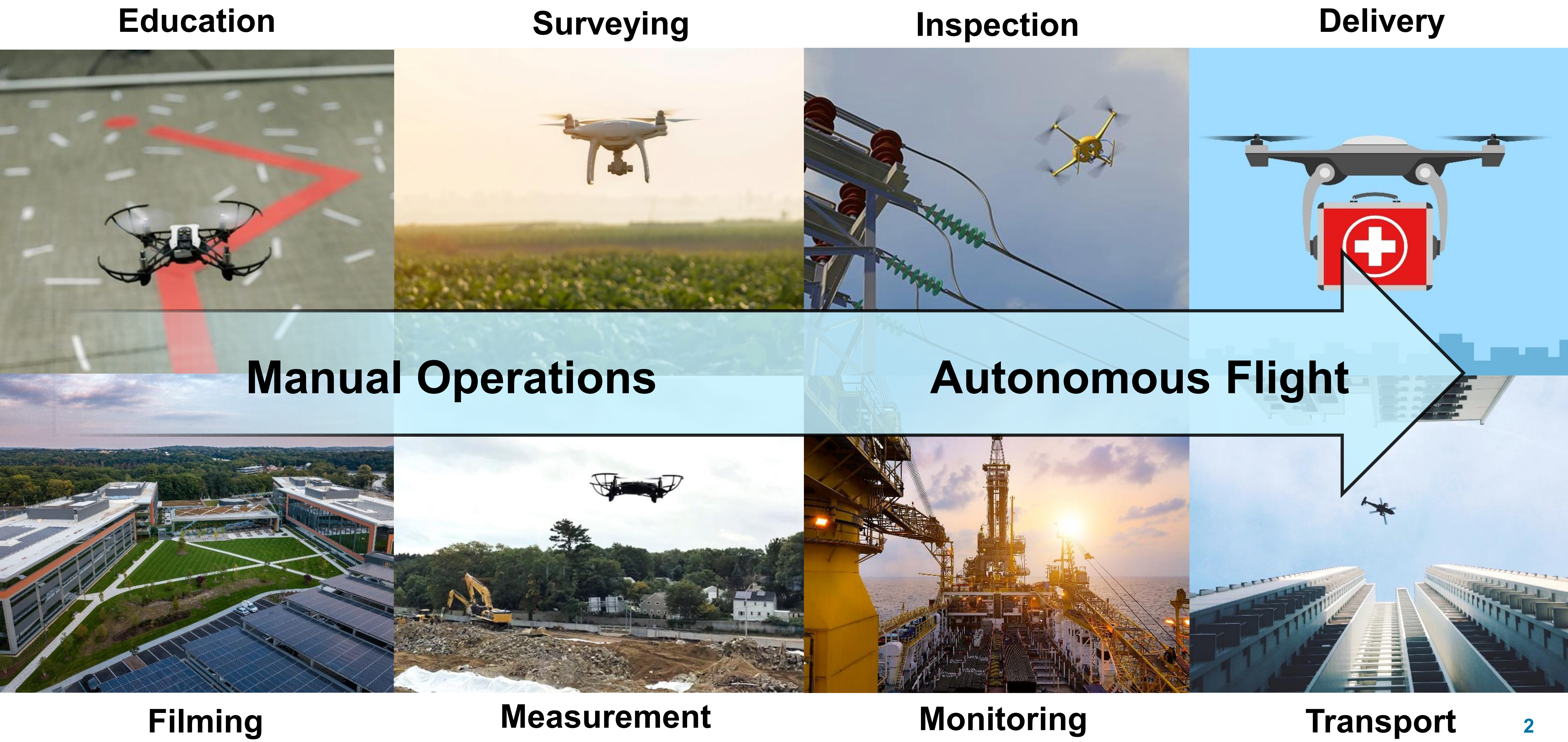
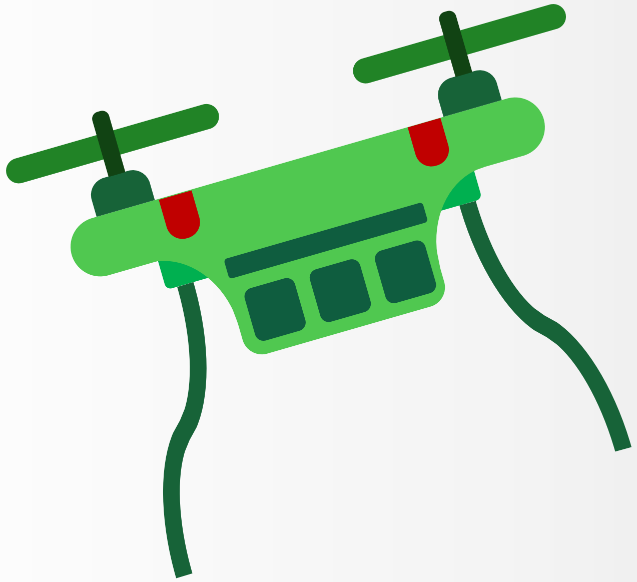
Enhancing GNC Through Intelligent Perception and Positioning

Andreas Andersson
Senior Technical Consultant

Increase in UAV Usage



Challenges in developing autonomous UAV systems & applications

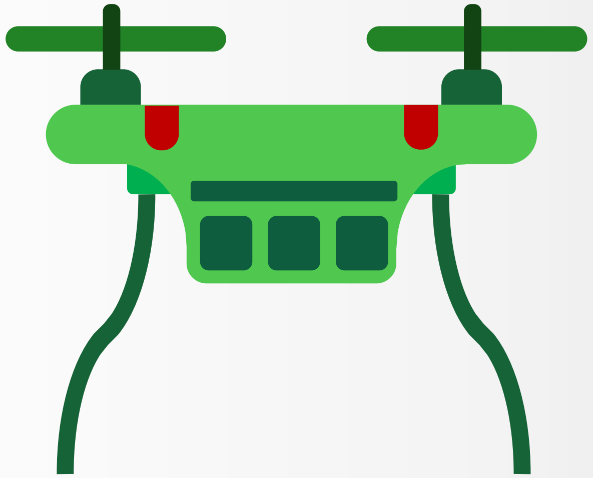


Complexity of advanced autonomous algorithms

Need of end-to-end workflows

Ensuring system quality and reducing flight risk

Solutions for developing autonomous UAV systems & applications



Robust tools and features for designing and testing UAV systems and algorithms

Integrated development environment that covers development from ideas to production

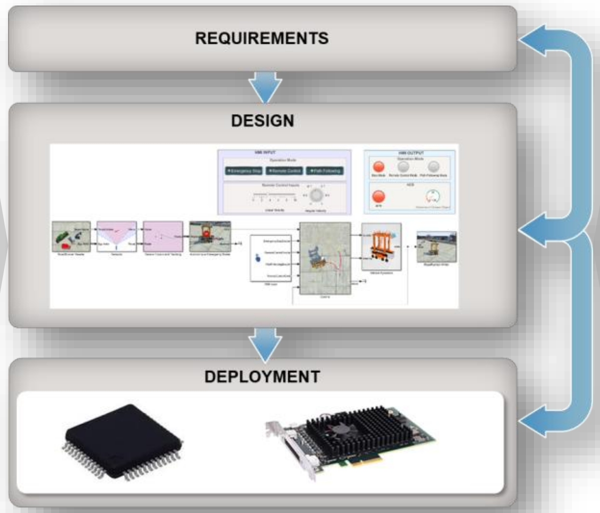
Extensive verification and validation tools to evaluate design quality through virtual testing

MathWorks Supports Autonomous Vehicles Development

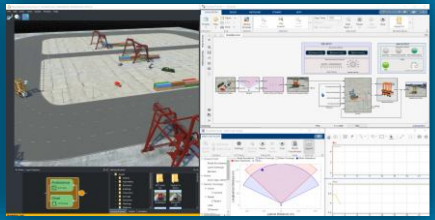
Design, Simulate, Test, and Deploy

- Systems Engineering
- Sensing
- Perception
- Planning
- Control
- Platform
- Deploy
- Connect
- V&V

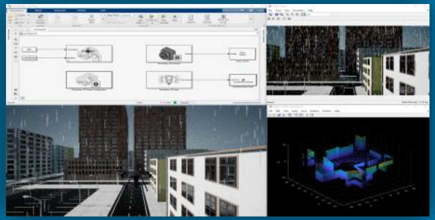
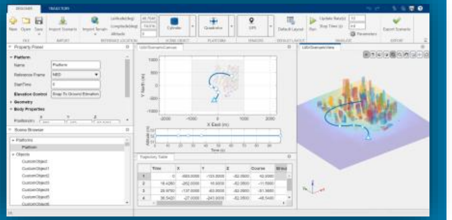
Model-Based Design



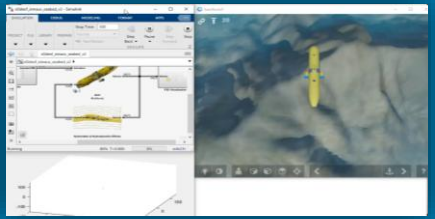
Ground Vehicle



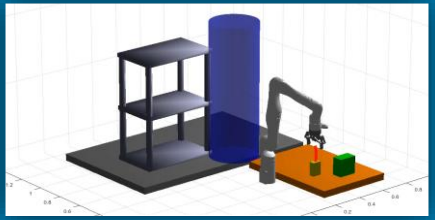
UAV

Marine Robot



Manipulator

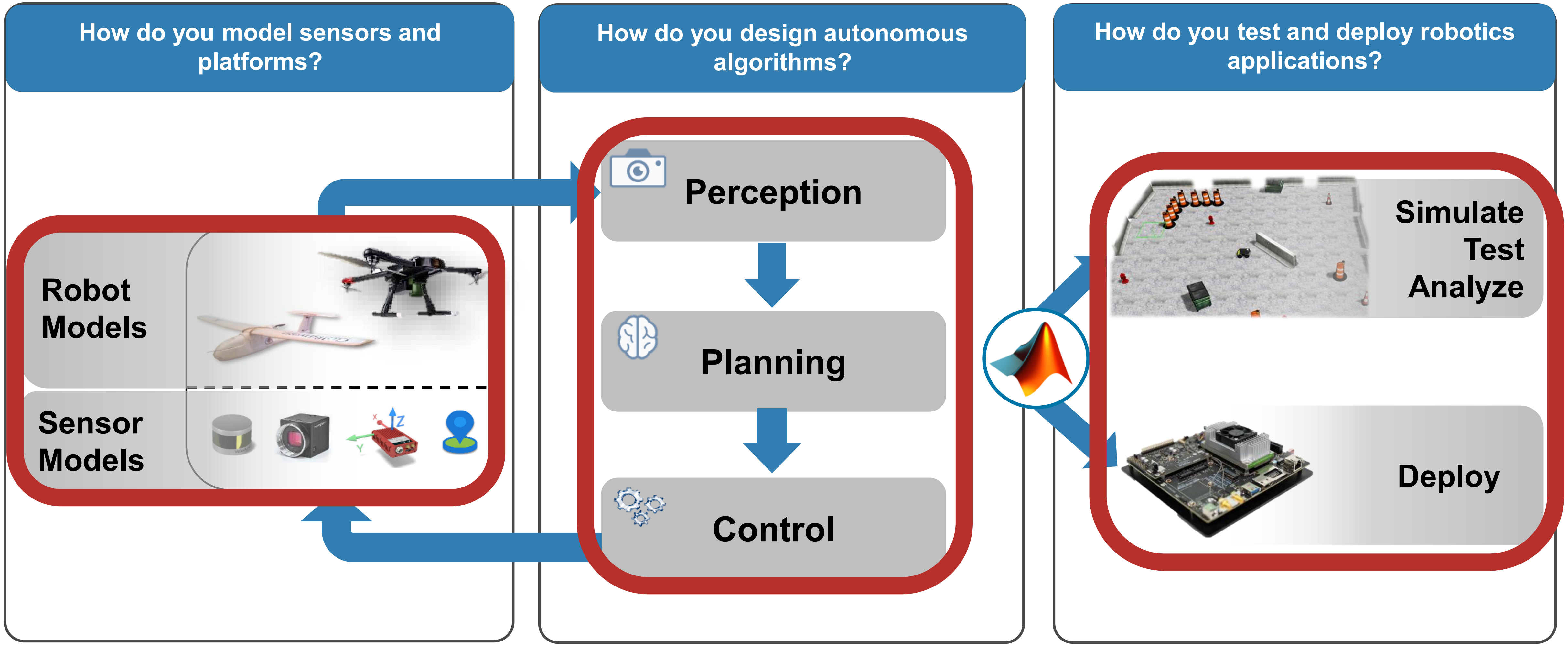


Common questions from customers

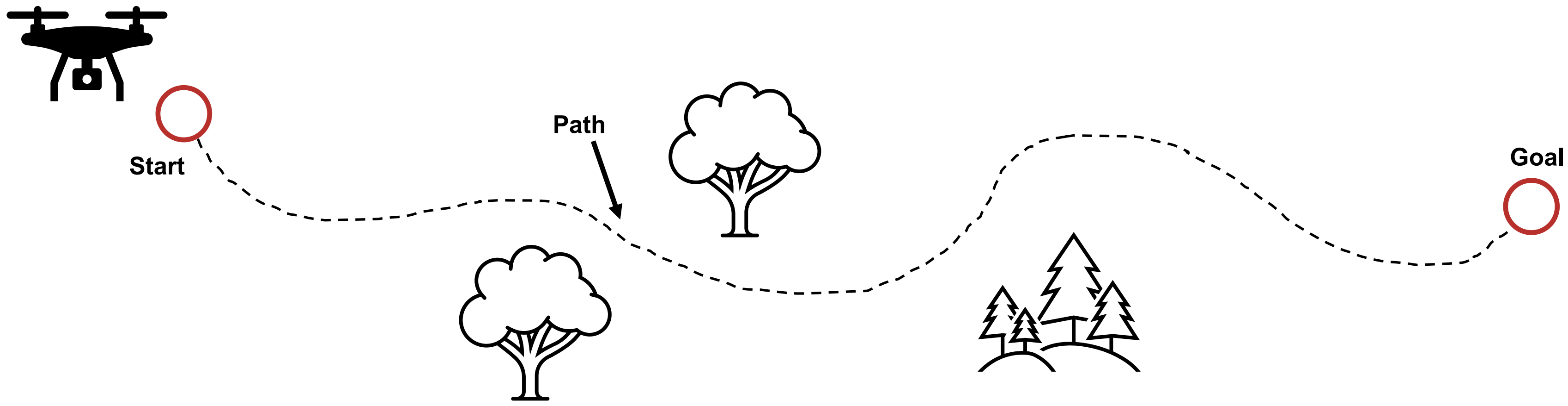
How do you model sensors and platforms?

How do you design autonomous algorithms?

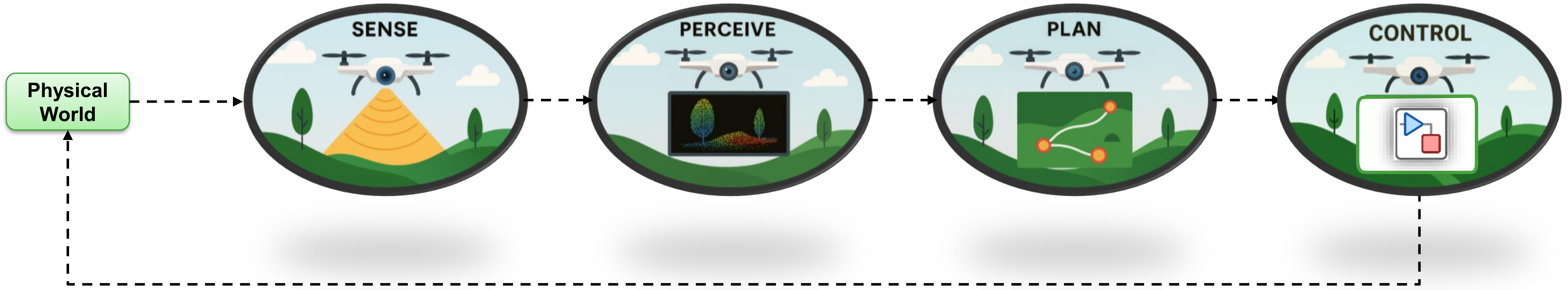
How do you test and deploy robotics applications?



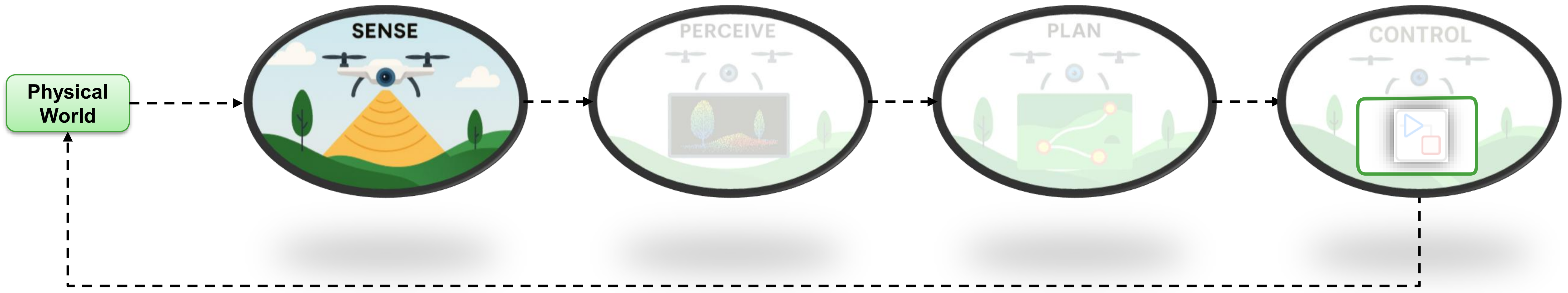
Perception, Planning, Control



Core of Autonomous Navigation



Core of Autonomous Navigation



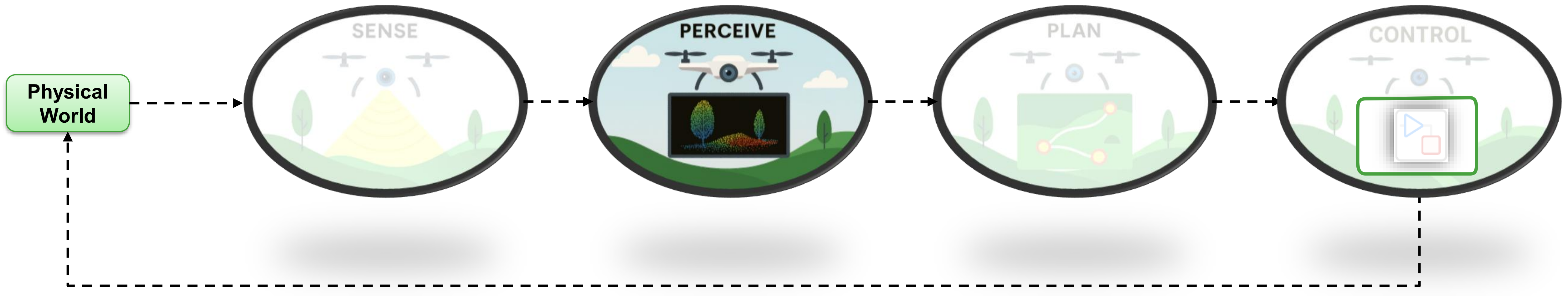
Definition: *Collect accurate and timely information about the surroundings and the system's own state.*

Goal: Gather information without interpretation.

Type of sensors:
- Lidar returns point clouds
- Camera captures pixel arrays
- GPS provides coordinates
- IMU gives acceleration and angular velocity

Think of it as:
"What do my sensors physically detect?"

Core of Autonomous Navigation



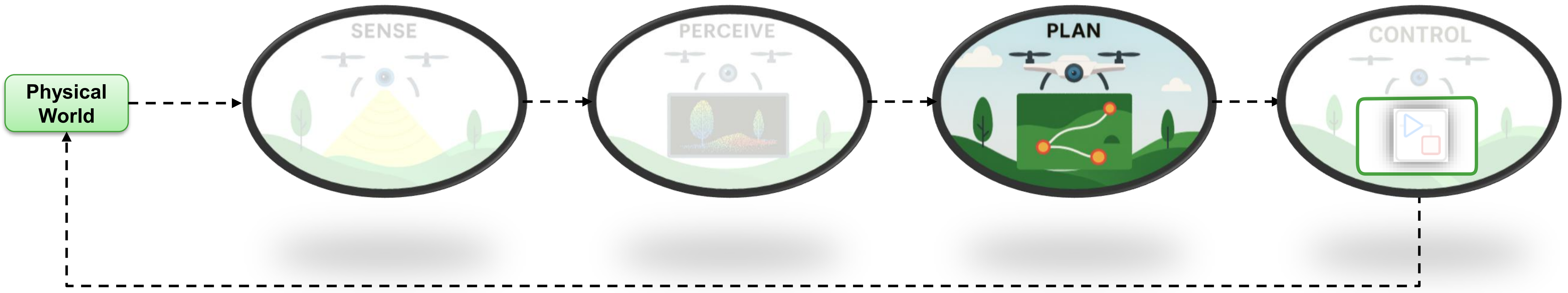
Definition: *Transform raw sensor data into meaningful representations.*

Goal: Understand the environment by extracting features and identifying entities.

