Assignment08_20133096_HyunjaeLee

May 16, 2019

0.1 20133096 Hyunjae Lee

[Apply K-means algorithm to both image value and its spatial domain]

For a given input image (either gray or color), apply a K-means algorithm that is designed to take into consideration of both the image intensity and its spatial domain with varying parameters: the number of clusters and the trade-off between the intensity energy and the spatial energy.

The objective function is given by:

$\sum$

```
In [197]: # Basic variables
          Function_X = np.array(im_color)
          Function_X_Size = list(Function_X.shape)
          width = Function_X_Size[1]
          height = Function_X_Size[0]

          print('Width is {}, Height is {}'.format(width, height))

          threshold = 0.005

          def initialize(labelsize, k):
              #print(labelsize)
              return np.random.randint(k, size = labelsize)

          def Average(lst):
              return sum(lst) / len(lst)

          def centroid(Image,X_matrix, Y_matrix, Label_Array, K, width, height):

              for row in range(height):
                  for col in range(width):
                      for k in range(K):
                          if Label_Array[row][col] == k:
                              centroid_list[k][0] += Image[row][col][0] # R
                              centroid_list[k][1] += Image[row][col][1] # G
                              centroid_list[k][2] += Image[row][col][2] # B

                              # X_centroid
                              centroid_list[k][3] += X_matrix[row][col]
```

```
                    # Y_centroid
                    centroid_list[k][4] += Y_matrix[row][col]

                    count_list[k] += 1

    return centroid_list/count_list

def labeling(Image,avg, X_matrix, Y_matrix, Label_Array, K, width, height, ramdavalue)
    energy = 0
    for row in range(height):
        for col in range(width):

            temp = []

            for k in range(K):

                inputNumber = ((Image[row][col][0] - avg[k][0])**2
                               + (Image[row][col][1] - avg[k][1])**2
                               + (Image[row][col][2] - avg[k][2])**2)
                + ramdavalue * (((X_matrix[row][col] - avg[k][3])**2
                                 + (Y_matrix[row][col] - avg[k][4])**2))

                temp.append(inputNumber)

            Label_Array[row][col] = np.argmin(temp)

    # Calculate Energy
    for row in range(height):
        for col in range(width):
            for k in range(K):
                if(Label_Array[row][col] == k):
                    energy += ((Image[row][col][0] - avg[k][0])**2
                               + (Image[row][col][1] - avg[k][1])**2
                               + (Image[row][col][2] - avg[k][2])**2)
                + ramdavalue * (((X_matrix[row][col] - avg[k][3])**2
                                 + (Y_matrix[row][col] - avg[k][4])**2))

    energy /= (row*col*3)

    return Label_Array,energy

# Print average image
def create_average_image(average, mean_G, std_G,Label_Array,K):
    im_avg = np.zeros(((177,284,3))) # r,g,b,x,y

    for row in range(177):
        for col in range(284):
            for k in range(K):
                if(Label_Array[row][col] == k):
                    # back_whitening
                    im_avg[row][col][0] = ( average[k][0] * std_G[0] ) + mean_G[0]
                    im_avg[row][col][1] = ( average[k][1] * std_G[1] ) + mean_G[1]
                    im_avg[row][col][2] = ( average[k][2] * std_G[2] ) + mean_G[2]
    return im_avg

def scailing():
    # Two matrices where one matrix represents the horizontal index
    # and the other matrix represents the vertical index.
    X_axis_matrix = np.zeros((height,width))
    Y_axis_matrix = np.zeros((height,width))
    temp_Y = 0

    # Assigne each matrix
    ## Y_axis
    for eachY in Y_axis_matrix:
        for eachIndex in range(0, width):
            eachY[eachIndex] = temp_Y
        temp_Y += 1 / (height-1)

    ## X_axis
    temp_X = 0

    for num in range(0, width):
        temp_X = num
        if num > 0:
            temp_X /= (width-1)
        for eachX in X_axis_matrix:
            eachX[num] = temp_X


    return X_axis_matrix, Y_axis_matrix

def whitening(Function_X):

    # Get Mean
    mean = np.zeros(3) # r,g,b
    std = np.zeros(3) #r,g,b
    for eachRow in Function_X:
        mean[0] += eachRow[:,0].mean()
        std[0] += eachRow[:,0].std()
        mean[1] += eachRow[:,1].mean()
        std[1] += eachRow[:,1].std()
        mean[2] += eachRow[:,2].mean()
        std[2] += eachRow[:,2].std()

    mean /= height
```

```
        std /= height
        print('RGB mean is {}'.format(mean))
        print('RGB std is {}'.format(std))

        return ( Function_X - mean ) / std, mean, std
```

```
Width is 284, Height is 177
```

