Open-CV ile Nesne Tespiti

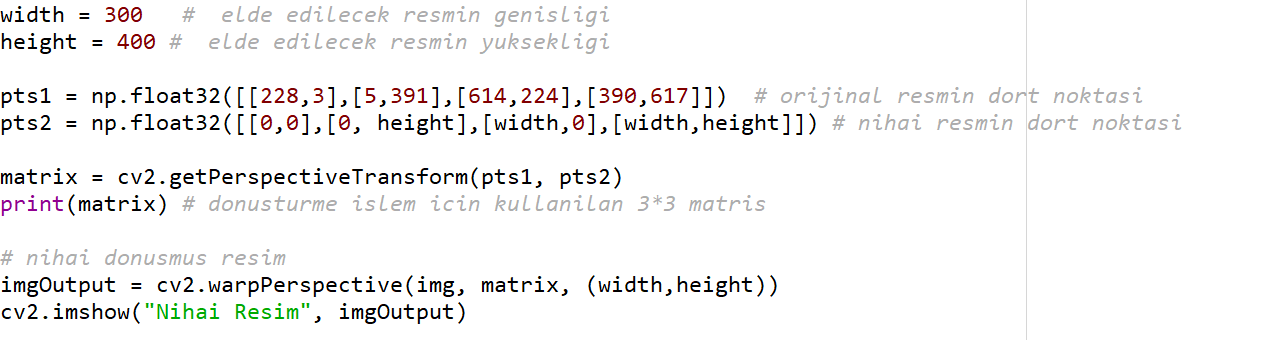
# OpenCv Görüntü İşleme

Bilgisayar Görüsü problemlerini çalışırken görüntülerin ya dosya olarak veya kameradan içe aktarılması gibi temel işlemler OpenCV komutlarıyla gerçekleştirilir. OpenCV temel komutlar olarak resmi/videoyu içe al, şeklini değiştir gibi komutların yanında nesne tespiti amaçlı kütüphanelere de (haar cascade) sahiptir.

## Perspektif Dönüştürme

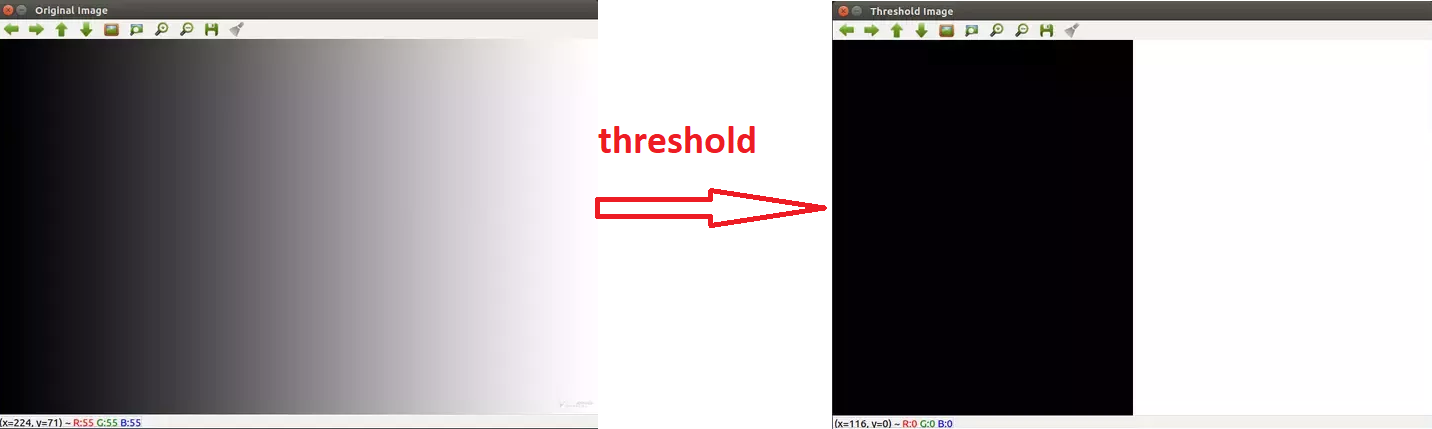
Bir görüntünün perspektifini değiştirmek gerekirse perspektif dönüştürme (perspective transformation/warping) kullanılır. Yamuk dikdörtgeni düzleştirmek bu tarz bir işlemdir.

