Computational Intelligence Group 9 Project

Aaron Neidlinger

Alicia Esquivel

Dewi Karismawati

Evan Teters

Jacob Krajewski

Video: https://youtu.be/yyl45kl9F08

Responsibilities:

Evan Teters: Worked on Fuzzy Local Information C-Means (FLICM)

Jacob Krajewski: Gathered data for analysis attempted to implement dynamic eta updating

Dewi Karismawati: Implementation of Possibilistic Local Information C-Means, Fuzzy C Means

Dewi Kharismawati and Jacob: Attempted Implementation of Sequential Possibilistic Local Information One-Means Clustering (SPLI1M) as a derivation of PLICM

Alicia Esquivel: Comparison between k-means and mean shift clustering.

Aaron Neidlinger: Fuzzy clustering visualization in RGB images.

Problem:

Image segmentation is the most important task in image analysis and computer vision. In crisp clustering, like k-means, one pixel just belongs to one cluster, which is not effective when the image has some issues such as poor contrast, overlapping intensities, and noise. Fuzzy clustering, such as Fuzzy C Means (FCM), have become the alternative since they have a degree of membership on each cluster for each pixel. However, FCM does not work well on noisy images. Fuzzy Local Information C-Means (FLICM) is introduced to tackle FCM problem on noisy images by enforcing local neighborhood to get the local boundary of the object. However, the sum-to-one constraint in FLICM still lead into problem when the local spatial position information of specific noise pixels is ambiguous. This problem is solved by Possibilistic Local

Information C-Means (PLICM). PLICM is designed to be more robust against noise without sum-to-one constraint. PLICM will fail under coincident cluster, when actual positive clusters in the image are less than cluster initialization. Fortunately, Sequence Possibilistic Local Information 1 Means (SPLI1M) was introduced to solve this problem. SPLI1M basically runs possibilistic local information one means sequentially until all local consistent clusters are found. Therefore, in this project we are going to investigate and compare the performance of FLICM, SPLI1M, and crisp clustering algorithm for image segmentation. Additionally, the display of segmentation for many fuzzy clustering algorithms shows a crisp representation where only the classes are displayed and the membership values are not seen. This has the disadvantage of not being able to see how "confident" the result is in its classification. In this project we will also develop a way to show the fuzzy membership values in the segmented image so "confidences" can be seen.

Algorithm Design:

The design of the algorithms (pseudo-code, flowcharts, or some other structured descriptive means)

Fuzzy Local Information C Means Pseudocode:

1. Initialize variables
    a. Variables cluster number, m (fuzzifier), **ε** (threshold), winSize are all given as input to the function
    b. U, the matrix for degree of membership
    c. Intermediate matrices needed for calculations of membership values
2. Repeat:
    a. Modify U with m, the fuzzifier
    b. Calculate the cluster centers C using the equation for the cluster center prototypes (v_k , equation 20, in [2])

$$v_k = \frac{\sum_{i=1}^{N} u_{ki}^m x_i}{\sum_{i=1}^{N} u_{ki}^m}.$$

c. Calculate the membership values for U (u_{ki}, equation 19, in [2]) In the code this is done through many intermediate steps to help get the Euclidean distances needed for each cluster center.

$$u_{ki} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|x_i - v_k\|^2 + G_{ki}}{\|x_i - v_j\|^2 + G_{ji}} \right)^{1/m-1}}$$

d. Also calculate the objective function result as matrix J (described as J in [2], equation 18)

$$J_m = \sum_{i=1}^{N} \sum_{k=1}^{c} \left[u_{ki}^m \|x_i - v_k\|^2 + G_{ki} \right]$$

e. If the norm of ||J-J_prev|| is < **ε**, (or iterations > maxIterations) **exit loop**

3. Determine labels by looking at what each pixel's highest membership is
4. Segment & return image

Possibilistic Local Information C-means Psuedocode:

1. Initialize variables:
   a. Variables cluster number, m (fuzzifier), **ε** (threshold), winSize are all given as input to the function
   b. Initialize eta as 3.5 (we did not successful to implement dynamic eta)
   c. Initialize degree of membership U and cluster center with FLICM
2. Repeat:
   a. Compute the cluster center V using (v_k, equation 14, in [3])

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m x_k}{\sum_{k=1}^{N} u_{ik}^m}$$

   b. Calculate the membership values for U (u_{ki}, equation 16, in [3]) In the code this is done through many intermediate steps to help get the Euclidean distances needed for each cluster center.

$$u_{ik} = \frac{1}{1 + \left(\frac{\|x_k - v_i\|^2 + G_{ik}}{\eta_i} \right)^{1/m-1}}$$

   c. Also calculate the objective function result as matrix J (described as J in [3], equation 15)

$$J_m = \sum_{k=1}^{N} \sum_{i=1}^{c} [u_{ik}^m ||x_k - v_i||^2 + G_{ik}] + \sum_{i=1}^{c} \eta_i \sum_{k=1}^{N} (1 - u_{ik})^m$$

d. If the distance of ||J - J_prev|| is < **ε**, (or iterations > maxIterations) **exit loop**

3. Determine labels by looking at what each pixel's highest membership is
4. Segment & return image

Sequential Local Information 1-means Psuedocode:

1. Initialize variables:
    a. Fuzzifier = 2, C = 100, Seed = 2234, epsilon = 0.01
    b. Initialize degree of membership U, and cluster V as empty.

