



KIRIKKALE ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ BİL1006-BİLGİSAYAR GRAFİĞİNE GİRİŞ

Doç. Dr. Serkan SAVAŞ

Görüntüde Tespit İşlemleri

- Kontur İşlemleri
- Histogramlar
- Şablon Eşleştirme
- Görüntü Segmentasyonu
(Watershed Alg.)



Kontur



Konturlar, aynı renk veya yoğunluğa sahip tüm sürekli noktaları (sınır boyunca) birleştiren bir eğri olarak tanımlanabilir. Konturlar, şekil analizi ve nesne algılama ve tanıma için kullanışlı bir araçtır.

- İkili görüntülerde daha verimli çalıştığı için konturları bulmadan önce eşik veya keskin kenar algılama uygulanabilir.
- `findContours()` fonksiyonu kaynak görüntüyü değiştirdiği için kaynak görüntünün kopyasını bulundurmak faydalıdır.
- OpenCV'de kontur bulmak, siyah arka plandan beyaz nesne bulmak gibidir. Bu nedenle, bulunacak nesnenin beyaz ve arka planın siyah olması gereklidir.

findContours()

cv2.findContours() işlevinde üç argüman vardır;

- birincisi kaynak görüntü,
- ikincisi kontur alma modu,
- üçüncü kontur yaklaşım yöntemidir. Konturları ve hiyerarşiyi verir.

Konturlar, görüntüdeki tüm konturların bir listesidir. Her bir bireysel kontur, nesnenin sınır noktalarının ((x,y) koordinatlarının) bir Numpy dizisidir.

Konturları Çizdirme

Konturları çizmek için `cv2.drawContours()` fonksiyonu kullanılır. Sınır noktalarına sahip olmak koşuluyla herhangi bir şekli çizmek için de kullanılabilir.

İlk argümanı kaynak görüntü, ikinci argüman Python listesi olarak iletilmesi gereken konturlar, üçüncü argüman kontur indeksi (Tekil kontur çizerken kullanışlıdır. Tüm konturları çizmek için -1 kullanılır). Diğer argümanlar renk, kalınlık vb...

Konturları Çizdirme

Orjinal



Kontur



```
from matplotlib import pyplot as plt
import cv2

img = cv2.imread('cont.jpg')
img2 = cv2.imread('cont.jpg')
imggray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imggray,127,255,0)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

cv2.drawContours(img, contours, -1, (255,0,0), 3)

plt.subplot(121), plt.imshow(img2), plt.title('Orjinal')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(img), plt.title('Kontur')
plt.xticks([]), plt.yticks([])
```

Kontur Yaklaşım Yöntemi

Konturlar, aynı yoğunluğa sahip bir şeklin sınırlarıdır. Bir şeklin sınırının (x,y) koordinatlarını saklar. Ama tüm koordinatları saklayıp saklamadığı bu kontur yaklaşımı yöntemi ile belirlenir.

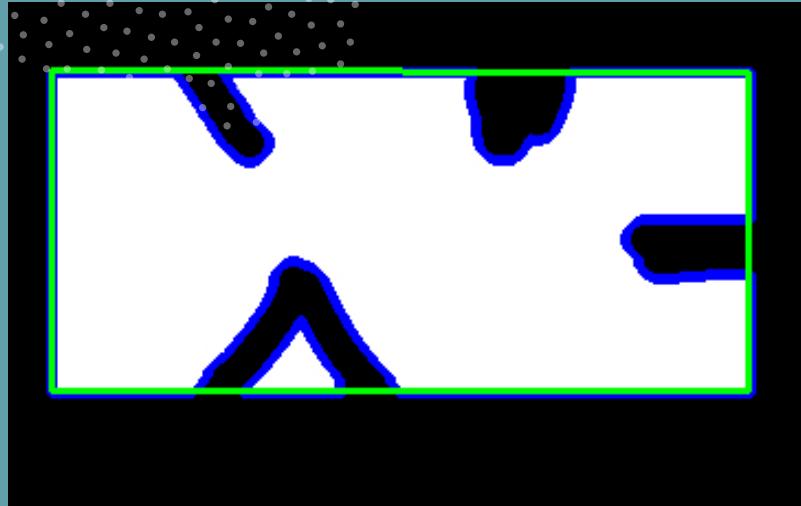
cv2.CHAIN_APPROX_NONE: Tüm sınır noktaları tutar.

Ancak tüm noktalara ihtiyaç var mı? Örneğin, düz bir çizginin konturunu bulunduğunda bu çizgiyi temsil etmek için çizgideki tüm noktalara ihtiyaç var mı? O çizginin sadece iki uç noktası yeterli.

cv2.CHAIN_APPROX_SIMPLE bunu yapar. Tüm gereksiz noktaları kaldırır ve konturu sıkıştırır, böylece bellekten tasarruf sağlar.

Sınırları Tamamlama

Dikdörtgen için kullanılan fonksiyon cv2.minAreaRect(). Bu dikdörtgeni çizmek için dikdörtgenin 4 köşesine ihtiyaç vardır. Bu da cv2.boxPoints() fonksiyonuyla elde edilir.



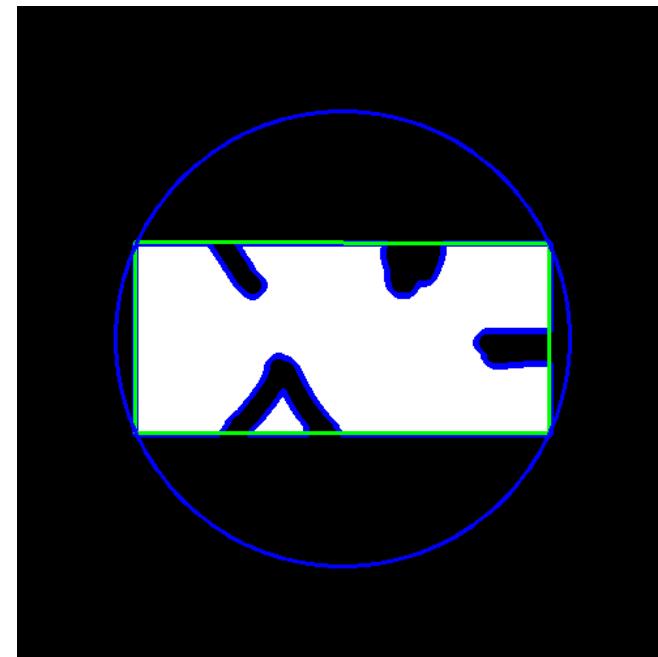
```
cnt = contours[1]
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int0(box)
cv2.drawContours(img, [box], 0, (0,255,0), 2)
cv2_imshow(img)
```

Kontur alanı

`cv2.minEnclosingCircle()` fonksiyonu kullanarak da bir nesnenin çevresi bulunabilir. Bu minimum alana sahip cismi tamamen kaplayan bir dairedir.

```
(x,y),radius = cv2.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)
cv2.circle(cont,center,radius,(255,0,0),2)
cv2_imshow(cont)
```

Elips ve Çizgi seçenekleri de mevcuttur...



