AI6121- Computer Vision Assignment 1

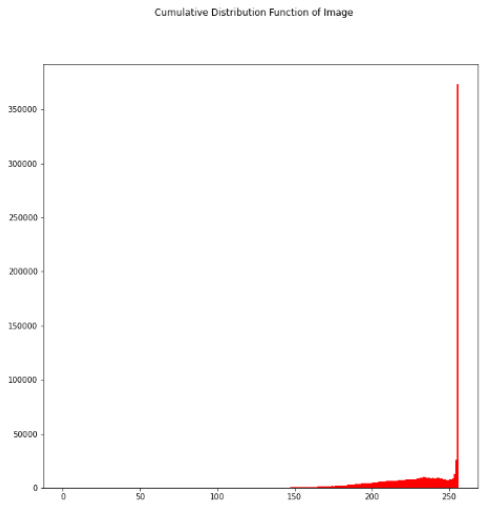
Brandon Chua Shaojie

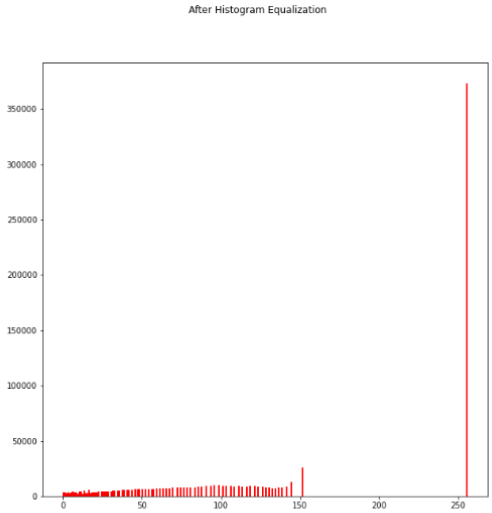
G1903442H

Part 1)

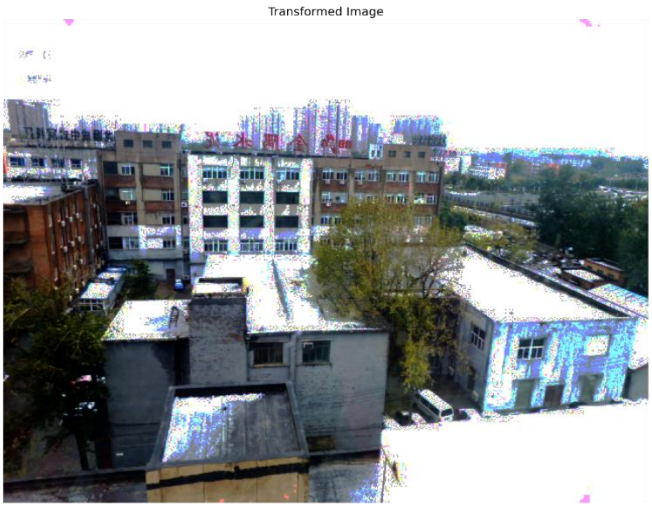
Images before and after applying HE algorithm

