Segmentation of Lung X-Ray Images through K-Means Clustering and Particle Swarm Optimization

Presented By:

Asfina Hassan Juicy
ID: 1818017
Department of Biomedical Engineering,
Bangladesh University of Engineering and Technology(BUET)
Dhaka, Bangladesh

INTRODUCTION

Objective:

Re-implementing a research study from existing literature to validate findings and gain hands-on experience with the methodology

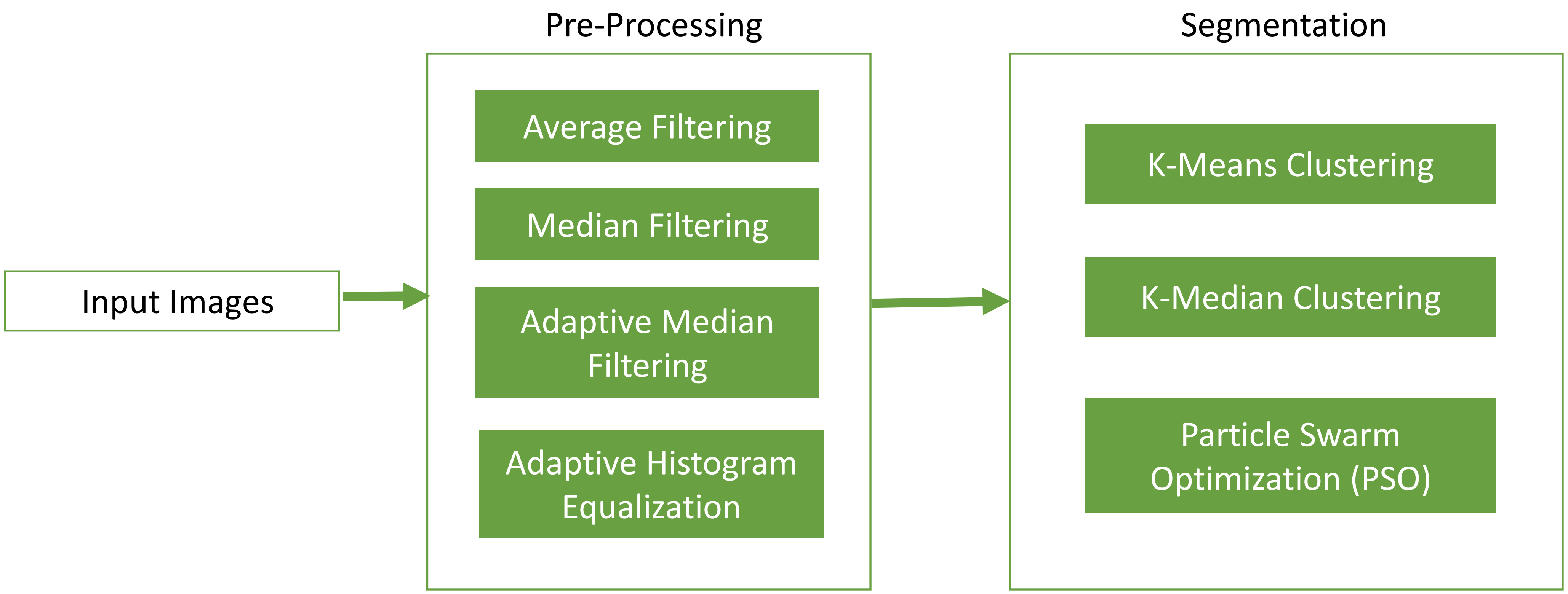
Explore Image Segmentation Algorithms

↓

Computed Tomography(CT) Scan Images

Literature Used: Lung Cancer Detection Using Image Segmentation by means of Various Evolutionary Algorithms
(https://onlinelibrary.wiley.com/doi/10.1155/2019/4909846)

