모델 기반 설계를 통한 이미지 프로세싱 시스템 개발

Image Processing System Development with Model Based Design

MathWorks
Senior Application Engineer
Takuya Otani (大谷卓也)
Model-Based Design Adoption Grid

- Model Verification and Validation
- Formal Analysis of Models
- Production Code

System Design and Simulation

- Prototyping System on PC
- Hardware-in-the-Loop Simulation

Algorithm Exploration

- Prototyping on PC
- Integrate C-Code with legacy code

Code Generation
Agenda

- Algorithm Development Using MATLAB
  - Development of Image Processing Algorithm
  - Acceleration and Prototyping on PC
    - Parallel Computing
    - GPU Computing
    - C Code Generation
  - Custom GUI for Analysis

- System Design Using Simulink
  - Camera Pipeline using Simulink
  - C Code Generation
  - HDL Code Generation
From Sensor Data to Image
(Converting raw data from image sensor to color adjusted RGB)

Image Sensor

Bayer pattern
Color Filter

Sensor Arrays

Raw Image

RGB Color Image
Example of digital camera pipeline
(Processes may be different for different cameras)
Development of Tone Mapping Algorithm

Color Chart Image
(Picture taken under D50 lighting condition)

Identify RGB Values from grids

sRGB Reference Data
- Read-in Color Chart sRGB reference data

RGB↔sRGB
Polynomial Curve Fitting

Create lookup table
Demo

- Development of tone mapping algorithm
- Development of a camera pipeline
Tone Mapping
(Importing sRGB reference data from ColorChecker)

```matlab
data = xlsread('color_checker_ref.xls');
ref_rgb_val = uint16(data(:, 4:6));
```

xlsread command to import Excel data to MATLAB workspace.
(xlswrite to write MATLAB workspace data to Excel)

Create Reference Table
Tone Mapping
(Using Raw Data Reader From C code)

C based decode function
(dcraw.C *1)

Adding gateway routine

```c
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
    int returnValue, idx;
    int nInputArguments;
    char *pInputArguments[NPARAMETERS];
    ...
}
```

MEX Function
(Compiled code can run easily from MATLAB command line)

```
>> mex dcraw.c
```

Using MEX functionality, legacy functions written in C/C++ or Fortran language can be compiled on MATLAB, and use the code as a MATLAB function.

*1: Open source raw data reader created by David Coffin
http://www.cybercom.net/~dcoffin/dcraw/
Tone Mapping
(Curve fitting camera raw data to reference)

Curve Fitting Toolbox GUI (cftool)
Validate Tone Mapping Result
(Using various MATLAB visualization)

Compare sRGB reference data against tone mapped camera RGB data.

Example MATLAB Code to convert sRGB to $L^*a^*b^*$

```
>> C = makecform('srgb2lab');
>> lab_image = applycform(rgb_image,C);
```

Verify the result using $RGB$ color space

Verify the result using $L^*a^*b^*$ color space
Example of digital camera pipeline
(Processes may be different for different cameras)
Noise Reduction
(One method to use: Median filter)

Intensity data around 3x3 pixel area
20, 10, 15, 22, 89, 30, 18, 24, 13

Sort
10, 13, 15, 18, 20, 22, 24, 30, 89

Replace the center value
20, 10, 15, 22, 20, 30, 18, 24, 13

Notice the value at center is much larger than the others.
⇒ Potential electrical or physical noise in the image.
Example of how median filter performs

MATLAB Code Example (Grayscale image example)

```matlab
>> Image = imnoise(I,'salt & pepper', 0.02); % Add Noise
>> ImageF = medfilt2(Image,[3 3]); % Remove Noise
```
Demosaic
(In case of bilinear algorithm)

RAW Data (Bayer Pattern)
(In case of “RGGB”)

(B1+B2) / 2
(B1+B3) / 2
(B3+B4) / 2
(B2+B4) / 2

(R1+R2) / 2
(R1+R3) / 2
(R2+R4) / 2
(R3+R4) / 2

(R1+R2+R3+R4) / 4

(G1+G2+G3+G4) / 4

(R1+R2) / 2
(R2+R4) / 2
(R1+R2+R3+R4) / 4
Performing of Demosaic

Image Sensor Raw Data

Perform Interpolation

Demosaiced RGB Image

Image Processing Toolbox Example

>> rgb_image = demosaic(raw_image, 'rggb');
White Balancing / Gamma Correction

Example Of “Gray World” Implementation

Find R/G/B correction factor

\[ R\_factor = \frac{\text{mean of RGB}}{\text{mean of R}} \]
\[ G\_factor = \frac{\text{mean of RGB}}{\text{mean of G}} \]
\[ B\_factor = \frac{\text{mean of RGB}}{\text{mean of B}} \]

Correct the illumination using “Gray World” algorithm

Gamma Correction (imadjust from Image Processing Toolbox)

