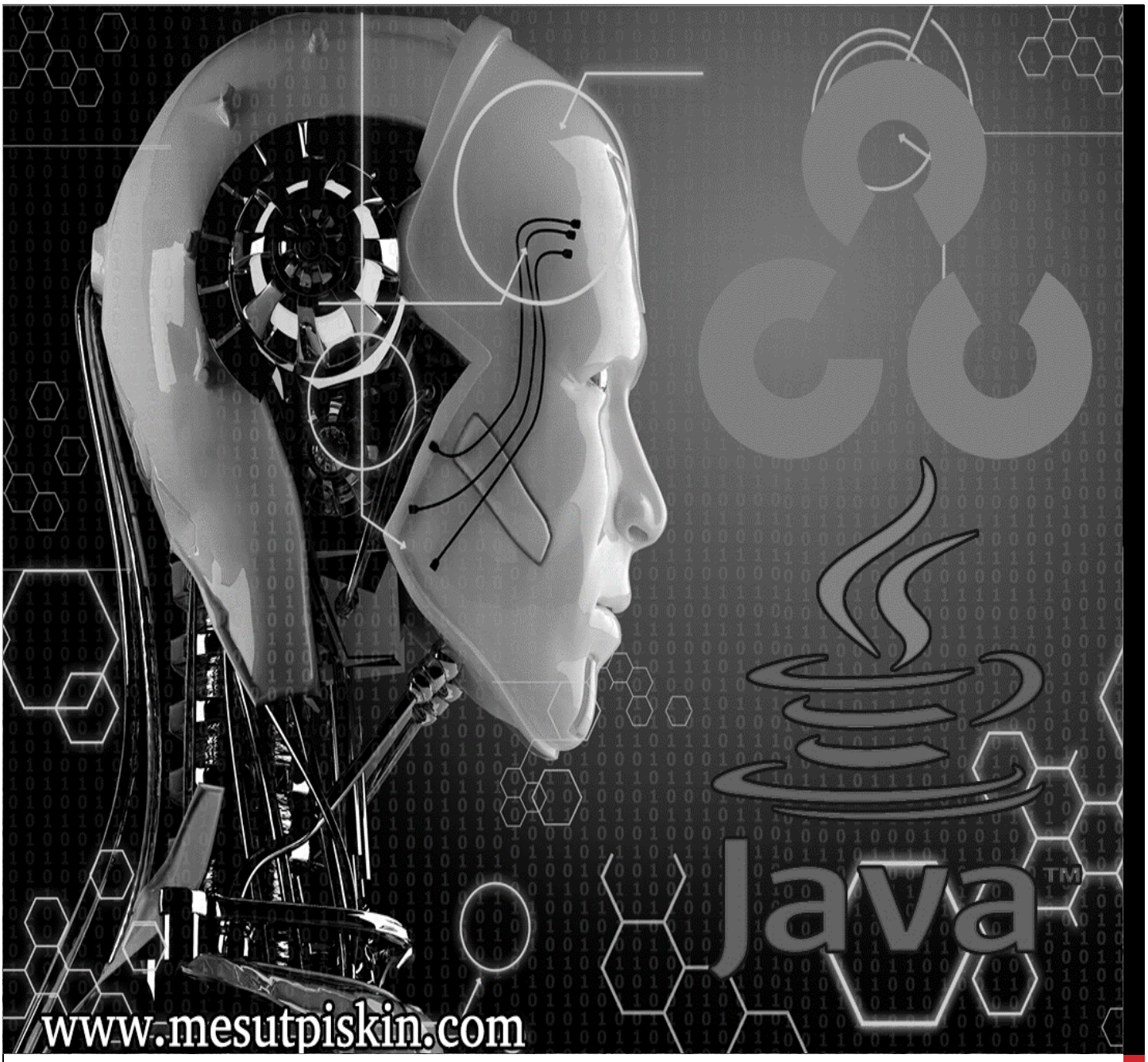


OPENCV İLE GÖRÜNTÜ İŞLEME

MESUT PİŞKİN

OPENCV İLE GÖRÜNTÜ İŞLEME

MESUT PIŞKIN

GİRİŞ

Bu kitap mesutpiskin.com/blog adresindeki blog yazılarımın derlenmesi ile oluşturulmuştur. Düzenleme, ekleme, görüş veya önerileriniz için e-posta adresimden benim ile iletişime geçebilirsiniz.

Yer alan örnekler çoğunlukla Java ile anlatılmış yeri geldiğinde ise Python örnekleri eklenmiştir. Daha önce görüntü işleme ile uğraşmamış veya farklı kütüphaneleri kullanmış OpenCV öğrenmek isteyenlere yöneliktir. Temel kavramlardan başlayarak birçok kavram ve algoritma ele alınmıştır.

Yer alan örnek uygulamalar gerek OpenCV 3.1 gerekse 2.4.x sürümleri kullanılarak geliştirilmiştir. Gereken yerlerde sürümler arası farklılıklara değinilmiştir.

mesutpiskin@outlook.com

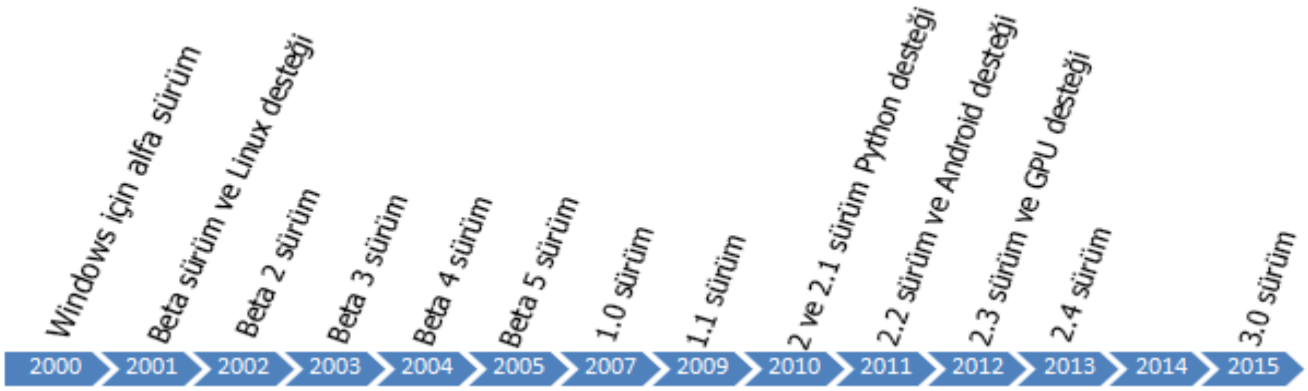
İÇİNDEKİLER

OPENCV NEDİR?

OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir. Open source yani açık kaynak kodlu bir kütüphanedir ve BSD lisansı ile altında geliştirilmektedir. BSD lisansına sahip olması bu kütüphaneyi istediğiniz projede ücretsiz olarak kullanabileceğiniz anlamına gelmektedir. OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir.

OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve makine öğrenmesine (machine learning) yönelik 2500'den fazla algoritma bulunmaktadır. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme, görüntü karşılaştırma, optik karakter tanımlama OCR (Optical Character Recognition) gibi işlemler rahatlıkla yapılabilmektedir.

Not: OpenCV geliştirici Itseez firması Intel tarafından satın alındı. OpenCV geliştirmesine Intel çatısı altından devam edeceğini duyurdu.



Yıllara göre OpenCV sürümleri

OpenCV Bileşenleri

OpenCV kütüphanesini daha iyi anlamak için mimarisinden ve OpenCV'yi oluşturan bileşenlerden bahsetmek istiyorum.

Core: OpenCV'nin temel fonksiyonları ve matris, point, size gibi veri yapılarını bulundurur. Ayrıca görüntü üzerine çizim yapabilmek için kullanılabilecek metotları ve XML işlemleri için gerekli bileşenleri barındırır.

