

Image Interpolation

Computational Photography (CSCI 3240U)

Faisal Z. Qureshi

<http://vclab.science.ontariotechu.ca>



How do we resize images?



Original



Upscaling



Downscaling

Let's consider a 1D image

7	4	3
---	---	---

We want to increase its width by
a factor of 2

Let's consider a 1D image

7	4	3
---	---	---

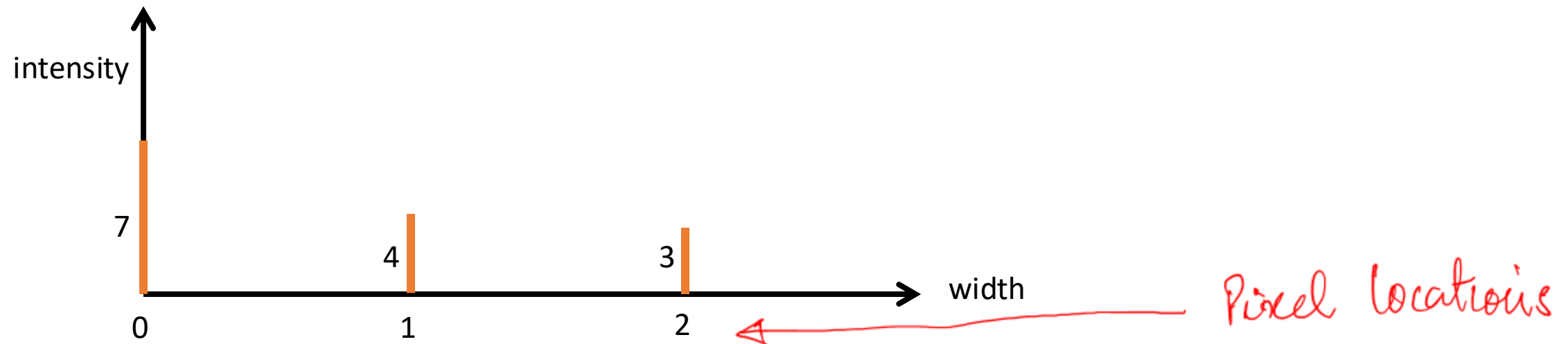
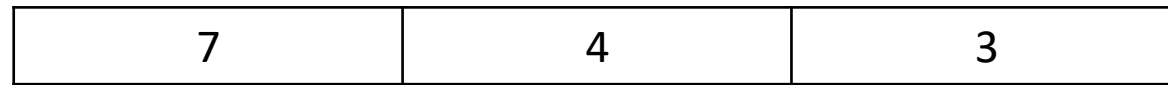
We want to increase its width by a factor of 2



7	7	4	4	3	3
---	---	---	---	---	---

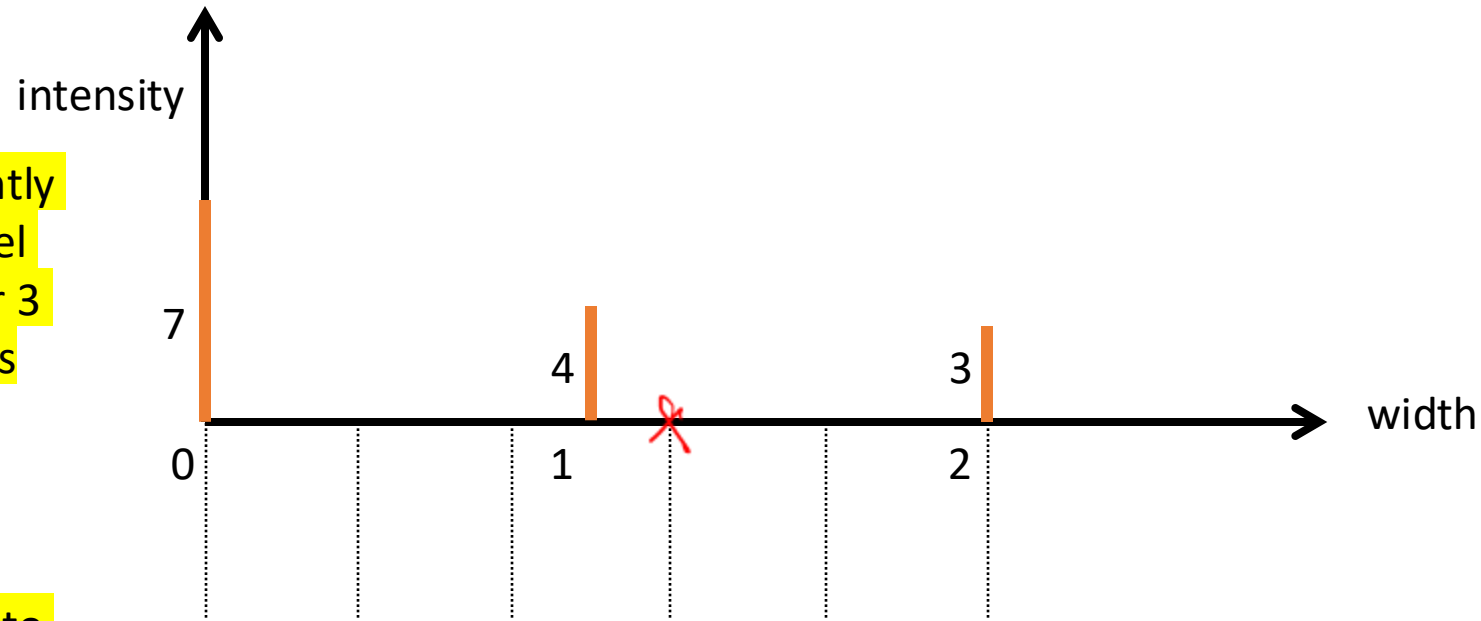
Upscaling

Resampling pixel locations

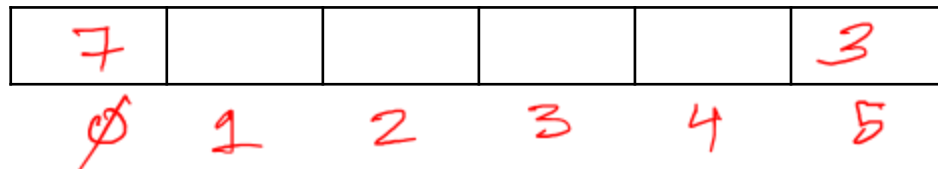


We currently have pixel values for 3 locations

Resampling pixel locations

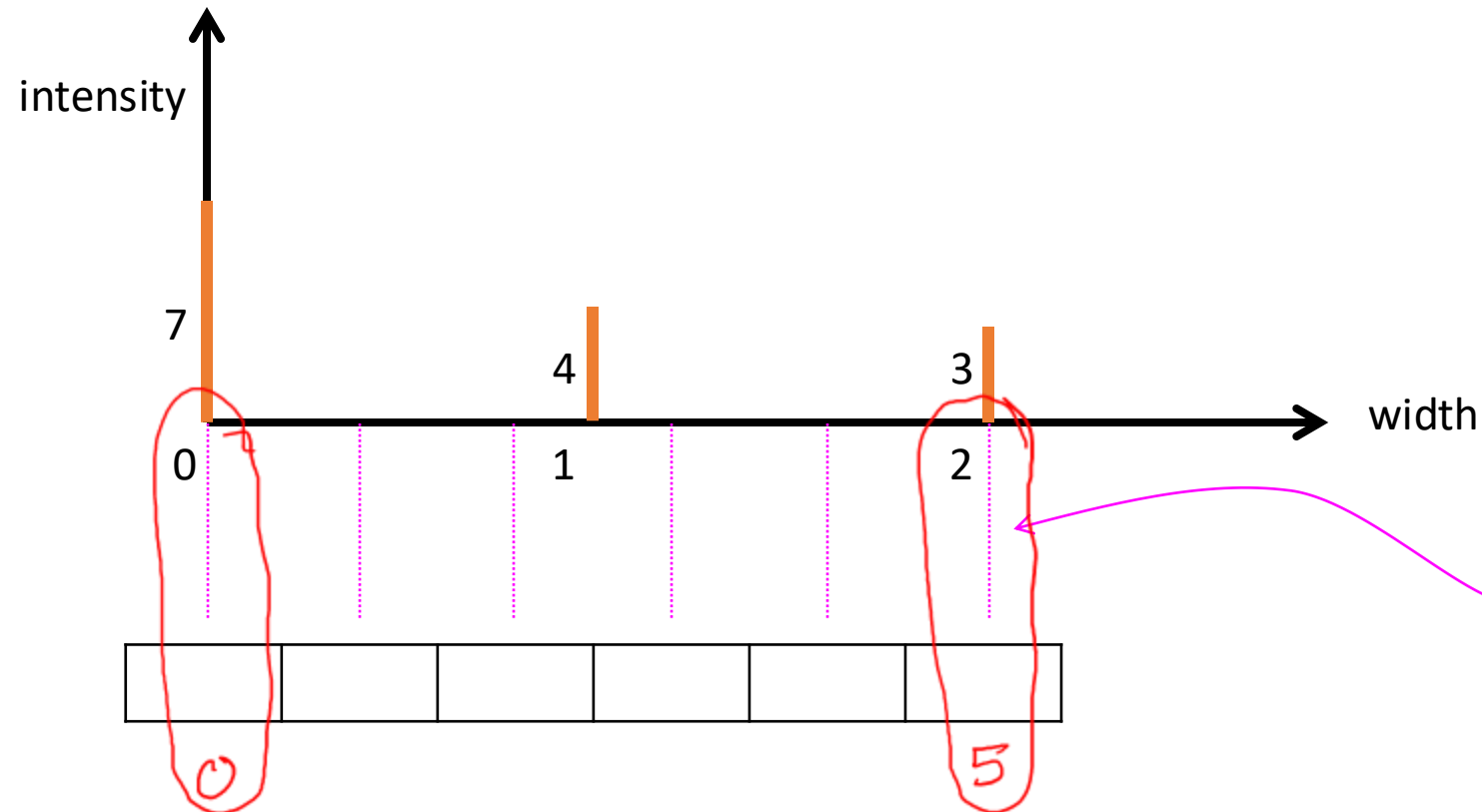
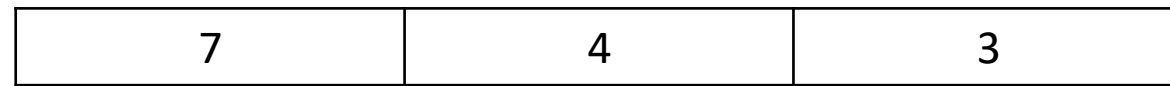


We currently have pixel values for 3 locations



We need to (re-sample) at 6 locations

Resampling pixel locations



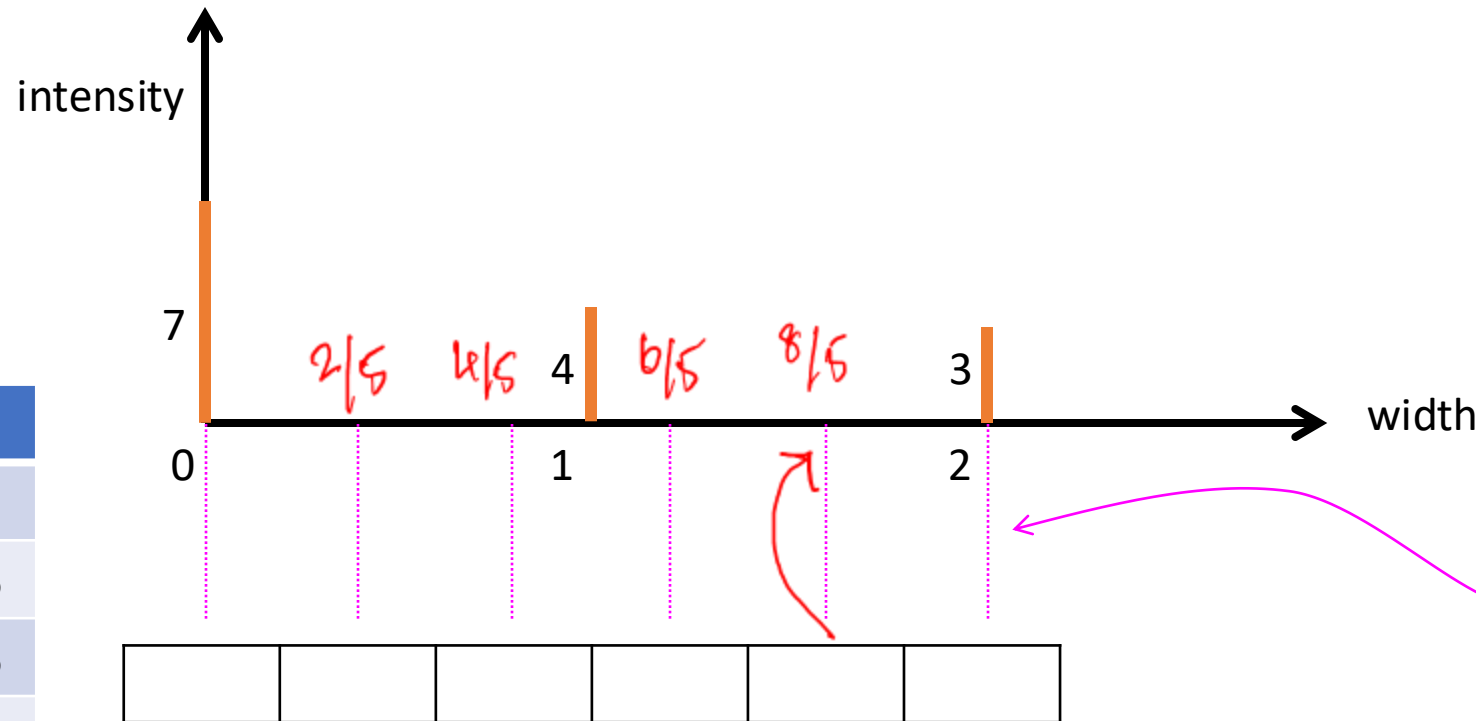
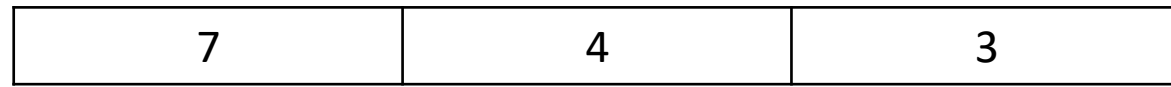
How do we compute location values for the 6 pixels between 0 and 2?

