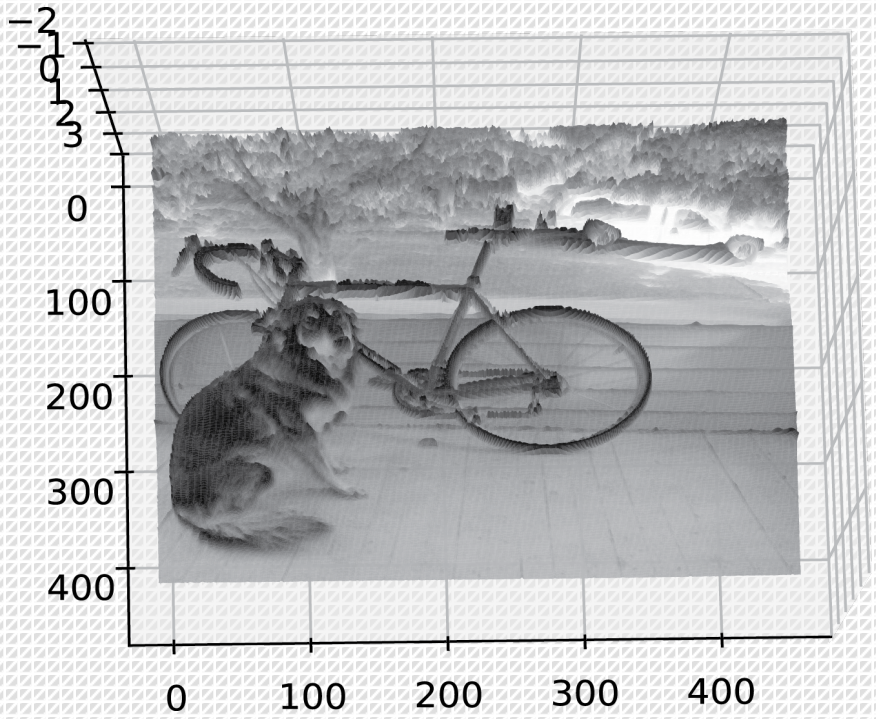
Image processing operators (2)

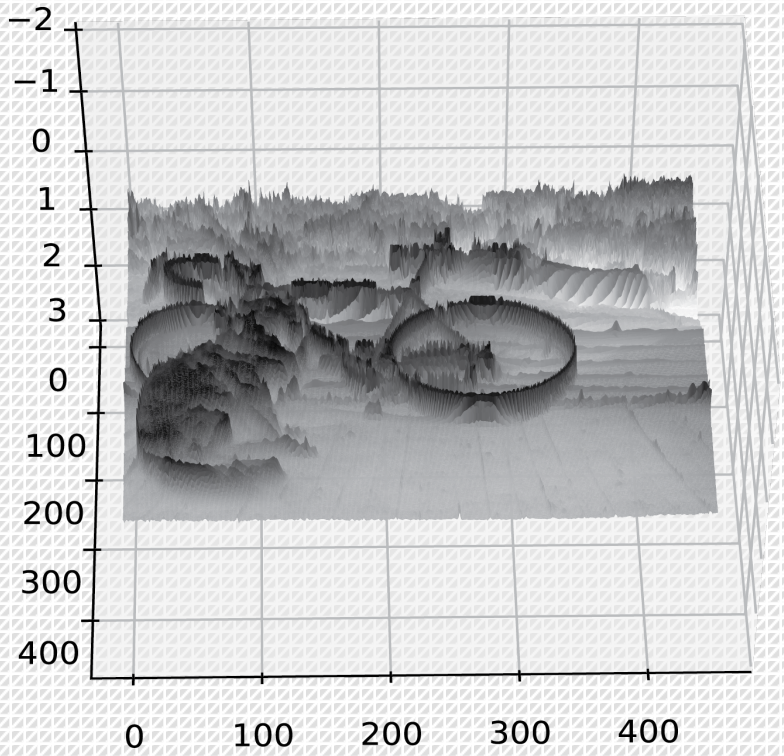
Bing Gong
（巩冰）
gongbing@shnu.edu.cn

Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- What is an “edge”?

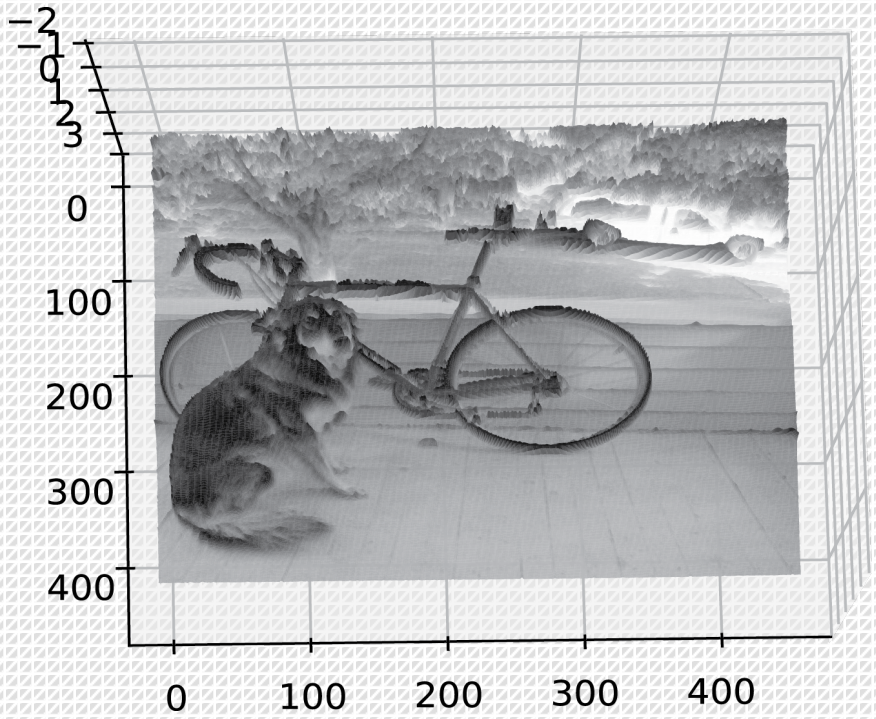


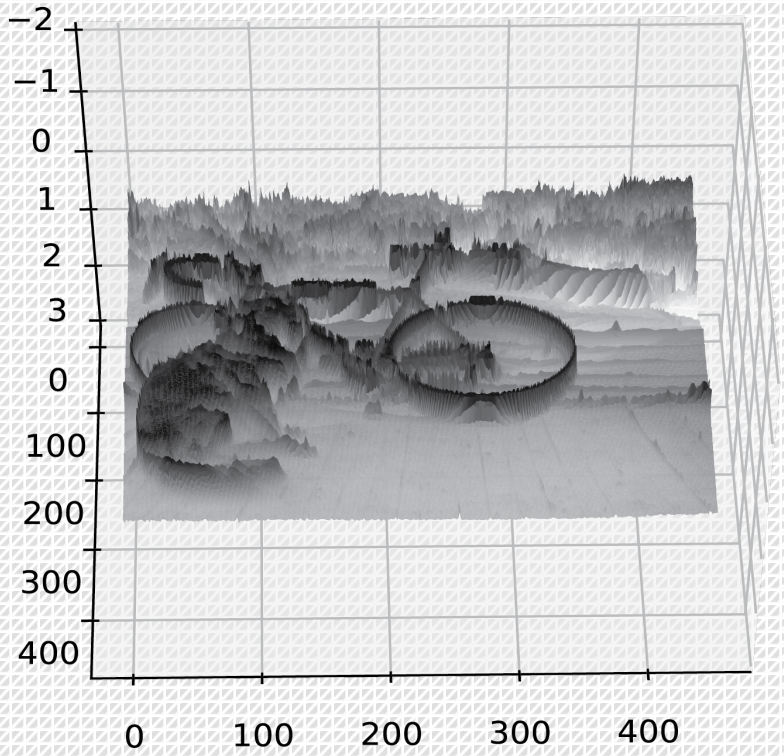


Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Image is a function.
- Think of the gray tones as HEIGHTS.
- Edges are rapid changes in this function

