KIRIKKALE ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BİL1006-BİLGİSAYAR GRAFİĞİNE GİRİŞ

Doç. Dr. Serkan SAVAŞ

FİLTRELEME İŞLEMLERİ

- Görüntü Eşikleme
  - Basit Eşikleme
  - Adaptif (Uyarlanabilir) Eşikleme
  - Otsu Eşikleme

Görüntü Eşikleme

Eşikleme, sağlanan eşik değerine göre piksel değerlerinin atanması gerçekleştirilen bir tekniktir. Eşiklemede her piksel değeri eşik değeri ile karşılaştırılır.

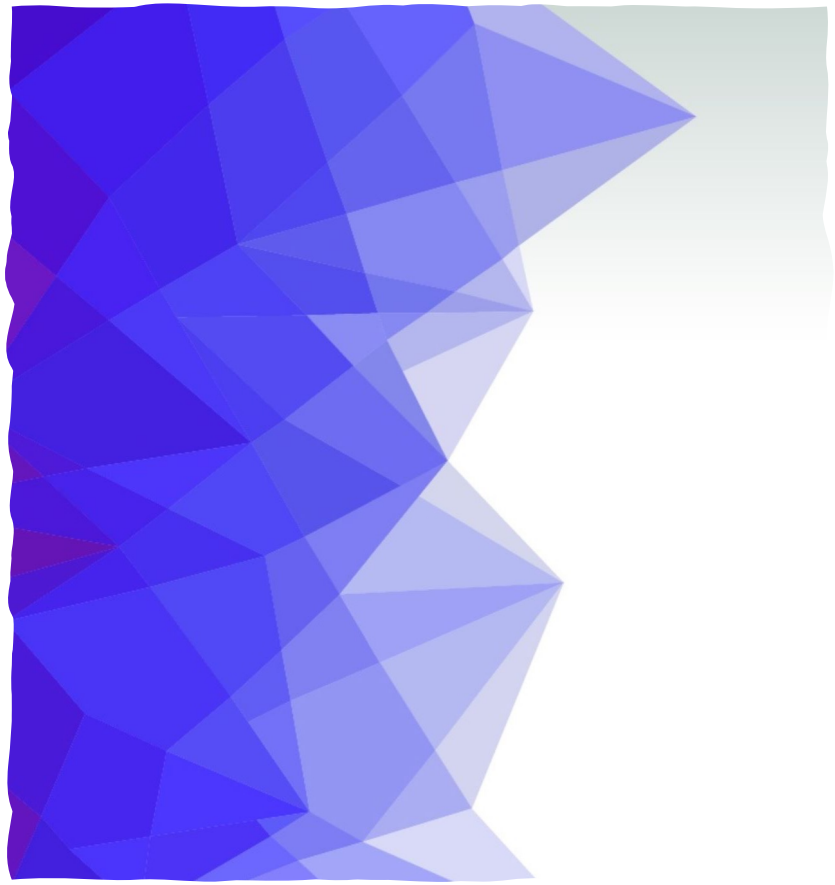
Piksel değeri eşikten küçükse 0, aksi takdirde maksimum değere (genellikle 255) ayarlanır.

Eşikleme, ön plan olarak kabul edilen bir nesneyi arka planından ayırmak için kullanılan çok popüler bir bölütleme tekniğidir.

Eşik, her iki tarafında, yani eşiğin altında veya üstünde iki bölge bulunan bir değerdir.

Computer Vision'da, bu eşikleme tekniği gri tonlamalı görüntüler üzerinde yapılır. Bu nedenle, başlangıçta görüntünün gri tonlamalı renk uzayına dönüştürülmesi gerekir.

Görüntü Eşikleme



cv2.threshold(kaynak, eşikDeğeri, maksDeğer, eşiklemeTekniği)

Parametreler:

***kaynak**: giriş görüntü dizisi (Gri tonlamalı).*

***eşikDeğeri**: Piksel değerlerinin buna göre değişeceği Alt ve Üst Eşik Değeri.*

***maksDeğer**: Bir piksele atanabilecek maksimum değer.*

***eşiklemeTekniği**: Uygulanacak eşikleme türü.*

Basit Eşikleme

Bir görüntüde herhangi bir piksel değeri bir eşik değerinden büyükse bir değer (beyaz olabilir), yoksa başka bir değer (siyah olabilir) atanır.

Bu işlem için;

cv2.threshold() fonksiyonu kullanılır.

