CS 168 Discussion-Week 5

Ali Hatamizadeh
5/1/2020

Overview

- Homogeneous Coordinates
- Transformations
- Bilinear Interpolation
- Hands-on coding practice

Homogeneous Coordinates

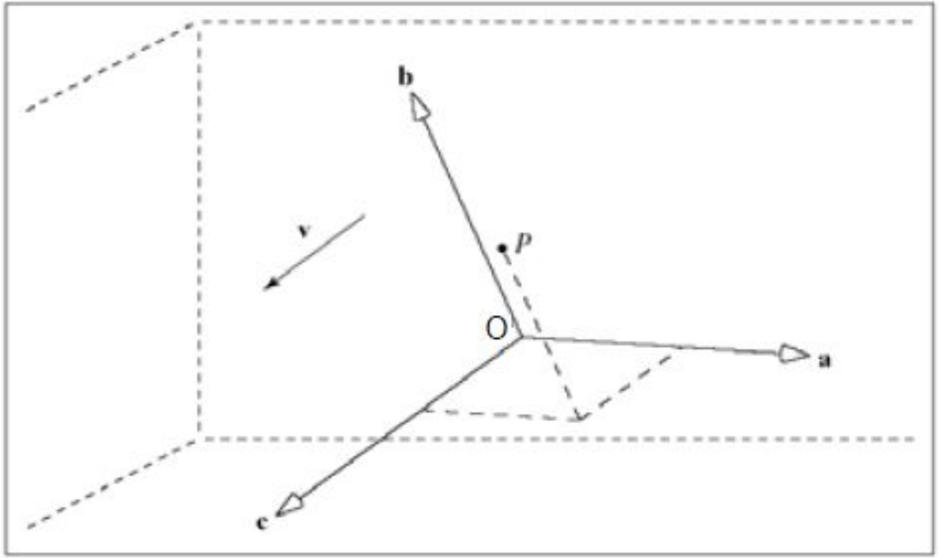
Homogeneous Coordinates

- Vectors and points are both presented as 4×1 column matrices:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Homogeneous Representation of Points and Vectors

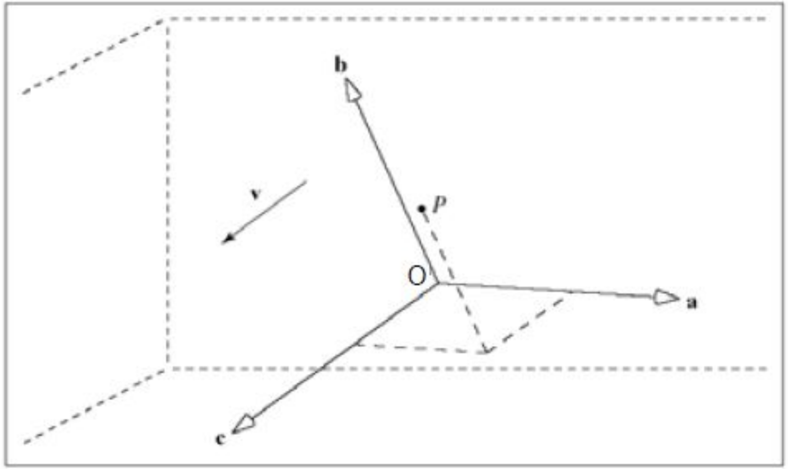
- Suppose we have a coordinate system represented by unit vectors of a,b and c as well as coordinate O for the origin



Homogeneous Representation of Points and Vectors

- Then a point $p = (p_1, p_2, p_3)$ can be represented as:

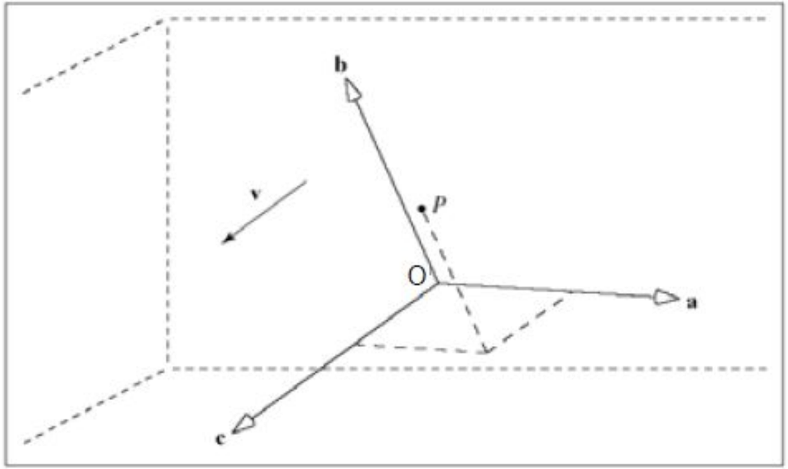
$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c}$$



Homogeneous Representation of Points and Vectors

- Similarly, a vector $v = (v_1, v_2, v_3)$ can be represented as:

$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c}$$

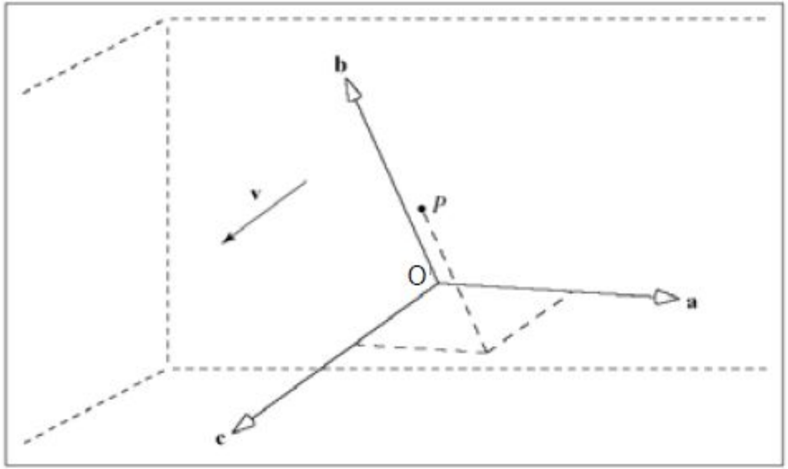


Homogeneous Representation of Points and Vectors

- In homogeneous coordinates, we can represent them as:

$$\mathbf{v} = [\mathbf{a} \;\; \mathbf{b} \;\; \mathbf{c} \;\; O] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

$$P = [\mathbf{a} \;\; \mathbf{b} \;\; \mathbf{c} \;\; O] \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$



Homogeneous Representation of Points and Vectors

- Points and vectors are both represented as 4×1 column matrices. Does it make sense to just add them together ? what's the outcome ?

$$[p_1, p_2, p_3, 1]^T + [v_1, v_2, v_3, 0]^T = [p_1 + v_1, p_2 + v_2, p_3 + v_3, 1]^T$$

Transformations

UCLA

Transforms

- Linear Transformation: function T: $R^n \rightarrow R^m$ is a linear transform if it satisfies:

$$T(c_1\vec{u} + c_2\vec{v}) = c_1 T(\vec{u}) + c_2 T(\vec{v})$$

- Additivity and homogeneity are two important characteristics of linear transformations.
- T can be obviously represented by a matrix.
- Intuitively, linear transforms leave the origin untouched.

Linear Transforms

- Linear Transformation can be compactly written as matrix multiplications:

$$
\begin{aligned}
Q &= T(P) \\
&= \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix} \\
&= \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} \\
&= \mathbf{M}P
\end{aligned}
$$

Linear Transforms

- What kind of transformations can we get from the following ?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

UCLA

Linear Transforms

- How about scaling ?

$$\begin{bmatrix} m_{1,1} & 0 \\ 0 & m_{2,2} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{1,1} P_x \\ m_{2,2} P_y \end{bmatrix}$$

Linear Transforms

- How about rotation ?

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

Linear Transforms

- How about shearing ?

$$\begin{bmatrix} 1 & -\alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} Q_x \\ Q_y \end{bmatrix}$$

Linear Transforms

- How about translation?

