t 番目のゲームフィールドを元にした操作 a_i^t にしたがって中心点 $\boldsymbol{p_i^t} = \{x_i^t, y_i^t\}$ から角度 θ_i^t，速度 v_i (画素/秒) でフィールド内を移動する．このとき，操作可能な角度の粒度は $2\pi/\psi$ で定まるものとする．なお，ψ は操作階



(a) サーバとプレイヤ端末間でやり取りする情報



(b) 想定するゲームフィールド

図 1: 想定するシューティングゲームモデル

調を意味する．相手プレイヤのキャラクタはサーバから受信した最新位置情報あるいは提案手法を元に推定した位置情報・相手プレイヤの操作にしたがってフィールド内を移動する．各キャラクタは相手プレイヤのキャラクタに向けて銃弾を発射する．プレイヤ i から発射された j 番目の銃弾は中心点 $\boldsymbol{b_j^i} = \{x_j^i, y_j^i\}$，半径 $\hat{r}_j$ の円でフィールド内に表示されるとともに，角度 $\hat{\theta}_j^i$，速度 $\hat{v}_j^i$ (画素/秒) にしたがってフィールド内を移動する．

2.2 過去のプレイヤ位置情報を用いた遅延補償手法

本手法では，各プレイヤ端末は自身が保持する相手プレイヤの過去のキャラクタ位置情報を用いて線形補間から相手プレイヤの現在のキャラクタ位置を推定する．具体的には，過去 M 個の相手プレイヤのキャラクタ位置情報を元にして回帰直線を算出する．回帰直線は python ライブラリである scikit-learn に含まれる LinearRegression クラスを用いて算出する．得られた回帰直線，プレイヤ端末上での相手プレイヤのキャラクタ位置情報，サーバからプレイヤ端末へのネットワーク遅延の大きさにしたがって，遅延時間経過後に相手プレイヤのキャラクタがいる位置を推定してプレイヤ端末上に表示する．なお，本方式では遅延時間の間，相手プレイヤのキャラクタが回帰直線にしたがってフィールドを移動し続けているものと仮定する．このとき，相手プレイヤのキャラクタは得られた回帰直線と，現在の位置から遅延時間後に移動可能な位置との交点にいると予想できる．相手キャラクタが 1 フレームあたりに移動できるピクセル数を v (画素/秒)，ネットワーク遅延によってプレイヤ端末上に表示されている相手プレイヤのキャラクタ位置が N フレーム前の位置であると仮定すると，相手プレイヤのキャラクタが N フレーム後にいる位置は，現在の相手プレイヤのキャラクタ位置を中心とする半径 Nv の円上に相当する．半径 Nv の円が得られた後，プレイヤ端末は回帰直線と半径 Nv の円との交点を相手プレイヤのキャラクタ位置と推定してゲームフィールド上に反映する．その後，各プレイヤ端末は推定した相手プレイヤのキャラクタ位置に照準を合わせて射撃する．このとき，推定した相手プレイヤのキャラクタ位置とサーバが保持する相手プレイヤのキャラクタ位置との間でずれが小さければ，プレイヤは相手プレイヤに対して銃弾を命中させることができる．

本手法では，ネットワーク遅延の大きさに応じて，相手キャラクタの位置情報を推定するパラメータ N を適切に定める必要がある．ネットワーク遅延に対して設定した N が小さすぎる，あるいは，大きすぎる場合，相手プレイヤのキャラクタ位置を正確に推定できず，命中率低下に起因するプレイヤのパフォーマンス低下を招く．一方で，ネットワーク遅延に対して設定した N 次第では，「偏差撃ち」と同じ現象がプレイヤの実力に関係なく発生してしまうため，プレイヤが持つ本来のパフォーマンス以上の命中率を招く可能性もある．偏差撃ちとは，オンラインシューティングゲームに長けたプレイヤが相手プレイヤのキャラクタに攻撃するとき，キャラクタの移動方向・移動速度・銃弾速度を元に，あえて相手プレイヤのキャラクタ位置から少しずれた位置に射撃して銃弾を命中させる技術である．

提案手法では，ネットワーク遅延に応じて，プレイヤが本来のパフォーマンスを発揮できる N を適切かつ適応的に設定することを考える．具体的には，ネットワーク遅延がない場合と同等のパフォーマンスが得られ，位置推定精度が高い N を算出する．ここで，サーバ上での時刻 t における相手プレイヤのキャラクタ位置を $\boldsymbol{p^t}$，ネットワーク遅延 l を体感した場合の時刻 t 時に N フレーム先を推定して得られた相手プレイヤのキャラクタ位置を $\boldsymbol{p}_{l,N}^t$ とする．また，ネットワーク遅延がないときに相手プレイヤのキャラクタに命中した総銃弾数を h，ネットワーク遅延 l を体感して N フレーム先を推定したとき，相手プレイヤのキャラクタに命中した総銃弾数を $h_{l,N}$ とする．ここで，提案手法ではネットワーク遅延 l における最適な N を式 (1) に基づいて定める．

