

# Designing a Pick and Place Robotics Application Using MATLAB and Simulink



**Carlos Santacruz-Rosero, PhD**  
**Sr Application Engineer – Robotics**

**Pulkit Kapur**  
**Sr Industry Marketing Manager – Robotics**

## Key Takeaway of this Talk

Success in developing an autonomous robotics system requires:

- Multi-domain simulation
- Great tools which make complex workflows easy and integrate with other tools
- Model-based design

# Challenges with Autonomous Robotics Systems

Applying Multidomain Expertise

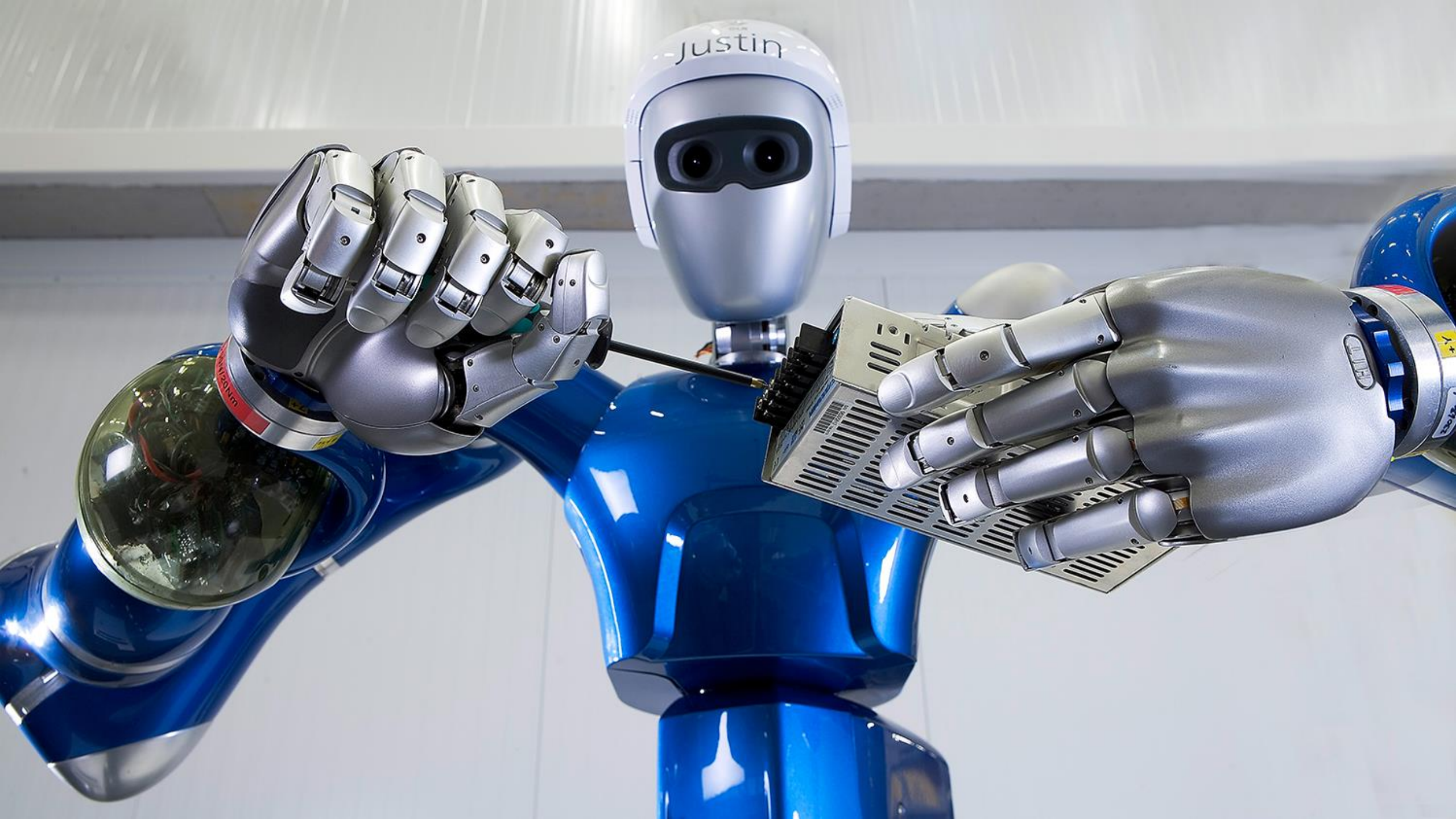
Complexity of Algorithms

