

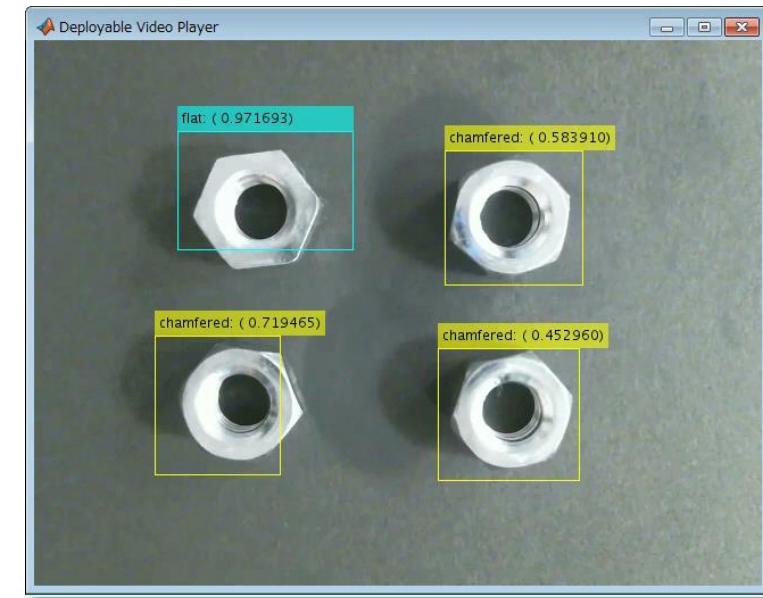
いまからはじめる組み込みGPU実装

～コンピュータービジョン・ディープラーニング編～

MathWorks Japan
アプリケーションエンジニアリング部
シニアアプリケーションエンジニア
大塚 慶太郎



コンピュータービジョン・ディープラーニングによる、様々な可能性



自動運転



ロボティクス



予知保全
(製造設備)



セキュリティ



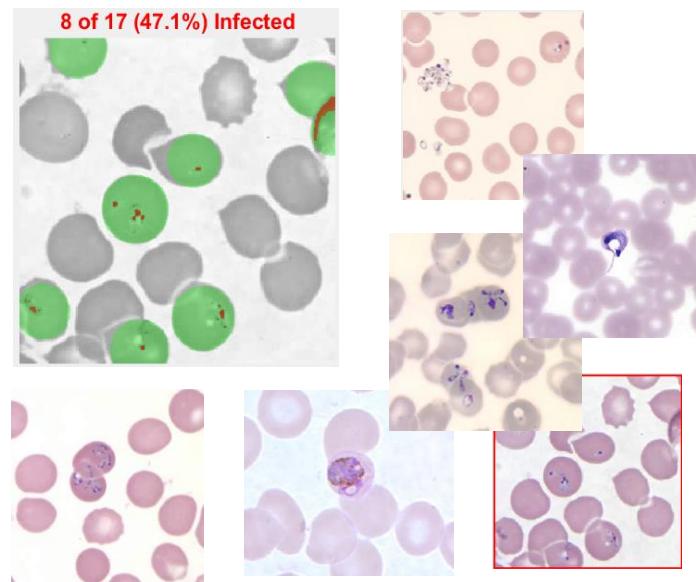
転移学習を使った画像分類

Deep Learning for Image Classification

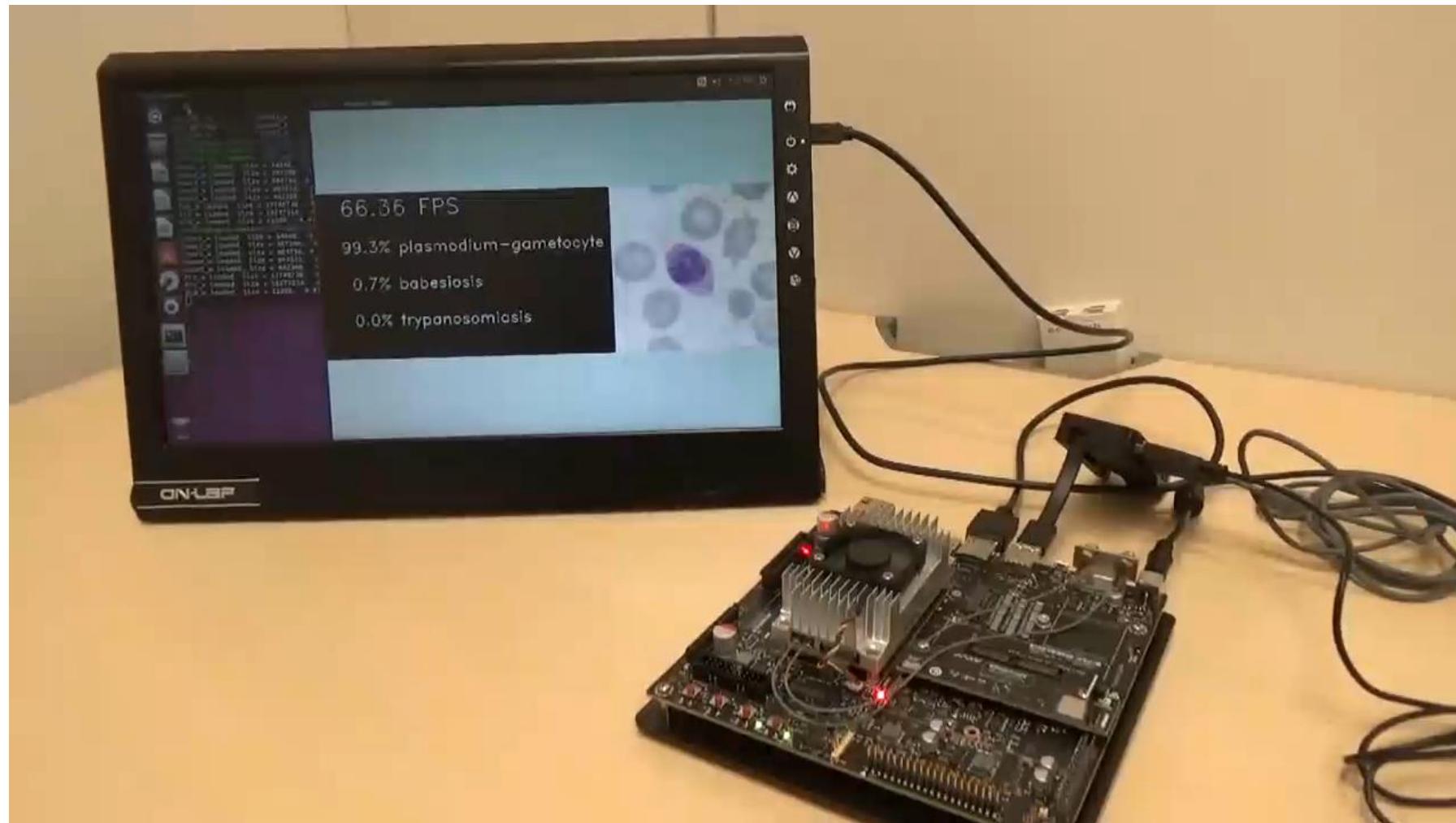


Demo : 血液検査画像の分類

- 3つの寄生感染症を分類
 - バベシア
 - マラリア原虫
 - トリパノソーマ
- 従来手法(局所特徴+SVM)では~70%程度の分類精度

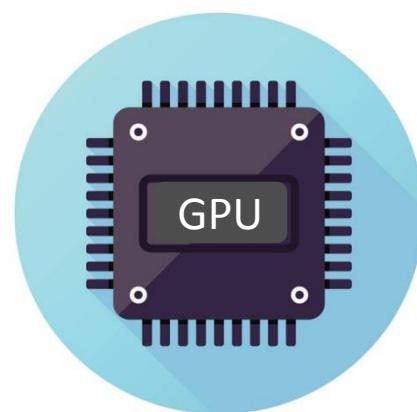


MATLAB®から組み込みGPUへの実装：血液検査画像の分類



ディープラーニング 実装ソリューション

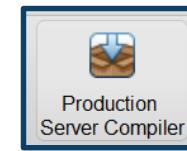
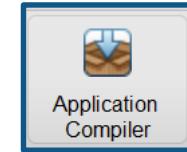
“組み込みGPU
への実装”



実装/配布

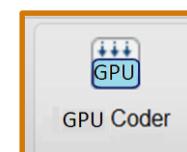
学習済みモデルのシェア

機器・デバイスへの実装



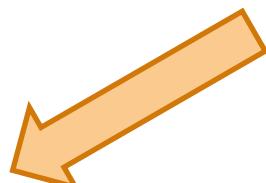
- デスクトップアプリケーション

- Web/エンタープライズ
アプリケーション



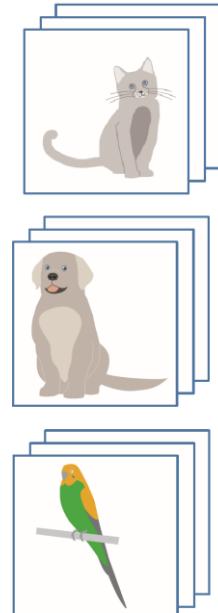
- GPUへの実装

GPU Coder™

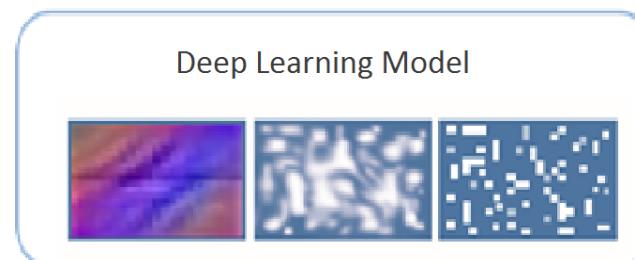


コンピュータービジョン向けディープラーニング ワークフロー

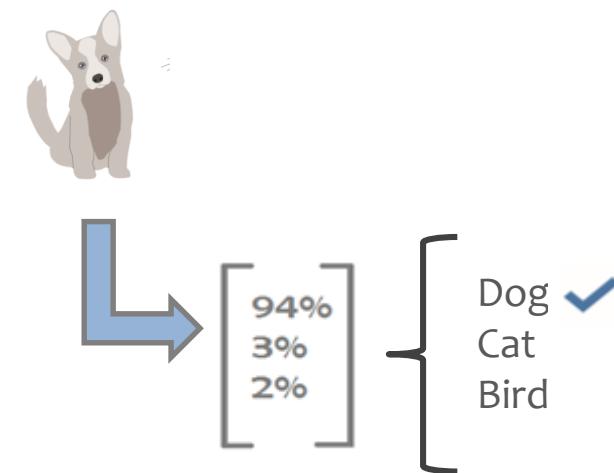
“膨大なデータの
取り扱い”



