Experimental Code and analysis:

1.1 Initialization to write code on MATLAB

clc;	Clear command window
close all;	Close one or more figures
clear all;	Remove items from workspace, freeing up system memory

1.2 Displaying image

image = imread('im1.jpg'),	Read image from graphics file
figure (1), subplot(3,3,1),	
imshow(image),	Display image
title ('Original image');	
image=rgb2gray(image);	Convert RGB image or colormap to grayscale

1.3 Addition of noise

n1 = imnoise(image,'salt & pepper',0.2);
subplot(3,3,2), imshow(n1),
title('Salt & pepper noise');
	Addition of Salt and pepper noise

n1 = imnoise(image,'gaussian',0.2,0.05);
subplot(3,3,2), imshow(n1),
title('Gaussian noise');
	Addition of gaussian noise

n1 = imnoise(image,'poisson');
subplot(3,3,2), imshow(n1),
title('Poisson noise');
	Addition of poisson noise

n1 = imnoise(image,'speckle',0.2);
subplot(3,3,2), imshow(n1),
title('Speckle noise');
	Addition of speckle noise

1.4 Analysis SNR by using different filters on a noisy image

f1=fspecial('average',3) cf1=imfilter(n1,f1) subplot(3,3,4), imshow(cf1),	Average filter
f2=fspecial('disk',10) cf2=imfilter(n1,f2) subplot(3,3,5), imshow(cf2),	Disk filter
f3=fspecial('motion',20,45) cf3=imfilter(n1,f3) subplot(3,3,6), imshow(cf3), cf3=double(cf3(:));	Motion filter
f4= fspecial('laplacian',0.2) cf4=imfilter(n1,f4) subplot(3,3,7), imshow(cf4),	Gaussian filter
f5=fspecial('log',3,0.5)	Log filter

cf5=imfilter(n1,f5) subplot(3,3,8), imshow(cf5),	
f6= fspecial('gaussian',3,0.5) cf6=imfilter(n1,f6) subplot(3,3,7), imshow(cf6),	Laplacian filter

cf6=double(cf6(:));
imm=mean(cf6(:));
ims=std(cf6(:));
b6 = 10*log((imm)./ims)
title(['Laplacian filter = ' num2str(round(b6,2))]);
	SNR calculation

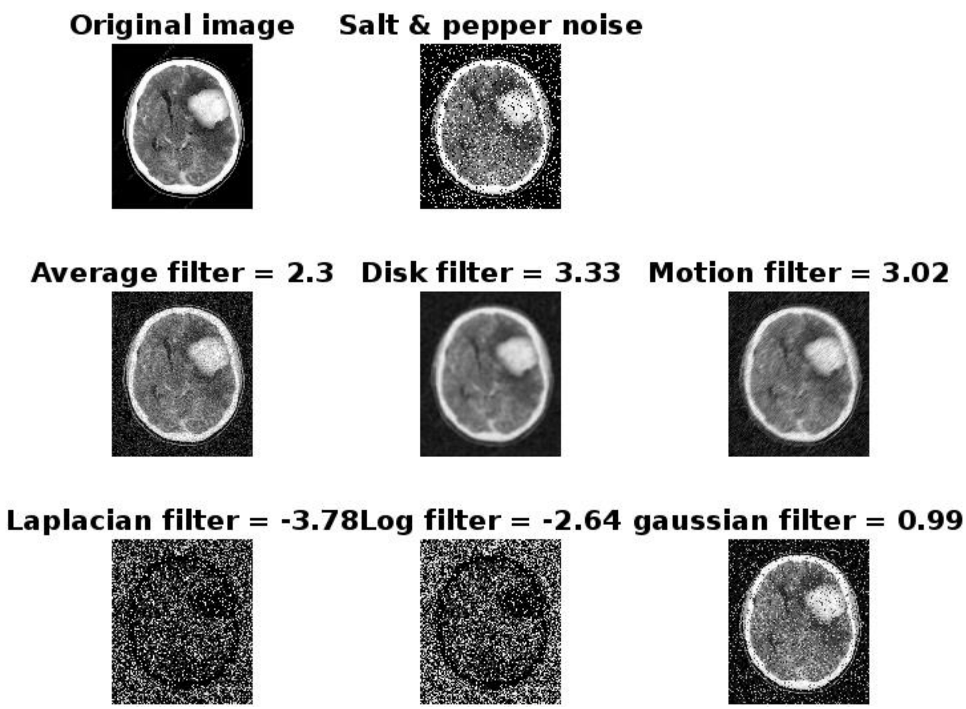

Figure 1.4.1 Analysis SNR by using different filters on an image with salt and pepper noise

