Jason Tan

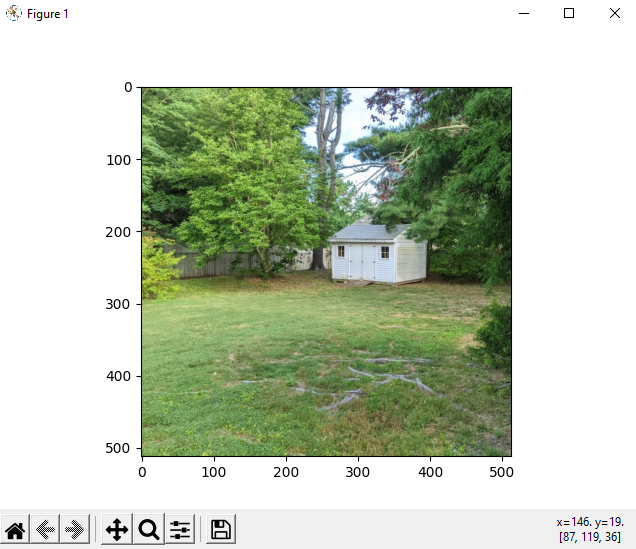
9/3/21

ESE 358

Professor Subbarao

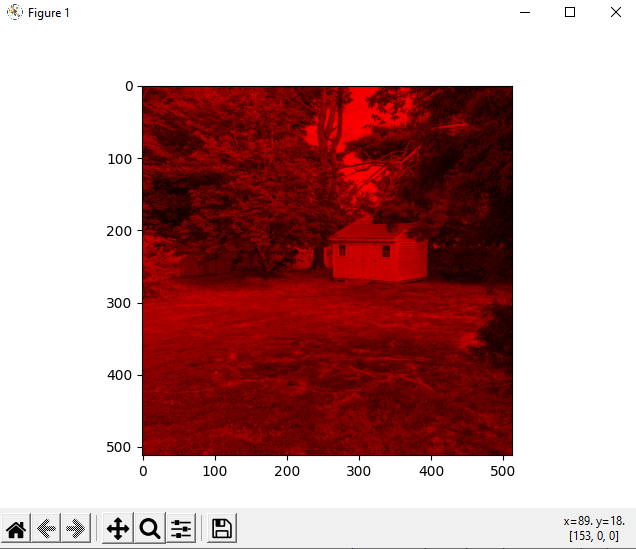
ESE 358 Project 1

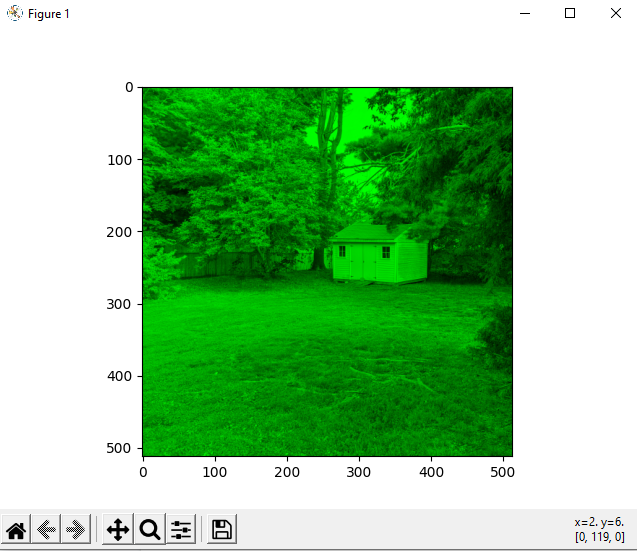
**Part 1:**

****

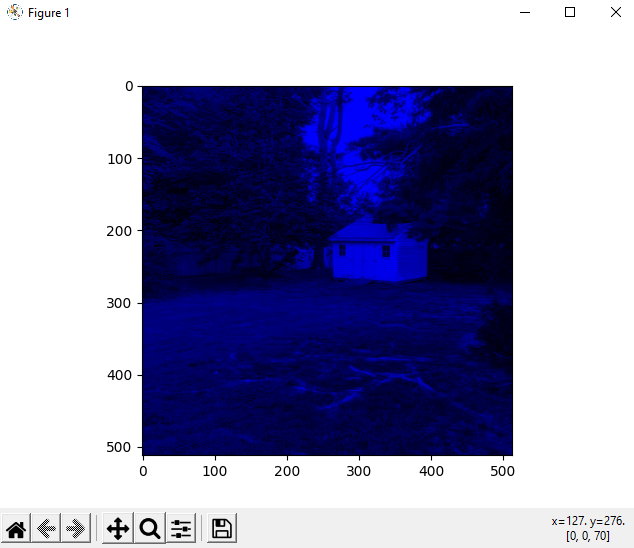
**Figure 1:** Displayed read input image for part 1

