Vol.2021-ICS-201 No.4
2021/3/9

深層強化学習による自動運転車両の円滑な追い越しの実現

西 佑希[†1,a)]　横山 想一郎[†2]　山下 倫央[†2]　川村 秀憲[†2]　蕨野 貴之[†3]　大岸 智彦[†3]　田中 英明[†3]

概要：本研究では、追い越し場面での渋滞を緩和して、交通流量を増加させるための車両同士の協調行動を獲得するために、ソフトウェアシミュレーション上で深層強化学習を行い走行実験を行う．追い越し車の追い越し行動の獲得のために、動的パスプランニング手法である Frenet Optimal Trajectory を使用し、追い越しを実現した．対向車を強化学習対象として追い越し車を譲る行動を学習させることで、追い越し車とのインタラクションを実現した．ルールベースでの走行と深層強化学習後の走行とで、車両全体の走行距離を測定・比較し、交通流量の増加を確認した．

1. はじめに

道路交通において解決すべき問題の一つとして渋滞が挙げられる．大都市では約 40 %の一般道路で渋滞が発生していて [1]，渋滞による損失が年間 38.1 億時間，貨幣価値換算で 12 兆円 [2]，自動車から排出される CO2 は渋滞で 50 %増加 [3] など，渋滞によって大きな損失が発生している．そのため，渋滞を緩和して交通流量を増加させることが課題であるといえる．

渋滞が発生する場面としては，追い越し，交差点，合流が挙げられる．交差点における渋滞の緩和策としては，深層強化学習 [4] により信号を制御する方法 [5] が挙げられる．合流における渋滞の緩和策としては，ファスナー合流 [6] という手法があり，本線と合流車線が交互に合流することで，ある程度渋滞を緩和できることが確認されている．本研究では，渋滞の発生を抑制したり，緩和する方法がまだ確立されていない追い越しに注目する．追い越し場面に注目して追い越し車と対向車が協調して行動することで，交通流量を増加させることを目指す．

車両が交通流量増加のための追い越しを行うためには，追い越し車の速度や台数，対向車の速度や台数，追い越しの対象の種類（低速走行車または動かない障害物），追い越し車と障害物または前方車両との距離といったことを考慮する必要がある．このような状況において，ルールベースの意思決定ではルール数が膨大になることが考えられるため，本研究では交通流量増加のための追い越し行動を強化学習によって獲得することを目指す．

追い越し車が対向車線を利用して追い越しを行う場合には，対向車が減速や一旦停止をおこなって道を譲る・譲らないという意思決定も重要となる．対向車が道を譲る場合には，減速や一時停止を行う．交通ルールの観点からは対向車が常に道を譲る必要はないが，交通流量の観点からからは追い越し車に多くの後続車がいる場合には，対向車が道を譲った方が交通流量が上がる可能性がある．追い越しは交差点などの限定した場所ではなく，あらゆる場所で発生する可能性がある．そのため，追い越しの対象となる車両の速度や追い越しは道路の形状による見通しの良し悪しも，追い越し車や対向車の意思決定に影響する．

本研究では，このような複雑な状況が想定される追い越しが行われる状況に対して，簡約化した事例として直線道路上に動かない障害物が存在する状況を扱う．交通ルールに基づいて走行した場合の走行距離と強化学習で協調行動を獲得した場合の走行距離を比較して，交通流量に対する影響を検証する．

本稿の構成を以下に示す．2 章で関連研究について概説し，3 章では追い越しに使用するアルゴリズムを示す．4 章では本研究で使用するソフトウェアシミュレーション環境の構成について示し，5 章では深層強化学習の設定について示す．6 章では協調行動獲得のための強化学習の実装方法，実験条件および実験結果を説明し，実験結果に対する考察をおこなう．最後に，7 章で本稿をまとめる．

†1 現在，北海道大学工学部
Presently with Faculty of Engineering, Hokkaido University
†2 現在，北海道大学大学院情報科学院
Presently with Faculty of Information Science and Technology, Hokkaido University
†3 現在，KDDI 総合研究所
Presently with KDDI Research Inc.
a) ynishi@ist.hokudai.ac.jp

2. 関連研究

2.1 障害物回避

走行している車両が障害物を回避するための手法に関する研究はこれまでに数多く行われてきた．遺伝的アルゴリズムを使用することで経路計画を最適化問題として解決する手法 [7] や Double Deep Q Learning を未知の環境の動的経路計画に適用する手法 [8]，Ant Colony Optimization アルゴリズムを使用する手法 [9] などがある．本研究では，多数の車両が存在するようなシチュエーションでも利用でき，車両の追い越しができるような手法として，Frenet Optimal Trajectory[10] という手法を使用する．

