

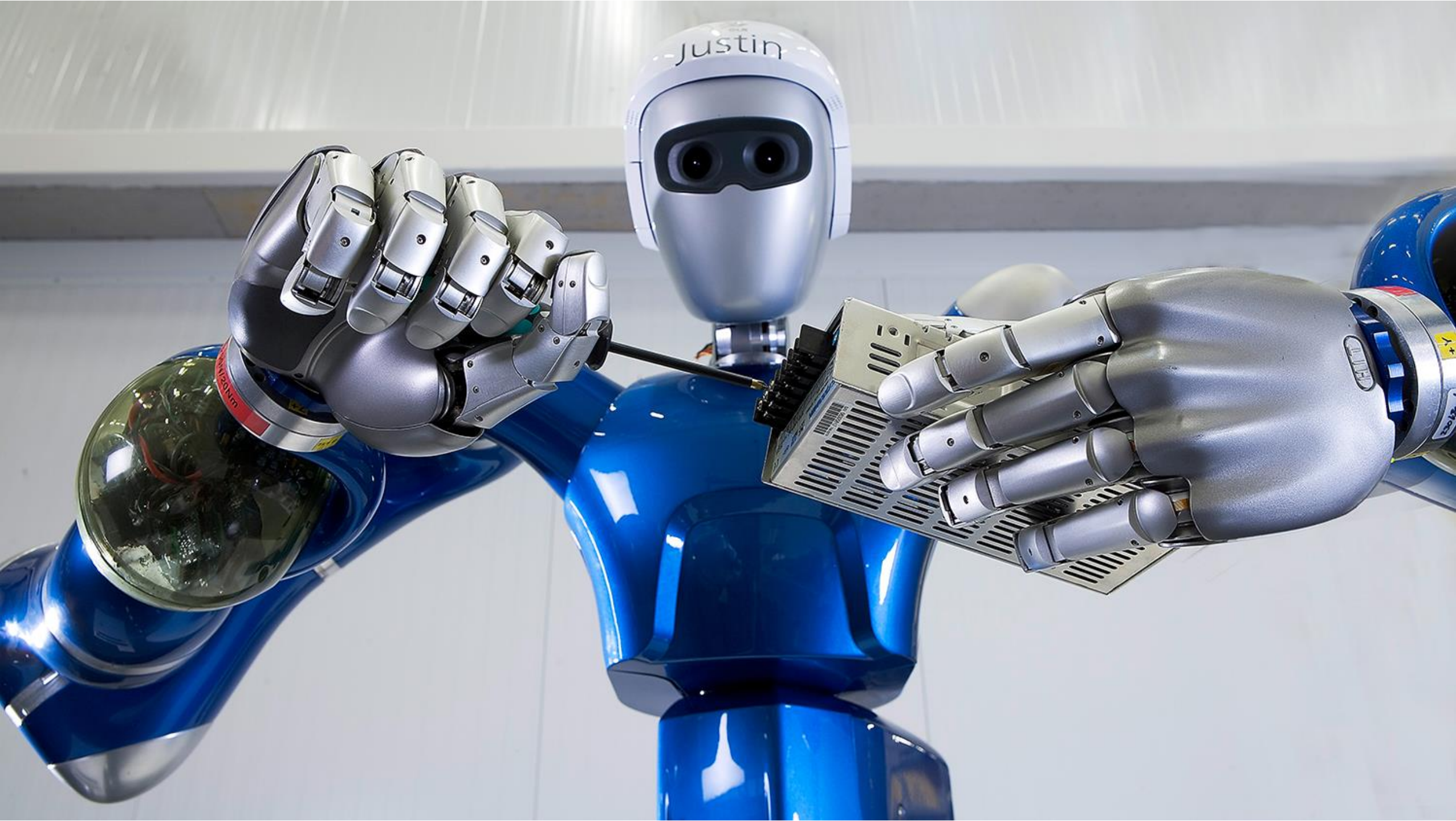
MATLAB EXPO 2018

Developing Algorithms for Robotics
and Autonomous Systems

Sameer M. Prabhu, Ph. D.
Worldwide Industry Marketing Director







Key Takeaway of this Talk

Success in developing an autonomous robotics system requires:

1. Multi-domain simulation
2. Trusted tools which make complex workflows easy and integrate with other tools
3. Model-based design

Challenges with Autonomous Robotics Systems

Applying Multidomain Expertise

Complexity of Algorithms

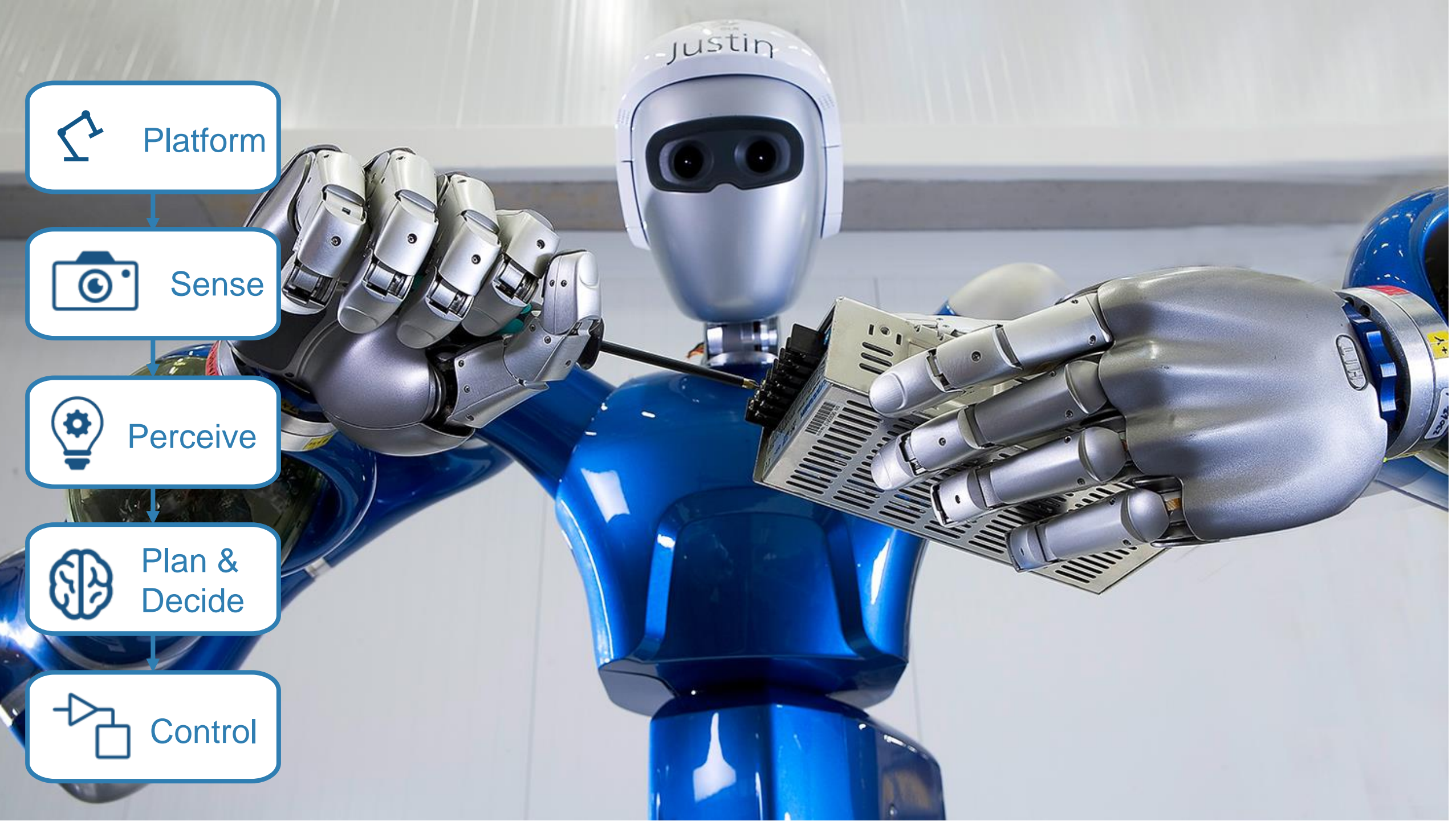
End-to-End workflows

Technical Depth and System Stability

IP Protection

What does success look like?





Platform



Sense



Perceive

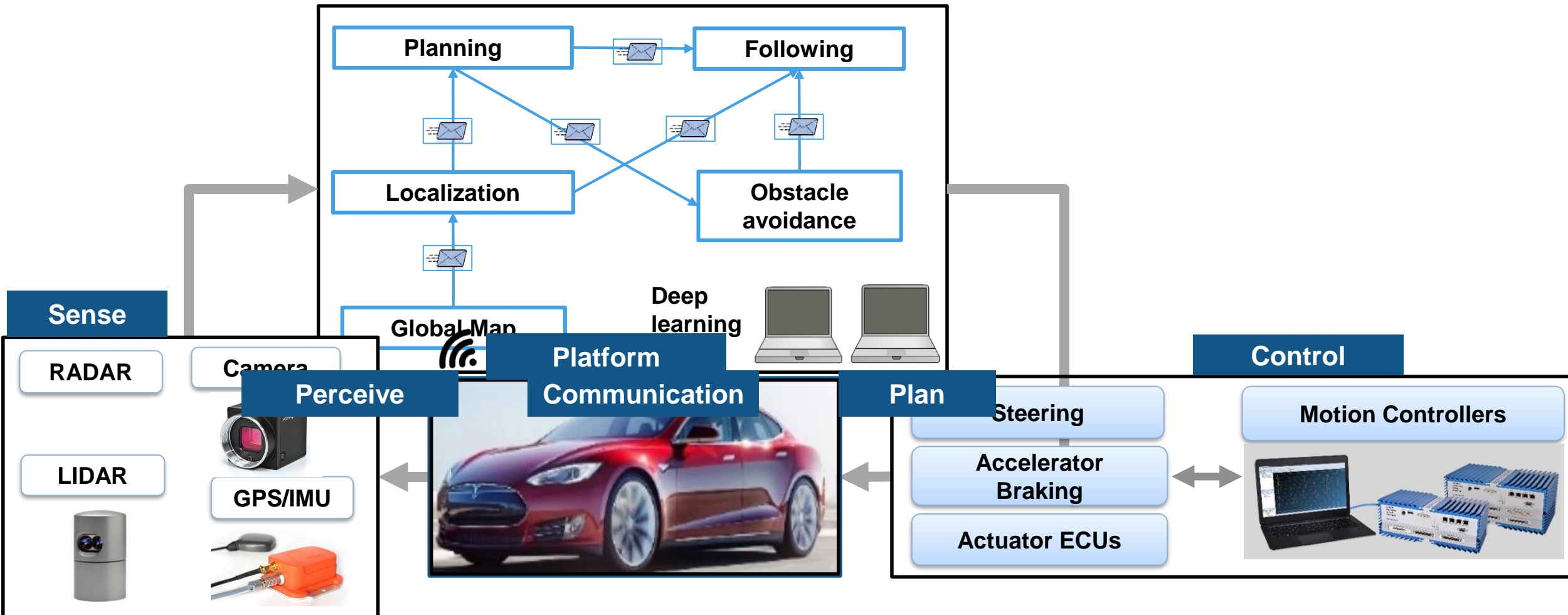


Plan &
Decide

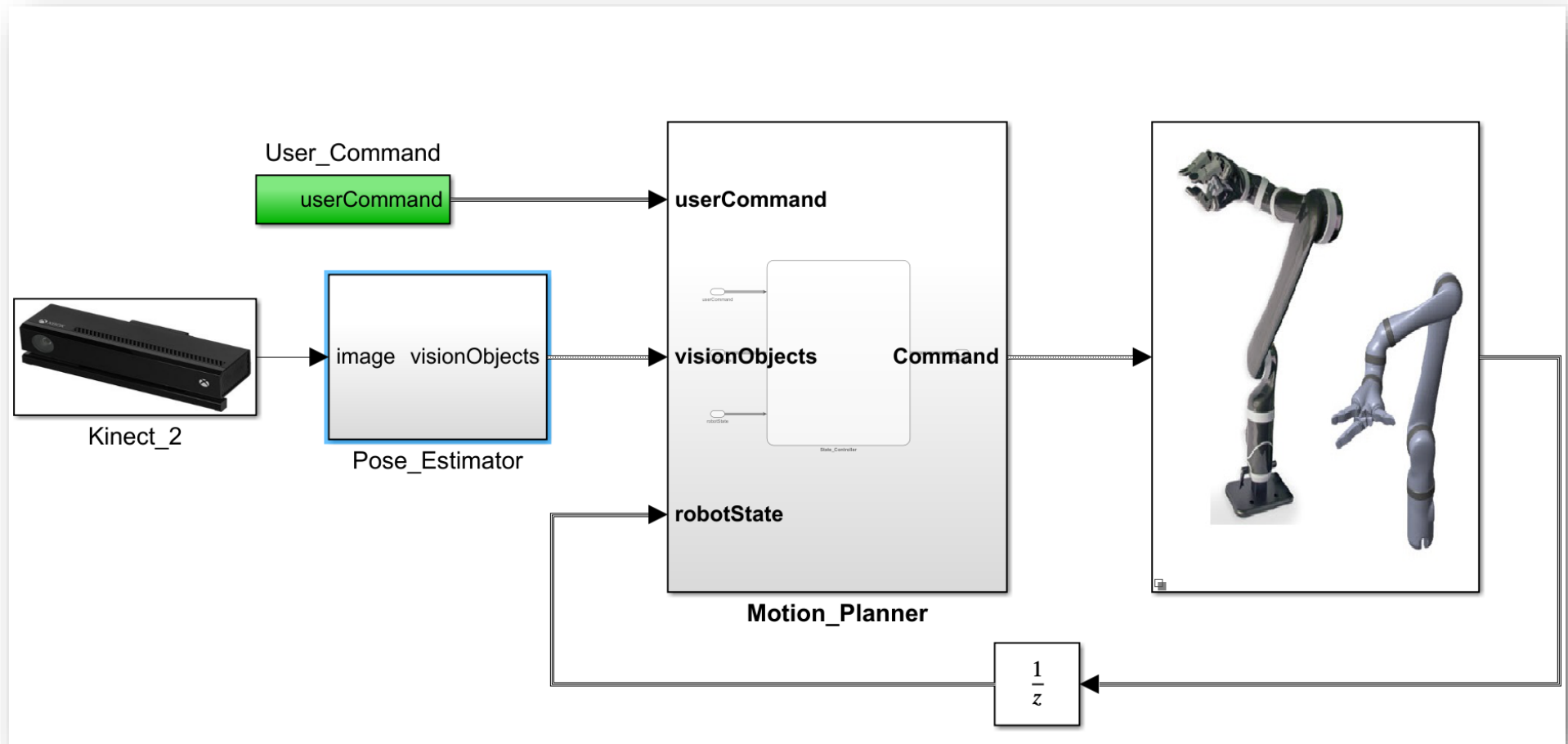
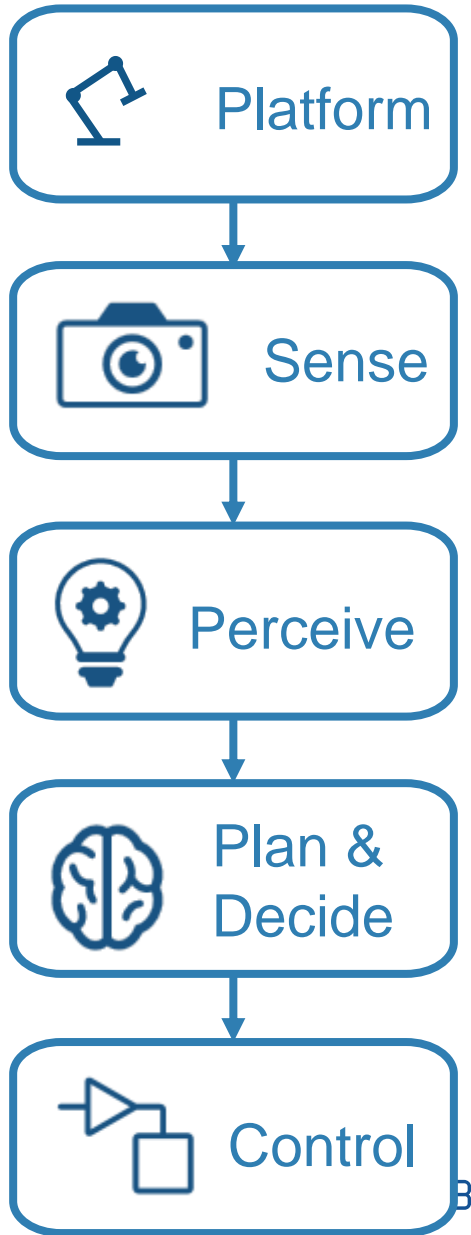


