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

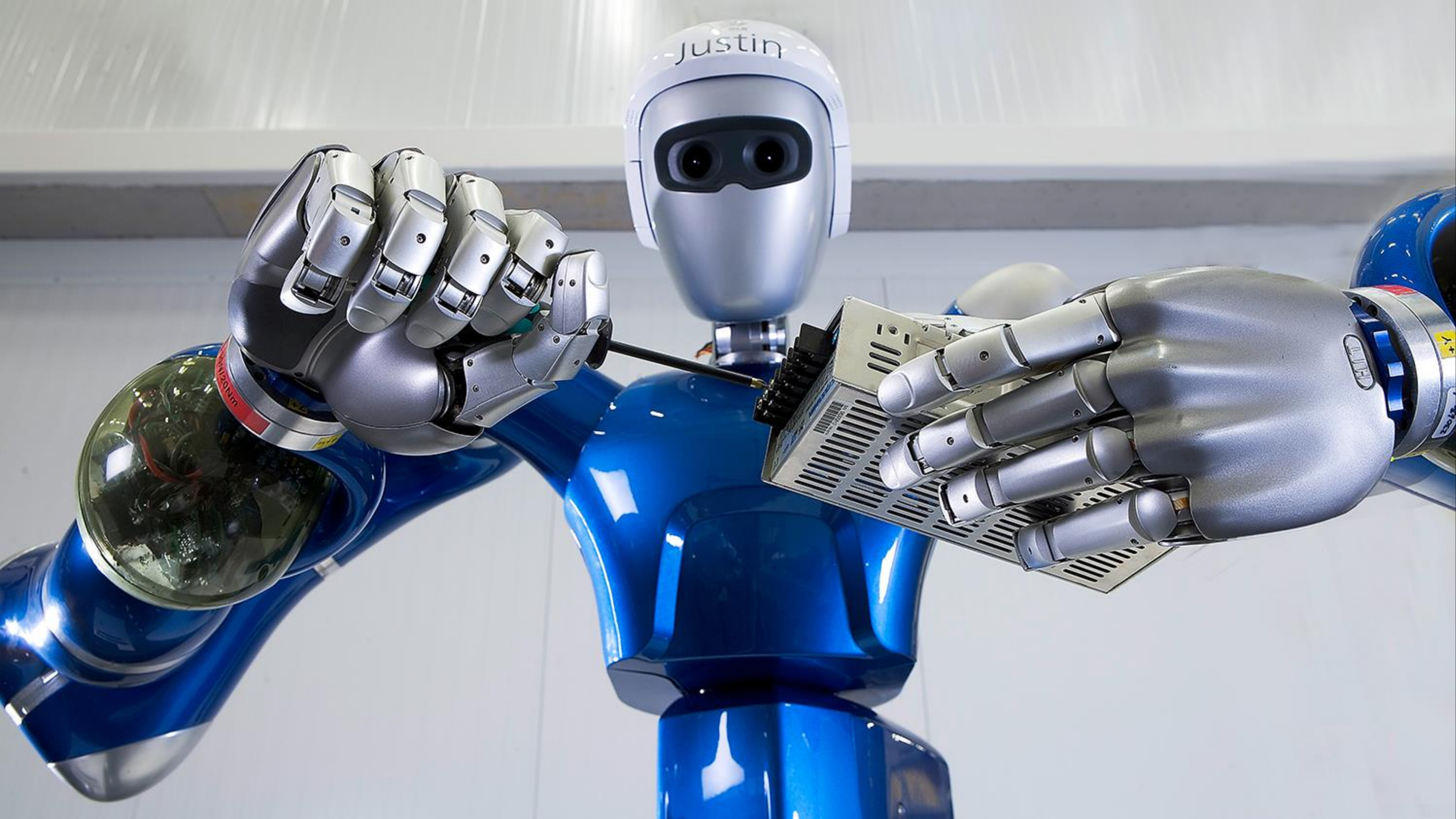
Complexity of Algorithms

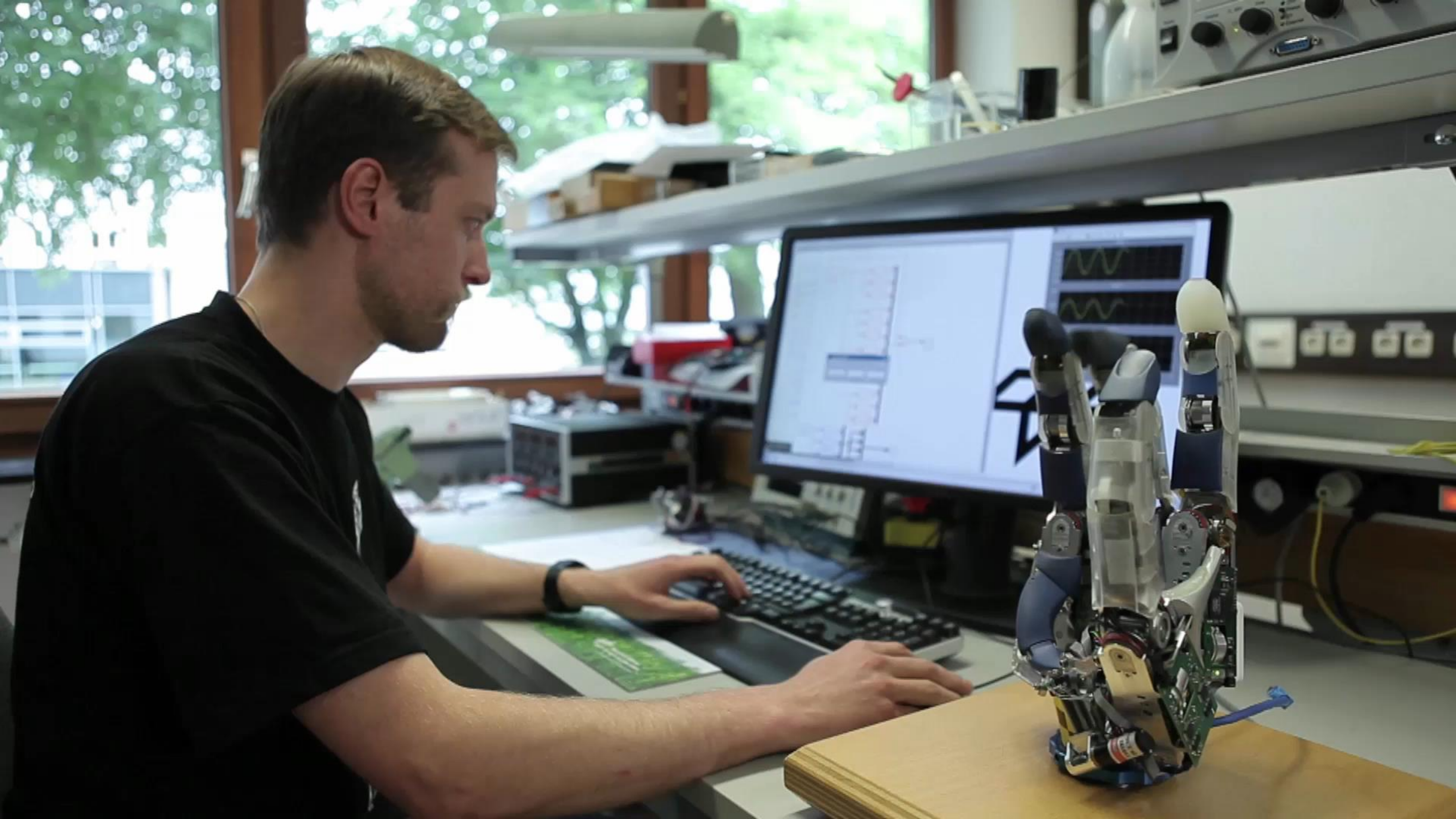
End-to-End workflows

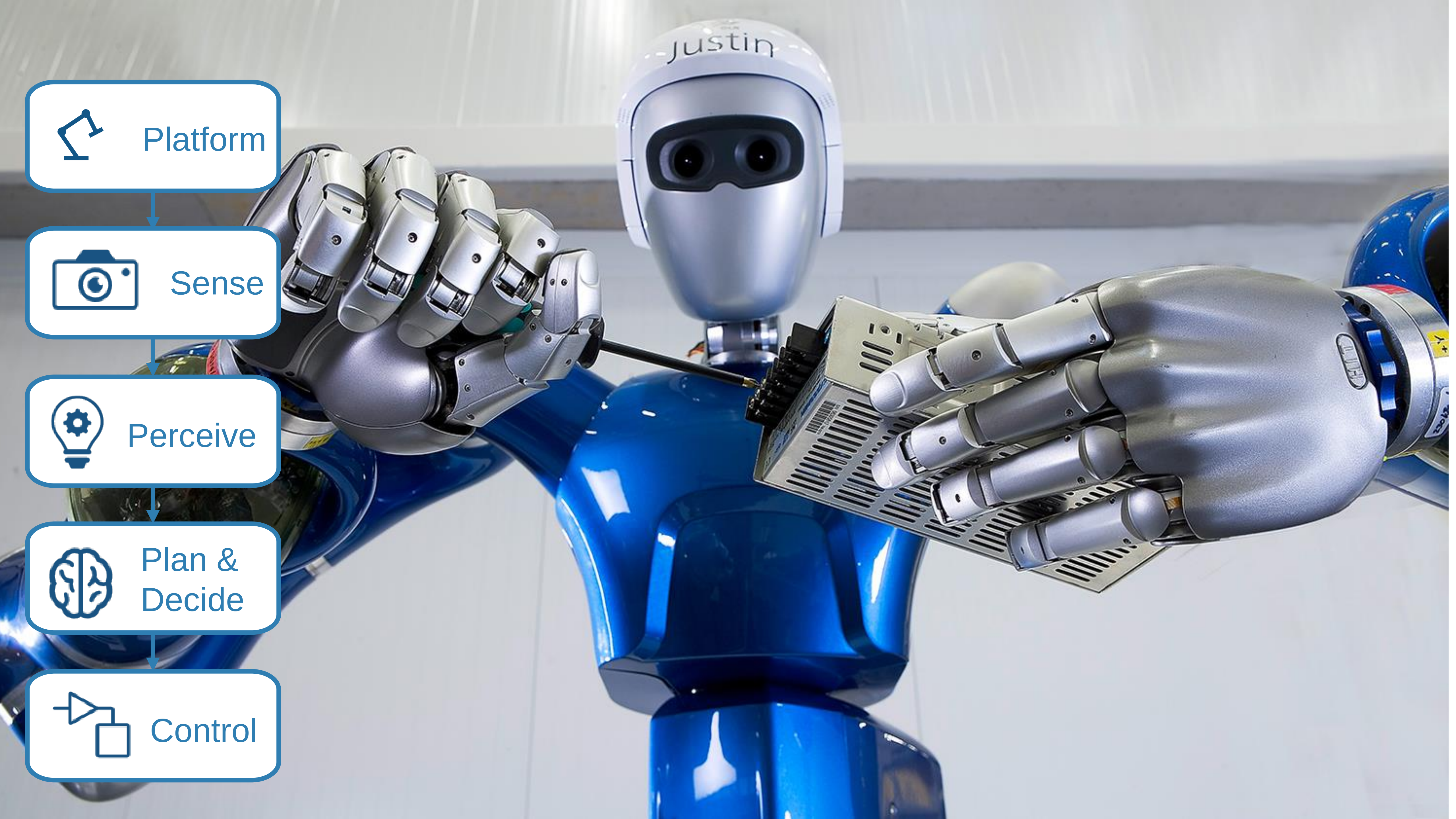
Technical Depth and System Stability

IP Protection

What does success look like?