Histogram before and after equalization

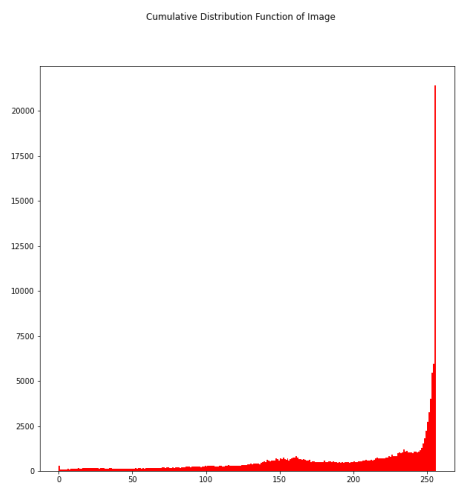


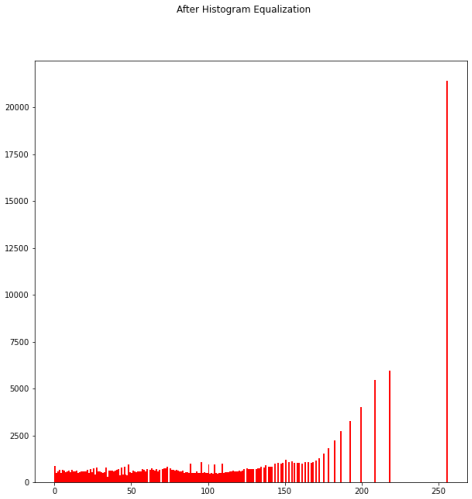


Images before and after equalization



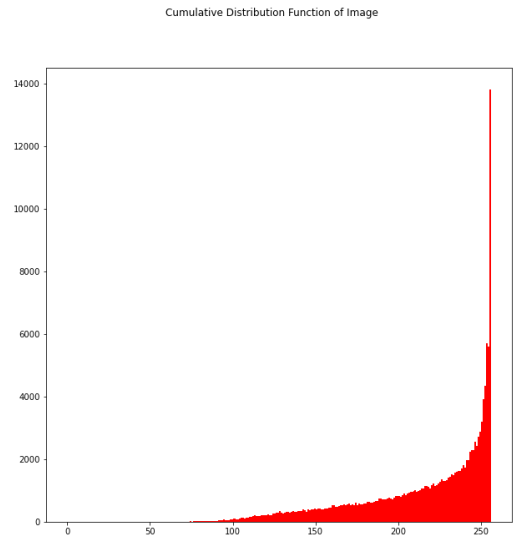
Histogram before and after equalization

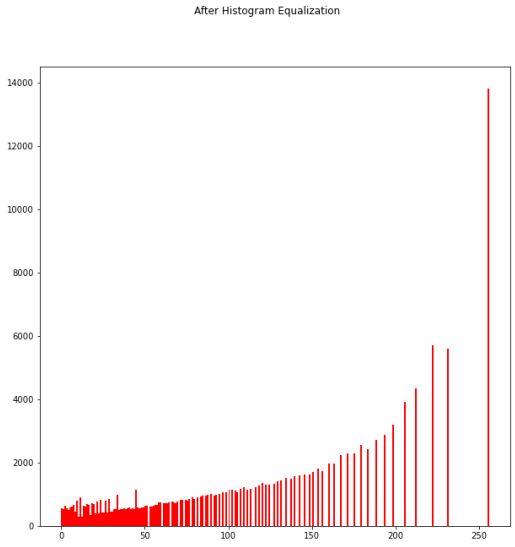




Images before and after equalization

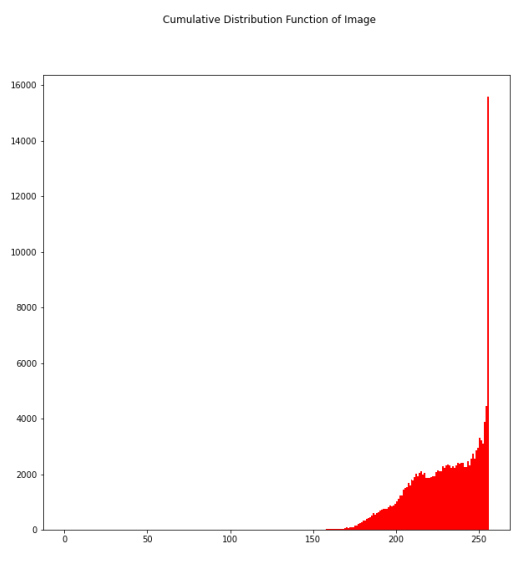
Histogram before and after equalization

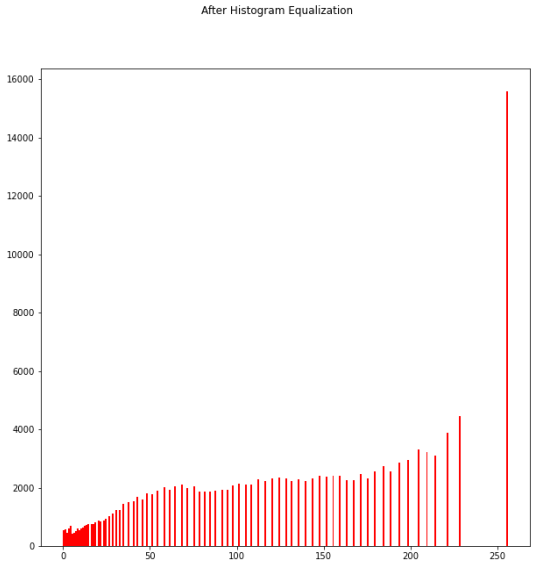




Images before and after equalization