Examples:
- Estimating UAV pose using GPS + IMU
- Detecting obstacles such as trees, buildings, power lines
- Classifying terrain or landing zones like grass, concrete, water
- Building maps
- Tracking moving objects such as other UAVs
- Detecting dynamic hazards like birds or any other temporary obstacle

Think of it as:
"What's around me? Where am I?"

Core of Autonomous Navigation



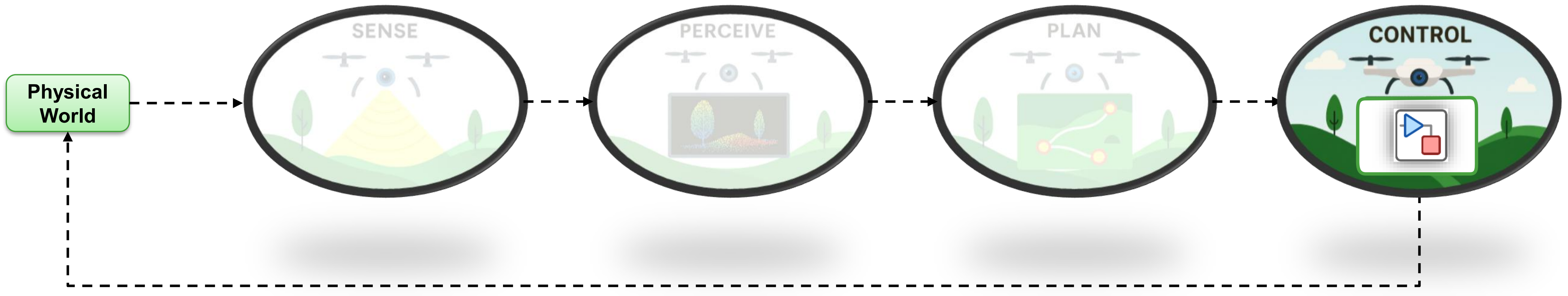
Definition: *Decide the UAV's future trajectory and actions based on mission objectives and perceived environment.*

Goal: Generate a safe, efficient flight path that satisfies constraints.

Examples:
- **Global/Mission Path Planning:** Plans the overall route from start to goal.
- **Behavioral Path Planning:** Chooses high-level maneuvers based on context.
- **Local Path Planning:** Computes short-term collision-free paths nearby.

Think of it as:
"How do I get there?"

Core of Autonomous Navigation

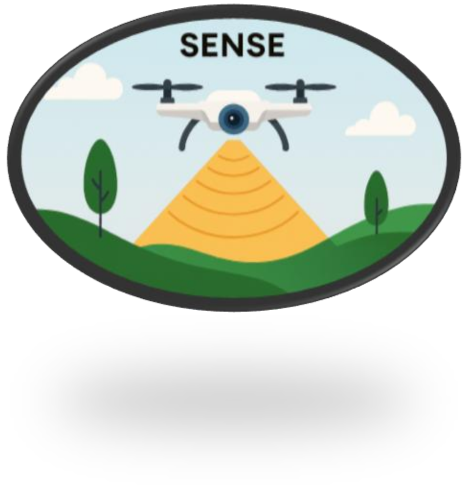


Definition: *Convert planned trajectory into actuator commands for stable flight.*

Goal: Execute the planned path while maintaining UAV stability and performance.

Examples:
- Adjust roll, pitch, yaw using PID or cascaded controllers.
- Maintain altitude and follow waypoints precisely.
- Regulate speed for smooth navigation and energy efficiency.

Think of it as:
"How do I physically fly along the planned path?"


SENSE

MathWorks

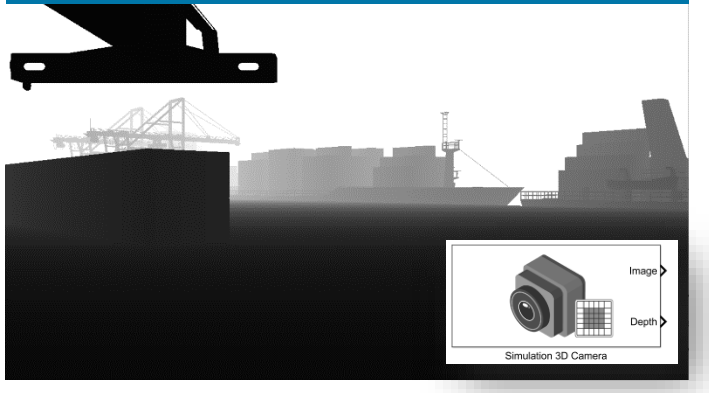
SENSE

Sensors

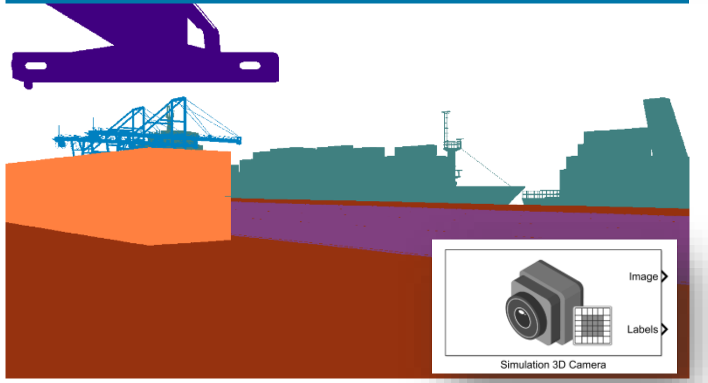
Ideal

Image
Simulation 3D Camera

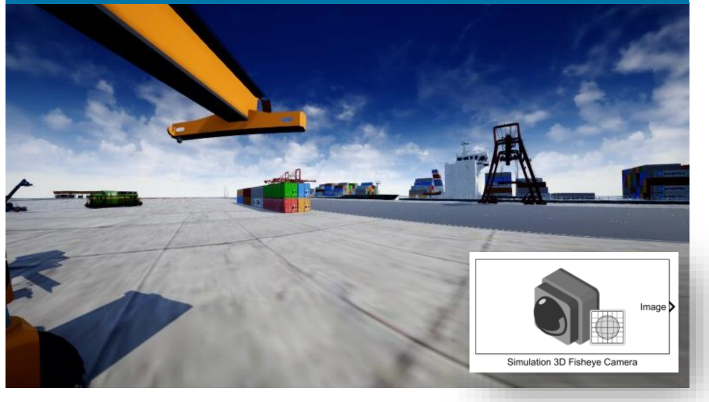
Depth



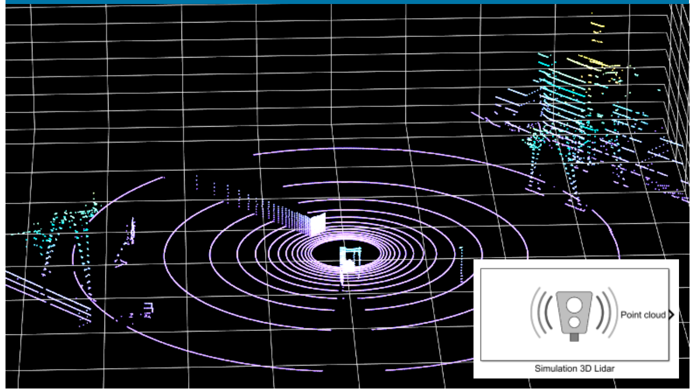
Semantic Segmentation



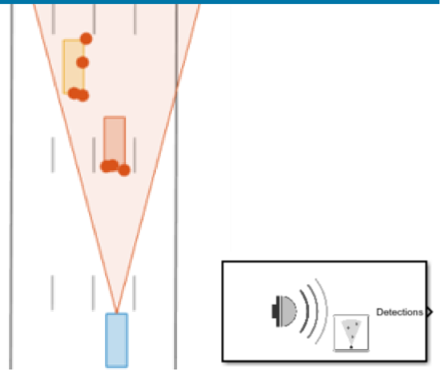
Fisheye



Lidar



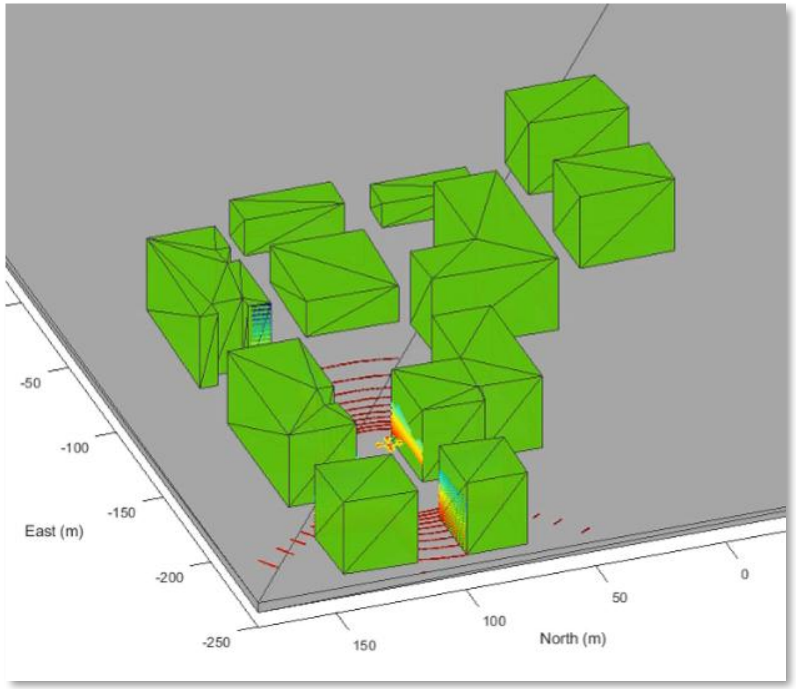
Radar

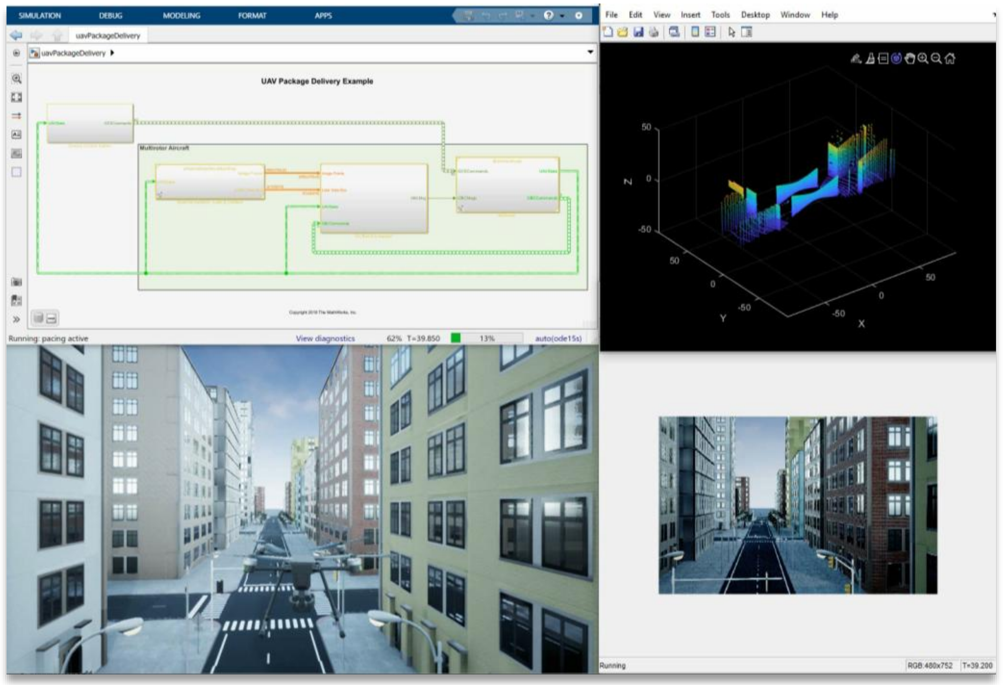


Integrated simulations with sensor models

Cuboid
Performance

Unreal Engine®
Photorealistic

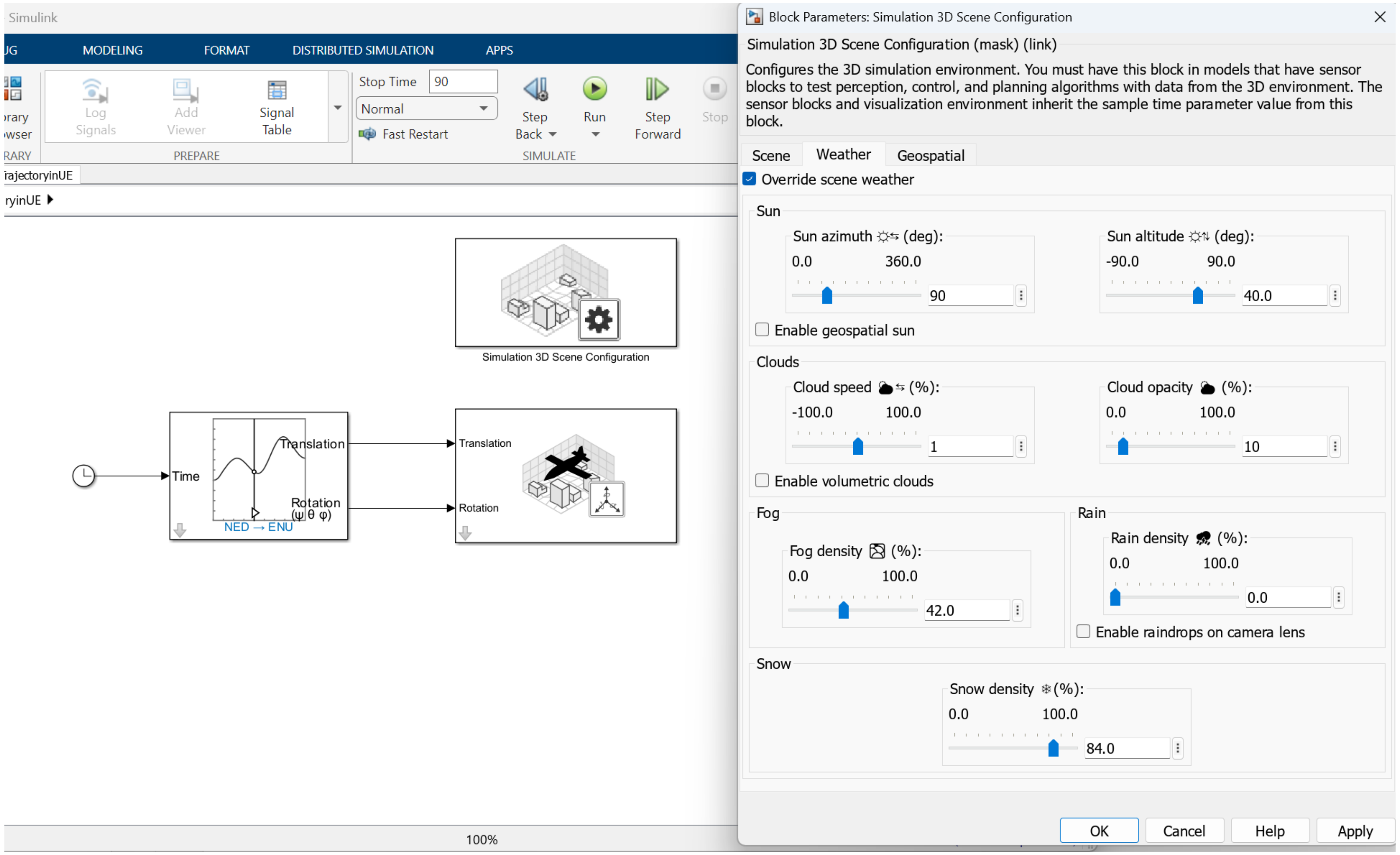




Rapidly author scenarios and generate sensor data

Realistic graphics to test autonomous algorithms in closed-loop simulations

Integrated simulations with sensor models



Block Parameters: Simulation 3D Scene Configuration

Simulation 3D Scene Configuration (mask) (link)

Configures the 3D simulation environment. You must have this block in models that have sensor blocks to test perception, control, and planning algorithms with data from the 3D environment. The sensor blocks and visualization environment inherit the sample time parameter value from this block.

Scene | Weather | Geospatial

[✓] Override scene weather

Sun
- Sun azimuth (deg): 0.0 – 360.0: 90
- Sun altitude (deg): -90.0 – 90.0: 40.0
- [ ] Enable geospatial sun

Clouds
- Cloud speed (%): -100.0 – 100.0: 1
- Cloud opacity (%): 0.0 – 100.0: 10
- [ ] Enable volumetric clouds

Fog
- Fog density (%): 0.0 – 100.0: 42.0

Rain
- Rain density (%): 0.0 – 100.0: 0.0
- [ ] Enable raindrops on camera lens

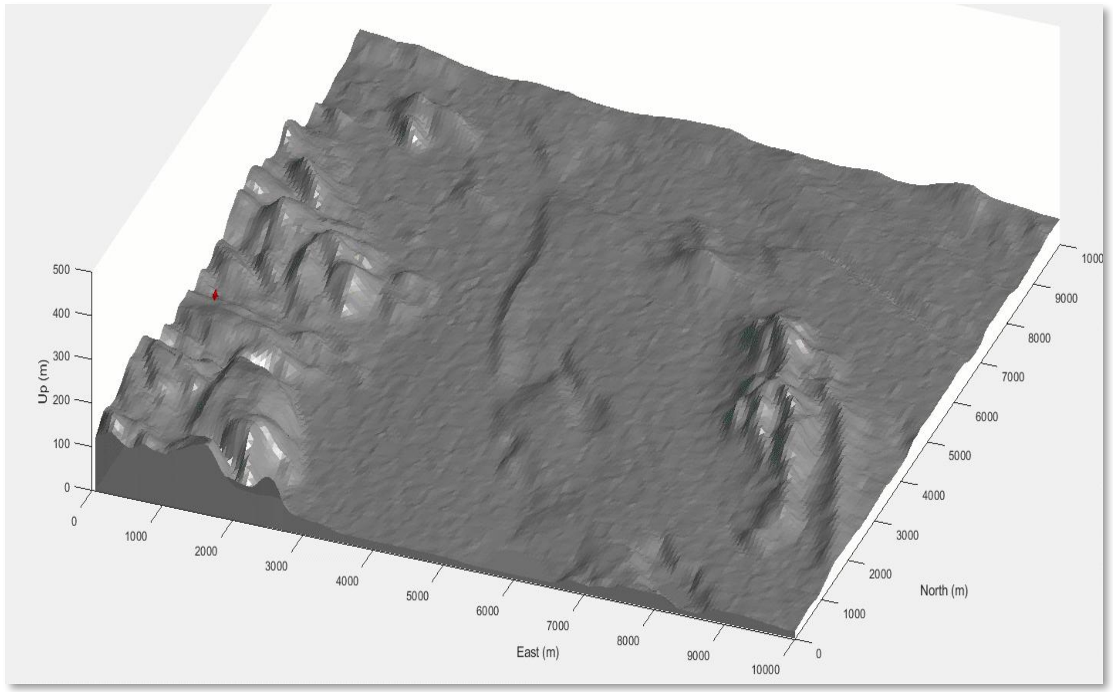
Snow
- Snow density (%): 0.0 – 100.0: 84.0

