# Visual Concepts and Definitions

**Introduction**

The human visual system (HVS) does not perceive light and intensity in a linear, objective way. These concepts explain the gap between physical light intensity and our subjective perception.

**Overview Understanding human vision is critical because it dictates how images are perceived and guides the choice of processing techniques.**

## 1. Digital Resolution Artifacts

**× False Contouring**

**Definition:** A visual artifact caused by **insufficient intensity resolution** (too few gray levels).

- **The Effect:** Smooth gradients turn into sharp, "stepped" bands (like a topographic map).
- **The Threshold:** To avoid this and blend colors naturally, the eye generally requires at least **24 distinct intensity levels**.


## 2. Human Visual Illusions

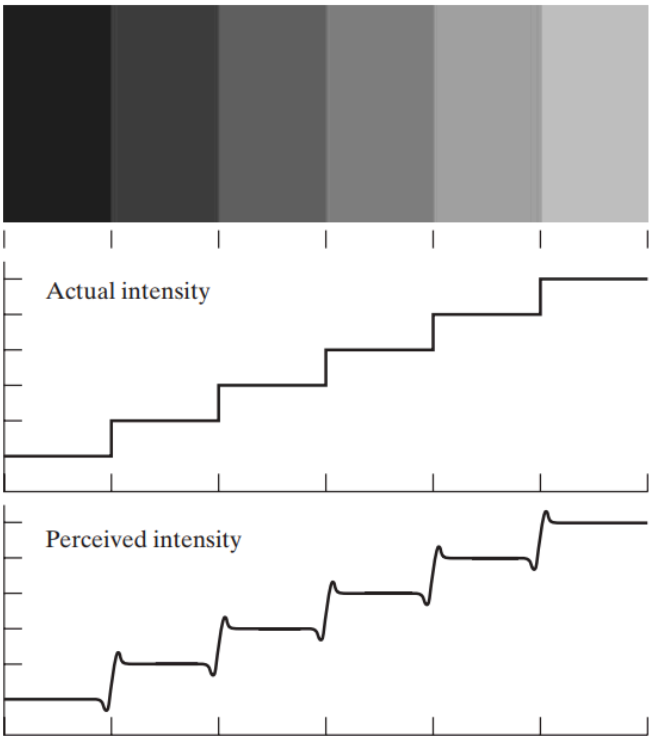
**Mach Bands**

**Biological Cause:** Lateral Inhibition (a form of natural edge enhancement).

**The Illusion:** Even when the intensity of a strip is **constant**, your brain creates false brighter and darker edges at the transition points to help you distinguish objects from the background.

a
b
c

**FIGURE 2.7**
Illustration of the Mach band effect. Perceived intensity is not a simple function of actual intensity.




## 3. Mathematical Sensitivity

### Brightness Discrimination (The Weber Ratio)

This measures how well the eye detects small changes in intensity ($I$) by finding the threshold $\Delta I_c$ detectable 50% of the time.

$$\text{Weber Ratio} = \frac{\Delta I_c}{I}$$

- **Small Ratio:** High sensitivity; a small change is easily discriminable.
- **Large Ratio:** Low sensitivity; a large change in intensity is required for the eye to notice.

## 4. System Adaptation

**Brightness Adaptation**

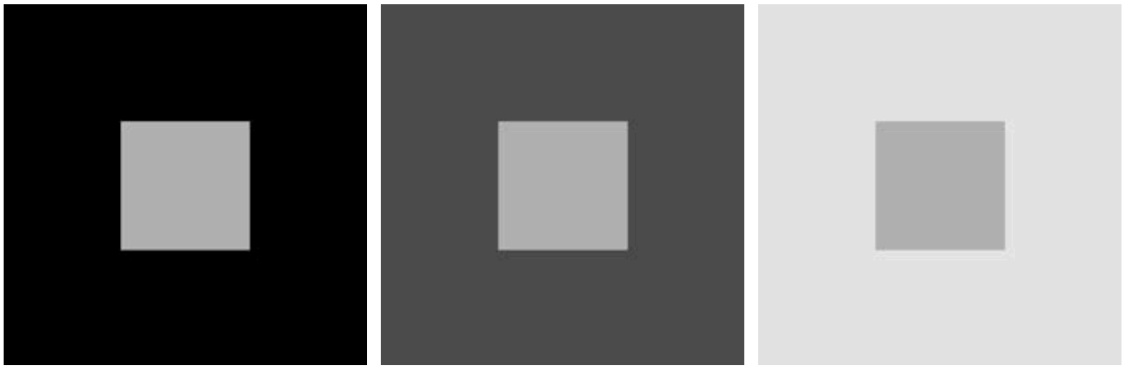
The HVS can perceive an enormous dynamic range ($10^{10}$), from the scotopic threshold to the glare limit.

**Key Limitation:** > The eye **cannot** operate over this entire range simultaneously. It adapts to a specific "average" brightness level, known as the **brightness adaptation level**. Think of it like a sliding window; we only see a small slice of the total range at any given moment.

## 5. Intensity variation

**Simultaneous Contrast A region's perceived brightness depends on the intensity of the surrounding background:**

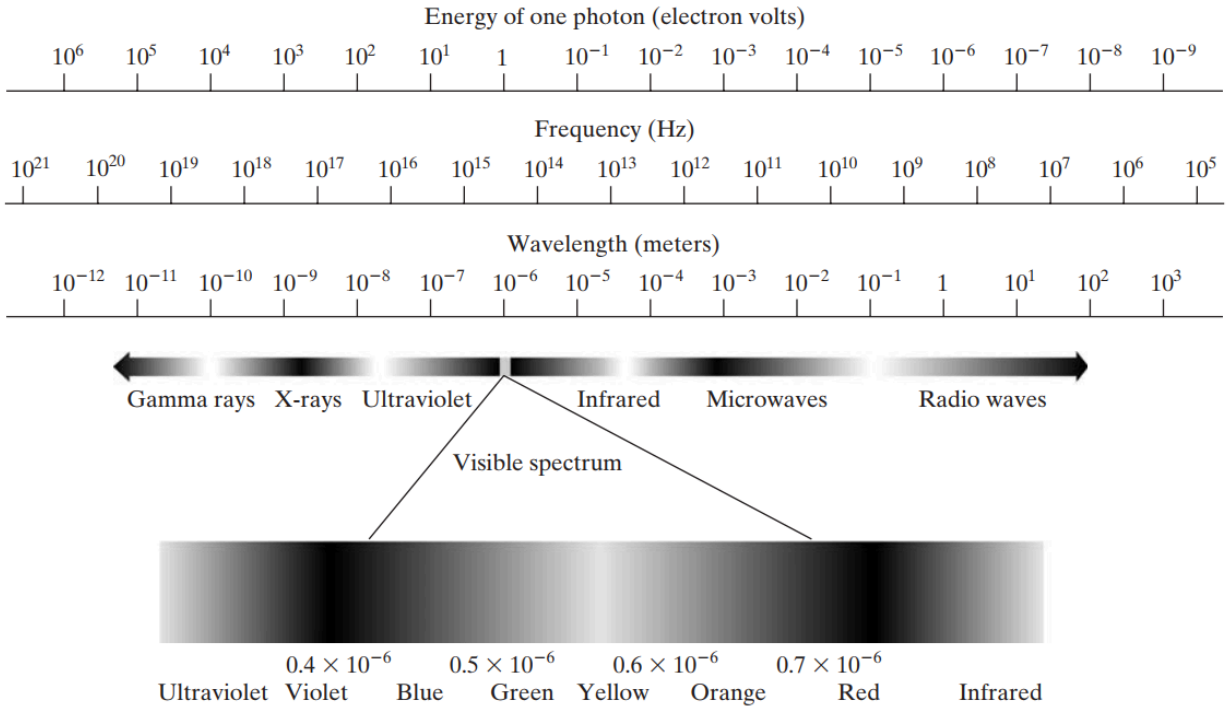
- Same gray square looks **darker** on a white background
- Same gray square looks **lighter** on a black background