Resampling pixel locations

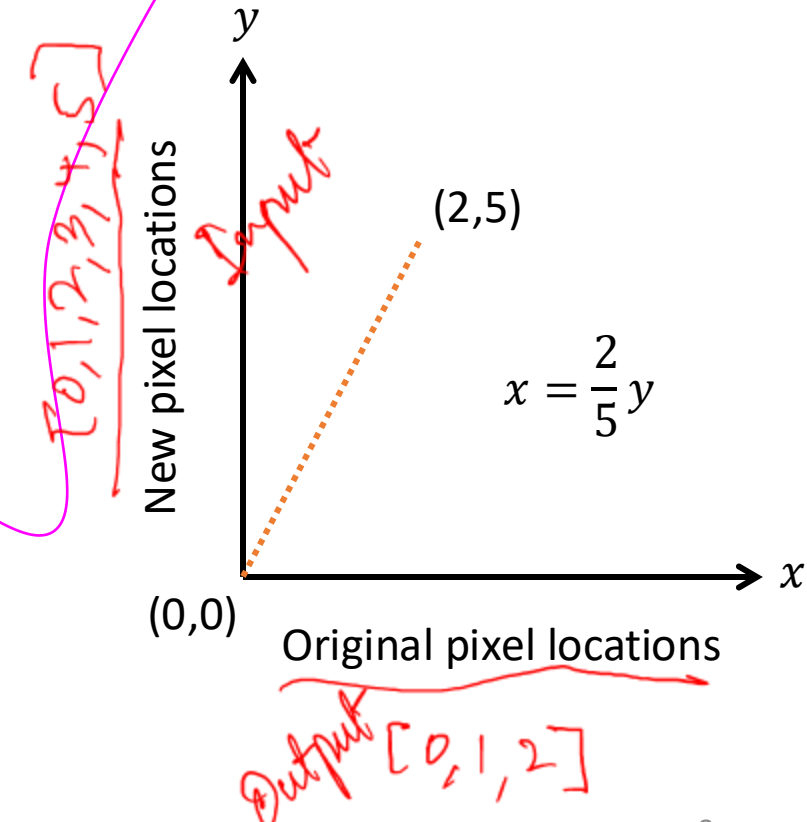
$$y = mx + b$$

↑ ↑
output input

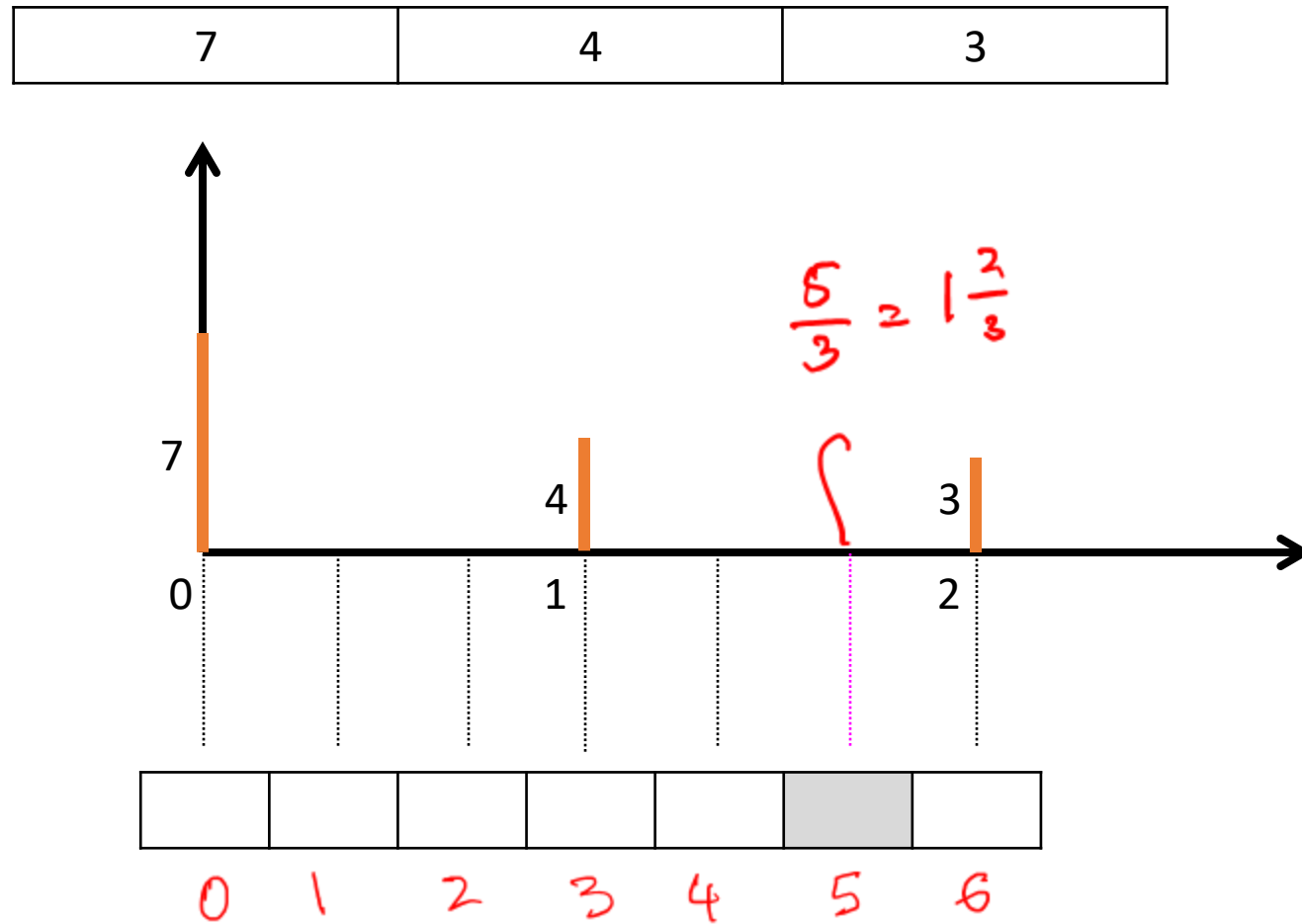
How do we compute location values for the 6 pixels between 0 and 2?



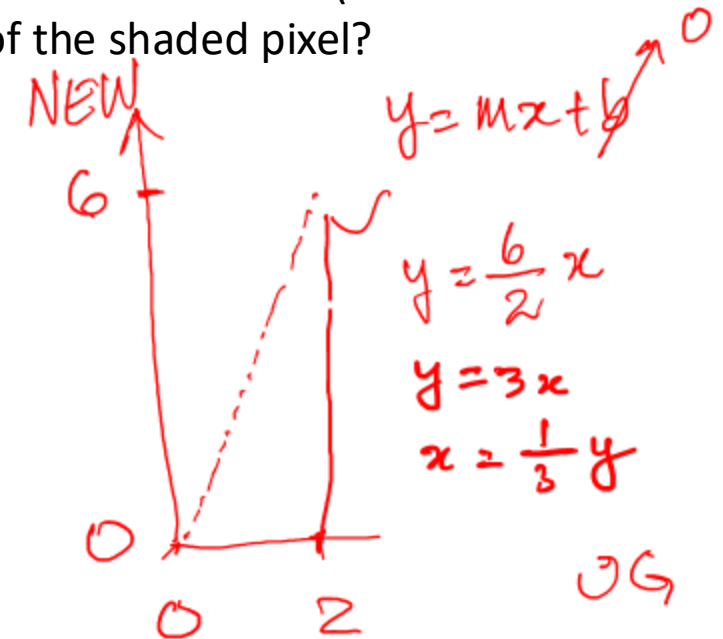
y	x
0	0
1	$2/5$
2	$4/5$
3	$6/5$
4	$8/5$
5	1



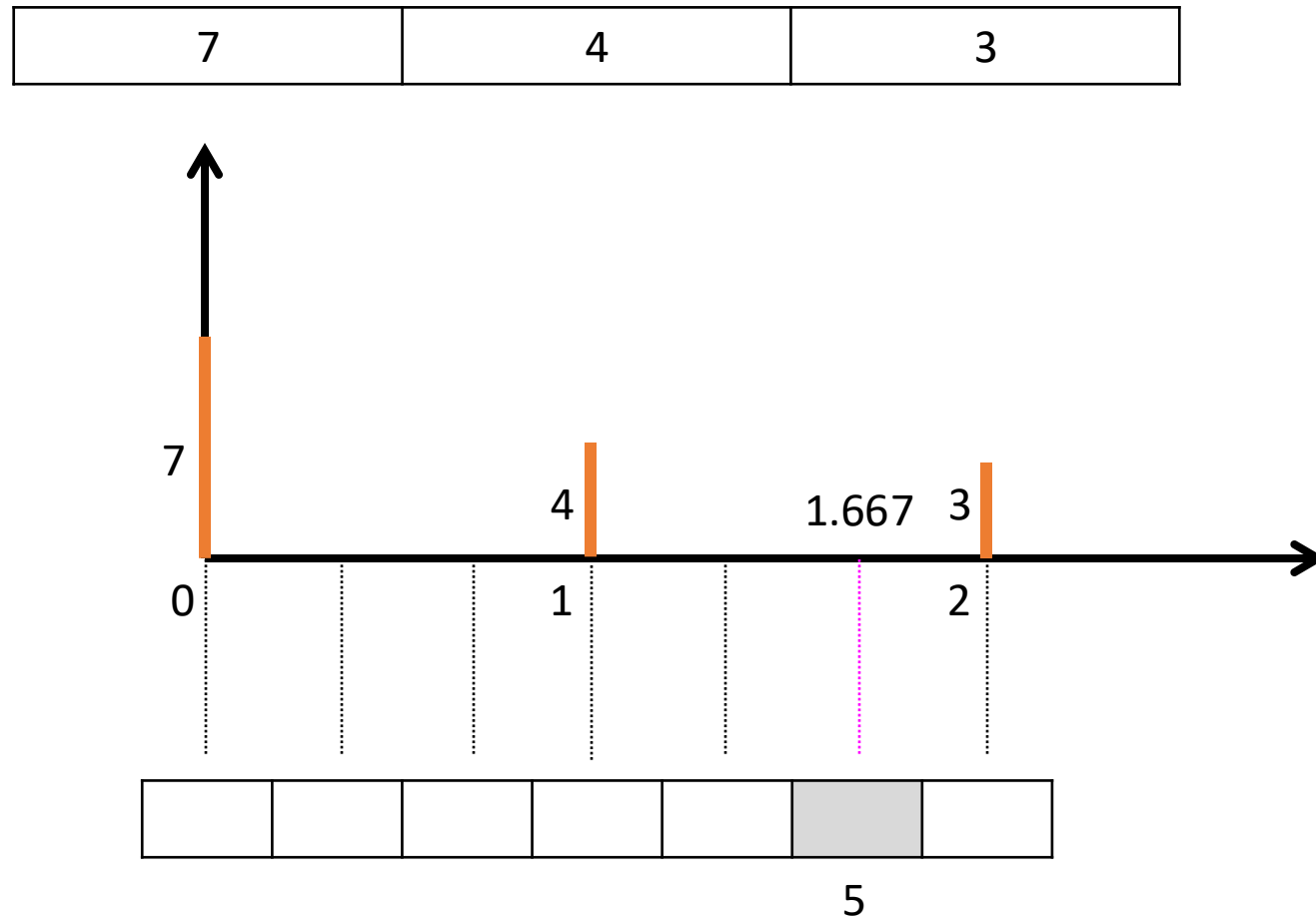
Resampling pixel locations



What is the **location** (between 0 and 2) of the shaded pixel?



Resampling pixel locations



What is the **location** (between 0 and 2) of the shaded pixel?

Given

Last pixel location in original image = 2
Last pixel location in resulting image = 6

Use the following relationship (that we developed in previous slides):

$$x = \frac{2}{6}y$$

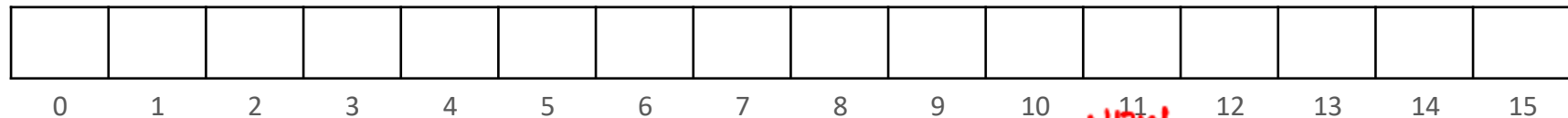
Sample location is

$$x = \frac{2}{6}(5) = 1.667$$

Resampling pixel locations

Consider a 16-pixel 1D image. You are asked to resize it to a 5-pixel 1D image. What is the location of pixel 2 (between 0 and 15) of the new image?

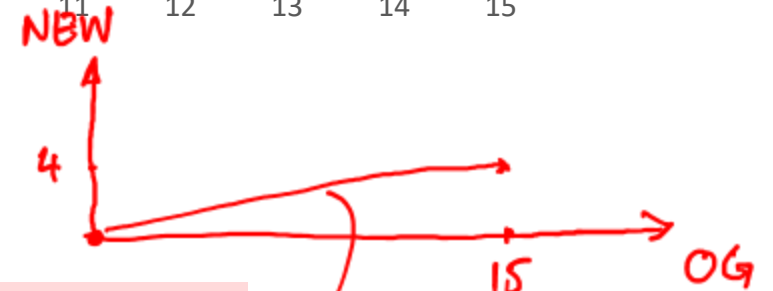
Original image



Resized image



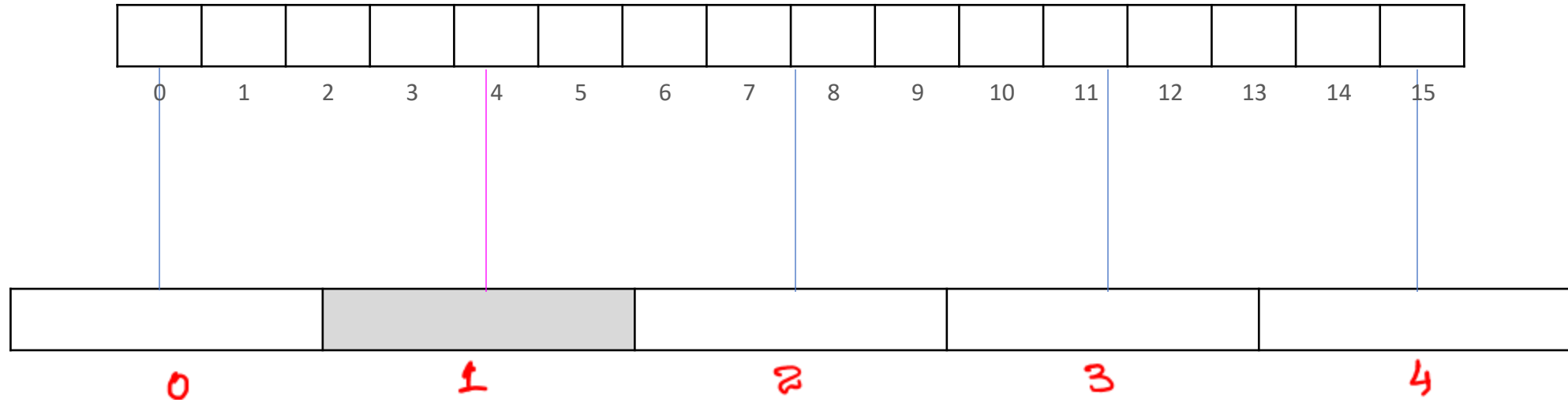
Downscaling



$$y = mx + b$$
$$\Rightarrow y = \frac{4}{15}x$$
$$\Rightarrow x = \frac{15}{4}x$$

Resampling pixel locations

Consider a 16-pixel 1D image. You are asked to resize it to a 5-pixel 1D image. What is the location of pixel 2 (between 0 and 15) of the new image?



Given

Last pixel location in original image = 15
Last pixel location in resulting image = 4

OG = $\frac{15}{4}$ NEW

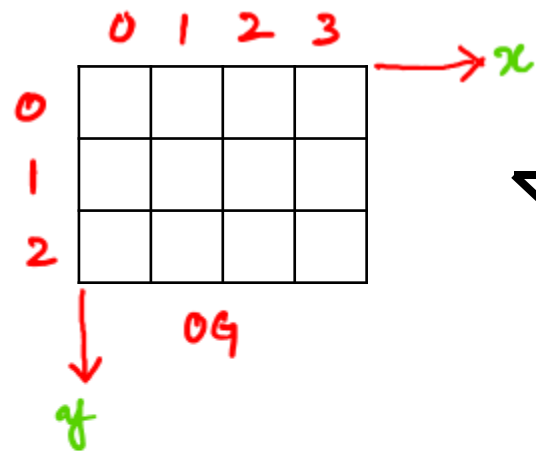
Use the relationship developed earlier

$$x = \frac{15}{4} y$$

Sample location is

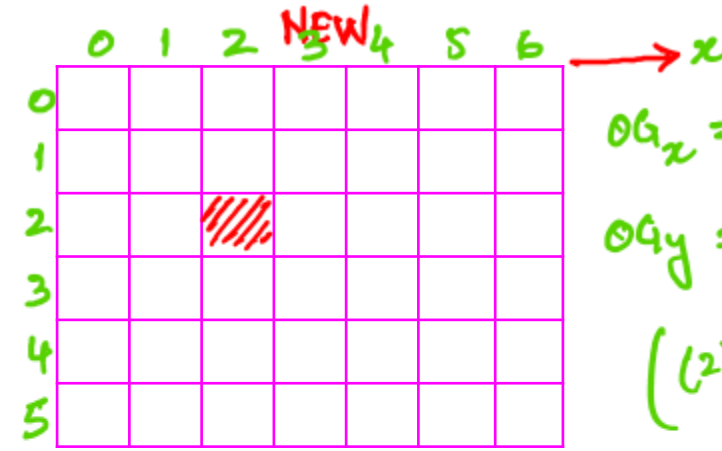
$$x = \frac{15}{4} (1) = 3.75 \approx 4$$

Resampling pixel locations (in 2D)



