In the game of God

Sharif University of Technology

Dr. Fatemizade

Amirreza Hatamipour

97101507

Theoretical exercise:

Question 1:

Level set methods have been widely used in image processing and computer vision. The level set evolution is derived as the gradient flow that minimizes an energy functional with a distance regularization term and an external energy that drives the motion of the zero level set toward desired locations. The distance regularization term is defined with a potential function such that the derived level set evolution has a unique forward-and-backward (FAB) diffusion effect, which is able to maintain a desired shape of the level set function, particularly a signed distance profile near the zero level set. This yields a new type of level set evolution called distance regularized level set evolution (DRLSE)

The distance regularization effect eliminates the need for reinitialization and thereby avoids its induced numerical errors.

The curve evolution can be expressed as:

$$\frac{\partial \mathcal{C}(s,t)}{\partial t} = F\mathcal{N}$$

the curve evolution equation (1) is converted to the following partial differential equation (PDE):

$$\frac{\partial C(\rho,t)}{\partial t} = F\vec{N} \qquad where \quad \vec{N} = -\frac{\nabla\varphi}{|\nabla\varphi|}$$

$$\nabla\varphi.\frac{\partial C(\rho,t)}{\partial t} + \frac{\partial\varphi}{\partial t} = 0 \qquad \Rightarrow \quad \nabla\varphi.\left(F\vec{N}\right) + \frac{\partial\varphi}{\partial t} = 0$$

$$\frac{\partial\varphi}{\partial t} = -\nabla\varphi.\left(F\left(-\frac{\nabla\varphi}{|\nabla\varphi|}\right)\right) = F(k)\,\nabla\varphi.\left(\frac{\nabla\varphi}{|\nabla\varphi|}\right)$$

$$\frac{\partial\phi}{\partial t} = F|\nabla\phi|$$

Question 2:

energy functional by:

$$\mathcal{E}(\phi) = \mu\mathcal{R}_p(\phi) + \mathcal{E}_{\text{ext}}(\phi)$$

The level set regularization term is defined by:

$$\mathcal{R}_p(\phi) \triangleq \int_\Omega p(|\nabla\phi|)d\mathbf{x}$$

A naive choice of the potential function is for the regularization term, which forces to be zero. Such a level set regularization term has a strong smoothing effect, but it tends to flatten the LSF and finally make the zero level contour disappear.

A simple and straightforward definition of the potential for distance regularization is:

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

And:

$$p_2(s) = \begin{cases} \frac{1}{(2\pi)^2}(1-\cos(2\pi s)), & \text{if } s \le 1 \\ \frac{1}{2}(s-1)^2, & \text{if } s \ge 1. \end{cases}$$

This potential has two minimum points at and. It is easy to verify that is twice differentiable in, with the first and second derivatives

Question 3:

This equation can be expressed as:

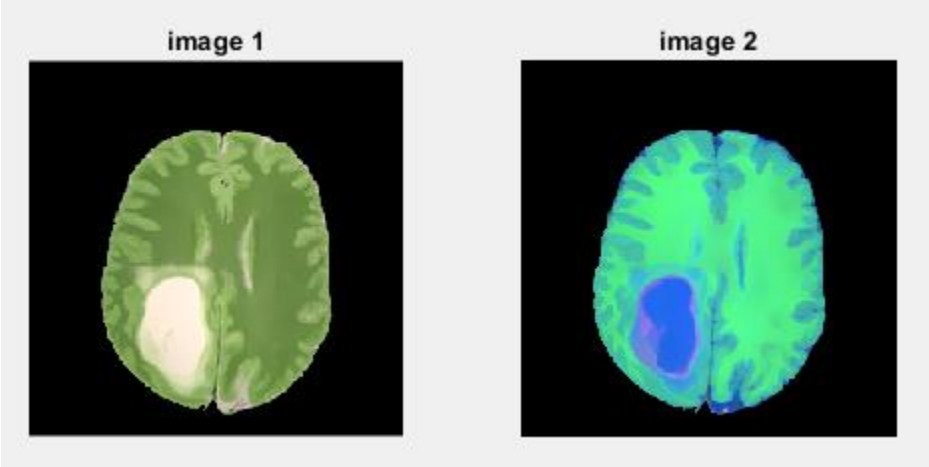
$$\phi_{i,j}^{k+1} = \phi_{i,j}^{k} + \Delta t L\left(\phi_{i,j}^{k}\right), \qquad k = 0, 1, 2, \ldots$$

And the profit of this method is given spacial steps, the choice of the time step for this finite difference scheme must satisfy the CFL condition for numerical stability.

Simulation exercises:

Question 1:

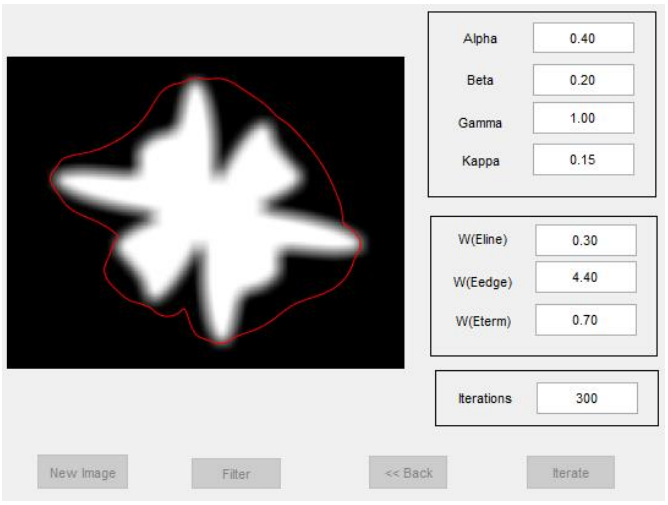
a) as we see, I think 3 class was enough to cluster this MRI**.(4 class with background)**

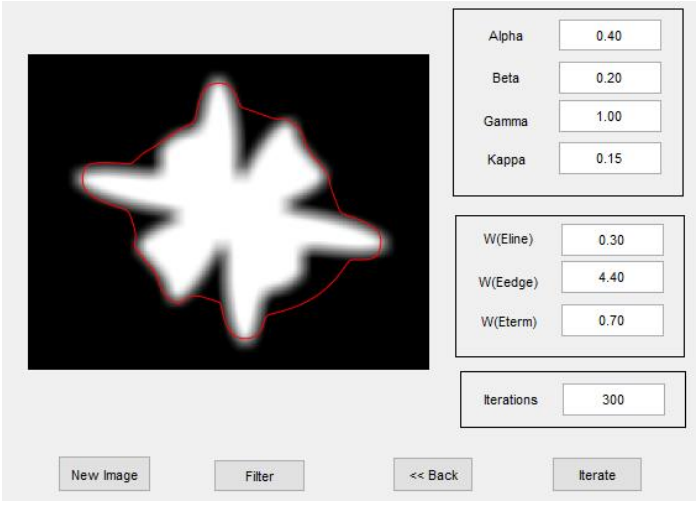


Question 2:

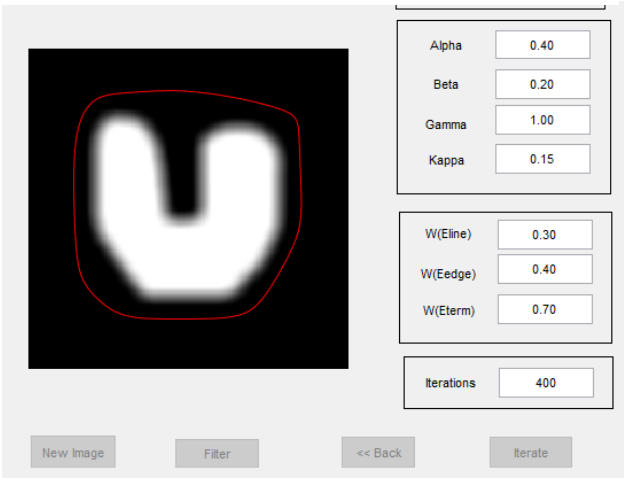
a) if we choose this parameters and basic countour like below we have:

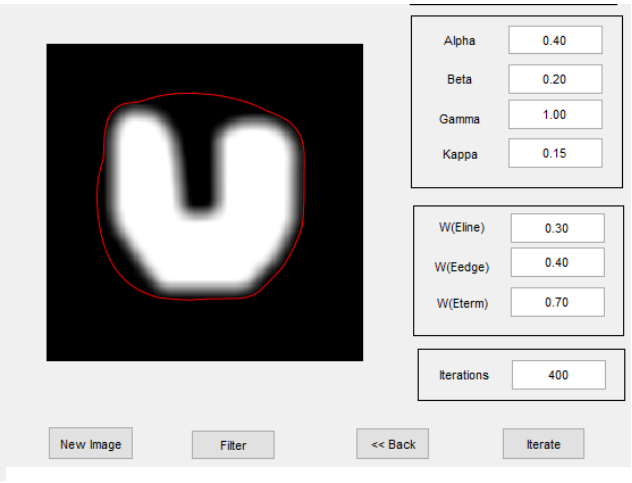
Another one increase W (Edge):



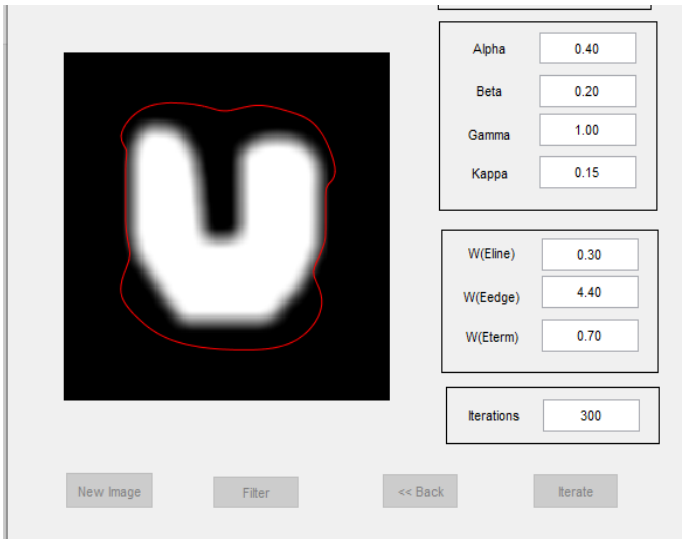


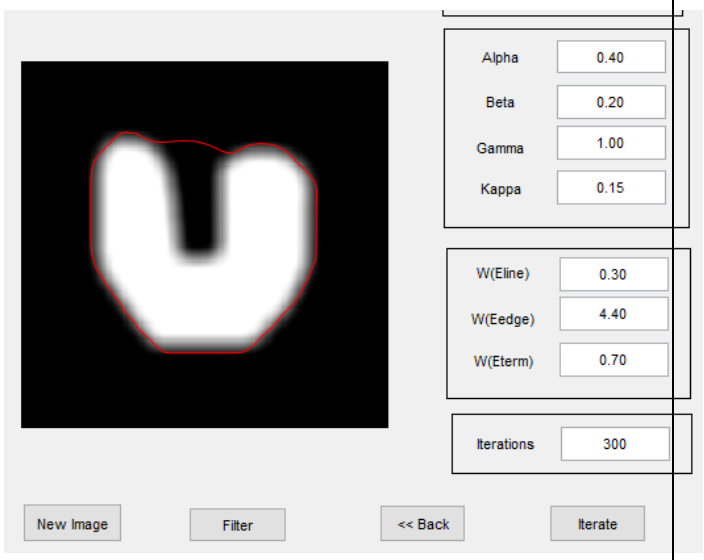
we have better result with this condition.





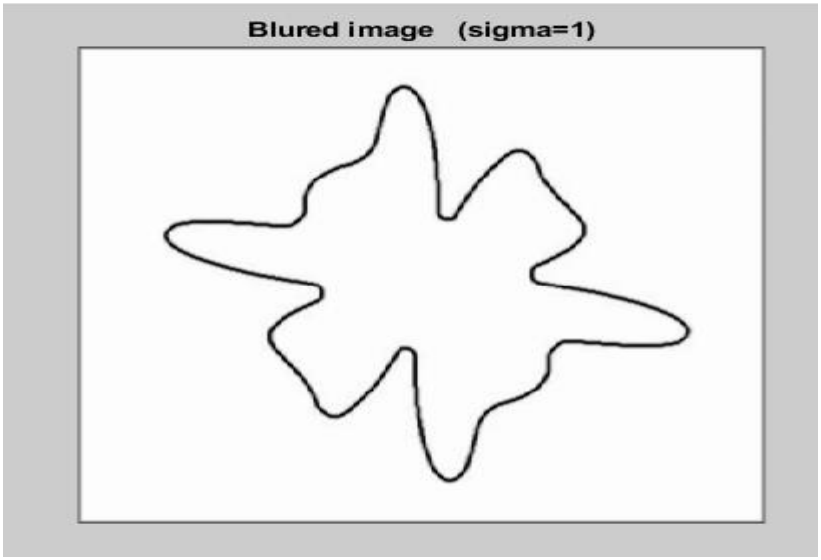
another one increase W (Edge):



