MATLAB 및 Simulink을 이용한 운전자 지원 시스템 개발

김종헌 차장
Senior Application Engineer
MathWorks Korea
Example:
Sensor Fusion with Monocular Vision & Radar

- **Configuration**
  - Monocular Vision installed on Wind Shield
  - Radar installed in Front Grill

- **High Level Sensor Fusion**

- **2 Different Scenario**
  - Closest In-Path Vehicle Selection
  - Stationary Vehicle Detection
Sensor Fusion

Why Sensor Fusion?

- To complement weakness of individual sensor

<table>
<thead>
<tr>
<th>Radar</th>
<th>Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate</td>
<td>Poor</td>
</tr>
<tr>
<td>Measureable Directly</td>
<td>Range Rate</td>
</tr>
<tr>
<td>Poor</td>
<td>Estimated</td>
</tr>
<tr>
<td>Not Available</td>
<td>Relatively Good</td>
</tr>
<tr>
<td>Classification</td>
<td>Available</td>
</tr>
</tbody>
</table>

- More Coverage
Issues?

- Communication

- Risk of Vehicle Test
How do we bridge the gap between engineers who speak different languages and reduce risk?

- Communication

- Risk of Vehicle Test
Solution:
Model based Multi-domain Modeling and System Level Simulation

- Model serves as the executable specification and test environment
- Multiple engineering teams can collaborate more effectively
**Example:**
Sensor Fusion with Monocular Vision & Radar

- **Vision**
  - Lane Detection
  - Vehicle Detection
  - Extended Kalman Filter based Tracking

- **Radar**
  - FMCW Radar Modeling
  - Extended Kalman Filter based Tracking

- **Sensor Fusion**
  - Data Association
  - Data Fusion
Abstract

- **Vision**
  - Lane Detection
  - Vehicle Detection
  - Etc.

- **Radar**
  - FMCW Radar Modeling

- **Tracking**
  - Extend Kalman Filter based Tracking

- **Sensor Fusion**
  - Integration
Lane Detection

ROI (Region Of Interest)

ROI (Region Of Interest) Cropped Image

Captured Image

Edge Mapping to World

Selecting Candidate Pts

Curve Fitting
Lane Detection

ROI (Region Of Interest)

Captured Image

Image Processing Function

Cropped Image

MATLAB Built-In Function

CVST Function

Easy Matrix Calculation

Code Generation

Edge Mapping to World

Selecting Candidate Pts

Easy to Debug Scripting Language

Pixel level, 2D, 3D Visualization

Curve Fitting
Lane Detection
Abstract

- Vision
  - Lane Detection
  - Vehicle Detection
  - Etc.
- Radar
  - FMCW Radar Modeling
- Tracking
  - Extend Kalman Filter based Tracking
- Sensor Fusion
  - Integration
Cascade Detector

Applications in Automotive of Cascade Detector

- Vehicle detection
- Pedestrian detection
- Traffic Sign Recognition

Difference between applications is “Feature”.
4 Easy Steps to Create a Car Detector

1. Collect Car Image Data (Training Images)
2. Label Objects of Interest
3. Train Object Detector
4. Test Detector Performance
Step 1: Collect Training Image Data

- Collect many images that represent object you want to detect
- Ensure that images represent different conditions
  - Lighting
  - Different variations of object – size, color, model etc.
Step 2: Label Training Images

- Select and assign regions of interest

Images from: www.emt.tugraz.at/~pinz/data/GRAZ_02/
Step 3: Train Object Detector

```matlab
>> trainCascadeObjectDetector(detectorFile, ... positiveInstances, negativeFolder,... 'FalseAlarmRate', 0.1,'NumCascadeStages', 5);
detector = vision.CascadeObjectDetector(detectorFile);
```
Step 4: Test Detector Performance
Comparison Data from Vehicle Dynamics S/W and Result of Cascade Detector
Further Information about Cascade Object Detector

Abstract

- Vision
  - Lane Detection
  - Vehicle Detection
  - Etc.