0.2 Example [1] K = 4 , ramda = 0.1

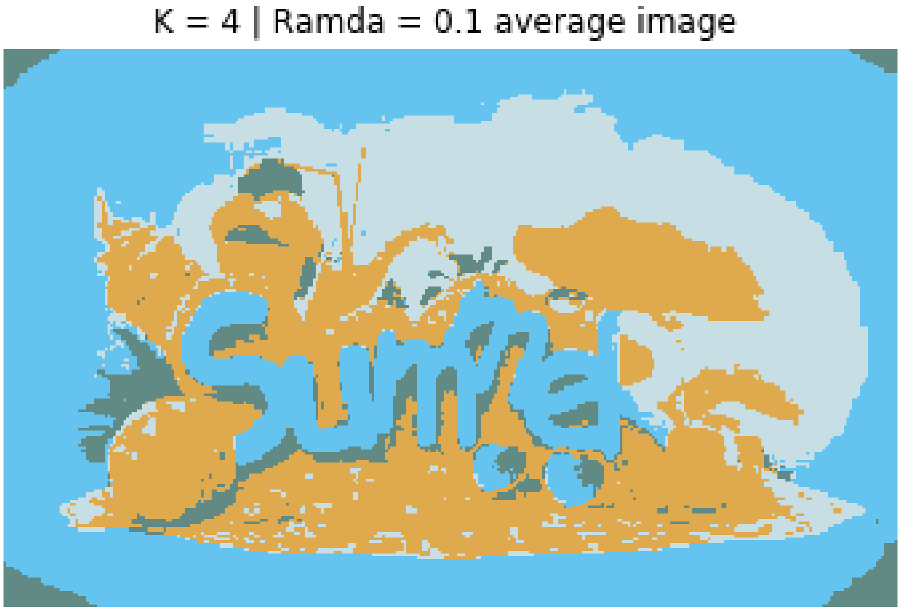
```
In [198]: # define variables
          iteration = 0
          k = 4
          ramda = 0.1
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

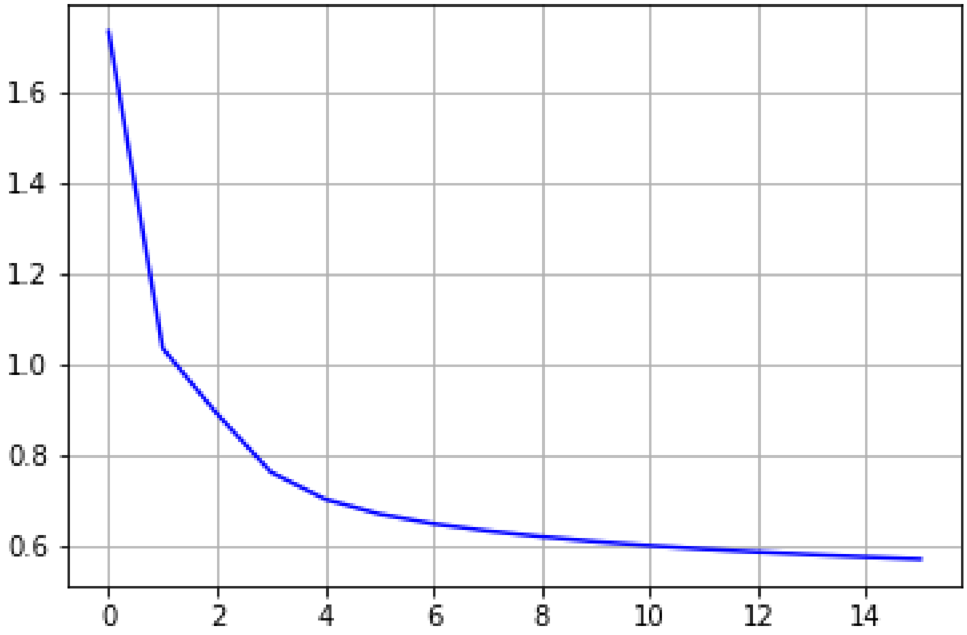
```
In [199]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                          , Label_Array, k, width, height ,ramda)
              Energy.append(energy)
              iteration += 1

              if Energy[-2] - Energy[-1] < threshold:
                  print('done')
                  break
```

```
done
```

```
In [200]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
plt.imshow(im_avg.astype(np.uint8))
plt.axis('off')
plt.show()

# energy graph
plt.figure(2)
plt.plot(Energy[1:],"b")
plt.grid(True)
plt.show()
```



.



0.3 Example [2] K = 4 , ramda = 10

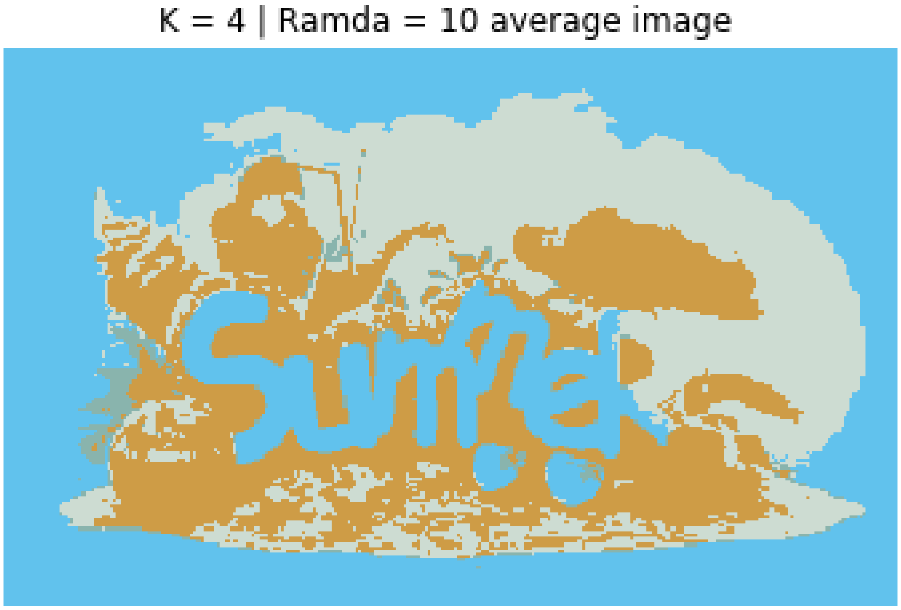
```
In [201]: # define variables
          iteration = 0
          k = 4
          ramda = 10
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

