

MATLAB EXPO 2017

KOREA

4월 27일, 서울

등록 하기 matlabexpo.co.kr

다중 센서 기반 자율시스템의 모델 설계 및 개발

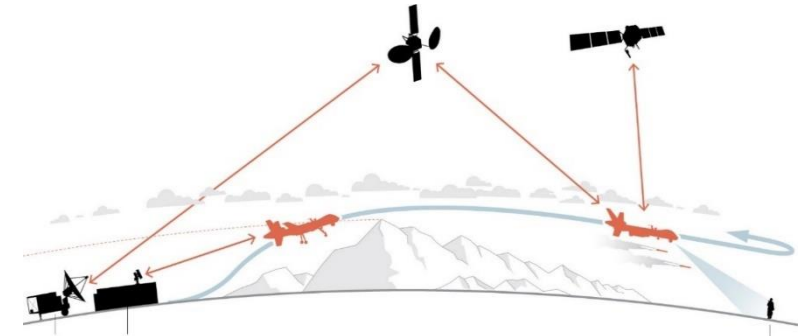
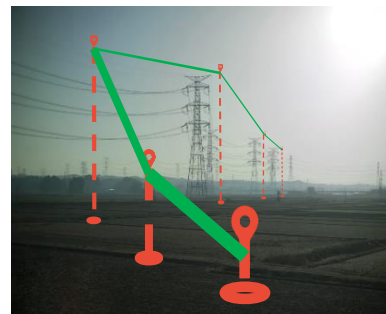
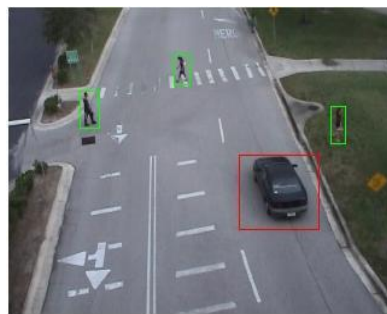
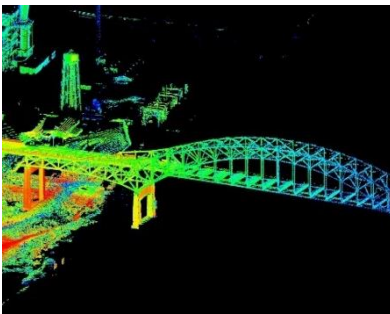
이제훈 차장

What we will see today...



Functional Segmentation of Autonomous System

Aircraft/ Platform	Sense	Perceive	Plan & Decide	Control	Connect/ Communicate
- Control Surfaces, slats, flaps - Lifting Body - Landing Gear - Battery - Power Management	- Radar - Camera - Lidar - EO/IR - IMU - GPS-INS - HW Certification	- Environment mapping - Classification - Segmentation - Object Detection - Sensor Fusion	- Object Avoidance - Path & motion planning - SLAM	- Guidance, Navigation & Control - Flight SW certification	- Communication with ground operator - Multi-agent communication - Satellite data link



Functional Segmentation of Autonomous System

Customer Groups

Aircraft	"Autonomous Algorithms"				Communication
Platform	Perception		Planning		Connect/ Communicate
	Sense	Perceive	Plan & Decide	Control	
• Control Surfaces, slats, flaps • Lifting Body • Landing Gear • Battery • Power Management	• Radar • Camera • Lidar • EO/IR • IMU • GPS-INS • HW Certification	• Environment mapping • Classification • Segmentation • Object Detection • Sensor Fusion	• Object Avoidance • Path & motion planning • SLAM	• Guidance, Navigation & Control • Flight SW certification	• Communication with ground operator • Multi-agent communication • Satellite data link



Autonomous System Development Workflow

Aerodynamics and Flight Control

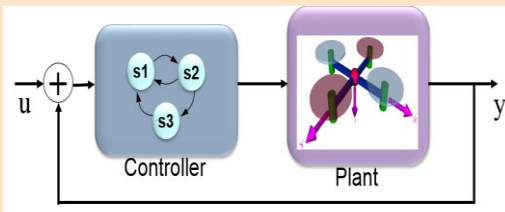
Autonomous algorithm

Test and Refine in Simulation

Test and Refine on Real Robot

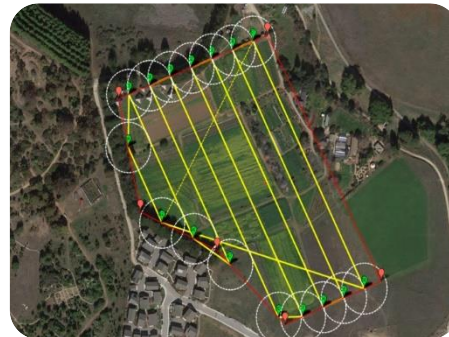
Challenge 1:

Understand the dynamics with control algorithm



Challenge 2:

Design and Verify autonomous algorithms

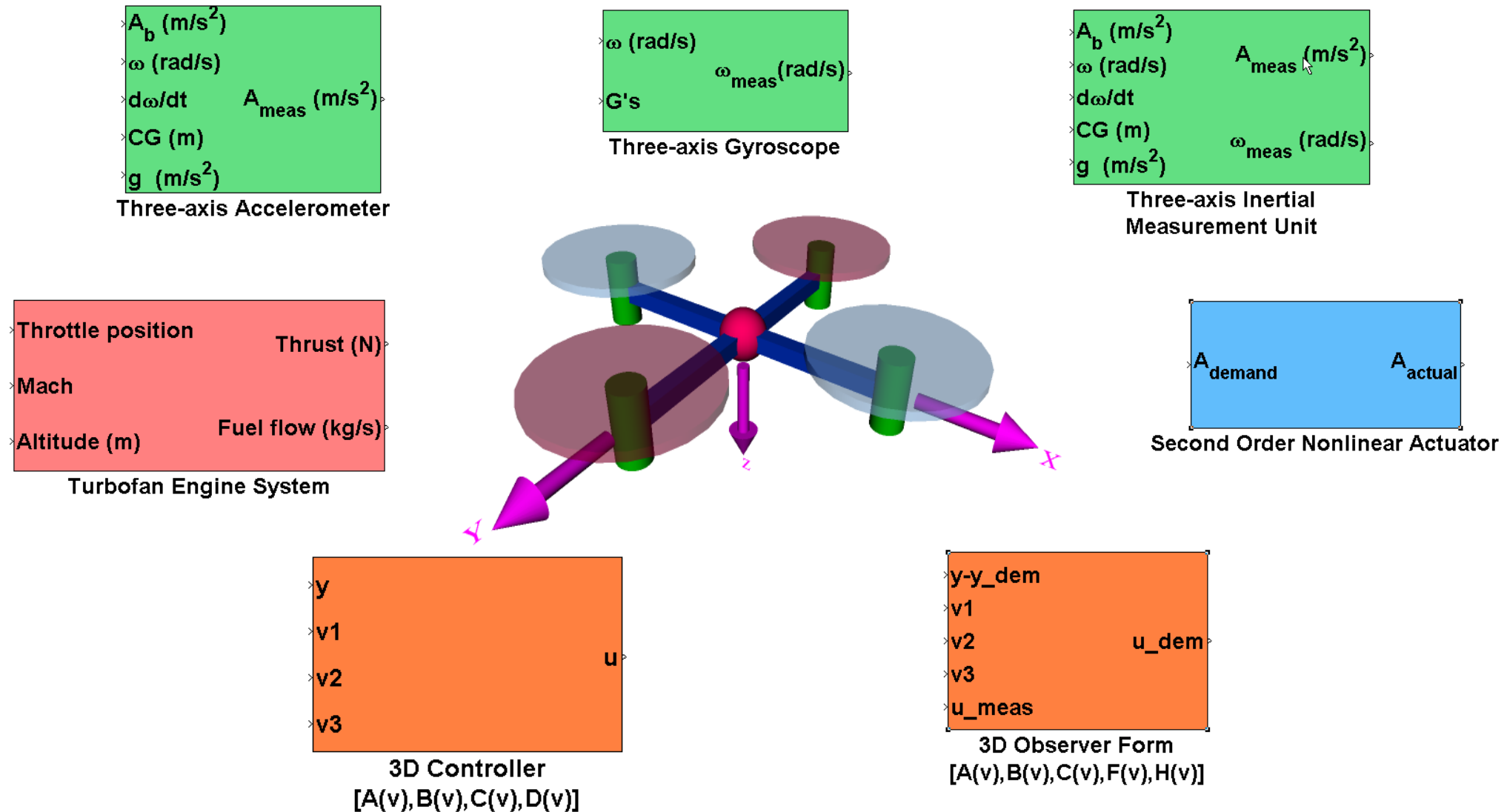


Challenge 3:

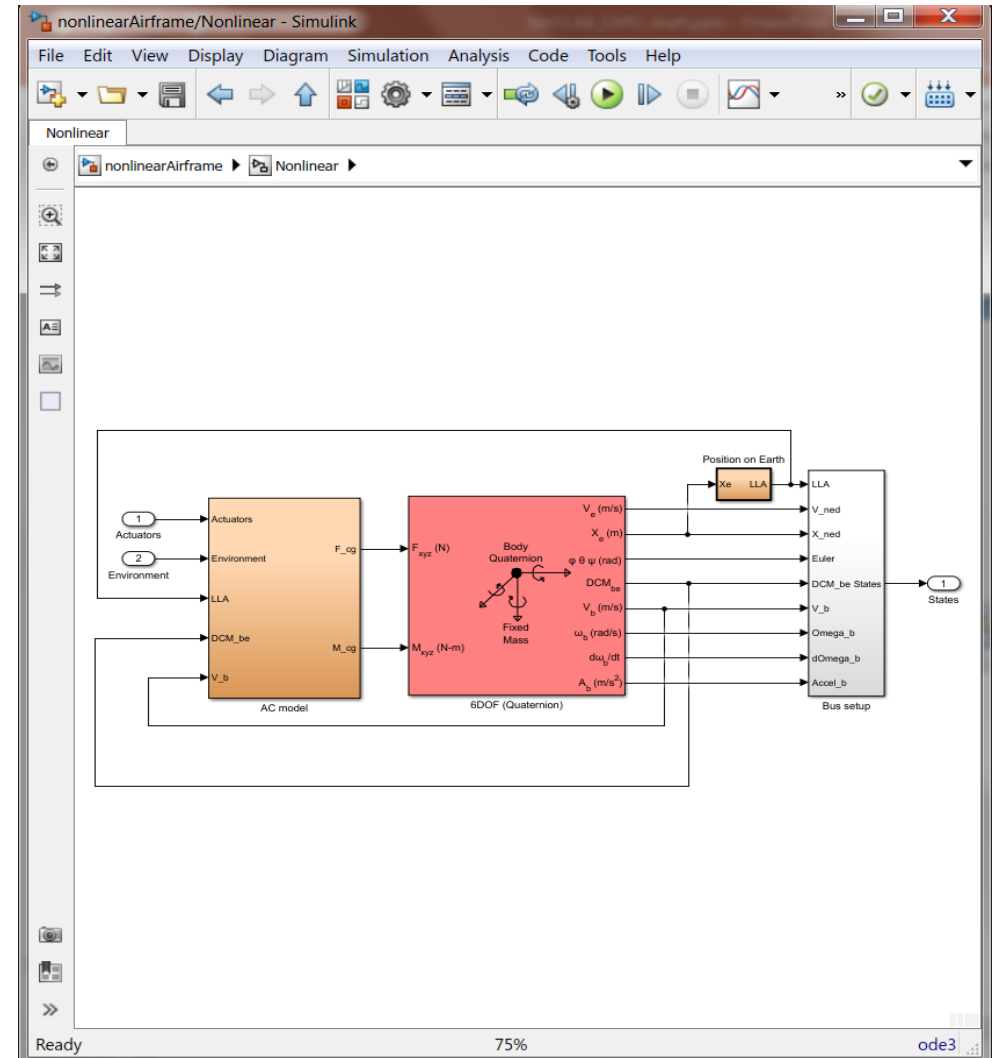
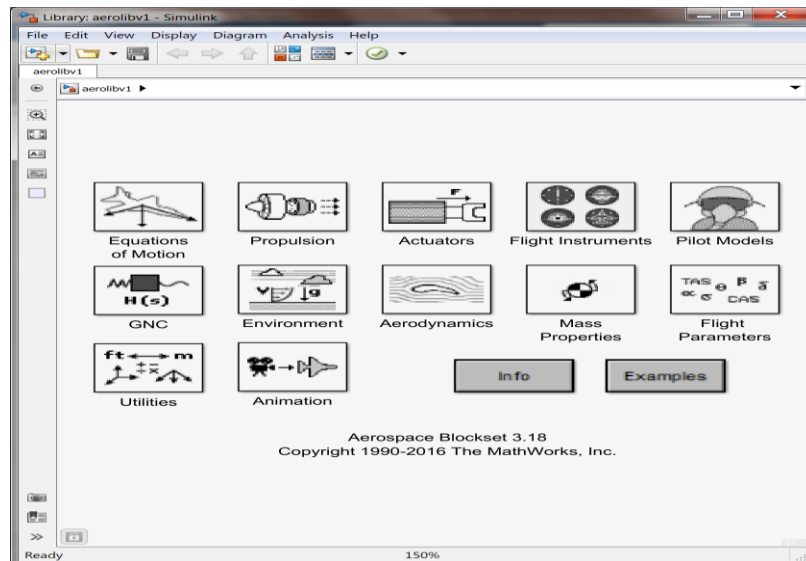
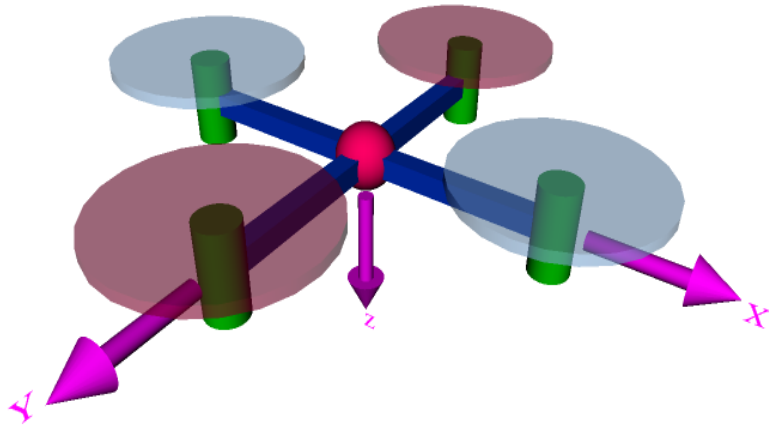
Verify and Implement the algorithm on to a real hardware