HighGui: Resim görüntüleme, pencereleri yönetme ve grafiksel kullanıcı arabirimleri için gerekli olabilecek metotları barındırır. 3.0 öncesi sürümlerde dosya sistemi üzerinden resim dosyası okuma ve yazma işlemlerini yerine getiren metotları barındırmaktaydı.

Imgproc: Filtreleme operatörleri, kenar bulma, nesne belirleme, renk uzayı yönetimi, renk yönetimi ve eşikleme gibi neredeyse tüm fonksiyonları içine alan bir pakettir. 3 ve sonra sürümlerde bazı fonksiyonlar değişmiş olsada 2 ve 3 sürümünde de birçok fonksiyon aynıdır.

Imgcodecs: Dosya sistemi üzerinden resim ve video okuma/yazma işlemlerini yerine getiren metotları barındırmaktadır.

Videoio: Kameralara ve video cihazlarına erişmek ve görüntü almak ve görüntü yazmak için gerekli metotları barındırır. OpenCV 3 sürümü öncesinde bu paketteki birçok metot video paketi içerisindeydi.

Tüm OpenCV modülleri için http://docs.opencv.org/3.0-beta/modules/refman.html adresine göz atabilirsiniz.

Alternatif Görüntü İşleme Kütüphaneleri

Görüntü işleme projelerinizde kullanacağınız kütüphaneyi amacınıza uygun olarak seçmeniz önemlidir. Bu seçimi yaparken ne yapmak istediğinize doğru karar vermelisiniz, örneğin sadece kameradan (usb, ip vs.) görüntü almak için projenize OpenCV entegre etmenize gerek olmayabilir. Bu gibi durumlar için ve OpenCV'nin neden iyi olduğunu anlayabilmek amacıyla alternatif olarak görüntü işleme kütüphanelerine de bakalım.

MATLAB: Matlab için bir görüntü işleme kütüphanesi olarak bahsetmek doğru değildir fakat içerisinde görüntü işlemeye yönelik temel algoritmaları barındırmaktadır.Dördüncü nesil ve çok amaçlı bir programlama dilidir. Akademik araştırmalarınızda, performansın önemli olmadığı durumlarda temel görüntü işlemleri için tercih edebilirsiniz.Matlab kullanarak OpenCV Kütüphanesi ile etkileşimli olarak da uygulamalarda geliştirmek mümkündür.

Halcon: Endüstriyel projeler için tercih edilen, kendi içerisinde geliştirme ortamının yanı sıra çeşitli programlama dilleri (C, C++, VS C++, C#, VB.NET) için kütüphanesi bulunan, yapay görme (machine vision) odaklı ticari bir yazılımdır. İçerisinde birçok hazır fonksiyon bulundurur bu sayede hızlı uygulamalar geliştirilebilir. OpenCV açık kaynak kodlu, ücretsiz bir kütüphanedir ve computer vision odaklıdır. Bu yönleri ile Halcon'dan ayrılmaktadır.

OpenFrameworks: Açık kaynak olarak geliştirilen bu kütüphane C++ programlama dili için geliştirilen bu proje OS X, Linux, Embedded Linux (ARM), iOS, Android platformlarında çalışabilmektedir. OpenCV kütüphanesinin birçok algoritmasını kullanır ve temel çıkış amacı kolay ve hızlı uygulama geliştirmektir. Örneğin OpenCV ile 2t sürede gerçekleştirdiğiniz bir işi 1t sürede gerçekleştirebilirsiniz, bunun temel sebebi ise bir çok fonksiyonu aracılığıyla standart hale getirilmiş olan işleri tek satır ile yapabilmesidir (Nesne tespiti,takibi renk belirleme, karşılaştırma vb.).

CIMG: Açık kaynak kodlu bir görüntü işleme kütüphanesidir. Windows, Linux ve OS X platformu üzerinde çalışmaktadır. Sadece C++ dili için desteği bulunmaktadır fakat yazılmış wrapperlar ile Java ve Python ile de uygulama geliştirilebilmektedir. Birçok algoritmayı barındırmaktadır fakat OpenCV kadar performanslı ve geniş bir algoritma altyapısına sahip değildir.

Fiji: Java platformu için geliştirilmiş açık kaynak kodlu GPL lisansına sahip bir görüntü işleme kütüphanesidir. Windows, Linux ve MAC OSX Intel 32-bit veya 64-bit üzerinde çalışır. Bilimsel görüntü analizi için geliştirilmiştir. Genetik, hücre biyolojisi, nöro-bilim gibi alanlar için özelleştirilmiş algoritmalara sahiptir.

Endrov, ImageJ, Lead tools, Pink, Image Magick, Boost ise görüntü işleme kütüphanelerinden bazılarıdır.

NEDEN OPENCV? NEDEN JAVA?

Neden OpenCV?

OpenCV görüntü işleme kütüphaneleri arasında en popüler ve en çok kullanılanıdır. 2016 verileri itibari ile OpenCV kütüphanesinin toplam indirme sayısı 7 milyonu geçmiştir. Yazılım geliştiriciler için bir kütüphanenin, teknolojinin popülerleşmesinin temel sebebi o teknoloji hakkındaki erişilebilecek kaynak çeşitliliğidir. OpenCV geniş bir kaynağa sahiptir, yapmak istediğiniz şeyle alakalı olarak size yardımcı olacak topluluklar ve bulabileceğiniz teknik dokumanlar oldukça fazladır. Bilişim sektöründe kullanım oranı fazla olan bütün programlama dillerine desteği bulunmaktadır, açık kaynak kodlu olması itibari ile de doğrudan desteği bulunmayan programlama dilleri için ara katmanlar yazılmış ve OpenCV bu dile entegre edilmiştir. Geniş işletim sistemi desteği bulunması itibariyle de geliştiriciler için, platformlar arası uygulama geçişini kolaylaştırmaktadır. Aktif olarak OpenCV kullanan bazı projelere göz atarsak neden en iyisi olduğunu daha iyi anlayabiliriz. Google tarafından cadde ve sokakları haritalamak amacıyla yürütülen street view projesi, NASA tarafından Marsa gönderilen keşif aracı (Curiosity) ile Mars yüzeyini görüntülemek, yorumlamak ve aracın bazı hareketlerini otonom olarak yapabilmek için OpenCV kullanılmıştır.

Neden Java?

Görüntü işleme yazılımları için performans önemli bir kriterdir. Bu yazılımlar yüksek işlemci, grafik kartı ve bellek ihtiyacı duyarlar bu yüzden kullanılan programlama dili ve seçilen platform yazılımın performansı için kritik öneme sahiptir. Daha önceleri görüntü işleme projelerinde C ve C++ programlama dilleri kullanılmaktaydı çünkü bu diller düşük seviyeli olması sebebiyle rakiplerine göre oldukça performanslıydı. Programlama dillerinin çeşitliliğinin artmasıyla birlikte bir çok dil açık kaynak oldu ve toplulukların veya firmaların desteği ile yapılan iyileştirmeler sayesinde performanslarında iyileştirilmeler yapıldı. Java programlama dili JVM'in (Java Virtual Machine) geliştirilmesiyle yazılan kodu daha iyi optimize edebilir hale geldi ve C++ programlama dili kadar performanslı çalışan kodlar üretmeye başladı. Java programlama dilinin platform bağımsızlığı, orta seviye bir dil olması ve proje geliştirmenin oldukça hızlı olması sebebiyle önce çıkmaktadır. Web, mobil, gömülü sistem ve masaüstü projelerin geliştirilebiliyor olması da başlı başına bir seçim nedeni de olabilir. Bu bağlamda OpenCV uygulaması geliştirmede Java tercih edilmektedir. OpenCV dokümanlarına bakıldığında örnekler C++, Java ve Python ağırlıklıdır. Son birkaç yıldır Python yazımının kolay olması, uygulama geliştirme süresini kısaltması gibi özellikleri ile programlamaya yeni başlayan kişiler tarafından kullanılarak popüler olsa da performans, büyük proje geliştirmenin zorluğu gibi sebeplerle de projelerde tercih edilmemektedir, bu durum şimdilik böyle olsa da aldığı destekler ve toplulukların katkıları ile görüntü işlemede tercih edilen bir dil haline gelebilir.