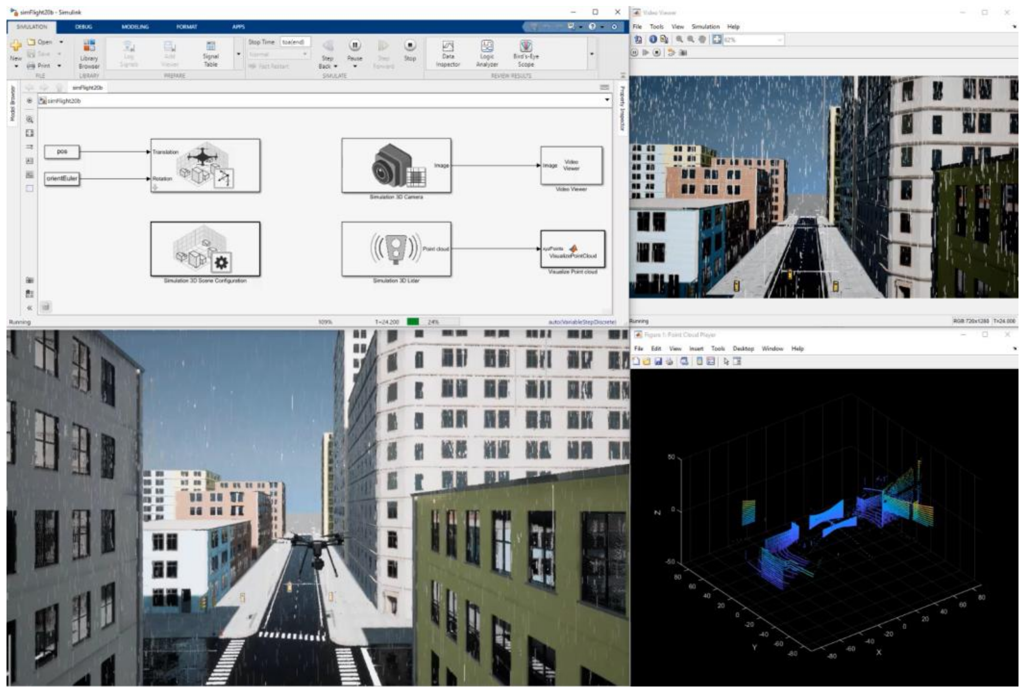
OK | Cancel | Help | Apply

Integrated simulations with sensor models

Cuboid
Performance

Unreal Engine®
Photorealistic

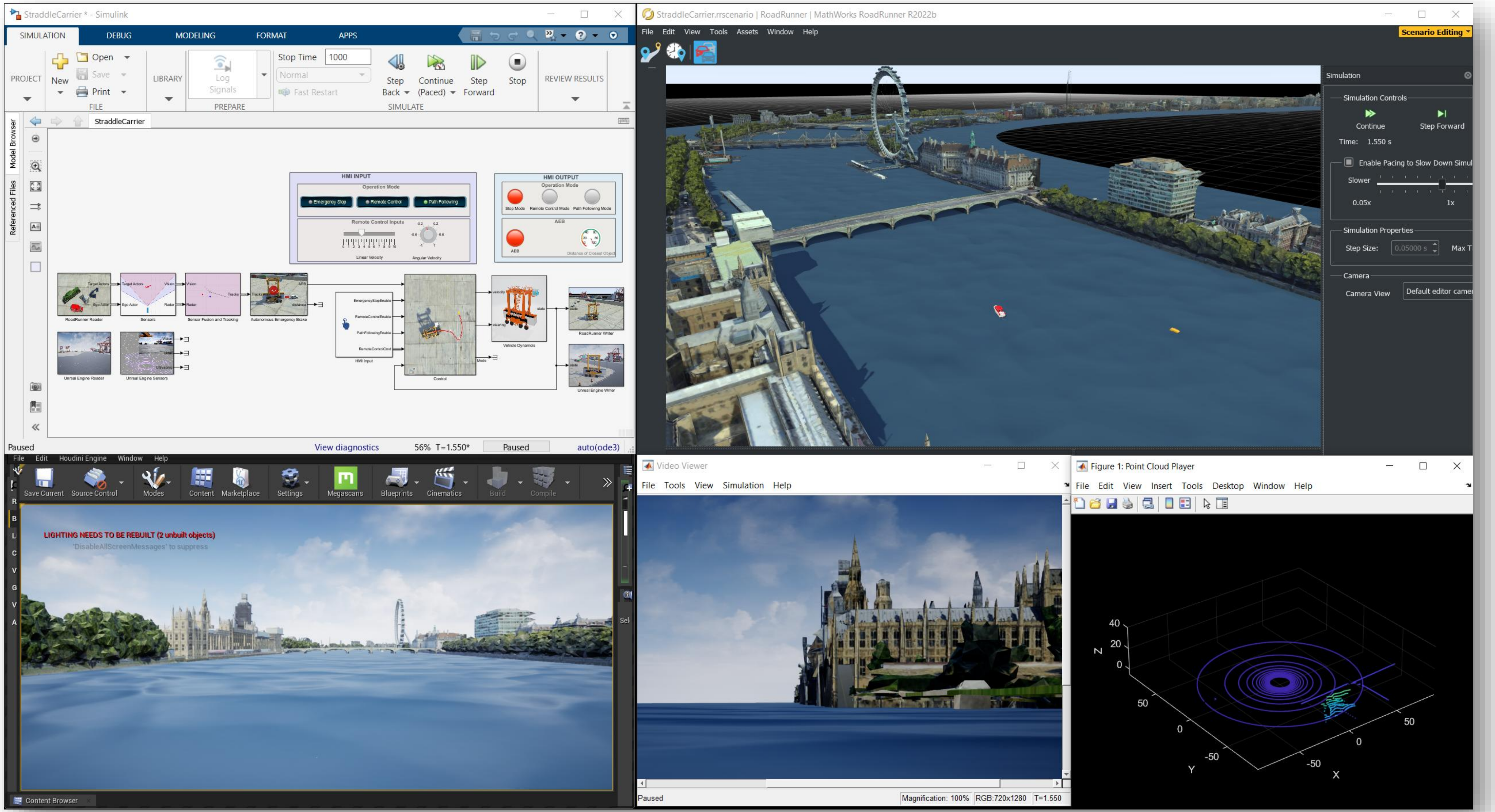




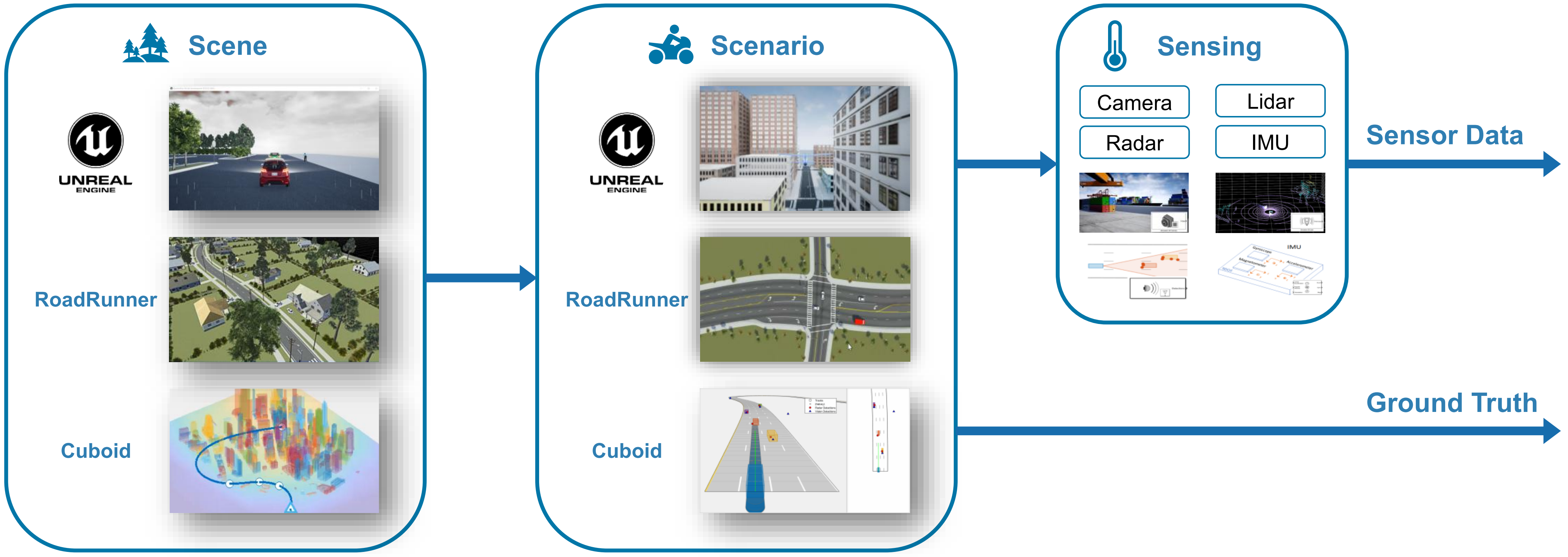
Rapidly author scenarios and generate sensor data

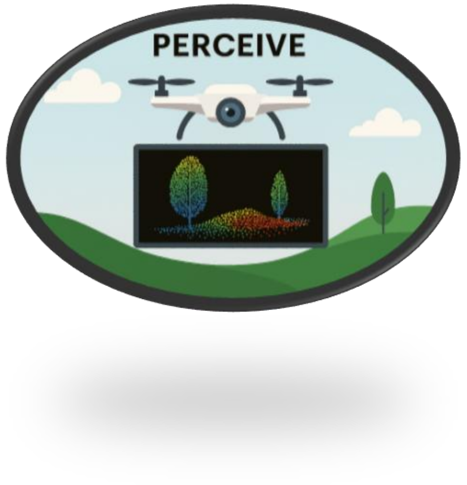
Realistic graphics to test autonomous algorithms in closed-loop simulations

Sense - Scene, Scenario and Sensor

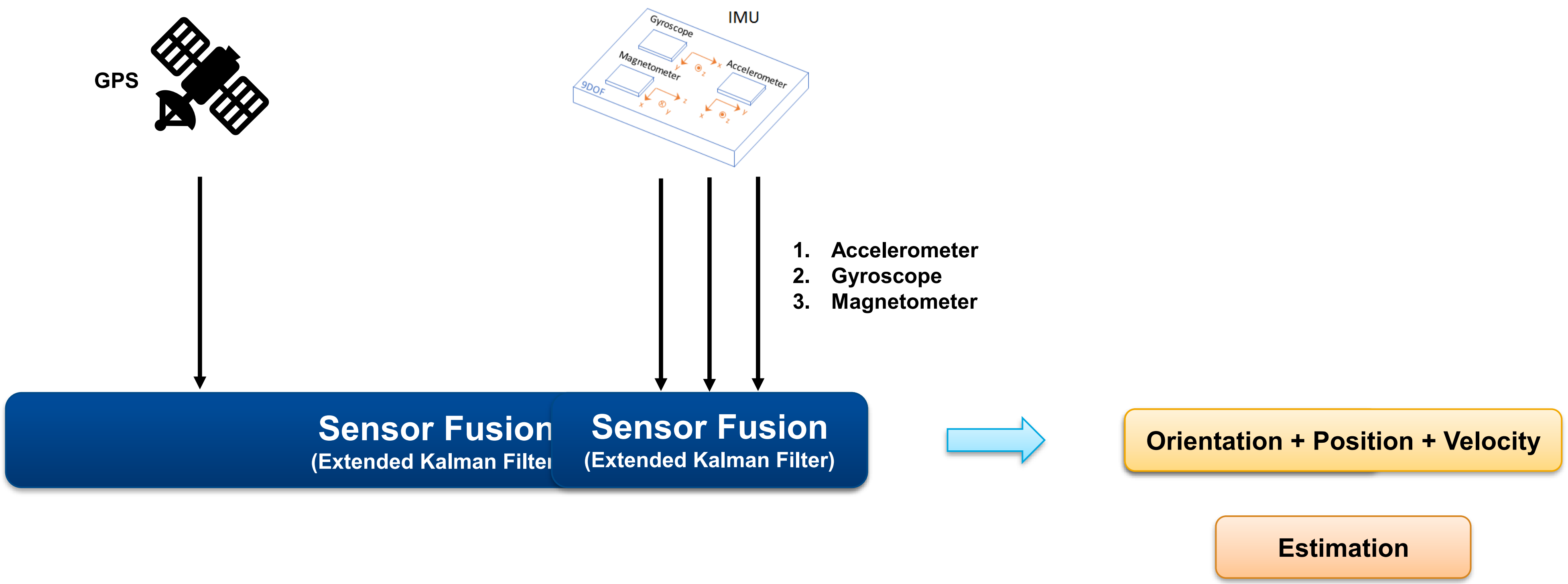


Sense - Scene, Scenario and Sensor

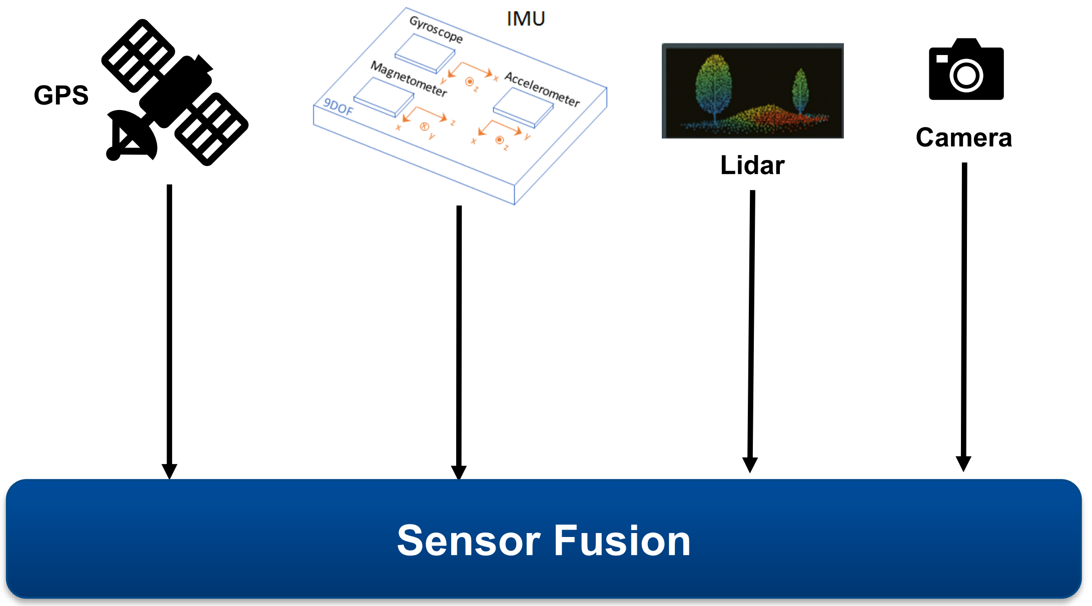



PERCEIVE

Perceive – Localization (self-awareness)



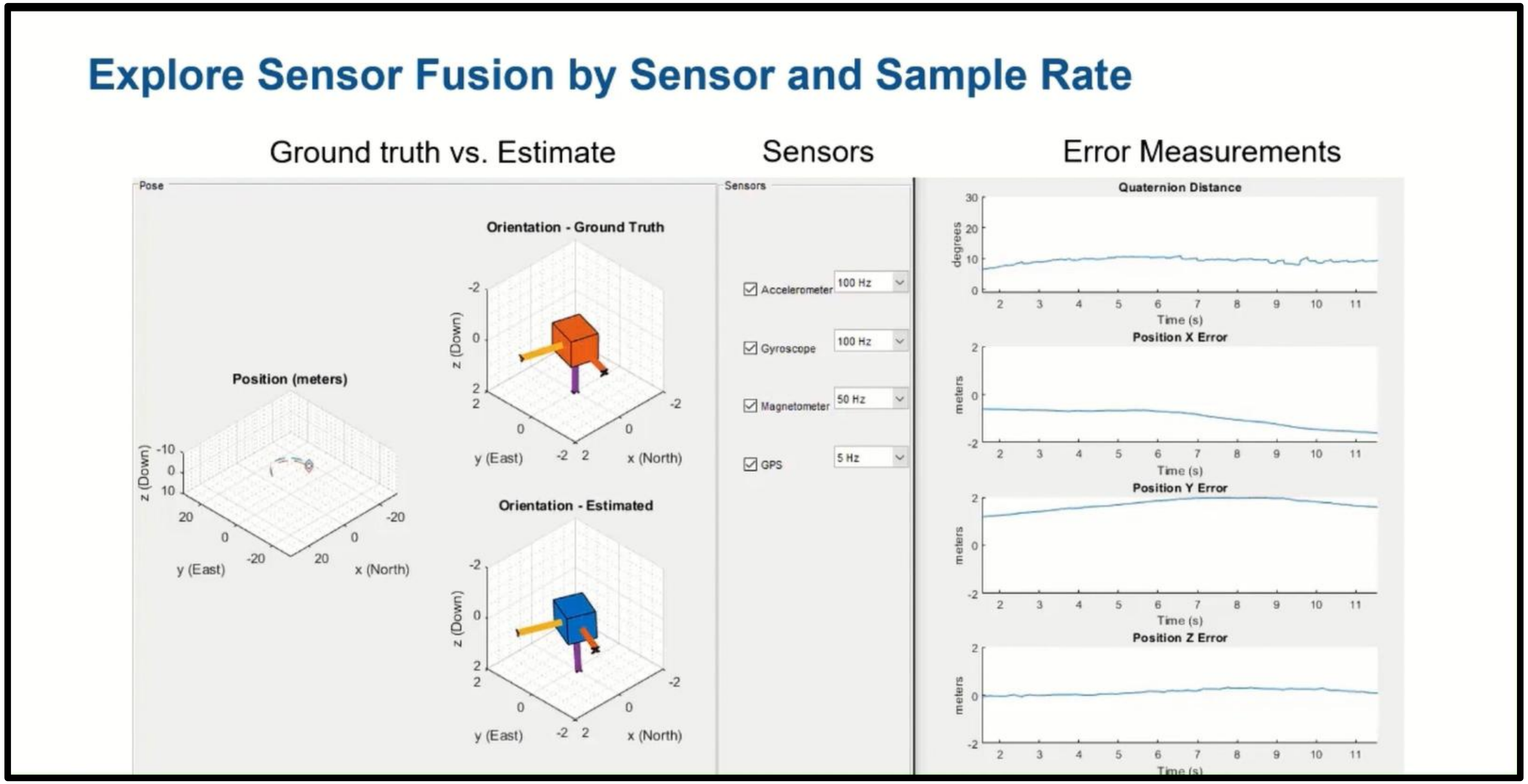
Perceive – Localization (self-awareness)



Increase robustness and accuracy

- **Tunnels**
- **IMU drift**
- **Weather**
- **etc.**

Perceive – positioning – self awareness


Explore Sensor Fusion by Sensor and Sample Rate

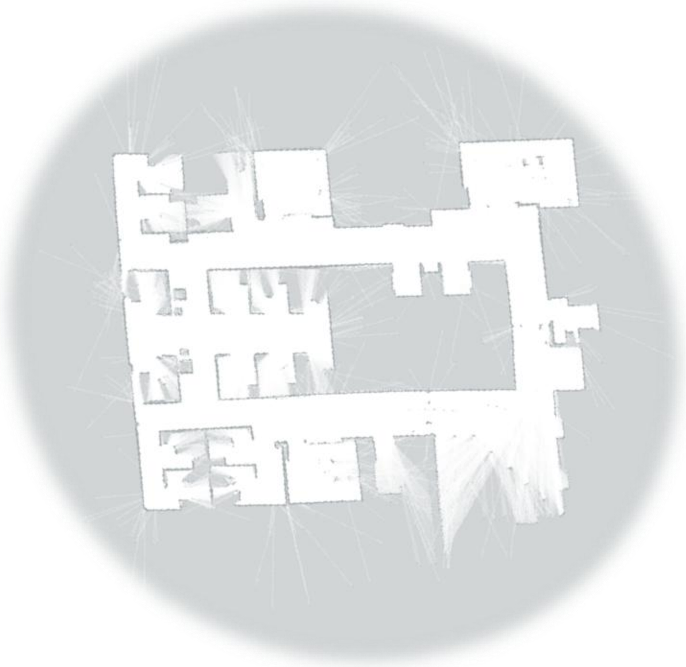
Ground truth vs. Estimate

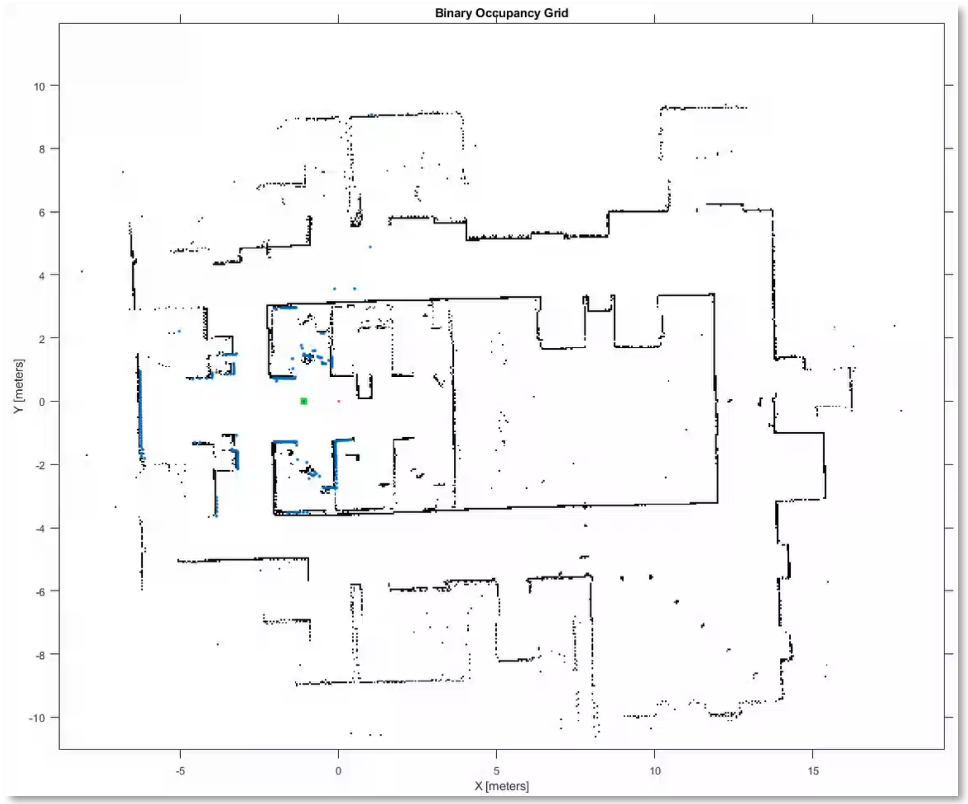
Sensors

Error Measurements

☑ Accelerometer 100 Hz
☑ Gyroscope 100 Hz
☑ Magnetometer 50 Hz
☑ GPS 5 Hz

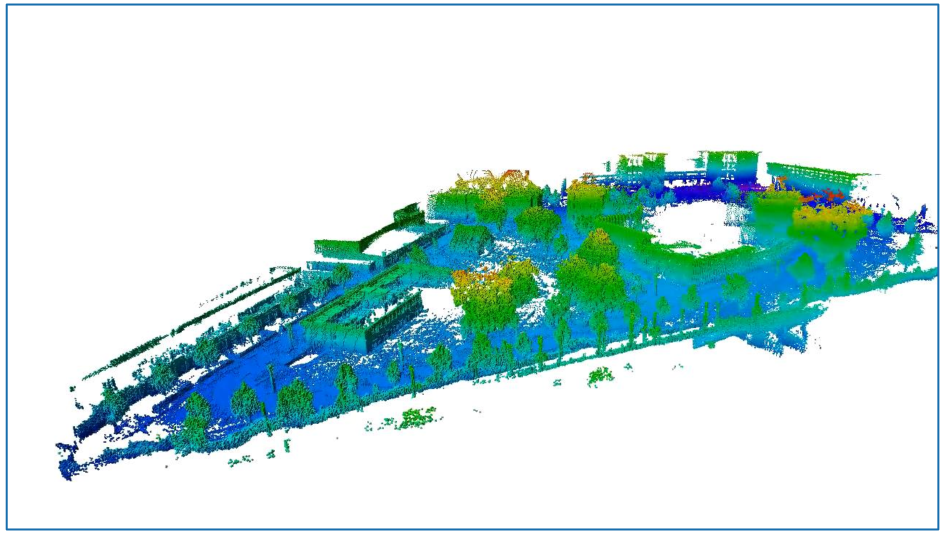
Perceive – Particle filter, Monte Carlo Localization