Dışbükey Kaplama

Dışbükey Gövde, kontur yaklaşımına benzer bir yaklaşımdır. Burada cv2.convexHull() işlevi, bir eğriyi kontrol eder ve dışbükeylik kusurlarını düzeltir.

Genel olarak dışbükey eğriler, her zaman dışarı doğru ya da en azından düz olan eğrilerdir. İçeride kalan böümlere ise dışbükeylik kusurları denir.



Dışbükey Kaplama

```
hull = cv2.convexHull(points[, hull[, clockwise[,  
returnPoints]]])
```

points, içinden geçtiğimiz konturlardır.

Hull, çıktıdır.

Clockwise, Yönlendirme bayrağıdır. True ise, çıkış dışbükey gövdesi saat yönünde yönlendirilir. Aksi takdirde, saat yönünün tersine yönlendirilir.

returnPoints, Varsayılan olarak True. Daha sonra gövde noktalarının koordinatlarını döndürür. False ise, gövde noktalarına karşılık gelen kontur noktalarının indekslerini döndürür. Ancak dışbükeylik kusurlarını bulmak istiyorsanız, returnPoints = False'ı geçmek gereklidir.

HULL

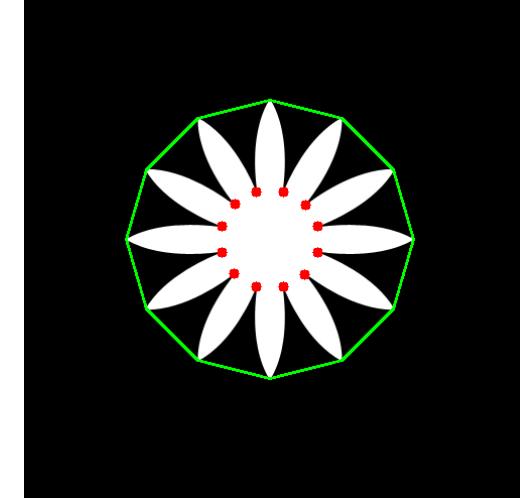
OpenCV'de dışbükey kusurları bulmak için bir fonksiyon:
cv2.convexityDefects().

```
star = cv2.imread('star.jpg')
star_gray = cv2.cvtColor(star, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(star_gray, 127, 255, 0)
star_contours, star_hierarchy = cv2.findContours(thresh, 2, 1)
star_cnt = star_contours[0]

hull = cv2.convexHull(star_cnt, returnPoints = False)
defects = cv2.convexityDefects(star_cnt, hull)

for i in range(defects.shape[0]):
    s,e,f,d = defects[i,0]
    start = tuple(star_cnt[s][0])
    end = tuple(star_cnt[e][0])
    far = tuple(star_cnt[f][0])
    cv2.line(star,start,end,[0,255,0],2)
    cv2.circle(star,far,5,[0,0,255],-1)

cv2_imshow(star)
```



Kontur Karşılaştırma



OpenCV, elde edilen konturlar üzerinden nesnelerin karşılaştırmasına da olanak sağlayan bir fonksiyon sunmaktadır. Sonuç ne kadar düşükse, o kadar iyi eşleşme vardır demektir.



cv2.matchShapes()

Kontur Hiyerarşisi

cv2.findContours() fonksiyonu kullanarak görüntüdeki konturlar bulunduğuanda, Contour Retrieval Mode adlı bir argümanı verildi.

Bu argümanlar genellikle cv2.RETR_LIST veya cv2.RETR_TREE şeklinde kullanılıyor.

Ayrıca çıktı olarak da üç dizi kullandık. İlkı Görüntü, ikincisi konturları ve sonucusu hiyerarsi.



Hiyerarşî

Normalde bir görüntüdeki nesneleri algılamak için **cv2.findContours()** kullanıldı.

Bu fonksiyonun kullanıldığı görüntülerde bazen nesneler farklı konumlardadır. Ancak bazı durumlarda bazı şekiller diğer şekillerin içindedir.

Bu durumda dıştaki **ebeveyn**, içteki **çocuk** olarak adlandırılır. Bu şekilde, bir görüntüdeki konturların birbirleriyle bir ilişkisi vardır ve bir konturun birbirine nasıl bağlı olduğunu belirtilebilir.

Bu ilişkinin temsiline **Hiyerarşî** denir.

RETR_LIST ve RETR_EXTERNAL

RETR_LIST

Tüm konturları alır, ancak herhangi bir ebeveyn-çocuk ilişkisi oluşturmaz. Ebeveynler ve çocuklar bu kurala göre eşittir ve onlar sadece dış hatlardır. Yani hepsi aynı hiyerarşi düzeyine aittir.

RETR_EXTERNAL

Bu bayrak kullanıldığında, yalnızca aşırı dış bayrakları döndürür. Tüm çocuk konturları geride bırakılır. Bu kurala göre, her ailede sadece en büyüğüne bakılır.

RETR_CCOMP ve RETR_TREE

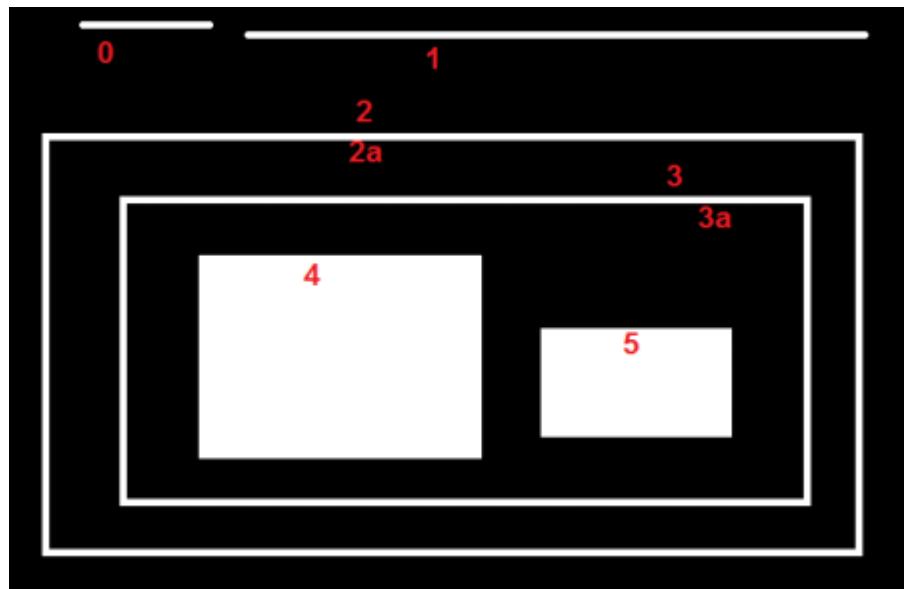
RETR_CCOMP

Bu bayrak tüm konturları alır ve onları 2 seviyeli bir hiyerarşije göre düzenler. Yani nesnenin dış konturları (sınırı) hiyerarşı-1'e yerleştirilir. Nesnenin içindeki deliklerin konturları (varsı) hiyerarşı-2'ye yerleştirilir.

RETR_TREE

Tüm konturları alır ve tam bir aile hiyerarşi listesi oluşturur.

