Lab Report 8

Aim

Image segmentation using point segmentation, line segmentation and edge detection.

Theory

Image segmentation is the process of dividing an image into multiple segments or regions, each of which corresponds to a different object or part of the image. Point segmentation, line segmentation, and edge detection are all techniques that can be used to perform image segmentation.

Point segmentation involves identifying individual points or pixels within an image that belong to a particular segment or object. This can be achieved using techniques such as clustering or thresholding, where pixels with similar characteristics are grouped together.

Line segmentation involves identifying linear features within an image, such as edges or contours, and using these features to separate the image into different segments. This can be achieved using techniques such as the Hough transform, which can detect lines or curves within an image.

Edge detection involves identifying abrupt changes in brightness or color within an image, which can be used to locate the boundaries between different objects or segments. This can be achieved using techniques such as the Sobel operator, which highlights areas of the image with high spatial gradients.

All three of these techniques can be used in combination to perform image segmentation. For example, edge detection can be used to identify the boundaries between different segments, while point segmentation and line segmentation can be used to group pixels or linear features within each segment. The choice of technique will depend on the specific requirements of the image segmentation task, as well as the characteristics of the image being segmented.

CODE

```
% POINT SEGMENTATION

fprintf('please Select an image');
y=uigetfile('*.*');
i=imread(y);
k=rgb2gray(i);
d = padarray(k,[1 1],0,'both');
[r,c]=size(d);
s=zeros(r+2,c+2)
```

```
for R =2:(r-1)
   for C=2:(c-1)
      s(R,C)= d(R+1,C)+d(R-1,C)+d(R,C+1)+d(R,C-1)-4*d(R,C);
%         if value>=100
%             d(R,C)=1;
%         else
%             d(R,C)=0;
%         end
   end
end

subplot(1,2,1);imshow(k);title('Original Image');
subplot(1,2,2);imshow(s);title('detected');
```

