Computer vision practical assignment 2

Hough Transform

20321233 zou letian

20321308 cao Xinyang

20321309 chen hanzheng

1.Objective

Study of the Hough transformation to find of geometric primitives.

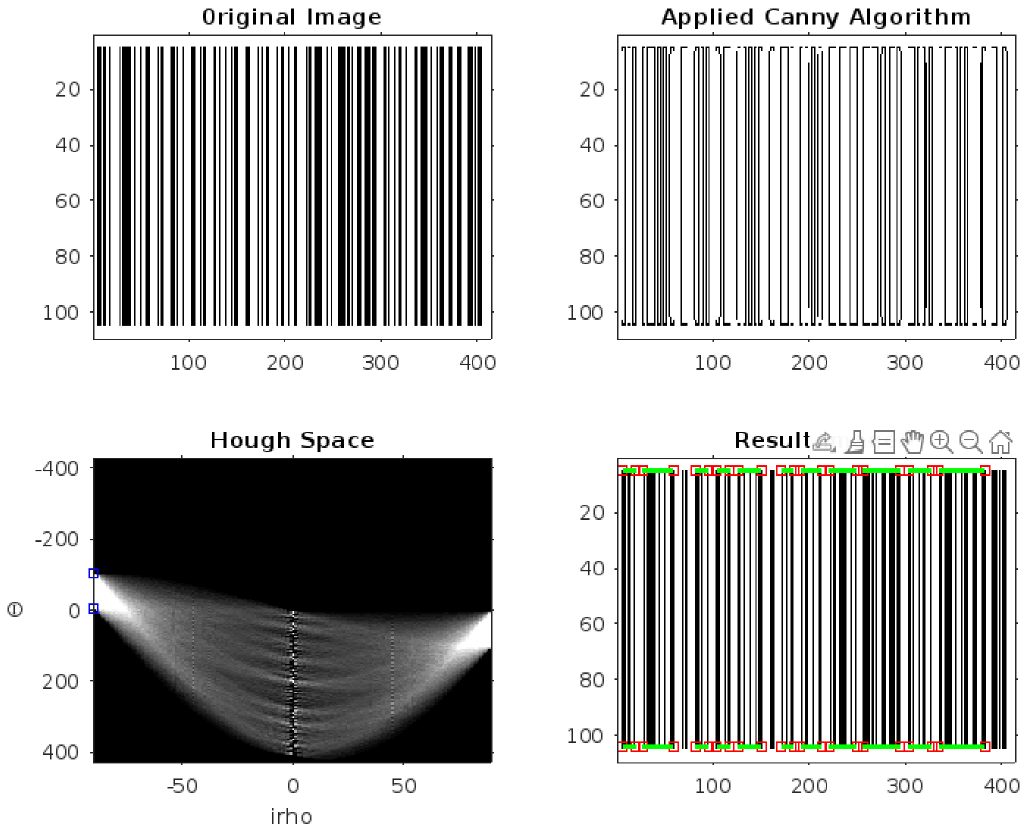
2.Procedure of Practical Assignment Performing

1. Search for lines. Select three arbitrary images containing lines. Perform to search for straight lines using the Hough transform both for the original image and for the image obtained using any differential operator. Plot the found lines on the original image. Mark the start and end points of the lines. Determine the lengths of the shortest and longest lines, calculate the number of lines found.
2. Search for circles. Select three arbitrary images containing circles. Search for circles of both a certain radius and from a given range using the Hough transform, both for the original image and for the image obtained using any differential operator. Plot the found circles on the original image.
3. Optional. Implement the classic Hough transform algorithms for lines and circles. Compare your implementation results with ones obtained in the first two points of the assignment. Highlight the selected points in the Hough parameter space.

Note. Please note that when doing the practical assignment you are not allowed to use the “Lenna” image or any other image that was used either in this book or during the presentation.

