Lecture 12: Corner Detection

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little, Leon Sigal and Fred Tung )
Menu for Today (February 8)

Topics:
— Corner Detection
— Autocorrelation
— Harris Corner Detector

Readings:
— Today’s Lecture: Forsyth & Ponce (2nd ed.) 5.3.0 - 5.3.1
— Next Lecture: Forsyth & Ponce (2nd ed.) 6.1, 6.3

Reminders:
— Assignment 2: Face Detection in a Scaled Representation is Due February 12
Today’s “fun” Example:
Today’s “fun” Example:
Today’s “fun” Example:
Lecture 11: Re-cap

Physical properties of a 3D scene cause “edges” in an image:
— depth discontinuity
— surface orientation discontinuity
— reflectance discontinuity
— illumination boundaries

Two generic approaches to edge detection:
— local extrema of a first derivative operator → Canny
— zero crossings of a second derivative operator → Marr/Hildreth

Many algorithms consider “boundary detection” as a high-level recognition task and output a probability or confidence that a pixel is on a human-perceived boundary
Laplacian is a Bandpass Filter

- **image**
- **FFT (Mag)**
- **Low pass**
- **filtered image**

- **complex element-wise multiplication**
- **lower sigma**
- **larger sigma**
- **filtered image**
Laplacian is a Bandpass Filter
Laplacian is a Bandpass Filter

- FFT (Mag) → complex element-wise multiplication → Low pass (Mag)
- Low pass (Mag) - lower sigma → complex element-wise multiplication → Low pass (Mag)
- Low pass (Mag) - larger sigma →
Image Blending

https://becominghuman.ai/image-blending-using-laplacian-pyramids-2f8e9982077f
Image Blending

https://becominghuman.ai/image-blending-dpecting-laplacian-pyramids-2f8e9982077f
Image Blending

https://becominghuman.ai/image-blending-using-laplacian-pyramids-2f8e9982077f
Image Blending

**High-level Intuition:** Smoother blending of flatter regions, sharper blending of more detailed regions
Motivation: Template Matching

When might **template matching** fail?

— Different scales
— Different orientation
— Lighting conditions
— Left vs. Right hand
— Partial Occlusions
— Different Perspective
— Motion / blur
Motivation: Template Matching in Scaled Representation

When might template matching in scaled representation fail?

- Different scales
- Different orientation
- Lighting conditions
- Left vs. Right hand
- Partial Occlusions
- Different Perspective
- Motion / blur
Motivation: Edge Matching in Scaled Representation

When might edge matching in scaled representation fail?

- Different scales
- Different orientation
- Lighting conditions
- Left vs. Right hand
- Partial Occlusions
- Different Perspective
- Motion / blur
Planar Object **Instance Recognition**

Database of planar objects

Instance recognition

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Recognition under **Occlusion**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Image Matching

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Image Matching

Slide Credit: Ioannis (Yannis) Gkioulkas (CMU)
Finding **Correspondences**

NASA Mars Rover images

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Finding **Correspondences**
What is a **Good Feature**?

Pick a point in the image. Find it again in the next image.

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
What is a **Good Feature**?

Pick a point in the image.
Find it again in the next image.

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
What is a corner?

We can think of a corner as any locally distinct 2D image feature that (hopefully) corresponds to a distinct position on an 3D object of interest in the scene.

Image Credit: John Shakespeare, Sydney Morning Herald
Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:
Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value.

*Image Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

*Image Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Why are corners distinct?

A corner can be localized reliably.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

— Place a small window over an edge.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)
Why are corners distinct?

A corner can be localized reliably.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.
— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change.

→ Cannot estimate location along an edge (a.k.a., aperture problem)

“edge”: no change along the edge direction

Image Credit: Ioannis (Yannis) Gkioulkas (CMU)
Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change

→ Cannot estimate location along an edge (a.k.a., **aperture problem**)

— Place a small window over a corner.

*Image Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Why are corners **distinct**?

A corner can be **localized reliably**.

Thought experiment:

— Place a small window over a patch of constant image value. If you slide the window in any direction, the image in the window will not change.

— Place a small window over an edge. If you slide the window in the direction of the edge, the image in the window will not change.

