Convolutions and Kernels

Convolution

Convolutions will be used for several applications in this course :

- Anti-Aliasing,
- Sharpening of images
- Dynamic Range (overcome limitations of monitor: We only have 2^8 levels of intensity, which is very far from sufficient - but we will study how to display images

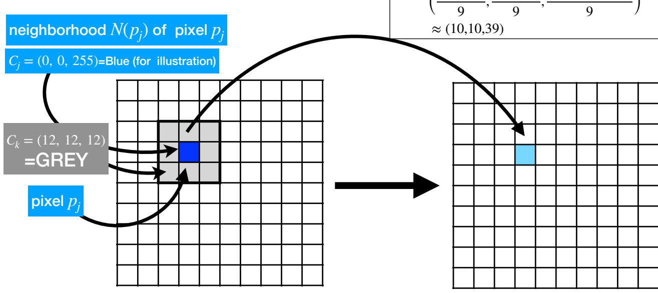
Big idea. Image could be improved, if when setting the value of a pixel, we also take into account values of nearby pixels

Neighborhood Filtering (Schematic)
Aka mean filter, aka smoothing

- **Let $N(p_j)$ be the 3×3 neighborhood of pixel p_j**
- **Sum the RGB value of all these pixels**
- **The average is the new value of p_j**

f(N$_j$)=average color in this region (neighborhood)=

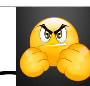
$$\left(\frac{(12\times 8)}{9}, \frac{(12\times 8)}{9}, \frac{255+(12\times 8)}{9}\right) \approx (10,10,39)$$



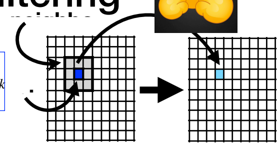
original image **filtered image**

Convolution

An Example: Mean Filtering



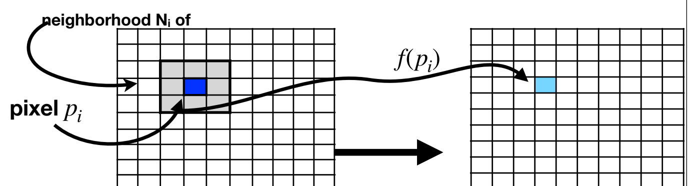
$$f\left(\textbf{pixel } p_i\right) = \sum_{\mathbf{p}_k \in N(p_i)} \overbrace{w_k}^{weight} \cdot \overbrace{C_k}^{color} = \sum_{\mathbf{p}_k \in N(p_i)} \frac{1}{9} C_k$$



https://www.geogebra.org/m/ta5cwm3a

- Mean filters sum colors of the pixels in a local neighborhood $N(p_i)$ of pixel p_i, and divide the by the total number (averaging)
- Where the $N(p_i)$ is a square, and pixels have same weight, we call these **box** filter.
- Sometimes we give less weight to pixels farther from p_i - tent filter , gaussian filter. **Weighted** average
- The weights $w_1 \dots w_k$ are **convex combination.** Meaning that they are all positive, and thier sum $w_1 + w_2 + \dots w_k = 1$. For example, $w_1 = w_2 = w_3 = \dfrac{1}{3}$.
- **Important**: the source image and target image has same number of pixels.

$$f\left(\textbf{pixel } p_i\right) = \sum_{\mathbf{p}_k \in N(p_i)} \overbrace{w_k}^{weight} \cdot \overbrace{C_k}^{color} = \sum_{\mathbf{p}_k \in N(p_i)} \frac{1}{9} C_k$$



original image **filtered image**

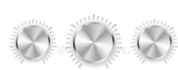
- Imagine placing a mask $\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$ (template) placed so its center on - the mask specifies

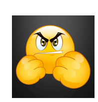
https://www.geogebra.org/m/ta5cwm3a

how much weight each pixel receives. This mask is called a **Kernel.**

- Other possible Kernels for this operation: $\frac{1}{12}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ or $\frac{1}{25}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$