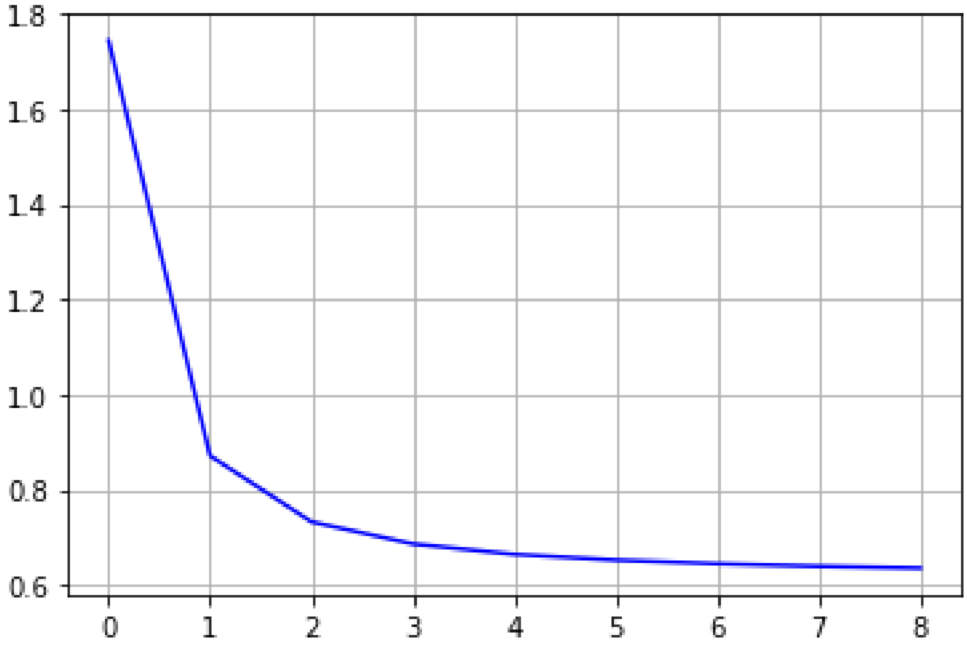
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

```
In [202]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                        , Label_Array, k, width, height ,ramda)
        Energy.append(energy)
        iteration += 1

        if Energy[-2] - Energy[-1] < threshold:
            print('done')
            break
```

```
done
```

```
In [203]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```





0.4 Example [3] K = 4 , ramda = 100

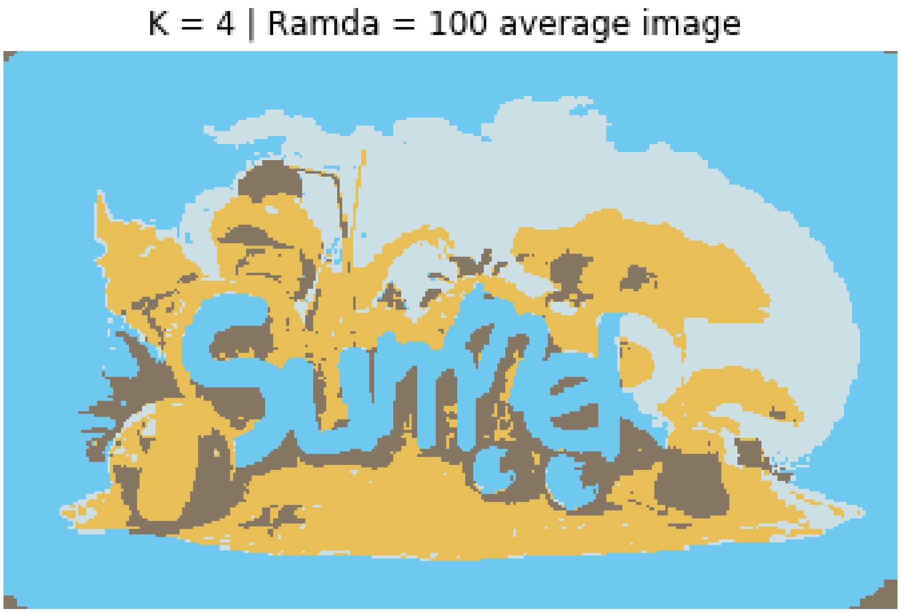
```
In [204]: # define variables
          iteration = 0
          k = 4
          ramda = 100
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