Picture taken in a dark room
Agenda

- Algorithm Development Using MATLAB
  - Development of Image Processing Algorithm
  - Acceleration and Prototyping on PC
    - Parallel Computing
    - GPU Computing
    - C Code Generation
  - Custom GUI for Analysis

- System Design Using Simulink
  - Camera Pipeline using Simulink
  - C Code Generation
  - HDL Code Generation
Parallel Computing Products

MATLAB Distributed Computing Server

Multicore CPU

GPU

Parallel Computing Toolbox
Parallel Computing enables you to …

Larger Compute Pool

Speed up Computations

Larger Memory Pool

Work with Large Data
GPU Support with Parallel Computing Toolbox

- NVIDIA GPUs with compute capability 1.3 or greater
  - Includes Tesla 10-series and 20-series products
    (e.g., NVIDIA Tesla C2075 GPU: 448 processors, 6 GB memory)
  - GPU enabled Image Processing Toolbox functions:
    - imrotate(), imfilter(), imdilate(), imerode(), imopen(),
    - imclose(), imtophat(), imbothat(), imshow(), padarray(),
    - bwlookup()

Example: imfilter : 37x37 Filter on 3840 x 5120 pixels image
13 seconds (CPU Only ) ➞ 1 seconds (GPU: Tesla C2050)
Automatic Translation of MATLAB to C
MATLAB Coder

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
Demo

- Parallel processing of data
- Running a camera pipeline in GPU
- C Code generation for PC based deployment
Agenda

- **Algorithm Development Using MATLAB**
  - Development of Image Processing Algorithm
  - Acceleration and Prototyping on PC
    - Parallel Computing
    - GPU Computing
    - C Code Generation
  - Custom GUI for Analysis

- **System Design Using Simulink**
  - Camera Pipeline using Simulink
  - C Code Generation
  - HDL Code Generation
GUIDE application allows easy-to-use GUI creation on MATLAB

Based on the layout of buttons, and menus, MATLAB automatically creates a template code
Using MATLAB for Algorithm Development

Easy to use interpreted programming *environment*
- Easy handling of vectors and matrices (minimum “loops” in the code)
- Scripting environment for easy debug
- Various algorithms for many applications
- 2D, 3D visualization for analysis

Various options for code acceleration, and deployment
- Seamless utilization of GPGPU
- Easy to use parallel execution
- C code generation for deployment
Agenda

- Algorithm Development Using MATLAB
  - Development of Image Processing Algorithm
  - Acceleration and Prototyping on PC
    - Parallel Computing
    - GPU Computing
    - C Code Generation
  - Custom GUI for Analysis

- System Design Using Simulink
  - Camera Pipeline using Simulink
  - C Code Generation
  - HDL Code Generation
Simulink System Design Environment

Input from Hardware

EDA Connection
(Ex: Xilinx, Altera, Mentor Graphics, Cadence)

Co-Simulation with FPGA

Co-Simulation with CPU/DSP

IDE Connection
(Ex: Texas Instruments, Analog Devices, Green Hills, Altium, Eclipse)
Camera Pipeline in Simulink

- Median Filter (MATLAB Code)
- Demosaic
- Tone Mapping (Lookup Table)
- White Balance
- Gamma Correction
Fixed-Point Analysis

- Convert floating point to **optimized** fixed-point models
  - Automatic tracking of signal range (also intermediate quantities)
  - Word / Fraction lengths recommendation
- Bit-true models in the same environment

Automatically identify and solve fixed-point issues
Automatically generate bit true, cycle accurate HDL code from Simulink, MATLAB and Stateflow.
Co-Simulation of HDL with MATLAB/Simulink

**Test Bench**
- Re-use MATLAB/Simulink test benches
- Automation through cosimWizard

```matlab
% Simulate for 1000 samples
for ii=1:1000
    % Read 1 sample from the sinus generator
    ComplexSinus = step(SinGenerator);
    
    % Send/receive 1 sample to/from the HDL FFT
    [RealFft, ImagFft] = step(hdl_fft, real(ComplexSinus), imag(ComplexSinus));
    
    % Store the FFT sample in a vector
    ComplexFft(ii) = RealFft + ImagFft*ii;
end
```
Integrated with HDL Workflow Advisor
Demo

- Designing a camera pipeline in Simulink
- C/HDL code generation
- System verification using FPGA in the loop
Model-Based Design

- Research
- Requirements
- Design
  - Environment Models
  - Physical Components
  - Algorithms
- Implementation
  - C, C++
  - VHDL, Verilog
  - Structured Text
  - MCU, DSP, FPGA, ASIC, PLC
- Integration
- Test and Verification
  - Algorithm (IP) design using MATLAB
  - Code acceleration and generation for prototyping
  - System Design as Executable Specification using Simulink
  - Functional verification with software/hardware in the loop
  - Production code generation, and documentation
감사합니다

ありがとう
gozaimasu

Thank you!

© 2013 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.