2.1 Part1　　Search for lines.

2.1.1 original images，resulting images



2.1.2 code of the scripts

```
clc
clear

%%
%First Task
%image1
figure
I = imread ('bar.png');
I = rgb2gray (I);
subplot(2,2,1);
imshow(I);
```

```
title('Original Image');
axis on,axis normal, hold on
```

```
%select the contours using the Canny algorithm
Iedge = edge(I,'Canny');
subplot(2,2,2);
imshow(~Iedge);
title('Applied Canny Algorithm');
axis on,axis normal, hold on
```

```
% apply the Hough transform using the hough()
[H,Theta,rho] = hough(Iedge);
subplot(2,2,3);
imshow(imadjust( rescale(H)),'YData',rho,'XData',Theta,'InitialMagnification','fit');
xlabel('irho'),ylabel('\Theta')
title('Hough Space');
axis on, axis normal, hold on
```

```
%calculate the peaks using the houghpeaks() function in the Hough space
peaks = houghpeaks(H,100,'threshold', ceil(0.5*max(H(:))));
x = Theta( peaks (:,2));
y = rho( peaks (:,1));
plot(x,y,'s','color','blue') ;
```

```
%Based on the peaks, we determine the straight lines using the houghlines()
lines = houghlines(Iedge,Theta,rho,peaks,'FillGap',5,'MinLength',10);
subplot(2,2,4);
imshow(I);
title('Result Image');
axis on,axis normal, hold on
```

```
minlen = 1000;
maxlen = 0;
for k = 1:length(lines)
    xy =[lines(k).point1; lines(k).point2] ;
    plot(xy(: ,1), xy(: ,2), 's ', 'color' , ' red ' ) ;
    plot(xy(: ,1),xy(: ,2),'Linewidth',2, 'color', 'green');

    len = norm(lines(k).point1 - lines(k).point2);
    %Find the length of the longest and shortest line
    if len < minlen
        minlen = len;
    end
    if len > maxlen
```

```
        maxlen = len;
    end
end
```

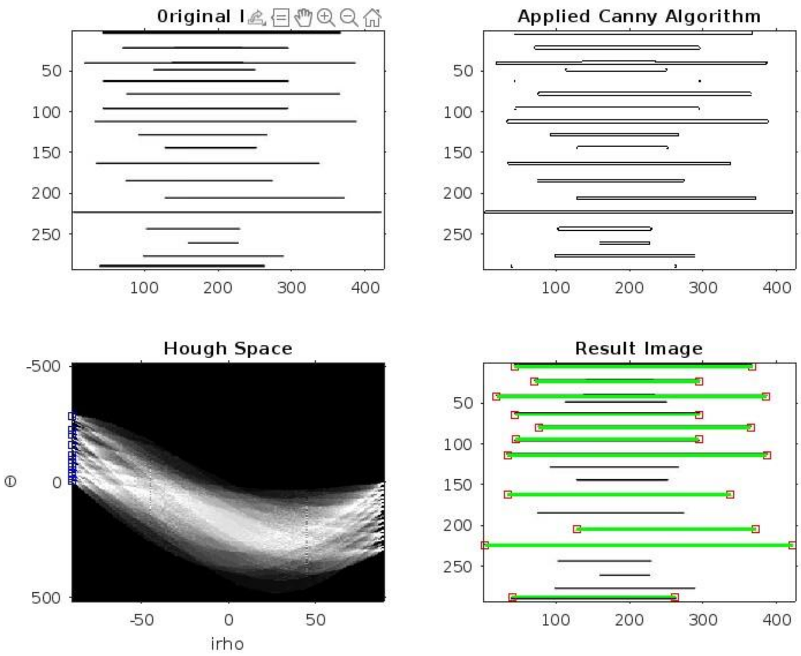
```
disp(['Minimum length: ',num2str(minlen)]);
disp(['Maximum length: ',num2str( maxlen)]);
disp(['Number of found lines:',num2str(length(lines))]);
```

