IMAGE DEBLURRING

KALICHARAN (22MA60R37)

IIT KGP

November 5, 2022

MASTER OF TECHNOLOGY
in
Computer Science and Data Processing

DEPARTMENT OF MATHEMATICS
IIT KHARAGPUR, WEST BENGAL-721302 INDIA

Outline

What is Image Deblurring ?

When we use a camera, we want the recorded image to be a faithful representation of the scene that we see - but every image is more or less *blurry*, depending on the circumstances.

- Image deblurring is the task of processing the image to make it a better representation of the scene - sharper and more useful.

Cause of Image Blurring

Some blurring always arises in the recording of a digital image, because it is unavoidable that scene information "spills over" to neighboring pixels.

Some blurring arises within the camera:

- The optical system in a camera lens may be out of focus, so that the incoming light is smeared out.
- The lens is not perfect, and light rays with different wavelengths follow slightly different paths (aberration).

Other kinds of blurring come from outside the camera:

- The camera or the object moved during the exposure.
  - **Example:** Motion blur
- In astronomical imaging the incoming light in the telescope is slightly bent by turbulence in the atmosphere.

Image as an Array of Number

Images are typically recorded by a CCD (charge-coupled device), an array of tiny detectors arranged in a rectangular grid, able to record the amount, or intensity, of the light that hits each detector.

- **Pixel :** A digital image is composed of picture elements called pixels.
- Each pixel is assigned an intensity, meant to characterize the color of a small rectangular segment of the scene. The intensity can be binary (black & white image), an integer (grayscale image) or a vector of integers (color/multispectral image).
- A small image typically has around $256^2 = 65536$ pixels while a high-resolution image often has 5 to 10 million pixels.
- We need to represent images as arrays of numbers in order to use mathematical techniques for deblurring.

Grayscale Image

Think of a grayscale digital image as a rectangular $m \times n$ array, whose entries represent light intensities captured by the detectors. Consider the following 10×16 array:

$$X = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 8 & 8 & 0 & 0 & 5 & 5 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

If we enter X into Matlab and display the array with the commands `imagesc(X), axis image, colormap(gray)`, then we obtain

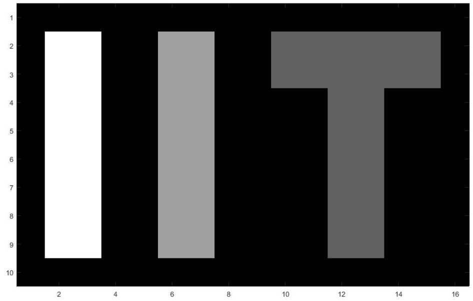


Figure: 1

where:

- 8 is displayed as white
- 0 is displayed as black.
- Values in between are shades of gray.

Color Image

Color images are stored as three components, which represent their intensities on the red, green, and blue scales. Examples:

- $(1;0;0)$ is red
- $(0;0;1)$ is blue
- $(1;1;0)$ is yellow.

Other colors can be obtained with different choices of intensities. For such RGB images, we need three arrays (of the same size) or, a single 3D-array, to represent a color image.

Example: Let X a 3-D array of dimension $10 \times 16 \times 3$ define as

$$X(:,:,1) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$X(:,:,2) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$X(:,:,3) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&0 \\ 0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

The command `imagesc(X)` gives the picture

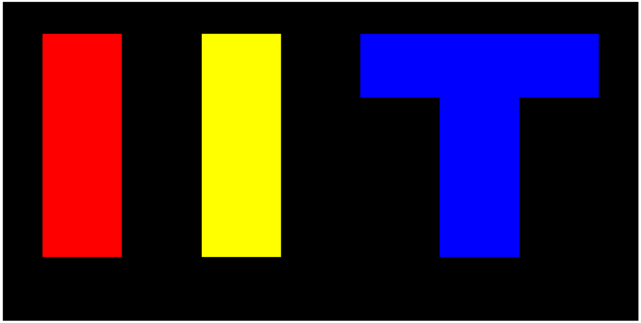


Figure: 2

Simple Linear Model

A mathematical model that relates the given blurred image to unknown true image

Figure: 3: A sharp image(left) and the corresponding blurred image (right)

Notation:

- $\mathbf{X} = \mathbb{R}^{m\times n}$ represent the desired sharp image, and
- $\mathbf{B} = \mathbb{R}^{m\times n}$ represent the recorded blurred image.

