



KIRIKKALE ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ  
BİL1006-BİLGİSAYAR GRAFİĞİNE GİRİŞ

Doç. Dr. Serkan SAVAŞ

# Morfolojik Dönüşümler

---

- Erozyon (Aşındırma) (Erosion)
- Genişletme (Dilation)
- Açma (Opening)
- Kapama (Closing)
- Morfolojik Gradyan (Morphological Gradient)
- Top Hat
- Black Hat
- Kernel
- Canny Kenar Tespiti



# Morfolojik Dönüşümler

Matematiksel morfoloji, lineer olmayan komşuluk işlemlerinde güçlü bir görüntü işleme analizidir. Morfolojik dönüşümler, görüntü şekline dayalı bazı basit işlemlerdir.

Normalde ikili görüntüler üzerinde gerçekleştirilir.

İki girdiye ihtiyaç duyar; biri orijinal görüntü, ikincisi ise işlemin doğasına karar veren yapılandırma elemanı veya çekirdek olarak adlandırılır.

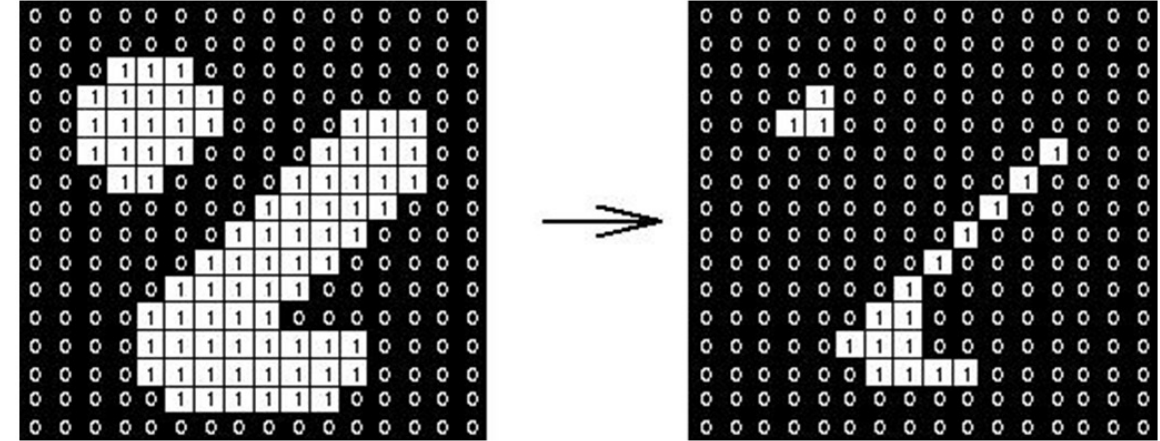
İki temel morfolojik operatör **Erozyon** ve **Genişletme**dir. Ardından Açılış, Kapanış, Gradyan vb. varyant formları da devreye girer.

# Aşındırma (Erosion)

İkili imgedeki nesneyi küçültmeye ya da inceltmeye yarayan morfolojik işlemdir.

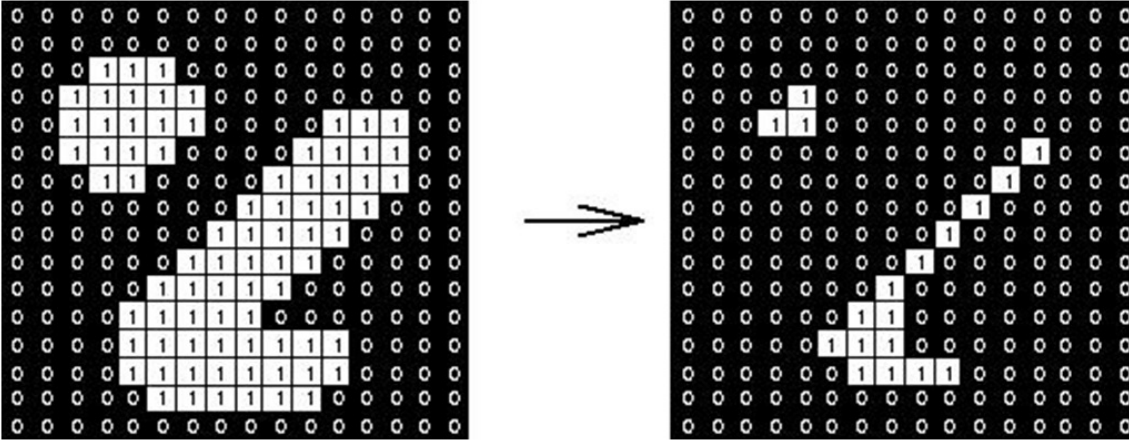
Aşındırma işlemi ile sayısal resim aşındırılmış olur.