Bu tür bir dönüştürme ilgili resmin dört noktasının belirlenerek ilgili şekle çevrilmesi şeklinde matris işlemleri ile gerçekleştirilir. Bu işlemleri bizim için OpenCV ortamında dönüşümü gerçekleştirilecek resme ait noktaların (ilgili şekilde dört nokta!) belirlenmesinden sonra perspectivetransform ve warpperspective komutları yardımıyla gerçekleştiririz. Bu dönüştürme aslında kamera görüşü olarak bilinen dönüştürmedir. Bir diğer ifade ile bir üç boyutlu cismin (perspective cismin duruşu mesafesinin algılanması ve iki boyutlu şekilde görünmesi transform) kamera da iki boyutlu görülmesidir.



## Threshold/Eşikleme

OpenCV de elimizde bir görüntüye ait piksellerin siyah beyaz iki gruba ayrılması olarak tanımlanır. Bir diğer deyişle grayscale bir resmi (0-255 arasında 256 farklı değer var!) sadece 0’lar ve 1’ler şeklinde tanımlanmış bir resme çevirmek olarak ifade edilir. Burada threshold-eşik bu ayrımı yapmak için kullanılan sınırı ifade eder. Bu sınırın altında kalan değeri bir sınıf, üstündeki değeri bir sınıf gibi düşünebiliriz.

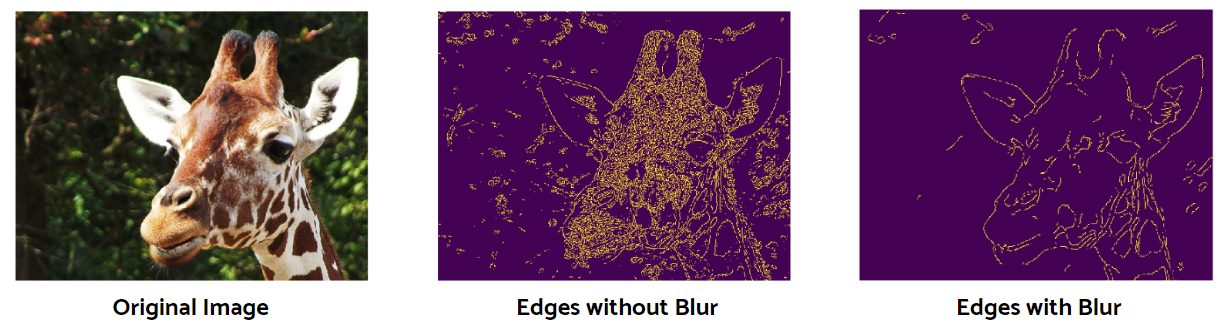


Bilgisayar görüsü için görüntü işlemede sık sık kullanılan “eşikleme” görüntünün bazı bölümlerini seçmek ayırmak -segmentasyon- gibi temel işlemlerde kullanılır. Daha somut bir ifadeyle resmin bir bölümünün arka plandan ayrılması gibi işlemler (özellikle tıbbi görüntülerde önemli!) için eşikleme kullanılır. Eşikleme bir sabit değerin tüm resme uygulanması şeklinde gerçekleşebileceği gibi adaptif veya işlenen resim boyunca değişken eşikleme kullanımı mümkündür.

## Blurring/Filtreleme/Gürültü Giderme

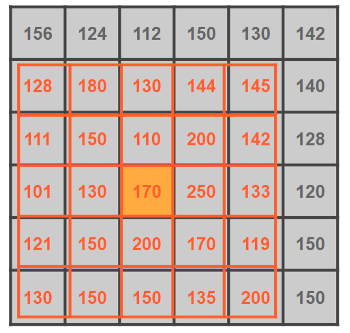
Blurring/Filtremele aracı örneğin kenar belirleme gibi bir amaç için başarımı etkileyen bir gürültü giderme yöntemidir. Daha net bir ifadeyle kenar veya kontur belirleme işlemleri gürültüden direkt etkilenen işlemlerdir. Bu bağlamda kenar/kontur belirlemede başarımı arttırmak thresholding, blurring ve morphological transformation adımları birlikte veya tek tek kullanılmalıdır.

Yüksel çözünürlüklü bir resimde kenar tespiti yapıldığında gereksiz kenarlar elde edilebilir. Blurring bu bağlamda gürültüyü azaltarak edge detection performansını arttırır. Ancak çok yüksek bir blurring uygulamasıda resimdeki veri kaybını arttırarak başarımı düşürebilir. Blurring ile edge detection başarımınının artmasını aşağıdaki şekilde görebiliriz.