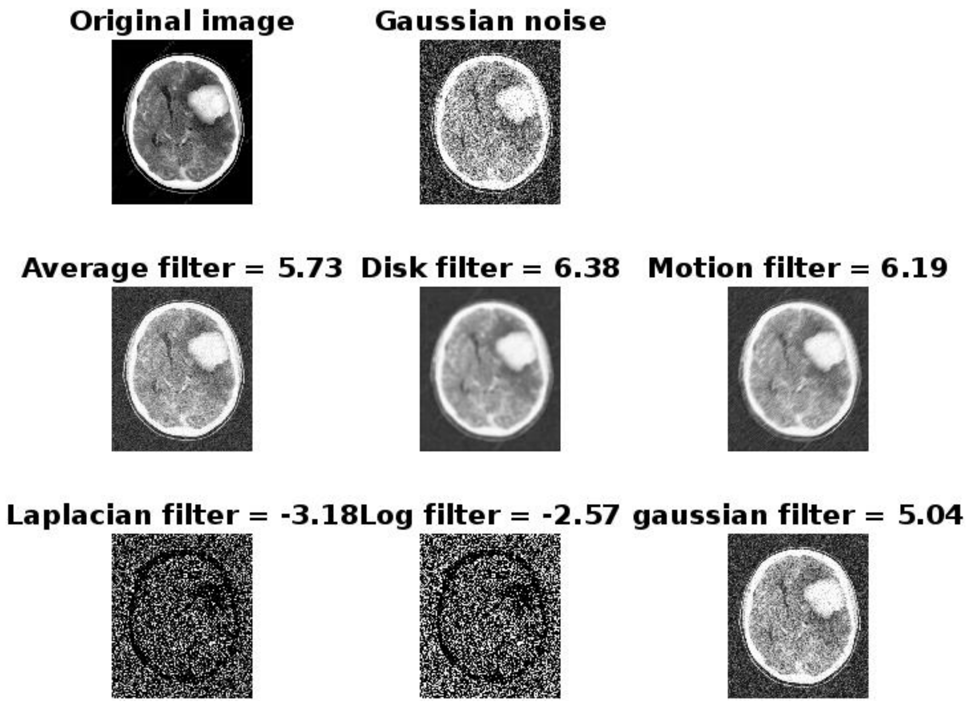

Figure 1.4.2 Analysis SNR by using different filters on an image with gaussian noise

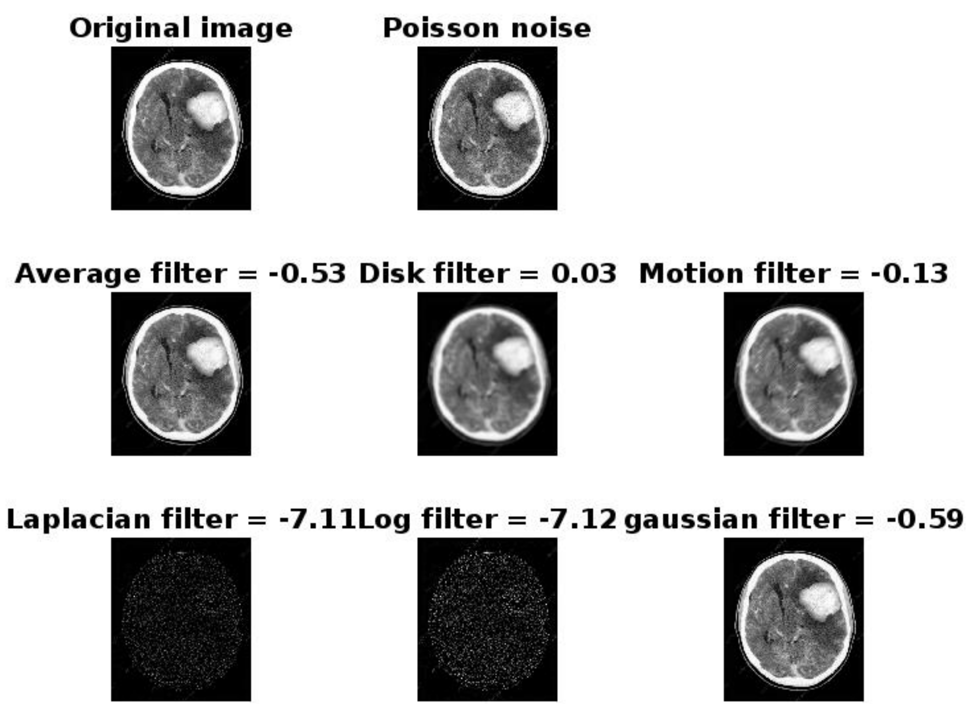

Figure 1.4.3 Analysis SNR by using different filters on an image with Poisson noise

