Lecture 10: Convolution

Mark Hasegawa-Johnson

ECE 401: Signal and Image Analysis

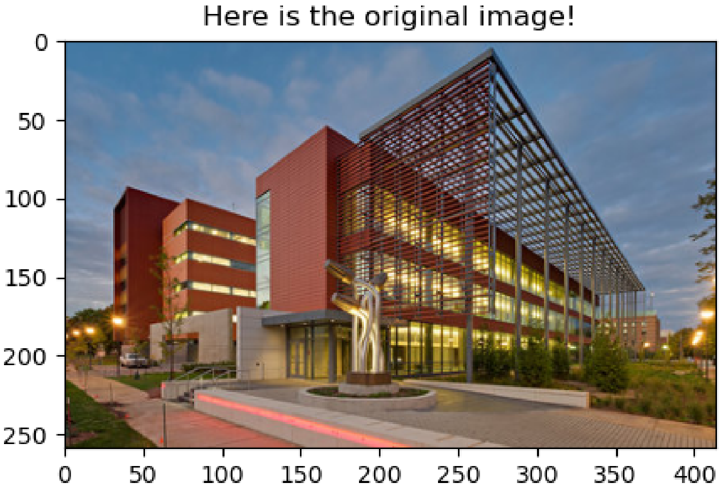
Outline

Outline of today's lecture

1. Local averaging
2. Convolution
3. Differencing
4. Edge Detection

Outline

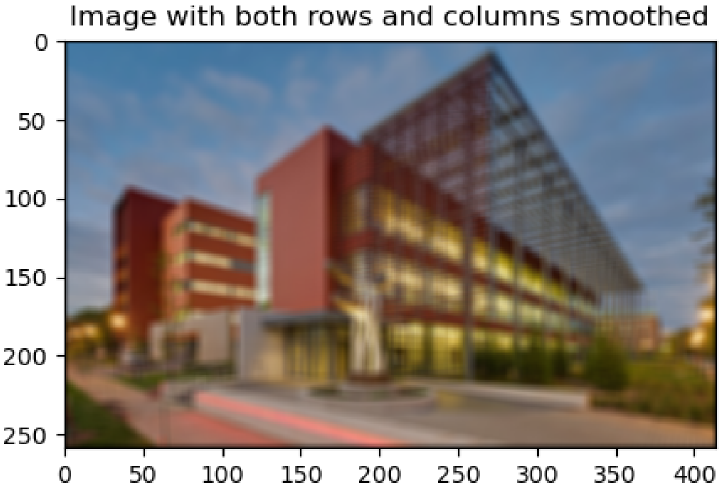
How do you treat an image as a signal?



How do you treat an image as a signal?

- An RGB image is a signal in three dimensions: $f[i, j, k] =$ intensity of the signal in the i^{th} row, j^{th} column, and k^{th} color.
- $f[i, j, k]$, for each (i, j, k), is either stored as an integer or a floating point number:
  - Floating point: usually $x \in [0, 1]$, so $x = 0$ means dark, $x = 1$ means bright.
  - Integer: usually $x \in \{0, \ldots, 255\}$, so $x = 0$ means dark, $x = 255$ means bright.
- The three color planes are usually:
  - $k = 0$: Red
  - $k = 1$: Blue
  - $k = 2$: Green

Local averaging



Local averaging

- "Local averaging" means that we create an output image, $y[i,j,k]$, each of whose pixels is an **average** of nearby pixels in $f[i,j,k]$.
- For example, if we average along the rows:

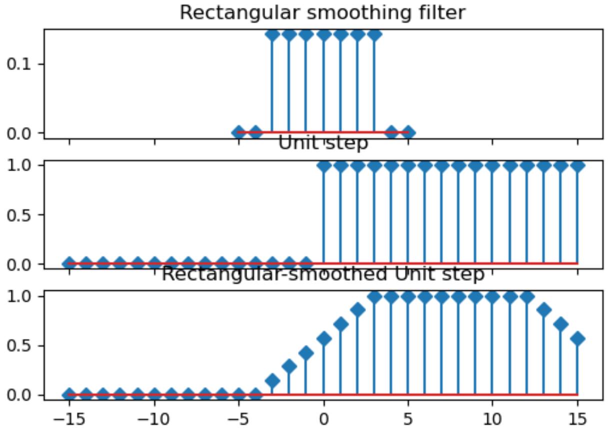
$$y[i,j,k] = \frac{1}{2M+1} \sum_{j'=j-M}^{j+M} f[i,j',k]$$

