Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

自動運転ソフトウェア向け Back-to-Backテストフレームワーク

佐藤 隆彦[1]　三浦 啓太[1]　藤倉 俊幸[2]　安積 卓也[1]

概要：自動車の運転手不足や高齢者による運転操作ミスが社会問題になっている．そのため，自動運転システムの開発が急がれており，自動車業界ではモデルベース開発が盛んに行われている．自動車機能安全規格 ISO 26262 では，モデルベース開発において，Back-to-Back テストを行うことが要求されている．そのため，製品の動作環境に組込む際に，制御仕様と動作が一致しているかの検証をする必要がある．本研究では，自動運転ソフトウェア向けの Back-to-Back テストフレームワークを提案する．既に正しく動作しているモジュールと，その機能を移植したモジュールの入出力結果を保存・比較する．本論文では，MATLAB/Simulink で作成したモデルが正しく動いているかを，自動運転ソフトウェアである Autoware のモジュールを用いて Back-to-Back テストで評価する．

Back-to-Back Test Framework for Self-Driving Software

Abstract: The shortages of car drivers and driving mistakes by the older people have become social problems. For this reason, the development of self-driving systems is urgently needed, and model-based development is being actively pursued in the automobile industry. Automotive functional safety standard ISO 26262 requires Back-to-Back testing in model-based development. Therefore, verifying that the control specifications and operation match is necessary when porting the product in the operating environment. In this research, we propose a Back-to-Back test framework for self-driving software. This framework saves and compares the input and output results of both a module that is already working correctly and a module that has ported the same functionality. In this paper, we evaluate whether the model created by MATLAB/Simulink works correctly by the Back-to-Back test using the module of Autoware which is the self-driving software.

1. はじめに

自動運転システムは現在着実に進歩しているが，機能が増えるにつれて複雑性が増す．さらに，自動運転車両は，カメラ，レーザ光による距離測定（LiDAR），全球測位衛星システム（GNSS），ステアリング制御デバイスなど，いくつかの異なるハードウェアで構成されている．カメラは信号の色や周囲の車・歩行者を認識できる．LiDAR は，周囲 360 度の物体を点群として認識し，そこまでの距離を測定できる．GNSS は車両の現在位置を把握できる．自動運転ソフトウェアは，各ハードウェアからのデータを取得し，リアルタイムに計算して制御する．

[1] 埼玉大学
Saitama University
[2] dSPACE Japan 株式会社
dSPACE Japan

自動運転システムの課題は，その安全性をどう保証するかである．安全性のためにはより良いテストが必要であり，様々なテスト手法がある．一般的な手法としては N-FOT（Naturalistic Field Operational Tests）が挙げられる．これは，FOT（Field Operational Test）と NDS（Naturalistic Driving Study）を組み合わせた手法である [1]．自動車技術における FOT とは，長期間，実際の運転環境下で交通環境・運転操作・車両挙動の三つを，多数の車両・運転者に対して観測することで，運転を支援する技術や知識の有効性を統計的に検証することである [1]．NDS とは，実環境下で観測されるデータの頻度分布に即して，システムの最適化を行う手法である [1]．N-FOT は高価であり，実際の路上で行うため事故などの危険も伴う．さらに，自動運転システムが期待される動きとは異なる動きを見せた時に，その状態（天候や周囲の車など）を再現するのは困難である．

Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

これらの側面を考慮して，シミュレータを用いたバーチャル空間でのテストが考えられる．現在，自動運転システムのテストとして，シミュレータを使用したシナリオベースのテストなどが積極的に実施されている．シミュレータの例としてはCARLA [2] やLGSVLシミュレータ [3] などがある．これらは自動運転ソフトウェア全体の挙動のテストができる．しかし，個々のモジュールが正しく動いているかが分からないため，シミュレータ環境では単体テストがしづらいといった問題がある．例えば，新しくモジュールを追加したり更新した時，そのモジュールが正しく動いているのか，もしくは偶然正しく動いているように見えるだけなのかは，シミュレータでは判断しにくい．自動運転システムのテスト手法としては，シミュレータを除いてまだいくつかの方法がある．

テスト手法を考えるにあたり，まず，自動運転ソフトウェアは将来的にスーパーコンピュータではなく，車のバッテリーで動かせる省電力・小型・高速処理のプラットフォームで動作する必要がある．例えばAutoware [4] [5] という自動運転ソフトウェアがKALRAY MPPA-256 [6] [7] というプラットフォームへ移植 [8] されつつある．この場合，コードの並列化や複数のクラスタやコアへの割り当てといった変更が生じるので，ソフトウェアがその環境でも正しく動作するか再度テストしなければならない．実際に移植前のプログラムでは問題がなくても，移植先のプログラムでは問題が発生することがある．

上記の問題を解決するために，本論文ではMATLAB/Simulinkモデル [9]（比較対象のAutowareのノードと同じ機能のモデル）のBack-to-Backテスト [10] を提案する．Back-to-Backテストとは，二つ以上の異なるコンポーネントやシステムにおいて，同じ入力で実行し，出力を比較するテスト手法である．相違がある場合はそれを分析する．これはISO 26262でもモデルベース開発において要求されている [11]．具体的に，比較対象のAutowareノードの入出力を保存し，そのテスト対象のノードと同じ機能を有するMATLAB/Simulinkモデルにも同じ入力を与え，それぞれの出力を比較する．本論文では，この処理を自動で行うフレームワークを提案する．

