MATLAB EXPO 2018

Developing Algorithms for Robotics and Autonomous Systems

이제훈 차장

Key Takeaway of this Talk

Success in developing an autonomous robotics system requires:

1. Multi-domain simulation
2. Trusted tools

   : make complex workflows easy & integrate with other tools
3. Model-based design

Challenges with Autonomous Robotics Systems

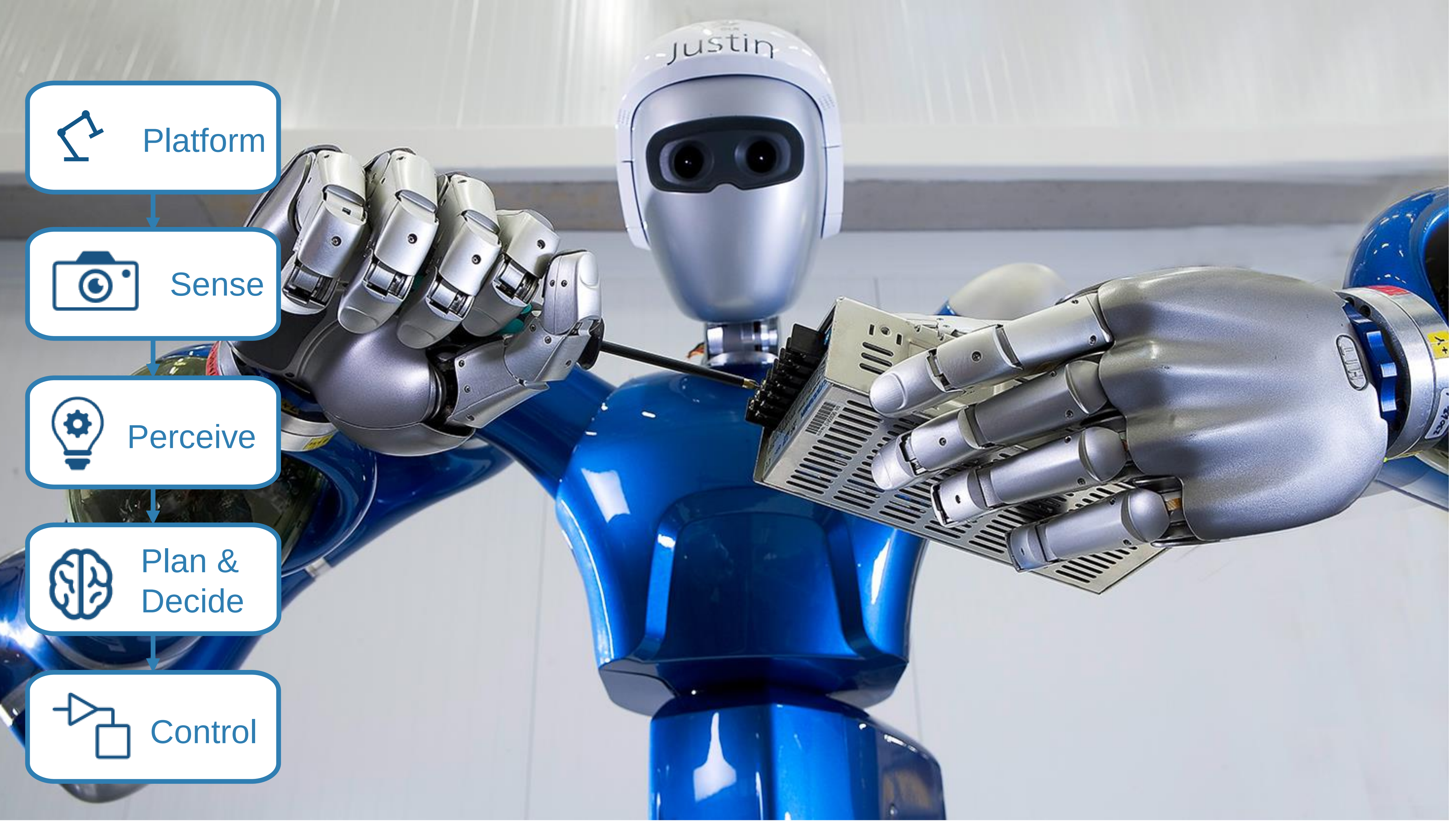
Applying Multidomain Expertise

Complexity of Algorithms

End-to-End workflows

Technical Depth and System Stability

What does success look like?


Platform
Sense
Perceive
Plan &
Decide
Control
Justin

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.

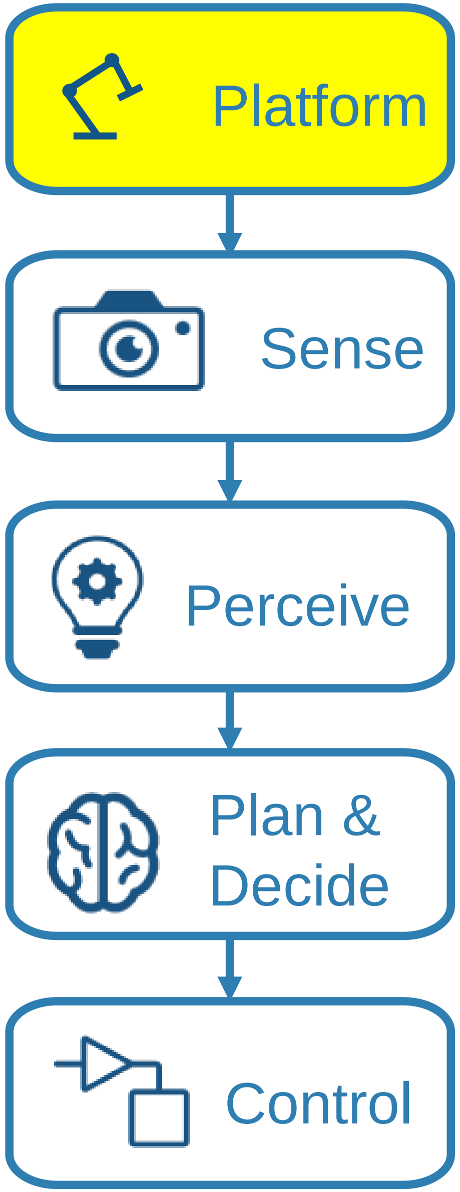
"Model-Based Design and automatic code generation enable us to cope with the complexity of Agile Justin's 53 degrees of freedom. Without Model-Based Design it would have been impossible to build the controllers for such a complex robotic system with hard real-time performance."

