

MathWorks
**AUTOMOTIVE
CONFERENCE 2023**
Europe

Targeting GPUs for Automotive Applications

Christoph Stockhammer, MathWorks



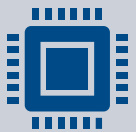
Key Takeaways



GPU Coder generates CUDA code from MATLAB & Simulink

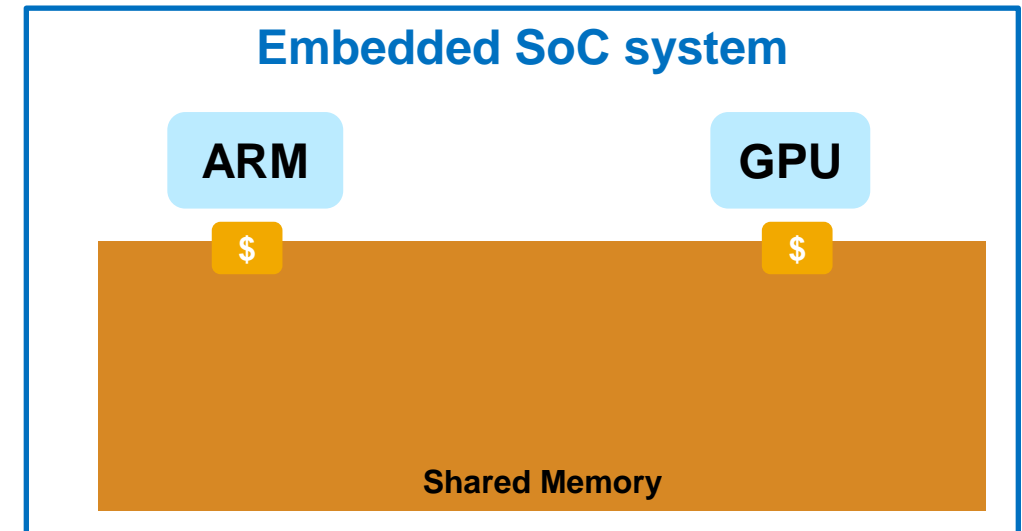
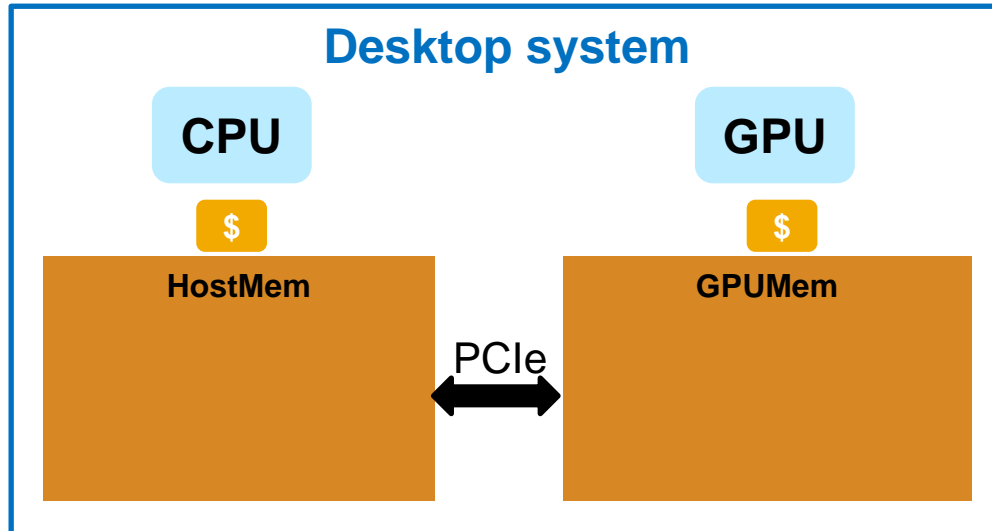


Accelerate MATLAB & Simulink simulations

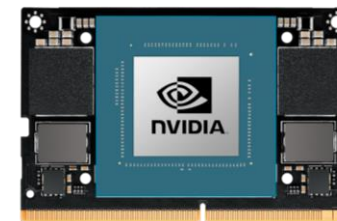


Deploy algorithms (signal/deep learning,...) to embedded GPUs

Types of GPUs



Desktop GPUs
(and Cloud GPUs)