OPENCV WRAPPERS

Wrapper Türkçe kelime anlamı olarak sarıcı, sarmalayıcı manasına gelmektedir. Ticari yada açık kaynak olarak geliştirilen, OpenCV kütüphanesini referans alan ve bu kütüphane içerisindeki fonksiyonları kullanarak kendi fonksiyonlarını geliştiren ve farklı platformlarla kullanılabilir hala getiren yazılım kütüphaneleri diyebilirz. Bildiğiniz üzere OpenCV açık kaynak kodlu bir yapıdadır ve tüm programlama dillerine doğrudan bir destek vermemektedir. Bu programlama dillerinden başlıcaları C#, Visaul Basic .Net, F# Ruby vb. dir. Doğrudan desteği olan programlama dilleri için de yazılmış wrapper'lar mevcuttur. Temel yazılış amaçları desteği olmayan programlama dilleri içinde bu kütüphaneyi kullanılabilir hale getirmek, olan fonksiyonları belirli bir platformda daha iyi çalışabilecek hale getirmek amacıyla optimize etmek veya bu kütüphanenin kullanımı kolaylaştırarak sadece belirli bir amaç için özelleştirmekdir. Bu kütüphanelerede değinmekte fayda olduğunu düşünüyorum ve yazılmış bazı wrapper'lara göz atalım.

EmguCV: Bu wrapper .Net framework çatısı altında bulunan C #, VB, VC++, Xamarin veya IronPython ve Unity ile görüntü işleme uygulamaları geliştirmeyi kolaylaştırmaktadır. Windows, Linux, Mac OS X, iOS, Android ve Windows Phone platformlarında çalışabilmektedir. Güncelliğini devam ettiren bir kütüphanedir ve farklı lisanslamalara sahiptir.

JavaCV: Java teknolojisi çatısı altında kullanmak için geliştirilmiş bir wrapper'dır. OpeCV kütüphanelerini referans alır ve java içerisinde C++ yazımında (syntax) uygulama geliştirmeyi destekler. Sıklıkla kullanılan bir çok algoritmayı kullanılabilirlik açısından kolaylaştırmışlardır. Sadece OpenCV değil FFmpeg, libdc1394, PGR FlyCapture, OpenKinect, videoInput, ARToolKitPlus, ve flandmark gibi kütüphaneleri de kullanmaktadır. Bytedeco tarafından açık kaynak kod olarak geliştirilmektedir ve güncelliğini devam ettiren bir kütüphanedir topluluk desteği bulunmaktadır.

Opencvsharp: .Net framework için yazılmış bir başka kütüphanedir. .Net dilleri için görüntü işleme yazılımları geliştirebilmeyi amaçlamaktadır. EmguCV'den farklı olarak açık kaynak kodlu olduğu için eksra bir lisans maliyeti yoktur. .Net framework 2.0 ve üzeri ile geliştirilen projeler için kullanılabilir, mono desteği ile Linux ve MacOS gibi platformlar için de uygulama geliştirilebilir. Shimat tarafından geliştirilmektedir ve güncelliğini sürdüren bir kütüphanedir.

EHE-LAB OpenCV Wrapper: Labview için geliştirilmiş bir OpenCV wrapper'dır. OpenCV 2.4.9 sürümü referans alınarak geliştirilmiştir. Ticari bir üründür ve 150$ gibi bir lisans ücreti bulunmaktadır. Windows platformunda çalışmaktadır. Ehe lab tarafondan geliştirilmektedir, çok fazla dokümanı olmamakla birlikte demo sürümünü ücretsiz olarak indirip deneyebilirsiniz.

Ruby-opencv: Ruby için geliştirilmiş bir wrapper'dır. Açık kaynak kodlu yürütülen bir projedir. OpenCV 2.4.10 sürümü referans alınarak geliştirilmiştir ve Ruby 1.9.3, 2.x desteklemektedir. Linux, MacOS ve Windows platformu için kullanılabilmektedir. Topluluk desteği ile geliştirilmektedir, https://github.com/ruby-opencv/ruby-opencv adresinden ulaşılabilir durumdadır.

Live CV: QML ile kolay bir şekilde OpenCV kütüphanesi ile uygulamalar geliştirmenizi sağlayan bir wrapper.

EmguCV

EmguCV bir OpenCV wrapper'ıdır. (OpenCV Wrapper'ları hakkında daha detaylı bilgi için buraya göz atabilirsiniz) .Net framework çatısı altında bulunan C #, VB, VC++, Xamarin veya IronPython ve Unity ile

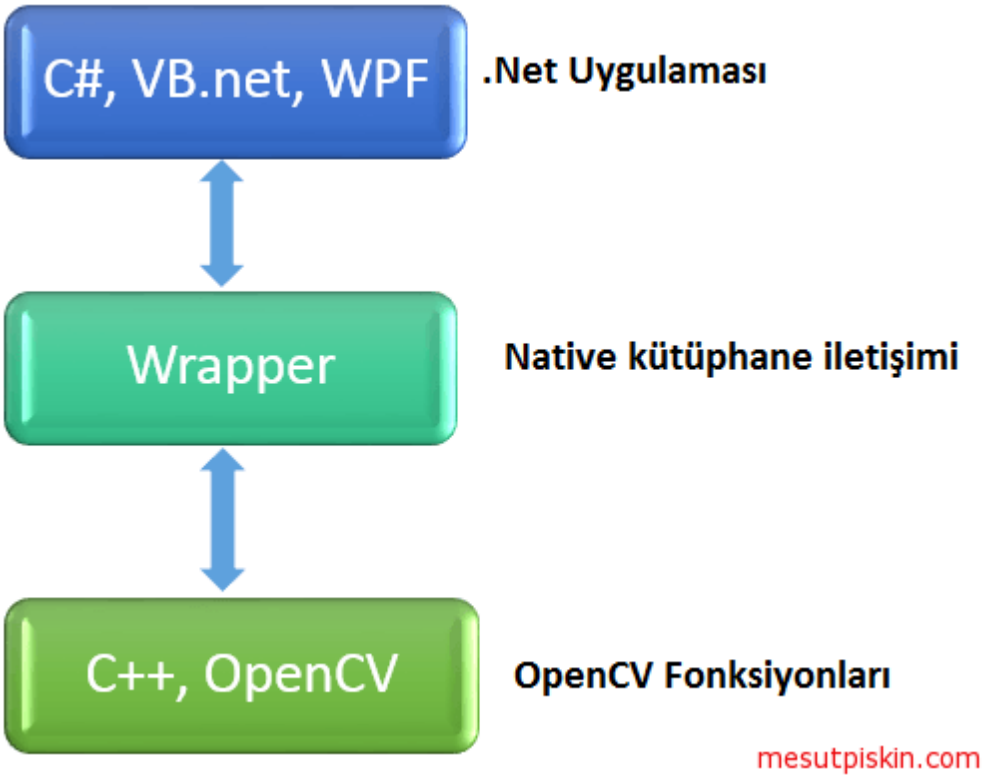
görüntü işleme uygulamaları geliştirmeyi kolaylaştırmaktadır. Windows, Linux, Mac OS X, iOS, Android ve Windows Phone platformlarında çalışabilmektedir. Güncelliğini devam ettiren bir kütüphanedir ve farklı lisanslamalara sahiptir. Resmi internet sitesi http://emgu.com

Alternatifleri ise;

- Opencvsharp
- OpenCVDotNet
- SharperCV

Wrapper'lar (EmguCV) ile OpenCV Arasındaki Farklar Nelerdir?

