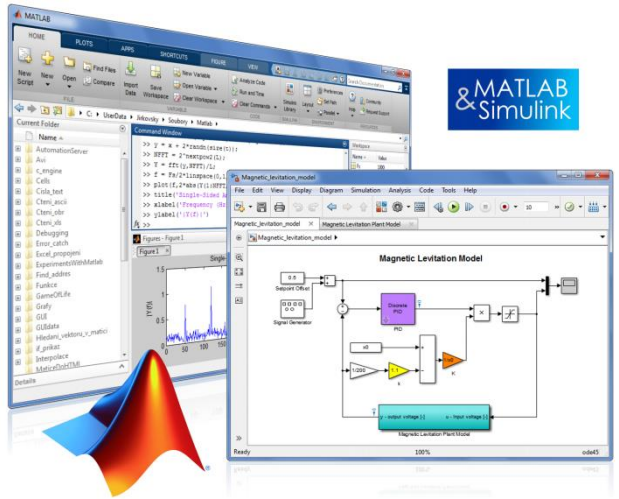
HUMUSOFT®

Robotic systems development in MATLAB



Michal Blaho

blaho@humusoft.sk

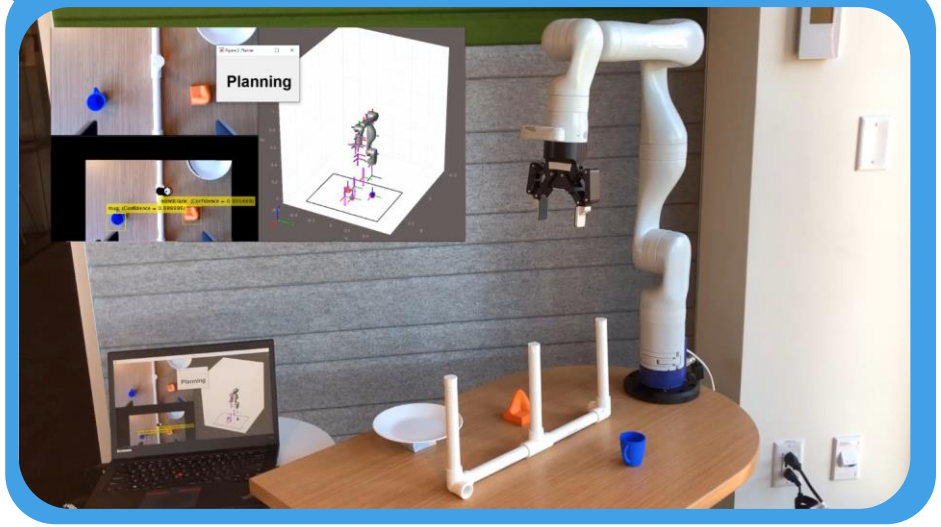
www.humusoft.cz

info@humusoft.cz

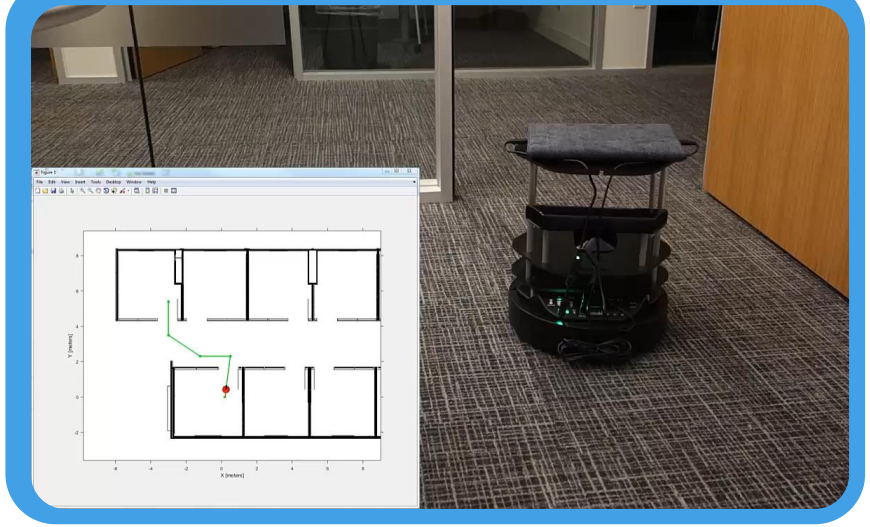
www.mathworks.com

Robot Applications

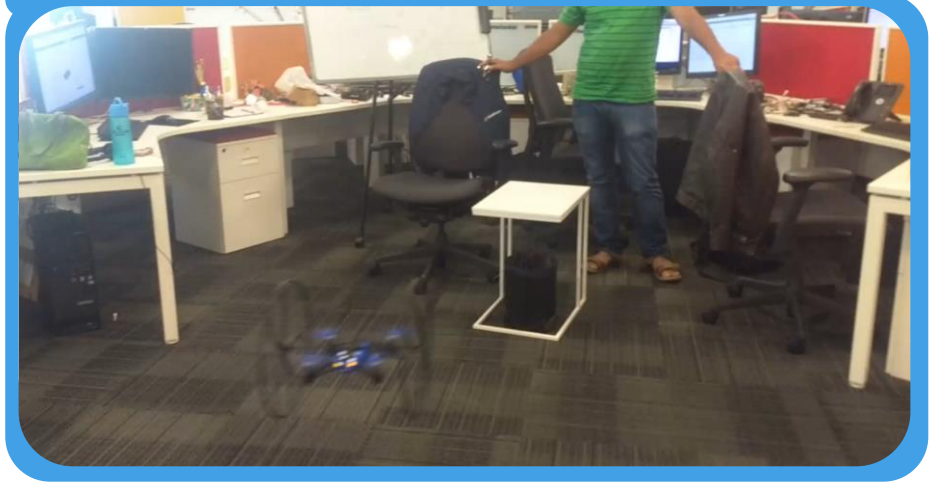
Manipulator Arms



Mobile Robots

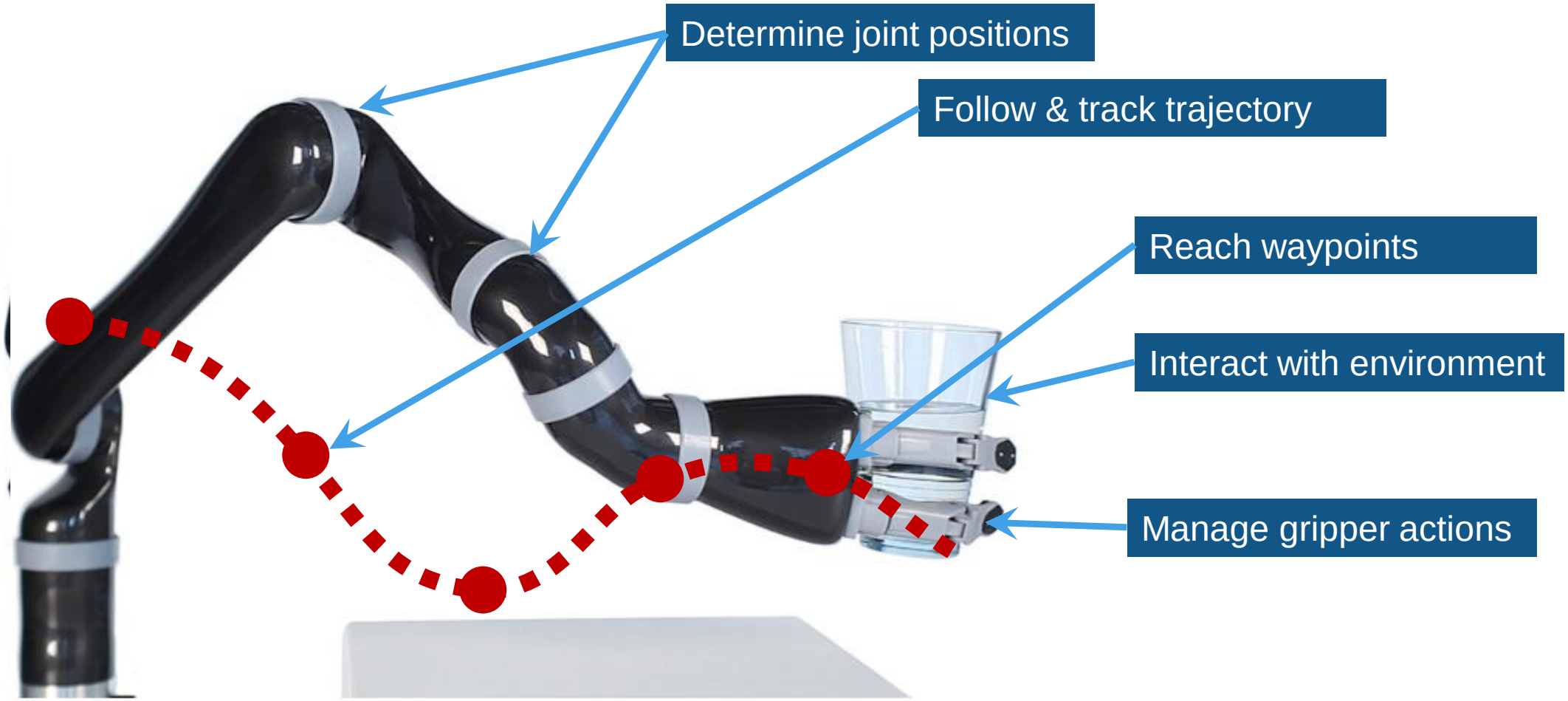


UAVs



Humanoids

Challenges in Designing Robotics System



Components of Robotics System Development

HUMUSOFT

V&V

Simulink Test | Simulink Coverage | Polyspace

Connect /Deploy

Coder Products

ROS Toolbox

Autonomous Algorithm

Navigation Toolbox

Localization & Mapping

Sensor Fusion and Tracking Tbx

Planning

Navigation Toolbox

Robotics System Toolbox

Computer Vision Toolbox

Sensing & Perception

Deep Learning Toolbox

Control

Stateflow

Reinforcement Learning Toolbox

Model Predictive Control Toolbox

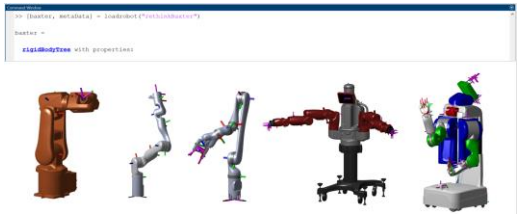
Platform

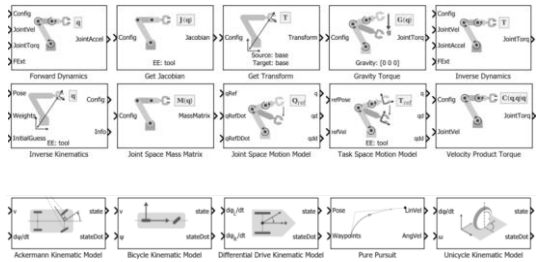
MATLAB | Simulink | Simscape

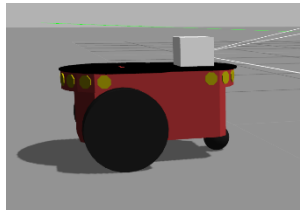
Robotics System Toolbox

Robot Model	Algorithms	Simulation	Deployment

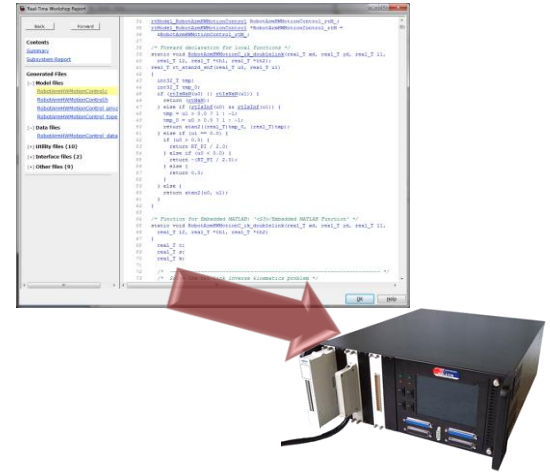
• Rigid body tree (RBT) representation
• Robot model library of commercially available robots
• Import RBT from URDF file, text, Simscape multibody model
	• Library of common robot algorithms
	  ○ Manipulator arms
	  ○ Mobile robots
	  ○ UAVs
		• Motion modeling and simulation to prototype algorithms quickly and test behavior of real-world systems.
		• Synchronized Gazebo co-simulation from Simulink
			• Generate C/C++ code and MEX functions for algorithm acceleration



