H1N1 Outbreak Contingency Planning

Computational Vision
- For instructor illness, lectures will be canceled and due dates will be delayed.
- For student illness, extensions will be given. Contact the professor via email. Do NOT come to class if you are sick!

Outline of Lecture 3
- Review from Lecture 2, then...
  - Gaussian smoothing and differentiation as neighborhood operators
  - Brief introduction to Fourier Transforms
  - Binary image analysis and mathematical morphology
  - Application: a simple motion and change detection system.

We will finish our discussion of non-linear neighborhood operations during Lecture 4.

More Generally
- Wash your hands often, especially after shaking hands
- Avoid close contact with ill individuals
- Cover your mouth and nose with a tissue or your sleeve when coughing or sneezing.
- Do not touch your eyes, nose, mouth, especially after contact with shared keyboards, microscopes, etc!
Review From Lecture 2

- Color and color transformations
- Pixel transformations based on histograms
- Neighborhood operations as the basis for smoothing and for boundary location.
- Correlation and convolution:
  - Formula
  - Boundary artifacts
  - Separability
We will review this last point at the start of lecture.

Smoothing and Differentiation as Linear Operators

Back to Averaging: 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, at least as long as we are considering linear filtering, is the Gaussian function.

Gaussian Weighting — Continuous Form

- Continuous form in 1d
  \[ g(x; \sigma) = \exp\left(-0.5x^2/\sigma^2\right). \]
- Continuous form in 2d
  \[ g(x, y; \sigma) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(-\frac{y^2}{2\sigma^2}\right). \]
- This is 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 $3\sigma$ ($g(3\sigma; \sigma) \approx 0.011$, whereas $g(0; \sigma) = 1$).
- To form the discrete Gaussian, we sample and normalize so that the sum of the weights is 1.
- Discrete form in 1d (with $\sigma = 1.5$)

\[
\frac{1}{27} \begin{array}{ccccccc}
-3 & -2 & -1 & 0 & 1 & 2 & 3 \\
1 & 4 & 6 & 7 & 6 & 4 & 1
\end{array}
\]
- Other, less-exact approximations are sometimes used in practice. People also use floating-point and double-precision arithmetic as well.

Increasing Sigma Increases the Smoothing

- The $\sigma = 1.5$ kernel has width 7, whereas the $\sigma = 3$ kernel has width 15.
- The practical effect is increasingly smoothed and blurred images as $\sigma$ increases.
- We will look at example images in class.
- As we will see, this has important implications for feature extraction.
Discrete Partial Differentiation as Convolution

- Recall the limit definition of the derivative:
  \[ f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta(x)) - f(x)}{\Delta x}. \]

- 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:
  \[
  \begin{bmatrix}
    -1 & 0 & 1 \\
    -1/2 & 0 & 1/2
  \end{bmatrix}
  \]

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.

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.

Looking Ahead to Edge and Corner Detection

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.
Motivation

Understand the “frequencies” or the rate of variation in an image.
- Fundamental tool of signal processing and image processing.
- For us, the most important use is to analyze kernels.
- Works in the complex domain
- Mathematically, it is a change of basis (functions).

Very Brief Introduction to Fourier Transformations

Discrete, 1d Mathematical Form

Let $j \equiv \sqrt{-1}$, $k$ be the “frequency” variable, let $H$ be the Fourier transformation of $h$ — i.e. $H = \mathcal{F}(h)$ — and let $h(x)$ be the kernel. Then,

$$
H(k) = \frac{1}{N} \sum_{x=0}^{N} h(x) e^{-j \frac{2\pi kx}{N}}
$$

$$
= \frac{1}{N} \sum_{x=0}^{N} h(x) \left( \cos \left( \frac{2\pi kx}{N} \right) + j \sin \left( \frac{2\pi kx}{N} \right) \right)
$$

The angular frequency is actually $k \frac{2\pi}{N}$.
- We can also compute an inverse Fourier transformation.

Understanding This With Some Examples

Look at different values of $k$:
- What happens when $k = 0, k = 1$, etc?
- What happens if we shift the image?
Important Properties

- Convolution in the “spatial domain” becomes multiplication in the Fourier domain.
- Combining with the inverse Fourier Transformation we get
  \[ f \ast h = \mathcal{F}^{-1}(\mathcal{F}(f)\mathcal{F}(h)). \]

Examples

- The Fourier transform of the Box filter — square with constant values — is the sinc function.
- The Fourier transform of the Gaussian is another Gaussian, proportional to
  \[ \exp \left( -\frac{\omega^2}{2(1/\sigma^2)} \right). \]

Implication

- Multiplying by the Gaussian in the Fourier domain attenuates higher frequencies — noise and sharp changes.
- This explains mathematically the averaging and blurring effects we already know about.
- The limited “support” in both the spatial and frequency domains is why the Gaussian is a popular smoothing filter.
  - It is also the only isotropic, separable 2d filter.
- The box filter, mapping to the sinc function, does not have these same properties.