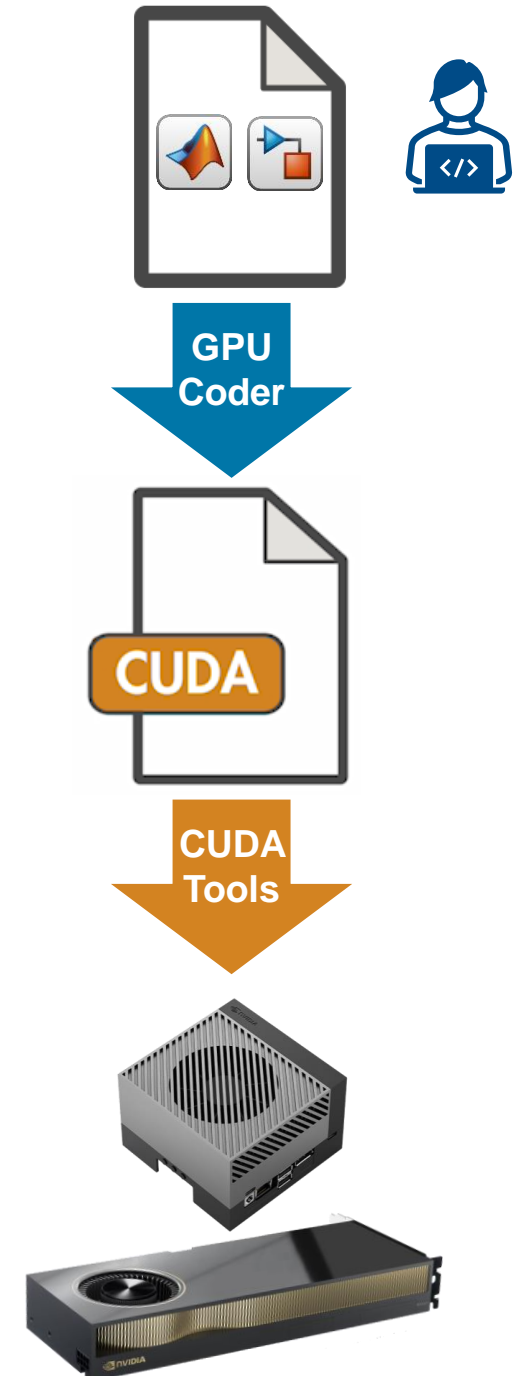


Embedded GPUs



CUDA code generation

- Generate optimized CUDA code from MATLAB and Simulink for deep learning, embedded vision, and autonomous systems
- Generated CUDA is portable across NVIDIA desktop GPUs
- Prototype algorithms on modern GPUs including the A100/ V100 and Jetson AGX Orin
- Accelerate computationally intensive portions of your MATLAB code and Simulink models using generated CUDA code

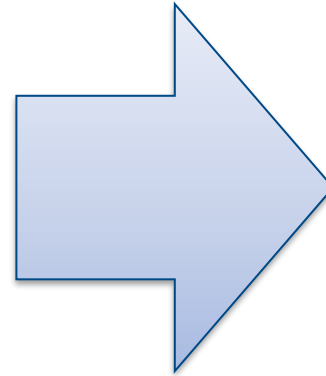




Why Use CUDA code generation ?

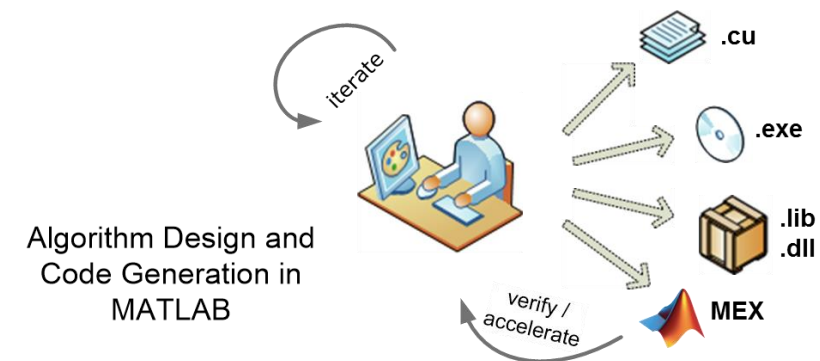
Pains: Hand code

- **Cannot code in CUDA**
- Time consuming
- Manual Coding Errors
- Multiple implementations
- Expensive



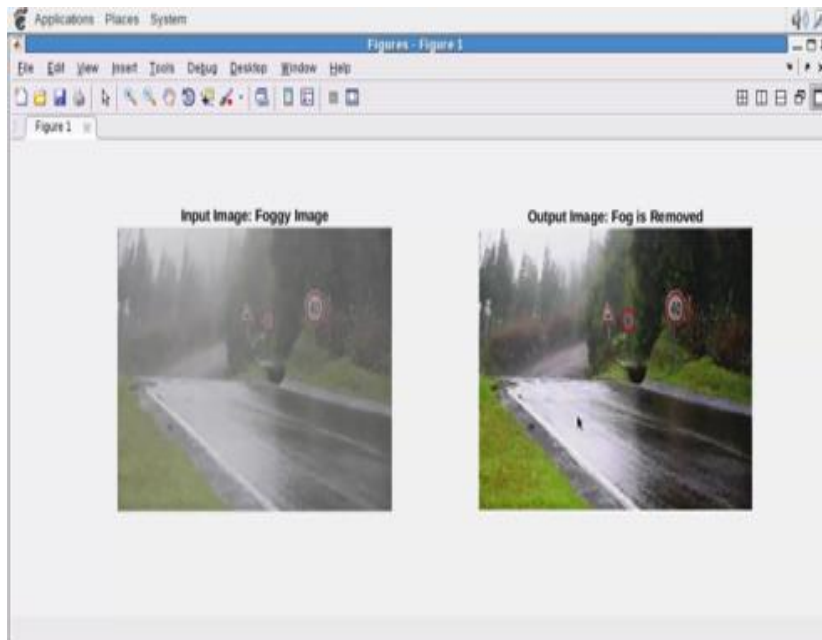
Solution: GPU Coder

- Automatically convert to CUDA
- Get to CUDA faster
- Eliminate manual coding errors
- Maintain Single “Truth”
- Stay within MATLAB/Simulink at a higher level