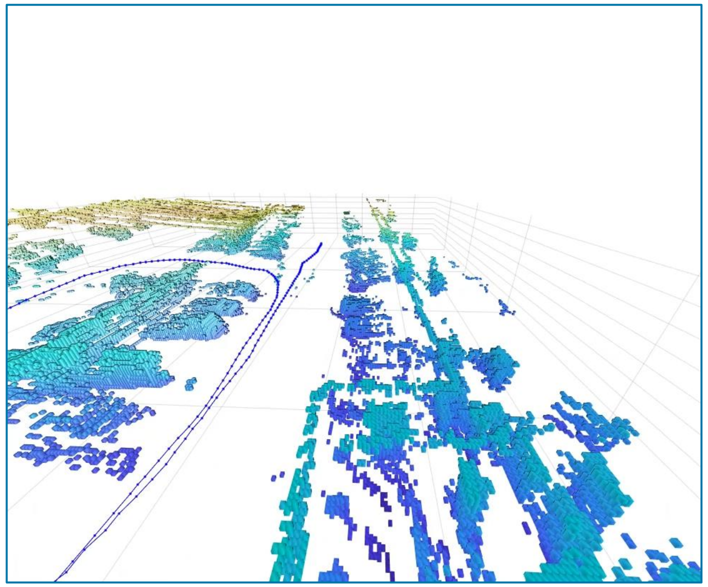




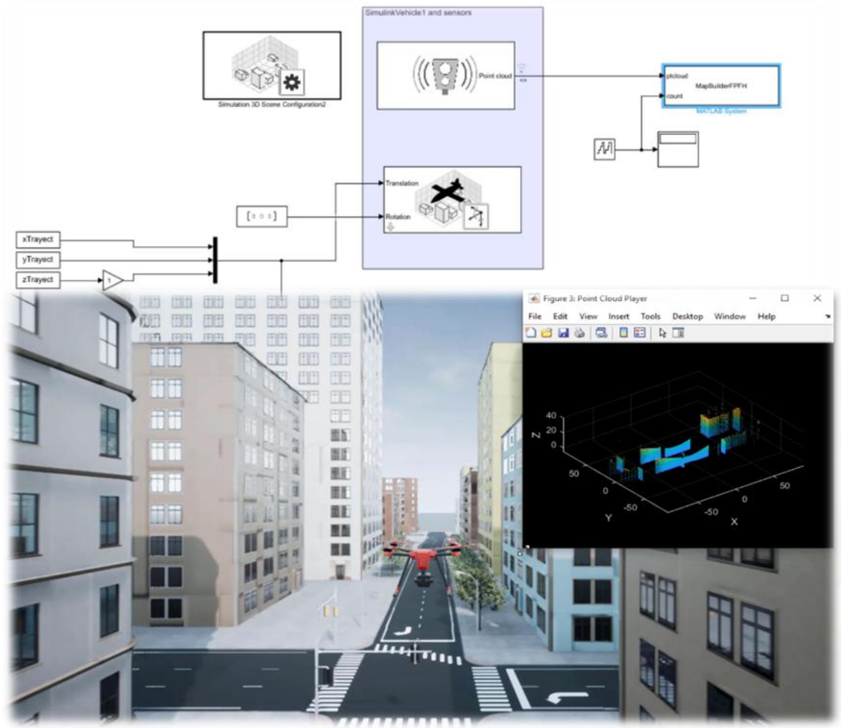
Perceive – Localize and mapping

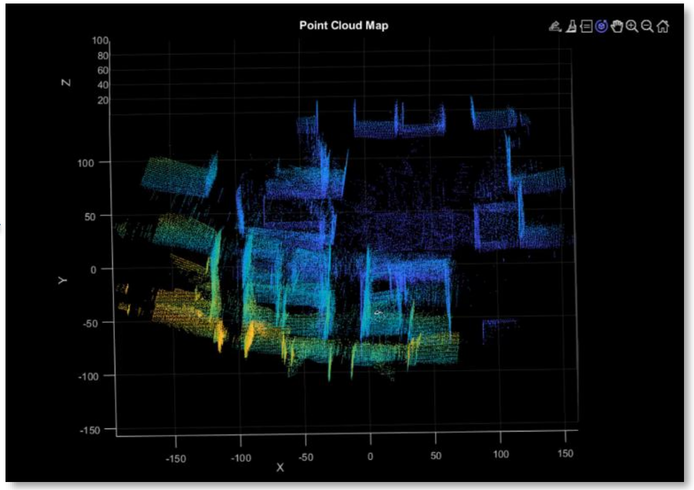
3D map for UAV





Perceive - Build 3D map using simulation Lidar point cloud data

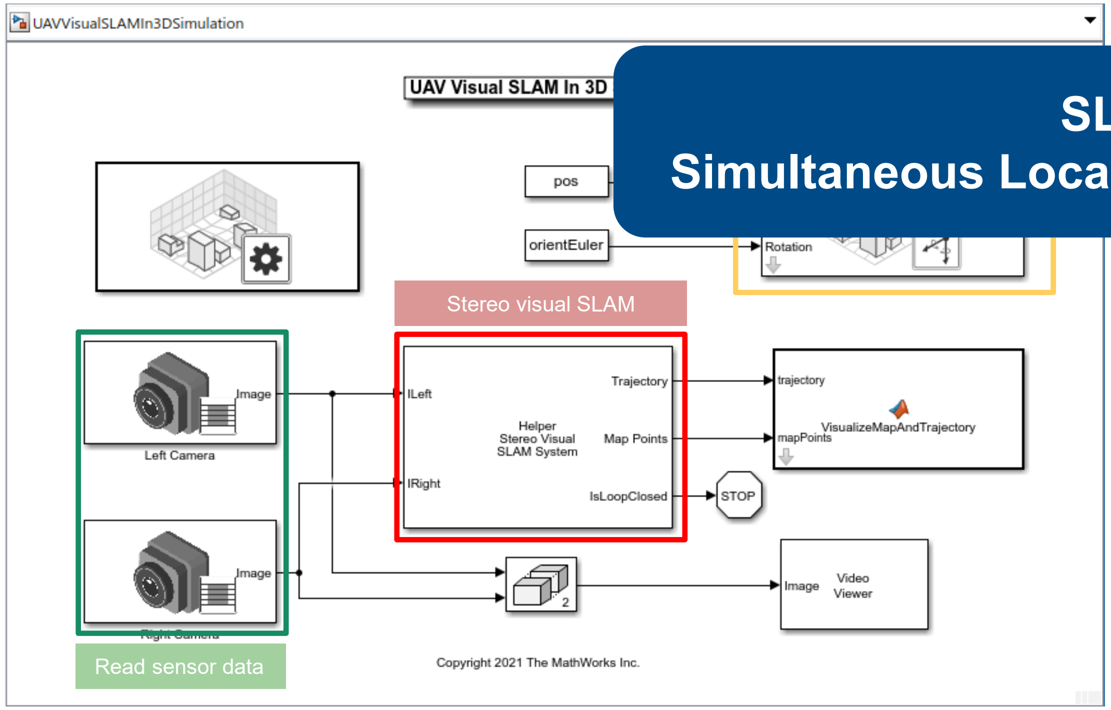

SimulinkVehicle1 and sensors
Simulation 3D Scene Configuration2
Point cloud
ptcloud
count
MapBuilderFPFH
MATLAB System
Translation
Rotation
xTrayect
yTrayect
zTrayect
Figure 3: Point Cloud Player
File Edit View Insert Tools Desktop Window Help



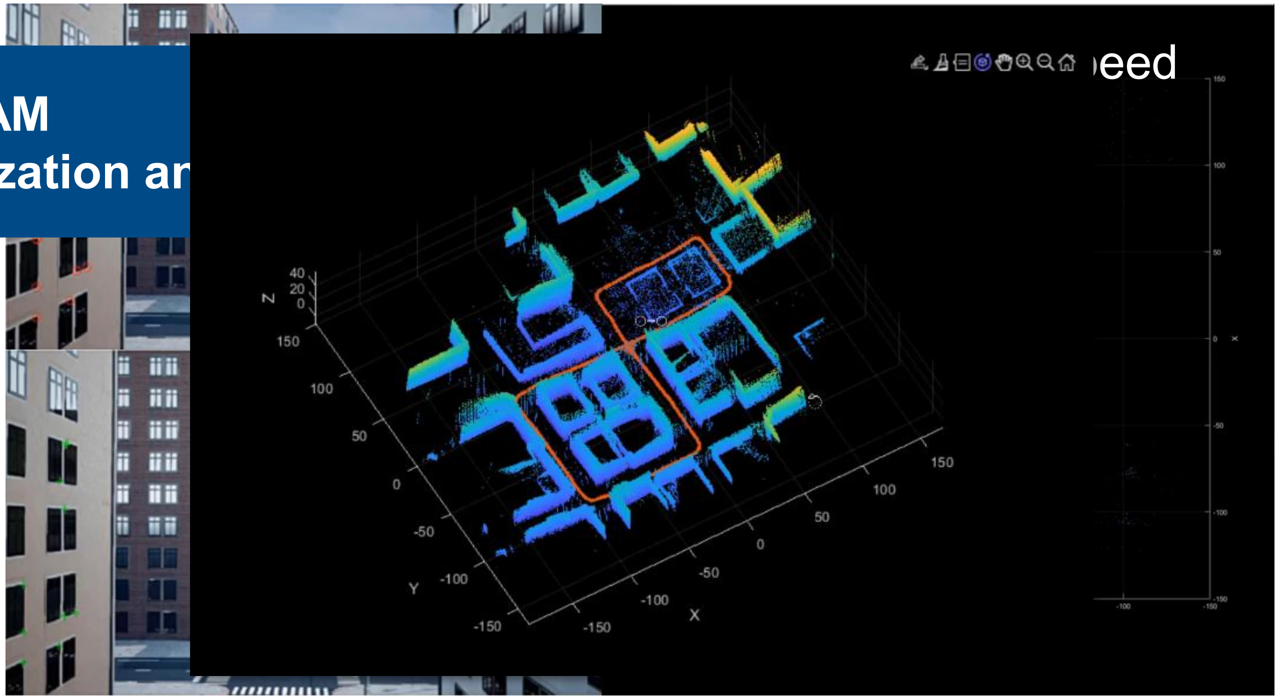
Execute simulation
Obtain sensor data

Extract and match features
Register and align point cloud

Map an Environment using Stereo Visual SLAM and verify with Unreal Simulation



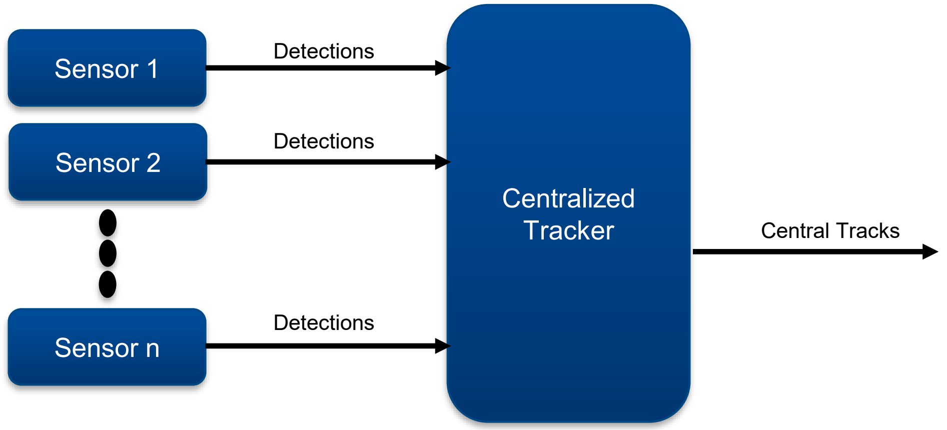
SLAM
Simultaneous Localization ar



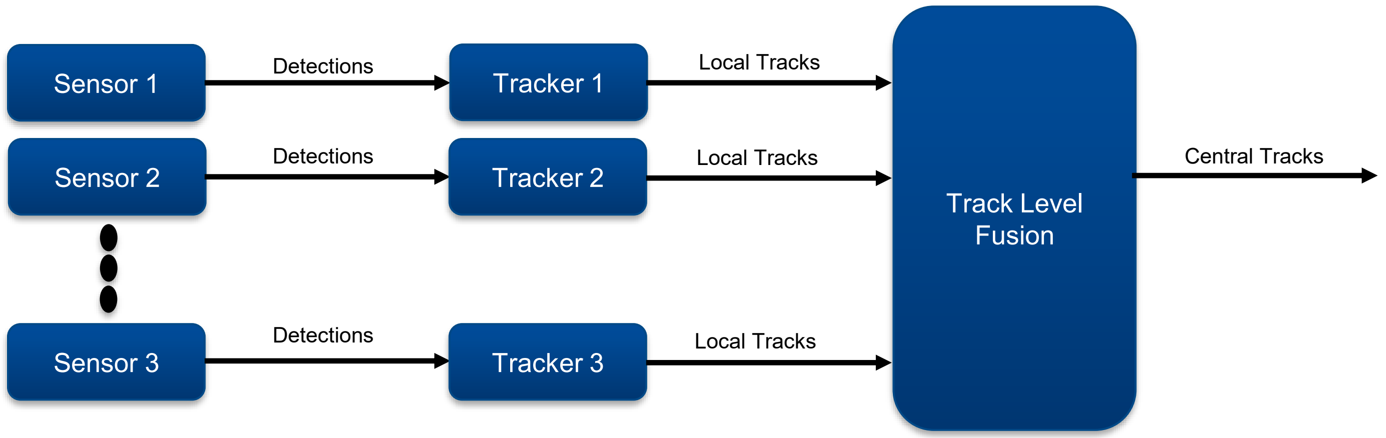
Detect loop-closures
Create pose graph
Optimize poses

Perceive - Tracking

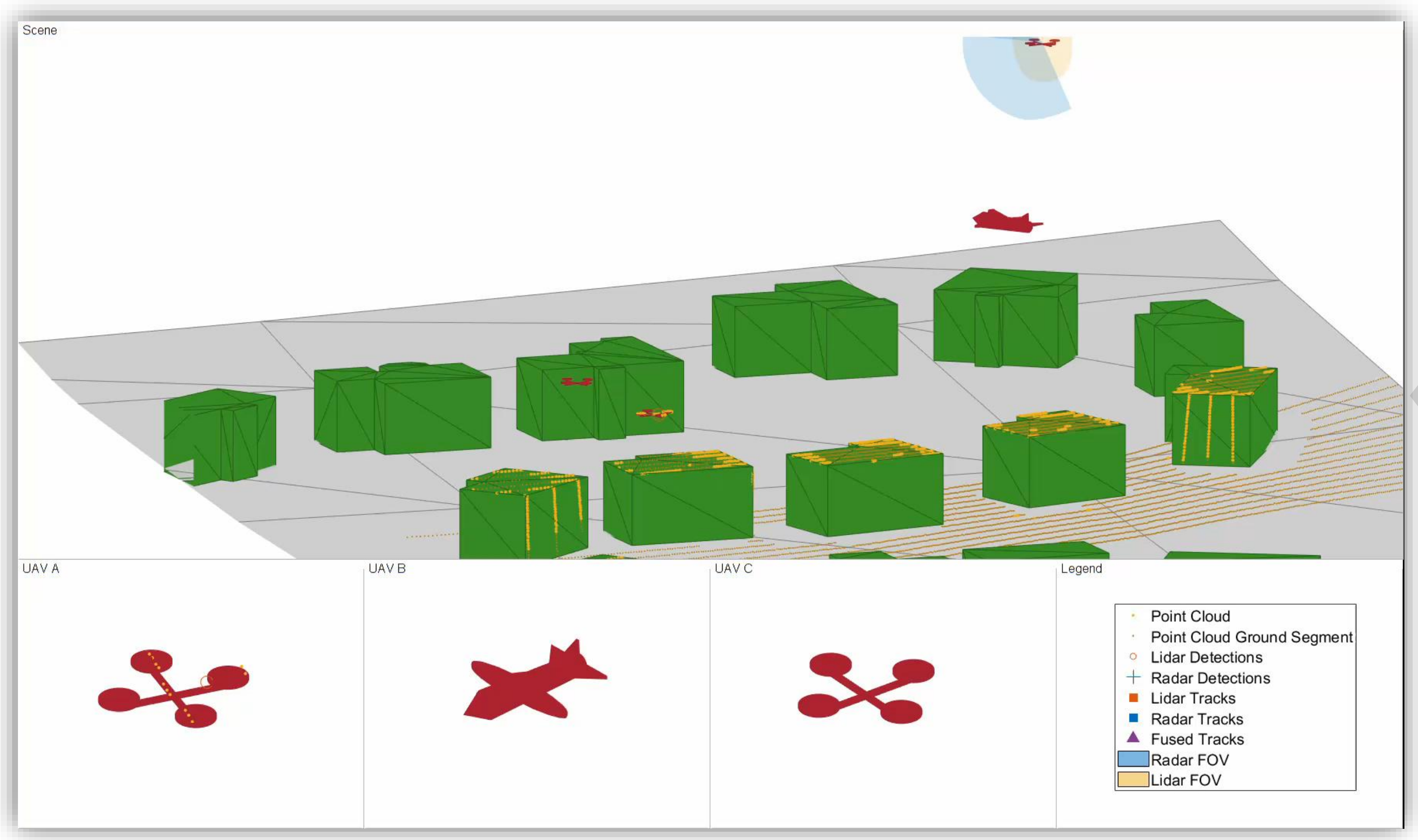
Centralized fusion of sensor data



Decentralized fusion of sensor data



Lidar and Radar Fusion in Urban Air Mobility Scenario



Performance Analysis with GOSPA Metric

$$d_b(x_i, y_{\pi(i)}) = d_p + d_v + d_y + d_{yr} + d_d$$

d_p: weighted positional error

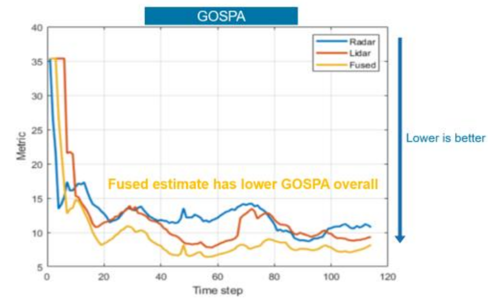
d_v: weighted velocity error

d_y: weighted yaw/orientation error

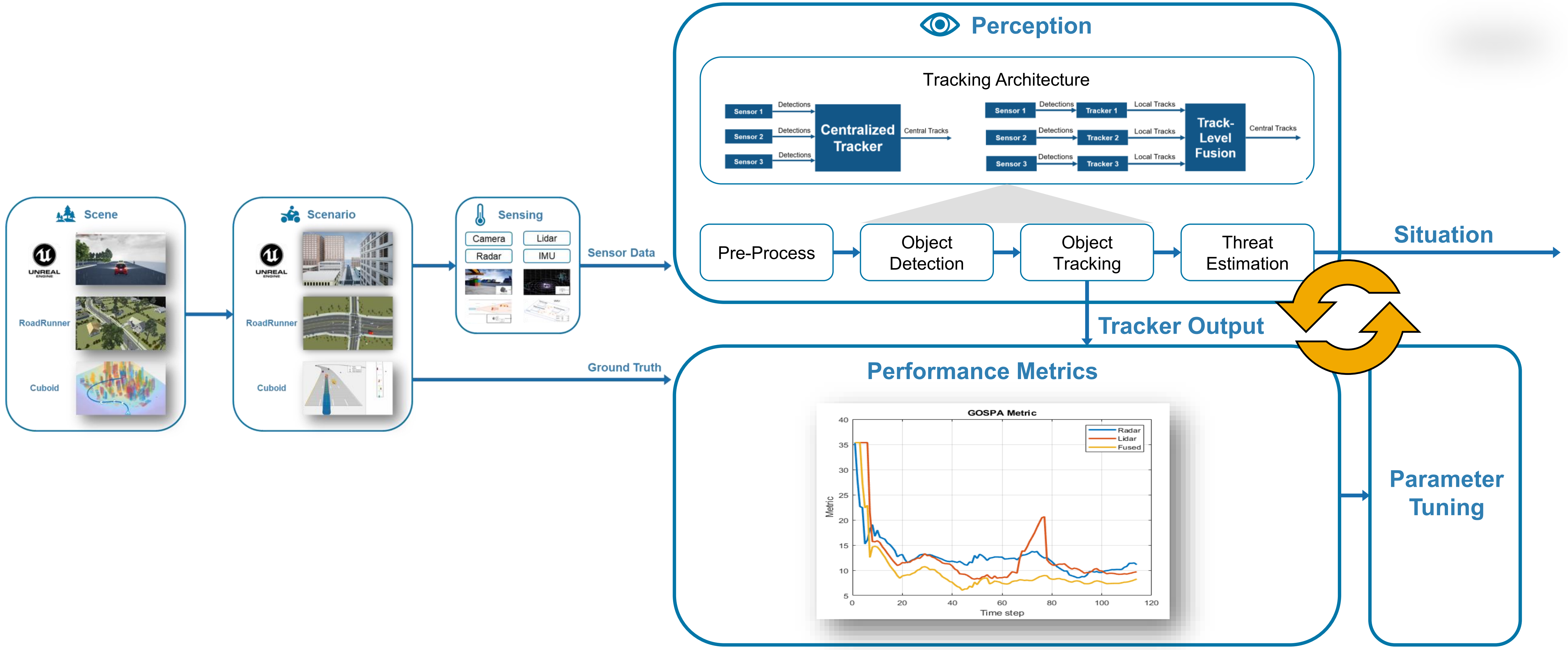
d_{yr}: weighted yaw rate error

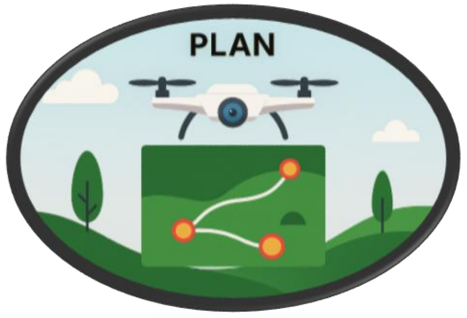
d_d: weights dimensions error

$$GOSPA = \left[\sum_{i=0}^{m} (min(d_b, c))^p + \frac{c^p}{\alpha}(n-m)\right]^{\frac{1}{p}}$$

