Integral and Discrete Transforms in Image Processing

When standard convolution comes short ...

David Svoboda

email: svoboda@fi.muni.cz
Centre for Biomedical Image Analysis
Faculty of Informatics, Masaryk University, Brno, CZ

November 13, 2022

Outline

1. Linear Recursive Filters
   - Motivation
   - Filter Analysis
   - Filter Design
   - Applications
   - Conclusion
2. Steerable Filters
   - Motivation
   - Filter Design
   - Conclusion

Linear Recursive Filters

Motivation

Common properties of linear filters based on convolution

- defined via convolution kernel
- naive convolution complexity: $O(n^2)$
- FFT based convolution complexity: $O(n \log n)$

Idea of an improvement

- do not evaluate the convolution process separately for each pixel
- include the already convolved neighbouring values into the convolution at the next pixel
- complexity: $o(n \log n)$

Linear Recursive Filters

An example

Let be given a simple recursive filter:

$$g: \ h(n) = \alpha h(n-1) + (1-\alpha) f(n)$$

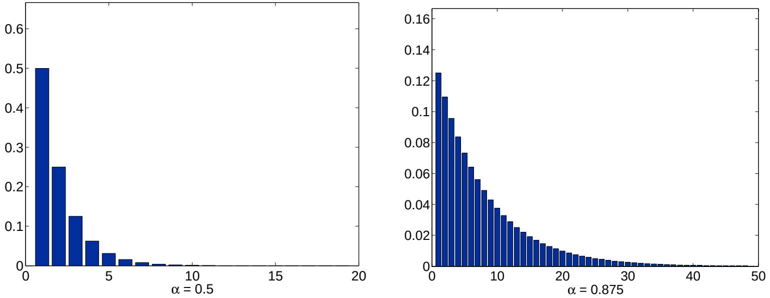
where α is a real constant, typically $\alpha \in \langle 0; 1 \rangle$.

- filter takes the fraction α from the previously calculated value
- filter works in certain direction
  - left to right – causal filter (this case)
  - right to left – anti-causal filter
  - both side – non-causal filter
- no convolution kernel is defined, recursion formula is used instead

Linear Recursive Filters

An example

Impulse response (PSF) for filter g: $h(n) = \alpha h(n-1) + (1-\alpha) f(n)$



Notice: PSF can be generated by passing a brief signal $f(n) = \delta(n) = [1, 0, 0, \dots]$ through the filter.

Question: What happens if $\alpha > 1$?

Impulse response (PSF)

- filter output when accepting a very brief signal (δ impulse)
- usually represented by convolution kernel ($g(n)$)
- expresses how the input signal is modified when passed through the filter

$$h(n) = f(n) * g(n)$$

(Optical) Transfer function

- Fourier transform of PSF ($\mathcal{G}(k)$)
- $\mathcal{G}(k) = \frac{\mathcal{H}(k)}{\mathcal{F}(k)}$

$$\mathcal{H}(k) = \mathcal{F}(k) \cdot \mathcal{G}(k)$$

Finite Impulse Response (FIR) filters

- defined via finite convolution kernel

$$g = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \Rightarrow h(k) = \sum_{i} f(k-i)g(i)$$

Infinite Impulse Response (IIR) filters

- defined via recursion formula

$$h(k) = \sum_{j=1}^{m} b_j h(k-j) + \sum_{i=0}^{n} a_i f(k-i)$$

Notice: Any recursive filter can be replaced by a nonrecursive filter (with a mask of infinite size). Its mask is given by the PSF of the recursive filter.

Linear Recursive Filters

Common properties

- causal – recursion formula uses only previously computed values

$$h(k) = \sum_{j=1}^{m} b_j h(k-j) + \sum_{i=0}^{n} a_i f(k-i)$$

- anti-causal – recursion formula goes from right to left

$$h(k) = \sum_{j=1}^{m} b_j h(k+j) + \sum_{i=0}^{n} a_i f(k-i)$$

- non-causal – filter "looks" both sides
- impulse response is infinite
  (we do not have to crop Gaussian hat when smoothing the image)
- recursive filters need not be stable in general
  (recursion may cumulate small errors)

Linear Recursive Filters

Tasks to solve

1. How to efficiently design any recursive filter?
2. How to guarantee its stability?
3. It is possible to design a recursive version of standard non-recursive filters like Gaussian, Sobel, Laplace, . . . ?

