3Dシミュレーション環境と AIを活用したロボット開発ワークフローのご紹介

MathWorks アプリケーションエンジニアリング部　Nobuaki Fukuchi

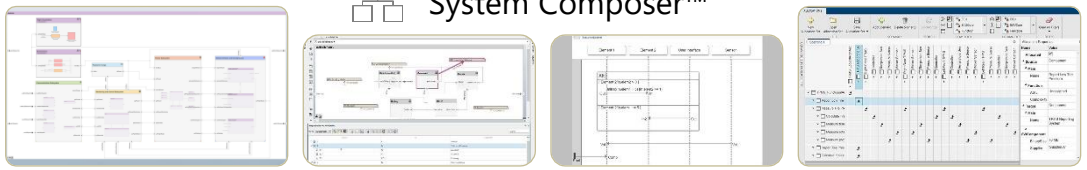
2023/09/11

自律ロボティクス開発ワークフロー

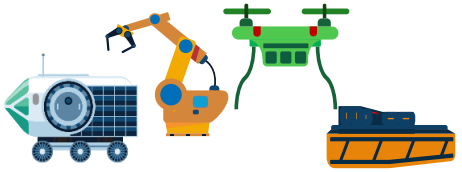
MATLAB® Simulink®

システムエンジニアリング

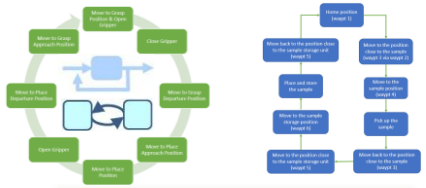
System Composer™



プラットフォーム設計

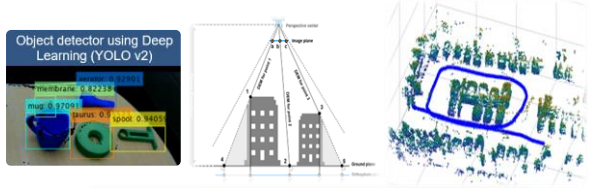


ロボットモデル

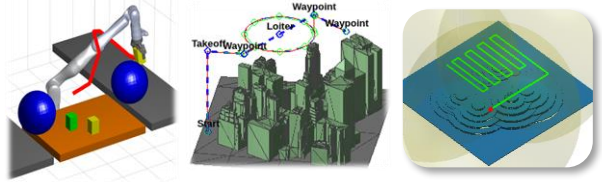


制御

AI・自律系設計

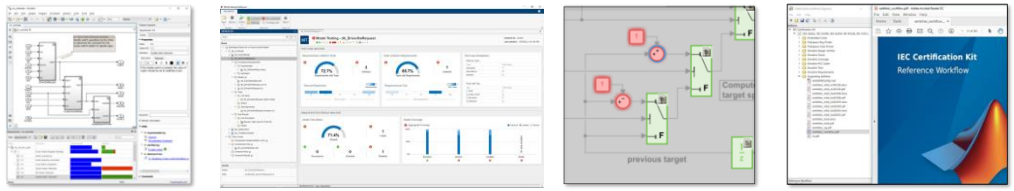


認識・自己位置推定



モーションプランニング

検証と妥当性確認



連成

テスト

実装

3Dシミュレーション

シーン・環境　　オブジェクト・アクター

ロボットとの接続

ハードウェアへの実装

Raspberry Pi™

ASIC, FPGA

PLC

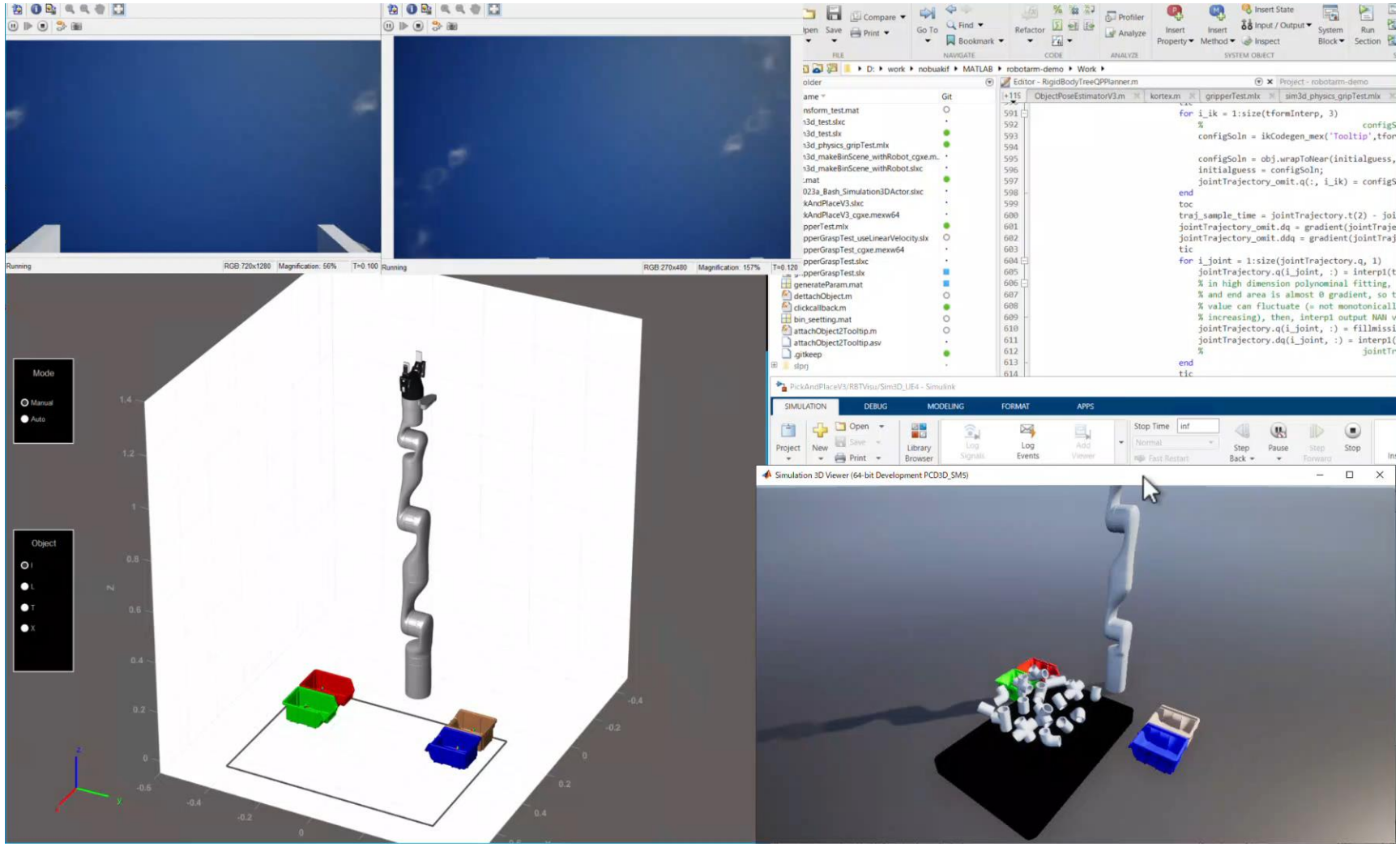
PX4 Autopilot

NVIDIA® Jetson®

ROS Nodes

Speedgoat

ばら積み物体ピッキングロボットのフルシミュレーション



実機適用

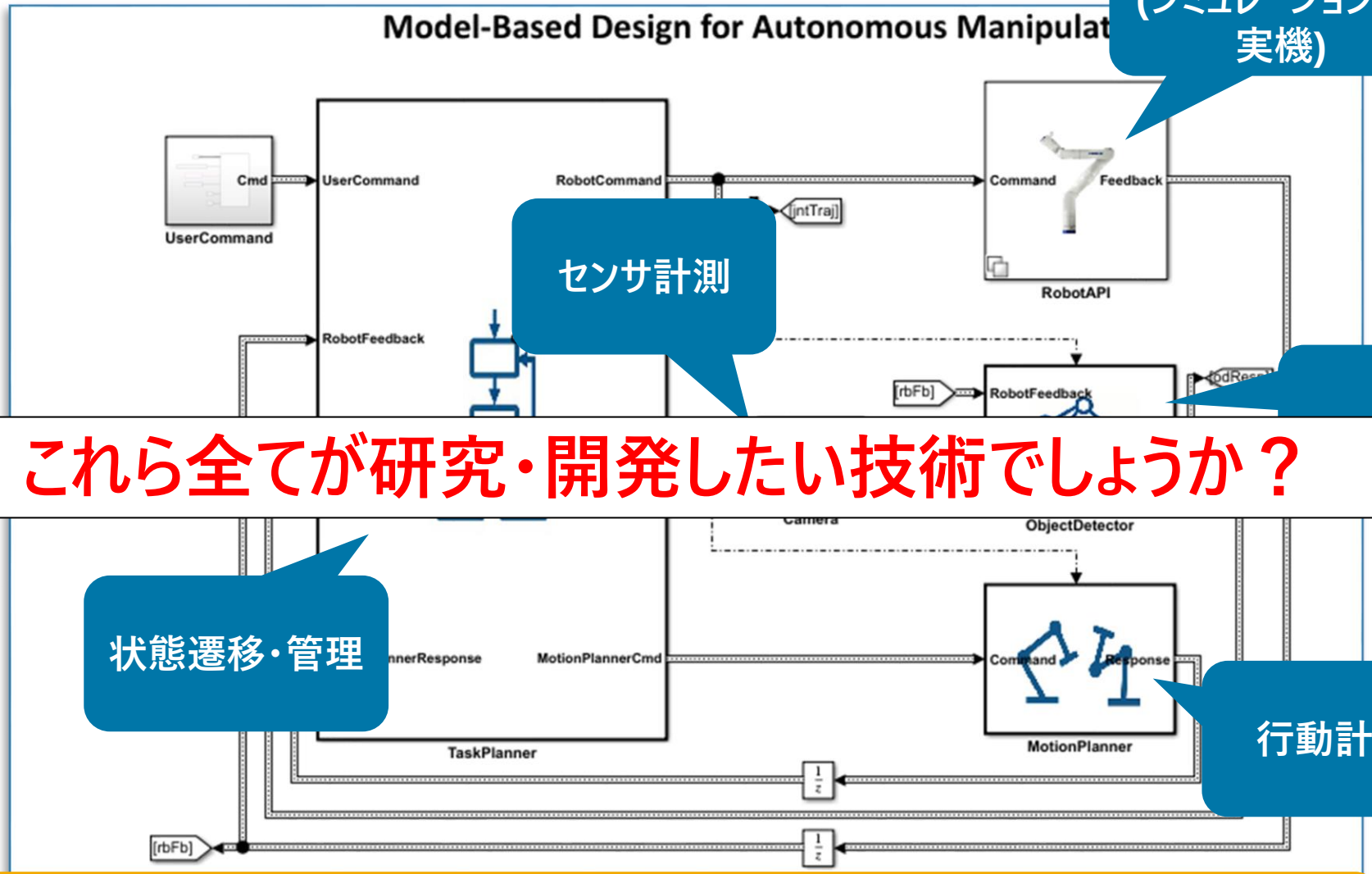
認知・判断・制御を全てシミュレーションで検証し、実機の課題をフロントローディング

ロボット開発・AI活用における課題

1. ロボットに必要な要素技術は多様
   →集中したい要素技術が1つであっても
   ロボットシステム全体の準備・検証が必要

2. AIの学習・検証に必要な**教師データの用意にかかる労力大**

ロボットシステムのシステムフロー



プラットフォーム
(シミュレーション or 実機)

センサ計測

認識

これら全てが研究・開発したい技術でしょうか？

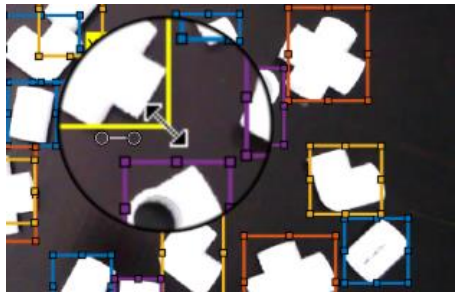
状態遷移・管理

行動計画

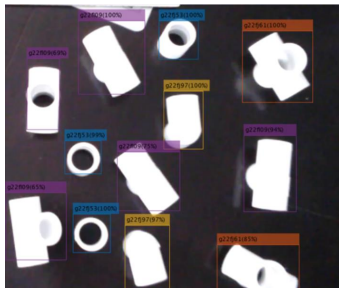
ロボットシステム全体の要素技術は多い
→注力領域以外はすぐに活用可能な技術資産を賢く流用