End-to-End workflows

Technical Depth and System Stability

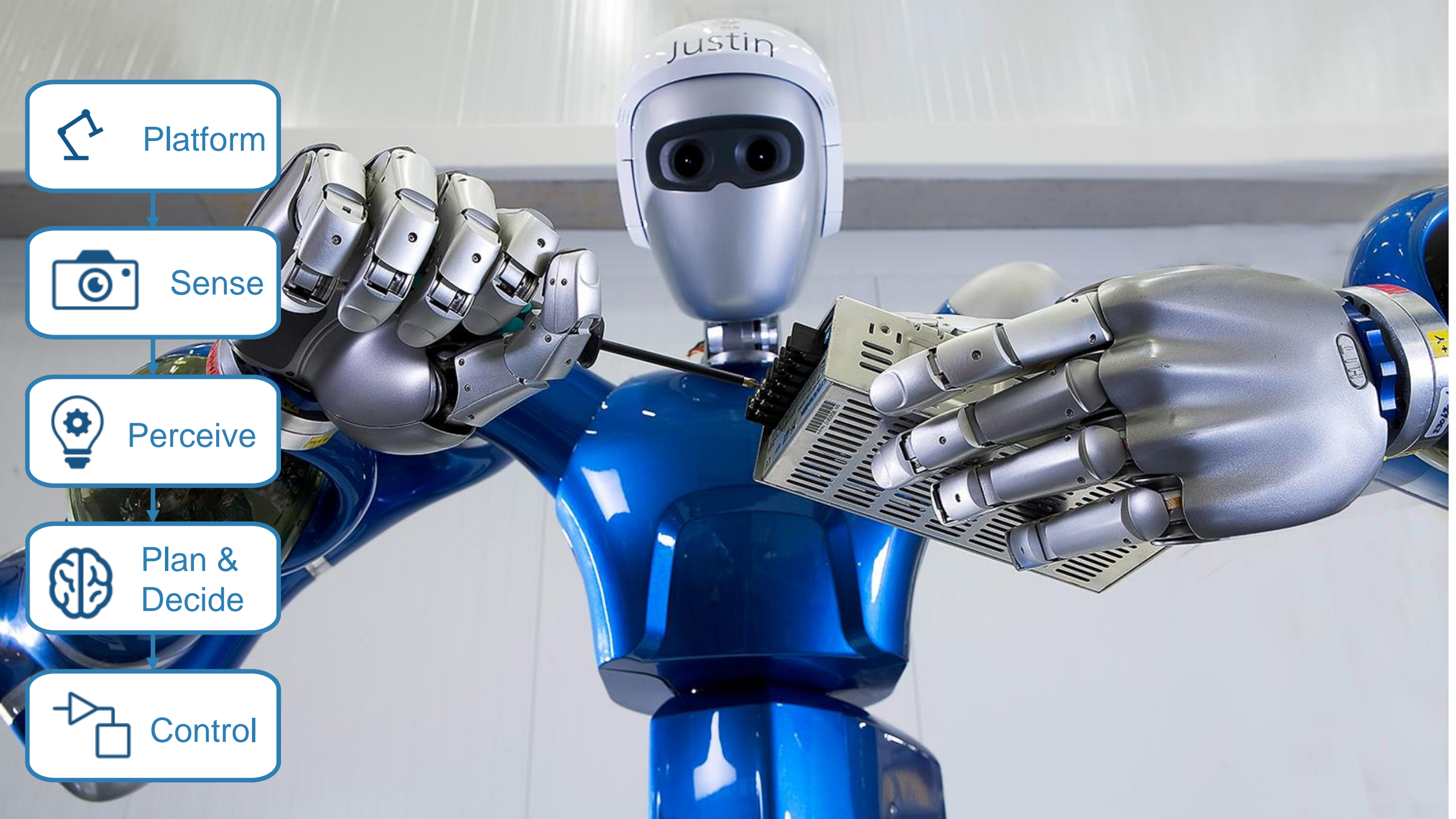
IP Protection

# What does success look like?









Platform



Sense



Perceive



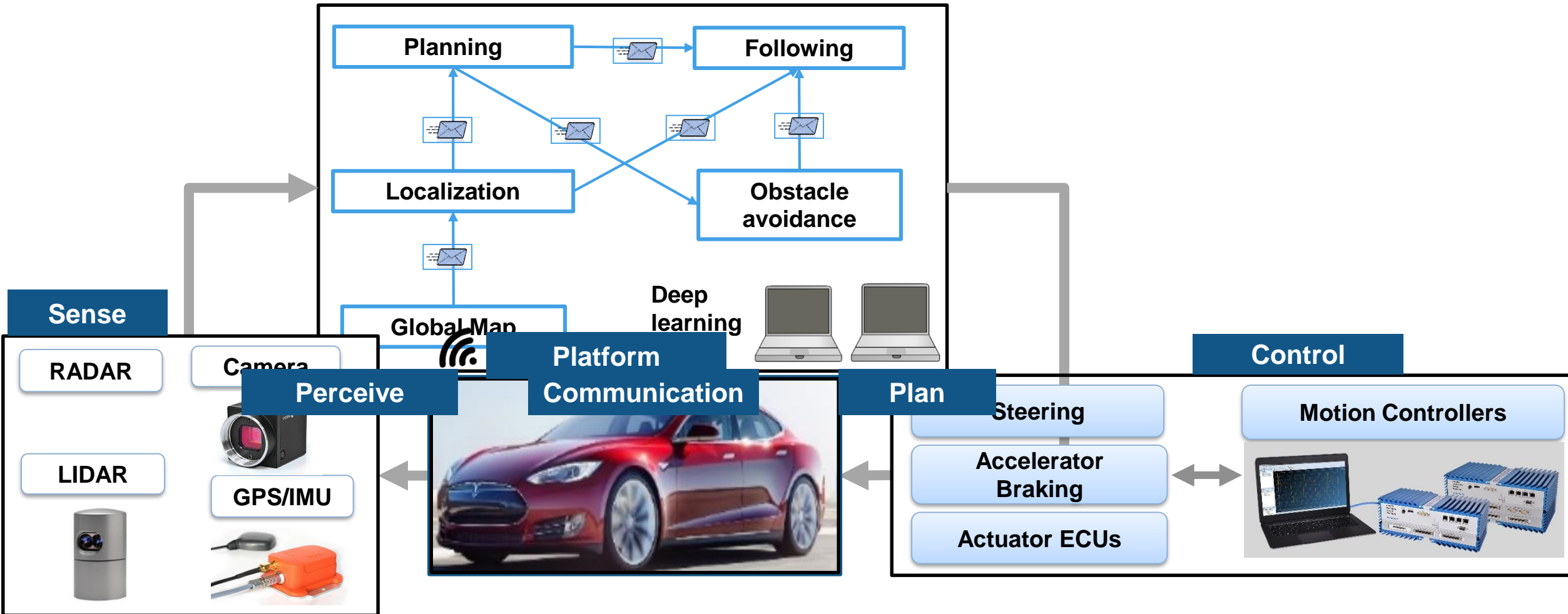
Plan &  
Decide



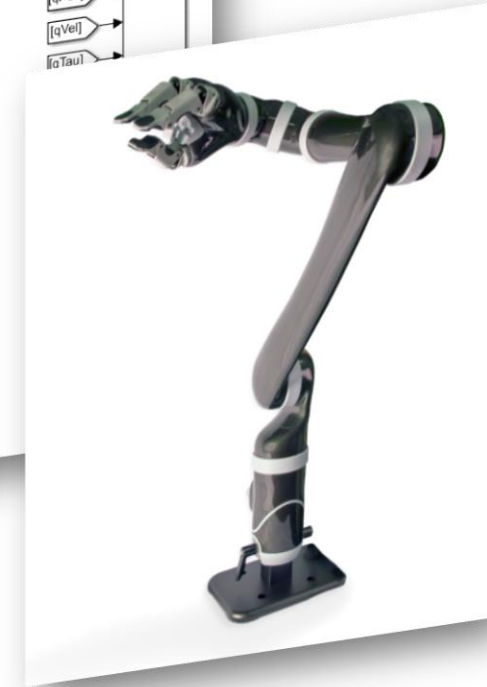
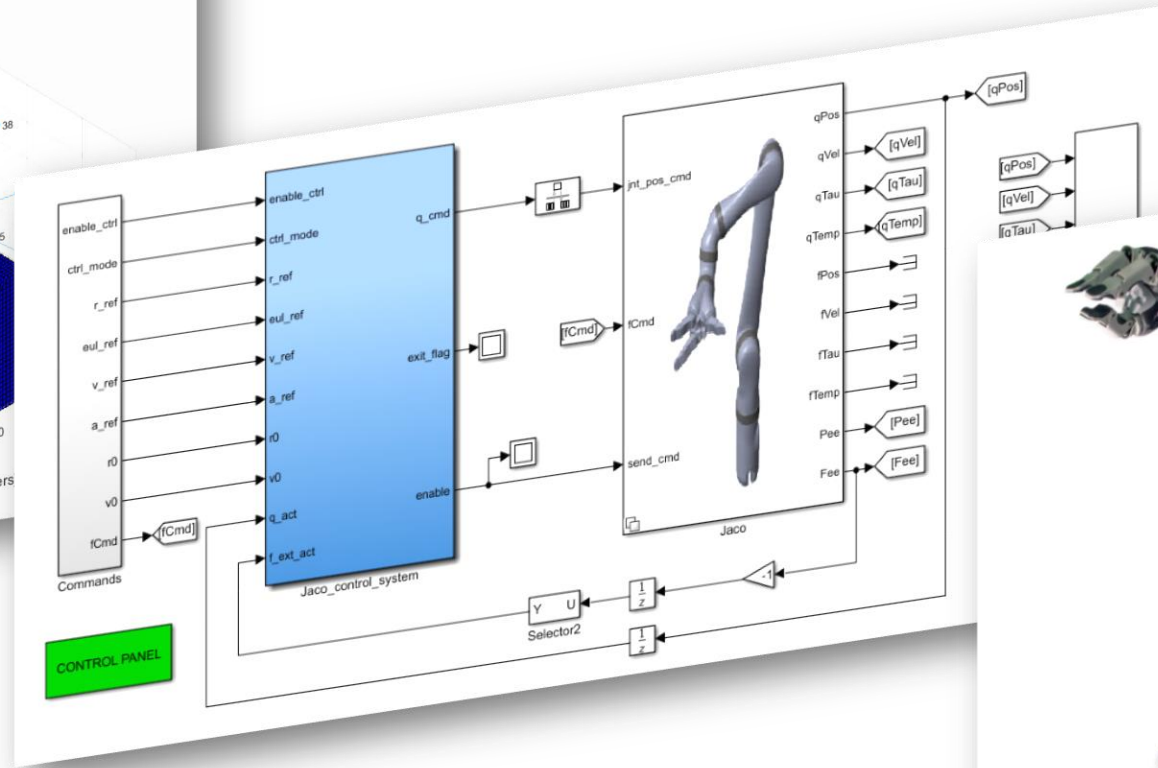
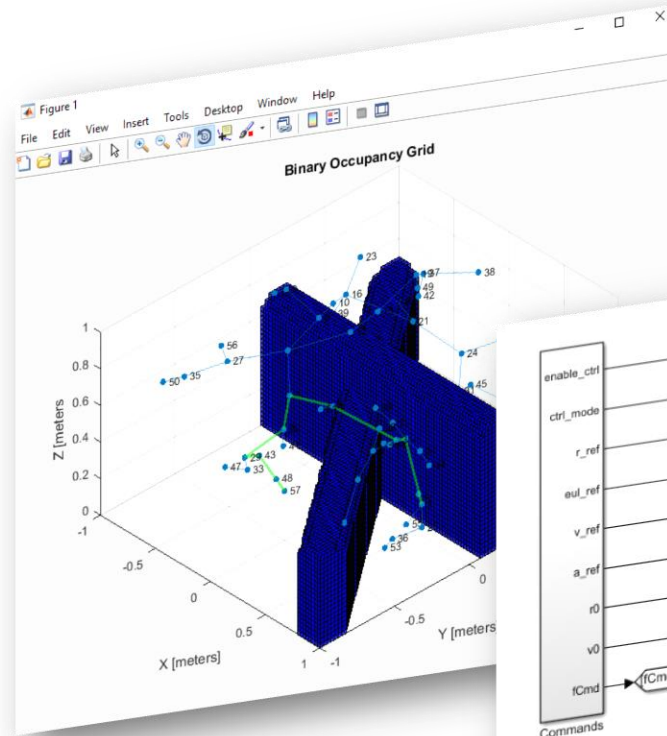
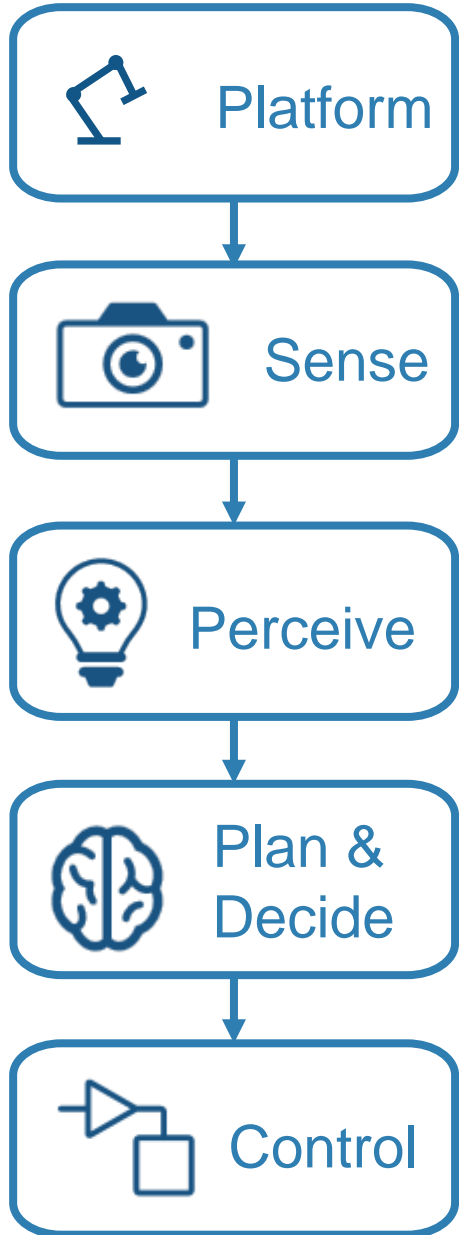
Control



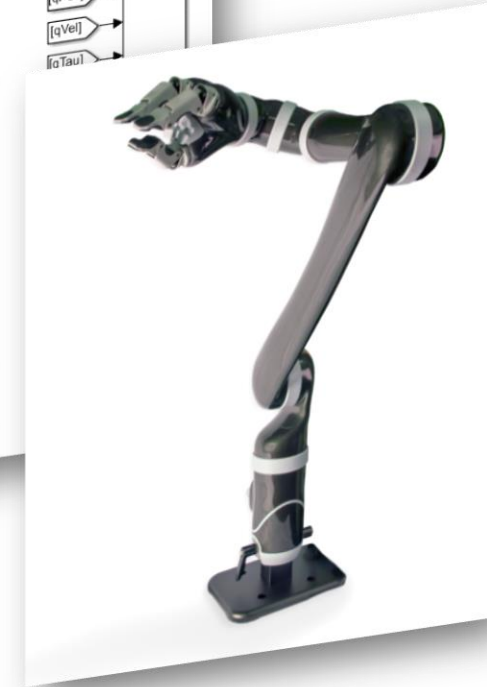
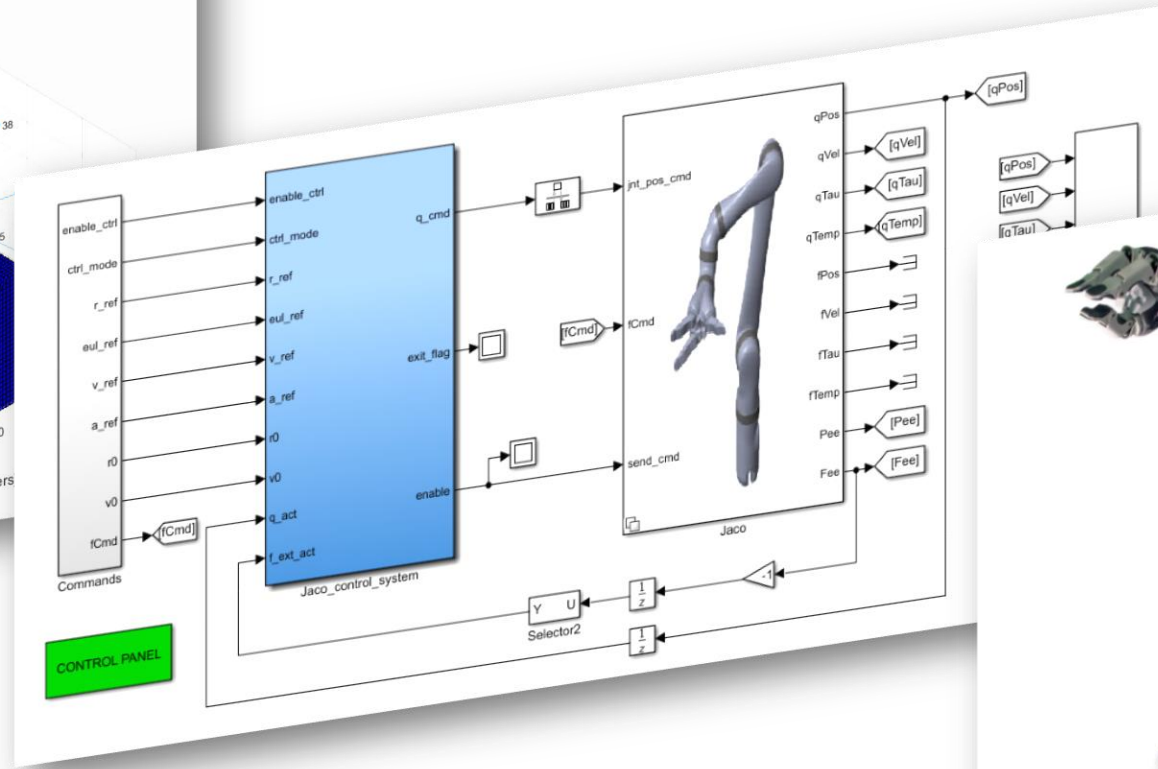
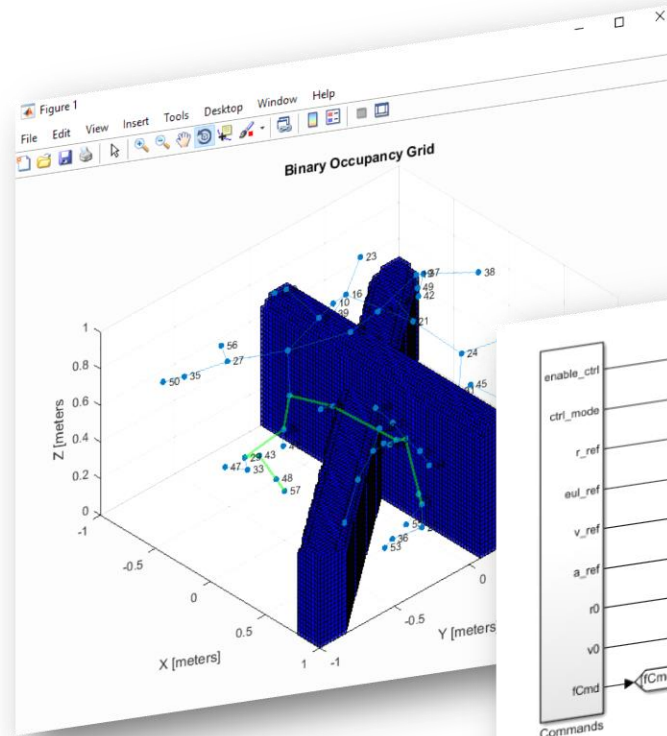
# Another Example: Self-Driving Cars