Motivation for Binary Image Analysis and Mathematical Morphology

Morphology:
- Operations on binary image
- First example of non-linear operators
- Concerned with “structure” of image regions
Looking again, in Lecture 4 we will discuss non-linear operators that are more closely analogous to convolution filtering.
Aside on Thresholding

- Given an image $f$ and a threshold $\theta$, compute a new image:
  \[
  b(x, y) = \begin{cases} 
  1 & f(x, y) \geq \theta \\
  0 & f(x, y) < \theta 
  \end{cases}
  \]
- Ideally, separate “foreground” (pixels of interest) from “background” (pixels that can be safely ignored).
- Ordinary photographs usually are ill-suited to thresholding.
- Special-purpose images, such as CT scans and document images, often are.
- Images that result from applying transformations to one or more original (input) images may be as well.
- Matlab functions to examine: graythresh and im2bw.

Set Theoretic Approach

- We will think of our binary image as a set of locations:
  \[
  B = \{(i, j) \mid b(i, j) = 1\}.
  \]
- We will apply a structuring element $K$ to an image, which may also be thought of as a set of locations.
  - $K$ plays a similar role in morphology that the convolution kernel plays in linear operations.
- Notationally, if $k \in K$ and $k = (x, y)$, then we write
  \[
  B_k = \{(i + x, j + y) \mid (i, j) \in B\}.
  \]
  In other words $B_k$ is a shift of the set $B$ by the vector $(x, y)$.

Dilation and Erosion

The two fundamental operations of morphology:

- Dilation:
  \[
  B \oplus K = \bigcup_{k \in K} B_k.
  \]
  In other words, translate $B$ by each element of $K$ and take the union.

- Erosion:
  \[
  B \ominus K = \bigcap_{k \in K} B_{-k}.
  \]
  In other words, translate $B$ by each $-k$ and take the intersection.

- We will look at examples in class.

Conceptual View

Assuming $(0, 0) \in K$.

- Dilation: Place the center of the structuring element on an image location (where there is a 1). All image locations covered by $K$ are in the result of the dilation.
- Erosion: Same intuition as dilation, but with the mirror image of $K$ and reversing the roles of 0 and 1 in the image.
Opening and Closing

- Opening: erosion followed by dilation with the same structuring element:
  \[ B \star K = B \ominus K \oplus K. \]
- Closing: dilation followed by erosion with the same structuring element:
  \[ B \circ K = B \oplus K \ominus K. \]
- Intuitions:
  - Opening: Consider any location where \( K \) may be placed such that \( K \) is entirely “inside” the image (the binary 1 region). All locations covered by \( K \) will “survive” the opening operation.
  - Closing: Consider any location where \( K \) may be placed such that \( K \) is entirely “outside” the image. All such locations will remain 0.
  - “Open holes” and “close gaps”.

Common Structuring Elements

- Disk of radius \( r \)
- Square of width \( w \) (usually odd)
- Oriented rectangles

Examples

We will play with some examples in class. Relevant Matlab functions include:
- \texttt{strel} — create a structuring element
- \texttt{imdilate}
- \texttt{imerode}
- \texttt{imclose}
- \texttt{imopen}

Connected Components Labeling

- Once we have binary image, we can gather pixels into regions and analyze the regions
- This is connected-component labeling
- Question about connectivity: 4-connected or 8-connected?
  - We’ll investigate in class.
- Think of the image as a graph!
  - Pixels that are “1” are the vertices
  - Edges formed to neighboring “1” pixels based on either 4-connectedness or 8-connectedness
- Standard breadth-first search or depth-first search may be used to extract connected components
- Faster, more specialized algorithms may be applied
- First example of thinking of the image as a graph.
Region Analysis

- 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 scatter matrix, as we will derive in class.

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
- Subtract images, pixel-by-pixel; compute the absolute value
- Apply a threshold
- Binary opening and binary closing to clean up
- Connected components and region analysis.

Code and Example

Matlab functions used:
- imread, rgb2gray, imabsdiff, imtool, im2bw, strel, imopen, imclose, bwlabel, label2rgb

Example images, after they have been aligned:

Summary

- Smoothing and differentiation are achieved through convolution
- The Gaussian is the most widely used smoothing operator.
- Fourier transforms help to analyze the frequencies of an image or a kernel.
- Mathematical morphology and region statistics are used to analyze binary images.
- Image differencing, thresholding and morphology can be combined to form a motion detection and change analysis system.
Project Idea

Motion and Change Detection

The algorithm we discussed for moving object and change detection is far simpler than many of the ideas that have been proposed in the research literature. Many algorithms proposed over the past ten years have focused on building up a model of the “natural” changes that can occur in the background. Research, implement and analyze one of these methods.