Assume the blurring of columns in the image is independent of blurring of rows. Then there exit two matrices $\mathbf{A}_c \in \mathbf{R}^{m\times m}$ and $\mathbf{A}_r \in \mathbb{R}^{n\times n}$,

$$\mathbf{A}_c \ \mathbf{X} \ \mathbf{A}_r^T \ = \ \mathbf{B} \tag{1}$$

Where matrix $\mathbf{A_c}$ applies the same vertical blurring operation to all n columns $\mathbf{x}_j$ of $\mathbf{X}$, because

$$\mathbf{A}_c\mathbf{X} = \mathbf{A}_c[x_1 \ x_2 \ \ldots \ x_n] = [\mathbf{A}_c x_1 \ \mathbf{A}_c x_2 \ \ldots \ \mathbf{A}_c x_n]$$

similarly, the right multiplication with $\mathbf{A}_r^T$ applies the same horizontal blurring to all m rows of $\mathbf{X}$.

Since matrix multiplication is associative i.e.

$$(\mathbf{A}_c\mathbf{X})\mathbf{A}_r^T = \mathbf{A}_c(\mathbf{X}\mathbf{A}_r^T)$$

It does not matter in which order we perform the blurring operation.
If the image blurring model is the simple form

$$\mathbf{A}_c\ \mathbf{X}\ \mathbf{A}_r^T\ =\ \mathbf{B}$$

then one might think that the naive solution is

$$\mathbf{X}_{naive}\ =\ \mathbf{A}_c^{-1}\ \mathbf{B}\ \mathbf{A}_r^{-T}$$

will yield the desired reconstruction, where $\mathbf{A}_r^{-T} = (\mathbf{A}_r^T)^{-1} = (\mathbf{A}_r^{-1})^T$

The naive reconstruction of pumpkin image , obtained by computing $\mathbf{X} = \mathbf{A}_c^{-1}\, \mathbf{B}\, \mathbf{A}_r^{-T}$ via Gaussian elimination on the both $\mathbf{A}_c$ and $\mathbf{A}_r$.

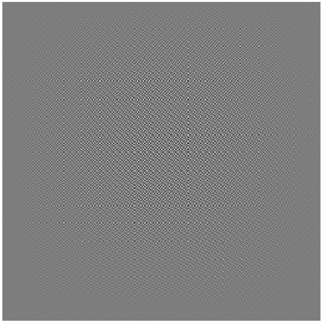


Figure: 4

X_{naive} is completely dominated by the influence of the noise.

What went Wrong ?

To understand the why this naive approach fails, we take a closer look.

- Exact (unknown) image $= \mathbf{X}$
- Noise-free blurred version of image $= \mathbf{B}_{exact} = \mathbf{A}_c \ \mathbf{X} \ \mathbf{A}_r^T$

Small random error (noise) present in the recorded data.

$$\mathbf{B} \ = \ \mathbf{B}_{exact} + \mathbf{E} \ = \ \mathbf{A}_c \ \mathbf{X} \ \mathbf{A}_r^T \ + \mathbf{E}$$

where the matrix $\mathbf{E}$ (of the same dimensions as $\mathbf{B}$) represents the noise in the recorded image.