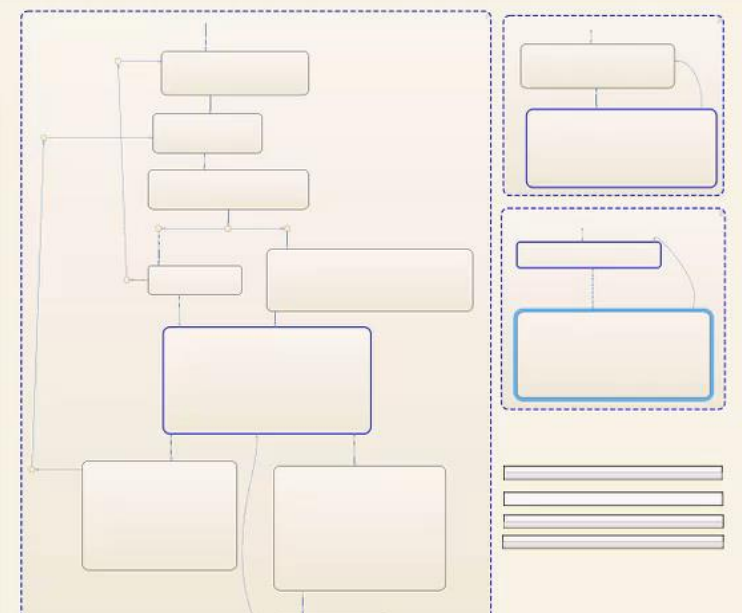
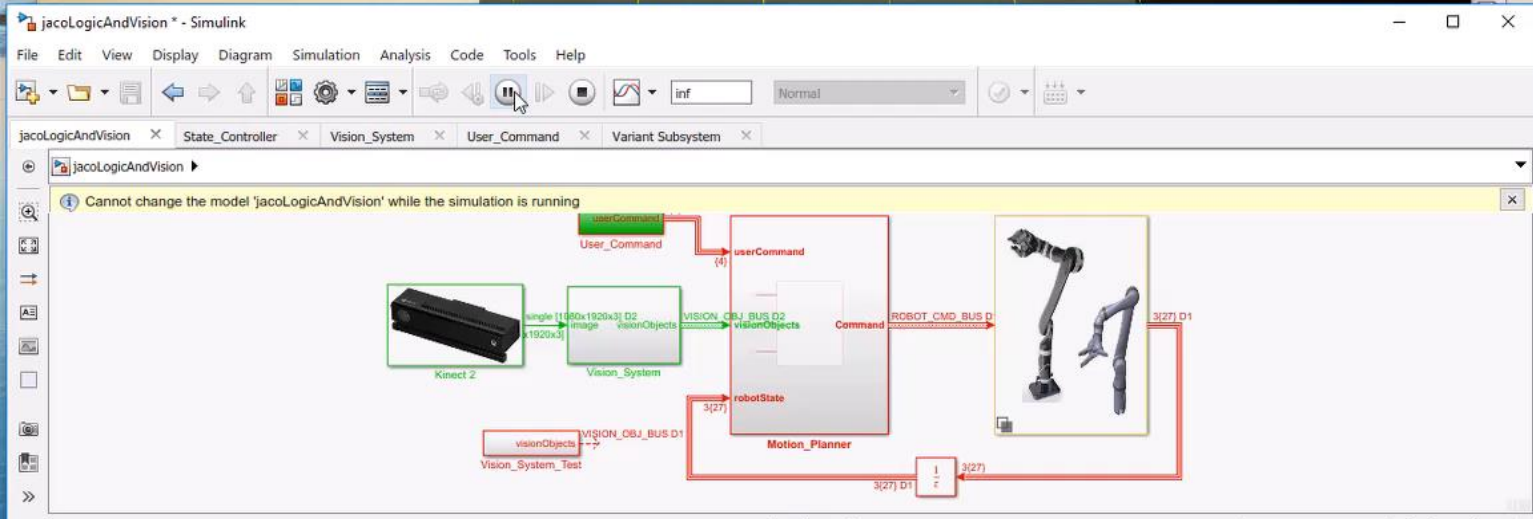
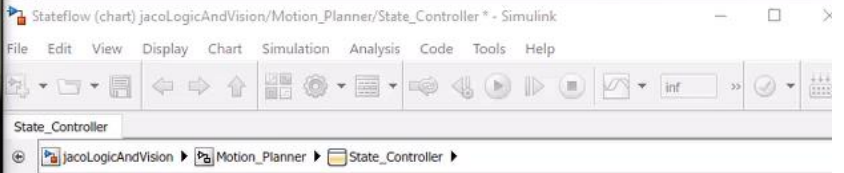
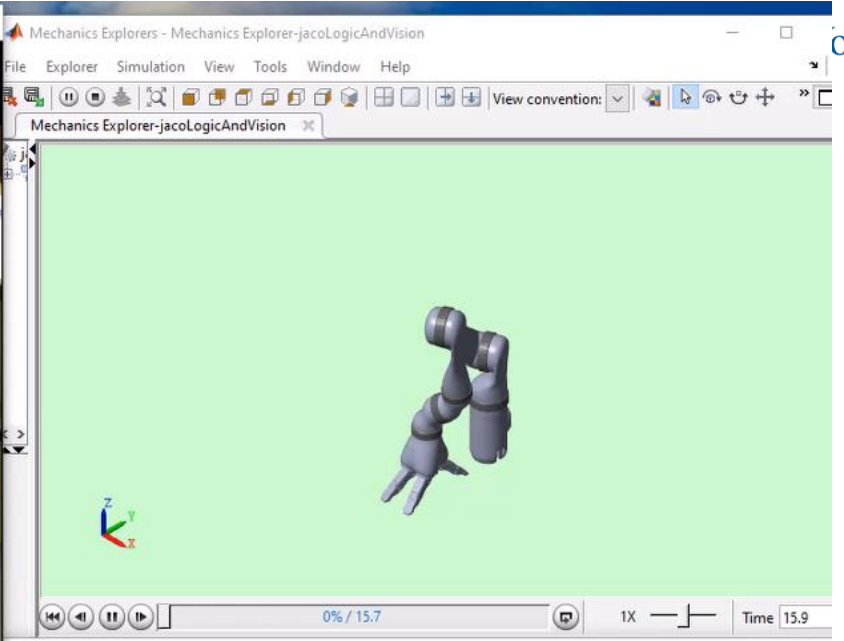
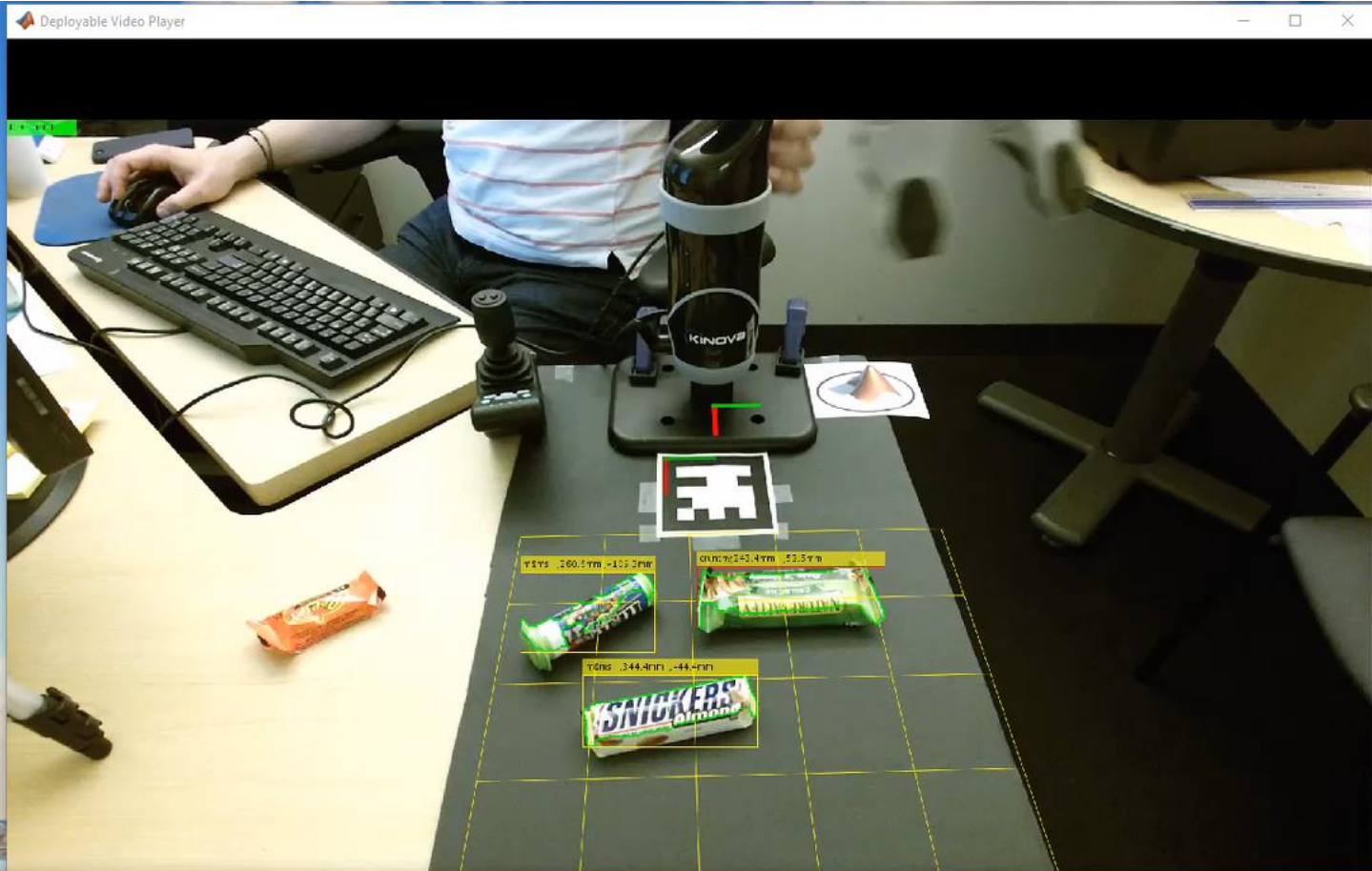
Control

