MATLAB EXPO

2024.06.11 | 그랜드 인터컨티넨탈 서울 파르나스

GPU기반 AI 어플리케이션 개발하기

신행재, 매스웍스코리아

MathWorks®

Key Takeaways

GPU Coder generates CUDA code from MATLAB & Simulink

Accelerate MATLAB & Simulink simulations

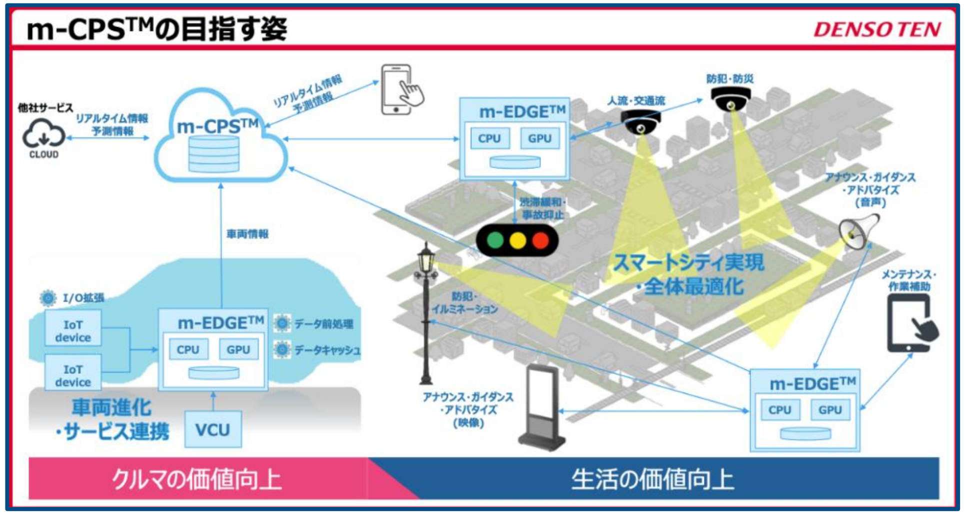
Deploy algorithms (signal/deep learning,…) to embedded GPUs

DENSO TEN Uses MATLAB to Develop Mobile Cyber-Physical System

DENSO TEN is developing mobility solutions using a cyber-physical system. The system consists of edge computers on vehicles and cloud AI. Mobility data collected from VCU and edge computers will be analyzed in the cloud to derive solutions for mobility problems.

Key Outcomes/Results

- DENSO TEN used MATLAB to develop and deploy a production-ready system that spanned hardware and cloud applications without manual recoding
- MATLAB enabled DENSO TEN to easily implement complex algorithms that utilized AI, image processing, probability calculations, and statistics
- MATLAB Production Server enabled DENSO TEN to run sophisticated analytics in a centralized location on the cloud



Mobility solution powered by edge devices and a centralized cloud to analyze and predict traffic flow.

> *"The superiority of MATLAB in data processing and visualization and its ability to consistently realize the entire process from conception to implementation is a major attraction and is the reason why we chose MATLAB for this project."*
>
> *- Natsuki Yokoyama, DENSO TEN*

Link to user story

Drass Develops Deep Learning System for Real-Time Object Detection in Maritime Environments

Challenge

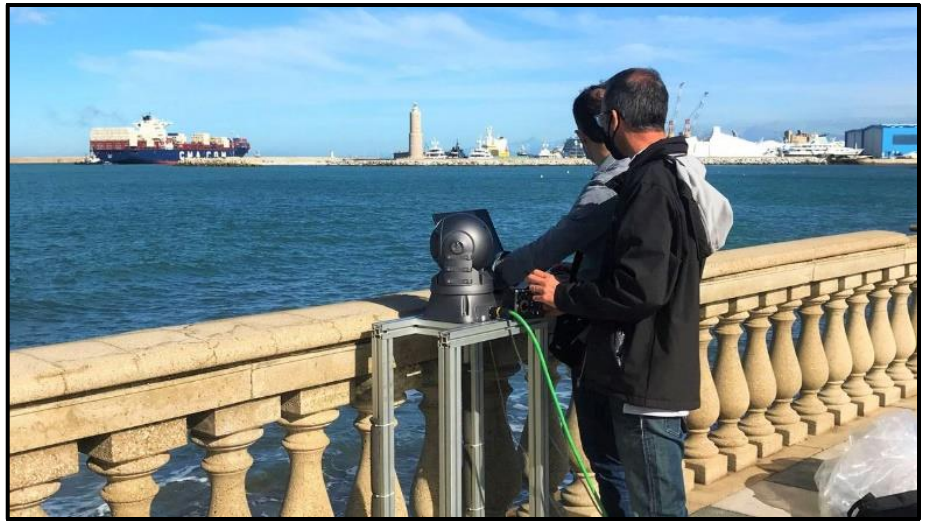
Help ship operators monitor sea environments and detect objects, obstacles, and other ships

Solution

Create an object-detection deep learning model that can be deployed on ships and run in real time

Results

- Data labeling automated
- Development time reduced
- Flexible and reproducible framework established



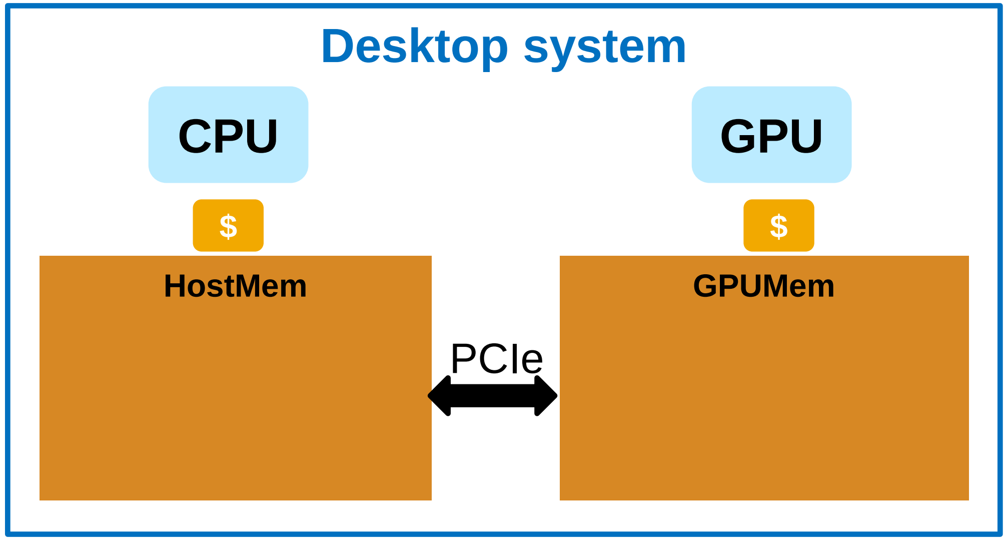
First day of object detection tests with optronic system prototype.

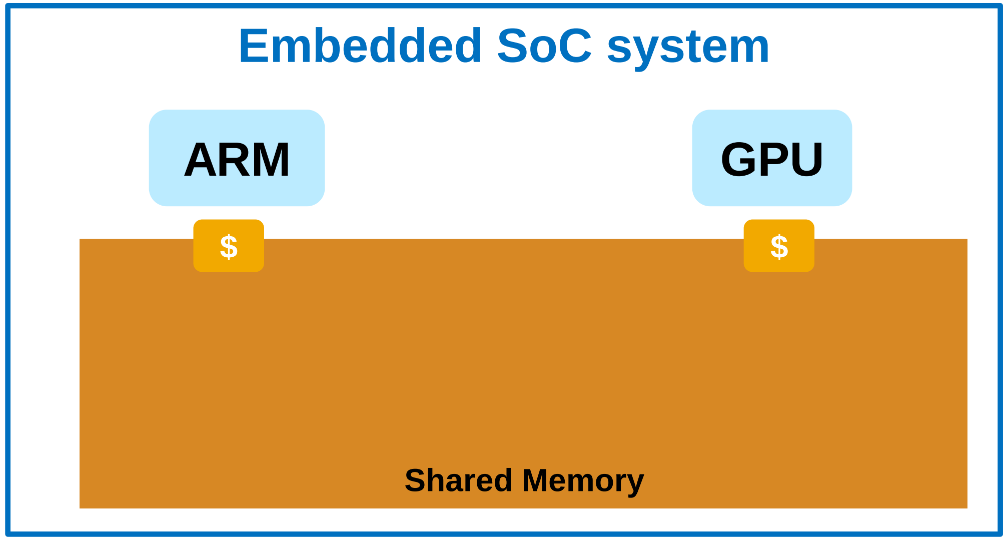
"From data annotation to choosing, training, testing, and fine-tuning our deep learning model, MATLAB had all the tools we needed—and GPU Coder enabled us to rapidly deploy to our NVIDIA GPUs even though we had limited GPU experience."

- Valerio Imbriolo, Drass Group

[Link to user story]

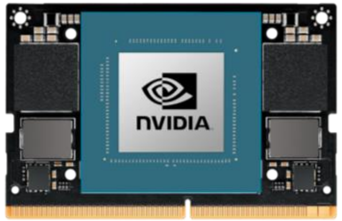
Types of GPUs





Desktop GPUs
(and Cloud GPUs)

Power Consumption:
300W~ vs 15~75W



Embedded GPUs

Industrial module: 10-year Operating Lifetime

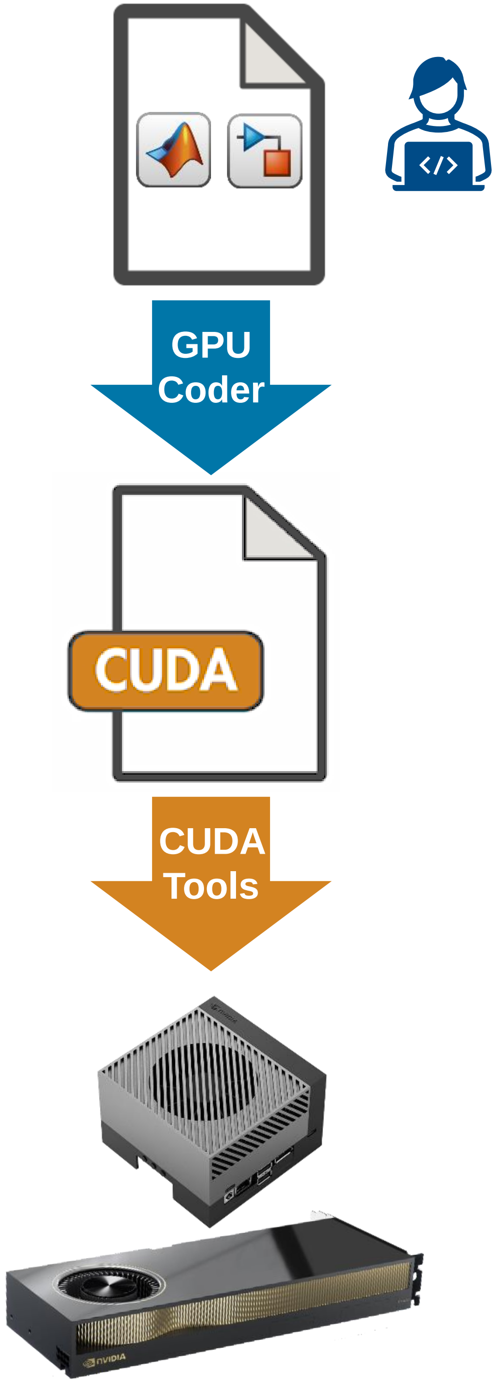
CUDA code generation

- 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
  Nvidia Data Center GPUs and Jetson AGX Orin

- Accelerate computationally intensive portions of your MATLAB code and
  Simulink models using generated CUDA code



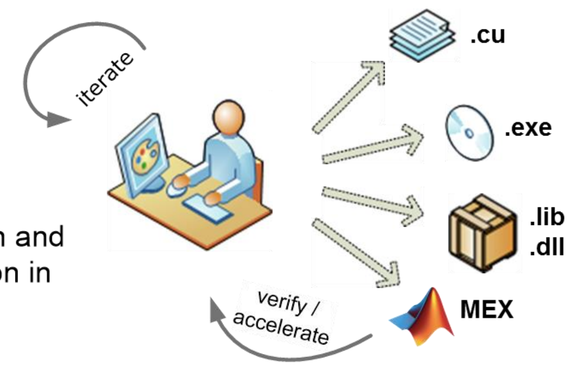
Why Use CUDA code generation ?

