IT5437 – Assignment 1: Intensity Transformations and Neighbourhood Filtering

**Name**: Kaumadi I. A. S.

**Index No:** 249311P

**Module:** IT5437 - Computer Vision

**GitHub Repository:** [Click here](https://github.com/kaumadi123/IT5437_Assignment_249311P)

## Q1. Intensity Transformation and Visualization

Creates a lookup table (LUT) to map input intensities to output intensities.

# Control points (matching the first LUT logic)

points = [(0,0), (49,49), (50,100), (150,150), (255,255)]

xs, ys = zip(\*points)

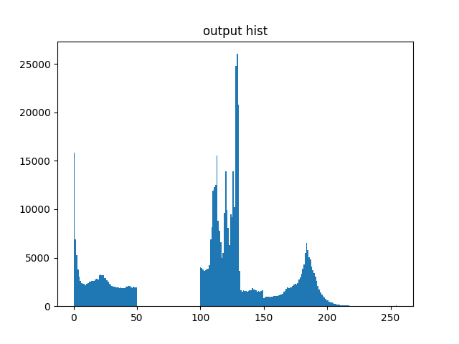
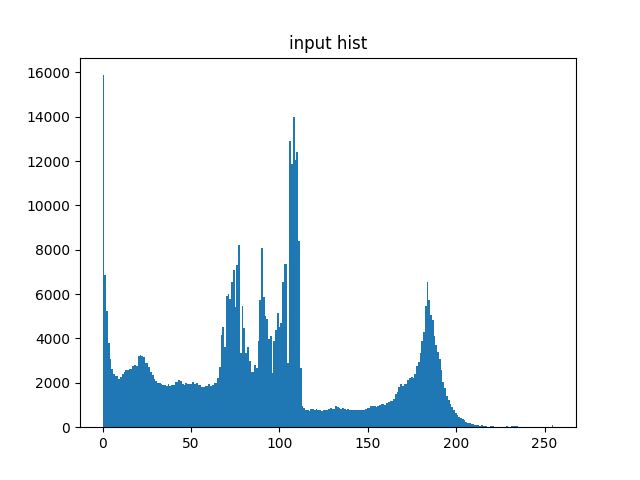
lut = np.interp(np.arange(256), xs, ys).astype('uint8')

# Apply LUT

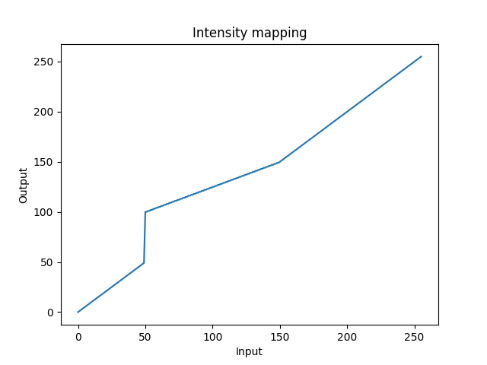
out = cv2.LUT(img, lut)

# Save transformed image

cv2.imwrite("results/q1\_out.png", out)







## Q2. Accentuating White/Gray Matter in Brain PD Images

Making white and gray matter more visible in brain scans.

# Step 2: create LUT functions

def make\_piecewise(points):

    """points = [(x\_in, x\_out), ...]"""

    xs, ys = zip(\*points)

    lut = np.interp(np.arange(256), xs, ys).astype('uint8')

    return lut

# Accentuate white matter

points\_white = [(0,0), (100,60), (140,180), (200,255), (255,255)]

lut\_white = make\_piecewise(points\_white)

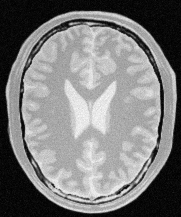
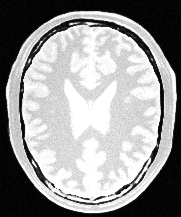
out\_white = cv2.LUT(img, lut\_white)

# Accentuate gray matter

points\_gray = [(0,0), (60,60), (120,200), (160,220), (255,255)]

lut\_gray = make\_piecewise(points\_gray)

out\_gray = cv2.LUT(img, lut\_gray)





## Q3. Gamma Correction on L\* Channel in Lab Color Space

Applying gamma correction (power-law transformation) to the lightness component (L\*) in Lab color space to adjust image brightness and contrast while preserving color information.

lab = cv2.cvtColor(img, cv2.COLOR\_BGR2LAB)

L, a, b = cv2.split(lab)

L\_norm = L / 255.0

gamma = 0.8

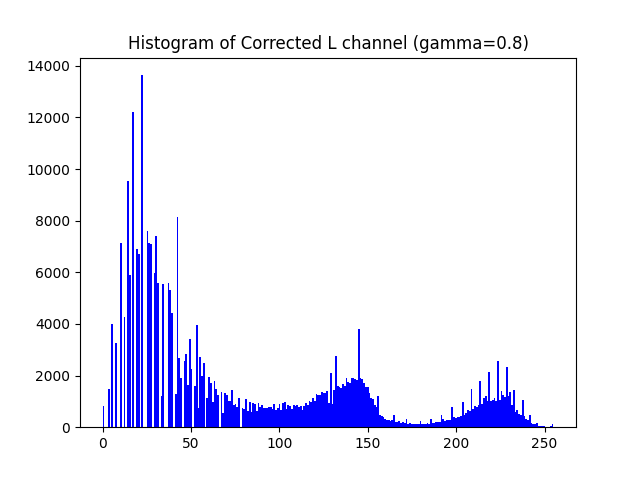
L\_gamma = np.power(L\_norm, gamma)

L\_corrected = np.clip(L\_gamma \* 255.0, 0, 255).astype("uint8")

lab\_corrected = cv2.merge([L\_corrected, a, b])

img\_corrected = cv2.cvtColor(lab\_corrected, cv2.COLOR\_LAB2BGR)





## Q4. Vibrance Enhancement via Nonlinear Saturation Transform

Enhancing image vibrancy by applying a Gaussian-based nonlinear transformation to the saturation channel in HSV color space, selectively boosting mid-saturation regions.

hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

h, s, v = cv2.split(hsv)

sigma = 70.0

a = 0.6

def vibrance\_curve\_gaussian(a=0.6, sigma=70.0):

    x = np.arange(256, dtype=np.float32)

    bump = a \* 128.0 \* np.exp(-((x - 128.0) \*\* 2) / (2.0 \* sigma \*\* 2))

    y = np.minimum(x + bump, 255.0)

    return y.astype(np.uint8)

# Build LUT and apply it to the S channel

lut = vibrance\_curve\_gaussian(a=a, sigma=sigma)

s\_enhanced = cv2.LUT(s, lut)

# Recombine and convert back to BGR

hsv\_enhanced = cv2.merge([h, s\_enhanced, v])

img\_enhanced = cv2.cvtColor(hsv\_enhanced, cv2.COLOR\_HSV2BGR)

