深層強化学習を用いた運転行動学習のための走行シミュレータの検討

米元聡[1]　菅河凌太[1]

概要：深層強化学習を用いた運転行動学習のための走行シミュレータについて述べる．フロントカメラから見た走行レーンを手がかりに車エージェントの行動学習を行う．行動学習では，走行シミュレータの生成する走行レーンの投影像を観測画像として用いる．獲得した学習モデルのテスト走行では，レーン推定の結果得られるレーンモデル画像を観測画像として用いる．実験では，走行シミュレータの道路コースをもとに行動学習した結果，推定したレーンモデル画像を利用したテスト走行の結果について示す．

キーワード：自動運転シミュレーション，深層強化学習，レーン推定

A Driving Simulator for Autonomous Driving Based on Deep Reinforcement Learning

SATOSHI YONEMOTO[†1]　RYOTA SUGAWA[†1]

Abstract: This paper presents a new driving simulator for autonomous driving. In the simulator, the front camera image is used as a screen image for deep reinforcement learning. In the learning process, the screen image is supplied by projecting 3D lane models in the road course. In test course driving for the learning model, lane detection is performed, then the estimated image is used as a screen image. In experiments, we show the learning results and the results of test driving using the estimated image.

Keywords: Self-Driving Simulator, Deep Reinforcement Learning and Lane Detection

1. はじめに

現在，自動運転技術が実用化され，自動運転のバスや運搬車両が公道を走る段階に来ている．自動運転は，LiDAR，ミリ波レーダなどの高性能なセンサの利用，リアルタイム画像解析，経路計画・軌道追従など多くの技術を結集することで実現されている．特に実世界を把握するための道路画像解析，車両制御のための行動決定では人工知能技術をベースにした手法が成果を挙げている．中でも，深層強化学習による自動運転技術に注目が集まっている．例えば，「End-to-End Learning」という，自動で特徴抽出までを行うアプローチが提案されており，自動運転の場合は，道路画像を入力，車の行動を出力とする形でニューラルネットワーク学習を行う．道路画像をそのまま用いずに処理画像を用いることもある．その他，先進的な自動運転に関連する研究には，Waymo 社の模倣学習（Google 社の Self-Driving Car Project）[1]，Apple 社の低リスク運転行動方策の学習[2]などがあり，自動車メーカー以外からのプロジェクトも進行している点が興味深い．

現在，自動運転に利用可能な走行シミュレータが数多く提案されている[3][4][5]．例えば，CARLA[3]では，最新の CG 技術を用いて，天候や街並みなどをリアリスティックに表現した道路シーンの生成が可能であり，公道を走る自動運転車の学習を想定した本格的なシミュレータとなっている．オープンソースシミュレータとして自動運転エンジニアに同じ走行環境を提供することで，開発手法の公正な性能比較ができるようになっている．一方，Donkey Car と呼ばれる小規模な「Self-Driving Car」の開発も盛んであり，個人レベルでも走行シミュレータや実機の開発ができるようになりつつある[6][7].

本研究の目的は，深層強化学習を用いた自動運転車エージェントのための走行シミュレータを実現することである．先行研究として，車上方から見た俯瞰画像を観測画像に用いた深層強化学習による行動学習を提案した[8]．実機では，フロントカメラを設置して用いることが多い状況を踏まえ，フロントカメラ視点の観測画像にも対応させた．本研究で実現する走行シミュレータは，自動運転行動の学習を実機で行う場合の学習済モデルの事前獲得やシミュレーションによる走行能力の検証に利用できると考える．

本研究の特徴は以下の通りである．

- OpenAI Gym[9], Tensorflow および Keras[10]を用いて

1 九州産業大学
Kyusyu Sangyo Univ.

