

```

4  import numpy
5  from scipy import misc
6  import matplotlib.pyplot as plt
7  import copy
8  from imageio import imread
9  import numpy as np
10
11  # Implement This Function
12
13  def get_hist (pic,M,N):
14      hist_out = np.zeros(65535)
15      for i in range(M):
16          for j in range(N):
17              itr = pic[i][j]
18              hist_out[itr] +=1
19      return hist_out
20
21  def get_cdf(a):
22      cdf = np.zeros(65535)
23      result = 0
24      for i in range(len(a)):
25          result += a[i]
26          cdf[i] = result
27      return cdf
28
29  def get_h(cdf,M,N):
30      h = np.zeros(65535)
31      cdf_min = 0
32      for i in range (len(cdf)):
33          if not(cdf[i] == 0):
34              cdf_min = 0
35              break
36      for j in range(len(cdf)):
37          result = int(((cdf[j]-cdf_min)/(M*N+1))*(65535-1))
38          h[j] = result
39      return h
40

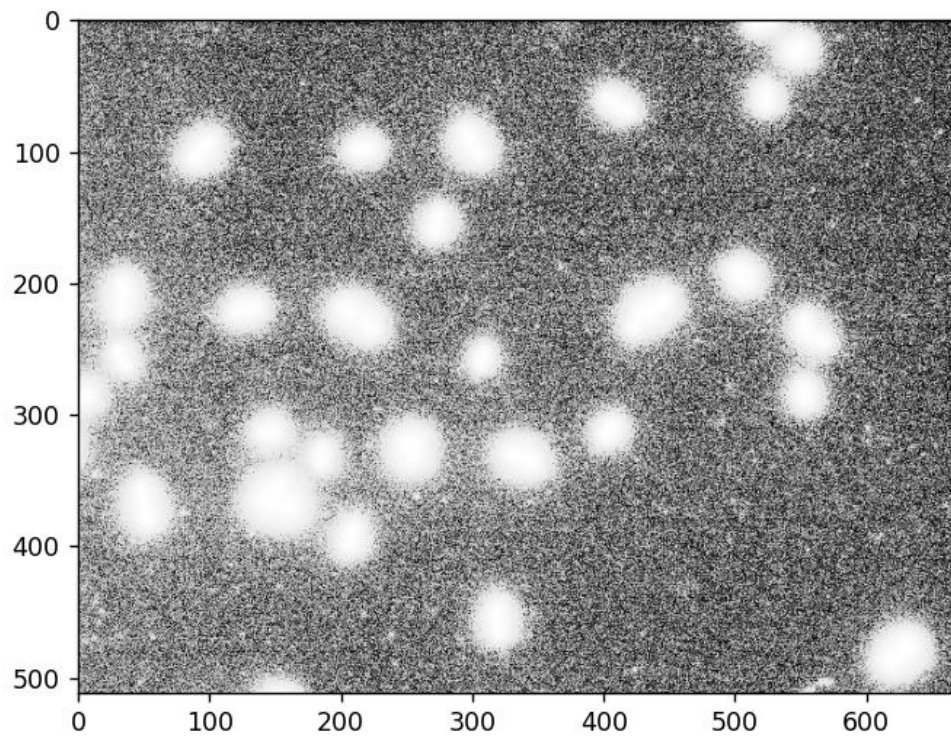
```

```

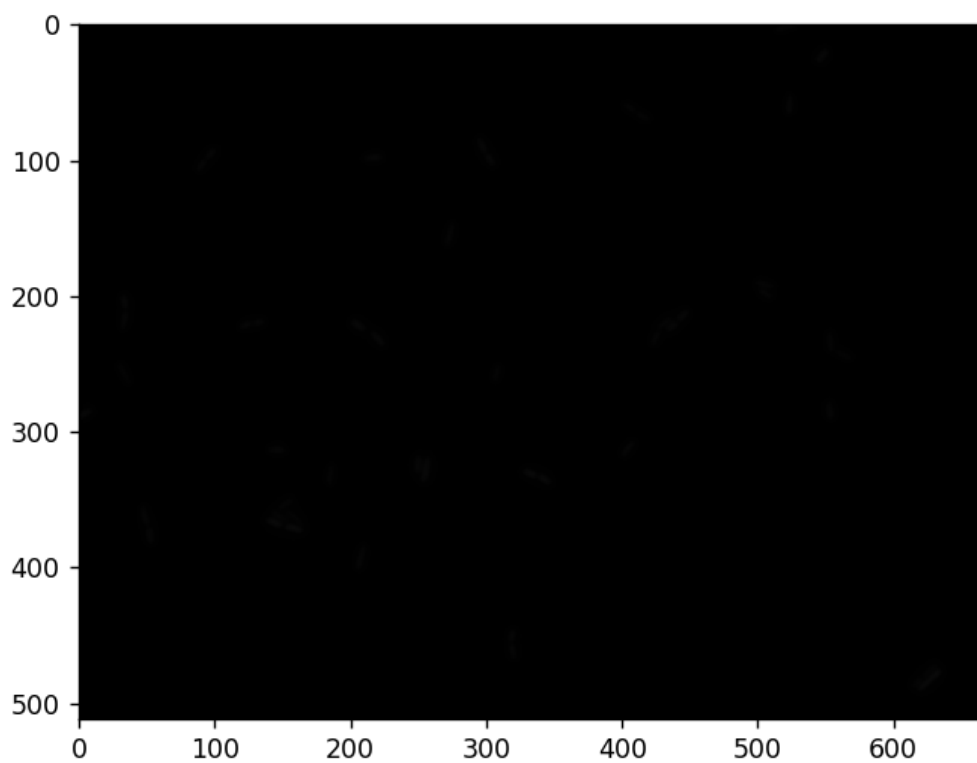
41
42 def histeq(pic):
43     # Follow the procedures of Histogram Equalizaion
44     # Modify the pixel value of pic directly
45     [M,N] = pic.shape
46     hist_cnt = get_hist(pic,M,N)
47     print ("hist generation complete")
48
49     cdf = get_cdf(hist_cnt)
50     print ("cdf generation complete")
51
52     h = get_h(cdf,M,N)
53     print ("h generation complete")
54
55
56     for k in range(M):
57         for l in range(N):
58             itr = pic[k][l]
59             pic[k][l] = h[itr]
60
61
62     return pic
63
64 # Histogram Equilization
65 eco_origin = imread('eco.tif')
66 eco_histeq = copy.deepcopy(eco_origin)
67 # Call to histeq to perform Histogram Equilization
68 eco_histeq = histeq(eco_histeq)
69 # Show the result in two windows
70 fig_eco_origin = plt.figure(1)
71 fig_eco_origin.suptitle('Original eco.tif', fontsize=14, fontweight='bold')
72 plt.imshow(eco_origin,cmap='gray',vmin = 0, vmax = 65535)
73 fig_eco_histeq = plt.figure(2)
74 fig_eco_histeq.suptitle('Histogram Equalized eco.tif', fontsize=14, fontweight='bold')
75 plt.imshow(eco_histeq,cmap='gray',vmin = 0, vmax = 65535)
76 plt.show()