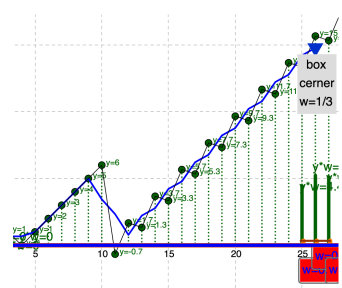
Kernels





https://www.geogebra.org/m/ta5cwm3a

- **Comparison In 1D - 2 box kernels, Tent and Gaussian. Note that with tent and gaussians we could also interpolate between the data points (e.g. if needed to increase the resolution of the image)**



Box Filtering

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

The matrix of weights is called a **Kernel**

$$1/9 * \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Box Filtering

$$1/9 * \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$$1/25 * \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$$

Kernels

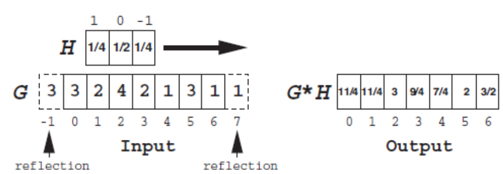
- Convolution employs a rectangular grid of coefficients, (that is, weights) known as a **kernel**
- Kernels are like a neighborhood mask, they specify which elements of the image are in the neighborhood and their relative weights.
- A kernel is a set of weights that is applied to corresponding input samples that are summed to produce the output sample.
- For **smoothing** purposes, the sum of weights must be 1 (convex combination).
- Promo: Sometimes some input value are not available (e.g. near boundaries) or we prefer not to be include them - we will have to adjust the kernels weights so the remain convex combination.

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \frac{1}{13}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \frac{1}{37}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 5 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

One-dimensional Convolution

- Can be expressed by the following equation, which takes a kernel (sometimes called "filter") H and **convolves** it with G: (note notation of convolution)

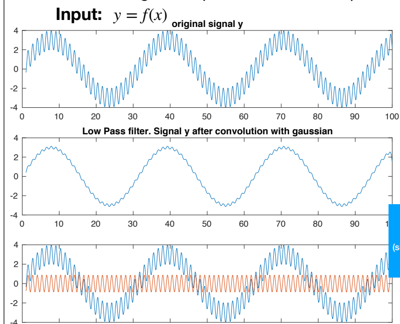
$$\hat{G}[i] = (\overbrace{G}^{input\ \ image} * \overbrace{H}^{kernel})[i] = \sum_{j=-1}^{j=+1} G[i-j] \cdot H[j]$$



Low-pass and high-pass filtering

The smoothing operation is always a low pass filter.
Only lower frequencies could pass.
It removes higher frequencies from the input.

Input: $y = f(x)$



We convolved the original signal f(x) a smoothing kernel H. For example

$$g(x) = \frac{f(x-1) + f(x) + f(x+1)}{3}$$

The output of the smoothing operation $g(x) = f(x) * H$ The higher frequencies are less noticeable: we need to move a lot (in x) to notice a large different in y

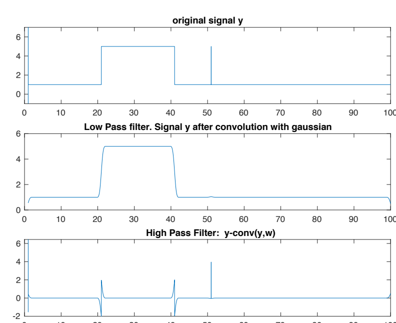
New idea: High-pass filter. $h(x) = f(x) - g(x)$ Only high frequencies pass (shown: Original signal (blue) and the result of the high pass filter (red))

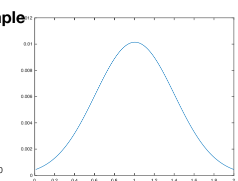
We remove (subtract) from the signal all lower frequencies

Twitter - could move very fast, but only small distances

Woofer - moves slowly but cold cover large distances

Low pass and hight pass filters - another example





We convolved the original signal y with this gaussian

2-Dimensional Version

- Given an image a and a kernel b with $(2r+1)^2$ values, the convolution of a with b is given below as a*b:

$$(a \bigstar b)[i,j] = \sum_{i'=i-r}^{i+r} \sum_{j'=j-r}^{j+r} a[i',j']\,b[i-i',j-j']$$

- The (i-i') and (j-j') terms can be understood as reflections of the kernel about the central vertical and horizontal axes.
- The kernel weights are multiplied by the corresponding image samples and then summed together.