表 1: ゲームフィールドのスナップショットを入力，相手プレイヤによる操作を出力とするネットワーク構造

```
Input((4, 84, 84))
Convolution(filters=32, kernel_size=8, stride=4)
Activation("ReLU")
Convolution(filters=64, kernel_size=4, stride=2)
Activation("ReLU")
Convolution(filters=64, kernel_size=3, stride=1)
Reshape((1, 1, 3136))
Activation("ReLU")
LSTM(out_size=512)
Linear(out_size=9)
```

$$\arg\min_{N}\left(\frac{f(l,N)-\min \boldsymbol{F}}{\max \boldsymbol{F}-\min \boldsymbol{F}}+\frac{g(l,N)-\min \boldsymbol{G}}{\max \boldsymbol{G}-\min \boldsymbol{G}}\right), \quad (1)$$

$$f(l,N)\in \boldsymbol{F},\ g(l,N)\in \boldsymbol{G}$$

$$f(l,N)=\frac{1}{|\boldsymbol{T}|}\sum_{t\in \boldsymbol{T}}||\boldsymbol{p}^t-\boldsymbol{p}^t_{l,N}||,$$

$$g(l,N)=\begin{cases} h-h_{l,N}, & h-h_{l,N}>0 \\ \emptyset, & \text{else} \end{cases}$$

第 1 項はネットワーク遅延 l 下においてパラメータ N で推定した相手プレイヤのキャラクタ位置とネットワーク遅延がない場合のキャラクタ位置とを比較したときの誤差量を示している．ここで，第 1 項が取りうる値の範囲は 0–1 である． なお，異なる N を設定したときの誤差量の集合を $\boldsymbol{F}$ としている．第 2 項は，同様に，ネットワーク遅延 l 下においてパラメータ N で相手プレイヤのキャラクタ位置を推定したときの銃弾命中数とネットワーク遅延がない場合の銃弾命中数とを比較したときの誤差量を示している．ここで，第 2 項が取りうる値の範囲も 0–1 である．同様に，異なる N を設定したときの誤差量の集合を $\boldsymbol{G}$ としている．提案手法では，ネットワーク遅延 l 下において，双方の指標の和が最小になる N を使用して相手プレイヤのキャラクタ位置を推定する．

2.3 過去のゲーム画面を用いた遅延補償手法

本手法では，Deep Q Network (DQN) [6] と Long Short-Term Memory (LSTM) を組み合わせて相手プレイヤによるキャラクタ操作を推定する．表 1 に提案手法のネットワーク構造を示す．本手法では，プレイヤ i による t 番目のゲームフィールドに対応する操作 a^t_i を推定するために，直近 4 フレーム分のゲームフィールド $\{f_{t-4},...,f_{t-1}\}$ のスナップショットを入力として与える．各スナップショットは 84×84 画素のグレースケール画像とした．入力したゲームフィールドのスナップショットに対して畳み込み層と活性化関数である ReLU (Rectified Linear Unit) を 3 回用いてゲームフィールド内・ゲームフィールド間の特徴を捉える．ReLU は負の値を 0 とする活性化関数 $f(x)=\max(0,x)$ である．Reshape 層では畳み込み層が出力した 3 次元テンソルを 1 次元テンソルに変換する．1 次元テンソルは LSTM 層を設けて長期にわたるゲームフィールド間の傾向から相手プレイヤの傾向を学習する．最後に Linear 層を用いて相手プレイヤによる操作 $\hat{a}^t_i$ を推定する．推定した相手プレイヤの操作にしたがってゲームフィールド f_t を生成してプレイヤ端末上に表示する．

相手プレイヤと同じ操作を推定できる学習済モデルを取得するため，報酬関数は下記の式にしたがうものとした．

$$l(\hat{a}^t_i,a^t_i)=\begin{cases} 1, & \hat{a}^t_i=a^t_i \\ 0, & \text{else} \end{cases}$$

具体的には，相手プレイヤの操作と提案手法が推定した操作が一致したら報酬を 1，一致しなければ報酬を 0 とする関数である．

ネットワーク遅延が大きくなるにつれて，提案手法は複数手先の相手プレイヤの操作を先読みする必要がある．しかしながら，従来の深層強化学習では，1 手先の相手プレイヤの操作を推定することを前提としているため，ネットワーク遅延の増大に対処できない．本研究では，相手プレイヤによる複数手先の操作を推定するために 2 種類の推定方法を提案する．

Single DRL Prediction: Single DRL Prediction では学習済モデルが一度推定した操作を相手プレイヤが実行し続けると仮定して相手プレイヤの操作を先読みする．例えば，t 番目および $t+1$ 番目のゲームフィールドに対する相手プレイヤの操作を推定することを考える．まず，$\{f_{t-4},...,f_{t-1}\}$ を入力として学習済モデルが t 番目のゲームフィールドに対する相手プレイヤの操作 $\hat{a}^t_i$ を推定する．推定した相手プレイヤの操作を元にして相手プレイヤのキャラクタ位置を更新したゲームフィールド画像 f_t を生成する．その後，$\hat{a}^{t+1}_i=\hat{a}^t_i$ であると仮定して，相手プレイヤのキャラクタ位置を更新した $t+1$ 番目のゲームフィールド f_{t+1} を生成する．Single DQN では提案手法による推定が 1 度だけで済むため，推定に要する計算時間を削減できる．一方で，推定した操作に含まれる誤差が後続のフレームに伝搬する．

Successive DRL Prediction: Successive DRL Prediction では学習済モデルによる推定を繰り返すことで相手プレイヤの操作を先読みする．同様に，t 番目および $t+1$ 番目のゲームフィールドに対する相手プレイヤの操作を先読みすることを考える．$\{f_{t-4},...,f_{t-1}\}$ を入力として学習済モデルが t 番目のゲームフィールドに対する相手プレイヤの操作を推定するとともに相手プレイヤのキャラクタ位置を更新したゲームフィールド画像 f_t を生成する．その後，過去のゲームフィールド画像 $\{f_{t-3},...,f_{t-1}\}$ および生成したゲームフィールド画像 f_t を入力として学習済みモデル