- Bu fonksiyon içerisindeki ilk parametre, gri tonlamalı bir görüntü olması gereken kaynak görüntüdür.
- İkinci parametre, piksel değerlerini sınıflandırmak için kullanılan eşik değeridir.
- Üçüncü parametre, piksel değeri eşik değerinden büyük (bazen daha düşük) ise verilecek değeri temsil eden maxVal'dir.

Basit Eşikleme

OpenCV farklı eşikleme stilleri sağlar ve buna fonksiyonun dördüncü parametresi tarafından karar verilir.

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

Binary

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Inverted Binary

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$$

Truncated

$$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

To Zero

$$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

To Zero Inverted

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$$

Basit Eşikleme

cv2.THRESH_BINARY: Piksel yoğunluğu ayarlanan eşikten büyükse, değer 255'e, aksi takdirde 0'a (siyah) ayarlanır.

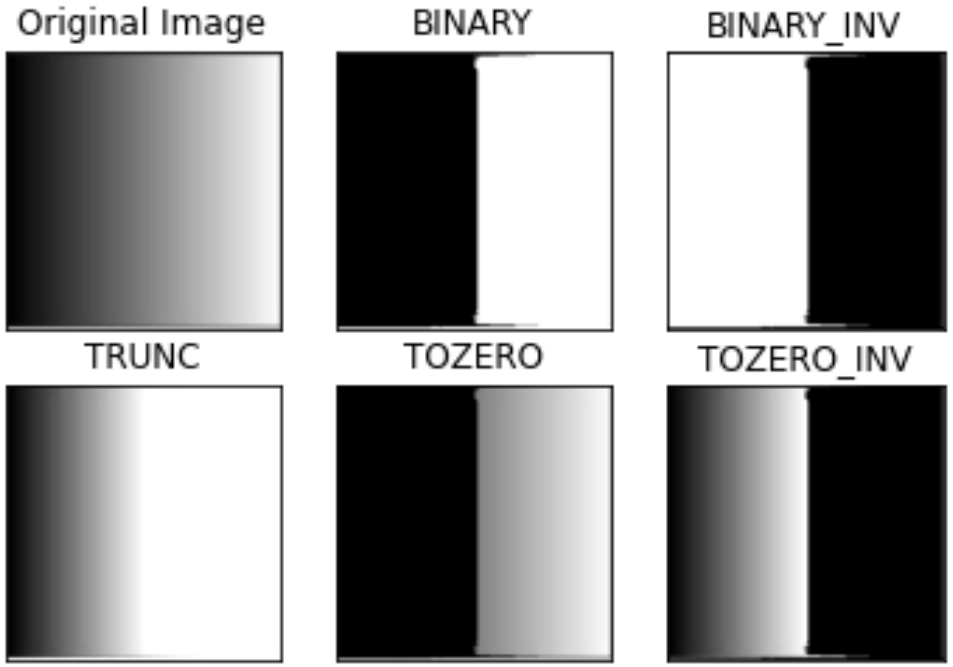
cv2.THRESH_BINARY_INV: cv2.THRESH_BINARY'nin Ters veya Zıt durumu.

cv.THRESH_TRUNC: Piksel yoğunluğu değeri eşikten büyükse eşik değerine kadar kısaltılır. Piksel değerleri, eşik ile aynı olacak şekilde ayarlanır. Diğer tüm değerler aynı kalır.

cv.THRESH_TOZERO: Piksel yoğunluğu, eşik değerinden daha az olan tüm piksel yoğunluğu için 0 olarak ayarlanır.

cv.THRESH_TOZERO_INV: cv2.THRESH_TOZERO'nun Ters veya Zıt durumu.

Basit Eşikleme




```python
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('gradient.png',0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)

titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

cv2.THRESH_BINARY

```
img = cv2.imread('high_way.jpg')
cv2_imshow(img)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray)
```

```
#THRESH_BINARY
(thresh, blackAndWhiteImage1) = cv2.threshold(gray, 20, 255, cv2.THRESH_BINARY)
(thresh, blackAndWhiteImage2) = cv2.threshold(gray, 80, 255, cv2.THRESH_BINARY)
(thresh, blackAndWhiteImage3) = cv2.threshold(gray, 160, 255, cv2.THRESH_BINARY)
(thresh, blackAndWhiteImage4) = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY)
images = [blackAndWhiteImage1,blackAndWhiteImage2,blackAndWhiteImage3,blackAndWhiteImage4]
for i in images:
  plt.imshow(i,'gray')
  plt.xticks([]),plt.yticks([])
  plt.show()
```

THRESH_BINARY_INV, TRESH_TRUNC, TRESH_TOZERO

