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:

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

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.

Diffusion

Last time we considered images as functions

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

Now we consider image evolutions over time

$$I : \Omega \times [0, t] \to \mathbb{R}^n$$

There are two physical models we are considering
First model: Fick's law
Differences of concentration in a field cause a flux in the direction opposing the concentration gradient

$$\vec{j} = -D \nabla I \qquad (1)$$

Image case: 2D-field in Ω:

$$j = \begin{pmatrix} j_1 \\ j_2 \end{pmatrix}, D = \begin{pmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{pmatrix}$$

D is called *Diffusion Tensor*, a positive, symmetric matrix in the 2D case.

Second model: Continuity equation

$$\frac{\partial I}{\partial t} + \text{div } j = c$$

Where the divergence is defined as

$$\text{div} j = \frac{\partial}{\partial x} j_1 + \frac{\partial}{\partial y} j_2 \tag{2}$$

and c is a constant for approaching/vanishing quantity.
Combining (1) and (2)

$$\frac{\partial I}{\partial t} = \text{div } (D \nabla I) \tag{3}$$

where ∇ is only with respect to x and y, i.e. $\nabla = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix}$.

Types of Diffusion

- Linear Diffusion D does not depend on Image I
  - isotropic:
$$D = \mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow \operatorname{div}(D\nabla I) = \Delta I \tag{4}$$
  - anisotropic: $\Rightarrow$ the resulting flux is not parallel to ∇I
- Nonlinear Diffusion: D depends on image I:
  - isotropic:
$$D = \mathbb{I} \cdot \varphi(I) = \begin{pmatrix} \varphi(I) & 0 \\ 0 & \varphi(I) \end{pmatrix} \Rightarrow \operatorname{div}(\varphi(I) \cdot \nabla I) = \Delta I \tag{5}$$
    where typically $\varphi(I)$ is in the range of $0 \leq \varphi(I) \leq 1$ for well-posedness purposes, but there are φ not fulfilling this constraint, which we are going to use.
  - anisotropic: D is only a positive definite matrix (with some restrictions on the eigenvalues)

Implementation and Discretization of the nonlinear isotropic diffusion

$$\frac{\partial I}{\partial t} = \mathrm{div}(\varphi \nabla I) = \frac{\partial}{\partial x}(\varphi \cdot \frac{\partial I}{\partial x}) + \frac{\partial}{\partial y}(\varphi \cdot \frac{\partial I}{\partial y})$$

Discretization of the temporal derivative by forward differences:

$$\frac{\partial I}{\partial t} \approx \frac{I(x,y,t+\tau) - \overbrace{I(x,y,t)}^{I}}{\tau}$$

Discretization of the outer spatial derivatives by means of central differences on half-pixel values:

$$\begin{aligned}
\frac{\partial}{\partial x}(\varphi \cdot \frac{\partial I}{\partial x}) &\approx (\varphi \cdot \frac{\partial I}{\partial x})(x + 1/2, y, t) - (\varphi \cdot \frac{\partial I}{\partial x})(x - 1/2, y, t) \\
\frac{\partial}{\partial y}(\varphi \cdot \frac{\partial I}{\partial y}) &\approx (\varphi \cdot \frac{\partial I}{\partial y})(x, y + 1/2, t) - (\varphi \cdot \frac{\partial I}{\partial y})(x, y - 1/2, t)
\end{aligned}$$

On the half-pixel values the function φ is approximated by arithmetic means and the derivatives are approximated by central differences of the adjacent pixels:

$$
\begin{aligned}
(\varphi \cdot \frac{\partial I}{\partial x})(x + 1/2, y, t) &\approx \underbrace{\frac{\varphi(x+1, y, t) + \varphi(x, y, t)}{2}}_{\varphi_r} \cdot (I(x+1, y) - I(x, y)) \\
(\varphi \cdot \frac{\partial I}{\partial x})(x - 1/2, y, t) &\approx \underbrace{\frac{\varphi(x-1, y, t) + \varphi(x, y, t)}{2}}_{\varphi_l} \cdot (I(x, y) - I(x-1, y)) \\
(\varphi \cdot \frac{\partial I}{\partial y})(x, y + 1/2, t) &\approx \underbrace{\frac{\varphi(x, y+1, t) + \varphi(x, y, t)}{2}}_{\varphi_u} \cdot (I(x, y+1) - I(x, y)) \\
(\varphi \cdot \frac{\partial I}{\partial y})(x, y - 1/2, t) &\approx \underbrace{\frac{\varphi(x, y-1, t) + \varphi(x, y, t)}{2}}_{\varphi_d} \cdot (I(x, y) - I(x, y-1))
\end{aligned}
$$

The final discretized scheme reads

$$\begin{aligned}
& \frac{I(x,y,t+\tau)-I(x,y,t)}{\tau} \\
= \; & \varphi_r I(x+1,y,t)+\varphi_l I(x-1,y,t) \\
+ \; & \varphi_u I(x,y+1,t)+\varphi_d I(x,y-1,t)-(\varphi_r+\varphi_l+\varphi_u+\varphi_d)I(x,y,t)
\end{aligned}$$

For $\varphi \equiv 1$ one gets the discretized Laplacian equation:

$$\begin{aligned}
& \frac{I(x,y,t+\tau)-I(x,y,t)}{\tau} \\
= \; & \underbrace{I(x+1,y,t)+(x-1,y,t)+I(x,y+1,t)+I(x,y-1,t)-4I(x,y,t)}_{\approx \Delta I}
\end{aligned}$$

Assuming we know the image at time t and want to compute it at time $t+1$:

$$\begin{aligned}
& I(x,y,t+\tau) \\
= \; & I(x,y,t)+\tau\cdot(\varphi_r I(x+1,y,t)+\varphi_l I(x-1,y,t) \\
+ \; & \varphi_u I(x,y+1,t)+\varphi_d I(x,y-1,t)-(\varphi_r+\varphi_l+\varphi_u+\varphi_d)I(x,y,t))
\end{aligned}$$

A natural assumption is to have the gradient vanish at the image boundaries, meaning that $\frac{\partial}{\partial x} I = 0$ at the left and right boundary and $\frac{\partial}{\partial y} I = 0$ at the top and bottom boundary. This ensures, that the average grey value of the image is preserved. you implement this by setting

$$\begin{aligned} I(-1, y, t) &:= I(0, y, t) \\ I(w, y, t) &:= I(w-1, y, t) \\ I(x, -1, t) &:= I(x, 0, t) \\ I(x, h, t) &:= I(x, h-1, t) \end{aligned}$$

If we see the image as a stacked-up vector:

$$I(t+\tau) = (\mathbb{I} + \tau A(\varphi(I(t))) \cdot I(t) \tag{6}$$

We want the matrix $(\mathbb{I} + A)$ to be non-negative, therefore, if $\varphi < 1$, $\varphi > 0$, we have $\varphi_r, \varphi_l, \varphi_u, \varphi_d \geq 0$, and the restriction of the time step size $\tau \leq 1/4$.

There are three ways of discretization of the continuous diffusion equation (3):

- Explicit: $I(t+\tau)$ is computed with spatial relations and diffusivity of time t:

$$
\begin{aligned}
\frac{I(t+\tau)-I(t)}{\tau} &= A(\varphi(I(t)))\cdot I(t)\\
I(t+\tau) &= (\mathbb{I}+\tau A(\varphi(I(t)))\cdot I(t)
\end{aligned}
$$

- Semi-Implicit: $I(t+\tau)$ is computed with spatial relations of time $t+\tau$ and diffusivity of time t:

$$
\begin{aligned}
\frac{I(t+\tau)-I(t)}{\tau} &= A(\varphi(I(t)))\cdot I(t+\tau)\\
I(t) &= (\mathbb{I}-\tau A(\varphi(I(t)))\cdot I(t+\tau)
\end{aligned}
$$

- Fully Implicit: $\Rightarrow$ Variational Method

Implement the linear diffusion with

$$\varphi(I) = 1$$

and the nonlinear diffusion with

$$\varphi(I) = \frac{1}{\sqrt{|\nabla I|^2 + \epsilon}}$$

Variational Methods

Given an image I^0 and wanting an image I, that is a smoother, denoised version of I^0. Define an energy:

$$E(I) = \int_{\Omega} (I - I^0)^2 + \lambda\phi(|\nabla I|^2) \, \mathrm{d}x \, \mathrm{d}y$$

The *data term* $(I - I^0)^2$ penalizes the difference of I to the original noisy image I^0.
The *regularity term* $\phi(|\nabla I|^2)$ penalizes the inhomogenity of I.
The scalar parameter λ weights closeness to the original image against regularity.
The minimizer I of this energy is an image that is close to the original noisy image, but is smooth as well.

There are two ways of minimizing a given energy (there are more, but those two are most intuitive):
1) Gradient Descent

$$\frac{\partial I}{\partial t} = -\frac{\mathrm{d}E}{\mathrm{d}I}$$

and look for a steady-state of this equation with

$$\frac{\partial I}{\partial t} = 0 \Rightarrow \frac{\mathrm{d}E}{\mathrm{d}I} = 0$$

2) Similar to "normal" calculus, a minimizer I^* of E has to fulfill the necessary condition

$$\frac{\mathrm{d}E}{\mathrm{d}I}(I^*) = 0$$

and for convex energies, this condition is sufficient and can be solved directly by means of the Euler-Lagrange equations.