- If we average along the columns:

$$y[i,j,k] = \frac{1}{2M+1} \sum_{i'=i-M}^{i+M} f[i',j,k]$$

Local averaging of a unit step

The top row are the averaging weights. If it's a 7-sample local average, $(2M+1)=7$, so the averaging weights are each $\frac{1}{2M+1}=\frac{1}{7}$. The middle row shows the input, $f[n]$. The bottom row shows the output, $y[n]$.



Outline

Weighted local averaging

- Suppose we don't want the edges quite so abrupt. We could do that using "weighted local averaging:" each pixel of $y[i,j,k]$ is a **weighted average** of nearby pixels in $f[i,j,k]$, with some averaging weights $g[n]$.
- For example, if we average along the rows:

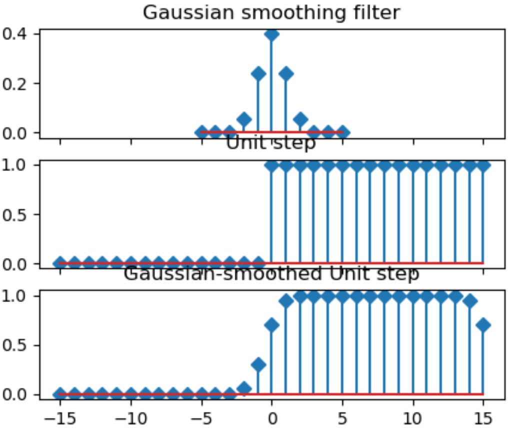
$$y[i,j,k] = \sum_{m=j-M}^{j+M} g[j-m]f[i,m,k]$$

- If we average along the columns:

$$y[i,j,k] = \sum_{i'=i-M}^{i+M} g[i-m]f[m,j,k]$$

Weighted local averaging of a unit step

The top row are the averaging weights, $g[n]$. The middle row shows the input, $f[n]$. The bottom row shows the output, $y[n]$.



Outline

Convolution

- A **convolution** is exactly the same thing as a **weighted local average**. We give it a special name, because we will use it very often. It's defined as:

$$y[n] = \sum_m g[m]f[n-m] = \sum_m g[n-m]f[m]$$

- We use the symbol $*$ to mean "convolution:"

$$y[n] = g[n]*f[n] = \sum_m g[m]f[n-m] = \sum_m g[n-m]f[m]$$

Convolution

$$\boxed{y[n] = g[n] * f[n] = \sum_m g[m]f[n-m] = \sum_m g[n-m]f[m]}$$

Here is the pseudocode for convolution:

1. For every output n:
    1. Reverse $g[m]$ in time, to create $g[-m]$.
    2. Shift it to the right by n samples, to create $g[n-m]$.
    3. For every m:
        1. Multiply $f[m]g[n-m]$.
    4. Add them up to create $y[n] = \sum_m g[n-m]f[m]$ for this particular n.
2. Concatenate those samples together, in sequence, to make the signal y.

Convolution: how should you implement it?

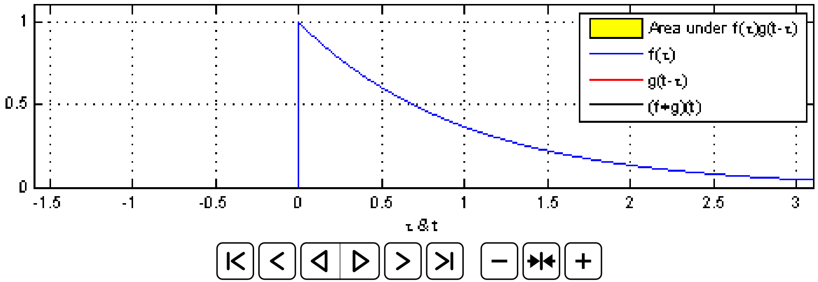
- When writing code: use the numpy function, `np.convolve`. In general, if numpy has a function that solves your problem, you are *always* permitted to use it.
- When solving problems with pencil and paper: use *graphical convolution*.