Enlarge the image

Reduce the image



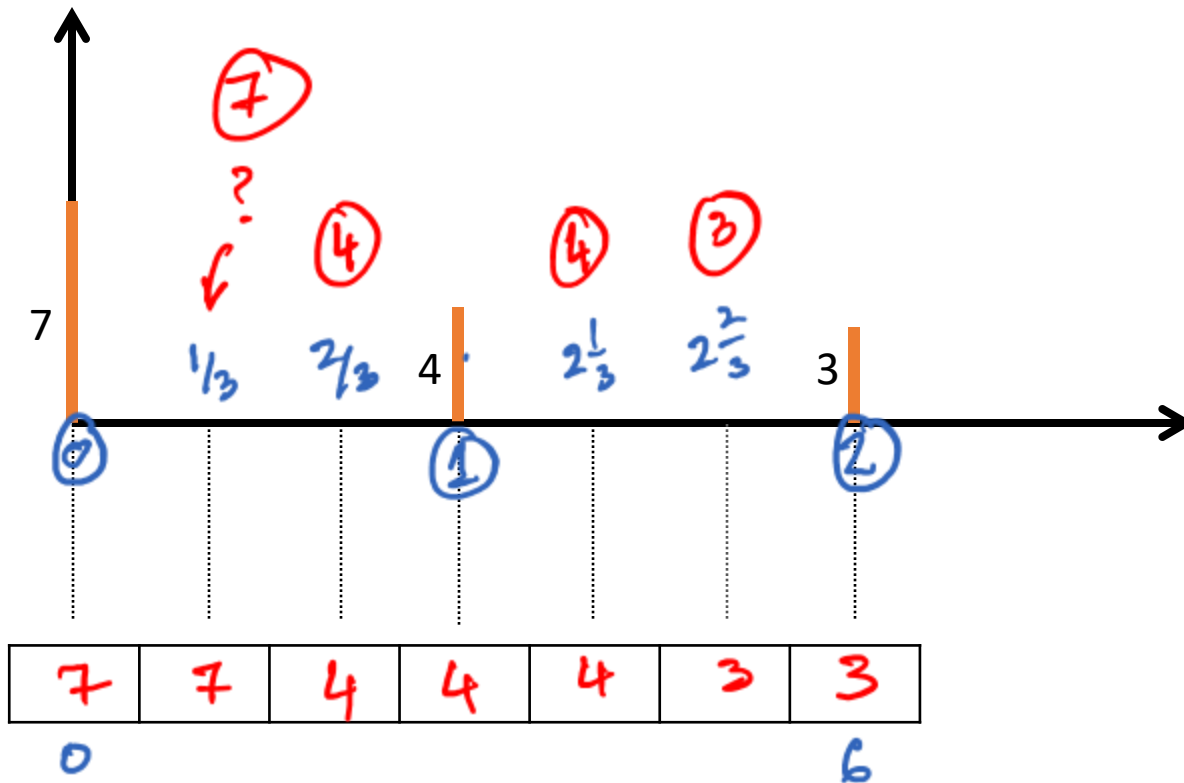
$$OG_x = \frac{3}{6} NEW_x$$

$$OG_y = \frac{2}{5} NEW_y$$

$$\left((2) \left(\frac{3}{6} \right), (2) \left(\frac{2}{5} \right) \right)$$



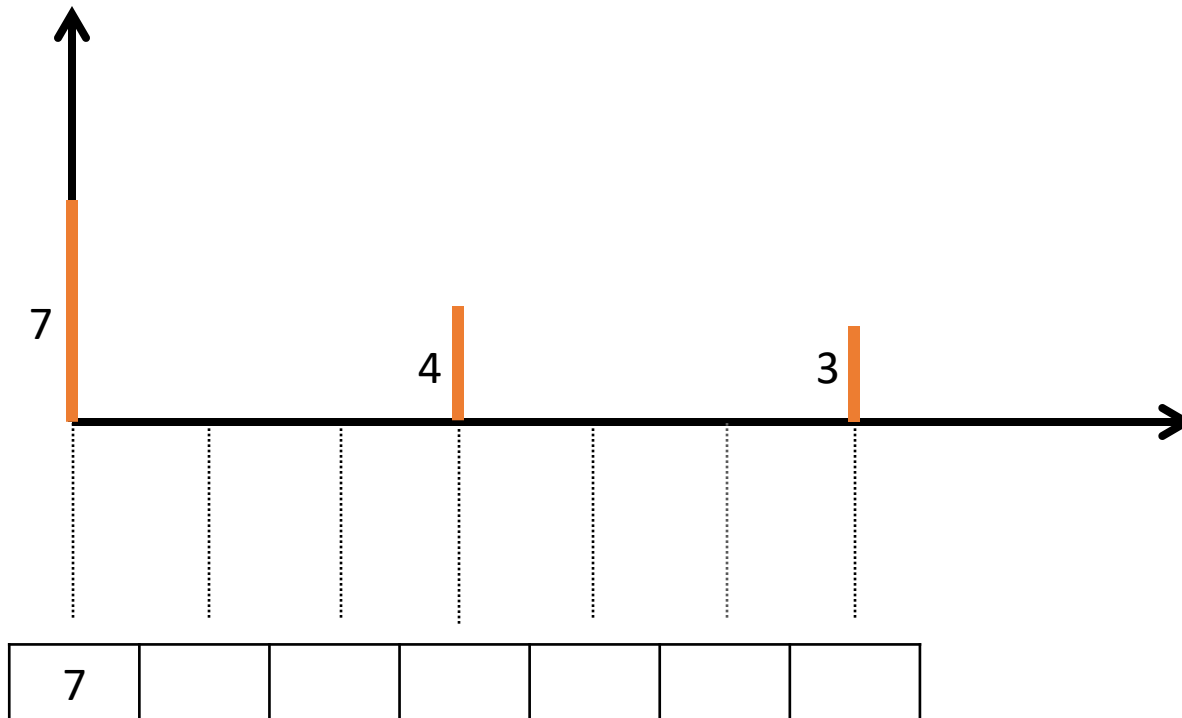
How to compute new pixel values?



$$O_9 = \frac{1}{3} \text{ NEW}$$

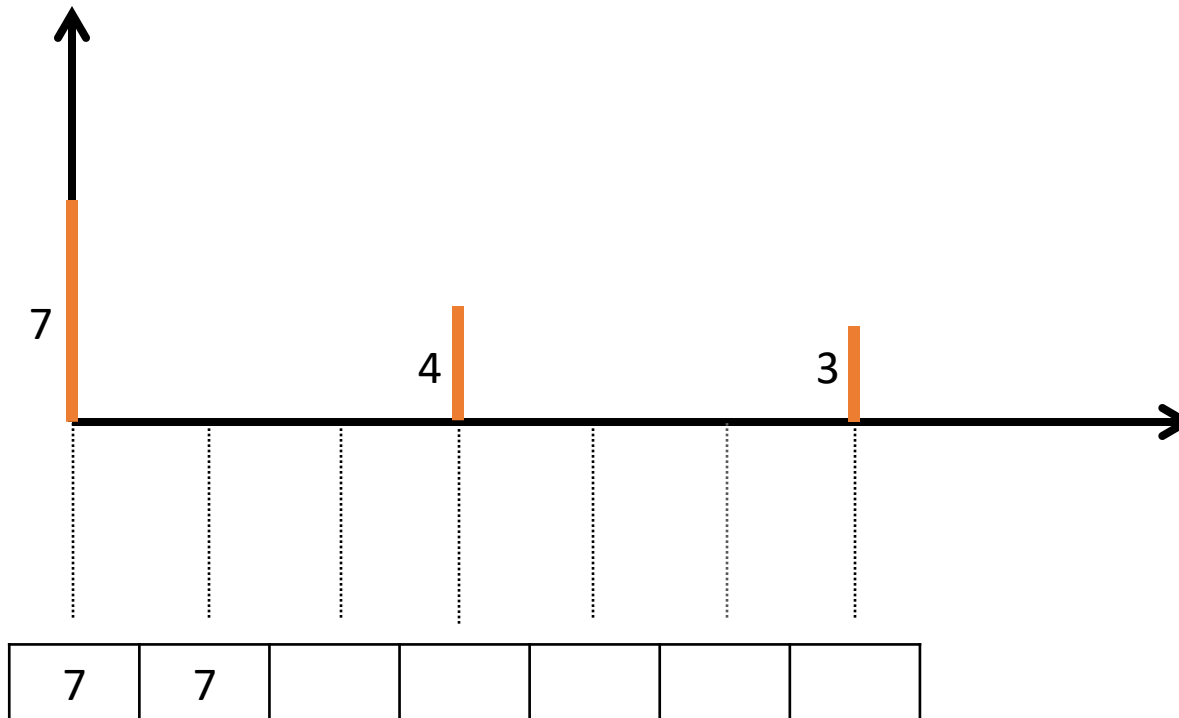
$|\frac{1}{3} - 0| \rightarrow$ Nearest neighbour
 ~~$|\frac{1}{3} - 1|$~~
 ~~$|\frac{1}{3} - 2|$~~

How to compute new pixel values?



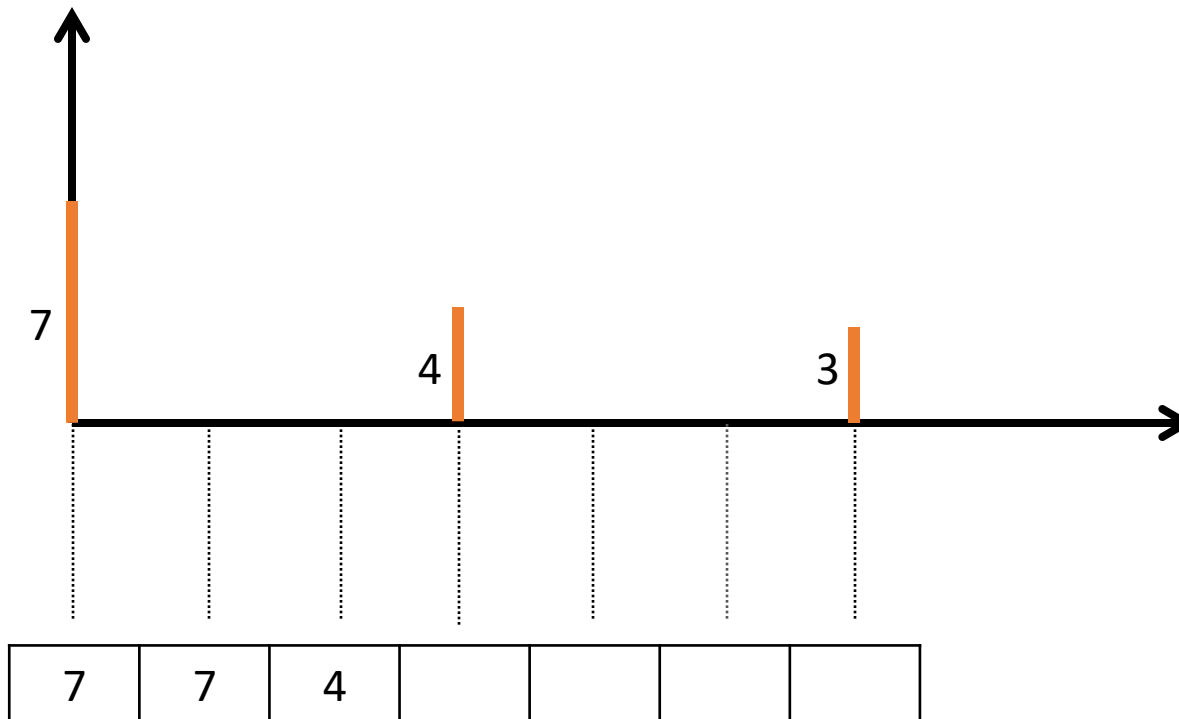
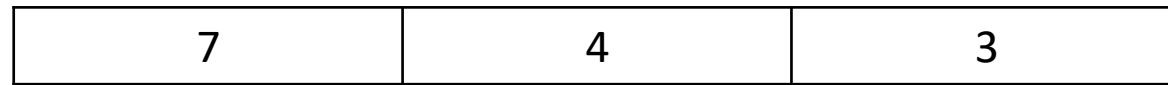
Nearest-Neighbor Interpolation

How to compute new pixel values?



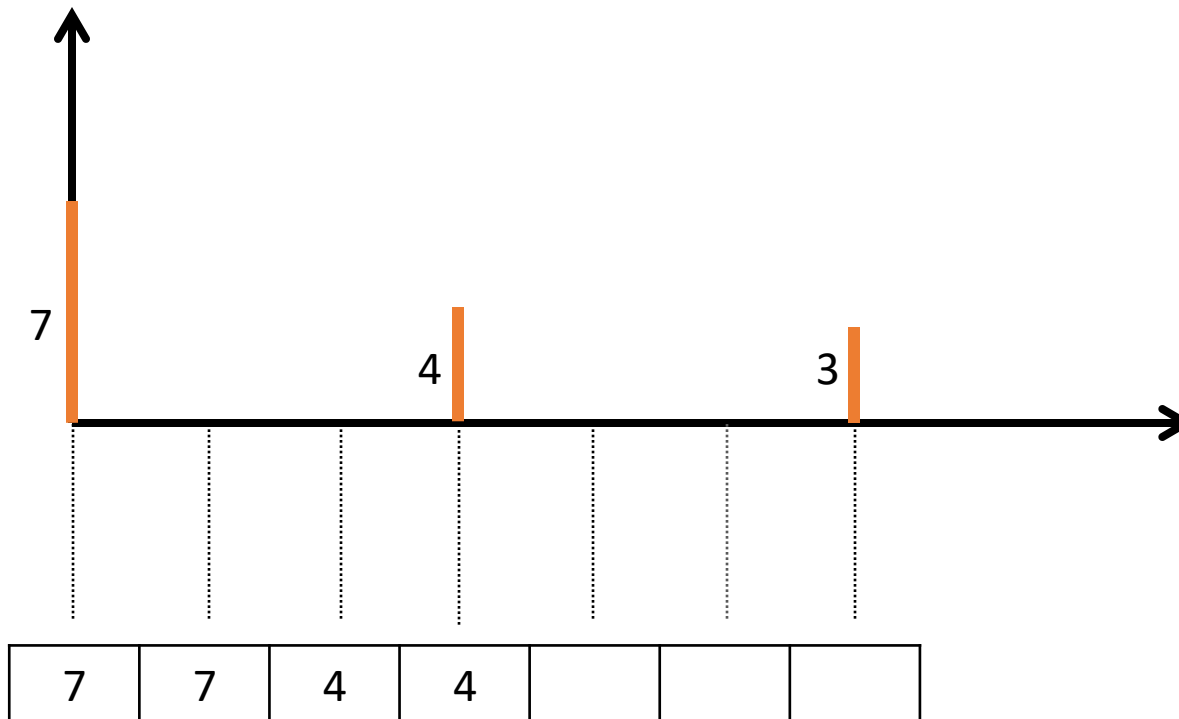
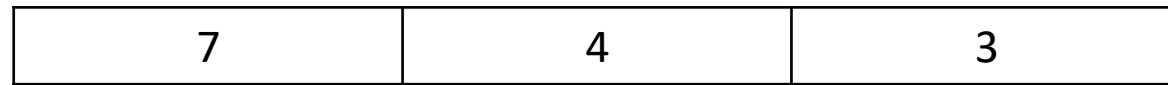
Nearest-Neighbor Interpolation

How to compute new pixel values?



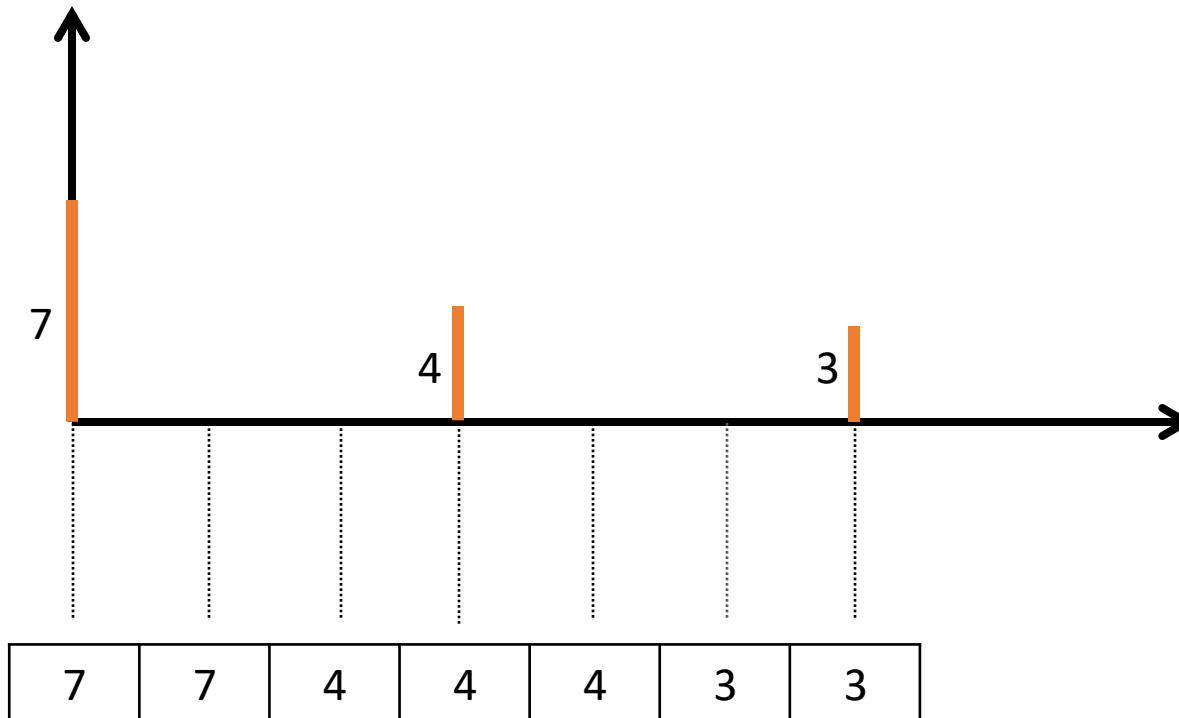
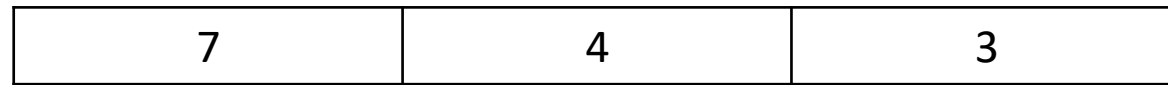
Nearest-Neighbor Interpolation

How to compute new pixel values?