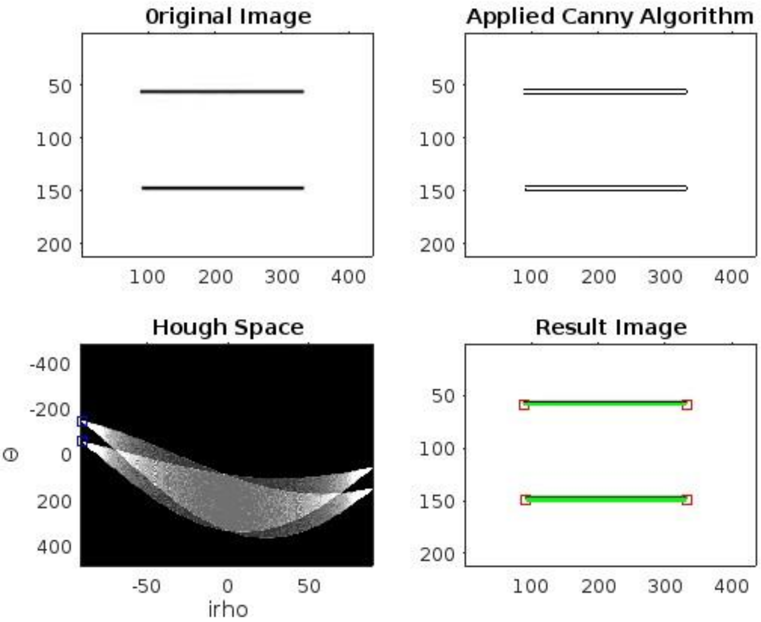
2.1.3 comments

The selected code is a MATLAB script that performs two tasks: detecting straight lines and detecting circles in images. For the first task, the script loads an image, converts it to grayscale, applies the Canny edge detection algorithm to select contours, and then applies the Hough transform to detect straight lines. The script then plots the original image, the Canny edge detection result, the Hough space, and the detected lines. Finally, the script outputs the minimum and maximum length of the detected lines and the number of lines found. For the second task, the script loads three images, applies the Canny edge detection algorithm to select contours, and then applies the circle transform Hough to detect circles. The script then plots the original image, the Canny edge detection result, and the detected circles. Note that the script uses several built-in MATLAB functions, including imread, rgb2gray, edge, hough, houghpeaks, houghlines, im2gray, imfindcircles, imshow, title, axis, hold, plot, norm, disp, and viscircles.

2.2 Part2　Search for circles.

2.2.1 original images, resulting images





2.2.2code of the scripts

```
%%
%image2
figure
I = imread ('line01.jpg');
```

```
I = rgb2gray (I);
subplot(2,2,1);
imshow(I);
title('Original Image');
axis on,axis normal, hold on

%select the contours using the Canny algorithm
Iedge = edge(I,'Canny');
subplot(2,2,2);
imshow(~Iedge);
title('Applied Canny Algorithm');
axis on,axis normal, hold on
```

```
% apply the Hough transform using the hough()
[H,Theta,rho] = hough(Iedge);
subplot(2,2,3);
imshow(imadjust( rescale(H)),'YData',rho,'XData',Theta,'InitialMagnification','fit');
xlabel('irho'),ylabel('\Theta')
title('Hough Space');
axis on, axis normal, hold on
```

```
%calculate the peaks using the houghpeaks() function in the Hough space
peaks = houghpeaks(H,100,'threshold', ceil(0.5*max(H(:))));
x = Theta( peaks (:,2));
y = rho( peaks (:,1));
plot(x,y,'s','color','blue') ;
```

```
%Based on the peaks, we determine the straight lines using the houghlines()
lines = houghlines(Iedge,Theta,rho,peaks,'FillGap',5,'MinLength',10);
subplot(2,2,4);
imshow(I);
title('Result Image');
axis on,axis normal, hold on
```

```
minlen = 1000;
maxlen = 0;
for k = 1:length(lines )
    xy =[lines(k).point1; lines(k).point2] ;
    plot(xy(: ,1), xy(: ,2), 's ', 'color' , ' red ' ) ;
    plot(xy(: ,1),xy(: ,2),'Linewidth',2, 'color', 'green');

    len = norm(lines(k).point1 - lines(k).point2);
    if len < minlen
        minlen = len;
    end
    if len > maxlen
        maxlen = len;
    end
end
```

