User manual (**Element wise Addition**)

/\*

\* Element wise Addition

\* 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= **ELM\_CUDA** or **ELM\_CUDA\_3D** (A, B, alpha, beta) C=A\*alpha+B\*beta

\* 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 “+” with parallel Matlab-CUDA syntax (with additional scaling options)

**Examples:**

**(First example**)

>> A 🡨--- (3D input array)

A(:,:,1) =

1 2 3

4 5 6

7 8 9

A(:,:,2) =

10 11 12

13 14 15

16 17 18

A(:,:,3) =

19 20 21

22 23 24

25 26 27

>> B 🡨--- (3D input array)

B(:,:,1) =

1 2 3

4 5 6

7 8 9

B(:,:,2) =

10 11 12

13 14 15

16 17 18

B(:,:,3) =

19 20 21

22 23 24

25 26 27

>> C=ELA\_CUDA\_3D (A, B, 1, 1) 🡨--- (Processing the addition) [C=A\*1+B\*1]

C(:,:,1) =

2 4 6

8 10 12

14 16 18

C(:,:,2) =

20 22 24

26 28 30

32 34 36

C(:,:,3) =

38 40 42

44 46 48

50 52 54

>> K=ELA\_CUDA\_3D (A, B, 2, -1) 🡨--- (Processing the addition) [K=A\*2+B\*(-1)]

K(:,:,1) =

1 2 3

4 5 6

7 8 9

K(:,:,2) =

10 11 12

13 14 15

16 17 18

K(:,:,3) =

19 20 21

22 23 24

25 26 27

>>

**(Second example**)

**3D image processing using element wise addition**

This example shows how to process two colour image using the three-dimensional Element wise Addition.

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

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

if ~isempty(map)

Im1 = ind2rgb(X,map);

end

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

if ~isempty(map)

Im2 = ind2rgb(Y,map);

end

Perform an element wise addition of the colour images using the ELA\_CUDA\_3D function.

K = ELA\_CUDA\_3D (Im1, Im2, 0.5, 1);

Display the original colour image alongside the processed image.

figure

imshowpair(Im1,K,'montage')

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

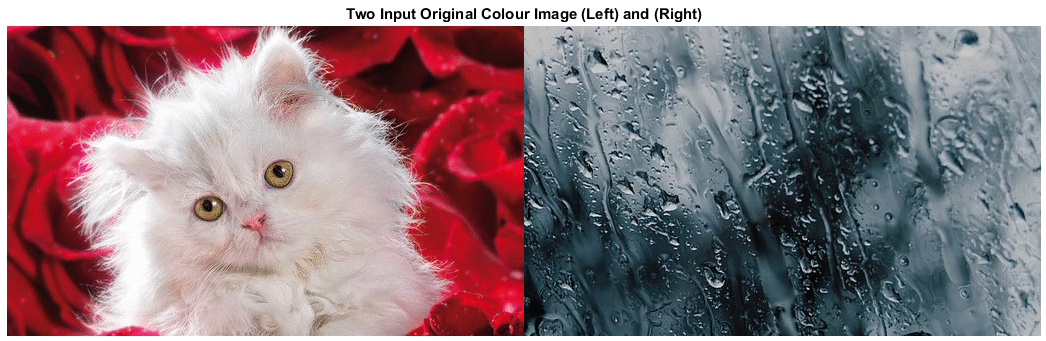
Name Size Bytes Class Attributes

Im1 310x517x3 (Image Dimensions) 3846480 double

Name Size Bytes Class Attributes

Im2 310x517x3 (Image Dimensions) 3846480 double

**Before processing:**

{\displaystyle \left({\begin{array}{ccc}\mathrm {a} \_{11}&\mathrm {a} \_{12}&\mathrm {a} \_{13}\\\mathrm {a} \_{21}&\mathrm {a} \_{22}&\mathrm {a} \_{23}\\\mathrm {a} \_{31}&\mathrm {a} \_{32}&\mathrm {a} \_{33}\end{array}}\right)\circ \left({\begin{array}{ccc}\mathrm {b} \_{11}&\mathrm {b} \_{12}&\mathrm {b} \_{13}\\\mathrm {b} \_{21}&\mathrm {b} \_{22}&\mathrm {b} \_{23}\\\mathrm {b} \_{31}&\mathrm {b} \_{32}&\mathrm {b} \_{33}\end{array}}\right)=\left({\begin{array}{ccc}\mathrm {a} \_{11}\,\mathrm {b} \_{11}&\mathrm {a} \_{12}\,\mathrm {b} \_{12}&\mathrm {a} \_{13}\,\mathrm {b} \_{13}\\\mathrm {a} \_{21}\,\mathrm {b} \_{21}&\mathrm {a} \_{22}\,\mathrm {b} \_{22}&\mathrm {a} \_{23}\,\mathrm {b} \_{23}\\\mathrm {a} \_{31}\,\mathrm {b} \_{31}&\mathrm {a} \_{32}\,\mathrm {b} \_{32}&\mathrm {a} \_{33}\,\mathrm {b} \_{33}\end{array}}\right)}

**After processing: [**K = ELA\_CUDA\_3D (Im1, Im2, 0.5, 1);**]**

****

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\_ELA\_CUDA\_cu(false)

See the file “debug\_ELA\_CUDA\_cu.m”