Bartın Üniversitesi

Mühendislik, Mimarlık ve Tasarım Fakültesi

Bilgisayar Mühendisliği

BSM423 Görüntü İşlemenin Temelleri

Dönem Projesi Raporu

PROJE KONUSU

Uzay Fotoğrafından Yıldız Adeti Tespiti

GRUP NO

23

ÖĞRENCİ

Mehmet Oğuz Ekşi

18010310034

TARİH

04/01/2022

Proje Konusu Anlatımı

Konumuz “Uzay Fotoğrafından Yıldız Adeti Tespiti”. Uzay fotoğrafları günümüzde sıklıkla çekilen görüntülerdir. Görüntüler içerisinde bir çok yıldız nokta halinde bulunmaktadır. Bu yıldızları normal bir insanın sayması uzun sürecektir. Dolayısıyla bu işlemi görüntü işleme teknikleri kullanarak yazılım halinde bilgisayara saydıracağız.

Amaç ve Hedefler

Temel amaç görüntü işleme metotlarını kullanarak, fotoğrafın içerisinde bulunan yıldız adetini yaklaşık bir sayıda saymaktır. Fakat her çekilen uzay fotoğrafı aynı değildir.

Bir çok farklı şekilde çekilmiş fotoğraflar mevcuttur. Bunun için fotoğraflar analiz edilebilmesi için temel gereksinim belirlenmesi lazım.

Temel Gereksinimler:

1. Fotoğrafta yıldızların parlaklığından daha fazla bir obje bulunmaması lazımdır.
2. Fotoğrafta bulunan yıldızların piksel boyutundan daha büyük bir boyutta yer kaplaması lazım. Yani, Görüntü kalitesi günümüzde ki ortalama kaliteden düşük olmaması lazım.
3. Işık kaynağı sadece gökyüzü olması lazım. Yıldızlardan hariç herhangi bir ışık kaynağı olmaması lazım.

Üstte belirtilen gereksinimler karşılandıktan sonra. Fotoğraflardan daha uygun bir sonuç çıkacaktır.

Yıldız adetinin hesaplanması için birden fazla farklı görüntü işleme metotunları kullanacaktır. Bunun sonucunda bir çalışma akışı oluşacaktır.

Temel Düşünce: Nokta sayımı gibi düşünüldü. Yıldızlar nokta halinde düşünelecek. Yıldızlar ayrı ayrı noktalara ayrılacak. İşlemler bittikten sonra yıldızlar nokta haline dönüşecek, nokta sayımı yapılır gibi sayılacak.

Çalışma Akışı:

1. Gri Fotoğrafa Dönüştürme: Fotoğraf Gri renk tonlamasına tek kanal fotoğrafa dönüştürelecektir.
2. Gamma ve Contrast: İşlemleri uygulayarak resmin ortalama parlaklığı azaltılacaktır. Açık ve koyu renkler arasında ki fark açılacaktır. Yıldızlar dahada belli olacak ve birbirinden ayrılacaktır.
3. ParlaklıkTemizleme: Fotoğraf içerisinde değerlerin iki tona ayrılmış olduğu kesinleştirecektir. Beyaz veya siyah olmak üzere iki ton. (255 ve 0) tonları.
4. OpenCV işlemi Threshold: Threshold işlemi uygulanarak ortaya çıkmış yıldızların köşe (Edge) değerleri çizilecektir. Ortaya çıkan köşelere göre yıldızlar ortaya çıkacaktır.
5. Dilation ve Erosion: Köşe diğer köşe değerlerinden uzak kalabilmesi için, köşe değerleri dahada kesilerek nokta değerliğine yaklaştıracaktır, ortaya çıkan yıldızı.
6. Label ve Count: Ortaya çıkan yıldızlar, labeling yöntemi ile etiketleştirilecek. Etikleştirilmiş yıldızlara renkler verilerek bölümler ortaya çıkarılacak. Bu sayede çıkan resmin yıldız yoğunluğunun nerede olduğu vb. gibi diğer bilgiler ortaya çıkacak.
   Ayrıca etiketlenirken yıldız adetide hesaplanacaktır.

Kullanılan Teknolojiler

Programlama Dili: Python 3.10.0

Kodlama Yapılan Ortam: Jupyter Notebook, Visual Studio Code

Görüntü İşleme Kütüphanesi: OpenCV2

Matematik Kütüphanesi: Numpy

Görüntü Çıktısı Alabilmek İçin: Matplot Kütüphanesi

Projenin Çalışması (KOD Anlatımı)

Çalıştırılması ve Ek Fonksiyon:

A. Resim OpenCV2 kütüphanesi ile okutulacak. İşlemler sırasıyla yapılarak, tüm fonksiyonlara image gönderilecek ve tekrardan işlem yapılmış hali alınacak.

```
original = cv2.imread('../Veri_Klasoru/86nokta.jpg')
#Run Order
image = ChangeToGrayScale(original)
image = GamaCorrection(image)
image = ParlaklikTemizleme(image)
image = Threshold(image)
image = DilationAndErosion(image)
image = LabelImage(image)
```

B. Resimlerin her işlem sonucu gösterilmesi için bir fonksiyon oluşturuldu. Matplot kütüphanesi kullanılarak plot şeklinde çıktıyı, IDE üzerinde her işlem için vermektedir. Fonksiyon her görüntü işleme fonksiyonunda çağrılacaktır.

```
def Show_plot(inImage, inTitle):
    size = 16
    fig = plt.figure(figsize=(size, size))
    plt.title(inTitle)
    plt.imshow(inImage, cmap='gray'), plt.xticks([]), plt.yticks([])
```

İşlemler:

1. Resimi cv2 kütüphanesi ile Gray Scale e dönüştürüldü.

```python
def ChangeToGrayScale(inImage):
    inImage = cv2.cvtColor(inImage, cv2.COLOR_BGR2GRAY)
    Show_plot(inImage, 'Original Image')
    return inImage
```