```

Histogram Equalized eco.tif



Original eco.tif

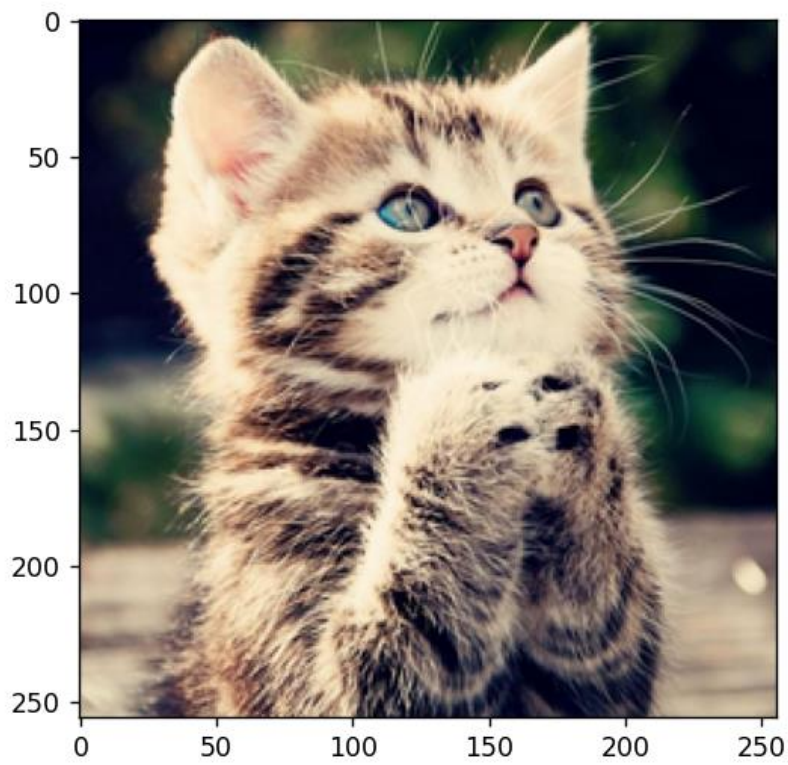


```

4  import numpy
5  from scipy import misc
6  import matplotlib.pyplot as plt
7  from imageio import imread
8  import numpy as np
9
10
11 # Function Definition Here
12
13 def conv2_1pix(pic_,kernel,i,j,M,N):
14     value_out = 0
15     [l,w] = kernel.shape
16     off_l = int((l-1)/2)
17     off_w = int((w-1)/2)
18     for x in range(l):
19         for y in range(w):
20             if (i-off_l+x<0) or (j-off_w+y<0) or (i-off_l+x>=M) or (j-off_w+y >= N):
21                 pic_data=0
22             else:
23                 #print (i-off_l+x,j-off_w+y)
24                 pic_data = pic_[i-off_l+x][j-off_w+y]
25                 pic_data = float(pic_data)
26                 value_out += pic_data * kernel[x][y]
27                 #print (np.uint8(value_out))
28     return int(value_out)
29
30 # Implement This funtion
31 def conv2(pic,kernel):
32     # Create a new pic with same size but float type
33     pic_conv = numpy.zeros(numpy.shape(pic), dtype=int)
34     [M,N,T] = pic.shape
35     print (M,N,T)
36     print(kernel)
37     # Perform 2-D Convolution with the given kernel
38     for layer in range(3):
39         pic_ = pic[:, :, layer]
40         for i in range (M):
41             for j in range (N):
42                 pic_conv[i][j][layer] = int(conv2_1pix(pic_,kernel,i,j,M,N))
43
44         print ("layer ",layer, "completed")
45         print (pic_)
46         print (pic_conv[:, :, layer])
47     return pic_conv
48

```

Original Kitten.png



Blurred Kitten.png

