MIAP HW04

Ali Ghavampour - 97102293

1 Theoretical Problems

1.1 Question 1

The basic idea of the level set method is to represent a contour as the zero level set of a higher dimensional function, called a level set function (LSF), and formulate the motion of the contour as the evolution of the level set function.
The general curve evolution function can be expressed as:

$$\frac{\partial C(s,t)}{\partial t} = FN$$

Level set function:

$$\Phi(C(s,t),t) = \Phi(x(s,t),y(s,t),t) = 0 \Rightarrow \frac{d\Phi}{dt} = \frac{\partial \Phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial \Phi}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial \Phi}{\partial t} = 0$$

$$\Rightarrow \nabla\Phi.\frac{\partial C(p,t)}{\partial t} + \frac{\partial \Phi}{\partial t} = 0 \qquad N = -\frac{\nabla\Phi}{|\nabla\Phi|}$$

Now we have:

$$\nabla\Phi.(VN) + \frac{\partial \Phi}{\partial t} = 0 \Rightarrow \frac{\partial \Phi}{\partial t} = \Phi.(V(-\frac{\nabla\Phi}{|\nabla\Phi|})) = V\nabla\Phi.(\frac{\nabla\Phi}{|\nabla\Phi|})$$

$$\Rightarrow \frac{\partial \Phi}{\partial t} = V|\nabla\Phi|(\text{PDE})$$

A desirable advantage of level set methods is that they can represent contours of complex topology and are able to handle topological changes, such as splitting and merging, which is not allowed in parametric active contour models easily. Another desirable feature of level set methods is that numerical computations can be performed on a fixed Cartesian grif without having to parameterize the points on a contour as in parametric active contour models and you can work in any desirable relolution

1.2 Question 2

The energy function is defined as:

$$\epsilon(\Phi) = \mu R_p(\Phi) + \epsilon_{ext}(\Phi)$$

And the regularization term is defined as:

$$R_p(\Phi) = \int_\Omega p(|\nabla\Phi|)dx$$

where is a potential (or energy density) function. A simple and straightforward definition of the potential for distance regularization is,

$$p_1(s) = \frac{1}{2}(s-1)^2$$

However, the derived level set evolution for energy minimization has an undesirable side effect on the LSF in some circumstances. To avoid this problem, another potential function p_2 is defined. This new potential function is aimed to maintain the signed distance property $|\nabla\Phi| = 1$ only in a close area arount zero level set, while keeping the LSF as a constant, with $|\nabla\Phi| = 0$ at locations far from the zero level set. p_2 is defined as:

$$p_2(s) = \begin{cases} \frac{1}{(2\pi)^2}(1 - cos(2\pi s)) & s \leq 1 \\ \frac{1}{2}(s-1)^2 & s \geq 1 \end{cases}$$

1.3 Question 3

The DRLSE eliminates the need for reintialization. They use a binary step function as the initial LSF becuase it can be generated efficiently.

$$\Phi_0(x) = \begin{cases} -c_0 & x \in R_0 \\ c_0 & otherwise \end{cases}$$

Also, the region R_0 can be obtained by a simple segmentation step, such as thresholding. Then, R_0 would be close to the region to be segmented. So only a few number of iterations would be needed to converge.

2 Question 1

2.1 Part 1

Four variation of merged images are shown in figure 1

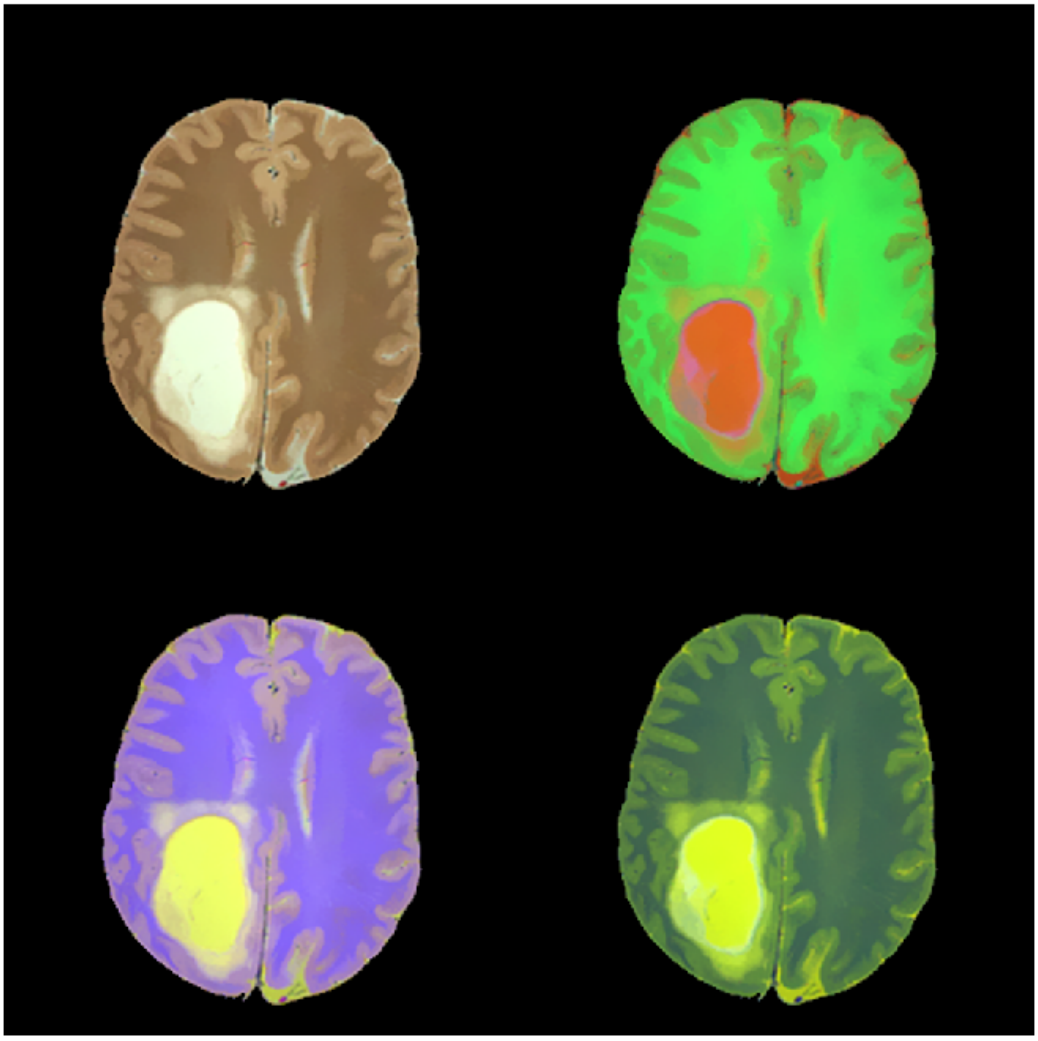

Figure 1

2.2 Part 2

For three different fuzziness parameters the FCM is applied on the MRI1.bmp image. The results are as follow.

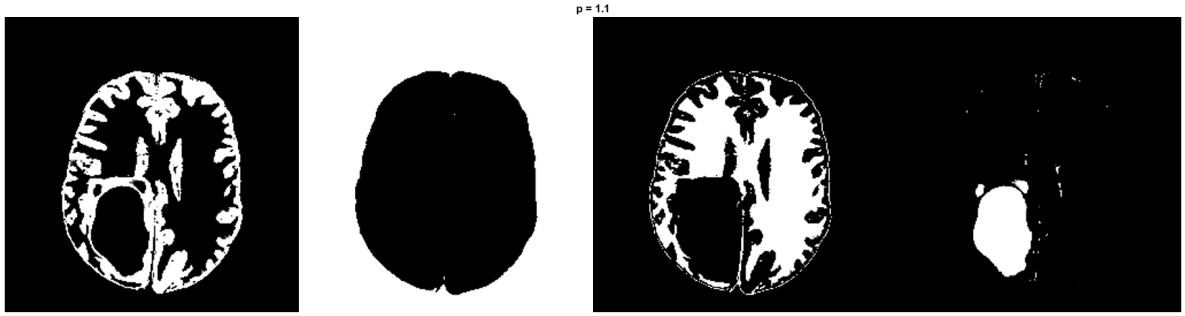

Figure 2

Because the parameter must be larger than 1.0, instead of 1.0, 1.1 is used. The fuzziness parameter determines the overlap between segments. Larger parameter values mean that the segments have more overlap and the output would be more probabilistic. The smaller values contribute to more deterministic $u_{i,j}$. For example, the probabilities of the pixels in larger parameters would be smaller than 1 and for larger parameters it would be closer to 1.