Nearest-Neighbor Interpolation

How to compute new pixel values?



Nearest-Neighbor Interpolation

Nearest-Neighbor Interpolation

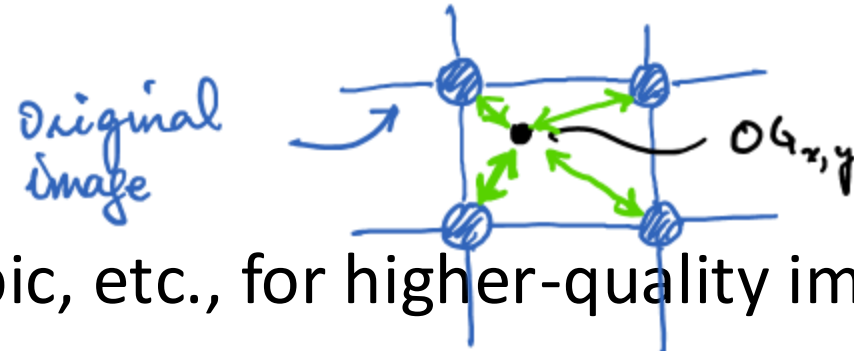
- Easy to implement.
- Results in blocky or pixelated results
- Does not consider neighboring pixels
- Losses details and smoothness
- Use other methods, e.g., bilinear, bicubic, etc., for higher-quality image resizing

Input image: $H \times W$

Output image: $H' \times W'$

$$OG_{x,y} = \text{Map}(NEW_{x,y})$$

$$\in [0, W-1] \times [0, H-1] \quad \in [0, W'-1] \times [0, H'-1]$$



$$d = \sqrt{(x_m - x)^2 + (y_m - y)^2}$$

Nearest-Neighbor Interpolation

2592 x 1944 px

Original



480 x 360 px

Nearest-Neighbor



Nearest-Neighbor Interpolation

480 x 360 px

Bilinear

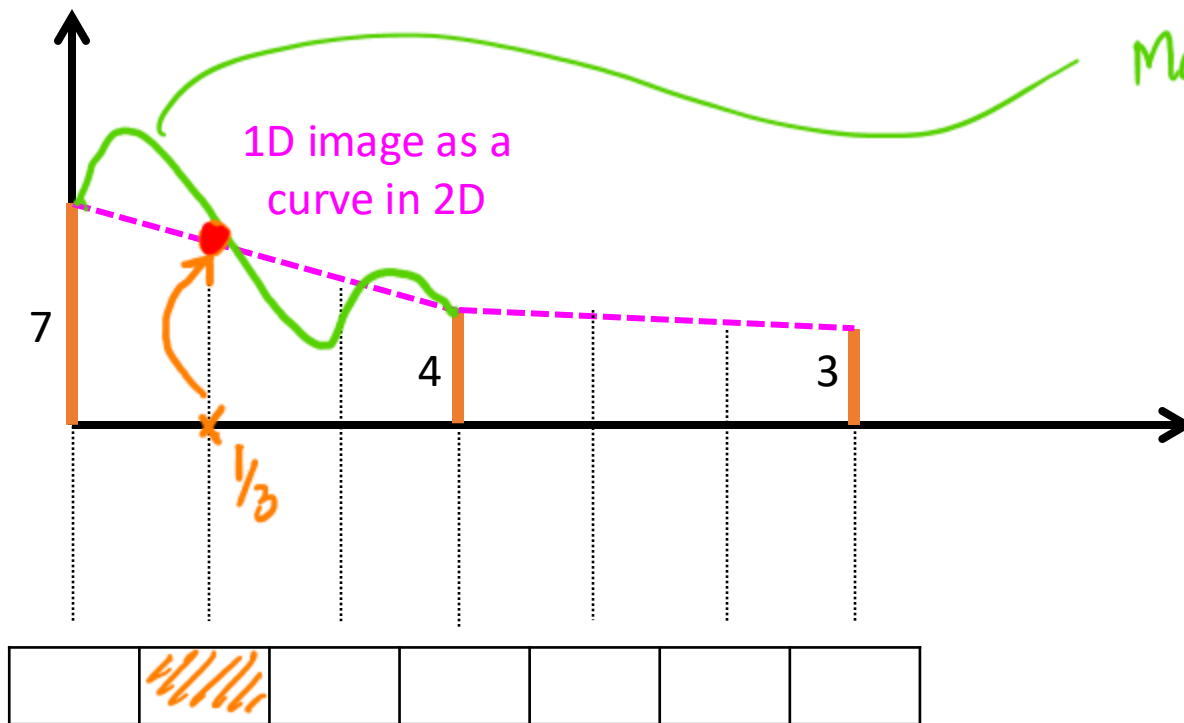


480 x 360 px

Nearest-Neighbor

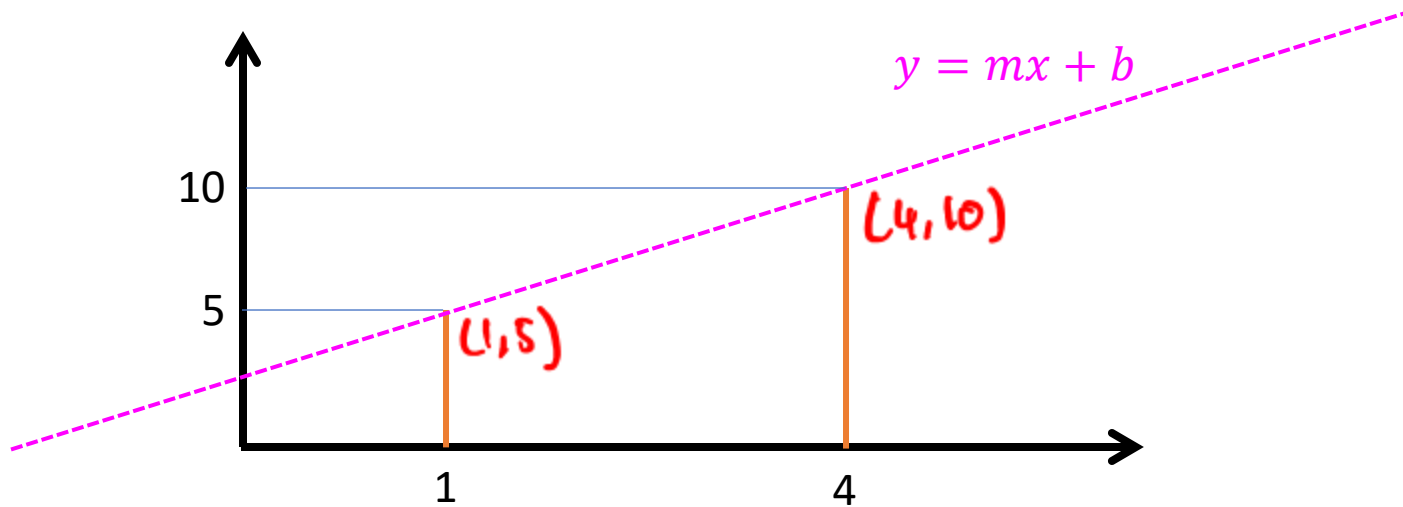


Linear Interpolation



Makes no sense to assume such a function!!

2D Line Fitting



2D Line Fitting

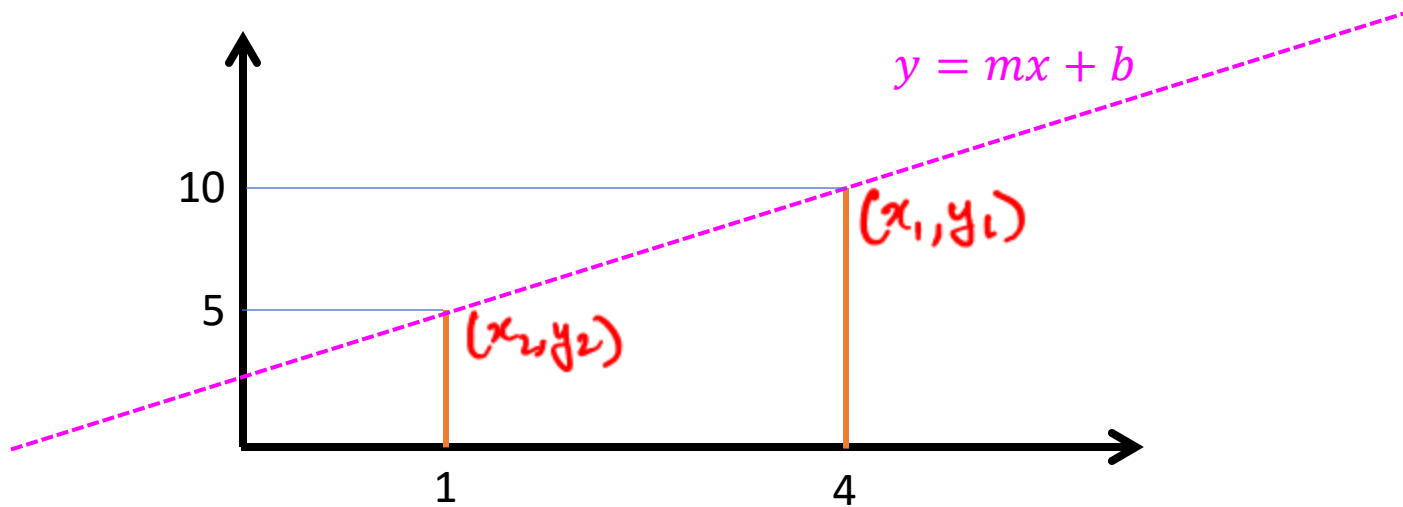
A line between (x_1, y_1) and (x_2, y_2) is given by:

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

↓

$$y = mx + b$$

2D Line Fitting



Output

2D Line Fitting

A line between (x_1, y_1) and (x_2, y_2) is given by:

Matrix form

Re-write

$$x_1 m + b = y_1 \longrightarrow \text{all lines that pass } (x_1, y_1)$$

$$x_2 m + b = y_2$$

as

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

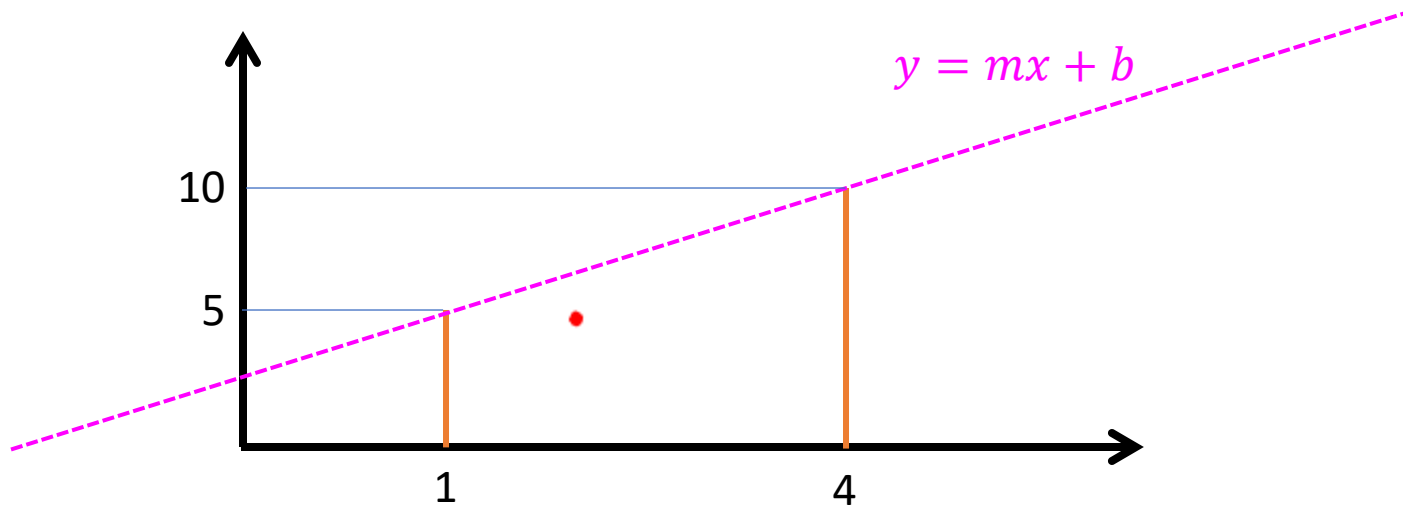
System of linear equations.

$$\begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

where

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

2D Line Fitting



A line between (x_1, y_1) and (x_2, y_2) is given by:

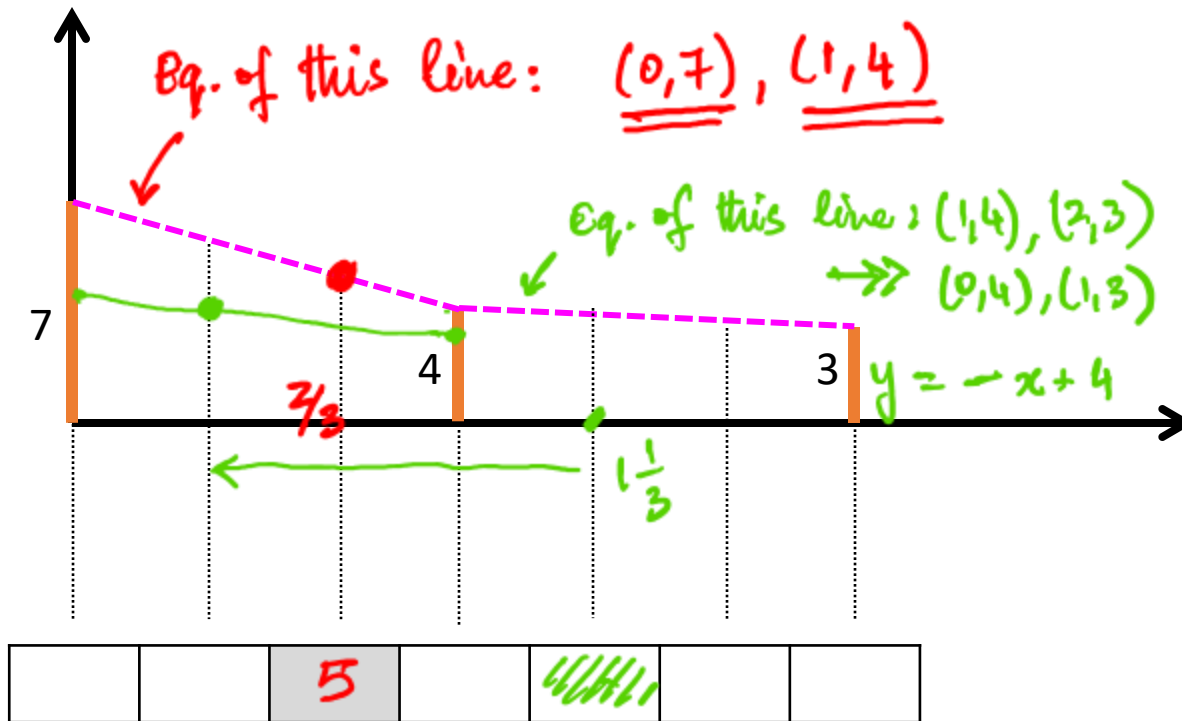
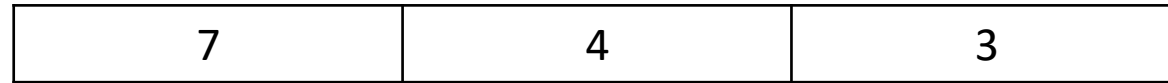
$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

cannot work for more than two pts.

Practice Question

Estimate (fit) the dotted-line shown on the left.

Linear Interpolation



Example

Compute the value for shaded pixel?

