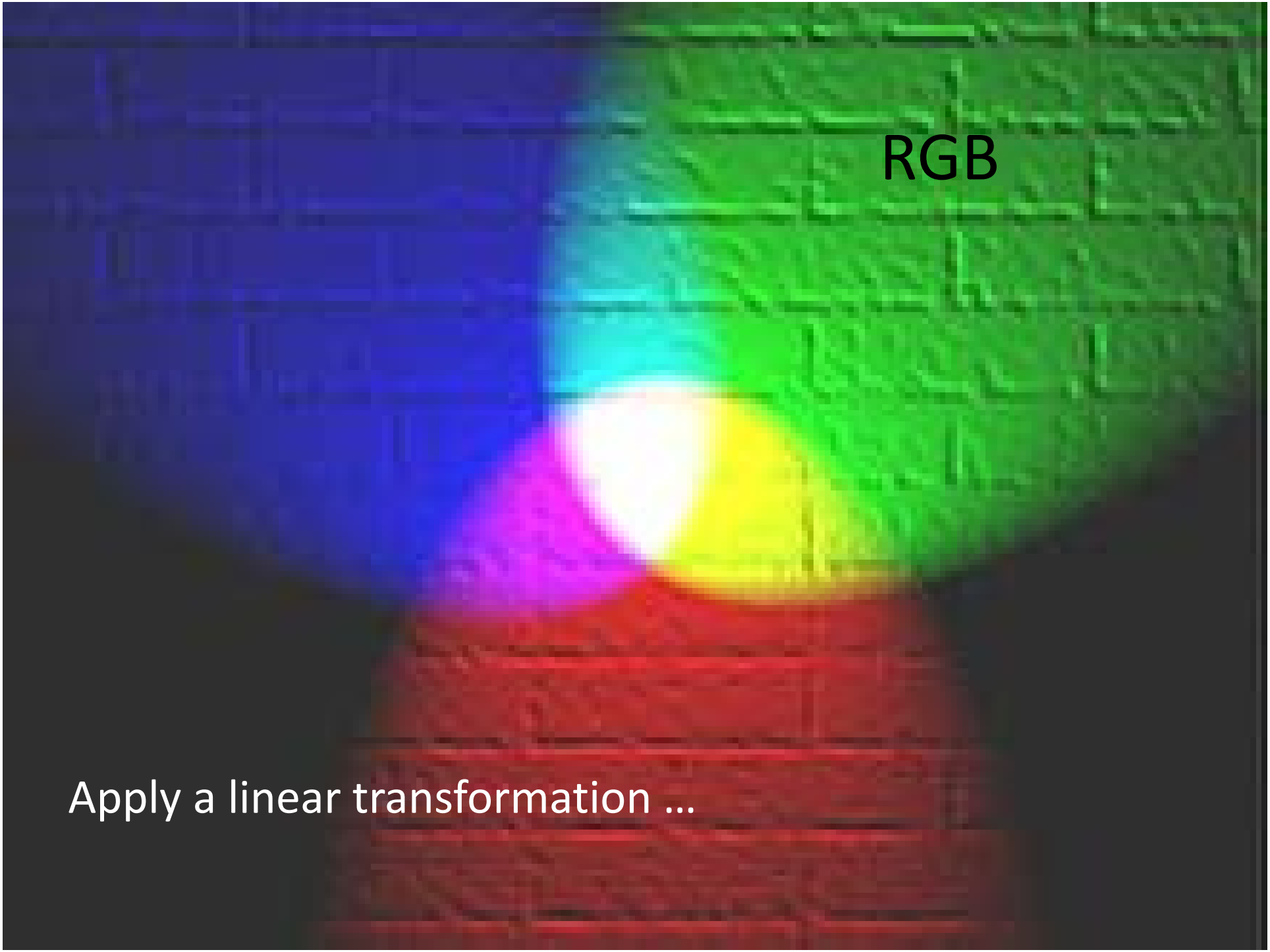