Hiyerarşî

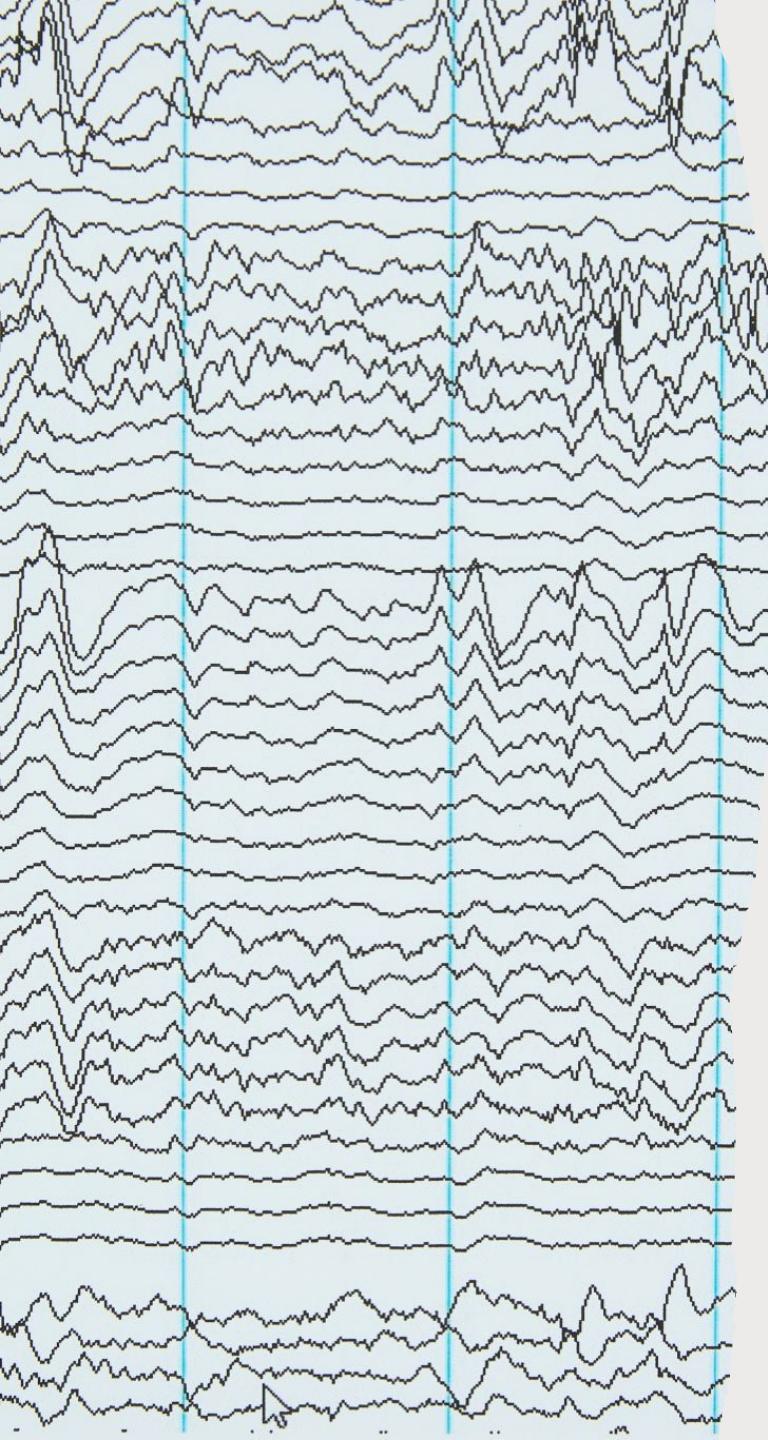


Bu görüntüde 0-5 arasında numaralandırılan şekiller var. 2 ve 2a, en dıştaki kutunun dış ve iç hatlarını belirtiyor.

Burada 0,1, 2 konturları dış veya en dıştadır. Hiyerarşî-0'da olduklarıını veya aynı hiperarşî düzeyinde oldukları söylenebilir.

Ardından kontur-2a gelir. Kontur-2'nin çocuğu olarak kabul edilebilir (veya tersi şekilde kontur-2, kontur-2a'nın ebeveynidir). Öyleyse hiperarşî-1 denilebilir.

Benzer şekilde kontur-3, kontur-2'nin çocuğuudur ve bir sonraki hiperarşide gelir. Son olarak konturlar 4, 5 ise kontur-3a'nın çocuklarıdır ve son hiperarşî seviyesinde gelirler.



Histogram

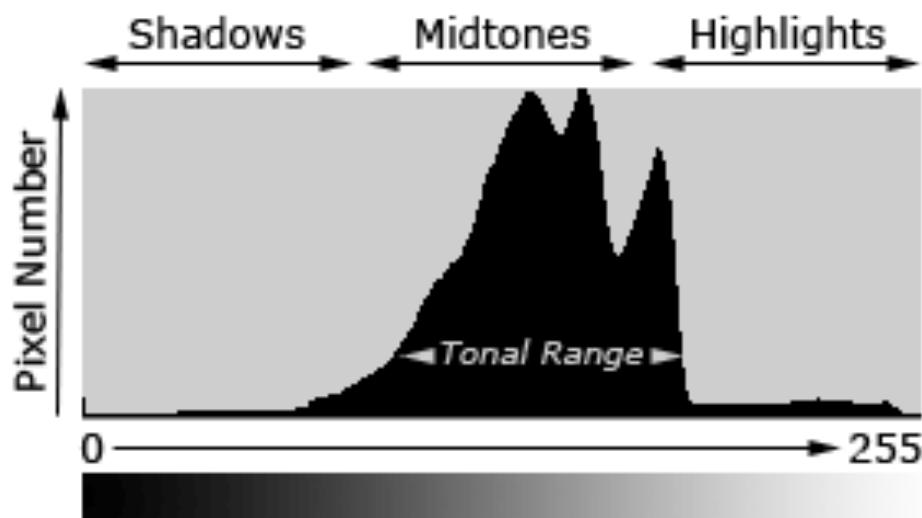
Histogram, bir görüntünün yoğunluk dağılımı hakkında genel bir fikir veren bir grafik veya çizim olarak düşünülebilir.

X ekseninde piksel değerleri (0 ile 255 arasında, her zaman değil) ve Y ekseninde görüntüde karşılık gelen piksel sayısıyla bir çizimdir.

Histogram, görüntüyü anlamanın başka bir yoludur. Bir görüntünün histogramına bakarak, o görüntünün kontrasti, parlaklığı, yoğunluk dağılımı vb. hakkında bilgiler elde edilebilir. Günümüzde hemen hemen tüm görüntü işleme araçları, histogram bilgilerini sağlamaktadır.