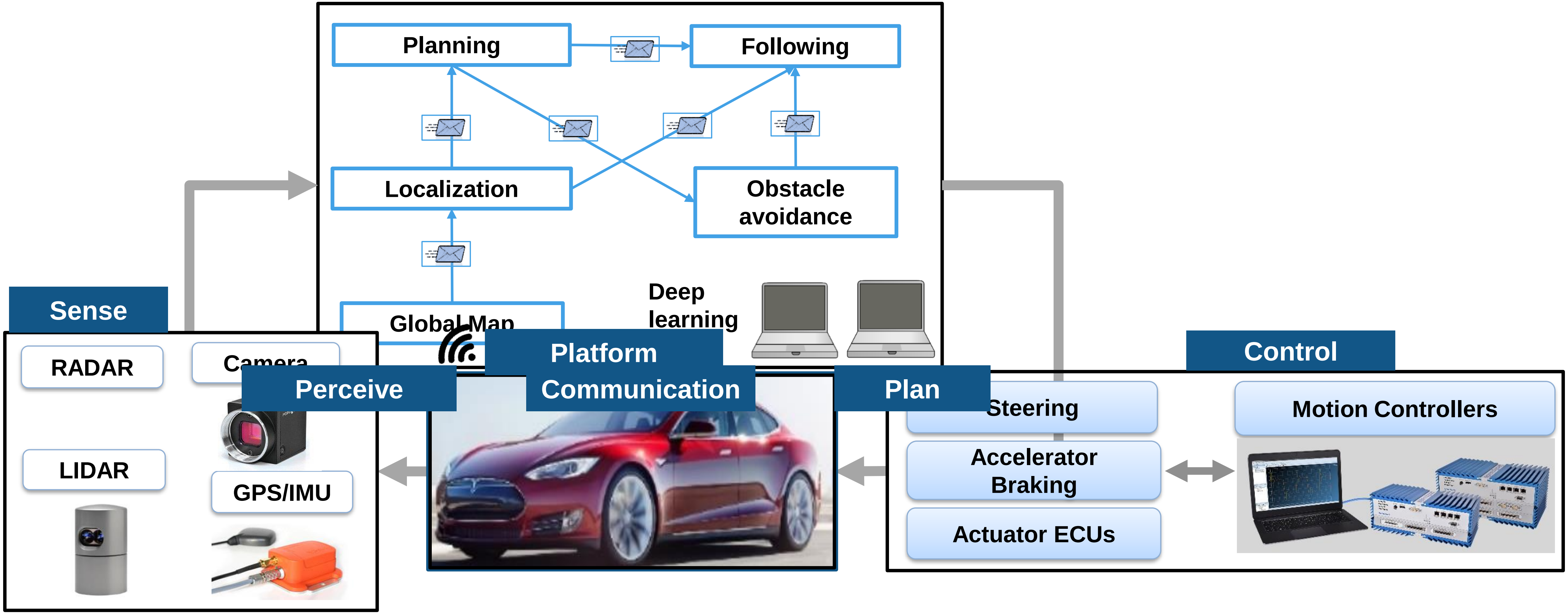

Justin





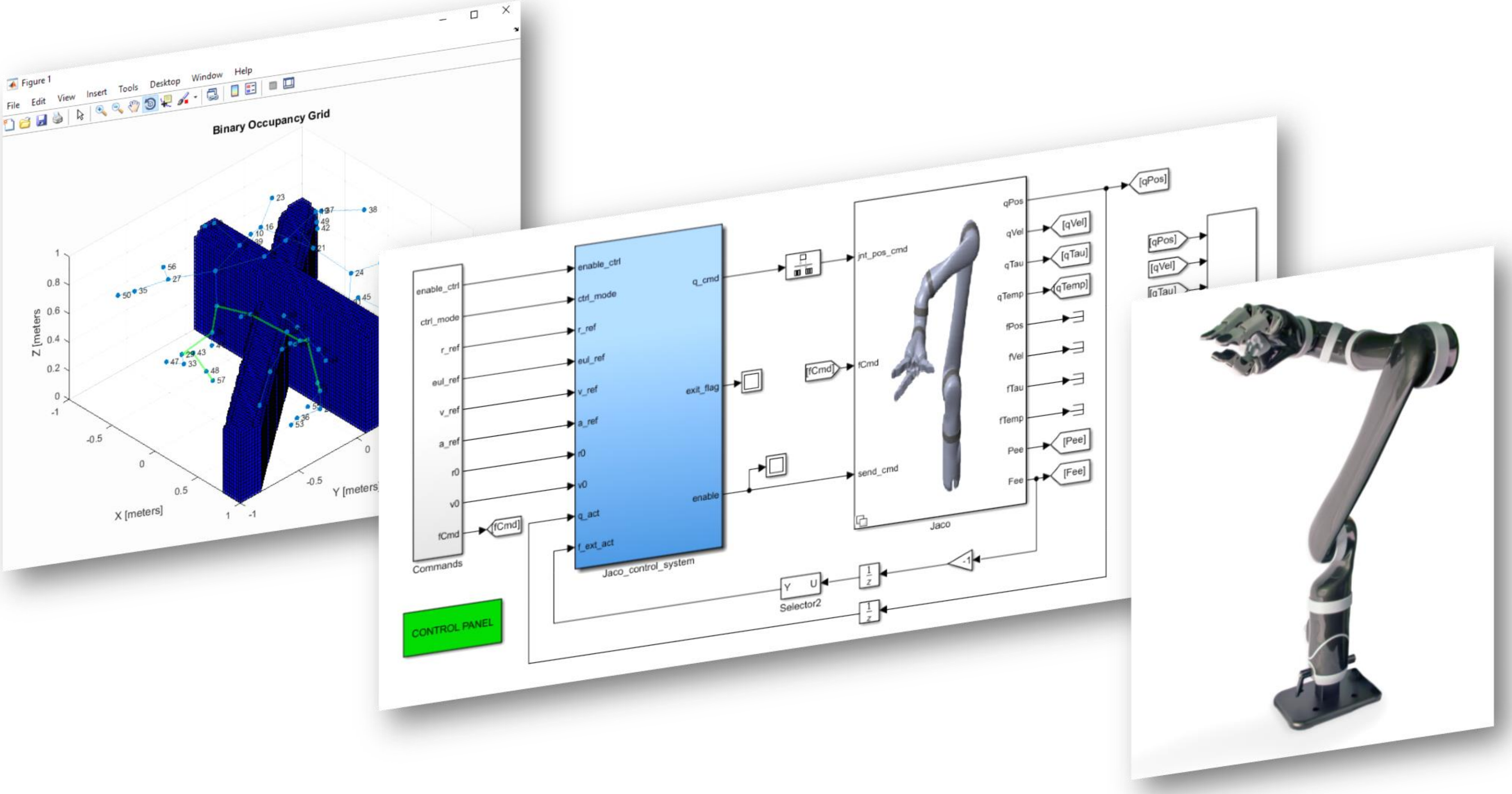
Platform
Sense
Perceive
Plan &
Decide
Control
Justin

Another Example: Self-Driving Cars



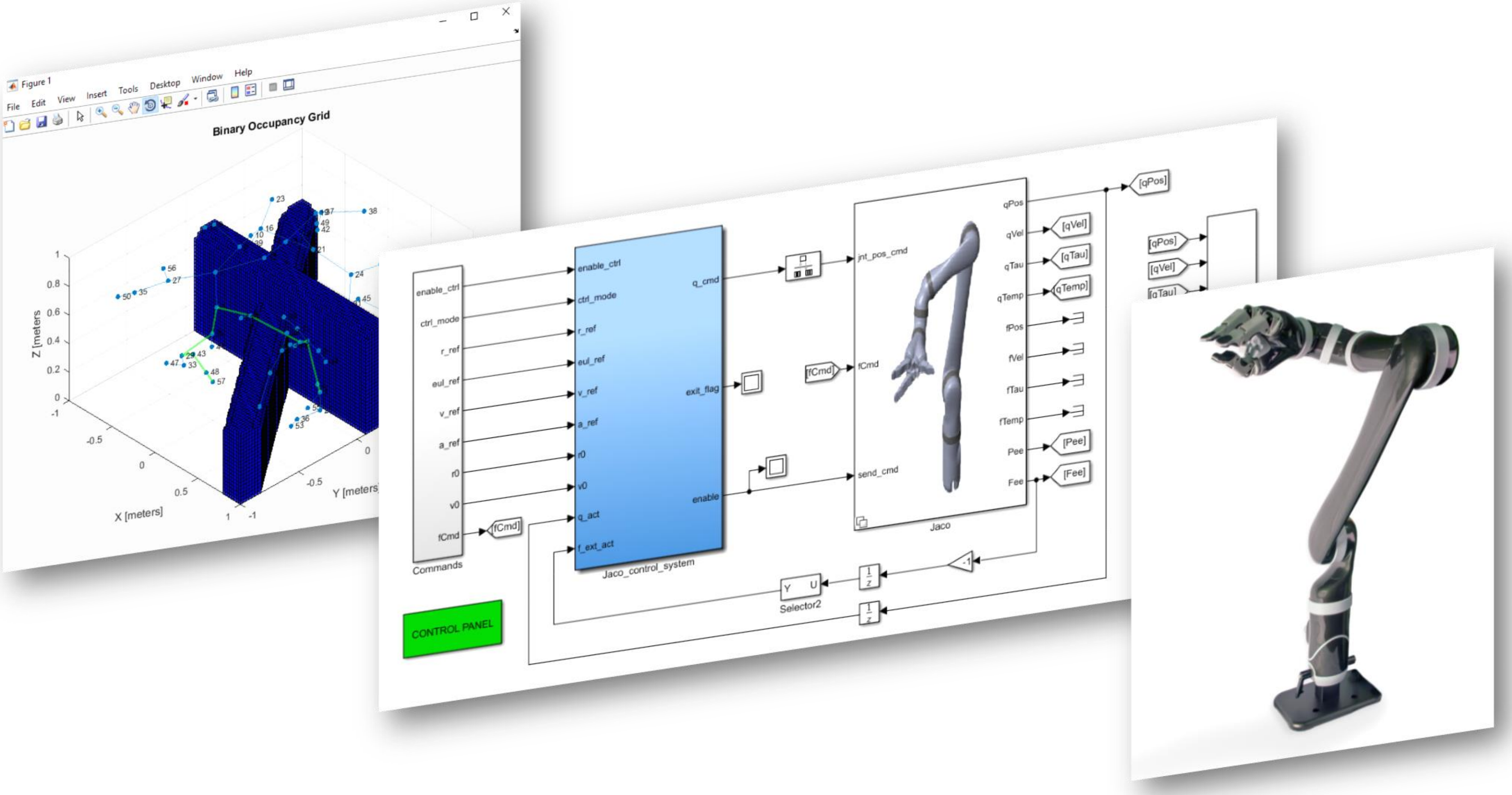
Today: Design Pick and Place Application