Pains: Hand code

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

Solution: GPU Coder

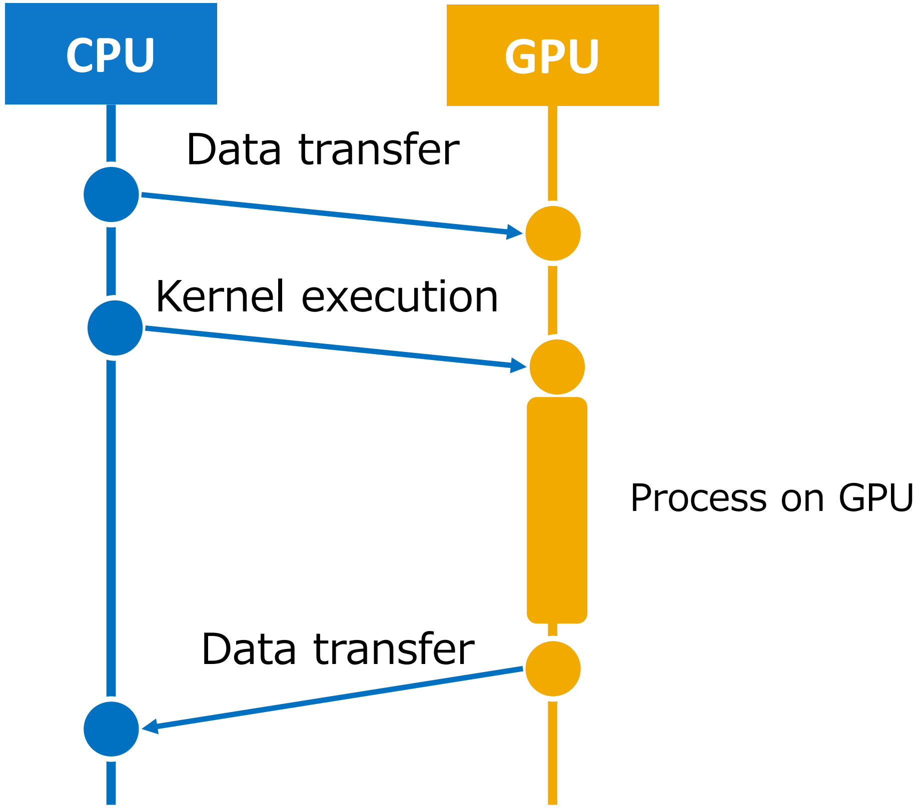
- Automatically convert to CUDA
- Get to CUDA faster
- Eliminate manual coding errors
- Maintain Single “Truth”
- Stay within MATLAB/Simulink at a higher level



Algorithm Design and Code Generation in MATLAB

What is CUDA?

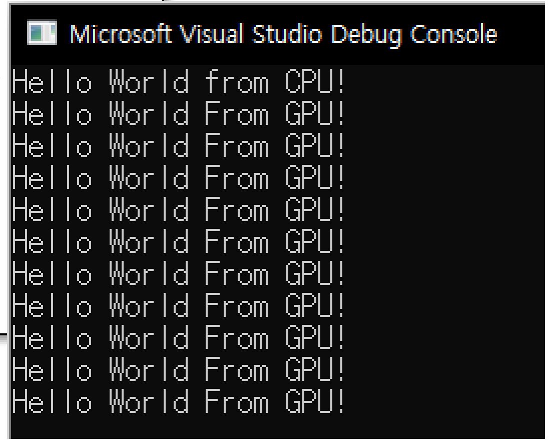
Run Hello World on GPU




```
__global__ void helloFromGPU()
{
   printf("Hello World from GPU!\n");
}

int main(int argc, char **argv)
{
   printf("Hello World from CPU!\n");

   helloFromGPU<<<1, 10>>>();
   return 0;
}
```



・Kernel call(special syntax)

```
kernelFunc<<<Block_dim, Thread_dim>>>(a, b, c);
```

For example, if you could do this ...

Linear Algebra routine, SAXPY example

Scalarized MATLAB

```
for i = 1:length(x)
    z(i) = a .* x(i) + y(i);
end
```

Vectorized MATLAB

```
z = a .* x + y;
```

GPU Coder

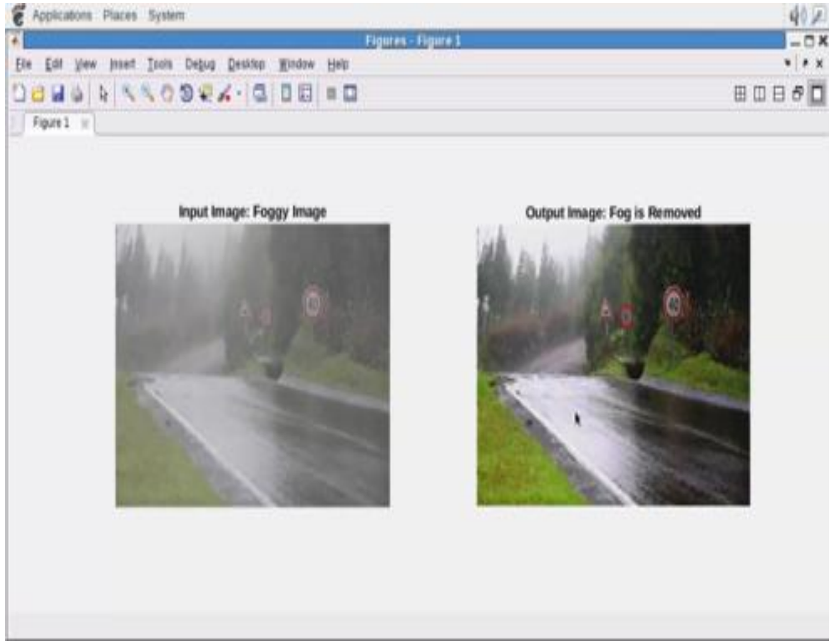
```
static __global__ __launch_bounds__(512, 1) void saxpy_kernel1(const real32_T *y,
  const real32_T *x, real32_T a, real_T *z)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (!(i >= 1048576)) {
    z[i] = (real_T)(a * x[i] + y[i]);
  }
}

void saxpy(real32_T a, const real32_T x[1048576], const real32_T y[1048576],
           real_T z[1048576])
{
  real32_T *gpu_y;
  real32_T *gpu_x;
  real_T *gpu_z;
  cudaMalloc(&gpu_z, 8388608UL);
  cudaMalloc(&gpu_x, 4194304UL);
  cudaMalloc(&gpu_y, 4194304UL);
  cudaMemcpy((void *)gpu_y, (void *)&y[0], 4194304UL, cudaMemcpyHostToDevice);
  cudaMemcpy((void *)gpu_x, (void *)&x[0], 4194304UL, cudaMemcpyHostToDevice);
  saxpy_kernel1<<<dim3(2048U, 1U, 1U), dim3(512U, 1U, 1U)>>>(gpu_y, gpu_x, a,
    gpu_z);
  cudaMemcpy((void *)&z[0], (void *)gpu_z, 8388608UL, cudaMemcpyDeviceToHost);
  cudaFree(gpu_y);
  cudaFree(gpu_x);
  cudaFree(gpu_z);
}
```

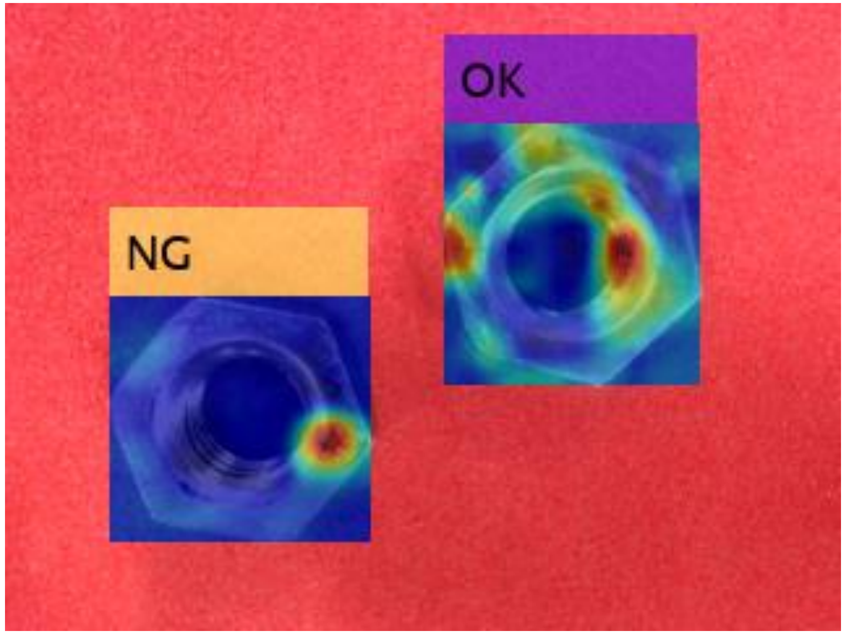
Automatic compilation from a highly extensible language to a high performance language