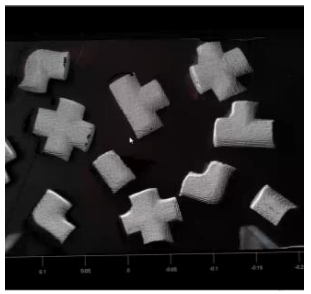
自律システム分野におけるAI活用の流れ

データ取得　データアノテーション



アルゴリズム開発





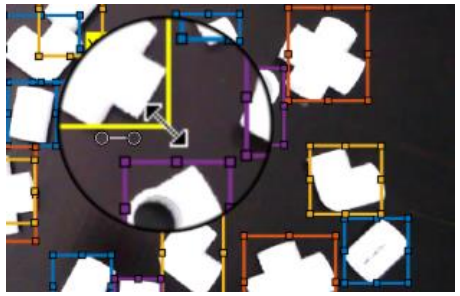
開発の本質！

非本質的な作業

本質的な課題

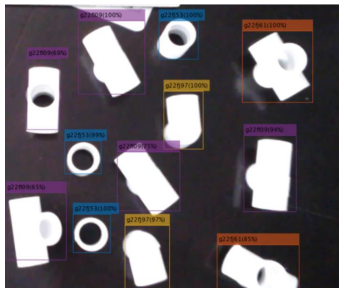
自律システム分野におけるAI活用の流れ

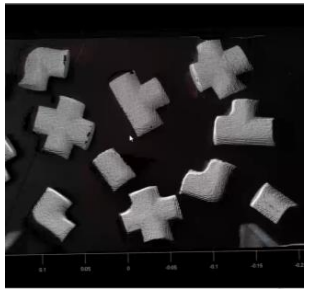
データ取得　データアノテーション



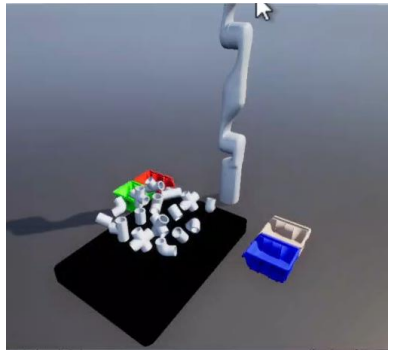
開発の本質！

アルゴリズム開発

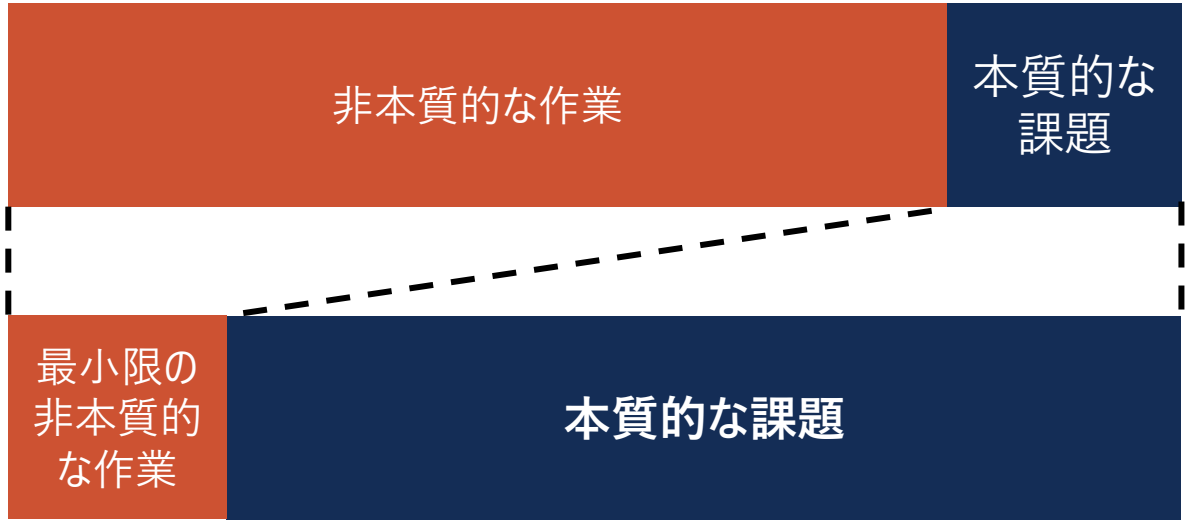




3Dシミュレーション



大量の
データ自動
取得



ボトルネックであるデータ取得プロセスを効率化し、開発全体を効率化

3Dシミュレーション環境（仮想環境）活用のメリット

開発の課題

- 実機(ハードウェア)・データがないと試験できない
- 実験は安全なもの、かつ、データ数が限られる
- 再現性の確保、課題発見時の解析が困難

仮想環境活用の利点

- 実機がなくても試験ができる
- 危険なシナリオやバリエーションを振った試験の実行
- 真値を使った、正確な評価・比較
- Deep Learningの教師データ、
  アルゴリズム検証用の正解ラベル付けが不要

ロボット開発・AI活用における課題

1. ロボットに必要な要素技術は多様
   →集中したい要素技術が1つであっても
   ロボットシステム全体の準備・検証が必要
   ⇒MATLAB/Simulinkには豊富なロボットライブラリ・例題がご用意