Rewrite the energy as

$$\int_{\Omega} L\left(I, \frac{\partial}{\partial x} I, \frac{\partial}{\partial y} I, x, y\right) \, \mathrm{d}x \, \mathrm{d}y$$

where $L(I, \frac{\partial}{\partial x} I, \frac{\partial}{\partial y} I, x, y)$ is the integrant $((I - I^0)^2 + \lambda\phi(|\nabla I|^2))$.

Computing the derivative with respect to a function (and setting it to 0) by means of the Euler-Lagrange equation:

$$\frac{\mathrm{d}E}{\mathrm{d}I} = \frac{\partial L}{\partial I} - \frac{\partial}{\partial x}\left(\frac{\partial L}{\partial(\underbrace{\frac{\partial I}{\partial x}}_{I_x})}\right) - \frac{\partial}{\partial y}\left(\frac{\partial L}{\partial(\underbrace{\frac{\partial I}{\partial y}}_{I_y})}\right)$$

In our example:

$$\begin{aligned}
\frac{\partial L}{\partial I} &= 2(I - I^0) \\
\frac{\partial L}{\partial I_x} &= \lambda\phi'(|\nabla I|^2)2 \cdot I_x \\
\frac{\partial L}{\partial I_y} &= \lambda\phi'(|\nabla I|^2)2 \cdot I_y
\end{aligned}$$

1) The Gradient Descent scheme now reads as

$$
\begin{aligned}
\frac{\partial I}{\partial t} &= -\left(2(I - I^0) - \frac{\partial}{\partial x}\left(\lambda\phi'(|\nabla I|^2)2 \cdot I_x\right) - \frac{\partial}{\partial y}\left(\lambda\phi'(|\nabla I|^2)2 \cdot I_y\right)\right) \\
&= 2\left(\left(I^0 - I\right) + \lambda \operatorname{div}\left(\phi'(|\nabla I|^2)\nabla I\right)\right)
\end{aligned}
$$

or

$$
I(t+\tau) = I(t) + 2\tau\lambda\left(\left(\frac{I^0 - I(t)}{\lambda}\right) + \operatorname{div}\left(\phi'(|\nabla I(t)|^2)\nabla I(t)\right)\right)
$$

This is a diffusion equation with an additional reaction term $\frac{I^0 - I}{\lambda}$ that produces non-flat steady states.

2) The elliptic Euler-Lagrange equation now reads as

$$\frac{\mathrm{d}E}{\mathrm{d}I} = 0 \Rightarrow 2(I - I^0) - 2\lambda \cdot \ \mathrm{div}(\phi'(|\nabla I|^2) \cdot \nabla I) = 0$$

Rewrite:

$$\frac{I - I^0}{\lambda} = \ \mathrm{div}(\phi'(|\nabla I|^2) \cdot \nabla I)$$

This can be interpreted as a diffusion equation with one large time step size λ and $I^0 = I(x, y, 0)$.
This is the fully implicit scheme

$$I = I^0 + \lambda \ \mathrm{div}(\phi'(|\nabla I|^2)\nabla I)$$

which can be rewritten as

$$I^0 = (\mathbb{I} - \lambda A(\phi'(|\nabla I|^2)))I$$

For the regularity function

$$\phi(|\nabla I|^2) = 2\sqrt{|\nabla I|^2} = 2|\nabla I|$$

we get the diffusivity

$$\phi'(|\nabla I|^2) = \frac{1}{\sqrt{|\nabla I|^2}}$$

To avoid problems with the unbounded diffusivity for $|\nabla I| = 0 \rightarrow$ smooth it:

$$\phi'_\epsilon(|\nabla I|^2) = \frac{1}{\sqrt{\epsilon + |\nabla I|^2}}$$

The diffusivity $\varphi = \phi'$ in the resulting diffusion equation is the derivative of the function ϕ in the regularity term.

The non-linear system of equations

$$I^0 = (\mathbb{I} - \lambda A(\phi'(|\nabla I|^2)))I$$

can be solved (for certain diffusivities) by a fixed-point iteration scheme of linear equation systems:
Start with $k = 0$
Compute I^{k+1} as the solution of

$$I^0 = (\mathbb{I} - \lambda A(\phi'(|\nabla I^k|^2)))I^{k+1}$$

update $A(\phi'(|\nabla I^{k+1}|^2)))$ and repeat until convergence or timeout.
The equation is now linear in I.

