An Introduction to Mathematical Image Processing
IAS, Park City Mathematics Institute, Utah
Undergraduate Summer School 2010

Instructor: Luminita Vese
Teaching Assistant: Todd Wittman
Department of Mathematics, UCLA
lvese@math.ucla.edu wittman@math.ucla.edu

Lecture Meeting Time: Mon, Tue, Thu, Fri 1-2pm (location: Coalition 1&2)
Problem Session: 4.30-5.30, 5.30-6.30 (location: computer lab or tent)

Working Version

Abstract

Image processing is an essential field in many applications, including medical imaging, astronomy, astrophysics, surveillance, video, image compression and transmission, just to name a few. In one dimension, images are called signals. In two dimensions we work with planar images, while in three dimensions we have volumetric images (such as MR images). These can be gray-scale images (single-valued functions), or color images (vector-valued functions). Noise, blur and other types of imperfections often degrade acquired images. Thus such images have to be first pre-processed before any further analysis and feature extraction.

In this course we will formulate in mathematical terms several image processing tasks: image denoising, image deblurring, image enhancement, image segmentation, edge detection. We will learn techniques for image filtering in the spatial domain (using first- and second-order partial derivatives, the gradient, Laplacian, and their discrete approximations by finite differences, averaging filters, order statistics filters, convolution), and in the frequency domain (the Fourier transform, low-pass and high-pass filters), zero-crossings of the Laplacian. If time permits, we will learn more advanced methods that can be formulated as a weighted Laplace equation for image restoration, and curve evolution techniques called snakes for image segmentation (including total variation minimization, active contours without edges, anisotropic diffusion equation, NL Means).

• Prerequisites for the course include basic curriculum in calculus and linear algebra. Most concepts will be defined as they are introduced. Some exposure, e.g. through a basic introductory course, to numerical analysis, Fourier analysis, partial differential equations, statistics, or probability will definitely be useful though we will try to cover the necessary facts as they arrive.

• For computational purposes, some familiarity with the computer language Matlab is useful. You may wish to explore in advance the introductory tutorial on *Image Processing Using Matlab*, prepared by Pascal Getreuer, on how to read and write image files, basic image operations, filtering, found at the online link: http://www.math.ucla.edu/˜ getreuer/matlabimaging.html

• The Discussion Sections will be devoted to problem solving, image processing with Matlab, summary of current lecture, or to exposition of additional topics.

• For an introduction to image processing, a useful reading textbook is:
[7] R.C. Gonzalez and R.E. Woods, Digital Image Processing, 3rd edition, Prentice-Hall.
See also
[8] R.C. Gonzalez, R.E. Woods and S.L. Eddins, Digital Image Processing Using Matlab, 2nd edition, Prentice-Hall.

• Please visit also the webpage of these two textbooks for review material, computational projects, problem solutions, images used in the textbook, imaging toolbox, and many other useful information:
http://www.imageprocessingplace.com/index.htm

• The website for the course can be found at: http://www.math.ucla.edu/˜ lvese/155.1.09w/PCMI_USS_2010 where lecture notes, problems, computer projects and other links are posted.

• An introduction and overview of the course can be found on the course webpage. Click on "Lecture1.pdf" for slides presentation.

Topics to be covered

Fundamental steps in image processing

A simple image formation model

Image sampling and quantization

Intensity transformations and spatial filtering
- Histogram equalization
- Linear spatial filters, correlation, convolution
- Smoothing (linear) spatial filters
- Sharpening linear spatial filters using the Laplacian

Filtering in the frequency domain
- 1D and 2D continuous and discrete Fourier transforms
- convolution theorem
- properties of the Fourier transform
- filtering in the frequency domain (smoothing and sharpening, low-pass and high-pass filtering)
- the Laplacian in the frequency domain, enhancement
- homomorphic filtering
- band-reject and band-pass filters

Image restoration and reconstruction
- noise models
- mean filters
- order statistics filters
- adaptive median filter
- periodic noise reduction
- NL Means filter
- linear, position-invariant degradations
- examples of degradation (PSF) functions
- inverse filtering (Wiener filter, constrained least squares filtering, total variation minimization)
- image reconstruction from projections (Radon transform, computed tomography, the Fourier slice Thm., filtered backprojections using parallel beam)

Image segmentation
- image gradient, gradient operators, gradient-based edge detection
- the Marr-Hildreth edge detector, Canny edge detector
- active contours
- global processing using the Hough transform

Contents

1 Fundamental Steps in Digital Image Processing

This field can be divided into several areas without clear boundaries:

- image processing
- image analysis
- computer vision

or into

- low-level vision
- mid-level vision
- high-level vision

In more details, the fundamental steps are as follows.

Image acquisition (*output: digital image*)

• **Low-level vision**: input = image, output = image

Image enhancement: subjective process (e.g. image sharpening)

Image Restoration: objective process, denoising, deblurring, etc (depends on the degradation)

Color Image Processing: there are several color modes. Color (R,G,B) images are represented by vector-valued functions with three components; natural extensions from gray-scale to color images (most of the time)

Wavelets: advance course, mathematical tool to allow representation of images at various degrees of resolutions, used in many image processing tasks .

Compression: reducing the storage required to save an image (jpeg 2000)

• **Midd-level vision**: input = image, output = image attributes

Mathematical morphology: tools for extracting image components useful in the representation and description of shape.

Segmentation: to partition an image into its constituent parts or objects.

• **High-level vision** Input: boundaries and regions. Output: image attributes

Representation and description: following segmentation, gives representation of boundaries and regions; description given by a set object attributes or features.

Recognition: assign a label (e.g., "vehicle") to an object based on its descriptors

This course will deal with low-level and midd-level tasks.

2 A simple image formation model

In the continuous case, a planar image is represented by a two-dimensional function $(x, y) \mapsto f(x, y)$. The value of f at the spatial coordinates (x, y) is positive and it is determined by the source of the image. If the image is generated from a physical process, its intensity values are proportional to energy radiated by a physical source. Therefore, $f(x, y)$ must be nonzero and finite:

$$0 < f(x, y) < \infty$$

The image-function f may be characterized by two components:

(1) the amount of source illumination incident on the scene (illumination) $i(x, y)$

(2) the amount of illumination reflected by the objects (reflectance) $r(x, y)$

We have

$$f(x,y) = i(x,y)r(x,y)$$

where

$$0 < i(x,y) < \infty \text{ and } 0 < r(x,y) < 1.$$

Reflectance is bounded below by 0 (total absorption) and above by 1 (total reflectance).

$i(x,y)$ depends on the illumination source

$r(x,y)$ depends on the characteristics of the imaged objects.

The same expressions are applicable to images formed via transmission of the illumination through a medium (chest X-ray, etc). Then we deal with *transmissivity* instead of *reflectivity*.

From the above construction, we have

$$L_{min} \leq l = f(x,y) \leq L_{max},$$

where $l = f(x,y)$ is the gray-level at coordinates (x,y). It is common to shift the gray-scale (or intensity scale) from the interval $[L_{min}, L_{max}]$ to the interval $[0, L-1]$. Then $l = 0$ is considered black and $l = L-1$ is considered white on the gray scale. The intermediate values are shades varying from black to white.

2.1 Image sampling and quantization

Need to convert the continuous sensed data into digital form via two processes: *sampling* and *quantization*.

Sampling = digitizing the coordinate values

Quantization = digitizing the amplitude values

A digital image is represented as a $M \times N$ matrix: $f = \begin{bmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,N-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,N-1} \\ \cdots & \cdots & \cdots & \\ f_{M-1,0} & f_{M-1,1} & \cdots & f_{M-1,N-1} \end{bmatrix}$.

The number of gray levels L is taken to be a power of 2: $L = 2^k$ for some integer $k > 0$. Many times, digital images take values 0, 1, 2, ..., 255, thus 256 distinct gray levels.

3 Intensity transformations and spatial filtering

3.1 Histogram equalization

The histogram equalization technique is a statistical tool for improving the contrast of images. The input image is $f(x,y)$ (as a low contrast image, dark image, or light image). The output is a high contrasted image $g(x,y)$. Assume that the gray-level range consists of L gray-levels.

In the discrete case, let $r_k = f(x,y)$ be a gray level of f, $k = 0, 1, 2, ...L-1$. Let $s_k = g(x,y)$ be the desired gray level of output image g. We find a transformation $T : [0, L-1] \mapsto [0, L-1]$ such that $s_k = T(r_k)$, for all $k = 0, 1, 2, ..., L-1$.

Define $h(r_k) = n_k$ where r_k is the kth gray level, and n_k is the number of pixels in the image f taking the value r_k.

Visualizing the discrete function $r_k \mapsto h(r_k)$ for all $k = 0, 1, 2, ..., L-1$ gives the histogram.

We can also define the normalized histogram: let $p(r_k) = \frac{n_k}{n}$, where n is the total number of pixels in the image (thus $n = MN$).

The visualization of the discrete mapping $r_k \mapsto p(r_k)$ for all $k = 0, 1, 2, ..., L-1$ gives the normalized histogram. We note that $0 \leq p(r_k) \leq 1$ and

$$\sum_{k=0}^{L-1} p(r_k) = 1.$$

See "Lecture1.pdf" for a slide showing several images and their corresponding histograms. We notice that a high-contrasted image has a more uniform, almost flat histogram. Thus, the idea is to change the intensity values of f so that the output image g has a more uniform, almost flat histogram. We will define $T : [0, L-1] \mapsto [0, L-1]$ as an increasing transformation using the cumulative histogram.

In the discrete case, let $r_k = f(x, y)$. Then we define the histogram-equalized image g at (x, y) by

$$g(x, y) = s_k = (L-1) \sum_{j=0}^{k} p(r_j). \tag{1}$$

We refer to "Lecture1.pdf" for the slide showing the effect of histogram equalization applied to poorly contrasted images.

Theoretical interpretation of histogram equalization (continuous case) We view the gray levels r and s as random variables with associated probability distribution functions $p_r(r)$ and $p_s(s)$. The continuous version of (1) uses the cumulative distribution function and we define

$$s = T(r) = (L-1) \int_0^r p_r(w) dw.$$

If $p_r(r) > 0$ on $[0, L-1]$, then T is strictly increasing from $[0, L-1]$ to $[0, L-1]$, thus T is invertible. Moreover, if T is differentiable, then we can use a formula from probabilities: if $s = T(r)$, then

$$p_s(s) = p_r(r) \Big| \frac{\partial r}{\partial s} \Big|,$$

where we view $s = s(r) = T(r)$ as a function of r, and $r = r(s) = T^{-1}(s)$ as a function of s.

From the definition of s, we have

$$\frac{\partial s}{\partial r} = (L-1) p_r(r) = T'(r),$$

thus

$$\frac{\partial r}{\partial s} = \frac{\partial}{\partial s} \big(T^{-1}(s) \big) = \frac{1}{T'(s)} = \frac{1}{(L-1) p_r(r(s))}.$$

Therefore

$$p_s(s) = p_r(r) \Big| \frac{\partial r}{\partial s} \Big| = p_r(r) \Big| \frac{1}{(L-1) p_r(r)} \Big| = \frac{1}{L-1}.$$

In other words, p_s is the uniform probability distribution function on the interval $[0, L-1]$, thus corresponding to a flat histogram in the discrete case.

3.2 Spatial Linear Filters

A spatial filter consists of

(1) a neighborhood (typically a small rectangle)

(2) a predefined operation that is performed on the image pixels encompassed by the neighborhood.

The output of the operation in (2) will provide the value of the output image $g(x, y)$.

Let $f(x, y)$ be the input image, and $g(x, y)$ the output image in the discrete case, thus (x, y) are viewed as integer coordinates, with $0 \leq x \leq M-1$ and $0 \leq y \leq N-1$. Therefore, both images f and g are of size $M \cdot N$. We define a neighborhood centered at the point (x, y) by

$$S_{(x,y)} = \Big\{(x+s, y+t),\ -a \leq s \leq a,\ -b \leq t \leq b\Big\},$$

where $a, b \geq 0$ are integers. The size of the patch $S_{(x,y)}$ is $(2a+1)(2b+1)$, and we denote by $m = 2a+1$ and $n = 2b+1$ (odd integers), thus the size of the patch becomes $m \cdot n$, and $a = \frac{m-1}{2}$, $b = \frac{n-1}{2}$.

For example, the restriction $f|_{S(x,y)}$ for a 3×3 neighborhood $S_{(x,y)}$ is represented by

$$f|_{S_{(x,y)}} = \begin{array}{|c|c|c|} \hline f(x-1, y-1) & f(x-1, y) & f(x-1, y+1) \\ \hline f(x, y-1) & f(x, y) & f(x, y+1) \\ \hline f(x+1, y-1) & f(x+1, y) & f(x+1, y+1) \\ \hline \end{array}.$$

We also define a window mask $w = w(s, t)$, for all $-a \leq s \leq a$, $-b \leq t \leq b$, of size $m \times n$. For the 3×3 case, this window is

$$w = \begin{array}{|c|c|c|} \hline w(-1,-1) & w(-1,0) & w(-1,+1) \\ \hline w(0,-1) & w(0,0) & w(0,+1) \\ \hline w(+1,-1) & w(+1,0) & w(+1,+1) \\ \hline \end{array}.$$

We are now ready to define a linear spatial filter. The output image g is defined by

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x+s, y+t), \tag{2}$$

with the condition that care must be taken at the boundary of the image f, in the case when $f(x+s, y+t)$ is not well defined. To solve this problem, we can extend the image f by zero in a band outside of its original domain, or we can extend it by periodicity, or by reflection (mirroring). In all these cases, all values $f(x+s, y+t)$ will be well defined.

It is left as an exercise to show that the operation $f \mapsto g$ defined by (2) is a linear operation (with respect to real numbers, satisfying the additivity and scalar multiplication).

For a 3×3 window mask and neighborhood, the filter in (2) becomes

$$\begin{aligned} g(x, y) &= w(-1,-1) f(x-1, y-1) + w(-1, 0) f(x-1, y) + w(-1, 1) f(x-1, y+1) \\ &+ w(0,-1) f(x, y-1) + w(0, 0) f(x, y) + w(0, 1) f(x, y+1) \\ &+ w(1,-1) f(x+1, y-1) + w(1, 0) f(x+1, y) + w(1, 1) f(x+1, y+1). \end{aligned}$$

We will consider in this section two classes of spatial linear filters:

(i) smoothing linear spatial filters

(ii) sharpening linear spatial filters

Smoothing can be understood as local averaging (or blurring),which is similar with spatial summation or spatial integration. Small details and noise will be lost in the smoothing process, but sharp edges will become blurry.

Sharpening has the opposite effect to smoothing. A blurry image f can be made sharper through this process, details like edges will be enhanced.

Smoothing linear spatial filters Let us give two examples of smoothing spatial masks w of size 3×3. The first one gives the average filter or the box filter, $w = \frac{1}{9}\begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&1&1\\\hline 1&1&1\\\hline\end{array}$, and the second one can be seen as a discrete version of a two-dimensional Gaussian function $e^{-\frac{x^2+y^2}{2\sigma^2}}$: $w = \frac{1}{16}\begin{array}{|c|c|c|}\hline 1&2&1\\\hline 2&4&2\\\hline 1&2&1\\\hline\end{array}$ (the coefficients of w decrease as we move away from the center). Note that, for both these filter masks w, the sum of the coefficients $\sum_{s=-1}^{1}\sum_{t=-1}^{1} w(s,t) = 1$. Thus, the input image f and the output image g have the same range of intensity values.

As we can see from the above two examples, it is convenient to directly normalize the filter, by considering the general *weighted average filter* of size $m \times n$:

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}.$$

The above equation is computed for all (x,y) with $x = 0,1,2,...,M-1$ and $y = 0,1,2,...,N-1$. Note that the denominator is a constant, thus it has to be computed only once for all (x,y).

Additional mean filters (linear and nonlinear) will be discussed later in the section on image denoising.

Sharpening linear filters In this section, we will present a method for image enhancement using the Laplacian. Let's assume for a moment that the image function f has second order partial derivatives. We define the Laplacian of f in the continuous case by

$$\triangle f(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y).$$

You will prove as an exercise that the mapping $f \mapsto \triangle f$ is linear and rotationally invariant.

We remember from numerical analysis, that in one dimension, the second order derivative of f, f'' at the point x, can be approximated by finite differences of values of f at x, $x+h$, $x-h$:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}, \tag{3}$$

where $h > 0$ is a small discretization parameter.

First we approximate $\frac{\partial^2 f}{\partial x^2}(x,y)$ using (3) keeping y fixed, then $\frac{\partial^2 f}{\partial y^2}(x,y)$ using (3) keeping x fixed, by

$$\frac{\partial^2 f}{\partial x^2}(x,y) \approx \frac{f(x+h,y) - 2f(x,y) + f(x-h,y)}{h^2},$$

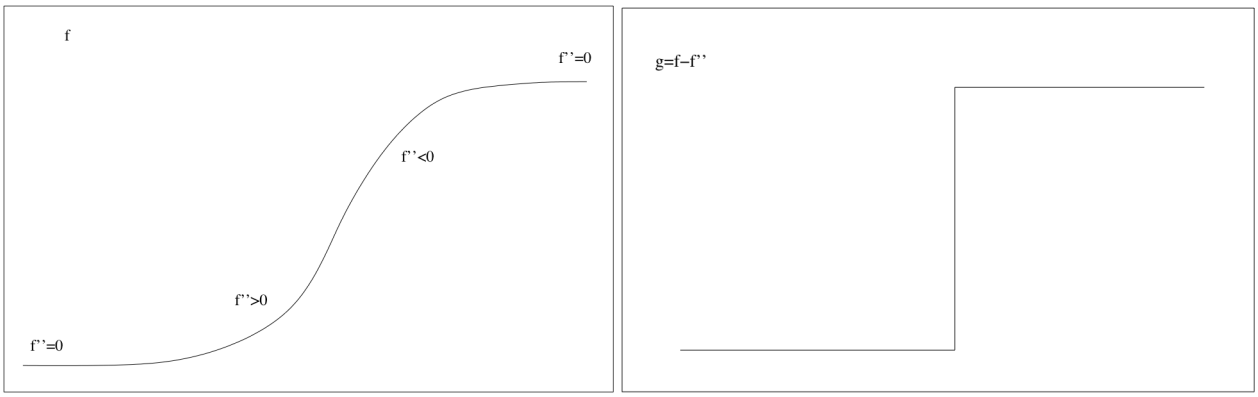

Figure 1: Illustration of the enhancement using the Laplacian. Left: a blurry 1D edge of the image f, and the sign of f''. Right: the resulting $g = f - f''$ as an ideal sharp edge.

$$\frac{\partial^2 f}{\partial y^2}(x,y) \approx \frac{f(x,y+h) - 2f(x,y) + f(x,y-h)}{h^2}.$$

For images we take $h = 1$. Summing up these equations, we obtain a discretization of the Laplacian naturally called the 5-point Laplacian:

$$\triangle f(x,y) \approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y), \tag{4}$$

which can be applied to discrete images f. We notice that the 5-point Laplacian from (4) can be obtained using a linear spatial filter with "Laplacian mask" $w = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$.

It is possible to show that the Laplacian can also be discretized by a 9-point Laplacian, with formula and corresponding Laplacian mask w

$$\triangle f(x,y) \approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)$$
$$+f(x+1,y+1) + f(x-1,y-1) + f(x+1,y-1) + f(x-1,y+1) - 8f(x,y),$$
$$w = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & -8 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}.$$

We notice that both Laplacian masks w have the sum of coefficients equal to 0, which is related with the sharpening property of the filter. We will now show how the Laplacian can be used to enhance images.

Consider a smoothed 1D edge profile f as shown in Figure 1 left, where the sign of f'' is also given. As we can see, applying the operation $g = f - f''$ as shown in Figure 1 right, produces an ideal sharp edge.

Similarly, for two dimensional images, if the input f is blurry, it can be enhanced by the operation $g(x,y) = f(x,y) - \triangle f(x,y)$. In the discrete case, if we use the 5-point Laplacian (4), this corresponds to the following spatial linear filter and mask:

$$g(x,y) = f(x,y) - \triangle f(x,y) = 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1), \tag{5}$$

$$w = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}.$$

Similarly, the enhancement using the 9-point Laplacian has the mask $w = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 9 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$.

Unsharp Masking and Highboost Filtering Image enhancement can also be obtained using smoothing spatial filters, instead of the Laplacian. Assume that f is the input image (that may be blurry). We first compute a smoother image f_{smooth} by applying a smoothing linear filter to f as discussed before. Then the sharper output image g is defined by

$$g(x, y) = f(x, y) + k \cdot \Big(f(x, y) - f_{smooth}(x, y)\Big), \tag{6}$$

where $k \geq 0$ is a coefficient. When $k = 1$ the method is called unsharp masking. When $k > 1$, it is called highboost filtering, since the edges and details are even more highlighted.

Consider the case $k = 1$ in equation (6). When we compute $f - f_{smooth}$, this gives negative values on the left of the edge, and positive values on the right of the edge, and values close to 0 elsewhere. Thus when we compute $f + (f - f_{smooth})$ this provides the sharper corrected edge in g.

4 The Fourier Transform and Filtering in the Frequency Domain

Illustration of Fourier series Consider the inner product vector space of functions

$$V = \{f : [0, 2\pi] \to C,\ f \text{ continuous}\}$$

with corresponding inner product

$$\langle f, g\rangle = \frac{1}{2\pi}\int_0^{2\pi} f(x)\overline{g(x)}dx.$$

Let $S = \{f_n(x) = e^{inx} = \cos nx + i\sin nx,\ n \text{ integer}\}$. We have that S is an orthonormal basis of V, based on the following:

If $m \neq n$, both integers, we have

$$\langle f_m, f_n\rangle = \frac{1}{2\pi}\int_0^{2\pi} e^{imx}\overline{e^{inx}}dx = \frac{1}{2\pi}\int_0^{2\pi} e^{i(m-n)x}dx = \frac{1}{2\pi i(m-n)}e^{i(m-n)x}\Big|_0^{2\pi} = 0.$$

On the other hand,

$$\langle f_n, f_n\rangle = \frac{1}{2\pi}\int_0^{2\pi} f_n(x)\overline{f_n(x)}dx = \frac{1}{2\pi}\int_0^{2\pi}(e^{inx}e^{-inx})dx = \frac{1}{2\pi}\int_0^{2\pi} 1dx = 1.$$

It is also easy to verify that for any $f \in V$,

$$f(x) = \sum_{n=-\infty}^{n=+\infty} c_n e^{inx},$$

where $c_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}dx$. Thus the function f can be expressed as a sum of sines and and cosines with appropriate coefficients.

The continuous and discrete Fourier transforms Remember that a linear discrete filter applied to an image $f(x, y)$ was defined by

$$g(x, y) = \sum_{m=-a}^{a}\sum_{n=-b}^{b} h(m, n)f(x + m, y + n),$$

where h was a window mask. In the continuous case, if h, f are functions in the plane, we can define the *correlation* of two functions (as a continuous version of the spatial linear filter)

$$h \circ f(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(m, n)f(x + m, y + n)dmdn. \tag{7}$$

If we make the change in the above formula (7): + signs into - signs inside f, we obtain the *convolution* of two functions defined by

$$h * f(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(m, n)f(x - m, y - n)dmdn. \tag{8}$$

Thus the convolution $h * f$ can also be seen as a linear filtering in the spatial domain: filtering the image f with convolution kernel h. For the discrete convolution, the computation $h * f$ is very expensive, especially if the support of h is not too small. It turns out that using the Fourier Transform $\mathcal{F}$ that we will define, we have:

if $H = \mathcal{F}(h)$ and $F = \mathcal{F}(f)$, then $h * f \Leftrightarrow HF$ (pointwise multiplication),
thus a fast computation.

We recall the Euler's formulas: if θ is a real number, then $e^{i\theta} = \cos\theta + i\sin\theta$. If α, β are real numbers, then $e^{\alpha+i\beta} = e^{\alpha}(\cos\beta + i\sin\beta)$, where i is the purely complex number such that $i^2 = -1$.

In one dimension first, we consider a planar function $f(x)$ satisfying $\int_{-\infty}^{\infty} |f(x)|dx < \infty$. Then we can define $F : R \to C$, the Fourier transform of f, by

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi iux}dx, \quad \mathcal{F}(f) = F,$$

and its inverse can be obtained by

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{2\pi iux}du, \quad \mathcal{F}^{-1}(F) = f$$

(x is called spatial variable, while u is called frequency variable).