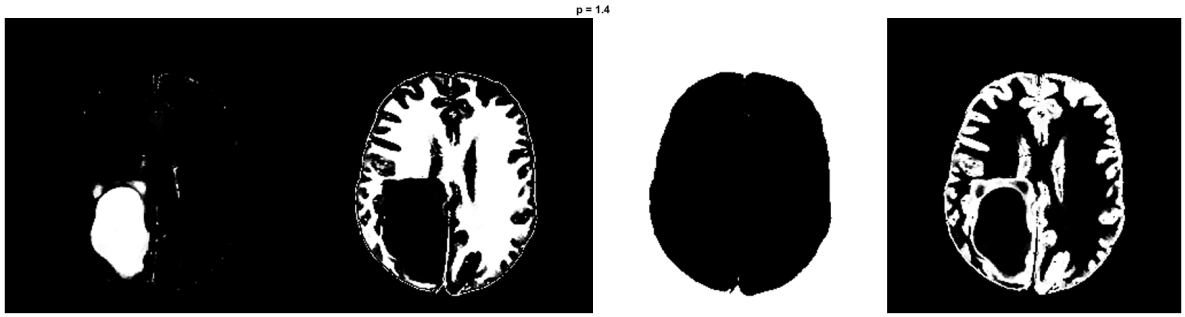

Figure 3

The results for 1.1 and 1.4 are quite similar and we can see that all the pixel are either white or black. That is because the more deterministic probabilities for smaller values. (Figure 2 and 3)

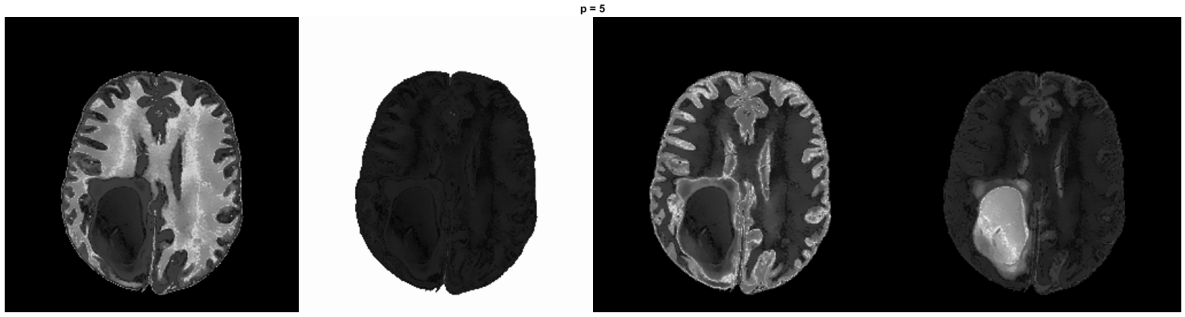

Figure 4

In figure 4 we can see more gray pixels. That is because the probabilities are less than 1 and each pixel could be related to either segments.

Using probablity maps, the outputs of FCM are labled using colors. The color is decided by comparing the values of u_{ij} for each pixel between different segments. We can see that in the right image, the colors are not that bright which is caused by overlap.

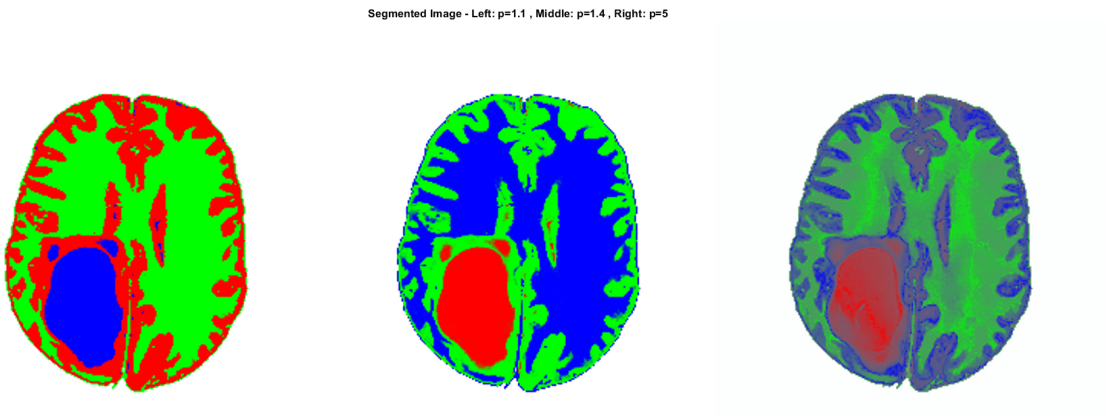

Figure 5

2.3 Part 3

The K-Means segmentation is shown in figure 6.

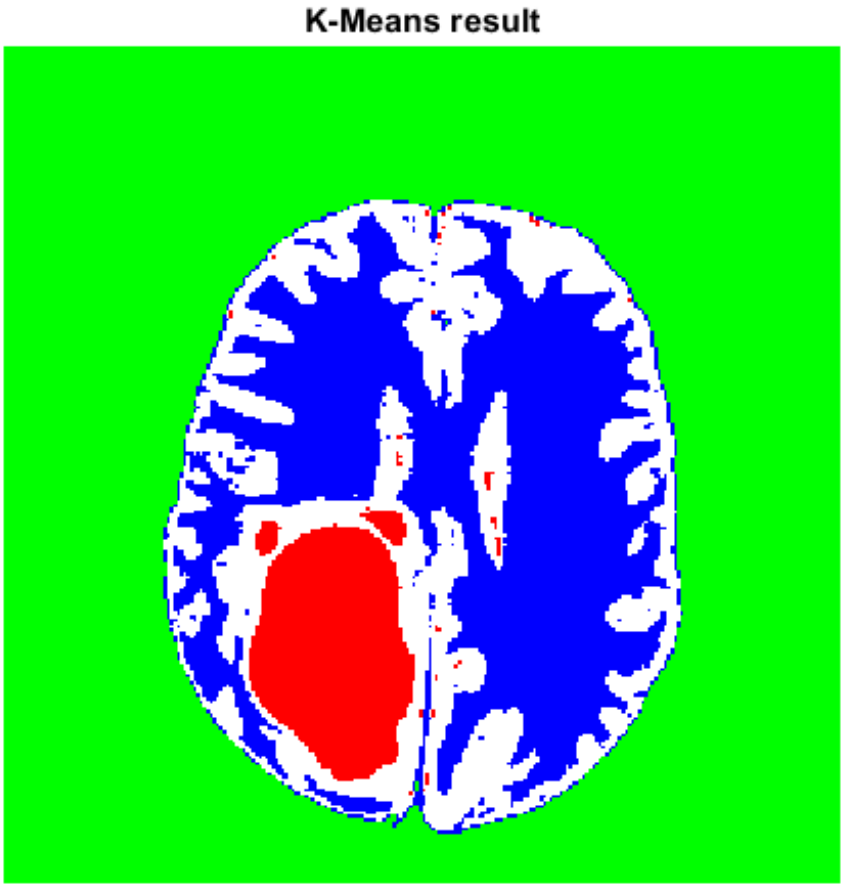

Figure 6

The FCM results are show in figure 7.

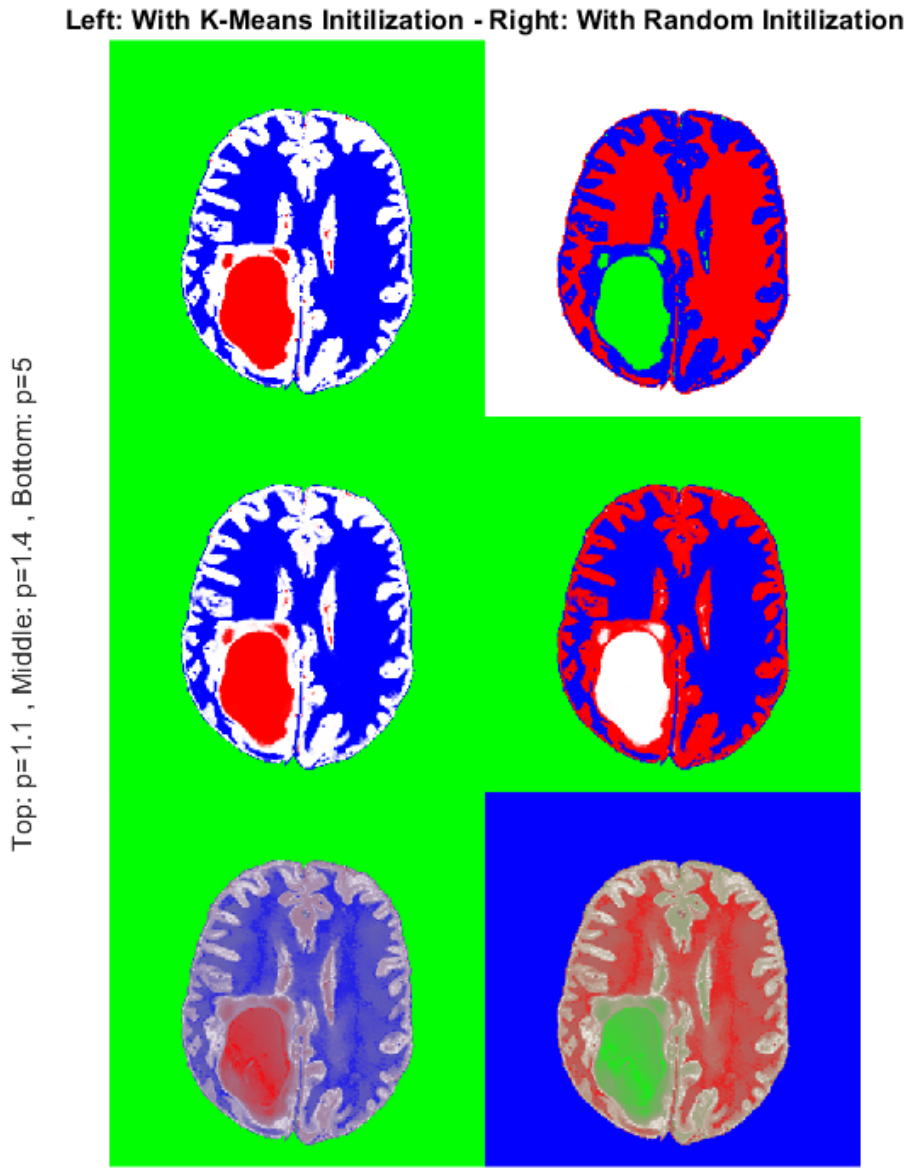

Figure 7

We can see that the segmentation result is not changed when we used k-means. So, what is the difference? The only observed difference is the number of iterations and execution time. It is shown in figure 8.