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

$$\Rightarrow \frac{y - 7}{4 - 7} = \frac{x - 0}{1 - 0}$$

$$\Rightarrow y - 7 = -3x$$

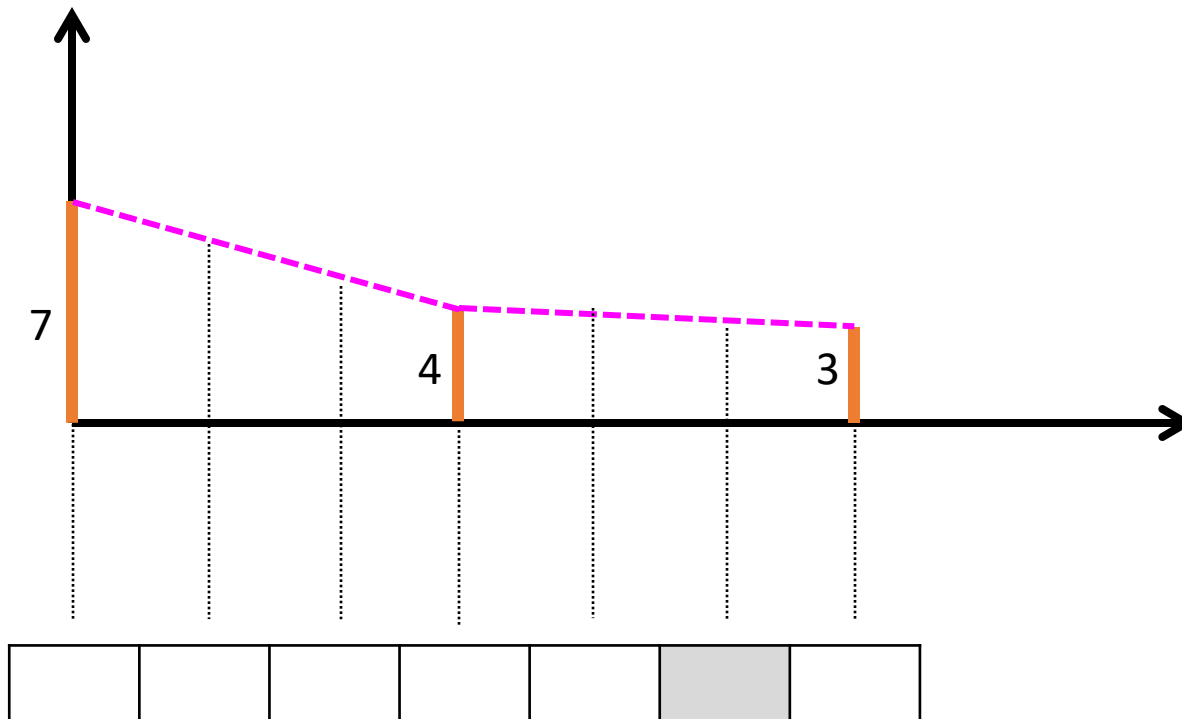
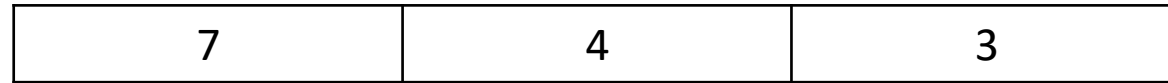
$$\Rightarrow \boxed{y = -3x + 7}$$

$$y = -3\left(\frac{2}{3}\right) + 7$$

$$= -2 + 7$$

$$= 5$$

Linear Interpolation



Practice Question

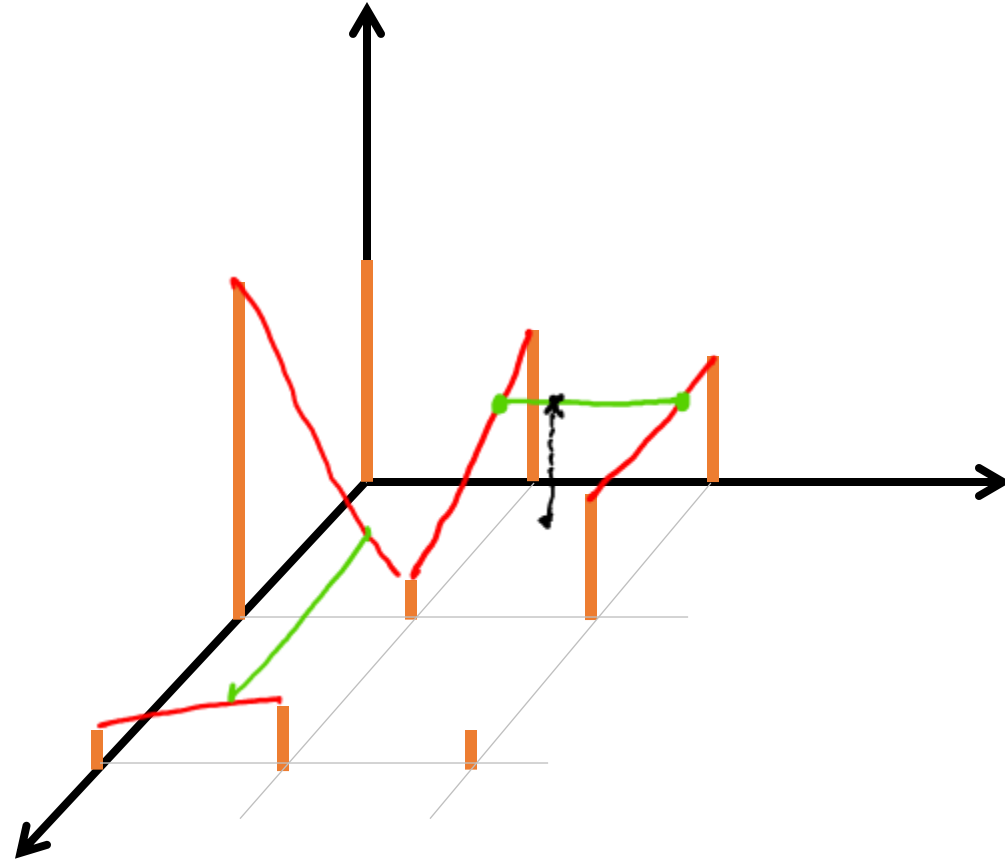
Compute the value for shaded pixels?

Images as surfaces

Images are not just 1D. How do we deal with a 2D image?

Consider the following 3x3 2D image.

7	4	3
9	1	3
1	2	1

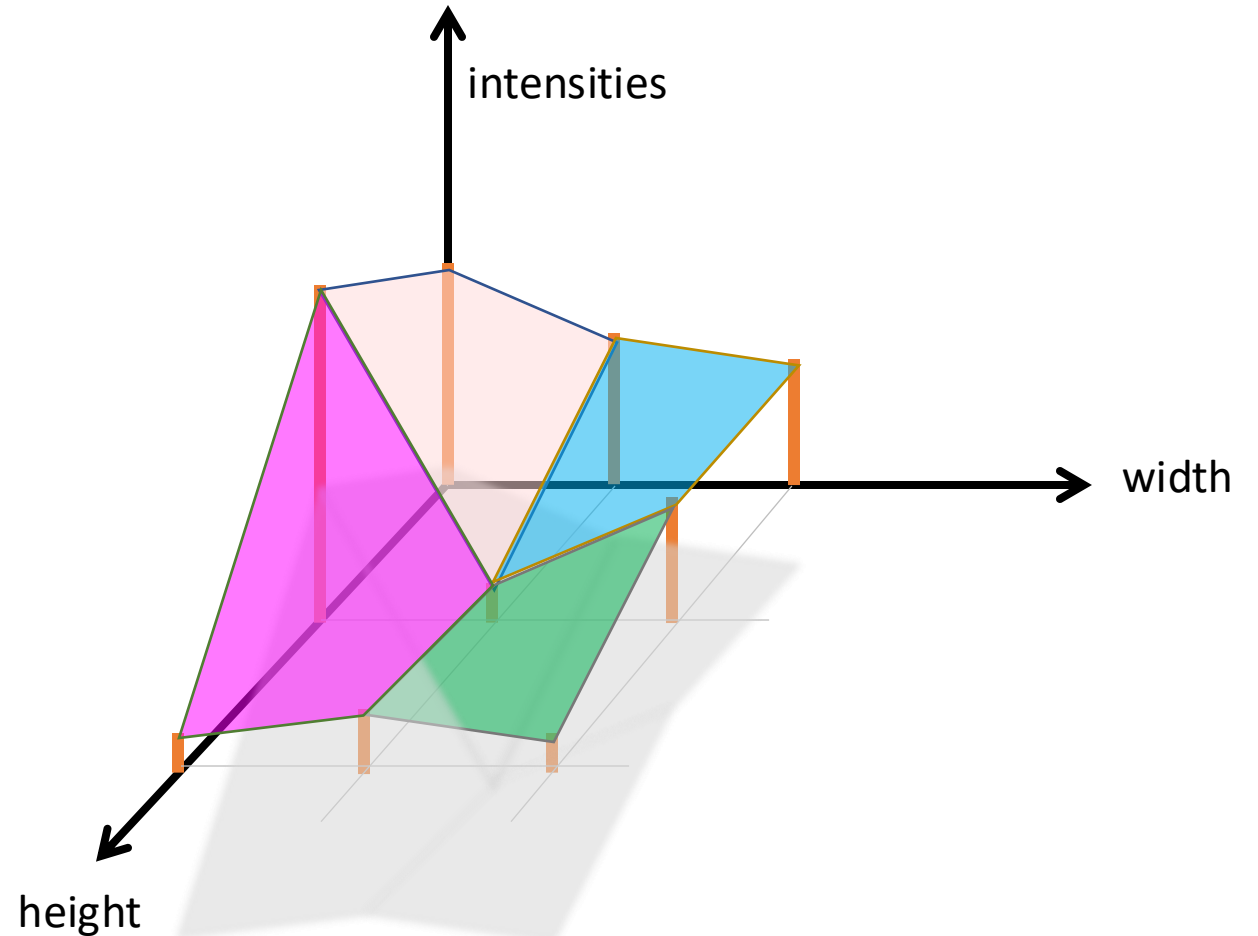


Images as surfaces

Images are not just 1D. How do we deal with a 2D image?

Consider the following 3x3 2D image.

7	4	3
9	1	3
1	2	1

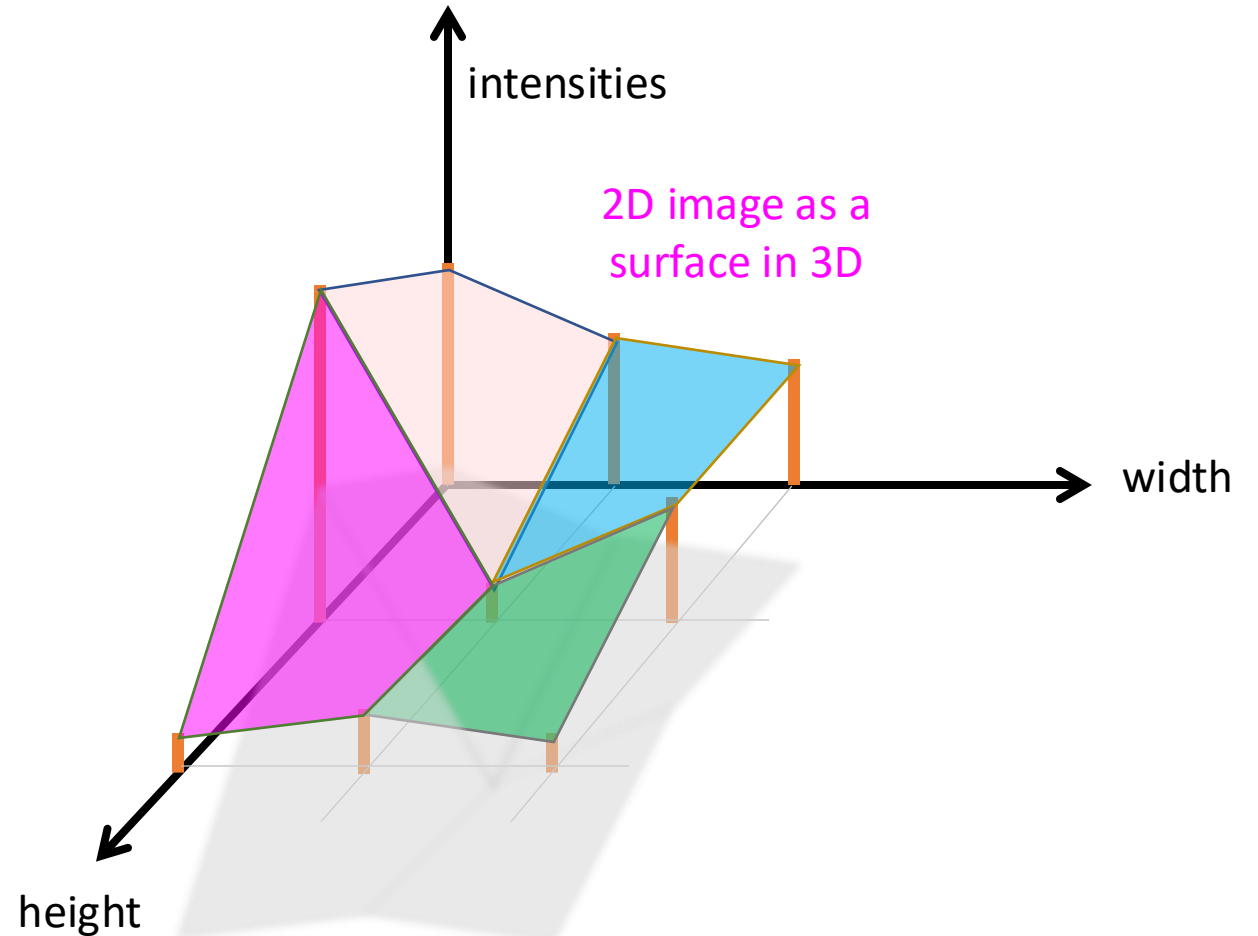


Images as surfaces

Images are not just 1D. How do we deal with a 2D image?

Consider the following 3x3 2D image.

7	4	3
9	1	3
1	2	1



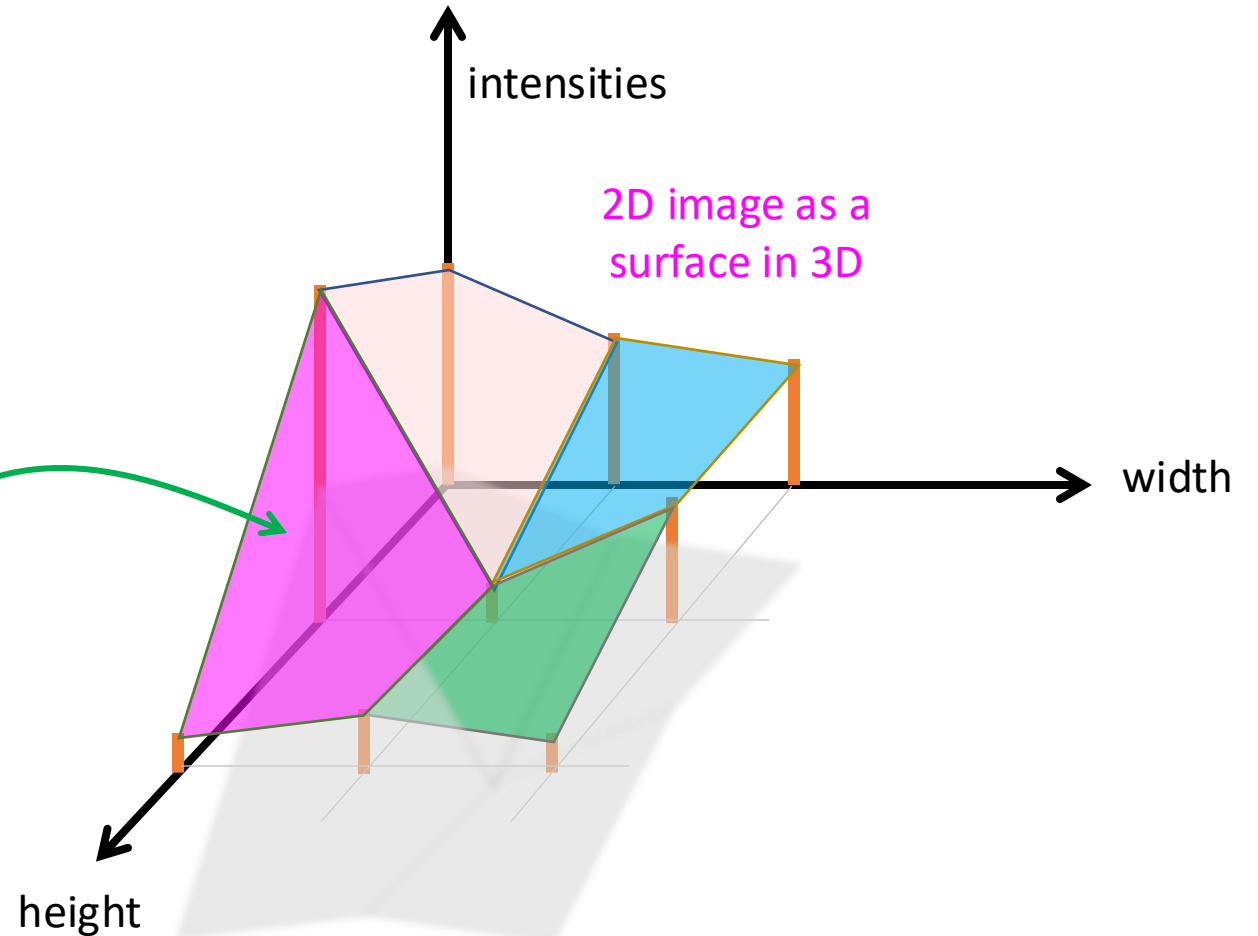
Images as surfaces

Images are not just 1D. How do we deal with a 2D image?

Consider the following 3x3 2D image.

7	4	3
9	1	3
1	2	1

Are these planar patches?