In two dimensions, the corresponding direct and inverse Fourier transforms for a function $f(x, y)$ are

$$F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)e^{-2\pi i(ux+vy)}dxdy, \quad \mathcal{F}(f) = F,$$

and its inverse can be obtained by

$$f(x, y) = \int_{-\infty}^{\infty} F(u, v)e^{2\pi i(ux+vy)}dudv, \quad \mathcal{F}^{-1}(F) = f.$$

We say that $[f, F]$ forms a transform pair. It is easy to verify that the mapping $f \mapsto \mathcal{F}(f)$ is a linear transformation.

Since the transform F takes complex values, we have $F(u, v) = R(u, v) + iI(u, v)$ (using the real and imaginary parts of F). We call the *Fourier spectrum* $|F(u, v)| = \sqrt{R(u, v)^2 + I(u, v)^2}$.

We have the convolution theorem, that we prove in one dimension:

Convolution Theorem If $H = \mathcal{F}(h)$ and $F = \mathcal{F}(f)$, then $\mathcal{F}(h * f) = HF$, or $\mathcal{F}(h * f) = \mathcal{F}(h)\mathcal{F}(f)$.

Proof: We present the details of the proof in one dimension. The two-dimensional case is similar.

Recall the 1D convolution

$$h * f(x) = \int_{-\infty}^{\infty} h(m)f(x - m)dm.$$

Then

$$\mathcal{F}(h * f) = \int_{-\infty}^{+\infty} h * f(x)e^{-2\pi iux}dx = \int_{-\infty}^{\infty} \Big[\int_{-\infty}^{\infty} h(m)f(x - m)dm\Big]e^{-2\pi iux}dx$$

$$= \int_{-\infty}^{\infty} h(m)\Big[\int_{-\infty}^{\infty} f(x - m)e^{-2\pi iux}dx\Big]dm.$$

By the change of variable $x - m = X$, we obtain

$$\mathcal{F}(h * f) = \int_{-\infty}^{\infty} h(m)\Big[\int_{-\infty}^{\infty} f(X)e^{-2\pi iu(m+X)}dX\Big]dm$$

$$= \Big[\int_{-\infty}^{\infty} h(m)e^{-2\pi ium}dm\Big]\Big[\int_{-\infty}^{\infty} f(X)e^{-2\pi iuX}dX\Big] = H(u)F(u).$$

In the discrete case, we substitute the real numbers x, y, u, v by $x\triangle x$, $y\triangle y$, $u\triangle u$, $v\triangle v$ where now x, y, u, v are integers and $\triangle x, \triangle y, \triangle u, \triangle v$ here denote step discretizations. We make the convention that $\triangle x\triangle u = \frac{1}{M}$, $\triangle y\triangle v = \frac{1}{N}$. Then for an image $f(x, y)$, with $x = 0, 1, ..., M - 1$, $y = 0, 1, ..., N - 1$, the two-dimensional discrete Fourier transform (2D DFT) and its inverse (2D IDFT) are defined by

$$F(u, v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^{-2\pi i(\frac{ux}{M}+\frac{vy}{N})}, \quad u = 0, 1, 2..., M - 1,\ v = 0, 1, ..., N - 1$$

and

$$f(x, y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u, v)e^{2\pi i(\frac{ux}{M}+\frac{vy}{N})}, \quad x = 0, 1, 2..., M - 1,\ y = 0, 1, ..., N - 1.$$

(the one-dimensional versions of the discrete transforms are defined similarly).

Note that

$$F(0, 0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^0 = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y) = (MN)average(f),$$

thus the average of f can be recovered from $F(0, 0)$:

$$average(f) = \frac{F(0, 0)}{MN}.$$

Shifting the center of the transform: We have the formula $\mathcal{F}(f(x, y)(-1)^{x+y}) = F(u-\frac{M}{2}, v-\frac{N}{2})$. Indeed, note that we have

$$(-1)^{x+y} = e^{i\pi(x+y)} = \cos(\pi(x + y)) + i\sin(\pi(x + y)).$$

Then

$$\mathcal{F}(f(x, y)(-1)^{x+y}) = \sum_{x=0}^{M}\sum_{y=0}^{N}\Big[f(x, y)e^{i\pi(x+y)}\Big]e^{-2\pi i(\frac{ux}{M}+\frac{vy}{N})}$$

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\Big[f(x, y)e^{-2\pi i(\frac{-xM}{2M}-\frac{yN}{2N})}\Big]e^{-2\pi i(\frac{ux}{M}+\frac{vy}{N})}$$

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\Big[f(x, y)e^{-2\pi i(x\frac{u-\frac{M}{2}}{M}+y\frac{v-\frac{N}{2}}{N})}\Big] = F(u - \frac{M}{2}, v - \frac{N}{2}).$$

From now on, we will assume that we work with the centered spectrum. Please see the handout posted on the course webpage as an illustration of the shifting of the origin of the transform 0 to $M/2$, in one dimension.

4.1 Principles of Filtering in the Frequency Domain

We assume that we work with the shifted transform based on the above property, thus the origin of the transform $(0, 0)$ has been moved to the center of the domain $(M/2, N/2)$.

Usually it is impossible to make direct associations between the components of an image f and its transform F. However, we can make connections between the frequency components of the transform and spatial features of the image. We have

• frequency is directly related to spatial rates of change

• we have seen that the slowest varying frequency component ($(u = 0, v = 0)$ before centering or $(u = M/2, v = N/2)$ after centering) is proportional to the average of the image intensity f (no variation)

• as we move away from the origin (or from the center $(u = M/2, v = N/2)$ after shifting), the low frequency components correspond to slowly varying intensity components in the image

• as we move farther away from the origin (or from the center $(u = M/2, v = N/2)$ after shifting), high frequency components correspond to faster gray-level changes

Basics of filtering in the frequency domain 1. Multiply $f(x, y)$ by $(-1)^{x+y}$

2. Compute $F(u, v) = DFT\Big(f(x, y)(-1)^{x+y}\Big)$ (here $F(u, v)$ is in fact $F(u - M/2, v - N/2)$, but we keep the simpler notation $F(u, v)$)

3. Multiply F by a real "filter" function H: $G(u, v) = H(u, v)F(u, v)$ (pointwise multiplication, not matrix multiplication)

4. Compute the Discrete IFT of G (of the result in 3.)

5. Take the real part of the result in 4 (in order to ignore parasitic complex components due to computational accuracies)

6. Multiply the result in 5 by $(-1)^{x+y}$ to obtained the filtered image g

H will suppress certain frequencies in the transform, while leaving other frequencies unchanged.

Let h be such that $\mathcal{F}(h) = H$. Due to the Convolution Theorem, we have that linear filtering in the spatial domain with h is equivalent with linear filtering in the frequency domain with filter H: $h * f \Leftrightarrow HF$

As we have mentioned, the low frequencies of the transform correspond to smooth regions in the image (homogeneous parts). High frequencies of the transform correspond to edges, noise, details.

A low-pass filter (LPF) H leaves low frequencies unchanged, while attenuating the high frequencies. This is a smoothing filter.

A high-pass filter (HPF) H leaves high frequencies unchanged, while attenuating the low frequencies. This is a sharpening filter.

Let $D(u, v) = dist\Big((u, v), (M/2, N/2)\Big) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$ the distance from the frequency (u, v) to the center of the domain. Let $D_0 \geq 0$ be a parameter.

Examples of low-pass filters:

• Ideal low-pass filter $H(u, v) = \begin{cases} 1 \text{ if } D(u, v) \leq D_0 \\ 0 \text{ if } D(u, v) > D_0 \end{cases}$. This filter produces the so-called ringing effect or Gibbs effect (oscillations near boundaries and edges), due to the sharp transition between 0 and 1.

The following two filters produce less ringing effect.

• Let n a positive parameter. The Butterworh low-pass filter of order n $H(u,v) = \frac{1}{1+(D(u,v)/D_0)^{2n}}$.

• The Gaussian low-pass filter $H(u,v) = e^{-D^(u,v)/2D_0^2}$

Examples of high-pass filters:

• Ideal high-pass filter $H(u,v) = \begin{cases} 0 \text{ if } D(u,v) \leq D_0 \\ 1 \text{ if } D(u,v) > D_0 \end{cases}$. This filter produces the so-called ringing effect or Gibbs effect (oscillations near boundaries and edges), due to the sharp transition between 0 and 1.

The following two filters produce less ringing effect.

• Let n a positive parameter. The Butterworh high-pass filter of order n is $H(u,v) = \frac{1}{1+(D_0/D(u,v))^{2n}}$.

• The Gaussian low-pass filter $H(u,v) = 1 - e^{-D^(u,v)/2D_0^2}$

The Laplacian in the frequency domain Assuming that $f(x)$ is a one-dimensional function defined on the real line in the continuous case, and that f has partial derivatives up to order n, we have the following formula that expresses the Fourier Transform of the derivative $\mathcal{F}\Big(\frac{\partial^n f}{\partial x^n}\Big)$ function of the Fourier Transform of f:

$$\mathcal{F}\Big(\frac{\partial^n f}{\partial x^n}\Big) = (2\pi i u)^n F(u). \tag{9}$$

To prove this formula, consider the IFT

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{2\pi i u x} du,$$

then differentiating both sides with respect to x (a parameter under the integral sign), we obtain

$$\frac{\partial f}{\partial x} = \int_{-\infty}^{\infty} \Big[F(u)(2\pi i u)\Big] e^{2\pi i u x} du,$$

so by the uniqueness of the Inverse Fourier Transform, we must have

$$\frac{\partial f}{\partial x} = (2\pi i u) F(u) du,$$

thus the above formula for $n = 1$. Repeating the process, we deduce formula (9) for any derivative of order n.

Now we apply formula (9) with $n = 2$ in each variable x and y to the Laplacian in two dimensions, using also the linearity of the Fourier transform:

$$\mathcal{F}(\triangle f) = \mathcal{F}\Big(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\Big) = \mathcal{F}\Big(\frac{\partial^2 f}{\partial x^2}\Big) + \mathcal{F}\Big(\frac{\partial^2 f}{\partial y^2}\Big)$$

$$= (2\pi i u)^2 F(u,v) + (2\pi i v)^2 F(u,v) = -4\pi^2 (u^2 + v^2) F(u,v)$$

thus we see that the computation of the Laplacian can be done by the filter

$$H_{lapl}(u,v) = -4\pi^2 (u^2 + v^2).$$

Due to the shifting property of the center, we define H_{lapl} as

$$H(u,v) = -4\pi^2\Big[(u-\frac{M}{2})^2 + (v-\frac{N}{2})^2\Big].$$

Enhancement using the Laplacian in the frequency domain

Let f be the input image, and g be the output enhanced image obtained by $g = f - \triangle f$. This operation can also be performed in the frequency domain, using the linearity of the transform and the above Laplacian filter H_{lapl}:

$$\begin{aligned}\mathcal{F}(f-\triangle f) &= \mathcal{F}(f) - \mathcal{F}(\triangle f) = F(u,v) + 4\pi^2(u^2+v^2)F(u,v)\\ &= (1+4\pi^2(u^2+v^2))F(u,v) = (1-H_{lapl}(u,v))F(u,v).\end{aligned}$$

Using the shifting property of the center, we obtain enhancement using the Laplacian in the Frequency domain by the following steps:

1. $F(u,v) = 2DFT(f(x,y)(-1)^{x+y})$
2. $H_{enhance}(u,v) = 1 + 4\pi^2\Big((u-M/2)^2 + (v-N/2)^2\Big)$
3. $G(u,v) = H(u,v)F(u,v)$
4. $g(x,y) = \Big[\text{Re}\Big(2DIFT(G)\Big)\Big](-1)^{x+y}$

Remark: Note that, in the spatial domain, images f and $\triangle f$ had comparable values (no additional rescaling was necessary to apply $g = f - \triangle f$). However, in the frequency domain, $\mathcal{F}(\triangle f)$ introduces DFT scaling factors that can be of several orders of magnitude larger than the maximum of f. Thus rescaling of f or of $\triangle f$ has to be appropriately introduced (e.g., normalizing f and $\triangle f$ between $[0,1]$).