2. AIの学習・検証に必要な**教師データの用意にかかる労力大**
   ⇒MATLAB/Simulinkの3Dシミュレーション活用で労力を最小化

AI駆動ロボットシステムの設計ワークフロー

1. データ準備
2. AI モデリング
3. シミュレーション
4. 実装

AI駆動ロボットシステムの設計ワークフロー

1 データ準備

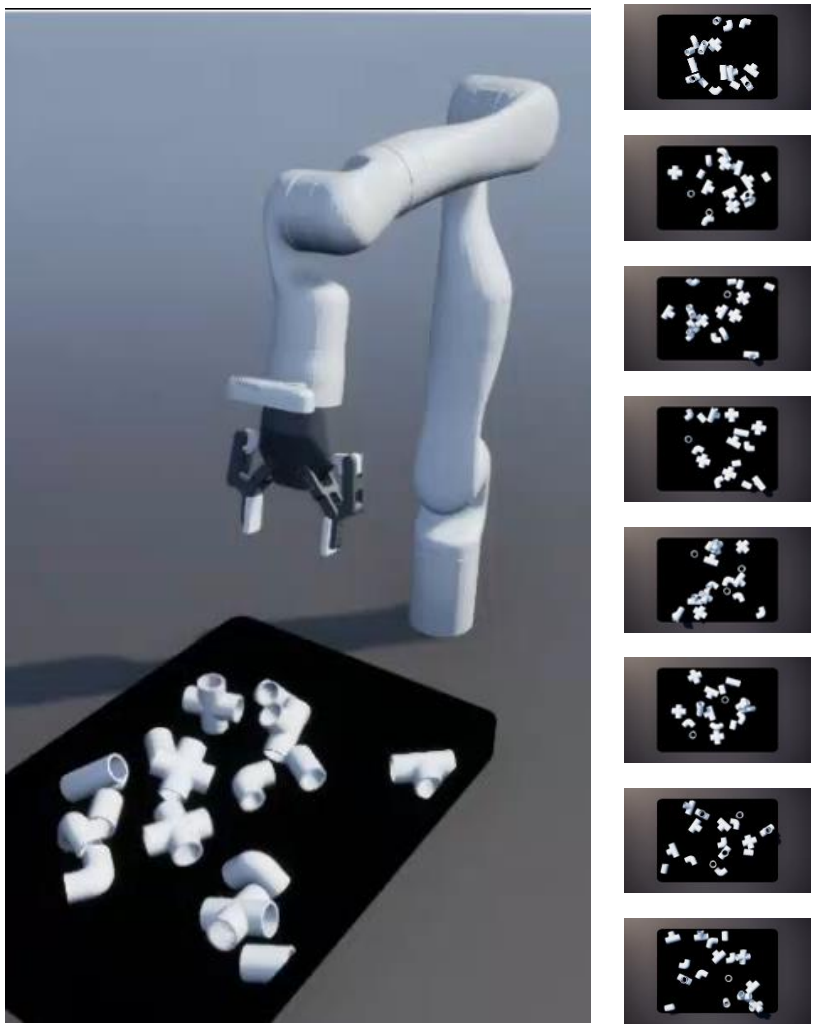
2 AI モデリング

3 シミュレーション

4 実装

学習データをどのように準備するか？

Simulink 3D Animation
Robotics System Toolbox
Computer Vision Toolbox

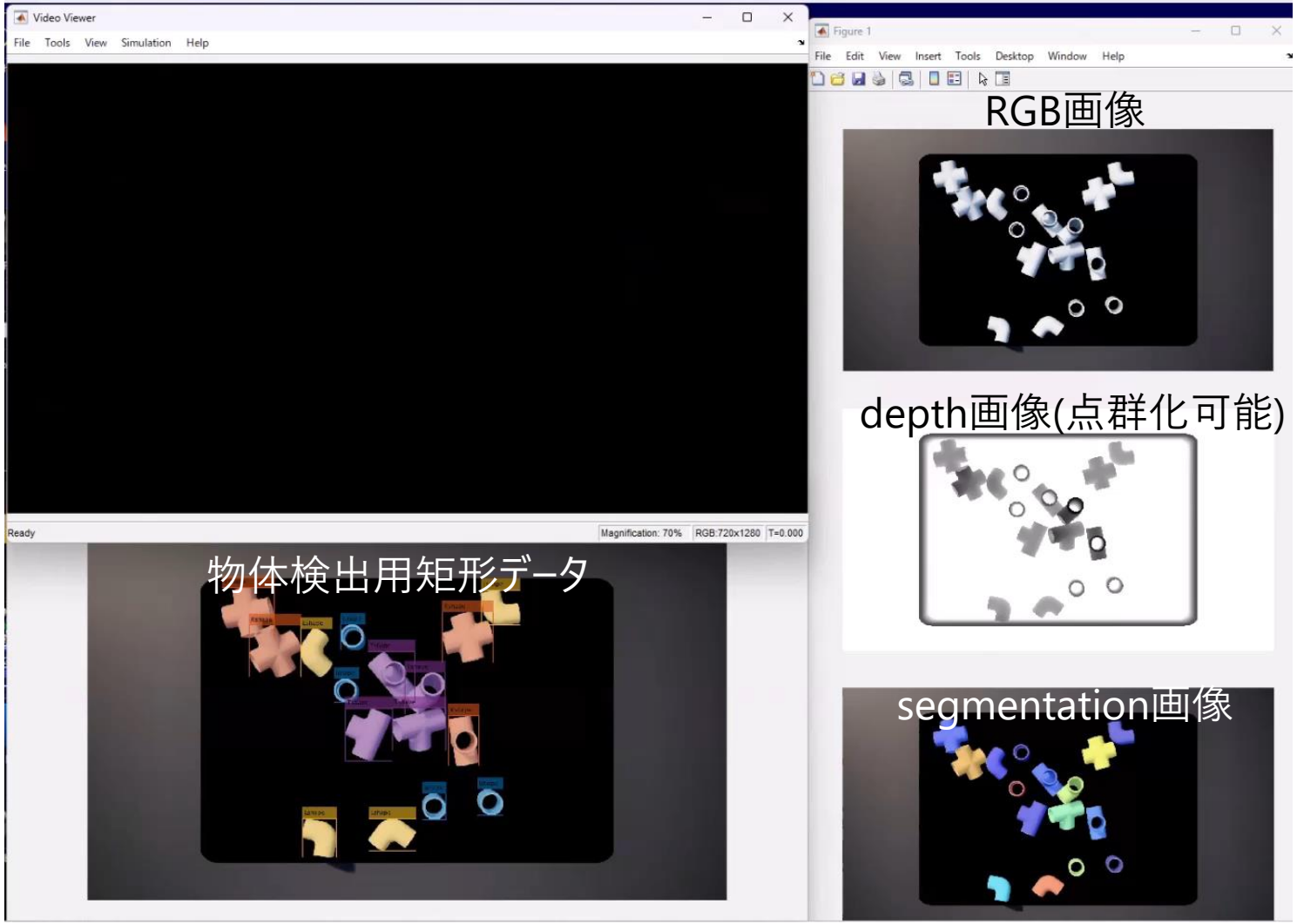


シミュレータによる教師データの自動生成

ハードウェア連携による自動撮影

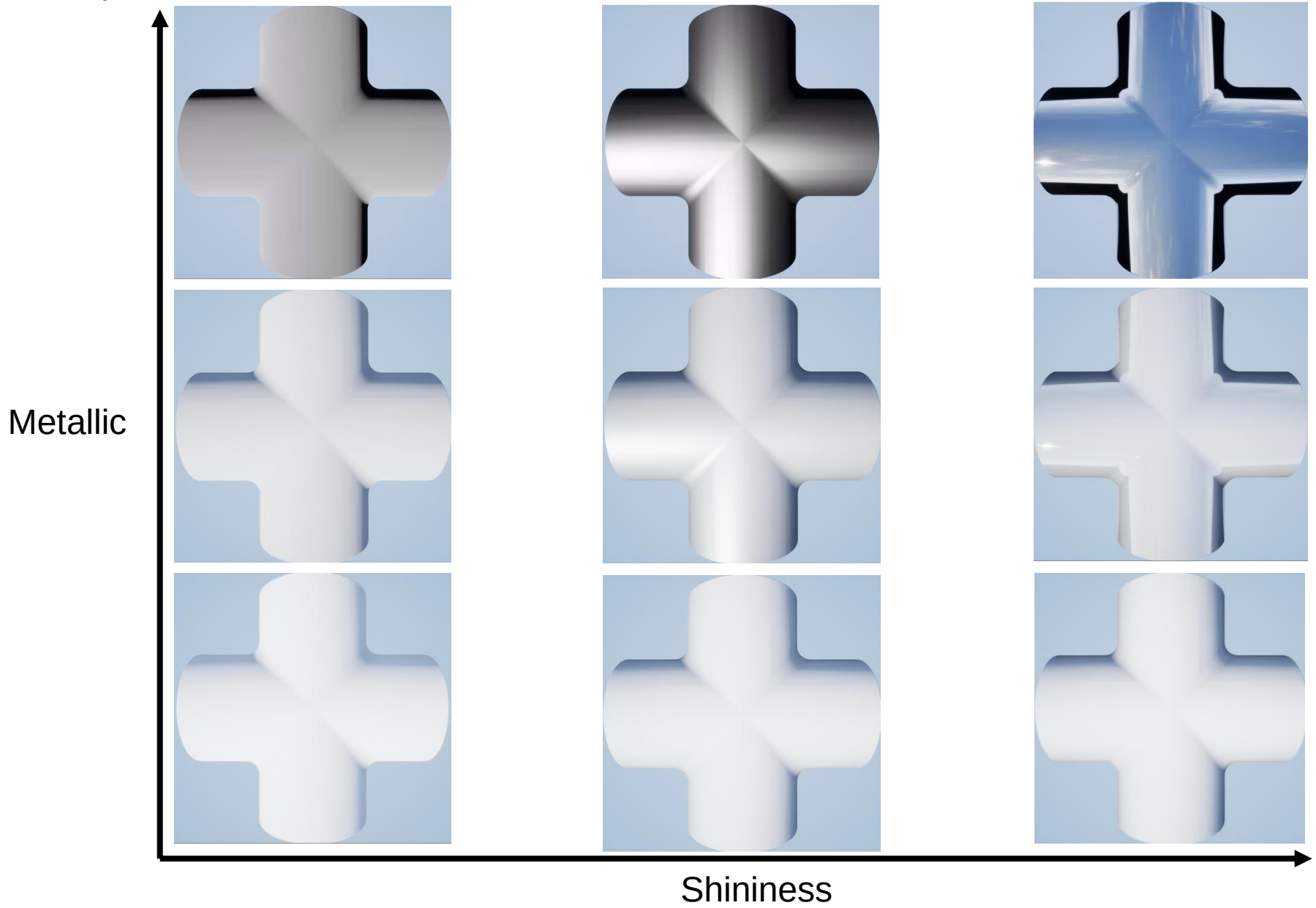
物体検出/セグメンテーション Deep Learningの教師データ作成

sparseInst:
instance segmentation用
Deep Learningモデル

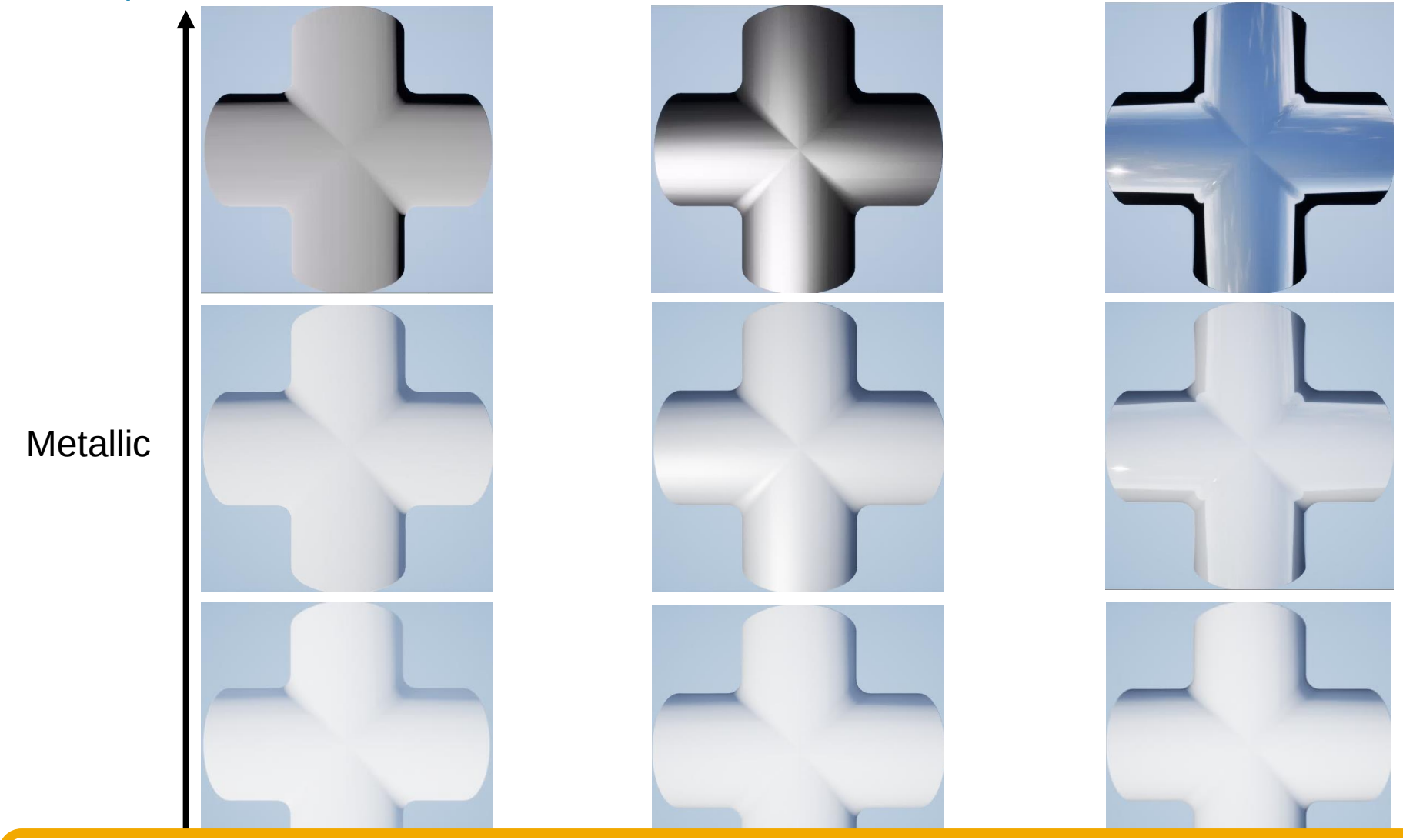


手間がかかる教師データ作成を"リアル"なシミュレーションで大量作成

Metallic, Shininessプロパティによる見た目の変化



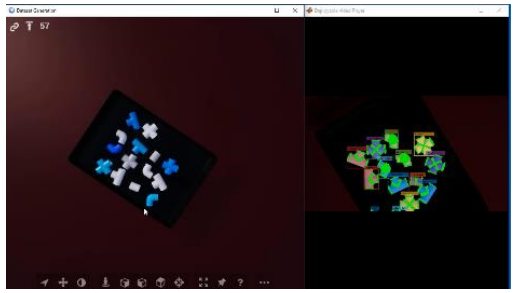
Metallic, Shininessプロパティによる見た目の変化



見た目のバリエーションを増やし、学習データのDomain randomizationに寄与

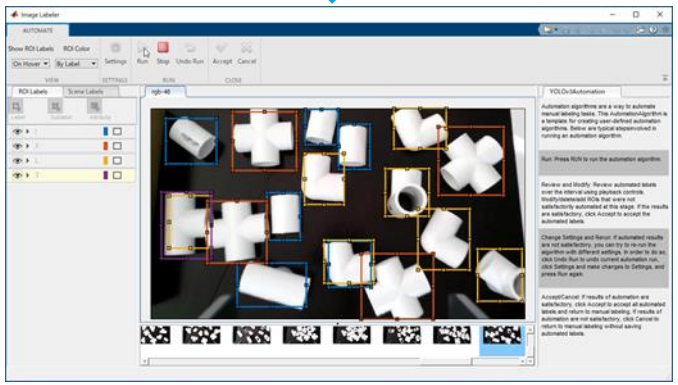
合成データの利用例

教師データの自動生成⇒リアルデータと合わせる

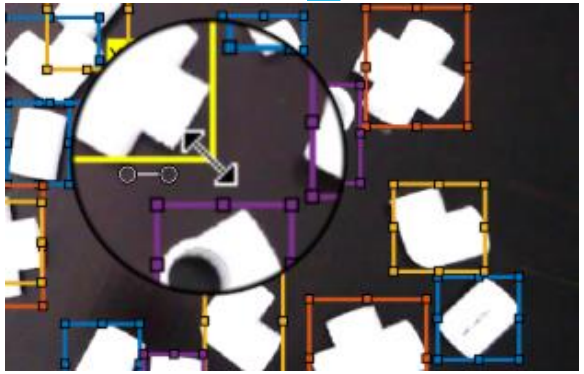


合成画像を生成

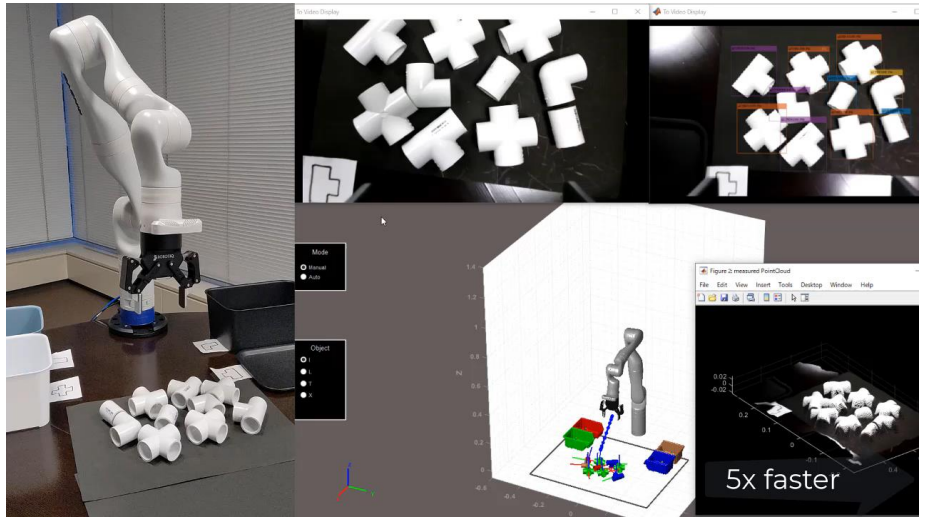
物体検出器を学習



現実のデータを
自動ラベリング



誤ったラベルのみ修正



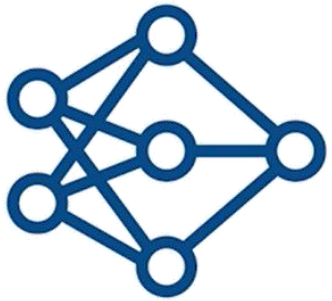
AI駆動ロボットシステムの設計ワークフロー

1 データ準備

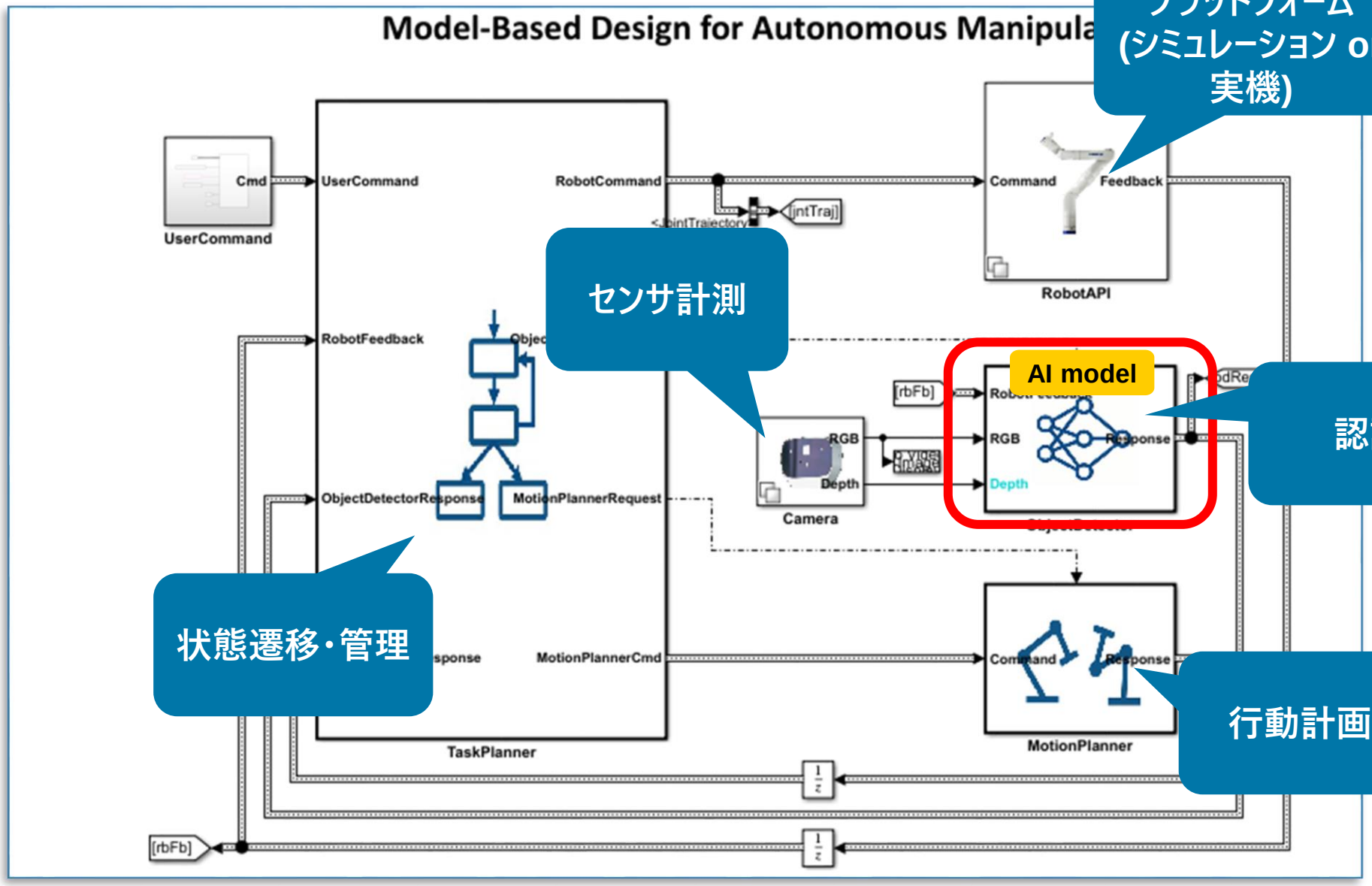
2 AI モデリング

3 シミュレーション

4 実装



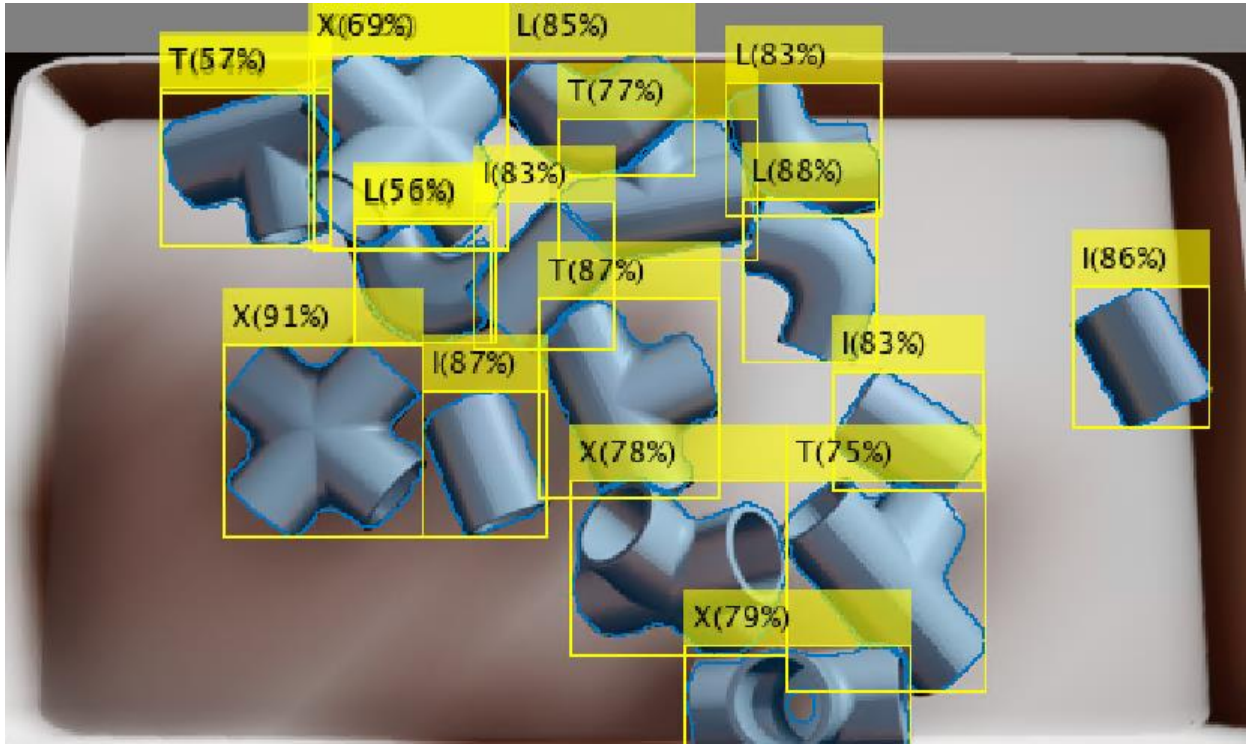
