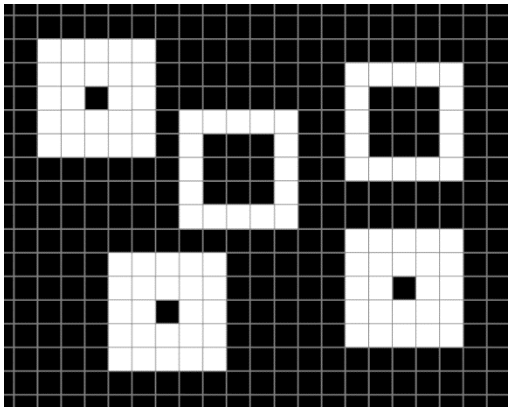
Video Signals
January, 19th 2026
Duration: 2 hours

Instructions. Answer all questions. Justify each step clearly. Calculators and handwritten notes are not allowed.



Exercise 1 (12 pt)

In a binary image (objects = 1, background = 0) there are many square frames (rings) having the same outer side length, but with an inner square hole of two possible sizes:

- Type A: *small* inner hole
- Type B: *large* inner hole

Frames appear at different locations, do not overlap, and no other shapes are present.
You must design a pipeline based on the Hit-or-Miss operator to:

1. detect (localize) all frames in the image;
2. robustly discriminate Type A from Type B;
3. output two binary masks: one containing only Type A frames and one containing only Type B frames.

Required: Describe an implementable solution, detailing:

- any pre-processing steps (if needed) and their motivation;
- the design of the *pair* of structuring elements (foreground/background) for Hit-or-Miss;
- how you deal with the presence of a *hole* (e.g., complements, filling, additional constraints);
- how Hit-or-Miss responses are converted into a final classification rule.

(*No code is required, but the description must be complete and unambiguous*.)

Exercise 2 (12 pt)

Consider the following grayscale image patch:

$$I = \begin{bmatrix} 5 & 5 & 4 & 2 & 4 \\ 5 & 3 & 0 & 0 & 2 \\ 4 & 3 & 3 & 4 & 4 \\ 2 & 1 & 4 & 4 & 4 \\ 1 & 1 & 4 & 5 & 5 \end{bmatrix}$$

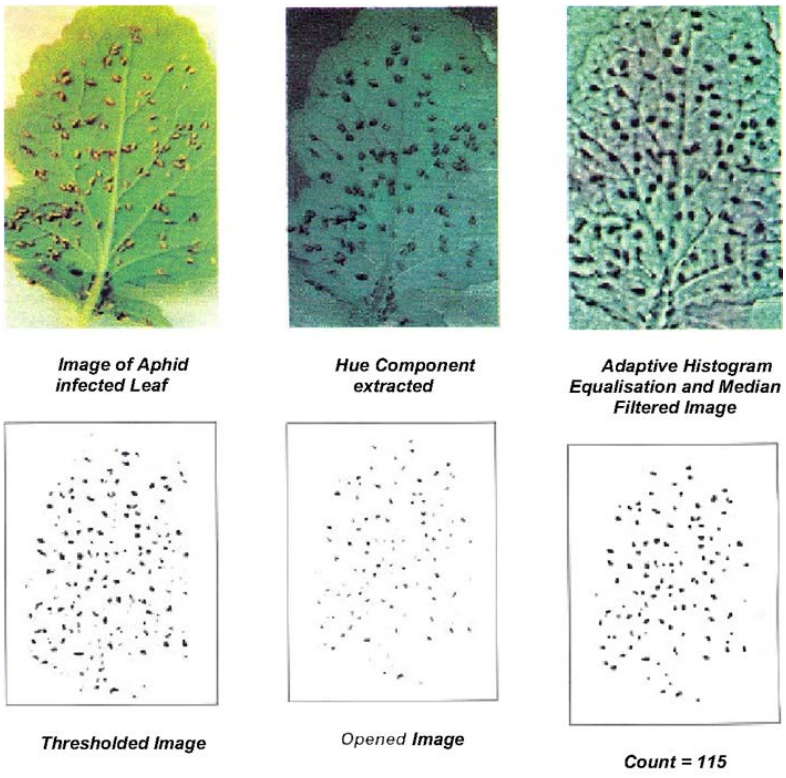
1. Define a 3×3 Laplacian filter (write the kernel explicitly) and compute the output matrix L obtained by convolving I with this kernel.
2. For boundary handling, assume values outside the matrix are obtained using mirror reflection.
3. Propose a rule to extract edges from L. Clearly state the chosen rule and justify it.
4. Explain how you would combine the Laplacian-based edges with a first-order operator (e.g., Prewitt or Sobel) to reduce false edges. Specify:
    - which first-order map is used,
    - how threshold(s) are selected,
    - the final logical decision rule (AND/OR, hysteresis, etc.).

Hint: clearly identify the steps that depend on the boundary condition.

[Continues on the back]

Exercise 3 (12 pt) Matlab Code

Visual inspection of plant leaves is an important task in agricultural monitoring, as the presence of insects such as aphids can significantly affect crop health and yield. Manual inspection is time-consuming and subjective, motivating the use of digital image processing techniques to automatically detect and quantify insect infestation. A color image (test.png) of a leaf affected by aphids is analyzed using color space transformation, contrast enhancement, filtering, thresholding, and morphological operations to detect and count aphids present on the leaf surface.



