University of Balamand

Faculty of Engineering

Departments of Electrical & Computer Engineering

Course: CPEN528 Machine Vision

Assignment 2: Corner Detection

Submitted to: **Dr. Issam Dagher**

Prepared by: **Hala Msalem** (A2020119)

Thursday, April 25, 2024

Table of Contents

Table of Figures

Objective

The objective of this assignment is to apply the Harris Corner Detector to two versions of an image to identify the corners in each version. After the corners are detected, the next step is to match the corners between the two images. This matching process involves finding corresponding corners in the two images that represent the same physical point in the scene.

MATLAB Code

Sobel's mask is applied to both image 1 and image 2 to compute Ix and Iy.

Image 1

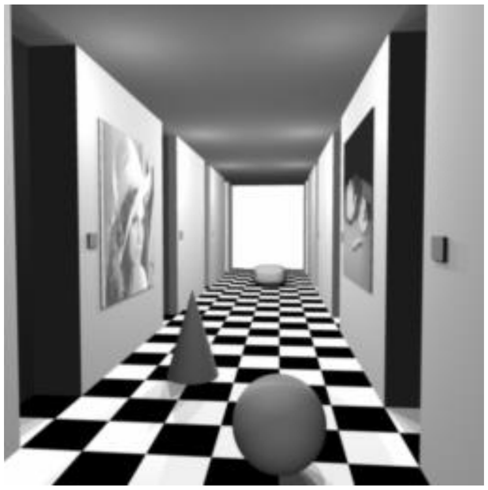

Figure 1: Image 1

Finding Ix and Iy

```
%sobel mask on I

I = imread('stereo-corridor_l (1).gif');


g1 = [-1 0 +1;

    -2 0 +2;

    -1 0 +1];

I1 = conv2(I, g1, 'same'); %fx

Ix=abs(I1); %|fx|


g2= [-1 -2 -1;

    0 0 0;

    1 2 1];

I3 = conv2(I, g2, 'same'); %fy

Iy=abs(I3); %|fy|
```

Corners detection

In this step, I computed the M matrix for each pixel in Image 1 and calculate its eigenvalues. Then, I examined whether the minimum eigenvalue is higher than the chosen threshold. If it was higher, the corresponding pixel is identified as a corner and assigned the minimum eigenvalue in the new lambda matrix. Otherwise, it is assigned a value of zero.

```
lamda_matrix = zeros(size(I));
for i = 2:size(I, 1) -1
    for j = 2:size(I, 2) -1
        a=  (Ix(i, j))^2 + (Ix(i, j - 1))^2+ (Ix(i, j + 1))^2 + (Ix(i - 1, j +
1))^2+(Ix(i + 1, j - 1))^2+ (Ix(i - 1, j))^2 + (Ix(i + 1, j))^2+ (Ix(i - 1, j -
1))^2+(Ix(i + 1, j + 1))^2;
        d=  (Iy(i, j))^2 + (Iy(i, j - 1))^2+ (Iy(i, j + 1))^2 + (Iy(i - 1, j +
1))^2+(Iy(i + 1, j - 1))^2+ (Iy(i - 1, j))^2 + (Iy(i + 1, j))^2+ (Iy(i - 1, j -
1))^2+(Iy(i + 1, j + 1))^2;
        b=(Ix(i, j))*(Iy(i, j))+(Ix(i, j - 1))*(Iy(i, j - 1))+(Ix(i, j + 1))*(Iy(i, j
+ 1))+(Ix(i - 1, j + 1))*(Iy(i - 1, j + 1))+(Ix(i + 1, j - 1))*Iy(i + 1, j - 1)+(Ix(i
- 1, j))*(Iy(i - 1, j))+ (Ix(i + 1, j))* (Iy(i + 1, j))+(Ix(i - 1, j - 1))*(Iy(i - 1,
j - 1))+(Ix(i + 1, j + 1))*(Iy(i + 1, j + 1));
        c=b;
        M=[a b;
            c d];
        eigenvalues = eig(M);
        lamda_min=min(eigenvalues);
        if (lamda_min> 200)
            lamda_matrix(i,j)=lamda_min;
        else  lamda_matrix(i,j)=0;
        end
    end
end
```