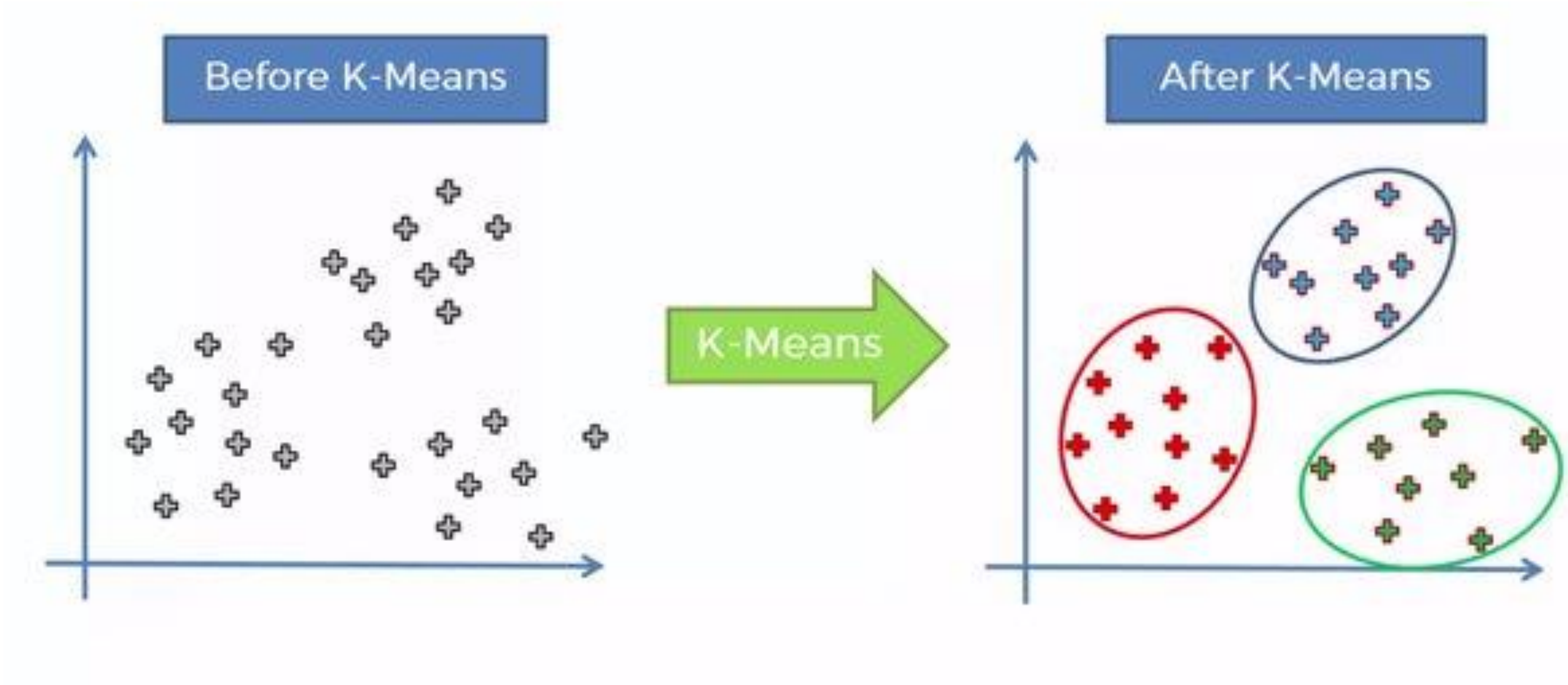
STEPS



After Segmentation, result of Tumor Extraction shown but process/algorithm not specified

ALGORITHM

K-Means Clustering



1. Specify the number of clusters ‘K’
2. Randomly initialize cluster centers(centroids)
3. Assign each data point to the closest cluster center
4. Update cluster center as the mean of all data in that cluster
5. Repeat steps 3 and 4 until max iteration reached/loss minimized

ALGORITHM

K-Means Clustering : Implemented Code

```
function [Out,lbl] = customKmeans(filtered_f,k)

    filtered_f=double(filtered_f); %input image
    b = size(filtered_f);
    N1 = b(1)*b(2); %total number of pixels

    x=filtered_f(:); %flattened image

    Initial_mean = [255*rand(1,k)]'; %initialize random cluster mean

    n=20; %number of iterations
    n1=5; %Treshold on the cost J
    i1=2; %initial index
    J1 = [0 0];

    while i1<n %loop till maximum iteration

        Rnk = zeros(N1,k); %A N-by-k vector giving 1 in the k-th cluster it is closest to
        d = zeros(N1,k); %A N-by-k vector calculating distance with each cluster centroid
        for i=1:N1
            for j=1:k
                d(i,j) = (x(i)-Initial_mean(j))*(x(i)-Initial_mean(j))'; %calculating distance
            end
```

ALGORITHM

K-Means Clustering : Implemented Code

```
    [min, Imin] = max(-d(i,:)); %find the index of minimum distance in a row
    Rnk(i,Imin) = 1; %update the Rnk matrix
end

J1(i1) = 0;
sumRnk = zeros(1,k);
for i=1:N1
    for j = 1:k
        J1(i1) = J1(i1)+Rnk(i,j)*d(i,j); %calculating loss with formula
    end
    sumRnk = sumRnk + Rnk(i,:); %Calculating how many pixels in one cluster
end

Initial_mean=zeros(k,1);

%updating the mean
for i=1:N1
    for j = 1:k
        temp = Rnk(i,j)*x(i);
        if (temp == 0)
        else
        Initial_mean(j) = Initial_mean(j)+temp;
```

ALGORITHM

K-Means Clustering : Implemented Code

```
            end
        end
    end

    for i =1:k
        if sumRnk(i)~=0
        Initial_mean(i) = Initial_mean(i)/sumRnk(i);
        end
    end

    if (abs(J1(i1)-J1(i1-1))<n1)
        break; %break the loop if loss is below threshold
    end
    i1 = i1+1;

    disp(J1(i1-1)); %display loss
end
```

ALGORITHM

K-Means Clustering : Implemented Code

```
    j2=1;
    p = 0;
    Out = zeros(b(1),b(2));
    for i2 = 1:b(2)*b(1)
        [temp1,Itemp] = max(Rnk(i2,:)); %determine which cluster it belongs
        Out(i2-p,j2,1) = Initial_mean(Itemp); %take that cluster's mean and
                                               %assign it to the pixel
        if i2-p == b(1) %reshaping to original image size
            p = p+b(1);
            j2 = j2+1;
        end
    end

    x_copy=x;

    for i = 1:k
        x_copy(find(Rnk(:,i)))=i; %instead of mean updating the pixel values
                                  %with the cluster number
    end

    lbl= reshape(x_copy,size(filtered_f,1),size(filtered_f,2));
end
```