Write a MATLAB script able to perform the following steps:

a) Load the RGB image of the leaf and display it.
b) Convert the RGB image to the HSV color space and extract the Hue (H), Saturation (S), and Value (V) components.
c) Apply adaptive histogram equalization to the Hue component in order to enhance contrast between the aphids and the leaf background.
d) Reduce noise in the enhanced Hue image using a 3x3 median filter.
e) Perform binary segmentation to isolate aphids by applying logical thresholding based on:
    i. low intensity (Value) (lower than 0.45),
    ii. low saturation (lower than 0.95),
    iii. exclusion of typical leaf hues (green–yellow range) (outside 0.2-0.4 range).
f) Using a 3x3 square structuring element, apply a morphological operation of your choice to remove small noise regions and improve object separation.
g) Label the connected components in the binary image and determine the total number of aphids present.

List of possible Matlab functions

```
figure
im2double
imread
rgb2gray
imcrop
imfilter
imhist
imopen
rgb2hvs
histeq
hist
imshow
fspecial
imerase
strel
hsv2rgb
bwlabel
```

Solutions
Ex 1

Detect all square frames (rings) and classify them into:

- Type A: small inner hole
- Type B: large inner hole

using Hit-or-Miss (HMT), then output two binary masks.

Let X be the binary image (objects=1). Let X^c be its complement. Hit-or-Miss at location p is:

$$\mathrm{HMT}(X; B_1, B_2) = (X \ominus B_1) \cap (X^c \ominus B_2),$$

where B_1 constrains the foreground (must be 1) and B_2 constrains the background (must be 0).

To avoid detecting generic background areas (not holes), add a constraint that *around the hole there must be object pixels*. Use a *ring constraint* around the hole:

- Foreground SE B_1 placed on the ring (must be 1 in X).
- Background SE B_2 placed inside the hole (must be 0 in X, i.e. 1 in X^c).

Concretely, for each type:

Type A detector: $\mathrm{HMT}(X; B_1^{(A)}, B_2^{(A)})$ Type B detector: $\mathrm{HMT}(X; B_1^{(B)}, B_2^{(B)})$

Where:

$$B_1^{(A)} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \quad B_2^{(A)} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

$$B_1^{(B)} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \quad B_2^{(A)} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

Where the origo is always in the center of the SE.

Ex 2

Given matrix

$$I = \begin{bmatrix} 5 & 5 & 4 & 2 & 4 \\ 5 & 3 & 0 & 0 & 2 \\ 4 & 3 & 3 & 4 & 4 \\ 2 & 1 & 4 & 4 & 4 \\ 1 & 1 & 4 & 5 & 5 \end{bmatrix}$$

1) Laplacian kernel

Choose the 4-neighbour Laplacian:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad L(i,j) = I(i-1,j) + I(i+1,j) + I(i,j-1) + I(i,j+1) - 4I(i,j).$$

2) Boundary condition: mirror reflection

We use mirror reflection (symmetric at the borders). For example:

$$I(0,j) = I(1,j), \quad I(n+1,j) = I(n,j),$$

and similarly for columns (and extended by reflection for farther indices if needed).

3) Convolution result

The Laplacian output is:

$$L = \begin{bmatrix} 0 & -3 & -5 & 2 & -4 \\ -3 & 1 & 10 & 8 & 2 \\ -2 & -1 & -1 & -5 & -2 \\ 0 & 6 & -4 & 1 & 1 \\ 1 & 3 & -2 & -2 & -1 \end{bmatrix}$$

4) Edge extraction rule from L

A standard Laplacian rule is **zero-crossings**: a pixel (or a location between pixels) is an edge if L changes sign across a neighbourhood.

A robust discrete rule:

- consider an 8-neighbourhood $\mathcal{N}(i,j)$;
- declare an edge at (i,j) if $\exists (u,v) \in \mathcal{N}(i,j)$ such that

  $$L(i,j) \cdot L(u,v) < 0$$

  and additionally

  $$|L(i,j) - L(u,v)| \geq T_L$$

  to suppress weak sign flips (noise/quantization).

A reasonable choice here is $T_L \in [2,4]$ since the strongest magnitudes are around 10.

5) Combine with a first-order operator to reduce false edges

Use Sobel (or Prewitt) and compute gradient magnitude:

$$G = \sqrt{G_x^2 + G_y^2}.$$

Then keep only zero-crossings that are also supported by a sufficiently strong gradient:

$$\text{Edge} = \text{ZeroCross}(L, T_L) \ \wedge \ (G \geq T_G).$$

Threshold T_G can be set:

- relative to the maximum: $T_G = \alpha \max(G)$ with $\alpha \in [0.15, 0.3]$,
- or by histogram-based selection (e.g., upper percentile).

Optionally, apply hysteresis on G (Canny-like): strong edges if $G \geq T_H$, weak edges if $T_L' \leq G < T_H$ kept only if connected to strong ones, always gated by the Laplacian zero-crossing.

Ex 3

```
close all;
clear all
clc;

imgPath = 'test.png';


% a) Load
Irgb = im2double(imread(imgPath));
figure
imshow(Irgb)
title('input image')

% b) RGB -> HSI
Ihsv = rgb2hsv(Irgb);
H = Ihsv(:,:,1);
S = Ihsv(:,:,2);
V = Ihsv(:,:,3);


% c) Adaptive histogram equalization
H_eq = adapthisteq(H);

figure
imshow(H_eq);
title('H_eq');

% d) median filter
H_f  = medfilt2(H_eq, [3 3]);
figure
imshow(H_f);
title('H_f');

% e) Threshold
BW =  (V < 0.45) & (S < 0.95) & ~((H_f >= 0.2) & (H_f <= 0.4));
```

```
% f) Morphological opening
se = strel('square', 3);
BW_open = imopen(BW, se);


%g) Count
CC = bwlabel(BW_open, 8);
count = max(CC(:));
```