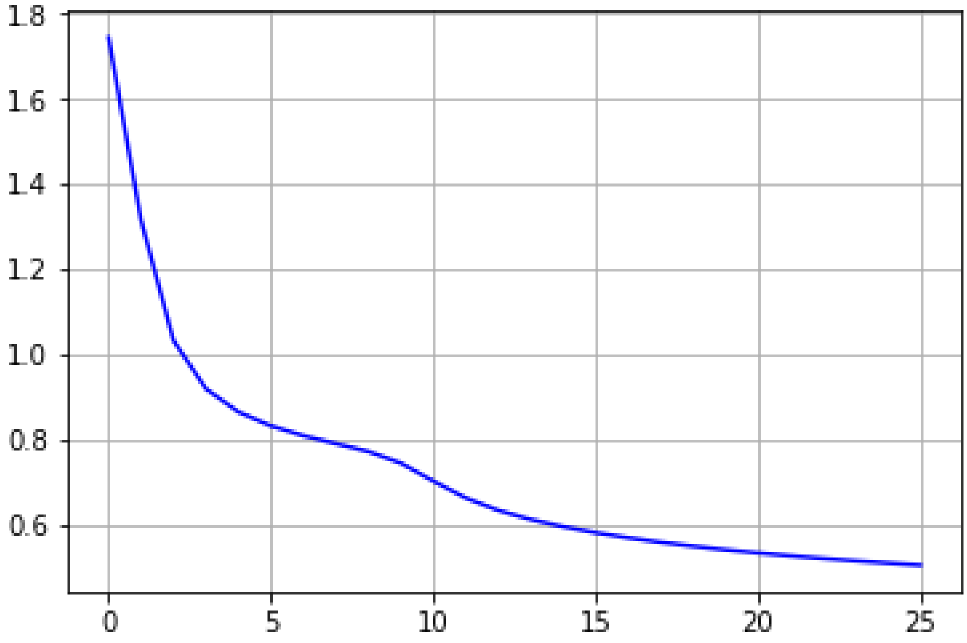
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

```
In [205]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                   , Label_Array, k, width, height ,ramda)
    Energy.append(energy)
    iteration += 1

    if Energy[-2] - Energy[-1] < threshold:
        print('done')
        break
```

```
done
```

```
In [206]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```





0.5 Example [4] K = 10 , ramda = 0.1

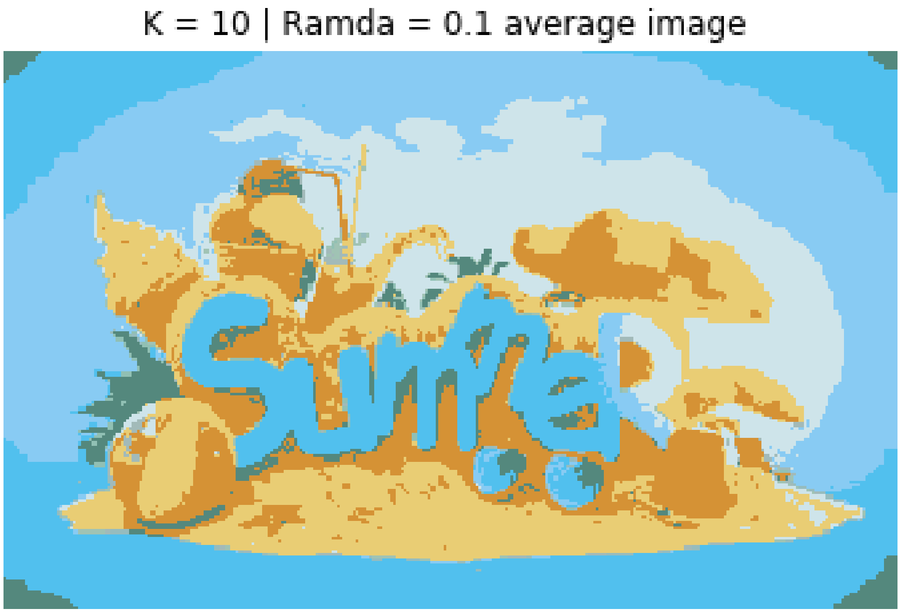
```
In [211]: # define variables
          iteration = 0
          k = 10
          ramda = 0.1
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

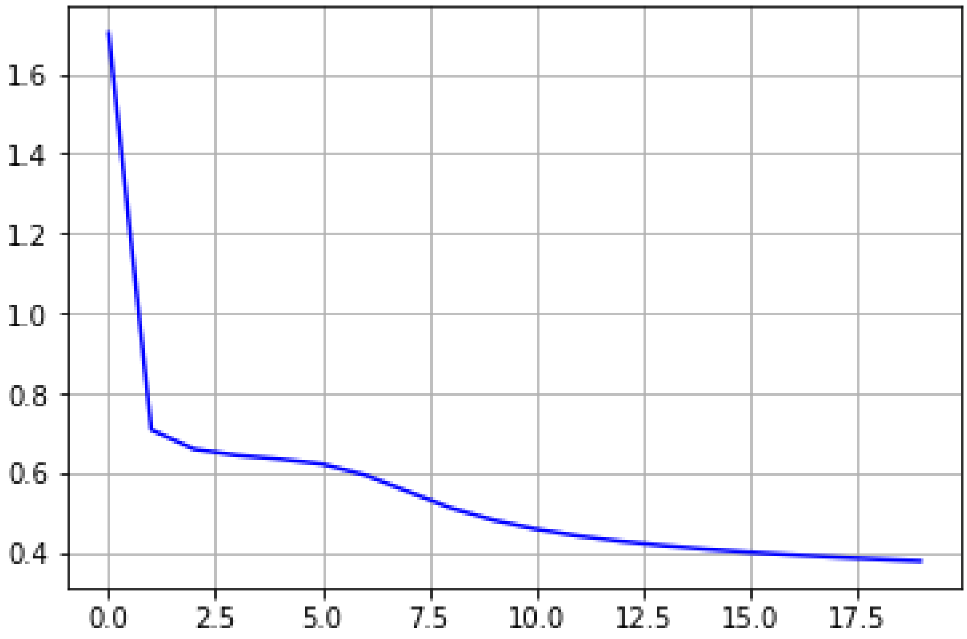
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

```
In [212]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                   , Label_Array, k, width, height ,ramda)
    Energy.append(energy)
    iteration += 1

    if Energy[-2] - Energy[-1] < threshold:
        print('done')
        break
```

```
done
```

```
In [213]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```





0.6 Example [5] K = 10 , ramda = 10