車エージェントを実装し，道路画像をもとにした深層強化学習による運転行動学習を行う．ステアリング，加減速の制御のみを扱う簡易な車両モデルを利用する．

- 走行シミュレーション用の道路コースを自由に作成できる．走行レーンは 2 本の曲線で表現する．走行レーンの観測画像として，フロントカメラ視点の画像（あるいは車上方から見た俯瞰画像）を用いる．
- 走行シミュレータでのテスト走行時に実画像を処理する場合と同じレーン推定アルゴリズムを適用できるようにする．

2. 提案手法

2.1 処理概要

本研究で想定する自動運転行動学習の概要を図 1 に示す．以下，図に示した各処理の詳細を述べる．

(1) レーンの推定(Lane Detection)

本研究では，両端を線で区切った走行レーンが存在するシーンを想定する．車の進行方向の 1 本の走行レーンについてレーンモデルを用いて画像中の走行レーンの領域を推定する．レーンモデルとは左右 2 本の曲線の対のことであり，フロントカメラから取得したエッジ情報を用いてパラメータ推定する．この推定手法は lane detection あるいは lane finding と呼ばれ，エッジを手がかりとした手法で直線近似するものから道路の曲率を求めるものまで数多く提案されている[5]．今回の報告では，走行シミュレータ上の行動学習に焦点を当てているため，実画像からレーンを直接推定する処理の詳細については省略するが，実機の場合の観測画像取得の際にはレーンモデルの推定を前提とする．

(2) 障害物の推定(Obstacle Avoidance)

レーン推定と同時に障害物の推定も行う．画像解析により障害物のセグメンテーション画像を求める．(1)の推定したレーン内について推定する．

(3) レーン画像を用いた行動決定

行動決定には「DQN 法による行動学習」により獲得したモデルを利用する．ここで，行動とは「レーン間の進行」を意味する．行動学習時は，3 次元レーン曲線の投影像を観測画像として用いる．テスト走行時は，推定したレーンモデルを画像中にリキャストし観測画像として用いる．

(4) ブレーキ制御

走行レーン内に障害物があると判断する場合は，行動を「ブレーキ制御」に切り替える．行動決定には行動学習により獲得したモデルを利用する．ブレーキ制御の行動は，レーンモデル画像，障害物のセグメンテーション画像をもとに学習する．

(5) Waypoint 情報の設定

交差点を想定する場合，自己位置を把握し，右左折・直進を判断する必要がある．進行のためには自己位置，進行方向に関する情報（waypoint 情報と呼ぶことにする）が必要となる．交差点侵入時，waypoint 情報を用いて右左折の方向に沿って仮想的なレーンを生成するのに利用する．

以上が，本研究が想定する自動運転行動学習の処理内容である．今回の報告では，実機ではなく走行シミュレータから得られる 3 次元レーン曲線の投影像を用いて行動学習する部分を中心に述べ，(2)(4)(5)の詳細については省略する．

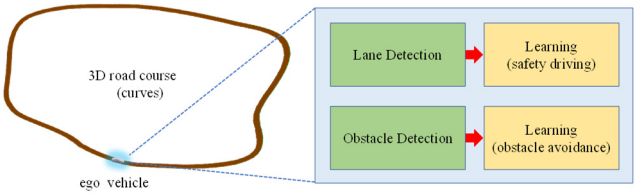

図 1 自動運転行動学習の概要

Figure 1 Our Framework for Self-driving Car Learning.

2.2 走行シミュレータ

実機で行動学習する場合でもシミュレーションにより事前に行動学習を進めておくことが一般的である．このためには，走行シミュレータにおいても実環境に近い道路シーンの表現や，実機に近い車両モデルの導入が望まれる．一方，特定の実機に依存する部分が多すぎると走行シミュレータとしての汎用性が失われてしまう恐れがある．そこで本研究では，走行シミュレータを道路画像生成による解析専用のツールと考える．具体的には，道路を 2 本の曲線対による走行レーンのみで表現する．図 2 に走行シミュレータで作成した道路コースの例を示す．この例のように道路コースは連結した曲線セグメントであり，曲線セグメントにベジェ曲線セグメントを用いている[8]．曲線セグメントを自由に連結し，制御点のパラメータを自由に変動させることで様々な道路コースを表現できる．必ずしもループ状にする必要はない．なお，このような曲線による道路コースは，実際の地図を参考に作成することも容易である．

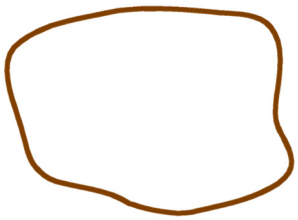

図 2 道路コースの例

Figure 2 An Example of a Road Course.