Non-maximum suppression

In this step, I applied non-maximum suppression to reduce the number of detected corners to a more accurate representation of the true corners in the image.

```
lamda_max = zeros(size(I));
% Non-maximum suppression
for i = 2:size(lamda_matrix, 1) -1
    for j = 2:size(lamda_matrix, 2) -1
        if lamda_matrix(i, j) <= lamda_matrix(i, j - 1) || lamda_matrix(i, j) <=
lamda_matrix(i, j + 1) || lamda_matrix(i, j) <= lamda_matrix(i - 1, j + 1) ||
lamda_matrix(i, j) <= lamda_matrix(i + 1, j - 1)|| lamda_matrix(i, j) <=
lamda_matrix(i - 1, j) || lamda_matrix(i, j) <= lamda_matrix(i + 1, j) ||
lamda_matrix(i, j) <= lamda_matrix(i - 1, j - 1) || lamda_matrix(i, j) <=
lamda_matrix(i + 1, j + 1)
            lamda_max(i, j) = 0;
        else lamda_max(i, j) = 1;
        end
    end
end
```

Results

The code below highlights the detected corners in image 1 as shown in Figure 2.

```
imshow(I);
hold on;
```

```
% Get the coordinates of the points in the binary image

[row, col] = find(lamda_max == 1);

% Set the size and color of the points

pointSize = 10;

pointColor = 'r';

% Display points on the image

scatter(col, row, pointSize, pointColor, 'filled');

% Add a title to the figure

title('Image 1 with Points');

% Hold off to stop overlaying on the image

hold off;

figure;
```

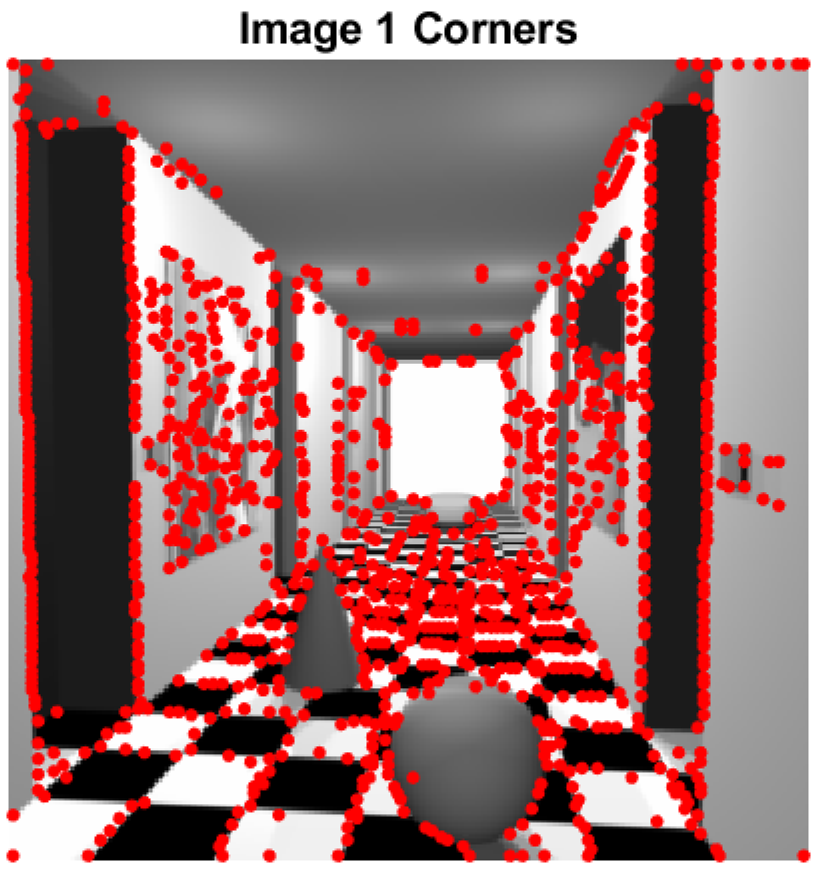

Figure 2: Image 1 corners

Image 2

Getting Image 2

I obtained image 2 from image 1 by cropping its width using the following code:

```
% Get the size of the original image

[rows, cols, ~] = size(I);

% Define the cropping dimensions

topCrop = 1;

bottomCrop = rows - 1;

leftCrop = 20;

rightCrop = cols - 20;

% Crop the image using array indexing

CI = I(topCrop:bottomCrop, leftCrop:rightCrop, :);
```