```
In [214]: # define variables
          iteration = 0
          k = 10
          ramda = 10
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

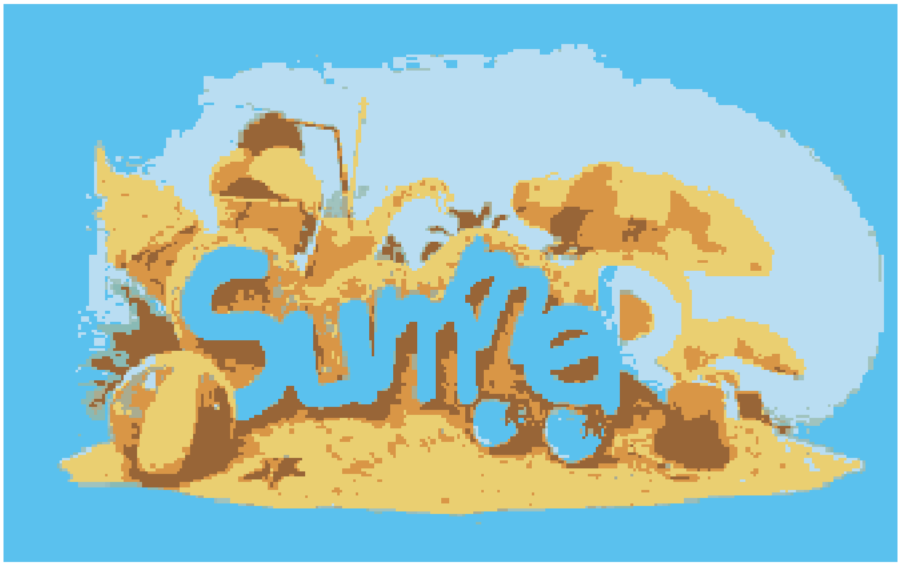
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

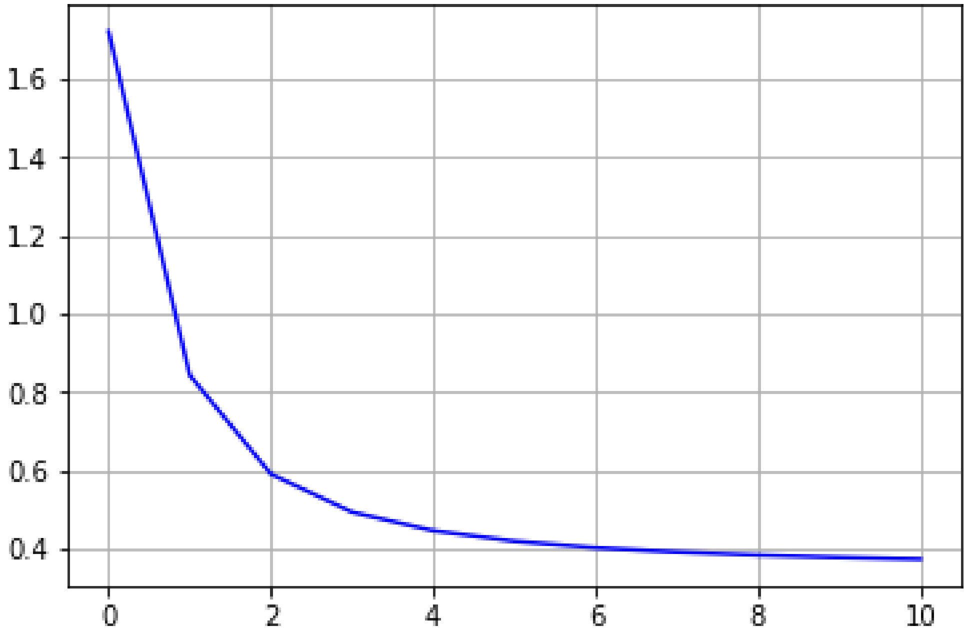
```
In [215]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                            , Label_Array, k, width, height ,ramda)
            Energy.append(energy)
            iteration += 1

            if Energy[-2] - Energy[-1] < threshold:
                print('done')
                break
```

```
done
```

```
In [216]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```

K = 10 | Ramda = 10 average image





0.7 Example [6] K = 10 , ramda = 100

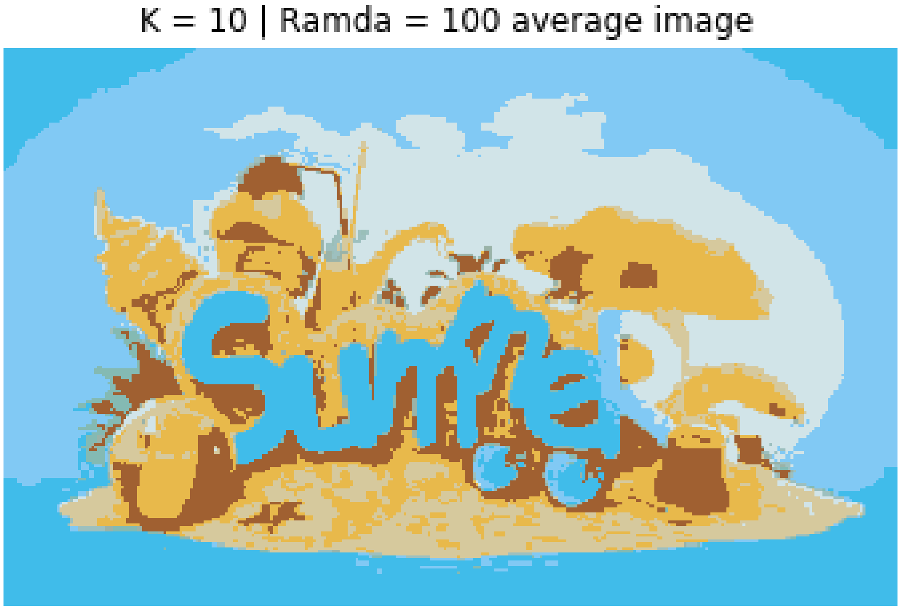
```
In [217]: # define variables
          iteration = 0
          k = 10
          ramda = 100
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