Two Application examples

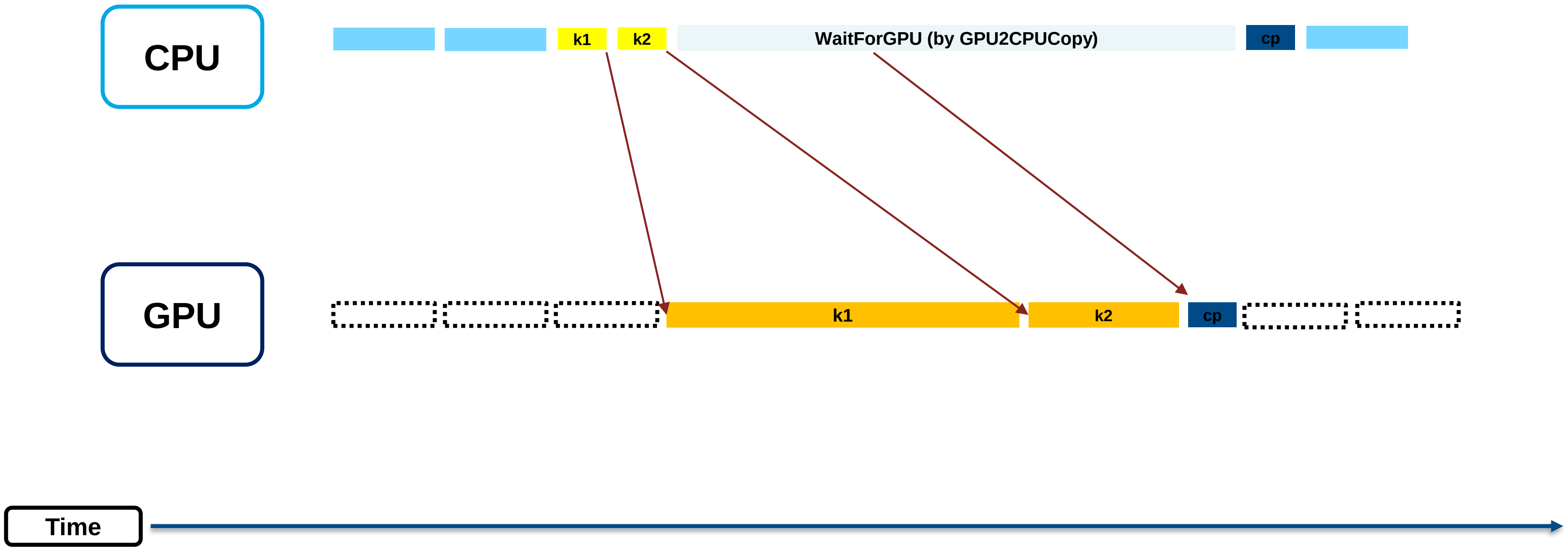
1) Fog Rectification



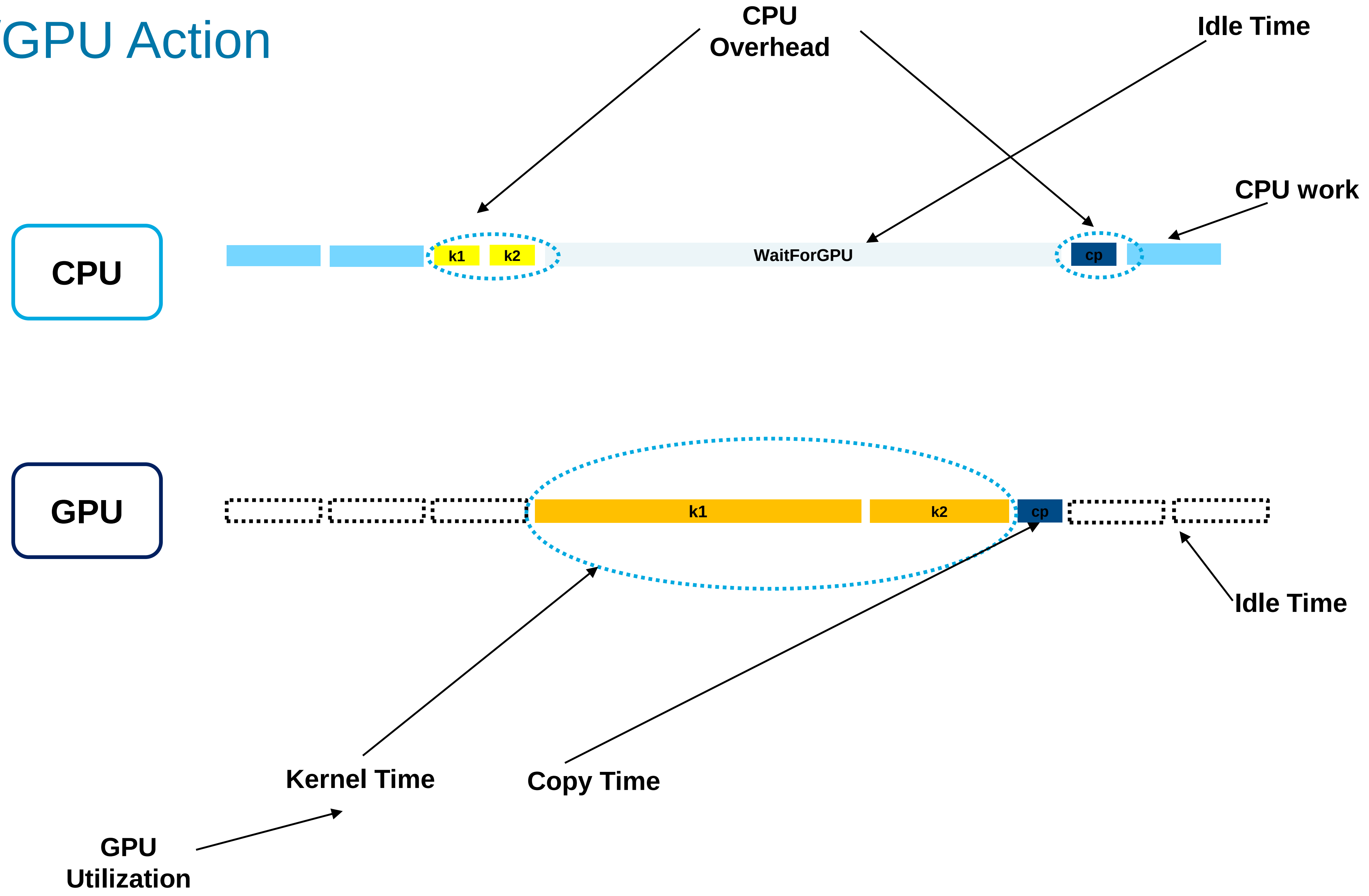
2) Anomaly Detection



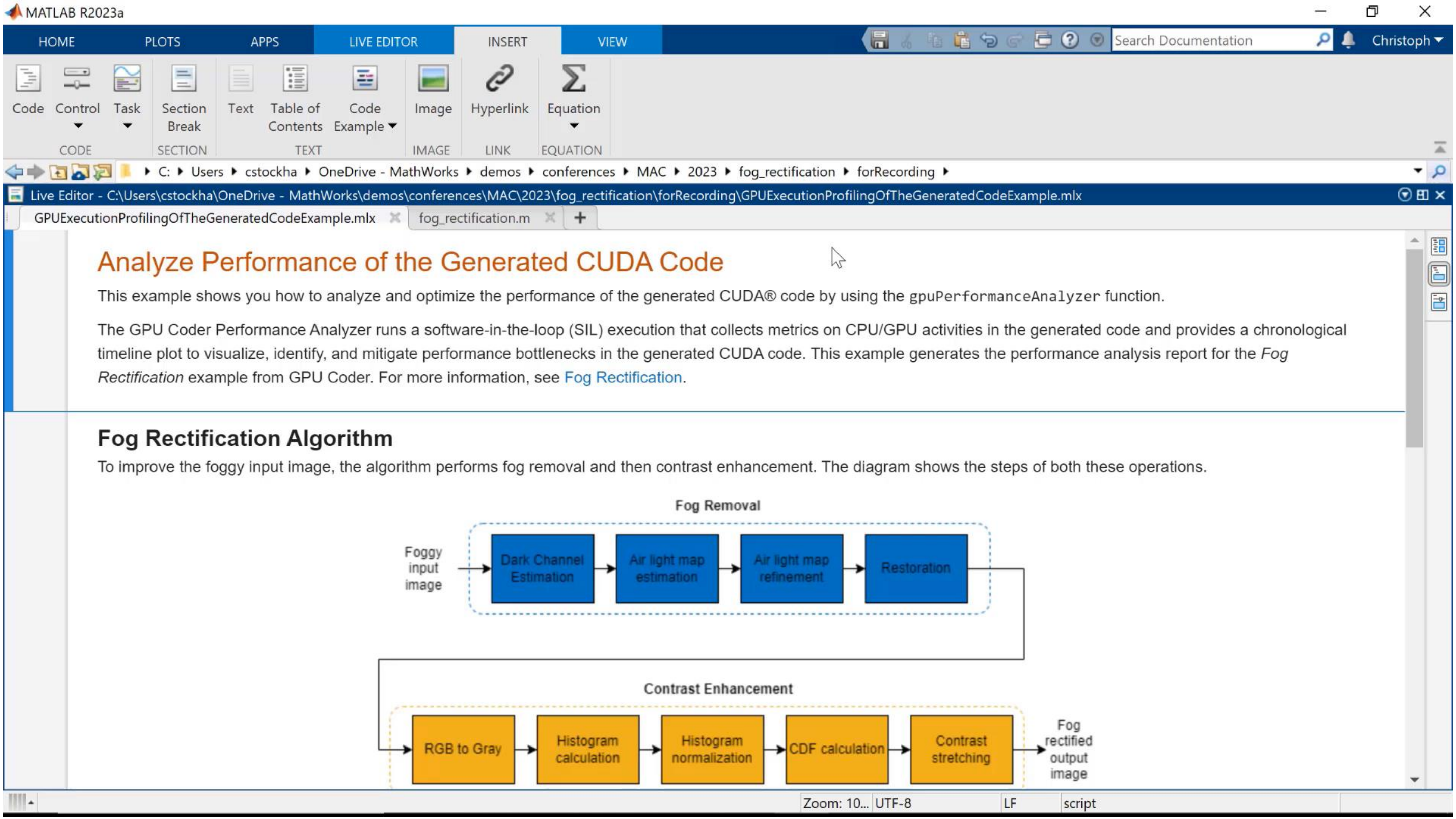
Analyze CPU/GPU interaction for performance improvements



