What is an Edge?

**Edge:** A place in the image where the brightness jumps

- **Step edge**
- **Ramp edge**

How to detect edges?

1. **First derivative:**
   - Has extrema at edge locations
   - Apply thresholding to reduce sensitivity to noise
   - Edges are thick and/or broken

2. **Second derivative:**
   - Has 0-crossing at edge locations
   - Too many edges obtained

\[
\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}
\]
Gradient-based techniques

- Approximate gradient by discrete gradient operators:
  - Roberts
  - Prewitt
  - Sobel
  - Frei&Chen

Effects of noise

- Consider a single row or column of the image
  
  \[ \frac{df}{dx} \]

- Where is the edge?
Solution: smooth first

Where is the edge?  • Look for peaks in \( \frac{\partial}{\partial x} (h \ast f) \)

Linear Filter

• Low-pass filter:
  - Removes noise, which has high frequency characteristics
  - Smooths image
  - Blurs edges
  - Amount of smoothing depends upon filter window; which is related with scale

• High-pass filter
  - Accentuates noise; edges, details in images

What is the appropriate scale for an image?

• Scale \( \sim \sigma \)
• Scale \( \sim 1/\text{resolution} \)

How many pixels is enough to represent an image?

Sampling and the Nyquist rate

• Aliasing can arise when you sample a continuous signal or image
  - occurs when your sampling rate is not high enough to capture the amount of detail in your image
  - Can give you the wrong signal/image—an alias
  - formally, the image contains structure at different scales
  - called “frequencies” in the Fourier domain
  - the sampling rate must be high enough to capture the highest frequency in the image

• To avoid aliasing:
  - sampling rate \( \geq 2 \times \text{max frequency in the image} \)
  - said another way: two samples per cycle
  - This minimum sampling rate is called the Nyquist rate

The Fourier Transform

• Represent function on a new basis
  - Think of functions as vectors, with many components
  - We now apply a linear transformation to transform the basis
    - dot product with each basis element

• In the expression, \( u \) and \( v \) select the basis element, so a function of \( x \) and \( y \) becomes a function of \( u \) and \( v \)

• basis elements have the form \( e^{-2\pi i (ux+vy)} \)

To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of \( x,y \) for some fixed \( u,v \). We get a function that is constant when \( (ux+vy) \) is constant. The magnitude of the vector \( (u,v) \) gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.
Here $u$ and $v$ are larger than in the previous slide.

And larger still...

This is the magnitude transform of the zebra pic.

This is the phase transform of the zebra pic.

Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function’s values at a set of sample points. We’ll assume that these sample points are on a regular grid, and can place one at each integer for convenience.
Sampling in 2D does the same thing, only in 2D. We’ll assume that these sample points are on a regular grid, and can place one at each integer point for convenience.

A continuous model for a sampled function

- We want to be able to approximate integrals sensibly
- Leads to
  - the delta function
  - model on right

\[
\text{Sample}_\text{2D}(f(x,y)) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(x,y) \delta(x-i,y-j)
\]

The Fourier transform of a sampled signal

\[
F(\text{Sample}_\text{2D}(f(x,y))) = F\left(f(x,y) \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x-i,y-j)\right)
\]

\[
= F(f(x,y)) * F\left(\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x-i,y-j)\right)
\]

\[
= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} F(u-i,v-j)
\]

Smoothing as low-pass filtering

- The message of the FT is that high frequencies lead to trouble with sampling.
- Solution: suppress high frequencies before sampling
  - multiply the FT of the signal with something that suppresses high frequencies
  - or convolve with a low-pass filter
- A filter whose FT is a box is bad, because the filter kernel has infinite support.
- Common solution: use a Gaussian
  - multiplying FT by Gaussian is equivalent to convolving image with Gaussian.
Sampling without smoothing. Top row shows the images, sampled at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.

Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.

**Image Pyramid**
- Collection of decreasing resolution images arranged in the shape of a pyramid

**The Gaussian pyramid**
- Smooth with Gaussians, because
  - a Gaussian*Gaussian=another Gaussian
- Synthesis
  - smooth and sample
- Analysis
  - take the top image
- Gaussians are low pass filters, so repetition is redundant

**Prediction Residual Pyramid**
- Approximation pyramid
- Prediction residual pyramid
- Difference image
The Laplacian Pyramid

- **Synthesis**
  - preserve difference between upsampled Gaussian pyramid level and Gaussian pyramid level
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other levels
- **Analysis**
  - reconstruct Gaussian pyramid, take top layer

2D Wavelet Transform

Wavelet transform

Back to Edge Detection

There are three major issues:
1) The gradient magnitude at different scales is different; which 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?