Design Control algorithm with Dynamics



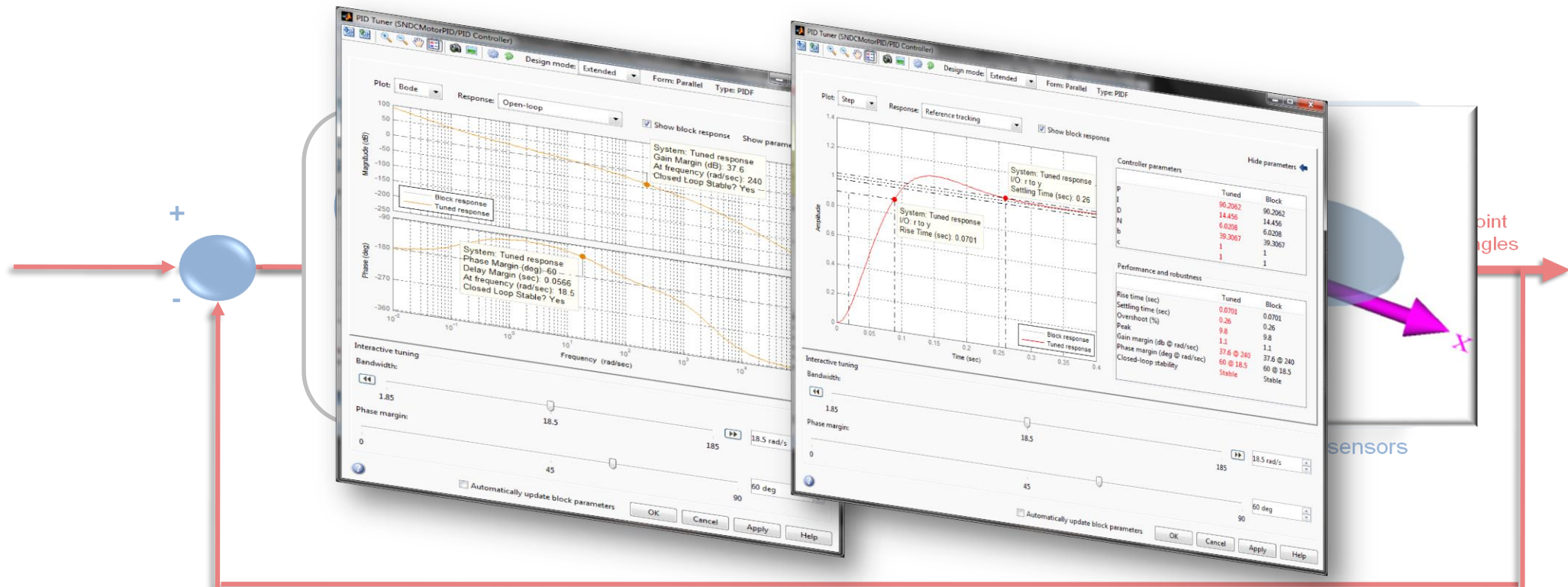
Design Control algorithm with Dynamics



Design Control algorithm with Dynamics

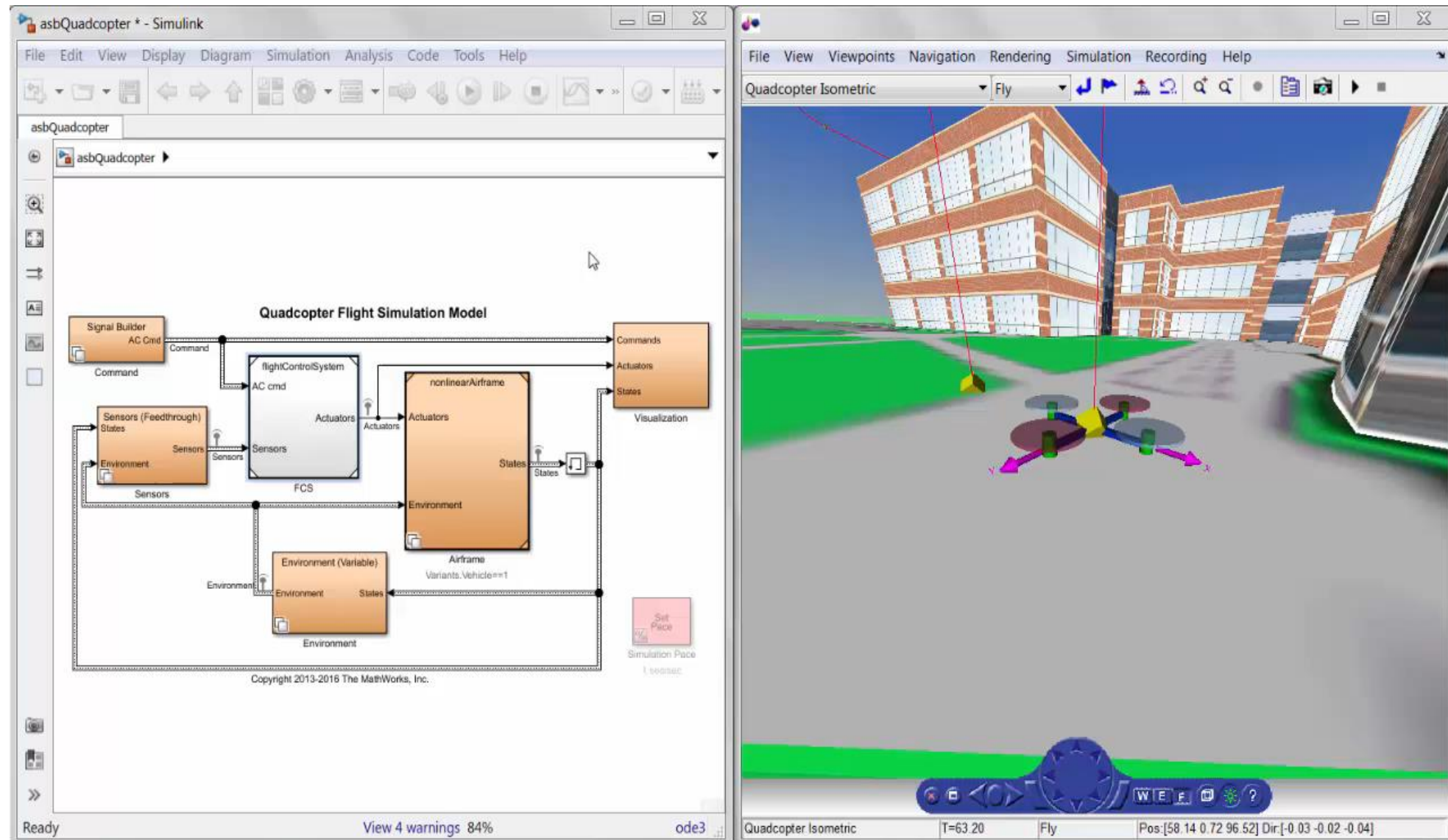
Simulink Control Design & Control System Toolbox

Simulating plant and controller *in one environment*
Optimize system-level performance & Closed-loop simulation



Simulink Control Design & Control System Toolbox
automatically linearize the plant, design and tune your PID controllers

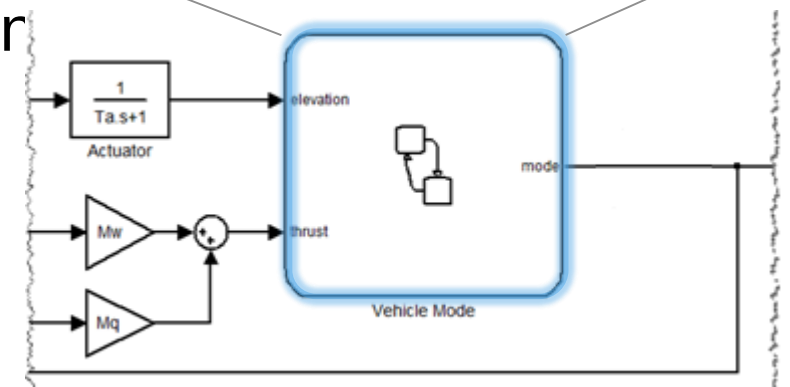
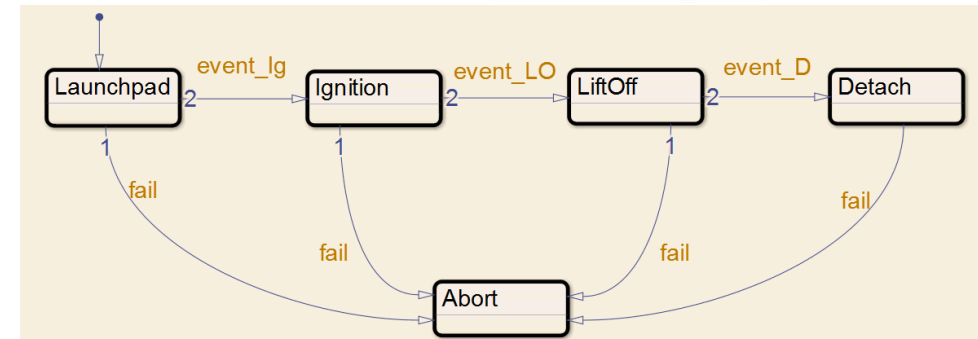
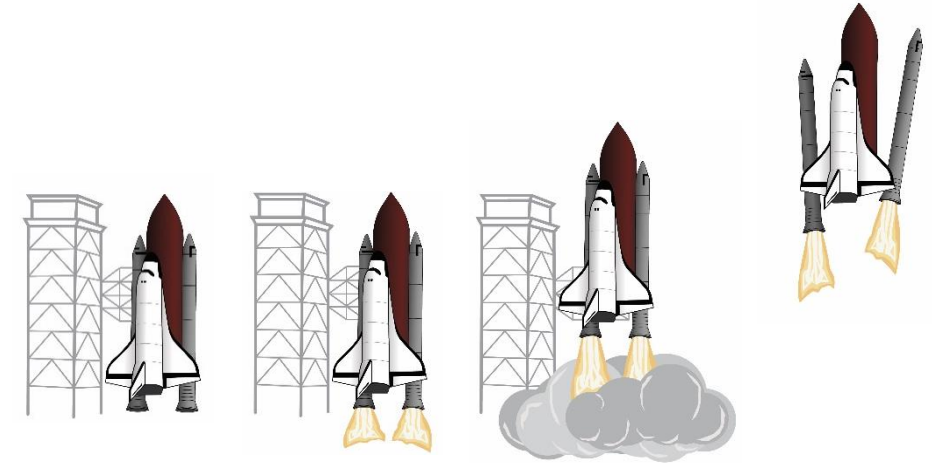
Design Control algorithm with Dynamics