Notice: We can find answers for all the questions above, but we need to be familiar with *Z-transform*.

Filter Analysis

Given a general recursive filter:

$$\begin{aligned} h(n) &= a_0 f(n) + a_1 f(n-1) + a_2 f(n-2) + \cdots + \\ &\quad b_1 h(n-1) + b_2 h(n-2) + b_3 h(n-3) + \ldots \end{aligned}$$

where

- $f(n)$ ... input signal
- $h(n)$ ... output signal

Applying the substitution

$$\begin{array}{rrrr} a_0 = & 0.389 & & \\ a_1 = & -1.558 & b_1 = & 2.161 \\ a_2 = & 2.338 & b_2 = & -2.033 \\ a_3 = & -1.558 & b_3 = & 0.878 \\ a_4 = & 0.389 & b_4 = & -0.161 \end{array}$$

we get the following formula:

$$\begin{aligned} h(n) &= 0.389 f(n) - 1.558 f(n-1) + 2.338 f(n-2) - 1.558 f(n-3) + 0.389 f(n-4) + \\ &\quad 2.161 h(n-1) - 2.033 h(n-2) + 0.878 h(n-3) - 0.161 h(n-4) \end{aligned}$$

Filter Analysis

Let us apply Z-transform to the recursion formula

$$\begin{aligned} h(n) &= a_0 f(n) + a_1 f(n-1) + a_2 f(n-2) + \dots + \\ &\quad b_1 h(n-1) + b_2 h(n-2) + b_3 h(n-3) + \dots \\ &\quad /\mathrm{Z}\{.\}/ \\ \mathcal{H}(z) &= a_0 \mathcal{F}(z) + a_1 z^{-1} \mathcal{F}(z) + a_2 z^{-2} \mathcal{F}(z) + \dots + \\ &\quad b_1 z^{-1} \mathcal{H}(z) + b_2 z^{-2} \mathcal{H}(z) + b_3 z^{-3} \mathcal{H}(z) + \dots \end{aligned}$$

If we state $\mathcal{H}(z) = \mathcal{F}(z) \cdot \mathcal{G}(z)$ then:

$$\mathcal{G}(z) = \frac{\mathcal{H}(z)}{\mathcal{F}(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + \dots}{1 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3} - \dots}$$

Filter Analysis

Let us substitute the particular values:

$$\begin{aligned}
\mathcal{G}(z) &= \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + \dots}{1 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3} - \dots} \\
&= \frac{0.389 - 1.558z^{-1} + 2.338z^{-2} - 1.558z^{-3} + 0.389z^{-4}}{1 - 2.161z^{-1} + 2.033z^{-2} - 0.878z^{-3} + 0.161z^{-4}} \Big/ \frac{z^4}{z^4} \Big/ \\
&= \frac{0.389z^4 - 1.558z^3 + 2.338z^2 - 1.588z + 0.389}{z^4 - 2.161z^3 + 2.033z^2 - 0.878z + 0.161} \text{/factoring/} \\
&= \frac{(z - z_1)(z - z_2)(z - z_3)\dots}{(z - p_1)(z - p_2)(z - p_3)\dots}
\end{aligned}$$

- p_i ... poles of transfer function $\mathcal{G}$
- z_i ... zeros of transfer function $\mathcal{G}$

Filter Analysis

Transfer function properties:

- poles and zeros are complex numbers
- each pole must lie within the unit circle of the z-plane in order to guarantee filter stability, i.e. $|p_i| \leq 1$
- poles and zeros uniquely define the shape of transfer function $\mathcal{G}$
- factoring polynomials of higher degrees is non-trivial task

Filter Design

From scratch

- design a completely new filter with specific conditions
- rather complicated

Approximation of an existing filter (e.g. Gauss, Sobel, Laplace, ...)

- analytical approach – direct computation of recursive coefficients [Jin & Gao, 1997]
- numerical approach – search for recursive coefficients by iterative minimization [Deriche, 1987], [Young & Vliet, 1995]

Notice: We will focus on Jin's approach. We will try to a design recursive version of Gaussian filter.