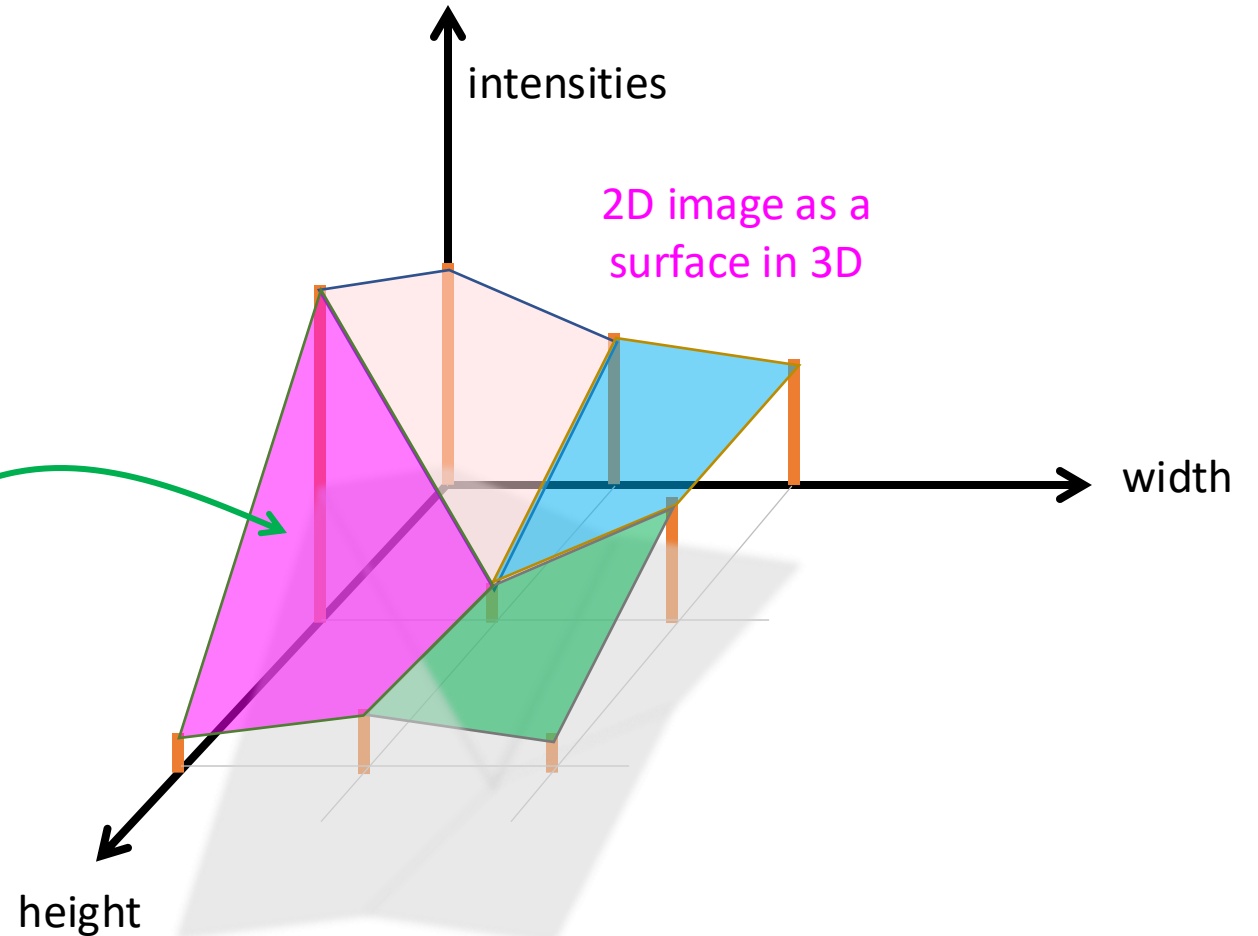
Images as surfaces

Images are not just 1D. How do we deal with a 2D image?

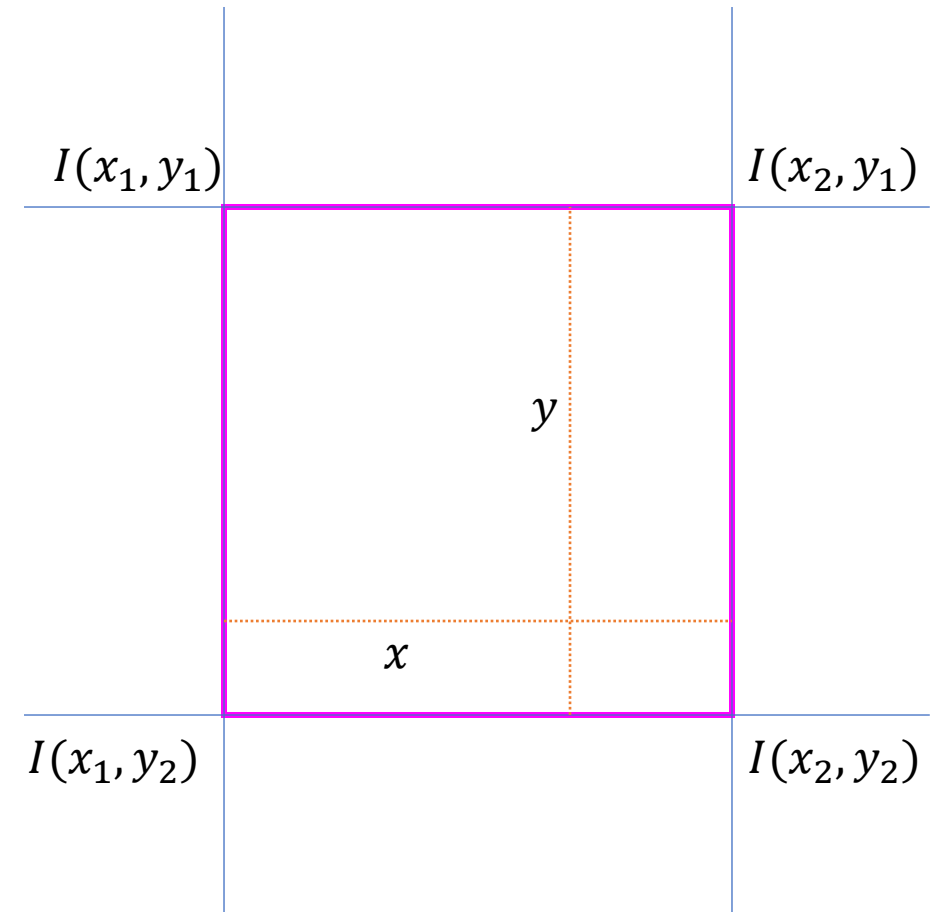
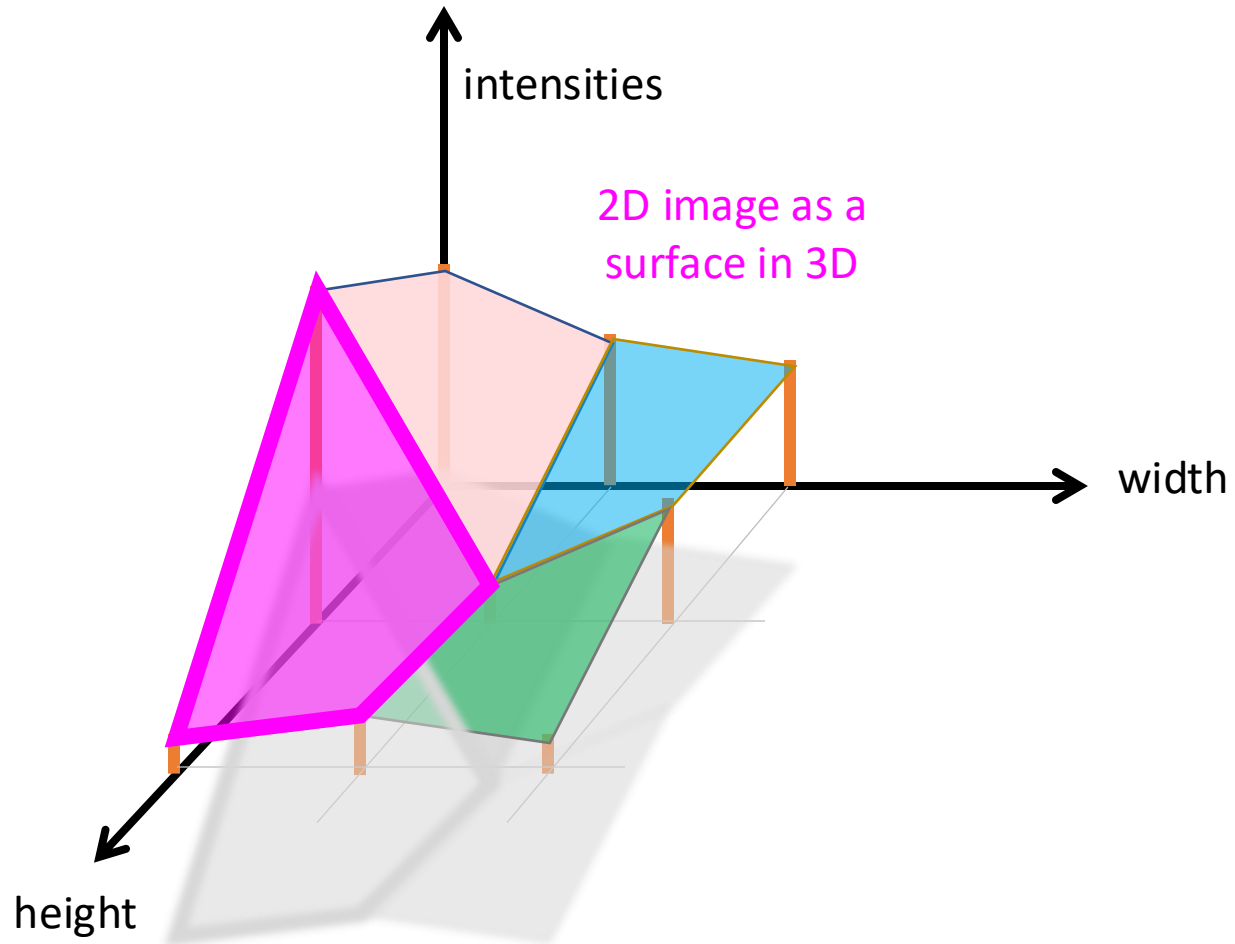
Consider the following 3x3 2D image.

7	4	3
9	1	3
1	2	1

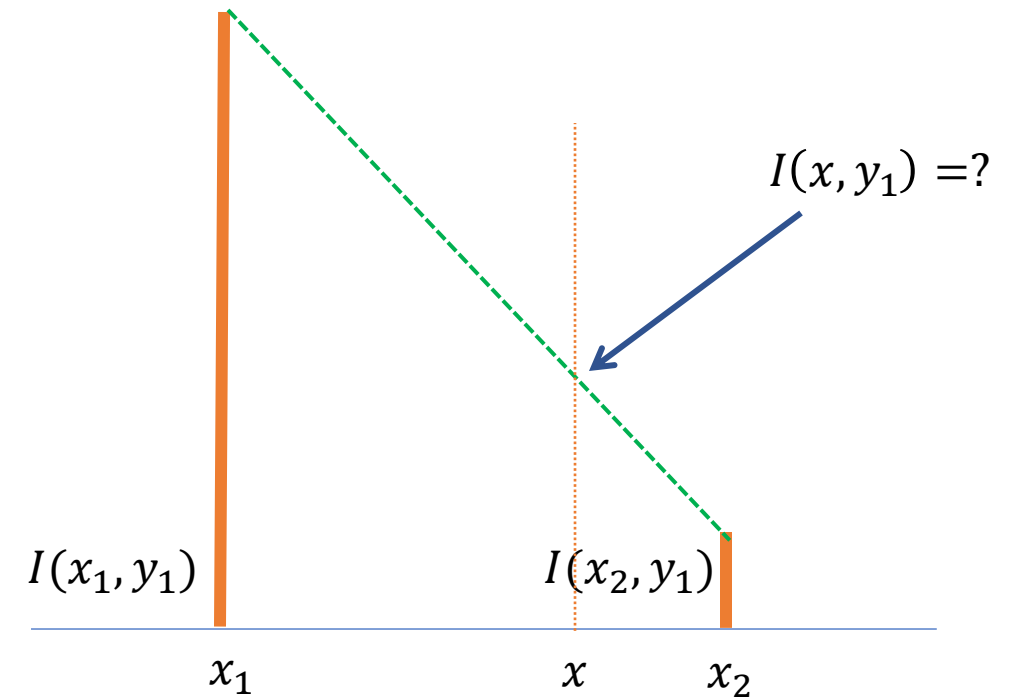
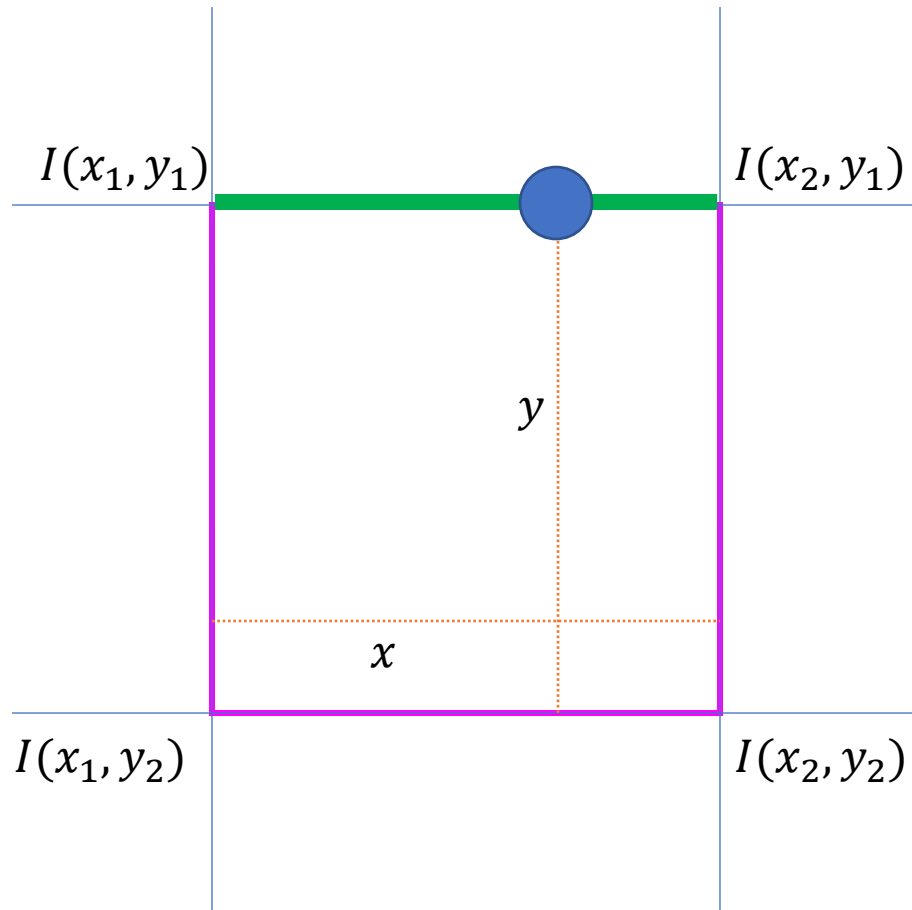
Are these planar patches? **No**