	P=1.1	**P=1.4**	**P=5**
With K-Means	0.4358 s	0.3217 s	0.8692 s
Without K-Means	0.3256 s	0.5729 s	1.1839 s

Figure 8

2.4 Part 4

Using fitgmdist function and component number of 4, we find the Gaussian distributions as shown in figure 9. The dashed vertical line are the mean of the fitted Gaussians.

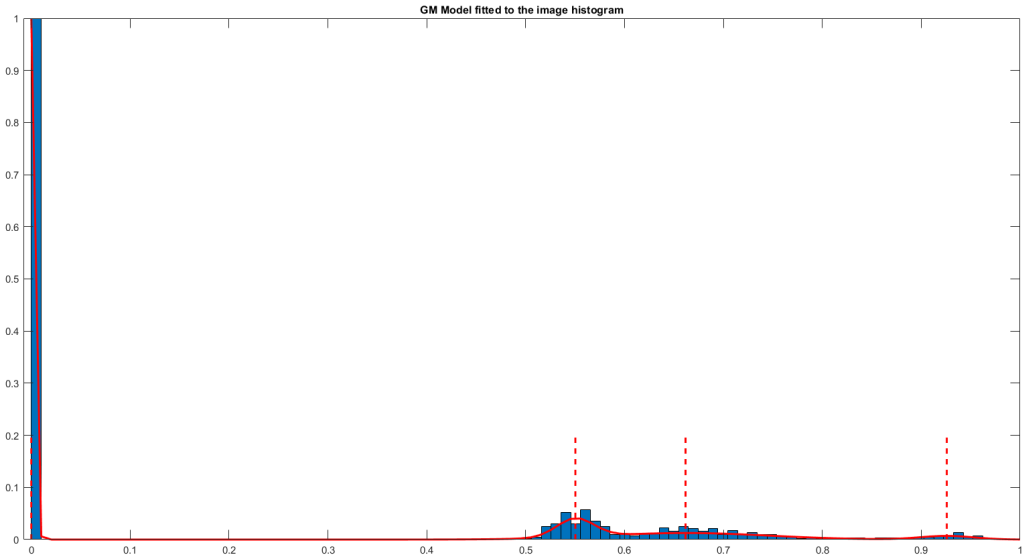

Figure 9

Result of segmentation is shown in figure 10. We can see that this result is as good as last parts.

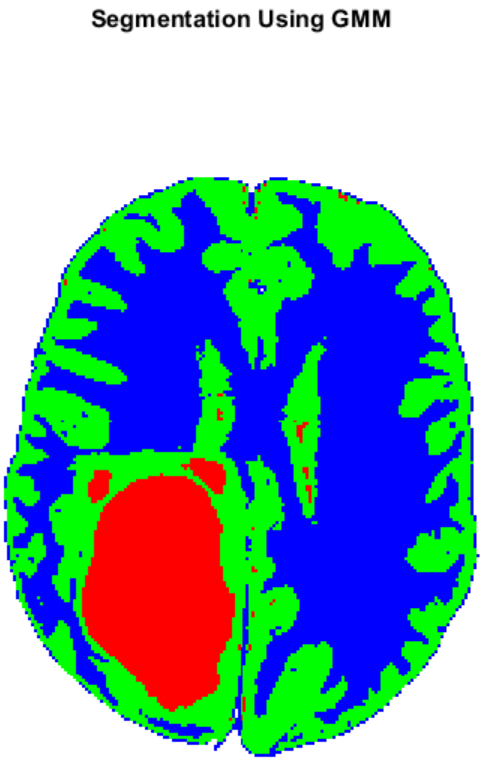

Figure 10

3 Question 2

3.1 Part 1

The basic snake method can not fill the U shaped areas because the force functions are zero in those areas. Wheras the GVF does not have this problem because the energy function is different for this method and is not zeros in the U areas.

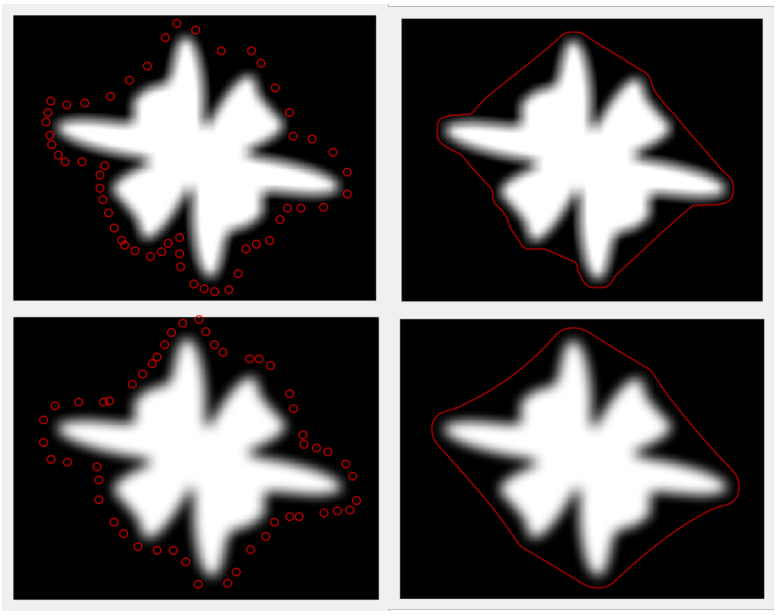

Figure 11: Snake Model - Blur1

We can see that the snake model can not fill the U shaped areas.

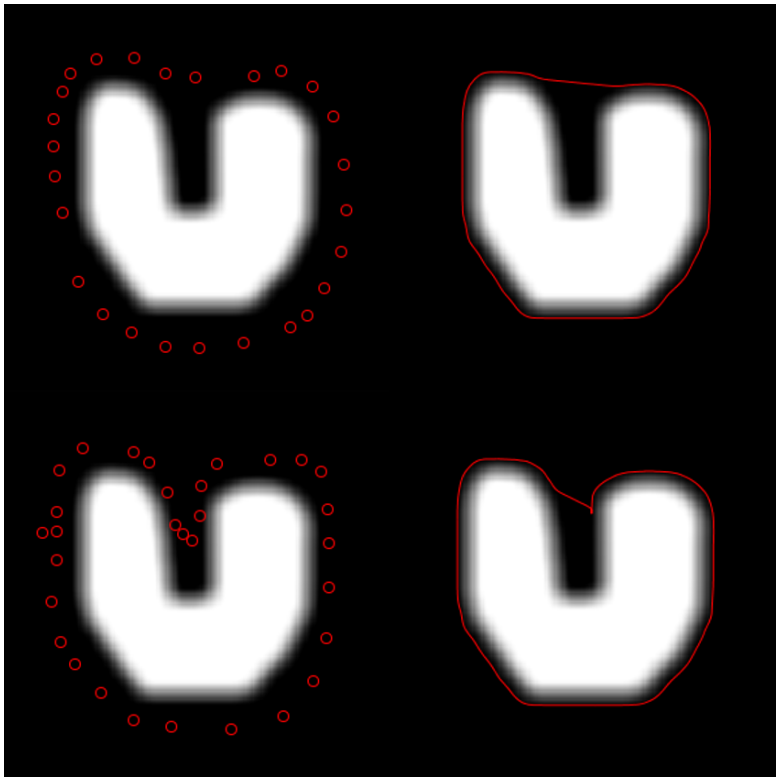

Figure 12: Snake Model - Blur2

The GVF method does better in this case unless the initial values are not good enough.

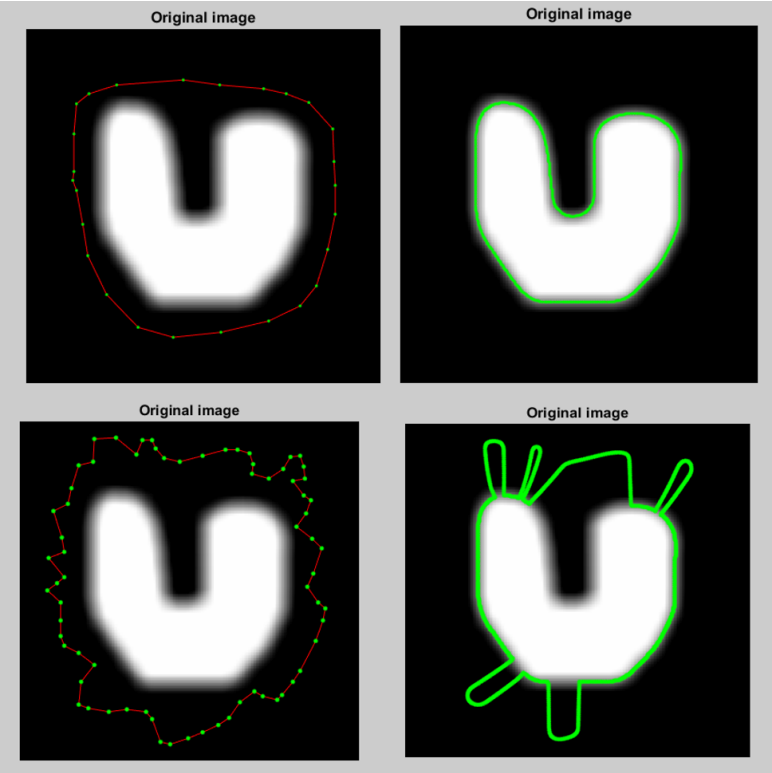

Figure 13: GVF - Blur1

We can see that the GVF compeletly takes the shape unless the initial value is not well choosed.