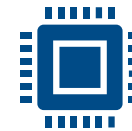
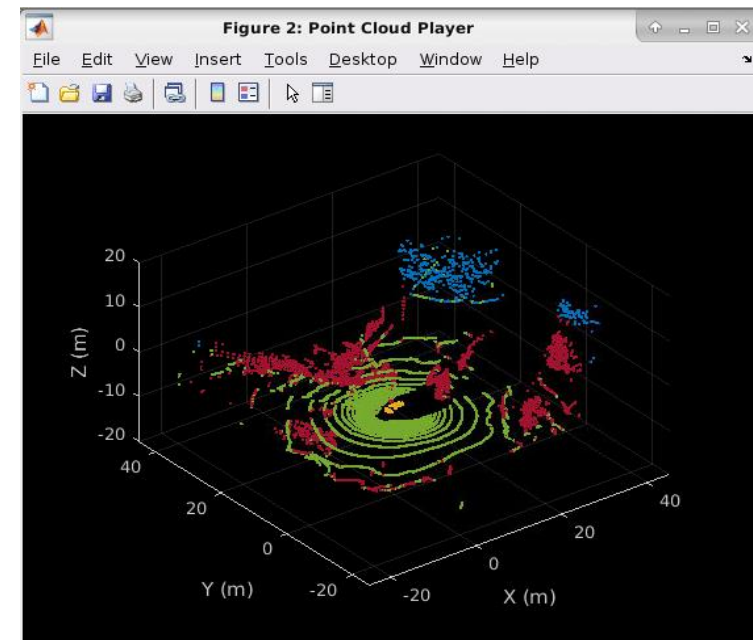


