Filtering and Pyramids

CSE P576

Dr. Matthew Brown

Filtering and Pyramids

- Linear filtering (convolution, correlation)
  - Blurring, sharpening, edge detection
- Gaussian and Laplacian Pyramids
  - Multi-scale representations

Linear Operators

- How are photo filters implemented?

original image

blur

sharpen

edge filter

Non-Linear Operators

- How are photo filters implemented?

original image

edge preserve smooth

median

canny edges

Correlation Example

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

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

=

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

element wise (dot) product

65	98	123
65	96	115
63	91	107

.*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$$= \begin{array}{l} 0.1*65+0.1*98+0.1*123+ \\ 0.1*65+0.2*96+0.1*115+ \\ 0.1*63+0.1*91+0.1*107 \end{array}$$

$$= 92$$

Correlation Example

$$*$$

$$=$$

- With colour images, perform the dot products over each band

Correlation

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

$$I(x, y)$$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$$k(x, y)$$

=

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

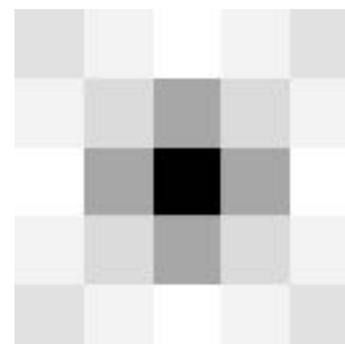
$$I_{cr}(x, y)$$

2.1

Correlation Example

- Centre-surround filter

*



=

59	81	82	104	139
52	77	93	112	133
69	96	100	110	124
89	115	100	118	124
96	118	118	132	141
75	105	112	136	154
63	99	130	147	145
59	114	140	151	142
58	132	145	149	142
58	131	146	140	131

... ...

*

0	2	3	2	0
2	0	-4	0	2
3	-4	-14	-4	3
2	0	-4	0	2
0	2	3	2	0

=

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

... ...

Correlation Example

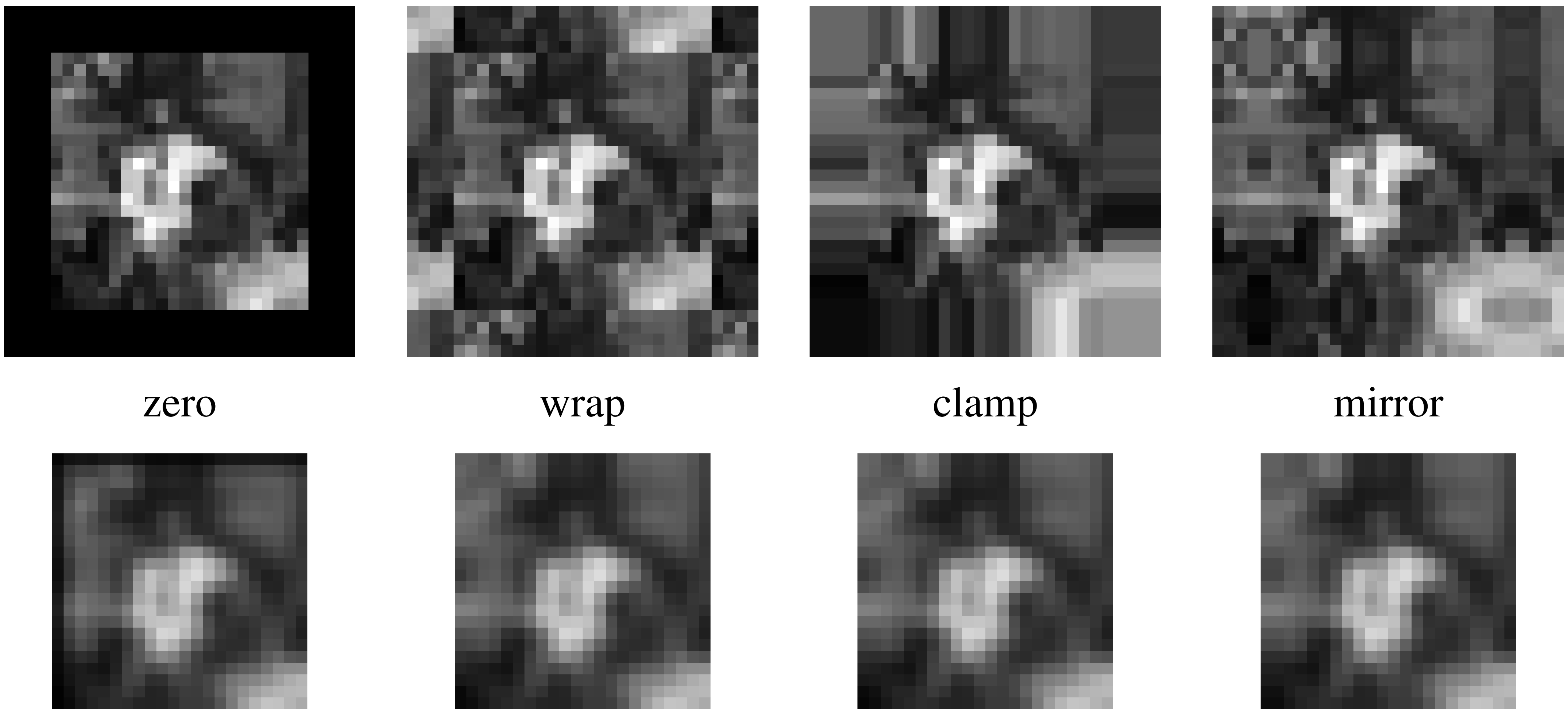
- Edge effects

$$* \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix} =$$

- To maintain the image size, we can **pad** the input by adding boundary pixels
- In this example the input has been **zero padded**

Padding

- What happens to pixels that overlap the boundary?



zero | wrap | clamp | mirror

blurred zero | normalized zero | blurred clamp | blurred mirror

"zero" and "clamp" (also called zero-order hold) are common in vision applications

Correlation and Convolution

- Correlation

$$I(x,y) \text{ corr } k(x,y) = \int_t \int_s I(x+s, y+t) k(s,t) \, ds \, dt$$

- Convolution

$$I(x,y) \ * \ k(x,y) = \int_t \int_s I(x-s, y-t) k(s,t) \, ds \, dt$$

 2.2

For symmetric kernels, correlation == convolution

Point Spread Function

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

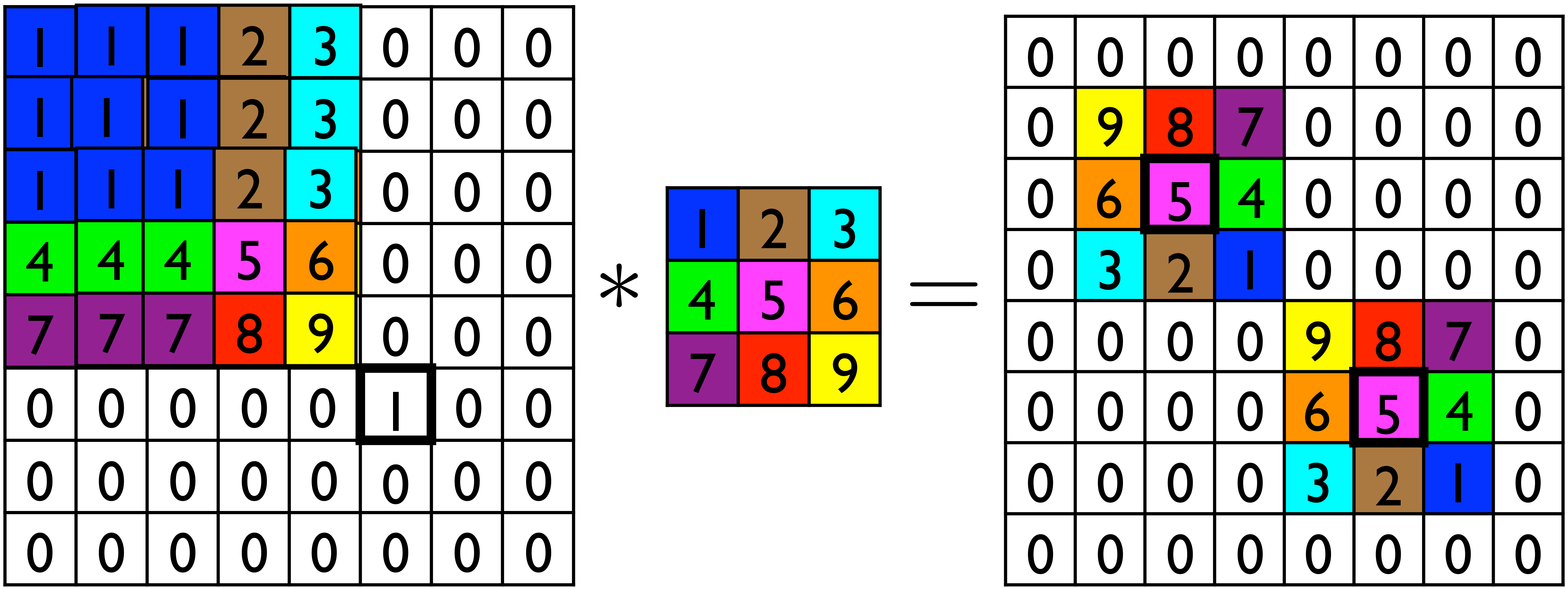
*

1	2	3
4	5	6
7	8	9

=

0	0	0	0	0	0	0	0
0	9	8	7	0	0	0	0
0	6	5	4	0	0	0	0
0	3	2	1	0	0	0	0
0	0	0	0	9	8	7	0
0	0	0	0	6	5	4	0
0	0	0	0	3	2	1	0
0	0	0	0	0	0	0	0

Point Spread Function

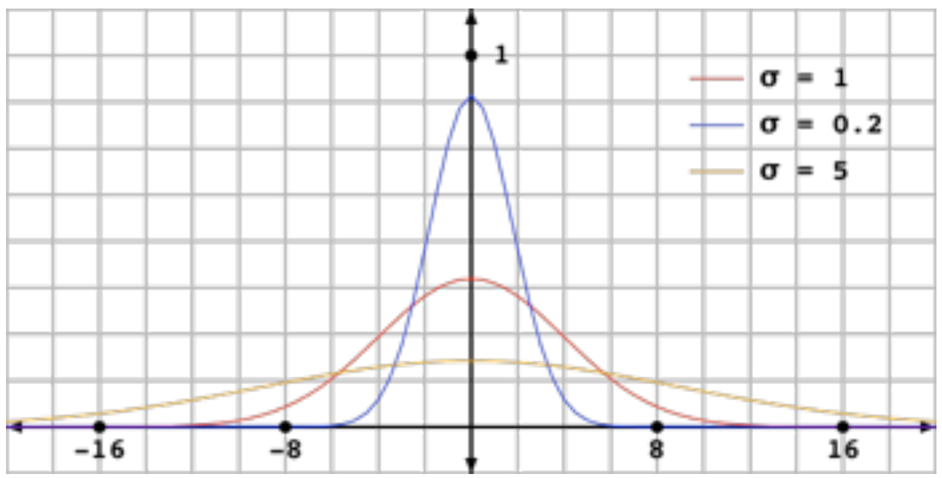


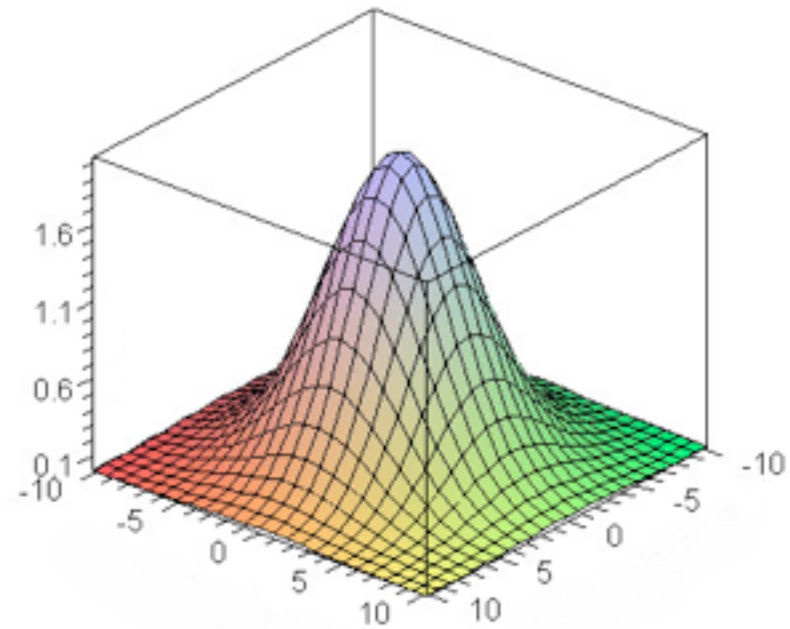
- The point spread function is the correlation kernel rotated by 180° (= the convolution kernel)