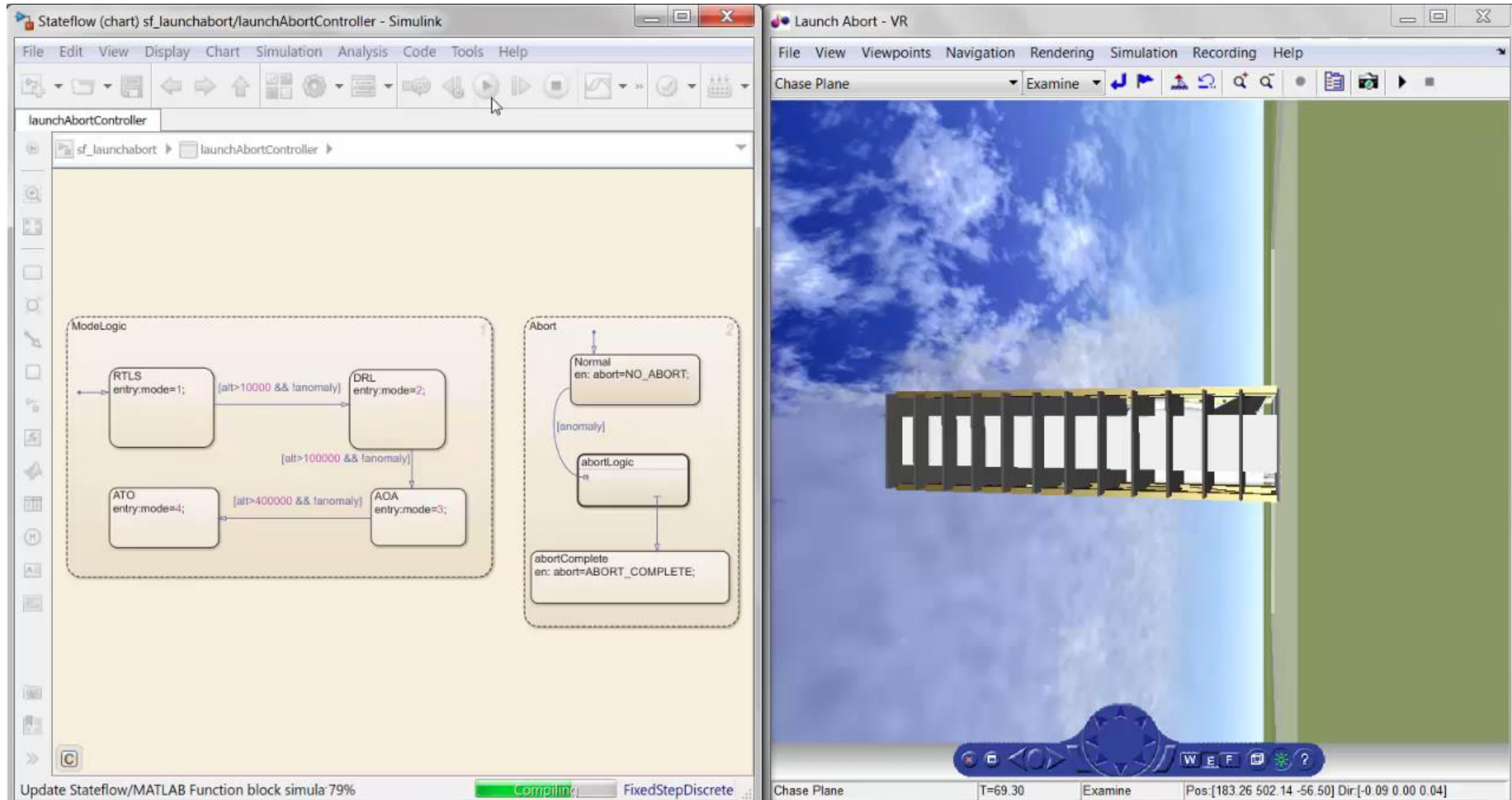
Design **Control algorithm** with Dynamics

Stateflow

- Model and simulate decision logic
 - supervisory control
 - task scheduling
 - fault management
- Develop mode-logic
 - using state machines and flow charts
- See how the logic behaves with diagram animation
 - and integrated debugger



Design Control algorithm with Dynamics

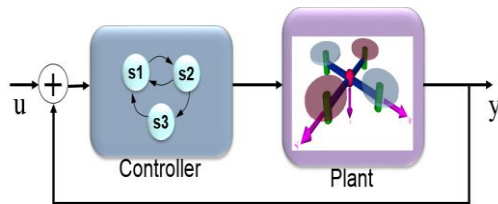


Autonomous System Development Workflow

**Aerodynamics and
Flight Control**

Challenge 1:

Understand
the dynamics
with control algorithm



**Autonomous
algorithm**

Challenge 2:

Design and Verify
autonomous
algorithms



**Test and Refine in
Simulation**

Challenge 3:

Verify and Implement
the algorithm
on to a real hardware



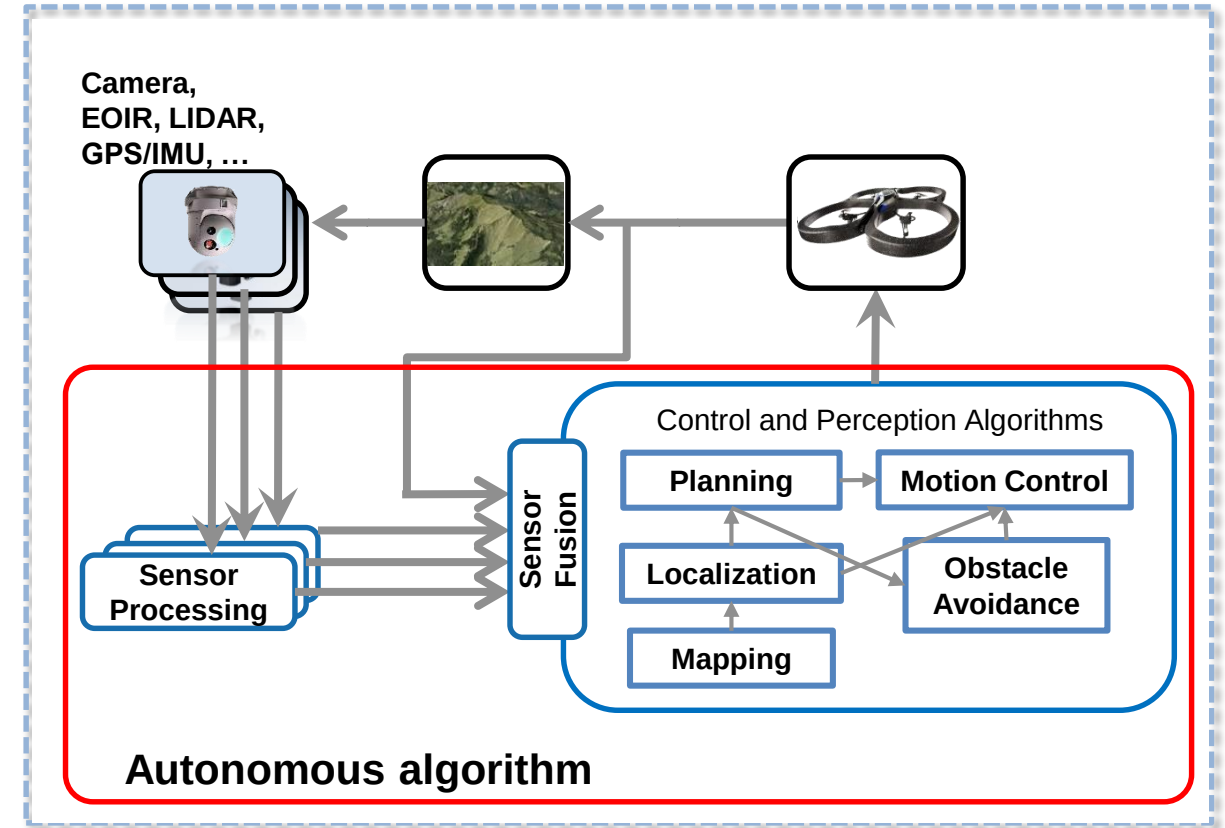
**Test and Refine on
Real Robot**



How to develop **Autonomous algorithms**?