**Part 2:**

**Figure 2:** Displayed output image for channel R in part 2

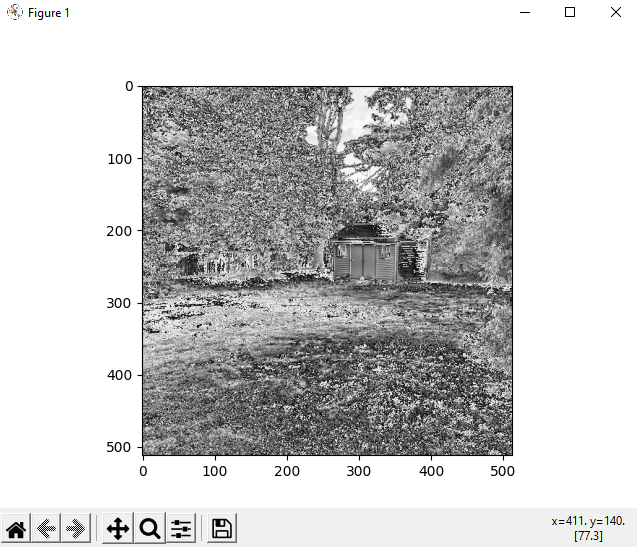


**Figure 3:** Displayed output image for channel G in part 2



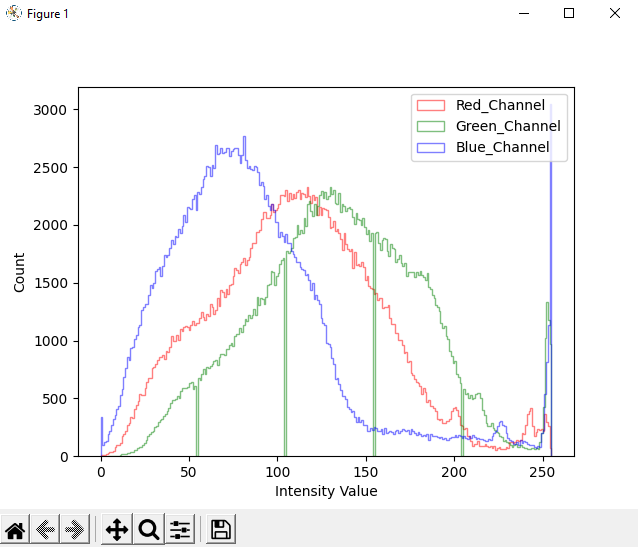
**Figure 4:** Displayed output image for channel B in part 2

**Part 3:**

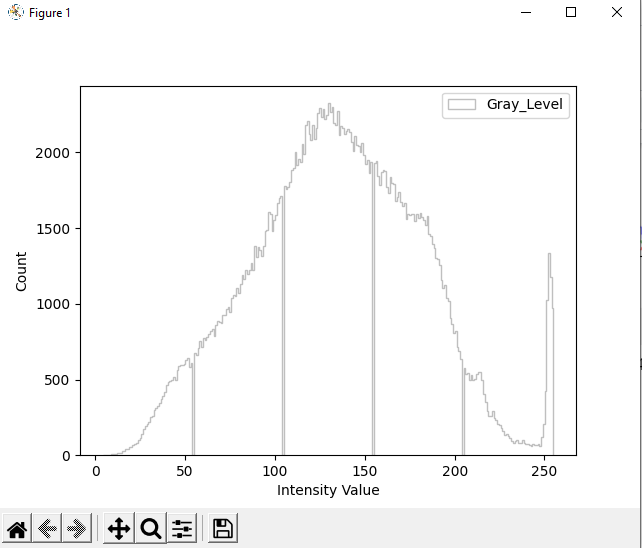
****

**Figure 5:** Displayed output image AG in part 3 after computation of average

**Part 4:**

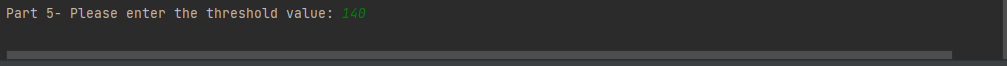
****

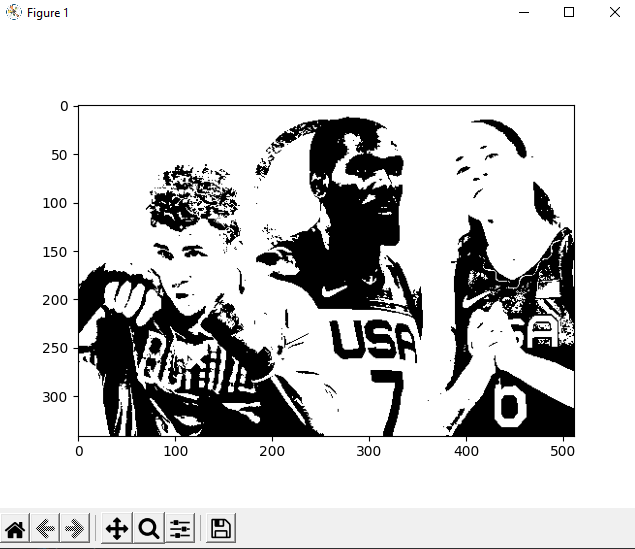
**Figure 6:** Displayed histogram for R, G, and B channels for part 4



