INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
NAGPUR

Digital Image Processing

Poster Report

Submitted by:

K.Uday Raj(BT19ECE066)

M.Saiteja(BT19ECE068)

M.Saikumar(BT19ECE073)

6th Semester
Department of Electronics and Communication Engineering

Submitted to:
Dr.Tapan Jain
Course Instructor

Title:

Color Quantization by Median-Cut Algorithm

Theory:

Color Quantization:

Color Image Quantization is regarded as one of the most important technique in image processing due to various applications in real-world scenario.Color Quantization methods can be classified into two groups: image-independent methods and image-dependent methods.The image-independent images define palettes without providing notice to any specific image, therefore, these methods are very fast but the results are poor.On the other hand, image-dependent methods describe the palette based on the color distribution. These image-dependent methods are slow when compared to the image-independent methods but improves the results.Color quantization is a process that reduces the number of colors in images without loss of quality and of important global information.Pixels in images could have associated 24-bits containing at most $2^{24} = 16{,}777{,}216$ different colors. These colors are represented as three dimensional vectors, each vector element with 8-bit dynamic range, allowing $2^8 = 256$ different values.These vectors are often called RGB triplets.A smaller set of representative colors of the image is called color palette.The Color Quantization means to classify similar

colors into one group and then replace the group with a quantized color.The quantization process can be described as: to search and calculate the color value of each pixel in the original image so as to define its corresponding pixels in the target image.The ultimate target of the Color Quantization is to reduce, with the least distortion, the amount of color of a image, therefore the main object of the Computer Graphics in the field of color quantization is to select the best color palette which can minimize the differences between the original image and the quantized one.

Introduction to Median Cut Algorithm:

- Median-Cut Algorithm is a popular algorithm.The idea behind this algorithm is to split the color space into sub-blocks using median value such that the sub-blocks have the same color dots.
- Generally the blocks that have the maximum number of color dots are choosen for the split so that the overall quantization error is low.
- The idea of Median-cut algorithm is to make every color at the Color Quantization Table represent approximately the same amount of pixels of original images.

Median Cut Algorithm:

1. Move all pixels into a single large bucket.
2. Find the color channel (red, green, or blue) in the image with the greatest range.
3. Sort the pixels by that channel values.
4. Find the median and cut the region by that pixel.
5. Repeat the process for both buckets until you have the desired number of colors.

For example, In your image, if the blue channel has the greatest range, then a pixel with an RGB value of (32, 8, 16) is less than a pixel with an RGB value of (1, 2, 24), because 16 ¡ 24. Sort the pixels along blue channel. After the bucket has been sorted, move the upper half of the pixels into a new bucket. Repeat the process on both buckets, giving you 4 buckets, then repeat on all 4 buckets, giving you 8 buckets,then repeat on all 8, giving you 16 buckets. Average the pixels in each bucket and you have a palette of 16 colors.Since the number of buckets doubles with each iteration, this algorithm can only generate a palette with a number of colors that is a power of two.

Code:

```
import matplotlib.pyplot as plt
import numpy as np
from skimage.io import imread
import cv2
img = cv2.imread('DIP\girl.jpg')
sample_img = cv2.imread('DIP\girl.jpg')

def median_cut_quantize(img, img_arr):
    # when it reaches the end, color quantize
    print("to quantize: ", len(img_arr))
    r_average = np.mean(img_arr[:,0])
    g_average = np.mean(img_arr[:,1])
    b_average = np.mean(img_arr[:,2])

    for data in img_arr:
        sample_img[data[3]][data[4]] = [r_average, g_average, b_average]
```