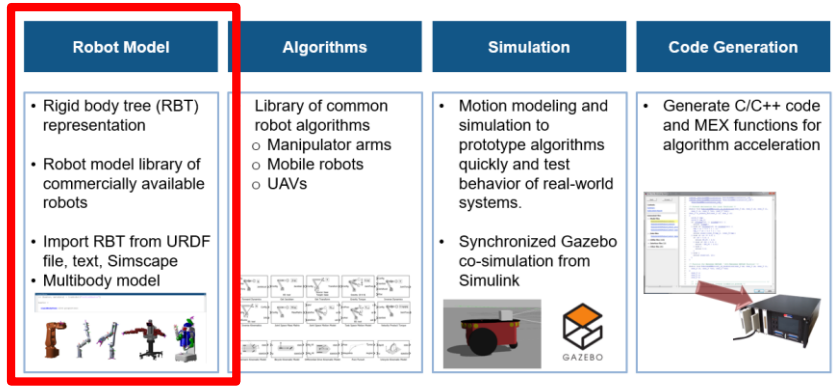




GAZEBO

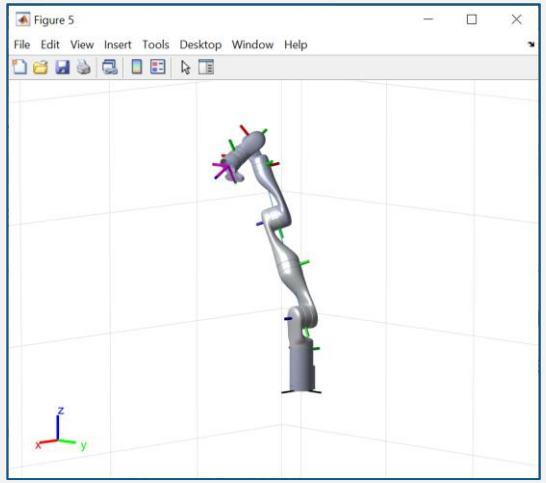


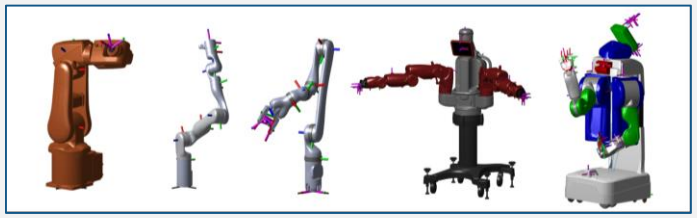
Robot Model

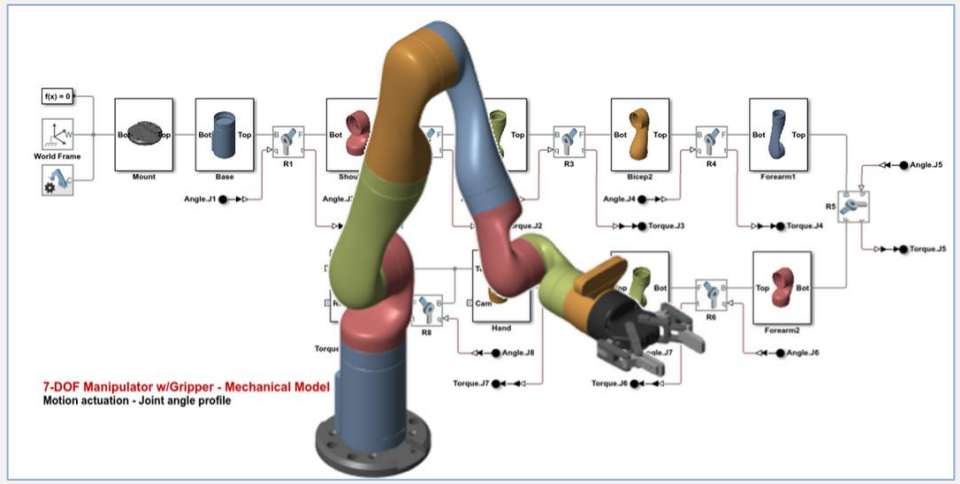


- Rigid body tree (RBT) representation
- Load a RBT robot model from a library of commonly used robots
- Import a RBT robot mode from URDF file

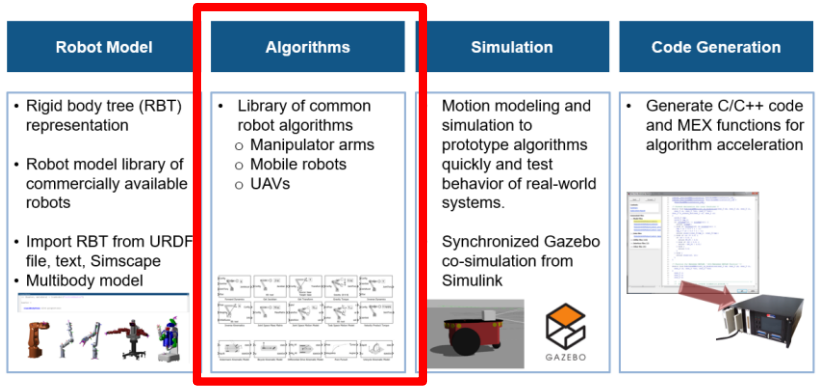
```
>> robot = loadrobot("kinovaGen3")

robot =

  rigidBodyTree with properties:

     NumBodies: 8
        Bodies: {1×8 cell}
          Base: [1×1 rigidBody]
     BodyNames: {1×8 cell}
      BaseName: 'base_link'
       Gravity: [0 0 0]
    DataFormat: 'struct'
```



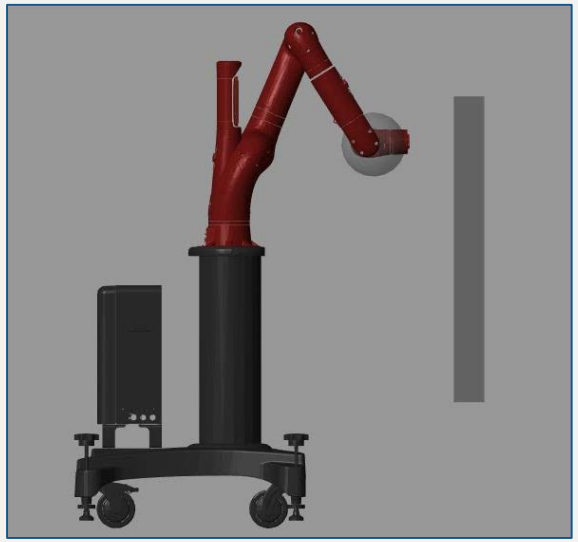


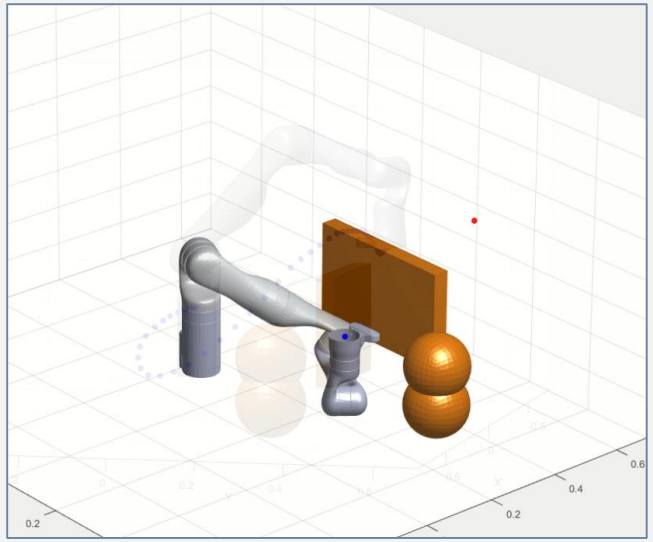


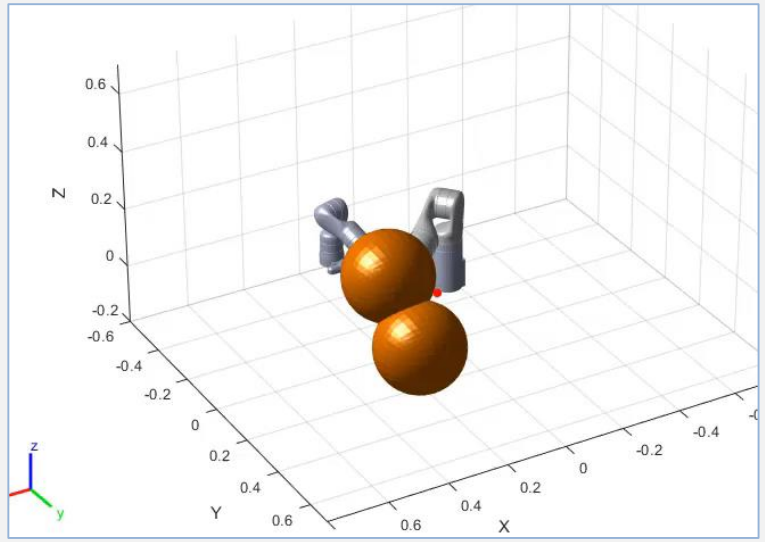
Algorithms - Manipulation



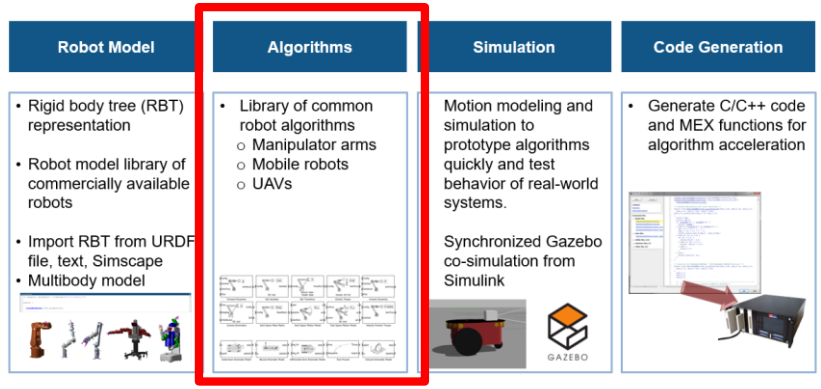
- Forward and inverse kinematics
- Generalized inverse kinematics & constraints
- Forward and inverse dynamics
- Trajectory generation
- Collision checking