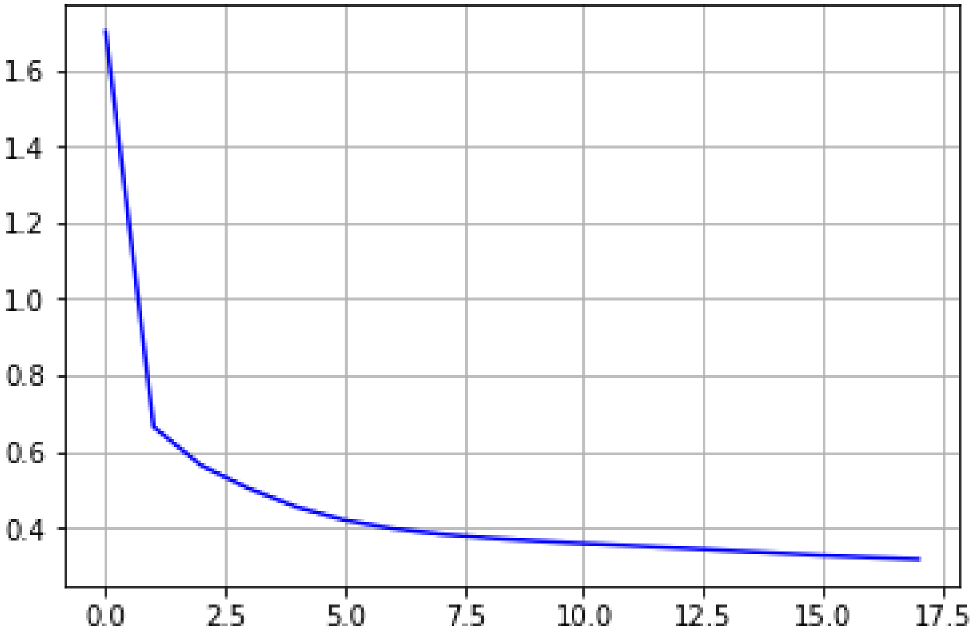
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

```
In [218]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                   , Label_Array, k, width, height ,ramda)
    Energy.append(energy)
    iteration += 1

    if Energy[-2] - Energy[-1] < threshold:
        print('done')
        break
```

```
done
```

```
In [219]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```





0.8 Example [7] K = 20 , ramda = 0.1

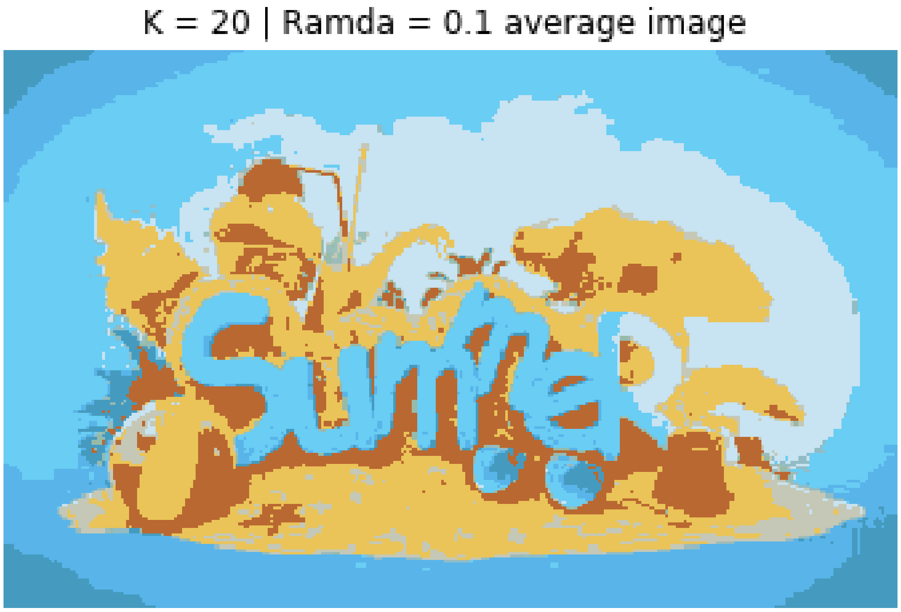
```
In [220]: # define variables
          iteration = 0
          k = 20
          ramda = 0.1
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