Gradient operator responding to noise

Increasing noise -> (this is zero mean additive gaussian noise)
The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realisations of an image of gaussian noise.

Smoothing and Differentiation

- Issue: noise
  - smooth before differentiation
  - two convolutions to smooth, then differentiate?
  - actually, no - we can use a derivative of Gaussian filter
  - because differentiation is convolution, and convolution is associative

Derivative of Gaussian -DoG

1 pixel 3 pixels 7 pixels

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

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: at which point is the maximum, and where is the next one?
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 $s$).

Remaining issues

- Check that maximum value of gradient value is sufficiently large
- Drop-outs? Use hysteresis
  - Use a high threshold to start edge curves and a low threshold to continue them.
The Canny edge detector

- original image (Lena)

The Canny edge detector

- norm of the gradient

Canny Edge Detection

- Gradient-based edge detector
  1. LPF with a Gaussian
  2. Compute the gradient magnitude and orientation using the masks:

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & -1
\end{array}
\]

3. Nonmaxima suppression to thin the resulting ridges
4. Apply double thresholding to detect and link edges: Tracking can only begin at a point on a ridge higher than \(T_1\). Tracking then continues in both directions out from that point until the height of the ridge falls below \(T_2\).

Nonmaxima Suppression

- Canny does edge thinning by nonmaxima suppression:
  - Classify gradient angle into one of 4 sectors:
    - 0: -22.5 to 22.5, 180-22.5 to 180+22.5
    - 1: 22.5 to 67.5, 180+22.5 to 180+67.5
    - 2: 67.5 to 112.5, 180+67.5 to 180+112.5
    - 3: 112.5 to 157.5, 180+112.5 to 180+157.5
  - Compare center with the 2 neighbors, set to 0 if not greater than both

\[
\begin{array}{c}
\text{Sector 0} \\
\text{Sector 1} \\
\text{Sector 2} \\
\text{Sector 3}
\end{array}
\]
Double Thresholding

- Canny does edge linking by: double thresholding:
  - edge starts after passing $T_{high}$
  - edge ends when gradient falls below $T_{low}$

Canny Edge Detection

Noisy original  Canny  Sobel

Effect of $\sigma$ (Gaussian kernel size)

original  Canny with $\sigma = 1$  Canny with $\sigma = 2$

The choice of $\sigma$ depends on desired behavior:
- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Laplacian-based edge detection

Discrete approximations to the Laplacian:

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

Drawback: very sensitive to noise

Laplacian of Gaussian (LoG)

Smooth the image first with a Gaussian, then take Laplacian

\[
\text{LoG}(x, y) = \frac{1}{\sqrt{2\pi}\sigma^4} \left[ e^{-\frac{x^2+y^2}{2\sigma^2}} \right] \nabla^2 \nabla^2
\]

To find 0-crossings, threshold the image and find neighboring (+) and (-) values.
Laplacian of Gaussian (LoG)

- Effect of $\sigma$: $\sigma=1.0$ $\sigma=2.0$ $\sigma=3.0$

Line detectors

Burns Line Finder

1. Compute the gradient magnitude and direction at each pixel
2. Do two separate angle quantizations generating two images: (5 is -45 degrees to 45, etc.)
   \[
   \frac{\partial f}{\partial x} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & 2 \\ 1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} I(x-1, y) \\ I(x, y) \\ I(x+1, y) \end{pmatrix}
   \frac{\partial f}{\partial y} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \cdot \begin{pmatrix} I(x-1, y-1) \\ I(x-1, y+1) \\ I(x+1, y-1) \\ I(x+1, y+1) \end{pmatrix}
   \]
3. Find the connected component of each image and compute line length for each component:
   - Each pixel votes for its longer component
   - Each component receives a count of pixels that voted for it
   - The components that receive the majority support are selected

Finding Corners and Interest Points

1. Harris
2. HOG: Histogram of Oriented Gradients
3. SIFT: Scale Invariant Feature Transform

Corners contain more edges than lines.

- A corner is easier
Edge Detectors Tend to Fail at Corners

Finding Corners

Intuition:
- Right at corner, gradient is ill defined.
- Near corner, gradient has two different values.

Harris

Consider SSD for a small shift in $x, y$

$$e(x, y) = \sum_{(x,y)} (I(x, y) - I(x + \Delta x, y + \Delta y))^2$$

Taylor expansion of $I(x, y)$

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + [L_x \quad L_y] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Harris Matrix

We look at matrix:

Sum over a small region, the hypothetical corner

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means all gradients in neighborhood are:
- $(k,0)$ or $(0,c)$ or $(0,0)$ (or off-diagonals cancel).