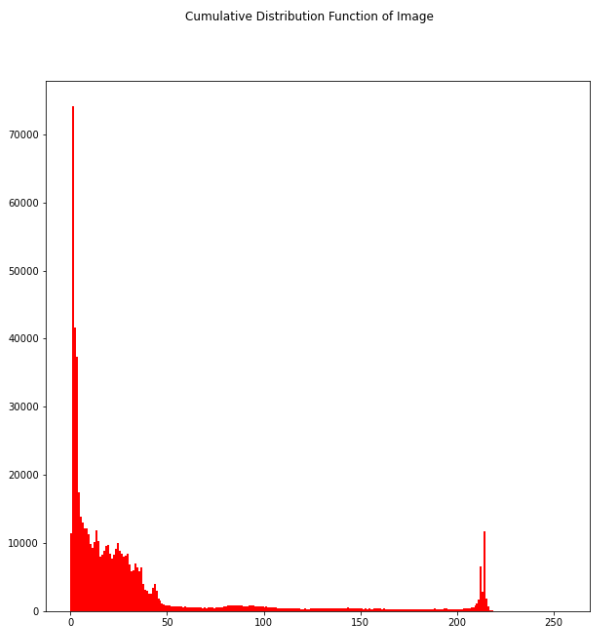
Histogram before and after equalization

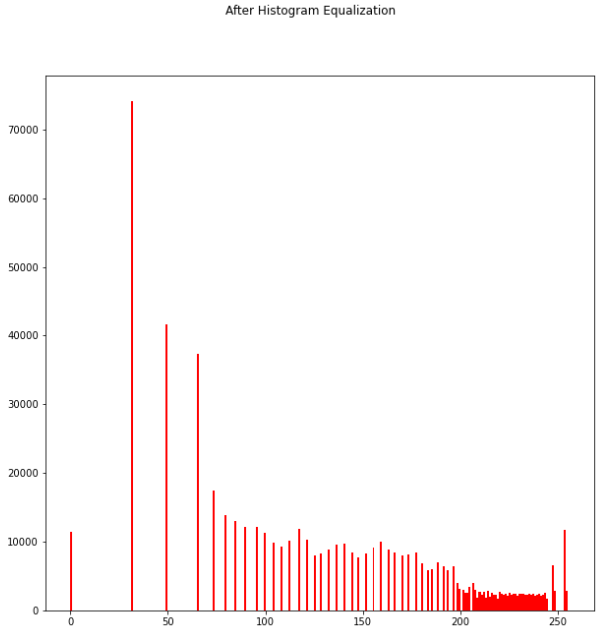




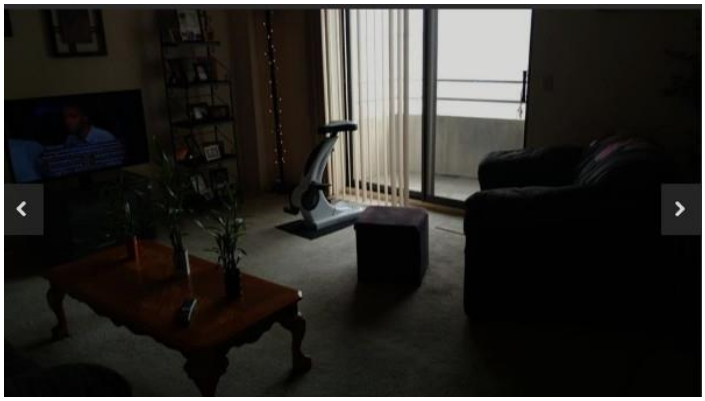
Images before and after equalization

Histogram before and after equalization

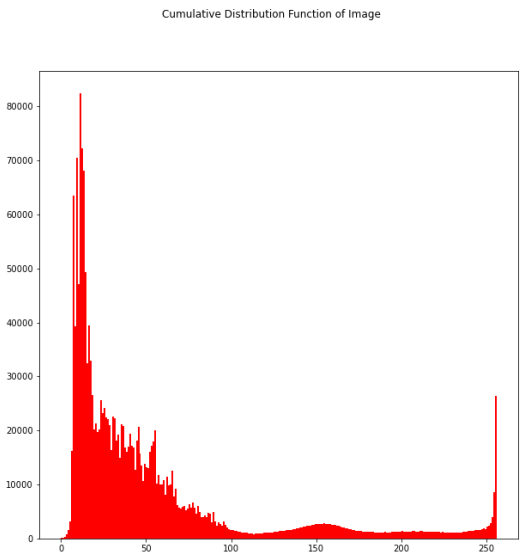


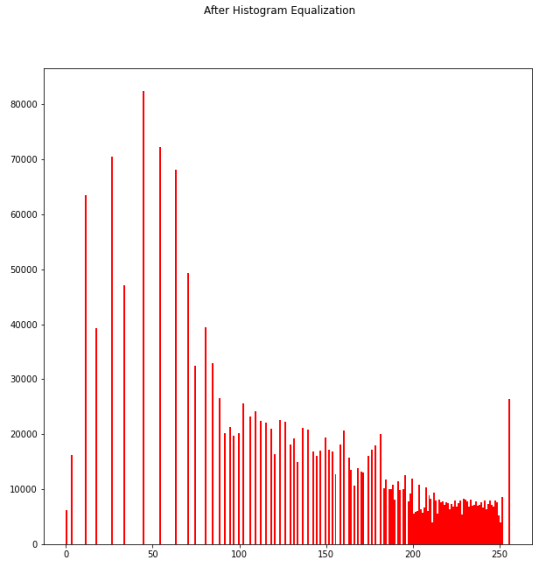


Images before and after equalization

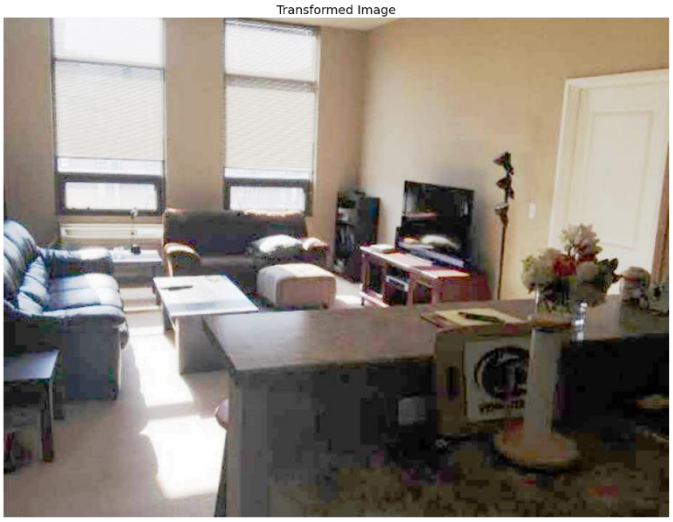


Histogram before and after equalization

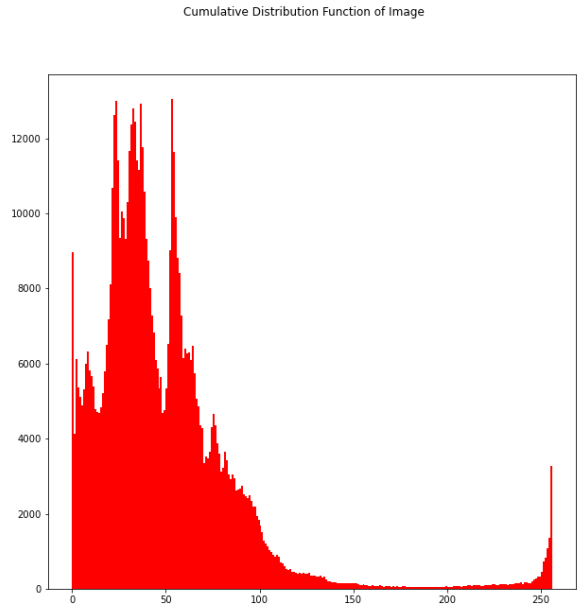


