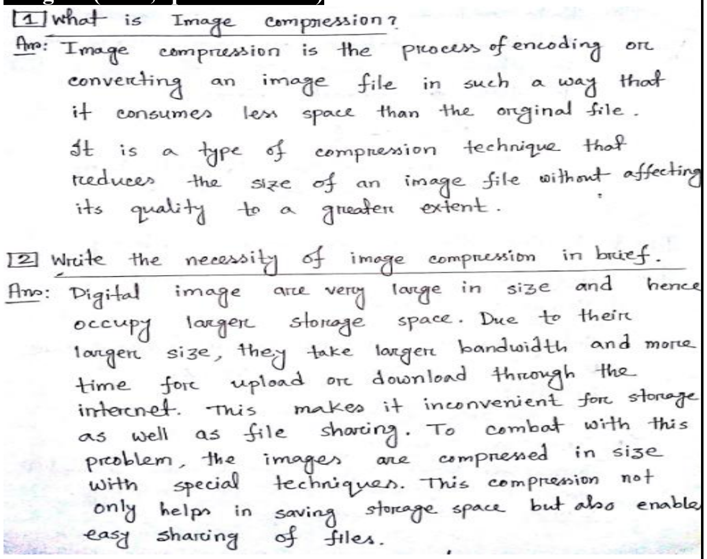
Bismillahhir Rahmanir Rahim

# Pattern Recognition and Image Processing || Part- A || Final Note

**Thanks to-** C201032, C201023, C201035, C201039, C201132

**1.a) Define image compression. Why we need to compress digital images? (au22, sp23 – 3 marks)**



[1] What is Image compression?

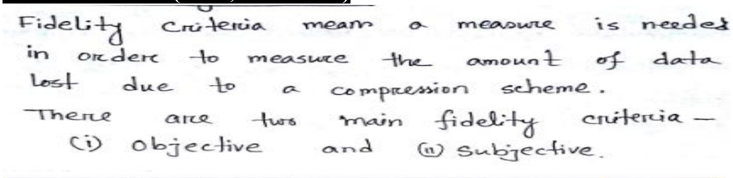
Ans: Image compression is the process of encoding or converting an image file in such a way that it consumes less space than the original file.

It is a type of compression technique that reduces the size of an image file without affecting its quality to a greater extent.

[2] Write the necessity of image compression in brief.

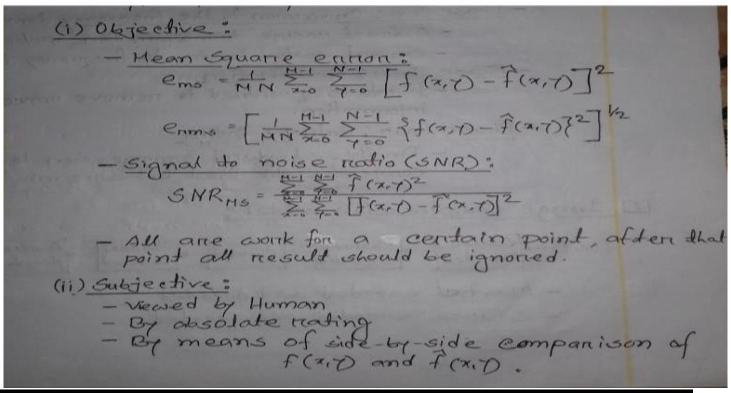
Ans: Digital image are very large in size and hence occupy larger storage space. Due to their larger size, they take larger bandwidth and more time for upload or download through the internet. This makes it inconvenient for storage as well as file sharing. To combat with this problem, the images are compressed in size with special techniques. This compression not only helps in saving storage space but also enable easy sharing of files.

**OR, 1.a) What are Fidelity criteria? Describe different Fidelity criteria in brief. (au22, – 3 marks)**



Fidelity criteria mean a measure is needed in order to measure the amount of data lost due to a compression scheme.

There are two main fidelity criteria — (i) Objective and (ii) Subjective.



(i) Objective:

— Mean Square error:

$$e_{ms} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)-\hat{f}(x,y)\right]^2$$

$$e_{rms} = \left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\{f(x,y)-\hat{f}(x,y)\}^2\right]^{1/2}$$

— Signal to noise ratio (SNR):

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)-\hat{f}(x,y)\right]^2}$$

— All are work for a certain point, after that point all result should be ignored.

(ii) Subjective:

— Viewed by Human
— By absolate rating
— By means of side-by-side comparison of $f(x,y)$ and $\hat{f}(x,y)$.

**1.A) Data combination of information and redundant data" - do you agree? Justify your answer with proper example. (sp22, au'21– 3 marks)**

**Yes, I agree that data is a combination of information and redundant data**. Let me justify my answer with an example:

Data refers to raw facts, observations, or measurements that have been collected but do not provide any context or meaning on their own. When this data is processed and organized in a meaningful way, it becomes information. Information is data that has been interpreted, analyzed, and presented in a way that is useful and valuable for decision-making or understanding.

However, within a dataset, there can be instances of redundant data. Redundant data refers to information that is duplicated or unnecessary for the intended purpose of the dataset. It does not add any additional value or new insights but occupies storage space and increases processing time.

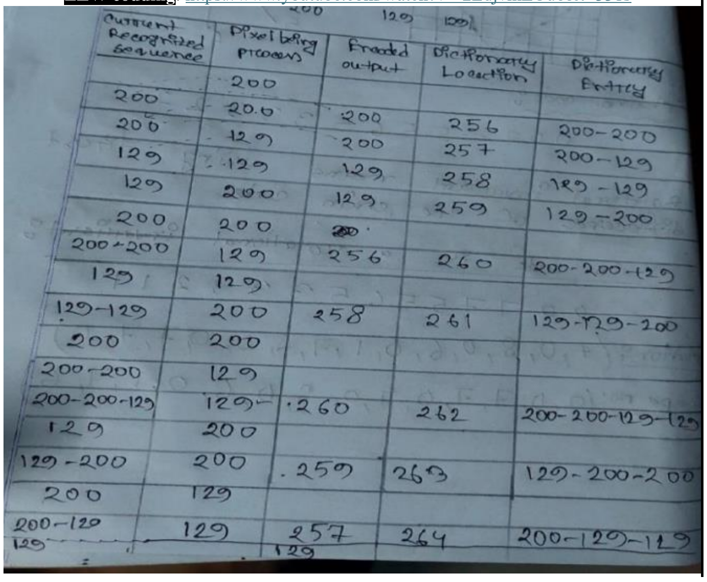
For example, let's consider a customer database for an e-commerce company. Each customer entry contains fields such as name, address, email, and order history. The name and address fields provide essential information about the customers. However, if there are multiple entries for the same customer with identical information, it creates redundancy. Storing and processing redundant data in this case would be inefficient and could lead to inaccuracies or inconsistencies if updates or changes are not uniformly applied across all redundant instances.

Therefore, while data can contain both valuable information and redundant data, it is important to identify and eliminate or minimize redundant elements to ensure efficient storage, processing, and analysis of the dataset.

**1.b) Encode the following 4 x 4, 8-bit image using LZW coding: (sp22, – 4 marks)**

| | | | |
|---|---|---|---|
| 200 | 200 | 129 | 129 |
| 200 | 200 | 129 | 129 |
| 200 | 200 | 129 | 129 |
| 200 | 200 | 129 | 129 |

**LZW codding**: https://www.youtube.com/watch?v=-2EdjvmZOuo&t=331s



| Current Recognized Sequence | Pixel being Process | Encoded output | Dictionary Location | Dictionary Entry |
|---|---|---|---|---|
| | 200 | | | |
| 200 | 20.0 | 200 | 256 | 200-200 |
| 200 | 129 | 200 | 257 | 200-129 |
| 129 | 129 | 129 | 258 | 129-129 |
| 129 | 200 | 129 | 259 | 129-200 |
| 200 | 200 | ~~200~~ | | |
| 200-200 | 129 | 256 | 260 | 200-200-129 |
| 129 | 129 | | | |
| 129-129 | 200 | 258 | 261 | 129-129-200 |
| 200 | 200 | | | |
| 200-200 | 129 | | | |
| 200-200-129 | 129 | 260 | 262 | 200-200-129-129 |
| 129 | 200 | | | |
| 129-200 | 200 | 259 | 263 | 129-200-200 |
| 200 | 129 | | | |
| 200-120 129 | 129 | 257 129 | 264 | 200-129-129 |

**Encoded output-**

200 200 129 129
256 258 260 259
257 129

**OR. 1a): Describe image compression model in brief (sp23, – 3 marks)**

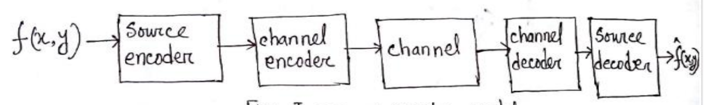
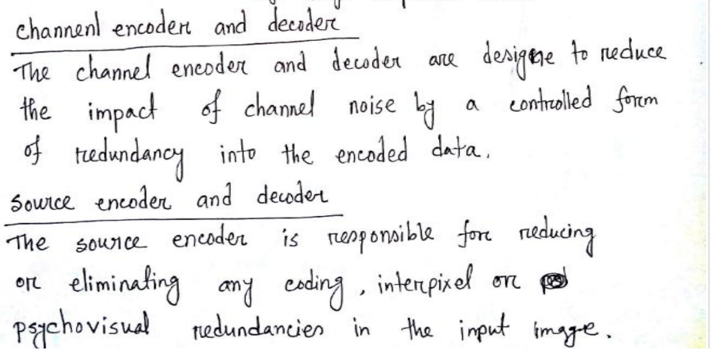


Fig: Image compression model.



