A Project Report
on

Segmentation of Brain images using Particle Swarm Optimization Algorithm

Submitted for the fulfillment
of
undergraduate B.Tech program
in
Computer Science & Engineering

By

Roshan Shaw

Under the supervision of

Dr. Sudip Kumar Adhikari
Assistant Professor
Department of Computer Science & Engineering
Cooch Behar Government Engineering College

Department of Computer Science & Engineering
Cooch Behar Government Engineering College
Vill – Harinchawra, P. O.- Ghughumari
Cooch Behar – 736170

2020

ACKNOWLEDGEMENT

The work summarized in this report, explores Segmentation of Brain image using Particle Swarm Optimization Algorithm which works in really efficient way (both in case of accuracy & time consumption).

This is a combined endeavor of a number of people who directly or indirectly helped us in completing our work. This documentation would never have been more educative and efficient without the constant help and guidance of our guide Dr. Sudip Kumar Adhikari (Assistant Professor of Department of Computer Science & Engineering, Coochbehar Govt. Engineering College). We would like to thank him for giving us the right guidance and encouraging us to complete this work within time.

We would also like to express our heartiest gratitude to Dr. Sourav De (Head of the Department of Computer Science & Engineering, Coochbehar Govt. Engineering College).

Last but not the least, we would like to express our gratitude to the Office staffs of Department of Computer Science & Engineering, Coochbehar Govt. Engineering College for their constant cooperation without which we would not be able to finish my work.

_____

Roshan Shaw

CERTIFICATE OF ORIGINALITY

The project entitled "Segmentation of Brain images using Particle Swarm Optimization Algorithm" has been carried out by ourselves in partial fulfillment of the degree of Bachelor of Technology in Computer Science & Engineering of Coochbehar Government Engineering College, Coochbehar, under Maulana Abul Kalam Azad University of Technology during the academic year 2019 - 2020

While developing this project no unfair means or illegal copies of software etc. have been used and neither any part of this project nor any documentation have been submitted elsewhere or copied as far in our knowledge.

Name: Roshan Shaw
University Roll No.: 34900116023

CERTIFICATE OF APPROVAL

This is to certify that the project entitled "Segmentation of Brain images using Particle Swarm Optimization Algorithm" has been carried out by Roshan Shaw under my supervision in partial fulfillment for the degree of Bachelor of Technology (B.TECH) in Computer Science & Engineering of Coochbehar Government Engineering College, Coochbehar affiliated to Maulana Abul Kalam Azad University of Technology during the academic year 2019-2020.

It is understood that by this approval the undersigned do not necessarily endorse any of the statements made or opinion expressed therein but approves it only for the purpose for which it is submitted.

Dr. Sourav De
Associate Professor and HOD
Department of Computer Sc. & Engg.

Dr. Sudip Kumar Adhikari
Assistant Professor and Supervisor
Department of Computer Sc. & Engg.

CONTENT

INTRODUCTION

Digital Image Processing is processing of images that are digital in nature by a digital computer. The process of dividing into multiple segments of a digital image is called as Digital Image segmentation. It is the most complex and important tasks of image processing system. This process is performed to represent the image in a clear way. It is used to partition an image into separate segments, to different real-world objects. It is a censorious step towards content stud and image sense. The result of image segmentation process is a collection of segments which merge to form the entire image.

In medical image segmentation, dissimilar image components are used for analysis of different structures, tissues and pathological regions. However, manual segmentation is a challenging and time consuming task and prone to error. Its success heavily depends on the expertise of the physician and thereby errors occur in the results and also augmented due to the intrinsic nature of the medical images. For example, in magnetic resonance imaging (MRI) brain images, different tissue regions have poorly outlined or blur edges and slender topological structures. This makes the process of MRI brain image segmentation into a challenging task. Moreover, accurate segmentation is also necessary for detecting and analysing the diseased regions in medical images. Therefore, computer aided segmentation is very significant to achieve effective results.

In India alone, it is estimated that 24142 new cases of brain cancer were diagnosed in 2018. Surgery is the most familiar treatment for brain tumors, radiation and chemotherapy may be used to slow the growth of tumors that cannot be physically removed. Magnetic Resonance Imaging (MRI) provides detailed images of the brain, and is one of the most accepted tests used to diagnose brain tumors. Overall, brain tumor segmentation from MR images can have great impact for improved diagnostics, growth rate prediction and treatment planning. While some tumors can be easily segmented, others are much more difficult to localize. These tumors are often diffused, poorly contrasted, and extend tentacle-like structures that make them difficult to segment. Another fundamental difficulty with segmenting brain tumors is that they can appear anywhere in the brain, in almost any shape and size.

In last decades, biomedical and medical image processing have become one of the most challenging fields of image processing and pattern recognition. Brain segmentation consists of separating the different tissues: gray matter (GM), white matter (WM), cerebrospinal fluid (CSF) and probably abnormal (tumor) tissue. The target of segmentation of MR Brain images is to study anatomical structure, identify region of interest: locate tumor and others abnormalities, measure tissue size to follow the evolution of tumor and help in treatment planning prior to radiation therapy.

Real world image segmentation problems have many objectives such as decrease the features, reduce overall deviation, decrease the error rate of the classifier or maximize connectivity, etc. Particle swarm optimization (PSO) is a changing computation technique developed by Kenney and Eberhart in 1995[1]. PSO is a population-based imaginary way for solving continuous and discontinuous optimization problems. In PSO, simple software agents, called particles, move in the search space of an optimized problem.

LITERATURE SURVEY

A number of algorithms for segmentation of medical images such as, MRI images have been presented in the past. Among them, intensity thresholding, region-based segmentation, edge-based segmentation and classification-based segmentation are the most frequently used techniques.

In intensity thresholding, the threshold level is automatically determined from the gray-level histogram of the image. The distribution of intensities in medical images, especially in MRI images is usually very complex as stated above, and therefore, thresholding methods fail due to lack of determining optimal threshold. In addition, intensity thresholding methods have disadvantage of spatial uncertainty as the pixel location information is ignored. The edge-based approaches first generate interrupted (scattered) contour lines around an object of interest using some edge detection algorithms. Next, these contour lines are joined based on some similarity criteria to detect the object of region of interest (ROI). However, these methods usually require computationally expensive post-processing to obtain hole free representation of the objects. The region growing methods extend the thresholding by integrating it with connectivity by means of an intensity similarity measure. These methods starts with a seed point (pixel) for each region and during the region growing, pixels in the neighbourhood are added to the regions based on homogeneity criteria, resulting connected regions. However, they are sensitive to noise and thus less suitable for medical image segmentation.

