18-799: Applied Computer Vision

Spring 2026

Image Processing

Carnegie Mellon University
Africa

Objectives

In today's class we will learn

- What is image processing
- What is Linear filtering and how to apply it

Image Representation

- A digital image is a finite array of numbers, arranged on a 2D grid, where each number encodes a measurement made by the sensor.

- Formally:
  - Image $\longleftrightarrow$ matrix (or tensor)

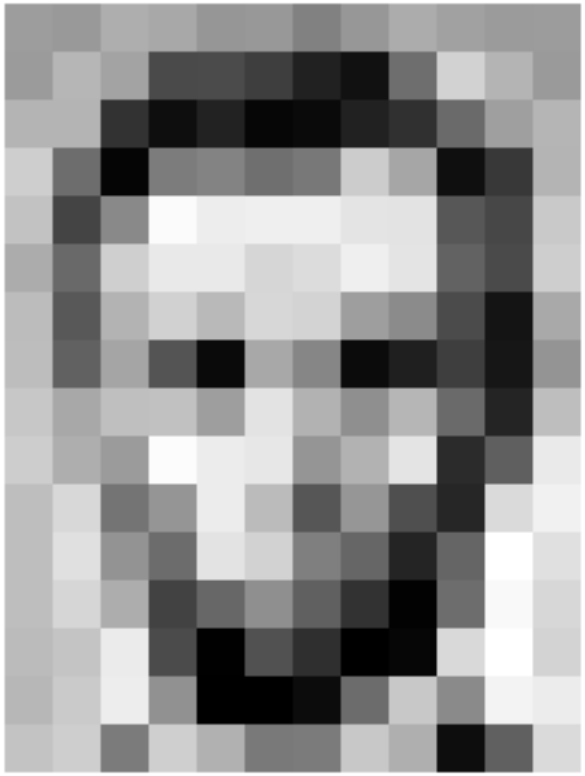
- This representation is what allows images to be processed using linear algebra, signal processing, and optimization.

Image Representation

$\longleftrightarrow$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

Image Representation



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

https://ai.stanford.edu/~syyeung/cvweb/Pictures1/imagematrix.png

Point operators

- The simplest kinds of image processing transforms
- Each output pixel's value depends on only the corresponding input pixel value (plus, potentially, some globally collected information or parameters)

Point operators

- The simplest kinds of image processing transforms
- Each output pixel's value depends on only the corresponding input pixel value (plus, potentially, some globally collected information or parameters)

Pixel transforms

- A general image processing operator maps one or more input images to an output image and can be written in pixel-wise form as

$$I'(x, y) = h(I(x, y))$$

- or, when multiple input images are involved,

$$I'(x, y) = h(I_0(x, y), I_1(x, y), \ldots, I_n(x, y)).$$

  - In both cases, the output value at location (x, y) is computed from the corresponding input pixel values.

Pixel transforms

- Two commonly used point processing operations are addition and multiplication by constants. A simple linear intensity transformation is given by