# Today: Design Pick and Place Application



# Today: Design Pick and Place Application



# Platform Design

*How to create a model of my system that suits my needs?*

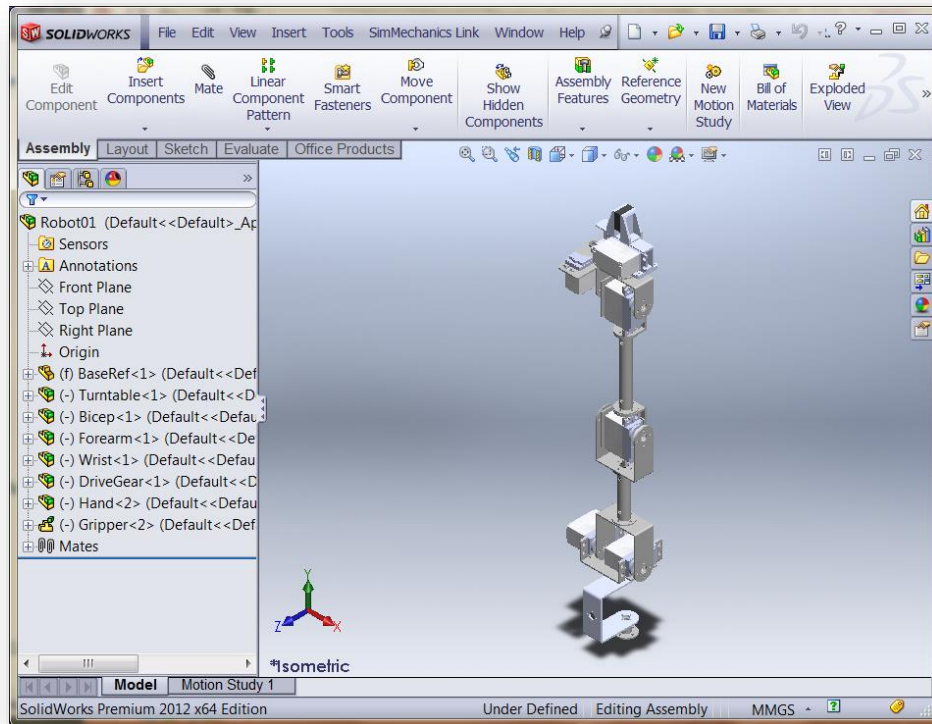
Mechanics

Actuators

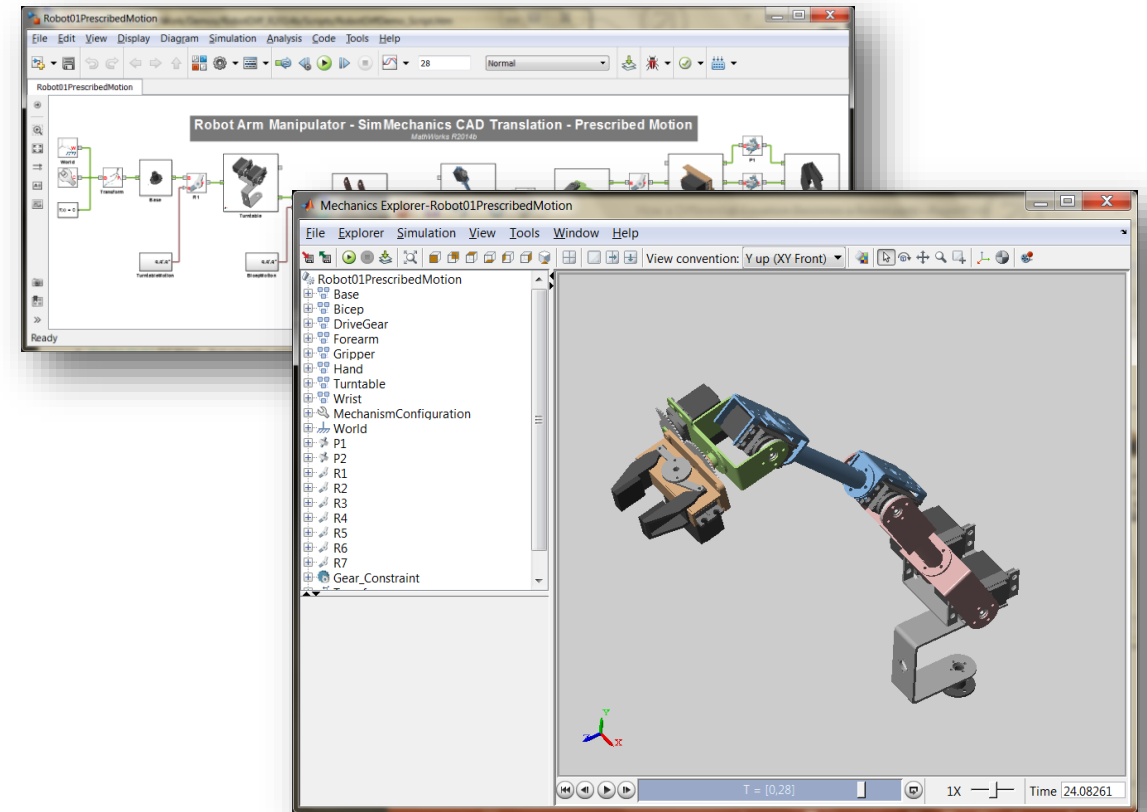
Environment

# Mechanics: Import models from common CAD Tools

## SolidWorks Model

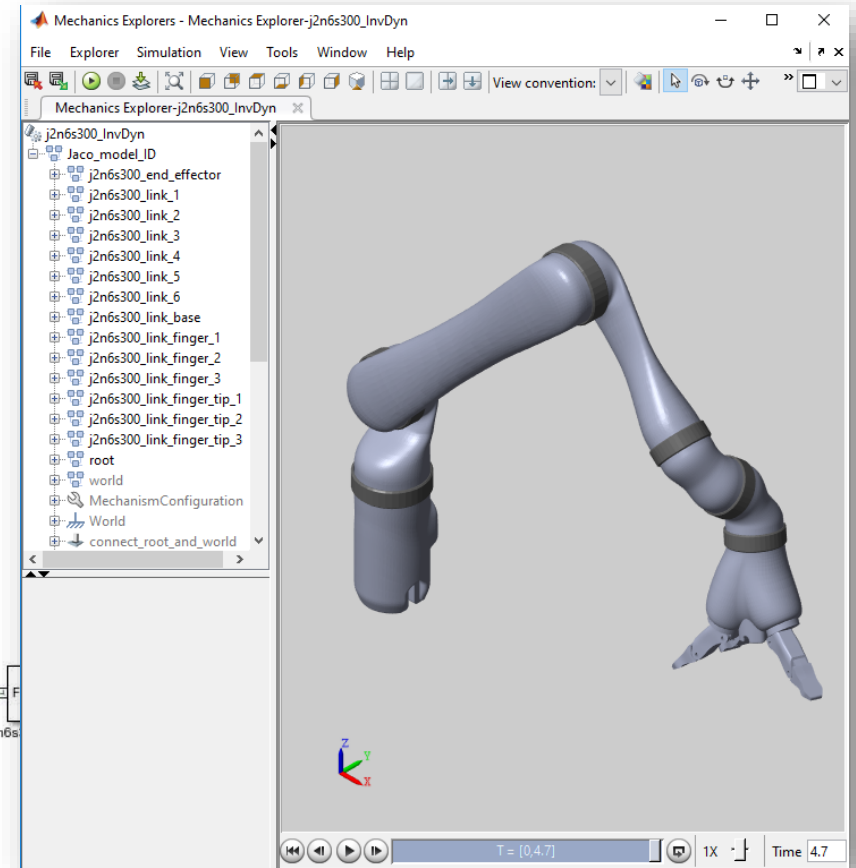
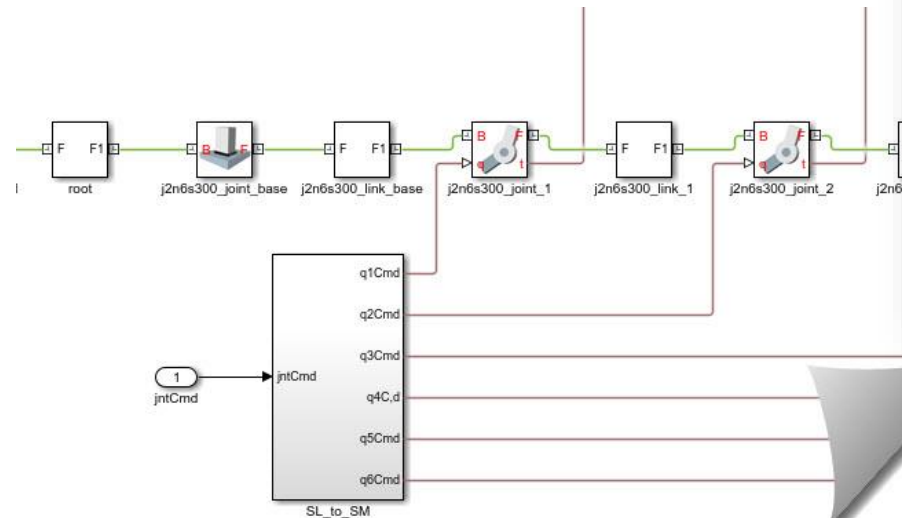
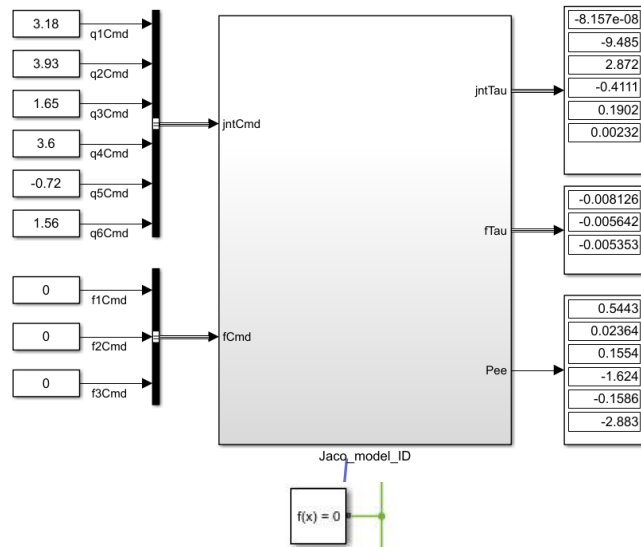


## Simscape Multibody Model



# Mechanics: One line import from URDF

```
%% Import robot from URDF
smimport('j2n6s300_standalone_stl.urdf');
```

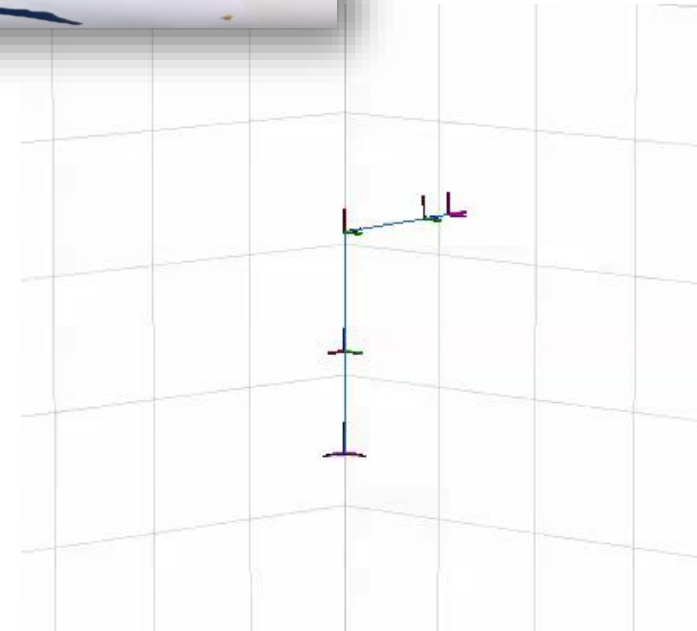
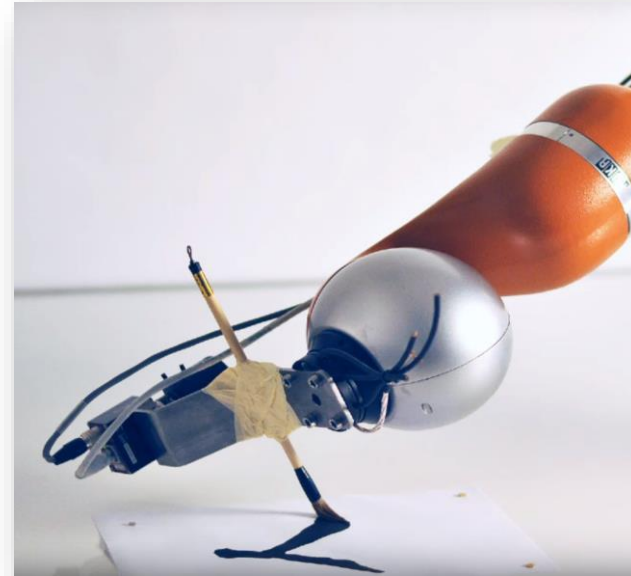


# Rigid Body Tree Dynamics

Compute rigid body tree dynamics quantities

- Specify rigid body inertial properties
- Compute for the rigid body tree
  - Forward dynamics
  - Inverse dynamics
  - Mass matrix
  - Velocity product
  - Gravity torque
  - Center of mass position and Jacobian