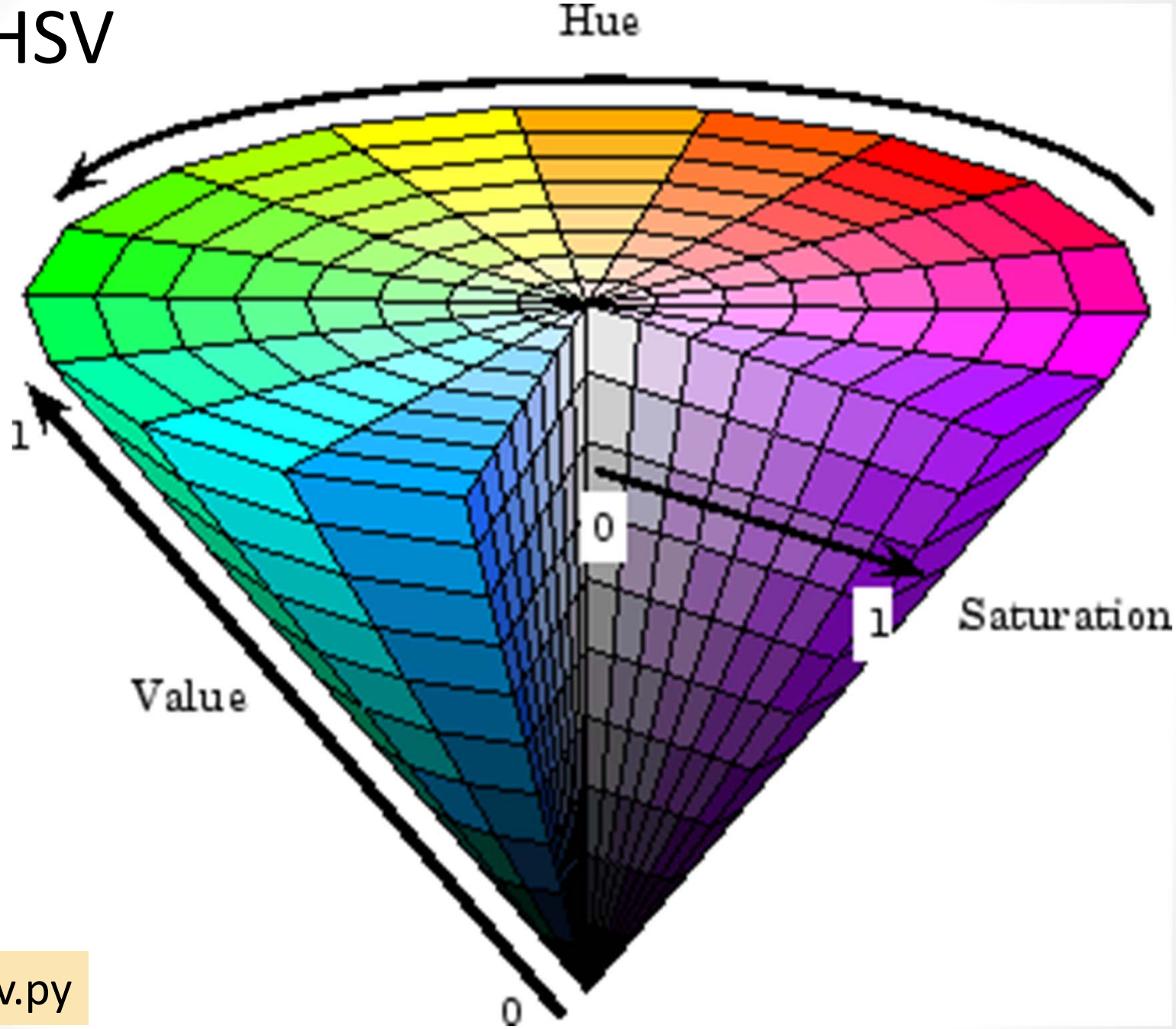
Color Transforms


