CSC479 Introduction to Computer Vision
Lecture 3

Review of Linear Algebra, Gaussian Filters, Histogram Equalization

Bei Xiao

Last lecture

- Image can represented as a matrix
- Point process
- Linear filter: convolution

Today's lecture

- A brief review of linear algebra
- Gaussian filters
- Image Histogram Equalization
- Basic image processing tutorial

- Homework 2 will be out today.

Essene of linear algebra

Youtube tutorial

https://www.youtube.com/watch?v=kjBOesZCoqc

Basic math of deep learning-based CV

1. Linear Algebra (Today)

http://www.deeplearningbook.org/contents/linear_algebra.html

2. Probability theory (In two weeks).

http://www.deeplearningbook.org/contents/prob.html

•

Take-home reading

Your textbook: Szesliski, Appendix A.

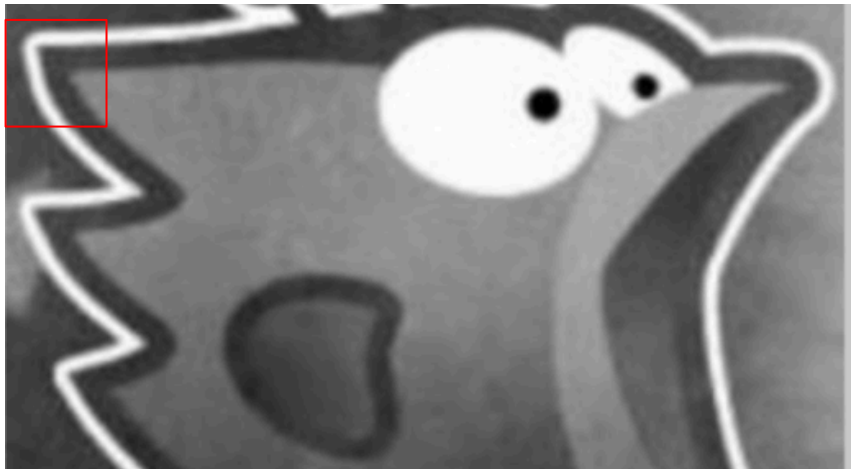
Geometric introduction to linear algebra.

- http://www.cns.nyu.edu/~eero/NOTES/geomLinAlg.pdf

An excellent chapter:

- http://www.deeplearningbook.org/contents/linear_algebra.html

Grayscale Intensity Image as Matrices



row

The pixel P(i,j) has intensity value of 151

column

61	82	121	182	238	246	205	148	96	68	69
59	75	109	170	229	244	214	160	105	73	68
56	63	90	152	217	245	227	178	118	82	71
56	55	74	135	206	246	241	198	135	94	73
57	54	64	118	186	237	248	219	162	119	85
57	55	55	94	151	214	247	238	196	160	116
57	55	48	73	119	193	240	249	222	195	148
58	52	50	66	86	157	206	247	247	225	194
57	53	50	57	69	108	161	219	246	247	229
54	54	50	49	52	72	117	178	226	252	252
53	55	52	51	52	52	74	115	161	201	227

8 bit intensity image

- 8 bit graphics is a method of storing image information in a computer's memory or in an image file, such that each pixel is represented by one 8-bit type.
- The range of 8 bit intensity image is [0 255].

A mini tutorial on linear algebra

Scalars

- A scalar is a single number
- Integers, real numbers, rational numbers, etc.
- We denote it with italic font:

$$a, n, x$$

Vectors

- A vector is a 1-D array of numbers:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.1}$$

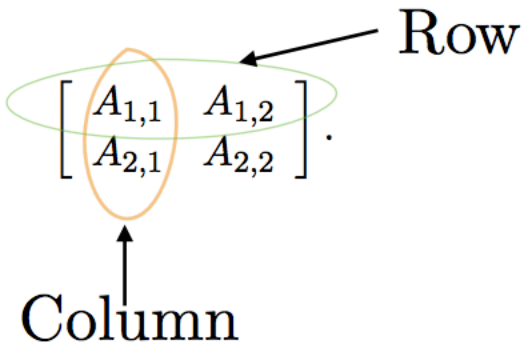
- Can be real, binary, integer, etc.

- Example notation for type and size:

$$\mathbb{R}^n$$

Matrices

- A matrix is a 2-D array of numbers:



(2.2)

- Example notation for type and shape:

$$\boldsymbol{A} \in \mathbb{R}^{m \times n}$$

Tensors

- A tensor is an array of numbers, that may have
  - zero dimensions, and be a scalar
  - one dimension, and be a vector
  - two dimensions, and be a matrix
  - or more dimensions.

Matrix Transpose

$$(\boldsymbol{A}^\top)_{i,j} = A_{j,i}. \tag{2.3}$$

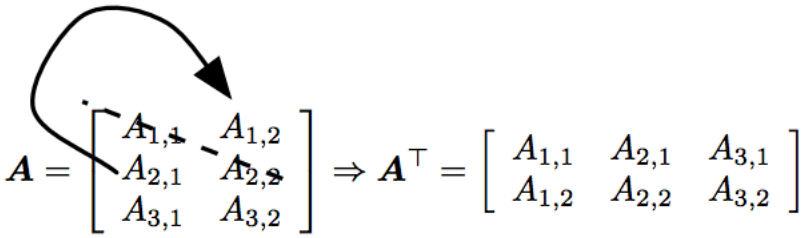


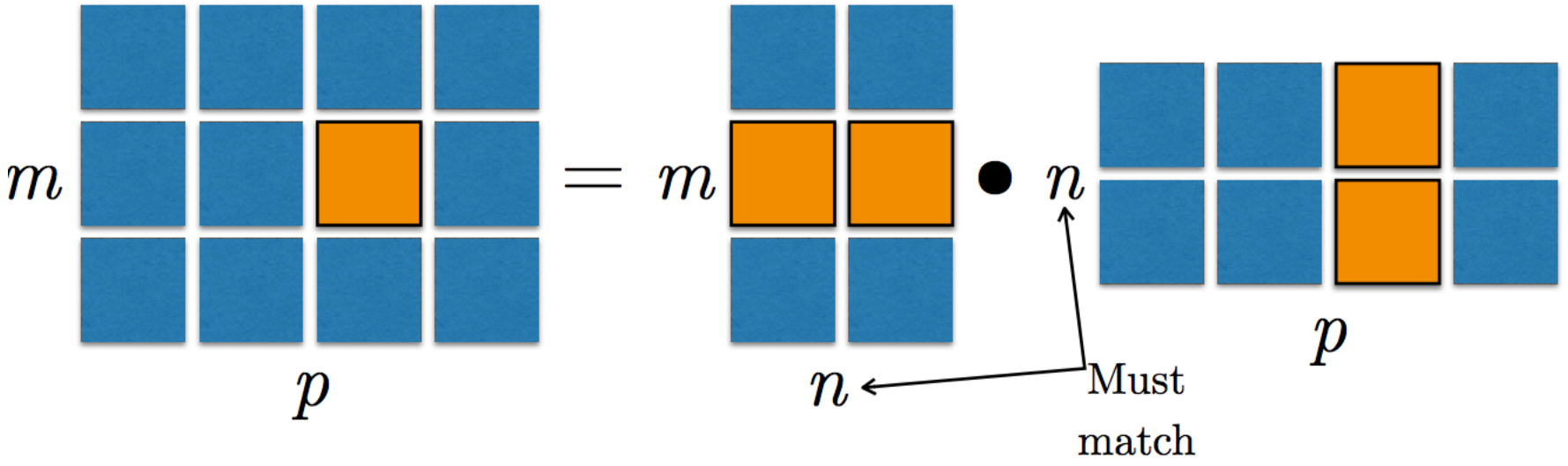
Figure 2.1: The transpose of the matrix can be thought of as a mirror image across the main diagonal.

$$(\boldsymbol{AB})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top. \tag{2.9}$$

Matrix (Dot) Product

$$\boldsymbol{C} = \boldsymbol{AB}. \tag{2.4}$$

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}. \tag{2.5}$$



Matrix product operations have many useful properties that make mathematical analysis of matrices more convenient. For example, matrix multiplication is distributive:

$$\boldsymbol{A}(\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{A}\boldsymbol{C}. \tag{2.6}$$

It is also associative:

$$\boldsymbol{A}(\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{A}\boldsymbol{B})\boldsymbol{C}. \tag{2.7}$$

Matrix multiplication is *not* commutative (the condition $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{A}$ does not always hold), unlike scalar multiplication. However, the dot product between two vectors is commutative:

$$\boldsymbol{x}^\top \boldsymbol{y} = \boldsymbol{y}^\top \boldsymbol{x}. \tag{2.8}$$

The transpose of a matrix product has a simple form:

$$(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top. \tag{2.9}$$

Basic Matrix Operations

Matrix products:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x1 \\ x2 \end{bmatrix}=\begin{bmatrix} ax_1 + bx_2 \\ cx_1 + dx_2 \end{bmatrix}$$

Vector dot product:

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \\ \\ a_m \end{bmatrix}$$

Transpose of the column vector is a row vector

$$\mathbf{a}^T = [a_1, a_2, \cdots, a_m]$$

Identity Matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 2.2: *Example identity matrix*: This is $\boldsymbol{I}_3$.