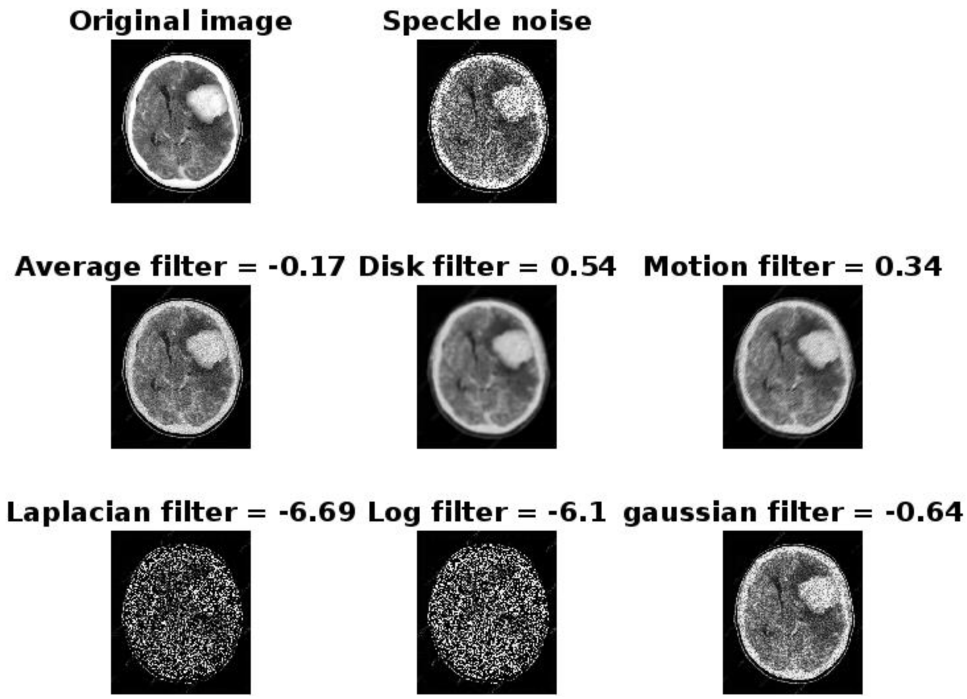

Figure 1.4.4 Analysis SNR by using different filters on an image with Speckle noise

1.5 Detection of edge in image

image = imread('im2.jpg'), image=rgb2gray(image);	
[BW,threshOut] = edge(image)	It returns the threshold value
edge_prewitt=edge(image,'prewitt',threshOut) edge_sobel=edge(image,'sobel',threshOut) edge_roberts=edge(image,'roberts',threshOut) edge_canny=edge(image,'canny',threshOut)	It returns all edges that are stronger than threshold.

figure(1), imshow(image), title('Original image');
figure(2), imshow(edge_prewitt), title('Edge detection using prewitt operator');
figure(3), imshow(edge_sobel), title('Edge detection using sobel operator');
figure(4), imshow(edge_roberts), title('Edge detection using roberts operator');
figure(5), imshow(edge_canny), title('Canny edge detection');	

