ECE359 Digital Image Processing

Section 2 — Pixel Relationships, Operations & Spatial Transforms

TA: Adnan Zahran

Ain Shams University

23 February 2026

Session Outline

Pixel Neighbors

Core Concept

Image processing often relies on defining which pixels are **connected** to one another.

For a pixel p at coordinates (x, y), three types of neighbors are defined:

4-Neighbors $N_4(p)$

Horizontal & vertical only:

$$(x \pm 1,\ y), \quad (x,\ y \pm 1)$$

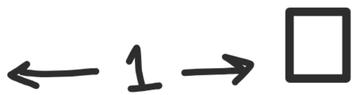
Diagonal $N_D(p)$

The four diagonal neighbors of p.

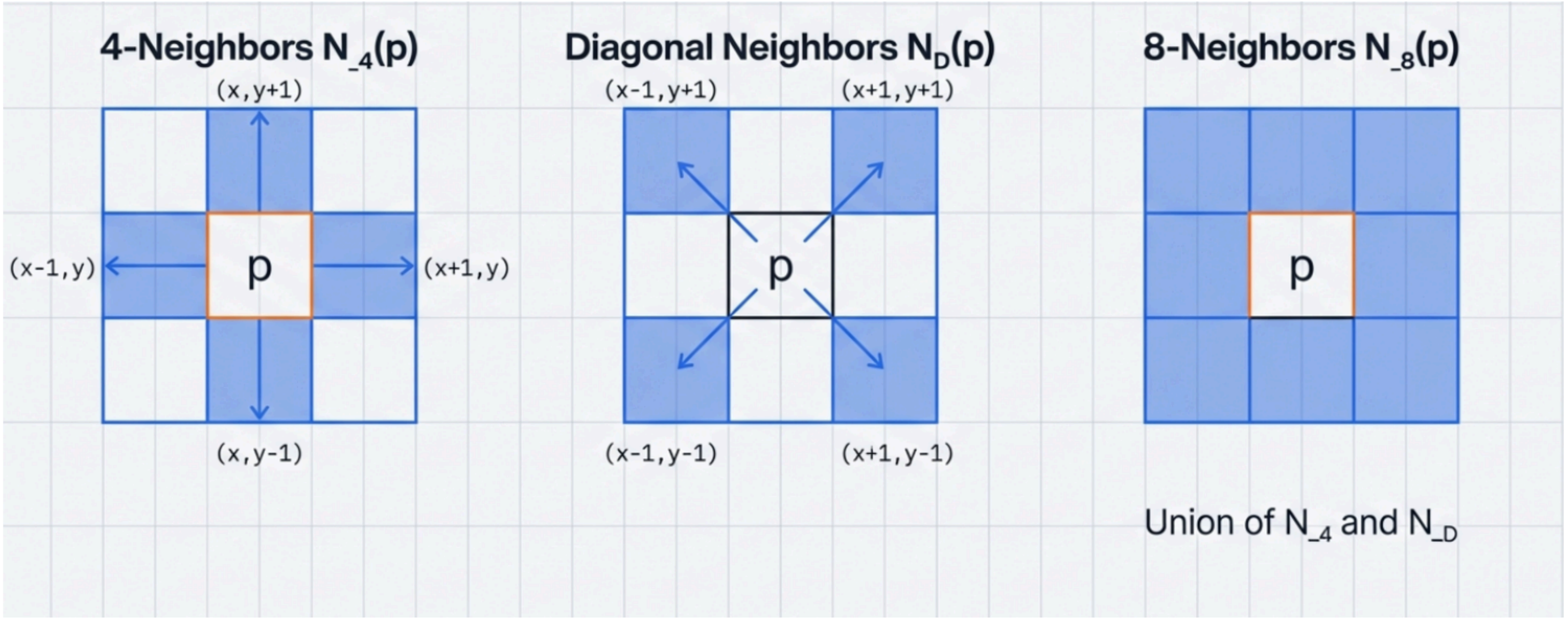
8-Neighbors $N_8(p)$

Union of N_4 and N_D:

$$N_8(p) = N_4(p) \cup N_D(p)$$



Pixel Neighbors — Illustration



Neighbor types for pixel p: 4-neighbors (top/bottom/left/right) and diagonal neighbors

Pixel Adjacency

Definition

Two pixels are **adjacent** if they are neighbors *and* their intensity values both belong to a specified set V.

4-Adjacency

Pixels are 4-neighbors with values in V.

8-Adjacency

Pixels are 8-neighbors with values in V.

m-Adjacency

Introduced to **eliminate ambiguity** in 8-adjacency paths.

The Problem with 8-Adjacency

Ambiguity in 8-Adjacency

Two pixels can be connected **both** via a diagonal link *and* via a horizontal/vertical path — creating redundant paths and closed loops.

- **Directly:** via the diagonal.
- **Indirectly:** e.g. go Right, then Down.

This redundancy confuses algorithms that need a single, unambiguous path.

Why m-Adjacency?

m-Adjacency eliminates redundant diagonal connections:

"Don't use the diagonal path if a horizontal/vertical path already exists."

m-Adjacency — Formal Rules

Pixels p and q with values from set V are **m-adjacent** if *either* condition holds:

Rule 1 — Horizontal/Vertical (4-neighbor)

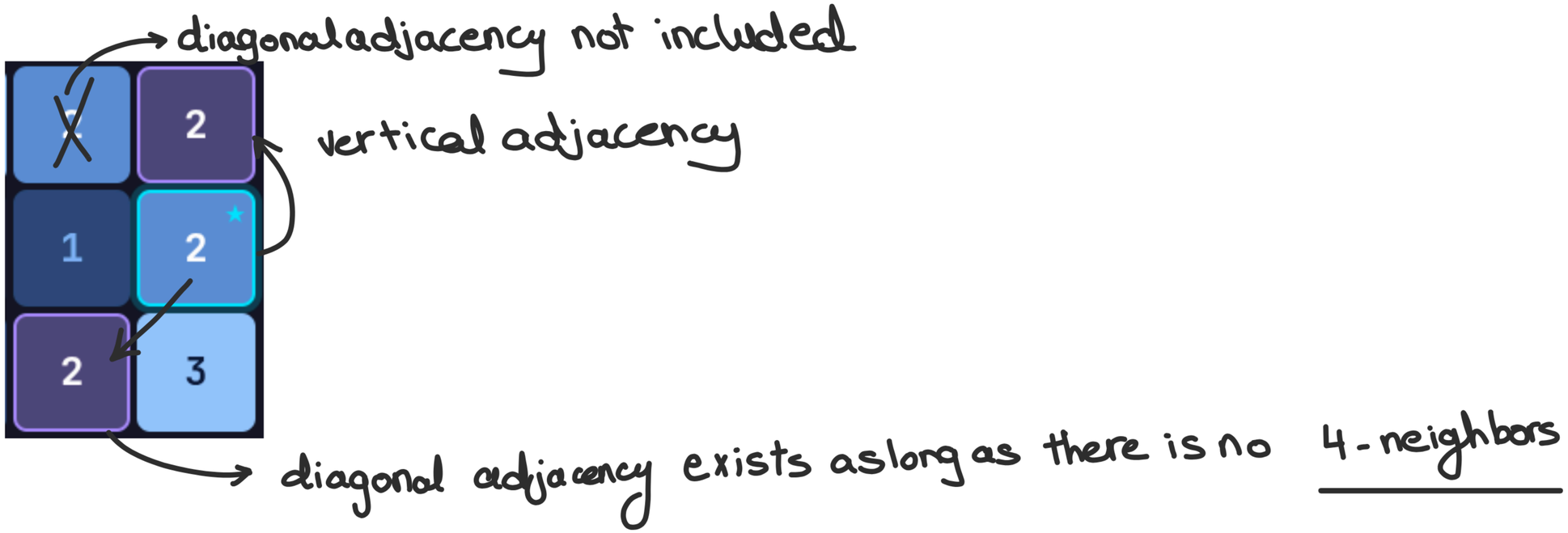
$q \in N_4(p)$ (standard horizontal or vertical connection)

