Computer Vision: Models, Learning and Inference
–
Image Filters

Oren Freifeld and **Meitar Ronen**

Computer Science, Ben-Gurion University

December 3, 2019

Examples of using Image Filters

1. Image processing (e.g., enhancement, sharpening, smoothing, etc.)
2. Spatial derivatives (*e.g.*, in optical flow)
3. One way to define high-order clique functions in MRFs (*e.g.*: Field of Experts; Steerable Random Filters)
4. Modeling image degradation (*e.g.*, blurring or motion artifacts)
5. Convolutional Neural Networks
6. Object Detection via template matching (AKA "Match Filter")

I: an image; h: a filter (another, usually-smaller) image

Definition (The correlation operator)

The **correlation operator**, $\otimes$, creates a new image, $I \otimes h$, via

$$(I \otimes h)(i,j) = \sum_{k,l} I(i+k, j+l)h(k,l) \overset{\substack{k'=i+k \\ l'=j+l}}{=} \sum_{k',l'} I(k',l')h(k'-i, l'-j)$$

Definition (The convolution operator)

The **convolution operator**, $*$, creates a new image, $I * h$, via

$$(I * h)(i,j) = \sum_{k,l} I(i-k, j-l)h(k,l) \overset{\substack{k'=i-k \\ l'=j-l}}{=} \underbrace{\sum_{k',l'} I(k',l')h(i-k', j-l')}_{(I \otimes h_{\text{flipped}})(i,j)}$$

$$= (h * I)(i,j)$$

Remarks

- Since $I * h = I \otimes h_{\text{flipped}}$, both operators coincide when the filter is symmetric around each of its axes.

Example

$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \Rightarrow (I * h)(i,j) = (I \otimes h)(i,j)$ is the average value of I in a 3×3 neighborhood around pixel (i,j):

$$\frac{1}{9}\left[I(i-1,j-1) + I(i-1,j) + I(i-1,j+1) + I(i,j-1)\right.$$
$$\left. + I(i,j) + I(i,j+1) + I(i+1,j-1) + I(i+1,j) + I(i+1,j+1)\right]$$

Another Example

The filter below is symmetric so convolution=correlation.

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

$*$

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

$=$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

Figure taken from Szelisk's book, 2010

Remarks

- There are standard ways to handle pixels near the image boundaries (*e.g.*, set "outside pixels" to zero)
- Internally, efficient implementations usually use the Fast Fourier Transform (FFT) instead of the formulas we saw in the definitions.
- In continuous domains, we have similar definitions, with integrals instead of sums.

Exercise

Show that convolution is linear. In effect, show that if I_1 and I_2 are 2D digital arrays, and $c_1, c_2 \in \mathbb{R}$, then

$$(c_1 I_1 + c_2 I_2) * h = c_1 (I_1 * h) + c_2 (I_2 * h) \tag{1}$$

Solution Let $g_1 = I_1 * h$, $g_2 = I_2 * h$, and $g_3 = (c_1 I_1 + c_2 I_2) * h$.

$$\begin{aligned}
&g_3(i,j) = \\
&\sum_{k,l} (c_1 I_1(i-k, j-l) + c_2 I_2(i-k, j-l)) h(k,l) \\
&= \left(\sum_{k,l} c_1 I_1(i-k, j-l) h(k,l) \right) + \left(\sum_{k,l} c_2 I_2(i-k, j-l) h(k,l) \right) \\
&= c_1 \left(\sum_{k,l} I_1(i-k, j-l) h(k,l) \right) + c_2 \left(\sum_{k,l} I_2(i-k, j-l) h(k,l) \right) \\
&= c_1 I_1(i,j) + c_2 I_2(i,j)
\end{aligned}$$

Exercise

Show that convolution is linear. In effect, show that if I_1 and I_2 are 2D digital arrays, and $c_1, c_2 \in \mathbb{R}$, then

$$(c_1 I_1 + c_2 I_2) * h = c_1 (I_1 * h) + c_2 (I_2 * h) \tag{1}$$

Solution Let $g_1 = I_1 * h$, $g_2 = I_2 * h$, and $g_3 = (c_1 I_1 + c_2 I_2) * h$.