2.2 ソフトウェアシミュレーション環境

オープンソースとして使えるソフトウェア交通シミュレータとしては Simulation of Urban MObility(SUMO)[11] や FLOW[12] といったものがある．ソフトウェアシミュレータの利点としては，導入・運用コストが非常に安価であること，様々な設定で実験を行えること，実時間に依らずシミュレーションできること，車両の数を数万台にするなど大規模なシミュレーションができることなどが挙げられる．欠点としては，外乱やハードウェア上の誤差などが取り込みづらく，実際の環境（実際の車両）と比較すると実現性が低くなってしまうことである．実際の車両を使用したシミュレーション環境を利用すれば，最も実際の環境に近い形でシミュレーションを再現できるが，コストが非常に高いことや多くの実験ができないことが欠点である．本研究では，追い越しのための動的パスプランニング手法の検討をするため，実時間によらず，かつ安全にシミュレーションできるソフトウェアシミュレーション上で実験を行う．ソフトウェアシミュレーションの種類としては，決まったレーン上を走行するレーン走行型シミュレータと，車両が自由に動ける 2 次元平面型シミュレータがある．本研究では，追い越し行動を扱うため，我々が開発した 2 次元平面型シミュレータ「Harmo Traffic Simulator」を使用する．Harmo Traffic Simulator ではステアリング角度も決定でき，追い越しの実現に向いている．

2.3 強化学習

強化学習は，何らかの行動をとるエージェントとその行動によって状態や報酬を返す環境で構成されている中で，エージェントがその報酬を大きくするような行動を獲得していく．強化学習手法は，モンテカルロ木探索や Q-Learning などのニューラルネットワークを使用しない手法から，ニューラルネットワークを使用する深層強化学習と呼ばれる DQN, A3C[13], AlphaGoZero[14] などがある．本研究では，状態に連続値，行動に離散値を用いることができ，Replay Memory にデータを貯めることで偏りを抑えた学習ができる価値ベースの手法 DQN を用いる．DQN には派生の手法や工夫として，Double DQN[15]，Dueling Network[16]，Prioritized Experience Replay[17] などがある．Double DQN は，行動の決定と評価をそれぞれ構造は同じだが別のネットワークを使用するという工夫である．Dueling DQN は，行動価値関数である Q 関数を状態のみに依存する Value 関数と状態と行動に依存する Advantage 関数にわけることで，ある状態のときに行動によって左右されない状態価値を学習することができる．Prioritized Experience Replay は，学習時に Replay buffer にためられた経験データから TD 誤差の大きい順に優先度をつけてデータをサンプリングすることである．これらの手法によって，DQN は安定した学習を進めることができる．

3. 動的パスプランニングアルゴリズム

本章では追い越しのための動的パスプランニングの説明をする．さらに，追い越しを実現するためにどのように動的パスプランニングを利用するかの説明をする．

3.1 Frenet Optimal Trajectory

自動車が車道を走行する場合，停止車や路肩にはみ出た雪山などの障害物になるようなものが進行したい車線を塞いでいて，障害物を避けて走らなければならない場合がある．そのような場合，本来通りたい車線から一旦外れて車線変更，車線またぎや対向車線を通って走行する必要がある．その際，ただ横にずれて障害物を避けるだけではなく，スピードを加減速しすぎない，障害物との距離が近すぎない，車線変更に時間をかけ過ぎないなどの一定の制約条件のもとで追い越さなければならない．そこで，本研究ではそれらを考慮した動的パスプランニングアルゴリズム Frenet Optimal Trajectory を使用した．4 章で後述する Harmo Traffic Simulator における Frenet Optimal Trajectory の実装に際しては文献 [18] を参考にしている．このアルゴリズムは渋滞時や高速走行時にも適用できる動的パスプランニングの手法で，他車のダイナミクスを考慮しなければならない場面でも速度と距離を考慮したパスプランニングができる．

Frenet Optimal Trajectory は，コスト関数として車の乗り心地に関係するジャーク（加速度の微分），目標パスからの距離，車線変更にかかる時間などを取り入れ，そのコストが最小となる経路を決定することで，自動運転車両の理想的な交通行動を実現する手法である．アルゴリズムの流れとしては，

