Numerical Linear Algebra
A.A. 2024/2025

Lecturer: Prof. P. F. Antonietti TA: Dott. M. Botti

Challenge 1 - Image filtering and denoising

Goal: Apply image filters and find the approximate solution of linear system to process a greyscale image.
Data: Download the file `einstein.jpg` (256px) and move it to your working directory.

1 Overview

1.1 Image filtering

First, we will look at the applications of matrix multiplication to the filtering of images. The most common operations are blurring/smoothing, sharpening, and edge detection. All of these are examples of so-called convolution filters, that work by changing the color of the pixels in the image using the colors of nearby pixels.

Consider an image described by the matrix $F = (f_{ij})$ where $(i; j)$ are integer coordinates of the pixel in an image, and the value f_{ij} returns the color of the image. Consider $H = (h_{kl})$ to be a convolution matrix (filter matrix). These matrices might have (and usually do have) different dimensions since H is usually relatively small. Then a new matrix $G = F \star H$ (i.e. the convolution of F and H) can be found by the following formula:

$$g_{ij} = \sum_k \sum_l f_{i-k+1,j-l+1} h_{kl},$$

where the summation runs over all the indices which make sense. In particular, if an element with a certain index does not exist, we simply assume the corresponding term to be zero. You can think of convolution being obtained by the filter matrix H sliding over the original image F. Two of the most common operations on images done with convolution filters include smoothing, sharpening, and edge detection. In the following we provide some examples:

- Two average smoothing filters are given by the matrices:

$$H_{av1} = \frac{1}{8}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \qquad H_{av2} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

- An opposite action to blurring is the sharpening of an image with filters such as:

$$H_{sh1} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \qquad H_{sh2} = \begin{pmatrix} 0 & -3 & 0 \\ -1 & 9 & -3 \\ 0 & -1 & 0 \end{pmatrix}.$$

- Finally, it is possible to use convolution to detect edges in the image. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer

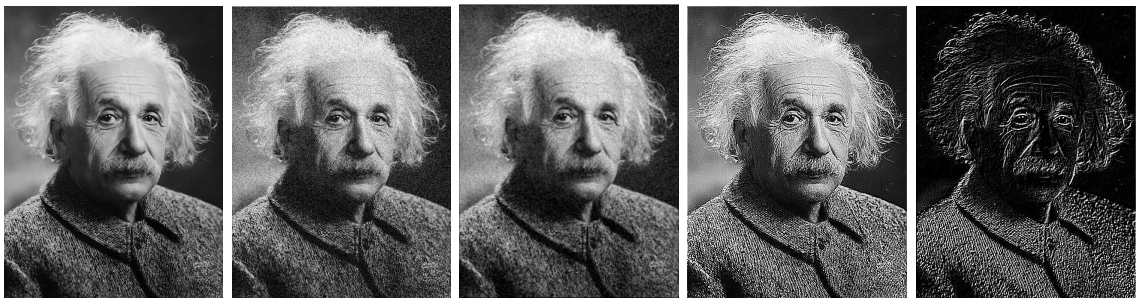

Figure 1: From left to right, original image, image with noise, smoothing filter applied to the noisy image, sharpening filter applied to the original image, edge detection filter applied to the original image.

vision, and machine vision. Two of the most common filters used for these are the Sobel horizontal and vertical filters:

$$H_{ed1} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \qquad H_{ed2} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}.$$

- Alternatively, Laplacian edge detection can be used with the following filter:

$$H_{lap} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

The Laplace kernel is a discrete analogue of a continuous Laplacian and may be interpreted as a sum of two discrete second order partial derivatives.

1.2 Image denoising via anisotropic diffusion

In image processing and computer vision, anisotropic diffusion (also called Perona–Malik diffusion) is a technique aiming at reducing image noise that focuses on preserving the edges of the various shapes present in the image. Anisotropic diffusion resembles the process that creates a scale space, where an image generates a parameterized family of successively more and more blurred images based on a diffusion process. Each of the resulting images in this family are given as a convolution between the image and an isotropic Gaussian filter. This diffusion process is a linear and space-invariant transformation of the original image. Anisotropic diffusion is a generalization of this process resulting in a combination between the original image and a filter that depends on the local content.

Consider a proper discretization of the anisotropic diffusion model, we can compute the denoised images as the solutions of the following sequence of linear systems

$$I_h^n + \Delta t\, A I_h^n = I_h^{n-1} \qquad \text{for all } \; 1 \leq n \leq N, \tag{1}$$

where I_h^0 is the input image with noise and Δt is the selected time-step. Notice that (1) can be interpreted as a sequence of filters applied to the noisy image.

2 Tasks

The tasks of this challenge are also reported in `https://forms.office.com/e/1QP9fq7BZQ`.

- Load the image as an Eigen matrix with size $m \times n$. Each entry in the matrix corresponds to a pixel on the screen and takes a value somewhere between 0 (black) and 255 (white). Report the size of the matrix.
- Introduce a noise signal into the loaded image by adding random fluctuations of color ranging between $[-50, 50]$ to each pixel. Export the resulting image in `.png` and upload it.
- Reshape the original and noisy images as vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, respectively. Verify that each vector has $m\,n$ components. Report here the Euclidean norm of $\boldsymbol{v}$.
- Write the convolution operation corresponding to the smoothing kernel H_{av2} as a matrix vector multiplication between a matrix A_1 having size $mn \times mn$ and the image vector. Report the number of non-zero entries in A_1.
- Apply the previous smoothing filter to the noisy image by performing the matrix vector multiplication $A_1\boldsymbol{w}$. Export and upload the resulting image.
- Write the convolution operation corresponding to the sharpening kernel H_{sh2} as a matrix vector multiplication by a matrix A_2 having size $mn \times mn$. Report the number of non-zero entries in A_2. Is A_2 symmetric?
- Apply the previous sharpening filter to the original image by performing the matrix vector multiplication $A_2\boldsymbol{v}$. Export and upload the resulting image.
- Export the Eigen matrix A_2 and vector $\boldsymbol{w}$ in the `.mtx` format. Using a suitable iterative solver and preconditioner technique available in the LIS library compute the approximate solution to the linear system $A_2\boldsymbol{x} = \boldsymbol{w}$ prescribing a tolerance of 10^{-9}. Report here the iteration count and the final residual.
- Import the previous approximate solution vector $\boldsymbol{x}$ in Eigen and then convert it into a `.png` image. Upload the resulting file here.
- Write the convolution operation corresponding to the detection kernel H_{lap} as a matrix vector multiplication by a matrix A_3 having size $mn \times mn$. Is matrix A_3 symmetric?
- Apply the previous edge detection filter to the original image by performing the matrix vector multiplication $A_3\,\boldsymbol{v}$. Export and upload the resulting image.
- Using a suitable iterative solver available in the Eigen library compute the approximate solution of the linear system $(I+A_3)\boldsymbol{y} = \boldsymbol{w}$, where I denotes the identity matrix, prescribing a tolerance of 10^{-10}. Report here the iteration count and the final residual.
- Convert the image stored in the vector $\boldsymbol{y}$ into a `.png` image and upload it.
- Comment the obtained results.