a b c

**FIGURE 2.8** Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

## Light and Electromagnetic spectrum



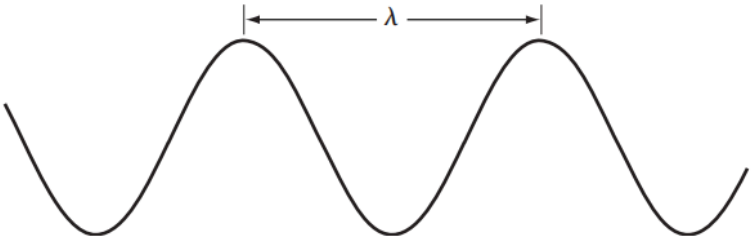

$$\lambda = \frac{c}{\nu}$$

$$E = h\nu$$

$c = 2.998 * 10^8$

Electromagnetic waves can be visualized as propagating sinusoidal waves with wavelength (Fig. 2.11), or they can be thought of as a stream of massless particles, each traveling in a wavelike pattern and moving at the speed of light. Each massless particle contains a certain amount (or bundle) of energy. Each bundle of energy is called a photon.

**FIGURE 2.11** Graphical representation of one wavelength.



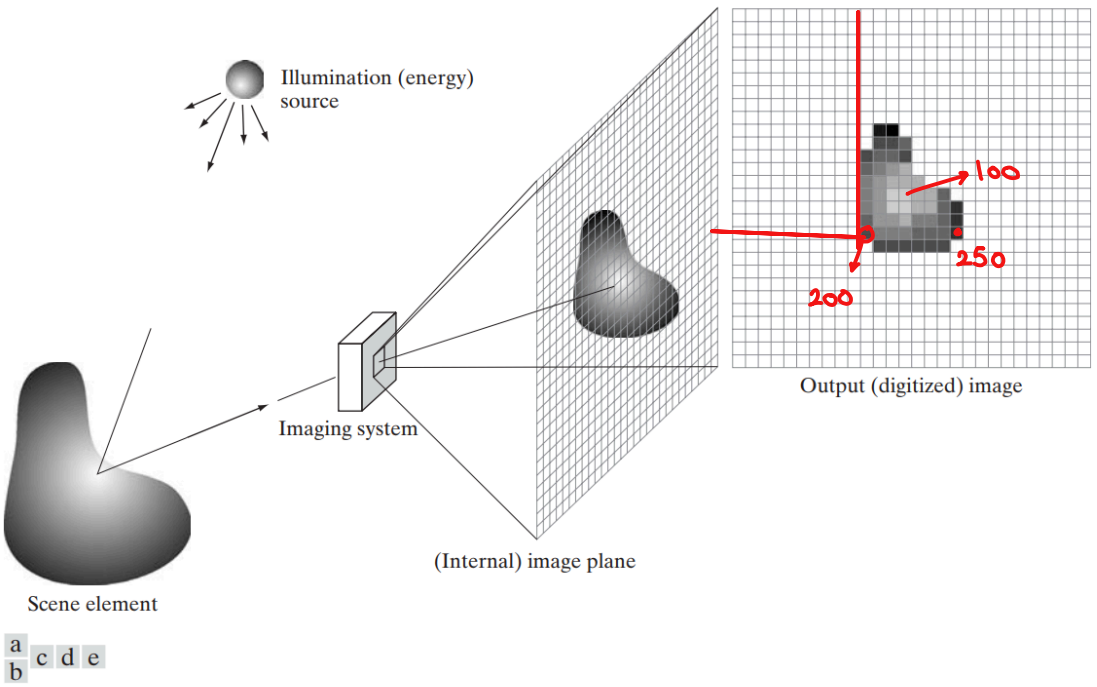

# Image Acquisition & Formation Model

**Concept Digital images are representations of continuous scenes generated by sensing energy.**

**representation** of two dimensional **image as** a finite set of picture elements called picture elements **pixels**

Pixel values represent **intensities** (gray levels)

Digitization means it is near to the original scene.




a b c d e

**FIGURE 2.15** An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

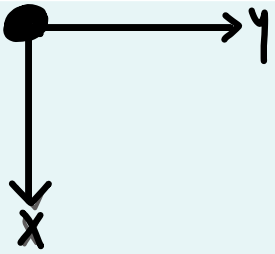
## The Mathematical Model

**Image Function An image is denoted by a 2-D function $f(x, y)$, where the value at spatial coordinates $(x, y)$ is a positive scalar intensity.**

### Fundamental Equation

$$f(x, y) = i(x, y) \cdot r(x, y)$$

↳ indepvariables




Where:

- **Illumination** ($i$): Amount of source illumination incident on the scene
  Range: $0 < i < \infty$ → theoritical
- **Reflectance** ($r$): Amount of illumination reflected by objects
  Range: $0 < r < 1$

*For transmission images (e.g., X-rays), this is transmissivity.*

---

## Intensity Limits and Gray Scale

- **Intensity Level** ($\ell$): The value of the monochrome image at any coordinates $(x_0, y_0)$ is denoted by $\ell = f(x_0, y_0)$. = $i(x_0, y_0)\, r(x_0, y_0)$
- **Practical Range**: $\ell$ lies in the interval $[L_{\min}, L_{\max}]$, where:

↳ classes ↳ (bits representing each pixel)

- $L_{\min} = i_{\min} r_{\min}$ (typically $\approx 10$ for indoor scenes).
- $L_{\max} = i_{\max} r_{\max}$ (typically $\approx 1000$ for indoor scenes).

- **Numerical Shifting**: In digital systems using gray scale, this interval is shifted to $[0, L-1]$.
  - $\ell = 0$: Represents **Black**.
  - $\ell = L-1$: Represents **White**.
  - All intermediate values represent varying shades of gray.
  - **k** : represents number of bits required to store those L-1 intensity levels


$2^k = L$ 4bits $L = 2^4 = 16$ level

**Important**

The only attribute of monochromatic light is its intensity or amount.
Because the intensity of monochromatic light is perceived to vary from black to grays and finally to white.
Terms **intensity** and **gray level** areinterchangeable

**Image Sensing Sensors (single, strip, or array) transform incoming energy (e.g., light, X-rays) into voltage waveforms proportional to the energy intensity.**

