Keypoint extraction: Corners

9300 Harris Corners Pkwy, Charlotte, NC
Why extract keypoints?

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

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

Step 1: extract keypoints
Step 2: match keypoint features
Why extract keypoints?

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

Step 1: extract keypoints
Step 2: match keypoint features
Step 3: align images
Characteristics of good keypoints

• Compactness and efficiency
  • Many fewer keypoints than image pixels

• Saliency
  • Each keypoint is distinctive

• Locality
  • A keypoint occupies a relatively small area of the image; robust to clutter and occlusion

• Repeatability
  • The same keypoint can be found in several images despite geometric and photometric transformations
Applications

Keypoints are used for:

• Image alignment
• 3D reconstruction
• Motion tracking
• Robot navigation
• Indexing and database retrieval
• Object recognition
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
Corner Detection: Mathematics

Change in appearance of window $W$ for the shift $[u,v]$:

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

Change in appearance of window $W$ for the shift $[u, v]$:  

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

Change in appearance of window $W$ for the shift $[u, v]$:

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

We want to find out how this function behaves for small shifts $E(u, v)$.
Corner Detection: Mathematics

• First-order Taylor approximation for small motions \([u, v]\):

\[
I(x + u, y + v) \approx I(x, y) + I_x u + I_y v
\]

• Let’s plug this into \(E(u,v)\):

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

\[
\approx \sum_{(x,y)\in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2
\]

\[
= \sum_{(x,y)\in W} [I_x u + I_y v]^2 = \sum_{(x,y)\in W} I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2
\]
Corner Detection: Mathematics

The quadratic approximation can be written as

\[ E(u, v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2 \]

\[ = \begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

**Second moment matrix** \( M \)
Interpreting the second moment matrix

- The surface $E(u, v)$ is locally approximated by a quadratic form. Let's try to understand its shape.
- Specifically, in which directions does it have the fastest/slowest change?
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 $E(u, v)$: $\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

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 $R$
Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical)

\[
M = \begin{bmatrix}
\sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\
\sum_{x,y} I_x I_y & \sum_{x,y} I_y^2
\end{bmatrix} = \begin{bmatrix}
a & 0 \\
0 & b
\end{bmatrix}
\]

If either \(a\) or \(b\) is close to 0, then this is not a corner, so look for locations where both are large.
Visualization of second moment matrices
Visualization of second moment matrices
Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

- **“Corner”**: $\lambda_1$ and $\lambda_2$ are large, $\lambda_1 \sim \lambda_2$; $E$ increases in all directions.
- **“Edge”**: $\lambda_2 \gg \lambda_1$.
- **“Flat” region**: $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions.
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)
The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix $M$ in a Gaussian window around each pixel:

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

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix $M$ in a Gaussian window around each pixel
3. Compute corner response function $R$

Harris Detector: Steps
Harris Detector: Steps

Compute corner response $R$
The Harris corner detector

1. Compute partial 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

Find points with large corner response: \( R > \text{threshold} \)
Harris Detector: Steps

Take only the points of local maxima of $R$
Harris Detector: Steps
Robustness of corner features

• What happens to corner features when the image undergoes geometric or photometric transformations?
Affine intensity change

- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a I$

*Partially invariant to affine intensity change*
Derivatives and window function are shift-invariant

Corner location is *covariant* w.r.t. translation
Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation
Scaling

Corner location is not covariant w.r.t. scaling!

All points will be classified as edges