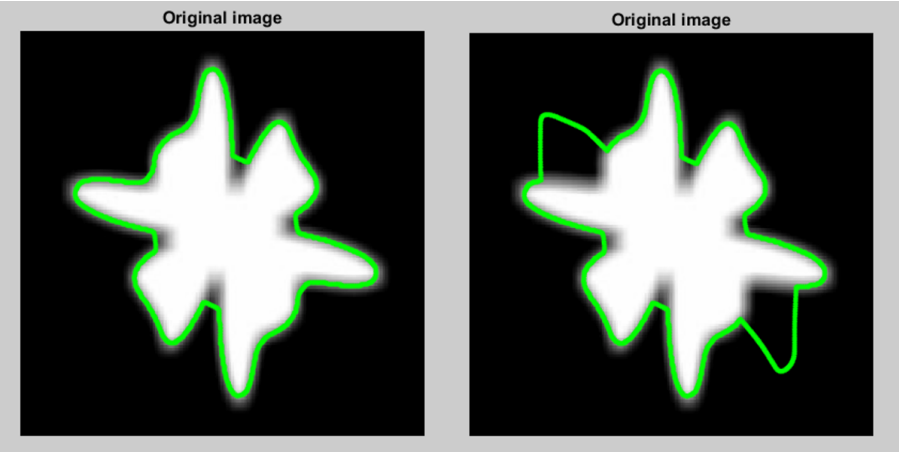

Figure 14: GVF Model - Blur2

3.2 Part 2

In this part we use the methods on some of the areas in the brain MRI images.

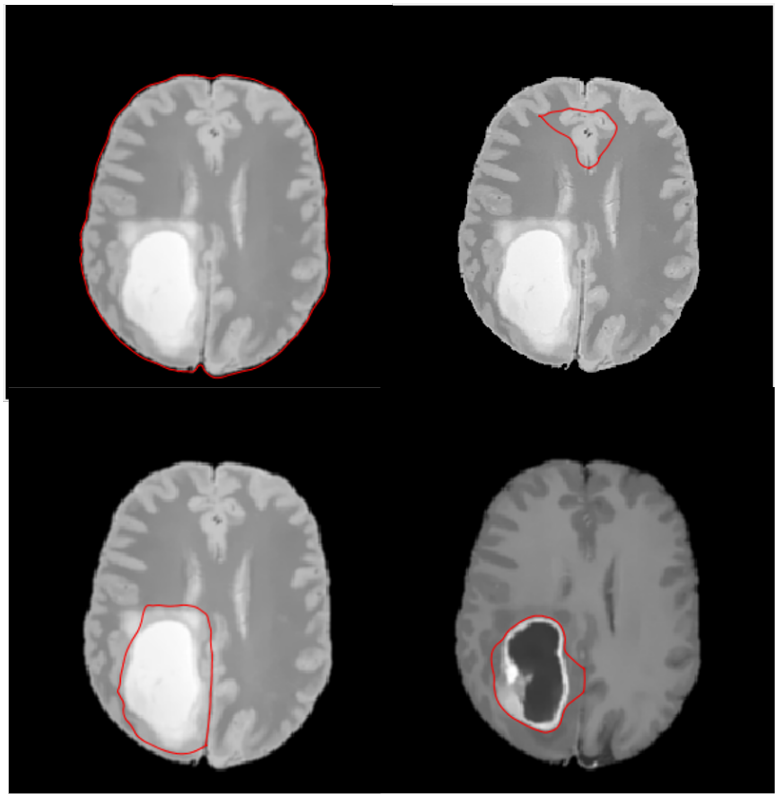

Figure 15: Snake Model - MRI

The snake model does not fit very well inside the brain. But it can fit the whole brain itself because it's in a dark background.

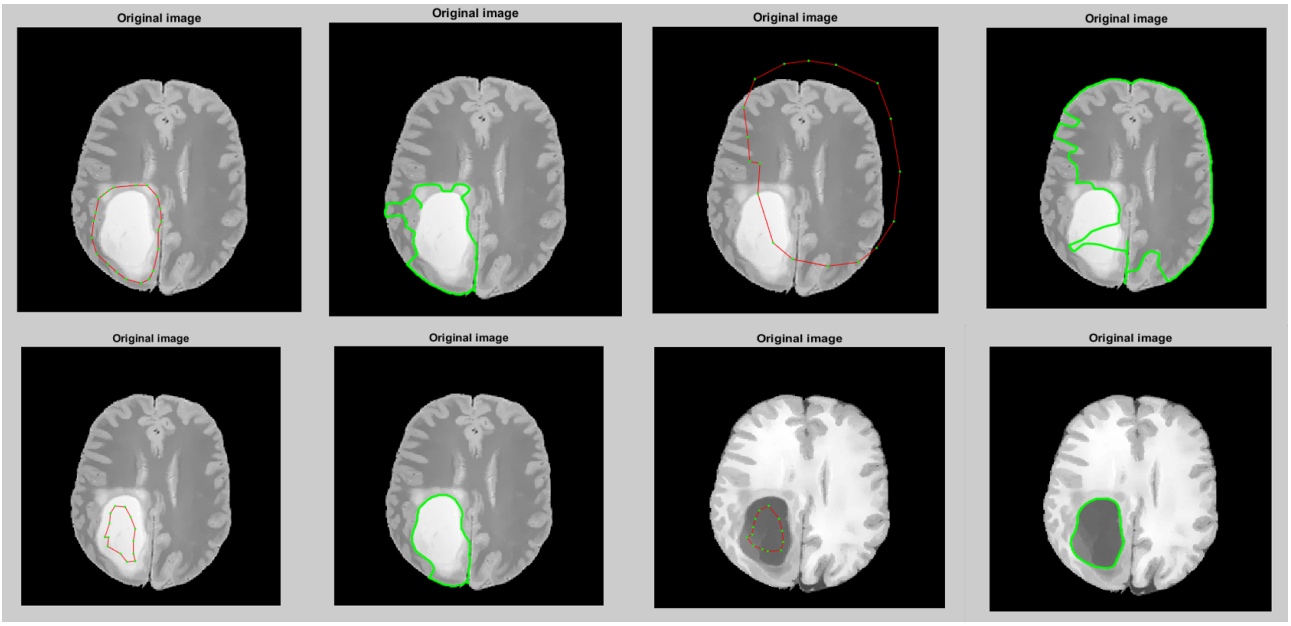

Figure 16: GVF - MRI

The GVF has done a better job in the MRI images since it fits the shape of the tumore in some modalities.

This is the vectore field for GVF model:

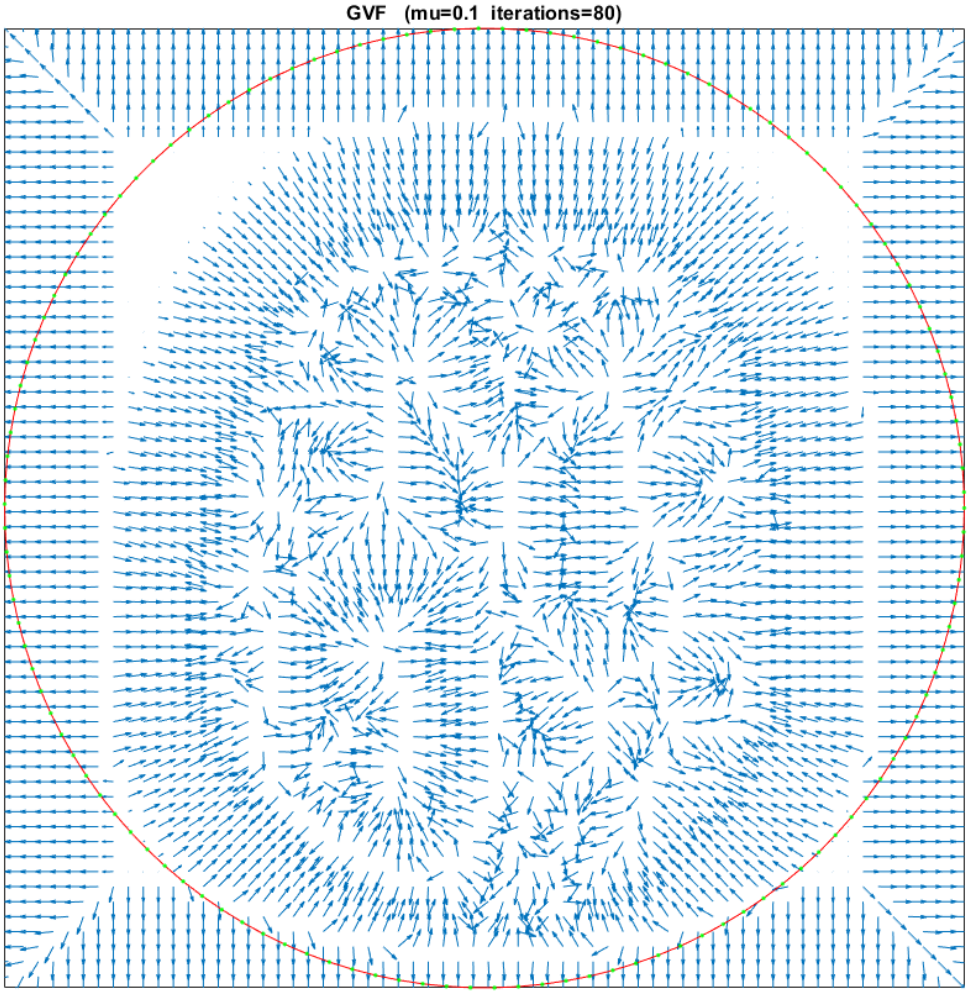

Figure 17: GVF Vector Field - MRI

The vectors around the tumor are pointing towards it in the bottom left side of the image.