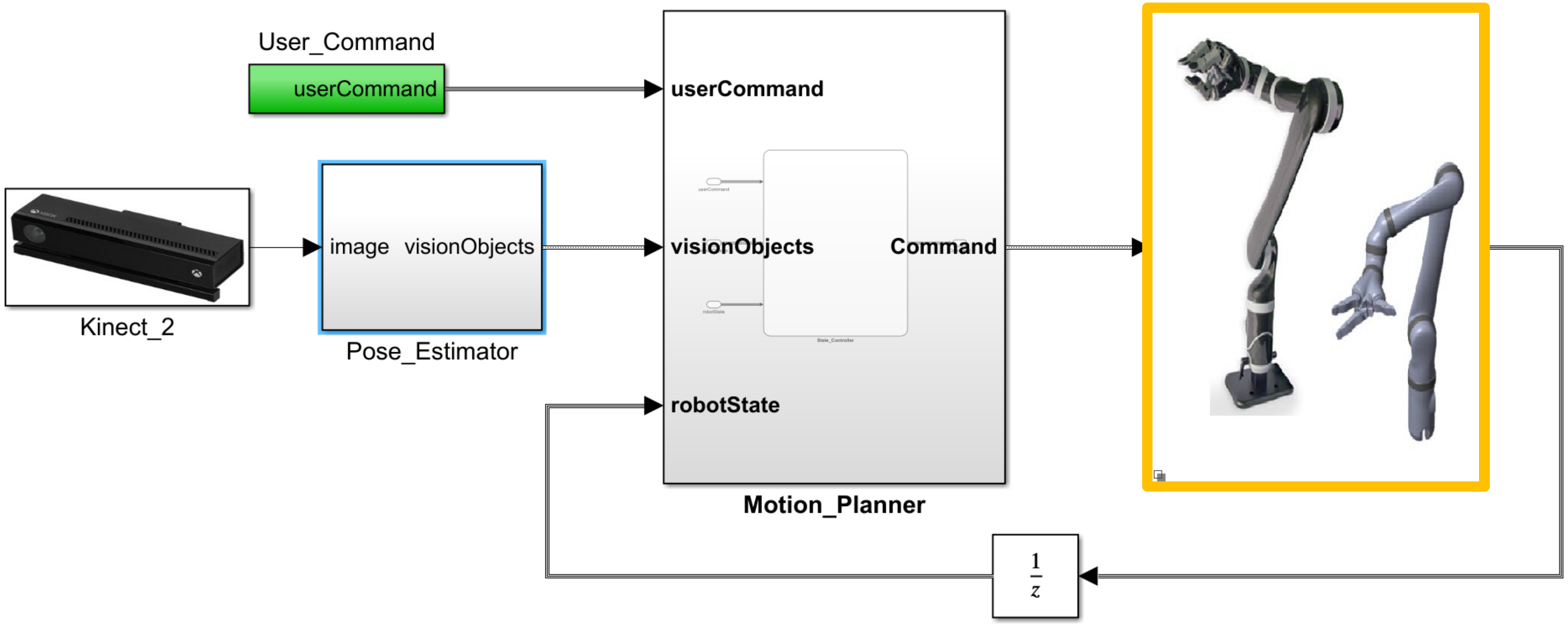
Berthold Bäuml

DLR

Link to user story

Today: Design Pick and Place Application





Platform Design

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

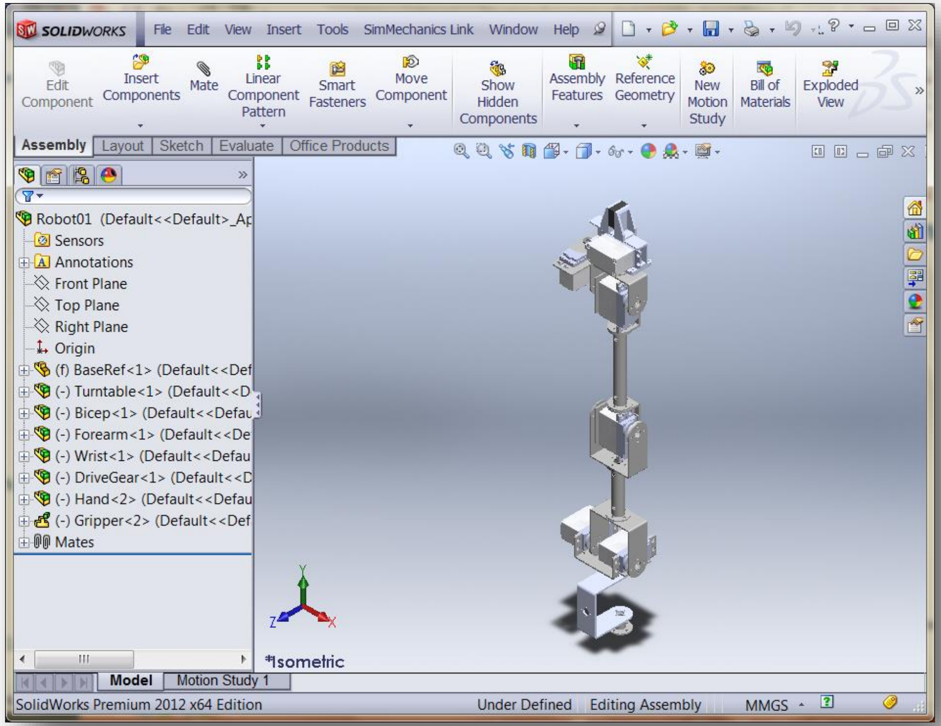
Mechanics

Actuators

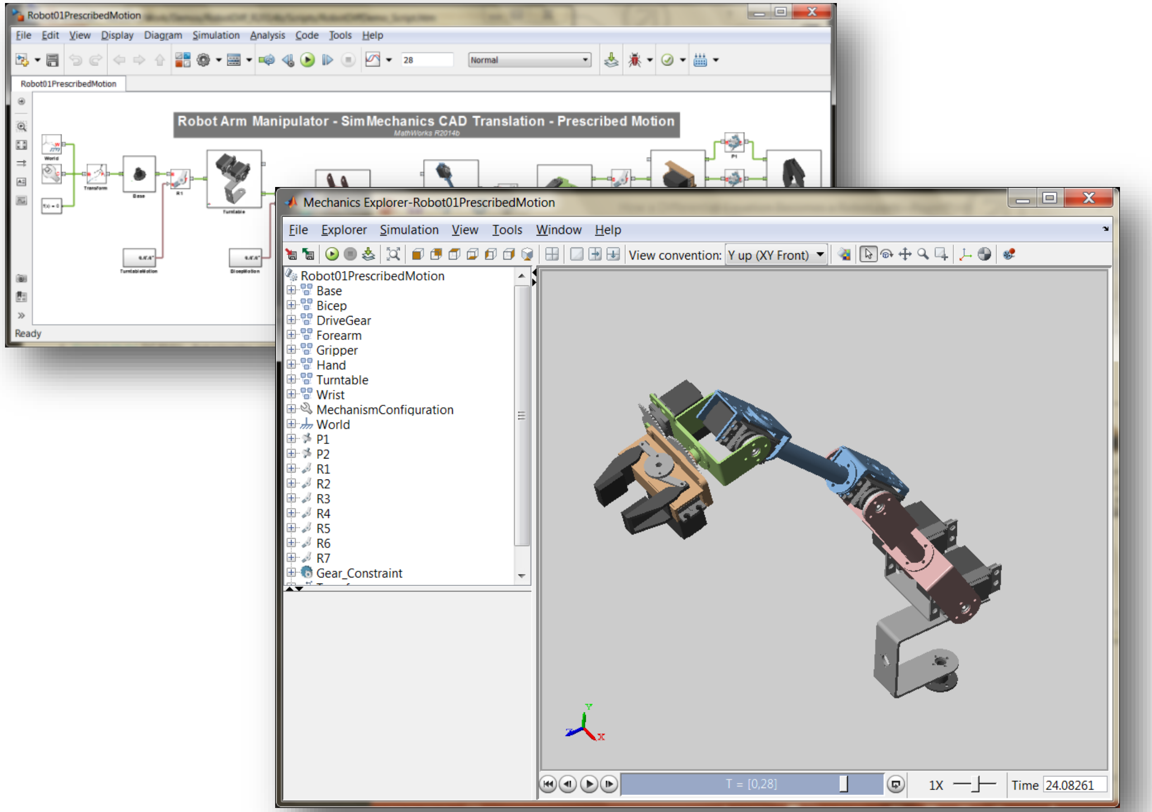
Environment

Mechanics: Import models from common CAD Tools

SolidWorks Model

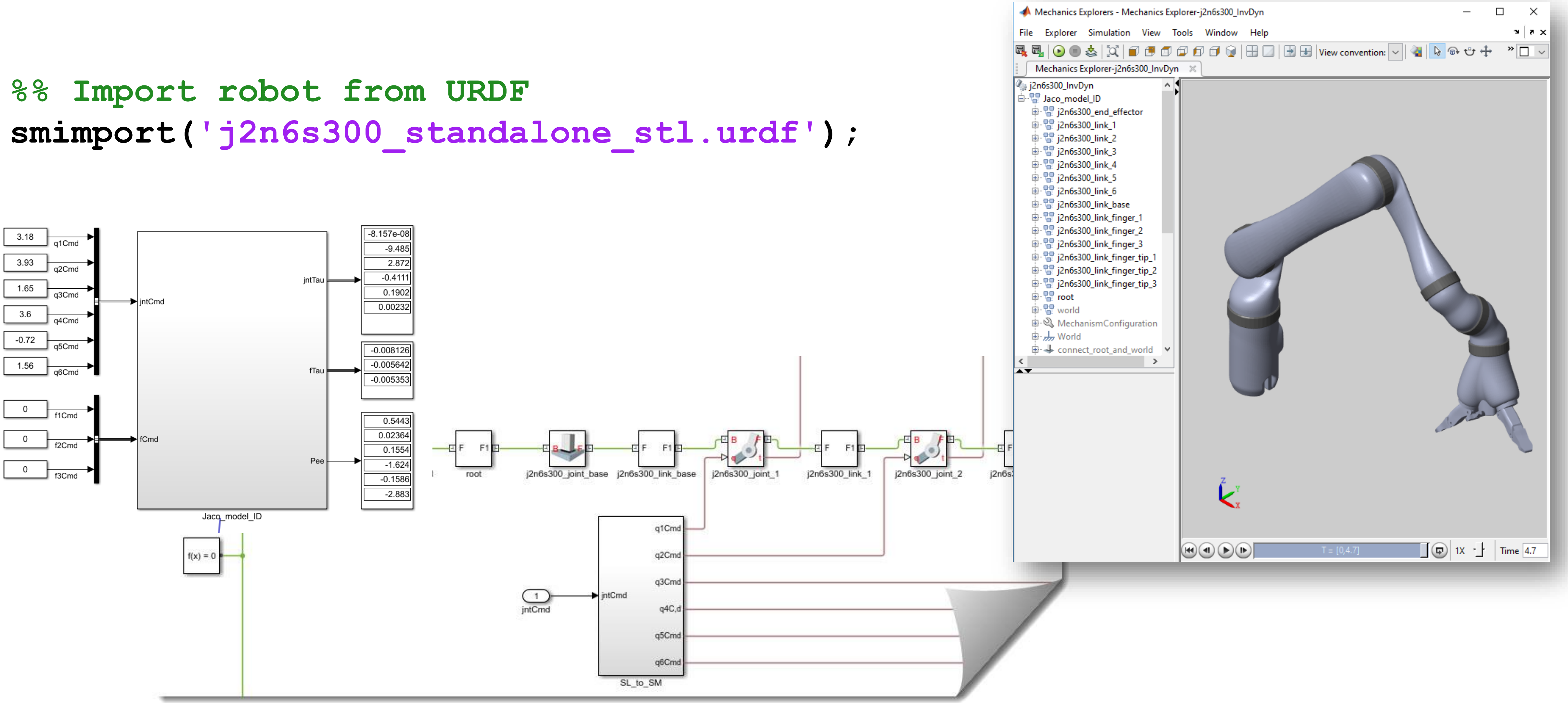


Simscape Multibody Model



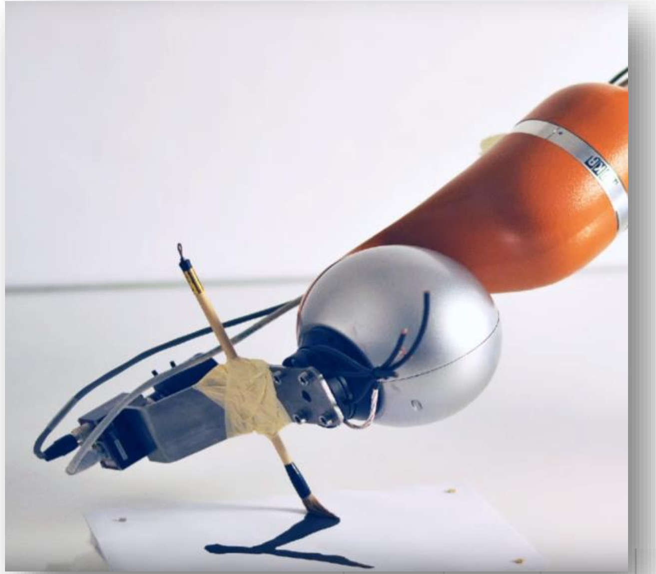
Mechanics: One line import from URDF

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

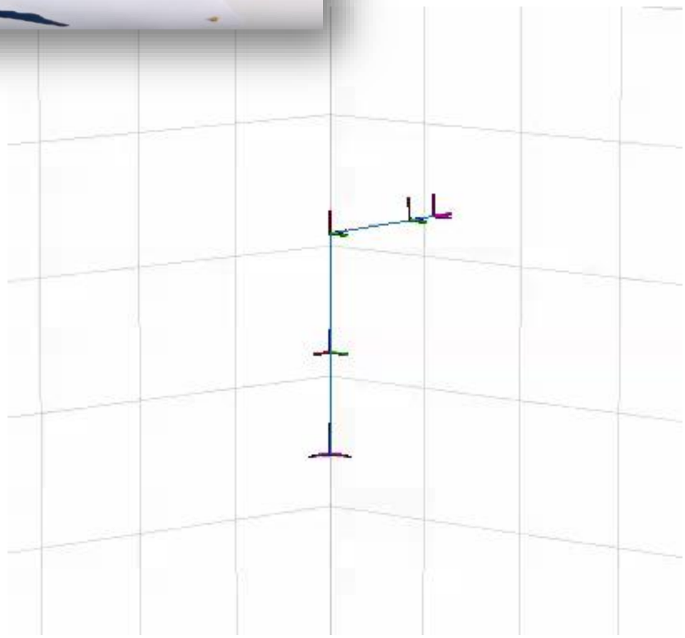