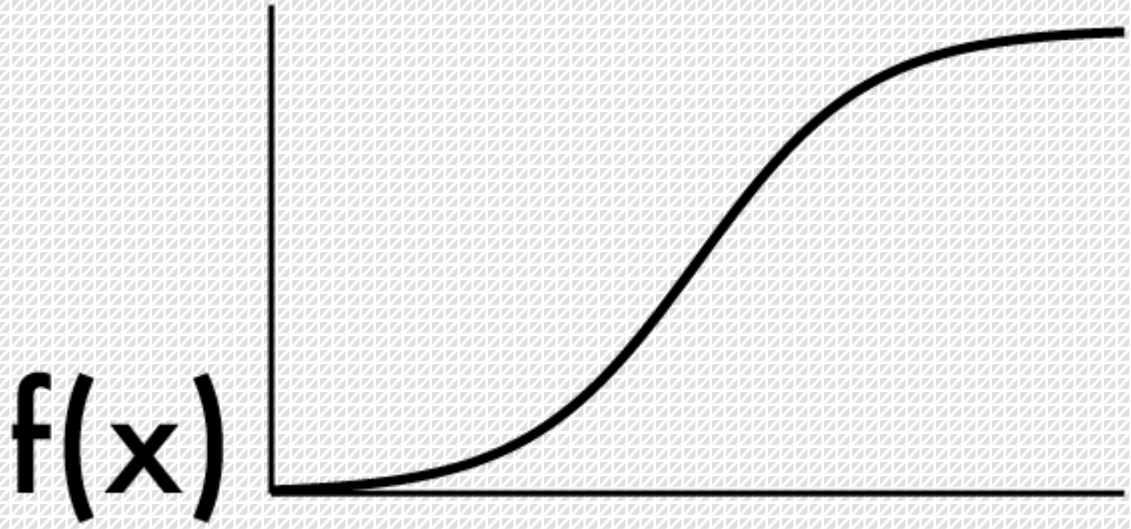




Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

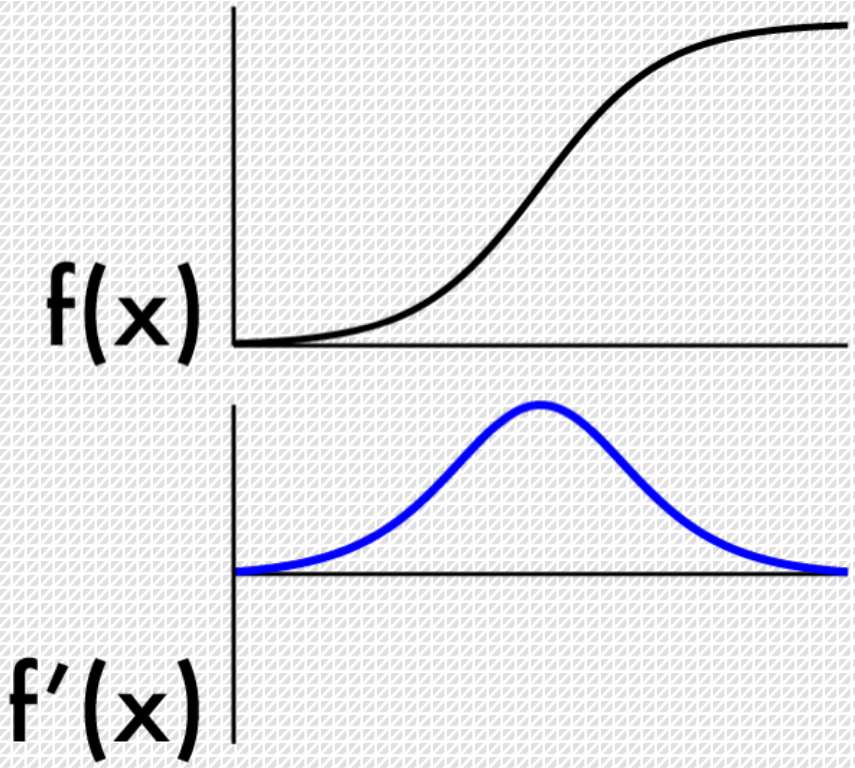
- Image is a function
- Edges are rapid changes in this function



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Image is a function
- Edges are rapid changes in this function



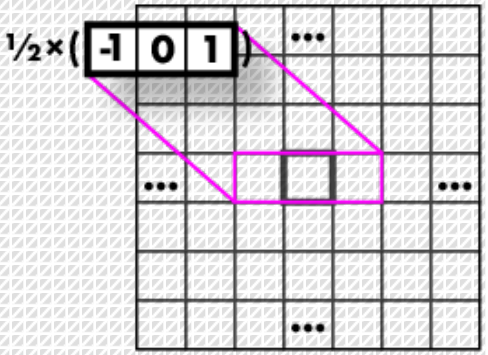
Edges

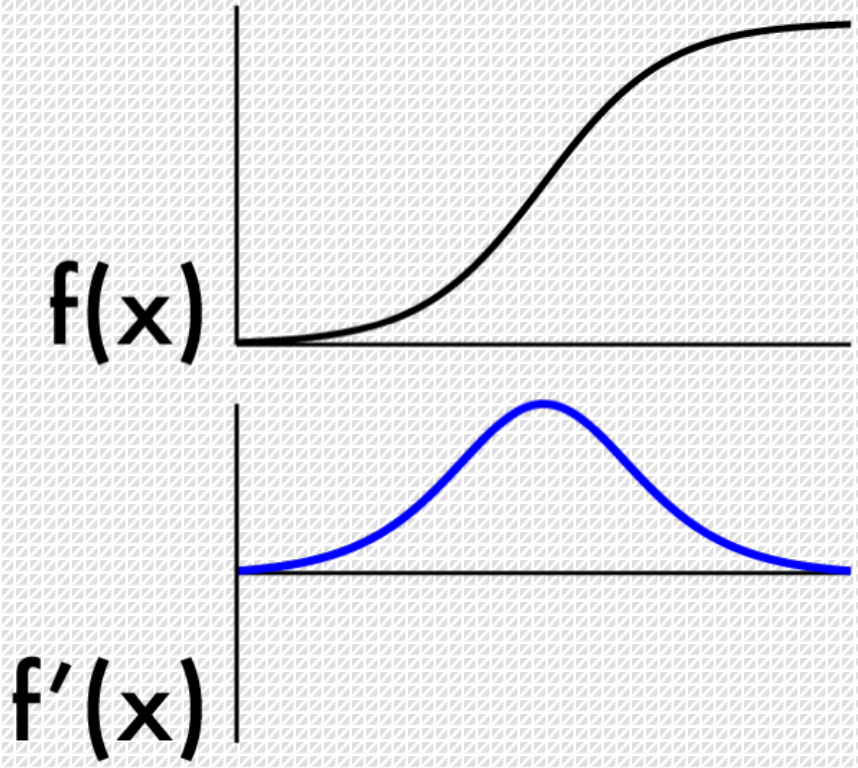
Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Recall:

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}.$$

- We don't have an "actual" function, must estimate
- Possibility: set h = 2
- What will that look like?

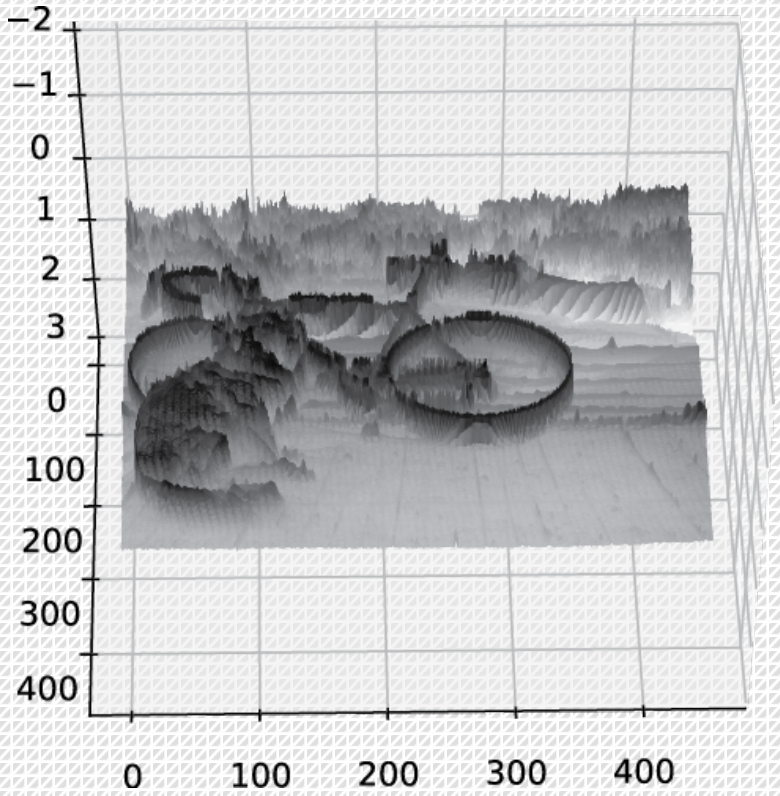


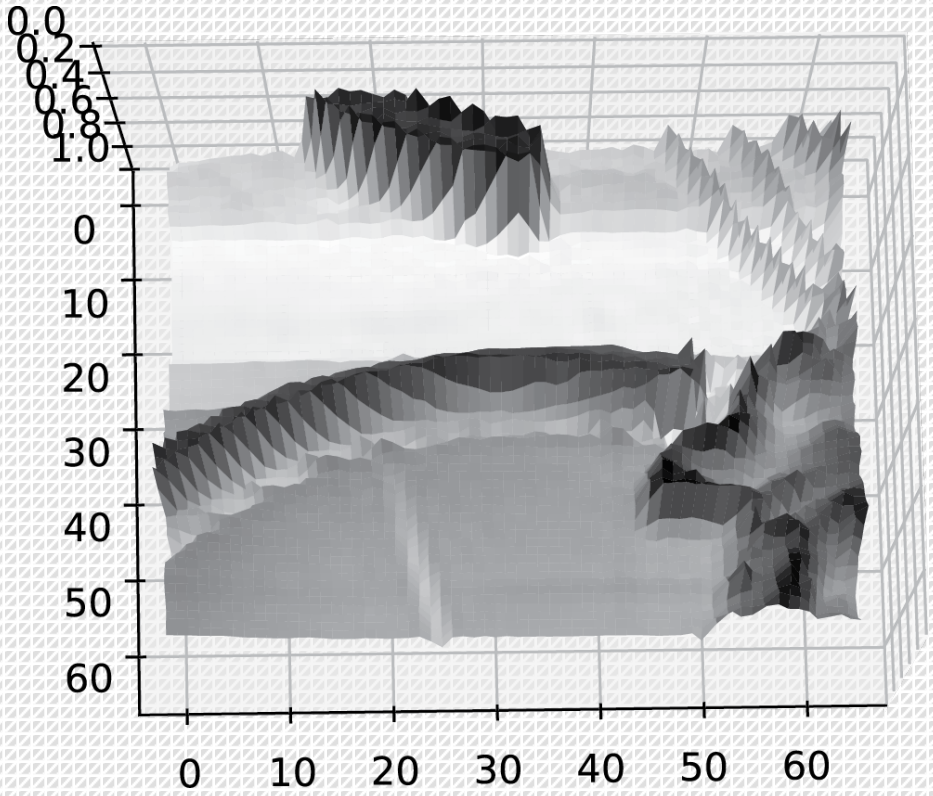


Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Images are noisy!