In classification-based segmentation method, the fuzzy C-means (FCM) clustering algorithm, is more effective with considerable amount of benefits. Unlike hard clustering methods, like k-means algorithm, etc., which assign pixels exclusively to one cluster, the FCM algorithm allows pixels to have relation with multiple clusters with varying degree of memberships and thus more reasonable in real applications. Although the FCM is a very popular unsupervised clustering method, it has some serious drawbacks as it does not consider the image spatial information and is therefore very sensitive to noise and imaging artifacts. It can also generate local optimal solution due to poor initialization. In order to make the FCM algorithm more robust to noise and outliers for image segmentation, many modified fuzzy clustering approaches have been reported in the past. Pedrycz introduced a conditional fuzzy c-means-based clustering method guided by an auxiliary or conditional variable. The method reveals a structure within a family of patterns by considering

their vicinity in a feature space along with the similarity of the values assumed by a certain conditional variable. Mohamed *et al.* modified the FCM algorithm through the incorporation of the spatial information. They introduced the spatial information into the computation of similarity measure. The similarity measure is modified to drag a pixel closer to the cluster center if it is in homogeneous region. The drawbacks of this algorithm are its sensitivity to the non-descriptive initial centers and its massively computational load. Ahemed *et al.* introduced the local grey level information by modifying the objective function with another similarity measure for bias field estimation and segmentation of MRI data. This method is also expensive in terms of computational time. Many researchers subsequently modified the objective functions and developed several robust FCM variants for image segmentation. These algorithms were shown to have better performance than the standard FCM algorithm. However, some of these methods depend on a fixed spatial factor which needs to be adjusted according to the real applications. In order to overcome the problem of over-smoothed edges, causes due to use of larger spatial window, adaptive selection mechanisms of the spatial parameters have been proposed. The performances of these methods are superior and are able to reduce partly the blurring effects, which arise due to use of filtering and larger spatial window.

Man *et al.* [1] developed an image clustering algorithm using Particle Swarm Optimization (PSO) with two improved fitness functions. The PSO clustering algorithm are often used to find centroids of a user specified number of clusters. Two new fitness functions are proposed. The PSO-based image clustering algorithm with the expected -fitness functions is compared to the K-means clustering. Experimental results show that the PSO-based image clustering approach, using the improved fitness functions, can perform better than K-means by generating more compact clusters and bigger inter-cluster separation.

Anita *et al.* [2] developed a PSO based methods to look cluster center within the arbitrary data set automatically with none input knowledge about the amount of present regions within the data, and their applications to image segmentation.

ASSAS *et al.* [3] introduced resonance imaging (MRI) which is very strong tool for clinical diagnostics because it allows to separate different tissues and allows multiple modalities (T1, T2) each having particular properties. The segmentation of MR Brain images is taken into account as an optimization problem and solved using evolutionary algorithms: particle swarm optimization

(PSO) and differential evolutionary algorithms. The process of segmentation is done with multilevel fuzzy thresholding. The performances of these three approaches were compared by using the fidelity criterion: the peak-to-signal-noise (PSNR) ratio. The methods adopted provide good leads to terms of accuracy and robustness. However, the PSO is the most efficient.

Samy *et al.* [4] developed a Particles Swarm Optimization (PSO) method to solve this optimization problem. The quality of segmentation is evaluated on the grounds of truth images using the Kappa index. The results show the power of the HMRF-PSO method compared to K-means and threshold based techniques. Segmentation of medical images is one among the elemental problems in image processing field. It aims to supply an important decision support to physicians. There are several methods to perform segmentation. Hidden Markov Random Fields (HMRF) constitutes a chic thanks to model the matter of segmentation. This modelling results in the minimization of an energy function.

Vasupradha *et al.* [5] introduced automated brain tumor segmentation and detection which are most important in medical diagnostics because it provides the information related to the functional structures also as potential abnormal tissue necessary to separate surgical plan. But automatic tumor segmentation remains challenging due to low contrast and ill-defined boundaries and accuracy problem. Therefore Enhanced Darwinian Particle Swarm Optimization (EDPSO) is proposed for the automated tumor segmentation which beats the disadvantages of existing Particle Swarm Optimization (PSO). This innovative method consists of 4 steps. First step is pre-processing, film artifacts and unwanted portions of MRI images are removed using tracking algorithm .Second step involves the process of removing the noises and high frequency component using Gaussian filter. Third step, segmentation is completed using Darwinian Particle Swarm Optimization and Fourth step is classification, which is completed by Adaptive Neuro Fuzzy Inference System. The performance of the proposed method is systematically calculated using the MRI brain images.

Paras *et al.* [6] proposes an approach which combines the Particle Swarm Optimization (PSO) techniques and Fuzzy C-Means (FCM) algorithm to perform image segmentation on Magnetic Resonance Imaging (MRI). The FCM algorithm has some limitations that it requires initialization of cluster centroids and therefore the number of cluster. In this, the PSO techniques is enforced to the MRI medical images for the aim of initiation of cluster centroids, which could overcome the

necessity of manual initialization of FCM algorithm. Natural Computing (NC) may be a novel approach to unravel real world problems inspired within the life itself. A diversity of algorithms had been proposed like evolutionary techniques and particle swarm optimization (PSO). This approach, alongside fuzzy c means (FCM), give powerful tools during a diversity of problems of optimization, classification, data analysis and clustering.

R. Punidha *et al.* [7] developed a brain tumor segmentation based on Particle Swarm Optimization (PSO). Particle Swarm Optimization is an optimization based computational method which is employed to optimize a drag by iteratively trying to enhance the candidate solution for a given measure of quality. PSO mainly depends on the objective function and the tuning or updating parameters such as position and velocity. The MRI image consists of film artifacts like patient's name, medical details that patient which makes the segmentation process to inefficient. To avoid such inefficiency, preprocessing can be carried out. Preprocessing may be a technique that removes the unwanted information (noise, film artifacts in MRI) present in an ingenious MRI image.

S. Mahalakhmi *et al.* [8] analyses about the detection & separation of brain tumor through Magnetic Resonance Imaging medical images using Particle Swarm Optimization. The algorithm is widely used & rapidly developed for its ease implementation. This work has stages that includes conversion, implementation, selection & extraction. The research work starts with converting the Digital Imaging & Communications in Medicine into image file format which is the 1st stage. Applying the PSO algorithm with the change in the values of n (segmentation level) is the 2nd stage. Based on the time, selecting the best resultant images is the 3rd stage. The final stage is extraction of tumor affected region with the acceptable filtering techniques. The research work takes the axial & coronal plane of the Magnetic Resonance images. This work concludes with the extraction of the resultant image, which is taken as input & using the best filtering technique the affected region is easily separated & identified efficiently.

Moussa *et al.* [9] proposed a completely unique initialization approach for the Fuzzy C-Means Algorithm supported Fuzzy Particle Swarm Optimization applied to brain MR image segmentation. The proposed method, named Fuzzy Particle Swarm Optimization for FCM uses the FPSO algorithm to urge the initial cluster centers of FCM consistent with a replacement fitness function which combines fuzzy cluster validity indices. The FPSOFCM was calculated on several

MR brain images perverted by different levels of noise & intensity non-uniformity. Experiment results show the proposed approach improves segmentation results.

PROPOSED ALGORITHM

Particle Swarm Optimization is a latest and emerging digital image segmentation techniques inspired from the nature. It has been widely used as an optimization tool in areas including tele communications, medical or biological science, computer graphics, data mining, signal processing, robotics, etc..

PSO is a swarm intelligence techniques that are used to solve optimization problems. Particle Swarm Optimization simulates the behaviors of bird flocking. It means, a group of birds are randomly searching food in an area. So there is only one piece of food in the area being searched .All the birds do not know where the food is, but they know how far the food is in each iteration. So the best way to find the food is to follow the bird which is nearest to the food. Flocking is a behavior, it exhibited when a group of birds called a flock are foraging.

Particles in particle swarm optimization is updated by following two best values:

1. **Pbest** – particles keeps track of its coordinates in the solution space. They are associated with best fitness that has achieved so far by that particle. It is called pbest or Personal best value.

2. **Gbest** – Gbest is tracked by the personal best value is the best value obtained so far by any particle in the neighbor of the particle. This is called Gbest or global best.