Yani resim içerisindeki nesneler ufalır, delik varsa genişler, bağlı nesneler ayrılma eğilimi gösterir.



# Aşındırma

---



Aşındırma, toprak erozyonu gibidir, ön plan nesnesinin sınırlarını aşındırır.

Çekirdek 2B evrişimdeki gibi görüntü boyunca kayar.

Orijinal görüntüdeki bir piksel (1 veya 0), yalnızca çekirdeğin altındaki tüm pikseller 1 ise 1 olarak kabul edilir, aksi takdirde aşınır (sıfır yapılır).

# Aşındırma

Sonuçta sınırın yakınındaki tüm pikseller, çekirdeğin boyutuna bağlı olarak atılır.

Böylece görüntüdeki ön plan nesnesinin kalınlığı veya boyutu azalır. Başka bir deyişle sadece beyaz bölge azalır.

Küçük beyaz gürültüleri gidermek, birbirine bağlı iki nesneyi ayırmak vb. için kullanışlıdır.

# Aşındırma

```
[1] import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Erozyon

```
[2] img = cv2.imread('morp.png',0)
kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(img, kernel, iterations = 1)
```



```
plt.subplot(121), plt.imshow(img, cmap='gray'), plt.title('Orjinal')
plt.xticks([],plt.yticks([]))
plt.subplot(122), plt.imshow(erosion, cmap='gray'), plt.title('Erosion')
