ホワイトペーパー

MATLAB/Simulink による 自動運転・ADAS 開発

MathWorks®

はじめに

近年、世界各国の OEM、サプライヤー、そして自動運転に特化した新興企業に至るまで、多くの企業・エンジニアが自動運転・ADAS の実現に向けて関連技術の開発に取り組んでいます。また、自動運転・ADAS を実現させるための法整備も進んでおり、公道での実証実験も開始されています。運転が自動化されることで交通事故の減少等多くのメリットが期待されています。

自動運転・ADAS を実現するために必要となる技術は多岐に渡り、開発の現場では様々な開発環境・プログラミング言語が利用されていることからシステムレベルのシミュレーション及び検証が困難になりつつあります。

その一方、テスト車両などを利用した試験はリスクを伴い、試験可能なシナリオも限られることから、シミュレーションの重要性が高まってきており、各分野の技術者が共通のプラットフォームとして利用できる統合開発環境が求められています。このホワイトペーパーでは、複雑化する自動運転・ADAS 開発の技術領域と、
MATLAB/Simulink によるソリューションを解説します。

自動運転・ADAS 開発における技術領域

自動運転はドライバーと自動化されたシステムが担う役割によって、レベル 0 から 5 まで 6 段階に分類されています。実現に必要な技術はレベル毎に異なりますが、人が運転する際に行っている周囲状況の認識、判断と運転操作の自動化を目指していく点から、自動運転に必要な要素技術としては以下 3 点を挙げることができます。

- 様々な物体の検出と物体までの距離の理解 (認識)
- 自車の位置推定と経路計画 (判断)
- 計画に沿った自車の制御 (操作)

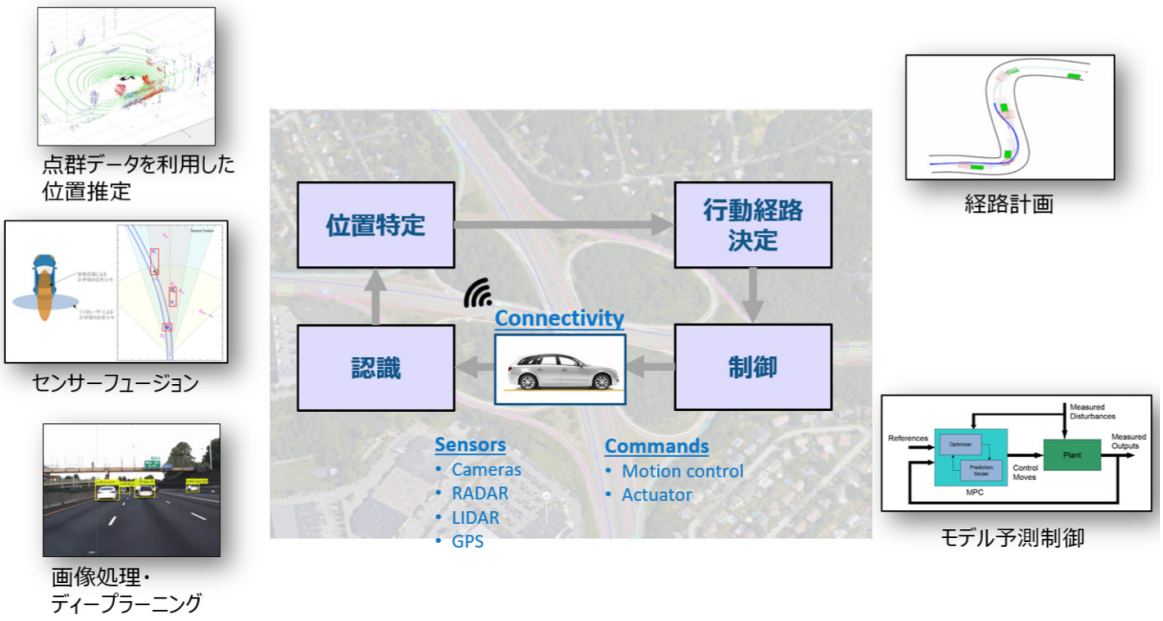

図1. 自動運転に必要な要素技術

様々な物体の検出と物体までの距離の理解 (認識)

自動運転を実現するために必要な要素技術として「認識」があります。自車周辺の環境を正しく認識するためには多数のセンサーが不可欠となります。利用される主なセンサーとしてはカメラ、レーダー、Lidar 等があり、それぞれのセンサーデータに対する処理、認識系アルゴリズムの開発が必要になります。他にも位置情報を取得するための GPS や、位置情報を補間するための加速度センサー、ジャイロセンサー等があります。

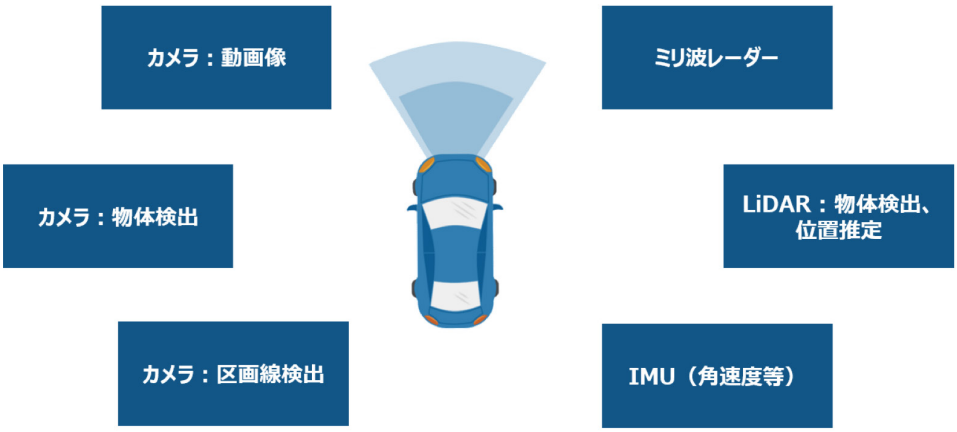

図2. 自動運転に必要なセンサーの例

これらのセンサーから得たデータに対して処理を施し、周辺環境の理解を進めていきます。例えば画像データであれば画像処理やディープラーニングを使ったシーン認識や物体認識、Lidar であれば点群データ処理による物体認識、自己位置の推定等になります。

これらのセンサーは得手不得手があります。例えばミリ波レーダーは分解能の関係で対象の形状を正確に把握することは得意ではありませんが、カメラからの画像データを利用した物体認識であれば歩行者や乗用車、トラックといった具合に精度良く物体に分類できます。このため、複数のセンサー出力結果を統合して活用するためのセンサーフュージョンが必要と考えられています。センサーフュージョンを行う場合、基本的にセンサー間の同期がとれておらず、データの次元はセンサー毎に異なることに注意する必要があります。また、センサーの特性、個数や取り付け位置に応じて最適なアルゴリズムを開発する必要があり、センサーフュージョンアルゴリズムの開発は非常に複雑なタスクの一つとなります。

また、センサーでは把握できない範囲の危険を回避するために、通信ネットワークを活用して情報の取得を行うV2X (Vehicle to X) も注目を集めています。自動車間で通信を行う V2V、自動車と路上設備で通信を行う V2I などがありますが、いずれも無線による通信となるため、マルチパスによる急峻な受信電力の変動等による誤りが発生する可能性があります。自動運転を実現するためには安定した通信方式の構築や十分な通信帯域の確保、高いセキュリティの確保が必要になります。

自車の位置推定と経路計画 (判断)

自車が走行可能な領域を把握して経路計画を設定するためには自車の位置推定が不可欠です。主に GPS を利用して位置情報を得ますが、GPS は測定誤差を含むことから、画像認識による周辺のランドマークの情報や Lidar から得られた点群の情報を利用し、推定の精度を高めます。自己位置推定のアルゴリズムとしては SLAM (Simultaneous Localization and Mapping) が著名ですが、点群データのマッチングを中心とした逐次 SLAM だけでなくノード間の拘束からなるグラフを最適化するグラフベースの SLAM や、カメラで撮影された画像を用いる Visual SLAM、さらにそれらの組み合わせ等、精度や計算コスト改善に向けた取り組みが盛んに行われています。

さらに、自車を目的地まで安全に進めていくためには、「判断」の要素が必要です。周辺環境の「認識」により得られた位置情報を基にしたグローバル・ローカルな経路計画の設定が必要となります。自車周辺の状況は時々刻々と変化することから、自車が安全に走行できるエリアを示すコストマップを作成・逐次更新し、最もコストが少なくなるような経路を選択していきます。

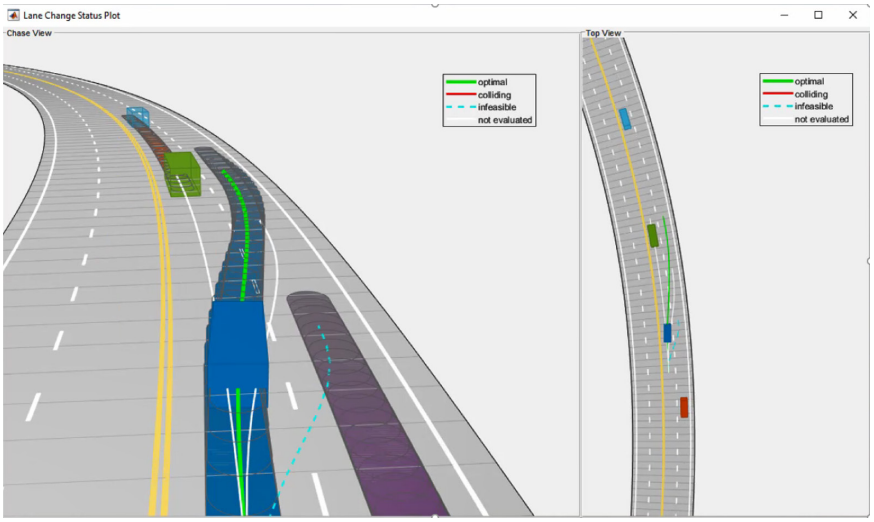

図3. ローカル・パスプランニング (青色直方体が自車、緑線が選択された経路)

