

CSci 4270 and 6270
Spring 2025
Lecture 6: Image Processing and Neighborhood
Operations
January 27, 2025


What We Have Covered

- Pixel values, color spaces and histograms
- Pixel-by-pixel operations such as thresholding and intensity stretching.
- Linear algebra
- Lines and estimation
- Transformations *& cameras*

Where We Are Heading

- *Feature Extraction:* Extract and describe geometric entities from images
 - Edges
 - Corners
- We will discuss the preprocessing steps we apply to images before feature extraction:
 - This is tiny subset of the field of *image processing*.
- Convolutional neural networks are motivated by some of the algorithms we are going to discuss here.

Today's Topics

- Linear neighborhood operations — convolution:
 - Smoothing
 - Differentiation
 - ~~Pyramids~~ 
- Non-linear operations:
 - Mathematical morphology
 - Median filtering
- Application: simple change detection

A Jupyter notebook will be demonstrated in class and posted along with these notes.

“Neighborhood Operation”

- Everything we discuss today will be a computation that replaces the values at each pixel value by a function computed over a neighborhood of the pixel.
- We want our neighborhood operations to accomplish (at least) two tasks as the basis for our feature extraction:
 - We want to smooth the image to make noise and minor features less prominent.
 - We want to locate potential boundaries:
 - * Locations where the intensity is changing most rapidly.
 - * Thinking in terms of calculus, these are locations of largest (in magnitude) first derivatives or zero second derivatives.
 - We’ll play with the image in Figure 1.

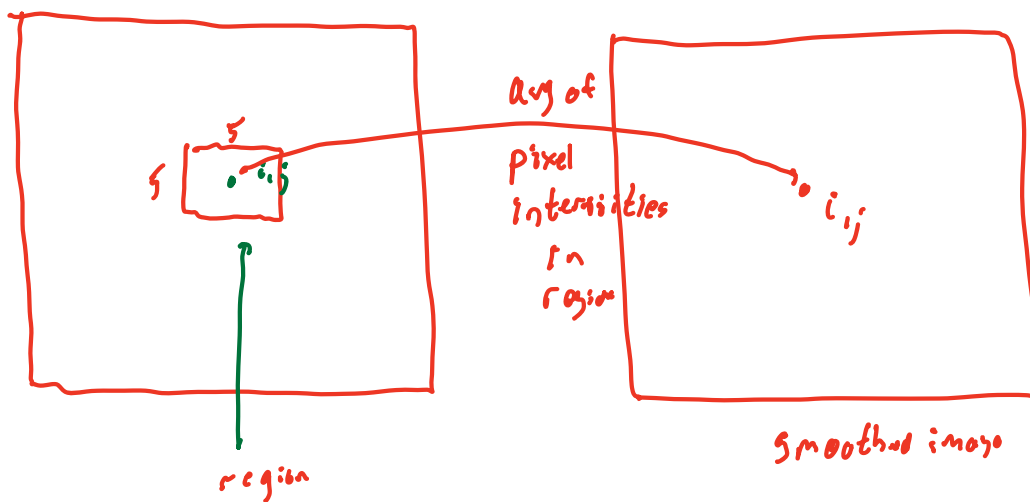


Figure 1: Examples of a picture of my (then) dogs more than ten years ago.

- We will we see this style of computation again in neural networks.

Noise Suppression as (Weighted) Averaging

- If all the pixels surrounding a given pixel are from the same relatively homogeneous image region, then we can reduce the noise and suppress detail by replacing the pixel with the average of its intensity and that of its neighbors.
 - * For example, we could center a 3x3 or a 5x5 neighborhood at each pixel.
- Conceptually, this is done simultaneously at each pixel in the image, requiring a second image on which to record the results.



– See Figure 2

blurr



Figure 2: The result of convolving the dogs image with a 5x5 box filter (left) and a 9x9 box filter (right).

