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

<table>
<thead>
<tr>
<th>Aircraft/Platform</th>
<th>Sense</th>
<th>Perceive</th>
<th>Plan &amp; Decide</th>
<th>Control</th>
<th>Connect/Communicate</th>
</tr>
</thead>
</table>
| • 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**

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

**Different Approaches for Modeling**

- **Autonomous Algorithms**
Autonomous System Development Workflow

**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

- **Three-axis Accelerometer**: $A_b$ (m/s²), $\phi$ (rad/s), $\dot{\phi}$ (rad/s²), CG (m), $g$ (m/s²)
- **Three-axis Gyroscope**: $\omega$ (rad/s), $\dot{\omega}$ (rad/s²), $A_{meas}$ (m/s²)
- **Three-axis Inertial Measurement Unit**: $A_b$ (m/s²), $\omega$ (rad/s), $\dot{\omega}$ (rad/s²), CG (m), $g$ (m/s²), $A_{meas}$ (m/s²)

Turbofan Engine System
- Throttle position
- Thrust (N)
- Mach
- Altitude (m)
- Fuel flow (kg/s)

Second Order Nonlinear Actuator
- $A_{demand}$
- $A_{actual}$

3D Controller
- $\begin{bmatrix} A(v), B(v), C(v), D(v) \end{bmatrix}$

3D Observer Form
- $\begin{bmatrix} u_{dem}, u_{meas}, v1, v2, v3, \phi, \dot{\phi}, \omega, \dot{\omega}, \gamma \end{bmatrix}$
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

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 develop **Autonomous algorithms**?

- Autoroute, gated, post-process automatically
- Efficient, speed, endurance, automation

Control and Perception Algorithms

- Planning
- Motion Control
- Localization
- Obstacle Avoidance
- Mapping

Sensor Fusion

- Camera, EOIR, LIDAR, GPS/IMU, …

Autonomous algorithm
How to develop *Autonomous algorithms*?

*Manage Sensor data*
How to develop **Autonomous algorithms**?

*Manage Sensor data*
How to develop **Autonomous algorithms**?

*Sensor data flow*

- **Planning**
- **Motion control**
- **Localization**
- **Obstacle avoidance**
- **Global Map**

EOIR: Electro Optical Infra-Red
How to develop **Autonomous algorithms?**

*System Level Design with MATLAB, Simulink*

Interface with Sensors and other application
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 system design and simulation
  - Can be used in MATLAB and Simulink
How to develop **Autonomous algorithms**?

*System objects*

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*

**Sensor fusion Framework**

- Assigns detections to tracks
- Creates new tracks
- Updates existing tracks
- Removes old tracks

**Sensors**

- Time
- Measurement
- Measurement Noise

**Tracks**

- Time
- State
- State Covariance
- Track ID
- Age
- Is Confirmed
- Is Coasted
How to develop **Autonomous algorithms**?

*Sensor fusion framework*

- **Track Manager**

```matlab
[assignments, unassignedVisions, unassignedRadars] = ...
assignDetectionsToTracks(costMatrix, param.costOfNonAssignment);
```
How to develop **Autonomous algorithms**?

**Sensor fusion framework**

- **Tracking Filter**

  - **Initial state & covariance**
    - \( \hat{x}_0 \)
    - \( P_0 \)
  
  - **Previous state & covariance**
    - \( \hat{x}_{k-1} \)
    - \( P_{k-1} \)

  - **Time Update (“Predict”)**
    
    \[
    [z_{\text{pred}}, x_{\text{pred}}, P_{\text{pred}}] = \text{predict}(\text{obj})
    \]
    
    - \( z_{\text{pred}} \): prediction of measurement
    - \( x_{\text{pred}} \): prediction of state
    - \( P_{\text{pred}} \): state estimation error covariance at the next time step

  - **Measurement Update (“Correct”)**
    
    \[
    [z_{\text{corr}}, x_{\text{corr}}, P_{\text{corr}}] = \text{correct}(\text{obj}, z)
    \]
    
    - \( z_{\text{corr}} \): correction of measurement
    - \( x_{\text{corr}} \): correction of state
    - \( P_{\text{corr}} \): state estimation error covariance

  - **Predicted state**
    - \( x_{\text{pred}} \)
  
  - **Current Measurement**
    - \( z \)
How to develop **Autonomous algorithms**?

**System Level Design with MATLAB, Simulink**

- **System Object**
- **Stateflow**
Design (and verify) **Autonomous algorithms**
How to develop **Autonomous algorithms**?
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

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

Feng Liang
BAE Systems Controls

An Eagle 150 unmanned aerial vehicle flight.
(Image courtesy of Composites Technology Research Malaysia.)

Link to user story
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

“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

Airnamics co-founders Marko Thaler and Zoran Bjelić with the R5 MSN1 prototype after its first flight.
### Summary of Aerial Autonomous System

#### Aircraft
- Control Surfaces, slats, flaps
- Lifting Body
- Landing Gear
- Battery
- Power Management

#### Perception
- Radar
- Camera
- Lidar
- EO/IR
- IMU
- GPS-INS
- HW Certification
- Environment mapping
- Classification
- Segmentation
- Object Detection
- Sensor Fusion

#### Planning
- Object Avoidance
- Path & motion planning
- SLAM

#### Control
- Guidance, Navigation & Control
- Flight SW certification

#### Communication
- Communication with ground operator
- Multi-agent communication
- Satellite data link

---

**Different Approaches for Modeling**

**"Autonomous Algorithms"**
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.