VIT-AP UNIVERSITY

PROJECT-1

TITLE- Input: Take a Campus Building image (e.g., CB, AB-1, AB-2, Rock plaza, Food Court, Hostel Buildings ....)
Write a program:
a) To Determine edges using Canny Edge detector
b) To identify Corners using Harris / Hessian Corner detector
c) Find edges and smooth boundaries using Hough transform

COURSE CODE: CSE4047
COURSE NAME : COMPUTER VISION

Name: Syed. Mahammed Sameer
RegNo: 21bce8463
Date: 27-08-2024

Google Drive : Link

Steps:

- **Load the Image**:

Read an image into MATLAB using the `imread` function.

- **Rotate the Image**:

Use the `imrotate` function to rotate the image by a specified angle.

- **Scale the Image**:

Use the `imresize` function to resize the image by a specified scaling factor.

- **Translate the Image**:

Define a translation matrix to shift the image in the x and y directions.

Use the `affine2d` and `imwarp` functions to apply the translation to the image.

- **Display the Images**:

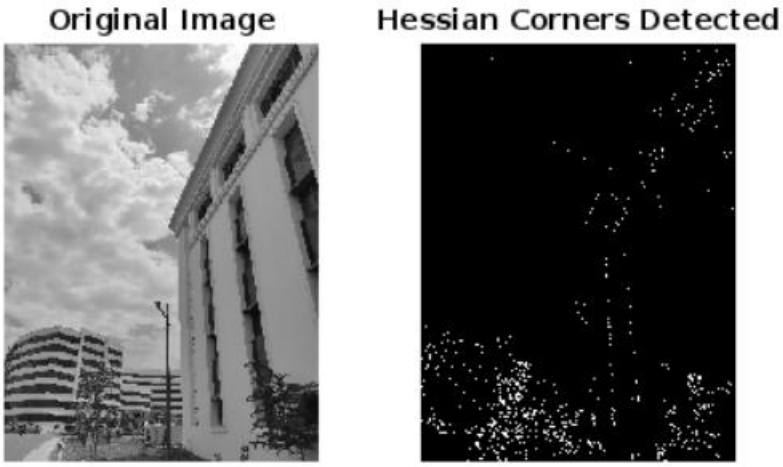
Display the original image, rotated image, scaled image, and translated image using the `imshow` function for visual comparison.

Code:

```
I = imread('clg.jpg');
if size(I,3) == 3
I = rgb2gray(I);
End

I = double(I);

sobel_x = [-1 0 1; -2 0 2; -1 0 1];
sobel_y = sobel_x';

Ix = conv2(I, sobel_x, 'same');
Iy = conv2(I, sobel_y, 'same');

Ixx = conv2(Ix, sobel_x, 'same');
Iyy = conv2(Iy, sobel_y, 'same');
Ixy = conv2(Ix, sobel_y, 'same');


detH = (Ixx .* Iyy) - (Ixy .^ 2);

detH = detH / max(detH(:));

corner_threshold = 0.01;
corners = detH > corner_threshold;

figure;
subplot(1,2,1), imshow(I, []), title('Original Image');
```

```
subplot(1,2,2), imshow(corners), title('Hessian Corners Detected');
```

Output:



Code:

```
% Harris Corner Detection

% Read the input image
img = imread('clg.jpg');
if size(img, 3) == 3
img = rgb2gray(img);
end

% Step 1: Compute Image Gradients
Ix = imfilter(double(img), fspecial('sobel')');
Iy = imfilter(double(img), fspecial('sobel'));

% Step 2: Compute Products of Gradients
Ix2 = Ix .* Ix;
Iy2 = Iy .* Iy;
Ixy = Ix .* Iy;

% Step 3: Gaussian Blurring
sigma = 1.0;
G = fspecial('gaussian', 5, sigma);
Ix2_blurred = imfilter(Ix2, G, 'same');
Iy2_blurred = imfilter(Iy2, G, 'same');
Ixy_blurred = imfilter(Ixy, G, 'same');

% Step 4: Compute Harris Response
k = 0.01; % Sensitivity parameter
R = zeros(size(img));
for i = 1:size(img, 1)
for j = 1:size(img, 2)
M = [Ix2_blurred(i,j), Ixy_blurred(i,j); Ixy_blurred(i,j), Iy2_blurred(i,j)];
R(i,j) = det(M) - k * trace(M)^2;
end
end
```

```
% Step 5: Thresholding
threshold = 0.02 * max(R(:));
corners = R > threshold;

% Step 6: Non-Maximum Suppression
corners = non_maximum_suppression(corners, R);

% Display results
figure;
subplot(1, 3, 1); imshow(img); title('Original Image');
subplot(1, 3, 2); imshow(corners, []); title('Detected Corners');
subplot(1, 3, 3); imshow(img); title('Original Image with Corners');

% Plot corners on the original image
[y_coords, x_coords] = find(corners);
hold on;
plot(x_coords, y_coords, 'r.', 'MarkerSize', 5);
hold off;

% Non-Maximum Suppression Function
function corners = non_maximum_suppression(binaryImage, R)
[rows, cols] = size(R);
corners = zeros(size(binaryImage));
se = strel('square', 3);
dilatedR = imdilate(R, se);

% Retain only local maxima
corners = binaryImage & (R == dilatedR);
end
```

Output: