Gaussian derivatives
UCU Winter School 2017

James Pritts
Czech Technical University

January 16, 2017

---

\(^1\)Images taken from Noah Snavely’s and Robert Collins’s course notes
Definition
An *image* (grayscale) is a function, $I$, from $\mathbb{R}^2$ to $\mathbb{R}$ such that $I(x, y)$ gives the intensity at position $(x, y)$.

Definition
A *digital image* (grayscale) is a sampled and quantized version of $I$. The discrete values are indexed as $I[x, y]$. 
Why do we need image derivatives?

Image derivatives will be used to construct discrete operators that detect salient differential geometry in the scene.

Some desiderata

- sparse representation of the image
- repeatability
- salient features
- invariance to photometric and geometric transforms of the image

Examples include

- Harris corners
- Hessian Affine (blobs)
- Maximally Stable Extremal Regions
Finite forward difference

- Taylor series expansion

\[ I(x+h) = I(x) + hI_x(x) + \frac{1}{2} h^2 I_{xx}(x) + \frac{1}{3!} h^3 I_{xxx}(x) + O(h^4) \implies \]

\[ \frac{I(x+h) - I(x)}{h} = I_x(x) + O(h) \implies \]

\[ I_x[x] \approx \frac{I[x + h] - I[x]}{h} \]

- Template

\[
\begin{array}{|c|c|}
\hline
-1 & 1 \\
\hline
\end{array}
\]
Finite backward difference

- Taylor series expansion

\[ I(x - h) = I(x) - hl_x(x) + \frac{1}{2} h^2 l_{xx}(x) - \frac{1}{3!} h^3 l_{xxx}(x) + \mathcal{O}(h^4) \quad \Rightarrow \]

\[ \frac{I(x) - I(x - h)}{h} = l_x(x) + \mathcal{O}(h) \quad \Rightarrow \]

\[ l_{x}[x] \approx \frac{I[x] - I[x - h]}{h} \]

- Template

\[
\begin{array}{|c|c|}
-1 & 1 \\
\end{array}
\]
Central difference

- Taylor series expansion

\[
I(x + h) = I(x) + hl_x(x) + \frac{1}{2}h^2l_{xx}(x) + \frac{1}{3!}h^3l_{xxx}(x) + O(h^4)
\]

\[
I(x - h) = I(x) - hl_x(x) + \frac{1}{2}h^2l_{xx}(x) - \frac{1}{3!}h^3l_{xxx}(x) + O(h^4)
\]

\[
I(x + h) - I(x - h) = I(x) + 2hl_x(x) + \frac{2}{3!}h^3l_{xxx}(x) \implies
\]

\[
\frac{I(x + h) - I(x - h)}{2h} = l_x(x) + O(h^2) \implies
\]

\[
l_x[x] \approx \frac{I[x + h] - I[x - h]}{2h}
\]

- Template

\[
\begin{array}{c|c|c}
-\frac{1}{2} & 0 & \frac{1}{2}
\end{array}
\]
Convolution in spatial domain

$$(I \ast f)[x, y] = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I[i,j] \ast f[x-i][y-j]$$

- convolution is equivalent to flipping the filter in both dimensions and correlating
- same result for symmetric kernels
- many libraries conflate convolution and correlation
- so why NOT just use cross-correlation?
Applying derivative to an image

- row \( i \) forward difference

\[
\text{for } (j = j_{\text{start}}; j <= j_{\text{end}}; j++)
\quad h[j] = I[j+1] - I[j];
\]

let \( f[0] = 1 \) and \( f[1] = -1 \), then

- row \( i \) Forward difference with mask \( f[0] = 1 \) and \( f[1] = -1 \).

\[
\text{for } (j = j_{\text{start}}; j <= j_{\text{end}}; j++)
\quad h[j] = f[0] \times I[j+1] + f[1] \times I[j];
\]

- row \( i \) with arbitrary derivative defined by mask \( f \), e.g, \( f[-1] = \frac{1}{2}, f[0] = 0, f[1] = -\frac{1}{2} \)

\[
\text{for } (j = j_{\text{start}}; j <= j_{\text{end}}; j++)
\{
\quad h[j] = 0;
\quad \text{for } (b = b_{\text{start}}; b <= b_{\text{end}}; b++)
\quad \quad h[j] += I[b] \times f[j-b];
\}
\]
Image derivatives are convolutions