Rule 2 — Diagonal (conditional)

$q \in N_D(p)$ **AND** the set $N_4(p) \cap N_4(q)$ contains **no** pixels with values in V.

$\Rightarrow$ The diagonal link is *only* allowed when no horizontal/vertical path exists.

m-Adjacency — Illustration



m-adjacency diagram: diagonal blocked when 4-neighbor path exists

m-Adjacency — Worked Example

Setup: $V = \{1\}$

$$\begin{matrix} 1_{(p)} & 1_{(A)} \\ 0_{(B)} & 1_{(q)} \end{matrix}$$

Solution

8-Connected? ✓ (diagonal neighbors)

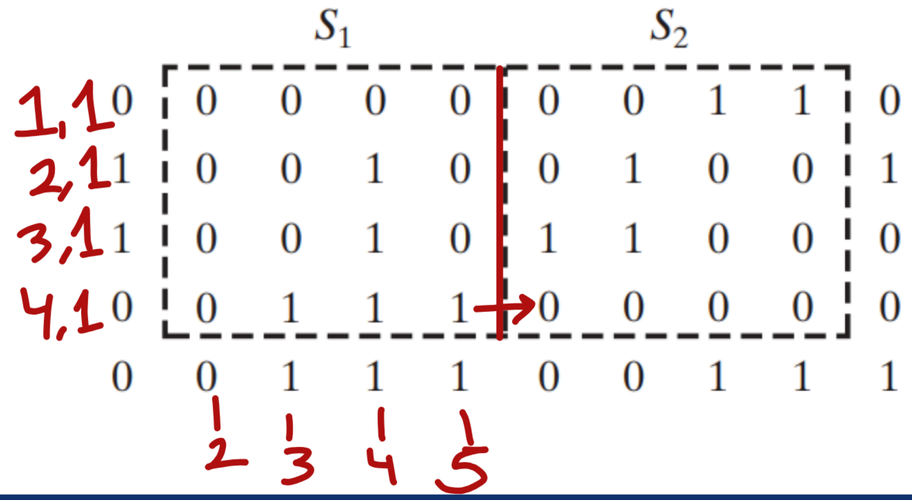
m-Connected? **No.**

A horizontal path $p \to A \to q$ exists, so the diagonal $p \to q$ is **blocked**.

Result: Only one path remains: $p \to A \to q$. Ambiguity resolved.

Textbook Problem 2.11 (3rd ed.)

★**2.11** Consider the two image subsets, S_1 and S_2, shown in the following figure. For $V = \{1\}$, determine whether these two subsets are (a) 4-adjacent, (b) 8-adjacent, or (c) m-adjacent.



Solution Space

a) 4- adj. doesn't have region

b) 8 adj between pixels (4,5) (3,6)

c) m adj Same as 8 adj

Textbook Problem 2.15 (3rd ed.)

2.15 Consider the image segment shown.

★(a) Let $V = \{0, 1\}$ and compute the lengths of the shortest 4-, 8-, and m-path between p and q. If a particular path does not exist between these two points, explain why.

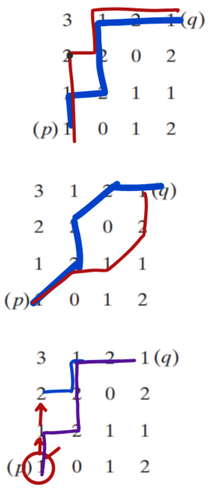
(b) Repeat for $V = \{1, 2\}$.

3	1	2	1 (q)
2	2	0	2
1	2	1	1
(p) 1	0	1	2

b) 4-path L=6

8-path L=4

m-path



Solution Space

a) V = {0, 1}

4-path doesn't exist

8-path (4,1)(4,2)(4,3)(3,3)(2,3)(1,4) L = 5 } unique

m-path → Same as path

Paths and Connectivity

Connected in S

Pixels p and q are **connected in subset** S if there is a path between them where every intermediate pixel belongs to S:

$$\forall\, i,\ 0 \leq i \leq n : \quad (x_i, y_i) \in S$$

Connected Component

The complete set of all pixels in S that are connected to a given pixel p.

Connected Set

A subset S where *every* pixel can reach every other pixel — i.e. only **one** component exists.

Regions and Image Structure

Region R

A **connected set** of pixels.

Adjacent vs. Disjoint Regions

Two regions are **adjacent** if their union forms a connected set; otherwise they are **disjoint**.

Image Terminology

- **Foreground:** Union of all K disjoint regions R_u
- **Background:** Complement $(R_u)^c$
- **Boundary:** Pixels inside R with at least one neighbor *outside* R
- **Hole:** Connected component of $(R_u)^c$ not touching the image border

Example: 5×5 Binary Image

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1_A} & \mathbf{1_B} & 0 & 0 \\ 0 & \mathbf{1_C} & \mathbf{1_D} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1_E} \end{bmatrix}$$

Analysis

- **Path:** $1_A \to 1_B \to 1_D$ connects A and D
- **Components:** Two — the 2×2 block $\{A, B, C, D\}$ and isolated E
- **Regions:** R_1 (block) and R_2 (E) are disjoint
- **Foreground/Background:** 1s $= R_u$; 0s $= (R_u)^c$
- **Boundary:** Every pixel in R_1 touches a 0 $\Rightarrow$ all of R_1 is its own boundary

What Makes a Valid Distance (Metric)?