2. FOR 1 to C:
    a. Pick cluster center v randomly based on probability

3. WHILE Difference in V is less than epsilon

4. While min distance ||v - w|| >= 2*eta (w is all cluster center in V)

5. WHILE eta Small < eta Large ** Updating eta dynamically**

- Get d from paper [3] for all data points using data points and current V
- Update U (matrix for degree of membership) using equation from paper [3]
- Determine average typicality (uTyp: matrix of average typicality) with summation of U divided by number of data points
- Increment eta Small

END WHILE

- Compute delta U by finding difference in uTyp matrix
- Compute delta delta U by finding difference in delta U matrix

IF delta delta U <= 0

- eta = Knee(uTyp)

ELSE

- continue

END IF

6. For all pixel
    a. Calculate the membership values for U (u_{ki} , equation 16, in [3]) In the code this is done through many intermediate steps to help get the Euclidean distances needed for each cluster center.

$$u_{ik} = \frac{1}{1+(\frac{\left\|x_k - v_i\right\|^2 + G_{ik}}{\eta_i})^{1/m-1}}$$

    b. Compute the cluster center V using (v_k , equation 14, in [3])

$$v_i = \frac{\sum_{k=1}^{N} u_{ik}^m x_k}{\sum_{k=1}^{N} u_{ik}^m}$$

END WHILE

- Add U to vector of all U
- Add V to vector of all V
- break if no more clusters can be found

END WHILE

END FOR

Fuzzy Visualization Pseudocode:

1. Get number of rows and columns in image
2. Get red, green and, blue values for pixel
3. Reshape labels into array matching image
4. Adjust pixel RGB values using 1:2:3 ratio
    a. red pixel = red pixel + (membership value * 0.5 * (255 - red pixel))
    b. blue pixel = blue pixel + (membership value * 1 * (255 - blue pixel))
    c. green pixel = green pixel + (membership value * 1.5 * (255 - green pixel))
5. Rebuild the image using tinted pixels
6. Repeat for all rows and columns in image array
7. Display image

Results & Analysis:

Notes on Experimentation:

Unless the experiment involves a parameter, values were set as follows

Number of clusters = 11

m = 2

Window Size = 3;

Max Iterations = 100;

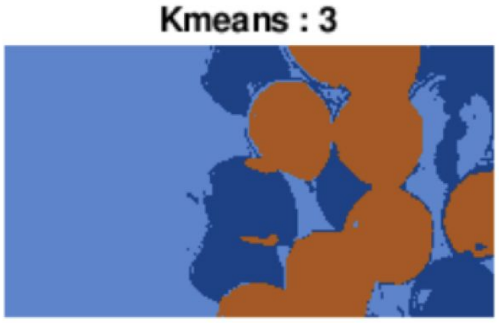
Threshold ε = 1.0e-2;

Typically, many of our experiments ran into the max iterations before hitting the threshold. We did not see this as an issue, as they usually looked as we expected. In the interest of time, we chose not to change the maximum iterations for any experiment.

Comparison between K-means and Mean Shift Clustering

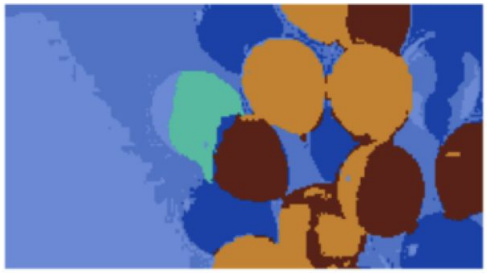
K-means is a type of clustering algorithm that subdivide data points of dataset into cluster based on nearest mean values. In K-means clustering algorithm, the number of clusters is specified in advance. In Mean Shift, there is a parameter called bandwidth that needs to be adjusted. In this result, we compared them and we concluded with the following statements:

- K-means and Mean Shift segmentation are two crisp clustering algorithms.
- They will have problems when clustering ambiguous color images.
- If the input of number of clusters K is small, then, some clusters (colors) are missing in the segmentation result.
- If the input of number of clusters K is big, then, the segmentation result will be better.



Original

Kmeans : 6

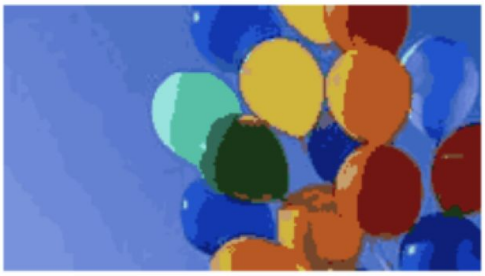


Original

Kmeans : 11

Original

Kmeans : 20



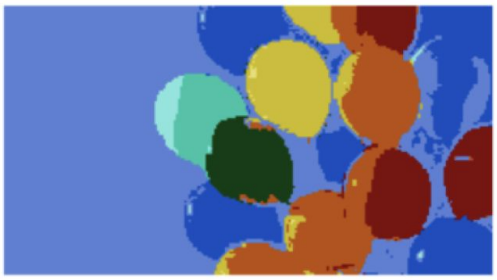
Original

Kmeans : 40

- As bandwidth values increases, the number of clusters (colors) decreases.
- As bandwidth values decreases, the number of clusters (colors) increases.
- The bandwidth parameter has a similar influence on image segmentation as the parameter eta in the SPLI1M, which controls the cluster size or the color range for each segment.