Gaussian Blur

- Gaussian kernels are often used for smoothing

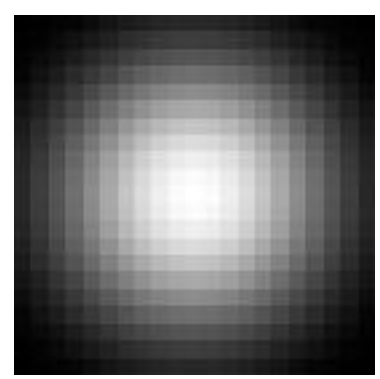
1D





2D

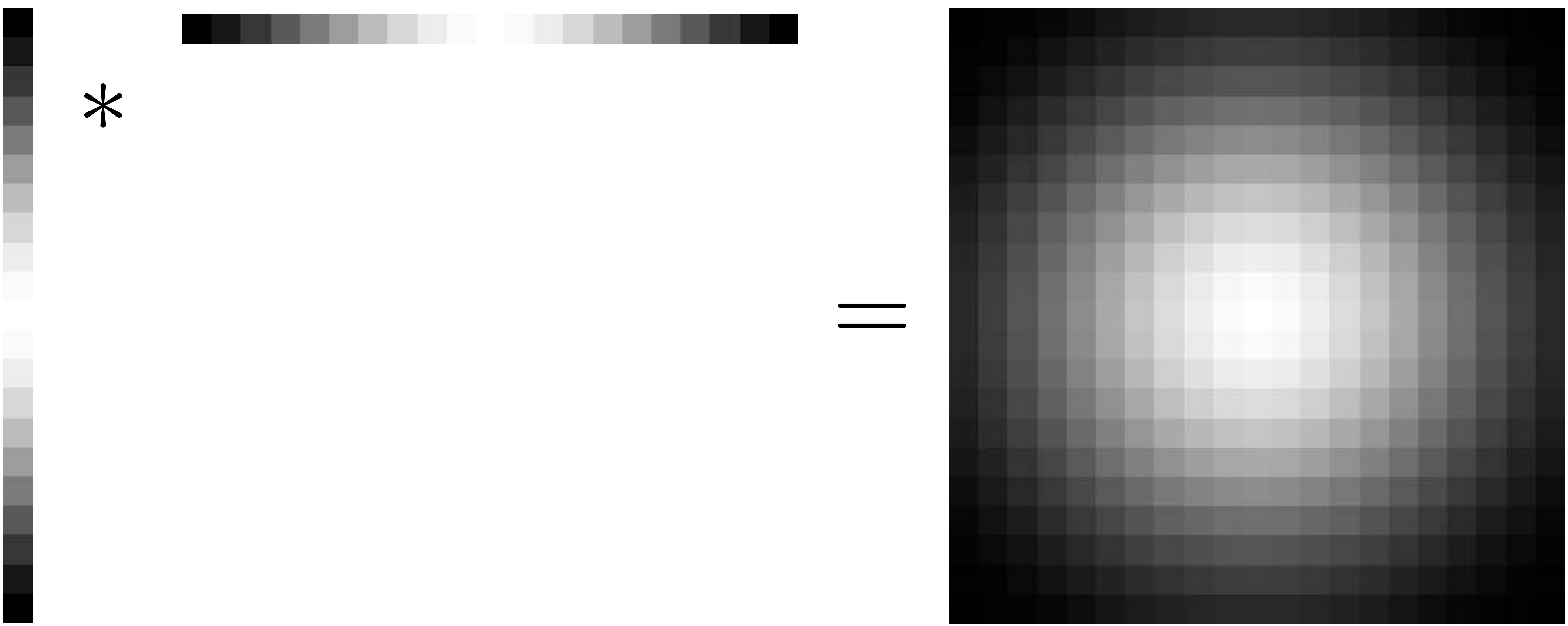
*



=

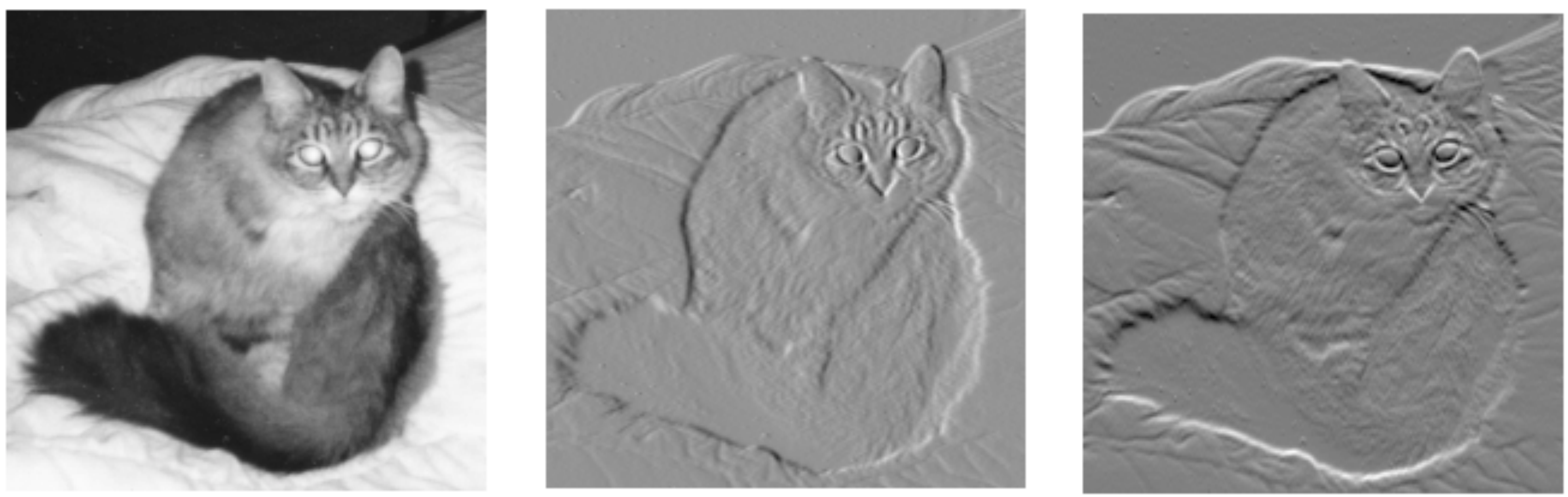
Gaussian Blur

- 2D Gaussian filter is a product of row and column filters



Edge Filtering

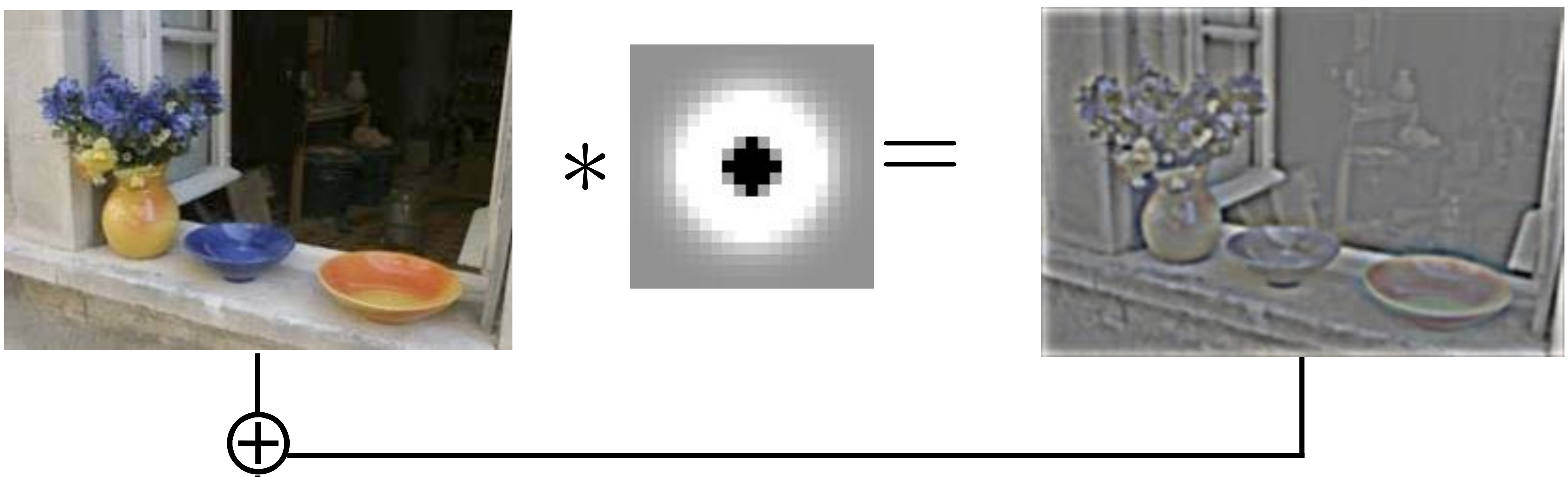
- Gradients can be computed using a finite difference approximation to the derivative, e.g., $g_x = I_{x+1} - I_x$



g_x g_y

Centre Surround Filter

- Useful for extracting features at a certain **scale**



We can implement a **sharpening** filter by adding a multiple of this high-frequency band back to the image

Properties of Convolution

- Linear + associative, commutative

2.3

Separable Filtering

- 2D Gaussian blur by horizontal/vertical blur

horizontal

vertical

vertical

horizontal

2.4

Separable Filtering

- Several useful filters can be applied as independent row and column operations

(a) box, $K = 5$

$\frac{1}{K^2}$

1	1	$\cdots$	1
1	1	$\cdots$	1
$\vdots$	$\vdots$	1	$\vdots$
1	1	$\cdots$	1

$\frac{1}{K}$

1	1	$\cdots$	1

(b) bilinear

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$\frac{1}{4}$

1	2	1

(c) "Gaussian"

