

MATLAB EXPO 2016

Robotics Development Workflow with MATLAB and Simulink

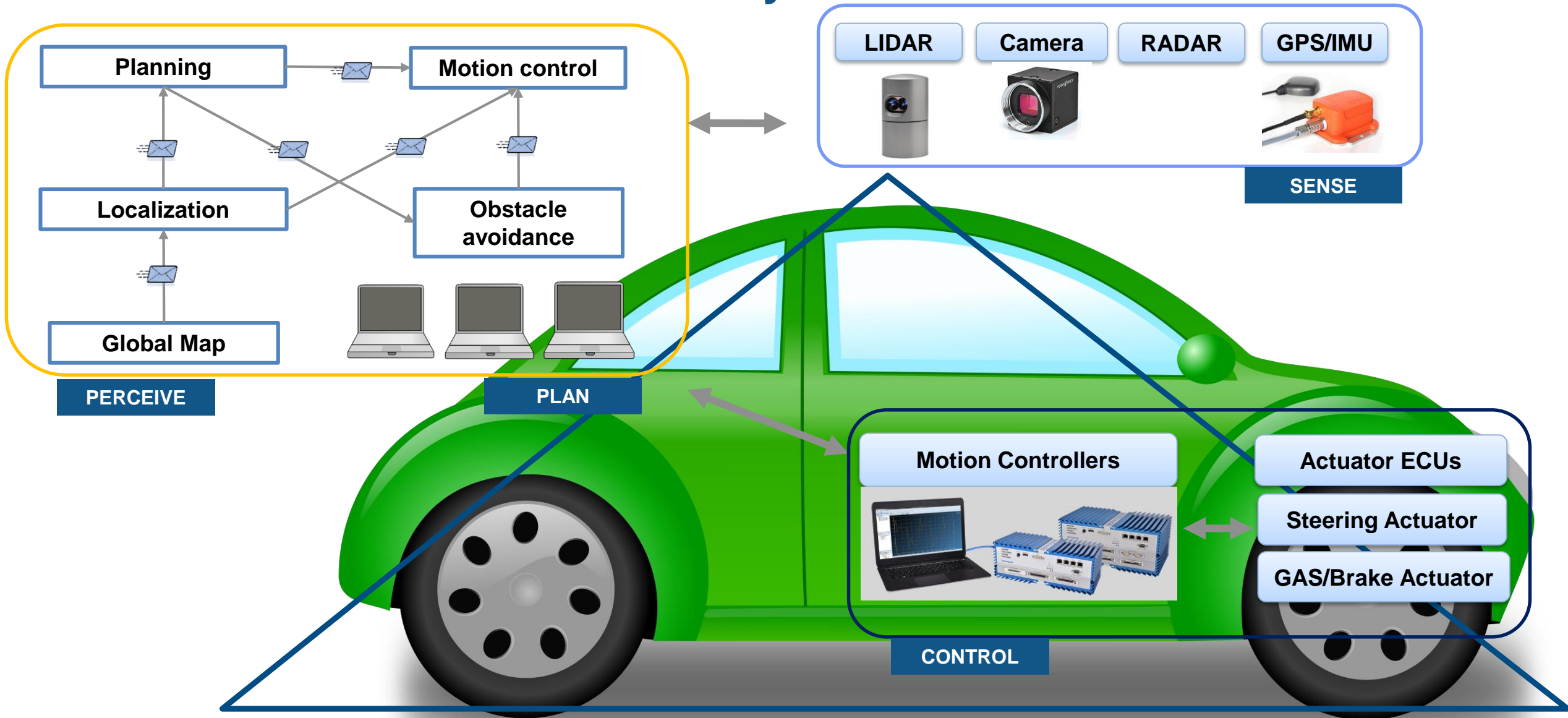
Carlos Santacruz-Rosero, Ph.D
Sr Application Engineer - Robotics



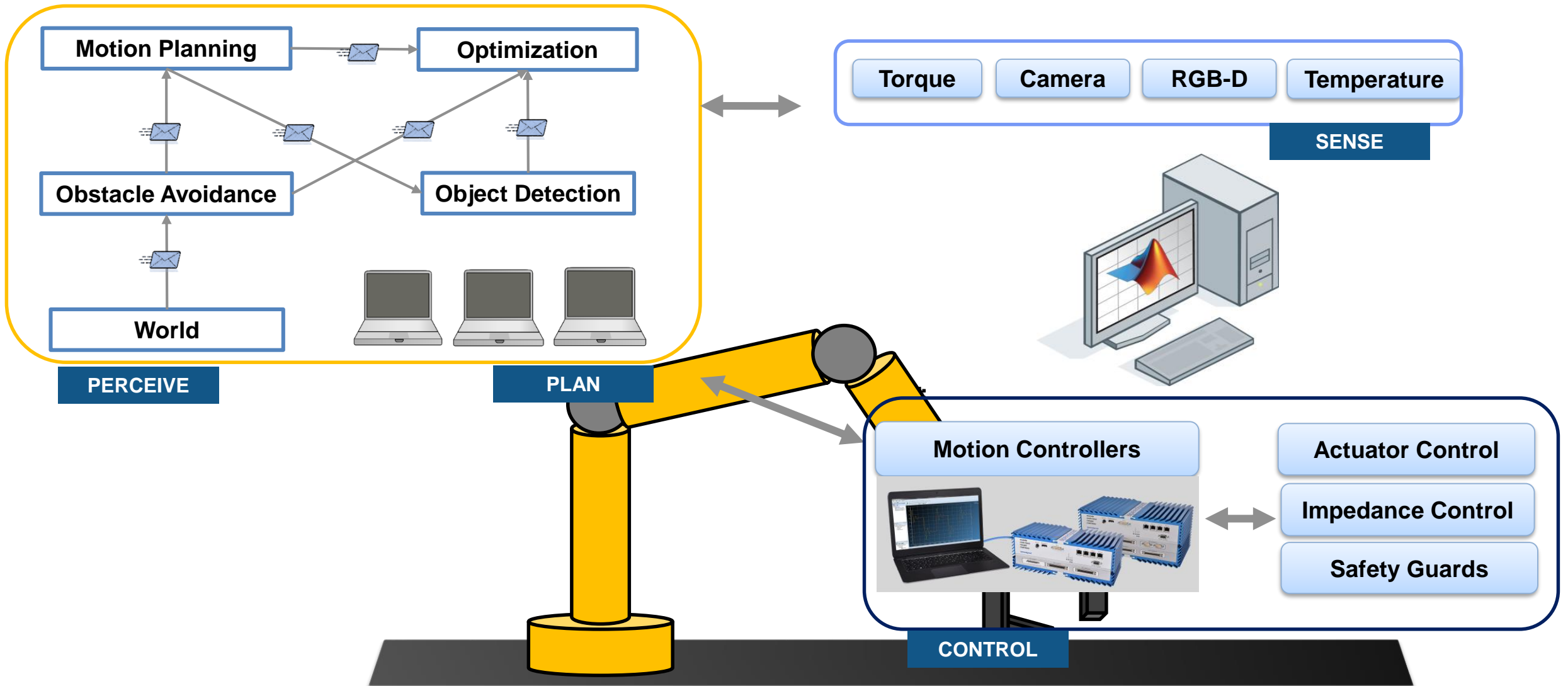
Agenda

- Introduction
- Advanced Robotics Systems
- Robotics Development Workflow with MATLAB and Simulink
- Takeaways

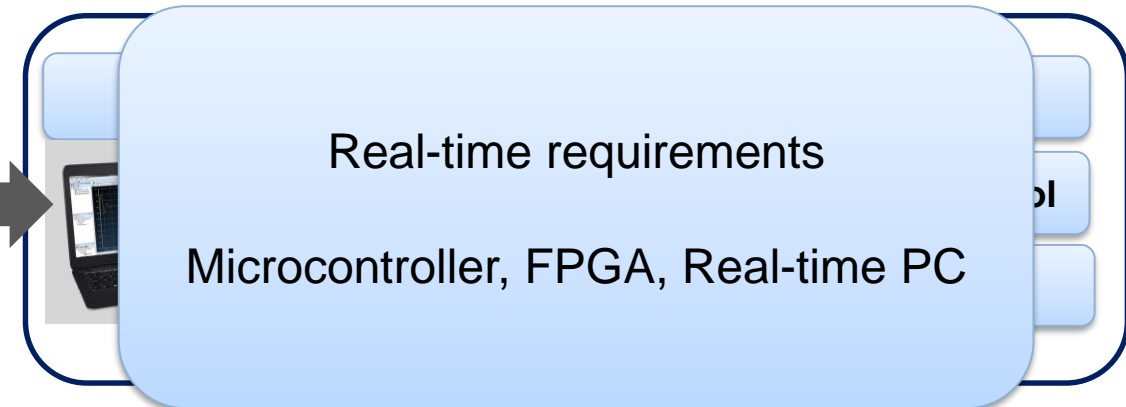
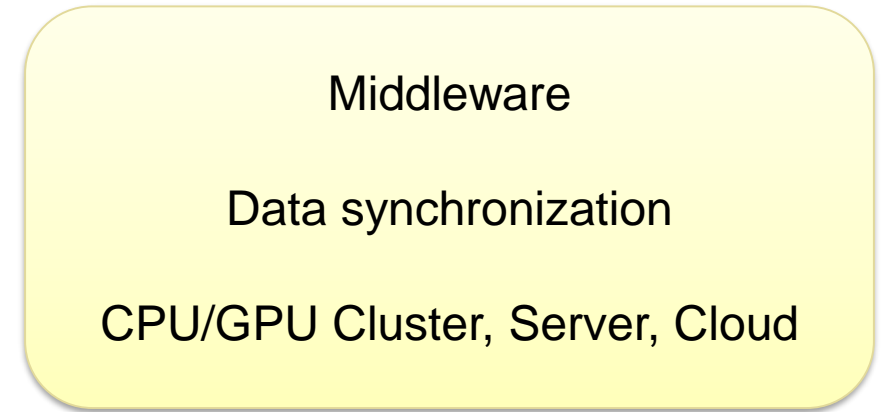
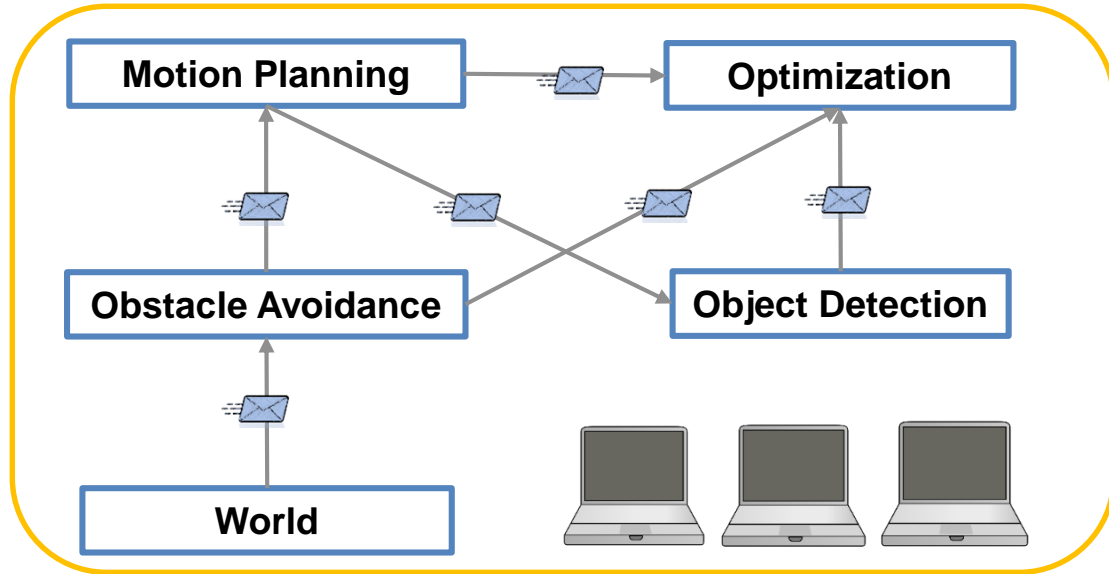
Car as an Advanced Robotics System