Three Required Properties

For a function $D(p, q)$ to be a valid **metric**:

1. Non-negativity

$$D(p, q) \geq 0$$

Distance is never negative.

2. Symmetry

$$D(p, q) = D(q, p)$$

Distance A$\rightarrow$B equals B$\rightarrow$A.

3. Triangle Inequality

$$D(p, z) \leq D(p, q) + D(q, z)$$

Direct path is never longer.

Common Distance Metrics (1 & 2)

1. Euclidean Distance D_e — Straight-Line

$$D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$$

Pixels equidistant from a center form a **circle**.

2. City-Block Distance D_4 — Manhattan

$$D_4(p,q) = |x-s| + |y-t|$$

Like a taxi navigating a city grid — no diagonal shortcuts.
Equidistant pixels form a **diamond**.

Common Distance Metrics (3)

3. Chessboard Distance D_8

$$D_8(p,q) = \max\big(|x-s|,\ |y-t|\big)$$

Like a **King in chess** — diagonal moves cover both X and Y simultaneously.
Equidistant pixels form a **square**.

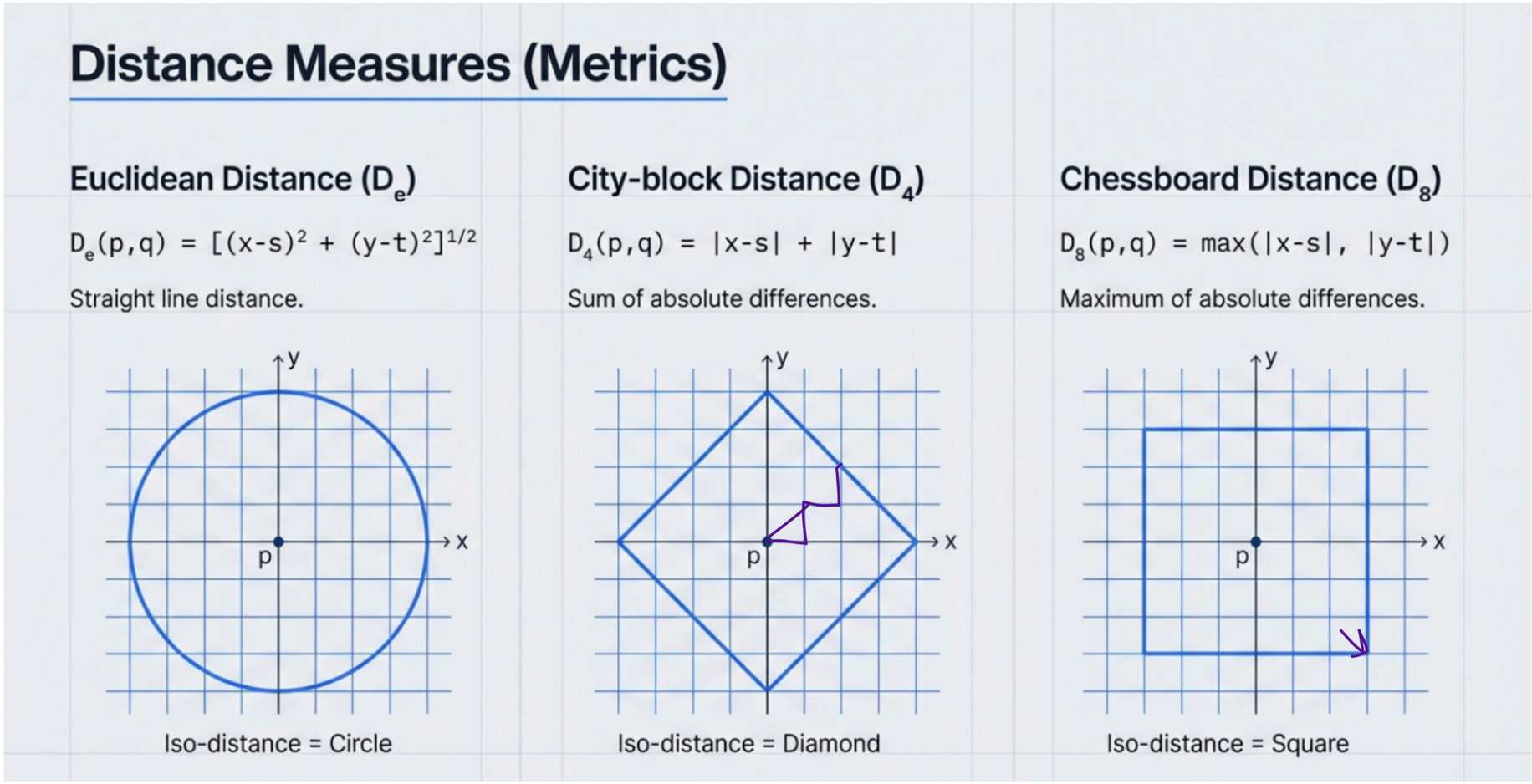
Worked Example: $p = (0,0)$, $q = (3,4)$

$$D_e = \sqrt{(0-3)^2 + (0-4)^2} = \sqrt{25} = \mathbf{5}$$

$$D_4 = |0-3| + |0-4| = 3 + 4 = \mathbf{7}$$

$$D_8 = \max(3,\ 4) = \mathbf{4}$$

Distance Measures — Visual Comparison



Equidistant pixel sets: circle (Euclidean), diamond (City-Block), square (Chessboard)