Unsharp masking and highboost filtering in the frequency domain Let f be the input image, to be made sharper. The equivalent of the unsharp masking and highboost filtering techniques $g = f + k(f - f_{smooth})$ in the Frequency domain is as follows. The component f_{smooth} can be obtained by applying a low-pass filter to f, using a filter function H_{LP}. Then, using the linearity of the Fourier transform, we have

$$\begin{aligned}g = \mathcal{F}^{-1}\Big[F(u,v) + k\Big(F(u,v) - H_{LP}(u,v)F(u,v)\Big)\Big] &= \mathcal{F}^{-1}\Big[\Big(1+k(1-H_{LP}(u,v))\Big)F(u,v)\Big]\\ &= \mathcal{F}^{-1}\Big[\Big(1+kH_{HP}(u,v)\Big)F(u,v)\Big].\end{aligned}$$

Recall that taking $k = 1$ represents "unsharp masking" and $k > 1$ represents "highboost filtering".

The frequency domain filter $1 + kH_{HP}(u,v)$ is called "high-frequency emphasis filter".

A more general filter can be obtained by

$$g = \mathcal{F}^{-1}\Big[\Big(k_1 + k_2H_{HP}(u,v)\Big)F(u,v)\Big],$$

with $k_1 \geq 0$ and $k_2 \geq 0$.

Final remarks:

- The direct and inverse discrete Fourier transforms are computed using the Fast Fourier Transform (FFT) algorithm.

- Additional properties of the discrete and Fourier transforms are given in the assigned exercises.

- Additional applications of filtering in the Fourier domain will be seen in the sections on Image Reconstruction and Image Segmentation.

5 Image Restoration

Image restoration is the process of recovering an image that has been degraded, using a-priori knowledge of the degradation process.

Degradation model: Let $f(x, y)$ be the true ideal image that we wish to recover, and let $g(x, y)$ be its degraded version. One relation that links f to g is the degradation model

$$g(x, y) = H\Big[f\Big](x, y) + n(x, y),$$

where H denotes a degradation operator (e.g. blur) and n is additive noise.

Inverse Problem: Knowing the degradation operator H and statistics of the noise n, find a good estimate $\hat{f}$ of f.

5.1 Image Denoising

We assume that the degradation operator is the identity, thus we only deal with denoising in this subsection. The linear degradation model becomes $g(x, y) = f(x, y) + n(x, y)$. Note that not all types of noise are addtitive.

Random noise We assume here that the noise intensity levels can be seen as a random variable, with associated histogram or probability distribution function (PDF) denoted by $p(r)$. We also assume that the noise n is independent of the image f and independent of spatial coordinates (x, y).

Most common types of noise

• The Gaussian noise (additive) with associated PDF given by $p(r) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(r-\mu)^2/2\sigma^2}$, where μ is the mean and σ denotes the standard deviation.

• Uniform noise (additive) with associated PDF given by $p(r) = \begin{cases} \frac{1}{B-A} \text{ if } A \leq r \leq B \\ 0 \text{ otherwise} \end{cases}$

• Impulse noise (salt-and-peper, or bipolar) (not additive), with associated PDF given by

$$p(r) = \begin{cases} p_A \text{ if } r = A \\ p_B \text{ if } r = B \\ 0 \text{ otherwise} \end{cases}$$

Mean filters for random noise removal

Let g be the input noisy image, and $\hat{f}$ be the output denoised image. Let $S_{(x,y)}$ be a neighborhood of the pixel (x, y) defined by

$$S_{(x,y)} = \Big\{(x + s, y + t),\ -a \leq s \leq a,\ -b \leq t \leq b\Big\},$$

of size mn, where $m = 2a + 1$ and $n = 2b + 1$ are positive integers.

• Arithmetic Mean Filter

$$\hat{f}(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{(x,y)}} g(s, t).$$

This filter is useful for removing Gaussian noise or uniform noise, but blurring is introduced.

• Geometric Mean Filter

$$\hat{f}(x,y) = \Big(\Pi_{(s,t)\in S_{(x,y)}} g(s,t)\Big)^{1/mn}.$$

This filter introduces blurring comparable with the arithmetic mean filter, but tends to lose less image detail in the process.

• Contraharmonic Mean Filter of order Q

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{(x,y)}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{(x,y)}} g(s,t)^{Q}}.$$

Here, the parameter Q gives the order of the filter.

$Q = 0$: reduces to the arithmetic mean filter.

$Q = -1$: it is called the harmonic mean filter (works well for salt noise, Gaussian noise; fails for pepper noise).

$Q > 0$: useful for removing pepper noise

$Q < 0$: useful for removing salt noise.

Needs to know the sign to be used, it cannot remove salt noise and pepper noise simultaneously.

As we can see, we still need a filter that could remove both salt and pepper noise simultaneously.

• Median Filter is an "order statistics filter", where $\hat{f}(x,y)$ depends on the ordering of pixel values of g in the window $S_{(x,y)}$. The median filter output is the 50% ranking of the ordered values:

$$\hat{f}(x,y) = median\Big\{g(s,t),\ (s,t) \in S_{(x,y)}\Big\}.$$

For example, for a 3×3 median filter, if $g_{S_{(x,y)}} = \begin{array}{|c|c|c|}\hline 1 & 5 & 20 \\ \hline 200 & 5 & 25 \\ \hline 25 & 9 & 100 \\ \hline \end{array}$ then we first order the values in the window as follows: 1, 5, 5, 9, 20, 25, 25, 100, 200. The 50% ranking (5th value here) is 20, thus $\hat{f}(x,y) = 20$.

The median filter introduces much less blurring than the other filters of the same window size. It can be used for salt noise, pepper noise, or salt-and-pepper noise. Note that the median operation is not a linear operation.

The following two filters can be seen as combinations of order statistics filters and averaging filters, and can be used for several types of noise.

• Midpoint Filter

$$\hat{f}(x,y) = \frac{1}{2}\Big[\max_{(s,t)\in S_{(x,y)}}\{g(s,t)\} + \min_{(s,t)\in S_{(x,y)}}\{g(s,t)\}\Big].$$

This filter works for randomly distributed noise, such as Gaussian noise or uniform noise.

• Alpha-trimmed mean filter

Let $d \geq 0$ be a even integer, such that $0 \leq d \leq mn-1$. We first order again the mn pixel values of the input image g in the window $S_{(x,y)}$, and then we remove the lowest $d/2$ and the largest $d/2$. We denote the remaining $mn-d$ values by g_r.

$$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{(x,y)}} g_r(s,t),$$

but in the sum only the remaining values are used.

$d = 0$ reduces to the arithmetic mean filter.

$d = \frac{mn-1}{2}$ reduces to the median filter.

This filter is useful for multiple types of noise (Gaussian noise, uniform noise, salt-and-pepper noise).

Note that, when the noise is stronger, the above methods require the use of larger window $S_{(x,y)}$; however, larger window will introduce more blurring. There are adaptive filters based on the previous methods, that are slightly more complex in design, but producing improved results with less blurring effect (for example, see "adaptive, local noise reduction filter" and the "adaptive median filter" in [7]).

• The Nonlocal Means Filter

Let g be the input image.

- For two different pixels $X = (x,y)$ and $X' = (x',y')$ of the image g, we consider their corresponding neighborhoods

$$S_X = \{(x+s, y+t),\ -a \leq s,t \leq a\}, \quad S_{X'} = \{(x'+s, y'+t),\ -a \leq s,t \leq a\},$$

thus these two neighborhoods have the same size m^2 and the same shape, with $m = 2a+1$.

- We consider two small images, as two patches $g_X = g|_{S_X}$ and $g_{X'} = g|_{S_{X'}}$. Thus each g_X and $g_{X'}$ is a image of size $m \times m$ (in other words, g_X is the patch from the image g around X and $g_{X'}$ is the patch from the image g around X').

- We apply a smoothing Gaussian filter to g_X and $g_{X'}$ obtaining smoothed versions $\tilde{g}_X$, $\tilde{g}_{X'}$, again each of size $m \times m$.

- Compute the norm distance between the two image patches $\tilde{g}_X$, $\tilde{g}_{X'}$:

$$\|\tilde{g}_X - \tilde{g}_{X'}\|^2 = \text{ sum of squares of coefficients of the matrix } \tilde{g}_X - \tilde{g}_{X'}$$

- For each two pixels X and X', define the weight $w(X,X') = e^{-\frac{\|\tilde{g}_X - \tilde{g}_{X'}\|^2}{h^2}}$, where h is the noise level parameter.

- Define the output filtered and denoised image $\hat{f}$ at pixel X by

$$\hat{f}(X) = \frac{\sum_{X'\in \text{image domain}} w(X,X')g(X')}{\sum_{X'\in \text{image domain}} w(X,X')}.$$

Remember that the normalized weighted linear filter with window mask w computes a weighted average over a spatial neighborhood of (x,y). Here, the NL Means filter computes a weighted average over an intensity neighborhood of $g(X)$. In the above formula, $\hat{f}(X)$ is a weighted average of all values $g(X')$, for which the patch(X) is similar to the patch(X').

This method can be tested from the website:

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

Periodic (deterministic) noise We assume here that the noise $n(x, y)$ depends on the spatial coordinates (x, y) and it may be due to electrical or electromechanical interference. An example of periodic noise is $n(x, y) = \alpha \cos(x + y) + \beta \sin(x + y)$.

We define a unit impulse (or delta function) by

$$\delta(u, v) = \begin{cases} 1 \text{ if } (u, v) = (0, 0) \\ 0 \text{ otherwise} \end{cases}$$

If we visualize δ as an image, it will be a bright white dot at $(0, 0)$ and black otherwise.

Let u_0 and v_0 be two parameters. It is possible to prove the following formula in the discrete case

$$\mathcal{F}\Big(\sin(2\pi u_0 x + 2\pi v_0 y)\Big) = \frac{i}{2}\Big[\delta(u + Mu_0, v + Nv_0) - \delta(u - Mu_0, v - Nv_0)\Big].$$

Thus, if we visualize the Fourier spectrum of the "noisy" image $g(x, y) = f(x, y) + \alpha n(x, y)$, with $n(x, y) = \sin(2\pi u_0 x + 2\pi v_0 y)$ as periodic noise, we would notice two bright regions, symmetrically located, corresponding to the two impulse functions appearing in the above formula.

Thus in order to remove such periodic noise, we define two small rectangles or disks A and B, each including the two bright regions, and a filter $H(u, v)$ in the frequency domain, such that

$$H(u, v) = \begin{cases} 0 \text{ if } (u, v) \in A \cup B \\ 1 \text{ otherwise} \end{cases} .$$

This type of filter is called a notch filter. Note that Gaussian-type filters, or Butterworth type filters with smoother transition between 0 and 1 can also be adapted to this application. The filtered image $\hat{f}$ should have the periodic pattern removed, while most of the image details f kept.

Note that there is a improved version of the notch filter for removing periodic noise, called "optimum notch filter" [7].

5.2 Image Deblurring

Linear and position invariant degradation functions Recall the degradation model $g = H[f] + n$, where H is a degradation operator and n is additive noise. Assume first that $n(x, y) = 0$ (thus there is no noise). We need the following definitions and properties on H:

- we assume that H is linear: let f_1 and f_2 be two functions, then

$$H[f_1 + f_2] = H[f_1] + H[f_2] \text{ (additivity)} , \quad H[\alpha f] = \alpha H[f] \text{ (scalar multiplication)} .$$

- we assume that the additivity property is extended to integrals

- we assume that H is position invariant: if $g(x, y) = H[f(x, y)]$, then $g(x - \alpha, y - \beta) = H[f(x - \alpha, y - \beta)]$ for all x, y, α, β and all functions f and g.

- we define a continuous impulse function δ, such that (by definition) $f * \delta = f$ for any other function f. In other words, δ is defined by $\delta(x, y) = \begin{cases} \infty \text{ if } (x, y) = (0, 0) \\ 0 \text{ otherwise} \end{cases}$ and the property $f * \delta = f$ becomes

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta)\delta(x - \alpha, y - \beta)d\alpha d\beta. \tag{10}$$

We prove that under the above assumptions and notations, the operator H is a convolution.