Histogram

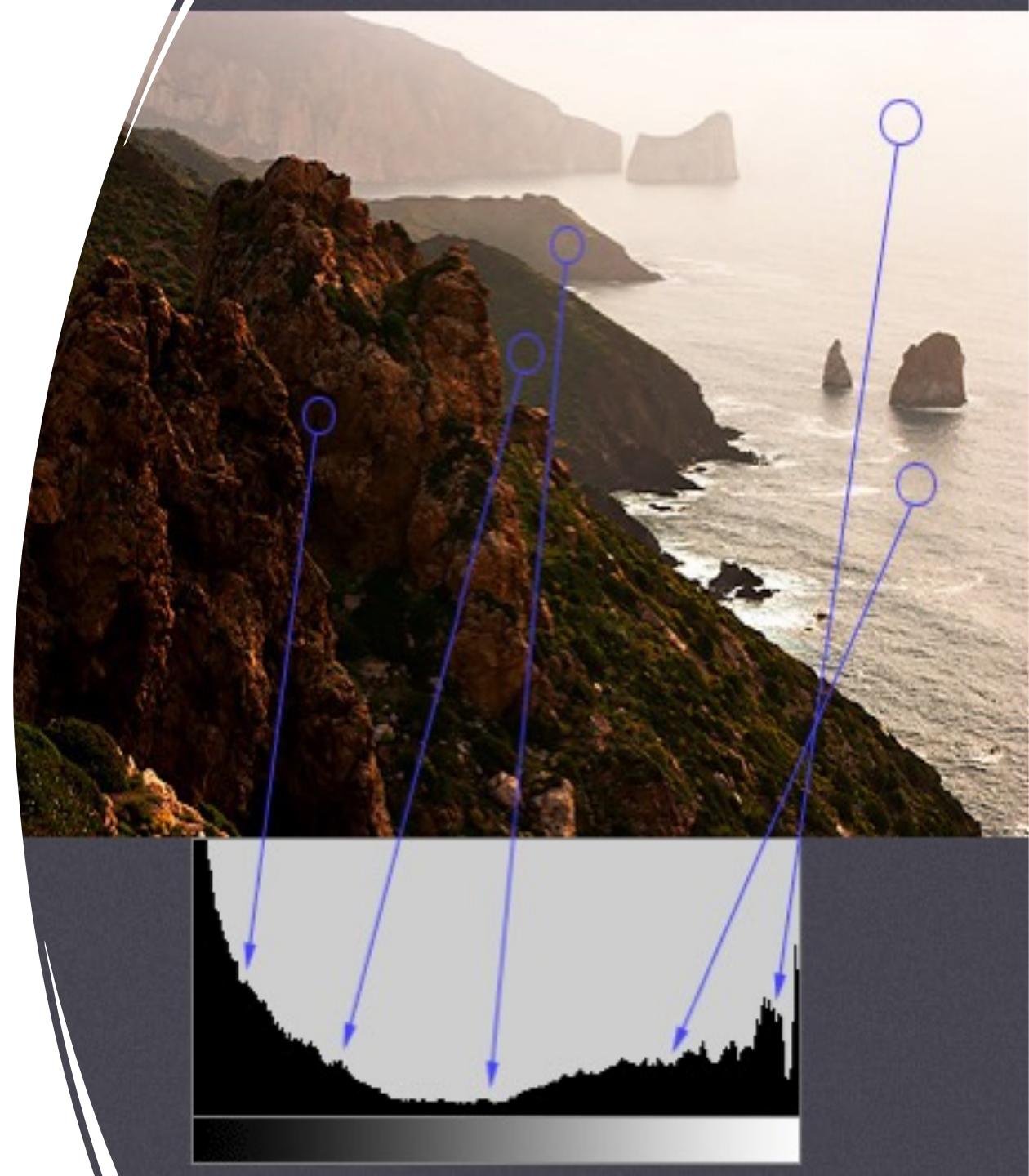
Bir histogram, görüntünün uygun şekilde pozlanıp pozlanmadığını, aydınlatmanın sert mi yoksa düz mü olduğunu ve hangi ayarların en iyi sonucu vereceğini belirtebilir.



Parlaklık değerlerinin çoğunun bulunduğu bölgeye "ton aralığı" denir. Ton aralığı görüntüden görüntüye büyük ölçüde değişebilir. Tüm görüntülerin taklit etmeye çalışması gereken tek bir "ideal histogram" yoktur.

Histogram

- Yandaki görüntü, geniş bir ton aralığı içeren bir örnektir. Bu görüntü çok az orta ton içeriyor, ancak sırasıyla görüntünün sol alt ve sağ üst kısımlarında bol miktarda gölge ve vurgu bölgeleri var.
- Bu, hem en sol hem de sağ tarafta yüksek piksel sayısına sahip bir histograma dönüşür.



Histogram

Tonların çoğunun gölgelerde meydana geldiği görüntülere "düşük anahtar" denirken, "yüksek anahtar" görüntülerde tonların çoğu açık tonlardadır.

BINS : Histogram her piksel değeri için piksel sayısını gösterir. Yani 0'dan 255'e kadar. Böylece bir histogramı göstermek için 256 değere ihtiyaç olur. Ancak tüm piksel değerleri için piksel sayısını ayrı ayrı değil de, piksel değerleri aralığındaki piksel sayısını bulmak gerekiyorsa Histogramı temsil etmek için yalnızca 16 değere ihtiyaç duyulur.

Örneğin, 0 ila 15, ardından 16 ila 31, ..., 240 ila 255 arasında uzanan piksel sayısını bulmak gerekiyor.

Histogram - Bins

Sonuç olarak burada yapılan tüm histogramı 16 alt parçaaya bölmektir.

Her bir alt parçasının değeri, içindeki tüm piksel sayısının toplamıdır. Bu her bir alt parçaaya “**BIN**” denir.

İlk durumda, 256 (her piksel için bir) olan bins sayısı, ikinci durumda sadece 16'dır. **BINS**, OpenCV belgelerinde **histSize** terimi ile temsil edilir.

Histogramlar, önceden tanımlanmış bir dizi kutuda (**BINS**) düzenlenen toplanan veri sayılarıdır.

Histogram

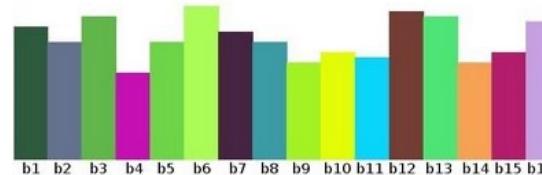
Veri sadece yoğunluk değerleriyle sınırlanmaz. Toplanan veriler, görüntüyü tanımlamak için yararlı olabilecek herhangi bir özellik olabilir.