Particles tries to modify their positions using

i) their current positions,

ii) their current velocities,

iii) Distance between the current position & personal best value(pbest)

iv) Distance between the current position & Global best value(Gbest)

After that find two best values & the particle updates its velocity & positions with the following equations

v[] = v[] + c1 * rand() * (Pbest[] - ppresent[]) + c2 * rand() * (Gbest[] -ppresent[]) ……. (1)

present[] = present[] + v[].....(2)

Here v[] is the particle Velocity & Present[] is the current particle .

rand() is a random no between (0,1) . c1 & c2 are learning factors. Usually c1 = c2 = 2.

The concept of modification of a searching point by Particle Swarm Optimization algorithm is depicted in Fig 1.

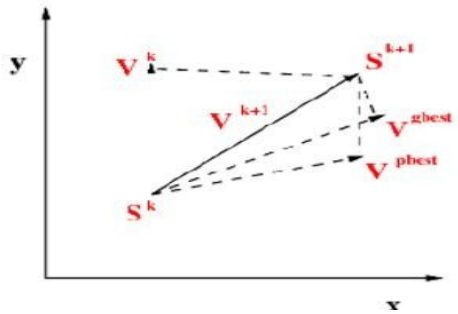


Figure 1:Modification of searching point by PSO

The pseudo code of the PSO procedure is as follows:

For each particle

    Initialize particle

End

Do

  For each particle

    Calculate fittness value

     If fitness value is better than best fitness value or pbest in history

    set current value as the new Pbest

End

Choose the particle which have the best fitness value of all particles as the Gbest

For each particle

    Calculate particle velocity according equation (1)

    Update particle position according equation (2)

End

Method

The following notations are used:

Np denotes the no of image pixels to be clustered

Nc denotes the no of clusters to be formed

Zp denotes the p-th pixel

mj denotes the mean of cluster j

Cj denotes the subset of pixel vectors that form cluster j

|Cj| denotes the number of pixels in cluster j

In this paper, the PSO based image clustering algorithm proposed. A single particle represents the Nc cluster means. Each particle xi is constructed as xi = (mij, …, mij, …, miNc) where mij refers to the j-th cluster centroid vector of the i-th particle. The quality of each particle is measured by the fitness function. The PSO-based clustering algorithm can be summarized below:

1. Initialize each particle to contain Nc randomly selected cluster means.

2. For t = 1 to tmax (maximum number of iterations)

(a) For each particle i

- For each pixel zp

Calculate d(zp, mij) for all clusters Cij

Assign zp to Cij where

$$d(z_p, m_{ij}) = min_{\forall c=1,\ldots,N_c}\{d(z_p, m_{ic})\}$$

d(zp, mij) represents the Euclidean distance between the p-th pixel zp and the centroid of j-th cluster of particle i.

- Calculate the fitness function f(xi(t),Z) where Z is a matrix representing the assignment of pixels to clusters of particle i.

(b) Update the personal best and the global best positions.

(c) Update the cluster centroids using Equations (1) and (2).

The fitness function proposed in [8,10] uses the following three evaluation criteria: Mean error, intra-cluster distance and inter-cluster separation. The Mean Error is defined below:

$$ME = \frac{\sum_{Zp \in Np} \|Zp - Np\|}{N}$$

where d(zp,mj) represents the Euclidean distance between the pth pixel zp and the centroid of j-th cluster mj.

The intra-cluster distance is measured by which is defined in [8,10] as

$$\bar{d}_{\max}(Z, x_i) = max_{j=1,\ldots,N_c}\{\sum_{\forall z_p \in C_{ij}} d(z_p, m_{ij})/|C_{ij}|\}. \quad (4)$$

In Equation (4), Z is a matrix representing the assignment of pixels to clusters of particle i. A smaller value of means that the clusters are more compact.

Another measure of quality is the inter-cluster separation. It is measured by the minimum Euclidean distance between any pair of clusters and is defined below:

$$d_{min}(x_i) = min_{\forall j1, j2, j1 \neq j2}\{d(m_{ij_1}, m_{ij_2})\}. \quad (5)$$

The above three criteria have been used by [10] to form the fitness function as shown in Equation

$$f1(xi, Z) = w1 * dmax(Z, xi) + w2 * (Zmax - dmin(xi)) + w3 * ME$$

where w1, w2 and w3 are user defined constants and determine the relative weights of intra-cluster distance (dmax), inter-cluster separation (dmin) and Mean error (ME) in the fitness function. zmax is the maximum pixel value in the image set, which is 255 for 8-bit grayscale image used in this

paper. One objective of the fitness function in Equation (6) is to minimize the intra-cluster distance (dmax) and the Mean error (ME). This will make the clusters compact. Another objective is to maximize the inter-cluster separation (dmin) which means the clusters are well separated.

IMPLEMENTATION OF THE PROPOSED ALGORITHM

In this project we use several packages and libraries, some are extensively used and some are minimally used. We categorize them in two types.

PRIMARY

PYTHON: We use PYTHON programming language throughout the project. It is very simple, robust object oriented and user friendly. Very complex algorithm can be implemented very easily in python. There are many user friendly library available in python for beginner and most of them are free to use. So we choose python as our main programming language for this project.

NUMPY: This is very useful for numerical calculation. It provides Array class which is 50 times faster than traditional python list. It also provides many methods which helps in matrix .We can create array with as many dimensions we want with it. So this is very helpful for us.

MATPLOTLIB: Matplotlib is very useful for plotting data as graph. It support many graph. In our project we mainly focus on PYPLOT module of MATPLOTLIB .We mainly use it for image visualization and data visualization.

RANDOM: Random module has some useful features. We mainly use random sample for taking a sample from a set or list of predefined size, and random for generating a fraction between 0 and 1.

MATH: This module has some useful function like math.sqrt, math.log etc.

SECONDARY:

Here we discuss which dataset we use in our project.

BRAINWEB DATA SET: This dataset provides us a very good quality brain images. It provides 3d image of brain. Number of pixels in each 3 dimensions are x-space : 181, y-space : 217, z-space : 181. Other quality factors are – Modality=T1, Protocol=ICBM, Phantom_name=normal, Slice_thickness=1mm, Noise=3%, INU=20% .

BRAINWEB OUTPUT DATASET: We use this dataset as ground truth to compare our output with actual output and find out our accuracy. This dataset cluster the image with 10 areas. They are – Background, CSF, White Matter, Gray Matter, fat, Muscle/Skin, Skin, Skull, Glial Matter, and Connective.

EDITOR:

JUPYTER NOTEBOOK: We use Jupyter notebook because it is fast, user friendly and one can run python code on it without creating an python file. Its command-line interpreter is very good. So overall its a good editor to start with.

For all the experiments, the following parameters are used for PSO-based clustering:

- Number of particles = 10
- Number of iterations for termination = 20
- Number of clusters = 4
- Acceleration constants c1 and c2 = 2

The number of particles used is problem-dependent. The common choice of number of particles varies from 20 to 50. 20 particles are used for PSO clustering as smaller number of particles reduces computation time and 20 particles provide good clustering performance when compared with K-means. The number of clusters is chosen to be 4 for PSO clustering to allow a fair comparison of their performance. For the inertia weight w, the initial weight value is 0.5 and w decreases linearly with the number of iterations. The final value is 0.4 when the termination condition (20 iterations) is reached. By linearly decreasing the inertia weight from a relatively large value to a small value through the course of the PSO run, the PSO tends to have more global search ability at the beginning of run while having more local search ability near the end of the run. The acceleration constants c1 and c2 are both set to 2. The settings of acceleration constants and the inertia weight are based on the recommendation. For fitness functions f2 and f3 in PSO clustering, each fitness function consists of three sub-objectives. The weighting of each sub-objective (w1, w2 and w3) that provides best performance is determined empirically in this paper. To eliminate the tuning of these weight values, multi-objective optimization approach can be used.