CPU/GPU Action



Example 1: Fog rectification + GPU Performance Analyzer

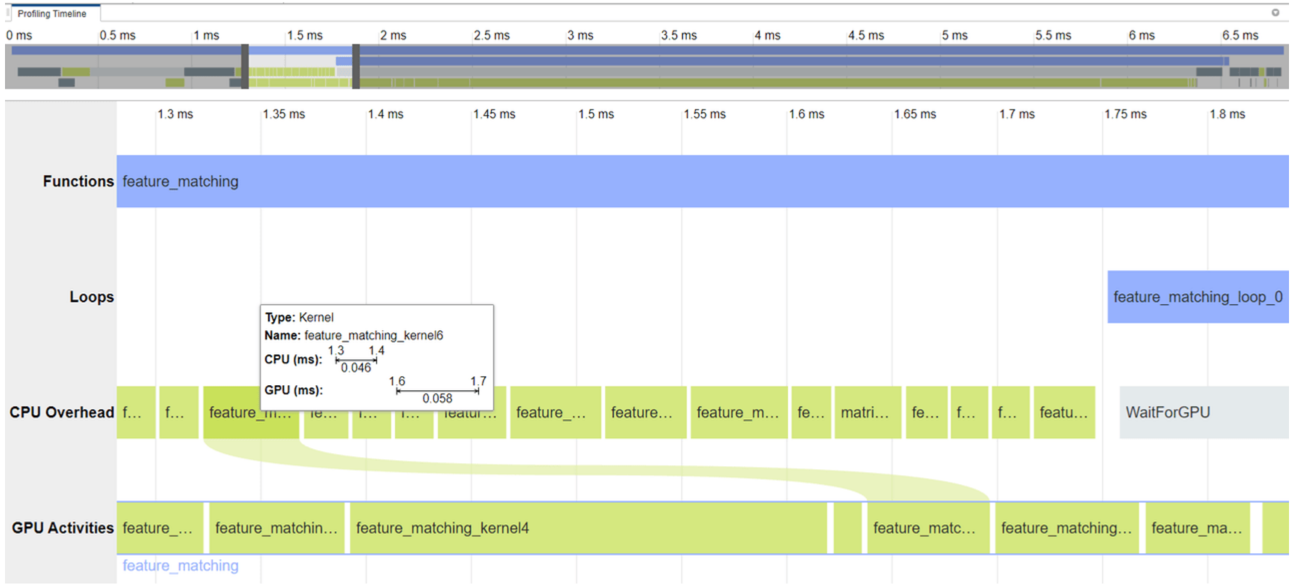


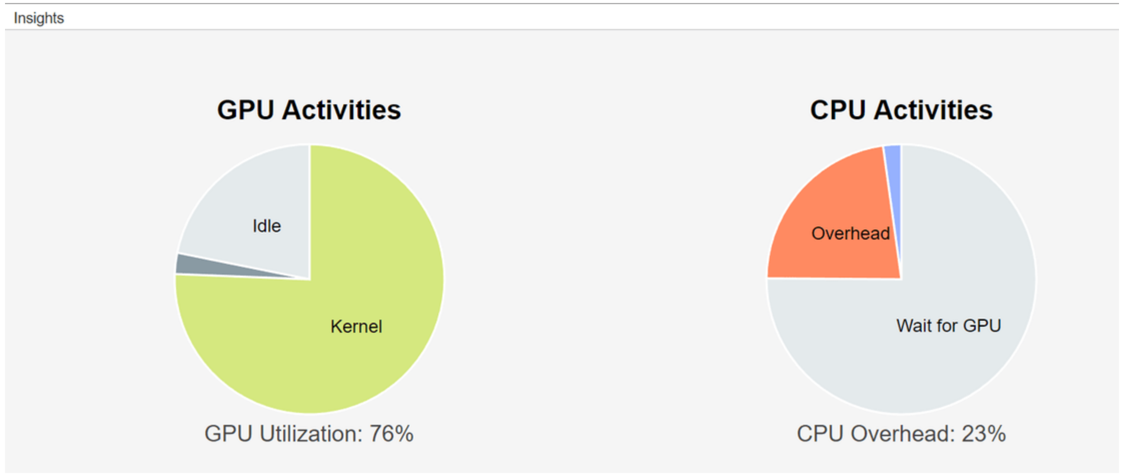
Code Profiling using GPU Performance Analyzer

MATLAB EXPO R2023a

Visualize code metrics and identify optimization and tuning opportunities

- 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



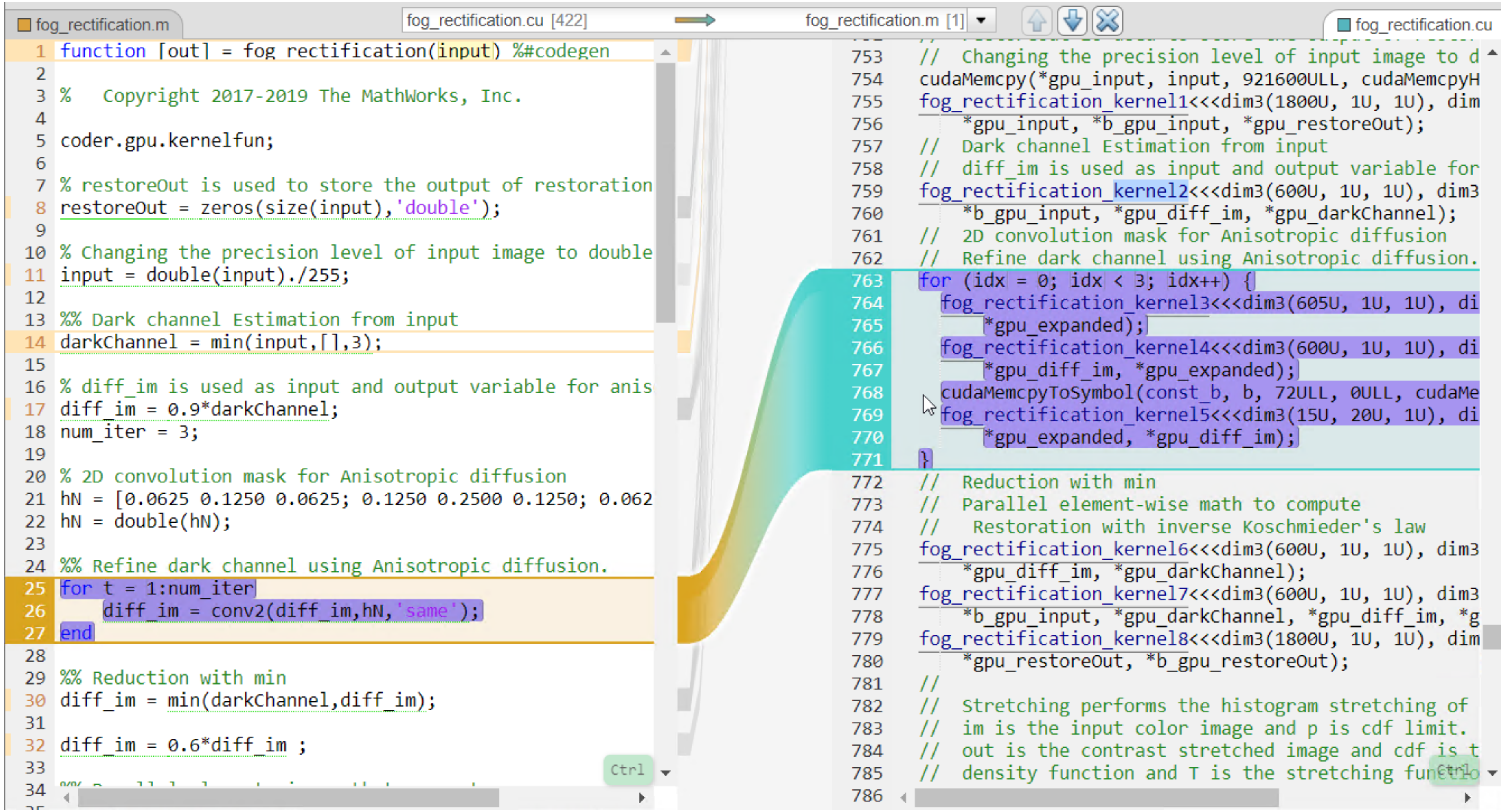


Event Statistics

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



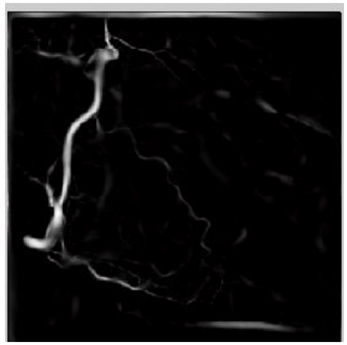
GPU Coder for Image Processing and Computer Vision

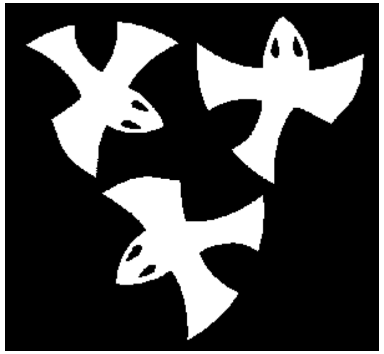
Fog removal

4x speedup

Frangi filter

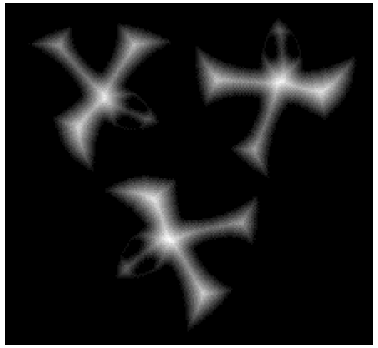
3x speedup





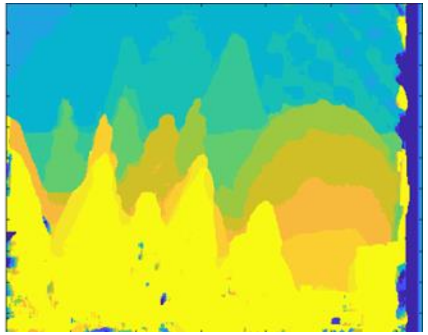
Distance transform

8x speedup



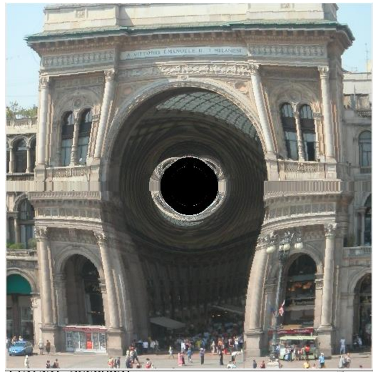
Stereo disparity

50x speedup



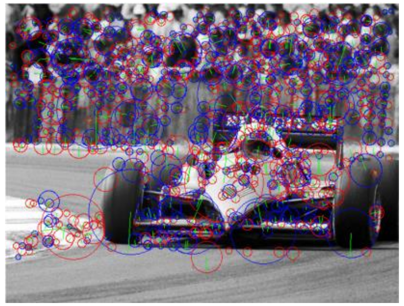
Ray tracing

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: Anomaly Detection

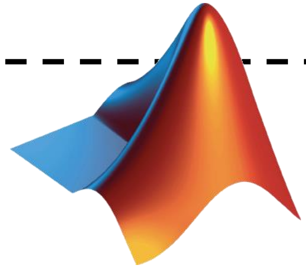
tb_workflow.m

Pretreatment 1 (Search for nut area) → Pretreatment 2 (Property analysis) → Pretreatment 3 (ROI extraction) → By CNN Pass / fail judgment → Post-processing 1 (**CAM generation**) → Post-processing 2 (Insert result)

