COMPUTER VISION
LECTURE 3-5 – LINEAR ALGEBRA REVIEW

Prof. Dr. Francesco Maurelli

2019-09-10

2019-09-17

JACOBS UNIVERSITY

OUTLINE

JACOBS UNIVERSITY

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

Chapter 1

Brief Summary

Academic calendar 2019/20 plus September

JACOBS UNIVERSITY

2019				2020								
September	October	November	December	January	February	March	April	May	June	July	August	September
1 Su	1 Tu	1 Fr	1 Su	1 We	1 Sa	1 Su	1 We	1 Fr	1 Mo Pentecost	1 We	1 Sa	1 Tu classes begin
2 Mo classes begin	2 We	2 Sa	2 Mo	2 Th	2 Su	2 Mo	2 Th	2 Sa	2 Tu	2 Th	2 Su	2 We
3 Tu	3 Th	3 Su	3 Tu	3 Fr	3 Mo classes begin	3 Tu	3 Fr	3 Su	3 We grades due: graduation	3 Fr	3 Mo	3 Th
4 We	4 Fr	4 Mo	4 We	4 Sa	4 Tu	4 We	4 Sa	4 Mo	4 Th	4 Sa	4 Tu	4 Fr
5 Th	5 Sa	5 Tu	5 Th	5 Su	5 We	5 Th	5 Su	5 Tu	5 Fr	5 Su	5 We	5 Sa
6 Fr	6 Su	6 We	6 Fr classes end	6 Mo	6 Th	6 Fr	6 Mo	6 We	6 Sa	6 Mo	6 Th	6 Su
7 Sa	7 Mo	7 Th	7 Sa reading day	7 Tu	7 Fr	7 Sa	7 Tu SPRING	7 Th	7 Su	7 Tu	7 Fr	7 Mo
8 Su	8 Tu	8 Fr	8 Su reading day	8 We	8 Sa	8 Su	8 We BREAK	8 Fr	8 Mo	8 We	8 Sa	8 Tu
9 Mo	9 We	9 Sa	9 Mo exam	9 Th	9 Su	9 Mo	9 Th	9 Sa	9 Tu	9 Th	9 Su	9 We
10 Tu	10 Th	10 Su	10 Tu period	10 Fr break ends	10 Mo	10 Tu	10 Fr Good Friday	10 Su	10 We	10 Fr	10 Mo	10 Th
11 We	11 Fr	11 Mo	11 We	11 Sa	11 Tu	11 We	11 Sa	11 Mo	11 Th	11 Sa	11 Tu	11 Fr
12 Th	12 Sa	12 Tu	12 Th	12 Su	12 We	12 Th	12 Su	12 Tu	12 Fr graduation	12 Su	12 We	12 Sa
13 Fr	13 Su	13 We	13 Fr	13 Mo grades due/ intersession	13 Th	13 Fr	13 Mo Easter Monday	13 We	13 Sa	13 Mo	13 Th	13 Su
14 Sa	14 Mo	14 Th	14 Sa	14 Tu begins	14 Fr grades due make-ups	14 Sa	14 Tu	14 Th	14 Su	14 Tu	14 Fr diplomas & transcripts	14 Mo grades due make-ups
15 Su	15 Tu	15 Fr	15 Su	15 We	15 Sa	15 Su	15 We	15 Fr classes end	15 Mo	15 We	15 Sa due	15 Tu drop/add
16 Mo drop/ add	16 We	16 Sa	16 Mo	16 Th	16 Su	16 Mo	16 Th	16 Sa reading day	16 Tu	16 Th	16 Su	16 We
17 Tu	17 Th	17 Su	17 Tu	17 Fr	17 Mo drop / add	17 Tu	17 Fr	17 Su reading day	17 We	17 Fr	17 Mo	17 Th
18 We	18 Fr	18 Mo	18 We	18 Sa	18 Tu	18 We	18 Sa	18 Mo exam	18 Th	18 Sa	18 Tu	18 Fr
19 Th	19 Sa	19 Tu	19 Th	19 Su	19 We	19 Th	19 Su	19 Tu period	19 Fr	19 Su	19 We	19 Sa
20 Fr	20 Su	20 We	20 Fr	20 Mo	20 Th	20 Fr	20 Mo	20 We	20 Sa	20 Mo	20 Th	20 Su
21 Sa	21 Mo	21 Th	21 Sa	21 Tu	21 Fr	21 Sa	21 Tu	21 Th Christi Himmelfahrt	21 Su	21 Tu	21 Fr	21 Mo
22 Su	22 Tu	22 Fr	22 Su	22 We	22 Sa	22 Su	22 We	22 Fr	22 Mo	22 We	22 Sa make-up period	22 Tu
23 Mo	23 We	23 Sa	23 Mo break begins	23 Th make-up period	23 Su	23 Mo	23 Th	23 Sa	23 Tu	23 Th	23 Su	23 We
24 Tu	24 Th	24 Su	24 Tu	24 Fr	24 Mo	24 Tu	24 Fr	24 Su	24 We remaining grades due	24 Fr	24 Mo	24 Th
25 We	25 Fr	25 Mo	25 We Christmas Day	25 Sa	25 Tu	25 We	25 Sa	25 Mo	25 Th	25 Sa	25 Tu	25 Fr
26 Th	26 Sa	26 Tu	26 Th Boxing Day	26 Su	26 We	26 Th	26 Su	26 Tu	26 Fr	26 Su	26 We O-Week begins	26 Sa
27 Fr	27 Su	27 We	27 Fr	27 Mo	27 Th	27 Fr	27 Mo	27 We	27 Sa	27 Mo	27 Th	27 Su
28 Sa	28 Mo	28 Th	28 Sa	28 Tu	28 Fr	28 Sa	28 Tu	28 Th	28 Su	28 Tu	28 Fr	28 Mo
29 Su	29 Tu	29 Fr	29 Su	29 We	29 Sa	29 Su	29 We	29 Fr	29 Mo	29 We	29 Sa	29 Tu
30 Mo	30 We	30 Sa	30 Mo	30 Th		30 Mo	30 Th	30 Sa	30 Tu	30 Th	30 Su	30 We
	31 Th		31 Tu	31 Fr		31 Tu		31 Su summer recess		31 Fr	31 Mo O-Week ends	

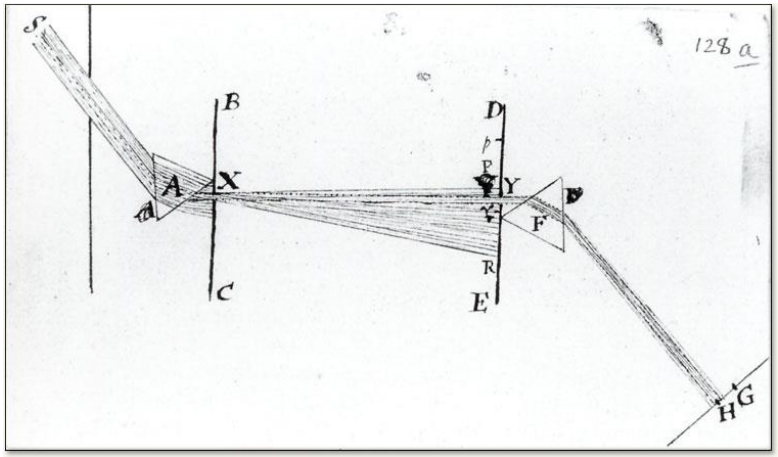
THE GOAL OF COMPUTER VISION

Bridging the gap between pixels and meaning

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

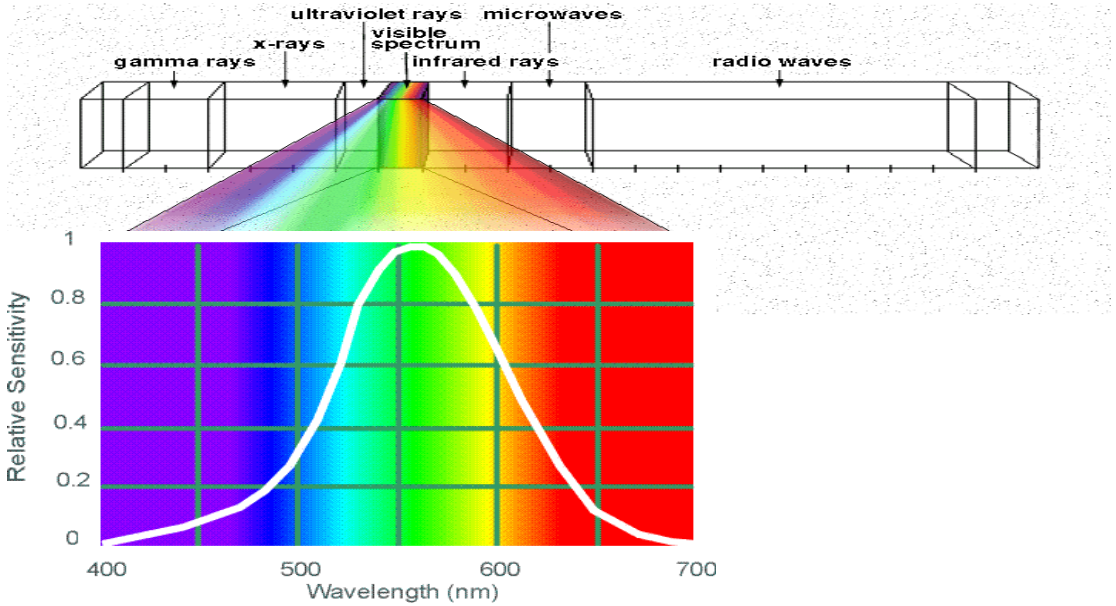
COLOR AND LIGHT

White light: composed of almost equal energy in all wavelengths of the visible spectrum



Newton 1665

ELECTROMAGNETIC SPECTRUM



Human Luminance Sensitivity Function

TWO TYPES OF LIGHT-SENSITIVE RECEPTORS

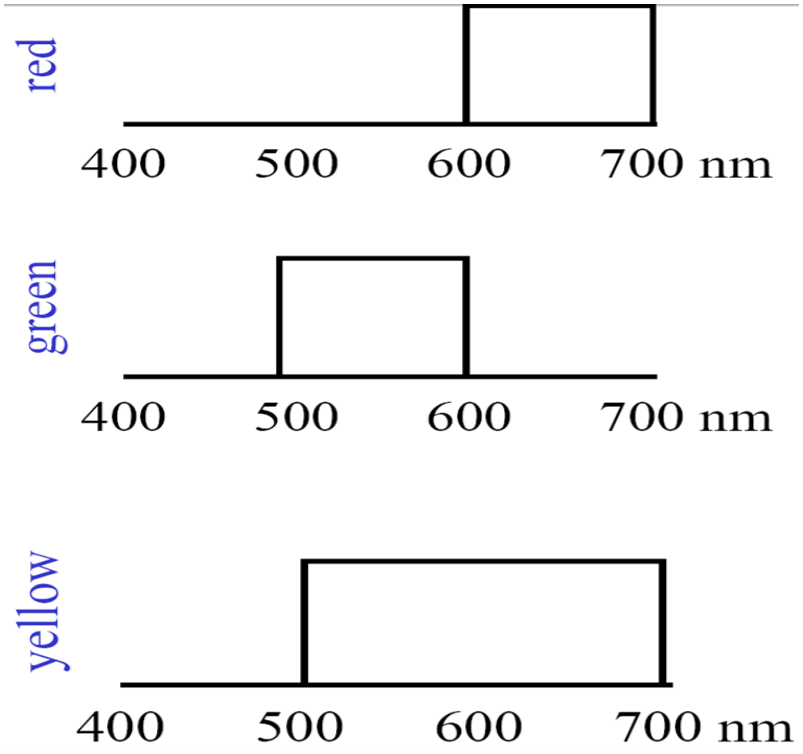
Cones

- cone-shaped
- less sensitive
- operate in high light
- color vision

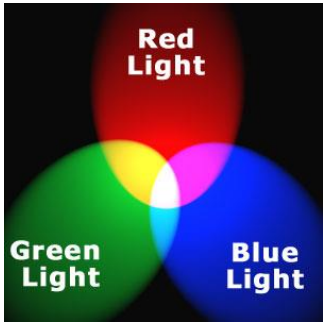
Rods

- rod-shaped
- highly sensitive
- operate at night
- gray-scale vision

ADDITIVE COLOR MIXING



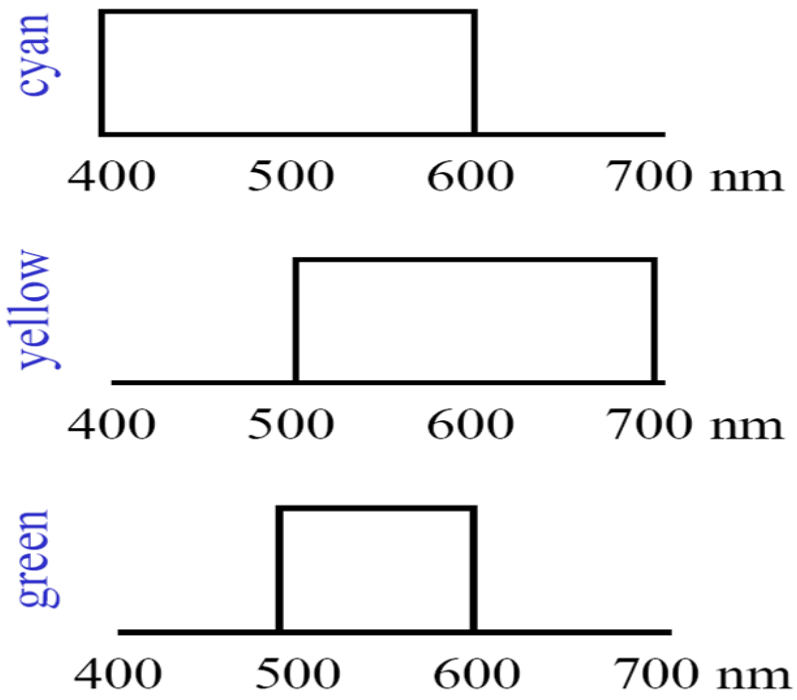
Colors combine by *adding* color spectra



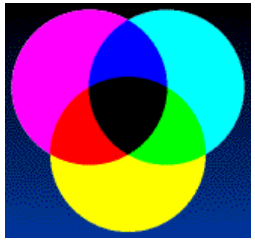
Light *adds* to existing black.