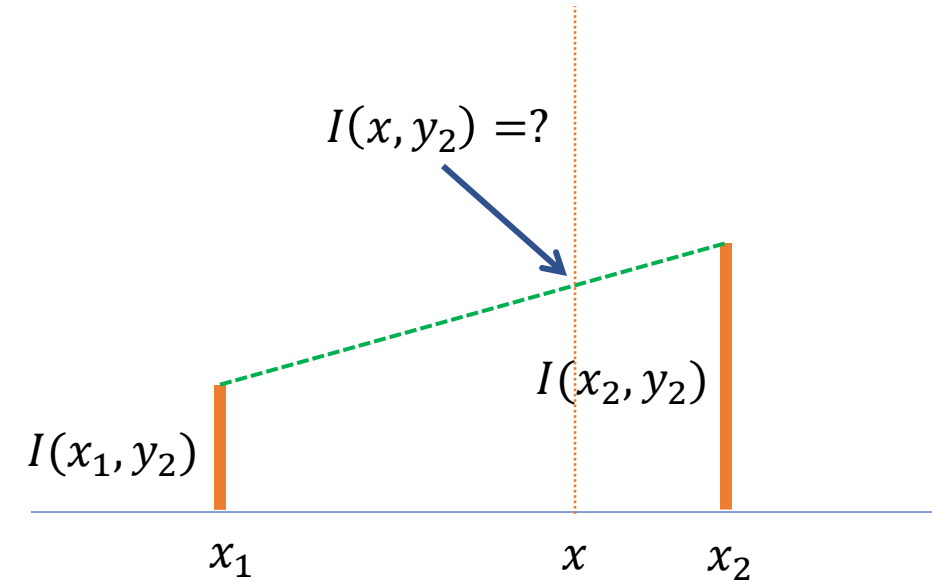
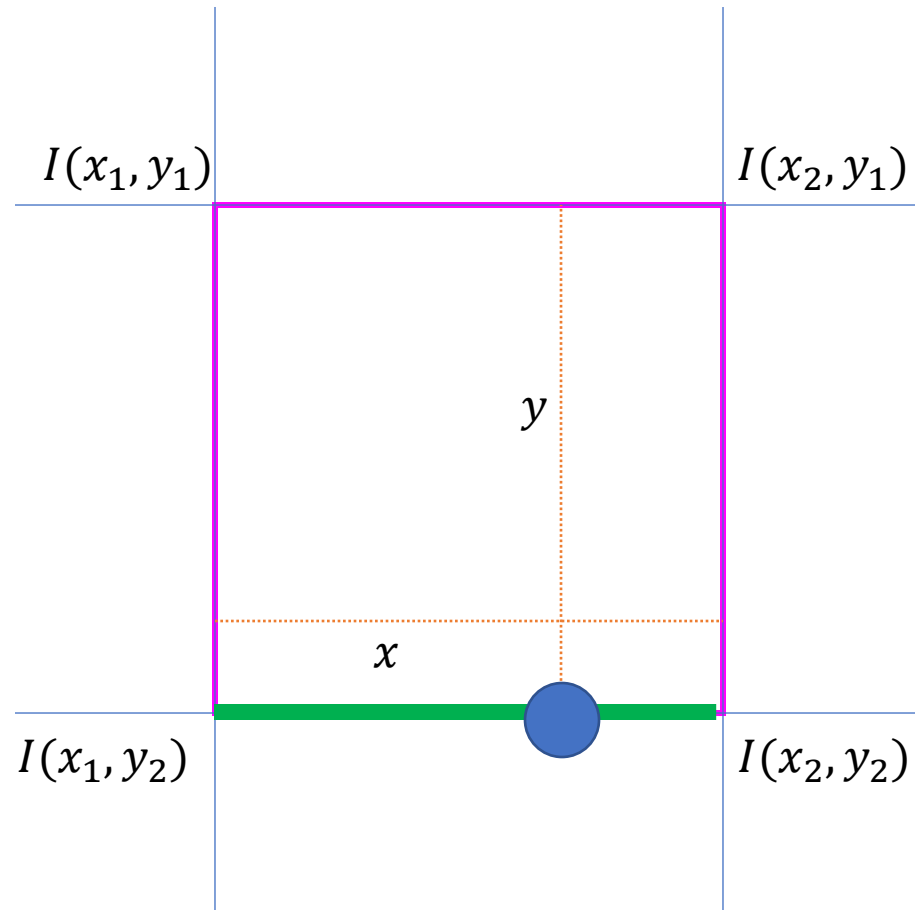
Images as surfaces



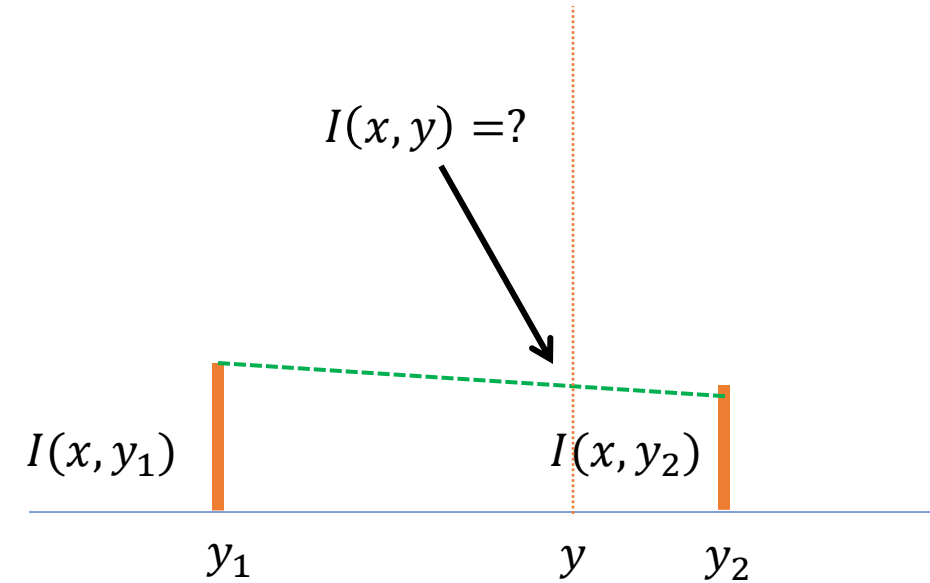
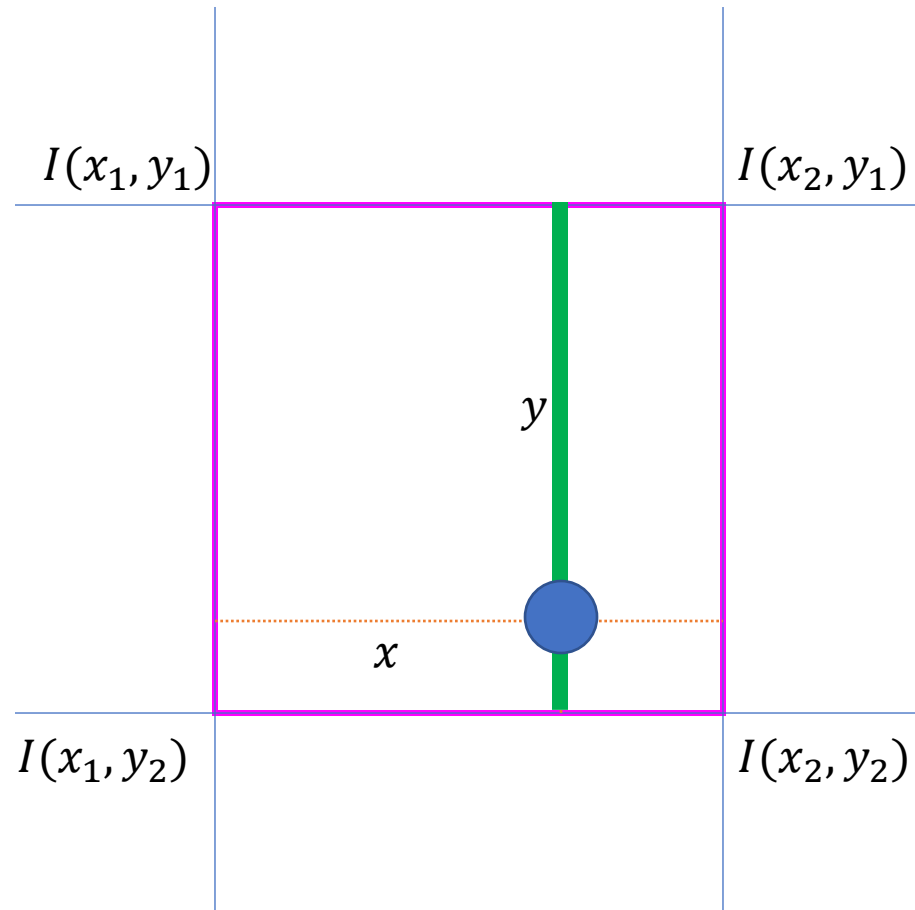
Bi-linear interpolation