RGB
Apply a linear transformation …

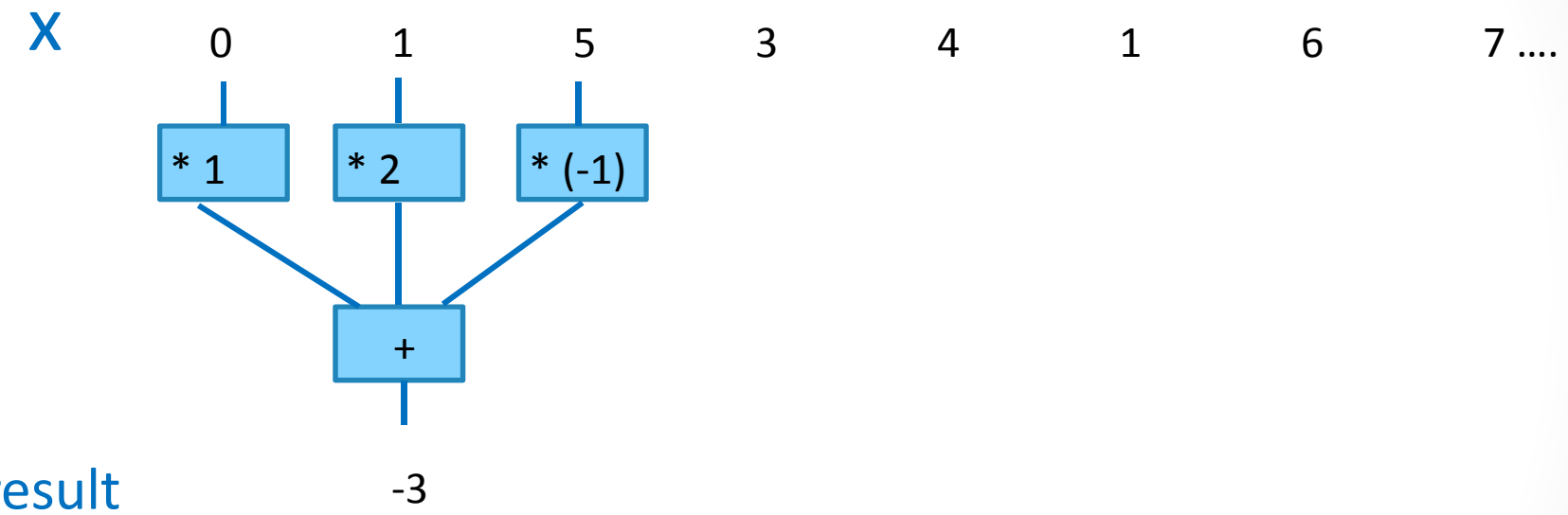
HSV



hsv.py

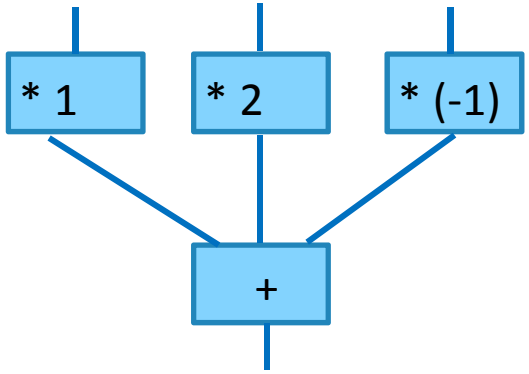
Image Filters

Convolution



Convolution

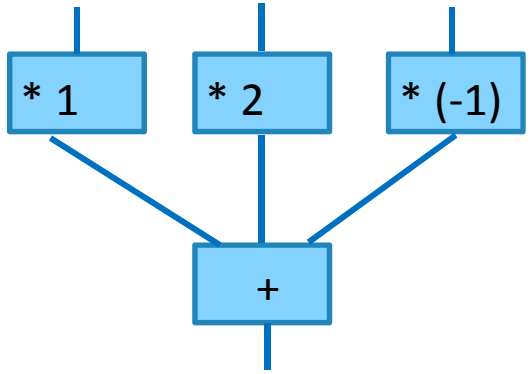
x 0 1 5 3 4 1 6 7 ….



result -3

Convolution

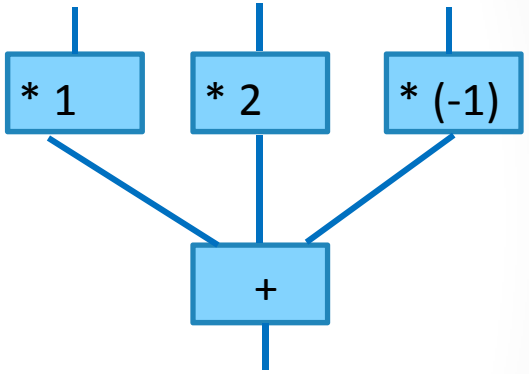
x 0 1 5 3 4 1 6 7 ….