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} = \begin{bmatrix} m_{11}P_x + m_{12}P_y \\ m_{21}P_x + m_{22}P_y \end{bmatrix}$$

Linear Transforms

- Let’s look at translation in more details. Translation can be formally described as:

$$Q = P + t$$

But this is not the same as :

$$Q = MP$$

Affine Transforms

- Translation is not a linear transformation.
- It’s an affine transformation.
- In essence, we can represent

affine transformation = linear + translation

Affine Transforms

- Transforming points:

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{array}{c} \begin{array}{cc} \text{Rotation/Scaling/Shearing} & \text{Translation} \end{array} \\ \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Affine Transforms

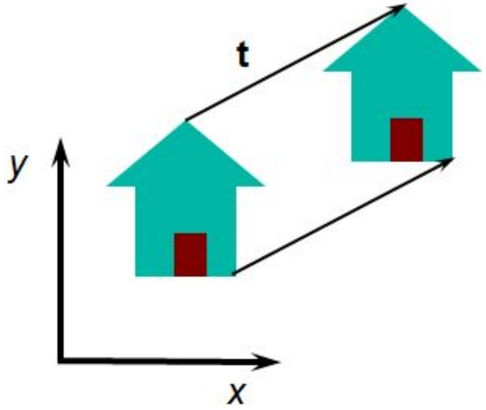
- Transforming vectors:

$$\begin{bmatrix} W_x \\ W_y \\ 0 \end{bmatrix} = \begin{array}{c} \begin{array}{cc} \text{Rotation/Scaling/Shearing} & \text{Translation} \end{array} \\ \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} V_x \\ V_y \\ 0 \end{bmatrix}$$

Affine Transforms: Translation

- Translation

$$Q = P + \mathbf{t}, \quad \mathbf{t} = (t_x \ \ t_y)^T$$



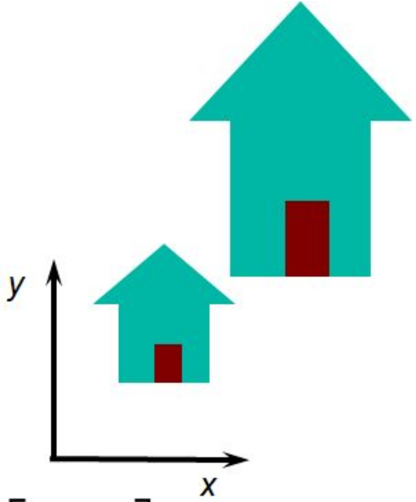
$$Q_x = P_x + t_x$$

$$Q_y = P_y + t_y$$

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Affine Transforms: Scaling

- Scaling about the origin



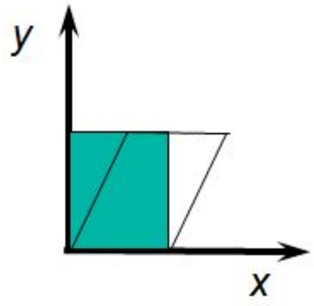
$$Q_x = s_x P_x$$
$$Q_y = s_y P_y$$

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Uniform scaling: $s_x = s_y$

Affine Transforms: Shearing

- Shearing about the origin (in x-direction)

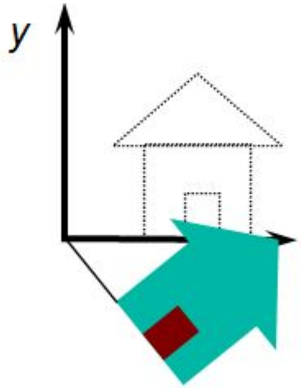


$$Q_x = P_x + aP_y$$
$$Q_y = P_y$$

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Affine Transforms: Rotation

- Rotation about the origin

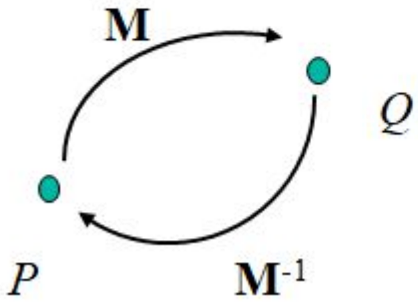


$$Q_x = \cos\theta \, P_x - \sin\theta \, P_y$$
$$Q_y = \sin\theta \, P_x + \cos\theta \, P_y$$