Another Example: Self-Driving Cars

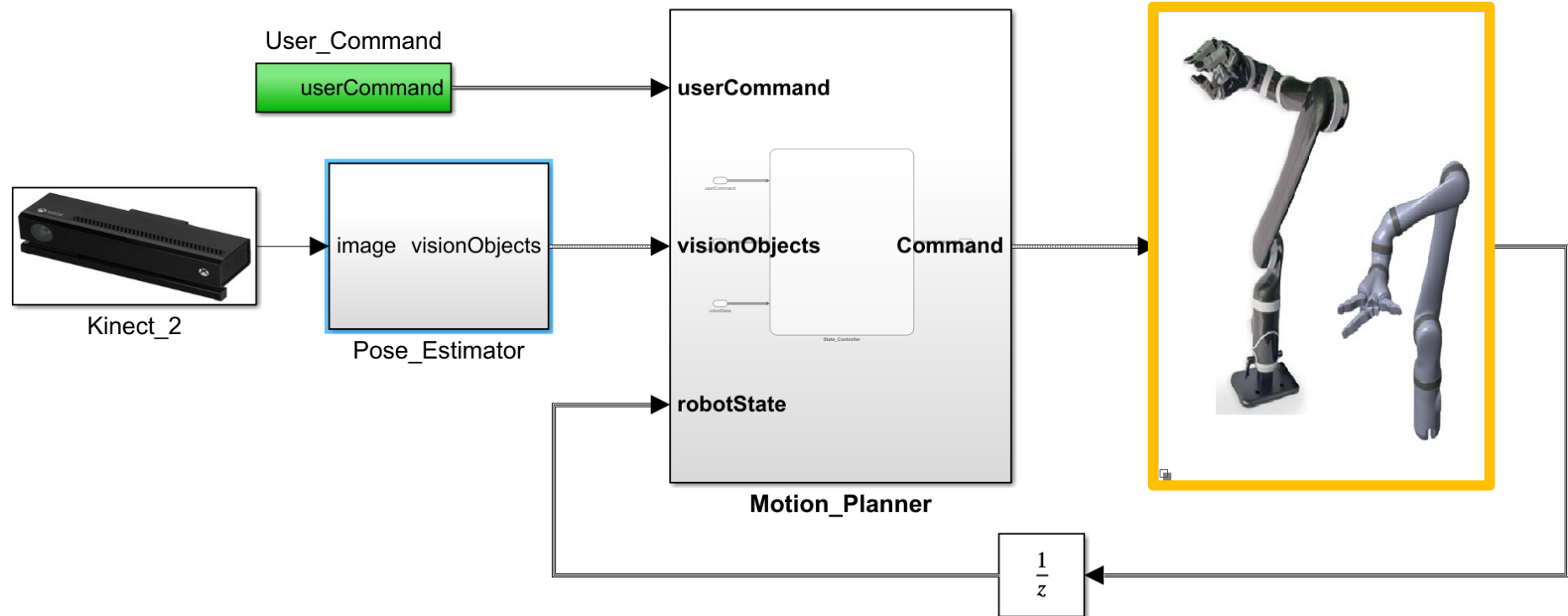
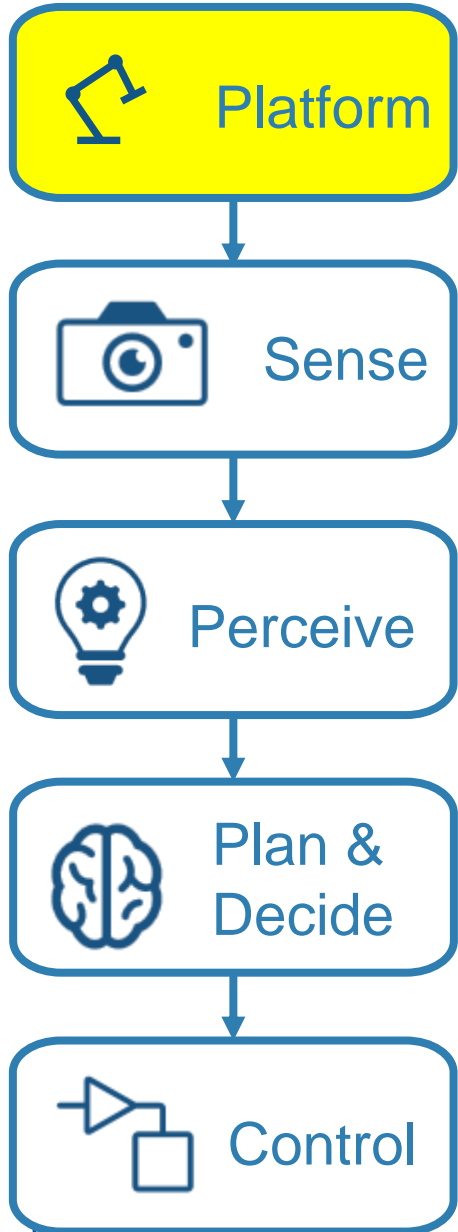


Today: Design Pick and Place Application





Today: Design Pick and Place Application



Platform Design

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

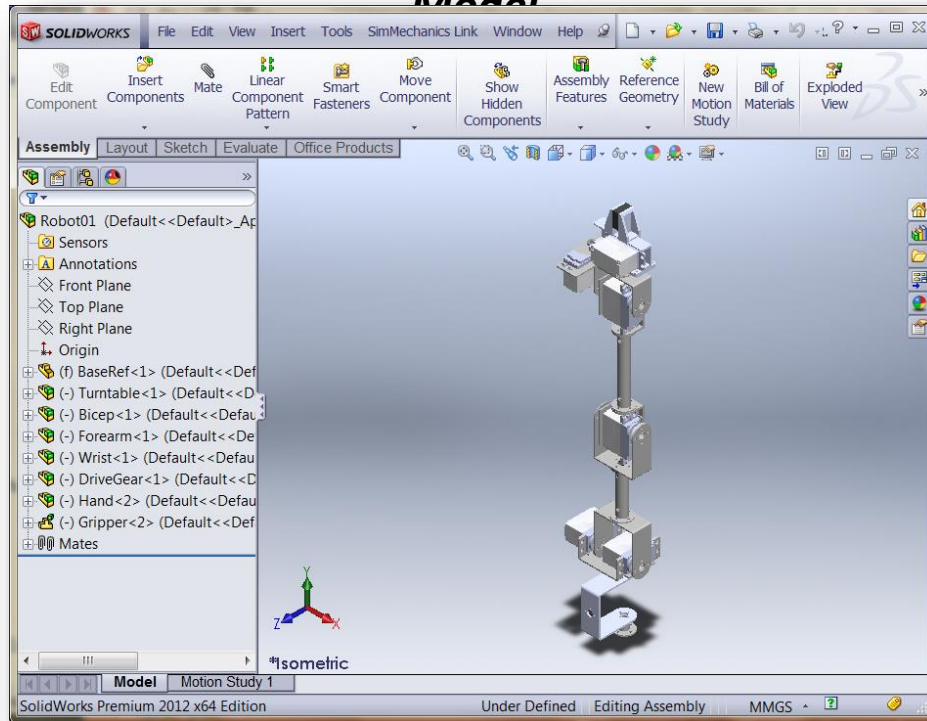
Mechanics

Actuators

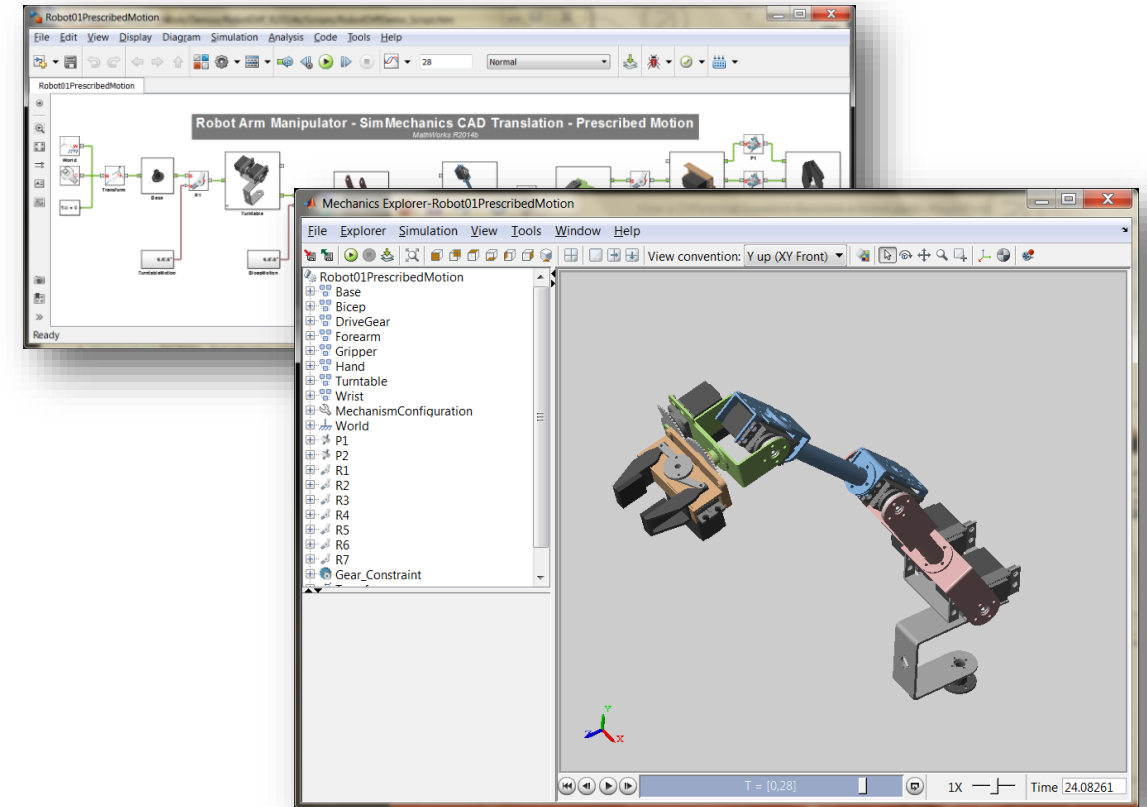
Environment

Mechanics: Import models from common CAD Tools

SolidWorks Model

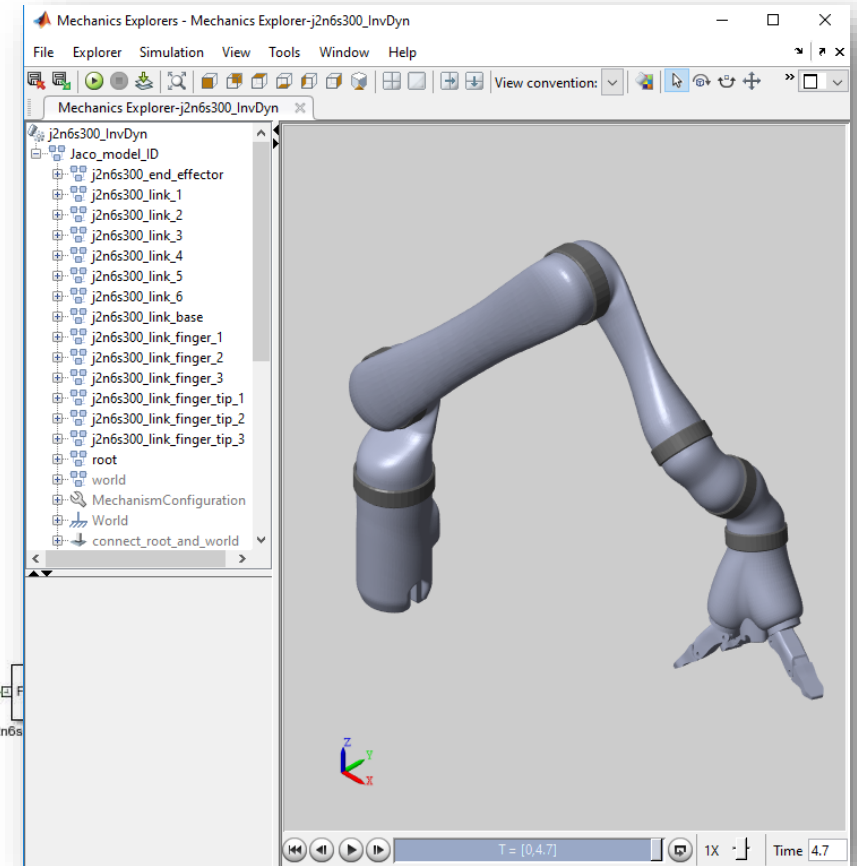
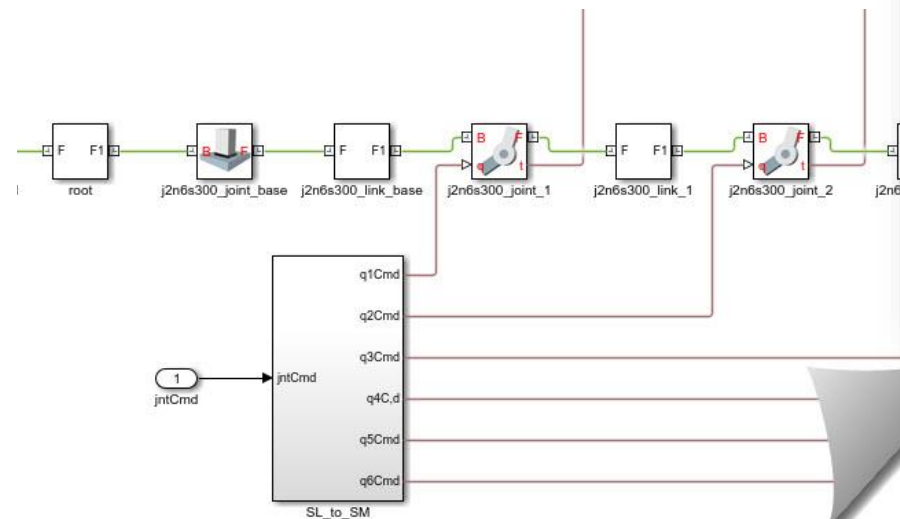
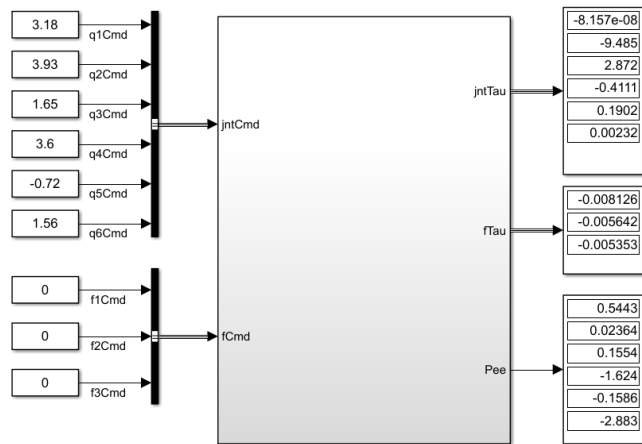


Simscape Multibody Model



Mechanics: One line import from URDF

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

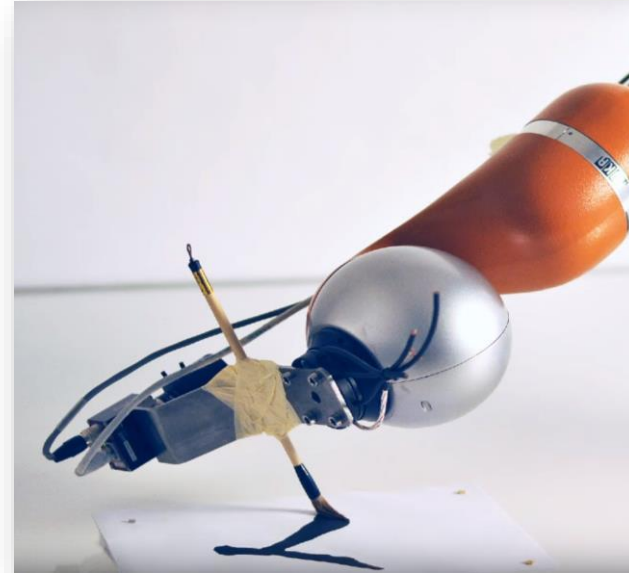
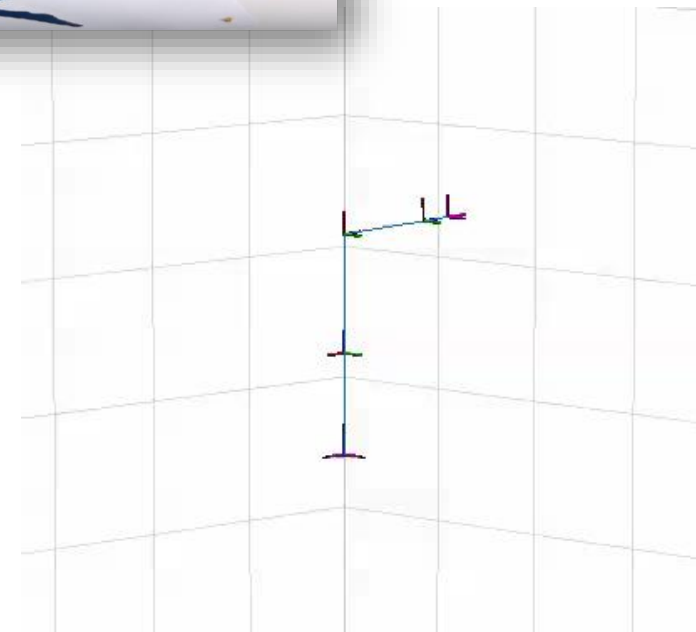