Figure 1.5.1 Original Image

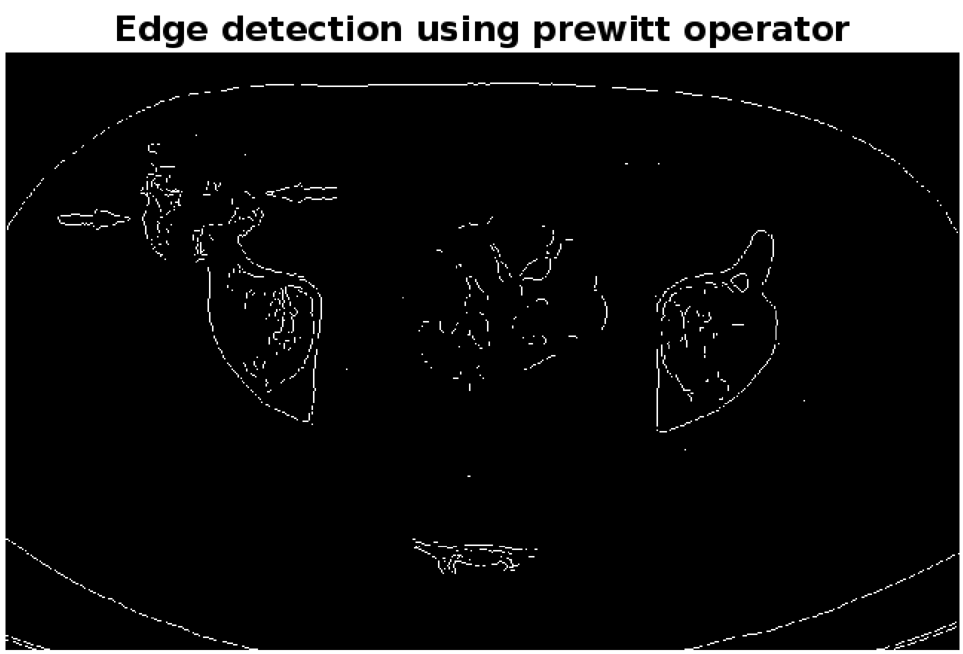

Figure 1.5.2 Edge detection using Prewitt operator

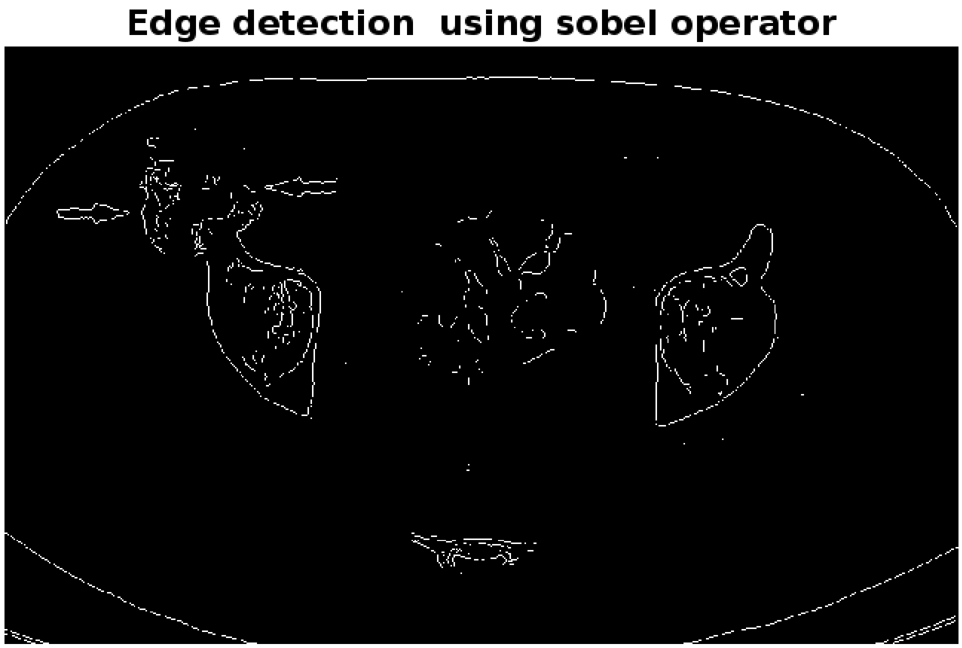

Figure 1.5.3 Edge detection using Sobel operator

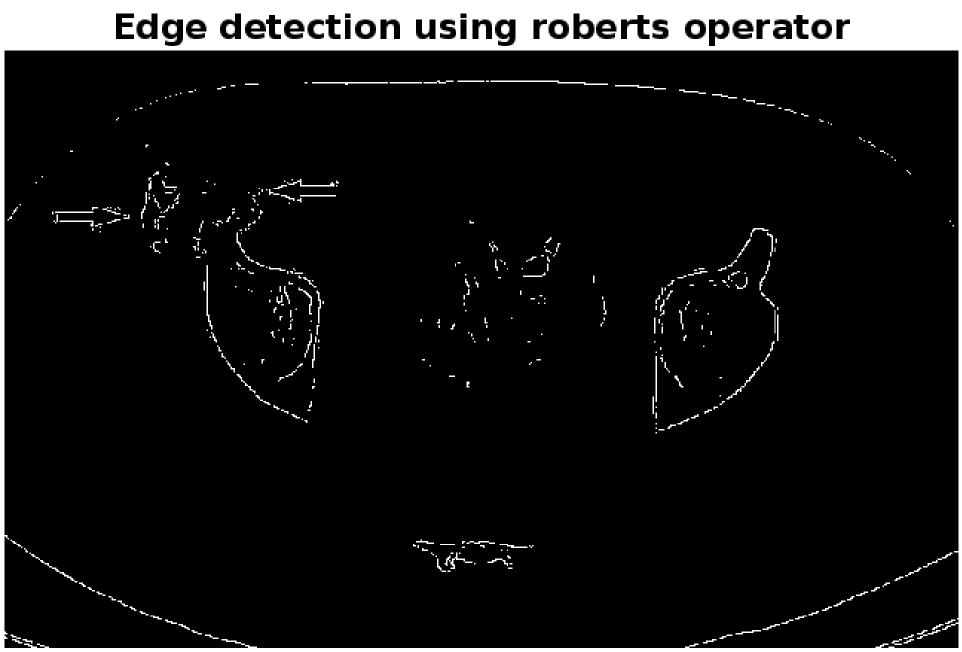

Figure 1.5.4 Edge detection using Roberts operator

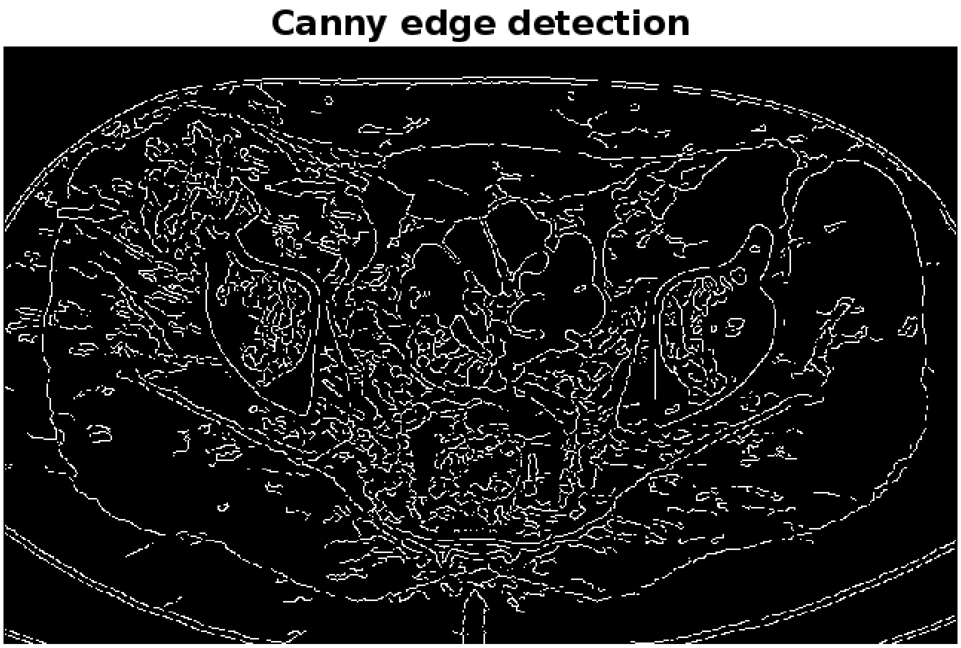

Figure 1.5.5 Canny edge detection

1.6 Image segmentation using morphological operations

k=imread("im4.png");
figure(1),
subplot(4,2,1),imshow(k),title('Original Image');	

k1=im2bw(k,graythresh(k)); subplot(4,2,2), imshow(k1); title('Binary image');	It converts image to binary image, based on threshold
SE=strel('disk',7,4); k2=imopen(k1,SE); subplot(4,2,3), imshow(k2), title('Morphologically open image'),	It converts small white portions or a bunch of bright pixels into dark portions without changing the size of a larger dark portion.
b=bwlabel(k2); subplot(4,2,4), imshow(b), title('Labelled image')	Label connected components in 2-D binary image
b1=label2rgb(b,'hot','w') subplot(4,2,5), imshow(b1), title('Heatmap labelled image');	It specifies the colormap cmap to be used in the RGB image and the RGB color of the background elements as white.
b(b~=7)=0 b(b==7)=1 subplot(4,2,6), imshow(b), title('Biggest connected component');	brain is component labeled as 7. Set all other component as 0 except brain.
k3=imclose(b,strel('disk',18)); subplot(4,2,7), imshow(k3), title('MOrphologically close image');	It converts small black portions or a bunch of dark pixels into bright portions without changing the size of the larger white portion.
x=im2double(k) k4=k3.*x subplot(4,2,8), imshow(k4), title('Morphologically segmented image');	Extract the brain image from original image