Demo

Rigid Body Tree Dynamics



R2017a

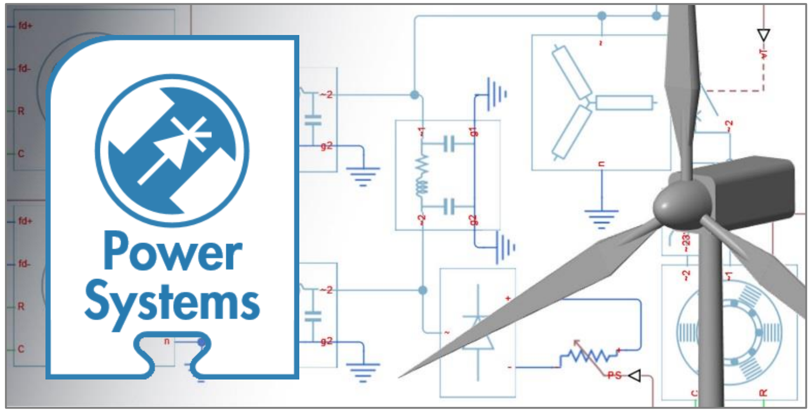


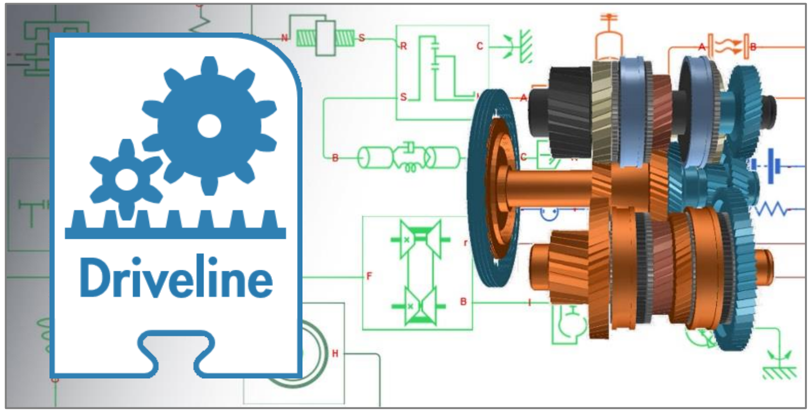
Compute rigid body tree dynamics quantities

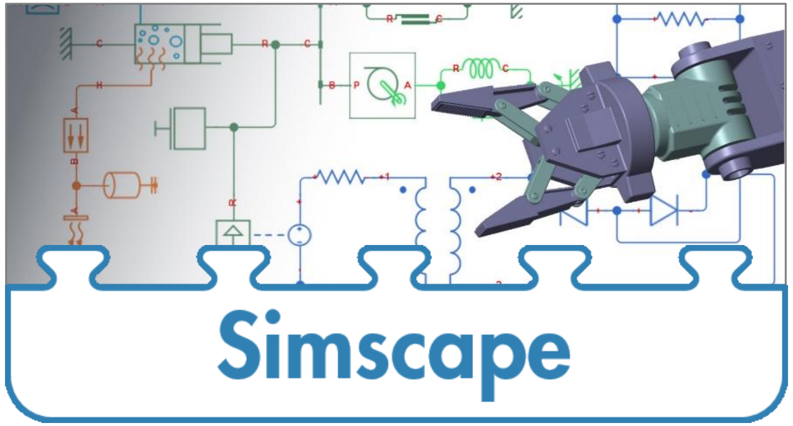
- Specify rigid body inertial properties
- Compute for the rigid body tree
  - Forward dynamics
  - Inverse dynamics
  - Mass matrix
  - Velocity product
  - Gravity torque
  - Center of mass position and Jacobian

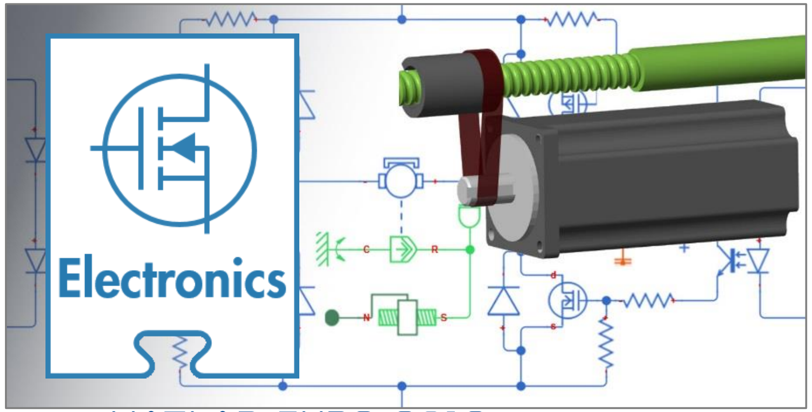
```
» load exampleRobots.mat
» lbr.DataFormat = 'column';
» q = lbr.randomConfiguration;
» tau = inverseDynamics(lbr, q);
```