(1) 経路候補を算出する
(2) 算出された経路の中から条件を満たすものを選ぶ
(3) 選ばれた経路の中から最もコストが低いものを最適経路とする

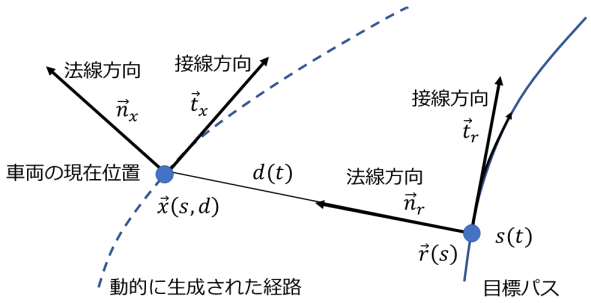

図 1 Frenet 機構による経路生成

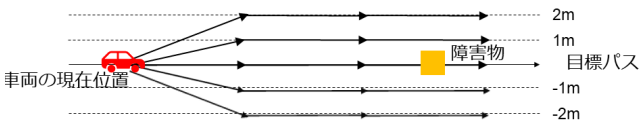

図 2 目標パスからの距離の変更による経路算出

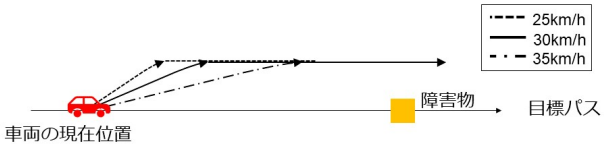

図 3 目標速度の変更による経路算出

という流れである．以下から具体的なアルゴリズムの説明をする．

3.1.1 様々な経路の算出

まずは，経路をいくつか算出する．経路算出の際，図 1 のように，目標となる経路に対して接線方向と法線方向にわけてそれぞれ 1 次元で考えている．これが Frenet Optimal Trajectory と呼ばれる所以である．式 1 で表されるように，目標パス上をどのくらい進んでいるかを接線方向の位置として考え，目標パスからどのくらい離れているかを法線方向の位置として考え，その時間変化を関数で近似することによって，目標パスからの相対位置によって経路を算出している．

$$\vec{x}(s(t),d(t)) = \vec{r}(s(t)) + d(t)\vec{n_r}(s(t)) \qquad (1)$$

式 1 中の $\vec{x}(s(t),d(t))$ は生成された経路の位置ベクトルであり，車両の接線方向の位置 $s(t)$ の位置ベクトル $\vec{r}(s(t))$ に法線方向のベクトル $d(t)\vec{n_r}(s(t))$ を足したものである．ここで，$d(t)$ は目標パスからの法線方向の距離係数で，$\vec{n_r}(s(t))$ は接線方向の位置 $s(t)$ での法線ベクトルである．ここで算出される経路のバリエーションは，図 2,3 のように目標パスからの距離を変更したり，目標速度を変更したり，目標速度に到達するまでの時間を変更することで変化する．

例えば，目標パスからの距離を-7m〜7m の範囲で 1m 間隔で変更すると，全部で 15 通りである．また，目標速度を 25km/h〜35km/h の範囲で 5km/h 間隔で変更すると，全部で 3 通りである．また，目標速度に到達するまでの時間を 4sec〜5sec の範囲で 0.2sec 間隔で変更すると全部で 6 通りである．これらの組み合わせから，$15 \times 3 \times 6 = 270$ 通りの経路ができることになる．このように，目標パスからの距離や目標速度などの組み合わせから複数の経路を算出する．

また，経路の算出は高速モードと低速モードで分かれている．車両の速度が極端に小さいときは上で述べたような方法で経路生成してしまうと曲率がかなり大きくなってしまう．目標パスの接線方向と法線方向で独立して経路生成しているため，速度が小さいと車両の非ホロノミック性が考慮されないからである．そこで，車両が停止する直前や停止状態から動き始める低速走行時には式 2 で経路生成を行う．

$$\vec{x}(s(t),d(t)) = \vec{r}(s(t)) + d(s(t))\vec{n_r}(s(t)) \qquad (2)$$

これにより，経路生成時に曲率が大きすぎる経路が生成されることがなくなる．また，ここではまだ障害物に衝突する経路かどうかは判定していない．

3.1.2 条件を満たす経路を選ぶ

次に，算出された経路の中から条件を満たすものだけを選ぶ．条件とは以下の 4 つである．

- 最大速度を超えない
- 最大加速度を超えない
- 最大曲率を超えない
- 障害物にぶつからない

最大速度，最大加速度，最大曲率はハイパーパラメータで定義され，それぞれの値を越してしまう経路は選ばれない．また，障害物は事前に場所がわかっているものとして，障害物の近傍を通ってぶつかってしまう経路は選ばれない．障害物と車両は，それぞれ長方形であると仮定して矩形同士の衝突判定を行っている．

3.1.3 最適経路の決定

最後に，3.1.2 小節の条件を満たした経路の中から，コストが一番低くなる経路を最適経路として決定する．Frenet Optimal Trajectory は経路生成時に，図 1 のとおり，法線方向と接線方向に分けて考えていると 3.2.1 小節で述べた．コストに関してもまた，同様に考えている．