- Platform
- Sense
- Perceive
- Plan & Decide
- Control



Today: Design Pick and Place Application

- Platform
- Sense
- Perceive
- Plan & Decide
- Control



Platform Design

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

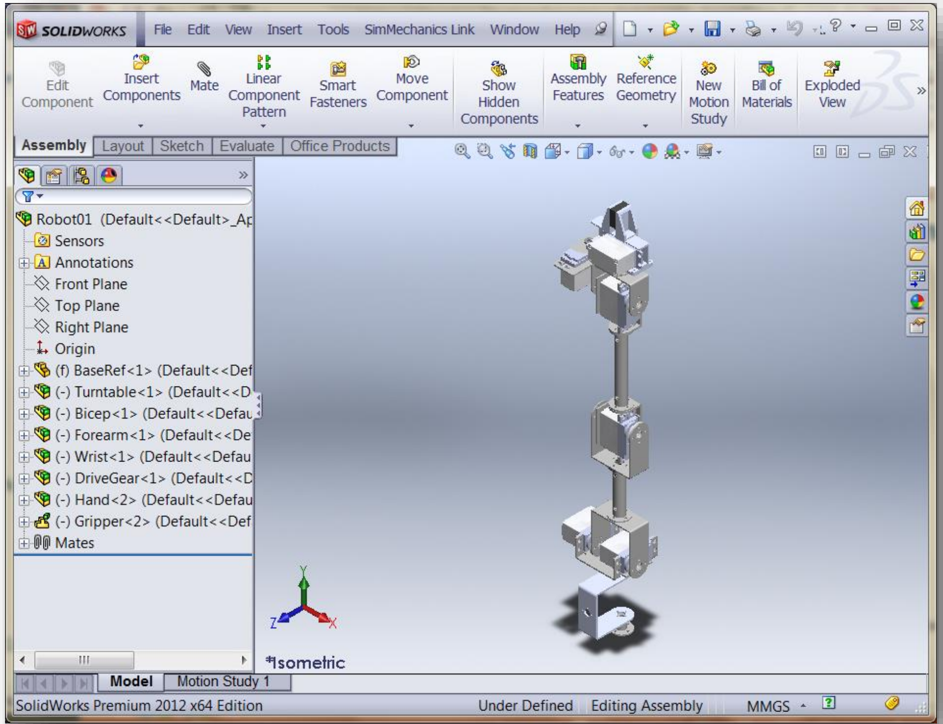
Mechanics

Actuators

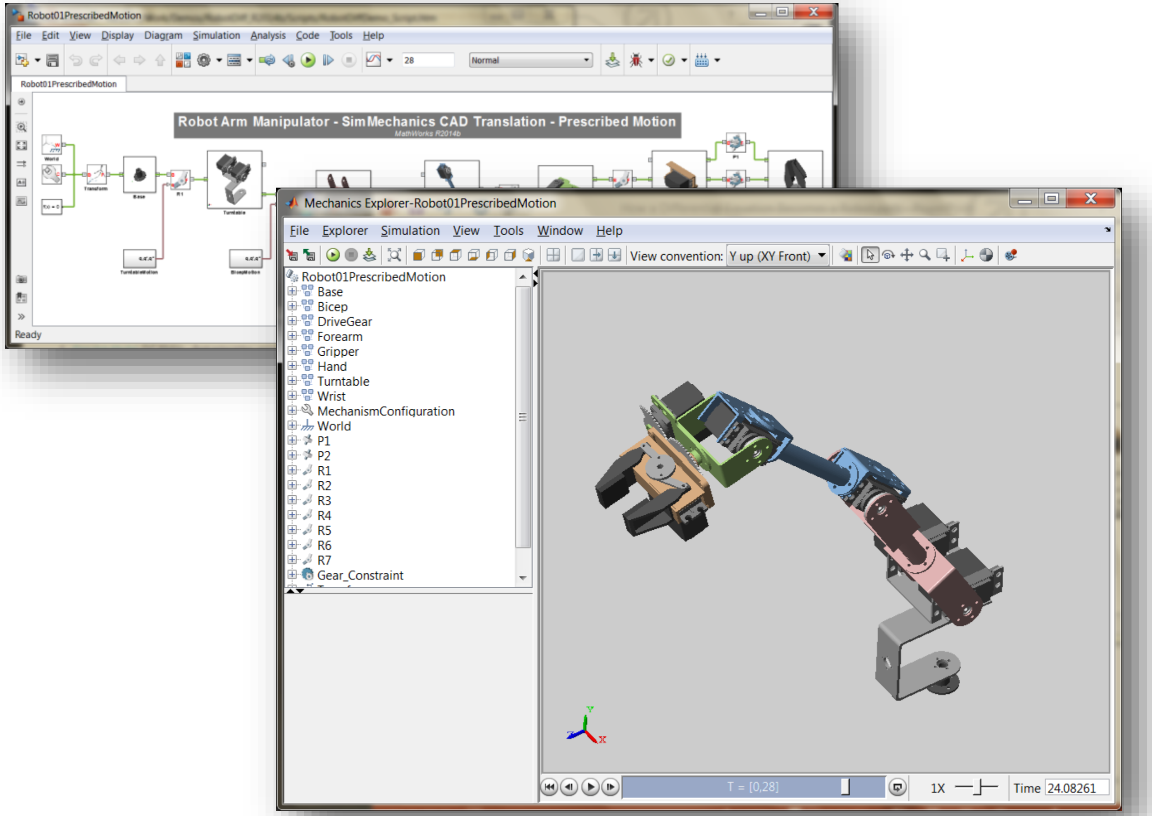
Environment

Mechanics: Import models from common CAD Tools

SolidWorks Model

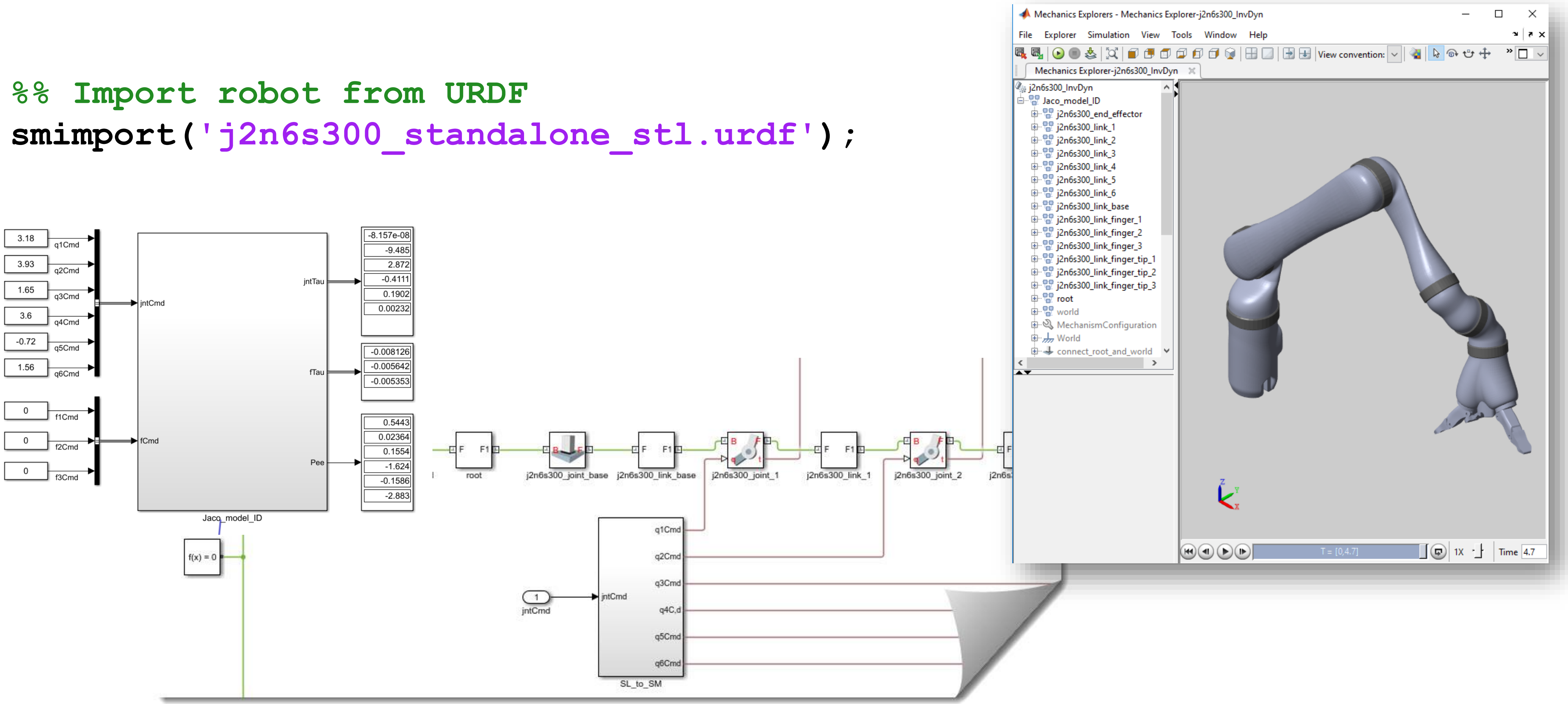


Simscape Multibody Model

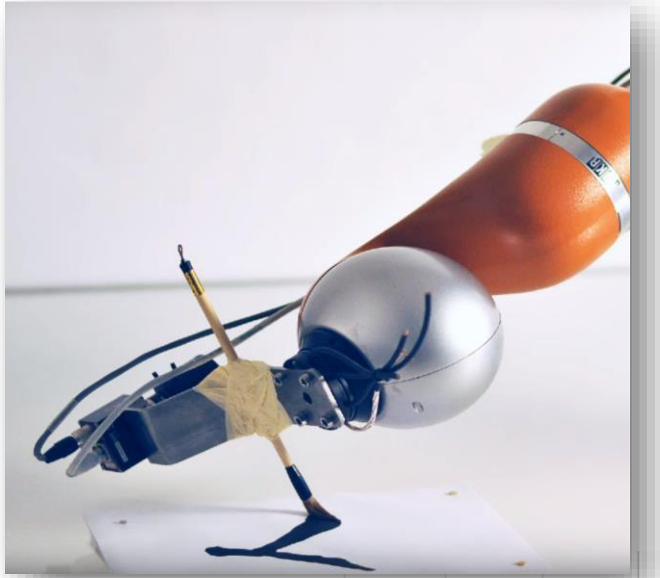


Mechanics: One line import from URDF

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

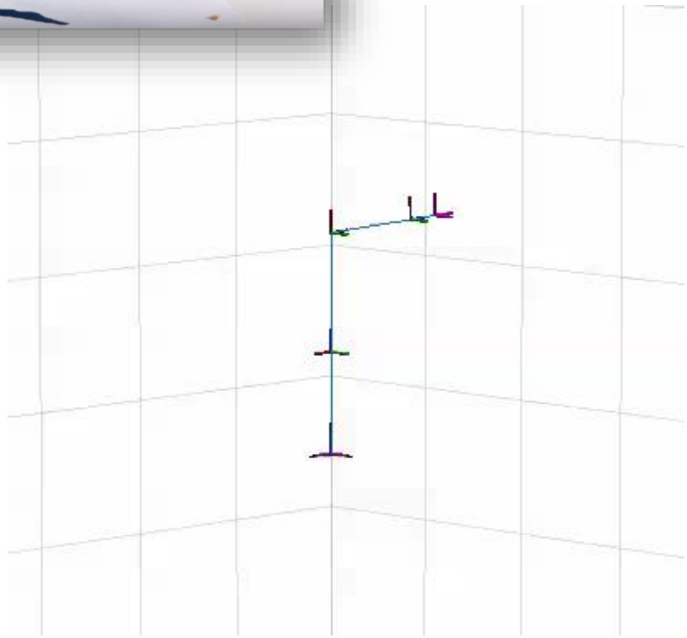


Rigid Body Tree Dynamics



Compute rigid body tree dynamics quantities

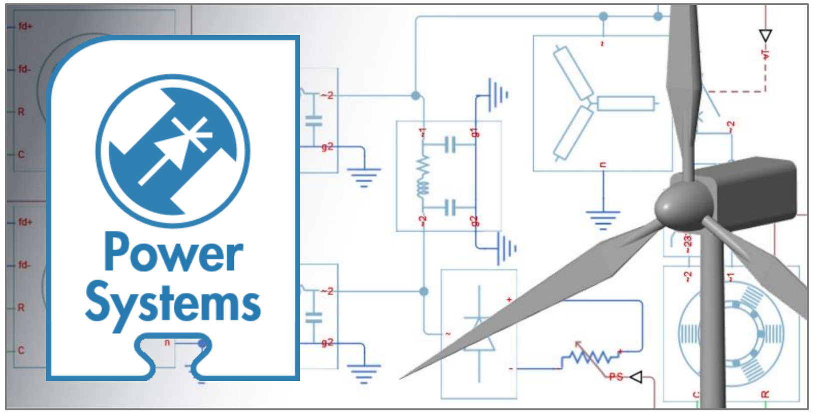
- Specify rigid body inertial properties
- Compute for the rigid body tree
  - Forward dynamics
  - Inverse dynamics
  - Mass matrix
  - Velocity product
  - Gravity torque
  - Center of mass position and Jacobian

