Assignment 1
Tudorache Vlad Adrian 442C

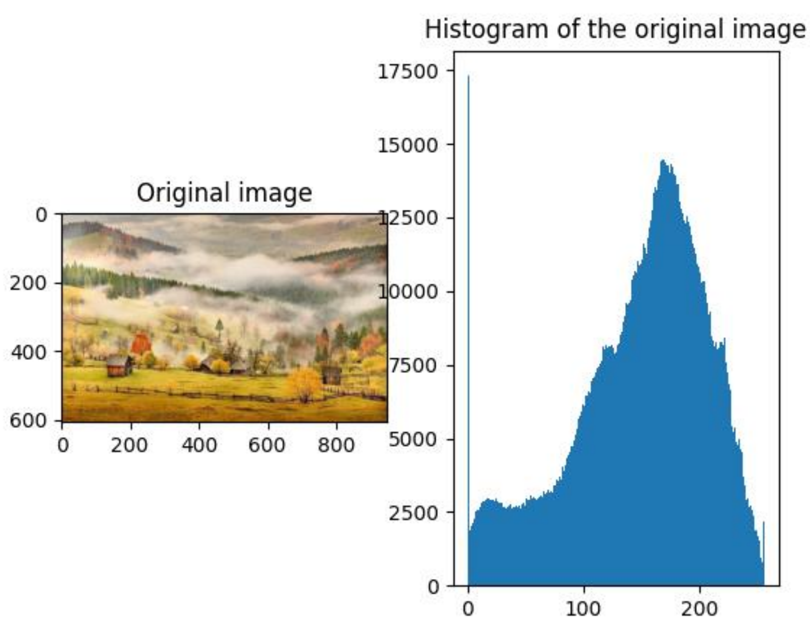
Assignment 1

Subsection 1:

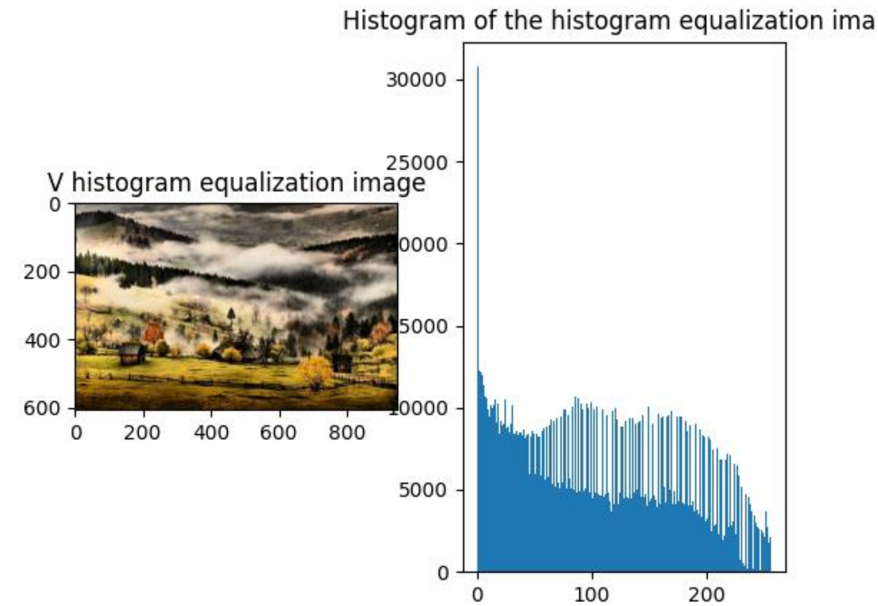
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# SUBSECTION 1 ------------------------------------------------------------
---------------------

img =
cv2.imread(r"C:\Users\Vladuts\Desktop\IPIVA\TEMA1\Background_images\landscape
7.jpg", 1)
imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# display the original picture
plt.imshow(img[:, :, ::-1])
plt.show()

plt.subplot(1, 2, 1); plt.imshow(img[:, :, ::-1]); plt.title("Original
image")
plt.subplot(1, 2, 2); plt.hist(img.ravel(), 256, [0, 256]);
plt.title("Histogram of the original image")

# histogram equalization:
plt.show()
H, S, V = cv2.split(imgHSV)

# to create the "luminance" effect we need to use the histogram equalization
on the V channel
#the V channel represents the brightness of each pixel

vOut = cv2.equalizeHist(V)
finalImg = cv2.merge((H, S, vOut))
imgOutHE = cv2.cvtColor(finalImg, cv2.COLOR_HSV2BGR)