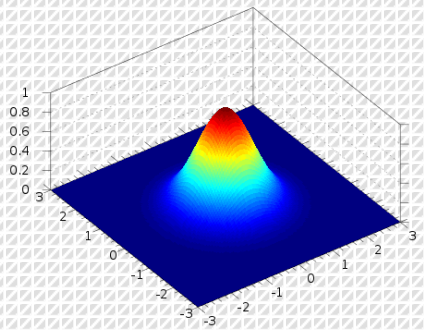


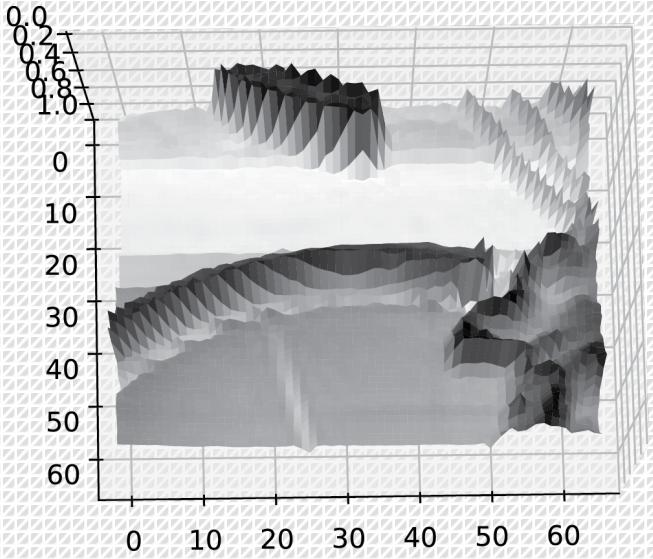


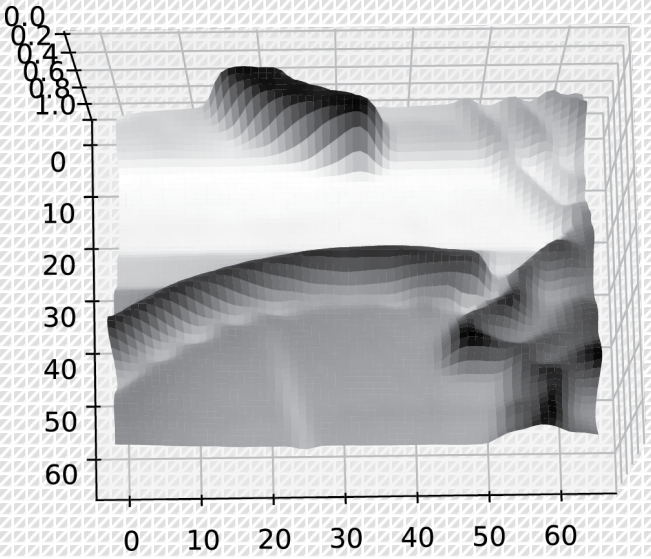
Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Images are noisy!



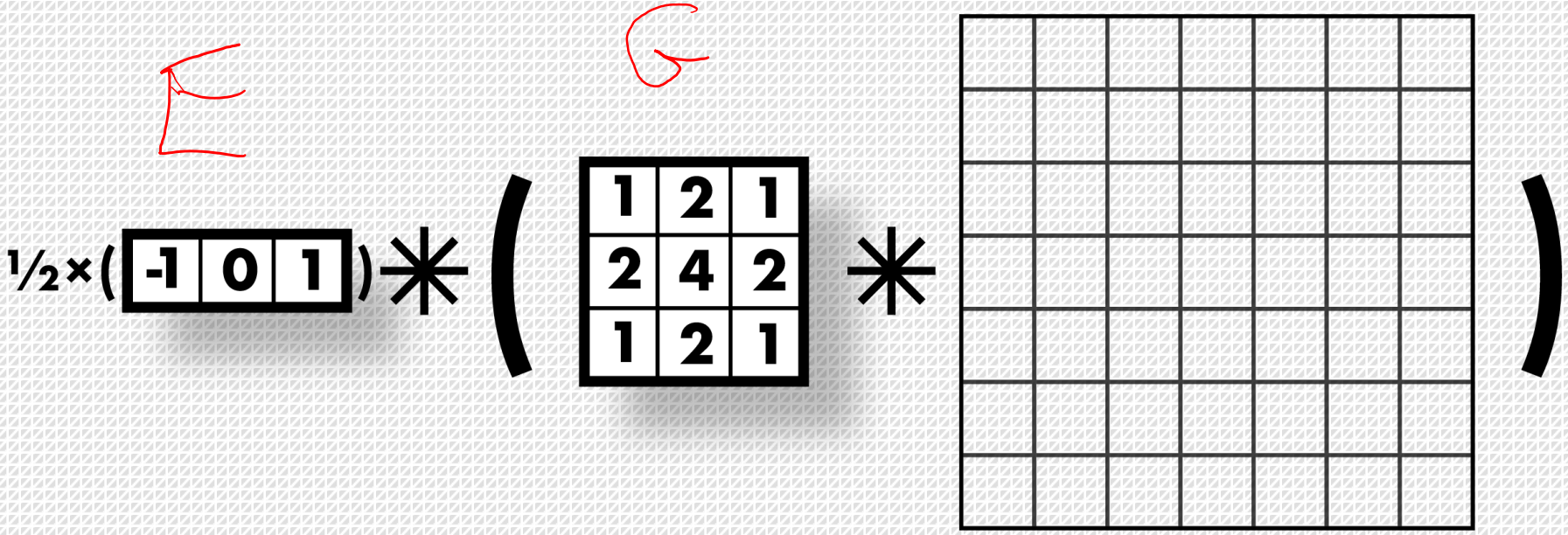




Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

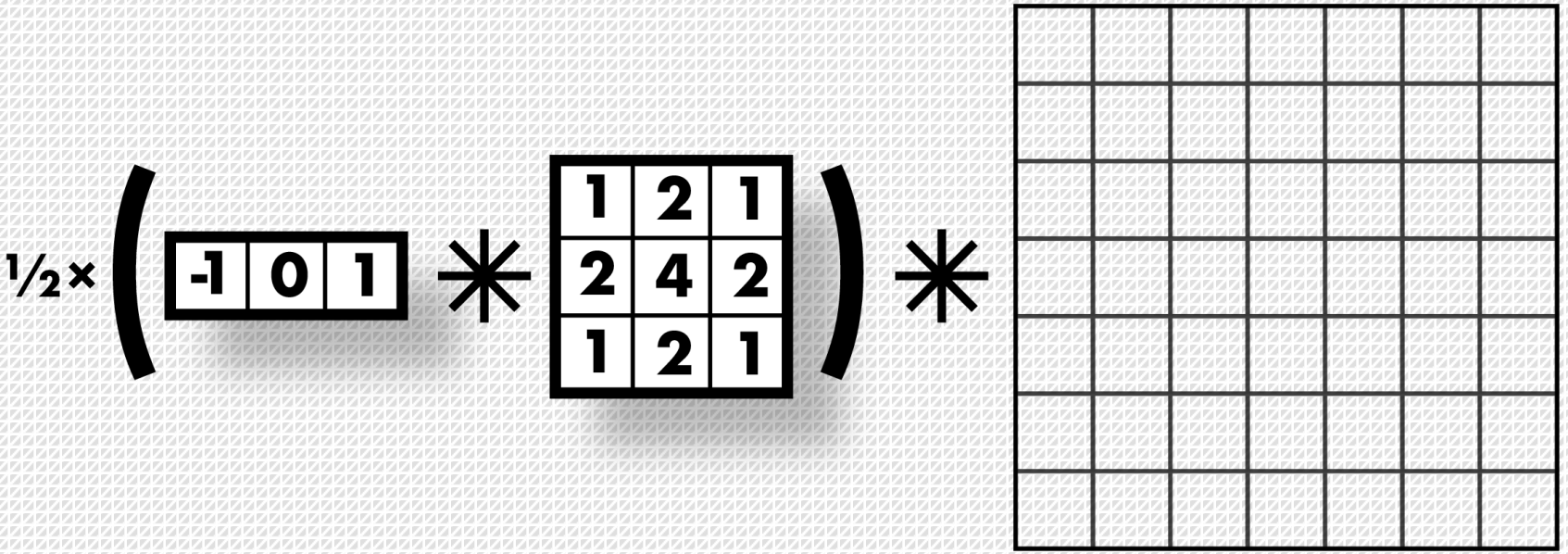
- Images are noisy!