A Note on Indexing

- Convolution **reflects** the filter to preserve orientation.
- **Correlation** does **not** have this reflection.
  - But we often use them interchangeably since most kernels are symmetric!!

Convolution reflects and shifts the kernel

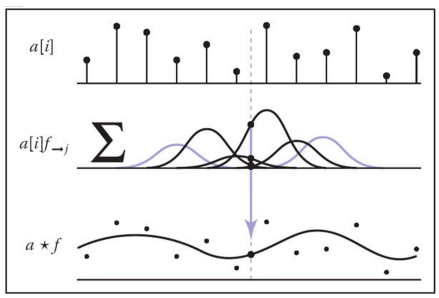
Given kernel H = $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$

$$\begin{matrix} 9 & 8 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 8 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

G*H

Convolution Can Also Convert from Discrete to Continuous

- Discrete signal a
- Continuous filter f
- Output a*f defined on positions x as opposed to discrete pixels i



$$(a \bigstar f)(x) = \sum_{i=\lceil x-r \rceil}^{\lfloor x+r \rfloor} a[i]\,f(x-i)$$

Types of Filters: Smoothing

Smoothing Spatial Filters

- Any weighted filter with positive values will smooth in some way, examples:

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

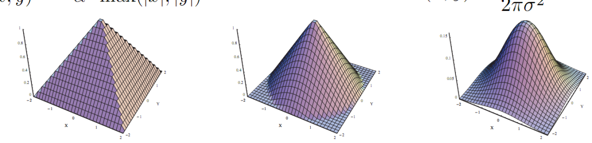
1	2	1
2	4	2
1	2	1

- Normally, we use integers in the filter, and then divide by the sum (computationally more efficient)
- These are also called **blurring** or **low-pass** filters

Smoothing Kernels

$$f(x,y) = -\alpha \cdot \max(|x|,|y|)$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$



(a) Pyramid. (b) Cone. (c) Gaussian.

$$f(x,y) = -\alpha \cdot \sqrt{x^2+y^2}$$

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

(a) Pyramid.

0	0	1	0	0
0	2	2	2	0
1	2	5	2	1
0	2	2	2	0
0	0	1	0	0

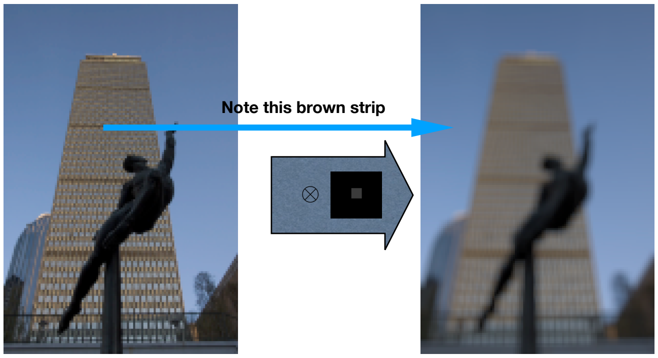
(b) Cone.

1	4	7	4	1
4	16	28	16	4
7	28	49	28	7
4	16	28	16	4
1	4	7	4	1

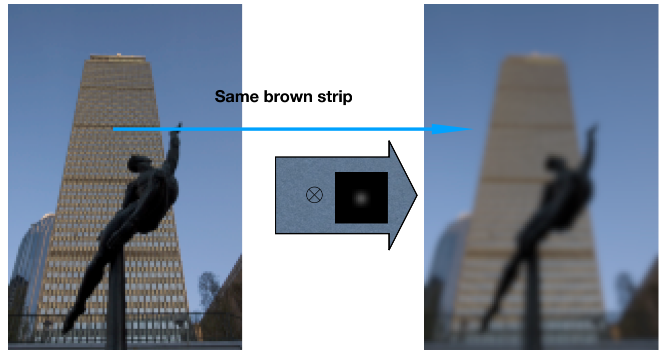
(c) Gaussian.

Table 6.1. Discretized kernels.