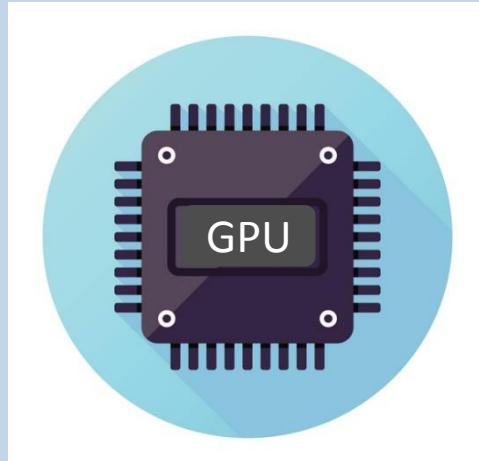
“学習済み
ネットワークの取り込み”



“マルチGPU、クラスタ環境を
使った効率的な学習”



“組み込みGPU
への実装”



データアクセス

モデル

学習

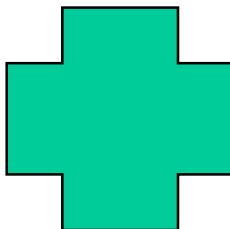
実装/配布

Agenda

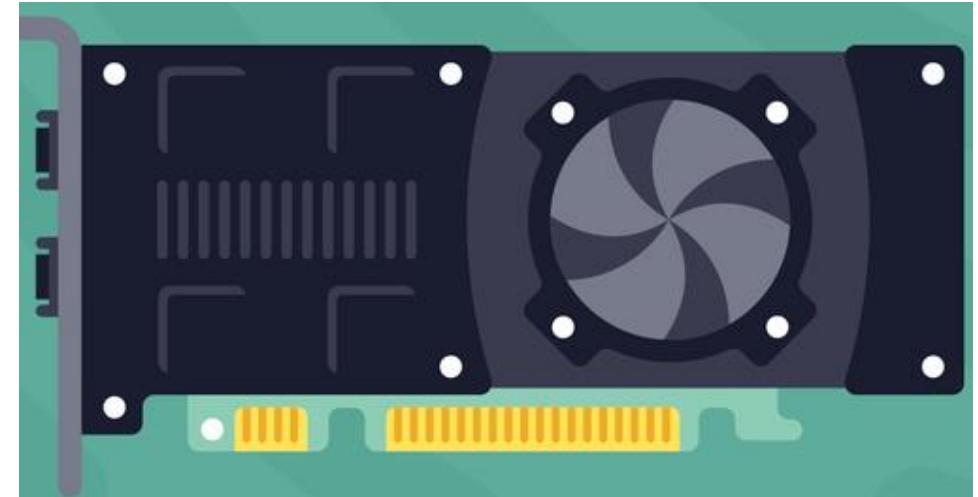
- Introduction
- GPU・CUDA Cについて
- GPU Coder™による効率的なGPU実装
- まとめ

GPUコンピューティングについて

CPU



GPUアクセラレータ



周波数 : ~4GHz

コア数 : ~24

シーケンシャルな処理が得意

周波数 : ~1.5GHz

コア数 : ~6000

並列処理が得意

GPUを動かす為には…
非常に優れたデバイスですが、ハードルも

- 例：ベクタ信号の総和計算

MATLAB

```
function s = vecSum(v)
    s = 0;
    for i = 1:length(v)
        s = s + v(i);
    end
end
```



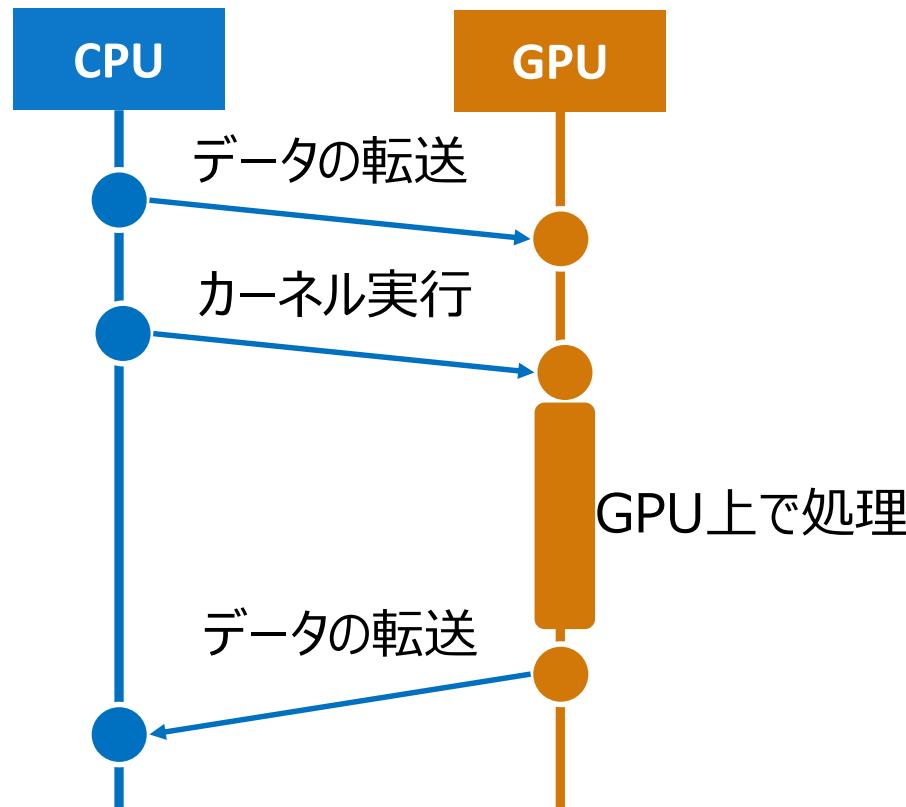
CUDA C

```
static __global__ __launch_bounds__(512, 1) void kernel1(const real_T *v, real_T
{
    uint32_T idx;
    real_T tmpRed;
    int32_T threadIdx;
    threadIdx = (int32_T)(blockDim.x * blockIdx.x + threadIdx.x);
    if (!(threadIdx >= 512)) {
        tmpRed = 0.0;
        for (idx = blockIdx.x * blockDim.x + threadIdx.x; blockDim.x * gridDim.x <
0U ? idx >= 511U : idx <= 511U; idx += blockDim.x * gridDim.x) {
            tmpRed += v[threadIdx];
        }
        tmpRed = workGroupReduction1(tmpRed, 0.0);
        if (threadIdx.x == 0U) {
            atomicOp1(s, tmpRed);
        }
    }
}
static __inline__
{
    unsigned long
    unsigned long
```



- CUDA Cプログラミングスキル
- ハードウェアを意識した効率の追求が必要
- 基のアルゴリズムとの等価性をいかに確保するか

GPUでHello World実行



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

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

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

・カーネルの呼び出し(特殊な構文)

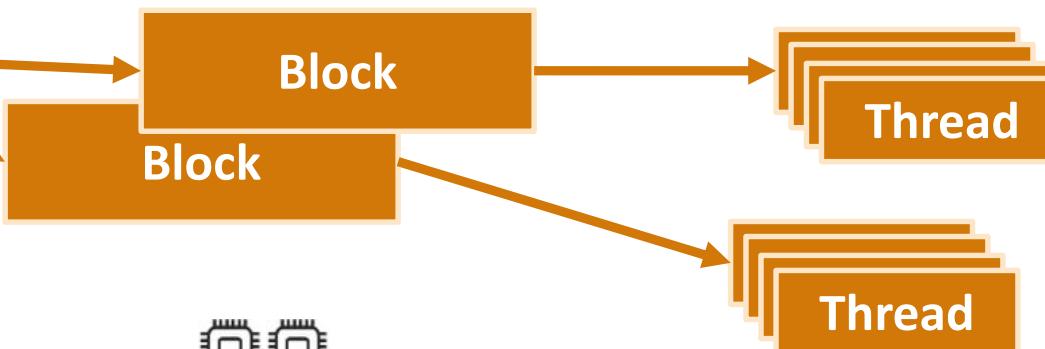
```
kernelFunc<<<Grid_dim, Block_dim>>>(a, b, c);
```

カーネル実行の階層について

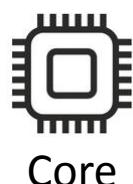
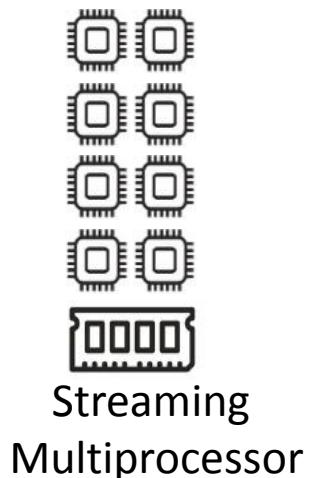
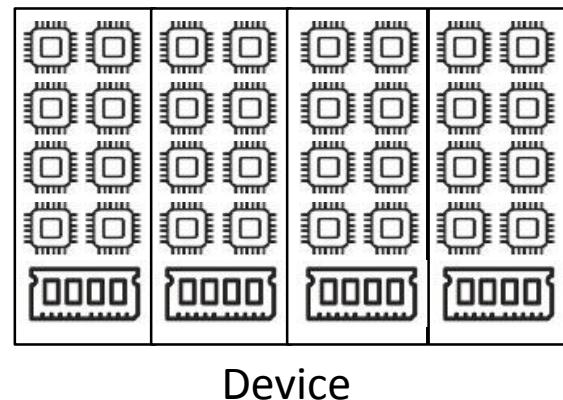
Grid :
ブロックをまとめたもの
ホスト側からの呼び出し単位