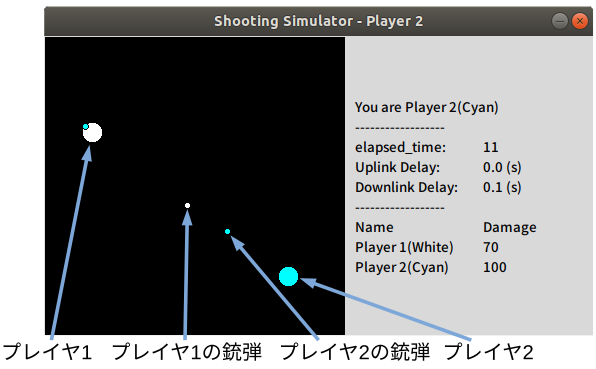


図 2: 開発した対戦型シューティングゲームシミュレータ

が $t+1$ 番目のゲームフィールドに対する相手プレイヤの操作 $\hat{a}_i^{t+1}$ を推定する．Successive DQN では，相手プレイヤによる操作の特徴を学習済モデルがうまく捉えている場合，相手プレイヤが操作を変更するタイミングを推定できると考えられる．

3. シミュレーション評価

3.1 評価環境

シミュレータ： ネットワーク遅延がプレイヤのパフォーマンスにもたらす影響を評価するために，図 2 に示すシューティングゲームシミュレータを開発した．ゲームの種類はクライアント・サーバ型で非同期型の対戦型シューティングゲーム，プレイヤ数は 2 人とした．対戦時間は特に指定しない場合 60 秒とした．フィールドは 300 × 300 画素の正方形とした．各プレイヤのキャラクタは 240 画素/秒でゲームフィールド内を移動し，各キャラクタが発射する銃弾は 2400 画素/秒でゲームフィールド内を進むものとした．

想定するネットワーク遅延： プレイヤ 1 端末はネットワーク遅延の影響を受けず，サーバとプレイヤ 2 端末間には N フレーム分に相当する下り遅延が生じているものと仮定した．このとき，プレイヤ 2 端末が提案手法を用いてプレイヤ 1 端末のゲーム情報を補償する．

キャラクタ操作方法： 各プレイヤによるキャラクタ操作方法として，キーボードとマウスを用いた人間による操作，事前に定義した移動モデルによる自動操作の 2 種類を可能とした．人間がキャラクタを操作する場合は，キーボードの w キー, s キー, a キー, d キーを用いてゲームフィールド内のキャラクタを上下左右斜めに移動できる．このとき，キャラクタの位置がゲームフィールドの端に到達するとゲームフィールドの反対側の端に移動する．相手プレイヤのキャラクタを攻撃するときはマウスカーソルを使用して射撃の照準を定め，自身のキャラクタの位置からマウスのカーソルがある方向に向かって左クリックで射撃する．

事前定義した移動モデルにしたがってキャラクタを自動操作する場合は，2 つの移動モデルを選択することができる．なお，両モデルにおける操作階調 ψ は 8 とした．移動モデル 1 は，各プレイヤのキャラクタが上，下，左，右に動く確率をそれぞれ 1/8，左上，右上，左下，右下に動く確率をそれぞれ 1/12，静止する確率を 1/6 として，各動作を平均 0.5 秒・標準偏差 0.125 秒のガウス分布にしたがって継続するものとした．移動モデル 2 は，相手プレイヤのキャラクタがいる方向に対して時計回りに 90 度回転した方向，すなわち相手の攻撃を避ける方向に移動する．このとき，各方向への移動は平均 0.17 秒・標準偏差 0.03 秒のガウス分布にしたがって継続するものとした．各プレイヤのキャラクタは相手プレイヤのキャラクタ位置を参照してその中心を狙って 0.1 秒に 1 回射撃する．

深層強化学習パラメータ： DQN および LSTM の実装には深層学習フレームワーク Chainer [7] を元にした深層強化学習ライブラリ ChainerRL[8] を利用した．ハイパーパラメータとして，Prioritized Experience Replay に利用するバッファサイズを 320,000，割引率は 0 とした．その他のパラメータは ChainerRL のデフォルトパラメータにしたがった．

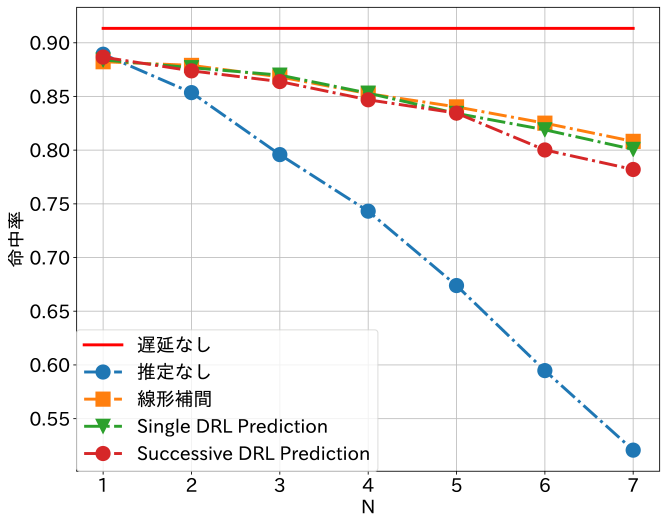
2 プレイヤが 60 秒間対戦するゲームを 1 エピソードとする．両プレイヤが移動モデル 1 にしたがってキャラクタを操作する 200 エピソード，両プレイヤが移動モデル 2 にしたがってキャラクタを操作する 300 エピソードをそれぞれトレーニングデータとして与え，各移動モデルに対する学習済モデルを生成した．