$\frac{1}{256}$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

$\frac{1}{16}$

1	4	6	4	1

(d) Sobel

$\frac{1}{8}$

−1	0	1
−2	0	2
−1	0	1

$\frac{1}{2}$

−1	0	1

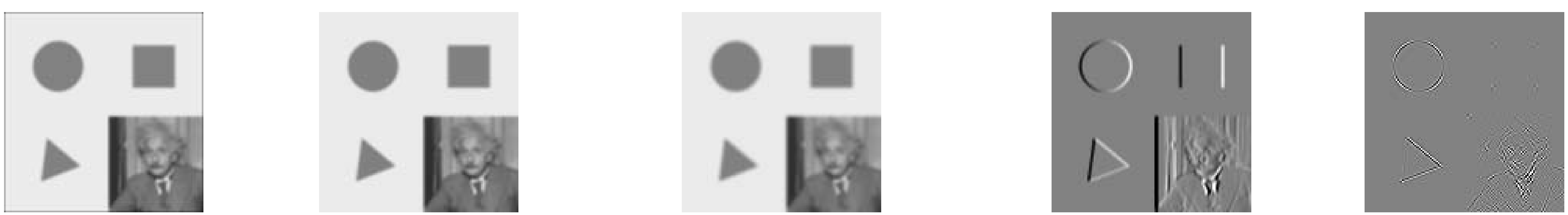
(e) corner

$\frac{1}{4}$

1	−2	1
−2	4	−2
1	−2	1

$\frac{1}{2}$

1	−2	1



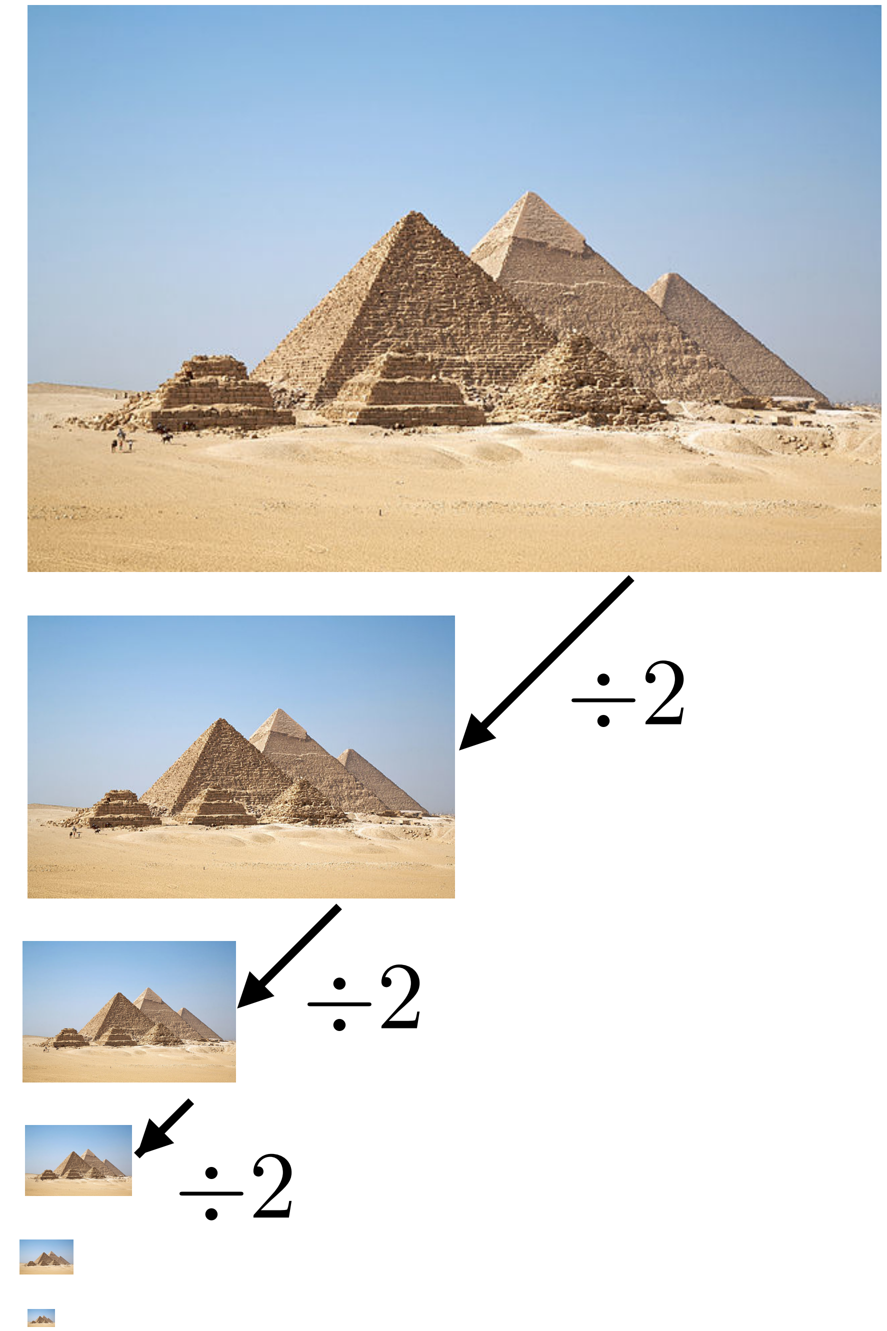
(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

Project 1

P1

- You are now ready to try the **Convolution and Image Filtering** section in Project 1
- `convolve_1d` : Implement 1D convolution. Hint: pad the input with zeros to avoid border cases.
- `convolve_gaussian` : you can transpose a kernel to flip horizontal/vertical, but make sure it is a 2D numpy array - use `np.expand_dims` if not

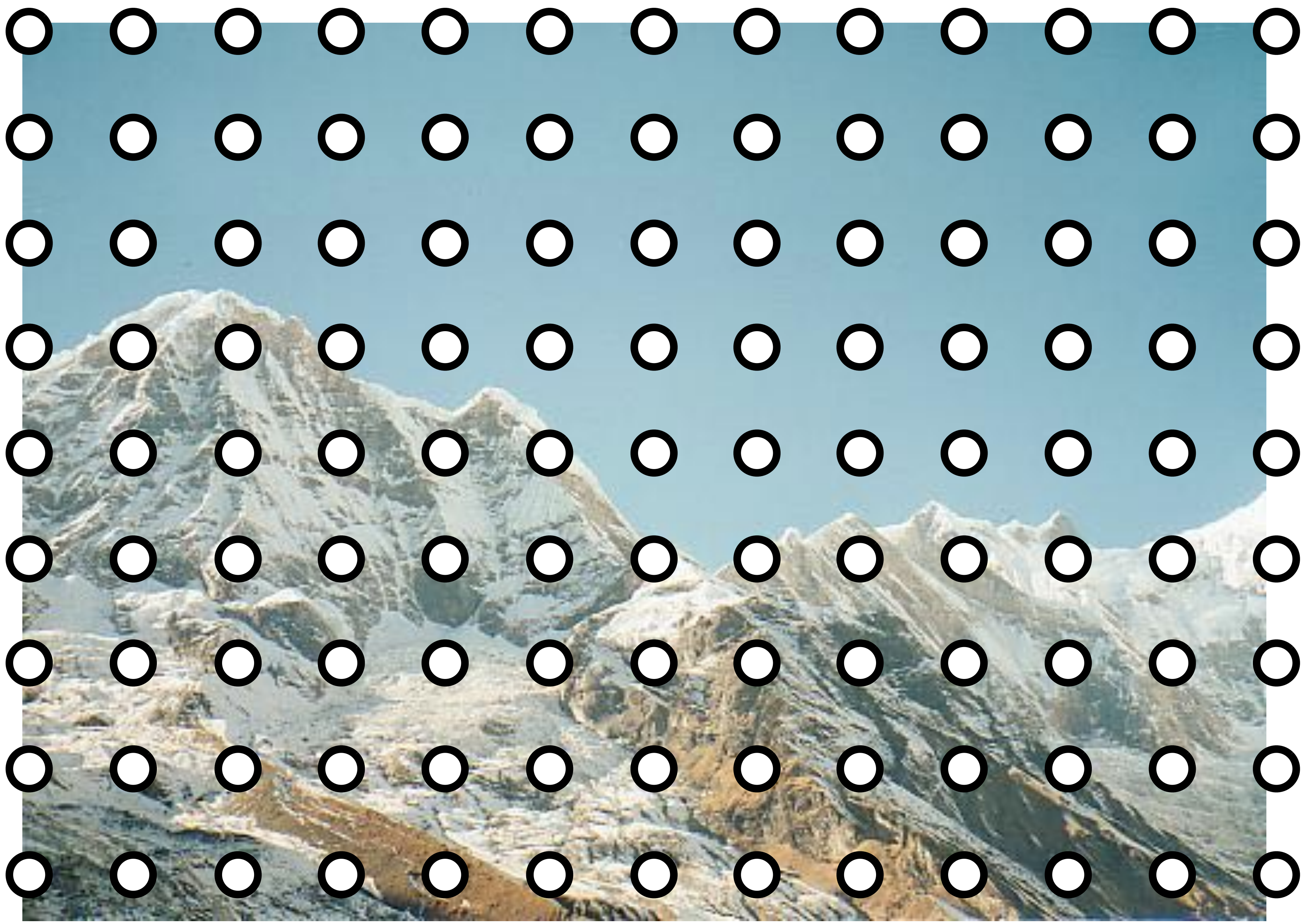
Image Pyramids



Used in Graphics (Mip-map) and Vision
(for **multi-scale** processing)

Resizing Images

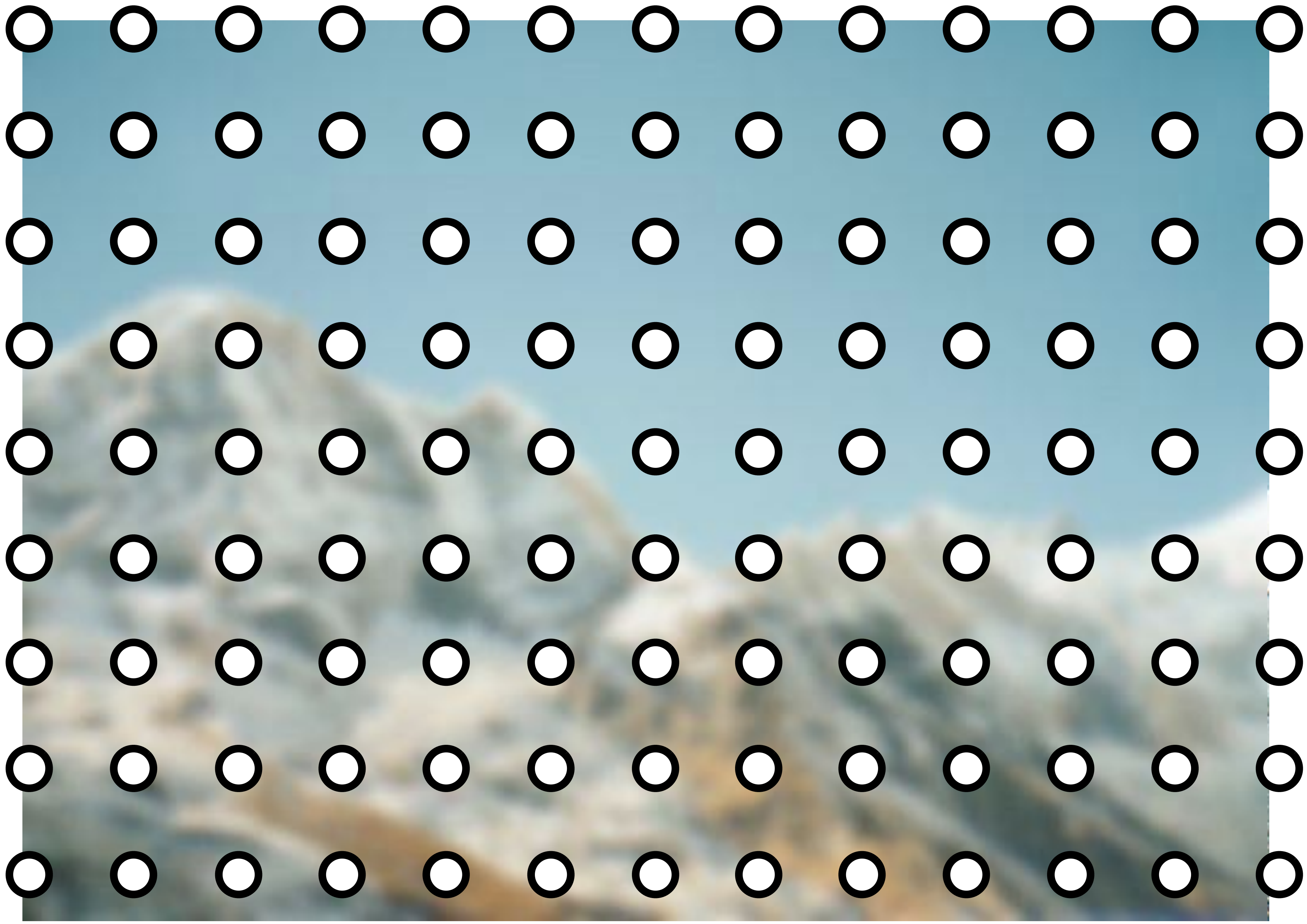
- Naive method: form new image by selecting every *n*th pixel



What is wrong with this method?