Channenl encoder and decoder

The channel encoder and decoder are designe to reduce the impact of channel noise by a controlled form of redundancy into the encoded data.

Source encoder and decoder

The source encoder is responsible for reducing or eliminating any coding, interpixel or psychovisual redundancies in the input image.

**Or 1.b)) Describe in details how color-based segmentation can be used in an attempt to segment "same-colored" object in real color images. What problems arise? What are some ways of dealing with them? (sp22, – 4 marks)**

Color-based segmentation is a technique used to segment objects of the same color in real color images. It involves identifying regions in an image that have similar color characteristics and grouping them together as the same-colored object. Here is a detailed description of how color-based segmentation can be applied and the problems that may arise, along with potential solutions:

**Color Space Selection:** The first step is to select an appropriate color space for representing the image. Commonly used color spaces include RGB, HSV, and Lab. Each color space has its own advantages in terms of separating color information and can affect the segmentation results.
**Color Thresholding:** Once the color space is chosen, color thresholding is performed by defining a range of acceptable color values for the desired object. Pixels falling within this range are considered part of the same-colored object. However, several challenges can arise:
**Illumination Variations:** Changes in lighting conditions can cause variations in color appearance. To address this, techniques such as histogram equalization, color normalization, or color constancy algorithms can be applied to reduce the impact of illumination variations.
**Similar Color Interference:** Objects with different colors may appear similar due to factors such as lighting conditions, texture patterns, or background clutter. Additional features such as texture, shape, or spatial relationships can be incorporated to differentiate between objects of similar colors.
**Noise and Artifacts:** Image noise and artifacts can negatively impact color-based segmentation. Preprocessing steps like noise reduction techniques (e.g., median filtering) or smoothing filters can help reduce the impact of noise and improve the segmentation results.
**Parameter Selection**: The selection of color thresholds and segmentation parameters is crucial for achieving accurate results. These parameters may vary depending on the specific image and object characteristics. Techniques such as adaptive thresholding or statistical approaches can be used to dynamically determine suitable thresholds.
**Post-processing:** After initial segmentation, post-processing steps can be applied to refine the results. These steps may include morphological operations (e.g., erosion, dilation) to remove small noise or gaps, region merging/splitting techniques for better connectivity, or contour smoothing to improve the object boundaries.
**Evaluation and Validation:** It is important to evaluate and validate the segmentation results using appropriate metrics and ground truth data. This helps assess the accuracy and effectiveness of the color-based segmentation method and make necessary adjustments if required.

While color-based segmentation is effective for segmenting same-colored objects, it has limitations. Some challenges include variations in lighting, similar colors of different objects, and noise interference. To address these challenges, techniques like illumination normalization, incorporating additional features, noise reduction, parameter tuning, and post-processing steps can be employed. Additionally, combining color-based segmentation with other segmentation methods or machine learning approaches can further enhance the segmentation accuracy and robustness in real-color images

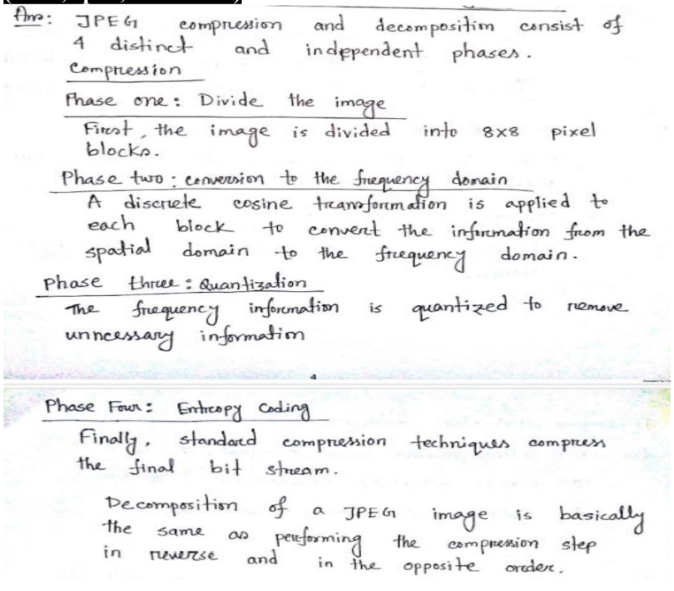
**1.c) What is Bit Plane decomposition? Write the benefit of bit-plane decomposition with necessary example. (sp22, – 3 marks)**

Bit plane decomposition is a technique used in image processing to decompose an image into its individual bit planes. In this technique, the binary representation of each pixel in the image is separated into different bit planes, with each bit plane containing the binary values of a specific bit position.
**The benefit of bit plane decomposition is that** it provides a way to analyze and manipulate the image at different levels of detail, based on the significance of the bits. Here's an example to illustrate the benefit:

***Let's consider a grayscale image where each pixel value ranges from 0 to 255***, represented by 8 bits (1 byte). By performing bit plane decomposition, we can extract the individual bit planes for each pixel value. For instance, consider a pixel value of 150, which in binary representation is 10010110. Through bit plane decomposition, we obtain the following bit planes:
Bit Plane 7: 1 0 0 1 0 1 1 0 (Most Significant Bit - MSB)
Bit Plane 6: 0 0 0 0 0 0 0 0
Bit Plane 5: 0 0 0 0 0 0 0 0
Bit Plane 4: 0 0 0 0 0 0 0 0
Bit Plane 3: 1 0 0 1 0 1 1 0
Bit Plane 2: 0 0 0 0 0 0 0 0
Bit Plane 1: 1 0 0 0 0 0 0 0
Bit Plane 0: 0 0 0 0 0 0 0 0 (Least Significant Bit - LSB)

Each bit plane represents a different level of information present in the image. The MSB (bit plane 7) contains the most significant information, while the LSB (bit plane 0) contains the least significant information. **The benefit of bit plane decomposition becomes apparent when we observe the impact of manipulating or discarding certain bit planes. For example:**
**Image Compression:** By discarding the bit planes with lower significance (e.g., bit planes 0 to 3), we can achieve lossy compression of the image, reducing the storage requirements without significant degradation in visual quality.
**Image Enhancement:** By manipulating or applying image processing operations on specific bit planes, we can enhance specific details or features in the image. For example, amplifying or attenuating the higher bit planes (e.g., bit planes 5 to 7) can enhance fine details or emphasize edges in the image.
**Image Watermarking:** Bit plane decomposition can be utilized in watermarking techniques. By embedding watermarks in specific bit planes, the watermark can be selectively extracted or detected without affecting the overall image appearance.

**1.c) Write the steps of JPEG algorithm with necessary example. (au22, sp23, – 3 marks)**


Ans: JPEG compression and decomposition consist of 4 distinct and independent phases.
Compression
Phase one: Divide the image
First, the image is divided into 8x8 pixel blocks.
Phase two: conversion to the frequency domain
A discrete cosine transformation is applied to each block to convert the information from the spatial domain to the frequency domain.
Phase three: Quantization
The frequency information is quantized to remove unncessary information
Phase Four: Entropy Coding
Finally, standard compression techniques compress the final bit stream.
Decomposition of a JPEG image is basically the same as performing the compression step in reverse and in the opposite order.

**or 2(a) i) In the pattern recognition and classification there are four basic steps involved, which are sensing an image, segmenting the image, extracting the features from the segmented objects, and classification to recognize the specific object. Based on this procedure explain how you can segment and classify a specific object from an 3 bit color image. (sp22, – 4+2 marks)**

To segment and classify a specific object from an 8-bit color image, we can follow the basic steps of pattern recognition and classification: sensing, segmentation, feature extraction, and classification. Here's a detailed explanation of the procedure:

**1. Sensing the Image:**
Capture the 8-bit color image using a sensing device, such as a camera or scanner. Ensure proper lighting conditions and capture the object of interest with sufficient image resolution.

**2. Segmentation:**
Apply segmentation techniques to separate the object from the background:

**Color-based Segmentation:** Set a threshold on the color channels to distinguish the object from the background based on color intensity. For example, you can use color thresholding techniques to identify regions that fall within a specified color range.

**Region-based Segmentation:** Utilize algorithms such as region growing, graph cuts, or watershed segmentation to group pixels with similar color characteristics that belong to the object.

**Edge-based Segmentation:** Employ edge detection algorithms like Canny or Sobel to identify object boundaries by detecting significant changes in color intensity or gradients.

**3. Feature Extraction:**

Extract relevant features from the segmented object. These features should capture discriminative information for classification.

In the case of an 8-bit color image, features can be derived from color attributes such as:

**Color Histogram**: Calculate the distribution of color intensities or color channels to represent the color distribution within the object.

Color Moments: Compute statistical moments, such as mean, standard deviation, or higher-order moments, to describe the color properties of the object.