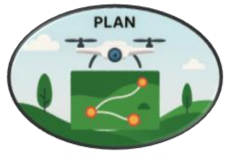


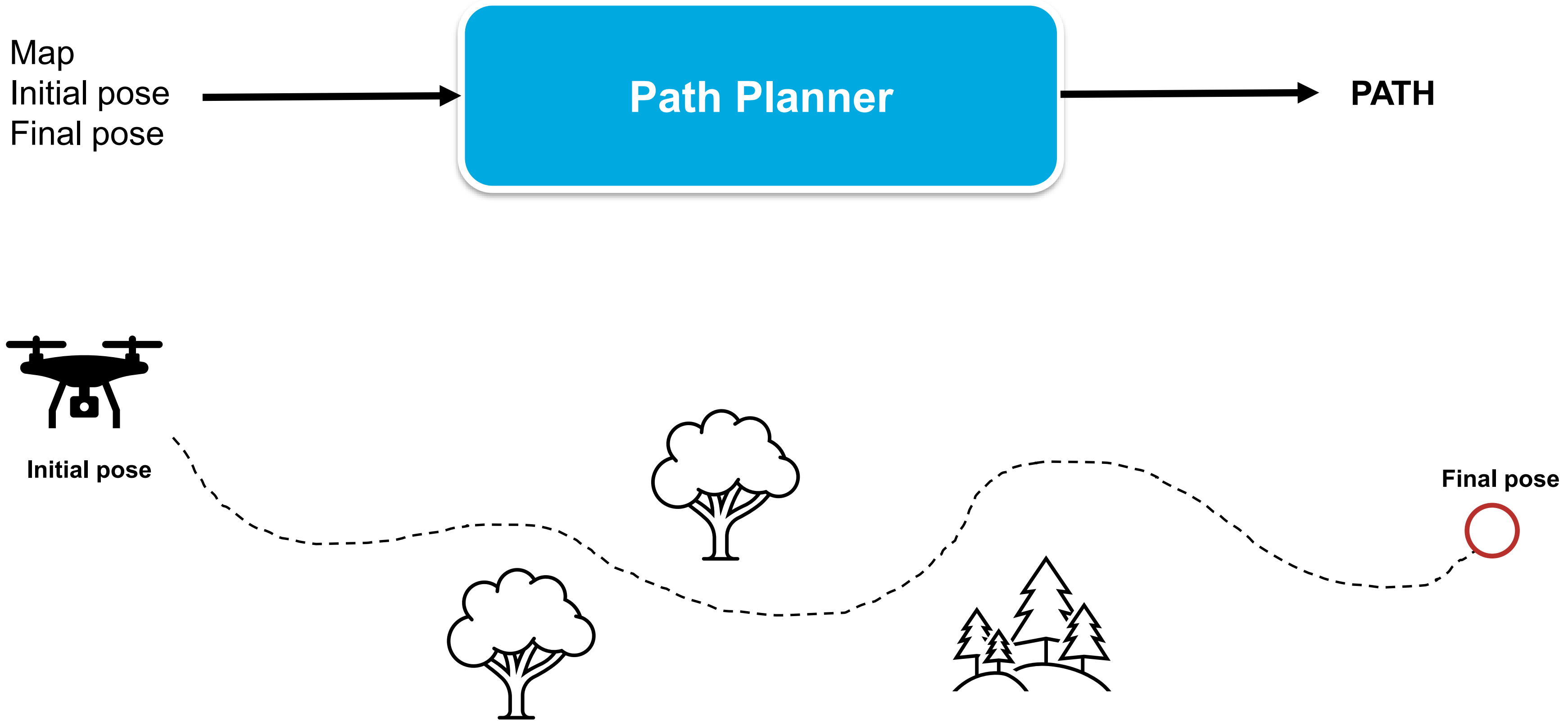
Perception Pipeline

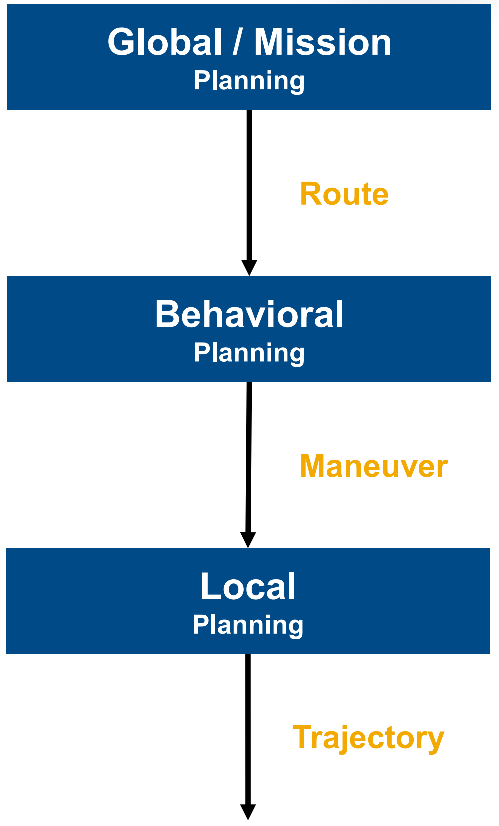



PLAN

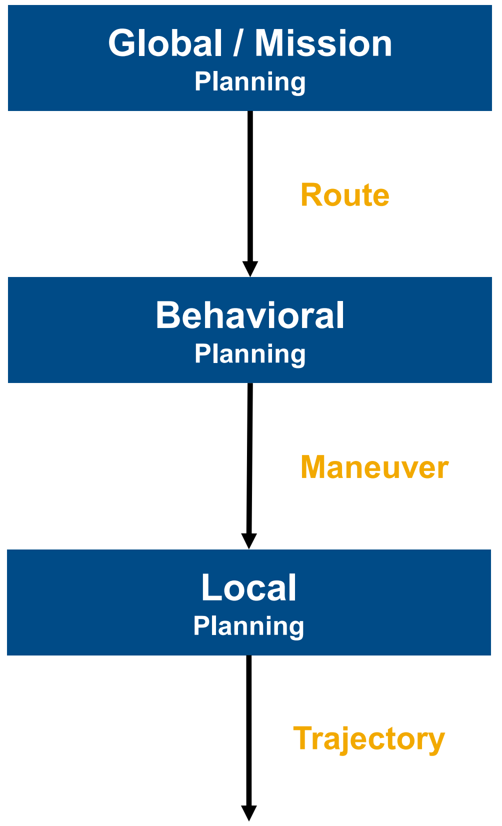
Path Planning

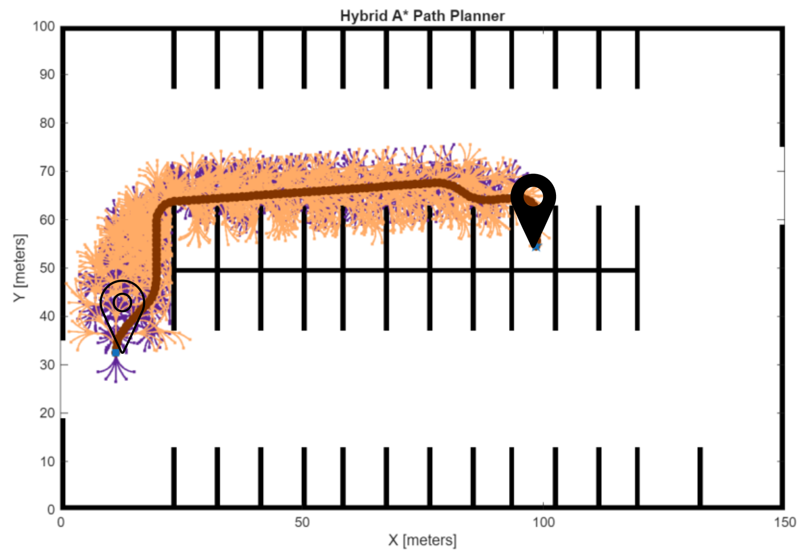


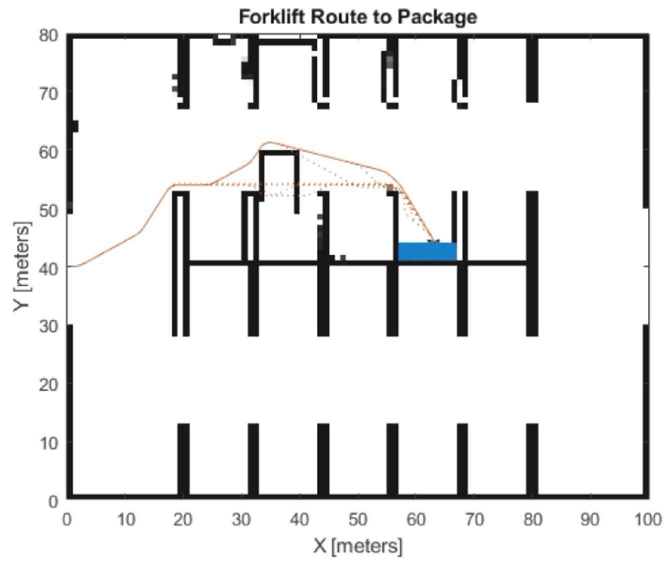




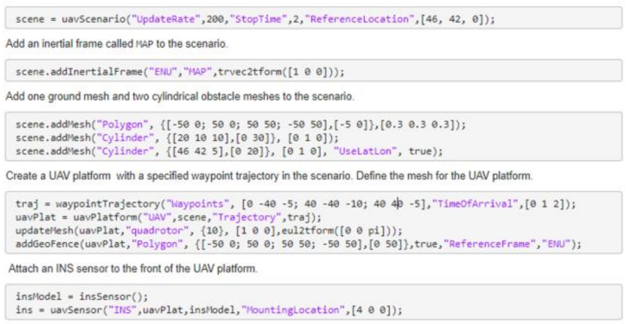
Path Planning







Path Planning



```
scene = uavScenario("UpdateRate",200,"StopTime",2,"ReferenceLocation",[46, 42, 0]);
```

Add an inertial frame called `MAP` to the scenario.

```
scene.addInertialFrame("ENU","MAP",trvec2tform([1 0 0]));
```

Add one ground mesh and two cylindrical obstacle meshes to the scenario.

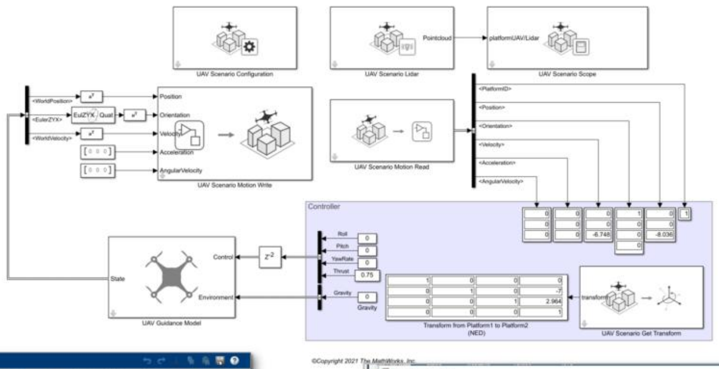
```
scene.addMesh("Polygon", {[-50 0; 50 0; 50 50; -50 50],[-5 0]},[0.3 0.3 0.3]);
scene.addMesh("Cylinder", {[20 10 10],[0 30]}, [0 1 0]);
scene.addMesh("Cylinder", {[46 42 5],[0 20]}, [0 1 0], "UseLatLon", true);
```

Create a UAV platform with a specified waypoint trajectory in the scenario. Define the mesh for the UAV platform.

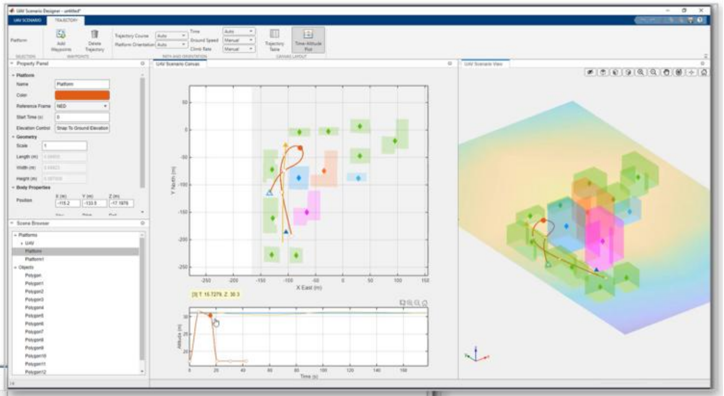
```
traj = waypointTrajectory("Waypoints", [0 -40 -5; 40 -40 -10; 40 40 -5],"TimeOfArrival",[0 1 2]);
uavPlat = uavPlatform("UAV",scene,"Trajectory",traj);
updateMesh(uavPlat,"quadrotor", {10}, [1 0 0],eul2tform([0 0 pi]));
addGeoFence(uavPlat,"Polygon", {[-50 0; 50 0; 50 50; -50 50],[0 50]},true,"ReferenceFrame","ENU");
```

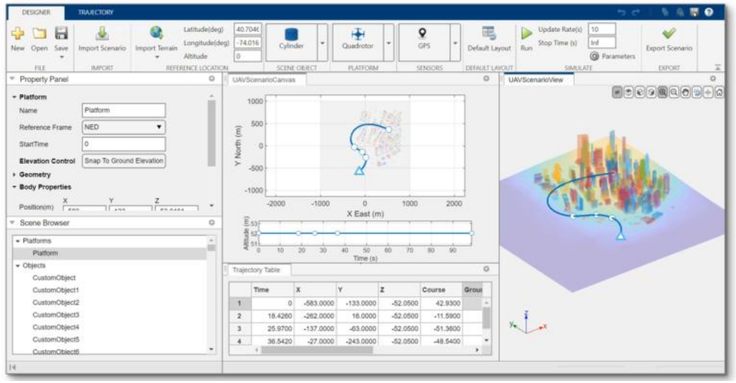
Attach an INS sensor to the front of the UAV platform.

```
insModel = insSensor();
ins = uavSensor("INS",uavPlat,insModel,"MountingLocation",[4 0 0]);
```

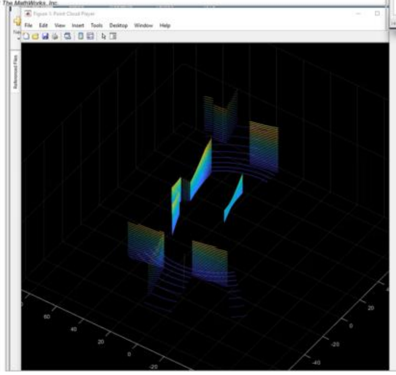


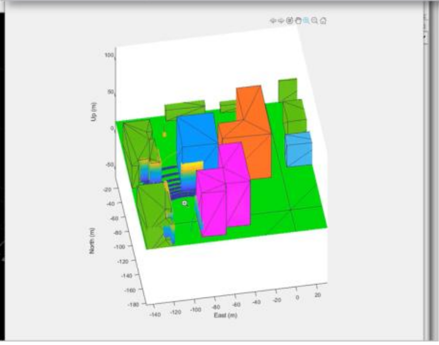
UAV Scenario Configuration
UAV Scenario Lidar
UAV Scenario Scope
UAV Scenario Motion Write
UAV Scenario Motion Read
Controller
UAV Guidance Model
Transform from Platform1 to Platform2 (NED)
UAV Scenario Get Transform



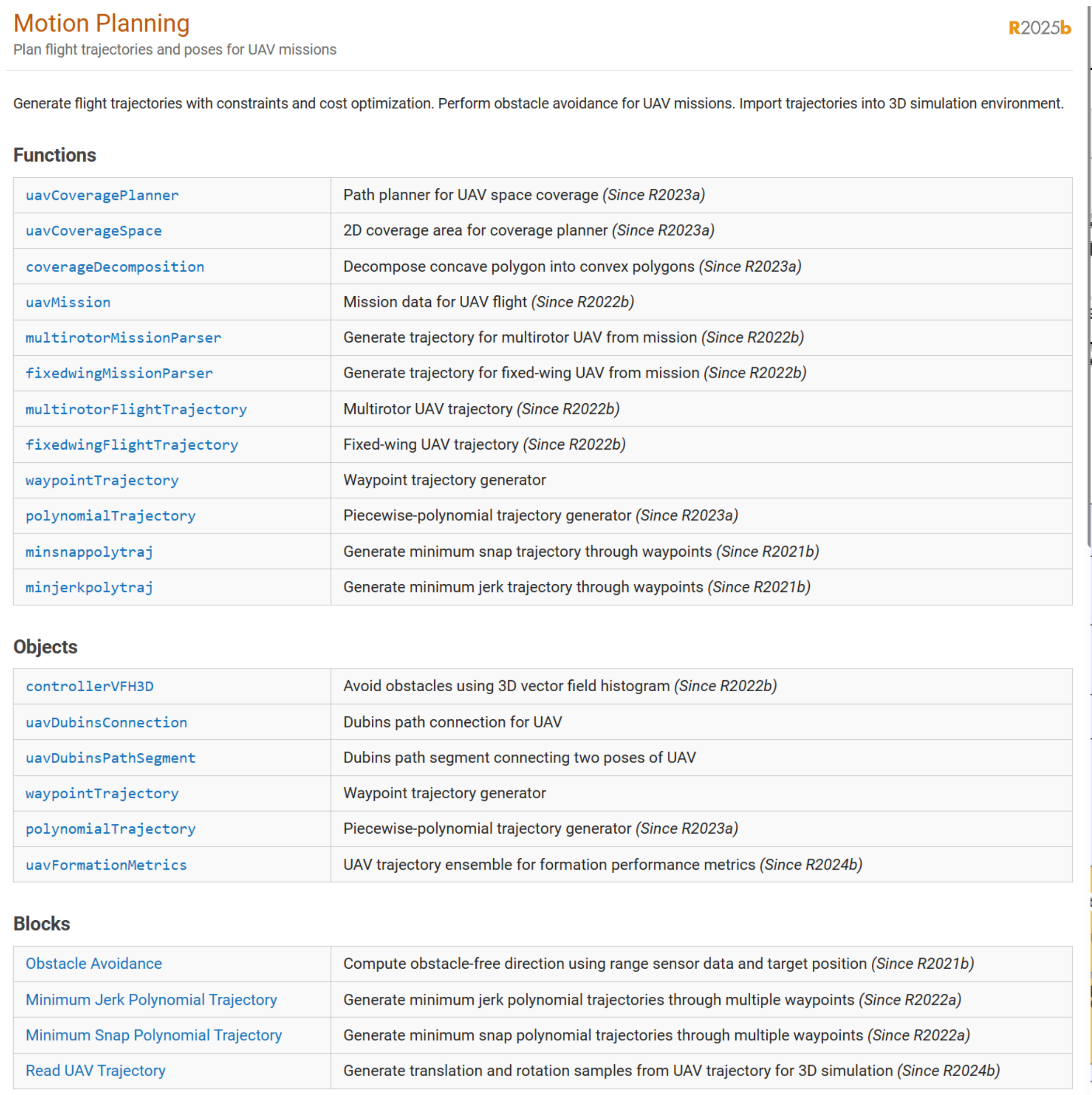


©Copyright 2021 The MathWorks, Inc.