本研究の貢献内容としては，Back-to-Backテストを自動で行える．MATLAB/Simulinkモデル，そのモデルから生成されたC++コード，人が書いたC++コード，これらを同様にテストできる．これにより，新規にコンポーネントを作成した時に動作の整合性が損なわれていないことを単体・結合レベルで確認できる．よって，開発の効率を上げることが可能となる．ROSの機能を用いれば，Autowareでなくても評価可能である．出力された値同士の比較をするので実行時間が異なっていても評価に影響しない．提案フレームワークはAutowareのバージョンに応じて自動生成される．本論文の貢献をまとめると以下となる:

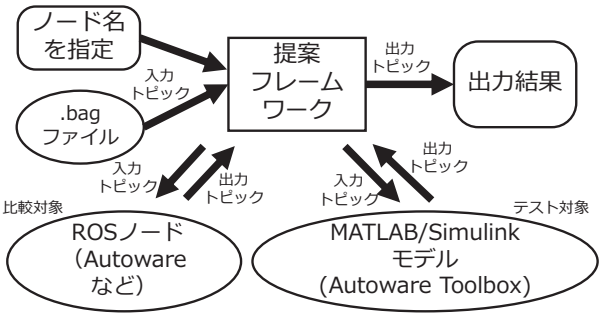

図 1 システムモデル
Fig. 1 System Model.

- ROS互換のプログラムに対してBack-to-Backテストできる
- Autowareノードと同じ機能を持つMATLAB/Simulinkモデルを，実行時間の差異に依存せず評価できる
- Autowareのバージョンに依存せず，自動で各ノードのトピック情報を取得してBack-to-Backテストできる

本論文の構成としては以下となる．2章では，本研究のシステムモデル及び前提知識について説明する．3章では，作成したBack-to-Backテストフレームワークの設計と実装について述べる．4章では，評価結果について述べる．5章では，関連研究を紹介し比較を行う．6章では，本研究のまとめとして結論を述べる．

2. システムモデル

図 1 では，提案フレームワークのシステムモデルを示している．提案フレームワークを用いることでMATLAB/Simulinkなどのノードが正しく動いているかを，bagファイル（ROSBAG（2.1.2）により，センサデータやトピック情報が記録されたファイル）とAutowareのノードを基にして，評価できる．

本章では，提案フレームワークの比較対象のROSノード（Autoware）と，テスト対象のAutoware Toolboxについて説明する．

2.1 Robot Operateing System（ROS）

カメラやLiDAR，GNSSといったハードウェアを動かすミドルウェアとして，ROS（Robot Operating System）[12] がある．ROSは，ロボット工学の分野において開発された，ライブラリ・ツールを提供するオープンソースのミドルウェアである．ROSは，世界中のコミュニティで利用されており，可能な限り分散・モジュール化されるように設計されている．ROSは，PCL（ポイントクラウドライブラリ）やMoveItといった他のライブラリとシームレスに統合できる．自動運転車両には多くのソフトウェアパッケー

Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

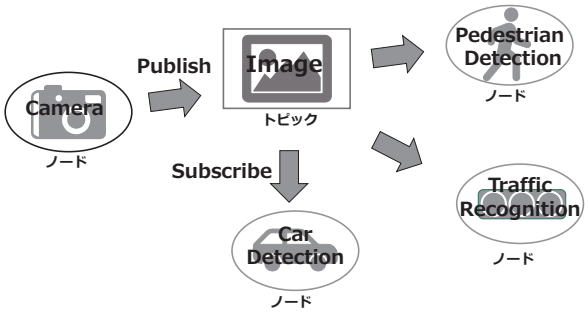

図 2　Publish/Subscribe モデル
Fig. 2　Publish Subscribe Model.

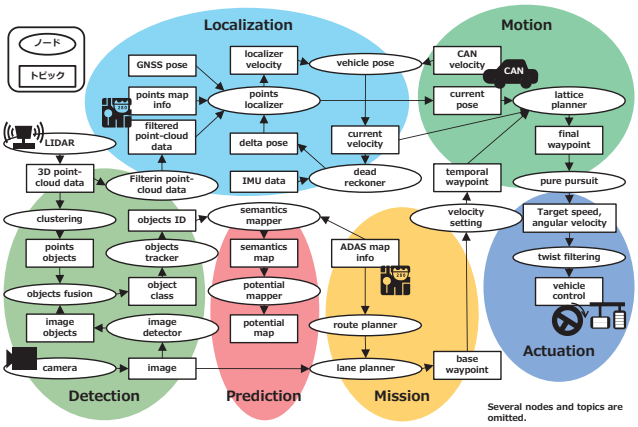

図 3　Autoware の処理の一連の流れ
Fig. 3　Autoware Process.