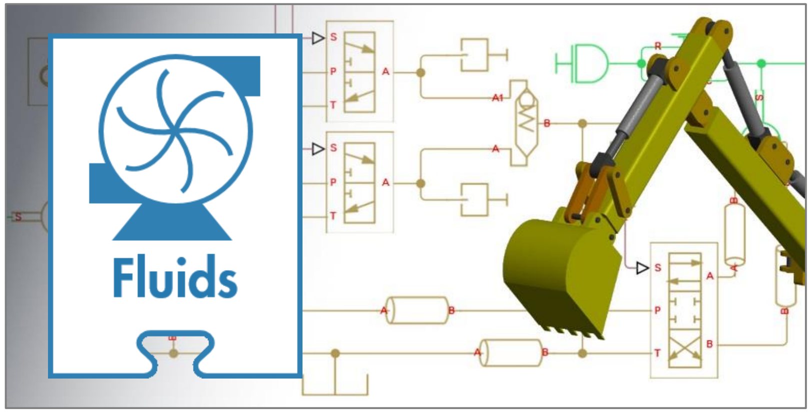
Actuators: Model other domains

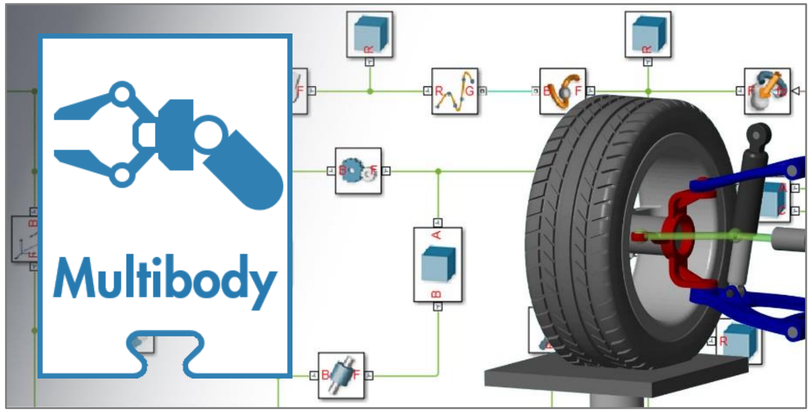




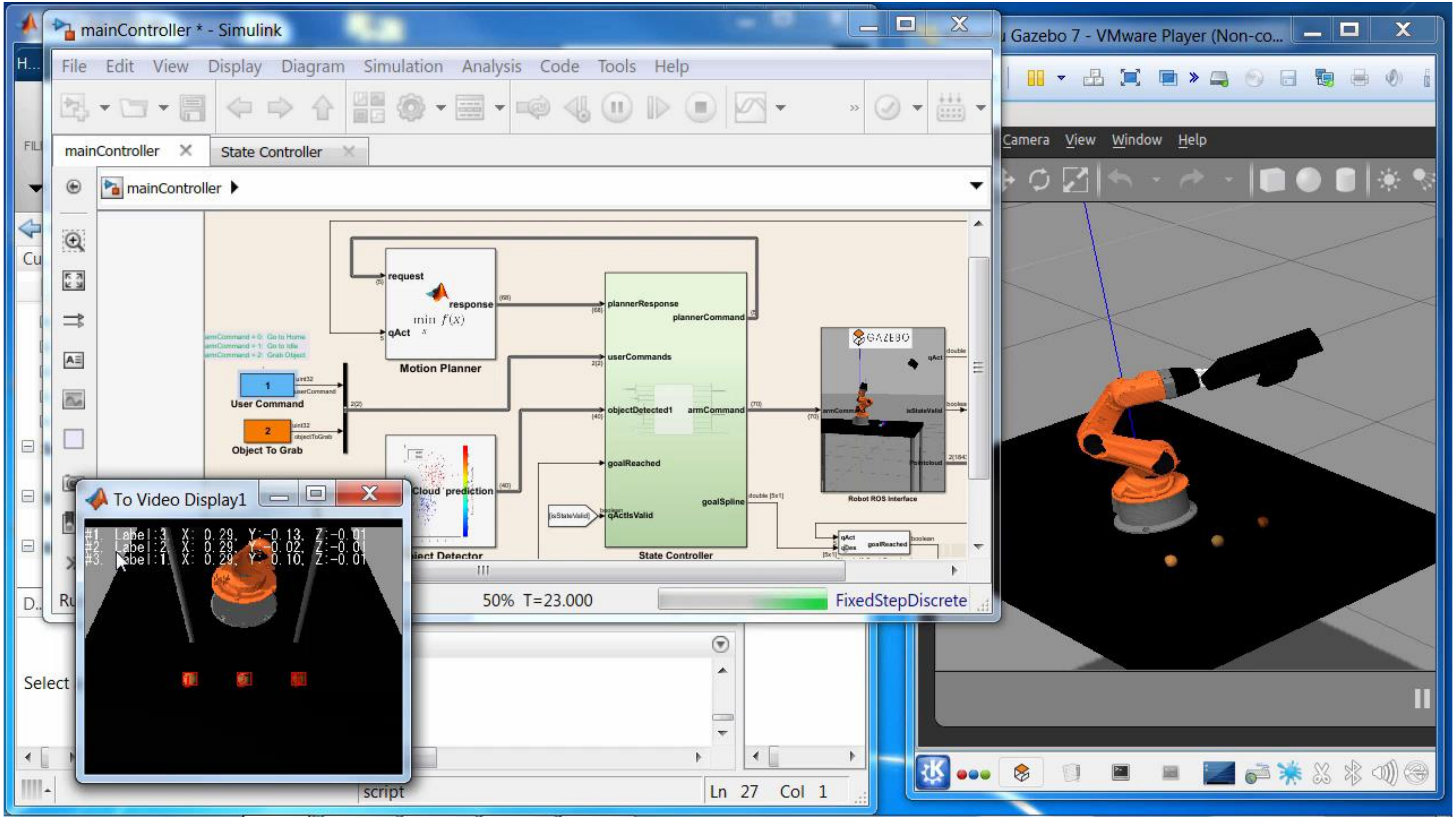




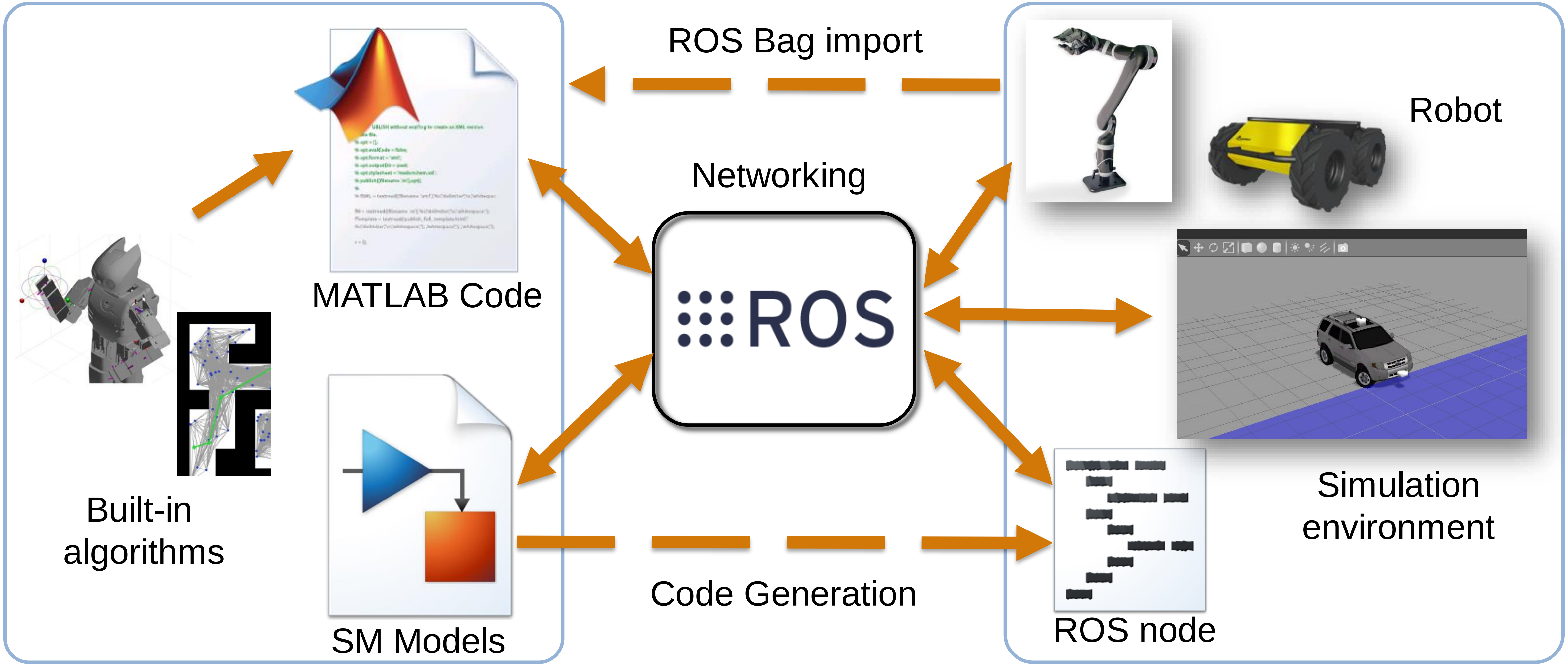




Environment: Connect to an external robotics simulator



Environment: Connect MATLAB and Simulink with ROS



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."
- Ilia Baranov, Clearpath Robotics

[Link to user story](#)