Averaging as Convolution

Mathematically, we describe this as follows:

- Let f be the original image, and g be the output image.
- Let h be the image or function of weights.
- Then we write:

$$\underline{g(i, j)} = \sum_{\boxed{k, l}} \underline{f(i + k, j + l)} \underline{h(k, l)},$$

k, l are offsets
in image

or more compactly as

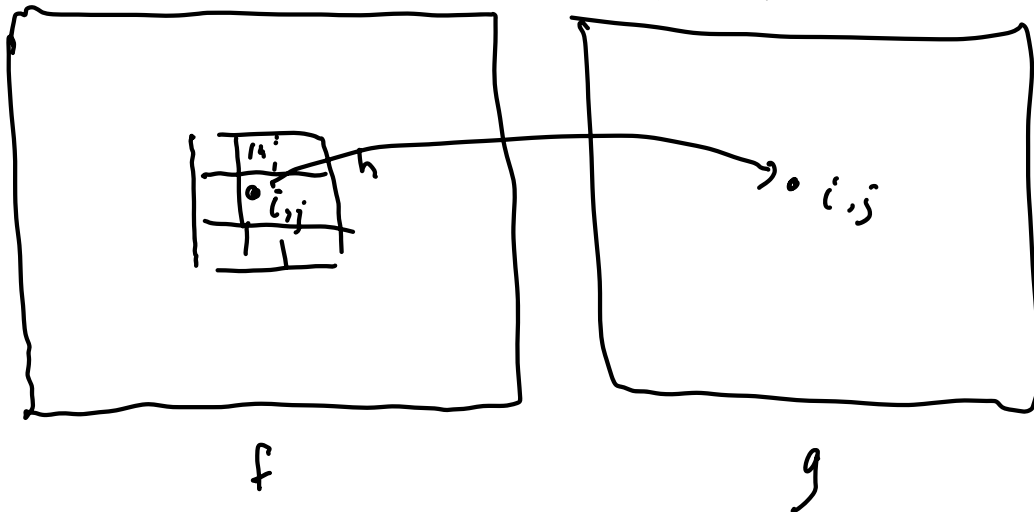
$$g = f * h.$$

$$h = \begin{array}{c|c|c} \begin{array}{c} -1 \\ 0 \\ 1 \end{array} & \begin{array}{c} 0 \\ 1 \\ 0 \end{array} & \begin{array}{c} 1 \\ 0 \\ 1 \end{array} \\ \hline \begin{array}{c} -1 \\ 0 \\ 1 \end{array} & \begin{array}{c} 1/4 \\ 1/4 \\ 1/4 \end{array} & \begin{array}{c} 1/4 \\ 1/4 \\ 1/4 \end{array} \\ \hline \begin{array}{c} -1 \\ 0 \\ 1 \end{array} & \begin{array}{c} 1/4 \\ 1/4 \\ 1/4 \end{array} & \begin{array}{c} 1/4 \\ 1/4 \\ 1/4 \end{array} \end{array}$$

Technically, this is called the *correlation* of f with h — we'd need $-k$ and $-l$ to for it to be a convolution — but the differences are not important to our purposes.

$$k = -1, 0, 1$$

$$l = -1, 0, 1$$



- In the case of local averaging, h is typically defined (non-zero!) over a small domain, such as $[-2, 2] \times [-2, 2]$ for averaging over a 5×5 neighborhood.

5×5

- Convolution is a linear operation, which effectively means that the value at each pixel is a dot product between the neighborhood values and the kernel weights.

| | -2 | -1 | 0 | 1 | 2 |
|----|----------------|----------------|----------------|---|----------------|
| -2 | $\frac{1}{25}$ | $\frac{1}{25}$ | $\frac{1}{25}$ | | |
| -1 | | | | | |
| 0 | | | $\frac{1}{25}$ | | |
| 1 | | | | | |
| 2 | | | | | $\frac{1}{25}$ |

Weight Dropoff

- The box filter produces an average over a neighborhood.
- Instead we might want a weighted average, with more weight given to the center pixel and less to the surrounding pixels.
- The question becomes, what is the appropriate weighting function?
- The answer, for several reasons, is the *Gaussian function*.