plt.xticks([],plt.yticks([]))
```

Orjinal



Erosion

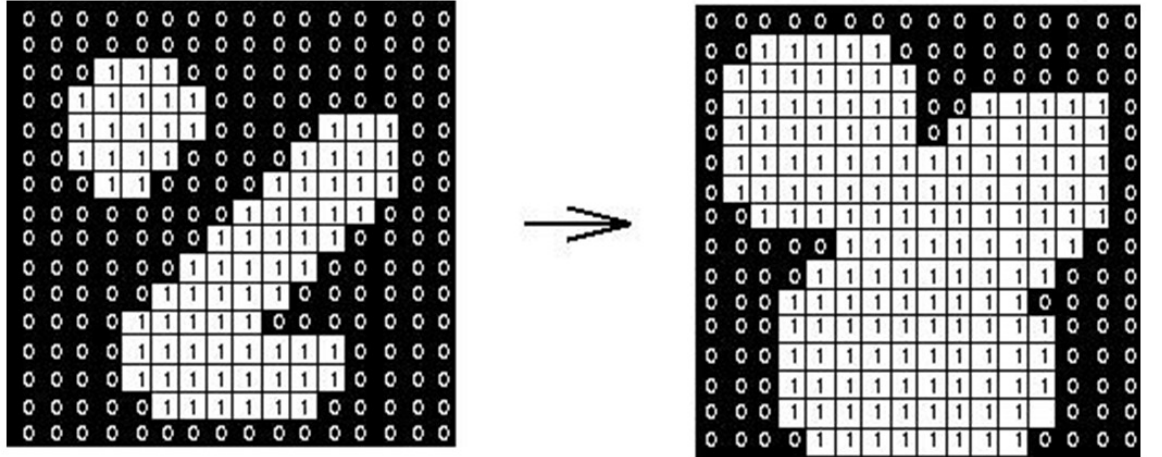


# Geniřletme (Dilation)

İkili imgedeki nesneyi büyütmeye ya da kalınlaştırmaya yarayan morfolojik işlemdir.

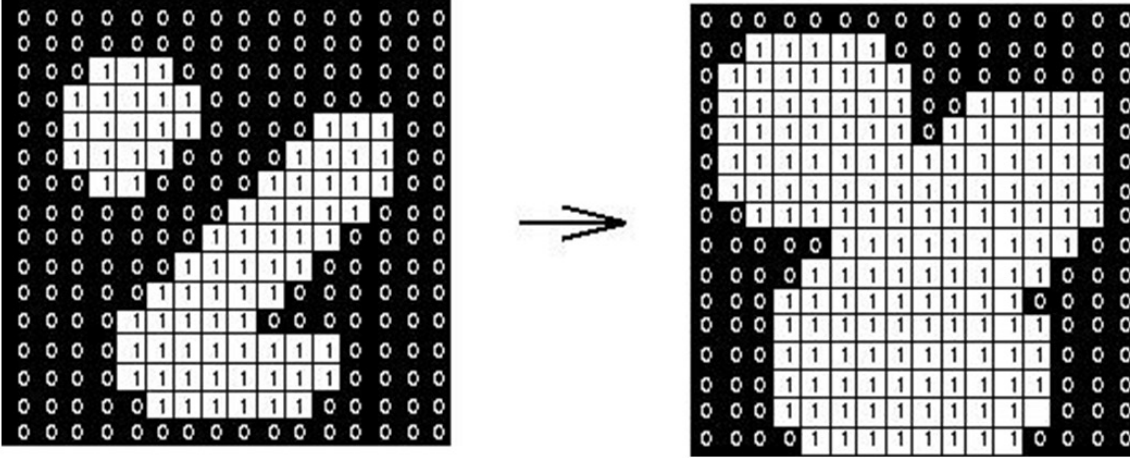
Sayısal bir resmi genişletmek resmi yapısal elemanla kesiştiği bölümler kadar büyütmek demektir.

Kalınlaştırma işleminin nasıl yapılacağını yapı elemanı belirler.





# Geniřletme



Çünkü erozyon beyaz sesleri ortadan kaldırır ama aynı zamanda nesneyi de küçültür.

Bu nedenle tekrar genişletme gerekir. Gürültü gittiği için geri gelmeyecektir ama nesne alanı artar.

Bir nesnenin kırık parçalarının birleştirilmesinde kullanılabilir.

Aşındırmanın tersidir. Burada, çekirdeğin altındaki en az bir piksel "1" ise bir piksel ögesi "1"dir.

Böylece görüntüdeki beyaz bölge artar veya ön plandaki nesnenin boyutu artar.

Normalde gürültü giderme gibi durumlarda aşındırmayı genişleme takip eder.

# Genişletme

```
img = cv2.imread('morp.png',0)
kernel = np.ones((5,5),np.uint8)
dilation = cv2.dilate(img,kernel,iterations = 1)
```