Design Pick and Place Application

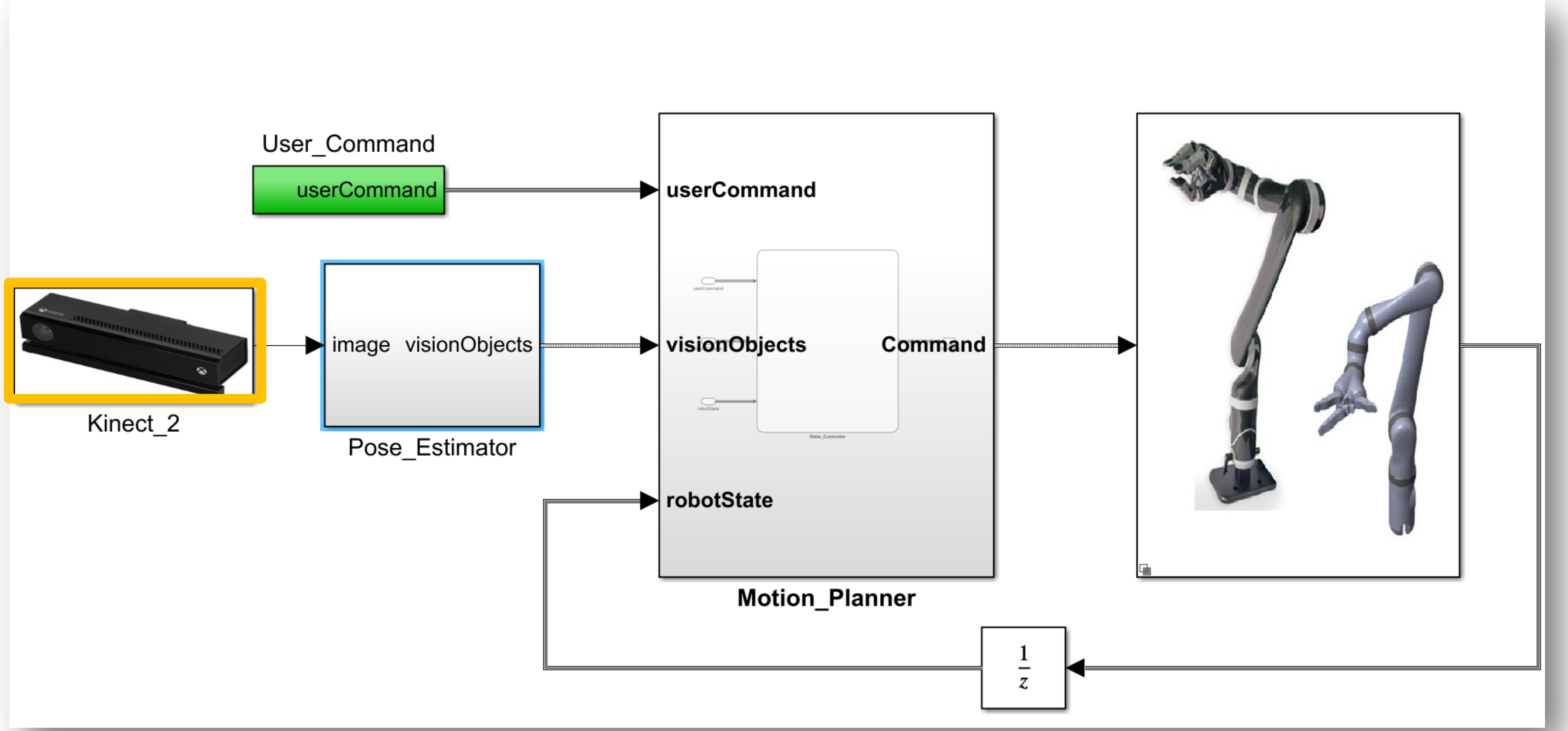
Platform

Sense

Perceive

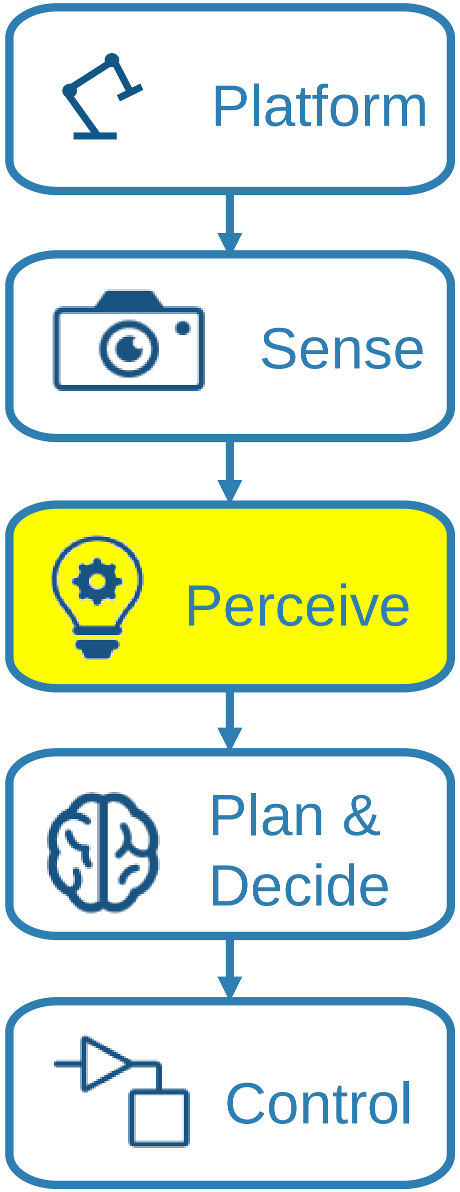
Plan & Decide

Control

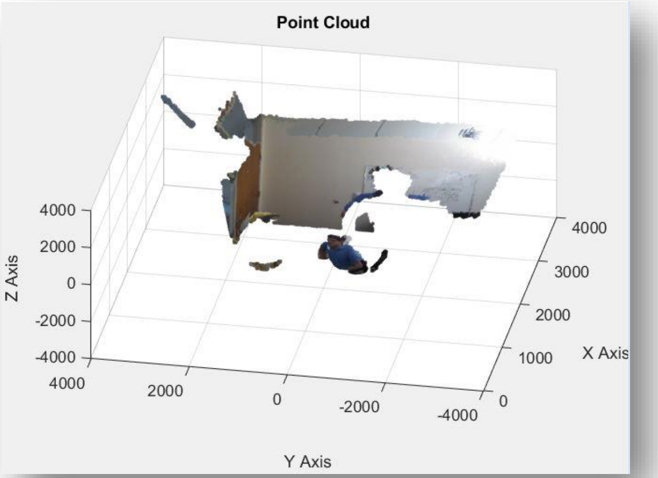


Demo

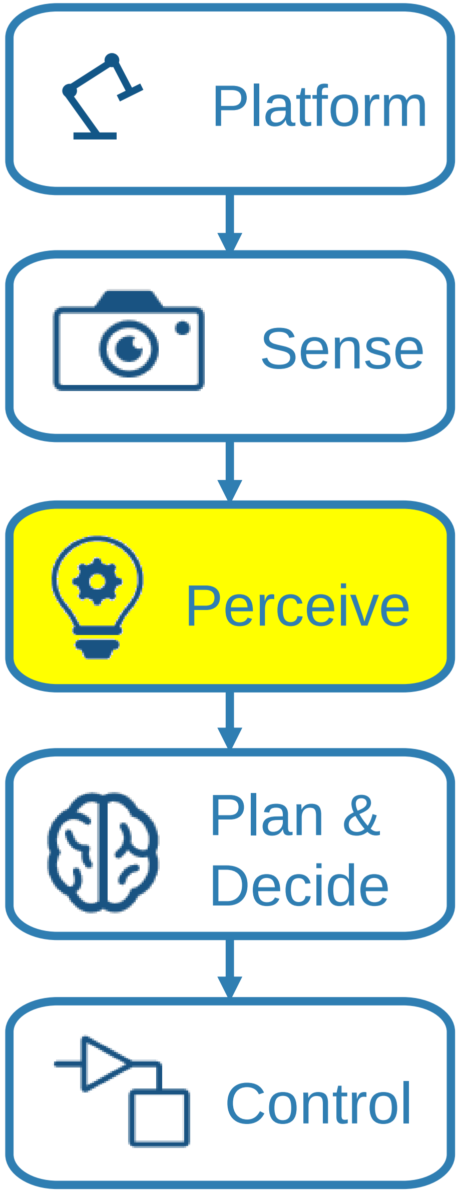
Design Pick and Place Application

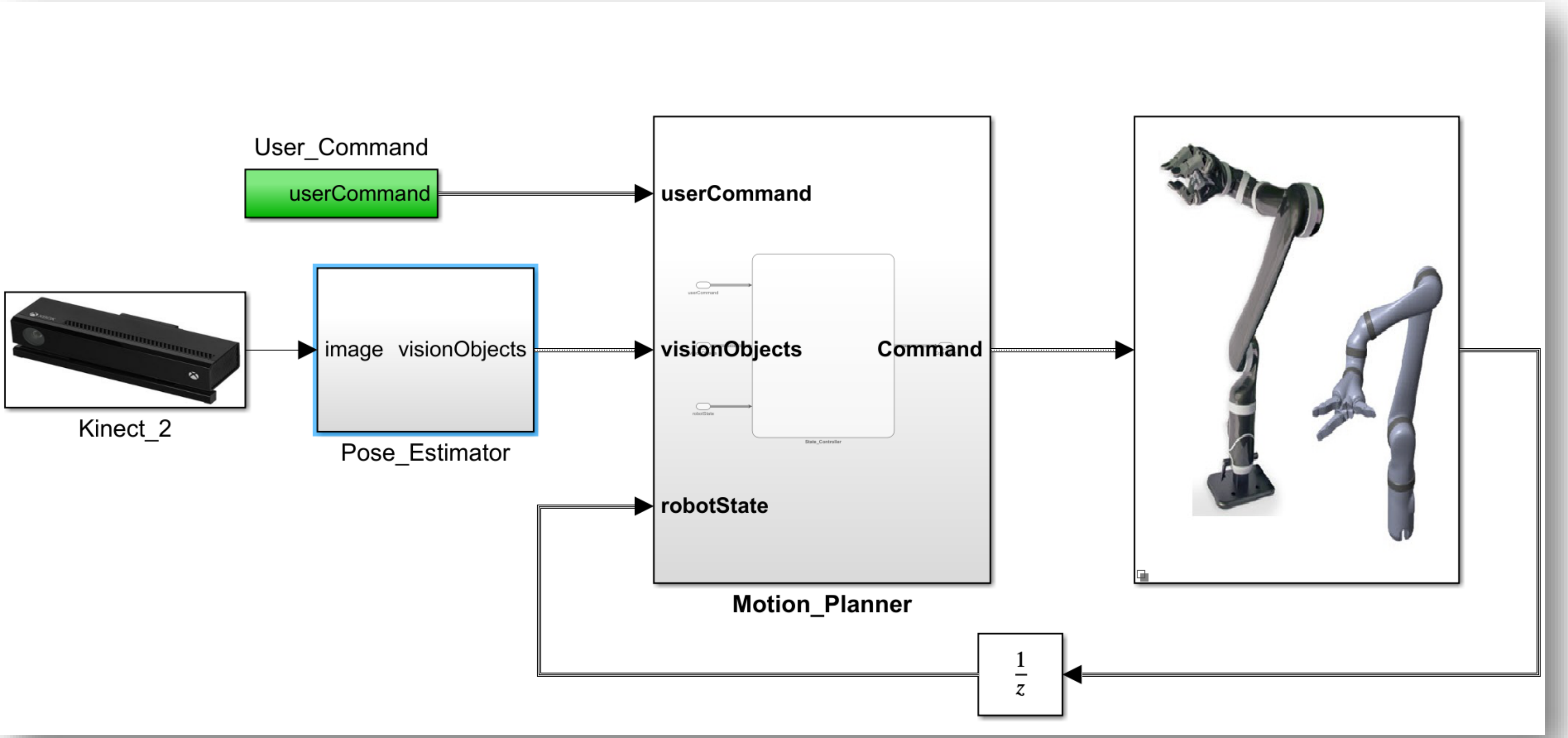


- Support for Common Sensors
- Image analysis
- Apps
- Image enhancement
- Visualizing Point Clouds



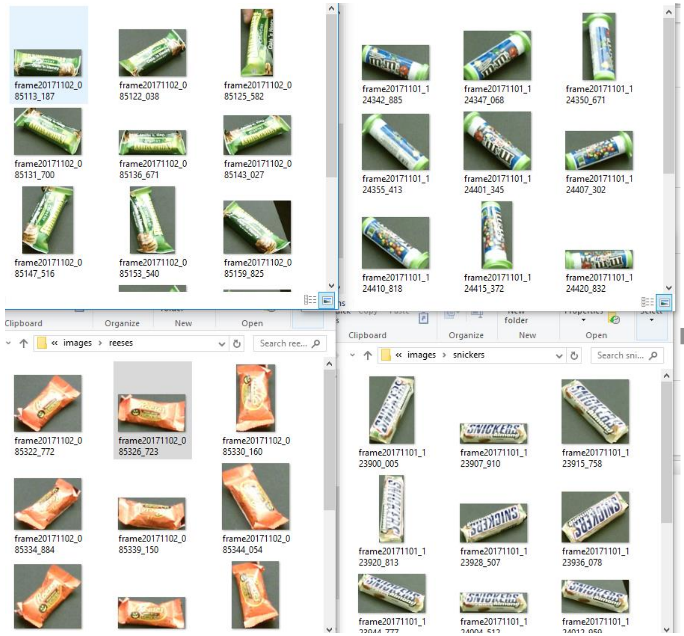
Today: Design Pick and Place Application





Object Classifier and Pose Estimator

Images

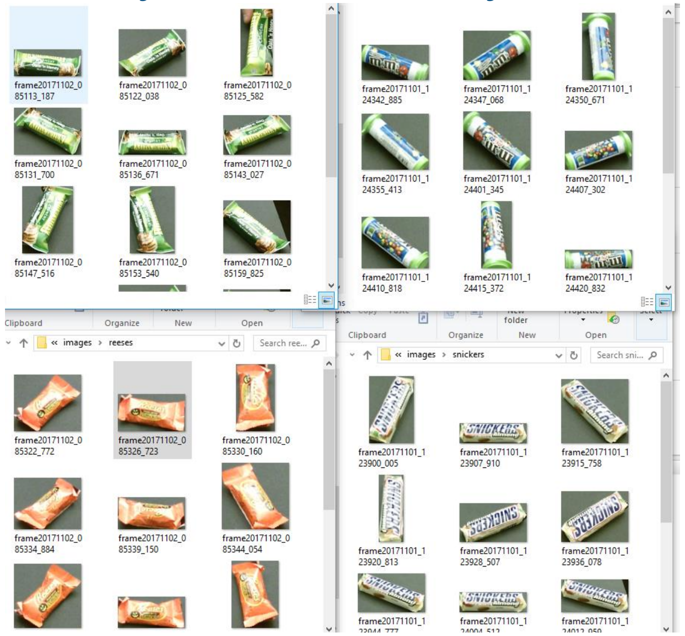


Pose Estimator

Labels and Poses

Object 1

Object 2



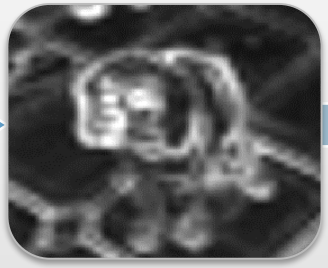
Object 3

Object 4

MATLAB makes machine learning easy and accessible

Traditional Machine Learning approach

Traditional Feature Extraction

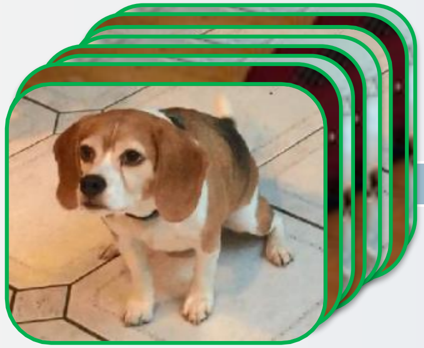


Classification

Machine Learning

Dog ✓
Boy ✗
•
•
Bicycle ✗

Deep Learning approach



Convolutional Neural Network (CNN)

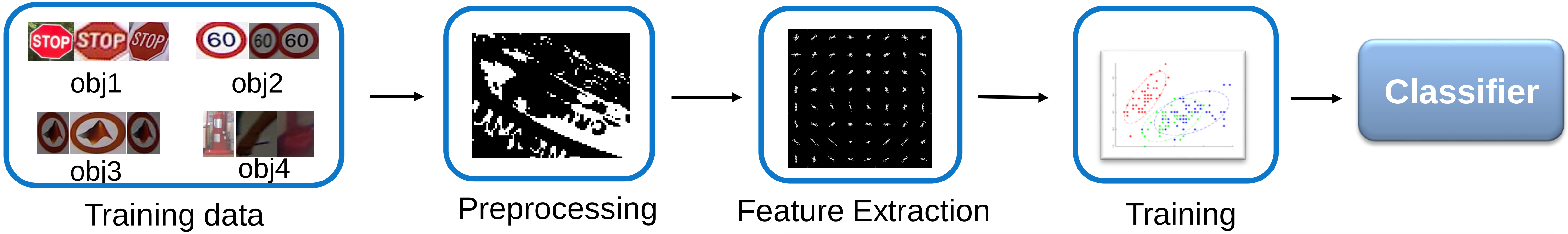
Learned features [95%]

