HITEC University, Taxila
Department of Computer Engineering

BS Computer Engineering Program

Course Title:	**EC-444 Parallel & Distributed Computing 3 (2+1)**
Batch / Semester:	Batch 2020 / 6th Semester
Lab Instructor:	Fasih Ahmad Janjua
Target PLO:	PLO9: Individual & Teamwork

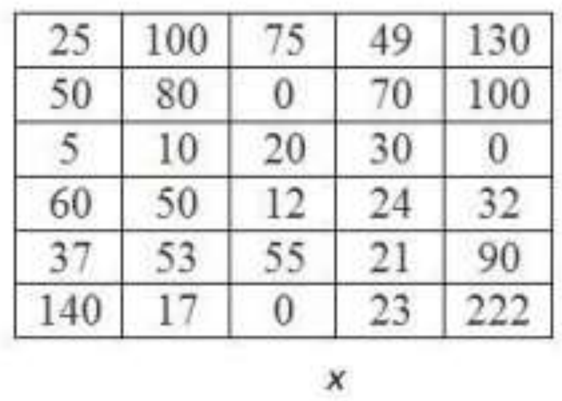
<u>Design Project</u>

Implementation of the 2D-Convolution in CUDA C

Convolution is the most important and fundamental concept in signal processing and analysis. Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. Mathematically, it describes a rule of how to combine two functions or pieces of information to form a third function. The feature map (or input data) and the kernel are combined to form a transformed feature map. The convolution algorithm is often interpreted as a filter, where the kernel filters the feature map for certain information.

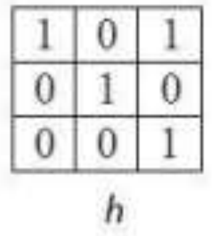
Mathematical Example of 2D Convolution

Let us try to compute the pixel value of the output image resulting from the convolution of 5×5 sized image matrix *x* with the kernel *h* of size 3×3, shown below in Figure 1.



25	100	75	49	130
50	80	0	70	100
5	10	20	30	0
60	50	12	24	32
37	53	55	21	90
140	17	0	23	222

x



1	0	1
0	1	0
0	0	1

h

Figure 1: Input matrices, where x represents the original image and h represents the kernel.

To accomplish this, the step-by-step procedure to be followed is outlined below.

Step 1: Matrix inversion

This step involves flipping of the kernel along, say, rows followed by a flip along its columns, as shown in Figure 2.

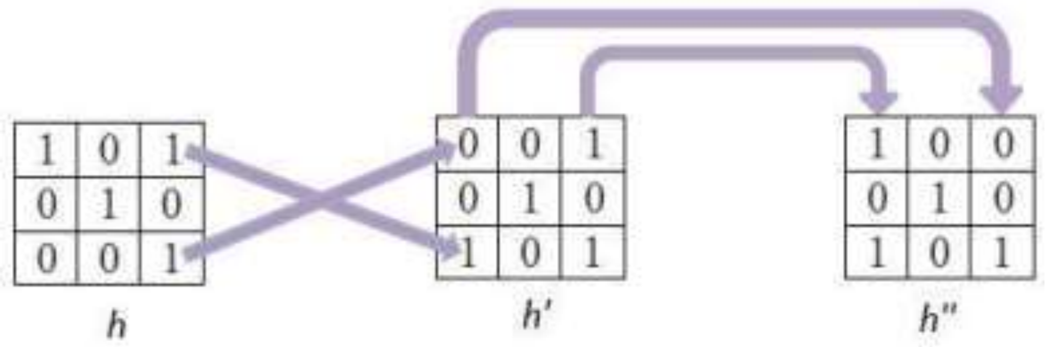


Figure 2: Pictorial representation of matrix inversion.

As a result, every (i,j)th element of the original kernel becomes the (j,i)th element in the new matrix.

Step 2: Slide the kernel over the image and perform MAC operation at each instant Overlap the inverted kernel over the image, advancing pixel-by-pixel.

For each case, compute the product of the mutually overlapping pixels and calculate their sum. The result will be the value of the output pixel at that particular location. For this example, non-overlapping pixels will be assumed to have a value of ‘0’. We'll discuss this in more detail in the next section on “Zero Padding”.

In the present example, we'll start sliding the kernel column-wise first and then advance along the rows.

Pixels Row by Row

First, let's span the first row completely and then advance to the second, and so on and so forth.