Source: W. Freeman

SUBTRACTIVE COLOR MIXING



Colors combine by *multiplying* color spectra.



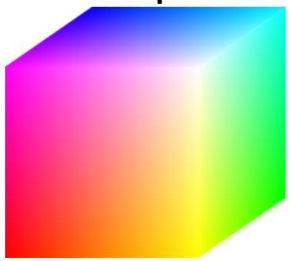
Pigments *remove* color from incident light (white).

Source: W. Freeman

RGB SPACE

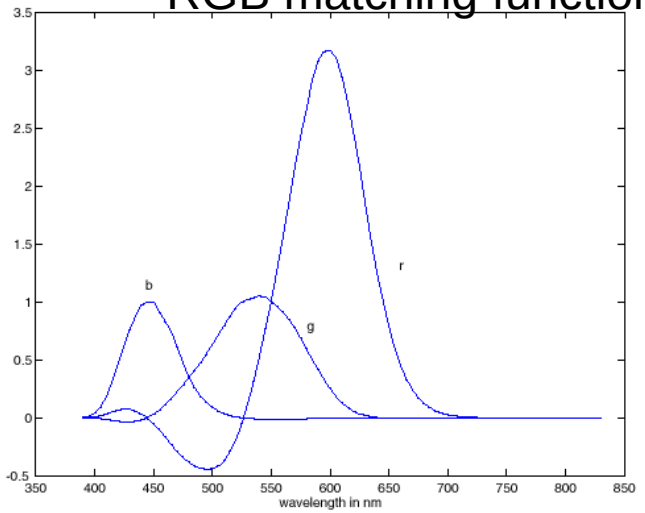
- Primaries are monochromatic lights (for monitors, they correspond to the three types of phosphors)
- *Subtractive matching* required for some wavelengths

RGB primaries

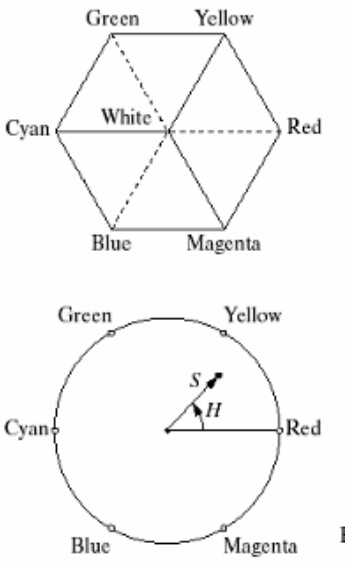


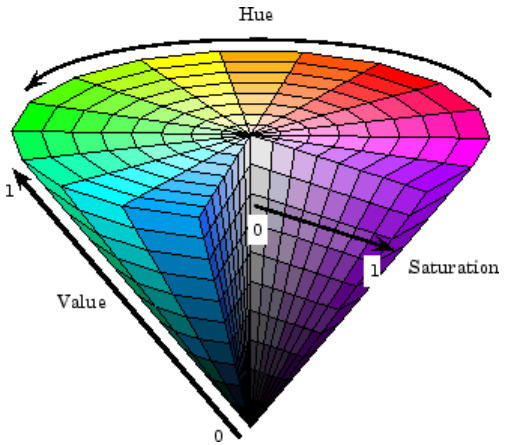
$p_1 = 645.2$ nm
$p_2 = 525.3$ nm
$p_3 = 444.4$ nm

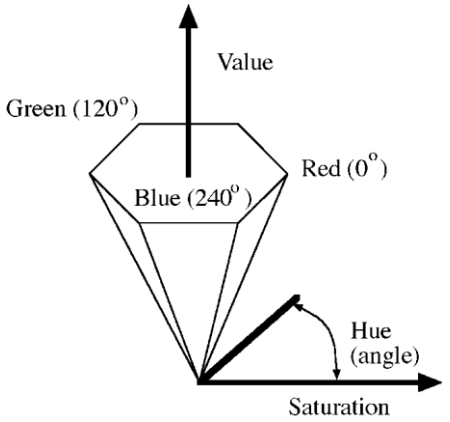
RGB matching functions



NONLINEAR COLOR SPACES: HSV

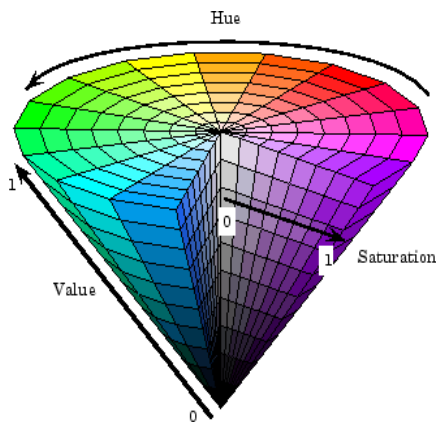


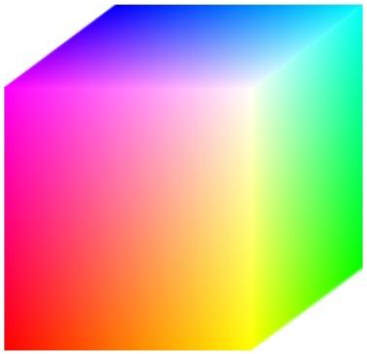




Perceptually meaningful dimensions:
Hue, Saturation, Value (Intensity)

CONVERSION





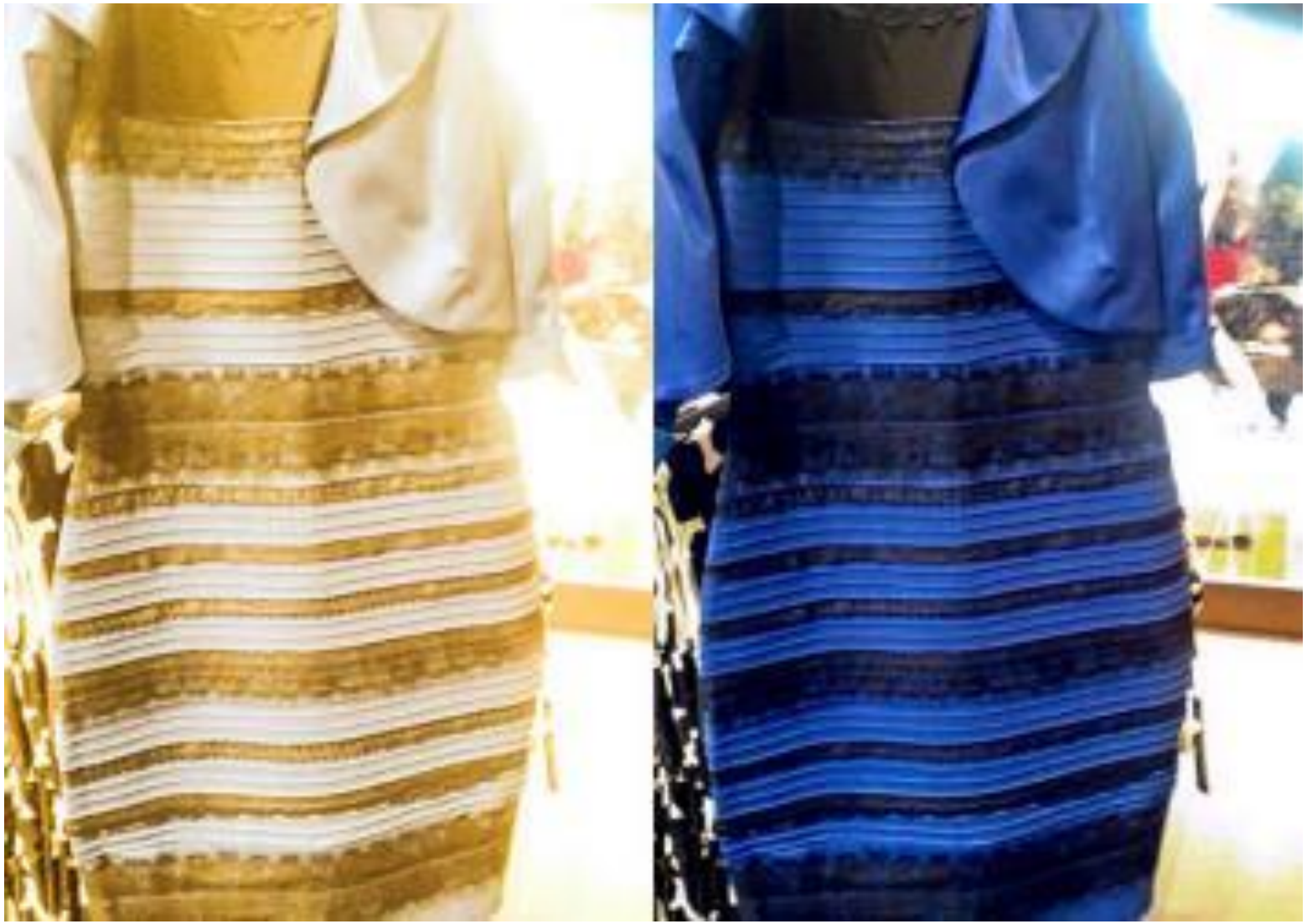
$$R' = R/255$$
$$G' = G/255$$
$$B' = B/255$$
$$Cmax = \max(R', G', B')$$
$$Cmin = \min(R', G', B')$$
$$\Delta = Cmax - Cmin$$

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left(\frac{G'-B'}{\Delta} mod 6\right) & , C_{max} = R' \\ 60^\circ \times \left(\frac{B'-R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^\circ \times \left(\frac{R'-G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

$$V = Cmax$$

COLOR PERCEPTION



OUTLINE

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

Vectors and matrices are just **collections of ordered numbers** that represent something: movements in space, scaling factors, pixel brightness, etc.

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

VECTOR

JACOBS UNIVERSITY

- 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

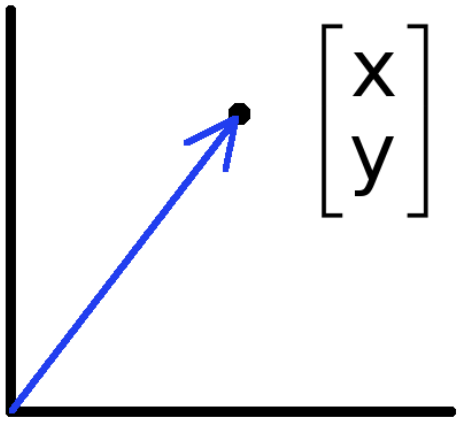
JACOBS UNIVERSITY

- We will only consider column vector

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

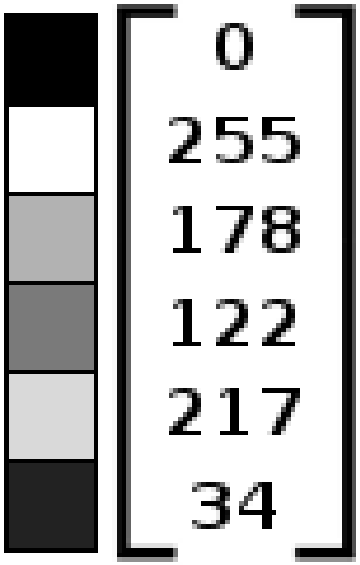
- Keep track of the orientation of your vectors when programming in MATLAB/python
- You can transpose a vector *V* in MATLAB by writing **V'**
- You can transpose a vector *V* in python by writing **V.t**.
- in class materials, we will always use V^T to indicate transpose

VECTORS HAVE TWO MAIN USES



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

- Data (pixels, gradients at an image keypoint, etc) can also be treated as a vector
- Such vectors don't have a geometric interpretation, but calculations like "distance" can still have value



JACOBS UNIVERSITY

- A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is an array of numbers with size by *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.