Original

MeanShift - bandwidth=0.2



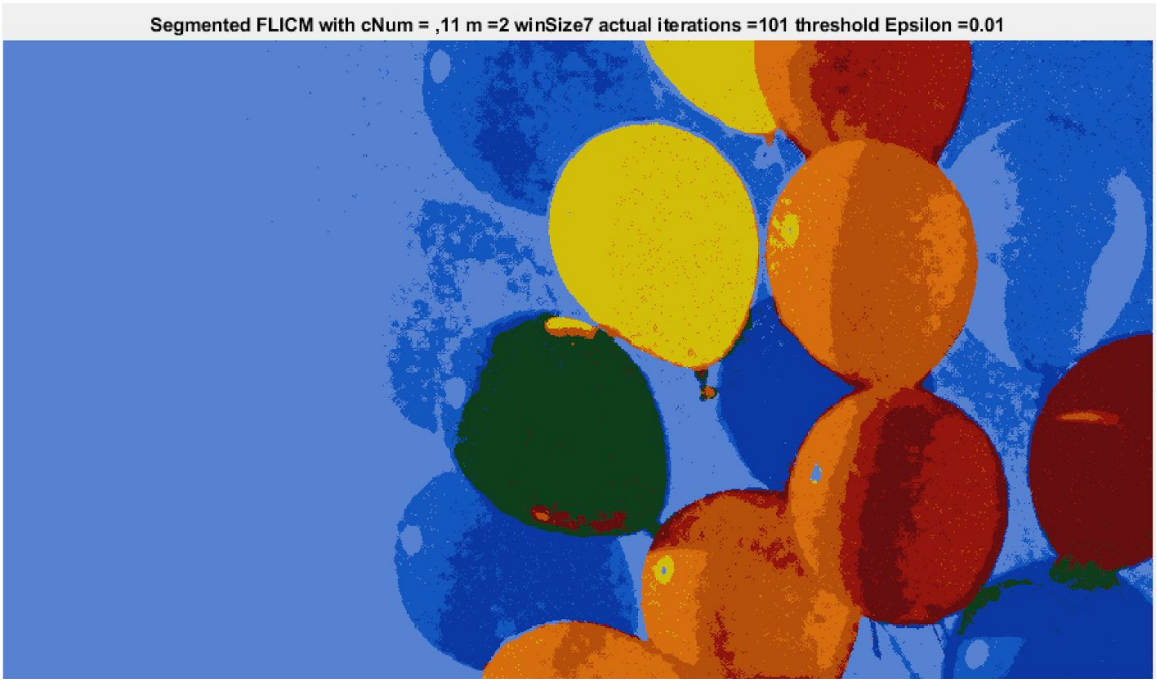
Original

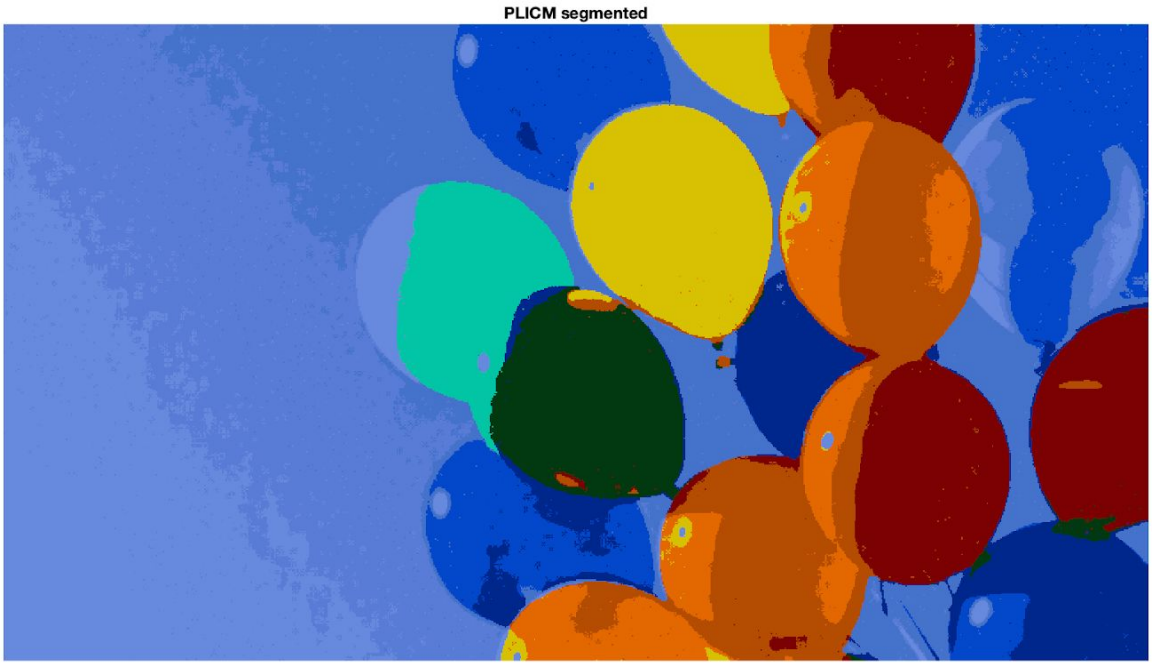
MeanShift - bandwidth=0.10

Original

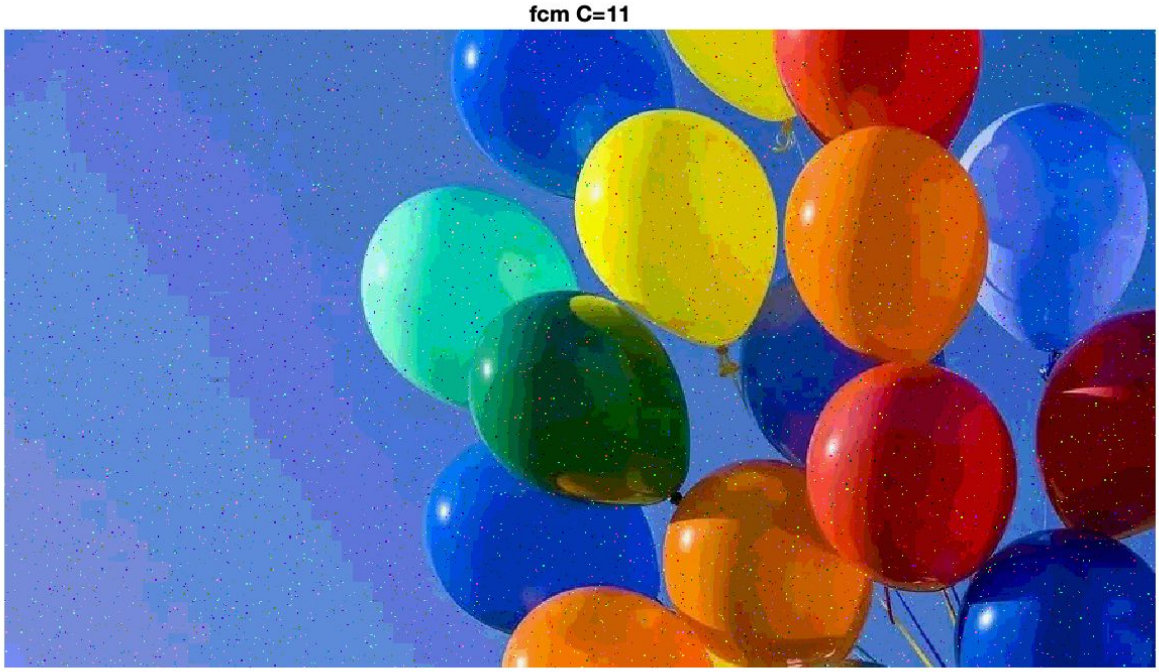
MeanShift - bandwidth=0.050

Noise Comparison





You can see above, that PLICM removes some of the noise that FLICM kept around (keep in mind that PLICM runs FLICM at the beginning to get a better cluster centers and membership values at initialization) so it is no surprise that