During this process, the first overlap between the kernel and the image pixels would result when the pixel at the bottom-right of the kernel falls on the first-pixel value at the top-left of the image matrix. Both of these pixel values are highlighted and shown in dark red color in Figure 3a. So, the first pixel value of the output image will be 25 × 1 = 25.

Next, let us advance the kernel along the same row by a single pixel. At this stage, two values of the kernel matrix (0, 1 – shown in dark red font) overlap with two pixels of the image (25 and 100 depicted in dark red font) as shown in Figure 3b. So, the resulting output pixel value will be 25 × 0 + 100 × 1 = 100.

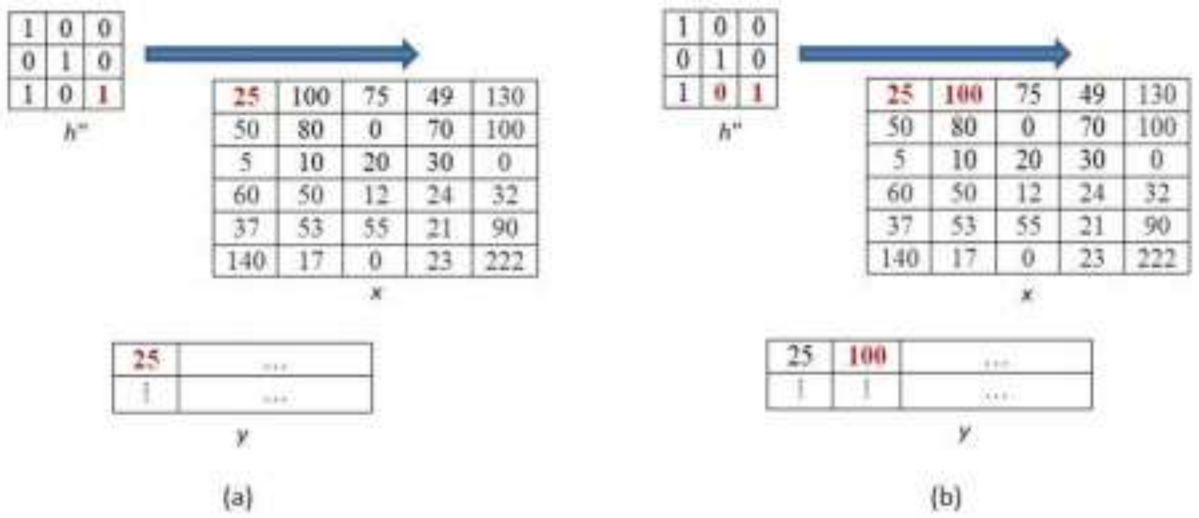


Figure 3a, 3b. Convolution results obtained for the output pixels at location (1,1) and (1,2).

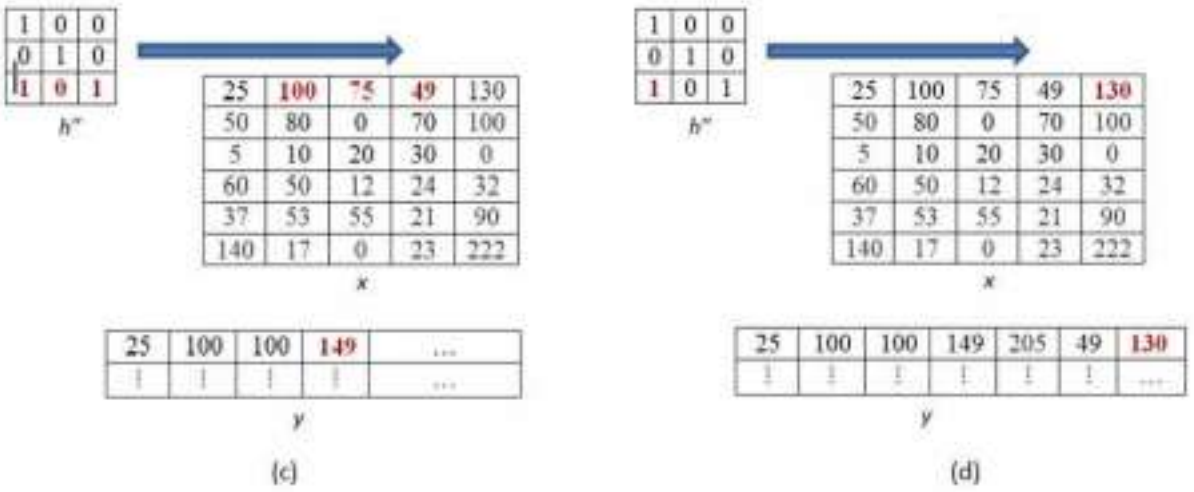


Figure 3c, 3d: Convolution results obtained for the output pixels at location (1,4) and (1,7)