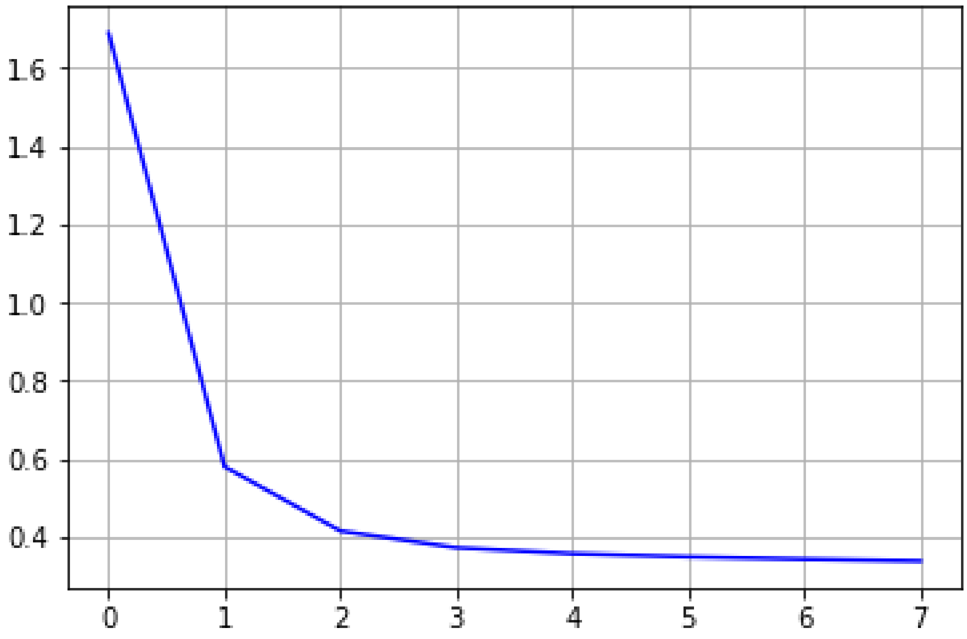
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

```
In [221]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                   k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                             , Label_Array, k, width, height ,ramda)
              Energy.append(energy)
              iteration += 1

              if Energy[-2] - Energy[-1] < threshold:
                  print('done')
                  break
```

```
done
```

```
In [222]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```





0.9 Example [8] K = 20 , ramda = 10

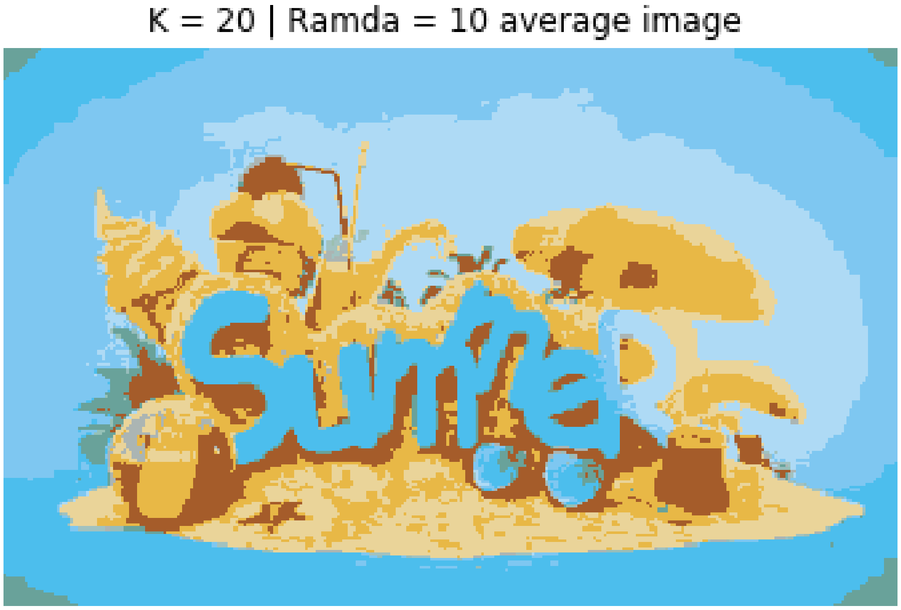
```
In [223]: # define variables
          iteration = 0
          k = 20
          ramda = 10
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

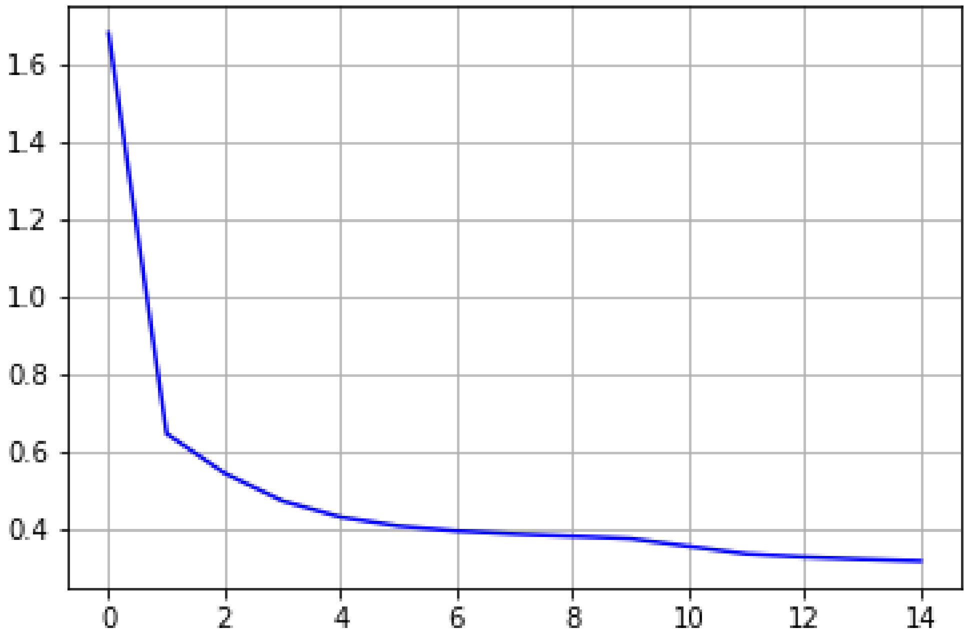
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

```
In [224]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                   , Label_Array, k, width, height ,ramda)
    Energy.append(energy)
    iteration += 1

    if Energy[-2] - Energy[-1] < threshold:
        print('done')
        break
```

```
done
```

```
In [225]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```





0.10 Example [9] K = 20 , ramda = 100