Student Activity — 10 minutes

Task

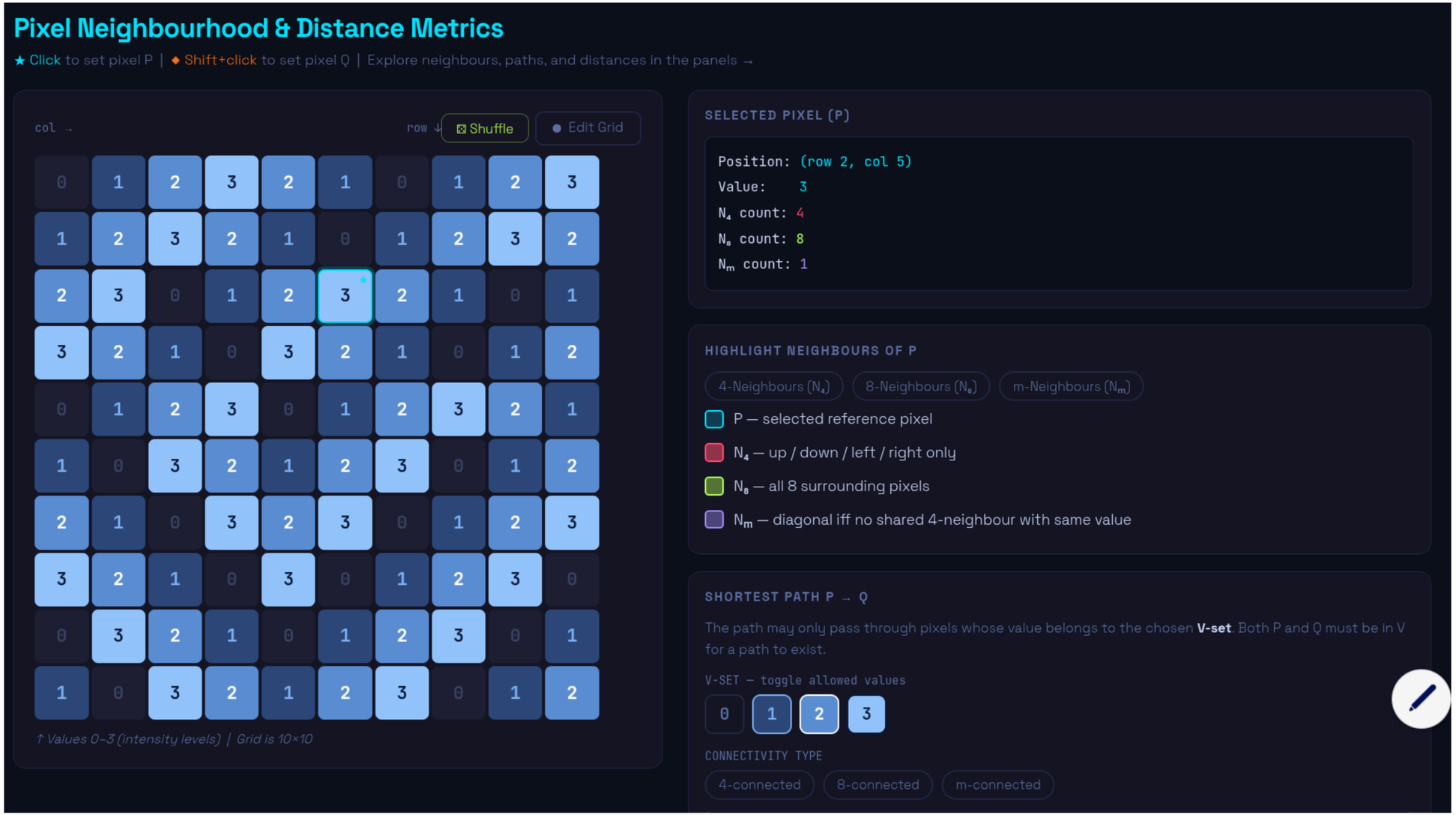
Using the interactive pixel grid (HTML file), choose **two pixels** that are moderately far apart, then find:

1. The shortest path between them using each adjacency type: 4, 8, and m
2. All three distances: D_e, D_4, D_8 (origin: row=0, column=0)

Submission

Draw the path on the grid (use Paint or equivalent).
Mark the **start pixel in blue** and the **end pixel in orange**.
Submit as a document.
Compare with the application.

Interactive Pixel Grid



Interactive pixel grid (see HTML file: Link Texthttps://3adyondawayer.github.io/ASU-Multimedia-Spring-2026/pixelinteractive.html)

Matrix Operations on Images

Element-wise (Pixel-wise)

Each pixel is operated on independently.

Default assumption throughout this course.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

Element-wise operation example (each pixel treated individually)

Normal Matrix Multiplication

Standard linear algebra matrix product.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Normal matrix multiplication example

Linearity: Additivity & Homogeneity

Additivity

Operator H is additive if:

$$\boxed{H[f_1(x,y) + f_2(x,y)] = H[f_1(x,y)] + H[f_2(x,y)]}$$

Homogeneity (Scaling)

Operator H is homogeneous if:

$$\boxed{H[a \cdot f(x,y)] = a \cdot H[f(x,y)]}$$

Complete Linearity

H is **linear** if and only if *both* properties hold simultaneously:

$$\boxed{H[a \cdot f_1 + b \cdot f_2] = a \cdot H[f_1] + b \cdot H[f_2]}$$

Example: Is $H[f] = 2f$ Linear?

Additivity Test

$$\text{Left: } H[f_1 + f_2] = 2(f_1 + f_2) = 2f_1 + 2f_2$$
$$\text{Right: } H[f_1] + H[f_2] = 2f_1 + 2f_2$$

✓ Additivity holds.

Homogeneity Test

$$\text{Left: } H[af] = 2(af) = 2af$$
$$\text{Right: } aH[f] = a \cdot 2f = 2af$$

✓ Homogeneity holds.

Conclusion

$H[f(x, y)] = 2f(x, y)$ is **LINEAR**. ✓

Counterexamples: Nonlinear Operators (1)

Squaring Operator: $H[f] = [f]^2$

$$\text{Left:} \quad H[f_1 + f_2] = (f_1 + f_2)^2 = f_1^2 + \underbrace{2f_1 f_2}_{\text{cross term}} + f_2^2$$

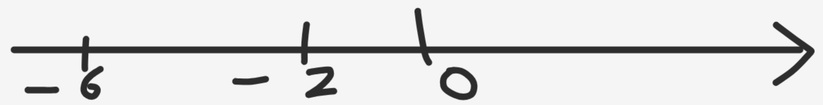
$$\text{Right:} \quad H[f_1] + H[f_2] = f_1^2 + f_2^2$$

× Additivity fails (cross term $2f_1 f_2$ appears). **Nonlinear.**

Max Operator: test with $a = 1$, $b = -1$, $f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$, $f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}$

$$\text{Left: } \max\{f_1 - f_2\} = \max \begin{bmatrix} -6 & -3 \\ -2 & -4 \end{bmatrix} = -2 \quad \neq \quad \text{Right: } 1\cdot 3 + (-1)\cdot 7 = -4$$

× Additivity fails. ⇒ **Nonlinear.**



Counterexamples: Nonlinear Operators (2)

Median Operator

$S_1 = \{1, -2, 3\}, \quad S_2 = \{4, 5, 6\}$:

-2, 1, 3 4, 5, 6

Left: $\text{median}(S_1 + S_2) = \text{median}\{5, 3, 9\} = 5 \neq$ Right: $\text{median}(S_1) + \text{median}(S_2) = 1 + 5 = 6$

3, 5, 9

× **Linearity fails.**

Element-wise Multiplication: $H[f] = f \times f$

$$H[af_1 + bf_2] = a^2(f_1 \times f_1) + 2ab(f_1 \times f_2) + b^2(f_2 \times f_2)$$
$$aH[f_1] + bH[f_2] = a(f_1 \times f_1) + b(f_2 \times f_2)$$