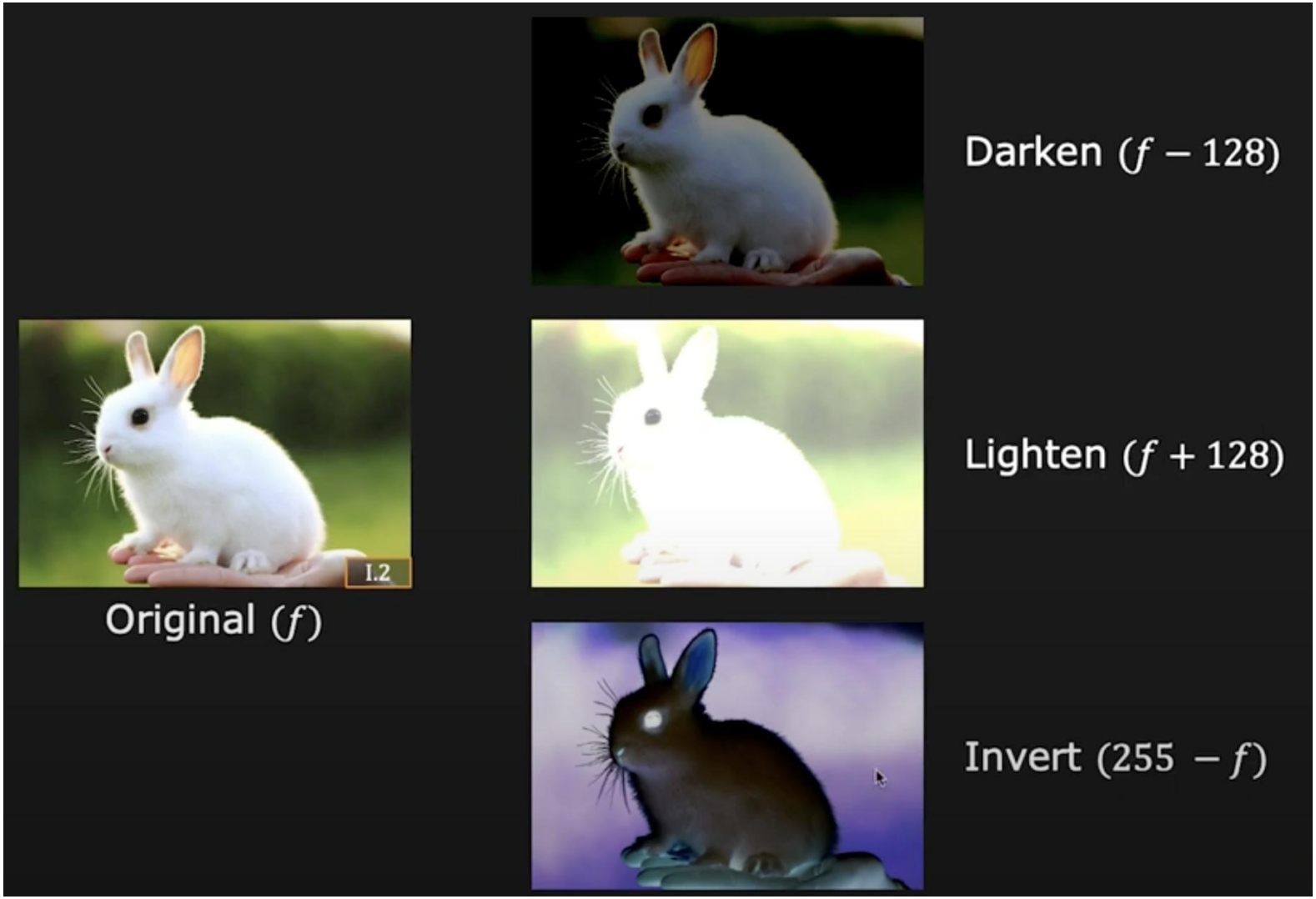
$$I'(x, y) = aI(x, y) + b$$

- where $a > 0$ and b are known as the gain and bias parameters; these are often interpreted as controlling contrast and brightness, respectively.
- In a more general, spatially varying form, the same operation can be written as

$$I'(x, y) = a(x, y)I(x, y) + b(x, y),$$

  - allowing the gain and bias to change across the image.

Pixel transforms

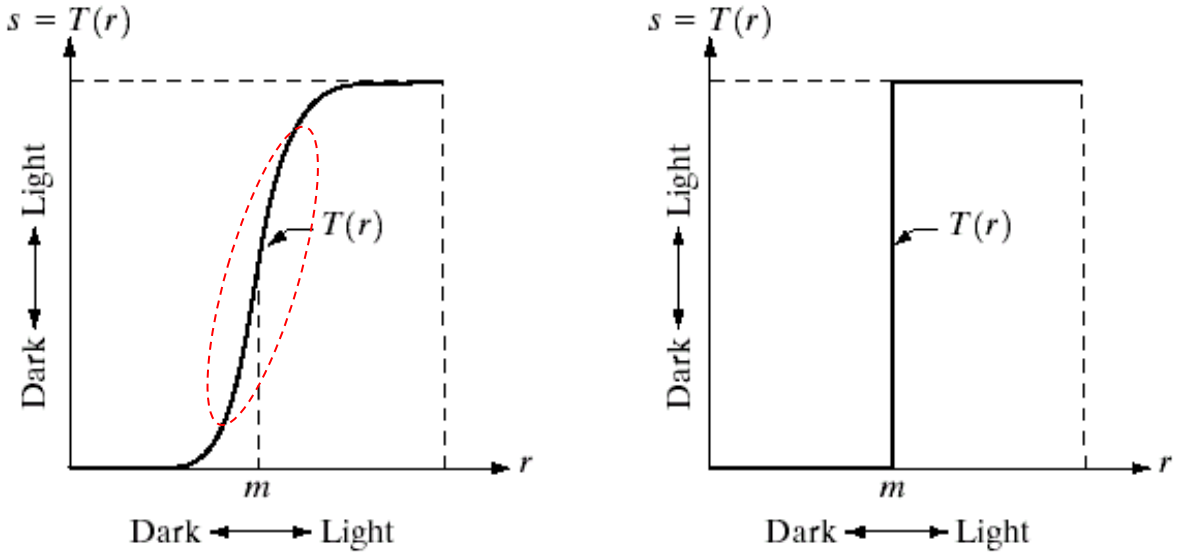


Pixel transforms

- Brightness and gain controls described can improve the appearance of an image
- How can we automatically determine their best values?
  - One approach might be to look at the darkest and brightest pixel values in an image and map them to pure black and pure white.
  - Another approach might be to find the average value in the image, push it towards middle gray, and expand the range so that it more closely fills the displayable values