PLICM does a better job segmenting. Compare them to naive FCM on the next page, that did a poor job handle noise and making clear segmentation.

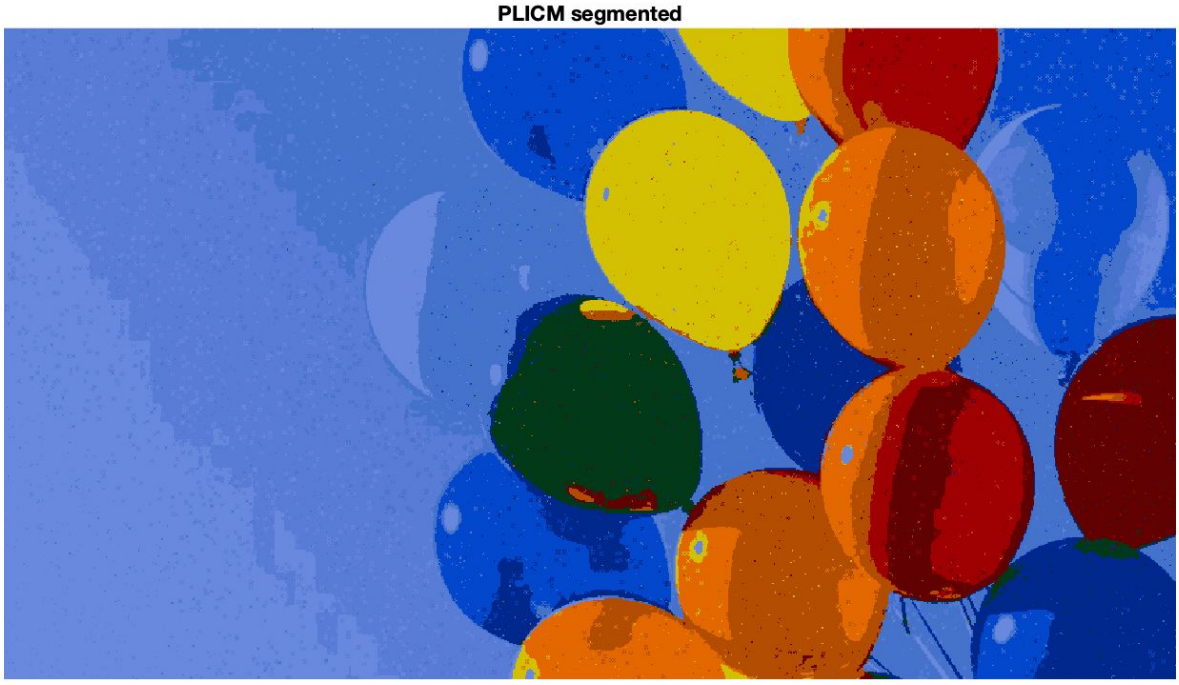


For K-means with noise, it did surprisingly well at removing noise, but did not segment individual balloons well. Its performance is arguably better than FLICM, which is better than we expected kmeans to perform. We will show later that the performance of FLICM and PLICM depends in part on window size.