Box Filter



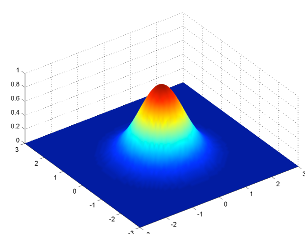
Gaussian Filter



Gaussians

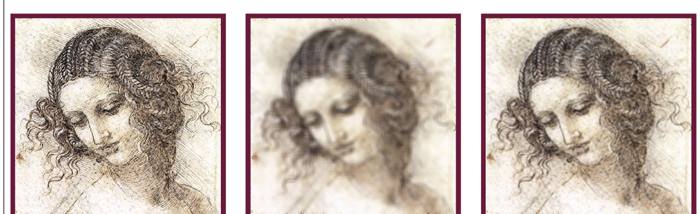
$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

- Gaussian kernel is parameterized on the standard deviation σ
- Large σ's reduce the center peak and spread the information across a larger area
- Smaller σ's create a thinner and taller peak
- Gaussians are smooth everywhere.
- Gaussians have infinite **support**
  - >0 everywhere
- But often truncate to 2σ or 3σ
- Volume =1 (sum of weights =1)



http://en.wikipedia.org/wiki/Gaussian_function

Smoothing Comparison

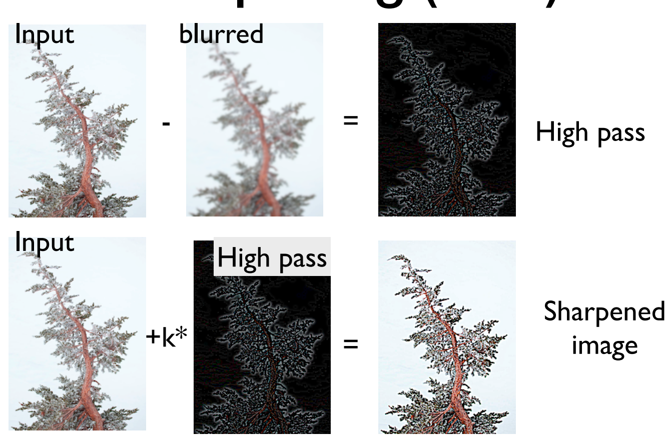


(a) Source image. (b) 17 × 17 Box. (c) 17 × 17 Gaussian.

Figure 6.10. Smoothing examples.

Types of Filters: Sharpening

Sharpening (Idea)



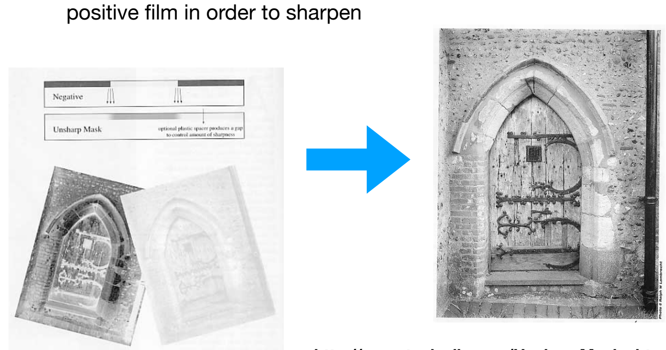
Another example

Original Image, Imaged convolved

Left: difference (only boundaries are non-black)
Right Imaged minus differences convolved

Unsharp Masks

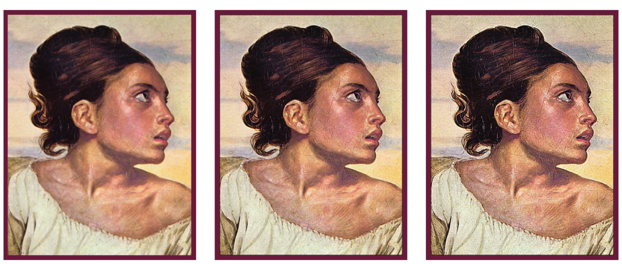
- Sharpening is often called "unsharp mask" because photographers used to sandwich a negative with a blurry positive film in order to sharpen



http://www.tech-diy.com/UnsharpMasks.htm

Edge Enhancement

- The parameter α controls how much of the source image is passed through to the sharpened image.



