Correlation

CS 510 Lecture #12

Feb 20, 2002

How do we (directly) compare two images?

A

B

Are these images the same?
Are they similar?

First Approach:

Pixelwise Comparison

$$\sum_{x}^{x<width}\sum_{y}^{y<height}\left|A(x,y)-B(x,y)\right|$$

Is this a good measure? Why or why not?

Sources of Variation

info

- Change in scene content
- Out-of-plane rotation, scale change
- In-plane translation, rotation

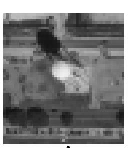
- Change in illumination

---

noie

- Change in mixed-pixels
- Change in gain, f-stop
- Electronic noise

Pixelwise Comparison (II)



A

B

$$8{,}140=\sum_{x}^{x<148}\sum_{y}^{y<161}\left|A(x,y)-B(x,y)\right|$$

Or, normalized by image area, approximately 5

A Better Approach:

Correlation

$$\frac{\sum_{x}\sum_{y}\left(\left(A[x,y]-\bar{A}\right)\left(B[x,y]-\bar{B}\right)\right)}{\sqrt{\sum_{x}\sum_{y}\left(A[x,y]-\bar{A}\right)^{2}}\sqrt{\sum_{x}\sum_{y}\left(B[x,y]-\bar{B}\right)^{2}}}$$

What is the underlying model?

Assumptions of Correlation

- The assumption of correlation is that the two signals vary linearly
  - Adding a constant to either signal does not change the correlation score
    - In Fourier terms, the DC component doesn't matter
  - Multiplying a zero-mean signal by a constant does not change the correlation score
    - Total range (power) doesn't matter
- This minimizes sensitivity to changes in (overall) illumination, f-stop or gain.

Special Cases

- Any two linear functions with positive slope have correlation 1.
  - Only the sign of the slope matters
- Any two linear functions with differently signed slopes have correlation -1.
  - This is called anti-correlation
  - If the goal is prediction, anti-correlation is as good as correlation
  - If the goal is matching, it may or may not be as good…
- Correlation is undefined for slope = 0 (σ=0)

Correlation (cont.)

- Correlation is <u>not</u> insensitive to:
  - Translation
  - Rotation
    - in-plane
    - out-of-plane
  - Scale
- In other words, correlation is not insensitive to any affine or perspective transformation. You are laying one image on top of another, and comparing pixel by pixel.

Computing Correlation

- Note that adding a constant to a signal does not change its correlation to any other signal, so
  - Let's substract –$\bar{A}$ from A(x,y)
  - Let's substract –$\bar{B}$ from B(x,y)
  - The mean of both signals is now zero
  - Their correlation reduces to:

$$\frac{A \cdot B}{\sqrt{\sum_x \sum_y \left(A[x,y]\right)^2}\sqrt{\sum_x \sum_y \left(B[x,y]\right)^2}}$$

Computing Correlation (II)

- For zero-mean signals, we can scale them without changing their correlation scores
  - Multiply A by the inverse of its length
  - Multiply B by the inverse of its length
  - Both signals are now unit length
  - Their correlation reduces to:

$$A \cdot B$$

Correlation Space

- Why zero-mean & unit-length your images?
  - Imagine correlating a new image with every image in a large database.
  - If the database images are zero-mean & unit-length
    - Preprocess A (zero-mean, unit-length)
    - Compute dot products
- New idea:
  - Think of your image as a point in an N dimensional space
    - N = width x height
    - An image database is a set of points
  - Then making your images zero-mean & unit-length projects them into an N-2 dimensional "correlation space" where the dot product equals correlation
    - This is a highly non-linear projection

Correlation Space (II)

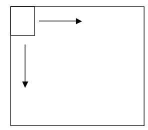
- In correlation space, Euclidean distance in inversely proportional to correlation