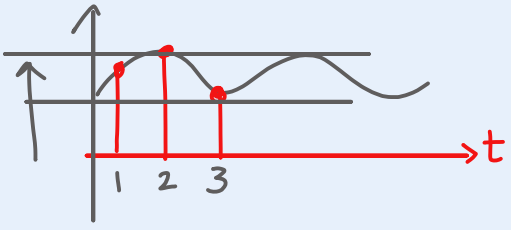
---

# Image Sampling and Quantization

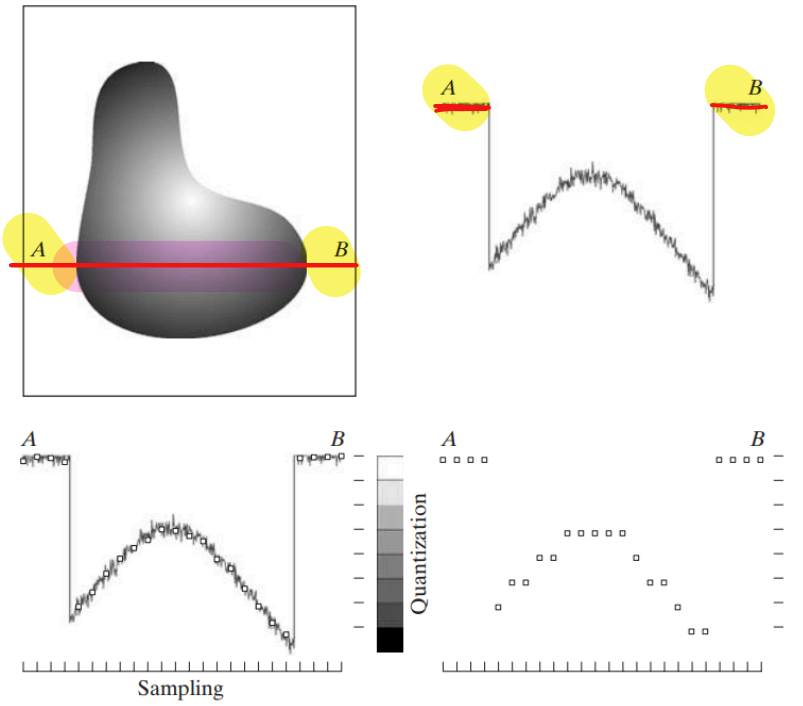
**Converting continuous data into a digital format requires two distinct processes.**

**The Two Processes**

1. **Sampling**: Digitizing the coordinate values (spatial location) Sampling in horizontal
   - Partitions the $xy$-plane into a grid
2. **Quantization**: Digitizing the amplitude (intensity) values → Sampling in vertical



- Converts continuous intensity levels into discrete quantities




a b
c d

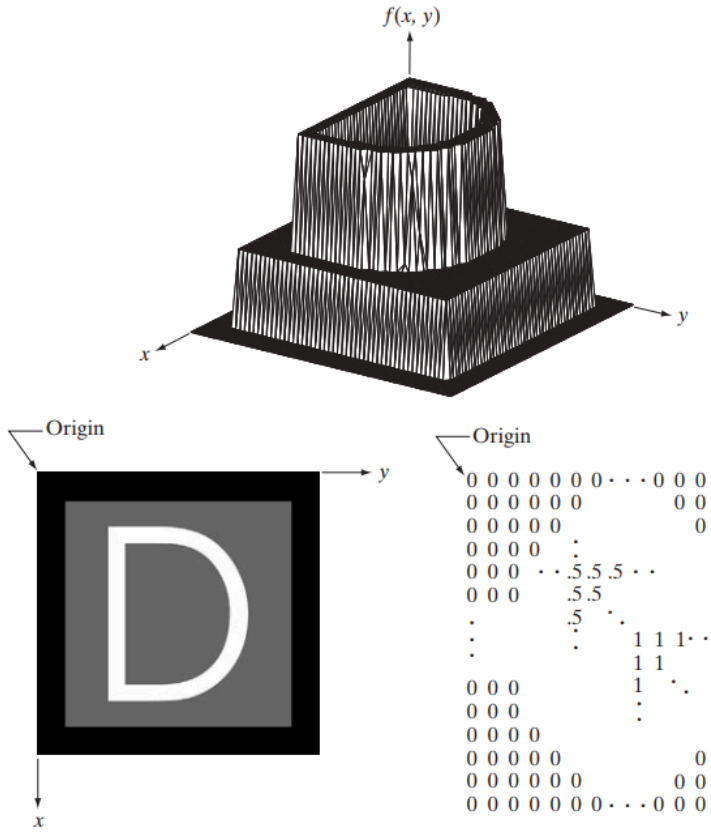
**FIGURE 2.16** Generating a digital image. (a) Continuous image. (b) A scan line from $A$ to $B$ in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

## Representing Digital Images

**Matrix Representation A digital image is represented as an $M \times N$ matrix (or array) of discrete elements called pixels (or pels).**

**Key Concepts**

- The section of the real plane spanned by the coordinates of an image is called the **spatial domain**, with x and y being referred to as **spatial variables** or **spatial coordinates**
- Discrete levels **L** are assumed to be equally spaced and integer power of 2 for quantizing hardware considerations.
-
    - There are no restrictions placed on M and N, other than they have to be **positive integers**.




a
b c

**FIGURE 2.18**
(a) Image plotted as a surface.
(b) Image displayed as a visual intensity array.
(c) Image shown as a 2-D numerical array (0, .5, and 1 represent black, gray, and white, respectively).

uint4 - 4bits
0 8 15
0 1 → 1bit
0 1 2 3 2bits

Number of bits required to store an image

$$b = M * N * K$$

rows x cols x no of bits to store every cell

## Matrix reading

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N-1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1, 0) & f(M-1, 1) & \cdots & f(M-1, N-1) \end{bmatrix} \quad (2.4\text{-}1)$$

M rows N cols

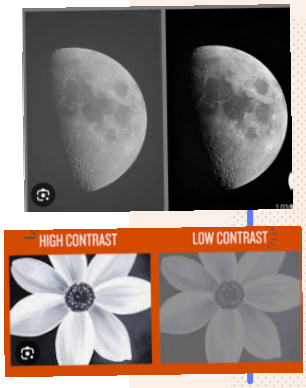
$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \quad (2.4\text{-}2)$$

## ⚠ Important concepts comparisons

- **Dynamic Range** : Ratio of maximum measurable intensity (saturation) to minimum detectable intensity (noise level).

  - **Upper Limit:** Determined by the saturation level (e.g., $L_{max}$ or white level).
  - **Lower Limit:** Determined by the noise floor or the minimum detectable intensity ($L_{min}$).
    **Typical Artifact** :Clipping (loss of detail in highlights/shadows).

- **Image contrast** : the **relative difference** in intensity between the highest and lowest gray levels *actually present* in a specific image.




  - An image with a high dynamic range capability can still be "low contrast" if all its pixels are bunched up in the middle of the gray scale.
  - **Low Contrast:** An image with a narrow range of gray levels (e.g., looking "washed out" or "grayish").
  - **High Contrast:** An image with a wide distribution of gray levels, ranging from deep blacks to bright whites.
    **Typical Artifact** : "Washed out" look or lack of detail between objects.

- **Saturation**: The highest value beyond which all intensity levels are clipped
- **Noise** masks the lowest detectable true intensity level.