EXPERIMENTAL RESULT AND DISCUSSION

The performance of the proposed method is evaluated on the simulated images of human brain both in qualitatively and quantitatively. In the comparative study, for a fair comparison, we have included the k-means, FCM and the proposed algorithm.

Simulated MRI Brain Images

The BrainWeb simulated MRI brain images are acquired from the McConnell Brain Imaging Center of the Montreal Neurological Institute, McGill University. 9% noise, 20% inhomogeneity and 9% noise, 40% inhomogeneity of simulated T1-weighted data volumes consisting of 50 images each, where the CSF, GM and WM regions are prominent, have been collected. The image resolutions are 181×217×181 voxels and sized 1×1×1 mm.

Segmentation Accuracy (SA)

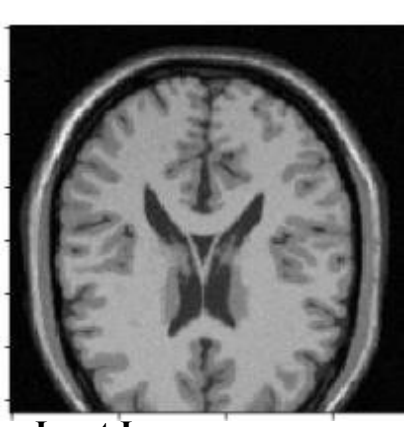
The *SA* is defined as the sum of the correctly classified pixels divided by the sum of the total number of pixels of the clustered image. The *SA* can be defined as follows:

$$\mathrm{SA} = \frac{\sum_{k=1}^{M} \mathrm{card}(A_k \cap C_k)}{\sum_{k=1}^{M} \mathrm{card}(C_k)}$$

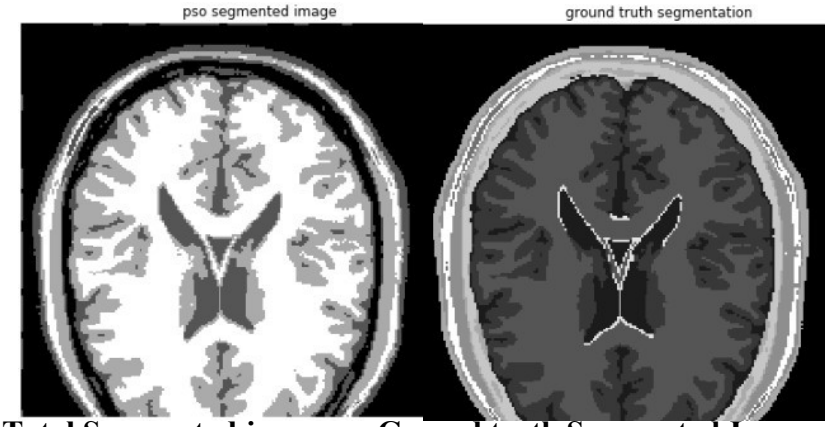
where M is the total number of pixels in a cluster, A_k is the set of pixels belonging to the kth cluster found by the algorithm, C_k is the set of pixels of the kth cluster in the ground truth image. For an ideal result, the value of *SA* will be 1, with higher values being “better”.

i) Qualitative Evaluation

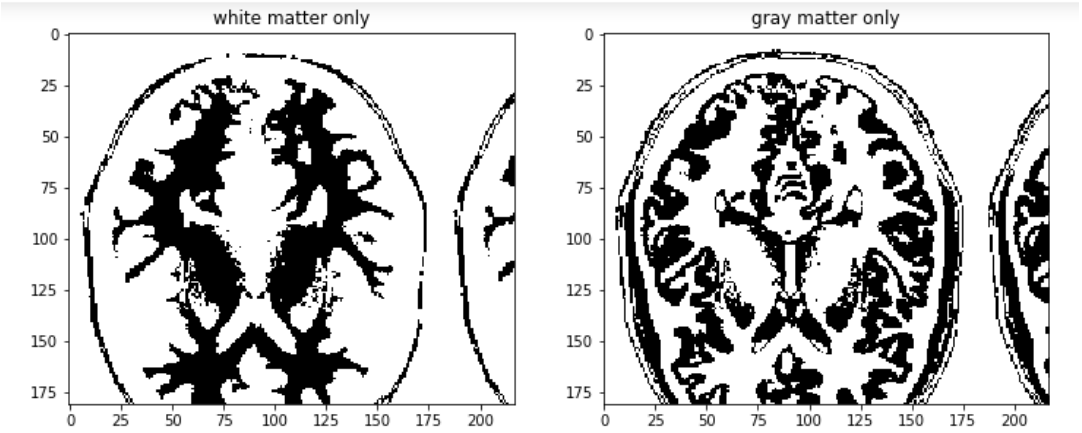
Brain Image 1:-

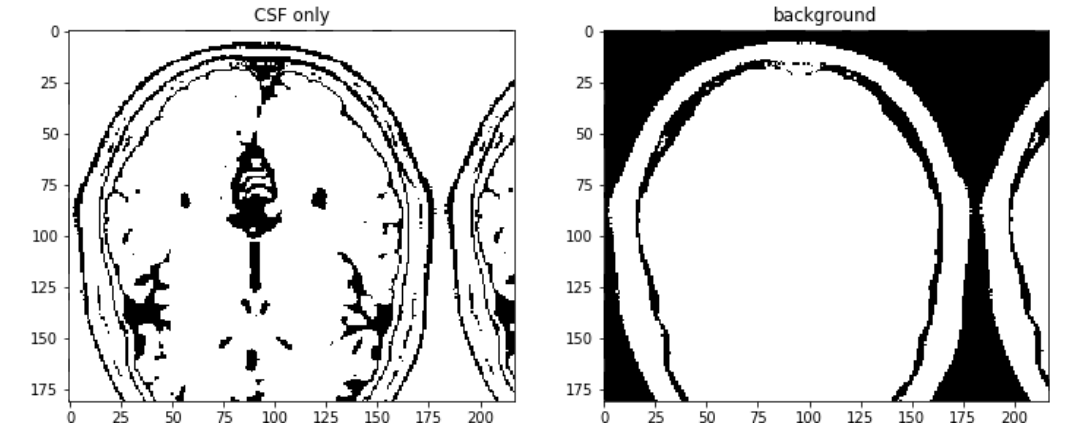


Input Image



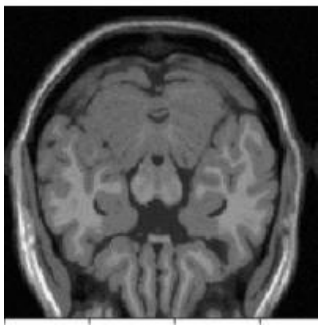
Total Segmented image Ground truth Segmented Image

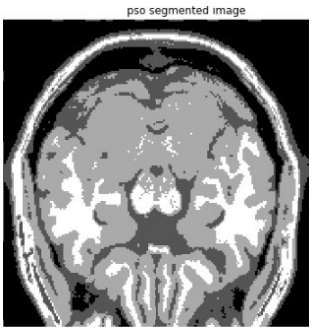


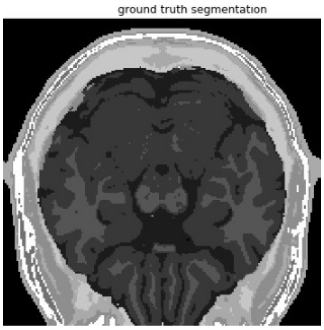



```
accuracy is 97.17553688141923%
```