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

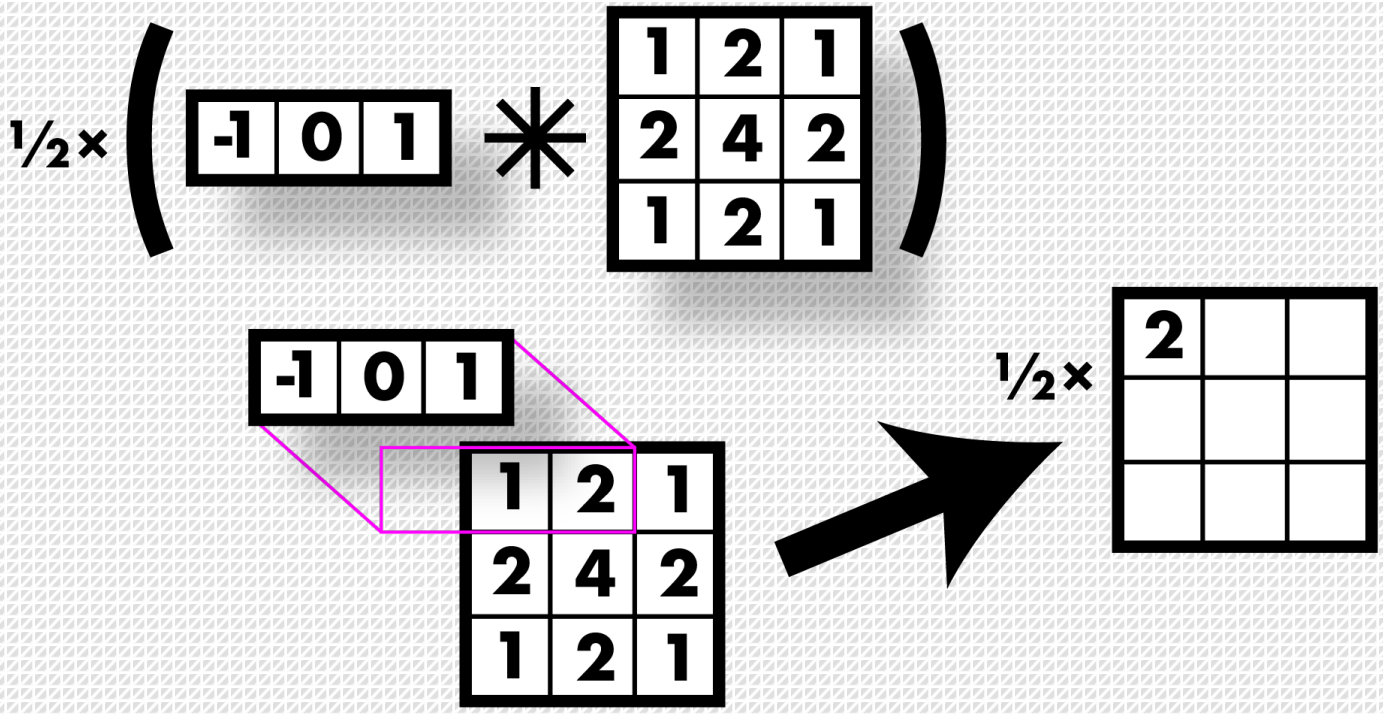
- Smooth first, then derivative



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

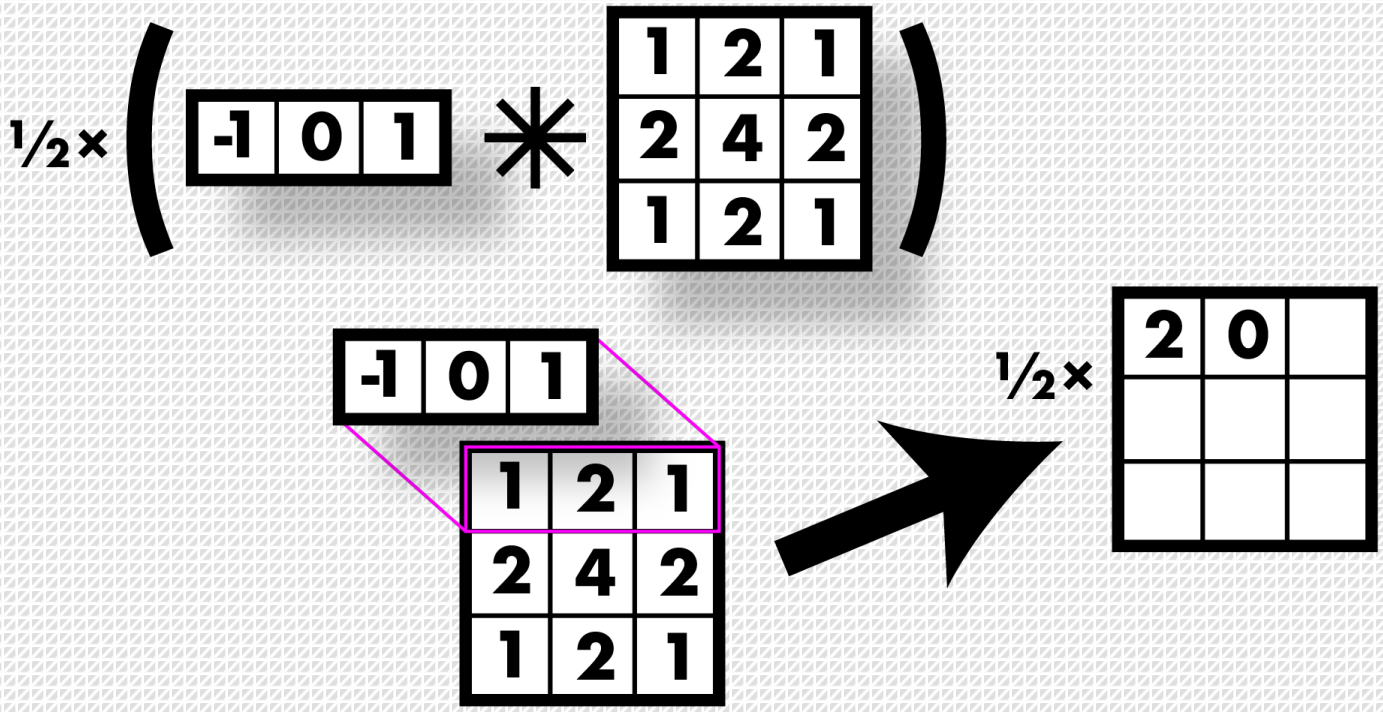
- Smooth first, then derivative



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

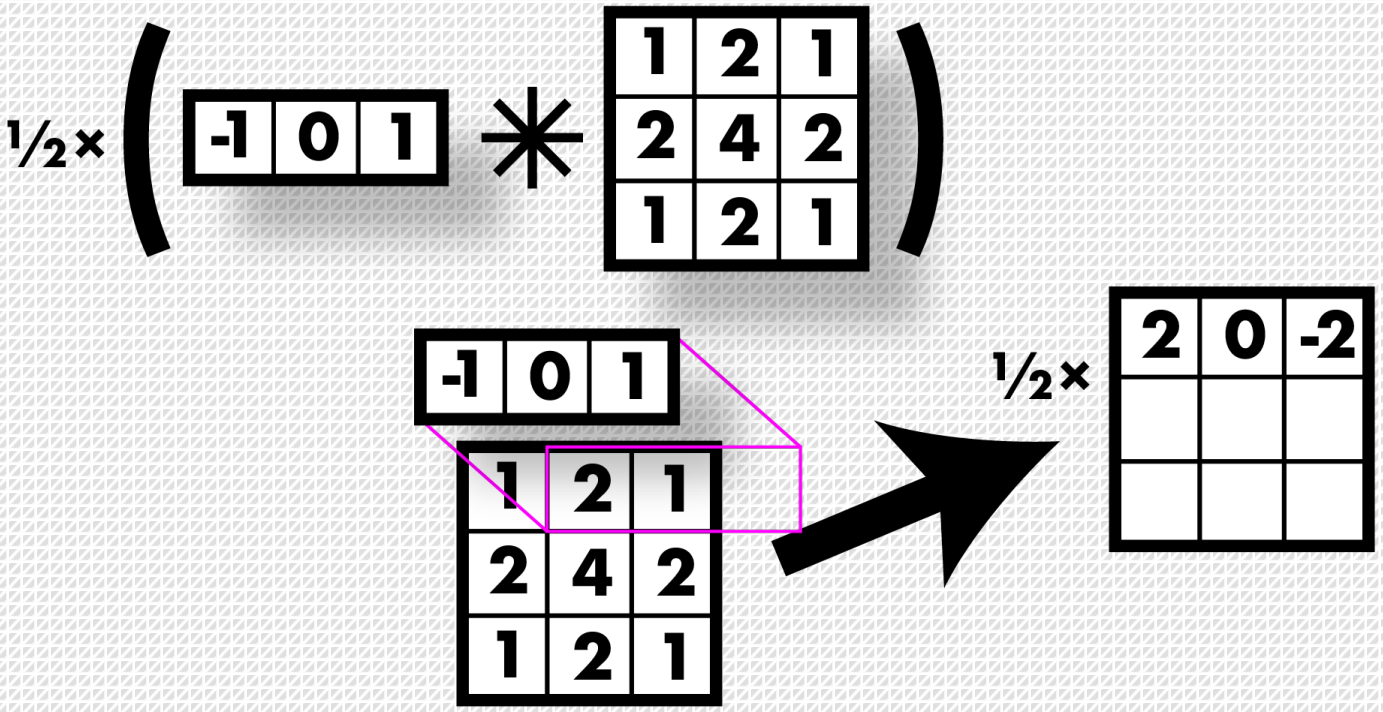
- Smooth first, then derivative



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

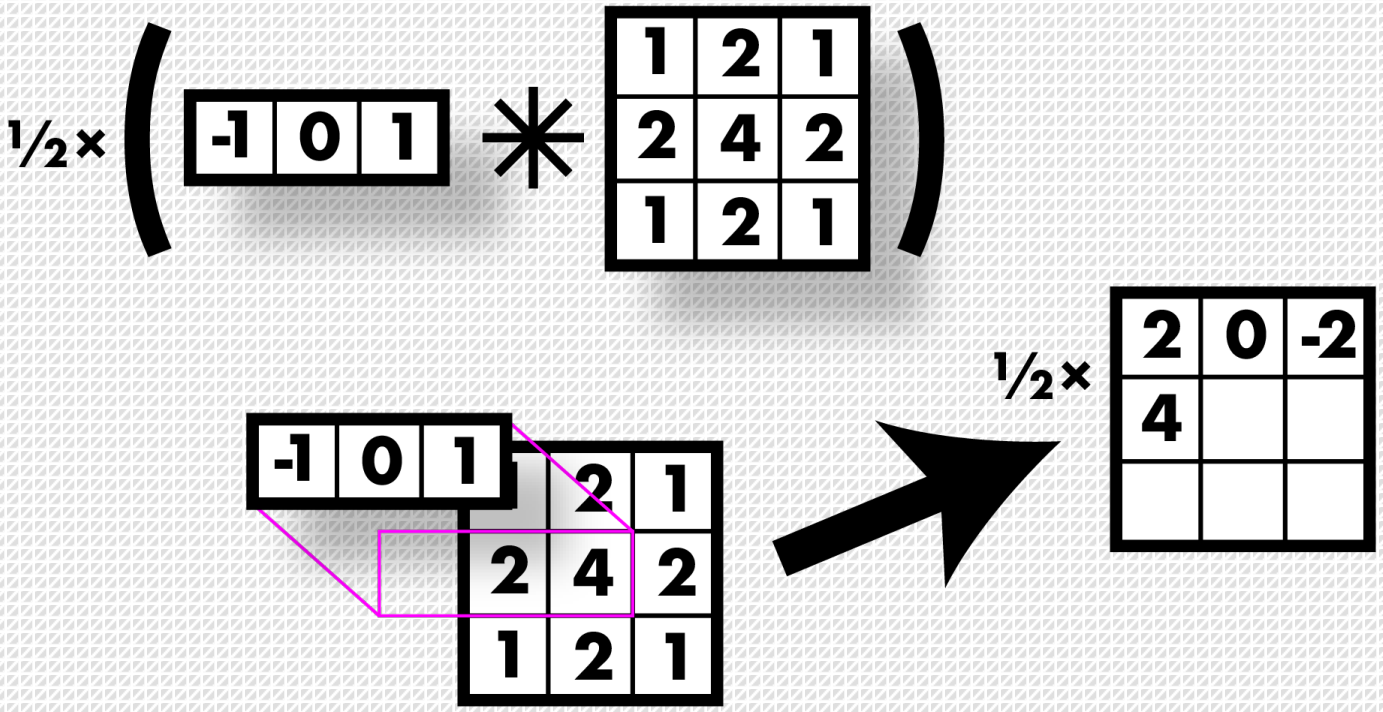
- Smooth first, then derivative



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

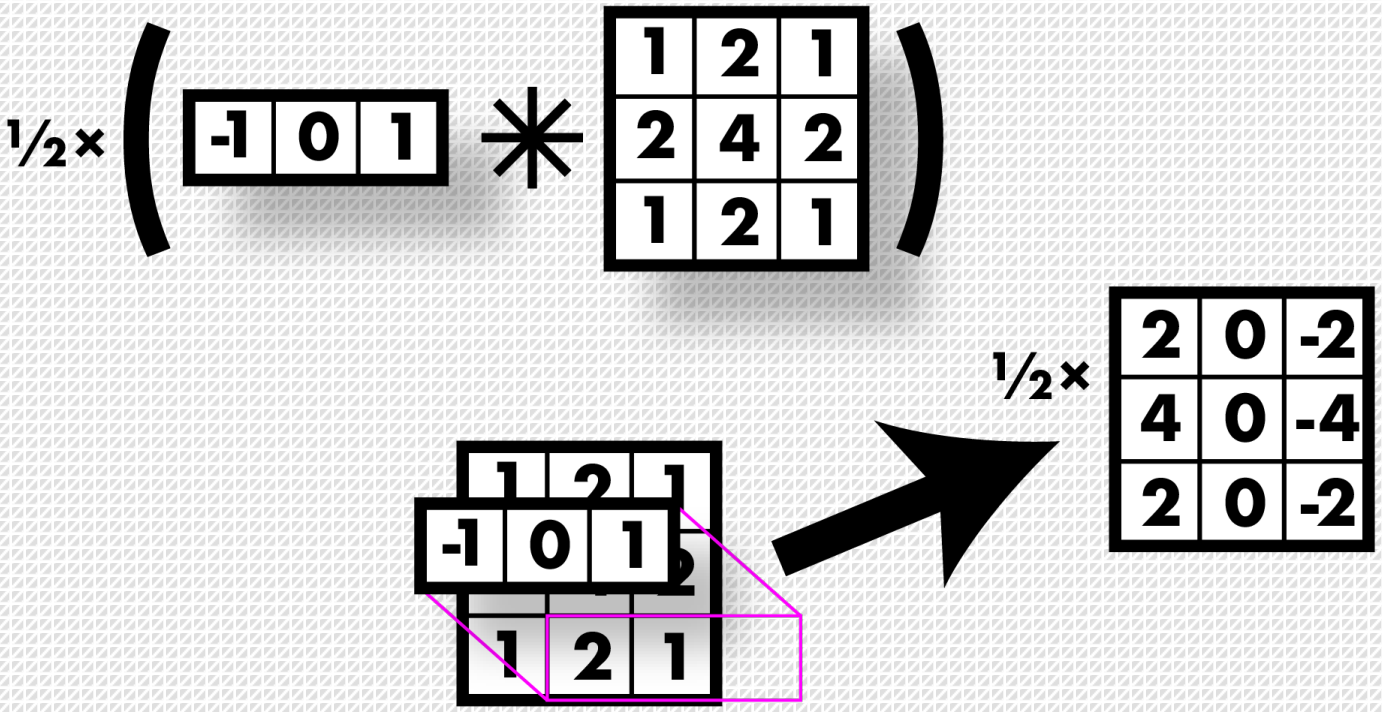
- Smooth first, then derivative



Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Smooth first, then derivative

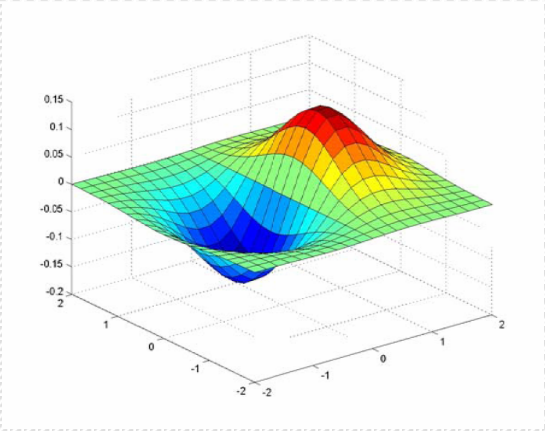


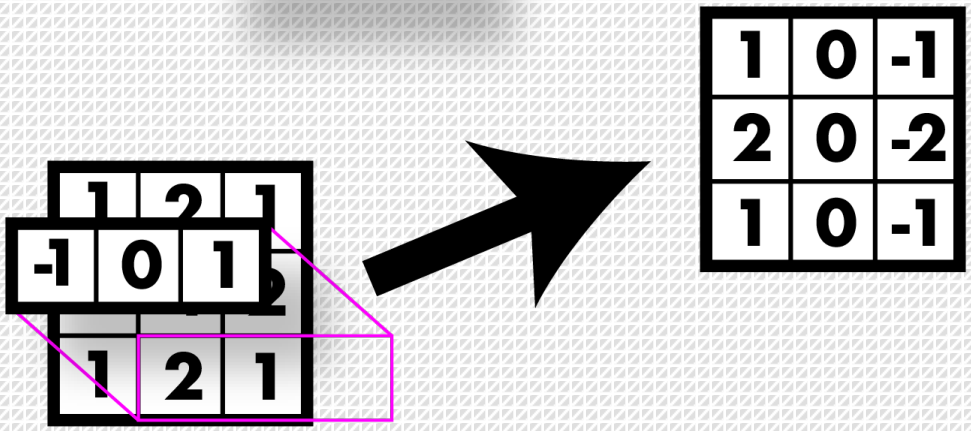
Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Smooth first, then derivative → Sobel filter!

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$





Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Smooth first, then derivative

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

1	1	⋯	1
1	1	⋯	1
⋮	⋮	1	⋮
1	1	⋯	1

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$\frac{1}{256}$