ALGORITHM

K-Medians Clustering

1. Specify the number of clusters 'K'
2. Randomly initialize cluster centers(centroids)
3. Assign each data point to the closest cluster center
4. **Update cluster center as the median of all data in that cluster**
5. Repeat steps 3 and 4 until max iteration reached/loss minimized

ALGORITHM

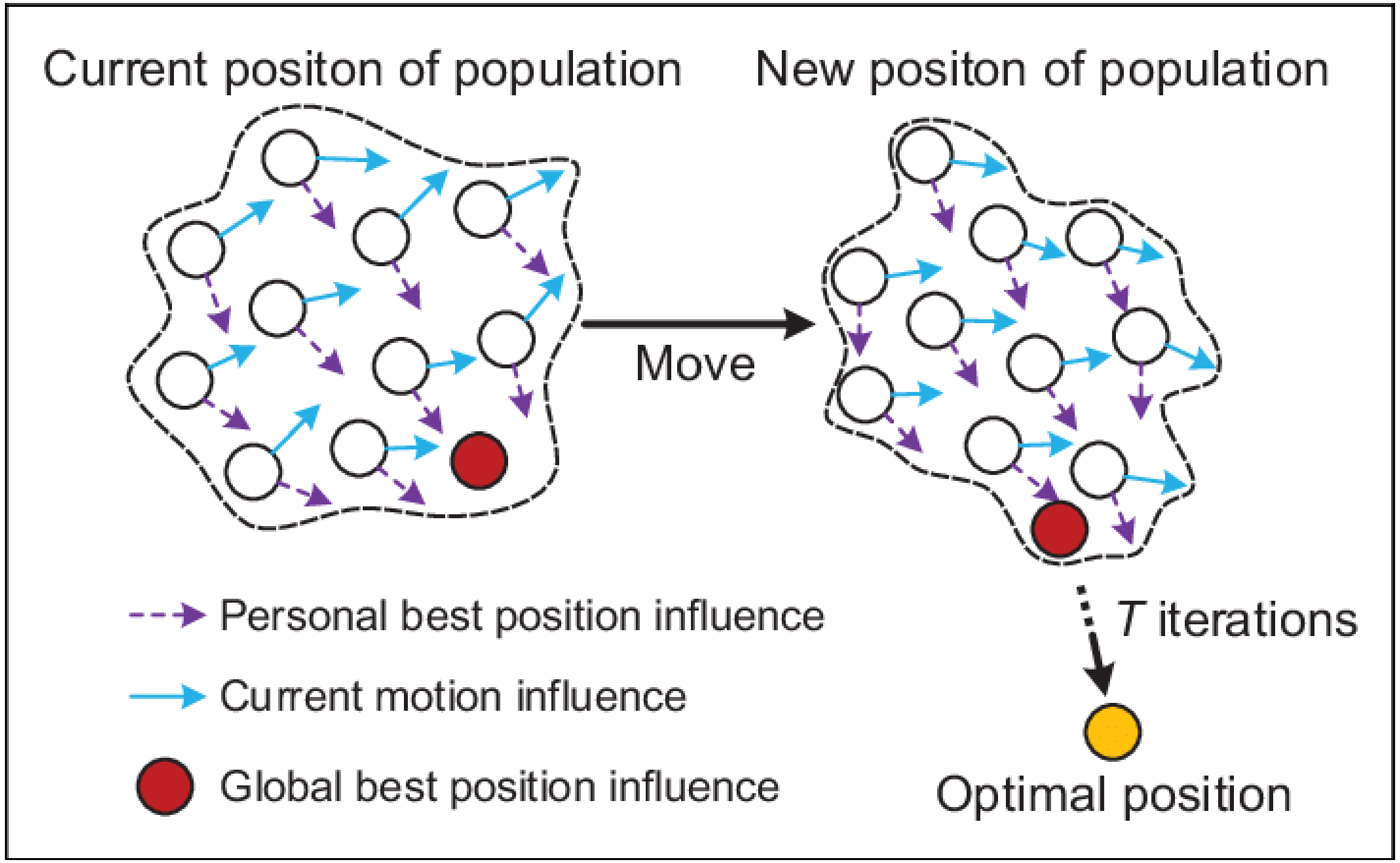
K-Median Clustering : Implemented Code

Changed Portion:

```
Initial_median = zeros(k, 1);

for i = 1:k
    temp = x(Rnk(:, i) == 1);
    if ~isempty(temp)
        Initial_median(i) = median(temp); % Calculate median for each cluster
    end
end
```

ALGORITHM

Particle Swarm Optimization (PSO)

