Convolutions and Kernels

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

With HDR + Tone Mapping

Neighborhood Filtering (Schematic)
Aka mean filter, aka smoothing

Let $N(p_i)$ be the $3 \times 3$ neighborhood of pixel $p_i$
- Sum the RGB value of all these pixels
- The average is the new value of $p_i$

An Example: Mean Filtering

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 are convex combination. Meaning that they are all positive, and thier sum $w_1 + w_2 + \ldots w_k = 1$. For example, $w_1 = w_2 = w_3 = \frac{1}{3}$.
- Important: the source image and target image has same number of pixels.

Kernels

- 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**

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

**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).
- Prom: Sometimes some input value are not available (e.g. near boundaries) or we prefer not to include them - we will have to adjust the kernels weights so the remain convex combination.

\[
\begin{bmatrix}
1/9 & 1/9 & 1/9 \\
1/13 & 1/13 & 1/13 \\
1/13 & 1/13 & 1/13 \\
\end{bmatrix}
\]

**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)

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

**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:** \( y = f(x) \) a smoothing kernel \( H \). For example

\[
g(x) = (x-1) + f(x) + f(x+1)
\]

The output of the smoothing operation.

**Low Pass Filter:** Signal after convolution with gaussian

We convolved the original signal \( y \) with this gaussian

**High Pass Filter:** Signal after convolution with gaussian

We convolved the original signal \( y \) with this gaussian

**Twitter** - could move very fast, but only small distances

**Woofer** - moves slowly but could cover large distances
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 \( a \ast b \):

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

- 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!

\[
\text{Given kernel } H = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}
\]

Convolution reflects and shifts the kernel

A Note on Indexing

Convolution reflects and shifts the kernel

Types of Filters: Smoothing

- Discrete signal \( a \)
- Continuous filter \( f \)
- Output \( a \ast f \) defined on positions \( x \) as opposed to discrete pixels \( i \)

\[
(a \ast f)(x) = \sum_{i=-r}^{r} a[i] f(x - i)
\]

Smoothing Spatial Filters

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

| \[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
| \[
\begin{array}{ccc}
1 & 2 & 1 \\
1 & 2 & 1 \\
1 & 2 & 1 \\
\end{array}
\]

- 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

- \( f(x, y) = -\alpha \cdot \max(|x|, |y|) \)
- \( G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \)

| \[
\begin{array}{cccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
3 & 6 & 9 & 6 & 3 \\
2 & 4 & 6 & 4 & 2 \\
1 & 2 & 3 & 2 & 1 \\
\end{array}
\]
| \[
\begin{array}{cccc}
0 & 0 & 1 & 0 & 0 \\
0 & 2 & 2 & 2 & 0 \\
1 & 2 & 5 & 2 & 1 \\
4 & 16 & 28 & 16 & 4 \\
1 & 4 & 7 & 4 & 1 \\
\end{array}
\]

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

Table 6.1: Discretized kernels.
Box Filter

Gaussian Filter

Gaussians

Gaussians

Smoothing Comparison

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 $\alpha$ controls how much of the source image is passed through to the sharpened image.

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

(Review?) Derivatives via Finite Differences

- We can approximate the derivative with a kernel $w$:

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

$$\frac{\partial f}{\partial x} \approx w_{dx} \odot f \quad w_{dx} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

$$\frac{\partial f}{\partial y} \approx w_{dy} \odot f \quad w_{dy} = \begin{bmatrix} 0 & \frac{1}{2} \end{bmatrix}$$
Let's convolves with one of the kernels
\[
\frac{df}{dx}(x,y) = \lim_{\Delta x \to 0} \frac{1}{\Delta x} \left( f(x + \Delta x, y) - f(x, y) \right) = \lim_{\Delta x \to 0} \frac{1}{2\Delta x} \left( f(x + 1, y) - f(x - 1, y) \right) =
\]
\[
\begin{bmatrix}
-1 & 0 & 1
\end{bmatrix}
\]

Gradients with Finite Differences
- These partial derivatives approximate the image gradient, \(\nabla f\).
- 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 \(G_x, G_y\).

\[
\nabla f(x,y) = \left( \frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right)
\]

\[
G_x = [1, 0, -1]
\]

\[
G_y = \begin{bmatrix}
0 \\
-1
\end{bmatrix}
\]

\[
\nabla f = (f \otimes G_x, f \otimes G_y)
\]

Gradients \(G_x, G_y\)

(a) Source Image. (b) \(\delta I / \delta x\). (c) \(\delta I / \delta y\).

(d) Center sample gradient. (e) Gradient. (f) Magnitude of gradient.

Figure 6.14. Numeric example of an image gradient.

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:

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

Second Derivatives: Sharpening filter; unbalanced counts!

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -8 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

Compare with

Problem: It blurs boundaries between region

\[
\begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\]

Back to smoothing (aka mean filter, averaging...)

Box Kernel

Box smoothed original data

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

How to smooth if some value are unwanted/unavailable

- Define a useful neighbors.
- Only if it is near \( i \) and its intensity is close to intensity at \( i \).
- For example, include only if:
- 
- Problem: The sum of weights is much smaller than 1.
- Idea: Define

\[
  s(i) = \sum_{j \in \text{nbr of } i} w[i - j] \cdot f(j)
\]

- Define

\[
  s(i) = \sum_{j \in \text{nbr of } i} f(j) w[i - j] \cdot g(i, j)
\]

- where

\[
  w[i - j] = w[i - j] / \sum_{j \in \text{nbr of } i} w[i - j] g(i, j)
\]

Lets simplify (for box filter)

- Define \( g(i, j) = \begin{cases} 
1 & \text{if } |f(i) - f(j)| \leq 10 \\
0 & \text{otherwise}
\end{cases} \)
- Instead of \( s(i) = \sum_{j \in \text{nbr of } i} f(j) \cdot w[i - j] \)
- Define

\[
  s(i) = \sum_{j \in \text{nbr of } i} f(j) \cdot w[i - j] \cdot g(i, j)
\]

Example for box filter

- Define \( g(i, j) = \begin{cases} 
1 & \text{if } |f(i) - f(j)| \leq 10 \\
0 & \text{otherwise}
\end{cases} \)

- Instead of \( s(i) = \sum_{j \in \text{nbr of } i} f(j) \cdot w[i - j] \)
- Define

\[
  s(i) = \frac{f(i - 1) + f(i) + f(i + 1)}{3} \cdot g(i, i - 1) + g(i, i + 1)
\]

- 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 weights is very useful if the number of terms is the summation is not fixed (for example, near boundaries) or when sliding a window.

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?

![Illustrations](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 reweights 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:

![Figure](a) Zero padding. (b) Circular indexing. (c) Reflected indexing.

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