Sayısal Görüntü İşleme

Dr. Muammer TÜRKOĞLU

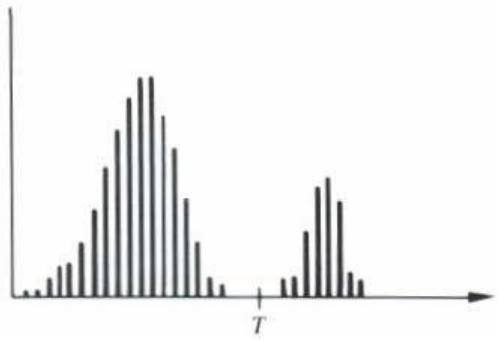
Eşikleme Yöntemleri

Eşikleme, görüntü bölütlemek için çok geniş bir alanda kullanılan, en önemli ve en temel yaklaşımlardan birisidir. Eşikleme işleminde amaç, görüntü içerisindeki nesneleri görüntü arka planından ayırmaktır. Eşikleme için, görüntüdeki gri seviye dağılımlarını gösteren histogramından faydalanılır.

İkili (Tekli) Eşikleme

Klasik eşikleme yöntemi gri düzeyli bir görüntüden ikili görüntü (siyah-beyaz) oluşturmak için kullanılan en basit eşikleme yöntemlerinden biridir. Bu yöntem, tek bir eşik değeri kullanarak görüntü bölütleme işlemini gerçekleştirir. Eşik değer belirleme işlemi gerçekleştirilirken, her bir piksel için gri değerlerinin belirlenen bir T eşik değerinden küçük ya da büyük olup olmadığına bakılır.

$$g(x,y) = \begin{cases} 1 & ; \quad eger\ f(x,y) > T \\ 0 & ; \quad eger\ f(x,y) \leq T \end{cases}$$



THRESH_BINARY;

Kaynak olarak alınan görüntü üzerindeki piksel,esikDegeri olarak verilen değerden büyükse maksDeger olarak verilen parametre değerine atanır.

THRESH_BINARY_INV;

Kaynak olarak alınan görüntü üzerindeki piksel,esikDegeri olarak verilen değerden küçükse maksDeger olarak verilen parametre değerine atanır.THRESH_BINARY_INV, THRESH_BINARY‘nin karşıtı olarak kullanılabilir.

THRESH_TRUNC;

Kaynak olarak girilen resimdeki piksel, eğer eşik değerinden büyükse, eşik değeri olarak girilen değere eşitlenir, büyük değil ise pikselin değeri aynı kalır.

THRESH_TOZERO;

Kaynak olarak girilen resimdeki piksel, eğer eşik değerinden büyükse piksel değeri aynı kalacak, büyük değilse 0 değerine eşit olacak.

THRESH_TOZERO_INV;

Kaynak olarak girilen resimdeki piksel, eğer eşik değerinden büyükse 0 değerini alacak, büyük değil ise piksel değeri aynı kalacak.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img1=cv2.imread('zebra.png')
img1 = cv2.resize(img1, (1000, 550))
gray = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)

ret,thresh1 = cv2.threshold(gray,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(gray,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(gray,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(gray,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(gray,127,255,cv2.THRESH_TOZERO_INV)

titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [gray, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray',vmin=0,vmax=255)
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Otsu Eşikleme

Otsu eşikleme, gri seviyeli görüntüler üzerinde uygulanabilen bir eşik değeri belirleme yöntemidir. Otsu yöntemi, görüntü üzerinden eşik değerini hesaplamaktadır. Bu yöntem, görüntünün arka plan ve ön plan olmak üzere iki ayrı sınıf olduğunu varsayarak, tüm eşik değerleri için bu iki sınıfın sınıf içi varyans değerini minimum yapacak değeri bulmaya çalışır. Sınıf içi varyans değerini minimum yapan eşik değeri, sınıflar arası varyans değerini maksimum yapar. Varyans belirli bir dizinin elemanlarının, bu dizinin aritmetik ortalaması etrafındaki dağılımın analizini sağlayan bir istatistiksel ölçüdür. Bu değere bakılarak, dizi içerisindeki değerlerin ne derecede dağınık oldukları yorumu yapılabilmektedir.