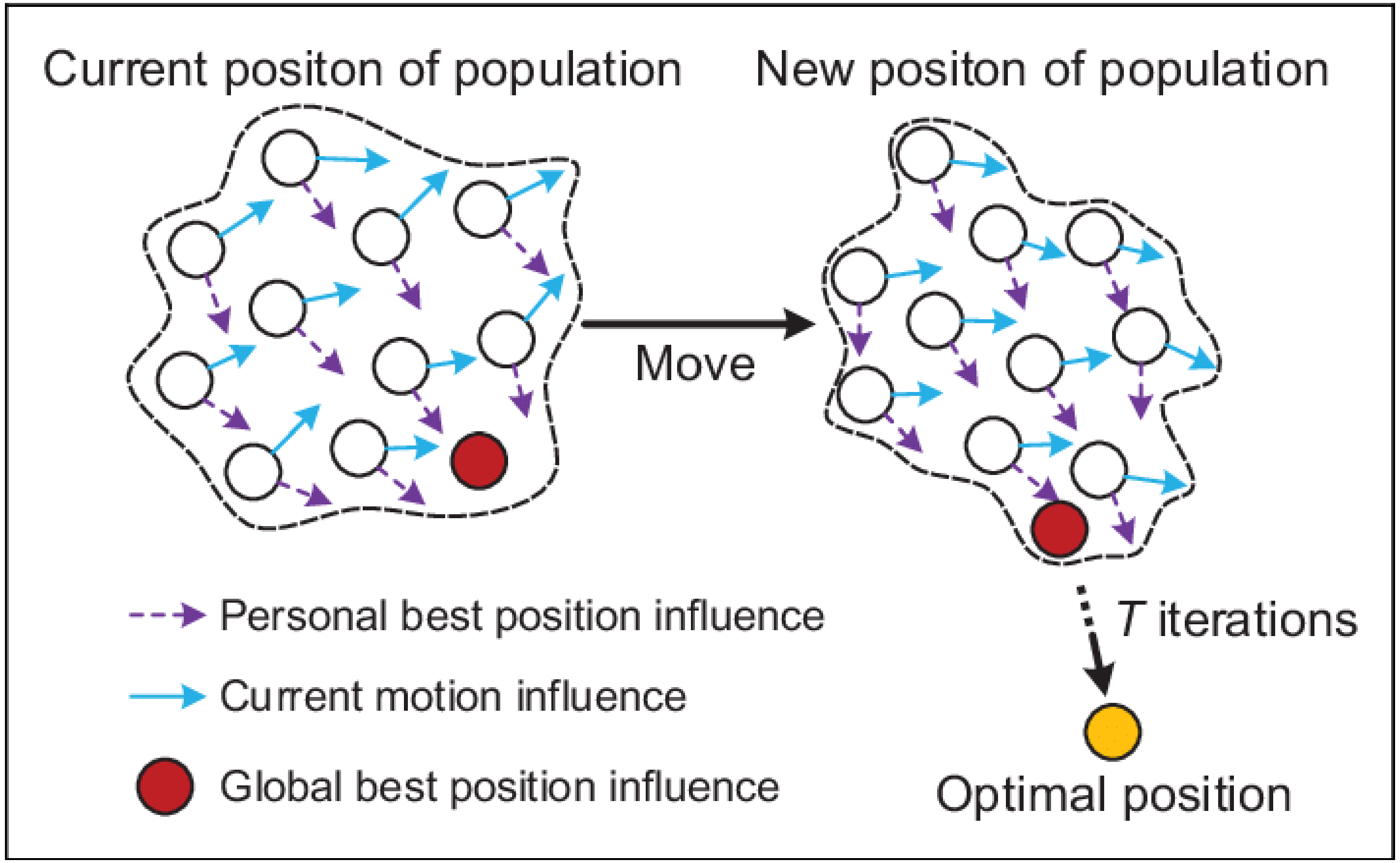


Initialization

1. For each particle i in a swarm population size P:
   - Initialize X_i randomly(Position)
   - Initialize V_i randomly(Velocity)
   - Evaluate fitness $f(X_i)$ [From a fitness fcn]
   - Initialize pbest with a copy of X_i
2. Initialize gbest with a copy of X_i of best fitness

ALGORITHM

Particle Swarm Optimization (PSO)



Update

1. For each particle i :
   - Update V_i^t and X_i^t with:

$$v(t+1)= v(t)+c_1r_1[\text{pbest}(t)-x(t)]+c_2r_2[\text{gbest}(t)-x(t)],$$
$$x(t+1)= x(t)+v(t+1),$$

2. Evaluate the fitness

3. $\text{pbest}_i = X_i^t$ if $f(\text{pbest}_i)<f(X_i^t)$
   $\text{gbest} = X_i^t$ if $f(\text{gbest})<f(X_i^t)$

ALGORITHM

Particle Swarm Optimization (PSO): Implemented Code

As the fitness function **was not defined properly** in the paper,
a different approach was taken with **a different fitness function** for utilizing the algorithm:

```
function fitness = fitness_fcn(data, x, num_clusters)
    % extract cluster centroids from x
    centroids = reshape(x, num_clusters, 2);

    % perform k-means clustering using the centroids as initial positions
    [idx, ~] = kmeans(data, num_clusters);

    % calculate the fitness as the sum of squared distances
    fitness = sum(sum((data - centroids(idx, :)).^2));
end
```

