Image Features

Slides revised from S. Seitz, R. Szeliski, S. Lazebnik, etc.

Edges and Scale

Origin of Edges

- depth discontinuity
- surface color discontinuity
- illumination discontinuity

Edges are caused by a variety of factors

Detecting edges

What’s an edge?
- intensity discontinuity (= rapid change)

How can we find large changes in intensity?
- gradient operator seems like the right solution

Effects of noise

Consider a single row or column of the image
- Plotting intensity as a function of position gives a signal

Where is the edge?

Solution: smooth first

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \ast f)$
Associative property of convolution

\[ \frac{\partial}{\partial x} ((h * f)) = (\frac{\partial}{\partial x} h) * f \]

This saves us one operation:

Laplacian of Gaussian

Consider \[ \frac{\partial^2}{\partial x^2} (h * f) \]

Where is the edge? Zero-crossings of bottom graph

2D edge detection filters

Laplacian of Gaussian

Gaussian derivative of Gaussian

\[ h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

\[ \frac{\partial}{\partial x} h(x, y) \]

\[ \nabla^2 = \text{Laplacian operator:} \]

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]

The Sobel operator

Common approximation of derivative of Gaussian

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

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term is needed to get the right gradient value, however

The effect of scale on edge detection

Scale space (Witkin 83)

Some times we want many resolutions

Gaussian Pyramids have all sorts of applications in computer vision

Known as a Gaussian Pyramid [Burt and Adelson, 1983]

- In computer graphics, a mip map [Williams, 1983]
- A precursor to wavelet transform
Gaussian pyramid construction

Repeat
- Filter (called “prefiltering”)
- Subsample (faster approach: filter only ¼ of pixels)
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

Subsampling without pre-filtering

Corners

Why extract features?
- Motivation: panorama stitching
  - We have two images – how do we combine them?
Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features

Characteristics of good features

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Applications

Feature points are used for:

- Motion tracking
- Image alignment
- 3D reconstruction
- Object recognition
- Indexing and database retrieval
- Robot navigation

Finding Corners

- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive


The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity

Source: A. Efros
Harris Detector: Mathematics

Change in appearance for the shift \([u,v]\):

\[
E(u,v) = \sum_{x,y} w(x,y) \left[ I(x+u, y+v) - I(x,y) \right]^2
\]

Second-order Taylor expansion of \(E(u,v)\) about \((0,0)\)
(bilinear approximation for small shifts):

\[
E(u,v) \approx E(0,0) + \left[ u \begin{bmatrix} E_{xx}(0,0) \\ E_{xy}(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} E_{xx}(0,0) & E_{xy}(0,0) \\ E_{xy}(0,0) & E_{yy}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right]
\]

The bilinear approximation simplifies to

\[
E(u,v) \approx [u \ v] \begin{bmatrix} M_{11} & M_{12} \\ M_{12} & M_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\]

where \(M\) is a 2×2 matrix computed from image derivatives:

\[
M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}
\]

\[
M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} [I_x I_y]^T = \sum \nabla I(\nabla I)^T
\]

Interpreting the second moment matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

\[
M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}
\]

If either \(\lambda\) is close to 0, then this is not a corner, so look for locations where both are large.

General Case

Since \(M\) is symmetric, we have

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

We can visualize \(M\) as an ellipse with axis lengths determined by the eigenvalues and orientation determined by \(R\)

Ellipse equation:

\[
[u \ v] \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \text{const}
\]
Visualization of second moment matrices

Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

- **Corner**: $\lambda_1$ and $\lambda_2$ are large, $\lambda_1 \approx \lambda_2$.
- **Edge**: $\lambda_1 > \lambda_2$.
- **Flat** region: $\lambda_1$ and $\lambda_2$ are small, $\lambda_1 \approx \lambda_2$.

Corner response function

$$R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

where $\alpha$ is constant (0.04 to 0.06).