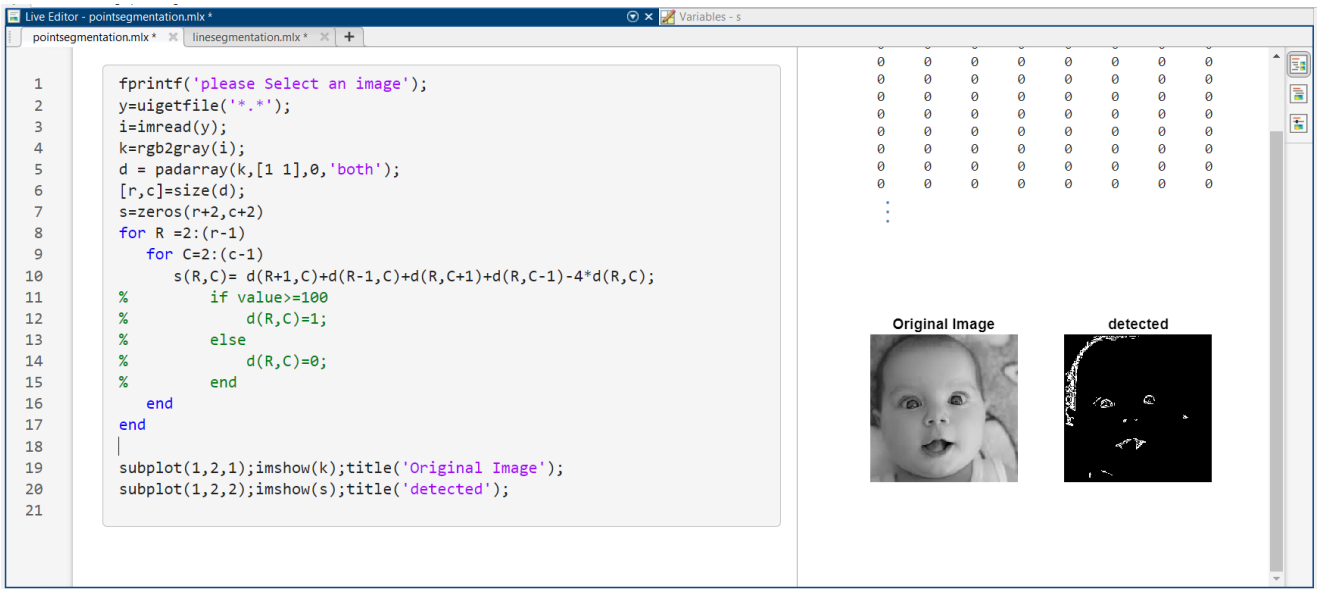

Figure 1: use of point segmentation to detect points in image in MATLAB

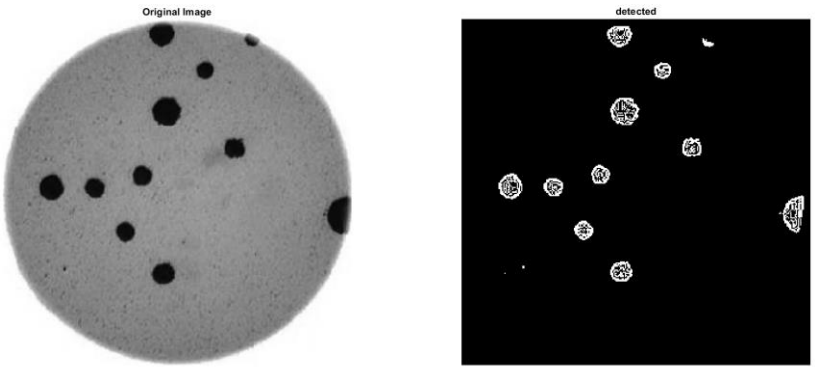

Figure 2: point detection using point segmentation in MATLAB

CODE

```
% LINE SEGMENTATION
fprintf('please Select an image');
y=uigetfile('*.*');
i=imread(y);
k=double(rgb2gray(i));
d = padarray(k,[1 1],0,'both');
[r,c]=size(d);
hori=zeros(r+2,c+2)
verti=zeros(r+2,c+2)
diag1=zeros(r+2,c+2)
diag2=zeros(r+2,c+2)
hor=[-1 -1 -1 ; 2 2 2 ; -1 -1 -1 ] % horizontal filter
ver=[-1 2 -1 ; -1 2 -1 ; -1 2 -1 ] % Vertical filter
dia1=[2 -1 -1 ; -1 2 -1 ; -1 -1 2 ] % diagonal 45 degree filter
dia2=[-1 -1 2 ; -1 2 -1 ; 2 -1 -1 ] % diagonal 135 degree filter
for x =2:(r-1)
   for y=2:(c-1)
      kernel= [d(x-1,y-1) d(x-1,y) d(x-1,y+1) ; d(x,y-1) d(x,y)
d(x,y+1) ; d(x+1,y-1) d(x+1,y) d(x+1,y+1)];
      h=sum(sum(hor.*kernel));
      v=sum(sum(ver.*kernel));
      d1=sum(sum(dia1.*kernel));
      d2=sum(sum(dia2.*kernel));
      hori(x,y)=h;
      verti(x,y)=v;
      diag1(x,y)=d1;
      diag2(x,y)=d2;
   end
end
all=hori+verti+diag1+diag2;
subplot(2,3,1);imshow(i);title('Original Image');
subplot(2,3,2);imshow(hori);title('horizontal lines');
subplot(2,3,3);imshow(verti);title('vertical lines');
subplot(2,3,4);imshow(diag1);title('diagonal line 45');
subplot(2,3,5);imshow(diag2);title('diagonal line 135');
subplot(2,3,6);imshow(all);title('all line in one');
```