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{I}_n \boldsymbol{x} = \boldsymbol{x}. \tag{2.20}$$

Dot product of two vectors (Algebraic definition)

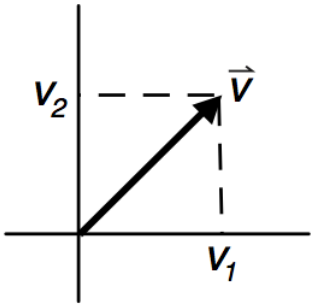
$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \\ \\ b_m \end{bmatrix} \qquad \mathbf{a}^T = [a_1, a_2, \cdots, a_m]$$

Dot product is a sum of pair-wise product of components

$$\begin{aligned} \mathbf{a}^T\mathbf{b} &= \mathbf{b}^T\mathbf{a} = a_1b_1 + a_2b_2 + \cdots + a_mb_m \\ &= \sum_{i=1}^{m} a_ib_i. \end{aligned}$$

What is a vector?

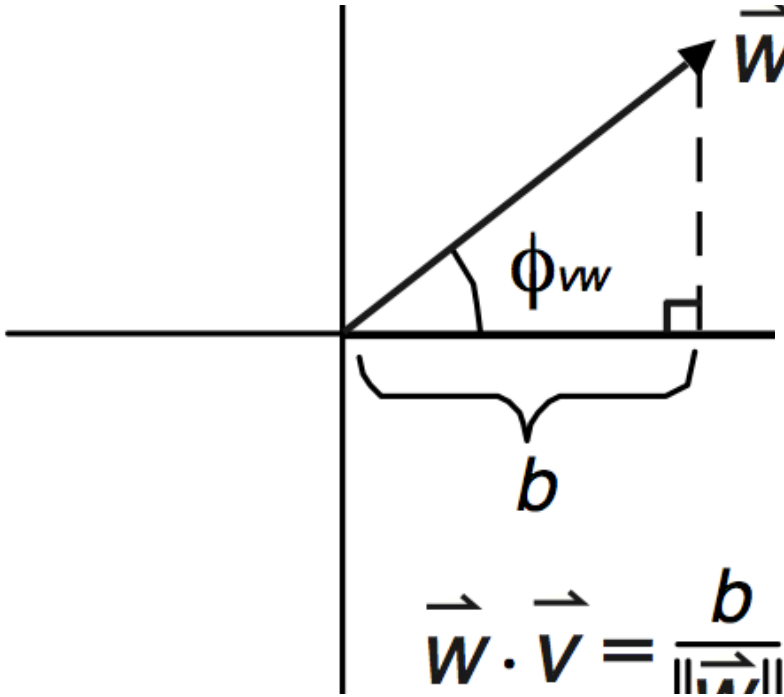
Vectors of dimension 2 or 3 can be graphically depicted as arrows, with the tail at the origin and the head at the coordinate location specific by the vector components.



Vector has a length and a direction. The norm or length is defined as:

$$\|v\| = \sqrt{\sum_n Vn^2}$$

Dot product (inner) of two vectors (Geometric definition)



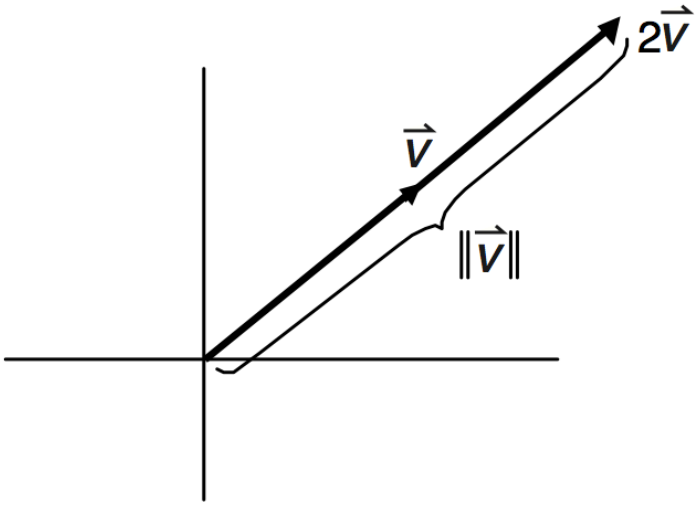
$$\vec{v} \cdot \vec{w} \equiv \|\vec{v}\| \, \|\vec{w}\| \, \cos(\phi_{vw})$$

$$\vec{w} \cdot \vec{v} = \frac{b}{\|\vec{w}\|} \cdot \|\vec{w}\| \cdot \|\vec{v}\|$$

Φ is the angle between the two vectors

Scalar product

- Multiplying a vector by a scalar simply changes the length of the vector by that factor



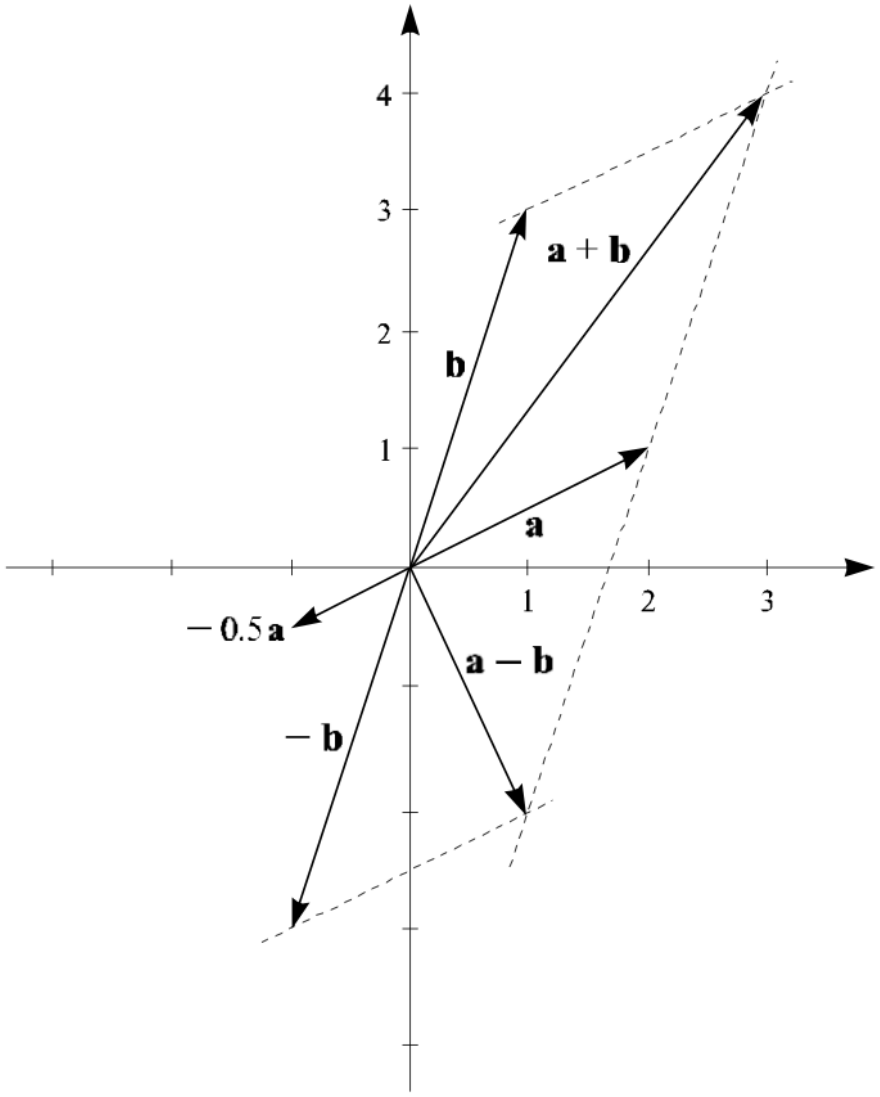
Vector Space

- A vector space is a collection of vectors that is closed under linear combination.

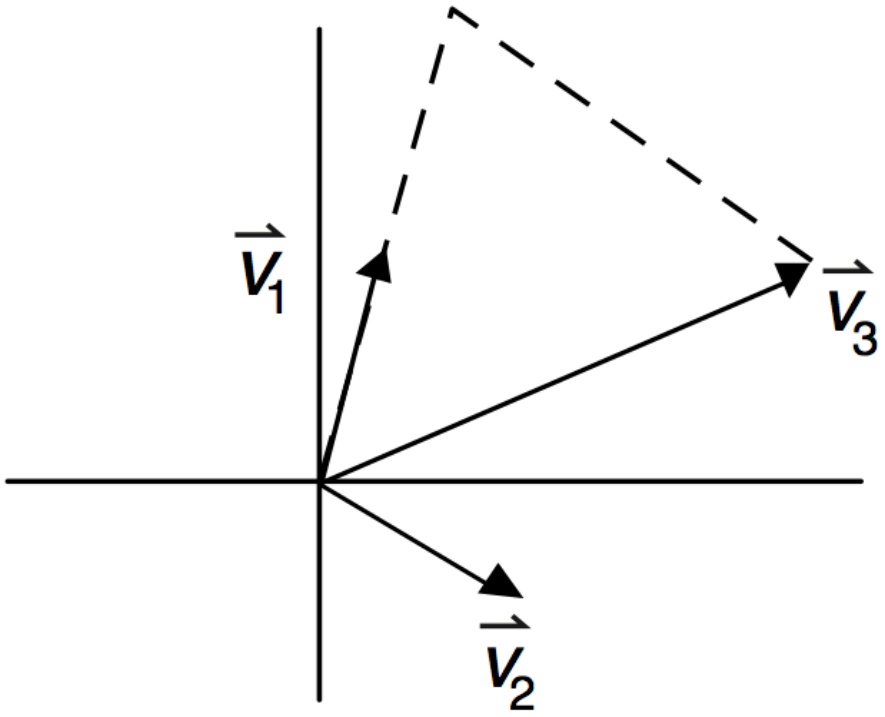
$$\mathbf{a} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad \mathbf{a} + \mathbf{b} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}. \qquad \mathbf{a} - \mathbf{b} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