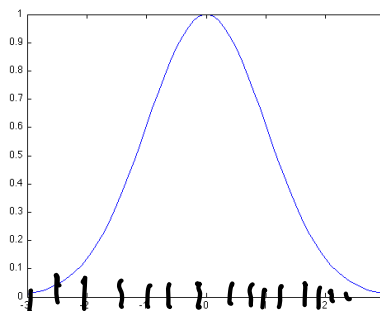
Gaussian Weighting — Continuous Form — 1d

- Continuous form in 1d

$$g(x; \sigma) = \exp(-0.5x^2/\sigma^2).$$

σ =
scale

$$e^{-\frac{x^2}{2\sigma^2}}$$

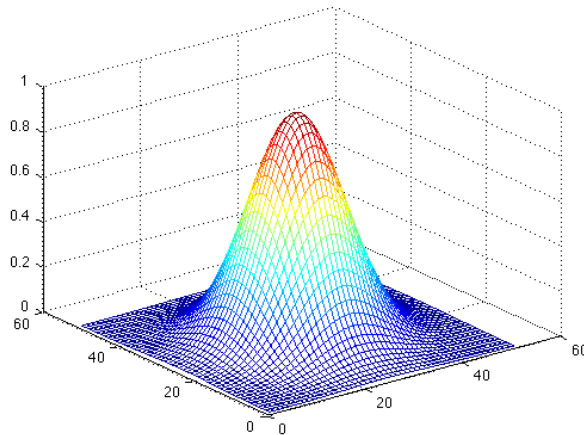


Gaussian Weighting — Continuous Form — 2d

- Continuous form in 2d

$$g(x, y; \sigma) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) = \underbrace{\exp\left(-\frac{x^2}{2\sigma^2}\right)} \underbrace{\exp\left(-\frac{y^2}{2\sigma^2}\right)}.$$

separable



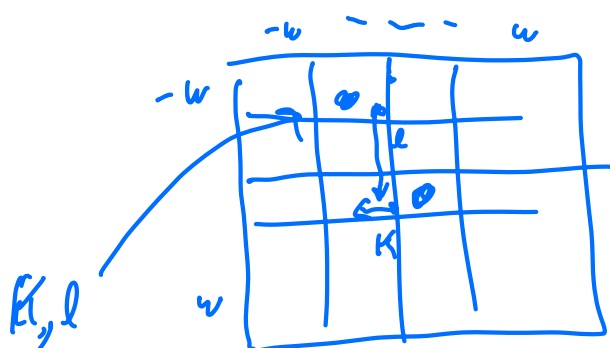
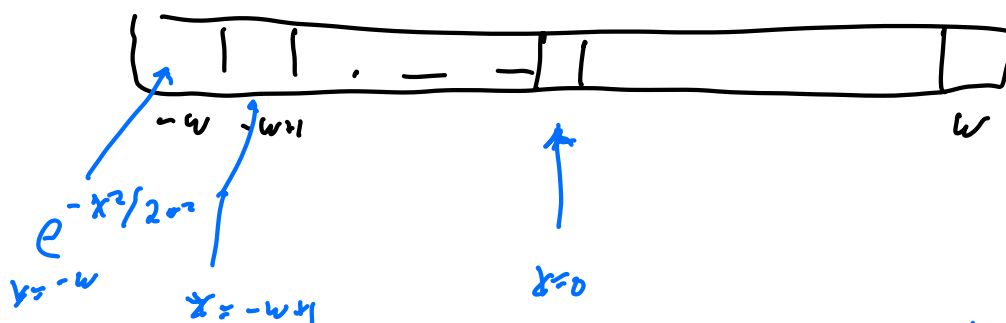
Gaussian Weighting — Discrete Form

- Gaussian is non-zero over an infinite domain!
- Practically, however, it can be treated as 0 for values of x at and beyond a given multiple of σ , usually 2, 2.5 or 3. In particular, at $x = 3\sigma$ $g(x; \sigma) \approx 0.011$, at $x = 2.5\sigma$ $g(x; \sigma) \approx 0.044$, while at $x = 0$ $g(0; \sigma) = 1$.
- To form the discrete Gaussian, we sample and normalize so that the sum of the weights is 1.
- For example, here is a vector of Gaussian weights with $\sigma = 1.5$ and a cut-off of $w = 2\sigma$

$$w = \lfloor 2\sigma \rfloor$$

| -3 | -2 | -1 | 0 | 1 | 2 | 3 |
|-------|-------|-------|-------|-------|-------|-------|
| 0.037 | 0.111 | 0.217 | 0.271 | 0.217 | 0.111 | 0.037 |

Kernel (1d form)



Normalize
at the
end

Increasing Sigma Increases the Smoothing

- The $\sigma = 1.5$ kernel has width $2w + 1 = 7$ (or 9 if a 3σ cut-off is used), whereas the $\sigma = 3$ kernel has width $2w + 1 = 15$ (or 19 if a 3σ cut-off is used).
- The practical effect is increasingly smoothed and blurred images as σ increases
- Examples are shown in Figure 3.



