Computer vision Assignment 1 report

Task 1:

Starting with an input colour image (let us call this image A), you should combine the three colour-bands into one band using the following equation:

I (x,y) = 0.299 $*$ r(x,y) + 0.587 $*$ g(x,y) + 0.114 $*$ b(x,y) Where r, g and b are colour-bands of image I.

so, basically we have three color bands for the input image , this has to be split into 3 separate matrix using opencv function cv2.split and we could perform arithmetic's operations on each matrix and combine them together using cv2.merge() to apply the given equation to every corresponding pixels on the given image, this will give a single band output (grey band image).
I converted the values of each pixel to the nearest integer using the round function for convenience. After that, i converted the output to the type uint8.

Functions used for task :
convert_to_gray_scale()
convert_to_uint8()
reduce_image()

Task 2:

Firstly i padded my output from task 1 with a width of int(window_size/2) ,where window_size is the size of the square shaped local neighborhood that we assigned. we are padding the matrix with value -1 to accommodate windows for inputs along the boundary elements .

To make a window and to traverse it across the whole picture, i have implemented two for loops, inside which i will select the window region and i will compute the histogram for that window where bin in range 0 to 255 and get the corresponding value of x where the histogram has a maximum value using numpy functions.As the boundary pixels are padded with values -1 and bin is in range 0 to 255, histogram wont consider padded pixels. The values corresponding to each pixel's window is calculated as we traverse along the image. These values are stored in a numpy matrix and it is finally converted to type uint8. This is the output for task 2.

Functions used for task:

padimage()
compute_histogram()
convert_to_uint8()

Task 3:

My task of traversing a window through the whole image for every pixel in image J was done using the same traversal technique in task 2 and if they had same values, their average of corresponding colour intensities of those pixels in image B in each band is calculated, this was done by traversing through every element in each window and checking if they had the same value as that of the element in the center of the window.checking for pixels as same as the center pixel was done using an if statement inside which addition of those corresponding values in each bands of the image and count variable was updated to keep count of the number of same pixels to compute the average and update values in B(x,y) in each band outside the loop of

traversing through the window.

Functions used for task:
oil_paint_effect()
convert_to_uint8()
padimage()

Challenges faced:

compute_histogram() and oil_paint_effect() has 2 and 4 for loops respectively , so their complexity of time will be of the factor n^2 and n^4. This has lead to a very long time(over an hour) to get the outputs while taking in these higher resolution images as inputs, so i included a function named reduce_image() to reduce the resolution of the image keeping the aspect ratio.

I have reduced sample images light_rail.jpg and dog.jpg with a factor of 30 and 40 respectively.

Functions used :

reduce_image(): Takes in image matrix and factor of reduction as parameters and returns a resolution reduced image.

compute_histogram(): This function does most of the task2 of the specification by computing the histogram in each window and finding the most frequent value in each window and creating a matrix with the result in each window and this matrix is returned.

convert_to_grayscale(): This function does most of the task1 of the specification and return the resultant matrix

convert_to_uint8(): This function converts the matrix to type uint8 where values from white to black range from values from 0 to 255

padimage(): Takes in window size and matrix to be padded as parameters .
pads the image according to the window size and returns the padded matrix.

display_image(): Function to display the image

save_image(): Function to save image.

oil_paint_effect(): This function does most of the the task 3 of the specification and returns the resultant image.

outputs and interpretation:

Task 1

Task 2

window size = 5

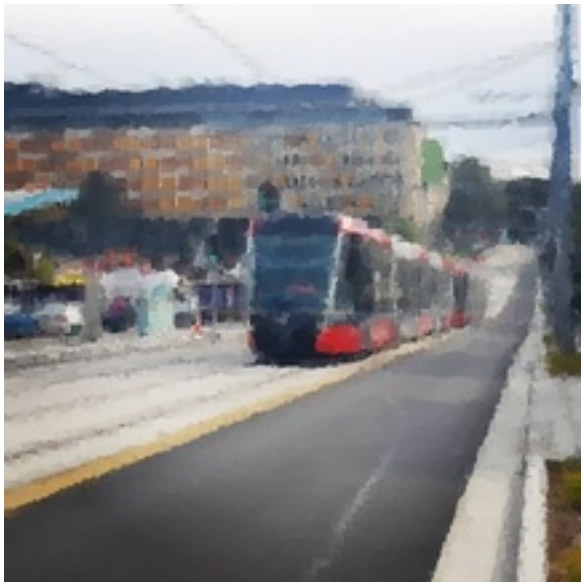
window size =11

window size=17

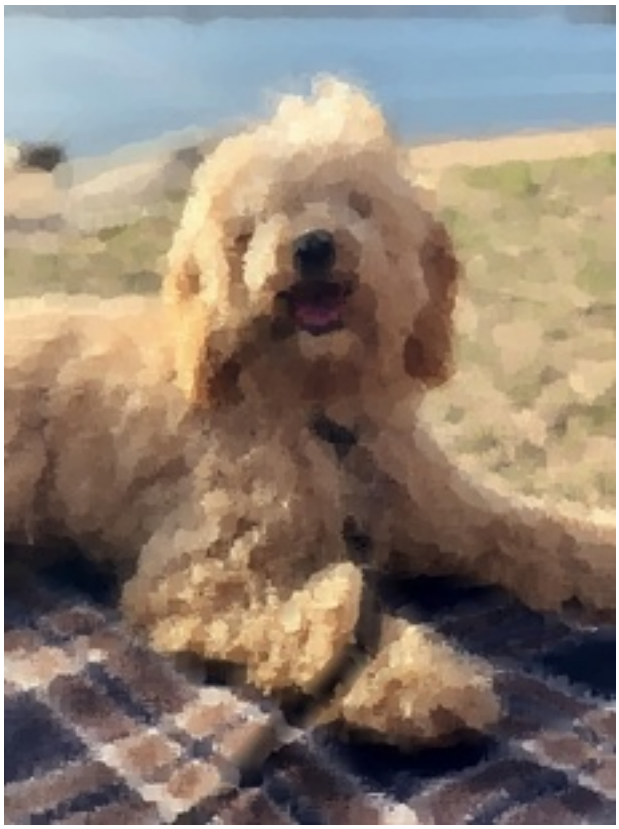
Task 3

window size=5

window_size=11



window size= 17



we can see that as the window size is increased the oil painting effect is intensified, this is reasonable as when we consider a higher window size , there is larger region of interest in computing the most frequent value as for task 2 and averaging the colour intensities as for task 3. Hence the outputs are as expected.