EmguCV'nin bir OpenCV wrapper'ı olduğundan daha önce bahsetmiştik, peki OpenCV ile arasındaki farklar nelerdir, projelerinizde OpenCV mi yoksa EmguCV mi tercih etmeliyiz sorusunun cevabına bakalım. Bildiğiniz üzere wrapper'lar geliştirilen kütüphaneyi referans alarak istenilen platform/teknoloji/dil için çalıştırılabilir/kullanılabilir yapmayı amaçlar. Bu amaç doğrultusunda EmguCV geliştiricileri native OpenCV kütüphanesinin fonksiyonlarını, sınıflarını ve algoritmalarını alırlar, bu fonksiyonları. NET ile çalıştırabilmek için native kütüphaneleri kullanan bir kütüphane yani. net karşılığı library project geliştirirler. .NET (C#,VB.NET vb.) geliştiricisi tarafından çağrılan bir fonksiyon veya kullanılan bir sınıf öncelikle EmguCV.dll'ine gider gelen/kullanılan istek/fonksiyon/sınıf işlenir ve karşılığı olan OpenCV kütüphanesi üzerinden çağrılır. İstekler ve fonksiyonlar bire bir olarak aynı değildir geliştiriciler platform/teknoloji arası farklılıklardan dolayı bazı değişiklikler yapmışlar gerek ekleme gerekse çıkarma yaparak. NET için kullanılabilir olmayı amaçlamışlardır. Bu sebeplerden dolayı çağrılan/kullanılan algoritmaya göre EmguCV ve OpenCV arasında performans farklıkları olabilir.



Teorik olarak bakıldığında her zaman en alt seviyedeki kütüphane daha hızlı çalışacaktır fakat platforma göre yapılan değişiklikler bu durumu tersine çevirerek bazı özel fonksiyonlarda yüksek seviye (EmguCV) kütüphaneleri daha performanslı kılmaktadır. Projeniz. NET ile geliştirilecekse doğrudan OpenCV kullanmanız mümkün olmayacaktır bu yüzden OpenCV mi EmguCV mi diye sormak yerine EmguCV mi yoksa diğer .NET wrapper'lar mı (Opencvsharp, OpenCVDotNet, SharperCV vb.)diye sormak daha doğru olacaktır. Bu soru da ücretlimi yoksa ücretsiz mi kaynak aradığınıza göre değişecektir. Lisans ücretleri sorun olmayacaksa EmguCV çok iyi bir seçim olacaktır fakat ücretsiz bir alternatif arıyorsanız güncel bir wrapper seçmelisiniz aksi taktirde elinizdeki wrapper OpenCV'nin eski versiyonlarını referans alarak geliştirildiği için bazı problemlerin çözümünde eski algoritmaları kullanmak zorunda kalabilirsiniz. Eski veya birçok bug bulunan algoritmalar performansı doğrudan etkileyen faktörlerdir. Opencvsharp en güncel sürüm olan OpenCV 3.1 sürümünü desteklemektedir. Mono, .NET Framework 2.0 ve sonrası versiyonları desteklemesi sayesinde diğer platformlar veya eski projeler için rahatlıkla kullanılabilir. İyi hazırlanmış wiki'si ve geniş örnek kütüphanesi ile aradığınız birçok şeyi kolayca bulabilirsiniz. Açık kaynak kod olarak BSD lisansı altında ücretsiz olarak geliştirilmektedir. OpenCVDotNet ve SharperCV güncel OpenCV versiyonlarını takip etmemekle birlikte uzun zamandır güncelleme gelmeyen ve geliştirilmesi neredeyse durdu denilebilecek kütüphanelerdir. Fakat fonksiyonlara getirdikleri farklı yaklaşımlarla temel görüntü işleme işleri için kullanılabilirler.

JavaCV

Java geliştiricileri OpenCV kütüphanesi ile uygulama geliştirken özelliklede Andorid platformu üzerinde bu işi yaparken bir çok zorluklarla karşılaşıyor. Örnekler veya dokümanlardaki anlatımlar Python ve C++ ile yapılıyor bu durum Java geliştiricileri için can sıkıcı bir durum, özellikle bazı C++ fonksiyonlarının Java karşılığının olmaması işleri daha da zora sokuyor. Bu durumda imdadınıza JavaCV yetişiyor.

JavaCV Nedir?

JavaCV, JavaCPP kullanılarak geliştirilmiş bir OpenCV wrapperıdır. Wrappar'ın ne olduğuna daha önce burada değinmiştik fakat kısaca tekrardan özetlememiz gerekirse; geliştirilen kütüphanenin, geliştirildiği kaynak dil, teknoloji referans alınarak istenilen hedef dil veya teknolojiye aktarılması bu platformda çalıştırılabilir hale getirilmesi denilebilir. JavaCV de OpenCV referans alınarak geliştirilmiş fakat üzerine birçok farklı kütüphaneler eklenerek genişletilmiş bir Java görüntü işleme kütüphanesidir. OpenCV den bağımsız değil, aksine paralel olarak aynı doğrultuda gelişim göstermektedir. Geliştirilen birçok metot sayesinde işleri kolaylaştırmakta ve geliştirme süresini kısaltmaktadır. Özellikle Android'in ivme kazanması ile kütüphaneye olan rağbet artmış ve bu platform için görüntü işlemeyi daha da kolay hale getirmek için ekstra modüller geliştirilmiştir. Örnek vermek gerekirse Geometric Calibrator, ProCam Color Calibrator, Canvas Frame,GLCanvas Frame, Parallel vb. olarak özetlenebilir. Kinect vb gibi donanımlar için geliştirilmiş sınıflar ile bu cihazlara yönelik uygulamalar kolaylıkla geliştirilebilir.

JavaCV mi? OpenCV mi?

Öncelikle projenizde ikisini de birlikte kullanabileceğinizi hatırlatmak isterim. Bu iki kütüphanenin performansını açıkça ortaya koyan bir grafik bulamadım bu yüzden tecrübelerime dayanarak anlatmayı tercih ettim. JavaCV ile OpenCV yi karşılaştırmak oldukça zor çünkü ne için kullanacağınıza ve amacınıza göre bu sorunun cevabı değişebilir. Amacınız Android platformuna uygulama geliştirmek veya OpenCV'nin C++ örneklerini Java projelerine implemente etmek ise JavaCV'yi tercih edebilirsiniz. Geliştirdiğiniz projede hız ve performans istiyor, gerçek zamanlı işler ile uğraşıyorsanız OpenCV

kütüphanesini tercih edebilirsiniz. Bu iki kütüphaneyi de keskin çizgilerle ayırmamak gerekiyor, aktarılan fonksiyonlar ve sınıflar çok fazla değiştirilmemiş üzerine eklemeler yapışmış durumdalar. Bu yüzden duruma göre iki kütüphaneyi de terci edebilirsiniz bir rakip, alternatif değil işinizi kolaylaştıran ara bir kütüphane olarak görebilirsiniz.

JavaCV Kurulumu

JavaCV kurulumu normal bir Java Kütüphanesi kurmaktan çokta farklı değil. Projeyi buradaki (https://github.com/bytedeco/javacv) Github bağlantısından indirebilir veya şuan ki en güncel sürümünü 1.2 yi binary olarak bu (http://search.maven.org/remotecontent?filepath=org/bytedeco/javacv/1.2/javacv-1.2-bin.zip) bağlantıdan indirebilirsiniz.

Sıkıştırılmış dosyayı çıkarttığınızda içerisinde birçok jar dosyası göreceksiniz, projenize JavaCPP, JavaCV, OpenCV ve kullanmış olduğunuz işletim sisteminize uygun opencv jar dosyasını projenize ekleyin (x64 windows için opencv-windows-x86_64.jar). Bağlantıdan örnekleri indirip deneyebilirsiniz. (https://github.com/bytedeco/javacv/tree/master/samples)

Live CV

Live CV Dinu SV tarafından geliştirilen açık kaynak kod bir geliştirme ortamı. Geliştirme ortamı diyorum çünkü farklı bir görüntü işleme kütüphanesi olarak düşünülmemeli. Live CV QML (Qt Meta Language veya Qt Modeling Language) dili ile json benzeri bir yapıda geliştirme yapmayı sağlayan OpenCV wrapperı. QML ile json formatındaki elementler şeklinde OpenCV fonksiyonları kullanılabiliyor. Hızlı prototipleme, yeni başlayanlar için OpenCV'yi kavrama veya akademik çalışmalar için oldukça kullanışlı bir ortam sunuyor. Live CV bir geliştirme ortamı ile birlikte geliyor bu geliştirme ortamı oldukça basit şekilde tasarlanmış ve yazdığınız kodun aynı anda çıktısını da görmenize olanak veriyor.