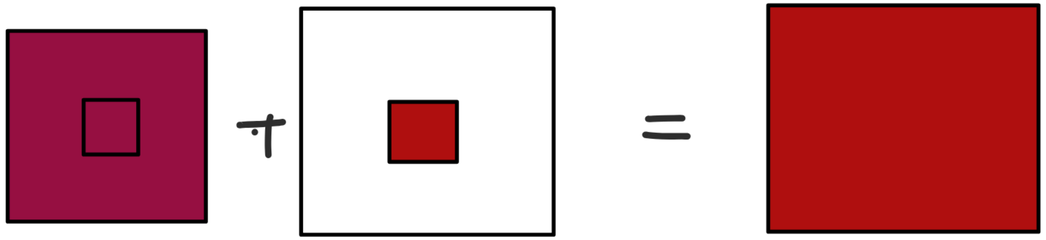
× Cross term $2ab(f_1 \times f_2)$ breaks linearity. **Nonlinear.**

Arithmetic (Array) Operations

Four Basic Pixel-Wise Operations

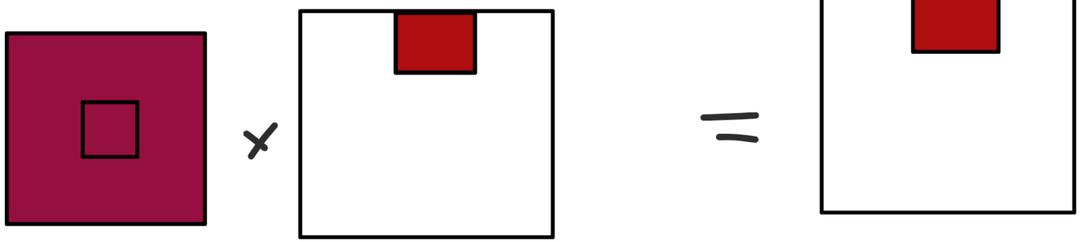
$$\begin{aligned}
&\textbf{Sum:} && s(x,y) = f(x,y) + g(x,y) \\
&\textbf{Difference:} && d(x,y) = f(x,y) - g(x,y) \\
&\textbf{Product:} && p(x,y) = f(x,y) \times g(x,y) \\
&\textbf{Quotient:} && v(x,y) = f(x,y) \div g(x,y)
\end{aligned}$$

Each operation is performed **element-wise** — pixel by pixel.



White → 0

red → 1



Noise Reduction by Image Averaging

Core Idea

If you average K noisy images of the same static scene:

- **Noise** is random $\Rightarrow$ cancels out over many images.
- **Signal** is constant $\Rightarrow$ reinforced.

The Averaging Formula

$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y)$$

where $g_i(x,y) = f(x,y) + n_i(x,y)$ is the i-th noisy observation.

Expected Value — Unbiasedness

Assuming noise $n_i(x, y)$ is **i.i.d. with zero mean**:

$$E\{\bar{g}(x, y)\} = E\left\{\frac{1}{K}\sum_{i=1}^{K}[f(x, y) + n_i(x, y)]\right\}$$

$$= f(x, y) + \frac{1}{K}\sum_{i=1}^{K}\underbrace{E\{n_i(x, y)\}}_{=0}$$

$$= f(x, y)$$

Key Takeaway

Averaging eventually recovers the **noise-free** image as $K \to \infty$.

Variance Reduction

Useful Identities

$$\mathrm{Var}(aX) = a^2\,\mathrm{Var}(X), \qquad \mathrm{Var}(\sum X_i) = \sum \mathrm{Var}(X_i)$$

$$\sigma_{\bar{g}}^2 = \mathrm{Var}\left(\frac{1}{K}\sum_{i=1}^{K} n_i\right) = \frac{1}{K^2}\cdot K\,\sigma_n^2 = \frac{\sigma_n^2}{K}$$

Interpretation

- Averaging K images reduces noise **variance** by a factor of K.
- Averaging 100 images $\Rightarrow$ noise variance is $100\times$ smaller.

Textbook Problem 2.25 (4th ed.)

2.25* Show that image averaging can be done recursively. That is, show that if $a(k)$is the average of k images, then the average of $k+1$ images can be obtained from the already-computed average, $a(k)$, and the new image, f_{k+1}.

Textbook Problem 2.25 statement

Solution: Recursive Averaging Formula

$$\boxed{a(k+1) = \frac{1}{k+1}\left[k \cdot a(k) + f_{k+1}\right]}$$

Derived by splitting the sum of $k+1$ terms into the previous average $a(k)$ plus the new image f_{k+1}.

Textbook Problem 2.25 Solution Steps

Solution Space

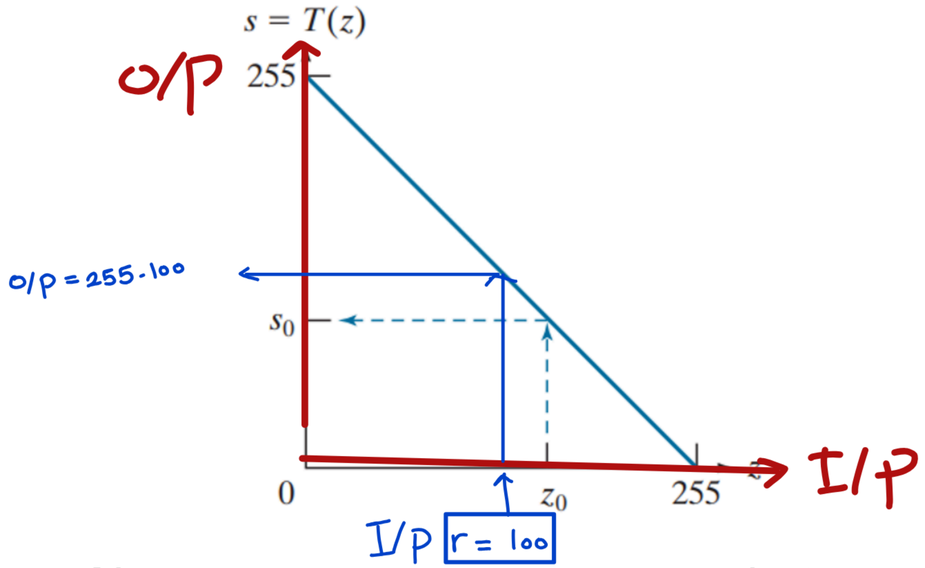
$$a(K) = \frac{1}{K} \sum_{j=1}^{K} f_j \longrightarrow K = K+1 \qquad a(K+1) = \frac{1}{K+1} \sum_{j=1}^{K+1} f_j$$