```
#THRESH_BINARY_INV
(thresh, blackAndWhiteImage5) = cv2.threshold(gray, 20, 255, cv2.THRESH_BINARY_INV)
(thresh, blackAndWhiteImage6) = cv2.threshold(gray, 80, 255, cv2.THRESH_BINARY_INV)
(thresh, blackAndWhiteImage7) = cv2.threshold(gray, 160, 255, cv2.THRESH_BINARY_INV)
(thresh, blackAndWhiteImage8) = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY_INV)
binary_inv_images = [blackAndWhiteImage5,blackAndWhiteImage6,blackAndWhiteImage7,blackAndWhiteImage8]
for i in binary_inv_images:
  plt.imshow(i,'gray')
  plt.xticks([]),plt.yticks([])
  plt.show()
```

```
#THRESH_TRUNC
(thresh, blackAndWhiteImage9) = cv2.threshold(gray, 20, 255, cv2.THRESH_TRUNC)
(thresh, blackAndWhiteImage10) = cv2.threshold(gray, 80, 255, cv2.THRESH_TRUNC)
(thresh, blackAndWhiteImage11) = cv2.threshold(gray, 160, 255, cv2.THRESH_TRUNC)
(thresh, blackAndWhiteImage12) = cv2.threshold(gray, 200, 255, cv2.THRESH_TRUNC)
trunc_images = [blackAndWhiteImage9,blackAndWhiteImage10,blackAndWhiteImage11,blackAndWhiteImage12]
for i in trunc_images:
  plt.imshow(i,'gray')
  plt.xticks([]),plt.yticks([])
  plt.show()
```

```
#THRESH_TOZERO
(thresh, blackAndWhiteImage13) = cv2.threshold(gray, 20, 255, cv2.THRESH_TOZERO)
(thresh, blackAndWhiteImage14) = cv2.threshold(gray, 80, 255, cv2.THRESH_TOZERO)
(thresh, blackAndWhiteImage15) = cv2.threshold(gray, 160, 255, cv2.THRESH_TOZERO)
(thresh, blackAndWhiteImage16) = cv2.threshold(gray, 200, 255, cv2.THRESH_TOZERO)
tozero_images = [blackAndWhiteImage13,blackAndWhiteImage14,blackAndWhiteImage15,blackAndWhiteImage16]
for i in tozero_images:
  plt.imshow(i,'gray')
  plt.xticks([]),plt.yticks([])
  plt.show()
```

Basit Eşikleme

```
coin = cv2.imread('coins.png',0)

ret, thresh1 = cv2.threshold(coin, 220, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(coin, 220, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(coin, 220, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(coin, 220, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(coin, 220, 255, cv2.THRESH_TOZERO_INV)

coins=[thresh1, thresh2, thresh3, thresh4, thresh5]
for c in coins:
  plt.imshow(c,'gray')
  plt.xticks([]),plt.yticks([])
  plt.show()
```

ADAPTİF EŞİKLEME

Buraya kadar eşik değeri olarak ortak bir değer kullanıldı.

Ancak görüntünün farklı alanlarda farklı aydınlatma koşullarına sahip olduğu tüm koşullarda bu tarz bir eşikleme işe yaramayabilir.

Böyle durumlarda **uyarlamalı** eşikleme kullanılabilir.

Bu tarz eşiklemede Adaptif algoritma, görüntünün küçük bir bölgesi için eşiği hesaplar. Böylece aynı görüntünün farklı bölgeleri için farklı eşikler elde eder ve bu da farklı aydınlatmaya sahip görüntüler için daha iyi sonuçlar verir.

ADAPTİF EŞİKLEME

Üç özel giriş parametresi ve yalnızca bir çıkış değeri vardır.

Uyarlanabilir Yöntem - Eşik değerinin nasıl hesaplanacağına karar verir.

- **cv2.ADAPTIVE_THRESH_MEAN_C** : Eşik değeri komşuluk alanının ortalamasıdır.
- **cv2.ADAPTIVE_THRESH_GAUSSIAN_C** : Eşik değeri, ağırlıkların bir gauss penceresi olduğu komşuluk değerlerinin ağırlıklı toplamıdır.

Blok Boyutu - Komşuluk alanının boyutunu belirler.

C - Sadece ortalamadan veya hesaplanan ağırlıklı ortalamadan çıkarılan bir sabittir.

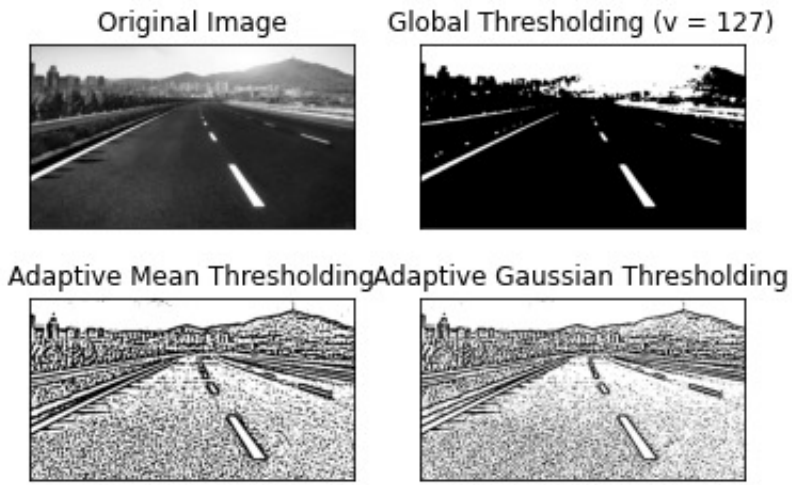
cv2.ADAPTIVE_THRESH_MEAN_C
cv2.ADAPTIVE_THRESH_GAUSSIAN_C

