

<b>Students Details</b>	
<b>Module Name:</b>	Digital Image Processing
<b>Module Lecturer/ Course Coordinator:</b>	Mr. S.C. Munasingha
<b>Department:</b>	School of Computing
<b>Submission Due on :</b>	12 <sup>th</sup> March 2023
<b>Type of Coursework:</b>	Group
<b>Title of the Coursework:</b>	End Module Group Assessment



***Students Details:***

	<b>Student No.</b>	<b>Student Name</b>
01	COHDSE221F-004	T. A. A. Sudaraka
02	COHDSE221F-015	K. Sanuga Lakdinu Kuruppu
03	COHDSE221F-021	A.D.C. Janaka
04	COHDSE221F-023 (Group Leader)	D.D. Thimbiripalage
05	COHDSE221F-034	P.H.M.W. Senevirathna

**Office use only :**

*Date Stamp Required of the Department*

**NATIONAL INSTITUTE OF BUSINESS MANAGEMENT  
HIGHER DIPLOMA IN SOFTWARE ENGINEERING  
END MODULE ASSESSMENT**

**DIGITAL IMAGE PROCESSING**

**A New Criterion for Automatic Multilevel Thresholding**

**SUBMITTED BY**

<b>T. A. A. Sudaraka</b>	<b>COHDSE221F-004</b>
<b>K. Sanuga Lakdinu Kuruppu</b>	<b>COHDSE221F-015</b>
<b>A.D.C. Janaka</b>	<b>COHDSE221F-021</b>
<b>D.D. Thimbiripalage</b>	<b>COHDSE221F-023 (Group Leader)</b>
<b>P.H.M.W. Senevirathna</b>	<b>COHDSE221F-034</b>

**Date of Submission:** ..... 12<sup>th</sup> March 2023

# Table of Contents

<b>Introduction.....</b>	<b>4</b>
<b>Core Concepts .....</b>	<b>5</b>
1. Maximum entropy criterion for bilevel thresholding: .....	5
2. MEC for multilevel thresholding: .....	5
3. Proposed maximum correlation criterion for bilevel thresholding: .....	6
4. Proposed automatic multilevel thresholding using MCC: .....	6
<b>Mathematical Background.....</b>	<b>8</b>
1. MEC for bilevel thresholding: .....	8
2. MCC for bilevel thresholding: .....	10
3. Automatic multilevel thresholding using (MCC): .....	11
<b>Important Code Blocks and Explanation .....</b>	<b>14</b>
1. Maximum entropy criterion for bilevel thresholding: .....	15
2. Maximum entropy criterion for multilevel thresholding: .....	17
3. Maximum correlation criterion for bilevel thresholding: .....	20
4. Maximum correlation criterion for Automatic multilevel thresholding: .....	22
<b>Image Results .....</b>	<b>27</b>
1. Maximum entropy criterion for bilevel thresholding: .....	27
2. Maximum correlation criterion for bilevel thresholding: .....	28
3. Maximum entropy criterion for multilevel thresholding: .....	29
4. Maximum correlation criterion for automatic multilevel thresholding: .....	30
<b>REFERENCES.....</b>	<b>31</b>

## Introduction

In digital image processing, one of the most important areas is image segmentation. The use of this field encompasses a wider range of digital industries specifically in industries related to computer vision and automations. Conventionally, we have different sort of approaches to determine threshold values and we can apply that values to regenerate the thresholder (Segmented) image using those methods such as Maximum entropy criterion.

However, when it comes to efficiency of those operations with previous approaches, these methods have some sort of drawbacks.

- Computational complexity is high.
- Need more resources (Time & CPU, RAM).
- Some parameters (Classification number) must be given by a supervision.

According to that, as a solution, in this research paper, they have proposed an efficient approach (Maximum correlation criterion) of determining threshold values for bilevel thresholding and then, they have proposed this new method into multilevel thresholding with automatic multilevel thresholding.

## Core Concepts

Before jumping into proposed method (MCC), it is better to review conventional approaches MEC for bilevel thresholding and multilevel thresholding.

### 1. Maximum entropy criterion for bilevel thresholding:

In this approach, it is going to calculate a single threshold value and it can be assumed that threshold value can separate the entire image into two segments.

- Background
- Foreground (Object area)

It can be assumed that all the gray values which are less than the selected threshold value belongs to the background, while the all gray values which are greater than the selected threshold value belongs to the object of that image.

Therefore, how can the optimal threshold value be found?

The basic concept here is, this approach is going to find the optimal threshold value such that the total amount of information provided by the background and the object is maximized. In order to do that, information of that image must be quantified. Therefore, this approach is going to use entropy concept for that. (This is going to find the threshold value where the total entropy provided by the object and the background is maximized).