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Inverse of Affine Transforms

- Given an affine transform $Q = MP$, one can use the cramer's rule and find the inverse of M and eventually solve for $P = M^{-1}Q$



Or be smart about the transformation and find the inverse without inverting M

Inverse of Affine Transforms

- Scaling

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Uniform scaling: $s_x = s_y$

Inverse

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverse of Affine Transforms

- Rotation

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverse of Affine Transforms

- Shear in the x-direction

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Inverse $\begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Inverse of Affine Transforms

- Translation

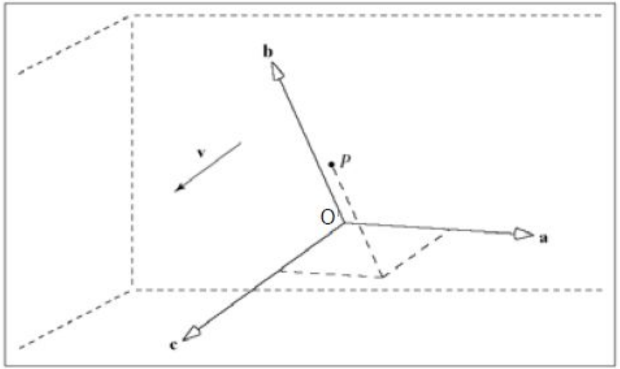
$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Inverse

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Representation of Points and Vectors

- Vector v and point P can be represented in terms of



$$
\begin{array}{cc}
 & \begin{array}{cc} \text{Rotate/Scale/Shear} & \text{Translate} \end{array} \\
\mathrm{M} = & \left[\begin{array}{cccc} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{array}\right]
\end{array}
$$

$$
\begin{array}{cc}
 & \begin{array}{cccc} \text{Basis Vector 1} & \text{Basis Vector 2} & \text{Basis Vector 3} & \textit{Origin Point} \end{array} \\
\mathrm{M} = & \left[\begin{array}{cccc} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{array}\right]
\end{array}
$$

Composite Affine Transforms

- Composing two affine transformations, produces another affine transformation:

$$Q = M_2(M_1 P) = M_2 M_1 P = MP$$

- The order of operations in affine transforms is important as affine transforms are not commutative.

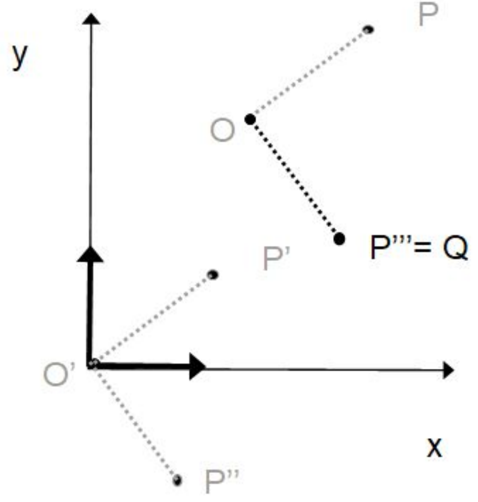
Rotate Around Arbitrary Point

- Rotating point P about an arbitrary point O:

Translate(- O_x***,-*** O_y***)***

Rotate(-90)

