

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 görüntülerini 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 önlandı.
 - **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. **Cıktılar** – 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 eşleştirme medde 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 görüntülerini 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]}\'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)

```

```

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)

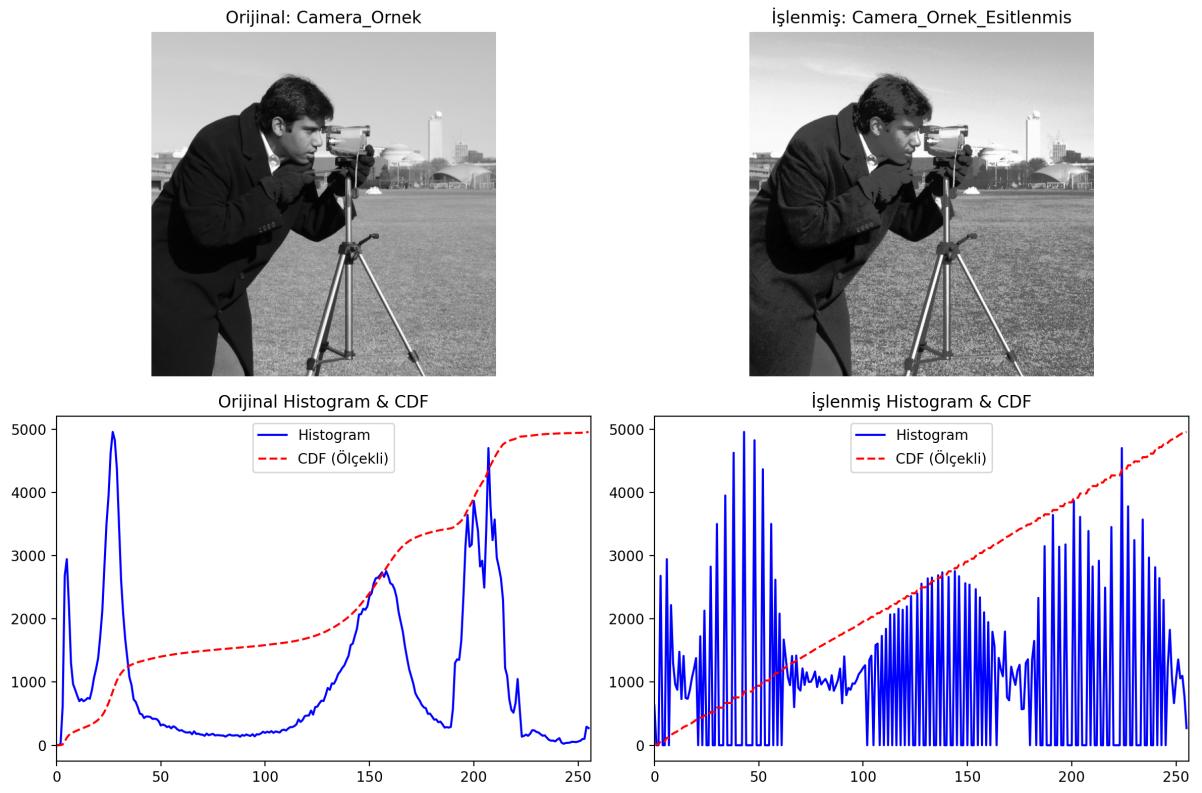
elif len(images_to_process) == 2:
    source_idx, ref_idx = 1, 0
    print("-> 3. görüntü olmadığı için ek eşleştirme yapılmıyor.")
    pass