```
plt.subplot(121), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.xticks([],plt.yticks([]))
plt.subplot(122), plt.imshow(dilation, cmap='gray'), plt.title('Dilation')
plt.xticks([],plt.yticks([]))
```

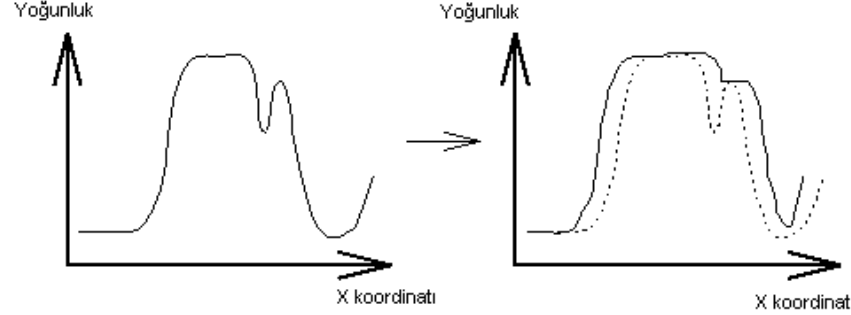
Original



Dilation



# Matematiksel Modelleri



$A \oplus B \leftrightarrow$  Genişletme

$A \ominus B \leftrightarrow$  Aşındırma

$A \circ B = (A \ominus B) \oplus B \leftrightarrow$  Açma işlemi

$A \bullet B = (A \oplus B) \ominus B \leftrightarrow$  Kapama işlemi

$$(f \oplus b)(x, y) = \max \{f(x - x', y - y') + b(x', y') \mid (x', y' \in D_b)\}$$

$$(f \ominus b)(x, y) = \min \{f(x - x', y - y') + b(x', y') \mid (x', y' \in D_b)\}$$

Genişletme işleminden sonra resim genelde daha parlaktır.

Aşındırma işleminden sonra resim genelde daha koyudur.

# Açma ve Kapama

$$f \circ b = (f \ominus b) \oplus b \leftrightarrow \text{Açma}$$

$$f \bullet b = (f \oplus b) \ominus b \leftrightarrow \text{Kapama}$$

# Açma İşlemi



Bu işlem, genişleme işlemi gerçekleştirdikten sonra yapılan aşındırma işlemidir. Görüntündeki gürültünün giderilmesinde faydalıdır. Bu işlemle birbirine yakın iki nesne görüntüde fazla değişime sebebiyet vermeden ayrılmış olurlar.



Burada **cv2.morphologyEx()** fonksiyonu kullanılır.  
(cv2.MORPH\_OPEN)

# Kapama işlemi

Açma işleminin tersidir. Aşındırma ve genişletme işleminin ardışıl uygulanmasıyla da kapama işlemi elde edilir.

Dolayısıyla birbirine yakın iki nesne görüntüde fazla değişiklik yapılmadan birbirine bağlanmış olur.

Ön plan nesnelerinin içindeki küçük deliklerin veya nesne üzerindeki küçük siyah noktaların kapatılmasında kullanışlıdır.



Burada **cv2.morphologyEx()** fonksiyonu kullanılır. (cv2.MORPH\_CLOSE)

# Açma

```
img2 = cv2.imread('morp-op.png',0)
op_kernel = np.ones((7,7),np.uint8)
opening = cv2.morphologyEx(img2, cv2.MORPH_OPEN, op_kernel)

plt.subplot(121), plt.imshow(img2, cmap='gray'), plt.title('Original')
plt.xticks([]),plt.yticks([])
plt.subplot(122), plt.imshow(opening, cmap='gray'), plt.title('Opening')
plt.xticks([]),plt.yticks([])
```

Original



Opening



# Kapama

```
img3 = cv2.imread('morp-cl.png',0)
cl_kernel = np.ones((7,7),np.uint8)
closing = cv2.morphologyEx(img3, cv2.MORPH_CLOSE, cl_kernel)
```

```
plt.subplot(121), plt.imshow(img3, cmap='gray'), plt.title('Original')
plt.xticks([],plt.yticks([]))
plt.subplot(122), plt.imshow(closing, cmap='gray'), plt.title('Closing')
plt.xticks([],plt.yticks([]))
```

Original



Closing





# Morfolojik Gradyan

Bir görüntünün genişlemesi ve aşınması arasındaki farktır.

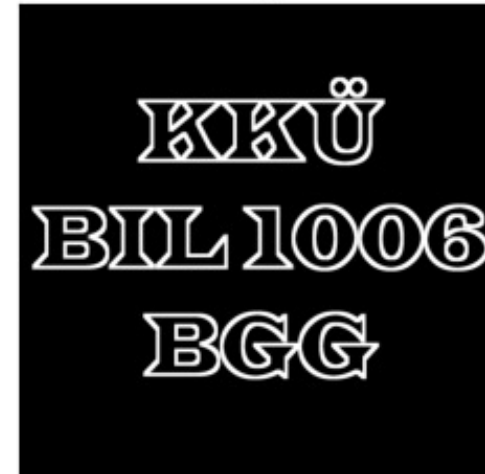
```
img_grd = cv2.imread('morp.png',0)
grd_kernel = np.ones((5,5),np.uint8)
gradyan=cv2.morphologyEx(img_grd, cv2.MORPH_GRADIENT, grd_kernel)
```