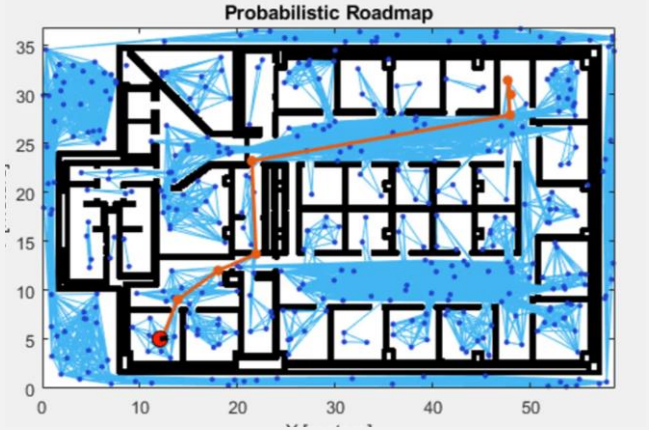


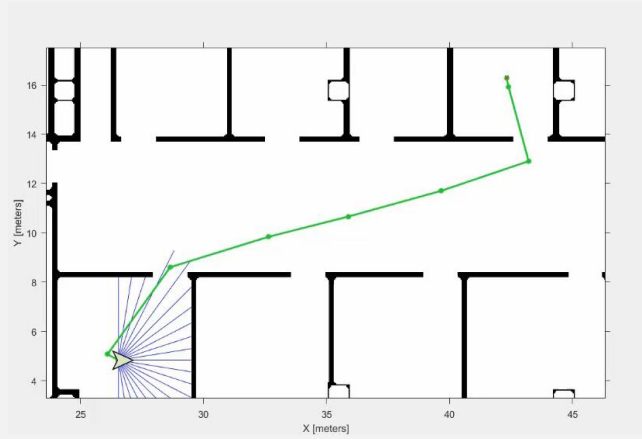


Algorithms – Mobile Robots

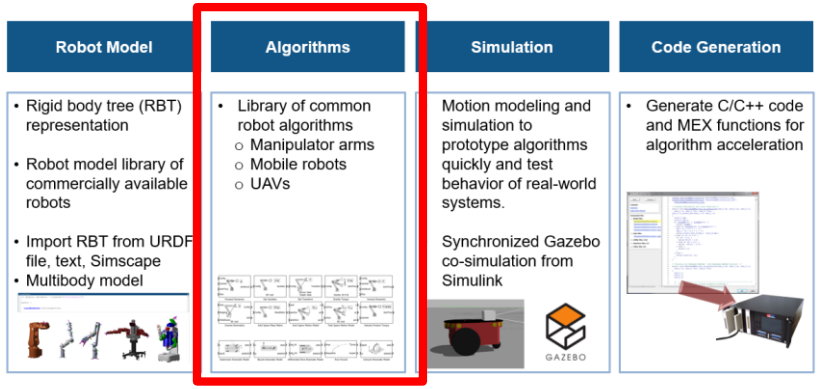


- Mapping and map representation
  - Binary occupancy grid
- Localization
  - Odometry
  - stateEstimatorPF
- Path planning
  - Probabilistic roadmap (PRM)
- Path following
  - Pure pursuit

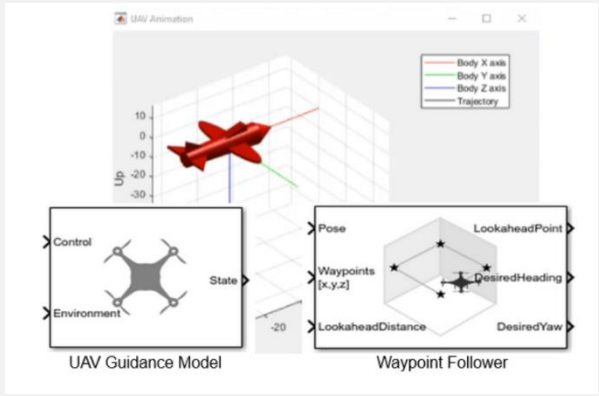


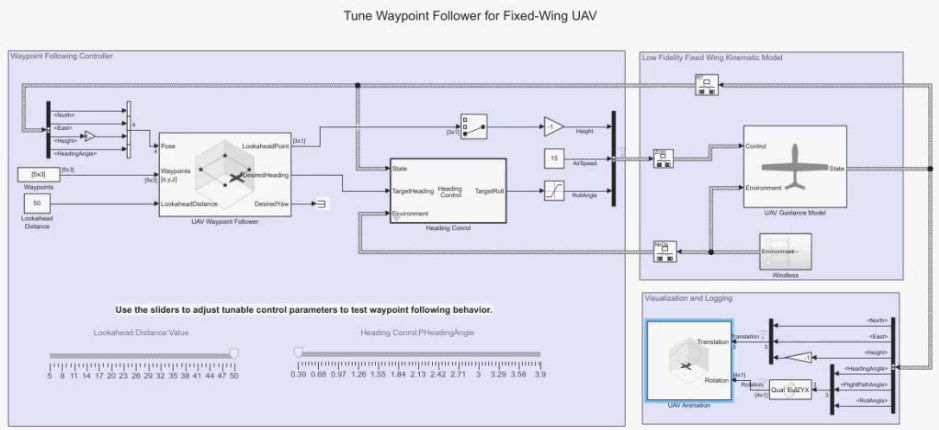


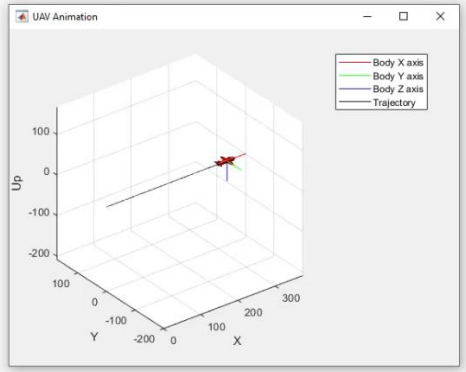
Algorithms – UAVs (Add-On Library)



- Guidance models
  - Reduced-order guidance model for fixed-wing and multi-rotor UAVs
- MAVLink communication
  - Communicate with simulated/physical UAV
  - Import and analyze UAV flight logs
- Waypoint following
  - Execute flight missions based on given waypoints