result -3 8

Convolution

x	0	1	5	3	4	1	6	7 ….
result		-3	8	12	10	0	6	



Convolution

x	0	1	5	3	4	1	6	7 ….
kernel	1	2	-1					
result		-3	8	12	10	0	6	

Convolution

x	0	1	5	3	4	1	6	7 ….
kernel	1	2	-1					
result		-3	8	12	10	0	6	

---

x	0	1	5	3	4	1	6	7 ….
y	-1	2	1					
z			-3	8	12	10	0	6

$$z = x \otimes y$$

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-1

Destination image

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

$3 * 1 + 2 * 1 + 0 * (-2) = 5$

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5				

3 * 1 + 2 * 1 + 0 * (-2) = 5

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5				

1 * 1 + 5 * 1 + 4 * (-2) = -2

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5	-2			

1 * 1 + 5 * 1 + 4 * (-2) = -2

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5	-2			

3 * 1 + 3 * 1 + 1 * (-2) = 4

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5	-2			
		4			

3 * 1 + 3 * 1 + 1 * (-2) = 4

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5	-2			
		4			
			?		

What is the value of this destination pixel?

A. -3
B. -2
C. 3
D. 4
E. Something else.

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5	-2			
	4				
			?		

What is the value of this destination pixel?

A. -3
B. -2
C. 3
D. 4
E. Something else.

2D-Convolution

Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image

	5	-2			
	4				
			4		

2D-Convolution

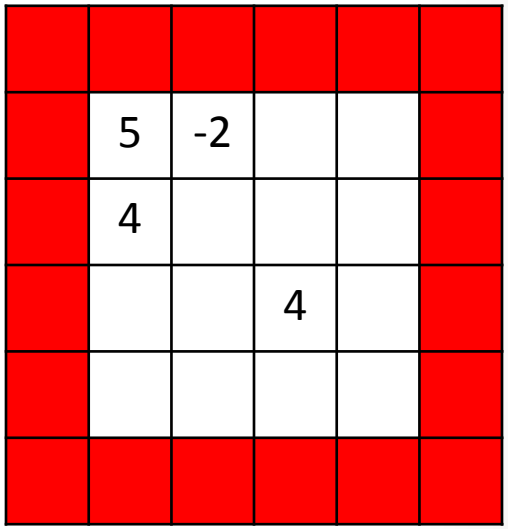
Source image

1	3	1	1	1	4
4	0	3	4	2	3
2	5	0	4	2	2
4	3	1	1	0	1
1	0	2	1	1	3
0	2	4	0	2	2

Kernel

0	1	0
0	0	0
1	0	-2

Destination image



1. Destination image is smaller
2. “Zero-padding” of source image

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Kernel

0	-1	0
-1	5	-1
0	-1	0

320					

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$

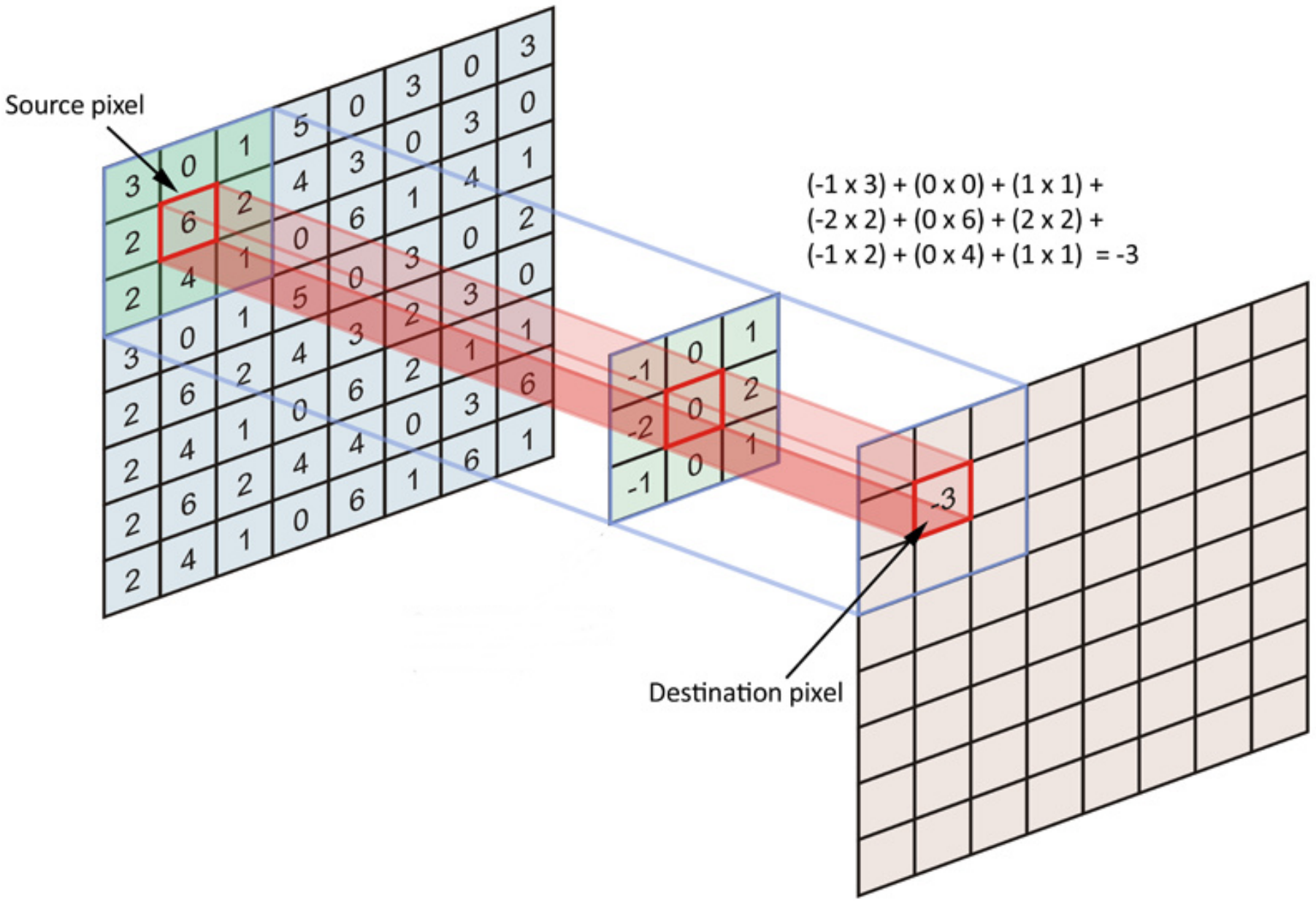

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
-3
Destination pixel

Image Filters

Kernel

-1		1
-2		2
-1		1

Assume grayscale images

Source pixel value 0 .. 255

Destination pixel value?

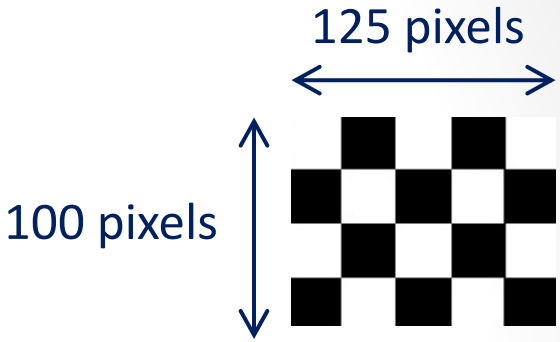
Normalize it so that it falls in the range 0 .. 255

Image Filters

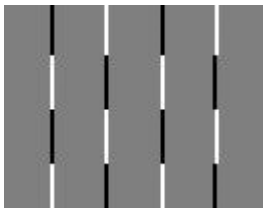
Kernel

-1		1

You apply this filter to a checkerboard image. What is the result after normalization?

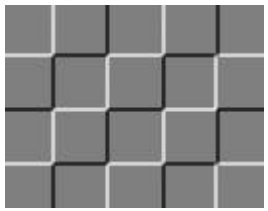


A.

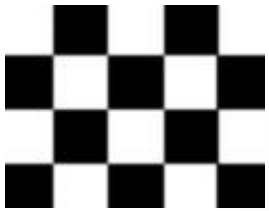


B.

C.



D.