Resizing Images

- Improved method: first **blur** the image (low pass filter)



With the correct filter, no information is lost (Nyquist)

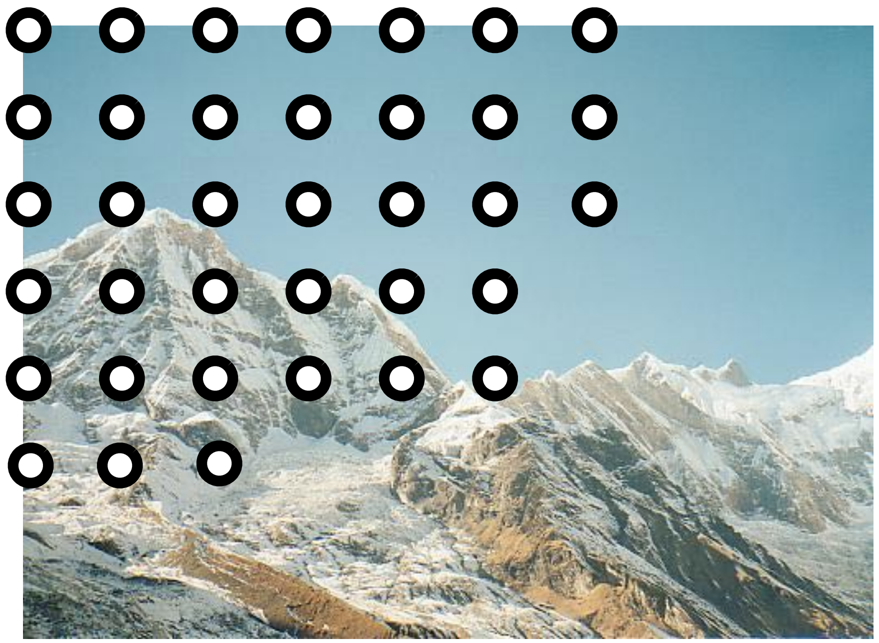
Aliasing Example

- Sampling every 5th pixel, with and without low pass filtering

No filtering

Gaussian Blur $\sigma = 3.0$

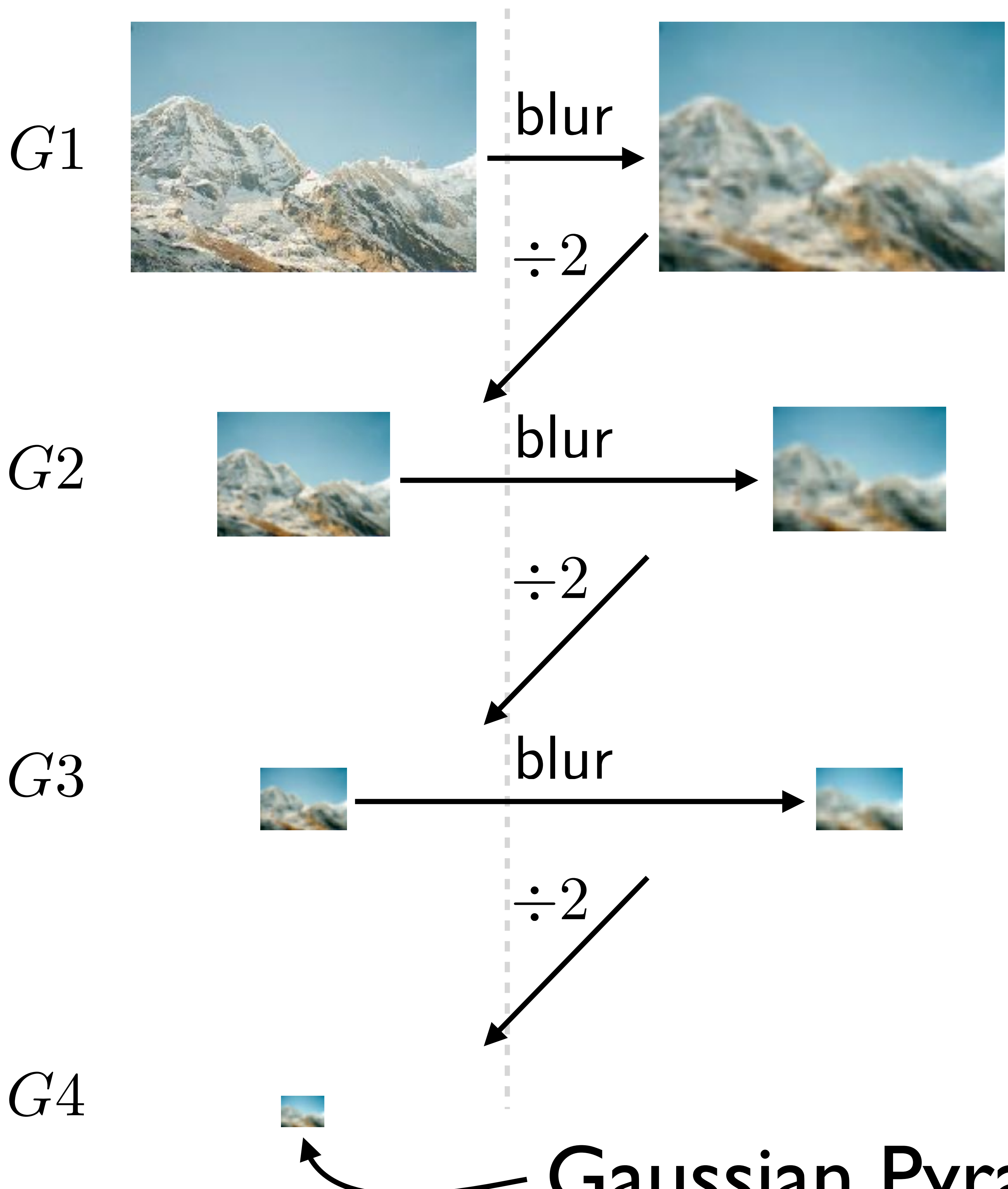
Resizing Images



every 10th pixel
(aliased)

low pass filtered
(correct sampling)

- Note that selecting every 10th pixel ignores the intervening information, whereas the low-pass filter (blur) smoothly combines it
- If we shifted the original image 1 pixel to the right, the aliased image would look completely different, but the the low pass filtered image would look almost the same



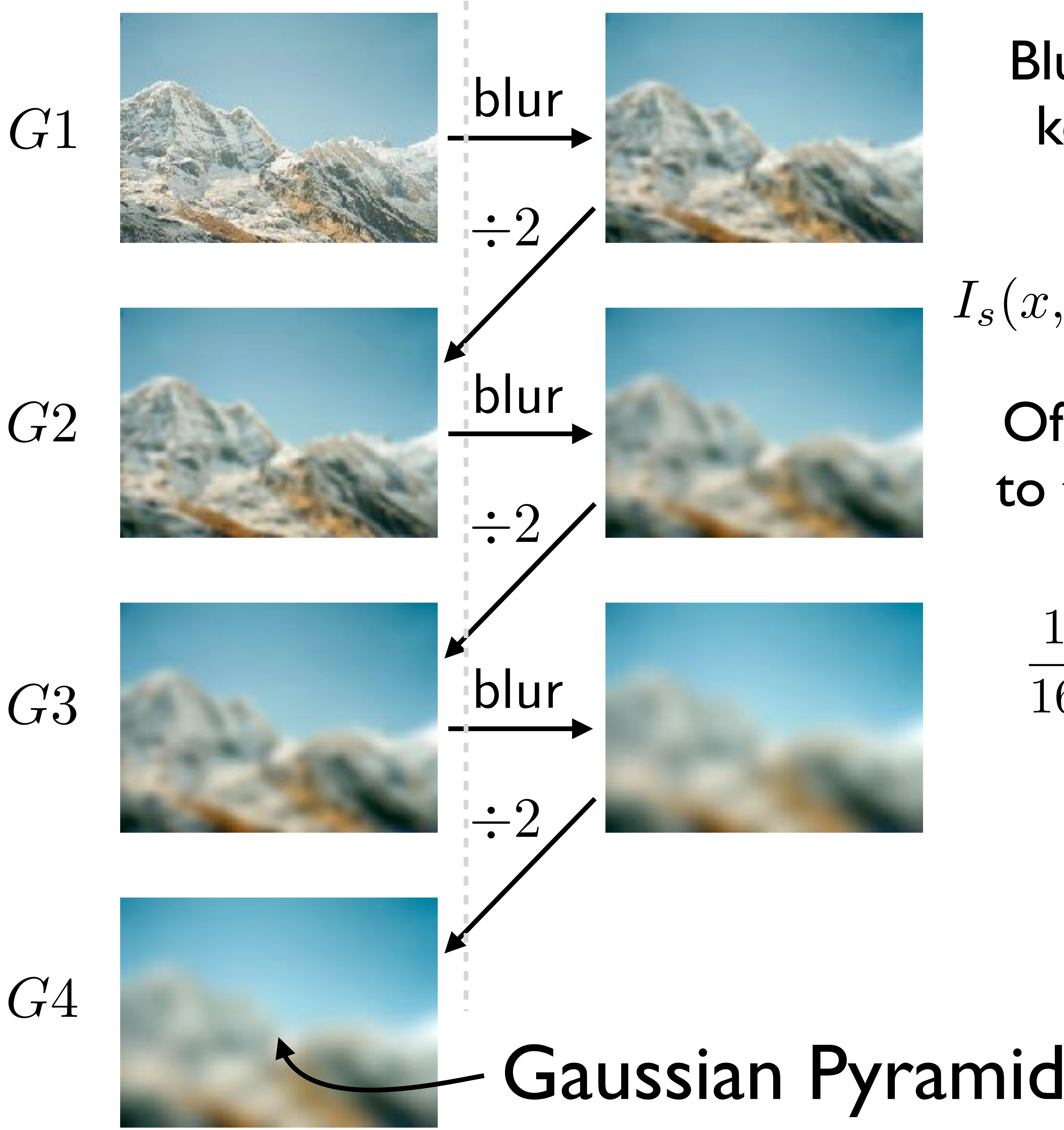
Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$