OTSU Global Eşikleme Algoritması

Sınıflar arası varyansı maksimize eder

Görüntünün normalize edilmiş histogram değerlerini $p_i, i = 0,1,\dots,L-1$ olarak düşünelim. k eşik değeri ile görüntüyü eşiklemek istersek bu durumda $0,1,2,\dots,k$ renk değerleri C_1 sınıfına, $k+1, k+2, \dots, L-1$ değerleri ise C_2 sınıfına ait piksel değerleri olur. Böylece sınıflar arası varyans değeri aşağıdaki şekilde bulunabilir:

$$\sigma_B^2(k) = \underbrace{P_1(k)[m_1(k) - m_G]^2}_{\text{Birinci sınıf}} + \underbrace{P_2(k)[m_2(k) - m_G]^2}_{\text{İkinci sınıf}}$$

$P_1(k) = \sum_{i=0}^{k} p_i$ — Birinci sınıftaki piksellerin toplam tekrarlanma sıklığı

$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$ — İkinci sınıftaki piksellerin toplam tekrarlanma sıklığı

$m_G = \sum_{i=0}^{L-1} i p_i$ — Tüm piksellerin ortalaması

$m_1(k)$ — Birinci sınıftaki piksellerin ortalaması

$m_2(k)$ — İkinci sınıftaki piksellerin ortalaması

$$OTSU\ eşik\ değeri = \max(\sigma_B^2(k))$$

```
import cv2

img1=cv2.imread('zebra.png')
img1 = cv2.resize(img1, (1000, 550))
gray = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)

ret,im1 = cv2.threshold(gray,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow("orjinal",gray)
cv2.imshow("otsu",im1)
```

```
import cv2
import cv2
import numpy as np
def getOtsuThreshold(im):
    size = 256
    buckets = np.zeros([size])
    imy= im.shape[0]
    imx = im.shape[1]
    image_size = imx*imy
    for i in range(imy):
        for j in range(imx):
            buckets[im[i][j]] += 1

    in_class_variance_list = []
    for i in range(1,size):
        wf = np.sum(buckets[i:]) / image_size
        wb = 1-wf
        mf = np.sum(np.dot(range(i,size),buckets[i:])) / np.sum(buckets[i:])
        mb = np.sum(np.dot(range(i),buckets[:i])) / np.sum(buckets[:i])
        varf = np.sum( np.dot( np.power( range(i, size) - mf, 2 ),buckets[i:size] ) )/ np.sum(buckets[i:])
        varb = np.sum( np.dot( np.power( range(i) - mb, 2 ),buckets[:i] ) )/ np.sum(buckets[:i])
        in_class_variance = varf * wf + varb * wb
        in_class_variance_list.append(in_class_variance)
    return in_class_variance_list.index(min(in_class_variance_list))

img1=cv2.imread('zebra.png')
img1 = cv2.resize(img1, (1000, 550))
gray = cv2.cvtColor(img1,cv2.COLOR_BGR2GRAY)
getOtsuThreshold(gray)
```

Görüntü Histogramının Oluşturulması

Sınıf-içi Varyans Değerinin Minimize Edilmesi
(Otsu eşik değer yöntemi mümkün olan bütün eşik değerler için(255'e kadar yani) bütün eşik değerler için sınıf-içi varyans denilen bir değer hesaplar ve bu değerin en düşük olduğu indeksi döndürür**)**
Sınıfların Ağırlıkları
Sınıfların Ortalamaları
Sınıfların Varyansları
Varyans İnterpolasyonu

Adaptive Thresholding

Adaptif eşiklemede resmin küçük bir bölgesi için eşik değeri hesaplanır. Bu nedenle, aynı görüntünün farklı bölgelerinde farklı eşikler elde ediyoruz ve farklı aydınlatmalara sahip görüntüler için daha iyi sonuç vermektedir.

Adaptif eşiklemede, eşik değerinin nasıl hesaplanacağına karar veren iki yöntem mevcuttur.

- **ADAPTIVE_THRESH_MEAN_C:** eşik değeri komşu piksellerin alanlarının ortalamasıdır.
- **ADAPTIVE_THRESH_GAUSSIAN_C:** eşik değeri, ağırlıkların bir gaussian penceresi olduğu komşuluk değerlerinin ağırlıklı toplamıdır.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('sudoku.jpg')
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
            'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]
for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```