E. Something else

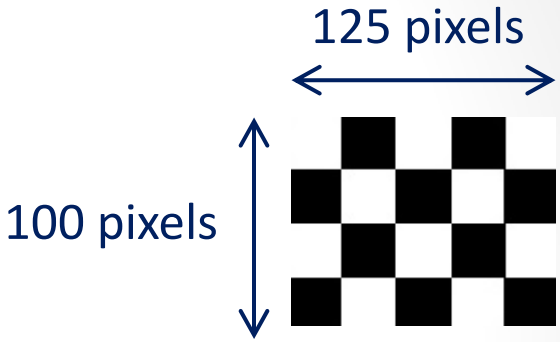
filterimage.py

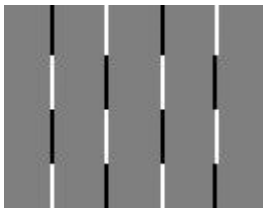
Image Filters

Kernel

-1		1

Vertical edge detection filter





Take absolute value
and re-normalize

Filterimage_comb.py

Image Filters

Kernel

-1		1

+ abs()
+ normalize

Vertical edge detection filter

Kernel

	1	
	-1	

+ abs()
+ normalize

Horizontal edge detection filter

Filterimage_comb.py

Image Filters

Kernel

-1		1

+ abs()
+ normalize

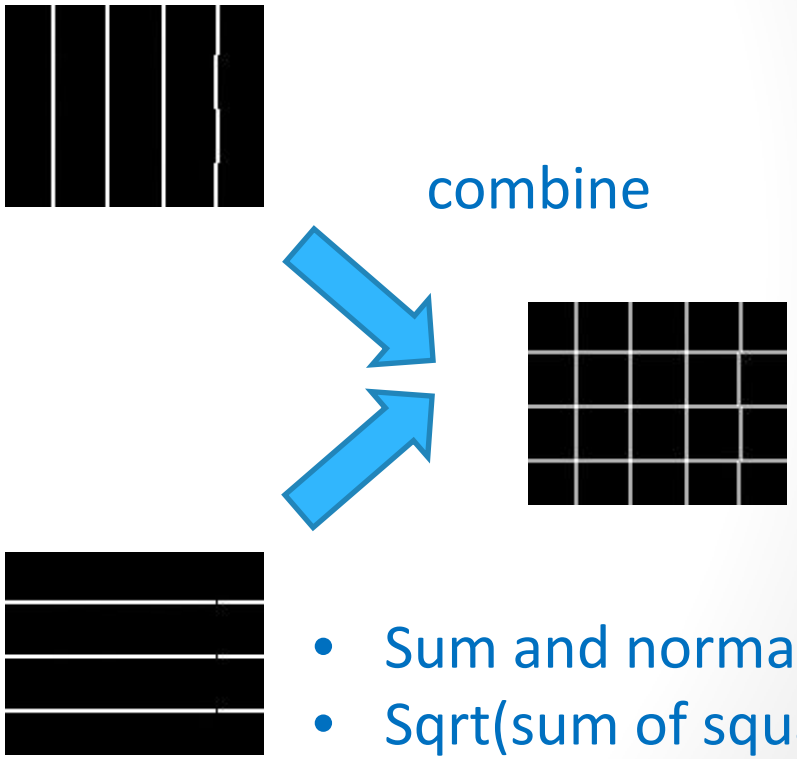
Vertical edge detection filter

Kernel

	1	
	-1	

+ abs()
+ normalize

Horizontal edge detection filter



- Sum and normalize
- Sqrt(sum of squares) and normalize

Image Filters

Kernel

	1	
1	1	1
	1	

+ normalize

What do you think this filter does to an image?

A. It finds diagonal edges
B. It blurs the image
C. It sharpens the image
D. It embosses the image
E. Something else.

Image Filters

Kernel

	1	
1	1	1
	1	

+ normalize

Blurring

Image Filters

Kernel

	1	
1	1	1
	1	

+ normalize

Blurring

Kernel

-1	-1	-1
-1	9	-1
-1	-1	-1

+ bound to 0 .. 255

Sharpening

Image Filters

Kernel

1	1	
1		-1
	-1	-1

+ normalize

Emboss

JPEG

- Lossy compression format

- We want to represent the image with fewer bits
  - Fewer pixels
  - Fewer bits per pixel
  - …