The design consists of the following three steps:

1. guarantee of stability

   – specify the pole-zero placement in the z-plane;
   – the position of poles defines whether the filter converges or diverges.

$$\mathcal{G}(z) = \frac{(z - z_1)(z - z_2)(z - z_3)\ldots}{(z - p_1)(z - p_2)(z - p_3)\ldots}$$

2. guarantee of accuracy

   – design the filter transfer function;
   – Z-transform of a recursive filter is a rational function. The accuracy corresponds to the degree of both polynomials.

$$\mathcal{G}(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + \ldots}{1 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3} - \ldots}$$

3. compute the recursion coefficients of the filter

Task

Let be Gaussian filter

$$g_\sigma(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{n^2}{2\sigma^2}} = k \cdot \alpha^{n^2}$$

where $k = \frac{1}{\sigma\sqrt{2\pi}}$ and $\alpha = e^{-\frac{1}{2\sigma^2}}$ fixed terms. The Z-transform pair of $g_\sigma(n)$ is:

$$\mathcal{G}_\sigma(z) = k \sum_{n=-\infty}^{+\infty} \alpha^{n^2} z^{-n}$$

The task is to design a recursive version of $g_\sigma(n)$ and $\mathcal{G}_\sigma(z)$.

non-recursive version	**recursive version**
$\mathcal{G}_\sigma(z) = k \sum_{n=-\infty}^{+\infty} \alpha^{n^2} z^{-n}$	$\mathcal{J}_\sigma(z) = k \sum_{n=-\infty}^{+\infty} ?$

Notice: Without lost of generality, let us split bilateral sequence $\mathcal{J}_\sigma(z)$ into two unilateral sequences (causal & anti-causal):

$$\mathcal{J}_\sigma(z) = \mathcal{J}_\sigma^+(z) + \mathcal{J}_\sigma^-(z),$$

i.e.

$$\mathcal{J}_\sigma^+(z) = k \sum_{n=0}^{+\infty} ? \quad \text{and} \quad \mathcal{J}_\sigma^-(z) = k \sum_{n=-\infty}^{0} ?$$

How to design $\mathcal{J}_\sigma^+(z)$, which should be a transfer function, i.e. rational function?

1. Let us use the second and third order polynomial. We get

$$\mathcal{J}_\sigma^+(z) = k\frac{1 + a_1 z^{-1} + a_2 z^{-2}}{(1 - pz^{-1})^3}$$

   The denominator has a unique pole of order 3 which should guarantee ($|p| \leq 1$) filter stability.

2. Using polynomial division the function $\mathcal{J}_\sigma^+(z)$ can be simply converted into infinite series in power of z:

$$\begin{aligned}
\mathcal{J}_\sigma^+(z) &= k\frac{1 + a_1 z^{-1} + a_2 z^{-2}}{(1 - pz^{-1})^3} \; / \cdot \frac{z^3}{z^3} / \\
&= k\frac{z^3 + a_1 z^2 + a_2 z}{z^3 - 3pz^2 + 3p^2 z + p^3} \; /\text{polynomial division}/ \\
&= k[z^0 + (3p + a_1)z^{-1} + (6p^2 + 3a_1 p + a_2)z^{-2} + (6a_1 p^2 + 3a_2 p + 10p^2)z^{-3} + \ldots]
\end{aligned}$$

$$\mathcal{J}_\sigma^+(z) = k[z^0 + (3p+a_1)z^{-1} + (6p^2+3a_1p+a_2)z^{-2} + (6a_1p^2+3a_2p+10p^2)z^{-3} + \ldots]$$

$$\mathcal{G}_\sigma(z) = k\sum_{n=-\infty}^{+\infty} \alpha^{n^2} z^{-n} = k[\cdots + \alpha z^{-1} + \alpha^4 z^{-2} + \alpha^9 z^{-3} + \ldots]$$

3. Comparing z-coefficients between $\mathcal{J}_\sigma^+(z)$ and $\mathcal{G}_\sigma(z)$

$$\begin{aligned} \mathbf{z}^{-1}: &\quad 3p + a_1 = \alpha \\ \mathbf{z}^{-2}: &\quad 6p^2 + 3a_1p + a_2 = \alpha^4 \\ \mathbf{z}^{-3}: &\quad 6a_1p^2 + 3a_2p + 10p^2 = \alpha^9 \end{aligned}$$

we finally get:

$$p = \frac{\alpha}{2}\left(3 - \alpha^2 - \sqrt{9 - 6\alpha^2 - 3\alpha^4}\right),$$

where $\alpha = e^{-\frac{1}{2\sigma^2}}$.