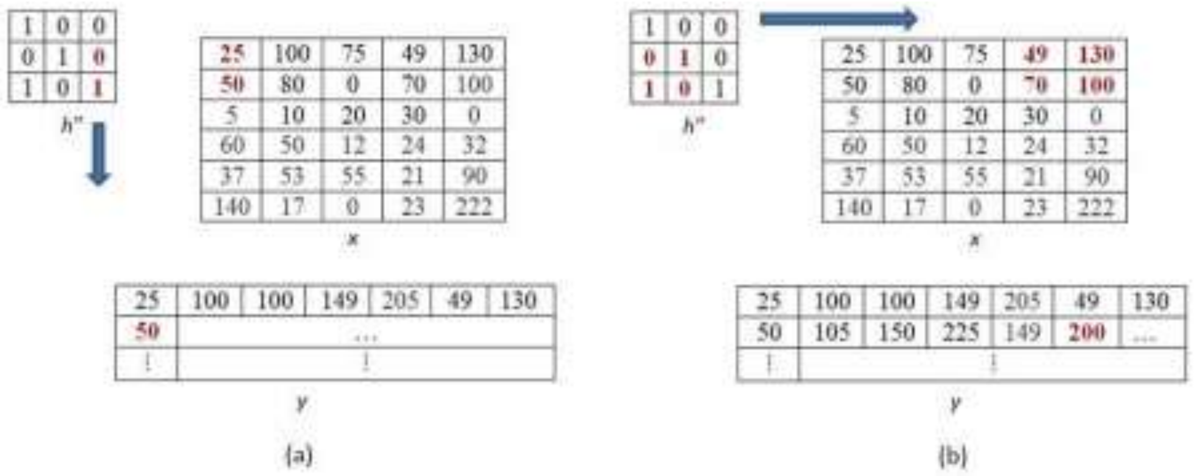
Advancing similarly, all the pixel values of the first row in the output image can be computed. Two such examples corresponding to fourth and seventh output pixels of the output matrix are shown in the figures 3c and 3d, respectively.

If we further slide the kernel along the same row, none of the pixels in the kernel overlap with those in the image. This indicates that we are done along the present row.

Move Down Vertically, Advance Horizontally

The next step would be to advance vertically down by a single pixel before restarting to move horizontally. The first overlap which would then occur is as shown in Figure 4a and by performing the MAC operation over them; we get the result as 25 × 0 + 50 × 1 = 50.

Following this, we can slide the kernel in horizontal direction till there are no more values which overlap between the kernel and the image matrices. One such case corresponding to the sixth pixel value of the output matrix (= 49 × 0 + 130 × 1 + 70 × 1 + 100 × 0 = 200) is shown in Figure 4b.



1	0	0
0	1	0
1	0	1

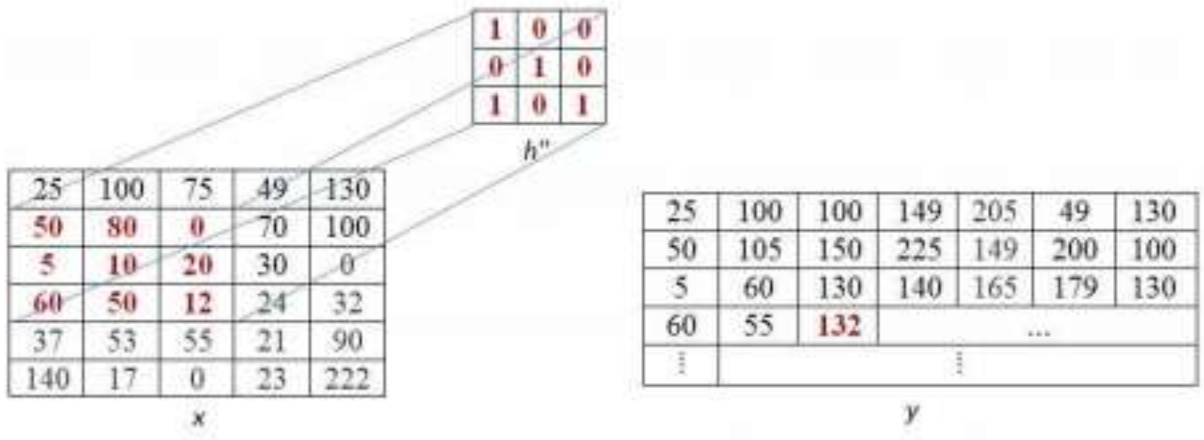
25	100	75	49	130
50	80	0	70	100
5	10	20	30	0
60	50	12	24	32
37	53	55	21	90
140	17	0	23	222

25	100	100	149	205	49	130
50	…					
⋮	⋮					

25	100	100	149	205	49	130
50	105	150	225	149	200	…
⋮	⋮					

Figure 4a, 4b. Convolution results obtained for the output pixels at location (2,1) and (2,6).