```
plt.subplot(121), plt.imshow(img_grd, cmap='gray'), plt.title('Orjinal')
plt.xticks([], plt.yticks([]))
plt.subplot(122), plt.imshow(gradyan, cmap='gray'), plt.title('Gradyan')
plt.xticks([], plt.yticks([]))
```

Orjinal



Gradyan



# Top Hat

Bu işlem **giriş görüntüsü ile görüntünün açılması arasındaki farktır**. Arka plandan farklı aydınlık seviyeli nesneleri araştıran gri seviyeli resimlerin segmentasyonunda kullanılan bir dönüşümdür.

Gri seviyeli morfolojik işlemler kullanılarak elde edilir. Tepe veya çukur bölgeleri belirginleştirme özelliğine sahiptir.



Aydınlık bölgeler için,

$$TopHat[A, B] = A - (A \circ B) = A - \max(\min(A))$$



Karanlık bölgeler için,

$$TopHat[A, B] = (A \bullet B) - A = \min(\max(A)) - A$$

# Top hat

```
img4 = cv2.imread('morp.png',0)
top_kernel = np.ones((7,7),np.uint8)
tophat = cv2.morphologyEx(img4, cv2.MORPH_TOPHAT, top_kernel)
```

```
plt.subplot(121), plt.imshow(img4, cmap='gray'), plt.title('Original')
plt.xticks([],plt.yticks([]))
plt.subplot(122), plt.imshow(tophat, cmap='gray'), plt.title('TopHat')
plt.xticks([],plt.yticks([]))
```

Original



TopHat



# Black Hat

Giriş ile giriş görüntüsünün kapanması arasındaki farktır.

Orjinal



BlackHat



```
img5 = cv2.imread('morp.png',0)
black_kernel = np.ones((7,7),np.uint8)
black_hat = cv2.morphologyEx(img5, cv2.MORPH_BLACKHAT, black_kernel)
```

```
plt.subplot(121), plt.imshow(img5, cmap='gray'), plt.title('Orjinal')
plt.xticks([],plt.yticks([]))
plt.subplot(122), plt.imshow(black_hat, cmap='gray'), plt.title('BlackHat')
plt.xticks([],plt.yticks([]))
```

# Çekirdek (Kernel)

Yapısal eleman olarak da adlandırılan kernel (çekirdek) istenilen boyutlarda ve istenilen şekilde hazırlanmış küçük ikilik bir resimdir.

Yapısal eleman çeşitli geometrik şekillerden biri olabilir; en sık kullanılanları kare, dikdörtgen ve dairedir.

1	1	1
1	1	1
1	1	1

0	1	0
1	1	1
0	1	0

0	1	0
0	1	1
0	0	0




# Çekirdek

Belirlenen kernel yapısı, birçok morfoloji işleminin gerçekleştirilmesinde en önemli öğedir.

Eğer morfolojik işlemin sonucunda resimdeki nesnelerin keskin hatları silinip yerlerine kavisli veya daha yumuşak hatlar getirilmek isteniyorsa dairesel yapısal eleman kullanılabilir.

Örneğin erozyon işleminde resim içerisindeki nesnelerin en ve boyları aynı oranda azaltılmak isteniyorsa yapısal eleman kare seçilmelidir.





## Çekirdek

Numpy yardımıyla manuel olarak bir yapılandırma elemanı oluşturulabilir. Örneklerde dikdörtgen şeklinde oluşturduk. Ancak bazı durumlarda eliptik/dairesel şekilli çekirdeklere ihtiyaç olabilir.

Bu amaçla OpenCV'nin **cv2.getStructuringElement()** adlı bir işlevi vardır. Sadece çekirdeğin şekli ve boyutu parametre olarak geçilir ve istenilen çekirdek elde edilir.

# Çekirdek

## Rectangular Kernel:

CR

## Elliptical Kernel:

`cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))`

## Cross-shaped Kernel:

`cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))`