=

$$\begin{bmatrix} 193 & 180 & 210 & 112 & 125 \\ 189 & 8 & 177 & 97 & 114 \\ 100 & 71 & 81 & 195 & 165 \\ 167 & 12 & 242 & 203 & 181 \\ 44 & 25 & 9 & 48 & 192 \end{bmatrix}$$

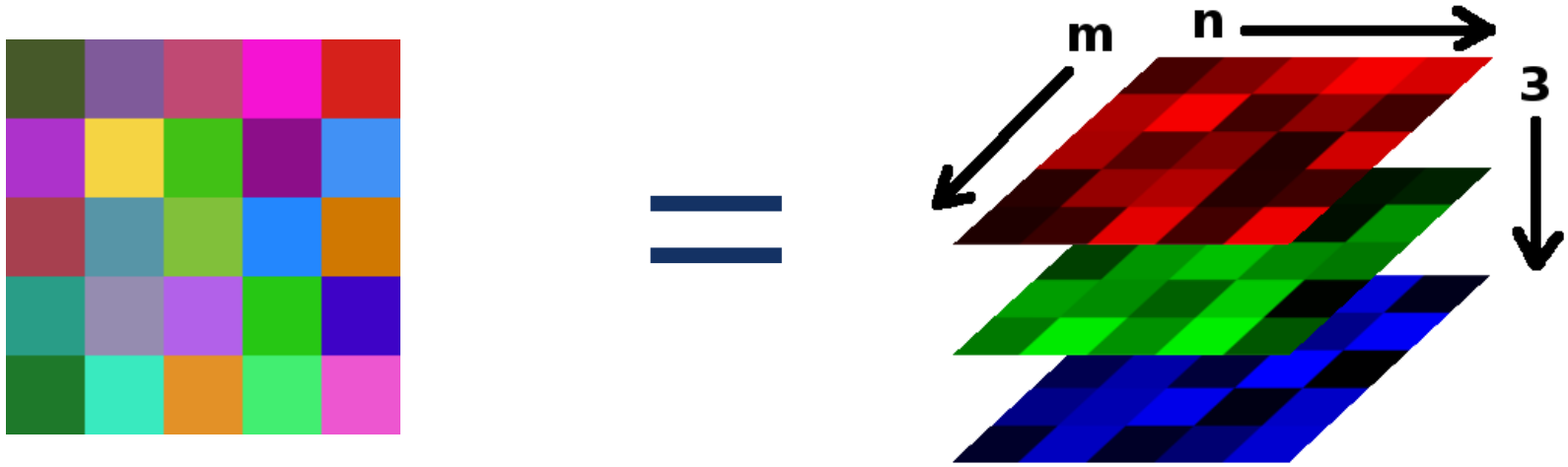
- MATLAB/Python represents an image as a matrix of pixel brightnesses
- Note that the upper left corner is

[y,x] = (1,1) for MATLAB

[y,x] = (0,0) for python

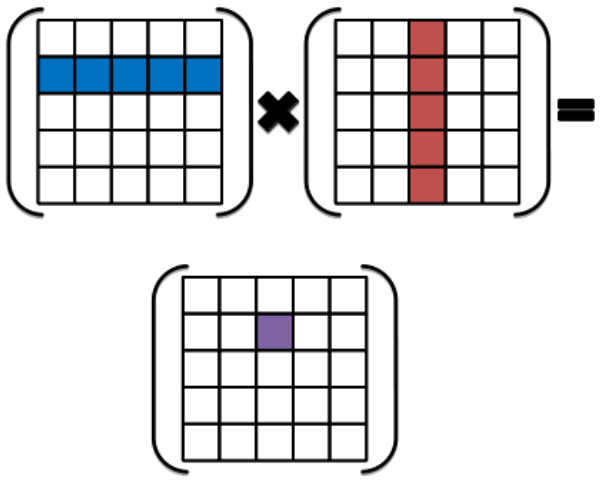
COLOR IMAGES

- Grayscale images have one number per pixel, and are stored as an m × n matrix.
- Color images have 3 numbers per pixel – red, green, and blue brightnesses (RGB)
- Stored as an m × n × 3 matrix



BASIC MATRIX OPERATIONS

- We will discuss:
  - Addition
  - Scaling
  - Dot product
  - Multiplication
  - Transpose
  - Inverse / pseudoinverse
  - Determinant / trace



JACOBS UNIVERSITY

- **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**

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

JACOBS UNIVERSITY

- **Vector Norm**

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

$$f : \mathbb{R}^n \to \mathbb{R}$$

More formally, a norm is any function that satisfies 4 properties:

- **Non-negativity:** For all $x \in \mathbb{R}^n,\ f(x) \geq 0$
- **Definiteness:** f(x) = 0 if and only if x = 0.
- **Homogeneity:** For all $x \in \mathbb{R}^n,\ t \in \mathbb{R},\ f(tx) = |t| f(x)$
- **Triangle inequality:** For all $x, y \in \mathbb{R}^n,\ f(x + y) \leq f(x) + f(y)$

JACOBS UNIVERSITY

- **Example Norms**

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \qquad \|x\|_\infty = \max_i |x_i|$$

- General ℓ_p norms:

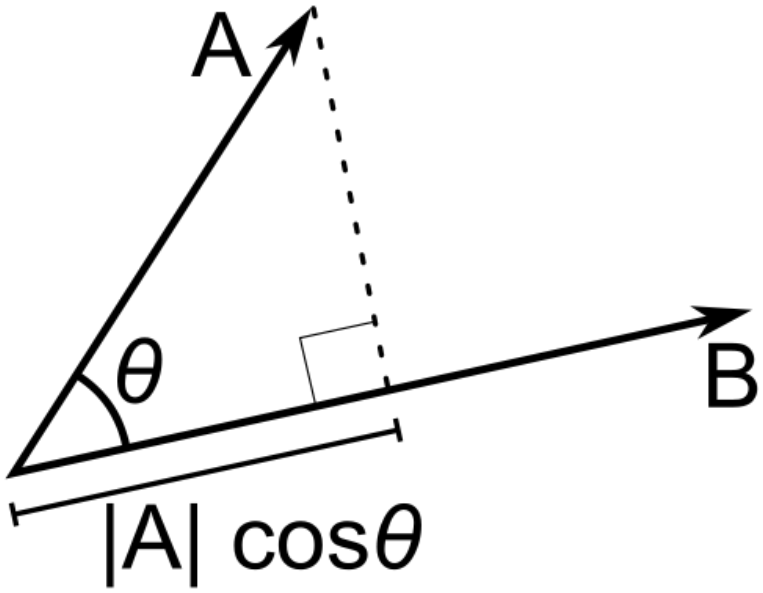
$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

JACOBS UNIVERSITY

- Inner product (dot product) of vectors
  - Multiply corresponding entries of two vectors and add up the result
  - x·y is also |x| |y| cos (*the angle between x and y*)

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

- Inner product (dot product) of vectors
  - If B is a unit vector, then A·B gives the length of A which lies in the direction of B



JACOBS UNIVERSITY

- The product of two matrices

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

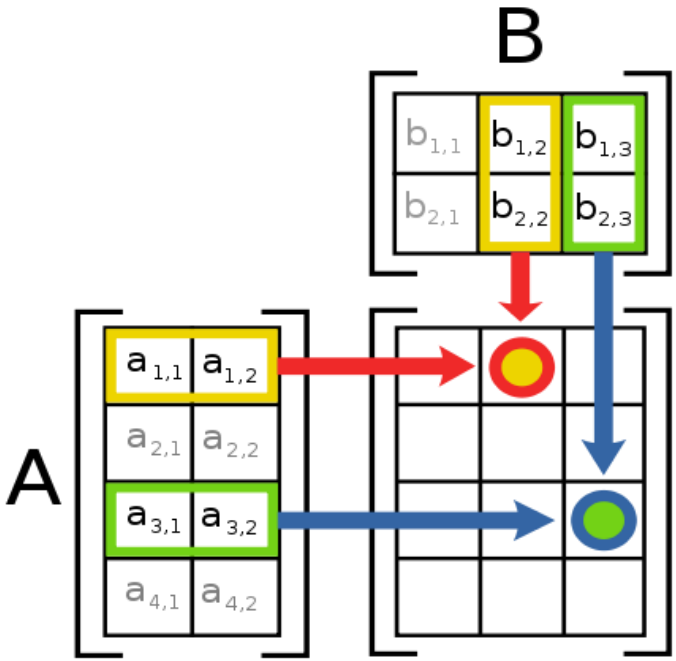
$$B \in \mathbb{R}^{n \times p}$$

$$C = AB \in \mathbb{R}^{m \times p}$$

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

$$C = AB = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_p \\ a_2^T b_1 & a_2^T b_2 & \cdots & a_2^T b_p \\ \vdots & \vdots & \ddots & \vdots \\ a_m^T b_1 & a_m^T b_2 & \cdots & a_m^T b_p \end{bmatrix}$$

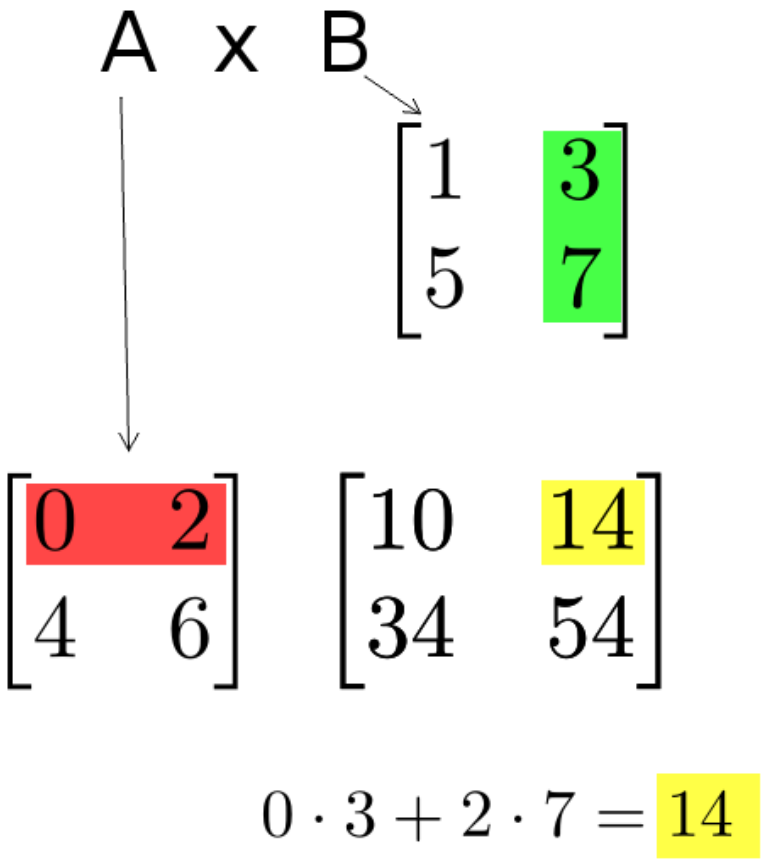
- Multiplication
- The product AB is:



- Each entry in the result is (that row of A) dot product with (that column of B)
- Many uses, which will be covered later

MATRIX OPERATIONS

- Multiplication example:



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.

JACOBS UNIVERSITY

- The product of two matrices - properties

Matrix multiplication is associative: $(AB)C = A(BC)$.

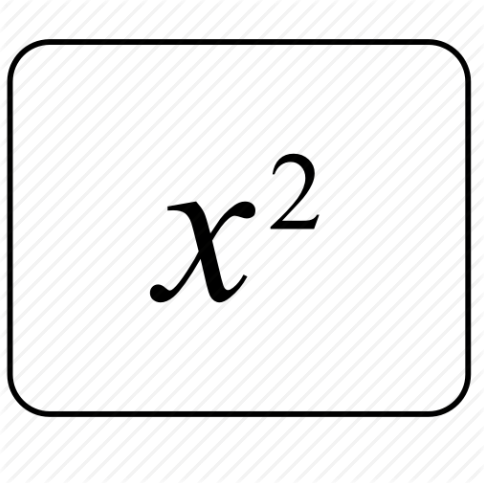
Matrix multiplication is distributive: $A(B + C) = AB + AC$.

Matrix multiplication is, in general, *not* commutative; that is, it can be the case that $AB \neq BA$. (For example, if $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times q}$, the matrix product BA does not even exist if m and q are not equal!)