Often approximations to the Gaussian kernel are used, e.g.,

$$\frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Gaussian Pyramid



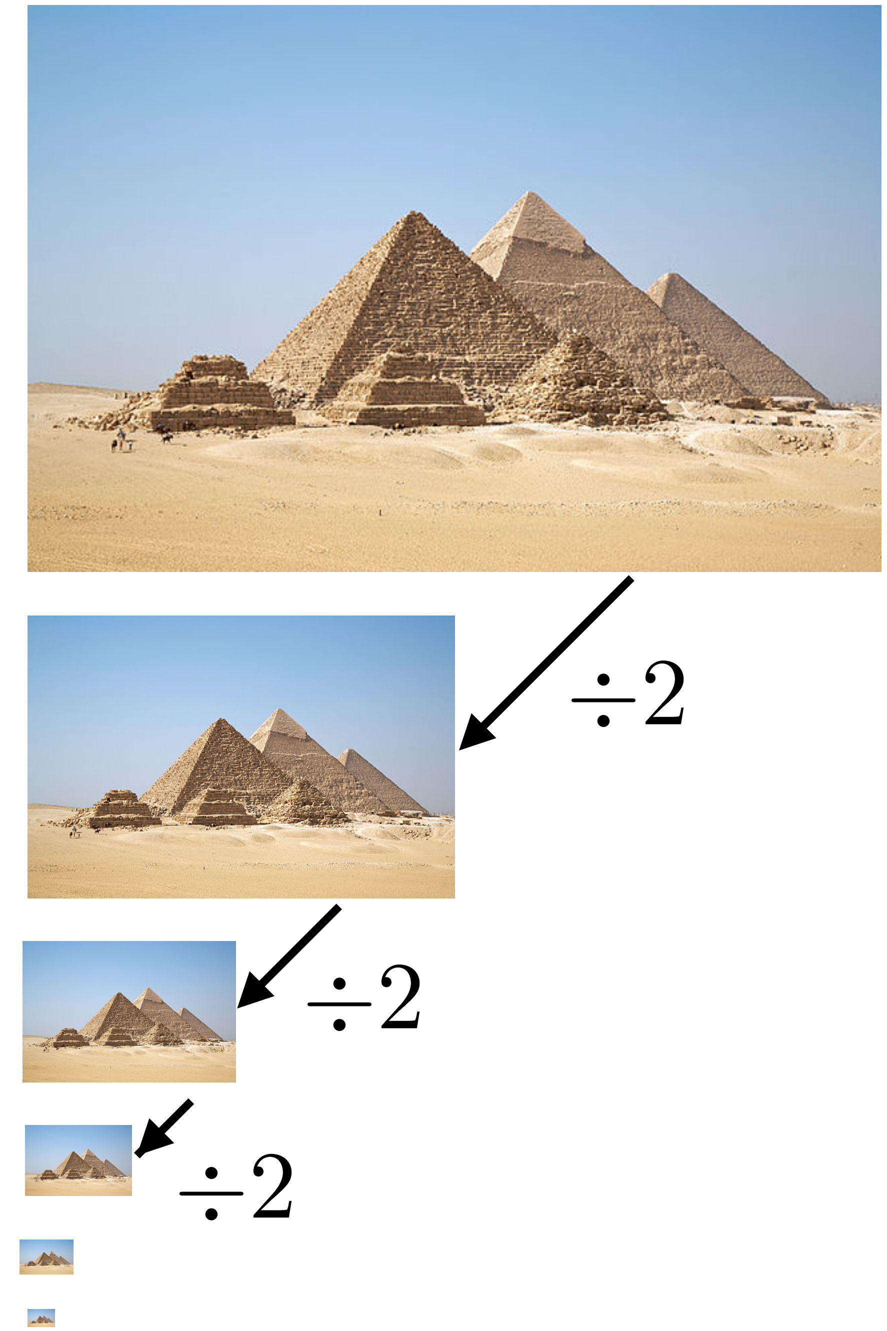
Blur with a Gaussian kernel, then select every 2nd pixel

$$I_s(x, y) = I(x, y) * g_\sigma(x, y)$$

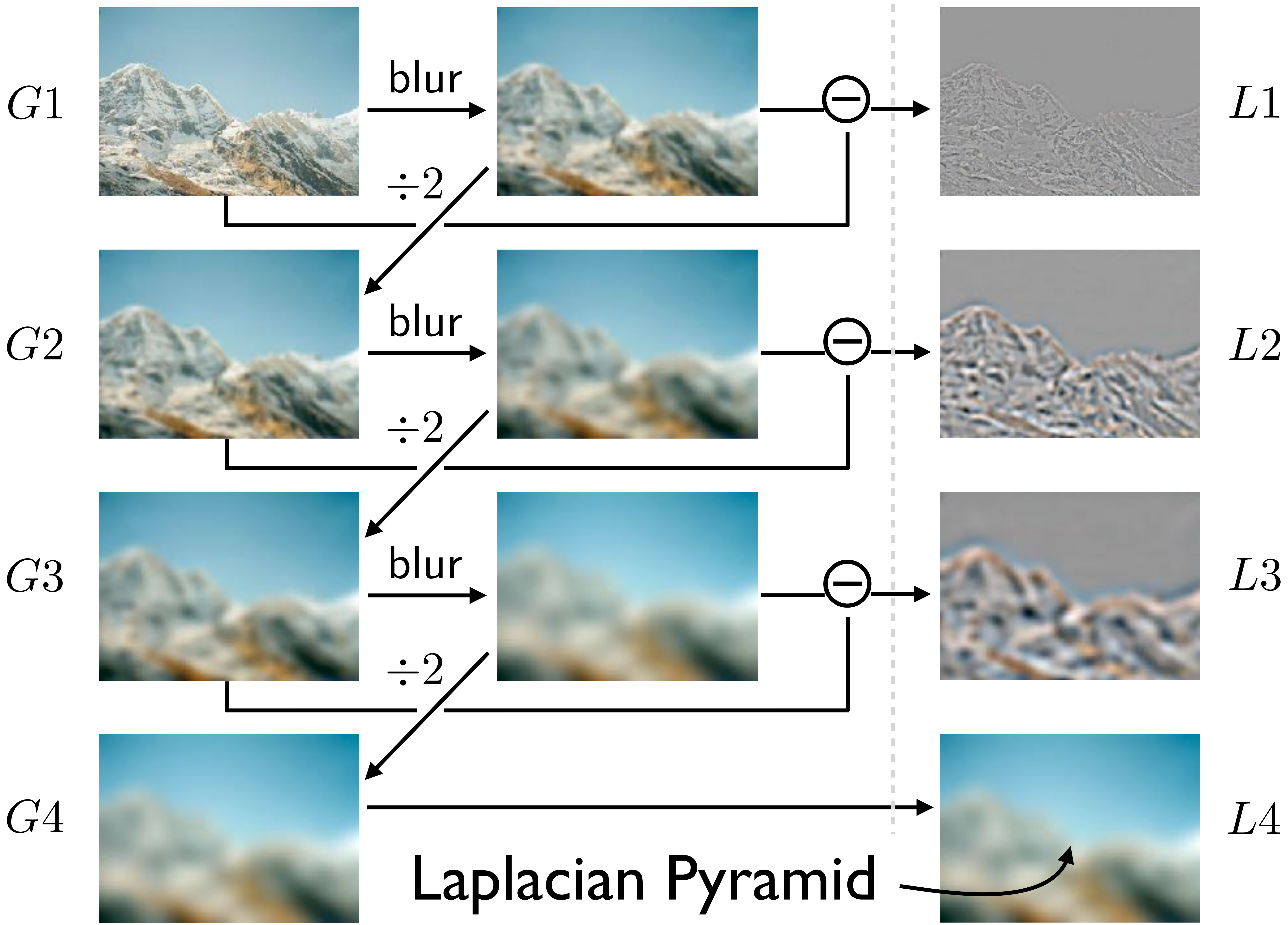
Often approximations to the Gaussian kernel are used, e.g.,

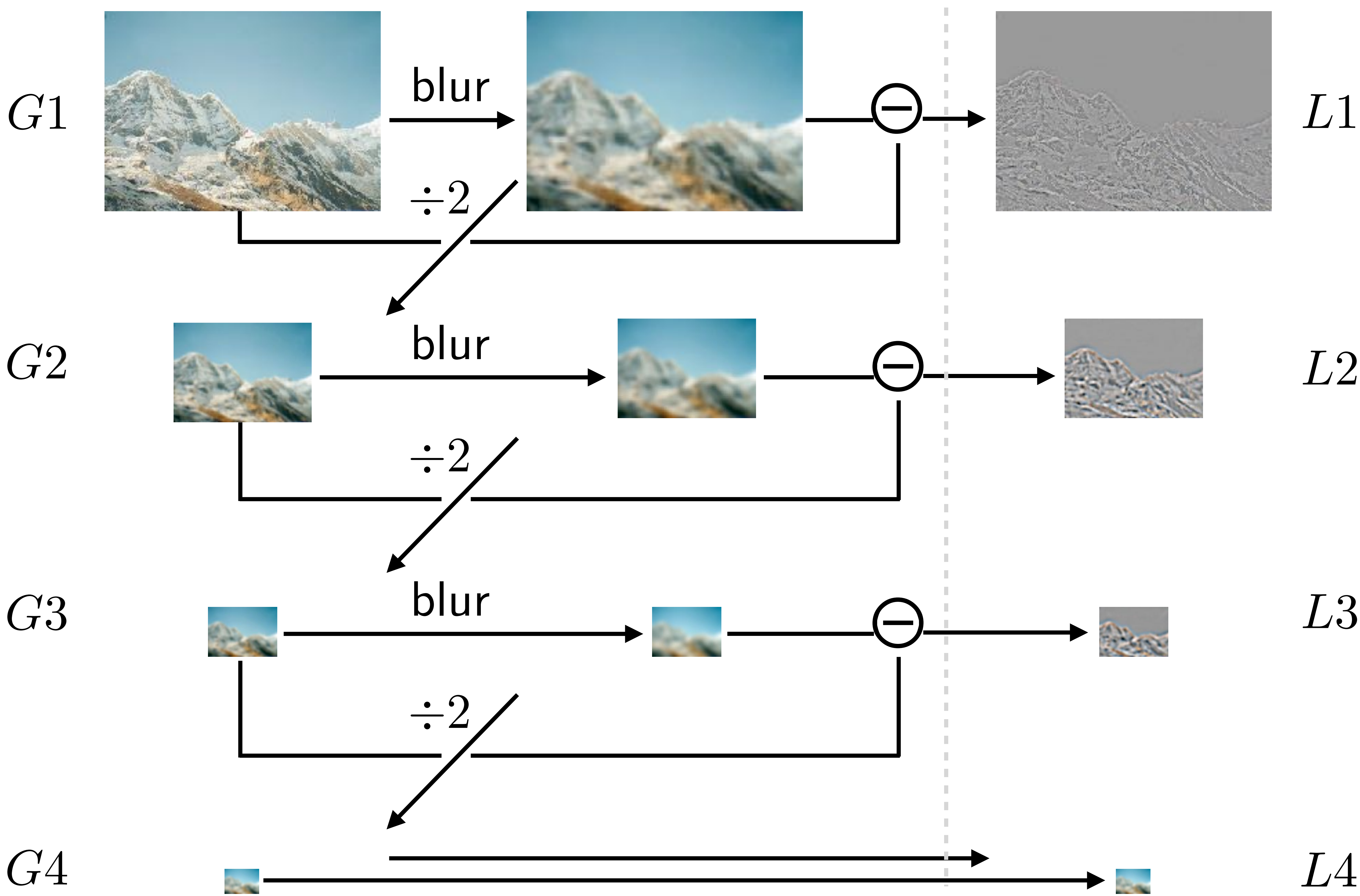
$$\frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Sampling with Pyramids