OpenCV’deki başlıca blurring yöntemleri Averaging blurring, Gaussian blurring, median blurring ve bilateral filtering olarak bilinir. Matematiksel olarak dört yöntem de “convolution” matematiğine dayanır.

Average blurring seçilen bir kernel (pencere genişliği) üzerinde “convolution” uygulanarak elde edilen değerin merkeze yazılması şeklinde ilerler. Aşağıdaki şekilde 5\*5 lik bir kernel’un çalışması görülmektedir.

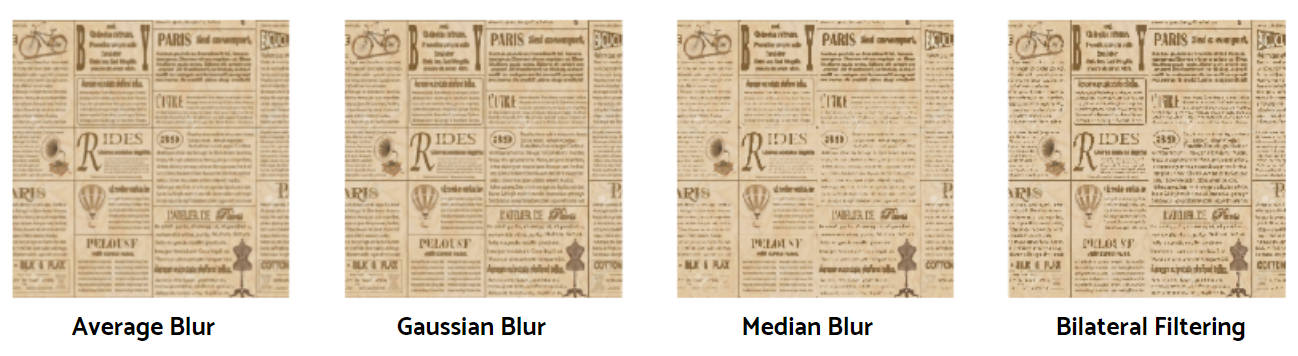


5\*5’lik piksel penceresinin convolution işleminden geçtikten sonra elde ettiği değeri merkezdeki hücreye yazar. Bu yaklaşımda pencere genişliğinin artması azalmasıyla etkisinin artıp azalacağı açıktır. kernel büyüdükçe resim daha çok bulanıklaşır!



Diğer yöntemler de kernel’ı işleme biçimleri ile farklılaşırlar. Genel olarak resimlerde “salt-pepper (tuz-biber, siyah-beyaz” olarak bilinen noktasal ani değişimler varsa median blurring, dağılım düzgünse gaussian blurring tercih edilebilir.

Aşağıda tüm blurring türleri yan yana karşılaştırılmaktadır.



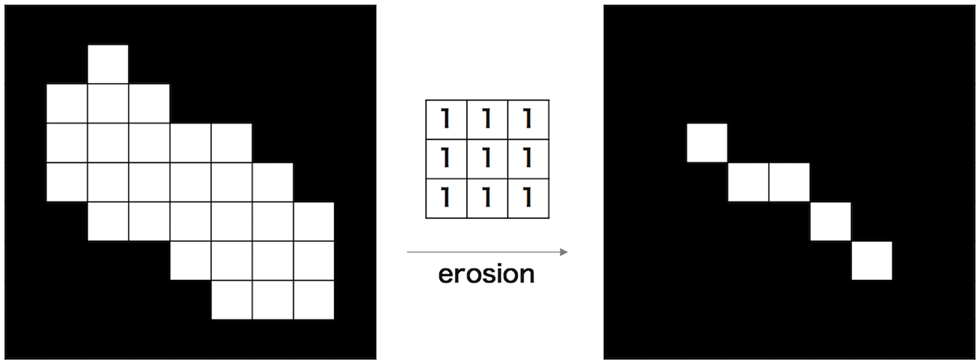
## Morfolojik (şekilsel, şekle ait) Görüntü İşleme Komutları

OpenCV kütüphanesinde en temel şekil işleme komutları erosion ve dilation olarak bilinmekte olup bu iki komutun türevleri olarak Opening, Closing, Gradient komutları kullanılır. Bu komutlar genel olarak binary (siyah-beyaz) resimlere uygulanırlar.

