Lecture 10: Edge Detection (cont.)

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

Topics:
- Corner Detection
- Image Structure
- Harris Corner Detection

Readings:
- Today’s Lecture: Szeliski 7.1-7.2, Forsyth & Ponce 5.3.0 - 5.3.1

Reminders:
- Assignment 2: Scaled Representations, Face Detection and Image Blending (due Monday Feb 13 23:59)
- Midterm: February 27th 3:30pm in class
Goal: Identify sudden changes in image intensity

This is where most shape information is encoded

Example: artist’s line drawing (but artist also is using object-level knowledge)
Good detection: minimize probability of false positives/negatives (spurious/missing) edges

Good localization: found edges should be as close to true image edge as possible

Single response: minimize the number of edge pixels around a single edge

<table>
<thead>
<tr>
<th>Approach</th>
<th>Detection</th>
<th>Localization</th>
<th>Single Resp</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobel</td>
<td>Gradient Magnitude Threshold</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td>Marr / Hildreth</td>
<td>Zero-crossings of 2nd Derivative (LoG)</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Canny</td>
<td>Local extrema of 1st Derivative</td>
<td>Best</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>0</td>
<td>70</td>
<td>128</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td>64</td>
<td>128</td>
<td>196</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>196</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>196</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>196</td>
<td>196</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[
\sqrt{64^2 + 70^2} = \sqrt{4096 + 4900} = \sqrt{8996} = 94.847
\]
\[
\| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}
\]

\[
\sqrt{64^2 + 70^2} = \sqrt{4096 + 4900} = \sqrt{8996} = 94.847
\]
\[ \| \nabla f \| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \]

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
**Sobel issues:**

- Brittle = result depends on threshold
- Thick edges = poor localization
Canny Edge Detector

3. **Non-maximum** suppression
   - thin multi-pixel wide “ridges” down to single pixel width

4. **Linking** and thresholding
   - Low, high edge-strength thresholds
   - Accept all edges over low threshold that are connected to edge over high threshold

![Gradient Magnitude](image)

![Gradient Direction](image)
Canny Non-Maxima Suppression

Gradient Magnitude

Gradient Direction
Canny Non-Maxima Suppression

Gradient Magnitude

Gradient Direction
Canny Non-Maxima Suppression

No longer considered as possible edge points

Can still be edge points

Gradient Magnitude

Gradient Direction
Canny Non-Maxima Suppression

No longer considered as possible edge points

Can still be edge points

Gradient Magnitude

Gradient Direction
Canny Non-Maxima Suppression

Gradient Magnitude

Gradient Direction

No longer considered as possible edge points

Can still be edge points

No longer considered as possible edge points

Can still be edge points
Canny Non-Maxima Suppression

Gradient Magnitude

Gradient Direction

No longer considered as possible edge points

Can still be edge points
Canny Non-Maxima Suppression

No longer considered as possible edge points

Can still be edge points

Gradient Magnitude

Gradient Direction
Canny Non-Maxima Suppression

No longer considered as possible edge points

Can still be edge points

Gradient Magnitude

Gradient Direction
Goal:

— Identify local maxima, which can be edge points
— Thin edges, so we can improve localization
Gradient Magnitude

Gradient Direction

Canny Non-Maxima Suppression

Linking Edge Points

gradient magnitude > $k_{high} = 100$

$k_{low} < \text{gradient magnitude} < k_{high}$

gradient magnitude < $k_{low} = 50$
Gradient Magnitude

Gradient Direction

Canny Non-Maxima Suppression

Linking Edge Points

gradient magnitude > $k_{high} = 100$

$k_{low} < \text{gradient magnitude} < k_{high}$

gradient magnitude < $k_{low} = 50$
### Canny Non-Maxima Suppression

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>196</th>
<th>0</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>128</td>
<td>277</td>
<td>0</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>95</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>95</td>
<td>82</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>93</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Linking Edge Points

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>196</th>
<th>0</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>128</td>
<td>277</td>
<td>0</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>95</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>95</td>
<td>82</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>93</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Gradient Magnitude

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>196</th>
<th>0</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>128</td>
<td>277</td>
<td>0</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>95</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>95</td>
<td>82</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>93</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Gradient Direction

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>90</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>45</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>48</td>
<td>45</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>42</td>
<td>45</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-45</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>-90</td>
<td>-90</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>90</td>
<td>45</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### k<sub>low</sub> < gradient magnitude < k<sub>high</sub>
Canny Non-Maxima Suppression