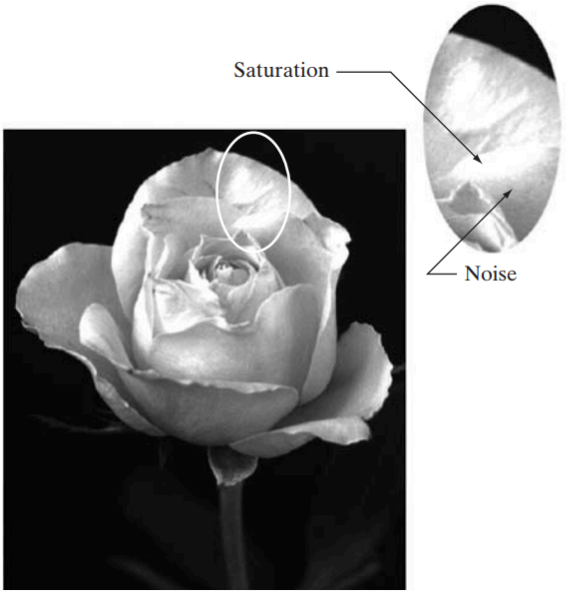



**FIGURE 2.19** An image exhibiting saturation and noise. Saturation is the highest value beyond which all intensity levels are clipped (note how the entire saturated area has a high, *constant* intensity level). Noise in this case appears as a grainy texture pattern. Noise, especially in the darker regions of an image (e.g., the stem of the rose) masks the lowest detectable true intensity level.

## Spatial and Intensity Resolution

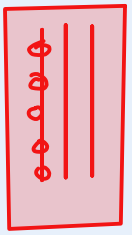
**Concept** The quality of a digital image depends on the number of samples and gray levels.

## Spatial Resolution

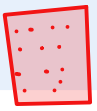
**Definition** A measure of the smallest discernible detail in an image.

**Common units:**

- Line pairs per unit distance
- Dots (pixels) per inch (dpi)



10 pixel / line
3 line/dis



20 pixel / dot
20 dot

## Intensity Resolution

**Gray Levels** The number of bits used to represent intensity levels.

- Typical systems use $2^k$ levels

Smooth transition
sharp

- Example: 8-bit images have **256 levels**

a b
c d

**FIGURE 2.20** Typical effects of reducing spatial resolution. Images shown at: (a) 1250 dpi, (b) 300 dpi, (c) 150 dpi, and (d) 72 dpi. The thin black borders were added for clarity. They are not part of the data.

### ⚠ *Remember*

False Contouring An artifact caused by **insufficient intensity resolution**, appearing as smooth, ridge-like boundaries in areas of gradual intensity change.

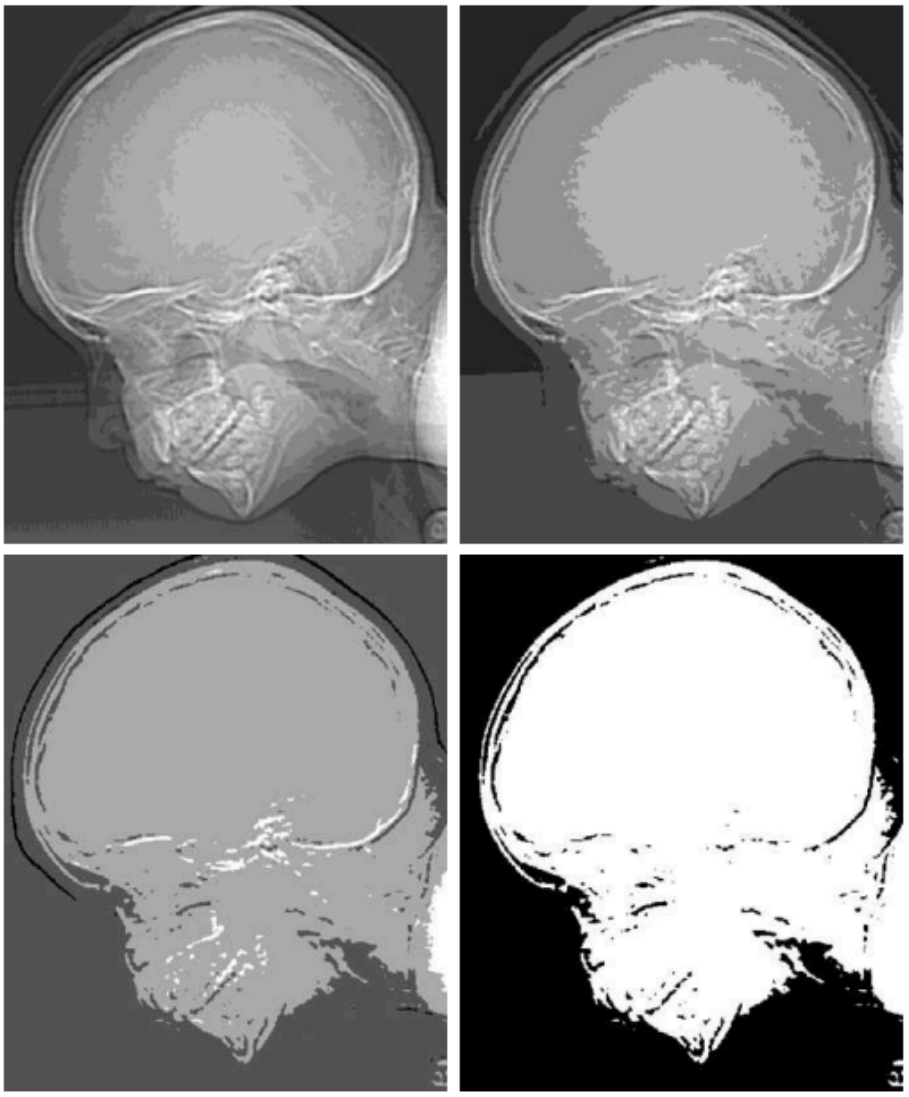
### Rule of Thumb

- As **spatial resolution decreases** → Image appears "blocky" (checkerboard effect)

- As **intensity resolution decreases** → False contouring increases

a b
c d

**FIGURE 2.21**
(a) 452 × 374, 256-level image. (b)–(d) Image displayed in 128, 64, and 32 intensity levels, while keeping the image size constant.



e f
g h

**FIGURE 2.21** (*Continued*) (e)–(h) Image displayed in 16, 8, 4, and 2 intensity levels. (Original courtesy of Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)

## Image Interpolation

**Definition**

The process of using known data to estimate values at unknown locations, required for zooming, shrinking, or rotating images.
We have empty spaces between known pixels, and we need to guess the intensity values for those spaces.

## Interpolation Methods

## Section Problems to solve

**2.2** When you enter a dark theater on a bright day, it takes an appreciable interval of time before you can see well enough to find an empty seat. Which of the visual processes explained in Section 2.1 is at play in this situation?

→ Brightness adaptation

**★2.3** Although it is not shown in Fig. 2.10, alternating current certainly is part of the electromagnetic spectrum. Commercial alternating current in the United States has a frequency of 60 Hz. What is the wavelength in kilometers of this component of the spectrum?

$\lambda = \frac{2.998 \times 10^8}{60} = 4997\,km$

**2.4** You are hired to design the front end of an imaging system for studying the boundary shapes of cells, bacteria, viruses, and protein. The front end consists, in this case, of the illumination source(s) and corresponding imaging camera(s). The diameters of circles required to enclose individual specimens in each of these categories are 50, 1, 0.1, and 0.01 μm, respectively.

**(a)** Can you solve the imaging aspects of this problem with a single sensor and camera? If your answer is yes, specify the illumination wavelength band and the type of camera needed. By "type," we mean the band of the electromagnetic spectrum to which the camera is most sensitive (e.g., infrared).

**(b)** If your answer in (a) is no, what type of illumination sources and corresponding imaging sensors would you recommend? Specify the light sources and cameras as requested in part (a). Use the *minimum* number of illumination sources and cameras needed to solve the problem.

