Announcements

- HW3 to be assigned

Efficient Implementation

Both, the Box filter and the Gaussian filter are separable:
- First convolve each row of input image $I$ with a 1-D row filter $R$ to produce an intermediate image $J$
  $$J = R * I$$
- Then convolve each column of $J$ with a 1-D column filter $C$.
  $$O = C * J = C * (R * I) = (C * R) * I$$

The Fourier Transform and Convolution

- If $H$ and $G$ are images, and $F(.)$ represents Fourier transform, then
  $$F(H * G) = F(H)F(G)$$
  $$H^*G = F^{-1}(F(H)F(G))$$
- This is referred to as the Convolution Theorem.
- Fast Fourier Transform: complexity $O(n \log n)$ -> complexity of convolution is $O(n^2)$.
- Thus, one way of thinking about the properties of a convolution is by thinking of how it modifies the frequencies of the image to which it is applied.
- In particular, if we look at the power spectrum, then we see that convolving image $H$ by $G$ attenuates frequencies where $G$ has low power, and amplifies those which have high power.

Fourier Transform

Discrete Fourier Transform (DFT) of $I[x,y]$

$$F[u,v] \equiv \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x,y]e^{-\frac{2\pi}{N} xu}e^{-\frac{2\pi}{N} vu}$$

Inverse DFT

$$I[x,y] \equiv \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F[u,v]e^{\frac{2\pi}{N} xu}e^{\frac{2\pi}{N} vu}$$

$x,y$: spatial domain
$u,v$: frequency domain
$N$ by $N$ image
Implemented via the “Fast Fourier Transform” algorithm (FFT)
Edge Detection and Corner Detection

Edges
What is an edge?
A discontinuity in image intensity.

Physical causes of edges
1. Object boundaries
2. Surface normal discontinuities
3. Reflectance (albedo) discontinuities
4. Lighting discontinuities

Noisy Step Edge
- Derivative is high everywhere.
- Must smooth before taking gradient.

Edge is Where Change Occurs: 1-D
- Change is measured by derivative in 1D

Numerical Derivatives
Take Taylor series expansion of \( f(x) \) about \( x_0 \)
\[
f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2} f''(x_0)(x-x_0)^2 + \cdots
\]
Consider samples taken at increments of \( h \) and first two terms, we have
\[
f(x_0+h) = f(x_0) + f'(x_0)h + \frac{1}{2} f''(x_0)h^2
\]
\[
f(x_0-h) = f(x_0) - f'(x_0)h + \frac{1}{2} f''(x_0)h^2
\]
Subtracting and adding \( f(x_0+h) \) and \( f(x_0-h) \) respectively yields
\[
f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h}
\]
\[
f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2}
\]

Implementing 1-D Edge Detection
1. Filter out noise: convolve with Gaussian
   \( G * I \)
2. Take a derivative: convolve with \( D = [-1 0 1] \)
   - We can combine 1 and 2.
   \[
   D^r(G*I) = (D*G)*I
   \]
3. Find the peak of the magnitude of the convolved image: Two issues:
   - Should be a local maximum.
   - Should be sufficiently high.
2D Edge Detection: Canny

1. Filter out noise
   – Use a 2D Gaussian Filter. \( J = I * G \)
2. Take a derivative
   – Compute the magnitude of the gradient:
     \[
     \|\nabla J\| = \sqrt{J_x^2 + J_y^2}
     \]

Smoothing and Differentiation

- Need two derivatives, in x and y direction.
- Filter with Gaussian and then compute Gradient, OR
- Use a derivative of Gaussian filter
  - because differentiation is convolution, and convolution is associative

Directional Derivatives

\[ \frac{\partial G_\theta}{\partial x} \]
\[ \cos \theta \frac{\partial G_\theta}{\partial x} + \sin \theta \frac{\partial G_\theta}{\partial y} \]

Finding derivatives

Is this dI/dx or dI/dy?

There are three major issues:
1. The gradient magnitude at different scales is different; which scale should we choose?
2. The gradient magnitude is large along thick trail; how do we identify the significant points?
3. How do we link the relevant points up into curves?
There is ALWAYS a tradeoff between smoothing and good edge localization!

Image with Edge

Edge Location

Image + Noise

Derivatives detect edge and noise

Smoothed derivative removes noise, but blurs edge

The scale of the smoothing filter affects derivative estimates

1 pixel

3 pixels

7 pixels

We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: which point is the maximum, and where is the next point on the curve?

Non-maximum suppression

• Execute following over all pixels q in the image
• Using gradient direction at q, find two points p and r on adjacent rows (or columns).
  • If $|∇q| > |∇p|$ and $|∇q| > |∇r|$ then q remains a candidate edge point
  • p & r are found by interpolation

Non-maximum suppression

Predicting the next edge point

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either $r$ or $s$).
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s). Link together to create Image curve.

Hysteresis Thresholding

- Start tracking an edge chain at pixel location that is local maximum of gradient magnitude where gradient magnitude > $\tau_{\text{high}}$.
- Follow edge in direction orthogonal to gradient.
- Stop when gradient magnitude < $\tau_{\text{low}}$.

\[ \frac{|\nabla I|}{\text{Position along edge curve}} \]

Hysteresis thresholding
Why is Canny so Dominant

• Still widely used after 25 years.
  1. Theory is nice (but end result same,).
  2. Details good (magnitude of gradient, non-max supression).
  3. Hysteresis an important heuristic.
  4. Code was distributed.