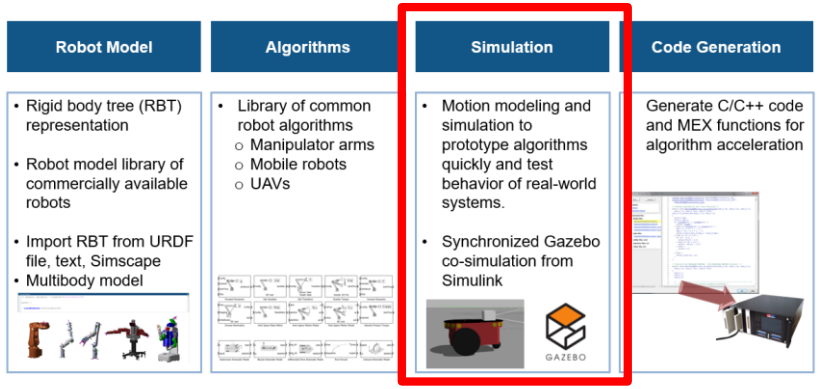


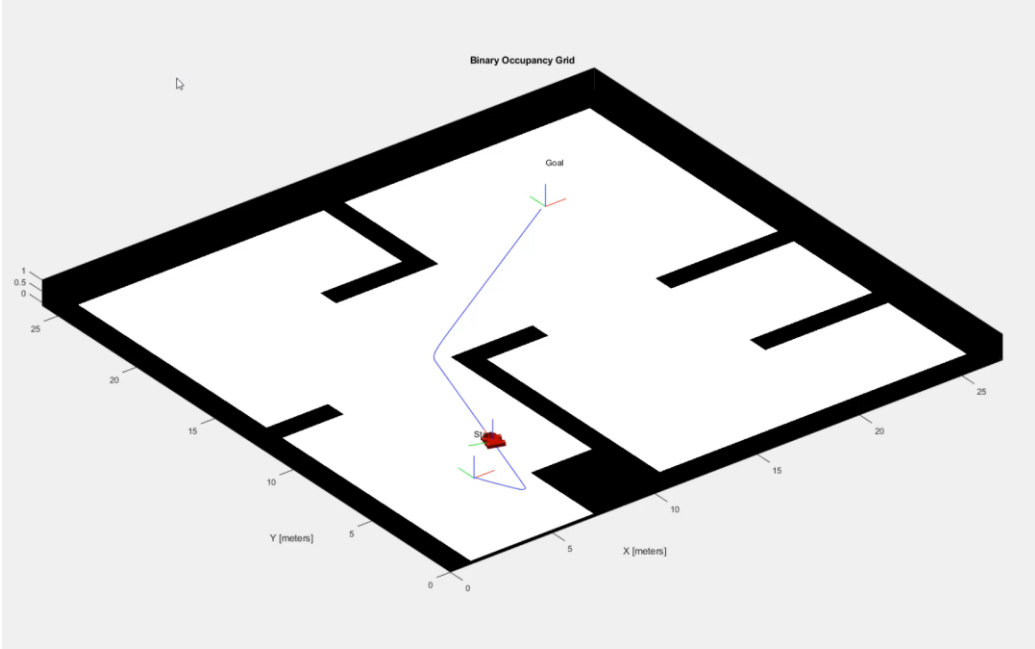


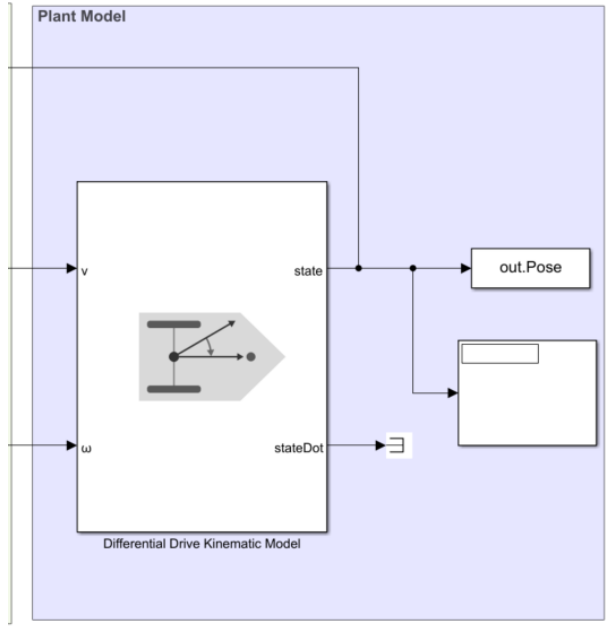


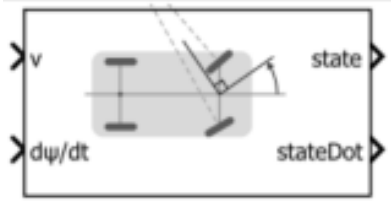
Simulation

- Low-fidelity simulation

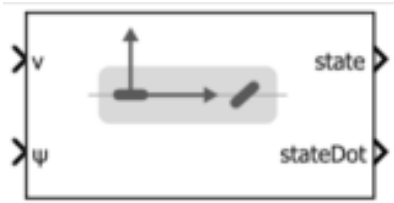




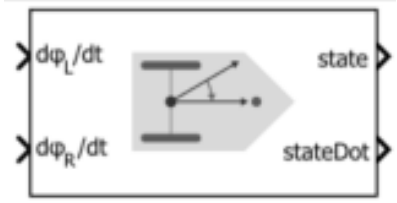




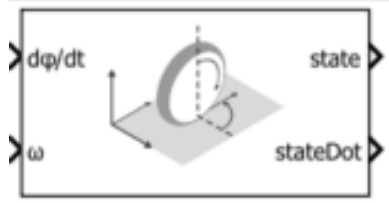
Ackermann Kinematic Model



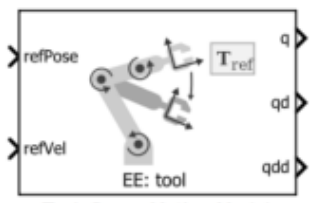
Bicycle Kinematic Model



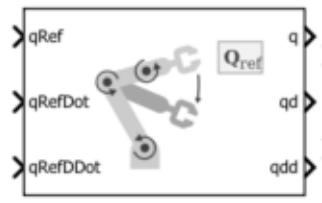
Differential Drive Kinematic Model



Unicycle Kinematic Model

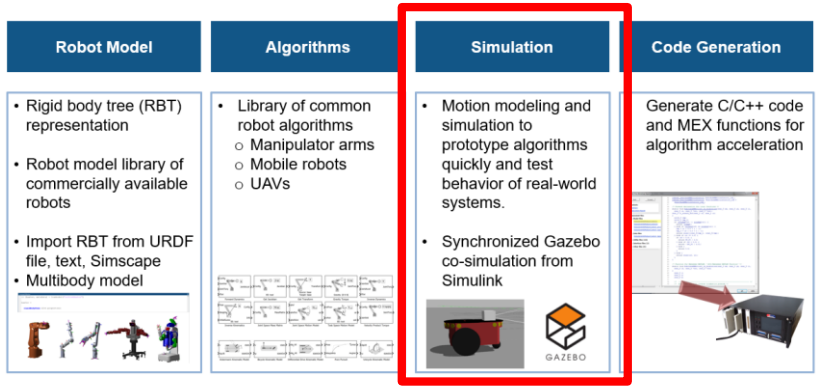


Task Space Motion Model

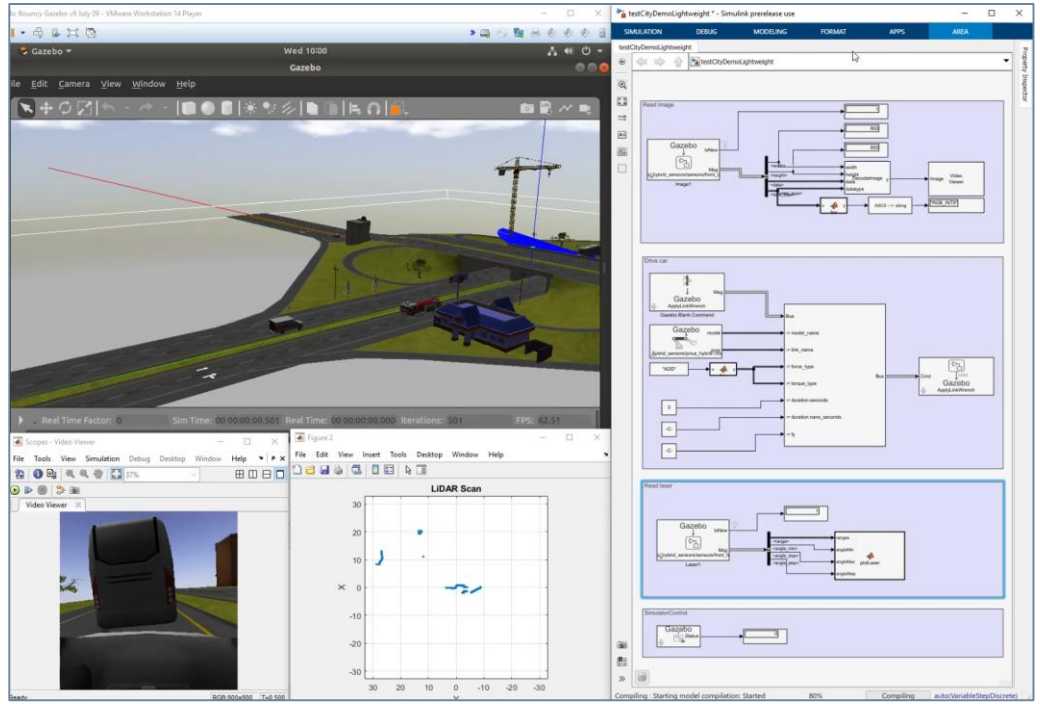


Joint Space Motion Model

Simulation

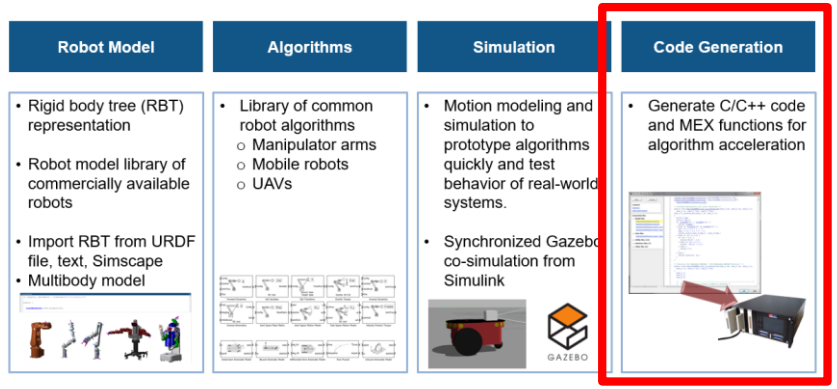


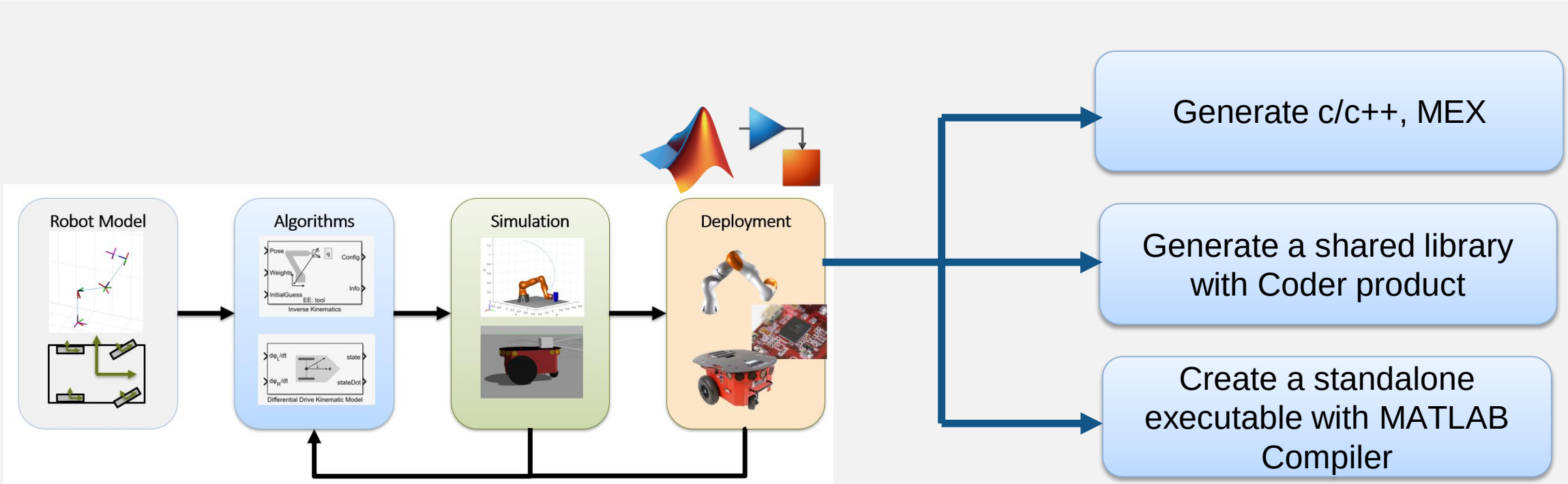
- Gazebo Co-simulation
  - Provides synchronized stepping between Simulink and Gazebo simulator
  - Retrieve sensor data and ground truth pose for models from Gazebo simulator
  - Actuate model links and joints in Gazebo simulator



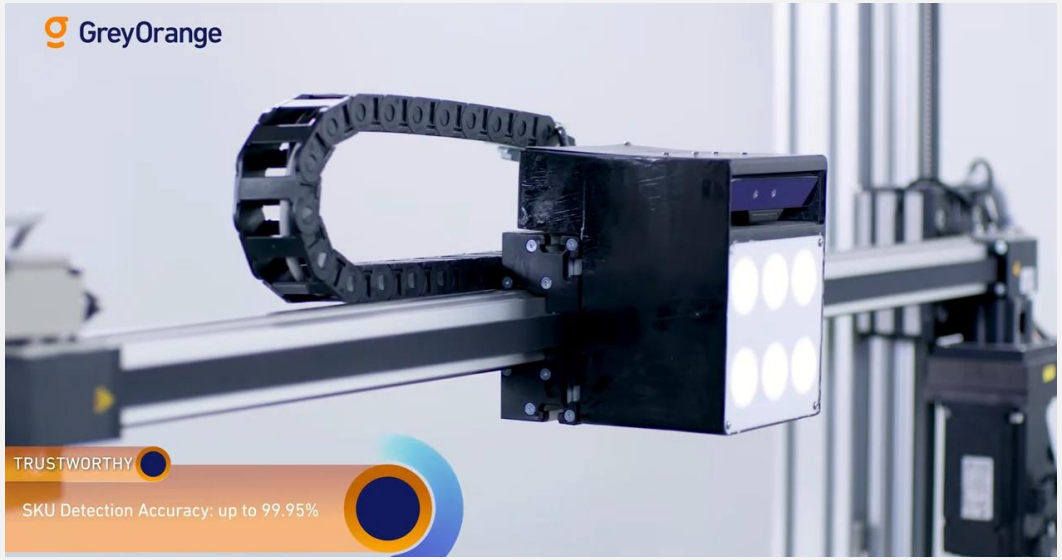
Deployment

- Accelerate robotics algorithms with code generation

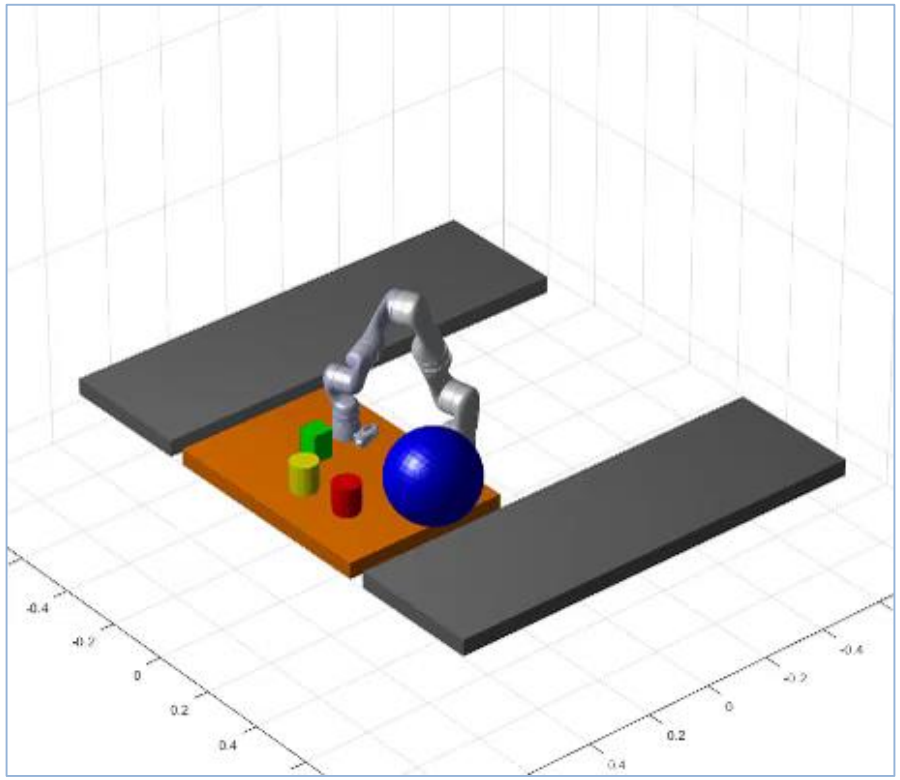




Example – Pick-and-Place Robot Arm



- Applications in **warehouses**, **manufacturing**, and **medical industries**
- RST: robot model, plan, control, and simulate robot
- MPC: trajectory optimization
- Stateflow: task-level planning and execution



Navigation is critical for autonomous systems

Qualcomm | Inven / Research / Areas of Research / Autonomous Robotics

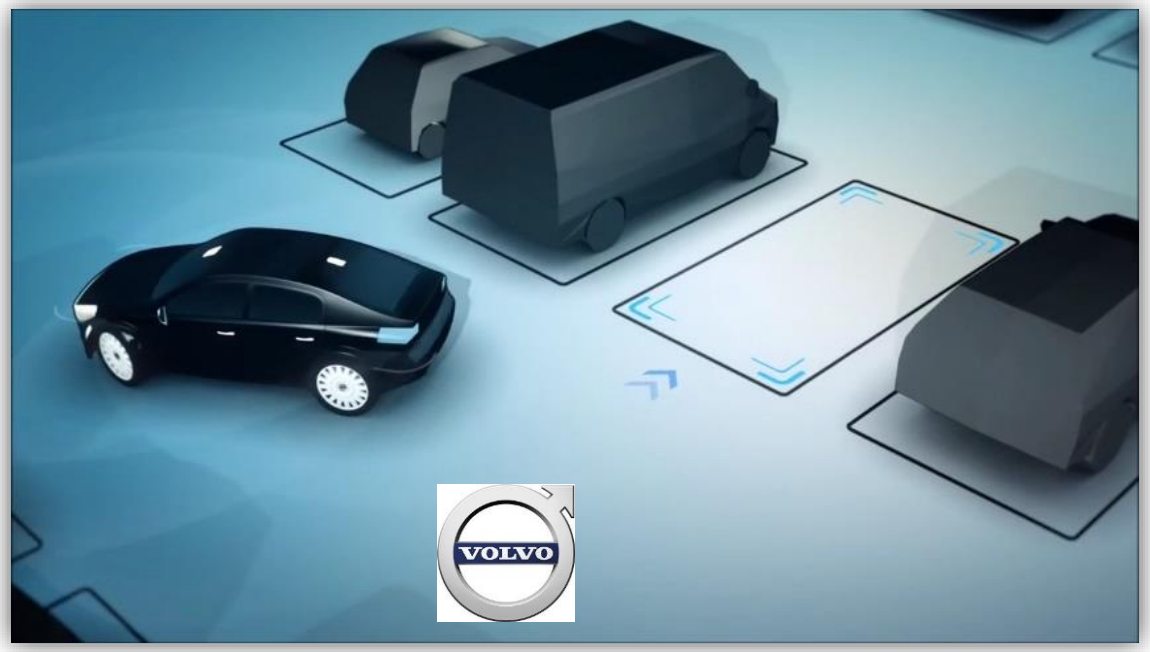
Autonomous Robotics

Overcoming obstacles in autonomous path planning and navigation.