Band $< 0.01\,\mu m$
Ultraviolet

**★2.9** A common measure of transmission for digital data is the *baud rate*, defined as the number of bits transmitted per second. Generally, transmission is accomplished in packets consisting of a start bit, a byte (8 bits) of information, and a stop bit. Using these facts, answer the following:

**(a)** How many minutes would it take to transmit a 1024 × 1024 image with 256 intensity levels using a 56K baud modem?

**(b)** What would the time be at 3000K baud, a representative medium speed of a phone DSL (Digital Subscriber Line) connection?

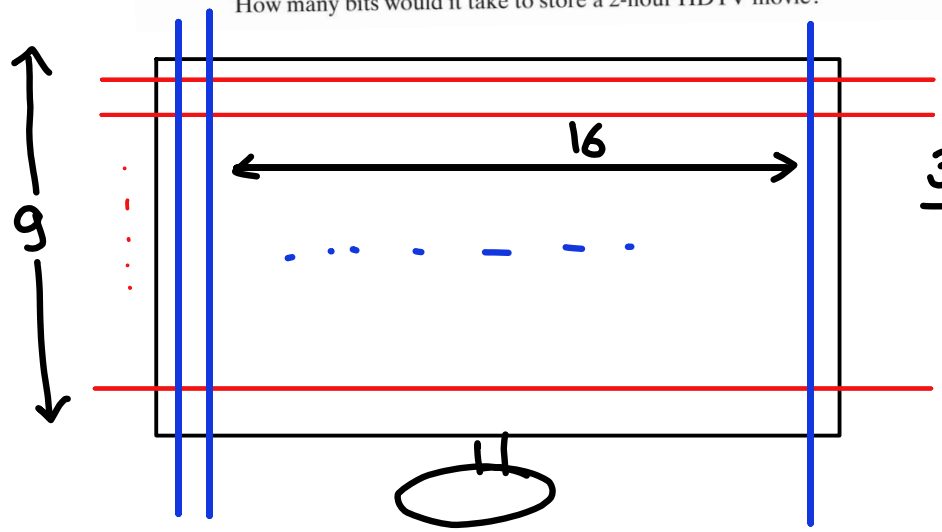
baudrate: bits/sec

Packet = 10 bits

a) $t_{min} = 1024 \times 1024 \times 8_{bits} \times \frac{1\ sec}{56\ kbit} = \frac{187.25\ sec}{60} = 3.1\ min$

[ □ ]

**2.10** High-definition television (HDTV) generates images with 1125 horizontal TV lines interlaced (where every other line is painted on the tube face in each of two fields, each field being 1/60th of a second in duration). The width-to-height aspect ratio of the images is 16:9. The fact that the number of horizontal lines is fixed determines the vertical resolution of the images. A company has designed an image capture system that generates digital images from HDTV images. The resolution of each TV (horizontal) line in their system is in proportion to vertical resolution, with the proportion being the width-to-height ratio of the images. Each pixel in the color image has 24 bits of intensity resolution, 8 bits each for a red, a green, and a blue image. These three "primary" images form a color image. How many bits would it take to store a 2-hour HDTV movie?




1125

$\frac{30 \text{ frame(image)}}{\text{Sec}}$

vertical lines are across width

Horizontal lines height

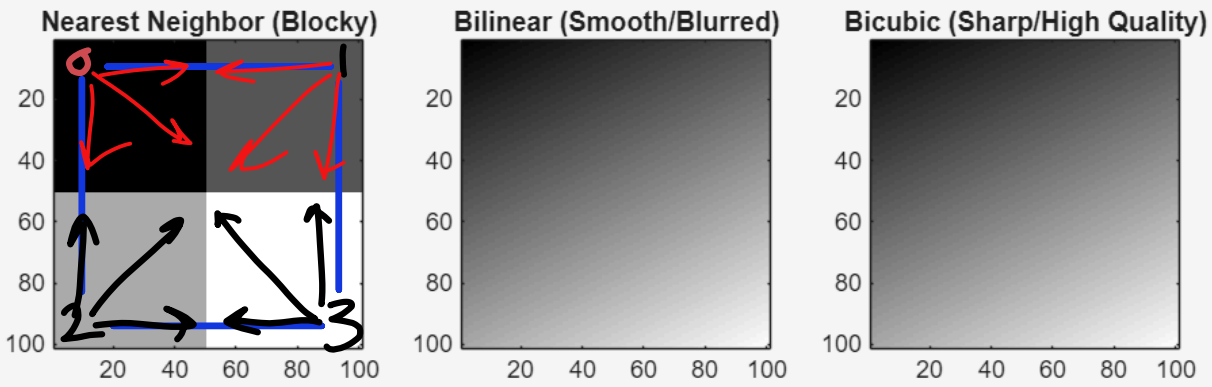
pixel per vertical line = $\frac{x}{1125} * \frac{16}{9} = 2000$

one Paint Size = $1125 \times 2000 \times \underbrace{8 \times 3}$

Paint/sec = Paint Size * framerate $= (1125)(2000)(24)(30)$

Paint for 2hrs = Paint/sec $\times 60 \times 60 \times 2 = 1.166 \times 10^{13}$ bits

# Interpolation part in section




# Interpolation Methods in Image Processing

### The Scenario

Imagine we have a **2x2** source image and we want to upscale it to **3x3**.

**Source Matrix ($A$):**

$$A = \begin{bmatrix} 10 & a & 20 \\ b & c & d \\ 30 & e & 40 \end{bmatrix}$$

We want to find the value of a new pixel at a fractional coordinate, for example, $(0.5, 0.5)$. which is (2,1) in the 3x3 coordinates

$a = 10 + \frac{1}{8} \times (20-10) = 10.625$

$c = 10 + \frac{3}{8} \times (20-10) =$

## 1. Nearest Neighbor Interpolation

### **Nearest** Neighbor Interpolation Method: Assigns the intensity of the nearest neighbor in the original grid to the new location.

This is the simplest and fastest method. It simply assigns the value of the **closest** pixel to the new coordinate.

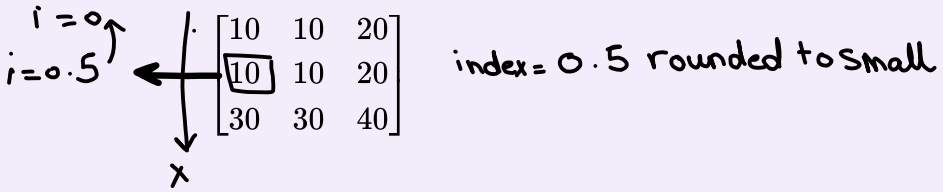
**Pros:** Fast

**Cons:** Produces "blocky" artifacts (jaggies)

× **The "Blocky" Problem Because Nearest Neighbor just "grabs" the closest existing value without looking at others, you get large blocks of the same color. This is why some types of zoomed-in images look "pixelated."**

### Calculation


**Resulting 3x3 Matrix:**




**Target indices mapping** : map the target indices $(0, 1, 2)$ back to the source coordinates using the step size of 0.5 ⟶
Calculated using this formula:

$$\text{Stretched Source Coordinate} = \text{Target Index} \times \frac{(\text{Source Size} - 1)}{(\text{Target Size} - 1)} = 1 \times \frac{1}{2}$$

**Square stretching coordinates step size is the same for x and y. However, if stretched unequally the step size is different**

**Mapping logic**

- **Target Index 0** maps to **Source 0.0**.
- **Target Index 1** maps to **Source 0.5**.
- **Target Index 2** maps to **Source 1.0**.