Two Application examples today

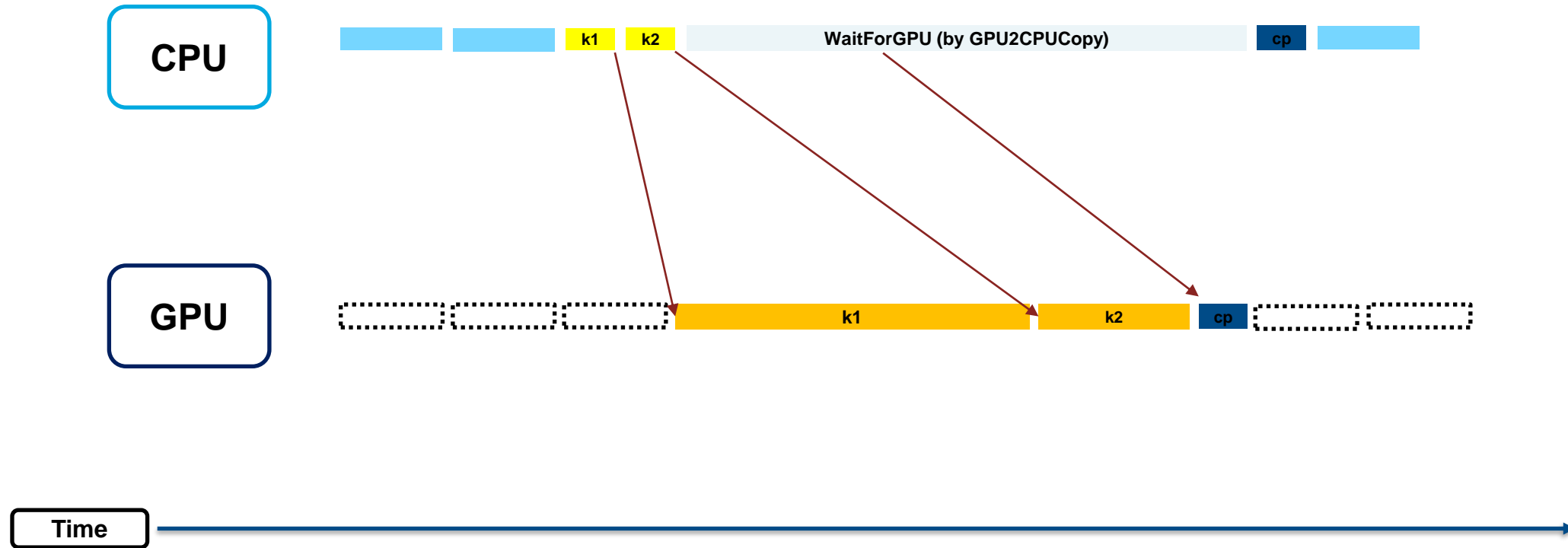
1) Fog Rectification



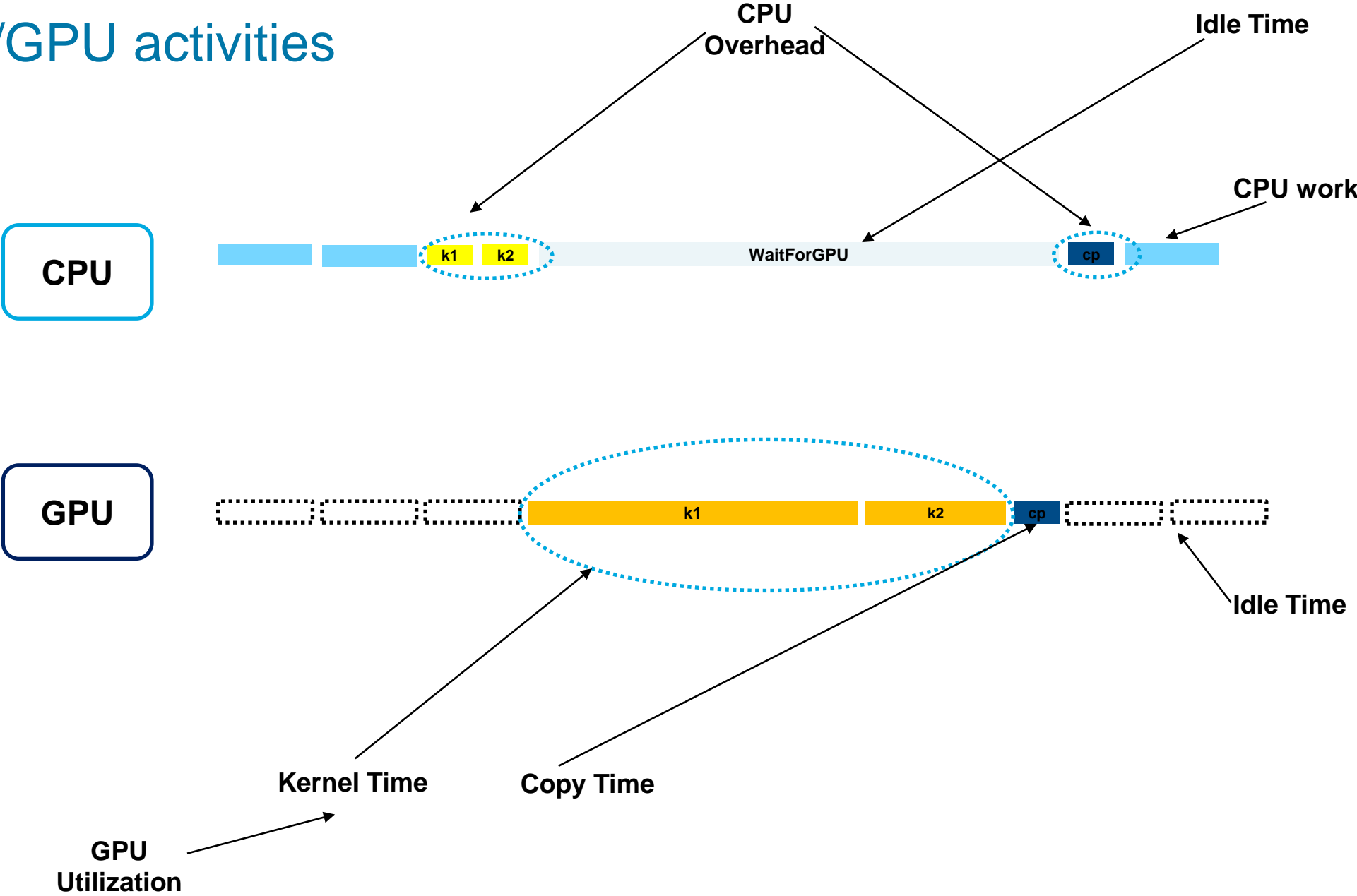
2) Lidar point cloud segmentation



Analyze CPU/GPU interaction for performance improvements



CPU/GPU activities



Example 1: Fog rectification + GPU Performance Analyzer



MATLAB R2023a

HOME PLOTS APPS LIVE EDITOR INSERT VIEW

Code Control Task Section Break Text Table of Contents Code Example Image Hyperlink Equation

CODE SECTION TEXT IMAGE LINK EQUATION

C:\Users\cstockha\OneDrive - MathWorks\demos\conferences\MAC\2023\fog_rectification\forRecording

Live Editor - C:\Users\cstockha\OneDrive - MathWorks\demos\conferences\MAC\2023\fog_rectification\forRecording\GPUExecutionProfilingOfTheGeneratedCodeExample.mlx

GPUExecutionProfilingOfTheGeneratedCodeExample.mlx fog_rectification.m

Analyze Performance of the Generated CUDA Code

This example shows you how to analyze and optimize the performance of the generated CUDA® code by using the `gpuPerformanceAnalyzer` function.

The GPU Coder Performance Analyzer runs a software-in-the-loop (SIL) execution that collects metrics on CPU/GPU activities in the generated code and provides a chronological timeline plot to visualize, identify, and mitigate performance bottlenecks in the generated CUDA code. This example generates the performance analysis report for the *Fog Rectification* example from GPU Coder. For more information, see [Fog Rectification](#).

Fog Rectification Algorithm

To improve the foggy input image, the algorithm performs fog removal and then contrast enhancement. The diagram shows the steps of both these operations.

```
graph LR; Input[Foggy input image] --> FogRemoval; subgraph FogRemoval; direction LR; A[Dark Channel Estimation] --> B[Air light map estimation]; B --> C[Air light map refinement]; C --> D[Restoration]; end; FogRemoval --> ContrastEnhancement; subgraph ContrastEnhancement; direction LR; E[RGB to Gray] --> F[Histogram calculation]; F --> G[Histogram normalization]; G --> H[CDF calculation]; H --> I[Contrast stretching]; end; ContrastEnhancement --> Output[Fog rectified output image];
```