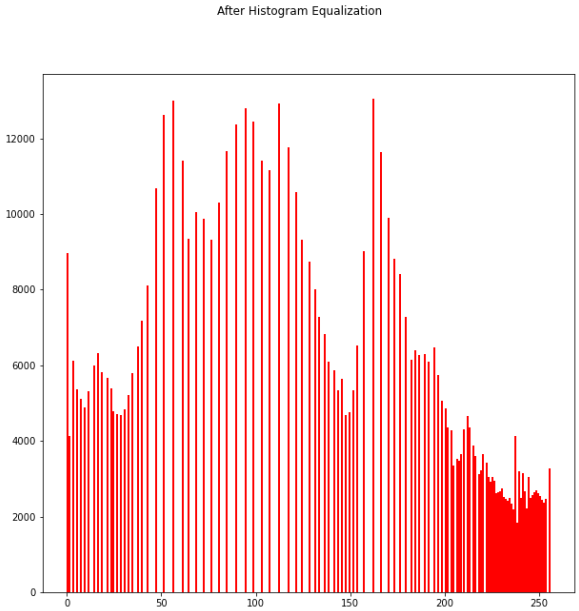


Images before and after equalization

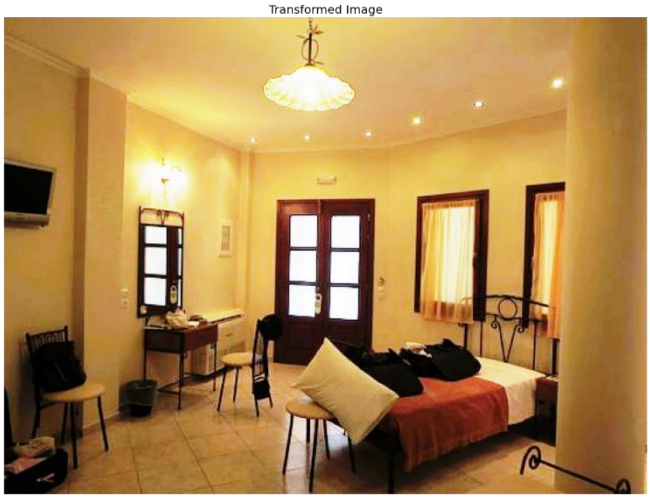


Histogram before and after equalization

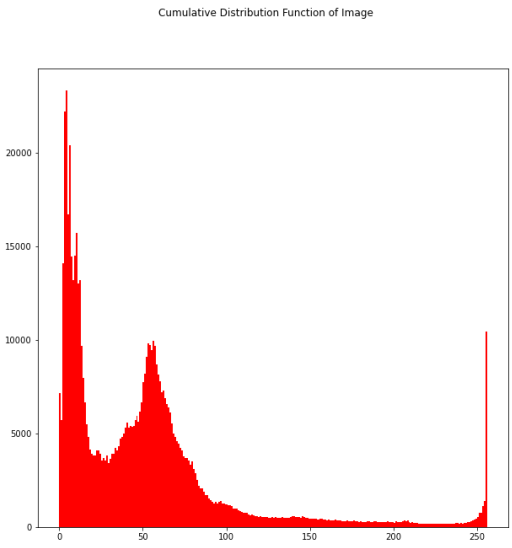


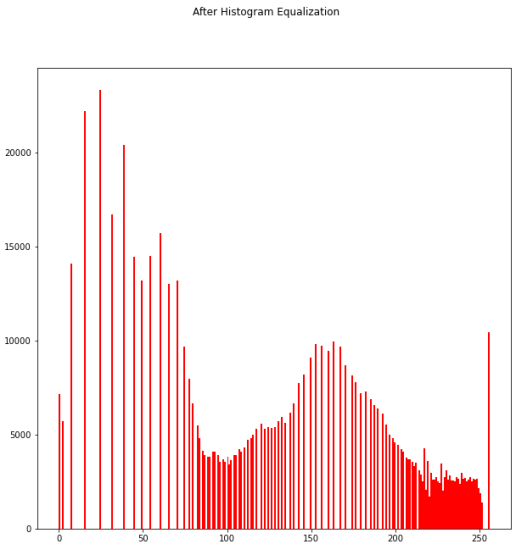


Images before and after equalization

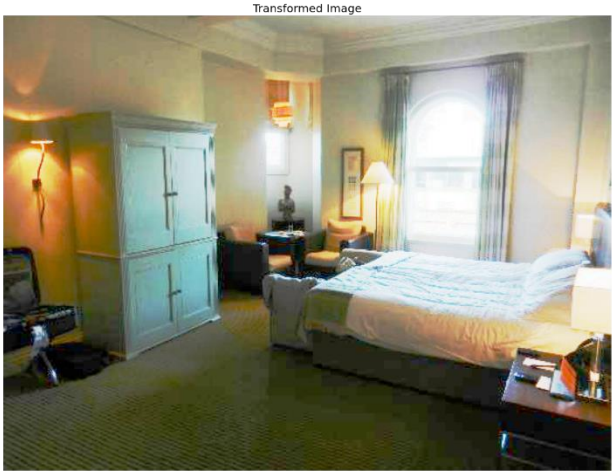


Histogram before and after equalization





Images before and after equalization



Code)

Listed below are the important parts of the code. The entire code will be included in the appendix.