Collaborative Robot as an Advanced Robotics System



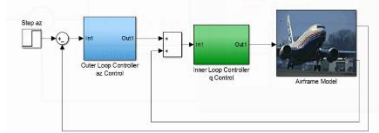
Architecture - Advanced Robotics System



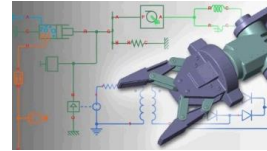
Technology Solutions for Autonomous Systems

CONTROL
SENSE
PERCEIVE
PLAN
CONNECT

Control System Tbx



Simscape Toolboxes



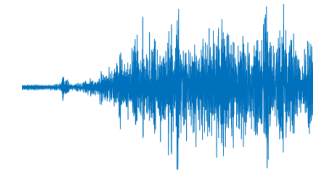
Simulink Real-Time



HW Support Packages



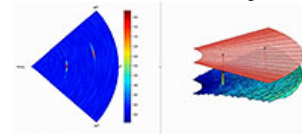
Data Acquisition Tbx



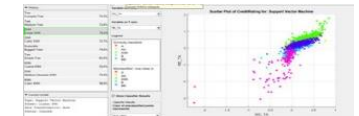
Computer Vision



Phased Array



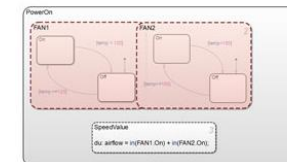
Statistics & Machine Learning



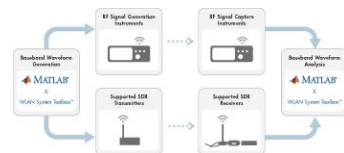
Robotics System Tbx



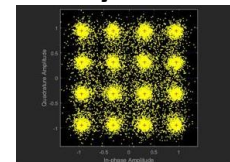
Stateflow



Communications Tbx



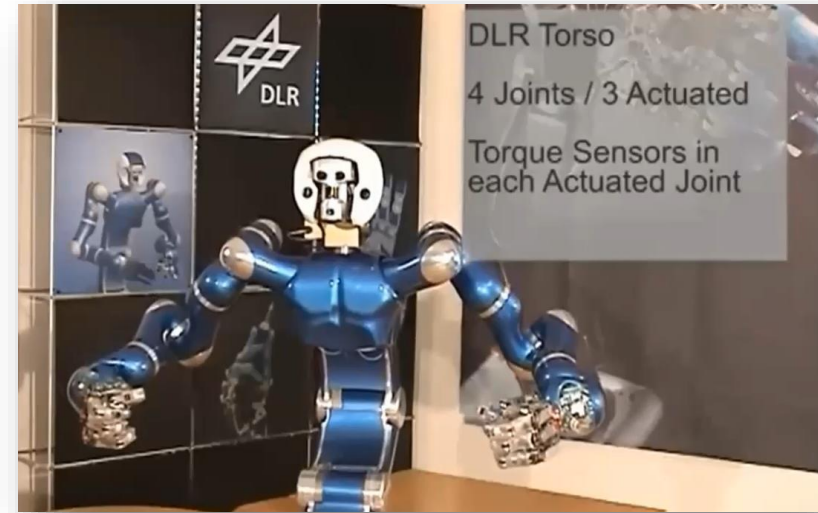
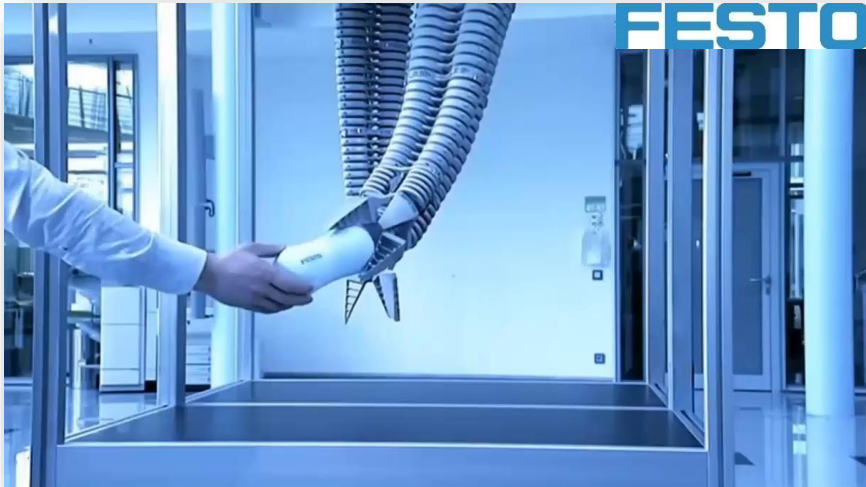
WLAN System Toolbox



Robotics System Tbx



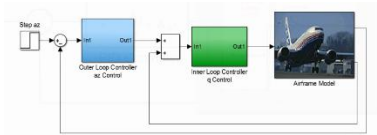
Success Stories



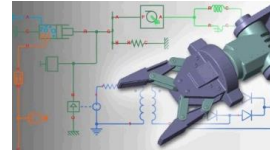
Technology Solutions for Autonomous Systems

CONTROL
SENSE
PERCEIVE
PLAN
CONNECT

Control System Tbx



Simscape Toolboxes



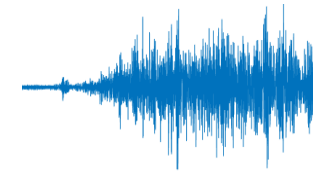
Simulink Real-Time



HW Support Packages



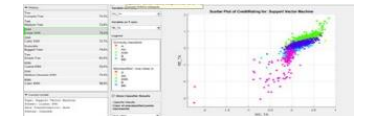
Data Acquisition Tbx



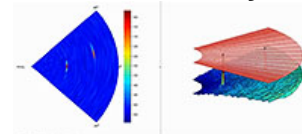
Computer Vision



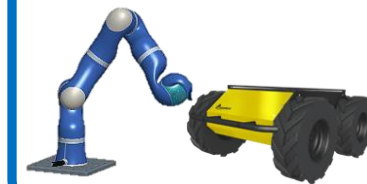
Statistics & Machine Learning



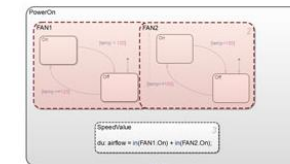
Phased Array



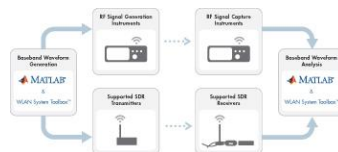
Robotics System Tbx



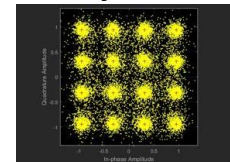
Stateflow



Communications Tbx



WLAN System Toolbox



Robotics System Tbx



Robotics System Toolbox

Interface and
Deployment to
ROS

ROS

Algorithms for
Mobile
Robotics

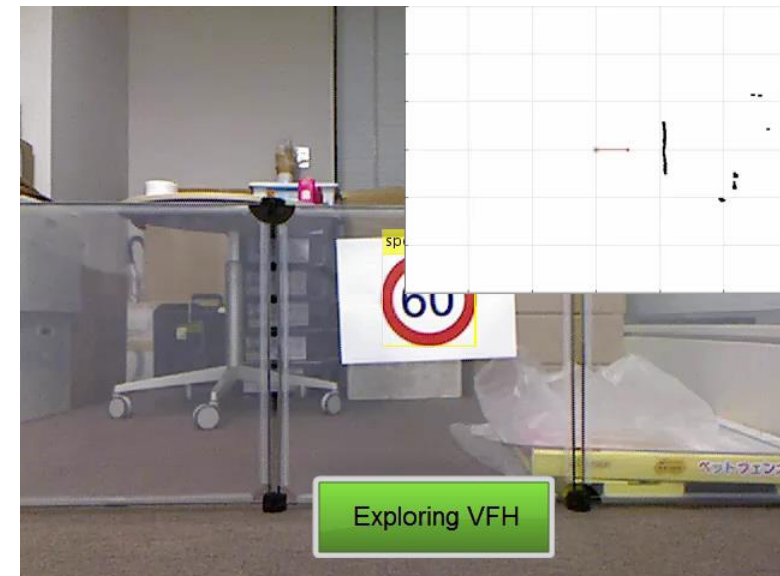
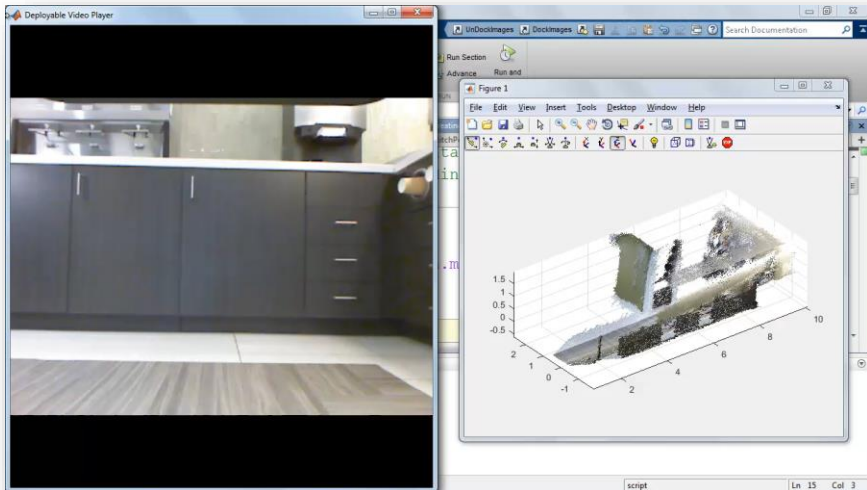
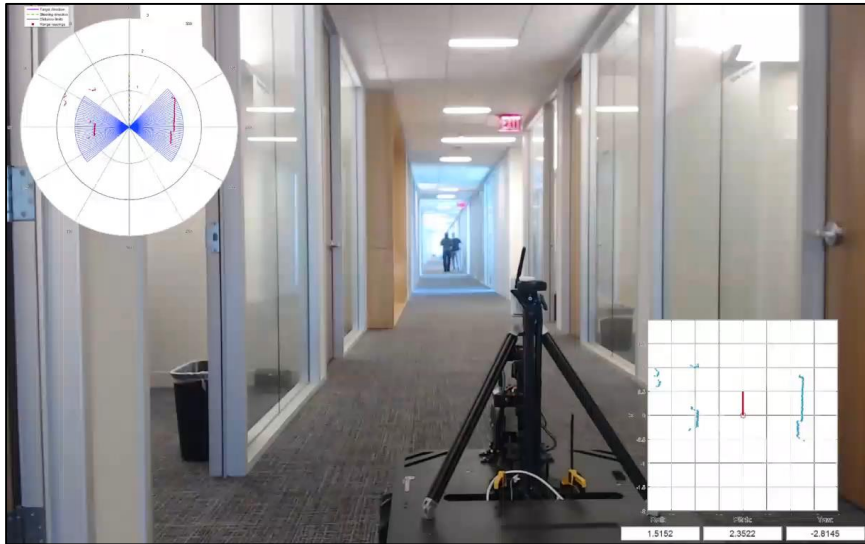