This process of moving one step down followed by horizontal scanning has to be continued until the last row of the image matrix. Three random examples concerned with the pixel outputs at the locations (4,3), (6,5) and (8,6) are shown in Figures 5a-c.



1	0	0
0	1	0
1	0	1

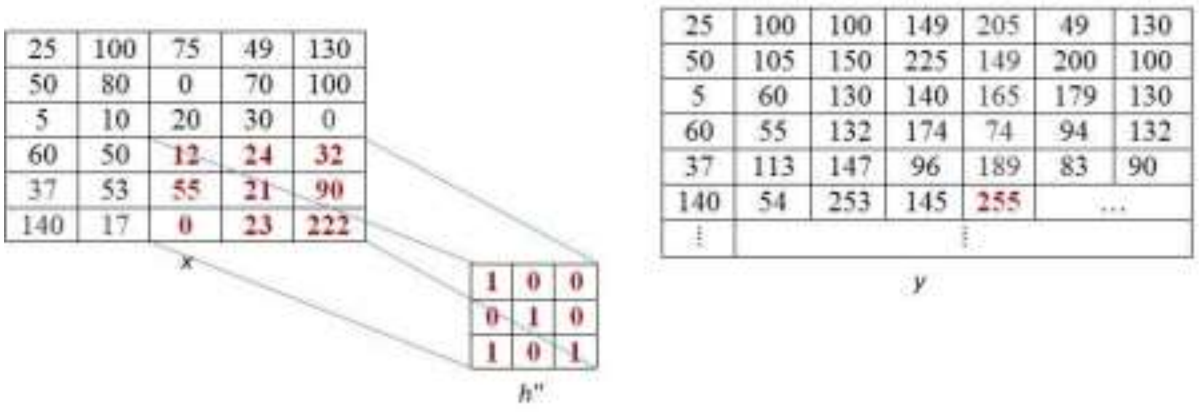
25	100	75	49	130
50	80	0	70	100
5	10	20	30	0
60	50	12	24	32
37	53	55	21	90
140	17	0	23	222

25	100	100	149	205	49	130
50	105	150	225	149	200	100
5	60	130	140	165	179	130
60	55	132	…			
⋮	⋮					

$$y(4,3) = 50 \times 1 + 80 \times 0 + 0 \times 0 + 5 \times 0 + 10 \times 1 + 20 \times 0 + 60 \times 1 + 50 \times 0 + 12 \times 1$$
$$= 50 + 0 + 0 + 0 + 10 + 0 + 60 + 0 + 12 = 132$$

Figure 5a. Convolution results obtained for the output pixels at (4,3)

HITEC University, Taxila
Department of Computer Engineering



25	100	75	49	130
50	80	0	70	100
5	10	20	30	0
60	50	12	24	32
37	53	55	21	90
140	17	0	23	222

x

1	0	0
0	1	0
1	0	1

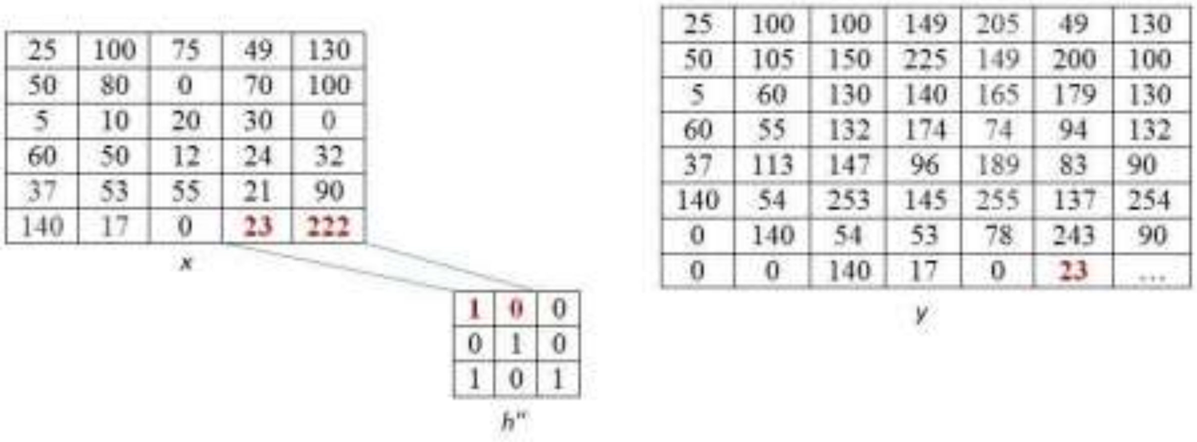
h"

