Image Processing

Slot: C2+TC2

Names (Register no):

1. Ghuwalewala Karan Rakesh (22BLC1201)
2. Piyush Arora (22BLC1217)
3. Ayush Arora (22BLC1218)
4. Ayush Negi (22BLC1259)-Group coordinator
5. Adarsh Singh (22BLC1372)

Introduction

Image enhancement is a type of image processing which can be done using Retinex Algorithms. These algorithms use Gaussian distribution function and logarithmic domain to perform image enhancement. Different types of Retinex Algorithms are there such as Single Scale Retinex for adding colour to the image, Multi Scale Retinex to improve the contrast of the image generated by SSR, Multi Scale with Colour Restoration to remove the grey filter generated by MSR and Multi Scale Retinex with colour Preservation to intensify the image generated by MSRCR. These algorithms will be implemented in Python using the libraries like opencv, cv2, numpy, scipy, pandas, etc.

Image Enhancement using Single Scale Retinex

- Image Enhancement uses Single Scale Retinex Algorithm.
- Logarithmic domain is used to reduce time complexity.(Multiplication changes to addition.)
- Gaussian function is used as it's logarithm produces a quadratic curve which is easier to manipulate.
- Single Scale Retinex is limited to lower values of delta(upto approximately 330). Hence, the filter can only be high pass.

Single Scale Retinex

Single-scale retinex (SSR) is defined as the difference between the image at a given pixel (x,y) and the center-surround average of that pixel (x,y).

The calculation of the above surrounding pixels average is equal to the inverse square spatial function for a given pixel. And we can use any high pass filter to calculate the surrounding average (Ex. Gaussian distribution) that satisfies the above conditions.

If we consider Gaussian function (G_σ) as center-surround function, then retinex image for each i-th channel in an image is,

$$SSR_i(x, y) = \log(I_i(x, y)) - \log(G_\sigma * I_i)(x, y))$$

That is the Single-scale retinex of an image is estimated by taking the logarithm difference of image and point-surround filter of the image at position (x,y). The operation $(G_\sigma * I_i)(x, y)$ is nothing but gaussian blur of an image with given scale (σ).

Image Enhancement using Multi Scale Retinex

- Image Enhancement can also be done using Multi Scale Retinex Algorithm.
- Logarithmic domain is used to reduce time complexity.(Multiplication changes to addition.)
- Guassian function is used as it's logarithm produces a quadratic curve which is easier to manipulate.
- In computer science, **255 is the maximum value representable by an eight-digit binary number**

Multi Scale Retinex

As the choice of σ varies for different images for good results and different scale values enhance different parts of an image, we can combine **SSR** of different scales and give weightage for each scale and take a summation of all weighted-SSR images. This method is called **Multi-scale retinex (MSR)** of an image and it is defined as the weighted average of **n** single-scale retinex images for different σ values.

$$\mathrm{MSR_i}(x, y) = \sum_{n=1}^{N} w_n \mathrm{SSR_i}(x, y)$$

As the output $\mathrm{MSR_i}(x, y)$ image might contain negative real values and the range of values is not suitable for image representaion i.e not in range **[0-255]**, normalize the **MSR** output for range **[0-255]** given by the following equation

$$\mathrm{MSR_i}(x, y) = 255 \frac{\mathrm{MSR_i}(x, y) - \min(\mathrm{MSR_i})}{\max(\mathrm{MSR_i}) - \min(\mathrm{MSR_i})}$$

where for channel i, $\mathrm{MSR_i}(x, y)$ is the pixel value at position (x, y), $\min(\mathrm{MSR_i})$ is minimum value of the channel, and $\max(\mathrm{MSR_i})$ is maximum value of the channel

Image Enhancement using Multi Scale Retinex with Color Restoration

As the **MSR** of an image looks colorless, the Multi-scale retinex output is multiplied with **Color-restoration function (CRF)** of chromaticity to restore the original colors of the input image approximately. And this method of calculating the Color-restoration function and applying it to Multi-scale retinex output is called **Multi-scale retinex with Color Restoration (MSRCR)**.