- Radar
  - FMCW Radar Modeling

- Tracking
  - Extend Kalman Filter based Tracking

- Sensor Fusion
  - Integration
Camera Calibration App.

MATLAB commands for camera calibration are also available.
Camera Calibration App.

- Intrinsic Matrix
  
  
  \[
  K = \begin{bmatrix}
  f_x & 0 & 0 \\
  s & f_y & 0 \\
  c_x & c_y & 1
  \end{bmatrix}
  \]

  - \((c_x, c_y)\) - camera center in pixels
  - \((f_x, f_y)\) - focal length in pixels per world unit
    
    \(f_x = \frac{f}{p_x}, f_y = \frac{f}{p_y}\)
  - \(s\) - sensor axes skew coefficient
    
    \(s = f_y \tan \alpha\)

- Tangential Distortion

  - \(x_{\text{tangential}} = x_{\text{radial}} + 2p_1xy + p_2(r^2 + 2x^2)\)
  - \(y_{\text{tangential}} = y_{\text{radial}} + p_1(r^2 + 2y^2) + 2p_2xy\)

- Radial Distortion

  - \(x_{\text{radial}} = (1 + k_1r^2 + k_2r^4 + k_3r^6)x\)
  - \(y_{\text{radial}} = (1 + k_1r^2 + k_2r^4 + k_3r^6)y\)
  - \(x\) and \(y\) are in normalized image coordinates
    
    - in world units relative to the optical center
Image Processing

- **Image Enhancement**
  - Deblurring
  - Noise Removal

- **Image Segmentation**
  - Active Contours
  - Watershed
Feature Detection, Extraction and Matching

- Edge
- Corner
- Template
- SURF
- MSER

Image Matching

Feature Point Tracking

Video Stabilization

(+23.0, +10.0)
Stereo Vision (2014a)

- Stereo calibration
- Semi-global disparity matching
  - Better results than block matching
- 3D scene reconstruction from disparity
- New demos available
  - Estimate real distance to objects in video
  - 3D scene reconstruction
  - Pedestrian detection (2014b)
Stixel based Object Detection

- Available to detect objects which are not easy using other sensors
  - Stationary object, overlapped object, kid….

* Stereo and City, Franke Uwe, Daimler Research Center
Stereo Vision Demo

- Computer Vision System Toolbox
  - Stereo camera calibration
  - SGM based Stereo matching
  - 3D reconstruction using camera calibration data
  - Kanade-Lucas-Tomasi point tracking
  - 2D/3D plotting
Point Cloud Processing (2015a)

- pointCloud object
  - Fast search using kdtree
  - Tree is constructed implicitly when a search is called first time

- Downsampling
  - Two method : ‘gridAverage’, ’random’

- Visualization

- Rigid registration
  - ‘Iterative Closest Point’ Algorithm
OCR – Optical Character Recognition (2014a)

- Highly requested in Frontlines
- Support for English, Japanese
- Users can download additional language support
- Shipping demo with text detection and OCR workflow
Abstract

- Vision
  - Lane Detection
  - Vehicle Detection
  - Etc.
- Radar
  - FMCW Radar Modeling
- Tracking
  - Extend Kalman Filter based Tracking
- Sensor Fusion
  - Integration
Modeling FMCW Radar for Automotive

- The received signal is a delayed copy of the transmitted signal

- Ultra-large signal bandwidth (>100MHz)
- Ultra high frequencies (>77GHz)
FMCW Radar: How Does it Work?
What Behavior Can Be Modeled?

Channel modeling (interference, noise)

Modeling of antenna arrays (no of elements, position)

Modeling of RF Impairments (noise, non-linearity, frequency dependency)

Determination of waveforms (range, resolution)

Algorithms for data analysis
Which MathWorks Tools Can Help?