ロボットシステム内でのAI活用



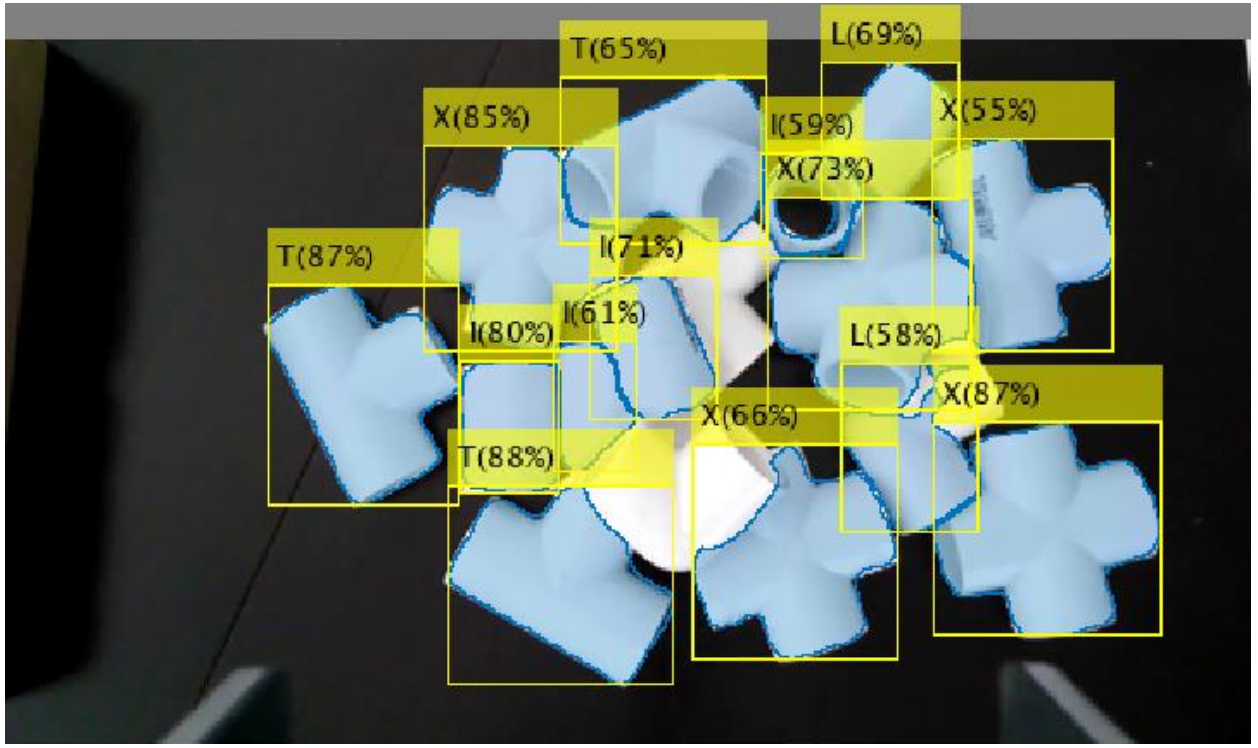
物体検出/セグメンテーション Deep Learningの教師データ作成
Deep Learningモデルによる検証

GitHub - hustvl/SparseInst: SparseInst: Sparse Instance Activation for Real-Time Instance Segmentation, CVPR 2022

sparseInst: instance segmentation用Deep Learningモデル
使用したモデルはSimulation生成画像900枚で学習



sparseInstでのSim画像推定結果

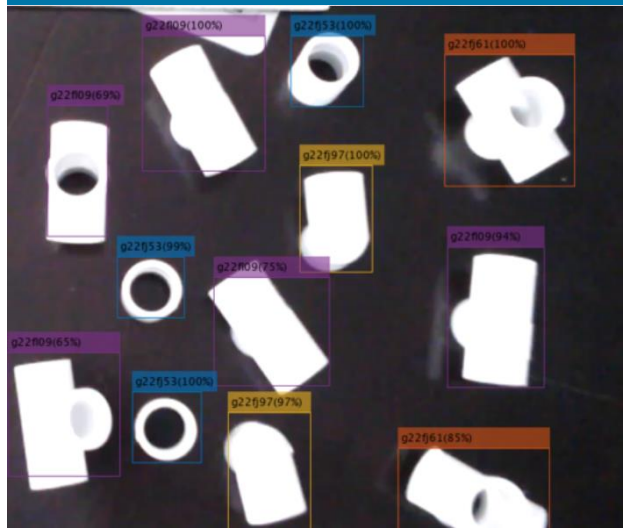


sparseInstでの**実画像**推定結果

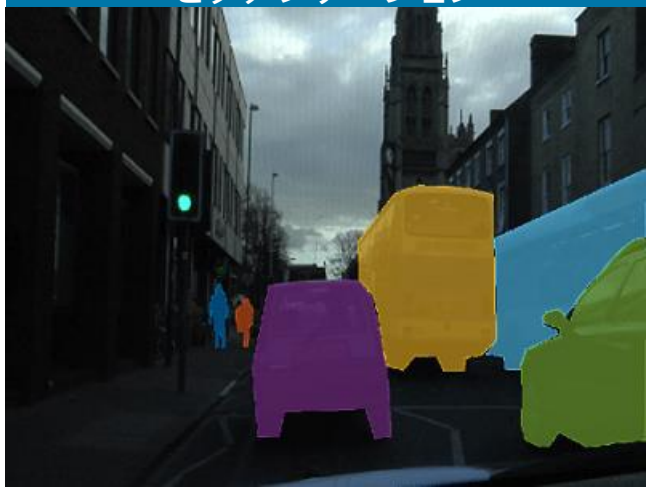
実画像のアノテーションなしで実画像の検出・セグメンテーションが可能！

実績のあるAIモデルを試す

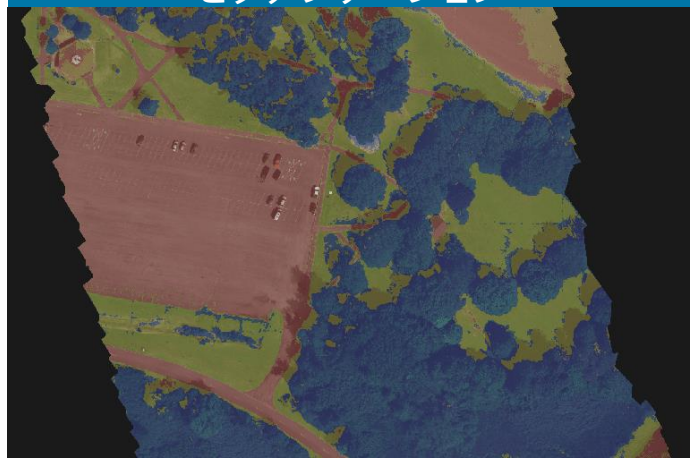
YOLOv4による
物体検出



Mask R-CNNによる
インスタンス
セグメンテーション

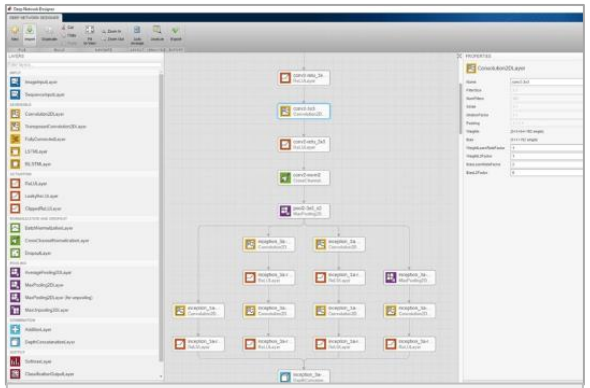


U-Netによる
セマンティック
セグメンテーション

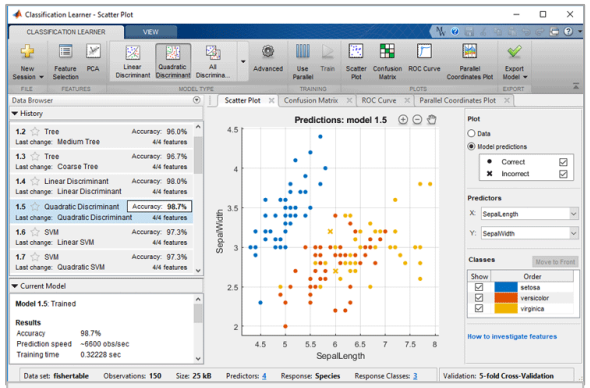


Deep Learning Toolbox
Image Processing Toolbox
Computer Vision Toolbox

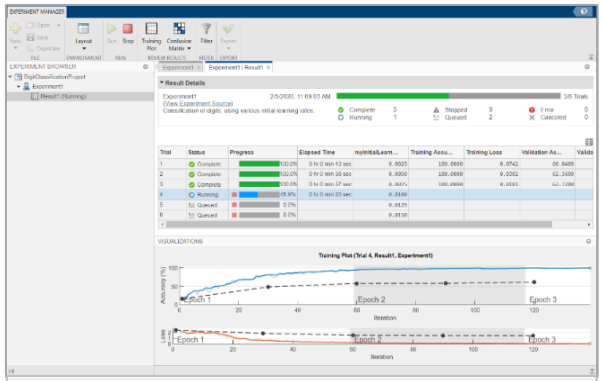
学習、チューニング、可視化を行うアプリ



Deep Network Designer app to build, visualize, and edit deep learning networks.



Classification Learner app to try different classifiers and find the best fit for data sets.



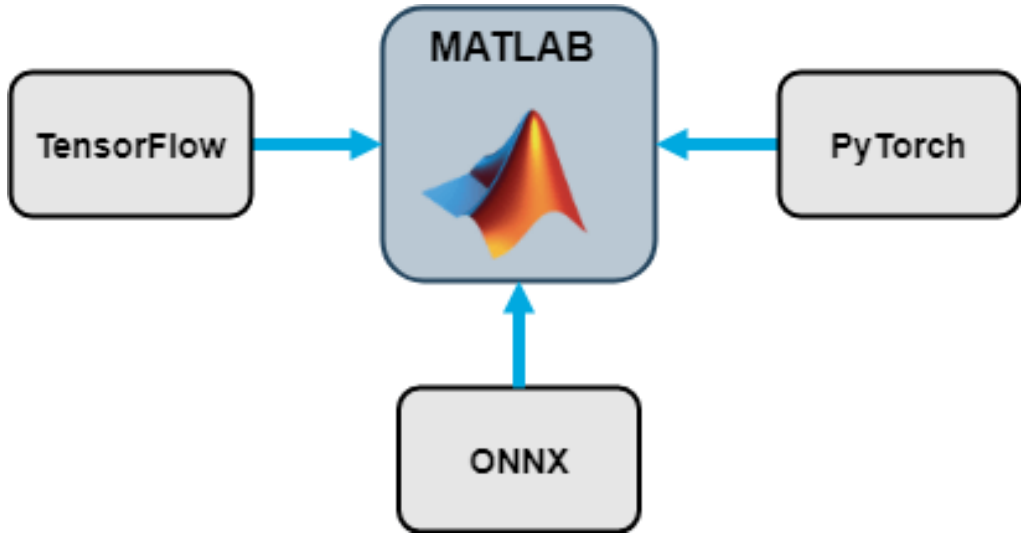
Experiment Manager app to run deep learning experiments to train networks and compare results.