計画に沿った自車の制御 (操作)

経路計画がなされた後は、その経路に沿って自車を「制御」していきますが、同乗者に不快感を与えない、滑らかな車両挙動を実現する必要があります。予め定められた軌跡を正確にトレースするためには PID に代表されるような古典制御では難しいことから、課題を克服するための先進制御としてモデル予測制御が注目を集めています。さらに近年では機械学習の一種である強化学習 (Reinforcement Learning) も登場し、自らの評価を基に適応していくアルゴリズムとして盛んに研究が行われています。

このように、自動運転を実現するためには「認識・判断・制御」と幅広い領域において検討・開発すべき多くの技術が存在しています。

MATLAB/Simulink による自動運転・ADAS 開発ソリューション

前述の通り、自動運転の領域で必要となる技術は多岐に渡っており、各分野のエンジニアが共通のプラットフォームとして利用できる統合開発環境が求められています。個々のアルゴリズム開発に優れた環境であるだけでなく、その安全性を担保するためのテスト・検証を統合して行えること、組み込み機器等への実装パスが選択できること等が非常に重要となります。

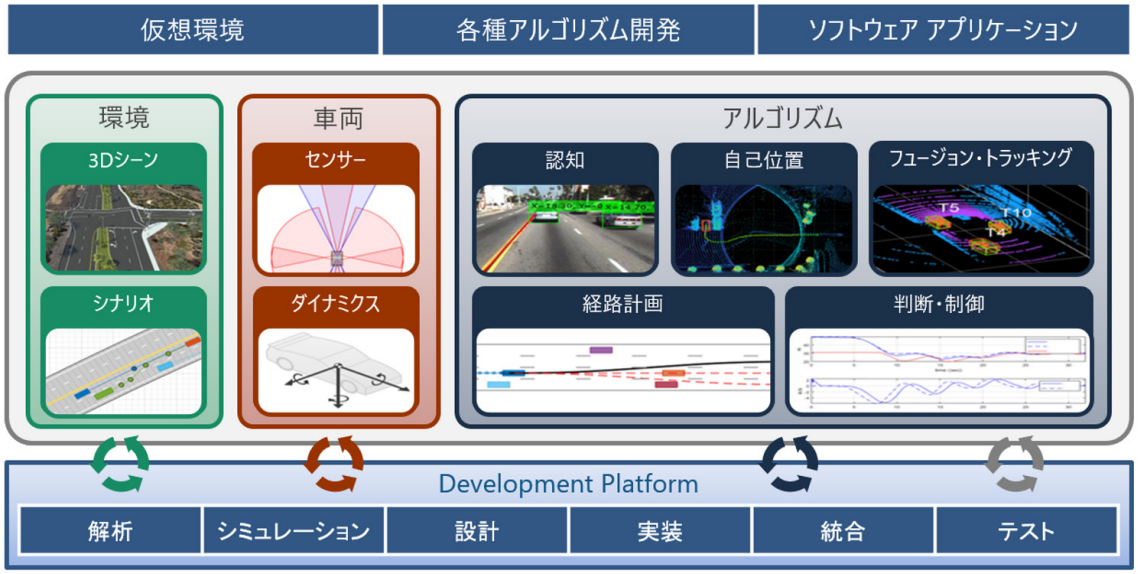

図4. 自動運転向け統合開発環境

上図は MATLAB®/Simulink® により実現できる、自動運転向け統合開発環境を示しています。このホワイトペーパーでは、この図に関連する自動運転開発のアプリケーションや機能の中から以下の重要な 7 つを取り上げ、それぞれについて MATLAB/Simulink を使用した機能・ソリューションをご紹介します。

1. 画像処理・ディープラーニング
2. Lidar 信号処理
3. センサーフュージョン
4. 先進走行制御
5. 各種アルゴリズム検証用運転シナリオ生成
6. 外部連携
7. コード生成

1. 画像処理・ディープラーニング

画像処理によるシーンの理解、物体の検出等のアルゴリズムは自動運転には欠かせないアプリケーションの一つです。画像処理の分野には多数のアルゴリズムや問題解決へのアプローチがあり、最適解を得るために様々な組み合わせを試す必要があります。

図5. 画像処理によるレーンの検出や走行可能領域の検出

生産性を向上させるための豊富なアプリ

MATLAB では豊富な画像処理ライブラリを組み合わせることにより、例えば C コード等で記述すると膨大なコード量になってしまう処理内容を、デバッグの容易な MATLAB コードで簡潔に実現することができます。

ディープラーニングについてもエンジニアや研究者の方が直感的に利用できる API や GUI ベースのアプリを利用でき、ネットワークの構築や学習を効率的に進めることができます。近年ネットワークは複雑になり、各レイヤが複数の入出力を持ち、多数の分岐があるケースも少なくありませんが、MATLAB の Deep Network Designer アプリを利用すれば、ネットワークのアーキテクチャを確認しながら、マウス操作により対話的にレイヤを選択、エディタ上にドラッグ&ドロップで展開し、必要に応じて配置変更や結線が可能です。 また、Experiment Manager アプリを利用することで、様々なデータセット・ネットワークアーキテクチャやハイパーパラメータの組み合わせによるネットワークの学習と結果比較を実行・管理することができます。

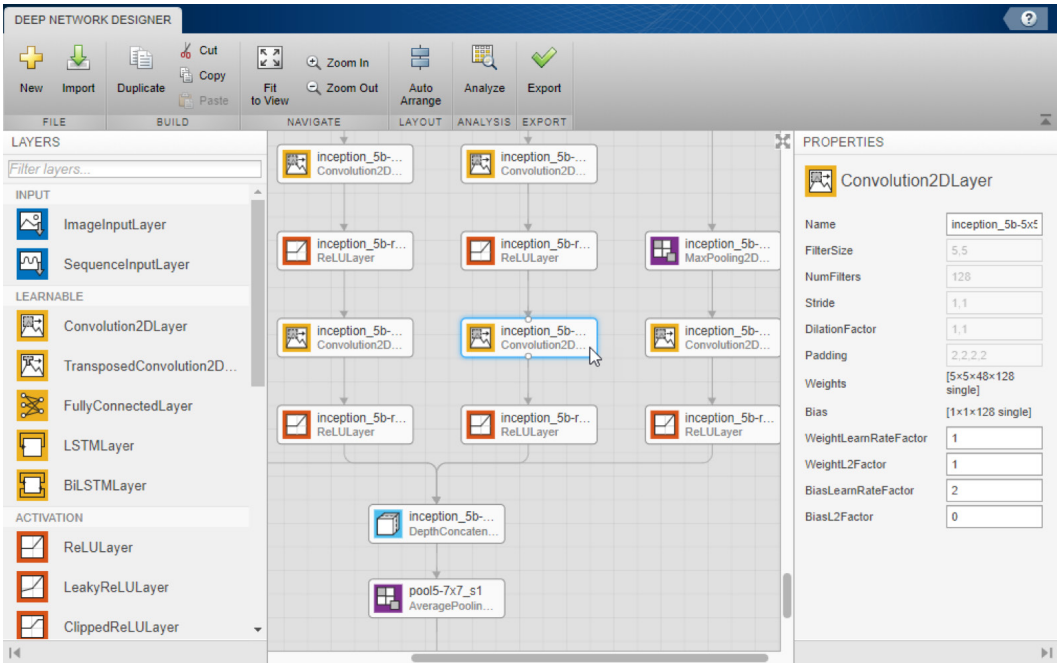

図6. Deep Network Designer でネットワークを構築、可視化、編集

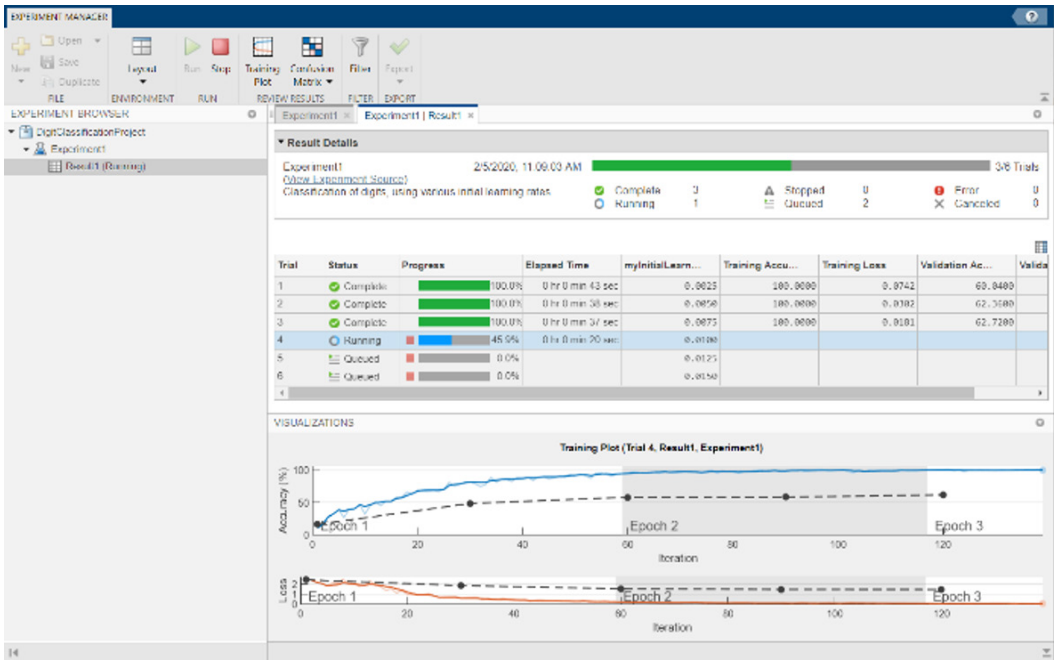

図7. Experiment Manager で様々なパラメータの組み合わせを試行

相互運用性

MATLAB は、他のフレームワークからの移行を支援するインポーターを提供しているため、既存のネットワークをインポートして利用することが可能です。Caffe や TensorFlow®-Keras については、専用のインポーターがあり、ONNX™ (Open Neural Network Exchange) と呼ばれる共通フォーマットを利用することで、他の様々なフレームワークとネットワークを共有することが可能です。