Proof. We have $g(x, y) = H[f(x, y)]$, thus according to (10) we have

$$g(x, y) = H\Big[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta)\delta(x - \alpha, y - \beta)d\alpha d\beta\Big],$$

then extending the additivity property to integrals, we obtain

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} H\Big[f(\alpha, \beta)\delta(x - \alpha, y - \beta)\Big]d\alpha d\beta.$$

Now $f(\alpha, \beta)$ is constant with respect to (x, y) thus by the scalar multiplication property, we obtain

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta)H\Big[\delta(x - \alpha, y - \beta)\Big]d\alpha d\beta.$$

Denote by $h(x, y) = H[\delta(x, y)]$, thus by the position invariance property of H, we have $h(x - \alpha, y - \beta) = H[\delta(x - \alpha, y - \beta)]$ and $g(x, y)$ becomes

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta)h(x - \alpha, y - \beta)d\alpha d\beta,$$

in other words, $g(x, y) = f * h(x, y)$ by the definition of the convolution. Since $f * h = h * f$, we finally obtain that $g(x, y) = H[f(x, y)] = h * f(x, y)$.

In the presence of additive noise, the (linear) degradation model becomes $g(x, y) = h * f(x, y) + n(x, y)$, or in the frequency domain, $G(u, v) = H(u, v)F(u, v) + N(u, v)$.

The convolution kernel h is called a point-spread-function (PSF), since if it is applied to a (sharp) impulse (white dot over black background), the result is a spreading out of the white dot (instead of seeing a bright dot over a black background, we see a larger and diffuse white disk over the black background).

The process of recovering f from g is now called denoising-deblurring or denoising-deconvolution. The deconvolution process is difficult, especially in the presence of noise, since this leads to a highly ill-posed problem.

Examples of the degradation functions

We give below a few examples of linear degradation functions h or H, obtained by modeling the physical phenomenon for the image acquisition process.

• Atmospheric Turbulence Blur

In the frequency domain, this is defined by $H(u,v) = e^{-k(u^2+v^2)^{5/6}}$, where k is a constant depending on the degree of turbulence:

$k = 0.0025$ corresponds to severe turbulence

$k = 0.001$ corresponds to mild turbulence

$k = 0.00025$ corresponds to low turbulence

A Gaussian low-pass filter can also be used to model atmospheric turbulence blur.

• Out of Focus Blur

In the spatial domain, this is defined by $h(x,y) = \begin{cases} 1 \text{ if } x^2+y^2 \leq D_0^2 \\ 0 \text{ otherwise} \end{cases}$

(larger parameter D_0 gives more severe blur).

• Uniform Linear Motion Blur

Assume that the image $f(x,y)$ undergoes planar motion during the acquisition. Let $(x_0(t), y_0(t))$ be the motion components in the x and y-directions. Here, t denotes time and T is the duration of the exposure. The motion blur degradation is

$$g(x,y) = \int_0^T f(x - x_0(t), y - y_0(t))dt.$$

We want to express the motion blur in the frequency domain. Let $G = \mathcal{F}(g)$, thus by the definition,

$$G(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)e^{-2\pi i(ux+vy)}dxdy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\Big[\int_0^T f(x - x_0(t), y - y_0(t))dt\Big]e^{-2\pi i(ux+vy)}dxdy$$

$$= \int_0^T\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x - x_0(t), y - y_0(t))e^{-2\pi i(ux+vy)}dxdy\Big]dt.$$

Using now the property $\mathcal{F}(f(x - x_0, y - y_0)) = F(u,v)e^{-2\pi i(ux_0+vy_0)}$, we obtain

$$G(u,v) = \int_0^T \Big[F(u,v)e^{-2\pi i(ux_0(t)+vy_0(t))}\Big]dt = F(u,v)\int_0^T \Big[e^{-2\pi i(ux_0(t)+vy_0(t))}\Big]dt,$$

thus $G(u,v) = H(u,v)F(u,v)$ with the motion degradation function in the frequency domain

$$H(u,v) = \int_0^T \Big[e^{-2\pi i(ux_0(t)+vy_0(t))}\Big]dt.$$

• Radon Transform Degradation in Computerized Tomography

In computerized tomography, parallel beams of X-rays are sent through the body. Energy is absorbed and a sensor measures the attenuated values along each beam. Thus the "image" $f(x,y)$ that we have to recover is an absorption image function of the tissue (each tissue absorbs the

energy or the radiation in a different way). The data g is given in the Radon domain, and the relation between f and g can be defined by

$$g(\rho,\theta) = \mathcal{R}(f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy,$$

or that

$$g(\rho,\theta) = \int_{linex\cos\theta+y\sin\theta=\rho} f,$$

where (ρ,θ) are the parameters describing a line. The image $g(\rho,\theta)$ for $0 \le \rho \le r$ and $0 \le \theta \le \pi$ is called a sinogram. The problem is to recover f from g. In reality, we only have a finite (limited) number of "projections" (lines), and $\mathcal{R}(f)$ is called the Radon transform of f.

Image reconstruction from blurry-noisy data We will discuss several models for image deconvolution (with or without noise).

The simplest approach is called *direct inverse filtering*. In the ideal case without noise, $G(u,v) = H(u,v)F(u,v)$, thus to recover F, we could define $\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$. However, this approach has at least two problems:

(i) in reality there is unknown noise N, thus the real relation would be $\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} = F(u,v) + \frac{N(u,v)}{H(u,v)}$. So even if we know the degradation function $H(u,v)$, we cannot recover F exactly because $N(u,v)$ is unknown.

(ii) The blurring filter $H(u,v)$ usually is zero or it has very small values (close to zero) away from the origin (or away from the center of the domain). Thus the division by $H(u,v)$ would not be defined at many values (u,v), or the term $N(u,v)/H(u,v)$ would dominate and lead to incorrect reconstruction.

The Wiener Filter is given by

$$\hat{F}(u,v) = \Big[\Big(\frac{1}{H(u,v)}\Big)\Big(\frac{|H(u,v)|^2}{K+|H(u,v)|^2}\Big)\Big]G(u,v),$$

where $K > 0$ is a positive parameter. K is chosen so that visually good results are obtained. This filter can be interpreted in the following way:

(i) First, the factor $\frac{|H(u,v)|^2}{K+|H(u,v)|^2}$ acts like a low-pass filter to remove the noise.

(ii) Then, the factor $\frac{1}{H(u,v)}$ acts like in the direct inverse filtering (direct deconvolution). But we no longer have division by very small values, since the filter is equivalent with

$$\hat{F}(u,v) = \Big[\frac{\overline{H(u,v)}}{K+|H(u,v)|^2}\Big]G(u,v).$$

Error measures Assume that we perform an artifficial experiment, thus we know the true image f that we wish to recover. Once we have a denoised or deblurred image $\hat{f}$, we can measure in several ways, how good is the restored result $\hat{f}$. We give two examples of measures:

- Root-Mean-Square-Error (RMSE) $RMSE = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f(x,y)-\hat{f}(x,y)]^2}$.
- Signal-to-Noise-Ratio (SNR) $SNR = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f(x,y)-\hat{f}(x,y)]^2}$

5.3 Energy minimization methods for image reconstruction

Consider first the denoising problem for simplicity. Let g be the given noisy image, f the image to be restored, linked through the relation $g(x,y) = f(x,y) + n(x,y)$.

Let's recall for a moment the enhancement method using the Laplacian: $g = f - \triangle f$, where f was the input blurry image and g was the sharper output image. However, to remove noise, we need the opposite process of smoothing. So let's invert the relation $g = f - \triangle f$ to obtain

$$f = g + \triangle f, \tag{11}$$

where g is the input noisy image and f is the denoised processed image. This is a linear partial differential equation, with unknown f. There are many ways to solve (11).

• One way is using the Fourier transform: (11) is equivalent with

$$\mathcal{F}(f) = \mathcal{F}(g + \triangle f),$$

thus by the linearity of the Fourier transform, we first have

$$\mathcal{F}(f) = \mathcal{F}(g) + \mathcal{F}(\triangle f),$$

or

$$F(u,v) = G(u,v) + \Big[-4\pi^2(u^2+v^2)\Big]F(u,v),$$

therefore

$$F(u,v) = \frac{1}{1+4\pi^2(u^2+v^2)}G(u,v),$$

and we notice that $\frac{1}{1+4\pi^2(u^2+v^2)}$ acts as a low-pass filter.

• Another way is by discretization in the spatial domain (assuming that f is extended by reflection outside of its domain): a discrete version of equation (11) can be obtained using the 5-point Laplacian and with (x,y) integer coordinates,

$$f(x,y) = g(x,y) + \Big[f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y)\Big], \ \forall (x,y). \tag{12}$$

Equation (12) can be solved by direct methods or iterative methods for linear systems. As an example of iterative method, the simplest one is as follows: let $g(x,y)$ be the given noisy data, defined for $x = 1, ..., M$ and $y = 1, ..., N$.

(i) Start with an initial guess $f^0(x,y)$ defined for all (x,y) in the original domain $x = 1, ..., M$ and $y = 1, ..., N$. Then extend f^0 by reflection to the domain $x = 0, ..., M+1$, $y = 0, ..., N+1$ (sometimes we chose $f^0 = g$).

(ii) For integer $n \geq 0$, and for all $x = 1, ..., M$, $y = 1, ..., N$:

$$f^{n+1}(x,y) = g(x,y) + \Big[f^n(x+1,y) + f^n(x-1,y) + f^n(x,y+1) + f^n(x,y-1) - 4f^n(x,y)\Big]$$

(iii) Impose boundary conditions by reflection:

Let $f^{n+1}(0,y) = f^{n+1}(2,y)$ and $f^{n+1}(M+1,y) = f^{n+1}(M-1,y)$ for all $y = 1, 2, ..., N$.

Let $f^{n+1}(x,0) = f^{n+1}(x,2)$ and $f^{n+1}(x,N+1) = f^{n+1}(x,N-1)$ for all $x = 1, 2, ..., M$.

Let $f^{n+1}(0,0) = f^{n+1}(2,2)$, $f^{n+1}(0,N+1) = f^{n+1}(2,N-1)$,

$f^{n+1}(M+1,0) = f^{n+1}(M-1,2)$, $f^{n+1}(M+1,N+1) = f^{n+1}(M-1,N-1)$.

(iv) Let $n = n+1$ and repeat steps (ii)-(iii) until convergence

(v) If matrix norm $\|f^{n+1} - f^n\| \leq tolerance$, stop and output $f = f^{n+1}$ on the domain $x = 1, ..., M$ and $y = 1, ..., N$.

We want to show now, that the solution f of (11) could also be obtained as a minimizer of the following discrete energy (taking into account the extension by reflection outside of the domain),

$$E(f) = \sum_{x=1}^{M}\sum_{y=1}^{N}[f(x,y) - g(x,y)]^2 + \sum_{x=1}^{M}\sum_{y=1}^{N}\Big[(f(x+1,y) - f(x,y))^2 + (f(x,y+1) - f(x,y))^2\Big]. \tag{13}$$

Indeed, assume now that some discrete image function f is a minimizer of E. Then, since E is differentiable, we must have $\frac{\partial E}{\partial(f(xy))} = 0$ for all points (x, y). Compute

$$\frac{\partial E}{\partial(f(xy))} = 2(f(x,y) - g(x,y)) + 2[f(x+1,y) - f(x,y)](-1) + 2[f(x,y+1) - f(x,y)](-1)$$

$$+2[f(x,y) - f(x-1,y)] + 2[f(x,y) - f(x,y-1)] = 0,$$

and rearranging the terms we obtain

$$f(x,y) = g(x,y) + \Big[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)\Big],$$

which is exactly equation (12).

We notice that, a continuous version of (13) is

$$E(f) = \int\int (f(x,y) - g(x,y))^2 dxdy + \int\int \Big[\Big(\frac{\partial f}{\partial x}\Big)^2 + \frac{\partial f}{\partial y}\Big)^2\Big] dxdy,$$

or

$$E(f) = \int\int (f(x,y) - g(x,y))^2 dxdy + \int\int |\nabla f|^2 dxdy,$$

where $\nabla f(x,y) = \Big(\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\Big)$ is the gradient of f and $|\nabla f|$ is the gradient magnitude.

In the above energy $E(f)$, the first term is called a data fidelity term, while the second term is a (isotropic) regularization (or isotropic smoother).