Vector Space (geometric)



Vector Space (geometric)



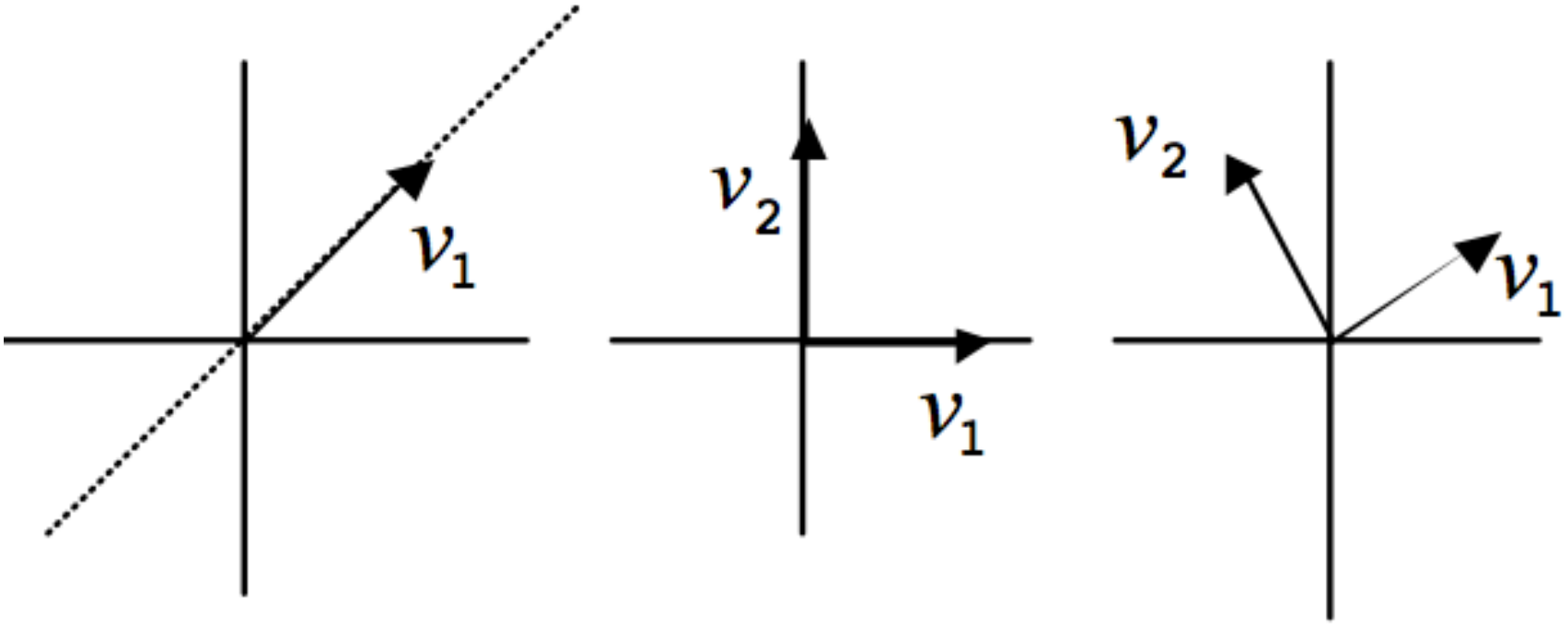
Basis

- A set of vectors in a vector space V is called a basis, or a set of basis vectors.
- A basis B of a vector space V is linearly independent if and only if:

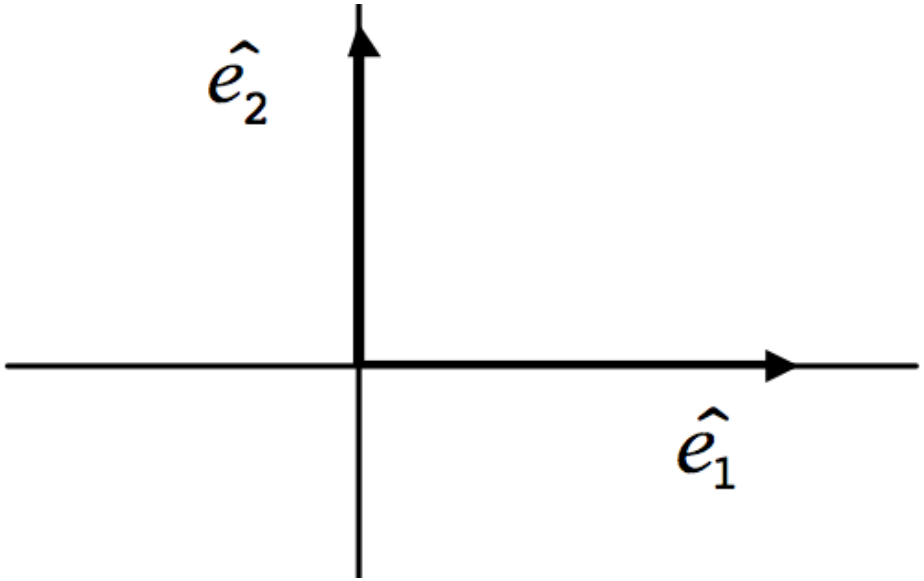
$$\sum_n \alpha_n \vec{v}_n = 0$$

- A basis for a vector space is linearly independent spanning set.

Basis vectors



Standard Basis (unit vectors)



$$\hat{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \; \hat{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \; \ldots \; \hat{e}_N = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

Linear Equations

$$
\begin{aligned}
a_{11}v_1 + a_{12}v_2 + \ldots + a_{1N}v_N &= \\
a_{21}v_1 + a_{22}v_2 + \ldots + a_{2N}v_N &= \\
&\vdots \\
{}_{M1}v_1 + a_{M2}v_2 + \ldots + a_{MN}v_N &=
\end{aligned}
$$

If we put the variables v_n and constant b_m into vectors and the constants am into a matrix A, these equations maybe written more compactly:

$$A\vec{v} = \vec{b}$$

To solve: $\vec{v} = A^{-1}b$

Systems of Equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.11}$$

expands to

$$\boldsymbol{A}_{1,:}\boldsymbol{x} = b_1 \tag{2.12}$$

$$\boldsymbol{A}_{2,:}\boldsymbol{x} = b_2 \tag{2.13}$$

$$\dots \tag{2.14}$$

$$\boldsymbol{A}_{m,:}\boldsymbol{x} = b_m \tag{2.15}$$

Solving Systems of Equations

- A linear system of equations can have:
  - No solution
  - Many solutions
  - Exactly one solution: this means multiplication by the matrix is an invertible function

Matrix Inversion

- Matrix inverse:

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n. \tag{2.21}$$

- Solving a system using an inverse:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.22}$$

$$\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{2.23}$$

$$\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{2.24}$$

- Numerically unstable, but useful for abstract analysis

Matrix Inverse

- Matrix Inverse:

  $A^{-1}A = I$, I is identity matrix

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Inversion of a 2× 2 matrix

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

http://mathworld.wolfram.com/MatrixInverse.html

The determinant (det(A))

The determinant of a square matrix, denoted det(A), is a function that maps matrices to real scalars.

Geometrically, it can be viewed as the volume scaling factor of the linear transformation described by the matrix.

The determinant (det(A))

In the case of a 2 × 2 matrix the determinant may be defined as:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Similarly, for a 3 × 3 matrix A, its determinant is:

$$\begin{aligned} |A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} &= a \begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b \begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c \begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix} \\ &= a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= aei + bfg + cdh - ceg - bdi - afh. \end{aligned}$$

Solving linear equations

Equations
$A\boldsymbol{x} = \boldsymbol{b}$

$$
\begin{aligned}
x_1 &= b_1 \\
-x_1 + x_2 &= b_2 \\
-x_2 + x_3 &= b_3
\end{aligned}
$$

Solution
$\boldsymbol{x} = A^{-1}\boldsymbol{b}$

$$
\begin{aligned}
x_1 &= b_1 \\
x_2 &= b_1 + b_2 \\
x_3 &= b_1 + b_2 + b_3.
\end{aligned}
$$

Norms

- Functions that measure how "large" a vector is
- Similar to a distance between zero and the point represented by the vector
  - $f(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{x} = \boldsymbol{0}$
  - $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (the *triangle inequality*)
  - $\forall \alpha \in \mathbb{R}, f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$

Norms

- L^p norm

$$||\boldsymbol{x}||_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

- Most popular norm: L2 norm, p=2
- L1 norm, p=1: $||\boldsymbol{x}||_1 = \sum_i |x_i|$. (2.31)
- Max norm, infinite p: $||\boldsymbol{x}||_\infty = \max_i |x_i|$. (2.32)