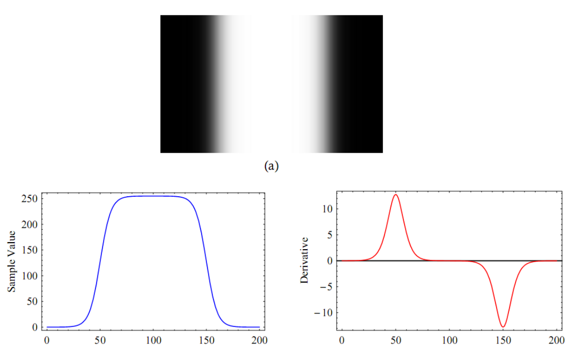
(a) Source image. (b) $\alpha = .5$. (c) $\alpha = 2.0$.

Figure 6.20. Image sharpening.

Defining Edges

- Sharpening uses negative weights to enhance regions where the image is changing rapidly
  - These rapid transitions between light and dark regions are called **edges**
- Smoothing reduces the strength of edges, sharpening strengthens them.
  - Also called **high-pass** filters
- Idea: smoothing filters are weighted averages, or integrals. Sharpening filters are weighted differences, or derivatives!

Edges



(a) (b) (c)

Figure 6.11. (a) A grayscale image with two edges, (b) row profile, and (c) first derivative.

(Review?) Derivatives via Finite Differences

- We can approximate the derivative with a kernel w:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+h,y)-f(x-h,y)}{2h} \approx \frac{f(x+1,y)-f(x-1,y)}{2}$$

$$\frac{\partial f}{\partial x} \approx w_{dx} \circ f \qquad w_{dx} = \begin{array}{|c|c|c|}\hline -\frac{1}{2} & 0 & \frac{1}{2} \\ \hline\end{array}$$

$$\frac{\partial f}{\partial y} \approx w_{dy} \circ f \qquad w_{dy} = \begin{array}{|c|}\hline -\frac{1}{2} \\ \hline 0 \\ \hline \frac{1}{2} \\ \hline\end{array}$$

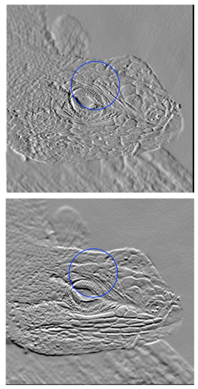
Lets convolves with one of the kernels

$$\frac{\partial}{\partial x} f(x,y) = \lim_{\Delta x \to 0} \frac{1}{\Delta x}\left\{f(x+\Delta x, y) - f(x,y)\right\} = \lim_{\Delta x \to 0} \frac{1}{2\Delta x}\left\{f(x+\Delta x, y) - f(x-\Delta x, y)\right\} =$$

$$\approx \frac{1}{2}\left\{f(x+1,y) - f(x-1,y)\right\} =$$

$$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix} \qquad [-1 \;\; 0 \;\; 1]$$

$$\begin{matrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix} \qquad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$



Gradients with Finite Differences

- These partial derivatives approximate the image gradient, ∇I.
- Gradients are the unique direction where the image is changing the most rapidly, like a slope in high dimensions
- We can separate them into components kernels G_x, G_y. $\nabla I = (G_x, G_y)$

$$\nabla I(x,y) = \begin{pmatrix} \delta I(x,y)/\delta x \\ \delta I(x,y)/\delta y \end{pmatrix}.$$

$$G_x = [1, 0, -1] \qquad G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix};$$

$$\nabla I = \begin{pmatrix} \delta I/\delta x \\ \delta I/\delta y \end{pmatrix} \simeq \begin{pmatrix} I \otimes G_x \\ I \otimes G_y \end{pmatrix}.$$

Figure 6.12. Image gradient (partial).

128	187	210	238	251
76	121	193	225	219
66	91	110	165	205
47	81	83	119	157
41	59	63	75	125

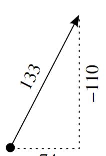
(a) Source Image.