In this approach, if we use one of the equations (11) or (12) to denoise images, the noise will be smoothed out but the edges will also become blurry, because the Laplacian is a isotropic diffusion operator in all directions. To overcome this problem, we use the gradient $\nabla f(x,y) = \Big(\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\Big)$ as an edge indicator. We know that, near a edge of the image f, where there are strong variations, the gradient magnitude $|\nabla f|$ must be large; on the contrary, away from edges, where there are slow variations, the gradient magnitude $|\nabla f|$ is small. Therefore, we do not want to diffuse the image f where $|\nabla f|$ is large. We modify the PDE $f = g + \triangle f \Leftrightarrow f = g + \frac{\partial}{\partial x}\Big(\frac{\partial f}{\partial x}\Big) + \frac{\partial}{\partial y}\Big(\frac{\partial f}{\partial y}\Big)$ as

$$f = g + \frac{\partial}{\partial x}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial x}\Big) + \frac{\partial}{\partial y}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial y}\Big). \tag{14}$$

This is a anisotropic diffusion equation, that also comes from the following energy minimization:

$$\min_f \Big\{J(f) = \frac{1}{2}\int\int (f(x,y) - g(x,y))^2 dxdy + \int\int |\nabla f| dxdy\Big\}. \tag{15}$$

When we deal with both denoising-deblurring, we modify (15) into

$$J(f) = \frac{1}{2}\int\int (h * f(x,y) - g(x,y))^2 dxdy + \lambda\int\int |\nabla f| dxdy, \tag{16}$$

introduced by Rudin, Osher and Fatemi in [11, 12]. The first term in (16) is a data fidelity term, while the second term is called total variation regularization. The parameter $\lambda > 0$ is a weight between the fidelity term and the regularization term.

The fidelity term $\frac{1}{2}\int\int (h * f(x,y) - g(x,y))^2 dxdy$ is appropriate for additive Gaussian noise. If the image g is noisy due to additive Laplacian noise or due to impulse noise (salt-and-pepper noise), then the fidelity term is modified into the 1-norm $\int\int |h * f(x,y) - g(x,y))| dxdy$.

We want now to show that the following anisotropic PDE

$$\tilde{h} * h * f = \tilde{h} * g + \lambda \frac{\partial}{\partial x}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial x}\Big) + \lambda \frac{\partial}{\partial y}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial y}\Big) \tag{17}$$

must be formally satisfied by a minimizer of the convex energy J in (16), where $\tilde{h}(x,y) = h(-x,-y)$. To do this, we formally impose $\frac{\partial J}{\partial f} = 0$ which will be an equivalent relation with (17).

Consider the discrete case for the purpose of illustration. Assume that the given noisy-blurry image g is a matrix of size $M \times N$. We want to recover a denoised-deblurred image f as a matrix of size $M \times N$. We denote by $H[f] = h * f$ for simplicity, where H must be a linear operator, since $h * (f_1 + f_2) = h * f_1 + h * f_2$, which is easy to verify. A discrete version of J is

$$J(f) = \sum_{x=1}^{M}\sum_{y=1}^{N}\Big(H[f](x,y) - g(x,y)\Big)^2 + \lambda \sum_{x=1}^{M}\sum_{y=1}^{N}\sqrt{\Big(f(x+1,y) - f(x,y)\Big)^2 + \Big(f(x,y+1) - f(x,y)\Big)^2}, \tag{18}$$

assuming the space discretization steps $\triangle x = \triangle y = 1$, and x, y integers. Differentiating with respect to $f(x,y)$, for (x,y) fixed, we obtain

$$\begin{aligned}
\frac{\partial J}{\partial f(x,y)} &= H^T\Big(H[f] - g\Big)(x,y) \\
&+ \lambda \frac{2\Big(f(x+1,y) - f(x,y)\Big)(-1) + 2\Big(f(x,y+1) - f(x,y)\Big)(-1)}{2\sqrt{\Big(f(x+1,y) - f(x,y)\Big)^2 + \Big(f(x,y+1) - f(x,y)\Big)^2}} \\
&+ \lambda \frac{2\Big(f(x,y) - f(x-1,y)\Big)}{2\sqrt{\Big(f(x,y) - f(x-1,y)\Big)^2 + \Big(f(x-1,y+1) - f(x-1,y)\Big)^2}} \\
&+ \lambda \frac{2\Big(f(x,y) - f(x,y-1)\Big)}{2\sqrt{\Big(f(x+1,y-1) - f(x,y-1)\Big)^2 + \Big(f(x,y) - f(x,y-1)\Big)^2}} = 0.
\end{aligned}$$

After simplifications, we obtain

$$\begin{aligned}
\frac{\partial J}{\partial f(x,y)} &= H^T\Big(H[f] - g\Big)(x,y) \\
&- \lambda \frac{f(x+1,y) - f(x,y)}{\sqrt{\Big(f(x+1,y) - f(x,y)\Big)^2 + \Big(f(x,y+1) - f(x,y)\Big)^2}} \\
&- \lambda \frac{f(x,y+1) - f(x,y)}{\sqrt{\Big(f(x+1,y) - f(x,y)\Big)^2 + \Big(f(x,y+1) - f(x,y)\Big)^2}}
\end{aligned}$$

$$
\begin{aligned}
&- \lambda \frac{f(x-1,y)-f(x,y)}{\sqrt{\Big(f(x,y)-f(x-1,y)\Big)^2+\Big(f(x-1,y+1)-f(x-1,y)\Big)^2}} \\
&- \lambda \frac{f(x,y-1)-f(x,y)}{\sqrt{\Big(f(x+1,y-1)-f(x,y-1)\Big)^2+\Big(f(x,y)-f(x,y-1)\Big)^2}} = 0,
\end{aligned}
$$

or

$$
\begin{aligned}
H^T\Big(H[f]-g\Big)(x,y) &= \lambda\Big\{ \frac{f(x+1,y)-f(x,y)}{\sqrt{\Big(f(x+1,y)-f(x,y)\Big)^2+\Big(f(x,y+1)-f(x,y)\Big)^2}} \\
&- \frac{f(x,y)-f(x-1,y)}{\sqrt{\Big(f(x,y)-f(x-1,y)\Big)^2+\Big(f(x-1,y+1)-f(x-1,y)\Big)^2}}\Big\} \\
&+ \lambda\Big\{ \frac{f(x,y+1)-f(x,y)}{\sqrt{\Big(f(x+1,y)-f(x,y)\Big)^2+\Big(f(x,y+1)-f(x,y)\Big)^2}} \\
&- \frac{f(x,y)-f(x,y-1)}{\sqrt{\Big(f(x+1,y-1)-f(x,y-1)\Big)^2+\Big(f(x,y)-f(x,y-1)\Big)^2}}\Big\},
\end{aligned}
$$

which can be seen as a discrete version of equation (17). But this is still a nonlinear difference equation in f, for which we do not have explicit solutions. We apply an iterative explicit method based on time-dependent gradient descent to solve it.

General gradient descent method In order to solve the above equation for all (x, y), with associated boundary conditions, we apply the gradient descent method, based on the following general idea: assume that we need to solve

$$\min_f J(f),$$

where f could be a number, a vector, or a function (in our case, f is a discrete image function, $(x, y) \mapsto f(x, y)$). We also define a time-dependent equation for $t \geq 0$. Let $f(0) = f_0$, and assume that $f(t)$ solves

$$f'(t) = -J'(f(t)), \quad t > 0.$$

(in our case for image functions f, we would have $(x, y, t) \mapsto f(x, y, t)$, $t \geq 0$, and the differential equation becomes $\frac{\partial f}{\partial t} = -J'(f(\cdot, \cdot, t))$).

We show that $J(f(t))$ is decreasing as t increases. Indeed,

$$\frac{d}{dt}J(f(t)) = J'(f(t))f'(t) = J'(f(t))(-J'(f(t)) = -\Big(J'(f(t))\Big)^2 \leq 0,$$

thus $J(f(t))$ must be deacreasing as t increases. So solving $f'(t) = -J'(f(t))$, $t > 0$ for $f(t)$, decreases the original energy $J(f)$, with $f(t) \to f$, as $t \to \infty$, with f a minimizer of J.

We apply this idea to our main problem, and we discretize the time t by $n\triangle t$, n integer. Let $f^0(x, y)$ be an initial quess, defined for all (x, y) with $1 \leq x \leq M$, $1 \leq y \leq N$ (and then extended

by reflection). We want to have, as above, $\frac{f^{n+1}-f^n}{\triangle t} = -J'(f^n)$, which becomes for pixels (x,y) with $x = 1, ..., M$, $y = 1, ..., N$, for $n \geq 0$,

$$\begin{aligned}
\frac{f^{n+1}(x,y) - f^n(x,y)}{\triangle t} &= -\tilde{h} * \Big(h * f^n - g\Big)(x,y) \\
&+ \lambda \frac{f^n(x+1,y) - f^n(x,y)}{\sqrt{\Big(f^n(x+1,y) - f^n(x,y)\Big)^2 + \Big(f^n(x,y+1) - f^n(x,y)\Big)^2}} \\
&+ \lambda \frac{f^n(x,y+1) - f^n(x,y)}{\sqrt{\Big(f^n(x+1,y) - f^n(x,y)\Big)^2 + \Big(f^n(x,y+1) - f^n(x,y)\Big)^2}} \\
&+ \lambda \frac{f^n(x-1,y) - f^n(x,y)}{\sqrt{\Big(f^n(x,y) - f^n(x-1,y)\Big)^2 + \Big(f^n(x-1,y+1) - f^n(x-1,y)\Big)^2}} \\
&+ \lambda \frac{f^n(x,y-1) - f^n(x,y)}{\sqrt{\Big(f^n(x+1,y-1) - f^n(x,y-1)\Big)^2 + \Big(f^n(x,y) - f^n(x,y-1)\Big)^2}},
\end{aligned}$$

where $\tilde{h}$ is defined by $\tilde{h}(x,y) = h(-x,-y)$.

The boundary conditions are the same as for the linear case (by reflection):
$f^{n+1}(0,y) = f^{n+1}(2,y)$ and $f^{n+1}(M+1,y) = f^{n+1}(M-1,y)$ for all $y = 1, 2, ..., N$.
$f^{n+1}(x,0) = f^{n+1}(x,2)$ and $f^{n+1}(x,N+1) = f^{n+1}(x,N-1)$ for all $x = 1, 2, ..., M$.
$f^{n+1}(0,0) = f^{n+1}(2,2)$, $f^{n+1}(0,N+1) = f^{n+1}(2,N-1)$,
$f^{n+1}(M+1,0) = f^{n+1}(M-1,2)$, $f^{n+1}(M+1,N+1) = f^{n+1}(M-1,N-1)$.

The above relations are repeated for several steps $n \geq 0$, until matrix norm $\|f^{n+1} - f^n\| \leq tolerance$.

In the implementation, we only work with two matrices $f^n = f^{old}$ and $f^{n+1} = f^{new}$, and after each main step n, we reset $f^{old} = f^{new}$.

The above discretization of equation (17) is the slowest one, but it is the simplest one to implement. Faster implementations can be made. The parameter $\triangle t$ must be chosen sufficiently small, so that $E(f^{new}) \leq E(f^{old})$. Also, a small parameter $\epsilon > 0$ can be added inside the square roots, to avoid division by zero when the gradient magnitude is zero (in constant areas of the image). The parameter λ is selected for best restoration results.

5.3.1 Computation of the first order optimality condition in the continuous case

We explain here the computation of the so-called "Euler-Lagrange equation" associated with the minimization of the energies that we have considered above in the continuous case. Assume that $f, g : \Omega \to R$, where Ω is the image domain (a rectangle in the plane) and $\partial\Omega$ denotes its boundary. We will use the notations $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$.

Consider a general energy of the form

$$J(f) = \int_\Omega e(x, y, f, f_x, f_y)dxdy, \tag{19}$$

to be minimized over a subspace $V \subset L^2(\Omega) = \{f : \Omega \to R, \ \int_\Omega f^2(x, y)dxdy < \infty\}$. We do not enter into the details of defining V; V will be a normed vector space over R (subspace of $L^2(\Omega)$, and we will have $f \in V$ iff $E(f) < \infty$. The minimization problem is

$$\min_{f \in V} J(f).$$

We assume that e is differentiable in all variables (or that e can be approximated by a differentiable function). The corresponding first order optimality condition (Euler-Lagrange equation) that must be satisfied by a minimizer f of J, can be expressed as

$$\frac{\partial e}{\partial f} = \frac{\partial}{\partial x}\Big(\frac{\partial e}{\partial (f_x)}\Big) + \frac{\partial}{\partial y}\Big(\frac{\partial e}{\partial (f_y)}\Big) \Leftrightarrow \partial J(f) = 0$$

inside Ω, where $\partial J(f) = \frac{\partial e}{\partial f} - \frac{\partial}{\partial x}\Big(\frac{\partial e}{\partial (f_x)}\Big) - \frac{\partial}{\partial y}\Big(\frac{\partial e}{\partial (f_y)}\Big)$, together with boundary conditions.

