CSE 527: Introduction to Computer Vision

Convolutions, Interest Points, Features
Recap
Today

More FT & convolutions

Laplacian Blending

Interest Points

Corner Detection

Harris Corner Detector

Human vision aspects

Descriptors: Teaser
The FT of a convolution is the pointwise product of FTs.

\[ \mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \]

\[ \mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\} \]
Example: Box Filter vs. Gaussian Filter

Why does the box filter have these artifacts? Maybe the FT can tell us.
Example: Box Filter vs. Gaussian Filter

Gaussian Filter, sigma = 3
Spectrum
Box filter, 5x5
Spectrum

Has a lot more high frequencies
Deconvolution

From the CT: If we know the convolution kernel, we can divide in the FD and reconstruct the original -- Cool feature!

We are given this very blurry input:

We know the kernel -- Can we recover the original??
Deconvolution

From the CT: If we know the convolution kernel, we can divide in the FD and reconstruct the original -- Cool feature!

We are given this very blurry input:

We know the kernel -- Can we recover the original??

Boom!
Deconvolution

Totally random kernel:

This is “cheating”, since we know the exact kernel. In reality we don’t know the kernel.
Deconvolution

Bokeh

Blury input:

Kernel:

Recovered:

[Köhler 2012]
Questions?
Image Pyramids

Half the resolution at each step.
Blur + Downsampling, to remove aliasing.

Useful for:
- Coarse to fine search
- Finding objects at different scales
- Blending

Most well-known example: **Laplacian Blending**

(but we will see another use for pyramids later)
Laplacian Pyramid

Blur + Downsample A: (low-pass)

Subtract original: (high-pass)

Combine HP of A+B:

Add combined to LP:
Laplacian Pyramid

Combine A+B halves
LB Implementation with OpenCV

# Gaussian pyramid
gpA = [G]
for i in xrange(6):
    G = cv2.pyrDown(G)  # Blur and downsample in one
    gpA.append(G)

# Laplacian pyramid
lpA = [gpA[5]]  # keep smallest Gaussian at bottom
for i in xrange(5,0,-1):
    GE = cv2.pyrUp(gpA[i])  # start going up
    L = cv2.subtract(gpA[i-1],GE)  # diff to get HPF
    lpA.append(L)

# combine A + B to a new pyramid
# ...

# reconstruct by adding HP to LP at each level.
Questions?
Interest Points
Matching Similar Views

Task: Mark a point on image A and the “same point” on image B. Which points did you choose and why?
Matching Similar Views

Key points should be invariant to: rotation, translation, intensity changes, scale… Some are more distinctive than other and less confusing.
Matching Similar Views

Fundamental task in computer vision: Find corresponding features (points, lines, etc.) in multiple images of the same scene:

Key Points should be:
- Repeatable
- Distinctive
- Invariant to ...
Example: Stereo

Small baseline → Small change in local intensity.
Example: Panorama

Wider baselines with rotation component and distortion.
Panorama Stitching

Given images with overlap: How do we stitch them together?
Panorama Stitching

Identify interest points in both images.

Hopefully we can find the “same point” in image A and image B.

Goal: Repeatability.
Panorama Stitching

Identify interest points in both images.

Hopefully the points we chose won’t be very confusing.

Goal: Distinctive.
Panorama Stitching

* Manual stitching
Why do we like image corners so much?

There are so many applications for them!

- Panorama
- 3D reconstruction
- Motion tracking and visual odometry
- Indexing and retrieval
- Object recognition
- ....

They roughly follow human vision’s edge acuity, and speculated models of human scene reconstruction.
In corners: A shift (w.r.t to a center) in any direction will result in a big change in intensity.
How to Detect a Corner?
Corner Detector

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

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

Sure, we can compute this, but it’s dreadfully slow… for each pixel we subtract $u \cdot v \cdot x \cdot y$ times

Taylor expansion: (just 1st order)

$$T(x, y) \approx f(a, b) + (x - a)f_x(a, b) + (y - b)f_y(a, b) + \ldots$$

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

Notation:

\[ I_x = \frac{\partial I(x, y)}{\partial x} \]

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

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

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

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

Now we have a quadratic approximation.
Corner Detector

\[ E(u, v) \approx (u, v)M \begin{pmatrix} x \\ y \end{pmatrix} \]

Each horizontal slice of a quadratic is an ellipse (or hyperbola, but not in our PD case).

If we take the Eigen factorization of \( M \) - the eigenvalues \( \lambda_1, \lambda_2 \) are the magnitudes the major axes of the ellipse:

The Autocorrelation Matrix:

\[
M = \begin{pmatrix}
\sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\
\sum_{x,y} I_x I_y & \sum_{x,y} I_y^2
\end{pmatrix}
\]
However taking the Eigen decomposition of M will take too long. Harris & Stephens came up with a fast approximation:

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

We can also precalculate the image derivatives \(I_x^2, I_y^2\) and \(I_{xy}\) as well as use a convolution with a kernel (Gaussian) to do the summation over the search windows.

\[
\text{Harris}(\hat{M}) = \det(\hat{M}) - \alpha \text{trace}(\hat{M})
\]

\[
= G(I_x^2)G(I_y^2) - G(I_x I_y)^2 - \alpha \left[ G(I_x^2) + G(I_y^2) \right]^2
\]
Corner Detector | Harris
**Invariance** and **Covariance** of Harris corners:

- Invariant to intensity changes (shift or scale), since we use derivatives
- Covariant w.r.t translation, since derivative and convolutions are shift-invariant
- Covariant w.r.t rotation, since ellipse eigenvalues will remain the same
- **Corners are not covariant w.r.t scaling (!!),** since the convolution has a single scale
Questions?
Human Vision Interest Points

We have two eyes, and perform an incredibly accurate 3D reconstruction from stereo vision in blazing fast speeds.

In this synthetic image all “natural” cues are removed, but we can easily see depth - are we extracting interest points and matching them?
Human Vision Interest Points

How do we deal with ambiguity in stereo?
Local Descriptors | Teaser
Wrap Up