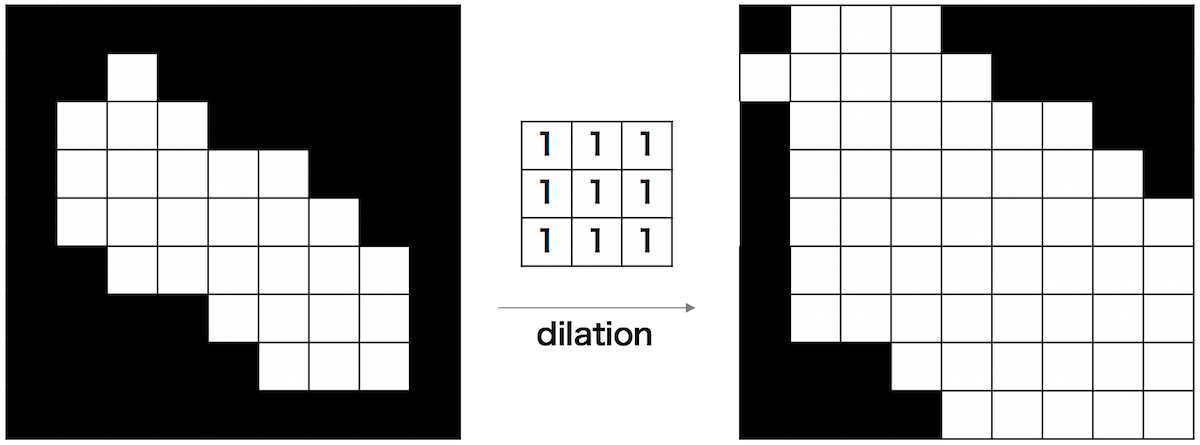
Erosion ile resimdeki hedef nesne zayıflar, güçsüzleşir, incelir. Çoğunlukla resimdeki beyaz gürültüleri ayırmada ve bağlı iki nesneyi birbirinden koparmada kullanılır.

Dilation ise erosion’un tersi olarak hedef nesneyi güçlendirir, kalınlaştırır. Genel olarak beyaz gürültüyü ayırırken (erozyonun amacı!) nesnemiz zayıflayacağı için “gürültüsü giderilmiş resimde” ilgilendiğimiz nesneyi yeniden belirginleştirmek için erosion un arkasından kullanılır.

Erosion hedef nesnenin kenarlarını bir kernel/pencere ile gezerek “hedef pikseli penceredeki convolution’un minimumu” ile değiştirmek şeklinde çalışır. Bir hedef piksel için kernel’un tüm piksel değerleri 1 ise hedef piksel 1 değerini alırken, diğer şekilde hedef piksel değeri 0 olur.



Dilation ise kabaca erosion’un tersi olup hedef nesneyi belirginleştirir. Yine bir kernel’ın resimde gezerken hedef piksele bu defa lokal maksimumu yazması şeklinde ilerler. Aşağıda yine 3\*3 bir kernel’ın dilation işlemini yerine getirmesi görülmektedir.



Erosion ve dilation’un kendine has avantaj ve dezavantajları vardır. Opening, closing ve gradient bu avantaj-dezavantajlara çözüm olarak geliştirilmiştir.

Örneğin erosion salt(tuz)-noise’u gidermede faydalı olurken aynı zamanda hedef nesne de çukurlar, aralıklar varsa erosion bunların derecesini arttırır. Eğer bu tarz çukur ve aralıkları hedef nesnenin kendisini bozmadan gidermek istersek o zaman erosion ve dilation işlemlerini ardışık olarak birleştiren (1: erosion; 2: dilation) opening işlemi yapılır.



Aşağıdaki şekilde opening ile gürültü (beyaz noktalar!) giderilip resmin orijinal hali korunmuş oluyor.



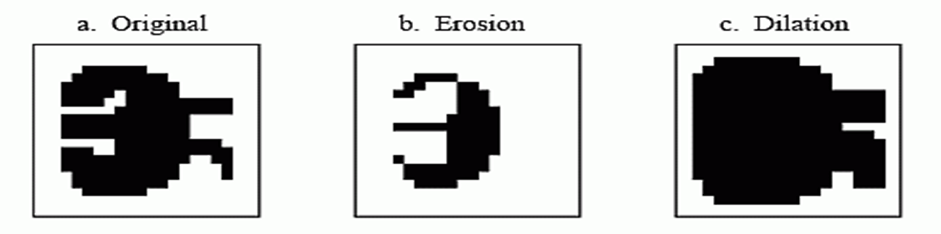
Closing ise aynı mantıkla önce dilation ve sonra erosion işlemlerini birleştiren bir yöntemdir. Nesnedeki delikler/aralıkları -nesneyi deforme etmeden- kapattığı için closing olarak bilinir. Aşağıdaki şekilde closing ile nesnedeki delikler (siyah noktalar) giderilip resim bozulmuyor.