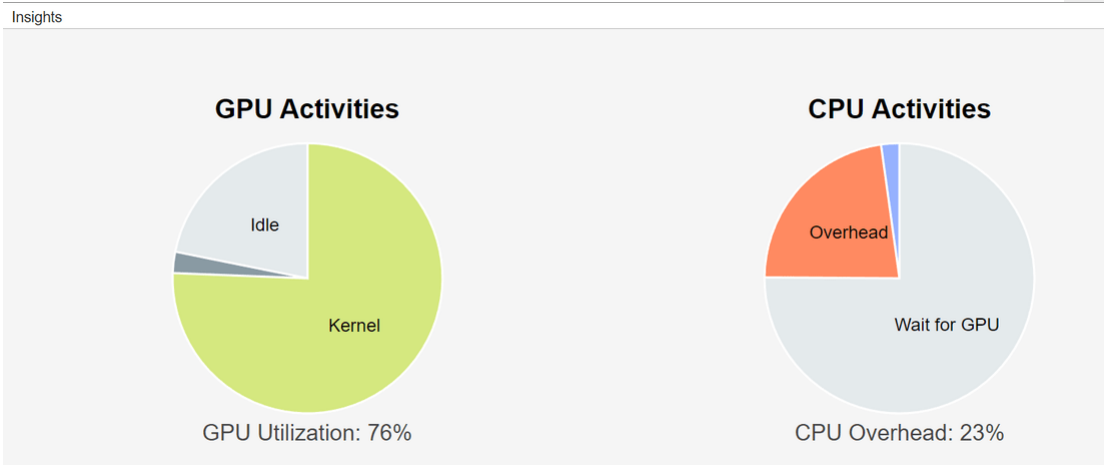
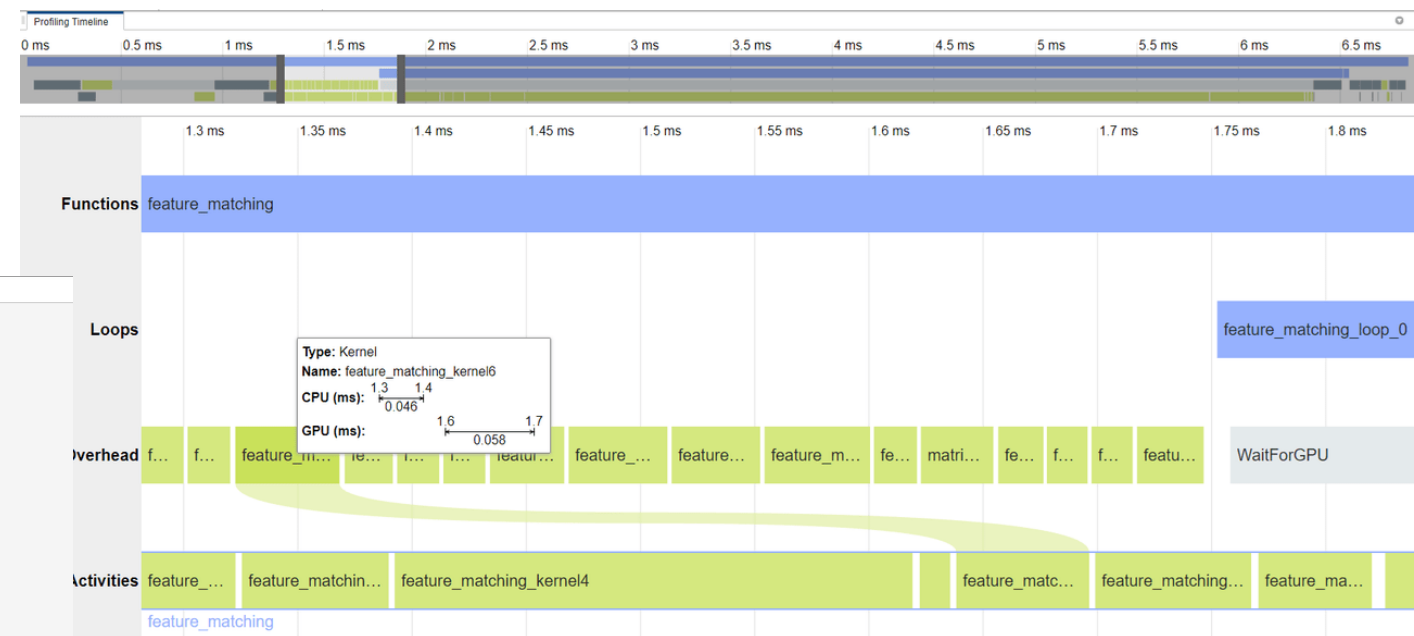
Zoom: 10... UTF-8 LF script



Code Profiling using GPU Performance Analyzer

- Profile and understand GPU and CPU activities, events, and performance metrics in a chronological timeline plot
- Use the profiling info to analyze and optimize the performance of the generated CUDA
- Visualize code metrics and identify optimization and tuning opportunities

Event Statistics	
Type	Kernel
Name	feature_matching_kernel2
Start time	1.209344 ms
End time	1.253440 ms
Duration	0.044096 ms
Launch Params	
Grid size	[261, 1, 1]
Block size	[512, 1, 1]
Total threads	133.63 K
Shared memory	0 Byte
Registers per thread	16