**Figure 7:** Displayed histogram for gray level for part 4

**Part 5:**

****

****

**Figure 8:** Threshold image output for part 5 with my input threshold entered by user

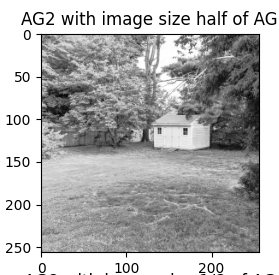
**Part 6:**

****

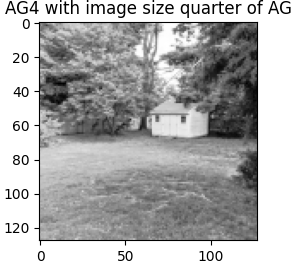
****

**Figure 8:** Edge Image output for part 6 with my input threshold entered by user

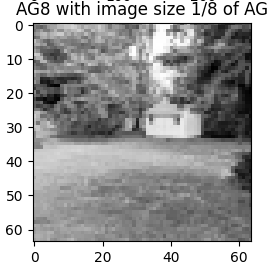
**Part 7:**

****

**Figure 8:** AG2 output for part 7 after half the size of AG (originally 512 x512 image read)



**Figure 9:** AG4 output for part 7 after quarter the size of AG (originally 512 x512 image read)



**Figure 10:** AG8 output for part 7 after quarter the size of AG (originally 512 x512 image read)

**My Source Code:**

# Understand the following code and modify it.

#

# -\*- coding: utf-8 -\*-

*"""*

*Created on Sept 6, 2021*

*Original template:*

*https://cs.brown.edu/courses/csci1430/2021\_Spring/resources/python\_tutorial/*

*Edited by*

*@author: Jason Tan, sbu, ece*

*"""*

import numpy as np

import skimage

from skimage import io

import matplotlib.pyplot as plt

#Part 1 - 4 of project 1

def rgb\_to\_gray(img):

grayImage = np.zeros(img.shape)

R = np.array(img[:, :, 0])

G = np.array(img[:, :, 1])

B = np.array(img[:, :, 2])

Avg = (R + G + B) / 3

grayImage = Avg

return grayImage

#To read the image of 'shed1-small.jpg': 512x512

rgbImage = io.imread('shed1-small.jpg')

plt.imshow(rgbImage)

plt.show()

(m,n,o) = rgbImage.shape[0: 3]

# Extract color channels.

redChannel = rgbImage[:,:,0] # Red channel

greenChannel = rgbImage[:,:,1] # Green channel

blueChannel = rgbImage[:,:,2] # Blue channel

# Create an all black channel.

allBlack = np.zeros((m, n), dtype=np.uint8)

# Create color versions of the individual color channels.

justRed = np.stack((redChannel, allBlack, allBlack), axis=2)

justGreen = np.stack((allBlack, greenChannel, allBlack),axis=2)

justBlue = np.stack((allBlack, allBlack, blueChannel),axis=2)

plt.imshow(justRed)

plt.show()

plt.imshow(justGreen)

plt.show()

plt.imshow(justBlue)

plt.show()

#Write images RC, GC, BC, in jpeg format

skimage.io.imsave('RCPart2.jpg', justRed)

skimage.io.imsave('GCPart2.jpg', justGreen)

skimage.io.imsave('BCPart2.jpg', justBlue)

grayImage = rgb\_to\_gray(rgbImage)

plt.imshow(grayImage, cmap=plt.cm.gray)

plt.show()

io.imsave('AG.jpg', grayImage)

#io.imsave('justRed1.jpg' , justRed)

\_ = plt.hist(rgbImage[:, :, 0].ravel(), bins = 256,histtype=u'step', color = 'red')

\_ = plt.hist(rgbImage[:, :, 1].ravel(), bins = 256,histtype=u'step', color = 'Green')

\_ = plt.hist(rgbImage[:, :, 2].ravel(), bins = 256,histtype=u'step', color = 'Blue')

\_ = plt.xlabel('Intensity Value')

\_ = plt.ylabel('Count')

\_ = plt.legend(['Red\_Channel', 'Green\_Channel', 'Blue\_Channel'])

plt.show()