```
In [226]: # define variables
          iteration = 0
          k = 20
          ramda = 100
          Energy = []
          Energy.append(99999)

          Label_Array = initialize((height,width), k)
          centroid_list = np.zeros((k,5)) # r,g,b,x,y
          average = np.zeros((k,5)) # r,g,b,x,y
          count_list = np.zeros((k,1))

          X_axis_matrix, Y_axis_matrix = scailing()

          Function_G, mean_G, std_G = whitening(Function_X)
```

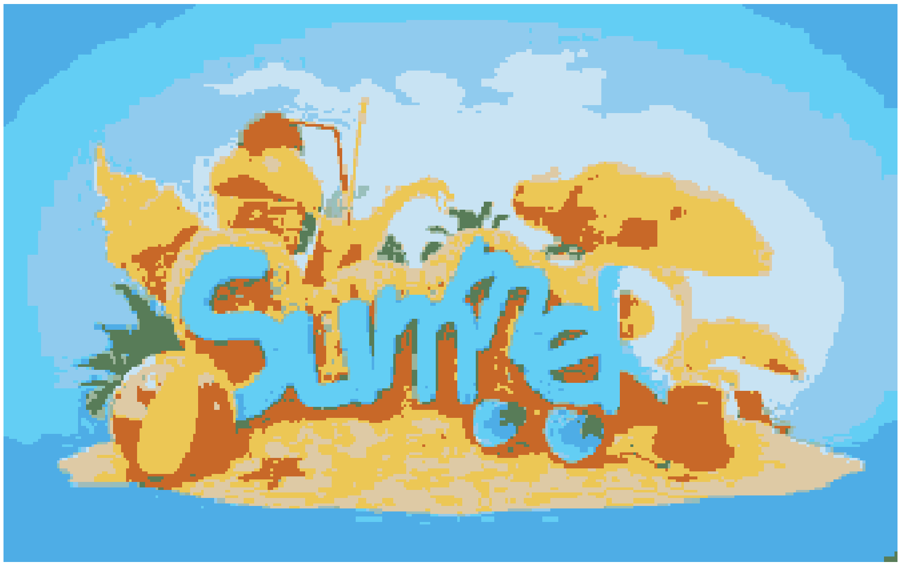
```
RGB mean is [153.36651548 190.83138378 186.338247  ]
RGB std is [54.39687384 33.80518268 60.07441439]
```

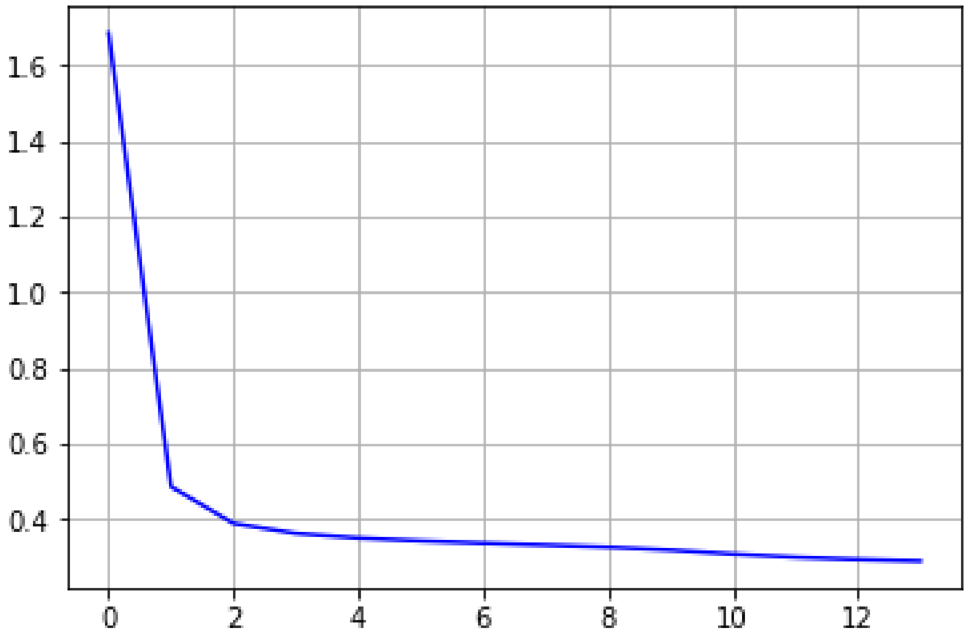
```
In [227]: while True:
              average = centroid(Function_G, X_axis_matrix, Y_axis_matrix, Label_Array,
                                 k, width, height)
              Label_Array , energy= labeling(Function_G,average, X_axis_matrix, Y_axis_matrix
                                            , Label_Array, k, width, height ,ramda)
                Energy.append(energy)
                iteration += 1

                if Energy[-2] - Energy[-1] < threshold:
                    print('done')
                    break
```

```
done
```

```
In [228]: im_avg = create_average_image(average, mean_G, std_G, Label_Array,k)
          plt.figure(1)
          plt.title('K = {} | Ramda = {} average image '.format(k, ramda))
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(2)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```

K = 20 | Ramda = 100 average image




1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
2
4
6
8
10
12