254	143	203	176	106	229	177	220	192	9	229	142	138	64	0	63	28	8	88	82
27	68	231	75	141	107	149	210	13	239	141	35	68	242	110	208	244	0	33	88
54	42	17	215	230	254	47	41	98	180	55	253	235	47	122	208	76	110	152	100
9	186	192	71	104	193	88	171	37	233	18	147	174	1	143	211	176	188	192	68
179	20	238	192	190	132	41	248	22	134	83	133	110	254	176	238	168	234	51	204
232	25	0	183	174	129	61	30	110	189	0	173	197	183	153	43	22	87	68	118
235	35	151	185	120	81	239	170	195	94	38	21	67	101	58	37	196	149	52	154
135	242	54	0	104	109	189	47	130	254	225	156	31	181	121	15	128	35	252	205
223	114	79	129	147	6	201	68	89	107	58	44	253	84	38	1	62	5	231	218
55	168	237	188	80	101	131	241	68	133	124	151	111	28	190	4	240	78	117	145
152	155	229	78	90	217	219	105	118	77	38	49	2	9	214	181	205	118	135	33
182	94	178	199	20	149	57	223	232	113	32	45	177	15	31	179	100	119	208	81
224	118	124	172	75	29	69	180	187	195	41	44	8	170	158	101	131	31	28	112
238	33	38	7	83	69	173	183	98	237	67	227	18	218	248	237	75	192	201	146
88	195	224	207	140	22	31	118	234	34	182	116	23	47	68	242	189	152	110	248
140	37	101	230	246	145	122	64	27	58	229	1	225	143	91	100	98	90	40	195
251	4	178	139	121	95	97	174	249	182	77	115	223	188	182	82	65	252	83	196
179	160	223	230	87	162	148	78	176	19	17	4	184	178	183	102	83	81	132	206
173	137	185	242	181	181	214	49	74	238	197	37	98	102	15	217	148	8	102	168
85	9	17	222	18	210	70	21	78	241	184	216	93	93	208	102	153	212	119	47

Bu verileri organize bir şekilde saymak için ve ayrıca bilgi değeri aralığının 256 değer olduğunu bildiğimiz için, aralığımızı aşağıdaki gibi alt bölümlere (bin'ler) ayıralım:

$$[0, 255] = [0, 15] \cup [16, 31] \cup \dots \cup [240, 255]$$
$$\text{range} = \text{bin}_1 \cup \text{bin}_2 \cup \dots \cup \text{bin}_{n=15}$$

ve her bin aralığına düşen piksel sayısını tutabiliyoruz. Bunu yukarıdaki örneğe uygulayarak aşağıdaki görüntüyü elde ederiz (x eksenini bin'ler ve y eksenini her birindeki piksel sayısını temsil eder).



Histogram

DIMS : Verilerini topladığımız parametre sayısıdır. Bu durumda, yalnızca tek bir şeyle, yani yoğunluk değeriyle ilgili veriler toplandığı için «1» dir.

RANGE : Ölçmek istenilen yoğunluk değerleri aralığıdır. Normalde [0, 256] yani tüm yoğunluk değerleridir.

OpenCV'de histogram için: **cv2.calcHist()**
kullanılır.

Histogram

cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]]])

images : uint8 veya float32 türündeki kaynak görüntüdür. köşeli parantez içinde verilmelidir, yani “[img]”.

channels : Yine köşeli parantez içinde verilmelidir. Histogramı hesaplanan kanalın indeksidir. Örneğin, giriş gri tonlamalı görüntü ise değeri [0]'dır. Renkli görüntü için sırasıyla mavi, yeşil veya kırmızı kanalın histogramını hesaplamak için [0], [1] veya [2]'yi yazılabilir.

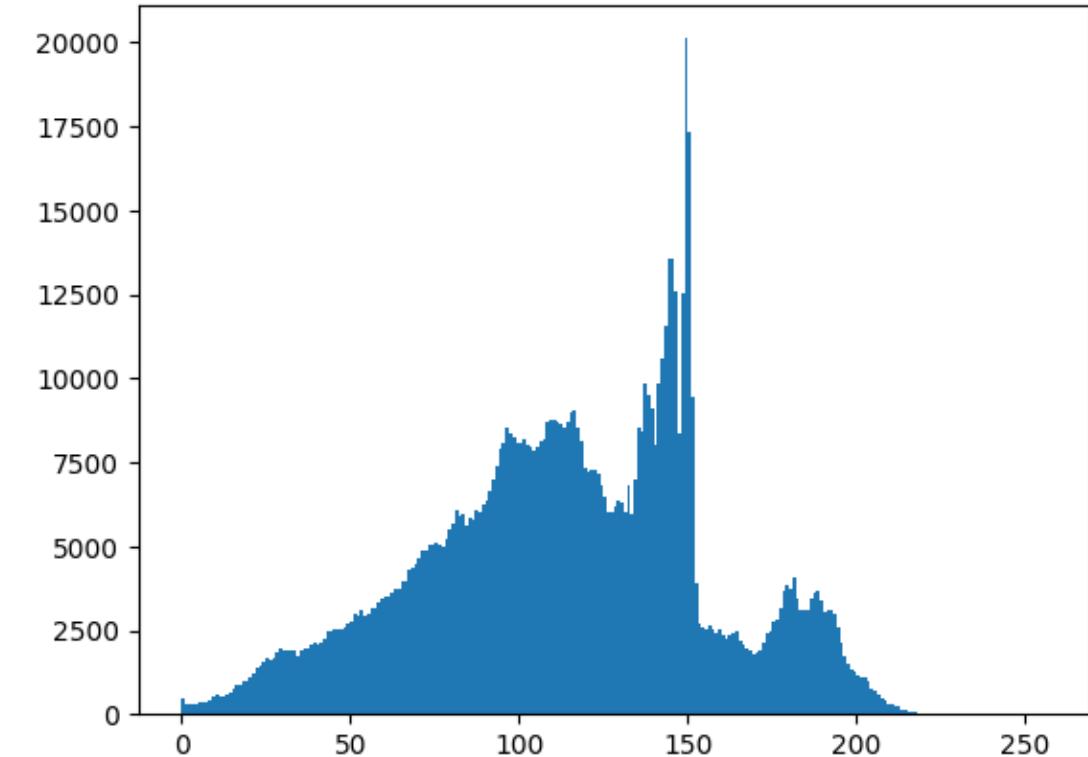
mask : Maske görüntüsü. Tam görüntünün histogramını bulmak için “None” olarak verilir. Ancak görüntünün belirli bir bölgesinin histogramı bulunacaksa, bir maske görüntüsü oluşturulmalı ve maske olarak verilmelidir.

histSize : BIN sayısını temsil eder. Köşeli parantez içinde verilmesi gereklidir. Tam ölçek için [256] verilir.

ranges : bu bizim RANGE'ımızdır. Normalde [0, 256]'dır.