JACOBS UNIVERSITY

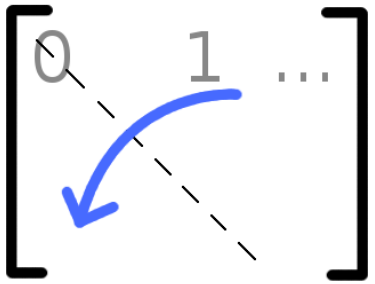
Powers

- By convention, we can refer to the matrix product AA as A^2, and AAA as A^3, etc.
- Obviously only square matrices can be multiplied that way



JACOBS UNIVERSITY

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



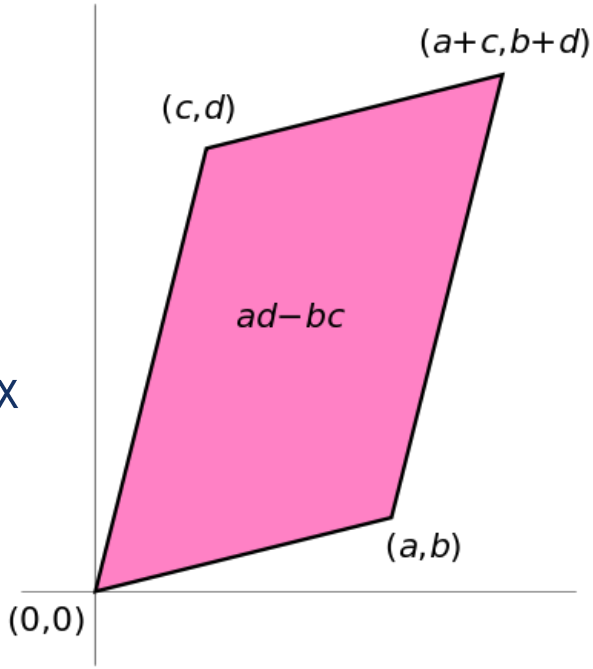
$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$$

- A useful identity:

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

JACOBS UNIVERSITY

- Determinant



- $\det(\mathbf{A})$ returns a scalar
- Represents area (or volume) of the parallelogram described by the vectors in the rows of the matrix
- $$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \det(\mathbf{A}) = ad - bc$$
- Properties:

$$\det(\mathbf{AB}) = \det(\mathbf{BA})$$

$$\det(\mathbf{A}^{-1}) = \frac{1}{\det(\mathbf{A})}$$

$$\det(\mathbf{A}^T) = \det(\mathbf{A})$$

$$\det(\mathbf{A}) = 0 \Leftrightarrow \mathbf{A} \text{ is singular}$$

JACOBS UNIVERSITY

- **Trace**

$$\mathrm{tr}(\mathbf{A}) = \text{sum of diagonal elements} \qquad \mathbf{tr}(\begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix}) = 1 + 7 = 8$$

  - Invariant to a lot of transformations, so it's used sometimes in proofs. *(rarely in this class though)*

  - Properties:

$$\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$$
$$\mathrm{tr}(\mathbf{A} + \mathbf{B}) = \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B})$$

JACOBS UNIVERSITY

- **Vector Norms**

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \qquad \|x\|_\infty = \max_i |x_i|$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}. \qquad \|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

- **Matrix norms**: Norms can also be defined for matrices, such as the Frobenius norm**:**

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2} = \sqrt{\mathrm{tr}(A^T A)}$$

JACOBS UNIVERSITY

- **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

JACOBS UNIVERSITY

- Symmetric matrix

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

- Skew-symmetric matrix

$$\mathbf{A}^T = -\mathbf{A} \qquad \begin{bmatrix} 0 & -2 & -5 \\ 2 & 0 & -7 \\ 5 & 7 & 0 \end{bmatrix}$$

OUTLINE

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

Matrix multiplication can be used to transform vectors. A matrix used in this way is called a transformation matrix.

JACOBS UNIVERSITY

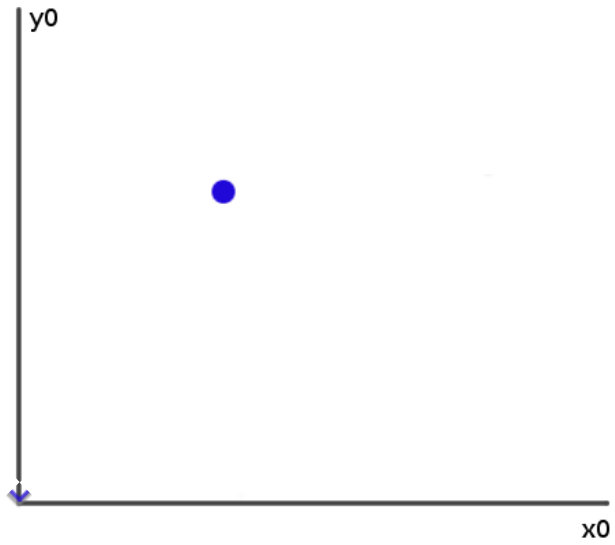
- Matrices can be used to transform vectors in useful ways, through multiplication: x'= Ax

- Simplest is scaling:

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

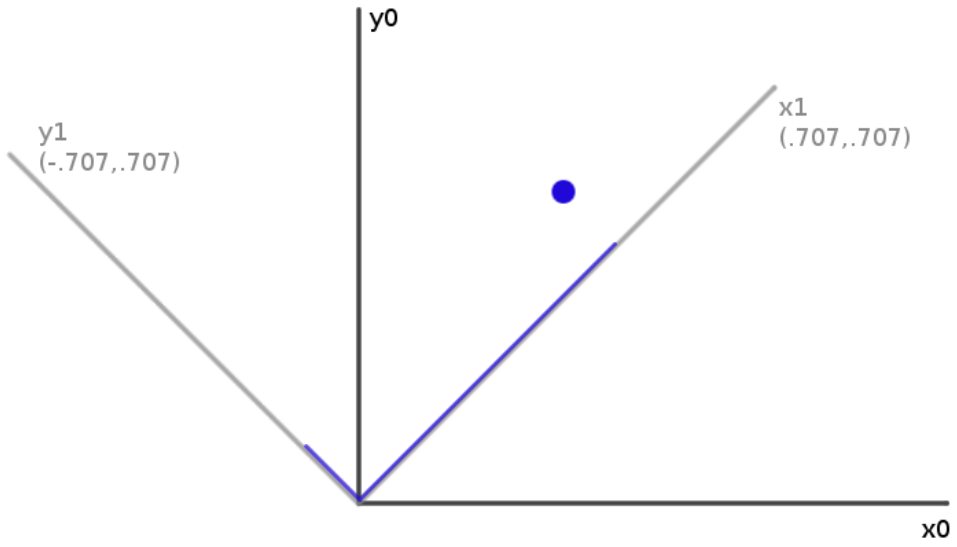
JACOBS UNIVERSITY

- How can you convert a vector represented in frame “0” to a new, rotated coordinate frame “1”?



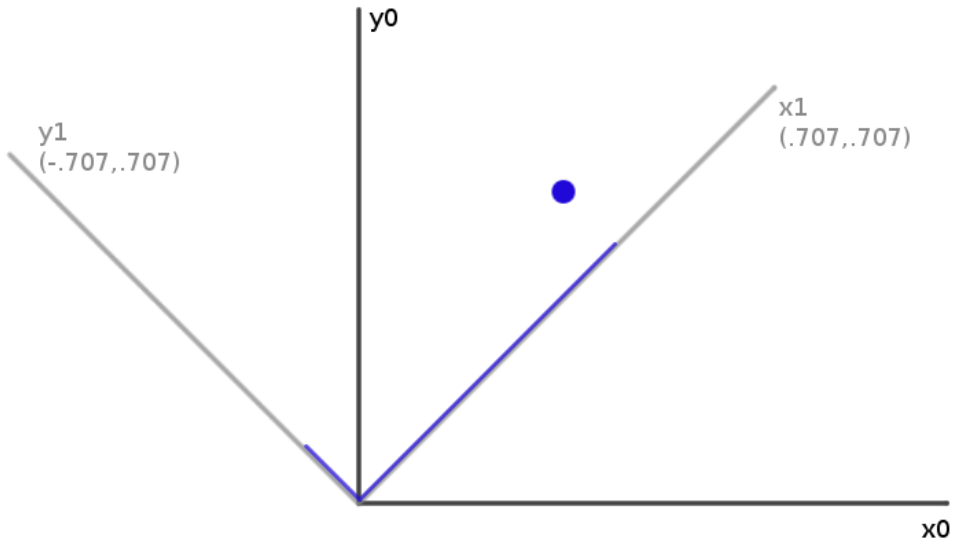
JACOBS UNIVERSITY

- How can you convert a vector represented in frame “0” to a new, rotated coordinate frame “1”?
- Remember what a vector is:
  [component in direction of the frame’s x axis, component in direction of y axis]



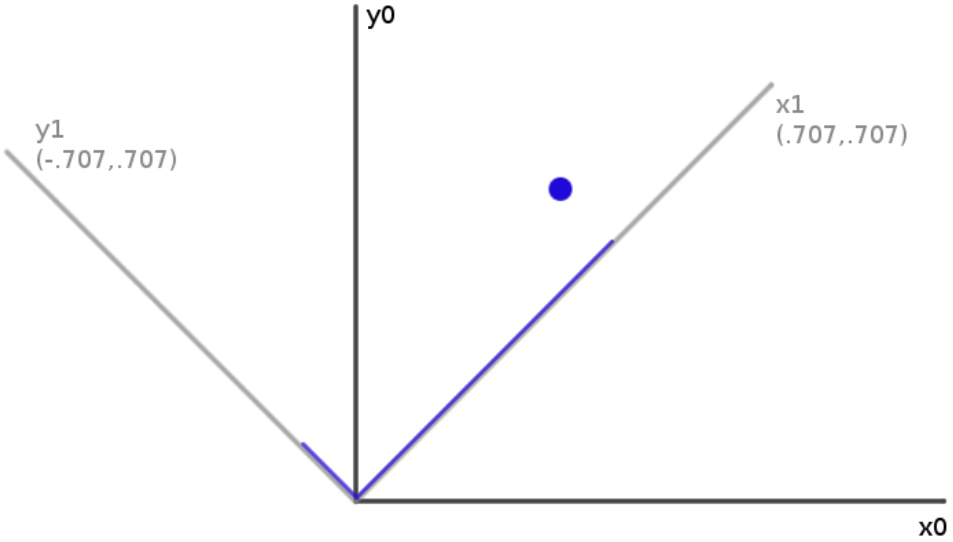
ROTATION

- So to rotate it we must produce this vector:
  [component in direction of **new** x axis, component in direction of **new** y axis]
- We can do this easily with dot products!
- New x coordinate is [original vector] **dot** [the new x axis]
- New y coordinate is [original vector] **dot** [the new y axis]



JACOBS UNIVERSITY

- Insight: this is what happens in a matrix*vector multiplication
  - Result x coordinate is:
    [original vector] **dot** [matrix row 1]
  - So matrix multiplication can rotate a vector p:

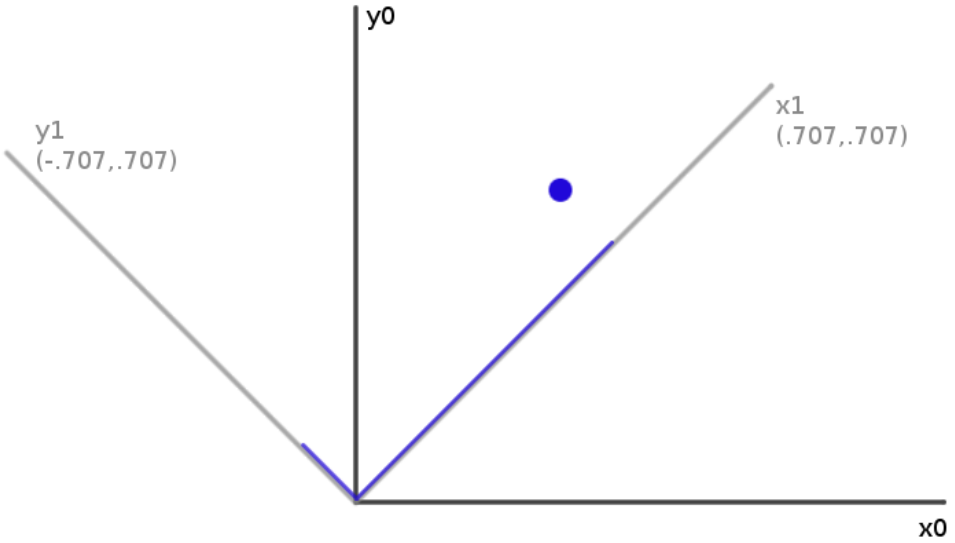


$$R \times p = \text{rotated } p' \qquad p = \begin{bmatrix} px \\ py \end{bmatrix}$$

$$R = \begin{bmatrix} .707 & .707 \\ -.707 & .707 \end{bmatrix} \qquad \begin{bmatrix} px' \\ py' \end{bmatrix}$$

