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:

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

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 $\Downarrow$

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

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

Successive Over-Relaxation (SOR) Method

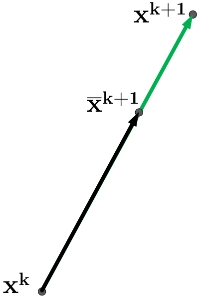

Fig. : Linear extrapolation.

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:

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

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:

$$x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{a_{ii}}\Big(b_i - \underbrace{\sum_{j>i} a_{ij}x_j^k}_{U} - \underbrace{\sum_{j<i} a_{ij}x_j^{k+1}}_{L_*}\Big) \tag{2}$$

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.