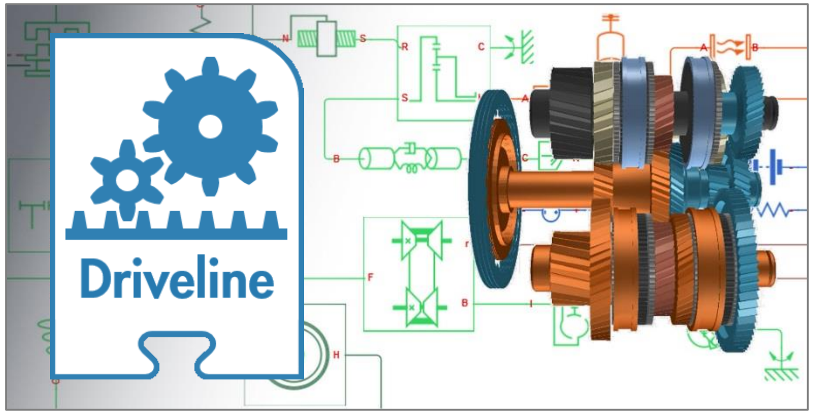


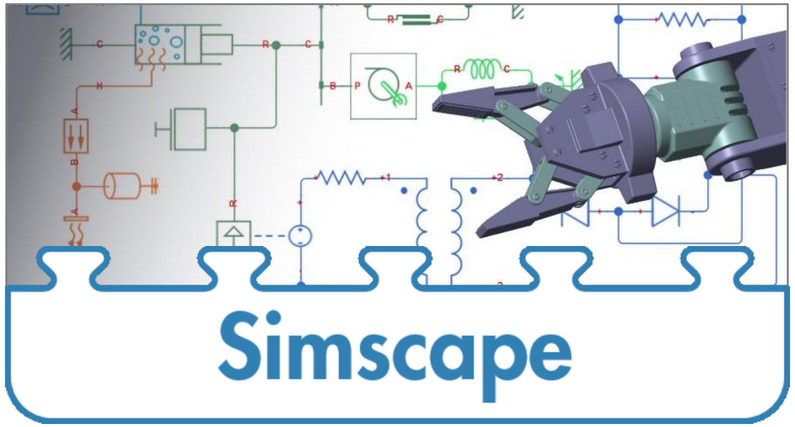
```
» load exampleRobots.mat
» lbr.DataFormat = 'column';
» q = lbr.randomConfiguration;
» tau = inverseDynamics(lbr, q);
```