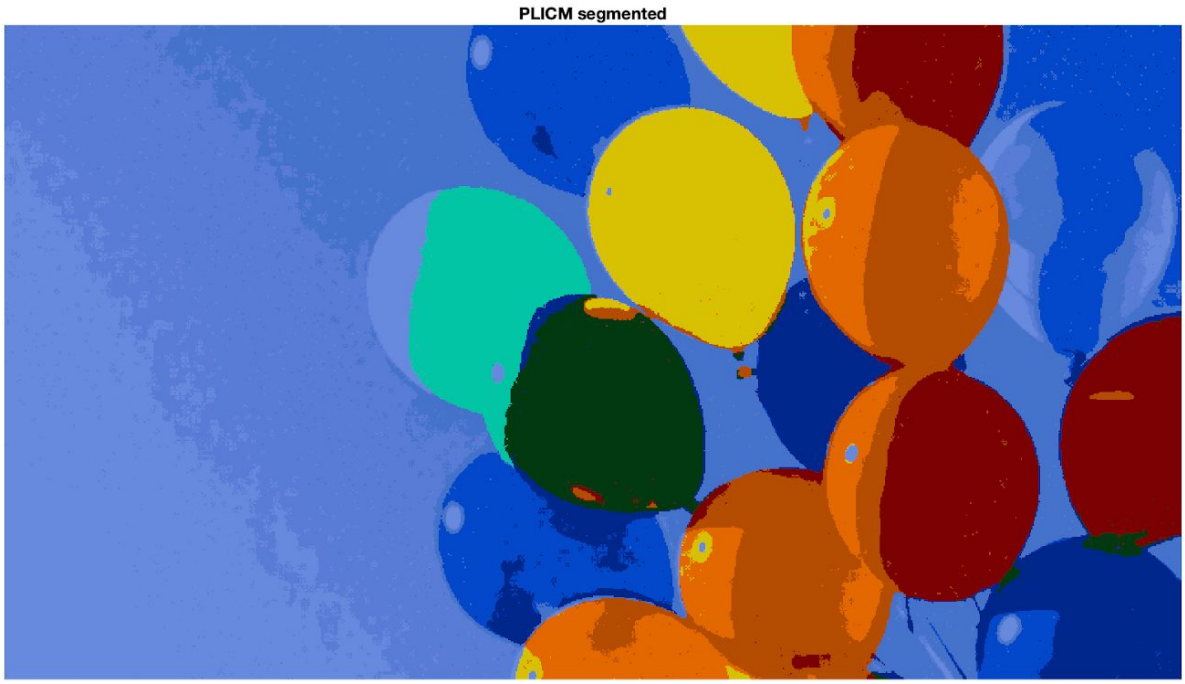
Our attempt to implement k means with spatial information has shown a significant *inability* to deal with noise compared to general k means. It seems when spatial information is accounted for the noise is too heavily weighted when trying to differentiate the different clusters:

kmeans + spatial (little noise) with cNum = 11

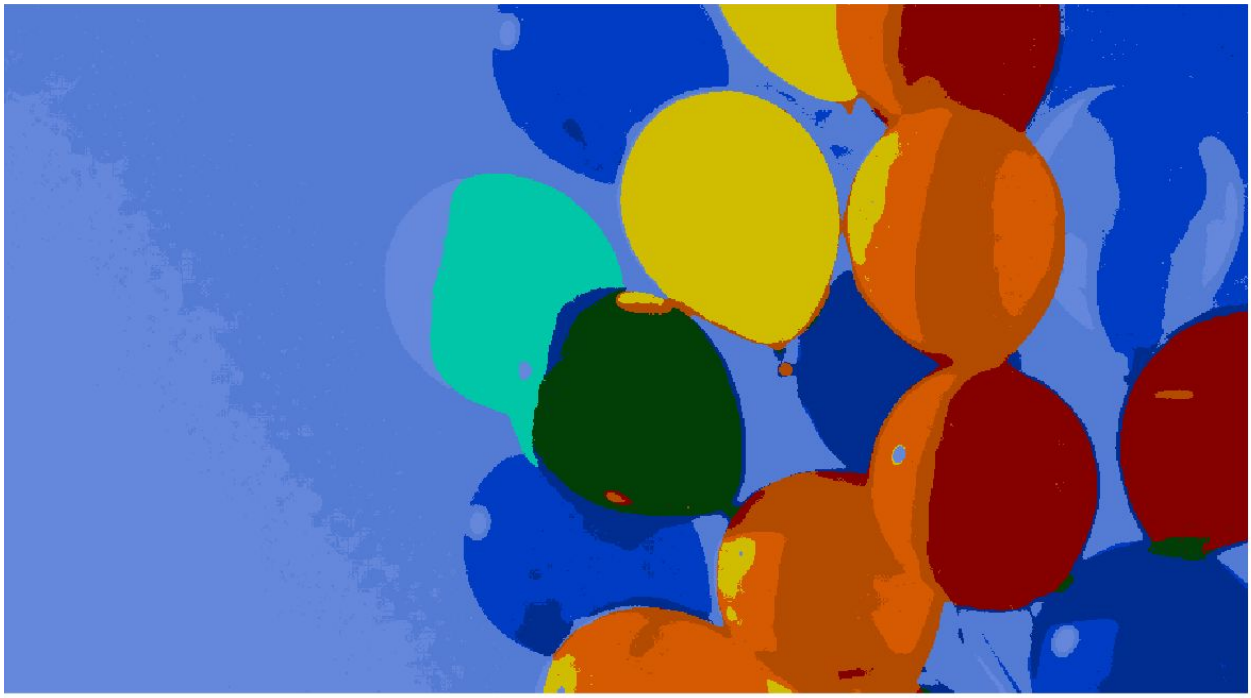
Comparison Of Window Size on Noise Reduction for FLICM



PLICM window size = 3



PLICM windows size = 7



PLICM windows size = 13

Window size drastically affects PLICM's ability to cope with noise. With a window size of 3 most of the noise remains in the segmented image. A window size of 7 removes most of the noise from the image. Our final test with a window size of 13 removed practically all noise from the segmented image and was nearly indistinguishable from tests without any noise at all. Increasing the window size allows the algorithm to handle noise better but decreases the performance (time to run the algorithm). Obviously to get the best image, window size can be greatly increased but to balance performance with results we found that a window size of 7 worked well for this image.

Effect of the fuzzifier

There is a parameter m, present in FLICM and PLICM, that is used to weight the fuzziness of membership values in these clustering methods. We found that the higher the number, the more the membership is "fuzzied" and the harder it is for the algorithm to differentiate between the clusters. Observe a low value of m (1.5) and a higher value (5):

Notice how for the m=5 segmentation, the turquoise balloon on the left is overridden by similar values that surround it. In addition, the sky on the left is segmented into four colors. The fuzzy membership values being too high causes a problem in showing what color is in a particular area.