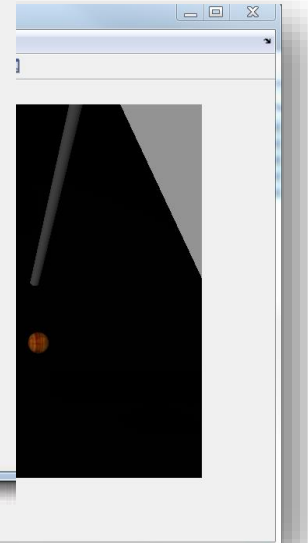
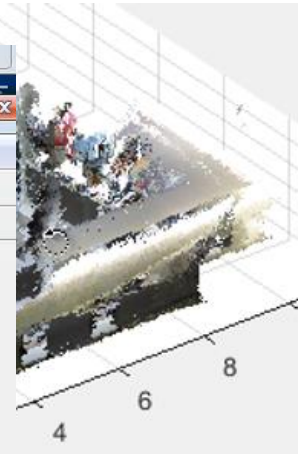
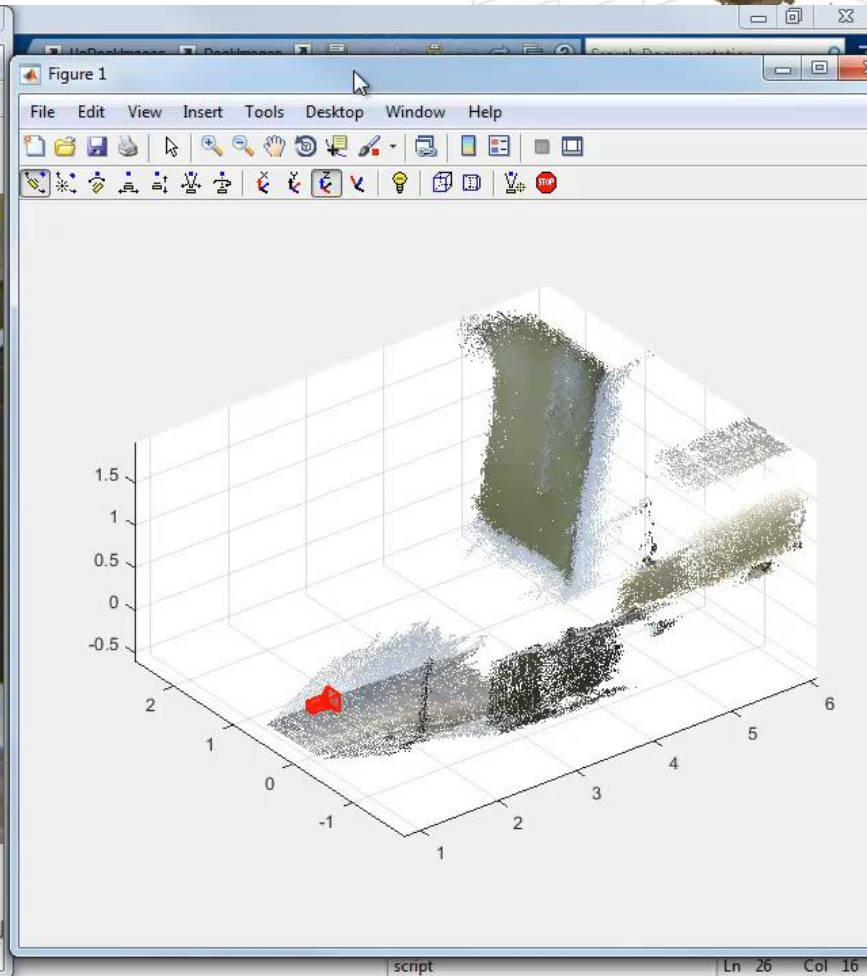
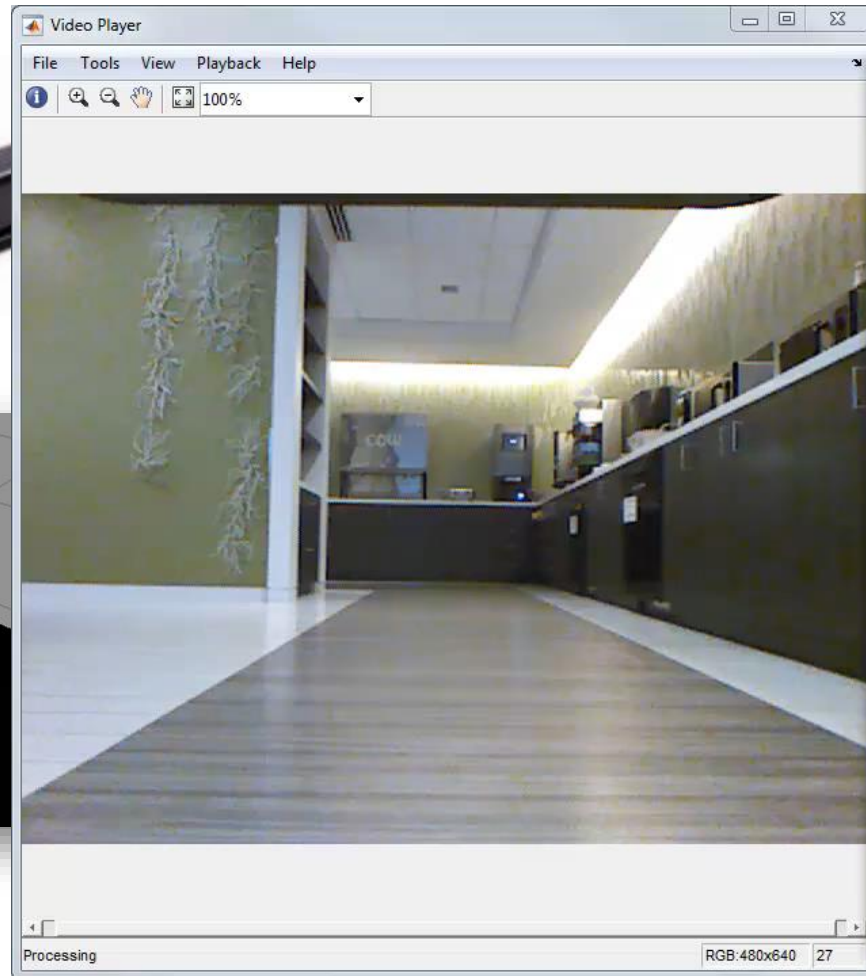
Autoroute, gated, post-process automatically



Efficient, speed, endurance, automation

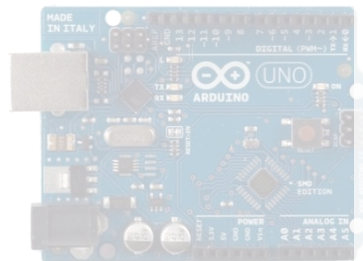
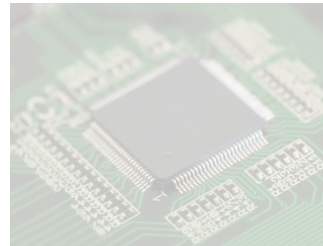
How to develop Autonomous algorithms?

Manage Sensor data



How to develop Autonomous algorithms?

Manage Sensor data



GiG
VISION

Data & RF

Embedded

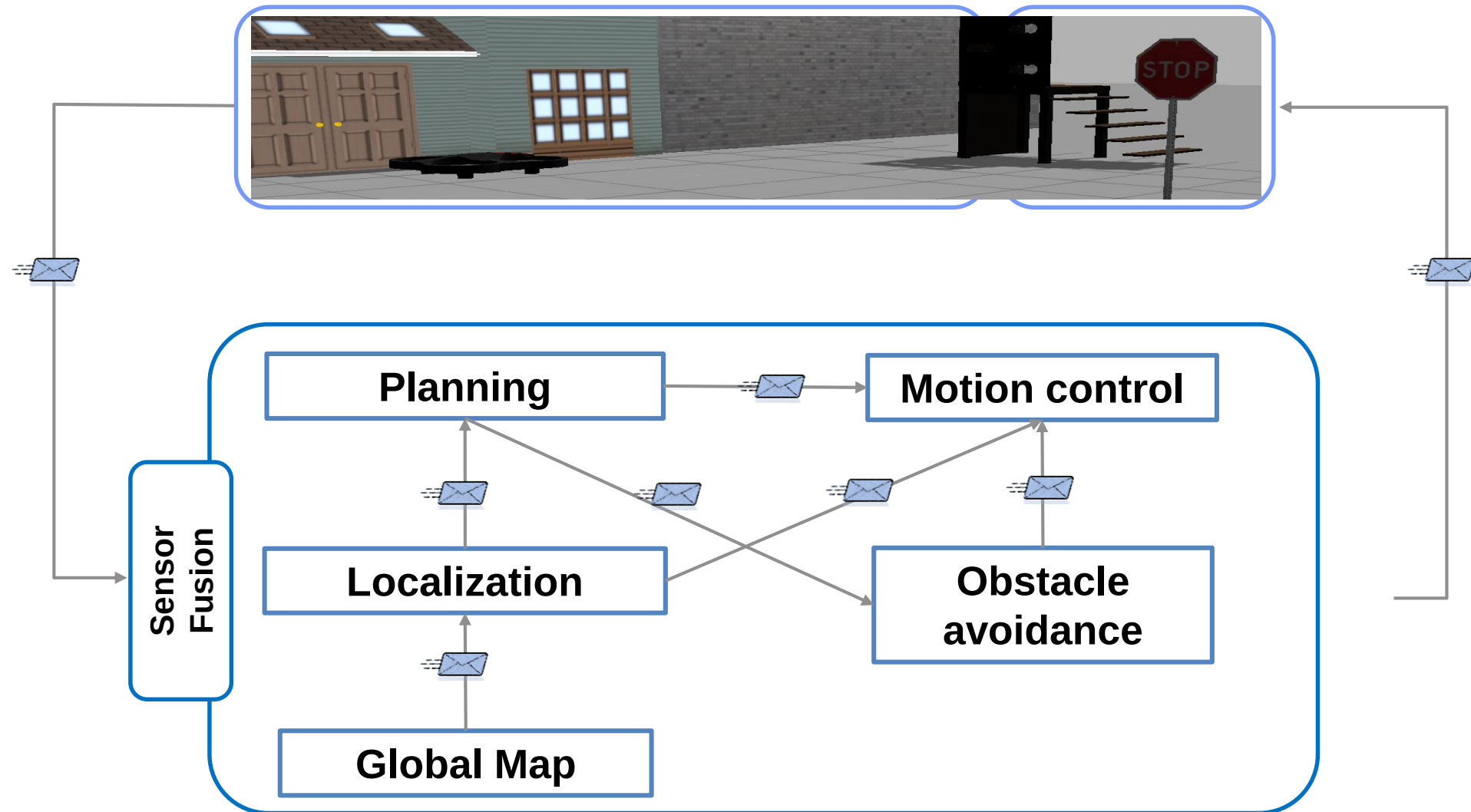
Imaging

Specialty

Standards

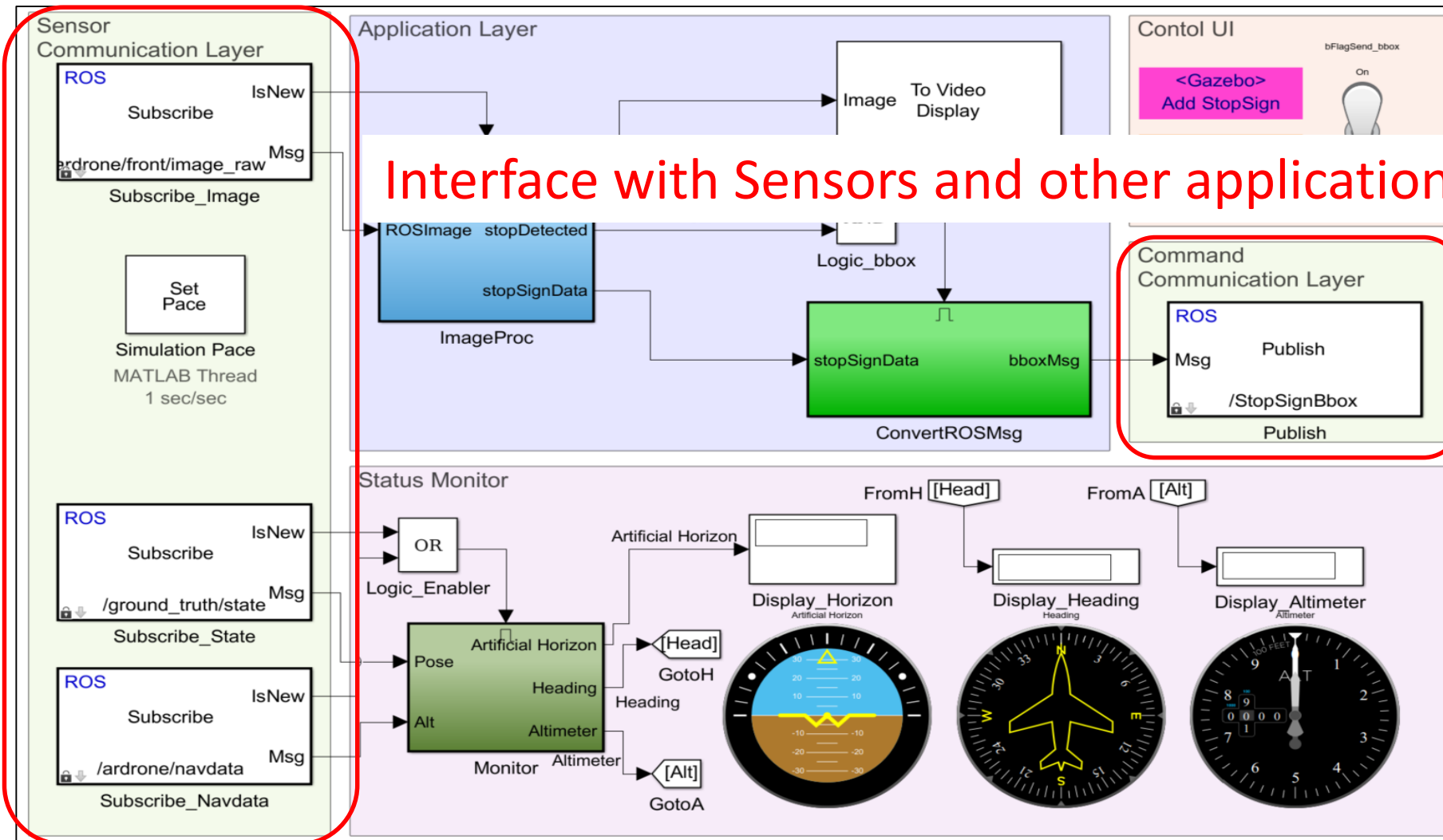
How to develop Autonomous algorithms?