The naive reconstruction is given by

$$\mathbf{X}_{naive} \;=\; \mathbf{A}_c^{-1}\,\mathbf{B}\,\mathbf{A}_r^{-T} \;+\mathbf{A}_c^{-1}\,\mathbf{E}\,\mathbf{A}_r^{-T}$$

and

$$\mathbf{X}_{naive} \;=\; \mathbf{X} \;+\; \mathbf{A}_c^{-1}\,\mathbf{E}\,\mathbf{A}_r^{-T}$$

where the term

$$\mathbf{A}_c^{-1}\mathbf{E}\mathbf{A}_r^{-T}$$

which is call **inverted noise**, represent the contribution to the reconstruction from additive noise.
This inverted noise will dominate the solution if the second term

$$\mathbf{A}_c^{-1}\mathbf{E}\mathbf{A}_r^{-T}$$

has larger elements than the first term $\mathbf{X}$.
Unfortunately, in many situations, the inverted noise indeed dominates.

A General Linear Model

What is mean of linearity ?

Assume that the operation of going from sharp image to the blurred image is linear *i.e*
if $\mathbf{B}_1$ and $\mathbf{B}_2$ are the blurred images of the exact images $\mathbf{X}_1$ and $\mathbf{X}_2$ respectively. then blurred image

$$\mathbf{B} = \alpha \mathbf{B}_1 + \beta \mathbf{B}_2$$

is the exact image of

$$\mathbf{X} = \alpha \mathbf{X}_1 + \beta \mathbf{X}_2$$

To form more general model, we rearrange the elements of the images $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$ into column vectors by stacking the columns of these images into two long vector $\mathbf{x}$ and $\mathbf{b}$, both of length $N = mn$.

The mathematical notation for this operation is vec,i.e

$$\mathbf{x} = vec(\mathbf{X}) = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \in \mathbb{R}^N, \ \mathbf{b} = vec(\mathbf{B}) = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} \in \mathbb{R}^N$$

Since the blurring is assumed to be linear operation, there must exist a large **blurring matrix A** $\in \mathbb{R}^{N \times N}$ such that **x** and **b** are related by linear model

$$\boxed{\mathbf{A}\mathbf{x} = \mathbf{b}}$$

and this is fundamental image blurring model and we will explain how it can be constructed from the imaging system and what is precise structure of the matrix in coming slides.

The linear model

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

has again naive solution

$$\mathbf{x}_{naive} = \mathbf{A}^{-1}\mathbf{b}$$

where

$$\mathbf{b} = \mathbf{b}_{exact} + \mathbf{e}$$

is the blurred image with noise, $\mathbf{b}_{exact} = vec(\mathbf{B}_{exact})$ is the noise free blurred image and $\mathbf{e} = vec(\mathbf{E})$ is the noise image and

$$\mathbf{x}_{naive} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{A}^{-1}\mathbf{b}_{exact} + \mathbf{A}^{-1}\mathbf{e} = \mathbf{x} + \mathbf{A}^{-1}\mathbf{e}$$

where the term $\mathbf{A}^{-1}\mathbf{e}$ is the inverted noise.

The important observation here is that the deblurred image consists of two components:

- The first component $\mathbf{x}$ is the exact image.
- The second component $\mathbf{A}^{-1}\mathbf{e}$ is the inverted noise

If the deblurred image looks unacceptable, it is because the inverted noise term contaminates the *reconstructed* image.

Important insight about the inverted noise term can be gained using the singular value decomposition (*SVD*), which is the tool-of-the-trade in matrix computations for analyzing linear systems of equations.

SVD Analysis

Singular Value Decomposition (SVD)

The *SVD* of a square matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is essentially unique and is defined as the decomposition

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$

where

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N] \ , \ \mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N]$$

Satisfying $\mathbf{U}^T\mathbf{U} = \mathbf{I}_N$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_N$ and

$$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_N)$$

is a diagonal matrix whose elements σ_i are nonnegative and appear in nonincreasing order,

