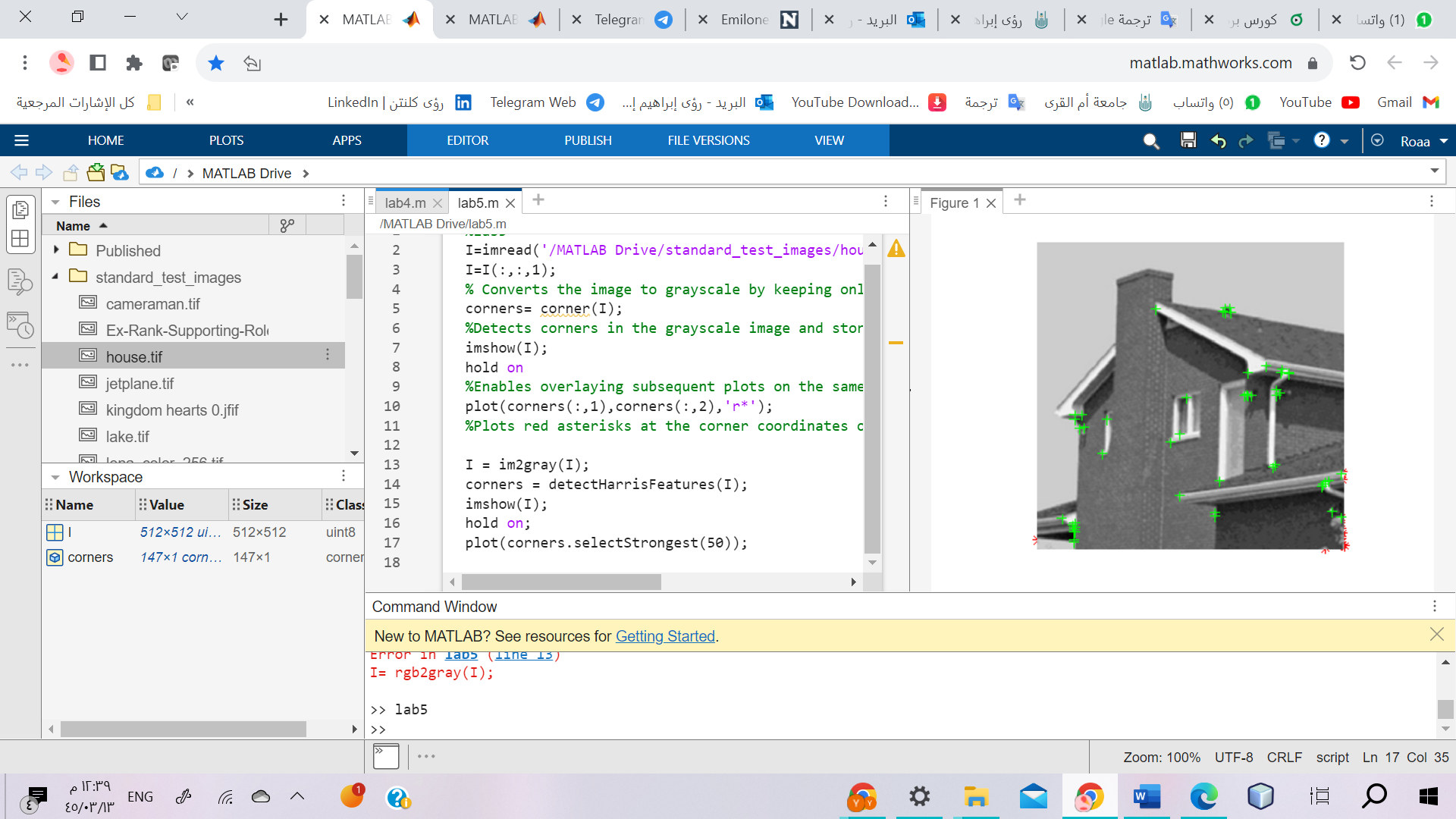


1. %lab5: This line is a comment and indicates that this code is for lab 5. It doesn't affect the code execution.
2. I = imread('/MATLAB Drive/standard\_test\_images/house.tif');: Reads the image file "house.tif" from the specified location in the MATLAB Drive and stores it in the variable I.
3. I = I(:,:,1);: Extracts the first channel of the image I (assuming it's a color image) and replaces I with the grayscale version. This is done to ensure that the corner detection is performed on a grayscale image.
4. corners = corner(I);: Detects corners in the image I using the corner function and stores the corner coordinates in the variable corners.
5. imshow(I);: Displays the grayscale image I.
6. hold on: This command ensures that the subsequent plotting operations are added to the existing figure instead of creating a new one.
7. plot(corners(:,1), corners(:,2), 'r\*');: Plots red asterisks (r\*) at the locations of the corners. The corners(:,1) and corners(:,2) represent the x and y coordinates of the corners, respectively.

Overall, this code reads an image, performs corner detection on the grayscale version of the image, and then displays the image with red asterisks indicating the detected corners.



1. I = rgb2gray(I);: Converts the RGB image I to grayscale using the rgb2gray function. This ensures that the image is in a single channel format, suitable for corner detection algorithms.
2. corners = detectHarrisFeatures(I);: Detects corners in the grayscale image I using the Harris corner detection algorithm. The detectHarrisFeatures function analyzes the image to identify corner features and returns a set of corner points in the variable corners.
3. imshow(I);: Displays the grayscale image I.
4. hold on;: Enables overlaying subsequent plots on the same figure.
5. plot(corners.selectStrongest(50));: Plots the 50 strongest corners on the image. The selectStrongest method is used to select the top N strongest corners from the detected corners, and the plot function is used to mark these corners on the image.