$$\begin{aligned}
&g_3(i,j) = \\
&\sum\nolimits_{k,l} (c_1 I_1(i-k, j-l) + c_2 I_2(i-k, j-l)) h(k,l) \\
&= \left(\sum\nolimits_{k,l} c_1 I_1(i-k, j-l) h(k,l) \right) + \left(\sum\nolimits_{k,l} c_2 I_2(i-k, j-l) h(k,l) \right) \\
&= c_1 \left(\sum\nolimits_{k,l} I_1(i-k, j-l) h(k,l) \right) + c_2 \left(\sum\nolimits_{k,l} I_2(i-k, j-l) h(k,l) \right) \\
&= c_1 I_1(i,j) + c_2 I_2(i,j)
\end{aligned}$$

Different Notations for Convolution

$$g(i,j) = \sum_{k,l} I(i-k, j-l)h(k,l) = \sum_{k,l} h(i-k, j-l)I(k,l)\,. \quad (2)$$

The summation is done over all relevant pixels (*i.e.*, where h is defined). A slightly different way of writing the same thing is as

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{x}_i} I(\boldsymbol{x}-\boldsymbol{x}_i)h(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_i} h(\boldsymbol{x}-\boldsymbol{x}_i)I(\boldsymbol{x}_i) \quad (3)$$

where $\boldsymbol{x}$ denotes the location of the pixel of interest, and the $\boldsymbol{x}_i$'s denote the locations of the pixels where h is defined.

Example (When h is Gaussian)

$$g(\boldsymbol{x}) = \tfrac{1}{c}\sum_{\boldsymbol{x}_i} \exp\left(-\tfrac{1}{2}\frac{||\boldsymbol{x}-\boldsymbol{x}_i||^2}{\sigma^2}\right) I(\boldsymbol{x}_i) \quad (4)$$

where c typically is taken as a normalizer: $c = \sum_{\boldsymbol{x}_i} \exp\left(-\tfrac{1}{2}\frac{||\boldsymbol{x}-\boldsymbol{x}_i||^2}{\sigma^2}\right)$.

Impulse Response

Definition (2D discrete-domain impulse signal)

$$\delta(i,j) = \begin{cases} 1 & \text{if } i = 0 \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

- h is also called the "impulse response", since its convolution with an impulse signal is $h * \delta = \delta * h = h$

Definition (separable filter)

A 2D filter, $\boldsymbol{K}$, is called separable if it can be written as the outer product of two 1D filters; namely:

$$\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T \tag{6}$$

where $\boldsymbol{v}$ and $\boldsymbol{h}$ are two column vectors, typically of the same length

Convolution with a Separable Filter

For a separable filter, $\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T$, the 2D convolution, $g = I * \boldsymbol{K}$, can be done more efficiently via a 1D horizontal convolution followed by a 1D vertical convolution; *i.e.*, if $g = I * \boldsymbol{K}$, then it can be computed as:

- Convolve each row i with $\boldsymbol{h}$

$$\begin{aligned} \tilde{g}(i,:) &= I(i,:) * \boldsymbol{h} \\ \tilde{g}(i,j) &= \sum_l I(i, j-l)\boldsymbol{h}(l) \end{aligned} \tag{7}$$

- Convolve each column j of the result with $\boldsymbol{v}$:

$$\begin{aligned} g(:,j) &= \tilde{g}(:,j) * \boldsymbol{v} \\ g(i,j) &= \sum_k \tilde{g}(i-k, j)\boldsymbol{v}(k) \end{aligned} \tag{8}$$

Example (An isotropic "Gaussian")

A 5×5 discrete and truncated approximation of an isotropic Gaussian:

$$\boldsymbol{h} = \boldsymbol{v} = \frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Example (horizontal Sobel filter – approximates $\frac{\partial}{\partial x}$)

$$\boldsymbol{h} = \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{v} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T = \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Example (An isotropic "Gaussian")

A 5×5 discrete and truncated approximation of an isotropic Gaussian:

$$\boldsymbol{h} = \boldsymbol{v} = \frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Example (horizontal Sobel filter – approximates $\frac{\partial}{\partial x}$)

$$\boldsymbol{h} = \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{v} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$\boldsymbol{K} = \boldsymbol{v}\boldsymbol{h}^T = \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Laplacian of Gaussian (LoG)

In a continuous domain:

- A normalized and isotropic Gaussian filter is given by:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x^2 + y^2}{2\sigma^2})$$

- The Laplacian of I is

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) I$$

Laplacian of Gaussian (LoG)