Block :
スレッドをまとめたもの
1つのブロックに格納できるスレッド数には
上限がある



Thread :
カーネルを動作させた時
多数のプログラムの最小単位



カーネルを実装する：配列A, 配列Bの和を求めるプログラム

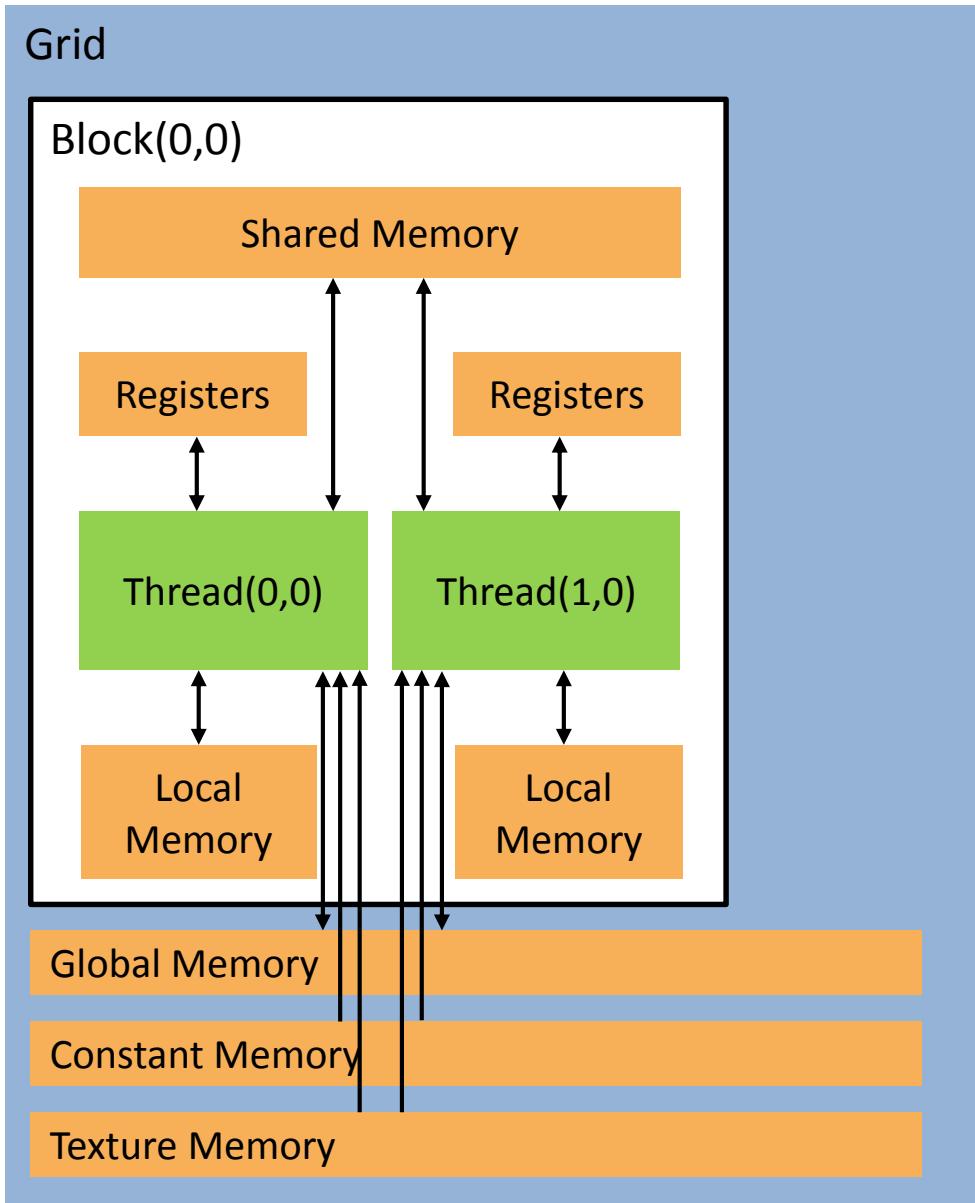
ホスト上で実行

```
void sumArraysOnHost(float *A, float *B, float *C, const int N)
{
    for (int idx = 0; idx < N; idx++)
        C[idx] = A[idx] + B[idx];
}
```

デバイス上で実行

```
__global__ void sumArraysOnGPU(float *A, float *B, float *C, const int N)
{
    int i = blockDim.x* blockIdx.x+ threadIdx.x;
    C[i] = A[i] + B[i];
}
```

メモリの特徴を踏まえたプログラミングの重要性



- **device**
 - グローバルメモリ領域を確保、ホスト側から読み書き可能。
- **constant**
 - コンスタントメモリ領域を確保。
- **shared**
 - シェアードメモリ領域を確保。

```
_constant_ int Coef = 3;
```

```
cudaMemcpyToSymbol(Coef, &host_Coef, sizeof(int));
```

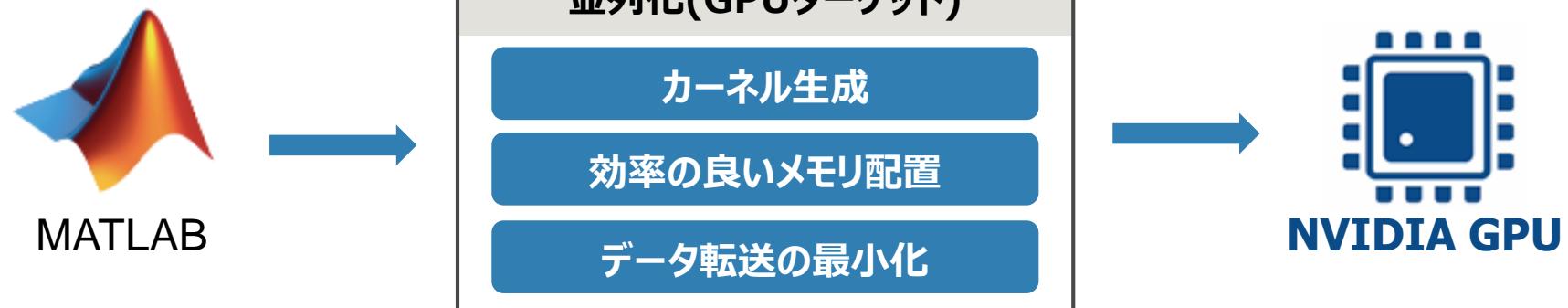
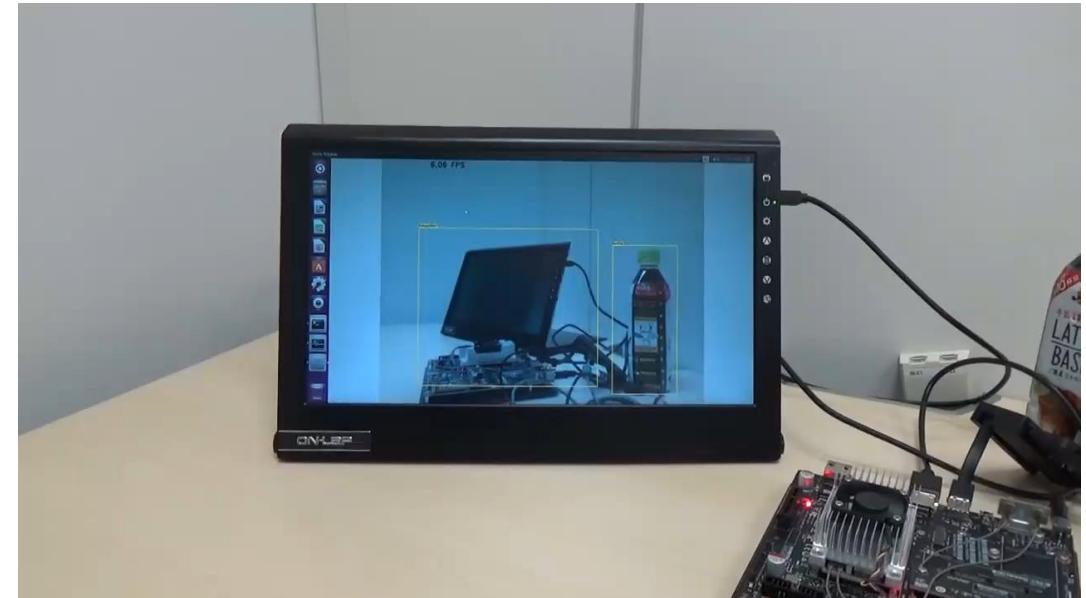
Agenda

- Introduction
- GPU・CUDA Cについて
- GPU Coder™による効率的なGPU実装
- まとめ

GPU Coder™

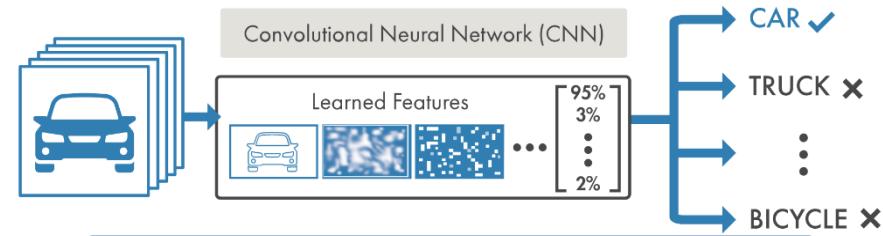
New in **R2017b** !!

