Math 155: Mathematical Imaging

Instructor: Fuqun Han (+Yunuo Chen) - Winter 2025

Textbook: R. Gonzalez & R. Woods - Digital Image Processing

Topics: Image representation, intensity transformations & histogram processing, spatial & frequency-domain filtering, image restoration, color image processing

Table of Contents

Image Sampling & Quantization (Lec. 1)

Image Acquisition (§1.1)

At any point (x, y), can represent the intensity/gray level at (x, y) by a function (x, y) [satisfying $0 < f(x, y) < \infty$].

↑ (*) for a grayscale image

In particular, can write $f(x, y)$ as:

$$f(x, y) = i(x, y) \cdot r(x, y) \qquad 0 < i < \infty,\ 0 < r < 1$$

where: $i(x, y)$ represents illumination [by external light sources], and $r(x, y)$ represents reflectance [from material/object properties].

Image Representation (§1.2)

After obtaining an image, we can find L_{min} & L_{max} s.t. $0 < L_{min} \leq f(x, y) \leq L_{max} < \infty$.

Then, we can pick some $L > 0$ and rescale $[L_{min}, L_{max}] \to [0, L-1]$ to obtain a [fixed] gray/intensity scale.

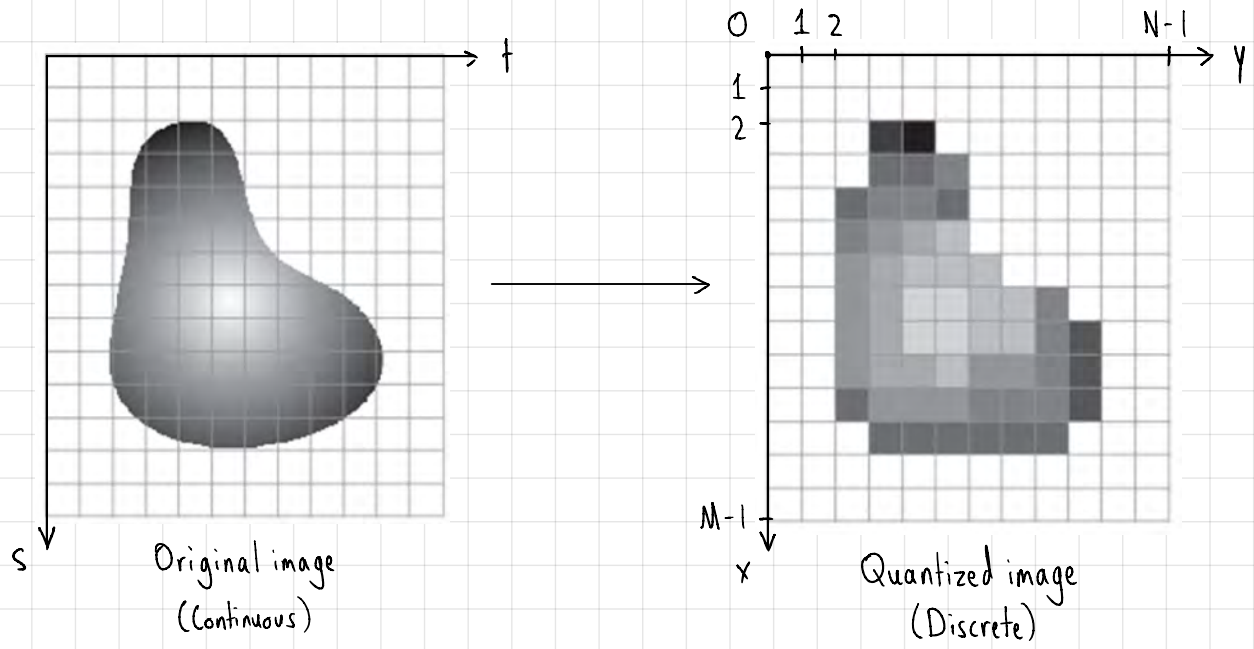
- Commonly: 0 = black, $L-1$ = white
- In computers, typically choose $L - 1 = 2^k - 1$ for some $k \in \mathbb{N}$ [call this a k-bit image / say the image has 2^k gray levels]
  - Ex: $k = 8 \longrightarrow L - 1 = 255$, gray scale is $[0, 255]$; 256 gray levels

Image Representation (cont.)

(assumed)

The original image is given by a continuous function $f(s,t) \geq 0$, representing intensity at each point (s,t).

A digital image is a 2D discrete function $f(x,y)$, where $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$.



Equivalently: can consider a digital image as a matrix:

$$\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Matrix elements also called:

- Picture/image elements
- Pixels

Intensity Transformations (Lec. 2-7)

Basics of Intensity Transformations (§2.1)

For an input [digital] image $f(x,y)$, consider a mapping operator T with output image g:

$$g(x,y) = T[f(x,y)]$$

"neighborhood" of (x_0, y_0)

Simple case: $g(x_0, y_0)$ only depends on $f(x_0, y_0)$ [i.e. does not depend on surrounding pixels]

+ $g(x_0, y_0)$ is location-invariant [→ notation: can represent $g(x,y)$ by a function $T(r)$]

  - $f(x_0, y_0) = f(x_1, y_1) \Rightarrow g(x_0, y_0) = g(x_1, y_1)$

  - function of intensity

Basic Intensity Transformations

1. Image Negatives

$$s = (L-1) - r$$

- Flips bright & dark regions of the image

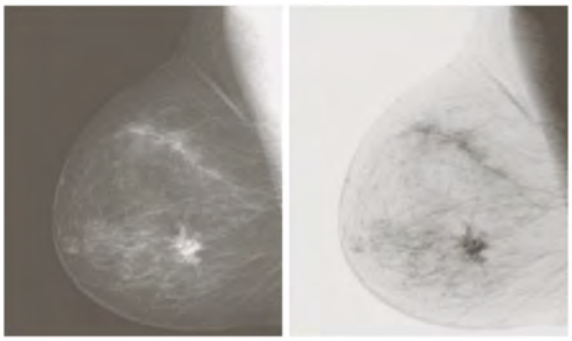


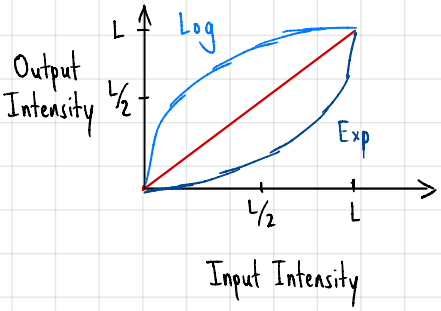
Figure 2: Negative Transform

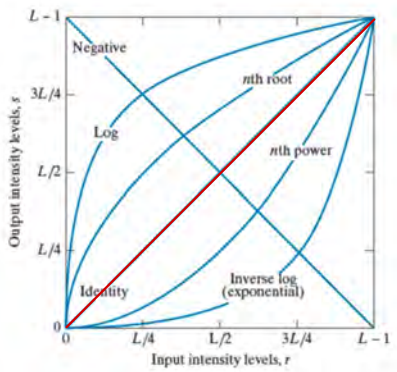
2. Log Transformation

$$s = c \cdot \log(1+r)$$

[for some constant c]

- Compresses high-level values; "expands" range of dark pixels





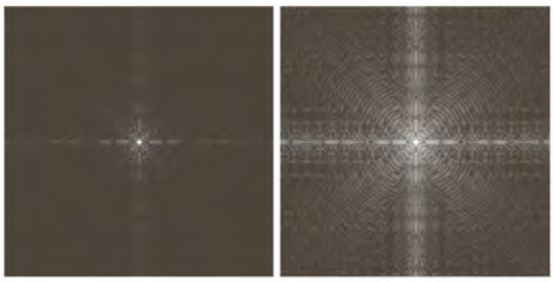


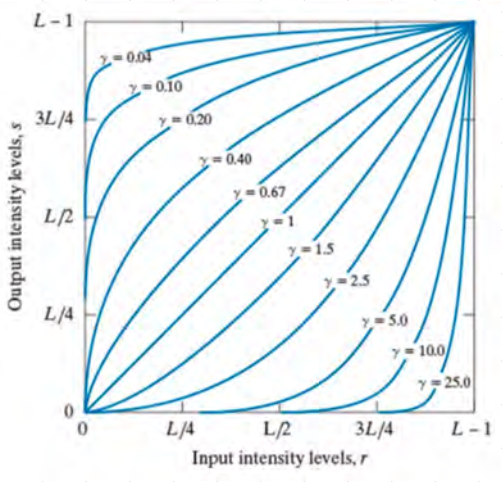
Figure 3: Log Transform

Basic Intensity Transformations (cont.)

3. Power Law (Gamma) Transformation

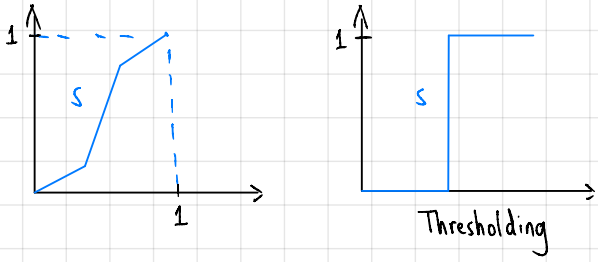
$$s = c \cdot r^{\gamma} \qquad [c, \gamma > 0]$$

- $\gamma \in (0, 1) \longrightarrow$ expands dark pixels
- $\gamma > 1 \longrightarrow$ compresses dark pixels



4. Piecewise Linear Transformation Functions

- Can use to compress nearly-black & nearly-white pixels, e.g. (contrast stretching)
- Limiting case: thresholding



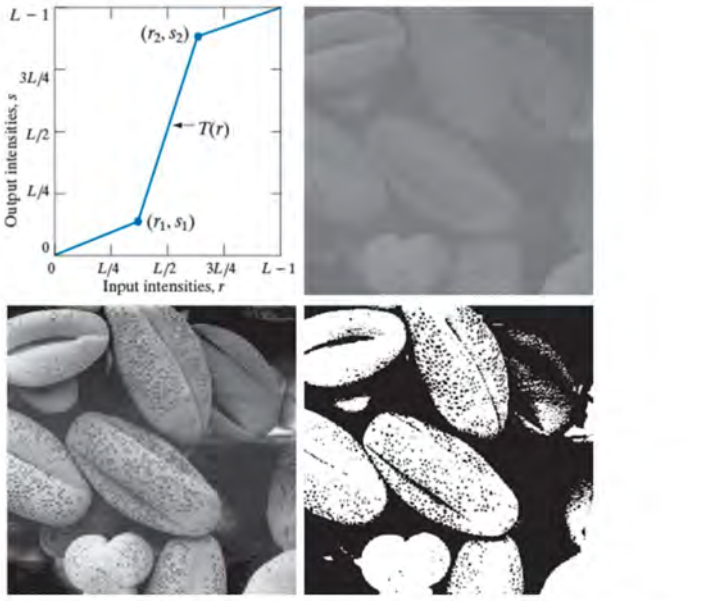


Figure 6: Piece-wise linear transformation function, original image of pollen, contrast stretching, thresholding.

Histogram Processing (§2.2)

Def: Histogram

Given a digital image with intensity levels in $[0, L-1]$, its histogram is the function:

$$h(r_k) = n_k$$

where:
- r_k is the k^{th} intensity value ($r_k = 0, 1, \ldots, L-1$), and
- n_k is the number of pixels with intensity value r_k

(*) Ex:

$$f = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 2 & 3 \\ 1 & 0 & 1 \end{bmatrix} \longrightarrow h(x) = \begin{cases} 2 & x=0 \\ 5 & x=1 \\ 1 & x=2 \\ 1 & x=3 \end{cases} \longrightarrow p(x) = \begin{cases} 2/9 & x=0 \\ 5/9 & x=1 \\ 1/9 & x=2 \\ 1/9 & x=3 \end{cases}$$

$(L-1=3)$

The normalized histogram is given by $p(r_k) = \frac{n_k}{n}$, (Also denoted $p_r(r)$)

where n is the total number of pixels in the image ($n = M \cdot N$)

- $p(r_k)$ represents the probability of occurrence of the intensity value r_k: $\mathbb{P}(f = r_k)$

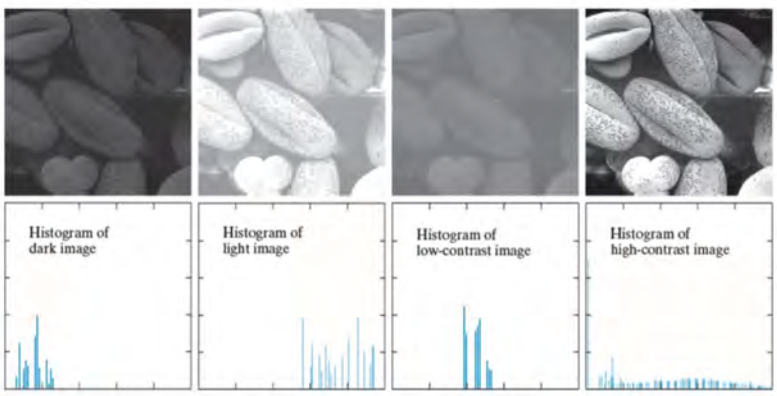


Figure 7: Histogram of different images, the horizontal axis is r_k and vertical axis is $p(r_k)$.

Notice: The histogram gives info regarding "lightness" & contrast of an image
- Lighter <=> more right-weighted
- Higher contrast <=> more uniform (Less concentrated)

Histogram Equalization

Motivation: In imaging, we often want our images to have higher contrast

- Higher contrast → more details visible (usually)

Recall: Previously, saw that high contrast image <=> more uniform histogram

→ **Now**: Want a transformation $T(r)$ $[g(x,y) = T(f(x,y))]$ that produces a more uniform histogram

$$r = f(x,y) \xrightarrow{T(r)} s = g(x,y)$$

Regarding a general transformation T, assume T satisfies the following:

(i) $T(r)$ is monotonically increasing $[r_1 \geq r_0 \Rightarrow T(r_1) \geq T(r_0)]$

(ii) $0 \leq T(r) \leq L-1$

In our case: we want to guarantee that T is invertible (T^{-1} exists) for histogram matching (later)

→ assume T is strictly monotonically increasing $[r_1 > r_0 \Rightarrow T(r_1) > T(r_0)]$

Histogram Equalization Transformation Functions

Continuous case: For an image r with continuous corresponding p_r, the transformation function is:

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

(*) Probability Terminology

- $p_r(r)$ and $p_r(s)$ are the probability density functions (PDFs) of the original & transformed intensities, resp.
- $\int_0^r p_r(w)\,dw$ is the cumulative distribution function (CDF) of r (sometimes denoted $F_r(x)$)

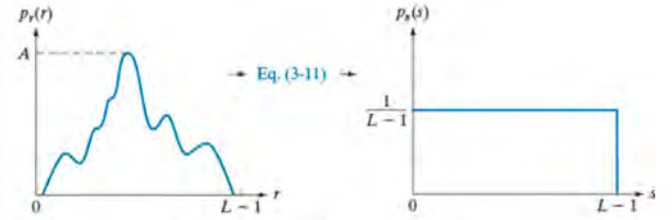


Figure 8: Result from histogram equalization

Histogram Equalization Transformation Functions (cont.)

Continuous Case (cont.)

Lemma: If $r = f(x,y)$ and $s = T(r) = (L-1)\int_0^r p_r(w)\,dw$ for $s = g(x,y)$, then the PDF of the output image $p_s(s)$ is uniform.

(*) Proof: Histogram Equalization Transformation (Continuous)

From probability theory: if $T(r)$ is continuous & differentiable, then:

$$|p_r(r)\,dr| = |p_s(s)\,ds| \implies p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

We can use this with our equation for $s = T(r)$ to verify that $p_s(s)$ is uniform:

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = \frac{2r}{(L-1)^2 \cdot \frac{1}{|ds/dr|}} = \frac{1}{L-1} \quad \checkmark$$

Discrete Case

For a real discrete image, the histogram equalization transformation function is given by:

$$s_k = T(r_k) = (L-1)\cdot\sum_{j=0}^{k} p(r_j)$$

↳ Replaces integration (continuous case) with a discrete sum

(*) Proof: Via same method as in the continuous case

Histogram Matching

Task: Given an input image with associated $p_r(r)$ and a target $p_z(z)$, find a transformation function $T(r)$ such that the output image has associated histogram/PDF $p_z(z)$.

(Discrete) (Cts.)

Process:

(1) Apply histogram equalization to r to get $s = T(r) \sim \text{Uniform}(0, L-1)$

(2) Apply histogram equalization to z to get $s' = G(z) \sim \text{Uniform}(0, L-1)$

(3) $s = T(r)$, $s' = G(z) \sim \text{Uniform}(0, L-1) \longrightarrow$ $z = G^{-1}(T(r))$

↳ Recall: G strictly monotonically increasing (assumed) ⇓ G invertible

(*) Discrete case: Round z to the nearest integer

Local Histogram Processing

So far: all previously mentioned histogram transformations have been global (i.e. done over the entire image).

→ Alternative: Can perform local histogram transformations within small neighborhoods to better capture/enhance very small details in an image.

Figure 10: Original image (Left); Global Histogram equalization (middle); and local Histogram equalization on a 3×3 neighborhood (Right).

Histogram Statistics (§2.3)

Def: Histogram Statistics

Given a histogram r, can compute the following statistics:

1. Mean of r:

$$m = \sum_{i=0}^{L-1} r_i p(r_i) = \frac{1}{M \cdot N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

2. n^{th} moment of r about its mean:

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$$

Note: (i) $\mu_0(r) = 1$

(ii) $\mu_1(r) = \sum_{i=0}^{L-1} (r_i - m) p(r_i) = \sum_{i=0}^{L-1} r_i p(r_i) - m \sum_{i=0}^{L-1} p(r_i) = 0$

(iii)
$$\mu_2(r) = \sigma^2(r) = \left(\frac{1}{M \cdot N}\right)^2 \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x,y) - m]^2$$

← called the sample variance (measures average contrast)

Local Histogram Statistics

Similar to histogram processing, can work with local histogram statistics within smaller neighborhoods to enhance smaller details.

