CS 1674/2074: Linear Algebra

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

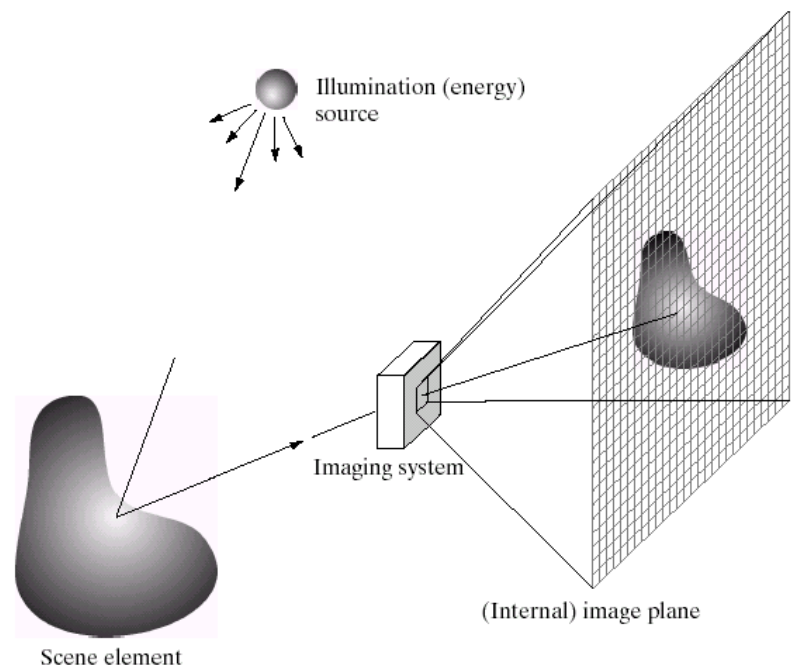
University of Pittsburgh

Linear Algebra Review

See http://cs229.stanford.edu/section/cs229-linalg.pdf for more

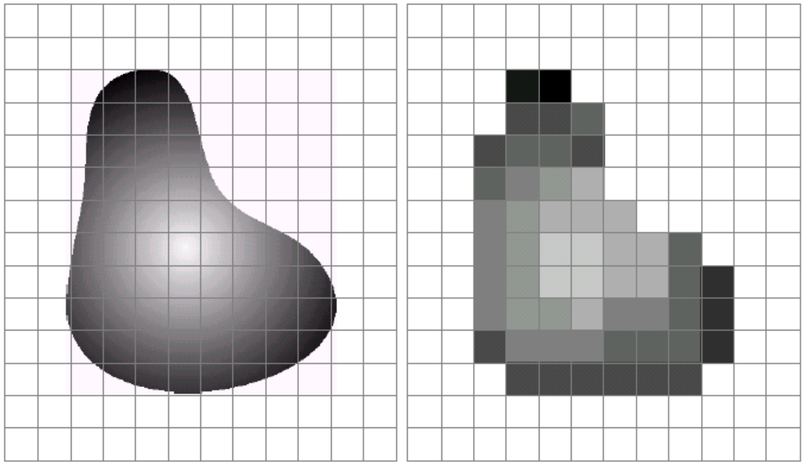
What are images? (in Python) – Image Formation

- Python treats images as matrices of numbers
- To proceed, let's talk very briefly about how images are formed



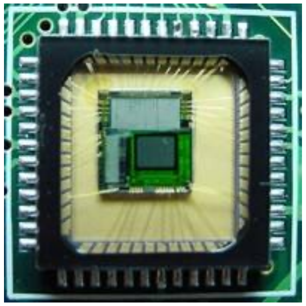
Slide credit: Derek Hoiem

Digital Images



a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.