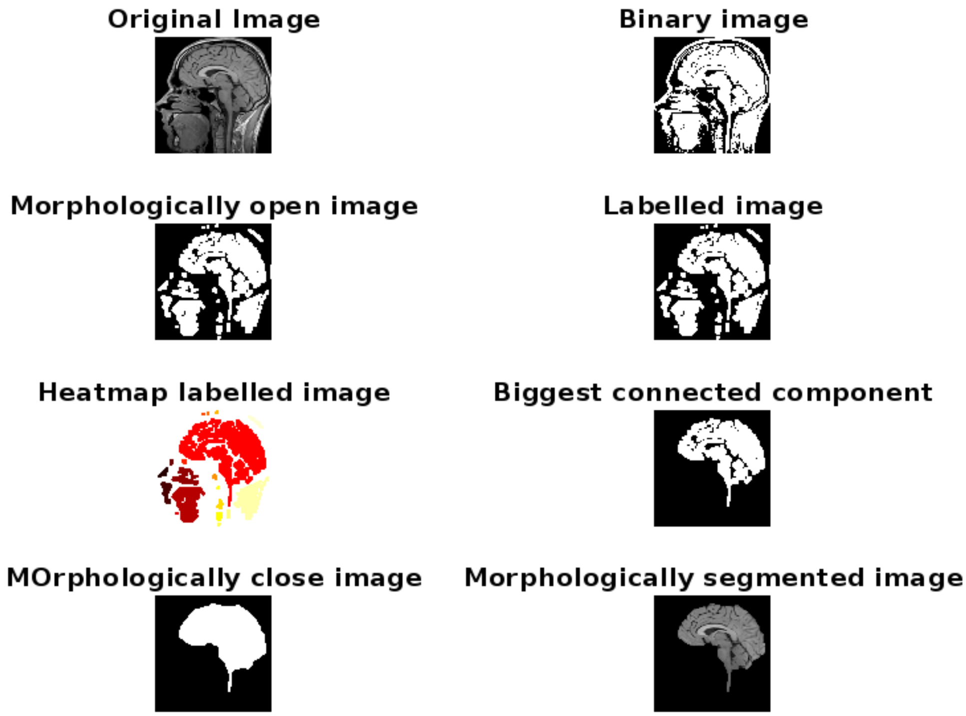

Figure 1.6.1 Brain extracted from the original image using morphological operation

1.7 Image segmentation using K-means cluster

```
a=rgb2gray(imread('im1.jpg'))
 imData = reshape(a,[],1);
imData = double(imData);
[IDX nn]=kmeans(imData,4);
imIDX=reshape(IDX,size(a));
figure(1),
subplot(2,2,1),imshow(imIDX==1,[]),  title('Observation 1');
subplot(2,2,2),imshow(imIDX==2,[]),  title('Observation 2');
subplot(2,2,3),imshow(imIDX==3,[]),  title('Observation 3');
subplot(2,2,4),imshow(imIDX==4,[]),  title('Observation 4');
```

```
bw=(imIDX==2);
se=ones(5);
bw=imopen(bw,se);
bw=bwareaopen(bw,100);
bw=~bw
figure(2),imshow(bw), title('K means clustering');
```

```
mask = false(size(a));
mask(25:end-25,25:end-25) = true;
a_new = activecontour(a, mask, 300);
similarity = dice(a_new, bw);
figure(3)
imshowpair(a_new, bw)
title(['Dice Index = ' num2str(similarity)])
```

