Computer Vision - Ex1

Omer Rugi

April 2020

1 Version and platform

Python version: 3.6.9
Platform: Pycharm

2 Introduction

Read me for the fist task in computer vision curse at Ariel university.
Submitting two .py files:
ex1_utils.py - This file contains function that: convert RGB to YIQ and vise versa, histogram equalization and quantization
gamma.py - This file contains function that creates a window with a slide bar that do gamma correction on a given image.

3 Functions:

def myID() $->$ np.int:
Return my ID (not the friend's ID I copied from)

def imReadAndConvert(filename: str, representation: int) $->$ np.ndarray:
Reads an image, and returns the image converted as requested
:param filename: The path to the image
:param representation: $GRAY\_SCALE$ or RGB
:return: The image object

def imDisplay(filename: str, representation: int):
Reads an image as RGB or GRAY_SCALE and displays it
:param filename: The path to the image
:param representation: GRAY_SCALE or RGB
:return: None

def transformRGB2YIQ(imgRGB: np.ndarray) $- >$ np.ndarray:
Converts an RGB image to YIQ color space
:param imgRGB: An Image in RGB
:return: A YIQ in image color space

def transformYIQ2RGB(imgYIQ: np.ndarray) $- >$ np.ndarray:
Converts an YIQ image to RGB color space
:param imgYIQ: An Image in YIQ
:return: A RGB in image color space

def hist_eq(img: np.ndarray) $- >$ (np.ndarray, np.ndarray, np.ndarray):
This function will do histogram equalization on a given 1D np.array
meaning will balance the colors in the image.
For more details:
https://en.wikipedia.org/wiki/Histogram_equalization
*Original function was taken from open.cv**
:param img: a 1D np.array that represent the image
:return: imgnew $- >$ image after equalization, hist$- >$ original histogram, hist-
new $- >$ new histogram

def hsitogramEqualize(imgOrig: np.ndarray) $- >$ (np.ndarray, np.ndarray, np.ndarray):
Equalizes the histogram of an image
The function will fist check if the image is RGB or gray scale
If the image is gray scale will equalizes
If RGB will first convert to YIQ then equalizes the Y level
:param imgOrig: Original Histogram
:return: imgnew $- >$ image after equalization, hist$- >$ original histogram, hist-
new $- >$ new histogram

def init_hist_limits(img: np.ndarray, nQuant: int):
Creating the histogram of a given 1D image
And given nQuant initializing an array size of nQuant+1
That will be the limits that will devied the histogram into equal parts.
:param imOrig: 1D image
:param nQuant: The number of pixels we wish to represent the new image
:return: hist $- >$ histogram of the image, z$- >$ an array of the limits

def calc_wg_avg_limits(z: np.ndarray, hist: np.ndarray, nQuant: int) $- >$ np.array:
Quantize a single time a given histogram of an image. Calculating the weighted
average (q) of each section of the histogram from z[i] to z[i+1] then placing the
new limits in the average between q[i] and q[i+1]
:param img: Original 1D image
:param hist: the histogram we wish to Quantize
:param nQuant: The number of color scale we wish to Quantize the image
:param z: The limits of the histogram
:return: the new z and q

def replace_to_n_quant(img: np.ndarray,z: np.ndarray, q: np.ndarray, nQuant: int)$-$ >np.ndarray:
reconstructing the new image after the quantization
:param img: Original 1D image
:param z: The limits of the histogram
:param q:the weighted average of each section
:param nQuant: The number of color scale we wish to Quantize the image
:return: newimg $-$ > The new image after Quantize

def quantizeImage(imOrig: np.ndarray, nQuant: int, nIter: int) $-$ > (List[np.ndarray], List[float]):
Quantized an image in to **nQuant** colors
The function will fist check if the image is RGB or gray scale
If the image is gray scale will quantize
If RGB will first convert to YIQ then quantized the Y level
:param imOrig: The original image (RGB or Gray scale)
:param nQuant: Number of colors to quantize the image to
:param nIter: Number of optimization loops
:return: (List[qImage_i],List[error_i])

def gammaDisplay(img_path: str, rep: int):
GUI for gamma correction
Will track nar to change the gamma rate in an image
The gamma will be set as:

$$s = cr^{\gamma}$$

The track bar will be [0-2] in rational numbers.
:param img_path: Path to the image
:param rep: grayscale(1) or RGB(2)
:return: None