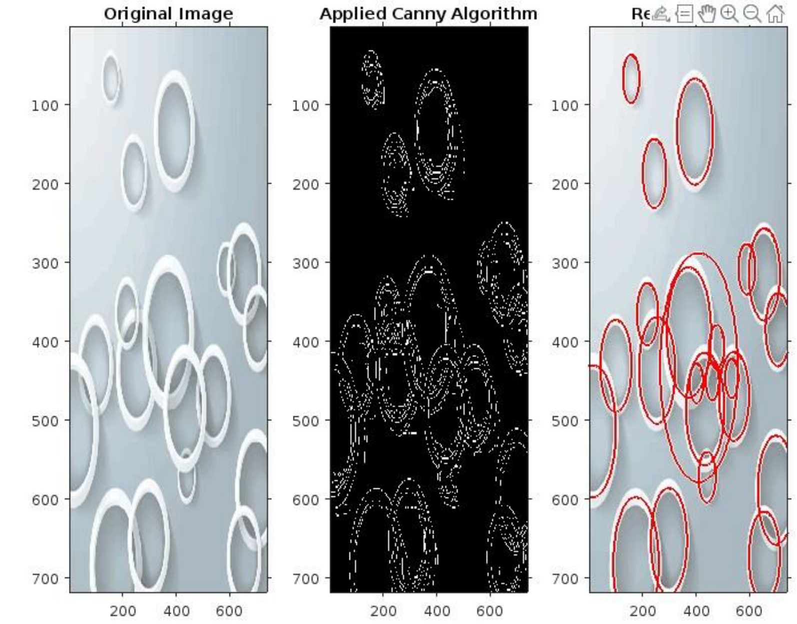
```
disp(['Minimum length: ',num2str(minlen)]);
disp(['Maximum length: ',num2str( maxlen)]);
disp(['Number of found lines:',num2str(length(lines))]);
```