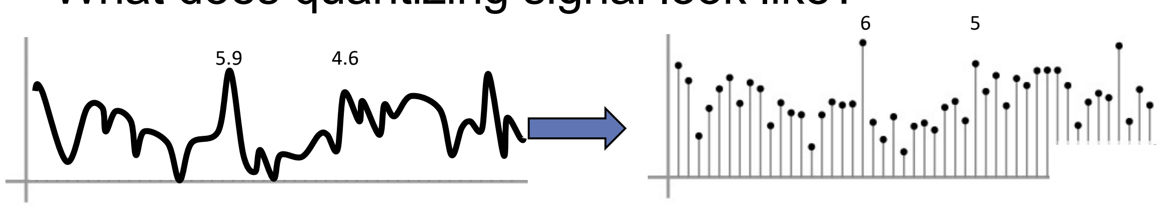
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)

Slide credit: Derek Hoiem, Steve Seitz

Digital Images

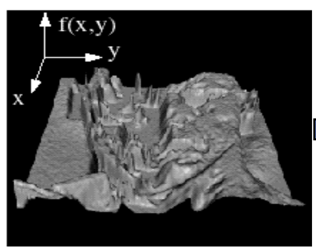
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- What does quantizing signal look like?

1D



- Image thus represented as a matrix of integer values.

2D



j →

i ↓

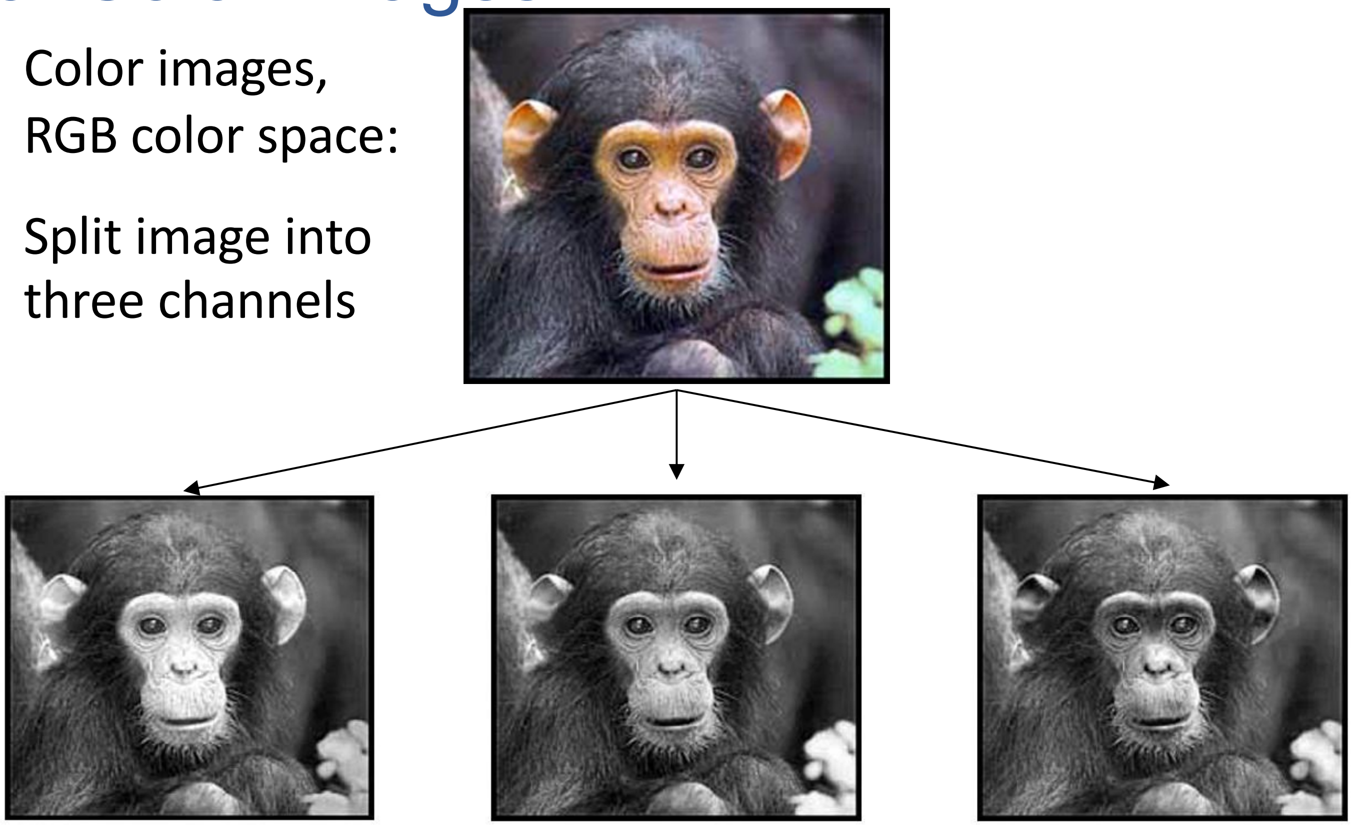
62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Adapted from S. Seitz