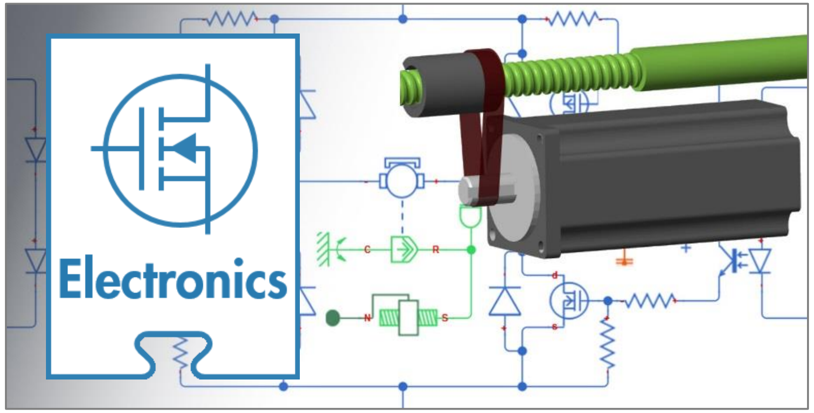
Actuators: Connect Motors

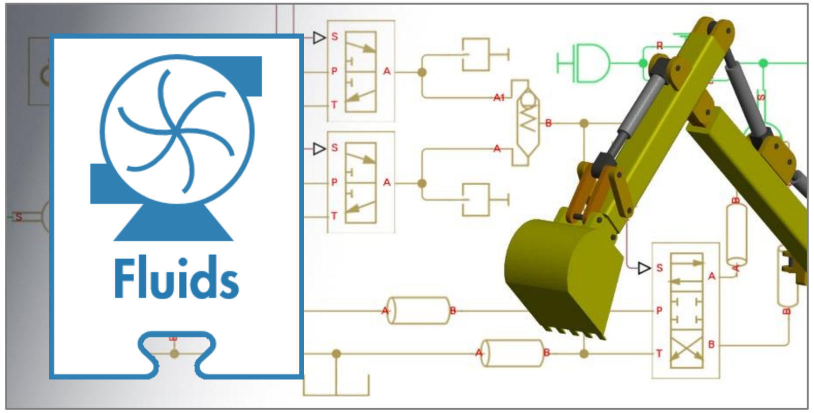
Actuators: Model other domains

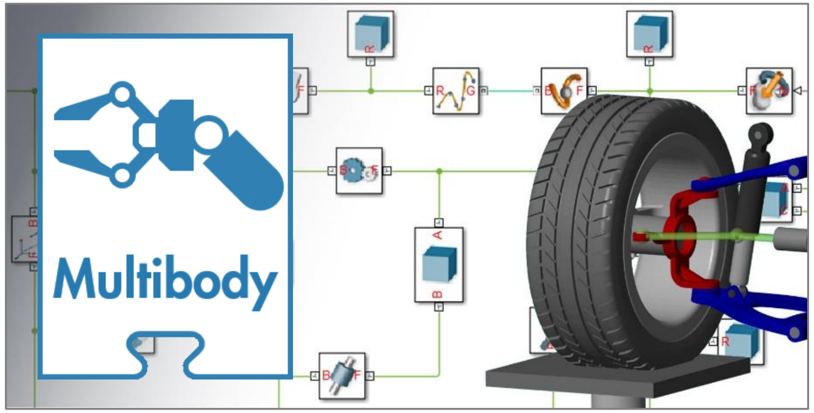






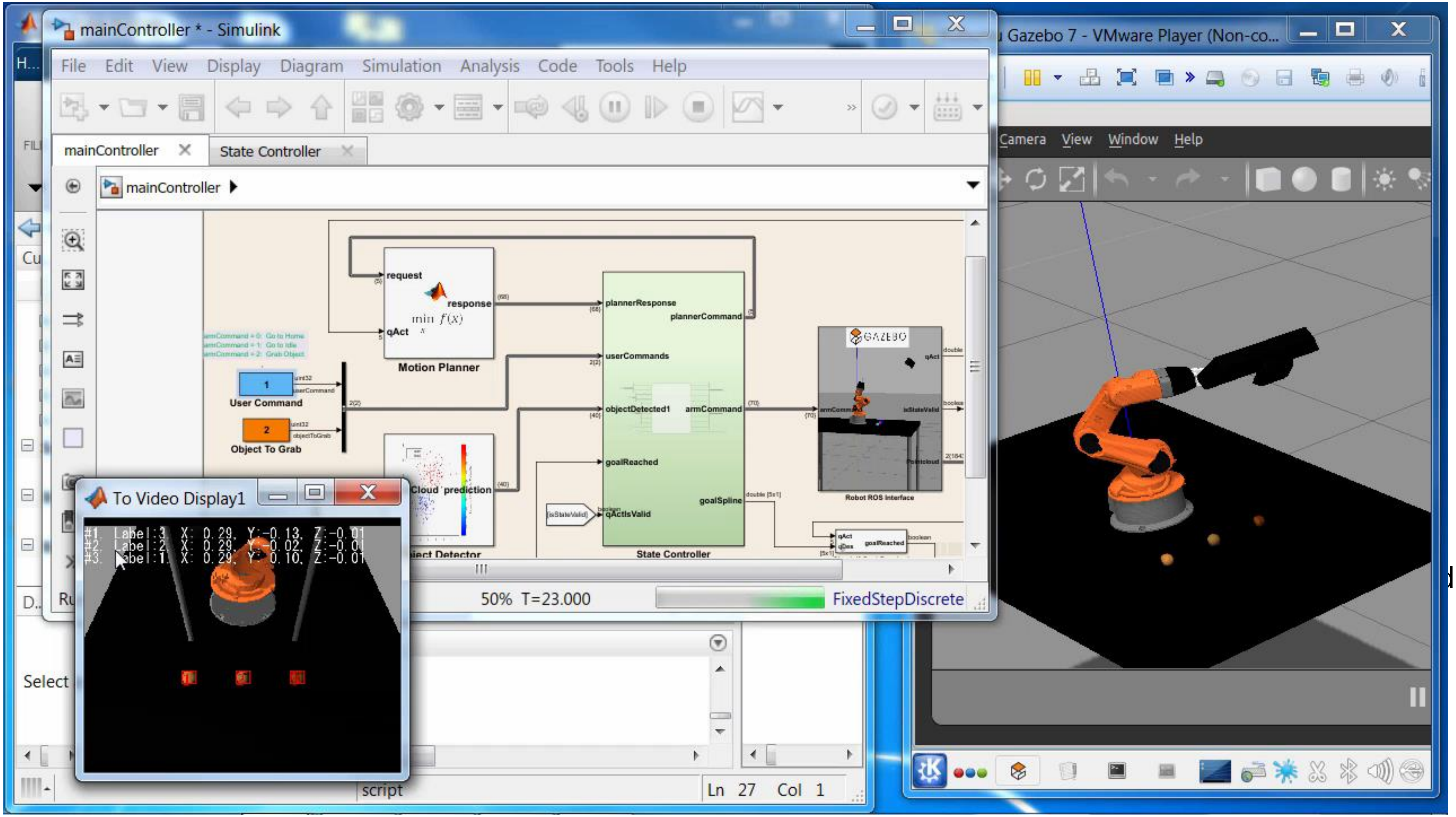




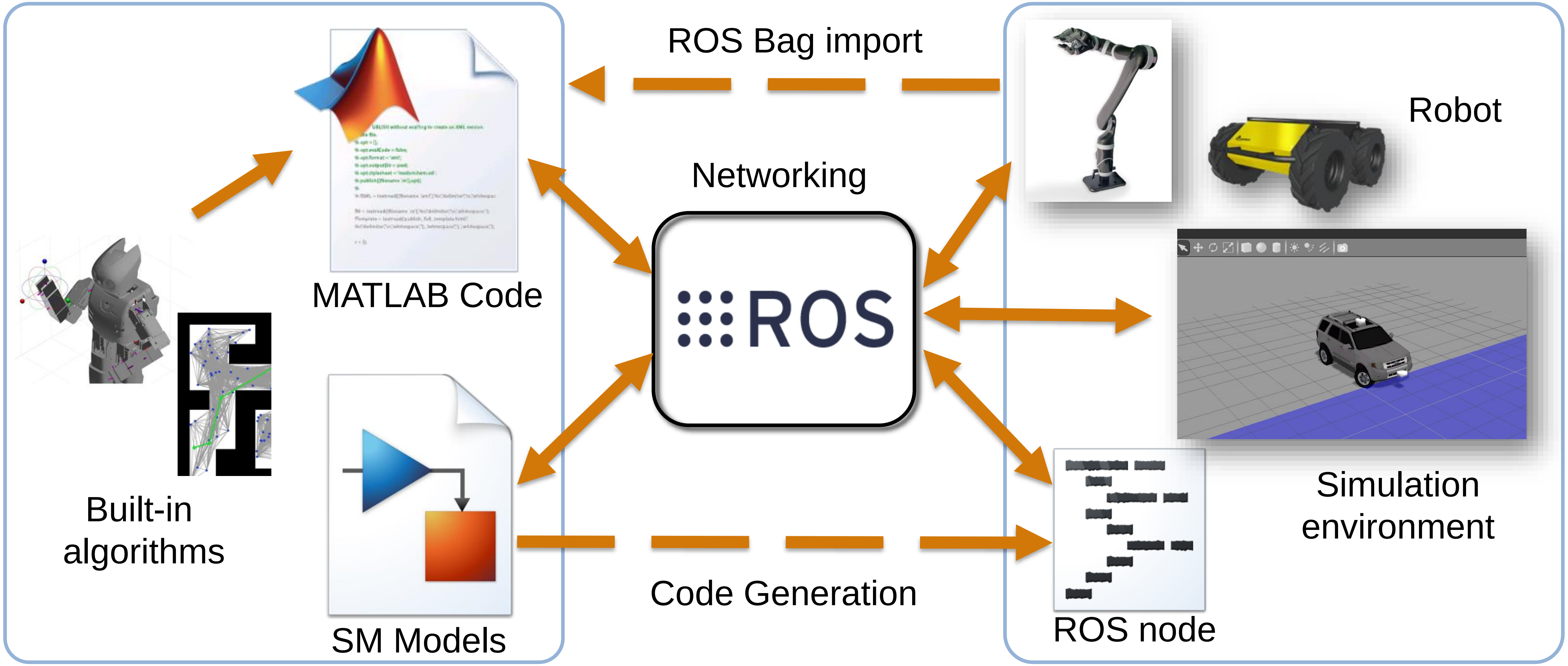


Environment: Connect to an external robotics simulator

Environment: Connect to an external robotics simulator

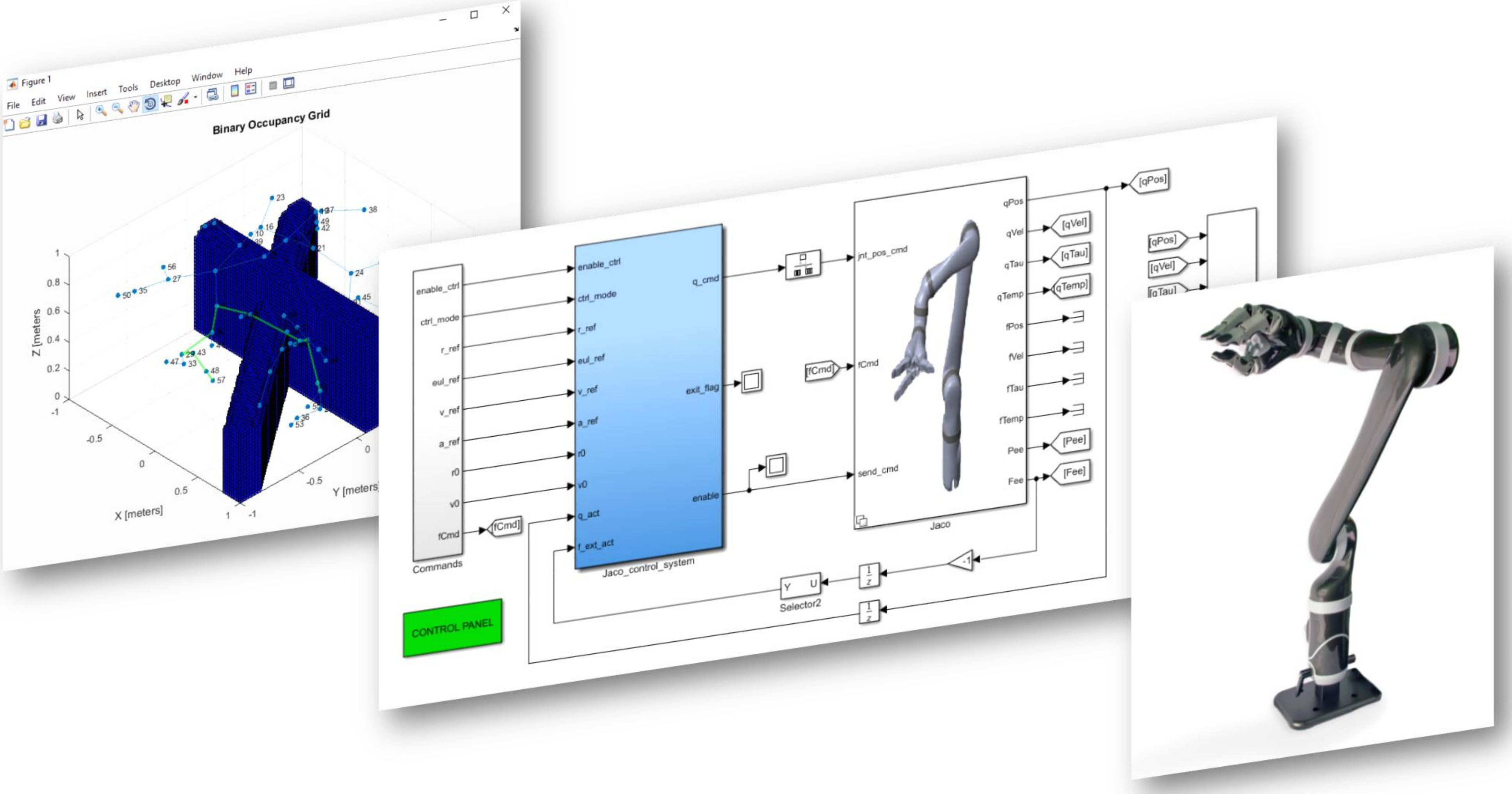


Environment: Connect MATLAB and Simulink with ROS

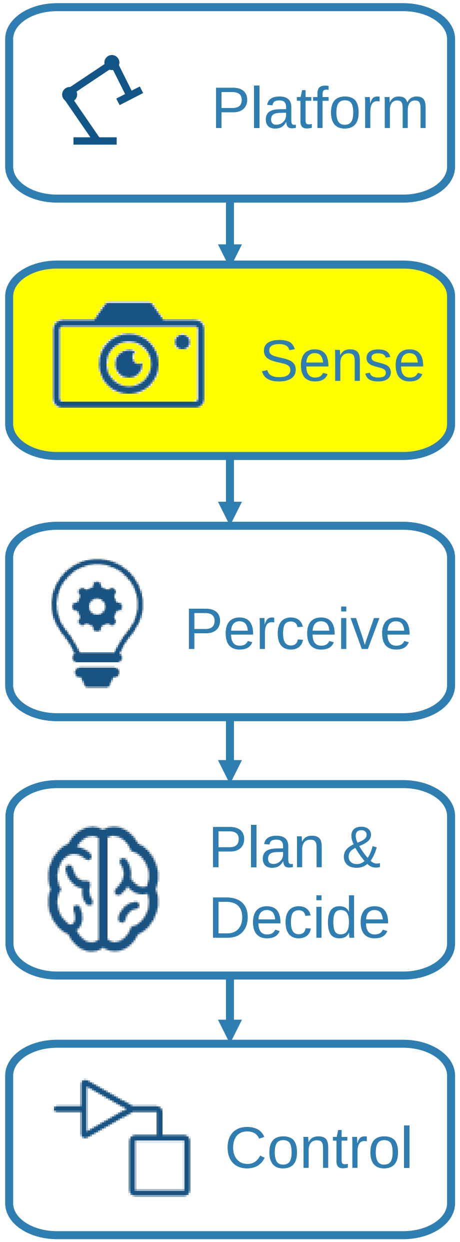


Today: Design Pick and Place Application

- Platform
- Sense
- Perceive
- Plan & Decide
- Control



Today: Design Pick and Place Application



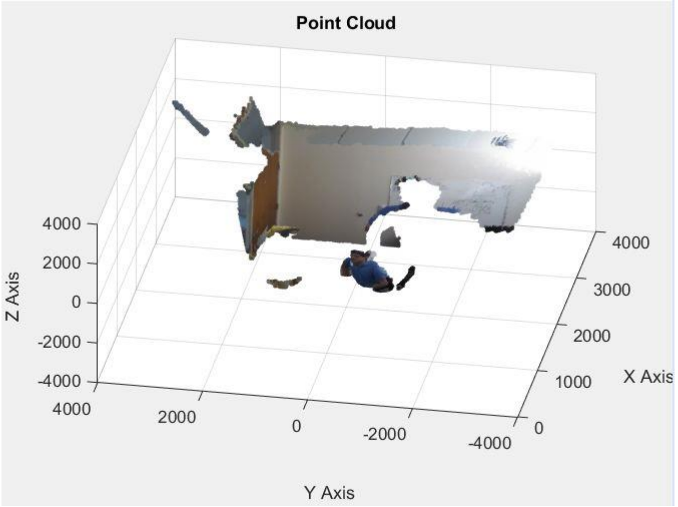
Support for Common Sensors
Cameras, Laser Scanners, Optical Encoders, IMU, GPS.

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

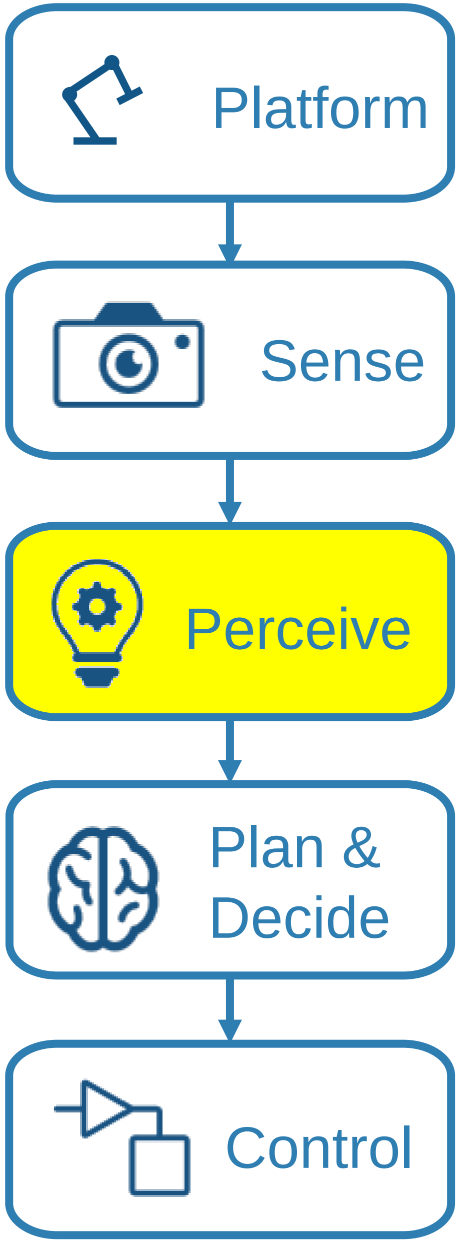
Apps for image region analysis, image batch processing, and image registration

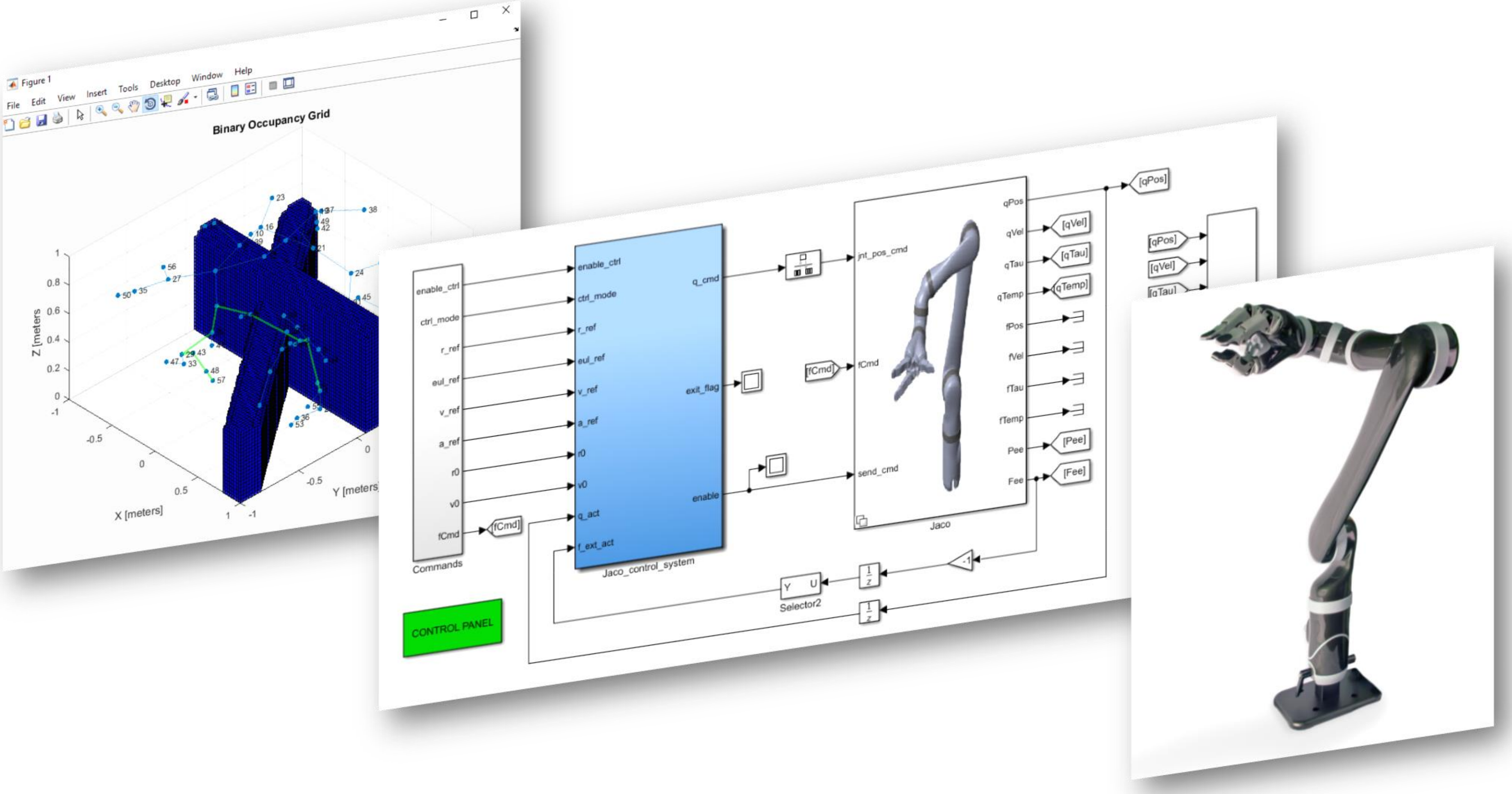
Image enhancement, filtering, geometric transformations, and deblurring algorithms
Intensity-based and non-rigid image registration.