Solution – Causal Part

$$
\begin{aligned}
\mathcal{J}_\sigma^+(z) &= k\frac{1+a_1z^{-1}+a_2z^{-2}}{(1-pz^{-1})^3} = k\frac{1+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}} \\
&\Downarrow \quad \text{/inverted Z-transform/} \\
h_+(n) &= k\{f(n)+a_1f(n-1)+a_2f(n-2)\} \\
&\quad -\{b_1h_+(n-1)+b_2h_+(n-2)+b_3h_+(n-3)\}
\end{aligned}
$$

where

$$
\begin{aligned}
b_1 &= -3p \\
b_2 &= 3p^2 \\
b_3 &= -p^3 \\
a_1 &= \alpha - 3p \\
a_2 &= \alpha^4 - 3\alpha p + 3p^2
\end{aligned}
$$

Filter Design

Jin's Approach

When dealing with (anti)symmetrical filters, it is unnecessary to apply two times the design procedure. We can simply mirror the causal part and eliminate the central point to avoid counting it twice.

Solution – Anti-Causal Part

$$
\begin{aligned}
\mathcal{J}_\sigma^-(z) &= k\left[\frac{1 + a_1 z^{+1} + a_2 z^{+2}}{1 + b_1 z^{+1} + b_2 z^{+2} + b_3 z^{+3}} - 1\right] \\
&= k\left[\frac{1 + a_1 z^{+1} + a_2 z^{+2} - (1 + b_1 z^{+1} + b_2 z^{+2} + b_3 z^{+3})}{1 + b_1 z^{+1} + b_2 z^{+2} + b_3 z^{+3}}\right] \\
&= k\left[\frac{(a_1 - b_1) z^{+1} + (a_2 - b_2) z^{+2} - b_3 z^{+3}}{1 + b_1 z^{+1} + b_2 z^{+2} + b_3 z^{+3}}\right] \qquad /a_3 = a_1 - b_1, \ldots / \\
&= k\,\frac{a_3 z^{+1} + a_4 z^{+2} + a_5 z^{+3}}{1 + b_1 z^{+1} + b_2 z^{+2} + b_3 z^{+3}}
\end{aligned}
$$

Solution – Anti-Causal Part

$$
\begin{aligned}
\mathcal{J}_\sigma^-(z) &= k\frac{a_3 z + a_4 z^2 + a_5 z^3}{1 + b_1 z + b_2 z^2 + b_3 z^3} \\
&\Downarrow \quad /\text{inverted Z-transform}/ \\
h_-(n) &= k\{a_3 f(n+1) + a_4 f(n+2) + a_5 f(n+3)\} \\
&\quad -\{b_1 h_-(n+1) + b_2 h_-(n+2) + b_3 h_-(n+3)\}
\end{aligned}
$$

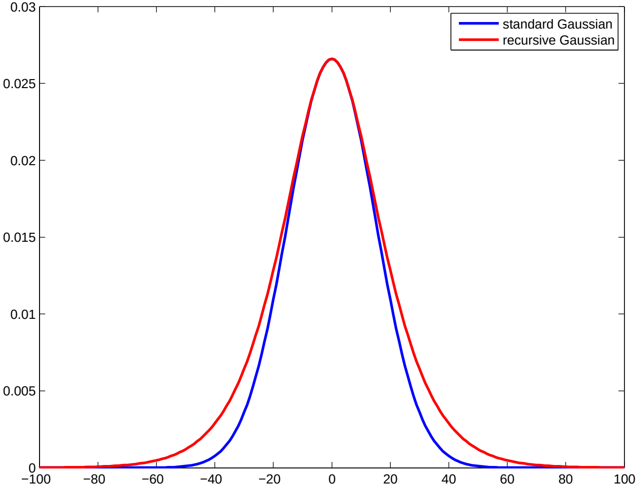
where

$$
\begin{aligned}
b_1 &= -3p \\
b_2 &= 3p^2 \\
b_3 &= -p^3 \\
a_3 &= a_1 - b_1 \\
a_4 &= a_2 - b_2 \\
a_5 &= -b_3
\end{aligned}
$$