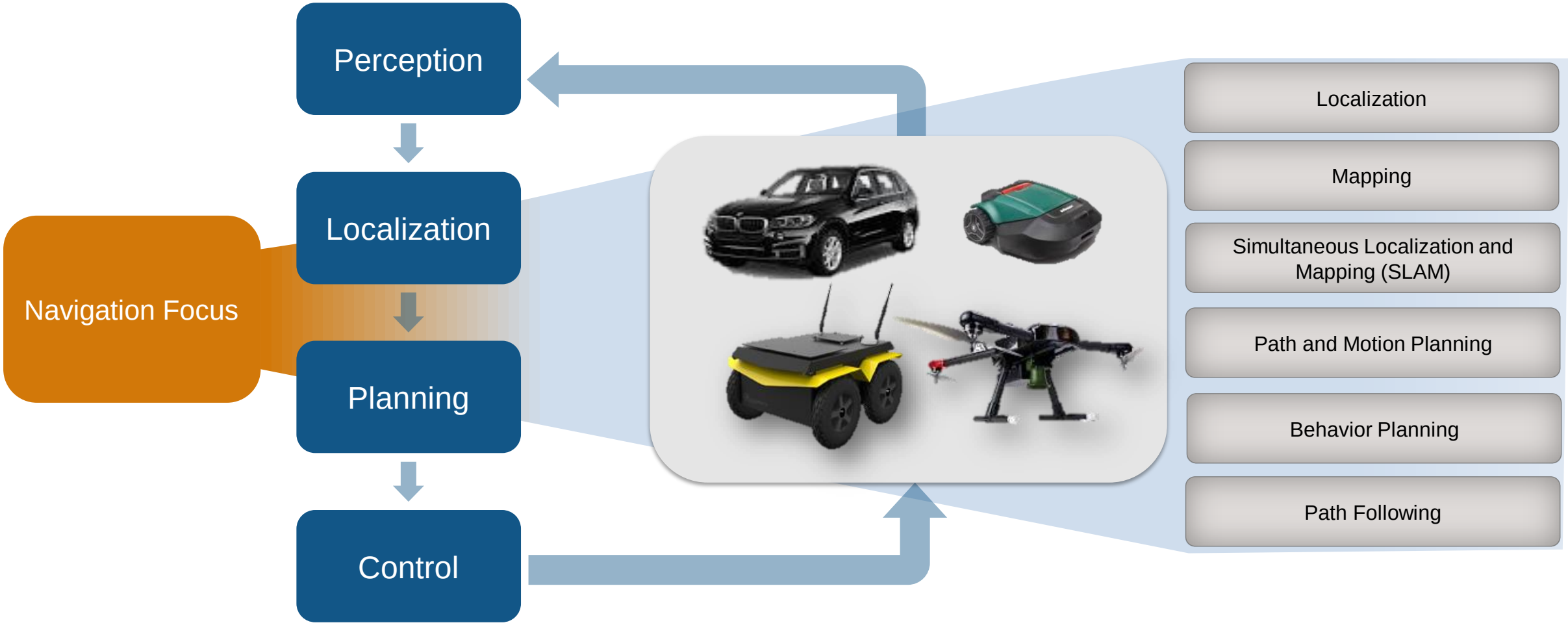
We have developed **autonomous path planning and navigation systems** for drone and robots, enabling them to move safely through indoor and outdoor environments. For example, prior to flight, the user designates where the drone should go and the bounds of the area it will fly through. The drone's path planning algorithm uses a 3D model of the world (generated through voxel mapping) to build a random graph of unoccupied points in space and safely transitions between them. The graph represents all the collision-free paths the drone could select to reach its goal. The drone may see multiple paths but will pick the shortest path to its destination. Every hundred milliseconds, it updates its 3D voxel map and re-checks the planned path to ensure it is still safe. If at some point the drone encounters a potential hazard or obstacle in its path, it will re-vector to an alternate route based upon its internal decision-making

Navigation Tools

- *Where am I going?* **Behavior Planning**
- *What's the best way there?* **Path / Motion Planning**
- *Where have I been?* **Mapping**
- *Where am I on map?* **Localization**
- *What if you don't have a map?* **SLAM**

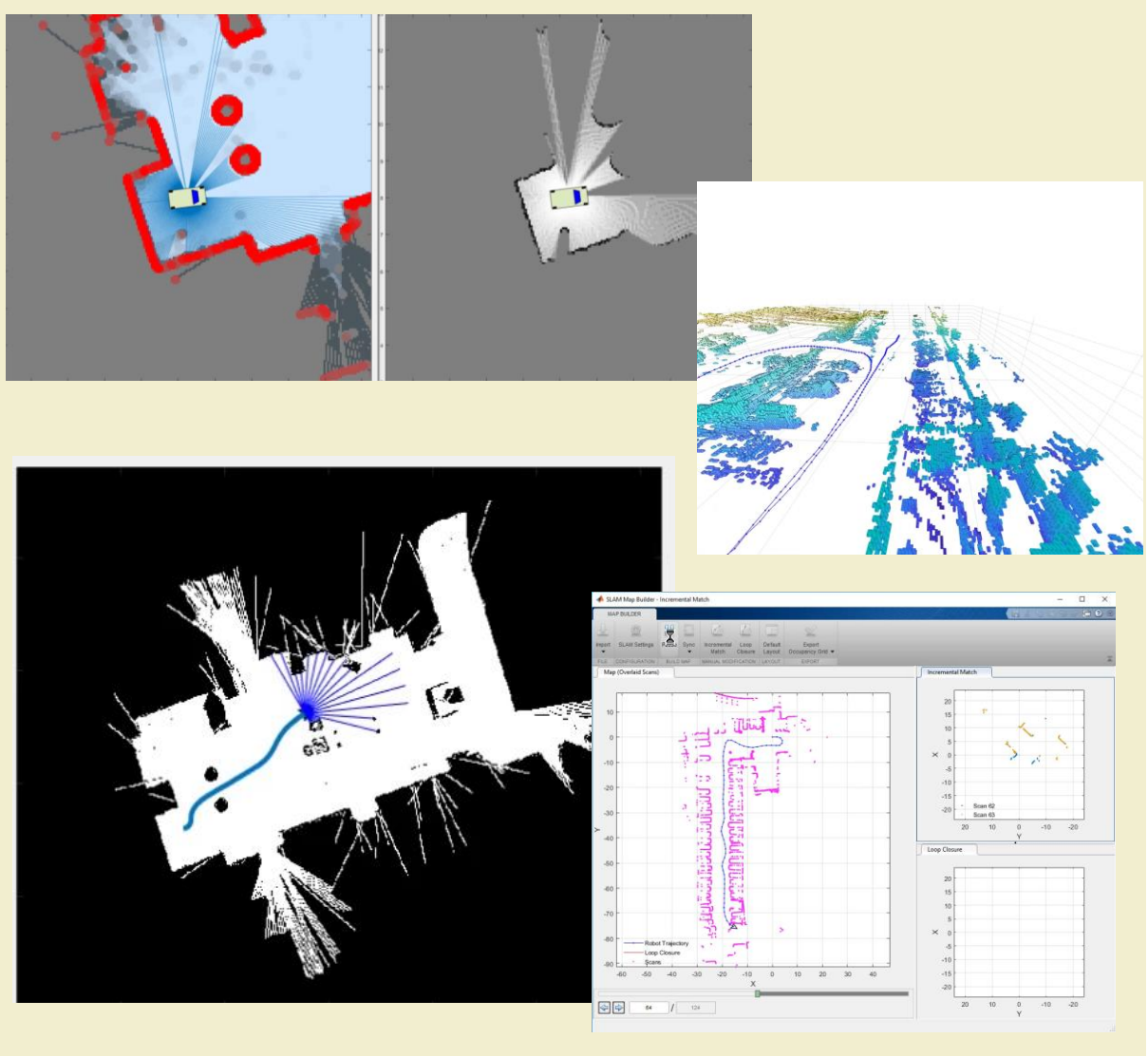


Autonomous Navigation Workflow



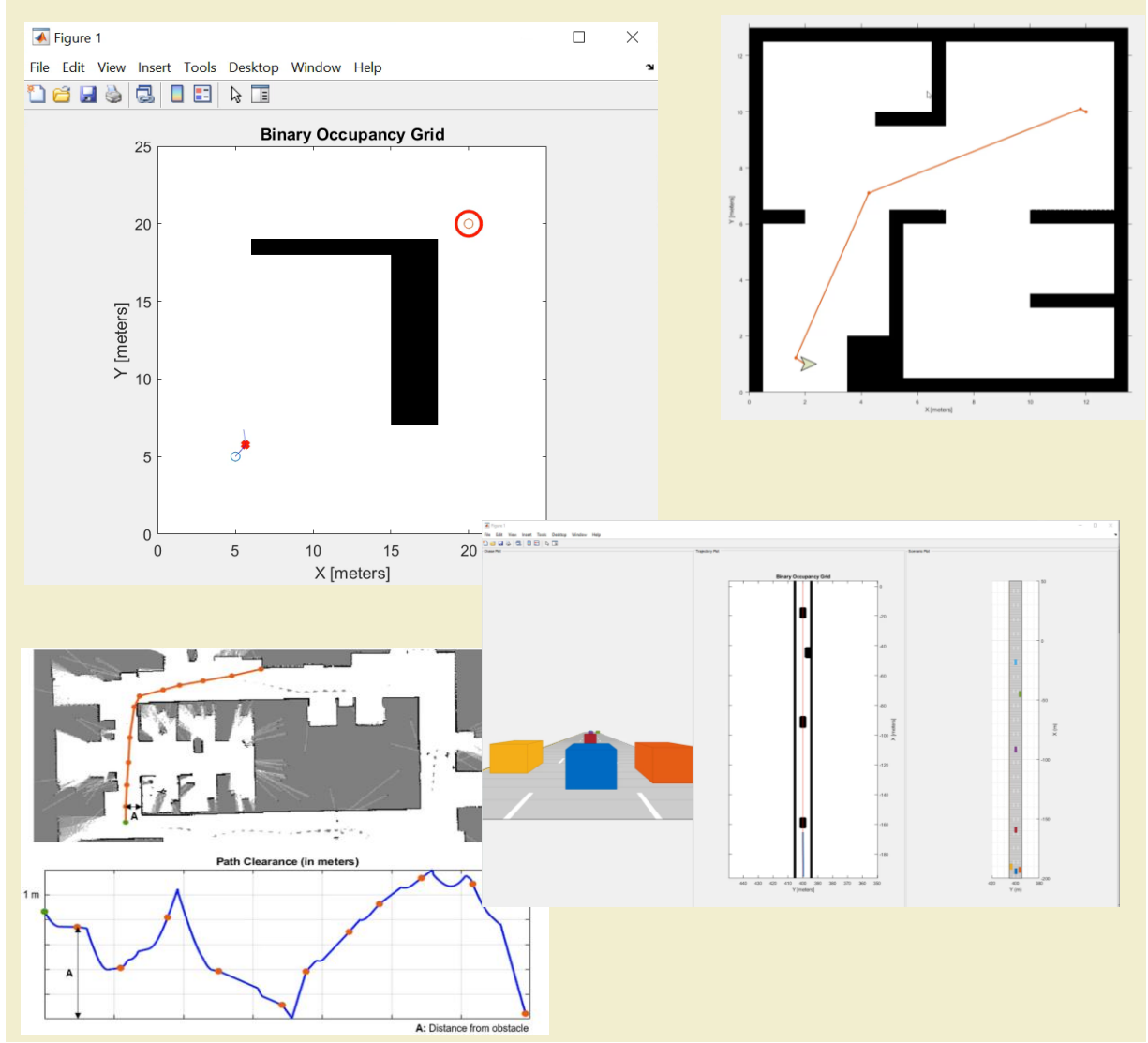
Navigation Toolbox

- Mapping and localization
  - 2D and 3D SLAM
  - Egocentric maps
  - SLAM map builder App