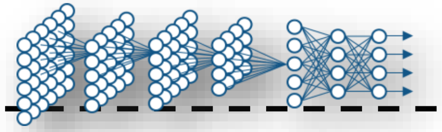
CUDA Optimized code (Pretreatment 1–3)

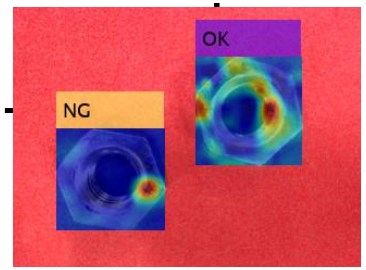
SqueezeNet (By CNN Pass / fail judgment)

CUDA Optimized code (Post-processing 1–2)



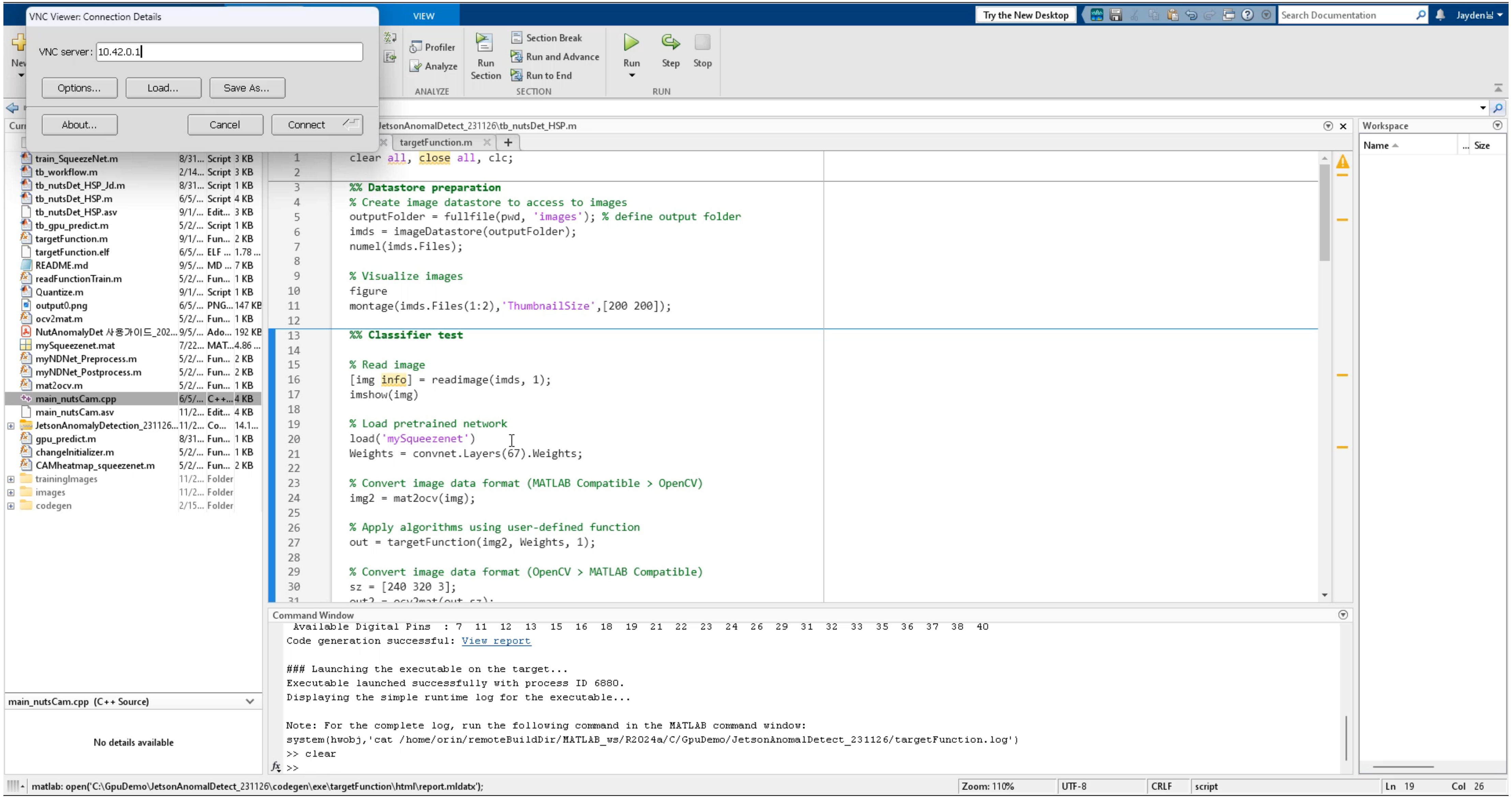
MATLAB





CUDA

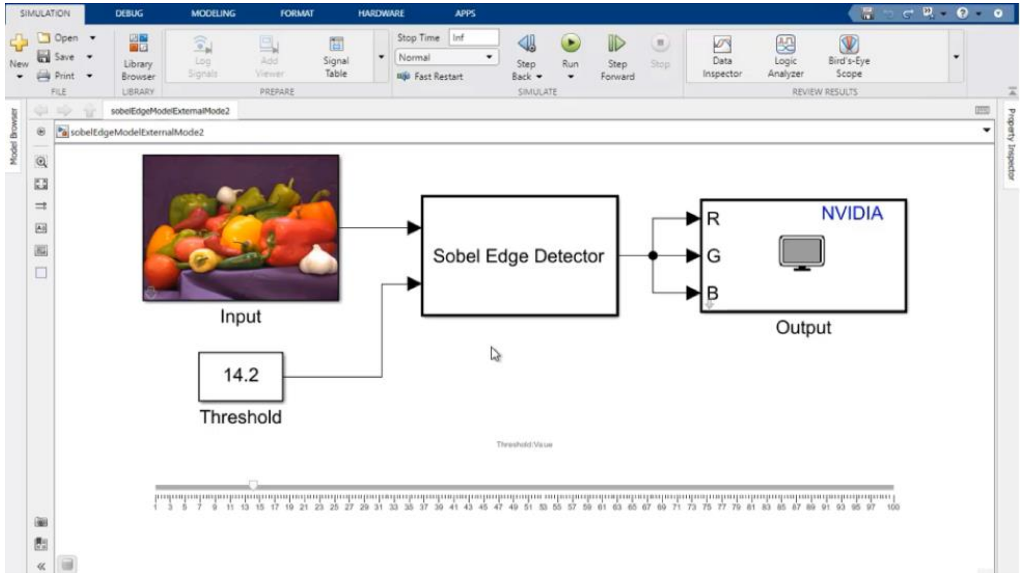
Example 2: Anomaly Detection



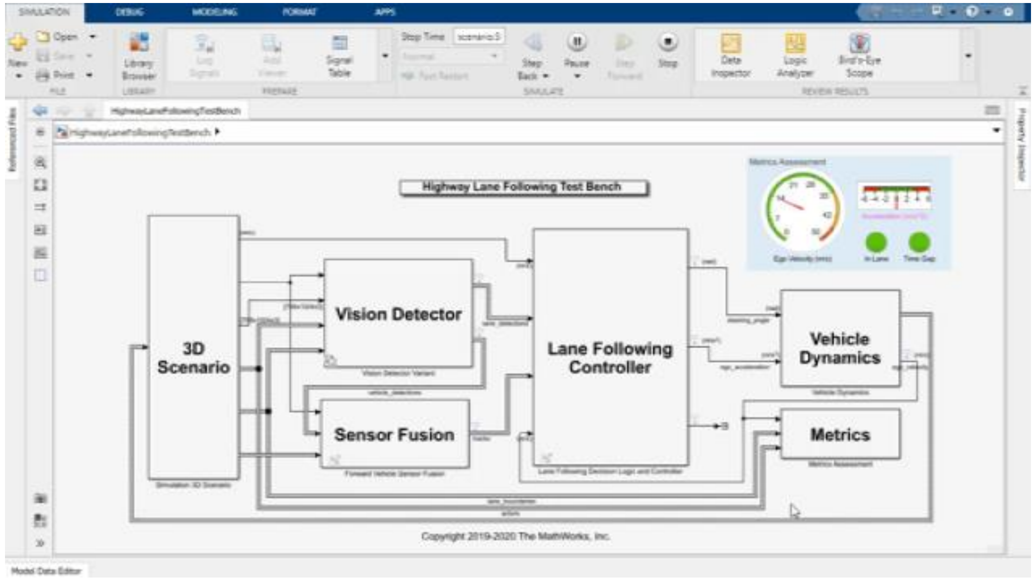
OK, but what about Simulink?

