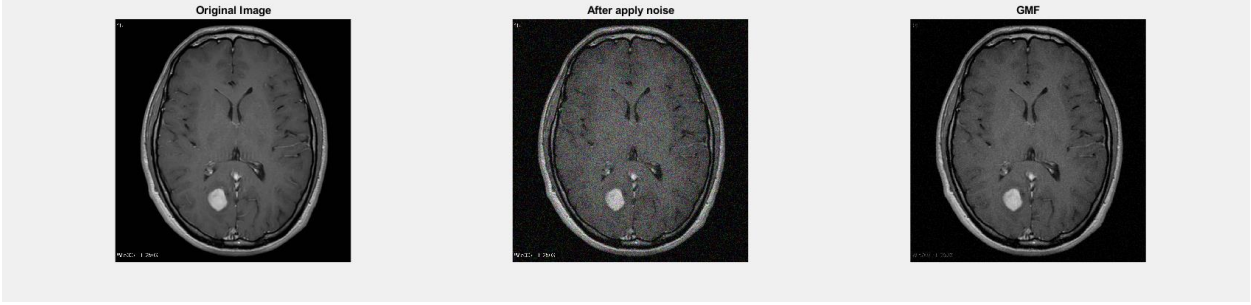
1. Apply Gaussian noise to Figure 1, and then use the following to restore the image:

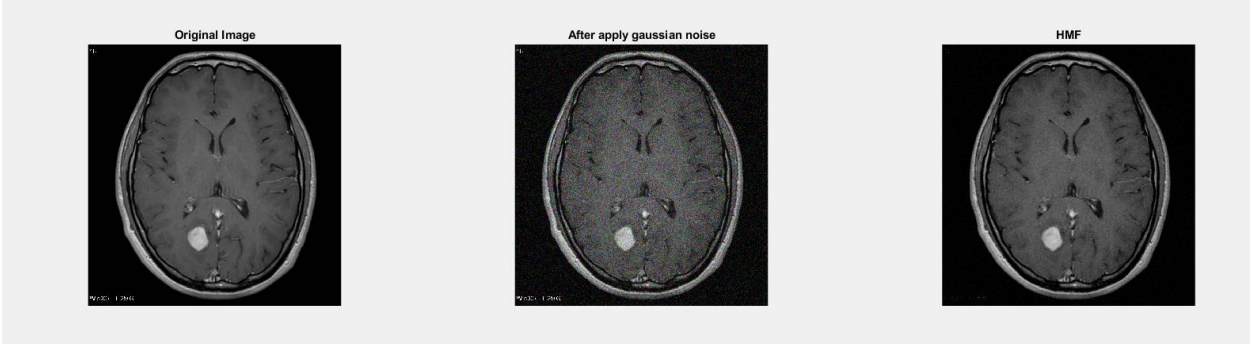
Geometric Mean filter

```
%% i. Geometric Mean filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
n = 3;
F = exp(imfilter(log(I), ones(n, n), 'replicate')).^(1/(n*n));
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply noise');
subplot(2, 3, 3), imshow(F), title('GMF');
```



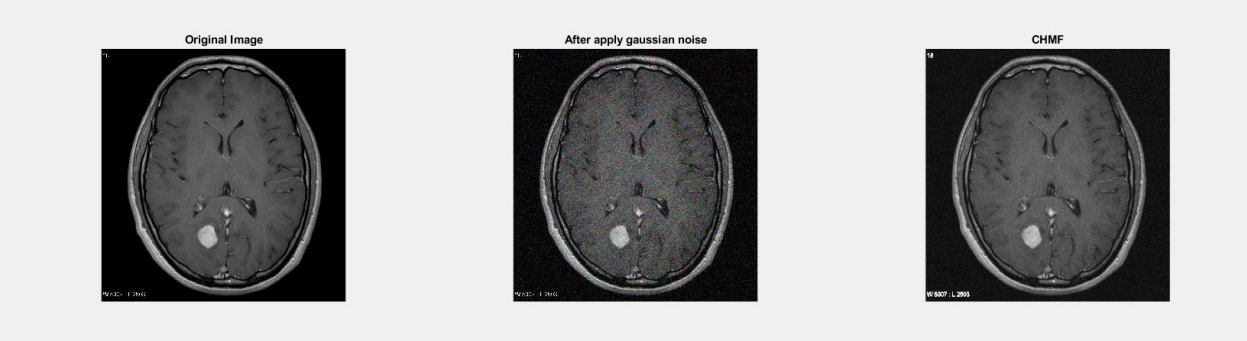
Harmonic Mean filter

```
%% ii. Harmonic Mean filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
n = 3;
SF = (n*n)./imfilter(1./(I+eps), ones(n, n),'replicate');
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(SF), title('HMF');
```