2. Gamma işlemi yapılarak parlaklık azaltıldı.

```python
def GamaCorrection(inImage):
    y = 1.5
    gray_Gamma = np.array(255 * (inImage / 255) ** y , dtype='uint8')
    Show_plot(gray_Gamma, 'Gamma')
    return gray_Gamma
```

3. Parlaklık Temizleme yaparak tonlamalar iki tona ayrılıldı. Beyaz ve Siyah tonlar. 255 ve 0.

```python
def ParlaklikTemizleme(inImage):
    S = inImage.shape
    # 0 < MaxDotStrength < 255
    MaxDotStrength = 40

    for row in range(S[0]):
        for column in range(S[1]):
            pixel = inImage[row, column]
            if(pixel < MaxDotStrength):
                inImage[row, column] = 0
            else:
                inImage[row, column] = 255

    Show_plot(inImage, 'ParlaklikTemizleme')
    return inImage
```

4. OpenCV2 işlemi Threshold yaparak köşeler hesaplandı. Yıldızların köşeleri ortaya çıkarıldı.

```python
def Threshold(inImage):
    # blockSize = Filtre boyutu
    blockSize = 3
    # C = Derinlik
    C = 5
    thresh = cv2.adaptiveThreshold(inImage, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, blockSize, C)
    # Bit değerliğinde NOT işlemi. Siyah ve beyaz değişimi.
    thresh = cv2.bitwise_not(thresh)
    Show_plot(thresh, 'Adaptive Threshold')
    return thresh
```

5. Dilation ve Erosion işlemi yaparak, köşeler daha içeriye doğru, nokta haline gelecek şekilde azaltıldı, kesildi.

```python
def DilationAndErosion(inImage):
    filtersize = 3
    kernel = np.ones((filtersize, filtersize), np.uint8)
    img_dilation = cv2.dilate(inImage, kernel, iterations=1)
    img_erode = cv2.erode(img_dilation,kernel, iterations=1)

    Show_plot(img_dilation, "Dilation")
    Show_plot(img_erode, "Dilation + Erosion")
    return img_erode
```

6. OpenCV2 Label işlemi yapıldı. Köşelerden birbirine bağlanan şekiller etiketlendi. Etiketlenen şekiller farklı renklere ayrıldı ve çıktı olarak gösterildi.
   Etiketleme işlemi yapılırken sayım işlemide yapıldı. Yıldız adeti hesaplandı.

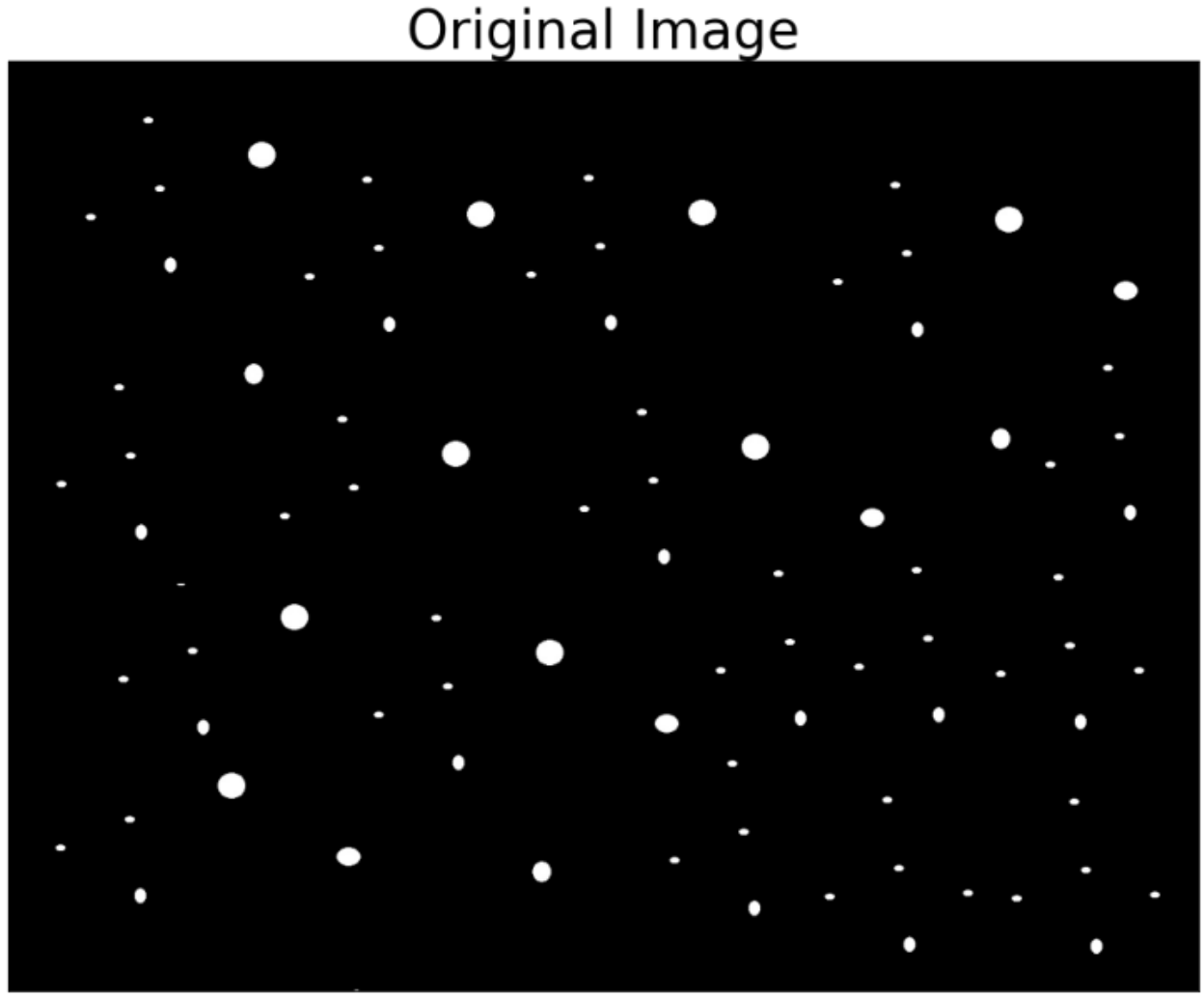
```python
def LabelImage(inImage):
    ret, labels = cv2.connectedComponents(inImage)
    label_hue = np.uint8(180 * labels / np.max(labels))
    blank_ch = 255 * np.ones_like(label_hue)
    labeled_img = cv2.merge([label_hue, blank_ch, blank_ch])
    labeled_img = cv2.cvtColor(labeled_img, cv2.COLOR_HSV2BGR)
    labeled_img[label_hue == 0] = 0

    print('Star Count is:', ret-1)

    Show_plot(labeled_img, 'Star counted:'+ str(ret-1))
    return labeled_img
```