Live CV geliştirme ortamını buradan (http://livecv.dinusv.com/download.html) indirebilirsiniz, github depolarına ise buradan (https://github.com/livecv/livecv) ulaşabilirsiniz. Windows, Linux ve MacOS platformları için çalıştırılabilir durumdadır, diğer platformlar için kaynak kodunu indirerek derleyebilirsiniz. Live CV'yi çalıştırdığınızda aşağıdaki gibi bir IDE sizi karşılıyor olacak. Sol tarafta QML ile geliştirmelerimizi yapacağız sağ tarafta ise yazdığımız kodun anlık olarak çıktısını görebileceğiz. Örneğin görüntü üzerine uyguladığınız bir thresholdingi aynı t anında görebiliyorsunuz, sadece görseller için değil videolar içinde bu durum geçerlidir.

Basit bir örnek yapalım,

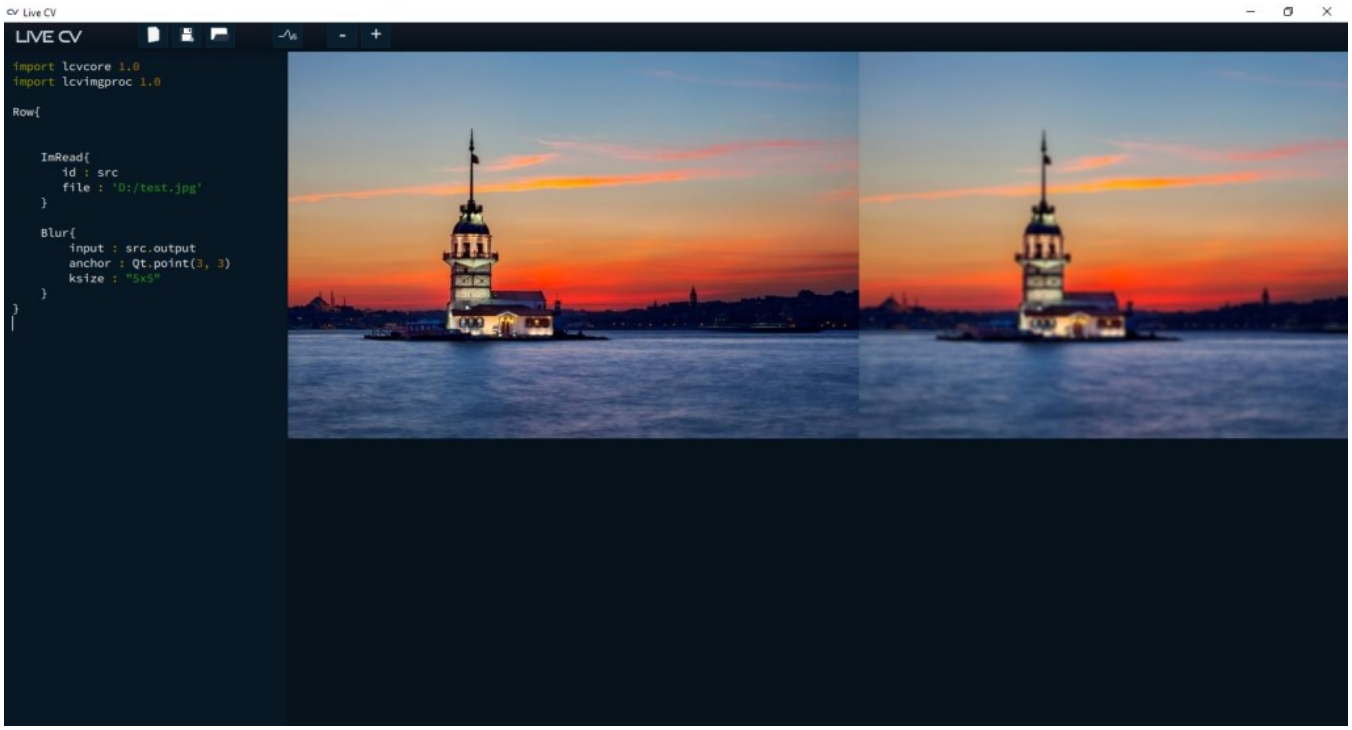
OpenCV de blur filtresi görüntüyü bulanıklaştırmak için kullanılır. Uygulamak için ise blur() metodu kullanılır. Bu metot parametre olarak kaynak görüntü mat nesnesi tipinde, mat tipinde bir sonuç ve Size tipinde uygulanacak olan bulanıklık değerini almaktadır.(çekirdek boyutu olarak da adlandırılır). Kullanımı ise aşağıdaki gibidir.

```
Imgproc.blur(kaynakGoruntu, hedefGoruntu, new Size(50,50));
```

Live CV ile aynı işlemi yapmak istediğimizde ise QML kodumuz aşağıdaki gibi olacaktır.

```
import lcvcore 1.0
import lcvimgproc 1.0
Row{
    ImRead{
       id : src
       file : 'D:/test.jpg'
    }
    Blur{
        input : src.output
        anchor : Qt.point(3, 3)
        ksize : "5x5"
    }
}
```

Görüntünün dosya sisteminden okunması ve filtrenin uygulanması bir akış şeklinde elementler ile temsil edilmektedir. Aynı değerler ile tanımlanan fonksiyonda sonuç matrisi bire bir aynı olacaktır.



Live CV Geliştirme Ortamı

OPENCV İÇİN PLATFORM VE GELİŞTİRME ORTAMI SEÇİMİ

OpenCV platform bağımsız bir kütüphanedir ve tüm platformlar için desteği bulunmaktadır. Projelerinizde kullanacağınız işletim sisteminde seçim şansınız var ise bu seçiminize yardımcı olabilmek ve platformların avantaj, dezavantajlarından bahsetmek istiyorum.

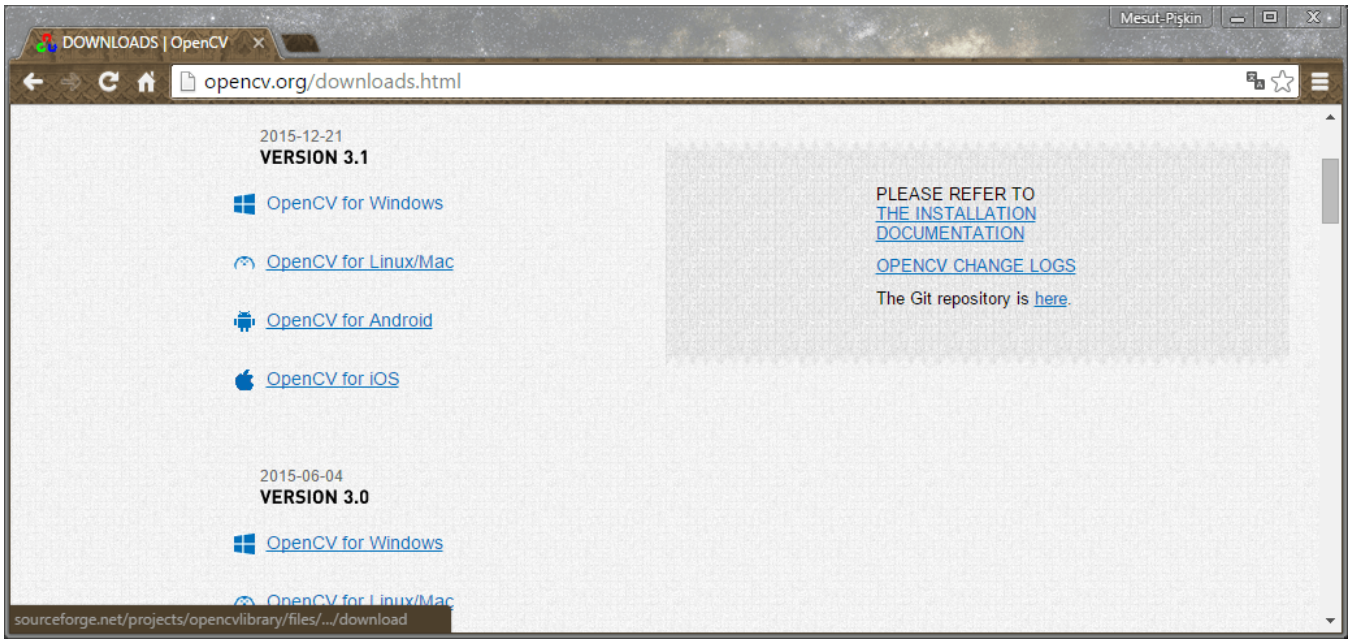
OpenCV'nin birçok sınıfı doğrudan grafik kartı sürücülerini kullanmaktadır. Grafik kartı sürücüleri platformlara göre farklılık göstermektedirler. Örneğin OpenCV 3.x sürümü Windows işletim sistemlerinde Nvidia grafik kartına sahip bazı sistemlerde mavi ekran (blue screen) olarak adlandırılan hataya sebep olmaktadır. Grafik kartını kullanan sınıfların bazı fonksiyonları çalıştırıldığında bu hata ortaya çıkmaktadır, OpenCV soru&cevap sitesinde bu sorundan dert yanan birçok kişiyi görebilirsiniz. Linux ve Mac platformunda ise şuan itibariyle böyle bir hata yoktur. Böylesi bir durumla karşılaştırsanız sürücü güncellemelerinizi yapmanızı çözüm bulamaz iseniz kullandığınız versiyonu veya platformu değiştirmenizi öneririm.