OSSフレームワークとの連携

- 他フレームワークによる学習済みモデルの取り込み
  - Tensorflow, Caffe, PyTorchモデル
  - ONNXフォーマット

他フレームワークとの連携

```
net = importTensorFlowNetwork(modelfile)
net = importNetworkFromPyTorch(modelfile)
net = importONNXNetwork(modelfile, ...
                        'OutputLayerType',outputtype)
```



AIモデル＋前処理・後処理によるAI認知アルゴリズムを構築

前処理

AI モデル

後処理

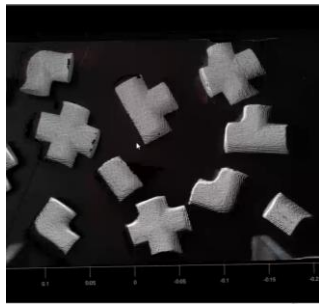
RGB-D 画像

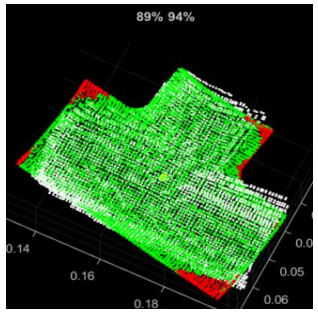
点群
再構築

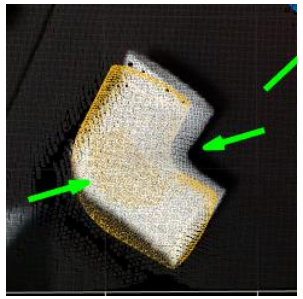
物体検出

CADモデル
マッチング

6 自由度の
姿勢推定







Deep Learning Toolbox
Image Processing Toolbox
Computer Vision Toolbox

AI駆動ロボットシステムの設計ワークフロー

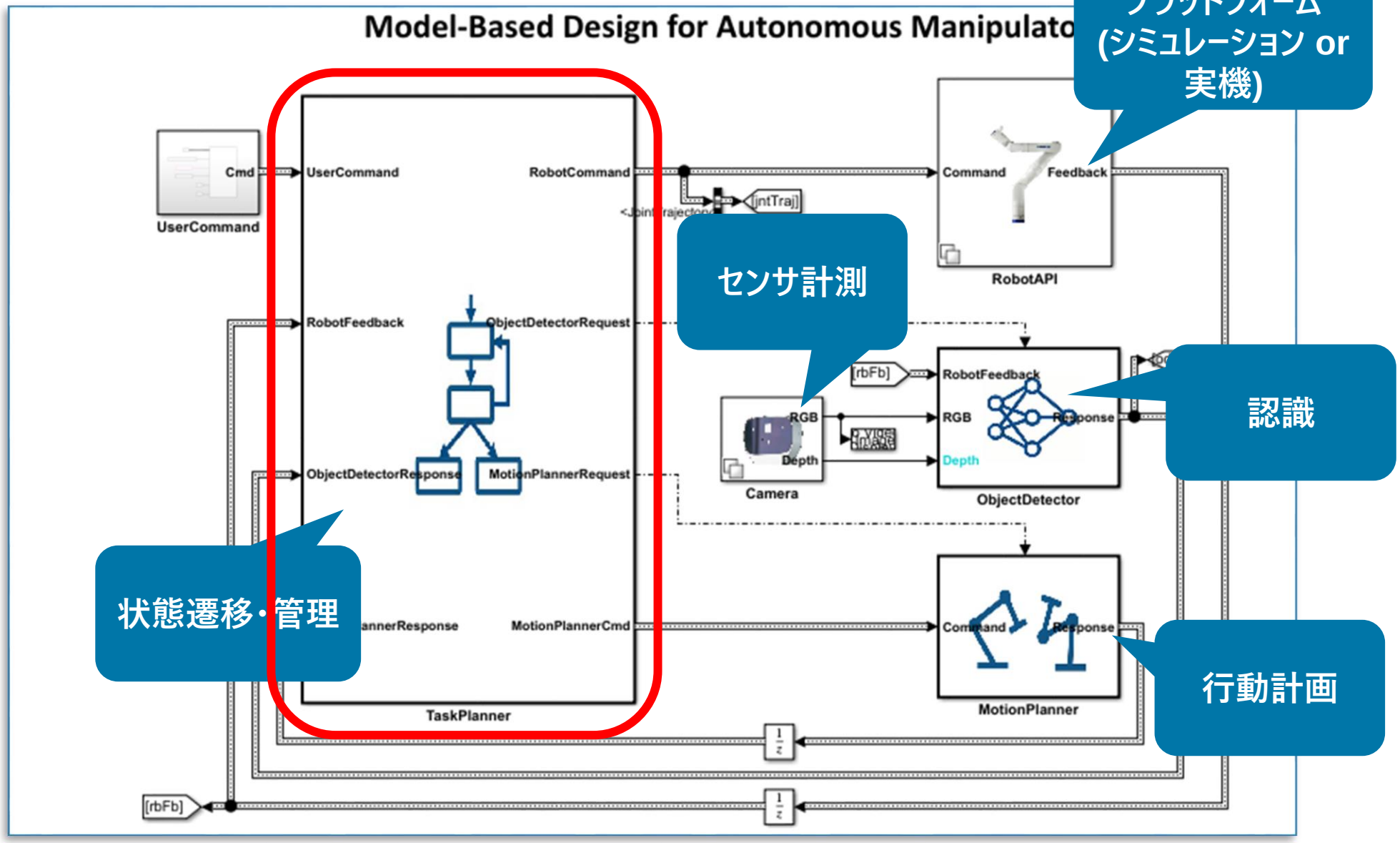
1 データ準備

2 AI モデリング

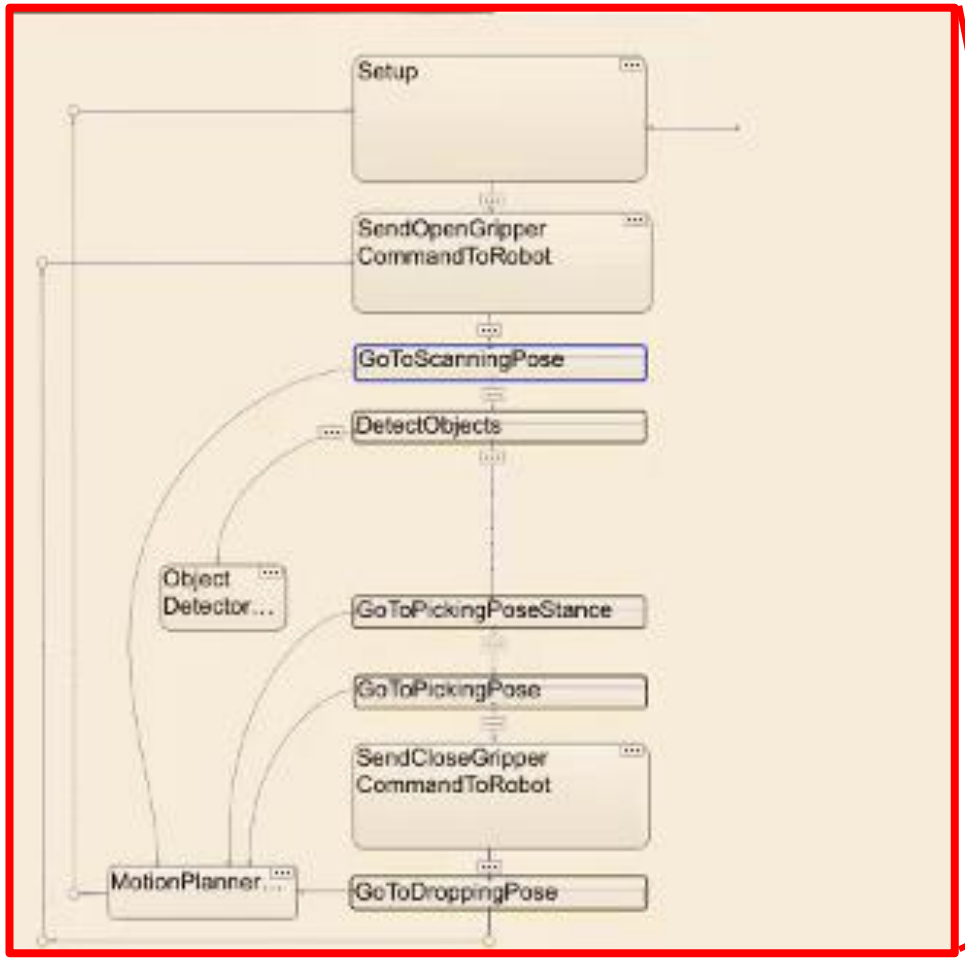
3 シミュレーション

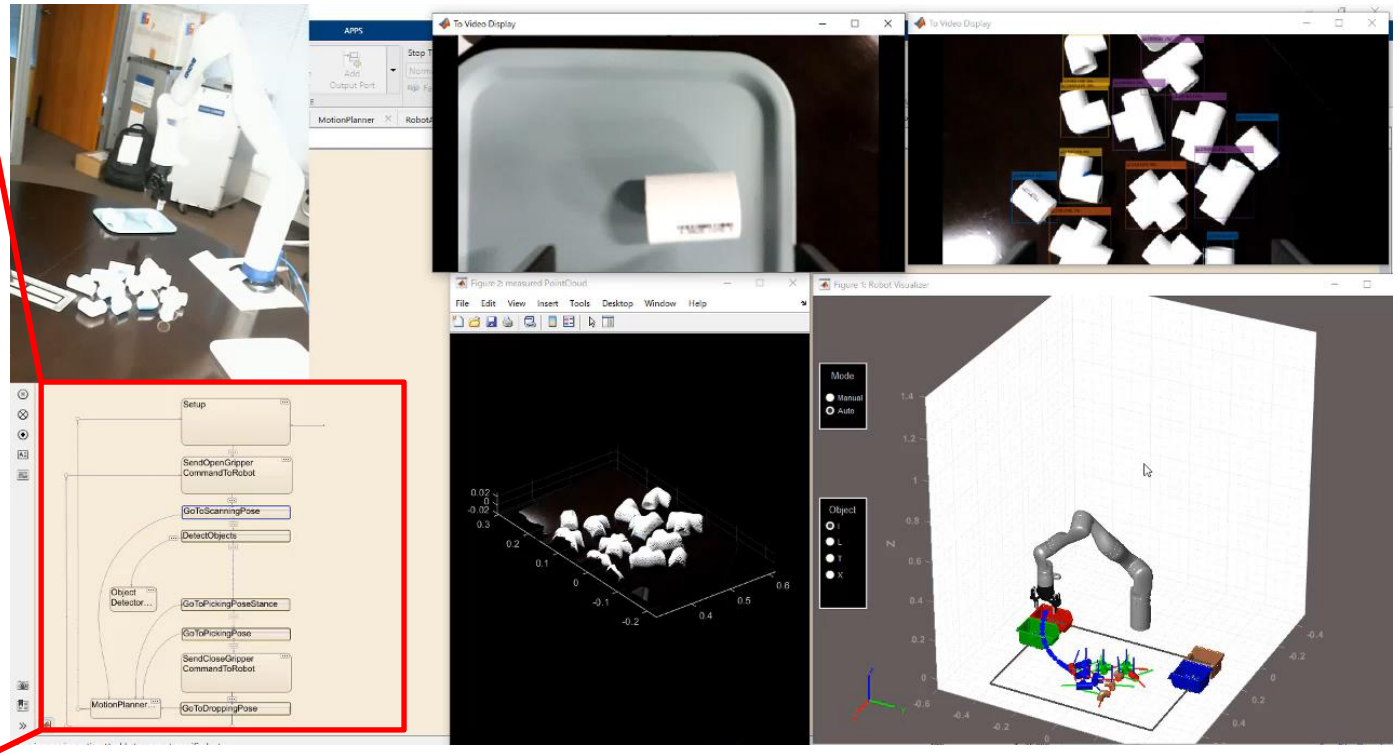
4 実装

ロボットシステム内でのAI活用



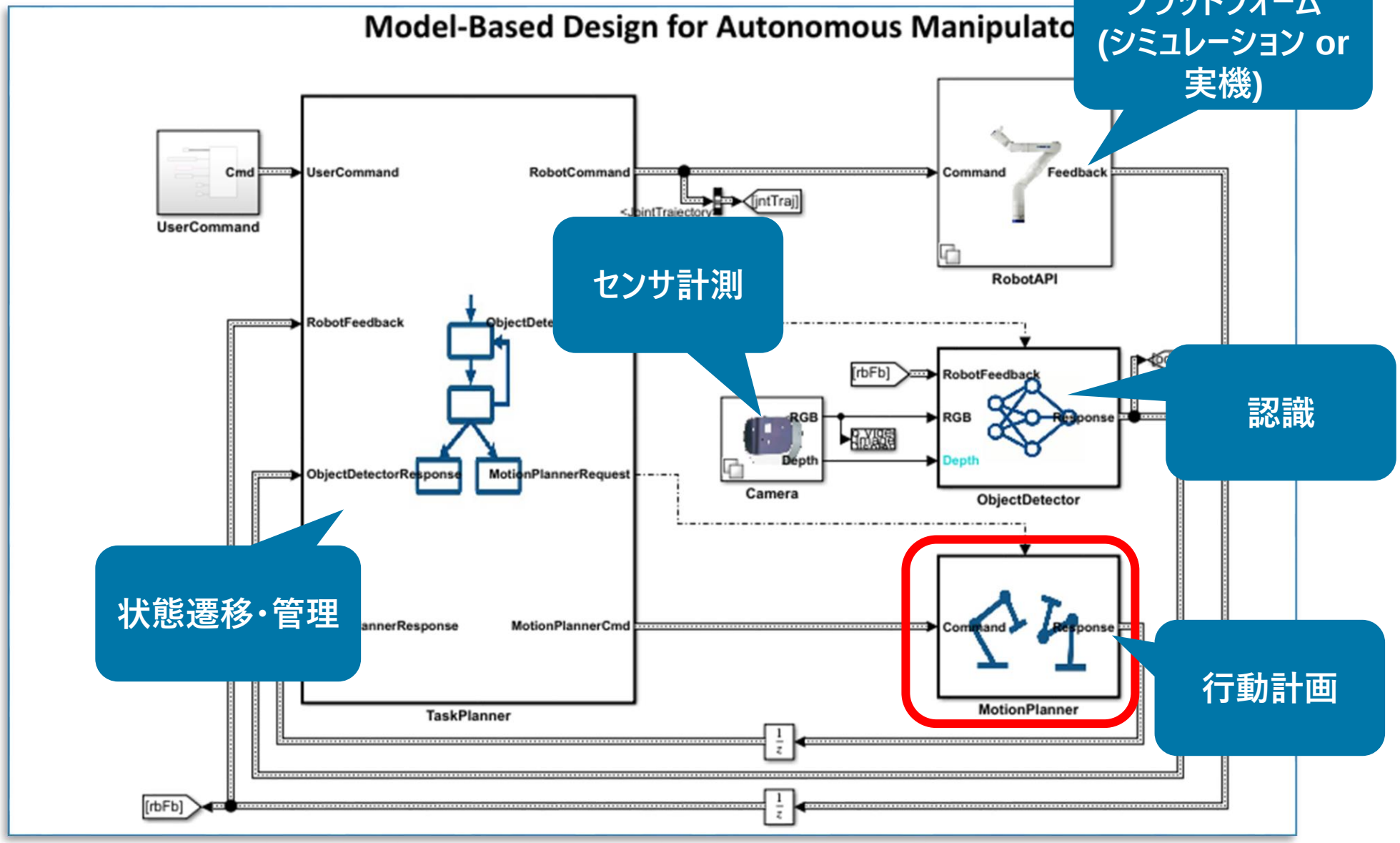
Stateflowによる直感的な状態遷移・管理





複雑な状態遷移をGUIでわかりやすく管理

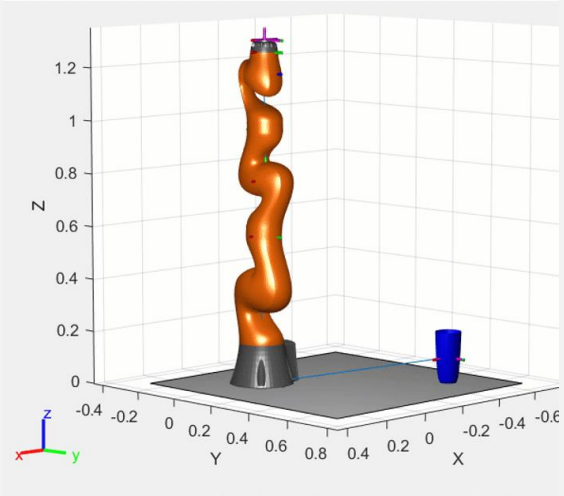
ロボットシステム内でのAI活用



ロボット分野の各種アルゴリズム