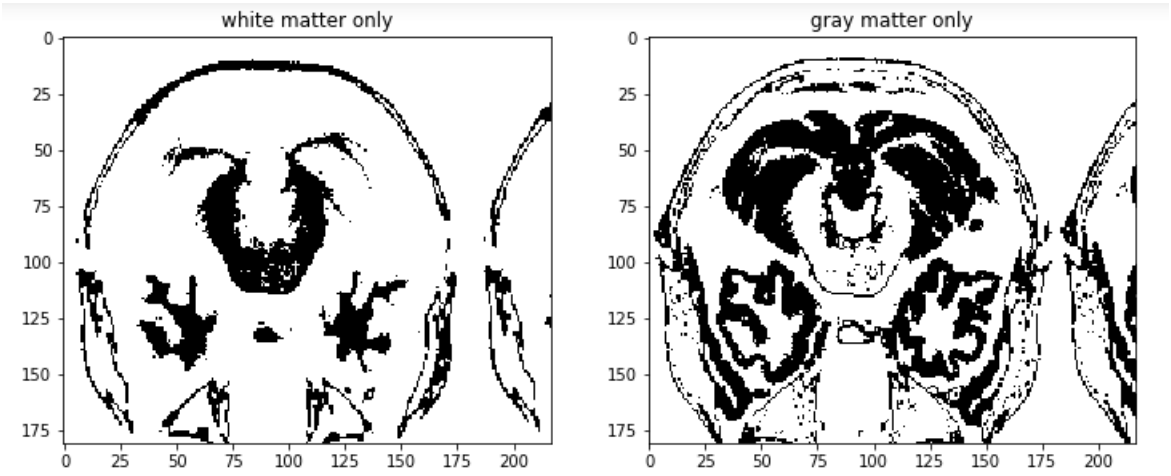
Brain Image 2:-

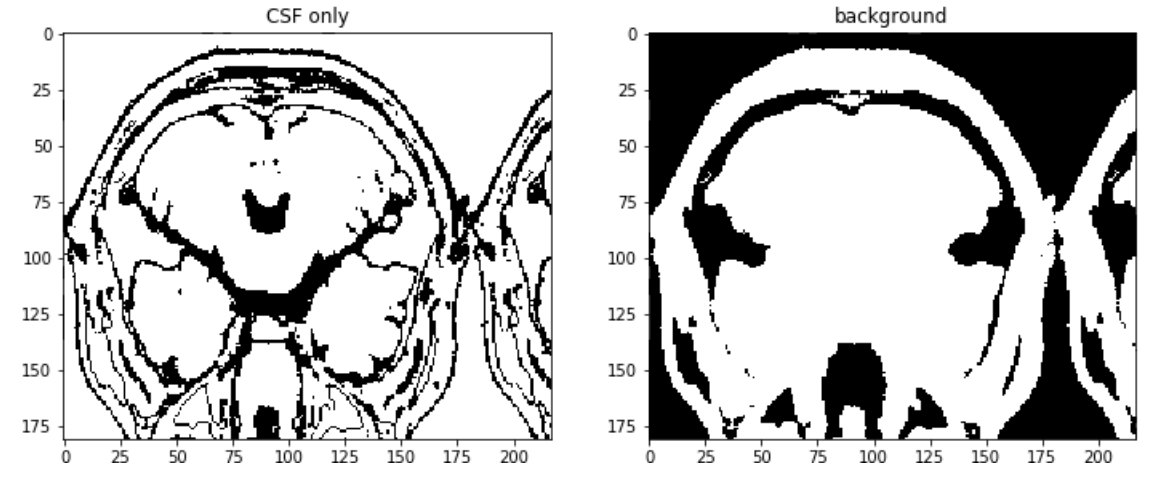






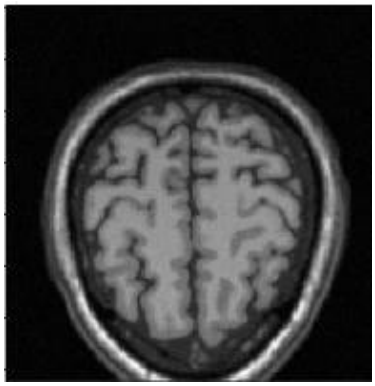
Input Image **Total Segmented image** **Ground truth Segmented Image**