Rigid Body Tree Dynamics

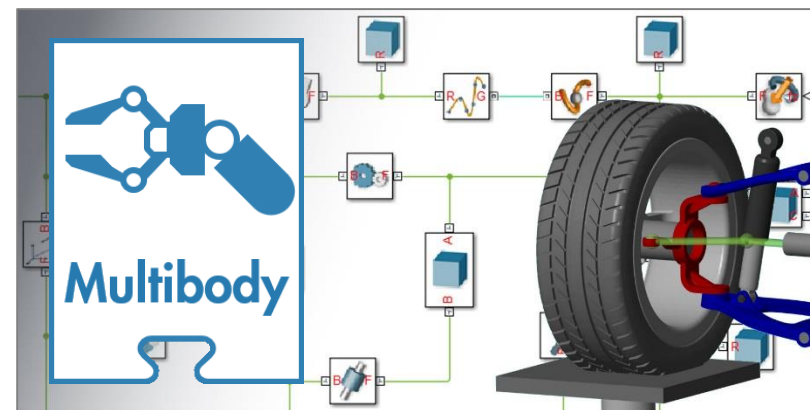
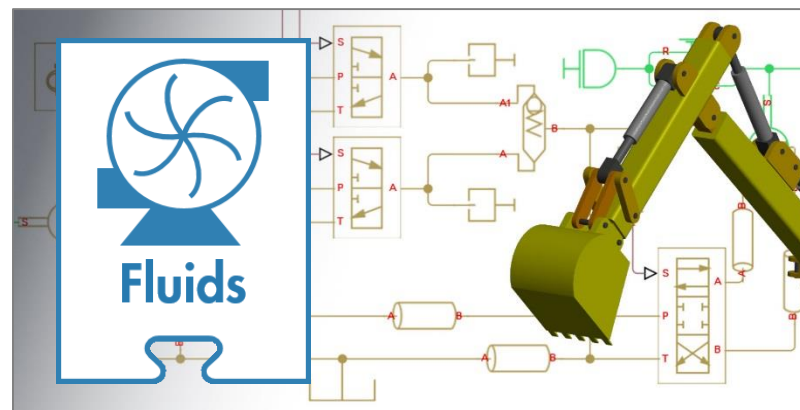
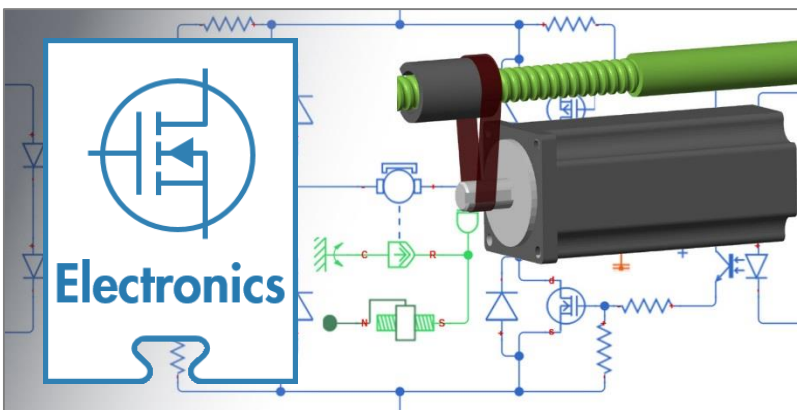
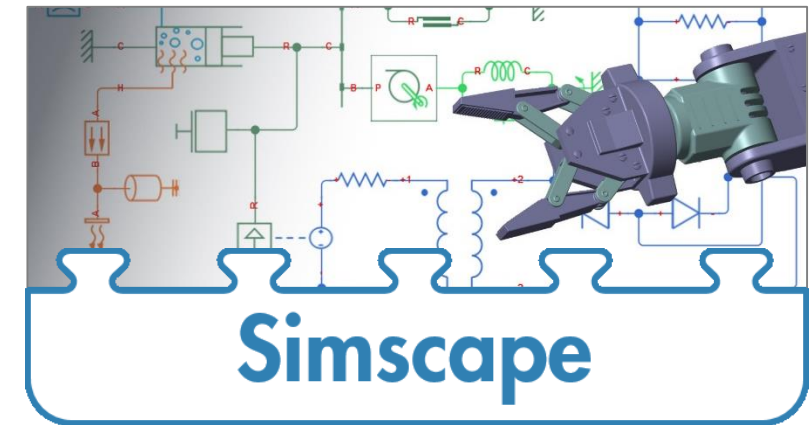
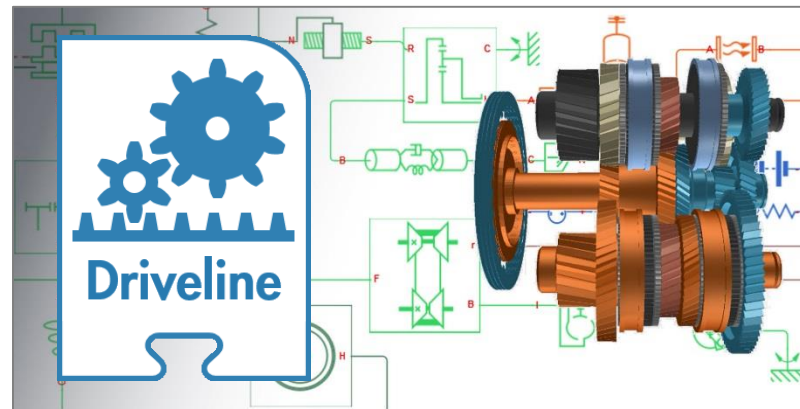
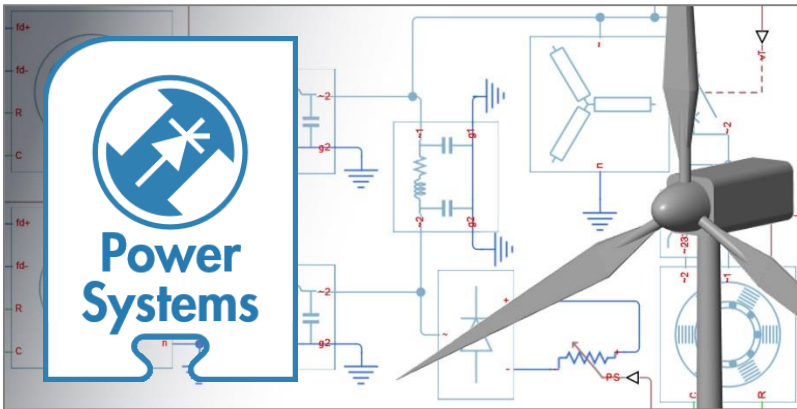
Compute rigid body tree dynamics quantities