Algorithms for
Manipulators and
Humanoids

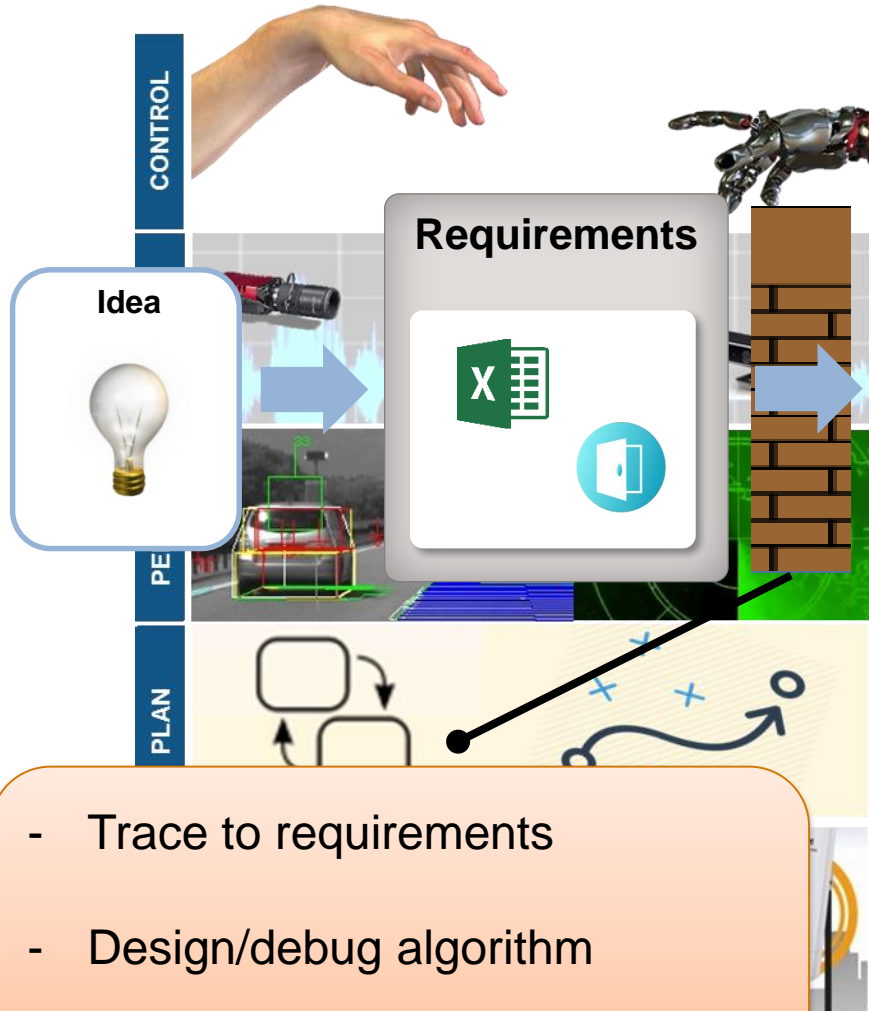


*Environment for **prototyping**, **simulating**, and **deploying** robotics applications*

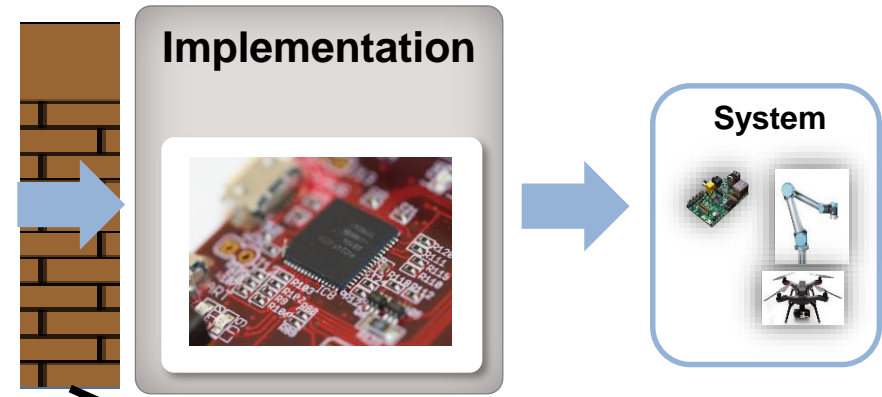
Robotics Applications with Robotics System Toolbox



Workflow Convergence is Needed



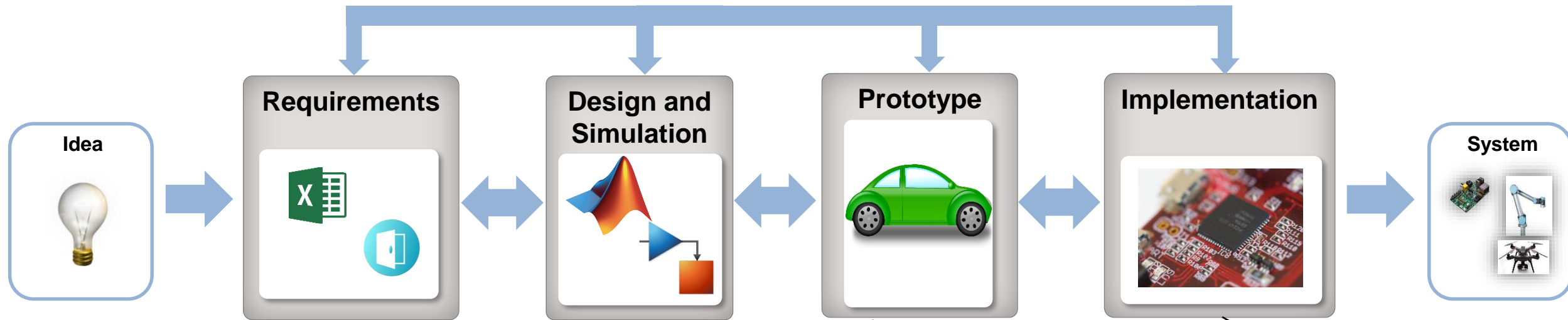
- Trace to requirements
- Design/debug algorithm
- System integration



- Testing on physical system
- Manual Coding
- Verification and Validation

Robotics Development Workflow with MATLAB and Simulink

RAPID ITERATIVE PROCESS



- Built-in algorithms and apps
- System-Level Simulation MBD
- Co-simulation

- C/C++ automatic code generation
- Processor-in-the-loop (PIL)
- Debug C/C++ with MATLAB Engine

Design independent of hardware implementation!