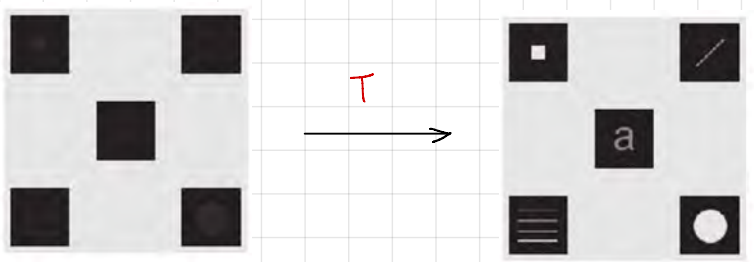
(*) Ex: Let S_{xy} a neighborhood of (x,y), ps_{xy} the normalized histogram for S_{xy} → can find the local mean and variance:

$$m_{S_{xy}} = \frac{1}{M_{S_{xy}} N_{S_{xy}}} \sum_{(x,y) \in S_{xy}} f(x,y) \quad ; \quad \sigma^2_{S_{xy}} = \frac{1}{M_{S_{xy}} N_{S_{xy}}} \sum_{(x,y) \in S_{xy}} [f(x,y) - m_{S_{xy}}]^2$$

→ Can use to enhance an image locally [via transformation]: let m_G, σ^2_G be the global mean & variance, and define:

$$g(x,y) = \begin{cases} C f(x,y) & \text{if } k_0 m_G \le m_{S_{xy}} \le k_1 m_G \text{ and } k_2 \sigma_G \le \sigma_{S_{xy}} \le k_3 \sigma_G \\ f(x,y) & \text{otherwise} \end{cases}$$

← $k_0 = 0$, $k_1 = 1/4$, $k_2 = 0$, $k_3 = 1$, $C = 22.8$



Spatial Filtering (§2.4)

2 main types of image processing:

1. Spatial domain:

   (a) Intensity Transformation: Operates on single pixels, i.e. $T: \mathbb{N} \to \mathbb{N}$ ← saw previously

   (b) Spatial Filtering: Operates on neighborhoods of pixels; $H: \mathbb{N}^{m \times n} \to \mathbb{N}$ ← now

2. Frequency Domain: Transforms to (& processes in) frequency domain; inverse transform to return to spatial domain

Basics of Spatial Filtering

Idea: Given integer coordinates (x, y) and an image $f(x,y)$, spatial filtering computes an output image $g(x,y)$ based on the neighborhood of $f(x,y)$ given by:

$$\text{neighborhood}[f(x,y)] = \{ f(x+s, y+t) : -a \leq s \leq a, -b \leq s \leq b \}$$

↖ for some $a, b \in \mathbb{N}$

where the size of the neighborhood is given by $m \times n = (2a+1) \times (2b+1)$.

Def: Linear Filtering

A linear spatial filter is a transformation of the form:

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

→ where:

(i) $w(s,t)$ is a given function $w: \mathbb{Z}^2 \to \mathbb{R}$ (sometimes called the mask/kernel), and

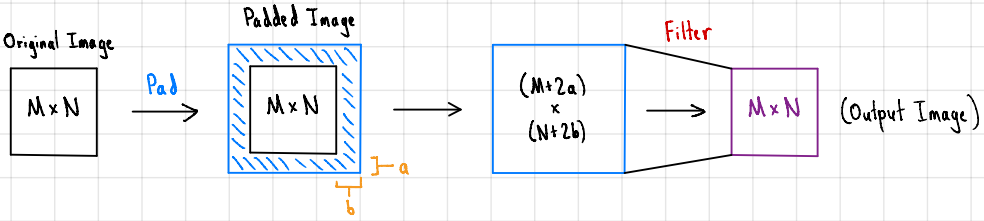
(ii) the neighborhood size $(2a+1) \times (2b+1)$ is referred to as the mask size/kernel size

(*) Padding

smaller than the original image (M×N)

Notice: The output image from applying an $m \times n$ $[(2a+1) \times (2b+1)]$ filter on an $M \times N$ image is $(M-2a) \times (N-2b)$.

→ To preserve image size, can pad the image [add additional elements on its border] by $2a$ rows & $2b$ columns.



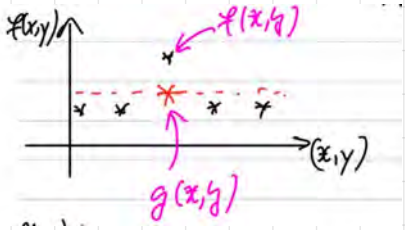
(*) 2 options for padding contents: (i) Zero padding: Pad with zeros (simplest)

(ii) Use some form of interpolation (ex: nearest-neighbor)

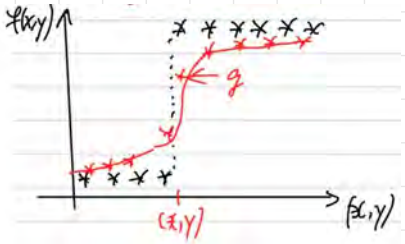
Smoothing Spatial Filters (§2.5)

Smoothing filters (linear & nonlinear) are useful for noise reduction in images

(*) Side effect: Blurring, particularly edge blurring



Noise Reduction (1D)



Edge Blurring (1D)

Linear Smoothing Filters

Def: The weighted average filter is given by:

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

[for some $(2a+1) \times (2b+1)$ kernel $v(s,t)$]

(*) Common 3×3 kernels: (i) $v_1(x,y) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ (ii) $v_2(x,y) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

(*) Ex. (Smoothing):

Original | 3×3 mask | 11×11 mask | 21×21 mask

Notice:

Larger mask → more blurring (generally)

Order-Statistic (Nonlinear) Smoothing Filters

Def: An **order-statistic filter** is a filter based on ordering the intensity values in a neighborhood $\{f(x+s, y+t) : s,t\}$ of $f(x,y)$.

☆ 1. **Median filter**: Replaces $f(x,y)$ with the **median** of $\{f(x+s, y+t)\}$

- Is especially good for reducing "salt-and-pepper noise" while minimizing edge blurring

(*) Salt-and-pepper noise simulates individual sensors/pixels dying within a camera/imager

Figure 12: From left to right: original image, by linear smoothing filter, by nonlinear median filter where the mask size is 21 × 21.

2. **Max filter**: Replaces $f(x,y)$ with the **maximum** of $\{f(x+s, y+t)\}$

3. **Min filter**: Replaces $f(x,y)$ with the **minimum** of $\{f(x+s, y+t)\}$

- Can use to find brightest & darkest parts of an image, respectively

Sharpening Spatial Filters (§2.6)

Motivation: Want to highlight transitions in intensity [corresponding to: object boundaries, details, etc. ... and also noise]

Idea: Can treat sharpening as the "inverse task" to smoothing [amplifying noise instead of reducing it]

- **Recall:** Weighted average filter (smoothing) takes a local average; is analogous to spatial integration, i.e. $\sum_{s,t} w(s,t) f(x+s, y+t) \approx \int\int f \cdot w$ ("a discrete version of")

→ **Principle:** For sharpening, we want to find a filter that acts like spatial differentiation

Spatial Differentiation (1D)

Know the continuous forms of differentiation: (i) $f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$, (ii) $f''(x) = \lim_{\delta\to 0} \frac{f'(x+\delta)-f'(x)}{\delta}$

→ For "discrete differentiation", approximate by taking $h=1$ & $\delta=1$:

(i) $\frac{df}{dx} \approx f(x+1) - f(x)$

(ii) $\frac{d^2f}{dx^2} \approx f'(x+1) - f'(x)$ $\xrightarrow{\text{substitute}}$ $\frac{d^2f}{dx^2} \approx f(x+1) - 2f(x) + f(x-1)$

Observe:
- Both $=0$ where intensity is constant
- Both $\neq 0$ at the start & end of an intensity "step/ramp"
- Key difference: $f'(x) \neq 0$ along "ramps" $f''(x) = 0$ along ramps

Spatial Differentiation (cont.)

Both the 1st & 2nd derivatives are used in image processing:

- 1st derivative used for edge detection ("identifying ramps")
- 2nd derivative used for image sharpening (identifying intensity transitions) ← what we want

2D Spatial Differentiation: Know the 2nd partial derivatives of $f(x,y)$ w.r.t. x, y:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) - 2f(x,y) + f(x-1,y) \quad ; \quad \frac{\partial^2 f}{\partial y^2} = f(x,y+1) - 2f(x,y) + f(x,y-1)$$

→ Use the 2D [discrete] Laplacian to filter:

$$\Delta f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f(x+1,y) + f(x,y+1) + f(x-1,y) + f(x,y-1) - 4f(x,y)$$

2 variations of Laplacian filter mask:

Isotropic / 5-Point Laplacian: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ (1)

Anisotropic / 9-Point [Digital] Laplacian: $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ (2)

For image sharpening: final output image is given by:

$$g(x,y) = f(x,y) - \Delta f(x,y)$$

← corresponds to composite mask:

$w = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ (5-Point)

or

$w = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ (9-Point)

(*) Ex:

(1) 5-Point

(2) 9-Point

Generally similar results in practice; anisotropic/9-point may be very slightly preferred

Unsharp Masking

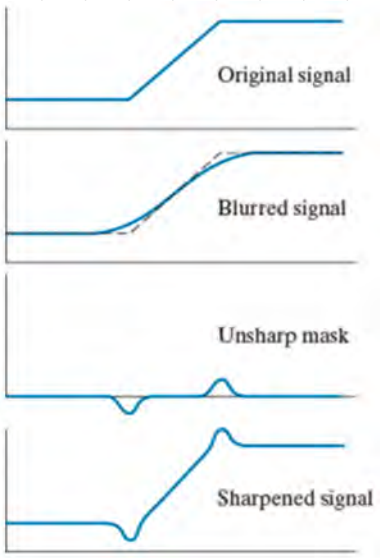
Can use blurring to sharpen images via unsharp masking:

1. Blur the original image $[f]$ to obtain a blurred image $\bar{f}$
2. Subtract the blurred image from the original to obtain a mask:

$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

3. Add the (optionally scaled) mask back to the image:

$$g(x,y) = f(x,y) + k \cdot g_{mask}(x,y) \quad \text{for } k \geq 0$$



Gradients for Image Sharpening

For a 2D image, can use the gradient/1st-order derivative $\nabla f = \langle \underbrace{\frac{\partial f}{\partial x}}_{f_x}, \underbrace{\frac{\partial f}{\partial y}}_{f_y} \rangle$ for edge detection.

In particular, can look at the magnitude (length) $|\nabla f|$ of ∇f: $|\nabla f| = \sqrt{(f_x)^2 + (f_y)^2} \approx |f_x| + |f_y|$ ← use this to approximate

$$\longrightarrow \quad g(x,y) = |\nabla f| = \begin{cases} 0 & (x,y) \text{ in a homogeneous region of } f \\ > 0 & (x,y) \text{ at an edge within } f \end{cases}$$

∃ multiple expressions for/ways to approximate f_x & f_y; ex (3x3):

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} := \begin{bmatrix} f(x-1,y-1) & f(x-1,y) & f(x-1,y+1) \\ f(x,y-1) & f(x,y) & f(x,y+1) \\ f(x+1,y-1) & f(x+1,y) & f(x+1,y+1) \end{bmatrix}$$

→ 1. One-sided finite difference approx.

$f_x \approx z_8 - z_5$; $f_y \approx z_6 - z_5$

2. Central finite difference approx.

$f_x \approx z_8 - z_2$; $f_y \approx z_6 - z_4$

3. Robin cross-gradient operator

$f_x \approx z_9 - z_5$; $f_y \approx z_8 - z_6$

4. Prewitt operator:

$$f_x \approx (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$
$$f_y \approx (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

5. Sobel operator:

$$f_x \approx (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$f_y \approx (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Remarks:

(i) The sum of coefficients in each mask is 1

(ii) To better visualize edges, can threshold at some T:

$$g(x,y) = \begin{cases} 255 \text{ [white]} & |\nabla f| \leq T \text{ (edge)} \\ 0 \text{ [black]} & |\nabla f| > T \text{ (!edge)} \end{cases}$$

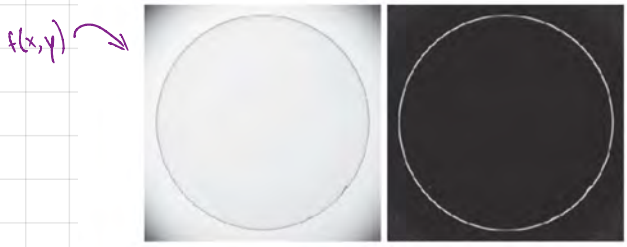


Figure 14: Edge detection using $|\nabla f|$ for an image of contact lens.

Filtering in the Frequency Domain (Lec. 8-16)

Background (§3.1)

Complex Numbers

A complex number $c \in \mathbb{C}$ can be expressed by:

$$c = R + iI$$

where $R, I \in \mathbb{R}$ represent the real and imaginary components, respectively, and i satisfies $i^2 = -1$.

Also define the [complex] conjugate $\bar{c} \in \mathbb{C}$ and magnitude $|c| \in \mathbb{R}$ of a complex number $c \in \mathbb{C}$

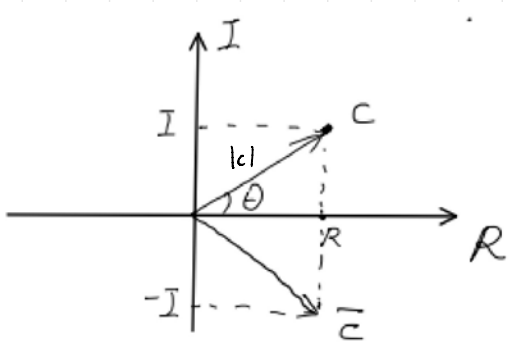
$$c = R + iI \implies \bar{c} := R - iI$$

$$\rightarrow |c|^2 = c \cdot \bar{c} = (R+iI)(R-iI) = R^2 + I^2$$

Recall (Euler's formula): For $\theta \in \mathbb{R}$: $\boxed{e^{i\theta} = \cos(\theta) + i\sin(\theta)}$

$\rightarrow$ Can express a complex number using polar coordinates:

$$\boxed{c = |c| \cdot e^{i\phi}}$$



- $|c| \in \mathbb{R}$; may also be denoted by r [radius]
- $\phi \in (-\pi, \pi]$, $\phi = \arctan(I/R)$ [may be $\pm\pi$]
  - called the phase

(*) Complex functions: Can express a complex function $F: \mathbb{R}^n \to \mathbb{C}$ by:

$$F(u) = R(u) + iI(u)$$

where $R, I: \mathbb{R}^n \to \mathbb{R}$ are real functions.

Can write its conjugate and magnitude as before:

$$\overline{F(u)} = R(u) - iI(u) \qquad |F(u)|^2 = R(u)^2 + I(u)^2$$

Background (cont.)

The Impulse Function

The continuous [1D] impulse function is (essentially) defined by:

$$\delta(t) = \begin{cases} \infty & t = 0 \\ 0 & t \neq 0 \end{cases}$$

Intuitively: define (for $a > 0$):

$$\delta^a(t) = \begin{cases} 1/2a & -a \le t \le a \\ 0 & \text{otherwise} \end{cases}$$

[Observe: $\int_{-\infty}^{\infty} \delta^a(t)\,dt = 1$]

Then, as $a \to 0$ $(1/2a \to \infty)$

$\delta^a(t) \xrightarrow{a \to 0} \delta(t)$

for a continuous variable t, such that $\delta(t)$ satisfies:

$$\int_{-\infty}^{\infty} \delta(t)\,dt = 1$$

In particular, the impulse function has the sifting property:

$$\int_{-\infty}^{\infty} f(t)\,\delta(t - t_0)\,dt = f(t_0) \quad \forall\, t_0 \in \mathbb{R} \text{ s.t. } f(t) \text{ continuous at } t = t_0$$

Can define the discrete 1D + continuous & discrete 2D impulse functions analogously:

Type	Function	Integral	Sifting Property
Continuous (1D)	$\delta(t) = \begin{cases} \infty & t=0 \\ 0 & t \neq 0 \end{cases}$	$\int_{-\infty}^{\infty} \delta(t)\,dt = 1$	$\int_{-\infty}^{\infty} f(t)\,\delta(t-t_0)\,dt = f(t_0)$
Continuous (2D)	$\delta(t,z) = \begin{cases} \infty & (t,z) = 0 \\ 0 & (t,z) \neq 0 \end{cases}$	$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(t,z)\,dt\,dz = 1$	$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t,z)\,\delta(t-t_0, z-z_0)\,dt\,dz = f(t_0, z_0)$
Discrete (1D)	$\delta(x) = \begin{cases} 1 & x=0 \\ 0 & x \neq 0 \end{cases}$	$\sum_x \delta(x) = 1$	$\sum_x f(x)\,\delta(x - x_0) = x_0$
Discrete (2D)	$\delta(x,y) = \begin{cases} 1 & x=y=0 \\ 0 & \text{otherwise} \end{cases}$	$\sum_x \sum_y \delta(x,y) = 1$	$\sum_x \sum_y f(x,y)\,\delta(x-x_0, y-y_0) = f(x_0, y_0)$

The Fourier Series

Given a periodic function $f_T(t)$ with period T (s.t. $f_T(t) = f_T(t+T)\ \forall\, t \in \mathbb{R}$), can express f_T in terms of its Fourier series:

$$f_T(t) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{2\pi n}{T}t} \quad \text{where} \quad c_n = \frac{1}{T}\int_{-T/2}^{T/2} f_T(t)\,e^{-i\frac{2\pi n}{T}t}\,dt \quad \text{for } n = 0, \pm 1, \pm 2, \ldots$$

("Fourier series coefficients")

(*) Gibbs phenomenon: Given a periodic function f_T with jump discontinuities, taking a Fourier series approximation of $f_T(t)$ with finitely-many Fourier series coefficients will result in oscillations around the discontinuity (more coefficients → narrower, taller band of oscillations)

Background (cont.)