### 2. MEC for multilevel thresholding:

In this approach, it is going to use the same concept call maximum entropy criterion in bilevel thresholding, but for multilevel thresholding, multiple threshold values must be found which can be used to segment original image into several objects and background.

- Background
- First object (threshold value 01)
- Second object (threshold value 02)

When there is multiple threshold values or classification number rather than having single threshold value, the discrepancy between original image and the thresholded image is going to decrease while required bits to represent the image is increased. Therefore, there must be a compromise between these two factors. But in this MEC approach, that will not be addressed and that will cause to

increase the time required to find the threshold values. Specially, in this case classification number (How many threshold values must be found) is given by a supervision.

Basic concept here is, according to the classification number provided the supervision, Total combinations of threshold values can be calculated and those values are used to find total information provided by the thresholder image. Likewise MEC for bilevel thresholding, entropy is used to find the total information of that image. If this approach gives us the maximum entropy among all the threshold value combinations, that combination is the optimal value for segmenting the original image.

Specially, in both MEC and MEC for multilevel thresholding, there are several drawbacks when it comes to efficiency.

- Time required to find the optimal threshold value combination in MEC for multilevel thresholding is very high.
- Complexity and logarithmic operations in finding entropy in MEC is larger, compared to proposed MCC method.

### **3. Proposed maximum correlation criterion for bilevel thresholding:**

Rather than using entropy to calculate the information provided by the thresholder image, in this approach, it uses correlation values. Therefore, optimal threshold values are selected when the total correlation of the image is maximized.

As a solution, this will decrease the number of logarithmic operations  $m^2 + m$  to  $m$ . Therefore, it will give us huge differences between computational complexity while the thresholder image are pretty much same.

### **4. Proposed automatic multilevel thresholding using MCC:**

As the previous method, when increasing the classification number, discrepancy between the original image and the thresholder image is decreased while image size is increased. Therefore, there must exist a compromised between these two factors. In this proposed approach, they introduce a cost function which takes into account these two factors.

Therefore, in this concept, Cost function value must be minimized in order to find a better compromised between two factors. Using this approach, we can go

through from the beginning by each class and find each class variance and the highest variance class. It then can be further divided into two classes if previously classified number is not the optimal value. Therefore, it is not necessary to go through every possible number for classification because of, while finding cost function value for each classification number, cost value will decrease up to optimal classification number. When it will be going to increase, loop can be wrapped up and it will give us the optimal classification number automatically and threshold values also, while computational complexity and time is decreasing.

## Mathematical Background

Equation explanation:

### 1. MEC for bilevel thresholding:

There are two distributions.

- Gray values from 0 up to threshold value (B)
- Threshold value + 1 to 255 (if it is 8-bit depth gray image) (F)

$$B \equiv \left\{ \frac{P_0}{P(s)}, \frac{P_1}{P(s)}, \dots, \frac{P_{s-1}}{P(s)} \right\}$$
$$F \equiv \left\{ \frac{P_s}{1 - P(s)}, \frac{P_{s+1}}{1 - P(s)}, \dots, \frac{P_{m-1}}{1 - P(s)} \right\}$$

B = Probability distribution of background

F = Probability distribution of foreground

P<sub>1</sub> = 1<sup>st</sup> Gray level probability

S = Assumed threshold value (0 - 255)

P(s) = Total probability up to (s-1) gray level

Calculating each gray level probability:

$$P(s) = \sum_{i=0}^{s-1} P_i$$

P<sub>i</sub> = Probability distribution of i<sup>th</sup> gray level

$$P_i = \frac{f_i}{N \times M}$$



According to MEC, entropy can be calculated as follows.

$$TE_{(s)} = EB_{(s)} + EF_{(s)}$$

1

$TE_{(s)}$  = Total entropy of image when threshold value = S

$EB_{(s)}$  = Total entropy of background when threshold value = S

$EF_{(s)}$  = Total entropy of foreground when threshold value = S

And entropy can be defined:

$$Entropy = - \sum_{i=0}^n P_i \ln P_i$$

If this will be substituted to equation (01):

$$EB_s = - \sum_{i=0}^{s-1} \frac{P_i}{P(s)} \ln \frac{P_i}{P(s)}$$

$$EF_s = - \sum_{i=s}^{m-1} \frac{P_i}{1 - P(s)} \ln \frac{P_i}{1 - P(s)}$$

$$TE_s = - \sum_{i=0}^{s-1} \frac{P_i}{P(s)} \ln \frac{P_i}{P(s)} - \sum_{i=s}^{m-1} \frac{P_i}{1 - P(s)} \ln \frac{P_i}{1 - P(s)}$$

In this equation, natural logarithm is inside the sigma notation, that means, there are many logarithmic operations when using this method.