Let's solve a real problem: Sign Detection System

Clearpath Husky Robot

- ROS Enabled
- Kinect (RGB and Point cloud)
- Hokuyo 2D Lidar



Simulation model in Gazebo

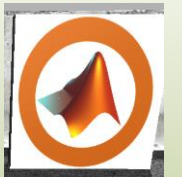
Track and Park



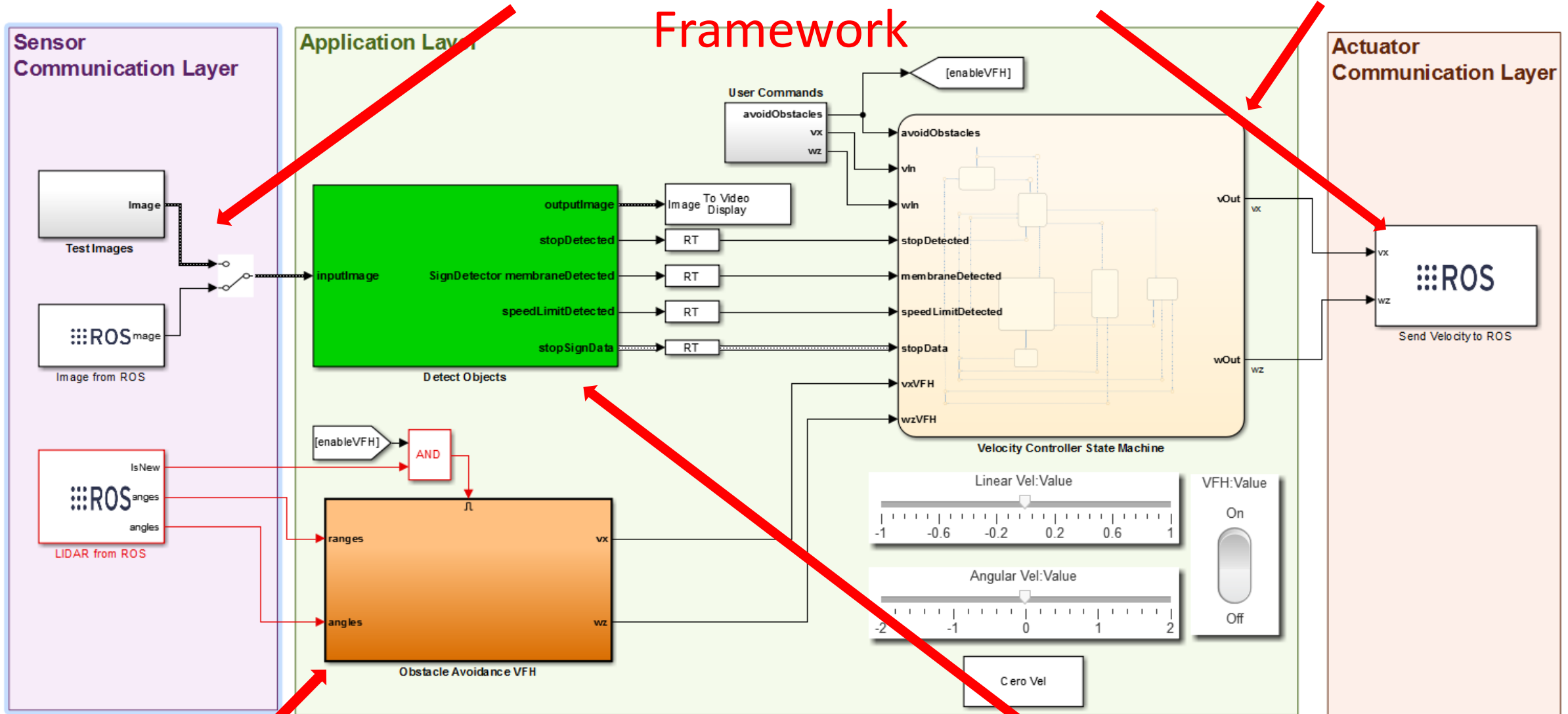
Reduce Speed



Collision Avoidance



System Level Design ROS as Communication Framework State Machine



Obstacle Avoidance

Object Classifier

Sign Recognition with Collision Avoidance

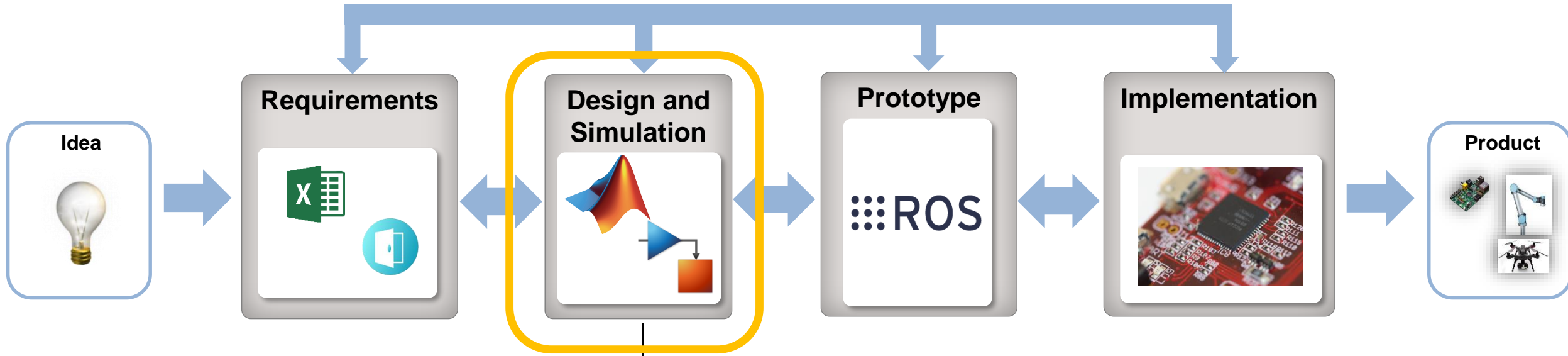
The image displays a Simulink control system for a robot in a Gazebo environment. The Simulink model, titled 'testSignDetector', consists of several interconnected blocks:

- Detect Objects:** A green block that processes 'inputImage' and outputs 'outputImage', 'stopDetected', 'speedLimitDetected', and 'stopSignData'.
- Obstacle Avoidance VFH:** An orange block that receives 'range' and 'angle' inputs and outputs 'vsVFH' and 'wsVFH'.
- Velocity Controller State Machine:** A yellow block that receives 'vsVFH' and 'wsVFH' and outputs 'vOut' and 'wOut'.
- Control Elements:** Includes an 'AND' gate, a 'Zero Vel' button, and two sliders for 'Linear Vel. Value' (ranging from -1 to 1) and 'Angular Vel. Value' (ranging from -2 to 2). A 'VFH.Value' toggle switch is also present.

The Gazebo simulation window shows a robot in a virtual environment with various obstacles, including a speed limit sign (60), a warning sign, and a traffic cone. A red laser line is visible, representing the robot's field of view or sensor range. The bottom status bar of the Gazebo window shows 'Real Time Factor: 0.57', 'Sim Time: 00:01:28.330', 'Real Time: 00:00:13.28.632', 'Iterations: 62017', and 'FPS: 15'.

Robotics Development Workflow with MATLAB and ROS

RAPID ITERATIVE PROCESS



- Import logged data
- Train a classifier
- Test component

Importing Simulation and Experimental Data

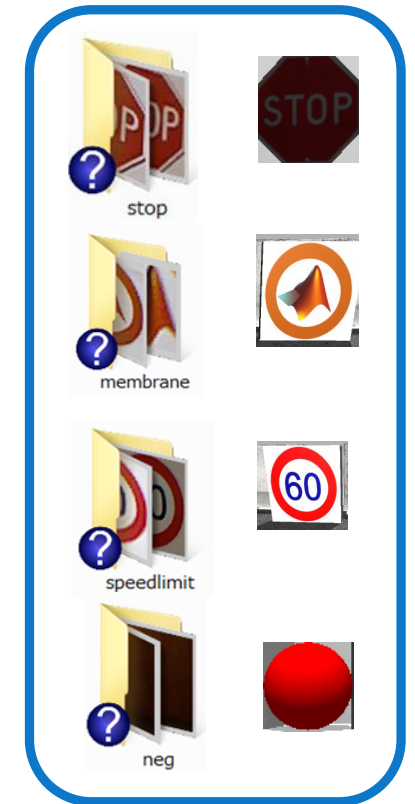
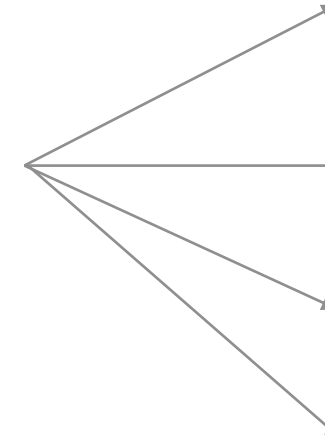


Experimental data or simulation Data

Import ROS Data



Filter your logged field data by topic or time interval



Data ready to design algorithms

```

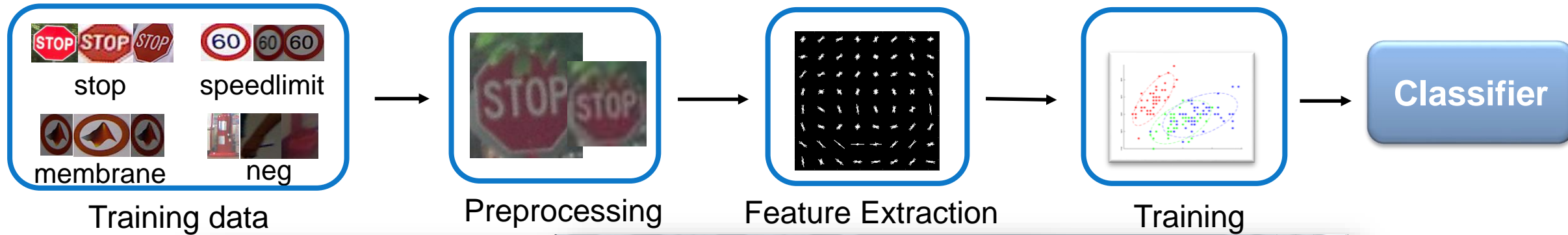
%% Load the file
carData = rosbag('\car_field_test_042016.bag');

%% Select all messages on the scan topic
odomMsg = select(carData, 'Topic', '/scan');

%% Get all RGB camera points
imagMsg = select(carData, 'Topic', '/camera/rgb/image_raw');
    
```

Robotics System Toolbox™

Visualize, Analyze, and Process Data: Classifier



Input

```

% Detect red
BW = createM...

% Fill image
BW = imfill(B...

% Get boundin
stats = regio...

% Filter base
targetIndex =

% Get boundin
testFeatures(
        
```

Classification Learner - Confusion Matrix

Model 1

True Class \ Predicted class	1	2	3	4
1	26			
2		491		1
3			57	
4		5		173

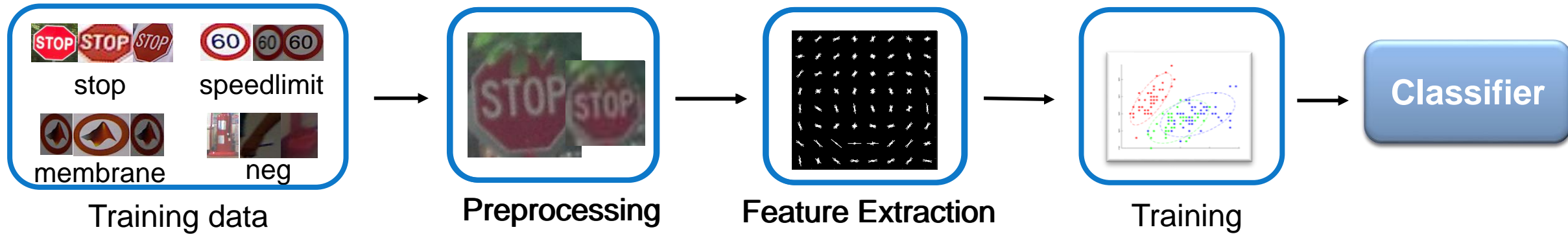
Model 1 Accuracy: 99.2%

Original Dataset: trainingData Observations: 753 Predictors: 1764 Response Variable: column_1765 Response Classes: 4 Size of Dataset: 5 MB Validation: 5-fold Cross Validation



Output

Visualize, Analyze, and Process Data: Classifier



Computer Vision System Toolbox™

Statistics and Machine Learning Toolbox™

Design and Test Algorithm

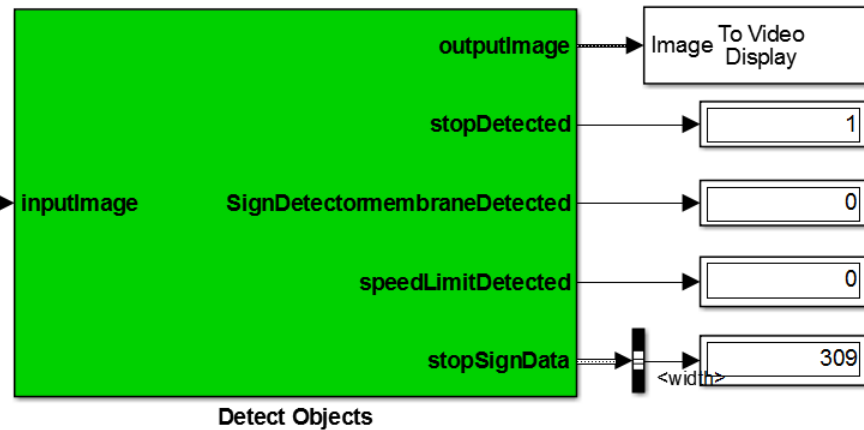
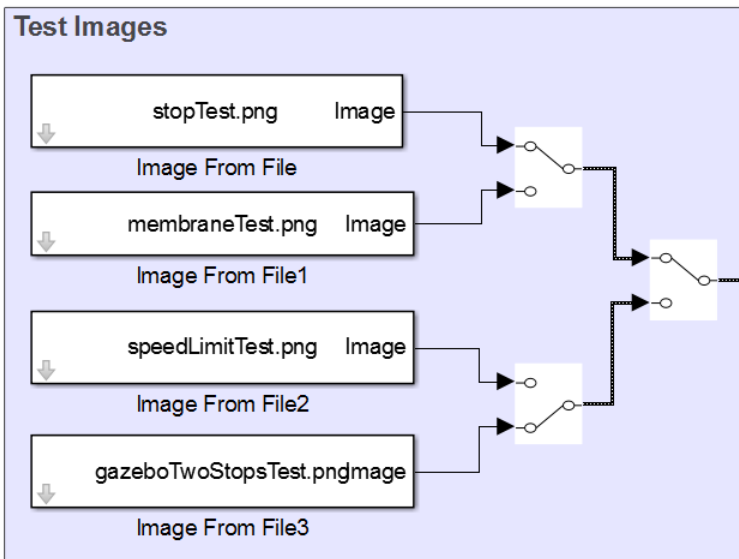
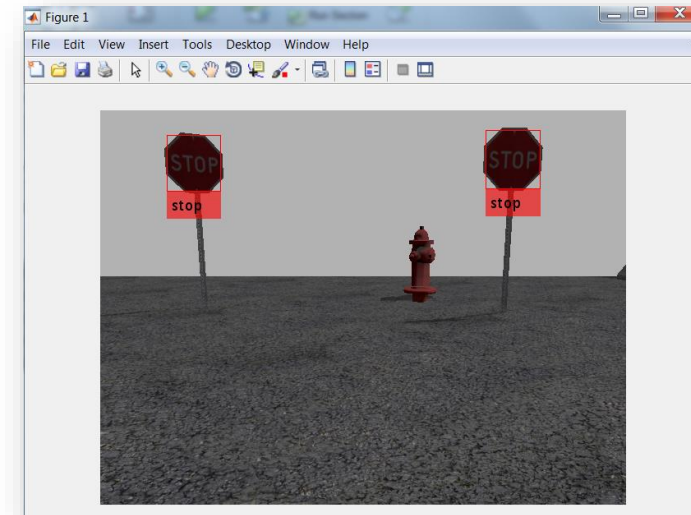
```

%% Create classifier object
sd = SignDetector();

%% Test algorithm with two stops signs
Iin = imread('gazeboTwoStopsTest.png');

%% Show algorithm result
[Iout,~,~,~,~] = sd.step(Iin);
imshow(Iout);