ROTATION

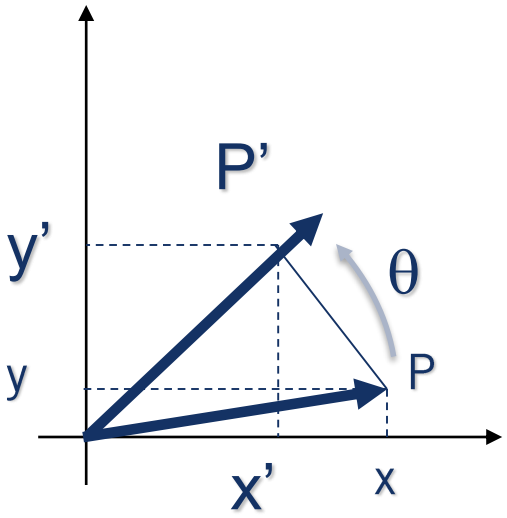
- Suppose we express a point in the new coordinate system which is rotated left
- If we plot the result in the **original** coordinate system, we have rotated the point right



Thus, rotation matrices can be used to rotate vectors. We'll usually think of them in that sense-- as operators to rotate vectors

JACOBS UNIVERSITY

Counter-clockwise rotation by an angle θ



$$x' = \cos\theta\ x - \sin\theta\ y$$

$$y' = \cos\theta\ y + \sin\theta\ x$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{R}\,\mathbf{P}$$

TRANSFORMATION MATRICES

- Multiple transformation matrices can be used to transform a point: $p' = R_2 R_1 S p$

TRANSFORMATION MATRICES

- Multiple transformation matrices can be used to transform a point: $p' = R_2 R_1 S p$
- The effect of this is to apply their transformations one after the other, from right to left.
- In the example above, the result is $(R_2 (R_1 (S p)))$

TRANSFORMATION MATRICES

JACOBS UNIVERSITY

- Multiple transformation matrices can be used to transform a point:
  $p' = R_2 R_1 S p$
- The effect of this is to apply their transformations one after the other, from **right to left**.
- In the example above, the result is
  $(R_2 (R_1 (S p)))$
- The result is exactly the same if we multiply the matrices first, to form a single transformation matrix:
  $p' = (R_2 R_1 S) p$

- In general, a matrix multiplication lets us linearly combine components of a vector

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

- This is sufficient for scale, rotate, skew transformations.
- But notice, we can't add a constant! ☹

HOMOGENEOUS SYSTEM

- The (somewhat hacky) solution? Stick a “1” at the end of every vector:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- Now we can rotate, scale, and skew like before, AND translate (note how the multiplication works out, above)
- This is called “homogeneous coordinates”

HOMOGENEOUS SYSTEM

JACOBS UNIVERSITY

- In homogeneous coordinates, the multiplication works out so the rightmost column of the matrix is a vector that gets added.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- Generally, a homogeneous transformation matrix will have a bottom row of [0 0 1], so that the result has a “1” at the bottom too.

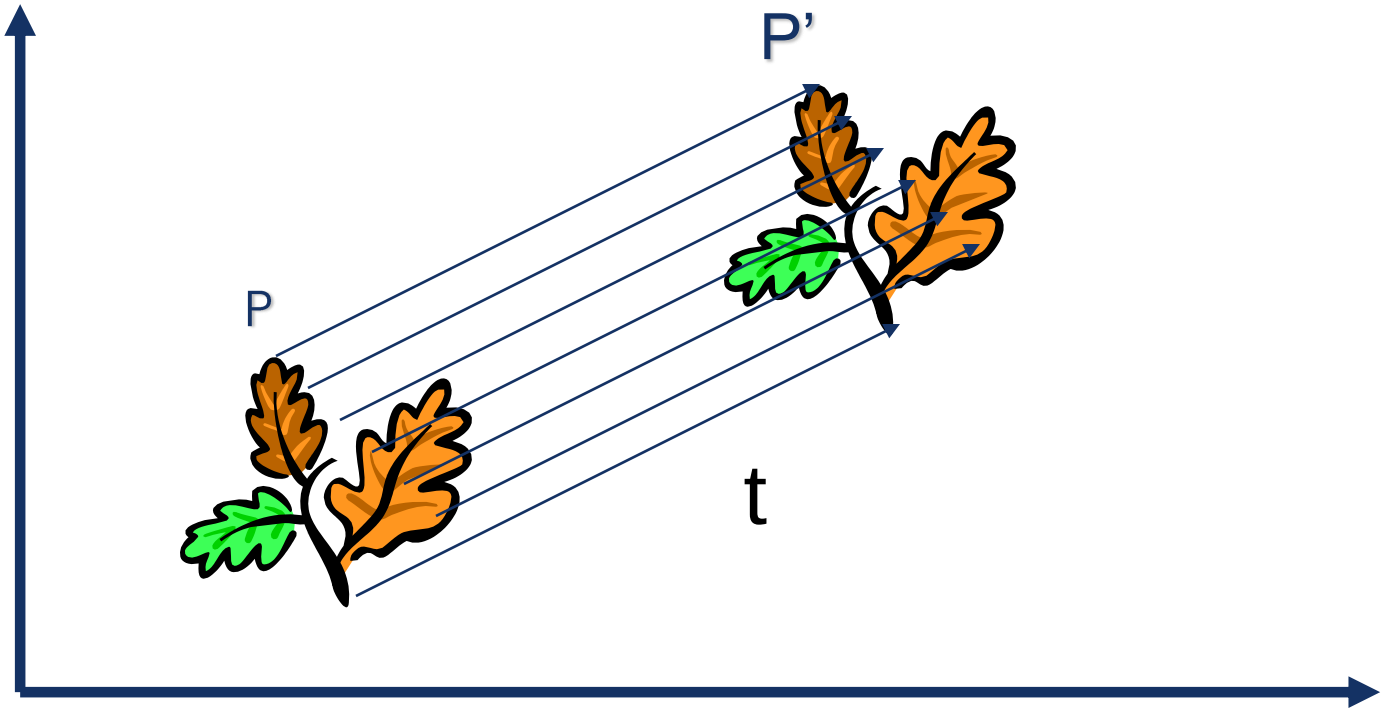
HOMOGENEOUS SYSTEM

JACOBS UNIVERSITY

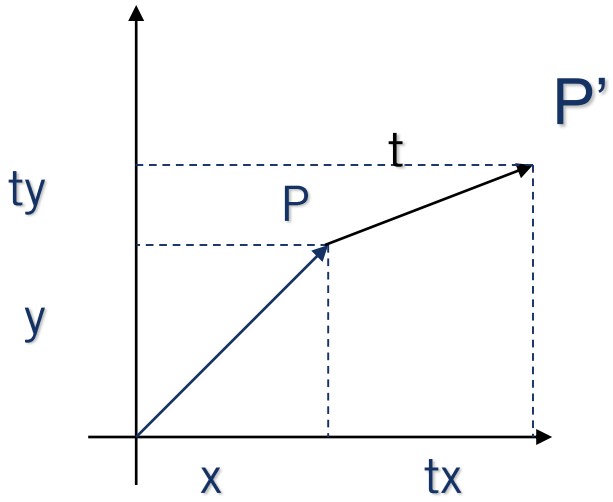
- One more thing we might want: to divide the result by something
  - For example, we may want to divide by a coordinate, to make things scale down as they get farther away in a camera image
  - Matrix multiplication can't actually divide
  - So, **by convention**, in homogeneous coordinates, we'll divide the result by its last coordinate after doing a matrix multiplication

$$\begin{bmatrix} x \\ y \\ 7 \end{bmatrix} \Rightarrow \begin{bmatrix} x/7 \\ y/7 \\ 1 \end{bmatrix}$$

2D TRANSLATION



2D TRANSLATION USING HOMOGENEOUS COORDINATES



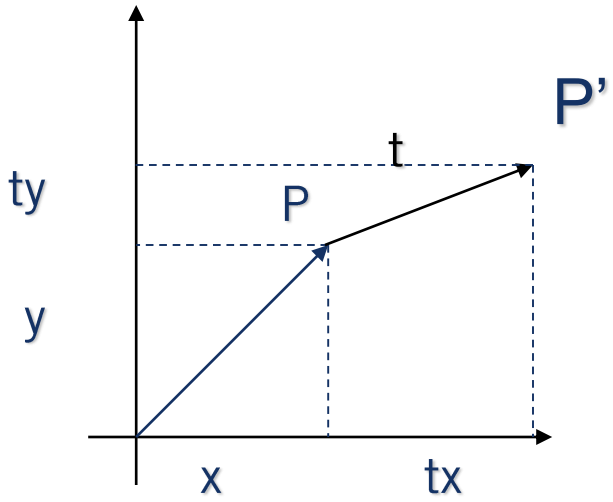
$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

$$\mathbf{P'} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

P

2D TRANSLATION USING HOMOGENEOUS COORDINATES



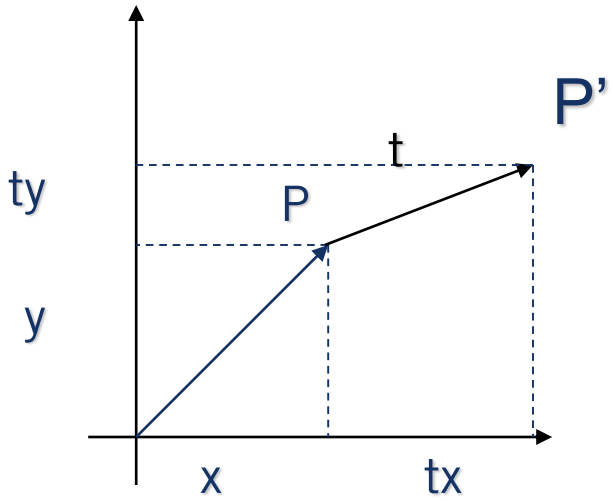
$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

$$\mathbf{P'} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & & \\ & & \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

P

2D TRANSLATION USING HOMOGENEOUS COORDINATES



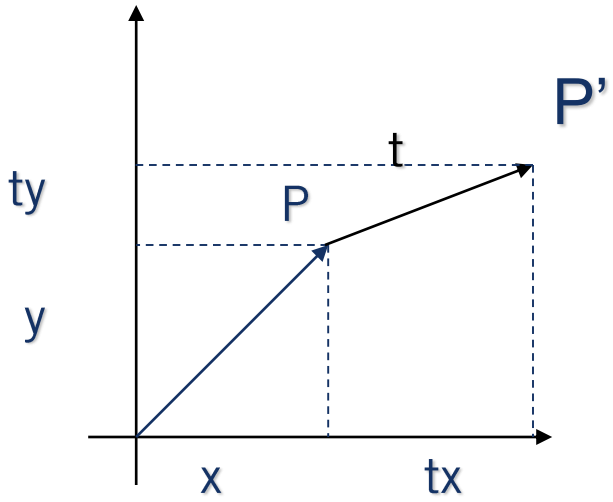
$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

$$\mathbf{P'} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \\ & & \\ & & \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

P

2D TRANSLATION USING HOMOGENEOUS COORDINATES



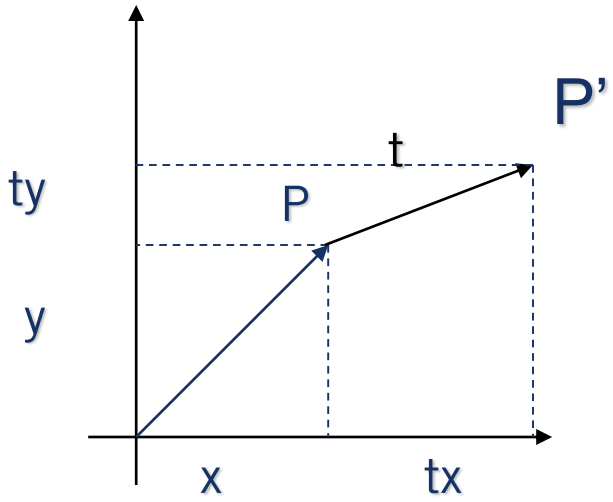
$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

P

$$\mathbf{P'} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ & & \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D TRANSLATION USING HOMOGENEOUS COORDINATES