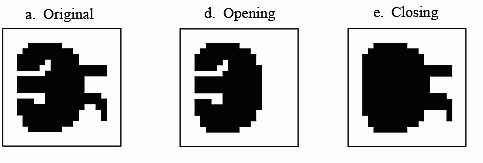


Gradient operatörü erosion ve dilation farkı olarak tanımlanır. Erosion ve dilation hedef nesne ile arka plan piksellerinin kesişme noktaları üzerinde çalıştığı için, gradient nesne kenar tespiti ve segmentasyon işlemlerinde kullanılır. Aşağıdaki şekilde gradient ile resmin sadece dış kenarları elde edildi.



Bilgisayar görmesinde genellikle bir resim/görüntü içindeki nesneleri ayırmak/tanımak gibi bir hedefimiz vardır. Buraya kadar anlatılan filtrelemeler ve morfolojik (şekil!) dönüştürmeleri bu amaca ulaşmak için kullanılır. Renkli veya gray-scale bir resim üzerinde bu tarz bir çalışmayı basitleştirmek için resim binary (siyah, beyaz) formata çevrilerek piksel değerleri siyah-0 ve beyaz-1 şeklinde iki değere indirgenir.

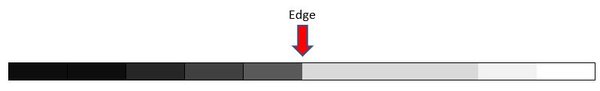




Bu tekniklerle gürültülü bir resimdeki yanlış pikseller temizlenir. Daha açık bir ifadeyle bir hedef nesnede delikler olmadığı biliniyorsa ancak nesne görüntüde delikli olarak elde edilmişse veya nesnenin kenarlarının düzgün olduğuna eminsek (opening değişimlerine dikkat edelim!) bunları opening ile giderebiliriz.

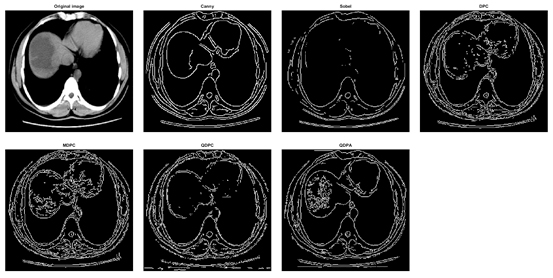
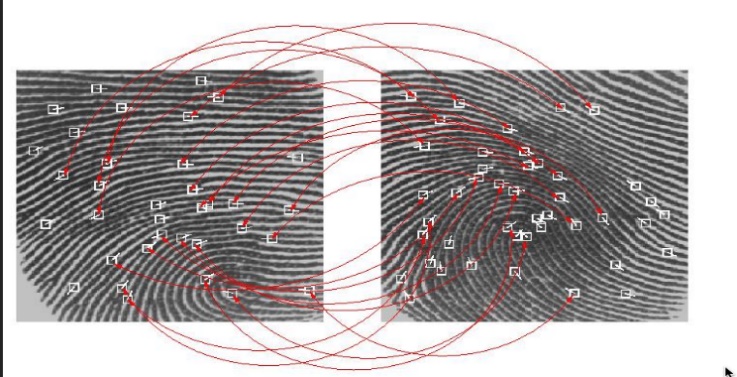
## Görüntü Gradienti

Bir görüntünün gradient’i işlemi genel olarak görüntüdeki nesnelerin kenarlarının tespitinde kullanılır.



Şekilin ortasında piksel yoğunluğunda ani bir değişim görülmektedir. Matematiksel olarak sobel gibi operatörler şekildeki pikseller üzerinde bir “kernel/matris” ile ilerlerken -yaptıkları hesaplama ile- işte bu yoğunluğun ani değişimlerini kenar olarak belirleyecektir.

doğrulama, medikal hastalık teşhisi ve plaka tanıma gibi sistemlerde belirgin kullanım alanına sahiptir.



Bu uygulamalar nesne yoğunluğunun (piksel değerlerinin değişimi) ani değişimlerini ön plana çıkararak geri kalan detayları ihmal eder. Görüntüdeki ani değişimler nesnenin kenarlarına işaret eder.