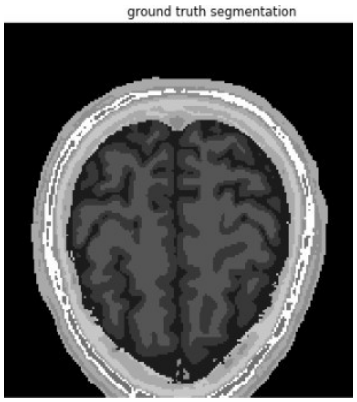
accuracy is 96.32469592808039%

Brain Image 3:-

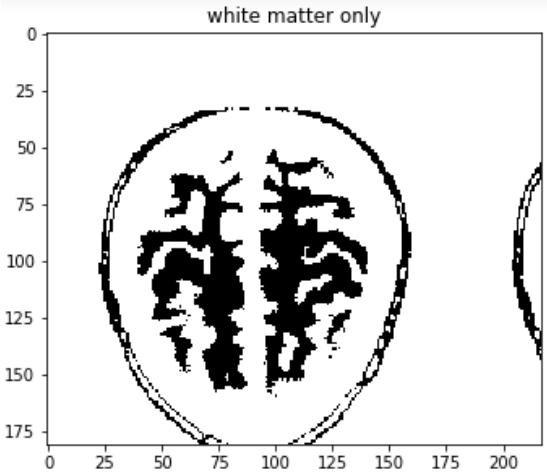


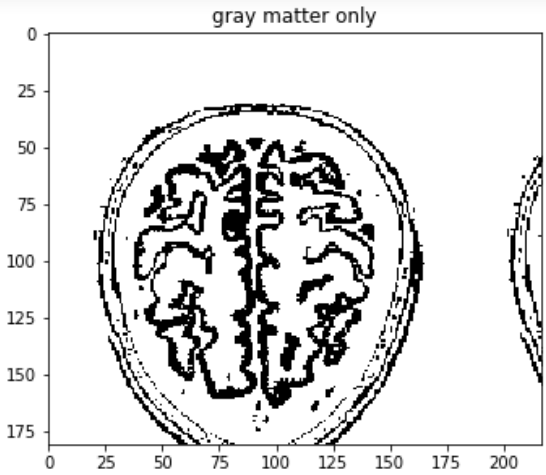
Input Image

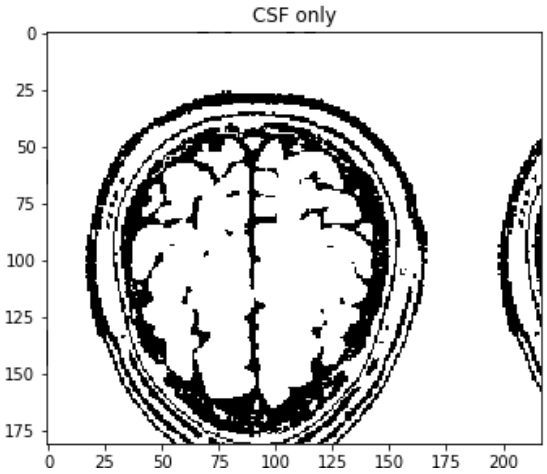
Total Segmented image

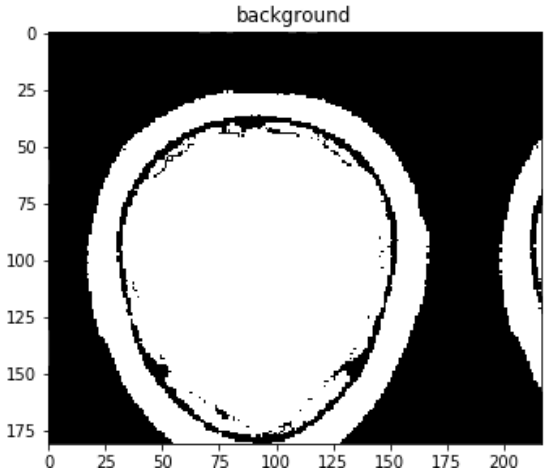


Ground truth Segmented Image



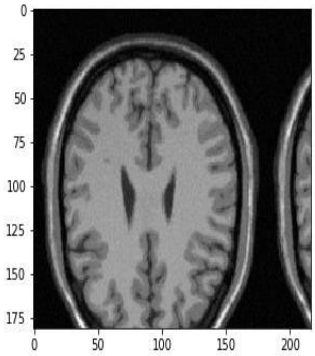


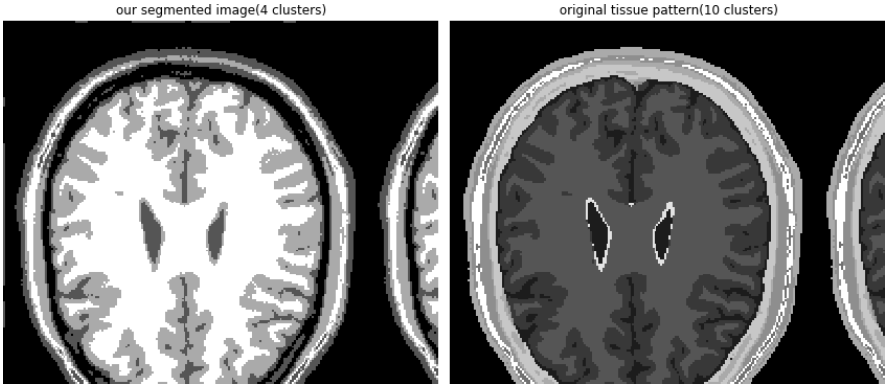




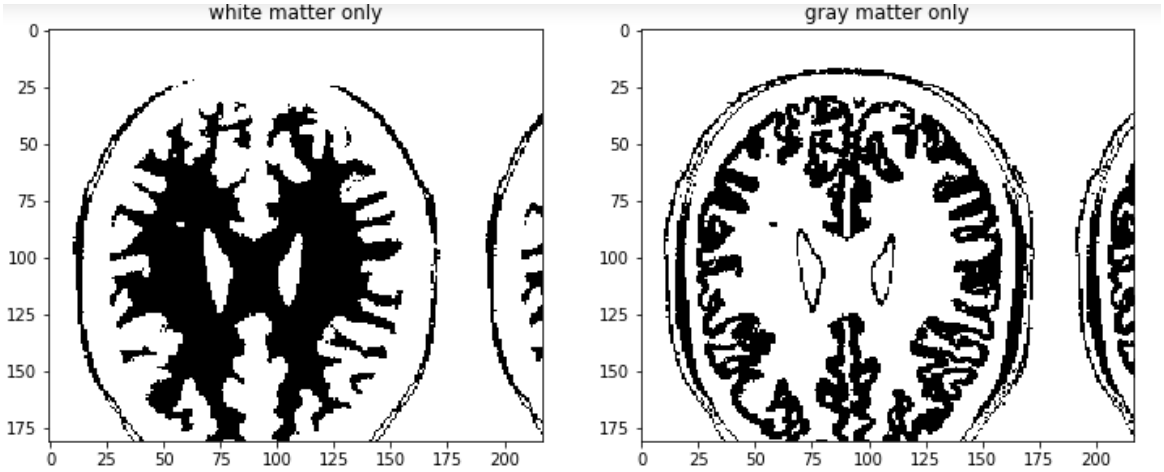
accuracy is 97.58256496227996%

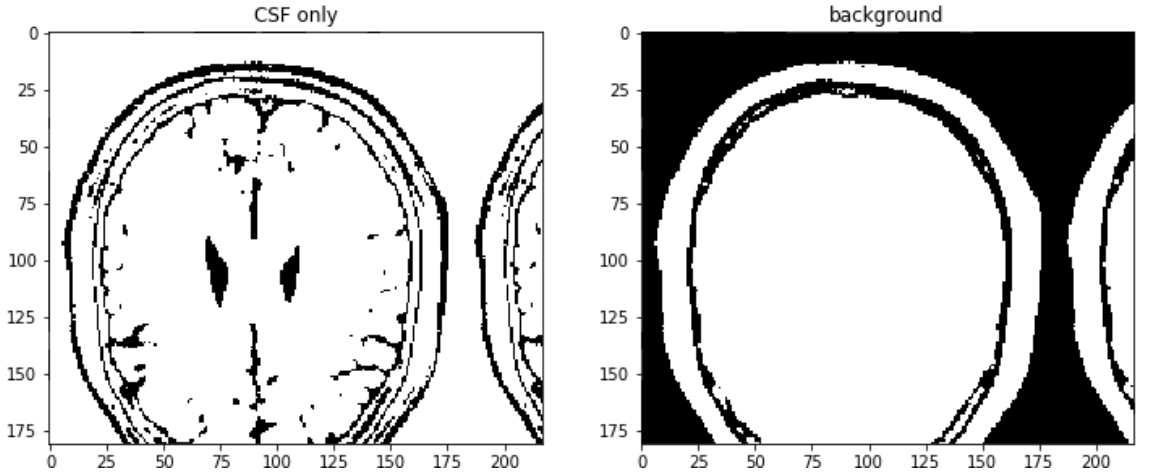
Brain Image 4:-





Input Image **Total Segmented image** **Ground truth Segmented Image**





accuracy is 96.08671077567577%

ii) Quantitative Evaluation

Table-1 Segmentation Accuracy of 4 images

Images	*Intra-Cluster Distance* d_{max}	*M.E* $ME = \frac{1}{n}\sum_{j=1}^{K}\sum_{z_p \in c_j}(z_p - m_j)$	*Inter-Cluster Distance* d_{min}	*Accuracy*
Brain Image 1	*57.51*	*6.90*	*39*	*97.17*
Brain Image 2	*65.35*	*6.09*	*39*	*96.32*
Brain Image 3	*64.29*	*6.33*	*37*	*97.58*
Brain Image 4	*41.77*	*5.91*	*42*	*96.08*

Table I shows the Segmentation accuracy of total images 4 above images.

Table II shows the average values of *SA* for three segmented parts of the 20 MRI brain images: CSF, GM and WM by the three different algorithms with different level of noise and IIH. The average *SA* values of the proposed algorithm for CSF, GM and WM regions are close to 1 and higher than that of the k-means and FCM algorithms. These results indicate that the proposed algorithm performs well and outperforms its said competing algorithm with presence of different levels of noise and IIH.