Navigation Toolbox

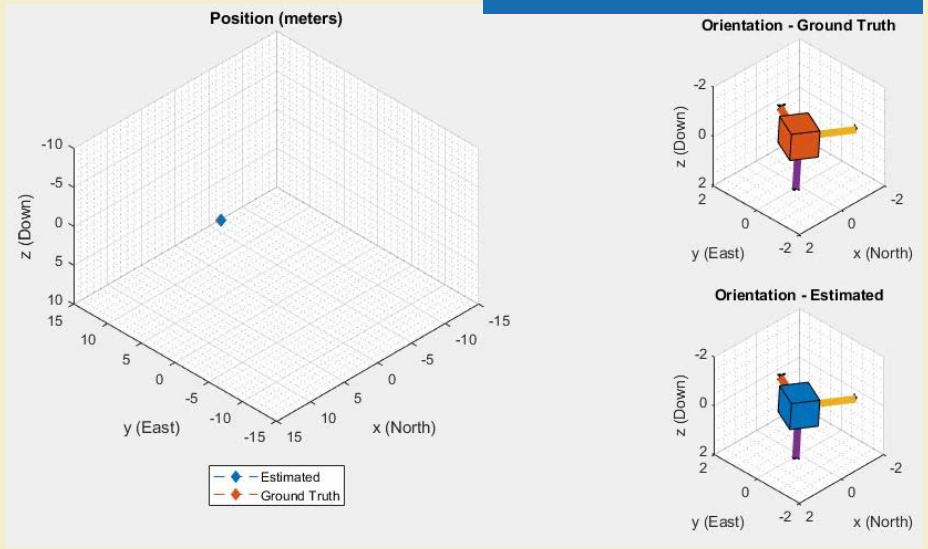
- Mapping and localization
  - 2D and 3D SLAM
  - Egocentric maps
  - SLAM map builder App
- **Path planning and Following**
  - **Algorithms for path planning**
  - **Planner interface**
  - **Path metrics**
  - **Path following and controls**



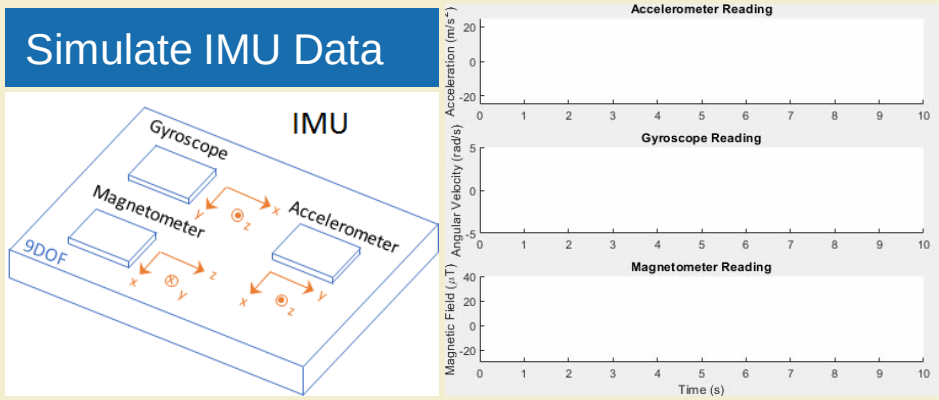
Navigation Toolbox

- Mapping and localization
  - 2D and 3D SLAM
  - Egocentric maps
  - SLAM map builder App
- Path planning and Following
  - Algorithms for path planning
  - Planner interface
  - Path metrics
  - Path following and controls
- **Sensor modeling and simulation**
  - **IMU, GPS, INS sensors**

IMU + GPS Fusion

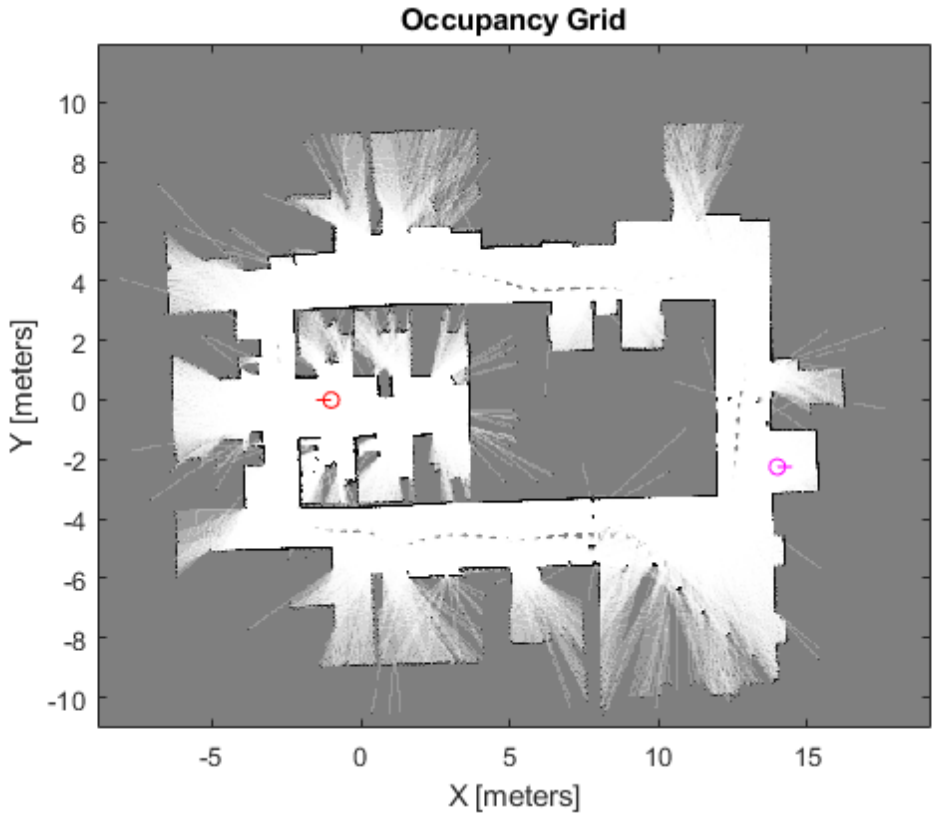


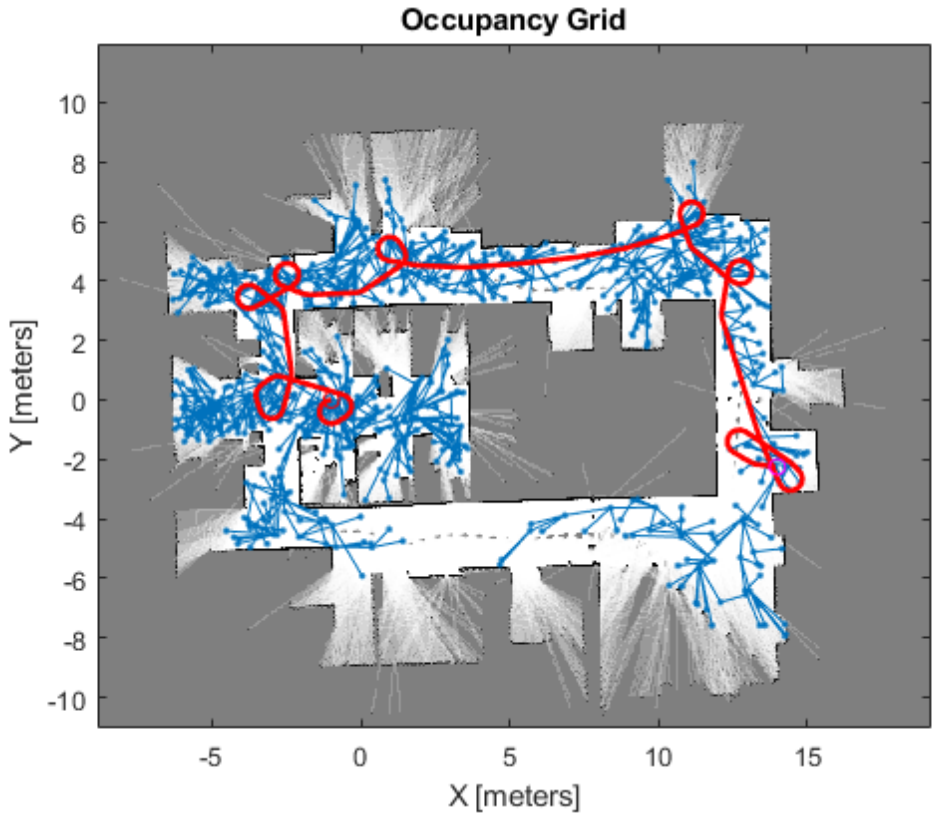
Simulate IMU Data



Example – Plan Mobile Robot Paths using RRT

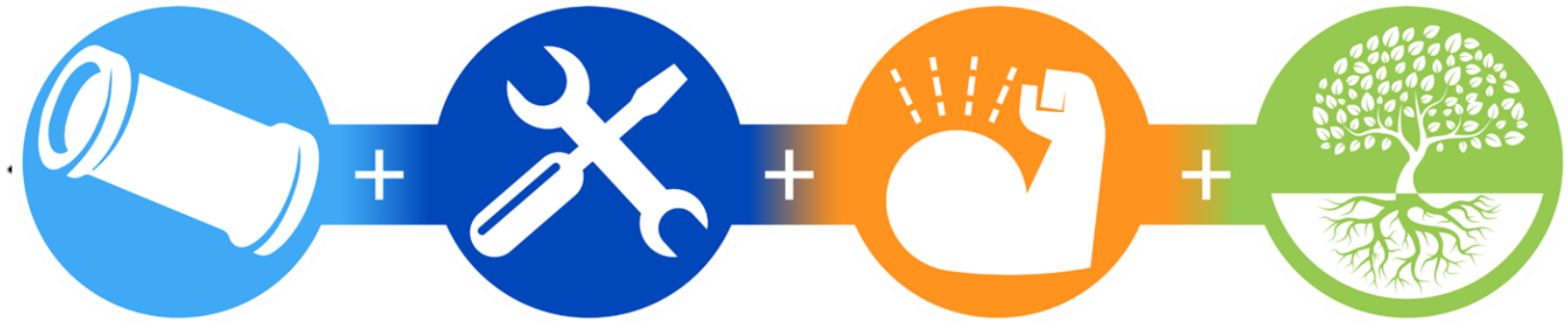
- Load an existing occupancy map of a small office space
- Specify the state space of the vehicle
- Plan a path for a vehicle





ROS – A Distribution in Software for Automation

- Open Source
- Established to prevent re-inventing the wheel
- Maintained by Open Robotics
- Reusable Software Components
- >1,000,000 user downloads/mo