<table>
<thead>
<tr>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>196</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>128</td>
<td>277</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>91</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>95</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>95</td>
<td>82</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>93</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Linking Edge Points

<table>
<thead>
<tr>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>196</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>128</td>
<td>277</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>91</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>95</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>95</td>
<td>82</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>93</td>
<td>0</td>
</tr>
<tr>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Gradient Magnitude

<table>
<thead>
<tr>
<th>0</th>
<th>5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>196</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>128</td>
<td>277</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>91</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>95</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>95</td>
<td>82</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>91</td>
<td>96</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>66</td>
<td>93</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>196</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Gradient Direction

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>45</td>
<td>45</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>45</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>42</td>
<td>45</td>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-45</td>
<td>45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-90</td>
<td>-90</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Canny Non-Maxima Suppression

Linking Edge Points

Gradient Magnitude

Gradient Direction

$k_{low} < \text{gradient magnitude} < k_{high}$
Canny Non-Maxima Suppression

Gradient Magnitude

Canny Edge Detector

Linking Edge Points

Gradient Direction
The fact that the edge is shifted can be addressed by better derivative filter (central difference).
How do humans perceive boundaries?

Edges are a property of the 2D image.

It is interesting to ask: How closely do image edges correspond to boundaries that humans perceive to be salient or significant?
Traditional Edge Detection

Generally lacks semantics (i.e., too low-level for many tasks)
How do humans perceive boundaries?

"Divide the image into some number of segments, where the segments represent 'things' or 'parts of things' in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance."

(Martin et al. 2004)
How do humans perceive boundaries?
How do humans perceive **boundaries?**

<table>
<thead>
<tr>
<th>Image</th>
<th>Illustration</th>
<th>Illustration</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="illustration1.png" alt="Illustration" /></td>
<td><img src="illustration2.png" alt="Illustration" /></td>
<td><img src="illustration3.png" alt="Illustration" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="illustration4.png" alt="Illustration" /></td>
<td><img src="illustration5.png" alt="Illustration" /></td>
<td><img src="illustration6.png" alt="Illustration" /></td>
</tr>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="illustration7.png" alt="Illustration" /></td>
<td><img src="illustration8.png" alt="Illustration" /></td>
<td><img src="illustration9.png" alt="Illustration" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="illustration10.png" alt="Illustration" /></td>
<td><img src="illustration11.png" alt="Illustration" /></td>
<td><img src="illustration12.png" alt="Illustration" /></td>
</tr>
</tbody>
</table>

**Figure Credit**: Martin et al. 2001
How do humans perceive boundaries?

Each image shows multiple (4-8) human-marked boundaries. Pixels are darker where more humans marked a boundary.

Figure Credit: Szeliski Fig. 4.31. Original: Martin et al. 2004
We can formulate **boundary detection** as a high-level recognition task

— Try to learn, from sample human-annotated images, which visual features or cues are predictive of a salient/significant boundary

Many boundary detectors output a **probability or confidence** that a pixel is on a boundary
Boundary Detection: Example Approach

— Consider circular windows of radii $r$ at each pixel $(x, y)$ cut in half by an oriented line through the middle

— Compare visual features on both sides of the cut

— If features are very different on the two sides, the cut line probably corresponds to a boundary

— Notice this gives us an idea of the orientation of the boundary as well
Boundary Detection: Example Approach

— Consider circular windows of radii \( r \) at each pixel \((x, y)\) cut in half by an oriented line through the middle

— Compare visual features on both sides of the cut

— If features are very different on the two sides, the cut line probably corresponds to a boundary

— Notice this gives us an idea of the orientation of the boundary as well

Implementation: consider 8 discrete orientations \((\theta)\) and 3 scales \((r)\)
Boundary Detection:

Features:

— Raw Intensity
— Orientation Energy
— Brightness Gradient
— Color Gradient
— Texture gradient
Boundary Detection:

For each feature type

— Compute non-parametric distribution (histogram) for left side
— Compute non-parametric distribution (histogram) for right side
— Compare two histograms, on left and right side, using statistical test

Use all the histogram similarities as features in a learning based approach that outputs probabilities (Logistic Regression, SVM, etc.)
Boundary Detection: Example Approach

Figure Credit: Szeliski Fig. 4.33. Original: Martin et al. 2004
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.
Lecture 10: Corner Detection

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little and Fred Tung )
Motivation: Template Matching

When might **template matching fail**?