$$a(K+1) = \frac{1}{K+1} \left[\sum_{j=1}^{K} f_j + f_{K+1} \right]$$

$$\sum_{j=1}^{K} f_j = K\, a(K)$$

Negative Operation

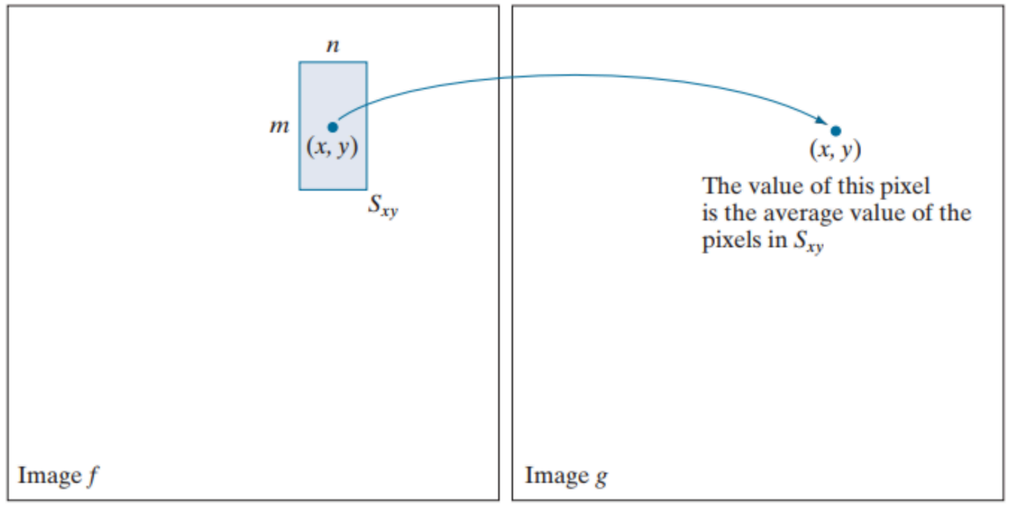
Inverts pixel intensities (complement).



Negative operation on a sample image

Averaging Kernel (Smoothing)

Each output pixel = average of its local neighborhood.



Averaging kernel / smoothing filter applied to image

Geometric Transformations — Overview

Two Main Steps

1. **Spatial transformation** of pixel coordinates
2. **Intensity interpolation** — assigning intensity values to the transformed locations

Forward Mapping

Scan input f, compute where each pixel (v, w) maps to in output g:

$$(x, y) = T\{(v, w)\}$$

Problem: multiple input pixels may land on the same output location.

Inverse Mapping

Scan output g, look up corresponding input location:

$$(v, w) = T^{-1}\{(x, y)\}$$

Generally **more efficient** — interpolation from nearby input pixels.

Image Registration — Main Idea

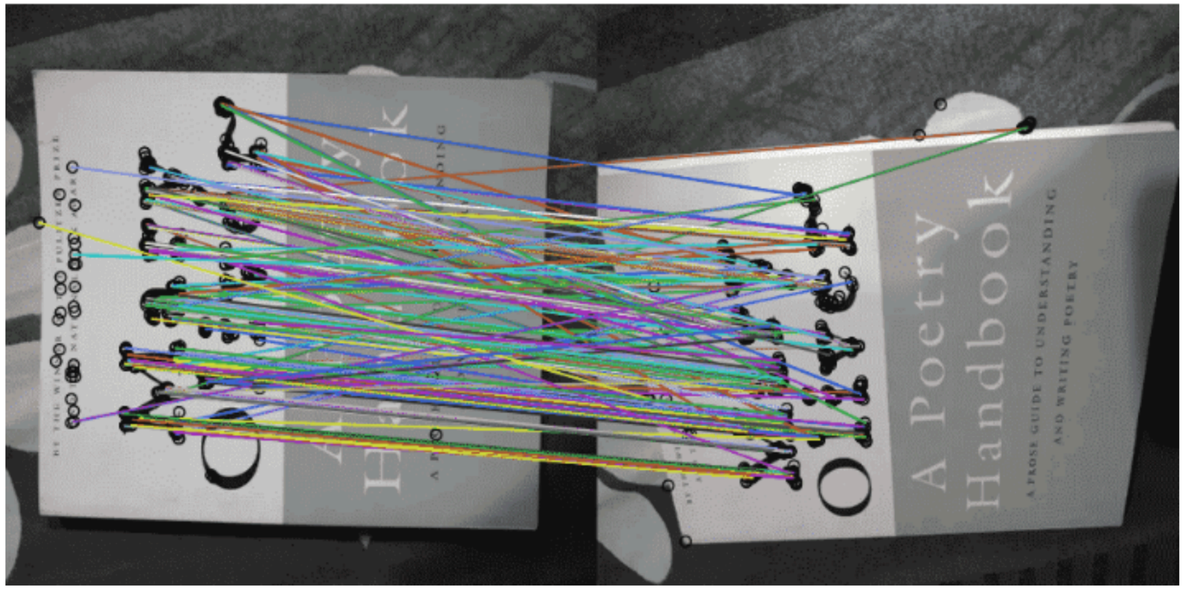
Process Overview

1. **Detection:** Find distinct reference points (tie points) in both images I_{source} and I_{target}
2. **Matching:** Pair up corresponding points
3. **Transformation:** Compute mapping matrix M from matched pairs
4. **Warping:** Apply M to produce the aligned image:

$$I_{\text{aligned}} = M \cdot I_{\text{source}}$$

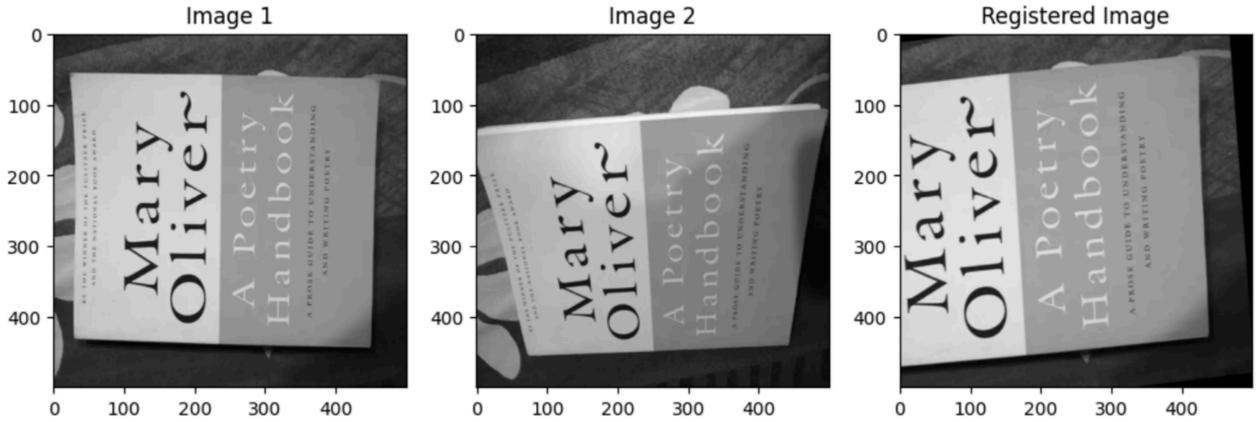
Image Registration — Tie Points & Output