ジが必要なため，ROS は Autoware の開発の強力な基盤を提供する．

2.1.1　Publish/Subscribe モデル

ROS の特徴として図 **2** に示している Publish/Subscribe 通信（以下 Pub/Sub 通信）がある．プロセスはノードと呼ばれ，データはトピックと呼ばれる．Pub/Sub 通信は非同期通信である．ノードがトピックを出力として配信することを Publish といい，入力として受信することを Subscribe という．各種ノードは，ノード間の通信に必要なトピックを必要に応じて Publish/Subscribe する．個々のノードはトピックを介して通信する．ノード間に明確なインターフェースを実装する必要があるので，Pub/Sub 通信はカプセル化を向上させ，ノードの再利用性を高めるのに役立つ．それゆえ，一つのノードの仕様を変えたり新しくノードを追加しても，他のノードを変更しなくて良いという利点がある．

ノードは Autoware の個々の処理を行う．図 **3** は，自動運転システムの一連の流れをノードとトピックで表している．ノードは丸，トピックは四角で表されている．ノード間のトピックの流れは矢印で表されている．ノード間の入力と出力はトピックを介して行われる．事前に決定された各トピックのタイプは特定のデータに対応している．例えば，sensor_msgs/Image タイプは画像データに対応する．このタイプには，トピックデータを表現するために必要なデータである多くの要素が含まれている．例えば，sensor_msgs/Image タイプには，幅，高さ，色データなどの要素がある．ROS ノードは通常，C++で記述された標準プログラムである．システムにインストールされている他のソフトウェアライブラリと連携もできる．例えば，統合された視覚化ツール（RViz など）やデータ駆動型シミュレーションツール（ROSBAG など）である．

2.1.2　RViz と ROSBAG

RViz は，ROS のトピックデータを可視化する，3D 視覚化ツールである．ロボットモデルのビューを提供し，ロボットセンサからのセンサ情報をキャプチャして，キャプチャしたデータを描画する．ユーザが Autoware を使用する際，RViz によって，LiDAR から得られた点群データや道路のデータを表示する．RViz ビューアはシステムの状態を監視するのに役立つ．

ROSBAG は，ROS トピックを記録及び再生するツールセットである．例えば，自動運転車が実際の路上を走った時のトピックデータを全て記録できる．ROSBAG を使用することで，実際のハードウェアがなくても自動運転ソフトウェアモジュールを評価できるため，開発プロセスがより効率的になる．

2.2　Autoware

Autoware は，ROS に基づく完全自動運転向けのオープンソースソフトウェアである．Autoware は，センシング，自己位置推定，環境認識，経路計画，経路追従などの自動運転システムに必要なモジュールを提供している．Autoware は，各ハードウェアからの情報（点群データ，現在位置など）を受け取り（Sensing），どのような経路で走るかを計算（Computing）し，実際に車のアクセルやブレーキ，ハンドルの舵角などをコントロール（Actuation）する．Sensing において，LiDAR スキャナは，スキャンされたオブジェクトの 3D 表現である点群データを生成する．カメラは主に信号機を認識し，スキャンされたオブジェクトの追加機能を抽出するために使用される．自動運転システムの重要な部分である自己位置推定モジュールは，車両の位置を認識する． 環境認識モジュールは，車両，人，交通信号などの周囲のオブジェクトを検出する．Computing において，ミッションとモーション計画に分かれている経路計画モジュールは，自動運転の経路を計画する．Actuation において，経路追従モジュールは，加速，減速，及び操作を制御する．

2.3　Autoware Toolbox

自動車業界では，エンジンなど制御設計に MAT-

Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

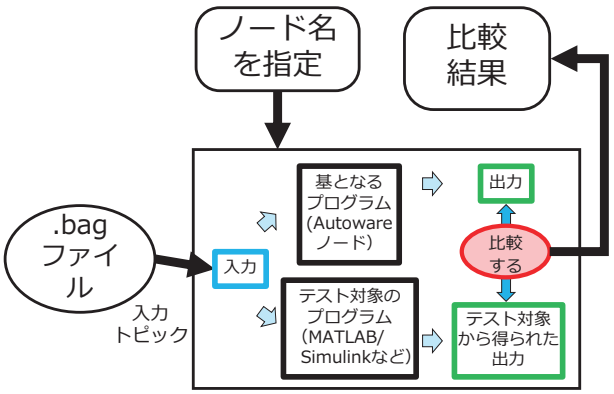

図 4　提案フレームワーク
Fig. 4　Proposed Framework.

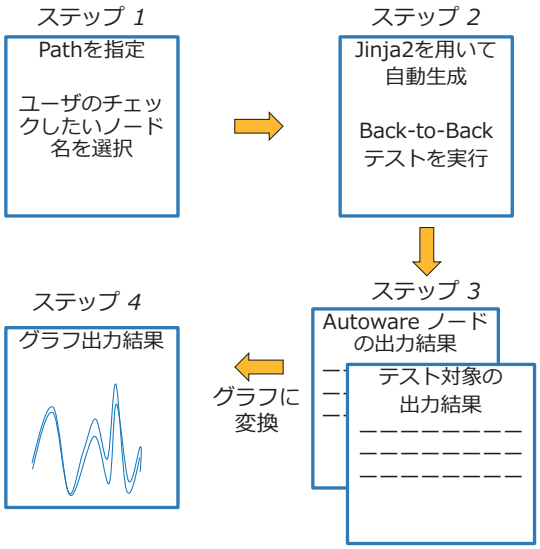

図 5　Back-to-Back テストの流れ
Fig. 5　Back-to-Back Test Process.