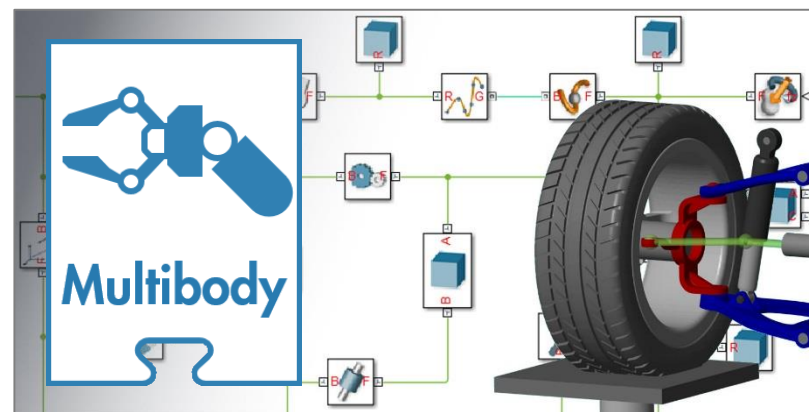
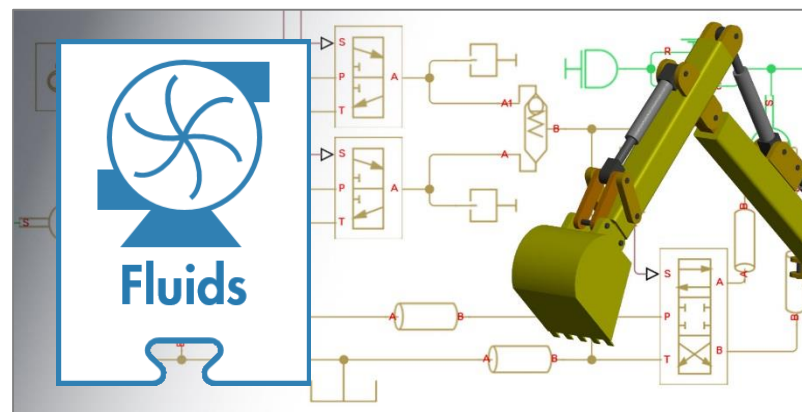
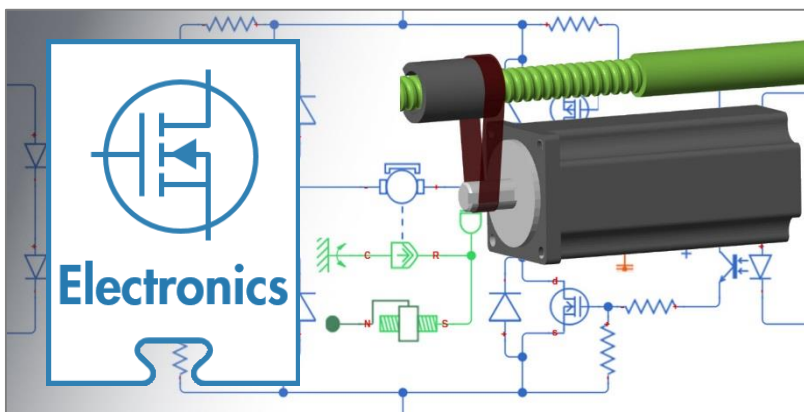
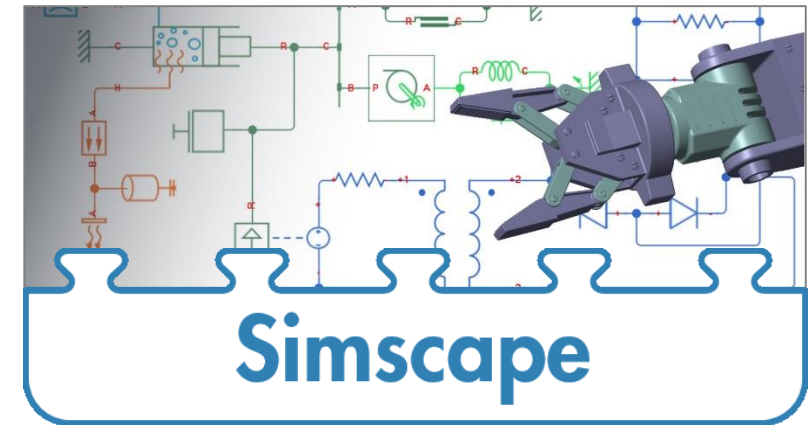
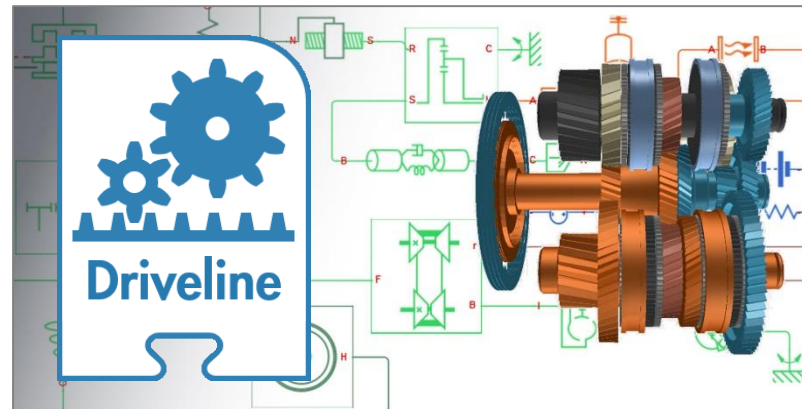
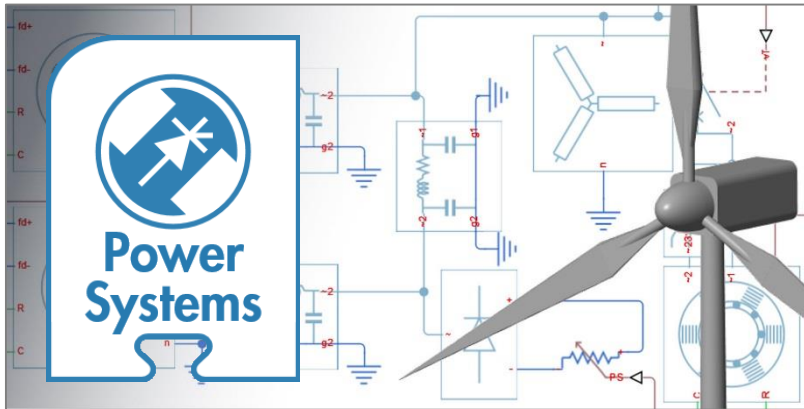
```
» load exampleRobots.mat  
» lbr.DataFormat = 'column';  
» q = lbr.randomConfiguration;  
» tau = inverseDynamics(lbr, q);
```



# Actuators: Connect Motors



# Actuators: Model other domains



# Environment: Connect to an external robotics simulator

# Environment: Connect to an external robotics simulator

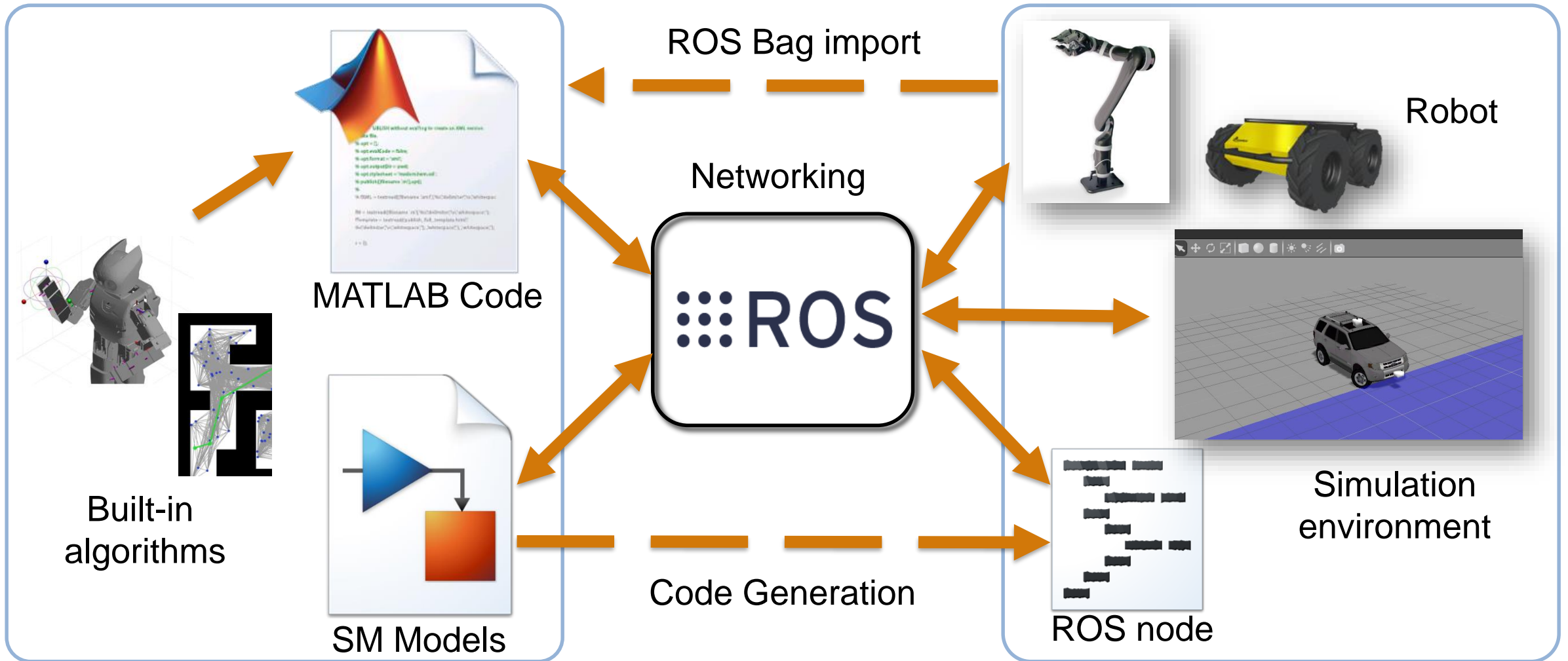
The image displays a Simulink environment connected to a Gazebo 7 simulator. The Simulink window, titled "mainController \* - Simulink", shows a control loop diagram. The diagram includes a "Motion Planner" block that receives a "request" (containing  $\min f(x)$ ) and outputs a "response" (containing "plannerResponse" and "plannerCommand"). The "State Controller" block receives "userCommands" and "goalReached" signals and outputs "armCommand" and "goalSpline". The "Robot ROS interface" block receives "armCommand" and "goalSpline" and outputs "qAct" and "goalReached". A "User Command" block (1) and an "Object To Grab" block (2) provide input to the system. A "Cloud prediction" block is also visible. The status bar shows "50% T=23.000" and "FixedStepDiscrete".

The Gazebo 7 - VMware Player (Non-co...) window shows a 3D simulation of an orange robotic arm on a table. A blue laser line indicates the target position. The status bar shows "Gazebo 7 - VMware Player (Non-co...)" and "Camera View Window Help".

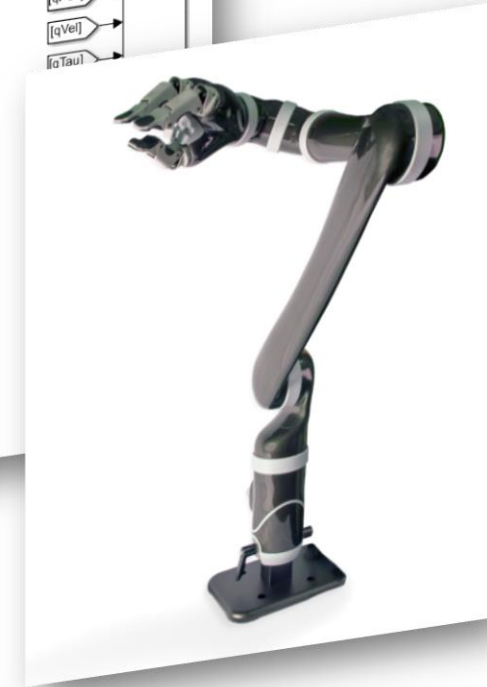
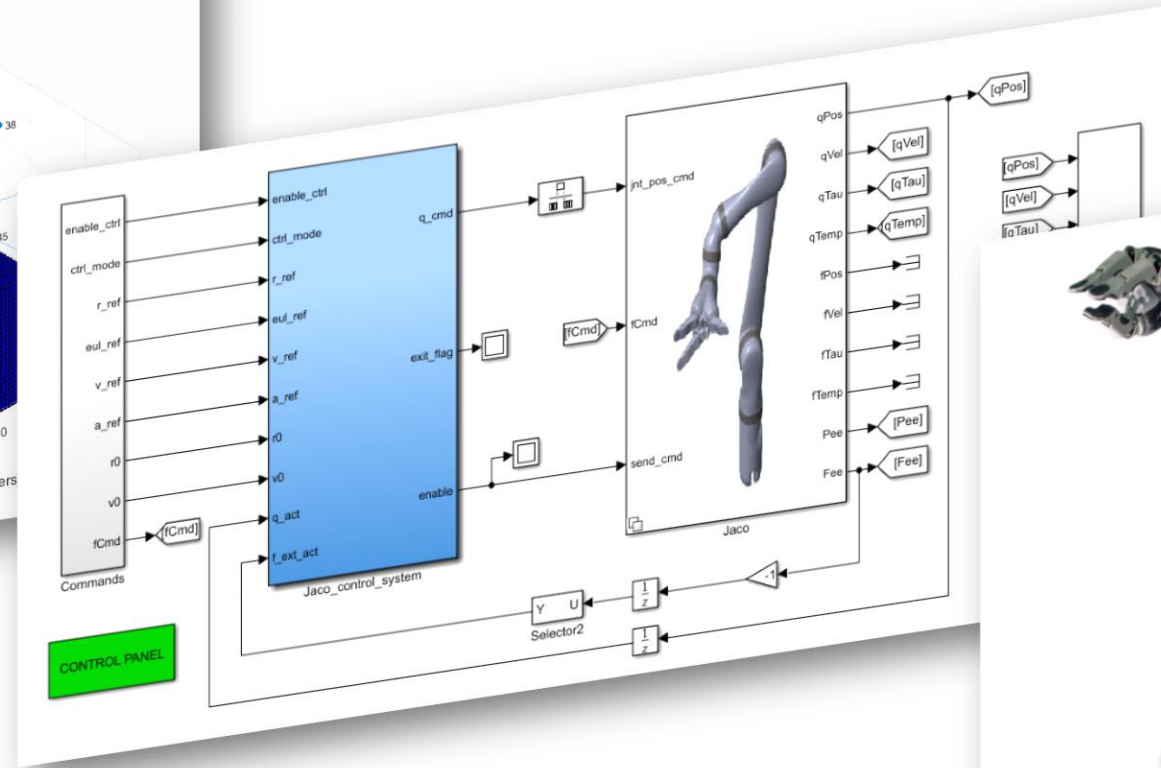
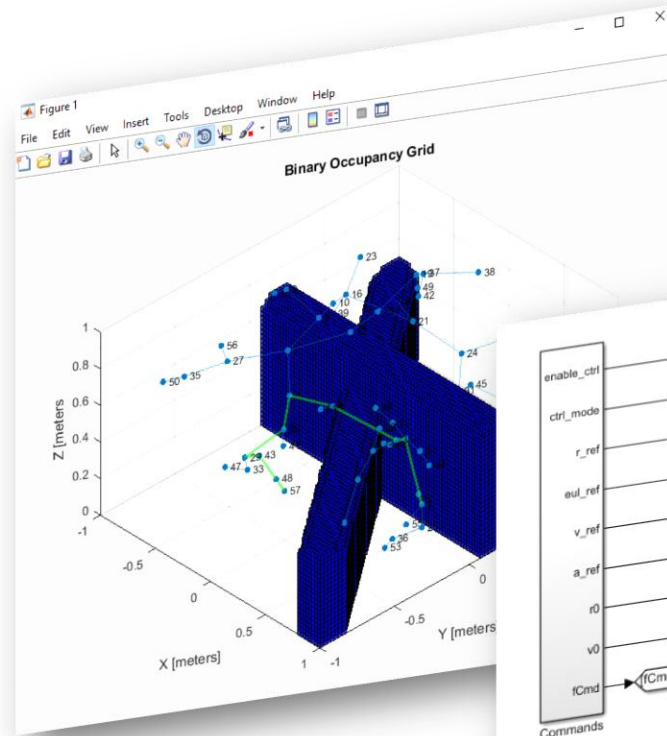
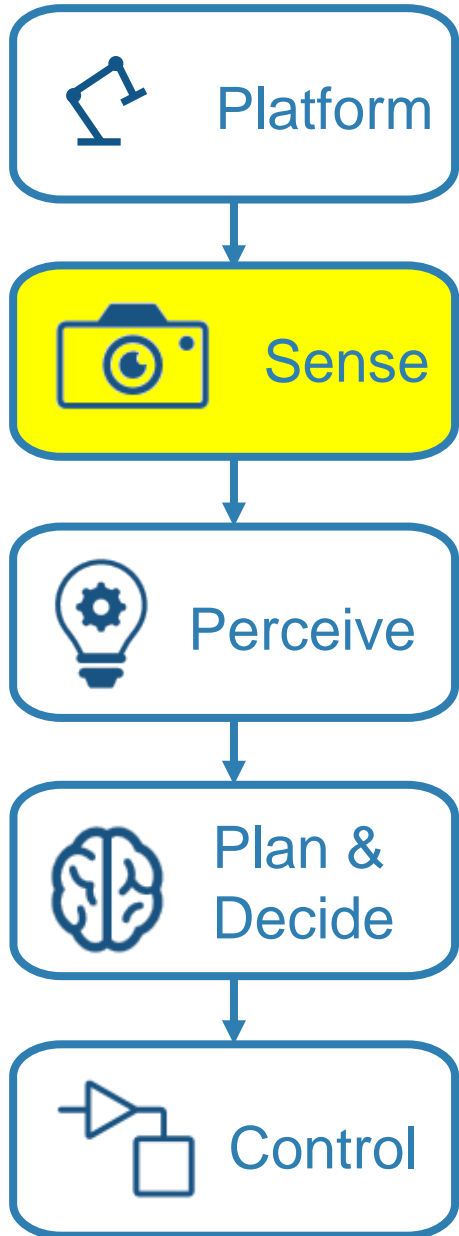
A "To Video Display1" window is overlaid on the Simulink window, showing a table of data:

#	label	X	Y	Z
+1	label : 3	X: 0.29	Y: -0.13	Z: -0.01
+2	label : 2	X: 0.29	Y: -0.02	Z: -0.01
+3	label : 1	X: 0.29	Y: 0.10	Z: -0.01

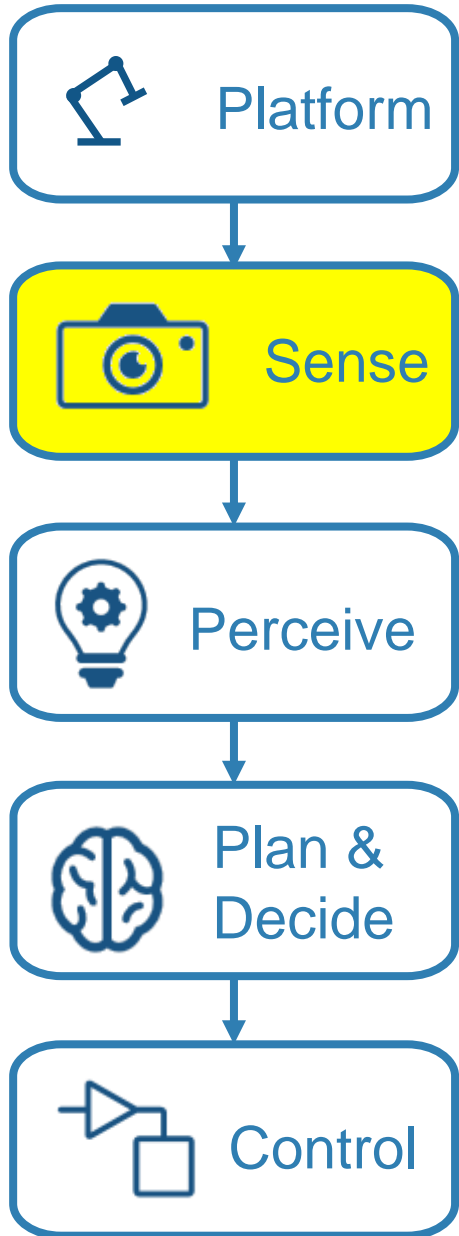
# Environment: Connect MATLAB and Simulink with ROS



# Today: Design Pick and Place Application



# Today: Design Pick and Place Application



Support for Common Sensors  
Cameras, Laser Scanners, Optical Encoders, IMU, GPS.

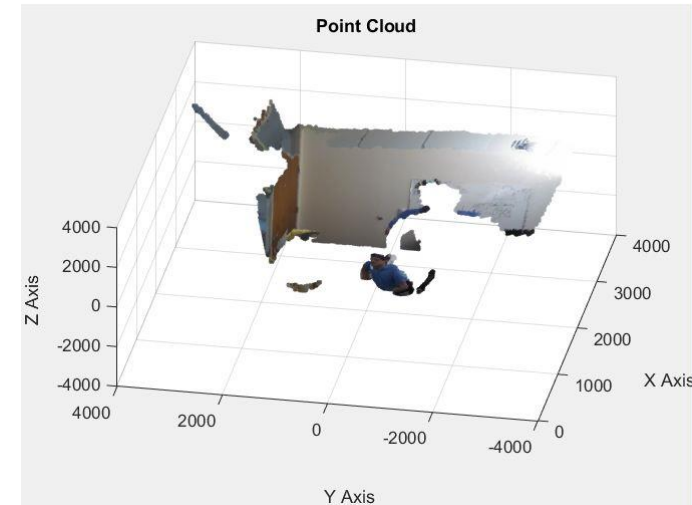
**Image analysis**, including segmentation, morphology, statistics, and measurement

**Apps** for image region analysis, image batch processing, and image registration

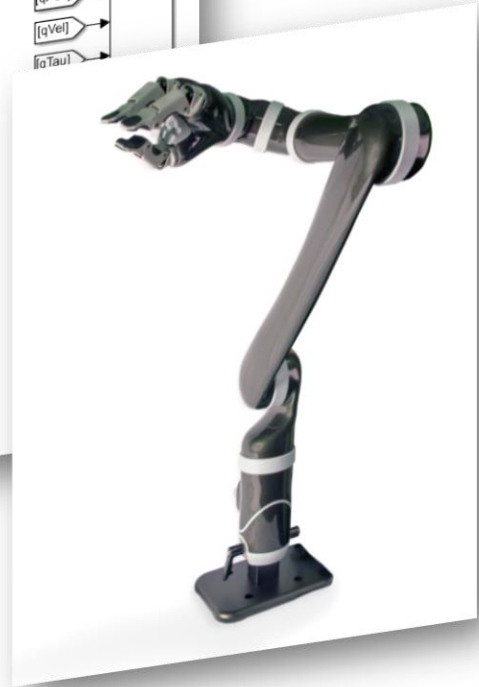
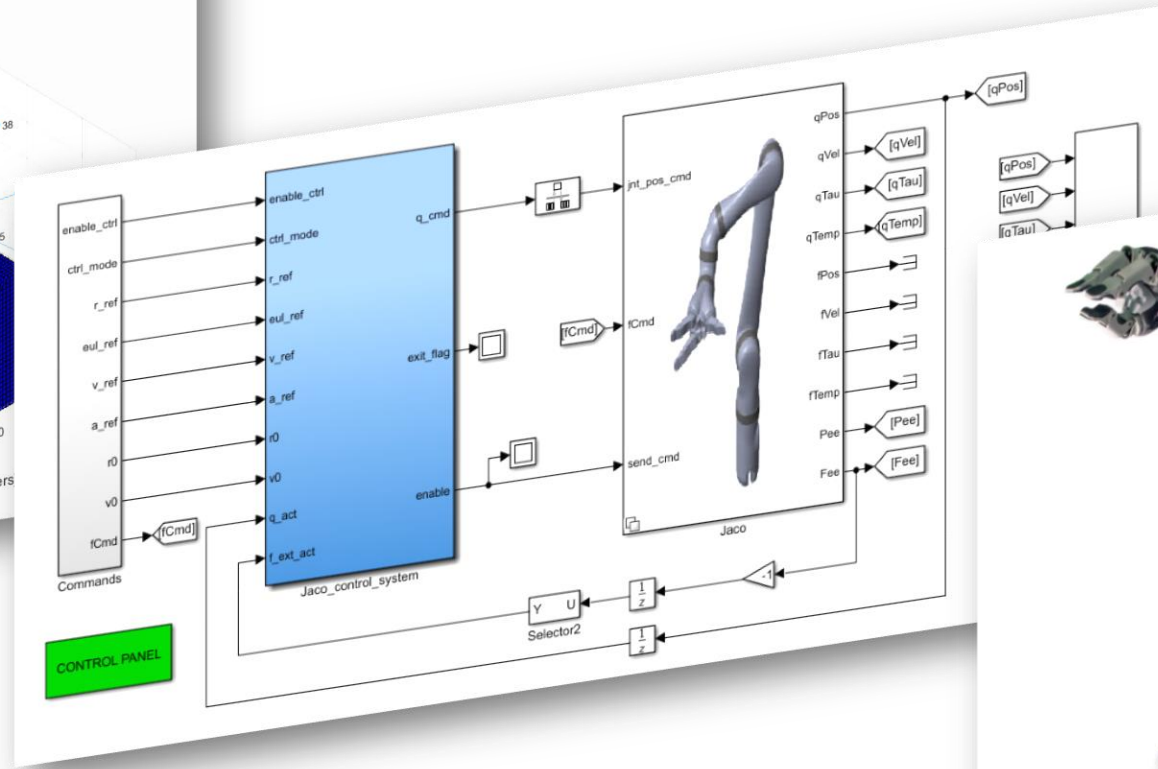
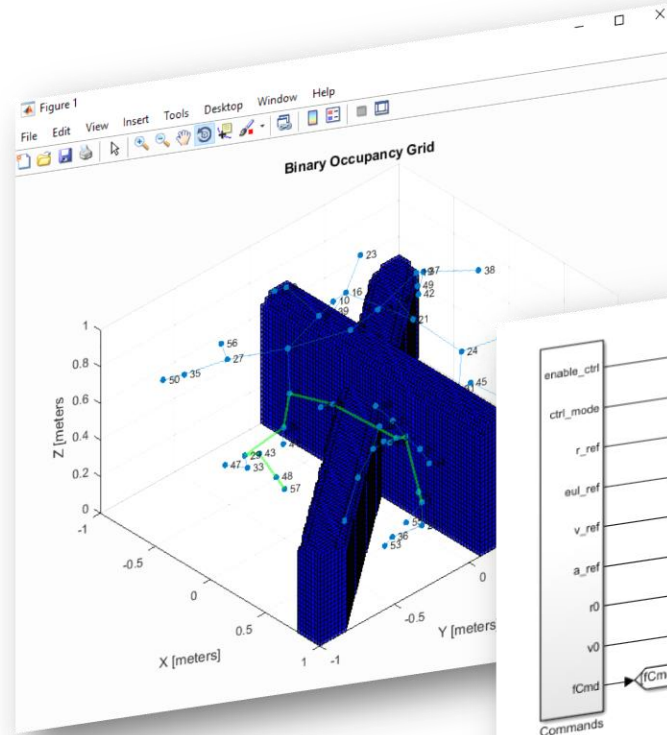
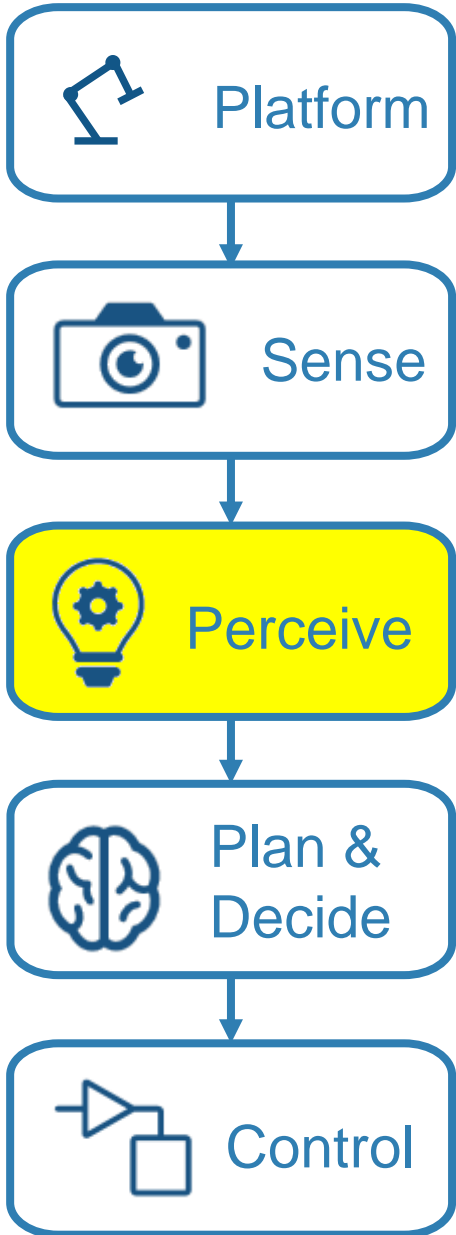
**Image enhancement**, filtering, geometric transformations, and deblurring algorithms  
Intensity-based and non-rigid image registration.

