CSPE31 IMAGE PROCESSING

Programming Assignment

on

Intensity Transformations

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPPALLI – 620 015

March 2020

submitted by

N. D. Jagan 106116058

G.Sai Charan 106116076

Table of Contents

PROBLEM STATEMENT

Intensity Transformation Operations on Images using following methods

1. Image Negatives (Linear)
2. Log Transformations
3. Power-Law (Gamma) Transformations
4. Piecewise-Linear Transformation Functions

f (x, y) - Input Image function represented in terms of pixels

g (x, y) - Output Image function represented in terms of pixels

g (x, y) = T (f (x, y))

T- operator to perform transformations

Image Negatives

Mathematically, assume that an image goes from intensity levels 0 to (L-1). Generally, L = 256. Then, the negative transformation can be described by the expression **s=L-1-r**
where r is the initial intensity level and s is the final intensity level of a pixel.

Log Transformations

Mathematically, log transformations can be expressed as **s = clog(1+r)**
Here c is a scaling constant. c is given by **255/ (log (1 + m))**
where m is the maximum pixel value in the image. It is done to ensure that the final pixel value does not exceed (L-1), or 255. Practically, log transformation maps a narrow range of low-intensity input values to a wide range of output values.

Power Law (Gamma) Transformations

It can be mathematically expressed as $\mathbf{s=cr^{y}}$. Here y is gamma value. Gamma correction is important for displaying images on a screen correctly to prevent bleaching or darkening of images when viewed from different types of monitors with different display settings

Piecewise- Linear Transformation

These functions, as the name suggests, are not entirely linear in nature. However, they are linear between certain x-intervals. One of the most used piecewise-linear transformation functions is contrast stretching.

Contrast can be defined as:

Contrast= $(I_{max} - I_{min})/ (I_{max} + I_{min})$

With (r1, s1), (r2, s2) as parameters, the function stretches the intensity levels by essentially decreasing the intensity of the dark pixels and increasing the intensity of the light pixels. If r1 = s1 = 0 and r2 = s2 = L-1, the function becomes a straight dotted line in the graph (which gives no effect). The function is monotonically increasing so that the order of intensity levels between pixels is preserved.

Code with Input and Output

jupyter Untitled5 Last Checkpoint: Last Wednesday at 2:27 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [43]:
```
import cv2
import numpy as np
from IPython.display import display, Image
import warnings
warnings.filterwarnings("ignore")

# Open the image.
img = cv2.imread('sample.jpg')
print('Input Image')
display(Image(filename='sample.jpg'))
```

```
Input Image
```

```
# Apply negative transform
negative_transformed = np.array(255-img, dtype = np.uint8)
cv2.imwrite('negative_transformed.jpg', log_transformed)
print('negative_transformed')
display(Image('negative_transformed.jpg'))
```

```
negative_transformed
```

```
# Apply log transform.
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)
log_transformed = np.array(log_transformed, dtype = np.uint8)
cv2.imwrite('log_transformed.jpg', log_transformed)
print('log_transformed')
display(Image(filename='log_transformed.jpg'))
```

log_transformed

```
#Power-Law (Gamma) Transformation for different gamma values
for gamma in [0.1, 0.6, 1.2, 1.8]:
    gamma_corrected = np.array(255*(img / 255) ** gamma, dtype = 'uint8')
    filename='gamma_transformed '+str(gamma)+'.jpg'
    cv2.imwrite(filename, gamma_corrected)
    print(filename)
    display(Image(filename))
```

gamma_transformed 0.1.jpg

gamma_transformed 0.6.jpg

gamma_transformed 1.2.jpg

gamma_transformed 1.8.jpg

```
#Piecewise-Linear Transformation
# Function to map each intensity level to output intensity level.
def pixelVal(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix
    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2

# Define parameters.
r1 = 50
s1 = 0
r2 = 120
s2 = 255

# Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelVal)

# Apply contrast stretching.
contrast_stretched = pixelVal_vec(img, r1, s1, r2, s2)
contrast_stretched_new=np.array(contrast_stretched,dtype = 'uint8')
cv2.imwrite('contrast_stretched_new.jpg', contrast_stretched_new)
display(Image('contrast_stretched_new.jpg'))
```