The Fourier Transform [1D]

To obtain the Fourier transform for general (incl. aperiodic) functions f, take the Fourier series limit as $T \to \infty$:

$$\mathcal{F}[f(t)] = \lim_{T\to\infty} \int_{-T/2}^{T/2} f_T(t) e^{-i\frac{2\pi n}{T}t} dt \longrightarrow \boxed{F(\mu) = \mathcal{F}[f(t)] = \int_{-\infty}^{\infty} f(t) e^{-i2\pi\mu t} dt}$$ (Fourier transform)

Also have the inverse Fourier transform [IFT]:

$$\boxed{f(t) = \mathcal{F}^{-1}[F(\mu)] = \int_{-\infty}^{\infty} F(\mu) e^{i2\pi\mu t} d\mu}$$ (Inverse Fourier Transform)

(*) Can find expressions for the real & imaginary components of $F(\mu)$: $F(\mu) = R(\mu) + iI(\mu)$ where:

- $R(\mu) = \int_{-\infty}^{\infty} F(\mu)\cos(i2\pi\mu t) dt$
- $I(\mu) = \int_{-\infty}^{\infty} F(\mu)\sin(i2\pi\mu t) dt$

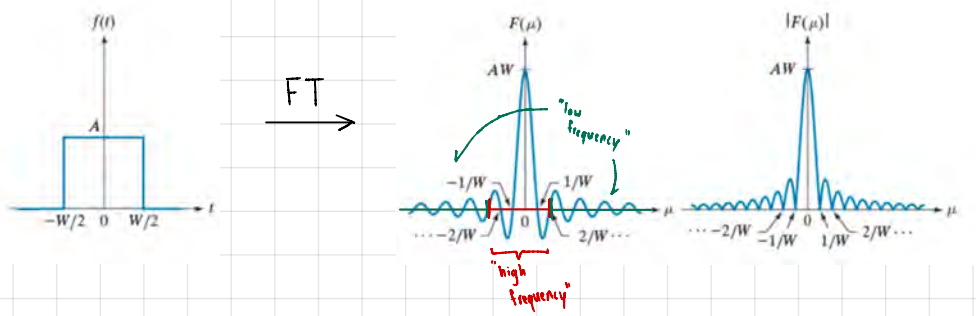
Define the [Fourier/frequency] spectrum of the Fourier transform by:

$$\boxed{|F(\mu)| = \sqrt{R(\mu)^2 + I(\mu)^2}}$$

The Fourier Transform (Motivation)

The Fourier transform moves from the spatial/time domain [$f(t)$] to the frequency domain [$F(\mu)$]:

- Low-frequency components ($|F|$ low) correspond to slowly-varying regions in the image (i.e. smooth regions)
- High-frequency components (high $|F|$) correspond to fast-varying components of the image (e.g. noise, edges)
- Filtering in the frequency domain is often faster (and potentially more effective) than in the spatial domain



Background (cont.)

Convolution

Let $f(t)$, $h(t)$ be two continuous functions on $(-\infty, \infty)$. Then their convolution (1D, continuous) is defined by:

$$(f * h)(t) = \int_{-\infty}^{\infty} f(\tau) h(t-\tau)\, d\tau$$

Properties of convolution:
- (i) $f * h = h * f$ (Commutativity)
- (ii) $f_1 * (f_2 * f_3) = (f_1 * f_2) * f_3$ (Associativity)
- (iii) $f_1 * (f_2 + f_3) = f_1 * f_2 + f_1 * f_3$ (Distributivity)

$\Downarrow$

The Convolution Theorem

$$\mathcal{F}[f * h] = \mathcal{F}[f] \cdot \mathcal{F}[h]$$

(*) Note: Also have a (less used) converse:

$$\mathcal{F}[f \cdot h] = \mathcal{F}[f] * \mathcal{F}[h]$$

Proof: (Convolution Thm.)

$$\mathcal{F}[f * h] = \int_{-\infty}^{\infty} [f*h](t) e^{-i2\pi\mu t}\, dt$$

$$= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\tau) h(t-\tau)\, d\tau \right] e^{-i2\pi\mu t}\, dt$$

note: $[\cdot]$ is indep. of τ!

$$[s = t-\tau] = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} h(s) e^{-i2\pi\mu s}\, ds \right] f(\tau) e^{-i2\pi\mu\tau}\, d\tau$$

Dot product much easier to compute than convolution integral

$$= \left[\int_{-\infty}^{\infty} h(s) e^{-i2\pi\mu s}\, ds \right] \left[\int_{-\infty}^{\infty} f(\tau) e^{-i2\pi\mu\tau}\, d\tau \right] = \mathcal{F}[h(t)] \cdot \mathcal{F}[f(t)]$$

▨

(*) Note: Many previously-seen operations (e.g. multiplying by a kernel matrix) can be expressed as convolutions.

Sampling

Previously, saw the Fourier transform for continuous functions; in practice, can only work with samples of a function.

Sampling: Want to approximate a function f using discrete samples (taken in intervals ΔT):

$$f(t) \xrightarrow{\text{sample}} f_{\text{sample}} = \sum_{n=-\infty}^{\infty} f(t)\, \delta(t - n\Delta T)$$

Can look at the Fourier transform $\tilde{F}$ of f_{sample}:

$$\tilde{F}(\mu) = \mathcal{F}[f_{\text{sample}}] = \int_{-\infty}^{\infty} f_s(t) e^{-i2\pi\mu t}\, dt$$

$$= \sum_{n=-\infty}^{\infty} f(t) e^{-i2\pi\mu t}\, \delta(t - n\Delta T)$$

$$= \sum_{n=-\infty}^{\infty} f(n\Delta T) e^{-i2\pi\mu n \Delta T}$$

$$= \frac{1}{T}\left[F(\mu) * \delta_{2\pi\Delta T}(\mu) \right] = \frac{1}{T} \sum_{n=-\infty}^{\infty} F(\mu + 2\pi n \Delta T)$$

(*) Notation: $\delta_T(t) = \sum_n \delta(t - nT)$

(*) Notice: $\tilde{F}$ is continuous and periodic with period $1/\Delta T$

Background (cont.)

Sampling (cont.)

$$\tilde{F}(\mu) = \frac{1}{T}\sum_{n=-\infty}^{\infty} F(\mu + 2\pi n \Delta T)$$

← Copies of $F(\mu)$, spaced $1/\Delta T$ apart

Let $\mu_{max} = \underset{F(\mu)\neq 0}{\arg\max} |\mu|$:

↳ take "central copy" as $F(\mu)$

Case 1: $1/\Delta T \geq 2\mu_{max}$ → can perfectly reconstruct $F(\mu)$ from $\tilde{F}(\mu)$

Case 2: $1/\Delta T < 2\mu_{max}$ → copies of $F(\mu)$ overlap; can no longer (generally) find $F(\mu)$

↑ "aliasing"

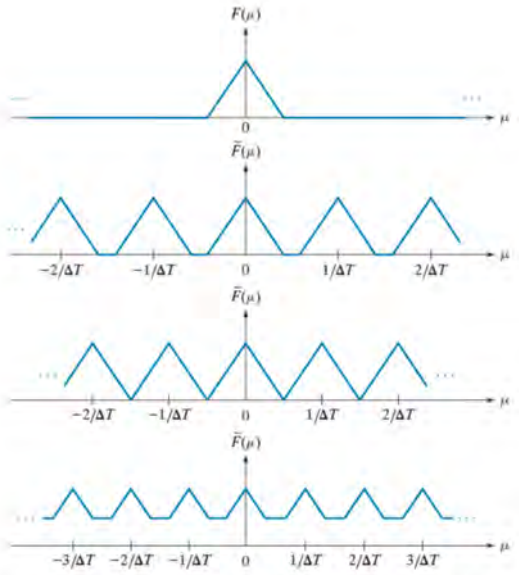


Figure 15: Exact Fourier transform, Sampling with $1/\Delta T > 2\mu_{max}$, $= 2\mu_{max}$, and $< 2\mu_{max}$

Main Idea: The sampling rate ΔT affects the accuracy of our approximation of f.

The Discrete Fourier Transform

Recall the expressions $\tilde{f}$ and $\tilde{F}$ for the approximation of a function $f(t)$ & its Fourier transform $F(\mu)$ via discrete samples of f; can define the discrete F.T./DFT (+ inverse discrete F.T./IDFT) via similar expressions:

Def: The Discrete Fourier Transform [1D]

Let $n = 0, 1, \ldots, M-1$ and $m = 0, 1, \ldots, M-1$ be integer-valued discrete spatial & frequency variables, resp.

Then the discrete Fourier transform & inverse discrete Fourier transform are given by:

Discrete Fourier Transform:

$$F(\mu) = \sum_{x=0}^{M-1} f(x) e^{-i\frac{2\pi\mu x}{M}}$$

← DFT[f]

Inverse Discrete Fourier Transform:

$$f(x) = \frac{1}{M}\sum_{\mu=0}^{M-1} F(\mu) e^{i\frac{2\pi\mu x}{M}}$$

↖ IDFT[F]

Background (cont.)

The Discrete Fourier Transform (cont.)

Notice: Both $F(\mu)$ and $f(x)$ [as given by the DFT & IDFT, respectively] are periodic with period M.

$\longrightarrow$ Consequently: can obtain all values of F, f on $\mathbb{R}$ within a single period M.

Discrete Convolution

Given discrete variable $x = 0, 1, \ldots, M-1$, the [1D, discrete] convolution $[f*h](x)$ of two functions f, h is defined by:

$$[f * h](x) = \sum_{m=0}^{M-1} f(m) \cdot h(x-m)$$

$\longrightarrow$ Discrete Convolution Theorem:

$$DFT(f * h) = DFT(f) \cdot DFT(h)$$

Notes: (i) If $f(x)$ and $h(x)$ are periodic, then $[f*h](x)$ is also periodic

(ii) In order to compute $[f*h](x)$, need values for $h(x)$ from $x = -(M-1)$ to $x = M+1$

Intro to 2D Fourier Analysis (§3.2)

The Fourier Transform [2D]

For a 2D continuous function $f(t,z)$, its Fourier transform $\mathcal{F}[f(t,z)]$ is defined by:

$$F(\mu,\nu) = \mathcal{F}[f(t,z)] := \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t,z)\, e^{-i2\pi(\mu t+\nu z)}\, dt\,dz \qquad \text{(2D Fourier Transform)}$$

and the [2D] inverse Fourier transform is given by:

$$f(t,z) = \mathcal{F}^{-1}[F(\mu,\nu)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)\, e^{i2\pi(\mu t+\nu z)}\, d\mu\,d\nu \qquad \text{(2D Inverse Fourier Transform)}$$

Prop: The Fourier transform of f is well-defined if $f \in L^1(\mathbb{R}^2)$, i.e.: $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |f(t,z)|\, dt\,dz$

The 2D Discrete Fourier Transform

Similar to 1D case: for 2D functions/images, need to choose sampling intervals ΔT, ΔZ small enough to be able to properly capture information in the frequency domain.

The Discrete Fourier Transform [2D]

Let $x = 0, 1, \dots, M-1$ and $y = 0, 1, \dots, N-1$ be two discrete spatial variables, and let $\mu = 0, 1, \dots, M-1$ and $\nu = 0, 1, \dots, N-1$ be two discrete frequency variables.

Then the 2D discrete Fourier transform and 2D inverse discrete Fourier transform are defined by:

2D Discrete Fourier Transform:

$$F(\mu,\nu) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-i2\pi\left[\mu\frac{x}{M}+\nu\frac{y}{N}\right]}$$

2D Inverse Discrete Fourier Transform:

$$f(x,y) = \frac{1}{MN}\sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1} F(\mu,\nu)\, e^{i2\pi\left[\mu\frac{x}{M}+\nu\frac{y}{N}\right]}$$

Notice (Periodicity): (i) $F(\mu,\nu) = F(\mu+k_1M,\nu) = F(\mu,\nu+k_2N) = F(\mu+k_1M,\nu+k_2N)$

(ii) $f(x,y) = f(x+k_1M,y) = f(x,y+k_2N) = f(x+k_1M,y+k_2N)$

Intro to 2D Fourier Analysis (cont.)

2D Convolution

2D Continuous Convolution: Define 2D continuous convolution by:

$$[f * h](t,z) = \int_{\mathbb{R}^2} f(\tau,\zeta)\, h(t-\tau, z-\zeta)\, d\tau\, d\zeta$$

$\longrightarrow$ Convolution Theorem: $\mathcal{F}[f * h] = \mathcal{F}[f] \cdot \mathcal{F}[h]$

2D Discrete Convolution: Let $x = 0, 1, \dots, M-1$ and $y = 0, 1, \dots, N-1$. Define 2D discrete convolution:

$$[f * h](x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)\, h(x-m, y-n)$$

$\longrightarrow$ Convolution Theorem: $DFT[f * h] = DFT[f] \cdot DFT[h]$

Relation to Linear Spatial Filtering

Recall: In linear spatial filtering, spatial-domain transformation is given by:

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

→ **Notice**: If $w(s,t) = w(-s,-t)$, this is equivalent to:

$$g = f * w = \mathcal{F}^{-1}[\mathcal{F}\{f * w\}] = \mathcal{F}^{-1}[\mathcal{F}[f] \cdot \mathcal{F}[w]]$$ (Filtering in the frequency domain)

Filtering in the Frequency Domain

To compute a transformation $f * h$ (for some mask h) in the frequency domain:

1. Compute $F = \mathcal{F}[f]$
2. Compute $G = F \cdot H$, where $H = \mathcal{F}[h]$
3. Compute $g = \mathcal{F}^{-1}[G]$

↳ Usually more efficient than computing $f * h$ directly

Fourier Transform: Additional Properties (§3.3)

F.T. of the Impulse Function:

$$\mathcal{F}[\delta] = 1$$

↑ True for 1D & 2D, continuous & discrete

(*) Proof: Know: (i) $\mathcal{F}[\delta * f] = \mathcal{F}[\delta] \cdot \mathcal{F}[f]$ (Convolution Theorem)

(ii) $\mathcal{F}[\delta * f] = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} \delta(\tau) f(t-\tau) d\tau \right] e^{-i2\pi\mu t} dt = \int_{-\infty}^{\infty} f(t) e^{-i2\pi\mu t} dt = \mathcal{F}[f]$

$\rightarrow \mathcal{F}[\delta] = 1$. ▨ [Proof for 1D continuous case]

Notice: For function f with F.T. $F = \mathcal{F}[f]$:

$$f_{mean} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \quad \Rightarrow \quad F(0,0) = MN f_{mean}$$

↑ Also holds for 1D/2D + cts./disc. cases

Remarks: (i) $F(0,0)$ is also called the direct circuit (dc) component of the Fourier transform

(ii) Important: Typically, $|F(0,0)|$ is the largest component of the spectrum → generally: brightest parts of Fourier spectrum occur at the corners (corresponding to $F(0,0) = F(M,0) = F(0,N) = F(M,N)$, low frequency components/smooth regions of f)

- In contrast: $F(M/2, N/2)$ are often the darkest region; corresponds to high frequencies (e.g. noise, edges)

→ Often: (want to) shift FT spectrum by $(M/2, N/2)$ to place $F(0,0)$ [i.e. brightest region] in the center

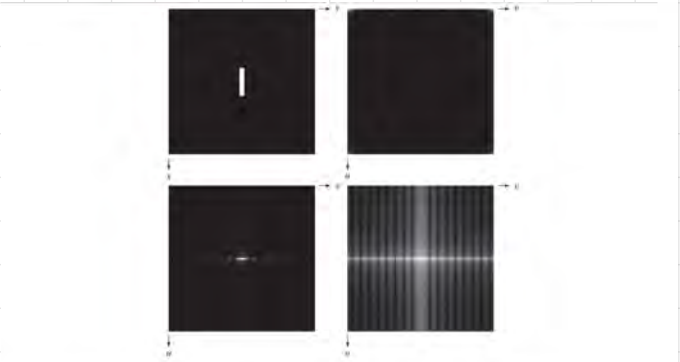


Figure 18: First row: original image and its Fourier transform; second row: shifted Fourier transform and the log of shifted Fourier transform.

Translation Properties

$$\mathcal{F}\left\{ f(x,y) e^{i2\pi\left[\mu_0 \frac{x}{M} + \nu_0 \frac{y}{N}\right]} \right\} = F(\mu - \mu_0, \nu - \nu_0) \quad \Rightarrow \quad \mathcal{F}\left[f(x,y)(-1)^{x+y} \right] = F\left(\mu - \frac{M}{2}, \nu - \frac{N}{2}\right)$$

Notice: $f(x-x_0, y-y_0) \overset{\text{F.T.}}{\Leftrightarrow} F(\mu,\nu) e^{-i2\pi\left[\mu \frac{x_0}{M} + \nu \frac{y_0}{N}\right]}$

$\rightarrow |\mathcal{F}[f(x,y)]| = |\mathcal{F}[f(x-x_0, y-y_0]|$ (Fourier spectrum is not affected by translation)

Fourier Transform: Additional Properties (cont.)