比較手法： 比較手法として遅延補償手法を利用しない場合，線形補間を用いた遅延補償手法，Single DRL Prediction を用いた遅延補償手法，Successive DRL Prediction を用いた遅延補償手法を用意した．

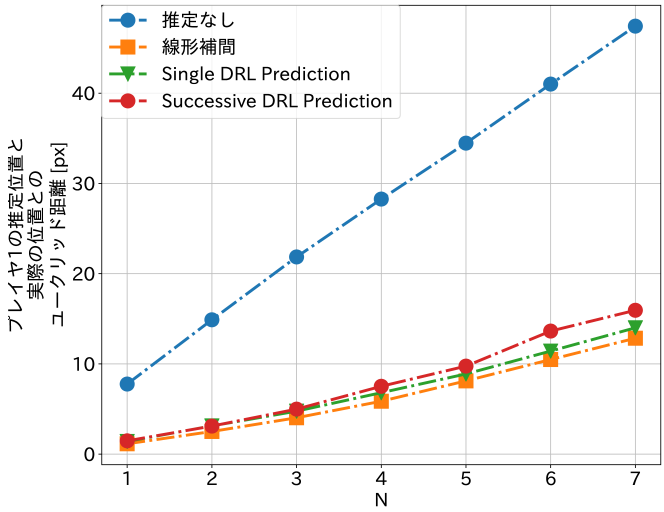
評価指標： 各提案手法による効果を示す指標として，対戦時間中にプレイヤ 2 のキャラクタが発射した全銃弾に対してプレイヤ 1 のキャラクタに命中した銃弾数の割合を示す命中率，プレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離を利用する．

3.2 評価結果

図 3 (a) および図 3 (b) に，サーバとプレイヤ 2 端末間に与えた下り遅延に対するプレイヤ 2 の命中率およびプレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離をそれぞれ示す．ここで，各プレイヤのキャラクタは移動モデル 1 にしたがって移動するものとした．同様に，図 4 (a) および図 4 (b) に，サーバとプレイヤ 2 端末間に与えた下り遅延に対するプレイヤ 2 の命中率およびプレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離をそれぞれ示す．図 4 (a) および図 4 (b) においては，各プレイヤのキャラクタが移動モデ

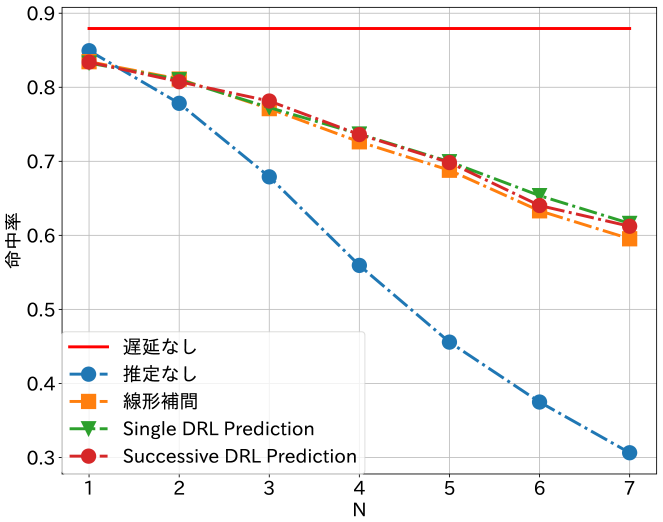


(a) 銃弾命中率

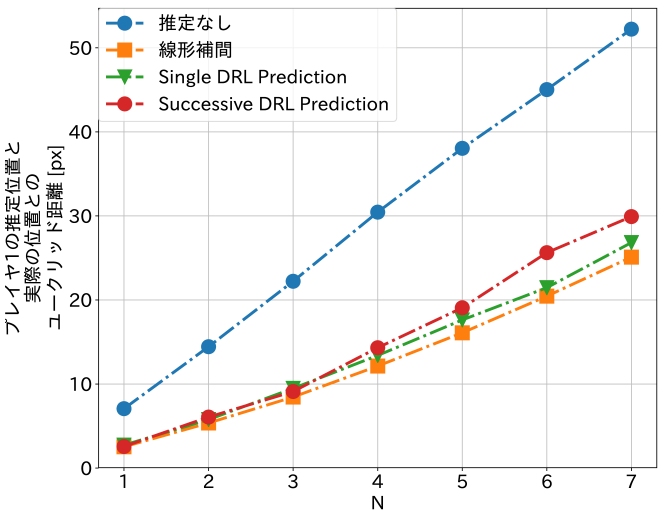


(b) 位置推定精度

図 3: 移動モデル 1 に対する各遅延補償手法の性能



(a) 銃弾命中率



(b) 位置推定精度

図 4: 移動モデル 2 に対する各遅延補償手法の性能

ル 2 にしたがって移動するものとした．なお，図 3 (a) および図 4 (a) における赤の実線は，ネットワーク遅延がない場合にプレイヤ 2 が達成した命中率である．

図 3 (a) から図 4 (b) を通して以下の 3 つのことが分かる．

- 相手プレイヤのキャラクタが移動モデル 1 にしたがって移動する場合，線形補間による遅延補償が命中率を高く維持できる．
- 相手プレイヤのキャラクタが移動モデル 2 にしたがって移動する場合，ネットワーク遅延が増大するにつれて Single DRL Prediction による遅延補償が高命中率を達成できる．
- 相手プレイヤのキャラクタが移動モデル 2 にしたがって移動する場合，低遅延時においては Successive DRL Prediction が命中率を高く維持できる．一方で，ネットワーク遅延が増大するにつれて命中率が低下している．推定を繰り返すごとに生成したゲームフィールド画像内に含まれる誤差が次の推定に伝搬していくことが 1 要因として考えられる．