117	104	26
44	74	95
36	38	74

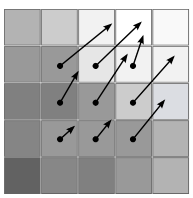
(b) $\delta I/\delta x$.

−96	−100	−73
−40	−110	−106
−32	−47	−90

(c) $\delta I/\delta y$.



(d) Center sample gradient.



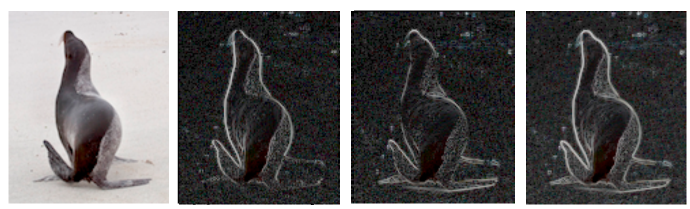
(e) Gradient.

151	144	77
59	133	142
48	60	117

(f) Magnitude of gradient.

Figure 6.14. Numeric example of an image gradient.

Gradients G_x, G_y



$|G_x|$ $|G_y|$ $|G|$

$$|G| = \sqrt{(G_x^2 + G_y^2)}$$

Second Derivatives (Sharpening, almost)

- Partial derivatives in x and y lead to two kernels:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

and, similarly, in the y-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

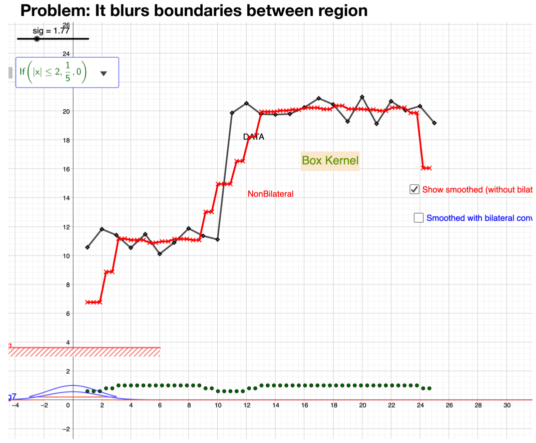
Compare with Sharpening filter: unbalanced counts!

$$\begin{bmatrix} -\alpha & -\alpha & -\alpha \\ -\alpha & (9+8\alpha) & -\alpha \\ -\alpha & -\alpha & -\alpha \end{bmatrix}$$

1	1	1
1	-9	1
1	1	1

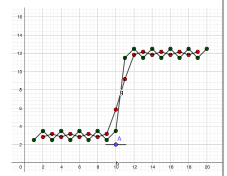
Back to smoothing (aka mean filter, averaging…)

Problem: It blurs boundaries between region



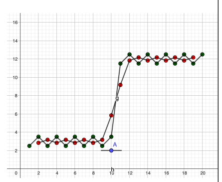
In non-bilateral kernel -
the smoothing operation blurs the boundaries between regions.

How to smooth if some value are unwanted /unavailable



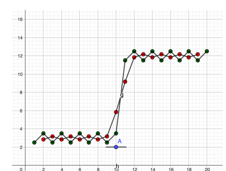
- $s(i) = \sum_{j\in \text{nbr of } i} w[i-j]\cdot f(j)$ and $\sum w_j = 1$ $\quad w_j \ge 0$
- First attempt: Define a useful neighbors.
- j only if if is near i, and its intensity is close to intensity at i. For example, include $j \in N(i)$ only if $|f(j) - f(j)| \le 10$:
- $s(i) = \sum_{\substack{j\in \text{nbr of } i \\ j \text{ useful nbr}}} w[i-j]\cdot f[j]$
- Problem: The sum of weights is much smaller than 1.
- Idea: Define $w'[i-j] = w[i-j] \Big/ \sum_{\substack{j\in \text{nbr of } i \\ j \text{ useful nbr}}} w[i-j]$.
- $s(i) = \sum_{\substack{j\in \text{nbr of } i \\ f(j) \text{ useful}}} w'[i-j]\cdot f[j]$

Lets simplify (for box filter)