$$\mathrm{MSRCR_i(x,y) = MSR_i(x,y) * CRF_i(x,y)}$$

where $\mathrm{CRF_i(x,y)}$ is color restoration vlaue for pixel (x,y) at i-th channel. The color restoration function is defined as

$$\mathrm{CRF_i(x,y) = \beta[\log(\alpha * I_i'(x,y)]}$$

where for i-th channel, at position $\mathrm{(x,y)}$, **CRF** depends on the ratio composition of the pixel at (x,y) for that channel value to the sum of all channel values which is equal to calculating chromaticity coordinates. Chromaticity coordinates are calculated as

$$\mathrm{I_i'(x,y) = \frac{I_i(x,y)}{\sum_{c=0}^{k-1} I_c(x,y)}}$$

where k equals to no. of image channels, α is to control non-linearity, and β is to control total gain. The above equation can be written as

$$\mathrm{CRF_i(x,y) = \beta[\log(\alpha * I_i(x,y)) - \log(\sum_{c=0}^{k-1} I_c(x,y))]}$$

To achieve better contrast results, the **MSRCR** equation is modified to include **gain (G)** and **offset (b)** values.

$$\mathrm{MSRCR_i(x,y) = G[MSR_i(x,y) * CRF_i(x,y) - b]}$$

The gain and offset values are introduced to transform the contrast range of the **MSRCR** image to distribute uniformly and to attenuate tails forming in the histogram graph. But these values are not general and don't work for every image. And to stretch the contrast, the final **MSRCR** image is **contrast stretched** using **Simplest Color Balance** algorithm with a clipping percentage of 1% at both ends.

Image Enhancement using Multi Scale Retinex with Colour Preservation

In the previous algorithm **MSRCR**, color restoration was the main issue, and to address that we have introduced many variables and operations. All these calculations are computed directly on each channel value that changes chromaticity coordinates/color composition and the final result is unwanted colors, reversed color order, and sometimes grayish region due to surrounding average. To maintain the chromaticity/color composition as it is and also enhance the color contrast globally on the image, we can apply Multi-scale retinex on **intensity** image-channel which is just an addition of all image channels divided by the total number of channels.

$$\mathbf{Int_{(x,y)}} = \frac{\sum_{c=0}^{k-1} \mathbf{I_c(x,y)}}{\mathbf{k}}$$

where $\mathbf{k}$ is no. of image channels.

After applying Multi-scale retinex to the intensity image, apply contrast stretch like applied in **MSRCR** to set the image values in the range **[0-255]** with uniform distribution of histogram values.

$$\mathbf{RInt_{(x,y)}} = \mathbf{MSR(Int_{(x,y)})}$$

after the **MSR**, apply the color balance step with a percentage clipping of 1% on both sides.

Finally, preserving the initial chromaticity ratio between each image-channel and intensity image, multiply the ratio with enhance-intensity image channel for each image-channel to get the whole image.

As we are applying retinex enhancement only on intensity-image, each color changes proportional to the ratio between enhance-intensity and normal intensity values. This keeps the relative intensity between surrounding colors locally and globally and gives better results than **MSRCR**.

$$\mathbf{R_i(x,y)} = \mathbf{I_i(x,y)} \frac{\mathbf{RInt_i(x,y)}}{\mathbf{Int_{(x,y)}}} = \mathbf{I_i(x,y)} * \mathbf{A_{(x,y)}}$$

$$\mathbf{A_{(x,y)}} = \mathbf{min}(\frac{\mathbf{MAX\_VALUE}}{\mathbf{B}}, \frac{\mathbf{RInt_{(x,y)}}}{\mathbf{Int_{(x,y)}}})$$

$$\mathbf{B_{(x,y)}} = \mathbf{max}(\mathbf{I_c(x,y)}, \mathbf{c} \in \{\mathbf{0 \ldots k-1}\})$$

Platform used

- Python is used for image enhancement. Modules like NumPy, open cv, SciPy etc are imported.
- Open cv is generally preferred over other modules like NumPy and SciPy as it uses Fast Fourier Transform whereas NumPy and SciPy use Fourier Transform.

Application

Application of Image Enhancement

- Image sharpening and restoration(can be achieved by the above discussed algorithms).
- Medical field.
- Remote sensing.
- Transmission and encoding.
- Machine/Robot vision.
- Color processing.
- Pattern recognition.
- Video processing.