$\sigma = 1$



$\sigma = 2$



$\sigma = 4$



$\sigma = 8$

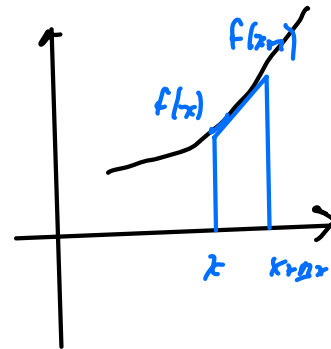
Figure 3: Examples of convolution with a Gaussian kernel at different scales.

Discrete Partial Differentiation as Convolution

- Recall the limit definition of the derivative:

Δx is
distance in pixels

$$f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$



- We will use this in class to derive a discrete approximation to the derivative as

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

- Aside: the measurements here are in "pixel" units.
- This produces the following simple correlation kernel for differentiation:

| | | |
|------|---|-----|
| -1 | 0 | 1 |
| -1/2 | 0 | 1/2 |

Smallest Δx is? 1

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

Kernel

| | | |
|---|----|---|
| 0 | -1 | 1 |
|---|----|---|

good at
 $x = 0.5$

Symmetric

$$\lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x - \Delta x)}{2 \Delta x}$$

$\Delta x = 1$

$$\frac{f(x+1) - f(x-1)}{2}$$

Conv
Kernel

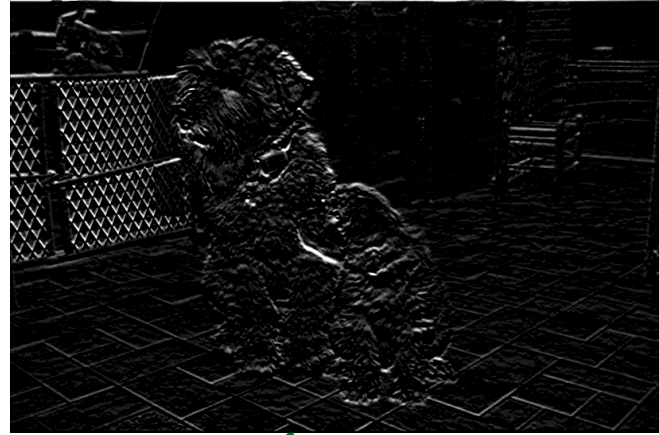
Partial Derivatives

In two dimensions,

- We obtain $\partial f / \partial x$ with the above kernel.
- We obtain $\partial f / \partial y$ by rotating the kernel 90° clockwise.



$\left[\partial f / \partial x \right]$



$\left[\partial f / \partial y \right]$

Figure 4: Examples of applying partial derivatives in the x and y directions.

Looking Ahead to Edge and Corner Detection

In Lecture 7 we will look at:

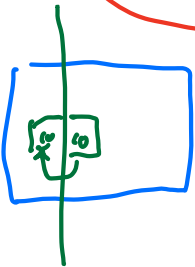
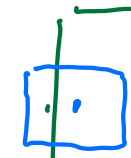
- Combining partial derivatives to compute gradient vectors
- Second derivative operators: Laplacian and Hessian
- Developing methods to “extract” edges and corners based on the results of smoothing and differentiation.

Implementation: Boundary Effects Can Be Handled in Several Ways

- Since the image domain is not infinite, there will be image locations (i, j) where computation of the convolution requires locations $(i+k, j+l)$ that are outside the image domain.
- This can be handled in several ways:

- Assign the convolution result to 0 at these locations. ← derivatives
- Mirror the values. For example, at the upper and left boundaries of the image $f(r, c) = f(-r, -c)$. ← Smoothing
- Assign $f(r, c) = 0$ to locations (r, c) outside the image domain for the purposes of computing the convolution.
- Wrap-around, treating the image as a torus. If the image has m rows and n columns, then when $r < 0$ and $c < 0$, $f(r, c) = f(m + r, n + c)$.