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

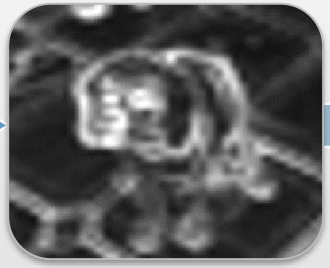
Sign Classifier

Labels

MATLAB makes machine learning easy and accessible

Traditional Machine Learning approach

Traditional Feature Extraction

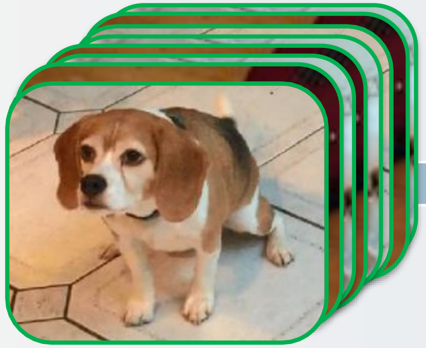


Classification

Machine Learning

- **Dog** ✓
- **Boy** ✗
- •
- •
- **Bicycle** ✗

Deep Learning approach



Convolutional Neural Network (CNN)

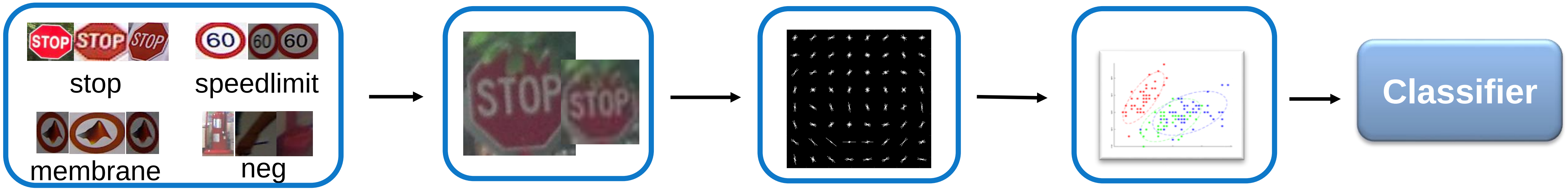
Learned features [95%]

End-to-end learning

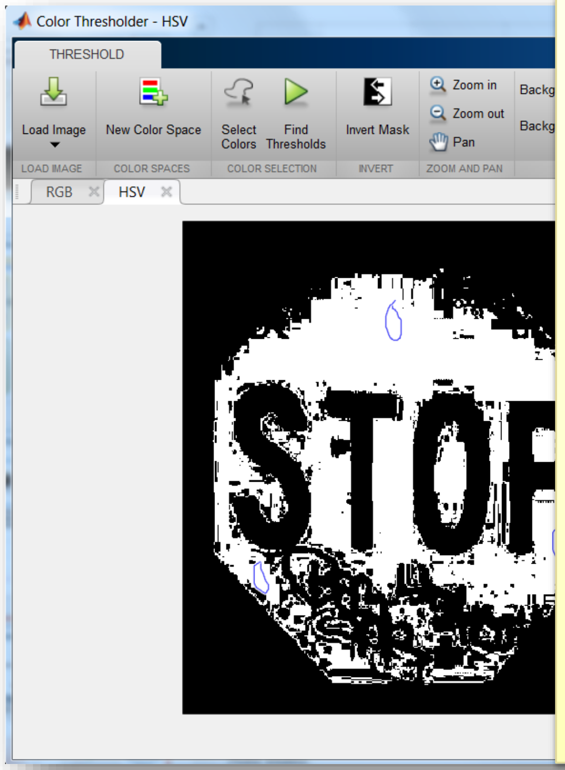
Feature learning + Classification

- **Dog** ✓
- **Boy** ✗
- •
- •
- **Bicycle** ✗

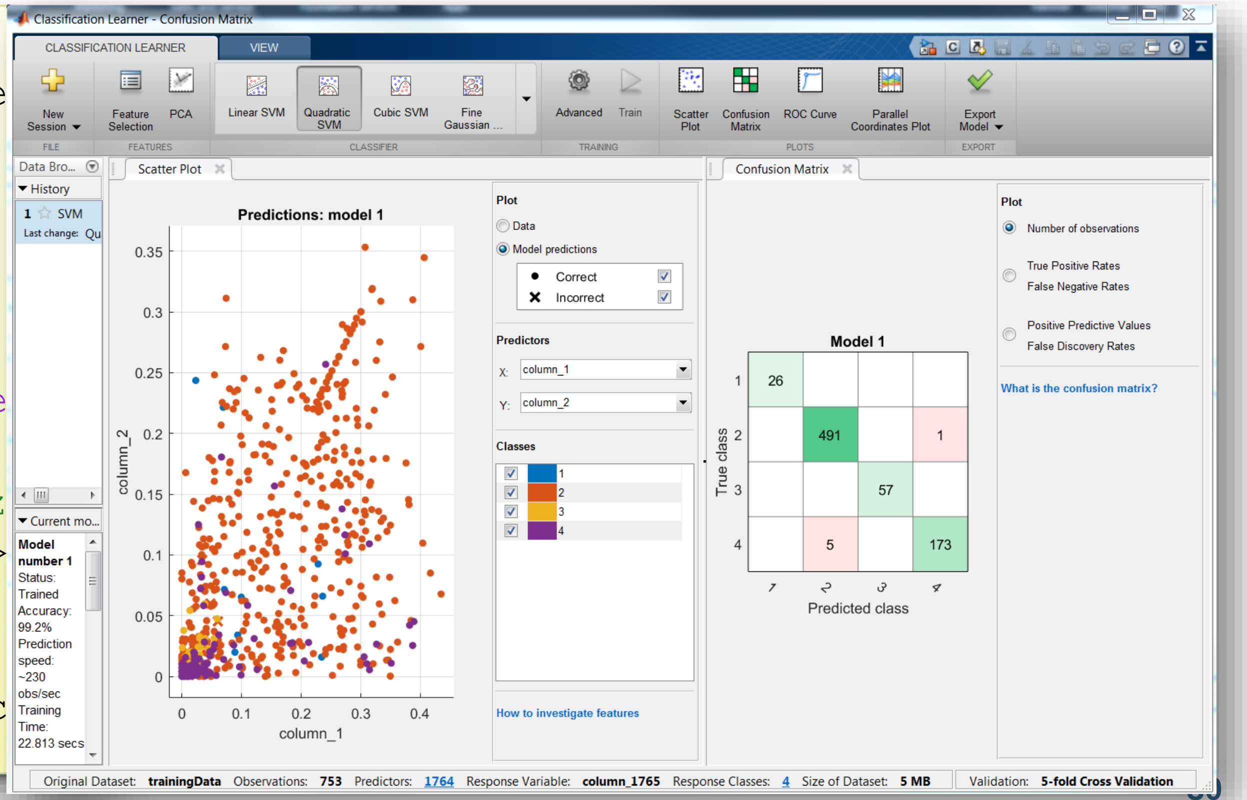
Complex workflows made easy with MATLAB



Training data → Preprocessing → Feature Extraction → Training → Classifier

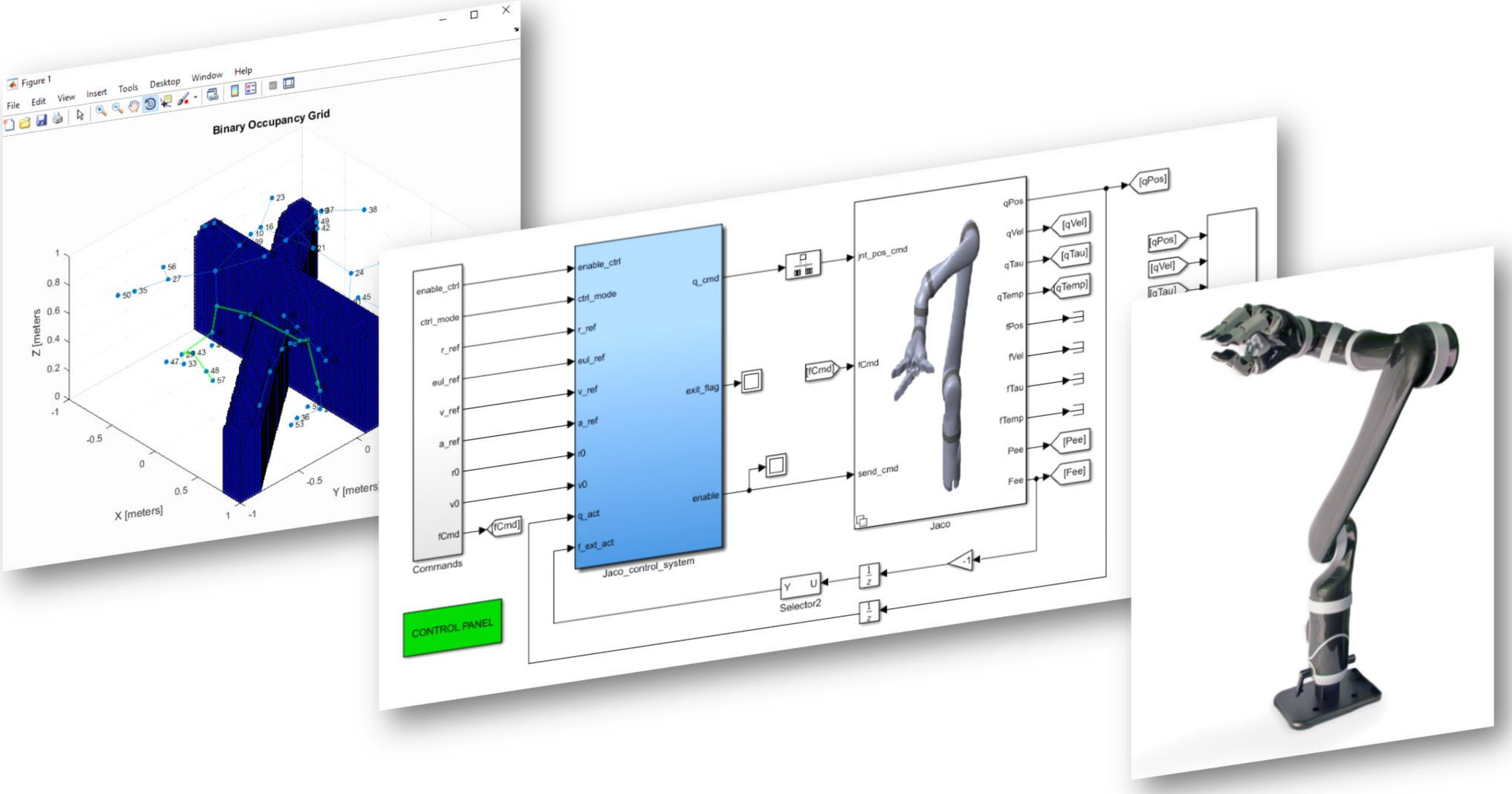



```
% Detect red regions
BW = createMask(videoFrame

% 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
```