- プラグマによる関数解析とカーネル生成
 - CUDAの文法を知らなくても利用できる
- 専用デザインパターンの利用も可能
 - より確実かつ効率の良いカーネル生成
- GPU Coder専用GUIを使ったコード生成
 - 初めてでも使いやすいGUI
- Simulinkへの統合
 - 生成したDLLを呼び出すAPIを作成可能

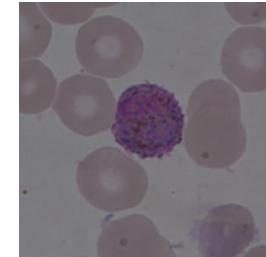


MATLABコードからCUDA Cを生成します

コンピュータービジョン・ディープラーニングの組み込み実装が可能に!



前処理
(コントラスト調整等)



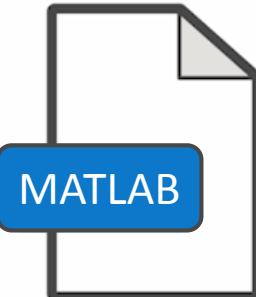
学習済み
ネットワーク



後処理
(ROI抽出等)



画像前処理・後処理+ディープラーニングで
コード生成可能!



MATLAB

前処理が必要になるケース

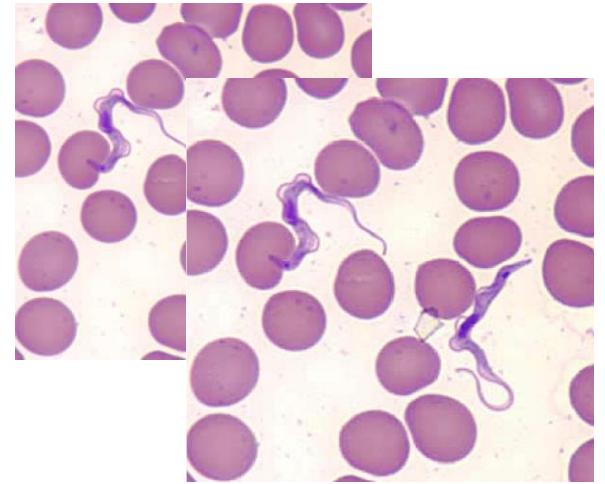


入力画像

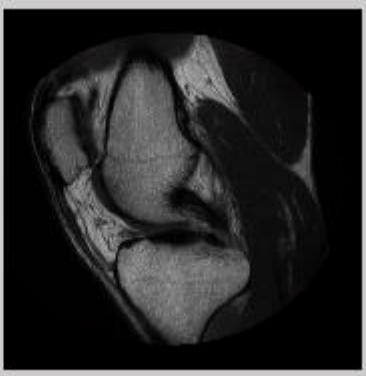
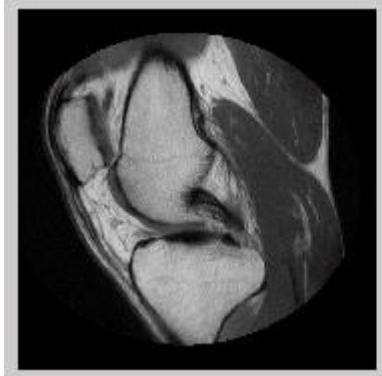


前処理後

細かいテクスチャの除去



輪郭強調



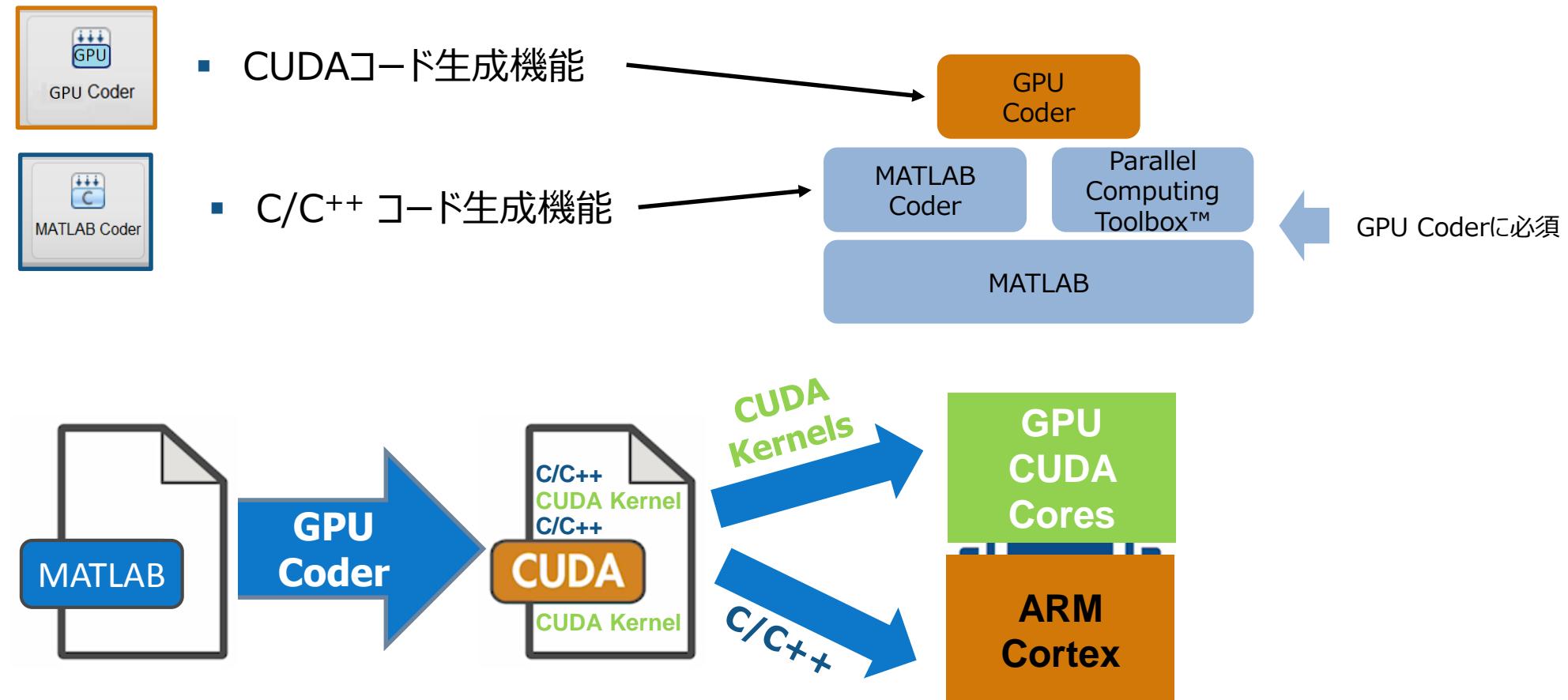
コントラスト調整



霧除去

ホスト・カーネル両方のプログラムを生成可能

MATLAB Coder™のコード生成機能を利用



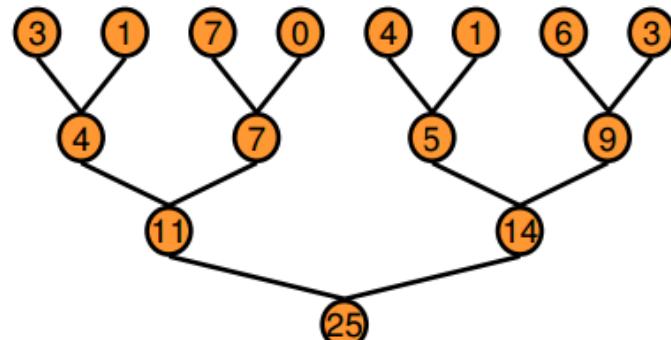
GPU Coder使用例

```
function s = vecSum(v)
    s = 0;
coder.gpu.kernelfun();
    for i = 1:length(v)
        s = s + v(i);
    end
end
```

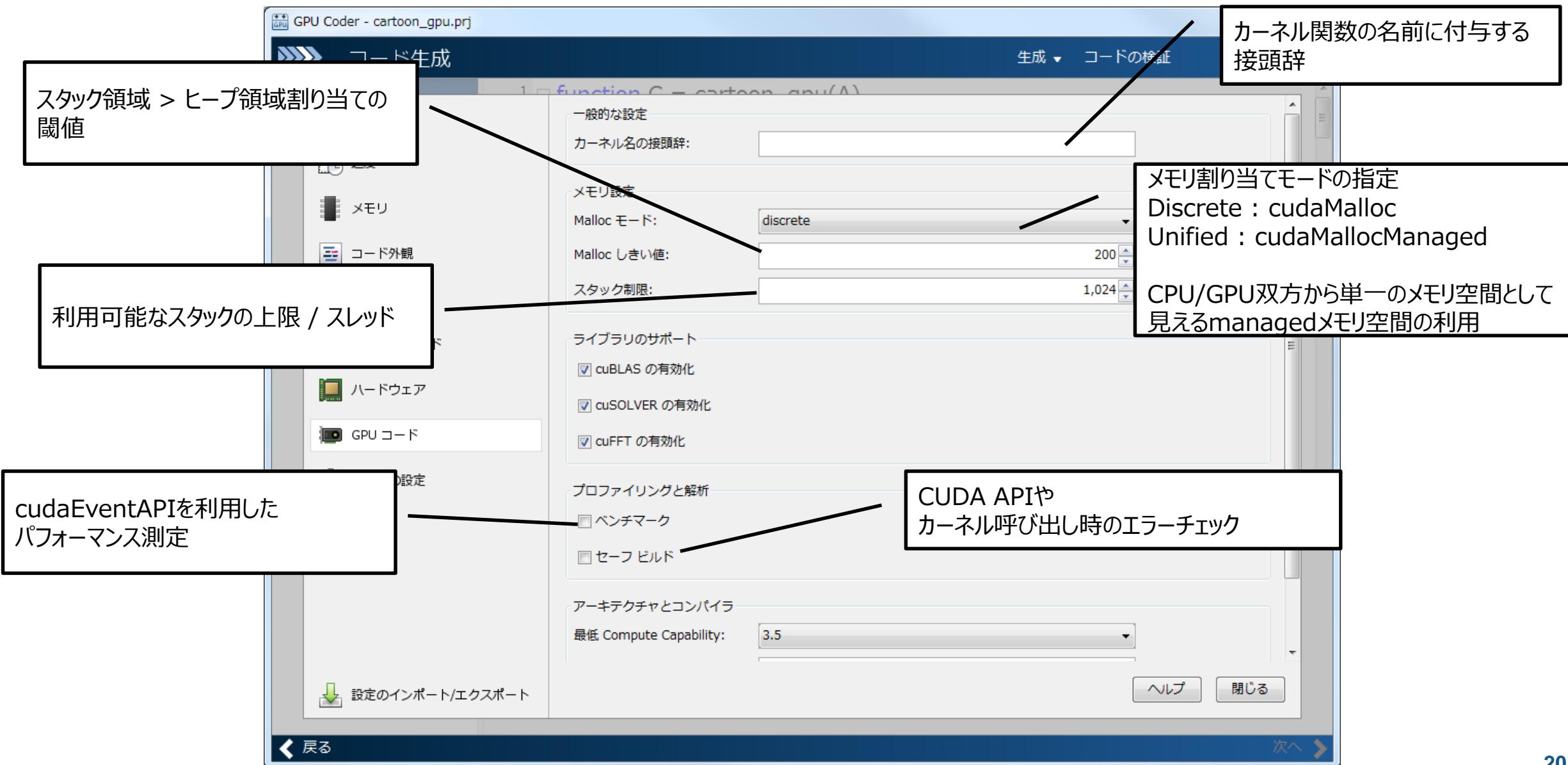
以下の関数でもカーネル生成が行われます

```
function s = vecSum(v)
    coder.gpu.kernelfun();
    s = sum(v);
end
```