Conjugate Symmetric Property:

If $f(x,y)$ is real [i.e. $f \in \mathbb{R}$], then:

$$\overline{F(\mu,\nu)} = F(-\mu,-\nu) \quad \Rightarrow \quad |F(\mu,\nu)| = |\overline{F(\mu,\nu)}| = F(-\mu,-\nu)$$

(*) Proof: Via the fact that $e^{-i\theta} = \overline{e^{i\theta}}$

Rotation Property

Using polar coordinates:

(i) $x = r\cos\theta$; $y = r\sin\theta$ $\quad [f(x,y) \to f(r,\theta)]$

(ii) $\mu = w\cos\phi$; $\nu = w\sin\phi$ $\quad [F(\mu,\nu) \to F(w,\phi)]$

$$\longrightarrow \quad f(r,\theta+\theta_0) \overset{FT}{\longleftrightarrow} F(w,\phi+\theta_0)$$

(*) Proof: Via polar change of variables

(*) Notes: (i) $w(\mu,\nu) = |F(\mu,\nu)|$ is the Fourier spectrum

(ii) $\phi(\mu,\nu)$ is called the phase angle

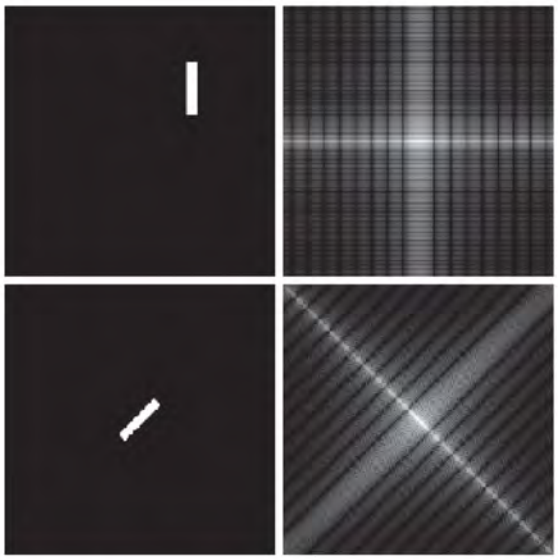


Figure 19: Fourier transform of a rotated and shifted image.

(*) Separability of the 2D DFT

Can write the 2D DFT as a sum of 1D DFTs:

$$F(\mu,\nu) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\, e^{-2\pi i\left[\mu\frac{x}{M} + \nu\frac{y}{N}\right]} = \sum_{x=0}^{M-1}\left[\sum_{y=0}^{N-1} f(x,y)\, e^{-2\pi i\nu\frac{y}{N}}\right] e^{-2\pi i\mu\frac{x}{M}}$$

$$\longrightarrow \quad F(\mu,\nu) = \sum_{x=0}^{M-1} F(x,\nu)\, e^{-2\pi i\mu\frac{x}{M}} \qquad [F(x,\nu): \text{1D DFT w.r.t. } y]$$

Intro to Frequency Filtering (§3.4)

General idea: Want to use the DFT to convert a spatial image $f(x,y)$ into its frequency-domain representation $F(\mu,\nu)$, then use the IDFT to obtain the filtered spatial output image $g(x,y)$.

Recall: (i) By the properties of $F(0,0)$, $|\mathcal{F}[f(x,y)]|$ is brightest at the four corners (and darkest at the center)

(ii) By the translation property, $|\mathcal{F}[f(x,y)(-1)^{x+y}]|$ is brightest at the center and darkest at the corners

→ Often, prefer to apply the DFT to $\tilde{f}(x,y) = f(x,y)(-1)^{x+y}$, rather than to $f(x,y)$ directly, so that:

1. The region near the center of $\tilde{F}$ (i.e. low-frequency components of F) correspond to uniform regions in the image f
2. Regions away from the center of $\tilde{F}$ (high-frequency components of F) correspond to edges + noise in f

Notation: Given a spatial filter $h(x,y)$, we write $H := \mathcal{F}[h]$ to refer to its filter transfer function in the frequency domain

Frequency Filtering Steps

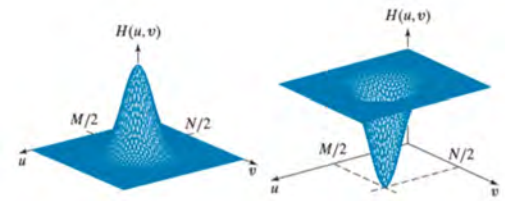
1. Multiply $f(x,y)$ by $(-1)^{x+y}$
2. Compute $F(\mu,\nu) = \text{DFT}[f(x,y)(-1)^{x+y}]$
3. Multiply $F(\mu,\nu)$ by a filter $H(\mu,\nu)$
4. Compute $\text{IDFT}[F(\mu,\nu)H(\mu,\nu)]$, and take the real part.
5. Multiply the result by $(-1)^{x+y}$ to obtain the filtered image

(*) Notes

- The Fourier transform $F(\mu,\nu)$ obtained in this process is actually $F(\mu - M/2, \nu - N/2)$.
  - (This also is true for $H(\mu,\nu)$)
- In theory, $\text{IDFT}[F \cdot H]$ should be a real image; however, due to rounding errors (or otherwise), may be complex in practice
  → keep only real component
  - Using a discrete version (DFT) of a continuous function (FT) can also lead to error

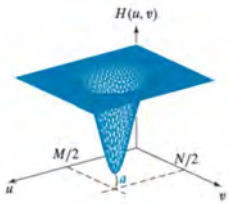
Def: A lowpass filter is a filter that leaves the low-frequency component unchanged (i.e. the center of $\tilde{F}(\mu,\nu)$), but attenuates the high-frequency components [near corners/edges]

Conversely: a highpass filter attenuates low frequencies, while leaving high-frequency components unchanged.



Lowpass Highpass

↑ Note: Lowpass, highpass both popular engineering terminology



"Offset Highpass"

Figure 22: Results of a lowpass, highpass, and an offset highpass filter.

Intro to Frequency Filtering (cont.)

Frequency filtering – output image given by:

$$g(x,y) = (-1)^{x+y} \cdot \text{Real}\left\{\text{IDFT}\left[H(\mu,\nu) \cdot \mathcal{F}\left\{(-1)^{x+y} f(x,y)\right\}\right]\right\}$$

Basic Filter Examples

1. A "special" notch filter:

$$H(\mu,\nu) = \begin{cases} 0 & (\mu,\nu) = (M/2, N/2) \\ 1 & \text{otherwise} \end{cases}$$

→ outputs an image with 0 average

2. **Gaussian filter:**

$$h(x) = \sqrt{2\pi\sigma^2}\, A e^{-2\pi^2\sigma^2 x^2} \longrightarrow H(\mu) = A e^{-\frac{\mu^2}{2\sigma^2}}$$

[A, σ parameters]

(Lowpass ⇒ smoothing filter)

Gaussian/Gaussian-based filters: very popular (DFT, IDFT very easy to compute)

3. **Difference of Gaussians ("DoG")**

$$h(x) = \sqrt{2\pi\sigma_1^2}\, A_1 e^{-2\pi^2\sigma_1^2 x^2} - \sqrt{2\pi\sigma_2^2}\, A_2 e^{-2\pi^2\sigma_2^2 x^2} \longrightarrow H(\mu) = A_1 e^{-\frac{\mu^2}{\sigma_1^2}} - A_2 e^{-\frac{\mu^2}{\sigma_2^2}}$$

[$A_1, A_2, \sigma_1, \sigma_2$ parameters]

(Can be either a lowpass or highpass filter, depending on choice of parameters)

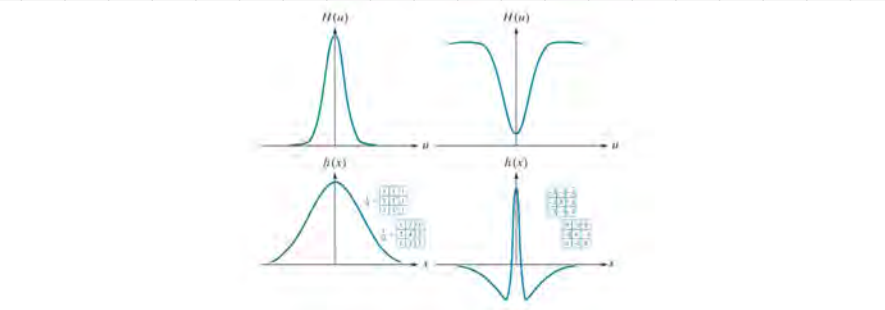


Figure 23: Left: Single Gaussian in the frequency and spatial domain which are lowpass filters. Right: composition of two Gaussians in the frequency and spatial domain which are highpass filters.

(*) To derive a generic highpass filter:

$$H_{hp}(\mu) = 1 - H_{lp}(\mu)$$ where $H_{lp}(\mu)$ is any lowpass filter

Smoothing Lowpass Filters (§3.5)

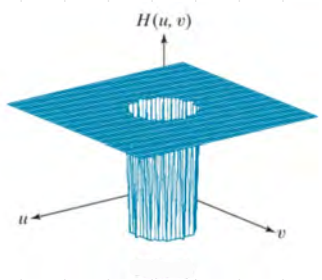
1. **Ideal Lowpass Filter (ILPF)** - Define distance function:

$$D(u,v) := \sqrt{\left(u - \tfrac{M}{2}\right)^2 + \left(v - \tfrac{N}{2}\right)^2}$$

→ The ideal lowpass filter is given by:

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & \text{otherwise} \end{cases}$$

[D_0: cutoff frequency, parameter]



The Ideal Lowpass Filter (Frequency Domain)

Ideal LPF - Issues

(i) **Bad results**: Applying the ideal LPF to an image results in artifacts around objects in the final image ("ringing")

(ii) **Inefficient**: Applying the ideal LPF to an image removes a relatively low % of information in the frequency domain (RHS, 2nd image: ~13.3%)

→ Want a "better" lowpass filter

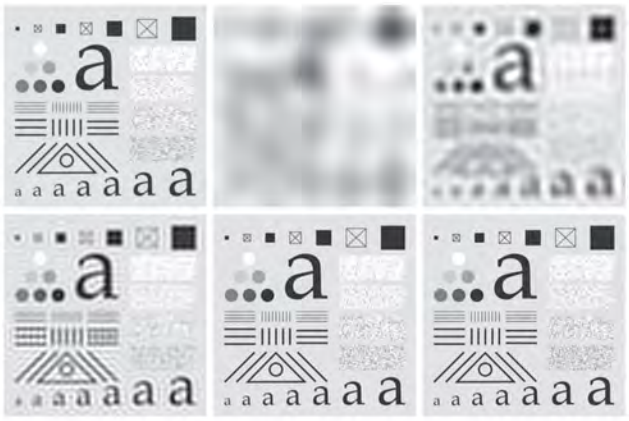


Figure 24: Results by Ideal lowpass filter with increasing D_0 (cut-off frequency).

Figure 25: Ideal lowpass filter in the frequency domain, spatial domain, and along a horizontal line in the spatial domain where the ringing effect is evident.

Two alternative types of lowpass filter: (i) Gaussian filter, (ii) Butterworth filter

Smoothing Lowpass Filters (cont.)

2. Gaussian Lowpass Filter (GLPF)

D_0: variance of the Gaussian f'n [larger → less filtering]

$$H(u,v) = e^{\frac{-D^2(u,v)}{2D_0^2}}$$

[D_0 a parameter]

Notice: $H(M/2, N/2) = 1$; decays to 0 smoothly as $D(u,v) \uparrow$

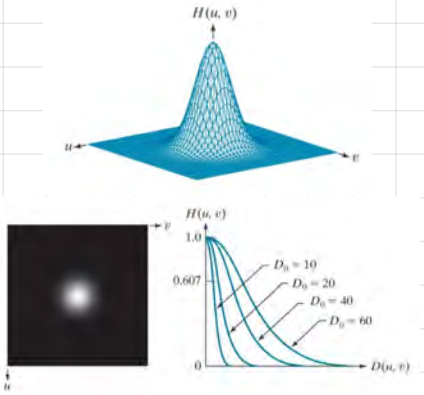


Figure 26: Gaussian lowpass filter in 2D, a contour plot, and a 1D plot for different D_0.

3. Butterworth Lowpass Filter

$$H(u,v) = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}}$$

with parameters:

(i) $D_0 > 0$

(ii) $n > 0$ a positive integer [called the order of the filter] (usually 2 or 3)

Note: $H(M/2, N/2) = 1$, $\to 0$ as $D(u,v) \uparrow$ [similar to Gaussian LPF]

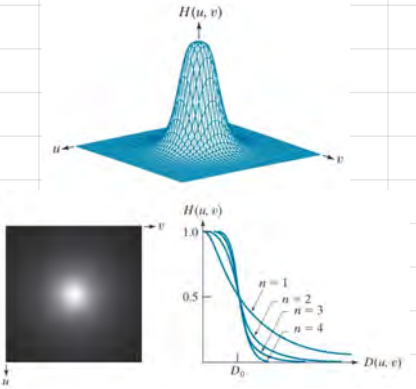


Figure 28: Butterworth lowpass filter in 2D, a contour plot, and a 1D plot for different D_0..

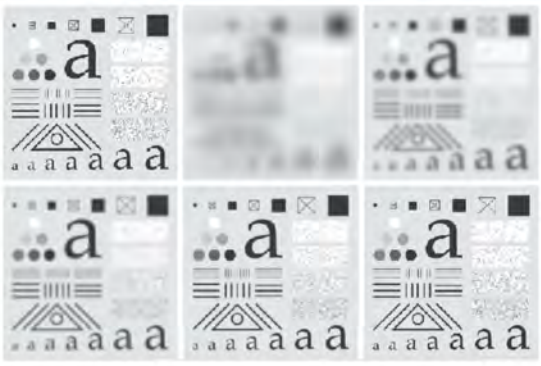


Figure 27: Application of Gaussian filter with increasing D_0.

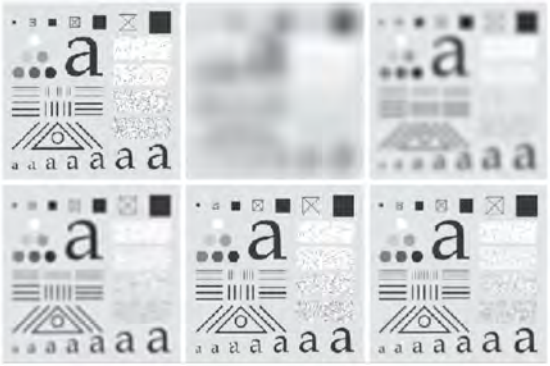


Figure 29: Application of Butterworth lowpass filter with increasing D_0.

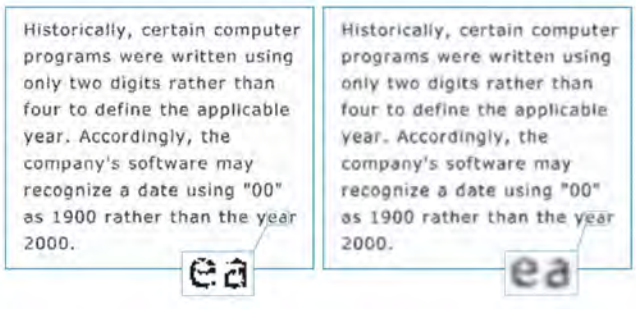


Figure 30: One simple application of a Gaussian lowpass filter to recover broken characters.

Remark: Unlike the ideal LPF, both the Gaussian & Butterworth LPFs are analytic (decay smoothly to 0)

→ intuitively: unsmooth decay (non-analytic filter) ⇒ instability in output image [see ILPF]

Sharpening Highpass Filters (§3.6)

Recall: To obtain a highpass filter, can simply take $H_{hp} := 1 - H_{lp}$ for some lowpass filter H_{lp}

Basic Highpass Filters

1. Ideal Highpass Filter (IHPF)

$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & \text{otherwise} \end{cases}$$

↑ D_0 larger ⇒ more info. filtered out

2. Gaussian Highpass Filter

$$H(u,v) = 1 - e^{\frac{-D(u,v)^2}{D_0}}$$

3. Butterworth Highpass Filter

$$H(u,v) = \frac{1}{1 + \left(\frac{D_0}{D(u,v)}\right)^{2n}}$$

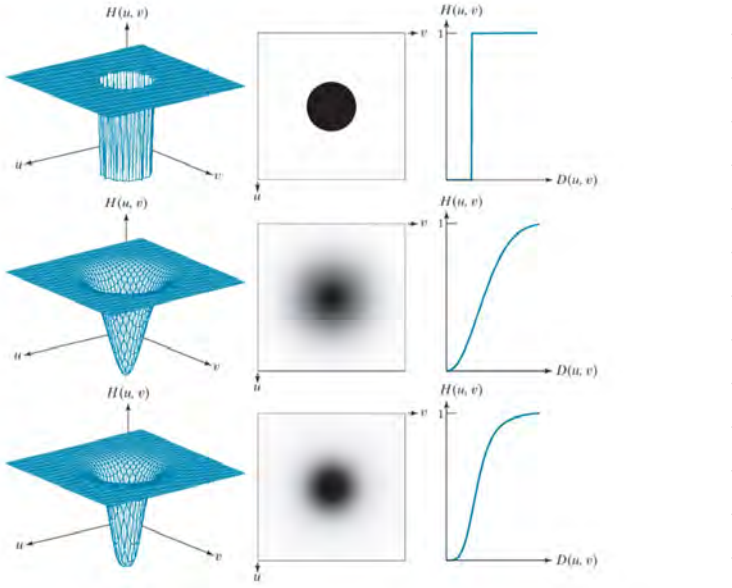


Figure 31: From top to bottom: ideal, Gaussian, and Butterworth highpass filter. From left to right: the value of the filter in the frequency domain and its cross sections.

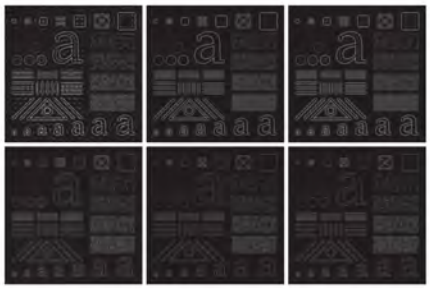


Figure 33: From left to right: application of ideal, Gaussian, and Butterworth highpass filter with $D_0 = 60$ (first row), $D_0 = 160$ (second row).