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_N \geq 0$$

- The quantities σ_i are called the singular values.
- The rank of A is equal to the number of positive singular values.
- The columns $\mathbf{u}_i$ of $\mathbf{U}$ are called the left singular vectors, while the columns $\mathbf{v}_i$ of $\mathbf{V}$ are the right singular vectors.
- $\mathbf{V}$ and $\mathbf{U}$ are are orthogonal matrix therefore $\mathbf{u}_i^T\mathbf{u}_j = 0$ if $i \neq j$ and $\mathbf{v}_i^T\mathbf{v}_j = 0$ if $i \neq j$
- If all singular values are strictly positive,then inverse of A is given by

$$\mathbf{A}^{-1} = (\mathbf{U}\Sigma\mathbf{V}^T)^{-1}$$

$$= (\mathbf{V}^T)^{-1}\Sigma^{-1}\mathbf{U}^{-1}$$

$$\because\ (\mathbf{V}^T)^{-1} = (\mathbf{V}^{-1})^T \text{ and } \mathbf{V}^{-1} = \mathbf{V}^T,\ \mathbf{U}^{-1} = \mathbf{U}^T$$

$$\implies \mathbf{A}^{-1} = (\mathbf{V}^{-1})^T\Sigma^{-1}\mathbf{U}^{-1}$$

$$\implies \mathbf{A}^{-1} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T$$

Another representation of $\mathbf{A}^{-1}$:

$$\begin{aligned}
\mathbf{A}^{-1} &= \mathbf{V}\Sigma^{-1}\mathbf{U}^T \\
&= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_N \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1} & & & \\ & \frac{1}{\sigma_2} & & \\ & & \ddots & \\ & & & \frac{1}{\sigma_N} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_N^T \end{bmatrix} \\
&= \mathbf{v}_1 \frac{1}{\sigma_1} \mathbf{u}_1^T + \mathbf{v}_2 \frac{1}{\sigma_2} \mathbf{u}_2^T + \dots + \mathbf{v}_N \frac{1}{\sigma_N} \mathbf{u}_N^T \\
&= \sum_{i=1}^{N} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T
\end{aligned}$$

The naive solution to our problem

$$\mathbf{x}_{naive} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{b} = \sum_{i=1}^{N}\left(\frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T\right)\mathbf{b} = \sum_{i=1}^{N}\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i$$

$$\because\ \mathbf{b} = \mathbf{b}_{exact} + \mathbf{e}$$

$$\therefore \mathbf{x}_{naive} = \mathbf{A}^{-1}\mathbf{b} + \mathbf{A}^{-1}\mathbf{e} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{b} + \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{e}$$

$$= \sum_{i=1}^{N}\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i + \sum_{i=1}^{N}\frac{\mathbf{u}_i^T\mathbf{e}}{\sigma_i}\mathbf{v}_i$$

and inverted noise contribution to the solution is

$$\mathbf{A}^{-1}\mathbf{e} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{e} = \sum_{i=1}^{N}\frac{\mathbf{u}_i^T\mathbf{e}}{\sigma_i}\mathbf{v}_i$$

Why does the error term Dominate ?

- The error components $|\mathbf{u}_i^T e|$ are small and typically of roughly the same order of magnitude for all i.
- The singular values decay to a value very close to zero. As a consequence the condition number

  $$cond(\mathbf{A}) = \frac{\sigma_1}{\sigma_N}$$

  is very large, indicating that the solution is very sensitive to perturbation and rounding errors.
- The singular vectors corresponding to the smaller singular values typically represent higher-frequency information. *i.e.* as i increases, the vectors $\mathbf{u}_i$, and $\mathbf{v}_i$,tend to have more sign changes.

The consequence of the last property is that the *SVD* provides us with basis vectors $\mathbf{v}_i$, for an expansion where each basis vector represents a certain "frequency," approximated by the number of times the entries in the vector change signs.

