INFO0013 Computer Vision

Linear Filtering
Justus H. Plater

Where are we at?

We know: how to recover stereo camera geometry (given internal camera parameters).
This involves: finding point correspondences.
We would like: to automate this – in particular, as we go from 2 to many images.
This requires some background. Here we go:

Overview

An image is characterized by
• intensity values
• the spatial distribution of these values
An analysis/quantification of these characteristics is fundamental to just about any higher-level vision application. How do we do this?

Linear filtering:
• Transformation of Intensity Distributions
  noise suppression, image enhancement, ...
• Feature Extraction
  edge detection, appearance characterization, ...

Introduction to Discrete Convolution
Example: Smoothing by Block Averaging

\[ R_{ij} = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F_{uv} \]

[Figure from Forsyth and Ponce 2003]

Another Way of Smoothing

[Figure from Forsyth and Ponce 2003]

Discrete Convolution

The second image was obtained by Gaussian-weighted averaging.

In general, if a filter kernel is represented by a matrix \( H \), the filtered result \( R \) of an image \( F \) is obtained by convolution:

\[ R_{ij} = \sum_{uv} H_{i-u,j-v} F_{uv} \]

[Figure from Forsyth and Ponce 2003]
**Shift-Invariant Linear System**

**Superposition:** 
\[ R(f + g) = R(f) + R(g) \]

**Scaling:** 
\[ R(kf) = kR(f) \]

**Shift Invariance:** 
The response to a translated stimulus is the translated response to the stimulus.

---

**Discrete Convolution in One Dimension**

Define: The \( j \)th element of vector \( \text{shift}(f, i) \) is \( f_{ji} \). Then,
\[ f = \sum_i f_i \text{shift}(f, i) \] where \( e_0 = \ldots 0, 0, 1, 0, 0, \ldots \)

Since \( R \) is linear and shift invariant, we have
\[ R(f) = R(\sum_i f_i \text{shift}(e_0, i)) = \sum_i f_i \text{shift}(R(e_0), i) \]

\( R(e_0) = g \) is the kernel or impulse response of \( R \), and
\[ R(f) = \sum_i f_i \text{shift}(g, i) = g * f \]

defines a discrete convolution, where \( R_j = \sum_i g_{ji} f_i \).

---

**Discrete Convolution in Two Dimensions**

For a 2D analog of \( e_0 \), we have
\[ R = G * F \]
\[ R_{ij} = \sum_{uv} G_{i-u,j-v} F_{uv} \]

The impulse response \( G \) of a 2D system is also called its point spread function.

---

**What To Do at the Image Boundaries?**

- **Ignore undefined pixels** – every convolution shrinks the image
- **Pad the image with constant values** – may create substantial gradients
- **Extend the image with its mirror image** – may create substantial second-order gradients
- **Extend the image by replicating its boundary pixels** – ditto (but often a simple and pragmatic solution)
Properties of Convolutions

- Convolutions are **symmetric** and **associative**:
  \[ g * h = h * g \]
  \[ (f * g) * h = f * (g * h) \]
- For fixed \( i, j \), a convolution \( R_{ij} = \sum_{u,v} G_{i-u,j-v} F_{uv} \) computes an **inner product** with a flipped version of \( G \) and thus acts as a **template** or a **feature detector**.

Representing Spatial Change

**Approach:** Change of basis: Instead of spatial intensities, represent **spatial frequencies**!

**Fourier Transform:** Represent a spatial signal as a weighted sum of sinusoids:

\[
F(g(x, y)(u, v)) = \iint_{-\infty}^{\infty} g(x, y) e^{-2\pi i (ux + vy)} \, dx \, dy
\]

\[
e^{-2\pi i (ux + vy)} = \cos(2\pi (ux + vy)) + i\sin(2\pi (ux + vy))
\]

For fixed \( u, v \): “inner product” of signal with sine ripples along \( ux + vy = \text{const} \) of orientation \( \tan\theta = \frac{v}{u} \), and frequency \( \sqrt{u^2 + v^2} \).

The Real Term of Some Fourier Bases

(Figure from Forsyth and Ponce 2003)
Real and Imaginary Parts of a Fourier Transform

More about the Fourier Transform

**Linearity:**
\[ F(g(x, y) + h(x, y)) = F(g(x, y)) + F(h(x, y)) \]
\[ F(kg(x, y)) = kF(g(x, y)) \]

**Inverse Fourier Transform:** symmetric, with the sign of the exponent reversed. Allows reconstruction of images from Fourier spectra!

**Convolution Theorem:** Convolution in the spatial domain \((f * g)(x, y)\) is multiplication in the Fourier domain \(F(f)F(g)(u, v)\), and vice versa.

**Magnitude and Phase:** A meaningful way of looking at the complex-valued function \(F(u, v)\).

Magnitude and Phase Spectra of a Fourier Transform

Zebra, Cheetah

(Figure from Forsyth and Ponce 2003)
Smoothing

Smoothing can be done
• in the frequency domain by multiplication with a box function
  \[ \text{Problem?} \]
• in the spatial domain by convolution with the inverse Fourier transform of the box function:
  \[ \frac{\sin x \sin y}{xy} \]
  \[ \text{Problem?} \]

Design of bandpass filters involves a trade-off between roll-off and ripple.
A good choice is often a Gaussian kernel.

Why Use a Gaussian For Smoothing?

• Because it’s intuitive.
• Relatively good band-pass characteristics and no ripples. Its Fourier transform is also a Gaussian:
  \[ \mathcal{F} \left( e^{-\frac{x^2}{2\sigma^2}} \right) = \sqrt{2\pi\sigma} e^{-2\pi^2\sigma^2} \]
  \[ \text{Axis-parallel Gaussians are separable:} \]
  \[ G_{\sigma}(x, y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \]
  \[ = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{y^2}{2\sigma^2}} \]

Thus, a 2D convolution is reduced to two 1D convolutions.

Why Use a Gaussian For Smoothing? (Continued)

• Gaussians (and their derivatives) have recursive implementations that are extremely efficient even for large values of \( \sigma \).
• Derivatives of Gaussians are steerable.
  \[ G_{\sigma_1} \ast G_{\sigma_2} = G_{\sqrt{\sigma_1^2 + \sigma_2^2}} \]

Repeated filtering with a small Gaussian can be more efficient than filtering with a large Gaussian, and leads naturally to a pyramid.
• The Central Limit Theorem: Repeated convolution with any kernel eventually yields Gaussian filtering.
• Gaussian filtering corresponds to a physical diffusion process:
Why Use a Gaussian For Smoothing? (Continued)

Designing a Discrete Filter Kernel

For our purposes: Simply sample the desired continuous kernel.

**Warning**

Choose a sufficiently large support!

Conclusions

Summary

What we’ve learned today:

- Linear filtering by convolution
- The Fourier transform
- A praise song to the Gaussian