```
rect_kernel= cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
elips_kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
cross_kernel =cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))
```

rect\_kernel

```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]], dtype=uint8)
```

elips\_kernel

```
array([[0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0]], dtype=uint8)
```

cross\_kernel

```
array([[0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0],
       [1, 1, 1, 1, 1],
       [0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0]], dtype=uint8)
```



# Canny

---

Canny Edge Detection, popüler bir kenar algılama algoritmasıdır. 1986 yılında John F. Canny tarafından geliştirilmiştir. Çok aşamalı bir algoritmadır.

1. Gürültü Azaltma
2. Görüntünün Yoğunluk Gradyanını Bulma
3. Maksimum Olmayanı İndirgeme
4. Histerezis Eşiği



# Canny (Gürültü azaltma ve yoğunluk gradyanı)

- Kenar algılama görüntüdeki gürültüye duyarlı olduğundan, ilk adım 5x5 Gauss filtresi ile görüntüdeki gürültüyü ortadan kaldırmak olacaktır.
- Düzleştirilmiş görüntü daha sonra yatay ve dikey yönde birinci türevi elde etmek için hem yatay hem de dikey yönde bir Sobel çekirdeği ile filtrelenir. Bu iki görüntüden, her piksel için kenar gradyanı ve yönü bulunabilir.

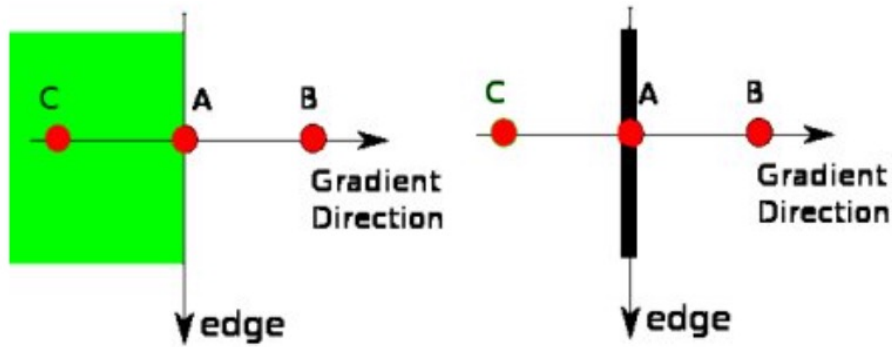
$$\text{Edge\_Gradient } (G) = \sqrt{G_x^2 + G_y^2}$$

$$\text{Angle } (\theta) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$

Gradyan yönü her zaman kenarlara diktir. Dikey, yatay ve iki çapraz yönü temsil eden dört açıdan birine yuvarlanır.