The expected output of this code is an image displayed in grayscale, with red markers indicating the strongest 50 corners detected by the Harris corner detection algorithm.

%lab5

I=imread('/MATLAB Drive/standard\_test\_images/house.tif');

I=I(:,:,1);

% Converts the image to grayscale by keeping only the first channel.

corners= corner(I);

%Detects corners in the grayscale image and stores their coordinates in the variable corners.

imshow(I);

hold on

%Enables overlaying subsequent plots on the same figure.

plot(corners(:,1),corners(:,2),'r\*');

%Plots red asterisks at the corner coordinates on the image.

I = im2gray(I);

corners = detectHarrisFeatures(I);

%Detects corners in the grayscale image I

% using the Harris corner detection algorithm

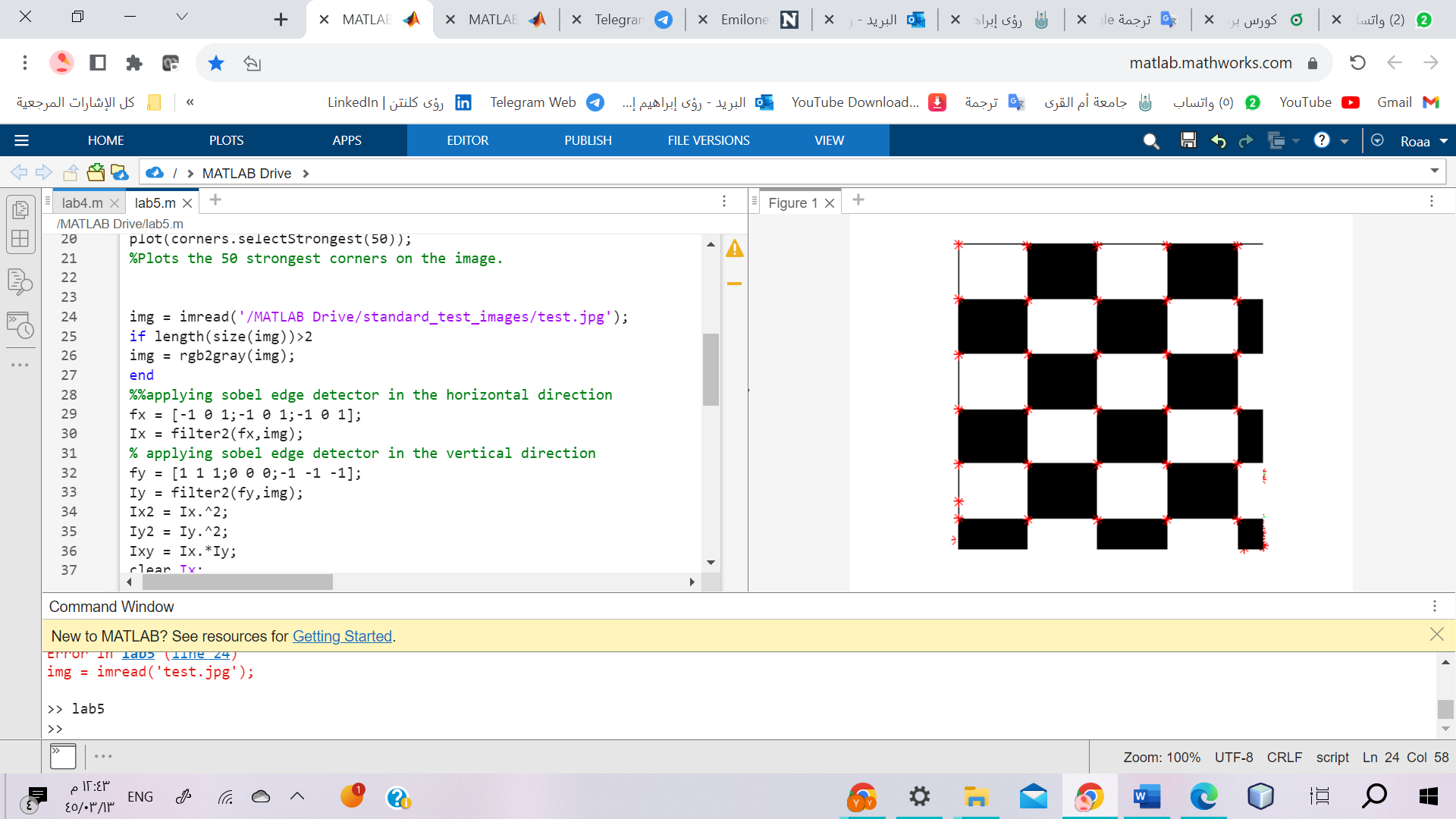
% and stores the corner information in corners.

imshow(I);

hold on;

plot(corners.selectStrongest(50));

%Plots the 50 strongest corners on the image.



img = imread('/MATLAB Drive/standard\_test\_images/test.jpg');

if length(size(img))>2

img = rgb2gray(img);

end

%%applying sobel edge detector in the horizontal direction

fx = [-1 0 1;-1 0 1;-1 0 1];

Ix = filter2(fx,img);

% applying sobel edge detector in the vertical direction

fy = [1 1 1;0 0 0;-1 -1 -1];

Iy = filter2(fy,img);

Ix2 = Ix.^2;

Iy2 = Iy.^2;

Ixy = Ix.\*Iy;

clear Ix;

clear Iy;

%applying gaussian filter on the computed value

h= fspecial('gaussian',[7 7],2);

Ix2 = filter2(h,Ix2);

Iy2 = filter2(h,Iy2);