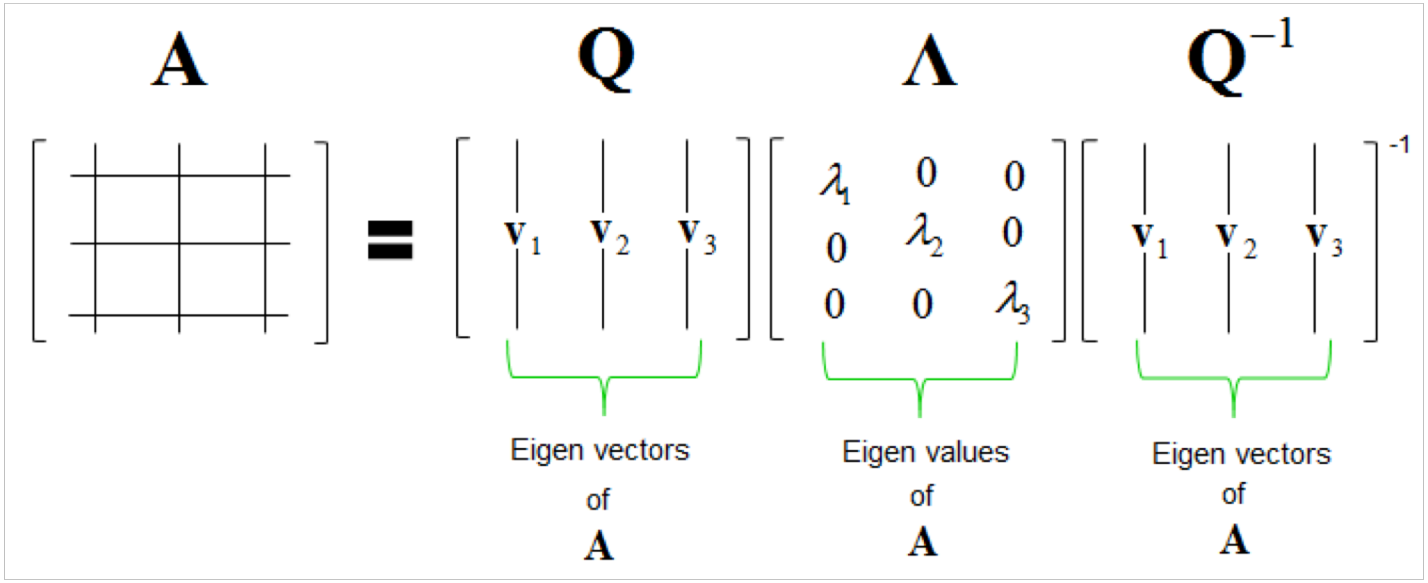
Eigen values and decomposition

Definition: An **eigenvector** of an *n*xn matrix *A* is a nonzero vector **x** such that $A\mathbf{x}=\lambda\mathbf{x}$ for some scalar λ.

- A scalar λ is called an **eigenvalue** of *A* if there is a nontrivial solution **x** of $A\mathbf{x}=\lambda\mathbf{x}$;
- such an **x** is called an *eigenvector corresponding to* λ.

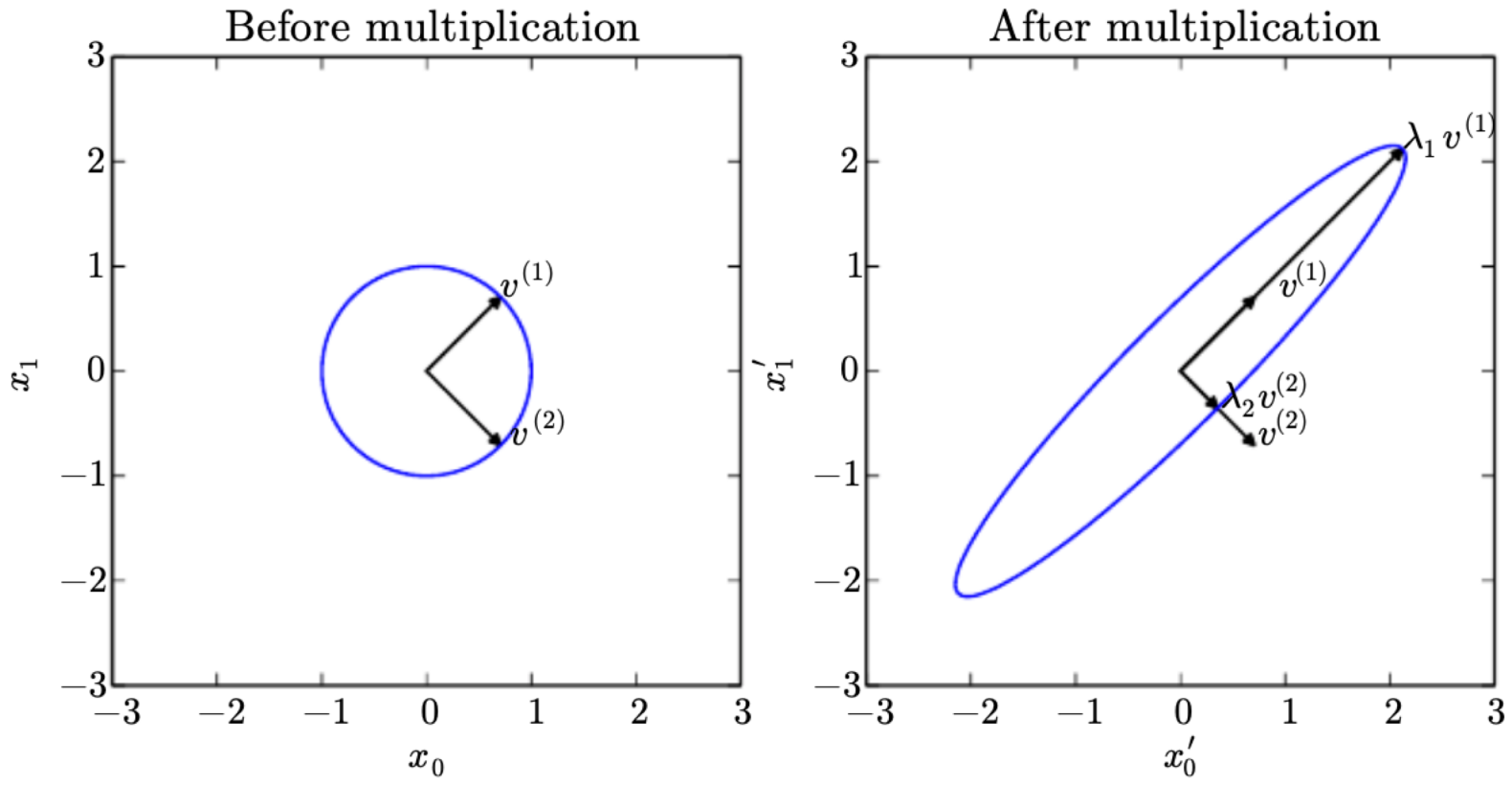
- Eigenvectors may **NOT** be **0**
- Eigenvalue may be 0

Eigen values and decomposition



- Any real symmetric matrix A is guaranteed to have an eigen decomposition, but the decomposition might be unique.
- If any two or more eigenvectorsbshare the same eigenvalue, then any set of orthogonal vectors lying in their span are also eigenvectors with that eigenvalue, and we could equivalently choose an Qb using those eigenvectors instead. By convention, we usually sort the entries ofbΛb in descending order.

Eigenvalues and decomposition



Positive semi-definite matrix

- Definition
  - A nxn real matrix **M** is positive semi-definite

    if $\underline{z}^{T} M \underline{z} \geq 0 \quad \forall \underline{z}$
- Equivalence at real symmetric martix **M**
  - All eigenvalues of **M** ≥ 0
  - All principal minor ≥ 0
  - All diagonal entries of LDU decomposition ≥ 0
  - There exist possibly singular matrix **R** s.t $M = R^{T} R$

Quiz

- Is the identity matrix positive semi-definite?

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- How about this matrix? How to check?

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Linear algebra with Python

Arrays

```
import numpy as np
from numpy.linalg import *
# create a single array
 a = np.array([1, 4, 5, 8], float)
# create a multidimensional arary
 a = np.array([[1, 2, 3], [4, 5, 6]], float)
```

Matrix multiplication Numpy

```
>>> a = np.array([[1, 0], ... [0, 1]])
>>> b = np.array([[4, 1], ... [2, 2]]
>>> np.matmul(a, b)

array([[4, 1], [2, 2]])
```

broadcasting

```
>>> a = np.arange(2 * 2 * 4).reshape((2, 2, 4))
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],


       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]]])

 >>> b = np.arange(2 * 2 * 4).reshape((2, 4, 2))
>>> np.matmul(a,b).shape (2, 2, 2)

 >>> np.matmul(a, b)[0, 1, 1]
98
```

np.dot()

If both *a* and *b* are 1-D arrays, it is inner product of vectors (without complex conjugation).

If both *a* and *b* are 2-D arrays, it is matrix multiplication, but using matmul or a @ b is preferred.

Eigenvalue decomposition: np.linalg.eig

```
>>> from numpy import linalg as LA
>>> a =np.diag((1,2,3))
>>> w, v = LA.eig(a)
>>> w
array([1., 2., 3.])
>>> v
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
>>> LA.eigvals(a)
```

Exercise:

- what is the result of the following matrix product:
- First do this by hand and then check your answer with Numpy.

$$\mathbf{A} = \begin{bmatrix} 1 & -2 \\ 3 & 2 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 1 \end{bmatrix}.$$

Exercise:

- Find out whether the following matrix is positive-semi definite by computing the eigen decomposition

$$M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Exercise: Check next class!!!