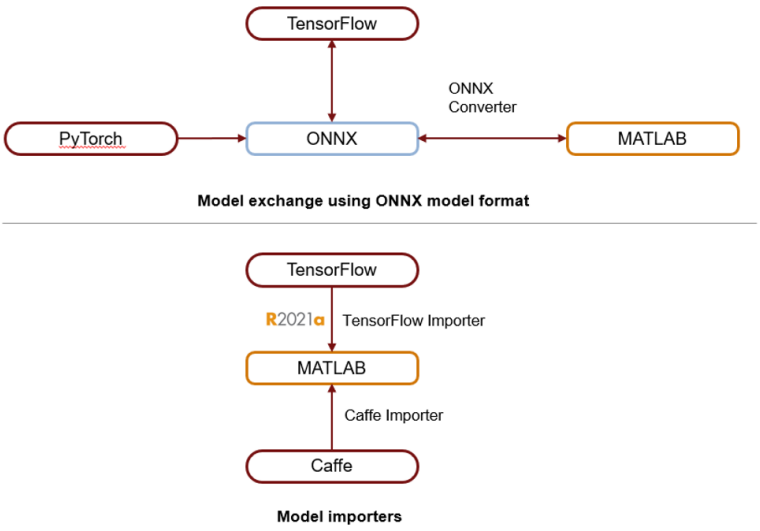

図8. インポーターを介して繋がるフレームワーク

ラベリングツール

ディープラーニングによるアプローチを試行するためには、大量の画像データとタスクに応じたラベリングが必要になります。ラベリングの精度は、最終的に学習されるネットワークの精度に大きく寄与するため非常に重要です。しかしながら、専門性の高い分野においては作業者個人のスキルに依存してしまうことが多く、また大量の画像データを扱うために膨大な時間と手間を要することが課題となっています。このような場合にはラベリングツールが有効です。MATLAB には、静止画用の Image Labeler、動画用の Video Labeler や動画と点群データ等の複数のセンサーデータが扱えるGround Truth Labelerがあり、各ツールには Grab Cut による半自動ラベリングなど作業を効率に行うための機能が備わっています。また、ユーザー側で定義したラベリング用のアルゴリズムを追加して利用することもできます。

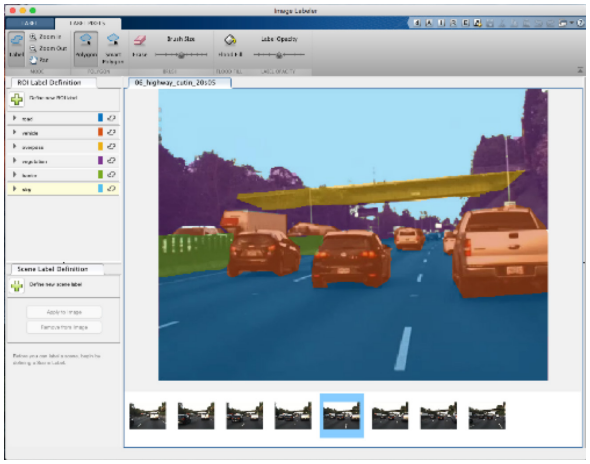

図9. 静止画・動画用ラベリングツール

関連リソース

- 今からでも遅くない！*AI*・ディープラーニング基礎と実践 – Webセミナー
- *GAN*や*VAE*で加速する画像ディープラーニング – Webセミナー
- ディープラーニング：*YOLOv2*による物体検出と追尾 – Webセミナー
- 機械学習・ディープラーニングのラベリング作業を半自動化 – Web セミナー

2. Lidar 信号処理

Lidar は、自動運転の実現において欠かせないコアセンサーの一つです。Lidar の特徴は、3 次元観測が可能であり、従来の電波を用いたレーダーに比べて光束密度が高く、短い波長を用いることでより正確な物体検出が可能な点です。

Lidar はこれまで主に地質学や気象学の分野で活用されていましたが、その精度の高さから自動運転・ADAS の分野でも注目を集めるようになりました。一方、そのような背景から Lidar を長く扱ったことのあるエンジニアが不足する傾向があり、異なる分野 (画像や制御等) のエンジニアが必要に迫られて Lidar を扱うケースも少なくありません。

MATLAB では Lidar 関連の豊富なライブラリや様々な可視化機能を扱うことができ、Lidar 信号処理に慣れていない方でもすぐに解析に着手できます。

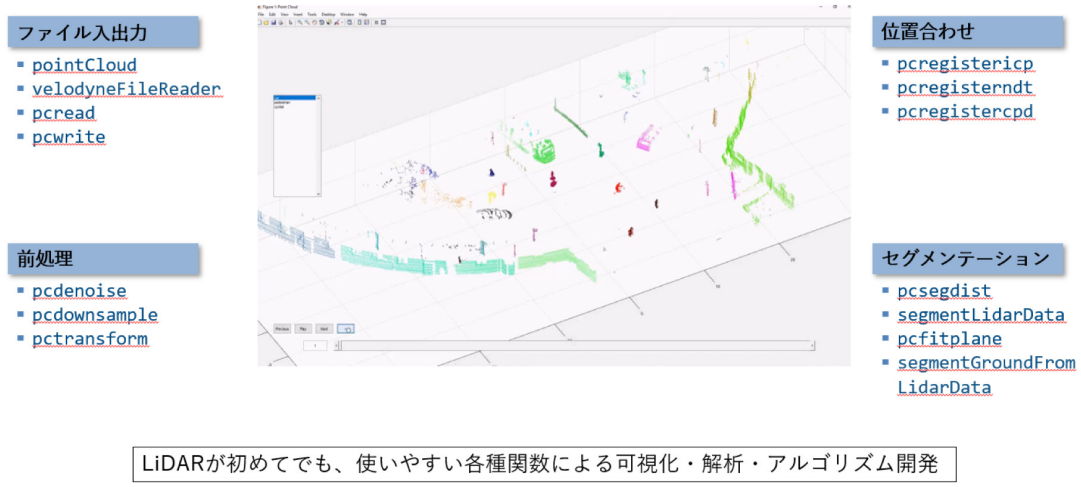

図10. Lidar Toolbox、Computer Vision Toolbox™ で利用できる点群処理関数群

Lidar はその測定精度の高さから他の測定結果の検証用として利用されるケースも多く、画像認識結果やセンサーフュージョン結果の精度を評価するための基準として活用されたり、また自己位置推定の精度を評価するための地図作成のベースとして用いられることもあります。MATLAB は様々なアプリケーションに特化したライブラリをツールボックスとして提供しています。点群データからの物体認識であれば Deep Learning Toolbox™ を利用して分類用のネットワークを作成、SLAM 等の位置推定に使いたいのであれば Navigation Toolbox™ で提供されている関数群と組み合わせて実現が可能となるなど、活用の幅を拡張していくことができます。

下図は点群データをリファレンスとして利用したもので、画像処理で認識した物体の位置を鳥瞰図変換した画像の上に重ね、さらにその上から点群データを座標変換して重畳したものです。車両後部のバンパー形状が確認できる点群の位置と画像による認識結果が概ね一致していることが確認できます。Automated Driving Toolbox™ を利用すれば、画像データの鳥瞰図変換、カメラ座標系と車両座標系の変換が簡潔な記述で実現できます。

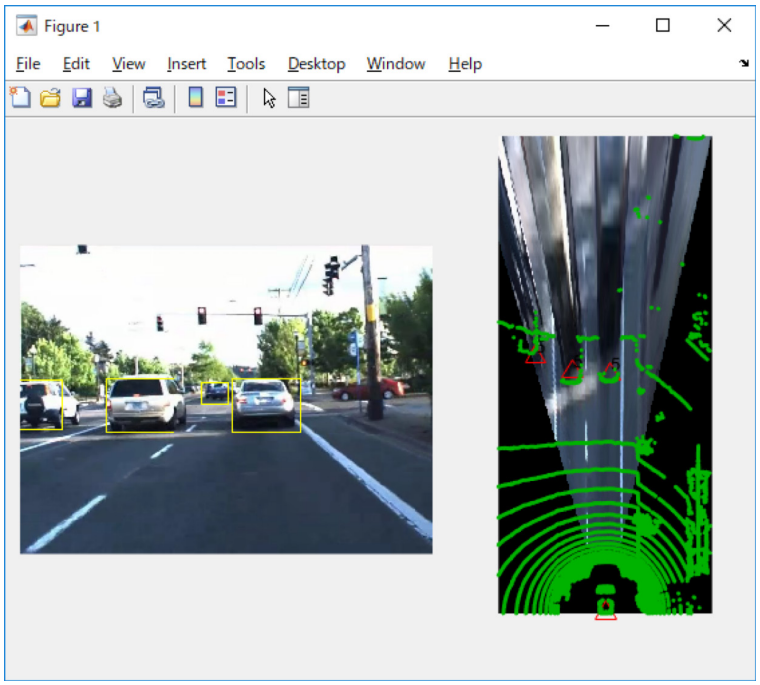

図11. 画像処理による物体検出結果と点群データの比較

また、Lidar点群データに対するラベリングを行いたい場合には、Lidar Labeler, Ground Truth Labelerを利用することで対話的ラベリングが可能となっています。点群データのクラスタリングや地表面の除去等の前処理がツール上で可能となっており、機械学習・ディープラーニングによる分類に必要なラベリングを支援します。Image Labeler, Video Labeler同様、ラベリングを半自動化するためのカスタムアルゴリズムの追加も可能となっています。