What is region like if:
1. $\lambda_1 = 0$?
   - EDGE
2. $\lambda_2 = 0$?
   - EDGE
3. $\lambda_1 = 0$ and $\lambda_2 = 0$?
   - UNIFORM
4. $\lambda_1 > 0$ and $\lambda_2 > 0$?
   - CORNER
General Case:
From Linear Algebra, it follows that because C is symmetric:

\[ C = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \]

With R a rotation matrix.
So every case is like one on last slide.

Harris
• Harris matrix is also called autocorrelation or 2nd moment matrix
  • 2 large eigenvalues => interest point
  • 1 large eigenvalue => contour
  • 0 large eigenvalues => uniform region
• Interest point detection function
  – Harris \[ f_H = \det(M) - k \text{tr}(M)^2 \]
  – Shi-Tomasi \[ f_S = \min(\lambda_1, \lambda_2) \]

See also [Triggs ECCV 04] – stable pos, ori, scale

Harris + Correlation
NCC matching plus RANSAC for fundamental matrix

Interest points extracted with Harris (~ 500 points)

Scale
• Scale makes a big difference

A bar in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose

99 inliers
89 outliers

[Zhang, Deriche, Faugeras, Luong 1995, Beardsley, Torr, Zisserman 1996 etc.]
The Gaussian pyramid

- Smooth with Gaussians, because
  - a Gaussian * Gaussian = another Gaussian
- Synthesis
  - smooth and sample
- Analysis
  - take the top image
- Gaussians are low pass filters, so repetition is redundant

Gaussian pyramid construction

Repeat
- Filter
- Subsample
Until minimum resolution reached
- can specify desired number of levels (e.g., 3-level pyramid)
The whole pyramid is only 4/3 the size of the original image!

Subsampling with Gaussian pre-filtering

G 1/2
Filter the image, then subsample

G 1/4

G 1/8

Subsampling without pre-filtering

1/2
1/4 (2x zoom)
1/8 (4x zoom)

Subsampling with Gaussian pre-filtering

Gaussian
1/2
G 1/4
G 1/8
Filter the image, then subsample

The Laplacian Pyramid

- Synthesis
  - preserve difference between upsampled Gaussian pyramid level and Gaussian pyramid level
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other levels
- Analysis
  - reconstruct Gaussian pyramid, take top layer
Detector
1. Find Scale-Space Extrema
2. Keypoint Localization & Filtering
   - Improve keypoints and throw out bad ones

Descriptor
3. Orientation Assignment
   - Remove effects of rotation and scale
4. Create descriptor
   - Using histograms of orientations

---

Scale Space
- Need to find ‘characteristic scale’ for feature
- Scale-Space: Continuous function of scale $\sigma$
  - Only reasonable kernel is Gaussian:
    \[
    L(x, y, \sigma) = G(x, y, \sigma)*I(x, y)
    \]

[Koenderink 1984, Lindeberg 1994]

Scale Selection
- Experimentally, Maxima of Laplacian-of-Gaussian gives best notion of scale:
  - Thus use Laplacian-of-Gaussian (LoG) operator: $\sigma^2 \nabla^2 G$

Mikolajczyk 2002

Approximate LoG
- LoG is expensive, so let’s approximate it
- Using the heat-diffusion equation:
  \[
  \sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} = \frac{G(k\sigma) - G(\sigma)}{k\sigma - \sigma}
  \]
- Define Difference-of-Gaussians (DoG):
  \[
  (k-1)\sigma^2 \nabla^2 G = G(k\sigma) - G(\sigma)
  \]
  \[
  D(\sigma) = (G(k\sigma) - G(\sigma))*I
  \]

DoG efficiency
- The smoothed images need to be computed in any case for feature description.
- We need only to subtract two images.
DoB filter (‘Difference of Boxes’)
- Even faster approximation is using box filters (by integral image)

![Image](66x606 to 279x674)

甚至更快的近似方法是使用盒子滤波器（通过积分图像）

Fig. 1. Left to right: the (discretized and cropped) Gaussian second order partial derivatives in $x$-direction and $y$-direction, and our approximations thereof using box filters. The grey regions are equal to zero.