(1) ランダムコースの自動生成

制御点のパラメータをランダムに変更することで類似の道路コースを自動生成する[8]．コースの難易度はその変動幅で決定できるが，変動幅が大きいとカーブの多い曲がりくねった道となるので注意する．行動学習では，エピソード

終了時にコースをランダムに自動生成することで様々なコース形状に対処できるようにする．レーン幅の違いは行動学習に影響を与えるが，今回は同じ幅の道路コースとした．

(2) **交差点の設定**

交差点を設定する場合，別のレーンを交差させる形で実現する．この場合，走行シミュレータにおいてどの進路を選択すればよいかを決定する必要がある．前述したように，交差点侵入時はwaypoint情報を用いて進行するレーンを決定する．

3. レーンモデルを用いた観測画像の生成

3.1 レーンモデルの定義

道路画像解析手法，画像処理専用ハードウェア技術の進展により安定したリアルタイムのレーン推定が実現されつつある．本研究では，実機のフロントカメラから得た画像に対し，安定してレーンモデルを推定できると仮定する．つまり，モデルベースのレーン推定手法を利用する前提とする．モデルベースのレーン推定結果を用いることによる利点は，レーン部分を途切れなく表現でき，シミュレータの場合と実画像の場合とをシームレスにつなぐ道路画像生成による解析が行えることである．すなわち，レーンモデルのリキャストにより画像上に連続したレーン画素をとらえることができる．本研究では，このレーンモデル画像をテスト走行時の観測画像として用いる．

図3は走行シミュレータ上の車エージェントのフロントカメラから見た走行レーンのエッジ画像(a)に対しレーンモデルの推定に利用したエッジおよび推定したモデルを重畳表示した結果(b)を示している．本走行シミュレータではレーンモデル画像(c)を観測画像としてテスト走行の際に利用する（わかりやすいように色付けしている）．この例は理想的なエッジ画像に対する推定結果である．(b)(c)のレーンモデルの推定結果は，レーン推定によく用いられている，多項式曲線フィッティングにより得たものである[5][11].

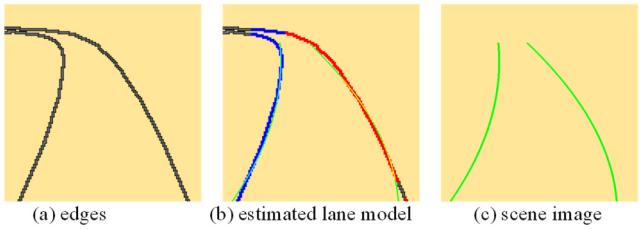

図3 レーンモデルの推定と観測画像

Figure 3 Lane Model Fitting and the Estimated Image.

3.2 車両モデルの定義

車エージェントの車両モデルとしての仕様について述べる．車両モデルの制御にはキネマティックモデルを利用する[8]．行動学習時は左右にハンドルを切るか直進するかの3種類の操作のみ行えるものとする．アクセルにより加速した後は一定の速度で運転する．ブレーキ制御の行動の際には減速によるブレーキ操作を行う．より高度なモデルや連続値の制御が必要等，この車両モデルで不十分な場合は入れ替える必要があるが，その他の車両モデルについては未検討である．以降の実験では，車の全長45，車幅20とし，地面から 20 の高さで車の中心付近にフロントカメラを設置した．実験では，ピッチ角θを30°に設定している．

3.3 レーン投影像の生成

走行シミュレータにおいて，車エージェントのフロントカメラから3次元レーン曲線の投影像を得る方法について述べる．走行シミュレータの場合，コース上の3次元のレーンを投影することにより生成することができる．この投影像はレーンモデル画像の真値に相当する．行動学習時の観測画像としてこの投影像を用いる．

フロントカメラの座標系の定義を図4に示す．ワールド座標系を車両中心（地面が高さ 0）に設定し，カメラ座標系を図のように設定した．カメラ座標系のパラメータとして，画角fovy，カメラの位置t，ピッチ角θを設定する．レーン上の3次元の点は，ワールド座標からカメラ座標への変換，ビューポイント変換（クリッピング，リサイズ含む）を経て観測画像上の点として求めることができる．なお，カメラのピッチ角の設定によってはボンネットが写ることがあるが，ボンネットは描画しないものとする．