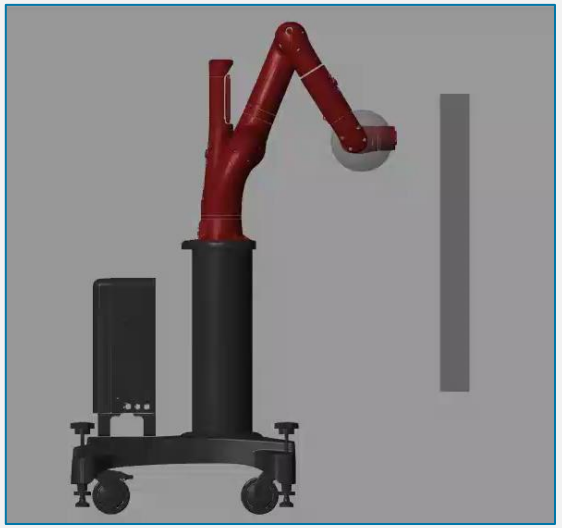
Robot manipulator algorithm design

- Forward /Inverse Kinematics
  (順逆運動学解析)
- Kinematic constraints
  (機構制約の設定)
- Generalized Inverse Kinematics(汎用IK)
- Trajectory generation and following
  (経路計画&追従アルゴリズム)
- Collision checking
  (干渉チェック)

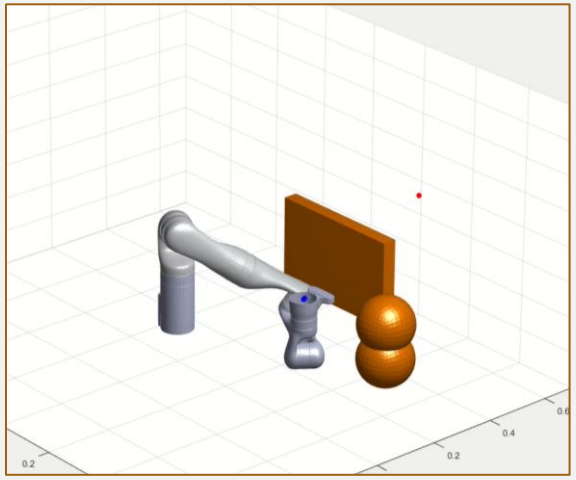


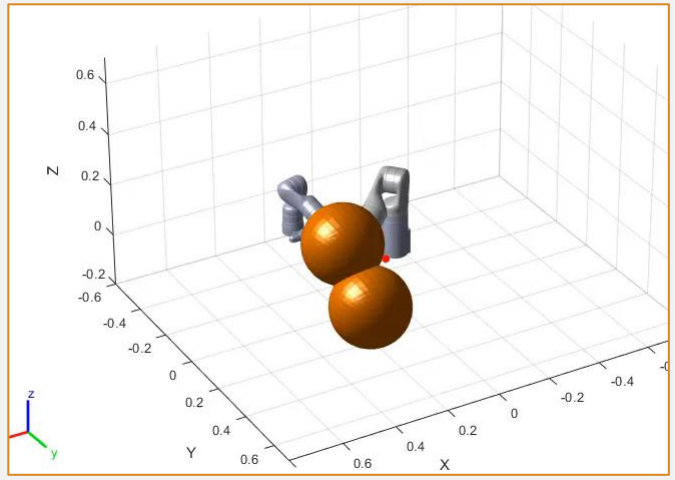
◀ Inverse Kinematics

Trajectory Tracking▶

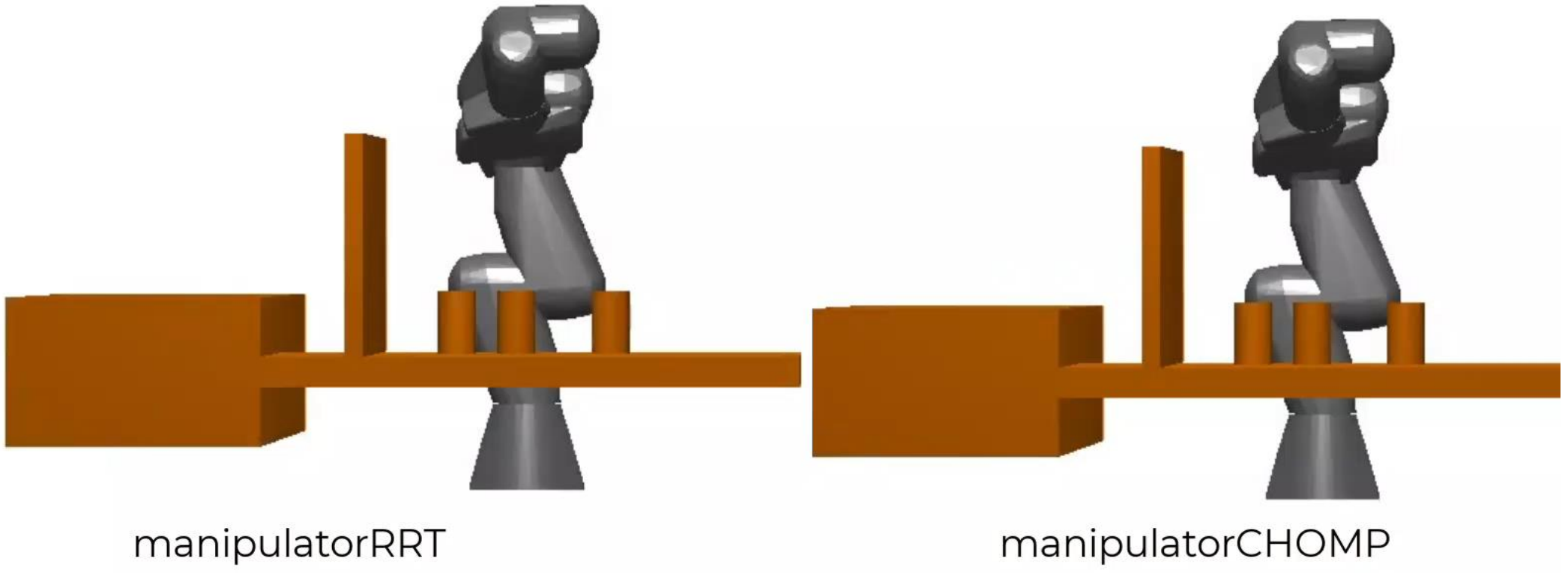


Collision Checking▶





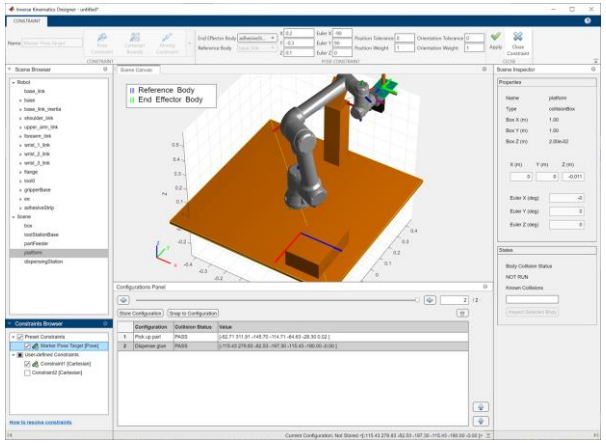
衝突判定を含めた軌道計画



Pick-And-Place Workflow Using CHOMP for Manipulators - MATLAB & Simulink - MathWorks 日本
Pick and Place Using RRT for Manipulators - MATLAB & Simulink - MathWorks 日本

モーションプランニング

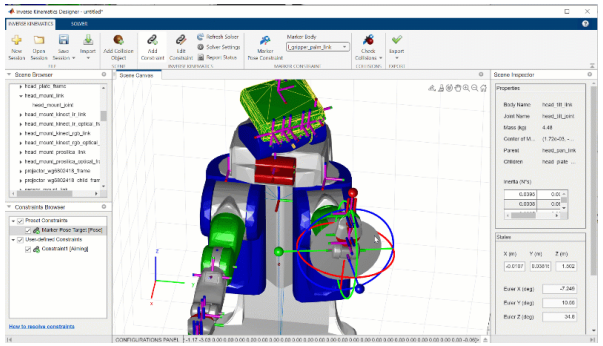
逆運動学アプリ



[Plan Manipulator Path for Dispensing Task Using Inverse Kinematics Designer]

Robotics System Toolbox™

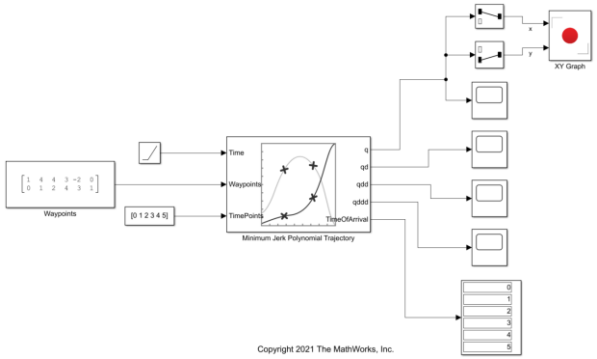
軌道生成アプリ



[Create Constrained Inverse Kinematics Solver Using Inverse Kinematics Designer]