Reshaped the each singular vectors $\mathbf{v}_i$ into an $m \times n$ array $\mathbf{V}_i$

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ v_{i2} \\ v_{i3} \\ v_{i4} \\ \vdots \\ \vdots \\ v_{iN} \end{bmatrix}_{N\times 1} = \begin{bmatrix} v_{i1} & v_{i(m+1)} & \cdots & v_{i((n-1)m+1)} \\ v_{i2} & v_{i(m+2)} & \cdots & v_{i((n-1)m+2)} \\ \vdots & \vdots & \ddots & \vdots \\ v_{im} & v_{i(2m)} & \cdots & v_{i(nm)} \end{bmatrix}_{m\times n} = \mathbf{V}_i$$

Interpreting the Coefficient of the solution

$$\mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T = \sum_{i=1}^{N} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i$$

An expansion of the form $\sum_{i=1}^{N} \xi_i \mathbf{v}_i$, where $\xi_i = \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}$. then the i^{th} expansion coefficient ξ_i , for the basis vector $\mathbf{v}_i$.

- For $\mathbf{A}^{-1}\mathbf{e}$, The quantities $\frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}$ are the expansion coefficients for the basis vectors $\mathbf{v}_i$.
- When these quantities are small in magnitude, the solution has very little contribution from $\mathbf{v}_i$
- but when we divide by a small singular value such as σ_N, we greatly magnify the corresponding error component, $\mathbf{u}_N^T\mathbf{e}$, which in turn contributes a large multiple of the high-frequency information contained in $\mathbf{v}_N$ to the computed solution

An Improvement solution Through Truncation

This is precisely why a naive reconstruction, such as the one in Figure 4, appears as a random image dominated by high frequencies.
Because of this, It is better to chop out of the high-frequency components altogether
we can replace

$$\mathbf{x}_{naive} = \mathbf{A}^{-1}\mathbf{b} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{b} = \sum_{i=1}^{N} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i$$

by

$$\mathbf{x}_k = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i == \mathbf{A}_k^\dagger\mathbf{b}$$

for some choice of $k < N$.
where

$$\mathbf{A}_k^\dagger = \sum_{i=1}^{k} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T$$

and pseudoinverse $\mathbf{A}_k^\dagger$ is the rank-k matrix :

$$\mathbf{A}_k^\dagger = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_k \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_k^T \end{bmatrix}$$

Point Spread function (PSF)

In the linear model, blur image **b** and exact image **x** related by the equation

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where **A** is the large matrix of dimension $N \times N$ with $N = mn$ and represent the blurring that is taking place in the process of going from the exact to the blurred image.
How to get it matrix **A**:

- Suppose we take the exact image to be all black, except for a single bright pixel. if we take a picture of this image, then the blurring operation will cause the single bright pixel to be spread over its neighbouring pixels.
- Single bright pixel is called a **point source**

Point Spread Function (PSF)

The function that describes the blurring and resulting image of the point source is called the point spread function (PSF)

In Mathematically

- the point source is equivalent to defining an array of all zeros, except a single pixel whose value is 1 i.e. $\mathbf{x} = \mathbf{e}_i$ (the i^{th} unit vector)
- The image of point source,

$$\mathbf{A}\mathbf{e}_i = \mathbf{a}_i = column\ i\ of\ \mathbf{A}$$

  representing the corresponding PSF.
- In this ways we can fully assemble the blurring matrix $\mathbf{A}$.

The blurring matrix **A** is determined from two ingredients:

- The PSF, which defines how each pixel is blurred
- and boundary conditions, which specify our assumptions on the scene just outside our image.

As a consequence of this linear and local nature of the blurring, to conserve the storage

- PSF can represent using an array **P** of much smaller dimension than blurred image.
- The light intensity of the PSF is confined to a small area around the center of the PSF (the pixel location of the point source) and outside a certain radius. the intensity is essentially zero.
- PSF is the same regardless of the location of the point source this show that blurring is the spatially invariant.