Translate(O_x***,*** O_y***)***



Bilinear Interpolation

Bilinear Interpolation

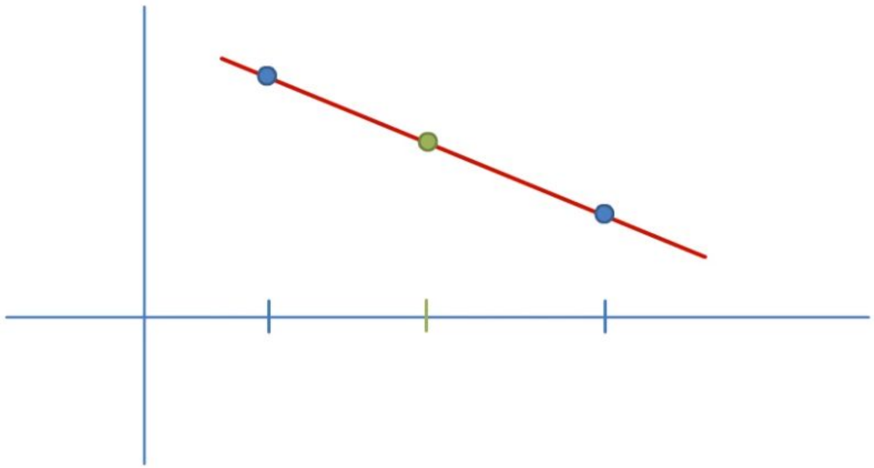
- From the first principle, what represents a line ?

$$m = \frac{(y - y_0)}{(x - x_0)}$$

$$(y - y_0) = m(x - x_0)$$

Bilinear Interpolation

- Given the information of two points, we can use the equation of a line an solve for a point that is located anywhere on that line (for instance the middle point). This is called ***interpolation.***

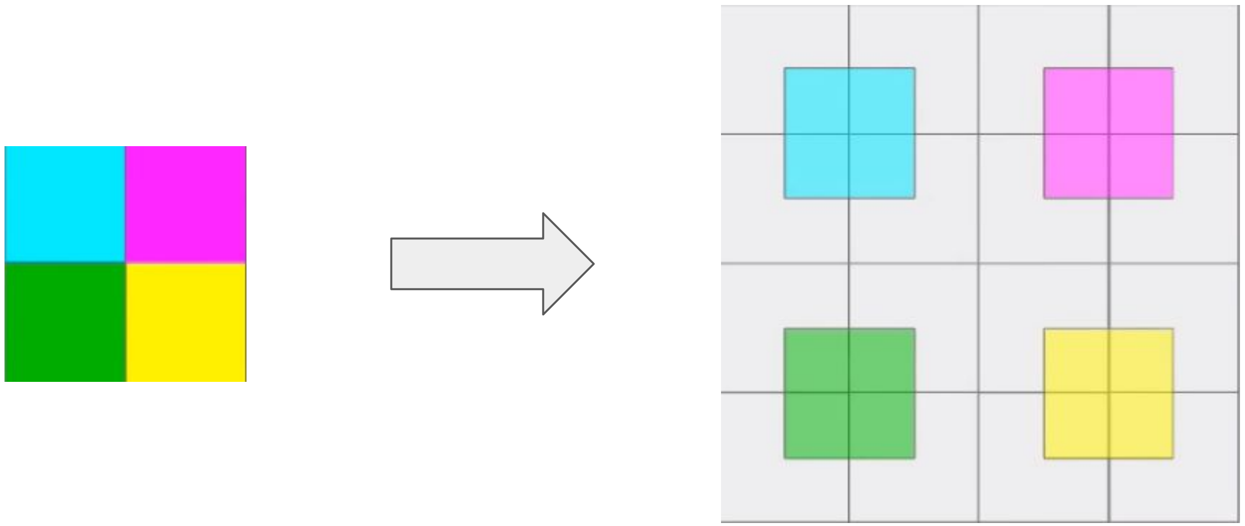


Bilinear Interpolation

- Note that here we are dealing with two variables (e.g. x and y).
  - What is we had another variable added ?
- In the case of three variables, we need to use **bilinear interpolation.**
- The general idea is to hold two variable constant, perform a linear interpolation and repeat the process for the other set of two variables.
- Bilinear interpolation is a method commonly used for resizing an image.
- The 3D equivalent of bilinear interpolation is called **trilinear interpolation,** that follows the same strategy but deals with 4 variables (one variable is a function of the other three).
- We focus on bilinear interpolation in this course.