上述した内容は走行シミュレータを用いる場合の仮想カメラの設定ついてであり，実機の実カメラに合わせる場合はキャリブレーションを行い，実カメラとワールド座標との関係(カメラの位置･姿勢)，用いるレンズの焦点距離，サイズなどの情報を事前に調べ仮想カメラに設定すればよい．また，レンズひずみへの対処も行う必要がある．仮想カメラの設定が，実機のカメラ設定と違いが生じないように対処する．

図5に示す，フロントカメラから見た道路の画像は上述の方法により得ることができる．この例は，走行レーンの中心に位置する車エージェントが図中央の正面を向いた方向を基準として，左に24°，右に23°向いたときの投影像を表しており，左右の画像は今回用いた仮想カメラの画角の設定で道路の両端のエッジが観測できる限界を意味する．

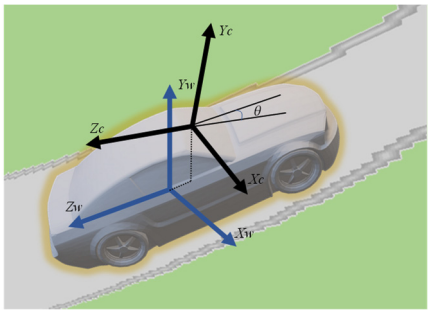

図4 フロントカメラの座標系

Figure 4 Front Camera Coordinate System.

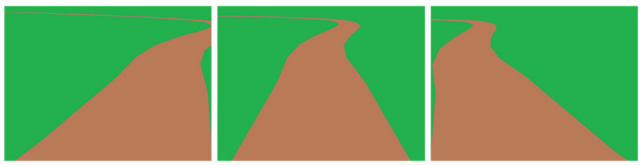

図 5　レーン投影像の生成
Figure 5　3D Lame Model Projection.

4. DQN 法による運転行動の学習

4.1 DQN(Deep Q-Network)法

車エージェントの DQN 法による運転行動の学習についてその詳細を述べる．深層強化学習は，強化学習に深層学習を組み合わせた手法であり，DQN 法はその手法の 1 つである．DQN 法は，ゲーム画面をもとに AI コントローラがよりスコアを稼ぐ行動を学習していくというゲーム AI の問題のために提案された方法[12]で，2 人ゲーム[13]，ロボットの制御，自動車の自動運転などに応用されている．行動の連続値の学習を行える DDPG 法[14]などの手法もあるが，本研究では離散的なステアリング角操作量を行動により制御できる DQN 法を用いる．DQN 法の改良手法として Rainbow[15]，D-DQN，Dueling Network などが利用可能であるが，運転行動の学習時に特段精度が向上しなかったため DQN 法を用いている．本報告では，既存のアルゴリズムを用い，行動学習のアルゴリズムの性能比較については議論しない．

4.2 車のエージェントの定義

本研究では，車エージェントは以下の 2 つのタスクを切り替える仕様となっている．1 つは，走行レーン内に安定に進行させる主タスクであり，もう 1 つは障害物存在時に車を停止させる副タスクである．簡単な走行実験を行う場合は，主タスクのみを遂行させればよい．

主タスク遂行における，車エージェントの要素は以下の通りである．

(1) **行動**

車エージェントは直進・左右回転の 3 種類のハンドル操作を行う．車両モデルをより複雑にし，さらに多くの操作量を追加することも考えられるが，ここでは最小の構成としている．なおブレーキは副タスクで制御する．

(2) **知覚**

レーン画像を観測画像として与える．サイズは 128×128 とした．DQN 法における window length は 4 とした．

(3) **報酬・終了条件**

進行できれば+1 を与える．レーン画素を横切ればエピソード終了もしくは目標のステップ数（走行距離）達成でエピソード終了となる．

5. 実験結果

5.1 走行シミュレータの実装

本研究では，OpenAI Gym という強化学習向けのエージェント開発のフレームワークを利用して車エージェントの実装を行った．この車エージェントを含む環境に DQN 法を組み込んだ．DQN 法の実装には Tensorflow および Keras を用いた．また，走行シミュレータの道路コース生成，フロントカメラから見た 3 次元レーン曲線の投影像の生成，レーン推定アルゴリズムについては独自に開発した．