Sensor data flow



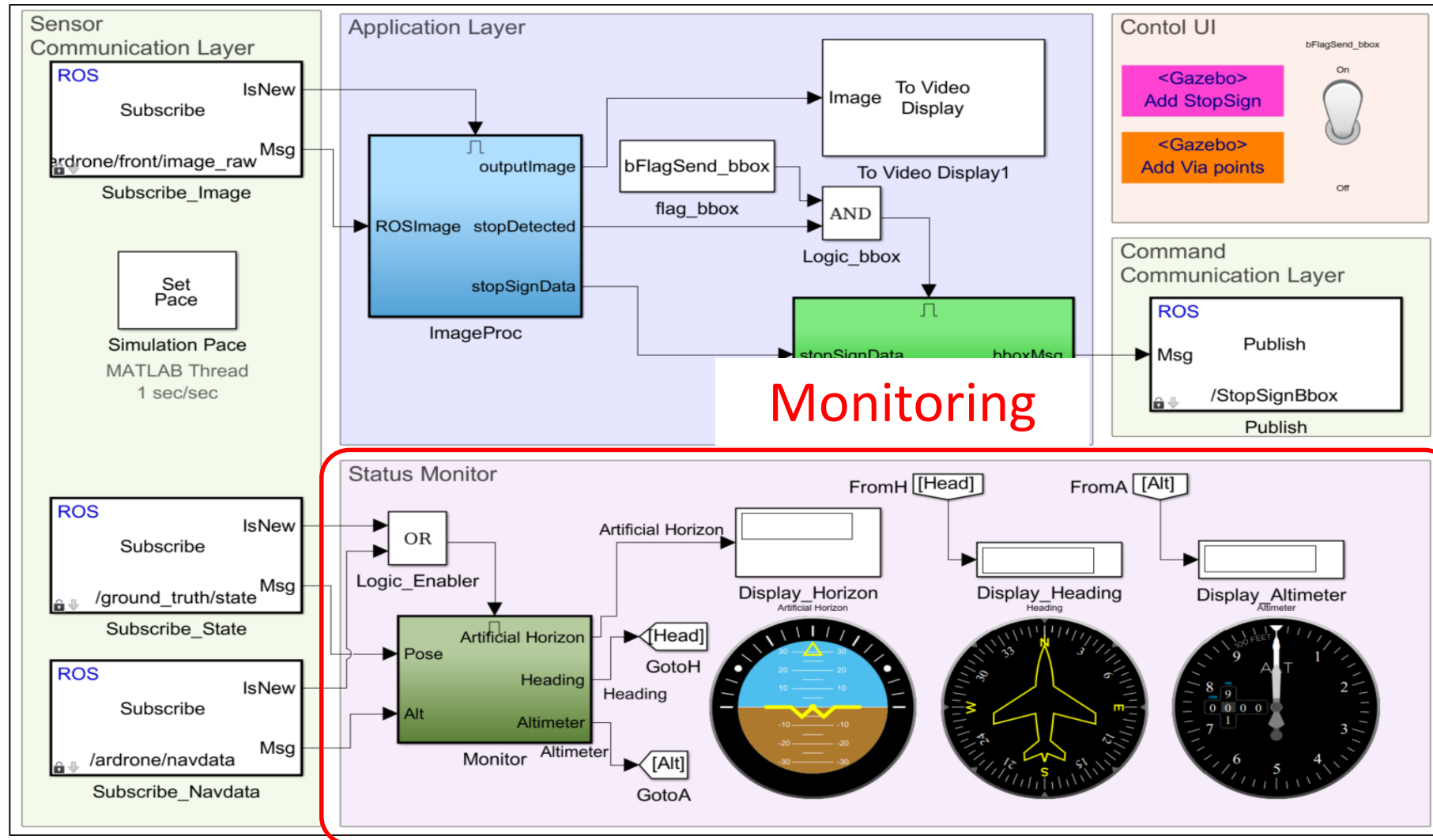
How to develop Autonomous algorithms?

System Level Design with MATLAB, Simulink



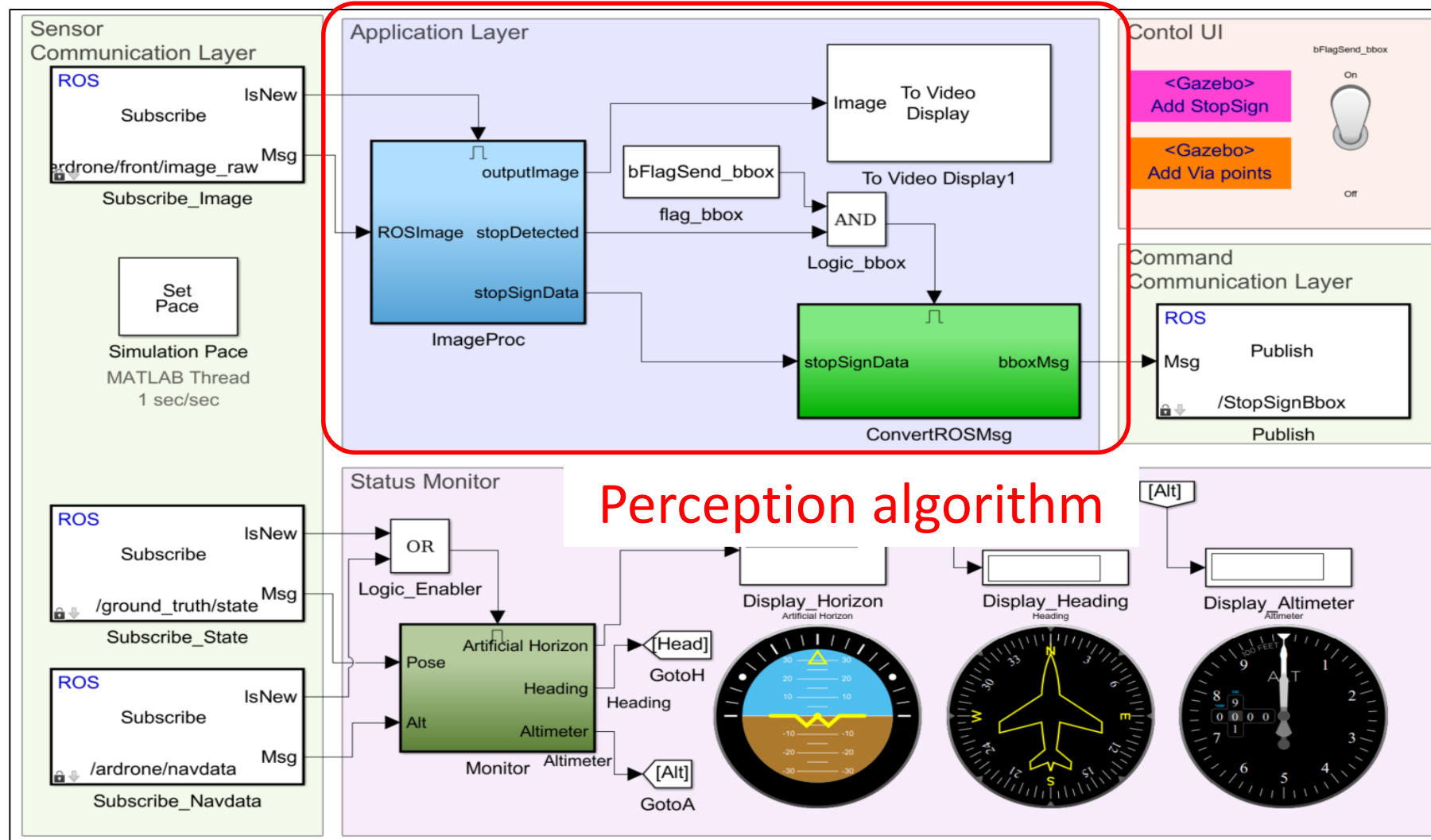
How to develop Autonomous algorithms?

System Level Design with MATLAB, Simulink



How to develop Autonomous algorithms?

System Level Design with MATLAB, Simulink

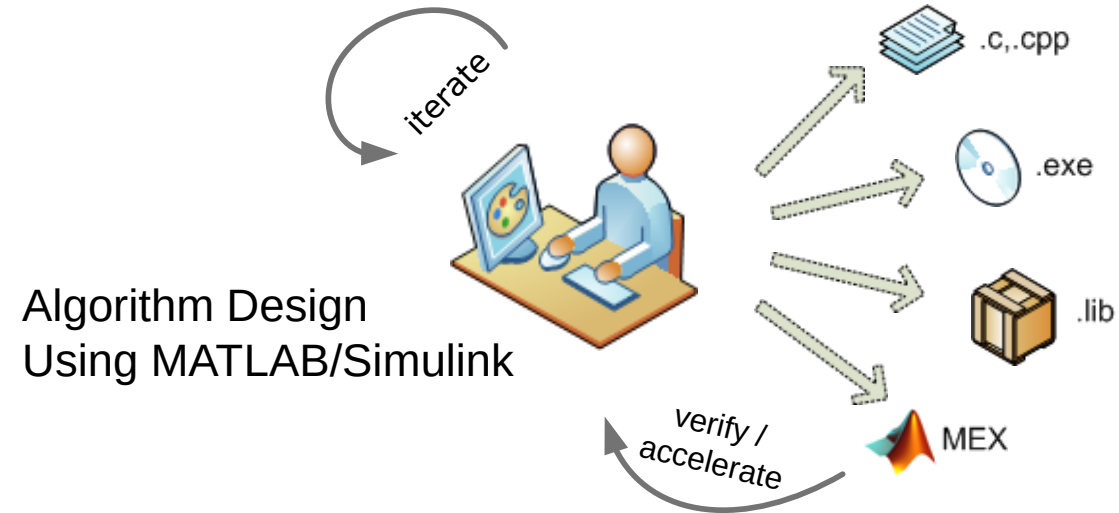


How to develop **Autonomous algorithms?**

System Level Design with MATLAB, Simulink

- Design Perception algorithm

- Matlab function
- S-function
- System Object
- State & flow control



- Calling Libraries Written in Another Language From MATLAB



- C/C++, Python, Java
- Fortran
- COM components and ActiveX® controls
- RESTful, HTTP, and WSDL web services

How to develop Autonomous algorithms?

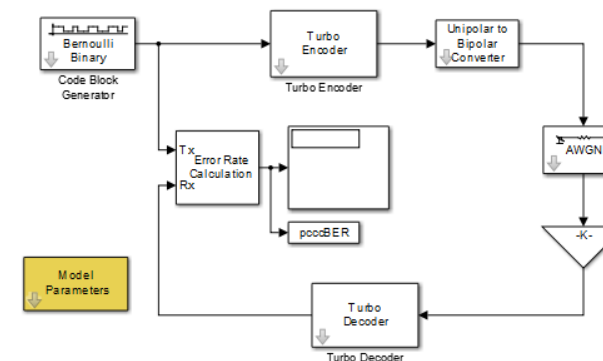
System objects

- Model dynamic systems using the MATLAB language
 - Algorithms implemented as MATLAB classes
 - API designed specifically for simulation
 - Can be used in MATLAB and Simulink

System objects



```
99 % Simulate the modified SUI-1 channel model.
100
101
102 Nsamp = 1.5e6; % Number of channel samples
103 Nsamp_f = 1e3; % Number of samples per frame
104 Nframes = Nsamp/Nsamp_f; % Number of frames
105 chanOut = zeros(Nsamp, Nr);
106 pathGains = zeros(Nsamp, length(tau), Nt);
107 for iFrames = 1:Nframes
108     inputSig = modulate(hModem, randi(S, [0 M-1], Nsamp_f, Nt));
109     idx = (1:Nsamp_f)+(iFrames-1)*Nsamp_f;
110     [chanOut(idx,:), pathGains(idx,:,:) = step(h, inputSig);
111 end
```



How to develop Autonomous algorithms?