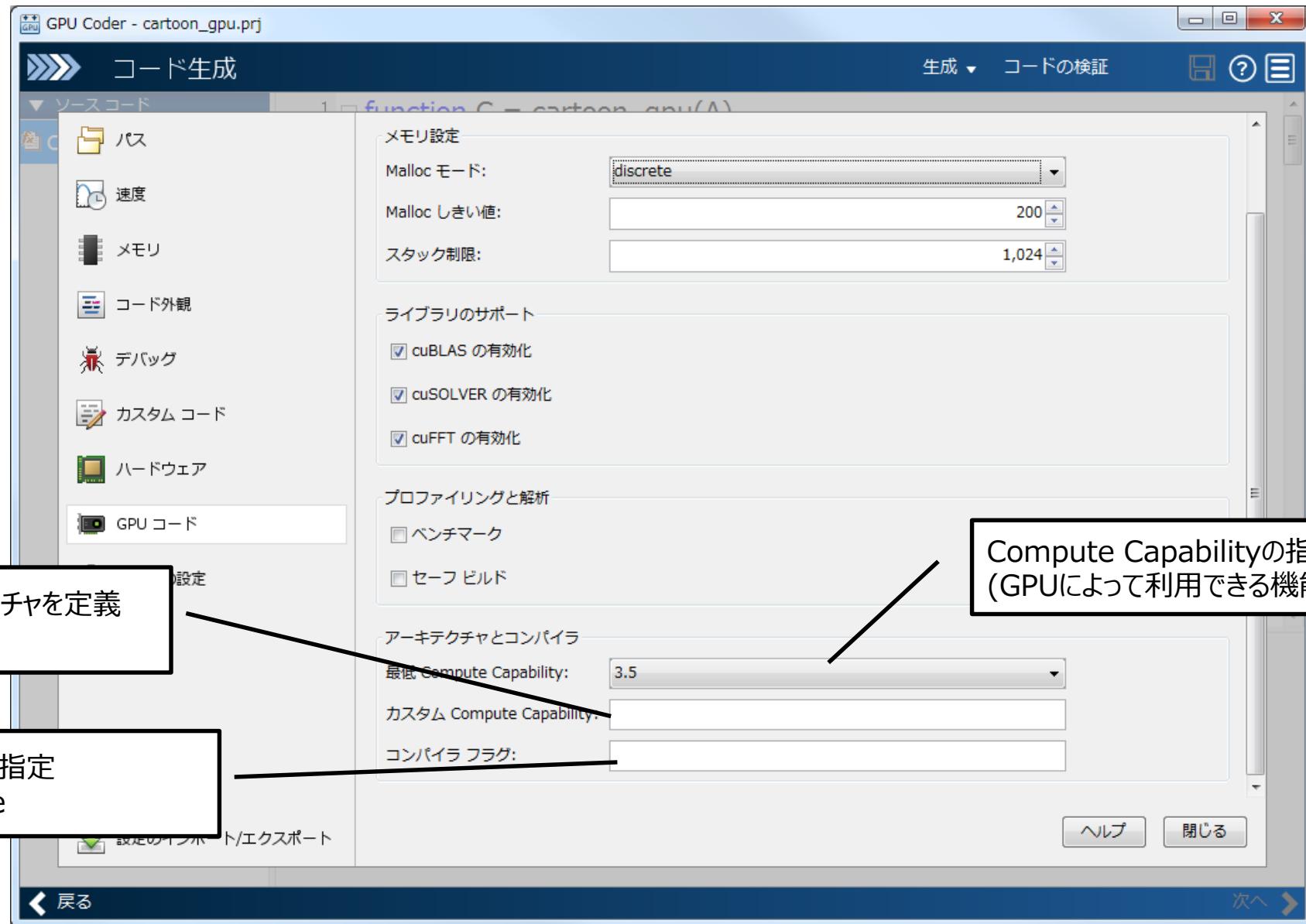
GPU Coderがコードを解析し、並列化できる部分を特定



GPU Coderで指定できるコード生成オプション(1/2)



GPU Coderで指定できるコード生成オプション(2/2)



GPU Coderで利用できるプログラマ(1/2)

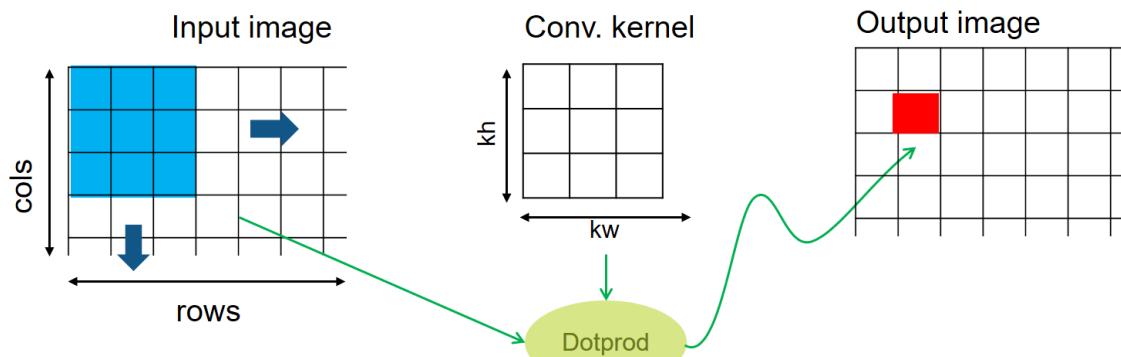
- `coder.gpu.kernelfun`
 - 最も利用頻度が高いプログラマ。関数を解析しカーネル生成
- `coder.gpu.kernel`
 - 指定したForループに対してカーネル生成
- `coder.gpu.constantMemory`
 - 指定した変数に対して、コンスタントメモリ領域を確保

GPU Coderで利用できるプログラマ(2/2)

- `gpucoder.stencilKernel`
 - ステンシル計算専用プログラマ

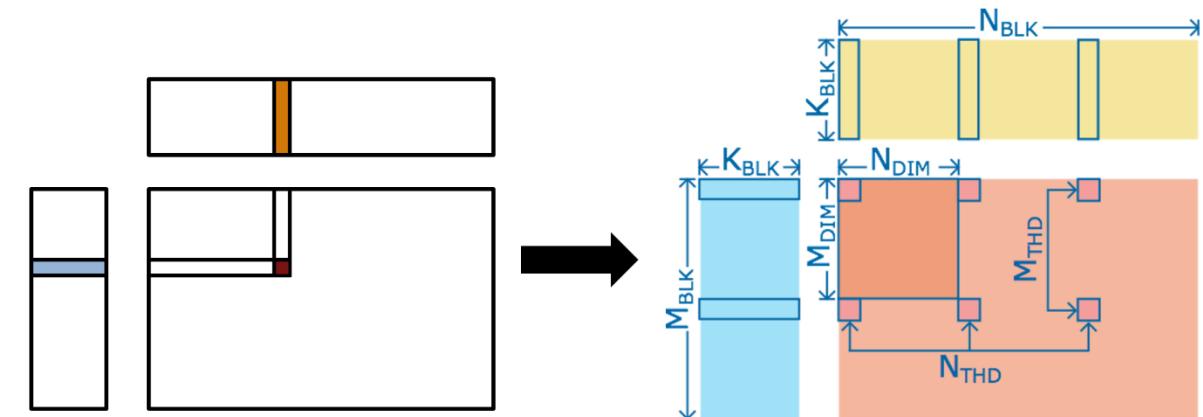
```
function B = meanImgFilt(A) %#codegen
    B = gpucoder.stencilKernel(@my_mean,A,[3 3],'same');

    function out = my_mean(A)
        out = cast(mean(A(:)), class(A));
    end
end
```



- `gpucoder.matrixMatrixkernel`
 - 行列-行列 計算専用プログラマ

```
function scores = matMul_nn(f1, f2)
    scores = gpucoder.matrixMatrixKernel(@times, f1, f2, 'nn');
end
```