UAV Motion Planning



Motion Planning

R2025b

Plan flight trajectories and poses for UAV missions

Generate flight trajectories with constraints and cost optimization. Perform obstacle avoidance for UAV missions. Import trajectories into 3D simulation environment.

Functions

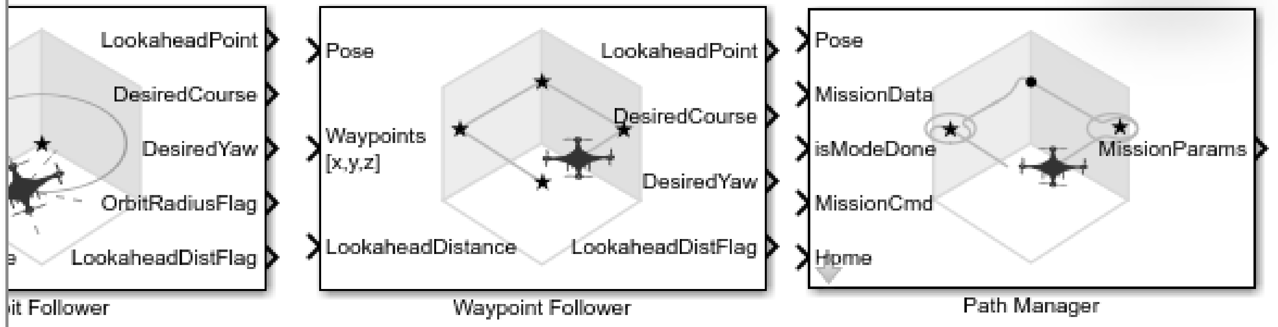
`uavCoveragePlanner`	Path planner for UAV space coverage *(Since R2023a)*
`uavCoverageSpace`	2D coverage area for coverage planner *(Since R2023a)*
`coverageDecomposition`	Decompose concave polygon into convex polygons *(Since R2023a)*
`uavMission`	Mission data for UAV flight *(Since R2022b)*
`multirotorMissionParser`	Generate trajectory for multirotor UAV from mission *(Since R2022b)*
`fixedwingMissionParser`	Generate trajectory for fixed-wing UAV from mission *(Since R2022b)*
`multirotorFlightTrajectory`	Multirotor UAV trajectory *(Since R2022b)*
`fixedwingFlightTrajectory`	Fixed-wing UAV trajectory *(Since R2022b)*
`waypointTrajectory`	Waypoint trajectory generator
`polynomialTrajectory`	Piecewise-polynomial trajectory generator *(Since R2023a)*
`minsnappolytraj`	Generate minimum snap trajectory through waypoints *(Since R2021b)*
`minjerkpolytraj`	Generate minimum jerk trajectory through waypoints *(Since R2021b)*

Objects

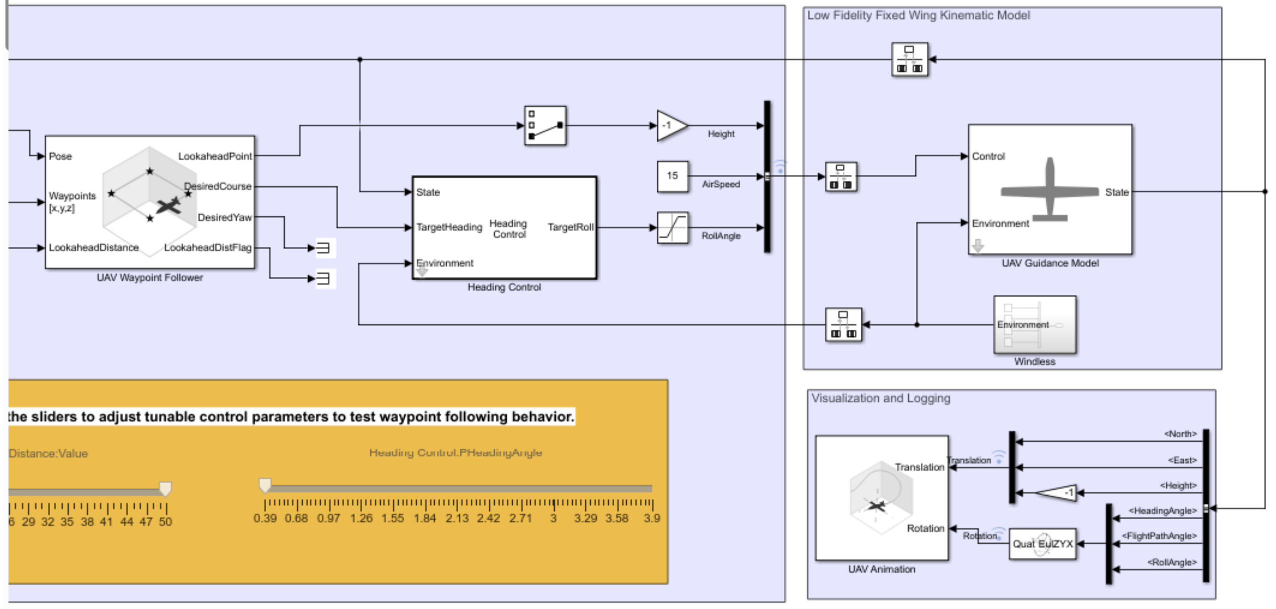
`controllerVFH3D`	Avoid obstacles using 3D vector field histogram *(Since R2022b)*
`uavDubinsConnection`	Dubins path connection for UAV
`uavDubinsPathSegment`	Dubins path segment connecting two poses of UAV
`waypointTrajectory`	Waypoint trajectory generator
`polynomialTrajectory`	Piecewise-polynomial trajectory generator *(Since R2023a)*
`uavFormationMetrics`	UAV trajectory ensemble for formation performance metrics *(Since R2024b)*

Blocks

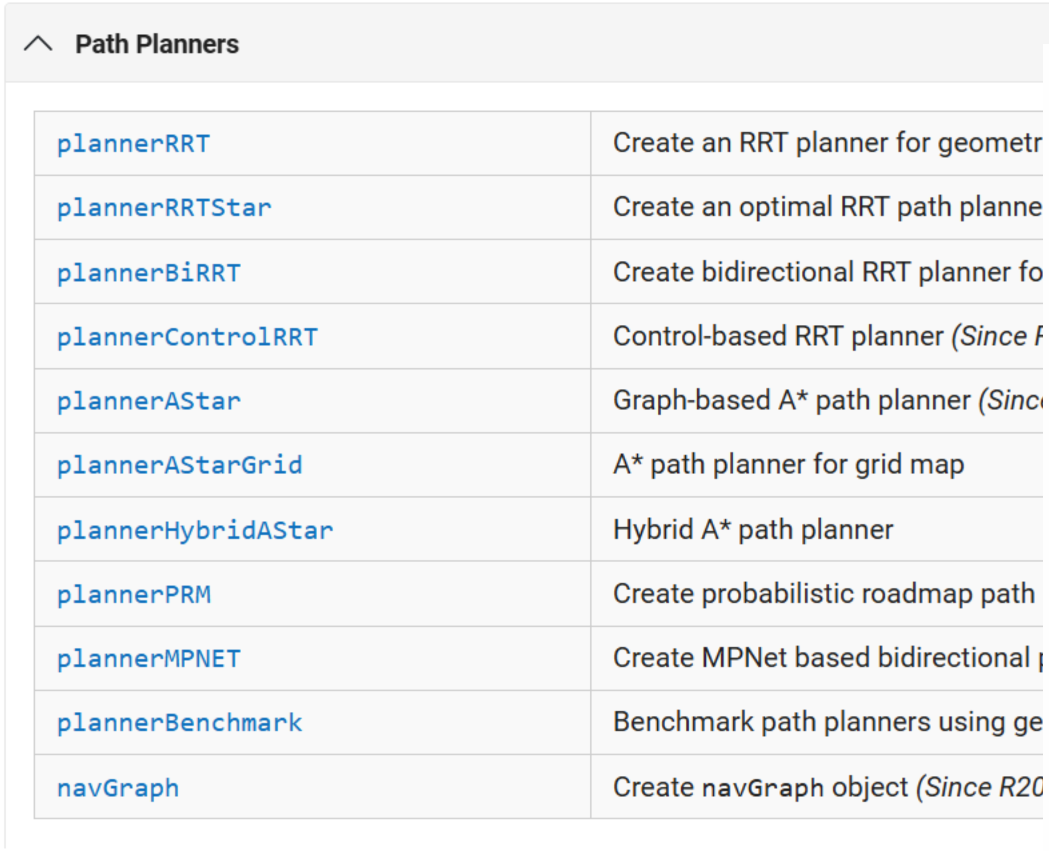
Obstacle Avoidance	Compute obstacle-free direction using range sensor data and target position *(Since R2021b)*
Minimum Jerk Polynomial Trajectory	Generate minimum jerk polynomial trajectories through multiple waypoints *(Since R2022a)*
Minimum Snap Polynomial Trajectory	Generate minimum snap polynomial trajectories through multiple waypoints *(Since R2022a)*
Read UAV Trajectory	Generate translation and rotation samples from UAV trajectory for 3D simulation *(Since R2024b)*



Tune Waypoint Follower for Fixed-Wing UAV

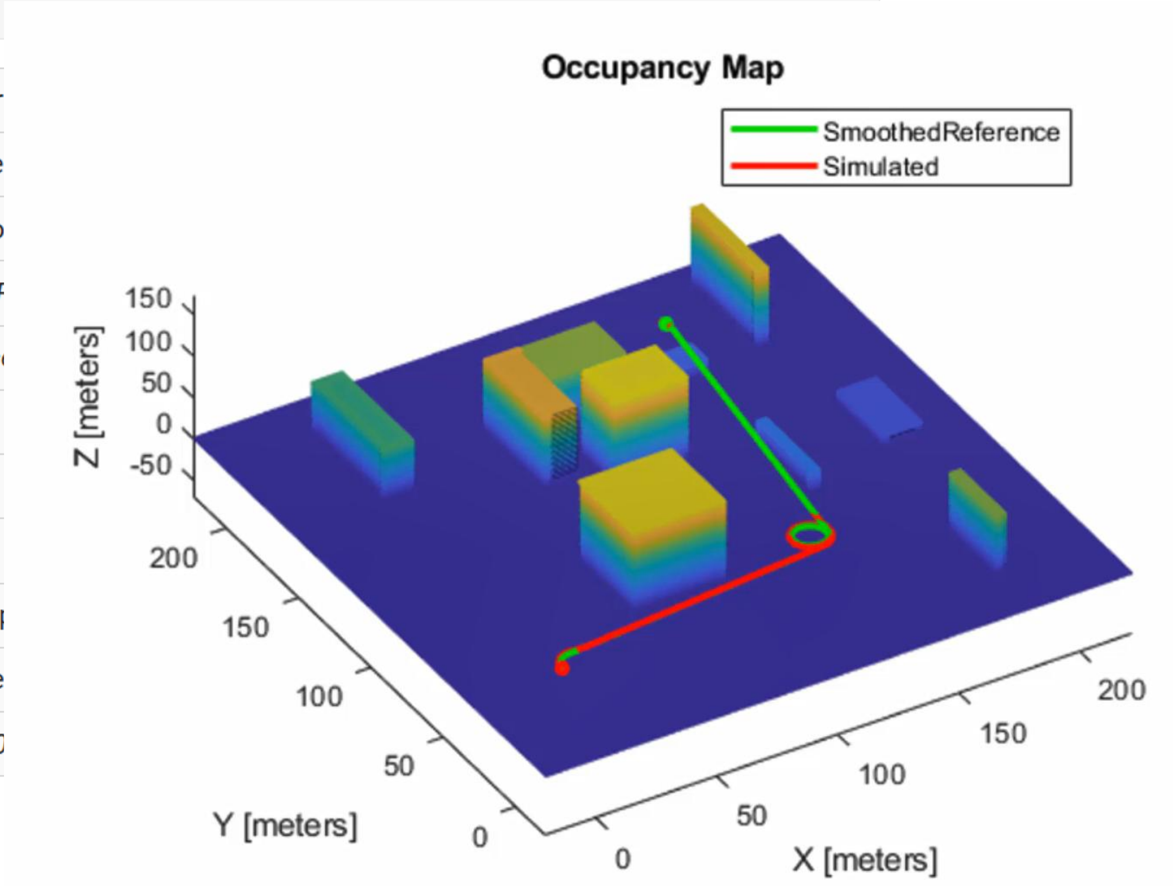


Autonomous UAV motion controls



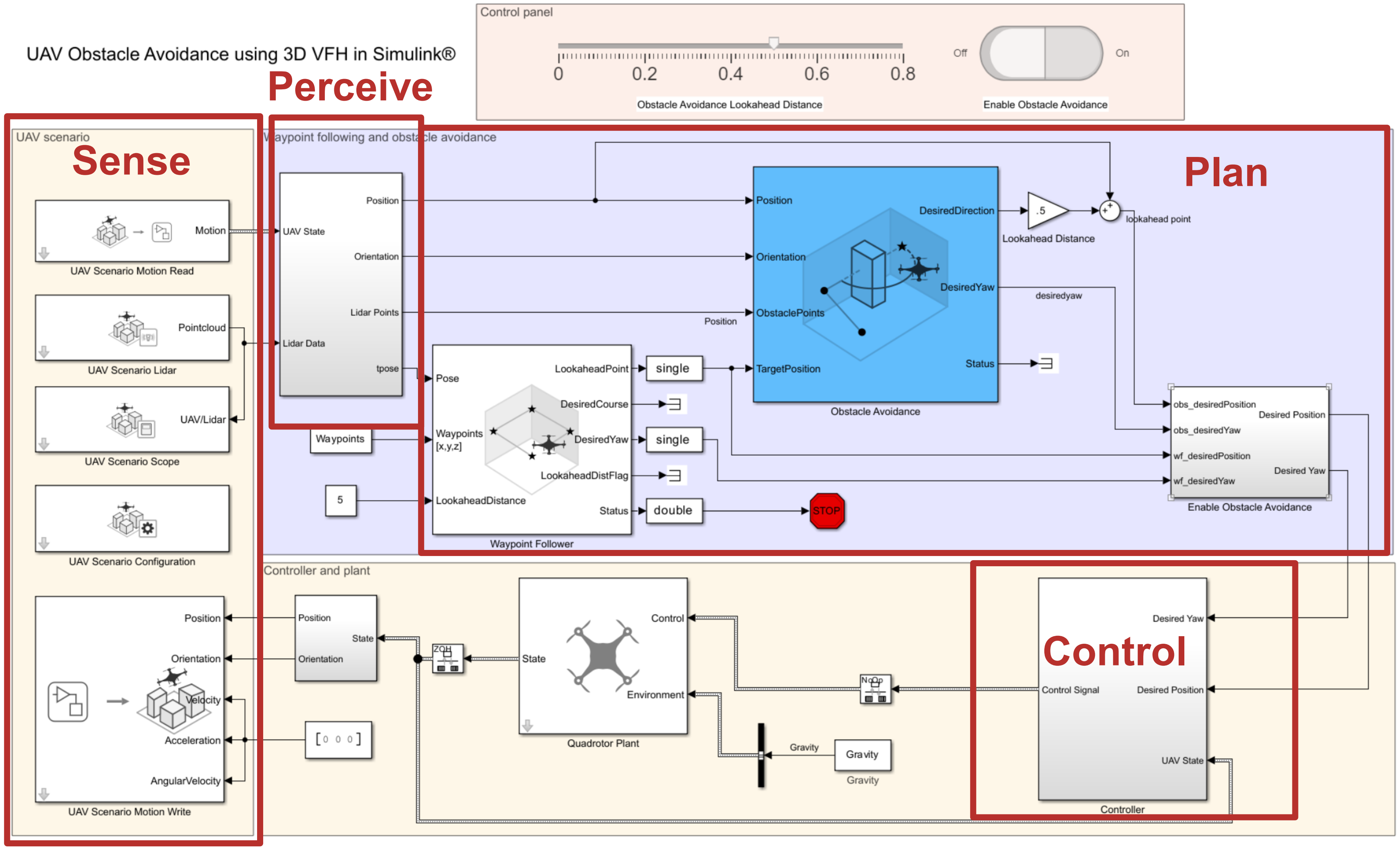
Path Planners

plannerRRT	Create an RRT planner for geometr
plannerRRTStar	Create an optimal RRT path planne
plannerBiRRT	Create bidirectional RRT planner fo
plannerControlRRT	Control-based RRT planner *(Since F*
plannerAStar	Graph-based A* path planner *(Sinc*
plannerAStarGrid	A* path planner for grid map
plannerHybridAStar	Hybrid A* path planner
plannerPRM	Create probabilistic roadmap path
plannerMPNET	Create MPNet based bidirectional
plannerBenchmark	Benchmark path planners using ge
navGraph	Create navGraph object *(Since R20*