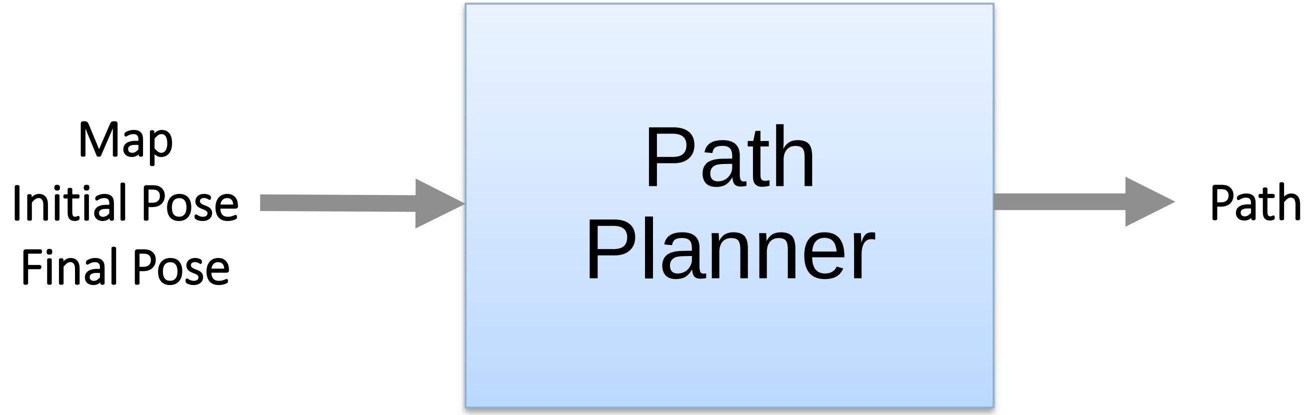


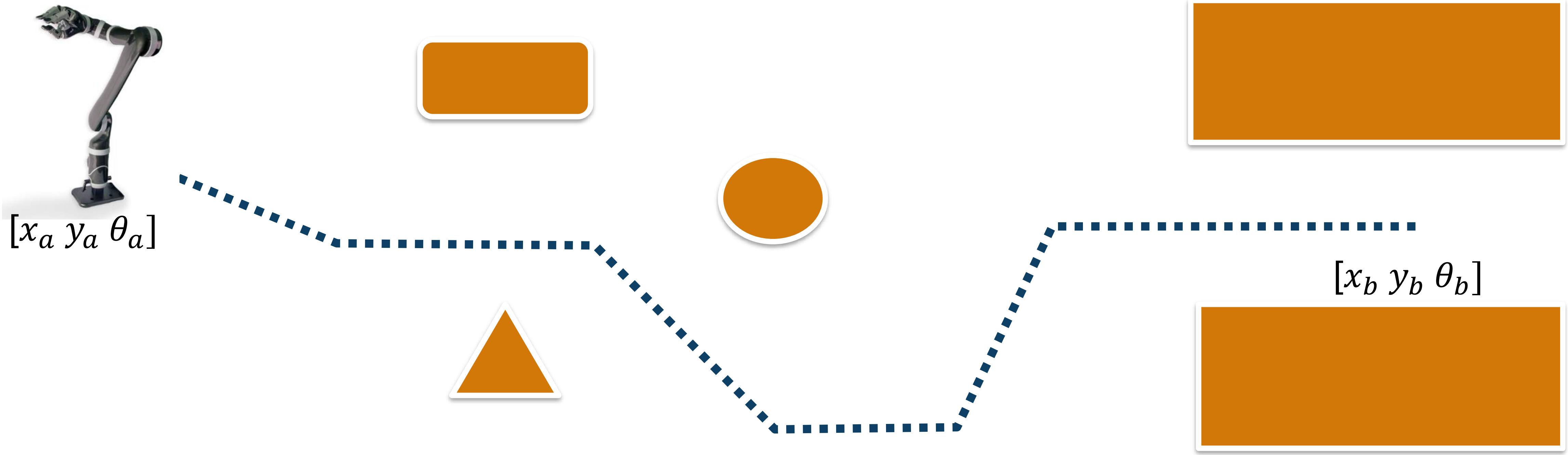
Today: Design Pick and Place Application

- Platform
- Sense
- Perceive
- **Plan & Decide**
- Control

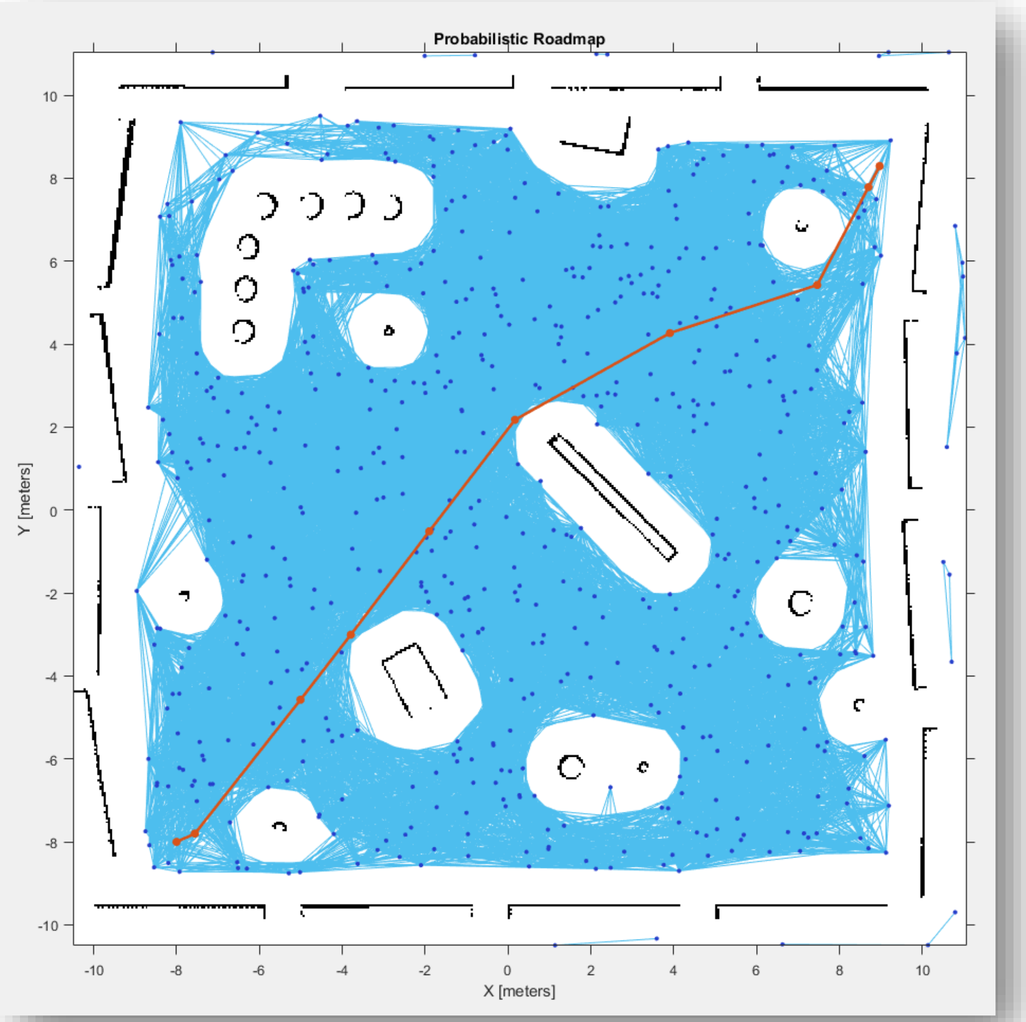


Planning


Map
Initial Pose
Final Pose
Path Planner
Path


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

Explore Built In Functions: PRM Planner

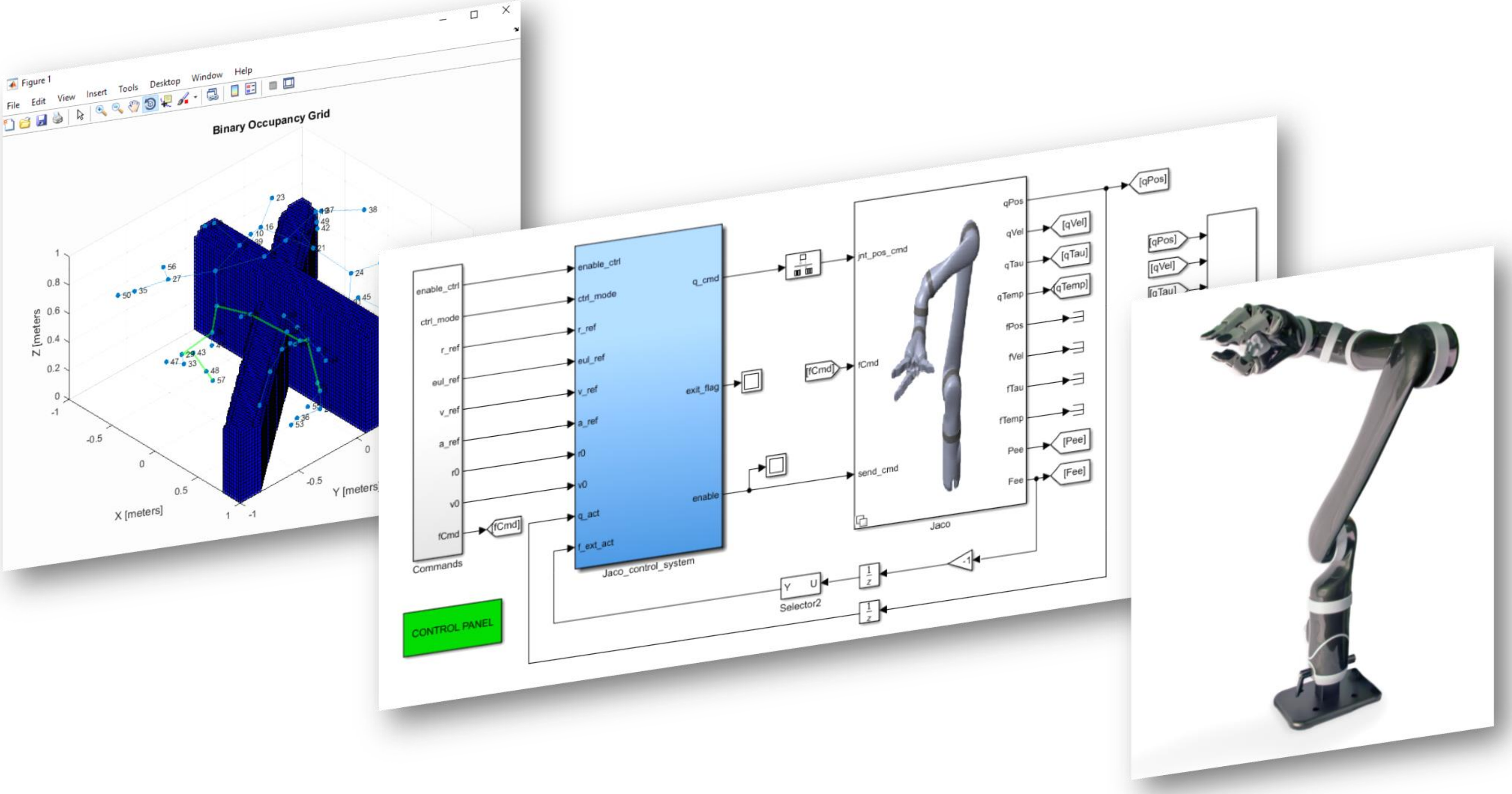



```
%% Create map as Binary Occupancy Grid
robotics.BinaryOccupancyGrid(binaryImage,20);
og.GridLocationInWorld = [-10.5, -10.5];
figure;
show(og);
```