- Think about which makes more sense for smoothing and makes more sense for differentiation



What do you
do about kernels
and pixel locs that
require values outside image

Motivations for Non-Linear Operations

- Morphological operations are most often applied to binary images — perhaps obtained through thresholding:
 - Example we will look at is change detection
- Different way to clean up images

From Linear to Non-Linear Operations

- Consider our convolution summation

$$g(i, j) = \sum_{k, l} f(i + k, j + l) h(k, l)$$

- Now, let's revise our equations so that we make the summation explicitly consider locations where $h(k, l) \neq 0$. We get

$$g(i, j) = \sum_{(k, l) \text{ s.t. } h(k, l) \neq 0} f(i + k, j + l) h(k, l)$$

binary

$$h(k, l) = 1 \text{ or } 0$$

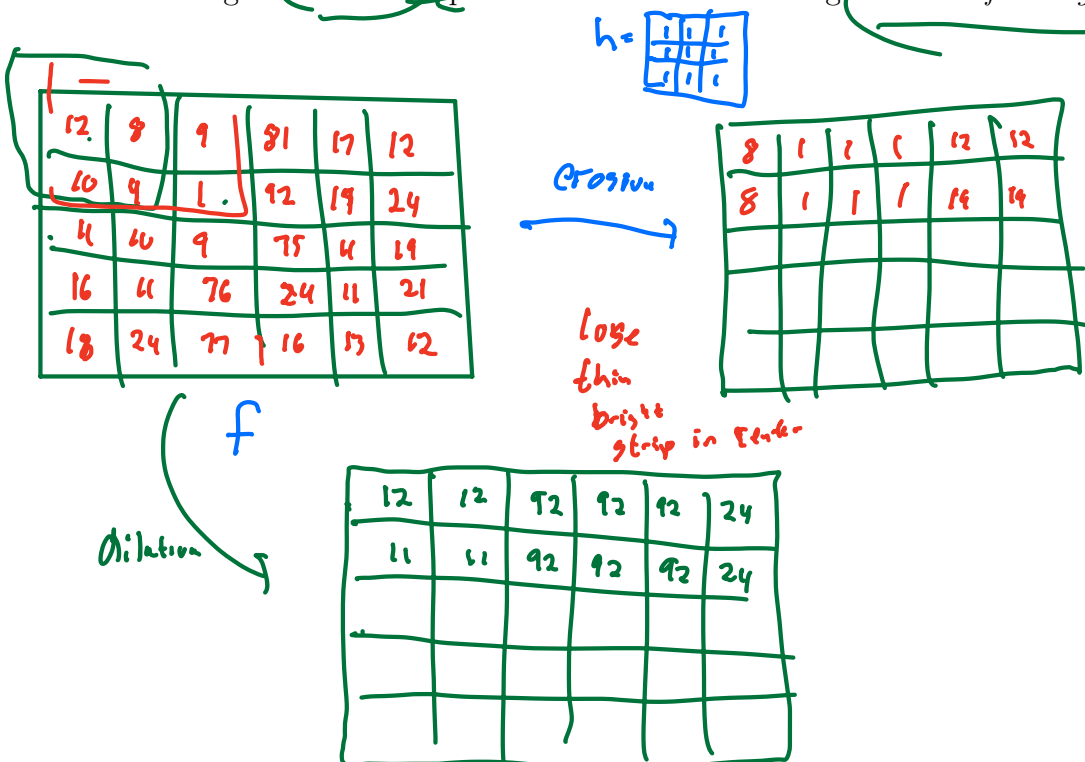
- Next, if $h(k, l)$ is binary, we can simplify:

$$g(i, j) = \sum_{(k, l) \text{ s.t. } h(k, l) \neq 0} f(i + k, j + l)$$

- We can replace the summation with a number of different operations. Specifically...
- We get fundamental techniques in what's called *mathematical morphology* as follows:
 - Using the minimum gets us "erosion",
 - Using the maximum gets us "dilation".