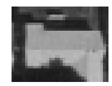
$$
\begin{aligned}
\sqrt{\sum_{x,y}\left(A[x,y]-B[x,y]\right)^2} &= \sqrt{\sum_{x,y}A[x,y]^2+\sum_{x,y}B[x,y]^2-2A[x,y]B[x,y]} \\
&= \sqrt{1+1-2\sum_{x,y}A[x,y]B[x,y]} \\
&= \sqrt{2-2A\cdot B} \\
&= \sqrt{2-2Corr(A,B)}
\end{aligned}
$$

- A nearest-neighbor classifier in correlation space maximized correlation

Cross-Correlation: Adding Translation

To find a small image in a large one, "slide" the small one across the large, computing the correlation at every possible position.





More on Cross-Correlation

- Increases complexity by a factor of **N** (# of pixels)
- Highly parallel
- Still sensitive to
  - Rotation
    - in-plane
    - out-of-plane
  - Scale
- Cross-correlating one template to M images increases complexity by M
  - The complexity constant drops a little...

Cross-Correlation and Rotation

- The brute force approach to making correlation insensitive to rotation is to generate N templates at N different angles.
  - Increase complexity by the factor N
  - Only accurate if N is fairly large
- If you could guess the orientation, you could apply only one template per pixel location
  - So how do you measure the orientation of a pixel?
  - What does it even mean?

Image as Surface

- View the image as a 3D surface
  - For every (x,y) pixel location, the intensity can be thought of as the z (height) value.
    - Color images are 5D surfaces - too hard to think about.
    - Color can also be thought of a 3 3D surfaces
  - Pretend the surface is continuous
- Every point on the image surface has a direction of maximum change (remember your multivariate calculus?), and a magnitude of change in that direction

Image Edges

- To compute the direction and magnitude of change, you can compute the magnitude of change in any two orthogonal directions and interpolate
  - Again, this assumes a continuous surface
- Compute the magnitude of change in the X & Y directions:
  - dx(x,y) = I(x,y) - I(x+1,y)
  - dy(x,y) = I(x,y) - I(x,y+1)
- Vector addition gives direction & orientation of slope.

Rotation-Free Cross Correlation

- For every pixel location:
  - compute edge orientation at pixel
  - rotate template until edge at center of template matches
    - using bilinear interpolation
  - correlate template with image
- Makes correlation insensitive to rotation
  - If edge direction estimates are accurate
- Assumes template is centered on an edge

Estimating Edge Orientation

- Problem: images are not really continuous surfaces
  - estimates of dx, dy based on grid sampling
  - note that if accurate,

$$I(x,y) - I(x+1,y) = I(x-1,y) - I(x,y)$$

  - estimating derivatives from two values is highly error prone.

Accurate Edge Estimation

- We want to compute a real-valued function
  - The partial derivatives dx & dy
- All we have are samples at equidistant points
- So model the function in terms of its Taylor series expansion:

$$f(x+h) = f(x) + \frac{h^1}{1!} f'(x) + \frac{h^2}{2!} f''(x) + \cdots$$

Accurate (II)

- So look at the equations for f(x+h) and f(x-h):

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \cdots$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \cdots$$

- Subtract (2) from (1) to get

$$f(x+h) - f(x-h) = 2hf'(x) + \cdots$$

- And solve for f'(x)

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

Accurate (III)

- So the best ±1 mask is

$$\frac{d}{dx} f(x,y) = \frac{f(x+1,y) - f(x-h,1)}{2}$$

- As an exercise, the best ±2 mask is

$$\frac{d}{dx} f(x,y) = \frac{-f(x+2,y) + 8f(x+1,y) - 8f(x-1,y) + f(x-2,y)}{12}$$

- The previous 3 slides are covered in T&V, appendix A.2

Discussion

- Correlation remains the base against which other image matching algorithms are compared
- It is sensitive to translation, rotation, and scale
  - cross-correlation compensates for translation
  - rotation-free cross-correlation compensates for rotation and translation
  - both are expensive
- PCA (eigenvectors) and Fourier Transforms both seem different, but are closely related.