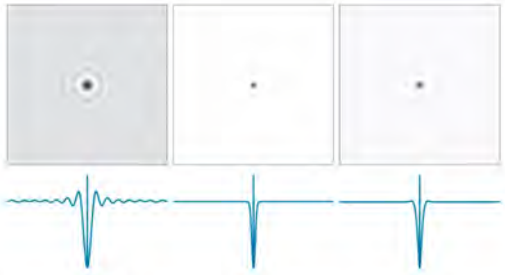


Figure 32: Ideal, Gaussian, and Butterworth highpass filter in the spatial domain in a two and one-dimensional cross-section

Notice:

$$H_{hp} = 1 - H_{lp} \Rightarrow \mathcal{F}^{-1}[H_{hp}] = \mathcal{F}^{-1}[1] - \mathcal{F}^{-1}[H_{lp}] = \delta_0 - h_{lp}$$

⇝ Consequently: Spatial domain version will have negative values [not representable by a digital image]

Common strategy: Set all negative values in output image to be 0, and set all positive values to 1 [L-1] ⟶ thresholding

Figure 34: From left to right: original thumbprint, results from Gaussian highpass filter, after thresholding.

Sharpening Highpass Filters (cont.)

The Laplacian Filter

Previously, defined the Laplacian filter for sharpening in the spatial domain

→ Prop.: The 2D continuous Laplacian filter has frequency-domain representation given by:

$$\mathcal{F}[\Delta f] = -4\pi^2(\mu^2+\nu^2)F(\mu,\nu)$$

Proof: Per the IFT, have that

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)e^{2\pi i(\mu x+\nu y)}\,d\mu\,d\nu \quad \Rightarrow \quad \frac{\partial}{\partial x}f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)(2\pi i\mu)e^{2\pi i(\mu x+\nu y)}\,d\mu\,d\nu$$

$$\Rightarrow \frac{\partial}{\partial x^2}f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)(2\pi i\mu)^2 e^{2\pi i(\mu x+\nu y)}\,d\mu\,d\nu = \mathcal{F}^{-1}\left[(2\pi i\mu)^2 F(\mu,\nu)\right]; \quad \frac{\partial^2}{\partial y^2}f(x,y) = \mathcal{F}^{-1}\left[(2\pi i\nu)^2 F(\mu,\nu)\right]$$

$$\Rightarrow \mathcal{F}[\Delta f] = \mathcal{F}\left[\frac{\partial^2}{\partial x^2}f(x,y)+\frac{\partial^2}{\partial y^2}f(x,y)\right] = \underbrace{-4\pi^2(\mu^2+\nu^2)}_{H(\mu,\nu)}F(\mu,\nu)$$

For image sharpening, use the composite Laplacian filter:

(replace with $\mu-M/2, \nu-N/2$ if F is shifted)

$$g(x,y) = f(x,y) - \Delta f(x,y) \iff \mathcal{F}[f-\Delta f] = (1+4\pi^2(\mu^2+\nu^2))F(\mu,\nu)$$

(*) Remark: Can generalize the derivative property:

$$\mathcal{F}\left[\frac{\partial^n}{\partial x^n}f(x)\right] = (2\pi i\mu)^n F(\mu)$$

(includes non-integer n (e.g. $n=1/2$))

Unsharp Masking

Previously, saw unsharp masking in the spatial domain $[g_{mask} = f - f_{smoothed}]$

→ Now, can look at the corresponding frequency-domain transformation:

$$g_{mask} = f - f_{LP} \iff F_{hp} = F - F\cdot H_{lp} \longrightarrow g = f + k\cdot g_{mask} \iff \left(1+k[1-H_{lp}]\right)F(\mu,\nu)$$

Define $H_{hp} = 1 - H_{lp}$ → $H = 1 + kH_{hp}$ called a high-frequency emphasis filter function; can write more generally as:

$$H(\mu,\nu) = a + b\cdot H_{hp}(\mu,\nu) \qquad [a\geq 0,\ b\geq a]$$

Advantage: Unlike regular highpass filters, $a>0$ ensures mean [DC component] $\neq 0$ (Otherwise: mean $0 \Rightarrow$ have negative pixels [undesirable])

Sharpening Highpass Filters (cont.)

Homomorphic Filtering

Recall: Previously, expressed image intensity via the illumination-reflectance model: $f(x,y) = i(x,y) \cdot r(x,y)$

In our case: want to be able to operate on each component separately in the frequency domain [but FT works better with sums than products]
→ can take the logarithm to convert the product into a sum, then take the Fourier transform:

$$z(x,y) := \ln[f(x,y)] = \ln[i(x,y)] + \ln[r(x,y)]$$

$$\longrightarrow \quad \mathcal{F}[z(x,y)] = \mathcal{F}[\ln\{i(x,y)\}] + \mathcal{F}[\ln\{r(x,y)\}] \iff Z(\mu,\nu) = F_i(\mu,\nu) + F_r(\mu,\nu)$$

To apply a filter $H(\mu,\nu)$:

$$S(\mu,\nu) := H(\mu,\nu) \cdot Z(\mu,\nu) = F_i(\mu,\nu)H(\mu,\nu) + F_r(\mu,\nu)H(\mu,\nu)$$

$$\rightarrow s(x,y) = \mathcal{F}^{-1}[S] = \mathcal{F}^{-1}[F_i \cdot H] + \mathcal{F}^{-1}[F_r \cdot H] = i'(x,y) + r'(x,y) = \ln(g(x,y))$$

$$\rightarrow g(x,y) = e^{s(x,y)} = e^{i'(x,y)} e^{r'(x,y)}$$

(*) Note: Homomorphic filtering only applies in scenarios where we can express f as $f = r \cdot i$ (Ex: Medical imaging)

Selective Filtering (§3.7)

In some cases, may want to selectively process certain regions in frequency spectrum ← can mitigate periodic noise
→ use bandpass, bandreject, & notch filters

Bandpass and Bandreject Filters

Bandpass and bandreject filters: permit only/reject only signals in a given frequency band, respectively

$$H_{BP}(u,v) = 1 - H_{BR}(u,v)$$

← similar to lowpass vs highpass [lourejct] filters

More formally: a bandreject filter must satisfy: (i) $H_{BR} \in [0, 1]$
(ii) $H_{BR}(u,v) = 0$ if $D(u,v) = C_0$ [parameter]
(*) Typically, 2 parameters: C_0 [band location], W [bandwidth]

Basic Bandreject Filters

Have 3 types (ideal, Gaussian, & Butterworth), similar to before:

1. Ideal Bandreject Filter

$$H_{BR}(u,v) = \begin{cases} 0 & \text{if } C_0 - \frac{W}{2} \le D(u,v) \le C_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$$

2. Gaussian Bandreject Filter

$$H_{BR}(u,v) = 1 - e^{-\left[\frac{D^2(u,v) - C_0^2}{D(u,v)W}\right]^2}$$

3. Butterworth Bandreject Filter

$$H_{BR}(u,v) = \frac{1}{1 + \left(\frac{D(u,v) \cdot W}{D^2(u,v) - C_0^2}\right)^{2}}$$

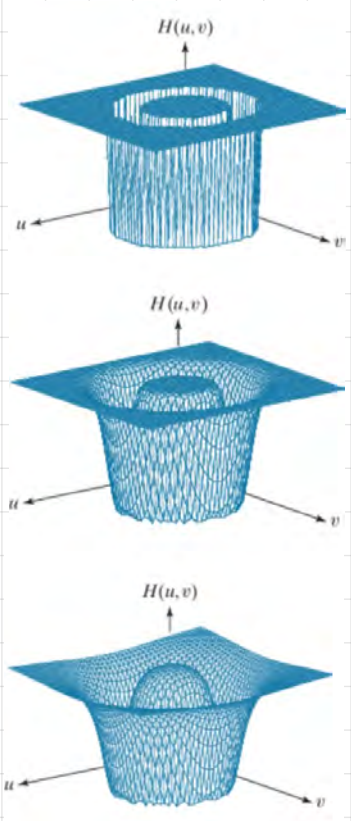


Figure 37: Ideal, Gaussian, and Butterworth band reject filter.

Selective Filtering (cont.)

Bandpass & Bandreject Filters (cont.)

Bandpass, bandreject: operate on a specific frequency band [around c_0]

⇓

Good for dealing with periodic structures/noise in images (e.g. grid lines)

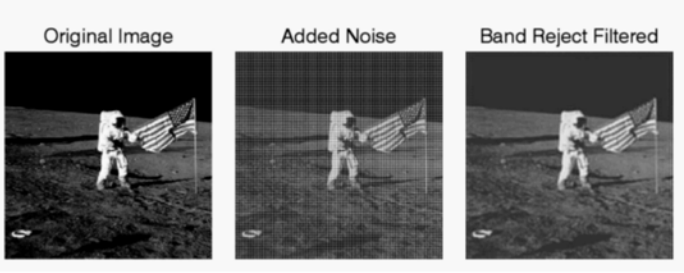


Figure 38: Application of Butterworth bandreject filter to remove periodic noise.

Notch Filters

Notch filters: a generalization of BP/BR filters; selectively pass/reject frequencies within a predefined region (of the frequency spectrum) that is symmetric about the center

Notch reject filters (general form): given as the product of highpass filter transfer functions centered around notch centers:

$$H_{NR} = \prod_{k=1}^{Q} H_{NR}^{(k)}(\mu,\nu) \cdot H_{NR}^{(-k)}(\mu,\nu)$$

where $H_{NR}^{(k)}, H_{NR}^{(-k)}$ are highpass filters centered at (μ_k, ν_k) and $(-\mu_k, -\nu_k)$, resp., for each notch pair $k = 1, \dots, Q$ (notch centers)

note: product of notch filters is a notch filter!

Equivalently: $H_{NR}^{(k)}, H_{NR}^{(-k)}$ are highpass filters with their distance functions replaced by $D_k(\mu,\nu)$ and $D_{-k}(\mu,\nu)$, resp.:

$$D_k(\mu,\nu) = \sqrt{(\mu - M/2 - \mu_k)^2 + (\nu - N/2 - \nu_k)^2}$$

$$D_{-k}(\mu,\nu) = \sqrt{(\mu - M/2 + \mu_k)^2 + (\nu - N/2 + \nu_k)^2}$$

Note: Have the following relationship between notch reject & notch pass filters:

$$H_{NR} = 1 - H_{NP}$$

similar to BP & BR filters

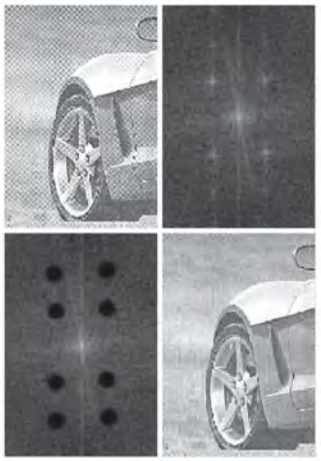


Figure 40: Original image with interference patterns (Moire pattern), its spectrum, spectrum after applying a Butterworth notch reject filter, new image.

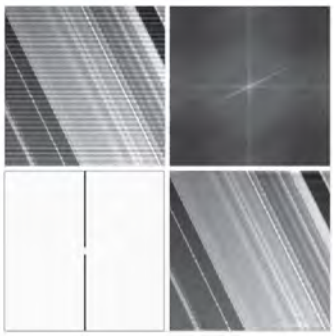


Figure 41: Image of Saturn rings with interference patterns, its spectrum, notch reject filter, new image.

Notch filters: are useful for filtering [spatially-dependent] periodic noise patterns, e.g. sine/cosine noise

see more next chapter

(*) Ex: Butterworth NR with N notch pairs:

$$H_{NR}(\mu,\nu) = \prod_{k=1}^{N} \left[1 + \left(\frac{D_{0k}}{D_k(\mu,\nu)}\right)^n\right]^{-1} \left[1 + \left(\frac{D_{0k}}{D_{-k}(\mu,\nu)}\right)^n\right]^{-1}$$

Selective Filtering (cont.)

(*) Why do notch filters have to be symmetric?:

Recall: The DFT of a real image is symmetric (← property of the DFT)

↳ In our case: want our output image $g = \text{IDFT}(H \cdot F)$ to be (approximately) a real image (↑ F is symmetric!)
- → Want H to be symmetric
- → For every $H_{NR}^{(k)}$ used, also multiply by $H_{NR}^{(-k)}$

(*) The Fast Fourier Transform

The **fast Fourier transform (FFT)**: An efficient (class of) algorithm for finding the DFTs/IDFTs of sequences exactly

- For an image with M-many pixels: reduces complexity from $O(M^2)$ [regular DFT formula] to $O(M \log M)$ [FFT]

The Cooley-Tukey FFT Algorithm

Idea: Recursively expresses DFTs of size N (N a composite #) into N_1-many smaller DFTs of size N_2, where $N = N_1 \cdot N_2$

Derivation ("Radix-2 DIT": $N \to N/2 + N/2$)

First, can separate DFT into 2 sums over even- and odd-numbered terms, respectively:

$$X(\mu) := \sum_{n=0}^{N-1} e^{-\frac{2\pi i}{N} n\mu} = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)\mu} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)\mu}$$

for $\mu \in \{0, 1, \dots, N-1\}$

$$= \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)\mu}}_{E(\mu)} + e^{-\frac{2\pi i}{N}\mu} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m)k}}_{O(\mu)} = E(\mu) + e^{-\frac{2\pi i}{N}\mu} O(\mu)$$

notice: this expression is a size-2 DFT of $E(\mu), O(\mu)$ $\Rightarrow N_1 = 2,\ N_2 = N/2$ (can find similar algs. for other choices of N_1, N_2)

→ $E(\mu), O(\mu)$ are 2 DFTs of size $N/2$ [for $\mu = 0, 1, \dots, N/2 - 1$] ← equality still holds even if $\mu > N/2 - 1$, but we want E, O to be "true" DFTs so we can apply recursion

Issue: $X(\mu)$ [size-N DFT] is defined for $\mu = 0, 1, \dots, N-1$, but $E(\mu), O(\mu)$ [size $N/2$-DFTs] are only defined for $\mu = 0, 1, \dots, N/2 - 1$ → what to do for $\mu = N/2, N/2 + 1, \dots, N-1$?

Solution: We can use the periodicity of the complex exponential (rec.: $e^{ix} = -e^{i(x+\pi)}$) to find $X(\mu + N/2)$ from $E(\mu), O(\mu)$ [$\mu = 0, \dots, N/2$]:

$$X(\mu + N/2) = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} m(\mu + N/2)} + e^{-\frac{2\pi i}{N}(\mu + N/2)} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} m(\mu + N/2)}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} m\mu} - e^{-\frac{2\pi i}{N}\mu} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} m\mu} = E(\mu) - e^{-\frac{2\pi i}{N}\mu} O(\mu) \longrightarrow$$

(Radix-2 DIT FFT)

$$X(\mu) = \begin{cases} E(\mu) + e^{-\frac{2\pi i}{N}\mu} O(\mu) & \mu \le \frac{N}{2} - 1 \\ E(\mu) - e^{-\frac{2\pi i}{N}\mu} O(\mu) & \mu > \frac{N}{2} - 1 \end{cases}$$

Notice: (i) Can compute $X(\mu)$ for each $\mu = 0, 1, \dots, N-1$ by only finding $E(\tilde{\mu}), O(\tilde{\mu})$ for $\tilde{\mu} = 0, 1, \dots, N/2 - 1$ [$N \cdot \text{DFT}(N) \mapsto 2 \cdot (N/2) \cdot \text{DFT}(N/2) + N \cdot \text{DFT}(2)$]

+ (ii) Can apply reduction process recursively to $E(\mu), O(\mu)$ ⟶ $O(N \log N)$

Image Restoration (Lec. 17-24)

Previously: saw image enhancement [improving image quality, based on some criteria]

→ Now: image restoration [reversing known degradations on an image]

Modeling Image Degradation (§4.1)

Commonly, represent a degradition $g(x,y)$ of an image $f(x,y)$ by a linear model:

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$

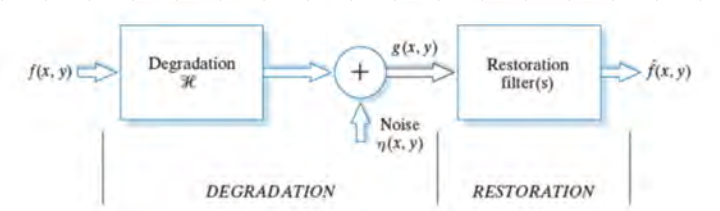
1. $h(x,y)$: degradation function [here: a kernel]
2. $\eta(x,y)$: additive noise

Goal: Given the following information: (i) The degraded image $g(x,y)$

(ii) The degradation function $h(x,y)$ / degradation operator $\mathcal{H}$

(iii) Some statistical information/properties about the noise $\eta(x,y)$

→ Want to find a restored image $\hat{f}(x,y) \approx f(x,y)$:



In the frequency domain:

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y) \quad <=> \quad G(\mu,\nu) = H(\mu,\nu)F(\mu,\nu) + N(\mu,\nu)$$

(*) Remark: It is generally not feasible to solve for F directly in the frequency domain:

(i) The noise component $N(\mu,\nu)$ is unknown

(ii) $H(\mu,\nu)$ may have very small (or zero) values → may not be able to divide

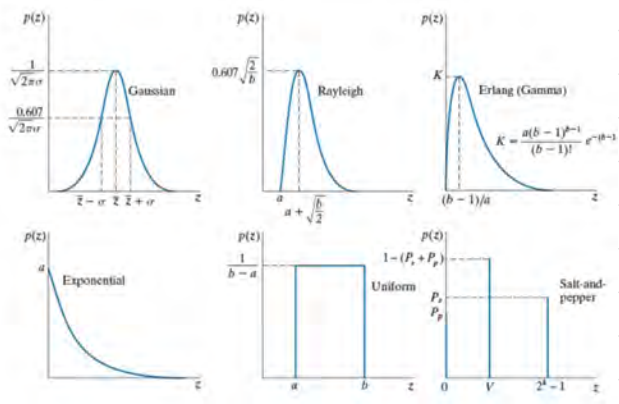
Noise Models (§4.2)

Spatially-Invariant Noise

Spatially-invariant noise models make the assumptions that:

(i) The noise is independent of spatial coordinates (x, y)

(ii) The noise is uncorrelated with the image $f(x, y)$



Some notable probability density functions:

1. **Gaussian Noise:**

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z-\bar{z})^2}{2\sigma^2}}$$

based on: (i) z [noise intensity] (ii) $\bar{z}$ [mean] (iii) σ [standard deviation]

- Can occur due to electronic circuit noise or sensor noise (due to poor illumination and/or high temperature, e.g.)