#display the HE image

plt.imshow(imgOutHE[:, :, ::-1])
plt.show()

#display the HE histogram

plt.subplot(1, 2, 1); plt.imshow(imgOutHE[:, :, ::-1]); plt.title("V
histogram equalization image")
plt.subplot(1, 2, 2); plt.hist(imgOutHE.ravel(), 256, [0, 256]);
plt.title("Histogram of the histogram equalization image")
```

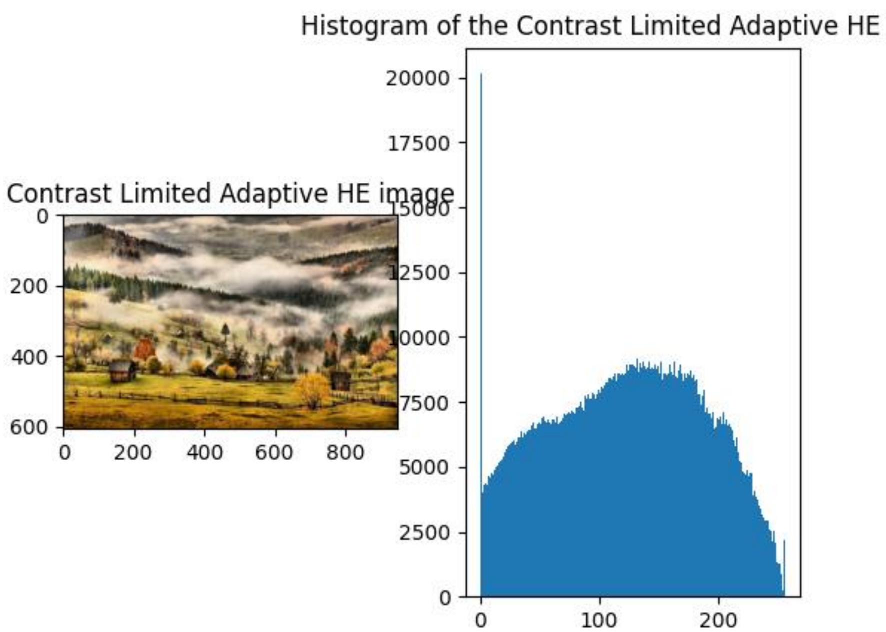
Assignment 1
Tudorache Vlad Adrian 442C

```
plt.show()

#Contrast Limited Adaptive HE:

imgHSV_copy = np.copy(imgHSV)
clahe = cv2.createCLAHE(clipLimit=2, tileGridSize=(8, 8))
imgHSV_copy[:, :, 2] = clahe.apply(imgHSV_copy[:, :, 2])
img_CLAHE = cv2.cvtColor(imgHSV_copy, cv2.COLOR_HSV2BGR)

# display the Contrast Limited Adaptive HE image:

plt.imshow(img_CLAHE[:, :, ::-1])
plt.show()

# display the Contrast Limited Adaptive HE image histogram:

plt.subplot(1, 2, 1); plt.imshow(img_CLAHE[:, :, ::-1]); plt.title("Contrast
Limited Adaptive HE image")
plt.subplot(1, 2, 2); plt.hist(img_CLAHE.ravel(), 256, [0, 256]);
plt.title("Histogram of the Contrast Limited Adaptive HE image")
plt.show()
cv2.imwrite('back_eq.jpg', img_CLAHE)