LAB/Simulink が広く使用されている．理由としては，MATLAB/Simulink といったモデルベース開発は，ハードウェアに実装する前にシステムをデザインし，シミュレーションを活用して開発ができるからである．仮に全てのエラーや故障を意図的に発生させて想定外をなくせるなら，実機で開発ができるが，現実には不可能である．そして，Simulink ブロックを用いれば，ソースコードを読んで理解するといったプログラミングのスキルがなくても，プログラムを書ける．そのため，開発者は自動運転システムを設計する際にも MATLAB/Simulink を使うことが考えられる．開発者が MATLAB/Simulink で作成したモデルを Autoware に組込む場合，二通り考えられる．一つは MATLAB の機能でシステムを C++コードに変換する手法である．この手法は ROS の Pub/Sub 通信には対応していないため，開発者に負担が生じる．もう一つの手法は，ROS と MATLAB とのインターフェースを提供した Robotics System Toolbox を使う手法である．この手法では，モデルのまま Autoware と連携できる．Autoware Toolbox は，Autoware と連携可能な MATLAB コードと Simulink モデルを提供する．これらは，Autoware のノードとして機能する．Autoware の開発においては MATLAB/Simulink のサンプルがなかったことや，Autoware Toolbox がオープンソースとして提供されていることから，開発者が MATLAB/Simulink で Autoware と連携した自動運転システムを設計する際に役立つ [9] [13].

Autoware Toolbox は，Autoware で利用可能な MATLAB コードと Simulink モデルを含んでいる．自動車分野では MATLAB/Simulink が自動運転システムの開発を加速するために使われている．そのため，自動車業界ではモデルベース開発が主流となっている．め，Autoware Toolbox によって開発がしやすくなった．

3. 設計と実装

図 4 では，提案フレームワークを示している．提案フレームワークを用いることで，bag ファイルの入力を基にして，MATLAB/Simulink モデルが Autoware のノードと同様の動作をするかを評価できる．bag ファイルのトピックをそれぞれのプログラムに入力し，その出力を保存して比較する．内部で行われる python のスクリプトによって，この一連の処理が自動で行われる．移植したノードが正しく動いているかを評価できるため，提案フレームワークは開発を効率化できる．

3.1 提案フレームワークの概要

Back-to-Back テストの評価方法としては，まず，評価対象のノード（MATLAB/Simulink で作成したモデル）を用意する．そして，評価対象と比較する正解データとして，ROSBAG で記録された bag ファイルを用意する．ROSBAG とは，上述の通り，実際に自動車を街中で走らせた時の ROS の任意のトピックを bag ファイルに保存する機能である．その bag ファイルに含まれる，テスト対象のノードの入力トピックを抽出する．提案フレームワークは，抽出したその入力トピックを，自動で評価対象のノードに入力する．そして，その評価対象の出力トピックを保存する．提案フレームワークは，その保存された出力トピックと，事前に抽出した出力トピックを比較する．出力が正しいか正しくないかの評価はグラフにプロットして行う．テストするノードによって入出力となるトピックの種類や個数は異なり，トピックごとに出力されるパラメータも異なるため，各ノードに応じたパラメータの抽出を行う必要がある．本研究では，python を用いてこの一連の処理のフレームワークを作成した．テスト対象のノードが選ばれると，自動で上述の処理を行い，グラフとして出力される．

Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

```
1 {% for item in node_name %}
2     {{item}}_test_file = open('result/{{item}}
          _test', 'w')
3     dump_{{item}}_test = subprocess.Popen(['
          rostopic', 'echo', '/{{item}}_test', '-p
          '], stdout={{item}}_test_file)
4
5     {{item}}_file = open('result/{{item}}', 'w')
6     dump_{{item}} = subprocess.Popen(['rostopic
          ', 'echo', '/{{item}}', '-p'], stdout={{
          item}}_file)
7 {% endfor %}
```

図 6　テンプレートコード
Fig. 6　Template Code.

3.2 Back-to-Back テストの流れ

図 5 に本研究の Back-to-Back テストの処理の流れを示す．以下に説明を順を追って記述する．

ステップ 1: 提案フレームワークで，テストしたい Autoware の Path を指定する．次にテストしたいノード名を選択する．提案フレームワークは，選択された Autoware の Path から YAML ファイルを読み込んで，自動でテスト対象のノードのトピック名を取得する．目的の入力トピックが含まれる bag ファイルも提案フレームワークで選択する．本論文で用いたのは実証実験の際のトピックを記録した bag ファイルである [14]．この bag ファイルから，選択したノードに必要な入力トピックのみを含む bag ファイルを新たに抽出・生成する．例えばユーザが twist_filter を選択すると二つの入力トピック（/config/twist_filter や/twist_raw）のみを含む bag ファイルが生成される．

ステップ 2: 選んだノードによって，出力されるトピックの数と種類が異なる．各トピックのパラメータもトピックごとに異なる．選ばれたノードによってプログラムを変更する必要がある．そこで，Jinja2 という python のテンプレートエンジンをスクリプトの記述に使用した．Jinja2 は，各ノードのトピックの個数に応じた出力ファイルを自動で生成できる．**図 6** に，テンプレートコードを示す．一行目の node_name に含まれるトピックの数だけ，for 文内の {{item}} の位置にトピック名が代入される．config.yaml ファイルに node_name の要素が記述されている．config.yaml ファイルが書き換わればそれに応じたソースコードが生成される．ユーザが選択した Autoware のバージョンに応じて config.yaml は自動生成される．