**Color Texture Descriptors:** Analyze texture patterns in the color channels, such as Local Binary Patterns (LBP) or Haralick features, to capture textural information.

**4. Classification:**

Utilize a classification algorithm to recognize the specific object based on the extracted features.

Various classification techniques can be employed, including:

**Supervised Learning**: Train a classifier, such as Support Vector Machines (SVM), Random Forests, or Neural Networks, using labeled training data with known object classes. Then, apply the trained classifier to classify new objects.

**Unsupervised Learning:** Utilize clustering algorithms like k-means or Gaussian Mixture Models (GMM) to group similar objects together based on extracted features.

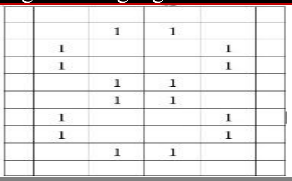
**Deep Learning:** Apply deep neural network architectures, such as Convolutional Neural Networks (CNNs), for end-to-end feature learning and object classification.

The classification step will provide the recognition or classification result, indicating the specific object present in the 8-bit color image.

It's important to note that the choice of segmentation techniques, feature extraction methods, and classification algorithms can vary depending on the specific characteristics of the objects and the application domain. Experimentation, evaluation, and fine-tuning of each step may be necessary to achieve accurate segmentation and classification results for a given 8-bit color image.

ii) Performa the region filling algorithm on the following image. Explain step by step how this algorithm works on this image. (sp22,

Or 2.b) Perform the region filling algorithm on the following image. Explain step by step how this algorithm works on this image(sp23, 3 marks)



| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | 1 | 1 | | |
| | 1 | | | 1 | |
| | 1 | | | 1 | |
| | | 1 | 1 | | |
| | | 1 | 1 | | |
| | 1 | | | 1 | |
| | 1 | | | 1 | |
| | | 1 | 1 | | |
| | | | | | |

**Ans: 1**

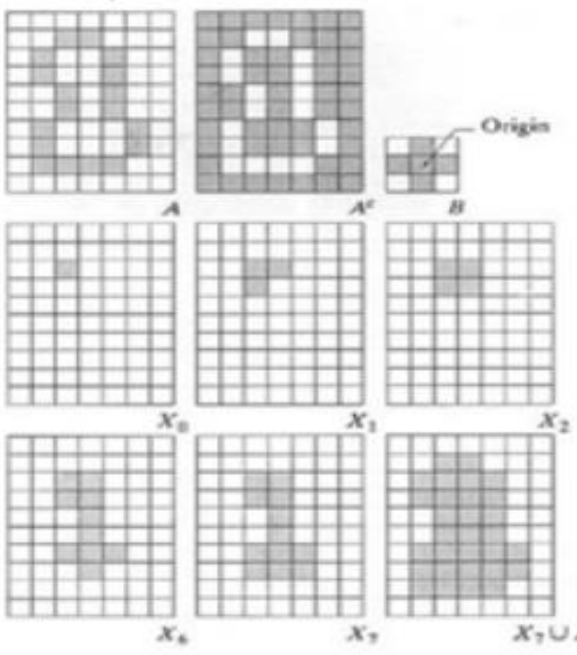
**B. Region Filling**

In this section, we develop an algorithm based on set dilation , complementation, and intersection for filling holes in an image. Let A denote a set whose elements are 8-connected boundaries, each boundary are enclosing a background region (i.e.hole). Given a point in each hole, the objective is to fill all the holes with 1's.

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \ldots$$

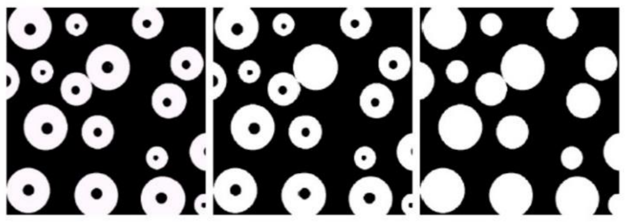
The region filling is also called hole filling.

Example:



**Ans:** 2

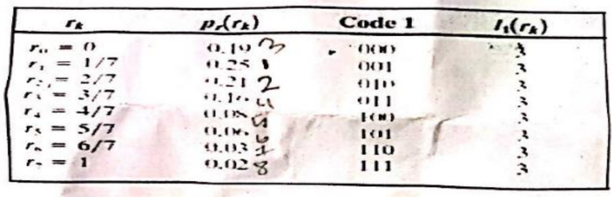
Example of Region filling



a b c

FIGURE 9.16 (a) Binary image (the white dot inside one of the regions is the starting point for the region-filling algorithm). (b) Result of filling that region (c) Result of filling all regions.

1.b) Generate Huffman code using source reduction method and calculate the data redundancy of the following image. (Where rk =gray levels, p(rk)=probability of the gray levels, Code 1= default code, I(rk)=length of code in bits) (au22, sp23, – 4 marks)

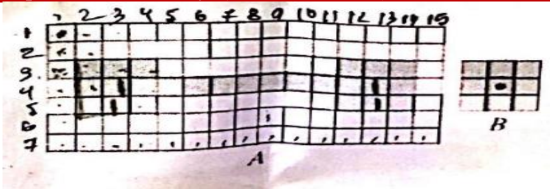


| $r_k$ | $p_r(r_k)$ | Code 1 | $l_1(r_k)$ |
|---|---|---|---|
| $r_0 = 0$ | 0.19 3 | 000 | 3 |
| $r_1 = 1/7$ | 0.25 1 | 001 | 3 |
| $r_2 = 2/7$ | 0.21 2 | 010 | 3 |
| $r_3 = 3/7$ | 0.16 4 | 011 | 3 |
| $r_4 = 4/7$ | 0.08 5 | 100 | 3 |
| $r_5 = 5/7$ | 0.06 6 | 101 | 3 |
| $r_6 = 6/7$ | 0.03 7 | 110 | 3 |
| $r_7 = 1$ | 0.02 8 | 111 | 3 |

1.b OR) Decode the following encoded code of a 8-bit image using LZW coding: 200,200, 80,80,256,258,260,259,257,80 (au22, – 4 marks)

**Ans:**

2.(a) Obtain the opening of the figure below using a 3 x 3 SE of 1's. Do all operations manually. (au22, – 4 marks)



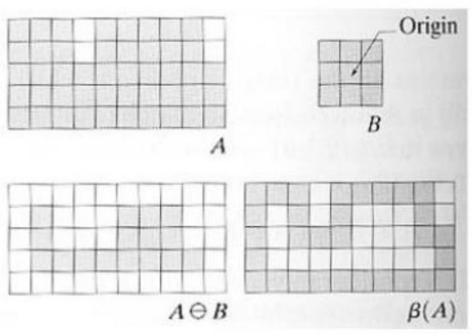
Ans:

**A. Boundary Extraction**

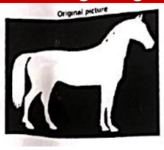
The boundary of a set A, denoted by β(A),can be obtained by first eroding A and B and then performing the set difference between A and its erosion.

$$\beta(A) = A - (A \ominus B)$$

And B is a structuring element.

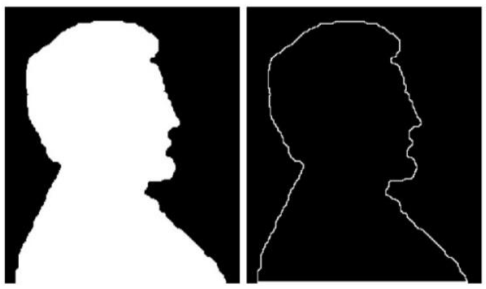


2: b) Find the steps of applying the convex hull operations on the following image. (au22, – 3 marks)



**Ans:** (same as)

**•Boundary Extraction Example**



a b
**FIGURE 9.14**
(a) A simple binary image, with 1's represented in white. (b) Result of using Eq. (9.5-1) with the structuring element in Fig. 9.13(b).

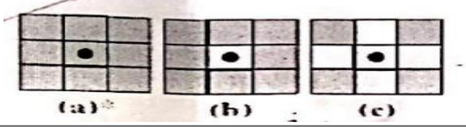
**How the morphological operation (A Θ X) Π (A' Θ (W-X)) might be applied:**

The image you sent appears to be a black and white silhouette of a horse. Here's how the operation might affect the image:**

1. **Erosion (A Θ X):**
   - If the structuring element A is small (e.g., a few pixels wide), erosion would slightly shrink the boundaries of the horse in the image X. This is because some of the edge pixels would not align perfectly with all the 1s (white pixels) in the structuring element.
   - If the structuring element A is larger than some of the narrower features of the horse (e.g., the legs), erosion could potentially remove those features entirely.
2. **Complement (A'):**
   - This would invert the black and white colors in the eroded image. So, the horse would become white on a black background.
3. **Dilation (W-X):**
   - As with erosion, the effect of dilation depends on the size of the structuring element A.
   - A small structuring element would slightly expand the boundaries of the white horse in the image.
   - A large structuring element could potentially fill in some gaps or holes within the horse, or even connect nearby objects in the image if they were close enough.
4. **Intersection (Π):**

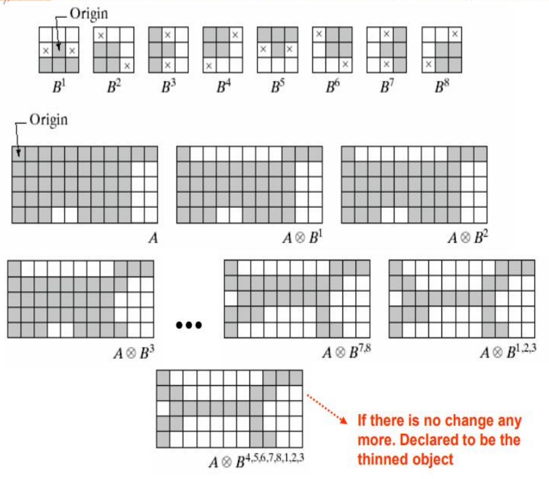
This would combine the eroded and dilated versions of the image. Only the areas that were both white in the eroded image and white in the dilated image would remain white in the final output.