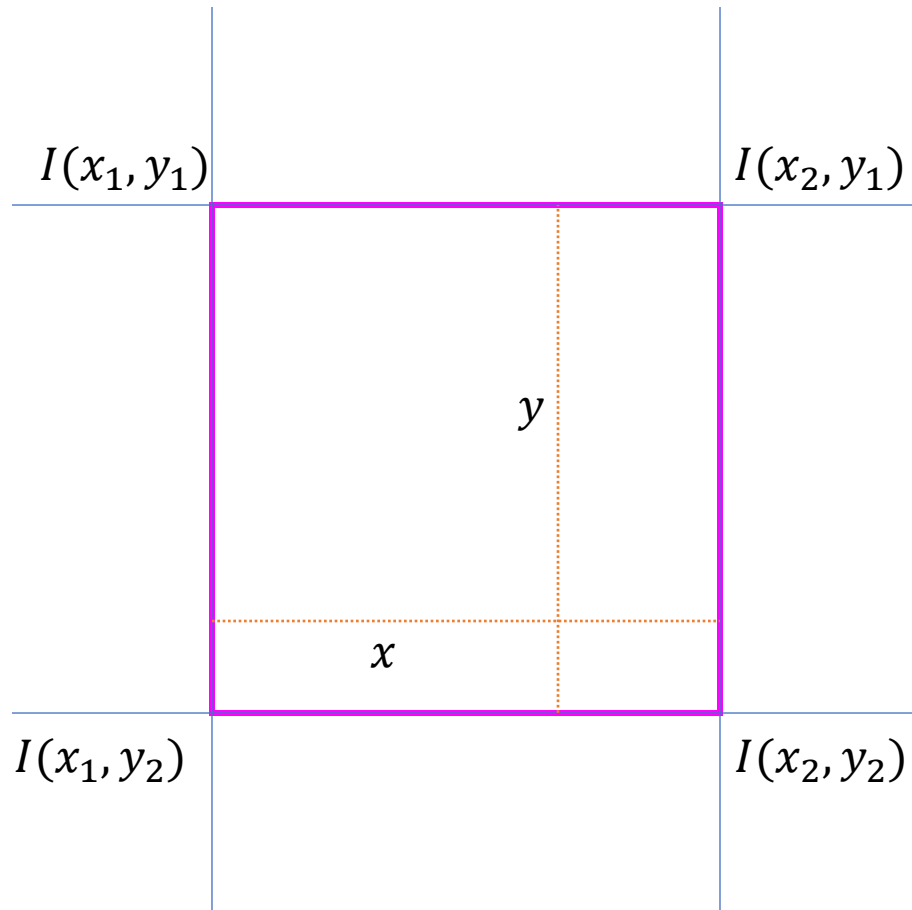
Bi-linear interpolation



Bi-linear interpolation



Bi-Linear interpolation



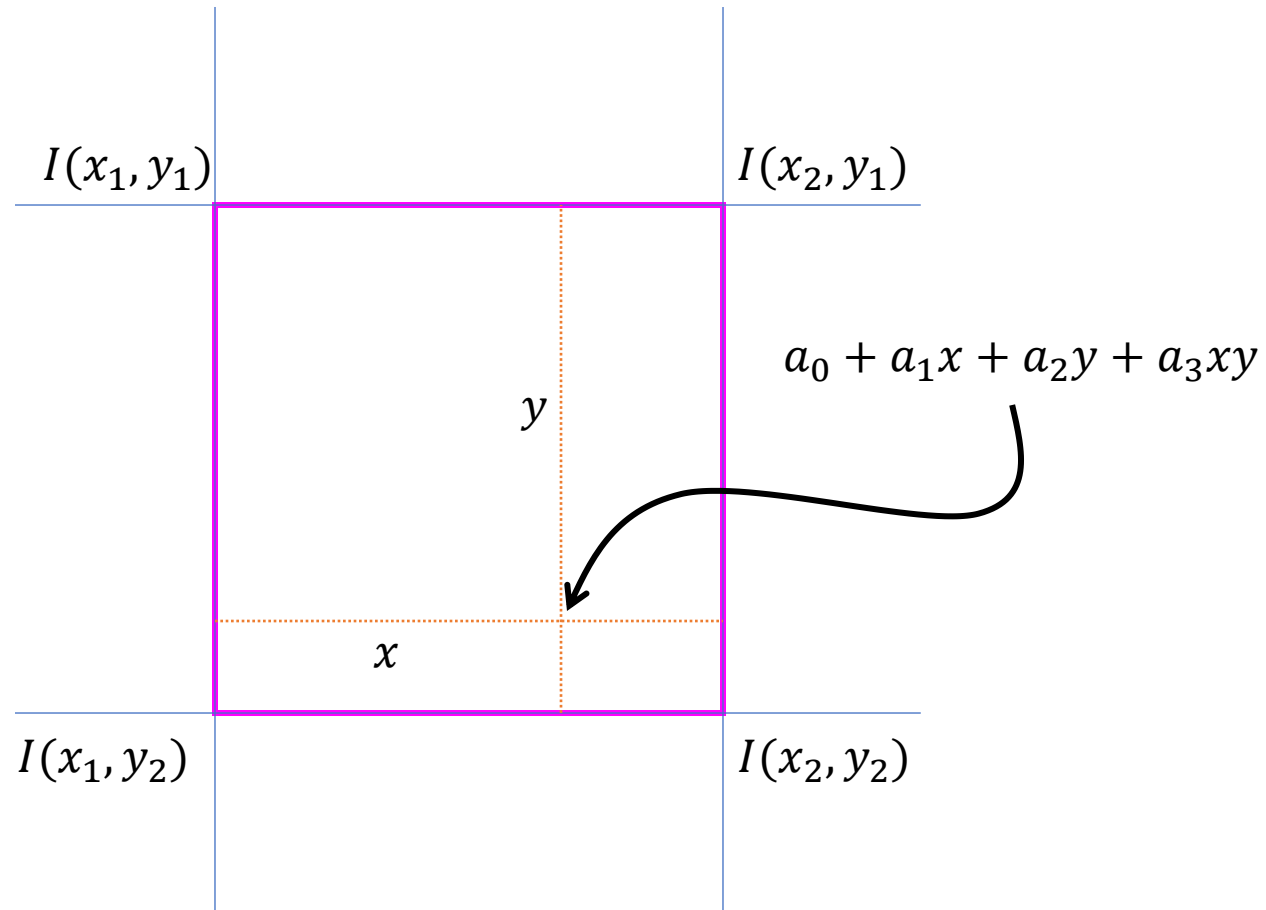
Multi-linear polynomial

$$f(x, y) = a_0 + a_1x + a_2y + a_3xy$$

Then for $i, j \in [1, 2]$

$$I(x_i, y_j) = a_0 + a_1x_i + a_2y_j + a_3x_i y_j$$

Bi-Linear interpolation



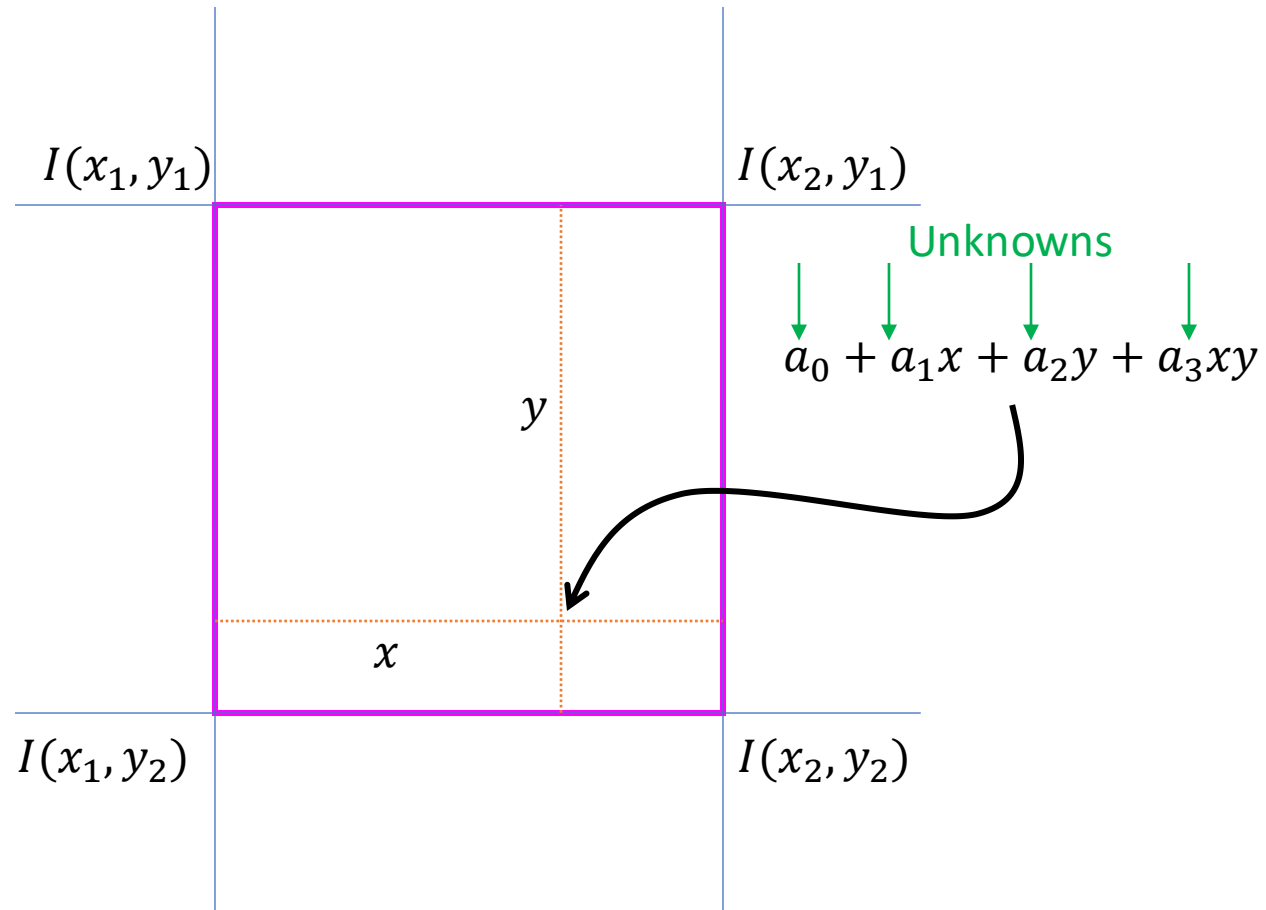
Multi-linear polynomial

$$f(x, y) = a_0 + a_1x + a_2y + a_3xy$$

Then for $i, j \in [1, 2]$

$$I(x_i, y_j) = a_0 + a_1x_i + a_2y_j + a_3x_i y_j$$

Bi-Linear interpolation



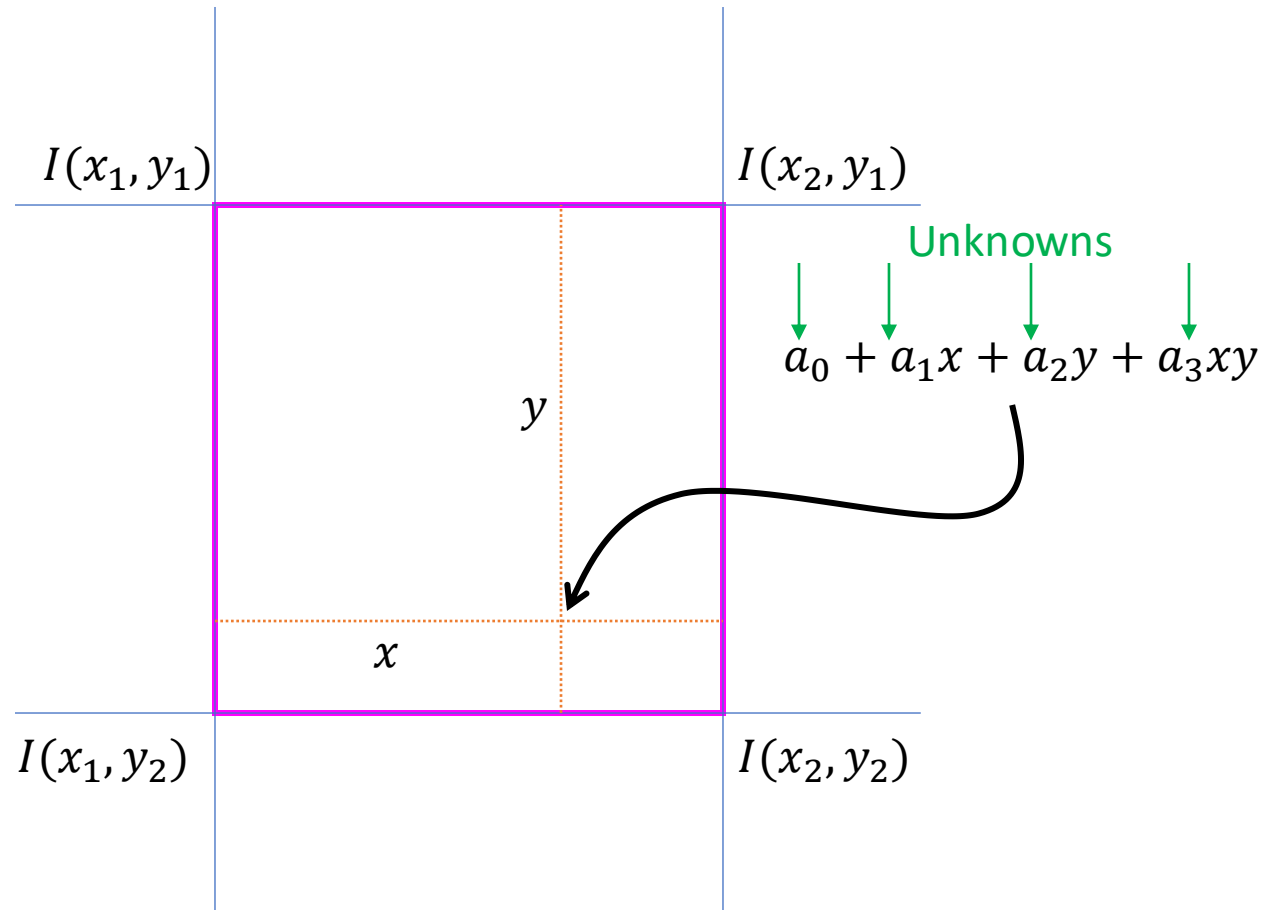
Multi-linear polynomial

$$f(x, y) = a_0 + a_1x + a_2y + a_3xy$$

Then for $i, j \in [1, 2]$

$$I(x_i, y_j) = a_0 + a_1x_i + a_2y_j + a_3x_i y_j$$

Bi-Linear interpolation



Multi-linear polynomial

$$f(x, y) = a_0 + a_1x + a_2y + a_3xy$$

Then for $i, j \in [1, 2]$

$$I(x_i, y_j) = a_0 + a_1x_i + a_2y_j + a_3x_i y_j$$

Solve for the unknowns using the following system of equations

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_1 & y_2 & x_1y_2 \\ 1 & x_2 & y_2 & x_2y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} I(x_1, y_1) \\ I(x_2, y_1) \\ I(x_1, y_2) \\ I(x_2, y_2) \end{bmatrix}$$

Bilinear interpolation: Pros

- Smoothing Effect, which helps reduce jagged edges and pixelation.
- Simple to Implement, requires fewer calculation and computational inexpensive as compared to other methods
- Maintains linearity between the known data points, which can be desirable in certain applications, such as computer graphics.

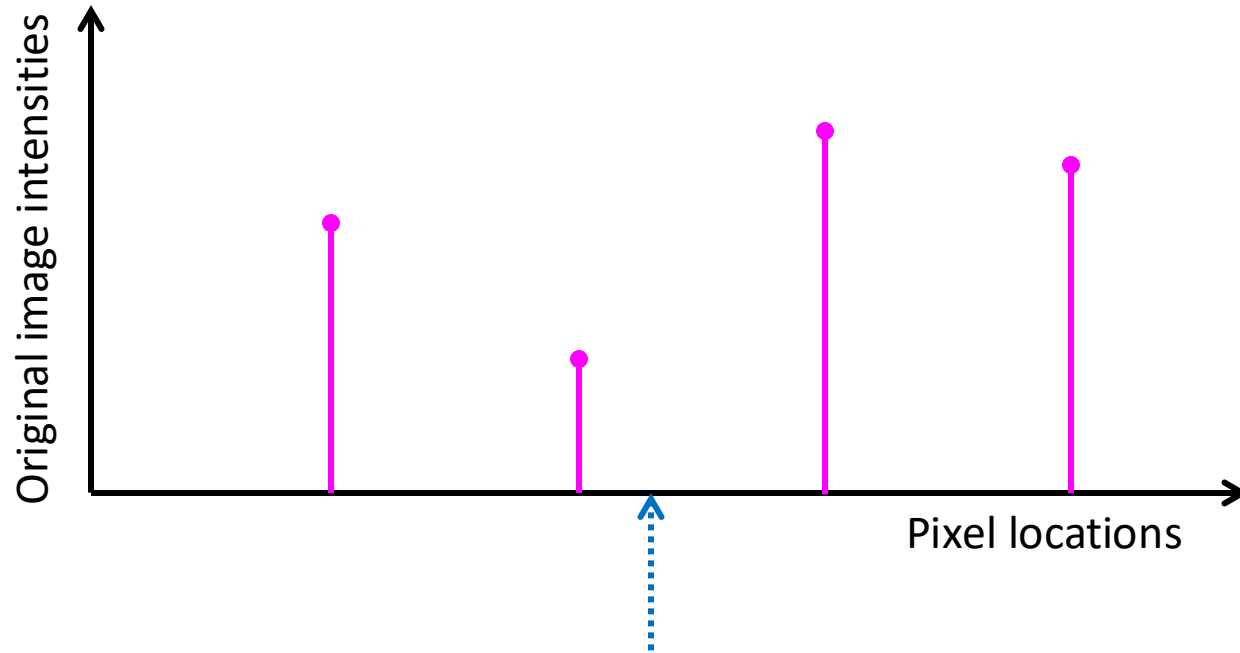
Bilinear interpolation: Cons

- Loss of sharpness and fine details
- Color artifacts
- No consideration for high-frequency components
 - Not suitable for images with intricate patterns or textures
- Not ideal for large scaling
- Limited accuracy and it may not be suitable for photometric applications

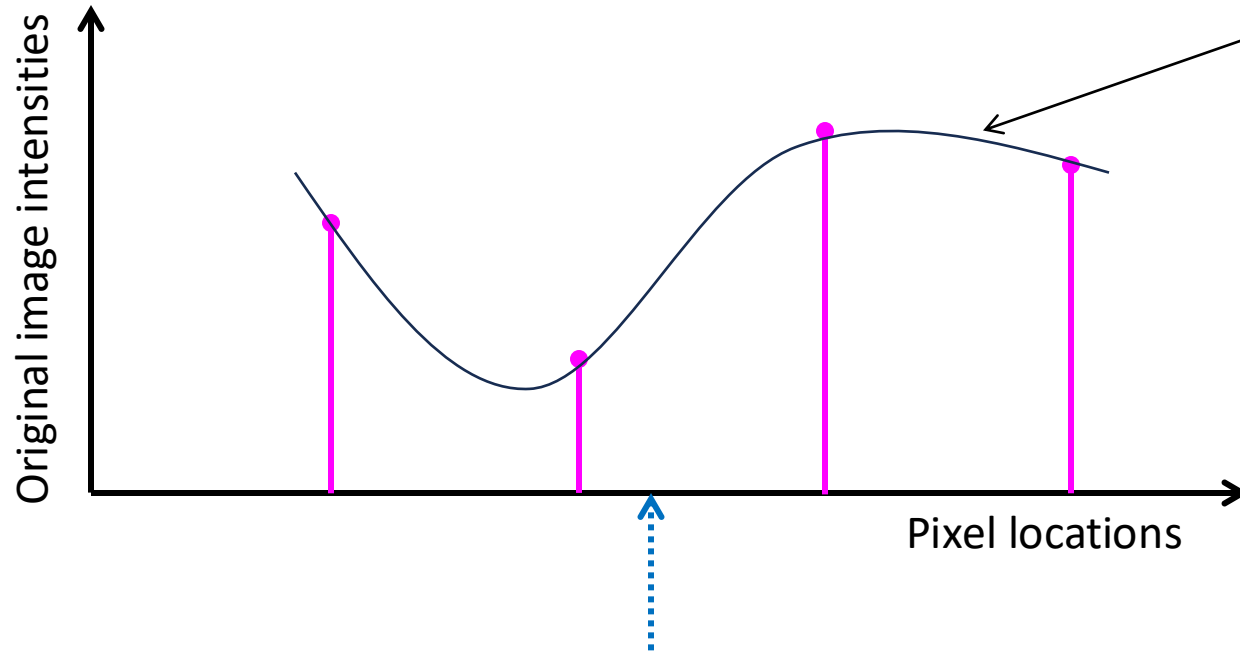
Bicubic interpolation

- Bicubic interpolation is a method for image resizing that calculates new pixel values using the nearest 16 pixels (a 4x4 grid).
- It produces smoother and higher-quality results compared to simpler methods like nearest-neighbor and bilinear interpolation.

Bicubic Interpolation (in 1D)



Cubic Interpolation (in 1D)



Approximate local structure using a cubic polynomial

$$f(x) = ax^3 + bx^2 + cx + d$$

This equation has four unknowns, so we need at least four points to fit this model (to the available image intensities)

Bicubic interpolation: Pros

- Better Image Quality:
 - Reduces jagged edges and pixelation, handling edges and gradients effectively.
- Improved Detail Preservation:
 - Retains fine details, ideal for upscaling images.
- Smooth Transitions:
 - Minimizes artifacts like sudden intensity changes, providing a natural look.
- Widely Used:
 - Implemented in many image processing tools, making it a well-established method.

Bicubic interpolation: Cons

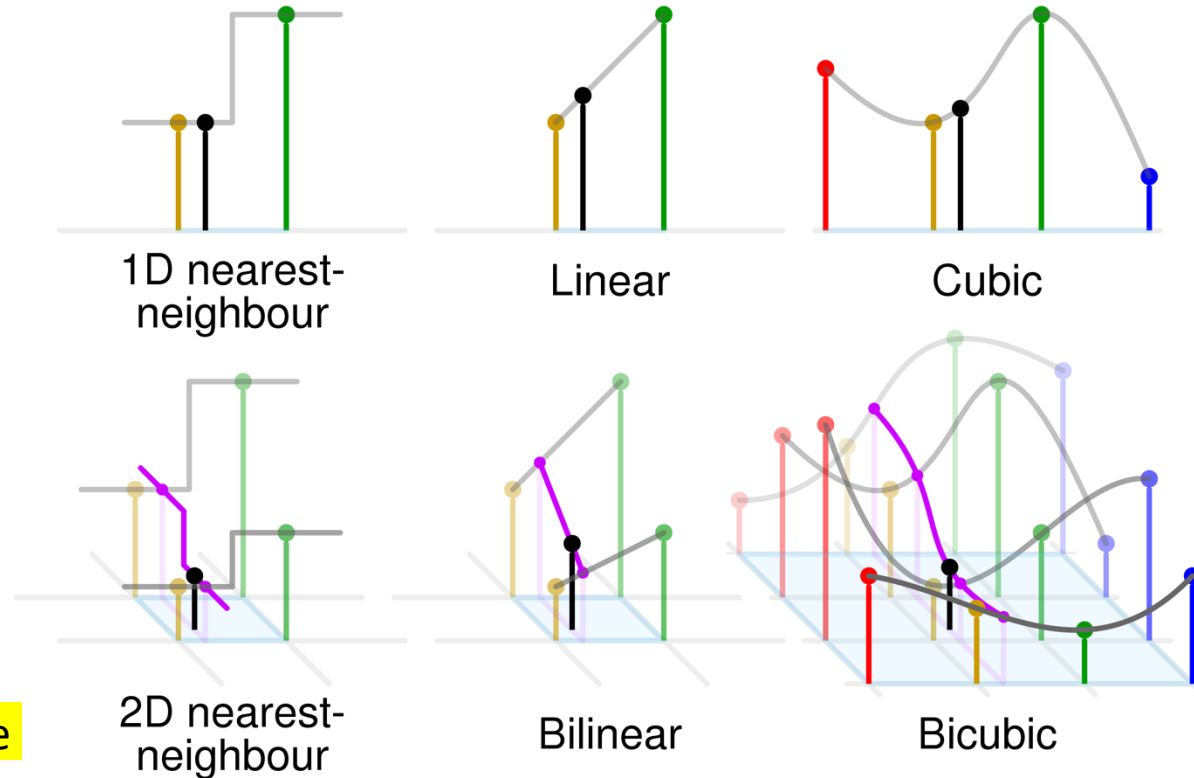
- **Slower Performance:**
 - More computationally expensive than simpler methods, making it slower on large images.
- **Blurring:**
 - Can introduce blurriness, especially when scaling down.
- **Halo Artifacts:**
 - Sometimes causes halo effects around edges in high-contrast areas.
- **Over-Smoothing:**
 - May smooth out fine details too much during upscaling, leading to a soft image.

Bicubic interpolation: Best use cases

- Moderate upscaling where image sharpness is not the highest priority but smoothness is.
- General-purpose resizing for photographs and images with a balance of speed and quality.

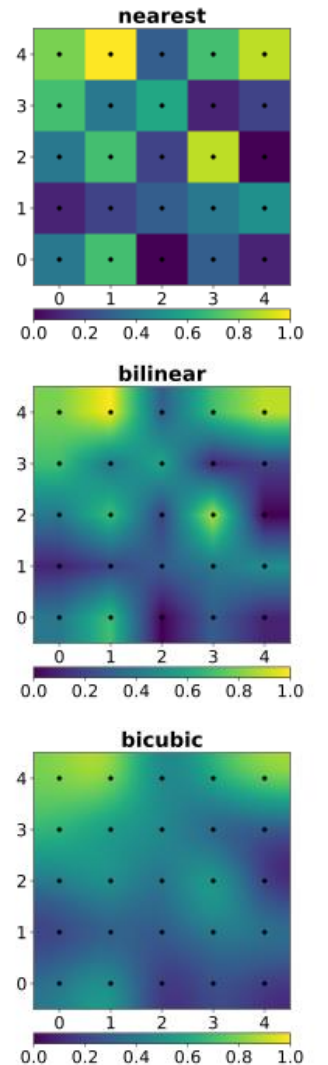
Summary

- Image interpolation methods
- Nearest neighbor interpolation
- Bilinear interpolation



Black dot denotes the sampled pixel value

(CMG Le. Wikipedia)





On image interpolation

<https://www.menti.com/bltyg9abucso>