Contra-harmonic Mean filter

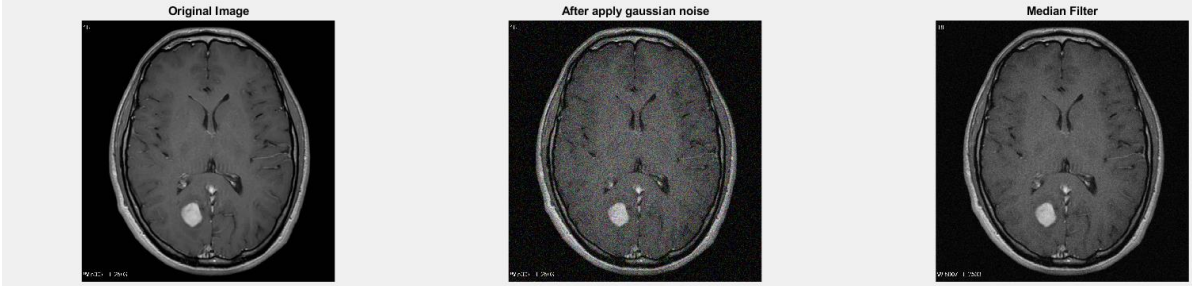
```
%% iii. Contra-harmonic Mean filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
n = 3;
order = 1;
SF = imfilter(I.^(order+1), ones(n, n),'replicate')./(imfilter(I.^(order),
ones(n, n),'replicate')+eps);
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(SF), title('CHMF');
```



2. Apply Gaussian noise to Figure 1, and then use the following order statistic filters to restore the image:

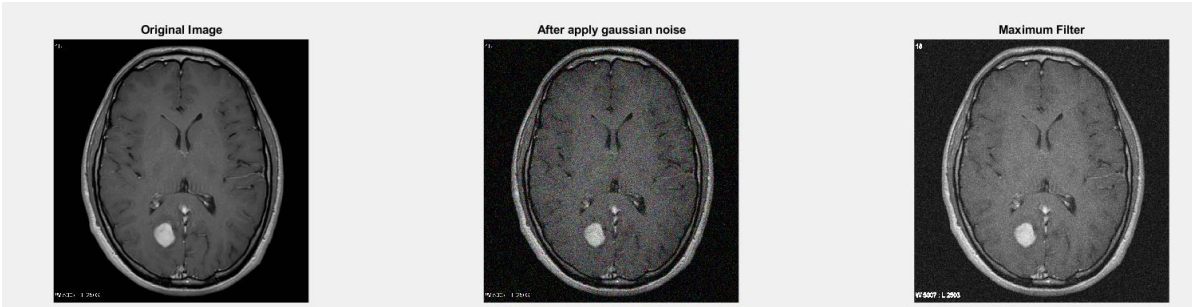
Median filter

```
%% i. Median filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
B = ordfilt2(I,5,ones(3,3));
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(B), title('Median Filter');
```