System objects

```
RemoveMean.m  X  +
1  classdef RemoveMean < matlab.System
2      %RemoveMean Remove estimated running mean
3
4  properties
5      % Memory weighting
6      Weight = 0.999
7  end
8
9  properties (DiscreteState)
10     Mean
11 end
12
13 methods (Access=protected)
14     function y = stepImpl(obj,u)
15         y = u - obj.Mean;
16         obj.Mean = u - y*obj.Weight;
17     end
18     function setupImpl(obj)
19         obj.Mean = 0;
20     end
21     function resetImpl(obj)
22         obj.Mean = 0;
23     end
24 end
25
26 end
```

Base Class is:
matlab.System

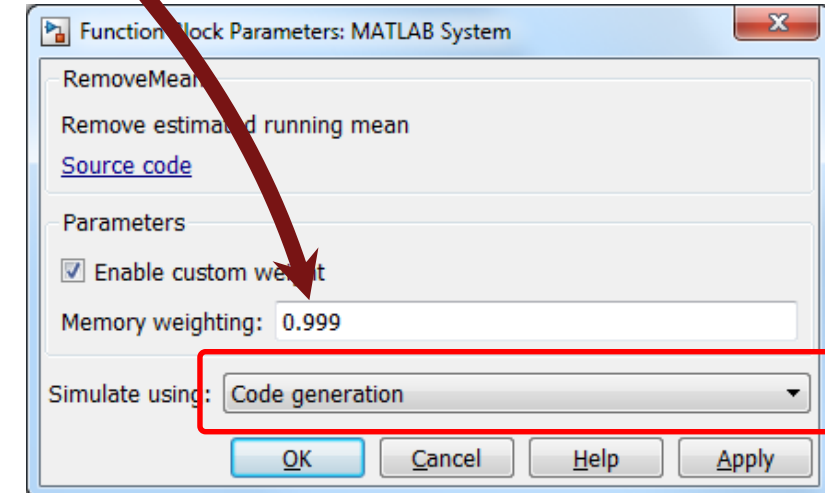
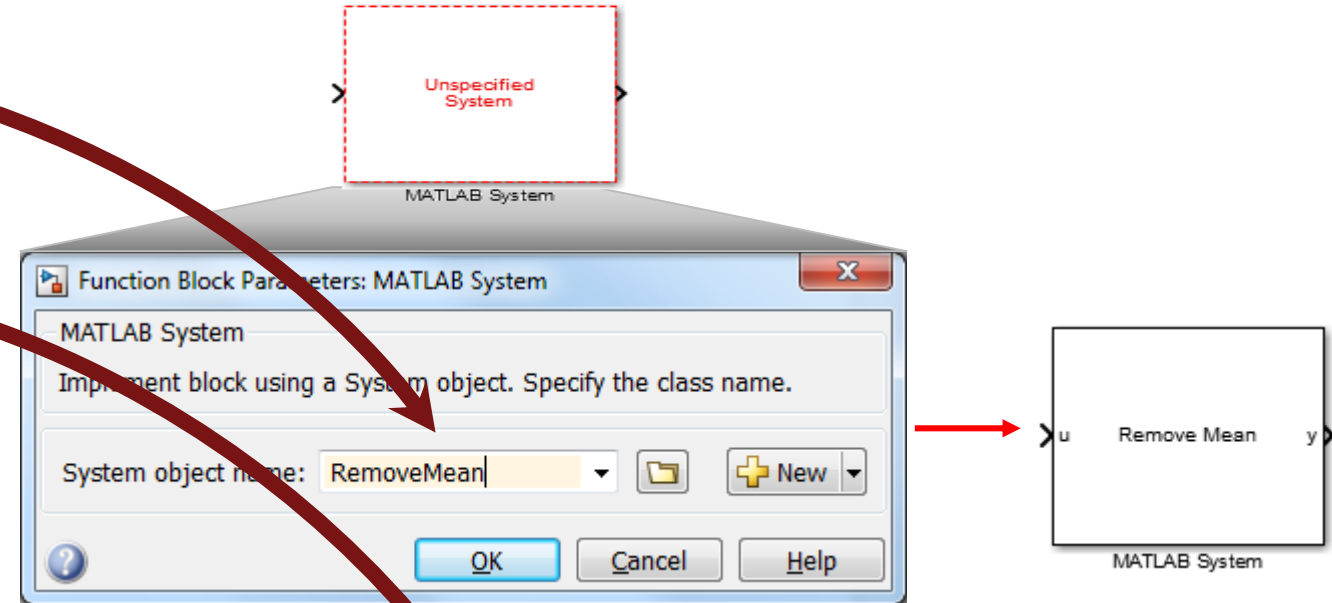
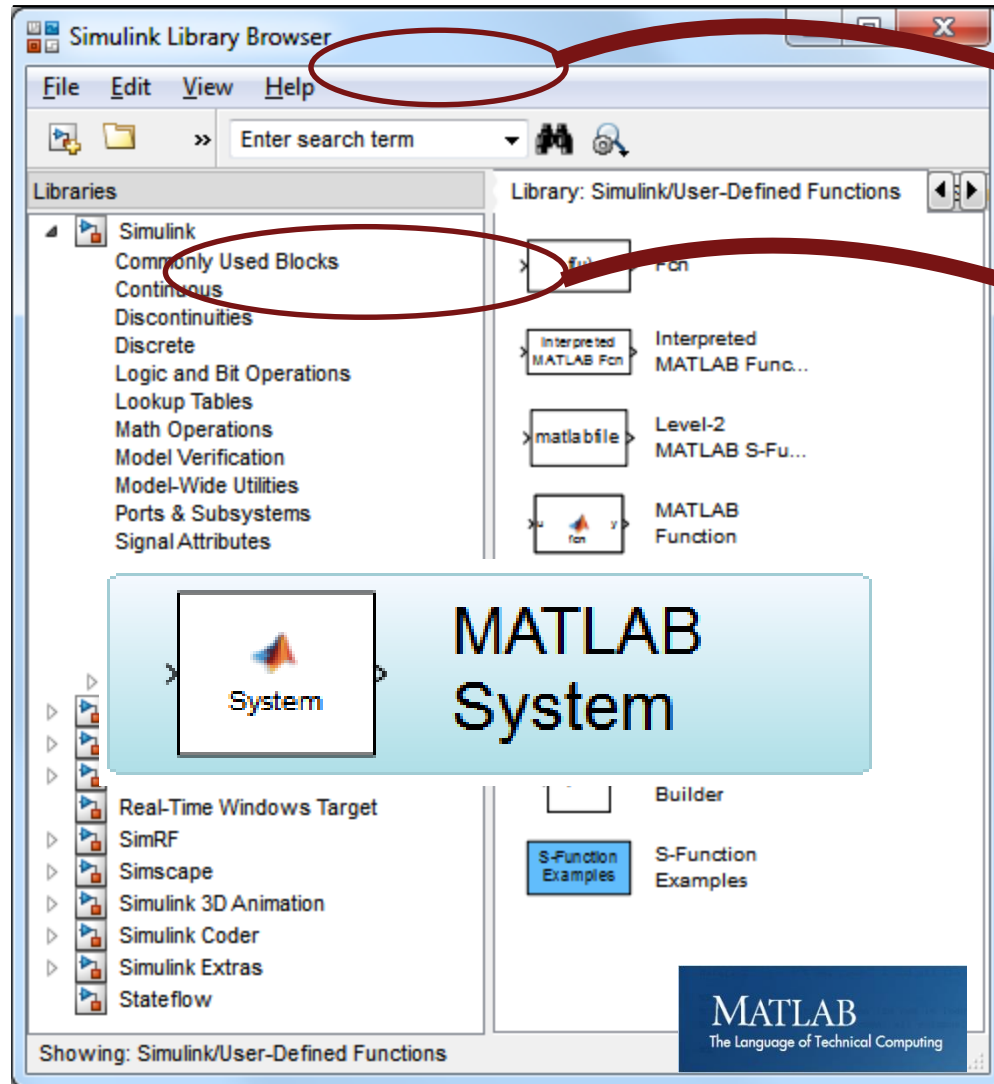
Properties provide persistent
memory within the object

Functions for each phase of your
system (validation, initialization,
reset, step, ...)

- MATLAB automatically calls
housekeeping functions,
e.g., validation, initialization, etc.

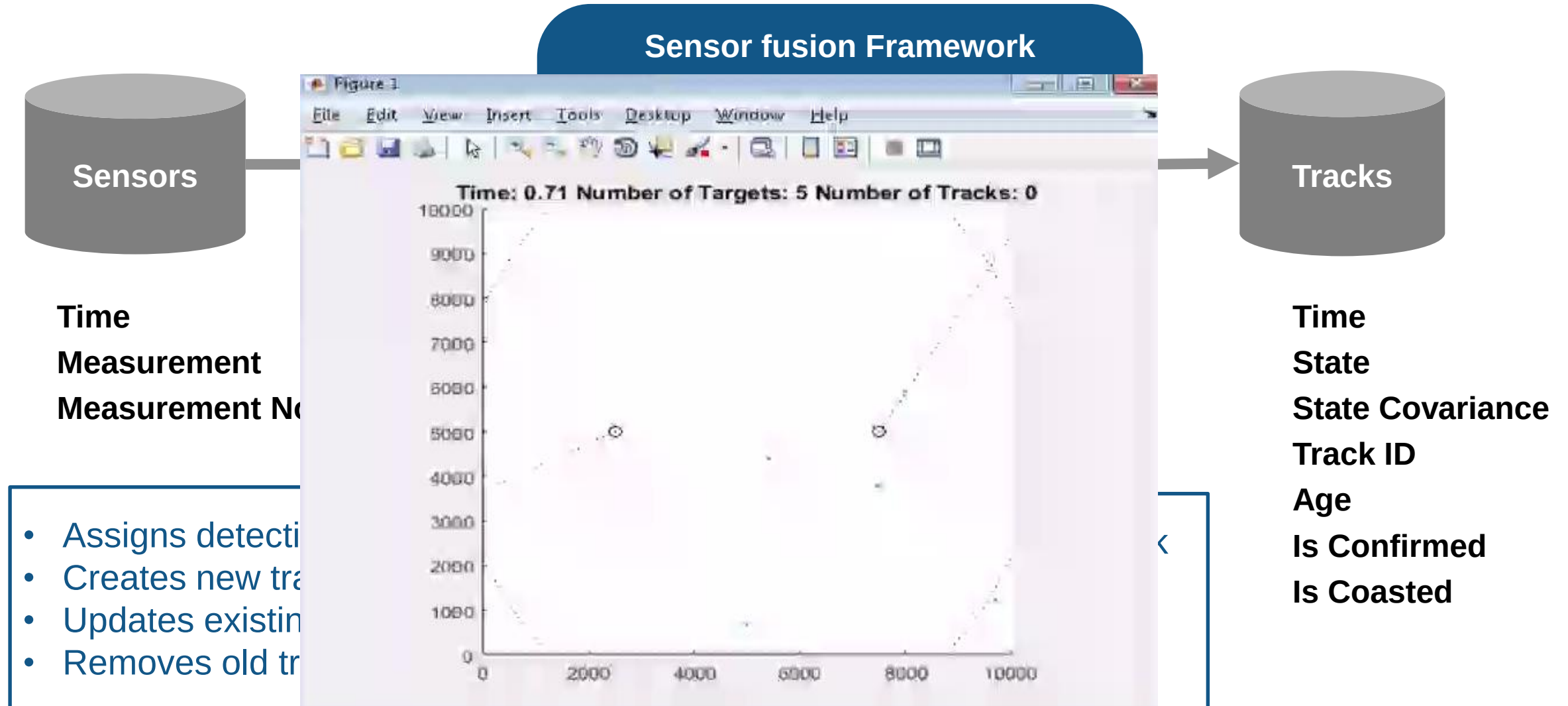
How to develop Autonomous algorithms?

System objects



How to develop Autonomous algorithms?