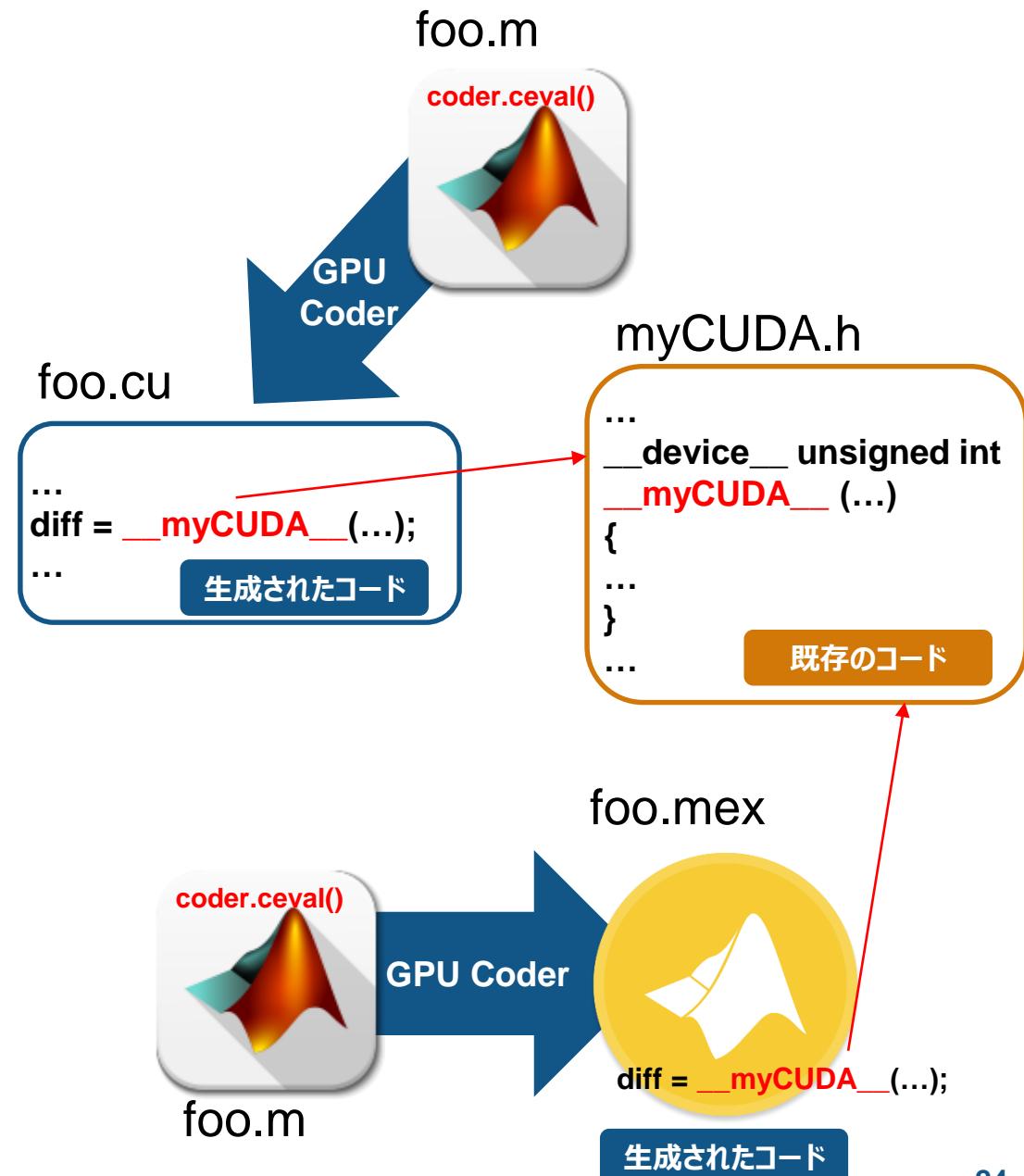


既存のCUDA資産の統合

- [coder.ceval](#) で外部関数を宣言できます

2つのワークフロー：

- **コード生成:**
既存のCUDAコードを、GPU Coderを使って生成されるコードに含めることができます
- **MATLAB上でのシミュレーション:**
既存CUDAコードを予めMEX化し、MATLAB上で利用できます



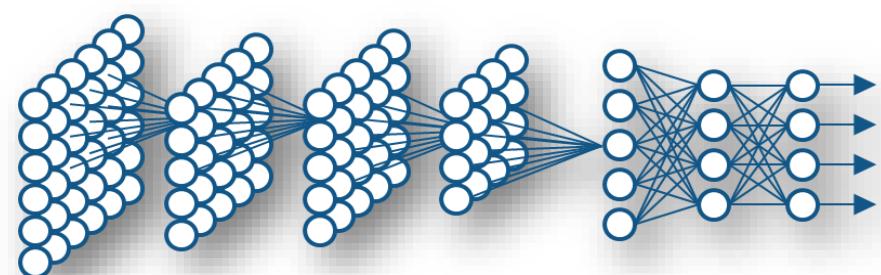
GPU Coder 使用例：ディープラーニング

- ディープラーニング(推論部分)

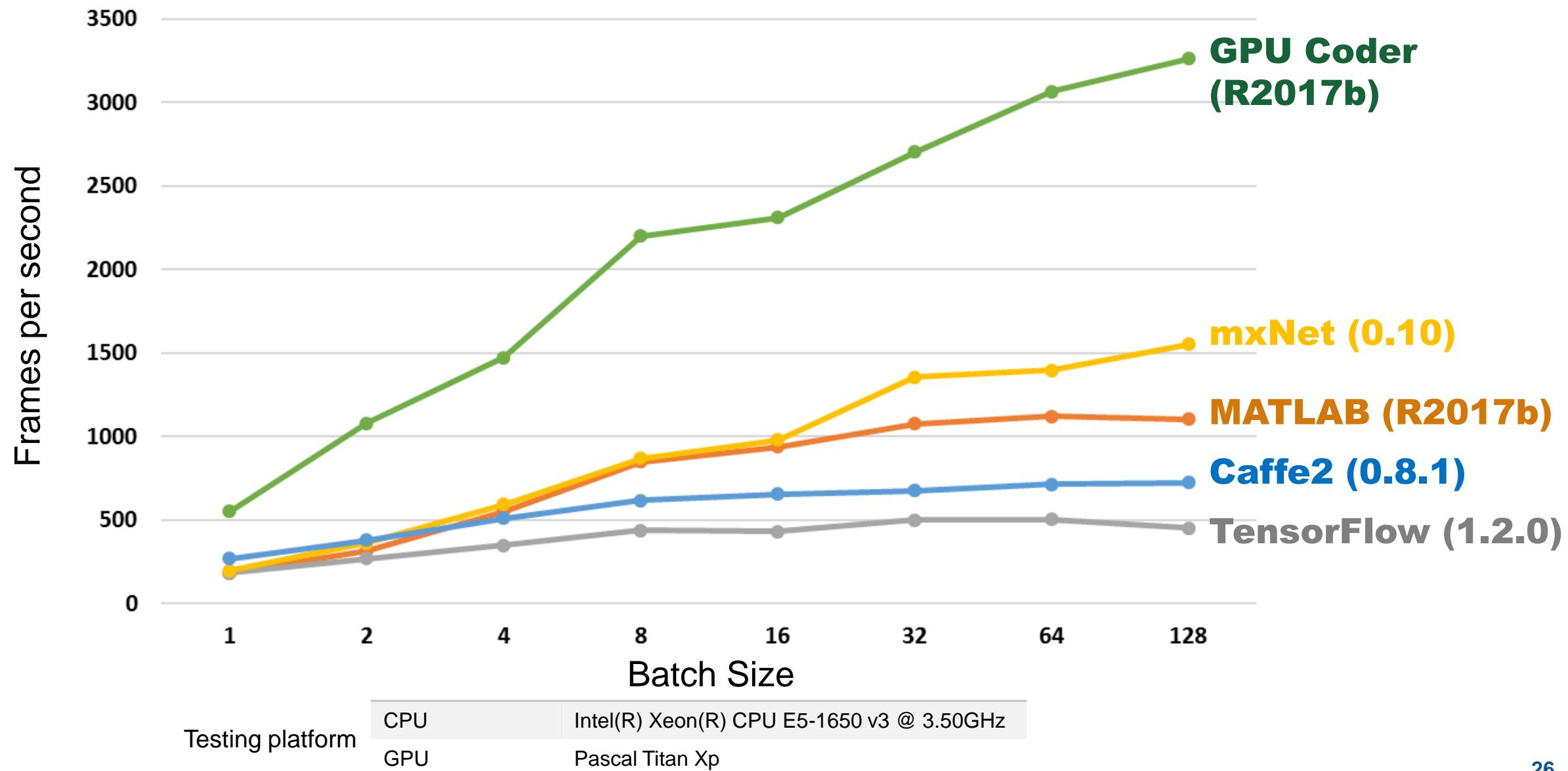
```
function out = alexnet_predict(in)
persistent mynet;

if isempty(mynet)
    mynet =
coder.loadDeepLearningNetwork('alexnet.mat','alexnet');
end

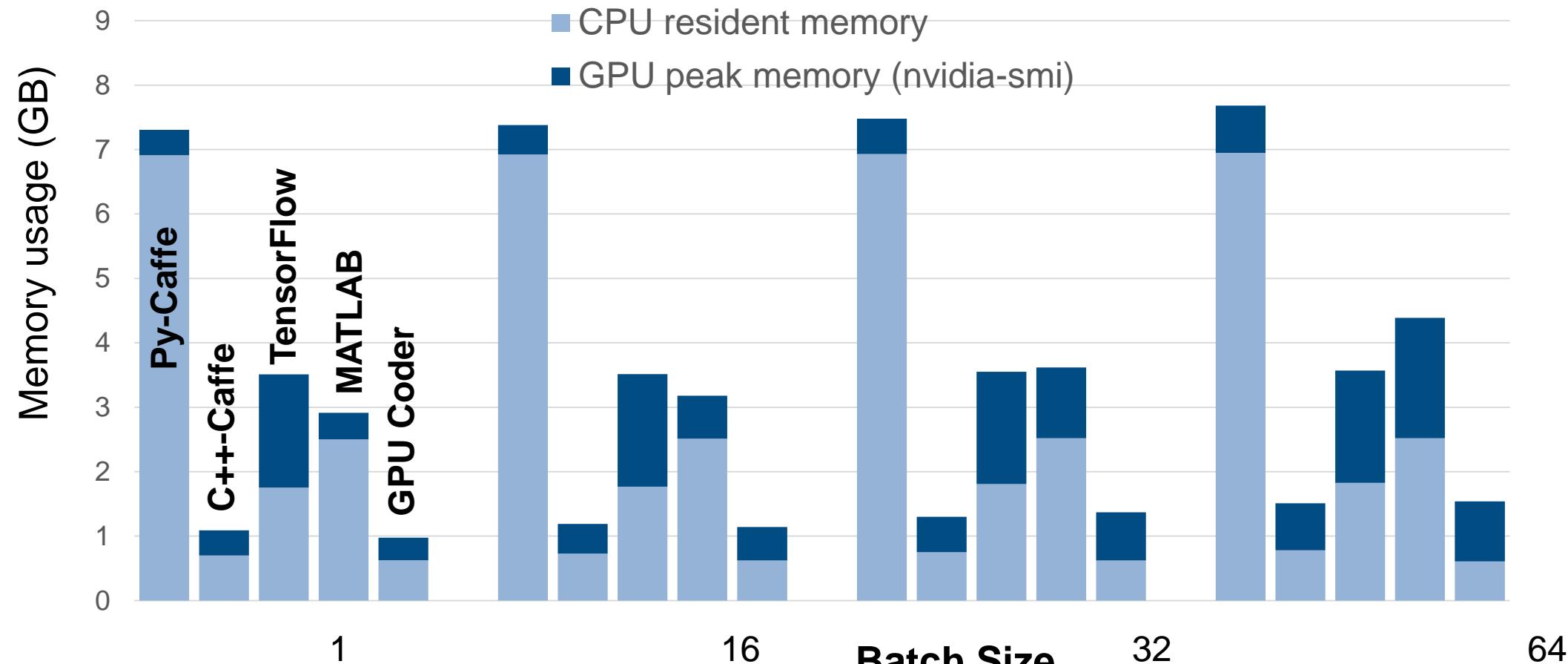
out = mynet.predict(in);
```