2. **Rayleigh Noise:**

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{\frac{-(z-a)^2}{b}} & \text{if } z \geq a \\ 0 & \text{if } z < a \end{cases}$$

- Can be used to model shape of skewed histograms or characterize noise in range imaging

3. **Erlang (Gamma) Noise** + 4. **Exponential Noise:**

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \qquad p(z) = \begin{cases} ae^{-az} & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- Exponential, gamma densities used in laser imaging
- Exponential noise as a special case of Erlang noise for $b = 1$; has mean $1/a$, variance $1/a^2$

5. **Uniform Noise:**

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

- Has mean $\frac{a+b}{2}$, variance $(b-a)^2/12$
- Often used as the basis for random number generators, e.g. for simulations

6. **Impulse Noise ("Salt-and-pepper"):**

$2^k - 1$ represents max intensity level

$$p(z) = \begin{cases} P_a & \text{if } z = 2^k - 1 \text{ [salt]} \\ P_b & \text{if } z = 0 \text{ [pepper]} \\ 1-(P_a+P_b) & \text{if } z = V \end{cases}$$

- Is a form of multiplicative noise [$g = f \cdot \eta$], unlike the previous types
- Can occur when quick transients (e.g. faulty switching) happen during imaging

(*) Special cases: (i) One of $P_a, P_b = 0 \rightarrow$ unipolar impulse noise, (ii) $P_a \approx P_b \rightarrow$ bipolar impulse noise

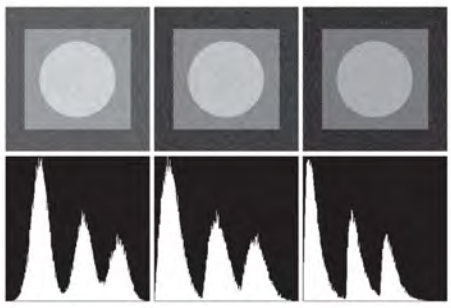


Figure 42: Images corrupted by Gaussian, Rayleigh, and Erlang noise with their histograms.

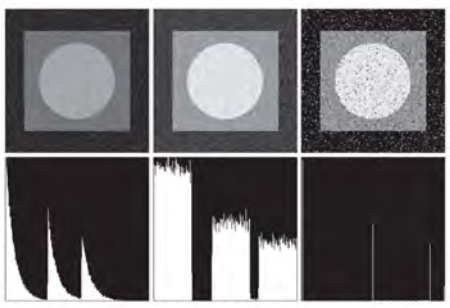


Figure 43: Images corrupted by Exponential, Uniform, and Impulse noise with their histograms.

Noise Models (cont.)

Spatially-Dependent Noise

Periodic noise: A common type of spatially-dependent noise

- Can occur due to electrical/electromechanical interference during image acquisition, e.g.

(*) Ex: (i) $\eta(x,y) = \sin(2\pi\mu_0 x + 2\pi\nu_0 y)$ [sine noise]
(ii) $\eta(x,y) = \cos(2\pi\mu_0 x + 2\pi\nu_0 y)$ [cosine noise]

Figure 44: Image corrupted by periodic sine noise and its spectrum.

Filtering Periodic Noise

Notice: $\mathcal{F}\left[\sin(2\pi\mu_0 x + 2\pi\nu_0 y)\right] = \frac{jMN}{2}\left[\delta(\mu+\mu_0, \nu+\nu_0) - \delta(\mu-\mu_0, \nu-\nu_0)\right]$ (& a similar expression for cosine noise)

↳ Impulses at $(-\mu_0, -\nu_0), (\mu_0, \nu_0)$ $\xrightarrow{\text{shift } (M/2, N/2)}$ $(M/2 - \mu_0, N/2 - \nu_0), (M/2 + \mu_0, N/2 + \nu_0)$

↳ symmetric about $(M/2, N/2)$ ⇒ can use notch filter to remove

Estimation of Noise Parameters

Noise Parameter Estimation (Steps)

1. Find a subregion S of the image with relatively constant background intensity
2. Estimate the noise type:
   (i) Generate the histogram of S
   (ii) Compare S's histogram with the probability density functions (PDFs) of known noise types, and choose the best match
3. Estimate the mean and variance: Compute:

$$\bar{z} = \sum_{i=0}^{L-1} z_i p_S(z_i) \quad \text{[mean]}, \qquad \sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 p_S(z_i) \quad \text{[variance]}$$

↳ Can use $\bar{z}, \sigma^2$ to estimate the noise parameters a, b

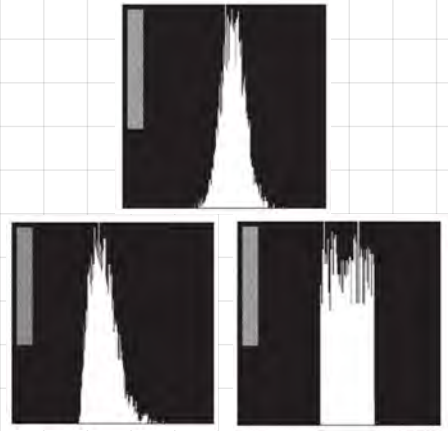


Figure 45: Histogram of subimages for Gaussian, Rayleigh, and uniform noise.

(*) For impulse noise, need a slightly different approach - look at distribution of black & white pixels:

1. Select a near-constant gray region S (i.e. neither black nor white)
2. Compute the normalized histogram and take the peaks corresponding to black and white pixels as P_b and P_a, respectively

Image Restoration with Spatial Filters (§4.3)

Goal: Assuming noise is additive (i.e. $g = f + \eta$), we want to recover f at a point (x,y) based on an $m \times n$ neighborhood S_{xy} of the point $g(x,y)$ using spatial filters.

Mean Filters

1. **Arithmetic Mean Filter** – Smooths local variances in an image: reduces noise, but tends to cause blurring

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

2. **Geometric Mean Filter**: Comparable to the arithmetic mean filter, but tends to preserve more image detail

$$\hat{f}(x,y) = \left[\prod_{(s,t) \in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

3. **Harmonic Mean Filter**: Works well for Gaussian noise and salt noise, but fails for pepper noise

($\frac{1}{g(s,t)}$ small for salt noise; $\frac{1}{g(s,t)}$ large for pepper noise)

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

4. **Contraharmonic Mean Filter**: A generalization of the harmonic filter

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

Q an order parameter:
- (i) $Q = 0$ → arithmetic mean filter
- (ii) $Q = -1$ → harmonic mean filter
- In general: $Q > 0$ reduces pepper noise; $Q < 0$ for salt noise

In summary: (i) Can use (1), (2) for random noise (e.g. Gaussian, uniform)
(ii) Can use (3) for salt, Gaussian noise
(iii) (4) works for impulse noise [provided Q has the appropriate sign]

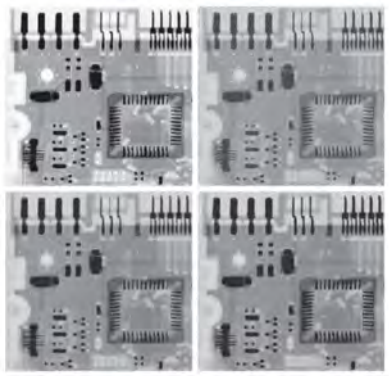


Figure 46: First row: Original image, image corrupted by Gaussian noise. Second row: Using arithmetic mean filter and geometric mean filter.

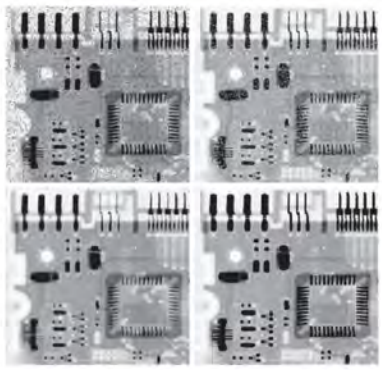


Figure 47: First row: Image corrupted by pepper noise, image corrupted by salt noise. Second row: Using contraharmonic mean filter with $Q = 1.5$ and -1.5.

Image Restoration with Spatial Filters (cont.)

Order-Statistic Filters

Order-statistic filters - filters that utilize the ordering of points (i.e. pixel values)

1. **Median Filter**: Effective for both unipolar and bipolar impulse noise; results in less blurring than a linear smoothing filter

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}$$

2. **(i) Min & (ii) Max Filter**: Work for salt noise & pepper noise, respectively

(i) $$\hat{f}(x,y) = \min_{(s,t)}\{g(s,t)\}$$

(ii) $$\hat{f}(x,y) = \max_{(s,t)}\{g(s,t)\}$$

3. **Midpoint Filter**: Combines order-statistics and averaging; is effective for randomly-distributed noise (e.g. Gaussian, uniform)

$$\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)}\{g(s,t)\} + \min_{(s,t)}\{g(s,t)\}\right]$$

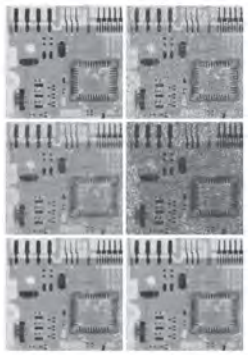


Figure 49: First row: Image corrupted by uniform noise, image additionally corrupted by additive salt and pepper noise. Second and third row: New images by arithmetic mean, geometric mean, median, and alpha-trimmed mean filter.

4. **Alpha-Trimmed Mean Filter**: Defined by:

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

where g_R is obtained by deleting the $d/2$ highest and $d/2$ lowest intensities in S_{xy} (for some even d [parameter])

(*) **Notice**: (i) $d=0 \rightarrow$ arithmetic mean filter (good for random noise, e.g. Gaussian)
(ii) $d = mn-1 \rightarrow$ median filter (good for impulse noise)
(iii) $0 < d < mn-1 \rightarrow$ a balance/compromise; can use for various noises, e.g. combination of Gaussian & salt-and-pepper

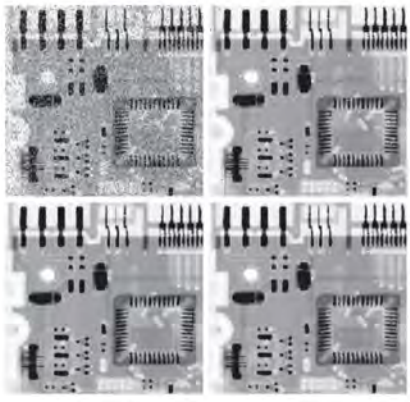


Figure 48: First row: Images corrupted by salt pepper noise, restoration by median filter. Second row: restoration by applying median filter twice and three times.

Noise Reduction with Frequency-Domain Filters (§4.4)

Periodic Noise Reduction

Can use notch reject filters to attenuate bright dots in the frequency domain (corresponding to noise spikes)

2 common filter types (for a fixed bandwidth D_{0k}):

1. Butterworth NR Filter

$$H_{NR}(\mu,\nu) = \prod_{k=1}^{Q}\left[\frac{1}{1+[D_{0k}/D_k(\mu,\nu)]^n}\right]\left[\frac{1}{1+[D_{0k}/D_{-k}(\mu,\nu)]^n}\right]$$

2. Gaussian NR Filter

$$H_{NR}(\mu,\nu) = \prod_{k=1}^{Q}\left[1-e^{\frac{-D_k(\mu,\nu)^2}{2D_{0k}^2}}\right]\left[1-e^{\frac{-D_{-k}(\mu,\nu)^2}{2D_{0k}^2}}\right]$$

where D_k, D_{-k} are defined by:

$$D_k(\mu,\nu) = \left[(\mu - M/2 + \mu_k)^2 + (\nu - N/2 + \nu_k)^2\right]^{1/2}$$

$$D_{-k}(\mu,\nu) = \left[(\mu - M/2 - \mu_k)^2 + (\nu - N/2 - \nu_k)^2\right]^{1/2}$$

and Q is the # of spikes to attenuate, where the k^{th} spike is centered at (μ_k, ν_k).

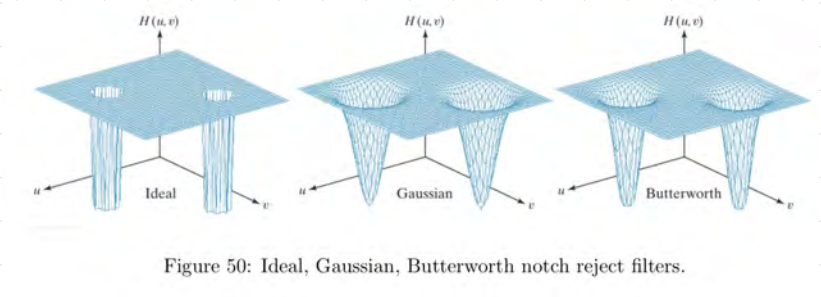


Figure 50: Ideal, Gaussian, Butterworth notch reject filters.

Figure 51: Using a notch filter to remove periodic noise, the second and third plots are the spectrum of the original image and notch filter with a width of 2 pixels.

Periodic Noise Extraction

Recall: Given a notch reject filter H_{NR}, can obtain a notch pass filter H_{NP} by $H_{NP} = 1 - H_{NR}$

Previously, saw that the FT of sine/cosine noise corresponds to two impulses, symmetric about $(M/2, N/2)$
→ more generally - can write:

$$g(x,y) = f(x,y) + \eta(x,y) \iff G(\mu,\nu) = F(\mu,\nu) + N(\mu,\nu)$$

where $N(\mu,\nu)$ [F.T. of η] consists of bright dots [Notch pairs]; then, find that:

$$\eta(x,y) = \mathcal{F}^{-1}[G-F] = \mathcal{F}^{-1}[G - G\cdot H_{NR}] = \mathcal{F}^{-1}[H_{NP}\cdot G]$$

(can compute this)

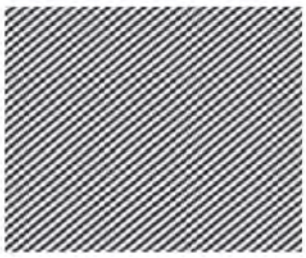


Figure 52: Extracted noise pattern by a notch pass filter in the previous example.

(*) The value of the noise $\eta(x,y)$ can be useful if the noise pattern is the same for other images (generated by the same imaging process, e.g.)

Estimating the Degradation Function (§4.5)

Returning to the general case: given a noised image $g = h * f + \eta$, want to estimate h.

(1) Estimation by Observation

Given $g(x,y)$, can first try to find a rectangular subimage G_s that mitigates the effect of the additive noise $[\eta]$. In particular, we can look for high-contrast G_s:

$$\rightarrow g(x,y) = h * f + \eta \approx g_s(x,y) = h_s(x,y) * f_s(x,y) \qquad [\text{hope/assumption}]$$

We can apply a sharpening filter to g_s to obtain an unblurred subimage $\hat{f}_s$ (recall: the true subimage f_s is unknown). We can use $\hat{f}_s$ and g_s (namely, their Fourier transforms) to estimate $H_s(\mu,\nu)$:

$$H_s(\mu,\nu) \approx \frac{G_s(\mu,\nu)}{F_s(\mu,\nu)} \approx \frac{G_s(\mu,\nu)}{\hat{F}(\mu,\nu)} \qquad (G_s = H_s \cdot F_s)$$

Using $H_s(\mu,\nu)$, we can then estimate $H(\mu,\nu)$ [larger scale, but keeping the same basic shape as $H_s(\mu,\nu)$].

(*) Note: Obtaining an accurate estimate of F_s is, in practice, potentially challenging; as such, this approach is mainly useful in specific scenarios (e.g. restoring historical photographs)

(2) Estimation by Experimentation

Given the equipment/source/etc. used to acquire the degraded image, can use it to find the degradation function:

1. Using the aforementioned source, use it to obtain the degradation $g(x,y)$ of a small dot of light, i.e. resembling (for some amplitude A):

$$f(x,y) = A\delta(x,y) \iff F(\mu,\nu) = A$$

   - Want as bright a dot as possible [maximum A] to minimize the influence of the noise $\eta(x,y)$

2. Using $F(\mu,\nu)$ and $G(\mu,\nu)$, can compute the degradation function $H(\mu,\nu)$:

$$G(\mu,\nu) \approx H(\mu,\nu) \cdot F(\mu,\nu) = H(\mu,\nu) \cdot A \Rightarrow H(\mu,\nu) = \frac{G(\mu,\nu)}{A}$$

(3) Estimation by Modeling

If the degradation is due to environmental conditions, a model can be developed based on physical parameters.

(*) Ex. Can model atmospheric turbulence by:

$$H(\mu,\nu) = e^{-K(\mu^2+\nu^2)^{5/6}}$$

where K is a turbulence parameter: $K = \begin{cases} 0.0025 & \text{(severe turbulence)} \\ 0.001 & \text{(mild turbulence)} \\ 0.00025 & \text{(low turbulence)} \end{cases}$

Figure 53: Degradation function for $\kappa = 0.0025$, 0.001, and 0.00025.

Estimating the Degradation Function (cont.)

(3) Estimation by Modeling (cont.)

Additionally, can develop models based on physical principles.