OR 2: b) Sketch the result of eroding Fig. a with each of the following structuring elements (b and c). (au22, – 3 marks)

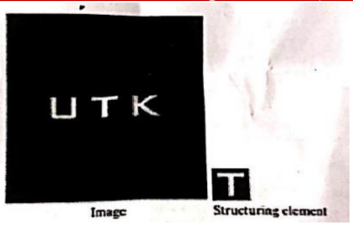


**Ans:**

• *Thinning: The following set of structuring elements are used for thinning operation.*



2: C) Sketch the result of applying the hit-or-miss transform to the image below, using the SE shown. Indicate clearly the origin and border you selected for the structuring element. (au22, sp23, – 3 marks)



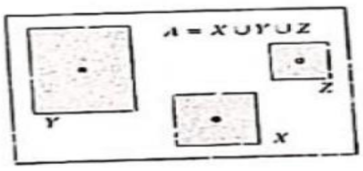
**Ans** (may be but PIC / sketh ta add kra lagbe--- Cheeg e solution ace): sketch the result of applying the hit-or-miss transform to the image:

1. **Identify the Hit and Miss Elements:**
   - The structuring element (SE) seems to have a white 'T' shape in the center and a black square on the bottom right. The white 'T' likely represents the "hit" element, and the black square represents the "miss" element.
2. **Origin and Border of the Structuring Element:**
   - The origin is usually chosen at a corner of the structuring element. In this case, a common choice would be the bottom left corner of the black square (marked with a red dot in the image). This would make the bottom right corner the border.
3. **Slide the Structuring Element:**
   - Slide the structuring element (SE) over the image.
4. **Match the Hit Element:**
   - At each position, check if the "hit" element (white 'T') of the SE matches the corresponding pixels in the image. The "hit" element matches if all the corresponding pixels in the SE are white and aligned with white pixels in the image.
5. **Check for the Miss Element:**
   - Simultaneously, check if the "miss" element (black square) of the SE overlaps any foreground pixels (white pixels) in the image.
6. **Mark Matching Locations:**
   - If both the "hit" and "miss" elements match their corresponding conditions at a particular position, mark that location in the output image.

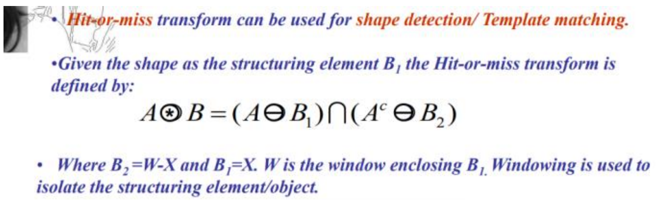
**Output Image:**

The hit-or-miss transform would likely identify some foreground pixels (white pixels) in the original image that form a 'T' shape. These matching 'T' shapes would be marked as white pixels in the output image. The background pixels (black pixels) would remain black.

2.b): Apply (A Θ X) Π (A'Θ (W-X)) morphological operator, or the following image. (sp22, – 4 marks)



**Ans:**

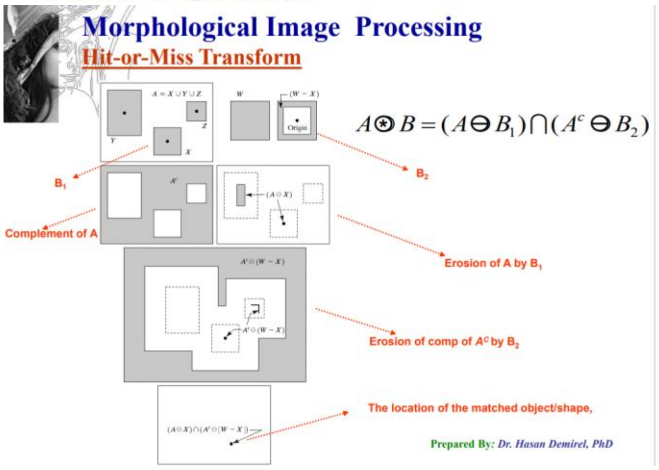


• *Hit-or-miss transform can be used for shape detection/ Template matching.*

•*Given the shape as the structuring element $B_1$ the Hit-or-miss transform is defined by:*

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

• *Where $B_2$=W-X and $B_1$=X. W is the window enclosing $B_1$. Windowing is used to isolate the structuring element/object.*

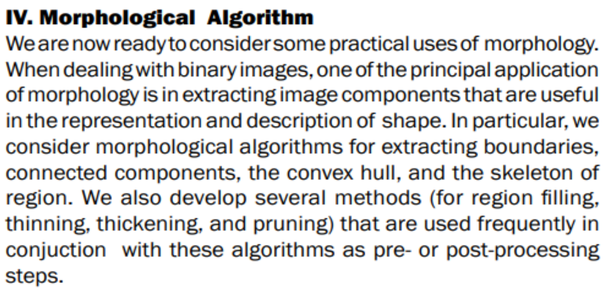


**Morphological Image Processing**

**Hit-or-Miss Transform**

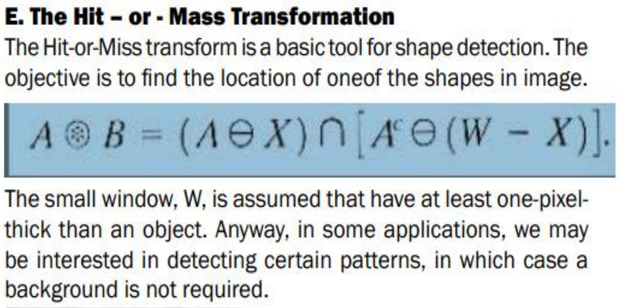
$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

A = X ∪ Y ∪ Z
W
(W − X)
Origin
B₁
B₂
Complement of A
Erosion of A by B₁
Erosion of comp of A^c by B₂
The location of the matched object/shape,
Prepared By: Dr. Hasan Demirel, PhD



**IV. Morphological Algorithm**

We are now ready to consider some practical uses of morphology. When dealing with binary images, one of the principal application of morphology is in extracting image components that are useful in the representation and description of shape. In particular, we consider morphological algorithms for extracting boundaries, connected components, the convex hull, and the skeleton of region. We also develop several methods (for region filling, thinning, thickening, and pruning) that are used frequently in conjuction with these algorithms as pre- or post-processing steps.

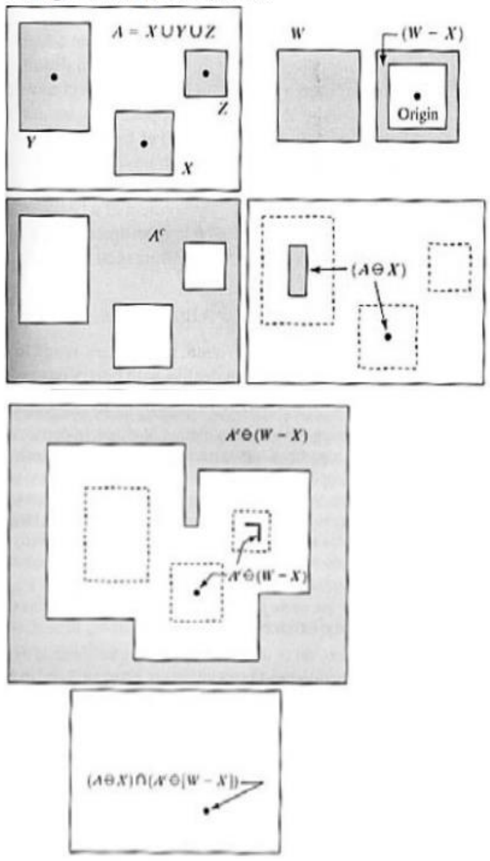


**E. The Hit – or - Mass Transformation**

The Hit-or-Miss transform is a basic tool for shape detection. The objective is to find the location of oneof the shapes in image.

$$A \circledast B = (A \ominus X) \cap [A^c \ominus (W - X)].$$

The small window, W, is assumed that have at least one-pixel-thick than an object. Anyway, in some applications, we may be interested in detecting certain patterns, in which case a background is not required.



