Feature extraction: Corners and blobs
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
Why extract features?

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

Step 1: extract features
Step 2: match features
Step 3: align images
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

Corner Detection: 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.

"flat" region: no change in all directions
"edge": no change along the edge direction
"corner": significant change in all directions

Source: A. Efros
Corner Detection: Mathematics

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

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

Window function

Shifted intensity

Intensity

Window function \(w(x, y) = \)

1 in window, 0 outside

Gaussian

Source: R. Szeliski
Corner Detection: Mathematics

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

\[
E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2
\]
Corner Detection: Mathematics

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

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

We want to find out how this function behaves for small shifts

Second-order Taylor expansion of \(E(u, v)\) about \((0, 0)\) (local quadratic approximation):

\[
E(u, v) \approx E(0, 0) + [u \ v]\begin{bmatrix} E_u(0, 0) \\ E_v(0, 0) \end{bmatrix} + \frac{1}{2} [u \ v]\begin{bmatrix} E_{uu}(0, 0) & E_{uv}(0, 0) \\ E_{uv}(0, 0) & E_{vv}(0, 0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}
\]
Corner Detection: Mathematics

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

Second-order Taylor expansion of \( E(u, v) \) about \((0,0)\):

\[
E(u, v) \approx E(0,0) + [u \quad v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} [u \quad v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} [u \quad v]
\]

\[
E_u(u, v) = \sum_{x, y} 2w(x, y)[I(x + u, y + v) - I(x, y)]I_x(x + u, y + v)
\]

\[
E_{uu}(u, v) = \sum_{x, y} 2w(x, y)I_x(x + u, y + v)I_x(x + u, y + v)
\]

\[
+ \sum_{x, y} 2w(x, y)[I(x + u, y + v) - I(x, y)]I_{xx}(x + u, y + v)
\]

\[
E_{uv}(u, v) = \sum_{x, y} 2w(x, y)I_y(x + u, y + v)I_x(x + u, y + v)
\]

\[
+ \sum_{x, y} 2w(x, y)[I(x + u, y + v) - I(x, y)]I_{xy}(x + u, y + v)
\]
Corner Detection: Mathematics

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

Second-order Taylor expansion of \( E(u,v) \) about (0,0):

\[
E(u, v) \approx E(0,0) + [u \quad v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} [u \quad v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} [u \quad v]
\]

\[
E(0,0) = 0 \\
E_u(0,0) = 0 \\
E_v(0,0) = 0 \\
E_{uu}(0,0) = \sum_{x,y} 2w(x, y)I_x(x, y)I_x(x, y) \\
E_{vv}(0,0) = \sum_{x,y} 2w(x, y)I_y(x, y)I_y(x, y) \\
E_{uv}(0,0) = \sum_{x,y} 2w(x, y)I_x(x, y)I_y(x, y)
\]
Corner Detection: Mathematics

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

Second-order Taylor expansion of \( E(u,v) \) about (0,0):

\[ E(u, v) \approx \begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} \sum_{x, y} w(x, y) I_x^2(x, y) & \sum_{x, y} w(x, y) I_x(x, y) I_y(x, y) \\ \sum_{x, y} w(x, y) I_x(x, y) I_y(x, y) & \sum_{x, y} w(x, y) I_y^2(x, y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ E(0,0) = 0 \]
\[ E_u (0,0) = 0 \]
\[ E_v (0,0) = 0 \]
\[ E_{uu} (0,0) = \sum_{x, y} 2w(x, y) I_x(x, y) I_x(x, y) \]
\[ E_{vv} (0,0) = \sum_{x, y} 2w(x, y) I_y(x, y) I_y(x, y) \]
\[ E_{uv} (0,0) = \sum_{x, y} 2w(x, y) I_x(x, y) I_y(x, y) \]
Corner Detection: Mathematics

The quadratic approximation simplifies to

\[ E(u, v) \approx [u \ v] \ M \ \begin{bmatrix} u \\ v \end{bmatrix} \]

where \( M \) is a second moment matrix computed from image derivatives:

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

\[ M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T \]
Interpreting the second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form. Let’s try to understand its shape.

$$E(u,v) \approx [u \ v] \ M \ [u \ v]$$

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
Interpreting the second moment matrix

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

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

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

Consider a horizontal “slice” of $E(u, v)$:

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.
Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$:  
$$[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.