Table II: Average values of the *SA* on 20 T1-weighted MRI brain images each in an image volume using different algorithms.

Image Volume	**Segmented Method**	**Segmentation Accuracy (*SA*)**		
		CSF	***GM***	***WM***
T1-Noise 9%, IIH 40%	k-means	0.752	0.801	0.817
	FCM	0.785	0.808	0.838
	Proposed algorithm	**0.890**	**0.897**	**0.931**
T1-Noise 9%, IIH 20%	k-means	0.764	0.808	0.866
	FCM	0.785	0.816	0.852
	Proposed algorithm	**0.904**	**0.920**	**0.957**

Conclusion & Future scopes

In this work, PSO was proposed to segmenting the MRI Brain images which can be very useful for medical images segmentation. The use of PSO algorithm reduce greatly the time complexity. The experimental results have shown the effectiveness and usefulness of the proposed algorithm for MRI Brain images segmentation. As a perspective of this work, use these algorithm to distinguish the normal and abnormal tissues.

For future research, a PSO based automatic clustering algorithm can be developed that can determine the optimum number of clusters of the image, find the cluster centers and perform image clustering in the same program run.

Enhancing these PSO segmentation techniques by employing the parallel PSO algorithms and extending the techniques to the 3D cases.

The method can be employed with Robotics and Neural Networks.

References

[1] Man To Wong, Xiangjian He, Wei-Chang Yeh "Image Clustering Using Particle Swarm Optimization" DOI: 10.1109/CEC.2011.5949627 · Source: DBLP June 2011.

[2] Anita Tandan, Rohit Raja, Yamini Chouhan "Image Segmentation Based on Particle Swarm Optimization Technique" International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 2, February 2014.

[3] ASSAS Ouarda "Segmentation of MR Brain Images Using Particle Swarm Optimization (PSO) and Differential Evolution (DE)" IJCSI International Journal of Computer Science Issues, Volume 11, Issue 6, No 2, November 2014 ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784.

[4] Samy Ait-Aoudia, El-Hachemi Guerrout, Ramdane Mahiou "Medical Image Segmentation using Particle Swarm Optimization" ESI - Ecole nationale Supérieure en Informatique, BP 68M, Oued-Smar 16270 Algiers, Algeria.

[5] Vasupradha Vijaya, Dr.A.R.Kavithab, S.Roselene Rebeccac "Automated Brain Tumor Segmentation and Detection in MRI using Enhanced Darwinian Particle Swarm Optimization (EDPSO)" 2nd International Conference on Intelligent Computing, Communication & Convergence (ICCC-2016).

[6] Paras Patel, Manish Patel " Tumor Detection using Particle Swarm Optimization to Initialize Fuzzy C-Means" Sankalchand Patel College of Engineering,Visnagar, India.

[7] R.Punidha, S.Sakthivel, V.Nehru, C.Gunasundari "SEGMENTATION OF BRAIN TUMOR MRI BASED ON PARTICLE SWARM OPTIMIZATION" International Journal of Pure and Applied Mathematics Volume 116 No. 23 2017, 1-7.

[8] S. Mahalakshmi and T. Velmurugan " Detection of Brain Tumor by Particle Swarm Optimization using Image Segmentation" Indian Journal of Science and Technology, Vol 8(22), DOI: 10.17485/ijst/2015/v8i22/79092 September 2015.

[9] Moussa SEMCHEDINE and Abdelouahab MOUSSAOU, "An Efficient Particle Swarm Optimization for MRI Fuzzy Segmentation" ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY Volume20, Number 3, 2017, 271–285.

[10] International Agency for research on cancer. The Global Cancer Observatory - - May, 2019.

[11] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard,d, Aaron Courville, Yoshua Bengio, Chris Palc, Pierre-Marc Jodoina, Hugo Larochellea "Brain Tumor Segmentation with Deep Neural Networks" May 23, 2016.

[12]https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

[13] https://www.sciencedirect.com/science/article/abs/pii/S0960148118307869

[14] https://www.hindawi.com/journals/mpe/2015/931256/

PROJECT SOURCE CODE

PARTICLE CLASS:

```
class Particles:
    def __init__(self,n_clusters,X):
        xshape0=X.shape[0]


        self.clusters_arr=random.sample(set(X),n_clusters)
        self.clusters_arr=sorted(self.clusters_arr)
        #print(self.clusters_arr)
        self.pixels_arr=np.zeros(xshape0)
        self.pbest_value = float('inf')
        self.pbest_position=self.clusters_arr
        self.p_value=float('inf')
        self.velocity = np.zeros(n_clusters)




    def move(self):
        #print("position is",self.clusters_arr)
        v=np.array(self.clusters_arr)+np.array(self.velocity)
        v=v.astype(int)
        for i in range(len(v)):
            if(v[i]<0):
                v[i]=0
            elif(v[i]>255):
                v[i]=255
```

```
        self.clusters_arr = sorted(list(v))
```

PSO CLASS:

```
class Pso:

    def __init__(self,n_clusters,X,a1=1,a2=.17,a3=6,w=.5,c1=2,c2=2,particle=10,iteration=20):
        self.W=w
        self.c1=c1
        self.c2=c2
        self.a1=a1
        self.a2=a2
        self.a3=a3

        self.X=X
        self.n_particles=particle
        self.n_iter=iteration
        self.gbest_value=float('inf')
        self.n_clusters=n_clusters
        self.gbest_position=np.zeros(self.n_clusters)
        self.n_pixels=X.shape[0]
        self.particles=[Particles(n_clusters,X) for i in range(self.n_particles)]
        self.final_arr=[]


    def pso_cluster(self):
```

```
    for t in range(self.n_iter):
        for particle in self.particles:#particles is the array of all particle
            for pixel in range(self.n_pixels):
                cluster=self.calculate_cluster(self.X[pixel],particle)
                particle.pixels_arr[pixel]=cluster
            fit_val=self.fitness(particle)
            particle.p_val=fit_val
            self.set_pbest(particle,fit_val)
        self.set_gbest()
        self.move_particles()
        print((t+1)*100/self.n_iter,"%")


#this function calculate a pixel is belongs to which cluster for a perticle
def calculate_cluster(self,pixel,particle):
    min=-1
    value=0
    for index in range(self.n_clusters):
        dist=abs(pixel-particle.clusters_arr[index])
        if(min!=-1):
            if(dist<min):
                min=dist
                value=index
        else:
            min=dist
            value=index
    return value
```

```
    #mean error
    def me(self,particle):
        sum_f=0
        p=particle.pixels_arr.shape[0]
        for pixel in range(p):
            sum_f+=abs(int(self.X[int(pixel)])-
int(particle.clusters_arr[int(particle.pixels_arr[int(pixel)])]))
        return(sum_f/p)


    #function to calculate the minimum distance between clusters
    def calculate_inter_cluster_separation(self,particle):
        arr=[]
        for i in range(self.n_clusters-1):
            for j in range(i+1,self.n_clusters):
                dist=abs(int(particle.clusters_arr[i])-int(particle.clusters_arr[j]))
                if(dist==0):
                    return -256
                arr.append(dist)
        return(min(arr))



    #function to calculate the maximum distance of any pixel within any clusters
    def calculate_intra_cluster_distance(self,particle):
```

```
    clusters=[[]]*self.n_clusters
    arr=[]
    for pixel in range(particle.pixels_arr.shape[0]):
        clusters[int(particle.pixels_arr[pixel])].append(self.X[pixel])
    for cluster in clusters:
        centeroid=particle.clusters_arr[clusters.index(cluster)]
        l=len(cluster)
        s=0
        for pixel in cluster:
            s+=abs(int(pixel)-int(centeroid))
        s/=l
        arr.append(s)
    return(max(arr))

#if a cluster is empty then no meaning to use this clusters. is_empty_cluster check this
def is_empty_cluster(self,particle):
    for i in range(self.n_clusters):
        if(list(particle.pixels_arr).count(i)==0):
            return 1
    return 0



def fitness(self,particle):

    inter_cluster_separation=self.calculate_inter_cluster_separation(particle)
    intra_cluster_distance=self.calculate_intra_cluster_distance(particle)
```

```
        ME=self.me(particle)
        #print(MSE,inter_cluster_separation,intra_cluster_distance)
        is_empty=self.is_empty_cluster(particle)
        fit_val=self.a1*intra_cluster_distance+self.a2*(255-
int(inter_cluster_separation))+self.a3*ME+(1000000000)*is_empty
        #print(fit_val)
        return(fit_val)


    def move_particles(self):


        for particle in self.particles:




            new_velocity = (self.W*np.array(particle.velocity))  + (self.c1*random.random())  *
(np.array(particle.pbest_position) + np.array(particle.clusters_arr)*(-1)) + \
                        (random.random()*self.c2) * (np.array(self.gbest_position) +
np.array(particle.clusters_arr)*(-1))
            particle.velocity = list(new_velocity)
            #print("velocity is ",particle.velocity)




            particle.move()




    def set_pbest(self,particle,fit_val):
        if(particle.pbest_value  > float(fit_val)):
            particle.pbest_value = float(fit_val)
```

```
            particle.pbest_position=particle.clusters_arr



    def set_gbest(self):
        for particle in  self.particles:
            if(self.gbest_value > particle.pbest_value):
                self.gbest_value = particle.pbest_value
                self.gbest_position = particle.pbest_position
                self.final_arr=particle.pixels_arr
```