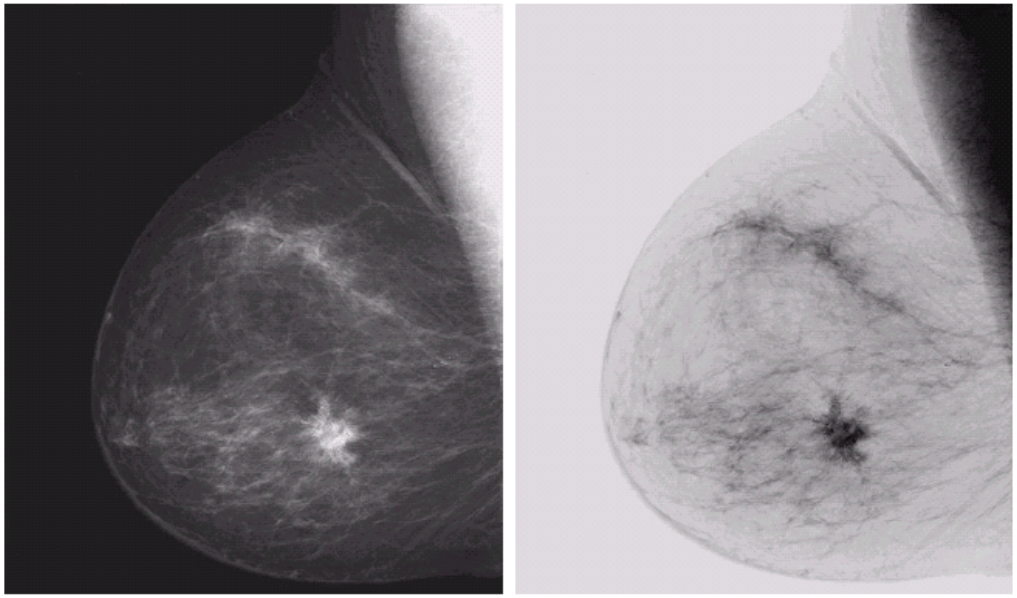
Pixel transforms

- **Contrast Stretching:** improves the contrast in an image by stretching the range of intensity values to span a desired range of values.



Pixel transforms

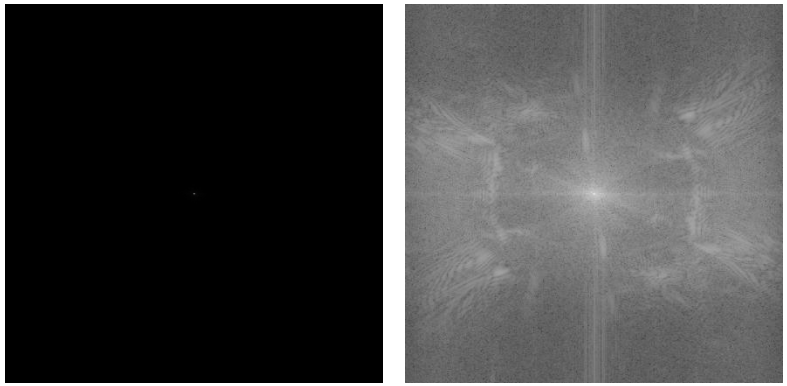
- **Negative transform:** Operation that produces the negative of an image ➔ Negative(I)=MaxIntensity−I