Digital Color Images

Color images,
RGB color space:

Split image into
three channels

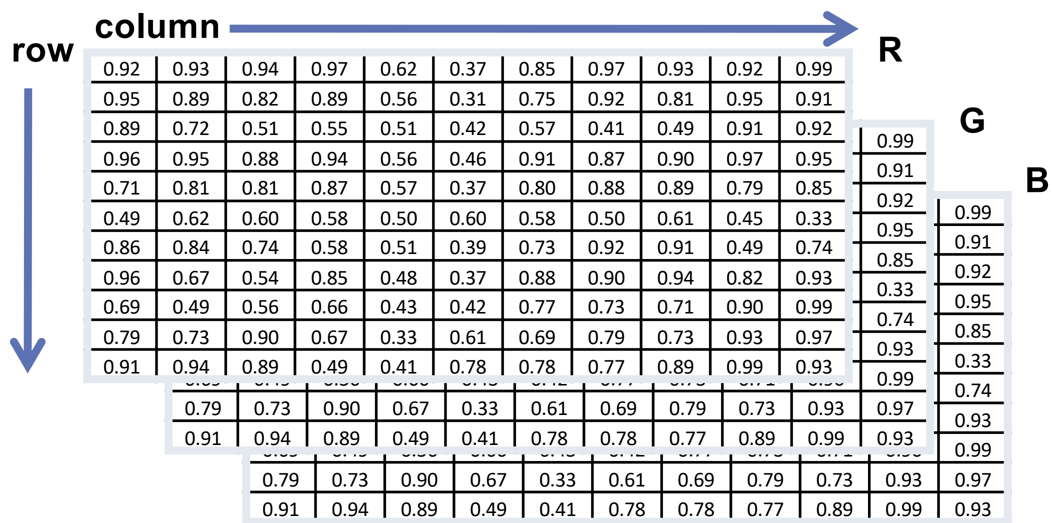


R G B

Adapted from Kristen Grauman

Images in Python

- Color images represented as a matrix with multiple channels (=1 if grayscale)
- Suppose we have a NxM RGB image called “im”
  - Im[0,0,0] = top-left pixel value in R-channel
  - im(y, x, b) = y pixels **down**, x pixels **to right** in the bth channel
  - im(N, M, 3) = bottom-right pixel in B-channel
- cv2.imread(filename) returns a uint8 image (values 0 to 255)



Adapted from Derek Hoiem

Vectors and Matrices

- Vectors and matrices are just collections of ordered numbers that represent something: movements in space, scaling factors, word counts, movie ratings, pixel brightness, etc.

- We'll define some common uses and standard operations on them.

Vector

- A column vector $\mathbf{v} \in \mathbb{R}^{n\times 1}$ where

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

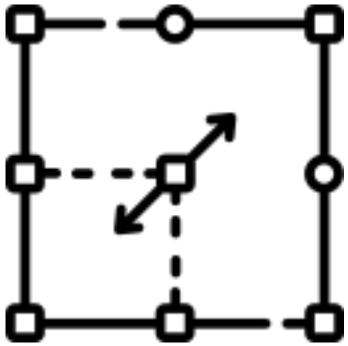
- A row vector $\mathbf{v}^T \in \mathbb{R}^{1\times n}$ where

$$\mathbf{v}^T = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

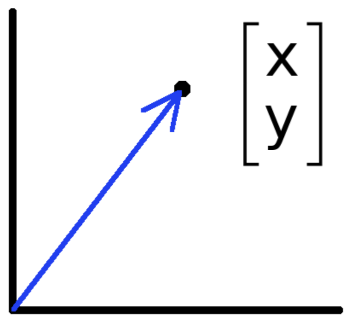
T denotes the transpose operation

Vector

- You'll want to keep track of the orientation of your vectors when programming in Python.
- You can transpose a vector *V* in Python by writing **V.T**.