まず，接線方向と法線方向のそれぞれについて，式 3 で表される，加速度の微分であるジャーク $J_t(p(t))$ を計算する．

$$J_t(p(t)) := \int_{t_0}^{t_1} \dddot{p}^2(\tau)d\tau \qquad (3)$$

時刻 t_0 での初期状態 $P_0 = [p_0,\ \dot{p_0},\ \ddot{p_0}]$ と $t_1 = t_0 + T$ での終点状態 $P_1 = [p_1,\ \dot{p_1},\ \ddot{p_1}]$ が与えられたとき，コスト関数にはジャーク $J_t(p(t))$，t_0 から $t_1 = t_0 + T$ までの時間，T 秒後の状態（位置，速度）の 3 つの項目を入れることで，コスト関数は式 4 のようになる．

$$C = k_j J_t + k_t g(T) + k_p h(p_1) \quad (4)$$

ここで，g と h は任意関数，k_j，k_t，$k_p > 0$ の重みである．よって，法線方向のコスト関数を式 5 のように表す．

$$C_d = k_j J_t(d(t)) + k_t T + k_d d_1^2 \quad (5)$$

式 5 中では，時刻 t における車両から目標パスまでの垂直距離を $d(t)$ で，目標とする「目標パスからの距離」にたどり着くまでの時間を T で，T 秒後の車両の位置から目標パスまでの距離を d_1 で表している．また，接線方向のコスト関数は式 6 で表される．

$$C_v = k_j J_t(s(t)) + k_t T + k_{\dot{s}}[\dot{s_1} - \dot{s_d}]^2 \quad (6)$$

式 6 中では，時刻 t におけるたどってきた目標パスの距離を t で，目標速度に達するまでの時間を T で，理想目標速度を s_d で，T 秒後の目標速度を s_1 で表している．例えば，理想目標速度を 30km/h として，目標速度を，理想目標速度と理想目標速度± 5km/h とすると，T 秒後に速度が 25km/h，30km/h，35km/h となる経路が生成されるので，理想目標速度 = 目標速度となる経路の方がコストが低いということになる．最終的なコスト関数は式 7 となる．

$$C = k_{lat} C_d + k_{lon} C_v \quad (7)$$

ここで，k_{lat}，k_{lon} はそれぞれ法線方向と接線方向の重みである．このコストが一番低くなる経路を最適経路として採用する．

3.2 Frenet Optimal Trajectory の利用

進行したい車線に障害物があるときに，その障害物を避けて追い越すような行動をとれるようにしたい．しかし，ただ単に目標パス上を通るだけだと障害物を避けられないので，3.1 節で紹介した障害物を避けるための経路を生成する動的パスプランニングを利用する．障害物がないときは目標パス上を通る経路を生成し，障害物があるときは障害物を避けてから目標パスに戻るような経路を生成し，その経路を追従することで追い越しを実現する．実際にどのようにソフトウェアシミュレーションに組み込んだのかは 4.2 節で説明する．

4. ソフトウェアシミュレーション環境

本章では，本研究の実験で使用するソフトウェアシミュレーションについて説明する．また，そのソフトウェアシミュレーションに動的パスプランニングをどのように適用したのかを説明する．

4.1 ソフトウェアシミュレーション環境の構成

図 4 に使用するソフトウェアシミュレーションの概要を示す．このソフトウェアシミュレーションを Harmo Traffic Simulator と呼ぶ．Harmo Traffic Simulator は，これまでに我々が構築した電動ラジオコントロールカー（RC カー）シミュレーションを模して構築したものである．処理のフローとしては，事前に用意されたマップ上に走行経路を生成し，その経路上を走るようなライントレース方式で走行制御を行う．生成される走行経路は障害物などがないときの理想的な走行経路となる．また，他車のデータは情報共有サーバを介して取得し，他車のデータを利用して実際の交通法規に従った行動をとる．

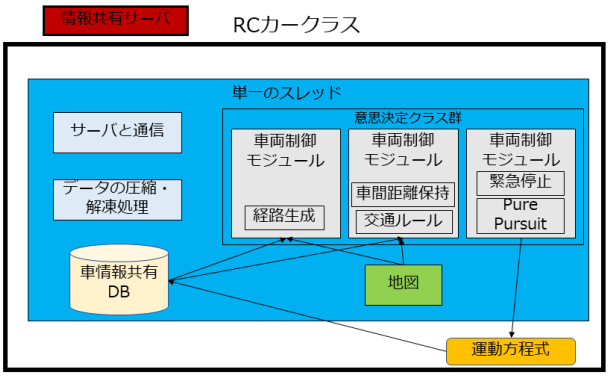

図 4 ソフトウェアシミュレーションのアーキテクチャ

車両の制御は図 4 の意思決定クラス群によって行われる．モジュール群は 3 つのモジュールから構成されており，目標経路を生成するモジュール，目標速度を決定するモジュール，ステアリング・アクセルを決定するモジュールがある．追い越しの実装のためには追い越しで生成した経路を追従することが必要であるので，以下で Harmo Traffic Simulator に実装されたステアリング，アクセル制御の手法について示す．

4.1.1 アクセル制御

アクセル制御には P 制御（比例制御）を使用している．P 制御は目標値と現在値との偏差に比例ゲインをかけた値を制御値とする制御方法である．車両のアクセル制御は，速度ではなく加速度を制御値として行う．そこで，加速度 a，比例ゲイン K_p，目標速度 v_{target}，現在速度 $v_{current}$ とすると制御地である加速度は以下の式で求められる．

$$a = K_p(v_{target} - v_{current}) \quad (8)$$

P 制御によって出された制御値（加速度）を利用してアクセル制御を行う．

4.1.2 ステアリング制御

ステアリング制御には Pure Pursuit を使用している．Pure Pursuit とは，車両の自己位置と目標パスから目標ステアリング角度を算出するアルゴリズムである．Pure Pursuit はそのアルゴリズムの性質から，急なカーブには対応しにくいが，設計パラメータが少なく直感的に理解しやすかったり，D 制御のように微分を必要としないのでノイズに強いという利点を持つ．