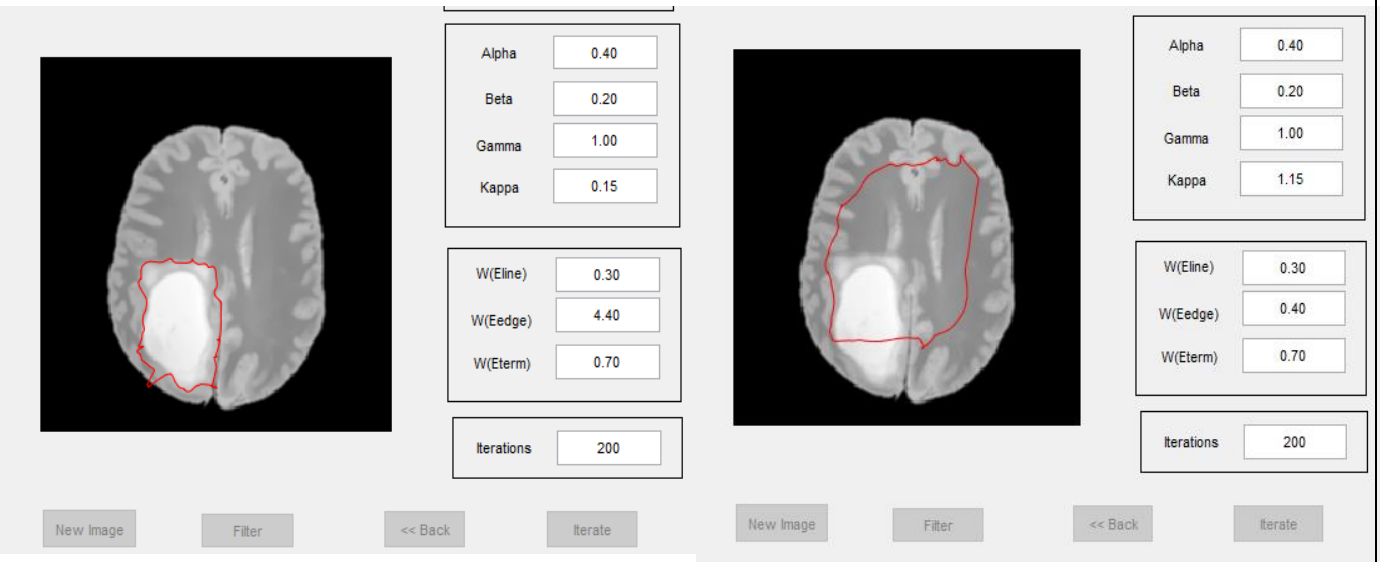


in this case we can't cover hole.

GVF:



b) for Mri image:



in this image every counter we choose with different parameters, we can't segment every class. We get the same result with another Mri images.

Question 3:

a) Due to potential existence of noise, bias field, and partial volume effect, segmentation of brain images remains challenging. the original FCM is very sensitive to noise and the accuracy of clustering in the presence of noise and image artifacts will decrease. To overcome this problem, modified the objective function by adding a term for the spatial information of neighboring pixels. This new algorithm called FCM_S, but this algorithm is computationally expensive.

After that, develop algorithm to FCM_S1 & FCM_S2.

Then, proposed a Gaussian kernel-based FCM method with the parameter ηj calculated in every iteration to replace α for every cluster. Similar to FCM S1 and FCM S2, this method has two forms: GKFCM1 and GKFCM2. These algorithms has to iterate many times to converge so To tackle the problem of parameter adjustment, Krinidis and Chatzis [13] proposed the FLICM algorithm with a fuzzy factor that combined both spatial and grayscale information of the neighboring pixels.

To enhance the FLICM algorithm, Gong et al. [14] developed KWFLICM algorithm with a trade-off weighted fuzzy factor to control the local neighbor relationship and replaced the Euclidean distance with kernel function.

After all of the previous method we have ARKFCM:

- The Introduced Regularization Term: The parameter α is usually set in advance to control the desirable amount of contextual information. Indeed, using a fixed α for every pixel is not appropriate since noise level differs from one window to another. In addition, setting such parameter needs prior knowledge about noise which is not always available in reality. Hence, adaptive calculation of α is necessary according to the pixel being processed.
- Devising a Weighted Image. In addition to making x, respectively, the grayscale of average/median filter of the original image, x can also be replaced with the grayscale of the newly formed weighted image ξ:
- . Measuring Distance Using Kernel Function. The Euclidean distance metric is generally simple and computationally inexpensive, but it is sensitive to perturbations and outliers.
- The Proposed Framework. The proposed adaptively regularized kernel-based FCM framework is denoted as ARKFCM

So The main steps for the proposed algorithms are as follows:

1. Initialize threshold ε = 0.001, m=2, loop counter t=0, V, and $u(0)$.
2. (2) Calculate the adaptive regularization parameter φi.
3. (3) Calculate xi for ARKFCM1 and ARKFCM2 or ξ for ARKFCMw.
4. (4) Calculate cluster centers V (t) j using $u(t)$ as in (19).
5. (5) Calculate the membership function $u(t$+1) with (18).
6. (6) If max $\|u(t$+1) − $u(t)\|$ 100 then stop; otherwise, update t=t+1 and go to step (4).

b) in this method we have this function:

- **Rawread :**
  RAWREAD Read a Portable Bitmap file, or a raw file and read two type image:
  1. If we have .raw, reads a "raw" image file
  2. If we have .pgm, reads a "pgm" (portable gray map) image