Why are we doing this? (in comparison to diffusion)

- For $\lambda = \tau \cdot \#$ of iterations and a large $\#$ of iterations, solving an equation system is significantly faster than diffusion (up to 1 order of magnitude, but it depends on the method)
- Additional benefit: ϵ can be smaller than in the explicit diffusion scheme

For the rest of this course, we will focus on solving the elliptic Euler-Lagrange equations, while in the second part of the project, you will encounter a variant of a gradient descent scheme.

Solving linear systems of equations

The Jacobi Method

Task is to solve

$$Ax = b$$

The idea behind is to split a into two matrices $A = D + R$, a diagonal matrix D and the off-diagonal matrix R.

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & a_{nn} \end{pmatrix} R = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{pmatrix}$$

Using this substitution one can derive a recursive formula for x which contains the inverse of D, but D is chosen to be inverted easily:

$$
\begin{aligned}
(D+R)x &= Dx + Rx = b \\
Dx &= b - Rx \\
x &= D^{-1}(b - Rx) \\
\Downarrow \quad \text{introduce time variable } k \quad &\Downarrow \\
x^{k+1} &= D^{-1}(b - Rx^k) \\
x_i^{k+1} &= \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^k \right)
\end{aligned}
$$

The Gauss-Seidel Method

Similar to the Jacobi method we want to solve

$$Ax = b$$

But matrix A is split differently into two matrices $A = L_* + U$, a lower triangular matrix with diagonal entries L_* and the upper triangular matrix U.

$$L_* = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & 0 \\ a_{n1} & \dots & a_{n,n-1} & a_{nn} \end{pmatrix} U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \dots & 0 & 0 \end{pmatrix}$$

Using the same transformations one again can derive a recursive formula for x, but this time the inversion of L_* is calculated by forward-substitution.

$$\begin{aligned}
(L_* + U)x &= L_*x + Ux = b \\
L_*x &= b - Ux \\
x &= L_*^{-1}(b - Ux)
\end{aligned}$$

$\Downarrow$ introduce time variable k $\Downarrow$

$$\begin{aligned}
x^{k+1} &= L_*^{-1}(b - Ux^k) \\
x_i^{k+1} &= \frac{1}{a_{ii}}\Big(b_i - \underbrace{\sum_{j>i} a_{ij}x_j^k}_{U} - \underbrace{\sum_{j<i} a_{ij}x_j^{k+1}}_{L_*}\Big)
\end{aligned}$$

The Gauss-Seidel method converges faster than the Jacobi method.

Successive Over-Relaxation (SOR) Method

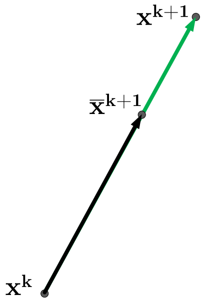

Fig.: Linear extrapolation.

Successive Over-Relaxation simple uses linear extrapolation of the result from the Gauss-Seidel method for faster convergence. If $\bar{x}^{k+1}$ is the result of one Gauss-Seidel step based on x^k one calculates the new x^{k+1} by linear extrapolation:

$$x^{k+1} = (1-\omega)x^k + \omega\bar{x}^{k+1} \tag{7}$$

where $\omega \in (0, 2)$ is a linear interpolation/extrapolation variable.

The method is proven to converge for values of ω between 0 and 2. The optimal choice for ω depends on the matrix A, in practice one uses values around $1.5 - 1.9, e.g. 1.7$. Note for values $\omega \in (0, 1)$ (interpolation) the convergences will slow down and for values $\omega \in (1, 2)$ (extrapolation) convergence is accelerated. For $\omega = 1$ the method reduces to the Gauss-Seidel method.

Hence a single SOR step results in:

$$x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{a_{ii}}\Big(b_i - \underbrace{\sum_{j>i} a_{ij}x_j^k}_{U} - \underbrace{\sum_{j<i} a_{ij}x_j^{k+1}}_{L_*}\Big) \tag{8}$$

A side-result of the derivation of the Euler-Lagrange Equations is that the the gradient of I vanishes at the image boundaries, which we have already implemented for the explicit diffusion. However, setting the off-boundary values to the boundary values, e.g. $I(-1, y, t) := I(0, x, t)$ will corrupt the Jacobi and SOR schemes. In the Jacobi update

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij}x_j^k\right)$$

this would mean replacing one x_j by x_i, which is not correct. The correct way is to eliminate the dependency of x_j for x_i in the system matrix A.