Final Solution

$$
\begin{aligned}
h_{+}(n) &= k\{f(n) + a_1 f(n-1) + a_2 f(n-2)\} \\
&\quad -\{b_1 h_{+}(n-1) + b_2 h_{+}(n-2) + b_3 h_{+}(n-3)\} \\
h_{-}(n) &= k\{a_3 f(n+1) + a_4 f(n+2) + a_5 f(n+3)\} \\
&\quad -\{b_1 h_{-}(n+1) + b_2 h_{-}(n+2) + b_3 h_{-}(n+3)\} \\
h(n) &= h_{+}(n) + h_{-}(n)
\end{aligned}
$$

Filter Design
Jin's Approach



A uniform averaging filter

$$h(k) = \sum_{i=0}^{n-1} f(k-i)$$

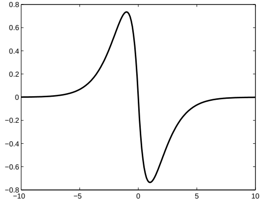
Its computational complexity depends on the width n. The same filter can be written in the recursive form:

$$h(k) = h(k-1) + f(k) - f(k-n)$$

Exercise: Show (using Z-transform) that it is formally equivalent!

- Deriche used essentially the same reasoning as Canny with one exception.
- While Canny sought an optimal filter of finite width W, Deriche derived an optimal filter of infinite width using the same optimality criteria as Canny.
- The solution is

$$g(x) \approx -cxe^{-\alpha|x|}$$



- Let h^+ and h^- denote two 1D arrays. Deriche computes the 1D gradient along one row using this recursive form:

$$\begin{aligned} h^+(m) &= f(m-1) - b_1 h^+(m-1) + b_2 h^+(m-2) \\ h^-(m) &= f(m+1) - b_1 h^+(m+1) + b_2 h^-(m+2) \\ |\nabla f(m)| &= -ce^{-\alpha}(h^+(m) + h^-(m)) \end{aligned}$$

with

$$b_1 = -2e^{-\alpha} \quad \text{and} \quad b_2 = e^{-2\alpha}$$

- The computational load is much smaller than that of the Canny filter.
- The computational time is independent of the size of the smoothing parameter α.

Horizontal Edge Map

$$
\begin{aligned}
g_{v1}(x,y) &= f(x,y-1) - b_1 g_{v1}(x,y-1) - b_2 g_{v1}(x,y-2) \\
g_{v2}(x,y) &= f(x,y+1) - b_1 g_{v2}(x,y+1) - b_2 g_{v2}(x,y+2) \\
g_{hv}(x,y) &= a(g_{v1}(x,y) - g_{v2}(x,y)) \\
g_{h1}(x,y) &= a_0 g_{hv}(x,y) + a_1 g_{hv}(x-1,y) - b_1 g_{h1}(x-1,y) \\
&\quad - b_2 g_{h1}(x-2,y) \\
g_{h2}(x,y) &= a_2 g_{hv}(x+1,y) + a_3 g_{hv}(x+2,y) - b_1 g_{h2}(x+1,y) \\
&\quad - b_2 g_{h2}(x+2,y)
\end{aligned}
$$

$$
V(x,y) = g_{h1}(x,y) + g_{h2}(x,y)
$$

Vertical Edge Map

$$
\begin{aligned}
g_{v1}(x,y) &= f(x-1,y) - b_1 g_{v1}(x-1,y) - b_2 g_{v1}(x-2,y) \\
g_{v2}(x,y) &= f(x+1,y) - b_1 g_{v2}(x+1,y) - b_2 g_{v2}(x+2,y) \\
g_{hv}(x,y) &= a(g_{v1}(x,y) - g_{v2}(x,y)) \\
g_{h1}(x,y) &= a_0 g_{hv}(x,y) + a_1 g_{hv}(x,y-1) - b_1 g_{h1}(x,y-1) \\
&\quad - b_2 g_{h1}(x,y-2) \\
g_{h2}(x,y) &= a_2 g_{hv}(x,y+1) + a_3 g_{hv}(x,y+2) - b_1 g_{h2}(x,y+1) \\
&\quad - b_2 g_{h2}(x,y+2)
\end{aligned}
$$

$$
H(x,y) = g_{h1}(x,y) + g_{h2}(x,y)
$$

Final solution:

$$|\nabla f(x,y)| = \sqrt{H(x,y)^2 + V(x,y)^2}$$

Where the constants in use are:

$$\begin{aligned}
a &= -(1-e^{-\alpha})^2 \\
b_1 &= -2e^{-\alpha} \\
b_2 &= e^{-2\alpha} \\
a_0 &= \frac{-a}{1-\alpha b_1 - b_2} \\
a_1 &= a_0(\alpha-1)e^{-\alpha} \\
a_2 &= a_1 - a_0 b_1 \\
a_3 &= -a_0 b_2
\end{aligned}$$

and α is the only one parameter.