5.2 行動学習実験

走行シミュレータを用い，ランダム生成したコースについて行動学習した結果について述べる．道路コースに用いた曲線セグメントの構成は図 2 に示す通りである．エピソード終了時，0.1 の確率で類似のコースに変更する．行動学習の際は投影像をそのまま観測画像として用いている．ただし，レーン推定に合わせるため，投影像上部をカットして用いる．

図 6 に 300 万ステップ（5040 エピソード）学習した結果を示す．横軸はエピソード数，縦軸は 20 エピソード区間ごとの報酬で，実線は移動平均，点線は区間最大，最小の報酬の推移を表している．学習後すぐに目標の報酬である 1200 を達成しているが，平均は横ばいとなった．図中，標準偏差を表す分布の幅が大きい理由として，ランダムコースでは生成されるカーブの位置が毎回異なってくることが考えられる．失敗時の報酬の値がカーブの位置で決まるため，失敗時の報酬のブレ幅が大きくなる．同様の実験を 3 回行ったが，目標ステップ数に到達する時間差はあるものの同様の結果となった．

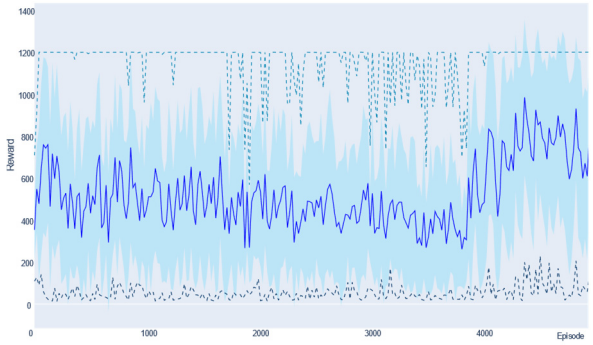

図 6　報酬の推移（行動学習時）
Figure 6　Reward History in Learning.

5.3 レーン推定実験

(1) **テスト走行時のレーン推定結果**

本研究で想定するレーン推定手法を実装し，実際に走行シミュレータに組み込んで動作確認を行った．レーン推定には 2.1 で説明したモデルベースの標準的な手法を用いた．本研究の走行シミュレータは簡易なものであるため，天候などの外乱，他の移動物体の影響などを表現できていない．

そのため, 5.2 で観測画像として用いたものは理想的なエッジ画像となっていた. 5.2 の学習済モデルをテスト走行させた際に得られるレーン投影像に対してレーン推定を適用した．実験では，急カーブのないコースを仮定し，推定不能とならないよう考慮した．真値であるエッジ画像との誤差を調べるためにモデル画素に対するエッジ画素の重なり率を調査した．平均重なり率は 0.4 と若干低い結果であったが，目視で確認する限りではほぼ重なっていると判断できた．これは図 7 に示す鳥瞰画像(b)への射影変換後，レーン上のエッジを集めた上で曲線フィッティングした後に再度射影変換する際にずれが生じたためと思われる．途中，平均重なり率が 0.1 程度に下がっている地点があったが，左側のレーン曲線の推定に一時的に失敗していることが原因であった．

(2) **レーンモデル画像を用いたテスト走行結果**

次に，推定したレーンモデル画像を観測画像として与え，テスト走行した．観測画像の違いによる走行ステップ数の変化を調査した．レーン推定が機能している区間ではレーンモデル画像を観測画像とした走行が可能であることが確認できた．投影像の場合は目標となる 1200 ステップを達成したのに対し，推定したレーンモデル画像の場合は走行距離が伸び悩んだ. 図 8 上に推定結果の例 (220 フレーム) を示す．順に，エッジ画像(a)，推定したレーンモデル（緑色）を重畳表示した結果(b)，観測画像として用いたレーンモデル画像(c)，射影変換時に計算に用いたエッジ画素とレーンモデル(d)である．エピソード失敗となる直前の結果（666 フレーム）を確認すると，カーブにおいて視界から左側のレーンが消えかけており，レーン推定に失敗しているのが確認された．図 8 下にその結果を示す．車の向きによってエッジが観測されなくなることによる曲線フィッティングの失敗はその他にも観測されたが，すぐに復帰できれば走行可能であることが確認できた．この実験ではレーン推定に簡易な手法を用いており，テスト走行の失敗は，ほぼ全てレーン推定の失敗によるものであった．多少のコースアウトに耐えられるような，よりロバストな手法の導入が必要である．