Most often these are applied to binary images

- Using the median in place of the summation we get *median filtering*



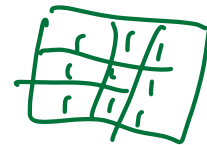
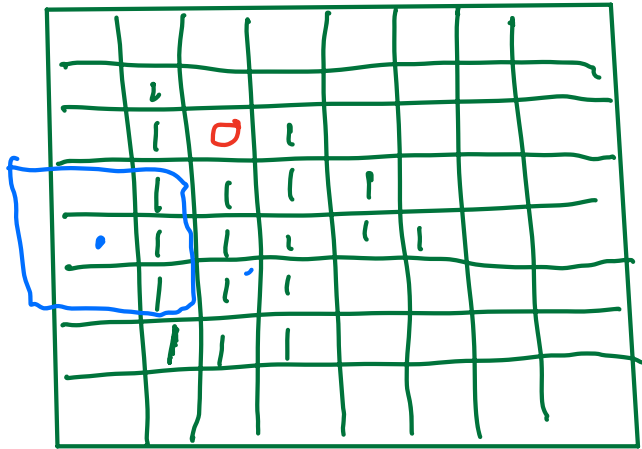
Mathematical Morphology

- Kernel == “structuring element”
- Common structuring elements:
 - Disk of radius r
 - Square of width w (usually odd)
 - Oriented rectangles
- Erosion “shrinks” binary image regions, dilation “expands” regions.
- Usually erosion and dilation are applied together:
 - An “opening” of an image is an erosion followed by a dilation, both with the same structuring element.
 - A “closing” of an image is a dilation followed by an erosion, both with the same structuring element.

Morphology: Important Intuitions

Assuming $k(0,0) = 1$, for kernel (structuring element) k :

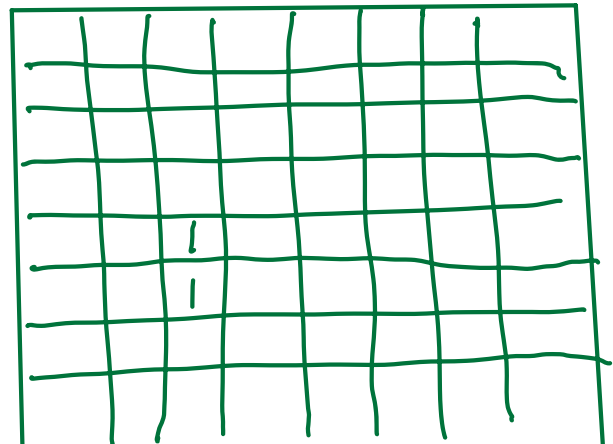
- Dilation: Place the center of the structuring element on an image location where there is a 1. Then, all image locations covered by ones in k are in the result of the dilation.