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

$\frac{1}{8}$

−1	0	1
−2	0	2
−1	0	1

$\frac{1}{4}$

1	−2	1
−2	4	−2
1	−2	1

$\frac{1}{K}$ | 1 | 1 | ⋯ | 1 |

$\frac{1}{4}$ | 1 | 2 | 1 |

$\frac{1}{16}$ | 1 | 4 | 6 | 4 | 1 |

$\frac{1}{2}$ | −1 | 0 | 1 |

$\frac{1}{2}$ | 1 | −2 | 1 |

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

Figure 3.14 Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by 2× and 4×, respectively, and added to a gray offset before display.

Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- But

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

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

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$\frac{1}{256}$

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

$\frac{1}{8}$

−1	0	1
−2	0	2
−1	0	1

$\frac{1}{4}$

1	−2	1
−2	4	−2
1	−2	1

$\frac{1}{K}$ | 1 | 1 | $\cdots$ | 1 |

$\frac{1}{4}$ | 1 | 2 | 1 |

$\frac{1}{16}$ | 1 | 4 | 6 | 4 | 1 |

$\frac{1}{2}$ | −1 | 0 | 1 |

$\frac{1}{2}$ | 1 | −2 | 1 |

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

Figure 3.14 Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by 2× and 4×, respectively, and added to a gray offset before display.