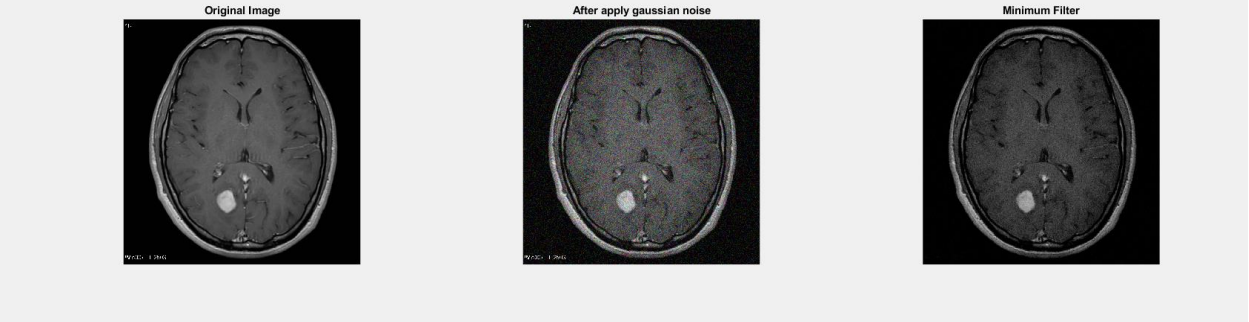
Maximum filter

```
%% ii. Maximum filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
B = ordfilt2(I,9,ones(3,3));
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(B), title('Maximum Filter');
```



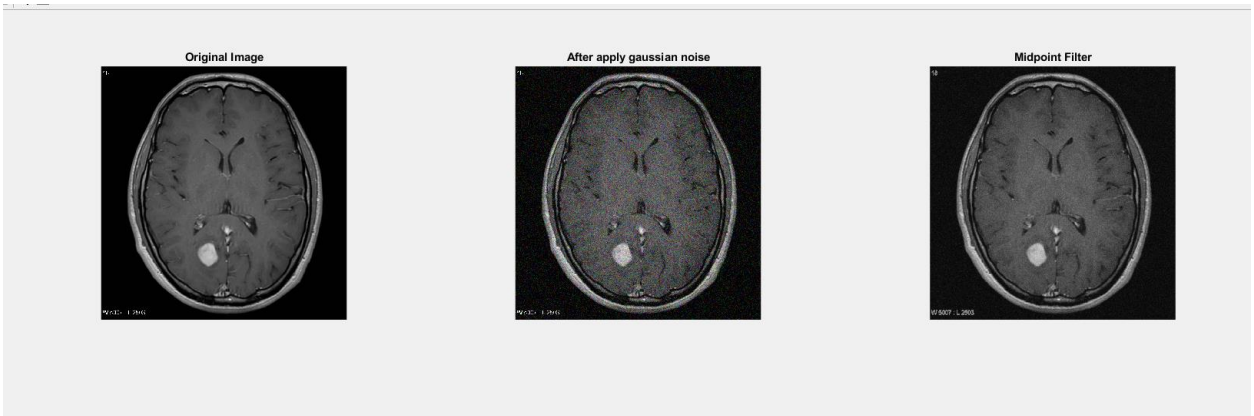
Minimum filter

```
%% iii. Minimum filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
B = ordfilt2(I,1,ones(3,3));
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(B), title('Minimum Filter');
```



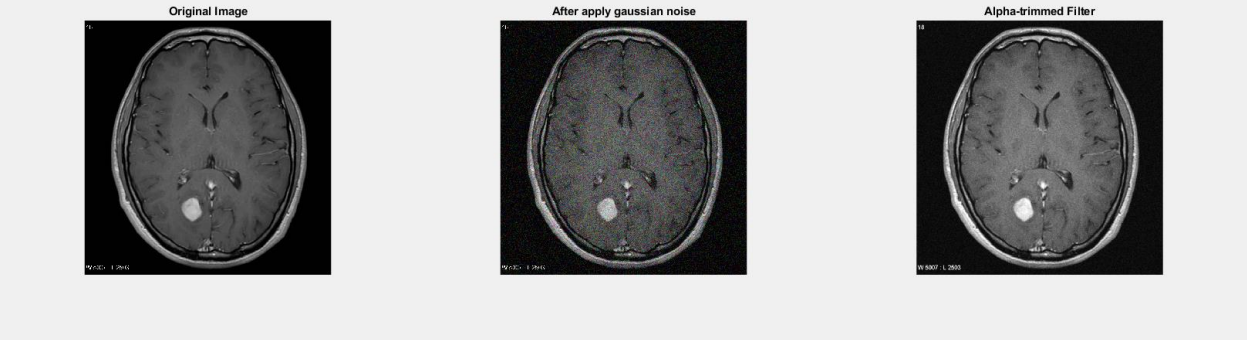
Midpoint filter

```
%% iv. Midpoint filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
min = ordfilt2(I,1,ones(3,3));
max = ordfilt2(I,9,ones(3,3));
B = imlincomb(0.5,min,0.5,max);
midpointimage = B;
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(B), title('Midpoint Filter');
```