Robotics System Toolbox™

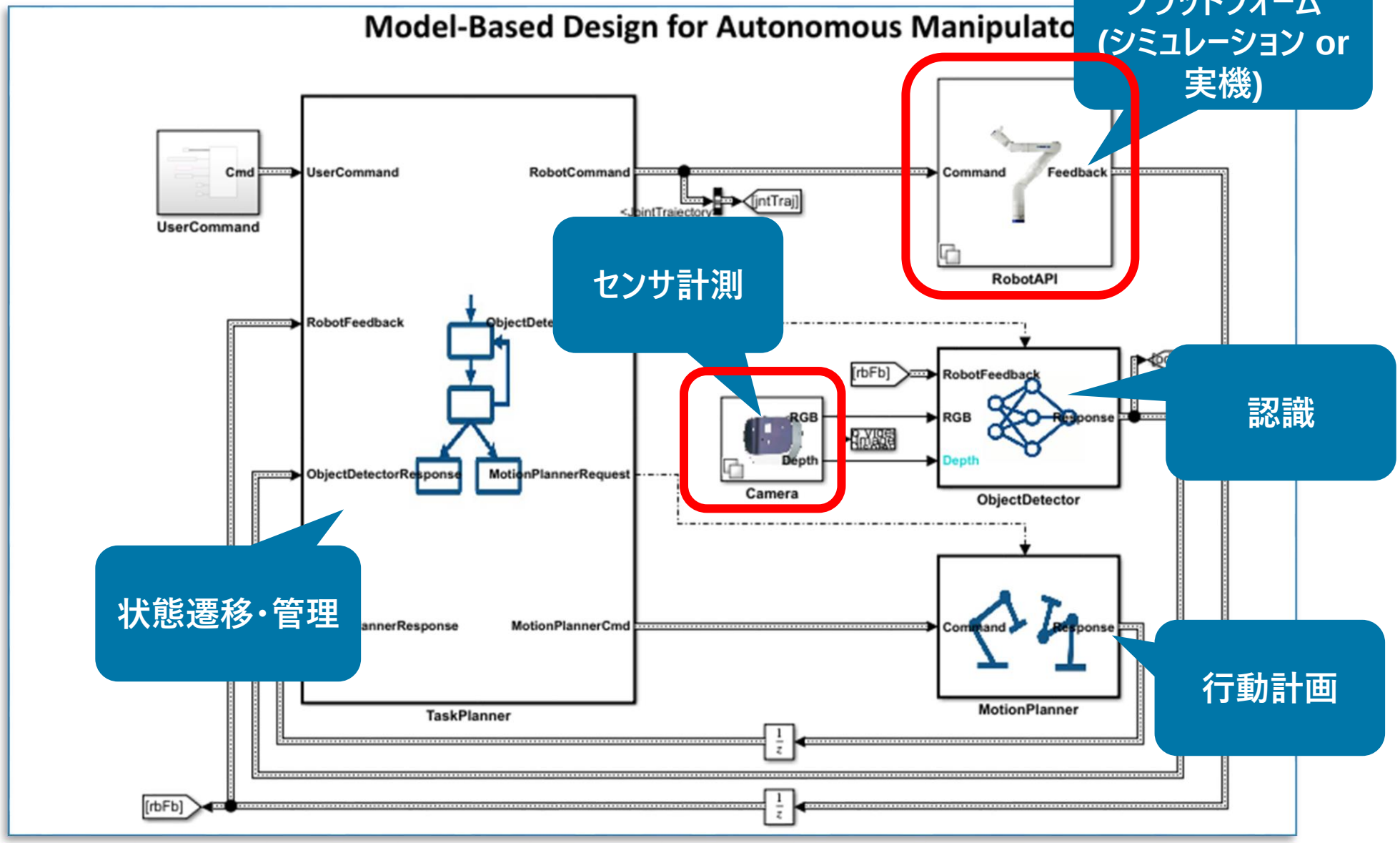
ジャーク/スナップ 最小軌道の生成



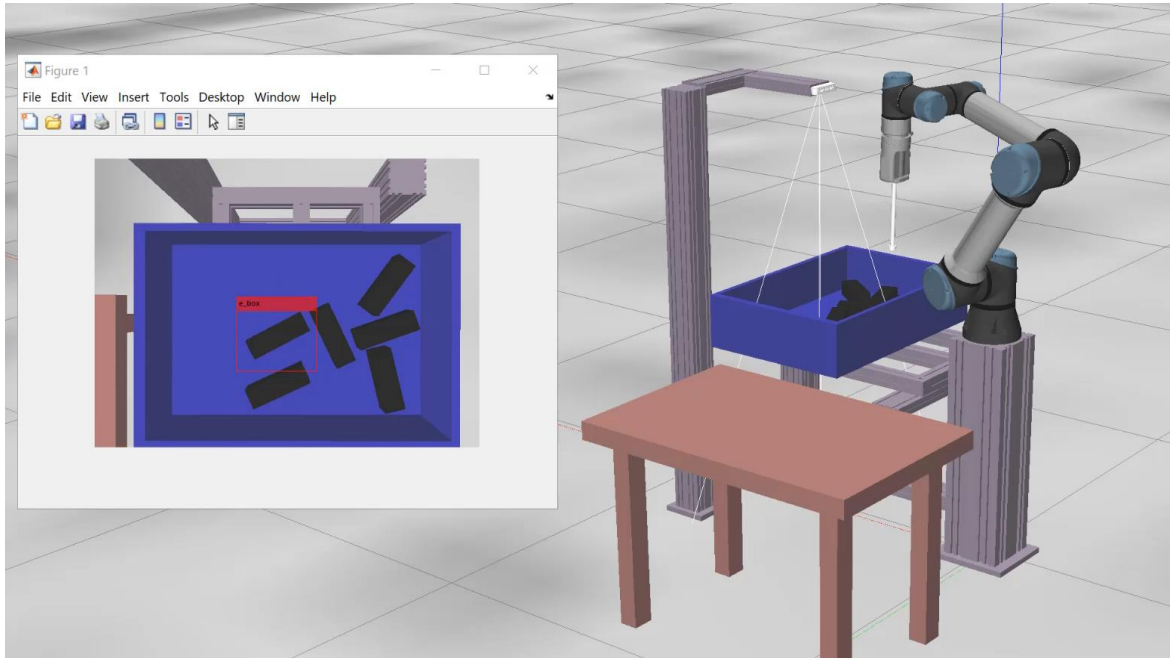
[Generate Minimum Jerk Trajectory]

Robotics System Toolbox™

ロボットシステム内でのAI活用

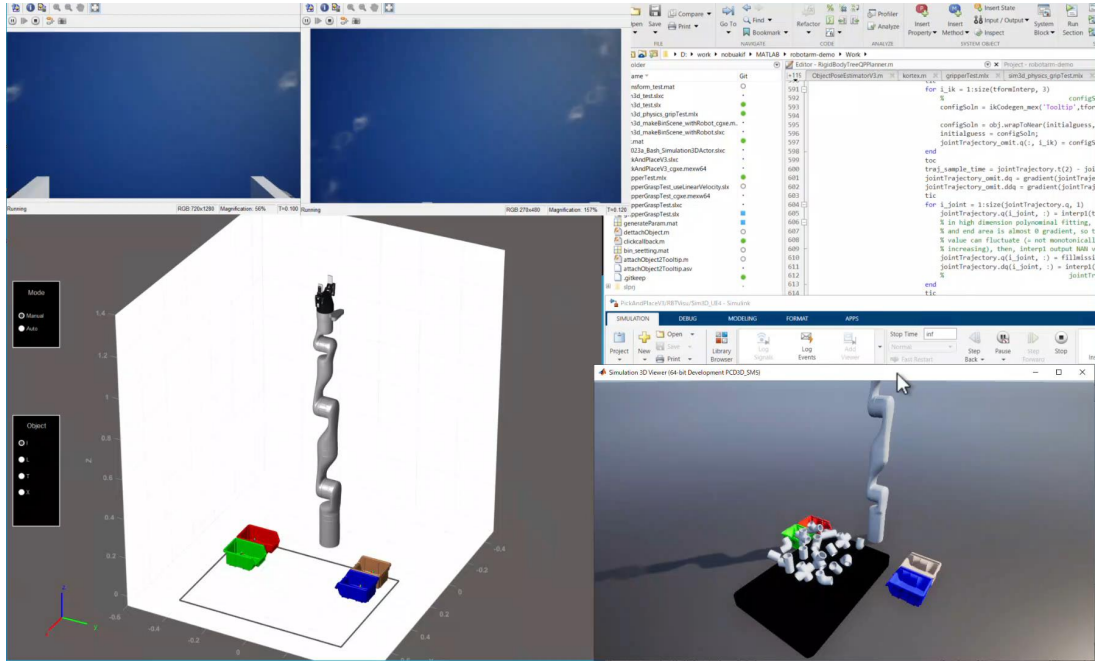


シミュレーションによるセンシング、制御開発



Universal RobotのGazebo シミュレーション

- ROSによるロボット制御
- GazeboとMATLABをROS経由で接続

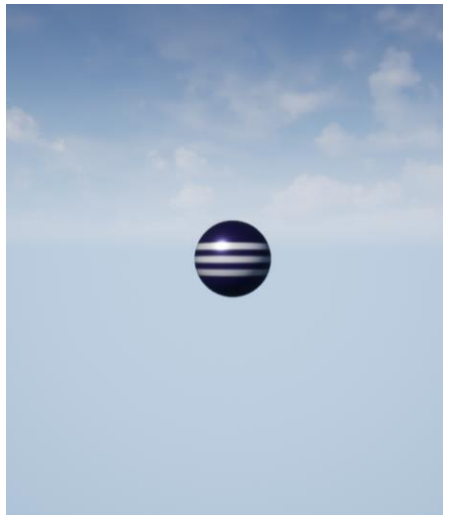


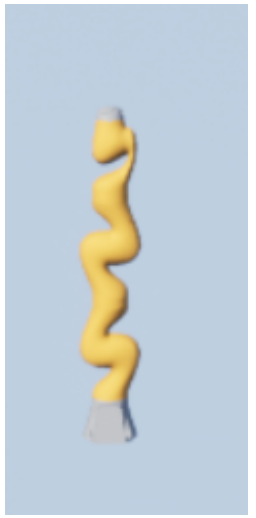
KinovaのUnreal Engine シミュレーション

- ROS、または、MATLAB用APIによるロボット制御
- MATLABのUnreal Engine連携機能を活用

Unreal Engine環境の作成

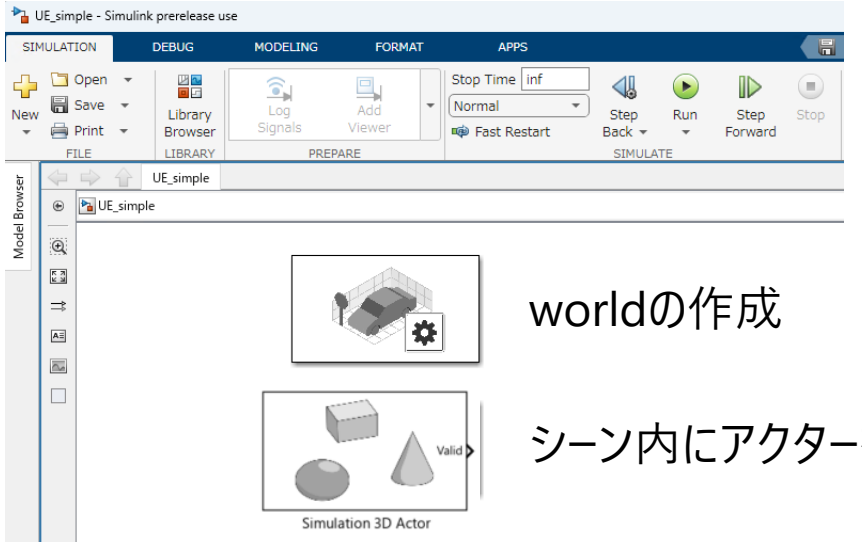
[Object used to define actors in the Unreal Engine viewer - MATLAB - MathWorks 日本]





URDFを指定可能！

```
% worldの作成
» world = sim3d.World();
% シーン内にアクターを作成
» sphere1 = sim3d.Actor('ActorName', 'sphere');
» sphere1.createShape("sphere");
» world.add(sphere1);
% アニメーションの実行
» world.run()
```