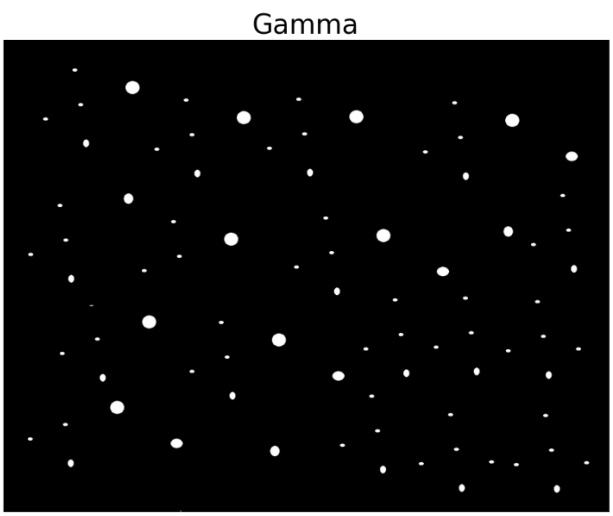
Sonuç

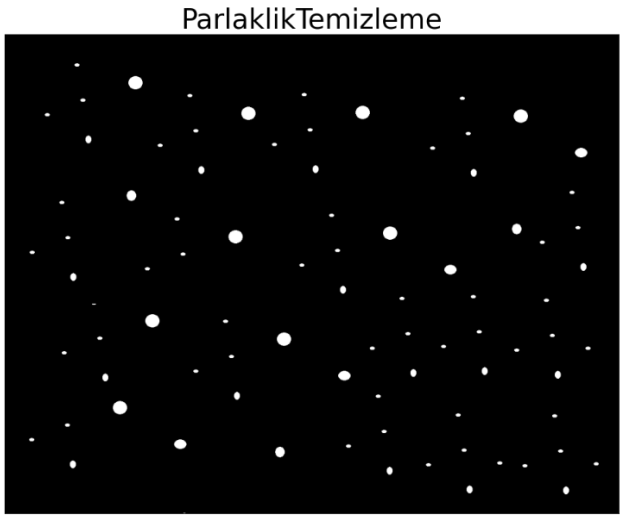
Görüntü işleme metotları kullanarak, bir uzay fotoğrafında bulunan yıldız adetini yaklaşık, hata payıyla birlikte saymaktadır.

Çalışma Örnekleri (Soldan Sağa Doğru)

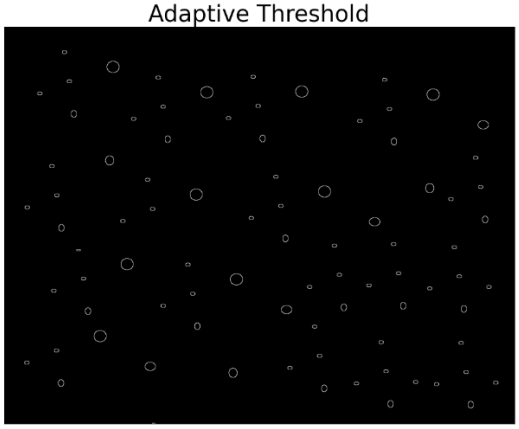
86 Stars (86 Nokta) Test Denemesi | Star Count: 86