OR. 1B) Encode and decode the following 4 x 4, 8-bit image using LZW coding: (sp23, – 4 marks)

| | | | |
|---|---|---|---|
| 208 | 208 | 129 | 129 |
| 208 | 208 | 129 | 129 |
| 208 | 208 | 129 | 129 |
| 208 | 208 | 129 | 129 |

**Ans:**

2.b) Basically, edges are created in an image due to the different types of discontinuity"- do you agree? Why? (sp23, 3 marks)

**Ans:** Yes, I agree that edges are created in an image due to different types of discontinuities. Here's why:

**Types of Discontinuities Leading to Edges:**

1. **Intensity Discontinuity:**
   - **Sharp Changes in Intensity:** When there is a significant change in the intensity values between adjacent pixels, an edge is formed. This is the most common type of edge and occurs at the boundaries of objects within the image.
   - **Gradient:** The gradient of the intensity function highlights regions with high-intensity changes, which correspond to edges.
2. **Color Discontinuity:**
   - **Color Changes:** When there is a sudden change in color between neighboring pixels, even if the intensity is similar, an edge is perceived. This is common in color images where different objects have distinct colors.
3. **Texture Discontinuity:**
   - **Changes in Texture Patterns:** An edge can also be formed where there is a change in texture patterns. This involves variations in pixel arrangements and their statistical properties rather than simple intensity or color changes.

**Why Discontinuities Create Edges:**

- **Human Perception:**
  - **Visual System:** The human visual system is highly sensitive to changes in light intensity and color, which it interprets as edges. This ability allows us to discern objects and their boundaries in our environment.
- **Mathematical Representation:**
  - **Edge Detection Algorithms:** Many edge detection algorithms, such as the Sobel, Canny, and Prewitt methods, rely on detecting discontinuities in intensity or color gradients to identify edges. These algorithms compute the rate of change of intensity or color to locate boundaries.
- **Segmentation and Object Recognition:**
  - **Segmentation:** Discontinuities help in segmenting an image into different regions or objects, which is crucial for image analysis and computer vision tasks.
  - **Object Recognition:** Identifying edges helps in recognizing and classifying objects within an image by providing shape and boundary information.

**Part: B Sp:23**

**Thanks to-** C201032, C201023, C201035, C201039, C201132

3(a). Analytics produce a decision boundary with the equation 5.1 $x_1$ - 0.3$x_2$ - 8.43. The first and second decision functions are produced as 6.6x1+1.2x2-28.33 and (5.6$x_1$+3.8x2-10.0) respectively. Determine the two-mean vector (m1, m2) from the above information.

**Ans:** To determine the two mean vectors m_1 and m_2 from the given information, we need to understand the relationship between the decision boundary and the decision functions. Here are the steps to solve the problem:

1. Decision Boundary: The decision boundary given is:

   $$5.1\,x_1 - 0.3\,x_2 - 8.43 = 0$$

   This equation separates the feature space into two regions, each corresponding to one of the classes.
2. Decision Functions: The decision functions are given as:

   $$g_1(x) = 6.6\,x_1 + 1.2\,x_2 - 28.33$$
   $$g_2(x) = 5.6\,x_1 + 3.8\,x_2 - 10.0$$

3. Finding the Mean Vectors: To find the mean vectors $m_1$ and $m_2$, we need to use the decision functions. The decision functions can be used to express the likelihood of a sample belonging to a particular class. When equating these decision functions to zero, we can find the means of the respective classes.
4. Equating the Decision Functions: At the decision boundary, the two decision functions should be equal:

   $$g_1(x) = g_2(x)$$

   Substituting the given decision functions, we get:

   $$6.6\,x_1 + 1.2\,x_2 - 28.33 = 5.6\,x_1 + 3.8\,x_2 - 10.0$$

5. Solving for the Boundary Equation: Simplifying the above equation:

   $$6.6\,x_1 + 1.2\,x_2 - 28.33 = 5.6\,x_1 + 3.8\,x_2 - 10.0$$
   $$\Rightarrow (6.6 - 5.6)x_1 + (1.2 - 3.8)x_2 = 28.33 - 10.0$$
   $$\Rightarrow 1.0\,x_1 - 2.6\,x_2 = 18.33$$
   $$\Rightarrow x_1 - 2.6\,x_2 = 18.33$$

   This is another form of the decision boundary equation, which can be compared with the given boundary (5.1 $x_1$ - 0.3 $x_2$ - 8.43 = 0).

6. Comparing the Decision Functions with the Mean Vectors: The decision functions are typically of the form:

   $$g_i(x) = w^T x + w_0$$

   Where w is the weight vector and $w_0$ is the bias term. For mean vectors, the decision functions can be interpreted as:

   $$g_1(x) = m_1^T x + w_0$$
   $$g_2(x) = m_2^T x + w_0$$

   Comparing:

   $$g_1(x) = 6.6\,x_1 + 1.2\,x_2 - 28.33$$
   $$g_2(x) = 5.6\,x_1 + 3.8\,x_2 - 10.0$$

   We can see that the coefficients of $x_1$ and $x_2$ represent the components of the mean vectors $m_1$ and $m_2$

7. Extracting Mean Vectors:

   From the coefficients of the decision functions, we can deduce:

   $m_1 = [6.6, 1.2]$ $m_2 = [5.6, 3.8]$

   Therefore, the two mean vectors $m_1$ and $m_2$ are:

   $$m_1 = \begin{pmatrix} 6.6 \\ 1.2 \end{pmatrix} \qquad m_2 = \begin{pmatrix} 5.6 \\ 3.8 \end{pmatrix}$$

3(b) Proof the equation *$P(R_1)$ = TRUE for region*. P(Ti) is a logical predicate defined over the points in set R, and Φ is the null set.

**Ans: Detailed Steps**

1. **Assume R1⊆R**:
   - Let RRR be the entire set of points.
   - Let R1⊆R be the region where the predicate PPP is TRUE.
2. **Define the Predicate PPP**:
   - The predicate P is a condition that can be evaluated at any point Ti in R.
   - For all Ti ∈ R1 , P(Ti)=TRUE
3. **Evaluate P over R1**:
   - Since P(Ti)=TRUE for all points Ti∈R1, the predicate P holds universally in R1.
4. **Conclusion**:
   - By definition, if a predicate holds for every point in a set, it holds for the entire set.
   - Therefore, P(R1)=TRUE.

3(c) Describe Canny edge detection algorithm.

**Ans:** The Canny edge detection algorithm is a popular and widely used method for detecting edges in digital
images. It was developed by John F. Canny in 1986 and is known for its ability to accurately detect edges while minimizing noise and producing thin, well-defined edges.

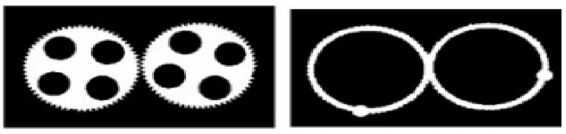
The Canny edge detection algorithm consists of several steps:

1. **Image smoothing:** The algorithm starts by applying a Gaussian blur to the input image. This step helps to reduce noise and remove small details that are not relevant to the edge detection process.
2. **Gradient calculation:** The next step involves calculating the gradients of the smoothed image. The image is convolved with two separate filters (Sobel operators) in the horizontal and vertical directions. These filters highlight regions of the image with strong intensity changes, which are likely to correspond to edges.
3. **Gradient magnitude and direction:** The gradient magnitudes and directions are computed based on the horizontal and vertical gradient components. The magnitude represents the strength of the gradient at each pixel, while the direction indicates the orientation of the gradient.
4. **Non-maximum suppression:** In this step, the algorithm aims to thin out the detected edges to obtain a single-pixel-wide edge. It involves scanning the gradient magnitude image and suppressing pixels that are not local maximums along the direction of the gradient. This ensures that only the most significant edges are preserved.
5. **Double thresholding:** A double thresholding technique is employed to classify the remaining edge pixels into strong, weak, or non-edges. This is done by defining high and low thresholds. If a pixel's gradient magnitude is above the high threshold, it is marked as a strong edge. If it is below the low threshold, it is considered a non-edge. Pixels with values between the low and high thresholds are labeled as weak edges.
6. **Edge tracking by hysteresis:** The final step involves connecting weak edges to strong edges to form continuous edges. This is achieved by analyzing the connectivity of weak edges. If a weak edge pixel is adjacent to a strong edge pixel, it is promoted to a strong edge. This process continues until no more weak edges can be connected.

The output of the Canny edge detection algorithm is a binary image where the detected edges are represented by white pixels and the non-edge regions are black.

The Canny edge detection algorithm is widely used in various computer vision and image processing applications, including object detection, image segmentation, and feature extraction.

4(a). Shown below A is an original binary image which has some defects. How would you detect the defects of the original image using morphological openers with different structural elements? Explain each structural element with proper justification. (sp23, 4 marks)



Binary image Detected defects

**Ans:** To detect defects in the binary image using morphological operators, follow these steps:

1. **Morphological Operators Overview**
   Morphological operations are used to process binary images and are based on the shape and structure of objects. The main morphological operations are dilation, erosion, opening, and closing.

2. **Structural Elements**
   Structural elements (SEs) are small binary images (usually small matrices) used to probe and interact with the binary image. Common SEs include:
   - Square: For general purposes.
   - Disk: For circular objects.
   - Line: For linear structures.