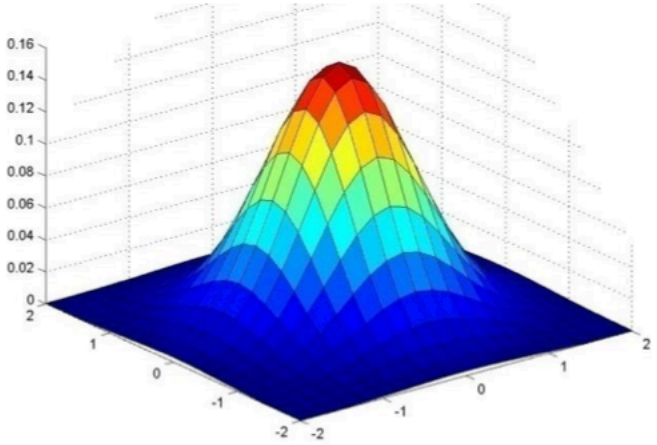
Find the linear combination $3\boldsymbol{s}_1 + 4\boldsymbol{s}_2 + 5\boldsymbol{s}_3 = \boldsymbol{b}$. Then write $\boldsymbol{b}$ as a matrix-vector multiplication $S\boldsymbol{x}$, with $3, 4, 5$ in $\boldsymbol{x}$. Compute the three dot products (row of S) $\cdot\, \boldsymbol{x}$:

$$\boldsymbol{s}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \boldsymbol{s}_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad \boldsymbol{s}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ go into the columns of } S\,.$$

Back to image processing

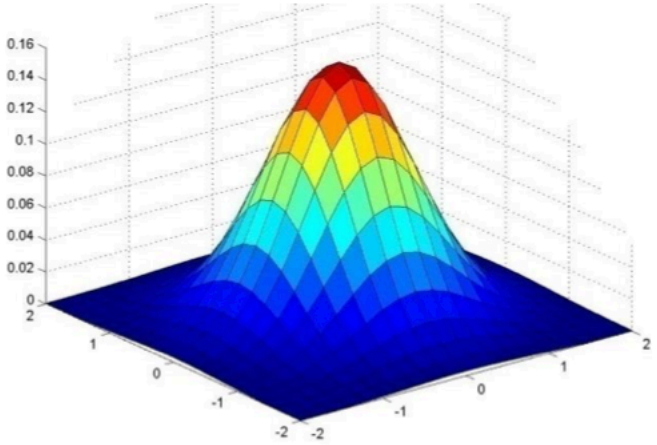
Gaussian Filter

- Gaussian = normal distribution function
-



Gaussian Filter

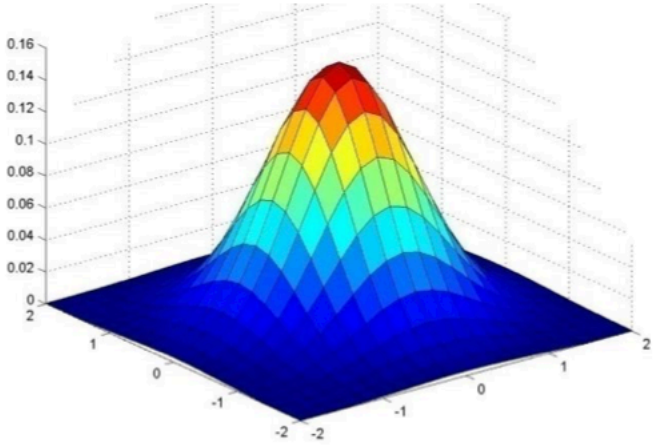
- Gaussian = normal distribution function
-

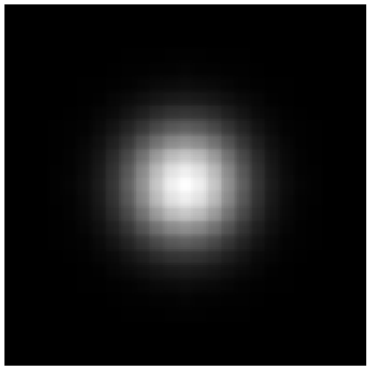


$$K(i,j) = \frac{1}{Z}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

Gaussian Filter

- Gaussian = normal distribution function
-

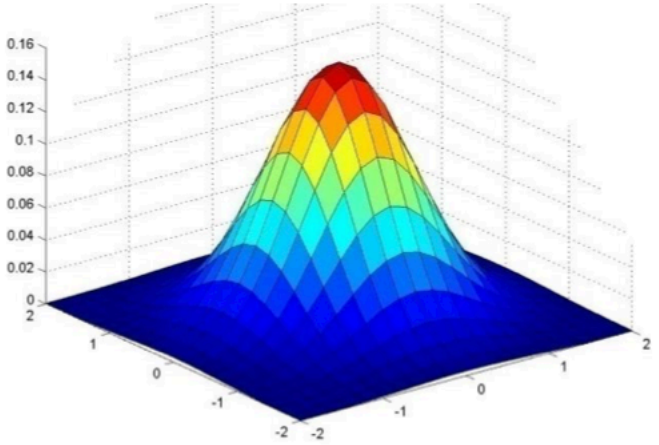


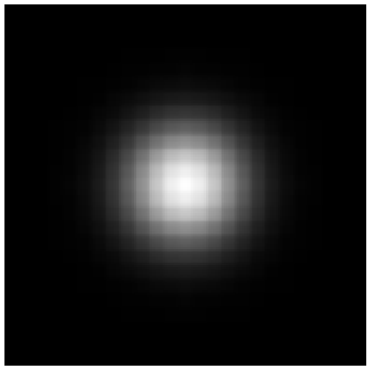


$$K(i,j) = \frac{1}{Z}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

Gaussian Filter

- Gaussian = normal distribution function
-





$$K(i,j) = \frac{1}{Z}\exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

Gaussian filter with different σ

Original

σ = 3

σ = 5

σ = 7

Gaussian Filter

- Remove “high-frequency” components from image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat and get same result as larger-width kernel would have.
  - Convolving two times with kernel width σ is same as convolving once with kernel width $\sigma/2$ (Can you proof this?)

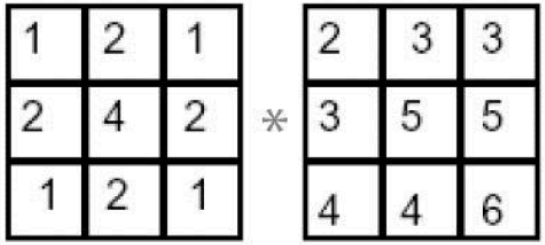
Separability of the Gaussian filter

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
$$

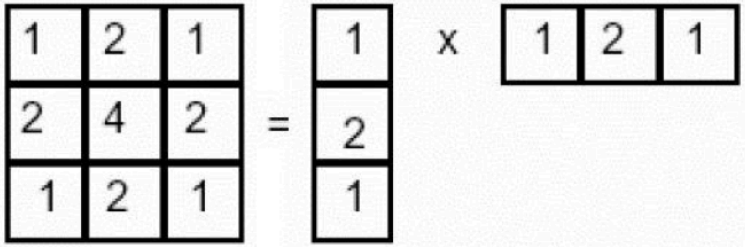
- The 2D Gaussian can be expressed as the product of two functions, one function of x and the other the function of y.
- In this case, the two functions are (identical) 1D Gaussian.

Separability example

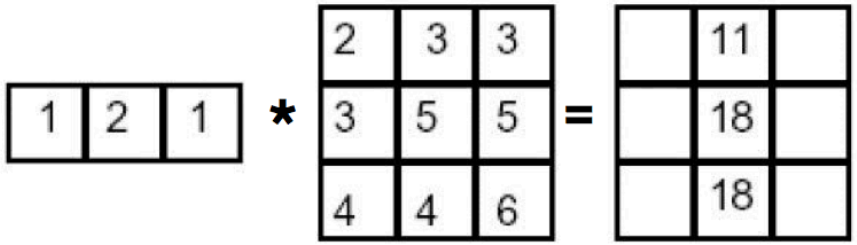
2D convolution
(center location only)



The filter factors
into a product of 1D
filters:



Perform convolution
along rows:



Followed by convolution
along the remaining column:

Source: K. Grauman

Python code

```
import numpy as np
import cv2
from scipy import ndimage

im = np.array([[2,3,3],[3,5,5],[4,4,6]])

mkernal = np.array([[1,2,1],[2,4,2],[1,2,1]])

mkernal_row = np.array([1,2,1])

out = np.outer(np.transpose(mkernal_row),mkernal_row)

res = ndimage.convolve(im,mkernal,mode="constant")

res1 = ndimage.filters.convolve1d(im,mkernal_row,mode="constant",axis=0)
res2 = ndimage.filters.convolve1d(res1,np.transpose(mkernal_row),mode="constant", axis=1)
```

Exercise

Given a 3x3 Gaussian filter, we want to show that

1. Convolve it with an image is the same as convolve once with one 1D Gaussian filter and another 1D Gaussian filter

2. Download the Gaussian.py

Matrix Multiplication

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^{\mathrm{T}} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 \end{bmatrix}.$$

$$[\mathbf{h}_x] = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$[\mathbf{h}_y] = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Why is separability useful?

- The process of performing a convolution requires K^2 operations per pixel
- Suppose v and h are horizontal and vertical kernels.
- K = vh^T
- 2K operations per pixel!