Conclusion

When designing recursive filter one meets the following tasks:

- replication – given slow (but nice) non-recursive filter, how to design its recursive counterpart
- stability – whether the new filter diverges (poles $|p_i| > 1$) or converges (poles $|p_i| \leq 1$)
- accuracy
  - polynomial degree
  - numerical method error

- non-recursive filter PSF $g(n)$ with its Z-transform transfer function:

$$\mathcal{G}(z) = \sum_{i=0}^{\infty} g(i) z^{-i}$$

- we want to design recursive filter defined using its transfer function

$$\overline{\mathcal{G}}(z) = \sum_{i=0}^{\infty} \overline{g}(i) z^{-i} = \frac{\sum_{i=0}^{n} a_i z^{-i}}{1 - \sum_{j=1}^{m} b_j z^{-j}}$$

Given a simple recursive filter:

$$h(n) = \alpha h(n-1) + f(n)$$

→ Z-transform (applied on both sides of the equation):

$$\mathcal{H}(z) = \frac{1}{1-\alpha z^{-1}} \mathcal{F}(z)$$

→ Z-transform based transfer function:

$$\mathcal{G}(z) = \frac{z}{z-\alpha}$$

Let us analyze the problem:

- $\mathcal{G}(z)$ has one pole at $z = \alpha$
- checking the filter against δ impulse $f(n) = [1, 0, 0, 0, \ldots]$ we get $h(n) = 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \ldots$
- for $|\alpha| < 1$ the filter is stable (series converges)
- for $|\alpha| > 1$ the filter is unstable (series diverges)

Conclusion
Accuracy

Given

$$\overline{\mathcal{G}}(z) = \frac{\sum_{i=0}^{n} a_i z^{-i}}{1 - \sum_{j=1}^{m} b_j z^{-j}}$$

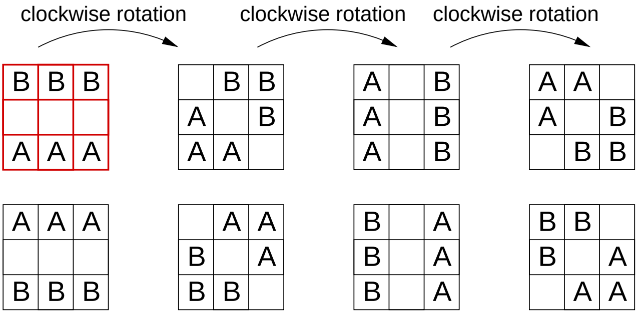
we search for a_i and b_i:

- directly – analytical approach (see the example)
- iteratively – numerical minimization:

$$E = \oint_c \left|\mathcal{G}(z) - \overline{\mathcal{G}}(z)\right|^2 \frac{dz}{2\pi i z} = /\text{energy theorem}/ = \sum_n |g(n) - \overline{g(n)}|^2$$

Let us recall template-based edge detection:

- The specified filter is rotated and applied *n*-times
- We perform *n* convolutions
- Each subsequent convolution uses kernel rotated by $n/360$ degrees.
- Can we decrease the task complexity?

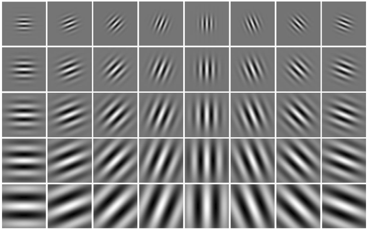


Gabor filters

$$Gabor(x, y) = Gauss_{\sigma}(x, y) \cdot FourierBasis_{\omega}^{\theta}(x, y)$$

where

- ω ... speed of waving
- θ ... orientation of the filter
- σ ... width of Gaussian envelope



The use of Gabor filters

- optical flow detection
- feature extraction
- ...

How to optimize their computation?