3. **Steps to Detect Defects**

**Step 1: Erosion**
- Purpose: To shrink the object, removing small imperfections.
- Example SE: A small disk.
- Explanation: Erosion removes pixels on object boundaries, which can help eliminate small defects and noise.

**Step 2: Dilation**
- Purpose: To expand the object, restoring its original shape after erosion.
- Example SE: A slightly larger disk than used in erosion.
- Explanation: Dilation adds pixels to object boundaries. When used after erosion, it helps in highlighting the defects.

**Step 3: Opening**
- Purpose: To remove small objects from the foreground.
  - Explanation: Opening is erosion followed by dilation. It smoothens the contour of an object and removes thin protrusions.

**Step 4: Closing**
- Purpose: To close small holes inside the object.
- Example SE: A disk.
- Explanation: Closing is dilation followed by erosion. It is used to fill small holes and gaps in the object.

4. **Detecting Defects**
   1. **Original Image (A):** Start with the binary image that contains objects with defects.
   2. **Closing Operation:**
      - Purpose: To fill small holes and connect broken parts within objects.
      - SE: Use a disk-shaped SE suitable for the size of the defects.
      - Result: A new image where defects are filled, and objects are connected.
   3. **Subtract Original from Closed Image:**
      - Purpose: To isolate defects.
      - Explanation: Subtracting the original image from the closed image will highlight the defects, as they will be the difference between the two images.

**Example Explanation**

- Binary Image: Shows gears with holes and potential defects.

Detected Defects: Shows the highlighted differences (defects) after applying the morphological operations.

4(b)(i) Evaluate the chain code of Figure 1 (use 8-direction).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | | | ■ | ■ | ■ | | |
| 2 | | ■ | | | | ■ | |
| 3 | ■ | | | | | | ■ |
| 4 | ■ | | | | | ■ | |
| 5 | | ■ | | | | ■ | |
| 6 | | ■ | ■ | ■ | ■ | ■ | |
| 7, 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

**Figure 1**

**Ans:**

4(b)(ii) Chain code has some limitations? Make the chain code in a rotational invariant

4(b) OR (i) Describe your method of computing the skeleton of the images. What kind of structural elements will you be using? What happens when you change the size of the structuring element?

**Ans: Method of Computing the Skeleton of Images**

To compute the skeleton of an image, the process typically involves morphological operations, specifically erosion and morphological thinning, using structuring elements. Here's a step-by-step outline of the method:

1. **Binarization**:
   - Convert the image to a binary image where the foreground (objects) is represented by 1s (or white) and the background is represented by 0s (or black).
2. **Morphological Thinning**:
   - Apply iterative morphological thinning operations to reduce the objects in the image to their skeletal form. Thinning repeatedly erodes the object pixels while preserving the connectivity.
3. **Erosion and Conditional Dilation**:
   - Erode the image using a structuring element (SE).
   - After each erosion, apply a conditional dilation to ensure that the connectivity of the image is preserved.
4. **Stopping Criterion**:
   - The process continues until no further changes occur in the image (i.e., the skeleton stabilizes).

**Structural Elements (SE) Used**

The choice of structuring element (SE) is crucial for the morphological operations. Common structuring elements for skeletonization include:

1. **Cross-shaped SE** *(Lower Left sided One)*:
   - A 3x3 cross-shaped SE is frequently used because it effectively captures the connectivity in both horizontal and vertical directions.

```
0 1 0          1 1 1
1 1 1          1 1 1
0 1 0          1 1 1
```

2. **Square SE** *(Upper Right sided One):*
   - A 3x3 square SE can also be used to ensure that all directions are considered in the erosion process.

**Effects of Changing the Size of the Structuring Element**

1. **Smaller SE (e.g., 2x2)**:
   - **Finer Details**: A smaller SE will preserve finer details of the objects in the image and produce a more detailed skeleton.
   - **Connectivity Issues**: There may be a risk of breaking the connectivity of the skeleton if the SE is too small.

2. **Larger SE (e.g., 5x5 or larger)**:
   - **Coarser Skeleton**: A larger SE will remove finer details and produce a coarser skeleton.
   - **Connectivity Preservation**: Larger SEs are better at preserving the overall connectivity of the object but may lose small structures.

4(b) OR (ii) Define Local Binary Pattern (LBP) with Example.

**Ans:** Local Binary Pattern (LBP) is a texture descriptor that captures the local patterns and variations within an image. It assigns a binary code to each pixel in an image based on the comparison of its intensity value with its neighboring pixels.

To define the Local Binary Pattern, let's consider a 3x3 neighborhood around a central pixel in a grayscale image.

We can assign a binary code to the central pixel based on the intensity comparison with its neighbors.

1. Central Pixel: The central pixel is denoted by P(x_c, y_c), where (x_c, y_c) represents its coordinates.
2. Thresholding: Compare the intensity value of the central pixel P(x_c, y_c) with its 8 neighboring pixels in a circular pattern. For simplicity, let's assume we move clockwise starting from the top-left neighbor. If the intensity of a neighbor pixel is greater than or equal to the intensity of the central pixel, assign it a value of 1. Otherwise, assign it a value of 0.
3. Binary Code: Concatenate the binary values of the 8 neighboring pixels in a clockwise manner to form an 8bit binary code. This binary code represents the local pattern around the central pixel.
4. Decimal Conversion: Convert the binary code to decimal. This decimal value represents the LBP value of the central pixel.

For example, let's consider a grayscale image with the intensity values of the 3x3 neighborhood around the central pixel P(x_c, y_c) as follows:

1 4 3
2 6 5
7 9 8

Let's assume the intensity of the central pixel P(x_c, y_c) is 6. We can compare the intensity values of the neighboring pixels with the central pixel and assign binary values:

1 0 0
0 1 1
1 1 1

The binary code formed by concatenating the binary values in a clockwise manner is "00111101". Converting this binary code to decimal, we get the LBP value for the central pixel P(x_c, y_c) as 61. This process is repeated for each pixel in the image, resulting in a new image where each pixel represents the LBP value of the corresponding pixel in the original image.

Local Binary Pattern is a powerful texture descriptor that captures the local structure and texture information in an image, making it useful for various computer vision tasks such as object recognition, face recognition, and texture analysis.

5(a) In the pattern recognition we have to extract the features from the object and that features are needed to recognized through recognizing algorithm for recognizing as specific object is it possible to recognize the object features through neural network that features are extracted from the object throw local binary pattern justify your answer with 4 x 4 8 bit color image.

**Ans:** Yes, it is possible to recognize object features through a neural network using features extracted from the object using the Local Binary Pattern (LBP) method. LBP is a widely used texture descriptor that captures
local patterns and variations in an image.

To justify the answer, let's consider a 4x4 8-bit color image as an example. The LBP algorithm can be applied to this image to extract local patterns as follows:

1. Image Conversion: Convert the 8-bit color image to grayscale. This can be done by taking the average of the RGB values for each pixel or by using other color-to-grayscale conversion methods.
2. LBP Calculation: Divide the grayscale image into a grid of cells, such as 4x4. For each pixel within a cell, compare its intensity value with the surrounding neighbor pixels (usually 8 neighbors in a circular pattern). Set a binary code for each neighbor pixel based on whether its intensity is greater or smaller than the central pixel. This results in an 8-bit binary pattern for each pixel within a cell.
3. Histogram Calculation: Count the occurrences of each unique binary pattern within each cell and create a histogram. The histogram represents the distribution of local patterns within the image.
4. Feature Vector: Concatenate the histograms from all the cells to form a feature vector. In the case of a 4x4 grid, we would have a feature vector of size (16 x Number of unique binary patterns).

Once the feature vector is extracted using the LBP method, it can be used as input to a neural network for object recognition. The neural network can be trained on a dataset of labeled objects, where the extracted LBP features are paired with their corresponding object classes. The network learns to recognize patterns and correlations between the extracted features and the object classes, enabling it to predict the class of new, unseen objects based on their LBP features.

By training a neural network with LBP features extracted from objects, it can learn to recognize object classes based on the extracted local patterns and variations. The network's ability to generalize and make predictions on unseen data depends on the quality and representativeness of the training dataset, the network architecture, and the training process.

5(b) “Template matching is popular way to identify patterns from images” i. Explain the formulation of correlation-based algorithm with example.

**Ans:** Template matching is a method used to find the best-matched location of a template image within a larger image. One common approach for template matching is correlation-based algorithms. Let’s dive into the details and illustrate with an example:

1. Correlation-Based Algorithms:
   - These algorithms aim to find the corresponding patch in a second image based on similarity with a template.
   - The key components are:
     - Appearance Similarity Function: We need a function to measure how similar the template and the image patches are.
     - Search Strategy: To find the location with the highest similarity, we use a search strategy.
   - The simplest (but least efficient) approach is exhaustive search, where we compare the template with all possible patches in the second image.
2. Correlation Function (Cross-Correlation):
   - If we perform exhaustive search over all image patches in the second image, this becomes the cross-correlation of the template with the image.
   - Mathematically, the cross-correlation score between the template (T) and an image patch (I) is given by:

$$C(T,I) = \sum_{x,y} T(x,y) \cdot I(x,y)$$

- Here, (T(x,y)) represents the intensity value of the template at position ((x,y)), and (I(x,y)) represents the intensity value of the image patch at the same position.