Tie Points



Tie points marked on source and target images

Registered Output



Aligned/registered output image overlaid on target

Affine Transformations

Affine transforms cover: **scaling, translation, rotation, and shearing** — expressed as a single matrix multiplication.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

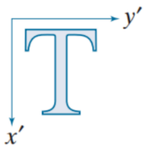
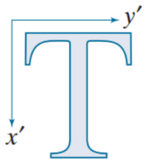
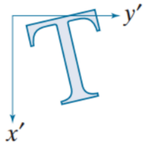
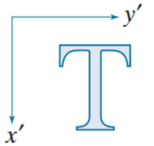
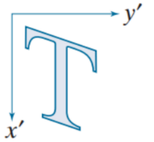
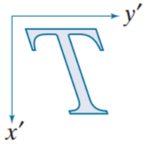
$x' = a_{11} x$

Prerequisite

Requires familiarity with linear algebra and matrix operations.

Affine Transformations — Special Cases

TABLE 2.3 Affine transformations based on Eq. (2-45).

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to –1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

T_S

T_R

$T_R T_S T_R$
$+45^\circ \quad -45^\circ$

Table of affine transformation types: scaling, translation, rotation, shearing with their matrices

Textbook Example 2.36 (4th ed.) — Class Task

2.36 With reference to Table 2.3, provide single, composite transformation functions for performing the following operations:

(a)* Scaling and translation.

(b)* Scaling, translation, and rotation.

(c) Vertical shear, scaling, translation, and rotation.

(d) Does the order of multiplication of the individual matrices to produce a single transformations make a difference? Give an example based on a scaling/translation transformation to support your answer. Yes

Solution Space

a) $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$

scaling

Textbook Example 2.37 (4th ed.)

2.37

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$X'_{Trans} = A\, X_{orig}$

$X = A^{-1} X'$

(a)* Find the inverse scaling transformation.

(b) Find the inverse translation transformation.

(c) Find the inverse vertical and horizontal shearing transformations.

(d)* Find the inverse rotation transformation.

(e)* Show a composite inverse translation/rotation transformation.

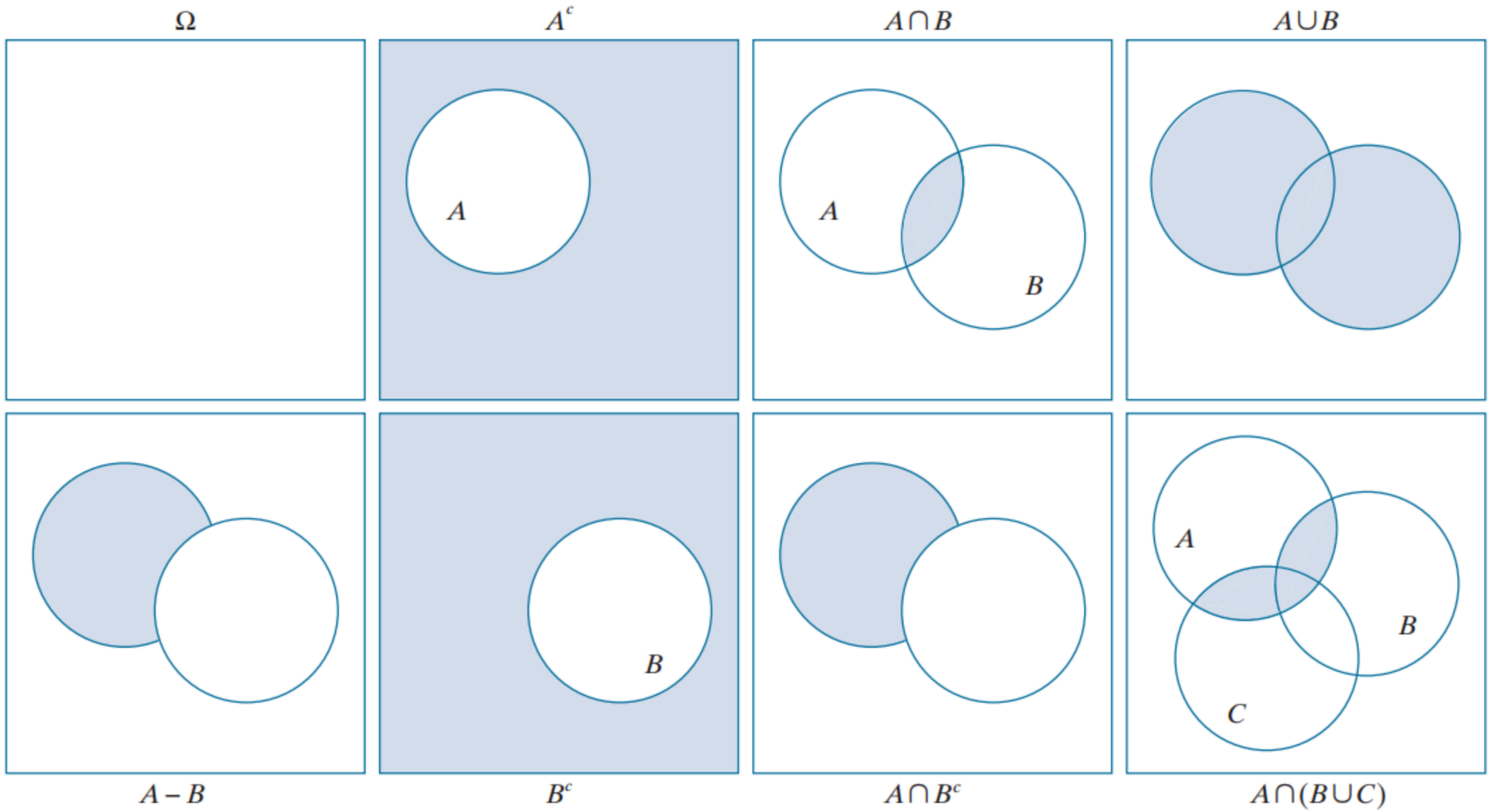
Solution Space

Set Theory — Reference 1

TABLE 2.1
Some important set operations and relationships.