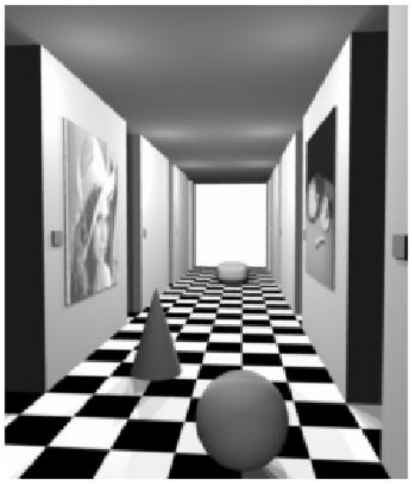

Figure 3: Image 2

Finding Ix and Iy

```
%sobel mask on CI (CI is image 2)

CI1 = conv2(CI, g1, 'same'); %fx

CIx=abs(CI1); %|fx|

CI3 = conv2(CI, g2, 'same'); %fy

CIy=abs(CI3); %|fy|
```

Corners detection

In this step, I computed the M matrix for each pixel in Image 2 and calculate its eigenvalues. Then, I checked whether the minimum eigenvalue is higher than the chosen threshold. If it was higher, the corresponding pixel is identified as a corner and assigned the minimum eigenvalue in the new lambda matrix. Otherwise, it is assigned a value of zero.

```
Clamda_matrix = zeros(size(CI));
for i = 2:size(CI, 1) -1
    for j = 2:size(CI, 2) -1
        Ca=  (CIx(i, j))^2 + (CIx(i, j - 1))^2+ (CIx(i, j + 1))^2 + (CIx(i - 1, j +
1))^2+(CIx(i + 1, j - 1))^2+ (CIx(i - 1, j))^2 + (CIx(i + 1, j))^2+ (CIx(i - 1, j -
1))^2+(CIx(i + 1, j + 1))^2;
        Cd=  (CIy(i, j))^2 + (CIy(i, j - 1))^2+ (CIy(i, j + 1))^2 + (CIy(i - 1, j +
1))^2+(CIy(i + 1, j - 1))^2+ (CIy(i - 1, j))^2 + (CIy(i + 1, j))^2+ (CIy(i - 1, j -
1))^2+(CIy(i + 1, j + 1))^2;
        Cb=(CIx(i, j))*(CIy(i, j))+(CIx(i, j - 1))*(CIy(i, j - 1))+(CIx(i, j +
1))*(CIy(i, j + 1))+(CIx(i - 1, j + 1))*(CIy(i - 1, j + 1))+(CIx(i + 1, j - 1))*CIy(i
+ 1, j - 1)+(CIx(i - 1, j))*(CIy(i - 1, j))+ (CIx(i + 1, j))* (CIy(i + 1, j))+(CIx(i
- 1, j - 1))*(CIy(i - 1, j - 1))+(CIx(i + 1, j + 1))*(CIy(i + 1, j + 1));
        Cc=Cb;
        CM=[Ca Cb;
            Cc Cd];
        Ceigenvalues = eig(CM);
        Clamda_min=min(Ceigenvalues);
        if (Clamda_min> 200)
            Clamda_matrix(i,j)=Clamda_min;
        else  Clamda_matrix(i,j)=0;
        end
    end
end
```

Non-maximum suppression

In this step, I applied non-maximum suppression to reduce the number of detected corners to a more accurate representation of the true corners in the image.

```
Clamda_max = zeros(size(I));
% Non-maximum suppression
for i = 2:size(Clamda_matrix, 1) -1
    for j = 2:size(Clamda_matrix, 2) -1
        if Clamda_matrix(i, j) <= Clamda_matrix(i, j - 1) || Clamda_matrix(i, j) <=
Clamda_matrix(i, j + 1) || Clamda_matrix(i, j) <= Clamda_matrix(i - 1, j + 1) ||
Clamda_matrix(i, j) <= Clamda_matrix(i + 1, j - 1)|| Clamda_matrix(i, j) <=
Clamda_matrix(i - 1, j) || Clamda_matrix(i, j) <= Clamda_matrix(i + 1, j) ||
Clamda_matrix(i, j) <= Clamda_matrix(i - 1, j - 1) || Clamda_matrix(i, j) <=
Clamda_matrix(i + 1, j + 1)
            Clamda_max(i, j) = 0;
        else Clamda_max(i, j) = 1;
        end
    end
end
```

Results

The code below highlights the detected corners in image 1 as shown in Figure 4.

```
imshow(CI);
hold on;
% Get the coordinates of the points in the binary image
```

```
[Crow, Ccol] = find(Clamda_max == 1);

% Set the size and color of the points

pointSize = 10;

pointColor = 'r';

% Display points on the image

scatter(Ccol, Crow, pointSize, pointColor, 'filled');

% Add a title to the figure

title('Image 2 Corners');

% Hold off to stop overlaying on the image

hold off;
```

