**2.1 Luminance normalization**

def rgb\_to\_ycbcr(image):

# RGB -> YCbCr, return: Y - luminance

R = image[:,:,0]

G = image[:,:,1]

B = image[:,:,2]

Y = 0.299 \* R + 0.587 \* G + 0.114 \* B

Y = Y.astype(np.float64)

return Y, R, G, B

def normalise\_luminance(Y):

# Y - luminance [0, 1] 범위로 정규화 (로그 정규화)

epsilon = 1

Y\_normalised = np.log(Y + epsilon) / np.log(Y.max() + epsilon)

return Y\_normalised

**2.2 Optimal gamma correction parameter estimation**

**Divide the histogram of the input image into two parts: dark and bright regions**

def divide\_luminance(Y):

threshold = 0.5

dark = Y <= threshold

bright = Y > threshold

# dark, bright은 2차원 boolean 배열. 해당부분이 True, 아닌 부분이 False 로 됨

return dark, bright

# 밝은 영역과 어두운 영역을 각각 (3024, 4032) 형태로 유지

def create\_region\_masks(Y\_normalised, dark, bright):

dark\_region = np.zeros\_like(Y\_normalised)

bright\_region = np.zeros\_like(Y\_normalised)

# 어두운 부분은 Y\_normalised의 값을 유지, 나머지는 0

dark\_region[dark] = Y\_normalised[dark]

# 밝은 부분은 Y\_normalised의 값을 유지, 나머지는 0

bright\_region[bright] = Y\_normalised[bright]

return dark\_region, bright\_region

**Newton Method**

def f(gamma, region, sigma, N):

# 최적화 위해서 0 만들어야됨

return (1/N) \* np.sum(region \*\* gamma) - sigma

def df(gamma, region, N):

# f의 도함수 정의

epsilon = 1e-5 # 로그 0 방지

return (1/N) \* np.sum(region \*\* gamma \* np.log(region + epsilon))

def newton\_method(gamma, region, sigma, N, gamma\_min, gamma\_max, es=1e-7, ea=100, iter\_count=0):

if ea <= es:

print(f"Converged gamma: {gamma:.10f} after {iter\_count} iterations.")

return gamma

gamma\_old = gamma

try:

f\_val = f(gamma, region, sigma, N)

df\_val = df(gamma, region, N)

# 중간 결과 출력

print(f"Iteration {iter\_count + 1}: gamma = {gamma:.10f}, f(gamma) = {f\_val:.10f}, df(gamma) = {df\_val:.10f}")

if np.isnan(f\_val) or np.isnan(df\_val) or df\_val == 0:

print("Newton's method 실패: f\_val 또는 df\_val에 문제가 발생했습니다.")

return gamma\_old

gamma = gamma - f\_val / df\_val

# 범위를 gamma\_min과 gamma\_max로 제한

gamma = min(gamma\_max, max(gamma\_min, gamma))

# 새로운 ea 계산

ea = abs(gamma - gamma\_old)

print(f"Updated gamma = {gamma:.10f}, ea = {ea:.10f}")

except Exception as e:

print(f"오류 발생: {e}")

return gamma\_old

# 다음 iteration

return newton\_method(gamma, region, sigma, N, gamma\_min, gamma\_max, es, ea, iter\_count + 1)

**2.3 Fusion of corrected images**

def calculate\_weight(bright\_region, sigma\_w=0.5):

return np.exp(-(bright\_region \*\* 2) / (2 \* sigma\_w \*\* 2))

def fusion(Y\_normalised, gamma\_dark, gamma\_bright, weight):

dark\_result = Y\_normalised \*\* gamma\_dark

bright\_result = Y\_normalised \*\* gamma\_bright

# 합성, 결과는 원본 이미지와 동일한 크기 (3024, 4032)

Y\_o = weight \* dark\_result + (1 - weight) \* bright\_result

Y\_o\_scaled = np.clip(Y\_o \* 255, 0, 255).astype(np.uint8) #0~1 인 Y\_o를 0~255로

return Y\_o\_scaled

**2.4 Adaptive color restoration**

def color\_restoration(Y\_o, Y\_channel, input\_R, input\_G, input\_B, bright\_region):

# 출력 결과 배열 초기화

output\_R = np.zeros\_like(input\_R, dtype=np.float64)

output\_G = np.zeros\_like(input\_G, dtype=np.float64)

output\_B = np.zeros\_like(input\_B, dtype=np.float64)

# s 계산

s = 1 - np.tanh(bright\_region)

epsilon = 1e-5 # 작은 값을 더해 0으로 나누는 경우를 방지

# 개별 채널별 출력 계산

output\_R = Y\_o \* ((input\_R / (Y\_channel + epsilon)) \*\* s)

output\_G = Y\_o \* ((input\_G / (Y\_channel + epsilon)) \*\* s)

output\_B = Y\_o \* ((input\_B / (Y\_channel + epsilon)) \*\* s)

output\_R = np.clip(output\_R, 0, 255)

output\_G = np.clip(output\_G, 0, 255)

output\_B = np.clip(output\_B, 0, 255)

output\_image = np.stack([output\_R, output\_G, output\_B], axis=-1).astype(np.uint8)

return output\_image

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명스크린샷, 텍스트, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

바퀴, 자전거 바퀴, 육상 차량, 차량이(가) 표시된 사진

자동 생성된 설명스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 도표, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 자전거, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 자전거, 소프트웨어이(가) 표시된 사진

자동 생성된 설명바퀴, 자전거 바퀴, 육상 차량, 차량이(가) 표시된 사진

자동 생성된 설명