さらに、Lidar以外にカメラを用いてフュージョン等を実現する場合、Lidarとカメラのクロスキャリブレーションを実行する必要があります。Lidar Toolbox™ではカメラとLidar点群から自動的にチェッカーボードを検出する関数やセンサー間の座標変換行列を推定するための関数が準備されていますので、それらの関数の組み合わせでLidarカメラキャリブレーションを実現できます。

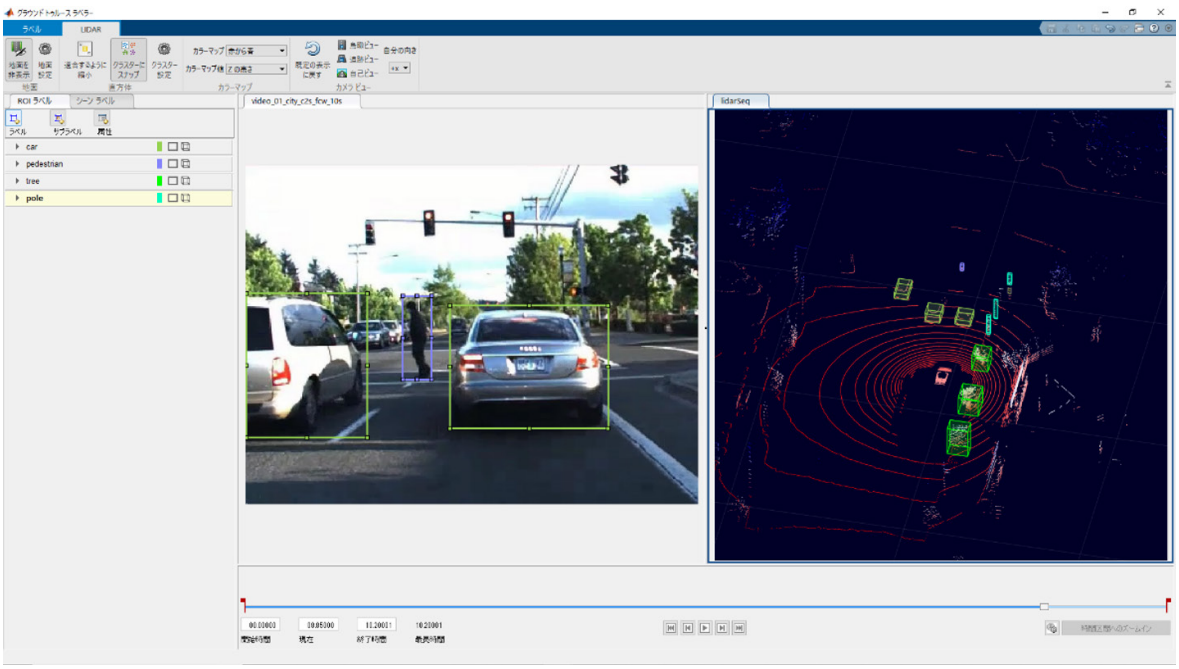

図12. Ground Truth Labeler(Multi-Signal対応)

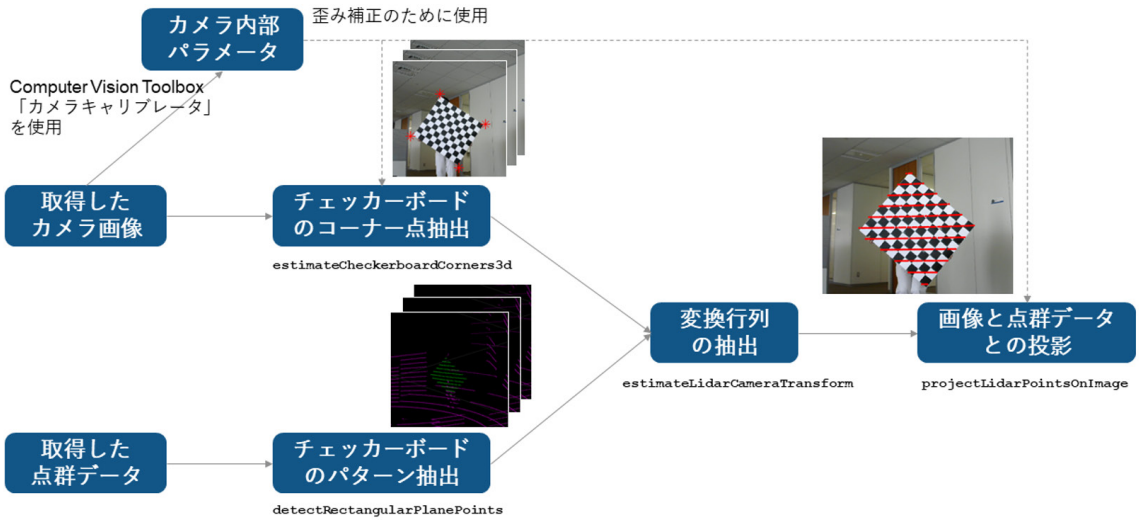

図13. Lidarカメラキャリブレーション

関連リソース

- 自動運転・*ADAS* の開発・検証プラットフォーム：*Automated Driving Toolbox* – Web セミナー
- センサー検証のための *Lidar* 点群アノテーションの自動化 – Web セミナー
- ディープラーニングを使用した *Lidar* 点群からの物体検出– Webセミナー

3. センサーフュージョン

自動運転・ADAS を実現するため、様々なセンサーが用いられるようになってきていますが、個々のセンサーの出力を統合し、単一のセンサーでは得られなかった情報を構築するアルゴリズム - センサーフュージョンの重要性が高まっています。

以下の図は、画像処理による物体検出が持つ誤差とレーダーによる誤差を比較したものです。それぞれ異なる特性を持っているため、フュージョンすることによってそれぞれの欠点を補い、精度を向上できる可能性があることがわかります。

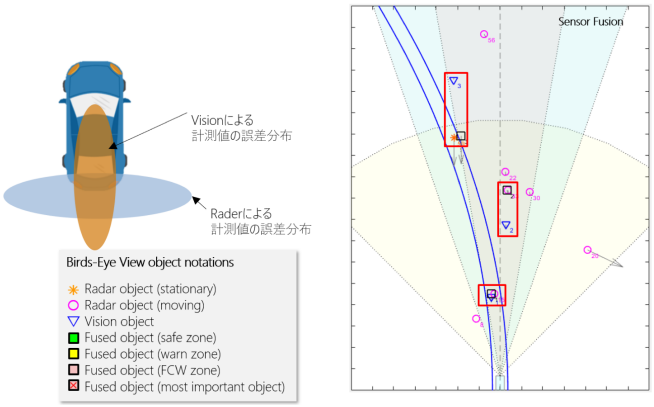

図14. センサーフュージョンの重要性

一方、センサーフュージョンを実現するためには、複数のセンサ出力の組み合わせの選定やその管理、カルマンフィルタによるトラッキングなどを組み合わせる必要があり、複雑なデータの割り当てと管理を記述する必要が出てきます。Automated Driving Toolbox では、高抽象度オブジェクトである multiObjectTracker が提供されており、センサー出力を入力としてオブジェクトに与えるだけでフュージョンを実現できます。

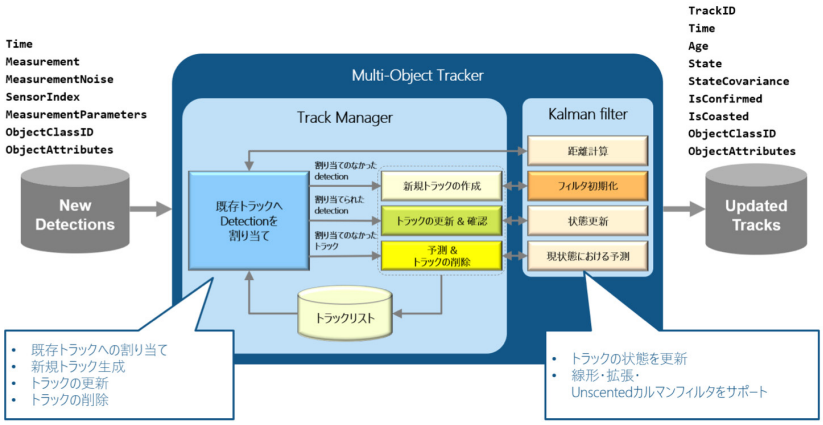

図15. Multi-Object Tracker 内部処理

さらに、Sensor Fusion and Tracking Toolbox™ を利用することで、JPDA (Joint Probabilistic Data Association) 方式による相関処理や複数の運動モデルを用いる IMM (Interaction Multiple Model) 法などが選択できるようになり、目的とするタスクに対して様々なアルゴリズムの組み合わせで最適解を探索できます。本Toolboxではトラッキングのパフォーマンスを解析するためのメトリクスも複数提供されていますので、定量的に各種アルゴリズムの比較を行いたい場合にも活用することができます。

センサーフュージョンについてこれから開発を着手される方、自身の専門分野ではないが取り扱う必要のある方は、Sensor Fusion and Tracking Toolboxに含まれる豊富な例題で使用例を学ぶことができます。自動運転領域のアプリケーションもカバーされており、IMUセンサーモデルとビジュアルオドメトリによる自己位置推定の例題、2D Radar、3D LidarのTrackレベルでのフュージョン例などが含まれます。