Pixel transforms

- **Logarithmic Transformations**

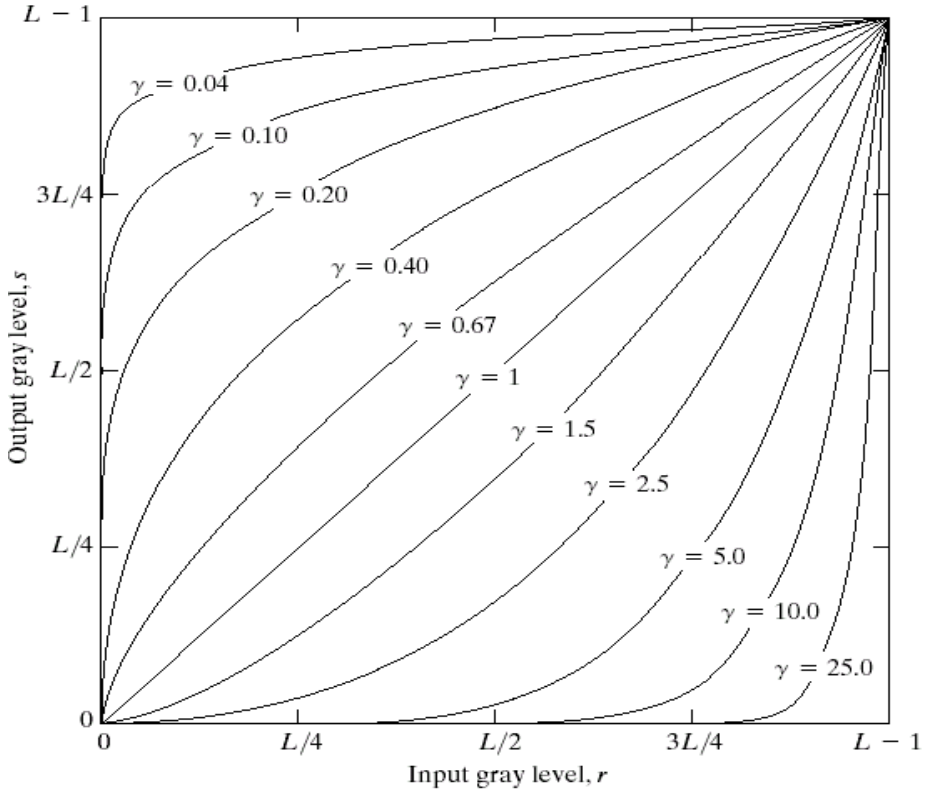
$$s = c \log (1 + r)$$



Pixel transforms

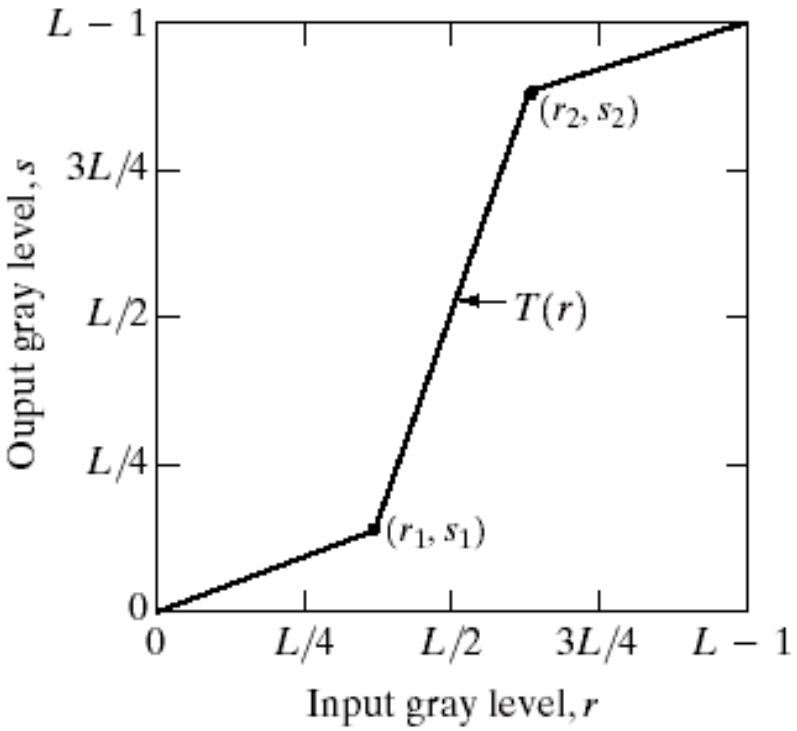
- **Logarithmic Transformations**

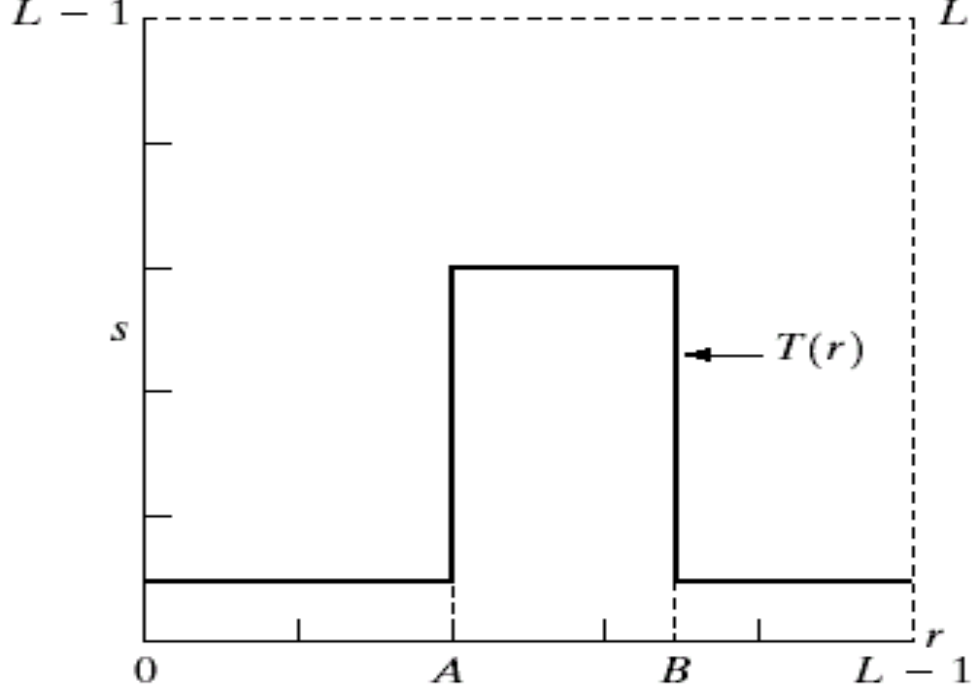
$$s = c\, r^{\gamma}$$

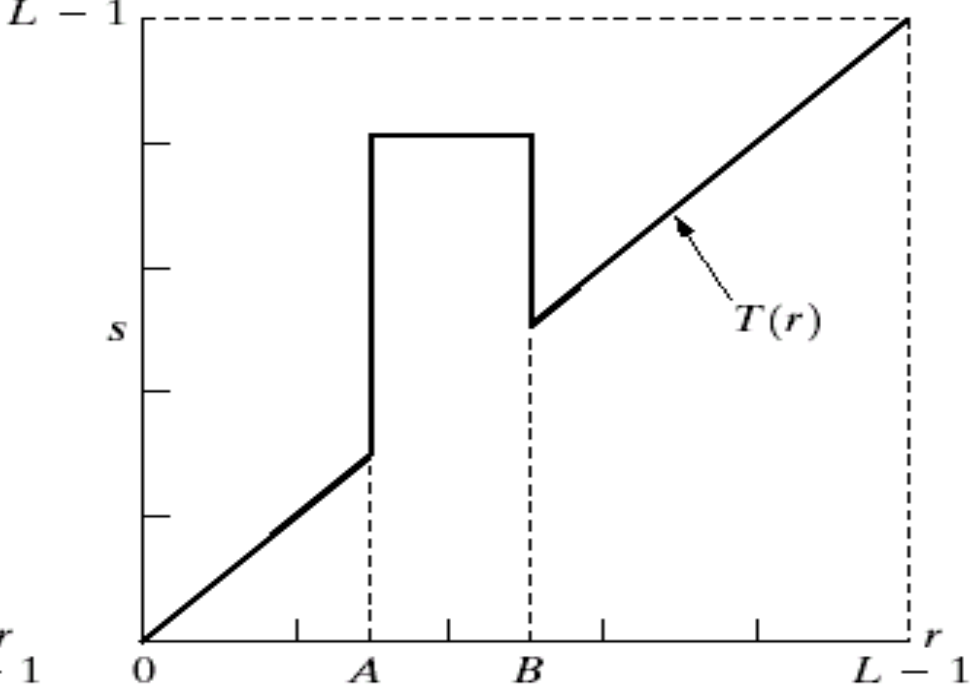


Carnegie Mellon University Africa