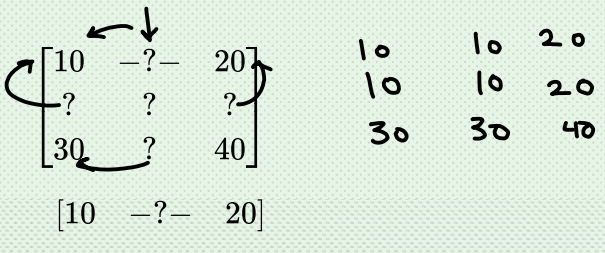
**Nearest Neighbor Rule:** Round the source coordinate to the nearest integer index ($0$ or $1$).
(in this example, $0.5$ is mapped to index $0$).

## Nearest Neighbour solution

✓ **Matrix Solution: Nearest Neighbor**

**Source Matrix ($A$):**

$$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$



**Target Row 0 (Source $y = 0.0$):**

- $x = 0.0 \rightarrow$ Round to $0$: Value = $A[0, 0] = \mathbf{10}$.
- $x = 0.5 \rightarrow$ Round to $0$: Value = $A[0, 0] = \mathbf{10}$.
- $x = 1.0 \rightarrow$ Round to $1$: Value = $A[0, 1] = \mathbf{20}$.
- **Result:** $[10, 10, 20]$.

$$[?\quad ?\quad ?]$$

**Target Row 1 (Source $y = 0.5$):**

- Since $y = 0.5$ rounds to **0**, this entire row replicates the logic of **Row 0**.
- $x = 0.0 \rightarrow$ Value = $A[0, 0] = \mathbf{10}$.
- $x = 0.5 \rightarrow$ Value = $A[0, 0] = \mathbf{10}$.
- $x = 1.0 \rightarrow$ Value = $A[0, 1] = \mathbf{20}$.
- **Result:** $[10, 10, 20]$.

$$[30\quad ?\quad 40]$$

**Target Row 2 (Source $y = 1.0$):**

- $y = 1.0$ maps to the second row of the source ($30$ and $40$).
- $x = 0.0 \rightarrow$ Round to $0$: Value = $A[1, 0] = \mathbf{30}$.
- $x = 0.5 \rightarrow$ Round to $0$: Value = $A[1, 0] = \mathbf{30}$.
- $x = 1.0 \rightarrow$ Round to $1$: Value = $A[1, 1] = \mathbf{40}$.
- **Result:** $[30, 30, 40]$.

Combining these rows gives your target $3 \times 3$ matrix:

$$\begin{bmatrix} 10 & 10 & 20 \\ 10 & 10 & 20 \\ 30 & 30 & 40 \end{bmatrix}$$

**Tip**

Notice the "blocky" nature of the result. Entire columns and rows are duplicated because the $0.5$ coordinate is forced to pick one side or the other rather than blending them.

# Bilinear Interpolation

2. **Bilinear Interpolation Method: Uses the four nearest neighbors to estimate intensity via linear equations in the $x$ and $y$ directions.**

**Result:** Produces smoother results than nearest neighbor

**The Equation:**

The value $v(x, y)$ at a location $(x, y)$ is approximated by the equation:

we need to find the 4 coefficients $(a, b, c, d)$. Since we have 4 known neighbors, we can set up 4 equations to solve for these 4 unknowns.

$$v(x, y) = ax + by + cxy + d$$

✓ **Example 1 square to square matrix**

$$A = \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

The step size is **0.5**.

Mapping :

- Target Index **0** → Source **0.0**
- Target Index **1** → Source **0.5**
- Target Index **2** → Source **1.0**

$$A' = \begin{bmatrix} 10 & 15 & 20 \\ 20 & & 30 \\ 30 & 35 & 40 \end{bmatrix}$$

When the coordinate is exactly $0.5$, the value is simply the average: $\frac{P_1+P_2}{2}$.

Target Row 0 (Top Edge: $y = 0.0$)

We interpolate horizontally between $10$ and $20$:

1 step = $\frac{1}{2} \times (20 - 10) = 5$ (row)

**Col 0 ($x = 0.0$):** $\rightarrow \mathbf{10}$

**Col 1 ($x = 0.5$):** Average of $10$ and $20 \rightarrow (10 + 20)/2 = \mathbf{15}$

**Col 2 ($x = 1.0$):** Exact match $\rightarrow \mathbf{20}$

**Row 0 Result:** $[10, 15, 20]$

same for other rows result is

$$\begin{bmatrix} 10 & 15 & 20 \\ 20 & 25 & 30 \\ 30 & 35 & 40 \end{bmatrix}$$

# Precautions for Bilinear Interpolation in MATLAB

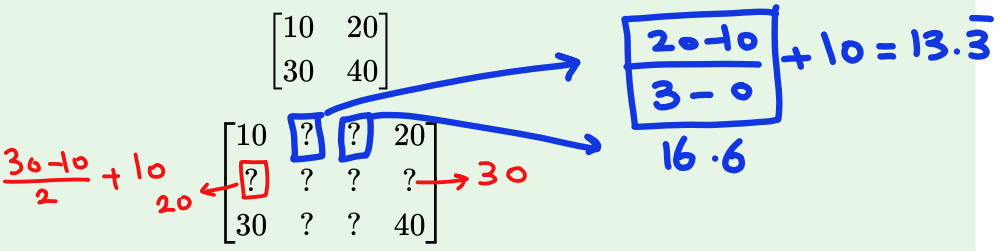
⚠ **Important Precautions**

1. **Blurring Effect:** Bilinear interpolation acts as a **Low-Pass Filter**. While it removes the "blockiness" of Nearest Neighbor, it softens sharp edges and can make an image look slightly out of focus.
2. **Boundary Conditions:** When the target coordinate falls exactly on $1.0$, make sure your code doesn't try to look for a pixel at index $2$ (which doesn't exist). Always use

`min(index, size-1)` or `clamp`.

3. **Precision Loss:** The division by 2 or calculation of $0.5$ can lead to rounding errors. Always check if your intermediate values need to be floating-point for accuracy.
4. **Aspect Ratio:** If you use a single step size for an image that isn't square (like $2 \times 2 \to 3 \times 4$), your results will be skewed. Always calculate $Step_x$ and $Step_y$ independently.