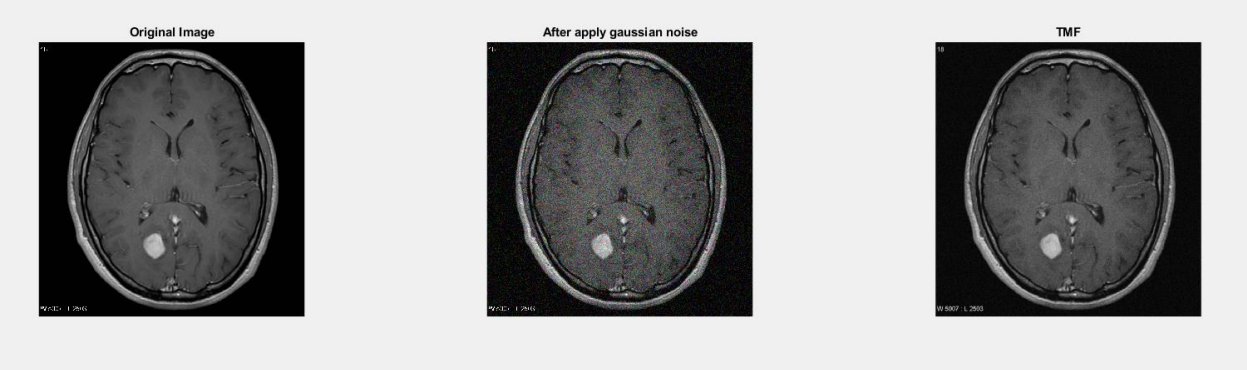
Alpha-trimmed filter

```
%% v. Alpha-trimmed filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
n = 3;
d = 2;
w = ones(n, n)/((n*n)-d);
B = imfilter(I, w,'replicate', 'same');
subplot(2, 3, 1), imshow(img), title('Original Image');
subplot(2, 3, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 3, 3), imshow(B), title('Alpha-trimmed Filter');
```



Trimmed filter

```
%% vi. Trimmed mean filter
img = imread('Picture1.png');
j = imnoise(img, "gaussian");
I = rgb2gray(j);
I = im2double(I);
n = 3;
w = ones(n, n)/(n*n);
SF = imfilter(I, w, 'replicate')+eps;
subplot(2, 2, 1), imshow(img), title('Original Image');
subplot(2, 2, 2), imshow(j), title('After apply gaussian noise');
subplot(2, 2, 3), imshow(SF), title('TMF');
```



3. By observing and comparing each of the outputs, determine which filter restores the image closest to its original state. Mention the reasoning behind your observation and choose the most suitable image for the following problems.

   Answer:
   We can see the following after applying each of the order statistic filters to the noisy image:

   Median filter: The image retains its edges, although it looks significantly blurrier than the original.

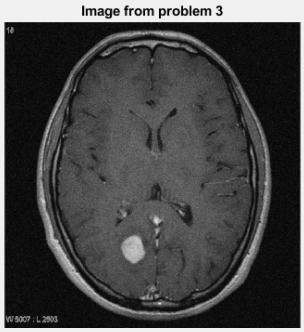
Maximum filter: The image seems brighter and some of the finer details are lost, but the edges are kept.
Minimum filter: The image seems darker and some of the finer details are lost, but the edges are intact.
Midpoint filter: In comparison to the Maximum and Minimum filters, the edges are kept, and the brightness and details are more evenly distributed. However, in contrast to the original, the image seems a little bit blurry.
Midpoint filter: In comparison to the Maximum and Minimum filters, the edges are kept, and the brightness and details are more evenly distributed. However, in contrast to the original, the image seems a little bit blurry.
Alpha-trimmed filter: The image seems sharper than the Median filter and the edges are retained. The image appears a little bit darker than the original and some of the finer details are gone.
Trimmed filter: The image is sharper than the Median filter and the margins are maintained. The image appears a little bit darker than the original and some of the finer details are gone.