In order to find the maximum entropy to get optimal threshold value,

$$TE_{(s^*)} = \max TE_{(s)}$$

$S^*$  = Optimal threshold value

## 2. MCC for bilevel thresholding:

In this approach, instead of using entropy to calculate the optimal value, Correlation can be adopted.

$$Correlation = - \ln \sum_{i=1}^n P_i^2$$

According to MCC, correlation can be calculated as follows,

$$TC_{(s)} = CB_{(s)} + CF_{(s)}$$

2

$TC_{(s)}$  = Total correlation of image when threshold value = S

$CB_{(s)}$  = Total correlation of background when threshold value = S

$CF_{(s)}$  = Total correlation of foreground when threshold value = S

If this will be substituted to equation (02):

$$\begin{aligned}
 CB_s &= - \ln \sum_{i=0}^{s-1} \left( \frac{P_i}{P(s)} \right)^2 \\
 CF_s &= - \ln \sum_{i=s}^{m-1} \left( \frac{P_i}{1 - P(s)} \right)^2 \\
 TC_s &= - \ln \sum_{i=0}^{s-1} \left( \frac{P_i}{P(s)} \right)^2 - \ln \sum_{i=s}^{m-1} \left( \frac{P_i}{1 - P(s)} \right)^2
 \end{aligned}$$

In this equation, natural logarithm is outside the sigma notation, that means, this equation will use less number of logarithmic operations than MEC, that will decrease the computational complexity of the calculation drastically.

In order to find the maximum correlation to get optimal threshold value,

$$TC_{(s^*)} = \max TC_{(s)}$$

$S^*$  = Optimal threshold value

### 3. Automatic multilevel thresholding using (MCC):

As mentioned earlier, in this approach, there are two factors to be considered.

- Discrepancy between the original and threshold image
- Number of bits required to represent the threshold image.

In this, these two factors will be addressed by the cost function as follows.

$$C(k) = \rho \sqrt{Dis(k)} + (\log_2 k)^2$$

$k$  = Classification number (Number of classes)

$C(k)$  = Total cost when we have  $k$  number of classifications

$\rho$  = Positive weighting constant (In this, 0.8)

$\sqrt{Dis(k)}$  = Cost for discrepancy between original image and threshold image

$(\log_2 k)^2$  = Cost for image size

Therefore, for keeping a compromise between these two factors, they proposed to get minimum value for cost function( $C(k)$ ). That  $k$  value is the optimal classification number for each image.

$$C(k^*) = \min C(k)$$

$k^*$  = Optimal classification number

To find  $dis(k)$  value for each classification number:

$$Dis(k) = \sum_{i=1}^k \omega_{k,i} \times \sigma_{k,i}^2$$

3

- $k$  = Classification number (Number of classes)  
 $\omega_{k,i}$  = Total probability of  $i^{\text{th}}$  class when having  $k$  number of classes  
 $\sigma_{k,i}^2$  = Variance of  $i^{\text{th}}$  class when having  $k$  number of classes

To find Total probability and variance, following equations can be adopted.

$$\begin{aligned}
 \omega_{k,i} &= \sum_{r=s_{k,i-1}}^{s_{k,i}-1} P_r \\
 \mu_{k,i} &= \sum_{r=s_{k,i-1}}^{s_{k,i}-1} r \times (P_r / \omega_{k,i}) \\
 \sigma_{k,i}^2 &= \sum_{r=s_{k,i-1}}^{s_{k,i}-1} (r - \mu_{k,i})^2 \times (P_r / \omega_{k,i})
 \end{aligned}$$

- $s_{k,i}$  = Threshold value of  $i^{\text{th}}$  class when having  $k$  number of classes  
 $s_{k,i-1}$  = Threshold value of  $(i-1)^{\text{th}}$  class when having  $k$  number of classes  
 $s_{k,i} - 1$  = Threshold value of  $i^{\text{th}}$  class -1 when having  $k$  number of classes  
 $\mu_{k,i}$  = Mean value of  $i^{\text{th}}$  class when having  $k$  number of classes

Finally, these values can be substituted to equation number (03) to get the  $\text{dis}(k)$ , and that will help to get the minimum cost value. After that, according to our core concepts in automatic multilevel thresholding, optimal classification number and all the threshold values can be obtained.