Description	Expressions
Operations between the sample space and null sets	$\Omega^c = \varnothing;\ \varnothing^c = \Omega;\ \Omega \cup \varnothing = \Omega;\ \Omega \cap \varnothing = \varnothing$
Union and intersection with the null and sample space sets	$A \cup \varnothing = A;\ A \cap \varnothing = \varnothing;\ A \cup \Omega = \Omega;\ A \cap \Omega = A$
Union and intersection of a set with itself	$A \cup A = A;\ A \cap A = A$
Union and intersection of a set with its complement	$A \cup A^c = \Omega;\ A \cap A^c = \varnothing$
Commutative laws	$A \cup B = B \cup A$ $A \cap B = B \cap A$
Associative laws	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributive laws	$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$
DeMorgan's laws	$(A \cup B)^c = A^c \cap B^c$ $(A \cap B)^c = A^c \cup B^c$

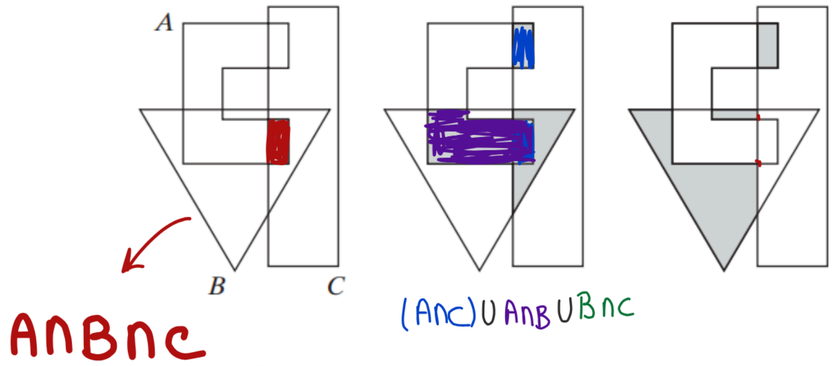
Set Theory — Reference 2



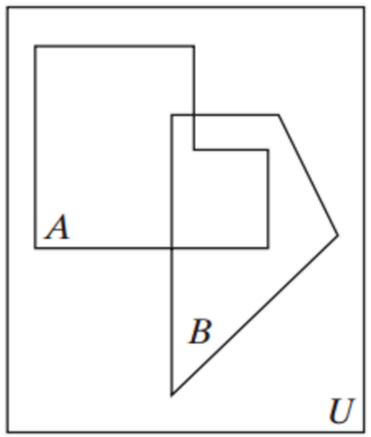
Textbook Problem 2.23 (3rd ed.) — Section Task

2.23 ★(a) With reference to Fig. 2.31, sketch the set $(A \cap B) \cup (A \cup B)^c$.

(b) Give expressions for the sets shown shaded in the following figure in terms of sets A, B, and C. The shaded areas in each figure constitute one set, so give one expression for each of the three figures.

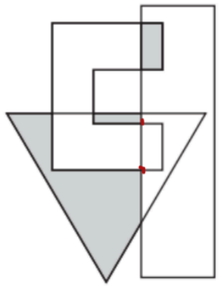


Problem 2.23 — set A



Task

Solve using Paint or any image editor. Draw the result directly on the image.



Problem 2.23 — solution/result image

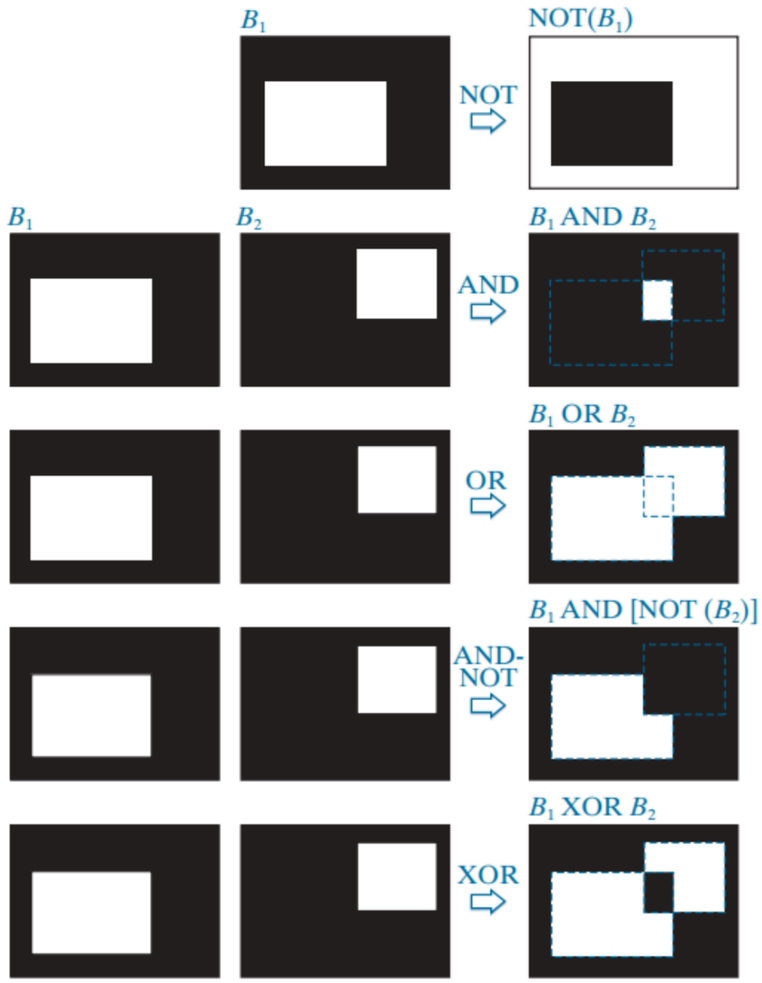
Logical Operations on Binary Images

a	b	**AND** $(a \wedge b)$	**OR** $(a \vee b)$	**NOT** $(\sim a)$	**XOR** $(a \oplus b)$	**XNOR** $(a \odot b)$
0	0	0	0	1	0	1
0	1	0	1	1	1	0
1	0	0	1	0	1	0
1	1	1	1	0	0	1

- **AND:** 1 only if *both* inputs are 1
- **OR:** 1 if *at least one* input is 1
- **NOT:** Inverts the input
- **XOR:** 1 only if inputs are *different*
- **XNOR:** 1 only if inputs are the *same* (inverse of XOR)

Logical Operations — Visual Examples

FIGURE 2.37
Illustration of logical operations involving foreground (white) pixels. Black represents binary 0's and white binary 1's. The dashed lines are shown for reference only. They are not part of the result.



Visual examples of AND, OR, NOT, XOR applied to binary images

End of Section 2

Digital Image Processing

Questions?