Some PSF Model

- **Horizontal Motion blur:** Smears a point source into a line, if lines cover r pixels - over which light is distributed, then magnitude of each nonzero element p_{ij} of in PSF array **P** is r^{-1} i.e.

  $$p_{ij} = \frac{1}{r}$$

  The same is true for vertical motion blur.

- **Out of focus blur:** given by

  $$p_{ij} = \begin{cases} \frac{1}{\pi r^2} & \text{if } (i-k)^2 + (j-l)^2 \le r^2 \\ 0 & \text{otherwise} \end{cases}$$

  where $(k, l)$is the center of P and r is the radius of blur.

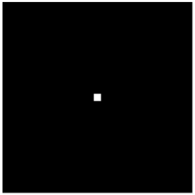
- **Atmospheric turbulence blur:** It can be described as two dimensional Gaussian function and given by

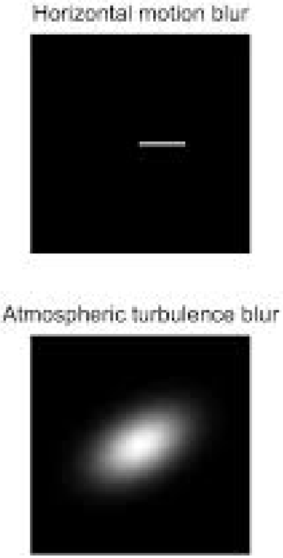
$$p_{ij} = \frac{1}{2\pi s_1 s_2} exp\left(-\frac{1}{2}\left(\frac{i-k}{s_1}\right)^2 - \frac{1}{2}\left(\frac{j-l}{s_2}\right)^2\right)$$

- **Astronomical telescope blur:** It is given by Moffat function as

$$p_{ij} = \left(1 + \left(\frac{i-k}{s_1}\right)^2 + \left(\frac{j-l}{s_2}\right)^2\right)^{-\beta}$$

Figure: point source




Horizontal motion blur
Out-of-focus blur
Atmospheric turbulence blur
Moffat blur

Basic Structures of Matrix A

The specific structure of the matrix **A** depends on the imposed boundary conditions.

one dimensional problem

- By convolving a PSF with a true image, we obtain a blurred image.
- Convolution is mathematical operation denoted by $*$,then the convolution of x and p is a function b having the form

$$b(s) = x(s) * p(s) = \int_{-\infty}^{\infty} x(t)p(s-t)dt$$

where $p(s)$ and $x(s)$ are continuous functions

Convolution in discrete:

- Pixel of the blurred image are obtained from a weighted sum of the corresponding pixel and its neighbors in true image.
- The weights are given by the element in the PSF array.
- Consider a small example, Suppose true image scene and PSF array are given, respectively

$$\mathbf{x} = \left[\begin{array}{c} w_1 \\ w_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline y_1 \\ y_2 \end{array}\right] \quad and \ \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix}$$

where w_i and y_i represent pixel in the original scene that are actually outside the field of view.

- Computing the convolution, **b** of **x** and **p** can be summarised as follows:
  - Rotate (i.e flip) the PSF array, **p** by 180 degree, writing its elements from bottom to top.
  - Match coefficient of the rotated PSF array with those in **x** by placing (i.e. shifting) the center of the PSF array(i.e., the entry corresponding to a shift of zero) over the i^{th} entry in **x**
  - Multiply corresponding component, and sum to get the i^{th} entry in **b**

For example, p_3 is the is the center of the PSF array, then

$$b_1 = p_5 w_1 + P_4 w_2 + p_3 x_1 + p_2 x_2 + p_1 x_3$$

$$b_2 = p_5 w_2 + P_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4$$

$$b_3 = p_5 x_1 + P_4 x_2 + p_3 x_3 + p_2 x_4 + p_1 x_5$$

$$b_4 = p_5 x_2 + P_4 x_3 + p_3 x_4 + p_2 x_5 + p_1 y_1$$

$$b_5 = p_5 x_3 + P_4 x_4 + p_3 x_5 + p_2 y_1 + p_1 y_2$$

Convolution can be written as a matrix-vector multiplication

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_5 & p_4 & P_3 & p_2 & p_1 & & & & \\ & p_5 & p_4 & P_3 & p_2 & p_1 & & & \\ & & p_5 & p_4 & P_3 & p_2 & p_1 & & \\ & & & p_5 & p_4 & P_3 & p_2 & p_1 & \\ & & & & p_5 & p_4 & P_3 & p_2 & p_1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline y_1 \\ y_2 \end{bmatrix}$$