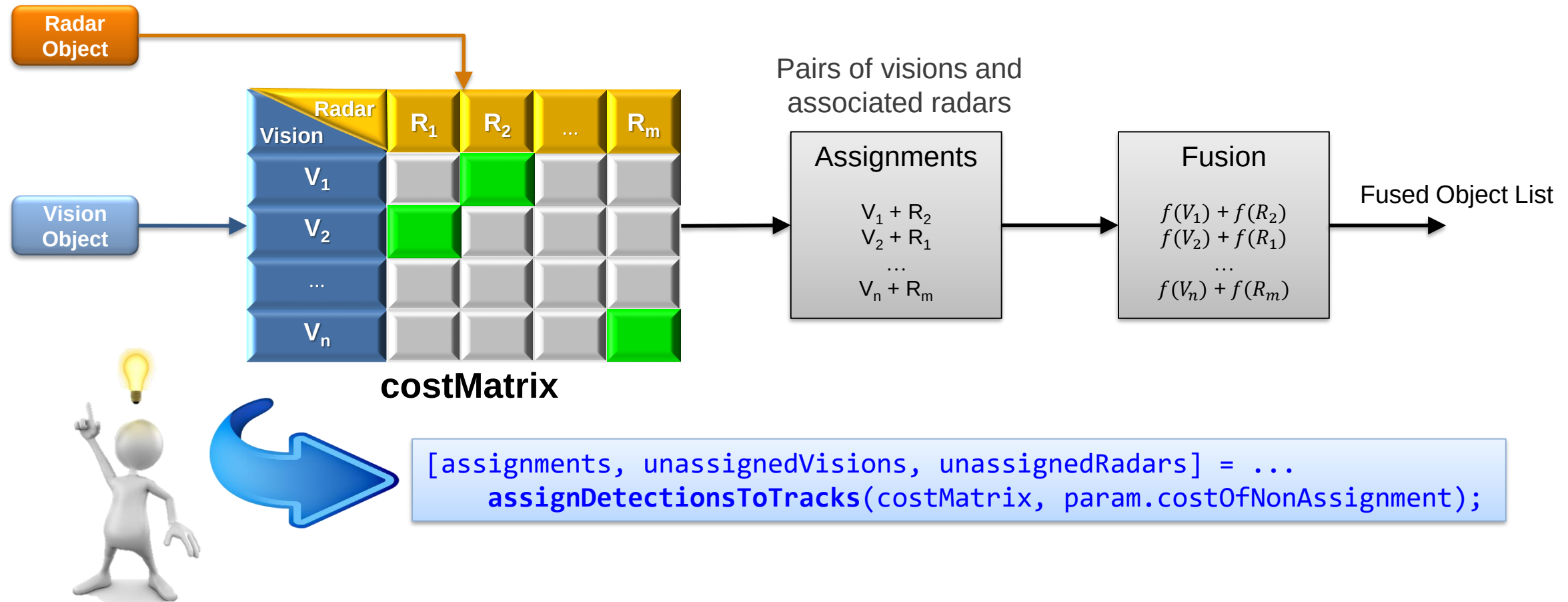
Sensor fusion framework



How to develop Autonomous algorithms?

Sensor fusion framework

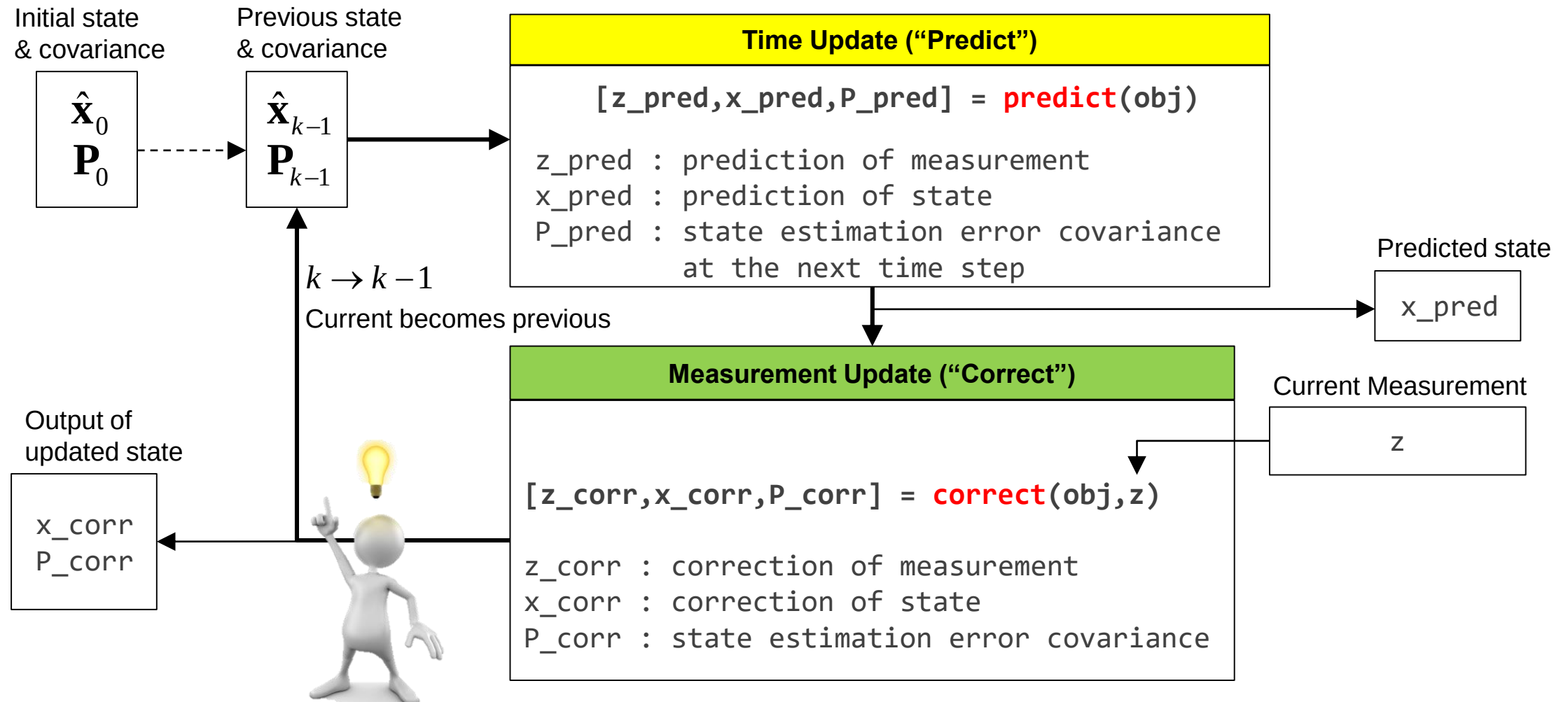
Track Manager



How to develop Autonomous algorithms?

Sensor fusion framework

Tracking Filter



How to develop Autonomous algorithms?

System Level Design with MATLAB, Simulink

C:\#00_WorkSpace\#10_Dev\#01_RampUpDemo_Drone\#02_Model\#SignDetector.m

```

classdef SignDetector < matlab.System & matlab.system.mixin.Propagates ...
    & matlab.system.mixin.CustomIcon
% Sign Detector System Object
%
% NOTE: When renaming the class name Untitled2, the file name
% and constructor name must be updated to use the class name.
%
% This template includes most, but not all, possible properties, attributes,
% and methods that you can implement for a System object in Simulink.

% Public, tunable properties
properties
end

% Public, non-tunable properties
properties(Nontunable)
    imDir = 'unclassifiedData'; % Unclassified data directory name.
    collectTrainingImages = false; % Save images for training
    recordFlag = false; % Record video
    classifierMATFile = 'trainedClassifier'; % Trained classifier MAT File Name.
end

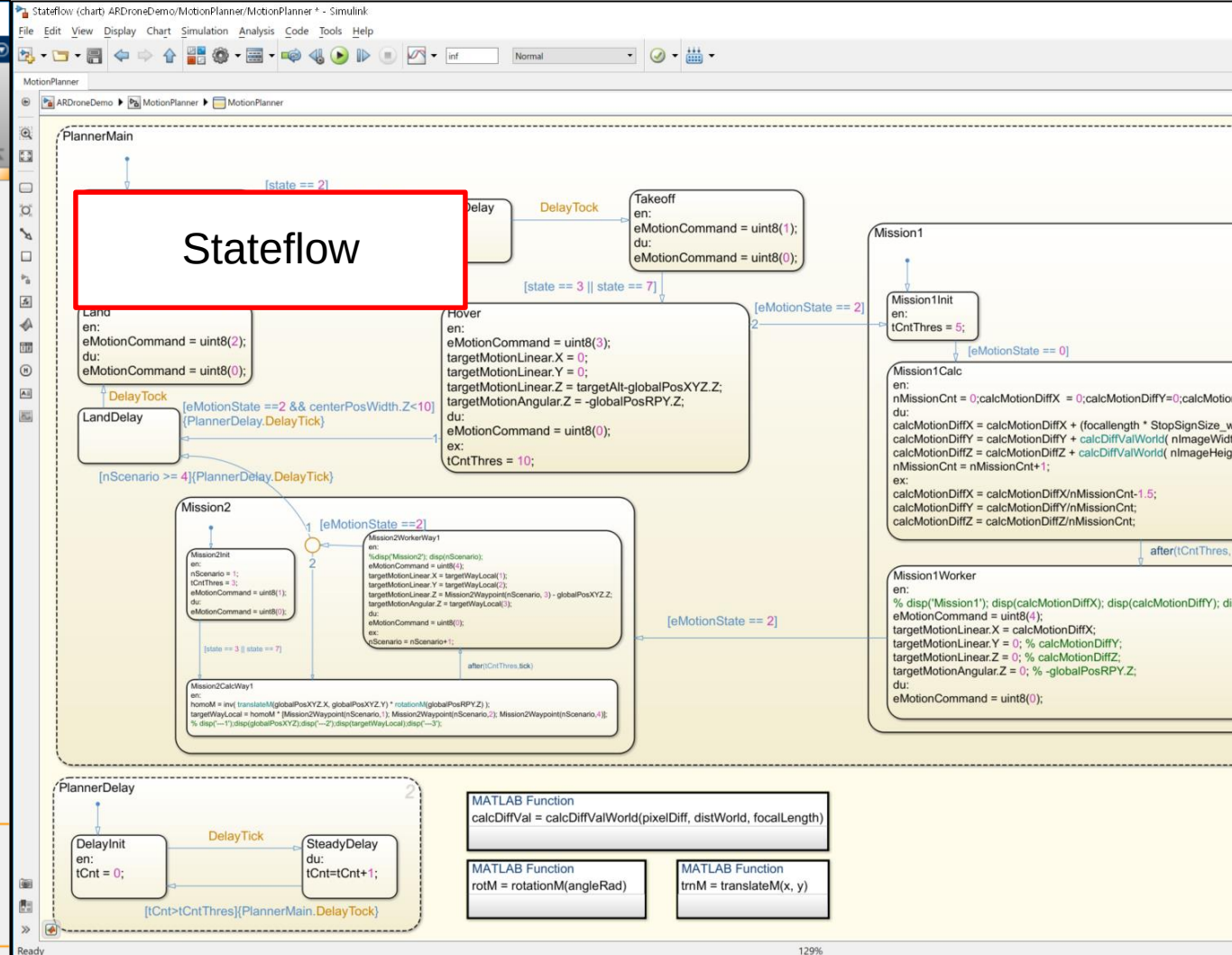
properties(DiscreteState)
end

% Pre-computed constants
properties(Access = private)
    outputBusName = 'stopSignBus_bbox';
    dateTxt = []; % string to save current date and time
    classifier = []; % trained classifier
    outputImage = []; % image with results from object detection
    BW = []; % black and white mask from image.
end

methods
% Constructor
function obj = SignDetector(varargin)

```

System Object



Design (and verify) Autonomous algorithms

The image displays a Simulink model running in a Gazebo environment, illustrating the design and verification of autonomous algorithms.