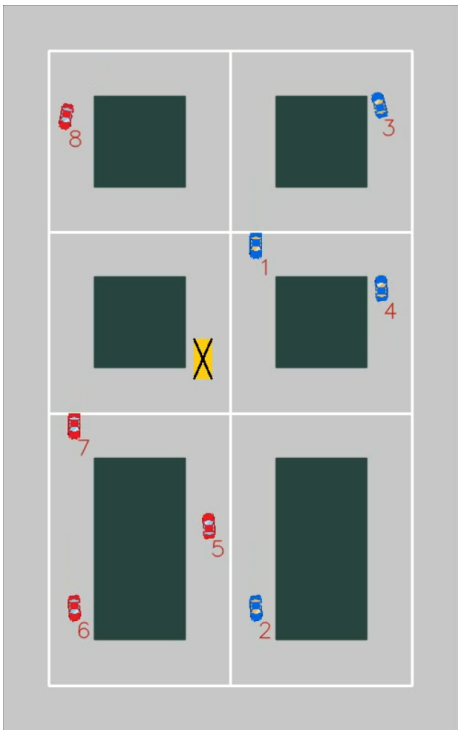

図 5 Harmo Traffic Simulator におけるマップ，車両と障害物の外観

4.2 Frenet Optimal Trajectory の適用

Frenet Optimal Trajectory によって障害物を追い越すように生成された経路を，P 制御で速度を制御し，Pure Pursuit でステアリング角度を制御することで追従して追い越しを実現する．Frenet Optimal Trajectory により走行途中に動的に経路を生成することで，経路を生成→経路追従→経路生成→経路追従→…を繰り返して追い越しをしていく．

5. 深層強化学習の設定

交通流量増加に向けた円滑な追い越し行動を学習するための環境や設定を示す．学習には Prioritized Double Dueling Deep Q-Network を使用し，対向車 1 台を学習対象とする．獲得したい行動を得るための学習を進めるためにどのように環境を設定をしたのかを説明する．

5.1 環境

強化学習ではエージェントが何らかの行動をとると，環境が何らかの状態，報酬を返す．今回はその環境にあたるのが Harmo Traffic Simulator であり，車両の位置，速度，角度などの状態が遷移していく．図 5 に今回使用する Harmo Traffic Simulator におけるマップや車両と障害物の外観を示す．赤い車両が追い越し車，青い車両が対向車となっていて車両にふられている番号が 1 番の車両が強化学習対象車である．マップ中央の黄色四角にバツが描かれているのが障害物である．追い越し車はマップ左側を左回りに走行し，対向車はマップ右側を左回りに走行するので，追い越し車が障害物を避けようとするマップ中央で追い越し車と対向車のインタラクションが必要となる．

5.2 強化学習の対象となるエージェント

強化学習対象は対向車として学習する．追い越しの場面では，追い越し車が追い越しできるかどうかは対向車の意思決定が関係している．現実の交通ルールで考えると，追い越し車は対向車がスピードを出して向かってきているときは追い越しできない．つまり，追い越し車の意思決定よりも対向車の意思決定のほうが行動決定に大きくかかわっている．円滑な追い越しを実現するためには，対向車が減速，もしくは停止して追い越し車を譲るという意思決定をすることが必要である．そこで，対向車は交通ルールに従って走行するか，追い越し車を譲るために減速，もしくは停止するかを学習し，円滑な追い越しのための行動を獲得する．

5.3 行動空間

対向車の行動空間は，追い越し車がいるか関係なく交通ルールで走行するか，追い越し車を譲るために減速（速度が 0 になれば停止）するかの 2 つにした．対向車が交通ルールで走行していれば追い越し車は追い越しせず，対向車が譲る行動をしているときはシグナルを追い越し車に出して追い越し車が追い越しをする．

5.4 観測空間

エージェントは返された状態をすべて使用きるわけではなく，観測として状態の一部分を得る．観測空間は，対向車の台数，追い越し車の台数，強化学習対象車の速度，先頭追い越し車の速度，強化学習対象車と障害物との距離，強化学習対象車と先頭追い越し車との距離の 6 つとした．深層強化学習は観測のスケールが異なると学習に影響を及ぼすことがあるので，それぞれ観測は正規化されている．

まず，対向車と追い越し車の台数は交通流量が増加するかどうかにかかわってくる．対向車が少なく，追い越し車が多い状況では追い越し車を譲ったほうが交通流量が大きくなる．しかし，逆の場合は譲っても交通流量は小さくなる．よって，車両の台数は強化学習車が追い越し車を譲るかどうかに，大きく影響を与える観測である．