4 Question 3

4.1 Part 1

The ARKFCM is designed to work well in noisy images. In order to measure noise locally LVC (local variation coefficient) is calculated for a neighborhood around every pixel. Larger LVCs correspond to noisier areas. Then based on these coefficients, a weight is calculated for that local window. Then based on the local window weights, some weights are calculated for every pixel of the image. If the intensity of a pixel equals to the average intensity of the local window around it, the algorithm works just like a normal FCM for this pixel. Also the distance in this algorithm is not the Euclidian distance since it is sensetive to noise and perturbations. The distances are calculated using a Gaussian kernel.
The cost function is:

$$J_{ARKFCM} = 2[\sum_{i=1}^{N}\sum_{j=1}^{c}(1 - K(x_i, v_j)) + \sum_{i=1}^{N}\sum_{i=1}^{N}\phi_i {u_{ij}}^m (1 - K(\bar{x}_i, v_j))]$$

4.2 Part 2

Every function in the code is explained.
pixWgt.m: It calculated the pixel weigths ϕ_i

gaussKernel.m: It is the Guassian kernel K

kerWidth.m: It calculates the sigma in the image σ

distARKFCM.m: It calcuates the distance of the pixels

ARKFCM.m: It solves the ARKFCM problem and minimizes the cost function in order to segment the image.

4.3 Part 3

The algorithm is run for three different images. The first image is corrupted with 7% noise and 20% grayscale nonuniformity. The result is as follow:

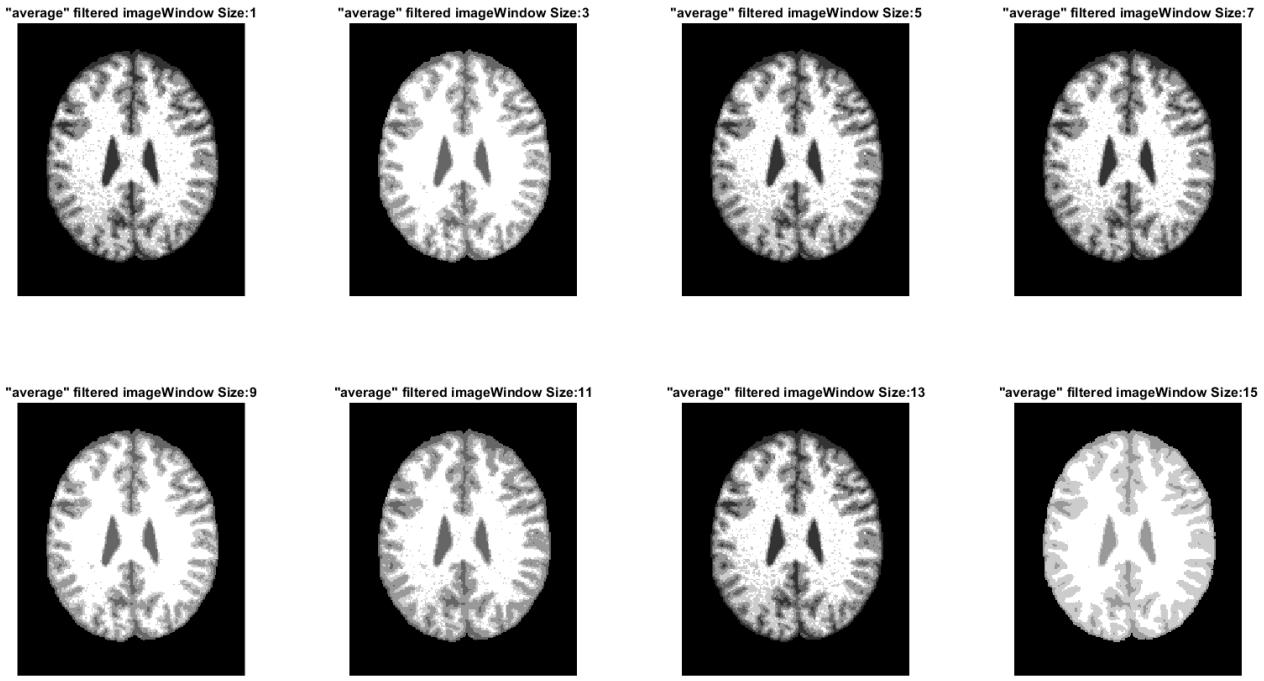

Figure 18: Image 1

It is interesting that for the window sizes of multiple of 3, the algorithm has worked the best.

The second image is another slice with the same corruption.

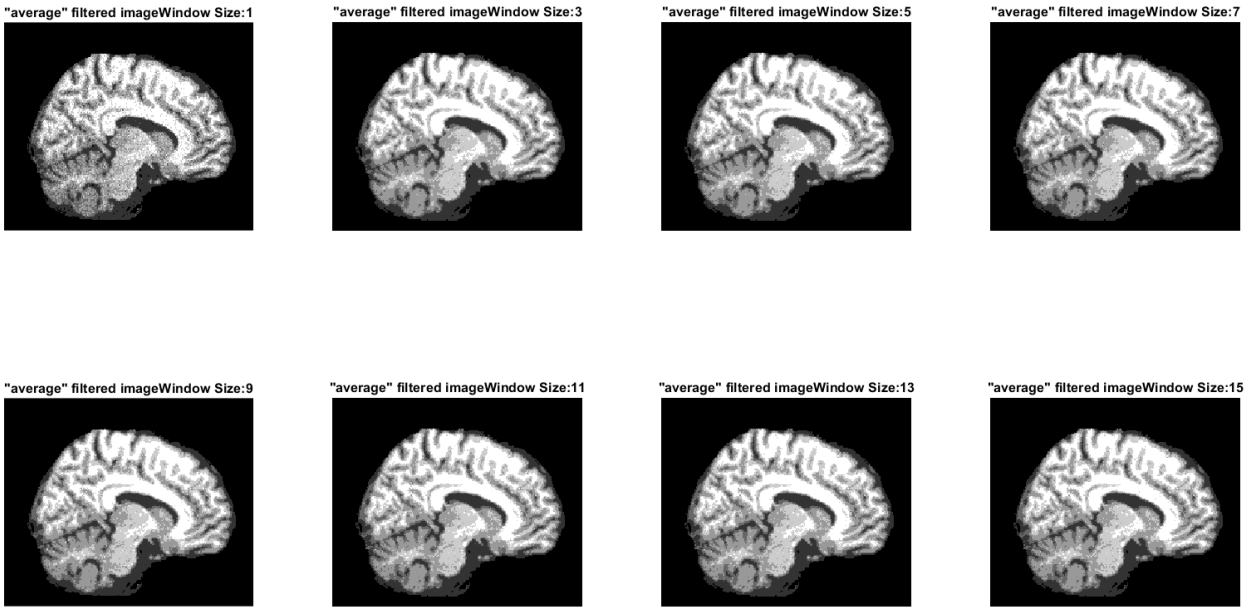

Figure 19: Image 2

The third image is corrupted with 10% Rician noise.

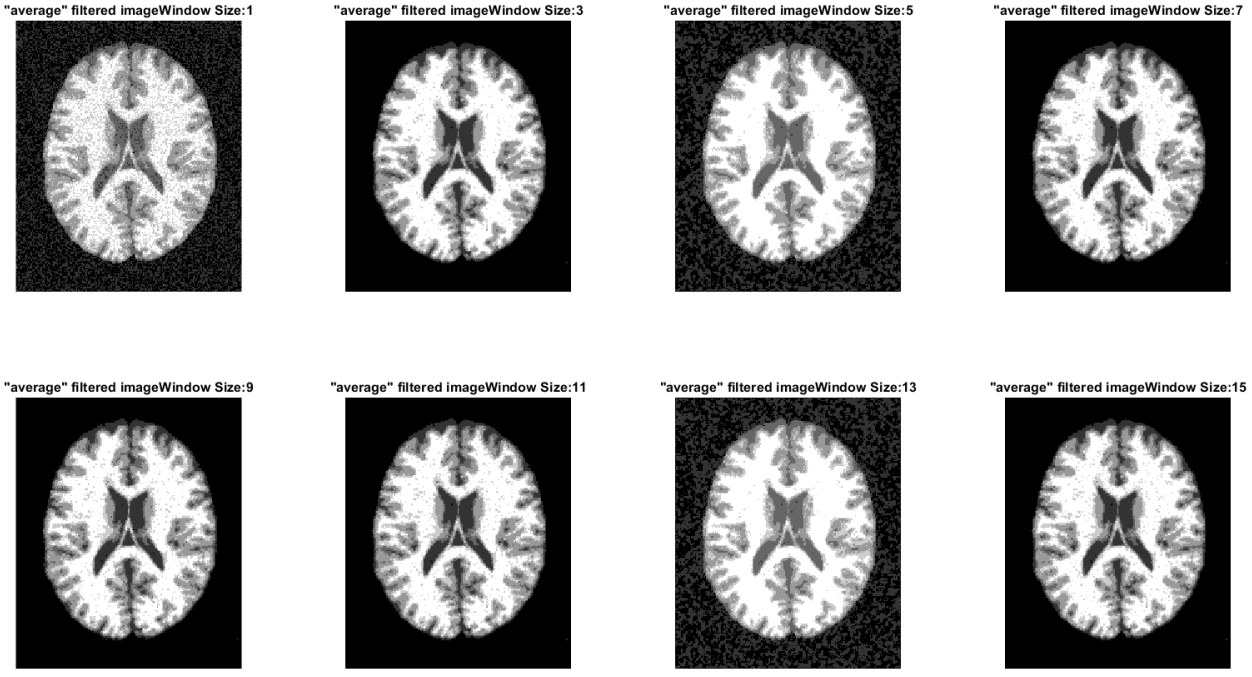

Figure 20: Image 3

For all of the noise levels, the algorithm has worked similarly and acted well.