Gömülü sistemlerde Raspberry Pi, OrangePi, Beaglebone, Banana Pi vb. kartlarda işletim sistemi seçerken daha önceki tecrübelerime dayanarak söyleyebilirim ki Debian Linux çatallaması işletim sistemleri çok daha performanslı çalışmaktadır.

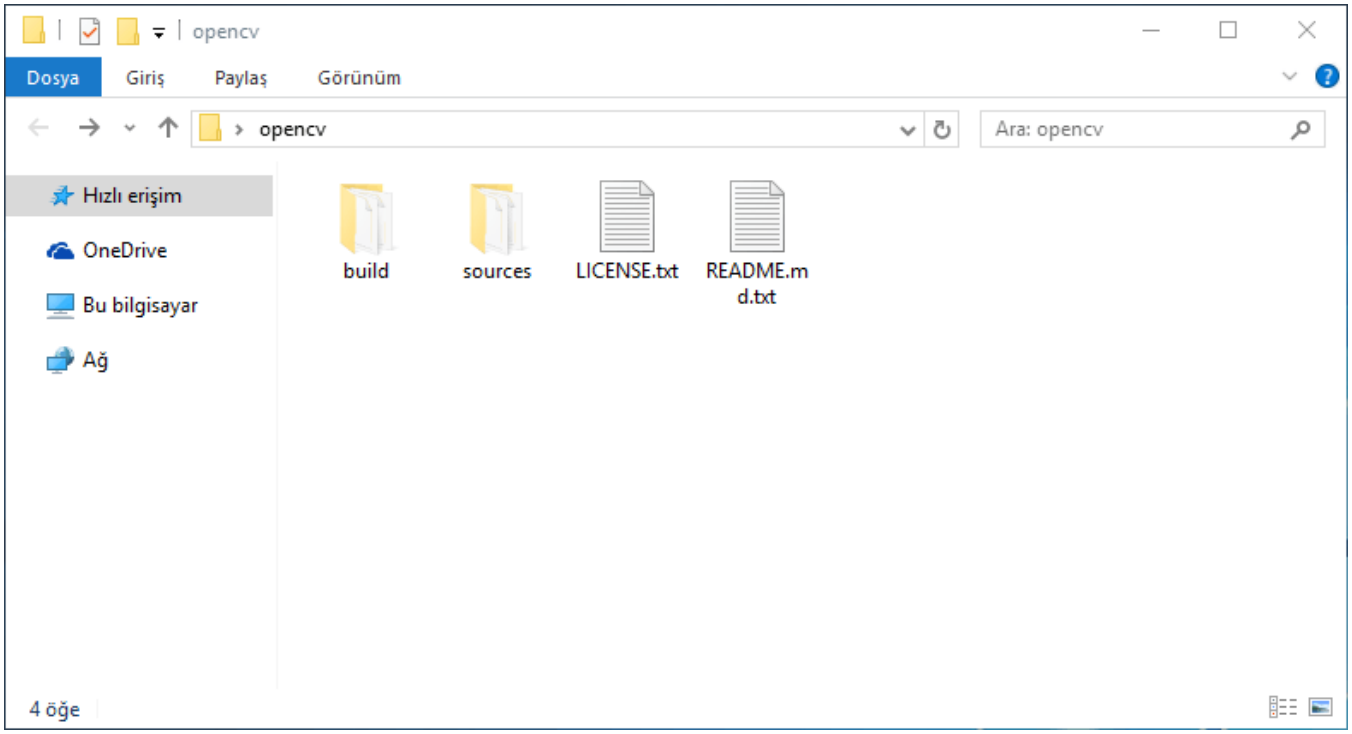
Kitap içerisinde yer alan uygulamalar Java programlama dili çatısı altında Eclipse, Netbeans ve Android Studio ile geliştirilmiştir. Java öğrenmeye yeni başlamış birisi iseniz bu öneriler sizin içindir. Grafiksel kullanıcı arabirime (GUI) sahip, etkileşimli uygulamalar geliştirmek istiyorsanız Netbeans IDE kullanabilirsiniz. Hızlı kod geliştirmek ve çok fazla grfiksel olmayan uygulamalar için ise Eclipse IDE tercih edebilirsiniz. Yeni başlayan birisi iseniz Eclipse IDE ve size karmaşık gelebilir bu durumda Netbeans kullanabilirsiniz. Kitap içerisindeki mobil Android uygulama örnekleri ise Android Studio ile geliştirilmiştir. Tüm geliştirme ortamları için OpenCV kurulması ve ayarların yapılması anlatılacaktır.

WİNDOWS İŞLETİM SİSTEMİ İÇİN OPENCV KURULUMU

OpenCV'nin, bu yazının yazıldığı tarih itibari ile en güncel sürümü 3.1 dir. Windows işletim sistemi için OpenCV derlenmiş, sistem kütüphanesi haline getirilmiş olarak bulunmaktadır. Bu sayede kaynak kodu tekrardan derlemeye ihtiyaç olmadan kullanılabilir haldedir. OpenCV'yi http://opencv.org/downloads.html adresine giriyoruz ve indirmek istediğimiz sürümün altındaki OpenCV for Windows linkine tıklıyoruz. Örneklerde şuanda en güncel olan 3.1 sürümünü kullanacağız. İndirme bağlantısı sourceforge.net sitesine yönlendirecek ve indirme işlemi başlayacak.



İndirdiğinizde sıkıştırılmış olarak gelecektir, çalıştırdığınızda OpenCV dosyalarını çıkartmak için bir dizin isteyecektir, burada çıkartılmasını istediğiniz dosya dizini yolunu yazarak Extract butonuna tıklayın. Dosyaları çıkarttığınız dizinde OpenCV klasörü içerisinde build ve sources diye 2 adet klasör bulunmaktadır. Build klasörü içerisinde Windows platformu için derlenmiş olarak sistem native kütüphaneler ve programlama dilleri için kütüphaneler bulunmaktadır. Sources klasöründe ise OpenCV kaynak kodları ve örnek uygulamalar yer almaktadır. Buradaki kaynak kodlar ile OpenCV'yi tekrardan derleyebilirsiniz.