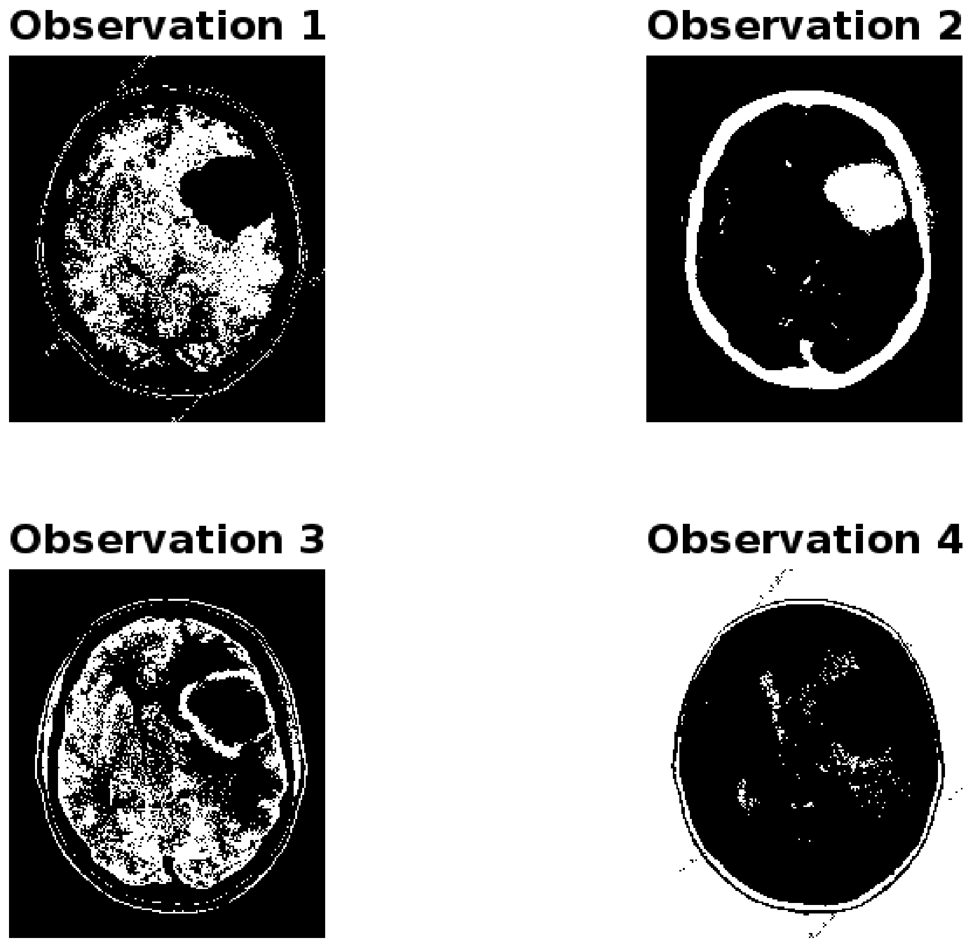

Figure 1.7.1 Observative steps for K-means clustering

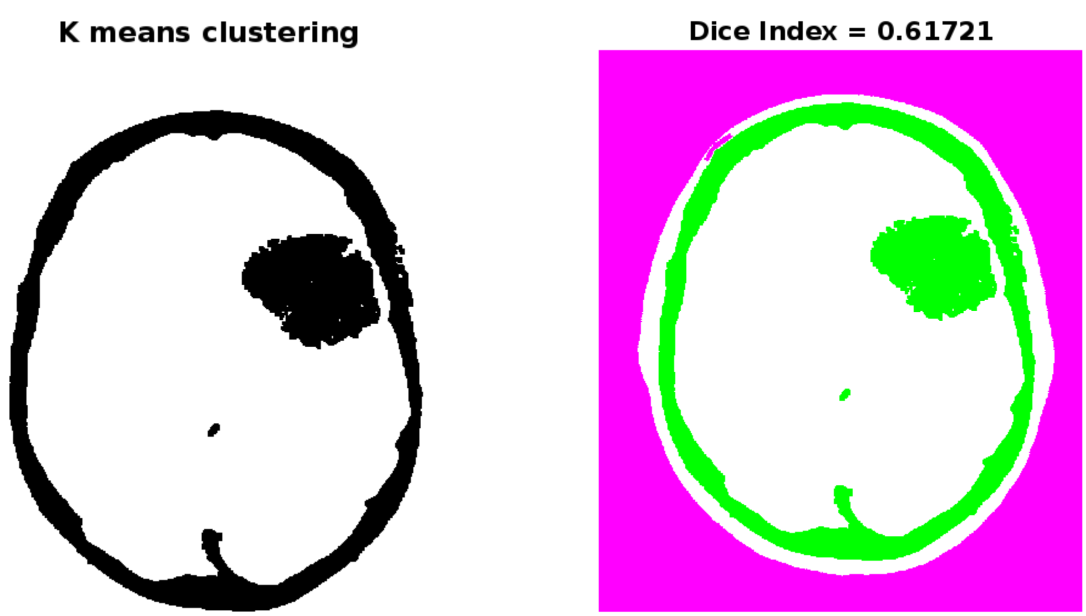

Figure 1.7.2 Dice score of the segmented image for K-means clustering

1.8 Image segmentation using watershed algorithm

img=imread("im1.jpg") img=rgb2gray(img) figure(1), subplot(3,2,1), imshow(img), title('Original image');	

se = strel('disk',20);
tophatFiltered = imerode(img,se);
subplot(3,2,2),
imshow(tophatFiltered) , title('Eroded image');
	Create the structuring element and perform erosion operation and display the image

c ontrastAdjusted = imadjust(tophatFiltered);
subplot(3,2,3),
imshow(contrastAdjusted) , title(' Contrast adjusted image');
	Use imadjust to improve the visibility of the result

level=0.6 bw = im2bw(contrastAdjusted,level) subplot(3,2,4), imshow(bw), title('Binary image');	Select the threshold level

c=~bw
subplot(3,2,5),
imshow(c),title('Altered image');
d=-bwdist(c)
d(c)=-Inf
l=watershed(d)
wi=label2rgb(l,'hot','w')
subplot(3,2,6),
imshow(wi), title('Heatmap labeled watershed');	

im=img im(l==0)=0; figure(2), imshow(im),title('Image segmented using watershed algorithm');	

mask = false(size(img));
mask(25:end-25,25:end-25) = true;
BW = activecontour(img, mask, 300);
im=imbinarize(im);
similarity = dice(BW, im);
figure(3)
imshowpair(BW, im)
title(['Dice Index = ' num2str(similarity)])	