Edges

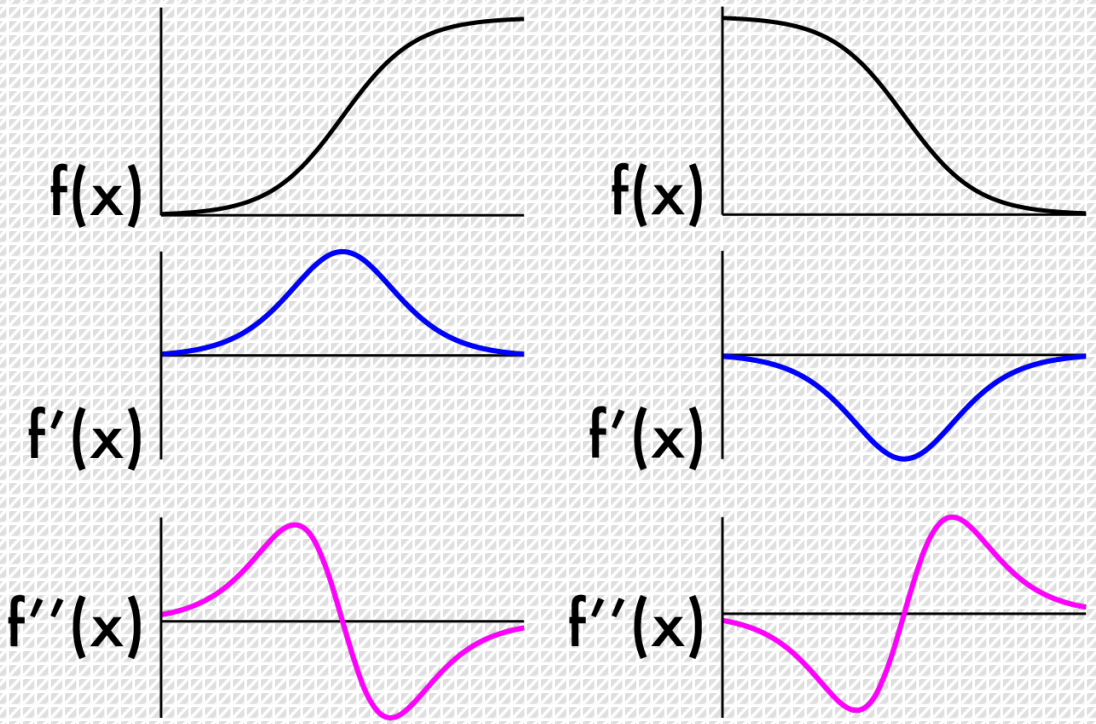
Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Could take derivative
- Find high responses
- Sobel filters!
- But…
- Edges go both ways
- Want to find extrema

Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Could take derivative
- Find high responses
- Sobel filters!
- But…
- Edges go both ways
- Want to find extrema



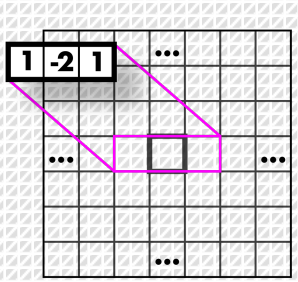
Edges

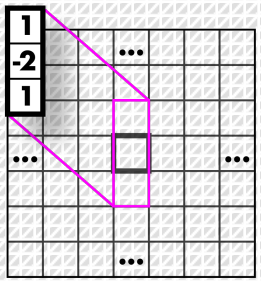
Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

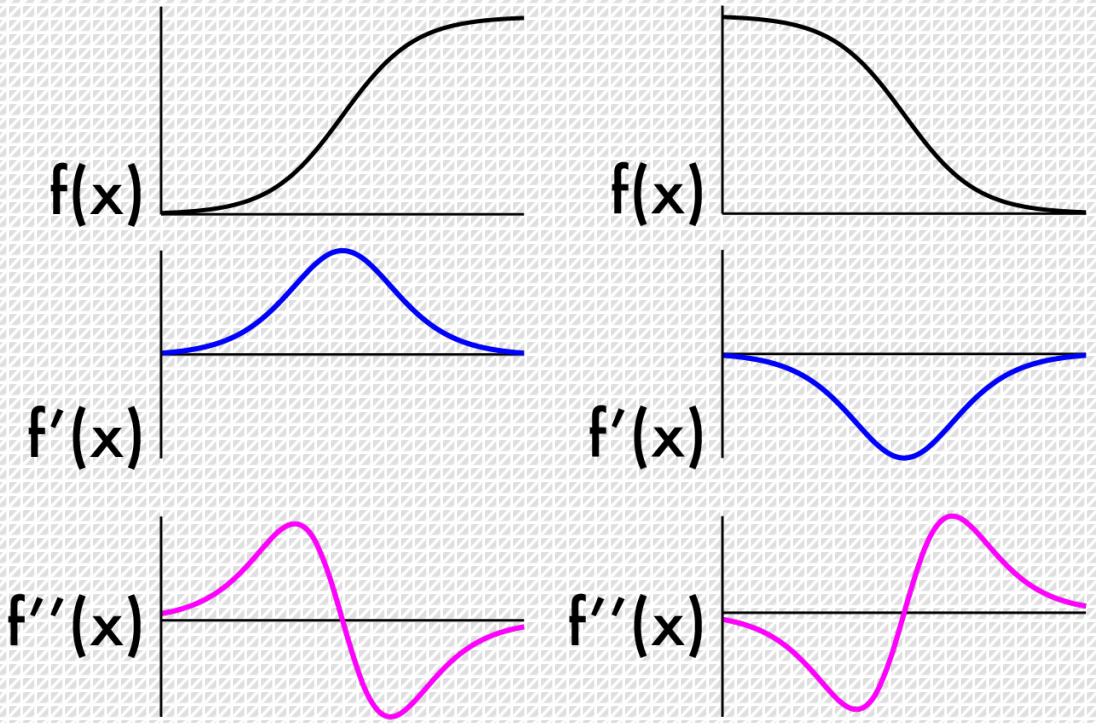
- Crosses zero at extrema
- Recall:
  - $$f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$
- Laplacian:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Again, have to estimate f’’(x):







Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Laplacians

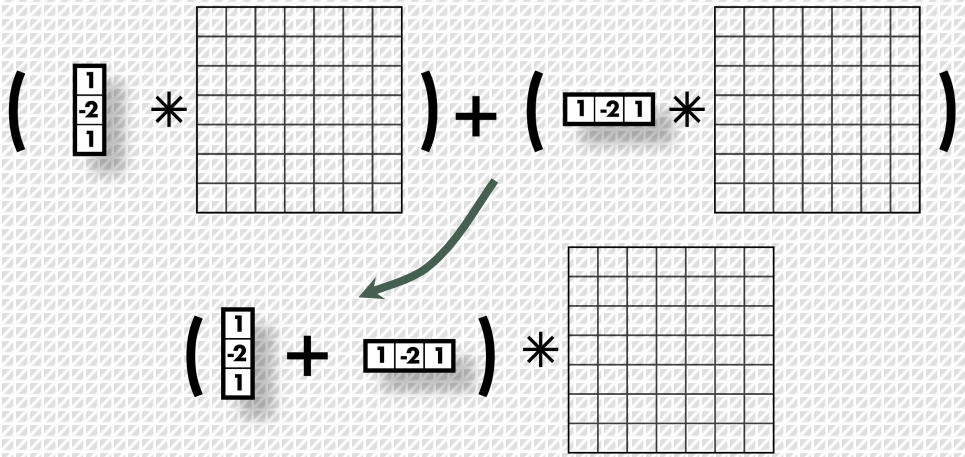
$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Laplacians

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

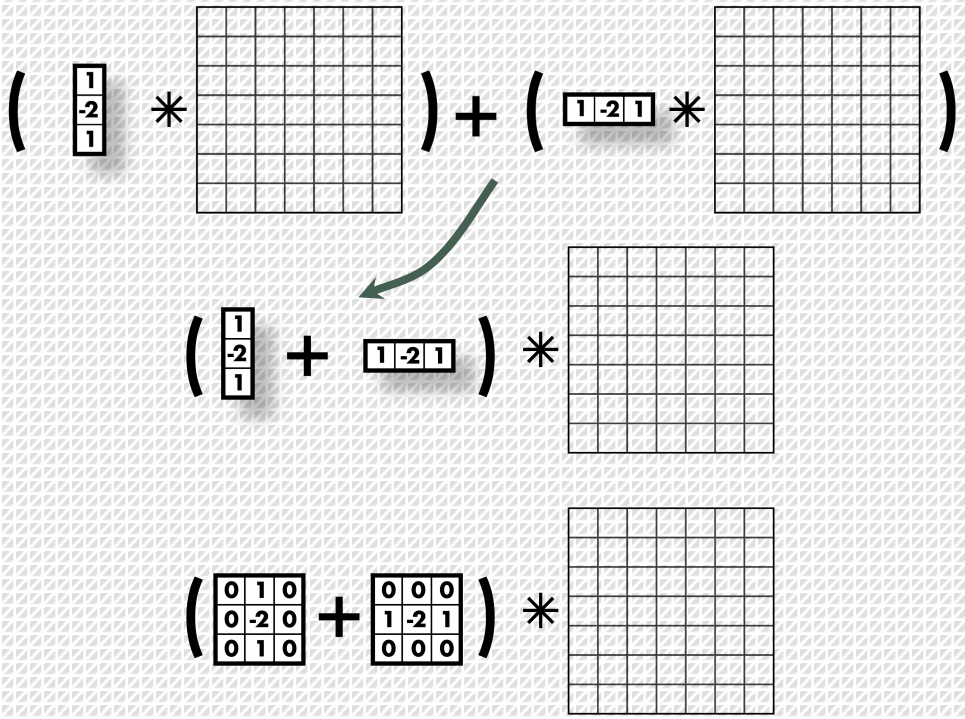


Edges

Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

- Laplacians

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

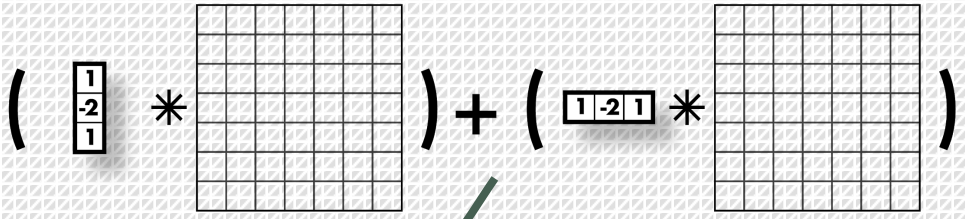


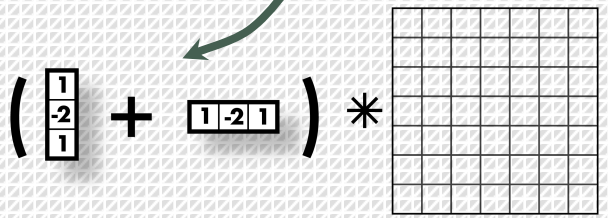
Edges

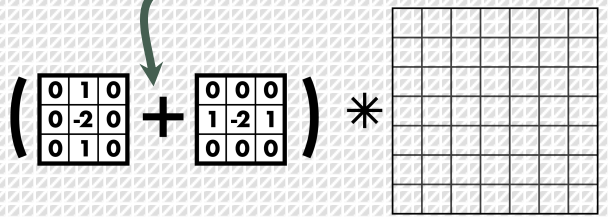
Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