  → Cannot estimate location along an edge (a.k.a., **aperture** problem)

— Place a small window over a corner. If you slide the window in any direction, the image in the window changes.

*Image Credit: Ioannis (Yannis) Gkioulekas (CMU)*
How do you find a **corner**?

Easily recognized by looking through a small window

Shifting the window should give large change in intensity

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Autocorrelation is the correlation of the image with itself.

— Windows centered on an edge point will have autocorrelation that falls off slowly in the direction along the edge and rapidly in the direction across (perpendicular to) the edge.

— Windows centered on a corner point will have autocorrelation that falls off rapidly in all directions.
Autocorrelation

Szeliski, Figure 4.5
Autocorrelation

Szeliski, Figure 4.5
Autocorrelation

Szeliski, Figure 4.5
Autocorrelation

Szeliski, Figure 4.5
Autocorrelation

Szeliski, Figure 4.5
Autocorrelation

Szeliski, Figure 4.5
Autocorrelation is the correlation of the image with itself.

— Windows centered on an edge point will have autocorrelation that falls off slowly in the direction along the edge and rapidly in the direction across (perpendicular to) the edge.

— Windows centered on a corner point will have autocorrelation that falls off rapidly in all directions.
Corner Detection

Edge detectors perform poorly at corners

**Observations:**
- The gradient is ill-defined exactly at a corner
- Near a corner, the gradient has two (or more) distinct values
Harris Corner Detection

1. Compute image gradients over small region

2. Compute the covariance matrix

3. Compute eigenvectors and eigenvalues

4. Use threshold on eigenvalues to detect corners

\[
I_x = \frac{\partial I}{\partial x} \\
I_y = \frac{\partial I}{\partial y}
\]

\[
\begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]
1. Compute **image gradients** over a small region (not just a single pixel)

\[
I_x = \frac{\partial I}{\partial x}
\]

array of x gradients

\[
I_y = \frac{\partial I}{\partial y}
\]

array of y gradients

*Slide Credit*: Ioannis (Yannis) Gkioulekas (CMU)
Visualization of Gradients

image

X derivative

Y derivative

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
What Does a **Distribution** Tells You About the **Region**?

\[ I_y = \frac{\partial I}{\partial y} \quad \quad I_x = \frac{\partial I}{\partial x} \]

\[ I_y = \frac{\partial I}{\partial y} \quad \quad I_x = \frac{\partial I}{\partial x} \]

\[ I_y = \frac{\partial I}{\partial y} \quad \quad I_x = \frac{\partial I}{\partial x} \]

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
**What Does a Distribution Tells You About the Region?**

Distribution reveals the **orientation** and **magnitude**
What Does a **Distribution** Tells You About the **Region**?

**Distribution reveals the **orientation** and **magnitude****

How do we quantify the **orientation** and **magnitude**?

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

\[ C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix} \]

- **Sum** over small region around the corner
- **Gradient** with respect to x, times gradient with respect to y
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]

- Sum over small region around the corner
- Gradient with respect to x, times gradient with respect to y

\[
\sum_{p \in P} I_x I_y = \text{sum}( \text{array of x gradients} \ast \text{array of y gradients} )
\]
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

Sum over small region around the corner

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]

Gradient with respect to x, times gradient with respect to y

Matrix is **symmetric**
2. Compute the **covariance matrix** (a.k.a. 2nd moment matrix)

By computing the **gradient covariance matrix** ...

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]

we are fitting a **quadratic** to the gradients over a small image region
Simple Case

Local Image Patch

\[ C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix} = ? \]
Simple Case

Local Image Patch

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
= ?
\]
Simple Case

- **Local Image Patch**: high value along vertical strip of pixels and 0 elsewhere

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix} = ?
\]
Simple Case

Local Image Patch

\[ I_x \]

high value along vertical strip of pixels and 0 elsewhere

\[ I_y \]

high value along horizontal strip of pixels and 0 elsewhere

\[ C = \begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix} = ? \]
Simple Case

Local Image Patch

\[ C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix} = \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} \]
General Case

It can be shown that since every $C$ is symmetric:

$$C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix} = R^{-1} \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} R$$

... so general case is like a rotated version of the simple one.
3. Computing **Eigenvalues** and **Eigenvectors**
Quick **Eigenvalue/Eigenvector** Review

Given a square matrix \( \mathbf{A} \), a scalar \( \lambda \) is called an **eigenvalue** of \( \mathbf{A} \) if there exists a nonzero vector \( \mathbf{v} \) that satisfies

\[
\mathbf{A}\mathbf{v} = \lambda\mathbf{v}
\]

The vector \( \mathbf{v} \) is called an **eigenvector** for \( \mathbf{A} \) corresponding to the eigenvalue \( \lambda \).