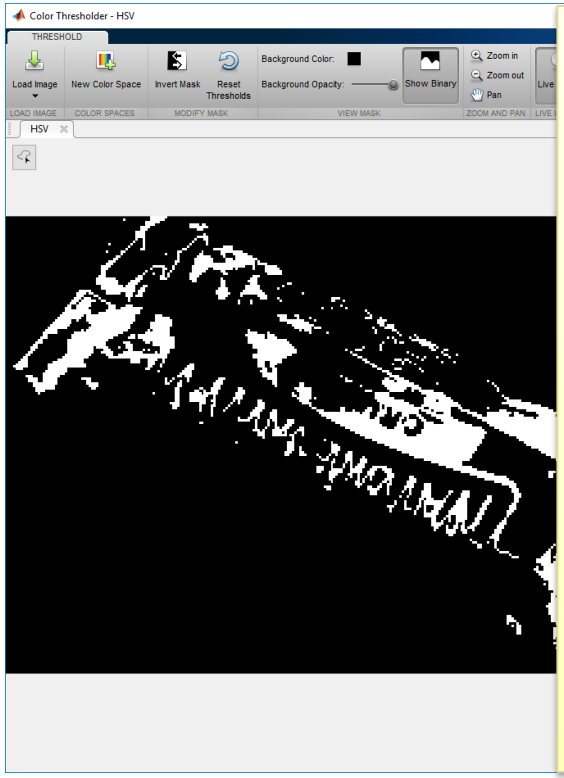
End-to-end learning

Feature learning + Classification

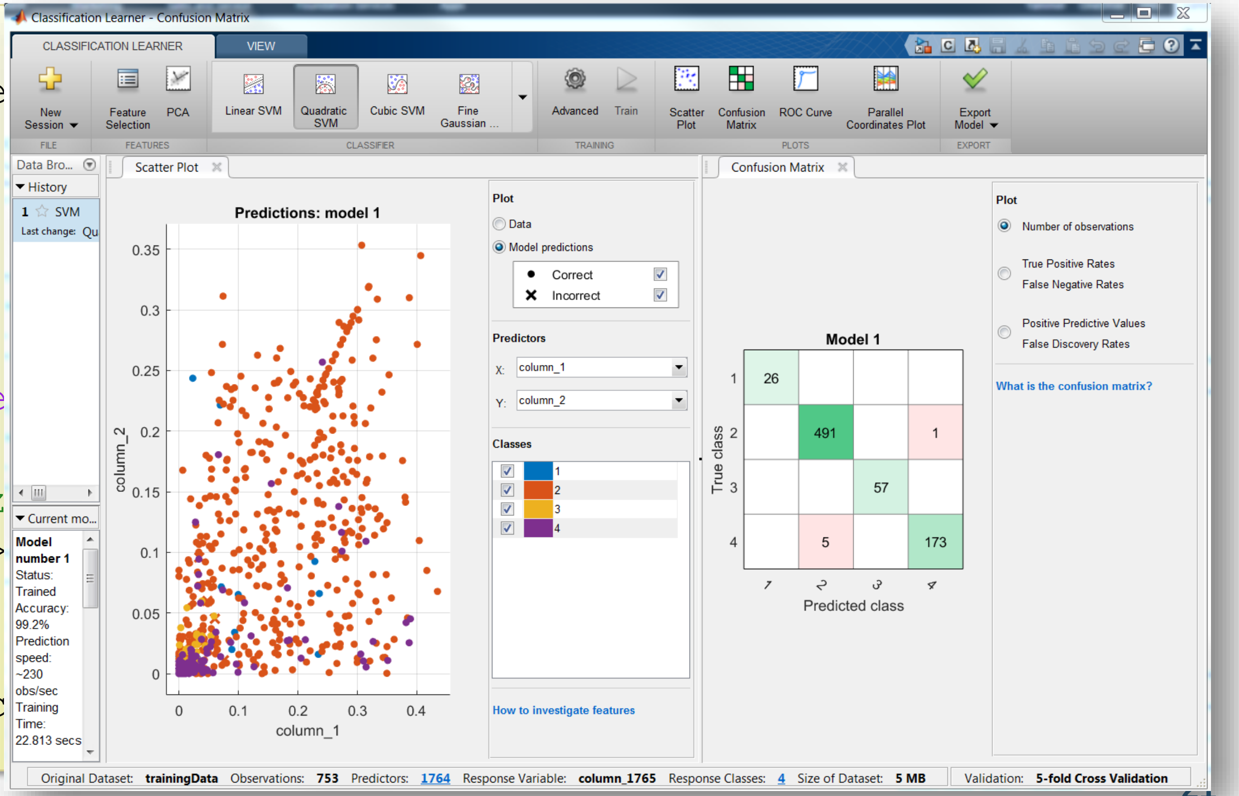
Dog ✓
Boy ✗
•
•
Bicycle ✗

Complex workflows made easy with MATLAB

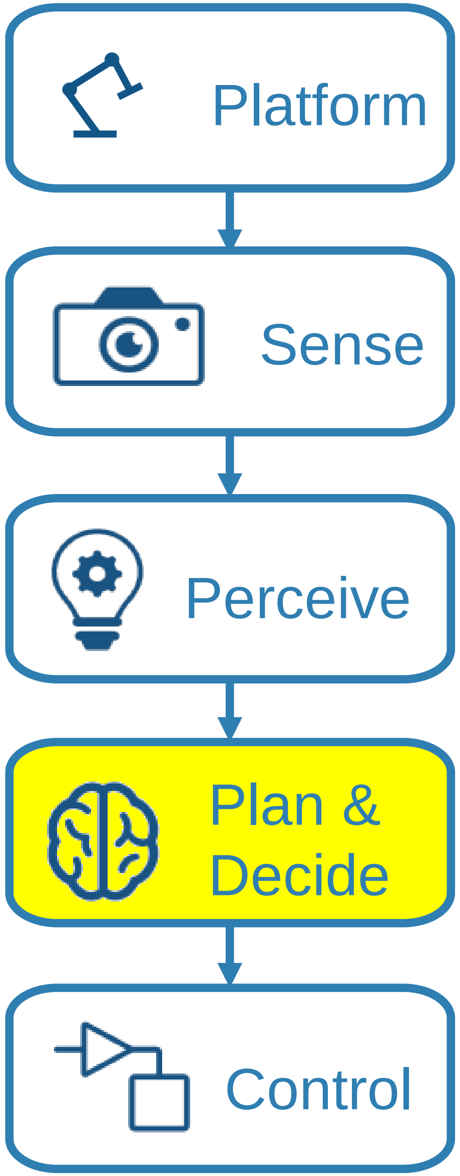


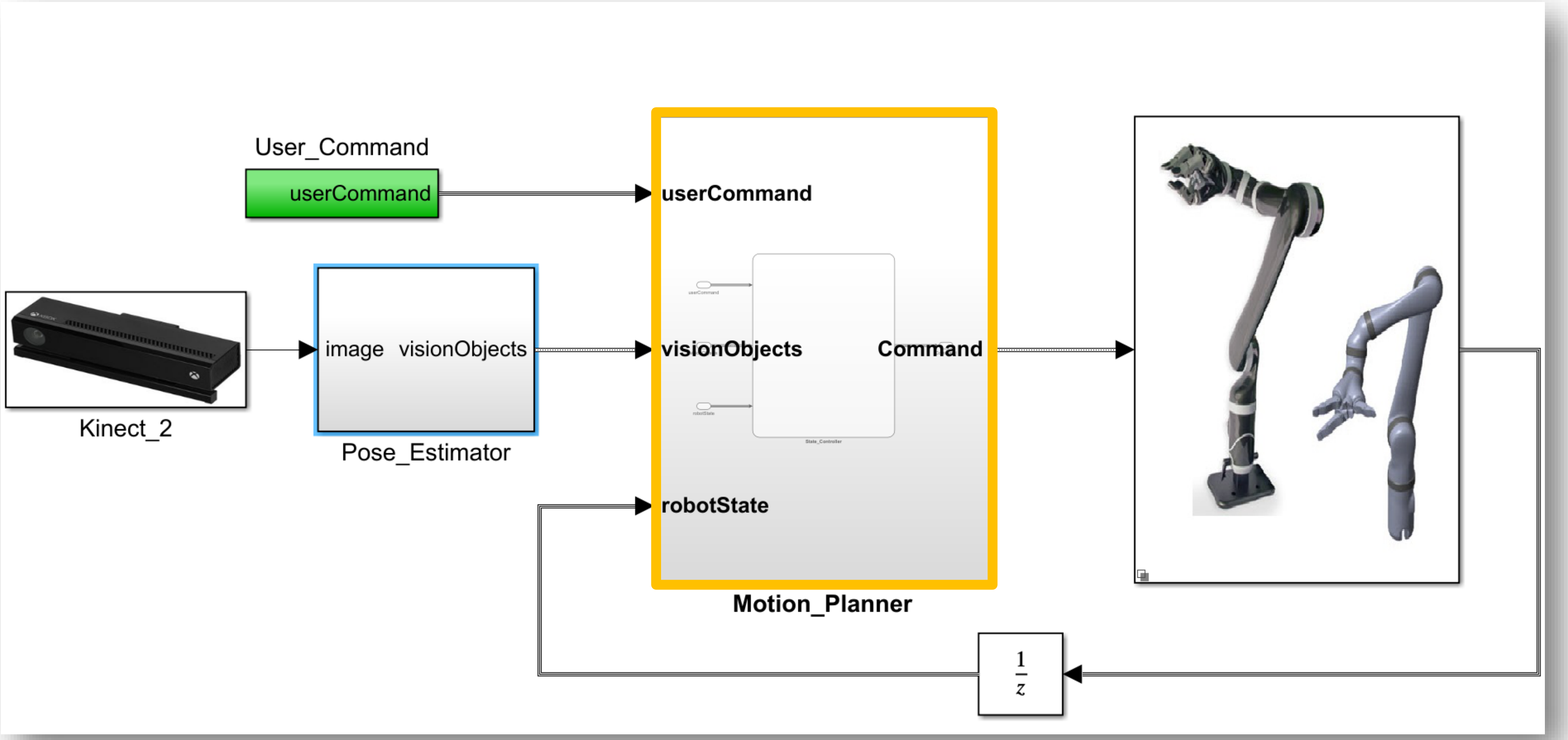



```
% Detect regions
BW = createMask(videoFrame

% 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
```