Ixy = filter2(h,Ixy);

height = size(img,1);

width = size(img,2);

result = zeros(height,width);

R = zeros(height,width);

Rmax = 0;

for i = 1:height

for j = 1:width

M = [Ix2(i,j) Ixy(i,j);Ixy(i,j) Iy2(i,j)];

R(i,j) = det(M)-0.01\*(trace(M))^2;

if R(i,j) > Rmax

Rmax = R(i,j);

end

end

end

cnt = 0;

for i = 2:height-1

for j = 2:width-1

if R(i,j) > 0.1\*Rmax && R(i,j) > R(i-1,j-1) && R(i,j) > R(i-1,j) && R(i,j) > R(i-1,j+1) && R(i,j) > R(i,j-1) && R(i,j) > R(i,j+1) && R(i,j) > R(i+1,j-1) && R(i,j) > R(i+1,j) && R(i,j) > R(i+1,j+1)

result(i,j) = 1;

cnt = cnt+1;

end

end

end

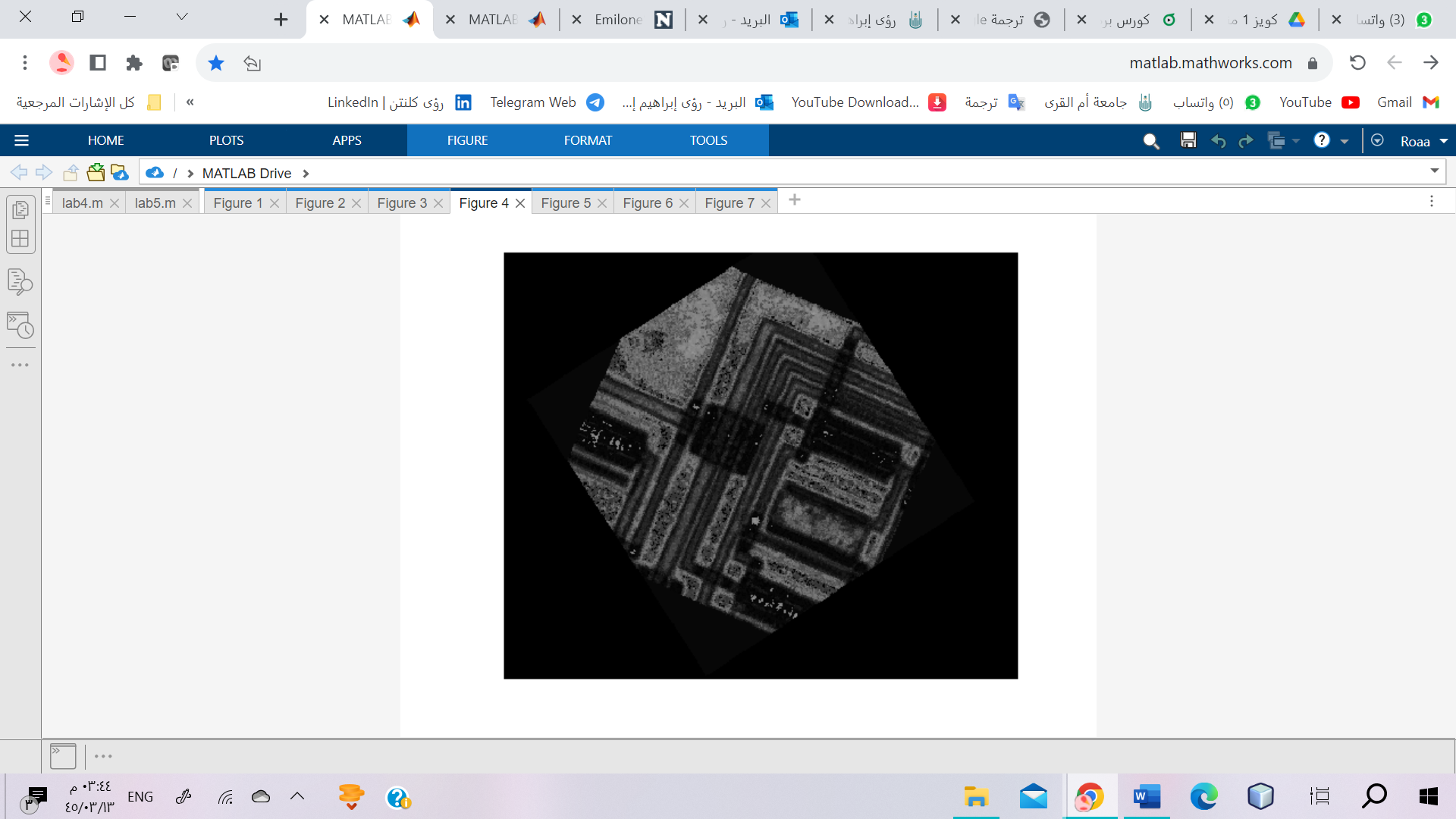
[posc, posr] = find(result == 1);