3. Example:
   - Consider a stereo pair of the "El Capitan" rock formation from the NASA Mars rover mission.
   - We have an image (Image 1) and a template (Template).
   - The cross-correlation score is computed as:

$$\text{Score}(x,y) = \sum_{u,v} [\text{Template}(u,v) \cdot \text{Image } 2(x+u, y+v)]$$

   - The highest score indicates the correct match, and we can locate the template within the image.
4. Problem with Raw Correlation:
   - Straight correlation with raw image templates has limitations.
   - For example, if the template is a constant grey value, the correlation score becomes sensitive to brightness changes.
   - Solution: Subtract the mean value of the template to ensure that the correlation score reflects only overlapping intensity patterns.

ii. Describe Pattern recognition model.

**Ans:** A pattern recognition model is a computational framework designed to identify patterns and regularities in data. It is widely used in fields such as image processing, speech recognition, bioinformatics, and machine learning. The goal of pattern recognition is to classify data (patterns) into different categories or to detect the presence of specific patterns within the data.

**Key Components of a Pattern Recognition Model**

1. **Data Acquisition**:
   - **Input Data**: The raw data collected from various sources such as sensors, cameras, microphones, etc.
   - **Preprocessing**: Involves cleaning and preparing the data for further analysis, which may include noise reduction, normalization, and transformation.
2. **Feature Extraction**:
   - **Features**: Characteristics or attributes of the data that are relevant for distinguishing between different patterns. Examples include edges in images, frequency components in audio signals, or statistical properties in text.
   - **Dimensionality Reduction**: Techniques like Principal Component Analysis (PCA) are used to reduce the number of features while retaining the essential information.
3. **Model Selection and Training**:
   - **Model Selection**: Choosing an appropriate model or algorithm for pattern recognition. Common models include decision trees, support vector machines (SVM), neural networks, and k-nearest neighbors (k-NN).
   - **Training**: Using a labeled dataset (training set) to train the model. The model learns the relationship between input features and output labels during this phase.
4. **Classification/Recognition**:
   - **Decision Making**: Applying the trained model to new, unseen data to classify it into one of the predefined categories or to recognize patterns.
   - **Output**: The result of the classification or recognition process, such as the identified category or detected pattern.
5. **Evaluation**:
   - **Performance Metrics**: Metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to evaluate the performance of the pattern recognition model.
   - **Validation**: Techniques like cross-validation and bootstrapping are used to assess the model's ability to generalize to new data.

5(b)OR. Describe minimum distance classified built the decision boundary for two pattern class w1 and w2 where to mean vectors are one 5.6 1.2 6.2 and 2 equal 1.40.3 3.7 explain how decision boundary works on classification stage

**Ans:** PC - SS

**Part: B Sp:23**

**Thanks to-** C201032, C201023, C201035, C201039, C201132

Q.3.a: "There are two types of segmentation algorithms- discontinuity and similarity"- Explain both of them with necessary examples.

Ans:

**Similarity approach:** This approach involves detecting similarity between image pixels to form a segment based on a given threshold. Machine learning algorithms like clustering are based on this type of approach to segment an image.

**Discontinuity approach:** This approach relies on the discontinuity of pixel intensity values of the image. Line, point and edge detection techniques use this type of approach for obtaining intermediate segmentation results that can later be processed to obtain the final segmented image.

**Discontinuity-based segmentation (Edge detection):**
This approach focuses on identifying sharp changes in intensity or color within an image, which often correspond to the boundaries between objects. Edges are essentially where different regions meet. Here's how it works:

**Example:** Imagine a grayscale image with a black cat on a white background. Discontinuity-based algorithms would detect the sharp transition from dark pixels (cat) to light pixels (background) along the cat's outline.

**Common discontinuity-based algorithms**:
Sobel filter
Prewitt filter
Canny edge detection

**Similarity-based segmentation:**
This approach groups pixels together based on their shared characteristics, such as color, intensity, texture, or a combination of these. The idea is that pixels within the same object tend to have similar properties.

**Example:** Consider a color image with a bowl of red apples. Similarity-based segmentation would group all the red pixels together, effectively segmenting the apples from the background (green table, for instance).

**Common similarity-based algorithms:**
Thresholding
K-means clustering
Region growing
Choosing the right approach:

The best segmentation method depends on the specific image and the desired outcome. Discontinuity-based methods work well when objects have distinct edges, while similarity-based methods are suitable for segmenting objects with more gradual transitions or when dealing with textured objects.
In many cases, a combination of both approaches might be used for optimal results.

**Q.3.a.OR**: -Edge linking similar points must be linked"- What are the similar points? Give necessary examples.

**Ans:** In the context of edge linking in image processing, "similar points" refer to points that belong to the same edge or contour in an image. These points typically have similar gradient magnitudes and directions, indicating that they are part of the same boundary or edge segment.

**What are Similar Points?**

Similar points in edge linking are usually defined by the following criteria:

1. **Gradient Magnitude**: Points with similar gradient magnitudes indicate that they are likely part of the same edge, as edges are characterized by significant changes in intensity.
2. **Gradient Direction**: Points with similar gradient directions suggest continuity in the edge. The direction of the gradient should be consistent along the edge.
3. **Proximity**: Points that are spatially close to each other are more likely to be part of the same edge.
4. **Intensity**: Points with similar intensity values can also be linked, especially in less complex edge detection scenarios.

Practical Example: Canny Edge Detection

The Canny edge detection algorithm provides a good practical example of edge linking using similar points:

1. **Gradient Calculation**:
   - Compute the gradient magnitude and direction for each pixel.
2. **Non-Maximum Suppression**:
   - Suppress non-maximum points in the gradient image to thin the edges.
3. **Double Thresholding**:
   - Apply two thresholds to identify strong and weak edges.
4. **Edge Linking (Hysteresis)**:
   - Link weak edges to strong edges if they are connected, ensuring that edges are continuous.

Q.3.b: Describe the segmentation based on motion

**Ans:** Motion segmentation is a computer vision technique that focuses on separating moving objects from the background or from each other in a video sequence. It analyzes how pixels move over time to group them into different segments based on their motion patterns.

Here's a breakdown of how it works:

1. Motion Estimation: The first step involves calculating the motion of each pixel between consecutive frames of the video. This can be done using techniques like optical flow, which estimates the apparent motion of points in the image.
2. Grouping by Motion: Pixels with similar motion patterns are then grouped together. This can be achieved through various algorithms like:
   - Clustering: Similar to image segmentation, clustering algorithms group pixels based on their motion vectors (representing direction and speed) into different clusters, each corresponding to a moving object.
   - Tracking: Here, individual points or features are tracked through the video frames, and points with similar trajectories are considered to belong to the same moving object.

**Applications of Motion Segmentation:**

Motion segmentation plays a crucial role in various computer vision tasks, including:

- Object tracking: Segmenting objects allows for tracking their movement across video frames. This is useful in applications like surveillance systems or self-driving cars.
- Action recognition: Understanding object motion patterns is essential for recognizing actions in videos, such as someone walking, running, or waving.
- Video analysis: Segmenting moving objects from the background helps analyze scene activity and content, useful for video summarization or anomaly detection.

Challenges of Motion Segmentation:

- Background motion: If the camera itself is moving, differentiating true object motion from the background motion can be challenging.
- Occlusions: When objects overlap or hide each other partially (occlusion), it can lead to segmentation errors.
- Similar motion patterns: Objects moving in the same direction or with similar speeds can be difficult to distinguish.

Overall, motion segmentation is a powerful tool for extracting valuable information about moving objects in videos. While it has its limitations, ongoing research continues to improve its accuracy and robustness.

Q.3.c: Find the optimal threshold of the following image using Otsu method,

**Ans:** https://www.youtube.com/watch?v=jUUkMaNuHP8

Q.4.a: describe your method of computing the skeleton of the images. What kind of structural elements will you be using? What happens when you change the size of the structuring element?

**Ans:**

**Common Skeletonization Techniques:**

**Morphological Thinning:** This iterative process iteratively removes pixels from the object's boundaries while preserving its overall shape and connectivity. It uses small, predefined shapes called structuring elements to "probe" the object's edges.

**Distance Transform-based methods**: These methods calculate a distance map where each pixel's value represents its distance to the nearest object boundary. The skeleton is then extracted based on ridges or specific features in this distance map.

**Structural Elements and their Impact:**

**Size:** The size of the structuring element significantly impacts the resulting skeleton

**Smaller structuring element**: This removes more pixels during thinning, resulting in a thinner and more detailed skeleton. However, it might be more susceptible to noise and break the object's connectivity if too small.

**Larger structuring element**: This removes fewer pixels, leading to a thicker and smoother skeleton. But it might miss finer details and potentially remove thin branches of the object.

**Choosing the Right Size:**

The ideal size depends on the specific image and desired outcome. A good balance is sought between capturing details and preserving connectivity. In some cases, using multiple structuring elements with varying sizes in a sequential manner might be necessary.

Q:4.b: ) What are the limitations of manual features engineering?