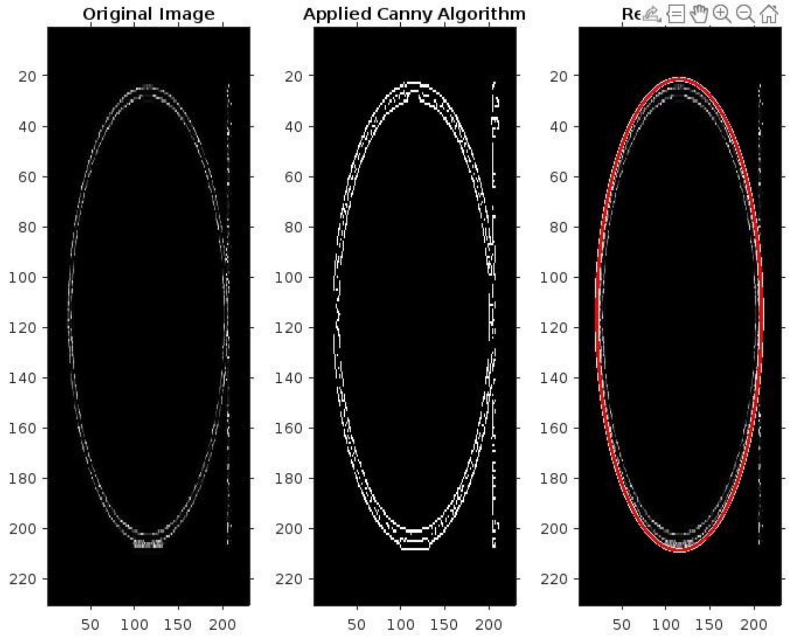
2.2.3 comments

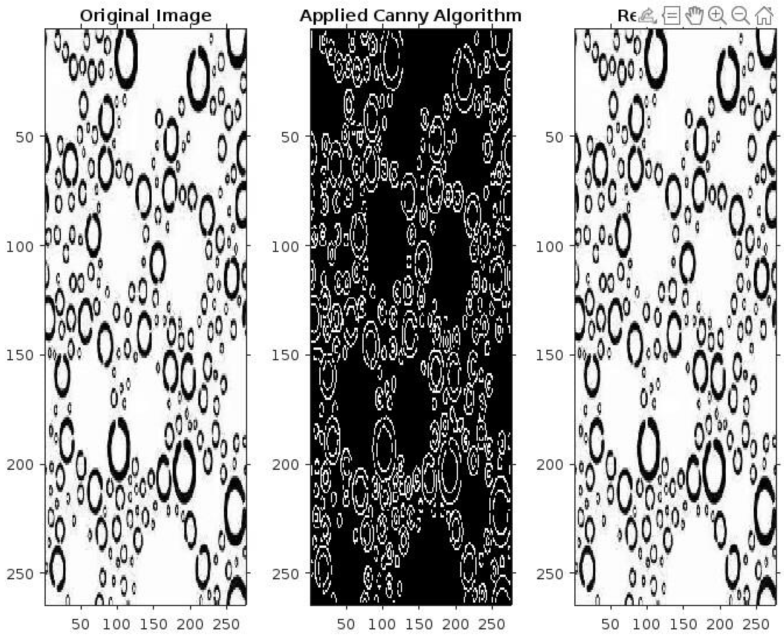
The selected code is a MATLAB script that performs Hough transform and circle detection on three different images. For each image, the script first displays the original image and then applies the Canny edge detection algorithm to extr. For each image, the script first displays the original image and then applies the Canny edge detection algorithm to extr act the edges of the objects in the image. The Hough transform is then applied to the edge image to detect circles using. Act the edges of the objects in the image. The hough transform is then applied to the edge image to detect circles using the imfindcircles function. The detected circles are then displayed on the original image using the viscircles function. The imfindcircles function. The detected circles are then displayed on the original image using the viscircles function . The script uses a loop to perform the same operations on each of the three images. The resulting images are displayed in. The script uses a loop to perform the same operations on each of the three images. The resulting images are displayed in separate figures.

2.3 Part3 Optional.

2.3.1 original images, resulting images







2.3.2 code of the scripts

```
%%
%image3
figure
I = imread ('line02.jpg');
I = rgb2gray (I);
subplot(2,2,1);
imshow(I);
title('0riginal Image');
axis on,axis normal, hold on

%select the contours using the Canny algorithm
Iedge = edge(I,'Canny');
subplot(2,2,2);
imshow(~Iedge);
title('Applied Canny Algorithm');
axis on,axis normal, hold on
```

```
% apply the Hough transform using the hough()
[H,Theta,rho] = hough(Iedge);
subplot(2,2,3);
imshow(imadjust( rescale(H)),'YData',rho,'XData',Theta,'InitialMagnification','fit');
xlabel('irho'),ylabel('\Theta')
title('Hough Space');
axis on, axis normal, hold on
```

```
%calculate the peaks using the houghpeaks() function in the Hough space
peaks = houghpeaks(H,100,'threshold', ceil(0.5*max(H(:))));
x = Theta( peaks (:,2));
y = rho( peaks (:,1));
plot(x,y,'s','color','blue') ;
```

```
%Based on the peaks, we determine the straight lines using the houghlines()
lines = houghlines(Iedge,Theta,rho,peaks,'FillGap',5,'MinLength',10);
subplot(2,2,4);
imshow(I);
title('Result Image');
axis on,axis normal, hold on
```

```
minlen = 1000;
maxlen = 0;
for k = 1:length(lines )
    xy =[lines(k).point1; lines(k).point2] ;
    plot(xy(: ,1), xy(: ,2), 's ', 'color' , ' red ' ) ;
    plot(xy(: ,1),xy(: ,2),'Linewidth',2, 'color', 'green');

    len = norm(lines(k).point1 - lines(k).point2);
    if len < minlen
        minlen = len;
    end
    if len > maxlen
        maxlen = len;
    end
end
```

```
disp(['Minimum length: ',num2str(minlen)]);
disp(['Maximum length: ',num2str( maxlen)]);
disp(['Number of found lines:',num2str(length(lines))]);
```

```
%%
%Second Taskfigure
%image1
figure
Icircles = imread('circles01.jpg');
subplot(1,3,1);
imshow(Icircles)
title('Original Image');
axis on,axis normal, hold on
```

```
%select the contours using the Canny algorithm
```

```
IgrayCircles = im2gray(Icircles) ;
IedgeCircles = edge(IgrayCircles,'Canny') ;
subplot(1,3,2);
imshow (IedgeCircles)
title('Applied Canny Algorithm');
axis on,axis normal, hold on
```

```
%imfindcircles: Use the circle transform hough to find the circle
[centers,rad] = imfindcircles(IedgeCircles,[20 150], 'Sensitivity',0.9);
subplot(1,3,3);
imshow( Icircles)
title('Result Image');
axis on,axis normal, hold on
h = viscircles(centers,rad);%viscircles: Create a circle
```

```
%image2
figure
Icircles = imread('circles02.png');
subplot(1,3,1);
imshow(Icircles)
title('Original Image');
axis on,axis normal, hold on
```

```
IgrayCircles = im2gray(Icircles) ;
IedgeCircles = edge(IgrayCircles,'Canny') ;
subplot(1,3,2);
imshow (IedgeCircles)
title('Applied Canny Algorithm');
axis on,axis normal, hold on
```

```
[centers,rad] = imfindcircles(IedgeCircles,[20 150], 'Sensitivity',0.9);
subplot(1,3,3);
imshow( Icircles)
title('Result Image');
axis on,axis normal, hold on
h = viscircles(centers,rad);
```

```
%image3
figure
Icircles = imread('circles03.jpg');
subplot(1,3,1);
imshow(Icircles)
title('Original Image');
axis on,axis normal, hold on
```

```
IgrayCircles = im2gray(Icircles) ;
IedgeCircles = edge(IgrayCircles,'Canny') ;
subplot(1,3,2);
imshow (IedgeCircles)
title('Applied Canny Algorithm');
axis on,axis normal, hold on
```

```
[centers,rad] = imfindcircles(IedgeCircles,[20 150], 'Sensitivity',0.9);
subplot(1,3,3);
imshow( Icircles)
title('Result Image');
axis on,axis normal, hold on
h = viscircles(centers,rad);
```

