User manual (**Matrix Multiplication using cublas**)

/\*

\* Three dimensional Matrix Multiplication using cublas

\*

\* This CUDA code can handle/work with any type of the input mxArrays,

\* GPUarray or standard matlab CPU array as input {prhs [0], prhs [1]:= mxGPUArray or CPU Array}

\* GpuArray output, C=**MM3D\_CUBLAS** (A, B, alpha) C=A\*B\*alpha.

\* Developed at UCL, Institute of Neurology, 12 Queen Square, WC1N 3AR, London

\* Welcome Trust Centre for Neuroimaging

\* Part of the project SPM (http://www.fil.ion.ucl.ac.uk/spm)

\* Copyright 2018

\* Kevin Bronik

\*/

Replacing Matlab operator “\*” for 3D calculation with parallel Matlab-CUDA syntax (with additional scaling options)

**Examples:**

**(First example**)

>> a= ones (5, 3, 3); 🡨--- (3D input array)

>> a

a(:,:,1) =

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

a(:,:,2) =

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

a(:,:,3) =

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

>> b= ones (3, 6, 3); 🡨--- (3D input array)

>> b

b(:,:,1) =

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

b(:,:,2) =

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

b(:,:,3) =

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

>> L=**MM3D\_CUBLAS** (a, b, 1) 🡨--- (Processing the multiplication) [L=a\*b\*1]

L(:,:,1) =

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

L(:,:,2) =

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

L(:,:,3) =

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

3 3 3 3 3 3

>>

**(Second example**)

>> A 🡨--- (3D input array)

A(:,:,1) =

1 2 3 4

5 6 7 8

9 10 11 12

A(:,:,2) =

13 14 15 16

17 18 19 20

21 22 23 24

>> B 🡨--- (3D input array)

B(:,:,1) =

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

B(:,:,2) =

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

>> C=MM3D\_CUBLAS (A, B, 1) 🡨--- (Processing the multiplication) [C=A\*B\*1]

C(:,:,1) =

10 10 10 10 10

26 26 26 26 26

42 42 42 42 42

C(:,:,2) =

58 58 58 58 58

74 74 74 74 74

90 90 90 90 90

>>

Or to get maximum performance

>> A=gpuArray (A) 🡨--- (3D input array-gpuArray object)

A(:,:,1) =

1 2 3 4

5 6 7 8

9 10 11 12

A(:,:,2) =

13 14 15 16

17 18 19 20

21 22 23 24

>> B=gpuArray (B) 🡨--- (3D input array-gpuArray object)

B(:,:,1) =

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

B(:,:,2) =

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

>> C=MM3D\_CUBLAS (A, B, 1) 🡨--- (Processing the multiplication) [C=A\*B\*1]

C(:,:,1) =

10 10 10 10 10

26 26 26 26 26

42 42 42 42 42

C(:,:,2) =

58 58 58 58 58

74 74 74 74 74

90 90 90 90 90

>>

**(Third example**)

**3D image processing-three dimensional Matrix Multiplication using mex-cuda-cublas**

This example shows how to process two colour image using three dimensional Matrix Multiplication.

Read an image into the workspace, then convert the image to double.

[X,map] = imread('**triangle**.tiff');

if ~isempty(map)

Im1 = ind2rgb(X,map);

end

[Y,map] = imread('**triangle**.tiff');

if ~isempty(map)

Im2 = ind2rgb(Y,map);

end

Perform an element wise addition of the colour image using the MM3D\_CUBLAS function.

K = **MM3D\_CUBLAS** (Im1, Im2, 0.0010);

Display the original colour image alongside the processed image.

figure

imshowpair(Im1,Im2,'montage')

title('Two original colour images (Left) and (Right)');

figure

imshowpair(Im1,K, 'montage')

title('Original colour image (Left) and the processed image (Right)');