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

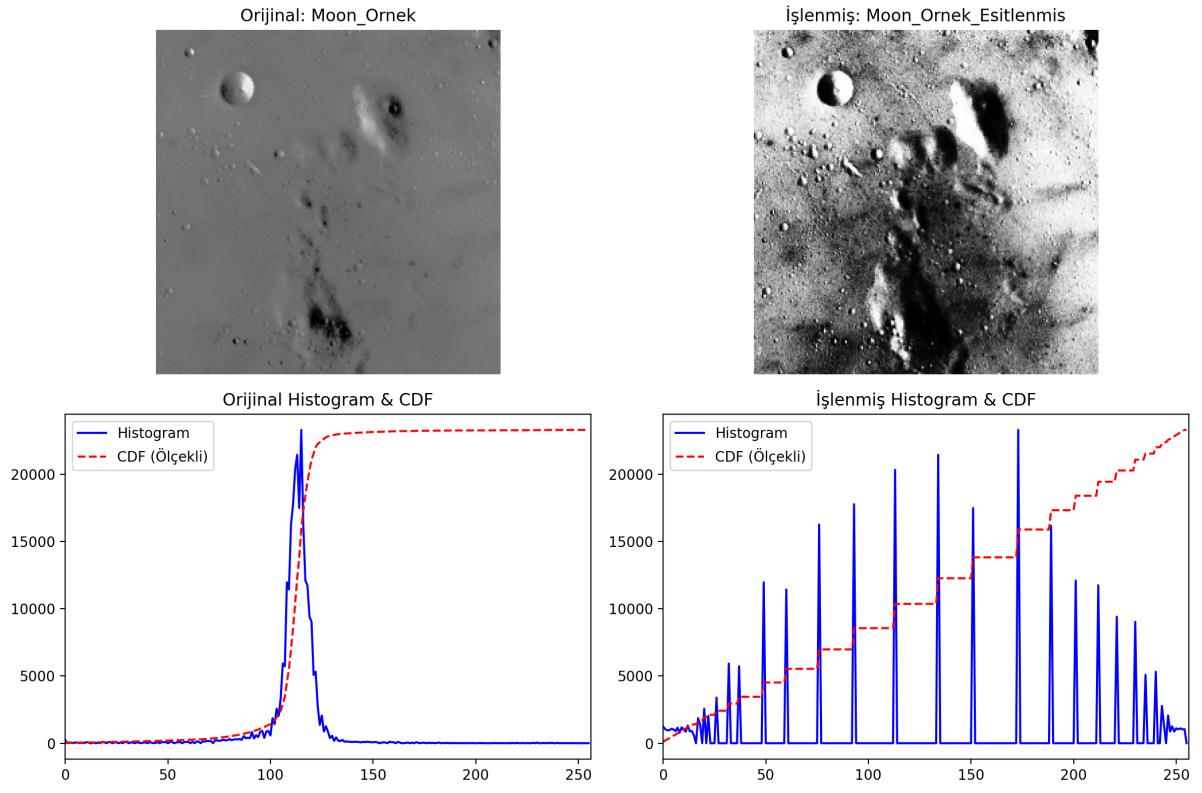
```

Ek: Çıktılar

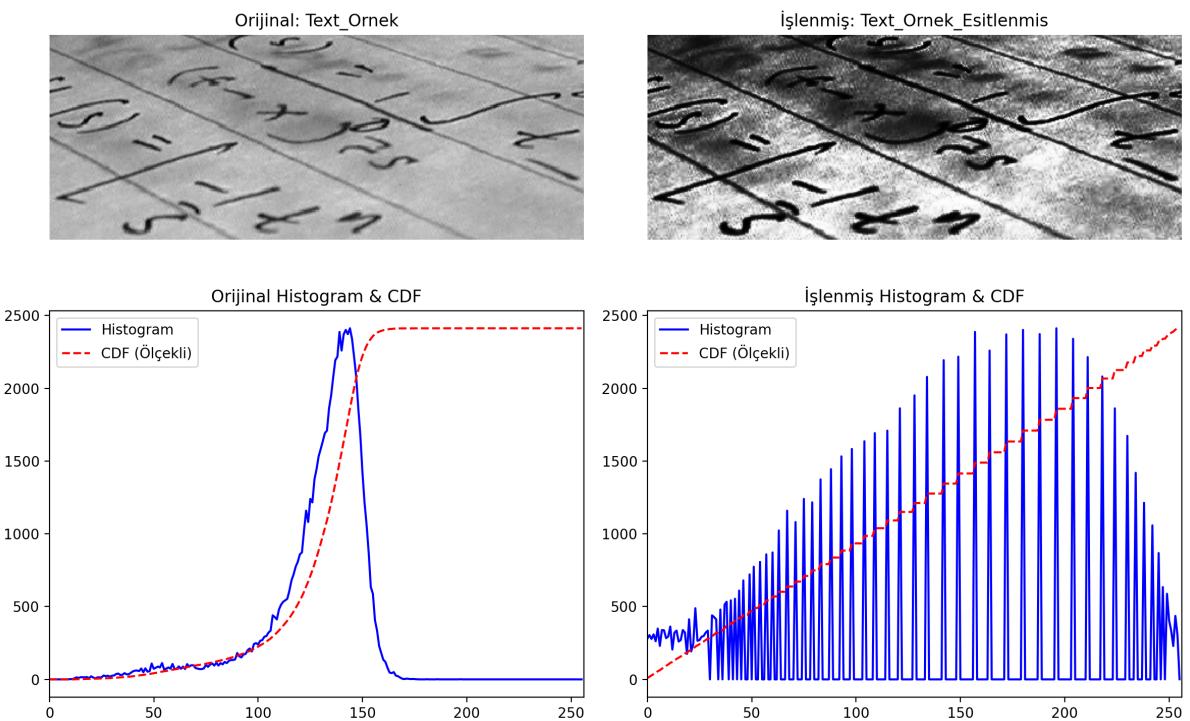
a) Histogram Eşitleme



Şekil 1) 1.görüntü için histogram eşitleme

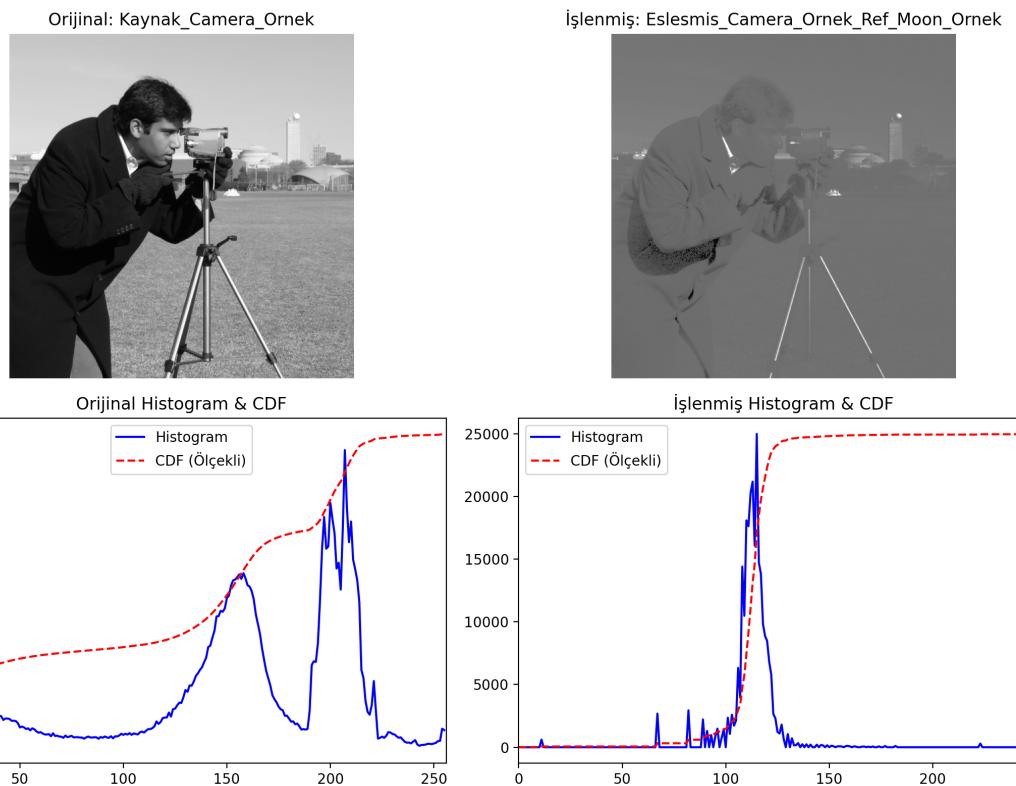


