Edge and corner detection

Prof. Stricker
Doz. Dr. G. Bleser

Computer Vision: Object and People Tracking
Example of the State of the Art (Video / Augmented Vision Group)
Artificial (computer) Vision: Tracking
Reminder: „tracking on one slide“

Object
Sensors
Context

Models

Object
Tracking

Vision

Tracking

Detection / Prediction

Measurement

Data association

Data fusion

State update

Image acquisition

Likelihood

Modalities
Goals

• Where is the information in an image?
• How is an object characterized?
• How can I find measurements in the image?

• The correct „Good Features“ are essential for tracking!
Outline

• Edge detection
• Canny edge detector
• Point extraction
Edge detection

**Goal:** Identify sudden changes (discontinuities) in an image

- Intuitively, most semantic and shape information from the image can be encoded in the edges
- More compact than pixels

**Ideal:** artist’s line drawing (but artist is also using object-level knowledge)
Edge Detection
What Causes Intensity Changes?

Geometric events

- surface orientation (boundary) discontinuities
- depth discontinuities
- color and texture discontinuities

Non-geometric events

- illumination changes
- specularities
- shadows
- inter-reflections
Why is Edge Detection Useful?

Important features can be extracted from the edges of an image (e.g., corners, lines, curves). These features are used by higher-level computer vision algorithms (e.g., recognition).
Effect of Illumination
Edge Descriptors

**Edge direction:**
perpendicular to the
direction of maximum
intensity change (i.e.,
edge normal)

**Edge strength:** related to
the local image contrast
along the normal.

**Edge position:** the image
position at which the
edge is located.
Characterizing edges

- An edge is a place of rapid change in the image intensity function.
Image gradient

The gradient of an image:

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid increase in intensity.

The gradient direction is given by

- how does this relate to the direction of the edge?

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

The edge strength is given by the gradient magnitude

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]

Source: Steve Seitz
Differentiation and convolution

Recall, for 2D function, \( f(x,y) \):

\[
\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)
\]

We could approximate this as:

\[
\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}
\]

(which is obviously a convolution)

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

Check!

Source: D. Forsyth, D. Lowe
Finite differences: example

Which one is the gradient in the $x$-direction (resp. $y$-direction)?
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]

Where is the edge?

Source: S. Seitz
Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response

- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors
Solution: smooth first

- To find edges, look for peaks in \( \frac{d}{dx}(f \ast g) \)

Source: S. Seitz
Derivative theorem of convolution

• Differentiation and convolution both linear operators: they “commute”

\[
\frac{d}{dx} (f \ast g) = \frac{df}{dx} \ast g = f \ast \frac{dg}{dx}
\]

• This saves us one operation:
Derivative of Gaussian filter

This filter is separable

* [1 -1] =
Derivative of Gaussian filter

$x$-direction

$y$-direction
Tradeoff between smoothing and localization

Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth
Finite difference filters

- Other approximations of derivative filters exist:

<table>
<thead>
<tr>
<th>Filter</th>
<th>$M_x$</th>
<th>$M_y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prewitt</td>
<td>$\begin{bmatrix} -1 &amp; 0 &amp; 1 \ -1 &amp; 0 &amp; 1 \ -1 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 1 &amp; 1 \ 0 &amp; 0 &amp; 0 \ -1 &amp; -1 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Sobel</td>
<td>$\begin{bmatrix} -1 &amp; 0 &amp; 1 \ -2 &amp; 0 &amp; 2 \ -1 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 2 &amp; 1 \ 0 &amp; 0 &amp; 0 \ -1 &amp; -2 &amp; -1 \end{bmatrix}$</td>
</tr>
<tr>
<td>Roberts</td>
<td>$\begin{bmatrix} 0 &amp; 1 \ -1 &amp; 0 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 0 \ 0 &amp; -1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
Implementation issues

- The gradient magnitude is large along a thick “trail” or “ridge”, so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth
Designing an edge detector

• Criteria for an “optimal” edge detector:
  • **Good detection**: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  • **Good localization**: the edges detected must be as close as possible to the true edges
  • **Single response**: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge

Source: L. Fei-Fei
Canny edge detector

• This is probably the most widely used edge detector in computer vision

• Theoretical model: step-edges corrupted by additive Gaussian noise

• Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization

Canny edge detector

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

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

4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

MATLAB: edge(image, ‘canny’)
Example

original image (Lena)
Example

norm of the gradient
Example

thresholding
Example

thinning
(non-maximum suppression)
Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

Source: D. Forsyth
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either $r$ or $s$).
Hysteresis thresholding

Check that maximum value of gradient value is sufficiently large

• drop-outs? use **hysteresis**
  
  – use a high threshold to start edge curves and a low threshold to continue them.
Hysteresis thresholding

Original image

High threshold (strong edges)

Low threshold (weak edges)

Hysteresis threshold

Source: L. Fei-Fei
Effect of $\sigma$ (Gaussian kernel spread/size)

The choice of $\sigma$ depends on desired behavior

- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Source: S. Seitz
Edge detection is just the beginning...

Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

image  human segmentation  gradient magnitude
Features
Image Matching
Image Matching
Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …

Feature Descriptors
Advantages of local features

Locality
  • features are local, so robust to occlusion and clutter

Distinctiveness
  • can differentiate a large database of objects

Quantity
  • hundreds or thousands in a single image

Efficiency
  • real-time performance achievable

Generality
  • exploit different types of features in different situations
More motivation…

Feature points are used for:

• Image alignment (e.g., mosaics)
• 3D reconstruction
• **Motion tracking**
• Object recognition
• Indexing and database retrieval
• Robot navigation
• … other
Interest point candidates
Steps in Corner Detection

1. For each pixel, the corner operator is applied to obtain a \textit{cornerness} measure for this pixel.

2. Threshold \textit{cornerness} map to eliminate weak corners.

3. Apply non-maximal suppression to eliminate points whose \textit{cornerness} measure is not larger than the \textit{cornerness} values of all points within a certain distance.
Steps in Corner Detection (cont’d)

- Apply Corner Operator
- Threshold Cornerness Map
- Non-maximal Suppression

Corners superimposed on Input Image
Local measures of uniqueness

Suppose we only consider a small window of pixels

• What defines whether a feature is a good or bad candidate?

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.
Feature detection

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.
Feature detection: the math

Consider shifting the window $W$ by $(u,v)$

- how do the pixels in $W$ change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” of $E(u,v)$:

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

Taylor Series expansion of $I$:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms}$$

If the motion $(u,v)$ is small, then first order approx is good

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

$$\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

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

Plugging this into the formula on the previous slide…
Feature detection: the math

Consider shifting the window $W$ by $(u,v)$
- how do the pixels in $W$ change?
- compare each pixel before and after by summing up the squared differences
- this defines an “error” of $E(u,v)$:

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

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

$$\approx \sum_{(x,y) \in W} \left[[I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}\right]^2$$
Feature detection: the math

This can be rewritten:

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

For the example above

- You can move the center of the green window to anywhere on the blue unit circle
- Which directions will result in the largest and smallest E values?
- We will show that we can find these directions by looking at the eigenvectors of \( H \)
Feature detection: the error function

- A new corner measurement by investigating the **shape** of the error function

\[
E(u, v) = \sum_{(x, y) \in W} [u \ v] \begin{bmatrix}
I_x^2 & I_xI_y \\
I_yI_x & I_y^2
\end{bmatrix} \begin{bmatrix}
u \\
v
\end{bmatrix}
\]

\[u^T H u\] represents a **quadratic function**;
Thus, we can analyze \(E\)'s shape by **looking at the property of** \(H\)
Feature detection: the error function

High-level idea: what shape of the error function will we prefer for features?

flat

edge

corner
Quadratic forms

Quadratic form (homogeneous polynomial of degree two) of $n$ variables $x_i$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_i x_j$$

Examples

$$4x_1^2 + 5x_2^2 + 3x_3^2 + 2x_1 x_2 + 4x_1 x_3 + 6x_2 x_3$$

$$= \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} 4 & 1 & 2 \\ 1 & 5 & 3 \\ 2 & 3 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$
Symmetric matrices

Quadratic forms can be represented by a real symmetric matrix $A$ where

$$a_{ij} = \begin{cases} c_{ij} & \text{if } i = j, \\ \frac{1}{2}c_{ij} & \text{if } i < j, \\ \frac{1}{2}c_{ji} & \text{if } i > j. \end{cases}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$$

$$= \left( \begin{array}{ccc} x_1 & \ldots & x_n \end{array} \right) \left( \begin{array}{ccc} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nn} \end{array} \right) \left( \begin{array}{c} x_1 \\ \vdots \\ x_n \end{array} \right)$$

$$= x^t A x$$
Eigenvalues of symmetric matrices

suppose \( A \in \mathbb{R}^{n \times n} \) is symmetric, i.e., \( A = A^T \)

**fact:** the eigenvalues of \( A \) are real

suppose \( Av = \lambda v, \ v \neq 0, \ v \in \mathbb{C}^n \)

\[
\overline{v}^T A v = \overline{v}^T (Av) = \lambda \overline{v}^T v = \lambda \sum_{i=1}^{n} |v_i|^2
\]

\[
\overline{v}^T A v = (\overline{Av})^T v = (\overline{\lambda v})^T v = \overline{\lambda} \sum_{i=1}^{n} |v_i|^2
\]

we have \( \lambda = \overline{\lambda} \), i.e., \( \lambda \in \mathbb{R} \)

(hence, can assume \( v \in \mathbb{R}^n \))
Eigenvectors of symmetric matrices

Suppose \( A \in \mathbb{R}^{n \times n} \) is symmetric, i.e., \( A = A^T \).