Communications System Toolbox

Phased Array System Toolbox

PA

LNA

SimRF

Phased Array System Toolbox
Signal Processing Toolbox
Instrument Control Toolbox

Phased Array System Toolbox
DSP System Toolbox

DSP
Phased Array System Toolbox

- Algorithms
- Simulation Tools

Waveforms
- Pulse, LFM, FMCW, etc.

Transmitter
- Monostatic and Bistatic
- Transmit Arrays
  - ULA, URA, etc.

Receiver
- Monostatic and Bistatic
- Receive Arrays
  - ULA, URA, etc.

Beamforming,
- Matched Filtering, Detection, CFAR, STAP, etc.

Environment effects,
- Impairments, interference

- Algorithms
- Simulation Tools
Phased Array System Toolbox

- Algorithms
- Simulation Tools
- Design & Analyze Tools

Beamforming, Matched Filtering, Detection, CFAR, STAP, etc.

Transmitter

Rx Antenna Arrays

Environment, Targets, and Interference

 Algorithms
 Simulation Tools
 Design & Analyze Tools

Monostatic and Bistatic ULAs, etc.

Receiver

Varied Signal Processing

Phased Array System Toolbox

MathWorks
Radar Simulation
Abstract

- Vision
  - Lane Detection
  - Vehicle Detection
  - Etc.

- Radar
  - FMCW Radar Modeling

- Tracking
  - Extend Kalman Filter based Tracking

- Sensor Fusion
  - Integration
Moving From Detections to Tracking

- Detect moving objects
- **Estimate** motion of each object
- Assign detections to tracks using **data association**
- **Innovate** track data from detected objects
# Kalman Filter

From Wikipedia,

<table>
<thead>
<tr>
<th></th>
<th>Kalman Filter (Linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predict</strong></td>
<td></td>
</tr>
<tr>
<td>a priori state estimate</td>
<td>$\hat{x}_{k</td>
</tr>
<tr>
<td>a priori estimate covariance</td>
<td>$P_{k</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td></td>
</tr>
<tr>
<td>Innovation residual</td>
<td>$\tilde{y}<em>k = z_k - H_k \hat{x}</em>{k</td>
</tr>
<tr>
<td>Innovation covariance</td>
<td>$S_k = H_k P_{k</td>
</tr>
<tr>
<td>Kalman gain</td>
<td>$K_k = P_{k</td>
</tr>
<tr>
<td>a posteriori state estimate</td>
<td>$\hat{x}_{k</td>
</tr>
<tr>
<td>a posteriori estimate covariance</td>
<td>$P_{k</td>
</tr>
</tbody>
</table>
Kalman Filter

- Function in MATLAB (Computer Vision System Toolbox)
  - Define:
    - \texttt{kalmanFilter} = \texttt{configureKalmanFilter(MotionModel, InitialLocation, InitialEstimateError, MotionNoise, MeasurementNoise)}
  - Prediction:
    - \texttt{pred\_location} = \texttt{kalmanFilter.predict()}
  - Correction:
    - \texttt{corrected\_location} = \texttt{kalmanFilter.correct( observed\_location)}
Kalman Filter

- Blocks in MATLAB (DSP System Toolbox)
Extended Kalman Filter

**Prediction**

\[ \hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1}) \]

\[ P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + Q_{k-1} \]

**Innovation**

\[ \tilde{y}_k = z_k - h(\hat{x}_{k|k-1}) \]

\[ S_k = H_k P_{k|k-1} H_k^T + R_k \]

\[ K_k = P_{k|k-1} H_k^T S_k^{-1} \]

\[ P_{k|k} = (I - K_k H_k) P_{k|k-1} \]

\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \]