4.4 Part 4

For different cluster numbers, the algorithm is run on the image 1.

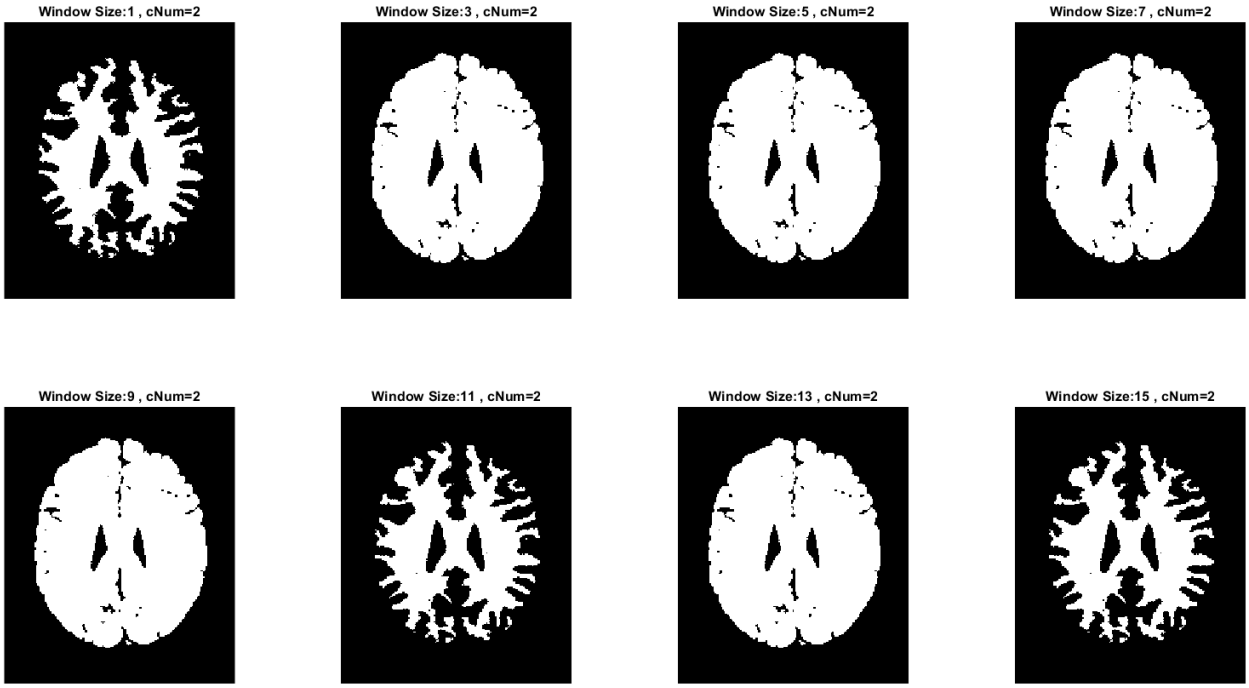

Figure 21

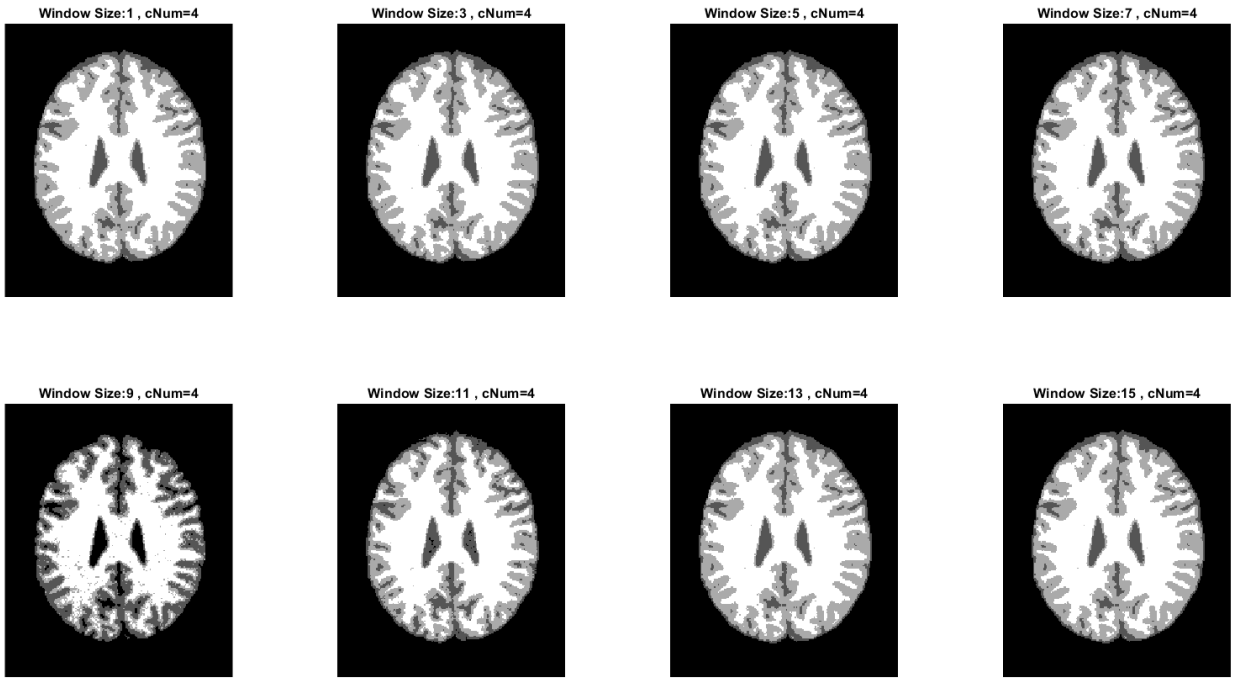

Figure 22

The image is no more than 5 or 6 clusters so for cNum=10 the output is not quite good. We can conclude that as the cNum increases, the output is more vulnerable to noise and more spots are visible inside the segments.