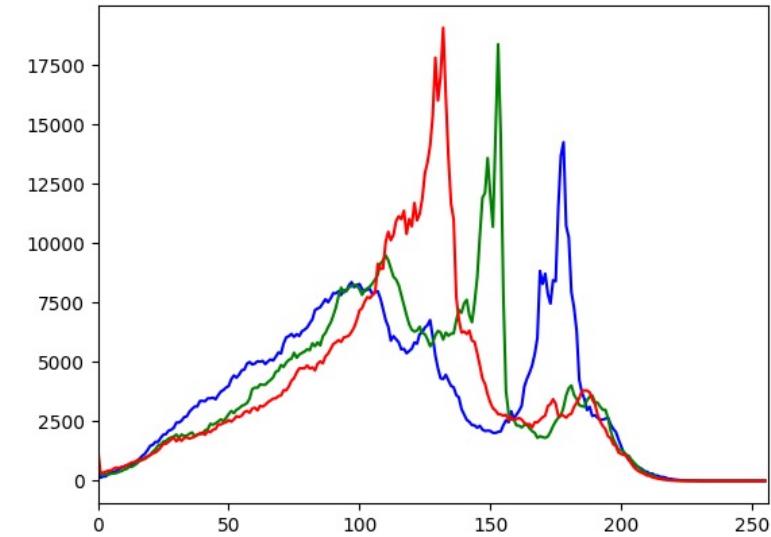
Histogram

```
kampus = cv2.imread("kku_kampus.jpg",0)
cv2_imshow(kampus)
hist = cv2.calcHist([kampus], [0], None, [256], [0,256])
plt.hist(kampus.ravel(), 256, [0,256]); plt.show()
```



Histogram

```
kampus2 = cv2.imread("kku_kampus.jpg")
cv2_imshow(kampus2)
color =('b', 'g', 'r')
for i, col in enumerate(color):
    color_hist = cv2.calcHist([kampus2], [i], None, [256], [0,256])
    plt.plot(color_hist, color=col)
    plt.xlim([0,256])
plt.show()
```



Histogram Eşitleme

Piksel değerleri yalnızca belirli bir değer aralığıyla sınırlandırılmış bir görüntü olsun. Örneğin, daha parlak görüntüde tüm pikseller yüksek değerlerle sınırlandırılır. Ancak iyi bir görüntüde, tüm bölgelerden piksellere sahip olur.

Bu nedenle, bu histogramı her iki uca uzatmak gereklidir ve buna **Histogram Eşitleme** denir. Bu işlem normalde görüntünün kontrastını iyileştirir.

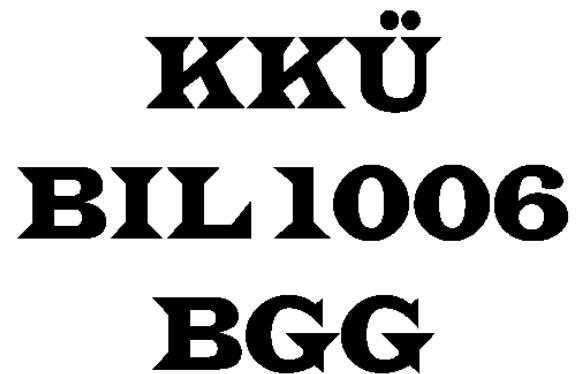
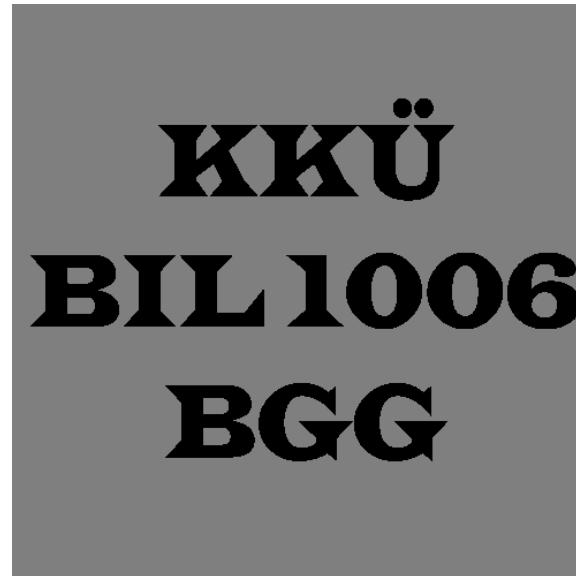
cv2.equalizeHist()

Histogram eşitleme

- equ = cv2.equalizeHist(img4)
- res = np.hstack((img4, equ))
- cv2_imshow(res)

Histogram eşitleme, görüntünün histogramı belirli bir bölgeyle sınırlandırıldığında iyi sonuç verir.

Histogramın geniş bir alanı kapsadığı, yani hem parlak hem de karanlık piksellerin bulunduğu büyük yoğunluk değişimlerinin olduğu yerlerde iyi çalışmayaçaktır.



Global Histogram Eşitleme'den farklı olarak adaptif Histogram Eşitleme de mevcut.
CLAHE (Contrast Limited Adaptive Histogram Equalization)

Şablon Eşleştirme

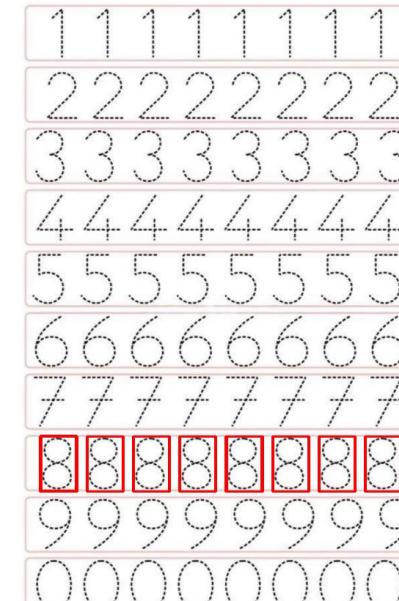
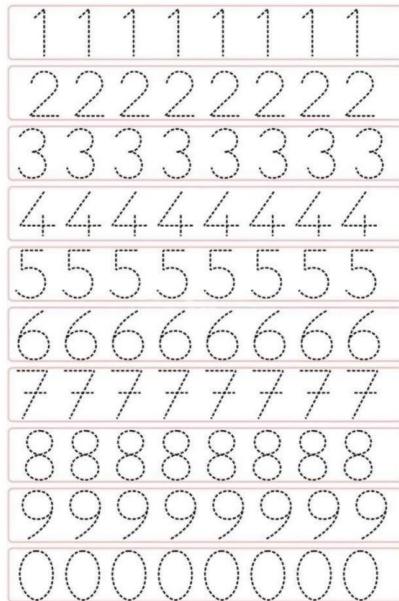
- Şablon Eşleştirme, daha büyük bir görüntüde bir şablon görüntüsünün konumunu aramak ve bulmak için kullanılan bir yöntemdir.
- OpenCV, bu amaç için `cv2.matchTemplate()` fonksiyonu kullanılır.
- Bu işlem, şablon görüntüsünü giriş görüntüsünün üzerine kaydırır (2B evrişimde olduğu gibi) ve şablon görüntüsünün altındaki şablon ve giriş görüntüsünün eşleşmesini karşılaştırır.
- OpenCV'de çeşitli karşılaştırma yöntemleri uygulanmaktadır.
 - `cv2.TM_CCOEFF`
 - `cv2.TM_CCOEFF_NORMED`
 - `cv2.TM_CCORR`
 - `cv2.TM_CCORR_NORMED`
 - `cv2.TM_SQDIFF`
 - `cv2.TM_SQDIFF_NORMED`

Şablon Eşleştirme

```
img_rgb = cv2.imread('say.jpg')
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
template = cv2.imread('8.JPG',0)
w, h = template.shape[::-1]

res = cv2.matchTemplate(img_gray,template, cv2.TM_CCOEFF_NORMED)
threshold = 0.8
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)

cv2.imwrite('res2.png',img_rgb)
cv2_imshow(img_rgb)
```