また，速度も重要な観測である．追い越し車の速度が強化学習対象車の速度よりも相対的に小さい場合は，追い越し車の台数が多い状況であっても追い越し車を譲らずに走り抜けていったほうが無駄に減速しなくてもよくなる．

障害物との距離は，近ければ急減速をしなければならなく，遠ければ少しの減速で譲ることができる．対向車との距離も，遠ければ譲る必要はなくなるので行動決定にかかわる．

表 1 学習に使用するハイパーパラメータ

ハイパーパラメータ	値
学習率	0.0001
バッチサイズ	32
経験再生のバッファサイズ	50,000
ターゲットネットワークの更新頻度	100
メインネットワークの更新頻度	1
割引率 γ	0.99
学習を開始するステップ数	100
学習開始ステップの ϵ	1.0
学習終了ステップの ϵ	0.05
ϵ-decay	100
優先度付き経験再生のパラメータ α	0.7
優先度付き経験再生のパラメータ β	0.4

5.5 報酬

報酬は 1 ステップでの車両全体の走行距離の平均とした．今回の設定では，強化学習車である対向車が障害物の横を通過した後は行動を学習しても意味がない．そこで，障害物の横を通過したときが最後のステップとし，そこから一定時間走行させ，その分の割引報酬の総和を最後の報酬として与えている．追い越し車を譲る行動を獲得する目的は全体の交通流量を増加させることである．この報酬が多くもらえることが全体の交通流量を増加することに直結する．

5.6 エピソード設定

図 5 に示されているマップ上を車両が走行する．追い越し車は赤色の車両で，マップ左側を左回りに走行し続ける．対向車は青色の車両で，マップの右側を左回りに走行し続ける．車両には番号が割り振られていて，強化学習対象車は 5 の 1 番の車両である．この車両がマップ一番上の真ん中の交差点に差し掛かるとエピソードが始まり，1 周して戻ってくるとエピソードが終わり，次のエピソードが始まる．1 つのシナリオを学習し続けても汎化性能がないので，エピソードの始まりには車両をランダムな台数で配置する．追い越し車，対向車それぞれ 8 か所の開始地点から 6 台を上限として配置している．エピソードごとに車両の台数をランダムに変更することでエピソードの偏りなく学習させている．

5.7 ハイパーパラメータ

学習に使用したハイパーパラメータ，ニューラルネットワークの構造は 1，2 の通りである．

6. ソフトウェアシミュレーション環境での実験・評価

6.1 実験目的

実験の目的は，深層強化学習によって得た行動によって，追い越し場面において車両全体の交通流量が増加することを検証することである．

表 2 ニューラルネットワークの構造

ハイパーパラメータ	値
入力層	6
隠れ層 1	256
隠れ層 2	128
出力層	2
活性化関数	Relu
初期値	He の初期値
乱数シード値	42

6.2 実験方法

Harmo Traffic Simulator 上で，対向車を強化学習対象として追い越し車を譲る行動を学習させる．交通ルールでの走行と，強化学習後の走行とで 3,000step（1 ステップ 0.1 秒，車両の速度 1m/s）分の全車両の走行距離の平均を測定，比較することで走行距離の増加を検証する．交通ルールでの走行とは，対向車は追い越し車を無視して速度を落とさずに走行し，追い越し車は対向車がいれば追い越しをしないというルールのもとでの走行である．また，学習時に報酬が最大となるモデルを使用したいが，今回の実験では車両の台数によって報酬のスケールが合わないので使用できない．台数が少ないときは報酬となる走行距離の平均が 1 台の減速や停止によって大きく変化してしまい，台数が多いときは走行距離の平均が 1 台減速，停止しただけではあまり変化しない．これにより，台数が少なければ極端に報酬が多いか少ないかになり，台数が多ければあまり報酬がエピソードごとに変化しなくなるので，スケールのずれが生じている．よって，今回の実験では 50,000 ステップでの学習後の走行と 100,000 ステップでの学習の学習後の走行での比較をする．実験は追い越し車と対向車の台数が 2 台-2 台, 2 台-4 台, 2 台-6 台, 4 台-2 台, 4 台-4 台, 4 台-6 台, 6 台-2 台, 6 台-4 台, 6 台-6 台の設定で行う．

6.3 実験結果

実験結果を Table.3，図 6，図 7 に示す．

Table.3 では，3,000step 分走行させ，交通ルールに即した走行をするときと，強化学習後に追い越し車を譲る行動を獲得した時との走行距離の比較をしている．図 6，図 7 では，それぞれ走行距離の増加率（学習後の走行距離÷交通ルールでの走行距離）を示している．

6.3.1 50,000 ステップ学習後

50,000 ステップで学習した後は，設定 1，2，4 を除いて走行距離が増加するという結果になった．追い越し車のほうが少ない状況でも対向車が追い越し車にスムーズな追い越しをさせてあげることによって交通流量が増加した（設