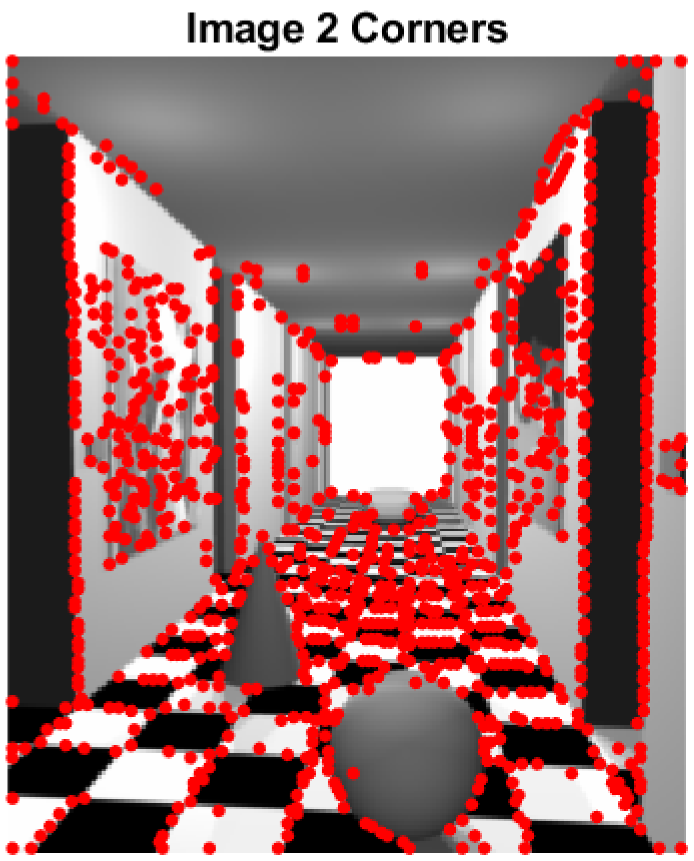

Figure 4: Image 2 corners

Corners matching

Now, to match the corners in both images, I selected a corner in image 1 with the coordinates X=101 and Y=154, as shown in Figure 5, to be matched with the corresponding corner in image 2, where the pixel has the coordinates X=82 and Y=154. To accomplish this, I began by identifying all corners on row 154 in image 2. Then, I computed the difference between the window around each corner found in this row in image 2 and the window around the selected corner in image 1, and I stored these difference values in an array. After that, I determined the minimum difference value in this array and obtained its index. Finally, I found the coordinates of the corresponding pixel using this index. The matched corner in image 2 is shown in Figure 6.

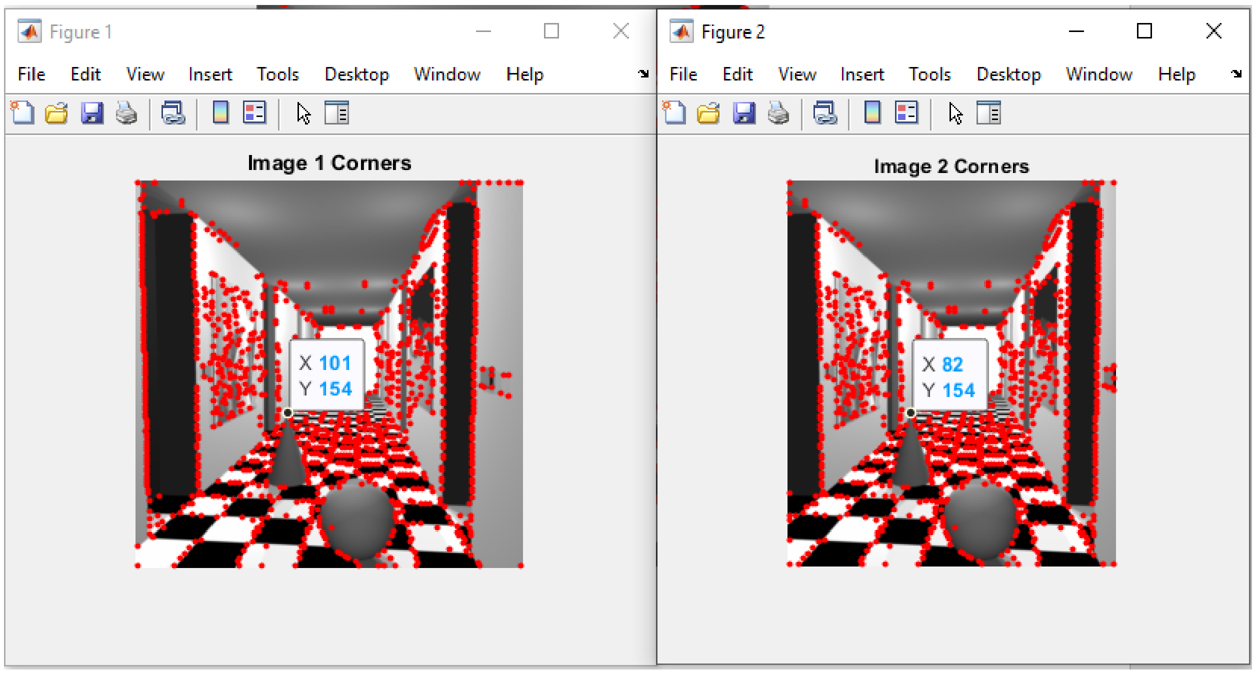
Figure 1
File Edit View Insert Tools Desktop Window Help
Image 1 Corners
X 101
Y 154
Figure 2
File Edit View Insert Tools Desktop Window Help
Image 2 Corners
X 82
Y 154