We show next the derivation of the Euler-Lagrange equation in two cases.

• **Quadratic Regularization (denoising only)**

Let

$$E(f) = \int_\Omega (f(x, y) - g(x, y))^2 dxdy + \int_\Omega |\nabla f|^2 dxdy = \int_\Omega (f - g)^2 dxdy + \int_\Omega \Big[(f_x)^2 + (f_y)^2\Big] dxdy.$$

Define

$$V = \{v \in L^2(\Omega) : \ E(v) < \infty\}.$$

If f is a minimizer of $E(f)$, then we must have $E(f) \leq E(f + \epsilon v)$, for all real numbers ϵ and all other functions $v \in V$. Denote by $s(\epsilon) = E(f + \epsilon v)$; therefore we must have $s(0) \leq s(\epsilon)$ for all ϵ, thus we must have $s'(0) = 0$ for all other test functions $v \in V$.

We first compute

$$s(\epsilon) = E(f + \epsilon v) = \int_\Omega (f(x, y) + \epsilon v(x, y) - g(x, y))^2 dxdy + \int_\Omega \Big[((f + \epsilon v)_x)^2 + ((f + \epsilon v)_y)^2\Big] dxdy,$$

$$s(\epsilon) = E(f + \epsilon v) = \int_\Omega (f(x, y) + \epsilon v(x, y) - g(x, y))^2 dxdy + \int_\Omega \Big[(f_x + \epsilon v_x)^2 + (f_y + \epsilon v_y)^2\Big] dxdy.$$

We compute $s'(\epsilon)$:

$$s'(\epsilon) = 2\int_\Omega (f(x, y) + \epsilon v(x, y) - g(x, y))v(x, y)dxdy + \int_\Omega \Big[2(f_x + \epsilon v_x)v_x + 2(f_y + \epsilon v_y)v_y\Big] dxdy.$$

We take $\epsilon = 0$:

$$s'(0) = 2\int_\Omega (f(x,y) - g(x,y))v(x,y)dxdy + \int_\Omega \Big[2f_x v_x + 2f_y v_y\Big]dxdy = 0 \text{ for all } v \in V.$$

We apply integration by parts in the second term,

$$2\int_\Omega (f(x,y)-g(x,y))v(x,y)dxdy - \int_\Omega \Big[2(f_x)_x v + 2(f_y)_y v\Big]dxdy + \int_{\partial\Omega} 2(f_x v n_1 + f_y v n_2)dS = 0 \text{ for all } v,$$

where $\vec{n} = (n_1, n_2)$ is the exterior unit normal to the boundary $\partial\Omega$.

We can rewrite the last relation as

$$\int_\Omega \Big[f - g - \triangle f\Big]v(x,y)dxdy + \int_{\partial\Omega} (\nabla f \cdot \vec{n})v dS = 0 \text{ for all } v \in V,$$

from where we obtain (by a theorem)

$$\begin{cases} f - g - \triangle f = 0 \text{ in } \Omega \\ \nabla f \cdot \vec{n} = 0 \text{ on } \partial\Omega \end{cases}$$

which is equation (11) with associated (free) boundary conditions.

• **Total Variation Regularization (denoising-deblurring)**

We denote by the linear operator H the mapping $f \mapsto h * f = Hf$ (where h is the blurring convolution kernel). Note that here, Hf is not a product.

We want to show in continuous variables that, if f is a minimizer of (16) recalled here,

$$J(f) = \frac{1}{2}\int\int (h * f(x,y) - g(x,y))^2 dxdy + \lambda \int\int |\nabla f| dxdy, \tag{20}$$

rewritten as

$$J(f) = \frac{1}{2}\int\int (Hf - g)^2 dxdy + \lambda \int\int |\nabla f| dxdy, \tag{21}$$

then f formally satisfies the anisotropic diffusion PDE (17) recalled here:

$$\tilde{h} * h * f = \tilde{h} * g + \lambda \frac{\partial}{\partial x}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial x}\Big) + \lambda \frac{\partial}{\partial y}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial y}\Big), \tag{22}$$

or

$$H^T[Hf] = H^T g + \lambda \frac{\partial}{\partial x}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial x}\Big) + \lambda \frac{\partial}{\partial y}\Big(\frac{1}{|\nabla f|(x,y)}\frac{\partial f}{\partial y}\Big) \tag{23}$$

(where H^T is the adjoint or transpose operator of H).

To obtain equation (23), we apply the same steps as for E: we define $s(\epsilon) = J(f + \epsilon v)$, where ϵ is real and v is another function like f. Then we compute $s'(\epsilon)$, we take $\epsilon = 0$ and we impose $s'(0) = 0$ for all functions v. Similarly, we apply integration by parts and we arrive to (23) in Ω, together with the boundary conditions $\frac{\nabla f}{|\nabla f|} \cdot \vec{n} = 0$ on the boundary $\partial\Omega$. The remaining details are left as an exercise. It remains to find H^T, which is presented next.

Computation of the adjoint H^T of H

For the convolution term, we assume that all image functions f, g, h are defined on the entire plane (by extension) and belong to L^2. The space L^2 is an inner product vector space associated with the product $\langle f, g\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)g(x,y)dxdy$, for any functions f and g in L^2. Also, $H : L^2 \to L^2$ is the linear convolution operator defined by

$$Hf(x,y) = h * f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\alpha,\beta)f(x-\alpha, y-\beta)d\alpha d\beta.$$

By the definition of the adjoint, we must find another linear operator H^T such that

$$\langle Hf, g\rangle = \langle f, H^T g\rangle$$

for all functions $f, g \in L^2$. We have

$$\langle Hf, g\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(h * f)(x,y)g(x,y)dxdy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\alpha,\beta)f(x-\alpha,y-\beta)d\alpha d\beta\Big]g(x,y)dxdy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\alpha,y-\beta)g(x,y)dxdy\Big]h(\alpha,\beta)d\alpha d\beta.$$

We make the change of variables in x, y: $x - \alpha = X$, $y - \beta = Y$ and we obtain

$$\langle Hf, g\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(X,Y)g(\alpha+X,\beta+Y)dXdY\Big]h(\alpha,\beta)d\alpha d\beta$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(X,Y)\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\alpha,\beta)g(\alpha+X,\beta+Y)d\alpha d\beta\Big]dXdY.$$

Now making the change of variables $\alpha = -a$, $\beta = -b$, we obtain

$$\langle Hf, g\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(X,Y)\Big[\int_{+\infty}^{-\infty}\int_{+\infty}^{-\infty} h(-a,-b)g(X-a,Y-b)(-1)(-1)dadb\Big]dXdY.$$

Since $\int_{+\infty}^{-\infty} = -\int_{-\infty}^{+\infty}$, we finally obtain

$$\langle Hf, g\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(X,Y)\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(-a,-b)g(X-a,Y-b)dadb\Big]dXdY$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(X,Y)\Big[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{h}(a,b)g(X-a,Y-b)dadb\Big]dXdY$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(X,Y)\tilde{h} * g(X,Y)dXdY = \langle f, H^T g\rangle,$$

where we introduced $\tilde{h}(a,b) = h(-a,-b)$. Therefore: $H^T g(x,y) = \tilde{h} * g(x,y)$.

6 Image Segmentation

Segmentation is a midd-level vision task, where the input is a pre-processed image f (sharp, without noise). The output could be an image g, or could no longer be an image, but image attributes: set of points representing the edges of f, boundaries of objects defined by implicit curves or by sets of points, etc. Segmentation is the essential step before image analysis and recognition. Segmentation can also be defined as the process of partitioning the input image f into its constituent objects or regions.

The segmentation can be based on several different criteria:

- It can be based on *discontinuity*: partition an image based on abrupt changes in intensity (edges)

- It can be based on *similarity*: partition an image into regions that are similar according to a set of predefined rules (based on color, texture, etc).

6.1 The gradient edge detector

Consider a differentiable function $(x, y) \mapsto f(x, y)$ in two dimensions. We define its gradient operator as being the vector of first-order partial derivatives,

$$\nabla f(x,y) = \Big[\frac{\partial f}{\partial x}(x,y) \;\; \frac{\partial f}{\partial y}(x,y)\Big],$$

and its gradient magnitude as the Euclidean norm of the vector ∇f,

$$|\nabla f|(x,y) = \sqrt{\Big(\frac{\partial f}{\partial x}\Big)^2 + \Big(\frac{\partial f}{\partial y}\Big)^2}.$$

The usual central finite differences approximations of the gradient are (assuming $\triangle x = \triangle y = 1$)

$$\frac{\partial f}{\partial x}(x,y) \approx \frac{f(x+1,y) - f(x-1,y)}{2}, \quad \frac{\partial f}{\partial y}(x,y) \approx \frac{f(x,y+1) - f(x,y-1)}{2}.$$

Neglecting the constant $\frac{1}{2}$, these approximations can be implemented using the following masks w centered at (x, y) and spatial linear filtering,

$$\begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \text{ in } x, \quad \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline -1 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \text{ in } y.$$

In image processing, since the input image f may still be noisy, a smoothed version of these approximations is used, called the Sobel gradient operators, computed by the following two masks w in x and y respectively:

$$\begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \text{ in } x, \quad \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \text{ in } y.$$

Note that all these four masks are sharpening masks, since the sum of their coefficients is zero.

The gradient magnitude can be used as an edge detector. Indeed, in the areas where the image f does not vary too much, the gradient magnitude $|\nabla f|$ is small (close to zero); on the contrary, in the areas where there are strong variations (at edges), the gradient magnitude $|\nabla f|$ is large. We define the output image $g(x,y) = |\nabla f|(x,y)$ (it will show white edges over black background), or a thresholded version of $|\nabla f|$.

Assuming that $|\nabla f|$ is approximated by some of the above discrete operators, we define

$$g(x,y) = |\nabla f|(x,y) \text{ (discrete version)}$$

(followed by rescaling), or a thresholded gradient map:

$$g(x,y) = \begin{cases} 255 & \text{if} \quad |\nabla f|(x,y) \geq \text{ tolerance } T, \\ 0 & \text{if} \quad |\nabla f|(x,y) < \text{ tolerance } T. \end{cases}$$

Note that the operation $f \mapsto g = |\nabla f|$ is nonlinear.

6.2 Edge detection by zero-crossings of the Laplacian (the Marr-Hildtreth edge detector)

A more advanced technique for edge detection uses the zero-crossings of the Laplacian. The pixels where the Laplacian $\triangle f$ changes sign, can be seen as centers of thick edges. The main steps for this method are as follows (these steps can be done in the spatial domain or in the frequency domain):

(1) Filter the input image f by a Gaussian low-pass filter (equivalent with $G_\sigma * f$, where $G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$)

(2) Compute the Laplacian $\triangle(G_\sigma * f)$ of step (2) by using for example the 3×3 Laplacian masks (5-point Laplacian or 9-point Laplacian)

(3) Find the zero-crossings of the result from step (2).

Remarks

(i) Since we have $\triangle(G_\sigma * f) = (\triangle G_\sigma) * f$, steps (i)-(ii) can be combined into a single step as follows: define the Laplacian of a Gaussian $LoG(x,y) = \triangle G_\sigma = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$ (" inverted mexican hat") and its discrete version. Then compute $LoG * f$; this can be done using the mask

0	0	1	0	0
0	1	2	1	0
1	2	−16	2	1
0	1	2	1	0
0	0	1	0	0

.

(ii) In order to find the zero-crossings in step (3) above, consider a pixel (x,y) and its 3×3 neighborhood centered at (x,y). We have a zero-crossings at (x,y) if the signs of at least two of its opposing neighboring pixels are different (we have to check four cases: left/right, up/down, and the two diagonals). Another way is first by partitioning the result of (2) into: assign white=255 to positive pixels and black=0 to negative pixels; then compute the edge map of the black-and-white image.

(iii) An improved version of this detector is the Canny edge detector, where the Laplacian is substituted by the second order derivative in the gradient direction: $f_{xx}(f_x)^2 + 2f_x f_y f_{xy} + (f_y)^2 f_{yy}$.

6.3 Boundary detection by curve evolution and active contours

We assume that we have given an image $f : \Omega \mapsto R$ that contains several objects forming the foreground, and a background. Curve evolution can be used to detect boundaries of objects in images, as follows:

(i) Start the process with an initial curve;

(ii) Move or deform the curve according to some criterion;

(iii) The curve has to stop on boundaries of objects.

