REPORT

Submitted by

Rohit katariya -2020CSZ8844

Geetha Pavani Pappu -2020EEZ8644

COURSE: DIGITAL IMAGE ANALYSIS (COL 783)

ASSIGNMENT -1

The Assignment is an implementation of the articles “Color Image Quantization for Frame Buffer Display” and “Transferring Color to Greyscale Images”.

Contents of Report

INTRODUCTION

IMPLEMENTATION

Part-1

Quantization

Popularity algorithm

Median Cut algorithm

Floyd Steinberg dithering

EXPERIMENTS & RESULTS

Varying the parameters

Comparison of Quantization methods

Part- 2

Color Transfer

Global Image Matching

Swatches

EXPERIMENTS & RESULTS

Varying the parameters

Comparison of Color Transfer Methods

INTRODUCTION:

This Assignment includes two different implementation of basic image processing techniques, i.e, Color Quantization and transferring Color. Color quantization is the general technique to reduce the number of color spaces with an intuition of obtaining a new image which identical to original image. Color transferring is a image processing algorithm through which gray scale image is converted into RGB image using source image of relative to gray scale image. Generally, there are many algorithms to implement these algorithms. However, we consider “Popularity algorithm”, “Median-Cut algorithm” and “Floyd Steinberg algorithm” for color quantization. In addition to this, we have considered Global Image Matching and Swatch Matching algorithms for color transferring.



(a)

(b)



Gray scale Image

Source Image

Color Transferred Image

Fig.1 Color Quantization (a) original images (b) Quantized images into K colors

Fig.2 Color Transferring of image from gray scale to RGB scale



PART 1- Quantization and Dithering

IMPLEMENTATION:

The assignment is implemented in python, Open CV with Keras library functions, Initially the higher resolution images collected open source are resized to 512 × 512.

Popularity Algorithm:

Initially, the images are processed to count the colors and respective frequencies. Each color vector (pixel), is considered as a single entity. To quantize the image using popularity algorithm following steps are considered.

Step 1: Find the count of each color on the image with respective coordinate values.

Step 2: Select the k highest frequency colours in the image and store it in a colour palette.

Step 3: Create a KD tree using the colour intensity values.

Step 3: Now, using the above KD tree, each colour pixel is replaced with the nearest pixel of color palette. i.e., with pixel having smallest distance from source pixel.

The algorithm quiet efficiently quantizes the images which don’t have a very skewed colour distribution. However, it is less appropriate in practical applications.

Median-cut Algorithm:

The other classical method, Median-cut is also similar to popularity algorithm. As a prior step similar to popularity, it counts the number of colors with respective size. The following steps are performed to obtain the quantized image with median cut method:

Step1: Create a box from original image which contains all the colors and respective count of them.

Step2: Now, the box is recursively splitted into smaller boxes using the median value obtained along its three axes. Now each sub box contains the nearest closet pixels.

Step3: The recursive splitting resultant of these colors is repeated until to obtain k number of sub boxes.

Step4: As a final step, the pixels in each sub box are replaced with respective average pixel values of the sub box.

The main advantage of this method is that color regions of high pixel density are split into smaller, which reduces the overall quantization error. However, in low pixel density regions, this process may lead to individual pixel deviations.

Floyd Steinberg Dithering:

It is an error dispersion-based algorithm. The following steps are followed to perform dithering approach:

Step1: Initially, consider the original image and quantized image obtained in the prior step.

Step2: Next iterate over the image in scan line algorithm.

Step3: Find the nearest colour pixel using KD Trees and consider the difference of each pixel of the original image with the nearest colour pixel of stored coloured palette. The difference is termed as “Quantization Error”.

Step4: The computed error dispersion is spread over to the next consecutive pixels in top, down and diagonal directions.

Step5: Repeat the step 3 and step 4 to all the pixels of the original image.