Figure 5: Corner to match

```
k=101; % coordinates of chosen corner to match in image 1

l=154; % coordinates of chosen corner to match in image 1

coordinates =[Crow, Ccol];

rowsWithRow154 = coordinates(:, 1) == 154;
```

```
% Get the columns where the row value is equal to 154 in image 2

columns = coordinates(rowsWithRow154, 2);

arraySize = size(columns);

W = zeros(arraySize); % the difference values array

for i = 1:length(columns)

    W(i)=(I(l,k)-CI(l,columns(i)))^2+(I(l-1,k)-CI(l-1,columns(i)))^2+(I(l+1,k)-
CI(l+1,columns(i)))^2+(I(l,k-1)-CI(l,columns(i)-1))^2+(I(l,k+1)-
CI(l,columns(i)+1))^2+(I(l+1,k+1)-CI(l+1,columns(i)+1))^2+(I(l-1,k-1)-CI(l-
1,columns(i)-1))^2+(I(l-1,k+1)-CI(l-1,columns(i)+1))^2+(I(l+1,k-1)-CI(l+1,columns(i)-
1))^2;

end

minIndex = find(W == min(W)); % index of the right corner ( minimum difference)

match=[154, columns(minIndex)]; % coordinate of the right corner
```

Below is the code to highlight the matched corner in image 2.

```
figure;

rrow = 154;

rcol = columns(minIndex);

imshow(CI);

hold on;

% Set the size and color of the points

pointSize = 10;

pointColor = 'g';

% Display points on the image

scatter(rcol, rrow, pointSize, pointColor, 'filled');

% Add a title to the figure

title('Matched corner in Image 2');

% Hold off to stop overlaying on the image

hold off
```

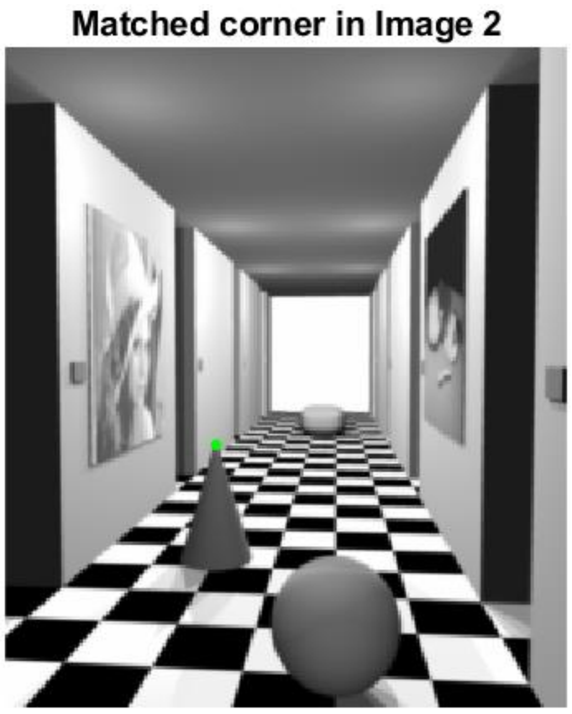

Figure 6: Matched corner.

Conclusion

To conclude, this assignment focused on the application of the Harris Corner Detector and corner matching in image processing. I started by detecting corners with the Harris Corner Detector in both images and then matched the appropriate corner. I used several image processing methods, such as corner detection, non-maximum suppression, and corner matching.