Image Histogram Equalization

- We have a low-contrast image:

Image Histogram Equalization

- We would like to increase the contrast

What is a histogram

$$p_n = \frac{\text{number of pixels with intensity } n}{\text{total number of pixels}} \qquad n = 0, 1, ..., L-1.$$

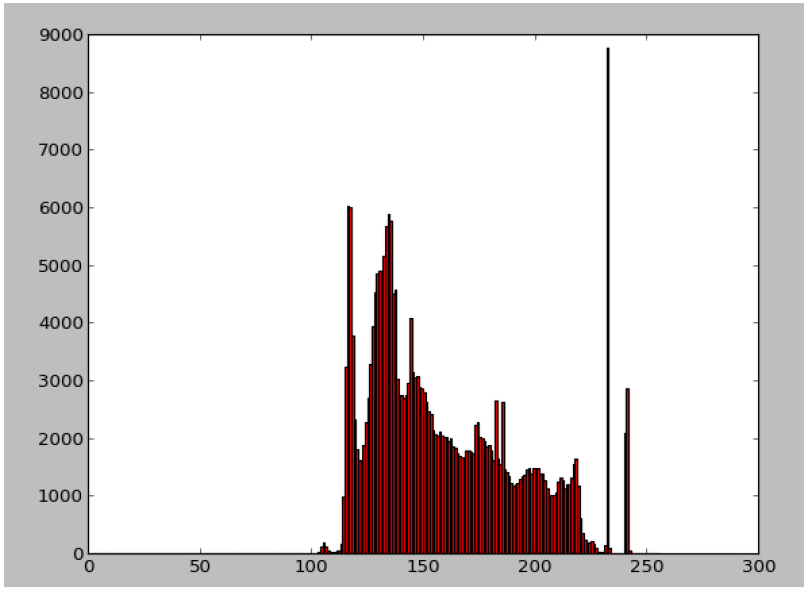
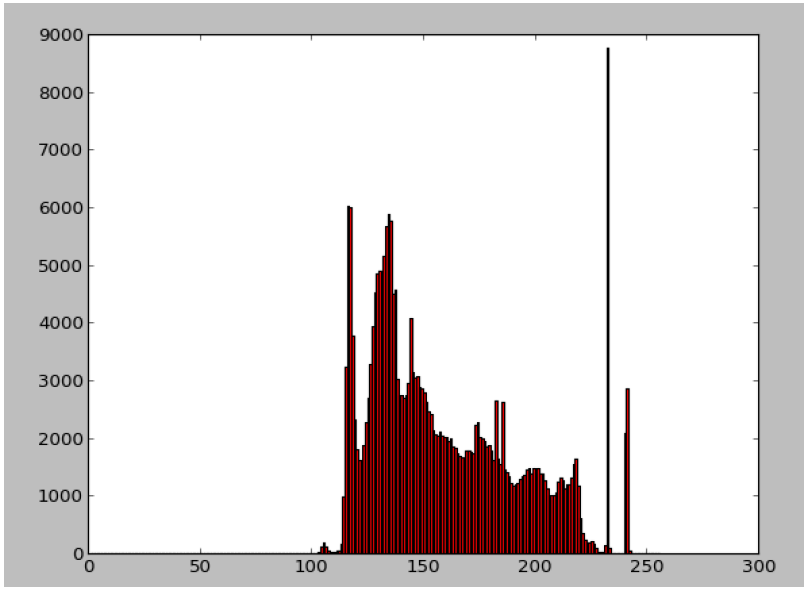


Image Histogram Equalization

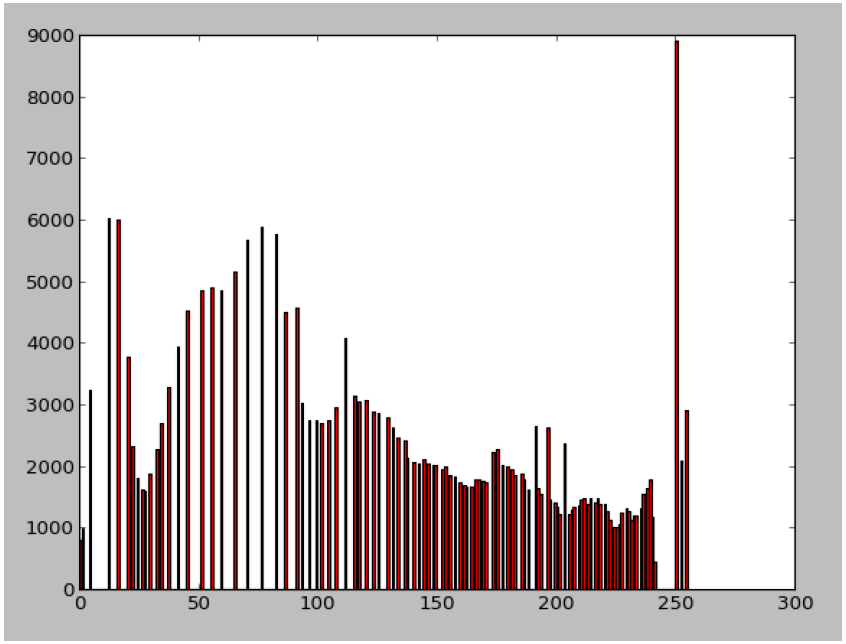
- We have a low-contrast image:



Limited image range

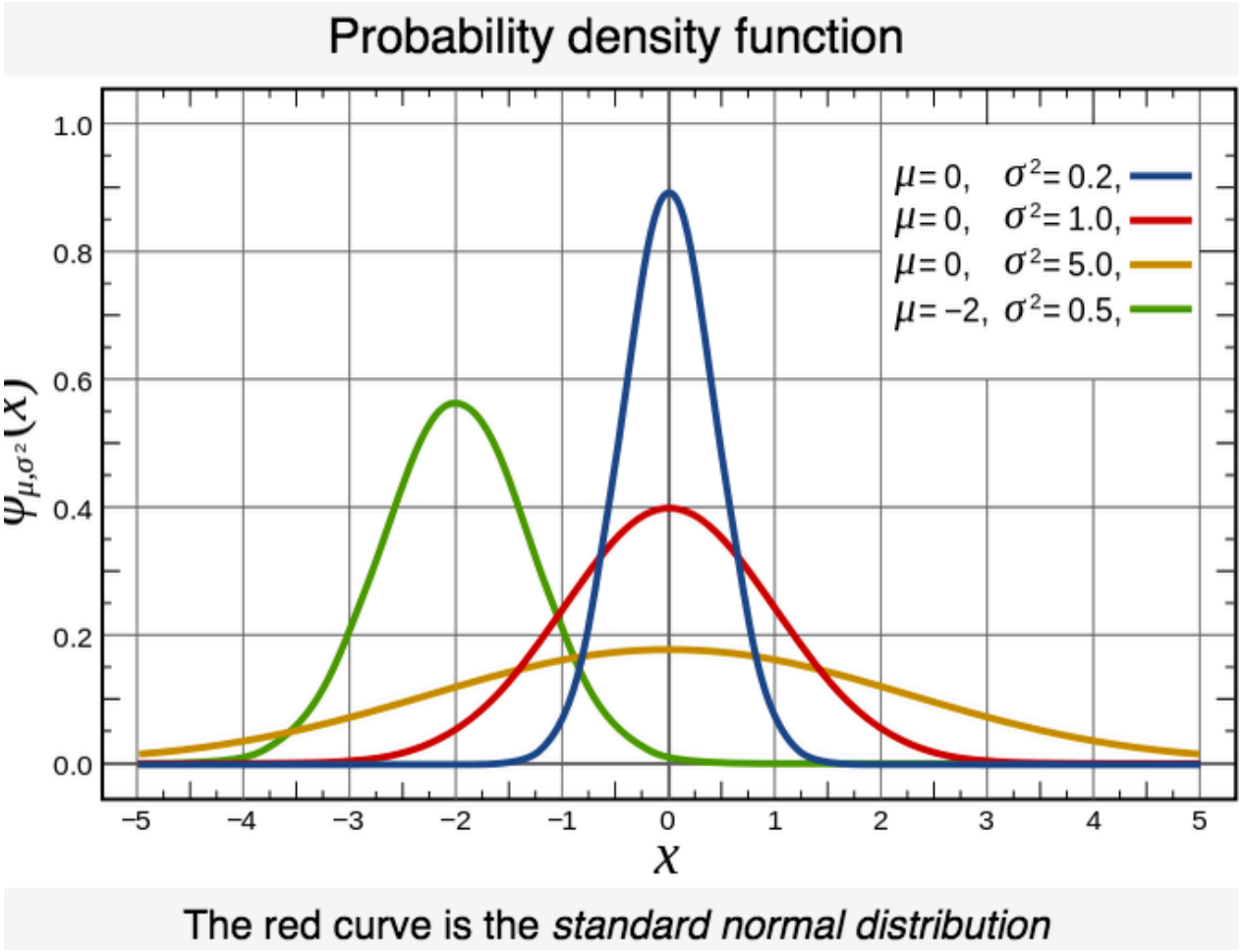
Image Histogram Equalization

- What we want is a histogram that covers the whole range of [0,255], but the shape must be preserved!