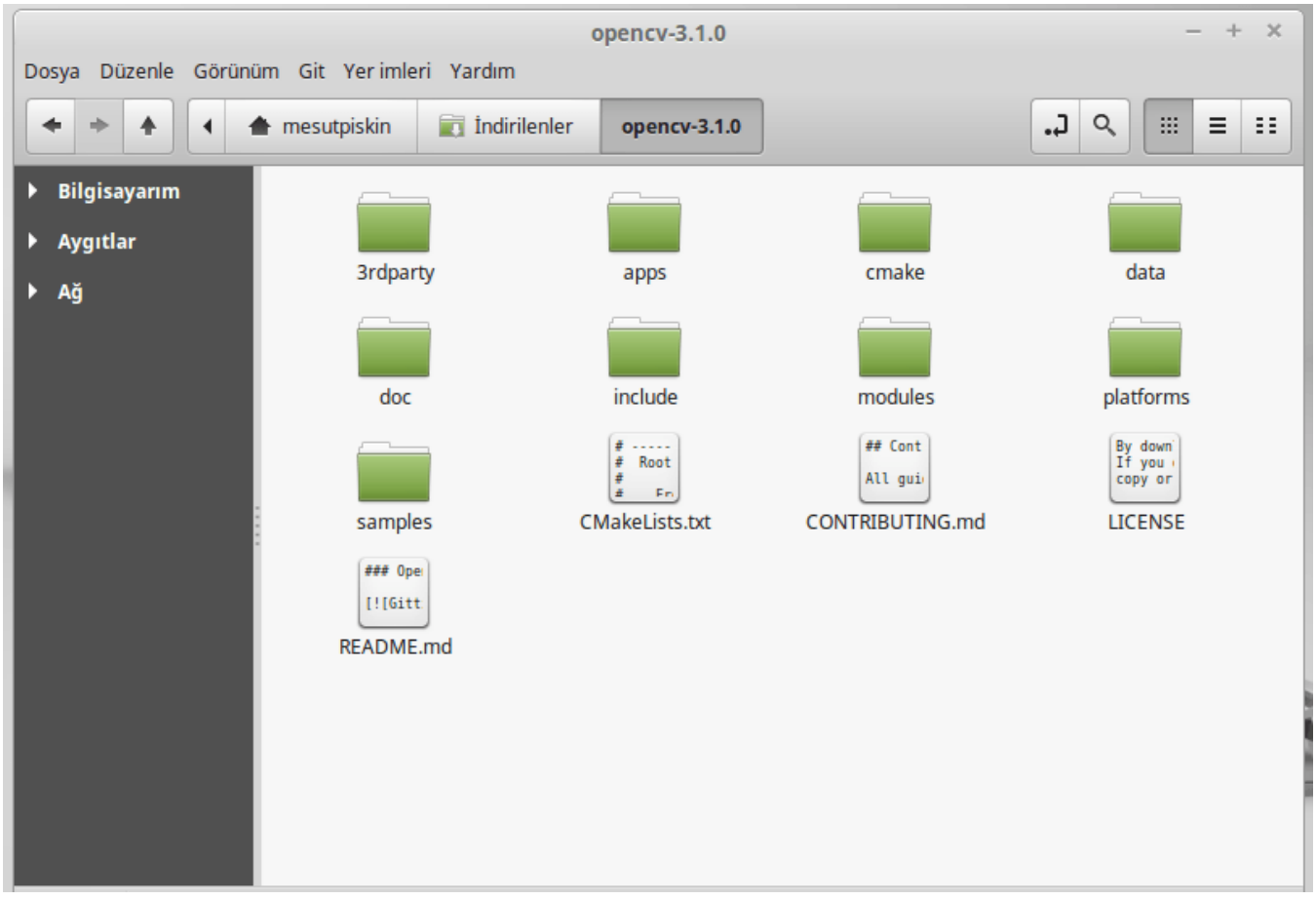
Java ile OpenCV uygulaması geliştirmek için build içerisindeki java klasöründe yer alan jar dosyasını ve kullanacağımız işletim sistemi mimarisine göre OpenCV Windows sistem kütüphanelerini kullanacağız.

LİNUX İŞLETİM SİSTEMİ İÇİN OPENCV KURULUMU

Linux işletim sistemi için OpenCV kütüphanesi kaynak kod olarak yani derlenmeden gelmektedir, bunun en temel sebebi birçok Linux dağıtımı olmasıdır. Temel olarak amaçlanan şey kullanmak isteyenin kendi sistemine ve kullanmak istediği programlama diline göre derlemesidir. Derleme işleminde Linux Mint kullanacağım ve tüm Debian dağıtımlarında tüm adımlar aynı olacaktır. Debian ve Debian çatallaması haricinde bir Linux dağıtıma sahipseniz, kullandığınız paket yöneticisine göre komutlar değişiklik gösterebilir, fakat bire bir olarak karşılıkları yer almaktadır.

OpenCV kurmak için öncelikle http://opencv.org/downloads.html adresinden kullanmak istediğiniz sürüm altında yer alan OpenCV for Linux/Mac bağlantısına tıklayarak OpenCV geliştiricisi olan Itseez'in Github hesabından kaynak kodları indiriyoruz. Dikkat ederseniz Linux ve Mac platformları için ayrı ayrı indirme bağlantısı bulunmamaktadır. OpenCV'nin C ve C++ dilleri ile platform bağımsız olarak geliştirildiği söylemiştik, bu sayede kaynak kodu indirerek istediğiniz platform üzerinde derleyerek çalıştırabilirsiniz.

İndirme işlemi tamamlandığında OpenCV'nin kaynak kodlarını içeren bir zip dosyası gelecektir. Bu zip formatında sıkıştırılmış dosyayı bir dizine çıkartınız. Bu dizinde kaynak kodun derlenmesi işlemini yapacağız.



Öncelikle işletim sistemi üzerinde JDK (Java Developer Kit) kurulu olması gerekmektedir. Java ile OpenCV uygulaması geliştireceğimiz için bunu kurmanız gerekmektedir. Kullandığınız dağıtımın paket yöneticisinden ya da aşağıdaki bağlantıdan veya aşağıdaki komutlar aracılığıyla depoyu ekleyip oradan indirip kurabilirsiniz.

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

sudo add-apt-repository ppa:webupd8team/java sudo apt-get update sudo apt-get install oracle-java8-installer sudo apt-get install oracle-java8-set-default

Java kurulumunun ardından Apache ANT aracına ihtiyacımız var kurmak için aşağıdaki komut ile deponuzdan indirebilirsiniz ya da https://ant.apache.org/ adresinden kurulumu indirebilirsiniz.

sudo apt-get install ant

Linux üzerinde OpenCV çalıştırabilmek için bazı paketlere ihtiyacımız var bu paketlerden bir kısmı bazı Linux dağıtımlarında kurulu olarak gelmektedir. Bu paketleri aşağıdaki komut ile kurabilirsiniz.

sudo apt-get install g++ libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev build-essential

Derleme işlemine başlamadan önce Java yolunu sisteme eklememiz gerekiyor varsayılan olarak JDK usr/lib/jvm/ dizinine kurulur. Bu dizinde, kurduğunuz sürümün klasörünü görebilirsiniz. Yukarıda Oracle JDK 8 kurmuştuk ve bu klasörü yol olarak göstereceğiz, sizde sisteminizdeki JDK dosya yoluna göre aşağıdaki dizini değiştirebilirsiniz.

export JAVA_HOME=/usr/lib/jvm/java-8-oracle

Derleme için cmake aracını kullanacağız. Cmake platform bağımsız bir derleme aracıdır, üzerinde birçok derleyici yer alır ve o derleyiciye göre çıktı üretebilmektedir.

sudo apt-get install cmake

Yukarıda yer alan komut ile cmake kurulumunu yapıyoruz, kullandığınız dağıtımın deposunda yok ise https://cmake.org/download/ adresinden indirip kurabilirsiniz.

Gerekli araçları indirip kurduktan sonra artık derleme işlemine geçebiliriz. İlk adım olarak OpenCV'yi zip dosyasından çıkarttığımız klasöre gidiyoruz ve sırası ile aşağıdaki adımları takip ediyoruz.