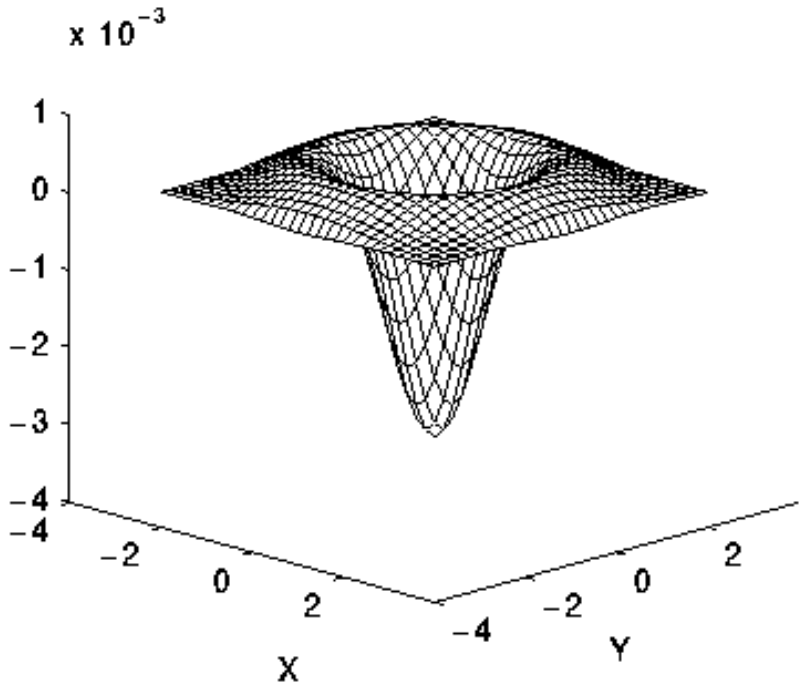
- Blurring I with $G(x, y, \sigma)$ followed up by applying ∇^2 can be done in a single operation, by convolving I with the LoG filter:

$$\nabla^2 G(x, y, \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y, \sigma)$$

- In practice, we approximate this using a finite filter, *e.g.*, 5×5.
- LoG is an example of an undirected filter: at a given point, it is invariant to the rotation of I around that point.

Laplacian of Gaussian (LoG)

$$\nabla^2 G(x, y, \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y, \sigma)$$



Definition (directional derivative)

The directional derivative of a function $f(x,y)$, in the direction of the unit vector $\boldsymbol{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$, denoted by $\nabla_{\boldsymbol{u}} = \frac{\partial}{\partial \boldsymbol{u}}$, is the scalar given by

$$\nabla_{\boldsymbol{u}} f \triangleq (\nabla_{\boldsymbol{x}} f)\boldsymbol{u} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} = \cos\theta \frac{\partial f}{\partial x} + \sin\theta \frac{\partial f}{\partial y}$$

where $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ and $\nabla_{\boldsymbol{x}} f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$.

Directed/Oriented Filters

- An example for an oriented filter is obtained via a Gaussian blurring followed by taking the directional derivative,

$$\underbrace{\nabla(G * I)}_{1\times 2}\underbrace{\boldsymbol{u}}_{2\times 1} = \nabla_{\boldsymbol{u}}(G * I) = (\nabla_{\boldsymbol{u}} G) * I \tag{9}$$

where $\boldsymbol{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ and

$$\nabla_{\boldsymbol{u}} G = u\frac{\partial G}{\partial x} + v\frac{\partial G}{\partial y} \tag{10}$$

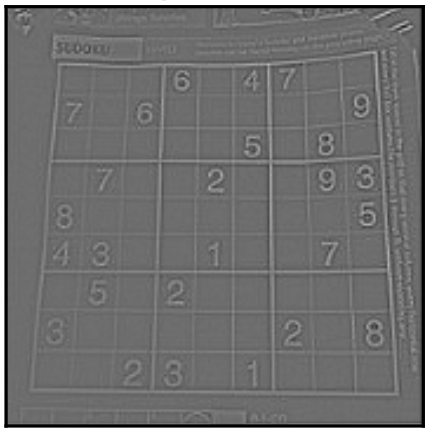
- The Sobel filter, $\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, approximates this for $\boldsymbol{u} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$.
- Remark: we briefly discussed the Steerable Random Field model: an MRF whose clique functions are responses to directed filters which were "steered" according to the spatial gradient.