**Ans:** Manual feature engineering, while powerful, has several limitations that can hinder the effectiveness of machine learning models:

Time-consuming and resource-intensive: Crafting informative features by hand is a tedious process that requires significant domain knowledge and experimentation. This can be especially challenging for large datasets with many potential features.

Subjectivity and bias: The selection of features is inherently subjective and depends on the data scientist's understanding of the problem. Unconscious bias can also creep in, leading to features that reinforce existing biases or overlook important relationships.

Limited feature discovery: Humans might miss complex or non-intuitive feature combinations that could be crucial for the model's performance. This can restrict the model's ability to learn intricate patterns in the data.

Data-specific and limited reusability: Features engineered for one dataset or task might not generalize well to others. This means significant effort needs to be repeated for every new problem, hindering efficiency.

Maintenance challenges: As data evolves or the problem definition changes, manually engineered features might need to be updated or redesigned. This adds to the ongoing maintenance burden.

Potential for overfitting: Over-engineering features to fit a specific dataset can lead to overfitting, where the model performs well on the training data but fails to generalize to unseen data.

## Q.4.c: Consider the following image and filter. Now extract the features using convolution. Use stride 1. Then reduce the dimensionality of the features vector using max polling.

**Ans:** Stride: https://www.youtube.com/watch?v=O2CBKXr_Tuc
Max Pooling : https://www.youtube.com/watch?v=8oOgPUO-TBY&t=1s

## Q.4.d: What does rotational invariant means in chain code? How can we make the chain code rotational invariant?

**Ans:** Probable solution: https://www.youtube.com/watch?v=t1OCX1j_zJE

In chain codes, rotational invariance refers to the property where the numerical representation of a closed object's boundary remains consistent regardless of its orientation.

Here's how a chain code can be rotationally variant:

Imagine a square. Traversing its boundary in a clockwise direction might generate a chain code like "0, 1, 2, 3, 0." However, if the square is rotated slightly, the same traversal would result in a different code, like "1, 2, 3, 0, 1." This difference arises solely due to the rotation, not the actual shape.

Techniques for Rotational Invariance:
There are two main approaches to achieve rotational invariance in chain codes:

**Normalization:** This method involves processing the chain code to find the starting point that yields the lexicographically smallest code (considering the code as a sequence of digits). Here's the process:

- Try starting the traversal at each point on the boundary.
- Generate the chain code for each starting point.
- Convert each code into a number by interpreting the sequence of directions as digits in a base-4 system (since there are typically four directions: up, right, down, left).
- Identify the starting point that generates the numerically smallest code.

The chain code starting from this point is considered the normalized or rotationally invariant version.

**Differential Chain Code:** This approach focuses on the relative changes in direction between consecutive codes in the sequence. Here's the idea:

- Calculate the difference between each pair of adjacent codes in the original chain code. The difference represents the number of positions you need to move counterclockwise to reach the second direction code from the first.
- This differential chain code is rotationally invariant because the relative direction changes remain the same regardless of the starting point.

Both normalization and differential chain codes address the issue of rotational variance in chain codes, allowing for shape recognition or comparison independent of object orientation.

## Q.5.a: Perceptron is nonlinear classifier- do you agree or not. Justify your answer

**Ans:** Perceptrons are linear classifiers because they use a linear function (w^T * x + b) to create a decision boundary. This boundary is a straight line (2D) or hyperplane (higher dimensions) that separates data points belonging to different classes. The weights (w) and bias (b) in the linear function determine the position and orientation of this decision boundary.

## 5.a.OR: If the original image is 256x256, 8 bits/pixel, it would occupy 65,536 bytes. After compression it occupies 6554/bytes. Calculate the compression ratio.

**Original Image Size:** 256 pixels (width) x 256 pixels (height) * 8 bits/pixel (color depth) = 65,536 bytes
**Compressed Image Size:** 6554 bytes
**Compression Ratio:** Original size / Compressed size = 65,536 bytes / 6554 bytes ≈ 10

Therefore, the compression ratio is approximately 10:1. This means that the compressed image file is roughly 10 times smaller than the original image file.

## Q5.b: Design a CNN architecture for the image classification? You can choose any real 4 world image classification problem as a case.

**Ans:** We design a Convolutional Neural Network (CNN) architecture for the real-world image classification problem of classifying dog breeds. The dataset we'll refer to for this task is the Stanford Dogs Dataset, which contains images of 120 different dog breeds. The **Architecture Details-**

1. **Input Layer**:
   - Input: (224, 224, 3) RGB images.
2. **First Convolutional Block**:
   - Conv2D layer with 64 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 64 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - MaxPooling2D layer with pool size (2, 2).
3. **Second Convolutional Block**:
   - Conv2D layer with 128 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 128 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - MaxPooling2D layer with pool size (2, 2).
4. **Third Convolutional Block**:
   - Conv2D layer with 256 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 256 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 256 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - MaxPooling2D layer with pool size (2, 2).
5. **Fourth Convolutional Block**:
   - Conv2D layer with 512 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 512 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 512 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - MaxPooling2D layer with pool size (2, 2).

6. **Fifth Convolutional Block**:
   - Conv2D layer with 512 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 512 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - Conv2D layer with 512 filters, kernel size (3, 3), activation 'ReLU', padding 'same'.
   - MaxPooling2D layer with pool size (2, 2).
7. **Flatten Layer**:
   - Flatten the output from the previous layer to create a feature vector.
8. **Fully Connected Layers**:
   - Dense layer with 4096 units, activation 'ReLU'.
   - Dropout layer with dropout rate 0.5.
   - Dense layer with 4096 units, activation 'ReLU'.
   - Dropout layer with dropout rate 0.5.
9. **Output Layer**:
   - Dense layer with 120 units (one for each dog breed), activation 'softmax'.

Q5.b. OR: In the pattern recognition and classification there are four basic steps involved, which are sensing an image, segmenting the image, extracting the features from the segmented objects, and classification to recognize the specific object. Based on this procedure explain how you can segment and classify a specific object from an 8-bit color image. Specify how you can apply the local binary pattern and artificial neural network algorithm in this application.

**Ans:** Segmenting and Classifying an Object Using LBP and Neural Networks in 8-bit Color Images

Here's how you can segment and classify a specific object from an 8-bit color image using Local Binary Patterns (LBP) and an Artificial Neural Network (ANN):

1. Preprocessing:

   Load the Image: Read the 8-bit color image into memory.

   Color Conversion (Optional): Depending on the application, you might convert the color image to grayscale for simplicity. This reduces processing time but discards potential color-based features.

2. Segmentation using Local Binary Patterns (LBP):

   Local Neighborhood Analysis: For each pixel in the image, define a small neighborhood (e.g., 3x3 pixels).

   Thresholding: Compare the intensity of the center pixel with its neighbors. If a neighbor's intensity is greater than or equal to the center, assign a 1; otherwise, assign a 0.

   Binary String Formation: Convert the resulting binary values in the neighborhood into a binary string (e.g., "10110001"). This string represents the local intensity pattern around the center pixel.

   Repeat: Perform this process for every pixel in the image, generating an LBP image of the same size.

3. Feature Extraction:

   Histogram Construction: Create a histogram for the entire LBP image. This histogram represents the frequency of each unique LBP pattern occurring in the image.

   Additional Features (Optional): Depending on the object's characteristics, you might extract additional features beyond the LBP histogram. These could include:

   Object size and shape descriptors (e.g., area, perimeter, aspect ratio)

   Color features (if the image remains in color) like average color intensity or color moments

   Edge features detected using edge detection algorithms

4. Classification using Artificial Neural Network (ANN):

   **Training Data Preparation:** Prepare a training dataset consisting of images containing the specific object you want to classify, along with corresponding labels indicating the object's presence or absence (binary classification) or its specific class (multi-class classification). Each image in the dataset should be pre-processed and have features extracted (LBP histogram and potentially others) similar to the test image.

   **ANN Architecture:** Design a multi-layer ANN with an input layer size matching the number of features extracted (e.g., size of the LBP histogram + size of additional features), hidden layers with a suitable number of neurons for learning complex patterns, and an output layer with one neuron (binary classification) or multiple neurons (multi-class classification) depending on the problem.

   **Training:** Train the ANN using the prepared training data. The network learns to associate the extracted features with the corresponding object labels.

   **Classification:** Once trained, present the features extracted from the test image (LBP histogram and potentially others) to the trained ANN. The network outputs a value (binary for presence/absence or a probability distribution for multi-class) indicating the predicted class of the object in the test image.

**Benefits of using LBP and ANN:**

**LBP**: Effective for capturing local texture patterns, which can be helpful for distinguishing objects from the background. It's also computationally efficient for 8-bit images.

**ANN**: Flexible and powerful for learning complex relationships between features and object classes. Can handle both binary and multi-class classification problems.

**Limitations:**

**LBP**: Might not be suitable for objects with smooth textures or complex shapes.

**ANN**: Requires a well-designed architecture and sufficient training data for optimal performance.

**Additional Considerations:**

Selecting an appropriate neighborhood size in LBP can impact the captured texture details.

The choice of ANN architecture and hyperparameters (learning rate, number of neurons) requires experimentation for achieving optimal classification accuracy.

By combining LBP for feature extraction and an ANN for classification, you can develop a robust system for segmenting and classifying specific objects in 8-bit color images.