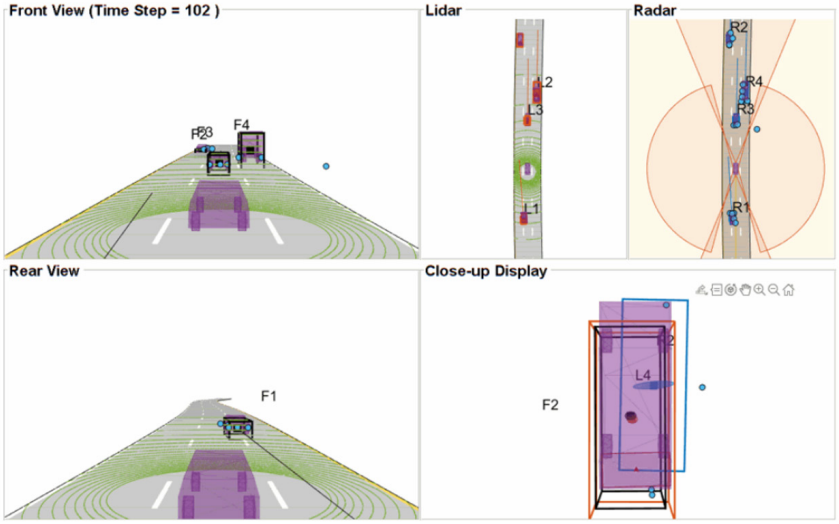

図16. 2D Radar, 3D Lidar Trackレベルフュージョン

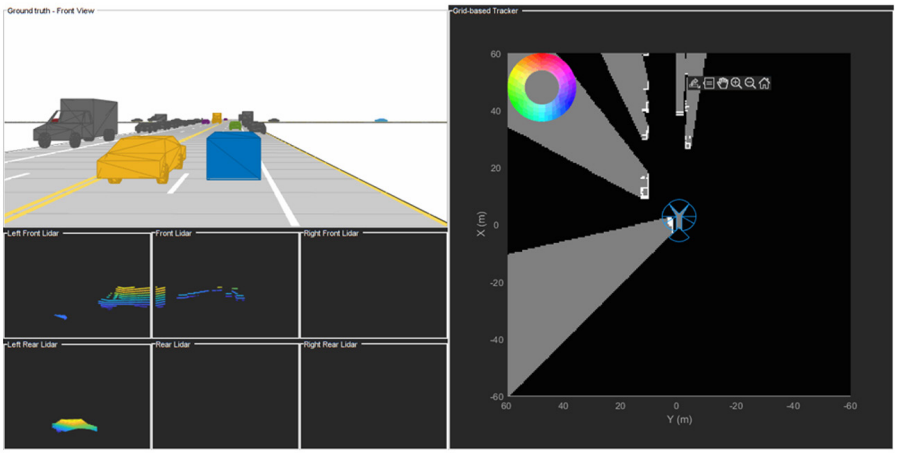

図17. Gridベースのトラッキング(環境の表現に動的占有マップを利用)

関連リソース

- センサーフュージョンアルゴリズムの開発と検証 – Web セミナー
- 自律移動システムのためのセンサーフュージョンとナビゲーション – Webセミナー
- *ADAS*/自動運転システム開発におけるモデルベースデザイン最新動向 – Webセミナー

4. 先進走行制御

モデル予測制御

自動運転においては安全・安心と快適性の両立を実現するような、より人間らしい滑らかな制御を目指す必要があります。PID や LQR のような従来の制御手法と比べ、より高応答・高精度かつ適合工数の削減を実現するような手法が求められます。理想的には人間的な運転を再現できることが望ましく、そのアプローチの一つとしてモデル予測制御 (MPC) が注目されています。モデル予測制御は制御対象の未来の挙動を観測とモデルに基づいて予測し、所定の評価関数の下で最適な制御入力を求め、実時間で実行する制御手法です。

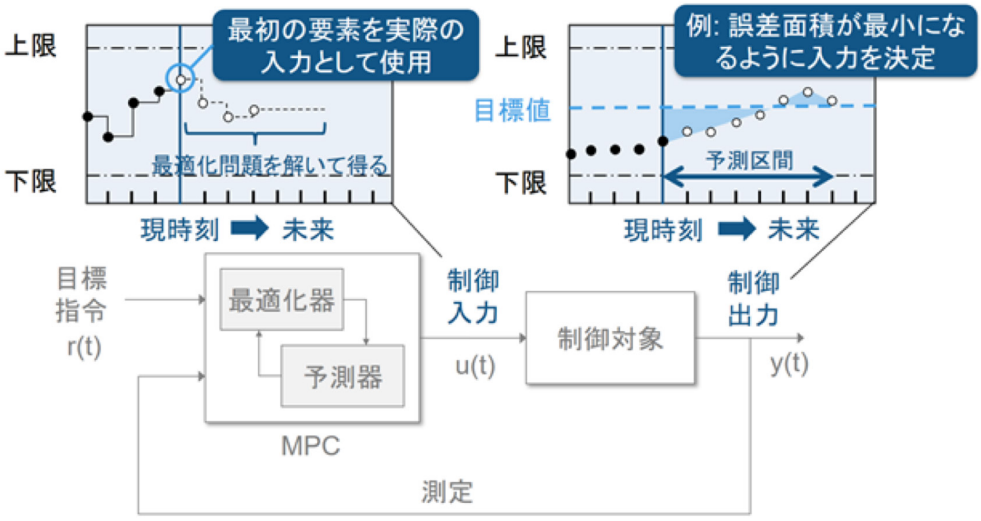

図18. モデル予測制御の仕組み

MATLAB におけるモデル予測制御は、設計やシミュレーションに関して Model Predictive Control Toolbox™ という専用のツールボックスが用意されています。本ツールは 20 年以上の長い歴史を持ち、著名な研究者 (Prof. M. Morari, Prof. N. Ricker, and Prof. A. Bemporad) と共同開発されており、主に次のような機能を提供しています。

- 専用の数値最適化 (QR) ソルバー
- 設計・シミュレーション用のユーザーインタフェース
- Simulink 用の MPC コントローラブロック

最適化問題の定式化やソルバーはツール側で準備されているため、予測モデルを設計し、必要なパラメータ (予測区間、目的関数の重み、制約条件等) を設定すれば、すぐにシミュレーション実行が可能です。モデル予測制御の性能評価にいち早く到達できるのが本ツールボックスの大きな利点です。本ツールボックスを利用して、モデル予測制御をアダプティブクルーズコントロールに適用した事例をはじめ、詳しくは関連リソースも併せてご覧ください。

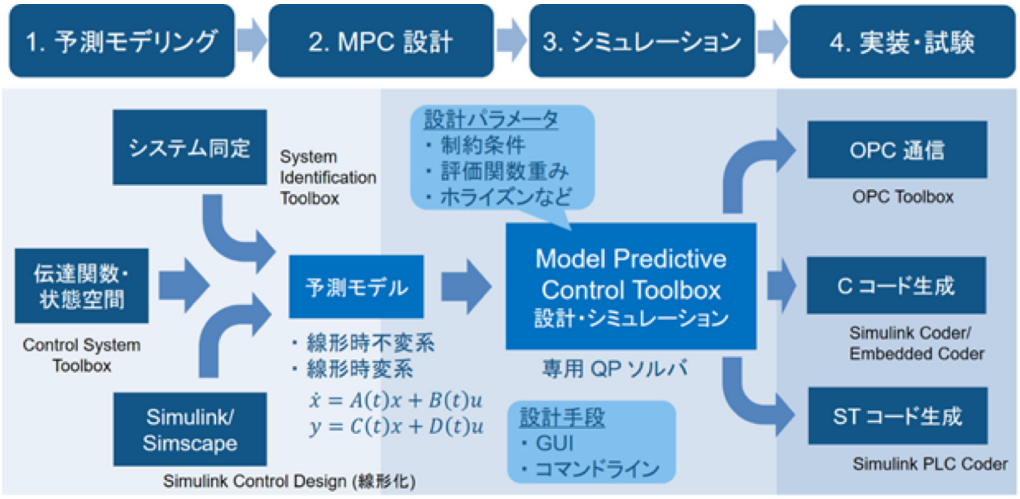

図19. MPC 設計のワークフロー

強化学習 (Reinforcement Learning)

モデル予測制御とは異なる手法として、機械学習の一種である強化学習 (Reinforcement Learning) によって、人間が行うような自然な運転を模倣させようというアプローチも存在します。強化学習は、特定の環境下において一連の行動に報酬と呼ばれるインセンティブを与え、自律的に適切な行動をとるように新しい環境への適用を教えていくもので、従来の手法とは一線を画すアプローチとしても注目を集めています。強化学習専用のツールボックスである Reinforcement Learning Toolbox™ では、DQNやA3C/PPO/SAC等の学習アルゴリズムを試すことができます。強化学習を試す場合には、制御対象となるプラントのモデリングや報酬関数の定義が必要になりますが、Simscape™ 等で提供される物理コンポーネントを利用することで直感的にプラントのモデリングが可能であり、MATLAB/Simulink が有する豊富なライブラリ群を利用して柔軟に報酬関数を定義できます。