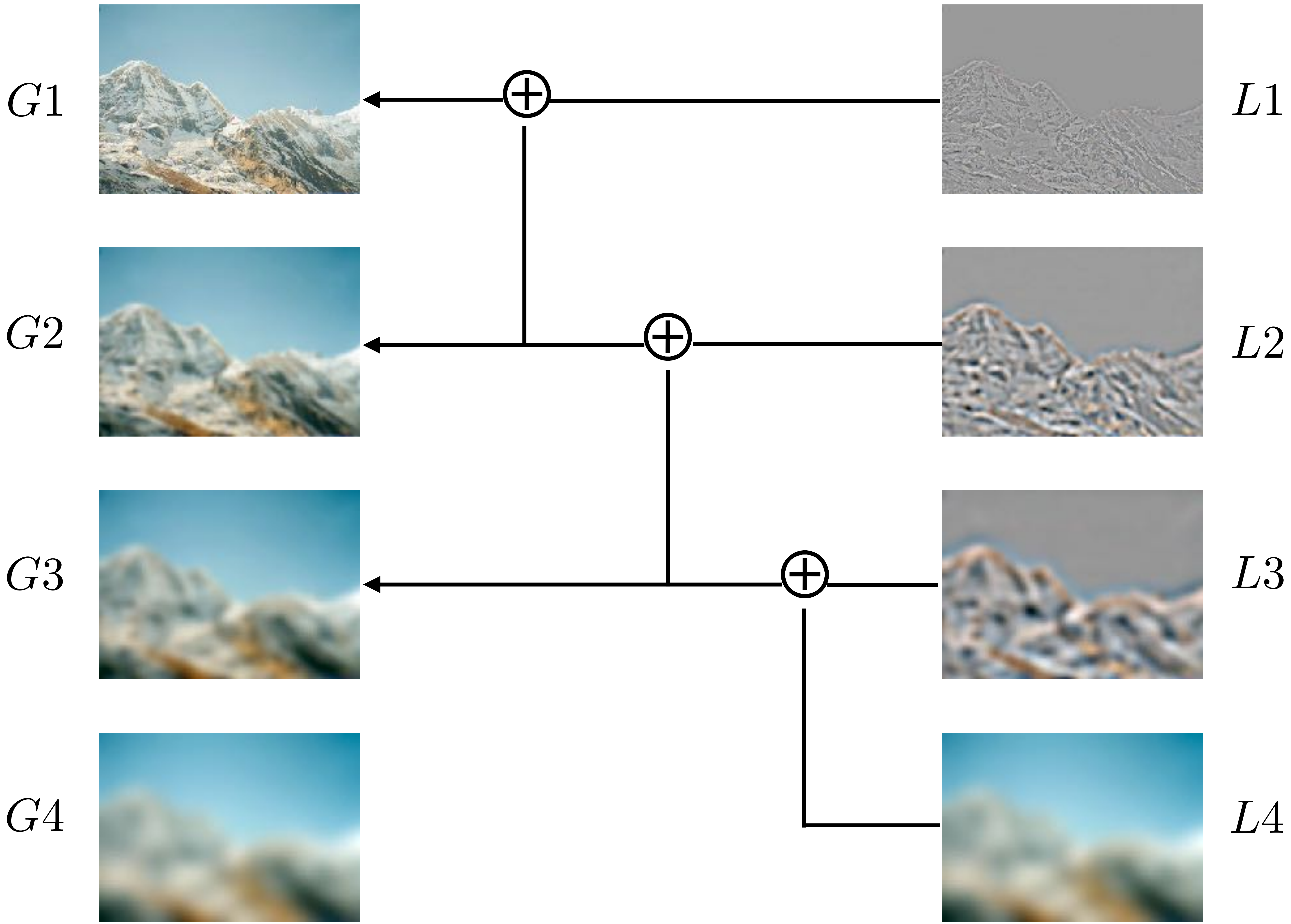


Find the level where the sample spacing is between 1 and 2 pixels, apply extra fraction of inter-octave blur as needed


$G1$
blur
$\div 2$
$L1$
$G2$
blur
$\div 2$
$L2$
$G3$
blur
$\div 2$
$L3$
$G4$
$L4$
Laplacian Pyramid