# Canny (Maksimum Olmayı İndirgeme)

- Gradyan büyüklüğü ve yönü alındıktan sonra, kenarı oluşturmayabilecek istenmeyen pikselleri çıkarmak için tam bir görüntü taraması yapılır. Bunun için her pikselde pikselin gradyan yönünde komşuluğunda yerel bir maksimum olup olmadığı kontrol edilir.



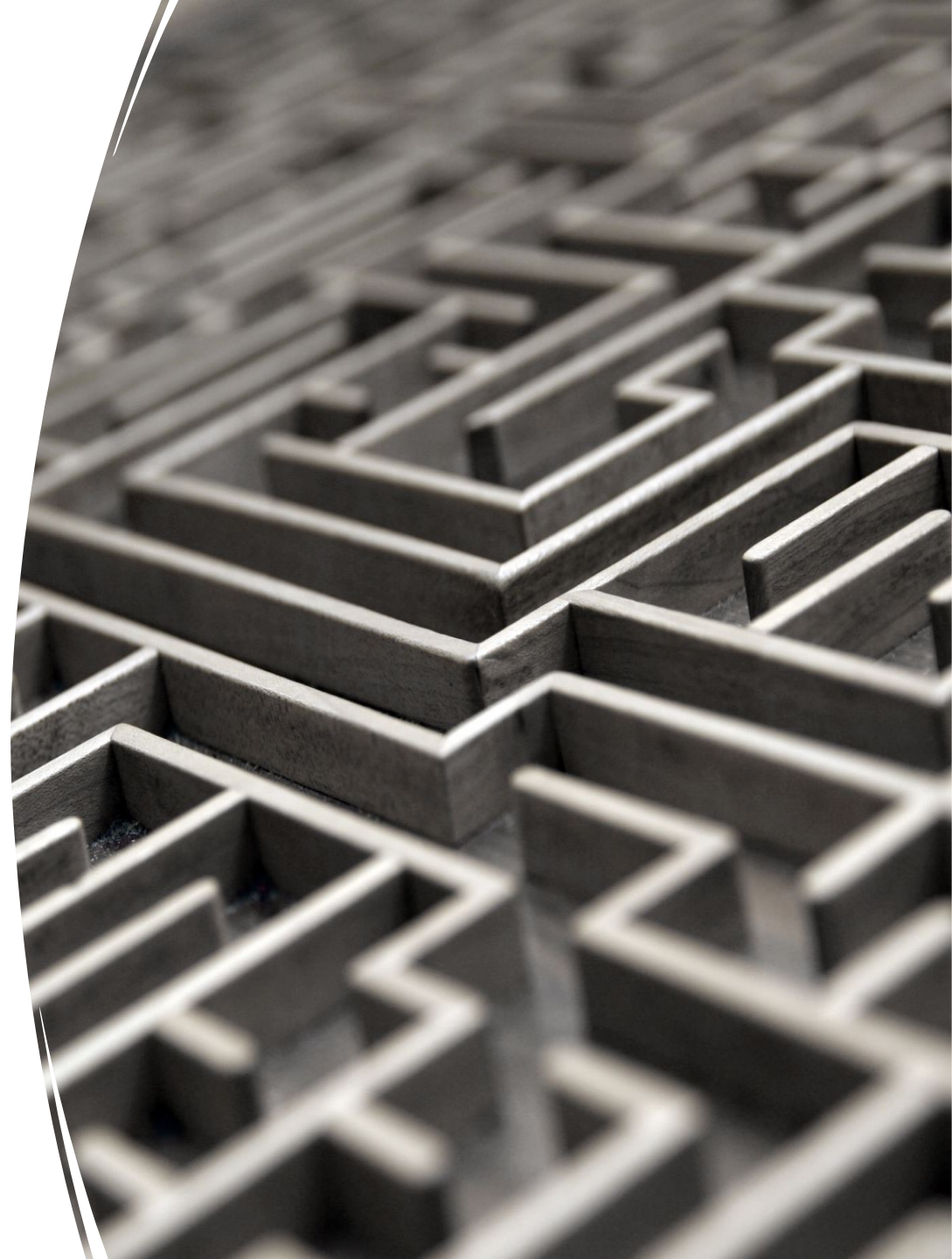
A noktası dikey yönde bir kenar. Gradyan yönü kenara diktir. B ve C noktaları gradyan yönlerindedir. Böylece A noktası, yerel bir maksimum oluşturup oluşturmadığını görmek için B ve C noktaları ile kontrol edilir. Eğer öyleyse, bir sonraki aşamaya geçiliyor, aksi takdirde sıfırlanıyor.

Burada sonuç olarak “ince kenarlı” ikili bir görüntü elde ediliyor.

# Canny (Histerezis Eşiğı)

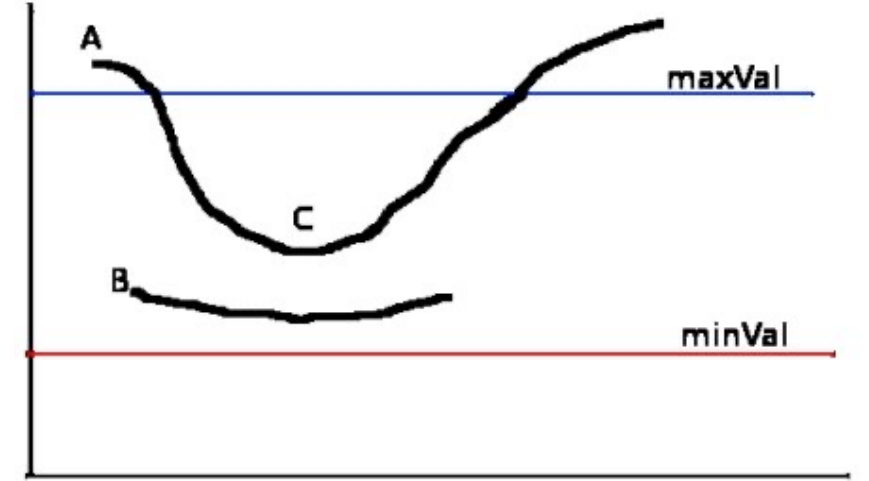
---

- Bu aşama, hangi kenarların gerçekten kenar olduğuna ve hangilerinin olmadığına karar verir. Bunun için minVal ve maxVal olmak üzere iki eşik değerine ihtiyaç duyulur.
- Yoğunluk gradyanı maxVal'den fazla olan kenarlar kesinlikle kenardır ve minVal'in altındakiler kenar değildir, bu nedenle atılır. Bu iki eşik arasında kalanlar, bağlantılarına göre kenarlar veya kenar olmayanlar olarak sınıflandırılır.
- "Kesin kenarlı" piksellere bağlanırlarsa, kenarların parçası olarak kabul edilirler. Aksi takdirde, onlar da atılır.



# Canny (Histerezis Eşiğı)

- A kenarı maxVal'in üzerindedir, bu nedenle "kesin kenar" olarak kabul edilir.
- C kenarı maxVal'in altında olmasına rağmen, A kenarına bağlıdır, bu da geçerli kenar olarak kabul edilir.
- Ancak B kenarı, minVal'in üzerinde olmasına ve C kenarı ile aynı bölgede olmasına rağmen, herhangi bir "kesin kenara" bağlı değildir, bu nedenle atılır.
- Bu yüzden doğru sonucu elde etmek için minVal ve maxVal'i buna göre seçmek çok önemlidir.