Pixel transforms



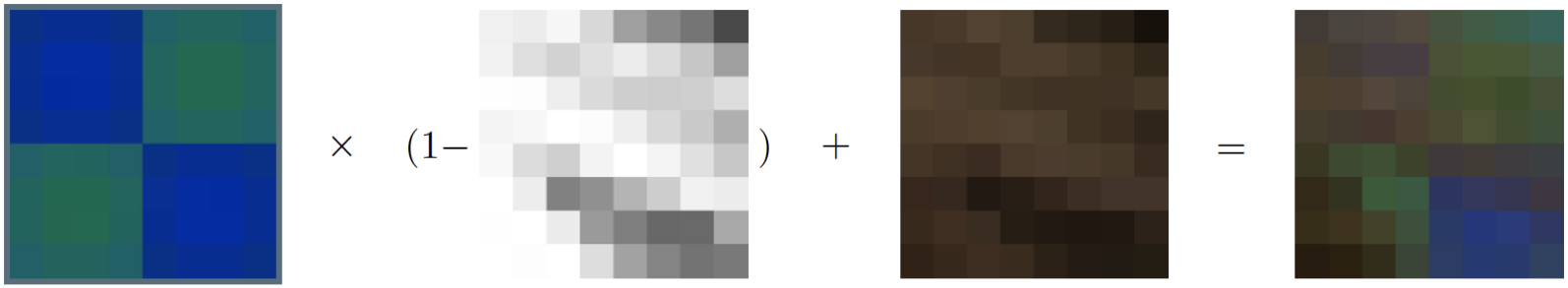

L − 1
s
T(r)
r
0
A
B
L − 1


L − 1
s
T(r)
r
0
A
B
L − 1

Compositing and matting

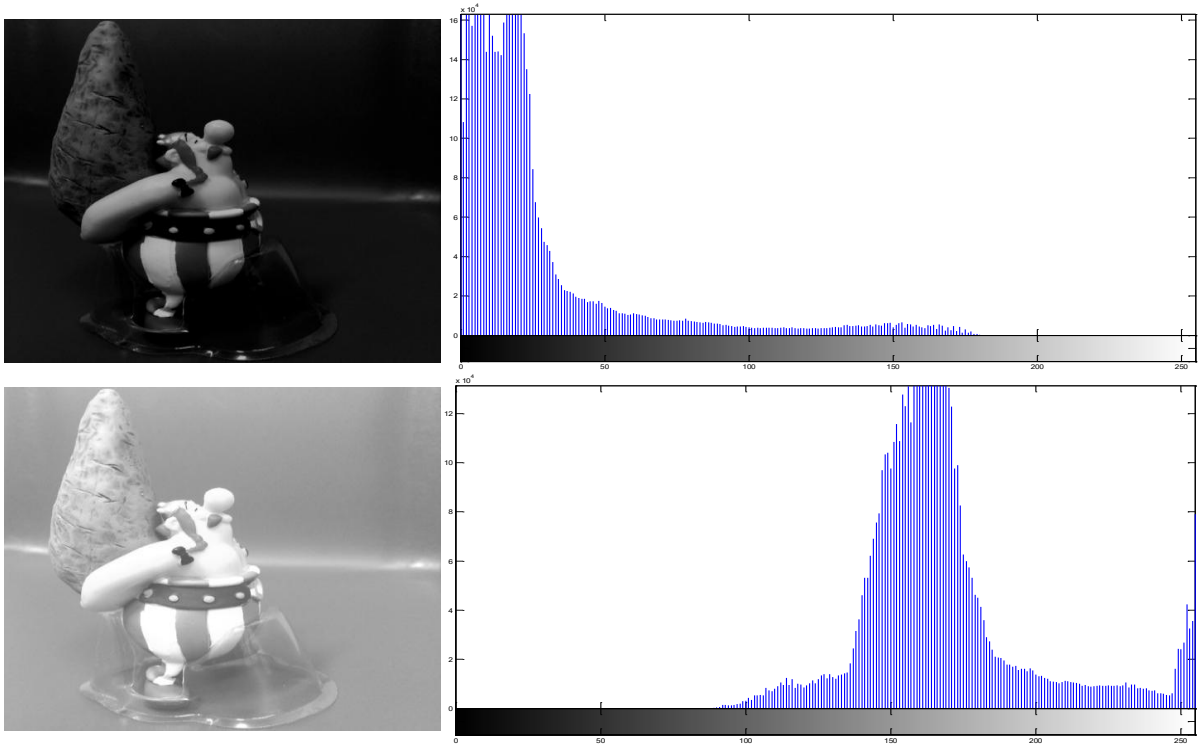
- This operator attenuates the influence of the background image B by a factor (1-α) and then adds in the color (and opacity) values corresponding to the foreground layer F