Diagonalization of $M$:
$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by $R$. 

direction of the fastest change

$$(\lambda_{\text{max}})^{-1/2}$$

$$(\lambda_{\text{min}})^{-1/2}$$

direction of the slowest change
Visualization of second moment matrices
Visualization of second moment matrices
Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

- $\lambda_2$ and $\lambda_2 \gg \lambda_1$: “Corner”
  - $\lambda_1$ and $\lambda_2$ are large,
  - $\lambda_1 \sim \lambda_2$;
  - $E$ increases in all directions

- $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions: “Flat” region

- $\lambda_1 \gg \lambda_2$: “Edge”
Corner response function

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

\( \alpha \): 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)

C. Harris and M. Stephens. "A Combined Corner and Edge Detector."
Harris Detector: Steps
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 $R$
Harris Detector: Steps
Invariance and covariance

- We want features to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance**: image is transformed and features do not change
  - **Covariance**: if we have two transformed versions of the same image, features should be detected in corresponding locations
Models of Image Change

Photometric
  - Affine intensity change \( (I \rightarrow aI + b) \)

Geometric
  - Rotation
  - Scale
  - Affine

valid for: orthographic camera, locally planar object
Affine intensity change

✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

✓ Intensity scale: $I \rightarrow a I$

Partially invariant to affine intensity change
Image rotation

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

Corner response $R$ is invariant w.r.t. rotation and corner location is covariant
Scaling

Corner

All points will be classified as edges

Not invariant to scaling
Achieving scale covariance

- 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
Blob detection with scale selection
Recall: Edge detection

\( f \)

\( \frac{d}{dx} g \)

\( f \ast \frac{d}{dx} g \)

Edge

Derivative of Gaussian

Edge = maximum of derivative

Source: S. Seitz
Edge detection, Take 2

\[ f \]

Second derivative of Gaussian (Laplacian)

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

Edge = zero crossing of second derivative

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

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

Original signal

Unnormalized Laplacian response

Scale-normalized Laplacian response

maximum
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: \( \nabla^2_{\text{norm}} g = \sigma^2 \left( \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right) \)
Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius $r$?
Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius \( r \)?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle.
- The Laplacian is given by (up to scale):
  \[
  (x^2 + y^2 - 2\sigma^2) e^{-(x^2+y^2)/2\sigma^2}
  \]
- Therefore, the maximum response occurs at \( \sigma = r / \sqrt{2} \).
Characteristic scale

- We define the characteristic scale of a blob as the scale that produces peak of Laplacian response in the blob center.

Scale-space blob detector

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

sigma = 11.9912
Scale-space blob detector: Example
Efficient implementation

Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

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

(Difference of Gaussians)
Efficient implementation

Invariance and covariance properties

• Laplacian (blob) response is *invariant* w.r.t. rotation and scaling
• Blob location is *covariant* w.r.t. rotation and scaling
• What about intensity change?
Achieving affine covariance

Consider the second moment matrix of the window containing the blob:

\[
M = \sum_{x,y} w(x,y) \begin{bmatrix}
I_x^2 & I_x I_y \\
I_x I_y & I_y^2
\end{bmatrix} = R^{-1} \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} R
\]

Recall:

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

This ellipse visualizes the “characteristic shape” of the window
Affine adaptation example

Scale-invariant regions (blobs)
Affine adaptation example

Affine-adapted blobs
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


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

• Affinely transformed versions of the same neighborhood will give rise to ellipses that are related by the same transformation.

• What to do if we want to compare these image regions?

• *Affine normalization*: transform these regions into same-size circles.
Affine normalization

- Problem: 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
Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram
From covariant regions to invariant features

1. Extract affine regions
2. Normalize regions
3. Eliminate rotational ambiguity
4. 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 => invariant description