The eigenvalues of \( \mathbf{A} \) are obtained by solving \( \det(\mathbf{A} - \lambda I) = 0 \).
3. Computing **Eigenvalues** and **Eigenvectors**

\[ Ce = \lambda e \]

\[ (C - \lambda I)e = 0 \]

*Slide Credit: Ioannis (Yannis) Gkioulkas (CMU)*
3. Computing **Eigenvalues** and **Eigenvectors**

\[ Ce = \lambda e \]

\[ (C - \lambda I)e = 0 \]

1. Compute the determinant of \( C - \lambda I \) (returns a polynomial)

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
3. Computing **Eigenvalues** and **Eigenvectors**

- **Eigenvalue**
  \[ Ce = \lambda e \]
- **Eigenvector**
  \[ (C - \lambda I)e = 0 \]

1. Compute the determinant of \( C - \lambda I \) (returns a polynomial)
2. Find the roots of polynomial \( det(C - \lambda I) = 0 \) (returns eigenvalues)

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
3. Computing **Eigenvalues** and **Eigenvectors**

1. Compute the determinant of
   (returns a polynomial)

2. Find the roots of polynomial
   (returns eigenvalues)

3. For each eigenvalue, solve
   (returns eigenvectors)

---

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example

\[ C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \]

1. Compute the determinant of \( C \) (returns a polynomial)

\[ C - \lambda I \]

2. Find the roots of polynomial \( C - \lambda I \) (returns eigenvalues)

\[ \det(C - \lambda I) = 0 \]

3. For each eigenvalue, solve \( (C - \lambda I)e = 0 \) (returns eigenvectors)

\[ (C - \lambda I)e = 0 \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
### Example

\[ C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \]

\[
\det \left( \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} \right) \\
(2 - \lambda)(2 - \lambda) - (1)(1)
\]

1. Compute the determinant of \( C - \lambda I \) (returns a polynomial)

2. Find the roots of polynomial (returns eigenvalues) \( \det(C - \lambda I) = 0 \)

3. For each eigenvalue, solve \( (C - \lambda I)e = 0 \) (returns eigenvectors)
Example

\[ C = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \]

\[ \det \left( \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} \right) \]

\[ (2 - \lambda)(2 - \lambda) - (1)(1) = 0 \]

\[ \lambda^2 - 4\lambda + 3 = 0 \]

\[ (\lambda - 3)(\lambda - 1) = 0 \]

\[ \lambda_1 = 1, \lambda_2 = 3 \]

1. Compute the determinant of \( C - \lambda I \) (returns a polynomial)

\[ C - \lambda I \]

2. Find the roots of polynomial \( (C - \lambda I) \) (returns eigenvalues)

\[ \det(C - \lambda I) = 0 \]

3. For each eigenvalue, solve \( (C - \lambda I)e = 0 \) (returns eigenvectors)

\[ (C - \lambda I)e = 0 \]
Visualization as **Quadratic**

\[ f(x, y) = x^2 + y^2 \]

can be written in matrix form like this…

\[ f(x, y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \]
Visualization as **Quadratic**

\[ f(x, y) = x^2 + y^2 \]

can be written in matrix form like this…

\[
\begin{bmatrix}
  x & y \\
  \end{bmatrix}
\begin{bmatrix}
  1 & 0 \\
  0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
\end{bmatrix}
\]

Result of Computing **Eigenvalues** and **Eigenvectors** (using SVD)

\[
\begin{bmatrix}
  1 & 0 \\
  0 & 1 \\
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 \\
  0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  1 & 0 \\
  0 & 1 \\
\end{bmatrix}
\]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Visualization as **Ellipse**

Since \( C \) is symmetric, we have

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

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

Ellipse equation:

\[
f(x, y) = \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \text{const}
\]
Visualization as **Ellipse**

Since $C$ is symmetric, we have

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

We can visualize $C$ as an ellipse with axis lengths determined by the eigenvalues and orientation determined by $R$.