$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

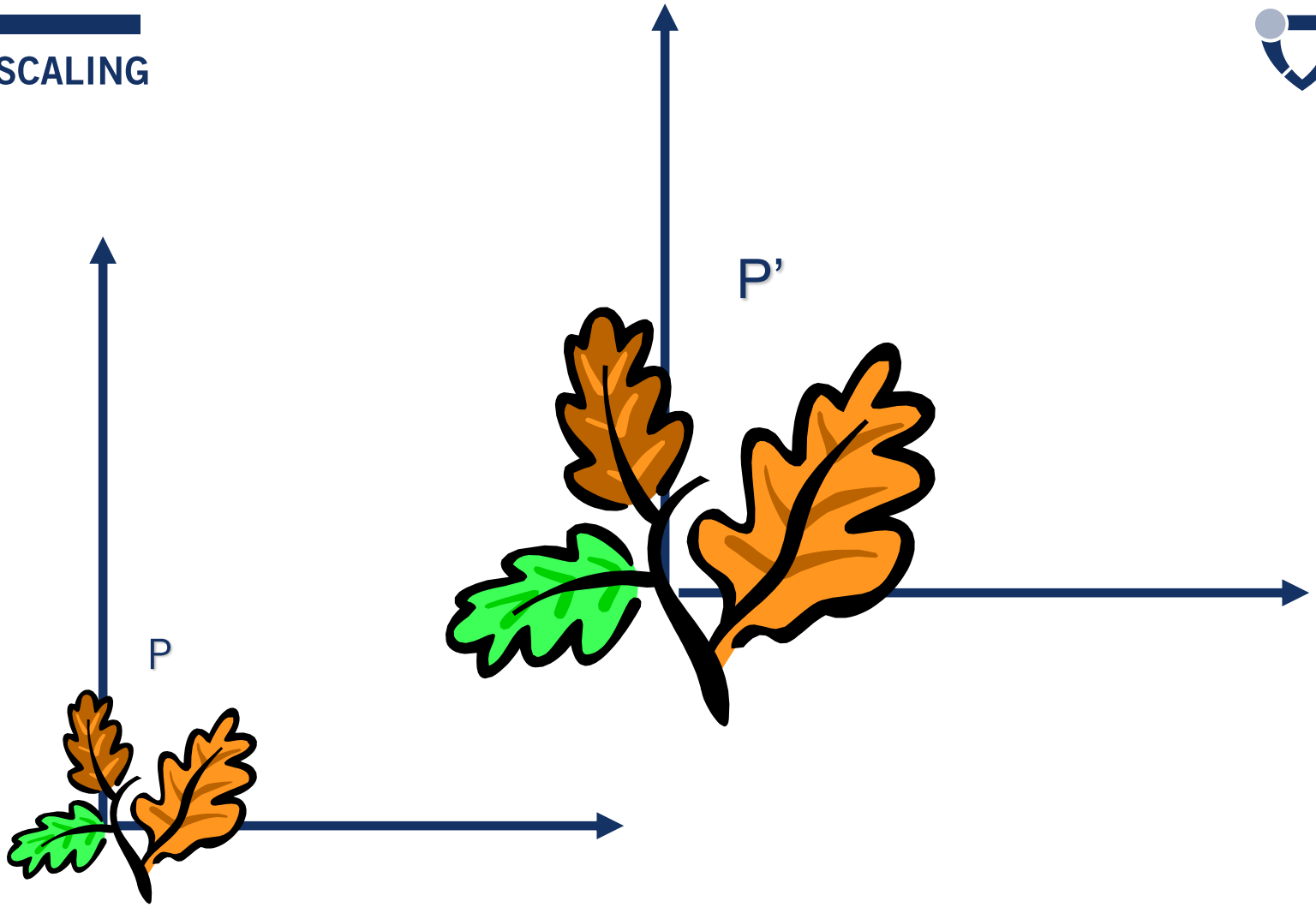
$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

t P

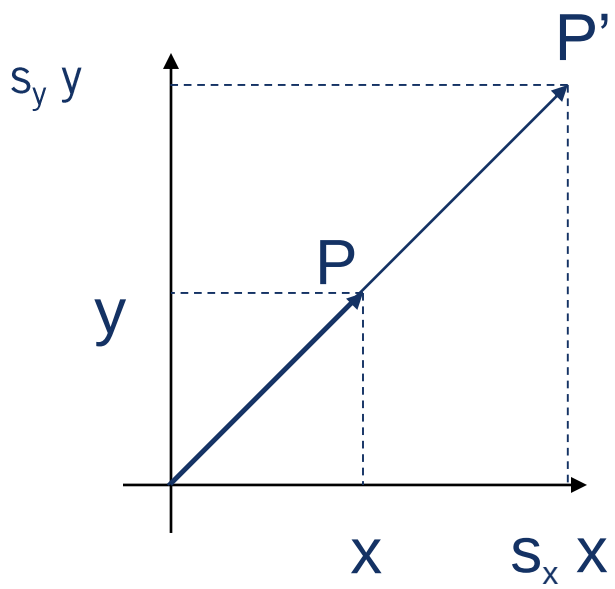
$$\mathbf{P'} \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{P} = \mathbf{T} \cdot \mathbf{P}$$

JACOBS UNIVERSITY



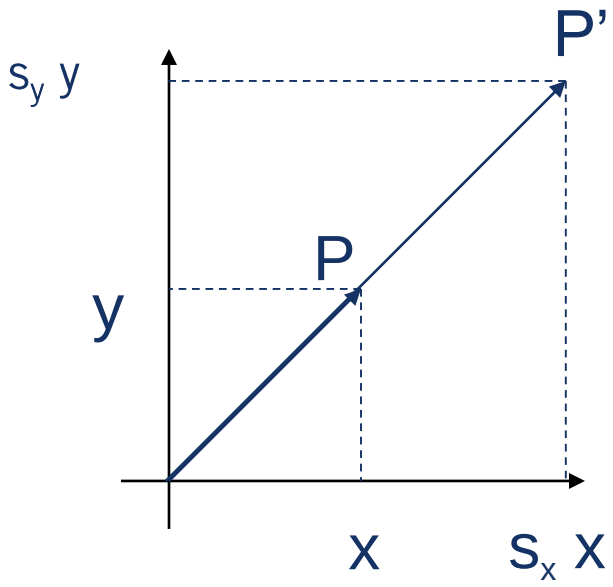
SCALING EQUATION



$$\mathbf{P} = (x, y) \rightarrow \mathbf{P}' = (s_x x, s_y y)$$

$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{P}' = (s_x x, s_y y) \rightarrow (s_x x, s_y y, 1)$$

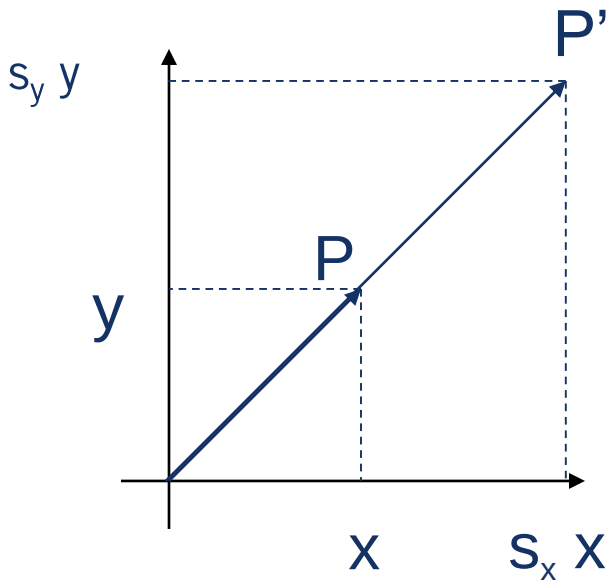


$$\mathbf{P} = (x, y) \rightarrow \mathbf{P}' = (s_x x, s_y y)$$

$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{P}' = (s_x x, s_y y) \rightarrow (s_x x, s_y y, 1)$$

$$\mathbf{P}' \rightarrow \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \begin{bmatrix} \quad \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

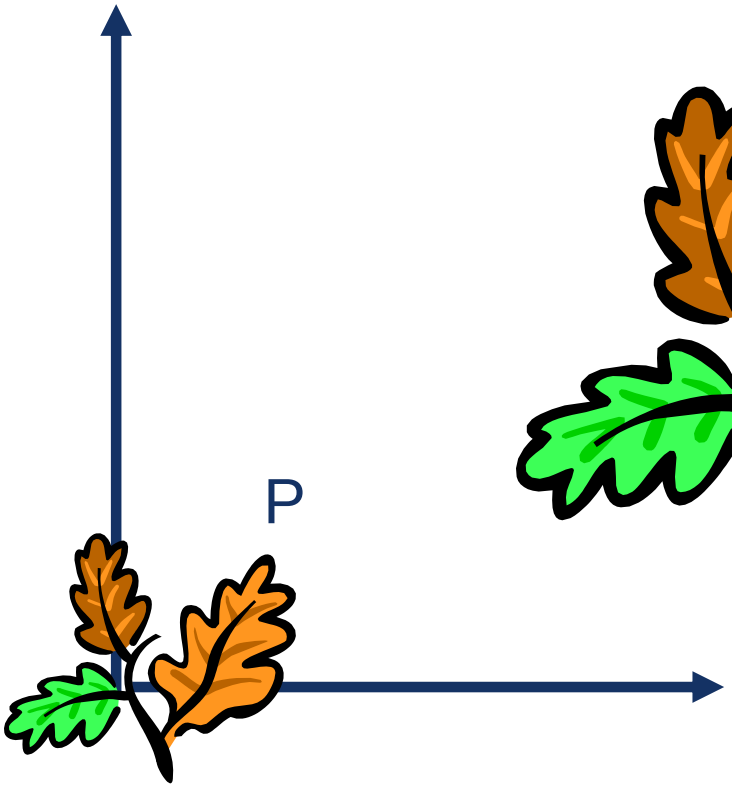


$$\mathbf{P} = (x, y) \rightarrow \mathbf{P}' = (s_x x, s_y y)$$

$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{P}' = (s_x x, s_y y) \rightarrow (s_x x, s_y y, 1)$$

$$\mathbf{P}' \rightarrow \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{S}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{S}' & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \mathbf{P} = \mathbf{S} \cdot \mathbf{P}$$



$P' = S \cdot P$

$P'' = T \cdot P'$

$$P'' = T \cdot P' = T \cdot (S \cdot P) = T \cdot S \cdot P$$

JACOBS UNIVERSITY

$$\mathbf{P}'' = \mathbf{T} \times \mathbf{S} \times \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

JACOBS UNIVERSITY

$$\mathbf{P''} = \mathbf{T} \times \mathbf{S} \times \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + t_x \\ s_y y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} S & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

TRANSLATING & SCALING VERSUS SCALING & TRANSLATING

$$\mathbf{P'''} = \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + t_x \\ s_y y + t_y \\ 1 \end{bmatrix}$$

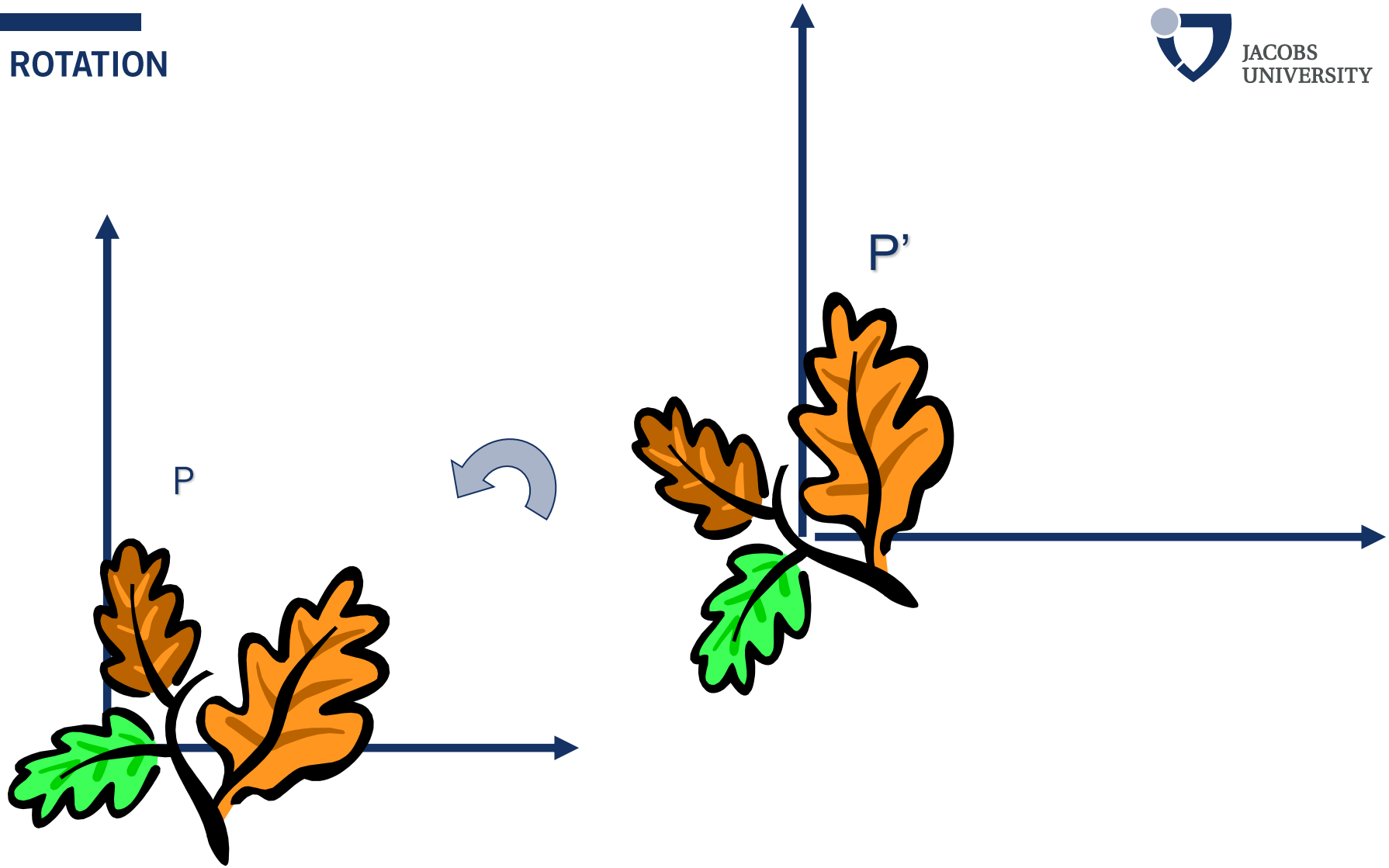
TRANSLATING & SCALING
!= SCALING & TRANSLATING

$$\mathbf{P'''} = \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + t_x \\ s_y y + t_y \\ 1 \end{bmatrix}$$

$$\mathbf{P'''} = \mathbf{S} \cdot \mathbf{T} \cdot \mathbf{P} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