```



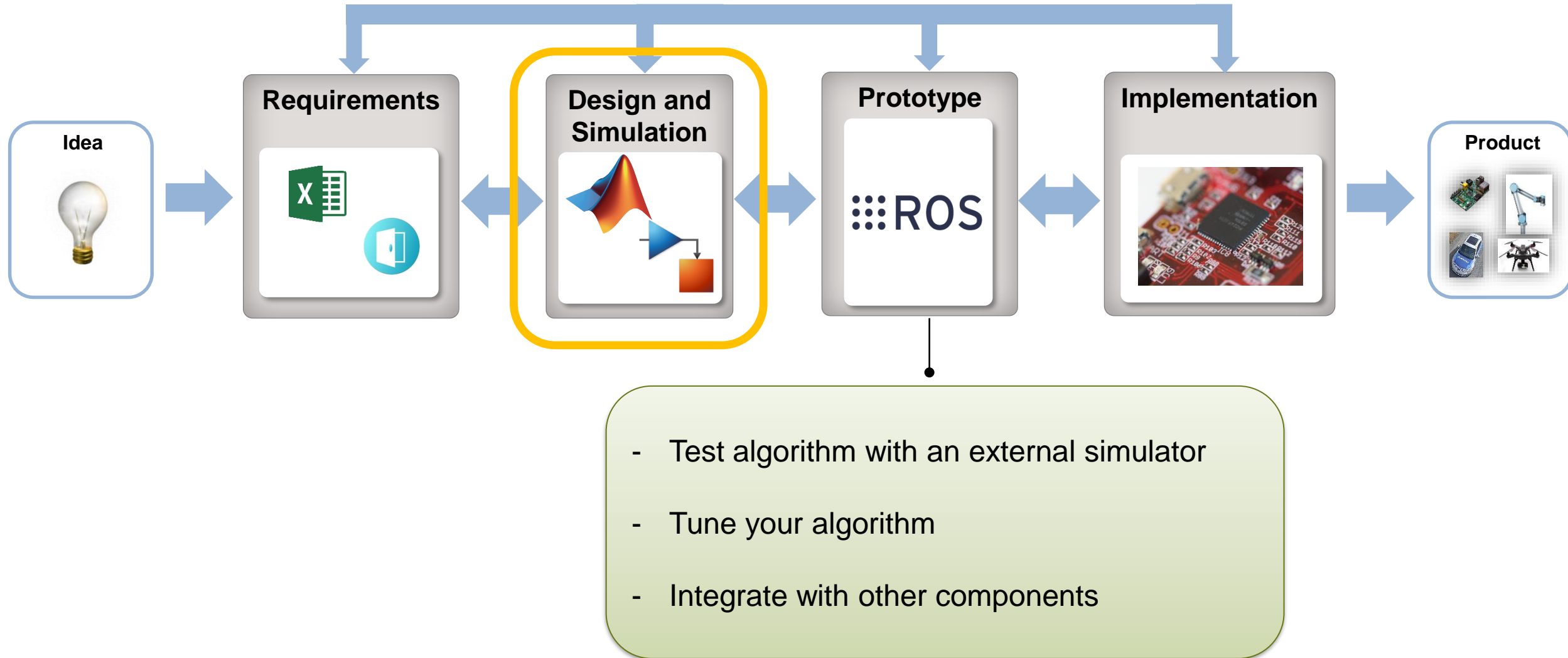
Design and Test Algorithm

The screenshot displays the Simulink environment for a model named 'testSignDetectorWithImages'. The model diagram includes a 'Test Images' block containing four 'Image From File' blocks: 'stopTest.png', 'membraneTest.png', 'speedLimitTest.png', and 'gazeboTwoStopsTest.png'. These images are routed to a 'Detect Objects' block (green). The 'Detect Objects' block has four outputs: 'outputImage', 'stopDetected', 'membraneDetected', and 'speedLimitDetected'. The 'outputImage' is connected to an 'Image To Video Display' block. The 'stopDetected' output is connected to a scope showing the value '1'. The 'membraneDetected' output is connected to a scope showing '0'. The 'speedLimitDetected' output is connected to a scope showing '0'. The 'stopSignData' output is connected to a scope showing '309'. The status bar at the bottom indicates 'Running' with 'View diagnostics 86% T=0.300' and 'FixedStepDiscrete'.

Overlaid on the right is a 'To Video Display1' window showing a video feed of a red octagonal stop sign with the word 'STOP' in white. The sign has some graffiti on it.

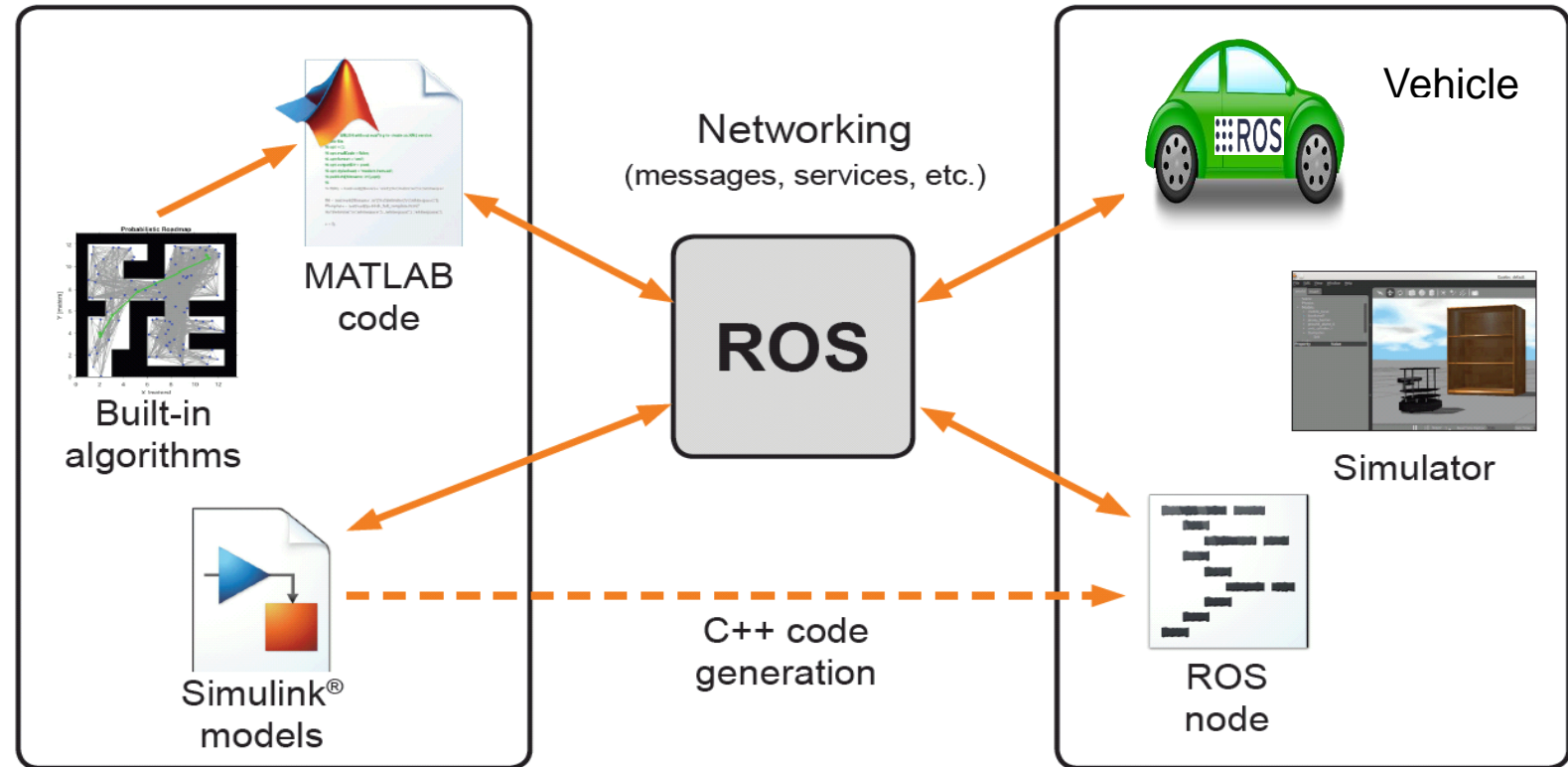
Robotics Development Workflow with MATLAB and ROS

RAPID ITERATIVE PROCESS



MATLAB and Simulink connect to the ROS network

- Multiple master support
- ROS publishers/subscribers
- ROS services
- ROS TF tree
- ROS Parameter server



Co-simulation with ROS

```
%% Connect to ROS
```

```
rosinit('192.168.204.144');
```