## Important Code Blocks and Explanation

Image Reading and getting probability distribution for each bin value for every criterion is follows. Every process is done as a function. Therefore, user only need to read the image and call the suitable function with necessary parameters.

```
%Image reading
colorImage = img;

%Converting to gray image
grayImage = rgb2gray(colorImage);

%Calculating image size
[row, col] = size(grayImage);
totalPixelCount = row * col;

% Create a frequency array of size 256
frequency = zeros(1, 256);

% Iterate over grayscale image matrix
% for every possible intensity value
% and count them
for i = 1 : 256
    frequency(i) = sum(sum(grayImage == i-1));
end

%Creating Probability Distribution using frequencies
probabilityDistribution = frequency / totalPixelCount;
```

## 1. Maximum entropy criterion for bilevel thresholding:

```
function [thresholdValue] = mecBilevel(img)
    %Image reading
    colorImage = img;
```

This function needs only input image as the parameter.

```
% ----- Entropy Calculation -----

entropyBackground = 0;
entropyFrontground = 0;
entropy = 0;
grayValue = 0;

for z = 0 : 255

    %Getting P(s) for each gray value
    totProbS = sum(probabiltyDistribtion(1:z));

    entropyFrontground = 0;
    entropyBackground = 0;

    %Getting total entropy of background( <=gray value)
    for x = 1 : z
        if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0
            fractionback = probabiltyDistribtion(x)/totProbS;
            if fractionback ~= 0
                entropyBackground = entropyBackground
+fractionback.* log(fractionback);
            end
        end
    end

    %Getting total entropy of object( >=gray value)
    for y = z+1 : 256
        if probabiltyDistribtion(y) ~= 0 && totProbS ~= 1
            fractionfront = probabiltyDistribtion(y)/(1 -
totProbS);
            if fractionfront ~= 0
                entropyFrontground = entropyFrontground +
fractionfront.* log(fractionfront);
            end
        end
    end
end
```

```

        %Getting maximum entropy for each gray value and getting that
gray
        %value
        if entropy <= -(entropyFrontground + entropyBackground)
            entropy = -(entropyFrontground + entropyBackground);
            grayValue = z;
        end
    end
end

```

Here, we have iterated through all the possible threshold values from 0 to 255 in order to find the optimal value (Main loop). In this, we have divided the process into two segments. First loop inside the main loop is used to find entropy value for background while second loop is used to find the entropy for foreground of the image. We have used necessary validations to bypass the exceptions provided when the abnormal situations happen. In final if condition, we've checked the total entropy with previous total entropy and, if the new entropy value is greater than the previous one, we have assigned new threshold value and keep new maximum entropy value to check incoming gray values.

```

%Generating thresholded image (Background -> black, object -> white)
last = zeros(row,col);
for a = 1 : row
    for b = 1 : col
        if grayValue > grayImage(a,b)
            last(a,b) = 255;
        end
    end
end

%Generating thresholded image (Background -> white, object ->
black)
last = 255 - last;

%Return threshold value
thresholdValue = grayValue;

%Show image

figure,
subplot(1,3,1), imshow(img), title('Original Image');
subplot(1,3,2), imshow(grayImage), title('Gray Image Image');

```



```
subplot(1,3,3), imshow(last), title('Segmented Image(MEC
Bilevel)');
```

Finally, we have developed the thresholder image using previous threshold value and showing the segmented image and return the threshold value.

## 2. Maximum entropy criterion for multilevel thresholding:

```
function [arrThresholdValues] = mecMultilevel(img,count)
%Image reading
colorImage = img;
```

This function needs only input image and the classification number provided by the supervision as the method parameters.

```
function combs = nmultichoosek(values, k)
% Return number of multisubsets or actual multisubsets.
if numel(values)==1
    n = values;
    combs = nchoosek(n+k-1,k);
else
    n = numel(values);
    combs = bsxfun(@minus, nchoosek(1:n+k-1,k), 0:k-1);
    combs = reshape(values(combs),[],k);
end
end
```

This is the function which is called automatically when calling the main function. This function is used to get all possible combinations of threshold values according to the classification number provided by the supervision. This will return the array of possible combinations of threshold values.

```

% ----- Entropy Calculation -----
-

numberOfThresholds = count;
numberOfClasses = numberOfThresholds + 1;
arrThresholdValues = zeros(1,numberOfThresholds);
thresholds = zeros(1, numberOfThresholds);
entropy = 0;
grayValue = 0;

combinations = nmultichoosek(1:255,numberOfThresholds);
[cr,cc] = size(combinations);

for j = 1 : cr

    entropy1 = 0;
    entropy3 = 0;
    entropy2 = 0;

    for g = 1 : cc
        thresholds(g) = combinations(j,g);
    end

    % Sort the threshold values in ascending order
    thresholds = sort(thresholds);

    for f = 1 : cc + 1
        if f == 1
            %Getting entropy value to first class
            totProbS =
sum(probabiltyDistribtion(1:thresholds(1,1)));
            for x = 1 : thresholds(1,1)
                if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0
                    fractionback =
probabiltyDistribtion(x)/totProbS;
                    if fractionback ~= 0
                        entropy1 = entropy1 + fractionback.*
log(fractionback);
                    end
                end
            end
        elseif f == cc + 1
            %Getting entropy value to last class
            totProbS = sum(probabiltyDistribtion(thresholds(1,f -
1):255));
            for x = thresholds(1,f-1) : 255
                if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0