表 3　走行距離の比較

設定	追い越し車の台数	対向車の台数	交通ルールでの走行距離（m）	学習後の走行距離（m）
1	2	2	294.7	286.6
2	2	4	287.6	281.1
3	2	6	225.3	254.7
4	4	2	289.7	289.7
5	4	4	242.0	250.6
6	4	6	196.8	210.9
7	6	2	248.6	250.7
8	6	4	198.5	210.7
9	6	6	166.9	177.6

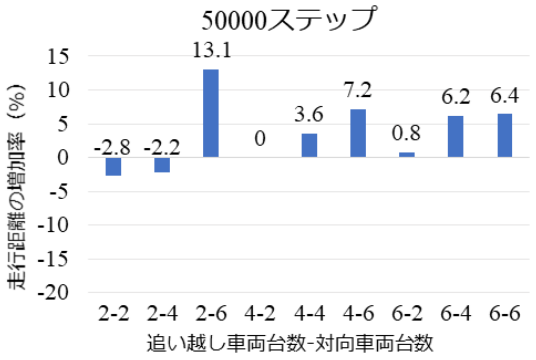

図 6　交通ルールと学習後との走行距離の割合比較（50,000 ステップ）

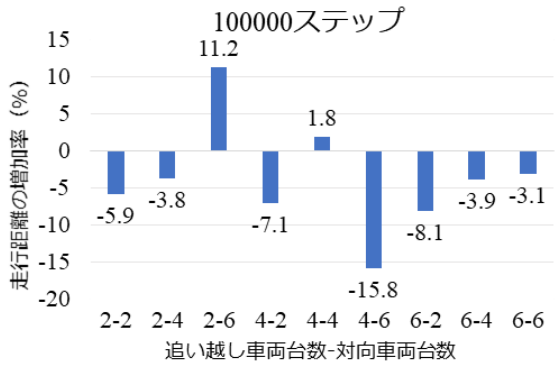

図 7　交通ルールと学習後との走行距離の割合比較（100,000 ステップ）

定 3，6，9）．その一例として，図 8 のような状況のとき，現実に即した形で走行すれば 8 番は追い越し始めているのでそのまま追い越しをして，7 番はまだ追い越しをしていないので減速して停止する．そうすると，7 番の車両は対向車の 1 番から 6 番までの車両が走り去るまで追い越しできずに停止していることになる．停止している車両が多いほど全体としての交通流量は低下してしまうので，強化学習後は図 8 の状況で 7 番の車両も追い越しさせることで交通流量を増加させることに成功している．

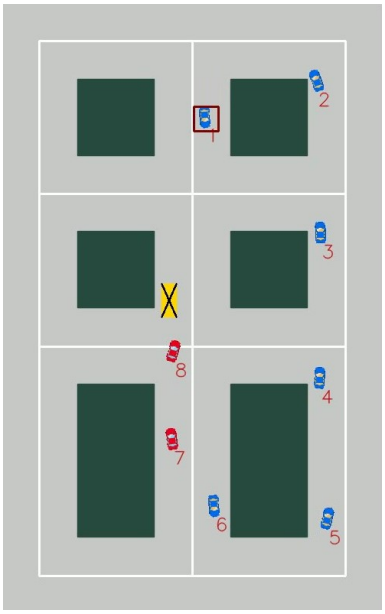

図 8　追い越し車を譲ることで交通流量が増加する場面

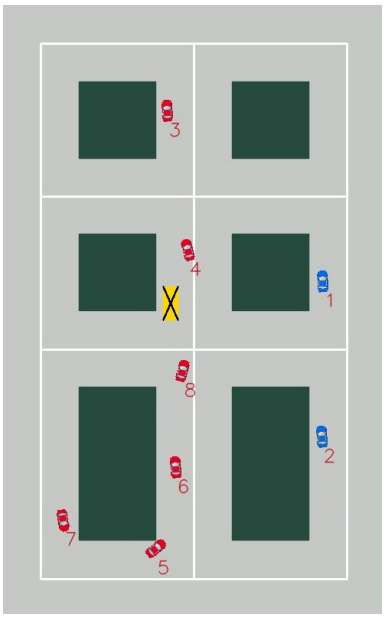

図 9　対向車の台数が少なく，追い越し車が減速することなく追い越しできる場面

対して，対向車が少ない状況ではほぼ交通ルールと学習後の走行距離は変化しなかった．これは，対向車が少ないと，追い越し車が譲られないと追い越しできないという状況があまりないことが原因である．その一例として，図 9 のように対向車が疎になっているときは追い越し車両は減速，停止をする車両はほぼいない．交通ルールの走行でも学習後の走行でも，追い越しのために譲るということが少なく，走行距離の増加は見られなかった．

また，設定 1，2 では交通ルールの走行のほうが走行距離が大きいという結果になっている．これは，車両が少ないと，追い越し車の追い越し時に，追い越し車と対向車のインタラクションが必要となることがあまりなく，学習時