各トピックのパラメータは”rostopic echo -p”というコマンドで，CSV 形式で出力できる．”-p”を削除すると，CSV 形式ではない本来の出力トピックの情報がそのまま渡される．提案フレームワークでどちらの形式にするかを選択できる．CSV 形式で出力すれば，グラフにプロットしてグラフィカルに評価ができる．CSV 形式ではない通常の形式で出力すれば，Meld などの GUI Diff ツールを用いて，ファイルの差分で評価ができ，評価対象の出力の汎用性が広がる．

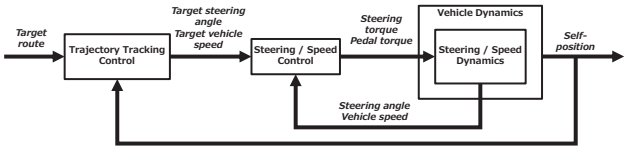

図 7　ブロック図
Fig. 7　Block Figure.

ステップ 3: 比較する二つの出力結果のうち，一つは Autoware ノードの出力トピックで，もう一つはテスト対象のノード（本論文では MATLAB/Simulink で作成された AutowareToolbox のノード）の出力トピックである．前者は，比較対象の Autoware ノードを使用する．bag ファイルのトピックを入力して，その出力結果を保存する．後者は，MATLAB/Simulnk で作成された AutowareToolbox のノードである．前者で用いた入力トピックと同じトピックを入力として与え，得られた出力をファイルに書き込んで保存する．前者と後者の出力トピックが一致すれば，両者のノードは同じ挙動をしていると評価できる．

ステップ 4: CSV 形式で出力した場合，ステップ 4 で出力トピックのパラメータはグラフに変換される．ここでも Jinja2 を用いて，出力されるトピックの数だけグラフに変換されて出力される．ただし，RViz で表示するために用意されたトピックもあり，そのトピックは CSV 形式で出力できないので，CSV 形式で出力する場合には除かれる．

4. 評価

本論文の，Back-to-Back テスト評価で用いたのは twist_filter と pure_pursuit である．これらのノードは自動運転を実現する上でも欠かせない「車両制御」に関わるノードである．

一般に自動運転において車両制御とは，経路追従制御及びステアリング・速度制御を合わせた領域を指す．経路追従制御とは，車両を目標経路に沿わせるための目標速度と角速度を計算する制御である．ステアリング・速度制御とは，ステアリング角と車速を目標値に合わせることである．ブロック図としては**図 7** に示される．ステアリング制御は，ステアリング角度が目標角度と合うための制御をする．しかしステアリングの運動は車速や重量，路面状況などによって特性が大きく変化する．この変化は外乱としてフィードバックで抑え込むことになるが，ステアリングにおいては外乱が非常に大きいため，非線形特性のコントロールが重要である．速度制御も同様に，車両速度を目標速度に合うための制御をする．ここでもエンジン特性といった非線形特性を考慮する必要がある．エンジンの特性は数式での記述が難しいため，通常，非線形性を吸収する

Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

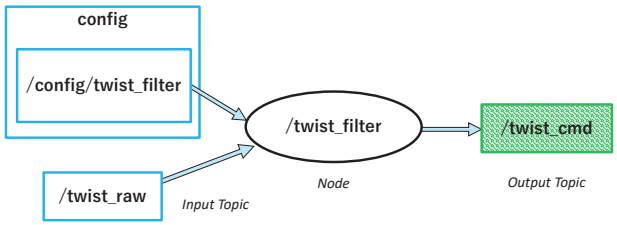

図 8　twist_filter のトピック
Fig. 8　twist_filter Topic.

マップを作成して対応する．速度の加減速は人が感じる乗り心地に直結するため，自動運転においても非常に重要な役割を持つ．経路追従制御は目標経路に沿って走行するためのステアリング目標値，速度目標値を計算し，上記二つの制御系に渡す．

以上の内容から，車両制御が正しく行われているかのチェックが，自動運転の実現にとって重要である．続いて，二つのノードの説明を述べる．

4.1　本研究のテスト対象のノード

twist_filter: このノードは，急な加速や減速を和らげる機能を持つ．ノードの処理はそこまで複雑ではない．**図 8** は twist_filter の入出力トピックを示している．twist_filter は/config/twist_filter（ユーザの設定したパラメータ）と/twist_raw（目標速度と角速度）という二つのトピックを受け取る（Subscribe）．必要な処理をした後に，/twist_cmd（車両制御信号（速度・角速度））を出力する（Publish）．/config/twist_filter では，三つのパラメータを調整できる．

- Lateral_accel_limit: 横加速度（m/s^2）をどこまで許容するか
- Lowpass_gain_linear_x: 目標並進速度に対するローパスフィルタの度合い
- Lowpass_gain_angular_z: 目標角速度に対するローパスフィルタの度合い