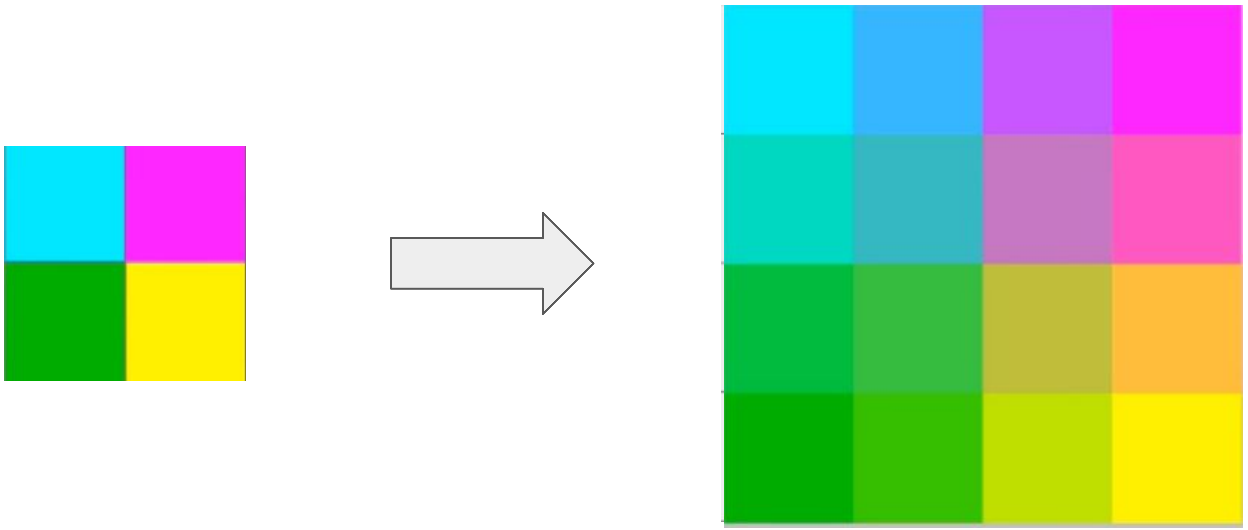
Bilinear Interpolation

- Suppose, you want to upsample a given image as shown below. What's the intuition behind this process?



Bilinear Interpolation

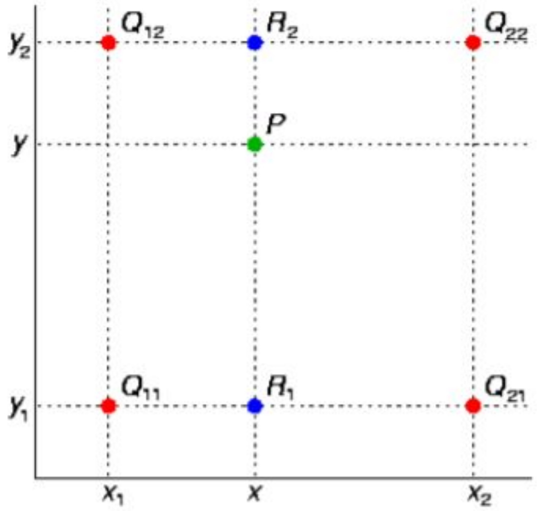
- In bilinear interpolation, each pixel looks at its 4 nearest neighboring pixels and takes into account the contribution by a weighted average.



Bilinear Interpolation

- Suppose we know the values at Q. If we were to interpolate for point P:
    - First interpolate horizontally to get R_1 and R_2:

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$
$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$



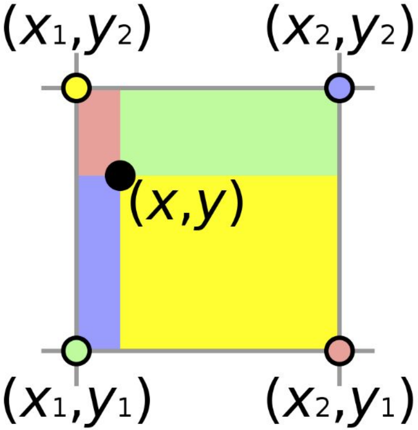
- Then interpolate vertically to get P:

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

Bilinear Interpolation

- We can further simplify by choosing a unit square where the values are known:

$$f(x,y) = f(0.0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy$$



By Cmglee (wikipedia)

End of Slides