- row $i$ with arbitrary derivative defined by mask $f$.

```
for (i = istart; i <= iend; i++)
    for (j = jstart; j <= jend; j++)
    {
        h[i][j] = 0;
        for (a = astart; a <= aend; a++)
            for (b = bstart; b <= bend; b++)
                h[i][j] += I[a][b]*f[i-a][j-b];
    }
```

- Loops over $a, b$ is discrete convolution at $i, j$.

$$h[i,j] := (l * f)[i,j] \equiv \sum_a \sum_b l[a,b]f[i-a,j-b]$$
Image Gradient

\[ \|
\n\n\| = \sqrt{\left( \frac{\partial I}{\partial x} \right)^2 + \left( \frac{\partial I}{\partial y} \right)^2} \]
Artificially added white noise

\[ \sim \mathcal{N}(\mu = 0, \sigma = 16) \]
Effect of white noise image derivatives

$\frac{d}{dx} I[x, y = j]$
Why do we use the Gaussian as a low-pass filter?

- white noise exists at all frequencies
- corners and edges are represented by high frequencies
- need to remove noise but maintain details
- Fourier transform of Gaussian is Gaussian: Gaussian does not have a sharp cutoff at some pass band frequency and does not oscillate.
- weighted averaging is spatial blurring is low-pass filtering.
- we will blur in the spatial domain with Gaussian

\[ \sigma = 0 \quad \sigma = 1 \quad \sigma = 5 \quad \sigma = 10 \quad \sigma = 30 \]
Box-filter vs Gaussian

- Box-filter garbles high-frequency signal while removing noise.
  (not doing good things in the frequency domain)
Gaussian kernel

\[ g_\sigma[x, y] = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

\[ \sigma = 1 \quad \sigma = 5 \quad \sigma = 10 \quad \sigma = 30 \]
Gaussian smoothing

\[ I \ast g \]

\[ \frac{d}{dx} (I \ast g) \]
Separability of Gaussian kernels

Definition

A 2D kernel $g$ is called separable if it can be broken down into the convolution of two kernels: $g = g^{(1)} * g^{(2)}$.

$$g_{\sigma}[x, y] = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{y^2}{2\sigma^2}}$$

$$= g^{(1)}_{\sigma}[x] \cdot g^{(2)}_{\sigma}[y]$$

and

$$(l * g_{\sigma})[x, y] = \sum_i \sum_j g_{\sigma}[x - i, x - j] l[i, j] = \ldots$$

$$\sum_i \sum_j g^{(1)}_{\sigma}[x - i] g^{(2)}_{\sigma}[x - j] l[i, j] = \sum_i g^{(1)}_{\sigma}[x - i] \sum_j g^{(2)}_{\sigma}[x - j] l[i, j] = \ldots$$

$$(g^{(1)}_{\sigma} * (g^{(2)}_{\sigma} * l))[x, y]$$
Complexity of convolution in spatial domain

What are the number of operations and complexity of \( k \times k \)-dimension kernel on and \( m \times n \)-dimension image for a non-separable kernel: \( k^2 \cdot m \cdot n \) operations, complexity \( \mathcal{O}(k^2) \).

- non-separable kernel: \( k^2 \cdot m \cdot n \) operations, complexity \( \mathcal{O}(k^2) \)
- separable kernel: \( 2 \cdot k \cdot m \cdot n \) operations, complexity \( \mathcal{O}(k) \)

It pays to take advantage of separability!
Important Gaussian derivative properties

- Image differentiation $\frac{d}{dx}$ is a convolution on image $I$.
- Smoothing by Gaussian kernel $g$ is a convolution on image $I$.
- 2D Gaussian kernel is separable $g = g^{(1)} * g^{(2)}$.
- Convolution is
  - commutative $I * g = g * f$
  - associative $(l * g) * h = l * (g * h)$

So $\frac{d}{dx}(l * g) = l * \frac{d}{dx}g = (l * (\frac{d}{dx}g^{(1)})) * g^{(2)}$
First Derivatives of a Gaussian

\[ \frac{d}{dx} g \]

\[ \frac{d}{dy} g \]
Gaussian derivatives like a boss.

If you want to level up, then you can exploit a recurrence relation of Hermite polynomials to algorithmically construct Gaussian derivatives of any order without convolution or symbolic differentiation.

\[ H_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{-\frac{x^2}{2}} = \left(x - \frac{d}{dx}\right)^n \cdot 1, \]