(*) Ex: For an image blurred due to linear motion during acquisition, can express it by:

exposure time → T; motion components → $x_0(t)$, $y_0(t)$

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

→ Can use the Fourier transform to derive $H(\mu,\nu)$:

$$\begin{aligned} G(\mu,\nu) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\, e^{-2\pi i[\mu x+\nu y]}\, dx\, dy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left[\int_0^T f(x-x_0(t),\ y-y_0(t))\, dt\right] e^{-2\pi i[\mu x+\nu y]}\, dx\, dy \\ &= \int_0^T \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-x_0(t),\ y-y_0(t))\, e^{-2\pi i[\mu x+\nu y]}\, dx\, dy\right] dt \\ &= \int_0^T \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')\, e^{-2\pi i[\mu x'+\nu y']}\, dx'\, dy'\right] e^{2\pi i[\mu x_0(t)+\nu y_0(t)]}\, dt = \underbrace{F(\mu,\nu)\int_0^T e^{2\pi i[\mu x_0(t)+\nu y_0(t)]}\, dt}_{H(\mu,\nu)} \end{aligned}$$

(*) Ex: Uniform linear motion $\left[x_0(t) = \frac{at}{T},\ y_0(t) = \frac{bt}{T}\right]$

$$\rightarrow H(\mu,\nu) = \frac{T}{\pi(\mu a+\nu b)}\sin\left(\pi(\mu a+\nu b)\right) e^{-i\pi(\mu a+\nu b)}$$

Figure 54: Image blurred with $a = b = 0.1$ and $T = 1$.

Inverse Filtering (§4.6)

Goal: Given the degraded image $g(x,y)$ [+ F.T. $G(\mu,\nu)$] and estimated degradation function $H(\mu,\nu)$, want to find $\hat{f}(x,y) \approx f(x,y)$.

If the noise term $\eta(x,y)$ is negligible, can approximate:

$$g(x,y) = (h * f) + \eta \approx (h * f)(x,y) \iff G(\mu,\nu) = H(\mu,\nu)F(\mu,\nu)$$

$$\downarrow$$

$$\hat{f}(x,y) = \mathcal{F}^{-1}\left[\hat{F}(\mu,\nu)\right] = \mathcal{F}^{-1}\left[\frac{G(\mu,\nu)}{H(\mu,\nu)}\right]$$

However, this approximation suffers when the noise $\eta(x,y)$ is significant:

$$G(\mu,\nu) = H(\mu,\nu)F(\mu,\nu) + \underbrace{N(\mu,\nu)}_{\text{noise}} \longrightarrow \hat{F}(\mu,\nu) = \frac{G}{H} = F + \frac{N}{H}$$

($H(\mu,\nu)$ can be small for regions outside of passband $\Rightarrow$ N/H becomes excessively large, throws off $\hat{F}(\mu,\nu)$)

One solution: can impose a cutoff threshold on G/H, set $\hat{F}(\mu,\nu) = 0$ for $D(\mu,\nu) > D_0$ [(μ,ν) outside of passband].

Figure 55: Results of exact inverse filtering with cut-off values at 40, 70, and 85.

An alternative solution: can apply a low-pass filter to suppress high-frequency noise:

$$\longrightarrow \hat{F} = \frac{G}{H} \cdot H_{LPF}$$

Wiener Filtering (§4.7)

Goal: Want to find $\hat{f}$ minimizing the mean square error ($e^2 = \mathbb{E}\{(f-\hat{f})^2\}$) given by:

$$\text{MSE} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)-\hat{f}(x,y)\right]^2$$

← smaller MSE ⇕ better restoration

Preliminaries: Define $|H(\mu,\nu)|^2 := H(\mu,\nu)H(\mu,\nu)^*$ → power spectrums of noise and of f: $S_\eta = |N(\mu,\nu)|^2$, $S_f = |F(\mu,\nu)|^2$

Additionally, assume the following:
(i) $\hat{F} = WG$ [linear w.r.t. G], where $W(\mu,\nu)$ is some linear operator (unknown, want to construct an "optimal" W)
(ii) Either F or N has mean 0
(iii) We know (or have an estimate for) the degradation function $H(\mu,\nu)$
(iv) The noise η and image f are uncorrelated, i.e.: $\sum_\mu\sum_\nu F(\mu,\nu)N(\mu,\nu) = \left[\sum_\mu\sum_\nu F(\mu,\nu)\right]\left[\sum_\mu\sum_\nu N(\mu,\nu)\right] = 0$

+ **Recall (Plancherel's Theorem):**

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}^d}|f|^2\,dx\,dy = \int_{\mathbb{R}^d}\int_{\mathbb{R}^d}|F|^2\,d\mu\,d\nu \quad \text{(Continuous case)}, \qquad \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}|f|^2 = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}|F|^2 \quad \text{(Discrete case)}$$

Derivation for real H

Writing $W(\mu,\nu) = R_W(\mu,\nu) + iI_W(\mu,\nu)$, obtain:

(i) $|W(\mu,\nu)|^2 = R_W(\mu,\nu)^2 + I(\mu,\nu)^2$ (ii) $|1-W(\mu,\nu)H(\mu,\nu)|^2 = \left(1-R_W(\mu,\nu)H(\mu,\nu)\right)^2 + \left(I_W(\mu,\nu)H(\mu,\nu)\right)^2$

→ Can rewrite the MSE: $\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)-\hat{f}(x,y)\right]^2 = \frac{1}{MN}\sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1}S_f\left[(1-R_WH)^2+(I_WH)^2\right] + S_\eta\left[R_W^2+I_W^2\right]$

Want to find R_W, I_W minimizing MSE → set partial derivatives w.r.t. R_W, I_W to 0:

(i) $0 = \frac{\partial \text{MSE}}{\partial R_W} = \frac{2}{MN}\left[S_f(-H+R_W|H|^2)+S_\eta R_W\right] \longrightarrow R_W(\mu,\nu) = \frac{S_f H(\mu,\nu)}{S_f|H(\mu,\nu)|^2+S_\eta}$

(ii) $0 = \frac{\partial \text{MSE}}{\partial I_W} = \frac{2}{MN}\left[S_f I_W|H|^2 + S_\eta I_W\right] \longrightarrow I_W(\mu,\nu) = 0$

From this, obtain the Wiener filter:

$$W(\mu,\nu) = \frac{S_f H(\mu,\nu)}{S_f|H(\mu,\nu)|^2+S_\eta} = \frac{1}{H(\mu,\nu)}\cdot\frac{|H(\mu,\nu)|^2}{|H(\mu,\nu)|^2+S_\eta/S_f}$$

(optimal solution)

in practice: S_η, S_f unknown → approximate by a constant k

$$W(\mu,\nu) = \frac{1}{H(\mu,\nu)}\cdot\frac{|H(\mu,\nu)|^2}{|H(\mu,\nu)|^2+k}$$

(in practice)

Wiener Filtering (cont.)

Notice: Looking at the Wiener filter multiplicand: $\frac{|H(\mu,\nu)|^2}{|H(\mu,\nu)|^2 + k}$

- $k \neq 0 \Rightarrow$ LPF
- $k = 0 \Rightarrow \hat{F} = G/H$, direct inverse filter

Remark: Besides MSE, can find other metrics for restoration quality; ex: signal-to-noise ratio [SNR]:

(i) Spatial SNR:

$$\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \left[f(x,y) - \hat{f}(x,y)\right]^2}$$

numerator: signal; denominator: $\approx$ noise

higher SNR $\Leftrightarrow$ better restoration

(ii) Frequency-Domain SNR:

$$\frac{\sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1} |\hat{F}(\mu,\nu)|^2}{\sum_{\mu=0}^{M-1}\sum_{\nu=0}^{N-1} |\hat{N}(\mu,\nu)|^2}$$

(*) In practice, signal may have a wide range $\rightarrow$ express using log scale

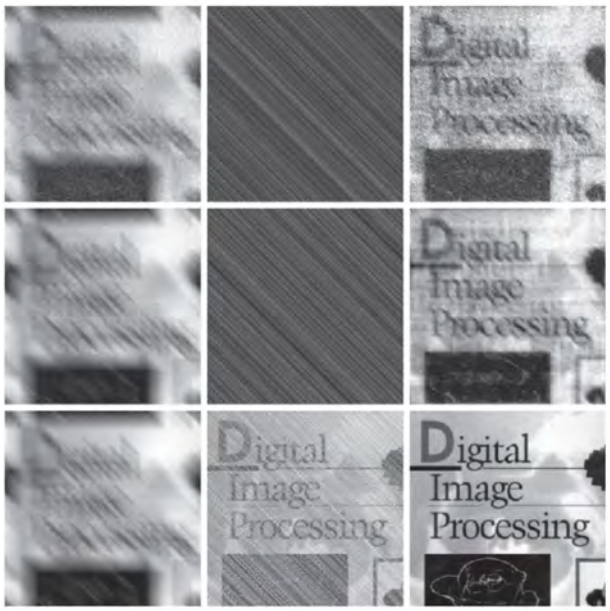


Figure 56: From left to right: blurred image, inverse filtering, Wiener filtering. From top to bottom: decreasing noise level.

Constrained Least-Squares Filtering (§4.8)

Previously: Saw the Wiener filter: used a constant k to approximate S_η / S_f

→ Now: Can use information about the mean and variance of the noise

Recall: $g = h * f + \eta \Rightarrow \eta = g - h * f$ → for accurate recovery, want $\hat{f}$ s.t. $\|g - h * \hat{f}\|_{L^2}^2 = \|\eta\|_{L^2}^2$

In this case, we are working with pixels => there may exist multiple solutions for $\hat{f}$. Can specify a "best" solution: for good reconstruction, typically want a smooth $\hat{f}$ → pick the $\hat{f}$ that minimizes the 2-norm of the Laplacian ("is most smooth"):

$$\hat{f} = \min_f \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(\Delta \hat{f}\right)^2 = \|\Delta \hat{f}\|_{L^2}^2$$
$$\text{subject to } \|g - h * f\|_{L^2}^2 = \|\eta\|_{L^2}^2$$

In practice, ☐ may be difficult to solve directly → can formulate a similar unconstrained problem using a Lagrange multiplier γ:

$$\hat{f} = \min_f \gamma \|\Delta \hat{f}\|_{L^2}^2 + \|g - h * f\|_{L^2}^2$$

← γ a regularization parameter: affects balance of smoothness vs. fidelity to the data

Can find that the solution is given by:

$$\hat{F}(\mu, \nu) = \left[\frac{1}{H(\mu, \nu)} \cdot \frac{|H(\mu, \nu)|^2}{|H(\mu, \nu)|^2 + \gamma |P(\mu, \nu)|^2}\right] G(\mu, \nu)$$

where $P(\mu, \nu) = -4\pi^2(\mu^2 + \nu^2)$ is the Laplacian filter in the frequency domain. (Proof on next page)

Algorithm (Constrained Least-Squares Filtering)

1. Set an initial value for γ + closeness threshold a, then compute $\hat{f}$
2. Stop if $\|\eta\|^2 - a \leq \|g - h * f\|^2 \leq \|\eta\|^2 + a$;

   otherwise: (i) If $\|g - h * f\|^2 < \|\eta\|^2 - a$, increase γ, recompute $\hat{f}$ & repeat

   (ii) If $\|g - h * f\|^2 > \|\eta\|^2 + a$, decrease γ, recompute $\hat{f}$ & repeat

Constrained Least-Squares Filtering (cont.)

Derivation:

Using Plancherel's theorem, obtain our energy function:

$$E(\hat{f}) := \gamma^2 \|\Delta \hat{f}\|_{L^2}^2 + \|g - h * f\|_{L^2}^2 = \gamma \int_{\mathbb{R}^d} |\mathcal{F}(\Delta f)|^2 d\mu d\nu + \int_{\mathbb{R}^d} |G - H\hat{F}| d\mu d\nu$$

Recall: $|\mathcal{F}(\Delta f)| = -4\pi^2(\mu^2+\nu^2)F(\mu,\nu) = P(\mu,\nu)F(\mu,\nu)$

Can decompose $\hat{F} = R_F + iI_F$, $G = R_G + iI_G$:

$$\rightarrow G - H\hat{F} = (R_G - HR_F) + i(I_G - HI_F) \Rightarrow |G - H\hat{F}|^2 = (R_G - HR_F)^2 + i(I_G - HI_F)^2$$

$$\Rightarrow E(R_F, I_F) = \gamma \int_{\mathbb{R}^d} |P(\mu,\nu)|^2 (R_F^2 + I_F^2) d\mu d\nu + \int_{\mathbb{R}^d} \left[(R_G - HR_F)^2 + i(I_G - HI_F)^2\right] d\mu d\nu$$

To minimize $E(\hat{f})$, can minimize E w.r.t. R_F, I_F by looking at its partial derivatives:

(i) $$0 = \frac{\partial E}{\partial R_F} = \gamma |P(\mu,\nu)|^2 2R_F + 2(R_G - HR_F)(-H) \Rightarrow R_F = \frac{HR_G}{|H|^2 + \gamma|P|^2}$$

(ii) $$0 = \frac{\partial E}{\partial R_I} = \gamma |P(\mu,\nu)|^2 2I_F + 2(I_G - HI_F)(-H) \Rightarrow I_F = \frac{HI_G}{|H|^2 + \gamma|P|^2}$$

$\rightarrow$ This gives us our solution:

$$\hat{F}(\mu,\nu) = \frac{HG}{|H|^2 + \gamma|P|^2}$$

▨

Remark: To compute $\|\eta\|_{L^2}^2$, can use the mean $\bar{\eta}$, variance σ_η^2 of η:

$$\|\eta\|^2 = \sum_{x=1}^{M}\sum_{y=1}^{N} \eta(x,y)^2 = \sum_{x=1}^{M}\sum_{y=1}^{N} \left[(\eta(x,y) - \bar{\eta})^2 + 2\eta(x,y)\bar{\eta} - \bar{\eta}^2\right]$$

$$= MN\left[\frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} (\eta(x,y) - \bar{\eta})^2 + \bar{\eta}^2\right] = MN(\sigma_\eta^2 + \bar{\eta}^2)$$

Figure 57: Results of constrained least squares filtering on images with different noise levels.

Geometric Mean Filtering (§4.9)

The geometric mean filter (a generalization of the inverse/Wiener filters) is defined by (for real constants $\alpha, \beta \geq 0$):

$$\hat{F}(\mu,\nu) = \left[\frac{H^*(\mu,\nu)}{|H(\mu,\nu)|^2}\right]^{\alpha} \left[\frac{H^*(\mu,\nu)}{|H(\mu,\nu)|^2 + \beta\frac{S_\eta(\mu,\nu)}{S_f(\mu,\nu)}}\right]^{1-\alpha} G(\mu,\nu)$$

(*) Special cases:

(i) $\alpha = 1 \longrightarrow$ inverse filter $\left[\hat{F} = G/H\right]$

(ii) $\alpha = 0 \longrightarrow$ parametric Wiener filter:

$$\hat{F}(\mu,\nu) = \frac{1}{H} \cdot \frac{|H|^2}{|H|^2 + \beta\frac{S_\eta}{S_f}} G$$

← Wiener filter with an extra parameter β

(iii) $\alpha = 1/2 \longrightarrow$ geometric mean filter (is the geometric mean of 2 terms)

(*) ☐ is useful in practice: can vary α, β to attain better results empirically

Image Reconstruction from Projection (§4.10)

In various applications/settings (e.g. medical imaging), the acquired data represents a projection of the original image, satisfying:

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

$\Updownarrow$ 1D F.T. (over ρ)

$$G(\omega,\theta) = \int_{-\infty}^{\infty} g(\rho,\theta)\,e^{-i2\pi\omega\rho}\,d\rho$$

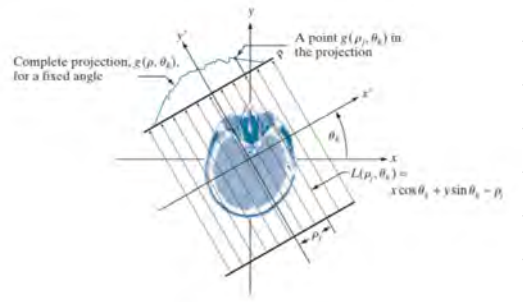


Figure 58: Geometry of parallel beam projection.

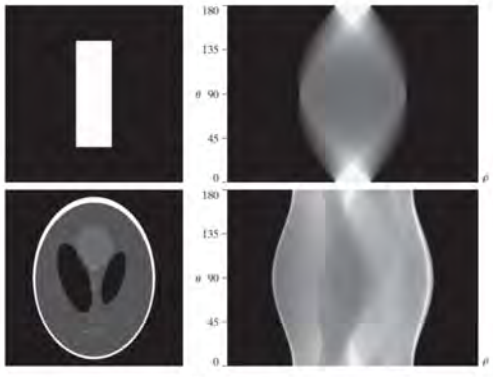


Figure 59: Original image and its corresponding projection data.

Image Reconstruction from Projection (cont.)