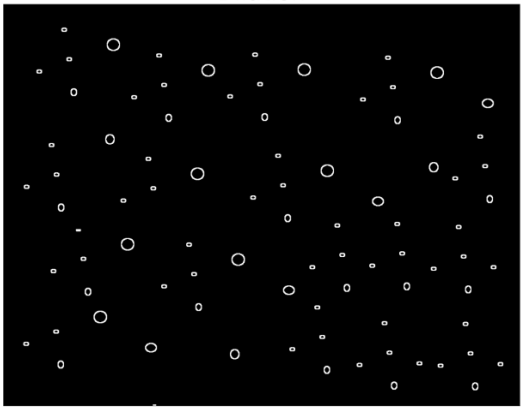




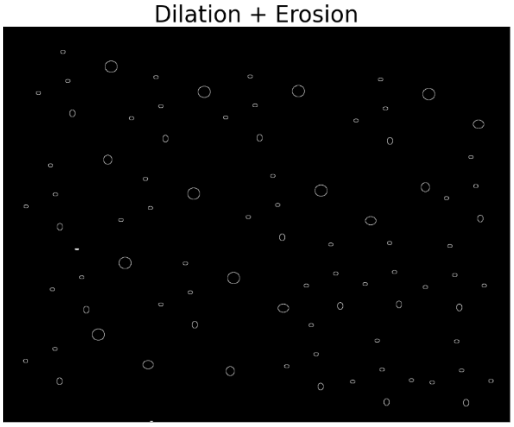
Adaptive Threshold



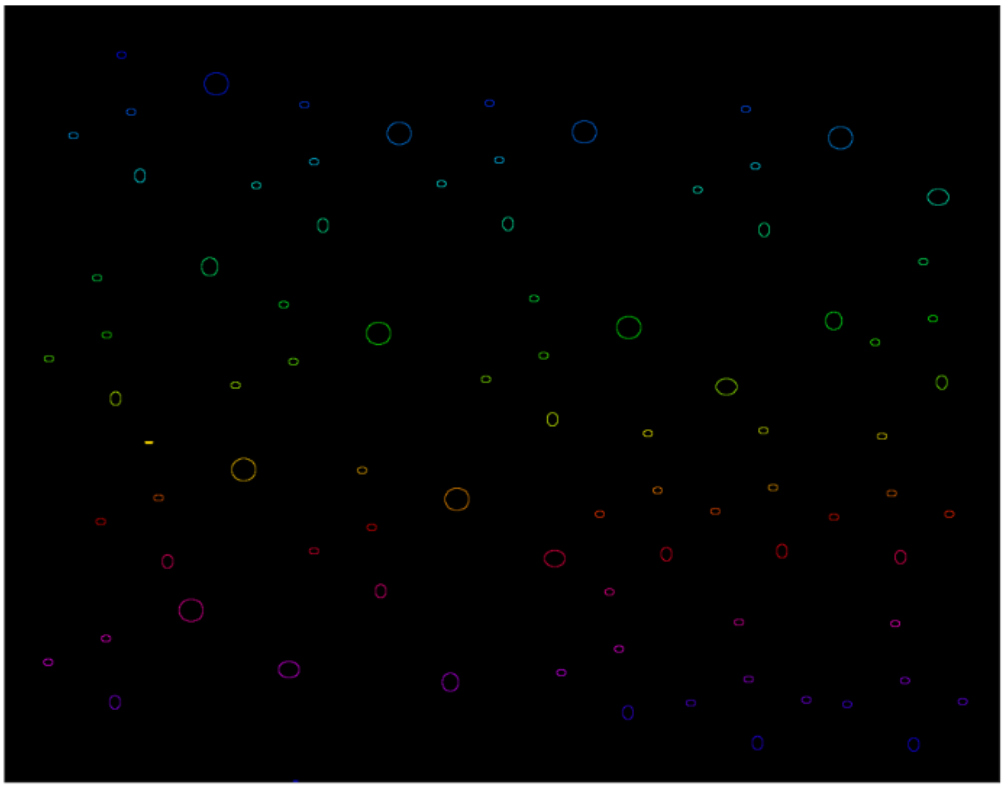
Dilation



Dilation + Erosion

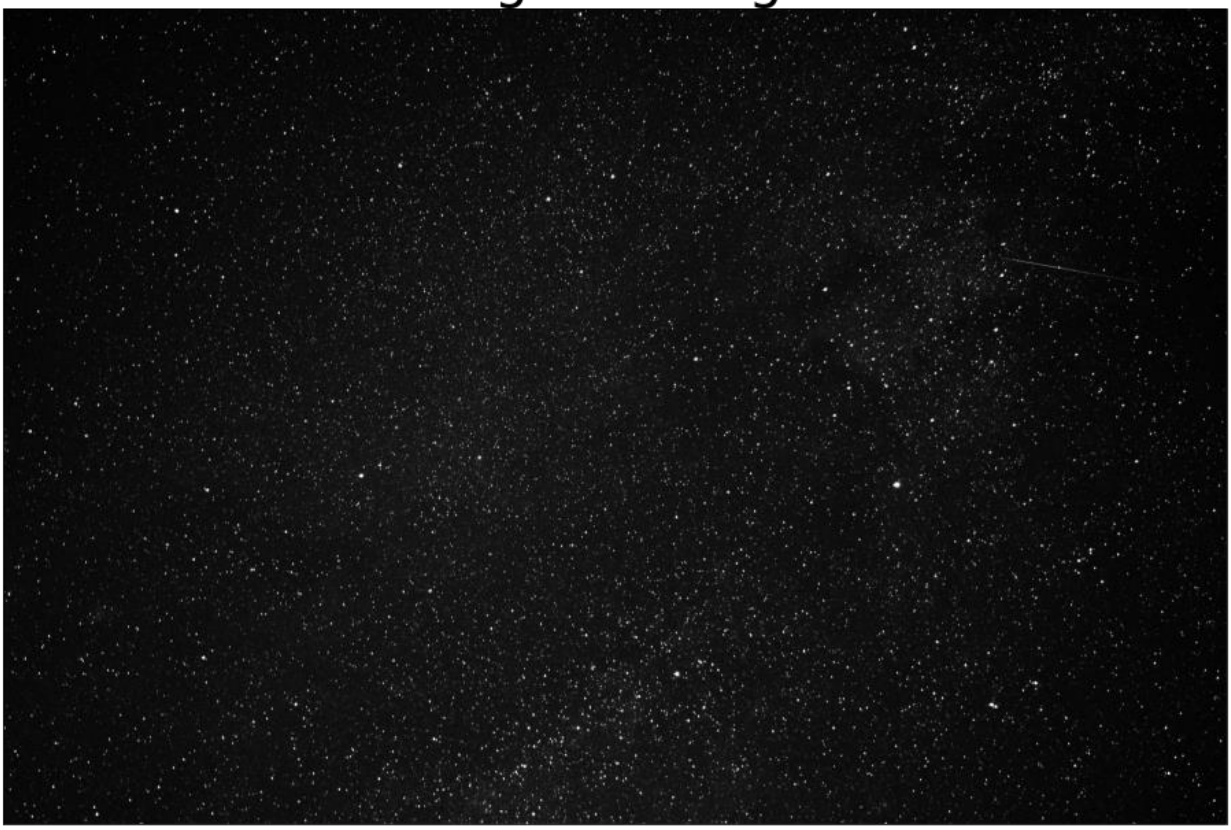


Star counted:86



Stars1 Image | Star Count: 15405

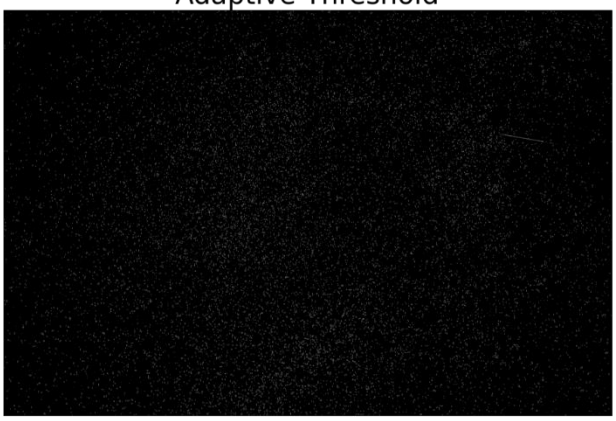