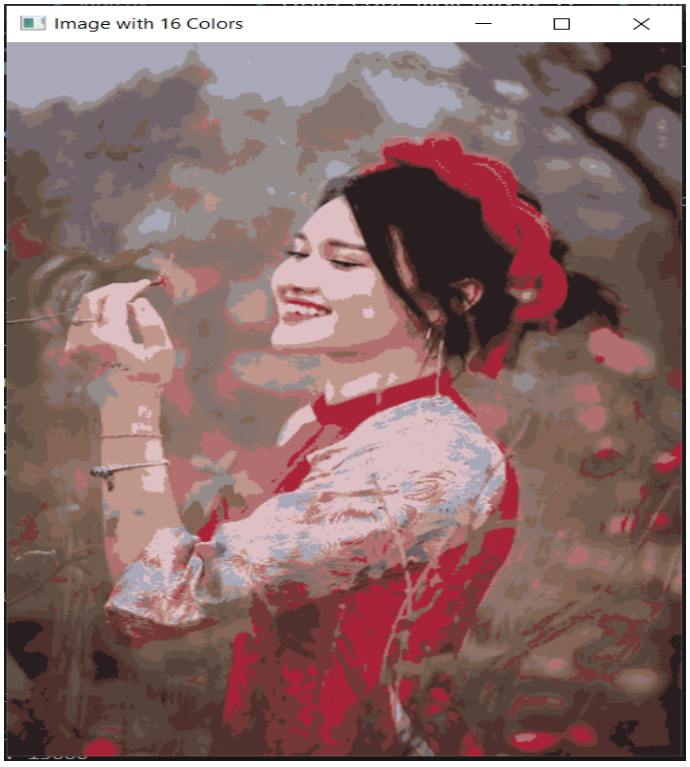
```
def split_into_buckets(img, img_arr, depth):

    if len(img_arr) == 0:
        return

    if depth == 0:
        median_cut_quantize(img, img_arr)
        return

    r_range = np.max(img_arr[:,0]) - np.min(img_arr[:,0])
    g_range = np.max(img_arr[:,1]) - np.min(img_arr[:,1])
    b_range = np.max(img_arr[:,2]) - np.min(img_arr[:,2])

    space_with_highest_range = 0

    if g_range >= r_range and g_range >= b_range:
        space_with_highest_range = 1
    elif b_range >= r_range and b_range >= g_range:
        space_with_highest_range = 2
    elif r_range >= b_range and r_range >= g_range:
        space_with_highest_range = 0

    print("space_with_highest_range:",space_with_highest_range)

    # sort the image pixels by color space with highest range
    # and find the median and divide the array.
    img_arr = img_arr[img_arr[:,space_with_highest_range].argsort()]
    median_index = int((len(img_arr)+1)/2)
    print("median_index:", median_index)


    #split the array into two buckets along the median
    split_into_buckets(img, img_arr[0:median_index], depth-1)
    split_into_buckets(img, img_arr[median_index:], depth-1)


flattened_img_array = []
for rindex, rows in enumerate(sample_img):
    for cindex, color in enumerate(rows):
        flattened_img_array.append([color[0],color[1],color[2],rindex,
   cindex])

flattened_img_array = np.array(flattened_img_array)

# the 3rd parameter represents how many colors are needed in the power
   of 2. If the parameter
# passed is 4 its means 2^4 = 16 colors
split_into_buckets(sample_img, flattened_img_array, 7)

cv2.imshow("Original Image",img)
```

```
cv2.waitKey(0)
cv2.imshow("Image with 128 Colors",sample_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Results:

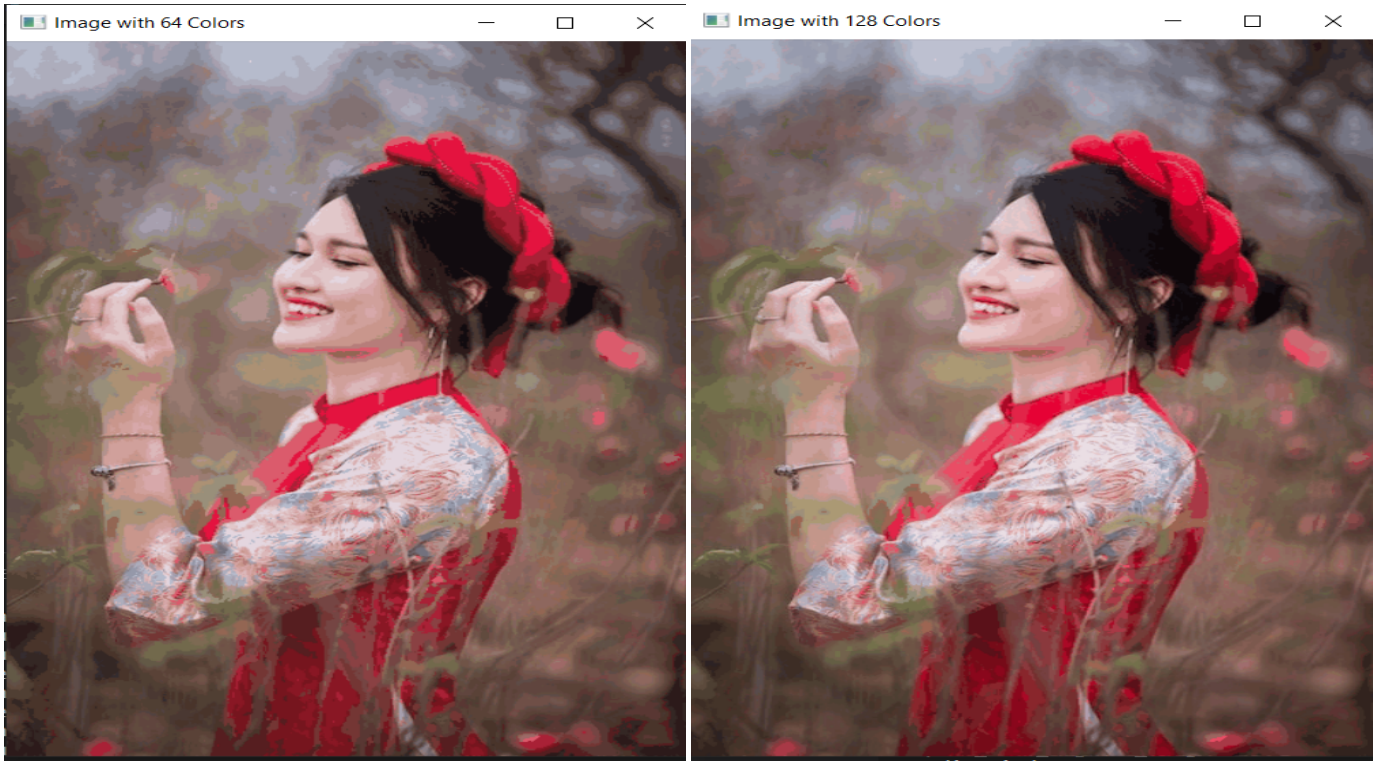
(a) Original Image

(b) 8 Colors



(a) 16 Colors

(b) 32 Colors



(a) 64 Colors (b) 128 Colors

Conclusion:

In the Poster,we presented Color Quantization using Median Cut Algorithm,Color Quantization is a process that reduces the number of distinct colors used in an image, usually with the intention that the new image should be as visually similar as possible to the original image.

Scan Me