25	100	100	149	205	49	130
50	105	150	225	149	200	100
5	60	130	140	165	179	130
60	55	132	174	74	94	132
37	113	147	96	189	83	90
140	54	253	145	255	…	
⋮			⋮			

y

$$y(6,5) = 12 \times 1 + 24 \times 0 + 32 \times 0 + 55 \times 0 + 21 \times 1 + 90 \times 0 + 0 \times 1 + 23 \times 0 + 222 \times 1$$
$$= 12 + 0 + 0 + 0 + 21 + 0 + 0 + 0 + 222 = 255$$

Figure 5b. Convolution results obtained for the output pixels at (6,5)



25	100	75	49	130
50	80	0	70	100
5	10	20	30	0
60	50	12	24	32
37	53	55	21	90
140	17	0	23	222

x

1	0	0
0	1	0
1	0	1

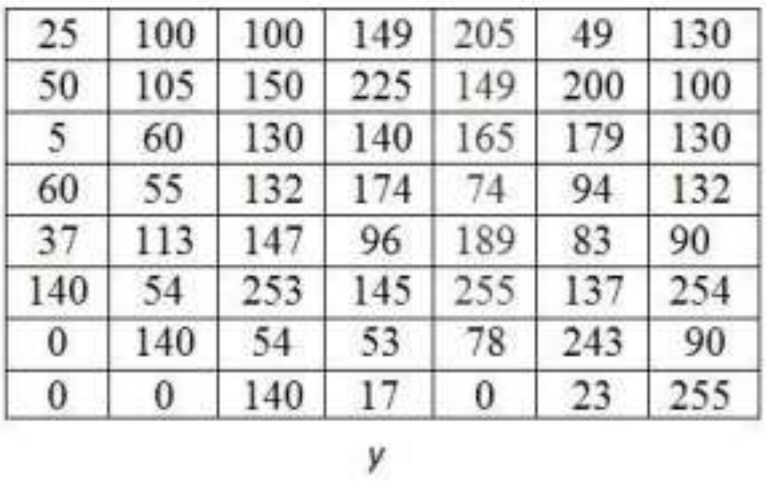
h"

25	100	100	149	205	49	130
50	105	150	225	149	200	100
5	60	130	140	165	179	130
60	55	132	174	74	94	132
37	113	147	96	189	83	90
140	54	253	145	255	137	254
0	140	54	53	78	243	90
0	0	140	17	0	23	…

y

$$y(8, 6) = 23 \times 1 + 222 \times 0 = 23 + 0 = 23$$

Figure 5c. Convolution results obtained for the output pixels at (8,6). Hence the resultant output matrix will be:



25	100	100	149	205	49	130
50	105	150	225	149	200	100
5	60	130	140	165	179	130
60	55	132	174	74	94	132
37	113	147	96	189	83	90
140	54	253	145	255	137	254
0	140	54	53	78	243	90
0	0	140	17	0	23	255

y

Figure 6. Our example's resulting output matrix.

HITEC University, Taxila
Department of Computer Engineering

Task:

Assume a grey scale image matrix-M (512 x 512) with **8-bits** pixel values initialized by random number generator. Use convolution-kernel as 15x15 matrix. Use Cuda thread block size of (16 x 16). Write complete cudaC code for implementing 2D convolution.

Deliverable

You need to submit the report, presentation and working source-code. Following should at least be the contents of the deliverable to clearly communicate your analysis, design and implementation.

Presentation

1. Not more than 10-slides in ppt.
2. Detailed profile and analysis
3. Parallelization strategy
4. Execution time and speedup results
5. Other details, challenges, important things you encountered

All deliverables must include in project report (hard form).

Source code

1. Properly formatted
2. Properly commented