## Visualizing Point Clouds

To visualize a point cloud in MATLAB, use the `showPointCloud` and `scatter3` command.



# Today: Design Pick and Place Application



# Sign Detector and Classifier

Images



Sign Classifier



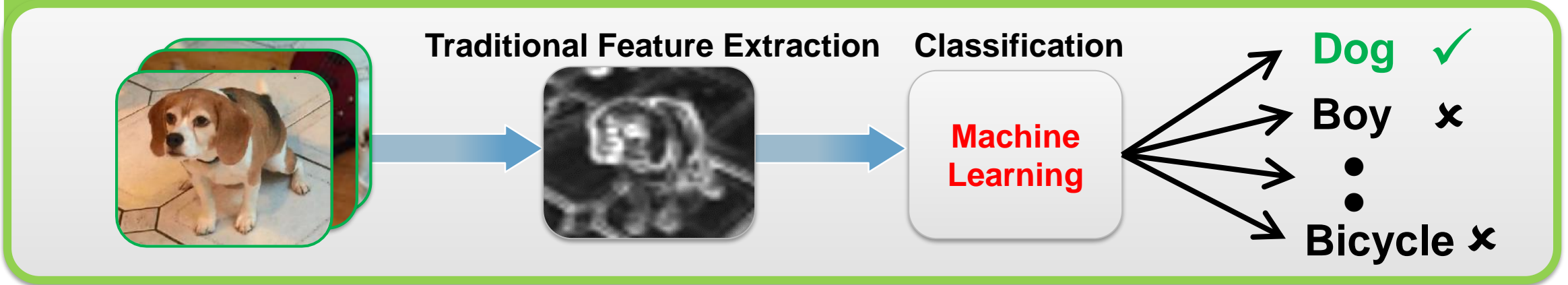
Labels



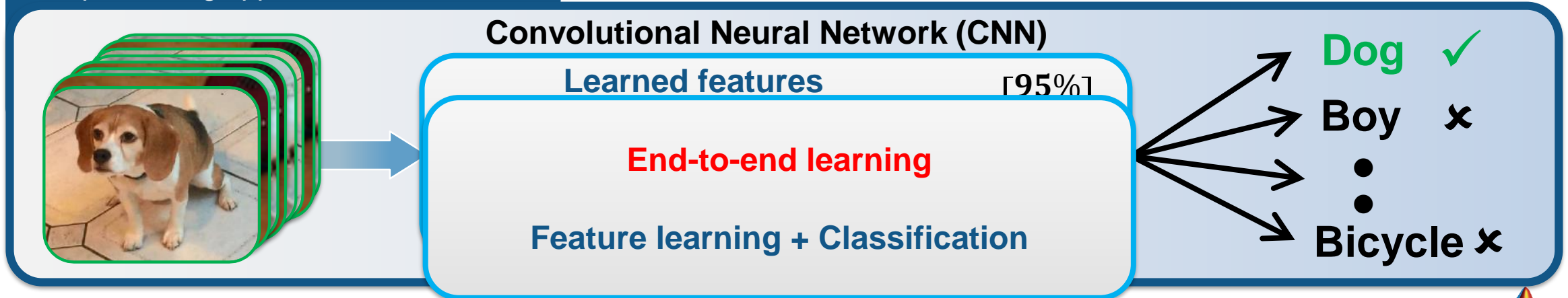


# MATLAB makes machine learning easy and accessible

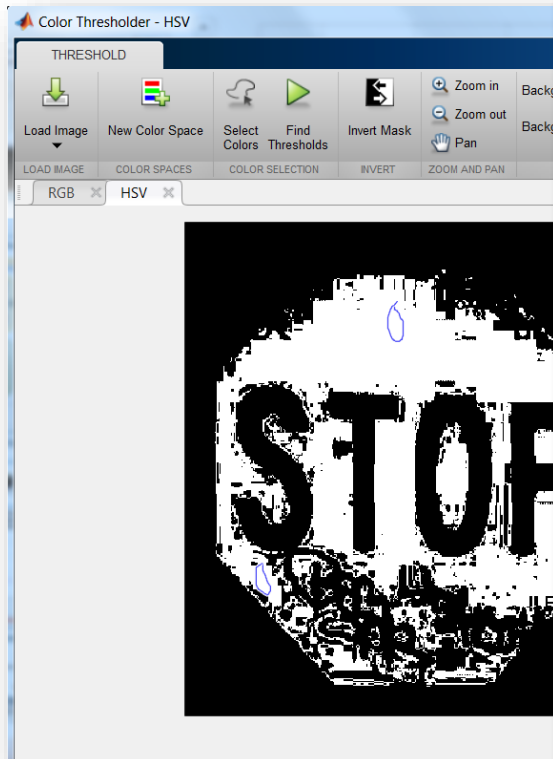
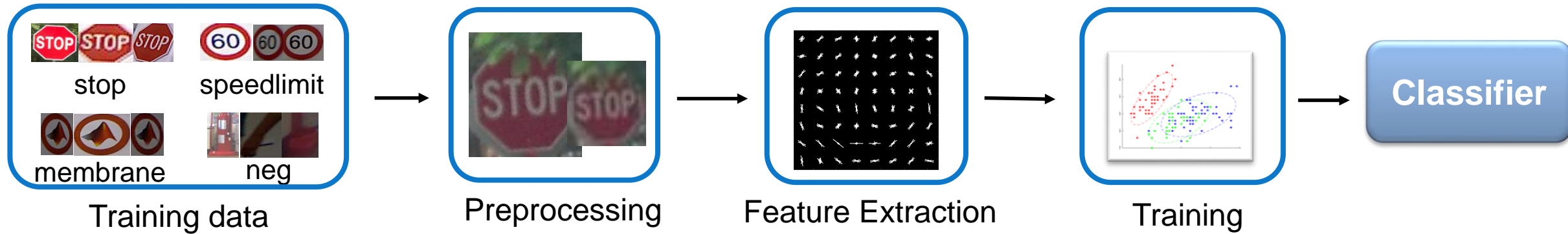
## Traditional Machine Learning approach



## Deep Learning approach



# Complex workflows made easy with MATLAB



```

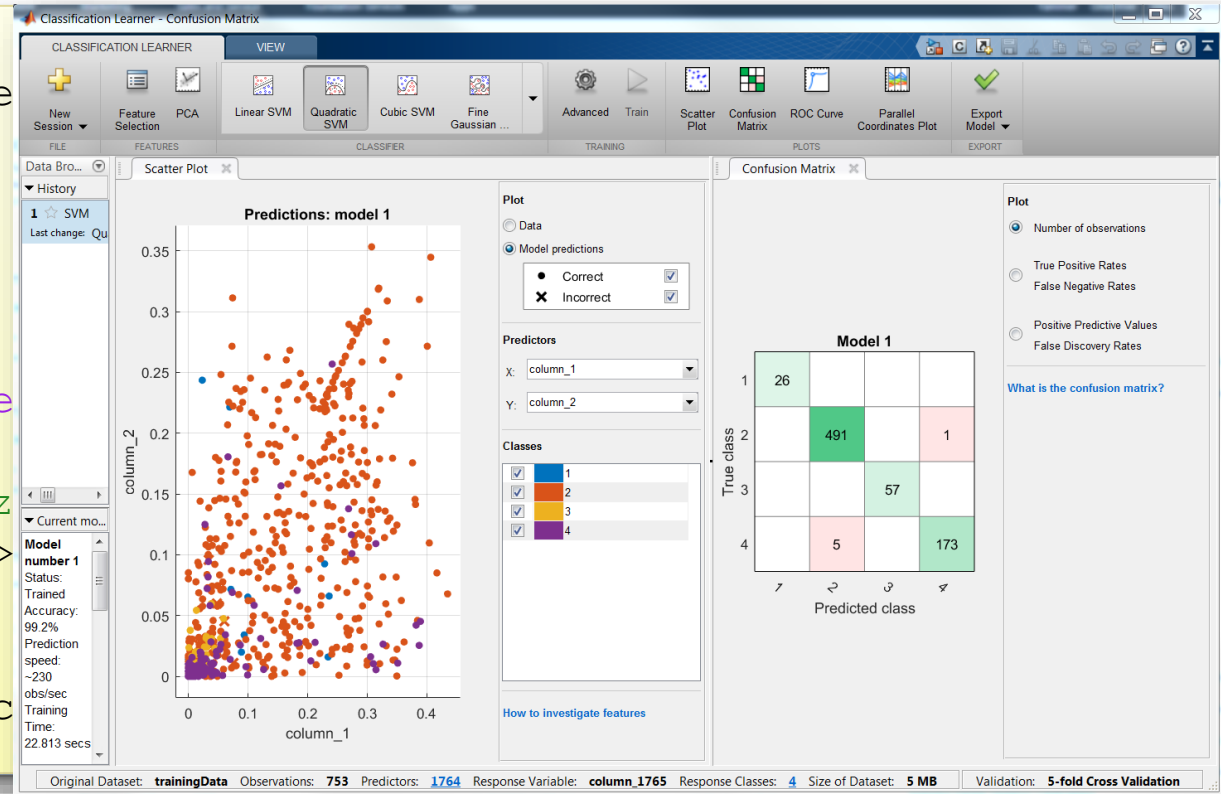
% Detect red regions
BW = createMask(videoFrame

% Fill image regions
BW = imfill(BW, 'holes');

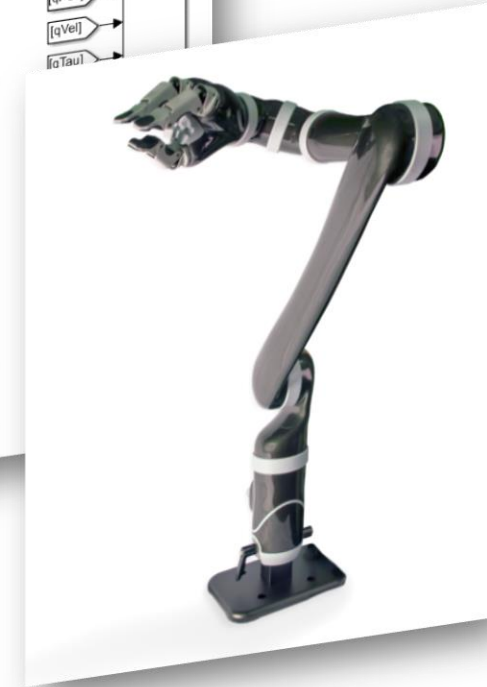
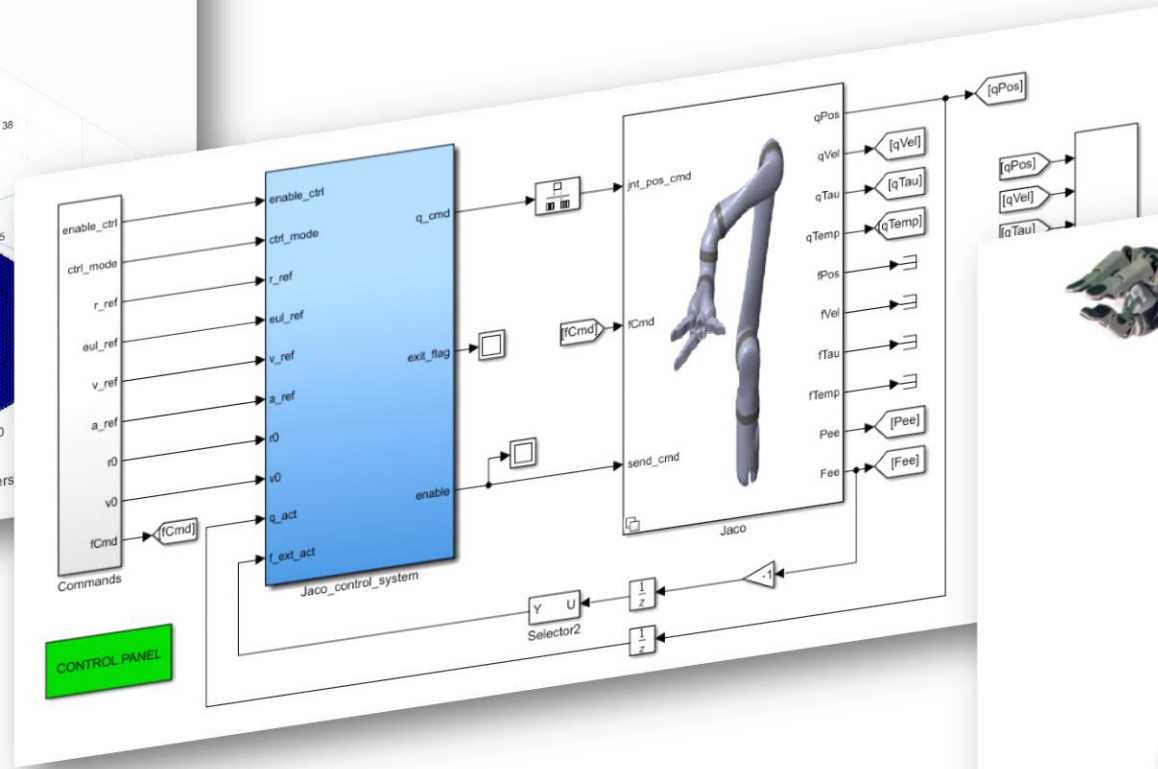
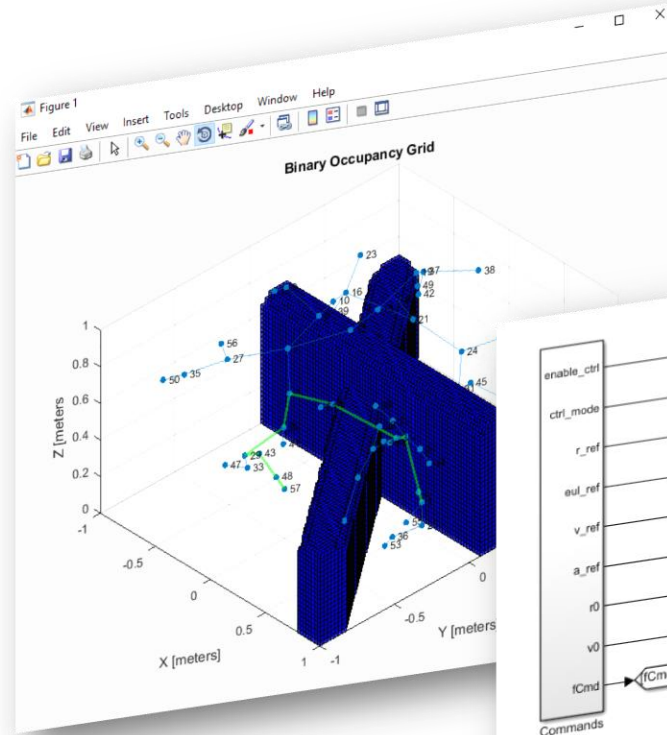
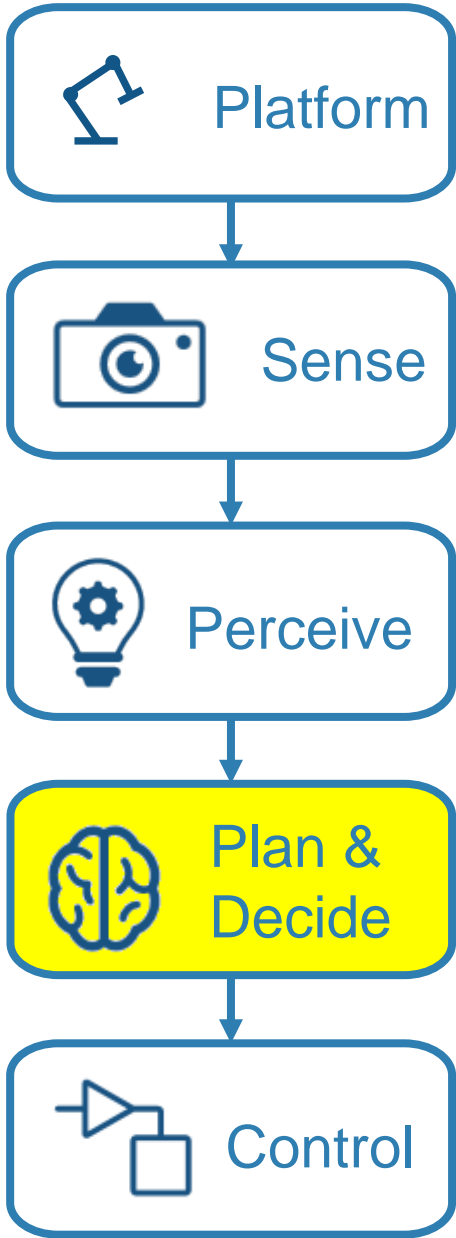
% Get bounding boxes
stats = regionprops('table

% Filter based on area siz
targetIndex = stats.Area >

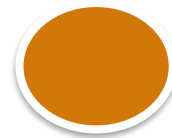
% Get bounding boxes from
testFeatures(k,:) = extrac
    
```