# Experiments and Observations

# 40 colours

|  |  |
| --- | --- |
| art3  Input image | |
| Popularity Algorithm | median40  Median cut |
| popularity_dither40  Popularity+Dithering | dither_median  Median Cut + Dithering |

# 128 colours

|  |  |
| --- | --- |
| art3  Input image | |
| popularity128  Popularity Algorithm | median128  Median cut |
| popularity_dither128art3  Popularity+Dithering | dither_median128  Median Cut + Dithering |

K - 128 Colors

|  |  |
| --- | --- |
| art2  Input image | |
| popularity128  Popularity Algorithm | median128  Median cut |
| popularity_dither128_art2  Popularity+Dithering | dither_median128  Median Cut + Dithering |

**Observations:**

From the above we observe that with 40 or less colours, the Popularity algorithm performs very poorly. The median cut algorithm is quiet efficient here. Median Dithering has removed sudden points of jittering.

We observed that Popularity algorithm takes much longer to compute than median cut. But this time was reduce to a certain extent on moving to KD trees rather than exahustive search.

The median cut gives reasonable output in most situations but popularity gives reasonable output only for images which don’t have very skewed color distribution.



Popularity -512 colours



Median cut 512 colors

Dithering removed sudden changes in color and jittered noise from the image.

PART 2- Transferring Colour to Grayscale Images

Here we explore 2 methods for colour transfer to a grayscale image. The first is the Global Image Matching that uses the mood/colour distribution of the whole image while computing chromatic values of any pixel. The other method is swatches-based method in which the we use only the user specified pixels to transfer the image colour for every point on the image.

We use 2 information i.e. intensity values obtained via L component of the LAB conversion of the image. The second is the neighbourhood statistics to capture the texture values while predicting the chromatic values of each pixel.

Global Image Matching

Step 1: We compute the LAB image of the colour image.

Step 2: We apply histogram matching on the luminance component to bring the histogram of the colour image to match the grayscale image.

Step 3: Now we do jittered sampling on the source-coloured image and choose one pixel from each box of the grid we made. This brings a good sampling from all parts of the image.

Step 4: Once we have samples from the above image, we pre-compute the neighbourhood statistics of each sample pixel. A 5x5 pixel box around pixel is used to compute the neighbourhood statistics.

Step 5: Now we go through the image pixels in scan line algorithm and find their intensity and neighbourhood statistics and give equal weightage to intensity and neighbourhood matching.

Step 6: We find the point in the sample points which is closest to this and assign the chromatic value of this pixel to the output image, and copying the intensity of the original grayscale image to the output image.

Swatch Colour Transfer

Step 1: We compute the LAB image of the colour image.

Step 2: We divide the source image into a grid and all the grid boxes one by one in scan line order.

Step 3: For each swatch we ask the user to select a corresponding swatch in the coloured source image.

Step 4: Once we have the 2 swatches, we apply colour transfer (luminance remapping -> neighbourhood statistics -> closest/lowest error sample points) Here we match each pixel to sample points from within the swatch in colour image selected.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Grayscale** | **Source Image** | **Global Image Matching** | **Swatches Color Transfer** |
| bw_ppr | color_ppr | 01_35_09ti_ppr | 00_01_41ti_ppr |
| bw_wheat | color_wheat | 00_09_13ti_wheat | 00_02_19ti_wheat |
| bw_ja | color_ja | 23_50_38ti_ja | 21_58_08ti_ja |
| bw_road2 | color_road2 | 23_59_47ti_road2 | 00_07_36ti_road2 |
| bw_agri | color_agri | 00_14_47ti_agri | 00_13_52ti_agri |

# Experiments and Observations

Observations:

We see that the global matching algo prefrorms well when there are not too many colors in the image. Whereas the swatch based matching is able to do much better because it is gets user’s input telling from what part of the image it should pick the colours.

With swatches, we are able to cover images with wide range of colours but we could find that image is getting distinct swatch boundaries which are not looking that good.