Vectors have two main uses



- Vectors can represent an offset in 2D or 3D space
- Points are just vectors from the origin

- Data can also be treated as a vector
- Such vectors don't have a geometric interpretation, but calculations like "distance" still have value

Norms

- L1 norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

- L2 norm

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

- L^p norm (for real numbers $p \geq 1$)

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Distances

- L1 (Manhattan) distance

- L2 (Euclidean) distance

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Example: feature representation

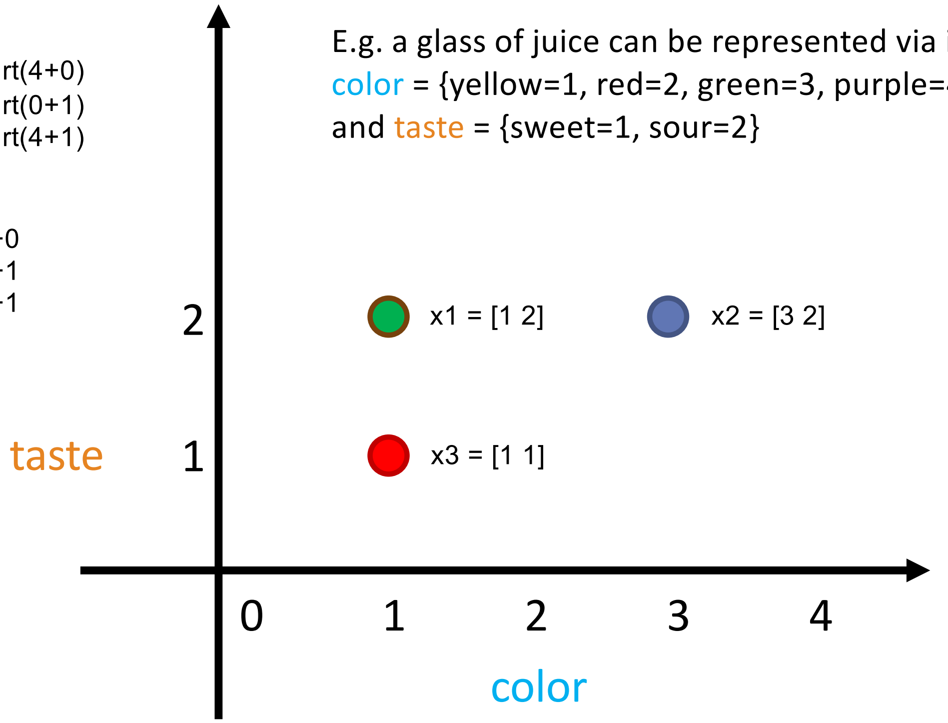
- A vector representing measurable characteristics of a data sample we have.
- E.g. a glass of juice can be represented via its color = {yellow=1, red=2, green=3, purple=4} and taste = {sweet=1, sour=2}
- A given glass *i* can be represented as a vector: $\boldsymbol{x}_i$ = [3 2] represents green , *sour* juice
- For *D* features, this defines a *D*-dimensional space where we can measure similarity between samples

Example: Feature representation

L2 distance:
d(x1, x2) = sqrt(4+0)
d(x1, x3) = sqrt(0+1)
d(x2, x3) = sqrt(4+1)

L1 distance:
d(x1, x2) = 2+0
d(x1, x3) = 0+1
d(x2, x3) = 2+1

E.g. a glass of juice can be represented via its color = {yellow=1, red=2, green=3, purple=4} and taste = {sweet=1, sour=2}



Matrix

- A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is an array of numbers with size $m \downarrow$ by $n \rightarrow$, i.e. m rows and n columns.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

- If $m = n$, we say that $\mathbf{A}$ is square.

Matrix Operations

- Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a+1 & b+2 \\ c+3 & d+4 \end{bmatrix}$$

  - Can only add a matrix with matching dimensions, or a scalar.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + 7 = \begin{bmatrix} a+7 & b+7 \\ c+7 & d+7 \end{bmatrix}$$

- Scaling

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times 3 = \begin{bmatrix} 3a & 3b \\ 3c & 3d \end{bmatrix}$$