Simulink Model Structure:

- Sensor Communication Layer:**
 - ROS Subscribe: `/image_raw` (Msg)
 - ROS Subscribe: `/state` (Msg)
 - Simulation Pace: MATLAB Thread 1 sec/1000
- Application Layer:**
 - ImageProc (ROSImage) receives input from `/image_raw`.
 - ImageProc outputs to `bFlagSend_tbox` and `flag_tbox`.
 - `bFlagSend_tbox` outputs to `Image To Video Display`.
 - `flag_tbox` outputs to `Logic_tbox`.
 - `Logic_tbox` outputs to `ConvertROSMsg`.
 - `ConvertROSMsg` outputs to `Msg`.
- Control UI:**
 - `bFlagSend_tbox` (On/Off)
 - `Add StopSign` (On/Off)
 - `Add Via points` (On/Off)
- Command Communication Layer:**
 - ROS Publish: `/StopSignBox` (Msg)
- Status Monitor:**
 - ROS Subscribe: `/state` (Msg)
 - ROS Subscribe: `/state` (Msg)
 - ROS Subscribe: `/state` (Msg)
 - Artificial Horizon (Display: Horizon)
 - Heading (Display: Heading)
 - Altitude (Display: Altitude)
 - Monitor (Display: Monitor)
 - Artificial Horizon (Display: Artificial Horizon)
 - Heading (Display: Heading)
 - Altitude (Display: Altitude)
 - Monitor (Display: Monitor)

Gazebo Environment:

- World: Insert Layers
- GUI: Scene, Spherical Coordinates, Physics, Models
- Models:
 - link
 - House 3
 - House 2
 - jersey_barrier
 - grey_wall
 - rust_stairs_120
 - rust_elevated_floor_120
 - rust_maze_wall_240
 - rust
 - quadrotor
 - stop_sign
 - Lights
- Property Value:

Property	Value
name	ground_plane
is_static	True
self_collide	False
pose	
link	ground_plane:link

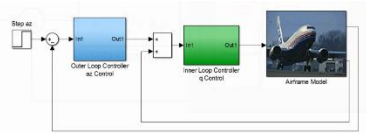
Status Bar:

- Running
- View diagnostics 96% T=1.500
- auto(FixedStepDiscrete)
- Kubuntu Gazebo 7 - VMware Workstation 12 Player (Non-commercial use only)
- Real Time Factor: 0.63
- Sim Time: 00:00:11:44.949
- Real Time: 00:00:01:12.076
- Iterations: 22607
- FPS: 20.8298

How to develop Autonomous algorithms?

CONTROL
SENSE
PERCEIVE
PLAN
CONNECT

Control System Tbx



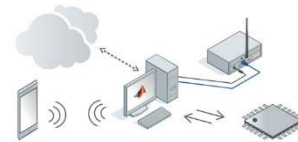
Simscape Toolboxes



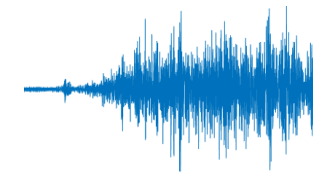
Simulink Real-Time



HW Support Packages



Data Acquisition Tbx



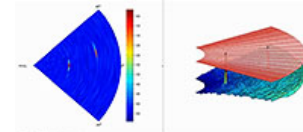
Computer Vision



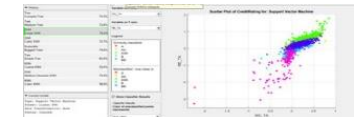
Automated Driving System Tbx



Phased Array



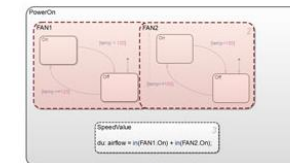
Statistics & Machine Learning



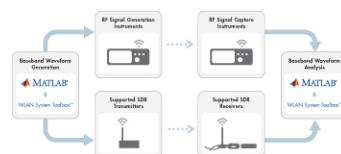
Robotics System Tbx



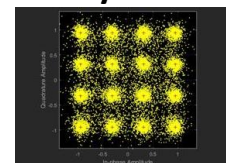
Stateflow



Communications Tbx



WLAN System Toolbox



Robotics System Tbx

ROS

Autonomous System Development Workflow

**Aerodynamics and
Flight Control**

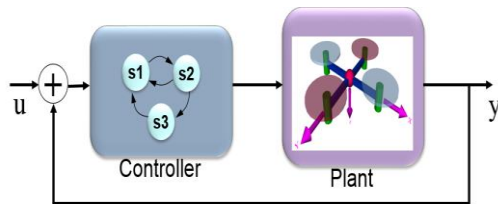
**Autonomous
algorithm**

**Test and Refine in
Simulation**

**Test and Refine on
Real Robot**

Challenge 1:

Understand
the dynamics
with control algorithm



Challenge 2:

Design and Verify
autonomous
algorithms



Challenge 3:

Verify and Implement
the algorithm
on to a real hardware



How to **Deploy** autonomous algorithm?

MATLAB Coder - Code from MATLAB

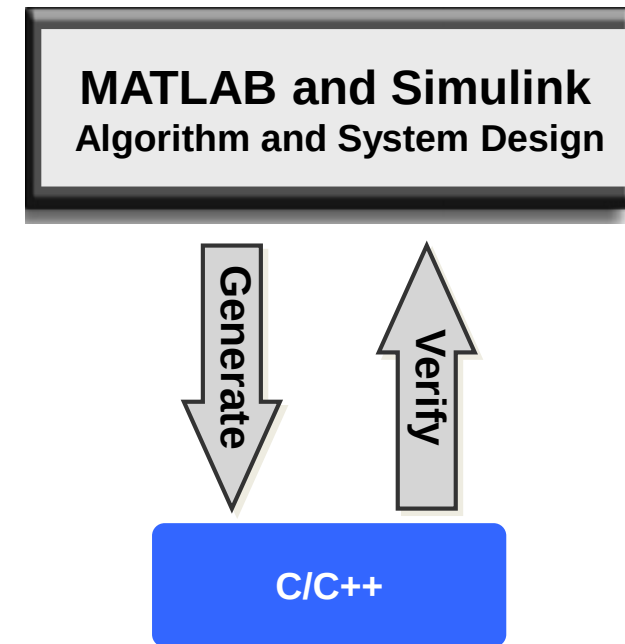
- Portable code for numerical algorithms
- Desktop applications (standalone, library)

Simulink Coder - Code from Simulink

- Rapid prototyping or HIL applications
- Real-time machines

Embedded Coder – Production optimized code

- Embedded applications
- MCU and DSP (fixed or float)
- Code verification (in-the-loop)
- Target-specific support (APIs and examples)



All coders generate portable code (ANSI/ISO C) by default.

How to **Deploy** autonomous algorithm?



Autonomous System with MATLAB/Simulink

BAE Systems Controls Develops Autopilot for Unmanned Aerial Vehicle Using MathWorks Tools

Challenge

Enable teams working in separate locations to design a sophisticated UAV autopilot system quickly and inexpensively

Solution

Use MathWorks tools, modify existing software designs with Model-Based Design, and automatically generate embedded control code

Results

- Design and rework costs substantially reduced
- Testing cycle time minimized
- Coding errors and manual documentation work minimized



An Eagle 150 unmanned aerial vehicle flight.

(Image courtesy of Composites Technology Research Malaysia.)

“MATLAB and Simulink greatly reduced development cycle time and cut system software design and testing costs by 50%.”

**Feng Liang
BAE Systems Controls**

Autonomous System with MATLAB/Simulink

Airnamics Develops Unmanned Aerial System for Close-Range Filming with Model-Based Design

Challenge

Design and develop an unmanned aerial camera motion system for close-range aerial filming

Solution

Use Model-Based Design with MATLAB and Simulink to accelerate the design, debugging, and implementation of the vehicle's fly-by-wire and flight management system software

Results

- Time-to-market shortened by up to an order of magnitude
- Test flight anomalies quickly resolved
- Debugging time reduced from weeks to hours



Airnamics co-founders Marko Thaler and Zoran Bjelić with the R5 MSN1 prototype after its first flight.

“With Model-Based Design our three-engineer team found more than 95% of control software bugs before the first flight. We used the test flights to increase our Simulink models’ fidelity and isolate remaining bugs with high precision. The result is a safer, more reliable, and higher-quality product.”

Marko Thaler
Airnamics

Summary of Aerial Autonomous System

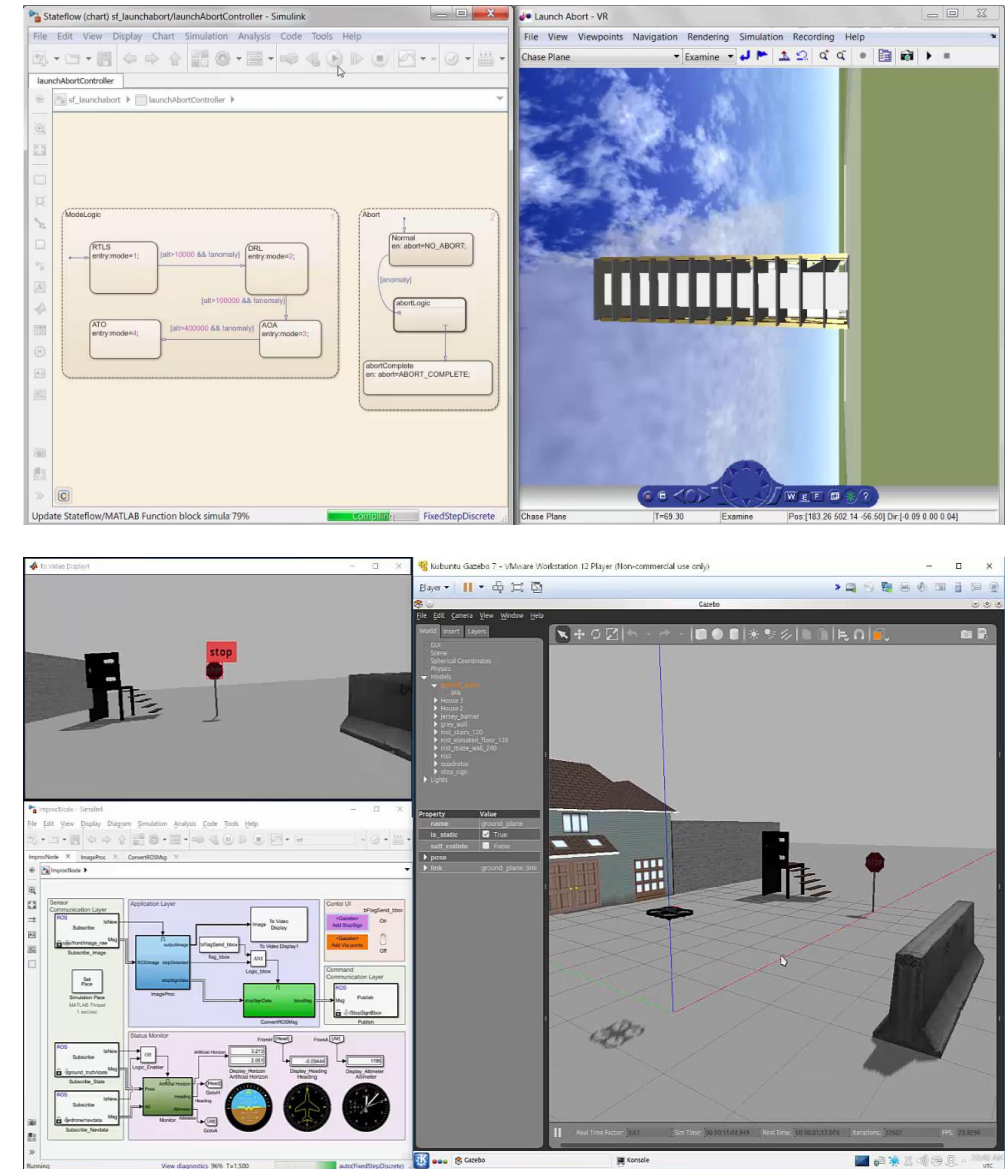
Aircraft	“Autonomous Algorithms”				Communication
Platform	Perception		Planning		Connect/ Communicate
	Sense	Perceive	Plan & Decide	Control	
• Control Surfaces, slats, flaps • Lifting Body • Landing Gear • Battery • Power Management	• Radar • Camera • Lidar • EO/IR • IMU • GPS-INS • HW Certification	• Environment mapping • Classification • Segmentation • Object Detection • Sensor Fusion	• Object Avoidance • Path & motion planning • SLAM	• Guidance, Navigation & Control • Flight SW certification	• Communication with ground operator • Multi-agent communication • Satellite data link



Key Takeaway

Designing Autonomous system using MATLAB and Simulink can help in :

- **Understanding the dynamics with control algorithm**
 - Model aerodynamics, propulsion and motion
 - Design control algorithm in single environment
- **Design vision, radar, perception algorithms**
 - Visualizing different sensor data
 - Develop and test sensor fusion and tracking algorithm
- **Implementing the algorithm on actual hardware**
 - Test and verify algorithm on 3D simulators
 - Automatic C/C++ code generation on to actual hardware



% Thank you

MATLAB

is the **easiest** and
most **productive** environment
for **engineers** and **scientists**