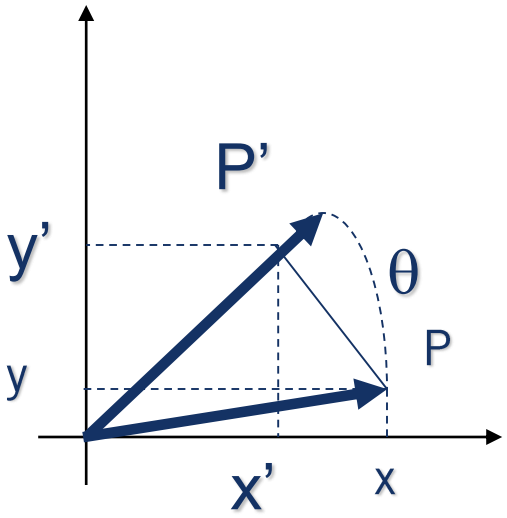
$$= \begin{bmatrix} s_x & 0 & s_x t_x \\ 0 & s_y & s_y t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + s_x t_x \\ s_y y + s_y t_y \\ 1 \end{bmatrix}$$

ROTATION



JACOBS UNIVERSITY

Counter-clockwise rotation by an angle θ



$$x' = \cos\theta\, x - \sin\theta\, y$$

$$y' = \cos\theta\, y + \sin\theta\, x$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{R}\,\mathbf{P}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

A 2D rotation matrix is 2x2

Note: R belongs to the category of *normal* matrices and satisfies many interesting properties:

$$\mathbf{R} \cdot \mathbf{R}^{\mathbf{T}} = \mathbf{R}^{\mathbf{T}} \cdot \mathbf{R} = \mathbf{I}$$

$$\det(\mathbf{R}) = 1$$

JACOBS UNIVERSITY

- Transpose of a rotation matrix produces a rotation in the opposite direction

$$\mathbf{R} \cdot \mathbf{R}^{\mathbf{T}} = \mathbf{R}^{\mathbf{T}} \cdot \mathbf{R} = \mathbf{I}$$

$$\det(\mathbf{R}) = 1$$

- The rows of a rotation matrix are always mutually perpendicular (a.k.a. orthogonal) unit vectors
- *(and so are its columns)*

JACOBS UNIVERSITY

P'= (T R S) P

$$\mathbf{P'} = \mathbf{T} \cdot R \cdot \mathbf{S} \cdot \mathbf{P} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} R\,S & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is the form of the general-purpose transformation matrix

OUTLINE

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

The inverse of a transformation matrix reverses its effect

JACOBS UNIVERSITY

- Given a matrix **A**, its inverse $\mathbf{A}^{-1}$ is a matrix such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$
- E.g. $$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$$
- Inverse does not always exist. If $\mathbf{A}^{-1}$ exists, **A** is *invertible* or *non-singular*. Otherwise, it's *singular*.
- Useful identities, for matrices that are invertible:

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A}$$
$$(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$
$$\mathbf{A}^{-T} \triangleq (\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$$

JACOBS UNIVERSITY

Pseudoinverse

- Say you have the matrix equation AX=B, where A and B are known, and you want to solve for X
- You could calculate the inverse and pre-multiply by it:

$$A^{-1}AX=A^{-1}B \rightarrow X=A^{-1}B$$

- But calculating the inverse for large matrices often brings problems with computer floating-point resolution (because it involves working with very small and very large numbers together).

- Or, your matrix might not even have an inverse.

- Directly ask python to solve for X in AX=B, by typing **np.linalg.solve(A, B)**
  [in MATLAB: **A \ B**]
- Python/Matlab will try several appropriate numerical methods (including the pseudoinverse if the inverse doesn't exist)
- Python/Matlab will return the value of X which solves the equation
  - If there is no exact solution, it will return the closest one
  - If there are many solutions, it will return the smallest one

JACOBS UNIVERSITY

- MATLAB example:

$$AX = B$$

$$A = \begin{bmatrix} 2 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

```
>> x = A\B
x =
    1.0000
   -0.5000
```

JACOBS UNIVERSITY

- Python example:

$$AX = B$$

$$A = \begin{bmatrix} 2 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

```
>> import numpy as np
>> x = np.linalg.solve(A,B)
x =
    1.0000
   -0.5000
```

OUTLINE

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

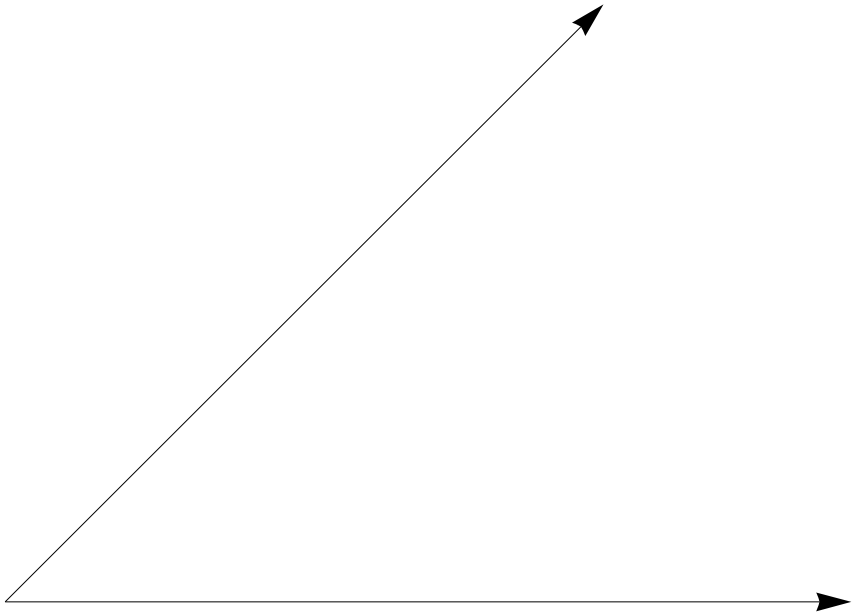
The rank of a transformation matrix tells you how many dimensions it transforms a vector to.

LINEAR INDEPENDENCE

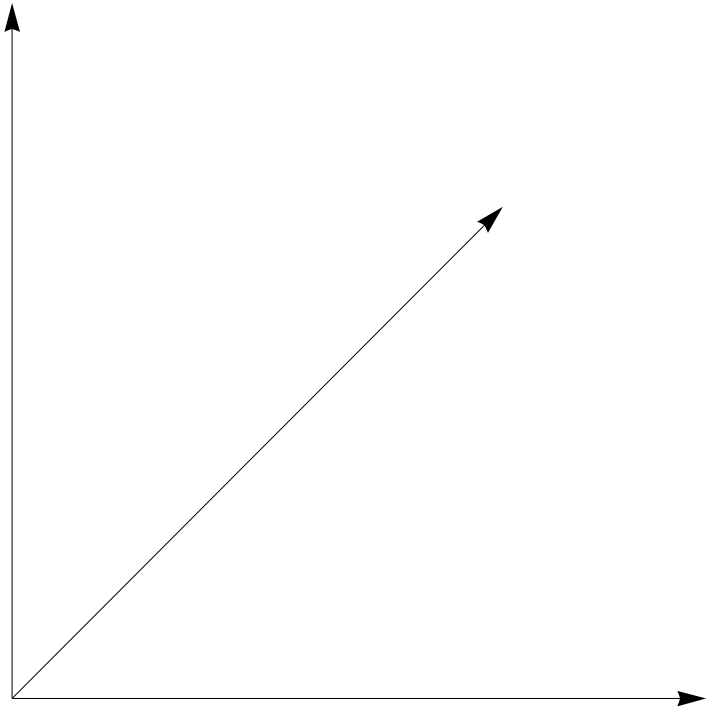
- Suppose we have a set of vectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$
- If we can express $\mathbf{v}_1$ as a linear combination of the other vectors $\mathbf{v}_2 \ldots \mathbf{v}_n$, then $\mathbf{v}_1$ is linearly *dependent* on the other vectors.
  - The direction $\mathbf{v}_1$ can be expressed as a combination of the directions $\mathbf{v}_2 \ldots \mathbf{v}_n$. (E.g. $\mathbf{v}_1 = .7\ \mathbf{v}_2 - .7\ \mathbf{v}_4$)
- If no vector is linearly dependent on the rest of the set, the set is linearly *independent*.
  - Common case: a set of vectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$ is always linearly independent if each vector is perpendicular to every other vector (and non-zero)

JACOBS UNIVERSITY

Linearly independent set



Not linearly independent



JACOBS UNIVERSITY

- Column/row rank

$$\begin{aligned}\text{col-rank}(\mathbf{A}) &= \text{ the maximum number of linearly independent column vectors of } \mathbf{A}\\ \text{row-rank}(\mathbf{A}) &= \text{ the maximum number of linearly independent row vectors of } \mathbf{A}\end{aligned}$$

  - Column rank always equals row rank

- Matrix rank

$$\text{rank}(\mathbf{A}) \triangleq \text{col-rank}(\mathbf{A}) = \text{row-rank}(\mathbf{A})$$

MATRIX RANK

JACOBS UNIVERSITY

- For transformation matrices, the rank tells you the dimensions of the output
- E.g. if rank of **A** is 1, then the transformation

$$\mathbf{p'=Ap}$$

maps points onto a line.

- Here's a matrix with rank 1:

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x+y \\ 2x+2y \end{bmatrix}$$

All points get mapped to the line y=2x

MATRIX RANK

JACOBS UNIVERSITY

- If an *m* x *m* matrix is rank *m*, we say it's "full rank"
  - Maps an *m* x 1 vector uniquely to another *m* x 1 vector
  - An inverse matrix can be found
- If rank < *m*, we say it's "singular"
  - At least one dimension is getting collapsed. No way to look at the result and tell what the input was
  - Inverse does not exist
- Inverse also doesn't exist for non-square matrices

OUTLINE

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

SVD is an algorithm that represents any matrix as the product of 3 matrices. It is used to discover interesting structure in a matrix

SINGULAR VALUE DECOMPOSITION (SVD)

JACOBS UNIVERSITY

- There are several computer algorithms that can “factorize” a matrix, representing it as the product of some other matrices
- The most useful of these is the Singular Value Decomposition.
- Represents any matrix **A** as a product of three matrices: $\mathbf{U\Sigma V^T}$
- MATLAB command: [U,S,V]=svd(A)
- Python: np.linalg.svd

JACOBS UNIVERSITY

$A = U\Sigma V^T$

- Where **U** and **V** are rotation matrices and **Σ** is a scaling matrix.

- For example:

$$\underset{U}{\begin{bmatrix} -.40 & .916 \\ .916 & .40 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 5.39 & 0 \\ 0 & 3.154 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.05 & .999 \\ .999 & .05 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 3 & -2 \\ 1 & 5 \end{bmatrix}}$$

JACOBS UNIVERSITY

- Beyond 2D:
  - In general, if **A** is *m* x *n*, then **U** will be *m* x *m*, **Σ** will be *m* x *n*, and $\mathbf{V^T}$ will be *n* x *n*.
  - (Note the dimensions work out to produce *m* x *n* after multiplication)

$$\underset{U}{\begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}$$

JACOBS UNIVERSITY

- **U** and **V** are always rotation matrices.
  - Geometric rotation may not be an applicable concept, depending on the matrix. So we call them "unitary" matrices – each column is a unit vector.