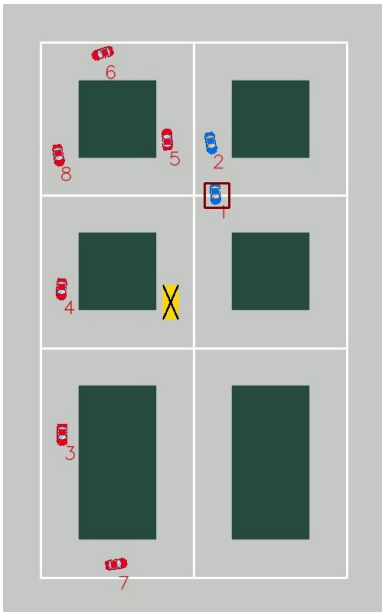

図 10 追い越し車がいない状況でも停止し続ける場面

に車両が少ない状況でのインタラクションを学習していないことが原因である可能性がある．

6.3.2 100,000 ステップ学習後

100,000 ステップ学習した後は，ほとんどの車両台数設定のときで走行距離の低下が見られた．学習後の走行を見てみると，対向車が意味のない停止をして追い越し車を譲るというような行動が見られた．図 10 のように，追い越し車がいない状況でも停止しつづけ，追い越し車が来ている途中で前進し始めるという走行をしていた．これにより，停止した強化学習対象車，その後続車，追い越し車がスムーズに走行できなくなり，走行距離が大きく低下した．50,000 ステップのときよりも 100,000 ステップの方が走行距離が低下したのは，割引率により，ステップ数を多くした方が報酬が大きくなると学習してしまったからだと考えられる．学習時にランダムに配置される車両の台数が決められることにより，報酬のスケールがエピソードごとに異なり，それによっても学習を難しくしてしまっている．

7. まとめ

交通流量増加に向けた追い越し時の譲り合いの行動を深層強化学習によって対向車に獲得させ，交通ルールの走行と強化学習後の走行とで走行距離を比較する実験を行った．50,000 ステップの学習により，追い越し車を譲ることによって円滑な追い越しが実現でき，減速や停止する車両が少なくなるように行動することで交通流量を増加させることができた．しかし，100,000 ステップ学習させると走行距離が低下した．現実では動かない障害物ではなく低速走行車の追い越しであったり，道路が曲がっていたりといった複雑な状況が考えられるが，今回は追い越しの状況としては直線の道路上にある動かない障害物を追い越すというシンプルな状況を想定して実験を行った．深層強化学習により，そのようなシンプルな状況で交通流量が増加することが検証できた．

参考文献

[1] 国土交通省道路局. 道路交通センサスから見た道路交通の現状，推移（データ集）. https://www.mlit.go.jp/road/ir/ir-data/data_shu.html.
[2] 国土交通省. 効果的な渋滞対策の推進. https://www.mlit.go.jp/road/ir/ir-perform/h18/07.pdf.
[3] 国土交通省. 高速道路の利用状況. https://www.mlit.go.jp/road/ir/ir-council/highway/4pdf/22.pdf.
[4] Python で学ぶ強化学習: 入門から実践まで. 機械学習スタートアップシリーズ. 講談社, 2019.
[5] Du X. Wang G. Liang, X. and Z Han. Deep reinforcement learning for traffic light control in vehicular networks, 2018.
[6] Ryosuke Nishi, Hiroshi Miki, Akiyasu Tomoeda, and Katsuhiro Nishinari. Achievement of alternative configurations of vehicles on multiple lanes. *Phys. Rev. E*, Vol. 79, p. 066119, Jun 2009.
[7] Chaymaa Lamini, Said Benhlima, and Ali Elbekri. Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. *Procedia Computer Science*, Vol. 127, pp. 180–189, 2018.
[8] Xiaoyun Lei, Zhian Zhang, and Peifang Dong. Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning. *Journal of Robotics*, 2018.
[9] Michael Brand, Michael Masuda, Nicole Wehner, and Xiao-Hua Yu. Ant Colony Optimization Algorithm for Robot Path Planning. *IEEE*, Vol. 2018, , 2010.
[10] Moritz Werling, Julius Ziegler, Soren Kammel, and Sebastian Thrun. Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame. *IEEE*, 2010.
[11] Eclipse sumo – simulation of urban mobility. https://www.eclipse.org/sumo/.
[12] Flow. https://flow-project.github.io/.
[13] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.
[14] Schrittwieser J. Simonyan K. Silver, D. Mastering the game of go without human knowledge, 2017.
[15] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30, No. 1, Mar. 2016.
[16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48 of *Proceedings of Machine Learning Research*, pp. 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR.
[17] T. Schaul, John Quan, Ioannis Antonoglou, and D. Silver. Prioritized experience replay. *CoRR*, Vol. abs/1511.05952, , 2016.
[18] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms, 2018.