Boundary Condition

- Zero Boundary Condition $\implies w_i = y_i = 0$

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} P_3 & p_2 & p_1 & & \\ p_4 & P_3 & p_2 & p_1 & \\ p_5 & p_4 & P_3 & p_2 & p_1 \\ & p_5 & p_4 & P_3 & p_2 \\ & & p_5 & p_4 & P_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

A Matrix whose entries are constant on each diagonal, is called **Toeplitz matrix** with parameters **p**

- Periodic Boundary Condition
  $\implies w_1 = x_4,\ w_2 = x_5,\ y_1 = x_1 \ and \ y_2 = x_2$

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} P_3 & p_2 & p_1 & p_5 & p_4 \\ p_4 & P_3 & p_2 & p_1 & p_5 \\ p_5 & p_4 & P_3 & p_2 & p_1 \\ p_1 & p_5 & p_4 & P_3 & p_2 \\ p_2 & p_1 & p_5 & p_4 & P_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

A Toeplitz matrix in which each row(column) is a periodic shift of its previous row(column) is called **Circulant matrix**

- Reflexive Boundary Condition

$\implies w_1 = x_2, \; w_2 = x_1, \; y_1 = x_5 \; and \; y_2 = x_4$

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \left(\begin{bmatrix} P_3 & p_2 & p_1 & & \\ p_4 & P_3 & p_2 & p_1 & \\ p_5 & p_4 & P_3 & p_2 & p_1 \\ & p_5 & p_4 & P_3 & p_2 \\ & & p_5 & p_4 & P_3 \end{bmatrix} + \begin{bmatrix} P_4 & p_5 & & & \\ p_5 & & & & \\ & & & & \\ & & & & p_1 \\ & & & p_1 & P_2 \end{bmatrix} \right) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$$= \begin{bmatrix} P_3 + p_4 & p_2 + p_5 & p_1 & & \\ p_4 + p_5 & P_3 & p_2 & p_1 & \\ p_5 & p_4 & P_3 & p_2 & p_1 \\ & p_5 & p_4 & P_3 & p_2 + p_1 \\ & & p_5 & p_4 + P_1 & P_3 + p_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

A matrix whose entries are constant on each antidiagonal is called a **Hankel matrix**. so above matrix is Toeplitz+Hankel matrix.

Two dimensional problems

The convolution operation for two dimension problem image is similar to the one dimensional case. To compute b_{ij}

- Rotate the **P** by 180 degree.
- Locate it at the desired position.
- Match coefficient of rotated PSF and **X**
- Multiply corresponding component and sum them.

For Example, to compute b_{22}

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & \color{red}{x_{22}} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, \ \mathbf{P} = \begin{bmatrix} \color{blue}{p_{11}} & \color{blue}{p_{12}} & \color{blue}{p_{13}} \\ \color{blue}{p_{21}} & \color{blue}{p_{22}} & \color{blue}{p_{23}} \\ \color{blue}{p_{31}} & \color{blue}{p_{32}} & \color{blue}{p_{33}} \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & \color{red}{b_{22}} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$\begin{aligned} b_{22} &= p_{33}.x_{11} + p_{32}.x_{12} + p_{31}.x_{13} \\ &+p_{23}.x_{21} + p_{22}.x_{22} + p_{21}.x_{23} \\ &+p_{13}.x_{31} + p_{12}.x_{32} + p_{11}.x_{33} \end{aligned}$$

