**BLM6105-2 Uzaktan Algılama Dersi Ödev 5**

**24501100 - Aleyna Nil Uzunoğlu**

**Giriş**

Bu çalışmada, scikit-image kütüphanesi ve temel NumPy fonksiyonları kullanılarak görüntüler üzerinde temel histogram tabanlı kontrast iyileştirme yöntemleri incelenmiştir. Özellikle iki ana teknik ele alınmıştır: ödevin a bölümü için Histogram Eşitleme (Histogram Equalization) ve b bölümü için Histogram Eşleştirme (Histogram Matching/Specification).

Histogram eşitleme, görüntünün kontrastını otomatik olarak artırmak için parlaklık değerlerinin dağılımını daha homojen hale getirmeyi amaçlar. Histogram eşleştirme ise, bir kaynak görüntünün histogramını, hedef (referans) bir görüntünün histogramına benzeterek daha kontrollü bir kontrast ayarı sağlar. Bu teknikler, düşük kontrastlı görüntüleri görsel analiz veya sonraki adımlar (sınıflandırma, nesne tespiti vb.) için hazırlamada kullanılan yaygın ön işleme adımlarıdır.

**Yöntem**

1. **Veri** – scikit-image.data modülünden alınan üç adet standart gri seviye test görüntüsü kullanıldı (moon, camera, text). Görüntüler 8-bit (0-255) formatına dönüştürüldü.
2. **Araçlar** – Python 3, NumPy (dizi işlemleri, histogram, cdf), Matplotlib (görselleştirme) ve Scikit-image (veri yükleme, yardımcı fonksiyonlar img\_as\_ubyte) kütüphaneleri kullanıldı.
3. **Adımlar**
   * **Histogram (256 Kutu)**: Görüntüdeki her piksel değeri için numpy.bincount kullanılarak verimli bir şekilde frekans sayımı yapıldı.
   * **Kümülatif Histogram (CDF)**: Normal histogram dizisi üzerinde numpy.cumsum() fonksiyonu uygulanarak kümülatif dağılım fonksiyonu elde edildi.
   * **Histogram Eşitleme (5a)***:* Kendi yazdığım custom\_histogram\_equalization fonksiyonu kullanıldı. Bu fonksiyon, CDF'yi normalize ederek (0-255 aralığına) bir dönüşüm fonksiyonu (Look-Up Table - LUT) oluşturur ve bu LUT'u orijinal görüntüye uygulayarak piksellerin gri seviyelerini yeniden dağıtır. Maskeleme (np.ma.masked\_equal) ile CDF'deki sıfır değerlerinin neden olabileceği hatalar önlendi.
   * **Histogram Eşleştirme (5b)** Kendi yazdığım custom\_histogram\_matching fonksiyonu kullanıldı. Kaynak ve referans görüntülerin normalize edilmiş CDF'leri hesaplandı. Kaynak görüntünün her gri seviyesi için, kaynak CDF değerine en yakın (eşit veya büyük) CDF değerine sahip referans gri seviyesini bulan bir LUT oluşturuldu. Bu LUT, kaynak görüntüye uygulanarak histogramı referansa benzetildi.
4. **Çıktılar** – Her işlem adımı için görsel çıktılar üretildi ve PNG formatında kaydedildi:
   * *Eşitleme:* Her görüntü için, orijinal ve histogramı eşitlenmiş hallerini, histogramlarını ve CDF'lerini içeren karşılaştırmalı bir grafik (plot\_comparison fonksiyonu ile).
   * *Eşleştirme:* Belirlenen görüntü çiftleri için, kaynak ve histogramı eşleştirilmiş hallerini, histogramlarını ve CDF'lerini içeren karşılaştırmalı bir grafik (plot\_comparison fonksiyonu ile). Ayrıca eşleştirmede kullanılan referans görüntüler de ayrı olarak kaydedildi.

Ek: main.py

import numpy as np

import matplotlib.pyplot as plt

from skimage import io, color, exposure, data

from skimage.util import img\_as\_ubyte

import os

import re

def sanitize\_filename(filename):

filename = filename.strip()

filename = re.sub(r'[\\/\*?:"<>|().\s]+', '\_', filename)

filename = re.sub(r'\_+', '\_', filename)

filename = filename.strip('\_')

if not filename:

filename = "unnamed\_plot"

return filename

def calculate\_histogram(image\_gray\_uint8):

flat\_image = image\_gray\_uint8.ravel()

hist = np.bincount(flat\_image, minlength=256)

return hist

def calculate\_cdf(hist):

cdf = hist.cumsum()

return cdf

def plot\_comparison(original\_img, original\_title, transformed\_img, transformed\_title, outdir):

original\_img\_uint8 = img\_as\_ubyte(original\_img)

transformed\_img\_uint8 = img\_as\_ubyte(transformed\_img)

original\_hist = calculate\_histogram(original\_img\_uint8)