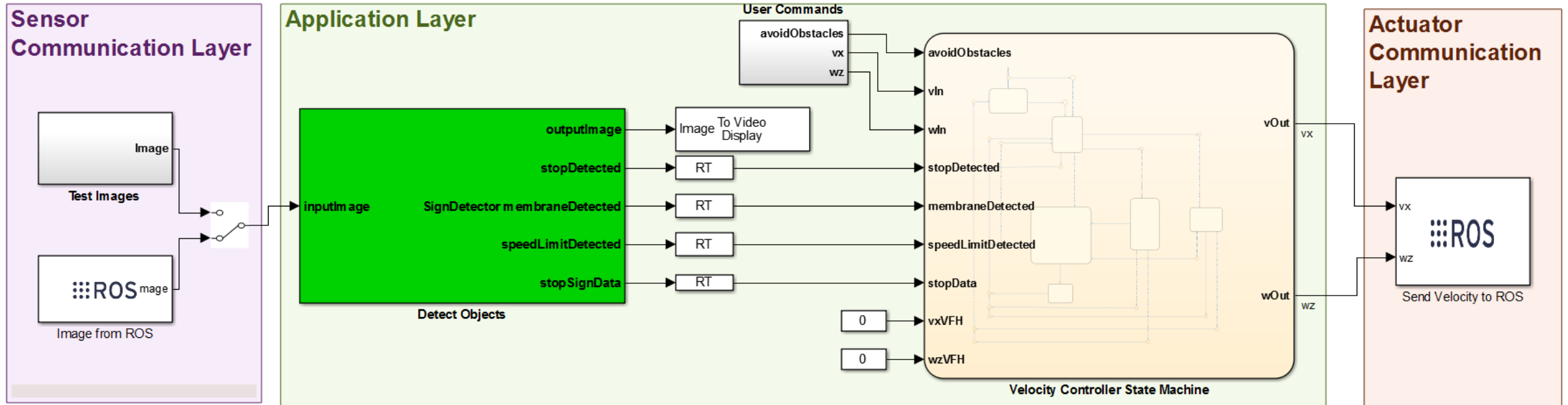
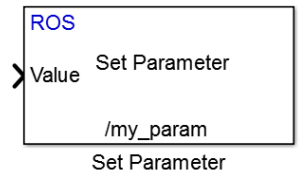
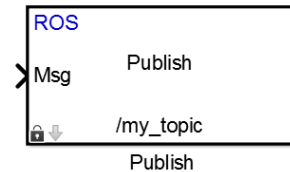
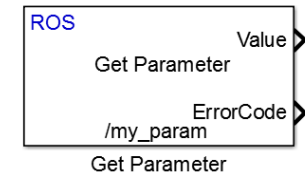
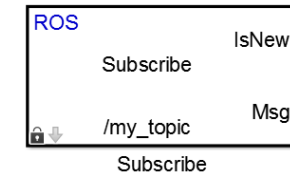
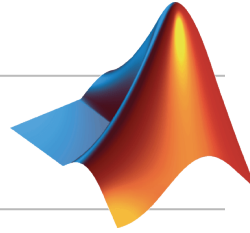
```
%% Create subscribers
```

```
imSub = rossubscriber('/camera/rgb/image_raw');
```

```
scanSub = rossubscriber('/scan');
```

```
%% Create publisher
```

```
[velPub, velMsg] = rospublisher('/husky_velocity_controller/cmd_vel');
```

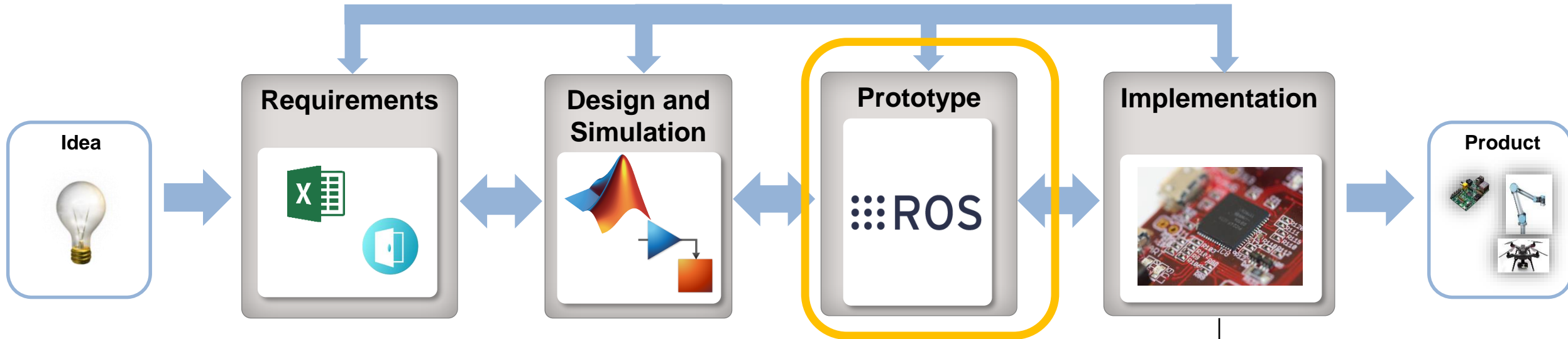


Co-simulation with ROS

The image displays a co-simulation setup between Simulink and Gazebo. The Simulink window, titled 'testSignDetectorWithGazebo * - Simulink', shows a 'Velocity Controller State Machine' block. This block receives several inputs: 'vx', 'vz', and 'wz' from an 'avoidObstacles' block; 'stopDetected', 'membraneDetected', 'speedLimitDetected', and 'stopSignData' from Real-Time (RT) blocks; and 'vxVFH' and 'wzVFH' from other blocks. The block outputs 'vOut' (0.3) and 'wOut' (0). Below the block, a 'Cero Vel' block and 'VFH: Value' indicator are visible. The Gazebo window shows a 3D simulation of a robot in an environment with a 60 speed limit sign and a stop sign. A 'Video Display1' window shows a top-down view of the robot's field of view.

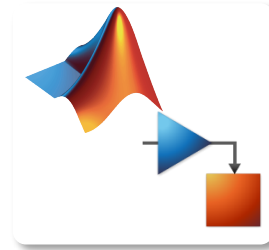
Robotics Development Workflow with MATLAB and ROS

RAPID ITERATIVE PROCESS



System level design to target a different middleware or framework

Deployment



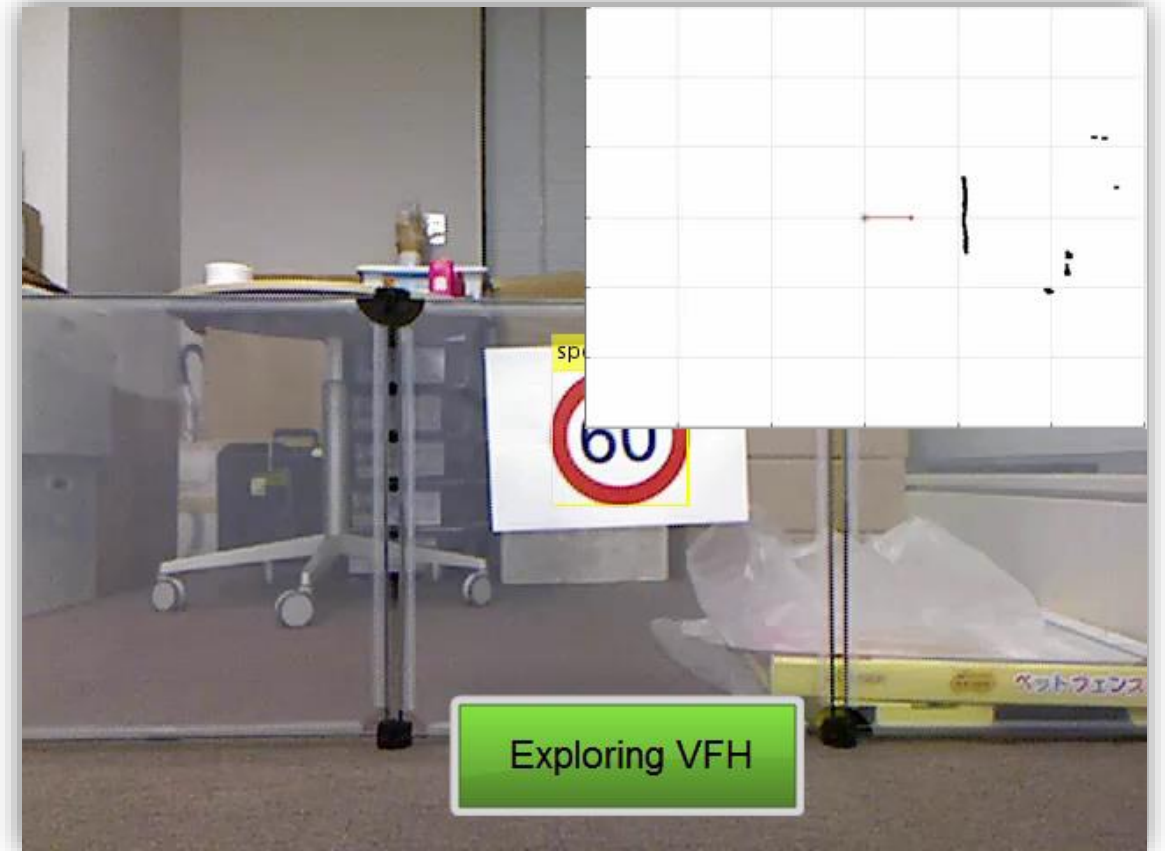
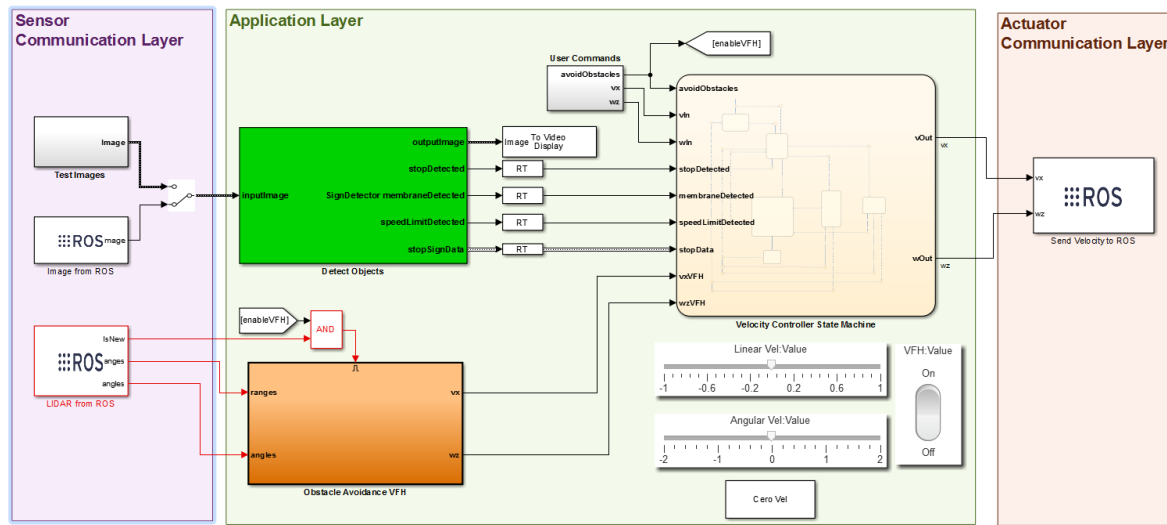
**Generate ROS
Node with
Simulink and
Embedded
Coder™**