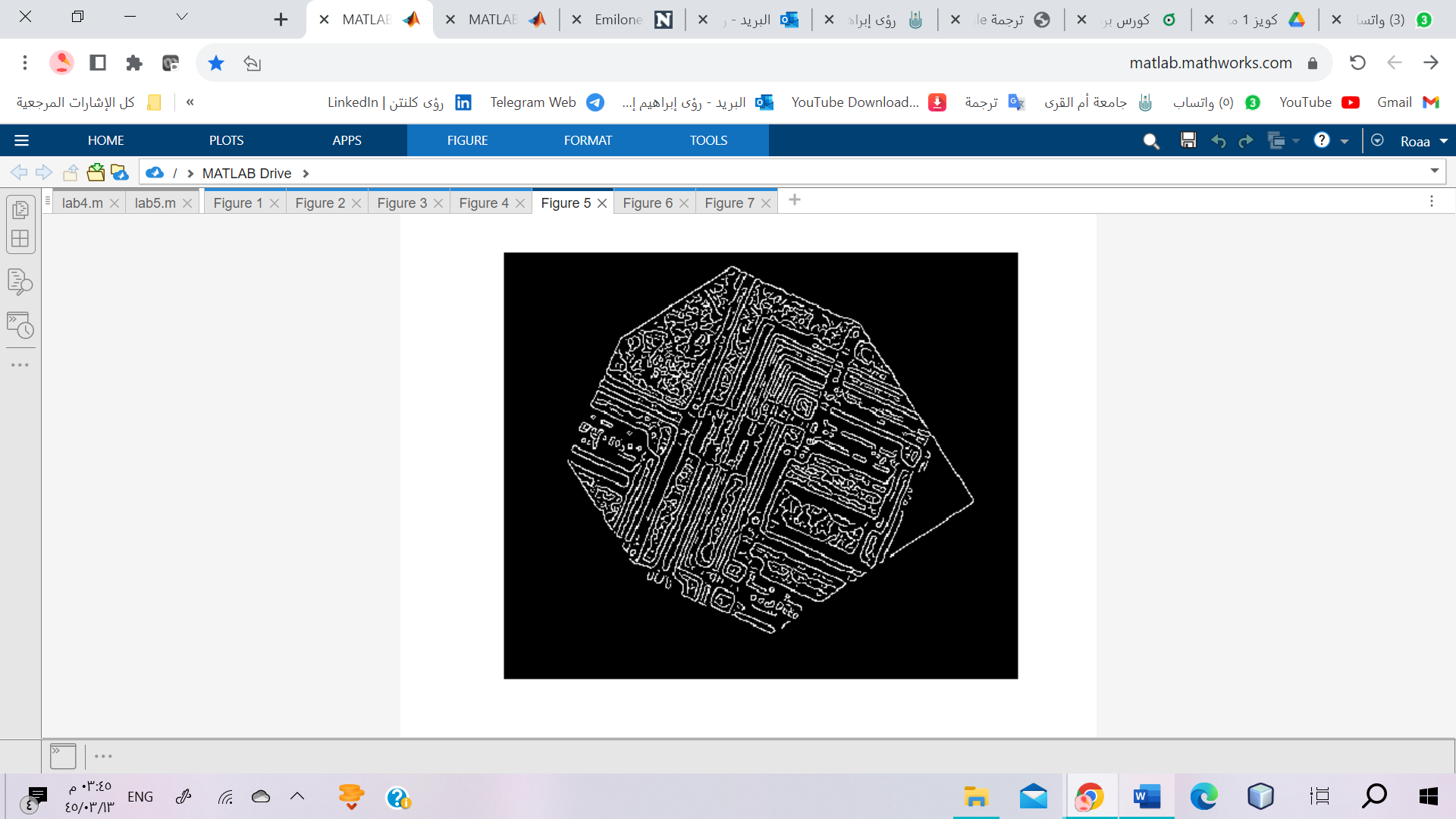
cnt ;

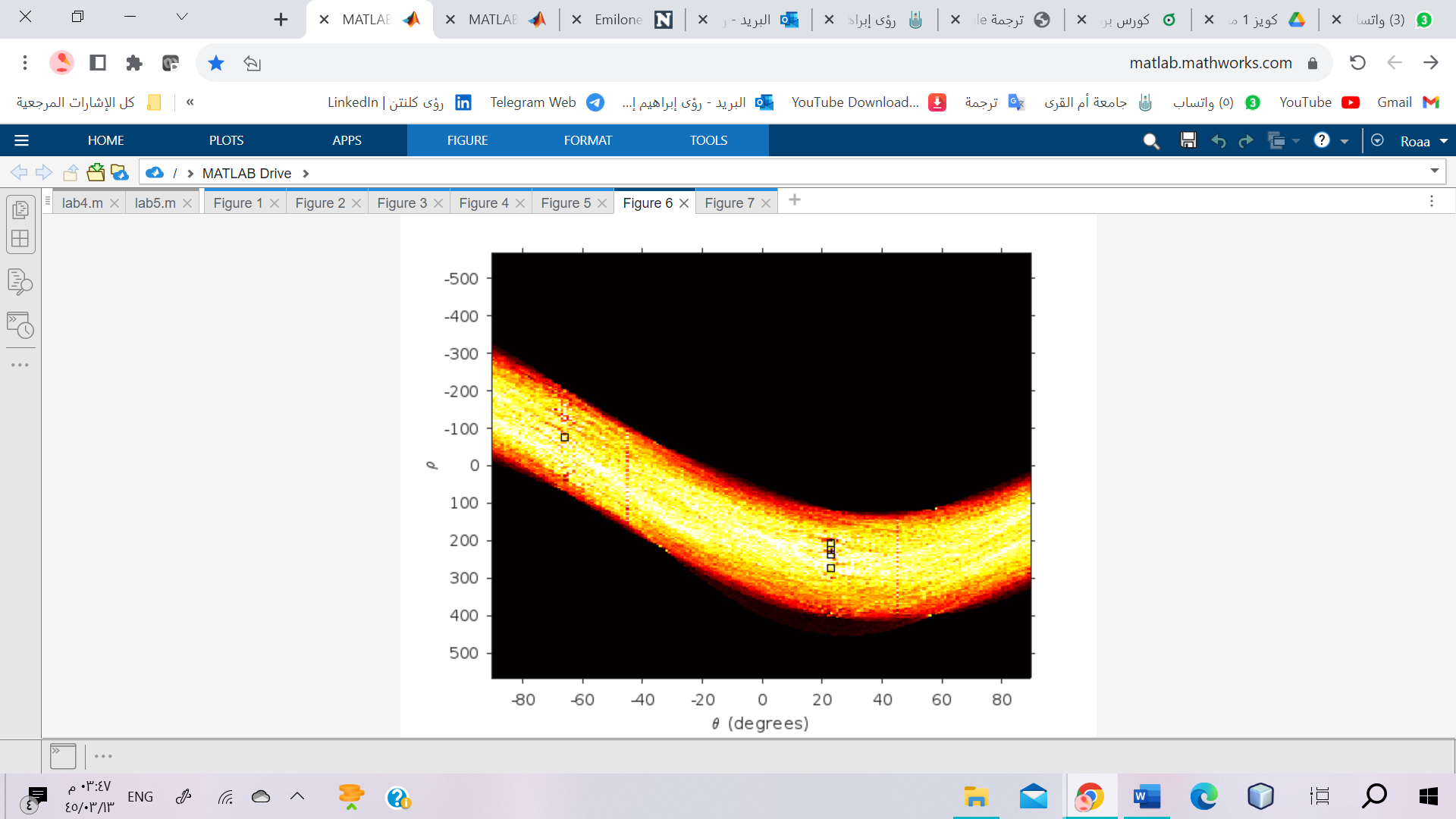
imshow(img);