**Jacobian Matrix !!!**
How to Make Jacobian Matrix

**Jacobian matrix and determinant**

From Wikipedia, the free encyclopedia

In vector calculus, the Jacobian matrix (/dʒəˈkoubiən/, /jəˈkuːbiən/) is the matrix of all first-order partial derivatives of a vector-valued function. Specifically, suppose \( f : \mathbb{R}^n \to \mathbb{R}^m \) is a function which takes as input the vector \( x \in \mathbb{R}^n \) and produces as output the vector \( f(x) \in \mathbb{R}^m \). Then the Jacobian matrix \( J \) of \( f \) is an \( m \times n \) matrix, usually defined and arranged as follows:

\[
J = \frac{df}{dx} = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n}
\end{bmatrix}
\]

or, component-wise:

\[
J_{i,j} = \frac{\partial f_i}{\partial x_j}.
\]
Solution

- **MuPAD in Symbolic Math Toolbox**

1. Define $X$ vector
2. Define $f$ vector
3. Calculate $\frac{\partial f}{\partial x}$
Solution

- **MuPAD in Symbolic Math Toolbox**
Data Association for Tracking Multiple Objects

- James Munkres's variant of the Hungarian assignment algorithm
  - \([\text{assignments}, \text{unassignedTracks}, \text{unassignedDetections}] = \text{assignDetectionsToTracks}(\text{costMatrix}, \text{costOfNonAssignment})\)
  - Example of costMatrix
    - costMatrix = zeros(nTracks, nDetect);
    - for i = 1:nTracks
      - costMatrix(i, :) = distance(predictedTracks(i), Detections);
    - end
Abstract

- Vision
  - Lane Detection
  - Vehicle Detection
  - Etc.

- Radar
  - FMCW Radar Modeling

- Tracking
  - Extend Kalman Filter based Tracking

- Sensor Fusion
  - Integration
Integration in Simulink

Input: Image & Target Info.

Camera Module
- Lane Detection
- Vehicle Detection
- Tracking

Radar Module
- Radar Modeling
- Detection
- Tracking

Fusion Module
- Data Association
- Data Fusion, TTC
- In-Path Vehicle

Visualization
What’s the problem which this model has in terms of “Integration”? 

- Use a single file for multiple engineers
- Chaotic
- Keeping track of file dependencies

→ How Can I Reduce Inefficiency when Integrating Models?
Partitioning a Model using Model Blocks

- Model Reference used to partition top-level model into components
- Signal Buses & Bus Objects used to define the interfaces between components
- One file-per-component fits well with Revision Control packages
- Enabler for parallel development of components
Variant Subsystems
What’s the problem which this model has in terms of “Integration”?

- Use a single file for multiple engineers
- Chaotic
- Keeping track of file dependencies

How Can I Reduce Inefficiency when Integrating Models?
Simulink Projects

Manage project-related files efficiently within Simulink

- Search, manage, and share project-related files
- Conduct peer review of changes using Simulink XML comparison tool
- Access source control functionality for team collaboration
Simulation Result

* : Vision Target
○ : Radar Target
x : Fusion Target

Yaw Rate Based
Lane Based
Simulation Result

Stationary Vehicle

No Radar Target
Simulation Result
Conclusion

- MATLAB/Simulink and etc. provide an environment to develop algorithm, modeling environment and integrate algorithms and various types of sensor models.
  - For Vision or Lidar Sensor:
    - Image Acquisition Toolbox
    - Image Processing Toolbox
    - Computer Vision System Toolbox
  - For Radar Sensor
    - Phased Array System Toolbox
    - SimRF
    - Signal Processing Toolbox

- Simulink is a good platform for team collaboration
  - Simulink Project
More Information

- **Computer Vision**
  - Computer Vision Workflow in MATLAB/Simulink
    - [http://www.matlabexpo.com/kr/2015/sessions/t1-s2.html](http://www.matlabexpo.com/kr/2015/sessions/t1-s2.html)
  - Object Detection and Tracking
  - Face Recognition with MATLAB

- **Radar Design**
  - Modeling and Simulating Radar Systems Using MATLAB and Simulink
  - Design and Verify RF Transceivers for Radar Systems

- **Users Stories**
  - Traffic Recognition System from Continental
  - Radar Sensor Alignment Algorithm from Delphi