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

- 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**
Image Matching

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

**Local**: features are local, robust to occlusion and clutter

**Accurate**: precise localization

**Robust**: noise, blur, compression, etc. do not have a big impact on the feature.

**Distinctive**: individual features can be easily matched

**Efficient**: close to real-time performance
What is a **Good Feature**?
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
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.
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) Gkioulkekas (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)

*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.

_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

[Moravec 1980]

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

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.
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

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

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

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

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

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**?

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

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)
What Does a **Distribution** Tells You About the **Region**?

**Distribution reveals the orientation and magnitude**

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

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)

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
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)

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

Matrix is **symmetric**

**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)

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
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

\[
C = \begin{bmatrix}
\sum_{p \in P} I_{x}I_{x} & \sum_{p \in P} I_{x}I_{y} \\
\sum_{p \in P} I_{y}I_{x} & \sum_{p \in P} I_{y}I_{y}
\end{bmatrix} = ?
\]

high value along vertical strip of pixels and 0 elsewhere

high value along horizontal strip of pixels and 0 elsewhere
Simple Case

Local Image Patch

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

\( I_x \) high value along vertical strip of pixels and 0 elsewhere

\( I_y \) high value along horizontal strip of pixels and 0 elsewhere
General Case

\[
C = \begin{bmatrix}
\sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\
\sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y
\end{bmatrix}
\]
It can be shown that since every $C$ is symmetric:

$$C = \left[ \begin{array}{cc} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{array} \right] = R^{-1} \left[ \begin{array}{cc} \lambda_1 & 0 \\ 0 & \lambda_2 \end{array} \right] R$$

... 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 $A$, a scalar $\lambda$ is called an **eigenvalue** of $A$ if there exists a nonzero vector $v$ that satisfies

$$Av = \lambda v$$

The vector $v$ is called an **eigenvector** for $A$ corresponding to the eigenvalue $\lambda$.

The eigenvalues of $A$ are obtained by solving

$$\det(A - \lambda I) = 0$$
3. Computing **Eigenvalues** and **Eigenvectors**

\[ Ce = \lambda e \]

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

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

\[ Ce = \lambda e \]

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

1. Compute the determinant of
   (returns a polynomial)

\[ C - \lambda I \]
3. Computing **Eigenvalues** and **Eigenvectors**

\[ Ce = \lambda e \]

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

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

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

\[ C - \lambda I \]

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

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

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

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

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

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

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

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

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

2. Find the roots of polynomial (returns eigenvalues)
\[ \text{det}(C - \lambda I) = 0 \]

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

\( \text{Slide Credit}: \) 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 (returns a polynomial) \( C - \lambda I \)

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

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

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

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

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

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

2. Find the roots of polynomial (returns eigenvalues)

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

3. For each eigenvalue, solve (returns eigenvectors)

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

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

1. Compute the determinant of
   (returns a polynomial)
   \[ \det \left( \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix} \right) \]

2. Find the roots of polynomial
   (returns eigenvalues)
   \[ (2 - \lambda)(2 - \lambda) - (1)(1) = 0 \]
   \[ \lambda^2 - 4\lambda + 3 = 0 \]
   \[ (\lambda - 3)(\lambda - 1) = 0 \]
   \[ \lambda_1 = 1, \lambda_2 = 3 \]

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

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

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

What kind of image patch does each region represent?

- $\lambda_2 \gg \lambda_1$
- $\lambda_1 \sim 0$
- $\lambda_2 \sim 0$
- $\lambda_1 \gg \lambda_2$

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

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

- 'horizontal' edge
- 'vertical' edge
- corner
- flat

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

\[ \lambda_2 \gg \lambda_1 \]

\[ \lambda_1 \sim \lambda_2 \]

\[ \lambda_1 \gg \lambda_2 \]

\[ \lambda_2 \gg \lambda_1 \]

'horizontal' edge

'vertical' edge

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

\[ \lambda_2 \gg \lambda_1 \]

\[ \lambda_1 \sim \lambda_2 \]

\[ \lambda_1 \gg \lambda_2 \]

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

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

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

\[
\min(\lambda_1, \lambda_2)
\]
4. Threshold on Eigenvalues to Detect Corners

(a function of )

\[ \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2 \]

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

(a function of )

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

(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 $\lambda$)

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

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

(more efficient)

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

(a function of )