**Fact:** There is a set of orthonormal eigenvectors of \( A \) given by

\[
A = Q \Lambda Q^T
\]

where \( Q \) is an orthogonal matrix (the columns of which are eigenvectors of \( A \)), and \( \Lambda \) is real and diagonal (having the eigenvalues of \( A \) on the diagonal).
Eigenvectors of symmetric matrices

Suppose $A \in \mathbb{R}^{n \times n}$ is symmetric, i.e., $A = A^T$

**Fact:** There is a set of orthonormal eigenvectors of $A$

$A = Q \Lambda Q^T$

$x^T A x = x^T Q \Lambda Q^T x$

$= (Q^T x)^T \Lambda (Q^T x)$

$= y^T \Lambda y$

$= (\Lambda^{\frac{1}{2}} y)^T (\Lambda^{\frac{1}{2}} y)$

$= z^T z$

$z^T z = 1$

$x^T x = 1$
Harris corner detector

Intensity change in shifting window: eigenvalue analysis

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

\( \lambda_- , \lambda_+ \) – eigenvalues of \( H \)

We can visualize \( H \) as an ellipse with axis lengths and directions determined by its eigenvalues and eigenvectors.

Ellipse \( E(u, v) = \text{const} \)
Visualize quadratic functions

\[ A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix}^T \]
Visualize quadratic functions

\[ A = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \]
Visualize quadratic functions

\[ A = \begin{bmatrix} 3.25 & 1.30 \\ 1.30 & 1.75 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T \]
Visualize quadratic functions

\[ A = \begin{bmatrix} 7.75 & 3.90 \\ 3.90 & 3.25 \end{bmatrix} = \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0.50 & -0.87 \\ -0.87 & -0.50 \end{bmatrix}^T \]
Feature detection: the math

This can be rewritten:

\[ E(u, v) = \sum_{(x,y) \in \mathcal{W}} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \]

Eigenvalues and eigenvectors of \( H \)

- Define shifts with the smallest and largest change (E value)
- \( x_+ \) = direction of largest increase in \( E \).
- \( \lambda_+ \) = amount of increase in direction \( x_+ \)
- \( x_- \) = direction of smallest increase in \( E \).
- \( \lambda_- \) = amount of increase in direction \( x_+ \)

\[
H x_+ = \lambda_+ x_+ \\
H x_- = \lambda_- x_- 
\]
Feature detection: the math

How are \( \lambda_+ \), \( x_+ \), \( \lambda_- \), and \( x_- \) relevant for feature detection?

- What’s our feature scoring function?

Want \( E(u,v) \) to be large for small shifts in all directions

- the minimum of \( E(u,v) \) should be large, over all unit vectors \([u \ v]\)
- this minimum is given by the smaller eigenvalue (\( \lambda_- \)) of \( H \)

\[ I \quad \lambda_+ \quad \lambda_- \]
Feature detection summary (Kanade-Tomasi)

Here’s what you do

• Compute the gradient at each point in the image
• Create the $H$ matrix from the entries in the gradient
• Compute the eigenvalues
• Find points with large response ($\lambda_\gamma >$ threshold)
• Choose those points where $\lambda_\gamma$ is a local maximum as features

Feature detection summary

Here’s what you do

• Compute the gradient at each point in the image
• Create the $H$ matrix from the entries in the gradient
• Compute the eigenvalues.
• Find points with large response ($\lambda_\lambda >$ threshold)

**Choose those points where $\lambda_\lambda$ is a local maximum as features**
The Harris operator

\( \lambda_- \) is a variant of the “Harris operator” for feature detection \((\lambda_- = \lambda_1 ; \lambda_+ = \lambda_2)\)

\[
f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)}
\]

- The \textit{trace} is the sum of the diagonals, i.e., \(\text{trace}(H) = h_{11} + h_{22}\)
- Very similar to \(\lambda_-\) but less expensive (no square root)*
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular

\[
\lambda_\pm = \frac{1}{2} \left[ (h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]
\]

*
The Harris operator

Measure of corner response (Harris):

\[
R = \det H - k(\text{trace} H)^2
\]

With:

\[
\det H = \lambda_1 \lambda_2 \\
\text{trace } H = \lambda_1 + \lambda_2 \\
\]

\(k = \text{empirical constant, } k = 0.04-0.06\)
The Harris operator
Harris detector example
f value (red high, blue low)
Threshold \( f > \text{value} \)
Find local maxima of $f$
Harris features (in red)
Harris detector: Steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix $H$ in a Gaussian window around each pixel
3. Compute corner response function $R$
4. Threshold $R$
5. Find local maxima of response function (non-maximum suppression)

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

$\alpha$: constant (0.04 to 0.06)

Thank you!
Quick review: eigenvalue/eigenvector

The **eigenvectors** of a matrix $A$ are the vectors $x$ that satisfy:

$$Ax = \lambda x$$

The scalar $\lambda$ is the **eigenvalue** corresponding to $x$

- The eigenvalues are found by solving:
  $$det(A - \lambda I) = 0$$

- In our case, $A = H$ is a 2x2 matrix, so we have
  $$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:
  $$\lambda_{\pm} = \frac{1}{2} \left[ (h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

Once you know $\lambda$, you find $x$ by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$