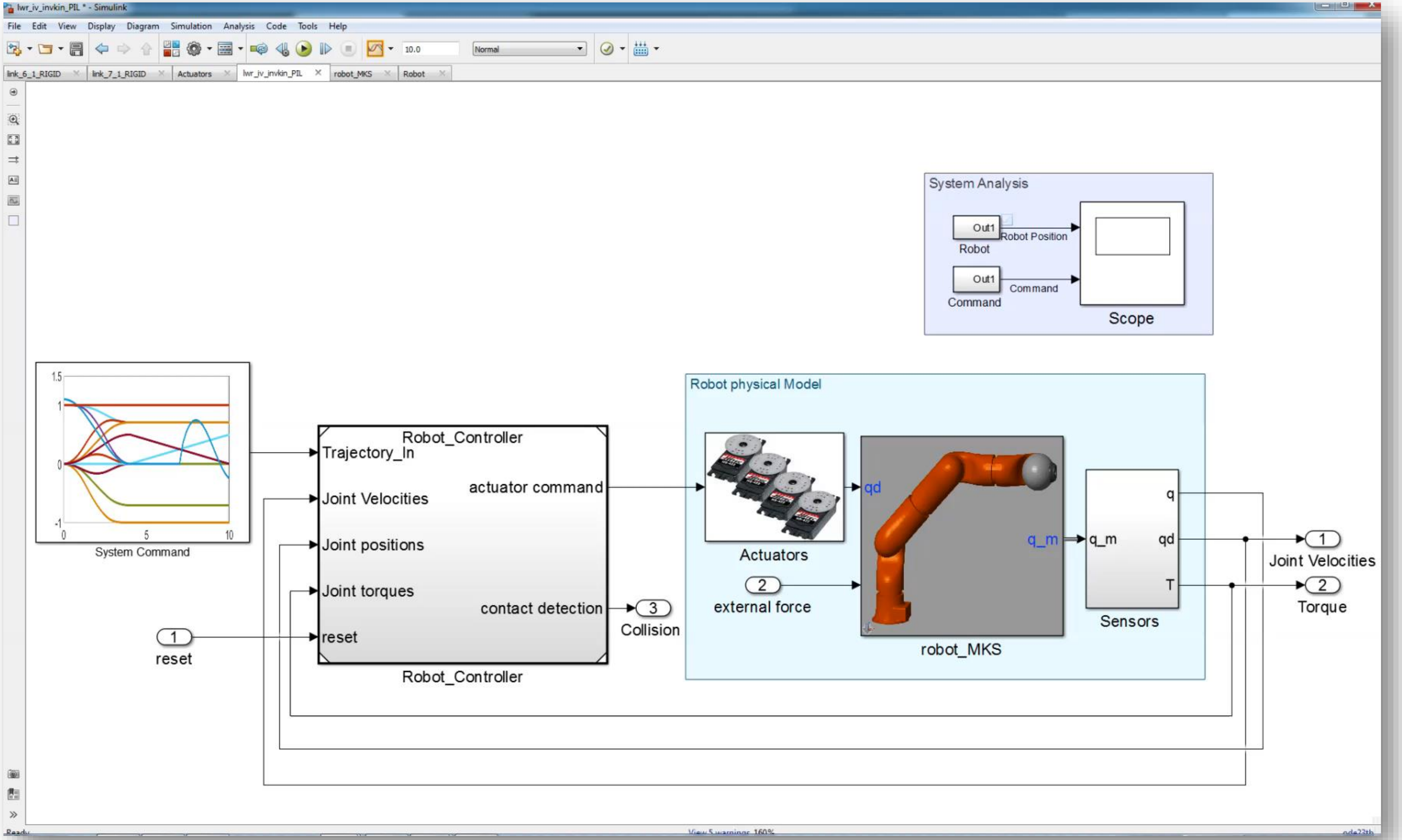
```
%% Create PRM path planner
planner = robotics.PRM;
planner.Map = og;
planner.NumNodes = 600;
planner.ConnectionDistance = 5;
show(originalOg); hold on;
show(planner, 'Map', 'off');
```

Today: Design Pick and Place Application

- Platform
- Sense
- Perceive
- Plan & Decide
- **Control**

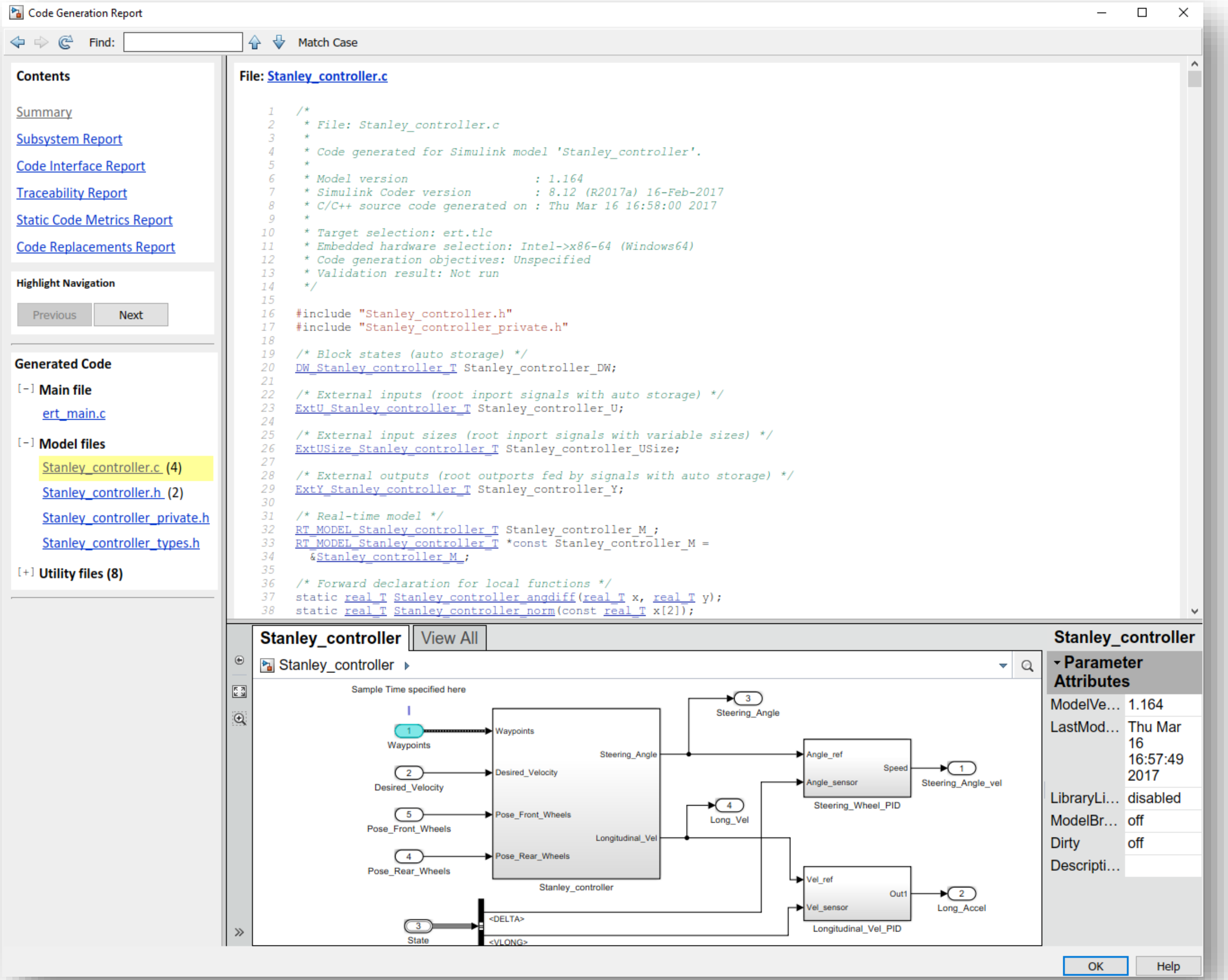


Control of Manipulator Arms



Model-Based Design for Motion Control

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



MathWorks

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

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

Link to user story

Festo Develops Innovative Robotic Arm Using Model-Based Design

The Festo Bionic Handling Assistant. Image © Festo AG.

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

"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

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

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

"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

[Link to article]

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

Patric Schenk, VP of Sales and Engineering, Speedgoat
12-May-2015

Mobileye chips are used in over 5.2 million vehicles

User Story: Bipedal Robot

The Challenge

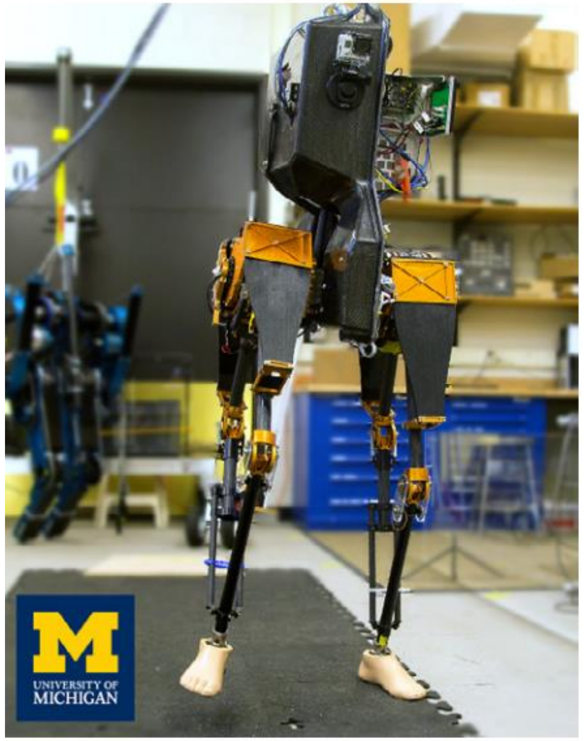
Develop a control system for an underactuated bipedal robot with 13 degrees of freedom

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



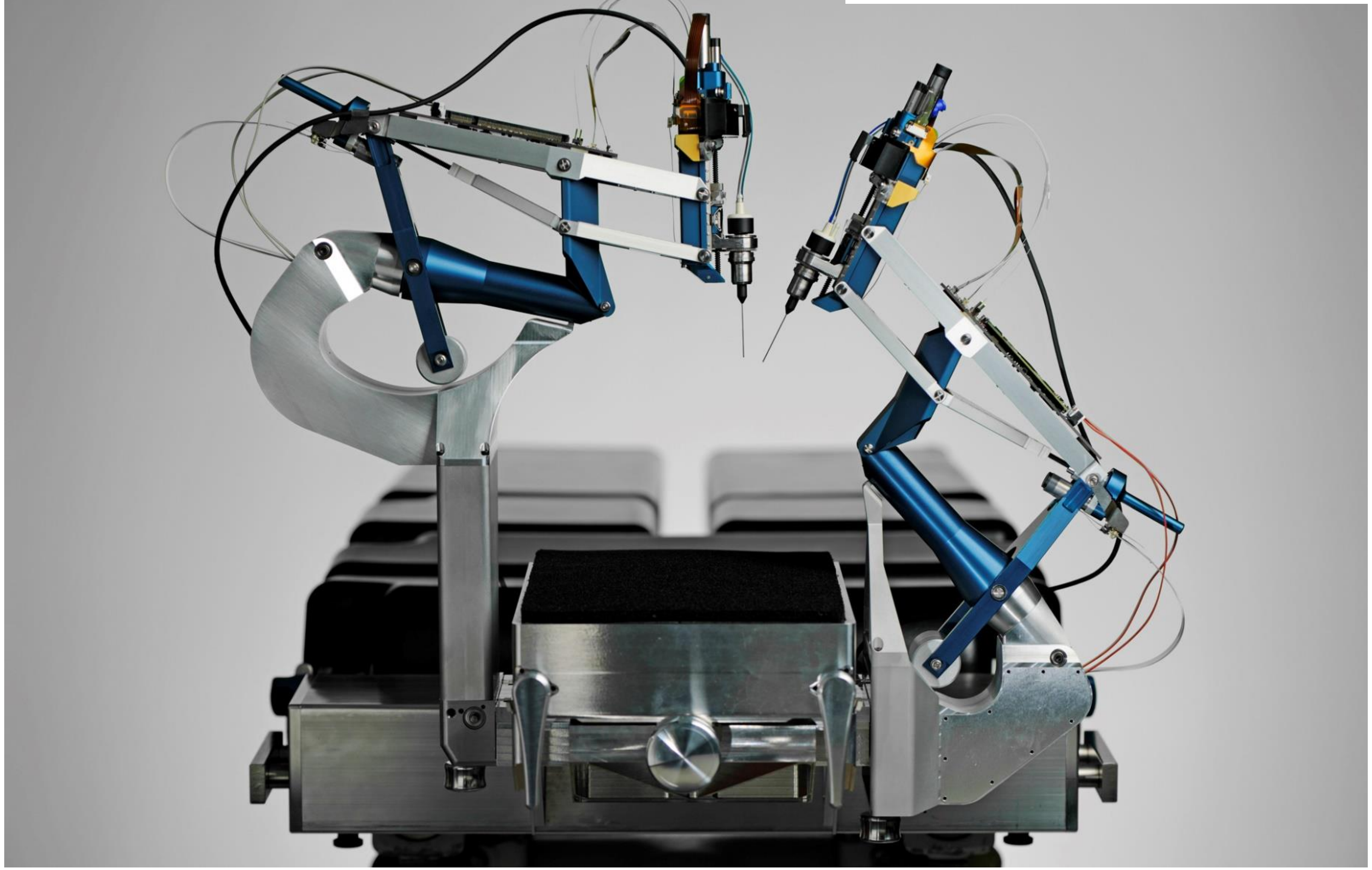
"When other researchers see that we've gone directly from controllers developed in MATLAB and Simulink to a real-time implementation with Simulink Real-Time, they get pretty excited. The approach we took is now being used in other departments at the University of Michigan and by robotics researchers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



“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
```