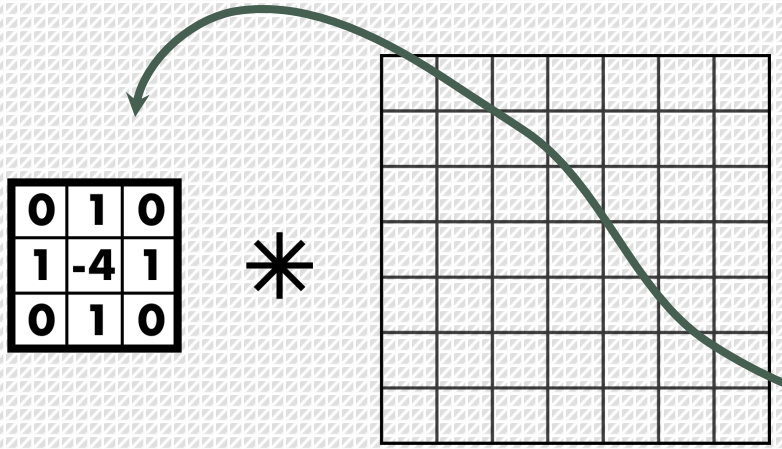
- Laplacians

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$







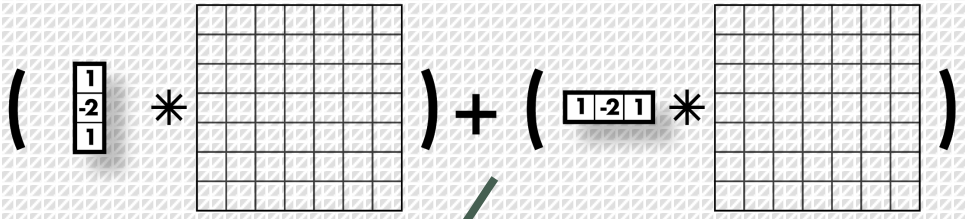


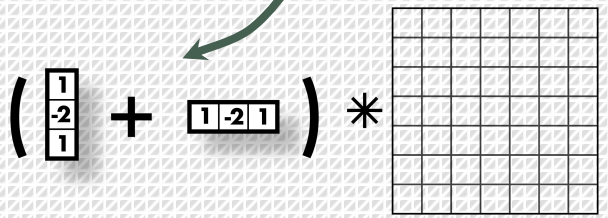
Edges

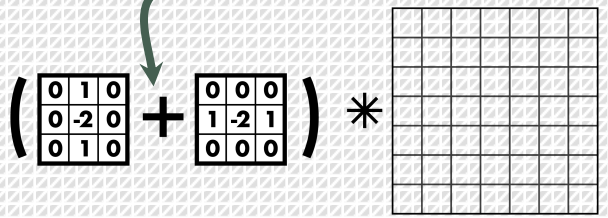
Source: https://courses.cs.washington.edu/courses/cse576/23sp/notes.html

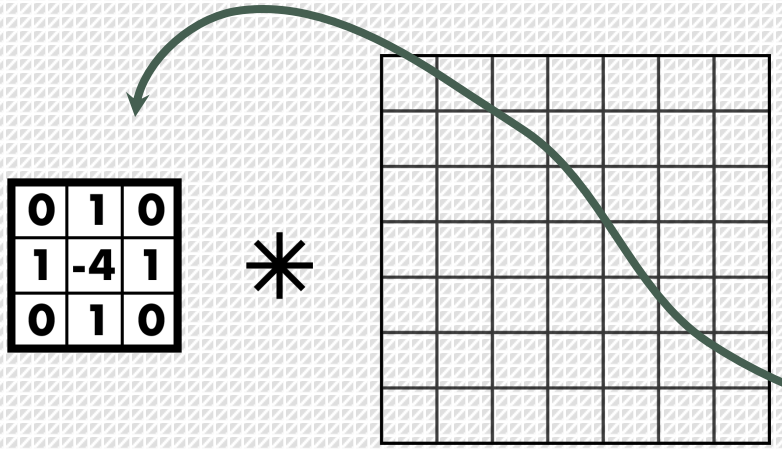
- Laplacians

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$









Other linear filters

- Summed area tables (*integral image*)
- … is used for face detection to compute simple multi-scale low-level features

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

Figure 3.17 Summed area tables: (a) original image; (b) summed area table; (c) computation of area sum. Each value in the summed area table $s(i, j)$ (red) is computed recursively from its three adjacent (blue) neighbors (3.31). Area sums S (green) are computed by combining the four values at the rectangle corners (purple) (3.32). Positive values are shown in **bold** and negative values in *italics*.

Non-linear filter

- Why we need non-linear filter?
- Regular blurring with a Gaussian filter fails to remove the noisy pixels and instead turns them into softer (but still visible) spots

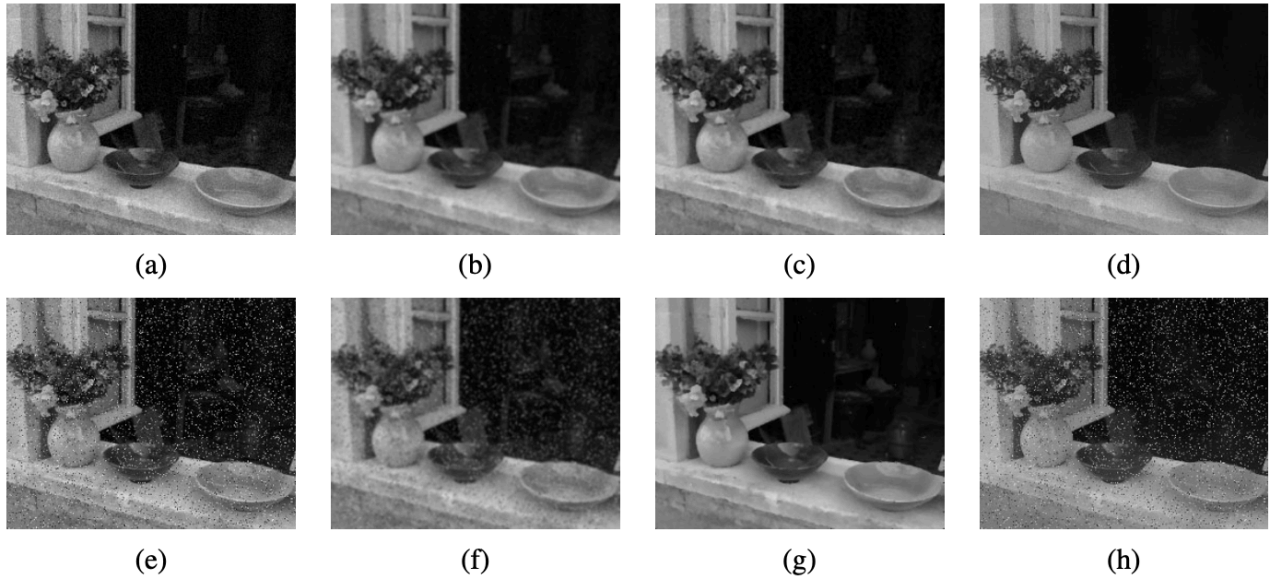


Figure 3.18 Median and bilateral filtering: (a) original image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered; (e) original image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered. Note that the bilateral filter fails to remove the shot noise because the noisy pixels are too different from their neighbors.

Non-linear filter

- Median Filtering:

…selects the median value from each pixel's neighborhood.

…since the shot noise value usually lies well outside the true values in the neighborhood, the median filter is able to filter away such bad pixels.