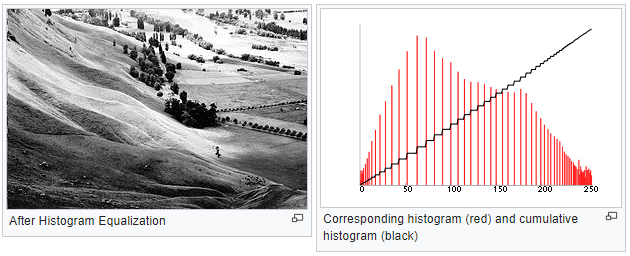
## Görüntü Histogramları

Bir görüntüdeki piksel değerlerinin yoğunluğunun -hangi pikselden kaç tane var- dağılımına histogram denir. Örneğin grayscale bir resim için düşünürsek.

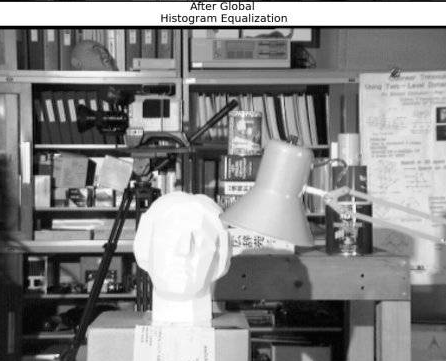
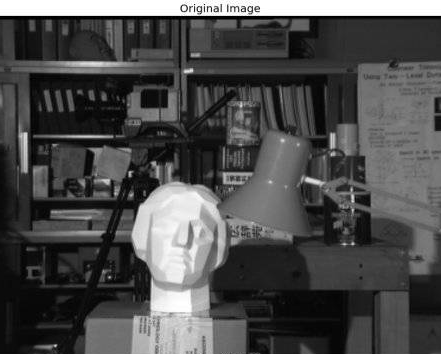


Resimdeki piksel yoğunluğunun ortalama da yığıldığını görmekteyiz.

Bu noktada histogram bulmanın aslında histogram eşitleme denilen bir işlem için elde edildiğini hatırlatalım. Histogram eşitleme histogram dağılımını (yukarıdaki ortalama değerlerde yığılmış!) daha dengeli hale getirerek düşük kontrastlı resmin kontrastını arttırmaya yarar.



Histogram eşitleme en yoğun piksel değerlerini tüm resme dağıtarak (yüksek kontrast değerlerini düşük kontrast değerlerini güncellemede kullanarak) gerçekleşir. Kümülatif kontrast dengelenirken, kırmızı dağılım da 0-255 arası piksel yoğunluğundaki dengelenmeyi gösterir.

Standart olarak OpenCv de “cv2.equalizeHist” komutu genel/global kontrast üzerinden işlem yapar. Ancak bazı durumlarda bu eşitleme resmin bazı bölgelerini geliştirirken bazı bölgelerini bozar

Global kontrast ile arka planın dağılımı iyileşse de, ön plandaki heykelde görüntü bozulmuştur. Bu tarz kayıpları engellemek için eşikleme de olduğu gibi resmin bütünü yerine bölüm bölüm yani adaptif olarak (Contrast Limited Adaptive Histogram Equalization-CLAHE) histogram eşitlemesi yapılır.

CLAHE, standart adaptive histogram eşitleme uygulamasında görülen aşırı kontrast üretme problemine çözüm olan bir versiyondur. CLAHE düşündüğümüz gibi resmin tamamına değil “tile” adı verilen alt bölümlere uygulanır ve daha sonra bu alt bölüm hesaplamaları birleştirilir. tileGridSize olarak bilinen bu parametre (varsayılan değeri 8\*8 lik matris!) dışında bu yöntemin ikinci parametresi kontrast değeri için kullanılan clipLimit adı verilen eşik değer parametresidir ve bu değer 2 veya 3 gibi seçilir.

# OpenCv Nesne Tespiti

Bir görüntü veya video da insan, hayvan, trafik ışığı, kanserli bölge, insan yüzü, araba gibi bölümleri tespit etmek bilgisayar görüsünde ve dolaylı olarak makine öğrenmesinde sürekli olarak kullanılan önemli tekniklerdendir.

Nesneleri kenarları, köşeleri, renkleri gibi özeliklerinden tespit etmeye çalışırız. Bu noktada nesneleri tespit için bize yardımcı olan literatürde “features” olarak geçen özellikler de vardır. Örneğin insan yüzü, kedi yüzü nesnelerin kendine has özellikleri vardır ve bunlar da nesne tespitinde kullanılırlar.