ALGORITHM

Particle Swarm Optimization (PSO): Implemented Code

```
image = double(filtered_image); %input image

number_clusters = 12; % Number of clusters
number_particles = 10; % Number of particles in the swarm
max_it = 10; % Max number of iterations
inertia = 0.9; %Randomly taken inertia weight
c1 = 0.5; % Randomly taken cognitive coefficient
c2 = 0.5; % Randomly taken social coefficient

image = image / max(image(:)); % Normalizing the image

data = image(:); % Flatten the image

% Initializing all position,velocities,pbest,gbest
number_features = number_clusters * 2; % Two parameters for each cluster (centroid x and y)
p_position = rand(number_particles, number_features); % Initialize particle positions
p_velocity = rand(number_particles, number_features); % Initialize particle velocities
pbest = p_position; % Initialize best position of each particle
p_best_fitness = inf(number_particles, 1); % Initialize best fitness for each particle
gbest = [];
g_best_fitness = inf;
```

ALGORITHM

Particle Swarm Optimization (PSO): Implemented Code

```
% PSO optimization
for iteration = 1:max_it
    for i = 1:number_particles
        % evaluating fitness of each particle with fitness fcn
        fitness = fitness_fcn(data, p_position(i, :), number_clusters);

        % updating pbest
        if fitness < p_best_fitness(i)
            p_best_fitness(i) = fitness;
            pbest(i, :) = p_position(i, :);
        end

        % updating gbest
        if fitness < g_best_fitness
            g_best_fitness = fitness;
            gbest = p_position(i, :);
        end
    end
```

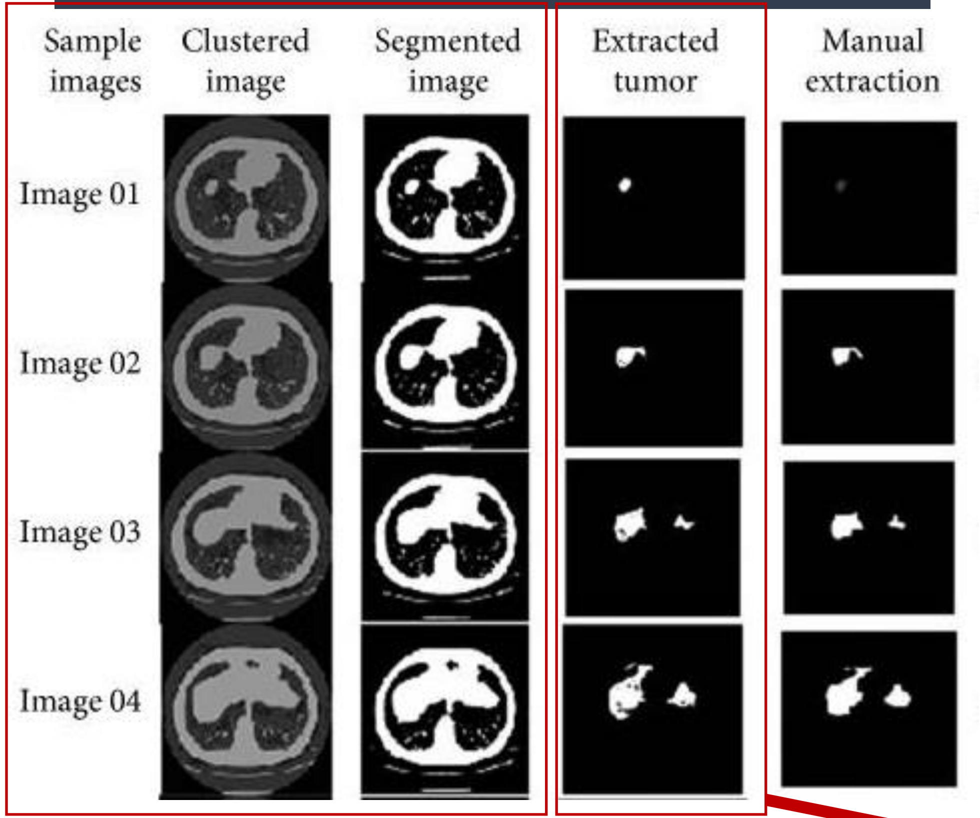
ALGORITHM

Particle Swarm Optimization (PSO): Implemented Code

```
    % random numbers for update formula
    r1 = rand(number_particles, number_features);
    r2 = rand(number_particles, number_features);

    p_velocity = inertia * p_velocity + ...
        c1 * r1 .* (pbest - p_position) + ...
        c2 * r2 .* (gbest - p_position);

    p_position = p_position + p_velocity;
end

% reshape the global best position into centroids
centroids = reshape(gbest, number_clusters, 2);

% perform k-means clustering using the centroids as initial positions
[idx, ~] = kmeans(data, number_clusters);

 % reshape the clustering result back into image size
 segmented_image = reshape(idx, size(image));

 % display the segmented image
 figure;
 imshow(segmented_image, []);
```