Function to get cumulative distribution:

```
def cumSum(array):
    cumArray = zeros([len(array)])
    cumArray = cumArray.astype(int)
    for i in range(len(array)):
        for j in range(i+1):
            cumArray[i] += array[j]
    return cumArray
```

Histogram Equalization

```
histEqu = ((cdfMasked - cdfMasked.min())*255)/(cdfMasked.max() - cdfMasked.min())
```

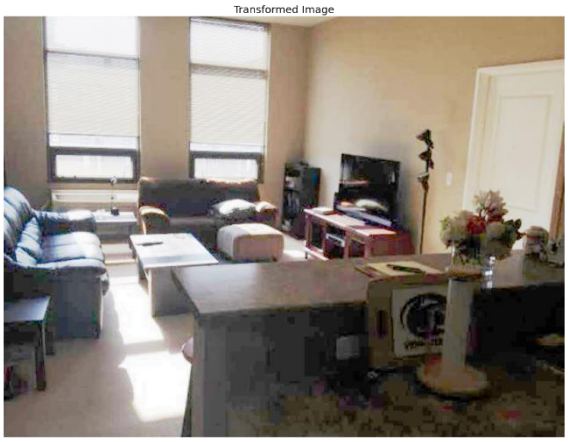
Part 2)

Advantages of histogram equalization

-Enhances contrast

-Not computationally expensive

-A simple and straightforward method



As seen in the above images, after histogram equalization, the improved contrast has allowed us to see the objects at the bottom right of the image more clearly than before.

Disadvantages of histogram equalization

-It is indiscriminate and may increase the contrast of background noise instead

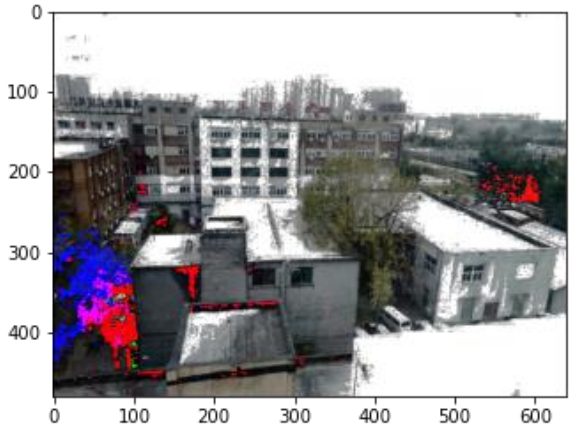
-Using it on color images is not ideal since false colors will be introduced. We should be performing histogram equalization on the intensity of the colors rather than applying on the color components themselves.

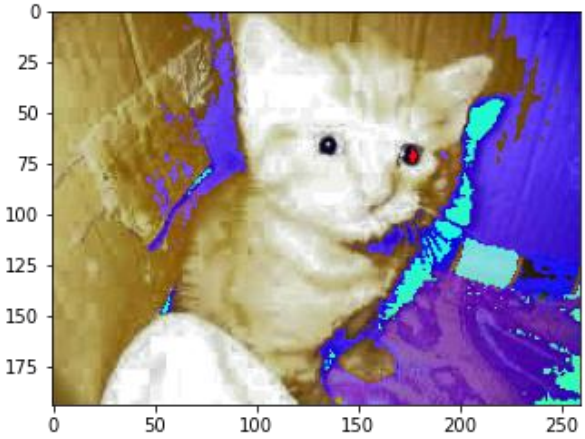
This can be seen in the above images where noise is seen more clearly after histogram equalization. This is undesired since we are introducing more noise into the image.

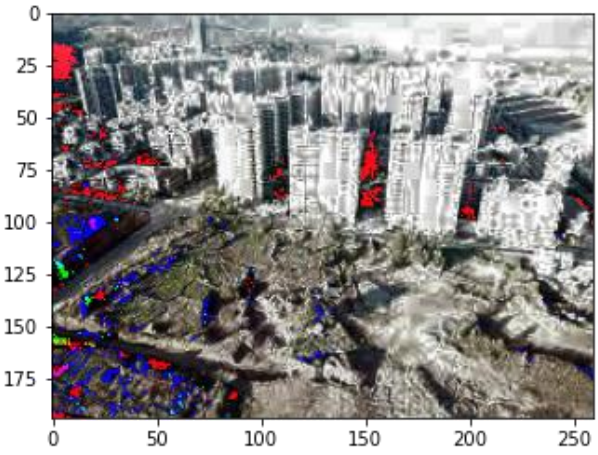
-Image gradient is also one of the disadvantages. This is caused when the image has low color depth.

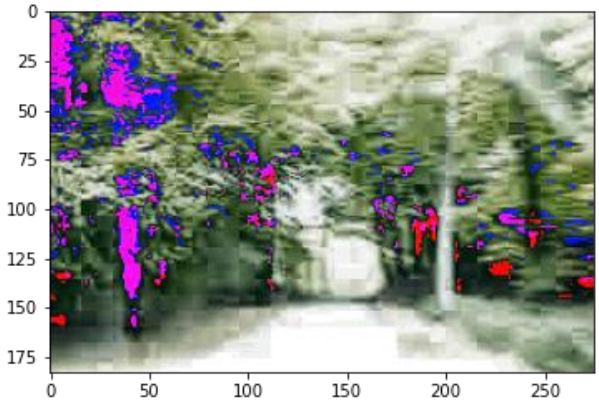
Part 3)

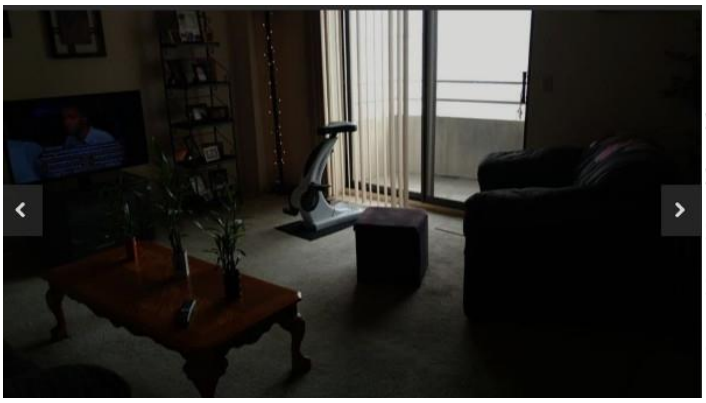
As mentioned before, applying Histogram Equalization (HE) directly to color images is not ideal. One possible improvement can be converting the image from RGB to YCbCr first, and then performing HE on the Y component of the image. Below are the results of my attempt at applying this idea on the 8 images.