4. 実験評価

4.1 実験環境

3 節では，相手プレイヤがキャラクタをランダムに操作する場合，攻撃を避けるように操作する場合，線形補間による遅延補償，深層強化学習による遅延補償がそれぞれ命中率低下を抑制できることを明らかにした．本節では，プレイヤ端末がネットワークを介して対戦する場合，人間がキャラクタを操作して対戦する場合における提案手法の効果を実験評価する．具体的には，3 節で開発した対戦型シューティングゲームシミュレータを用いて，下り遅延が発生したネットワークを経由してプレイヤ端末同士が対戦するとき，プレイヤがキーボードとマウスを使用してキャラクタを操作するとき，提案手法がどの程度命中率の低下および位置推定誤差を抑制できるか評価する．

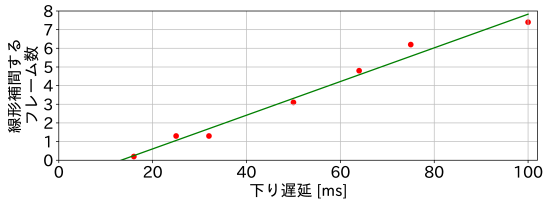


図 5: 移動モデル 1 における下り遅延と適切な N の関係

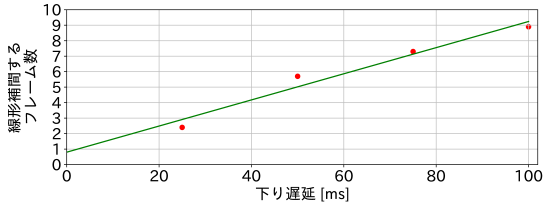
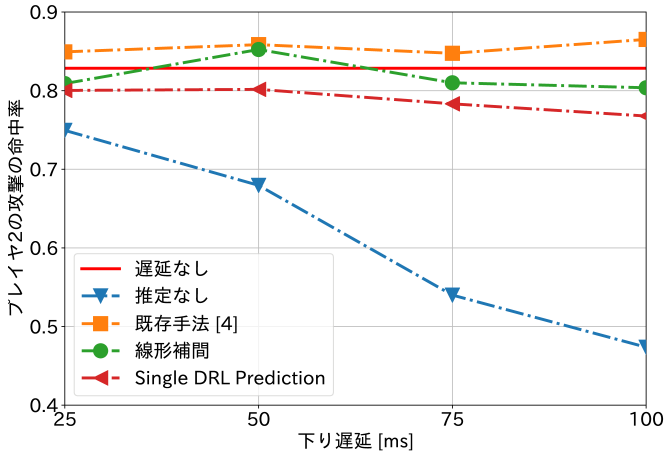
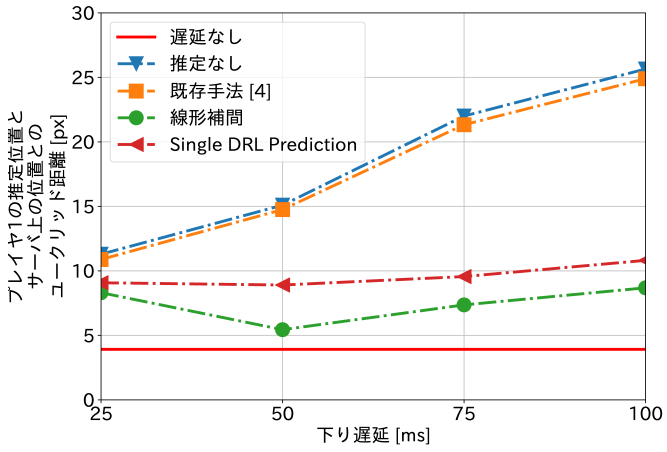


図 6: 移動モデル 2 における下り遅延と適切な N の関係



(a) 銃弾命中率

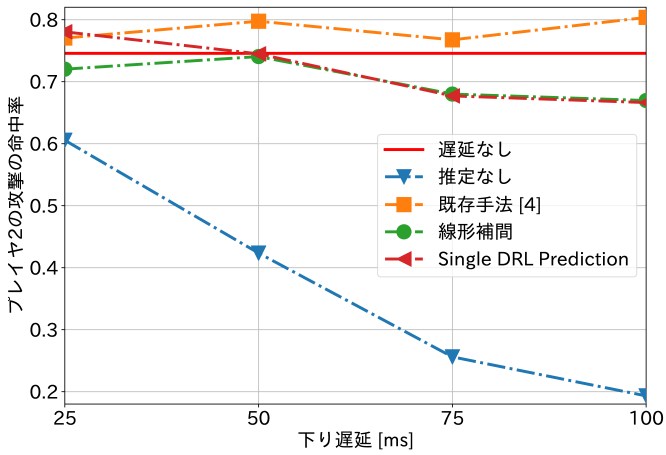


(b) 位置推定精度

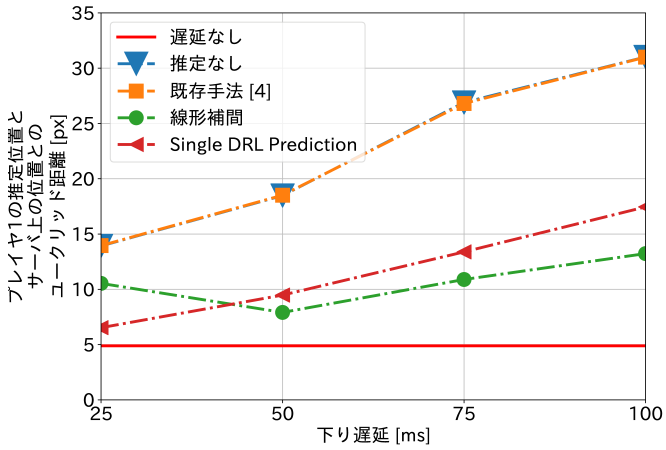
図 7: 各プレイヤのキャラクタが移動モデル 1 にしたがって移動するとき，下りネットワーク遅延に対する各遅延補償手法の性能