```

```

                                fractionback =
probabiltyDistribtion(x)/totProbS;
                                if fractionback ~= 0
                                    entropy3 = entropy3 + fractionback.*
log(fractionback);
                                end
                            end
                        end
                    else
                        %Getting entropy value to middle classes
                        totProbS = sum(probabiltyDistribtion(thresholds(1,f-
1):thresholds(1,f)));
                        for x = thresholds(1,f-1) : thresholds(1,f)
                            if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0
                                fractionback =
probabiltyDistribtion(x)/totProbS;
                                if fractionback ~= 0
                                    entropy2 = entropy2 + fractionback.*
log(fractionback);
                                end
                            end
                        end
                    end
                end

            end

            %Getting maximum entropy for each combination and getting that
            %index
            if entropy <= -(entropy1 + entropy2 + entropy3)
                entropy = -(entropy1 + entropy2 + entropy3);
                grayValue = j;
            end
        end
    end
end

```

Here, after getting all the combinations, in the main loop we have iterated through all the combinations. Inside each iteration, we've once again iterated through each class in each combination in order to find the total entropy provided by all the classes. But here, we have divided that process into three segments.

- First class in every combination. (0 - first threshold value).
- Last class in every combination. (Last threshold value - 255).
- Middle classes in every combination.

After getting all the entropies for above three segments. In final if condition, we've checked the total entropy with previous total entropy and, if the new entropy value is greater than the previous one, we have assigned new combination as the final threshold values and keep new maximum entropy value to check with the incoming combinations.

### 3. Maximum correlation criterion for bilevel thresholding:

```
function [thresholdValue] = mccBilevel(img)

%Image reading
colorImage = img;
```

This function needs only input image as the parameter.

```
% ----- Correlation Calculation -----
-----

correlationBackground = 0;
correlationFrontground = 0;
correlation = 0;
grayValue = 0;

for z = 0 : 255

    %Getting P(s) for each gray value
    totProbS = sum(probabiltyDistribtion(1:z));

    correlationBackground = 0;
    correlationFrontground = 0;

    %Getting total correlation of background( <=gray value)
    for x = 1 : z
        if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0
            fractionback = (probabiltyDistribtion(x)/totProbS).^2;
            correlationBackground = correlationBackground +
fractionback;
        end
    end

    %Getting total correlation of object( >=gray value)
    for y = z+1 : 256
        if probabiltyDistribtion(y) ~= 0 && totProbS ~= 1
```

```

        fractionfront = (probabilityDistribution(y)/(1 -
totProbS)).^2;
        correlationFrontground = correlationFrontground +
fractionfront;
    end
end

    %Getting maximum correlation for each gray value and getting
that gray
    %value
    if correlationFrontground ~= 0 && correlationBackground ~= 0
        correlationFrontground = -log(correlationFrontground);
        correlationBackground = -log(correlationBackground);
        if correlation < correlationFrontground +
correlationBackground
            correlation = correlationFrontground +
correlationBackground;
            grayValue = z;
        end
    end
end
end

```

Here, we have iterated through all the possible threshold values from 0 to 255 in order to find the optimal value (Main loop). In this, we have divided the process into two segments. First loop inside the main loop is used to find correlation value for background while second loop is used to find the correlation for foreground of the image. We have used necessary validations to bypass the exceptions provided when the abnormal situations happen. In final if condition, we've checked the total correlation with previous total correlation and, if the new correlation value is greater than the previous one, we have assigned new threshold value and keep new maximum correlation value to check incoming gray values.