$$C = (1 - \alpha)B + \alpha F$$



Histogram equalization

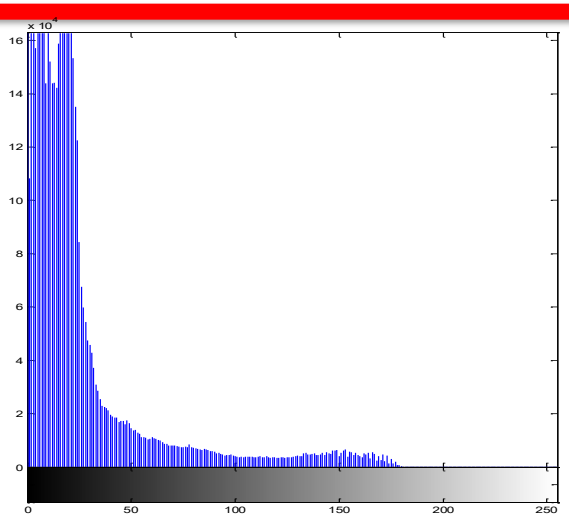
- A histogram provides a visual representation of the distribution of pixel intensities in an image

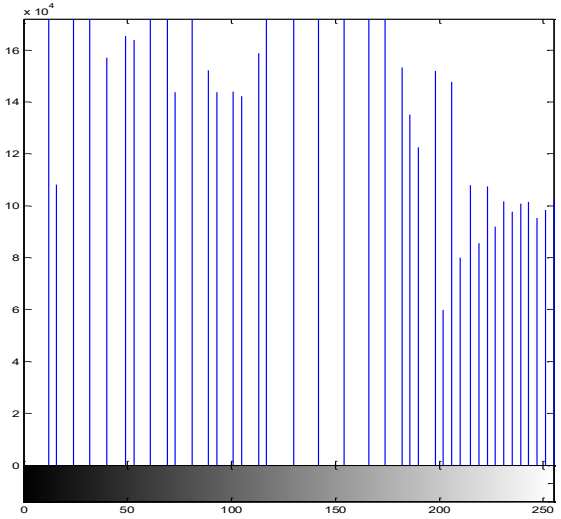


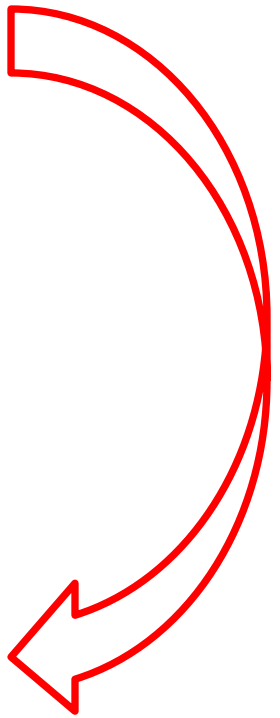
Histogram equalization

- A method which increases the dynamic range of the gray-levels in a low-contrast image to cover full range of gray-levels.

Histogram equalization



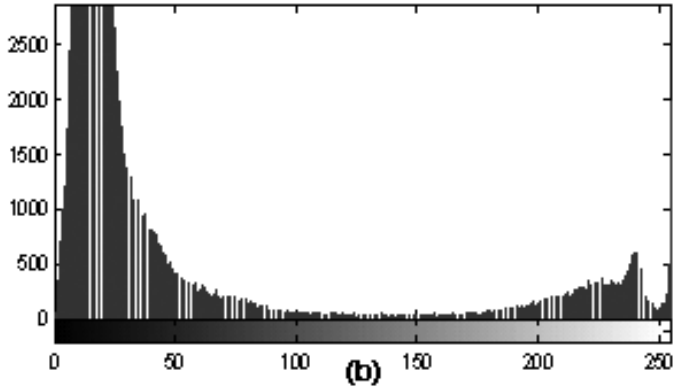




Histogram Equalization

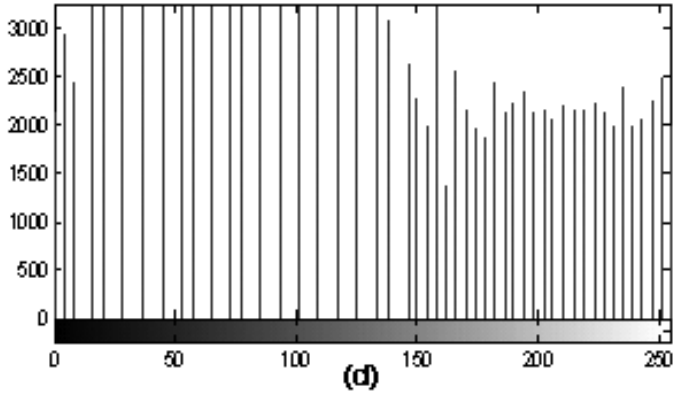
Histogram equalization

(a)



(b)

(c)



(d)

Histogram equalization

- Example:
  - 2D image ($\mathbf{4 \times 4}$), 3-bit grayscale ($L = 8$, levels 0-7)

$$- I = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 4 & 6 & 6 & 7 \end{bmatrix}, N = 16, L = 8$$

Histogram equalization

- Given a grayscale image with gray levels $k \in \{0,1, \ldots, L-1\}$:

$$p(k) = \frac{h(k)}{N}, F(k) = \sum_{j=0}^{k} p(j), T(k) = \lfloor (L-1)F(k) \rfloor$$

- Where:
  - k : gray-level value
  - $h(k)$: number of pixels with gray level k
  - N : total number of pixels in the image
  - $p(k)$: probability that a pixel has gray level k
  - $F(k)$: cumulative distribution function (CDF), i.e., fraction of pixels with intensity $\leq k$
  - L : total number of possible gray levels (e.g., $L = 256$ for 8 -bit images)
  - $\lfloor \cdot \rfloor$: floor operator (rounds down to nearest integer)
  - $T(k)$: histogram-equalized gray level assigned to pixels with original value k