Harris detector: Steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix $M$ in a Gaussian window around each pixel
3. Compute corner response function $R$
4. Threshold $R$
5. Find local maxima of response function (nonmaximum suppression)
Harris Detector: Steps

Compute corner response $R$

Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$

Harris Detector: Steps

Take only the points of local maxima of $R$

Invariance

- We want features to be detected despite geometric or photometric changes in the image: if we have two transformed versions of the same image, features should be detected in corresponding locations

Models of Image Change

Geometric

- Rotation

- Scale

- Affine
  valid for: orthographic camera, locally planar object

Photometric

- Affine intensity change ($I \rightarrow aI + b$)
Harris Detector: Invariance Properties

Rotation

Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response $R$ is invariant to image rotation

Affine intensity change

- Only derivatives are used => invariance to intensity shift $I \to I + b$
- Intensity scale: $I \to aI$

Partially invariant to affine intensity change

Harris Detector: Invariance Properties

Scaling

Corner

All points will be classified as edges

Not invariant to scaling

Blobs

Scale-invariant feature detection

- Goal: independently detect corresponding regions in scaled versions of the same image
- Need scale selection mechanism for finding characteristic region size that is covariant with the image transformation

Scale-invariant features: Blobs
Recall: Edge detection

\[ f \]

\[ \frac{df}{dx} \]

\[ f \frac{df}{dx} \]

Edge = maximum of derivative

Source: S. Seitz

Edge detection, Take 2

\[ f \]

\[ \frac{d^2f}{dx^2} \]

\[ f \frac{d^2f}{dx^2} \]

Edge = zero crossing of second derivative

Source: S. Seitz

From edges to blobs

- Edge = ripple
- Blob = superposition of two ripples

Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Scale selection

- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases:

Why does this happen?

Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as \( \sigma \) increases

\[
\frac{1}{\sigma \sqrt{2\pi}}
\]

Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as \( \sigma \) increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by \( \sigma \)
- Laplacian is the second Gaussian derivative, so it must be multiplied by \( \sigma^2 \)
**Effect of scale normalization**

Unnormalized Laplacian response

Scale-normalized Laplacian response

**Blob detection in 2D**

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]

**Blob detection in 2D**

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

Scale-normalized: \[ \nabla^2_{\text{norm}} g = \sigma^2 \left( \frac{\partial^2 g}{\partial x^2} \frac{\sigma^2}{\partial^2 y^2} \right) \]

**Scale selection**

- At what scale does the Laplacian achieve a maximum response for a binary circle of radius \( r \)?

**Characteristic scale**

- We define the characteristic scale as the scale that produces peak of Laplacian response

Scale-space blob detector
1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space

Approximating the Laplacian with a difference of Gaussians:

\[ L = \sigma^2 \left( G_{\sigma}(x,y,\sigma) + G_{\sigma}(x,y,\sigma) \right) \]

(Laplacian)

\[ D\sigma G = G(x,y,\sigma) - G(x,y,\sigma) \]

(Difference of Gaussians)

Efficient implementation

From scale invariance to affine invariance

Affine adaptation

Recall:  
\[
M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R^T
\]

We can visualize \( M \) as an ellipse with axis lengths determined by the eigenvalues and orientation determined by \( R \).

**Ellipse equation:**

\[
[a \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}
\]

Affine adaptation example

Scale-invariant regions (blobs)

Affine adaptation example

Affine-adapted blobs

Affine normalization

- The second moment ellipse can be viewed as the "characteristic shape" of a region
- We can normalize the region by transforming the ellipse into a unit circle

Orientation ambiguity

- There is no unique transformation from an ellipse to a unit circle
- We can rotate or flip a unit circle, and it still stays a unit circle
Orientation ambiguity

- There is no unique transformation from an ellipse to a unit circle
  - We can rotate or flip a unit circle, and it still stays a unit circle
  - So, to assign a unique orientation to keypoints:
    - Create histogram of local gradient directions in the patch
    - Assign canonical orientation at peak of smoothed histogram

Affine adaptation

- Problem: the second moment “window” determined by weights $w(x,y)$ must match the characteristic shape of the region
- Solution: iterative approach
  - Use a circular window to compute second moment matrix
  - Perform affine adaptation to find an ellipse-shaped window
  - Recompute second moment matrix using new window and iterate

Iterative affine adaptation

![Initial](image1)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Initial" /></td>
<td><img src="image1" alt="Initial" /></td>
<td><img src="image1" alt="Initial" /></td>
<td><img src="image1" alt="Initial" /></td>
</tr>
</tbody>
</table>


http://www.robots.ox.ac.uk/~vgg/research/affine/

Summary: Feature extraction

- Extract affine regions
- Normalize regions
- Eliminate rotational ambiguity
- Compute appearance descriptors

Invariance vs. covariance

**Invariance:**
- $\text{features}(\text{transform(image)}) = \text{features(image)}$

**Covariance:**
- $\text{features}(\text{transform(image)}) = \text{transform(features(image)))}$

Covariant detection $\Rightarrow$ invariant description