- Bu aşama ayrıca kenarların uzun çizgiler olduğu varsayımıyla küçük piksel parazitlerini de ortadan kaldırır.
- Sonunda görüntüdeki güçlü kenarlar elde edilir.





# OpenCV'de Canny Edge Detection

---

OpenCV, sayılan işlemlerin hepsini tek bir işleve yerleştirir, `cv2.Canny()`.

Burada ilk argüman, giriş resmidir. İkinci ve üçüncü argümanlar sırasıyla `minVal` ve `maxVal`'dir. Üçüncü argüman `apertür_boyutu`'dur.

Görüntü gradyanlarını bulmak için kullanılan Sobel çekirdeğinin boyutudur. Varsayılan olarak 3'tür. Son argüman, gradyan büyüklüğünü bulmak için denklemini belirten `L2gradient`'tir.

# Canny

---

Original Image



Edge Image



```
canny_img = cv2.imread('kku_logo.jpeg',0)
edges = cv2.Canny(canny_img,100,200)
```

```
plt.subplot(121),plt.imshow(canny_img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([], plt.yticks([]))
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([], plt.yticks([]))
plt.show()
```



# 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\\_morphological\\_ops/py\\_morphological\\_ops.html#morphological-ops](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html#morphological-ops)
- Boztoprak, H., Çağlar, M. F., & Merdan, M. (2007). Alternatif morfolojik bir yöntemle plaka yerini saptama, XII. *Elektrik, Elektronik, Bilgisayar, Biyomedikal Mühendisliği Ulusal Kongresi, Eskişehir, Kasım*.
- <https://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm>
- Gonzalez, R. C., & Woods, R. E., (2008), Digital Image Processing, Part 9, 3rd Edition, Pearson International Edition, Pearson Education.
- 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\\_canny/py\\_canny.html#canny](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_canny/py_canny.html#canny)