worldの作成

シーン内にアクターを作成

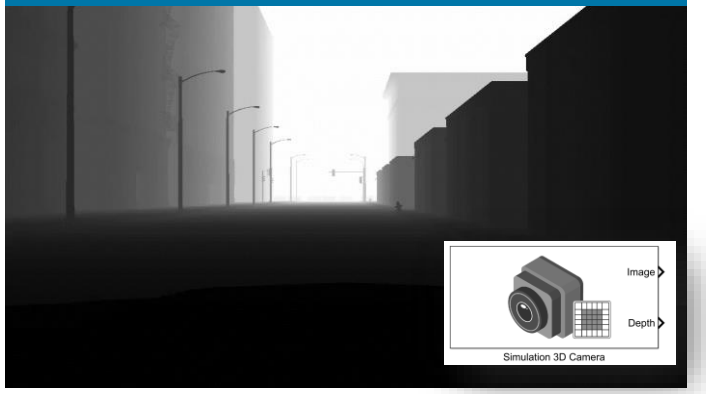
3Dシミュレーションの知識なしで容易に環境構築、アクターのスポーンが可能！

Unreal Engine連携用センサーモデルの提供

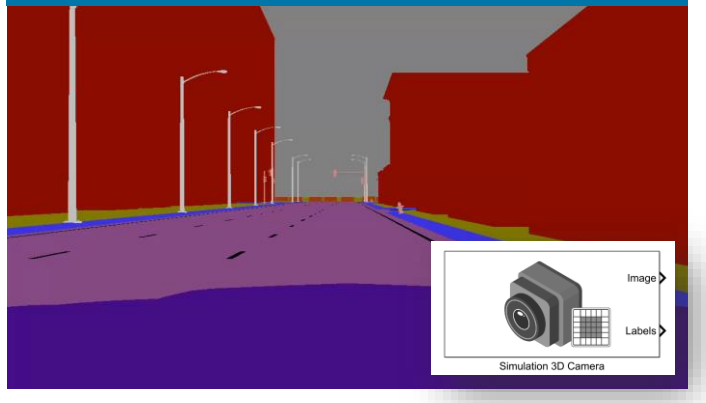
Automated Driving Toolbox

理想のピンホールカメラ

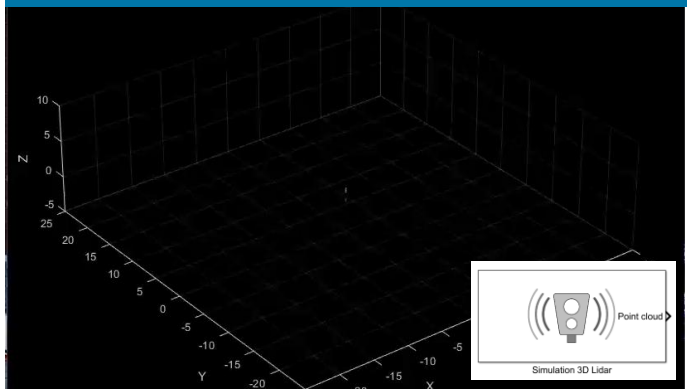
depthカメラ



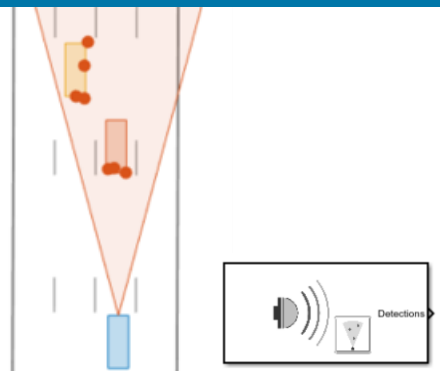
セマンティック セグメンテーション

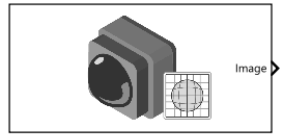


LiDAR



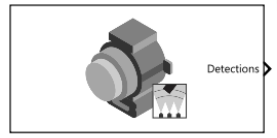
ミリ波レーダー





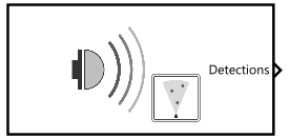
Simulation 3D Fisheye Camera

魚眼カメラ



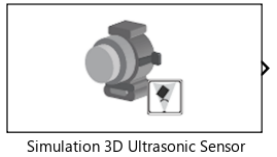
Simulation 3D Ultrasonic Array

超音波センサ



Simulation 3D Probabilistic Radar

ミリ波レーター（確率的検出点）



Simulation 3D Ultrasonic Sensor

超音波センサ（確率的検出点）

カメラ、LiDAR、ミリ波、超音波センサのセンサモデルブロックを用意

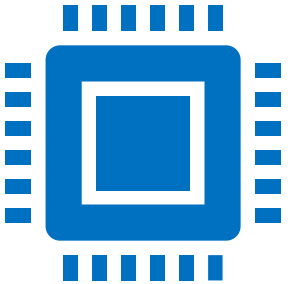
AI駆動ロボットシステムの設計ワークフロー

1 データ準備

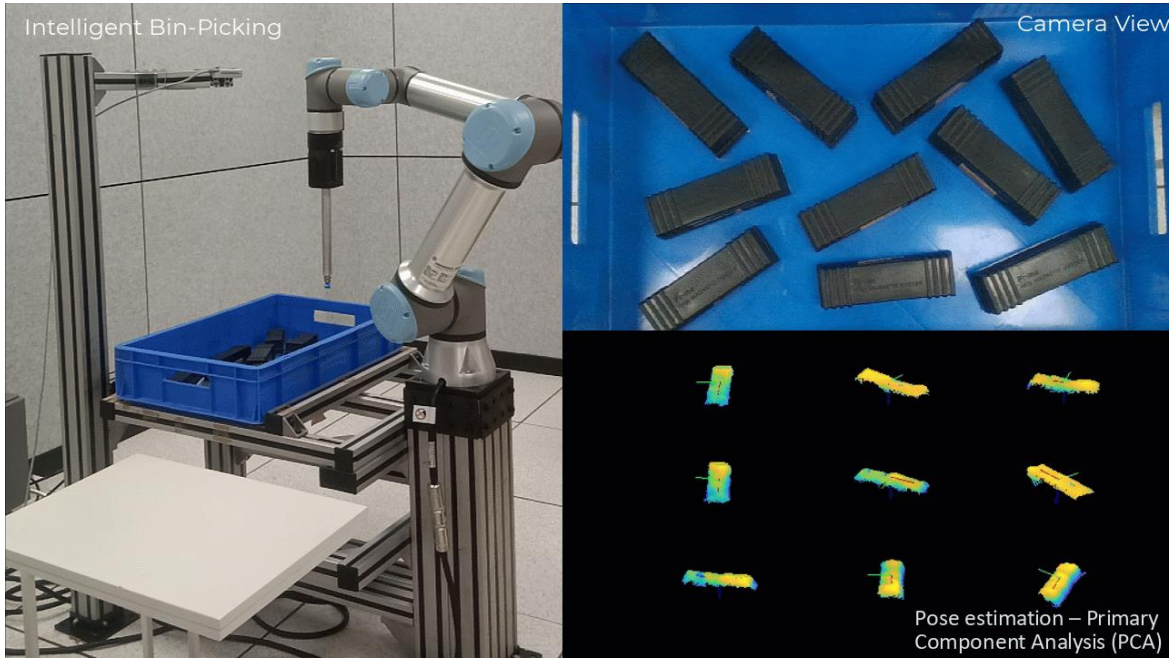
2 AI モデリング

3 シミュレーション

4 実装

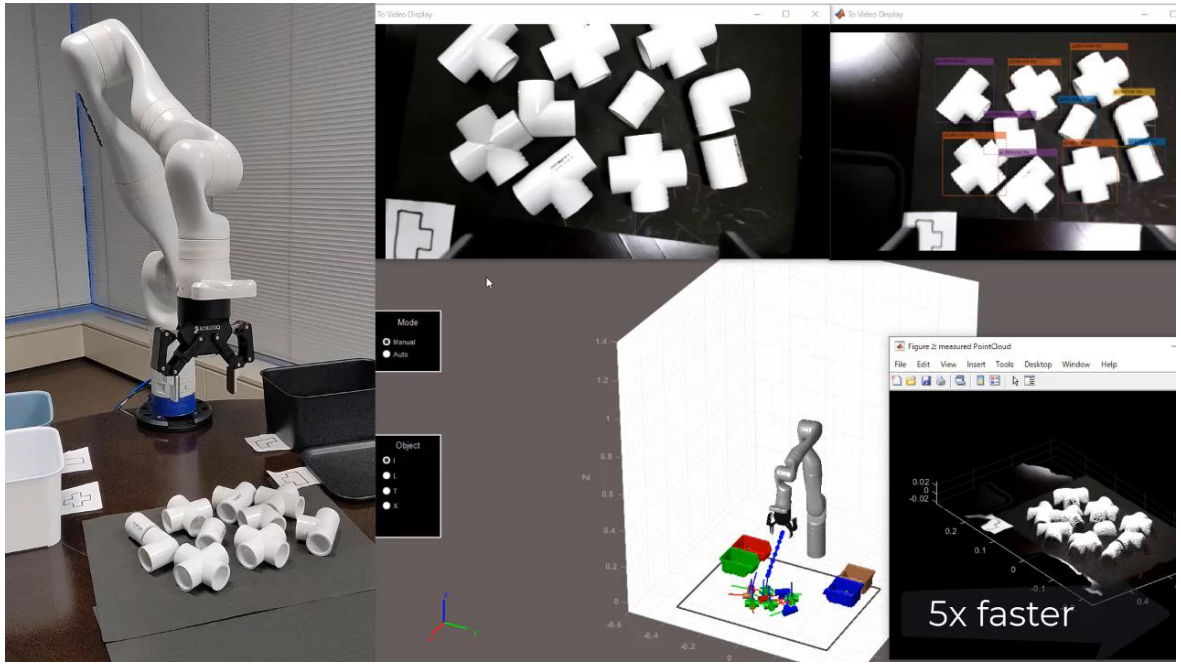


実機への接続



Universal Robot

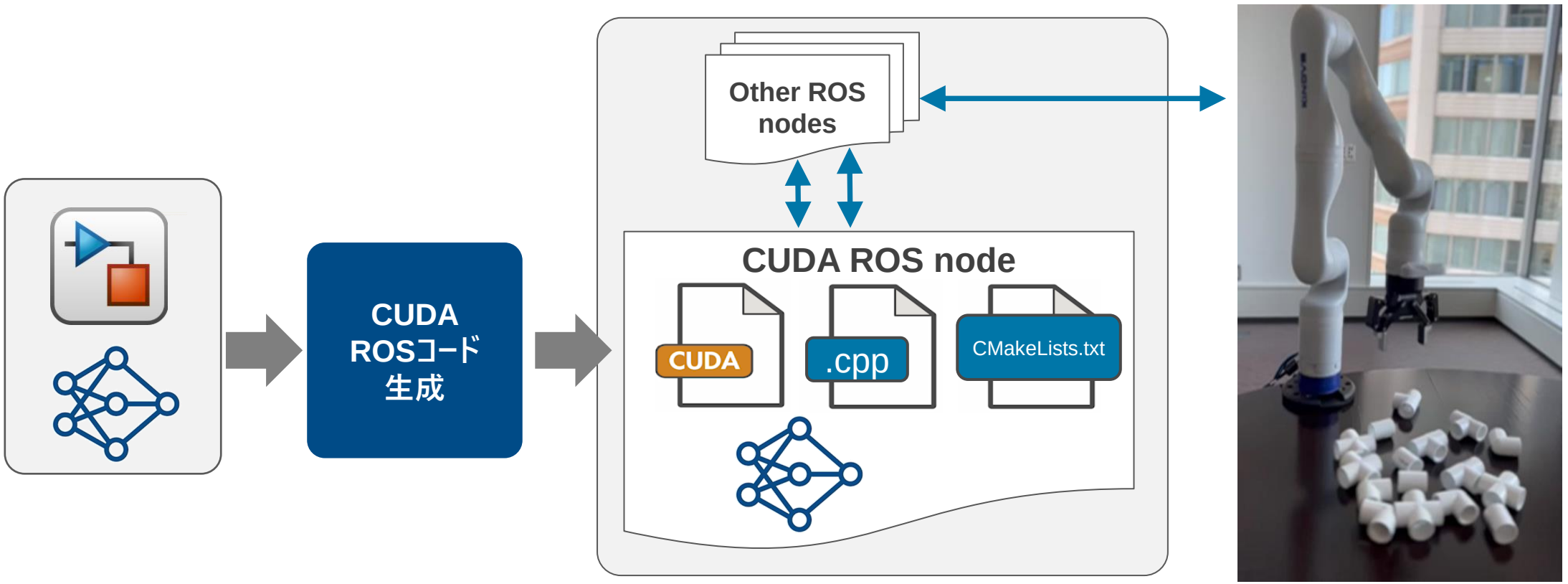
- ROSによるロボット制御
- GazeboとMATLABをROS経由で接続



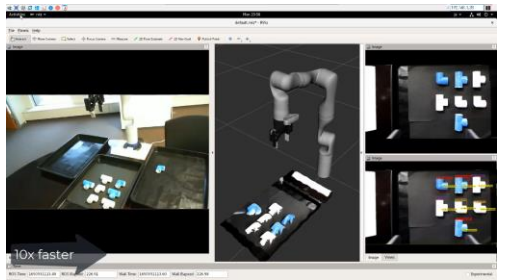
Kinova

- ROS、または、MATLAB用APIによるロボット制御
- MATLABのUnreal Engine連携機能を活用

AIモデルをCUDA ROSノードとしてJetsonに実装



NVIDIA® Jetson™



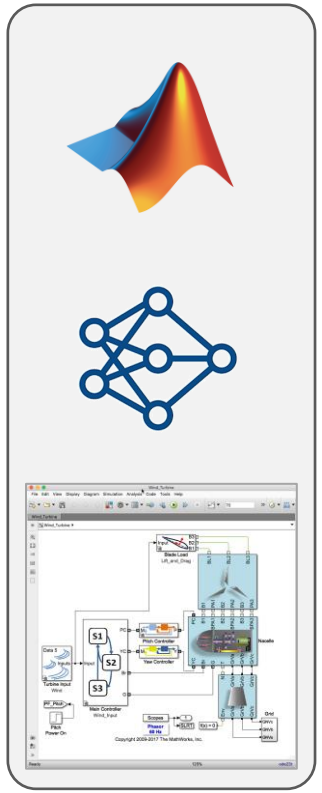
エッジデバイスへの実装例：SparseInstによるInstance segmentation

ONNX

80種類の物体をピクセル単位で検出可能な最新モデルを取り込み
エッジデバイスへの実装までサポート

各種ハードウェアへのAIアルゴリズムの実装

1. Data Preparation
2. AI Modeling
3. Simulation and Test
4. Deployment

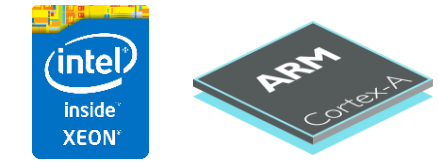


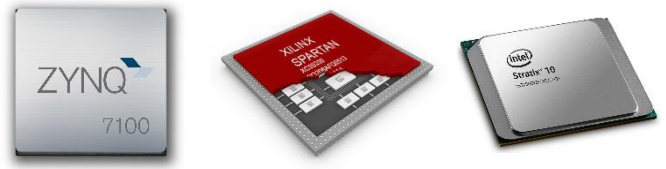
Code Generation

CPU

GPU

FPGA





詳細は次のセッションでご紹介

まとめ

1. ロボットに必要な要素技術は多様
   →集中したい要素技術が1つであっても
   ロボットシステム全体の準備・検証が必要
   ⇒MATLAB/Simulinkには豊富なロボットライブラリ・例題がご用意

2. AIの学習・検証に必要な**教師データの用意にかかる労力大**
   ⇒MATLAB/Simulinkの3Dシミュレーション活用で労力を最小化

本日のアンケート　QRコード

アンケートご記入の方に
講演資料を配布します