このノードは，ターゲットと角速度から横方向の加速度を計算する．横方向の加速度が特定の値を超えると，目標速度が低下する．自動運転車両は，フィルタリングされたコマンドに基づいて制御される．twist_cmd は field_twist_linear_x，field_twist_linear_y，field_twist_linear_z，field_twist_angular_x，field_twist_angular_y，field_twist_angular_z の六つのパラメータを持つ．field_twist_linear_x，field_twist_angular_z 以外の四つは値が全て 0 であったので，この残りの二つのトピックについて出力を比較する．

pure_pursuit: pure_pursuit は，自動運転車の actuation コマンドを生成する．このノードは，追従する経路を読み込み，現在位置から経路に沿って動くための速度と角速度を計算する．自己位置と目標点を結ぶ円の曲率を計算する．その曲率から目標点までの角速度を計算する．**図 9** は pure_pursuit の入出力トピックを示している．pure_pursuit は/current_velocity（現在速度），/config/waypoint_follower（ユーザの設定したパラメータ），/current_pose（自己位置），/final_waypoints（最終経路）の四つのトピックを受け取る．処理をした後に，/next_target_mark（次の目標点．RViz の表示に使う），/ctrl_cmd（速度とアクセルとハンドル角），/twist_raw（速度と角速度），/trajectory_circle_mark（車が追従する円弧．RViz の表示に使う）のトピックを出力する．曲率は final_waypoint，current_pose，current_velocity を使用して計算され，twist_raw として結果が出力される．twist_raw は，速度と角速度を含んでいる．twist_raw は，twist_cmd と同様に，field_twist_linear_x，field_twist_linear_y，field_twist_linear_z，field_twist_angular_x，field_twist_angular_y，field_twist_angular_z の六つのパラメータを持つ．twist_cmd と同様に，field_twist_linear_x，field_twist_angular_z 以外の四つは値が全て 0 であったので，この残りの二つのトピックについて出力の比較を行う．

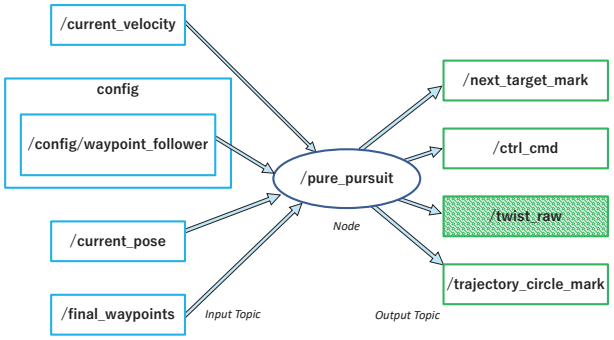

図 9　pure_pursuit のトピック
Fig. 9　pure_pursuit Topic.

表 1　評価環境
Table 1　Evaluation Environment.

OS	Ubuntu 16.04
ROS	kinetic
Memory	16GB
Autoware	1.8.0
MATLAB	2019a

4.2　評価結果と考察

評価環境としては，**表 1** である．bag ファイルの再生速度は 0.02 倍にし，Simulink モデルは新たなトピックを受け取ってから出力トピックを Publish する．twist_filter の出力トピックは約 400 個，pure_pursuit の出力トピックは約 180 個を保存して比較した．**図 10** は，提案フレームワークがその出力トピックの値を y 軸に取って作成したグラフ

である．図 10 から，同じ入力に対して，Autoware のノードの出力と，Simulink モデルの出力がほぼ一致していることが分かる．

5. 関連研究

Back-to-Back テストの統合ツールや単体テストツールが存在する．以下にその例を二つ示す．加えて，自動運転システムのテスト方法としてシミュレータでのテストが挙げられる．以下にその例を三つ示す．

MC-Verifier [15]: MC-Verifier というモデルベース開発における Back-to-Back テスト統合ツールがある．しかし，2020 年 1 月現在において，C++には対応していないので，Autoware のノードには適用できない．

カバレッジマスター winAMS [16]: 自動車業界のデファクトスタンダードツールで，単体テスト設計/テスト実行/レポート生成/レグレッションテストを自動化できる．ISO 26262/IEC 61508 を取得していて，C コードを解析してテストデータを自動生成できる．C++ 11/14 にも対応しており，xUnit や Google Test などのテストフレームワークの結果をそのまま利用できる．

MontiSim framework [17]: MontiSim framework は実世界の地図を簡素化して取り込み，その環境の中で自動運転システムの動作をテストするフレームワークである．大規模な地図を読み込んでシステム挙動をシミュレーションすることも，一つのコンポーネントレベルでの細かなシミュレーションも可能となっている．取り込んだ地図に対し，交差点ごとに道路標識や信号を付加した地図を生成できる．歩行者も出現させることができるが，直線にしか動かないため，赤信号で飛び出してくる歩行者といったシナリオを生成できない．地面の摩擦係数を変えて滑りやすくできるが，雨などの天候はモデル化できない．

CAT Vehicle Testbed [18]: 車両と車両の相互作用をテストできるマルチ車両シミュレータプラットフォームである．自動運転ソフトウェアを検証するテストセットをオープンソースで提供する．実際の車両に見立てた仮想車両を検証する物理プラットフォームを提供し，様々な大学や企業が開発している独自プラットフォームを，コストを抑えてテストできる．新たな車両モデルの生成も用意された設定ファイルを用いて行うことができる．しかし，提案フレームワークでテストした移植後のプログラム（Simulink モデルなど）が正しく動作しているかは，物理プラットフォームで動かす前に行うのが望ましい．本論文で提案した自動的な Back-to-Back テストには対応していない．