Şekil 2) 2.görüntü için histogram eşitleme



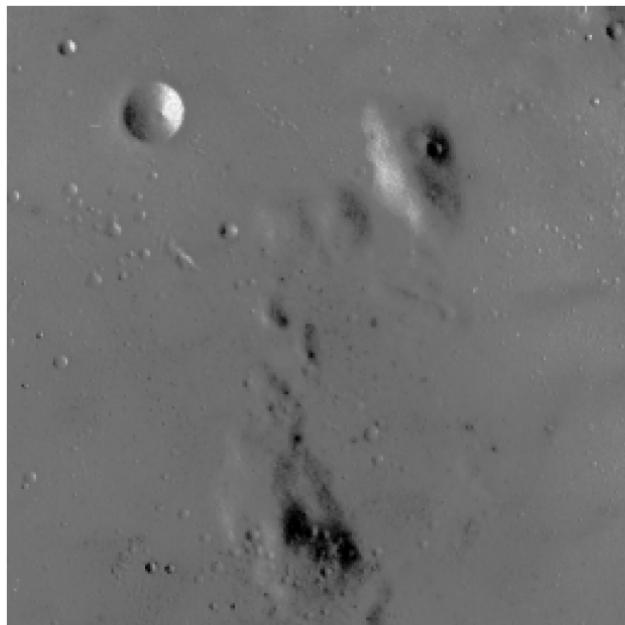
Şekil 3) 3.görüntü için histogram eşitleme

b) Histogram Uydurma



Şekil 4) Camera ve Moon görüntülerini için bir örnek

Referans_Moon_Ornek



Şekil 5) Referans Moon görüntüsü