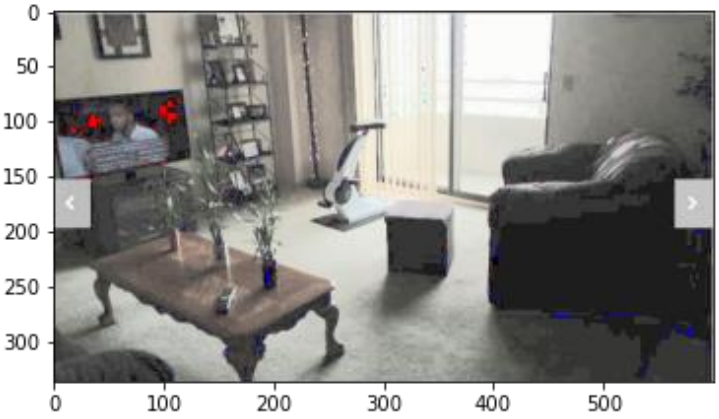


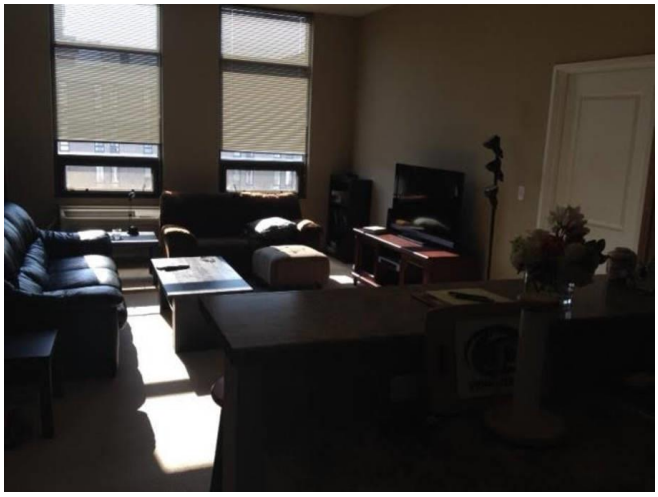


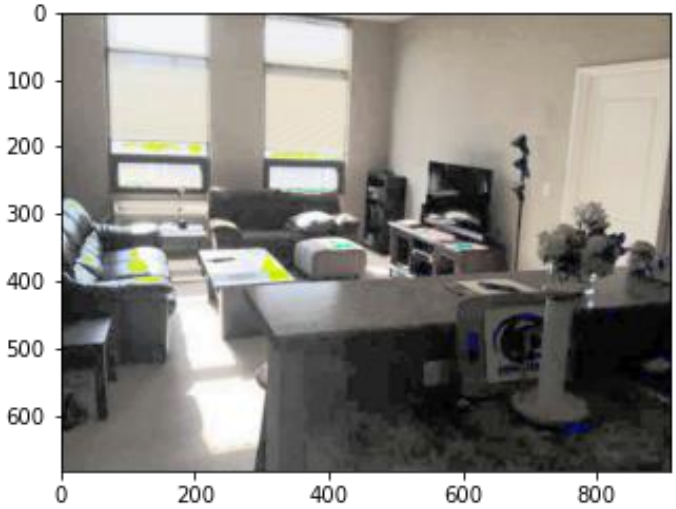


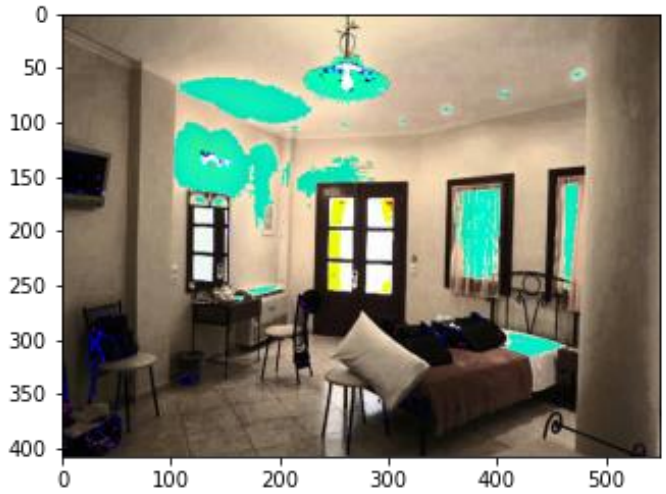

0
25
50
75
100
125
150
175
0
50
100
150
200
250

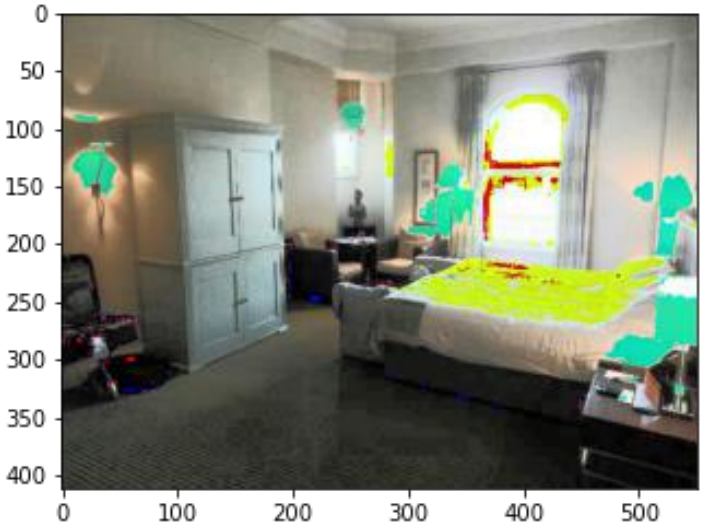



0
50
100
150
200
250
300
0
100
200
300
400
500




0
100
200
300
400
500
600
0
200
400
600
800





From the images, it can be seen that contrast has improved when compared to the original images. However, in all the images, false colors are introduced. Below shows the code used for converting RGB to YCbCr and vice versa, the entire code will be included in the appendix.

Function to Convert YCbCr to RGB

```
def ycbcrToRgb(im):
    xform = np.array([[1, 0, 1.402], [1, -0.34414, -.71414],
[1, 1.772, 0]])
    rgb = im.astype(np.float)
    rgb[:,:,[1,2]] -= 128
    return np.uint8(rgb.dot(xform.T))
```

Function to Convert image to YCbCr

```
ycbcr = image.convert('YCbCr')
```

References

1. Histogram-Equalization, Retrieved from https://github.com/rupav/Histogram-Equalization

2. Histogram equalization, Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Histogram_equalization

Appendix

```
#pip install numpy==1.15.0


import numpy as np

from PIL import Image

from matplotlib import pyplot as plt

from numpy import zeros


#loading image

image = Image.open('sample01.jpg')


#get width and height of image

imgWidth, imgHeight = image.size


#changing image to bytes so as to get pixel intesities

image_to_float = image.tobytes()

pixel_intensities = [image_to_float[i] for i in
range(len(image_to_float))]


# To plot cumulative frequency of pixel intensities

img = np.array(pixel_intensities).reshape((imgHeight,imgWidth,3))


hist, bins = np.histogram(img.flatten(),256,[0,256])

#cumulative distribution function
```

```
def cumSum(array):
    cumArray = zeros([len(array)])
    cumArray = cumArray.astype(int)
    for i in range(len(array)):
        for j in range(i+1):
            cumArray[i] += array[j]
  return cumArray

  cdf = cumSum(hist)

  #plot histogram of cdf
  fig,ax = plt.subplots(1,1,figsize=(10,10))
  fig.suptitle("Cumulative Distribution Function of Image")
  ax.hist(img.flatten(),256,[0,256],color='r')

  plt.savefig('Histogram Before.png')
  plt.show()

  #masking the zeroes in the array
  cdfMasked = np.ma.masked_equal(cdf,0)

  #Histogram Equalization
  histEqu = ((cdfMasked - cdfMasked.min())*255)/(cdfMasked.max() -