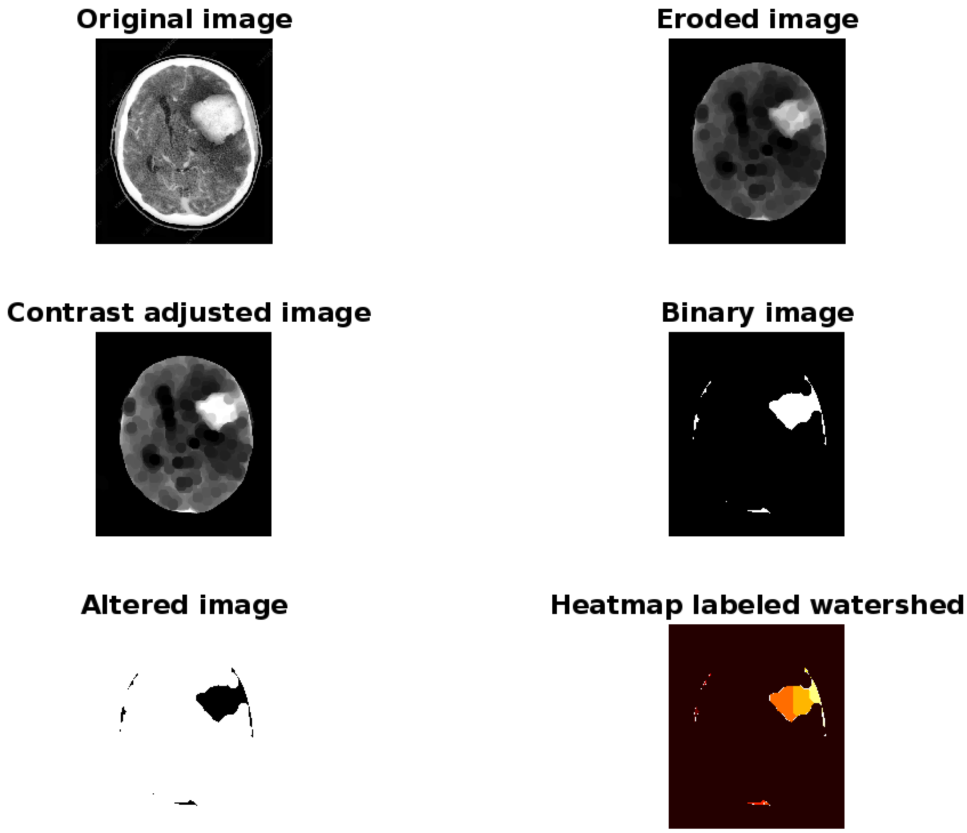

Figure 1.8.1 Observative steps for watershed algorithm

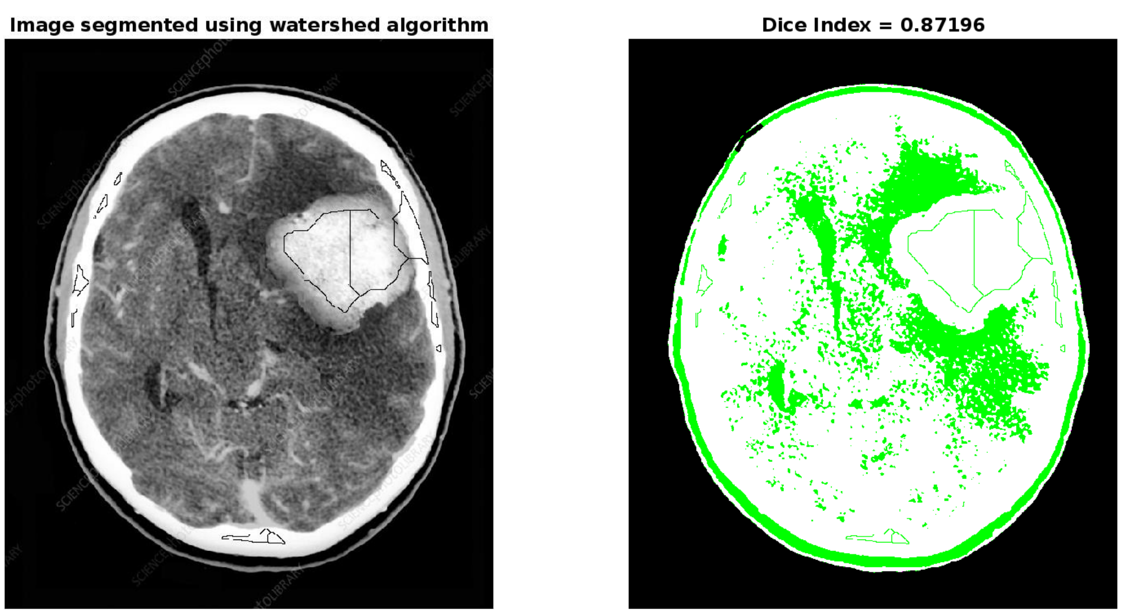

Figure 1.8.2 Dice score of the segmented image for watershed algorithm