✓ **Example 2 square to skewed matrix**




**Vertical Step ($y$):** From 2 rows to 3 rows.

$$Step_y = \frac{\text{Source Rows} - 1}{\text{Target Rows} - 1} = \frac{2-1}{3-1} = \mathbf{0.5}$$

- *Target Y-coordinates:* $0.0, 0.5, 1.0$
- **Horizontal Step ($x$):** From 2 columns to 4 columns.

$$Step_x = \frac{\text{Source Columns} - 1}{\text{Target Columns} - 1} = \frac{2-1}{4-1} = \frac{1}{3} \approx \mathbf{0.333}$$

*Target X-coordinates:* $0.0, 0.333, 0.666, 1.0$

- **Average values**

$$\begin{bmatrix} 10 & ? & ? & 20 \\ 20 & ? & ? & 30 \\ 30 & ? & ? & 40 \end{bmatrix}$$

- **Delta Steps**

نسبة الـ Source

$$f(x, y_{fixed}) = f(x_1) + \text{fractional\_distance} \cdot (f(x_2) - f(x_1))$$

$$v(x,0) = \underbrace{10}_{Q_{11}} + \underbrace{x}_{d/3}\underbrace{(20-10)}_{\Delta}$$

$$\begin{bmatrix} 10 & 13.33 & 16.66 & 20 \\ 20 & 23.33 & 26.66 & 30 \\ 30 & 33.33 & 36.66 & 40 \end{bmatrix}$$

*full equation*

When we include the $y$ dimension and the substitution $y = 0$, the full Bilinear rule is:

$$v(x, y) = (Q_{21} - Q_{11})x + (Q_{12} - Q_{11})y + (Q_{11} + Q_{22} - Q_{21} - Q_{12})xy + Q_{11}$$

- $d$ = $Q_{11}$ (the base value)
- $a$ = $Q_{21} - Q_{11}$ (the x slope (vertical))
- $b$ = $Q_{12} - Q_{11}$ (the y slope (horizontal))
- $c$ = $Q_{11} + Q_{22} - (Q_{21} + Q_{12})$ (the "twist" or curvature)

بعوض بـ ل Virtual Coordinates

$$\begin{bmatrix} Q11 & Q12 \\ Q21 & Q22 \end{bmatrix}$$

$$V(0.333, 0.5) = 10 + 10(0.333) + 20(0.5) + 0(0.333)(0.5)$$

$$V = 10 + 3.33 + 10 + 0 = \mathbf{23.33}$$

---

# Padding

> **Original Matrix $A$:**
>
> $$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

## 1. Replication Padding (Standard)

We repeat the boundary pixels to fill the $4 \times 4$ space. This assumes the scene continues at the same intensity.

$$A_{rep} = \begin{bmatrix} 10 & 10 & 20 & 20 \\ 10 & 10 & 20 & 20 \\ 30 & 30 & 40 & 40 \\ 30 & 30 & 40 & 40 \end{bmatrix}$$

## 2. Zero Padding (Black Borders)

We surround the $2 \times 2$ with zeros.

$$A_{zero} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 10 & 20 & 0 \\ 0 & 30 & 40 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- The Bicubic algorithm sees a sharp drop to $0$ at the edges. Because cubic curves "overshoot," the values near the edges of your $10, 20, 30, 40$ matrix will actually **dip lower** before rising. This creates a dark "halo" effect around your image.

**Flatness:** The sum of all weights in any interpolation must always equal **1.0** to ensure the brightness of the image doesn't change.
**Continuity:** The pieces of the function must meet smoothly at $x = 1$ (the derivative must be continuous).

$$\text{Pixel Value} = \sum_{i=-1}^{2} \sum_{j=-1}^{2} A(i,j) \cdot W_y(\text{dist}_y) \cdot W_x(\text{dist}_x)$$

## 3. Bicubic Interpolation **Extra Reading might be used in codes**

✓ 3. **Bicubic Interpolation Method: Uses the sixteen nearest neighbors.**

**Advantages:**

- Preserves fine detail better than bilinear interpolation
- Standard for commercial image editing software
- **Why 16?** 2 for each direction the x and the y = $2^2 * 2^2 = 16$ 2 blocks moved in each direction from the pixel chosen

### The Cubic Weighting Function $W(t)$ (Catmull-Rom)

The **Catmull-Rom** spline ($\alpha = -0.5$) is a popular choice because it is interpolatory (it passes exactly through control points).
For $\alpha = -0.5$, the weight $W$ based on distance $t$ is:

$$W(t) = \begin{cases} 1.5|t|^3 - 2.5|t|^2 + 1 & 0 \le |t| < 1 \\ -0.5|t|^3 + 2.5|t|^2 - 4|t| + 2 & 1 \le |t| < 2 \\ 0 & |t| \ge 2 \end{cases}$$

This part of the curve handles the two pixels closest to the "virtual" point. We apply these four conditions:
$W(0) = 1$: If the distance is zero, we must get 100% of that pixel.
$W(1) = 0$: The influence must drop to zero when we reach the next pixel.
$W'(0) = 0$: The slope must be flat at the center for a smooth peak.
$W'(1) = \alpha$: The slope at the edge is controlled by a "sharpening" parameter.

## example on bicubic

**Source Matrix ($A$):**
Plaintext

```
[10, 20]
[30, 40]
```

To perform Bicubic interpolation, we need a 4x4 neighborhood. We apply **Replication Padding**:

> **Padded Grid ($4 \times 4$):**
>
> $$P = \begin{bmatrix} 10 & 10 & 20 & 20 \\ 10 & 10 & 20 & 20 \\ 30 & 30 & 40 & 40 \\ 30 & 30 & 40 & 40 \end{bmatrix} \begin{matrix} \leftarrow \text{Row -1 (Replicated)} \\ \leftarrow \text{Row 0 (Original)} \\ \leftarrow \text{Row 1 (Original)} \\ \leftarrow \text{Row 2 (Replicated)} \end{matrix}$$

## Phase B: Horizontal Pass (Weighting $x$)

Mapping 4 target columns to source coordinates: $x \in \{0, 0.33, 0.66, 1\}$.

### 1. Horizontal Weight Calculation ($W_x$)

| Target x | p−1 (dist) | p0 (dist) | p1 (dist) | p2 (dist) | Weights [w−1,w0,w1,w2] |
|---|---|---|---|---|---|
| **0.00** | 1.0 | 0.0 | 1.0 | 2.0 | $[0, 1, 0, 0]$ |
| **0.33** | 1.33 | 0.33 | 0.67 | 1.67 | $[-0.071, 0.781, 0.331, -0.041]$ |
| **0.66** | 1.66 | 0.66 | 0.34 | 1.34 | $[-0.041, 0.331, 0.781, -0.071]$ |
| **1.00** | 2.0 | 1.0 | 0.0 | 1.0 | $[0, 0, 1, 0]$ |

We plug each distance ($|t|$) into the Catmull-Rom kernel to get the desired values.

### 2. Intermediate Matrix Calculation

We apply these weights to every row of our padded grid. Since Rows -1 and 0 are identical, and Rows 1 and 2 are identical, we only calculate two sets:

- **Intermediate Rows -1 & 0:**
  - Col 0: $10$
  - Col 1: $10(-0.071) + 10(0.781) + 20(0.331) + 20(-0.041) = \mathbf{12.9}$
  - Col 2: $10(-0.041) + 10(0.331) + 20(0.781) + 20(-0.071) = \mathbf{17.1}$
  - Col 3: $20$
- **Intermediate Rows 1 & 2:**

- Col 0: $30$
- Col 1: $30(-0.071) + 30(0.781) + 40(0.331) + 40(-0.041) = \mathbf{32.9}$
- Col 2: $30(-0.041) + 30(0.331) + 40(0.781) + 40(-0.071) = \mathbf{37.1}$
- Col 3: $40$

**Intermediate Matrix ($I$):**

```
[10.0, 12.9, 17.1, 20.0]  (Row -1)
[10.0, 12.9, 17.1, 20.0]  (Row 0)
[30.0, 32.9, 37.1, 40.0]  (Row 1)
[30.0, 32.9, 37.1, 40.0]  (Row 2)
```

## Phase C: Vertical Pass (Weighting $y$)

Mapping 3 target rows to source coordinates: $y \in \{0, 0.5, 1\}$.