We developed a visualization to show the level of uncertainty of the class chosen for each pixel. If it is more uncertain, we add more tint (white value) to the pixel. The results for m=1.5 & 5:

FLICM

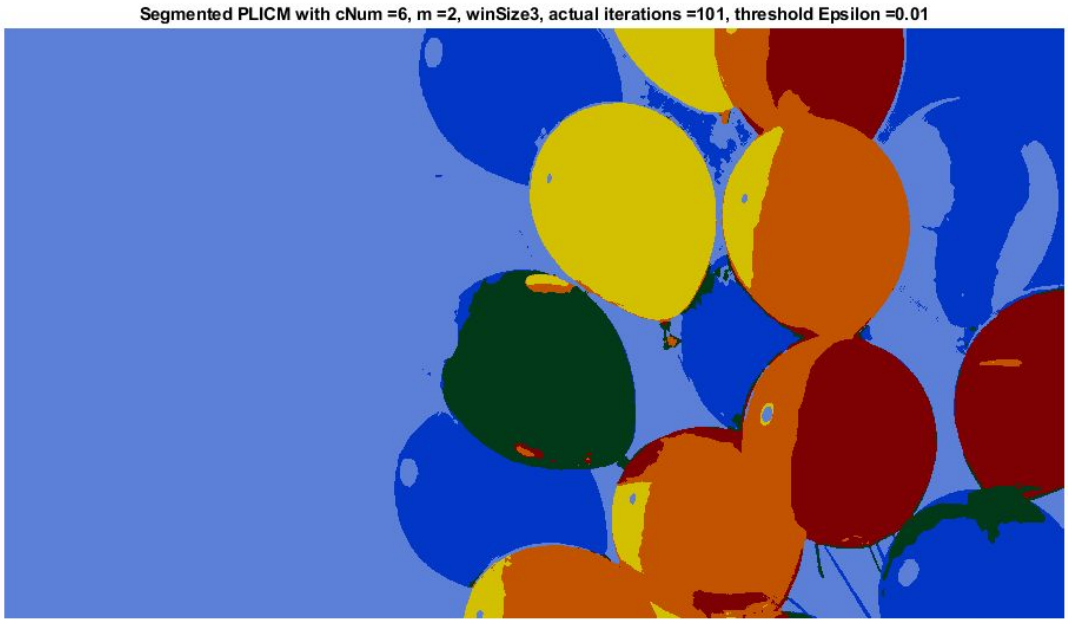
FLICM with m =5

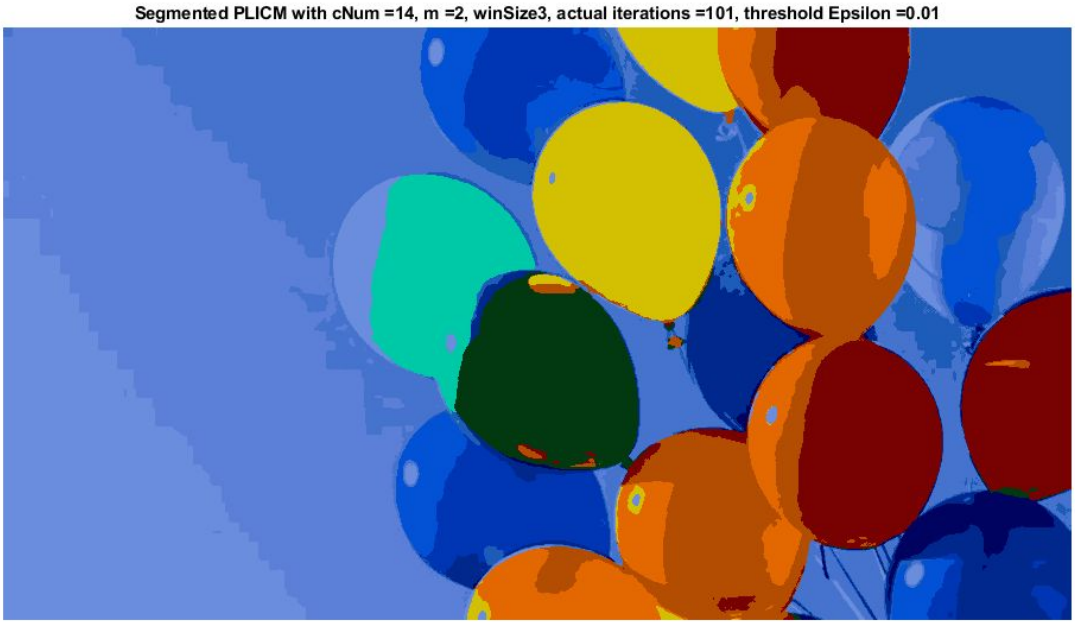
The m=5 visualization shows the effect of the fuzzifier. The maximum uncertainty that is chosen of the classes is much lower than it would be while using a lower fuzzifier - and so it is harder for the algorithm to distinguish between clustered areas. We predicted, given these results, that m=7 would be a mostly blue and orange image because those colors are most prevalent. The following image shows we were correct:

Segmented FLICM with cNum =11, m =7, winSize3, actual iterations =101, threshold Epsilon =0.01

Effect Of Changing Cluster Number

Testing the number of clusters yields important results. Because it was found in [3] that the optimal number of clusters for this image was 11, we used a standard of c=11 for most testing. The following shows PLICM with 6 and 14 clusters, respectively.





The general accuracy of segmentation was not particularly affected. There were, however, simply more colors in the resultant image for c=14. There are three classes for the sky as opposed to one, and balloons have slightly more colors where shadows were present.

Conclusion

In our project, we tested FCM, FLICM, PLICM, kmeans, kmeans with local information, and mean shift clustering methods. With respect to clustering we found that, in general, local/spatial information tends to allow for better segmentation with adequate window size, but kmeans can still perform well with no spatial information. PLICM and FLICM were better at dealing with noisy images with large enough window sizes but struggled with too small window sizes. PLICM was best with noise because it does not need membership values to sum to one - hence noisy pixels will receive less membership than with FLICM. When using FLICM and PLICM, the selection of m is important just picking a value of 2 does not always provide the best performance. Depending on the problem, more or less fuzziness and uncertainty may be desired in the image. Choosing cluster number will always be a difficult decision with algorithms such as FLICM and PLICM, as one needs to experiment or "guess and check" to find the right number. Accuracy of segmentation is not *necessarily* affected by cluster number selection, but it is possible that, like in our result, a desired object is clustered into the background due to color similarities and an insufficient number of classes.