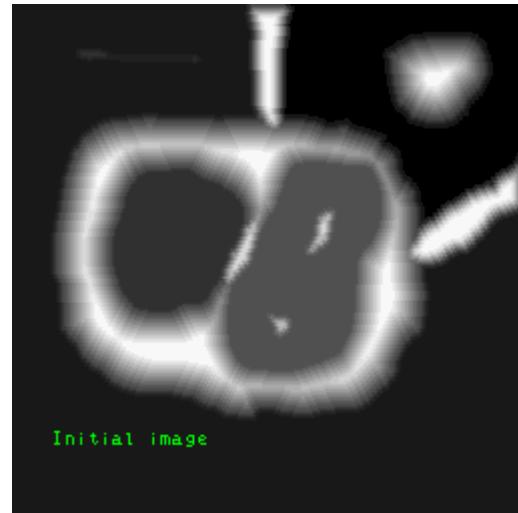


Watershed Algoritması ile Görüntü Segmentasyonu

Herhangi bir gri tonlamalı görüntü, yüksek yoğunluğun tepeleri, düşük yoğunluğun ise vadileri gösterdiği bir topografik yüzey olarak görülebilir. İzole edilmiş her vadiyi (yerel minimum) farklı renkli su (etiketler) ile doldurmaya başlanır. Su yükseldikçe, yakınlardaki tepelere (gradyanlara) bağlı olarak, farklı vadilerden gelen, farklı renklerde sular birleşmeye başlayacaktır.

Bunu önlemek için suyun birleştiği yerlere bariyerler örülür. Tüm tepeler sular altında kalana kadar su doldurma ve bariyerler inşa etme işine devam edilir. Ardından oluşturulan engeller segmentasyon sonucunu verir.

Watershed Algoritması ile Görüntü Segmentasyonu



<https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html>

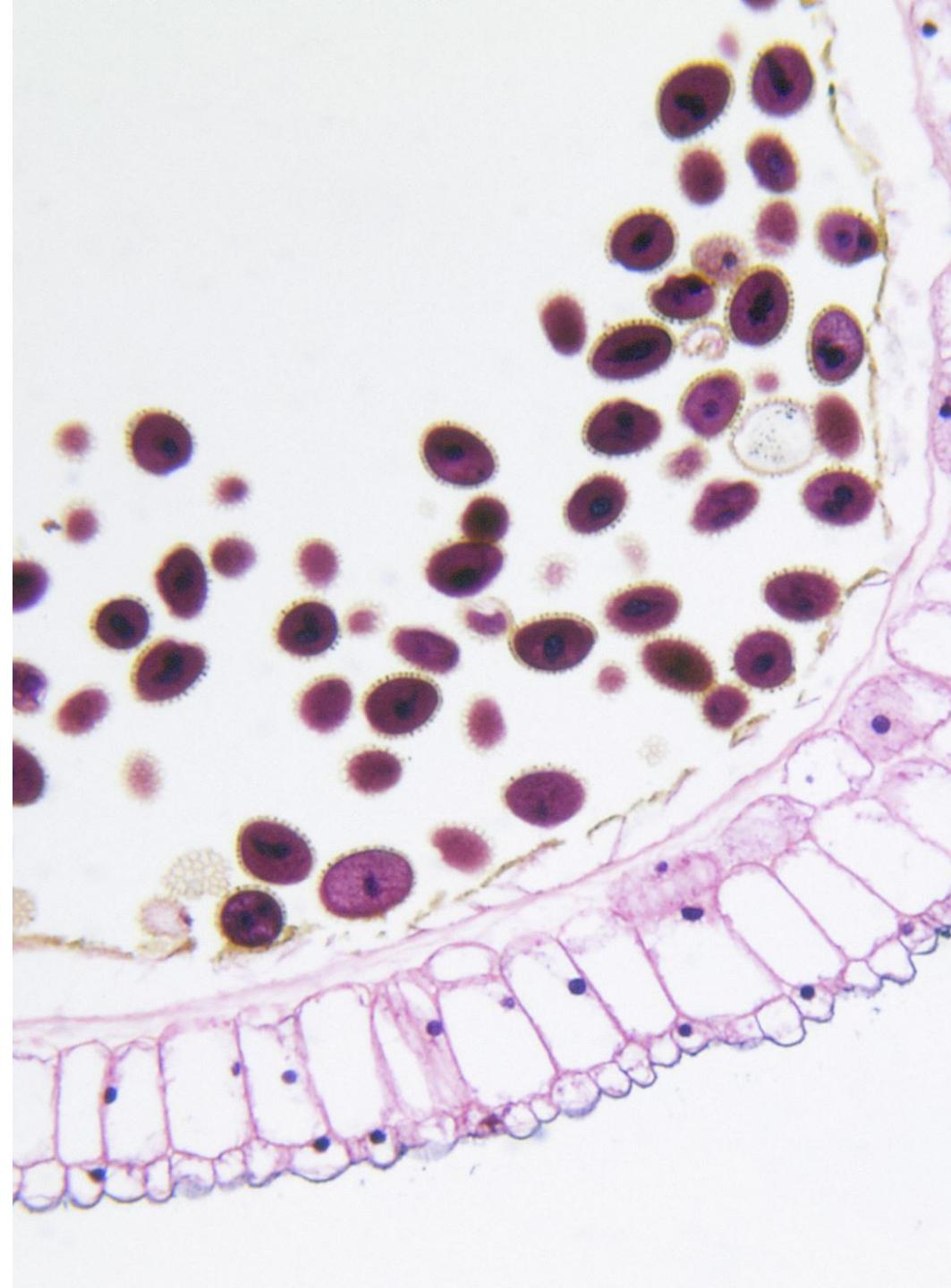
Watershed Algoritması ile Görüntü Segmentasyonu

Ancak bu yaklaşım, görüntüdeki parazit veya diğer düzensizlikler nedeniyle size aşırı böülümlere ayrılmış sonuçlar verebilir.

Bu nedenle OpenCV, hangi vadi noktalarının birleştirileceğini ve hangilerinin birleştirilmeyeceğini belirleyen, işaretleyici tabanlı bir havza algoritması uygulamıştır.

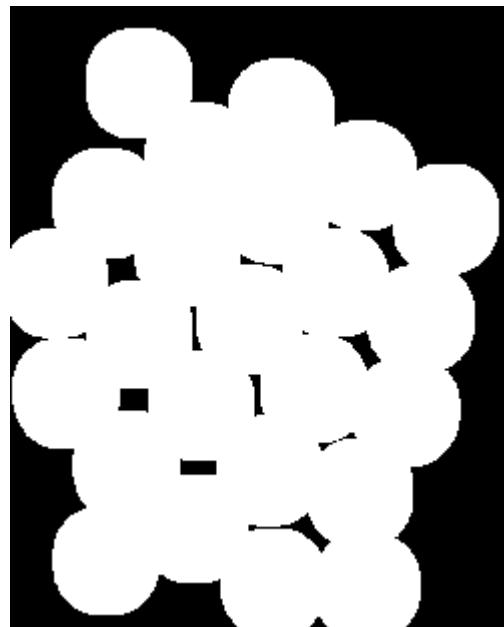
Bu algoritma etkileşimli bir görüntü segmentasyonudur. Burada yapılan ön plan veya nesne olduğundan emin olunan bölgeyi bir renk (veya yoğunluk) ile arka planda olduğundan veya nesne olmadığından emin olunan bölgeyi ise başka bir renkle ve son olarak hiçbir şeyden emin olunmayan bölgeyi de 0 ile etiketlemektir. Bu etiket işaretcidir.