Why ROS? Growth and Adoption of ROS

ROS

10 years huge growth

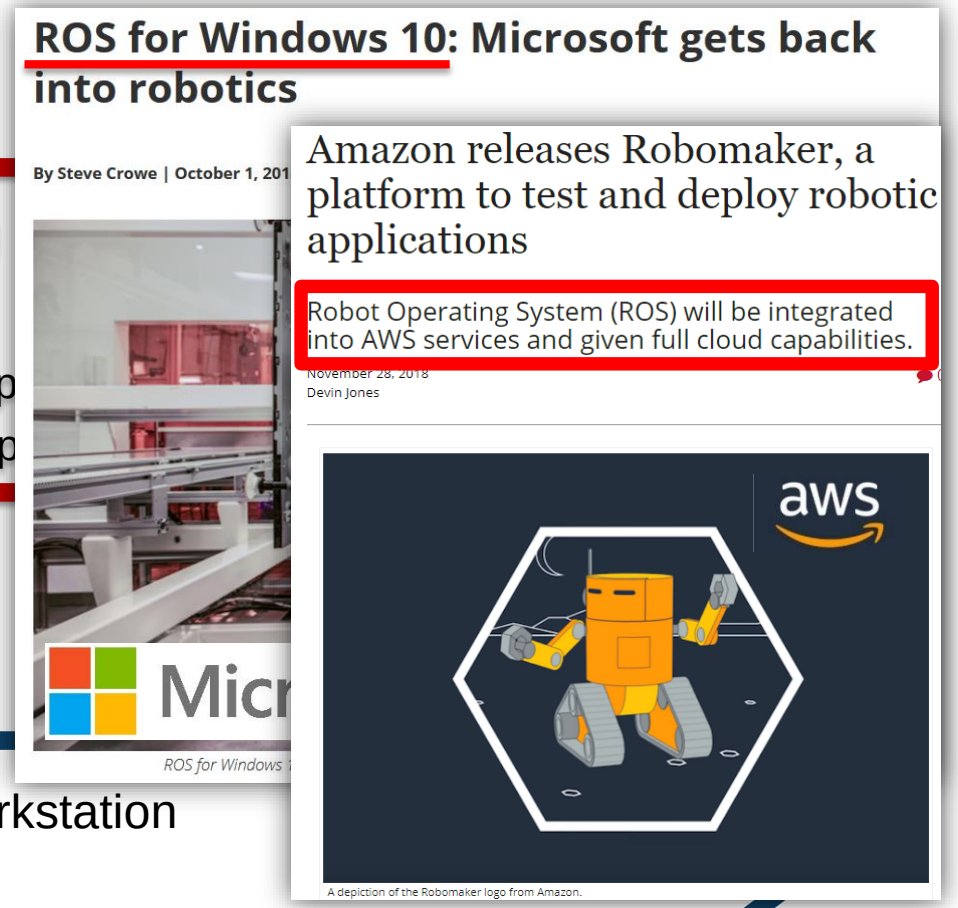
- Plenty of development tools
- Active community (ROS wiki p
- 5,000 packages, 18,000 wiki p

ROS1_Bridge

Completely Re-Written

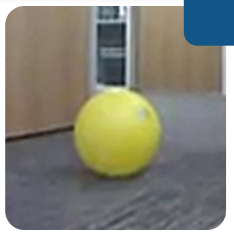
2

- Run on range of systems: embedded to workstation
- For use in real-time systems
- For safety- and mission-critical applications and productions

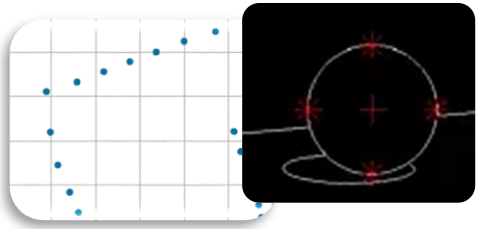


Development Workflow for ROS-based Applications

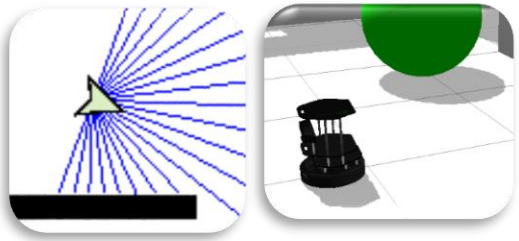
```
>> rostopic list
/mobile_base/commanc
/mobile_base/sensors
/odom
/rosout
/scan
```



Explore ROS Network Interface



Develop Algorithm

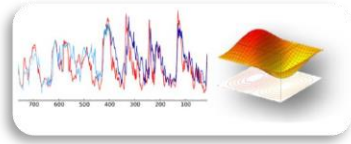


Test and Refine in Simulation

Deploy a ROS node to the Robot

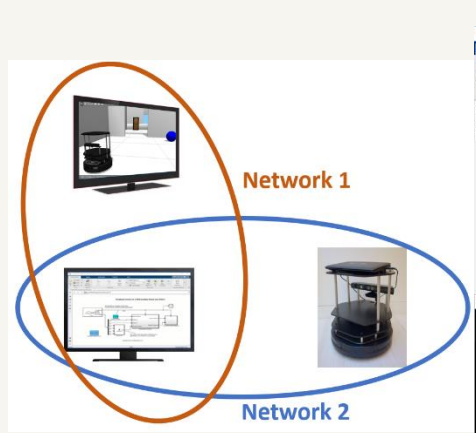
Test and Refine on Real Robot

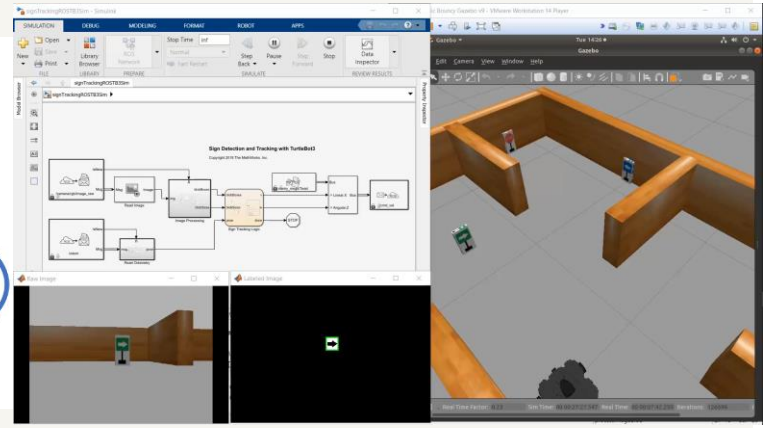
Performance Analysis

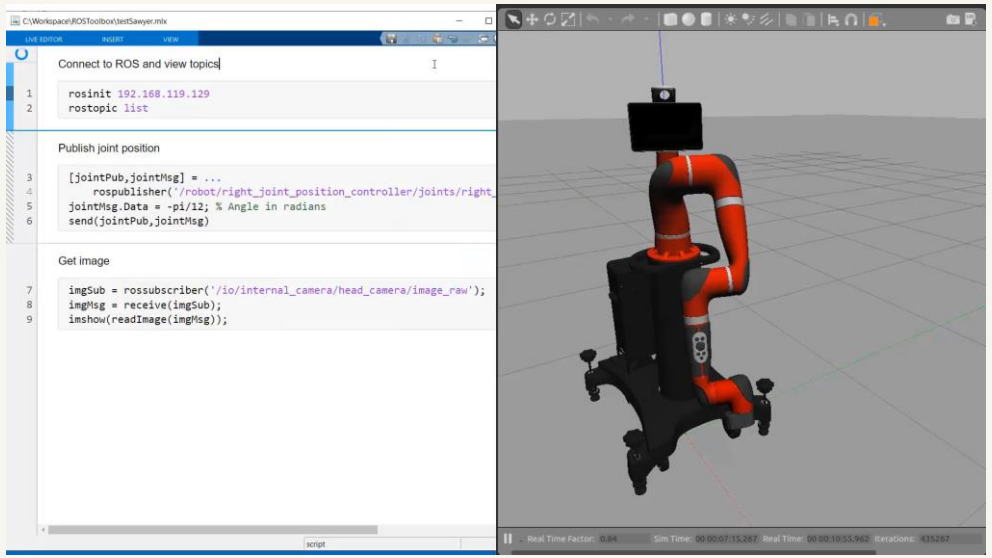


ROS Toolbox

- **ROS network and communication**
  - **Live connectivity from MATLAB and Simulink to ROS and ROS2**