4.2 ネットワーク遅延を考慮した実験評価

本評価では，サーバに接続する 2 プレイヤ端末に異なるネットワーク遅延を与える．具体的には，プレイヤ 1 とサーバ間にはネットワーク遅延を与えず，プレイヤ 2 と



(a) 銃弾命中率



(b) 位置推定精度

図 8: 各プレイヤのキャラクタが移動モデル 2 にしたがって移動するとき，下りネットワーク遅延に対する各遅延補償手法の性能

サーバ間には下り方向に 0 ms から 100 ms までのネットワーク遅延を与えるものとした．プレイヤ 2 とサーバ間の下り方向の遅延が大きくなるにつれて，プレイヤ 2 が保持するプレイヤ 1 のゲーム情報とプレイヤ 1 の実際のゲーム情報とのずれが大きくなる．

比較手法としてプレイヤ 1 のゲーム情報を補償しない手法，サーバでゲーム情報を補償する既存手法 [4]，線形補間を用いた遅延補償手法，Single DRL Prediction による遅延補償手法を用いた．既存手法 [4] では，サーバでの命中判定時にプレイヤ 1 のキャラクタ位置情報をネットワーク遅延分遡って利用する．

本研究における遅延補償手法では，プレイヤ 2 端末が式 (1) を満たす N を用いて，プレイヤ 1 のキャラクタ位置を推定することで，ネットワーク遅延に起因するプレイヤ 2 のパフォーマンス低下を抑制する．プレイヤ 2 端末がネットワーク遅延を観測したとき，ネットワーク遅延に応じて最適な N を適応的に決定するために，ネットワーク遅延に対して式 (1) を満たす N を評価した．図 5 および

図 6 に，各プレイヤが移動モデル 1 および移動モデル 2 にしたがってキャラクタを操作するとき，サーバとプレイヤ 2 間の下り遅延に対する最適な N を示す．横軸は下り遅延の大きさ，縦軸はそれぞれの下り遅延における式 (1) を満たす N を示す．下り遅延が大きくなるにつれて，先読みする必要があるフレーム数 N が線形的に大きくなることが分かる．また，それぞれの下り遅延に対する最適な N の値を元にして，次式のとおり，下り遅延を変数とする 1 次関数から N を求めることができる．

$$N_{model1} = -1.2 + 0.090 \times l_{\mathrm{downlink}} \qquad (2)$$

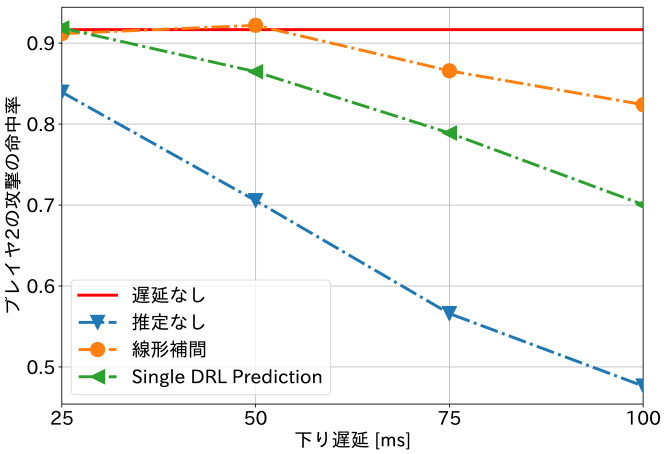
$$N_{model2} = 0.8 + 0.084 \times l_{\mathrm{downlink}} \qquad (3)$$

ここで，サーバとプレイヤ 2 間の下り遅延は l_{downlink} としている．各遅延補償手法では，プレイヤ 2 が式 (2), および式 (3) に基づいて決定した N を用いてプレイヤ 1 のキャラクタ位置を推定する．ただし，上式から得られた N が 0 を下回る場合には，下り遅延が十分に小さいと考えられるため，プレイヤ 2 はプレイヤ 1 の位置情報を推定しないものとした．

