개발과정에서의 MATLAB과 C의 연동
(영상처리분야)

Application Engineer
Caleb Kim
Algorithm Development with MATLAB for C/C++ Programmers

- **Objectives**
  - Use MATLAB throughout algorithm development process
  - Highlight MATLAB external interfaces
  - Move MATLAB algorithm into product

- **Design, implement and verify a lane detection system**
Algorithm Development Process

- **Research & Design**
  - Explore and discover
  - Gain insight into problem
  - Evaluate options, trade-offs

- **Implementation**
  - Migrate design to production
  - Optimize performance
  - Deploy / Integrate / Test

- **Test & Verification**
  - .NET
  - dll
  - C/C++
  - .exe
  - Verilog
  - VHDL
Representative Tools Used in Algorithm Development

- General-purpose languages
  - C, C++
  - Fortran, Java

- Technical computing languages
  - MATLAB

- Specialized and single-purpose tools
  - Image processing application
  - Speech recognition application

- Spreadsheets
  - Excel
Algorithm Development Process

**Requirements**

**Research & Design**
- Explore and discover
- Gain insight into problem
- Evaluate options, trade-offs

**Implementation**
- Migrate design to production
- Optimize performance
- Deploy / Integrate / Test

**Test & Verification**
Design the Lane Detection System

- Process images from actuated camera mounted on vehicle
- Prototype algorithm for identifying lanes in the road
- Elaborate algorithm to process streaming video
MATLAB Capabilities for Algorithm Design

- Interactively develop algorithms
- Edit and debug large programs
- Profile code performance
- Publish reports
- Use application-specific toolboxes
Algorithm Development Process

Requirements

Research & Design

- Explore and discover
- Gain insight into problem
- Evaluate options, trade-offs

Implementation

- Migrate design to production
- Optimize performance
- Deploy / Integrate / Test
Verify the Lane Detection System Harness

- Test the harness using working MATLAB code
- Leverage the MATLAB Engine interface to connect C with MATLAB
Plotting the Data in Your C Project Using the MATLAB Engine Library

- Add header file and libraries to your project
- Package data into MATLAB data structure (mxArray)
- Use engine routines to:
  - Send data to MATLAB
  - Perform computations in MATLAB
  - Plot data in MATLAB
- On Windows, enable MATLAB Automation Server
# C Engine Routines: Call MATLAB from C

## External Interfaces

### The Engine Library

The engine library contains the following routines for controlling the MATLAB computation engine. Their names all begin with the three-letter prefix `eng`. These tables list all the available engine functions and their purposes.

## C Engine Routines

<table>
<thead>
<tr>
<th>Function</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>engOpen</code></td>
<td>Start up MATLAB engine</td>
</tr>
<tr>
<td><code>engClose</code></td>
<td>Shut down MATLAB engine</td>
</tr>
<tr>
<td><code>engGetVariable</code></td>
<td>Get a MATLAB array from the MATLAB engine</td>
</tr>
<tr>
<td><code>engPutVariable</code></td>
<td>Send a MATLAB array to the MATLAB engine</td>
</tr>
<tr>
<td><code>engEvalString</code></td>
<td>Execute a MATLAB command</td>
</tr>
<tr>
<td><code>engOutputBuffer</code></td>
<td>Create a buffer to store MATLAB text output</td>
</tr>
<tr>
<td><code>engOpenSingleUse</code></td>
<td>Start a MATLAB engine session for single, nonshared use</td>
</tr>
<tr>
<td><code>engGetVisible</code></td>
<td>Determine visibility of MATLAB engine session</td>
</tr>
<tr>
<td><code>engSetVisible</code></td>
<td>Show or hide MATLAB engine session</td>
</tr>
</tbody>
</table>
Algorithm Development Process

Requirements

Research & Design
- Explore and discover
- Gain insight into problem
- Evaluate options, trade-offs

Implementation
- Migrate design to production
- Optimize performance
- Deploy / Integrate / Test

Test & Verification

Tools:
- Verilog
- .NET
- .dll
- C/C++
- .exe
- Java
- VHDL
Introducing MATLAB Coder
Generate C Source Code

MATLAB Coder: MEX Function
File Edit Project Debug Desktop Window Help
SimpleExample.prj
Overview Build
Entry-Point Files
ex01.m
   b single(100 x 100)
c single(100 x 1)

Global Variables
If you use global variables in your MATLAB algorithm, add a global type definition and initial value for each before building the project. If you do not do this, you must create a variable in the global workspace.

