FuzzyGenerator

July 20, 2021

1 FUZZYGenerator

[1]:
```
!pip show opencv-python
```

```
Name: opencv-python
Version: 4.5.3.56
Summary: Wrapper package for OpenCV python bindings.
Home-page: https://github.com/skvark/opencv-python
Author: None
Author-email: None
License: MIT
Location: e:\programs\anaconda\lib\site-packages
Requires: numpy
Required-by:
```

[8]:
```
!pip install opencv-python
```

```
Collecting opencv-python
  Downloading opencv_python-4.5.3.56-cp38-cp38-win_amd64.whl (34.9 MB)
Requirement already satisfied: numpy>=1.17.3 in e:\programs\anaconda\lib\site-
packages (from opencv-python) (1.19.2)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.3.56
```

[3]:
```
import math

import cv2
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, Markdown
from glob2 import glob

PATH = 'ImagenesPropias'
```

[13]:
```
# Gaussian Function:
def G(x, mean, std):
    return np.exp(-0.5*np.square((x-mean)/std))

# Membership Functions:
```

```
def ExtremelyDark(x, M):
    return G(x, -50, M/6)

def VeryDark(x, M):
    return G(x, 0, M/6)

def Dark(x, M):
    return G(x, M/2, M/6)

def SlightlyDark(x, M):
    return G(x, 5*M/6, M/6)

def SlightlyBright(x, M):
    return G(x, M+(255-M)/6, (255-M)/6)

def Bright(x, M):
    return G(x, M+(255-M)/2, (255-M)/6)

def VeryBright(x, M):
    return G(x, 255, (255-M)/6)

def ExtremelyBright(x, M):
    return G(x, 305, (255-M)/6)
```

[11]:
```
def OutputFuzzySet(x, f, M, thres):
    x = np.array(x)
    result = f(x, M)
    result[result > thres] = thres
    return result

def AggregateFuzzySets(fuzzy_sets):
    return np.max(np.stack(fuzzy_sets), axis=0)

def Infer(i, M, get_fuzzy_set=False):
    # Calculate degree of membership for each class
    VD = VeryDark(i, M)
    Da = Dark(i, M)
    SD = SlightlyDark(i, M)
    SB = SlightlyBright(i, M)
    Br = Bright(i, M)
    VB = VeryBright(i, M)

    # Fuzzy Inference:
    x = np.arange(-50, 306)
    Inferences = (
        OutputFuzzySet(x, ExtremelyDark, M, VD),
        OutputFuzzySet(x, VeryDark, M, Da),
```

```
        OutputFuzzySet(x, Dark, M, SD),
        OutputFuzzySet(x, Bright, M, SB),
        OutputFuzzySet(x, VeryBright, M, Br),
        OutputFuzzySet(x, ExtremelyBright, M, VB)
    )

    # Calculate AggregatedFuzzySet:
    fuzzy_output = AggregateFuzzySets(Inferences)

    # Calculate crisp value of centroid
    if get_fuzzy_set:
        return np.average(x, weights=fuzzy_output), fuzzy_output
    return np.average(x, weights=fuzzy_output)
```

[ ]:
```
data = np.array()
```

[4]:
```
data = np.array([cv2.cvtColor(cv2.imread(p), cv2.COLOR_BGR2RGB) for p in␣
↪glob(f'{PATH}/*')])
data.shape
```

[4]:
```
(1, 299, 299, 3)
```

[9]:
```
# Proposed fuzzy method
def FuzzyContrastEnhance(rgb):
    # Convert RGB to LAB
    lab = cv2.cvtColor(rgb, cv2.COLOR_RGB2LAB)

    # Get L channel
    l = lab[:, :, 0]

    # Calculate M value
    M = np.mean(l)
    if M < 128:
        M = 127 - (127 - M)/2
    else:
        M = 128 + M/2

    # Precompute the fuzzy transform
    x = list(range(-50,306))
    FuzzyTransform = dict(zip(x,[Infer(np.array([i]), M) for i in x]))

    # Apply the transform to l channel
    u, inv = np.unique(l, return_inverse = True)
    l = np.array([FuzzyTransform[i] for i in u])[inv].reshape(l.shape)

    # Min-max scale the output L channel to fit (0, 255):
    Min = np.min(l)
```

```
    Max = np.max(l)
    lab[:, :, 0] = (l - Min)/(Max - Min) * 255

    # Convert LAB to RGB
    return cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)

# Traditional method of histogram equalization
def HE(rgb):
    lab = cv2.cvtColor(rgb, cv2.COLOR_RGB2LAB)
    lab[:, :, 0] = cv2.equalizeHist(lab[:, :, 0])
    return cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)

# Contrast Limited Adaptive Histogram Equalization
def CLAHE(rgb):
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    lab = cv2.cvtColor(rgb, cv2.COLOR_RGB2LAB)
    lab[:, :, 0] = clahe.apply(lab[:, :, 0])
    return cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)
```

[15]:
```
for i in range(data.shape[0]):
    img = data[i]
    fce = FuzzyContrastEnhance(img)
    he = HE(img)
    clahe = CLAHE(img)
    display(Markdown(f'### <p style="text-align: center;">Sample Photo {i+1}</
 ↪p>'))
    plt.figure(figsize=(224, 224))
    plt.subplot(2, 2, 1)
    plt.imshow(data[i])
    plt.title('Original Image')


    plt.subplot(2, 2, 2)
    plt.imshow(clahe)
    plt.title('CLAHE')


    plt.show()
```

###

Sample Photo 1


DX
ORTO


DX
ORTO