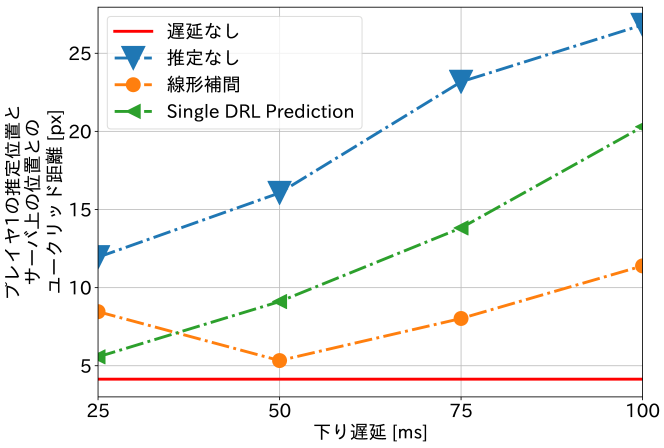
各手法によるプレイヤパフォーマンスへの影響を明らかにするため，サーバとプレイヤ 2 間に下り遅延を与えたときの命中率および位置推定精度を評価した．図 7 (a) および図 7 (b) に，各比較手法におけるサーバとプレイヤ 2 端末間に与えた下り遅延に対するプレイヤ 2 の命中率およびプレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離をそれぞれ示す．ここで，各プレイヤは移動モデル 1 にしたがってキャラクタを操作するものとした．同様に，図 8 (a) および図 8 (b) に，各比較手法におけるサーバとプレイヤ 2 端末間に与えた下り遅延に対するプレイヤ 2 の命中率およびプレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離をそれぞれ示す．ここで，各プレイヤは移動モデル 2 にしたがってキャラクタを操作するものとした．なお，図 7 (a) および図 8 (a) における赤の実線は，ネットワーク遅延がない場合のプレイヤ 2 の命中率，図 7 (a) および図 8 (a) における赤の実線は，ネットワーク遅延がない場合のプレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離である．

図 7 (a) から図 8 (b) に示した評価結果から，以下の 4 つのことがわかる．

- 下り遅延の大きさに関わらず，遅延補償手法はネットワーク遅延がない場合と同等の命中率と位置推定精度を得ることができる．
- サーバによる遅延補償手法は命中率の低減を抑制できる一方，プレイヤ 2 端末上に表示されるプレイヤ 1 のキャラクタ位置情報が実際のキャラクタ位置情報と比較して大きく外れる．
- 遅延補償手法を利用しない場合，下り遅延が増加するにしたがって命中率と位置推定精度がともに悪くなる．
- 各プレイヤが移動モデル 2 にしたがってキャラクタを操作する場合，Single DRL Prediction を用いた遅延補償手法は線形補間を用いた遅延補償手法と同等の性能を達成できる．



(a) 銃弾命中率



(b) 位置推定精度

図 9: 人間による操作に対する各遅延補償手法の性能

4.3 人間による操作を考慮した実験評価

本評価では，移動モデル 2 にしたがって行動する相手プレイヤのキャラクタと 1 プレイヤがキーボードとマウスを使用して 60 秒間対戦した操作ログを元にして，人間による操作に対して提案手法がどの程度パフォーマンス低下を抑制できるか評価した．なお，各プレイヤによる操作ログを取得するとき，各プレイヤとサーバ間のネットワーク遅延は与えないものとした．操作ログを元にして，移動モデル 2 にしたがってキャラクタを操作したプレイヤ端末 (以下，プレイヤ 2) に任意の下り遅延を与えたとき，キーボードとマウスを使用してキャラクタを操作したプレイヤ (以下，プレイヤ 1) のキャラクタ位置がサーバからどのよう

に得られるかシミュレートした．このとき，線形補間を用いた遅延補償手法あるいは Single DRL Prediction を用いた遅延補償手法をプレイヤ 2 端末上で用いてプレイヤ 1 が操作するキャラクタ位置を推定する．

図 9 (a) に，サーバとプレイヤ 2 端末との間の下り遅延に対する各遅延補償手法利用時の命中率を示す．また，図 9 (b) に，サーバとプレイヤ 2 端末との間の下り遅延に対する各遅延補償手法利用時のプレイヤ 1 のキャラクタ位置とプレイヤ 2 が推定したプレイヤ 1 のキャラクタ位置とのユークリッド距離を示す．線形補間を用いた遅延補償手法では下り遅延が 50ms のときまで命中率の低減を抑制できることがわかった．一方で，Single DRL Prediction を用いた遅延補償手法では下り遅延が 25ms のときまで命中率を維持できることがわかった．下り遅延が増大するにつれて深層強化学習で推定した行動と実際の行動との誤差が累積するためだと考えられる．ただし，プレイヤ 1 のキャラクタ位置を推定しない手法と比べると命中率が改善できていることから，提案手法は事前定義した移動モデルだけではなく人間による操作にも適用可能であることがわかった．

5. 関連研究

本研究はネットワーク遅延抑制技術，模倣学習に関する研究と関連する．

5.1 ネットワーク上での遅延抑制技術

ネットワークにおける遅延増大の要因には，データ送受信に要する処理遅延の増大，データ損失や輻輳に起因するネットワーク伝送遅延の増大が挙げられる．例えば，データ送受信時に要する処理遅延を削減する方法として Remote Direct Memory Access (RDMA) が設計されている．RDMA は高性能コンピューティング分野でノード間でメモリからメモリへ OS や CPU の介在なくネットワークを通してデータ転送できるため，データ転送にかかる CPU 負荷の関与が最小限に抑えられることから低遅延通信が可能となる．近年では，RDMA を Ethernet に適用した RDMA over Converged Ethernet (RoCE) [9] や既存のプロトコルスタックを利用可能とする RDMA for Proxying TCP connections (RoPT) が提案されている．また，輻輳やデータ損失に起因するネットワーク伝送遅延を抑制する手法として [10], [11] が提案されている．例えば，[10] においては，ネットワーク中で保持しているキャッシュデータを元にして高速にパケット再送を実現することでネットワーク伝送遅延の低下を図る．