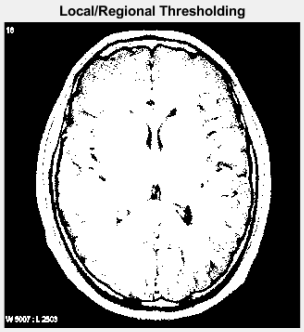
These findings lead us to the conclusion that the Midpoint filter recovers the image the most closely to its initial state. Although it might not be the sharpest or most detailed filter, it offers a nice blend of edges, brightness, and details preservation. As a result, the image that has been filtered at the Midpoint will be the one we choose for further processing.

4. Detect the tumor from the image from Problem 3 using the segmentation approaches listed below:
   (***Outline the segmented object to highlight the tumor***. *You can crop the image for accurate segmentation.*)
   i) Similarity approaches:

Local/Regional Thresholding

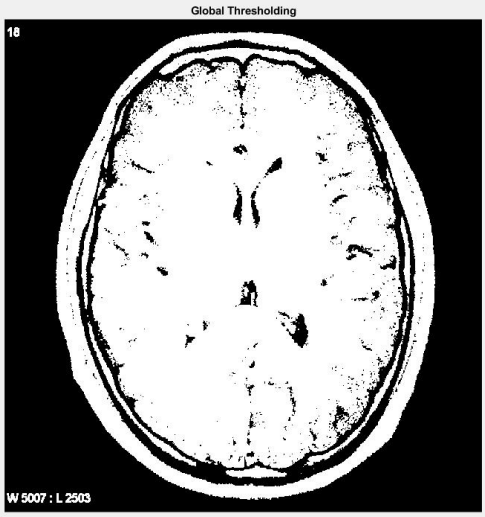
```
I = midpointimage;
T = graythresh(I);
a = im2bw(I, T);
subplot(2, 2, 1), imshow(I), title('Image from problem 3');
subplot(2, 2, 2), imshow(a), title('Local/Regional Thresholding');
```





Global Thresholding

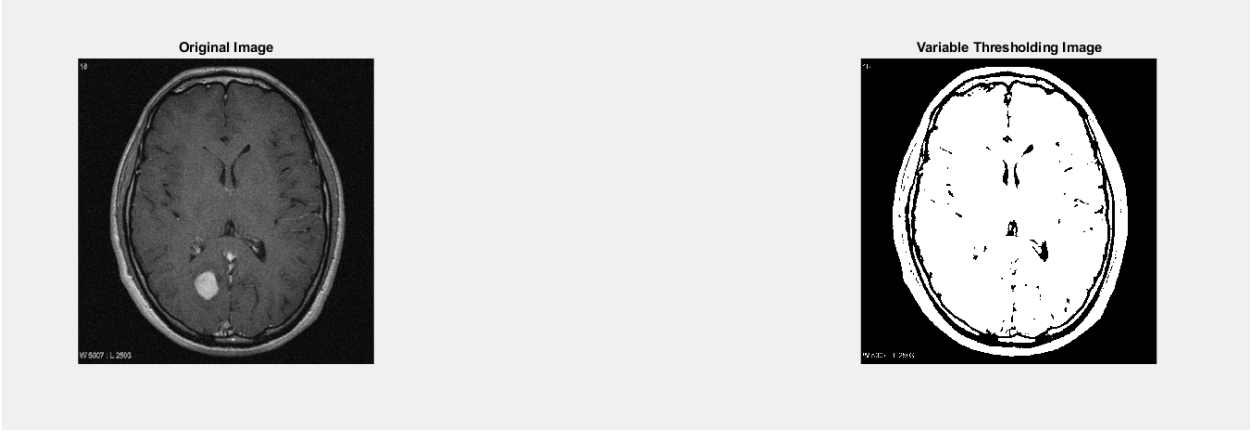
```
%% b) Global Thresholding
I = midpointimage;
BW = imbinarize(I,'global');
subplot(1,2,1);imshow(img);title('Original Image');
subplot(1,2,2);imshow(BW);title('Global Thresholding');
```