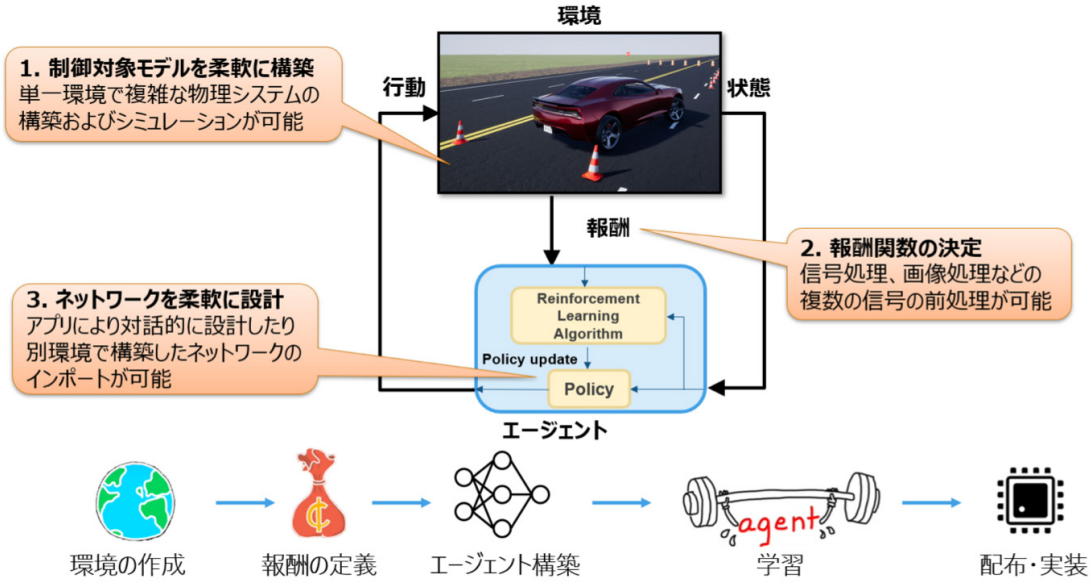

図20. 強化学習ワークフロー

関連リソース

- モデル予測制御 *(MPC)* の応用 ~アダプティブ・クルーズ・コントロールとセンサーフュージョン – Web セミナー
- 強化学習ビデオシリーズ – Web セミナー
- モデル予測制御による渋滞にも使える実用的な*ACC*の開発 – テクニカルペーパー
- 日立オートモーティブシステムズ、モデルベースデザインによる車間距離制御装置*(ACC)*用のモデル予測コントローラーを開発 - ユーザー事例
- トヨタ自動車、モデル予測制御を用いた「上手い運転」を実現する自動運転制御 – MATLAB EXPO 2020 講演資料
- ***Model Predictive Control Toolbox :*** 自動運転アプリケーション例 - 製品例題
- 【強化学習入門】複雑な強化学習を*MATLAB x Simulink*で簡単に！ - Webセミナー

5. 各種アルゴリズム検証用運転シナリオの生成

様々なセンサー、フュージョンを含む新しいアルゴリズムの登場により検証が必要となる項目は劇的に増加し、シミュレーションの必要性は益々重要視されています。コーナーケースでのアルゴリズム検証、異なる特性のセンサーの組み合わせや配置位置の検討など、より多くのテストケースバリエーションで検証を行うためには、いかに容易にテストシナリオを定義できるかが重要です。

近年、Photo Realisticなシミュレーション環境が注目を集めていますが、3Dシーンの作成や取り扱いには専門的な知識・多くの工数が必要となります。また、センサーフュージョンや制御系エンジニアなどは物体検出結果が得られれば良く、Photo Realisticなシーンを必ずしも必要としていないケースも多く存在します。Automated Driving Toolbox ではそのようなエンジニアの方が検証を実施するのに十分な環境、Driving Scenarioを提供しています。3D CGに関する知識は不要で、Driving Scenario Designer アプリによりマウス操作で道路やアクターの定義、センサーの配置などを直感的に行うことが可能です。

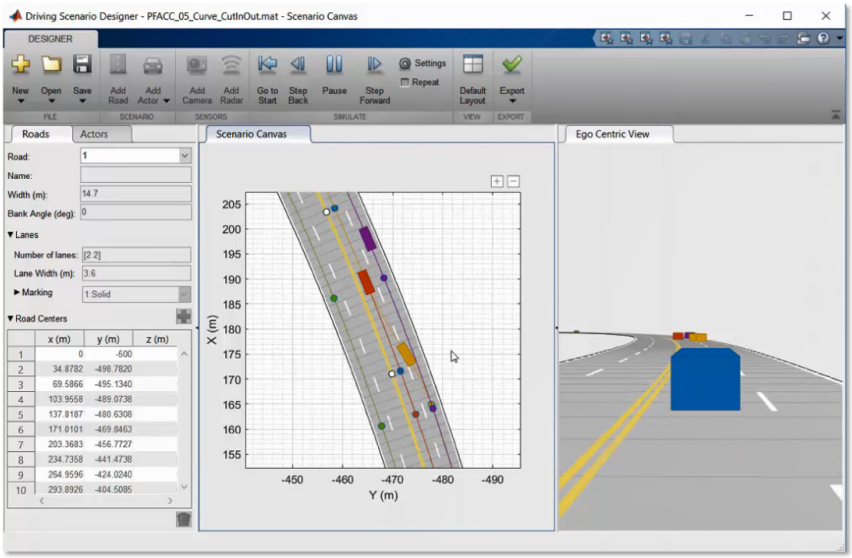

図21. Driving Scenario Designer

本アプリ上で作成したシナリオは、MATLAB 関数としてエクスポートが可能で、MATLAB のコマンドラインからの再修正や MATLAB Function Block として Simulink への統合が可能です。

また、道路の定義については地図データベースを利用することで半自動化することも可能です。MATLAB は Web サービスからコンテンツを読み取ることができるため、オープンな地図データベースにアクセスして道路情報を取得し、Driving Scenario Designer にインポートして利用できます。テスト車両等実車から取得した GPS のデータや Lidar 点群データがあれば、GPS データを基準として周辺の道路情報を地図データベースから取得できます。点群データからは、物体認識やフュージョンのアルゴリズムを組み合わせることで、交通参加者の軌跡を抽出することができるため、シナリオ作成自体を半自動化することも可能です。アクセス可能な地図はオープンなデータベースに限定されておらず、HERE 社が提供している HD Live Map やゼンリンデータコム社が提供されているいつもNAVI 等にもアクセスができ、より詳細な道路情報 (車線規制や制限速度等) を活用し、シナリオのバリエーションを増やすことができます。 また、 作成したシナリオはASAM OpenSCENARIO®フォーマットでエクスポートが可能、 道路情報はOpenDRIVE®形式でインポート/エクスポートすることが可能となっています。

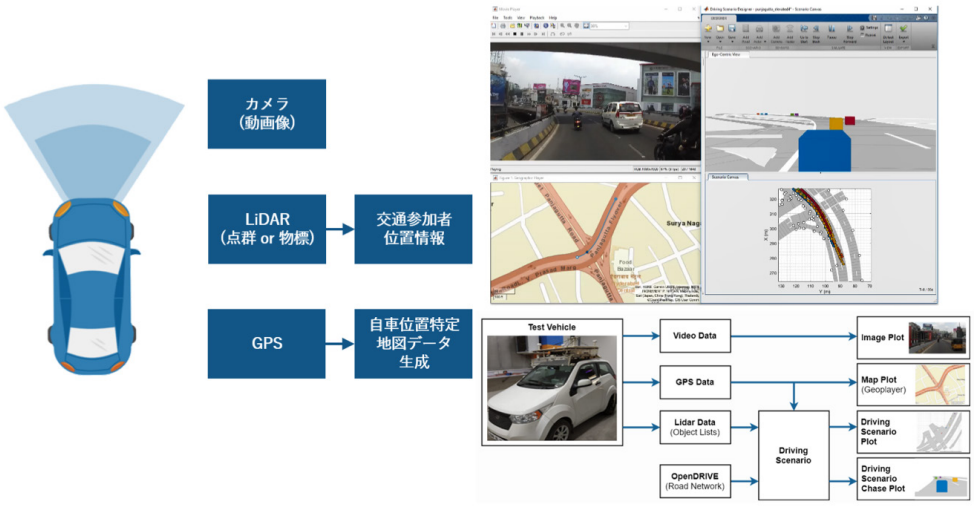

図22. テスト車両から取得したデータに基づくシナリオ自動生成

Driving Scenario Designerで利用できるセンサーモデルはカメラ、Radar、LidarとINSセンサーの4種類となり、カメラ、Radarに関しては統計的なセンサー特性に基づいてオブジェクトリストを出力、Lidarに関しては理想的な点群データを出力するようなモデルとなります。オブジェクトリストで出力されるためセンサーのRawデータ処理は不要で、センサーフュージョンや制御系のアルゴリズム開発者の方が必要とする粒度の情報が返却されます。

一方、センサーのRawデータを取得したい場合や物理現象をある程度考慮したセンサー出力を希望する場合はカスタマイズも可能となっています。下図はRadarセンサーにおけるマルチパスの影響を再現した例ですが、Radar Toolboxで提供されている関数群の組み合わせでノイズや反射の影響を考慮したセンサー出力を得ることができ、ガードレールによるゴーストの影響等をシミュレーションできます。

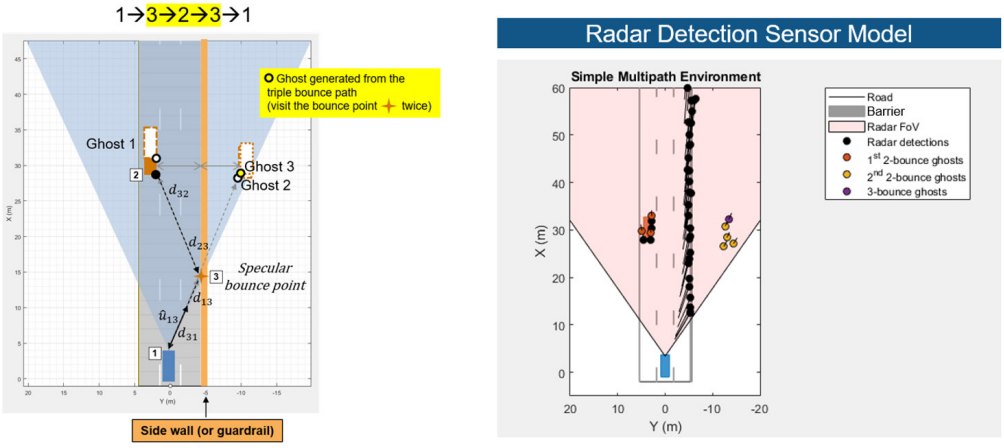

図23. マルチパスの影響を考慮したシミュレーション

関連リソース

- 自動運転開発関連機能入門 – MATLAB EXPO 2019 講演資料
- 交通シナリオにおける意思決定、経路計画、制御モジュールの設計とテスト – Web セミナー
- *Driving Scenario Designer* – Web セミナー
- *ADAS*/自動運転システム開発におけるモデルベースデザイン最新動向 – Webセミナー