**Harris & Stephens (1988)**

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

**Kanade & Tomasi (1994)**

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

**Nobel (1998)**

\[
\frac{\text{det}(C')}{\text{trace}(C') + \epsilon}
\]

*Slide Credit: Ioannis (Yannis) Gkioulekas (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 \( \lambda \)'s

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

Harris & Stephens (1988)

$$\det(C) - \kappa \text{trace}^2(C)$$
Example: Harris Corner Detection

```
0 0 0 0 0 0
0 1 0 0 0 1
0 1 1 1 1 0
0 1 1 1 1 0
0 0 1 1 1 0
0 0 1 1 1 0
0 0 1 1 1 0
0 0 1 1 1 0
```

Example: Harris Corner Detection

Let's compute a measure of “corner-ness” for the green pixel:
**Example: Harris Corner Detection**

Let's compute a measure of “corner-ness” for the green pixel:
Example: Harris Corner Detection

Let's compute a measure of “corner-ness” for the green pixel:

$$I_x = \frac{\partial I}{\partial x}$$

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example: Harris Corner Detection

Let's compute a measure of "corner-ness" for the green pixel:

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

\[ I_y = \frac{\partial I}{\partial y} \]
Example: Harris Corner Detection

Let's compute a measure of “corner-ness” for the green pixel:

\[
\sum \begin{bmatrix}
0 & 0 & 0 \\
0 & -1 & 1 \\
0 & 1 & 0
\end{bmatrix} \circ \begin{bmatrix}
0 & 0 & 0 \\
-1 & 1 \\
0 & 1 & 0
\end{bmatrix} = 3
\]

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

\[
I_y = \frac{\partial I}{\partial y}
\]
Example: Harris Corner Detection

Lets compute a measure of “corner-ness” for the green pixel:

\[
C = \begin{bmatrix}
3 & 2 \\
2 & 4 \\
\end{bmatrix}
\]

\[
I_x = \frac{\partial I}{\partial x} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & 1 \\
-1 & 0 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 1 \\
0 & -1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
I_y = \frac{\partial I}{\partial y} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Example: Harris Corner Detection

Let's compute a measure of “corner-ness” for the green pixel:

\[ C = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} \Rightarrow \lambda_1 = 1.4384; \lambda_2 = 5.5616 \]

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

\[ I_y = \frac{\partial I}{\partial y} \]
Example: Harris Corner Detection

Let’s compute a measure of “corner-ness” for the green pixel:

\[
C = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix} \Rightarrow \lambda_1 = 1.4384; \lambda_2 = 5.5616
\]

\[
det(C) - 0.04\text{trace}^2(C) = 6.04
\]
Example: Harris Corner Detection

Let's compute a measure of “corner-ness” for the green pixel:

\[
C = \begin{bmatrix}
3 & 0 \\
0 & 0 \\
\end{bmatrix}
\]

\[
\Rightarrow \lambda_1 = 3; \lambda_2 = 0
\]

\[
\det(C) - 0.04 \text{trace}^2(C) = -0.36
\]
**Example: Harris Corner Detection**

Let's compute a measure of "corner-ness" for the green pixel:

\[
C = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \Rightarrow \lambda_1 = 3; \lambda_2 = 2
\]

\[
\det(C) - 0.04\text{trace}^2(C) = 5
\]
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 effect performance

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Properties: NOT Invariant to Scale Changes

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

Find local maxima in both position and scale

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example 1:

- Harris corners
  - Originally developed as features for motion tracking
  - Greatly reduces amount of computation compared to tracking every pixel
  - Translation and rotation invariant (but not scale invariant)
Example 2: Wagon Wheel (Harris Results)

\[ \sigma = 1 \text{ (219 points)} \quad \sigma = 2 \text{ (155 points)} \quad \sigma = 3 \text{ (110 points)} \quad \sigma = 4 \text{ (87 points)} \]
Example 3: Crash Test Dummy (Harris Result)

corner response image

\[ \sigma = 1 \text{ (175 points)} \]

*Original Image Credit:* John Shakespeare, Sydney Morning Herald
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 \((x, y, s)\)
Multi-Scale Harris Corners
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>
Summary

**Edges** are useful image features for many applications, but suffer from the aperture problem.

**Canny** Edge detector combines edge filtering with linking and hysteresis steps.

**Corners / Interest Points** have 2D structure and are useful for correspondence.

**Harris** corners are minima of a local SSD function.

**DoG** maxima can be reliably located in scale-space and are useful as interest points.