Variable Thresholding

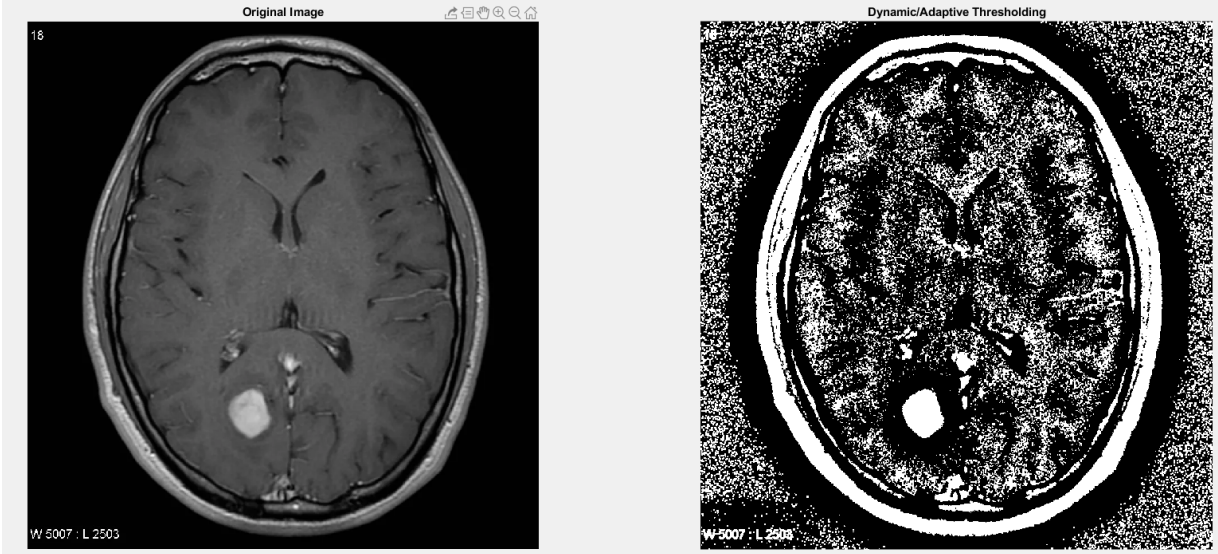
```
%% c) Variable Thresholding
I = midpointimage;
```

```
[r c]=size (I);
output=zeros (r, c);
starts=floor (1:c/10:c);
ends = starts (2:length(starts));
ends=[ends c];
figure, imshow(I);
hold on;
for i = 1:10
    plot([ends(i) ends(i)], [1,r], 'w');
end
for i = 1:10
    temp = img(:, starts(i):ends(i));
    out(:, starts(i):ends(i))=im2bw(temp,graythresh(temp));
    plot([ends(i) ends(i)], [1,r], 'w');
end
subplot(2, 2, 1), imshow(I), title('Original Image');
subplot(2, 2, 2), imshow(out), title('Variable Thresholding Image');
```



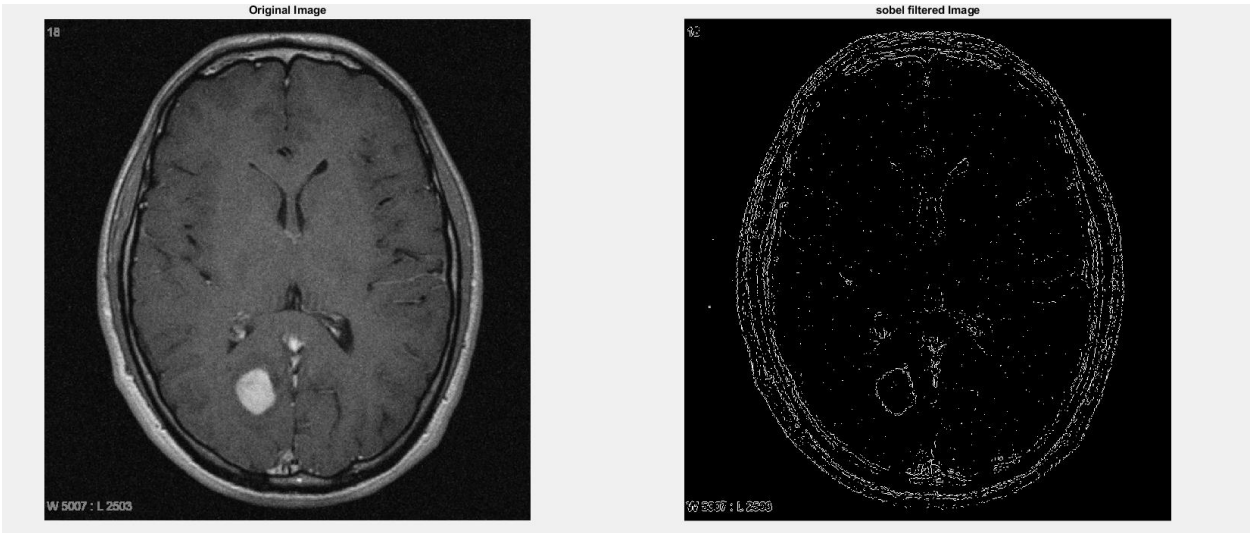
Dynamic/Adaptive Thresholding

```
%% d) Dynamic/Adaptive Thresholding
I = midpointimage;
BW = imbinarize(I,'adaptive');
subplot(1,2,1);imshow(img);title('Original Image');
subplot(1,2,2);imshow(BW);title('Dynamic/Adaptive Thresholding');
```