Outline

Graphical Convolution

1. Choose one of the two functions whose breakpoints are easier to shift (i.e., it breaks at easy values like $n = 0$), and call that $f[n]$. Call the other function $g[n]$.
2. Plot $g[m]$ as a function of m.
3. Underneath, plot $f[n-m]$ as a function of m for some particular n.
4. Under that, plot $g[m]f[n-m]$ for the same particular n.
5. Use your plot as a guide to help you write the equation $\sum_m g[m]f[n-m]$ in a solvable form. Solve it to find $y[n]$.
6. If this gives you enough information to find $y[n]$ for every other n, then do so. If there's some other n that's not yet obvious to you, then repeat above process for the other n.

Graphical Convolution: A Video from Wikipedia



by Brian Amberg, CC-SA 3.0,

`https://commons.wikimedia.org/wiki/File:Convolution_of_spiky_function_with_box2.gif`

Quiz

Do the quiz! Go to the course webpage, and try the quiz.

Outline

Differencing is convolution, too

Suppose we want to compute the local difference:

$$y[n] = f[n] - f[n-1]$$

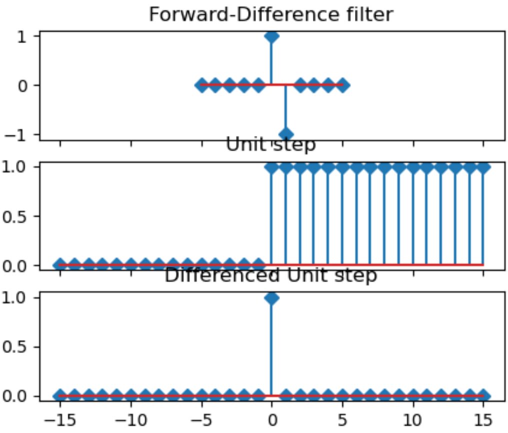
We can do that using a convolution!

$$y[n] = \sum_m f[n-m]h[m]$$

where

$$h[m] = \begin{cases} 1 & m = 0 \\ -1 & m = 1 \\ 0 & \text{otherwise} \end{cases}$$

Differencing as convolution



Outline

Weighted differencing as convolution

- The formula $y[n] = f[n] - f[n-1]$ is kind of noisy. Any noise in $f[n]$ or $f[n-1]$ means noise in the output.
- We can make it less noisy by
  1. First, compute a weighted average:

$$y[n] = \sum_m f[m]g[n-m]$$

  2. Then, compute a local difference:

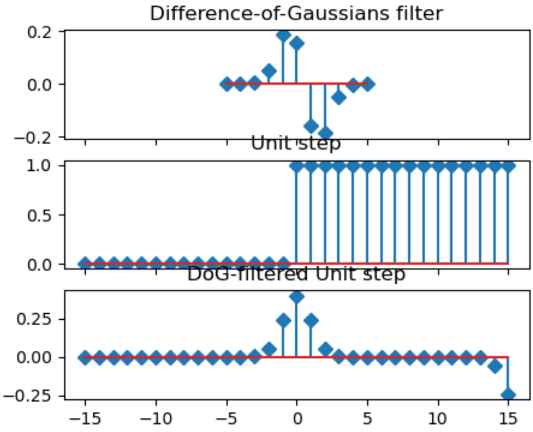
$$z[n] = y[n] - y[n-1] = \sum_m f[m]\left(g[n-m] - g[n-1-m]\right)$$