Code:

Well documented, structured, modular program listings.

FCM Code:

```
clear;
clc;

imIn = imread('ballons.jpeg');
imIn= imnoise(imIn, 'salt & pepper', 0.001);

imIn = double(imIn);

[row, col, channel] = size(imIn);

imIn = reshape(imIn, row*col*channel, 1);

[center, U] = fcm(imIn,11);

[~, label] = max(U, [], 1);

imOut = reshape(center(label,:), row, col, channel);

figure,
imshow(uint8(imOut), [0 255]);
title('fcm C=11')
```

FLICM Code:

```
function [imOut,C,U,iter] = My_FLICM( img, cNum, m, winSize, maxIter, thrE )
%My_FLICM  This function performs Fuzzy Local Information C Means
%image clustering/segmentation
% img       - the image to segment
% cNum      - the number of clusters
% m         - the fuzzifier [1-inf)
% winSize   - the size of the sliding pixel window winSizexWinsize
% maxIter   - the maximum number of iterations before forced "convergence"
% thrE      - the number used to determine convergence
if nargin < 6
    thrE = 1.0e-2;
end

if nargin < 5
    maxIter = 100;
end

if nargin < 4
    winSize = 3;
end

if nargin < 3
    m = 2;
end

if nargin < 2
    cNum = 2;
end

%cast image into double for calculation purposes
image = double(img);
```

```
%initialization
iter = 0;

[row,col,channel] = size(image);

totalPixel = row*col;

%initialize degree of membership matrix based on the image dimension
U = rand( row, col, cNum-1 )*(1/cNum);

%initialize degree members with it channels (channels = numCluster)
U(:,:,cNum) = 1 - sum(U,3);

%reshape the image with row = row*col and col=channel
X = reshape(image, row*col, channel);

%reshape the degree of membership with the same size as the reshaped imageX
P = reshape(U, row*col, cNum);

%padding height and weight for filter -> to maintain the size of the image
h_pad = round((winSize-1)/2);
w_pad = round((winSize-1)/2);

%initialize degree of membership with padding
U_padded = ones(row+2*h_pad, col+2*w_pad, cNum);
%initialize the
DD_padded = zeros(row+2*h_pad, col+2*w_pad, cNum);
J_prev = inf;
J = [];
while true
   iter = iter + 1;
   if iter>maxIter
       break;
   end

   %degree of membership*m m is parameter that can be pick from 1 to inf
   t = P.^m;

   %calculate cluster center
   C = (t'*X)./(sum(t)'*ones(1, channel));

   %calculate the distance from data x to cluster center
   dist = sum(X.*X, 2)*ones(1, cNum) + (sum(C.*C, 2)*ones(1, totalPixel))'-(2*X*C');

   %reshape
   DD_padded(1+h_pad:row+h_pad, 1+w_pad:col+w_pad, :) = reshape(dist, row, col, cNum);

   U_padded(1+h_pad:row+h_pad, 1+w_pad:col+w_pad, :) = U;


   %calculate Fuzzy factor G
   GG = zeros(row, col, cNum);
   for ii = -h_pad:h_pad
       for jj = -w_pad:w_pad
           if ii~=0&&jj~=0
               GG = GG + 1/(1+ii*ii+jj*jj)*(1-U_padded(1+h_pad+ii:row+h_pad+ii,
1+w_pad+jj:col+w_pad+jj, :)).^m.*DD_padded(1+h_pad+ii:row+h_pad+ii,
1+w_pad+jj:col+w_pad+jj, :);
           end
       end
   end

   G = reshape(GG, row*col, cNum);
```

```
   rd = dist + G;

   t2 = (1./rd).^(1/(m-1));

   P = t2./(sum(t2, 2)*ones(1, cNum));

   U = reshape(P, row, col, cNum);

   J_cur = sum(sum((P.^m).*dist+G))/totalPixel;

   J = [J J_cur];
%    Check for convergence
   if norm(J_cur-J_prev) < thrE
       break;
   end

   J_prev = J_cur;

end

[~, label] = max(P, [], 2);
% Send the data to our vizualizer!
imOut = reshape(C(label, :), row, col, channel);
title = strcat('FLICM with m = ', num2str(m));
fuzzyIm(imOut, U, label, title);
```