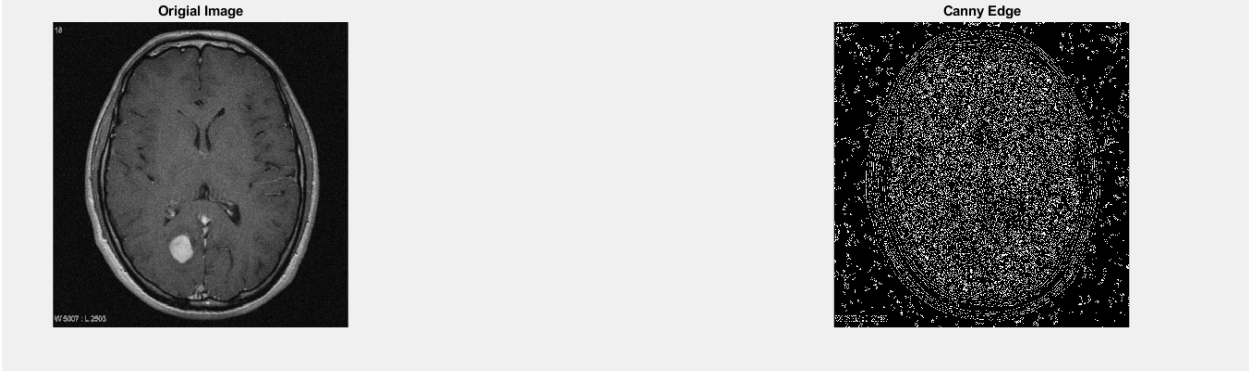


Discontinuity approaches: Edge Detection (Sobel, Canny, Prewitt)

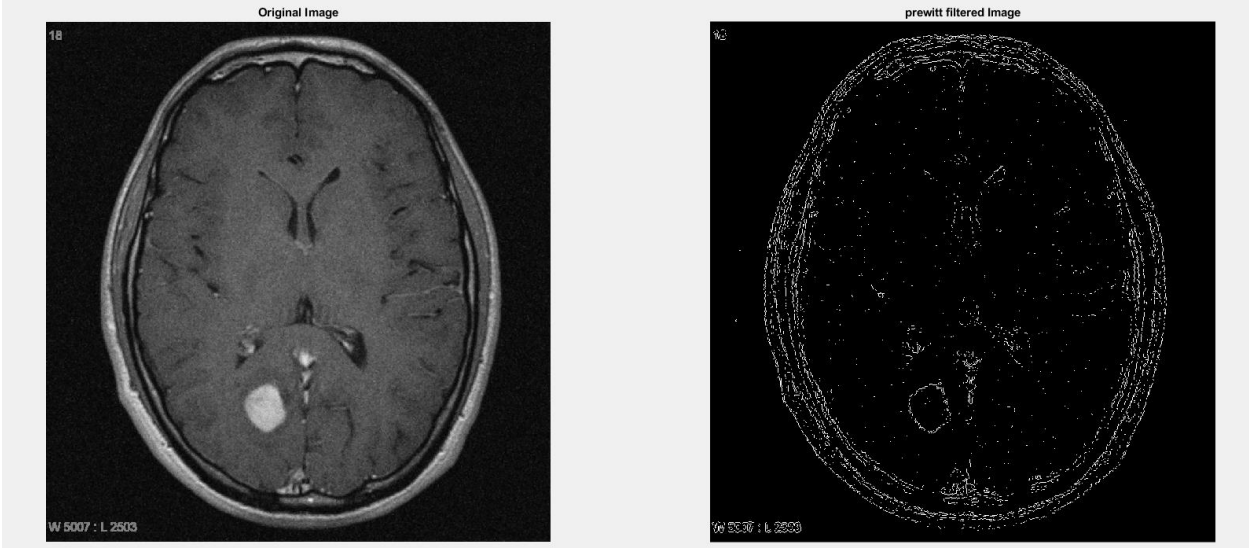
```
%% a)
img = midpointimage;
sobel = edge(img, 'Sobel');
subplot(1,2,1);imshow(img);title('Original Image');
subplot(1,2,2);imshow(sobel);title('sobel filtered Image');
```




```
%% b)
img_canny=edge(img, 'Canny');
subplot(2,2,1);imshow(img);title('Origial Image');
subplot(2,2,2);imshow(img_canny);title('Canny Edge');
```