6.3.1 Curve Representation

Assume that we have a closed curve C in the plane, with parameterization $s \in [0,1] \mapsto C(s) = (x(s), y(s))$, with $C(0) = C(1)$.

Explicit Representation Numerically, we can represent a moving curve C by a sequence of points in the plane (discrete points belonging to the curve). However, by this approach, it is hard to handle change of topology (merging or breaking); also, re-parameterization and re-discretization are often necessary. On the other hand, such representation is not computationally expensive, and it easily allows representation of open curves.

Implicit Representation Another way to represent a curve C, which is the boundary of an open domain, is as the isoline of a Lipschitz continuous function ϕ. Such representation, called implicit representation or level set representation [10], allows for automatic change of topology, and discretization on a fixed regular (rectangular) grid. On the other hand, the method is computationally more expensive and it does not directly allow the representation of open curves. This is the approach that we will consider.

Let $\phi : \Omega \to$ be a Lipschitz continuous function, and define $C = \{(x,y) \in \Omega : \ \phi(x,y) = 0\}$. In other words, C is the zero isoline of the function ϕ. Note that here C can be made of several connected components.

As an example, consider a circle C centered at (x_0, y_0) and of radius r; we can define

$$\phi(x,y) = r - \sqrt{(x-x_0)^2 + (y-y_0)^2}.$$

Then

$\phi(x,y) = 0$ if $(x,y) \in$ circle C;

$\phi(x,y) < 0$ if $(x,y) \in$ outside of circle C;

$\phi(x,y) > 0$ if $(x,y) \in$ inside circle C.

We notice that for a general curve C which is the boundary of an open domain, we can define ϕ as the *signed distance function* to C:

$$\phi(x,y) = \begin{cases} dist\Big((x,y), C\Big) & \text{if} \quad (x,y) \in \text{ inside } C \\ -dist\Big((x,y), C\Big) & \text{if} \quad (x,y) \in \text{ outside } C \\ 0 & \text{if} \quad (x,y) \in C \end{cases} .$$

Note that, under some assumptions, the (signed) distance function to C satisfies the eikonal equation: $|\nabla \phi| = 1$ over Ω in the sense of viscosity solutions.

Using the differentiable level set function ϕ, we can define several intrinsic geometric quantities of the curve C:

• Unit normal to the curve $\vec{N} = (N_1, N_2) = -\frac{\nabla\phi}{|\nabla\phi|}$

• Curvature at $(x, y) \in C$: $K(x, y) = \frac{\partial}{\partial x}\left(\frac{\phi_x}{|\nabla\phi|}\right) + \frac{\partial}{\partial y}\left(\frac{\phi_x}{|\nabla\phi|}\right) = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2+\phi_y^2)^{3/2}}$.

• Heaviside function $H(\phi) = \begin{cases} 1 \text{ if } \phi \geq 0 \\ 0 \text{ if } \phi < 0 \end{cases}$, and the delta function $\delta(\phi) = H'(\phi)$ in the weak sense

• Area enclosed by the curve: $A\{\phi \geq 0\} = \int_\Omega H(\phi(x, y))dxdy$

• Length of the curve: $L\{\phi(x, y) = 0\} = \int_\Omega |\nabla H(\phi(x, y)|$

• If $f : \Omega \to R$ is the given image over Ω, then we can define the mean of f over the regions $\{\phi \geq 0\}$ and $\{\phi \leq 0\}$:

$$mean(f)_{\phi\geq 0} = \frac{\int_\Omega f(x, y)H(\phi)dxdy}{\int_\Omega H(\phi)dxdy}, \quad mean(f)_{\phi\leq 0} = \frac{\int_\Omega f(x, y)(1 - H(\phi))dxdy}{\int_\Omega(1 - H(\phi))dxdy}.$$

For curve evolution, we need to work with a family of curves $C(t)$, over time $t \geq 0$. This family of curves is represented implicitly, as isolines of the function $\phi(x, y, t)$, $(x, y) \in \Omega$, $t \geq 0$, such that

$$C(t) = \{(x, y) \in \Omega : \ \phi(x, y, t) = 0\}.$$

Assume that $C(t) = (x(t, s), y(t, s))$ moves in the normal direction, according to the equation

$$C'(t) = F\vec{N},$$

where F is a scalar speed function. Assume also that $C(t)$ is implicitly represented by ϕ: thus, $\phi(x(t, s), y(t, s), t) = 0$. Differentiating with respect to t, we obtain

$$\frac{d}{dt}\phi(x(t, s), y(t, s), t) = 0,$$

or

$$\frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial t} = 0,$$

but since $C'(t) = (\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}) = F\vec{N}$, we obtain

$$\frac{\partial\phi}{\partial t} + \frac{\partial\phi}{\partial x}N_1F + \frac{\partial\phi}{\partial y}N_2F = 0,$$

or

$$\frac{\partial\phi}{\partial t} + F\nabla\phi \cdot \vec{N} = 0.$$

Substituting $\vec{N}$ by $-\frac{\nabla\phi}{|\nabla\phi|}$, we finally obtain

$$\frac{\partial\phi}{\partial t} = F|\nabla\phi|,$$

which is the main level set equation. The speed F can depend on (x, y), on the curvature $K(x, y)$, could be a constant, or could depend on the image f to be segmented.

Active contours for boundary detection Usually models are based on two concepts: (i) regularization of the curve (such as motion by curvature) and (ii) stopping criterion.

Edge-based models Let $g(x,y) = \frac{1}{1+|\nabla(G_\sigma * f(x,y))|^2}$. This is an "edge-function": it is close to zero (small) near edges (where the gradient magnitude is high) and large away from edges. Thus such models segment objects from the background based on discontinuities in the image f.

• Model proposed by Caselles et al. in [3]

$$\phi(x,y,0) = \phi_0(x,y), \quad \frac{\partial \phi}{\partial t} = g(x,y)K(x,y)|\nabla \phi|. \tag{24}$$

Here the speed function is $F = g(x,y)K(x,y)$. If $g(x,y) > 0$ (away from edges), the curve moves by motion with a speed proportional to the mean curvature. If $g(x,y) \approx 0$ (on an edge), $F \approx$ thus the curve stops on boundaries of objects (edges). This nonlinear partial differential equation does not come from an energy minimization.

• Geodesic active contours Caselles et al [4] , Kichenassamy et al [9]: find a curve C that minimizes the weighted length term

$$\min_C \int_C g(C) dS.$$

By this method, a minimizer C should overlap with edges in the image f, since the energy is minimal when $g(C) \approx 0$. In the level set implicit representation, we define $C(t) = \{(x,y) : \phi(x,y,t) = 0\}$, thus the model becomes

$$\min_\phi \Big\{ J(\phi) = \int_\Omega g(x,y)|\nabla H(\phi(x,y)|dxdy.$$

The first order extremality condition associated with the minimization, combined with gradient descent, $\frac{\partial \phi}{\partial t} = -\partial J(\phi)$, gives

$$\phi(x,y,0) = \phi_0(x,y), \quad \frac{\partial \phi}{\partial t} = \delta(\phi)\Big[\frac{\partial}{\partial x}\Big(\frac{g(x,y)}{|\nabla \phi|}\frac{\partial \phi}{\partial x}\Big) + \Big(\frac{g(x,y)}{|\nabla \phi|}\frac{\partial \phi}{\partial y}\Big)\Big].$$

It is possible to extend the motion to all level lines of ϕ (even if we are interested only in the zero-level line), substituting $\delta(\phi)$ by $|\nabla \phi|$, to obtain a more geometric motion

$$\phi(x,y,0) = \phi_0(x,y), \quad \frac{\partial \phi}{\partial t} = |\nabla \phi|\Big[\frac{\partial}{\partial x}\Big(\frac{g(x,y)}{|\nabla \phi|}\frac{\partial \phi}{\partial x}\Big) + \Big(\frac{g(x,y)}{|\nabla \phi|}\frac{\partial \phi}{\partial y}\Big)\Big]. \tag{25}$$

We notice that the difference between models (24) and (25) is that the stopping edge-function g is outside the curvature term in (24), while it is inside the curvature term in (25). Model (25) has a better attraction of the curve C towards edges.

For both of these models, the initial curve defined by $\phi_0(x,y) = 0$ has to enclose the objects to be detected. Also, these models have more difficulty in segmenting strongly concave shapes.

Region-based models This model [5, 6] is based on the partitioning of the input image f into "objects" and "background" based on pixels intensity similarity, and no longer on discontinuities. The following energy is minimized

$$\min_{C,c_1,c_2} E(C,c_1,c_2) = \int_{\text{inside}(C)} (f-c_1)^2 dxdy + \int_{\text{outside}(C)} (f-c_2)^2 dxdy + \lambda Length(C). \tag{26}$$

We assume that the foreground can be described by the region where $f(x, y) \approx$ constant c_1, and the background can be described by the region where $f(x, y) \approx$ constant c_2, with $c_1 \neq c_2$.

Thus the first two terms form the stopping (segmentation) criterion. The last term is a regularization (penalty on the length of the curve).

In the implicit representation, model (26) can be expressed as

$$\min_{\phi,c_1,c_2} E(\phi, c_1, c_2) = \int_\Omega (f - c_1)^2 H(\phi) dxdy + \int_\Omega (f - c_2)^2 (1 - H(\phi)) dxdy + \lambda \int_\Omega |\nabla H(\phi)|. \quad (27)$$

Minimizing the energy $E(\phi, c_1, c_2)$ with respect to its variables, we obtain the associated extremality conditions

$$c_1 = \frac{\int_\Omega f(x, y) H(\phi) dxdy}{\int_\Omega H(\phi) dxdy},$$

$$c_2 = \frac{\int_\Omega f(x, y)(1 - H(\phi)) dxdy}{\int_\Omega (1 - H(\phi)) dxdy},$$

and for the minimization in ϕ we use gradient descent, $\frac{\partial \phi}{\partial t} = -\frac{\partial E}{\partial \phi}$, or after simplifications,

$$\phi(x, y, 0) = \phi_0(x, y), \quad \frac{\partial \phi}{\partial t} = \delta(\phi)\Big[- (f - c_1)^2 + (f - c_2)^2 + \lambda K(x, y)\Big].$$

This model is less dependent on the choice of the initial curve (this does not have to enclose the objects). Also, interior contours and strongly concave shapes are automatically detected.

For more details we refer to [5, 6].

References

[1] A. Buades, B. Coll, J.M. Morel, *A non local algorithm for image denoising*, IEEE Computer Vision and Pattern Recognition 2005, Vol 2, pp: 60-65, 2005.

[2] A. Buades, B. Coll, J.M. Morel, *A review of image denoising methods, with a new one*, Multiscale Modeling and Simulation, Vol4 (2), pp: 490-530, 2006.

[3] V. Caselles, F. Catte, T. Coll, F. Dibos, *A geometric model for active contours*, Numerische. Mathematik 66, pp. 1-31, 1993.

[4] V. Caselles, R. Kimmel, G. Sapiro, *Geodesic Active Contours*, IJCV 1997.

[5] T. Chan and L. Vese, *An active contour model without edges*, Scale-Space Theories in Computer Vision, Lecture Notes in Computer Science, 1682:141-151, 1999.

[6] T.F. Chan and L.A. Vese, *Active contours without edges*, IEEE Transactions on Image Processing, 10 (2), Feb. 2001, pp. 266 -277.

[7] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 3rd edition, Prentice-Hall, 2008.

[8] R.C. Gonzalez, R.E. Woods and S.L. Eddins, *Digital Image Processing Using Matlab*, 2nd edition, Prentice-Hall, 2010.

[9] S. Kichenssamy, Kumar, P. Olver, A. Tannenbaum, A. Yezzi, *Conformal curvature flows: From phase transitions to active vision*, Archive Rat.Mech. Anal. 134 (3): 275-301 1996.

[10] Osher, S.; Sethian, J. A. (1988), *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys. 79: 1249.

[11] L. Rudin, S. Osher, E. Fatemi, *Nonlinear Total Variation Based Noise Removal Algorithms*, Physica D: Nonlinear Phenomena, Vol. 60, Issues 1-4, 1992.

[12] L. Rudin and S. Osher, *Total Variation Based Image Restoration with Free Local Constraints*, ICIP (1) 1994, pp. 31-35.