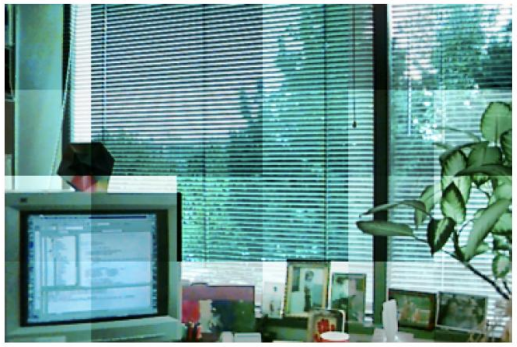
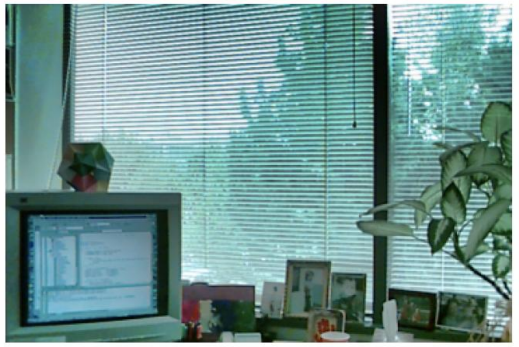
Histogram equalization

Gray Level k	Number of Pixels $h(k)$	PDF $p(k)$	CDF $F(k)$	Histogram Eq. Level $T(k) = \lfloor 7F(k) \rfloor$
0	2	0.1250	0.1250	0
1	3	0.1875	0.3125	2
2	3	0.1875	0.5000	3
3	2	0.1250	0.6250	4
4	3	0.1875	0.8125	5
5	0	0.0000	0.8125	5
6	2	0.1250	0.9375	6
7	1	0.0625	1.0000	7

$$\text{Original: } \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 4 & 6 & 6 & 7 \end{bmatrix} \leftrightarrow \text{Equalized: } I' = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 2 & 3 & 3 & 3 \\ 4 & 4 & 5 & 5 \\ 5 & 6 & 6 & 7 \end{bmatrix}$$

Histogram equalization

- Subdivide the image into M ×M pixel blocks and perform separate histogram equalization in each sub-block
- How to eliminate blocking artifacts:
  - Moving window
  - Adaptive histogram equalization

Linear filtering

- Neighborhood operator or local operator:
  - uses a collection of pixel values in the vicinity of a given pixel to determine its final output value

- Linear filtering operators:
  - involve fixed weighted combinations of pixels in small neighborhoods

(Kopf, Uyttendaele et al. 2007).

Linear filtering

- Neighborhood operator or local operator:
  - uses a collection of pixel values in the vicinity of a given pixel to determine its final output value

- Linear filtering operators:
  - involve fixed weighted combinations of pixels in small neighborhoods

(Kopf, Uyttendaele et al. 2007).

Linear filtering

- Linear filtering computes each output pixel as a linear combination of neighboring input pixels:

$$I'(x,y) = \sum_{i,j} h(i,j)I(x-i, y-j)$$

  - where $h(i,j)$ is a fixed filter kernel applied uniformly over the image.

- A filter kernel (mask) is a small matrix of weights defining the local neighborhood.
- Example (3×3 kernel):

$$h = \begin{bmatrix} h_{-1,-1} & h_{-1,0} & h_{-1,1} \\ h_{0,-1} & h_{0,0} & h_{0,1} \\ h_{1,-1} & h_{1,0} & h_{1,1} \end{bmatrix}$$

- Kernel size controls the spatial scale of filtering.

Linear filtering

- Gaussian filtering uses spatially weighted averaging:
- $h(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
- Parameter σ controls the degree of smoothing and scale of detail removal.
- For a Gaussian with standard deviation $\sigma \approx 1$, a common $\mathbf{3} \times \mathbf{3}$ discrete approximation is:

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Linear filtering

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

- output pixel's value is a weighted sum of pixel values within a small neighborhood N
  - $g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$

[Kopf, Uyttendaele et al. 2007].

Linear filtering

- The entries in the weight kernel or mask $h(k,l)$ are often called the filter coefficients.
- This can be more compactly notated as

$$g = f \otimes h.$$

- A common variant on this formula is

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l),$$

- where the sign of the offsets in f has been reversed, This is called the convolution operator,

$$g = f * h,$$

Both correlation and convolution are linear shift-invariant (LSI) operators, which obey both the superposition principle