```
high_way = cv2.imread('high_way.jpg',0)
high_way = cv2.medianBlur(high_way,5)

ret, th1 = cv2.threshold(high_way,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(high_way,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 15, 2)
th3 = cv2.adaptiveThreshold(high_way,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 15, 2)

titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
h_w = [high_way, th1, th2, th3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(h_w[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
    plt.show()
```



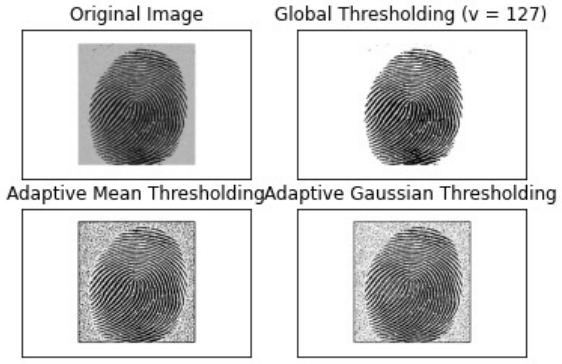
cv2.ADAPTIVE_THRESH_MEAN_C
cv2.ADAPTIVE_THRESH_GAUSSIAN_C

```
finger_print = cv2.imread('fingerprint.jpg',0)
finger_print = cv2.medianBlur(finger_print,5)

ret, th4 = cv2.threshold(finger_print,127,255,cv2.THRESH_BINARY)
th5 = cv2.adaptiveThreshold(finger_print,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 15, 2)
th6 = cv2.adaptiveThreshold(finger_print,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 15, 2)

titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
f_p = [finger_print, th4, th5, th6]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(f_p[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
    plt.show()
```



Otsu Binarizasyonu

Global eşiklemede, eşik değeri için rastgele bir eşik değeri kullanılır.

Ancak bu değerin doğru olup olmadığı tamamen sezgiseldir.

En iyi değere ulaşmak için deneme yanılma yolu kullanılır.

İki modlu bir görüntünün ise eşik değeri olarak yaklaşık olarak tepe noktaların ortasında bir değer alınabilir.

OTSU

Otsu'nun algoritması, ilişki tarafından verilen ağırlıklı sınıf içi varyansı en aza indiren bir eşik değeri (t) bulmaya çalışır.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_1(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_1(t)]^2 \frac{P(i)}{q_2(t)}$$

Temel olarak, her iki sınıftaki varyansların minimum olacağı şekilde iki tepe arasında yer alan bir t değeri bulur.

Otsu Binarizasyonu

Otsu ikilileştirmesinin yaptığı, iki modlu bir görüntü için görüntü histogramından otomatik olarak bir eşik değeri hesaplamaktır.

(İki modlu olmayan görüntüler için ikilileştirme doğru olmayacaktır.)

Not: İki baskın değer olan görüntü türü –iki modlu- (bimodal)

Otsu Binarizasyonu

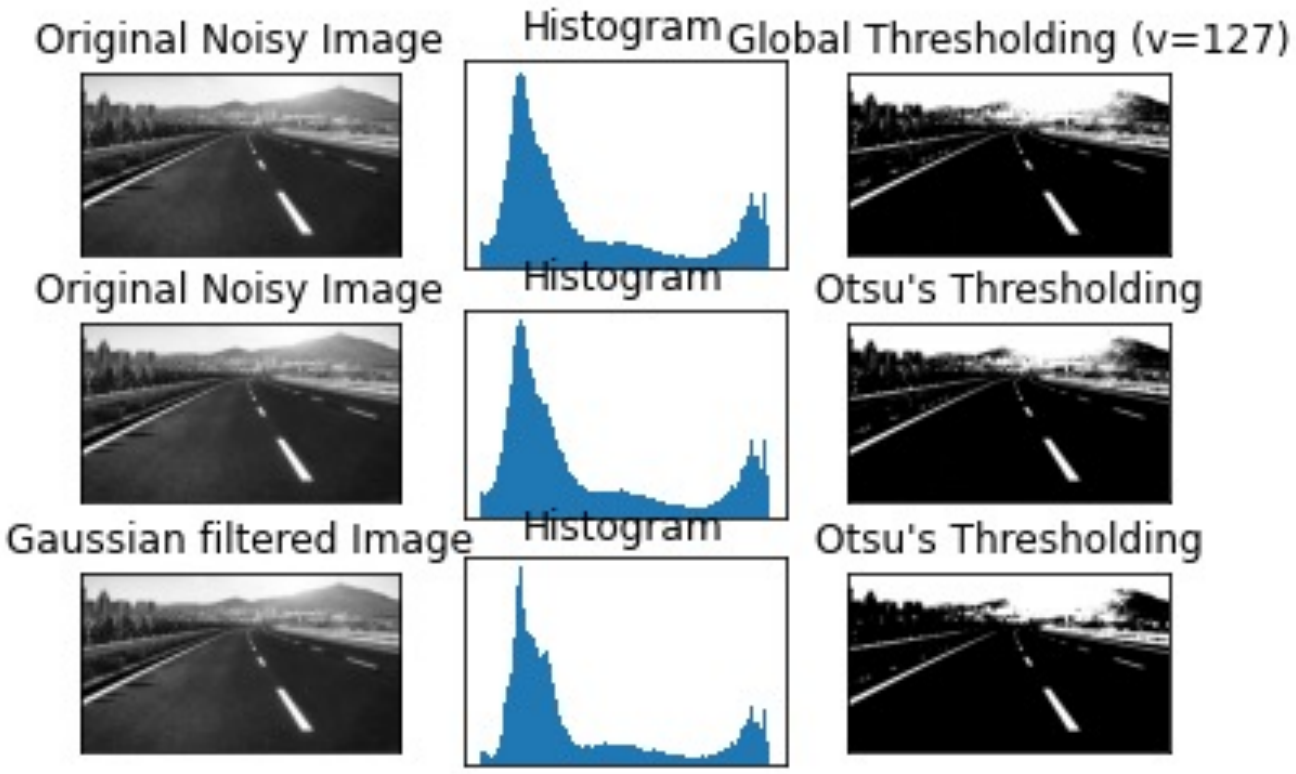
Otsu için **cv2.threshold()** işlevi kullanılır, ancak parametre olarak fazladan bir bayrak olan **cv2.THRESH_OTSU** iletir.

Eşik değeri için sıfırdan farklı bir değer olması gerekir.

Ardından algoritma en uygun eşik değerini bulur ve ikinci çıktı olarak retVal döndürür.

Otsu eşikleme kullanılmıyorsa, retVal değeri kullandığınız eşik değeri ile aynıdır.

Otsu Binarizasyonu




```python
img_h_w= cv2.imread('high_way.jpg',0)

# global thresholding
ret1, h_w_global = cv2.threshold(img_h_w,127,255,cv2.THRESH_BINARY)

# Otsu's thresholding
ret2, h_w_otsu = cv2.threshold(img_h_w,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur = cv2.GaussianBlur(img_h_w,(5,5),0)
ret3, h_w_otsu_blur = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img_h_w, 0, h_w_global,
          img_h_w, 0, h_w_otsu,
          blur, 0, h_w_otsu_blur]
titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',
          'Original Noisy Image','Histogram',"Otsu's Thresholding",
          'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

for i in range(3):
    plt.subplot(3,3,i*3+1), plt.imshow(images[i*3],'gray')
    plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+2), plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+3), plt.imshow(images[i*3+2],'gray')
    plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()
```

KAYNAKLAR

OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#thresholding, Erişim T.: 01.11.2021.

OpenCV, Online: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering, Erişim T.: 01.11.2021.

GeeksforGeeks, https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/ , Erişim T.: 01.11.2021