- 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);
```

**R2017a**

Actuators: Model other domains



MATLAB EXPO 2018

Environment: Connect to an external robotics simulator

The image displays a Simulink environment connected to a Gazebo 7-VMware Player. The Simulink window, titled 'mainController * - Simulink', shows a control system diagram with the following components and data flow:

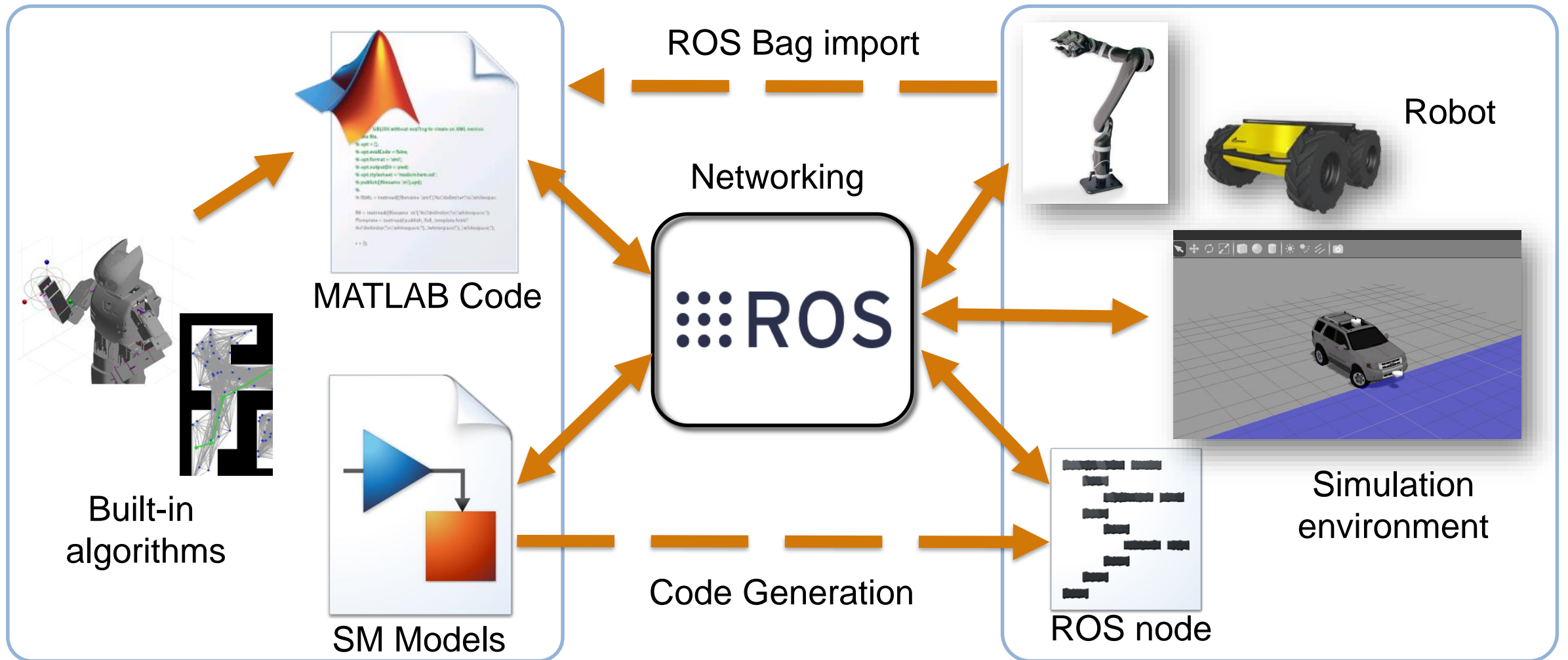
- User Command** (Block 1) and **Object To Grab** (Block 2) provide input to the **Motion Planner**.
- The **Motion Planner** outputs a **request** (containing $\min f(x)$) and a **response** (containing $qAct$).
- The **State Controller** receives **plannerResponse** and **plannerCommand** from the Motion Planner, and outputs **userCommands**.
- The **State Controller** also receives **objectDetected1** from the **Object Detector** and outputs **armCommand**.
- The **State Controller** outputs **goalReached** and **goalSpline**.
- The **Robot ROS interface** block receives **armCommand** and **goalReached** from the State Controller and outputs **qAct** and **goalReached** to the Gazebo environment.
- The **Object Detector** block receives **goalSpline** and outputs **objectDetected1**.
- The **Cloud prediction** block receives **goalSpline** and outputs a visualization of the predicted path.

The Gazebo 7-VMware Player window shows a 3D simulation of an orange robotic arm in a virtual environment. A blue line indicates the target path for the arm. The status bar at the bottom of the Simulink window shows '50% T=23.000' and 'FixedStepDiscrete'.

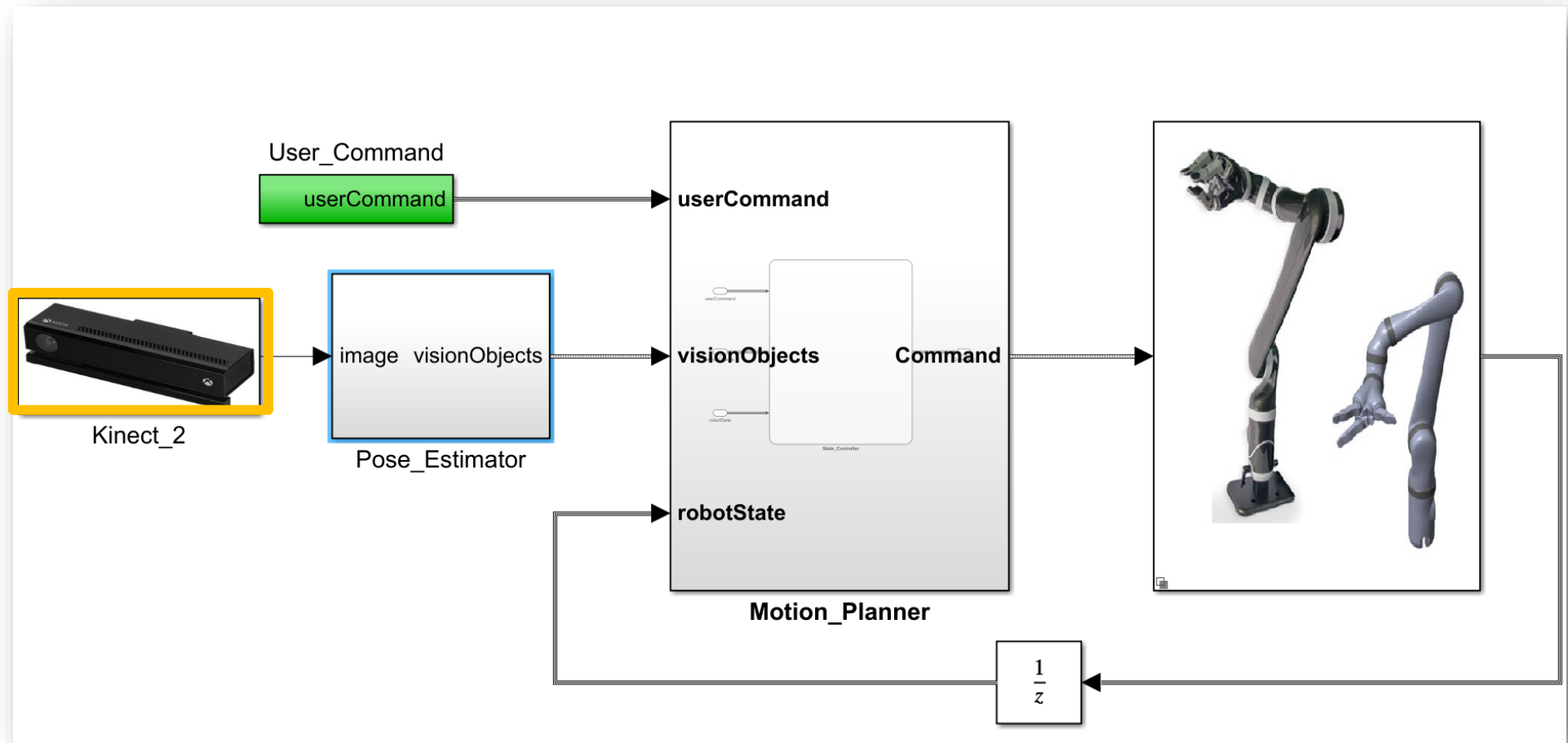
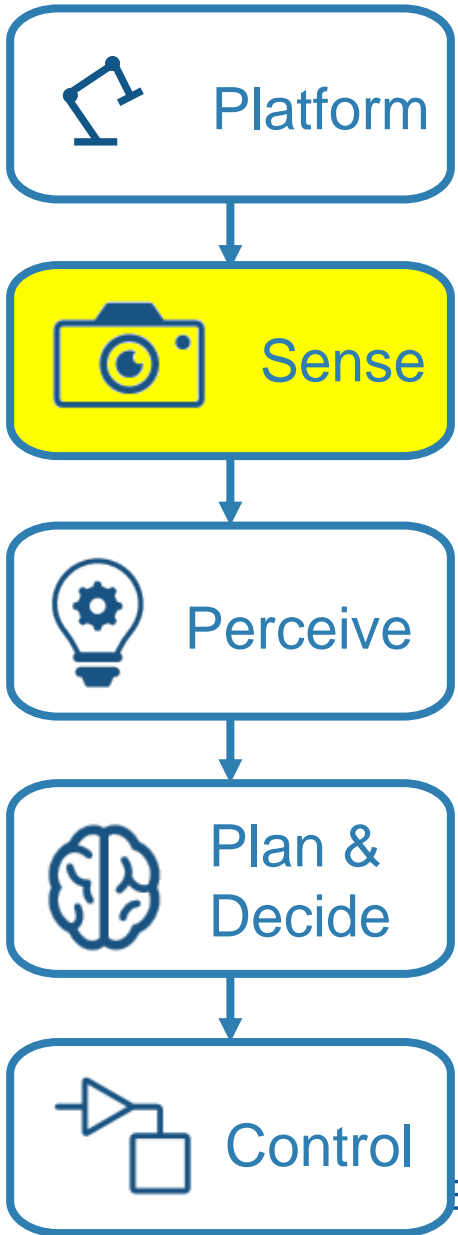
A 'To Video Display1' window is overlaid on the Simulink interface, showing a table of data:

| # | 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

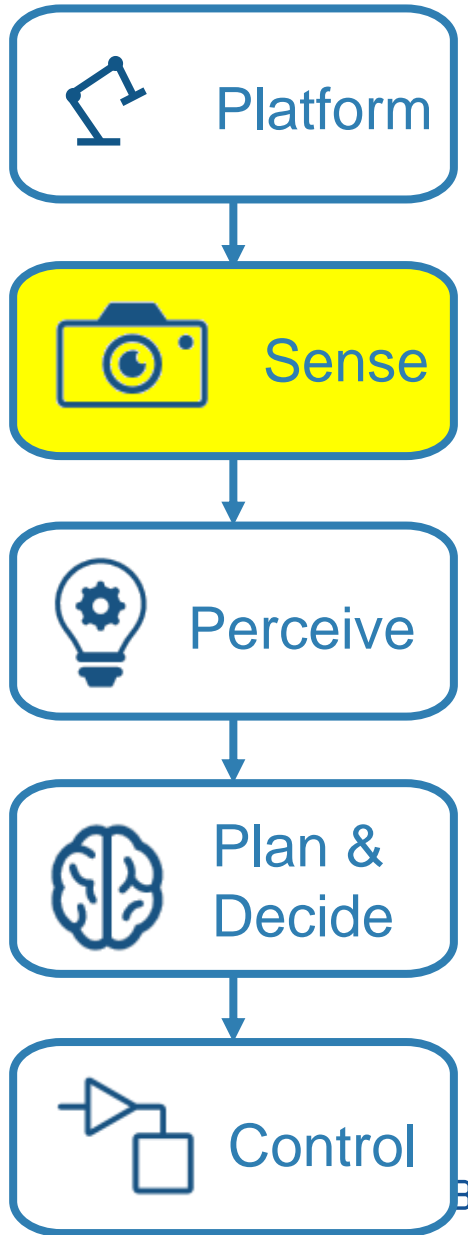


Design Pick and Place Application

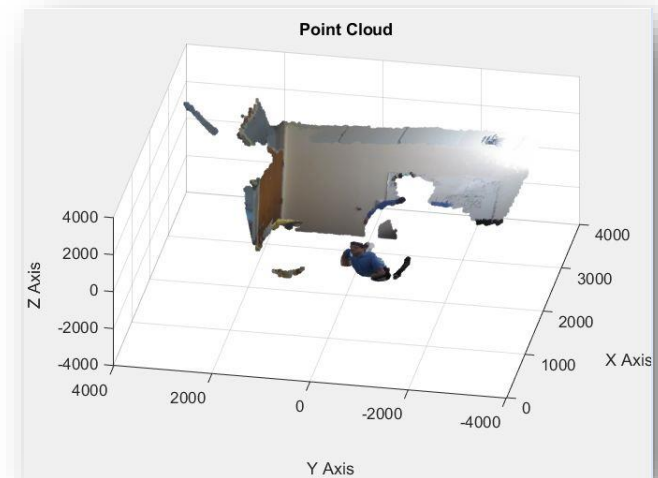


Demo

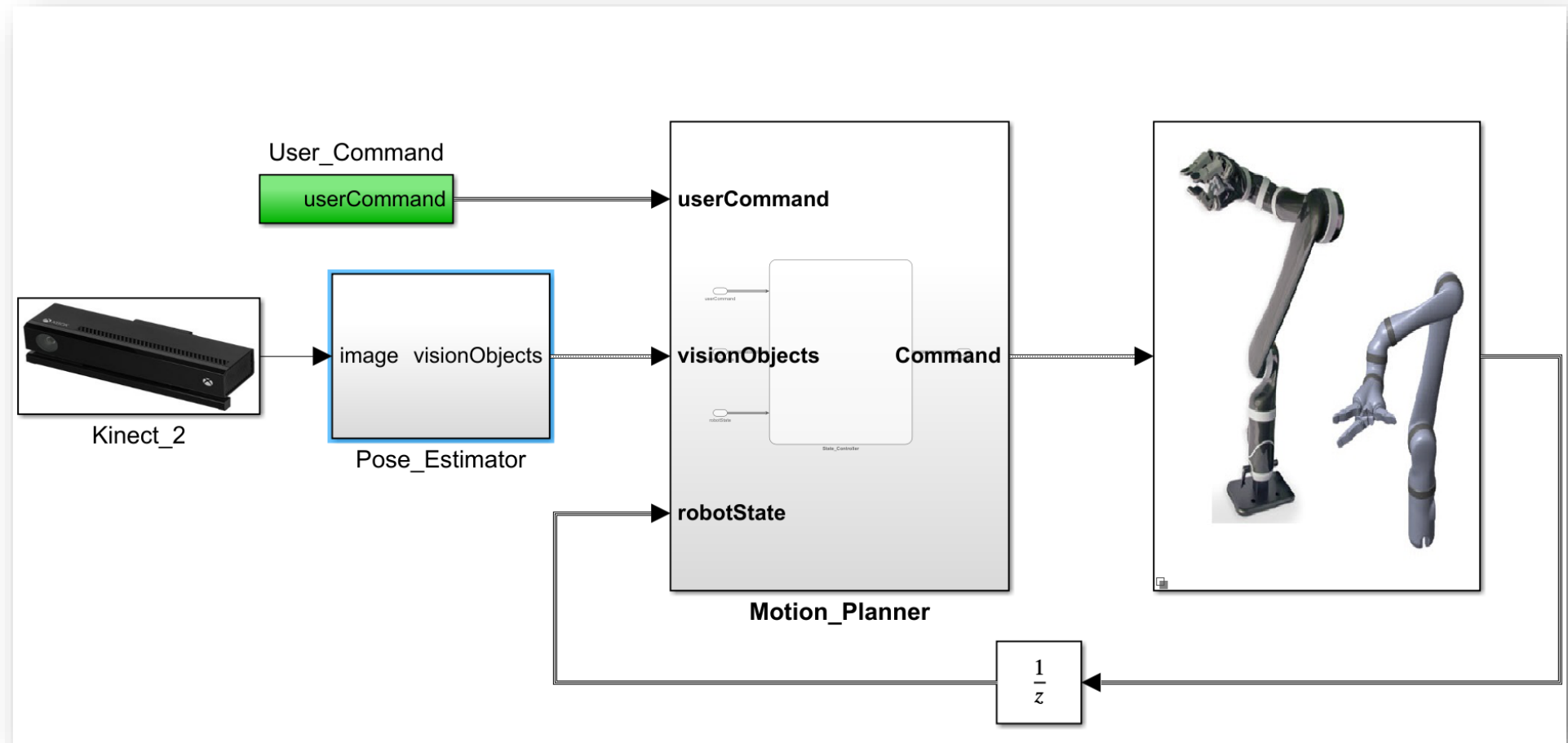
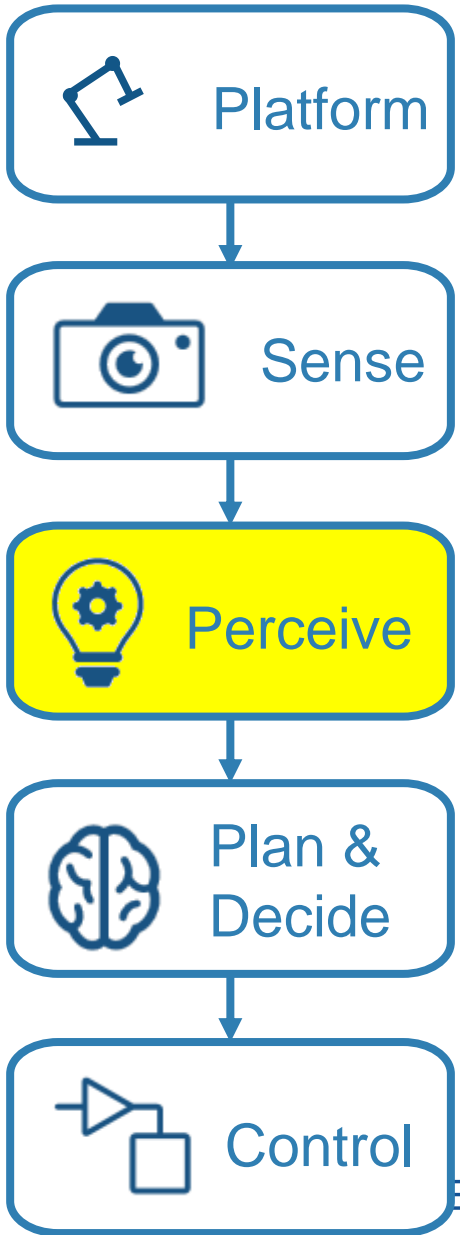
Design Pick and Place Application



- **Support for Common Sensors**
- **Image analysis**
- **Apps**
- **Image enhancement**
- **Visualizing Point Clouds**

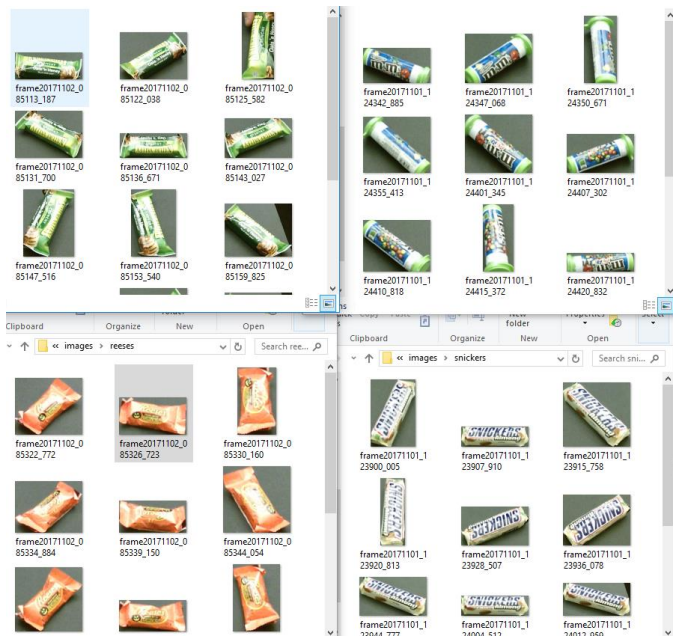


Today: Design Pick and Place Application



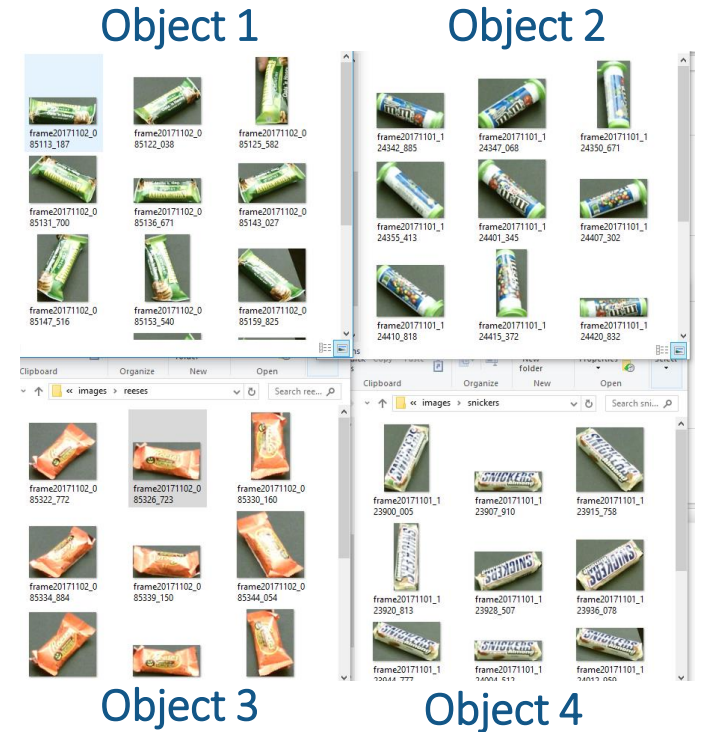
Object Classifier and Pose Estimator

Images



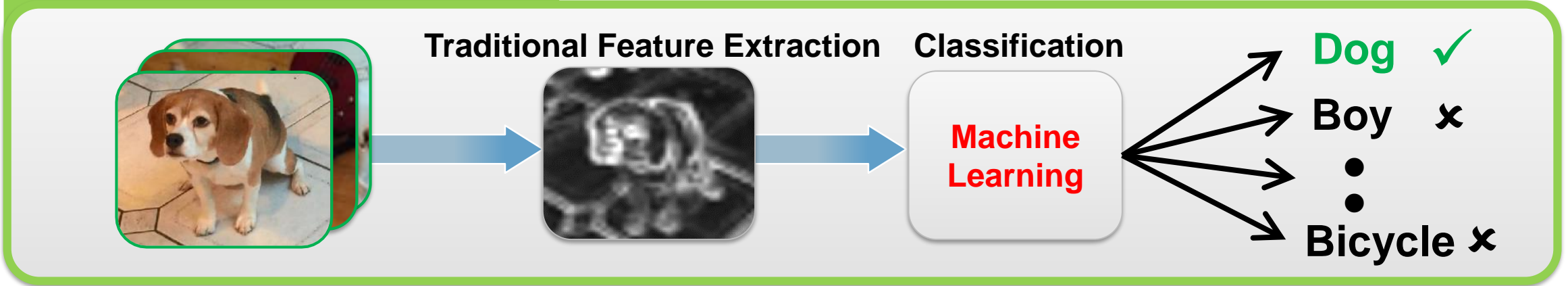
Pose Estimator

Labels and Poses

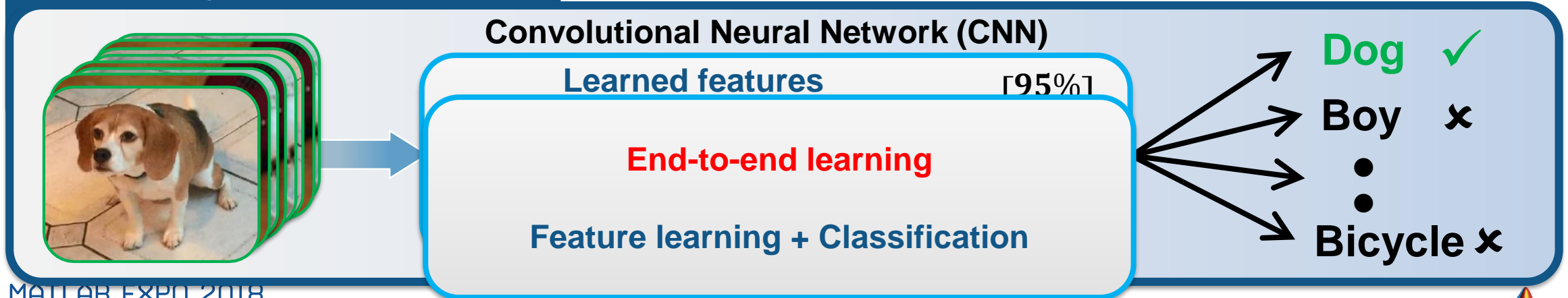


MATLAB makes machine learning easy and accessible

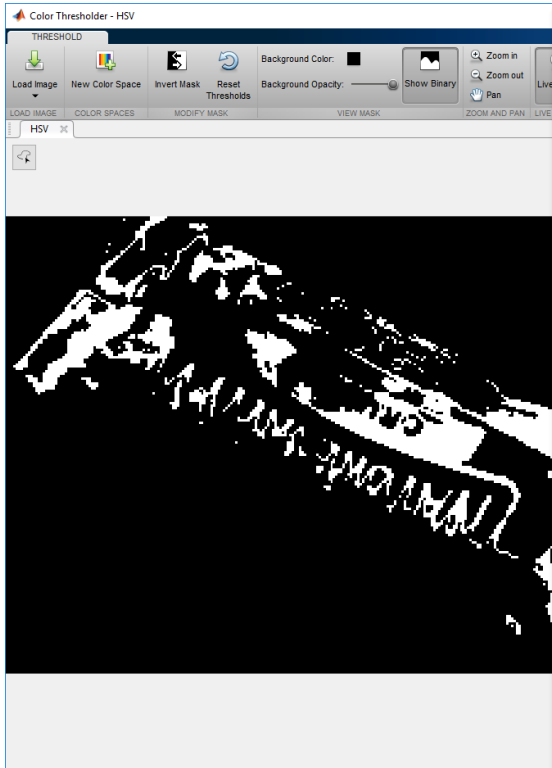
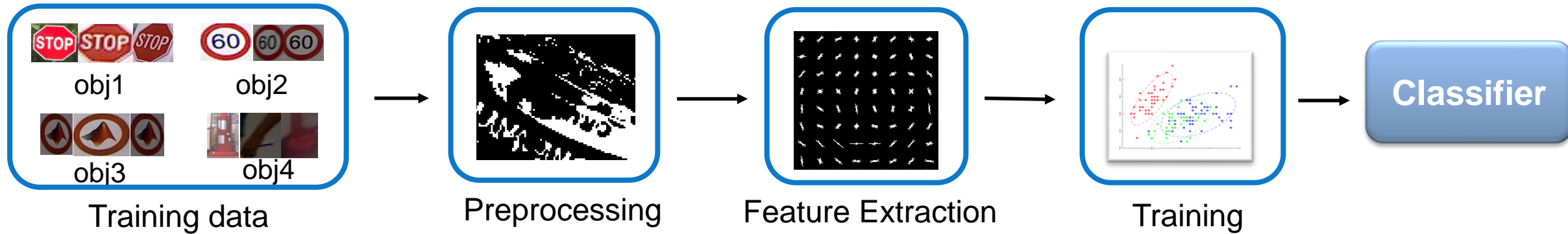
Traditional Machine Learning approach



Deep Learning approach



Complex workflows made easy with MATLAB