Bu işlemin ardından havza algoritması uygulanır. Daha sonra işaretçi verilen etiketlerle güncellenecek ve nesnelerin sınırları -1 değerine sahip olacaktır.



Watershed

```
img = cv2.imread('coins.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
cv2_imshow(thresh)
```



Watershed

Görüntüdeki küçük beyaz gürültüleri kaldırmak için morfolojik açılımı kullanılır.

Nesnedeki küçük delikleri kaldırmak için de morfolojik kapatma kullanılır.

Böylece nesnelerin merkezine yakın olan bölgenin ön plan ve nesneden çok uzak olan bölgenin arka plan olduğunu kesin olarak bilinir.

Emin olunmayan tek bölge, madeni paraların sınır bölgesidir.

Watershed

Madeni para olduğundan emin olunan alanı çıkarmak gerekiyor. **Erozyon**, sınır piksellerini kaldırdığını biliyoruz. Yani kalanların madeni para olduğundan emin olunabilir.

Nesneler birbirine değmeseydi bu işlem işe yarayabilirdi ancak birbirlerine deðdikleri için, bir başka iyi seçenek de mesafe dönüşümünü bulmak ve uygun bir eşik uygulamak olacaktır.

Daha sonra bozuk para olmadığından emin olduğumuz alanı bulmamız gerekiyor. Bunun için sonucu genişletiyoruz.

Genişletme, nesne sınırını arka plana artırır. Bu şekilde, sınır bölgesi kaldırıldığından, sonuçta arka planda hangi bölgenin gerçekten bir arka plan olduğundan emin olabiliriz.

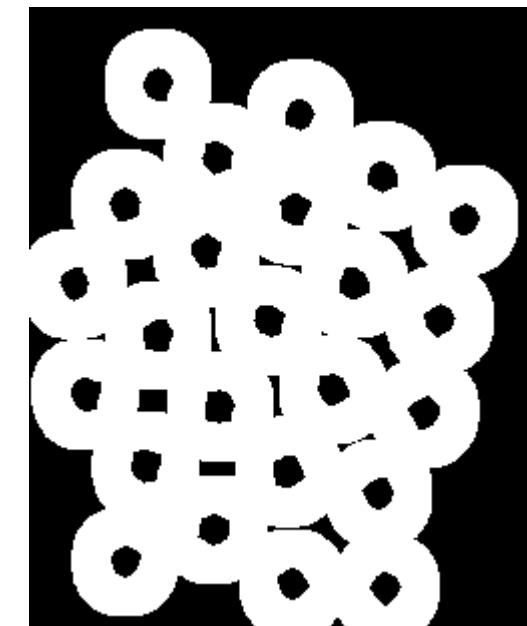
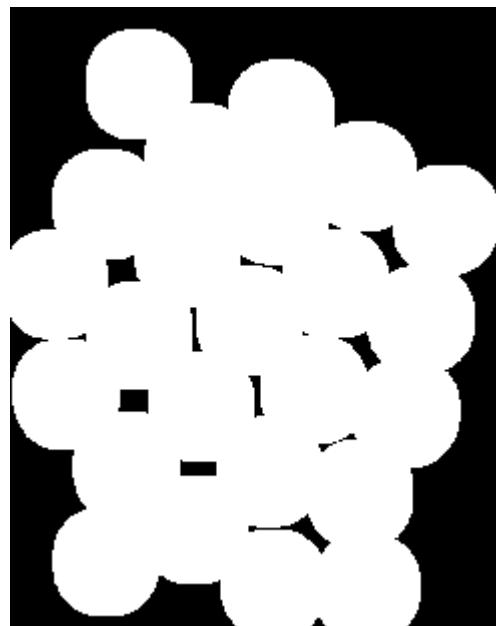
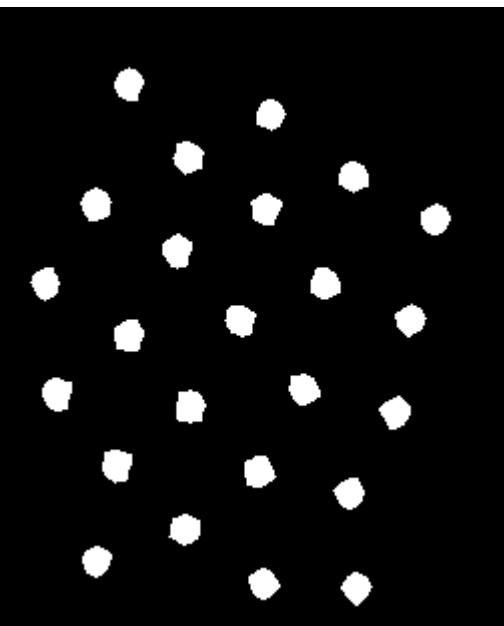
Watershed

```
#Gürültü temizleme
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)

# Kesin olunan arkaplan
sure_bg = cv2.dilate(opening,kernel,iterations=3)

# Kesin olunan ön plan
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)

# Kesin olmayan alan
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)
```





Watershed

Bu işlemin ardından etiketleme işlemi başlar.

Artık hangisinin madeni para bölgesi, hangisinin arka plan ve hepsi olduğunu kesin olarak bilindiği için, işaretleyici (orijinal görüntü ile aynı boyutta, ancak int32 veri tipine sahip bir dizidir) oluşturuyoruz ve içindeki bölgeleri etiketliyoruz.

Kesin olarak bildiğimiz bölgeler (ön plan veya arka plan) herhangi bir pozitif tamsayı, ancak farklı tamsayılarla etiketlenir ve kesin olarak bilmediğimiz alan sadece sıfır olarak bırakılır.

Bunun için cv2.connectedComponents() kullanılır. Görüntünün arka planını 0 ile etiketler, ardından diğer nesneler 1'den başlayarak tam sayılarla etiketlenir.

Watershed

```
ret, markers = cv2.connectedComponents(sure_fg)
markers = markers+1
markers[unknown==255] = 0
cv2_imshow(markers)
```

```
markers = cv2.watershed(img, markers)
img[markers == -1] = [255,0,0]
cv2_imshow(img)
```



Kaynaklar

- Gonzalez, R. C., & Woods, R. E., (2008), Digital Image Processing, 3rd Edition, Pearson International Edition, Pearson Education.
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html#contours-getting-started
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy.py_contours_hierarchy.html#contours-hierarchy
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_histograms/py_histogram_begin/py_histogram_begins.html#histograms-getting-started
- <https://www.cambridgeincolour.com/tutorials/histograms1.htm>
- https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html#histogram-calculation
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_watershed/py_watershed.html#watershed
- <https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html>