## Kenar Tespiti

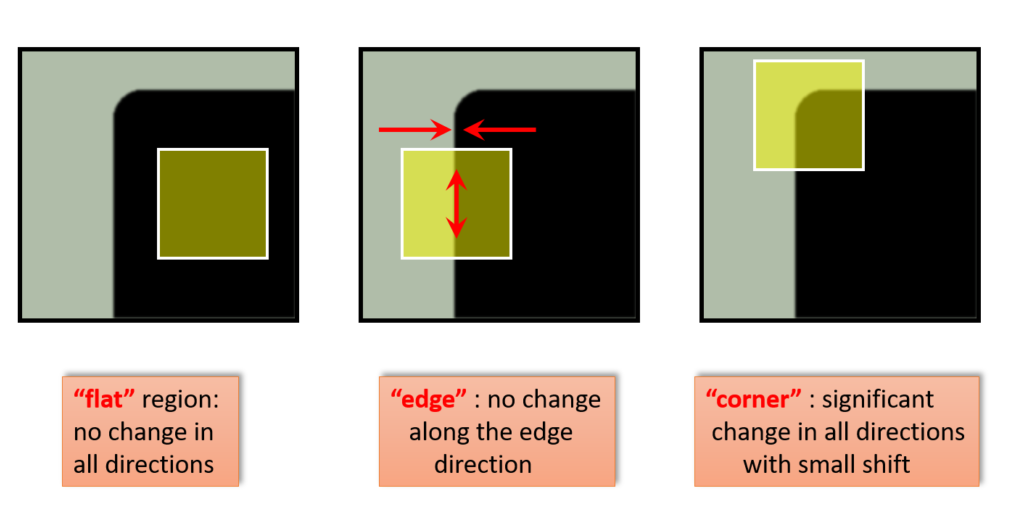
Örnek olarak güncel algoritmalardan “canny” algoritmasını çalışacağız. Bu kenar bulma algoritması gray-scale görüntüde çalışır. Algoritmanın iki temel parametresi olup bunlar düşük eşik ve yüksek eşik olmak üzere iki değerdir.

edges = cv2.Canny(image = img, threshold1 = low, threshold2 = high)

Literatürde eşik değerleri low: 255/3 ve high: 255 olarak alınabildiği gibi eşik değeri görüntüden hesaplanan “median” değeri üzerinden low = 0.7\*median ve high= 1.3\*median olarak kullanılabilir. Algoritmanın gereksiz bölümlerde kenar bulması sorunu ortaya çıktığından önce bir blur filtresi kullanılır ve sonra çalıştırılarak kenarlar gözlenir.

## Köşe Tespiti

Görüntülerdeki nesneleri kenarları ile ayırmaya çalıştığımız gibi köşeleri ile de ayırırız. Köşeler kenarlarda keskin değişimlerin (dönüşlerin) olduğu noktalardır. Köşe ayırmada yaygın kullanılan bir algoritma “Harris”’tir. Harris algoritması sobel kenar bulma algoritması ile çalışır. Bu nedenle yine nesne üzerinde ilerleyen kernel (3\*3, 5\*5) kenar ve köşeyi aşağıdaki şekilde bulur.



İlk planda hiçbir yönde değişim (gradient,türev, değişim diye bahsettiğimiz süreç!) yoktur ne kenar ne köşe tespit edilmiştir. İkinci aşamada kenar tespiti olup bu defa değişim sadece iki yönlüdür ve diğer iki yönde değişim yoktur. Son aşamada ise dört yönde bir değişim yakalandığı için buradaki pikseller köşe olarak tanımlanır.

cv2.cornerHarris(img, blockSize = 2, ksize = 3, k = 0.04)

Harris kenarları tespit ederek ilerler ve kenar tespitinde sobel'i kullandığı için sobel kernel size (ksize) seçimi yapılır. Kernel büyüdükçe kenarlar yumuşar.