また，走行距離が伸びなかった要因の 1 つとして，行動学習時に理想的なエッジ画像を用い，レーンモデル画像を直接用いていないことも考えられる. しかし, 学習初期は，道路コースを頻繁に外れることによりレーン推定が失敗し，学習が進まない可能性があるため，レーン推定の影響を除外した形で投影像を用いた学習を行っている．

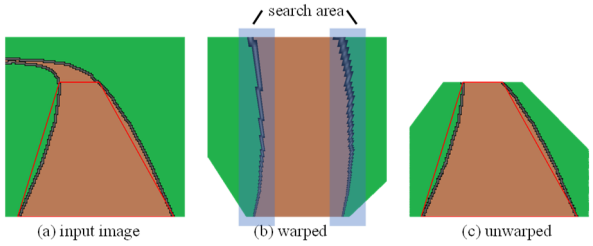

図 7　レーン推定時の射影変換

Figure 7　Image Warping in Lane Detection.

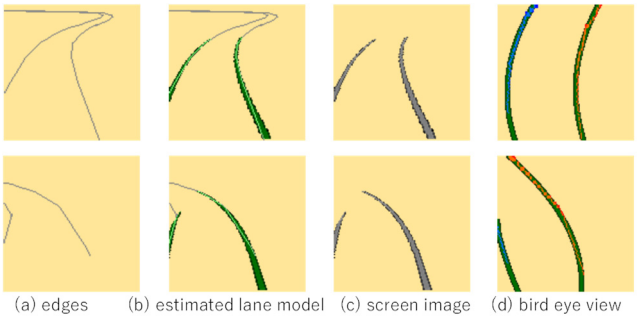

図 8　推定したレーンモデル画像の観測画像化
（上: 220 フレーム，下: 666 フレーム）

Figure 8　Screen Image Generation Using Estimated Lane Model (Up: 220 frame, Bottom: 666 frame).

最後に，学習したモデルを用いてテスト走行させた際の報酬の推移を調査した．図 9 にランダムに生成した 20 種類のコースに対してテスト走行した結果を示す．青色は投影像を観測画像に用いた結果，緑色は推定したレーンモデル画像を用いた結果である．観測画像が学習時と同じ投影像の場合はほぼすべて目標の 1200 ステップ達成しているのに対し，レーン推定結果を用いた場合は，1/5 が早い段階で失敗している．確認したところ，やはりカーブにおけるレーン推定の失敗が原因であった．また，今回設定した仮想カメラの設定がレーン推定にとっては厳しい条件となったため, 実カメラの設定を参考に見直しを行う予定である．

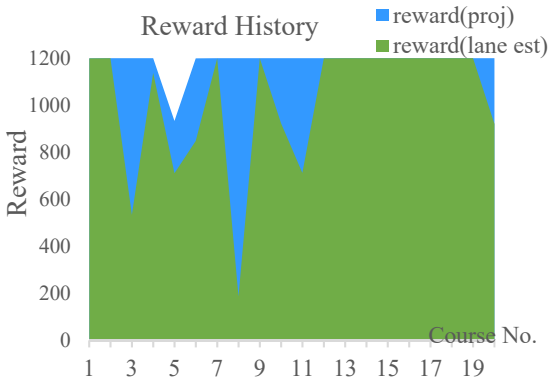

図 9　報酬の推移（テスト走行時）

Figure 9　Reward History in Test Course Driving.