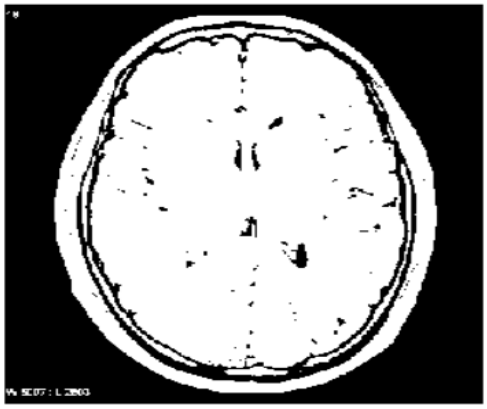

```
%% c)
prewitt = edge(img, 'Prewitt');
subplot(1,2,1);imshow(img);title('Original Image');
subplot(1,2,2);imshow(prewitt);title('prewitt filtered Image');
```



5. Show how the Similarity and Discontinuity techniques compare.

Similarity:

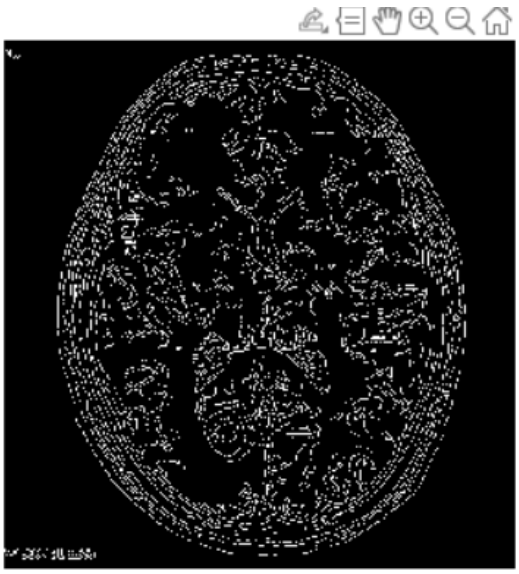
```
img = imread('Tumor.jpg');
img_gray = rgb2gray(img);
level = graythresh(img_gray);
img_thresh = imbinarize(img_gray, level);
imshow(img_thresh)
```



Similarity techniques are used to segment an image based on the similarity of pixels or regions. Examples include thresholding, region growing, and clustering. Advantages include simplicity, fast computation, and robustness to noise, but limitations include inability to accurately segment complex shapes.

Discontinuity:

```
img_gray = rgb2gray(img);
edges = edge(img_gray, 'canny');
imshow(edges)
```



Discontinuity techniques are used to segment an image based on the detection of abrupt changes or discontinuities. Examples include edge detection, corner detection, and line detection. Advantages include accurate segmentation of complex shapes and objects with varying properties, but limitations include noise and inability to accurately segment objects with smooth or gradual changes in properties.

Both similarity and discontinuity strategies have advantages and disadvantages, and the methodology to adopt relies on the characteristics of the item to be divided and the segmentation objectives. Combining both approaches can sometimes result in segmentation that is more precise and reliable.