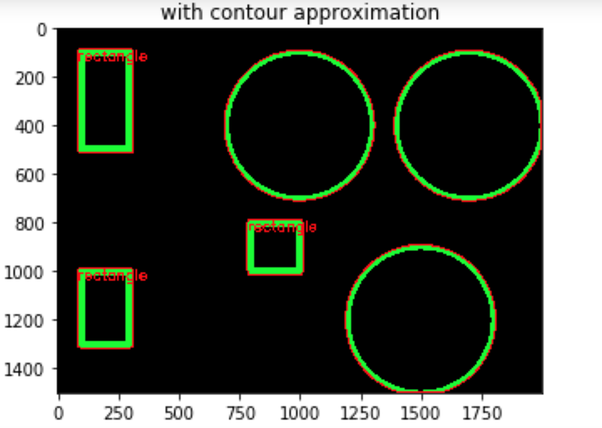
blockSize komşuluk oranı (genelde 2, kaç komşuya bakacağız hesaplama için!) ve k harris parametresi (genelde 0.04) olarak seçilir.

## Kontur Belirleme

Kontur belirli bir aralıktaki sürekli noktaları (aynı renk ve yoğunluğa sahip!) birbirine bağlayan eğri olarak tanımlanır. Kontur belirleme nesne tespiti ve nesne analizi için kullanılır. Kontur belirleme binary (siyah beyaz ! grayscale değil!) resimlerle çalışır bu nedenle resim siyah beyaz değilse threshhold ile binary hale gelir.

Edge detection ve contour detection birbirine benzer iki kavramdır. İkisi arasındaki temel fark edge detection elde edilen kenarların kapalı birbirine bağlı eğriler olmasına gerek yokken zaten kenar bulma algoritmaları üzerine yürüyen contour bulma algoritmalarında elde edilen noktalar birbirine tam bağlı -kapalı- eğriler oluşturur.

Contour detection ise nesnenin etrafında mutlaka kapalı eğriler oluşturur. Ve ek olarak bu nesenelerin etrafına renkli hatlar (kontur!) çizer. Böylece konturları bulunmuş nesnelerin şekillerini (3 kare, 1 daire) saymak ve bir görüntüden bir nesneyi almak gibi işlevler gerçekleştirilir.



# Renk İle Nesne Tespiti

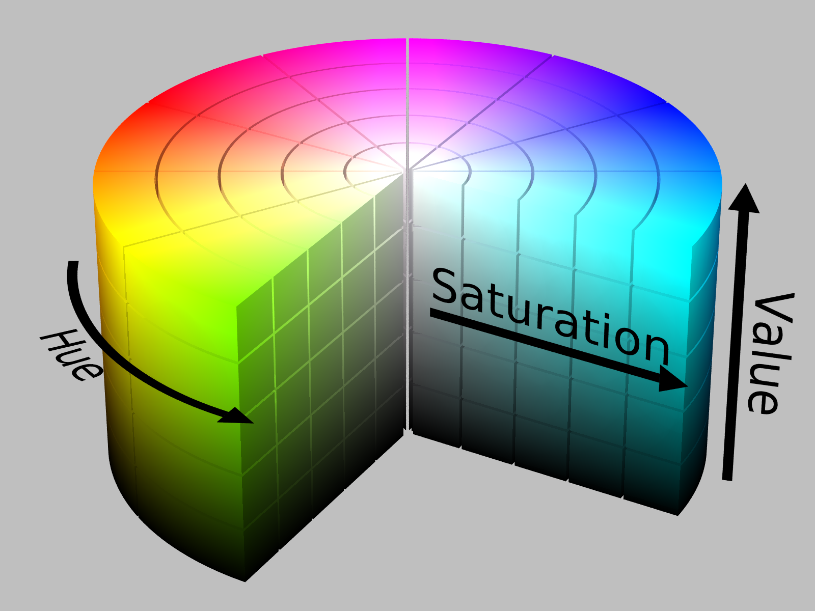
## Rgb Yerine Neden Hsv

rgb, bgr yerine neden hue, saturation value (hsv) kullanılıyor. rgb bileşenleri nesneye çarpan ışık miktarına çok bağımlıdır o nedenle de nesne tespitinde zorlanır. hsv insan görmesine daha yakın olur.

hue: rengi tanımlar 0 kırmızı olup 0-255 arasında değerler alır.

saturation: 0 (cansız, soluk) ve 255 (en canlı) değerlerini alır.

value : parlaklık (0 kapkaranlık) ve (255 en aydınlık)



# Nasıl Çalışır

İlk önce tespit edilecek rengin low ve high Hsv değerlerini verilir. Sonra verilen videoda gerekiyorsa görüntü işlenir ve sonra görüntüden low ve high değerleri ile bir maske oluşturulur. Oluşan maskede gerekiyorsa Morfolojik işlemler uygulanır ve Maske kenarları konturlanarak orijinal görüntüye çizdirilir