[Add global]
Automatic Translation of MATLAB to C

With MATLAB Coder, design engineers can

- Maintain one design in MATLAB
- Design faster and get to C/C++ quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB
Implementing the Algorithm
Generate C Source Code

- Integrate algorithm into test harness

- Use MATLAB Coder to generate C source code from MATLAB algorithm
Implementation Constraints

- Polymorphism
- Memory allocation
- Processing matrices & arrays
- Fixed-point data types

7 Lines of MATLAB
107 Lines of C
MATLAB Language Support for Code Generation

- visualization
- variable-sized data
- struct
- functions
- malloc
- numeric
- System objects
- fixed-point
- arrays
- global
- persistent
- complex
- classes
- System objects
- sparse
- cell arrays
- nested functions
- Java
- graphics
- MATLAB Language Support for Code Generation

17
Supported MATLAB Language Features and Functions

Broad set of language features and functions/system objects supported for code generation

<table>
<thead>
<tr>
<th>Matrices and Arrays</th>
<th>Data Types</th>
<th>Programming Constructs</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Matrix operations</td>
<td>• Complex numbers</td>
<td>• Arithmetic, relational, and logical operators</td>
<td>• MATLAB functions and subfunctions</td>
</tr>
<tr>
<td>• N-dimensional arrays</td>
<td>• Integer math</td>
<td>• Program control (if, for, while, switch)</td>
<td>• Variable-length argument lists</td>
</tr>
<tr>
<td>• Subscripting</td>
<td>• Double/single-precision</td>
<td></td>
<td>• Function handles</td>
</tr>
<tr>
<td>• Frames</td>
<td>• Fixed-point arithmetic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Persistent variables</td>
<td>• Characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Global variables</td>
<td>• Structures</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Numeric class</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Variable-sized data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MATLAB Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• System objects</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported algorithms
• More than 1100 MATLAB operators, functions, and System objects for:
  • Communications
  • Computer vision
  • Image processing
  • Phased Array signal processing
  • Signal processing
  • Statistics
Supported Functions

- Aerospace Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- DSP System Toolbox
- Image Processing Toolbox
- Phased Array System Toolbox
- Signal Processing Toolbox
- Statistics Toolbox

1100+
MATLAB Coder Use Cases

- Hand off C code to software engineers
- Integrate algorithms into C/C++ simulations
- Accelerate user-written MATLAB algorithms (via MEX)
- Migrate algorithms to embedded processors (using Embedded Coder)
Algorithm Development Process

Requirements

Research & Design
- Explore and discover
- Gain insight into problem
- Evaluate options, trade-offs

Implementation
- Migrate design to production
- Optimize performance
- Deploy / Integrate / Test
MATLAB Compiler and MATLAB Coder
Implementing the Algorithm

Generate C Shared Library

- Integrate algorithm into test harness
- Use MATLAB Compiler to create C shared library from MATLAB algorithm
Deploying Applications with MATLAB

1. MATLAB Application
2. MATLAB Compiler
   - MATLAB Desktop
3. .dll
   - End-User Machine
   - MATLAB Compiler Runtime (MCR)
# MATLAB Compiler and MATLAB Coder

<table>
<thead>
<tr>
<th></th>
<th>MATLAB Compiler</th>
<th>MATLAB Coder</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td>Executable or software component/library</td>
<td>Portable and readable C source code</td>
</tr>
<tr>
<td><strong>MATLAB language support</strong></td>
<td>Full</td>
<td>Subset</td>
</tr>
<tr>
<td><strong>Additional libraries</strong></td>
<td>MATLAB Compiler Runtime (MCR)</td>
<td>None</td>
</tr>
<tr>
<td><strong>Supported toolboxes</strong></td>
<td>Most toolboxes</td>
<td>Some toolboxes</td>
</tr>
<tr>
<td><strong>License model</strong></td>
<td>Royalty-free</td>
<td>Royalty-free</td>
</tr>
<tr>
<td><strong>Extensions</strong></td>
<td>Builder Products</td>
<td>Embedded Coder</td>
</tr>
</tbody>
</table>
Summary
Reduce Development Time and Cost

- Develop algorithm in MATLAB
  - High level language
  - Existing functions available within the MATLAB product family

- Move quickly to implementation
  - MATLAB Compiler and builders for deployable applications
  - MATLAB Coder for ANSI C code of algorithms

- Distribute your application freely
  - No need for a MATLAB license
Q&A