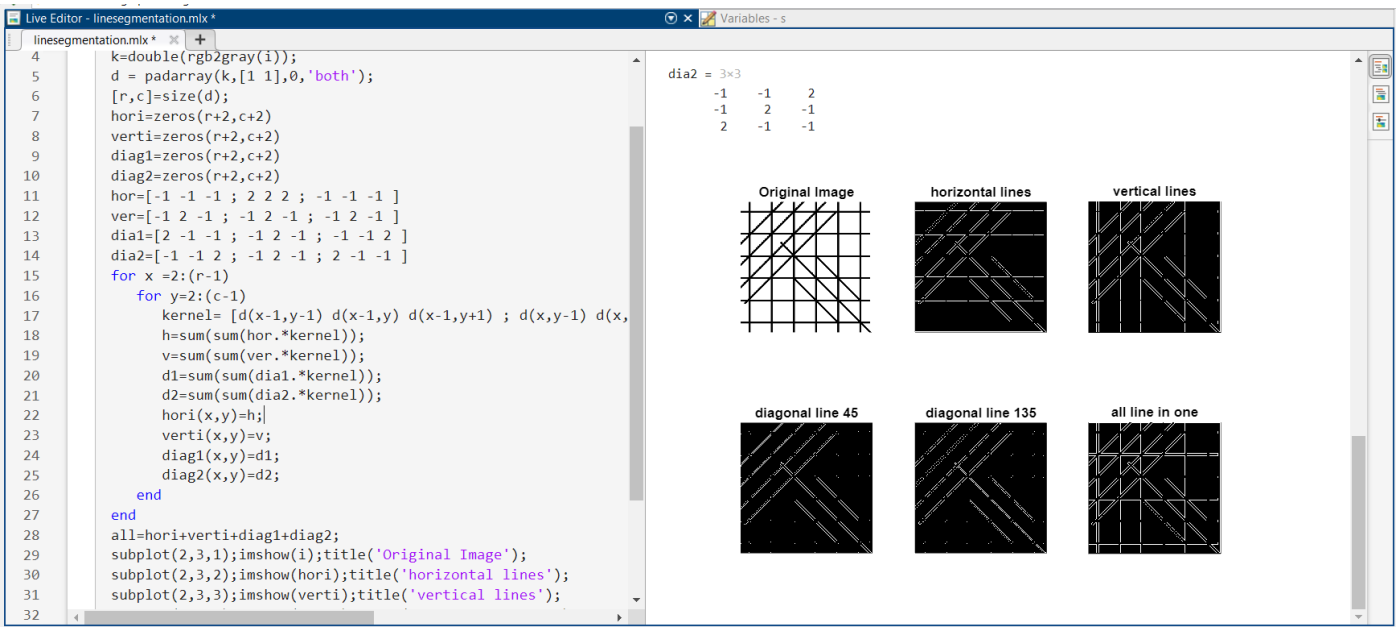

Figure 3: use of line segmentation to detect lines in MATLAB

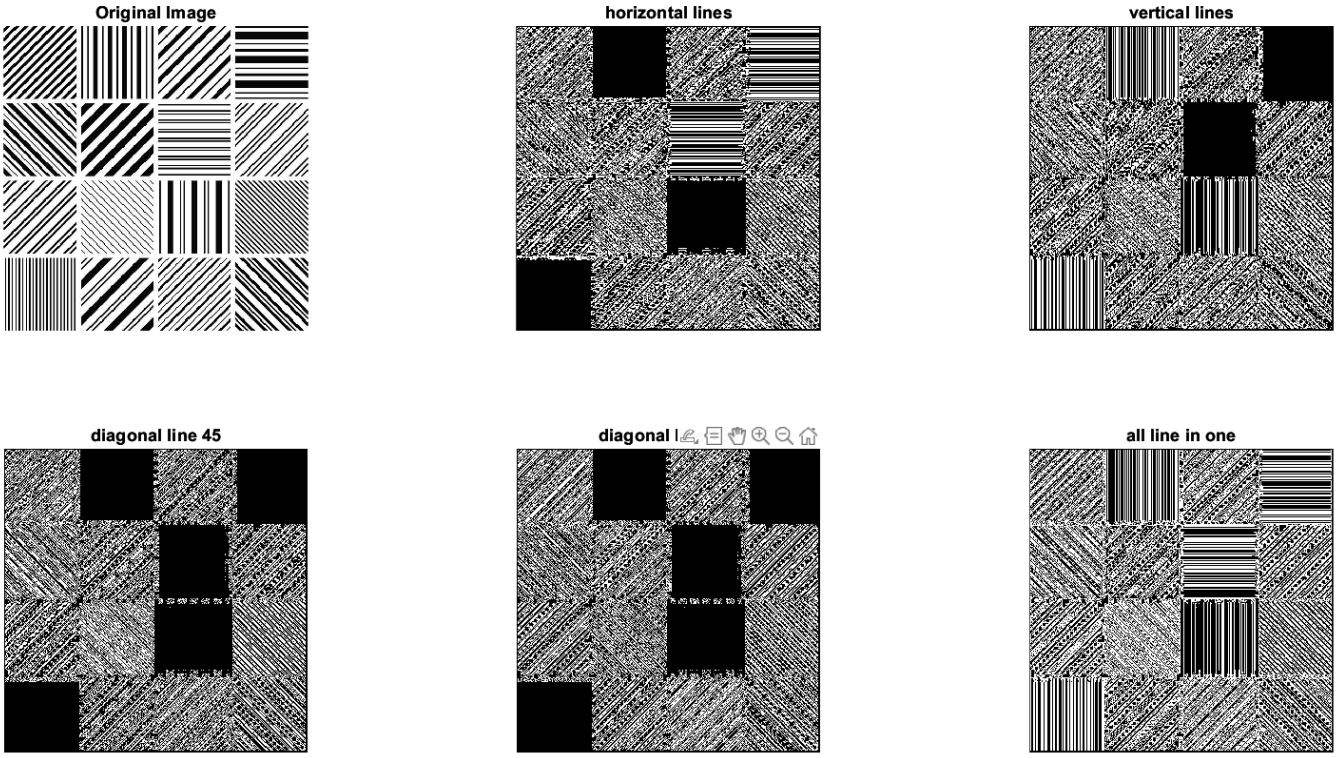

Figure 4: detection of various types of lines in MATLAB

EDGE DETECTION

Edge detection is a computer vision technique that involves identifying the boundaries of objects within an image. It is a crucial step in many image processing applications, such as object recognition, image segmentation, and feature extraction. Edge detection algorithms typically work by analyzing the variations in

brightness, color, or texture across an image. The most commonly used edge detection algorithms include the Sobel, Prewitt, and Canny edge detection algorithms. The Sobel and Prewitt algorithms use convolution filters to identify edges based on the intensity gradients in the image. The Canny algorithm, on the other hand, uses a more sophisticated approach that involves detecting edges at multiple scales and suppressing false positives. Once the edges have been detected, they can be further processed to extract useful information, such as the shape and size of objects in the image. This information can then be used for a wide range of applications, such as object tracking, face recognition, and autonomous driving.