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

(a) median = 4

Non-linear filter

- Median Filtering (α-trimmed mean):

….averages together all of the pixels except for the α fraction that are the smallest and the largest

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

(b) α-mean= 4.6

Non-linear filter

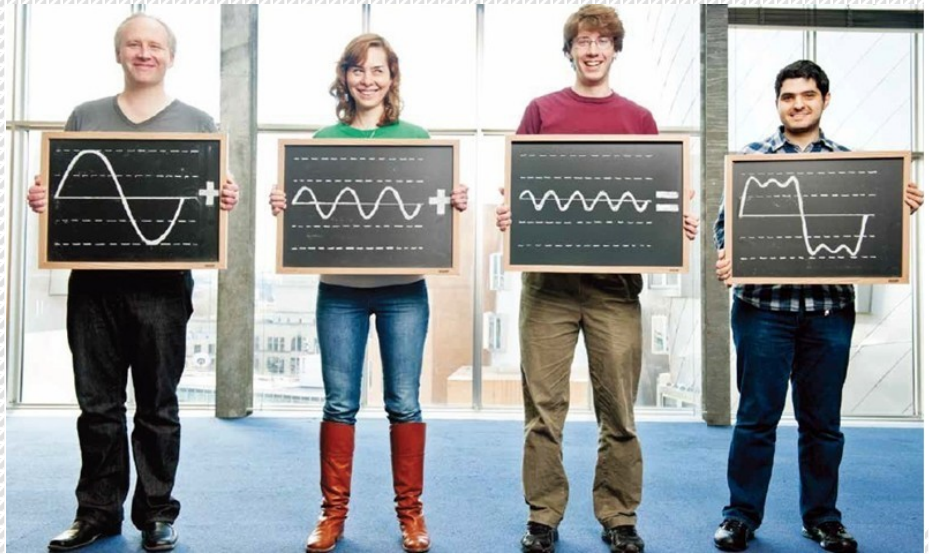
- Median Filtering (weighted median):
- …. each pixel is used a number of times depending on its distance from the center

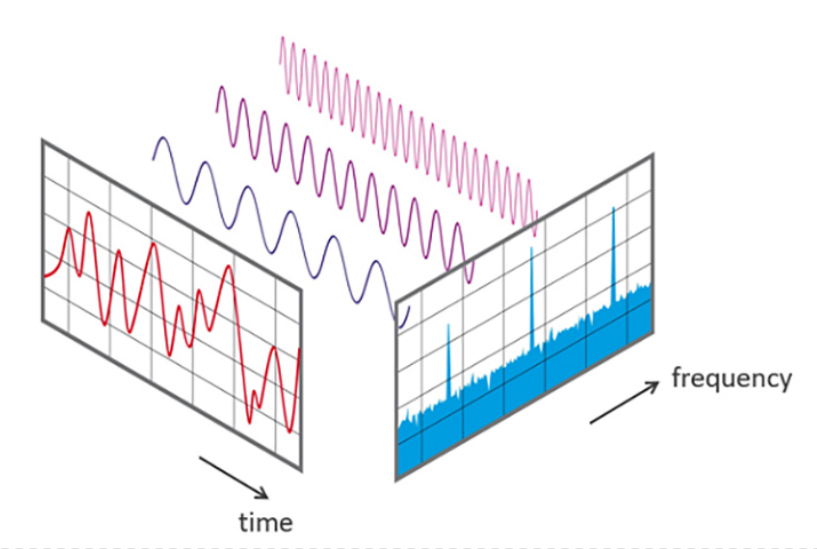
1	2	1	2	4
2	1	3	5	8
1	3	7	6	9
3	4	8	6	7
4	5	7	8	9

(b) α-mean= 4.6

Fourier transforms

- Spatial domain → Frequency domain

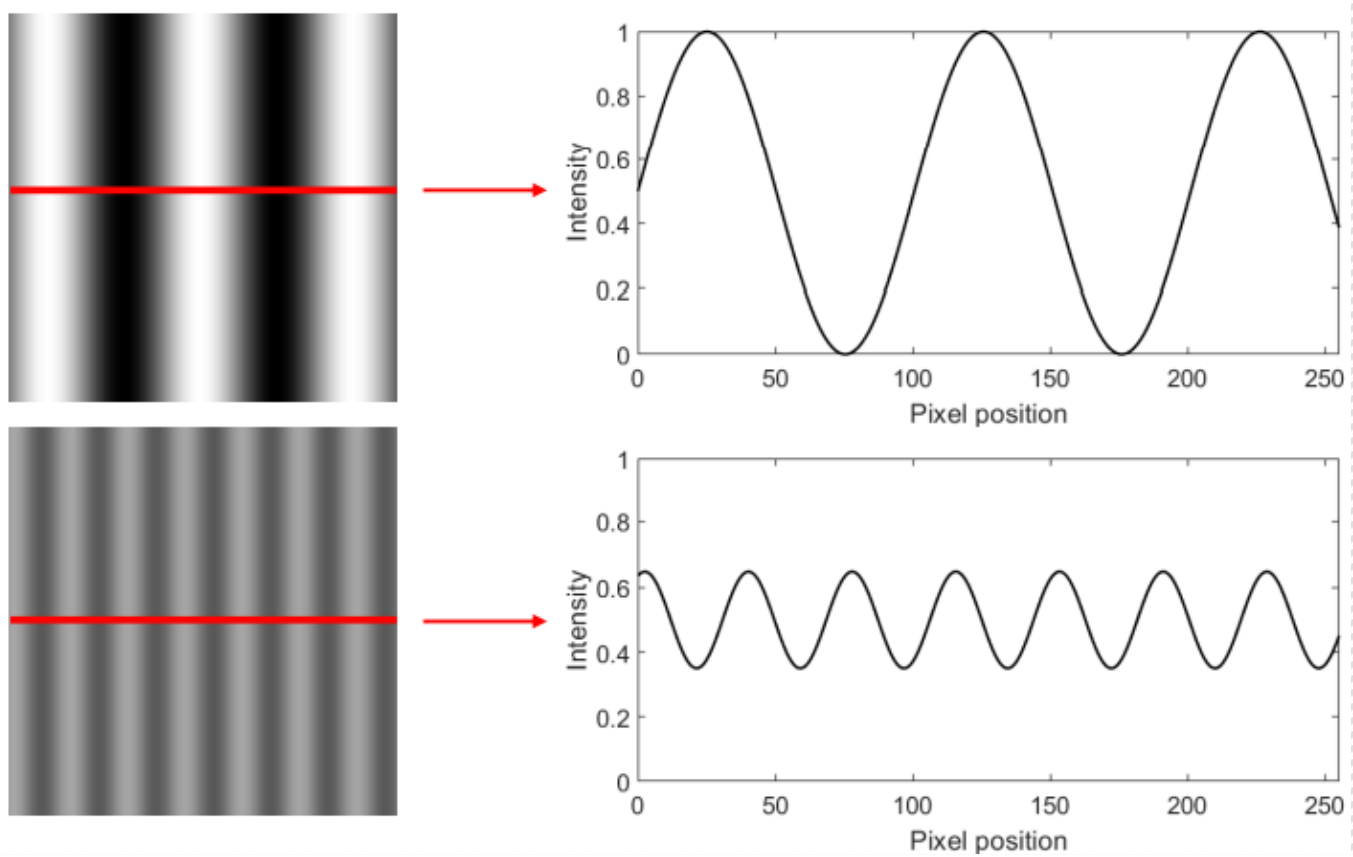




https://www.bilibili.com/video/BV1uY411z7uk/?spm_id_from=333.788.recommend_more_video.0&vd_source=1d98d45f728c64dff48447f704f66bae

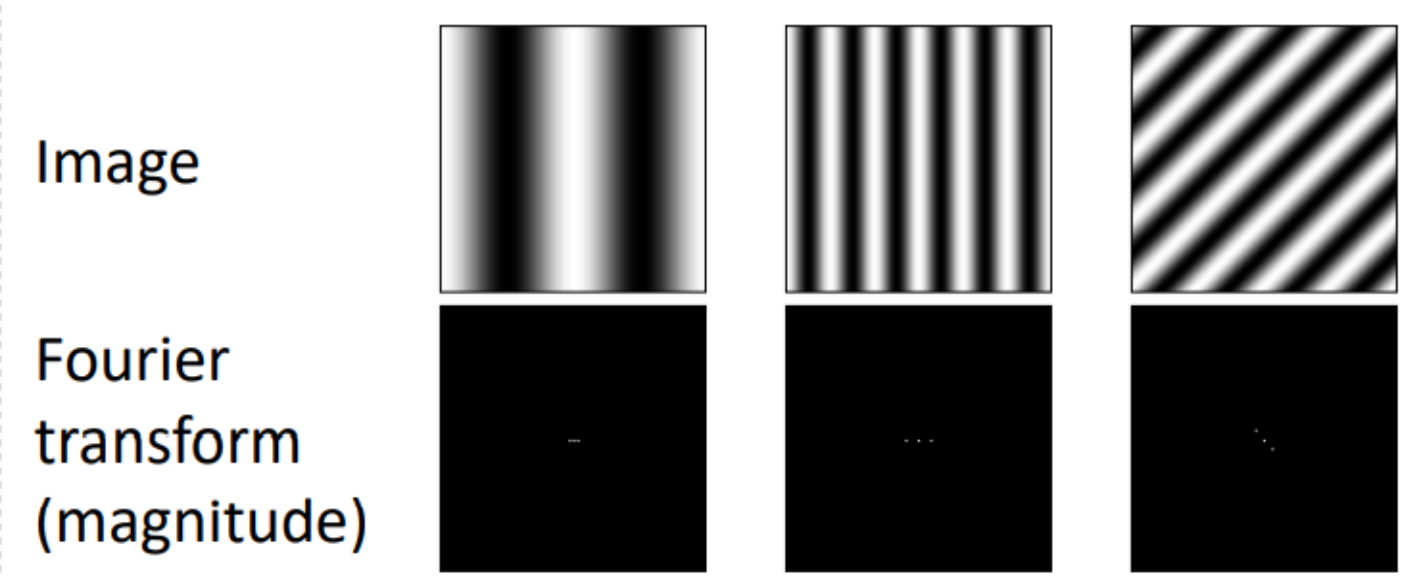
Fourier transforms

- Spatial domain → Frequency domain



Fourier transforms

- Spatial domain → Frequency domain
  - 逆傅里叶变换将频域转化为空间域
  - 频率域处理是先将图像变换到频率域，然后在频率域对图像进行处理，最后通过反变换将图像变为空间域。



Talk from the top scientist

AI 如何拯救人性