Sobel and Laplacian Example

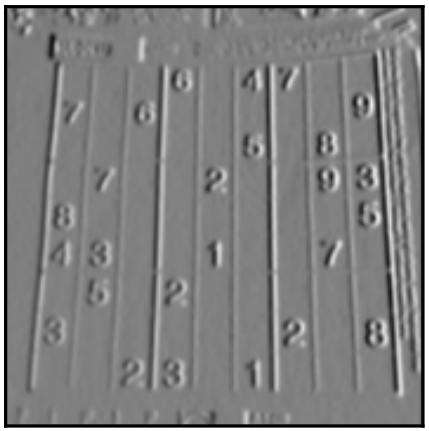
Horizontal Sobel $\approx \frac{\partial}{\partial x}$
Vertical Sobel $\approx \frac{\partial}{\partial y}$
Laplacian $\approx \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

Original

Laplacian

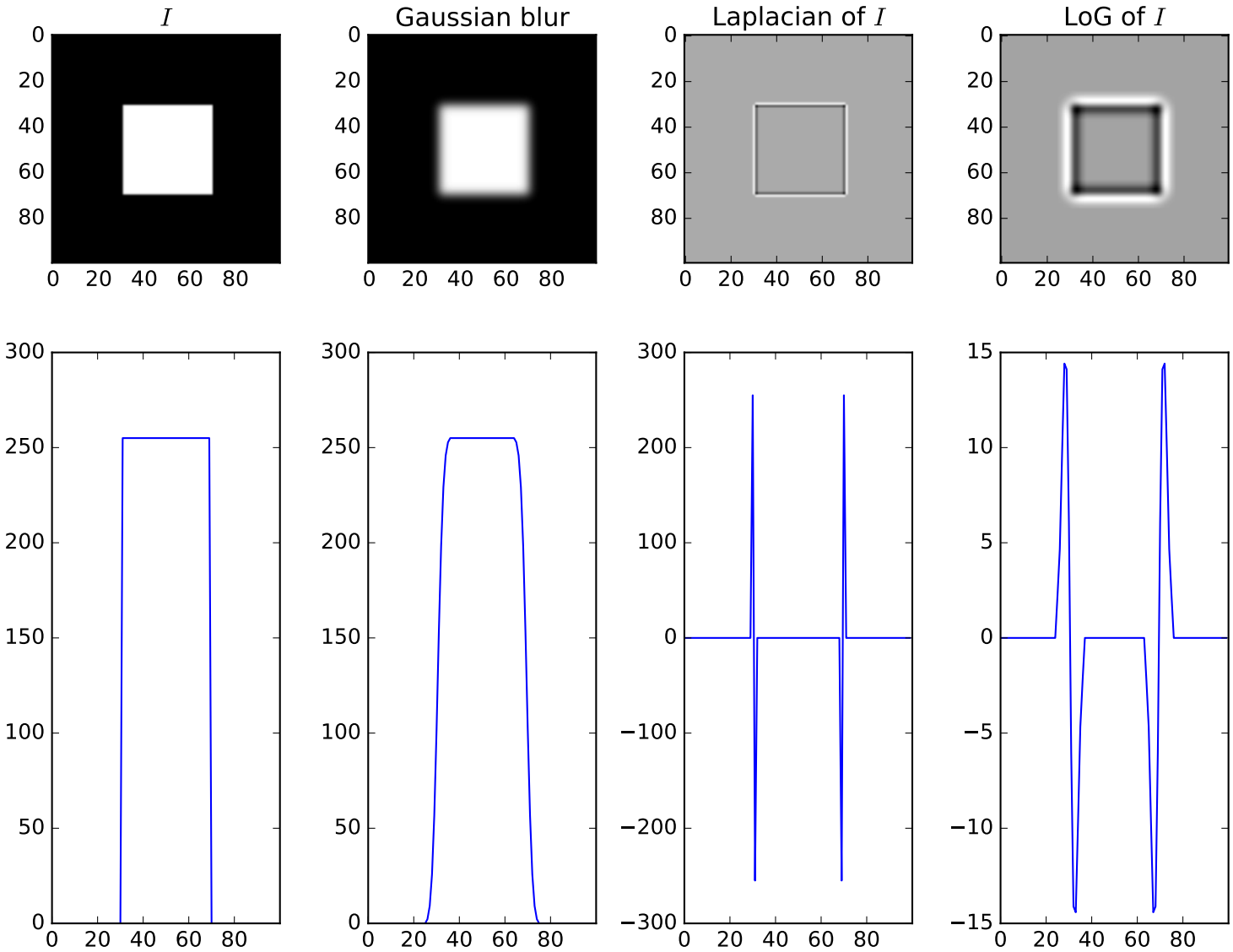


Sobel x



Sobel y

LoG Example



Median Filtering

- Replace pixel (i, j) with the median of the value in, say, a 5 by 5, neighborhood around it. This is a nonlinear operation.

- This is an example of a nonlinear filtering (*e.g.*, usually median(array1)+(array2)$\neq$ median(array1)+median(array2)

Images taken from Wikipedia