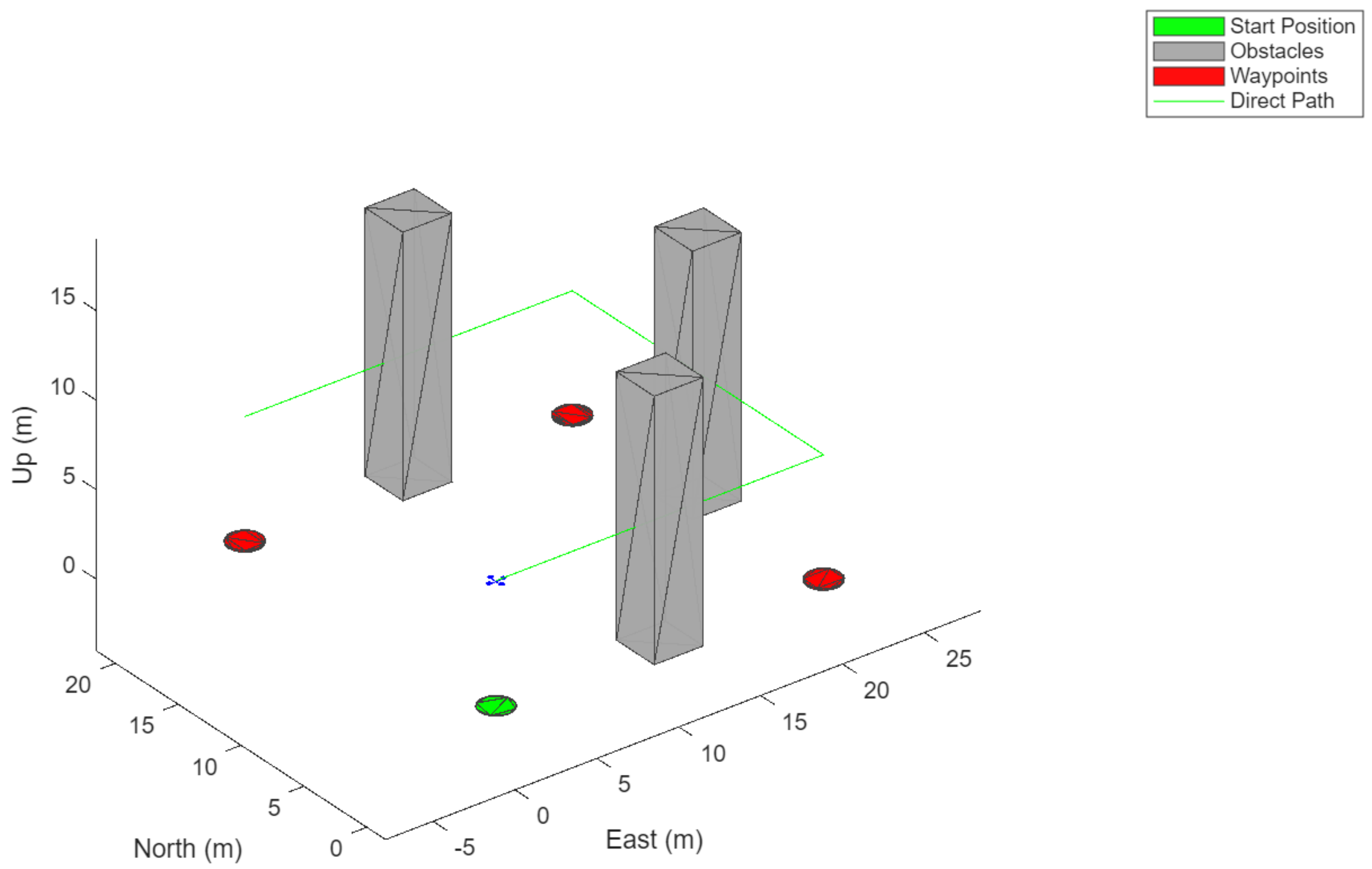


CONTROL

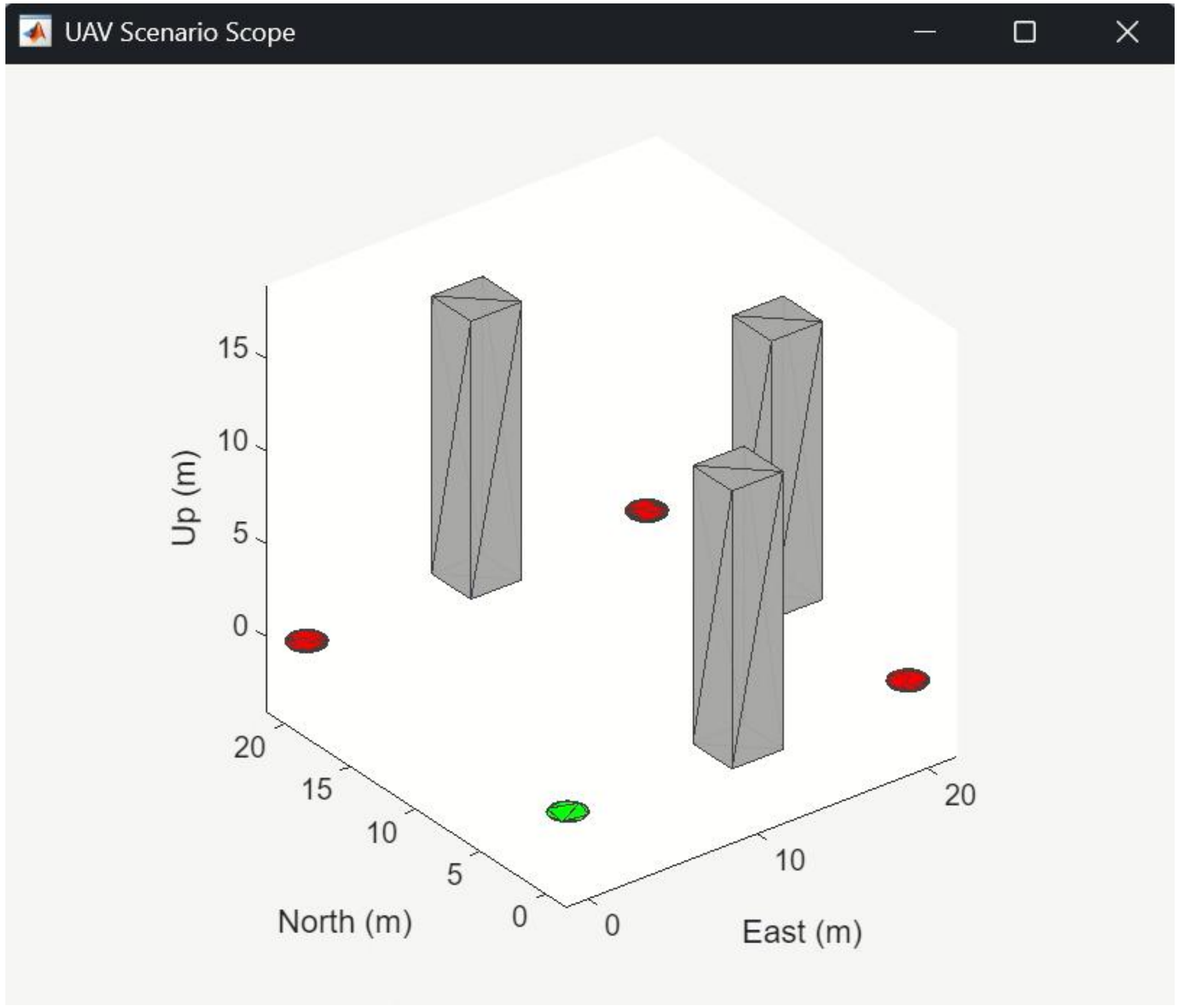
Implementing control algorithms with Simulink



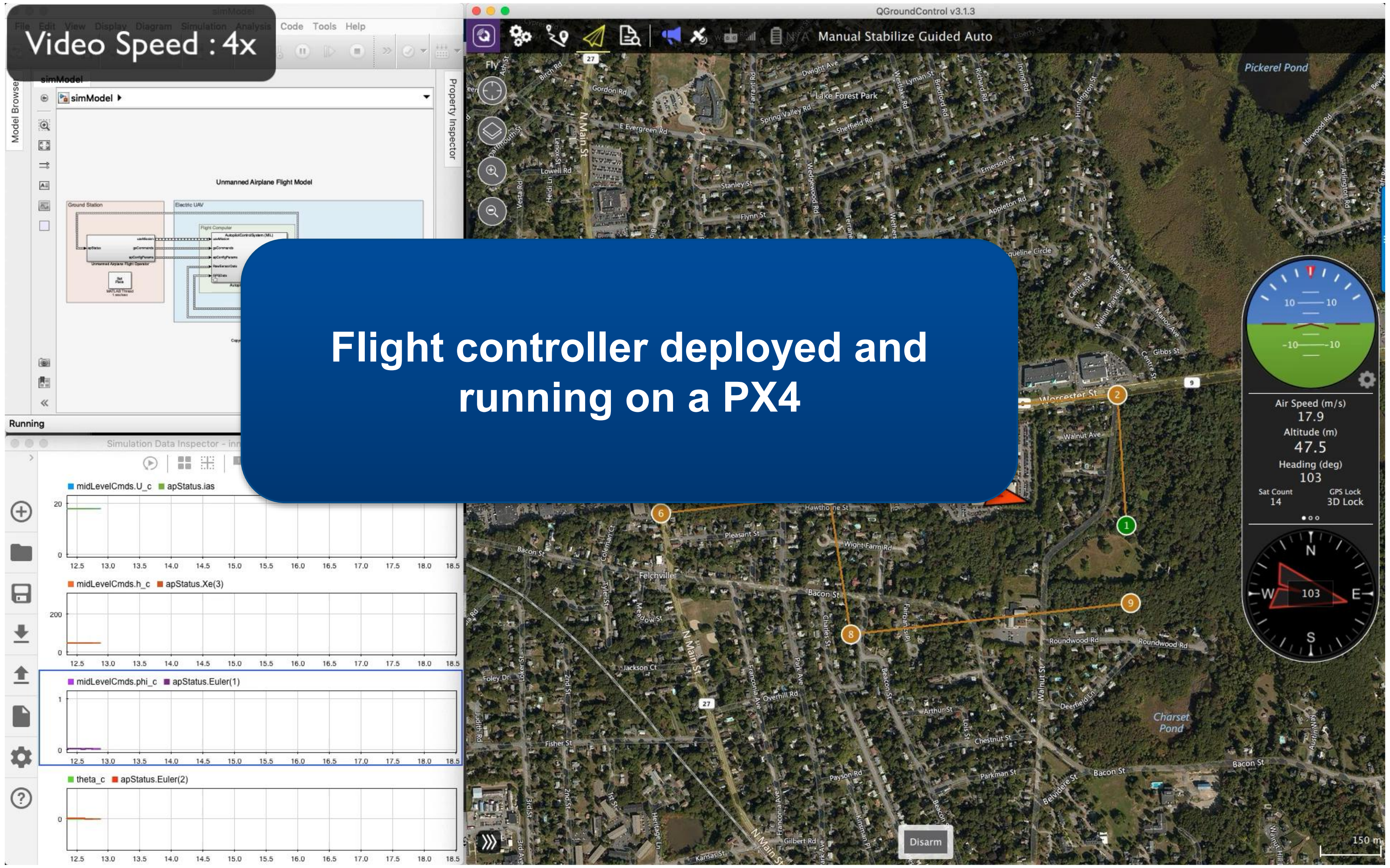
Implementing control algorithms with Simulink



Implementing control algorithms with Simulink

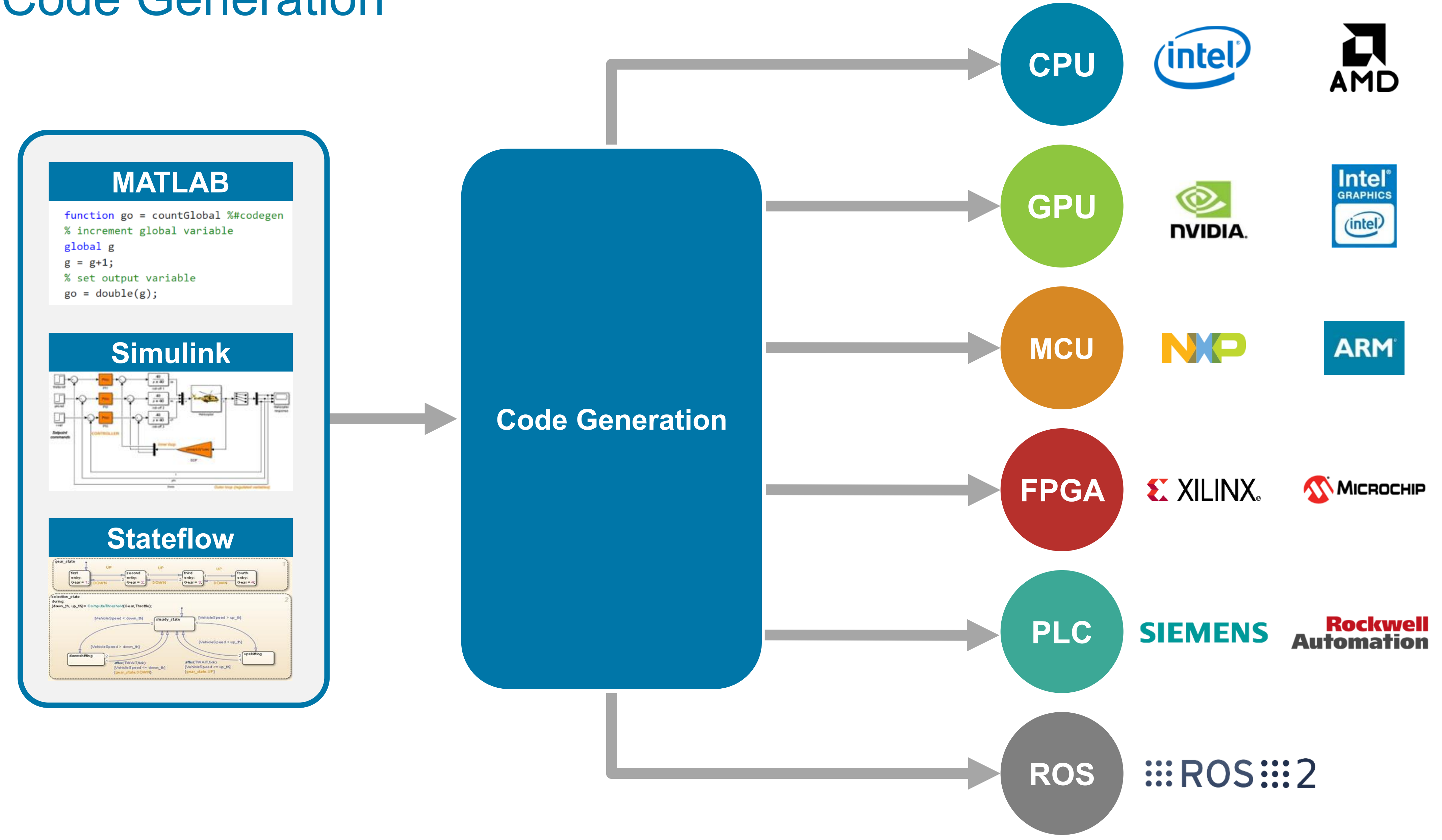


Control – Path Following



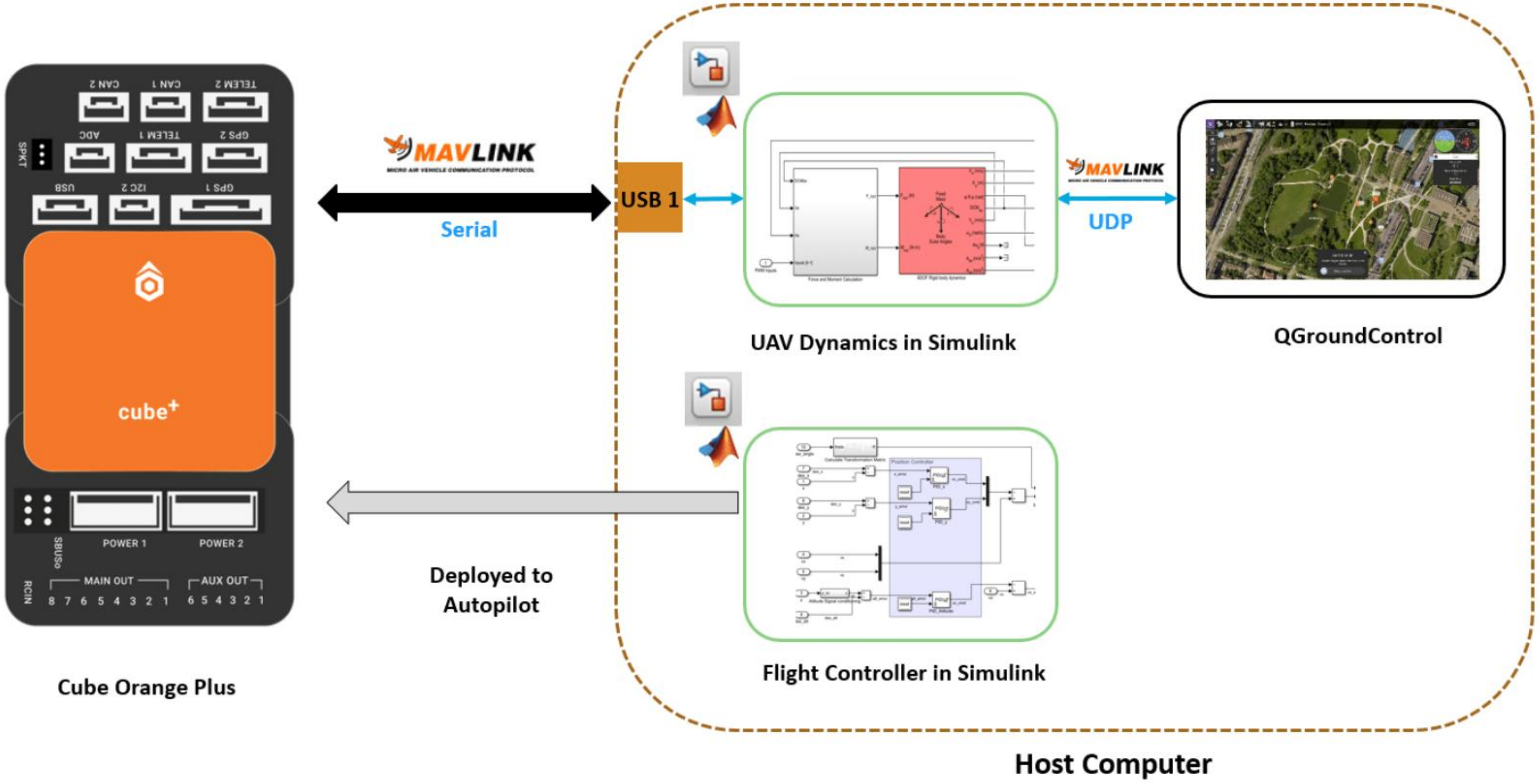
Flight controller deployed and running on a PX4

Code Generation



```
function go = countGlobal %#codegen
% increment global variable
global g
g = g+1;
% set output variable
go = double(g);
```