**Generate a
shared library
with MATLAB
Coder™**

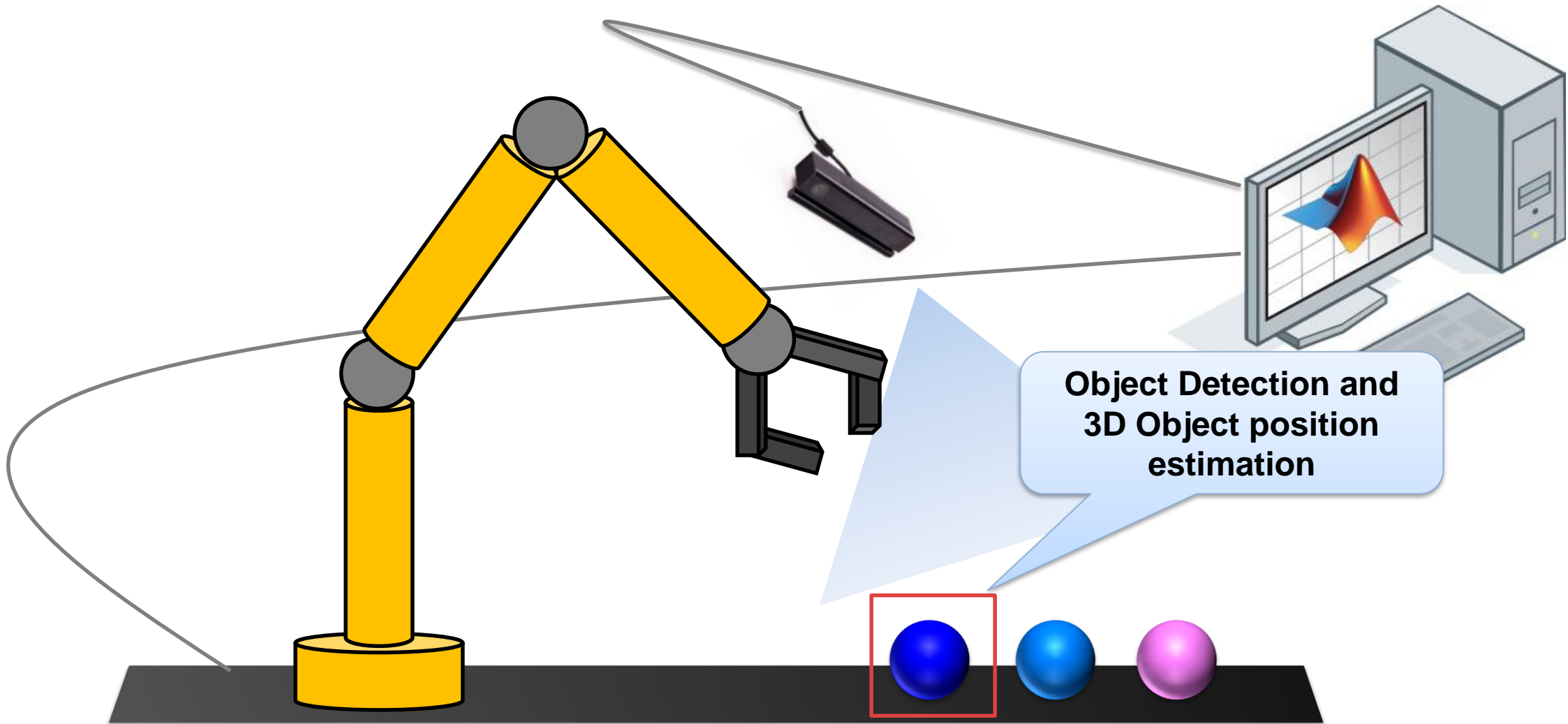
**Create a
Stand Alone
Executable
with MATLAB
Compiler™**

Determine deployment methods based on application

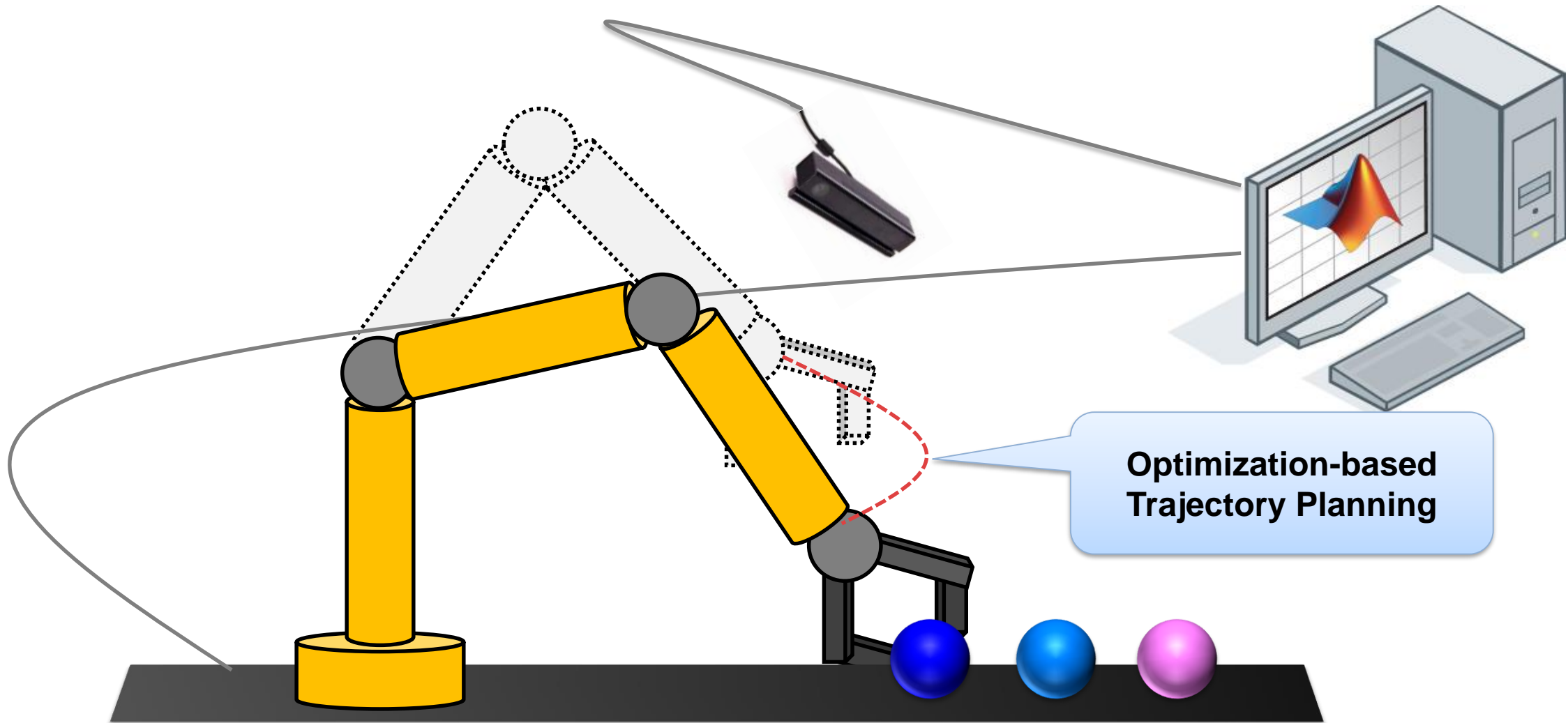
Implementation



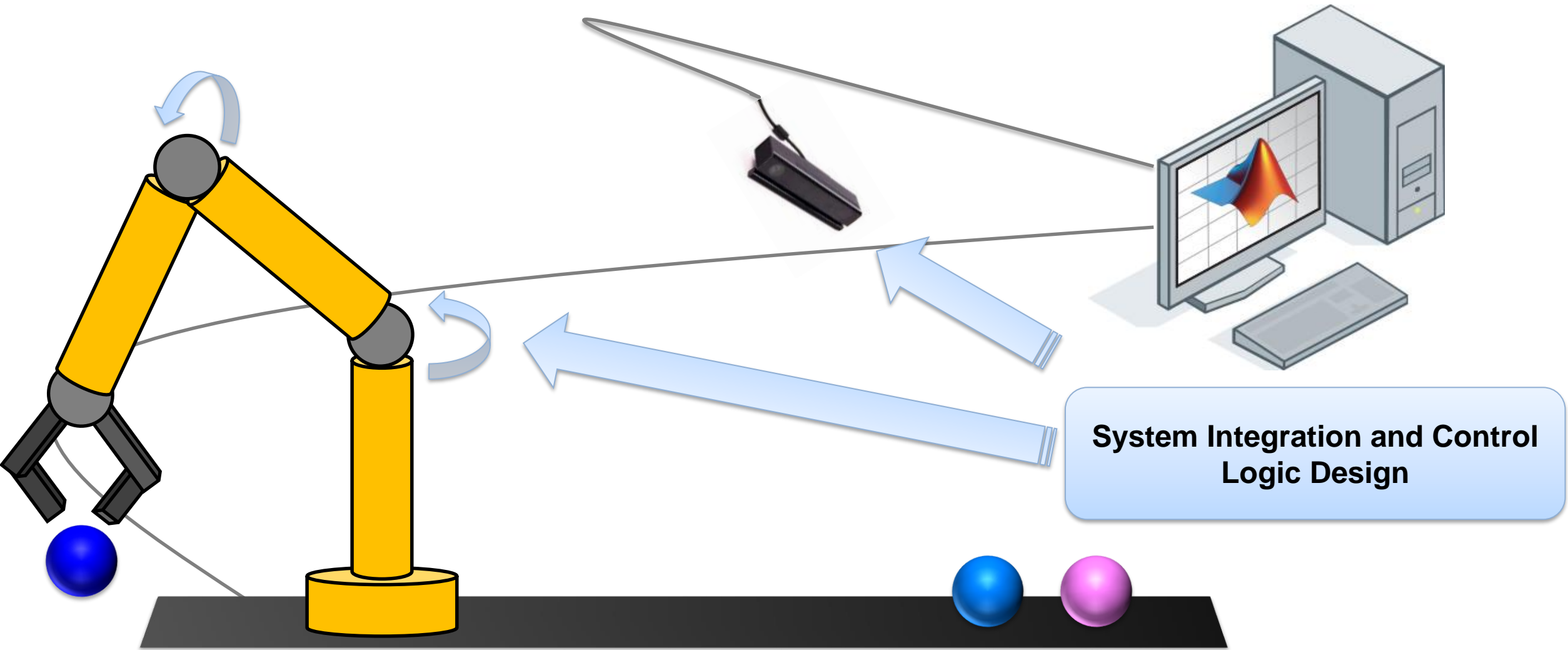
Part1: Object Detection and Position Estimation