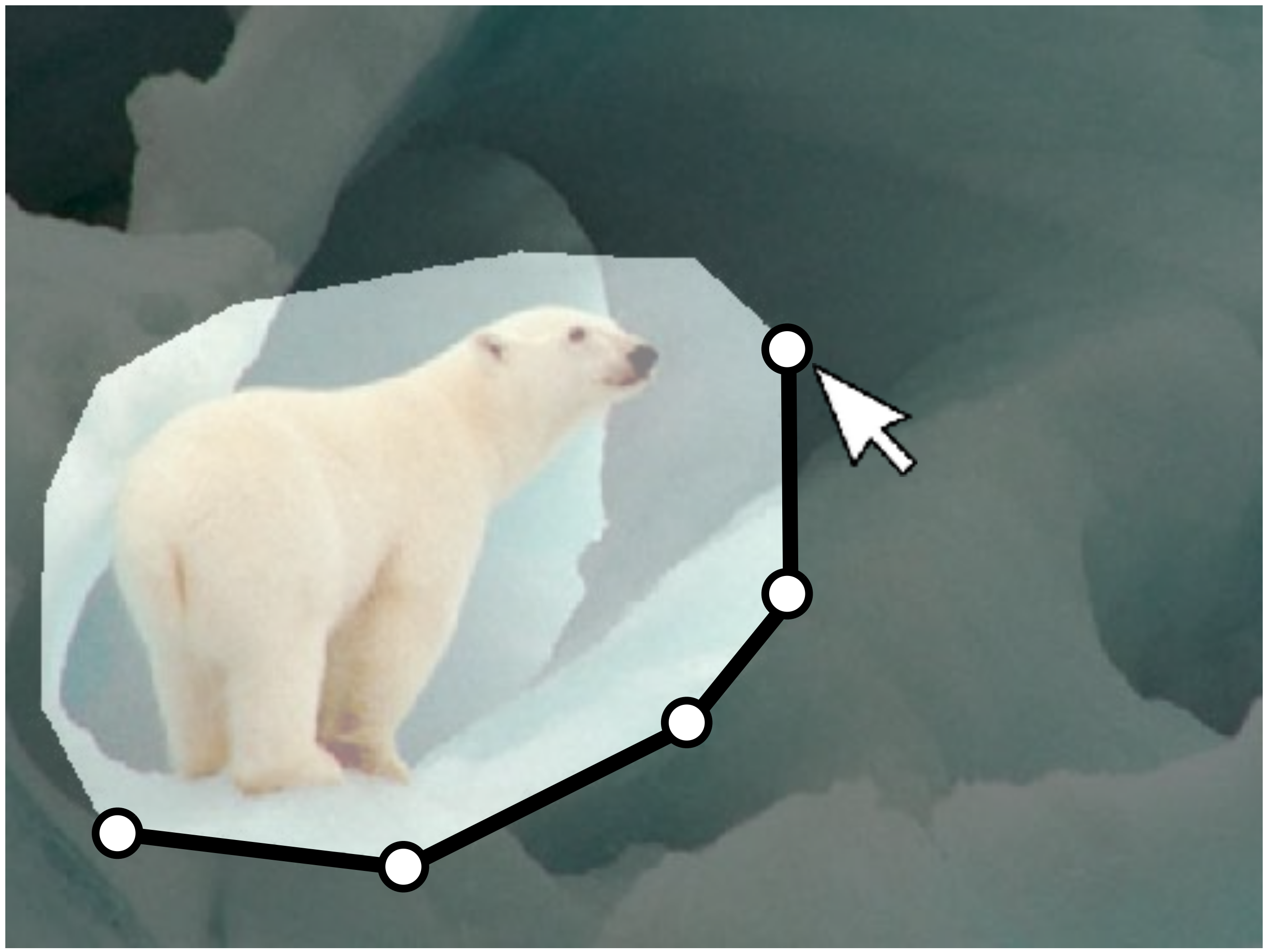


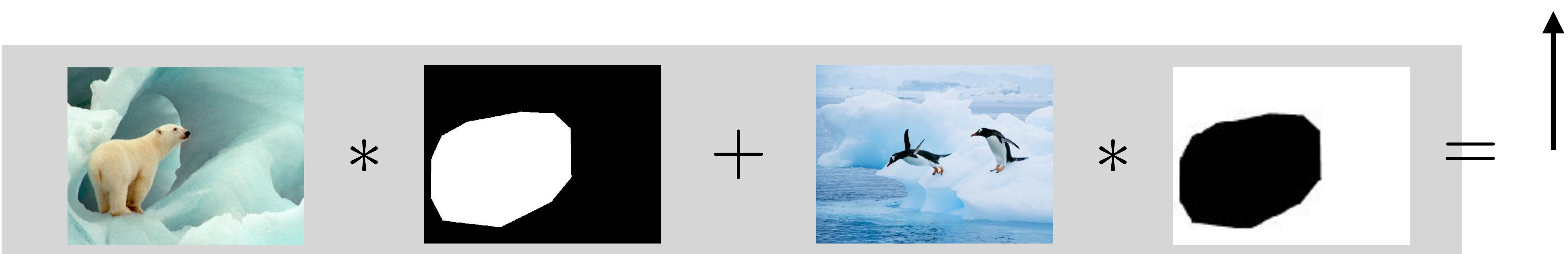
Laplacian Pyramid


G1
G2
G3
G4
L1
L2
L3
L4

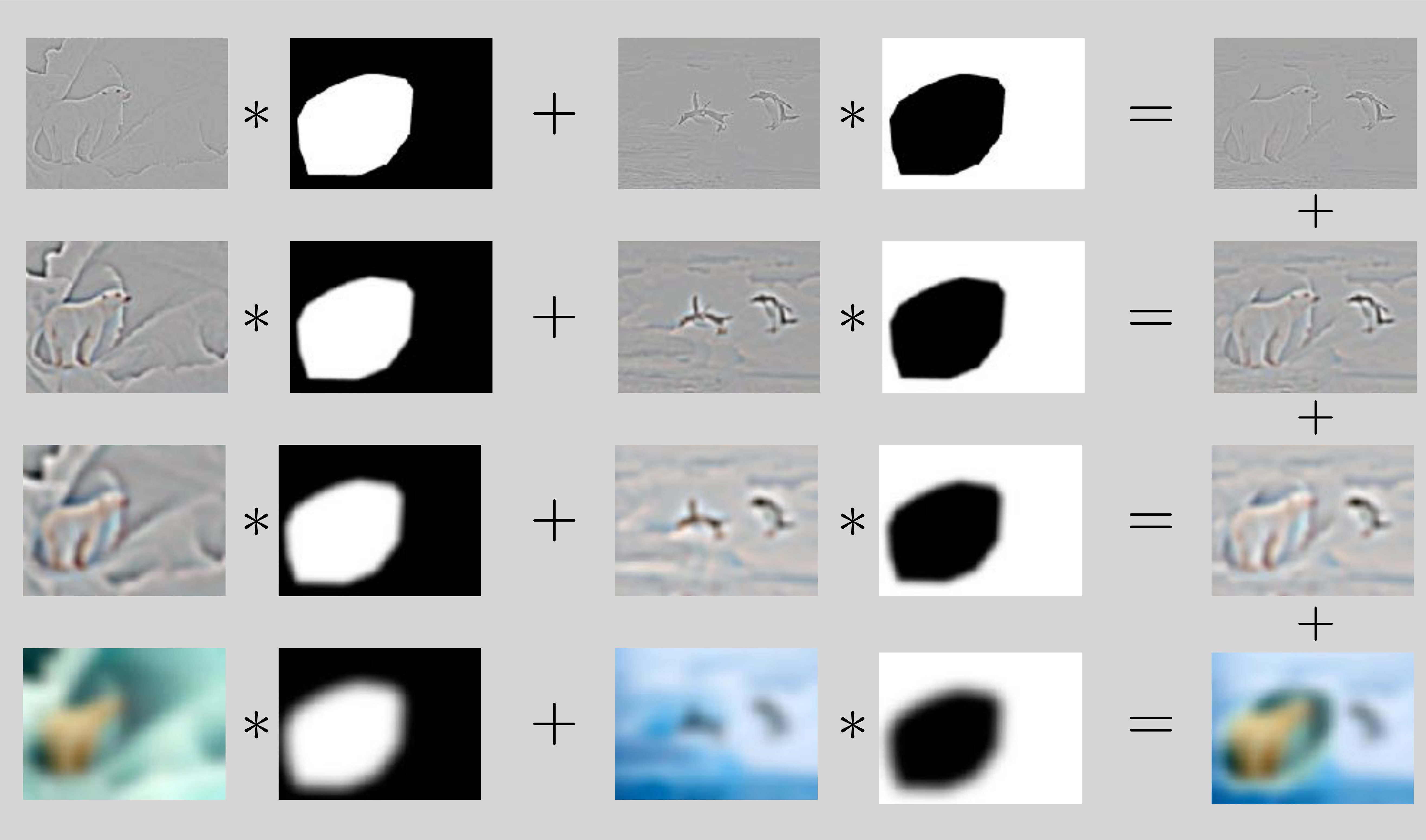
Pyramid Blending

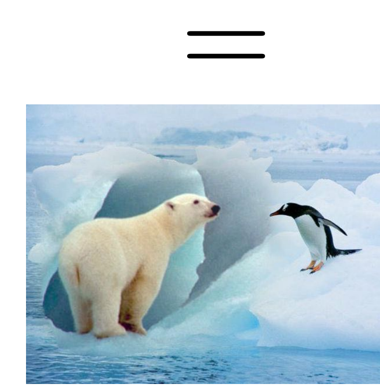
Pyramid Blending





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

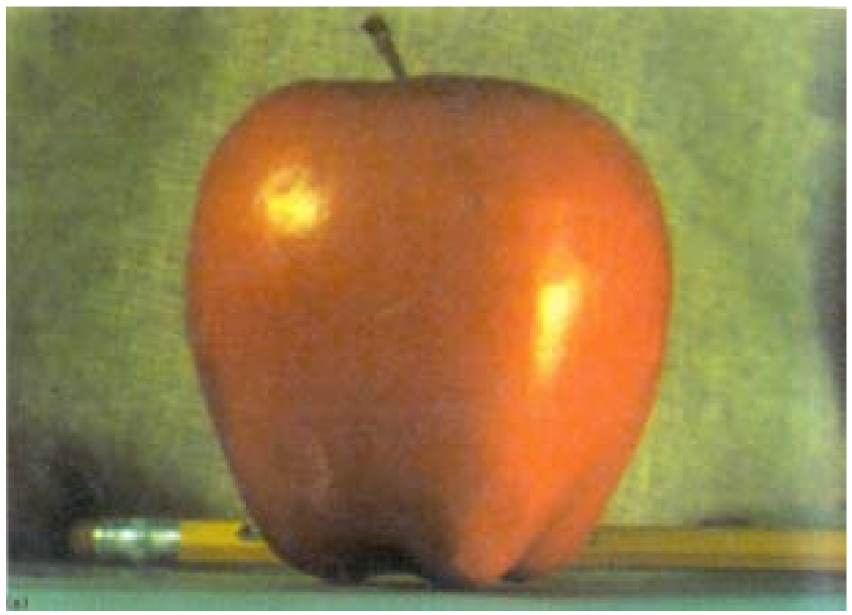




Pyramid Blending: blend lower frequency bands over larger spatial ranges

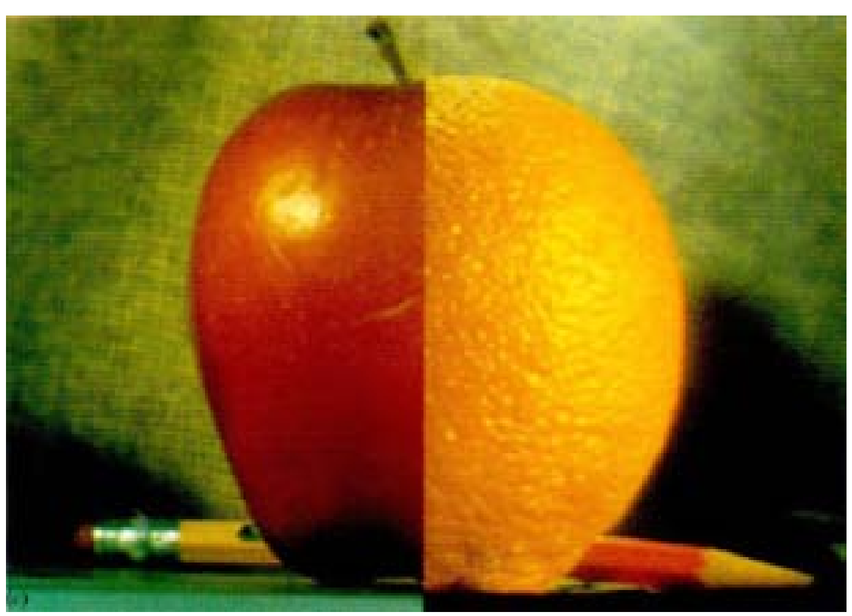
Pyramid Blending

- Smooth low frequencies, whilst preserving high frequency detail



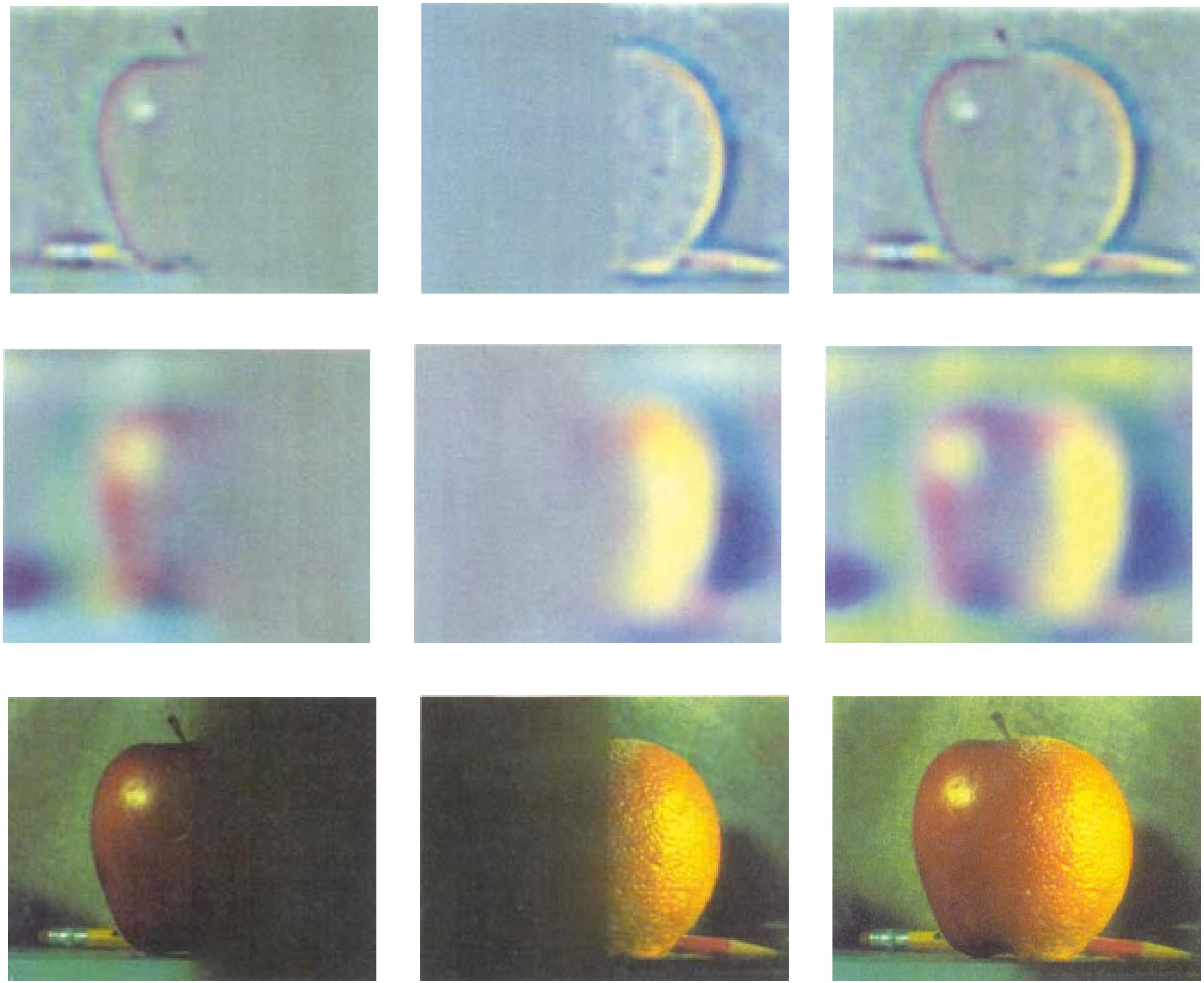
(a)

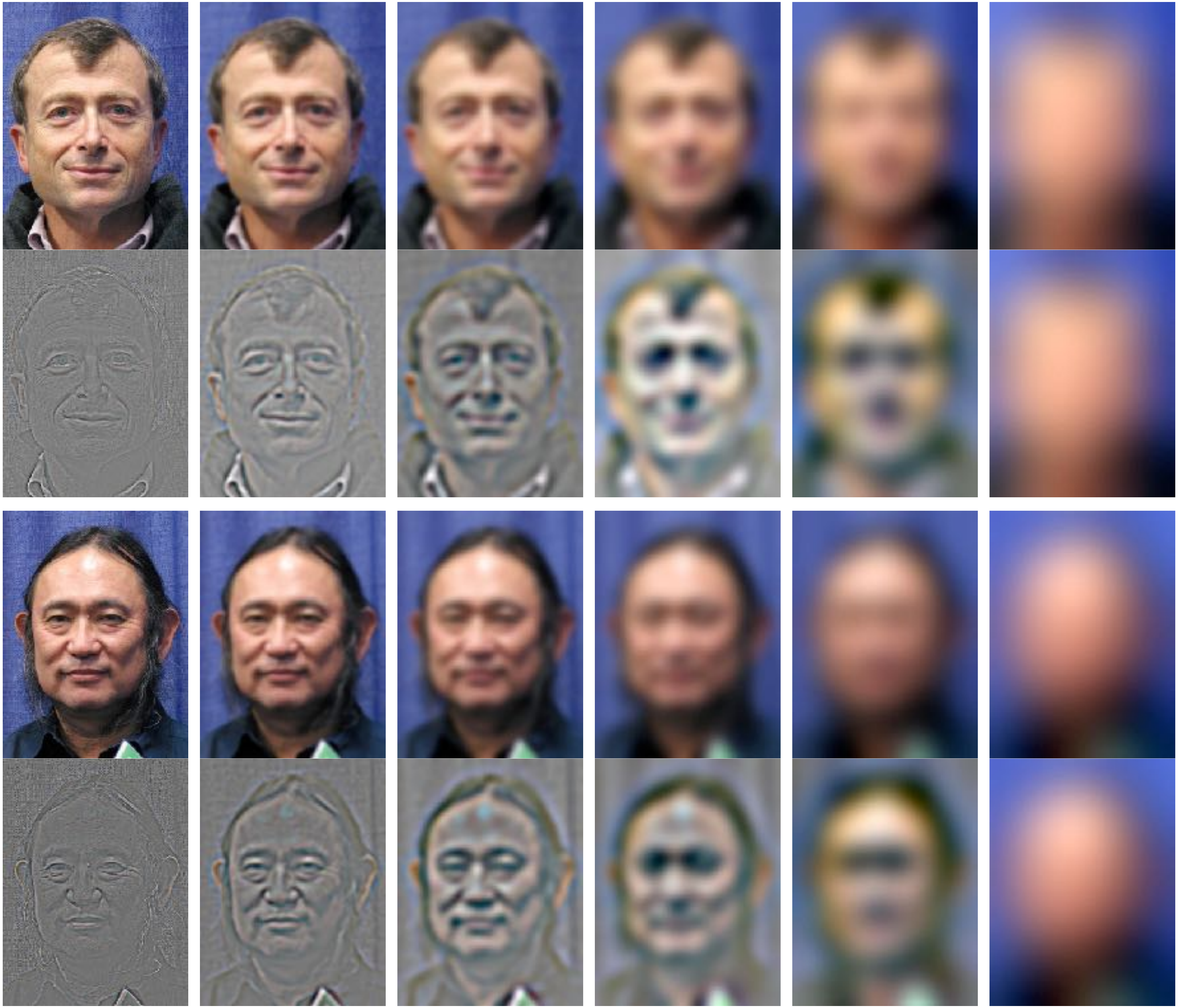
(b)