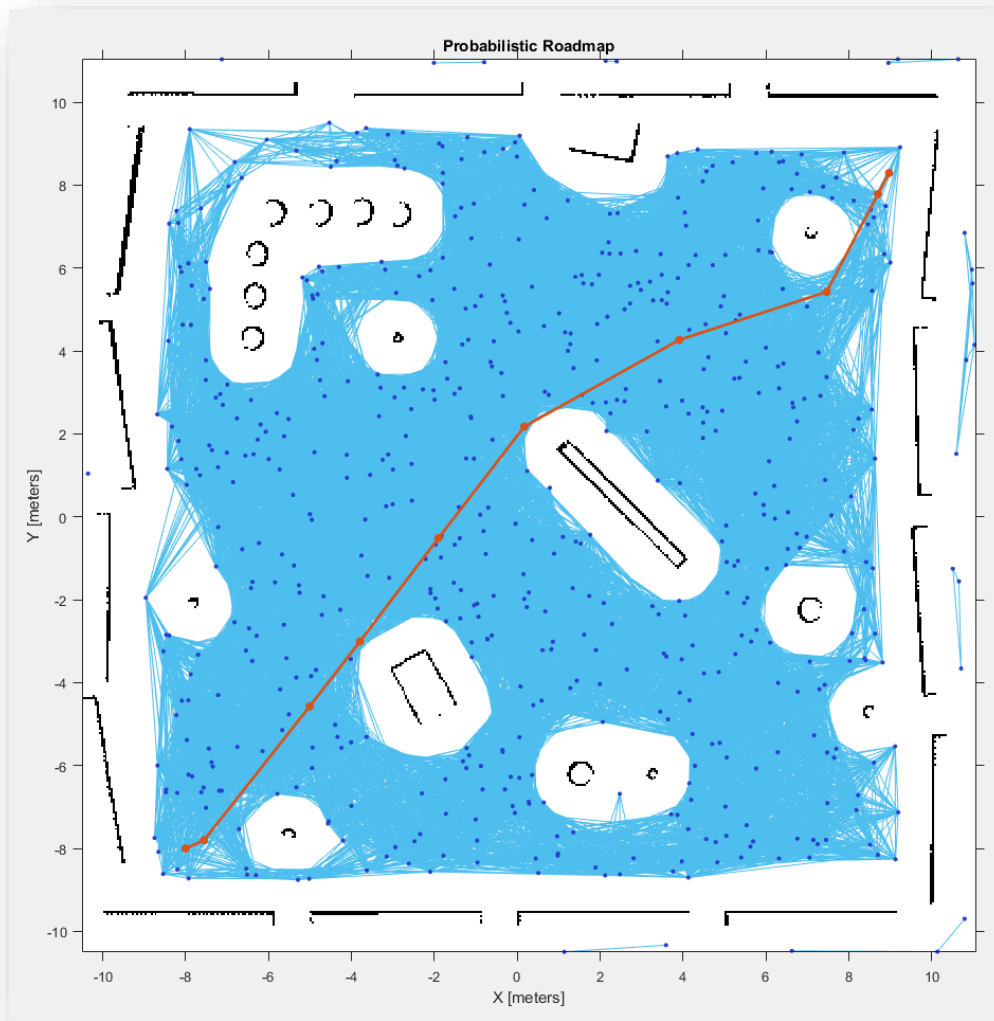
# Today: Design Pick and Place Application



# Planning

 $[x_a \ y_a \ \theta_a]$  $[x_b \ y_b \ \theta_b]$ 

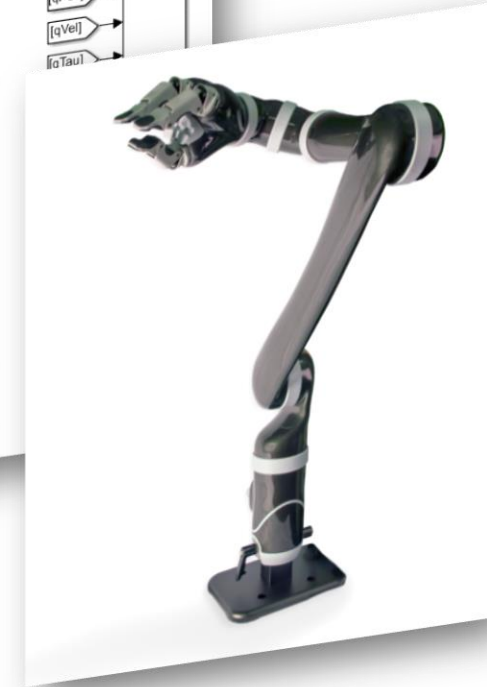
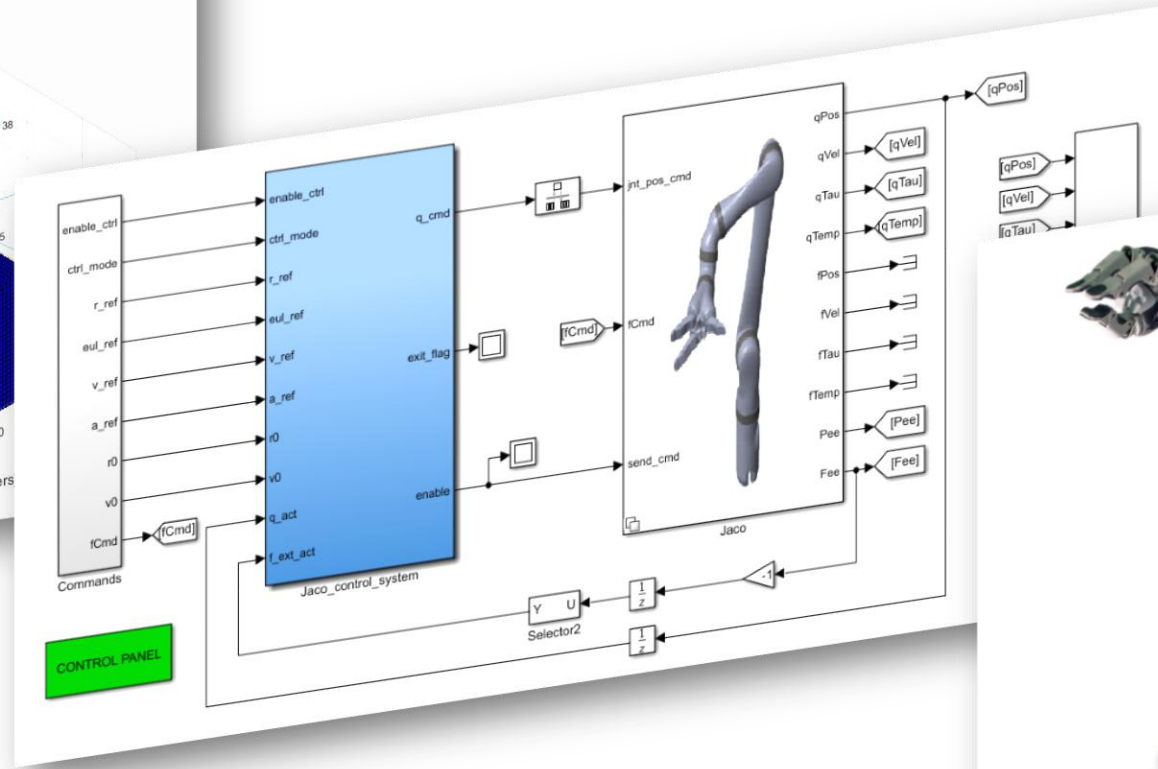
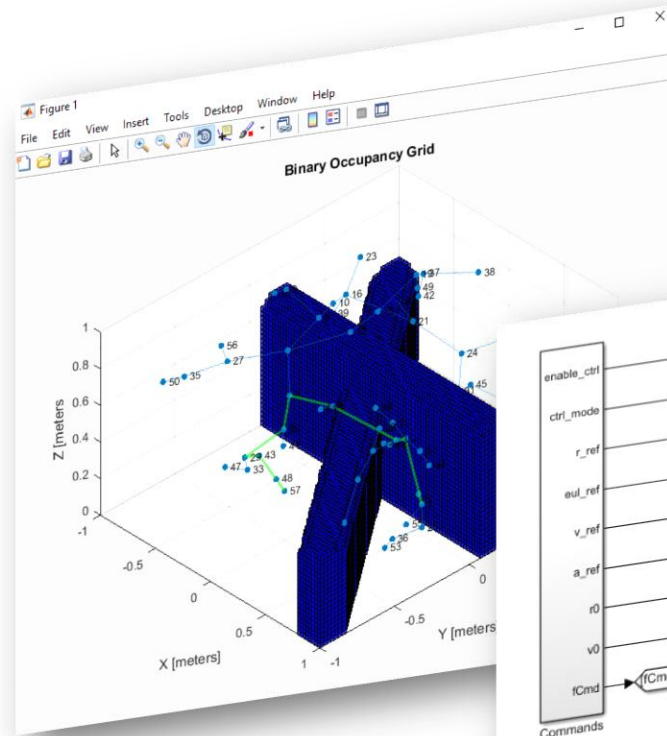
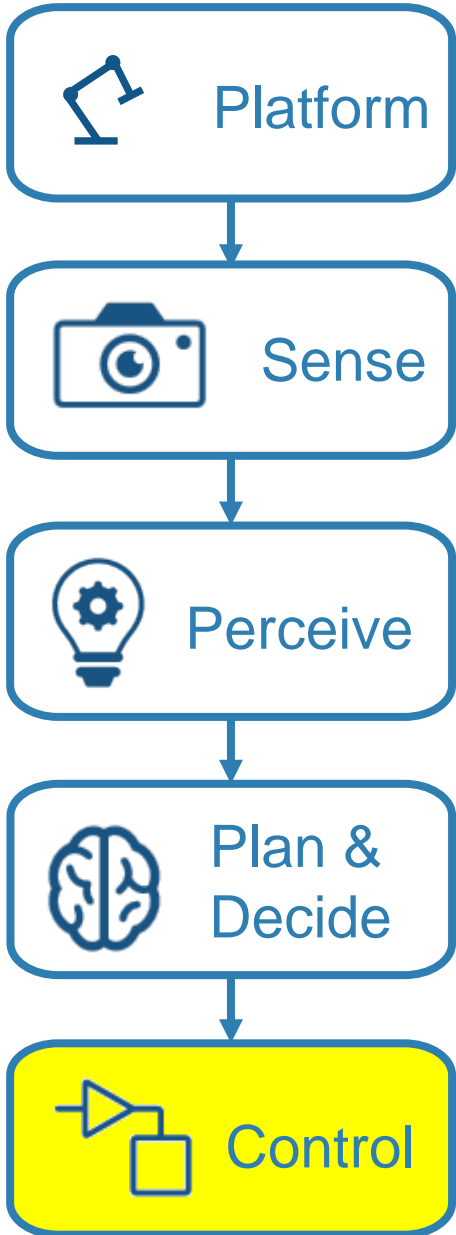
# Explore Built In Functions: PRM Planner