パフォーマンス(実行速度)比較 : Alexnet(推論部分)



パフォーマンス(メモリ使用率)比較 : Alexnet(推論部分)



Testing platform

CPU	Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz
GPU	Tesla K40c

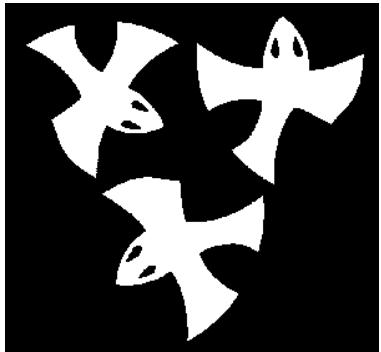
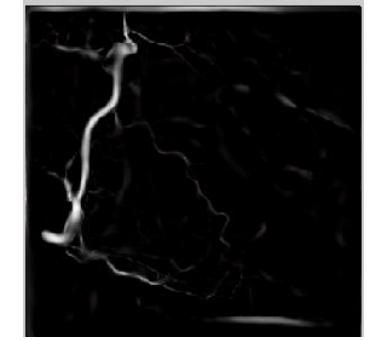
GPU Coderですぐに試せるコンピュータービジョン系サンプル



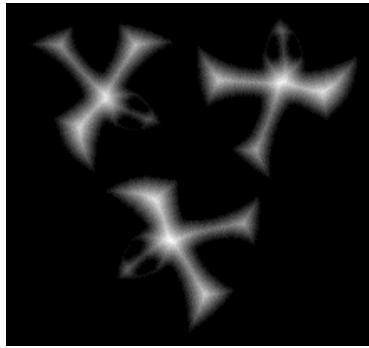
霧(ノイズ)除去
5x speedup



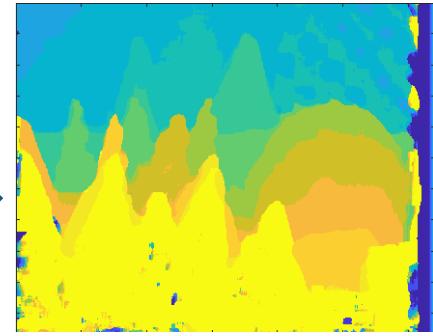
線強調フィルタ
3x speedup



距離変換
8x speedup



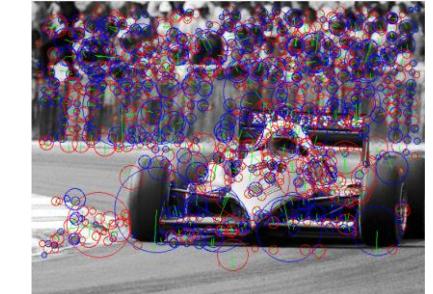
ディスパリティ算出
50x speedup



レイトレーシング
18x speedup

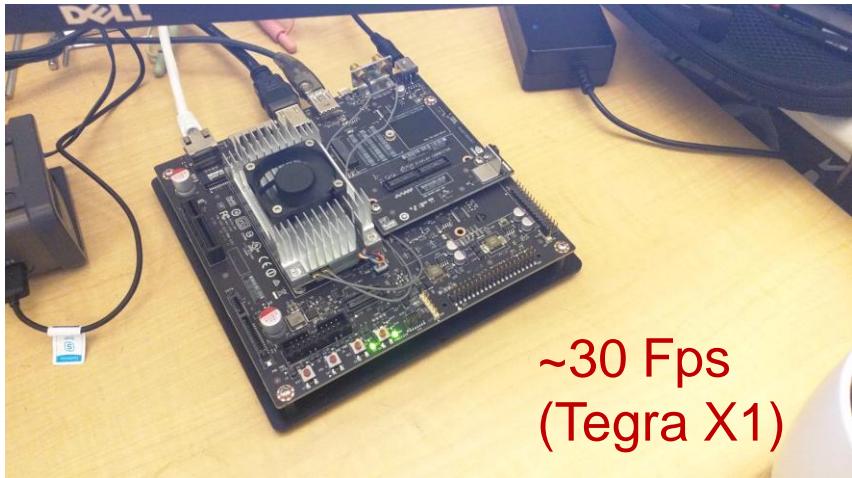


特徴点抽出
700x speedup



GPU Coderですぐに試せるディープラーニング サンプル

一般物体認識
(Alexnet)



~30 Fps
(Tegra X1)

自動車
検出



~20 Fps
(K40c)

人物検出



~66 Fps
(Tegra X1)

白線検出



~130 Fps
(K40c)

画像処理以外でも利用可能 – 多くのMATLAB関数をサポート

Fully Supported Functions

You can generate efficient CUDA® C code for a subset of MATLAB® built-in functions and toolbox functions that you call from MATLAB code. These functions appear in alphabetical order in the following table.

To find partially-supported functions, see [Partially Supported Functions](#).

Name	Product
abs	MATLAB
accumneg	Fixed-Point Designer™
accumpos	Fixed-Point Designer
acos	MATLAB
acosd	MATLAB
acosh	MATLAB
acot	MATLAB
acotd	MATLAB
affine2d	Image Processing Toolbox™
and	MATLAB
angle	MATLAB
asin	MATLAB
asind	MATLAB
asinh	MATLAB
atan	MATLAB
atan2	MATLAB
atan2d	MATLAB
atand	MATLAB
atanh	MATLAB
bin2dec	MATLAB
bitand	MATLAB



$$\begin{aligned}
 & \Rightarrow x^2 + px + q = 0 \quad W = \int_{s_1}^{s_2} F(s) \cdot \cos \alpha ds \quad v = \frac{ds}{dt} \\
 & \Rightarrow x_{\text{sol}} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q} \quad \tan x = \frac{e^x - e^{-x}}{e^x + e^{-x}}; \quad \theta = I \cdot N \\
 & F_r = \frac{1}{2\pi} \cdot \frac{1}{TLC}; \quad w = 2\pi f_r \quad u_c = U(1 - e^{-t/RC}); \quad C + O_2 \rightarrow CO_2 \\
 & 4Fe + 3O_2 \rightarrow 2Fe_2O_3 + 8SO_4 \\
 & -\frac{d}{dt} \int_B dA = \oint_E dl = - \int_A \left(\frac{\partial B}{\partial t} + \text{rot}(B \times v) \right) dA \quad ?x \# Y; \quad z = x \\
 & HCl + H_2O \rightleftharpoons Cl^- + H_3O^+ \quad a^2 = b^2 + c^2 \quad \omega_{\text{int}} = \frac{1}{2} \cdot J \omega^2 \\
 & V = \frac{1}{6} \pi h (3e_1^2 + 3e_2^2 + L^2) \quad P_v = \int_{\rho=0}^{\rho=L} \int_{\theta=0}^{\pi} \frac{r^2}{80_z} H_p H_p^* \sin \theta d\theta dr
 \end{aligned}$$



377個の組み込み関数に対応!