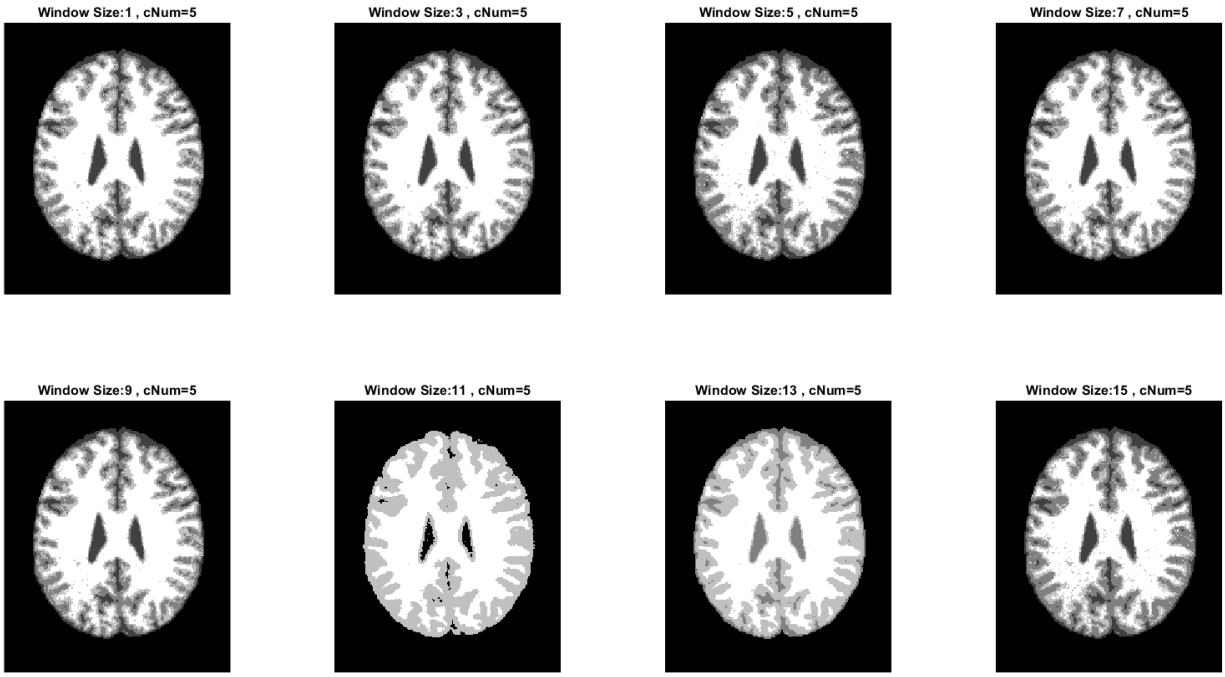

Figure 23

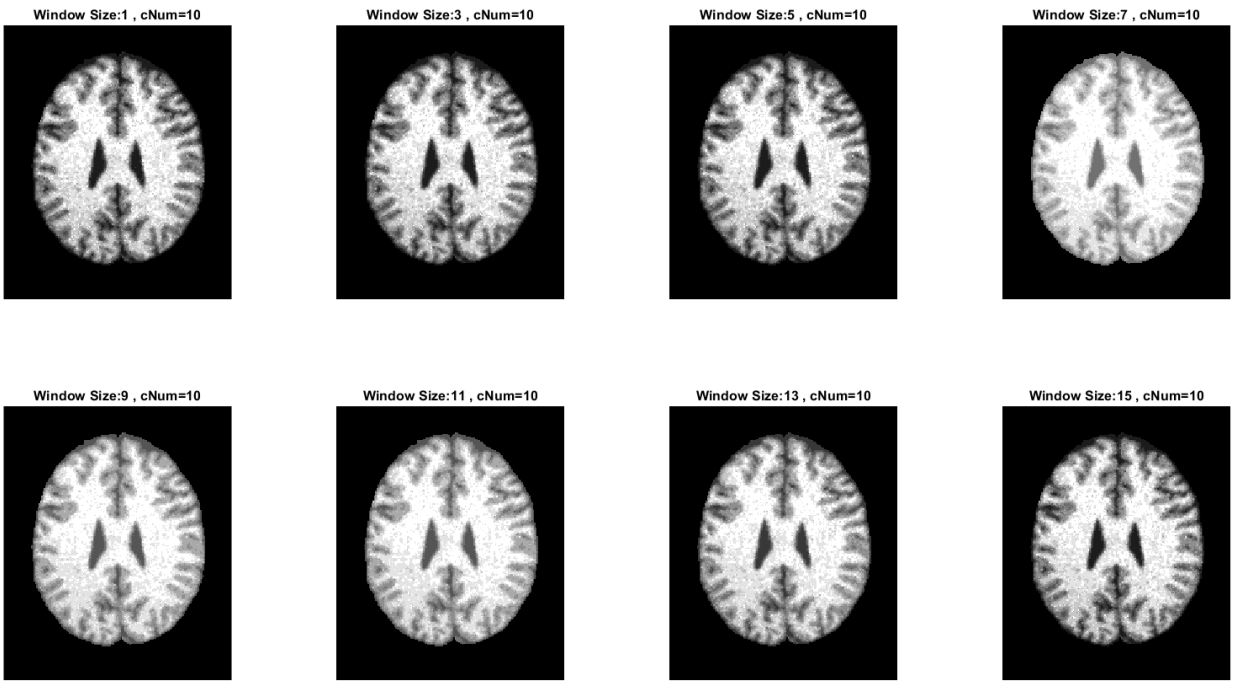

Figure 24

4.5 Part 5

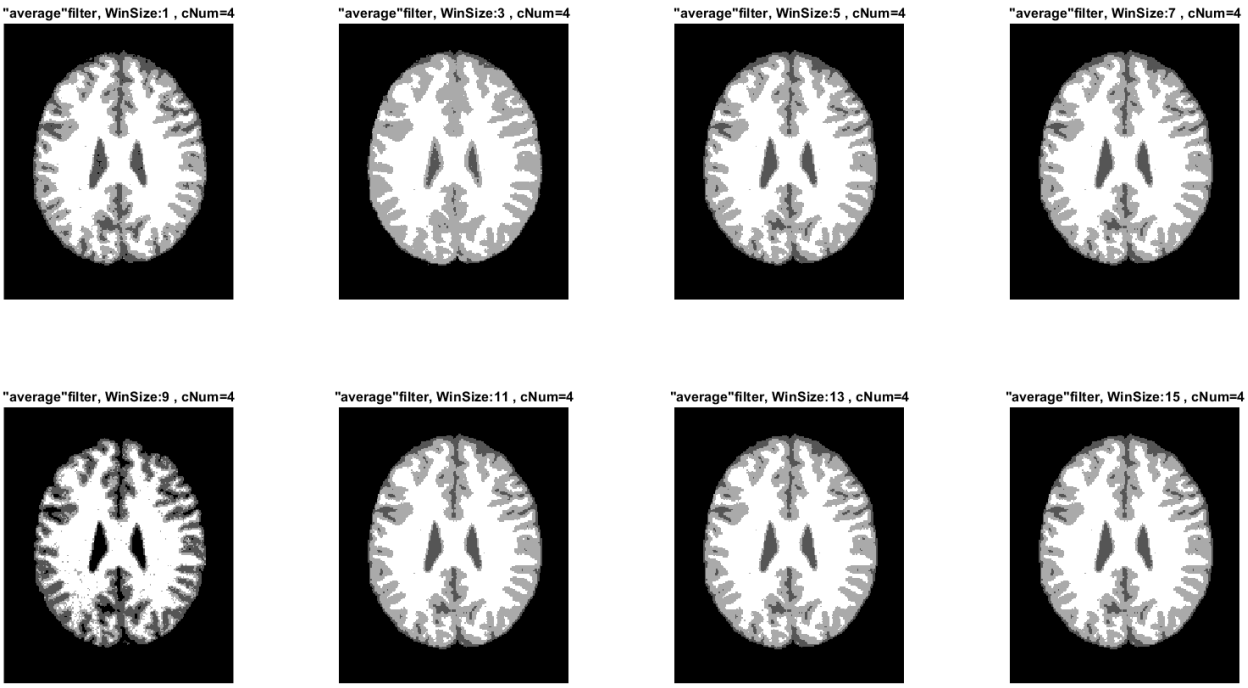

Figure 25

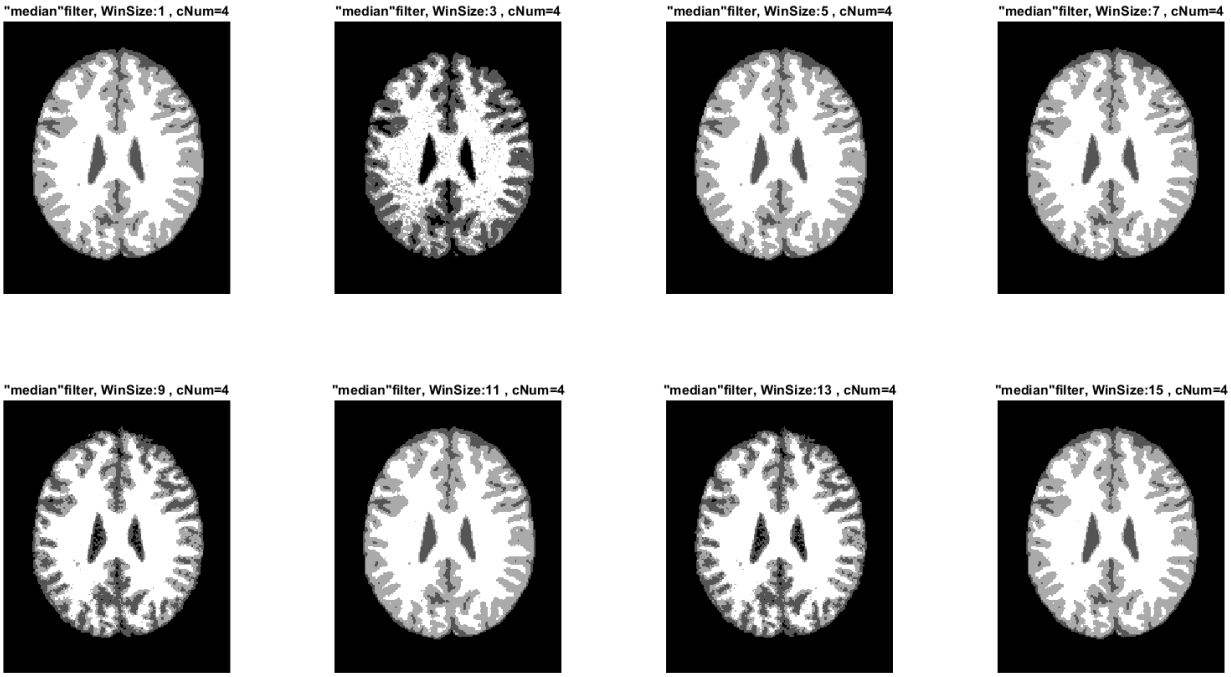

Figure 26

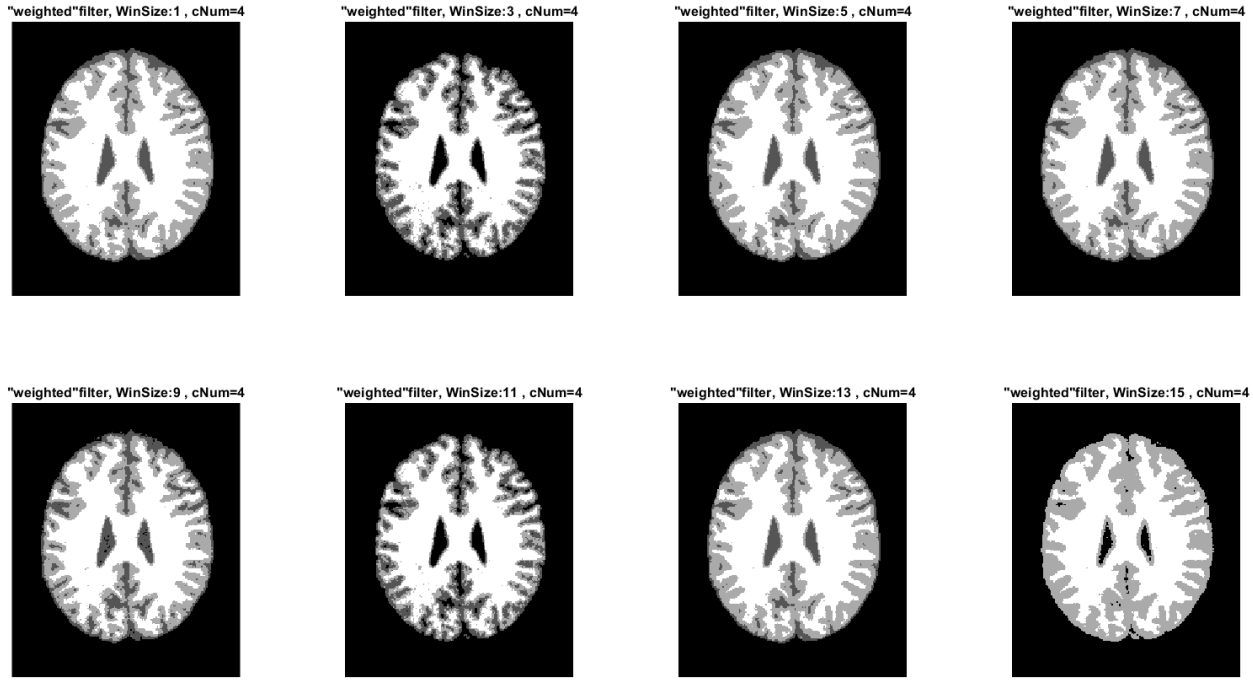

Figure 27

The filters seem to act similarly and there is not any special difference in them. Similarly, table 1 of the paper did not show any difference between the filters.

