Representing Images as Functions

Image

$$I : \Omega \to \mathbb{R}^n$$

Domain of the function
$\Omega \subset \mathbb{R}^{\{1,2,3\}}$ is a rectangular domain.
1D $\to$ signal
2D $\to$ image
3D$\to$ volume

Image of the Function
n=1 $\to$ gray values
n=3 $\to$ RGB, HSV, etc.
n=4 $\to$ e.g. matrix valued images

Filters (Transformations on Images)
(sloppy)

$$f : \{I : \Omega \to \mathbb{R}^n\} \times \Omega \to \mathbb{R}^m$$

from now on: $I(x, y) =: I$.
Examples:
Inversion:

$$f(x, y) = -I(x, y) \text{ or } f(x, y) = I_{max} - I(x, y)$$

Saturation: (RGB valued image $\to$ gray valued image)

$$S(x, y) = \begin{cases} 0 & \text{if} \quad R(x, y) = G(x, y) = B(x, y) = 0 \\ \frac{\max\{R,G,B\} - \min\{R,G,B\}}{\max\{R,G,B\}} & \text{else} \end{cases}$$

Gradient:

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}^\top$$

Gradient Magnitude:

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

Laplacian:

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

Convolution:

$$f(x,y) = K * I(x,y) = \int_{\Omega} K(a,b) \cdot I(x-a, y-b) \mathrm{d}a\mathrm{d}b$$

Properties:

- Commutativity:
$$K * I = I * K$$
- Associativity:
$$(K_2 * K_1) * I = K_2 * (K_1 * I)$$
- Distributivity:
$$(K_2 + K_1) * I = (K_2 * I) + (K_1 * I)$$
- Associativity of Scalar Multiplication:
$$\lambda(K * I) = (\alpha K) * I = K * (\lambda I) \qquad \text{with scalar } \lambda$$

Convolution:

$$f(x,y) = K * I(x,y) = \int_{\Omega} K(a,b) \cdot I(x-a, y-b) \mathrm{d}a\mathrm{d}b$$

Properties:

- Associativity of Scalar Multiplication:

$$\lambda(K * I) = (\alpha K) * I = K * (\lambda I) \qquad \text{with scalar } \lambda$$

- Associativity of Derivatives:

$$(K * I)' = K' * I = K * I'$$

- The derivative of a function can be expressed as its convolution with the derivative of the Dirac distribution

$$I = I * \delta \Rightarrow I' = I * \delta'$$

Convolution:

$$f(x,y) = K * I(x,y) = \int_{\Omega} K(a,b) \cdot I(x-a, y-b) \mathrm{d}a\mathrm{d}b$$

In the discrete setting, a convolution of an image is computing a weighted sum in each pixel.

$$f(x,y) = K * I(x,y) = \sum_{S_K} K(a,b) \cdot I(x-a, y-b)$$

where S_K is the support of K, i.e. the positions where $K \neq 0$.

Although the convolution is commutative, in computer vision one often deals with the convolution of an *image* I with a small-support *kernel* K. Examples are the **Gaussian kernel** and the *Derivative kernel*.

Representing Images as Vectors

Filters like inversion (without a maximum value), gradient, Laplacian, an convolution are linear. This means, one can represent them as a *linear operator* F with the properties

$$F(\alpha I_1 + \beta I_2) = \alpha F I_1 + \beta F I_2 \qquad \alpha, \beta \in \mathbb{R}$$

If we regard a discretized image as a vector of n pixels $I \in \mathbb{R}^n$(and on a computer we have to discretize it eventually), we can regard the linear operator as a *matrix* $F \in \mathbb{R}^{m \times n}$.
The size of F is quadratic in the number of pixels, so F ist really large. But it is usually very *sparse*, meaning only a small constant number of entries in every row is not zero.