Boundary Condition

- **Zero Boundary condition:** element b_{11} and b_{12} at boarder of **B** is given by

$$\begin{aligned} b_{11} &= p_{33}.0 + p_{32}.0 + p_{31}.0 \\ &+p_{23}.0 + p_{22}.x_{11} + p_{21}.x_{12} \\ &+p_{13}.0 + p_{12}.x_{21} + p_{11}.x_{22} \end{aligned}$$

$$\begin{aligned} b_{21} &= p_{33}.0 + p_{32}.x_{11} + p_{31}.x_{12} \\ &+p_{23}.0 + p_{22}.x_{21} + p_{21}.x_{22} \\ &+p_{13}.0 + p_{12}.x_{31} + p_{11}.x_{32} \end{aligned}$$

Similarly we can write all element of **B**

And $\mathbf{b} = vec(\mathbf{B})$, $\mathbf{x} = vec(\mathbf{X})$ are related by

$$\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \\ b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & & p_{21} & b_{11} & & & & \\ p_{32} & p_{22} & p_{12} & p_{31} & b_{21} & p_{11} & & & \\ & p_{32} & p_{22} & & b_{31} & p_{21} & & & \\ \hline p_{23} & p_{13} & & p_{22} & b_{12} & & p_{21} & p_{11} & \\ p_{33} & p_{23} & p_{13} & b_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ & p_{33} & p_{23} & & p_{32} & b_{22} & & p_{31} & p_{21} \\ \hline & & & p_{23} & p_{13} & & p_{22} & b_{12} & \\ & & & p_{33} & p_{23} & b_{13} & p_{32} & b_{22} & b_{12} \\ & & & & p_{33} & b_{23} & & b_{32} & b_{22} \end{array}\right] \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}$$

Matrix has a **block Toeplitz** structure and each **block** is itself a **Toeplitz matrix**. Such a matrix called **block Toeplitz with Toeplitz blocks (BTTB)**

- **Periodic Boundary Condition :** element b_{21} at border of **B** is given by

$$\begin{aligned} b_{21} &= p_{33}.x_{13} + p_{32}.x_{11} + p_{31}.x_{12} \\ &+ p_{23}.x_{23} + p_{22}.x_{21} + p_{21}.x_{22} \\ &+ p_{13}.x_{33} + p_{12}.x_{31} + p_{11}.x_{32} \end{aligned}$$

similarly we can compute other element of **B**. and matrix-vector form given by:

$$\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \\ b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & p_{32} & p_{21} & b_{11} & p_{31} & p_{23} & p_{13} & p_{33} \\ p_{32} & p_{22} & p_{12} & p_{31} & b_{21} & p_{11} & p_{33} & p_{23} & p_{13} \\ p_{12} & p_{32} & p_{22} & p_{11} & b_{31} & p_{21} & p_{13} & p_{33} & p_{23} \\ \hline p_{23} & p_{13} & p_{33} & p_{22} & b_{12} & p_{32} & p_{21} & p_{11} & p_{31} \\ p_{33} & p_{23} & p_{13} & b_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & b_{22} & p_{11} & p_{31} & p_{21} \\ \hline p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} & p_{22} & b_{12} & p_{32} \\ p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & b_{13} & p_{32} & b_{22} & b_{12} \\ p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & b_{23} & p_{12} & b_{32} & b_{22} \end{array}\right] \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}$$

Such matrix **A** called **Block circulant with circulant blocks (BCCB)**

REFRENCES

1. Hansen P.C., Nagy, J. G., and O'Leary, D. P. Deburring Images: Matrices, Spectra, and Filtering, eISBN:978-0-89871-887-4, SIAM Philadelphia,2006
2. Biswa Nath Datta,Numerical linear algebra and applications,second edition, ISBN:978-0-89871-685-6,Society for Industrial and Applied Mathematics3600 University City Science Center Philadelphia, PA, United States, 20 January 2010

THANK YOU