This is exactly the same thing as convolving with

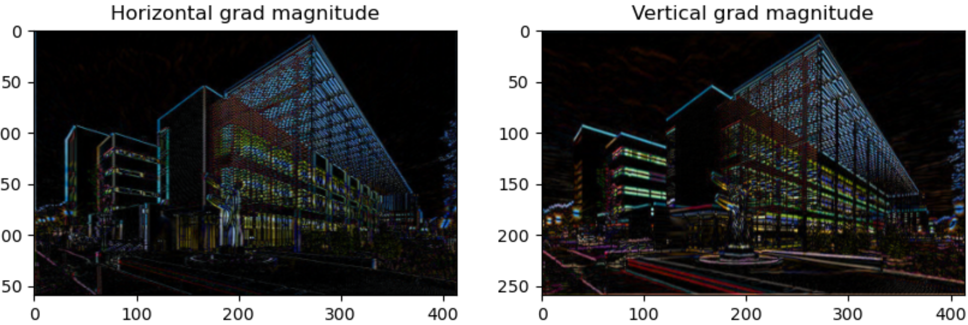
$$h[n] = g[n] - g[n-1]$$

A difference-of-Gaussians filter

The top row is a "difference of Gaussians" filter, $h[n] = g[n] - g[n-1]$, where $g[n]$ is a Gaussian. The middle row is $f[n]$, the last row is the output $z[n]$.



Difference-of-Gaussians filtering in both rows and columns



Outline

Image gradient

- Suppose we have an image $f[i,j,k]$. The 2D image gradient is defined to be

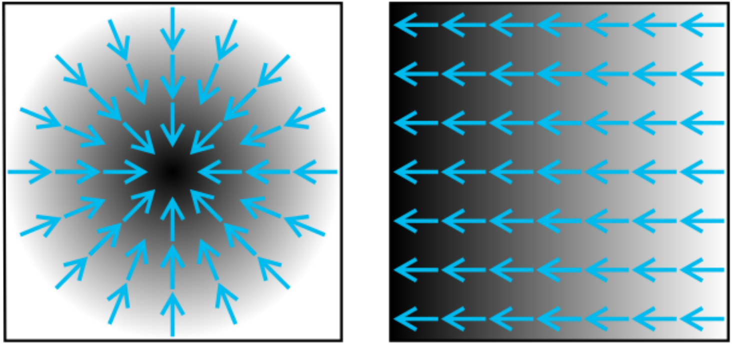
$$\vec{G}[i,j,k] = \left(\frac{df}{di}\right)\hat{i} + \left(\frac{df}{dj}\right)\hat{j}$$

  where $\hat{i}$ is a unit vector in the i direction, $\hat{j}$ is a unit vector in the j direction.
- We can approximate these using the difference-of-Gaussians filter, $h_{dog}[n]$:

$$\frac{df}{di} \approx G_i = h_{dog}[i] * f[i,j,k]$$
$$\frac{df}{dj} \approx G_j = h_{dog}[j] * f[i,j,k]$$

The gradient is a vector

The image gradient, at any given pixel, is a vector. It points in the direction of increasing intensity (this image shows "dark" = greater intensity).



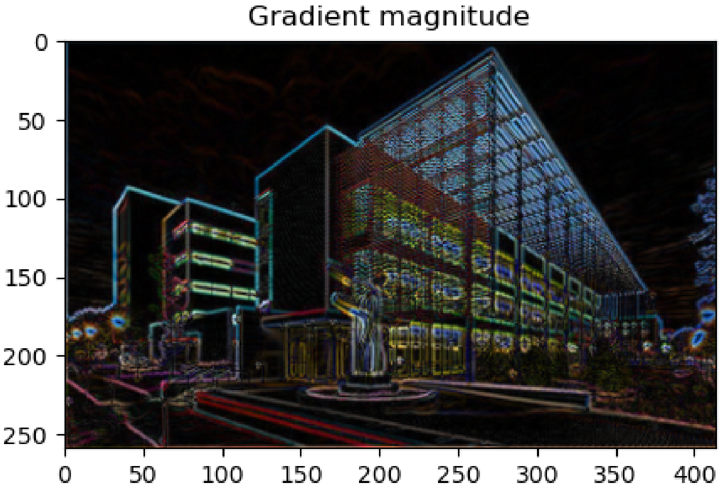
By CWeiske, CC-SA 2.5, `https://commons.wikimedia.org/wiki/File:Gradient2.svg`

Magnitude of the image gradient

- The image gradient, at any given pixel, is a vector.
- It points in the direction in which intensity is increasing.
- The magnitude of the vector tells you how fast intensity is changing.

$$\|\vec{G}\| = \sqrt{G_i^2 + G_j^2}$$

Magnitude of the gradient = edge detector



Outline

Summary

$$y[n] = g[n] * f[n] = \sum_m g[m]f[n-m] = \sum_m g[n-m]f[m]$$