Control – PX4 Hardware-in-the-loop Simulation

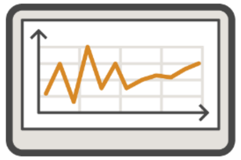


Building Blocks for UAV Simulation

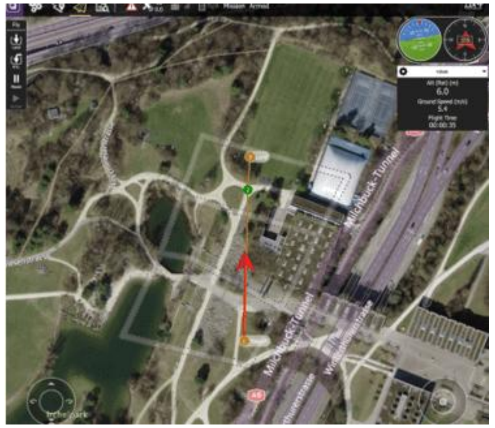
Plan Mission

Generate flight path for the mission

Simulated Waypoints



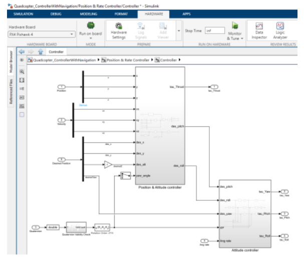
Ground Control Station



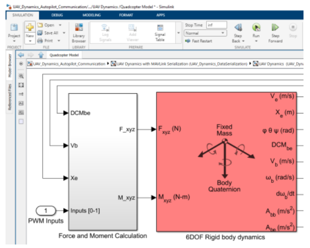
Design & Simulate

Design flight controller and simulate plant behavior in virtual scenarios

Flight Controller



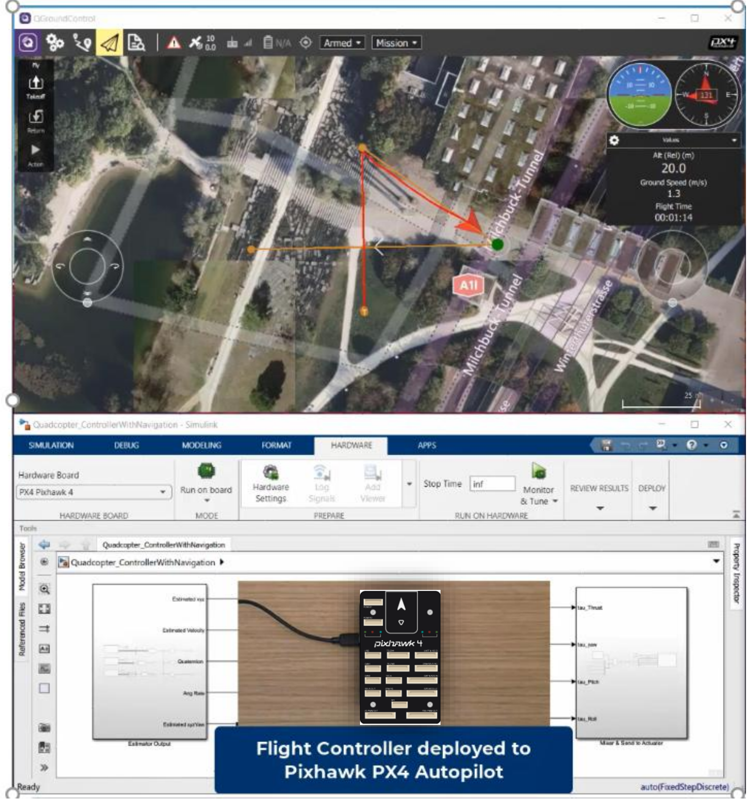
Plant Model



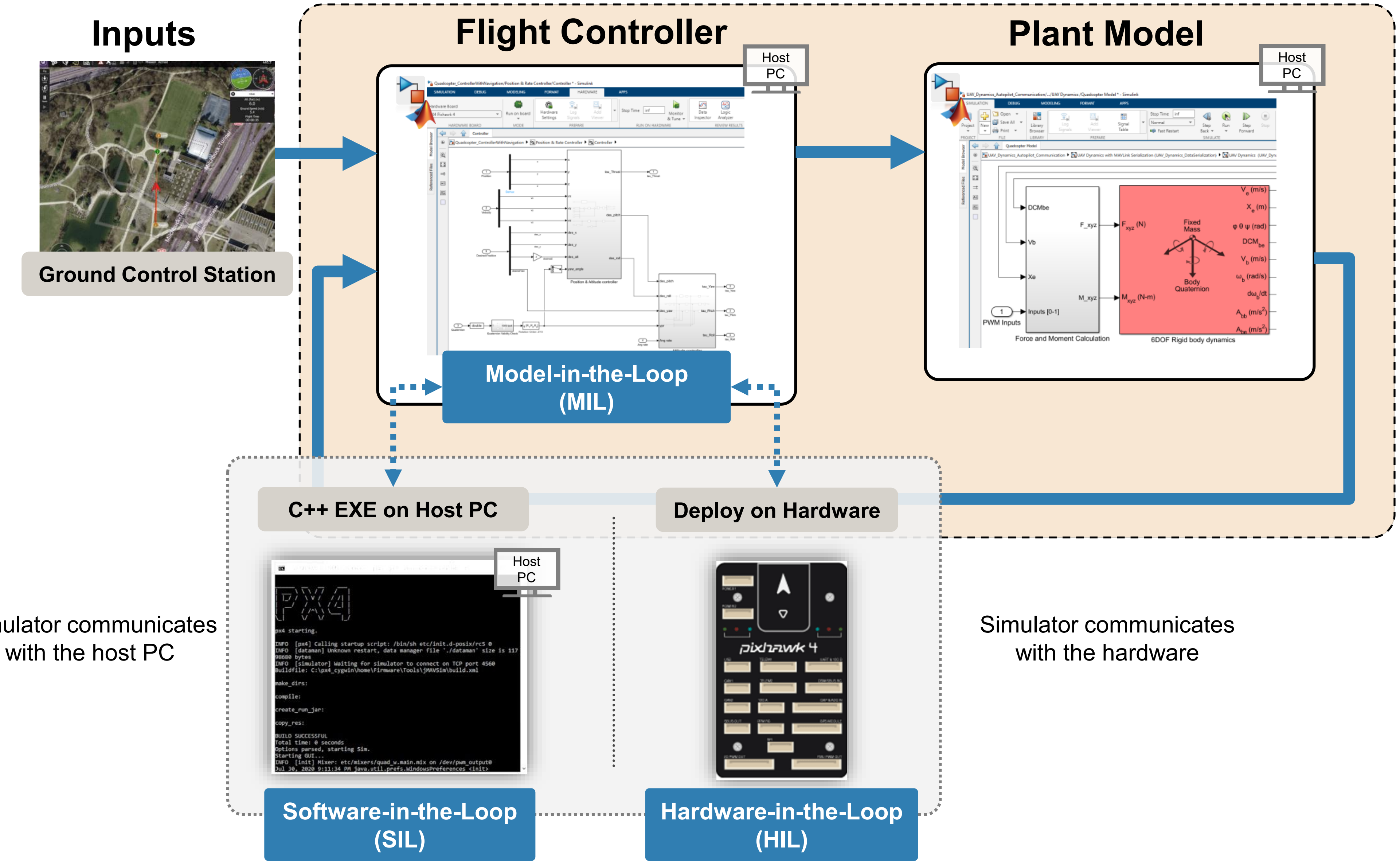
Scenario Simulation

Validate & Deploy

Deploy flight controller and autonomy algorithms to the platform



MIL, SIL and HIL Workflows for UAV Simulation

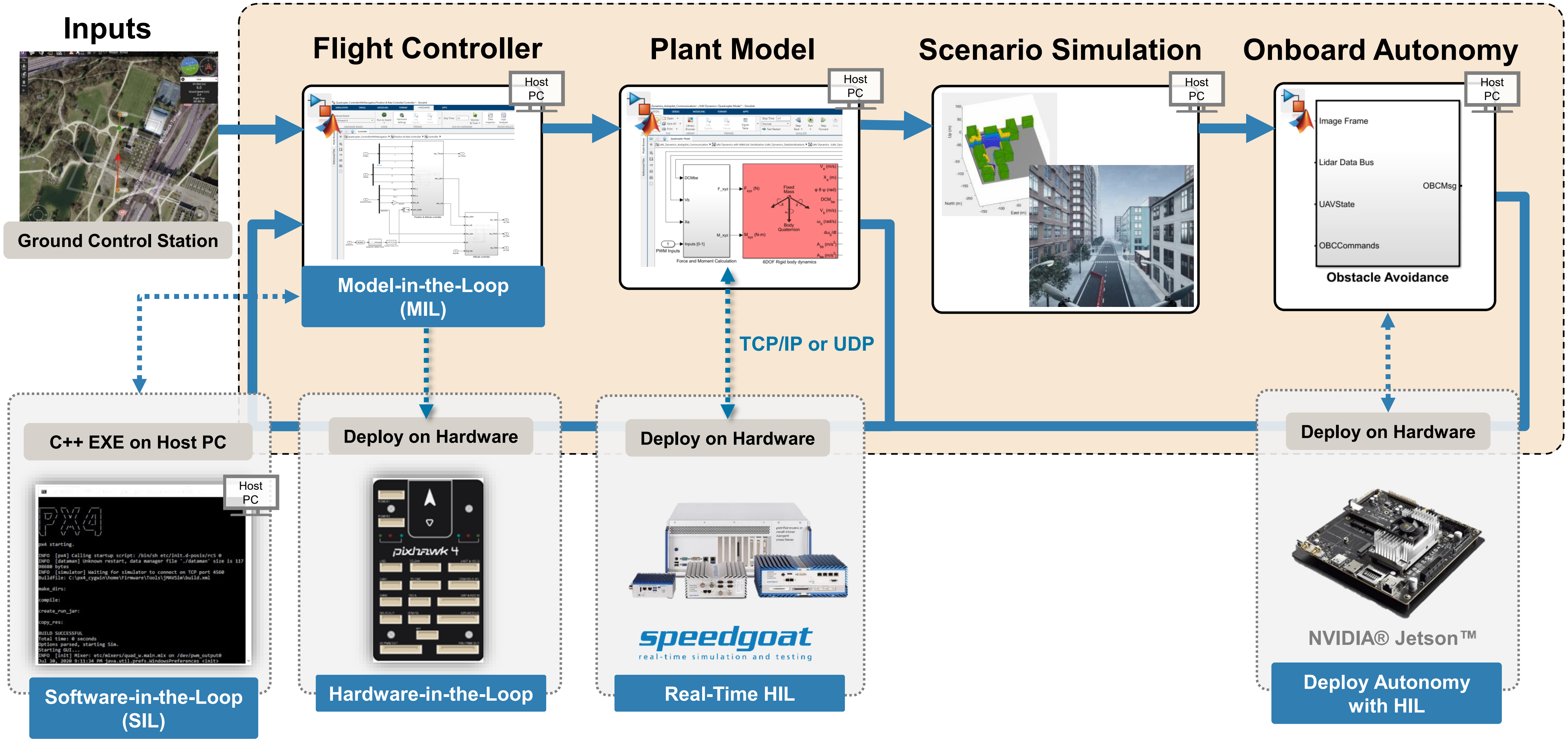


```
px4 starting.

INFO  [px4] Calling startup script: /bin/sh etc/init.d-posix/rcS 0
INFO  [dataman] Unknown restart, data manager file './dataman' size is 117
98680 bytes
INFO  [simulator] Waiting for simulator to connect on TCP port 4560
Buildfile: C:\px4_cygwin\home\Firmware\Tools\jMAVSim\build.xml

make_dirs:

compile:

create_run_jar:

copy_res:

BUILD SUCCESSFUL
Total time: 0 seconds
Options parsed, starting Sim.
Starting GUI...
INFO  [init] Mixer: etc/mixers/quad_w.main.mix on /dev/pwm_output0
Jul 30, 2020 9:11:34 PM java.util.prefs.WindowsPreferences <init>
```

Simulator communicates with the host PC

Simulator communicates with the hardware

MIL, SIL and HIL Workflows for UAV Simulation

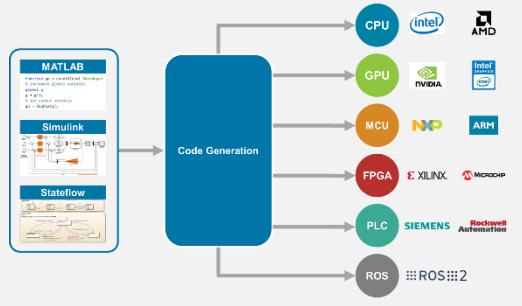


```
px4 starting.

INFO  [px4] Calling startup script: /bin/sh etc/init.d-posix/rcS 0
INFO  [dataman] Unknown restart, data manager file './dataman' size is 117
98680 bytes
INFO  [simulator] Waiting for simulator to connect on TCP port 4560
Buildfile: C:\px4_cygwin\home\Firmware\Tools\jMAVSim\build.xml

make_dirs:

compile:

create_run_jar:

copy_res:

BUILD SUCCESSFUL
Total time: 0 seconds
Options parsed, starting Sim.
Starting GUI...
INFO  [init] Mixer: etc/mixers/quad_w.main.mix on /dev/pwm_output0
Jul 30, 2020 9:11:34 PM java.util.prefs.WindowsPreferences <init>
```

MathWorks

Develop Embedded Software for Autonomous Vehicles

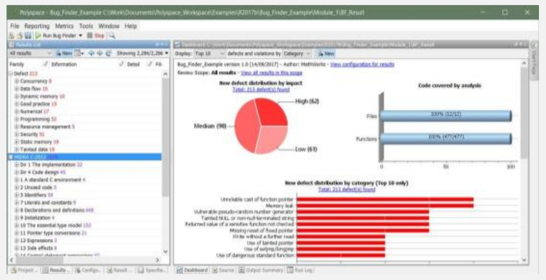
Code Generation



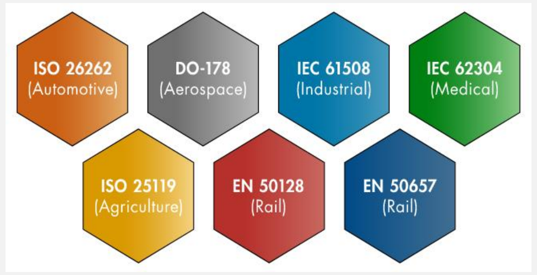
Software Standard Compliance

- MISRA C:2004
- MISRA C++:2008
- Naming Rules
- MISRA C:2012
- AUTOSAR C++-14
- JSF AV C++
- CERT C
- CERT C++
- ISO/IEC TS 17961

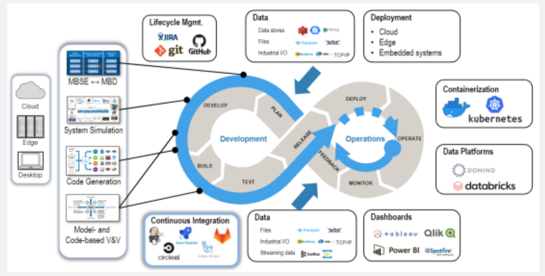
Static Code Analysis



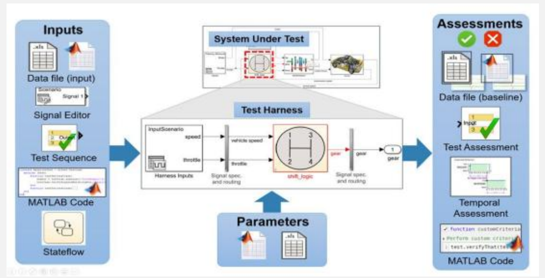
Functional Safety



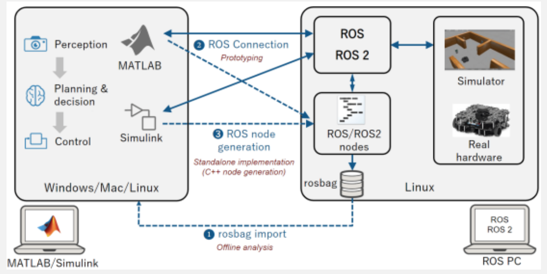
Continuous Integration



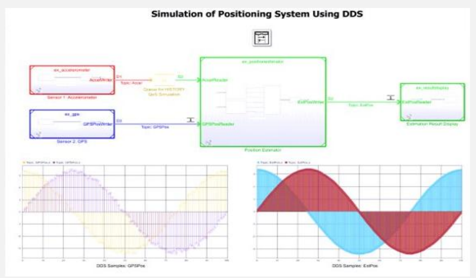
Automated Testing



ROS / ROS2

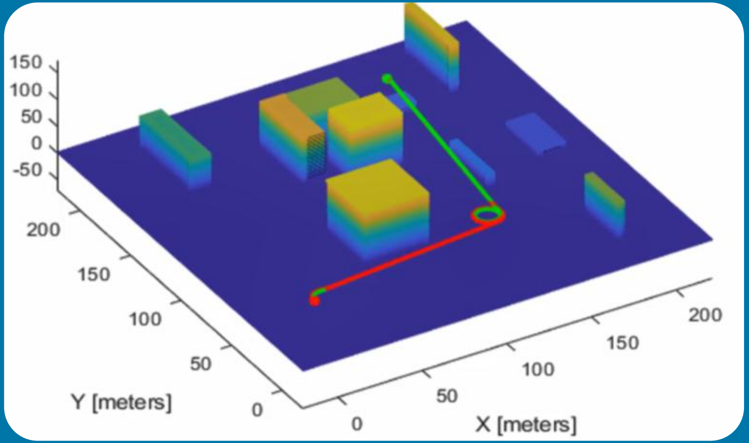


DDS

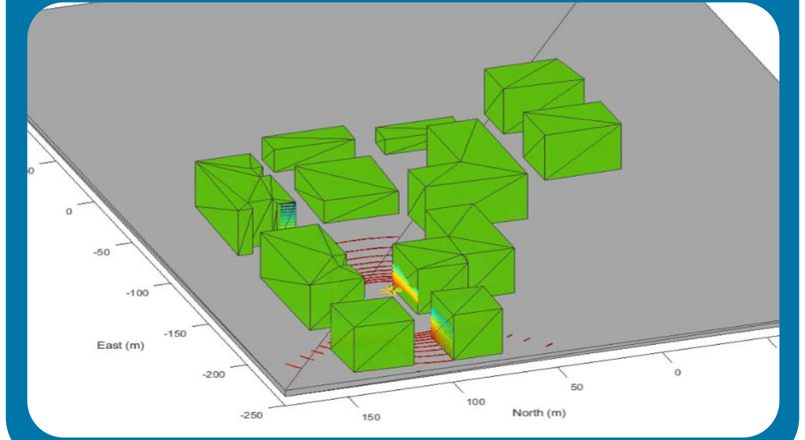


Shipping examples

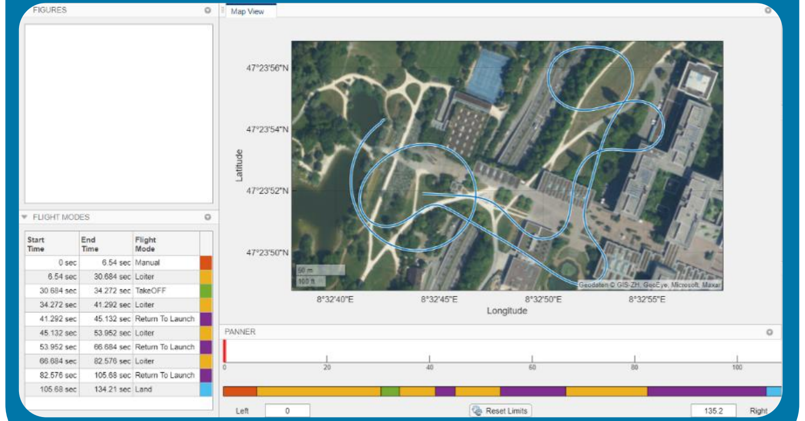
UAV Algorithms for Planning and Control



Scenario Design & Low-Fidelity Sensor Simulation



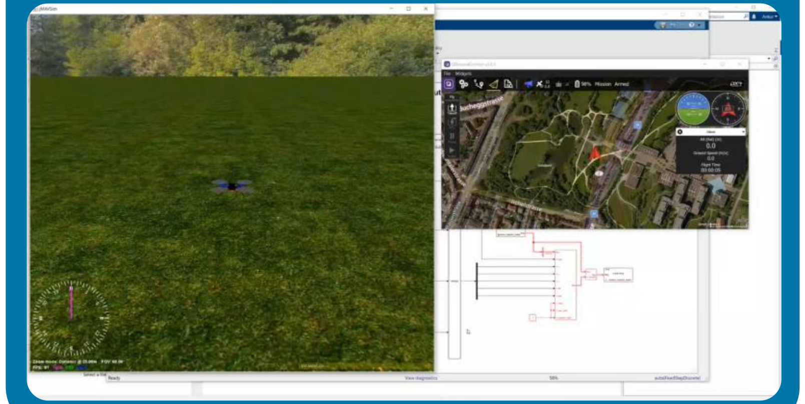
Flight Telemetry Data Analysis Flight Log Analyzer App



FLIGHT MODES

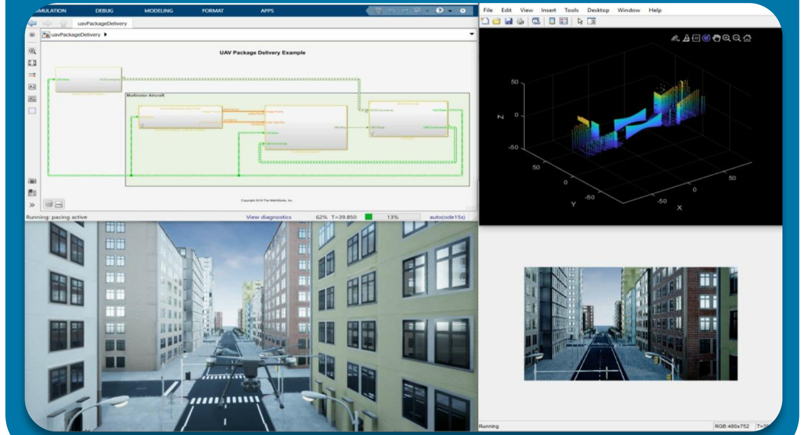
Start Time	End Time	Flight Mode	
0 sec	6.54 sec	Manual	🟥
6.54 sec	30.684 sec	Loiter	🟨
30.684 sec	34.272 sec	TakeOFF	🟩
34.272 sec	41.292 sec	Loiter	🟨
41.292 sec	45.132 sec	Return To Launch	🟪
45.132 sec	53.952 sec	Loiter	🟨
53.952 sec	66.684 sec	Return To Launch	🟪
66.684 sec	82.576 sec	Loiter	🟨
82.576 sec	105.68 sec	Return To Launch	🟪
105.68 sec	134.21 sec	Land	🟦

Connectivity and Deployment with MAVLink and PX4

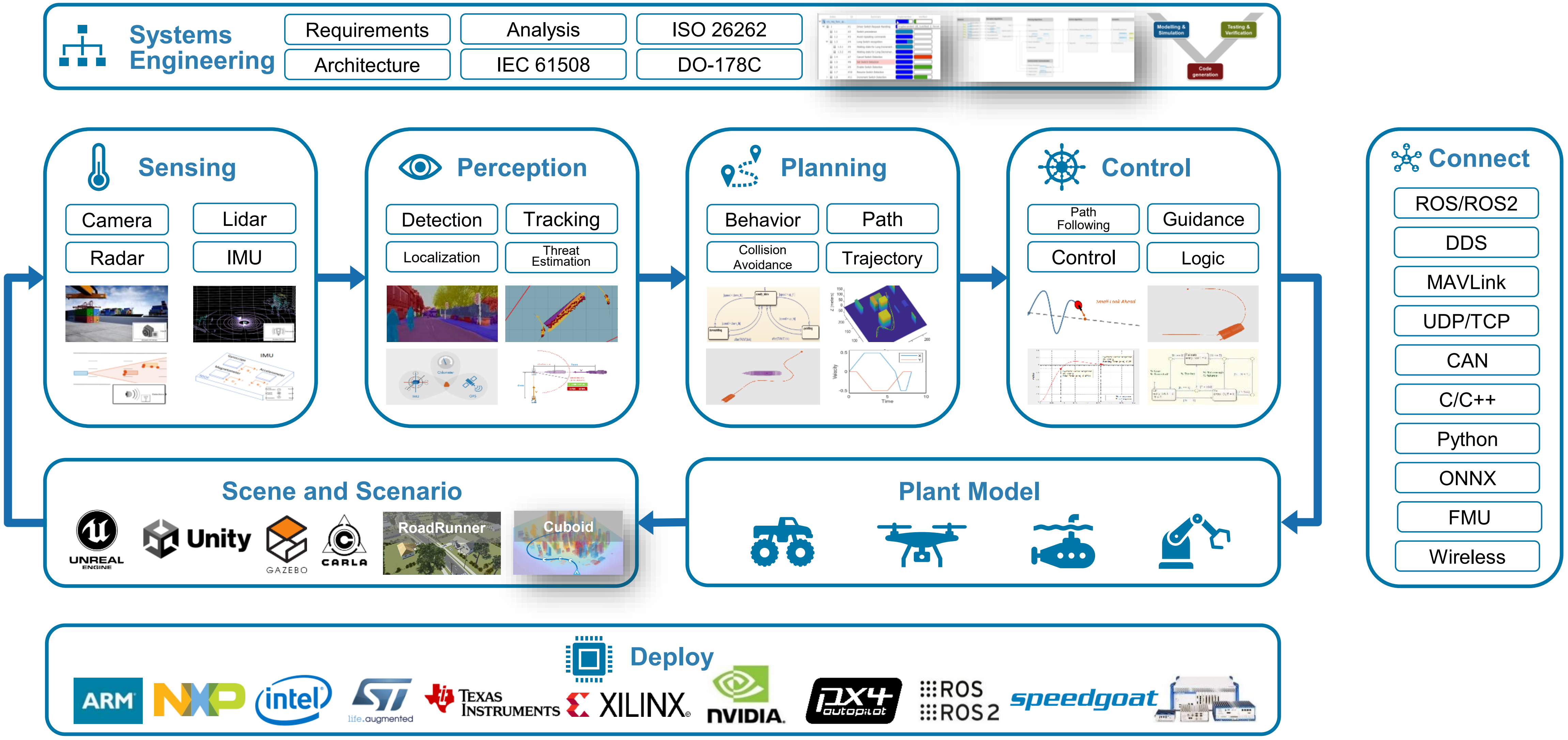


Unreal Engine Co-Simulation with Sensor Models

Reference Applications



Autonomous Vehicle Development Workflow



MathWorks®

Accelerating the pace of engineering and science