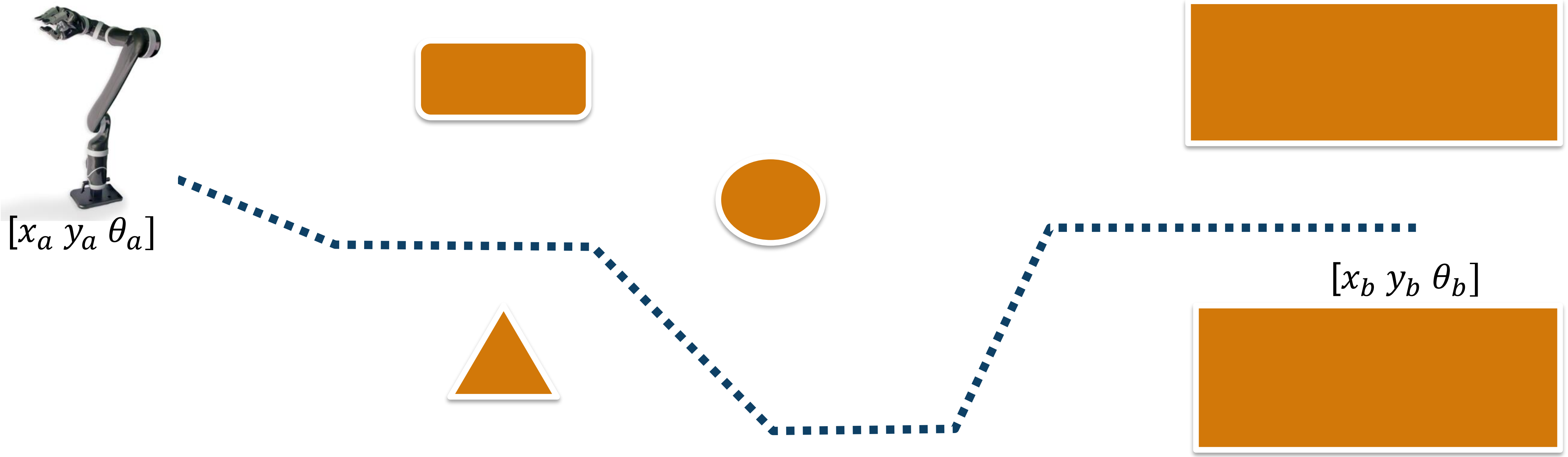


Design Pick and Place Application

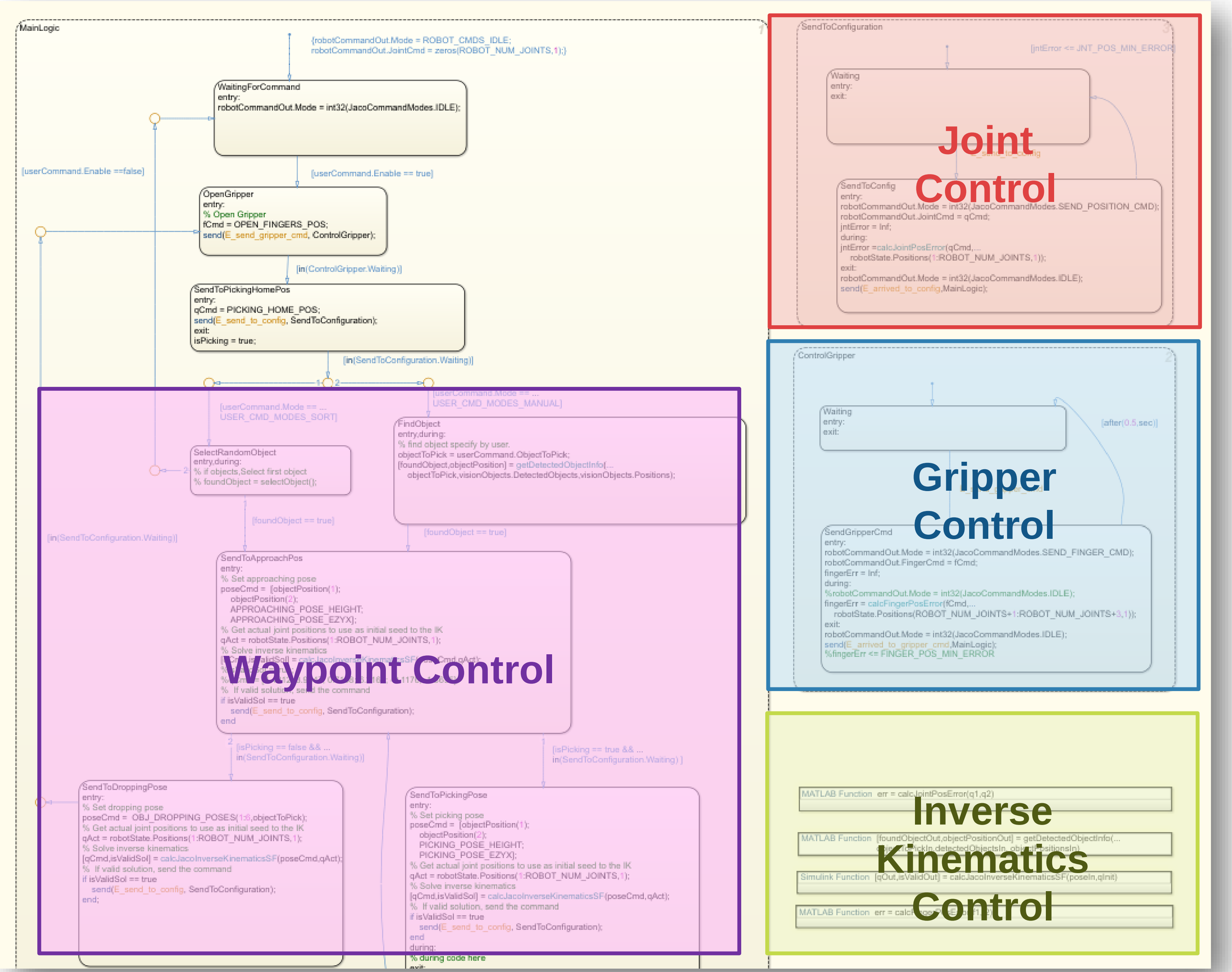




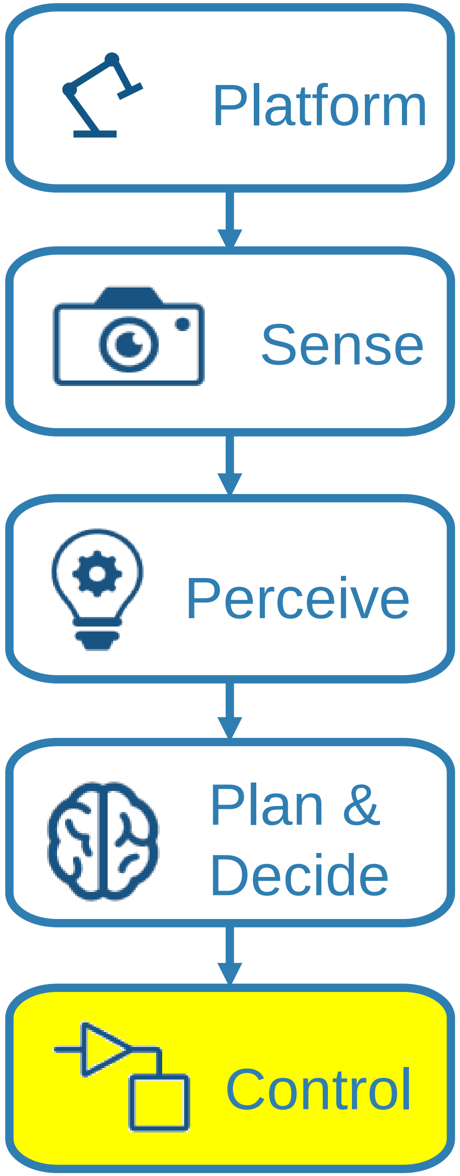
Planning: Find a path

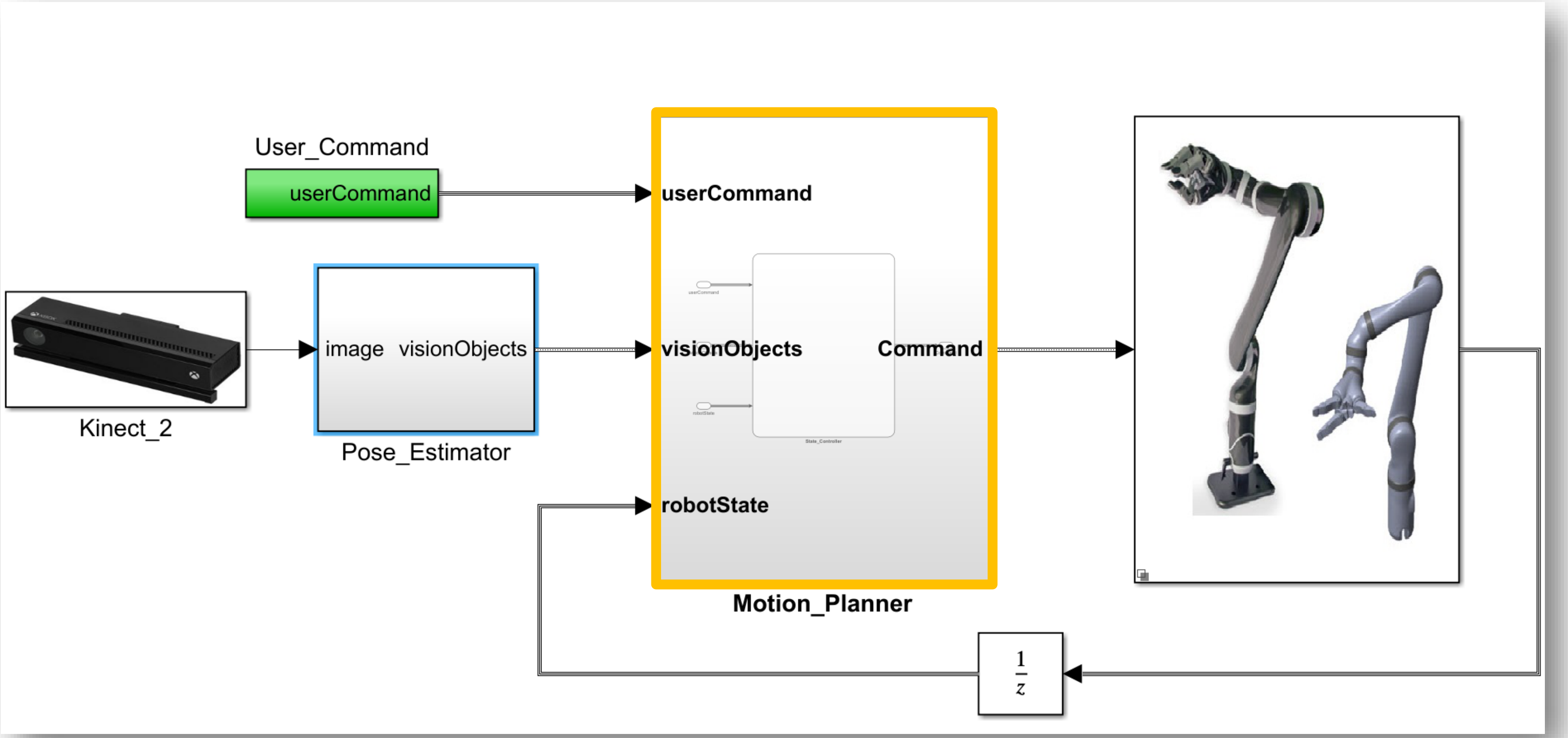


Plan with Stateflow



Design Pick and Place Application

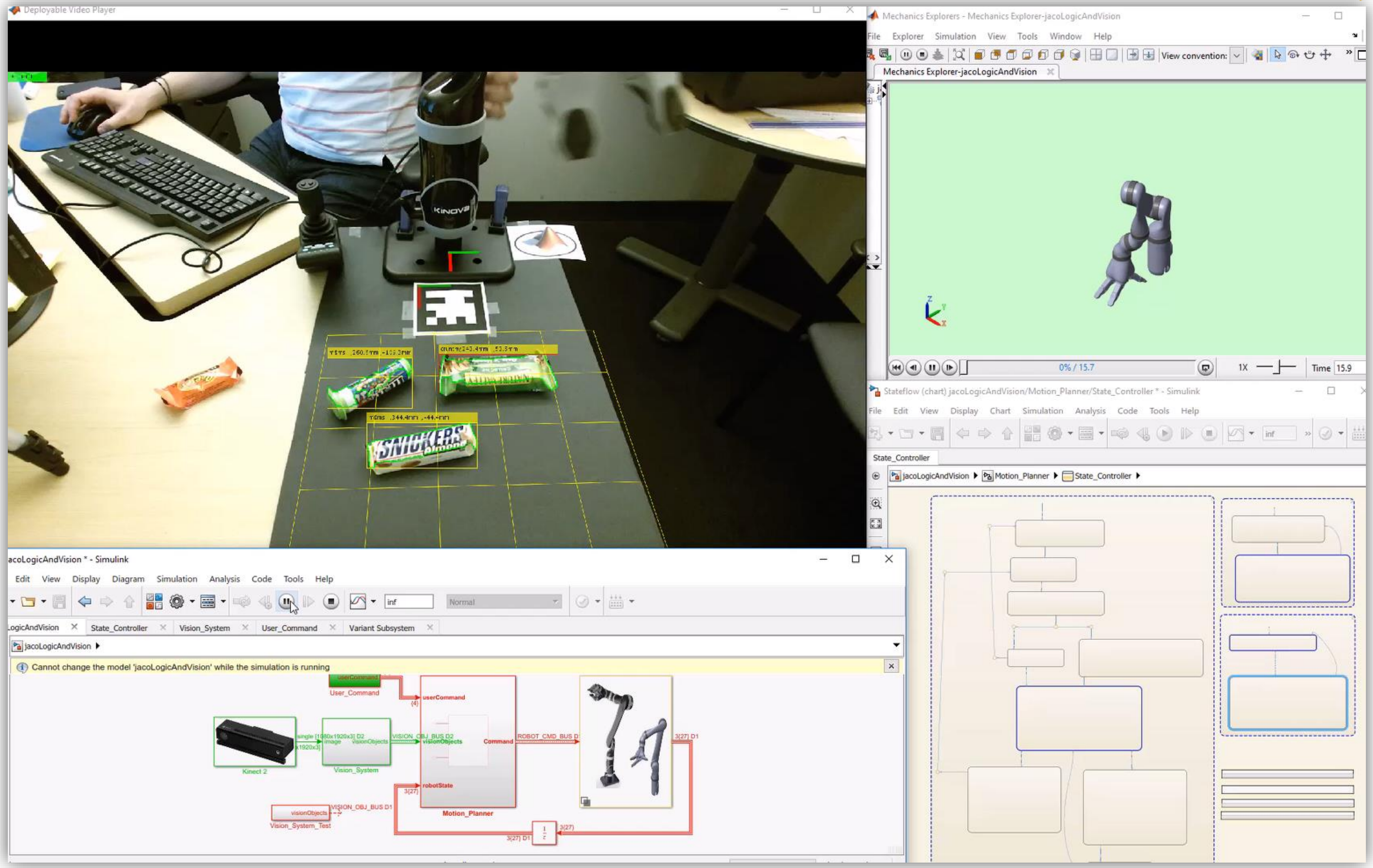




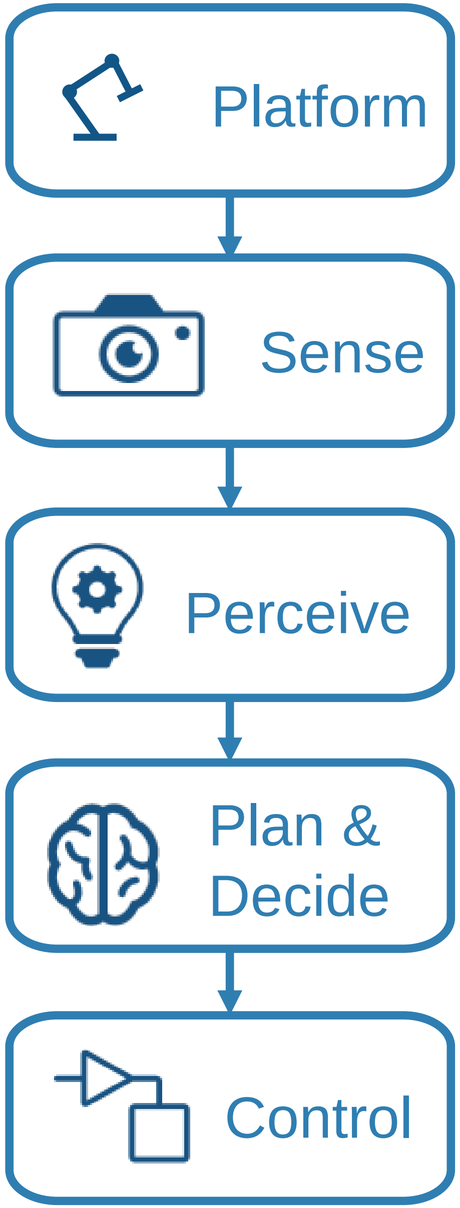
Explore Built In Functions: Inverse Kinematics

```
% 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);
```


Demo


Deployable Video Player
Mechanics Explorers - Mechanics Explorer-jacoLogicAndVision
Stateflow (chart) jacoLogicAndVision/Motion_Planner/State_Controller * - Simulink
acoLogicAndVision * - Simulink
Cannot change the model 'jacoLogicAndVision' while the simulation is running
User_Command