AD-EYE [19]: 再利用可能なシナリオデータベースに基づいて，機能安全要件を改善するシミュレーションプラットフォームである．シミュレーションによって設計の決定を評価し，運転中に遭遇する可能性のある様々なシナリオをテストし，安全性をより高める．学生が自動運転システムを使用して実際に経験できる教育ツールとしても機能する．

上記五つのツールやフレームワークはいずれも自動運転ソフトウェアの Back-to-Back テストに対応していない．提案フレームワークはユーザが選択したノードを自動で Back-to-Back テストできる．

6. おわりに

本論文では，自動運転ソフトウェア向けの Back-to-Back テストフレームワークを提案した．これにより，自動運転ソフトウェアの効率的な開発を促進できる．提案フレームワークは ROS の機能を有するプログラムにおいて，Back-to-Back テストできることを示した．本論文では，Autoware Toolbox の MATLAB/Simulink モデルを，提案フレームワークによって Back-to-Back テストできることを示した．評価結果から，実行時間に依存せず，対応する出力同士を比較して評価できることが示された．提案フレームワークは，本論文の評価で利用した twist_filter と pure_pursuit 以外の Back-to-Back テストにも利用できる．そして，ノード単体において Back-to-Back テストを行ったが，複数のノードを結合した状態での Back-to-Back テストもできる．Autoware のバージョンが新しくなっても，提案フレームワークに必要な YAML ファイルは自動生成されるため，提案フレームワークは高い汎用性を有している．本論文の Back-to-Back テストツールを，Autoware の KALRAY MPPA-256 などへの移植 [8] の動作確認として適用できる．

Back-to-Back テストの出力結果の比較方法として本論文ではグラフによる比較を行った．しかし，テスト対象が正しく動作していても，出力結果が必ずしも一致するとは限らない．環境の変化などによって生じ得る多少の偏差を許容する必要がある [20]．正しく動いているのに間違った評価がされることを避けるために，どのような偏差を許容するかが重要な問題の一つである．今後，出力結果の評価方法についてさらなる検討が必要である．

謝辞 本研究の成果は，JST さきがけ JPMJPR1751 の支援から得られたものである．

参考文献

[1] 平野清美，武田一哉：サイバーフィジカルシステム：3. フィールド実証実験（FOT）に向けて，情報処理，Vol. 55, No. 9, pp. 922–927 (2014).
[2] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. and Koltun, V.: CARLA: An Open Urban Driving Simulator, *In Proc. of 1st Conference on Robot Learning (CoRL)* (2017).
[3] LG Electronics: LGSVL Simulator — An Autonomous Vehicle Simulator, `https://www.lgsvlsimulator.com`.
[4] Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S.,