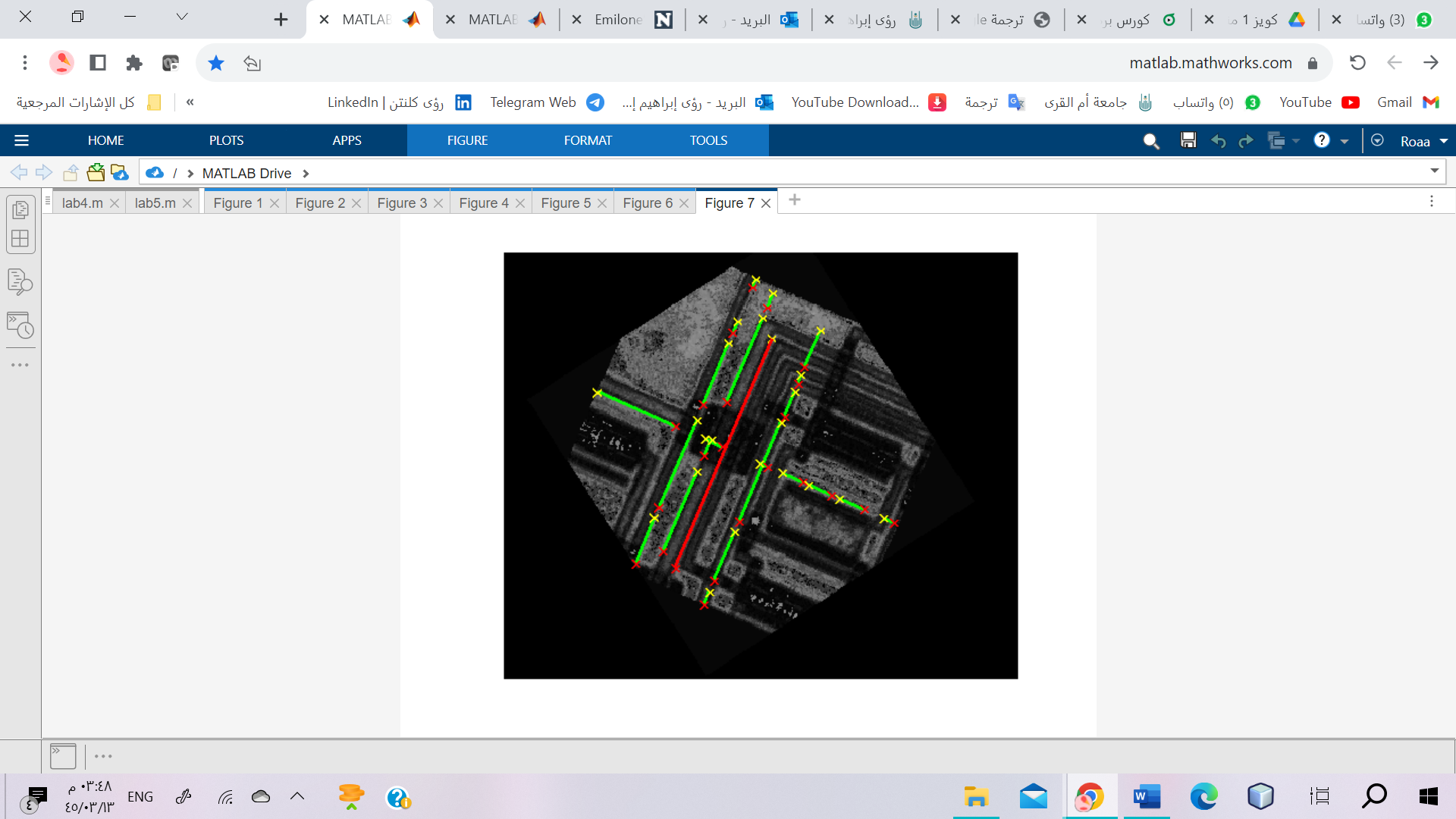
hold on;

