CS4501: Introduction to Computer Vision

Interest Points: Corners and Blobs

Various slides from previous courses by:
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL). S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook). J. M. Frahm (UNC), V. Ordonez (UVA).
Last Class

• Frequency Domain
• Filtering in Frequency
• Edge Detection – Canny
Today’s Class

• Corner Detection - Harris
• Interest Points
• Local Feature Descriptors
Edge Detection
Edge Detection

• Sobel + Thresholding

\[ g(x, y) = \begin{cases} 
1, & f(x, y) \geq \tau \\
0, & f(x, y) < \tau 
\end{cases} \]
Digression

- Thresholding is often the most under-rated but effective Computer Vision technique. Sometimes this is all you need!

Example: intensity-based detection. Warning when door is opened or closed.

Looking for dark pixels

\[ \text{fg\_pix} = \text{find} (\text{im} < 65); \]

Example by Kristen Grauman, UT-Austin
Canny Edge Detector

• Obtains thin edges (1px wide)
• Tries to preserve edge connectivity.

Source: Juan C. Niebles and Ranjay Krishna.
Canny edge detector

1. Filter image with \( x, y \) derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   • Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   • Define two thresholds: low and high
   • Use the high threshold to start edge curves and the low threshold to continue them

Source: Juan C. Niebles and Ranjay Krishna.
Canny Edge Detector

• Classic algorithm in Computer Vision / Image Analysis
• Commonly implemented in most libraries

• e.g. in Python you can find it in the skimage package. OpenCV also has an implementation with python bindings.

Corners (and Interest Points)

- How to find corners?
- What is a corner?
Corners: Why “Interest” Points?

If we can find them, then maybe we can “match” images belonging to the same object!

Then maybe we can also “triangulate” the 3D coordinates of the image.

Example from Silvio Savarese
Corner Detection: Basic Idea

• We should easily recognize the point by looking through a small window
• Shifting a window in any direction should give a large change in intensity

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions

Source: A. Efros
Harris Corner Detection

• Compute the following matrix of squared gradients for every pixel.

\[ M = \sum_{\text{patch}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \]

\( I_x \) and \( I_y \) are gradients computed using Sobel or some other approximation.

\[ M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad M = \begin{bmatrix} 0 & 0 \\ 0 & b \end{bmatrix} \quad M = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \quad M = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \]
Simple Corner Detection

\[ M = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad M = \begin{bmatrix} 0 & 0 \\ 0 & b \end{bmatrix} \quad M = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \quad M = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \]

- If both \(a\) and \(b\) are large then this is a corner, otherwise it is not. Set a threshold and this should detect corners.

Compute \(\det(M)\) for every pixel and if bigger than \(\tau\) it is a corner, otherwise not

\[ \det(M) > \tau \]
Problem: Doesn’t work for these corners:

\[
M = \sum_{\text{patch}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}
\]
Harris Corner Detection!

Works for these corners!

\[ M = \sum_{\text{patch}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix} \]

Use the following criteria to decide if it is a corner instead

\[ \det(M) - 0.06 \text{trace}(M)^2 > \tau \]
Why $\det(M) - 0.06 \times \text{trace}(M)$?

$$M = \sum_{\text{patch}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$$

Under a rotation $M$ can be diagonalized

$$M = R_m^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R_m$$

$\lambda_1$ and $\lambda_2$ are the eigenvalues of $M$

From your linear algebra class finding them requires solving

$$\det(M - \lambda I) = 0$$
However Harris argues that there’s no need to find lambdas (the eigenvalues), instead it is enough to know the following:

\[
\det M = \lambda_1 \lambda_2
\]

\[
\text{trace } M = \lambda_1 + \lambda_2
\]
Theorems

Let $A$ be an $n \times n$ matrix.

- The matrix $A$ has $n$ eigenvalues (including each according to its multiplicity).

- The sum of the $n$ eigenvalues of $A$ is the same as the trace of $A$ (that is, the sum of the diagonal elements of $A$).

- The product of the $n$ eigenvalues of $A$ is the same as the determinant of $A$.

- If $\lambda$ is an eigenvalue of $A$, then the dimension of $E_\lambda$ is at most the multiplicity of $\lambda$.

- A set of eigenvectors of $A$, each corresponding to a different eigenvalue of $A$, is a linearly independent set.

- If $\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0$ is the characteristic polynomial of $A$, then $c_{n-1} = -\text{trace}(A)$ and $c_0 = (-1)^n|A|$.

- If $\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0$ is the characteristic polynomial of $A$, then $A^n + c_{n-1}A^{n-1} + \cdots + c_1A + c_0I = O$. (The Cayley-Hamilton Theorem.)
Corner response function

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

\( \alpha \): constant (0.04 to 0.06)
If you still think you need more details

• You can look up the original published work with the Harris Corner Detector from 1988.

Alternative Corner response function

“edge”:
\[ \lambda_1 \gg \lambda_2 \]
\[ \lambda_2 \gg \lambda_1 \]

“corner”:
\[ \lambda_1 \text{ and } \lambda_2 \text{ are large}, \]
\[ \lambda_1 \sim \lambda_2; \]

“flat” region
\[ \lambda_1 \text{ and } \lambda_2 \text{ are small}; \]

\[ f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} \]

Szeliski
Harmonic mean

\[ \det M = \lambda_1 \lambda_2 \]
\[ \text{trace } M = \lambda_1 + \lambda_2 \]
Harris Detector: Steps
Harris Detector: Steps

Compute corner response $R = \text{det}(M) - 0.06 \times \text{trace}(M)^2$
Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$ (not a good idea as you can see)
Harris Detector: Steps

Instead threshold but take only the points of local maxima of $R$ (non-max suppression)
Harris Detector: Steps
Next topic: Keypoint detection with scale selection

- We want to extract keypoints with characteristic scale that is covariant with the image transformation.
Blob filter

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]
Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*.

Blob detection

• Find maxima and minima of blob filter response in space and scale

Source: N. Snavely
Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
Scale-space blob detector: Example
Scale-space blob detector: Example

sigma = 11.9912
Scale-space blob detector

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

• Laplacian has strong response along edge
Efficient implementation

- Approximating the Laplacian with a difference of Gaussians:

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

(Laplacian)

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

(Difference of Gaussians)
Efficient implementation

SIFT keypoint detection

Questions?