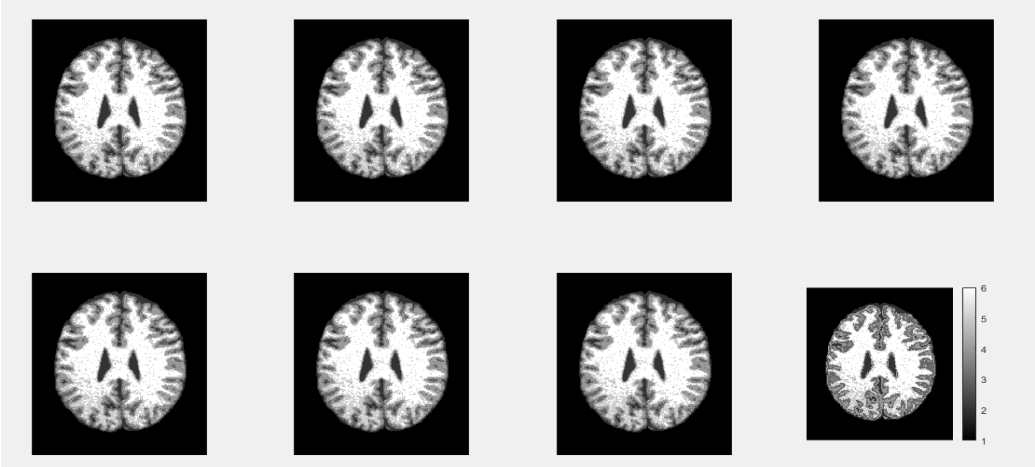
  returns both the image and a color map, so that will display the result with the proper colors.
- **ARKFCM:**
  This function is ARKFCM clustering main function.
  As input, we give it the image, number of cluster, weight associated each pixel, Size of the local window, weighting exponent on each fuzzy membership, Stopping threshold.
  And as output we have:
  1. The converged membership function
  2. Number of iterations to converge
  3. The objective function at each iteration
  4. The cluster centers after convergence
- **Demo:**
  This script to run the ARKFCM clustering on different images
- **distARKFCM:**
  This function calculates the distance between cluster centers and the pixels using GRBF Kernel.
  We have these element as input:
  1. center is the cluster center vector
  2. the input image
  3. the kernel width

  dist is an array with distance from every pixel to each cluster center as output.

  This function used to determine which class center each pixel belongs to.
- **gaussKernel:**
  This function calculates the Gaussian RBF kernel used as distance metric
  We have these element as input:
  1. the cluster center
  2. the input image
  3. the kernel width of GRBF

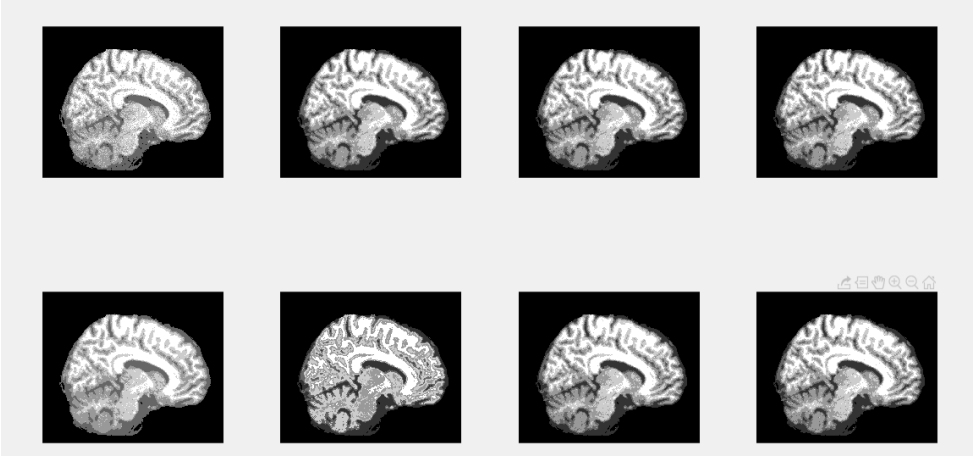
  and the output kernel used to calculate the distance

c) first we show the results.
   The first image:
   with 7% noise and 20% grayscale non-uniformity



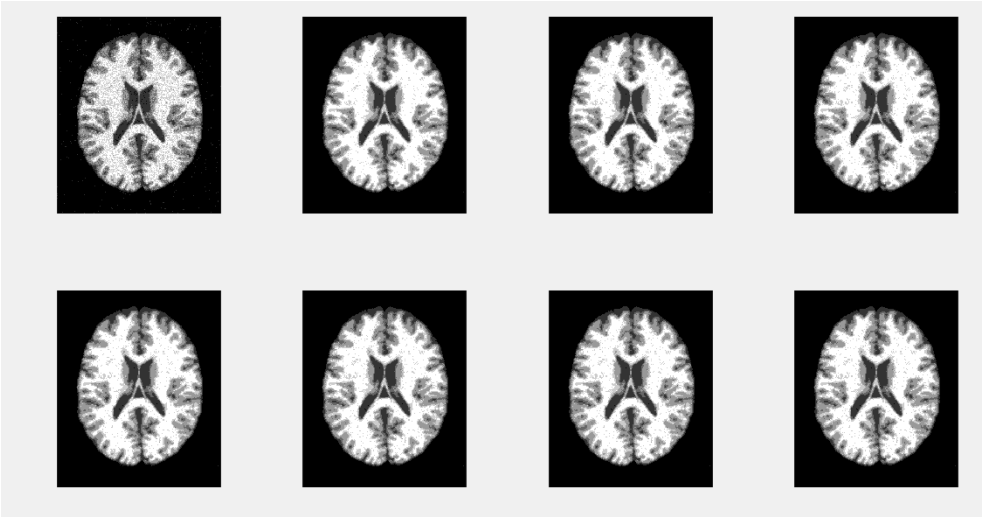
The second image:

with 7% noise and 20% grayscale non-uniformity



The third image:

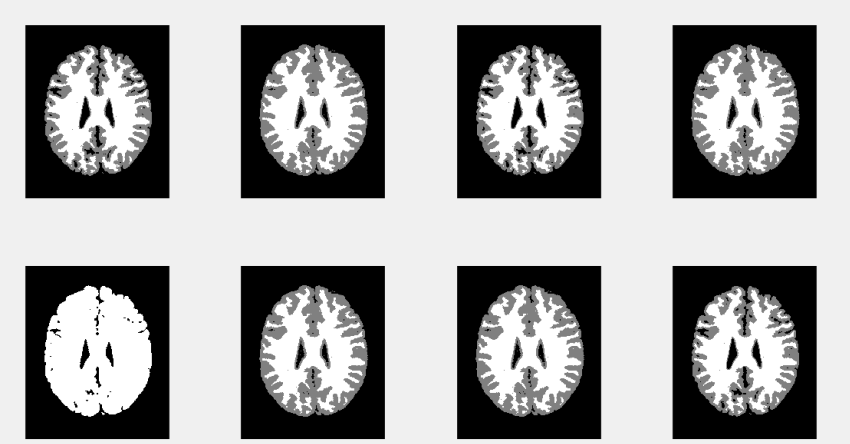
with 10% Rician noise



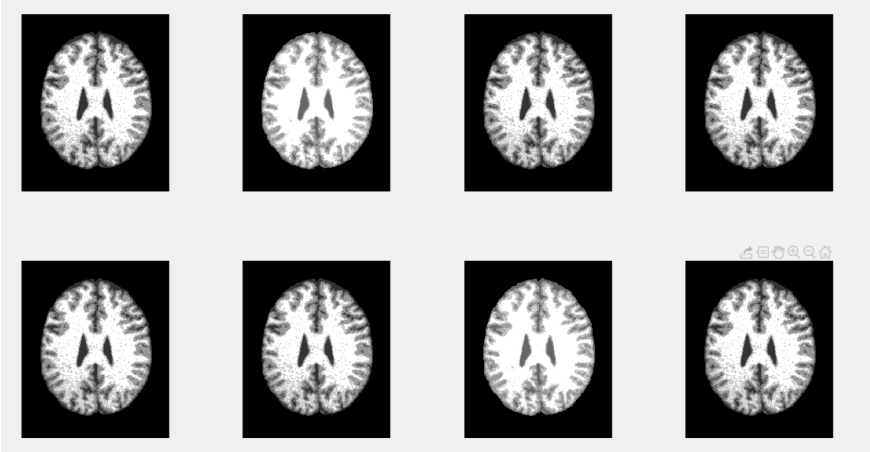
This method very good for noisy image and we can see in the third example (fig..) can remove noise very well and segment image.

d)

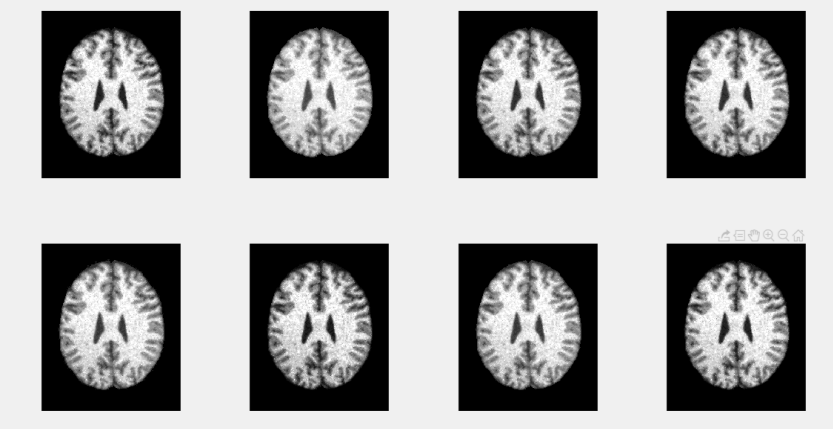
First image, Cnum=3:



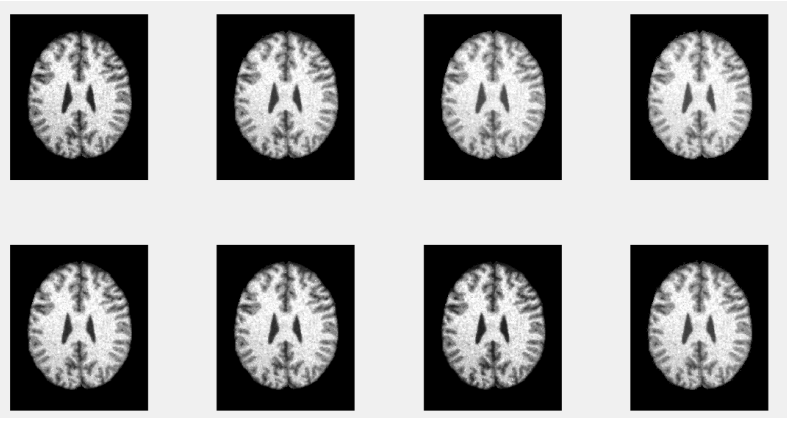
First image, Cnum=5:



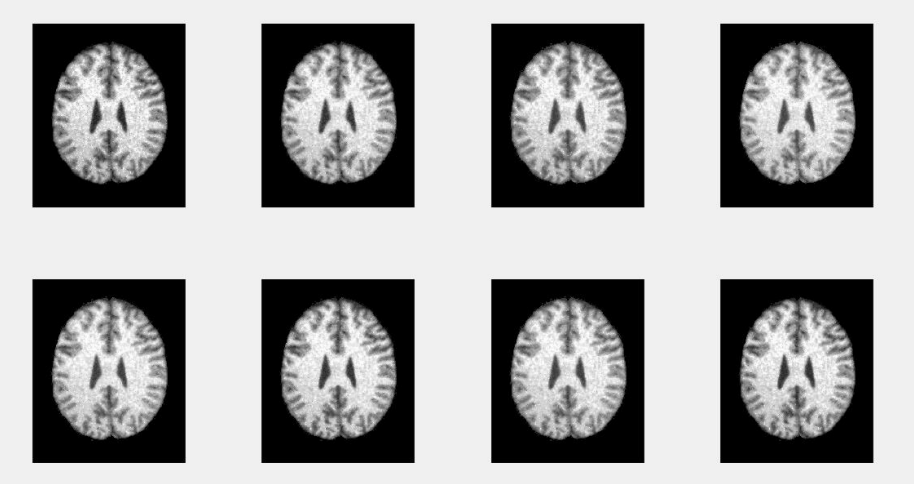
First image, Cnum=16:



First image, Cnum=25:



First image, Cnum=100:

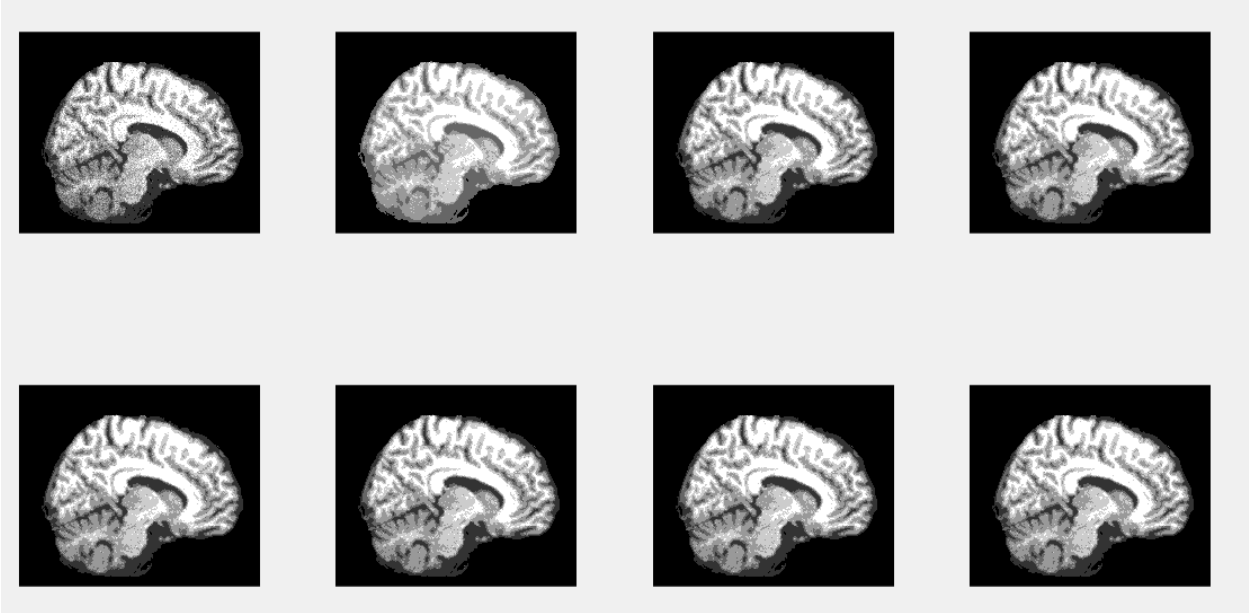


As we expect, if we increase the number of classes, we get to the original photo.

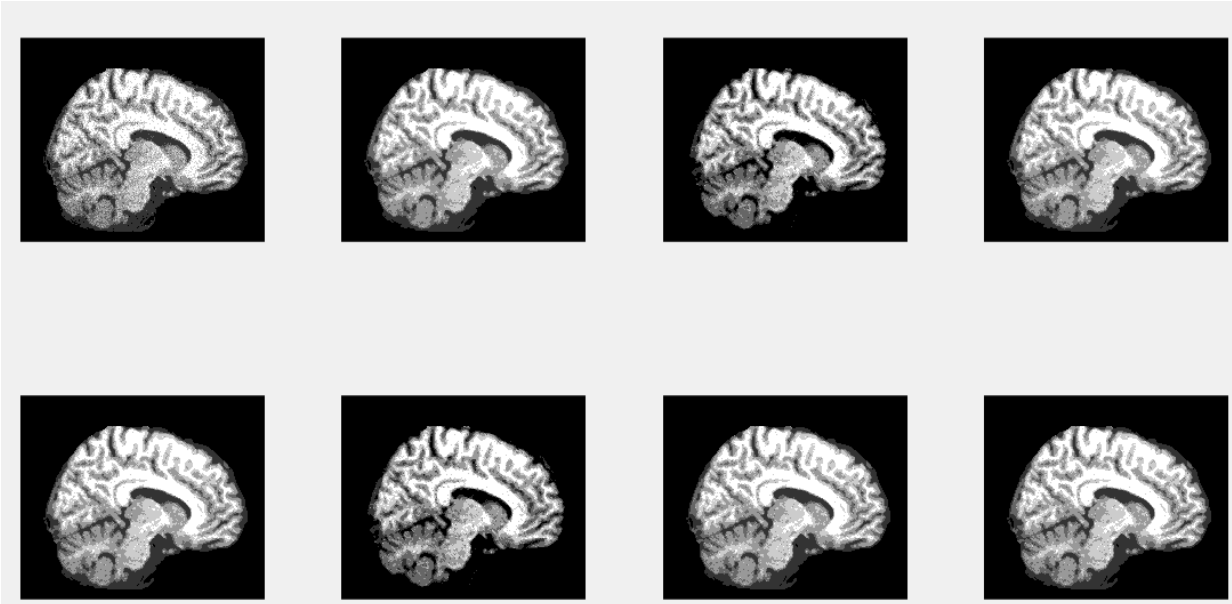
For example, if the number of classes is 100, the result very bad for segmentation.

If the number of classes is about 6 to 15, I think have a good segmentation in this case.

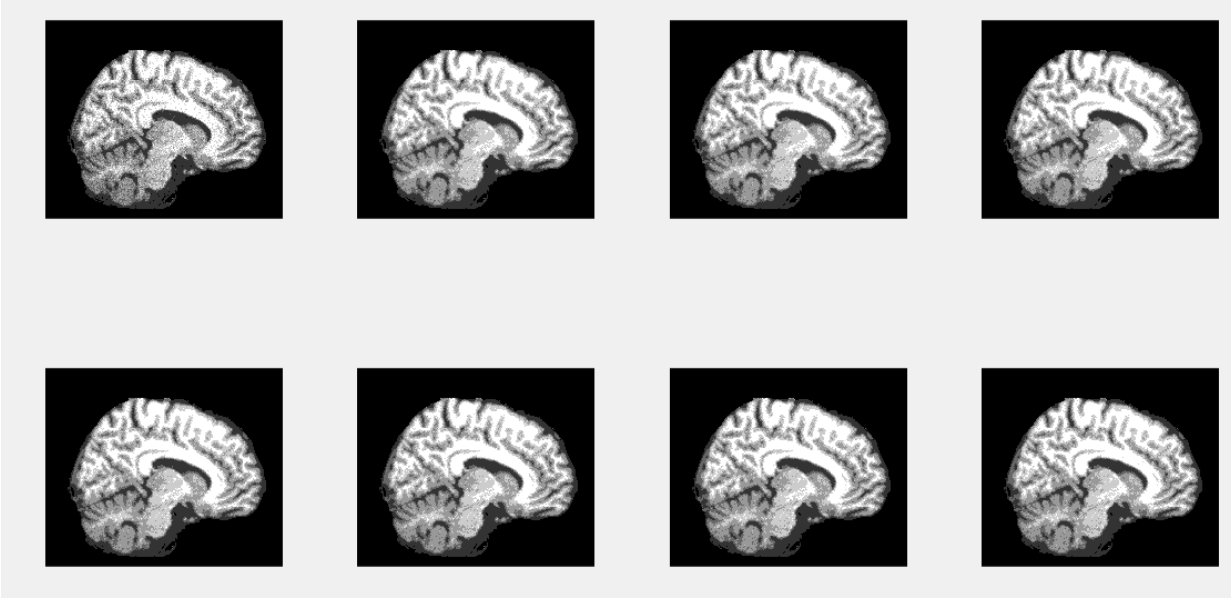
d) Cnum=6 , opt=’average’:



Cnum=6 , opt='median':