上述したネットワーク遅延抑制技術を用いることでノード間のネットワークで生じる遅延量を低減することが可能となる．一方で，ノード間のネットワーク遅延をなくすことは伝搬遅延等に挙げられる要因から困難であるため，ネットワークを介して複数のプレイヤが対戦プレイ・協力プレイする非同期型のオンラインゲームにおいては，ネットワーク遅延に起因するプレイヤパフォーマンスへの影響を軽減する補償技術が必要である．本稿で提案する各遅延補償手法は過去のゲーム情報を元にしてプレイヤ端末間で生じるオンラインシューティングゲーム情報の差異を補償する．性能評価から線形補間および深層強化学習を用いた遅延補償手法がプレイヤ同士の対戦プレイにおいてネットワーク遅延を体感したプレイヤのパフォーマンス低下を抑制できることを明らかにした．

5.2 模倣学習

模倣学習とは教示者の行動と設定した報酬関数を元にして，報酬の期待値が最大化するように教示者の行動を模倣する技術である．模倣学習を実現する方法として大きく 2 通りの方法が提案されている．1 つ目は教師あり学習を用いて模倣対象の行動を学習する Behavioral Cloning である [12]．Behavioral Cloning では教示者の行動を教師データとして与えて学習したモデルにしたがって各状態から教示者の行動を模倣する次の行動を推定する．2 つ目は模倣対象の行動を入力として報酬関数を推定する逆強化学習 [13] と強化学習を組み合わせて教示者の行動を推定する方法である．本手法は教師データを必要としない一方で，逆強化学習で得られた報酬関数を用いた強化学習が必要となる．逆強化学習と強化学習の組み合わせによる学習時間の増大を抑制するために Generative Adversarial Imitation Learning (GAIL) [14] が提案されている．GAIL では，Generative Adversarial Network(GAN) を元にして模倣対象である教示者の行動から報酬関数を介せず教示者の行動と類似した行動を推定する．

本稿では，オンラインシューティングゲームにおける相手プレイヤの行動を模倣するために Behavioral Cloning に基づく深層強化学習を用いる．具体的には，相手プレイヤの行動と類似する行動を推定したときは高い報酬，異なる行動を推定したときは低い報酬を与えて相手プレイヤの行動を模倣するモデルを生成することでネットワーク遅延時における相手プレイヤのゲーム情報を補間する．教師あり学習を元にした Behavioral Cloning とは異なり，ゲーム画像を入力として相手プレイヤの行動を深層強化学習で学習することで，多様なゲーム局面における相手プレイヤの行動を推定できる．

6. おわりに

本稿では，オンラインシューティングゲームにおいて，ネットワーク遅延に起因するプレイヤのパフォーマンス低下を軽減する遅延補償手法を提案した．具体的には，相手プレイヤのキャラクタ位置を入力とする情報線形補間を用いた位置推定，ゲーム画面を入力とする深層強化学習を用いた行動推定を用いてネットワーク遅延によって生じた

端末間でのゲーム情報の差異を補償する．実験評価から，ネットワーク遅延がプレイヤのパフォーマンスに与える影響を提案手法によって軽減できること，画面上に表示される相手プレイヤのキャラクタ位置を提案手法を用いてより正確にできることがわかった．

謝辞

本研究は JSPS 科研費 (JP20K19783，JP19H01101，JP18H03231) および NTT アクセスサービスシステム研究所の支援の下で行った．

参考文献

[1] 中嶋謙互，オンラインゲームを支える技術—壮大なプレイ空間の舞台裏，技術評論社，2014.

[2] M. Claypool, and K. Claypool, “Latency and player actions in online games,” Communications of the ACM, vol.49, no.11, pp.40–45, 2006.

[3] 本生崇人，川崎慈英，藤橋卓也，猿渡俊介，渡辺尚，“ネットワーク遅延がもたらすオンラインゲームプレイヤへの影響に関する基礎評価，” 電子情報通信学会ソサイエティ大会，pp.1–1，2019.

[4] Y. Bernier, “Latency compensating methods in client/server in-game protocol design and optimization,” Proc. Game Developers Conference, 2001.

[5] S.W.K. Lee, and R.K.C. Chang, “Enhancing the experience of multiplayer shooter games via advanced lag compensation,” ACM Multimedia Systems Conference, pp.284–293, 2018.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., “Human-level control through deep reinforcement learning,” Nature, vol.518, no.7540, pp.529–533, 2015.

[7] S. Tokui, R. Okuta, T. Akiba, Y. Niitani, T. Ogawa, S. Saito, S. Suzuki, K. Uenishi, B. Vogel, and H. Yamazaki Vincent, “Chainer: A deep learning framework for accelerating the research cycle,” Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp.2002–2011, 2019.

[8] “ChainerRL,” `https://github.com/chainer/chainerrl`.

[9] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M.H. Yahia, and M. Zhang, “Congestion control for large-scale rdma deployments,” ACM SIGCOMM Computer Communication Review, vol.45, no.4, pp.523–536, 2015.

[10] J. Chen, S. Yan, Q. Ye, W. Quan, P.T. Do, W. Zhuang, X. Shen, X. Li, and J. Rao, “An sdn-based transmission protocol with in-path packet caching and retransmission,” IEEE International Conference on Communications, pp.1–6, 2019.

[11] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, and K. Lee, “Exll: An extremely low-latency congestion control for mobile cellular networks,” ACM International Conference on emerging Networking EXperiments and Technologies, pp.307–319, 2018.

[12] D.A. Pomerleau, “Efficient training of artificial neural networks for autonomous navigation,” Neural computation, vol.3, no.1, pp.88–97, 1991.

[13] A.Y. Ng, S.J. Russell, et al., “Algorithms for inverse reinforcement learning.,” Icml, vol.1, pp.663–670, 2000.

[14] J. Ho, and S. Ermon, “Generative adversarial imitation learning,” Advances in neural information processing systems, pp.4565–4573, 2016.