Bidirectional Traceability



Understand how GPU Coder maps the MATLAB algorithm to CUDA kernels

```

1 function [out] = fog_rectification(input) %#codegen
2
3 % Copyright 2017-2019 The MathWorks, Inc.
4
5 coder.gpu.kernelfun;
6
7 % restoreOut is used to store the output of restoration
8 restoreOut = zeros(size(input),'double');
9
10 % Changing the precision level of input image to double
11 input = double(input)./255;
12
13 %% Dark channel Estimation from input
14 darkChannel = min(input,[],3);
15
16 % diff_im is used as input and output variable for anis
17 diff_im = 0.9*darkChannel;
18 num_iter = 3;
19
20 % 2D convolution mask for Anisotropic diffusion
21 hN = [0.0625 0.1250 0.0625; 0.1250 0.2500 0.1250; 0.062
22 hN = double(hN);
23
24 %% Refine dark channel using Anisotropic diffusion.
25 for t = 1:num_iter
26     diff_im = conv2(diff_im,hN,'same');
27 end
28
29 %% Reduction with min
30 diff_im = min(darkChannel,diff_im);
31
32 diff_im = 0.6*diff_im ;
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73 // Changing the precision level of input image to d
74 cudaMemcpy(*gpu_input, input, 921600ULL, cudaMemcpyH
75 fog_rectification_kernel1<<<dim3(1800U, 1U, 1U), dim
76 *gpu_input, *b_gpu_input, *gpu_restoreOut);
77 // Dark channel Estimation from input
78 // diff_im is used as input and output variable for
79 fog_rectification_kernel2<<<dim3(600U, 1U, 1U), dim3
80 *b_gpu_input, *gpu_diff_im, *gpu_darkChannel);
81 // 2D convolution mask for Anisotropic diffusion
82 // Refine dark channel using Anisotropic diffusion.
83 for (idx = 0; idx < 3; idx++) {
84     fog_rectification_kernel3<<<dim3(605U, 1U, 1U), di
85     *gpu_expanded);
86     fog_rectification_kernel4<<<dim3(600U, 1U, 1U), di
87     *gpu_diff_im, *gpu_expanded);
88     cudaMemcpyToSymbol(const_b, b, 72ULL, 0ULL, cudaMe
89     fog_rectification_kernel5<<<dim3(15U, 20U, 1U), di
90     *gpu_expanded, *gpu_diff_im);
91 }
92 // Reduction with min
93 // Parallel element-wise math to compute
94 // Restoration with inverse Koschmieder's law
95 fog_rectification_kernel6<<<dim3(600U, 1U, 1U), dim3
96 *gpu_diff_im, *gpu_darkChannel);
97 fog_rectification_kernel7<<<dim3(600U, 1U, 1U), dim3
98 *b_gpu_input, *gpu_darkChannel, *gpu_diff_im, *g
99 fog_rectification_kernel8<<<dim3(1800U, 1U, 1U), dim
100 *gpu_restoreOut, *b_gpu_restoreOut);
101 //
102 // Stretching performs the histogram stretching of
103 // im is the input color image and p is cdf limit.
104 // out is the contrast stretched image and cdf is t
105 // density function and T is the stretching functio
106

```

GPU Coder for Image Processing and Computer Vision



Fog removal



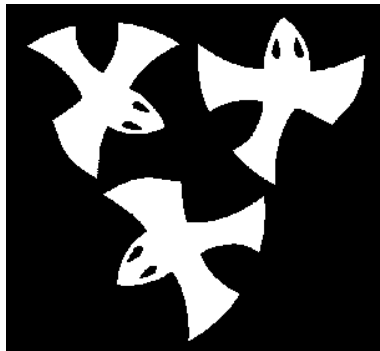
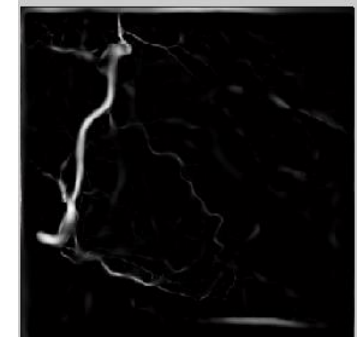
4x speedup



Frangi filter



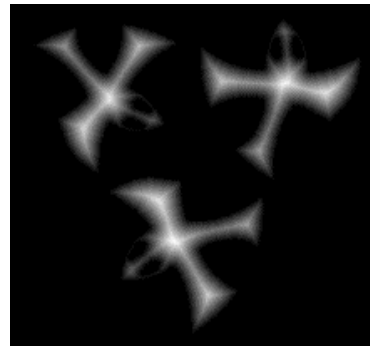
3x speedup



Distance transform



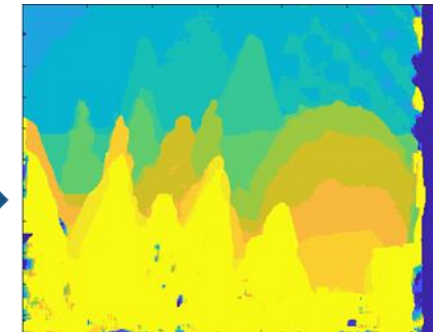
8x speedup



Stereo disparity



50x speedup



Ray tracing



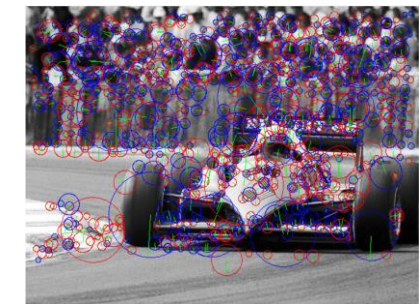
18x speedup



SURF feature extraction



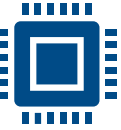
700x speedup



Optimizations for Generated CUDA Code



- Accelerated library support
 - cuFFT, cuBLAS, cuSolver, Thrust, cuDNN, & TensorRT
- Data Transfer Minimization
 - Analyzes data dependency between the CPU and GPU partitions to determine minimum set of locations where data must be copied between CPU and GPU using `cudaMemcpy`