**Ellipse equation:**

$$f(x, y) = \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \text{const}$$

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Interpreting Eigenvalues

What kind of image patch does each region represent?

\[ \lambda_2 \gg \lambda_1 \]

\[ \lambda_1 \approx 0 \]

\[ \lambda_2 \approx 0 \]

\[ \lambda_1 \gg \lambda_2 \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Interpreting **Eigenvalues**

- \( \lambda_2 \) >> \( \lambda_1 \)
- \( \lambda_1 \sim \lambda_2 \)
- \( \lambda_1 \gg \lambda_2 \)
- 'horizontal' edge
- 'vertical' edge
- flat
- corner

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Interpreting Eigenvalues

\[ \lambda_2 \quad \text{`horizontal' edge} \]
\[ \lambda_2 \gg \lambda_1 \]
\[ \lambda_1 \sim \lambda_2 \]
\[ \lambda_1 \gg \lambda_2 \]

\[ \text{`vertical' edge} \]

\[ \text{flat} \]

\[ \text{corner} \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Interpreting Eigenvalues

- $\lambda_2 >> \lambda_1$
- $\lambda_1 \sim \lambda_2$
- $\lambda_1 >> \lambda_2$

'Square' edge: flat
'Horizontal' edge
'Vertical' edge

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Interpreting Eigenvectors
4. Threshold on Eigenvalues to Detect Corners
4. **Threshold on Eigenvalues to Detect Corners**

(a function of )

Think of a function to score ‘cornerness’

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
4. **Threshold on Eigenvalues to Detect Corners**

(a function of )

Think of a function to score ‘cornerness’

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
4. **Threshold on Eigenvalues to Detect Corners**

(a function of )

Use the **smallest eigenvalue** as the response function

\[ \min(\lambda_1, \lambda_2) \]

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
4. Threshold on Eigenvalues to Detect Corners

(a function of )

Eigenvalues need to be bigger than one:

$$\lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2$$

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
4. **Threshold on Eigenvalues to Detect Corners**

(a function of $\lambda$)

Eigenvalues need to be bigger than one:

$$\lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

$$= \det(C') - \kappa \text{trace}^2(C')$$

(more efficient)

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
4. **Threshold on Eigenvalues to Detect Corners**

\( \det(M) - \kappa \text{trace}^2(M) < 0 \) (a function of)

\[
\lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2 = \det(C') - \kappa \text{trace}^2(C')
\]

Eigenvalues need to be bigger than one:

(more efficient)

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
4. **Threshold on Eigenvalues to Detect Corners**

(a function of $\kappa$)

- **Harris & Stephens (1988)**
  \[ \det(C') - \kappa \text{trace}^2(C') \]

- **Kanade & Tomasi (1994)**
  \[ \min(\lambda_1, \lambda_2) \]

- **Nobel (1998)**
  \[ \frac{\det(C')}{\text{trace}(C') + \epsilon} \]

*Slide Credit: Ioannis (Yannis) Gkioulkas (CMU)*
Harris Corner Detection Review

— Filter image with **Gaussian**

— Compute magnitude of the x and y **gradients** at each pixel

— Construct C in a window around each pixel
  — Harris uses a **Gaussian window**

— Solve for product of the λ’s

— If λ’s both are big (product reaches local maximum above threshold) then we have a corner
  — Harris also checks that ratio of λs is not too high
Compute the **Covariance Matrix**

**Sum** can be implemented as an (unnormalized) box filter with

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]

Harris uses a **Gaussian** weighting instead
Compute the **Covariance Matrix**

Sum can be implemented as an (unnormalized) box filter with

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]

Harris uses a **Gaussian** weighting instead

(has to do with bilinear Taylor expansion of 2D function that measures change of intensity for small shifts … remember AutoCorrelation)
Harris Corner Detection Review

- Filter image with **Gaussian**

- Compute magnitude of the x and y **gradients** at each pixel

- Construct C in a window around each pixel
  - Harris uses a **Gaussian window**