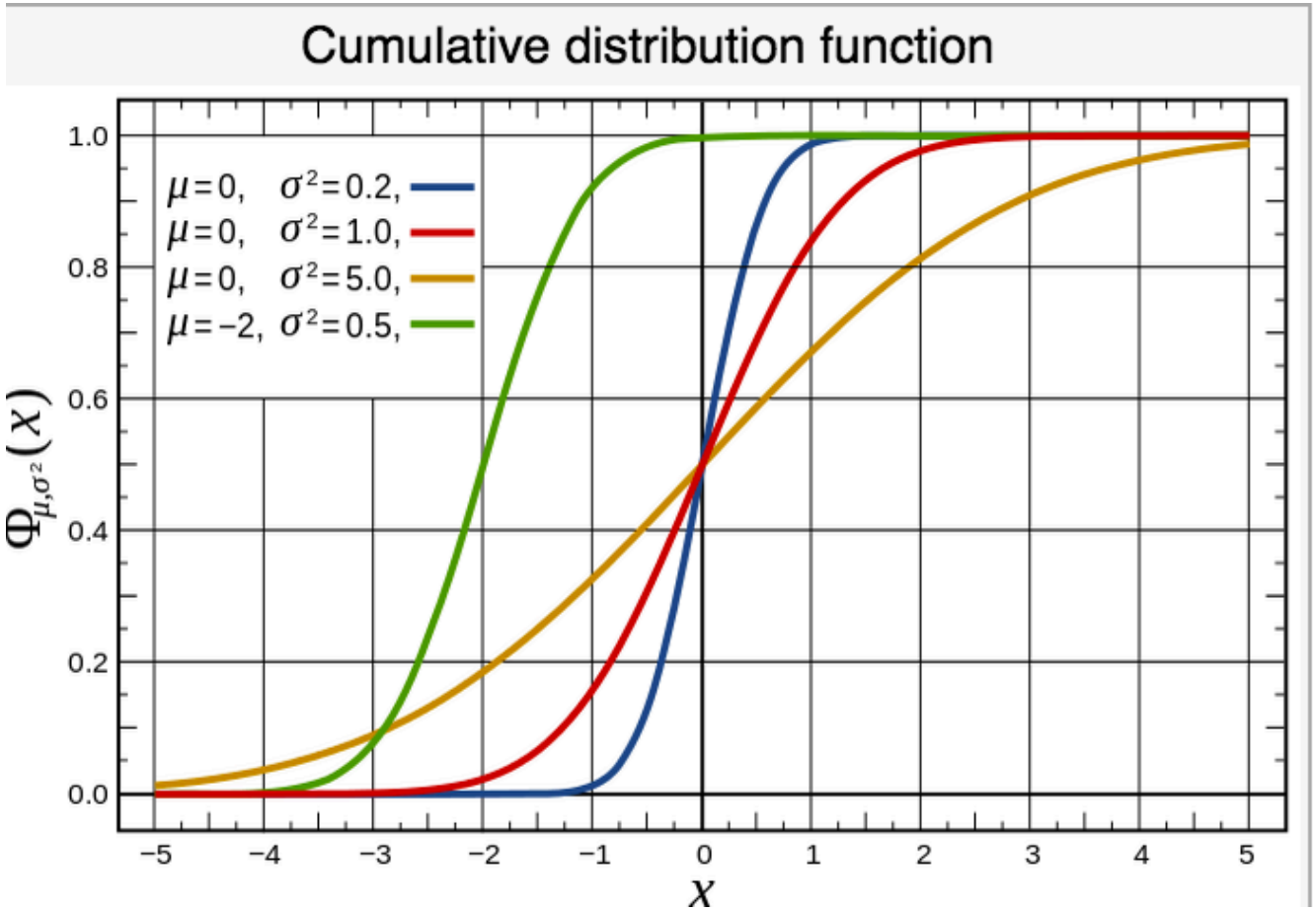
Ideal histogram

Mini detour of Probability Theory



PDF, density of a continuous random variable, is a function that describes the relative likelihood for this random variable to take on a given value.

Mini detour of Probability Theory

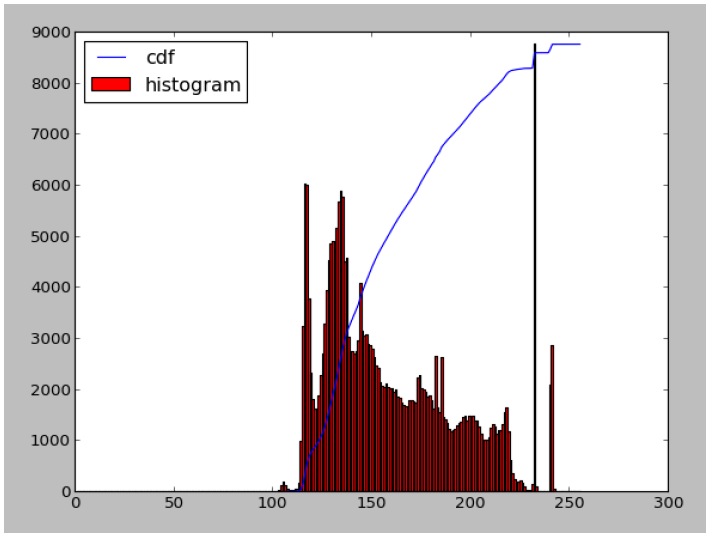


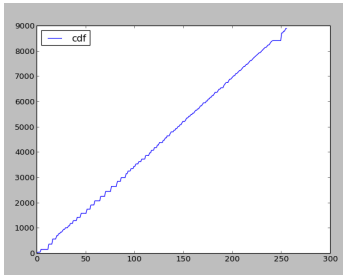
cumulative distribution function (CDF), describes the probability that a real-valued random variable X with a given probability distribution will be found to have a value less than or equal to x.

How does it work?

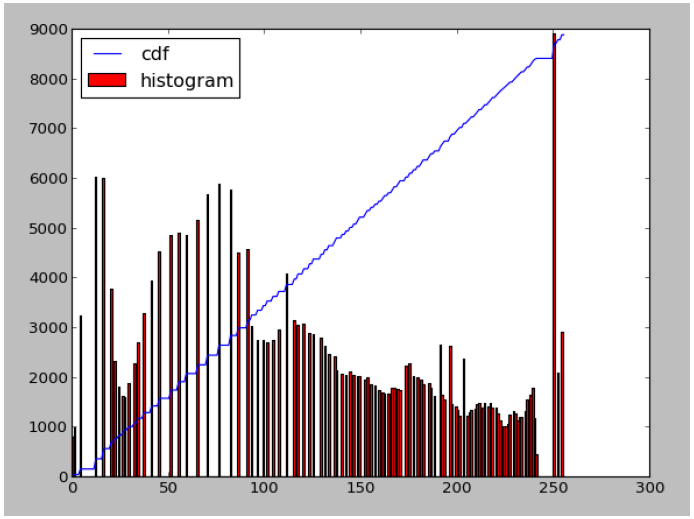
- Mapping one distribution to another distribution (a wider and more uniform of intensity values) so that the intensity values are spreading over the whole range
- The mapping should be the cumulative density function (CDF).

Stretching the CDF





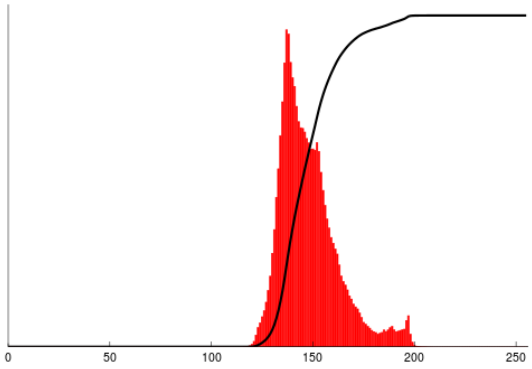
Transform



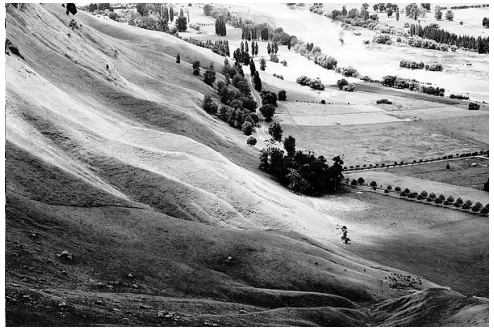
Before

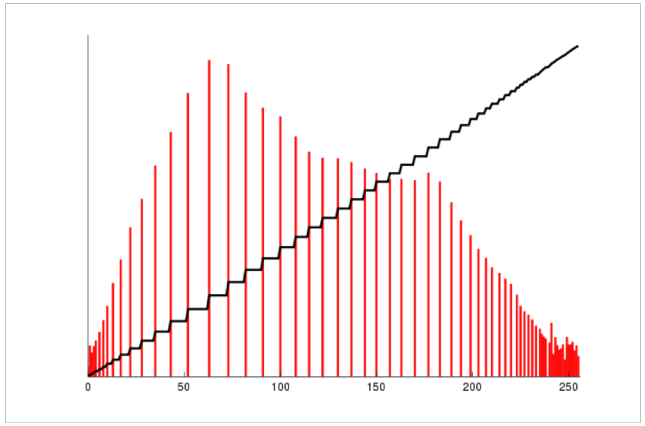
After

Image Histogram Equalization



Notice the “shape” of the histogram is preserved!





Same number of pixels In each bin!

Questions

1. Why don’t we do the equalization directly on the histograms? What will happen to the image if it has a perfectly flat histograms?
2. What is CDF? Why are there steps (zig-zags) on the CDF?
3. How to make sure the “shape” of the PDF is preserved in the transformation?

Let’s look at in more details

We start with a gray-scale jpeg image of 8 by 8

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

https://en.wikipedia.org/wiki/Histogram_equalization

Let’s look at in more details

The histograms for this image (8 by 8 pixels) is shown in the following table (intensity that has zero pixels are skipped:

Value	Count	Value	Count	Value	Count	Value	Count	Value	Count
52	1	64	2	72	1	85	2	113	1
55	3	65	3	73	2	87	1	122	1
58	2	66	2	75	1	88	1	126	1
59	3	67	1	76	1	90	1	144	1
60	1	68	5	77	1	94	1	154	1
61	4	69	3	78	1	104	2		
62	1	70	4	79	2	106	1		
63	2	71	2	83	1	109	1		

Let's look at in more details

The cumulative distribution function (CDF) is shown below

Value	cdf	scaled cdf
52	1	0
55	4	
58	6	
59	9	
60	10	
61	14	
62	15	
…	…	
154	64	255

Number of pixels

This cdf shows that the min value in the subimage is 52 and max is 154. The cdf of value 154 corresponding to the total number of pixels (64)

Let's look at in more details

How do we compute the normalized CDF?