Example 2: Lidar point cloud segmentation

The image shows the MATLAB R2023a Live Editor interface. The main window displays a presentation slide with the following content:

Ground Plane Segmentation and Obstacle Detection on NVIDIA Jetson Xavier™ NX Embedded platform

This example shows ground plane segmentation of 3-D lidar data from a vehicle on NVIDIA® embedded platforms to find nearby obstacles. The example uses ground plane segmentation and obstacle detection application to illustrate:

- C++ and CUDA® code generation for the ground plane segmentation and obstacle detection algorithm by using MATLAB® Coder™ and GPU Coder™.
- Verify behavior of the generated code on the target platform by using processor-in-the-loop (PIL) simulation.
- Compare of the performance of the application on the CPU (C++) and the GPU (CUDA).

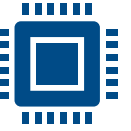
Third-Party Prerequisites

Target Board Requirements

- NVIDIA Jetson Xavier™ NX Embedded platform.

The interface also shows a file explorer on the left with folders like 'codegen' and 'ObstacleDetectionDeplo.'. The workspace on the right shows a variable 'ans' with a value of 0. The command window at the bottom shows the prompt 'fx >>'.

OK, but what about Simulink?



Tuning Parameters using External Mode

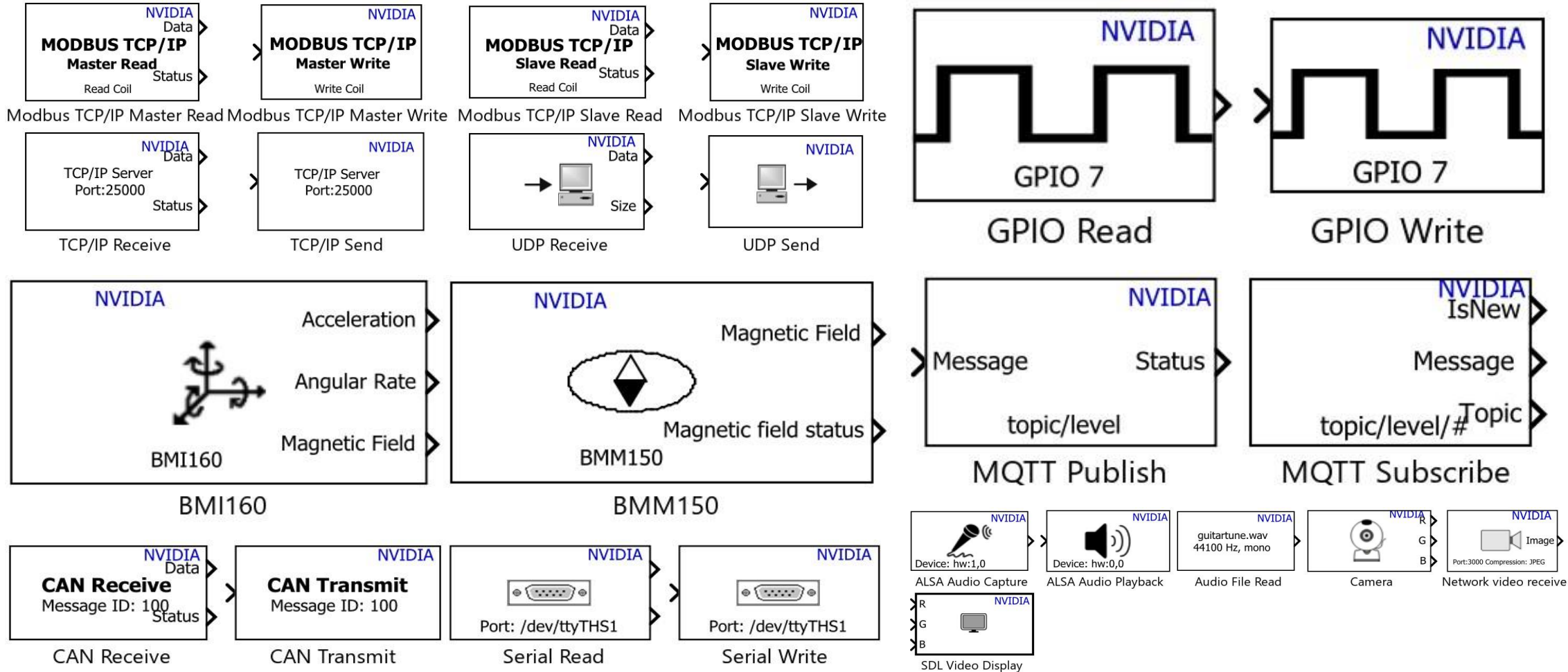
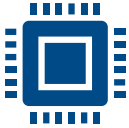
The screenshot displays the MATLAB/Simulink software interface in External Mode. The top menu bar includes SIMULATION, DEBUG, MODELING, FORMAT, HARDWARE, and APPS. The toolbar contains various simulation and analysis tools such as Open, Save, Print, Log Signals, Add Viewer, Signal Table, Stop Time, Normal, Fast Restart, Step Back, Run, Step Forward, Stop, Data Inspector, Logic Analyzer, and Bird's-Eye Scope.