```

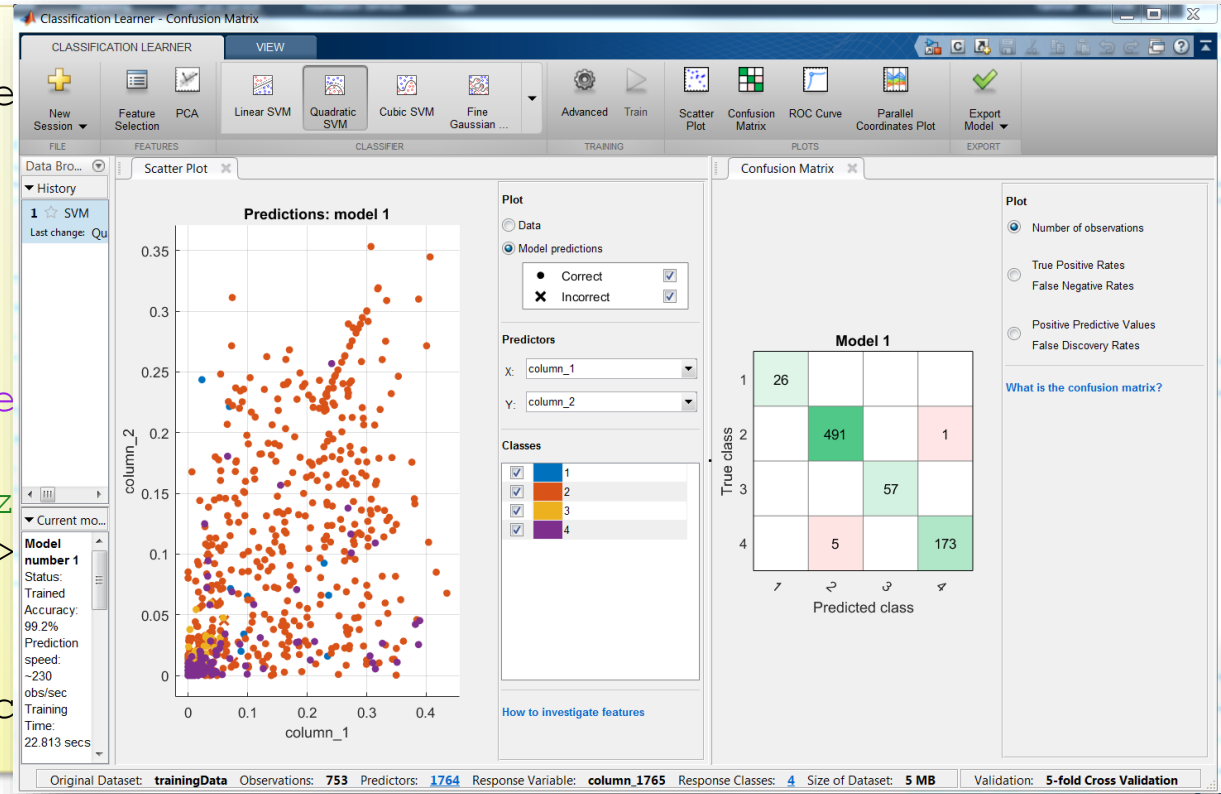
% Detect regions
BW = createMask(videoFrame

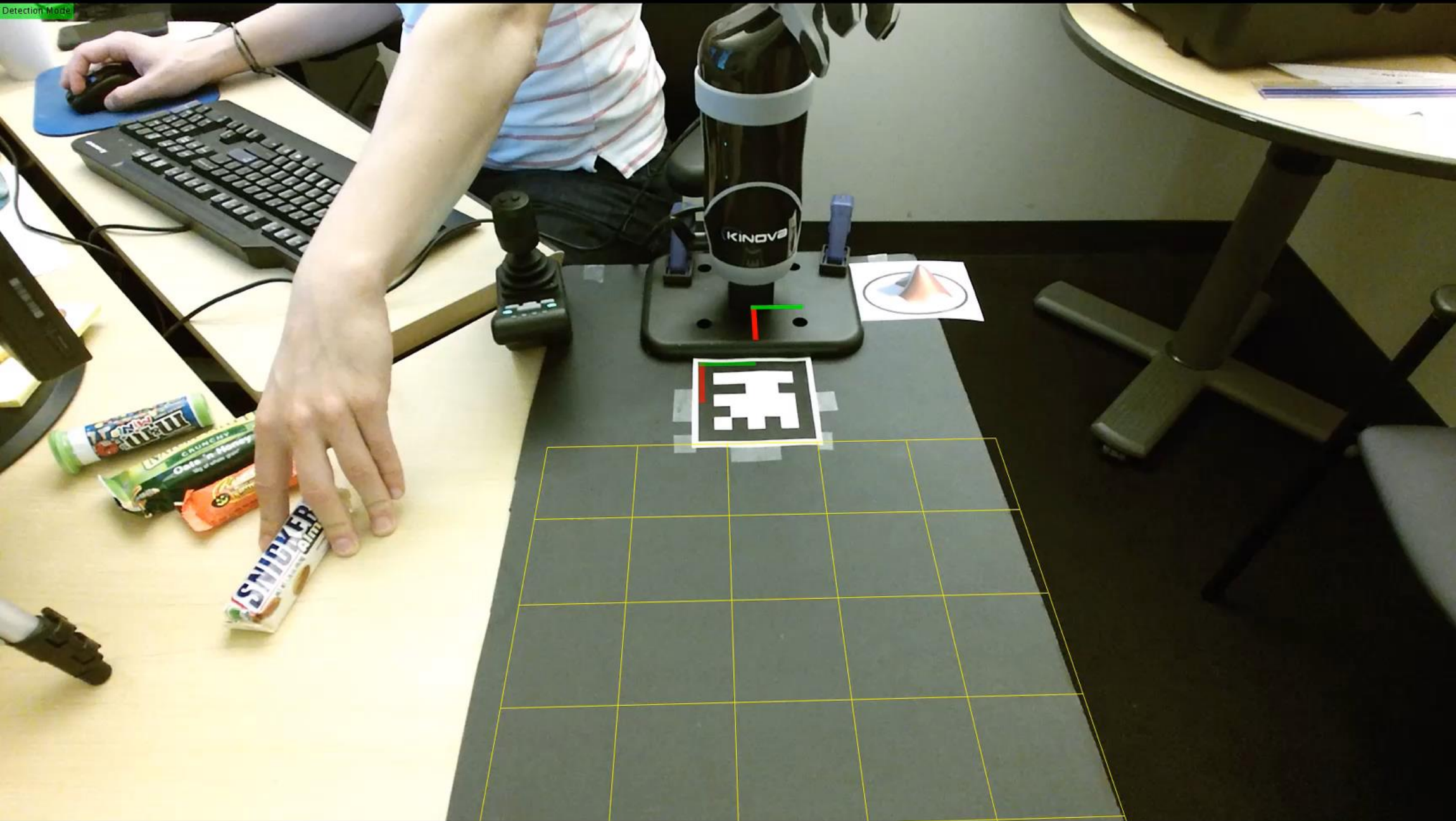
% Fill image regions
BW = imfill(BW, 'holes');