To reconstruct $f(x,y)$ from $g(\rho,\theta)$, can decompose $G(\omega,\theta)$:

$$G(\omega,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\delta(x\cos\theta + y\sin\theta - \rho)\, e^{-i2\pi\omega\rho}\, d\rho\right] dx\,dy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-i2\pi\omega(x\cos\theta + y\sin\theta)}\, dx\,dy = F(\omega\cos\theta,\ \omega\sin\theta)$$

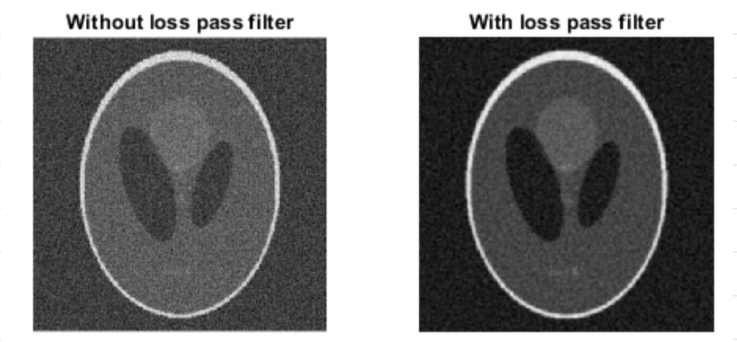
→ Can use the IFT to recover $f(x,y)$:

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\mu,\nu)\, e^{i2\pi(\mu x + \nu y)}\, d\mu\, d\nu$$

convert to polar coordinates

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\omega\cos\theta,\ \omega\sin\theta)\, e^{i2\pi\omega(x\cos\theta + y\sin\theta)}\, \omega\, d\omega\, d\theta = \int_{0}^{2\pi}\int_{0}^{\infty} G(\omega,\theta)\, e^{i2\pi\omega(x\cos\theta + y\sin\theta)}\, \omega\, d\omega\, d\theta$$ [Back-projection]

In practice: direct back-projection can amplify high-frequency noise → can use a lowpass filter to stabilize



Color Image Processing (Lec. 24-25)

Previously: Worked exclusively with grayscale images - each pixel is associated with a single value [intensity], visualize as a black-and-white image (0 = black, 1 [or 255] = white)

→ Now: Look at methods for representing & processing color images

The RGB Color Model (§5.1)

The RGB model: represents colors as combinations of 3 "primary colors": red, green, and blue

- Each color represented as a point in Cartesian space:

$$p = (r, g, b)$$

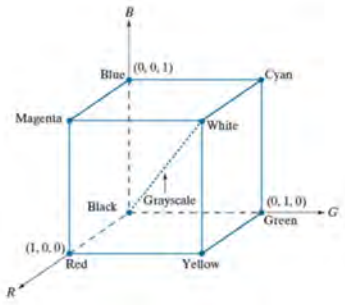


Figure 62: RGB color space representation.

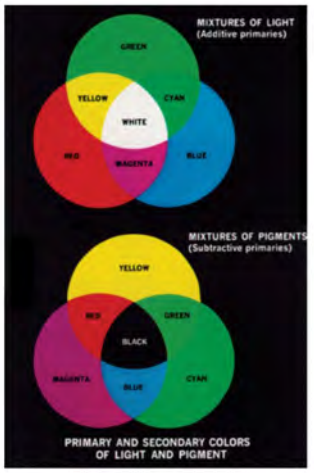


Figure 61: Color mixing in the RGB model.

- RGB: most popular color model; hardware-oriented

Properties of the RGB Model:

(i) RGB color images are composite images: consist of 3 component grayscale images (one per primary color)

(ii) Define the pixel depth of an image as the # bits used to represent each RGB pixel

(*) Ex: If each component has 8-bit intensity values:

$$p = \begin{bmatrix} r \in [0, 255] \\ g \in [0, 255] \\ b \in [0, 255] \end{bmatrix} \longrightarrow \text{RGB pixel depth: } (3 \cdot 8) = 24 \text{ bits; can represent } 2^{24} \text{ different colors}$$

Previously: represented a grayscale image f via (i) a scalar function, or (ii) a 2D matrix $[M \times N]$

→ Now: RGB image - (i) vector-valued function, or (ii) 3D matrix $[M \times N \times 3]$

$$f(x, y) = \vec{c}(x, y) = \begin{bmatrix} c_R(x, y) \\ c_G(x, y) \\ c_B(x, y) \end{bmatrix}$$

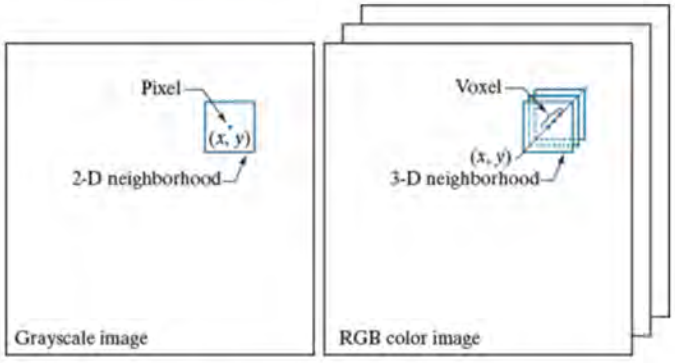


Figure 63: Representation of an RGB image as a function.

The RGB Color Model (cont.)

Basics of Full-Color Image Processing

For an $M \times N$ RGB image, each pixel is a vector $\vec{c}(x,y) \in \{0,\dots,255\}^3$ $[x=0,1,\dots,M-1;\ y=0,1,\dots,N-1]$

→ 2 main categories of full-color image processing techniques:

1. Per-component processing: Process each grayscale component separately, then combine at the end to form the final color image (popular approach)
2. Vector-based processing: Processes color pixels directly as vectors in 3D space ← HSI filters (later)

RGB Image Smoothing & Sharpening

For smoothing, can apply a linear smoothing filter (e.g. arithmetic mean) to each component:

$$\vec{g}(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} \vec{c}(s,t) = \frac{1}{mn} \sum_{s,t} \begin{bmatrix} c_R(s,t) \\ c_G(s,t) \\ c_B(s,t) \end{bmatrix}$$

[Arithmetic mean filter]

For sharpening, can perform a similar process with the Laplacian operator $[g = f - \Delta f]$:

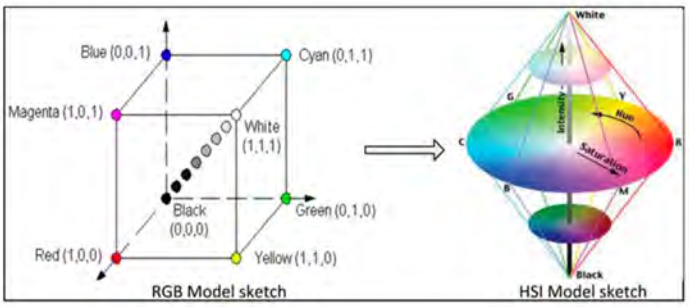
$$\vec{g}(x,y) = \vec{c}(x,y) - \Delta\vec{c}(x,y) = \begin{bmatrix} c_R(x,y) \\ c_G(x,y) \\ c_B(x,y) \end{bmatrix} - \begin{bmatrix} \Delta c_R(x,y) \\ \Delta c_G(x,y) \\ \Delta c_B(x,y) \end{bmatrix}$$

Figure 68: RGB representation of the image.

The HSI Color Model (§5.2)

Hue, Saturation, & Intensity [HSI]: A more natural & intuitive method for representing colors (compared to RGB)

1. **Hue (H)**: Represents "color type" as an angle $\in [0°, 360°]$
2. **Saturation (S)**: Represents "purity" of a color (i.e. the degree to which a pure color has been diluted by white light); given within a range from 0 (gray, completely diluted) to 1 (pure color)
3. **Intensity (I)**: Represents the brightness of the color, $\in [0, 1]$



Computing HSI

Given an RGB image $\vec{c}(x,y) \in [0,1]^3$, can convert to HSI pixel-by-pixel:

1. **Intensity**:

$$I = \frac{1}{3}(c_R + c_G + c_B)$$

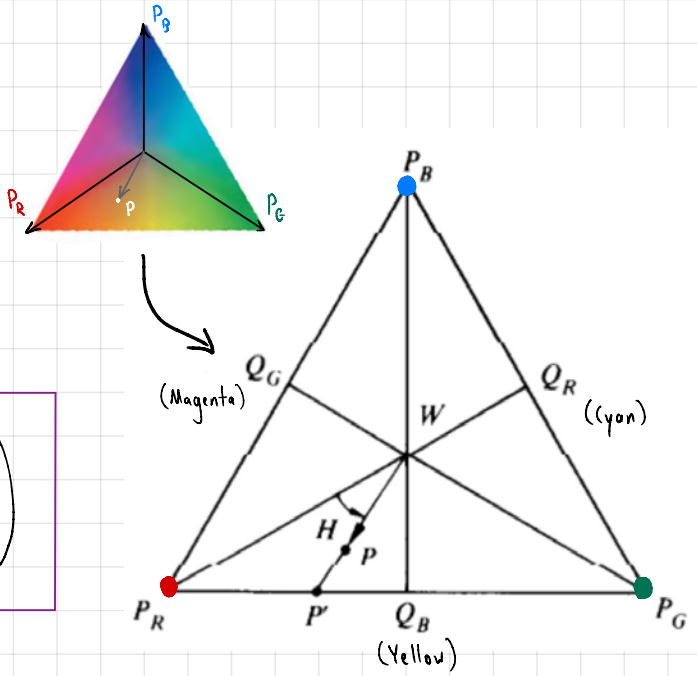
2. **Saturation**:

$$S = 1 - \frac{\min\{c_R, c_G, c_B\}}{I}$$

3. **Hue**: Given by the angle between the vectors $P_R - W$, $P - W$ (pictured right):

$$H(x,y) = \arccos\left(\frac{(P_R - W)\cdot(P - W)}{|P_R - W|\cdot|P - W|}\right)$$

$$\Theta = \arccos\left(\frac{(c_R - c_G) + (c_R - c_B)}{2\sqrt{(c_R - c_B)^2 + (c_R - c_B)(c_G - c_B)}}\right)$$



Can also convert HSI → RGB:

(i) For $0° \le H \le 120°$:

$$c_B = I(1-S)$$

$$c_R = I\left(1 + \frac{S\cos(H)}{\cos(60° - H)}\right)$$

$$c_G = 3I - (c_R + c_B)$$

(ii) If $120° \le H \le 240°$:

$$c_B = I(1-S)$$

$$c_R = I\left(1 + \frac{S\cos(H - 120°)}{\cos(60 + H)}\right)$$

$$c_G = 3I - (c_R + c_B)$$

(iii) If $240° \le H \le 360°$:

$$c_B = I(1-S)$$

$$c_R = I\left(1 + \frac{S\cos(H - 240°)}{\cos(180 + H)}\right)$$

$$c_G = 3I - (c_G + c_B)$$

Figure 66: An image with its hue, saturation, and intensity components.

The HSI Color Model (§5.2)

HSI Color Image Processing

HSI decouples color information (hue & saturation) from intensity (I) → better than RGB (RGB: coupled color & intensity representation => cannot process intensity w/o also modifying color) for various applications:

- Can apply previously-seen grayscale processing techniques to HSI images
- Only need to process intensity (1 component) instead of R + G + B (3) → more computationally efficient

HSI Histogram Processing

Recall: Histogram equalization aims to enhance image contrast by redistributing intensity values (toward a more uniform histogram):

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p(r_j)$$

Applying histogram equalization separately to each RGB channel can lead to color distortions

→ Instead: apply histogram equalization to just the intensity component of an HSI image

↖ may need to convert from & back to RGB

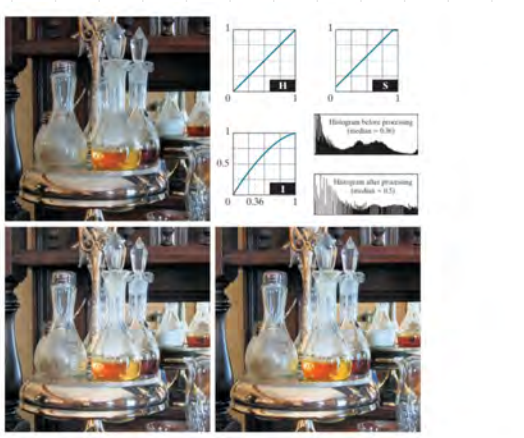


Figure 67: From left to right, top to bottom: original image, transformation used for HSI component and intensity histogram, results with only histogram adjustment on intensity, results with further adjustment on Saturation.

(Original image) ↘

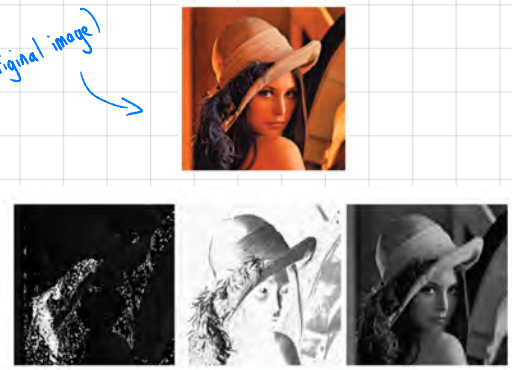


Figure 69: HSI representation of the image.

Figure 70: Comparison of smoothing in RGB, HSI models and their difference. The HSI model preserves color.

Figure 71: Comparison of sharpening in RGB, HSI models, and their difference. The HSI model preserves color.

HSI Smoothing & Sharpening

For image smoothing & sharpening, filtering only the I component while keeping H, S unchanged prevents unwanted color alterations

("color")

(Vs. RGB: diff. channels are filtered independently → may result in color changes/distortions)

Color Image Completion & Segmentation (§5.3)

Color Image Completion

HSI hue represented as an angle $\in [0°, 360°]$
→ can use to form a color circle (right)

For any color, define its complementary color as its opposite on the color circle.

Complementary colors analogous to grayscale negatives; like negatives, can use to enhance image details in dark regions

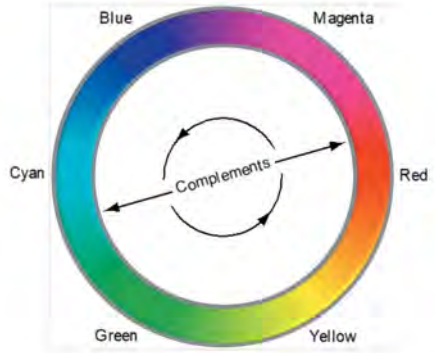


Figure 72: Color circle showing complementary colors.

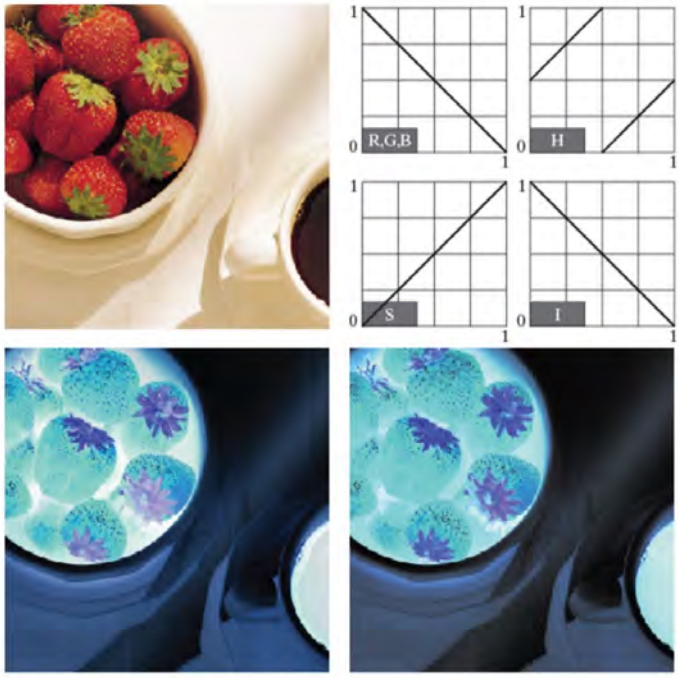


Figure 73: Color completion where the saturation is unchanged, while the intensity and hue are altered. The two images on the bottom are completion results with HSI and RGB model.

Color Image Completion & Segmentation (cont.)

Color Image Segmentation

For color-based segmentation, using HSI is advantageous:

(i) Hue represents color information more effectively/concisely than RGB, particularly for non-primary colors (not red, green, or blue) & mixtures of primary colors

(ii) Saturation can be used as a masking image to further refine segmentation

(iii) Brightness is isolated to its own component [intensity] → resulting segmentation is less sensitive to changes in lighting

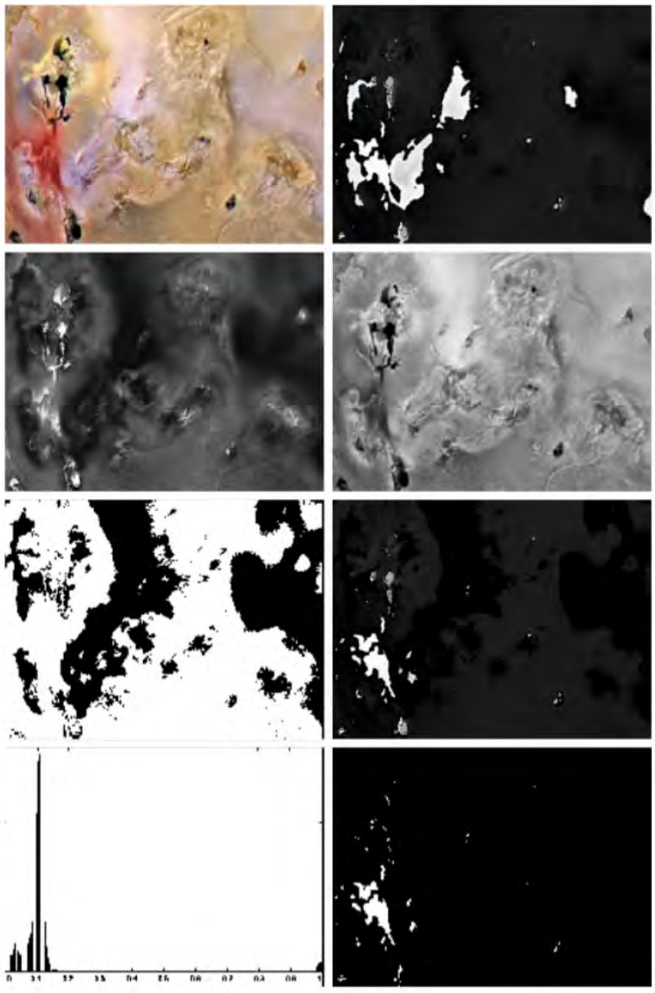


Figure 74: Image segmentation in HSI space. We are aiming to separate the red region in the image. From left to right, top to bottom: Original; Hue; Saturation; Intensity; Binary saturation mask (black = 0); Product of hue and mask; Histogram of the product; Segmentation of red components from (a)..