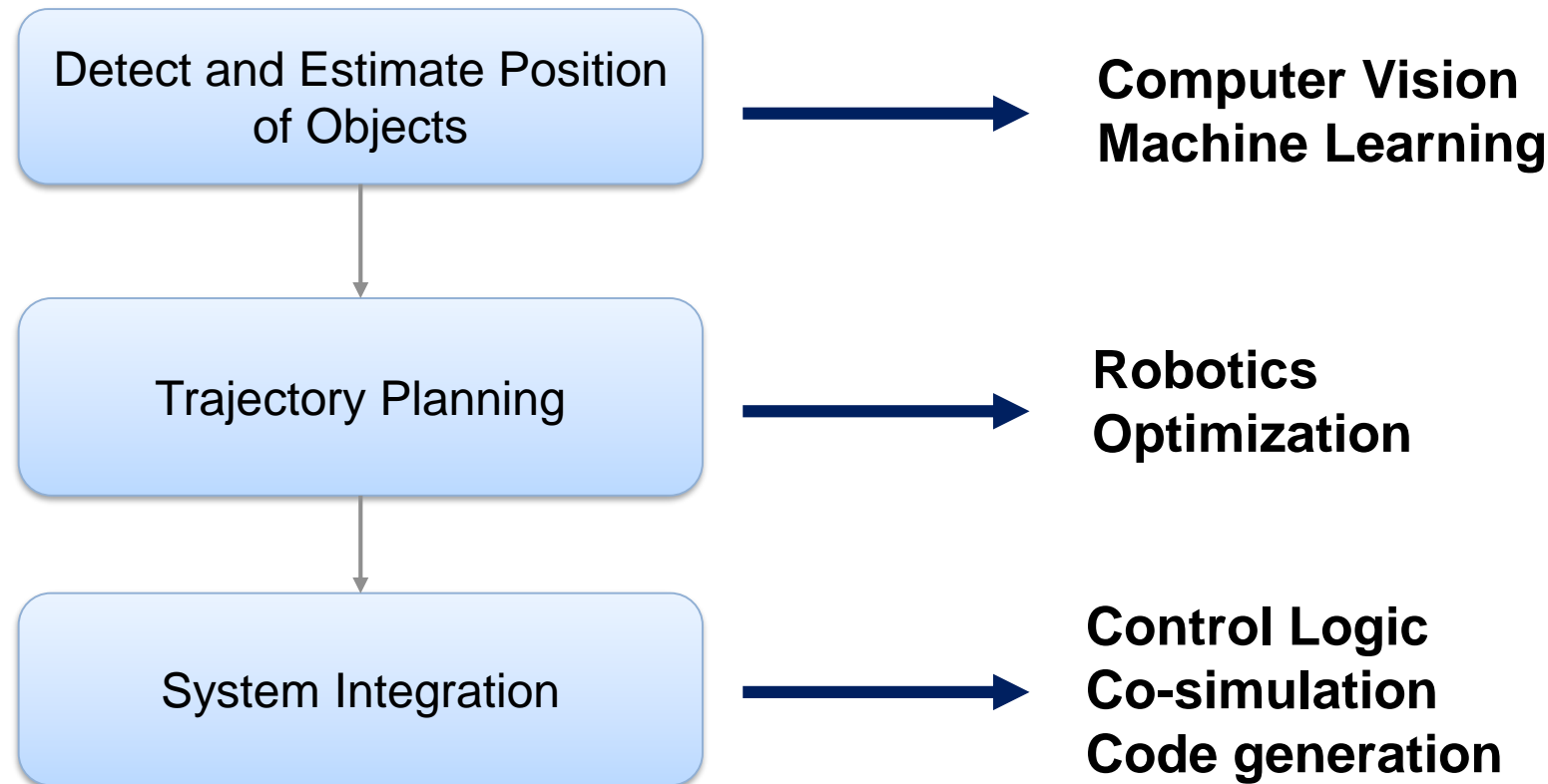
Part2: Trajectory Planning



Part3: System Integration



Advanced Robotics Application Requires Multiple Technologies



MATLAB and Simulink: very powerful tools to design advanced robotics applications

Trajectory Planning with RGB-D Sensor

The image displays a Simulink workspace for a robot control system. The main window shows a block diagram titled "Pick and Place Controller" with the following components and connections:

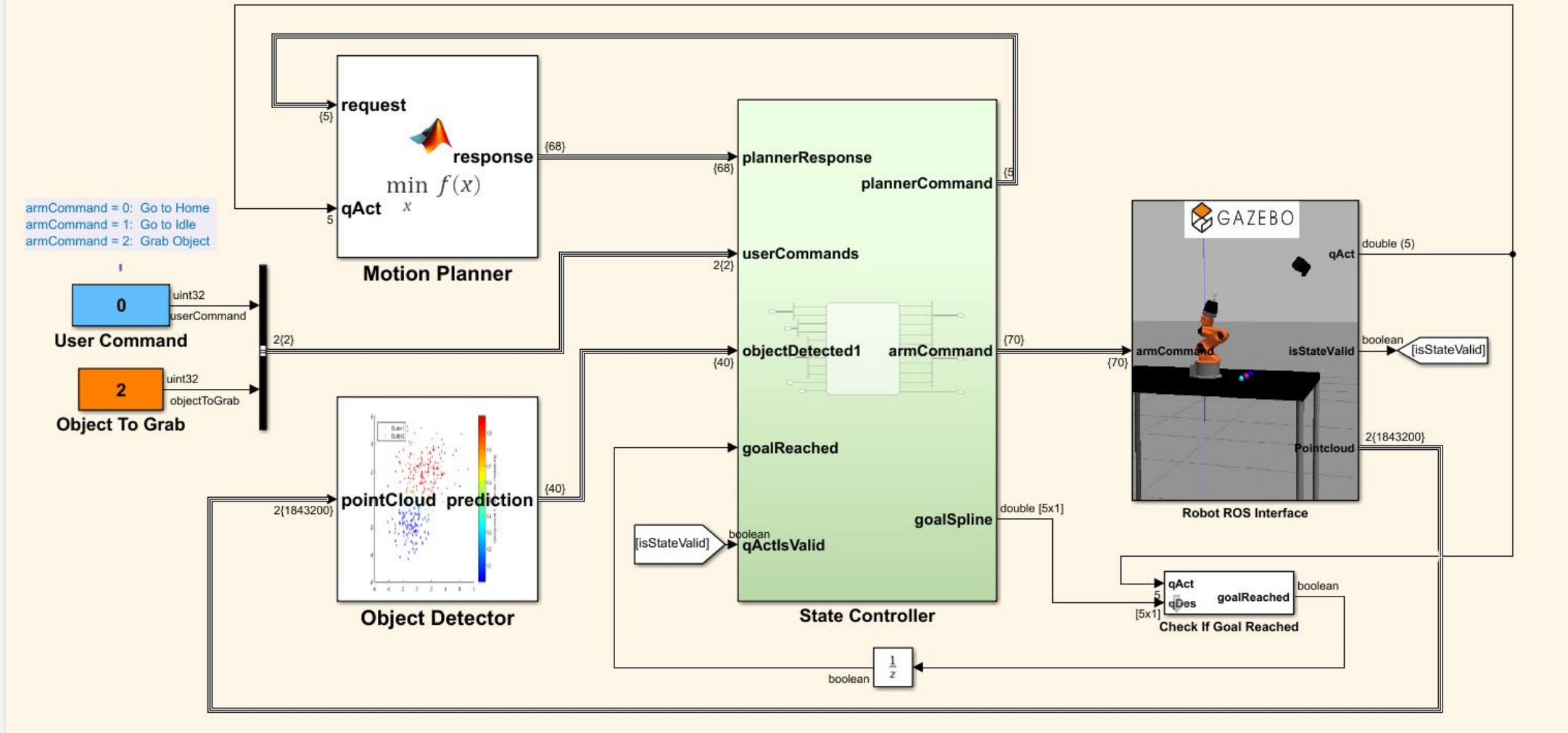
- User Command** and **Object To Grab** blocks provide input to the **Motion Planner**.
- The **Motion Planner** (containing a `min f(x)` block) sends a `request` to the **State Controller** and receives a `response` back.
- The **Object Detector** (receiving `pointCloud prediction`) sends `objectDetected1` to the **State Controller**.
- The **State Controller** sends `plannerCommand` back to the **Motion Planner** and outputs `userCommands` to the **Robot ROS Interface**.
- The **State Controller** also outputs `goalReached` and `goalValid` signals.
- The **Robot ROS Interface** block is connected to a **Robot** block, which in turn controls the **Robot ROS Interface** via a `Robot ROS Interface` block.

At the bottom of the Simulink window, the status bar shows "初期化中" (Initializing), "41%", and "FixedStepDiscrete".

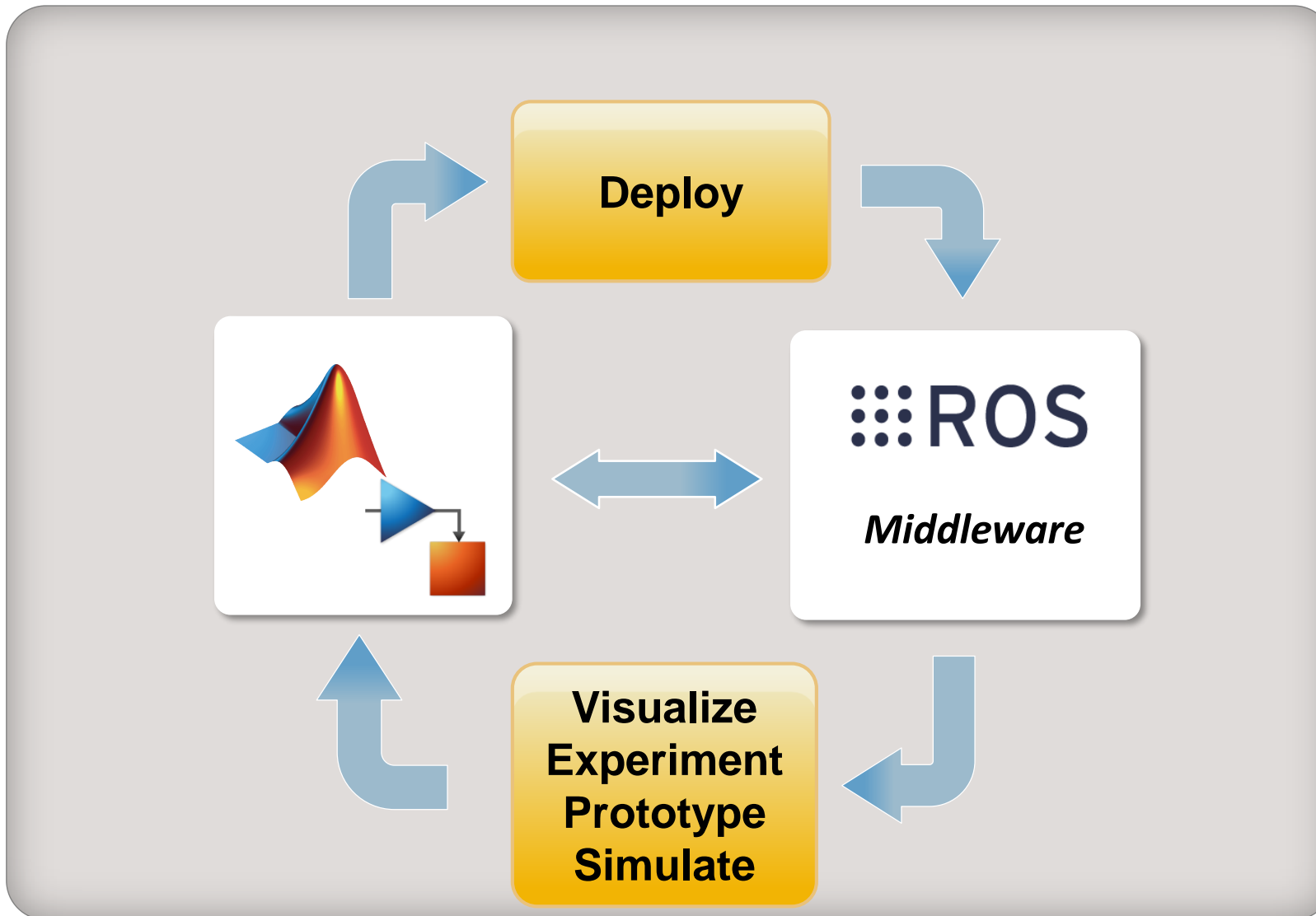
To the right of the Simulink window is a 3D simulation of an orange robotic arm in a virtual environment. The arm is positioned above a dark floor with several orange spheres scattered on it, representing the objects to be picked up.

System Level Design

Pick and Place Controller



Robotics Development Workflow with MATLAB and Simulink



Questions

```
% Thank you
```