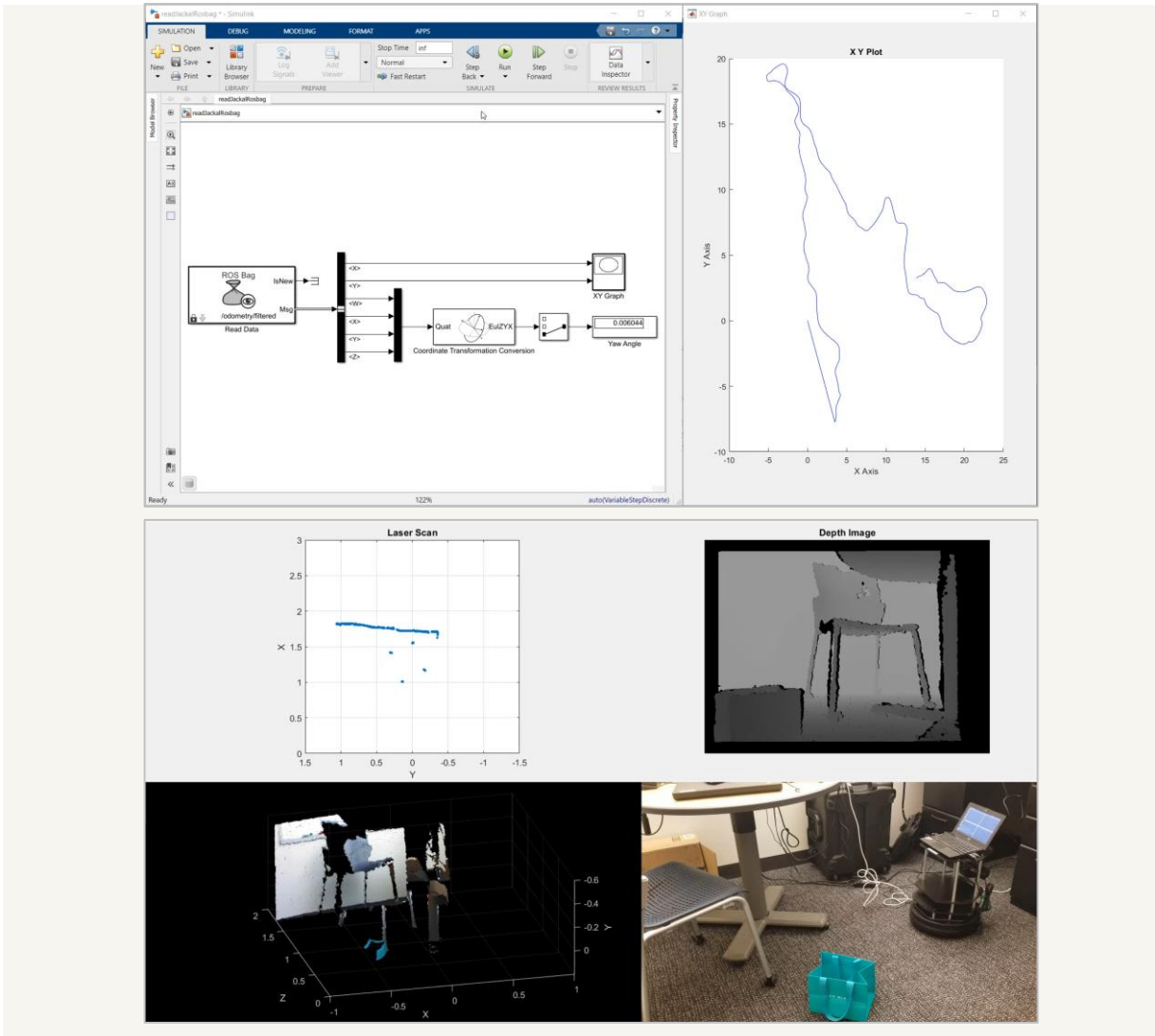






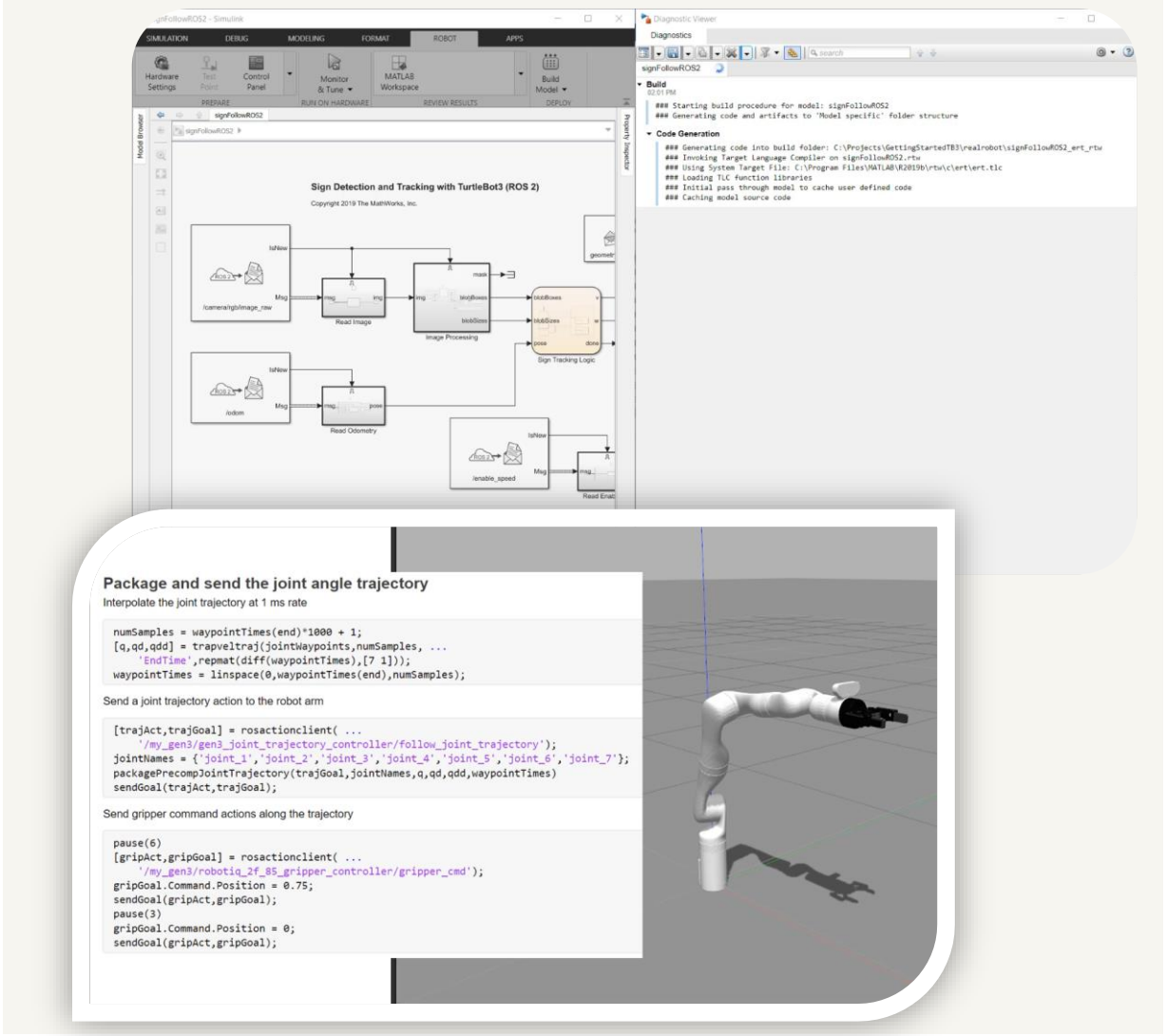
ROS Toolbox

- ROS network and communication
  - Live connectivity from MATLAB and Simulink to ROS and ROS2
- **ROS Message**
  - **rosbag data import and playback**
  - **Specialized ROS message**

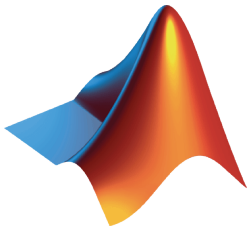
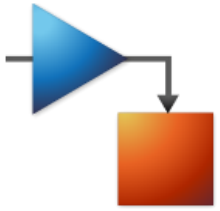


ROS Toolbox

- ROS network and communication
  - Live connectivity from MATLAB and Simulink to ROS and ROS2
- ROS Message
  - rosbag data import and playback
  - Specialized ROS message
- **ROS node generation**
  - **Node generation from Simulink for prototyping and deploying autonomous systems**



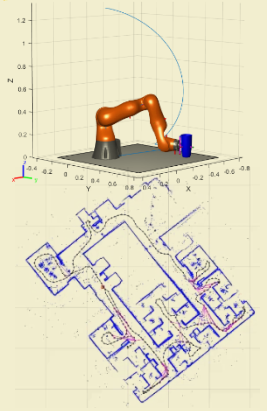
MATLAB/Simulink ROS Functionality

	ROS	ROS 2	ROS Common
	• Topic – Pub / Sub • Service – Server / Client • Action – Client • Parameter Server – Get/Set • Custom Message • rosbag read	• Topic – Pub / Sub • Custom Message	• Read Data • Read / Write Image • Read Point Cloud • Read Occupancy Map
	• Topic – Pub / Sub • Service – Call • Parameter – Get / Set • ROS Time • rosbag playback • Code Generation	• Topic – Pub / Sub • Code Generation	• Read Data • Read Image • Read Point Cloud

ROS Toolbox enables you to communicate with a ROS

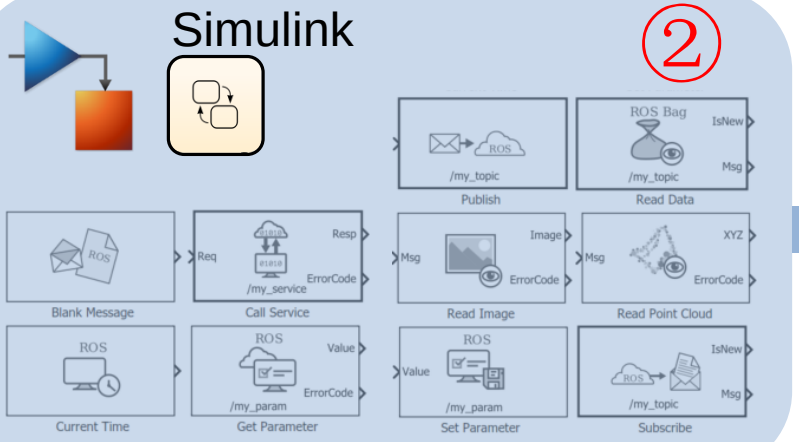
Traditional ROS users

MATLAB ①



```
ROS (Robot Operating System)
  rosinit          - Initialize the ROS system
  rosshutdown      - Shut down the ROS system

  rosmessage       - Create a ROS message
  rospublisher     - Create a ROS publisher
  rossubscriber    - Create a ROS subscriber
  rossvcclient     - Create a ROS service client
  rossvcserver     - Create a ROS service server
  rosactionclient  - Create a ROS action client
  rostype          - View available ROS message t

  rosaction        - Get information about actio
  rosmsg           - Get information about messa
  rosnode          - Get information about nodes
  rosservice       - Get information about servic
  rostopic         - Get information about topic

  rosbag           - Open and parse a rosbag log
  rosparam         - Get and set values on the pa
  rosrate          - Execute fixed-frequency loo
  rostf            - Receive, send, and apply ROS

  rosduration      - Create a ROS duration obje
```

Simulink ②

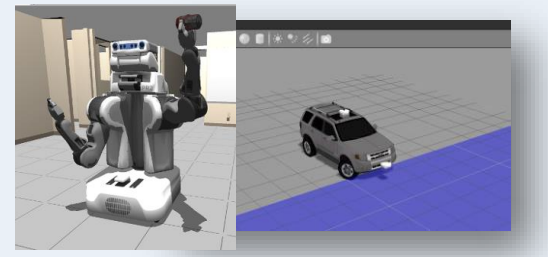


rosbag import ③

ROS 2

C++ Code Generation & Auto Deployment ④

Robot hardware & Sensors



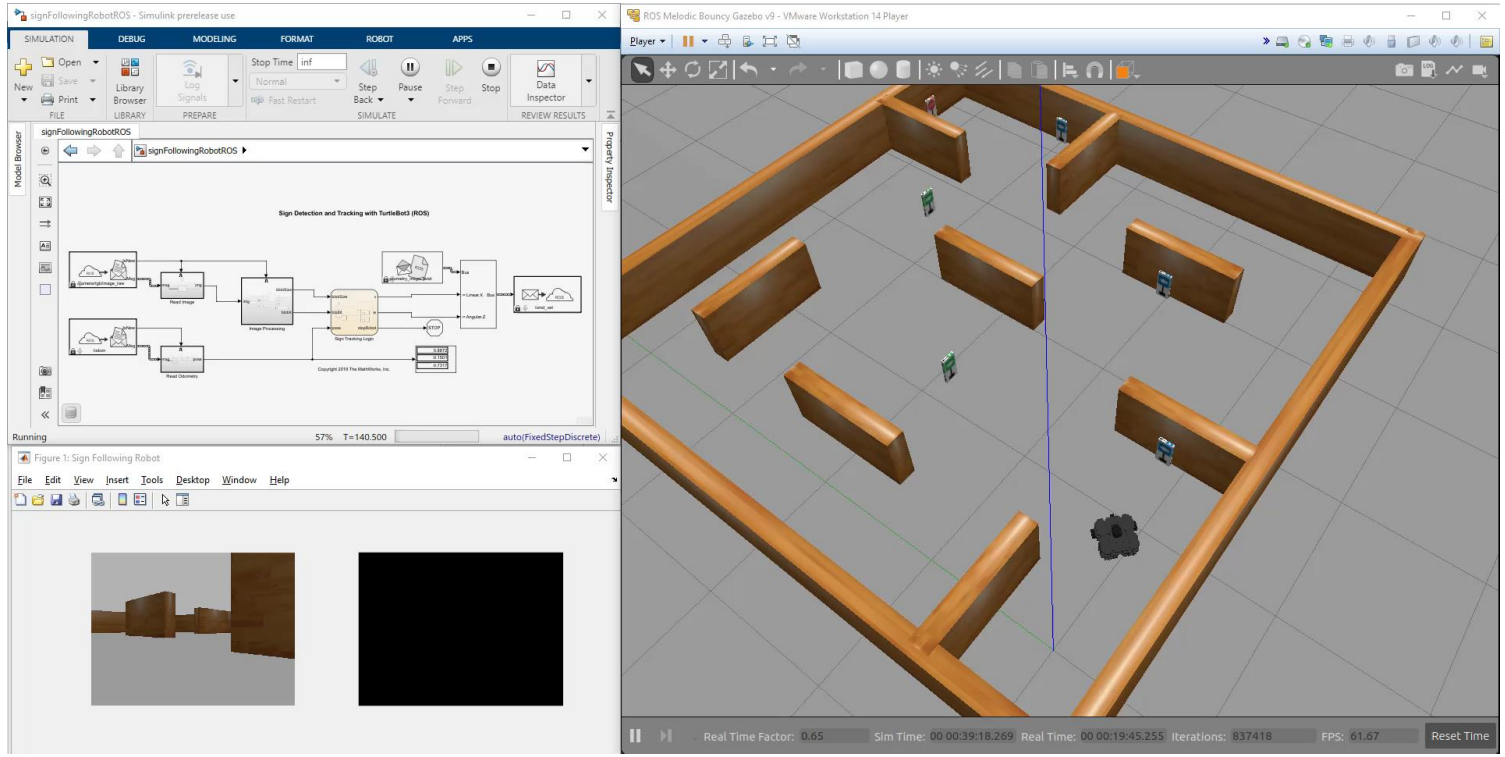
Simulation environment

Generated ROS node

Embedded targets

Example – Sign-following Robot

- Detect the color of the sign and send the velocity commands to turn the robot
- Connect with ROS-enabled simulator, i.e., Gazebo
- And connect with hardware



Thank you