\_ = plt.xlabel('Intensity Value')

\_ = plt.ylabel('Count')

\_ = plt.hist(rgbImage[:, :, 1].ravel(), bins = 256,histtype=u'step', color = 'gray')

\_ = plt.legend(['Gray\_Level'])

plt.show()

#Part 5: to threshold the image of the gray image if the value is greater than 170

import cv2

import matplotlib.pyplot as plt

grayImage1 = cv2.imread('justGrayImage.jpg')

#User has to enter in the threshold value

thresholdValue = input("Part 5- Please enter the threshold value: ")

thresholdValue = int(thresholdValue)

\_ ,thresholdImage = cv2.threshold(grayImage1, thresholdValue, 255, cv2.THRESH\_BINARY)

plt.imshow(thresholdImage)

plt.show()

#io.imsave('thresholdPart5.jpg', thresholdImage)

#Part 6 Compute the gradient to get the edge detection.

import cv2

import numpy as np

import matplotlib.pyplot as plt

part6Image = cv2.imread('olympics1.jpg')

thresholdValue1 = input("Part 6- Please enter the threshold value: ")

thresholdValue1 = int(thresholdValue1)

size = part6Image.shape

for row in range(part6Image.shape[0] - 1):

for column in range(part6Image.shape[1] - 1):

# averageX1 = (int(part6Image[row, column + 1, 0]) + int(part6Image[row, column + 1, 1]) + int(part6Image[row, column + 1, 2]))/3

# averageX = (int(part6Image[row, column, 0]) + int(part6Image[row, column, 1]) + int(part6Image[row, column, 2]))/3

# averageY1 = (int(part6Image[row + 1, column, 0]) + int(part6Image[row + 1, column, 1]) + int(part6Image[row + 1, column, 2]))/3

# averageY = (int(part6Image[row, column, 0]) + int(part6Image[row, column, 1]) + int(part6Image[row, column, 2]))/3

averageX1 = (part6Image[row, column + 1, 0] + part6Image[row, column + 1, 1] +

part6Image[row, column + 1, 2]) / 3

averageX = (part6Image[row, column, 0] + part6Image[row, column, 1] +

part6Image[row, column, 2]) / 3

averageY1 = (part6Image[row + 1, column, 0] + part6Image[row + 1, column, 1] +

part6Image[row + 1, column, 2]) / 3

averageY = (part6Image[row, column, 0] + part6Image[row, column, 1] +

part6Image[row, column, 2]) / 3

# gradientRowX = np.subtract(part6Image(row, column + 1), part6Image(row, column))

# gradientRowY = np.subtract(part6Image(row + 1, column), part6Image(row, column))

gradientRowX = averageX1 - averageX

gradientRowY = averageY1 - averageY

gradientMagnitude = np.sqrt((np.square(gradientRowX)) + (np.square(gradientRowY)))

# threshold the pixel

if gradientMagnitude > thresholdValue1:

part6Image[row, column] = 255

else:

part6Image[row, column] = 0

plt.imshow(part6Image)

plt.show()

#Part 7 of Project 1

import numpy as np

from matplotlib import pyplot as plt

import matplotlib.cm as cm

from skimage import io

from skimage import color

from skimage.util.shape import view\_as\_blocks

l = color.rgb2gray(io.imread('shed1-small.jpg'))

# -- size of blocks

block\_shape = (2, 2)

block\_shape1 = (4, 4)

block\_shape2 = (8, 8)

view = view\_as\_blocks(l, block\_shape)

view1 = view\_as\_blocks(l, block\_shape1)

view2 = view\_as\_blocks(l, block\_shape2)

# -- collapse the last two dimensions in one

flatten\_view = view.reshape(view.shape[0], view.shape[1], -1)

flatten\_view1 = view1.reshape(view1.shape[0], view1.shape[1], -1)

flatten\_view2 = view2.reshape(view2.shape[0], view2.shape[1], -1)

mean\_view = np.mean(flatten\_view, axis=2)

mean\_view1 = np.mean(flatten\_view1, axis=2)

mean\_view2 = np.mean(flatten\_view2, axis=2)

# -- display resampled images

fig, axes = plt.subplots(2, 2, figsize=(8, 8))

ax0, ax1, ax2, ax3 = axes.ravel()

ax0.set\_title("AG original image")

ax0.imshow(l, cmap=cm.Greys\_r)

ax1.set\_title("AG2 with image size half of AG")

ax1.imshow(mean\_view, cmap=cm.Greys\_r)

ax2.set\_title("AG4 with image size quarter of AG")

ax2.imshow(mean\_view1, cmap=cm.Greys\_r)

ax3.set\_title("AG8 with image size 1/8 of AG")

ax3.imshow(mean\_view2, cmap=cm.Greys\_r)

plt.show()