Original Image



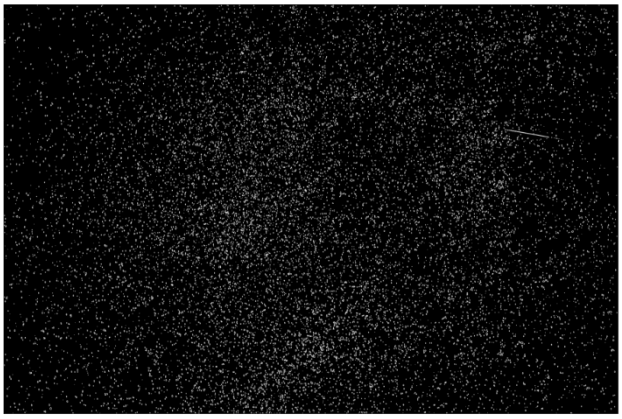
Gamma

ParlaklikTemizleme

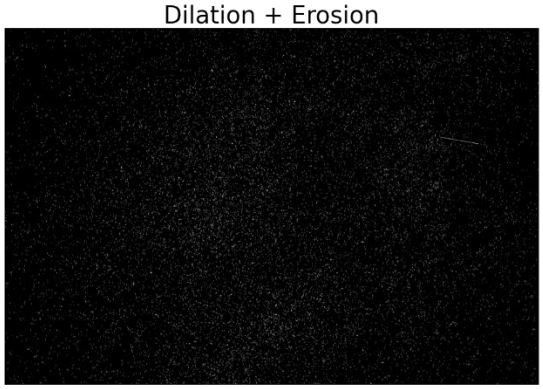
Adaptive Threshold



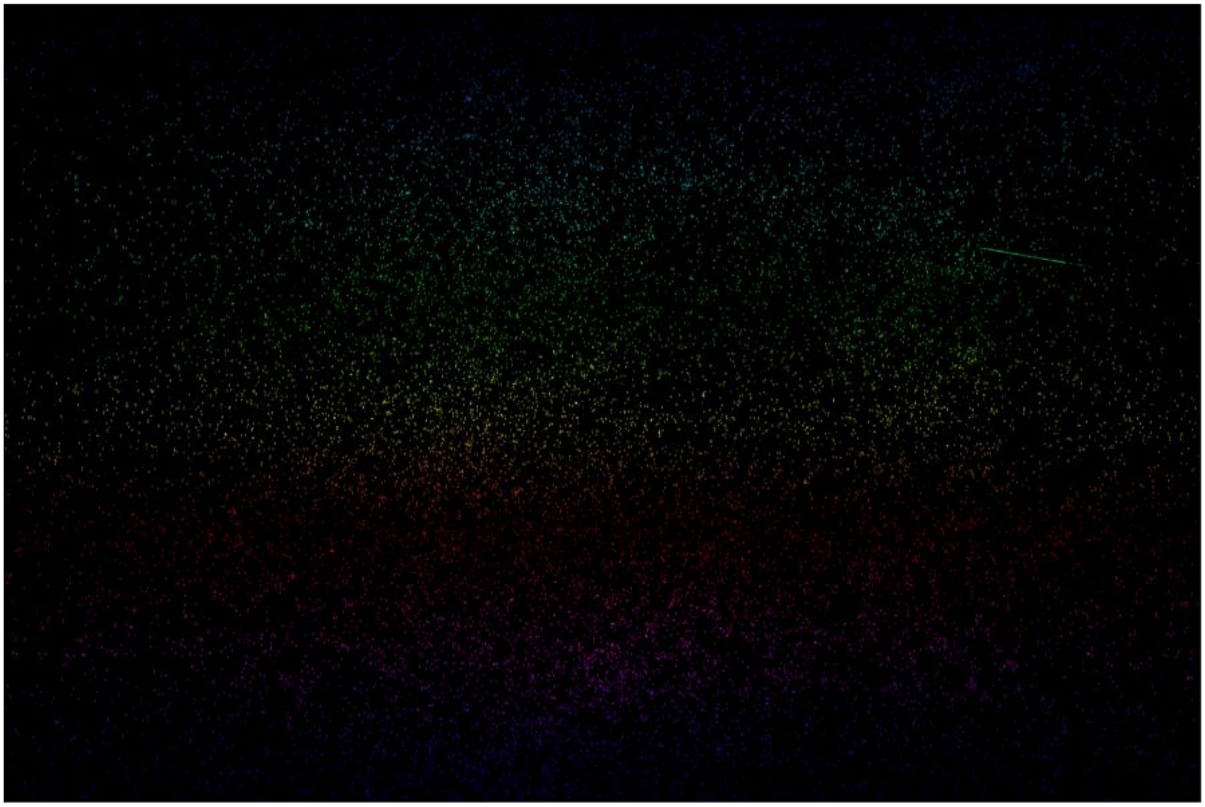
Dilation



Dilation + Erosion

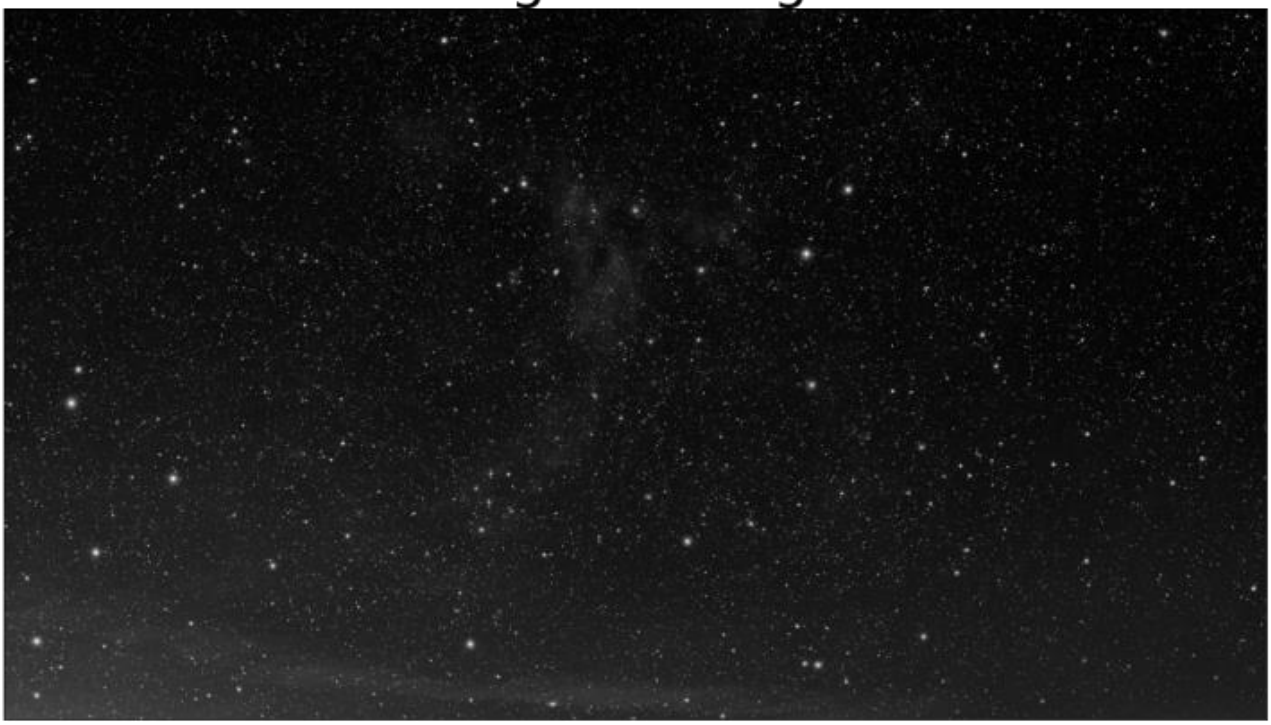


Star counted:15405



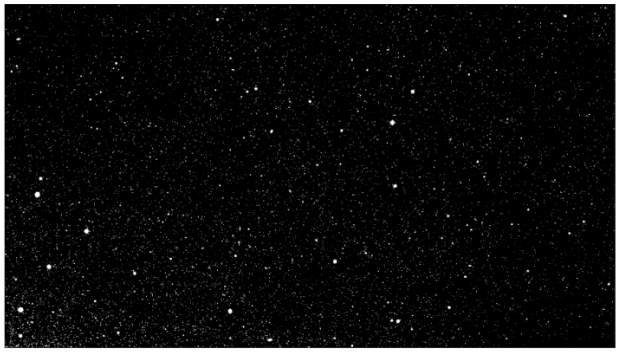