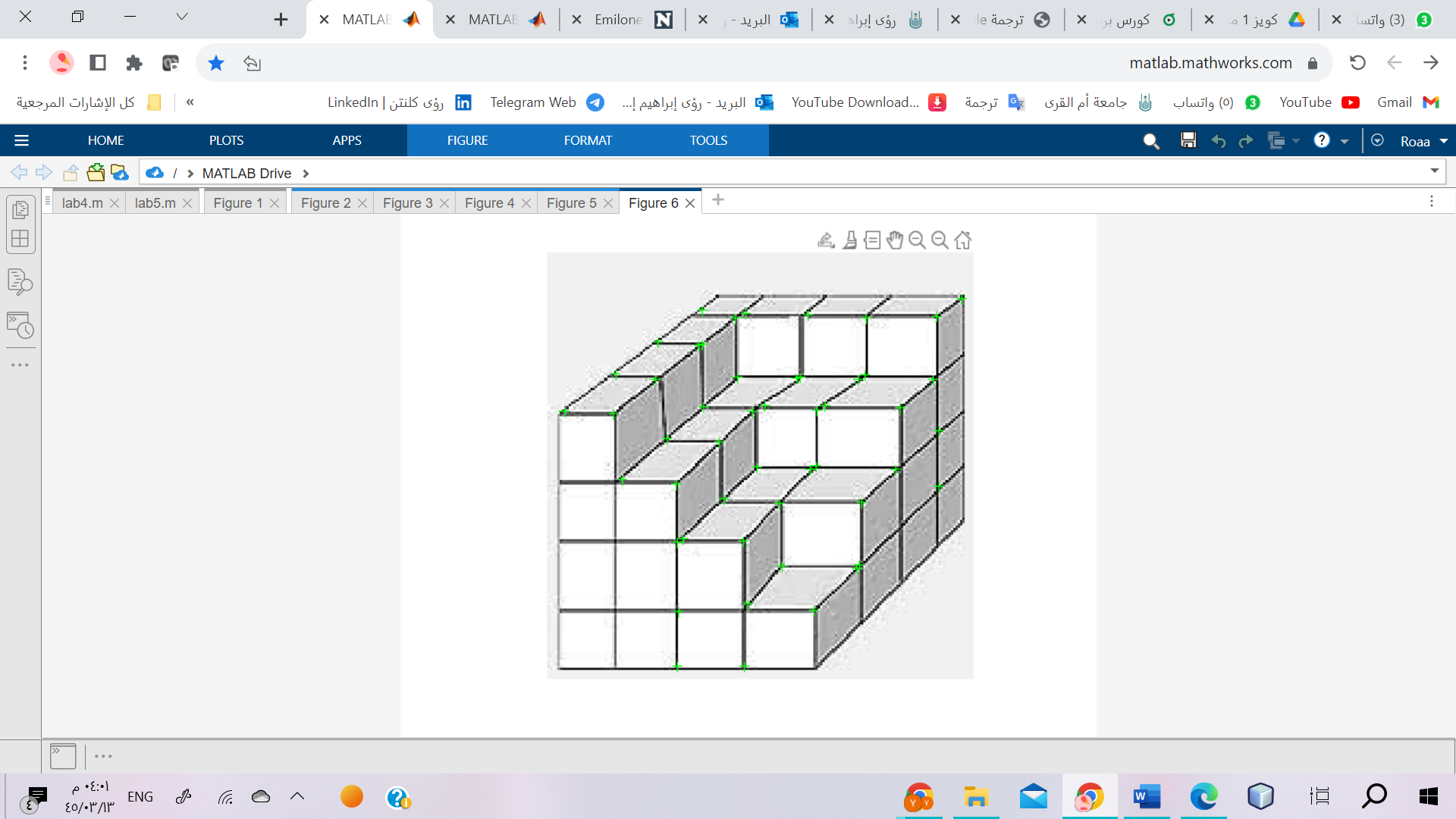
plot(posr,posc,'r\*');