6. Unreal Engine® • ROSとの連携、3Dシーン作成

Unreal 連携

先に紹介した Driving Scenario Designer で利用できるセンサーモデルは、簡易的表現を利用しており、統計的なセンサー特性に基づいて簡易シミュレーションを行います。多くのケースではこの簡易シミュレーションで十分ですが、よりリアルな環境と物理センサーモデルを用い、より現実に近いシミュレーションを行いたいケースも存在します。

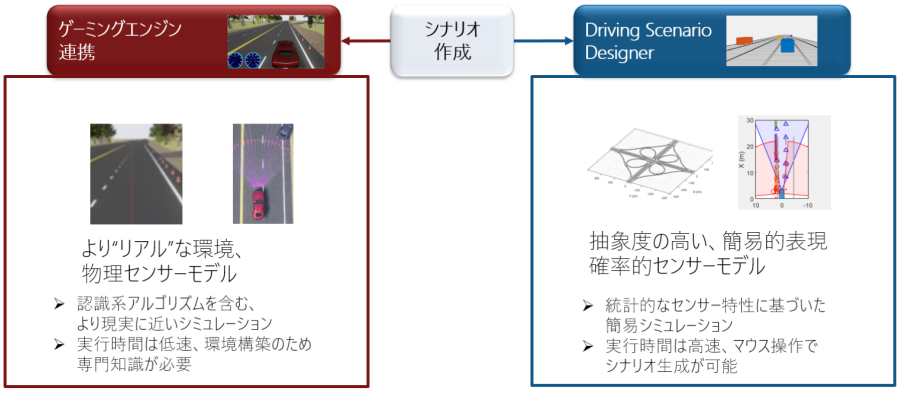

図24. 目的に応じた環境の選択

このようなケースでは、Simulink + Unreal Engine を連携させたシミュレーションの環境構築が有効です。Automated Driving Toolbox及び Vehicle Dynamics Blockset™ は、仮想 3 次元環境 (Unreal Engine®) との連携シミュレーションを行うための機能を備えており、Unreal 上に配置された車両の制御、センサーからのデータの取得等、双方向通信を可能としています。センサーから取得したデータは MATLAB/Simulink 上で処理することができ、例えば画像であれば画像処理やディープラーニングによる物体検出等のアルゴリズムを実際に実行し、その結果を基に判断・制御まで行うようなクローズドループのシミュレーション環境を構築することも可能です。

Unreal Engineとの連携の部分については“3D Scene Configuration”と呼ばれる専用のSimulinkブロックが設けられており、使用するシーンの選択、Viewerの視点の切り替えや天候の設定等がSimulink上から可能となっています。また、センサーについてはカメラ、Radar、Lidarといったセンサーモデルを選択可能で、取り付け位置やセンサー仕様等もSimulink上から設定を行うことができます。

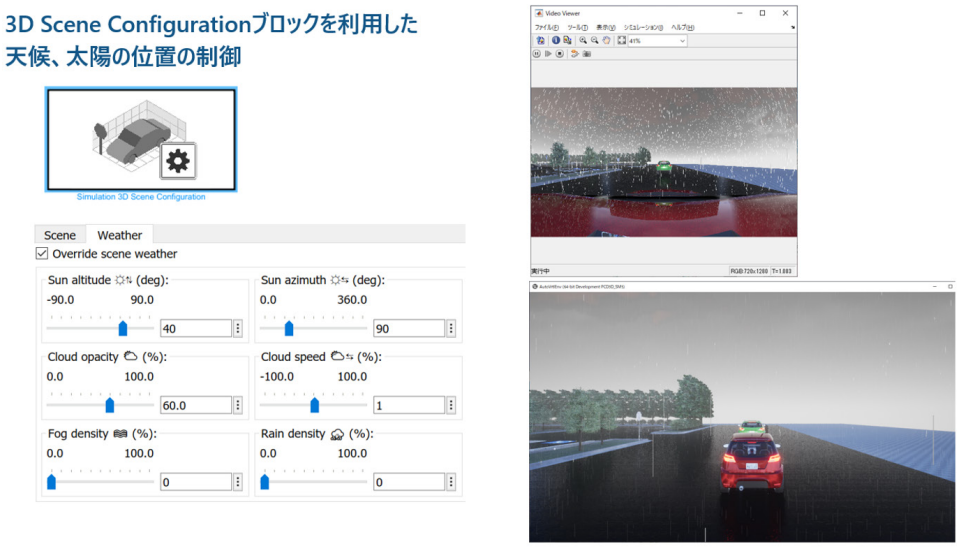

図25. Unreal Engine との連携で天候の影響を確認

3Dシーン作成

Unreal Engineを利用する場合、シーンの作成は大きな課題となります。シーン作成には殆どのケースでビジュアルスクリプトが利用されますが、このようなスクリプトに習熟したエンジニアの協力を得られる状況にある方は非常に稀であり、既存のエンジニアの方が直感的に利用できるシーンエディタが望まれています。

RoadRunnerは高度な表現力を持つ3D環境・道路ネットワークエディタであり、CGエンジニアでなくとも直感的なGUI操作でシーン作成が可能となっています。作成した3D環境は様々な形式で外部にエクスポートすることが可能で、Unreal Engine等でのシーン構築に利用できます。さらに、オプション製品であるRoadRunner Scene Builderを利用すれば、HD Mapにアクセスして所望のエリアの道路情報を取得し、自動的に道路ネットワークを構築することも可能です。

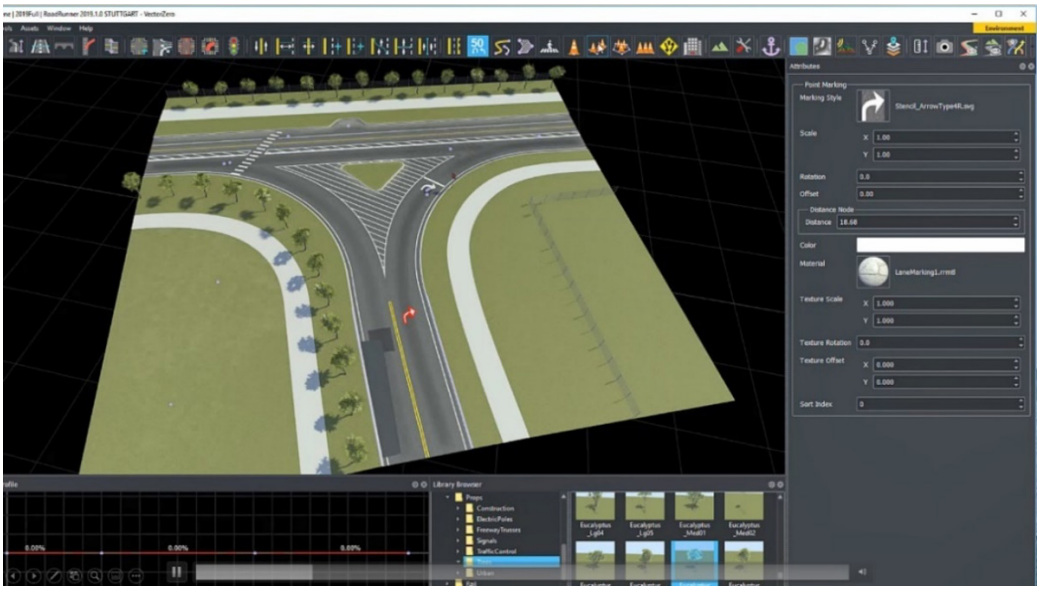

図26. RoadRunner(3D環境・道路ネットワークエディタ)

ROS 連携

自動運転の分野では、オープンソースソフトウェアである ROS (Robot Operating System) を活用した各機能モジュールの開発も広く行われています。3D のバーチャルシミュレータ等も ROS のインタフェースを有するものが数多く存在しており、自動運転システム用オープンソースソフトウェアとして著名な Autoware も ROS ベースとなっています。

MATLAB の Robotics System Toolbox™ により、MATLAB に ROS の IO インタフェースを追加することができます。これにより MATLAB を ROS ノードとして ROS マスタに登録し、他のノードと通信することや、MATLAB を ROS マスタとして機能させることも可能です。上述した Autoware についても ROS を介して連携が可能で、ROS ノードサンプルもユーザーコミュニティサイト上に公開されています。

また、rosbag 形式で保存されたデータの解析も MATLAB 上で可能であるため、ROS ベースの自動運転開発プラットフォームと連携してアルゴリズムの開発を進めることもできます。