transformed\_hist = calculate\_histogram(transformed\_img\_uint8)

original\_cdf = calculate\_cdf(original\_hist)

if original\_cdf.max() > 0 and original\_hist.max() > 0:

original\_cdf\_normalized = original\_cdf \* original\_hist.max() / original\_cdf.max()

else:

original\_cdf\_normalized = original\_cdf

transformed\_cdf = calculate\_cdf(transformed\_hist)

if transformed\_cdf.max() > 0 and transformed\_hist.max() > 0:

transformed\_cdf\_normalized = transformed\_cdf \* transformed\_hist.max() / transformed\_cdf.max()

else:

transformed\_cdf\_normalized = transformed\_cdf

fig, axes = plt.subplots(2, 2, figsize=(12, 8))

ax = axes.ravel()

ax[0].imshow(original\_img, cmap=plt.cm.gray)

ax[0].set\_title(f"Orijinal: {original\_title}")

ax[0].axis('off')

ax[1].imshow(transformed\_img, cmap=plt.cm.gray)

ax[1].set\_title(f"İşlenmiş: {transformed\_title}")

ax[1].axis('off')

ax[2].plot(original\_hist, color='b', label='Histogram')

ax[2].plot(original\_cdf\_normalized, color='r', linestyle='--', label='CDF (Ölçekli)')

ax[2].set\_title(f"Orijinal Histogram & CDF")

ax[2].set\_xlim([0, 256])

ax[2].legend()

ax[3].plot(transformed\_hist, color='b', label='Histogram')

ax[3].plot(transformed\_cdf\_normalized, color='r', linestyle='--', label='CDF (Ölçekli)')

ax[3].set\_title(f"İşlenmiş Histogram & CDF")

ax[3].set\_xlim([0, 256])

ax[3].legend()

fig.tight\_layout()

safe\_orig\_title = sanitize\_filename(original\_title)

safe\_trans\_title = sanitize\_filename(transformed\_title)

save\_filename\_base = f"comparison\_{safe\_orig\_title}\_vs\_{safe\_trans\_title}.png"

save\_filepath = os.path.join(outdir, save\_filename\_base)

try:

print(f"-> Grafik kaydediliyor: {save\_filepath}")

plt.savefig(save\_filepath, dpi=200)

except Exception as e:

print(f"HATA: Grafik kaydedilemedi '{save\_filepath}': {e}")

plt.show()

plt.close(fig)

def custom\_histogram\_equalization(image):

img\_uint8 = img\_as\_ubyte(image)

hist = calculate\_histogram(img\_uint8)

cdf = calculate\_cdf(hist)

cdf\_masked = np.ma.masked\_equal(cdf, 0)

if cdf\_masked.min() is np.ma.masked or cdf\_masked.max() is np.ma.masked or (cdf\_masked.max() - cdf\_masked.min()) == 0:

cdf\_final = np.zeros\_like(cdf, dtype='uint8')

else:

cdf\_normalized = (cdf\_masked - cdf\_masked.min()) \* 255 / (cdf\_masked.max() - cdf\_masked.min())

cdf\_final = np.ma.filled(cdf\_normalized, 0).astype('uint8')

equalized\_img\_uint8 = cdf\_final[img\_uint8]

return equalized\_img\_uint8

def custom\_histogram\_matching(source\_image, reference\_image):

source\_uint8 = img\_as\_ubyte(source\_image)

reference\_uint8 = img\_as\_ubyte(reference\_image)

source\_pixels = max(1, source\_uint8.size)

reference\_pixels = max(1, reference\_uint8.size)

source\_hist = calculate\_histogram(source\_uint8)

reference\_hist = calculate\_histogram(reference\_uint8)

source\_cdf = calculate\_cdf(source\_hist)

reference\_cdf = calculate\_cdf(reference\_hist)

source\_cdf\_norm = source\_cdf / source\_pixels

reference\_cdf\_norm = reference\_cdf / reference\_pixels

lut = np.zeros(256, dtype='uint8')

ref\_level\_idx = 0

for src\_level in range(256):

while ref\_level\_idx < 255 and reference\_cdf\_norm[ref\_level\_idx] < source\_cdf\_norm[src\_level]:

ref\_level\_idx += 1

lut[src\_level] = ref\_level\_idx

matched\_img\_uint8 = lut[source\_uint8]

return matched\_img\_uint8

if \_\_name\_\_ == "\_\_main\_\_":

OUTDIR = "output\_odev5"

print("Odev 5 - Remote Sensing\n")

print("Yazar: Aleyna Nil Uzunoğlu\n")

print("Histogram Eşitleme ve Eşleştirme Uygulaması\n")

try:

os.makedirs(OUTDIR, exist\_ok=True)

print(f"Çıktı klasörü '{OUTDIR}' hazırlandı veya zaten mevcut.")