Two Application examples

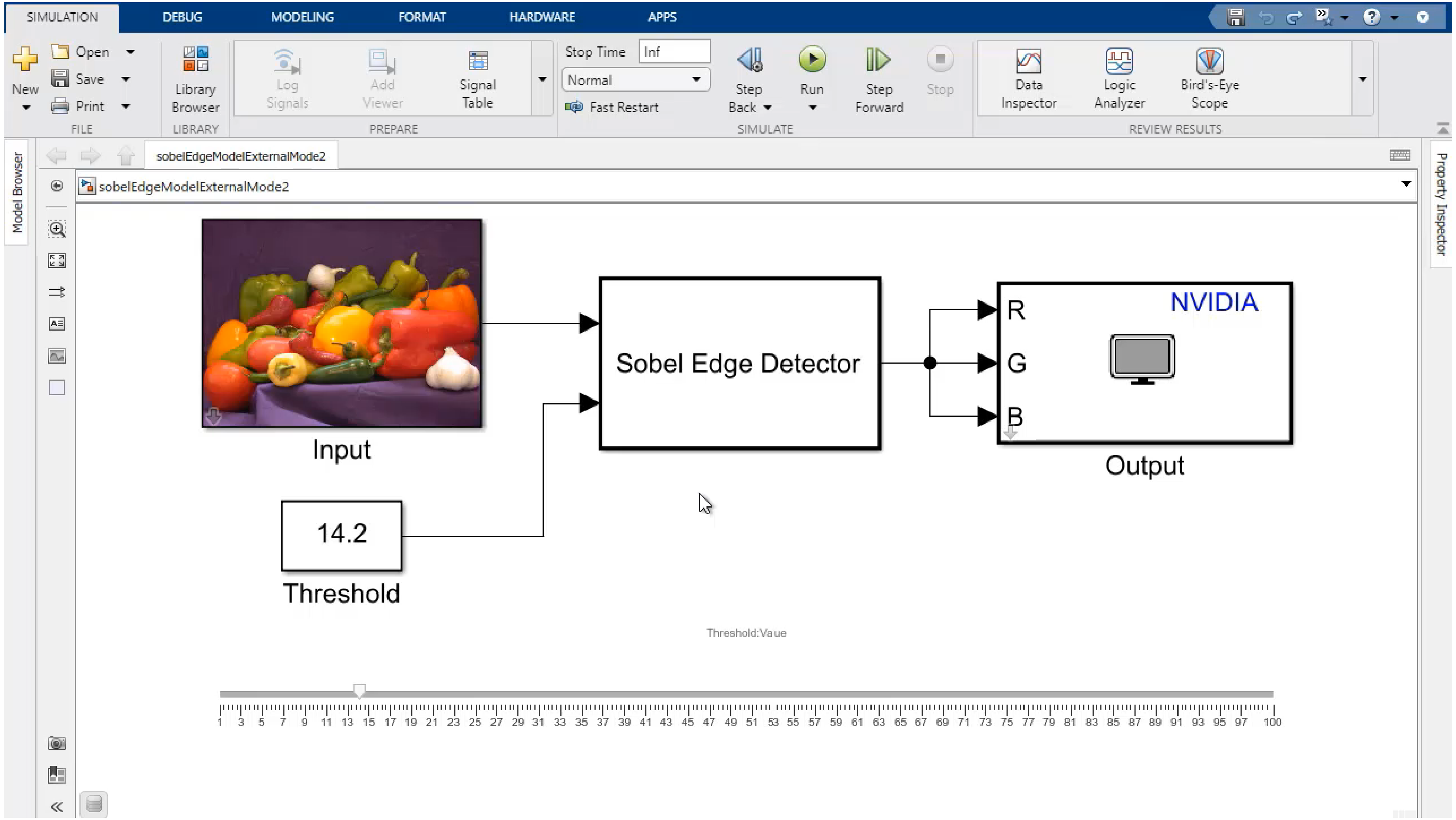
3) Sobel Edge Detection



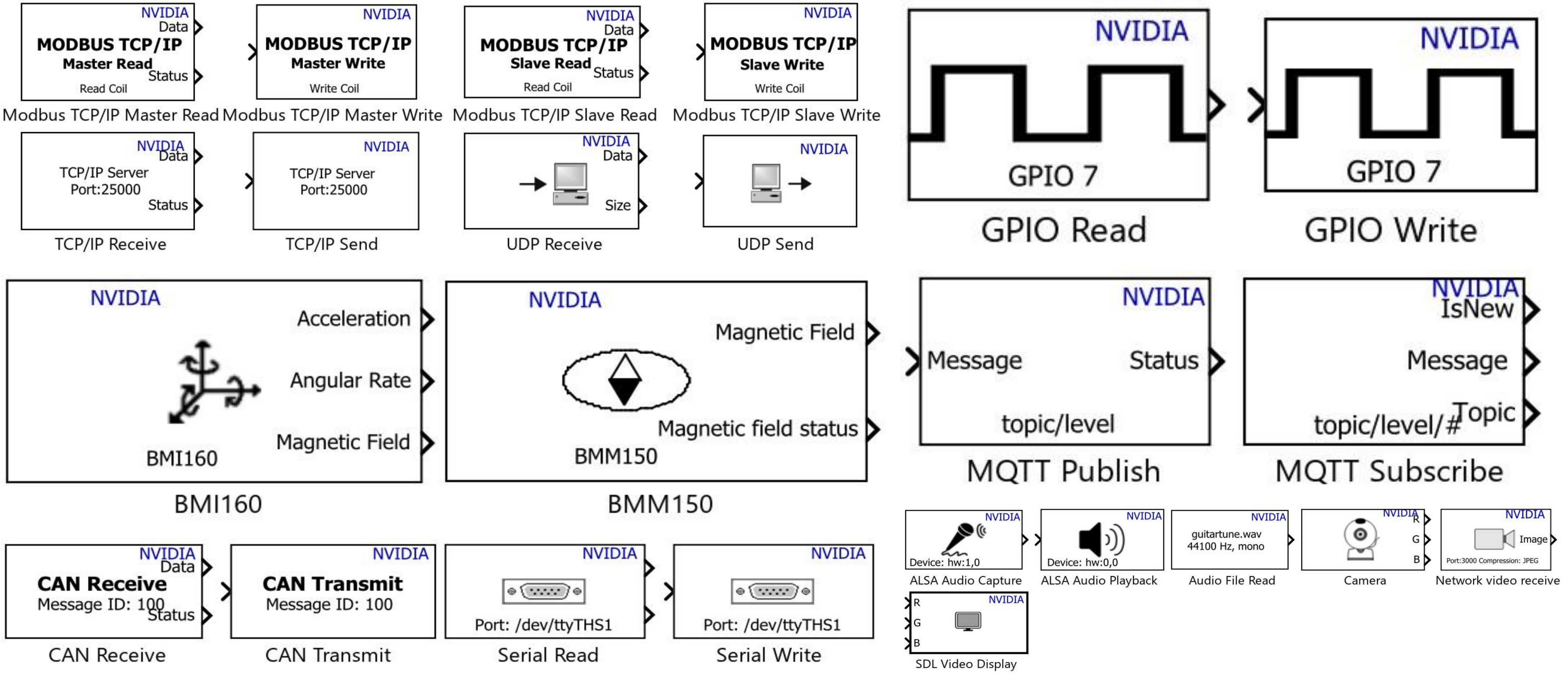
4) Highway Lane Following Model



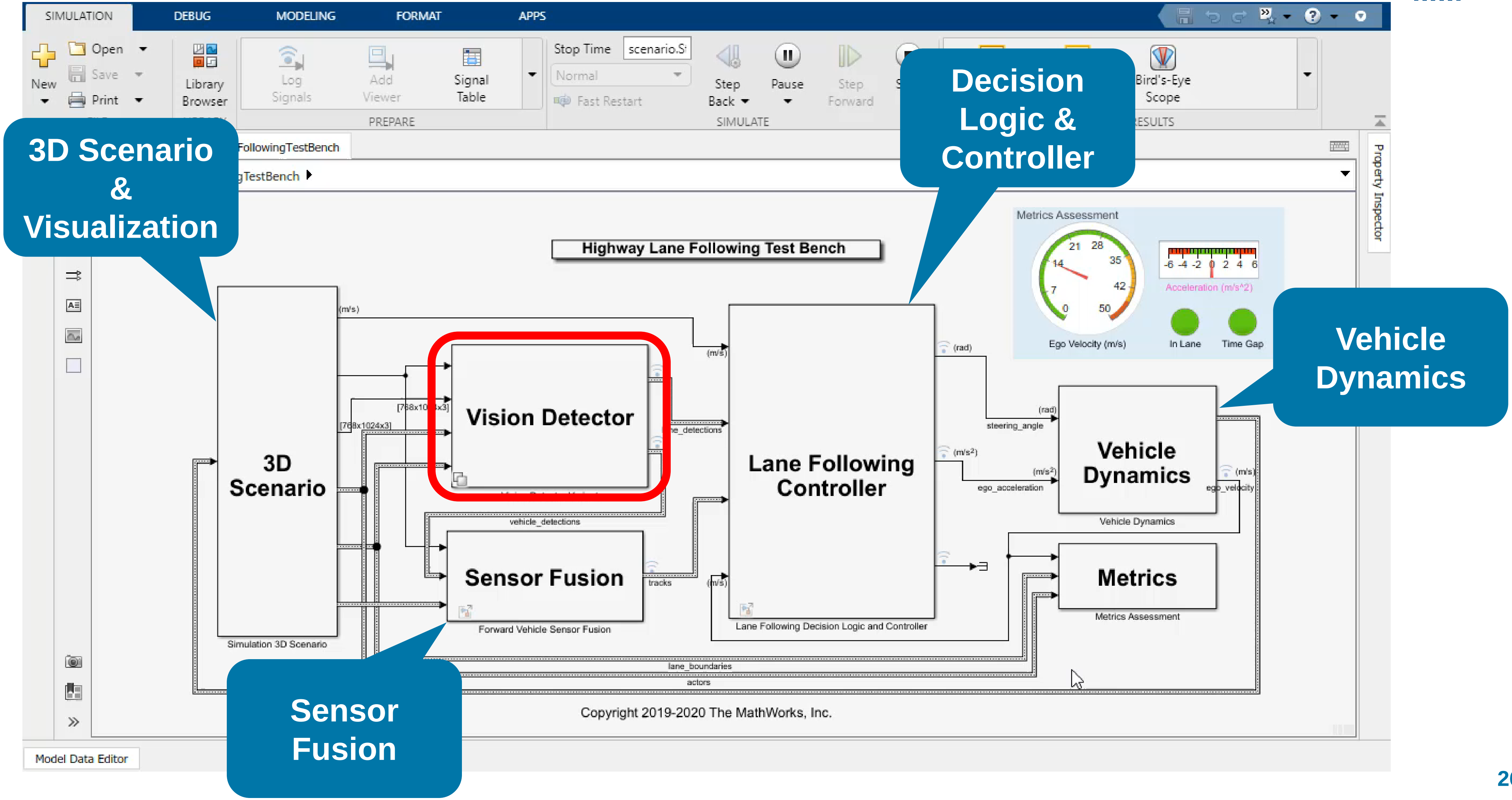
Example 3: Tuning Parameters using External Mode



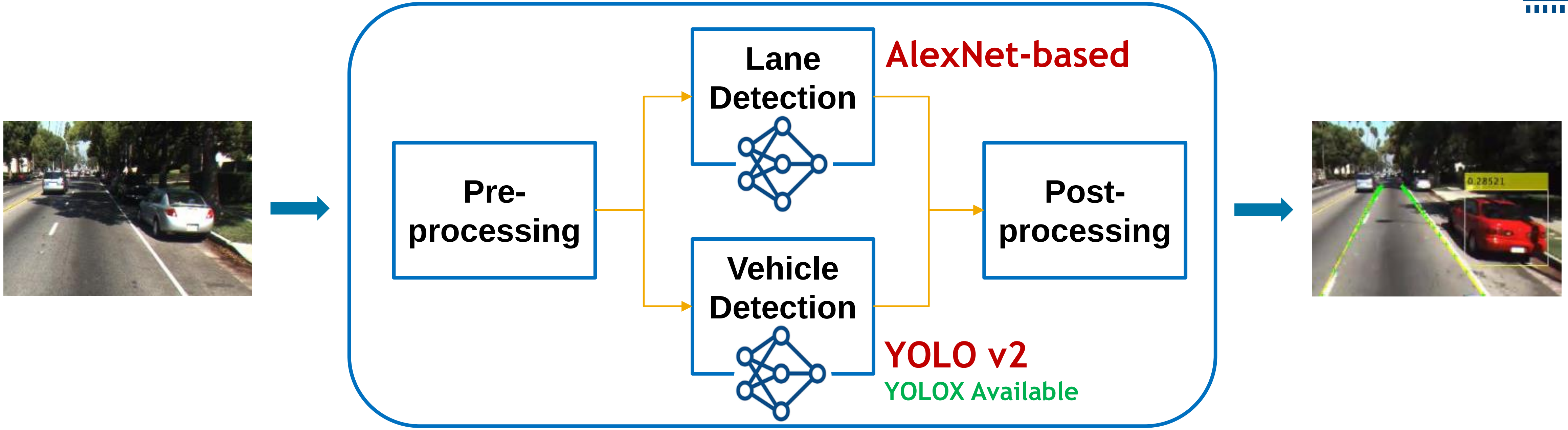
NVIDIA Peripheral Support – block library