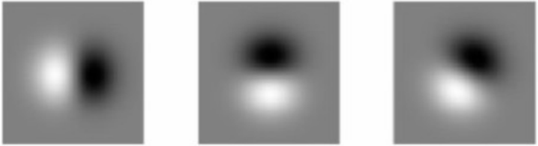
Steerable Filters

Definition

A steerable filter $f^{\theta}(x, y)$ is an orientation-selective convolution kernel used for image enhancement and feature extraction that can be expressed via a linear combination of a small set of rotated versions of itself:

$$f^{\theta}(x, y) = \sum_{j=1}^{M} k_j(\theta) f^{\theta_j}(x, y)$$

where $f^{\theta_j}(x, y)$ are called *basis functions* and $k_j(\theta)$ are *interpolation functions*.



Notice: We wish the value of M to be the lowest possible.

Steerable Filters

Example (1st derivative)

Task

We are looking for arbitrary oriented 1^{st} derivative of Gaussian $G_1^{\theta}(x, y)$.

Consider simple 2D Gaussian function G:

$$G(x, y) = e^{-\left(x^2+y^2\right)}$$

Let us perform the two first-order axis-oriented derivatives:

$$\begin{aligned} G_1^{0^\circ}(x, y) &= \frac{\partial}{\partial x} e^{-\left(x^2+y^2\right)} = -2xe^{-\left(x^2+y^2\right)} \\ G_1^{90^\circ}(x, y) &= \frac{\partial}{\partial y} e^{-\left(x^2+y^2\right)} = -2ye^{-\left(x^2+y^2\right)} \end{aligned}$$

- supscript . . . orientation of derivative
- subscript . . . derivative order

Steerable Filters

Example (1st derivative – cont'd)

The first derivative of Gaussian G at any arbitrary orientation θ can be expressed as:

$$G_1^{\theta}(x,y) = \cos(\theta) G_1^{0^\circ}(x,y) + \sin(\theta) G_1^{90^\circ}(x,y)$$

- $G_1^{0^\circ}$ and $G_1^{90^\circ}$ are called *basis functions*

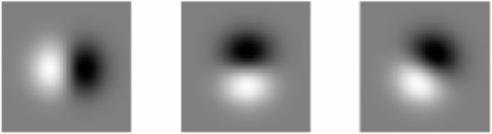
Detection of edges in image I at any orientation can be obtained by:

$$\begin{aligned} R_1^{0^\circ} &= G_1^{0^\circ} * I \\ R_1^{90^\circ} &= G_1^{90^\circ} * I \\ R_1^{\theta} &= \cos(\theta) R_1^{0^\circ} + \sin(\theta) R_1^{90^\circ} \end{aligned}$$

Notice: A whole family of filters can be evaluated with very little cost by first convolving the image with basis functions.

Steerable Filters

Example (1st derivative – cont'd)



$$G_1^{60^\circ} = \frac{1}{2} G_1^{0^\circ}(x, y) + \frac{\sqrt{3}}{2} G_1^{90^\circ}(x, y)$$

$$R_1^{60^\circ} = \frac{1}{2} R_1^{0^\circ}(x, y) + \frac{\sqrt{3}}{2} R_1^{90^\circ}(x, y)$$

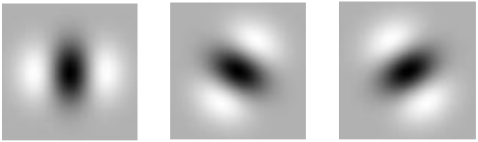
Steerable Filters

Example (2nd derivative)

Task

We are looking for arbitrary oriented 2^{nd} derivative of Gaussian $G_2^{\theta}(x, y)$.

2^{nd} derivative of Gaussian ($\approx$ Laplacian): $G_2^{0^\circ}(x, y) = (4x^2 - 2)e^{-\left(x^2+y^2\right)}$



$G_2^{0^\circ}(x, y)$ $\quad G_2^{60^\circ}(x, y)$ $\quad G_2^{120^\circ}(x, y)$

$$G_2^{\theta}(x, y) = k_1(\theta) G_2^{0^\circ}(x, y) + k_2(\theta) G_2^{60^\circ}(x, y) + k_3(\theta) G_2^{120^\circ}(x, y)$$

where

$$k_j(\theta) = \frac{1}{3}\left[1 + 2\cos\left(2\left(\theta - \theta_j\right)\right)\right]$$

Task

Given a function $f(x, y)$ we wish to derive its steerable version when using the least possible number of basis functions.