Agenda

- Introduction
- GPU・CUDA Cについて
- GPU Coder™による効率的なGPU実装
- まとめ

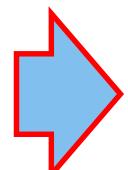
まとめ：いまからはじめる組み込みGPU実装 コンピュータービジョン・ディープラーニングのGPU実装

①統合開発環MATLAB

- ✓ コンピュータービジョン・ディープラーニングのアルゴリズム開発環境として強力なMATLAB
- ✓ アルゴリズム開発からGPUまで、同一環境上で実現可能

②新製品：GPU Coder

- ✓ CUDAの文法を知らなくても自動コード生成でGPUを利用可能
- ✓ エンジニアのスキルに依存しない、再現性の高いコード生成
- ✓ パフォーマンスの高いコード：処理速度、メモリ使用量
- ✓ すぐに始められるサンプル集

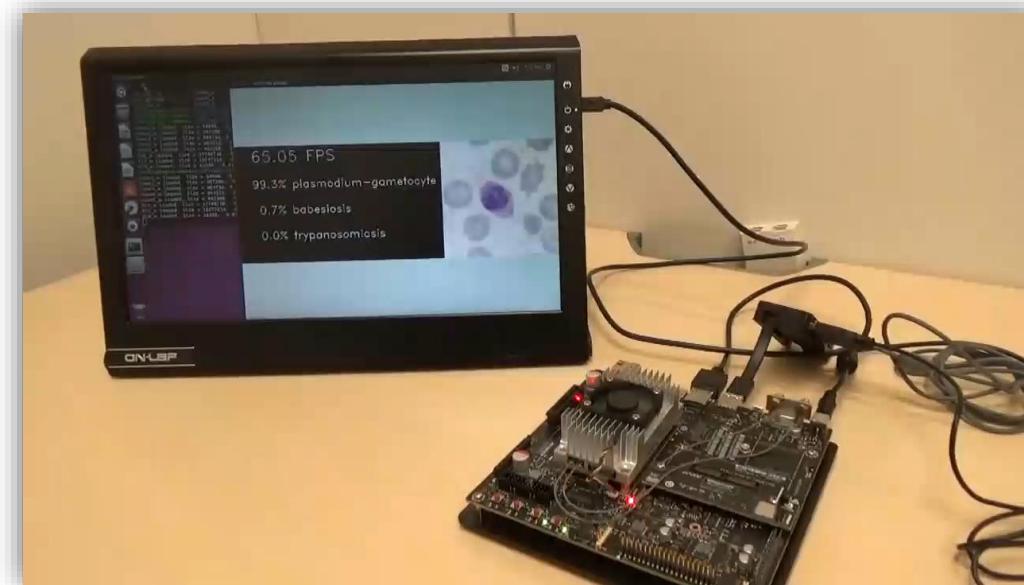


MATLABを使って、コンピュータービジョン・ディープラーニングのアルゴリズムの開発から実装までを効率的に実現！

Next Steps : 展示ブースへ是非お越し下さい



物体検出
(YOLO ネットワーク)



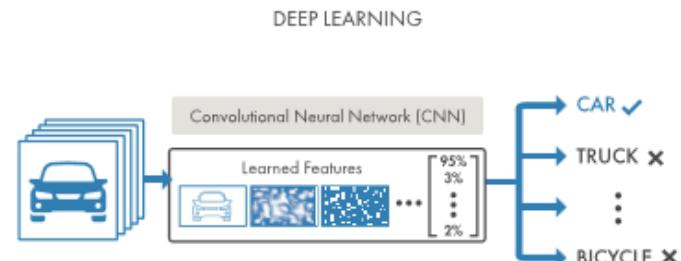
血液検査画像分類



Next Steps : 関連セッションのご案内

ディープラーニングによる画像認識の基礎と
実践ワークフロー

B3, 15:10 ~ 15:50



MathWorks Japan
福本 拓司

MATLABコードからの組み込み用Cコード生成の
ワークフローと最適化のコツ

F5, 17:10 ~ 17:50



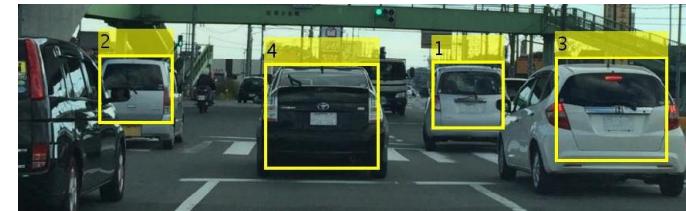
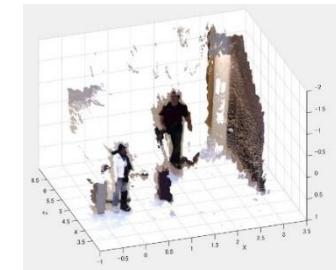
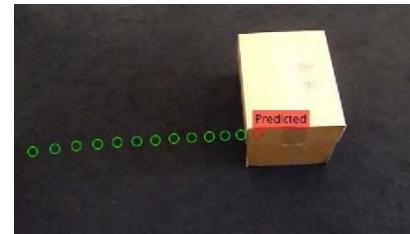
MathWorks Japan
松本 充史

Next Steps : 画像処理・コンピュータービジョン・機械学習無料セミナー

申し込みは弊社ウェブサイトより

<https://jp.mathworks.com/company/events/seminars/ipv-tokyo-2258970.html>

- 日時 : 2017年11月21日 13:30-17:00
- 場所 : 品川シーズンテラスカンファレンス(JR品川駅より徒歩6分)
(アクセス : <http://www.sst-c.com/access/index.html>)
- **画像処理、コンピュータービジョン、機械学習の機能をご紹介！**
 - MATLABではじめる画像処理ワークフロー
 - 例題で実感するMATLABの画像処理機能
 - MATLABで試す！機械学習の応用例





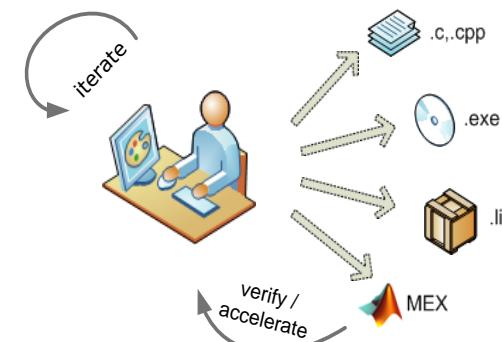
© 2017 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

GPU Coder関連製品

GPU Coderに
必須となります

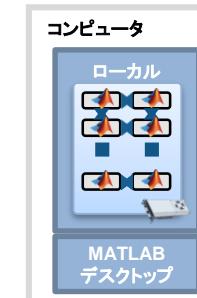
MATLAB Coder™

- MATLABプログラムからC/C++コードを生成
- MATLAB上で、アルゴリズム開発から実装までフローを統合



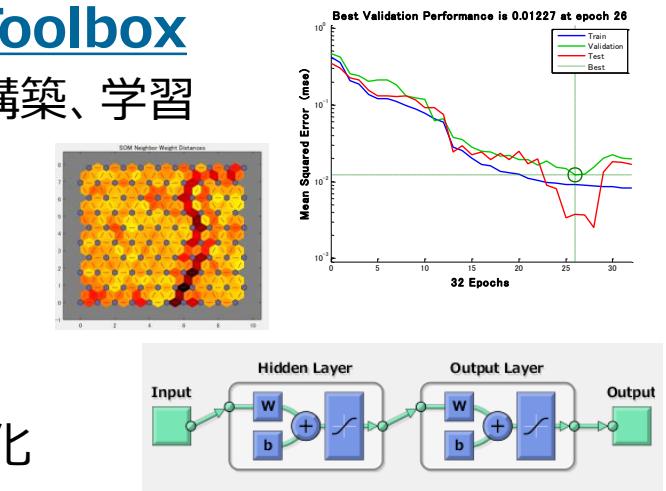
Parallel Computing Toolbox

- MATLAB & Simulinkと連携した並列処理
- 対話的な並列計算実行
- GPGPUによる高速演算
- ジョブおよびタスクの制御



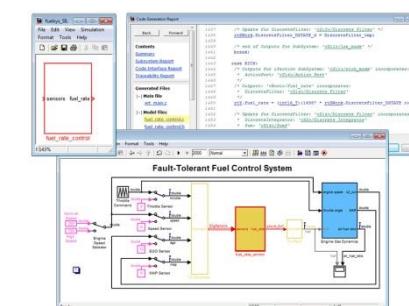
Neural Network Toolbox

- ニューラルネットワークの構築、学習
- データフィッティング
- クラスタリング
- パターン認識
- 深層学習
- GPUによる計算の高速化



Embedded Coder®

- MATLABプログラム/Simulinkモデルから組込み用C/C++コードを自動生成

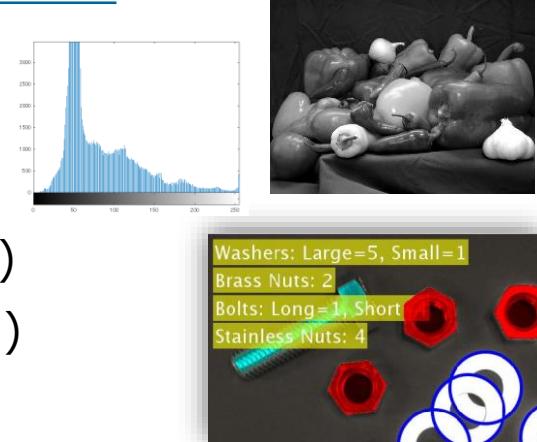


```
#include "structuredflow.h"
#include "Chart.h"
void Chart(void);
void Chart(void)
{
    start();
    do {
        if(input1) {
            one();
        } else if(input2) {
            two();
        } else {
            three();
        }
        loop();
    } while(looptest());
    end();
}
```

GPU Coder関連製品：画像処理・コンピュータビジョン

Image Processing Toolbox™

- コーナー、円検出
- 幾何学的変換
- 各種画像フィルタ処理
- レジストレーション（位置合せ）
- セグメンテーション（領域分割）
- 画像の領域の定量評価



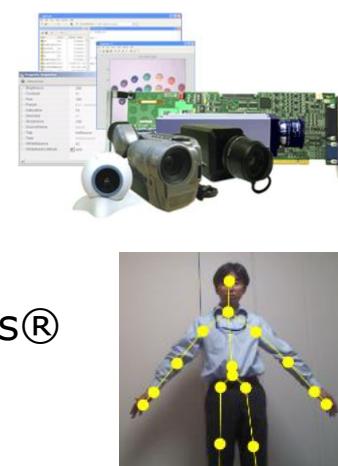
Computer Vision System Toolbox™

- カメラキャリブレーション
- 特徴点・特徴量抽出
- 機械学習による物体認識
- 動画ストリーミング処理
- トラッキング
- ステレオビジョン・3D表示



Image Acquisition Toolbox™

- デバイスから画像、動画直接取り込み
 - フレームグラバード
 - DCAM, Camera Link®
 - GigE Vision®, Webカメラ
 - Microsoft® Kinect® for Windows®



Statistics and Machine Learning Toolbox™

- 機械学習
- 多変量統計
- 確率分布
- 回帰と分散分析
- 実験計画
- 統計的工程管理