Kinect 2
Vision_System
Motion_Planner
Vision_System_Test

Key Takeaway of this Talk

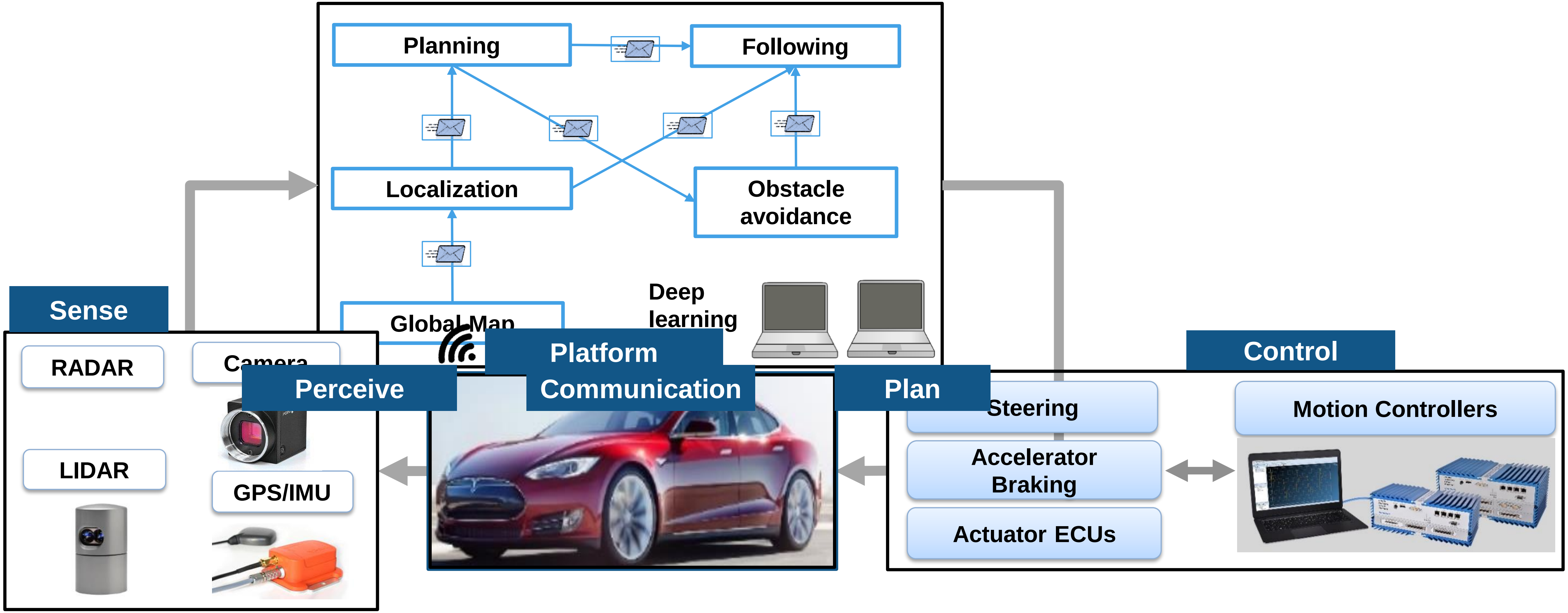


Success in developing an autonomous robotics system :

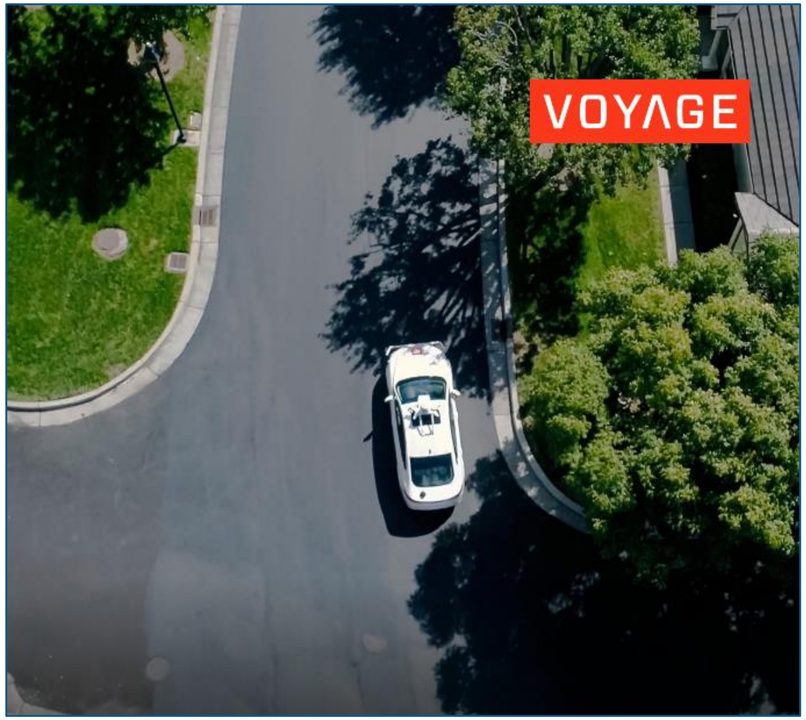
1. Multi-domain simulation
2. Trusted tools

   : make complex workflows easy & integrate with other tools
3. Model-based design

Another Example: Self-Driving Cars



Voyage develops longitudinal controls for self-driving taxis



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

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

"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

Link to user story

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

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

Link to user story

% Thank you