1. Assume $f(x, y) = W(r)P_N(x, y)$ $/r = \sqrt{x^2 + y^2}/$
   - $W(r)$ ... an arbitrary windowing function (e.g. Gaussian, Hamming)
   - $P_N(x, y)$ ... N^{th} order polynomial in x and y
2. Function $f(x, y)$ rotated to any angle can be synthesized as a linear combination of $2N + 1$ basis functions
   - $P_N(x, y)$ contains only even/odd order terms $\rightarrow$ $N + 1$ basis function are sufficient for synthesis.

3. The interpolation functions $k_j(\theta)$ must hold the following:

$$\begin{pmatrix} 1 \\ e^{i\theta} \\ \vdots \\ e^{iN\theta} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ e^{i\theta_1} & e^{i\theta_2} & \dots & e^{i\theta_M} \\ \vdots & \vdots & \ddots & \vdots \\ e^{iN\theta_1} & e^{iN\theta_2} & \dots & e^{iN\theta_M} \end{pmatrix} \begin{pmatrix} k_1(\theta) \\ k_2(\theta) \\ \vdots \\ k_M(\theta) \end{pmatrix}$$

   Use only the lines corresponding to the degree of non-zero coefficients from $P_N(x, y)$

4. Solve the above system. For reasons of symmetry and robustness against noise, the angles are equally sampled in the range 0 to π.
5. $f^\theta(x, y) = \sum_{j=1}^{M} k_j(\theta) f^{\theta_j}(x, y)$, where $\theta = \{0, \frac{\pi}{M}, \frac{2\pi}{M}, \dots, \frac{(M-1)\pi}{M}\}$

Filter Design

Example

Task

Assume we want to make the 1^{st} order derivative of 2D Gaussian steerable:

1. $G_1^{0^\circ}(x,y) = -2xe^{-\left(x^2+y^2\right)}$
   - $W(r) = e^{-\left(x^2+y^2\right)}$ ... windowing function
   - $P_N(x,y) = -2x$ ... first order odd polynomial
2. $N = 1 \rightarrow$ we need $2(= N+1)$ basis functions
3. Use only the complex exponential constraints corresponding to the degree of non-zero coefficients from $P_N(x,y)$

$$\left(\begin{array}{c} e^{i\theta} \end{array} \right) = \left(\begin{array}{cc} e^{i\theta_1} & e^{i\theta_2} \end{array} \right) \left(\begin{array}{c} k_1(\theta) \\ k_2(\theta) \end{array} \right)$$

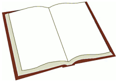
4. Solving the system pro particular values $\theta_1 = 0^\circ$ and $\theta_2 = 90^\circ$ we obtain:
   - $k_1(\theta) = \cos(\theta)$
   - $k_2(\theta) = \sin(\theta)$
5. $G_1^{\theta}(x,y) = \cos(\theta) G_1^{0^\circ}(x,y) + \sin(\theta) G_1^{90^\circ}(x,y)$

Conclusion

- All functions that are bandlimited in angular frequency, are steerable, given enough basis functions.
- The most useful functions require small number of basis functions.

Bibliography

- Smith S. W. "The Scientist and Engineer's Guide to Digital Signal Processing, 1998, www.DSPguide.com
- Jin, J. S., Gao Y. Recursive Implementation of LoG Filtering, Real-Time Imaging 3, 59-65 (1997)
- Deriche R. Separable recursive filtering for efficient multi-scale edge detection, in Proc. Int. Workshop Machine Vision and Machine Intelligence, Tokyo, Japan, 1987, pp. 18-23
- Young I. T., Vliet L. J. van. Recursive implementation of the Gaussian filter. Signal Process. 44, 2 (Jun. 1995), 139-151
- Freeman W. T., Adelson E. H. "The Design and Use of Steerable Filters," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 9, pp. 891-906, September, 1991



You should know the answers . . .

- Check, how the filter g: $h(n) = \alpha h(n-1) + (1-\alpha) f(n)$ behaves for $\alpha \in \{0, 0.5, 1, 1.5\}$.
- Describe the difference between the transfer function of FIR and IIR filters.
- What is the direction of computation of causal filters?
- How do we check the stability of an existing filter?
- Prove that $h(k) = \sum_{i=0}^{n-1} f(k-i)$ is equal to $h(k) = h(k-1) + f(k) - f(k-n)$
- What is the time-complexity of recursive filters (compared to standard FIR filters)?
- How do the steerable filters speed up the computation?
- Show, how to make the steerable version of the first derivative of 2D Gaussian.