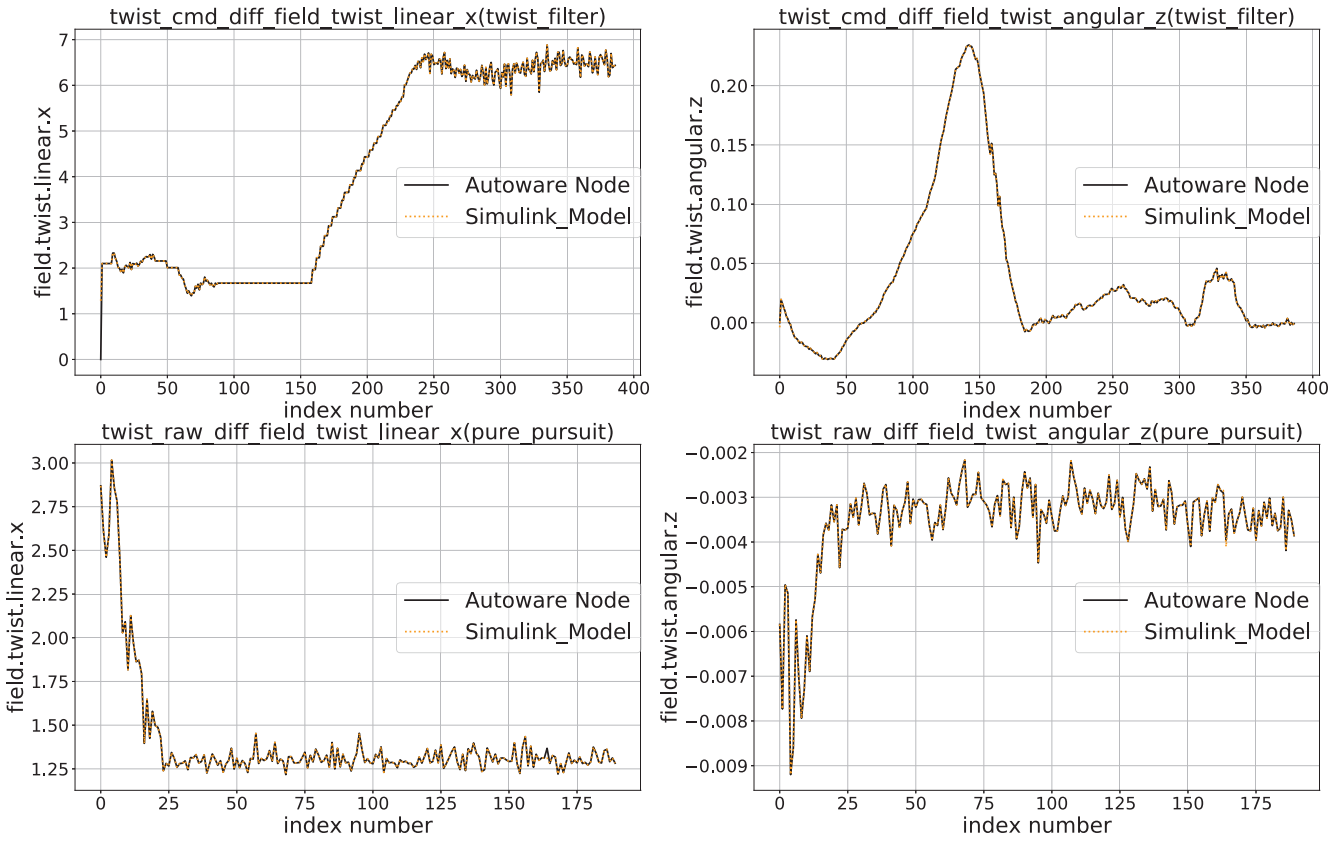

図 10 Back-to-Back テストの結果
Fig. 10 Back-to-Back Test Results.

Hirabayashi, M., Kitsukawa, Y., Monrroy, A., Ando, T., Fujii, Y. and Azumi, T.: Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems, *In Proc. of International Conference on Cyber-Physical Systems* (2018).

[5] Autoware Foundation: Autoware.AI · GitLab, `https://gitlab.com/autowarefoundation/autoware.ai`.

[6] Maruyama, Y., Kato, S. and Azumi, T.: Exploring Scalable Data Allocation and Parallel Computing on NoC-based Embedded Many Cores, *In Proc. of the 2017 IEEE International Conference on Computer Design (ICCD)*, IEEE, pp. 225–228 (2017).

[7] Honda, K., Kojima, S., Fujimoto, H., Edahiro, M. and Azumi, T.: Mapping Method of MATLAB/Simulink Model for Embedded Many-Core Platform, *In Proc. of the 28th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2020)* (2020).

[8] 安積卓也，丸山雄也，前田青也，加藤真平: Autoware on Many-core Platform: NoC ベース組込みメニーコアプロセッサ向け自動運転プラットフォーム，組込みシステムシンポジウム 2019 (2019).

[9] Tokunaga, S., Horita, Y., Oda, Y. and Azumi, T.: IDF-Autoware: Integrated Development Framework for ROS-Based Self-Driving Systems Using MATLAB/Simulink, *Workshop on Autonomous Systems Design (ASD 2019)*, pp. 3:1–3:9 (2019).

[10] 鍾　兆前, 枝廣正人: モデルベース開発における KALRAY MPPA メニーコア向け並列化，第 79 回全国大会講演論文集，Vol. 2017, No. 1, pp. 27–28 (2017).

[11] : MBD Back-to-Back テスト 定着運用サポート — ガイオ・テクノロジーソフト検証ツール/モデルベース/エンジニアリングサービスプロバイダー，`https://www.gaio.co.jp/service/mcd2/`.

[12] Hellmunda, A.-M., Wirges, S., Ömer Şahin Taş, Bandera, C. and Salscheider, N. O.: Robot Operating System: A Modular Software Framework for Automated Driving, *In Proc. of IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)* (2016).

[13] Miura, K., Tokunaga, S., Ota, N., Tange, Y. and Azumi, T.: Autoware Toolbox: MATLAB/Simulink Benchmark Suite for ROS-based Self-driving Software Platform, *In Proc. of the 30th International Workshop on Rapid System Prototyping (RSP'19)*, ACM, pp. 8–14 (2019).

[14] Tier IV: Autoware Data, `https://data.tier4.jp/rosbag_details/?id=21`.

[15] ガイオ・テクノロジー: MC-Verifier — ガイオ・テクノロジーソフト検証ツール/モデルベース/エンジニアリングサービスプロバイダー，`https://www.gaio.co.jp/products/mcv/`.

[16] ガイオ・テクノロジー: カバレッジマスター winAMS — ガイオ・テクノロジーソフト検証ツール/モデルベース/エンジニアリングサービスプロバイダー，`https://www.gaio.co.jp/products/coveragemaster/`.

[17] Grazioli, F., Kusmenko, E., Roth, A., Rumpe, B. and von Wenckstern, M.: Simulation Framework for Executing Component and Connector Models of Self-Driving Vehicles, *In Proc. of International Conference on Model Driven Engineering Languages and Systems (MODELS 2017)* (2018).

[18] Bhadani, R. K., Sprinkle, J. and Bunting, M.: The cat

Vol.2019-ARC-240 No.46
Vol.2019-SLDM-191 No.46
Vol.2019-EMB-53 No.46
2019/2/28

vehicle testbed: A simulator with hardware in the loop for autonomous vehicle applications, *In Proc. of International Workshop on Safe Control of Autonomous Vehicles (SCAV 2018)* (2018).

[19] Naveen, M. and Martin, T.: AD-EYE: A Co-Simulation Platform for Early Verification of Functional Safety Concepts, *In Proc. of WCX SAE World Congress Experience* (2018).

[20] Wang, S. W.: Study of the Back-to-Back Test Method for Embedded Systems in Hardware-Software Integration Context, Master's thesis, KTH, Machine Design (Div.) (2013).