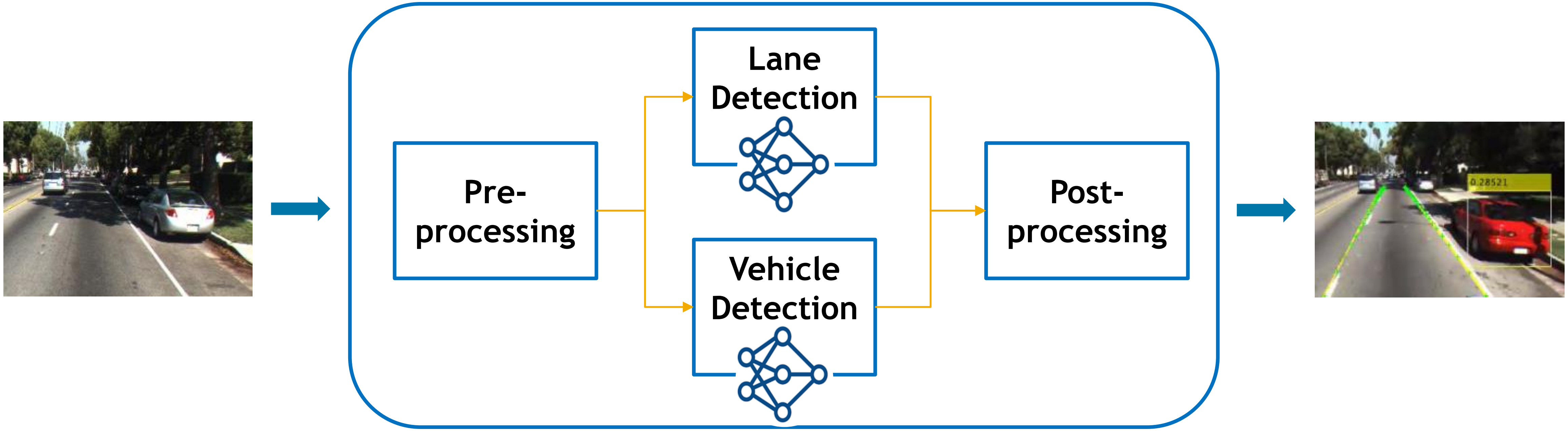
Example 4: Highway Lane Following Model

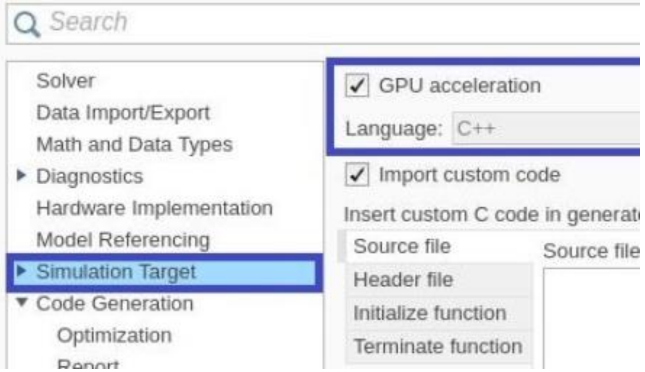


Lane and Vehicle Detection



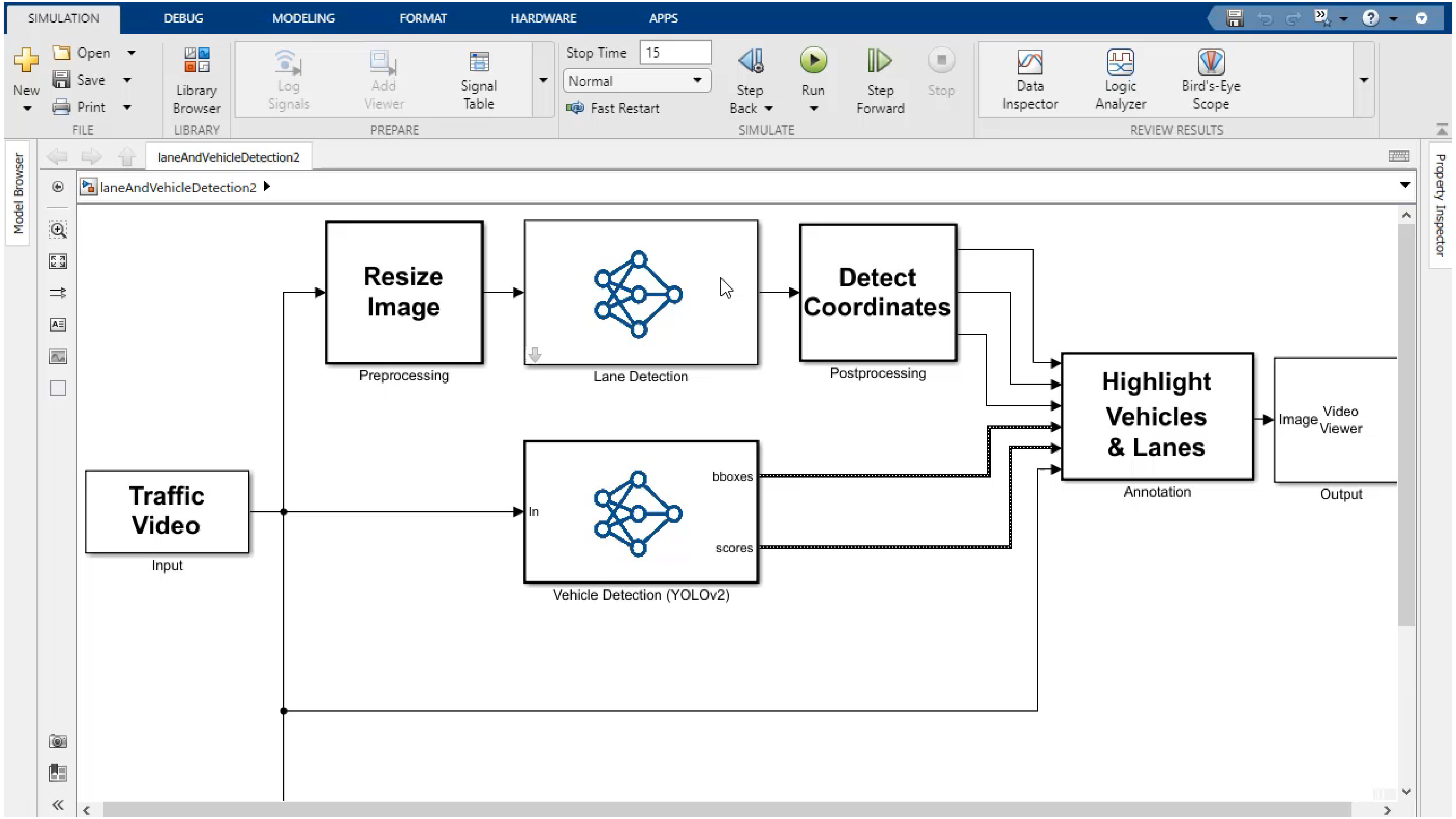
Run Simulation on Desktop GPU



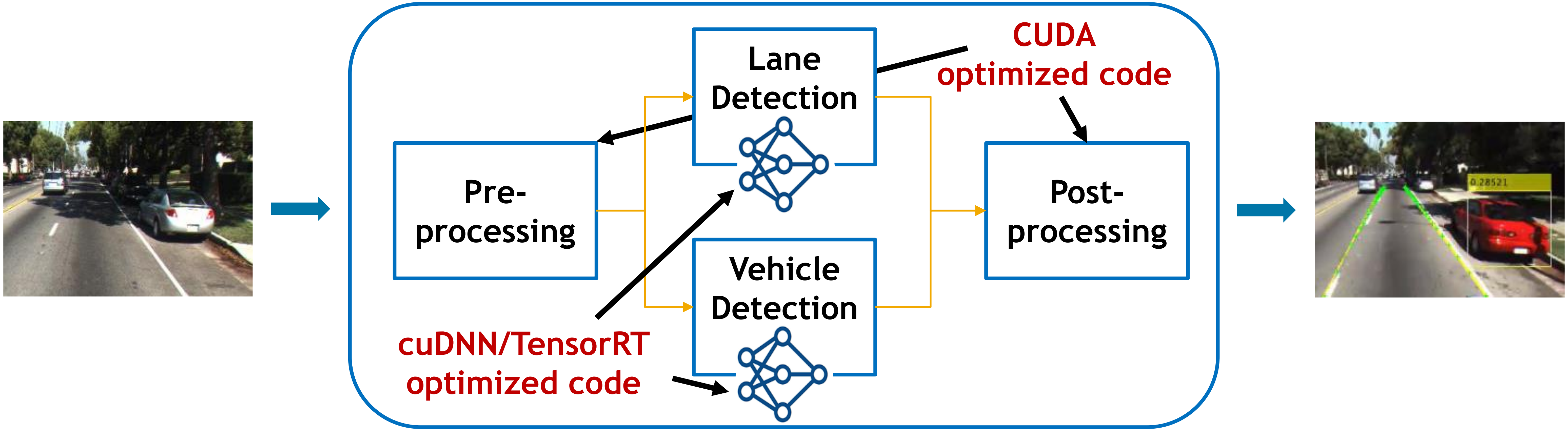


Speed up MATLAB Function blocks

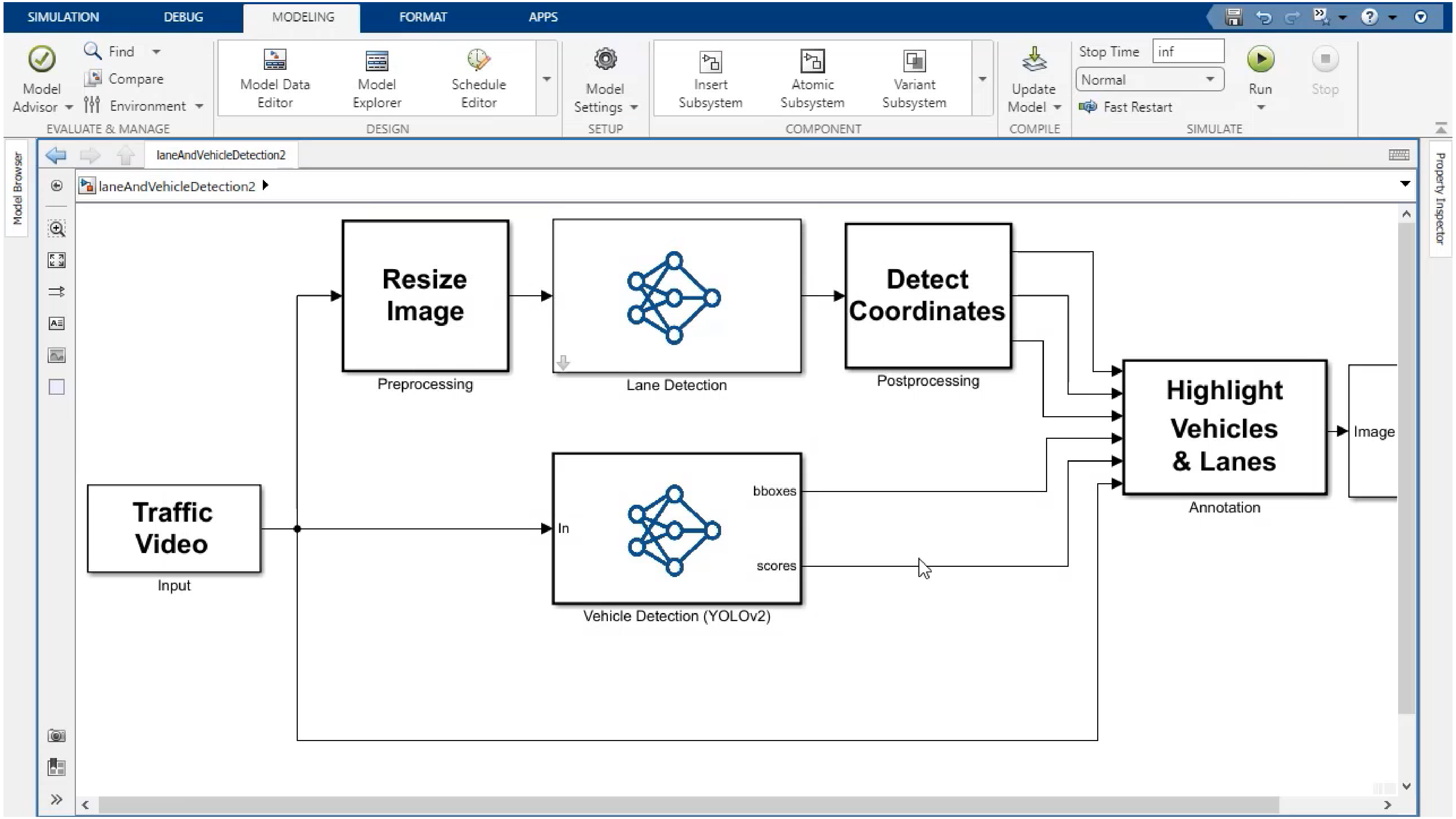
Run Simulation on Desktop GPU



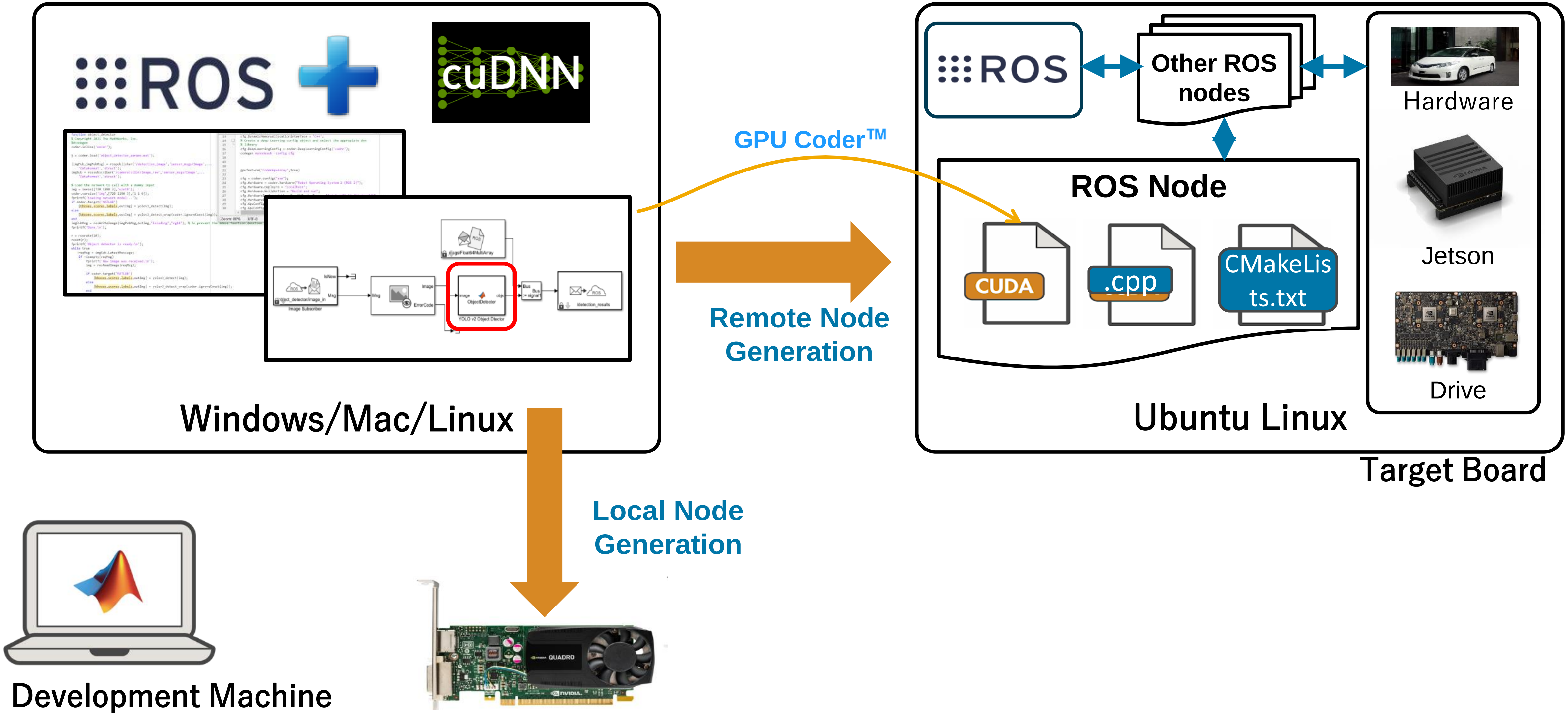