PLICM Code:

```
function [imOut,C,U,iter] = My_PLICM( img, cNum, m, winSize, maxIter, thrE )
%My_PLICM  This function performs Possibilistic Local Information C Means
%image clustering/segmentation
% img       - the image to segment
% cNum      - the number of clusters
% m         - the fuzzifier [1-inf)
% winSize   - the size of the sliding pixel window winSizexWinsize
% maxIter   - the maximum number of iterations before forced "convergence"
% thrE      - the number used to determine convergence

img = double(img);
[row, col, channel] = size(img);

if nargin < 6
    thrE = 1.0e-2;
end

if nargin < 5
    maxIter = 100;
end

if nargin < 4
    winSize = 7;
end

if nargin < 3
    m = 2;
end

if nargin < 2
    cNum = 2;
end
% We use this seed to make sure results are equal always
rng(1);
```

```
%initialize C, U from FLICM
[imOut_FLICM,C,U,iter_FLICM] = My_FLICM(img, cNum, m, winSize, maxIter, thrE);
% Display FLICM results for ease of testing
title1 = strcat('Segmented FLICM with cNum = ', num2str(cNum), ', m = ', num2str(m),
', winSize ', num2str(winSize), ', actual iterations = ', num2str(iter_FLICM), ',
threshold Epsilon = ', num2str(thrE));
figure,
imshow(uint8(imOut_FLICM),[0 255]),
title(title1);

eta=ones(1,cNum)*3.5;

iter = 0;

totalPixel = row*col;

X = reshape(img, row*col, channel);

P = reshape(U, row*col, cNum);

h_pad = round((winSize-1)/2);
w_pad = round((winSize-1)/2);

%initialize degree of membership with padding
U_padded = ones(row+2*h_pad, col+2*w_pad, cNum);
%initialize the
DD_padded = zeros(row+2*h_pad, col+2*w_pad, cNum);
C_prev = inf;
C_temp = [];
J_prev = inf;

while true
   iter = iter + 1;
   if iter>maxIter
       break;
   end

   %degree of membership^m
   t = P.^m;

   %calculate cluster center
   C = (t'*X)./(sum(t)'*ones(1, channel));

   %calculate the distance from data x to cluster center
   dist = sum(X.*X, 2)*ones(1, cNum) + (sum(C.*C, 2)*ones(1, totalPixel))'-(2*X*C');

   %reshape
   DD_padded(1+h_pad:row+h_pad, 1+w_pad:col+w_pad, :) = reshape(dist, row, col, cNum);

   U_padded(1+h_pad:row+h_pad, 1+w_pad:col+w_pad, :) = U;


   %calculate Fuzzy factor G
   GG = zeros(row, col, cNum);
   for ii = -h_pad:h_pad
       for jj = -w_pad:w_pad
           if ii~=0&&jj~=0
               GG = GG + 1/(1+ii*ii+jj*jj)*(1-U_padded(1+h_pad+ii:row+h_pad+ii,
1+w_pad+jj:col+w_pad+jj, :)).^m.*DD_padded(1+h_pad+ii:row+h_pad+ii,
1+w_pad+jj:col+w_pad+jj, :);
           end
       end
   end
```

```
   G = reshape(GG, row*col, cNum);

   rd = dist + G;

   %degree of membership
   P = 1./(ones(1, cNum)+(rd./eta).^(1/(m-1)));

   U = reshape(P, row, col, cNum);

   %disortion
   J_cur = (sum(sum(((P.^m).*dist)+G))/totalPixel)+ (sum(eta)*sum((ones(1, cNum) -
P).^m));

   C_temp = [C_temp C];
%    Check convergence
   if norm(J_cur-J_prev) < thrE
       break;
   end
   J_prev = J_cur;

    C_prev = C;

end
% class is set to the max fuzzy membership
[~, label] = max(P, [], 2);

imOut = reshape(C(label, :), row, col, channel);

end
```

Fuzzy Visualization Code:

```
function fuzzyIm( imOut, U, label, stringType)
%fuzzyIm  Converts a crisp segmented image to a fuzzy representation
%using each pixels cluster membership value and tinting.
%Pixels become lighter the less sure it is that they belong to their
%assigned cluster.

%needs ImOut, U, label
%imOut is an array HxWx3 that makes up a true color rgb image
%label is a single column with all membership labels
%U is a reshaped array with membership values for every pixel over a number of
channels equal to the number of clusters

%uses size of image array to get number of rows and columns
[row,col,channel] = size(imOut);

%cycles through every pixel in all rows and columns
for Col=1:col
    for Row=1:row

            %gets the red green and blue value for a pixel
        R_pixel=imOut(Row,Col,1);
            G_pixel=imOut(Row,Col,2);
            B_pixel=imOut(Row,Col,3);

            %uses the columns of labels to reshape an array
            %matches labels to pixel position
        L = reshape(label,row,col);

            %gets membership value for pixel according to label
        mem = 1-U(Row,Col,L(Row,Col));
```

```
            %changes rgb value for pixel using a 1:2:3 ratio to tint it
            %larger adjustments make pixel lighter
        R_pixel = R_pixel + (mem * .5 * (255 - R_pixel));
        G_pixel = G_pixel + (mem * 1 * (255 - G_pixel));
        B_pixel = B_pixel + (mem * 1.5 * (255 - B_pixel));

            %rebuilds image using tinted pixels
        imFuz(Row,Col,1)=R_pixel;
        imFuz(Row,Col,2)=G_pixel;
        imFuz(Row,Col,3)=B_pixel;

    end
end

%shows the fuzzy representation image
figure,
imshow(uint8(imFuz),[0 255]),
title(stringType);
```

References

[1]Peeples, Joshua, et al. “Possibilistic Fuzzy Local Information C-Means with Automated Feature Selection for Seafloor Segmentation.” *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXIII*, 2018.

[2]Krinidis, Stelios, and VassiliosChatzis. “A Robust Fuzzy Local Information C-Means Clustering Algorithm.” *IEEE Transactions on Image Processing*, vol. 19, no. 5, 2010, pp. 1328–1337.

[3] Wu, Wenlong, et al. “Sequential Possibilistic One-Means Clustering with Dynamic Eta.” *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018.

[4]Runkler, Thomas A., and James M. Keller. “Sequential PossibilisticOne-Means Clustering.” *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017.

[5] Celik,Turgay, and HweeKuanLee. “Comments on ‘A Robust Fuzzy Local Information C-Means Clustering Algorithm.’” *IEEE Transactions on Image Processing*, vol. 22, no. 3, 2013, pp. 1258–1261.

[6] Reference for FLICM code: Krindis, S., Chatsize V. retrieved from https://github.com/chenlovep/biye/tree/b9139177f1874572101200045e0f83c913ed7140/模糊聚类program/other%20FCM/FLICM/46273147FLICM-FLICM-模糊局部信息FCM-C代码（原作者）/46273147FLICM-FLICM-模糊局部信息FCM-20170103（原作者）