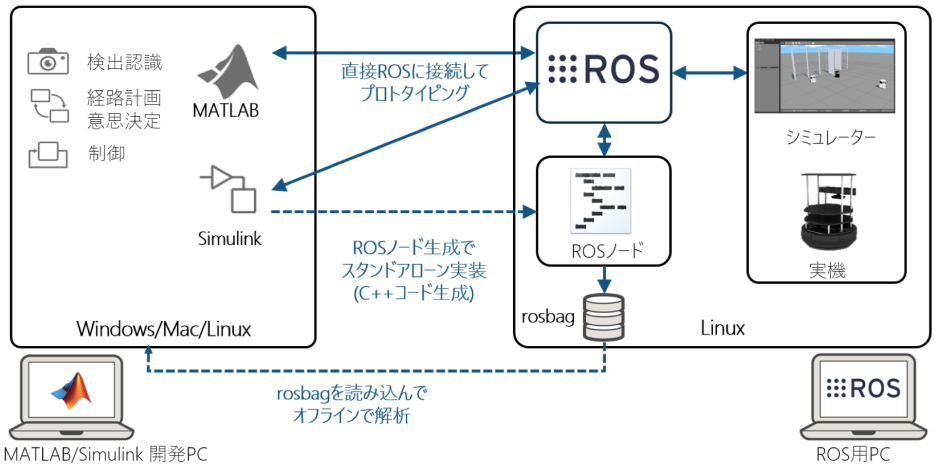

図27. MATLAB/SimulinkとROS連携

関連リソース

- リアルタイムのドライバーインザループシミュレーション – Web セミナー
- *ROS* とつながる *MATLAB* – Web セミナー
- *MATLAB* ではじめる自立移動システム開発 ~ *SLAM* とパスプランニング ~ - Web セミナー
- *ADAS*/自動運転システム開発におけるモデルベースデザイン最新動向 – Webセミナー
- *ADAS*/自動運転：*Unreal Engine*を活用した自動運転システムの開発 – Webセミナー
- *RoadRunner* 入門 – ビデオシリーズ

7. コード生成

MATLAB/Simulink 上で開発・検証された各種アルゴリズムは、コード生成関連のツールを利用することで CPU/GPU/FPGA 等の組み込みデバイス上に実装することが可能です。ラピッドプロトタイピングや量産向けコードのリファレンスとして利用できます。コード生成系の製品としては、

- MATLAB Coder™ – C/C++ コード生成
- HDL Coder™ – VHDL®/Verilog® コード生成
- GPU Coder™ – CUDA C/C++ コード生成

などがあります。

自動運転で用いられるアルゴリズムは、ディープラーニング等計算コストが高いものも少なくなく、高い計算能力を有するプロセッサを利用するケースも増えてきています。自動運転では特に消費電力が重要視されることから、GPU のようなデバイスの利用については静観されていたところがありましたが、近年低消費電量のチップや自動運転向けプラットフォームが登場したことにより、量産への適用も進みつつあります。

一方、GPU のようなデバイスを利用するためには CUDA 等の文法・言語を扱う必要があり、アルゴリズム開発者が主に利用する言語 (MATLAB 等) とのギャップが問題でした。従来は多くのケースで MATLAB から CUDA 言語への書き換えが行われていましたが、GPU Coder™ を利用することにより MATLAB 言語から CUDA C++ 言語への自動変換が可能です。

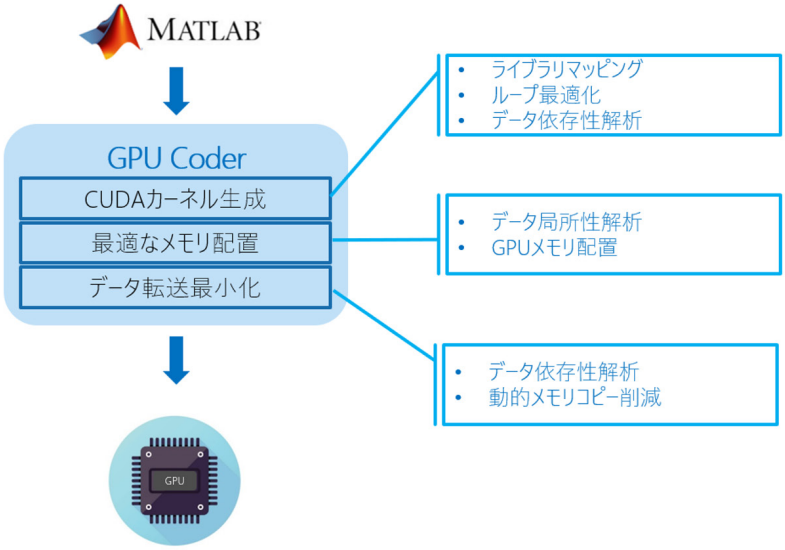

図28. GPU Coder 概要

さらに、NVIDIA GPU 向けサポートパッケージと組み合わせて利用することで、NVIDIA® Tegra®, DRIVE プラットフォームに接続されたセンサーからのデータ取得、GPU Coder で生成したコードのコピーとリモートビルト等が MATLAB 上で完結できるようになります。

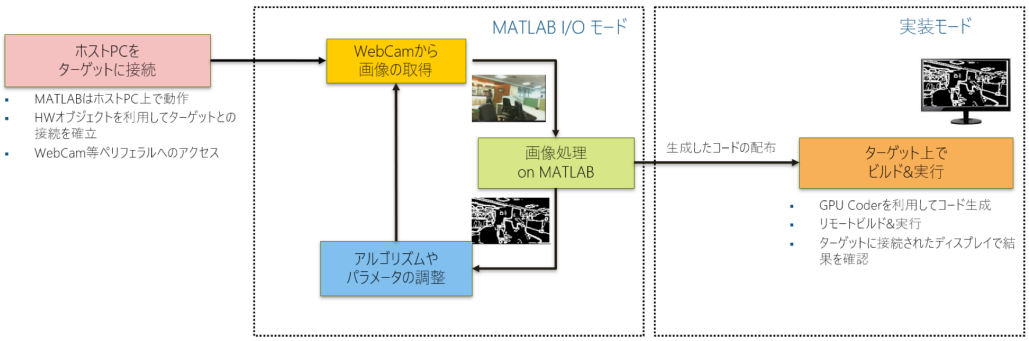

図29. Support Package を利用した NVIDIA GPU へのアクセス

GPUではなくFPGA/ASICをターゲットデバイスとして選択したい場合にはHDL Coderが利用できます。HDL Coderは300を超えるSimulinkブロックやStateflow®チャートから、コンパイラ依存のないトレーサブルなVHDL/Verilogコードを生成することができますが、通信や画像処理等特定のアプリケーションについてはVision HDL Toolbox™等のオプション製品が提供され、高抽象度ブロックからのコード生成が可能となっています。また、ディープラーニングのFPGA/ASIC実装についてはDeep Learning HDL Toolbox™という専用のオプション製品が提供されており、ディープラーニングのネットワークの量子化や実装時のパフォーマンス解析、カスタムのIPプロセッサの生成がサポートされています。

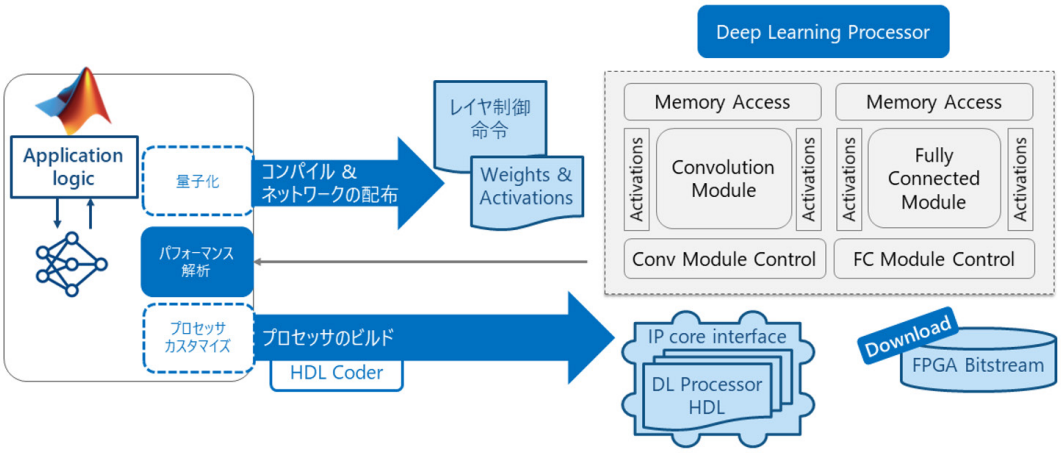

図30. MATLABからのDeep Learning プロセッサIPの生成

関連リソース

- ディープラーニングの組み込み機器実装ソリューション~*GPU/CPU*編~ - Web セミナー
- ハード/ソフト用アルゴリズムの協調設計 – Web セミナー
- ディープラーニングを組み込み*CPU*、*GPU*、*FPGA*に展開 – Webセミナー

まとめ

MATLAB/Simulink で実現できる自動運転・ADAS 向け統合開発環境について、主要な機能・ソリューションをご紹介しました。本統合開発環境の特徴をまとめると、以下の 5 点を挙げることができます。

- 自動運転に必要な各種アルゴリズムの開発が同一環境で可能
- 各種アルゴリズムを結合し、シミュレーションが可能
- 自動運転で想定されるシナリオを定義でき、アルゴリズムの検証が可能
- 外部の環境やツールとの豊富な連携機能
- 開発したアルゴリズムはコード生成により配布が可能

自動運転では、実際に実車を走らせて全てのケースをカバーすることは難しくリスクも伴うことから、シミュレーションの重要性が謳われています。デバッグの容易な MATLAB コード、マルチドメインシミュレーションを可能にする Simulink、自動運転に必要な要素技術をカバーする豊富なライブラリで、「認識・判断・制御」の各アルゴリズム開発から統合シミュレーションまで同一環境で実現できます。また、コード生成機能を利用すれば外部の環境へのアルゴリズムの配布が可能で、コードの書き換えの工数削減や書き換えによるバグ混入のリスクを低減できます。MATLAB/Simulink は自動運転の時代においてもエンジニア・研究者の皆様を強力に支援します。