cdfMasked.min())
  histEqu = np.ma.filled(histEqu,0).astype('uint8')
  img2 = histEqu[img]

  #plot equalized histogram
  fig,ax = plt.subplots(1,1,figsize=(10,10))
  fig.suptitle("After Histogram Equalization")
  ax.hist(img2.flatten(),256,[0,256],color='r')

  plt.savefig('Histogram After.png')
```

```
  plt.show()

  #display equalized image
  font_dict = {'fontsize': 20,
               'fontweight' : 20,
               'verticalalignment':'baseline',
               }

  fig,axes = plt.subplots(1,1,figsize=(20,15),sharey = True)
  axes.imshow(img2.astype('uint8'),cmap = 'gray')
  axes.set_axis_off()
  axes.set_title("Transformed Image",fontdict = font_dict)

  plt.savefig('transformed_image2.png')
  plt.show()

  #Convert YCbCr to RGB
  def ycbcrToRgb(im):
      xform = np.array([[1, 0, 1.402], [1, -0.34414, -.71414], [1,
1.772, 0]])
      rgb = im.astype(np.float)
      rgb[:,:,[1,2]] -= 128
      return np.uint8(rgb.dot(xform.T))

  #Convert image to YCbCr
  ycbcr = image.convert('YCbCr')
  ycbcrToFloat =
np.ndarray((image.size[1],image.size[0],3),'u1',ycbcr.tobytes())

  #transforming the Y
  imgY = ycbcrToFloat[:,:,0]

  histY, bins = np.histogram(imgY.flatten(),256,[0,256])
```

```
  #cumulative distribution function
  cdfY = cumSum(histY)

  #plotting histogram
  fig,ax = plt.subplots(1,1,figsize=(10,10))
  fig.suptitle("YCbCr Image cdf")
  ax.hist(imgY.flatten(),256,[0,256],color='r')

  plt.savefig('Histogram of YCbCr image Before.png')
  plt.show()

  #Equalizing the Y Component
  cdfMaskedY = np.ma.masked_equal(cdfY,0)
  cdfMaskedY = ((cdfMaskedY -
cdfMaskedY.min())*255)/(cdfMaskedY.max() - cdfMaskedY.min())
  cdfScaledY = np.ma.filled(cdfMaskedY,0).astype('uint8')
  img3 = cdfScaledY[imgY]

  #plotting histogram
  fig,ax = plt.subplots(1,1,figsize=(10,10))
  fig.suptitle("After Histogram Equalization for Y")
  ax.hist(img3.flatten(),256,[0,256],color='r')

  plt.savefig('Histogram of YCbCr image After.png')
  plt.show()

  ycbcrToFloat.setflags(write = 1)

  ycbcrTransformed = ycbcrToFloat
  ycbcrTransformed[:,:,0] = img3

  #transformed YCbCr image
```

```
plt.imshow(ycbcrTransformed[:,:,:].astype('uint8'),cmap='gray')
plt.savefig("RGB-YCbCr-transformed.png")
plt.show()

#transforming image back to RGB
RGBTransformed = ycbcrToRgb(ycbcrTransformed)
plt.imshow(RGBTransformed.astype('uint8'),cmap='gray')
plt.savefig('RGB-YCbCr-RGB_transfomed.png')
plt.show()
```