(3) 実カメラとの整合

現在の走行シミュレータの実装では，投影計算が実カメラを考慮したものとなっていない．そこで，仮想カメラを実カメラを考慮したものに改良しテスト走行に用いる道路画像解析に影響がないか調査した．図 10 に実画像への 3 次元レーン曲線の投影結果を示す．事前にカメラキャリブレーションを行い，わかりやすいように画面中央付近の床面にワールド座標原点を配置した．図 10 上は直線分を，図 10 下はカーブした 3 次元レーン曲線を投影した結果である．直線分の投影については想定通りの結果となった．これより，本走行シミュレータは実機によるテスト走行と同様の走行レーンを生成できると判断する．曲線群は，床面に真値を設定していないため今回は精度の確認ができなかった．実機の道路コースを設ける際に再度検証を行う予定である．

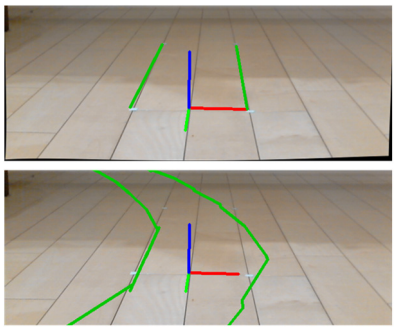

図 10　実画像への 3 次元レーン曲線の投影
Figure 10　Projection of 3D Lane Curves.

6. おわりに

本報告では，車エージェントの自動運転行動学習のための走行シミュレータについて述べた．具体的には，深層強化学習の 1 つである DQN 法を用いて生成する走行レーンの投影像から運転行動を学習する機能を有し，走行レーンの推定結果を観測画像として利用できる走行シミュレータについて提案した．また，車エージェントの知覚としてフロントカメラ視点の道路画像を用いる改良を行った．

実験では，想定する道路コースについて，レーンモデルの投影像を観測画像として用いた行動学習の性能を確認した後，レーン推定結果を観測画像として用いたテスト走行の結果を確認した．実験の結果，目標とする走行距離を達成する学習モデルが得られたが，ランダムに与えるコースによっては，カーブを曲がり切れないことがあり走行に失敗すること，実装したレーン推定アルゴリズムの性能が不十分なため，急なカーブで視界からレーンが外れる場合に曲線の推定に失敗することを確認した．これは図 5 に示したことからも明らかなように視界にレーンを捉えることができない場合があり，見失った場合のリカバリー機能が必要であるということである．今後，レーン推定および仮想カメラの設定について改良を進める．

その他の今後の課題としては，障害物判定や交差点侵入時のシミュレーションなどの動作検証，その他の車両モデルの追加検討，実際に実機を用いて走行シミュレータで学習させ事前学習の効果を検証することが挙げられる．

参考文献

[1] M. Bansal, et al. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst, 2018, arXiv:1812.03079.
[2] Y.C. Tang, et al. Worst Cases Policy Gradients, 2019, arXiv:1911.03618. J. Chen, et al. Model-free deep reinforcement learning for urban autonomous driving, In Proc. of IEEE Intelligent Transportation Systems Conference (ITSC), 2019, p.2765-2771.
[3] Dosovitskiy, Alexey, et al. CARLA: An open urban driving simulator. In: Conference on robot learning, PMLR, pp.1-16, 2017.
[4] Shah, Shital, et al. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, In: Field and service robotics. Springer, Cham, pp.621-635, 2018.
[5] Udacity Self-Driving Car Nanodegree Project, Finding lane lines, https://in.udacity.com/nanodegree .
[6] Donkey Car. Available online at http://www.donkeycar.com/ .
[7] NVIDIA JetBot, https://github.com/nvidia-ai-iot/jetbot .
[8] 米元聡，國武佑哉，道路俯瞰画像を用いた車エージェントの運転行動学習，火の国シンポジウム, 2020 年.
[9] OpenAI Gym, https://github.com/openai/gym/.
[10] Keras: The Python Deep Learning library, https://keras.io/ .
[11] Luca Venturi, Krishtof Korda, Hands-On Vision and Behavior for Self-Driving Cars, Packt pub., 2020.
[12] V. Mnih, et al. Playing atari with deep reinforcement learning, 2013, arXiv:1312.5602.
[13] D. Silver, et al. Mastering the game of Go with deep neural networks and tree search, 2016, nature, 529(7587), 484.
[14] T.P. Lillicrap et al. Continuous control with deep reinforcement learning, Proc. of ICLR2016, 2016.
[15] Hessel, Matteo, et al. Rainbow: Combining improvements in deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2018.