- **Σ** is a diagonal matrix
  - The number of nonzero entries = rank of **A**
  - The algorithm always sorts the entries high to low

$$\underset{U}{\begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}$$

- Let's look at a non-geometric interpretation

JACOBS UNIVERSITY

$$\underset{U}{\begin{bmatrix} -.39 & -.92 \\ -.92 & .39 \end{bmatrix}} \times \underset{\Sigma}{\begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}$$

- Column 1 of **U** gets scaled by the first value from **Σ**.

$$\underset{U\Sigma}{\begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} \qquad \underset{A_{partial}}{\begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix}}$$

- The resulting vector gets scaled by row 1 of $\mathbf{V^T}$ to produce a contribution to the columns of **A**

JACOBS UNIVERSITY

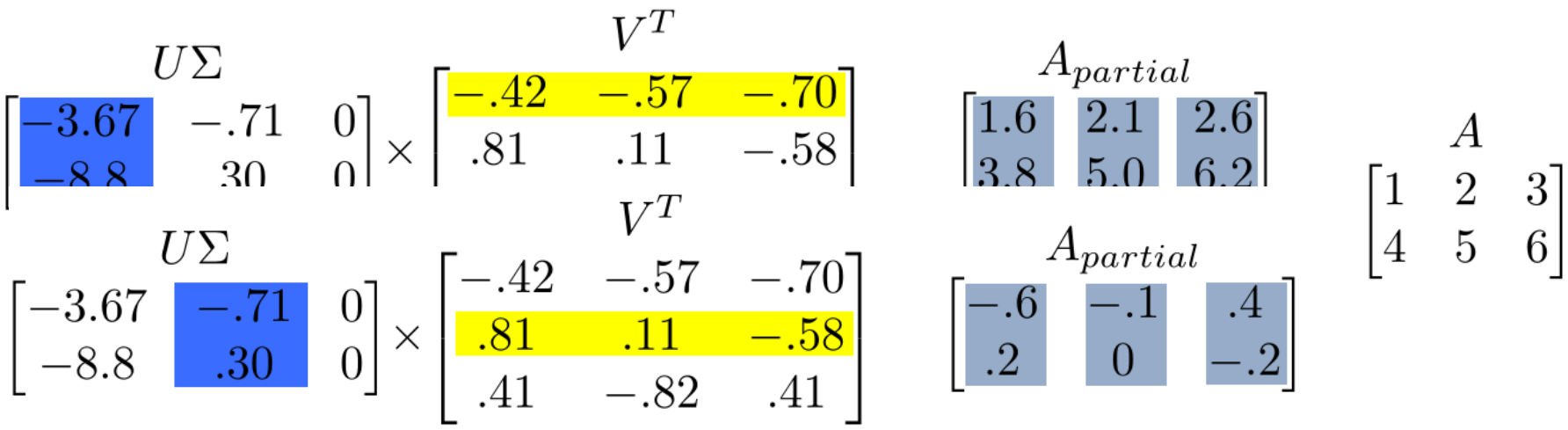
$$\underset{U\Sigma}{\begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} \quad \underset{A_{partial}}{\begin{bmatrix} 1.6 & 2.1 & 2.6 \\ 3.8 & 5.0 & 6.2 \end{bmatrix}}$$

$$+ \quad \underset{U\Sigma}{\begin{bmatrix} -3.67 & -.71 & 0 \\ -8.8 & .30 & 0 \end{bmatrix}} \times \underset{V^T}{\begin{bmatrix} -.42 & -.57 & -.70 \\ .81 & .11 & -.58 \\ .41 & -.82 & .41 \end{bmatrix}} \quad \underset{A_{partial}}{\begin{bmatrix} -.6 & -.1 & .4 \\ .2 & 0 & -.2 \end{bmatrix}}$$

$$= \quad \underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}$$

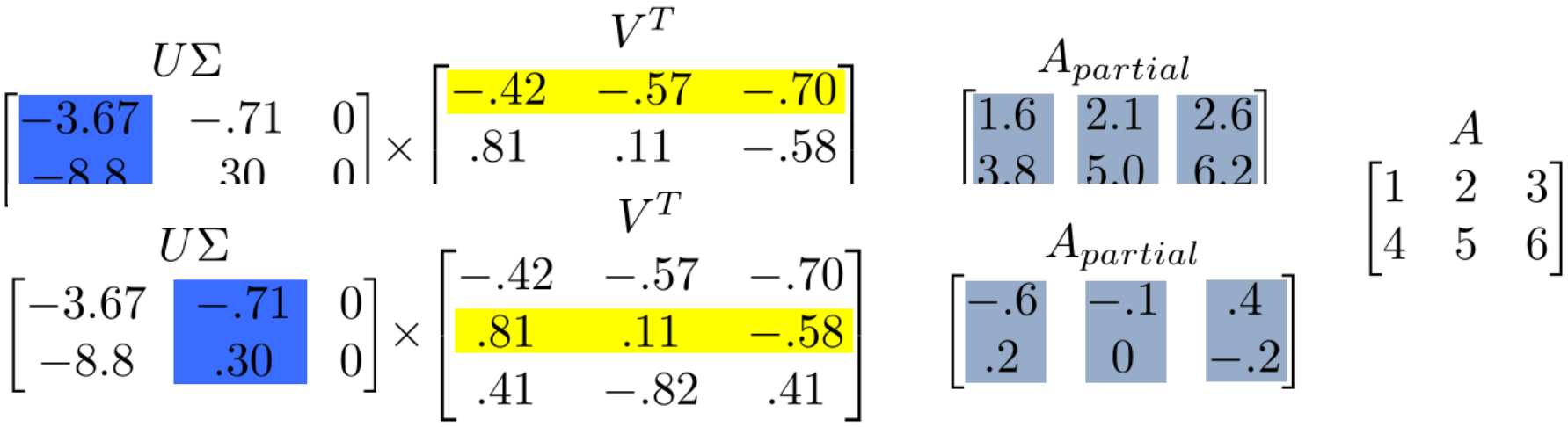
- The resulting vector gets scaled by row 1 of V^T to produce a contribution to the columns of A
- Each product of (*column i of* U)·(*value i from* Σ)·(*row i of* V^T) produces a component of the final **A**.

SINGULAR VALUE DECOMPOSITION (SVD) - APPLICATIONS



- We're building **A** as a linear combination of the columns of ***U***
- Using all columns of ***U***, we'll rebuild the original matrix perfectly
- But, in real-world data, often we can just use the first few columns of ***U*** and we'll get something close (e.g. the first $\boldsymbol{A_{partial}}$ above)

SINGULAR VALUE DECOMPOSITION (SVD) - APPLICATIONS



- We can call those first few columns of ***U*** the *Principal Components* of the data
- They show the major patterns that can be added to produce the columns of the original matrix
- The rows of $\boldsymbol{V^T}$ show how the *principal components* are mixed to produce the columns of the matrix

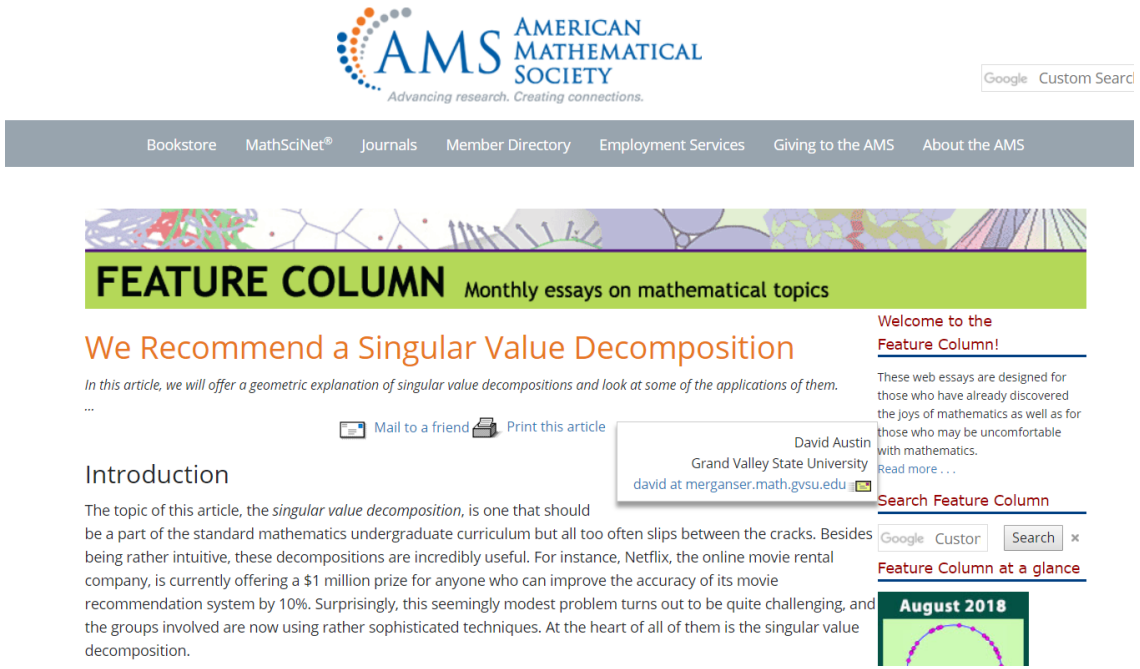
$$\Sigma$$
$$\begin{bmatrix} 9.51 & 0 & 0 \\ 0 & .77 & 0 \end{bmatrix}$$

- First column large effect
- Second column much smaller effect

SINGULAR VALUE DECOMPOSITION (SVD) - APPLICATIONS

- A geometric explanation of SVD is here:

http://www.ams.org/publicoutreach/feature-column/fcarc-svd



EIGENVECTOR AND EIGENVALUE

JACOBS UNIVERSITY

- An eigenvector **x** of a linear transformation *A* is a non-zero vector that, when *A* is applied to it, does not change direction.

$$Ax = \lambda x, \quad x \neq 0.$$

EIGENVECTOR AND EIGENVALUE

JACOBS UNIVERSITY

- An eigenvector **x** of a linear transformation *A* is a non-zero vector that, when *A* is applied to it, does not change direction.
- Applying *A* to the eigenvector only scales the eigenvector by the scalar value λ, called an eigenvalue.

$$Ax = \lambda x, \quad x \neq 0.$$

- We want to find all the eigenvalues of A:

$$Ax = \lambda x, \quad x \neq 0.$$

- Which can we written as:

$$Ax = (\lambda I)x \quad x \neq 0.$$

- Therefore:

$$(\lambda I - A)x = 0, \quad x \neq 0.$$

EIGENVECTOR AND EIGENVALUE

JACOBS UNIVERSITY

- We can solve for eigenvalues by solving:

$$(\lambda I - A)x = 0, \quad x \neq 0.$$

- Since we are looking for non-zero **x**, we can instead solve the above equation as:

$$|(\lambda I - A)| = 0.$$

JACOBS UNIVERSITY

- The trace of a A is equal to the sum of its eigenvalues:

$$\mathrm{tr}A = \sum_{i=1}^{n} \lambda_i.$$

- The determinant of A is equal to the product of its eigenvalues

$$|A| = \prod_{i=1}^{n} \lambda_i.$$

- The rank of A is equal to the number of non-zero eigenvalues of A.

- Calculating the SVD consists of finding the eigenvalues and eigenvectors of AA^T and A^TA.

OUTLINE

JACOBS UNIVERSITY

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix Calculus

JACOBS UNIVERSITY

- Let a function $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ take as input a matrix A of size m × n and returns a real value.
- Then the **gradient** of **f:**

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

JACOBS UNIVERSITY

- Every entry in the matrix is: $$\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}.$$

- the size of $\nabla_A f(A)$ is always the same as the size of A. So if A is just a vector x:

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

For $x \in \mathbb{R}^n$, let $f(x) = b^T x$ for some known vector $b \in \mathbb{R}^n$

$$f(x) = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\frac{\partial f(x)}{\partial x_k} = ?$$

$$\nabla_x f(x) = ?$$

For $x \in \mathbb{R}^n$, let $f(x) = b^T x$ for some known vector $b \in \mathbb{R}^n$

$$f(x) = \sum_{i=1}^{n} b_i x_i$$

$$\frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{n} b_i x_i = b_k.$$

$$\nabla_x b^T x = b$$

Properties

- $\nabla_x(f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x).$
- For $t \in \mathbb{R}$, $\nabla_x(t\ f(x)) = t\nabla_x f(x).$

JACOBS UNIVERSITY

- The Hessian matrix with respect to x, written $\nabla_x^2 f(x)$

  or simply as H is the n × n matrix of partial derivatives

$$\nabla_x^2 f(x) \in \mathbb{R}^{n \times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

$$\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}.$$

WHAT WE HAVE LEARNED

JACOBS UNIVERSITY

- Vectors and matrices
  - Basic Matrix Operations
  - Determinants, norms, trace
  - Special Matrices
- Transformation Matrices
  - Homogeneous coordinates
  - Translation
- Matrix inverse
- Matrix rank
- Singular Value Decomposition
- Eigenvectors
- Matrix calculus

SEE YOU TOMORROW!