Stars2 Image | Star Count: 8572

Original Image

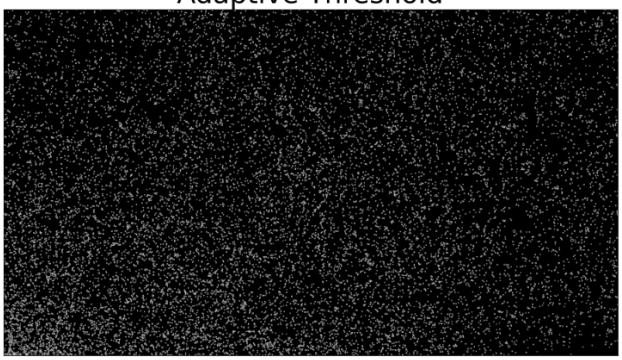


Gamma

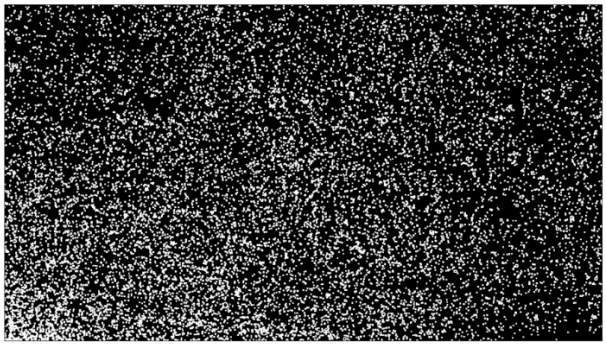
ParlaklikTemizleme



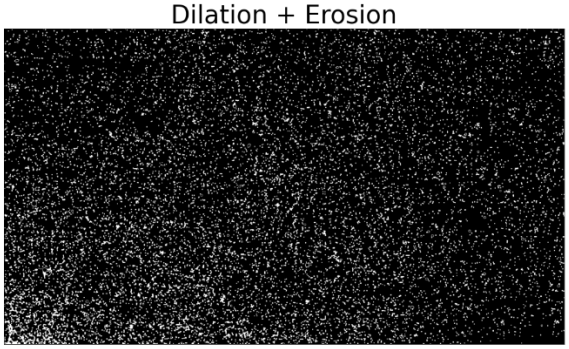
Adaptive Threshold



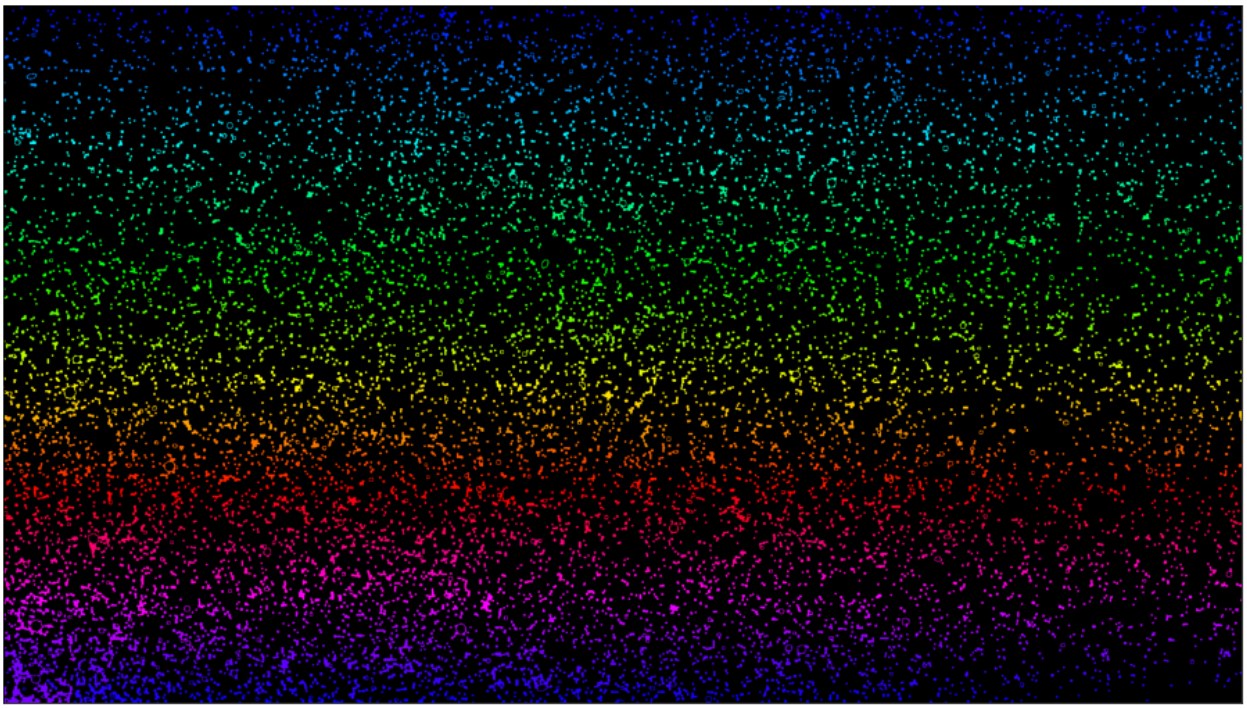
Dilation



Dilation + Erosion



Star counted:8572