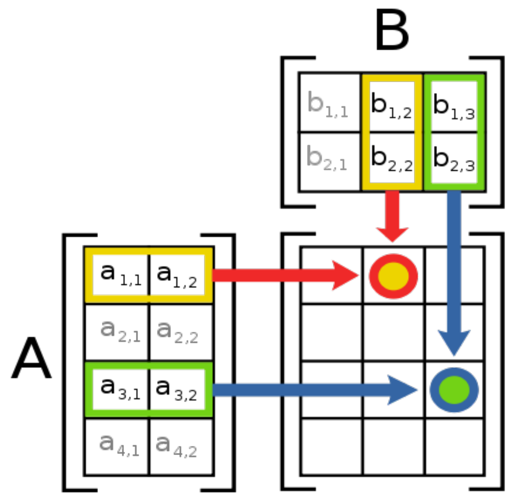
Matrix Multiplication

- Let X be an *a*x*b* matrix, Y be an *b*x*c* matrix
- Then Z = X*Y is an *a*x*c* matrix

- **Second dimension of first matrix, and first dimension of second matrix** have to be the **same**, for matrix multiplication to be possible

Matrix Multiplication

- The product AB is:



- Each entry in the result is (that row of A) dot product with (that column of B)

Matrix Multiplication

- Example:

A x B

$$\begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 \\ 4 & 6 \end{bmatrix} \quad \begin{bmatrix} \; & 14 \\ \; & \; \end{bmatrix}$$

$$0 \cdot 3 + 2 \cdot 7 = 14$$

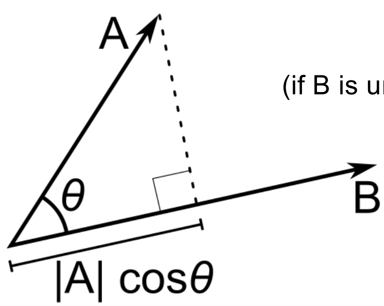
– Each entry of the matrix product is made by taking the dot product of the corresponding row in the left matrix, with the corresponding column in the right one.

Inner Product

- Multiply corresponding entries of two vectors and add up the result

$$\mathbf{x}^T\mathbf{y} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i \qquad \text{(scalar)}$$

- x·y is also |x||y|Cos(*angle between x and y*)
- If B is a unit vector, then A·B gives the length of A which lies in the direction of B (projection)



(if B is unit-length hence norm is 1)

Different Types of Product

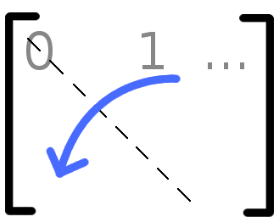
- $\boldsymbol{x}, \boldsymbol{y}$ = column vectors (nx1)
- $\boldsymbol{X}, \boldsymbol{Y}$ = matrices (mxn)
- x, y = scalars (1x1)

- $\boldsymbol{x}^T \boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y}$ = inner product (1xn x nx1 = scalar)
- $\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{x}\, \boldsymbol{y}^T$ = outer product (nx1 x 1xn = matrix)

- $\boldsymbol{X} * \boldsymbol{Y}$ = matrix product
- $\boldsymbol{X}$.* $\boldsymbol{Y}$ = element-wise product

Matrix Operations

- Transpose – flip matrix, so row 1 becomes column 1



$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$$

- A useful identity:

$$(ABC)^T = C^T B^T A^T$$

Matrix Operation Properties

- Matrix addition is commutative and associative
  - A + B = B + A
  - A + (B + C) = (A + B) + C

- Matrix multiplication is associative and distributive but *not* commutative
  - A(B*C) = (A*B)C
  - A(B + C) = A*B + A*C
  - A*B != B*A

Special Matrices

- Identity matrix **I**
  - Square matrix, 1's along diagonal, 0's elsewhere
  - **I** · [another matrix] = [that matrix]

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

- Diagonal matrix
  - Square matrix with numbers along diagonal, 0's elsewhere
  - A diagonal · [another matrix] scales the rows of that matrix

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 2.5 \end{bmatrix}$$

Special Matrices

- Symmetric matrix

$$\mathbf{A}^T = \mathbf{A} \qquad \begin{bmatrix} 1 & 2 & 5 \\ 2 & 1 & 7 \\ 5 & 7 & 1 \end{bmatrix}$$

Python tutorial

Setup Conda Environment:
https://canvas.pitt.edu/courses/288692/pages/conda-environment?module_item_id=4926894

Python tutorial

https://github.com/nineil-pitt/cs1674_2074_fall23/blob/main/module_1_python/tutorial.ipynb