4.6 Part 6

Window size of 29 seem to be better than window size of 1 in most of the images. Except in on of them that the output is compeletely white.

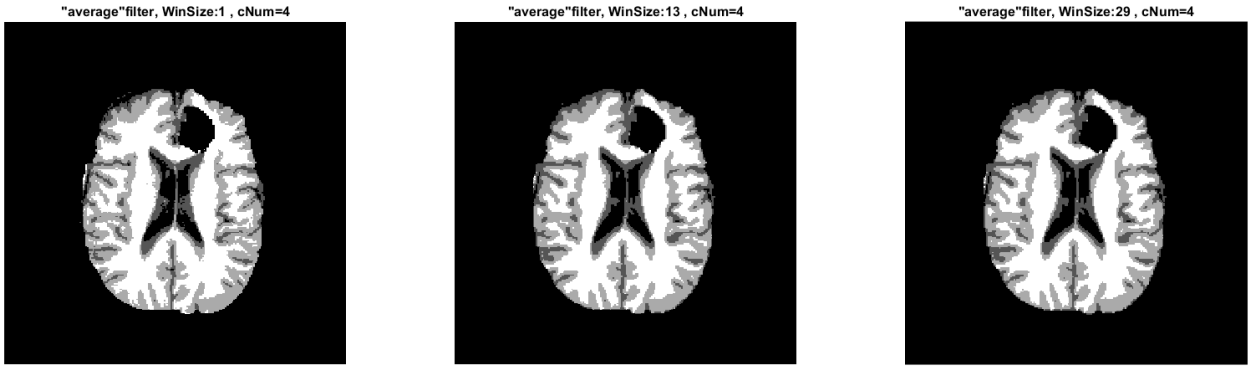

Figure 28

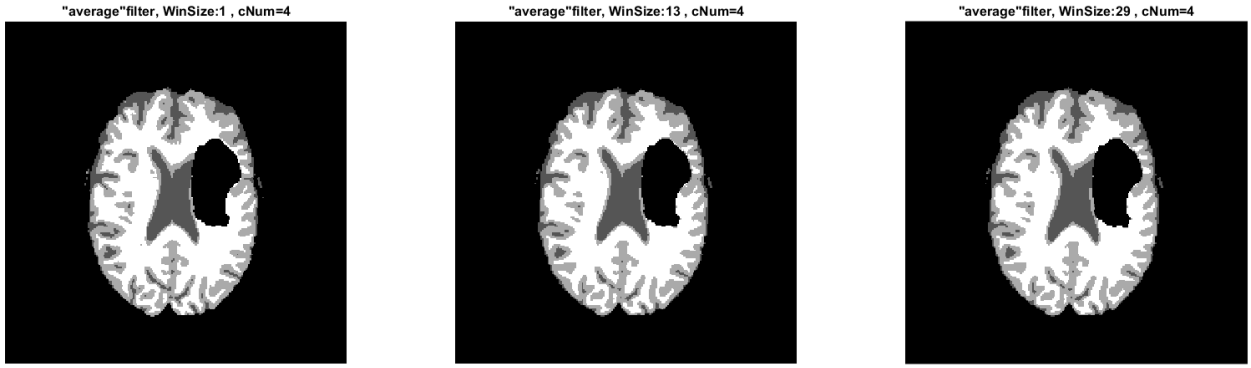

Figure 29

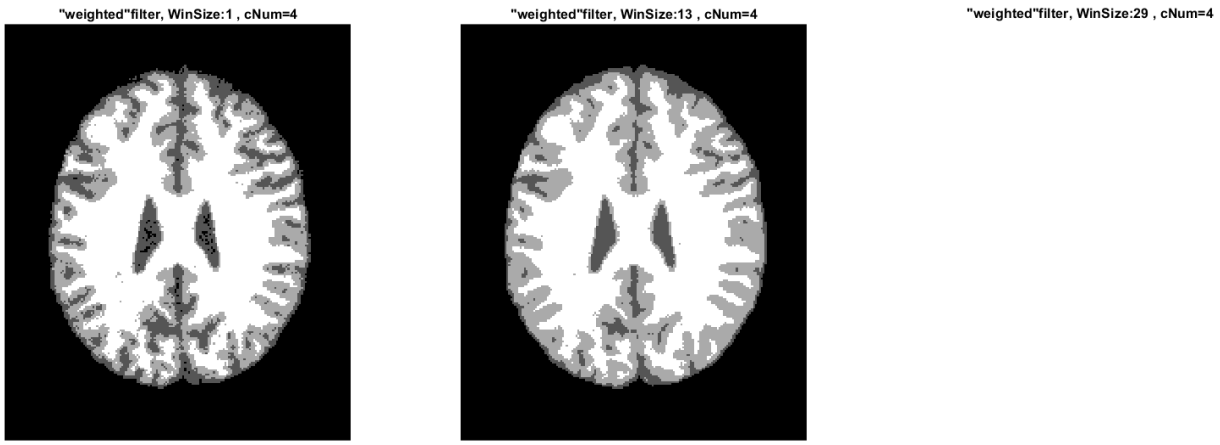

Figure 30

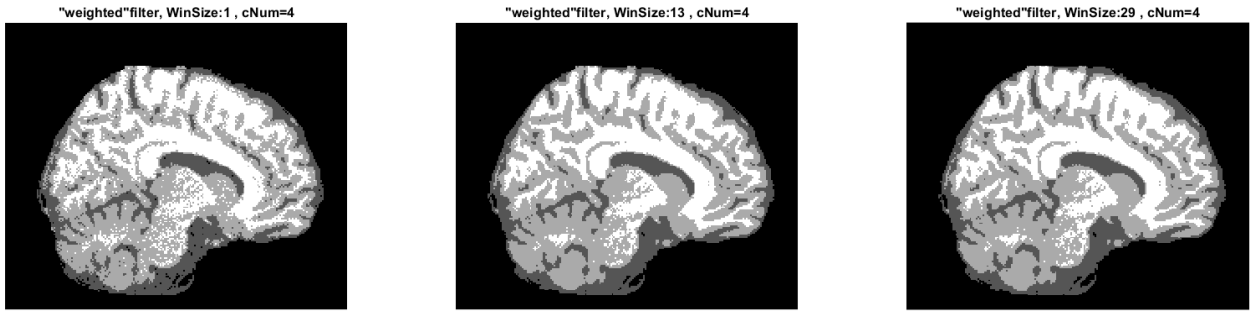

Figure 31

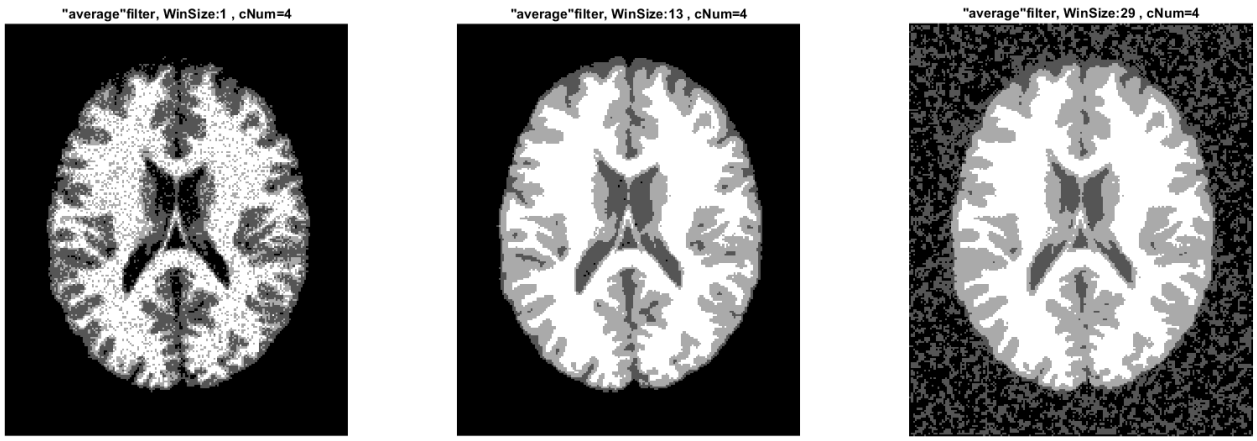

Figure 32

In most of the images the output of window size 1 is noisier.