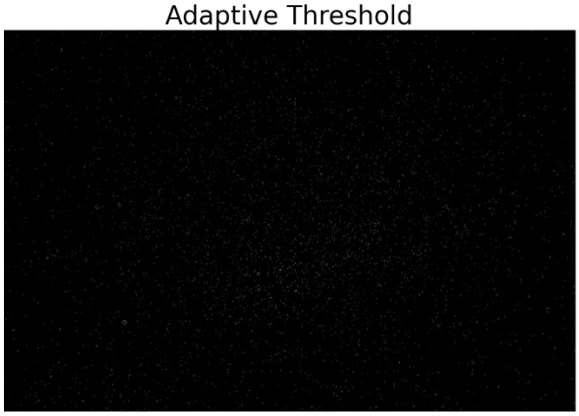
Stars3 Image | Star Count: 4477

Original Image

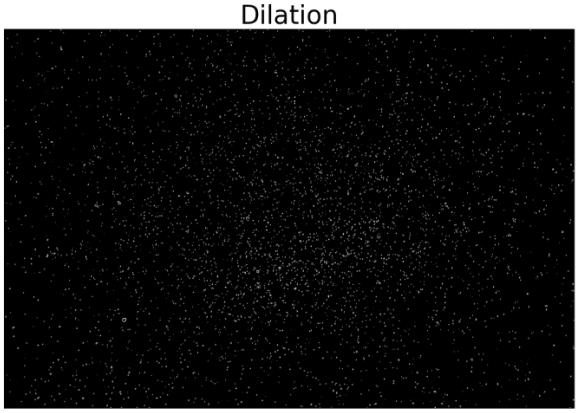
Gamma

ParlaklikTemizleme

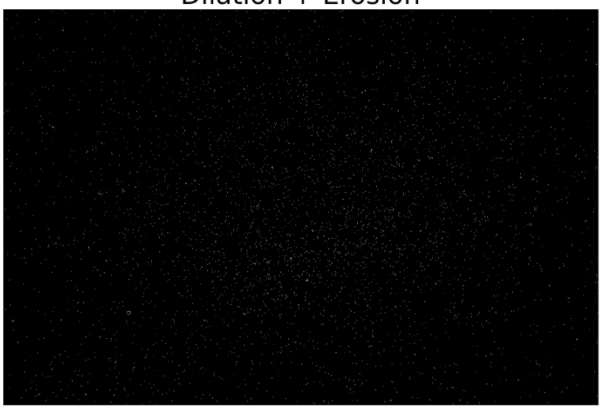
Adaptive Threshold



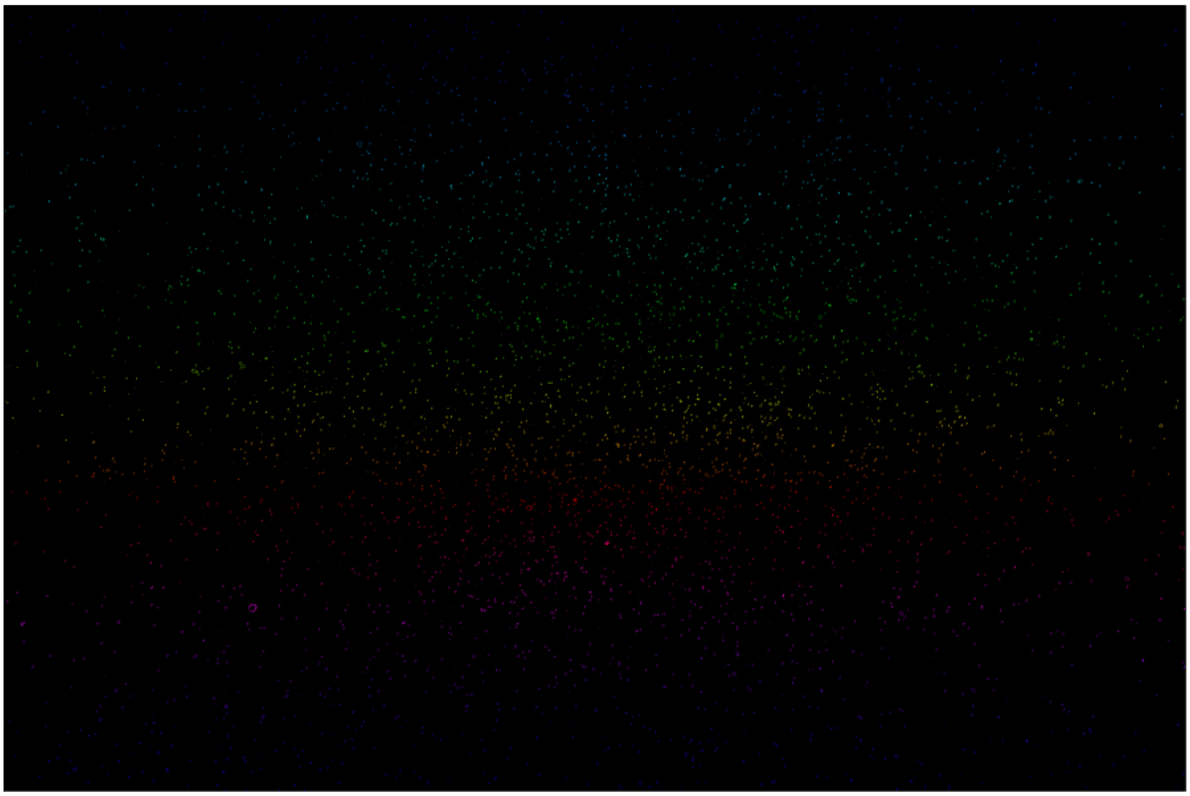
Dilation



Dilation + Erosion