#As we can see the CLAHE method gives as a softer contrast effect, the image
is more clear.
#Also the histogram of the CLAHE image is more uniform then the orginal and
HE image.
```



Assignment 1
Tudorache Vlad Adrian 442C





Assignment 1
Tudorache Vlad Adrian 442C

Figure 1 - back_eq

Explanation: After analyzing the histograms, we can see that the CLAHE equalized image gives us the most uniform distribution. We want this so that the "luminance" effect is pleasant and each pixel is brightened in the necessary way.

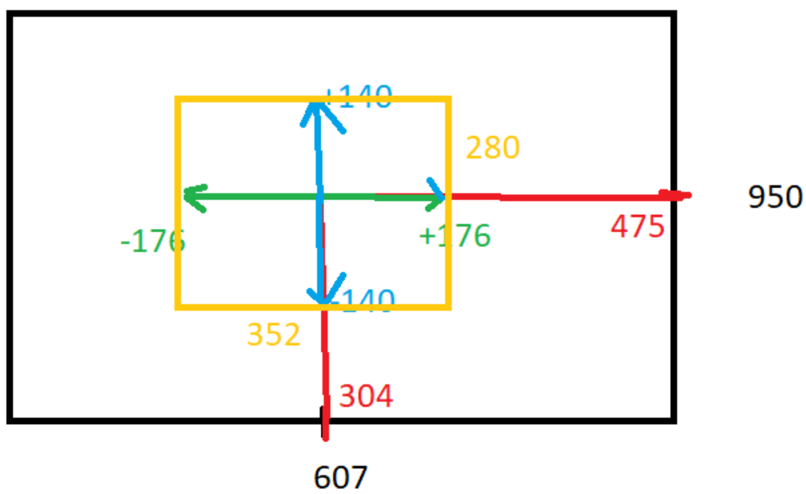
For the histogram equalization equalized image we applied the equalization on the V channel of the image because the V channel represents the brightness of each pixel. The HE yields a more brutal contras effect and the resulted histogram is not so uniform. In this manner we are losing details in the darker areas, like in the distant trees.

Subsection 2:

```
# SUBSECTION 2 --------------------------------------------------------------
---------------------

back_eq = np.copy(img_CLAHE)
noisyImg =
cv2.imread(r"C:\Users\Vladuts\Desktop\IPIVA\TEMA1\working_images_2021\4_noisy
.jpg", 1)
print(noisyImg.shape) #352, 280
print(back_eq.shape) #607, 950

#i have calculated the coordinates like this:
```

Assignment 1
Tudorache Vlad Adrian 442C

```
#(607/2) - (352/2):(607/2) + (352/2), (950/2) - (280/2):(950/2) + (280/2)]

img_cropped = back_eq[128:480, 335:615]
print(img_cropped.shape)
plt.imshow(img_cropped[:, :, ::-1])
plt.show()
cv2.imwrite('back_eq_crop.jpg', img_cropped)
```



Assignment 1
Tudorache Vlad Adrian 442C

Figure 2 - back_eq_crop

Subsection 3:

```
# SUBSECTION 3 ---------------------------------------------------------------
---------------------

plt.imshow(noisyImg[:, :, ::-1])
plt.show()

#The noise is the Salt and Pepper type
#The best type of filtering is using the median filter

kernelSize = 5 # from multiple tries this is kernel gives the best results
img_median = cv2.medianBlur(noisyImg, kernelSize )
plt.imshow(img_median[:, :, ::-1])
plt.show()
cv2.imwrite('im_4_filt.jpg', img_median)
```

Explanation: After analyzing the noisy image, I determined that noise is the salt-and-pepper type. As discussed in the second laboratory, the best way to tackle this noise is by using a median blur. After multiple tries, the best results came from a kernel size equal to 5.

Assignment 1
Tudorache Vlad Adrian 442C

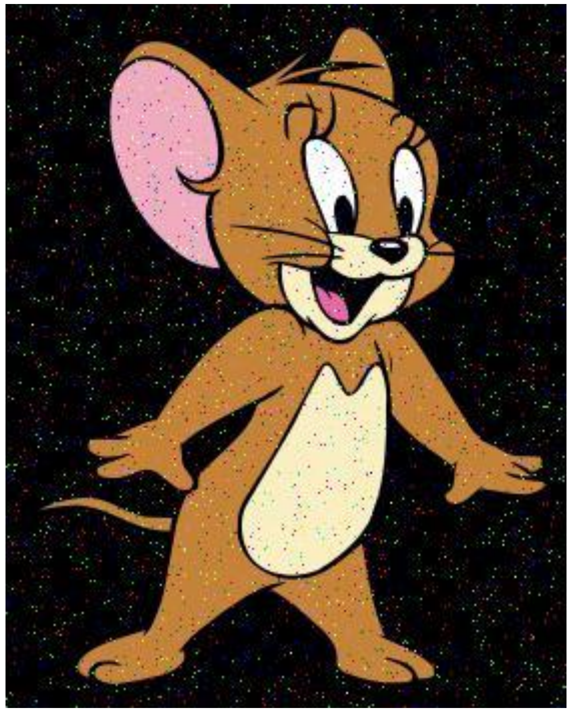


Figure 3 - im_4_noisy

Figure 4 - im_4_filt

Assignment 1
Tudorache Vlad Adrian 442C

Subsection 4:

```
# SUBSECTION 4 ---------------------------------------------------------------
---------------------
im_4_filt = np.copy(img_median)

#create the x axis:

x = list(range(0, 256))

#create the arrays with values from the graph:

y_init = [0, 50, 100, 150, 200, 250]
x_init = [0, 50, 100, 150, 200, 250]
y_final = [0, 100, 140, 170, 200, 250]

#interpolate this values:

interpol = np.interp(x, y_init, y_final)

#display the obtained graph
plt.figure()
plt.plot(x, interpol,'-r', x_init,y_init,'--k'), plt.grid();plt.title("Tone_3
line representation")
plt.show()

#Crate the look-up table:
table = np.array([round(i) for i in interpol])

#spliting the image in 3 channels (R, G, B):

B, G, R = cv2.split(im_4_filt)

#applying the tone on each channel using the look up table:

im_4_tone_B = cv2.LUT(B, table)
im_4_tone_G = cv2.LUT(G, table)
im_4_tone_R = cv2.LUT(R, table)

#reconstructing the image:

im_4_tone = cv2.merge((im_4_tone_B, im_4_tone_G, im_4_tone_R))
plt.imshow(im_4_tone[:, :, ::-1])
plt.show()

cv2.imwrite('im_4_tone.jpg', im_4_tone)
```

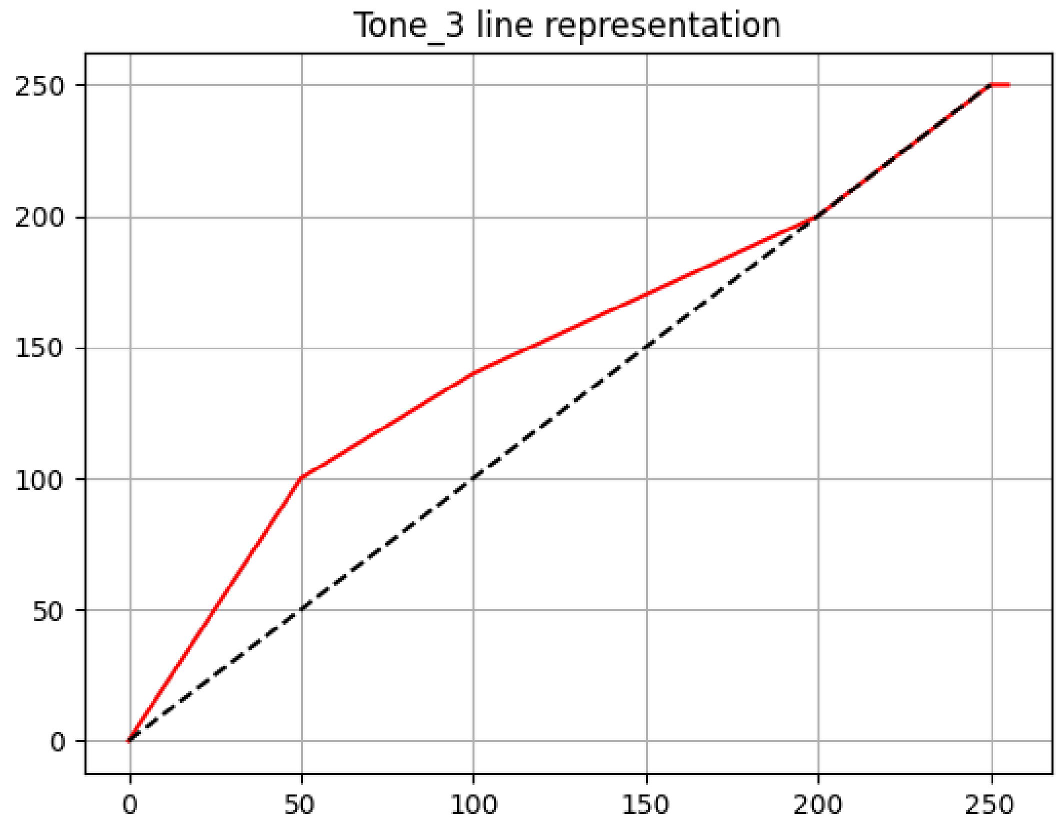


Figure 5 - im_4_tone

Assignment 1
Tudorache Vlad Adrian 442C

Subsection 5:

```
# SUBSECTION 5 ---------------------------------------------------------------
---------------------
im_4_alpha =
cv2.imread(r"C:\Users\Vladuts\Desktop\IPIVA\TEMA1\working_images_2021\4_alpha
.jpg")

# we are going to use arithmetic operations, so we have to transform the
data:

jerrMask = np.uint8(im_4_alpha/255)
jerrBGR = np.uint8(im_4_tone)

#get the cropped part of the background image
BKG = img_cropped.copy()

#defining the region of intrest
roi = BKG[:, :]

#overlaying the region of intrest, the mask, and the color subject

maskedROI = cv2.multiply(roi,(1- jerrMask))
maskedJerry = cv2.multiply(jerrBGR, jerrMask)
BKG_with_Jerry = cv2.add(maskedROI, maskedJerry)

#get the whole background:
big_image = img_CLAHE.copy()

#adding over the intrested area in the whole background
#the resulted image from overalying

big_image[128:480, 335:615] = BKG_with_Jerry

plt.imshow(big_image[:, :, ::-1])
plt.show()
cv2.imwrite('im_4_final.jpg', big_image)
```

Assignment 1
Tudorache Vlad Adrian 442C

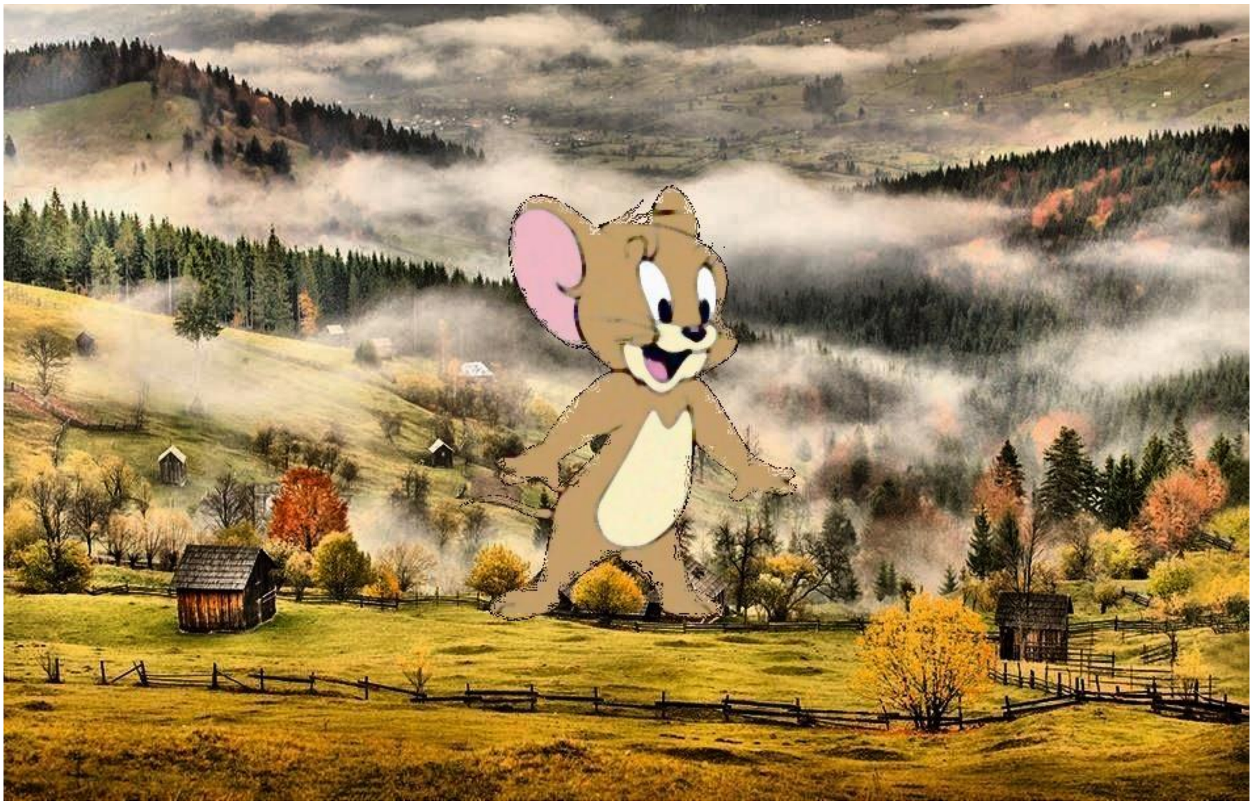


Figure 6- im_4_final