Bay, ECCV 2006

Scale-Space Construction
- First construct scale-space:

![Diagram](nanx549436 to nanx549267)

Difference-of-Gaussians
- Now take differences:

![Diagram](nanx549436 to nanx549436)

Scale-Space Extrema
- Choose all extrema within 3x3x3 neighborhood.
- Low cost – only several usually checked

![Diagram](nanx549436 to nanx549436)

SIFT Overview
- Detector
  1. Find Scale-Space Extrema
  2. Keypoint Localization & Filtering
     - Improve keypoints and throw out bad ones
- Descriptor
  3. Orientation Assignment
     - Remove effects of rotation and scale
  4. Create descriptor
     - Using histograms of orientations

![Box](555x25 to 72x715)

Keypoint Localization & Filtering
- Now we have much fewer points than pixels.
- However, still lots of points (~1000s)...
  - With only pixel-accuracy at best
    - At higher scales, this corresponds to several pixels in base image
    - And this includes many bad points

Brown & Lowe 2002
Keypoint Localization

- The problem:
  - Sampling
  - Detected Extrema
  - True Extrema

- The Solution:
  - Take Taylor series expansion:
    \[ D(\hat{x}) = D + \frac{\partial D}{\partial x} \hat{x} + \frac{1}{2} \hat{x}^T \frac{\partial^2 D}{\partial x^2} \hat{x} \]
  - Minimize to get true location of extrema:
    \[ \hat{x} = -\frac{\partial^2 D}{\partial x^2} \frac{\partial D}{\partial x} \]

Brown & Lowe 2002

Finding Keypoints – Scale, Location

- Sub-pixel Localization
  - Fit Trivariate quadratic to find sub-pixel extrema

- Eliminating edges
  - Similar to Harris corner detector

\[ H = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yy} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \]

Keypoint Filtering - Low Contrast

- Reject points with bad contrast

\[ D(\hat{x}) \] is smaller than 0.03 (image values in [0,1])

Keypoint Filtering - Edges

- Reject points with strong edge response in one direction only
- Like Harris - using Trace and Determinant of Hessian

(b) 832 DOG extrema
Keypoint Filtering - Edges

- check if ratio of principal curvatures is below some threshold, \( r \), check:

\[
\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}
\]

- \( r=10 \)
- Only 20 floating points operations to test each keypoint

Keypoint Filtering

SIFT Overview

Detector
1. Find Scale-Space Extrema
2. Keypoint Localization & Filtering
   – Improve keypoints and throw out bad ones

Descriptor
3. Orientation Assignment
   – Remove effects of rotation and scale
4. Create descriptor
   – Using histograms of orientations

Orientation Assignment

- Now we have set of good points
- Choose a region around each point
  – Remove effects of scale and rotation

Finding Keypoints – Orientation

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)

Orientation Assignment

- Create gradient histogram (36 bins)
  – Weighted by magnitude and Gaussian window ($\sigma$ is 1.5 times that of the scale of a keypoint)
Orientation Assignment

- Any peak within 80% of the highest peak is used to create a keypoint with that orientation
- ~15% assigned multiple orientations, but contribute significantly to the stability
- Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy

SIFT Overview

**Detector**
1. Find Scale-Space Extrema
2. Keypoint Localization & Filtering
   - Improve keypoints and throw out bad ones

**Descriptor**
3. Orientation Assignment
   - Remove effects of rotation and scale
4. Create descriptor
   - Using histograms of orientations

Ideal Descriptors

- Robust to:
  - Affine transformation
  - Lighting
  - Noise
- Distinctive
- Fast to match
  - Not too large
  - Usually L1 or L2 matching

SIFT Descriptor

- Each point so far has x, y, σ, m, θ
- Now we need a descriptor for the region
  - Could sample intensities around point, but...
    - Sensitive to lighting changes
    - Sensitive to slight errors in x, y, θ
- Look to biological vision
  - Neurons respond to gradients at certain frequency and orientation
  - But location of gradient can shift slightly!

SIFT Descriptor

- 4x4 Gradient window
- Histogram of 4x4 samples per window in 8 directions
- Gaussian weighting around center ($σ$ is 0.5 times that of the scale of a keypoint)
- $4 \times 4 \times 8 = 128$ dimensional feature vector

Histogram of Oriented Gradients (HOG)

- Also used as a local interest operator
- Based on gradients
- Orientation histograms
- Tested with
  - RGB
  - LAB
  - Grayscale

- Gamma Normalization and Compression
  - Square root
  - Log

- Histogram of gradient orientations
  - Orientation
  - Position
  - Weighted by magnitude

- R-HOG
- C-HOG

- L1 - norm: \( \| \| \| x \| \|_1 \) + \( \| \| \| x \| \|_2 \) + \( \| \| \| x \| \|_\infty \)
- L2 - L1-norm: plus clipping at 2 and normalizing
Homework II

- Find edges in an image using Sobel gradient method. Then compare with Canny edge detector result using a library routine for Canny.
- Implement the Harris corner detector and apply it to an image. Then, compare with a library routine for an interest operator (SIFT, SURF, or KLT)