Convolution

Convolution kernel, ω

1	-1	-1
1	2	-1
1	1	1

Rotate 180°

1	1	1
-1	2	1
-1	-1	1

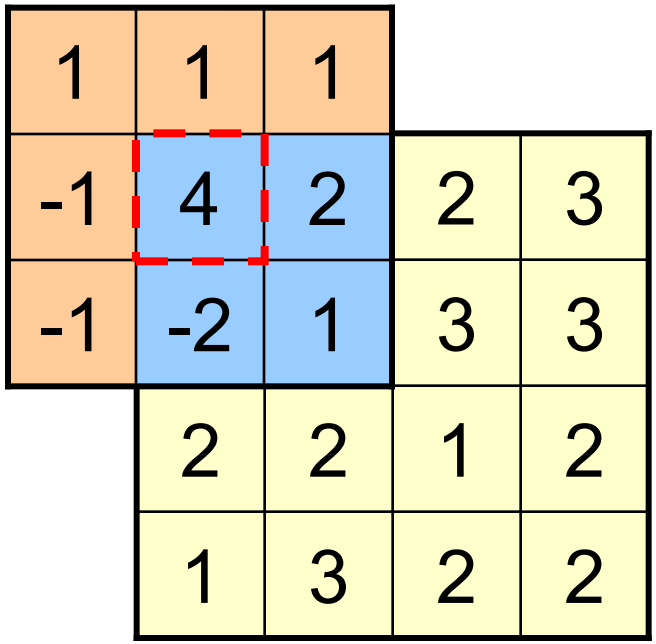
Input Image, f

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

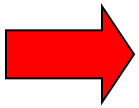
Convolution

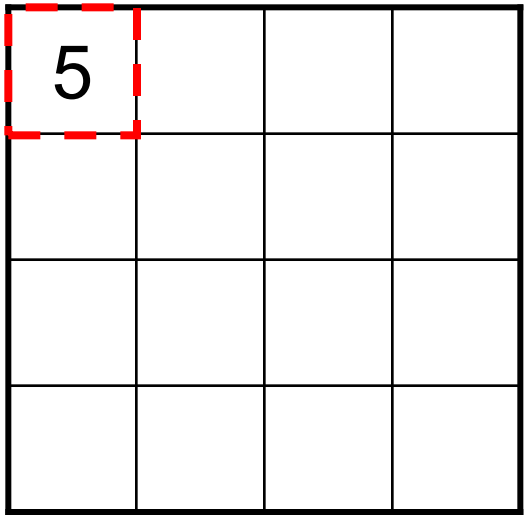
1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



Input Image, *f*



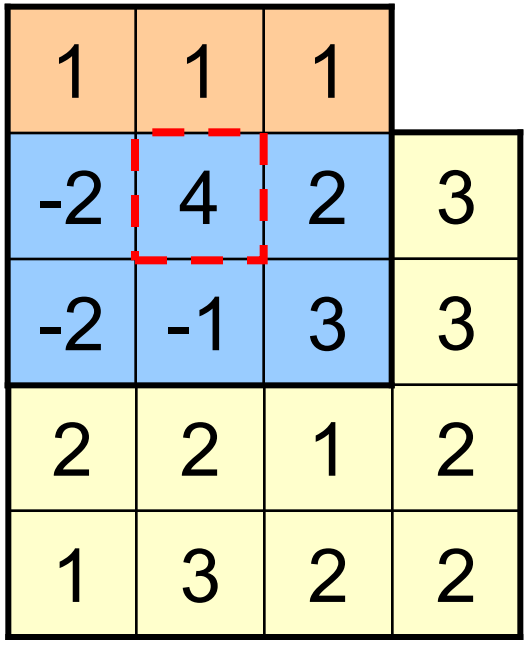


Output Image, *g*

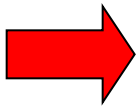
Convolution

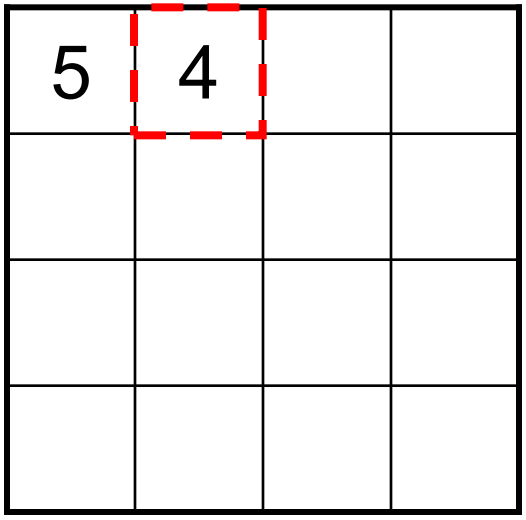
1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



Input Image, *f*





Output Image, *g*

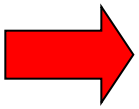
Convolution

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

	1	1	1
2	-2	4	3
2	-1	-3	3
2	2	1	2
1	3	2	2

Input Image, *f*



5	4	4	

Output Image, *g*

Convolution

1	1	1
-1	2	1
-1	-1	1

		1	1	1
2	2	-2	6	1
2	1	-3	-3	1
2	2	1	2	
1	3	2	2	

Input Image, *f*

5	4	4	-2

Output Image, *g*

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

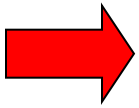
Convolution

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	2	2	2	3
-1	4	1	3	3
-1	-2	2	1	2
	1	3	2	2

Input Image, *f*



5	4	4	-2
9			

Output Image, *g*

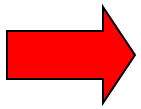
Convolution

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2

Input Image, *f*



5	4	4	-2
9	6		

Output Image, *g*

Convolution

5	4	4	-2
9	6	14	5
11	7	6	5
9	12	8	5

Final output Image, *g*

Padding (border effects)

- zero: set all pixels outside the source image to 0 (a good choice for alpha-matted cutout images);
- constant (border color): set all pixels outside the source image to a specified border value;
- clamp (replicate or clamp to edge): repeat edge pixels indefinitely;
- mirror: reflect pixels across the image edge;
- extend: extend the signal by subtracting the mirrored version of the signal from the edge pixel value.