In [1]:
```
#pip install numpy==1.15.0
```

In [ ]:
```
import numpy as np
from PIL import Image
from matplotlib import pyplot as plt
from numpy import zeros

#Loading image
image = Image.open('sample01.jpg')

#get width and height of image
imgWidth, imgHeight = image.size

#changing image to bytes so as to get pixel intesities
image_to_float = image.tobytes()
pixel_intensities = [image_to_float[i] for i in range(len(image_to_float))]
```

In [ ]:
```
# To plot cumulative frequency of pixel intensities
img = np.array(pixel_intensities).reshape((imgHeight,imgWidth,3))

hist, bins = np.histogram(img.flatten(),256,[0,256])
```

In [ ]:
```
#cumulative distribution function
def cumSum(array):
    cumArray = zeros([len(array)])
    cumArray = cumArray.astype(int)
    for i in range(len(array)):
        for j in range(i+1):
            cumArray[i] += array[j]
    return cumArray
```

In [ ]:
```
cdf = cumSum(hist)
```

In [ ]:
```
#plot histogram of cdf
fig,ax = plt.subplots(1,1,figsize=(10,10))
fig.suptitle("Cumulative Distribution Function of Image")
ax.hist(img.flatten(),256,[0,256],color='r')

plt.savefig('Histogram Before.png')
plt.show()
```

In [ ]:
```
#masking the zeroes in the array
cdfMasked = np.ma.masked_equal(cdf,0)
```

In [ ]:
```
#Histogram Equalization
histEqu = ((cdfMasked - cdfMasked.min())*255)/(cdfMasked.max() - cdfMasked.min())
histEqu = np.ma.filled(histEqu,0).astype('uint8')
img2 = histEqu[img]
```

In [ ]:
```
#plot equalized histogram
fig,ax = plt.subplots(1,1,figsize=(10,10))
fig.suptitle("After Histogram Equalization")
ax.hist(img2.flatten(),256,[0,256],color='r')

plt.savefig('Histogram After.png')
plt.show()
```

In [ ]:
```
#display equalized image
font_dict = {'fontsize': 20,
             'fontweight' : 20,
             'verticalalignment':'baseline',
             }

fig,axes = plt.subplots(1,1,figsize=(20,15),sharey = True)
axes.imshow(img2.astype('uint8'),cmap = 'gray')
axes.set_axis_off()
axes.set_title("Transformed Image",fontdict = font_dict)

plt.savefig('transformed_image2.png')
plt.show()
```

In [ ]:
```
#Convert YCbCr to RGB
def ycbcrToRgb(im):
    xform = np.array([[1, 0, 1.402], [1, -0.34414, -.71414], [1, 1.772, 0]])
    rgb = im.astype(np.float)
    rgb[:,:,[1,2]] -= 128
    return np.uint8(rgb.dot(xform.T))
```

In [ ]:
```
#Convert image to YCbCr
ycbcr = image.convert('YCbCr')
ycbcrToFloat = np.ndarray((image.size[1],image.size[0],3),'u1',ycbcr.tobytes())
```

In [ ]:
```
#transforming the Y
imgY = ycbcrToFloat[:,:,0]

histY, bins = np.histogram(imgY.flatten(),256,[0,256])
```

In [ ]:
```
#cumulative distribution function
cdfY = cumSum(histY)
```

In [ ]:
```
#plotting histogram
fig,ax = plt.subplots(1,1,figsize=(10,10))
fig.suptitle("YCbCr Image cdf")
ax.hist(imgY.flatten(),256,[0,256],color='r')

plt.savefig('Histogram of YCbCr image Before.png')
plt.show()
```

In [ ]:
```
#Equalizing the Y Component
cdfMaskedY = np.ma.masked_equal(cdfY,0)
cdfMaskedY = ((cdfMaskedY - cdfMaskedY.min())*255)/(cdfMaskedY.max() - cdfMaskedY.min())
cdfScaledY = np.ma.filled(cdfMaskedY,0).astype('uint8')
img3 = cdfScaledY[imgY]
```

In [ ]:
```
#plotting histogram
fig,ax = plt.subplots(1,1,figsize=(10,10))
fig.suptitle("After Histogram Equalization for Y")
ax.hist(img3.flatten(),256,[0,256],color='r')

plt.savefig('Histogram of YCbCr image After.png')
plt.show()
```

In [ ]:
```
ycbcrToFloat.setflags(write = 1)

ycbcrTransformed = ycbcrToFloat
ycbcrTransformed[:,:,0] = img3
```

In [ ]:
```
#transformed YCbCr image
plt.imshow(ycbcrTransformed[:,:,:].astype('uint8'),cmap='gray')
plt.savefig("RGB-YCbCr-transformed.png")
plt.show()
```

In [ ]:
```
#transforming image back to RGB
RGBTransformed = ycbcrToRgb(ycbcrTransformed)
plt.imshow(RGBTransformed.astype('uint8'),cmap='gray')
plt.savefig('RGB-YCbCr-RGB_transfomed.png')
plt.show()
```