The main workspace shows a Simulink model titled "sobelEdgeModelExternalMode2". The model consists of the following components:

- Input:** An image of various vegetables (peppers, tomatoes, etc.) is fed into the model.
- Sobel Edge Detector:** A central processing block that performs edge detection on the input image.
- Threshold:** A parameter block set to the value 14.2, which is used to filter the detected edges. A slider below the model allows for adjusting this threshold from 1 to 100.
- Output:** The results of the edge detection are output to three channels labeled R (Red), G (Green), and B (Blue), which are then displayed on an NVIDIA monitor icon.

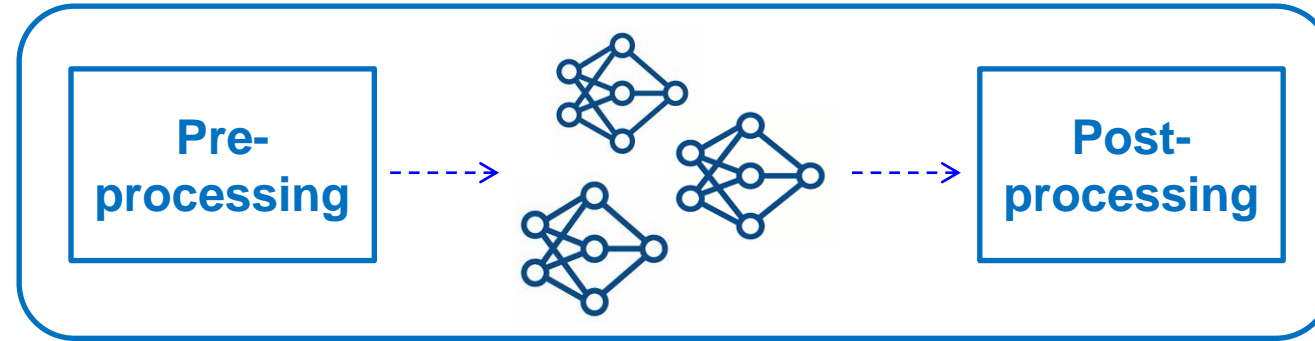
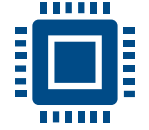
The Property Inspector on the right side of the interface shows the "Threshold:Value" property for the Sobel Edge Detector block.

NVIDIA Peripheral Support – block library



OK, but what about AI?

Deploy Complete Deep Learning Application



GPU Coder



Automatic Defect detection at [Airbus](#)



Deep Learning-based Visual inspection at [Musashi](#)



Shipping Examples

<https://www.mathworks.com/help/gpuCoder/examples.html>

Help Center

Hilfe-Center Suche

INHALTSVERZEICHNIS

« Documentation Home

« Examples

« Code Generation

Category

Fixed-Point Designer

GPU Coder

- Get Started with GPU Coder 2
- Kernel Creation 13
- Performance 3
- Deep Learning with GPU Coder 26
- Deployment 20
- HDL Coder
- HDL Verifier
- IEC Certification Kit
- MATLAB Coder
- Simulink Code Inspector
- Simulink Coder
- Simulink PLC Coder

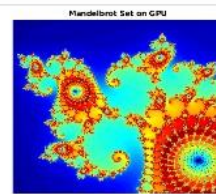
Type

- All 61
- MATLAB 48
- Simulink 13

Documentation Examples Functions Apps Videos Answers

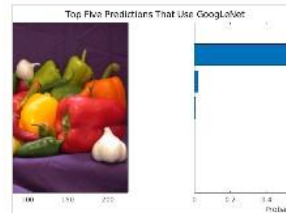
GPU Coder – Examples

Get Started with GPU Coder



GPU Code Generation: The Mandelbrot Set

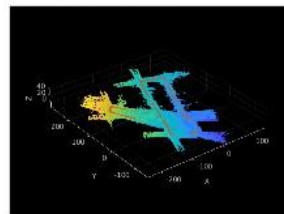
Generate CUDA® code from a simple MATLAB® function by using GPU Coder™. A Mandelbrot set implementation by using standard



Code Generation for Deep Learning Networks

Get started with CUDA code generation for image classification networks such as MobileNet-v2, ResNet, and GoogleNet.

Kernel Creation



Build a Map from Lidar Data



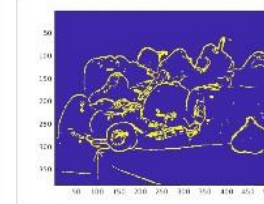
Feature Extraction Using



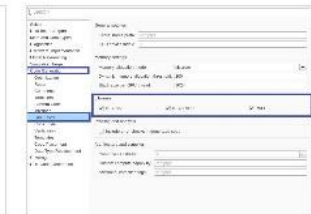
Feature Matching



Lane Detection on the GPU



Edge Detection with Sobel



GPU Code Generation for

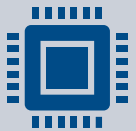
Key Takeaways



GPU Coder generates CUDA code from MATLAB & Simulink



Accelerate MATLAB & Simulink simulations



Deploy algorithms (signal/deep learning,...) to embedded GPUs

MathWorks
**AUTOMOTIVE
CONFERENCE 2023**
Europe

Thank you