#### 4. Maximum correlation criterion for Automatic multilevel thresholding:

```
function [thresholdArray] = mccMultilevel(img)
%Image reading
colorImage = img;
```

This function needs only input image as the parameter.

```
%Getting best threshohld values
for k = 1 : 256
    discrepancyFirstClass = 0;
    discrepancyLastClass = 0;
    discrepancyMiddleClasses = 0;

    %Traverse through all the classed based on current thresholdvalues
    for a = 1 : k
        meanFirstClass = 0;
        meanLastClass = 0;
        meanMiddleClasses = 0;
        varLastClass = 0;
        varFirstClass = 0;
        varMiddleClasses = 0;

        if a == 1
            %Getting discrepany of first class
            totProbS =
sum(probabiltyDistribtion(1:thresholdValueArray(1,1)));

            %Getting mean value of first class
            for x = 1 : thresholdValueArray(1,1)
                if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0
                    meanFirstClass = meanFirstClass + x *
probabiltyDistribtion(x)/totProbS;
                end
            end

            %Getting variance value of first class
            for x = 1 : thresholdValueArray(1,1)
                if probabiltyDistribtion(x) ~= 0 && totProbS ~= 0
                    varFirstClass = varFirstClass + (x -
meanFirstClass).^2 * probabiltyDistribtion(x)/totProbS;
                end
            end
            discrepancyFirstClass = discrepancyFirstClass + totProbS *
varFirstClass;
```

```

        %Getting highest varied(Highest variance) class
        if varFirstClass >= previousHighetVar
            previousHighetVar = varFirstClass;
            previousHishetVarClass = a;
        end
    elseif a == k
        %Getting discrepany of last class
        totProbS =
sum(probabiltyDistription(thresholdValueArray(1,k-1):255));

        %Getting mean value of last class
        for x = thresholdValueArray(1,k-1) : 255
            if probabiltyDistription(x) ~= 0 && totProbS ~= 0
                meanLastClass = meanLastClass + x *
probabiltyDistription(x)/totProbS;
            end
        end

        %Getting variance value of last class
        for x = thresholdValueArray(1,k-1) : 255
            if probabiltyDistription(x) ~= 0 && totProbS ~= 0
                varLastClass = varLastClass + (x -
meanLastClass).^2 * probabiltyDistription(x)/totProbS;
            end
        end
        discrepanyLastClass = discrepanyLastClass + totProbS *
varLastClass;

        %Getting highest varied(Highest variance) class
        if varLastClass >= previousHighetVar
            previousHighetVar = varLastClass;
            previousHishetVarClass = a;
        end
    else
        %Getting discrepany of middle classes
        totProbS =
sum(probabiltyDistription(thresholdValueArray(1,a-
1):thresholdValueArray(1,a)));

        %Getting mean value of middle class
        for x = thresholdValueArray(1,a-1)
: thresholdValueArray(1,a)
            if probabiltyDistription(x) ~= 0 && totProbS ~= 0
                meanMiddleClasses = meanMiddleClasses + x *
probabiltyDistription(x)/totProbS;
            end
        end
    end
end

```

```

        %Getting variance value of middle class
        for x = thresholdValueArray(1,a-1)
: thresholdValueArray(1,a)
            if probabiltiyDistribtion(x) ~= 0 && totProbS ~= 0
                varMiddleClasses = varMiddleClasses + (x -
meanMiddleClasses).^2 * probabiltiyDistribtion(x)/totProbS;
            end
        end
        discrepencyMiddleClasses = discrepencyMiddleClasses +
totProbS * varMiddleClasses;

        %Getting highest varied(Highest variance) class
        if varMiddleClasses >= previousHighetVar
            previousHighetVar = varMiddleClasses;
            previousHishetVarClass = a;
        end
    end
end
end

```

Here, the main loop is used to iterate through all the possible classification number because it is not provided by the supervision. But it is not going to iterate via all the values from 1-256 because, before that value, best classification can be found. In the main loop, there are 3 segments in order to traverse through all the classes according to that classification number.

- First class for every classification number (0 - first threshold value).
- Last class for every classification number. (Last threshold value - 255).
- Middle classes for every classification number.

Apart from that, mean value, variance of that class is calculated to find the classes which has the highest variance in order to further divide that class if the optimal classification number is not found. And finally, this loop is used to find the total discrepancy of that classification number.



```

%Calculating Cost Function value and getting optimal number of
%thresholds
    if costFunctionValue > 0.8 *
sqrt(discrepancyFirstClass+discrepancyMiddleClasses+discrepancyLastClass) + (log2(k))^2

        %if new cost funtion value is less than previous one
        %that means we can divide that classes further using highest
        %variance class
        costFunctionValue = 0.8 *
sqrt(discrepancyFirstClass+discrepancyMiddleClasses+discrepancyLastClass) + (log2(k)).^2;
        if previousHishetVarClass == 1
            thresholdValueArray(1,k+1) =
mccBilevel(probabiltyDistription(1:thresholdValueArray(1,previousHishetVarClass)),1,thresholdValueArray(1,previousHishetVarClass));
        elseif previousHishetVarClass == k-1
            thresholdValueArray(1,k+1) =
mccBilevel(probabiltyDistription(thresholdValueArray(1,previousHishetVarClass) : 255),thresholdValueArray(1,previousHishetVarClass),255);
        else
            thresholdValueArray(1,k+1) =
mccBilevel(probabiltyDistription(thresholdValueArray(1,previousHishetVarClass-1),thresholdValueArray(1,previousHishetVarClass)),thresholdValueArray(1,previousHishetVarClass-1),thresholdValueArray(1,previousHishetVarClass));
        end

        thresholdValueArray = sort(thresholdValueArray);

    else
        %if new cost funtion value is greater than previous one
        %that means previous number of thresholds is the optimal value
        break;
    end
end
thresholdArray = thresholdValueArray;