RESULTS

Result from paper



Implementable with given algorithms

Limitations:

- No parameters(k value, initial mean) for k-means clustering was given
- No parameters(constants and weights of equation) for PSO was given
- **Fitness function** and the **feature of gbest** was not defined enough for implementation.
  So, a **custom fitness function and feature** for gbest was employed for PSO.
  This limited the scope for GCPSO
- Tumour extraction method not specified, not deducible from the result photo

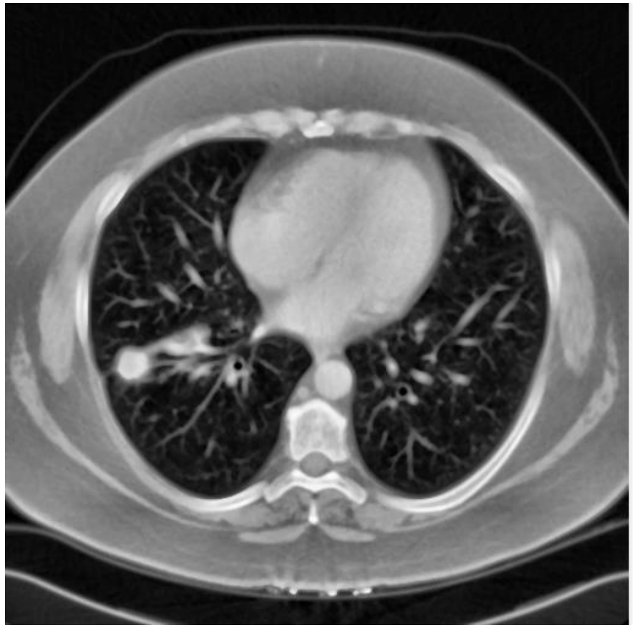
RESULTS

Results

Result on a CT scan of implemented algorithms:

Original Image

Preprocessed Image



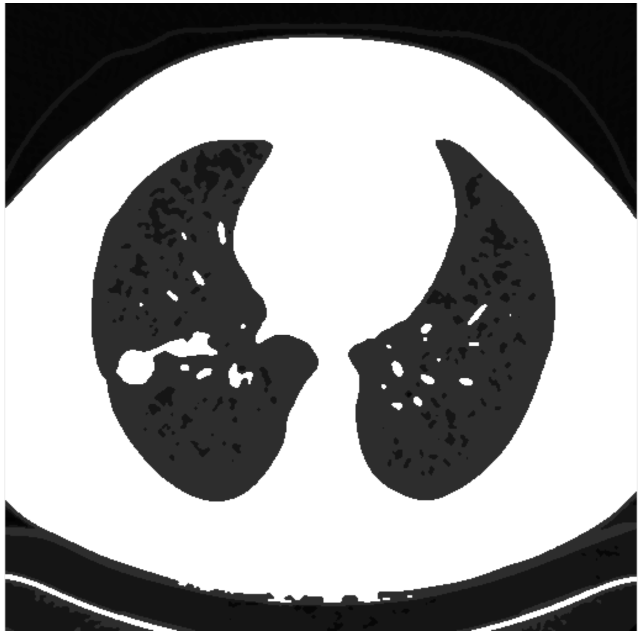
RESULTS

Results

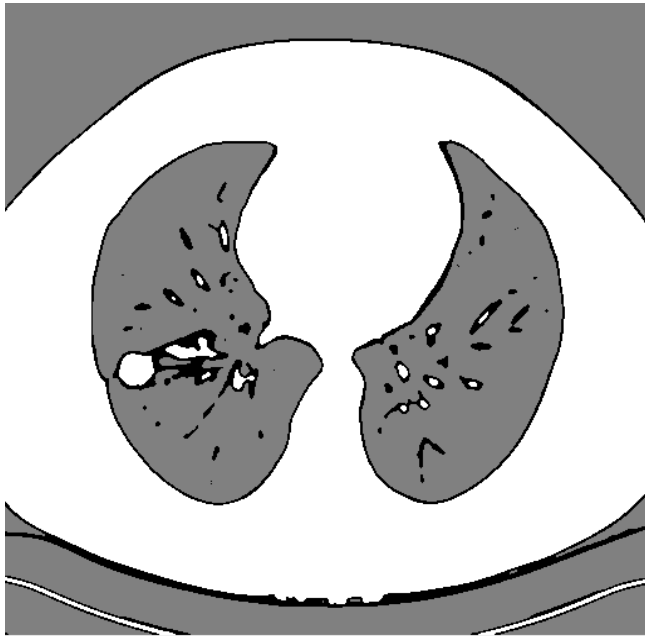
Clustered Image:

K-means

K-medians



PSO with optimizing
K-means centroids



RESULTS

Results

Segmented Image:

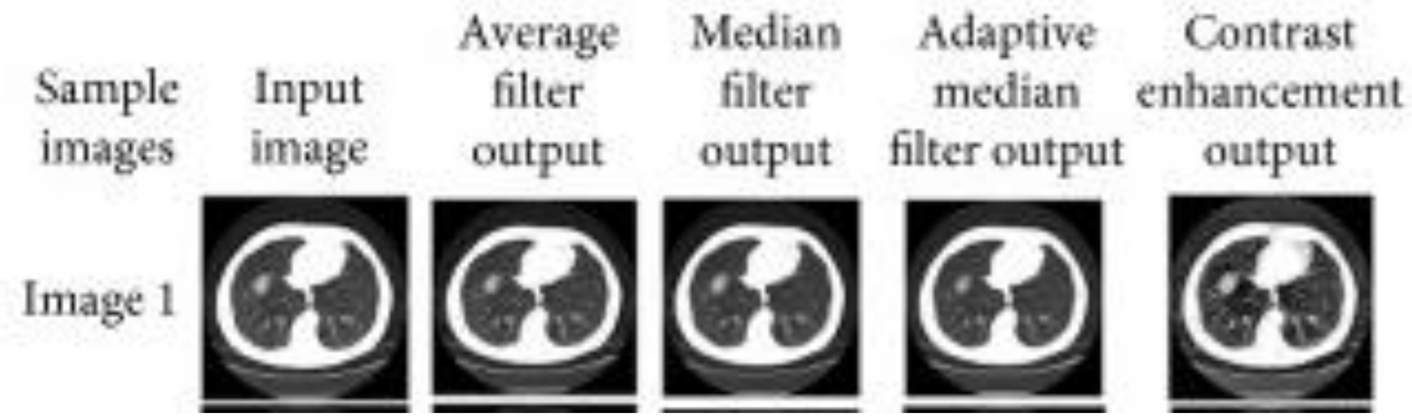
K-means

K-medians

PSO with optimizing
K-means centroids

RESULTS

Comparison



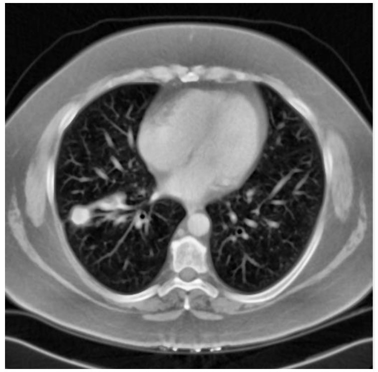
Input

Average filter

Median filter

Adaptive Median filter

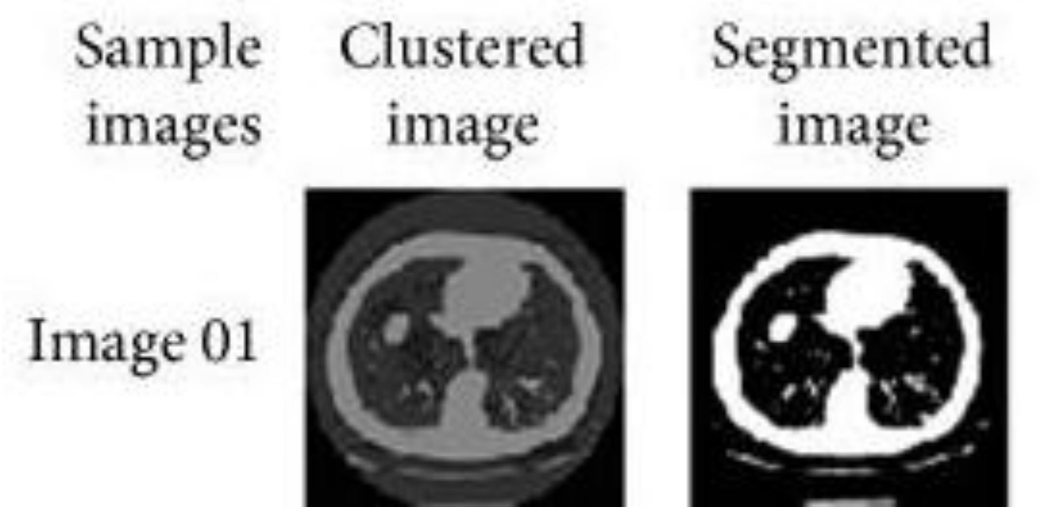
Contrast Enhanced



RESULTS

Result from paper

K-means:



Paper

Clustered

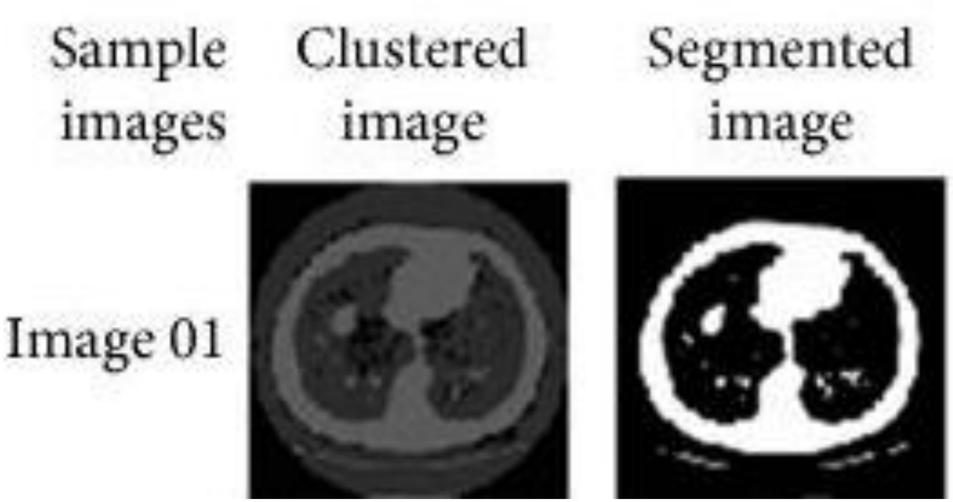
Segmented

Implemented

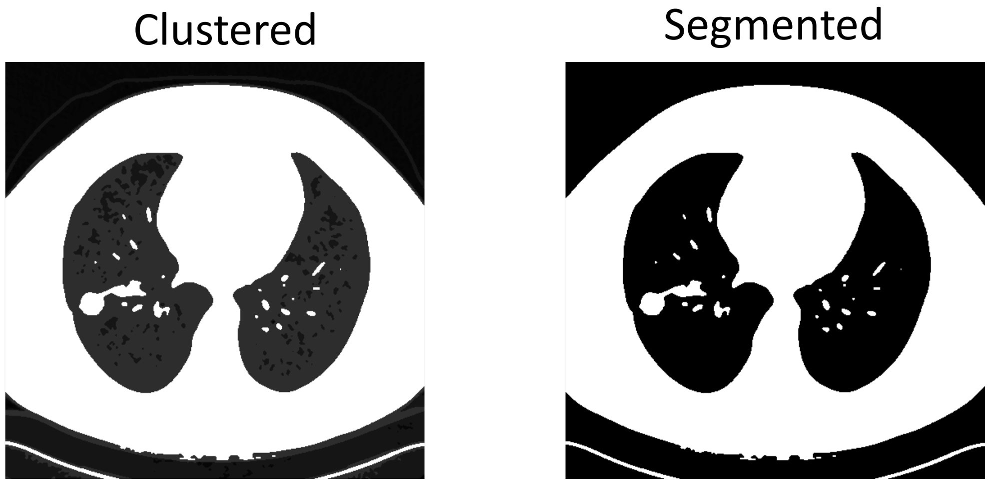
RESULTS

Result from paper

K-medians:



Paper

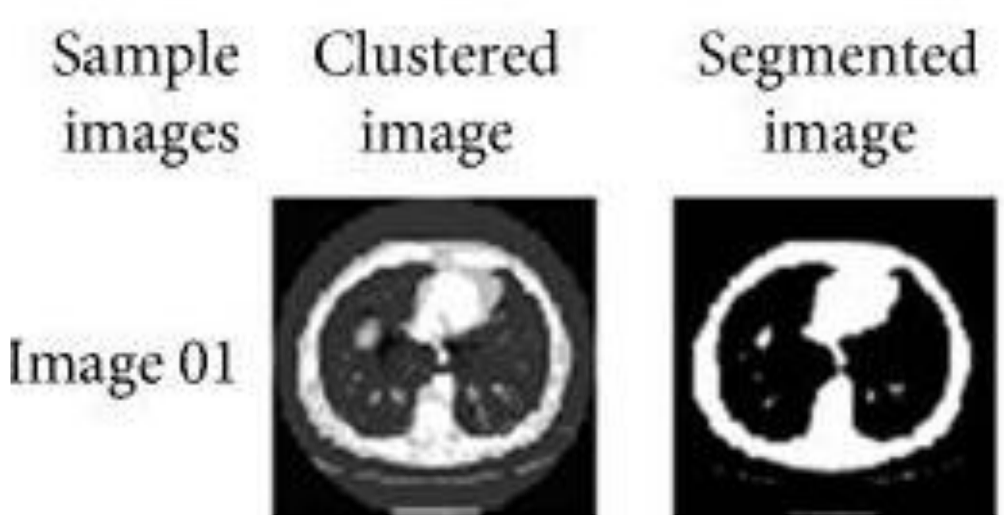


Implemented

RESULTS

Result from paper

PSO:



Paper

Clustered

Segmented

Implemented

RESULTS

Limitations

- No parameters(k value, initial mean) for k-means clustering was given
- No parameters(constants and weights of equation) for PSO was given
- **Fitness function** and the **feature of gbest** was not defined enough for implementation. So, a **custom fitness function and feature** for gbest was employed for PSO. This limited the scope for GCPSO.
- Tumour extraction method not specified, not deducible from the result photo

RESULTS

Other Possible Algorithms

Otsu's Thresholding: Uses histogram to find the optimal threshold value for segmentation

Mean-Shift Clustering : Iterates each pixel towards the mode of the data distribution

Fuzzy C-means Clustering : extends the traditional K-Means clustering method to assign data points to multiple clusters with degrees of membership rather than exclusively assigning them to a single cluster.

Watershed Segmentation: particularly useful for separating objects that are close or touching

RESULTS

Key Take-aways

- The k-means clustering method is a dependable method for multi-level intensity segmentation with proper pre-processing.
- PSO is an optimizing method which can be used for versatile segmentation if appropriate fitness function is used.

THANK YOU