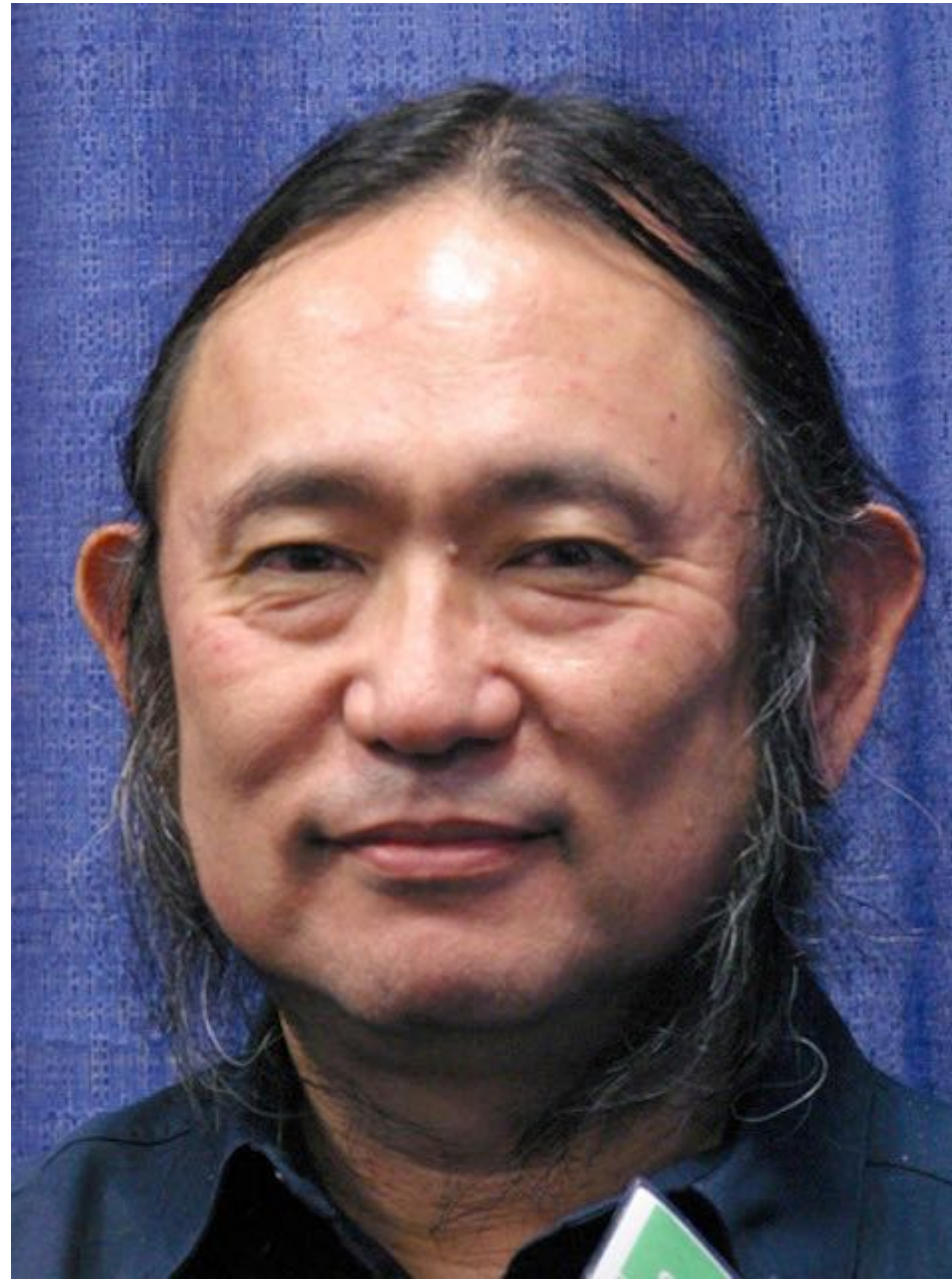


[Burt Adelson 1983]

Pyramid Blending







Alpha blend with sharp fall-off

Alpha blend with gradual fall-off

Pyramid Blend

Non-linear Filtering

- Example: Median filter

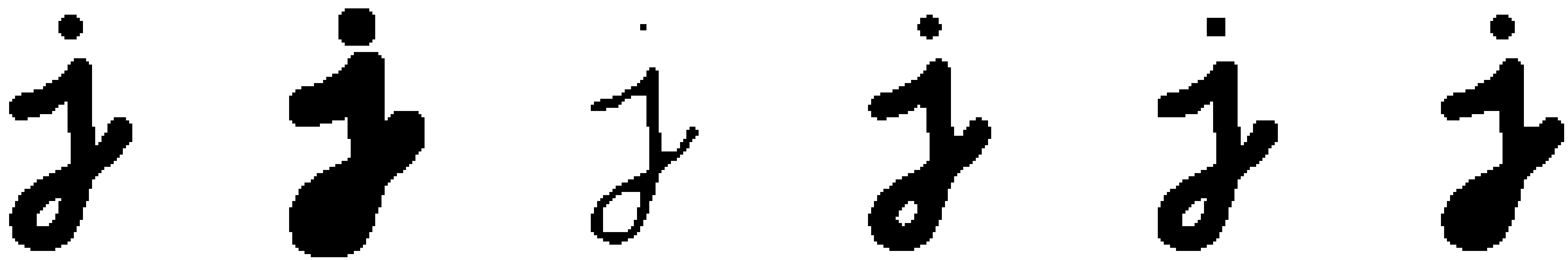
"shot" noise

gaussian blurred

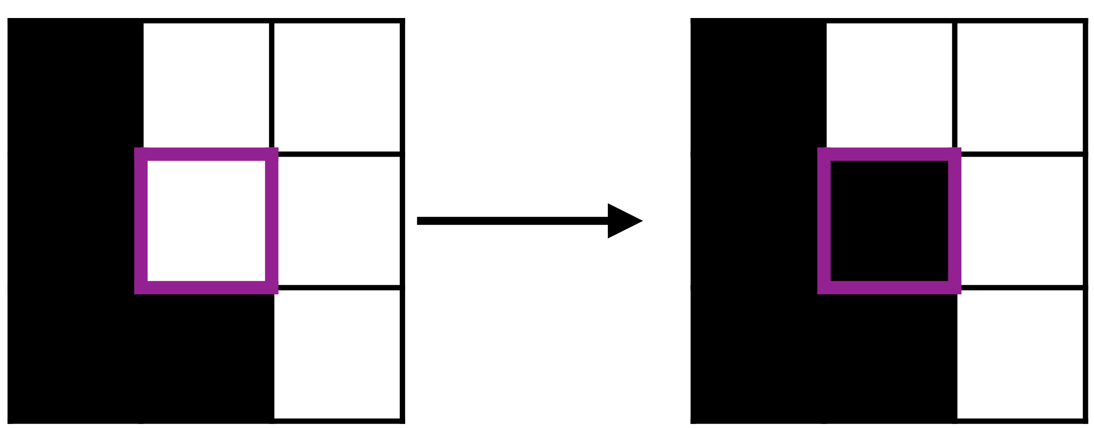
median filtered

Morphology

- Non-linear binary image operations



original dilate erode majority open close

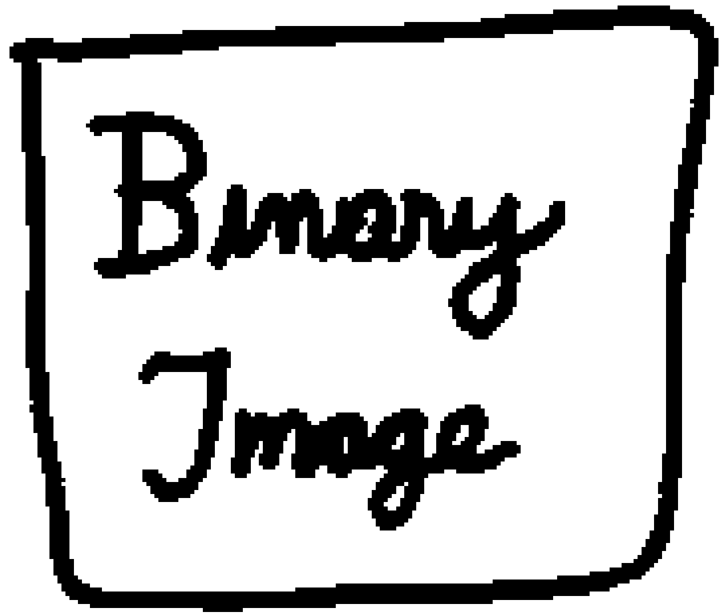


Threshold function in local structuring element

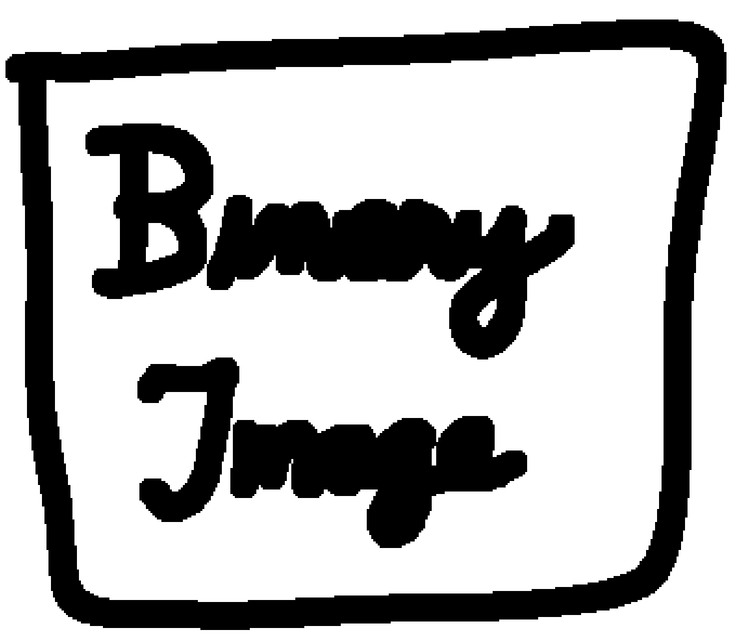
close(.) = erode(dilate(.)) etc., see Szeliski 3.3.2

Binary Operators

- More operators that apply to binary images



original image



dilate

distance transform

connected components

Next Lecture

- Feature Extraction and Matching