% Get bounding boxes
stats = regionprops('table

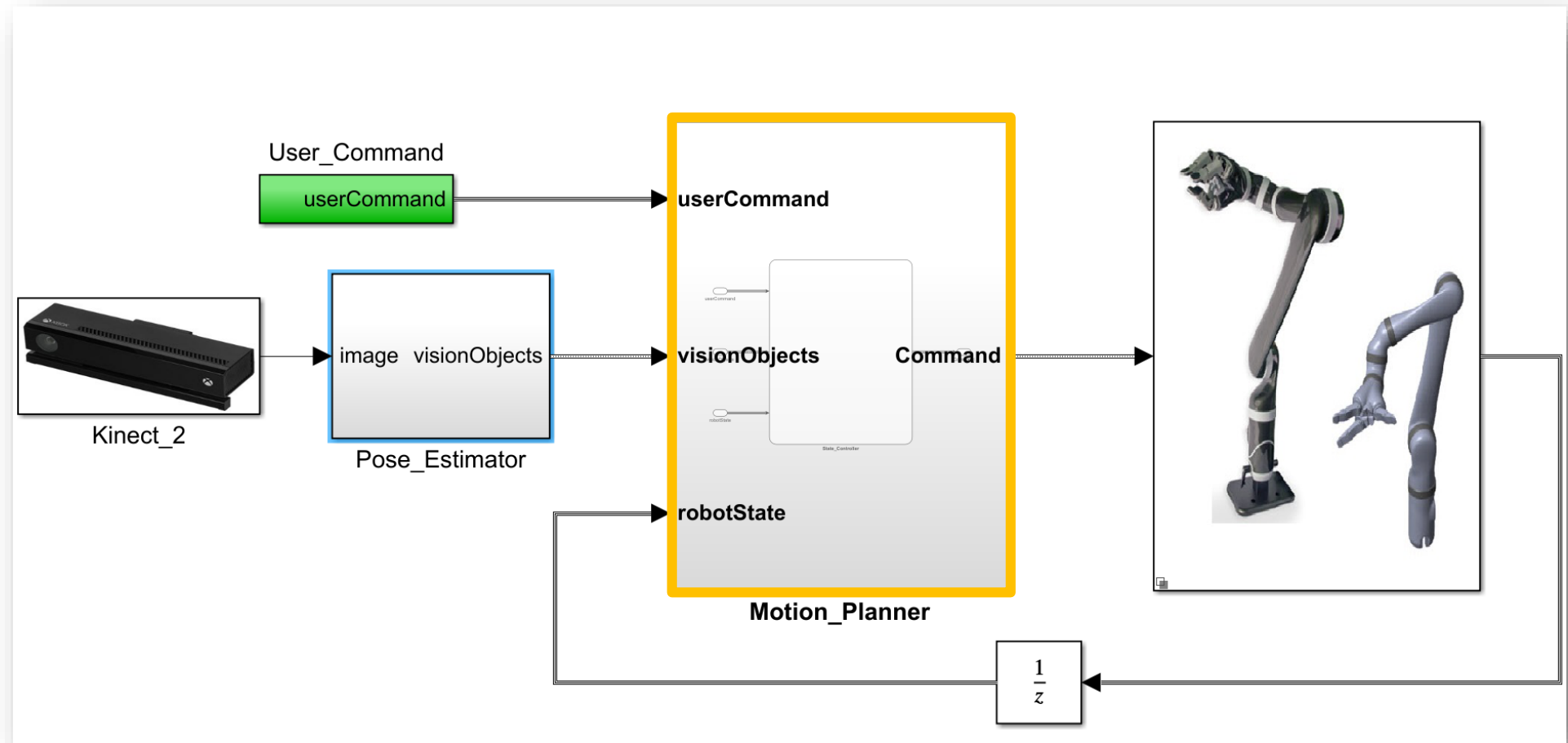
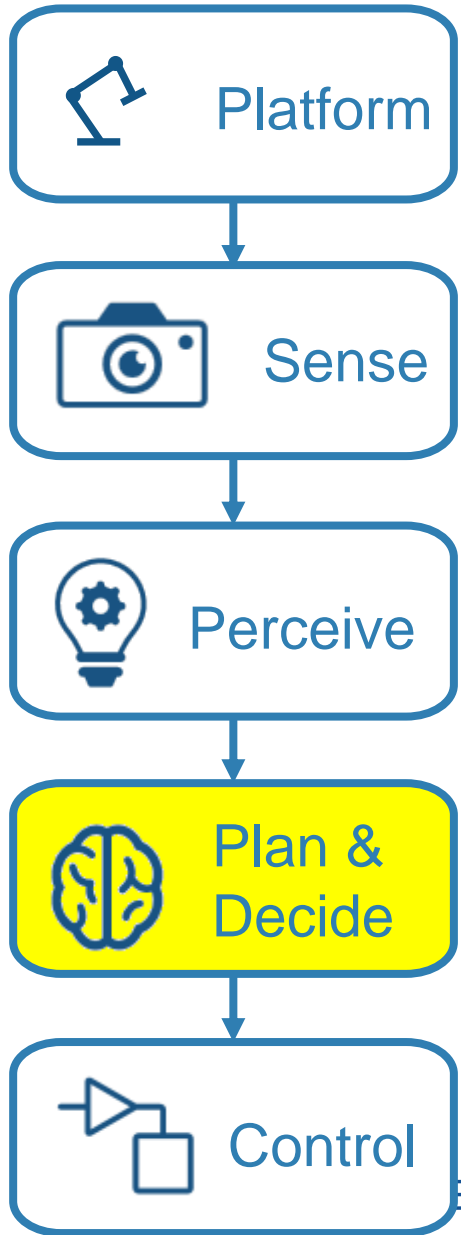
% Filter based on area size
targetIndex = stats.Area >

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





Design Pick and Place Application

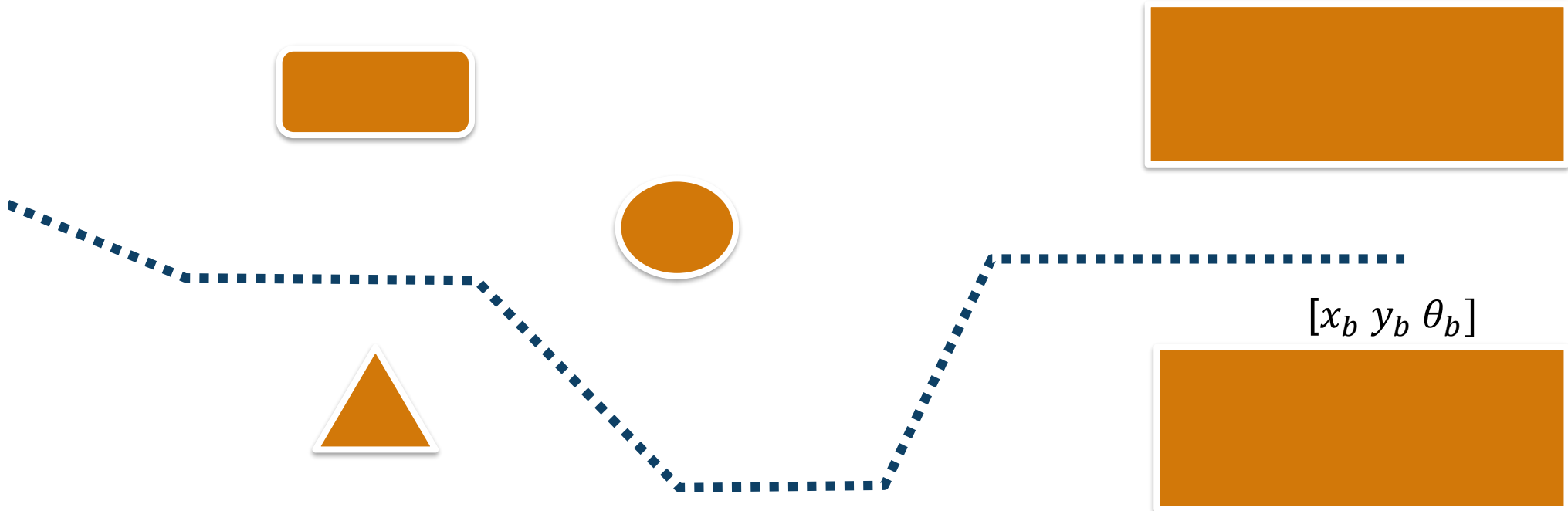


Planning: Find a path

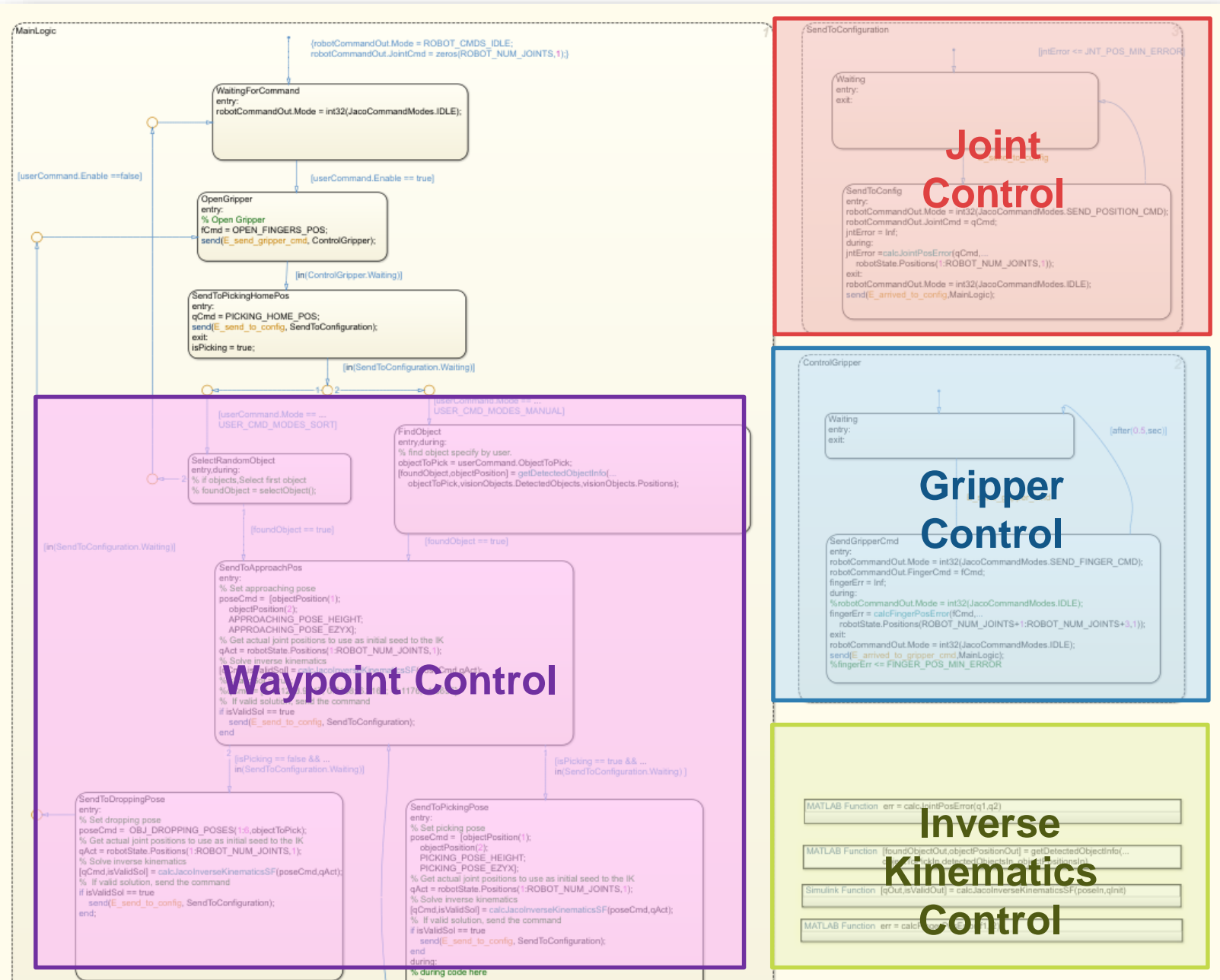
Map
Initial Pose
Final Pose



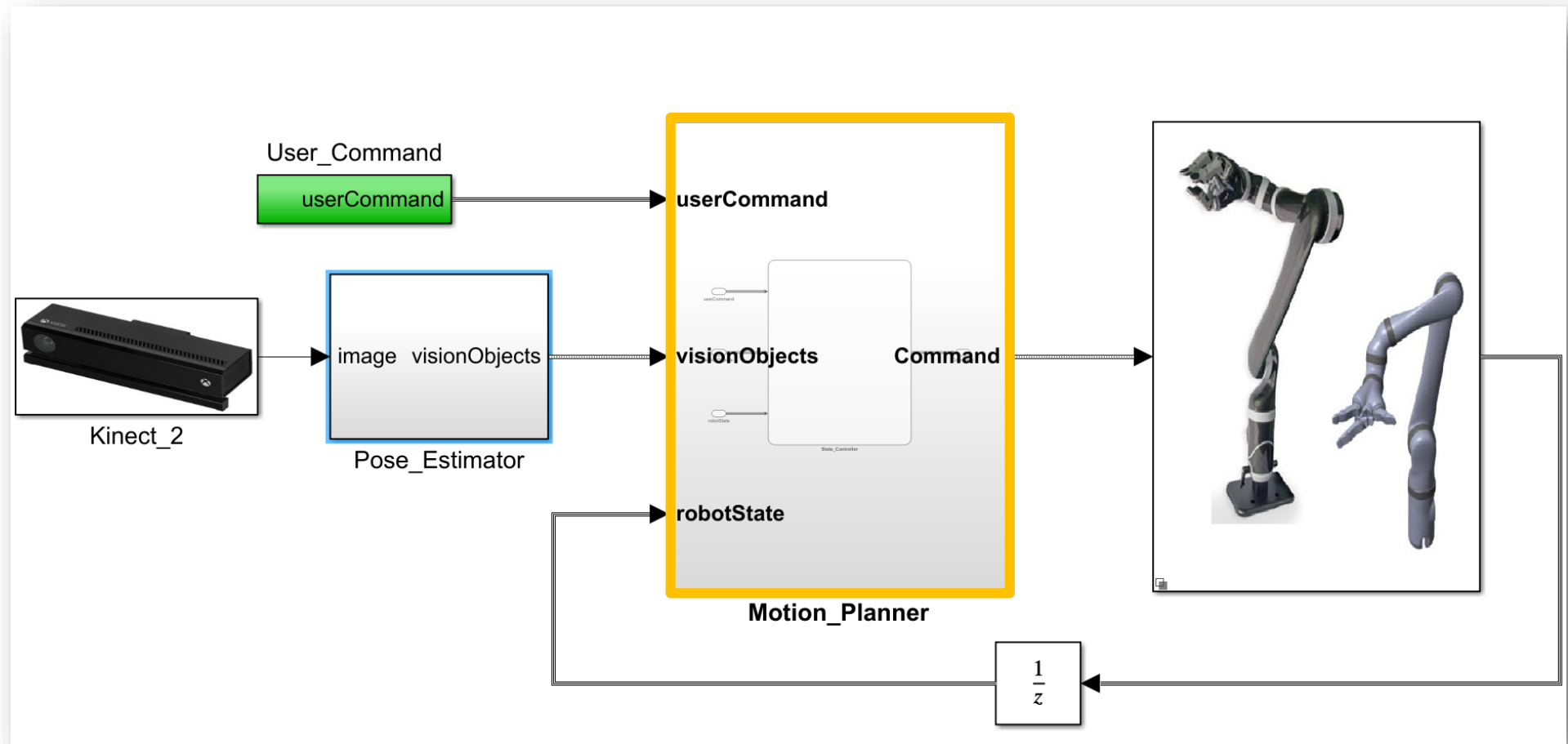
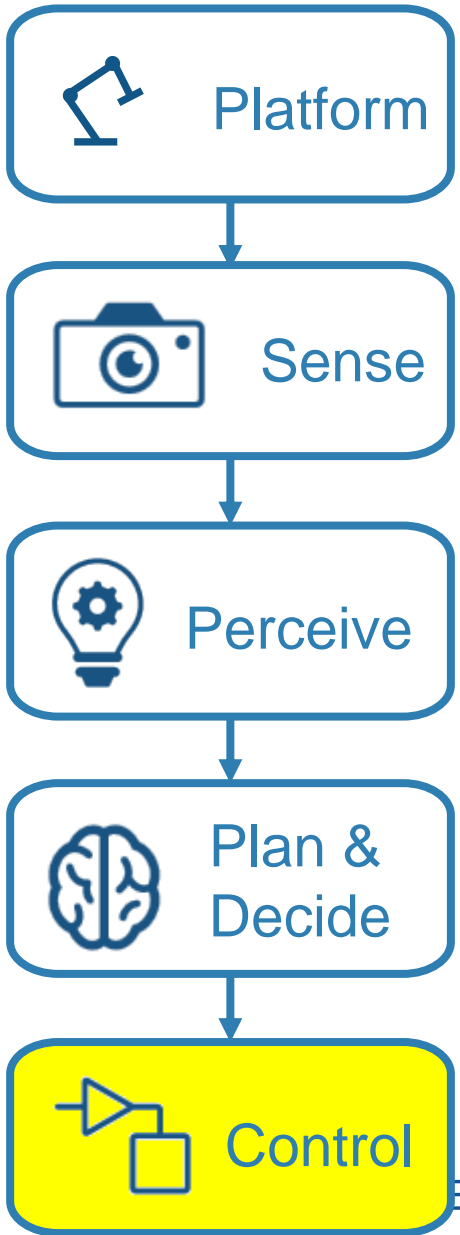
$[x_a \ y_a \ \theta_a]$