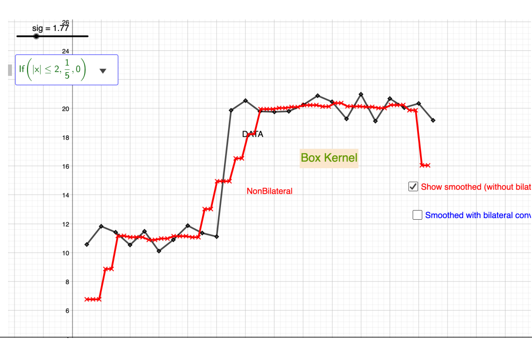


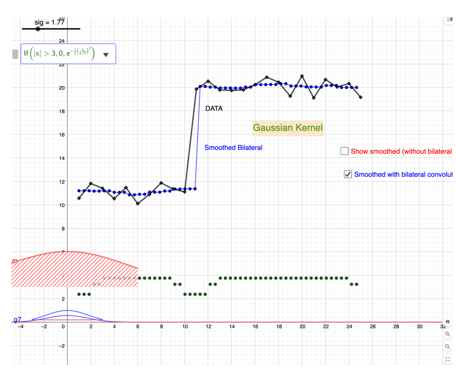
- Define $g(i,j) = \begin{cases} 1 & if\,|f(i) - f(j)| \le 10 \\ 0 & othewise(\end{cases}$
- Instead of $s(i) = \sum_{j\in \text{nbr of } i} f(j)w[i-j]$
- Define
  - $s(i) = \sum_{j\in \text{nbr of } i} f(j)\; w'[i-j]\; g(i,j)$
  - where
  - $w'[i-j] = w[i-j] \Big/ \sum_{j\in \text{nbr of } i} w[i-j]g(i,j)$

Example for box filter



- Define
  $g(i,j) = \begin{cases} 1 & if\,|f(i) - f(j)| \le 10 \\ 0 & othewise(\end{cases}$
- Instead of
  $s(i) = \big(f(i-1) + f(i) + f(i+1)\big) / 3$
- Define
  - $s(i) = \dfrac{f(j-1)g(i,i-1) + f(i) + f(i+1)g(i,i+1)}{1 + g(i,i-1) + g(i,i+1)}$
  - Note that the denominator $1 + g(i,i-1) + g(i,i+1)$ is either 1,2 or 3





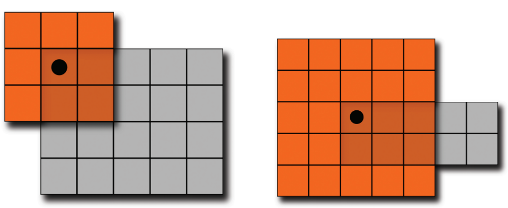
This will lead us to bilateral filters

This trick with normalizing the wights is very useful if the number of terms is the summation is not fixed (for example, near boundaries) or when sliding a window

Boundaries

Handling Image Boundaries

- What should be done if the kernel falls off of the boundary of the source image as shown in the illustrations below?



(a) Kernel at $I(0,0)$. (b) Kernel larger than the source.

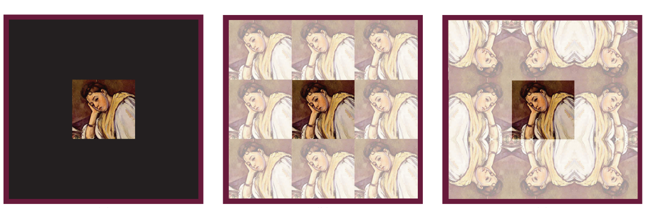
Figure 6.4. Illustration of the edge handling problem.

Handling Image Boundaries

- When pixels are near the edge of the image, neighborhoods become tricky to define
- Choices:
  1. Shrink the output image (ignore pixels near the boundary)
  2. Expanding the input image (padding to create values near the boundary which are "meaningful")
  3. Shrink the kernel (skip values that are outside the boundary, and reweigh accordingly)

Boundary Padding

- When one pads, they pretend the image is large and either produce a constant (e.g. zero), or use circular / reflected indexing to tile the image:



(a) (b) (c)

Figure 6.5. (a) Zero padding, (b) circular indexing, and (c) reflected indexing.