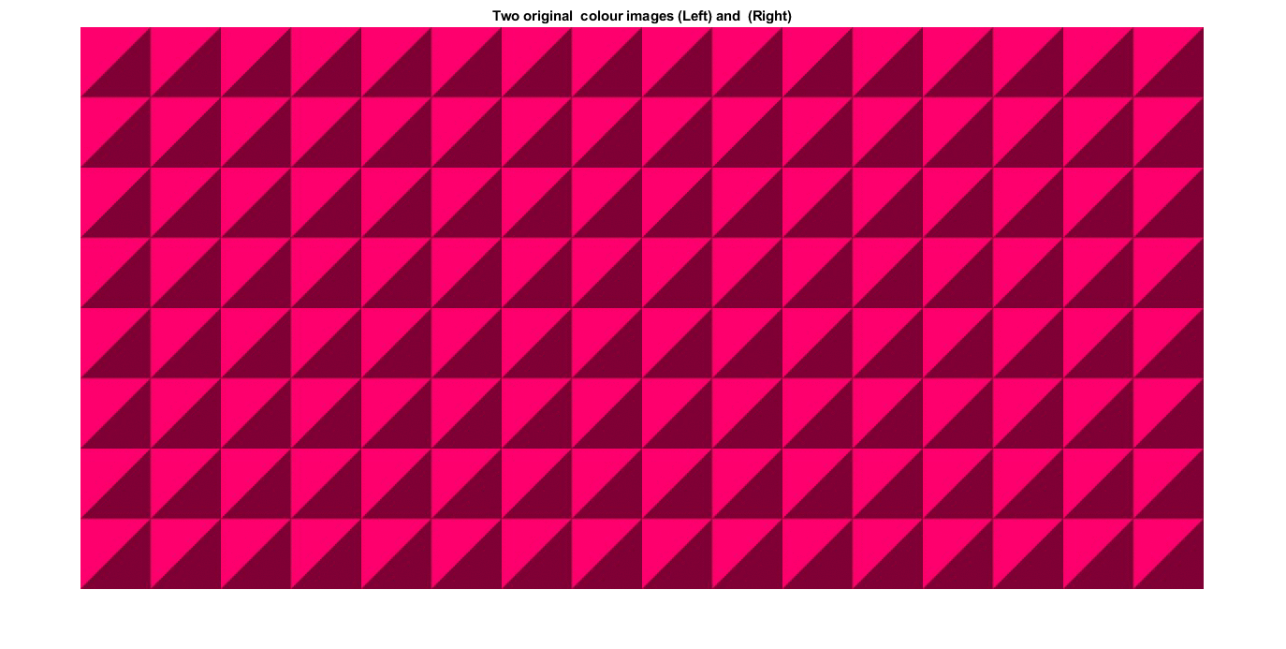
Name Size Bytes Class Attributes

Im1 600x600x3 (Image Dimensions) 8640000 double

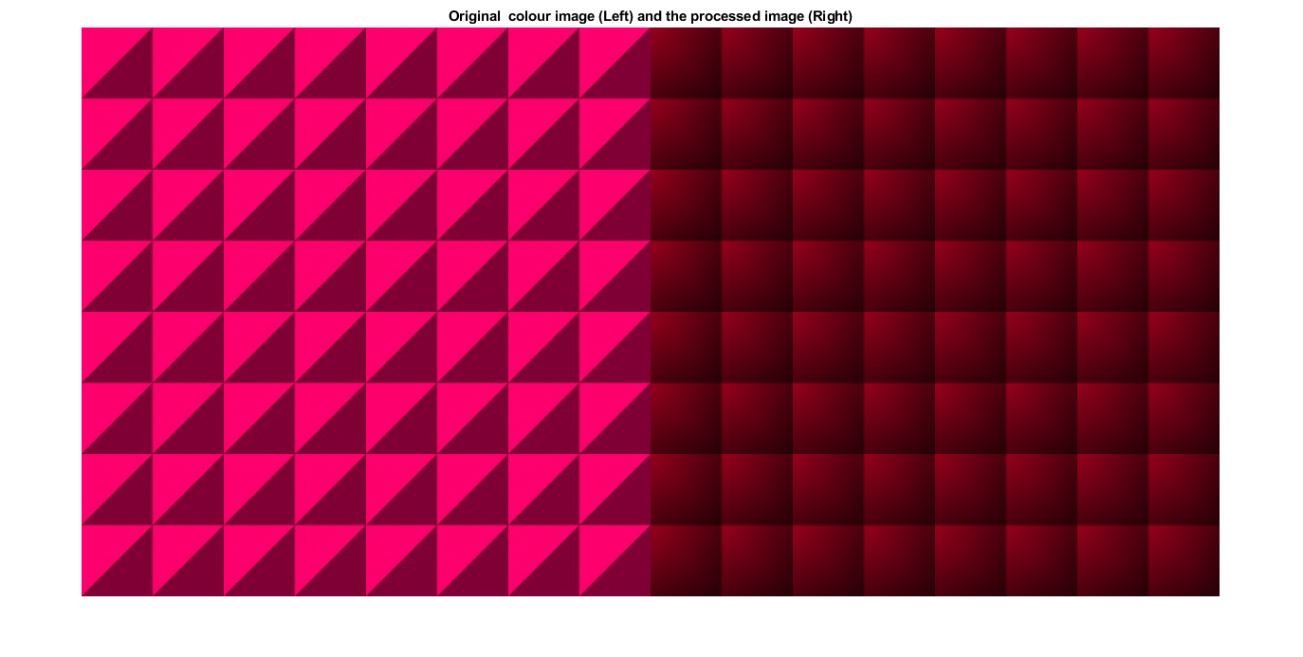
Name Size Bytes Class Attributes

Im2 600x600x3 (Image Dimensions) 8640000 double

**Before processing:**



**After processing: [**K = **MM3D\_CUBLAS**(Im1,Im2,0.0010);]



To compile:

First try the method described here:

<https://uk.mathworks.com/help/distcomp/run-mex-functions-containing-cuda-code.html>

After successful compiling running and testing then simply try following statement (copy and paste in Matlab and enter):

>> debug\_MM3D\_CUBLAS\_cu (false)

See the file “debug\_MM3D\_CUBLAS\_cu.m”