2.3.3 comments

This code mainly uses the image processing toolbox in MATLAB to achieve two tasks. The first task is to detect straight lines in a given image, The process includes converting the color image into gray image, using Canny algorithm for edge detection, using Hough transform to detect the line, finding the peak value of the line through Hough peaks function, using Hough lines function to determine the start and end point of the line, and finally drawing a line in the original image. The second task is to detect the circle in the given image. The process includes converting the color image to gray image, using the Canny algorithm for edge detection, using the imfindcircles function for circle detection, and drawing the circle in the original image. In the second task, three different images were used respectively for detection, and the number and position of detected circles were output.

3.Conclusion

The given code performs two tasks: detecting straight lines and detecting circles in images. For the first task, the code reads in three images and applies the Canny edge detection algorithm to each image. It then applies the Hough transform to detect straight lines in each image. The Hough transform is a technique for detecting shapes in an image by mapping the image to a parameter space. In this case, the Hough transform is used to detect straight lines in the image. The code then calculates the minimum and maximum length of the detected lines and displays the number of lines found. Finally, the code displays the original image with the detected lines overlaid in green. For the second task, the code reads in three images and applies the Canny edge detection algorithm to each image. It then applies the circle transform Hough to detect circles in each image. The circle transform Hough is a technique for detecting circles in an image by mapping the image to a parameter space. The code then displays the original image with the detected circles overlaid in red. Overall, the code demonstrates how to use the Hough transform to detect shapes in an image, specifically straight lines and circles.

4. Answers to questions

1. What is the main principle of the Hough transform?

The main principle of the Hough transform is to identify shapes or patterns in an image by representing them as mathematical equations in parameter space. This allows powerful detection and recognition of shapes, even if they are partially obscured or distorted.

2. May the Hough transform be used to find arbitrary contours that cannot be described analytically?

No, the Hough transform is not designed to find arbitrary contours that cannot be described analytically. The Hough transform is a technique used to detect and extract features in an image that can be described by mathematical equations, so if Outlines cannot be analytically described, the Hough transform may not be applicable.

3. What are the recurrent and generalized Hough transforms?

Cyclic Hough transform : RHT is an extension of the standard Hough transform to detect and track multiple instances of a feature in an image. It uses a recursive algorithm based on the original Hough transform to iteratively search for other features in the image.

Generalized Hough transform : GHT is a more flexible version of the standard Hough transform that can detect and extract features that cannot be described by simple mathematical equations. GHT works by creating templates of the features to be examined and then searching for the best match between the template and the image. This allows GHT detection of features that have complex shapes or are not easily described by mathematical equations.

4. What are the ways of parametrization in the Hough transform?

Standard, polar coordinates and generalized Hough transformation.