- Solve for product of the λ’s

- If λ’s both are big (product reaches local maximum above threshold) then we have a corner
  - Harris also checks that ratio of λs is not too high

\[ \text{det}(C) - \kappa \text{trace}^2(C) \]

Harris & Stephens (1988)
Harris Corner Detection Review

— Filter image with **Gaussian**

— Compute magnitude of the x and y **gradients** at each pixel

— Construct C in a window around each pixel
  — Harris uses a **Gaussian window**

— Solve for product of the λ’s

— If λ’s both are big (product reaches local maximum above threshold) then we have a corner
  — Harris also checks that ratio of λs is not too high
Properties: Rotational Invariance

Ellipse rotates but its shape (eigenvalues) remains the same

Corner response is invariant to image rotation

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Properties: (partial) Invariance to Intensity Shifts and Scaling

Only derivatives are used -> Invariance to intensity shifts

Intensity scale could affect performance
Properties: NOT Invariant to Scale Changes
Example 1:
**Example 2: Wagon Wheel (Harris Results)**

$\sigma = 1$ (219 points) $\sigma = 2$ (155 points) $\sigma = 3$ (110 points) $\sigma = 4$ (87 points)
Example 3: Crash Test Dummy (Harris Result)

corner response image

$\sigma = 1$ (175 points)

Original Image Credit: John Shakespeare, Sydney Morning Herald
Summary Table

Summary of what we have seen so far:

<table>
<thead>
<tr>
<th>Representation</th>
<th>Result is. . .</th>
<th>Approach</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>intensity</td>
<td>dense</td>
<td>template matching</td>
<td>(normalized) correlation</td>
</tr>
<tr>
<td>edge</td>
<td>relatively sparse</td>
<td>derivatives</td>
<td>$\nabla^2 G$, Canny</td>
</tr>
<tr>
<td>corner</td>
<td>sparse</td>
<td>locally distinct features</td>
<td>Harris</td>
</tr>
</tbody>
</table>
Properties: NOT Invariant to Scale Changes

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Intuitively …

Find local maxima in both position and scale

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Formally ...

Highest response when the signal has the same characteristic scale as the filter
Characteristic Scale

characteristic scale - the scale that produces peak filter response

we need to search over characteristic scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian Filter at Different Scales**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian** Filter at Different **Scales**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian** Filter at Different **Scales**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian Filter** at Different **Scales**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Applying Laplacian Filter at Different Scales

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian** Filter at Different **Scales**
Applying **Laplacian** Filter at Different **Scales**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Applying **Laplacian** Filter at Different **Scales**

Full size

3/4 size

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Applying **Laplacian** Filter at Different **Scales**

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian** Filter at Different **Scales**

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian** Filter at Different **Scales**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Applying **Laplacian** Filter at Different **Scales**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Applying **Laplacian Filter at Different Scales**

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Applying **Laplacian** Filter at Different **Scales**

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Applying **Laplacian** Filter at Different **Scales**
Applying **Laplacian** Filter at Different **Scales**

2.1 4.2 6.0

9.8 15.5 17.0

peak!
Applying \textbf{Laplacian} Filter at Different \textbf{Scales}

2.1

4.2

6.0

9.8

15.5

17.0

\textit{maximum response}
Optimal Scale

2.1  4.2  6.0  9.8  15.5  17.0

Full size image

2.1  4.2  6.0  9.8  15.5  17.0

3/4 size image
Optimal Scale

2.1 4.2 6.0 9.8 15.5 17.0

Full size image

2.1 4.2 6.0 9.8 15.5 17.0

3/4 size image
Optional subtitle

Cross-scale maximum

Local maximum

4.2

Local maximum

6.0

Local maximum

9.8
Implementation

For each level of the Gaussian pyramid compute feature response (e.g. Harris, Laplacian)

For each level of the Gaussian pyramid
  if local maximum and cross-scale
    save scale and location of feature \((x, y, s)\)
Summary

A **corner** is a distinct 2D feature that can be localized reliably

**Edge** detectors perform poorly at corners

→ consider corner detection directly

**Harris** corner detection

— corners are places where intensity gradient direction takes on multiple distinct values

— interpret in terms of autocorrelation of local window

— translation and rotation invariant, but not scale invariant