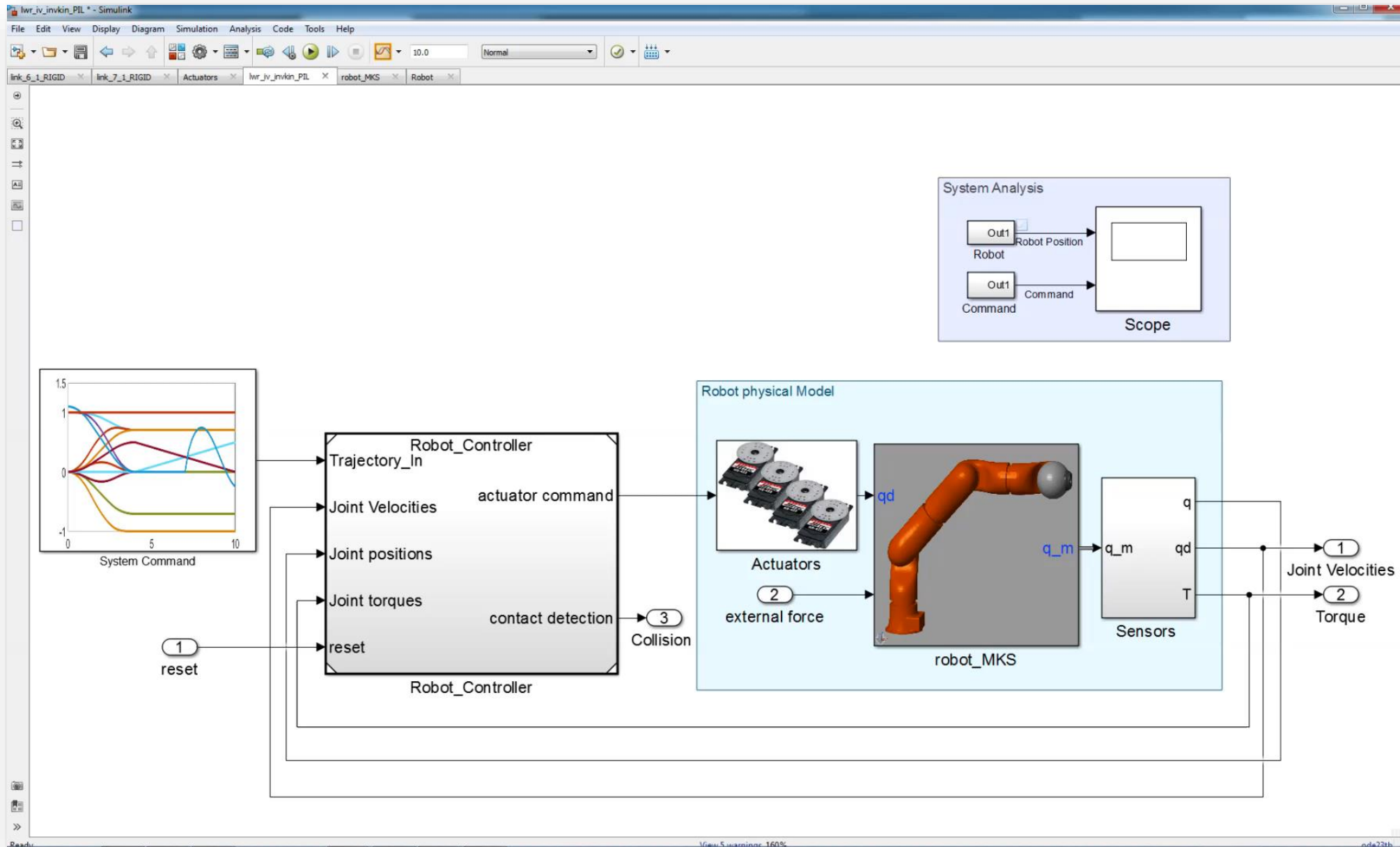
```
%% Create map as Binary Occupancy Grid  
robotics.BinaryOccupancyGrid(binaryImage,20);  
og.GridLocationInWorld = [-10.5, -10.5];  
figure;  
show(og);
```

```
%% Create PRM path planner  
planner = robotics.PRM;  
planner.Map = og;  
planner.NumNodes = 600;  
planner.ConnectionDistance = 5;  
show(originalOg); hold on;  
show(planner, 'Map', 'off');
```

# Today: Design Pick and Place Application



# Control of Manipulator Arms



# Model-Based Design for Motion Control

- Requirements traceability
- Early verification of requirements
- Automatic code generation.
- Automatic report generation
- Test automation

**Code Generation Report**

File: [Stanley\\_controller.c](#)

```

1  /*
2  * File: Stanley_controller.c
3  *
4  * Code generated for Simulink model 'Stanley_controller'.
5  *
6  * Model version           : 1.164
7  * Simulink Coder version  : 8.12 (R2017a) 16-Feb-2017
8  * C/C++ source code generated on : Thu Mar 16 16:58:00 2017
9  *
10 * Target selection: ert.tlc
11 * Embedded hardware selection: Intel->x86-64 (Windows64)
12 * Code generation objectives: Unspecified
13 * Validation result: Not run
14 */
15
16 #include "Stanley_controller.h"
17 #include "Stanley_controller_private.h"
18
19 /* Block states (auto storage) */
20 DW_Stanley_controller_T Stanley_controller_DW;
21
22 /* External inputs (root inport signals with auto storage) */
23 ExtU_Stanley_controller_T Stanley_controller_U;
24
25 /* External input sizes (root inport signals with variable sizes) */
26 ExtUSize_Stanley_controller_T Stanley_controller_USize;
27
28 /* External outputs (root outports fed by signals with auto storage) */
29 ExtY_Stanley_controller_T Stanley_controller_Y;
30
31 /* Real-time model */
32 RT_MODEL_Stanley_controller_T Stanley_controller_M;
33 RT_MODEL_Stanley_controller_T *const Stanley_controller_M =
34     &Stanley_controller_M;
35
36 /* Forward declaration for local functions */
37 static real_T Stanley_controller_angdiff(real_T x, real_T y);
38 static real_T Stanley_controller_norm(const real_T x[2]);

```

**Stanley\_controller**

Sample Time specified here

Waypoints (1) → Waypoints

Desired\_Velocity (2) → Desired\_Velocity

Pose\_Front\_Wheels (5) → Pose\_Front\_Wheels

Pose\_Rear\_Wheels (4) → Pose\_Rear\_Wheels

State (3) → State

Waypoints → Steering\_Angle

Desired\_Velocity → Steering\_Angle

Pose\_Front\_Wheels → Longitudinal\_Vel

Pose\_Rear\_Wheels → Longitudinal\_Vel

State → Longitudinal\_Vel

Steering\_Angle → Angle\_ref

Longitudinal\_Vel → Long\_Vel

Angle\_ref → Steering\_Wheel\_PID

Long\_Vel → Vel\_ref

Steering\_Wheel\_PID → Steering\_Angle\_vel (1)

Vel\_ref → Longitudinal\_Vel\_PID

Longitudinal\_Vel\_PID → Long\_Accel (2)

**Stanley\_controller** Parameters:

- ModelVersion: 1.164
- LastModified: Thu Mar 16 16:57:49 2017
- LibraryLibrary: disabled
- ModelBrowser: off
- Dirty: off
- Description: off



## Key Takeaway of this Talk

Success in developing an autonomous robotics system requires:

- Multi-domain simulation
- Great tools which make complex workflows easy and integrate with other tools
- Model-based design

## German Aerospace Center (DLR) Robotics and Mechatronics Center Develops Autonomous Humanoid Robot with Model-Based Design

### Challenge

Develop control systems for a two-armed mobile humanoid robot with 53 degrees of freedom

### Solution

Use Model-Based Design with MATLAB and Simulink to model the controllers and plant, generate code for HIL testing and real-time operation, optimize trajectories, and automate sensor calibration

### Results

- Programming defects eliminated
- Complex functionality implemented in hours
- Advanced control development by students enabled

[Link to user story](#)



DLR's humanoid robot Agile Justin autonomously performing a complex construction task.

**“Model-Based Design and automatic code generation enable us to cope with the complexity of Agile Justin’s 53 degrees of freedom. Without Model-Based Design it would have been impossible to build the controllers for such a complex robotic system with hard real-time performance.”**

Berthold Bäuml  
DLR

## Festo Develops Innovative Robotic Arm Using Model-Based Design

### Challenge

Design and implement a control system for a pneumatic robotic arm

### Solution

Use Simulink and Simulink PLC Coder to model, simulate, optimize, and implement the controller on a programmable logic controller

### Results

- Complex PLC implementation automated
- Technology and innovation award won
- New business opportunities opened



The Festo Bionic Handling Assistant. Image © Festo AG.

**“Using Simulink for Model-Based Design enables us to develop the sophisticated pneumatic controls required for the Bionic Handling Assistant and other mechatronic designs. With Simulink PLC Coder, it is now much easier to get from a design to a product.”**

**Dr. Rüdiger Neumann**  
Festo

[Link to user story](#)

# Scania Develops Advanced Emergency Braking Systems with Model-Based Design

## Challenge

Develop an advanced emergency braking system to reduce rear-end collisions

## Solution

Use Model-Based Design to develop sensor fusion algorithms, simulate and verify designs, and generate code for implementation on a production ECU

## Results

- 1.5 million kilometers of recorded sensor data simulated in 12 hours
- Design changes quickly implemented
- Optimized production C code generated



A controlled road test of Scania's advanced emergency braking system software.

**“To deploy the sensor fusion system to the ECU, we generated C code from our Simulink model with Embedded Coder. With code generation, we were able to get to an implementation quickly, as well as avoid coding errors.”**

**Jonny Andersson**  
Scania

# Real-time control prototyping of driver assistance and autonomous driving technologies at Mobileye

Patric Schenk, VP of Sales and Engineering, Speedgoat

12-May-2015



© 2015 The MathWorks, Inc. and Speedgoat GmbH

Mobileye chips are used in over 5.2 million vehicles

# User Story: Bipedal Robot

## The Challenge

Develop a control system for an underactuated bipedal robot with 13 degrees of freedom

## The Solution

Use Model-Based Design with MATLAB and Simulink to model the legs and torso, develop and simulate the control algorithms, and generate code for the real-time implementation

## The Results

- Controller development accelerated
- Focus on high-level objectives maintained
- Approach adopted at other institutions



“When other researchers see that we’ve gone directly from controllers developed in MATLAB and Simulink to a real-time implementation with Simulink Real-Time, they get pretty excited. The approach we took is now being used in other departments at the University of Michigan and by robotics researchers at other universities, including MIT and Oregon State University.”

- Prof. Jesse Grizzle,  
University of Michigan

# Preceyes

## First Successful Eye Surgeries Performed on the Preceyes System

SEPTEMBER 13TH, 2016 TOM PEACH OPTHALMOLOGY



“Simulink and Simulink Real-Time enabled controls algorithm design and implementation of these algorithms on the device,” Gerrit Naus, COO and co-founder at Preceyes.

% Thank you