OTHER METHODS:

```
#convert RGB image to grayscle image
def makeGray(img):
    X=img[  :,:  , 0]
    Y=img[  :,:  , 1]
    Z=img[ :,: , 2]
    new_img=0.2989*X+0.5870*Y+0.1140*Z
    return new_img



#returns a 1d array and two dimention
def create_image(K):
    path1="/home/susovan/Downloads/t1_ icbm_normal_1mm_pn3_rf20.rawb"
```

```
    f=open(path1,"rb")
    f1=f.read()
    img=[]
    k=K
    for i in range(181):
        x=[]
        for j in range(217):
            x.append(f1[39277*k+181*i+j])
        img.append(x)
    img=np.array(img)
    img=img.astype(np.uint8)
    sx,sy = img.shape
    plt.imshow(img,cmap="gray")
    img=img.reshape(sx*sy).astype(int)
    for i in range(sx*sy):
        if(img[i]>=165):
            img[i]=0
    return (img,sx,sy)


def show_image(img1,img2,str1="our  segmented image(4 clusters)",str2=" original tissue
pattern(10 clusters)"):
    fig, ax = plt.subplots(1, 2, figsize = (12, 8))
    ax[0].imshow(img1.reshape(sx,sy),cmap="gray")
    ax[0].set_title(str1)
    ax[1].imshow(img2.reshape(sx,sy),cmap="gray")
    ax[1].set_title(str2)
    for ax in fig.axes:
```

```
        ax.axis('off')
    plt.tight_layout();



def accuracy_4(x,z):
    path2="/home/susovan/Downloads/phantom_1.0mm_normal_crisp.rawb"
    f2=open(path2,"rb")
    f2=f2.read()
    img=[]
    k=x
    z=z
    for i in range(181):
        x=[]
        for j in range(217):
            x.append(f2[39277*k+181*i+j])
        img.append(x)
    img=np.array(img)
    img1=img.reshape(sx*sy)
    test=np.ndarray.tolist(img1)
    no_of_point=0
    true_point=0
    for i in range(sx*sy):
        if(test[i]==0):
            if(z[i]==0):
                true_point+=1
            no_of_point+=1
        elif(test[i]==1):
```

```
            if(z[i]==1):
                true_point+=1
            no_of_point+=1
        elif(test[i]==2):
            if(z[i]==2):
                true_point+=1
            no_of_point+=1
        elif(test[i]==3):
            if(z[i]==3):
                true_point+=1
            no_of_point+=1
    accuracy=((true_point*100)/no_of_point)

    print("accuracy is {}%".format(accuracy))
    show_image(z,img1)
def remove_bg(img_bg,it):
    new_img=img_bg.copy()
    factor=1
    for j in range(it):
        if(j>0):
            img_bg=new_img.copy()
        for i in  range(sx*sy):
            if(img_bg[i]<=40):
                total=0
                count=0
                try:
                    total+=img_bg[i-1]
```

```
    count+=1
except:
    pass
try:
    total+=img_bg[i+1]
    count+=1
except:
    pass
try:
    total+=img_bg[i-sy-1]
    count+=1
except:
    pass
try:
    total+=img_bg[i-sy+1]
    count+=1
except:
    pass
try:
    total+=img_bg[i-sy]
    count+=1
except:
    pass
try:
    total+=img_bg[i+sy-1]
    count+=1
except:
```

```
                pass
            try:
                total+=img_bg[i+sy+1]
                count+=1
            except:
                pass
            try:
                total+=img_bg[i+sy]
                count+=1
            except:
                pass
            avarage=total//count
            if(avarage>40):
                new_img[i]=int(img_bg[i]+factor*(avarage-img_bg[i]))
    return new_img


def main():
    img_num=int(input("enter the image number(1-180):"))
    if(img_num>180 or img_num<0):
        print("invalid input !")
        print("try again")
    else:
        print("please wait while computation ...")
        (img,sx,sy)=create_image(img_num)
        img=remove_bg(img,10)
        pso=Pso(4,img,a1=.9,a2=.17,a3=6,w=.5,c1=2,c2=2,particle=10,iteration=20)
        pso.pso_cluster()
```

```
        final_image=pso.final_arr
        show_clusters(final_image,sx,sy)
        accuracy_4(img_num,final_ image)


def show_clusters(final_image,sx,sy):
    fig,ax=plt.subplots(2,2,figsize=(12,12))
    new_img=final_image.copy()
    for j in range(sx*sy):
        if(new_img[j]==3):
            new_img[j]=0
        else:
            new_img[j]=255 ax[0]
    [0].imshow(new_img.reshape(sx,sy),cmap="gray") ax[0]
    [0].set_title("white  matter only")
    new_img=final_image.copy()
    for j in range(sx*sy):
        if(new_img[j]==2):
            new_img[j]=0
        else:
            new_img[j]=255 ax[0]
    [1].imshow(new_img.reshape(sx,sy),cmap="gray")
    ax[0][1].set_title("gray matter only")
    new_img=final_image.copy()
    for j in range(sx*sy):
        if(new_img[j]==1):
            new_img[j]=0
```

```
    else:
        new_img[j]=255 ax[1]
[0].imshow(new_img.reshape(sx,sy),cmap="gray")
ax[1][0].set_title("CSF only")
new_img=final_image.copy()
for j in range(sx*sy):
    if(new_img[j]==0):
        new_img[j]=0
    else:
        new_img[j]=255 ax[1]
[1].imshow(new_img.reshape(sx,sy),cmap="gray")
ax[1][1].set_title("background")
```