except OSError as e:

print(f"HATA: Çıktı klasörü '{OUTDIR}' oluşturulamadı: {e}")

print("Skimage örnek görüntüleri kullanılıyor...")

img1 = data.moon()

img2 = data.camera()

img3 = data.text()

images\_to\_process = [img\_as\_ubyte(img1), img\_as\_ubyte(img2), img\_as\_ubyte(img3)]

image\_titles = ['Moon\_Ornek', 'Camera\_Ornek', 'Text\_Ornek']

print("Örnek görüntüler yüklendi.\n")

# a

print("--- 5-a: Histogram Eşitleme ---")

equalized\_images = []

for i, img in enumerate(images\_to\_process):

print(f"\* {image\_titles[i]} için histogram eşitleme yapılıyor...")

equalized\_img = custom\_histogram\_equalization(img)

equalized\_images.append(equalized\_img)

plot\_comparison(img, image\_titles[i],

equalized\_img, f"{image\_titles[i]}\_Esitlenmis", OUTDIR)

print("Histogram eşitleme tamamlandı.\n")

print("--- 5-b: Histogram Özelleştirme (Eşleştirme) ---")

if len(images\_to\_process) >= 2:

source\_idx, ref\_idx = 1, 0

print(f"\* {image\_titles[source\_idx]}'ı {image\_titles[ref\_idx]}'in histogramına eşleştirme...")

matched\_img = custom\_histogram\_matching(images\_to\_process[source\_idx], images\_to\_process[ref\_idx])

plot\_comparison(images\_to\_process[source\_idx], f"Kaynak\_{image\_titles[source\_idx]}",

matched\_img, f"Eslesmis\_{image\_titles[source\_idx]}\_Ref\_{image\_titles[ref\_idx]}", OUTDIR)

fig\_ref, ax\_ref = plt.subplots(figsize=(6,4))

ax\_ref.imshow(images\_to\_process[ref\_idx], cmap='gray')

ref\_title = f"Referans\_{image\_titles[ref\_idx]}"

ax\_ref.set\_title(ref\_title)

ax\_ref.axis('off')

safe\_ref\_title = sanitize\_filename(ref\_title)

save\_filename\_base = f"{safe\_ref\_title}.png"

save\_filepath = os.path.join(OUTDIR, save\_filename\_base)

try:

print(f"-> Referans grafik kaydediliyor: {save\_filepath}")

fig\_ref.savefig(save\_filepath, dpi=200)

except Exception as e:

print(f"HATA: Referans grafik kaydedilemedi '{save\_filepath}': {e}")

plt.show()

plt.close(fig\_ref)

if len(images\_to\_process) >= 3:

source\_idx, ref\_idx = 2, 1

print(f"\* {image\_titles[source\_idx]}'i {image\_titles[ref\_idx]}'in histogramına eşleştirme...")

matched\_img = custom\_histogram\_matching(images\_to\_process[source\_idx], images\_to\_process[ref\_idx])

plot\_comparison(images\_to\_process[source\_idx], f"Kaynak\_{image\_titles[source\_idx]}",

matched\_img, f"Eslesmis\_{image\_titles[source\_idx]}\_Ref\_{image\_titles[ref\_idx]}", OUTDIR)

fig\_ref, ax\_ref = plt.subplots(figsize=(6,4))

ax\_ref.imshow(images\_to\_process[ref\_idx], cmap='gray')

ref\_title = f"Referans\_{image\_titles[ref\_idx]}"

ax\_ref.set\_title(ref\_title)

ax\_ref.axis('off')

safe\_ref\_title = sanitize\_filename(ref\_title)

save\_filename\_base = f"{safe\_ref\_title}.png"

save\_filepath = os.path.join(OUTDIR, save\_filename\_base)

try:

print(f"-> Referans grafik kaydediliyor: {save\_filepath}")

fig\_ref.savefig(save\_filepath, dpi=200)

except Exception as e:

print(f"HATA: Referans grafik kaydedilemedi '{save\_filepath}': {e}")

plt.show()

plt.close(fig\_ref)

elif len(images\_to\_process) == 2:

source\_idx, ref\_idx = 1, 0

print("-> 3. görüntü olmadığı için ek eşleştirme yapılmıyor.")

pass

print("\nHistogram eşleştirme tamamlandı.")

print(f"Ödev 5 tamamlandı. Çıktılar '{OUTDIR}' klasörüne kaydedildi.")

Ek: Çıktılar

1. Histogram Eşitleme

metin, giyim, ekran görüntüsü, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 1) 1.görüntü için histogram eşitleme

metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 2) 2.görüntü için histogram eşitleme

diyagram, çizgi, ekran görüntüsü, metin içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 3) 3.görüntü için histogram eşitleme

1. Histogram Uydurma

metin, diyagram, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Şekil 4) Camera ve Moon görüntüleri için bir örnek

ay, doğa, Astronomi nesnesi, krater içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.Şekil 5) Referans Moon görüntüsü