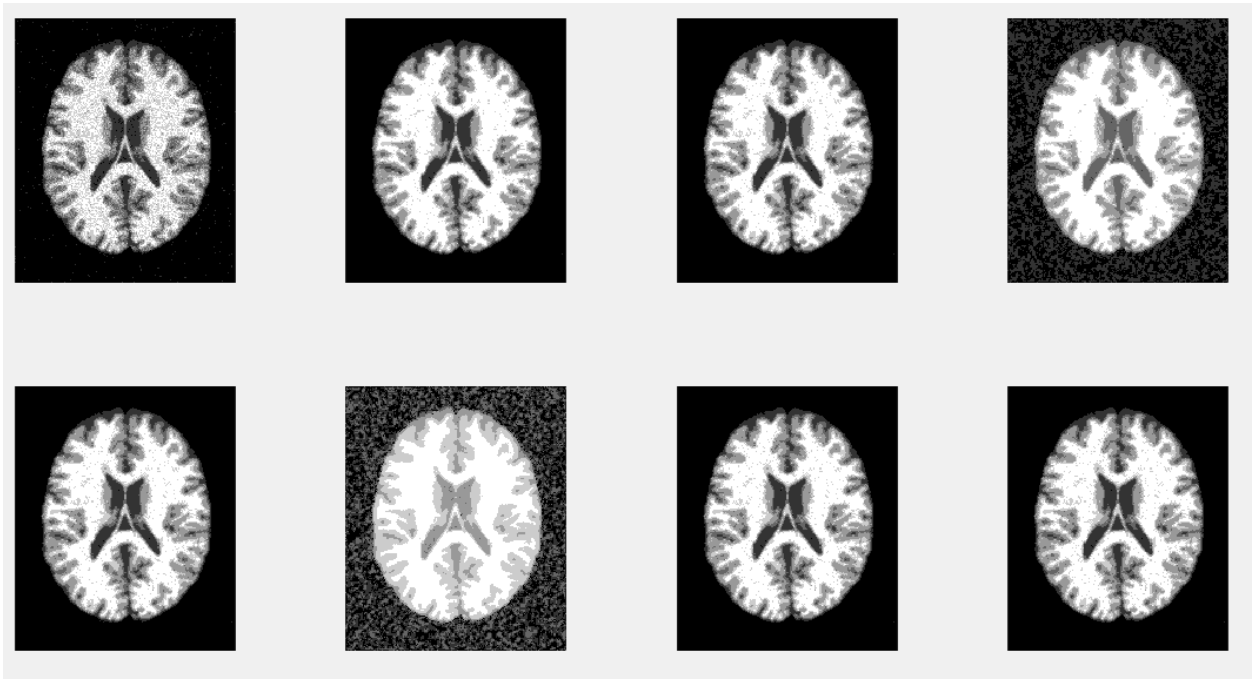


Cnum=6 , opt='weighted':

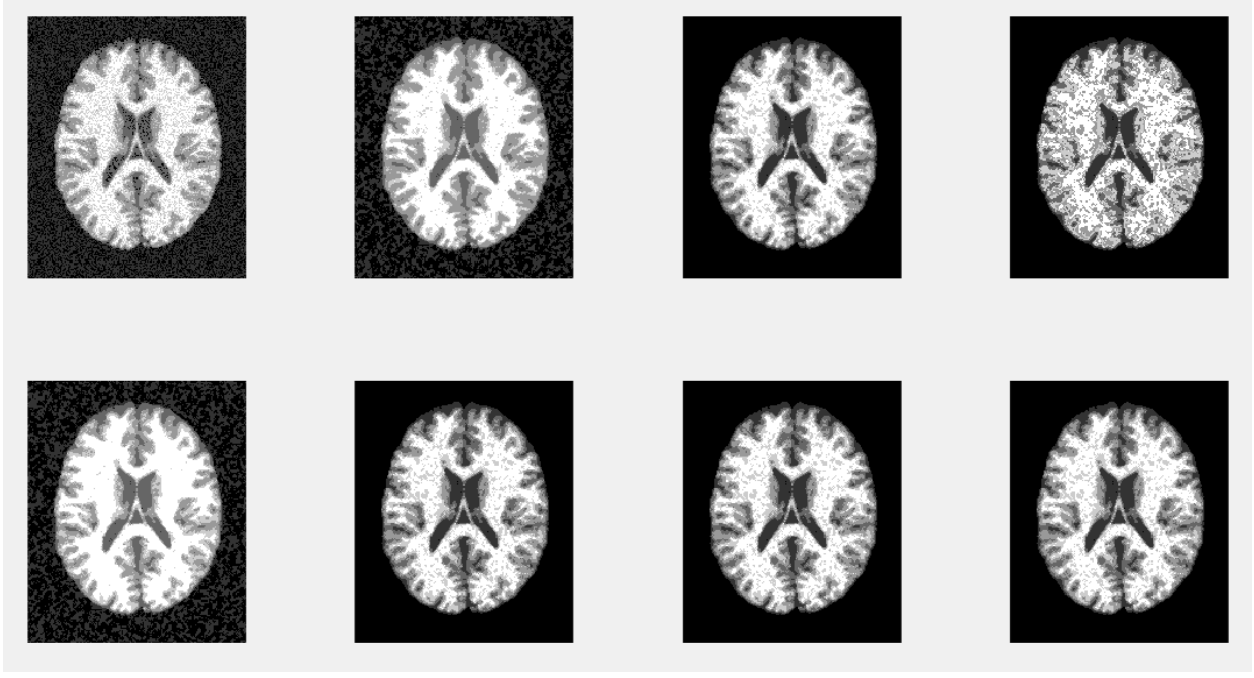


In this case we don't see difference between the results. Try with another image:

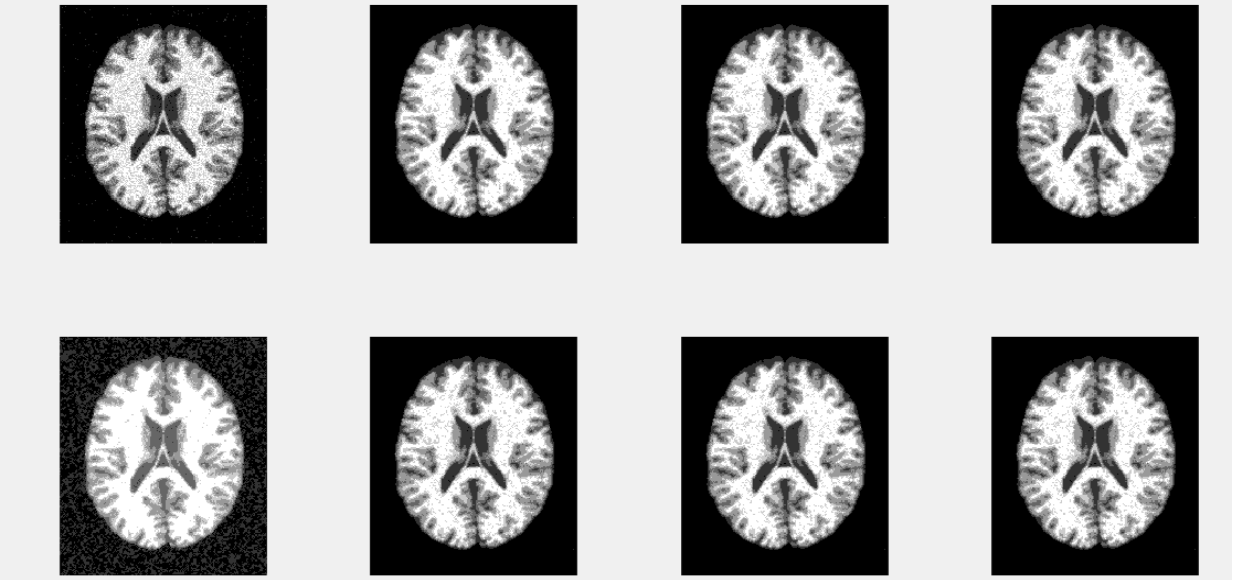
Cnum=6 , opt=’average’:



Cnum=6 , opt=’median’

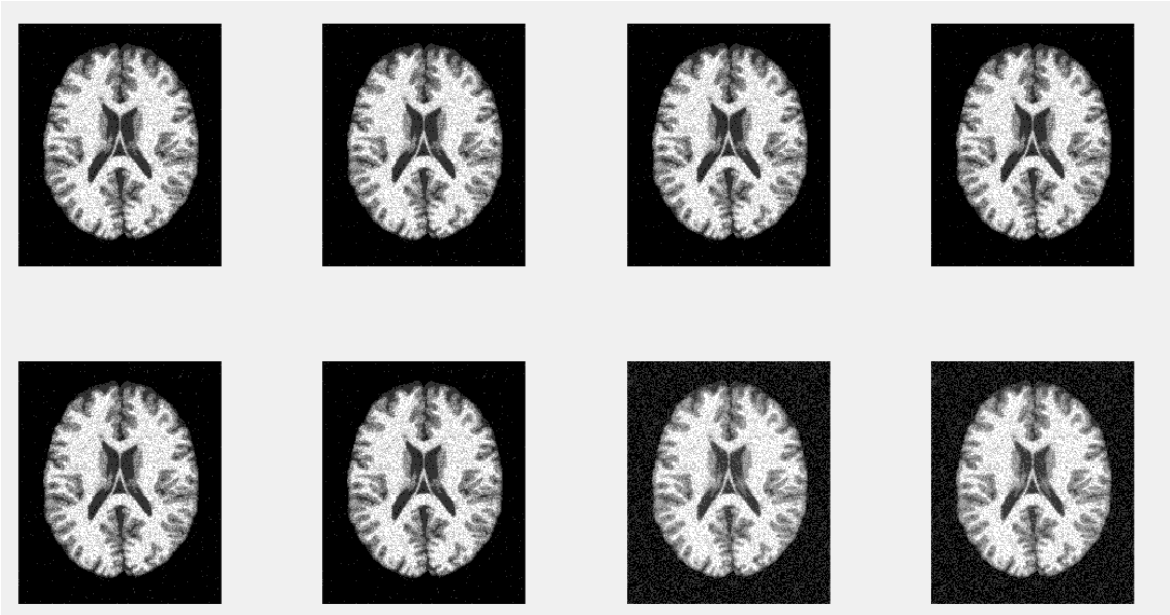


Cnum=6 , opt='weighted'

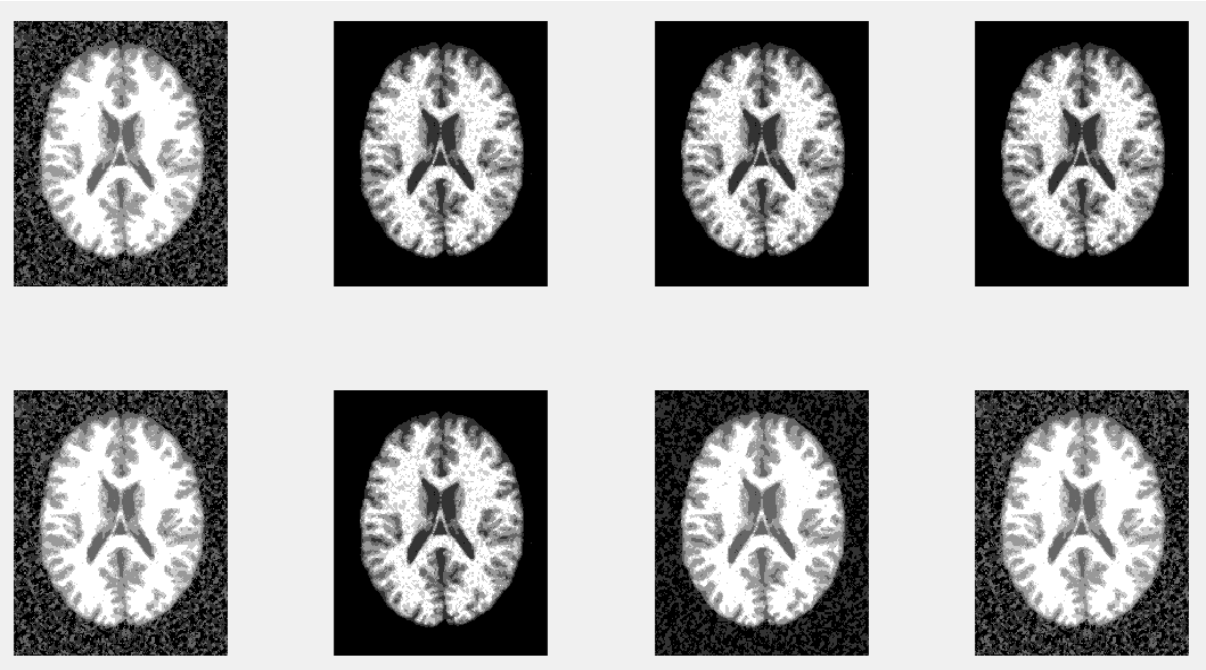


And in this case, we can see a little difference between results and It is not enough to see the impact of change of filters.

e) Winsize=1:



Winsize=29:



As we can see in both case we didn’t get good result.

Finish ☺