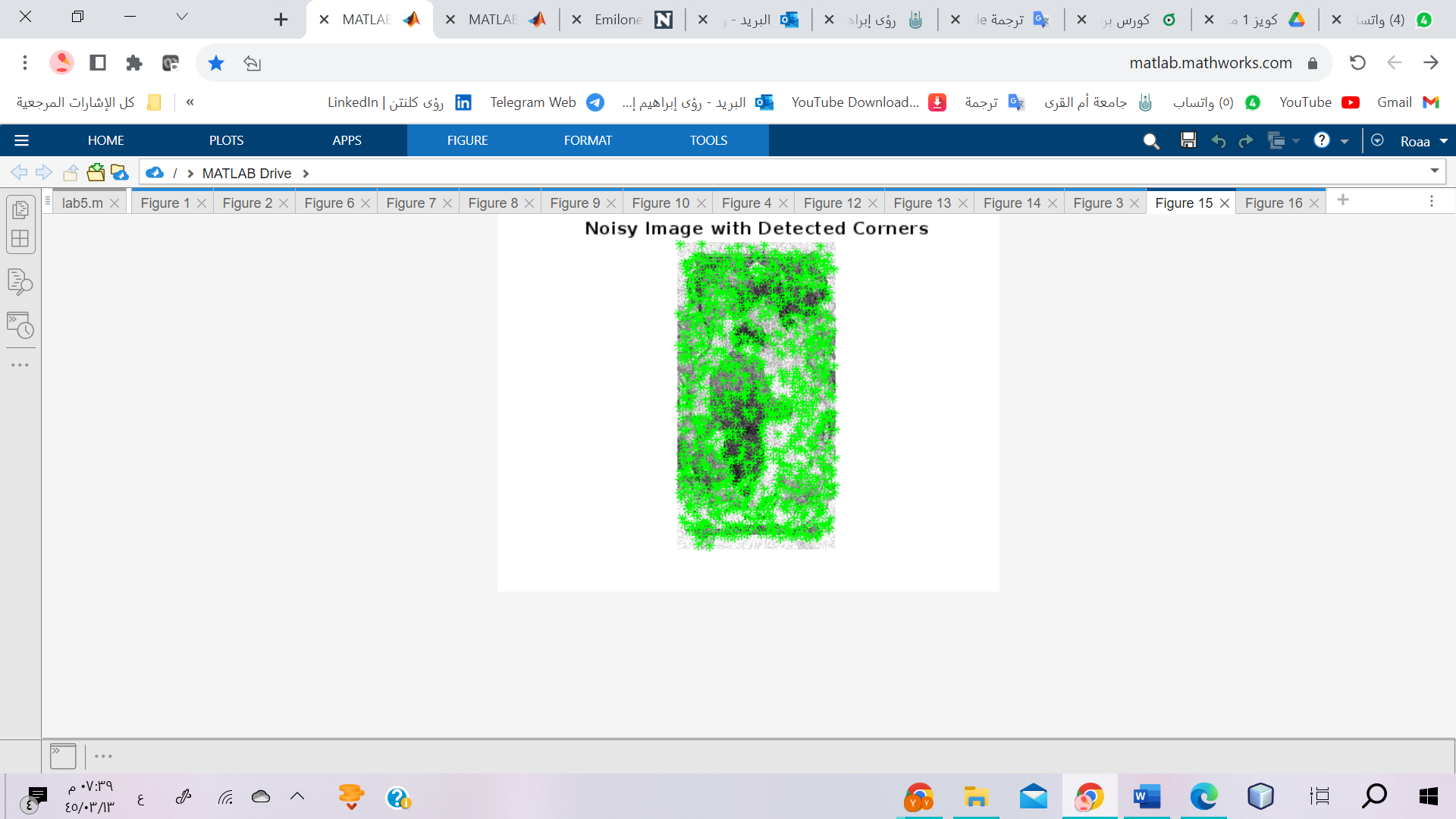


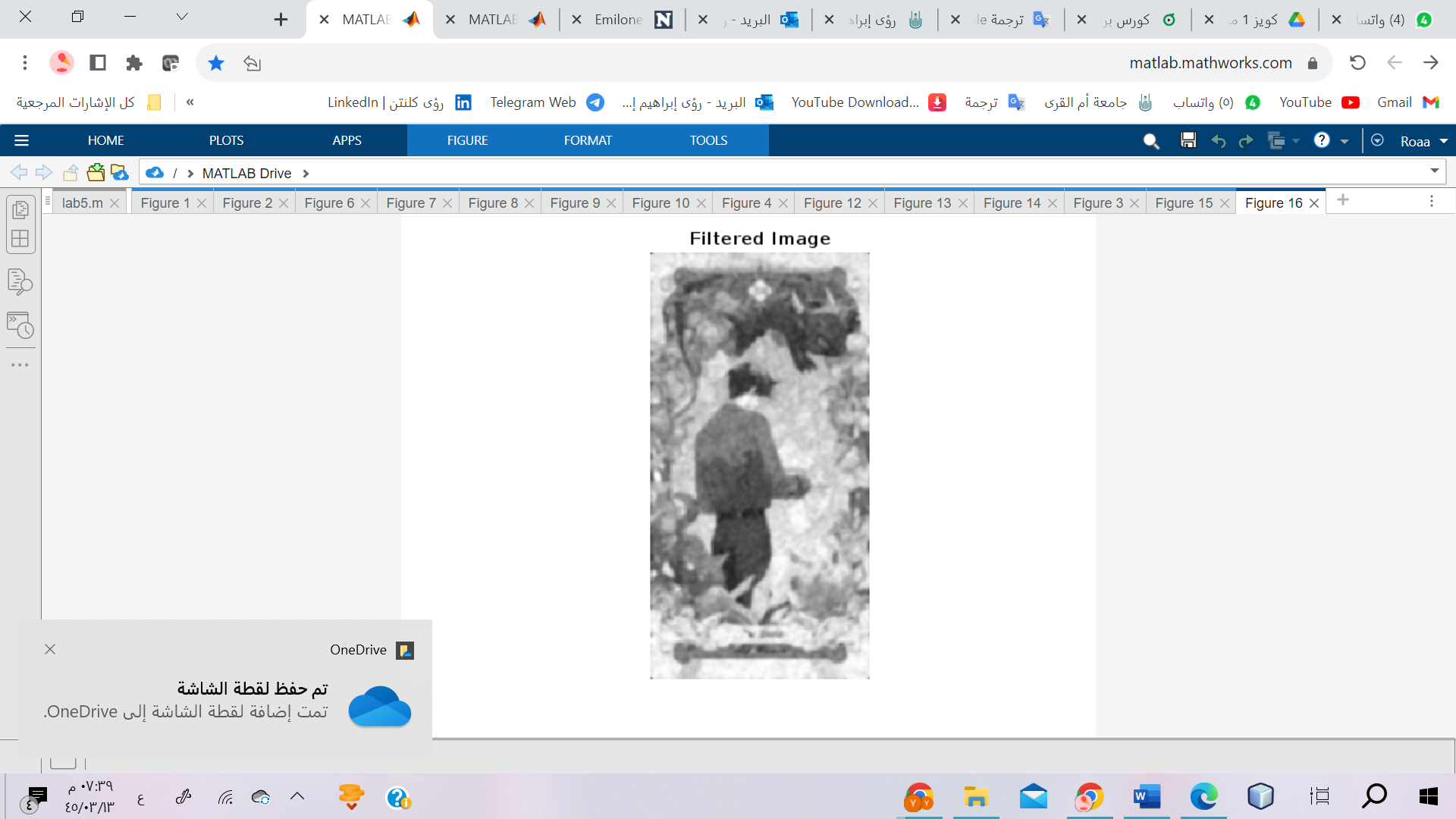
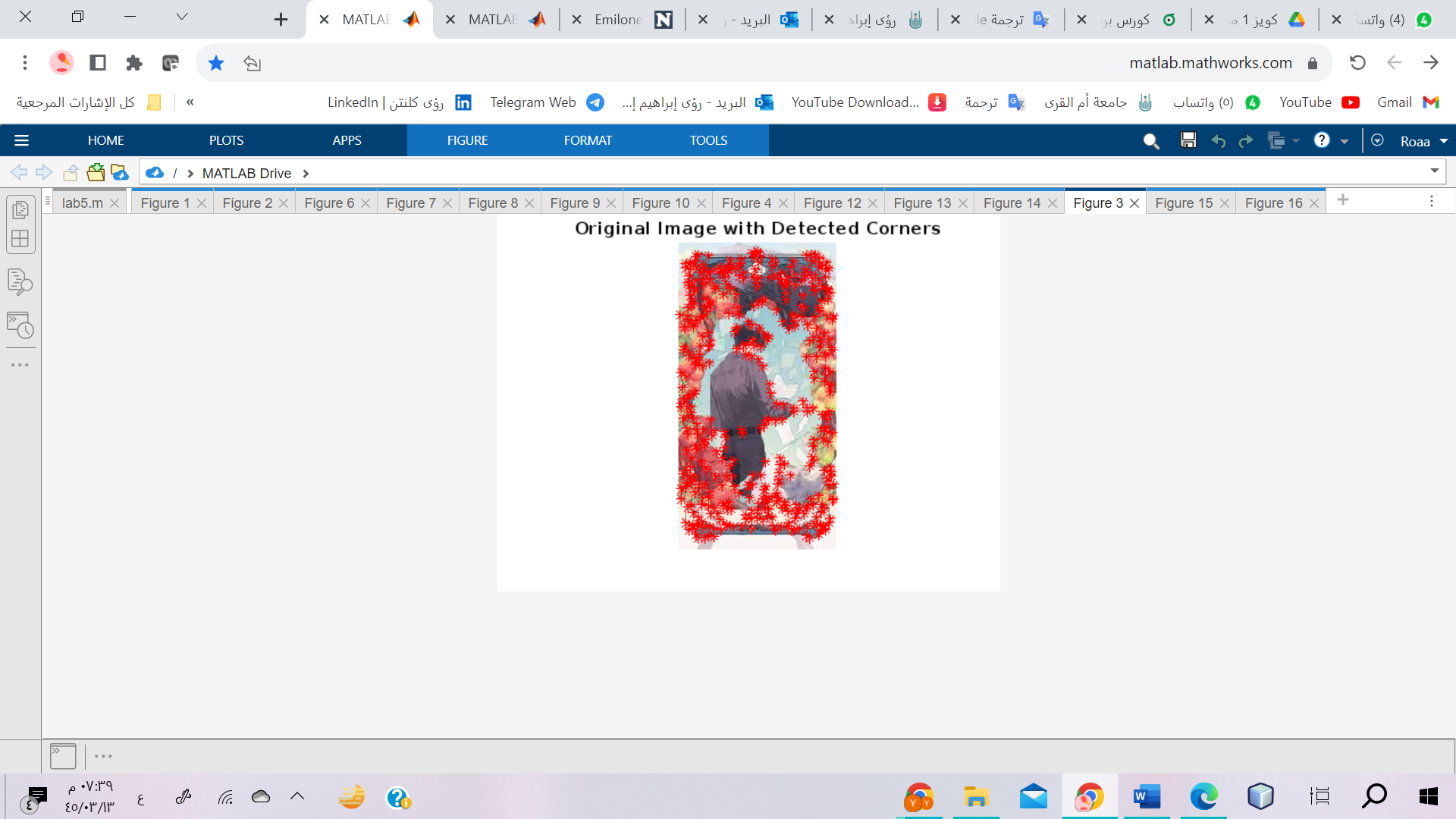












%Q2

I = imread('images (4).jfif');

I\_gray = rgb2gray(I);

% Apply the Harris corner detector to the original image

% and obtain the corner coordinates

corners\_original = detectHarrisFeatures(I\_gray);

coords\_original = corners\_original.Location;

% Add noise to the image

noisy\_image = imnoise(I\_gray, 'gaussian', 0, 0.01);

% Apply the Harris corner detector to the noisy image and obtain the corner coordinates

corners\_noisy = detectHarrisFeatures(noisy\_image);

coords\_noisy = corners\_noisy.Location;

% Visualize the results

figure;%9

imshow(I);

hold on;

plot(coords\_original(:, 1), coords\_original(:, 2), 'r\*');

title('Original Image with Detected Corners');

hold off;

figure;%10

imshow(noisy\_image);

hold on;

plot(coords\_noisy(:, 1), coords\_noisy(:, 2), 'g\*');

title('Noisy Image with Detected Corners');

hold off;

% Apply median filtering to remove noise

denoised\_image = medfilt2(noisy\_image);

% Define the sigma value for Gaussian filtering

sigma = 1; % قم بضبط القيمة حسب الحاجة

% Apply Gaussian filtering to the denoised image

filtered\_image = imgaussfilt(denoised\_image, sigma);

% Display the filtered image

figure;%11

imshow(filtered\_image);

title('Filtered Image');