### 1. Vertical Weight Calculation ($W_y$)

| Target y | w−1 (dist 1.5) | w0 (dist 0.5) | w1 (dist 0.5) | w2 (dist 1.5) |
|---|---|---|---|---|
| **0.0** | 0 | 1 | 0 | 0 |
| **0.5** | -0.0625 | 0.5625 | 0.5625 | -0.0625 |
| **1.0** | 0 | 0 | 1 | 0 |

### 2. Final Output Construction

- **Target Row 0 ($y = 0$):** Copy of Intermediate Row 0.
- **Target Row 2 ($y = 1$):** Copy of Intermediate Row 1.
- **Target Row 1 ($y = 0.5$):** Interpolate using vertical weights:
  - $Val = (-0.0625 \times I_{-1}) + (0.5625 \times I_0) + (0.5625 \times I_1) + (-0.0625 \times I_2)$
  - Col 0: $10(-0.0625) + 10(0.5625) + 30(0.5625) + 30(-0.0625) = \mathbf{20.0}$
  - Col 1: $12.9(-0.0625) + 12.9(0.5625) + 32.9(0.5625) + 32.9(-0.0625) = \mathbf{22.9}$
  - Col 2: $17.1(-0.0625) + 17.1(0.5625) + 37.1(0.5625) + 37.1(-0.0625) = \mathbf{27.1}$
  - Col 3: $20.0(-0.0625) + 20.0(0.5625) + 40.0(0.5625) + 40.0(-0.0625) = \mathbf{30.0}$

---

## 3. Final $3 \times 4$ Result

```
[[10.0, 12.9, 17.1, 20.0],
 [20.0, 22.9, 27.1, 30.0],
 [30.0, 32.9, 37.1, 40.0]]
```

⚠ **General Implementation Rules**

1. **Separability:** Bicubic interpolation is separable. You can compute all horizontal interpolations first, then use those results to compute vertical interpolations.
2. **Neighborhood:** Always identify the 16 surrounding pixels. If the point is at $(1.2, 3.4)$, the $x$-indices are $\{0, 1, 2, 3\}$ and $y$-indices are $\{2, 3, 4, 5\}$.
3. **Boundary Handling:** Use **Replication** (clamping) or **Reflection** to fill the 4x4 grid when near edges.
4. **Normalization:** The sum of weights in a Catmull-Rom kernel always equals 1.0. If your calculated weights don't sum to 1, there is a calculation error.
5. **Preservation:** If the target coordinate is an integer (e.g., $x = 1.0$), the weights must become $[0, 0, 1, 0]$, effectively selecting the original pixel value.

## Matrix calculation for bicubic

we calculate every element in a row multiplication

$$I(x,y) = \begin{bmatrix} w_{y,-1} & w_{y,0} & w_{y,1} & w_{y,2} \end{bmatrix} \begin{bmatrix} P_{-1,-1} & P_{-1,0} & P_{-1,1} & P_{-1,2} \\ P_{0,-1} & P_{0,0} & P_{0,1} & P_{0,2} \\ P_{1,-1} & P_{1,0} & P_{1,1} & P_{1,2} \\ P_{2,-1} & P_{2,0} & P_{2,1} & P_{2,2} \end{bmatrix} \begin{bmatrix} w_{x,-1} \\ w_{x,0} \\ w_{x,1} \\ w_{x,2} \end{bmatrix}$$

calculate the pixel at **Target Row 1, Col 1** ($x = 0.33, y = 0.5$):

**The Weights:**

- $\mathbf{W_y} = [-0.0625 \quad 0.5625 \quad 0.5625 \quad -0.0625]$
- $\mathbf{W_x}^T = [-0.071 \quad 0.781 \quad 0.331 \quad -0.041]^T$

**Inner Product ($\mathbf{P} \cdot \mathbf{W_x}^T$):** This performs the horizontal interpolation for all 4 rows simultaneously, resulting in a $4 \times 1$ column of intermediate values.

**Outer Product ($\mathbf{W_y} \cdot \text{Intermediate}$):** This collapses those 4 intermediate values into the final single scalar value for that pixel.

## Horizontal Weights ($W_x$) for $x = 0.33$

Distances ($|t|$) to columns $\{-1, 0, 1, 2\}$ are $\{1.33, 0.33, 0.67, 1.67\}$.

- $w_{x,-1}$ ($|t| = 1.33$): $-0.5(1.33)^3 + 2.5(1.33)^2 - 4(1.33) + 2 \approx \mathbf{-0.074}$
- $w_{x,0}$ ($|t| = 0.33$): $1.5(0.33)^3 - 2.5(0.33)^2 + 1 \approx \mathbf{0.782}$
- $w_{x,1}$ ($|t| = 0.67$): $1.5(0.67)^3 - 2.5(0.67)^2 + 1 \approx \mathbf{0.330}$
- $w_{x,2}$ ($|t| = 1.67$): $-0.5(1.67)^3 + 2.5(1.67)^2 - 4(1.67) + 2 \approx \mathbf{-0.038}$

**Vector** $W_x = [-0.074 \quad 0.782 \quad 0.330 \quad -0.038]$

## Vertical Weights ($W_y$) for $y = 0.5$

Distances ($|t|$) to rows $\{-1, 0, 1, 2\}$ are $\{1.5, 0.5, 0.5, 1.5\}$.

- $w_{y,-1}$ **and** $w_{y,2}$ ($|t| = 1.5$): $-0.5(1.5)^3 + 2.5(1.5)^2 - 4(1.5) + 2 = \mathbf{-0.0625}$
- $w_{y,0}$ **and** $w_{y,1}$ ($|t| = 0.5$): $1.5(0.5)^3 - 2.5(0.5)^2 + 1 = \mathbf{0.5625}$

**Vector** $W_y = [-0.0625 \quad 0.5625 \quad 0.5625 \quad -0.0625]$

$$I(x, y) = W_y \cdot P \cdot W_x^T$$

## Step 1: $P \cdot W_x^T$ (Horizontal Interpolation)

This multiplies our $4 \times 4$ padded pixel matrix by the horizontal weight column. It gives the intermediate column we calculated before.

$$\begin{bmatrix} 10 & 10 & 20 & 20 \\ 10 & 10 & 20 & 20 \\ 30 & 30 & 40 & 40 \\ 30 & 30 & 40 & 40 \end{bmatrix} \cdot \begin{bmatrix} -0.074 \\ 0.782 \\ 0.330 \\ -0.038 \end{bmatrix} = \begin{bmatrix} 12.92 \\ 12.92 \\ 32.92 \\ 32.92 \end{bmatrix}$$

*Calculation for row 1:* $(10 \times -0.074) + (10 \times 0.782) + (20 \times 0.330) + (20 \times -0.038) = 12.92$.

## Step 2: $W_y \cdot (\text{Result of Step 1})$ (Vertical Interpolation)

Now we apply the vertical weights to these 4 intermediate row values.

$$[-0.0625 \quad 0.5625 \quad 0.5625 \quad -0.0625] \cdot \begin{bmatrix} 12.92 \\ 12.92 \\ 32.92 \\ 32.92 \end{bmatrix} = \text{Final Value}$$

**Final Result:**

$$(-0.0625 \times 12.92) + (0.5625 \times 12.92) + (0.5625 \times 32.92) + (-0.0625 \times 32.92)$$
$$= -0.8075 + 7.2675 + 18.5175 - 2.0575 = \mathbf{22.92}$$