CODE

```
% EDGE DETECTION
fprintf('please Select an image');
y=uigetfile('*.*');
i=imread(y);
J=rgb2gray(i);



K=padarray(J,[1,1],0);
K=double(K);
[rows,columns]=size(K);
L=zeros(rows,columns);
gx=[-1 -1 -1;0 0 0;1 1 1];
gy=[-1 0 1;-1 0 1;-1 0 1];
s1=[0 0 0;0 0 0;0 0 0];
s3=[0 0 0;0 0 0;0 0 0];
for r=2:rows-1
    for c=2:columns-1
      kernel=[K(r-1,c-1) K(r-1,c) K(r-1,c+1); K(r,c-1) K(r,c)
K(r,c+1); K(r+1,c-1) K(r+1,c) K(r+1,c+1)];
      s1=kernel.*gx;
      s2=sum(s1,"all");
      s3=kernel.*gy;
      s4=sum(s3,"all");
      L(r,c)=sqrt(s2.^2+s4.^2);
    end
end
subplot(1,2,1);imshow(i);title("Original image");
subplot(1,2,2);imshow(uint8(L),[]);title('edges detected');
```

```
fprintf('please Select an image');
y=uigetfile('*.*');
i=imread(y);
J=rgb2gray(i);


K=padarray(J,[1,1],0);
K=double(K);
[rows,columns]=size(K);
L=zeros(rows,columns);
gx=[-1 -1 -1;0 0 0;1 1 1];
gy=[-1 0 1;-1 0 1;-1 0 1];
s1=[0 0 0;0 0 0;0 0 0];
s3=[0 0 0;0 0 0;0 0 0];
for r=2:rows-1
    for c=2:columns-1
      kernel=[K(r-1,c-1) K(r-1,c) K(r-1,c+1); K(r,c-1) K(r,c) K(r,c+1); K(r+
      s1=kernel.*gx;
      s2=sum(s1,"all");
      s3=kernel.*gy;
      s4=sum(s3,"all");
      L(r,c)=sqrt(s2.^2+s4.^2);
    end
end
subplot(1,2,1);imshow(i);title("Original image");
subplot(1,2,2);imshow(uint8(L),[]);title('edges detected')
```

```
please Select an image
```

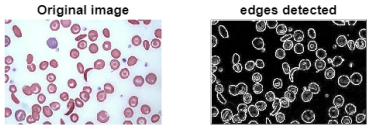

Figure 5: use of edge detection code to detect edges in MATLAB

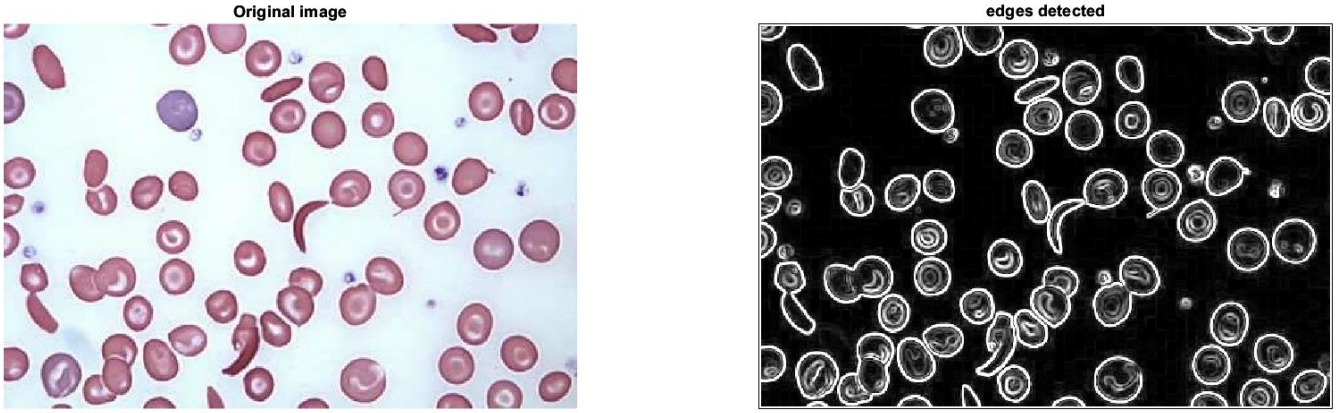

Figure 6 edge detection *in MATLAB*

Shreenandan Sahu |120BM0806