```

This code block is used to calculate the total cost value for each classification number and to get decisions based on that value. In this, if the total cost function value is less than the previous cost function value, that means this is not the optimal classification number and threshold values. Therefore, we have further divided the class which is having the highest variance that is calculated previously. In order to find the new threshold value in that class we have use

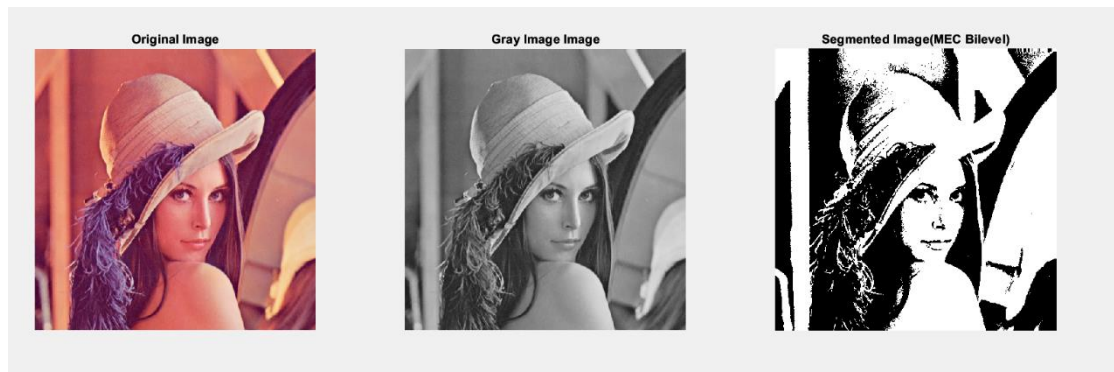
MCC method and each class is further divided into two parts like MCC in bilevel thresholding. To do that we've used the previously developed function. If the cost function value is not less than the previous cost function value, that means, cost function value is going to increase and previous classification number is the optimal classification number. Therefore, in that code block, we've stopped the iteration and display the classification number and threshold values.

## Image Results

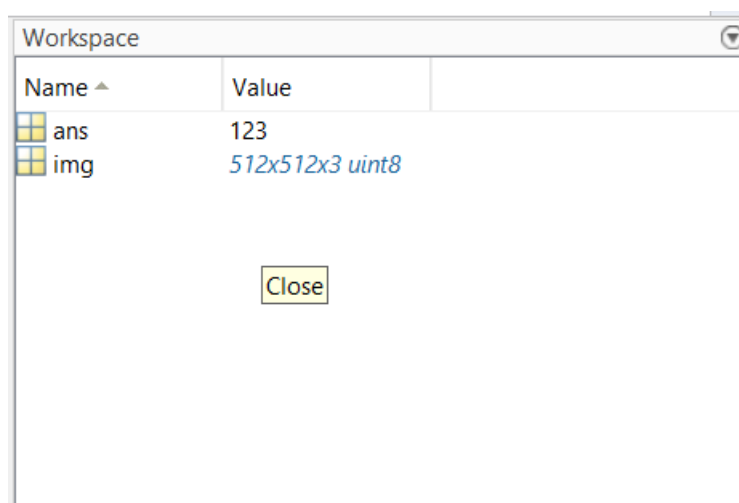
### 1. Maximum entropy criterion for bilevel thresholding:

Threshold value - 123

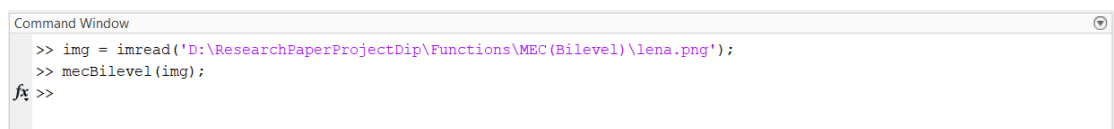
Output image:



Workspace:



Command Window:



## 2. Maximum correlation criterion for bilevel thresholding:

Threshold value - 121

Output image:



Workspace:

Workspace		
Name ^	Value	
ans	121	
img	512x512x3 uint8	

Command Window:

```
Command Window
>> img = imread('D:\ResearchPaperProjectDip\Functions\MCC (Bilevel)\lena.png');
>> mccBilevel(img);
fx >> |
```

### 3. Maximum entropy criterion for multilevel thresholding:

Given Classification number - 3.