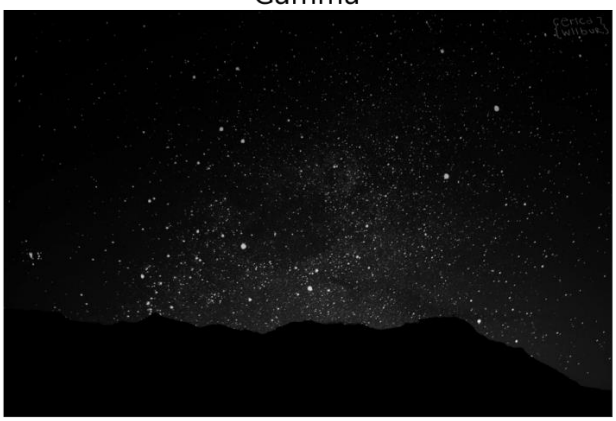
Star counted:4477



Stars4 Image | Star Count: 2544

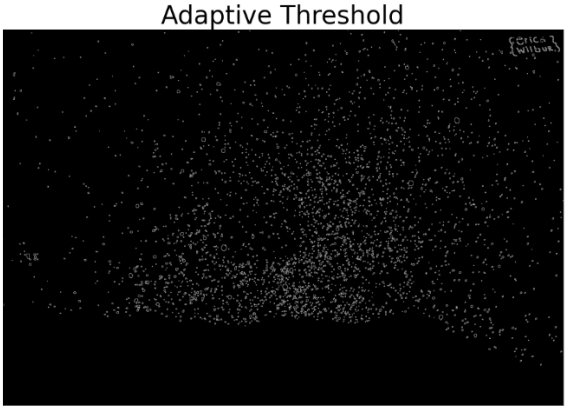
Original Image

Gamma

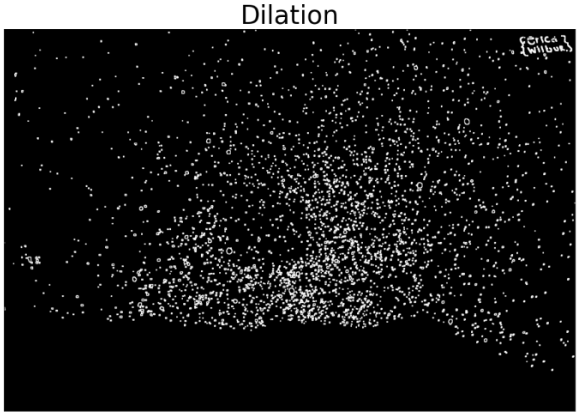


ParlaklikTemizleme

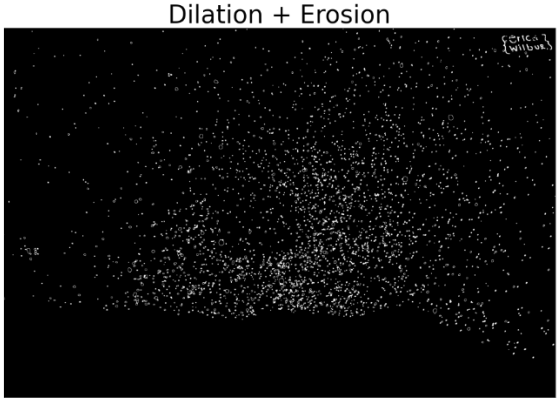
Adaptive Threshold



Dilation



Dilation + Erosion



Star counted:2544