Plan with Stateflow

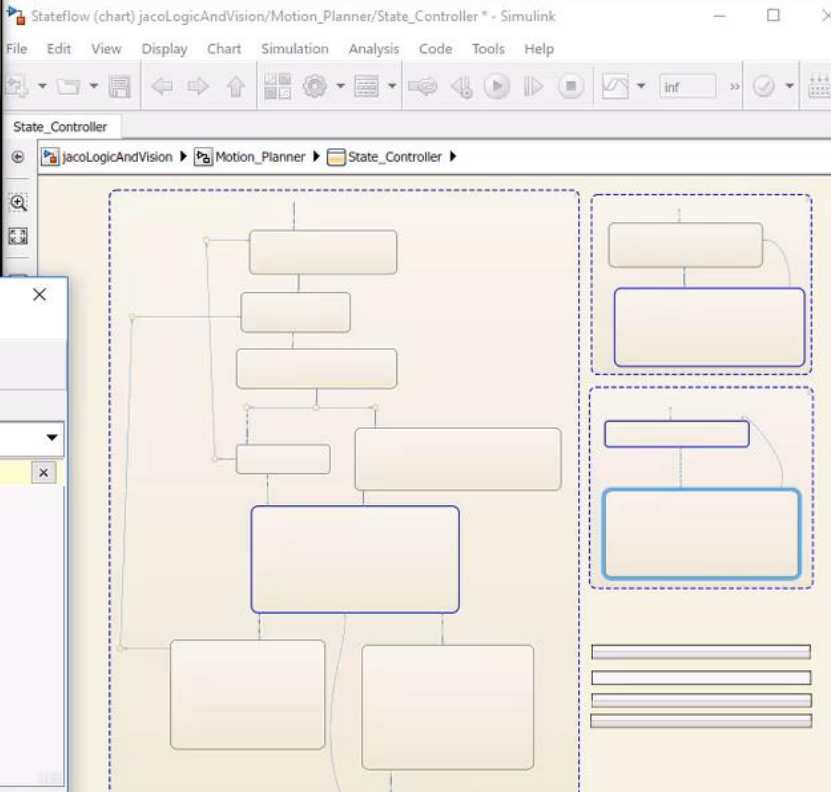
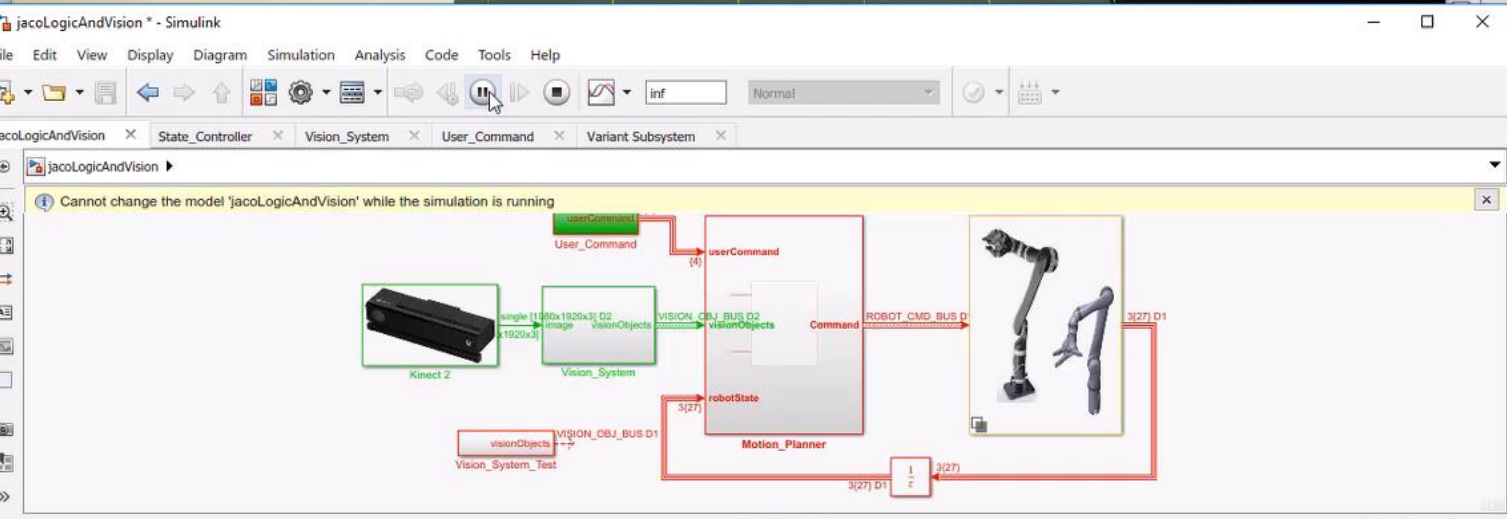
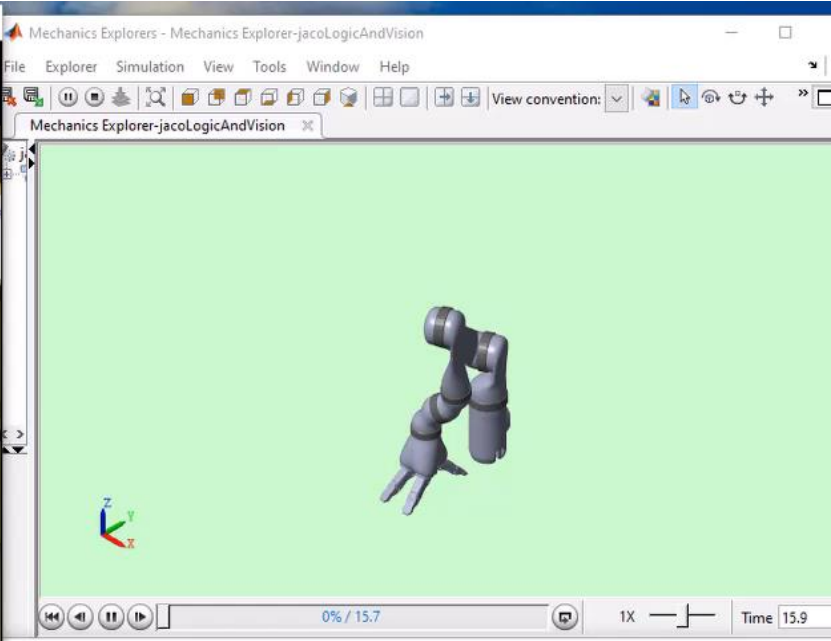
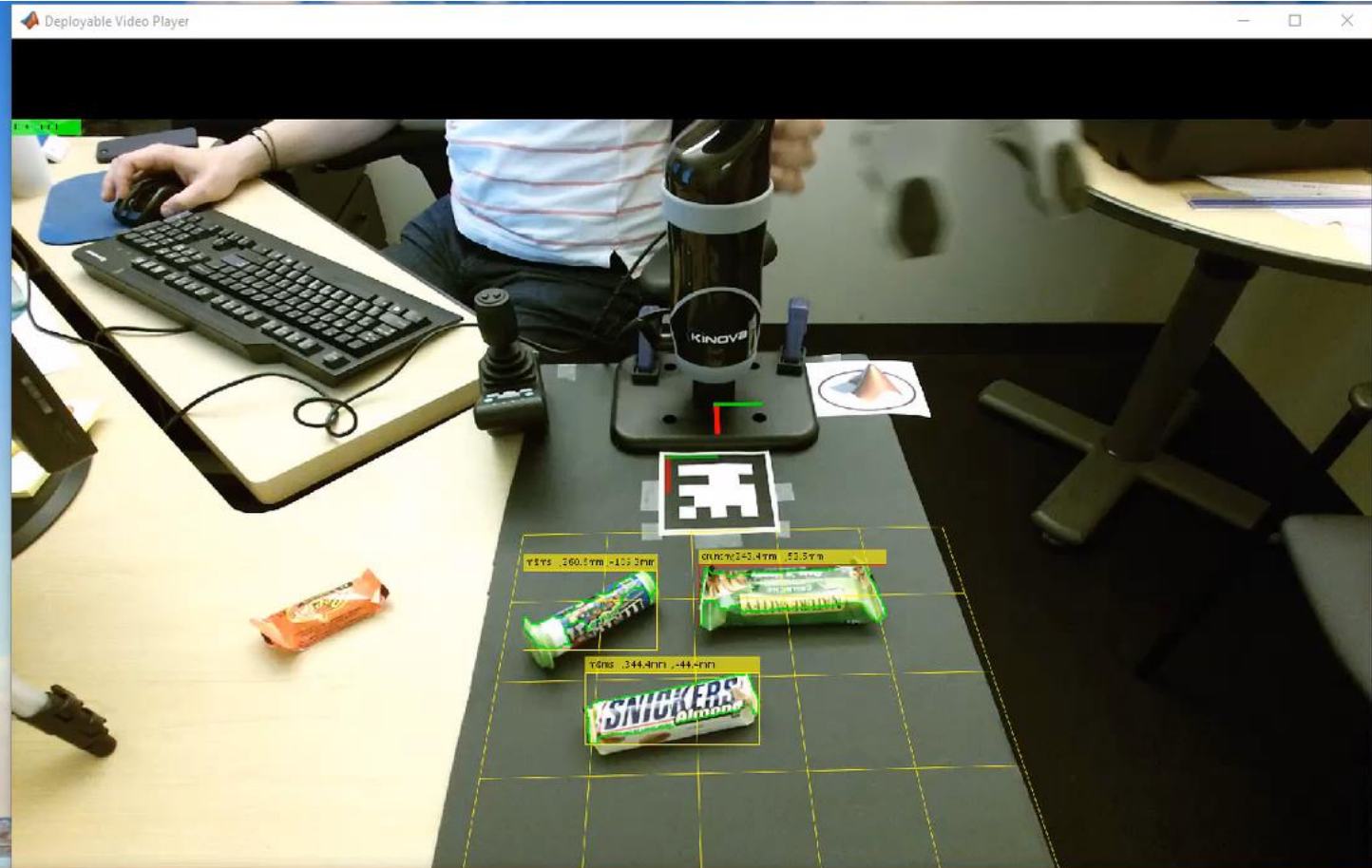


Design Pick and Place Application



Explore Built In Functions: Inverse Kinematics

```
% Create ik solver object  
ik = robotics.InverseKinematics('RigidBodyTree',jaco2n6s300)  
% Disable random restarts  
ik.SolverParameters.AllowRandomRestart = false;  
% Parameters to pass to the solver  
weights = [1, 1, 1, 1, 1, 1];  
q_init = 0.1*ones(numel(q_home),1);
```



Key Takeaway of this Talk

Success in developing an autonomous robotics system requires:

- Multi-domain simulation
- Trusted 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

Clearpath Robotics Accelerates Algorithm Development for Industrial Robots

Challenge

Shorten development times for laser-based perception, computer vision, fleet management, and control algorithms used in industrial robots

Solution

Use MATLAB to analyze and visualize ROS data, prototype algorithms, and apply the latest advances in robotics research

Results

- Data analysis time cut by up to 50%
- Customer communication improved
- Cutting-edge SDV algorithms quickly incorporated



An OTTO self-driving vehicle from Clearpath Robotics.

“ROS is good for robotics research and development, but not for data analysis. MATLAB, on the other hand, is not only a data analysis tool, it’s a data visualization and hardware interface tool as well, so it’s an excellent complement to ROS in many ways.”
- Iliia Baranov, Clearpath Robotics

Voyage develops longitudinal controls for self-driving taxis

Challenge

Develop a controller for a self-driving car to follow a target velocity and maintain a safe distance from obstacles

Solution

Use Simulink to design a longitudinal model predictive controller and tuned parameters based on experimental data imported into MATLAB using Robotics System Toolbox. Deploy the controller as a ROS node using Robotics System Toolbox. Generate source code using MATLAB Coder into a Docker Container.

Results

- Development speed tripled
- Easy integration with open-source software
- Simulink algorithms delivered as production software



Voyage's self driving car in San Jose, California.

"We were searching for a prototyping solution that was fast for development and robust for production. We decided to go with Simulink for controller development and code generation, while using MATLAB to automate development tasks."

- Alan Mond, Voyage

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

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

Preceyes Accelerates Development of World's First Eye-Surgery Robot Using Model-Based Design

Challenge

Develop a real-time control system for robot-assisted surgical procedures performed within the human eye

Solution

Use Model-Based Design with MATLAB and Simulink to model and simulate the control system and use Simulink Coder and Simulink Real-Time to deploy it to a real-time target

Results

- Development Core controller developed by one engineer
- Patient safety assured
- Road map to industrialization set

[Link to user story](#)

MATLAB EXPO 2018



The PRECEYES Surgical System. Image copyright and courtesy Preceyes.

“MATLAB and Simulink provided a single platform that supported our complete workflow and all the components and protocols we needed for our robotic system. That enabled us to quickly develop a safe, real-time device, ready for clinical investigation.”
- Maarten Beelen, Preceyes

```
% Thank you
```