Generate CUDA Code and Run on Jetson



Generate CUDA Code and Run on Jetson

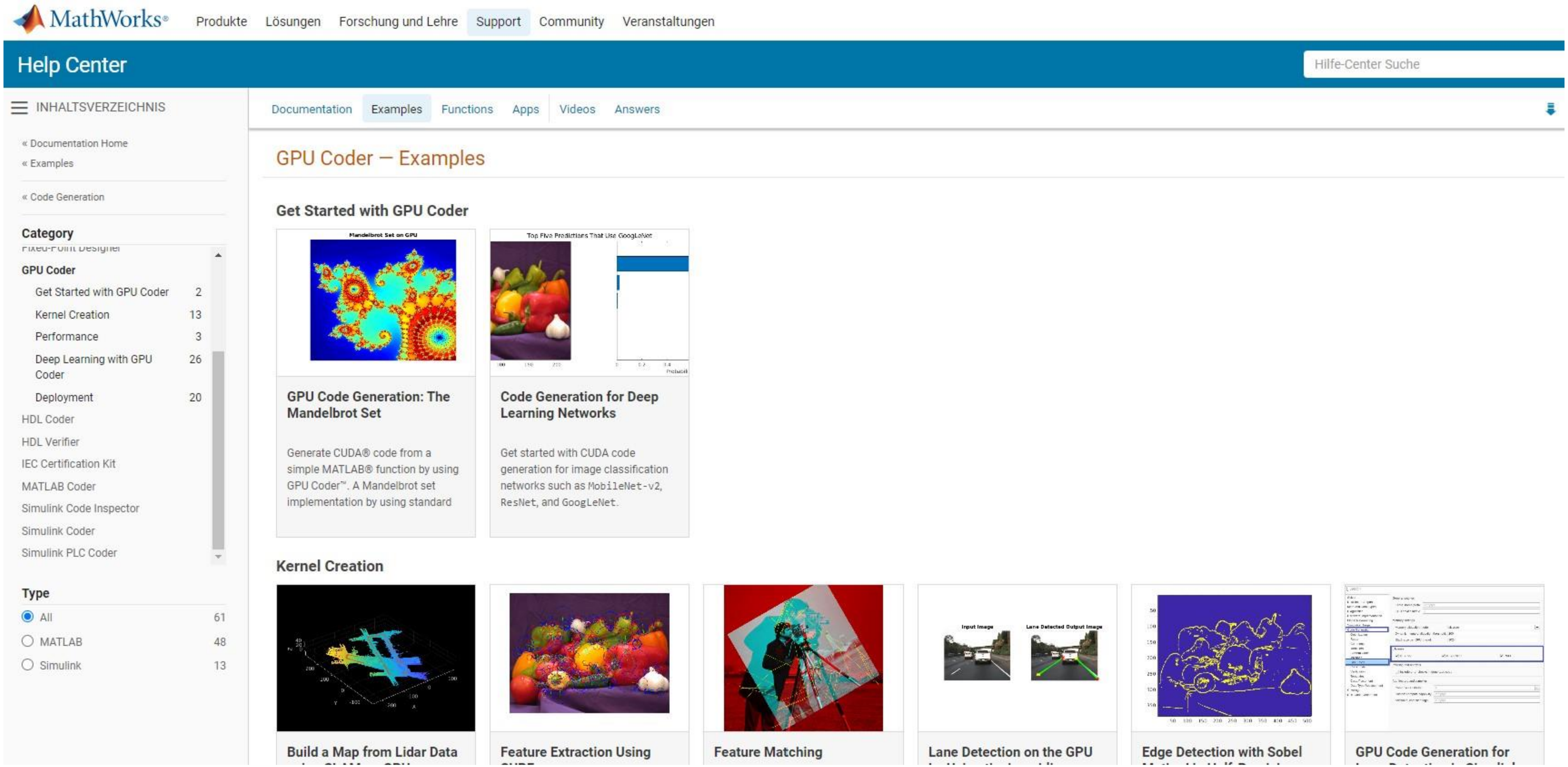


Generate CUDA Enabled ROS and ROS2 Node from MATLAB



Shipping Examples

https://www.mathworks.com/help/gpucoder/examples.html



Key Takeaways

GPU Coder generates CUDA code from MATLAB & Simulink

Accelerate MATLAB & Simulink simulations

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

DEMO Booth

MATLAB EXPO

MathWorks®