Threshold value Array – [82,126,175]

Output image:



Workspace:

Workspace	
Name ^	Value
ans	[82,126,175]
img	512x512x3 uint8

Command Window:

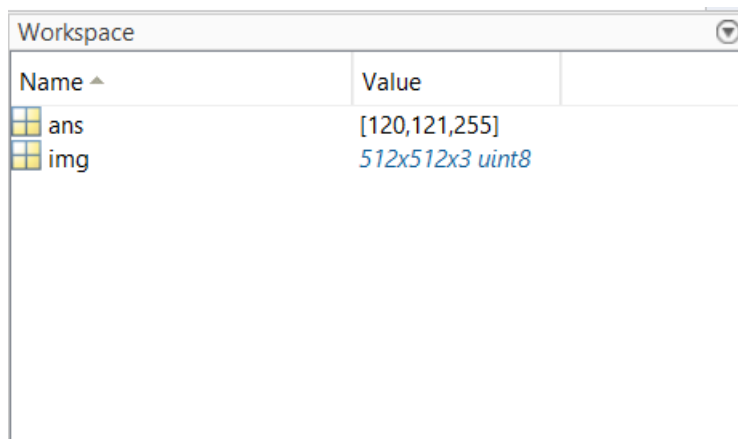
```
Command Window
>> img = imread('D:\ResearchPaperProjectDip\Functions\MCC (Bilevel)\lena.png');
>> mecMultilevel(img,3);
fx >>
```

#### 4. Maximum correlation criterion for automatic multilevel thresholding:

Threshold value Array – [120,121,255]

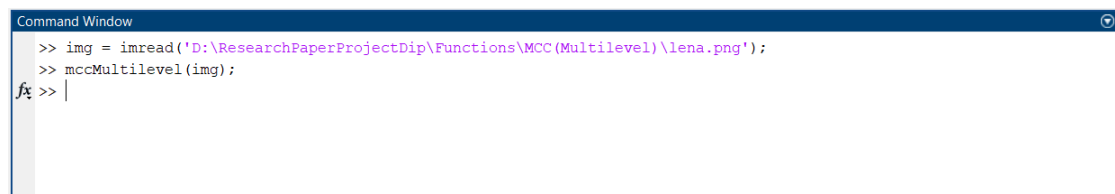
Output classification number – 3

Workspace:



Name ^	Value
ans	[120,121,255]
img	512x512x3 uint8

Command Window:



```
Command Window
>> img = imread('D:\ResearchPaperProjectDip\Functions\MCC (Multilevel)\lena.png');
>> mccMultilevel(img);
fx >> |
```

## REFERENCES

Gonzalez, C.R. and Woods, R.E. (2008) *Digital Image Processing*. New jersey: Pearson

Yen, J.C, Chang, F.J, Chang, S. (1995) *A new Criterion for Automatic Multilevel Thresholding*. China: National Tsing Hua University

Jiang, Y., Tsai, P., Hao, Z., Cao, L. (2014) *Automatic multilevel thresholding for image segmentation using stratified sampling and Tabu Search*. China: Jiangxi Agricultural University

## Task Delegation

COHDSE221F-004	<ul style="list-style-type: none"> <li>• Developing algorithms to understand the flow of control of the algorithm.</li> <li>• Software development in MATLAB (Multilevel thresholding in MEC).</li> </ul>
COHDSE221F-015	<ul style="list-style-type: none"> <li>• Understanding the research paper and mathematics behind it.</li> <li>• Software development in MATLAB (Bilevel thresholding in MEC).</li> </ul>
COHDSE221F-021	<ul style="list-style-type: none"> <li>• Understanding the research paper and mathematics behind it.</li> <li>• Software development in MATLAB (Bilevel thresholding in MCC).</li> </ul>
COHDSE221F-023 (Group Leader)	<ul style="list-style-type: none"> <li>• Developing algorithms to understand the flow of control of the algorithm.</li> <li>• Software development in MATLAB (Multilevel thresholding in MCC).</li> </ul>
COHDSE221F-034	<ul style="list-style-type: none"> <li>• Understanding the research paper and mathematics behind it.</li> <li>• Software development in MATLAB (Multilevel thresholding in MCC).</li> <li>• Preparing final report &amp; Presentation.</li> </ul>