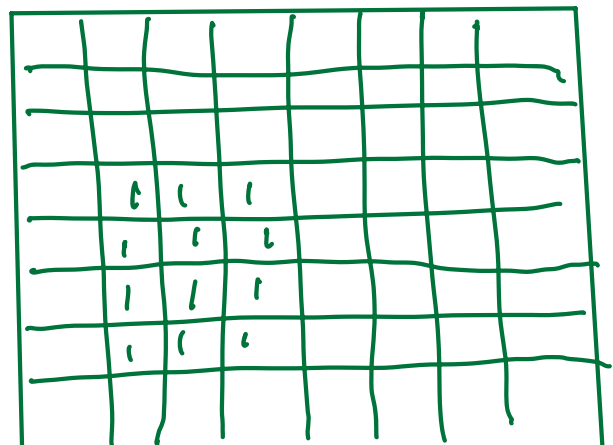
Kernel

Erosion

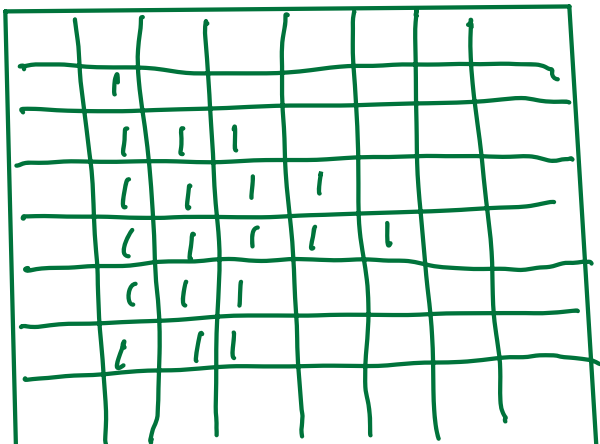
- dilate



dilate



erode



Closing

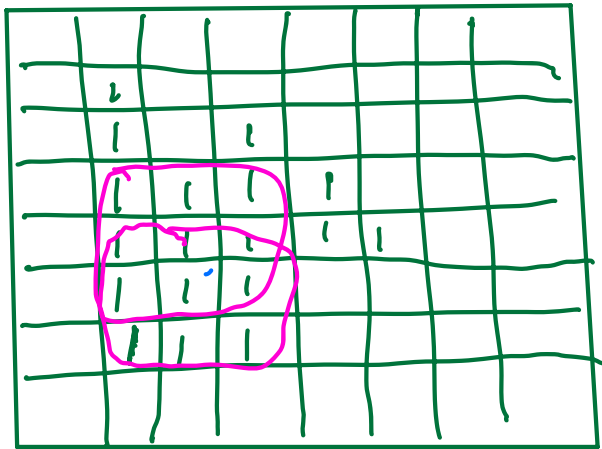
21

gap gone

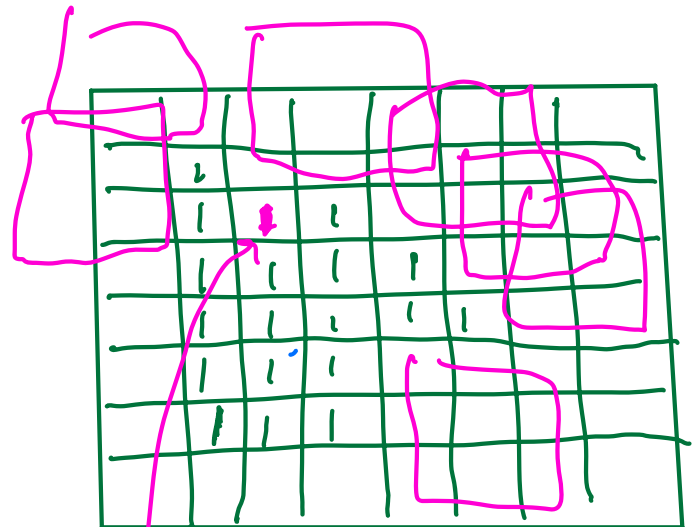
opening - small peripheral
1's are gone

- Erosion: Same intuition as dilation, but with the mirror image of k (unnecessary if k is symmetric) and reversing the roles of 0 and 1 in the image.

- Opening: Consider any location where k may be placed such that the locations where $k(h,l) = 1$ are entirely “inside” the image (covered by the pixels that are 1’s). These locations all remain 1 in the result image; everything else is 0.
- Closing: Consider any location where k may be placed such that k is entirely “outside” the image. All such locations will remain 0; everything else is a 1.
- “Open holes” and “close gaps”.



opening



closing

becomes 1

Application: Identifying Regions of Change

Problem:

Given an approximately stationary camera monitoring an approximately stationary scene, find the regions of change when something or someone is moving.

Solution

- Align images to remove small camera motions (not discussed here).
- Subtract images, pixel-by-pixel; compute the absolute value
- Apply a threshold
- Binary opening and binary closing to clean up

Here are the example images we will work with:

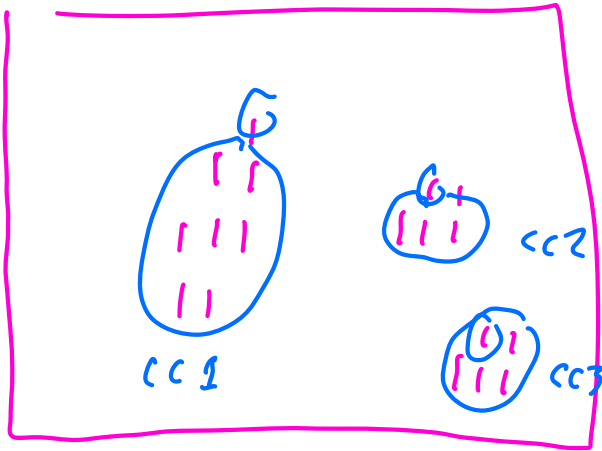


We Still Have An Image

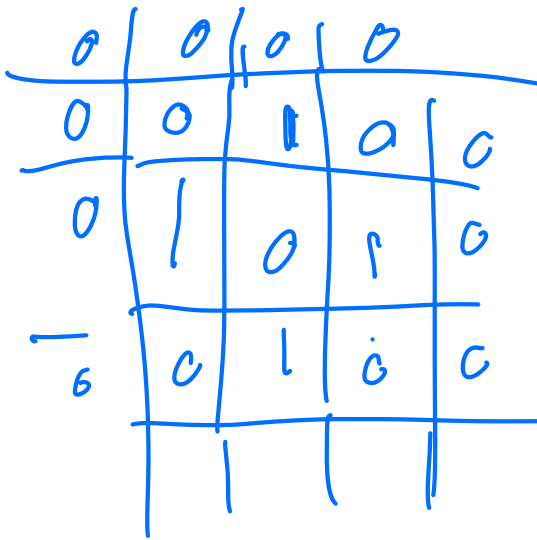
- After all this computation, we still only have an image.
 - To us, the image may look like it has semantic content, but it is really just an image.
- We need to now extract information, starting with the regions.
- Properties of these regions tell us useful information:
 - Where is something that has changed?
 - How wide spread is the change?
 - How big is the region of change?
- We do this through connected components region extraction and statistics to summarize the regions.

Connected Components Labeling

- Each set of pixels that are adjacent to each other forms a “connected component”
- Think of the image as a graph!
 - Pixels that are “1” are the vertices
 - Edges formed between neighboring pixels
 - Run a standard connected-components labeling algorithm!



- Question about connectivity: 4-connected or 8-connected?
 - We'll investigate in class.
- Standard breadth-first search or depth-first search may be used to extract connected components
- Faster, more specialized algorithms may be applied
- This is our first example of thinking of the image as a graph.



How many connected components are here?

8 connected

NW N, NE

W E

SW S SE

1 CC

4 connected

N

W E

S

4 CC

Usually
assume

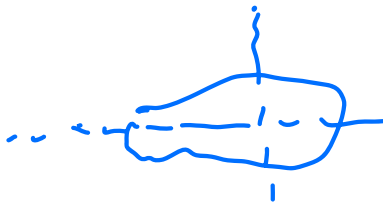
foreground 8 connected

background 4 connected

Region Analysis

list of pixel locs

- Once we have the sets of connected components, we can analyze each component to discover its area, center and second moments.
- Area and center are easy
- Second moments giving the major and minor axes of the region, are found from the eigenvectors of the 2x2 scatter matrix.
 - Sound familiar?



Properties of region

Finally, Non-Linear Smoothing: Median

- We described the median filter above as

$$g(i, j) = \underset{(k, l) \text{ s.t. } h(k, l) \neq 0}{\text{median}} f(i + k, j + l)$$

- Effective for removing “salt-and-pepper” noise in images, common in grainy old images:



- As with Gaussian smoothing, we get more smoothing and less noise sensitivity with larger regions.
- We will experimentally compare Gaussian smoothing and median filtering in class.
- There are even more sophisticated, “data sensitive” smoothing operations such as bilateral filtering.

Summary of Low-Level Image Processing

- Apply function defined over an image neighborhood at each pixel to compute new image.
- The neighborhood is defined by a kernel or structuring element.
- Smoothing and differentiation are achieved through linear operations — convolutions.
- The Gaussian is the most widely used smoothing operator.
- Image pyramids allow us to efficiently examine and match at different scales.
- The fundamental operations of mathematical morphology, erosion and dilation, are obtained by computing the min and max over a neighborhood of binary values.
- Using the median gives us the ability to remove salt and pepper noise.
- Image differencing, thresholding and morphology can be combined to form a motion detection and change analysis system.