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

Cdf(v): original cdf of pixel v
Cdfmin: minimum non-zero vale of the cdf
M×N: number of pixels, e.g. 64 (8x8)
L: 256

Quiz:

How do we compute normalized CDF of pixel 62?

Value	cdf	scaled cdf
52	1	0
55	4	
58	6	
59	9	
60	10	
61	14	
62	15	?
…	…	
154	64	255

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

Quiz:

How do we compute normalized CDF of pixel 62?

Value	cdf	scaled cdf
52	1	0
55	4	
58	6	
59	9	
60	10	
61	14	
62	15	?
…	…	
154	64	255

H(62) = round((15-1)/63*255)
= 57

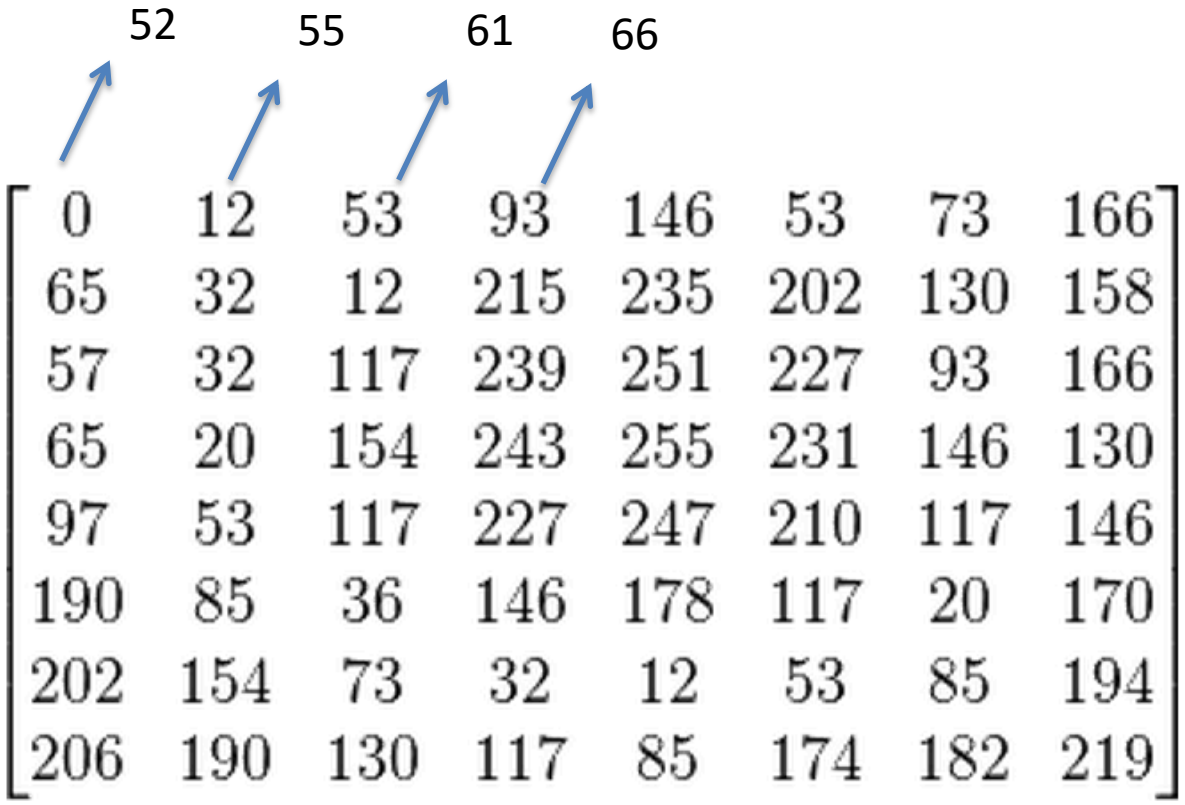
Let’s look at in more details

The cumulative distribution function (CDF) is shown below

Value	cdf	scaled cdf
52	1	0
55	4	12
58	6	22
59	9	32
60	10	36
61	14	53
62	15	57
…	…	…
154	64	255

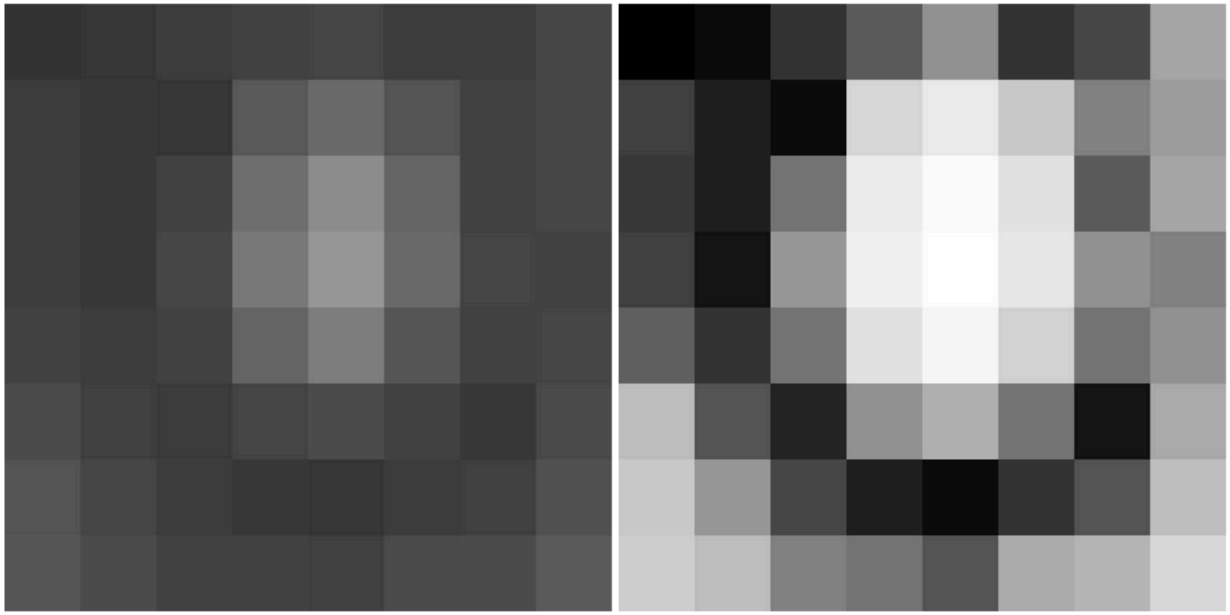
Let's look at in more details

Now we can directly map the scaled cdf back to pixel values using the look up table we derived above



Let’s look at in more details

Notice that the minimum value (52) is now 0 and the maximum value (154) is now 255.

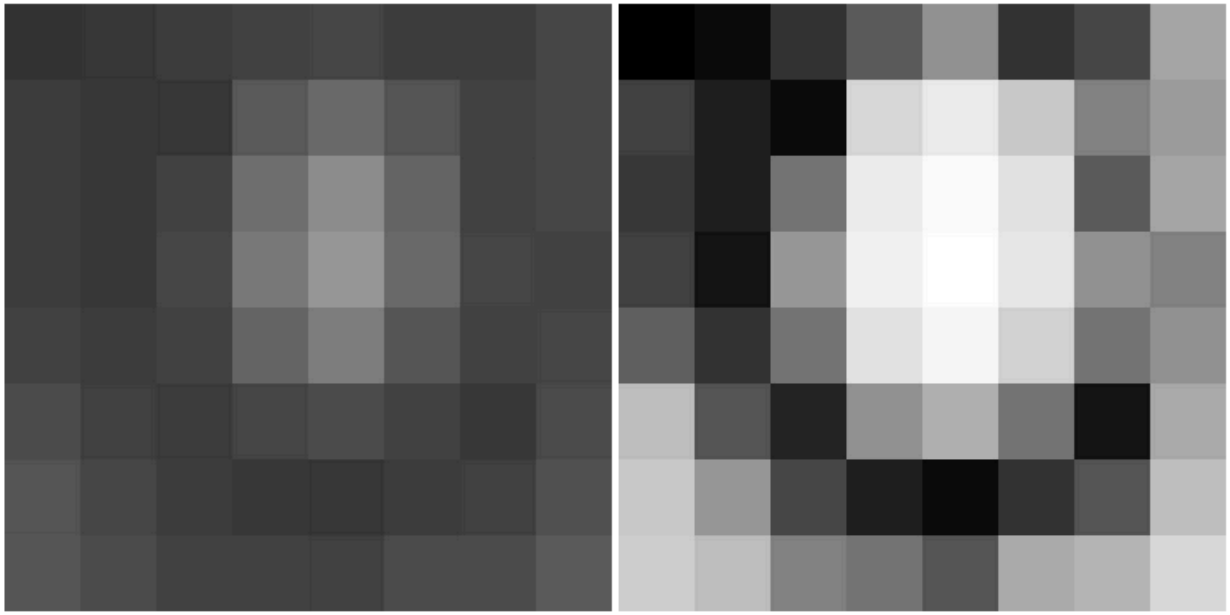


Original

Equalized

Let’s look at in more details

Notice that the minimum value (52) is now 0 and the maximum value (154) is now 255.



Original

Equalized

Homework 2: histogram equalization

Hint:

#Compute cdf in Python:

```
np.histogram(im.flatten(),nbr_bins,normed=True)

cdf = imhist.cumsum()

# normalize
cdf = 255 * cdf / cdf[-1]
# Using linear interpretation of cdf to find new pixels.

im2 = np.interp(im.flatten(),bins[:-1],cdf)
```

Next class

- Bordering effect
- Image Derivatives