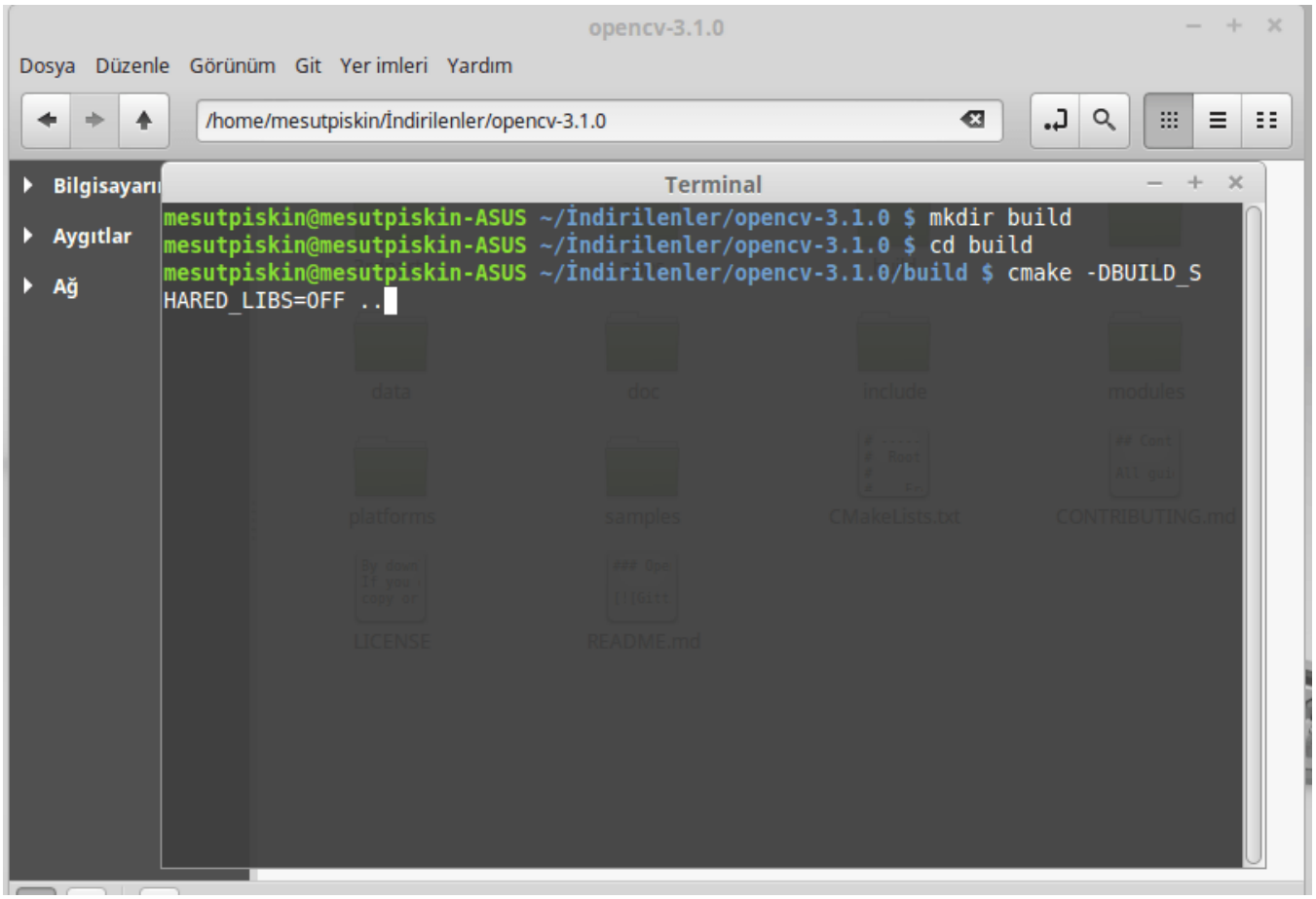
mkdir build

İndirdiğimiz dizinde derleme işlemi sonucu için bir klasör oluşturuyoruz.

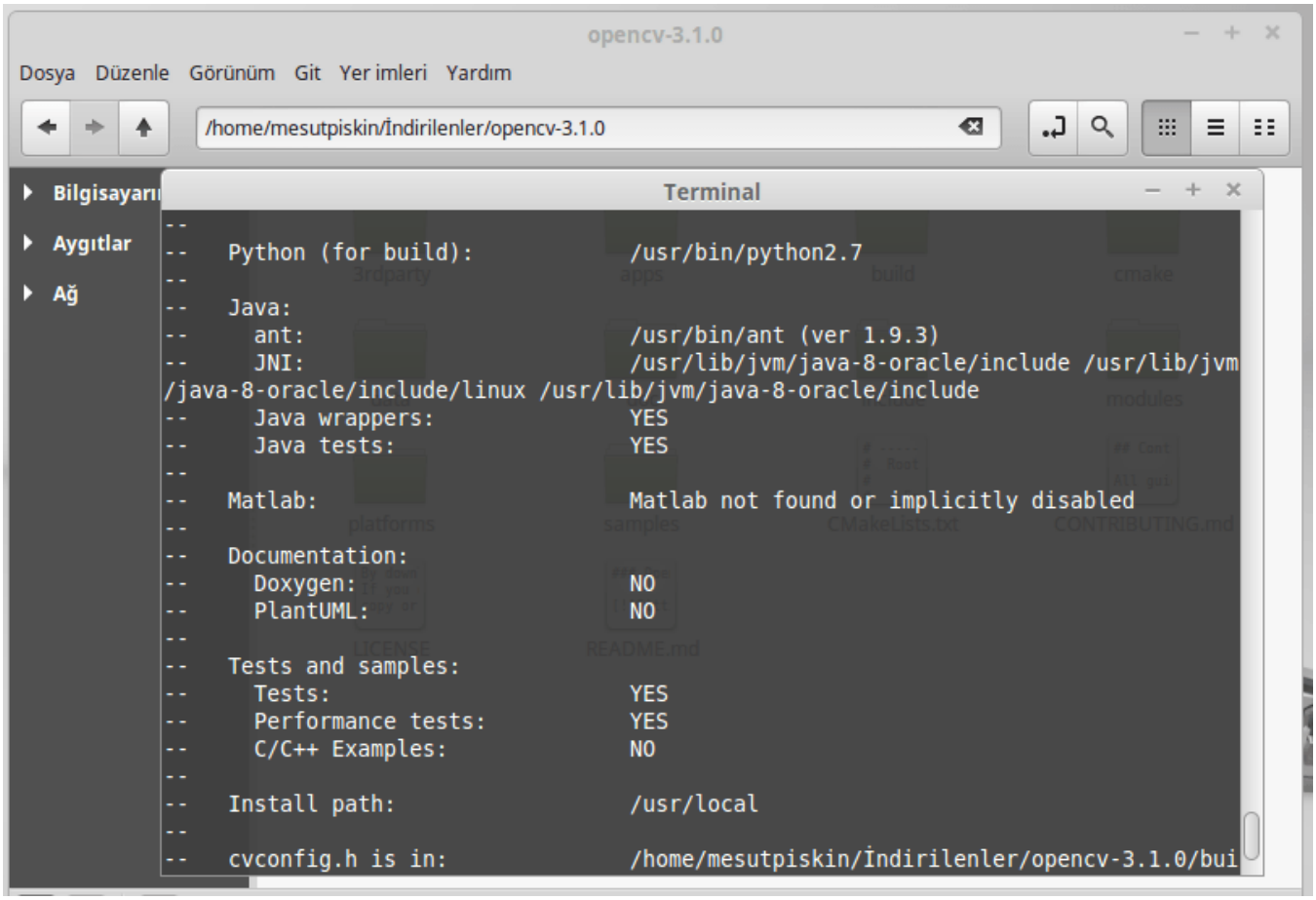
cd build

Oluşturduğumuz build klasörü içerisine gidiyoruz ve derleme işlemi için cmake aracını başlatıyoruz.

cmake -DBUILD_SHARED_LIBS=OFF ..



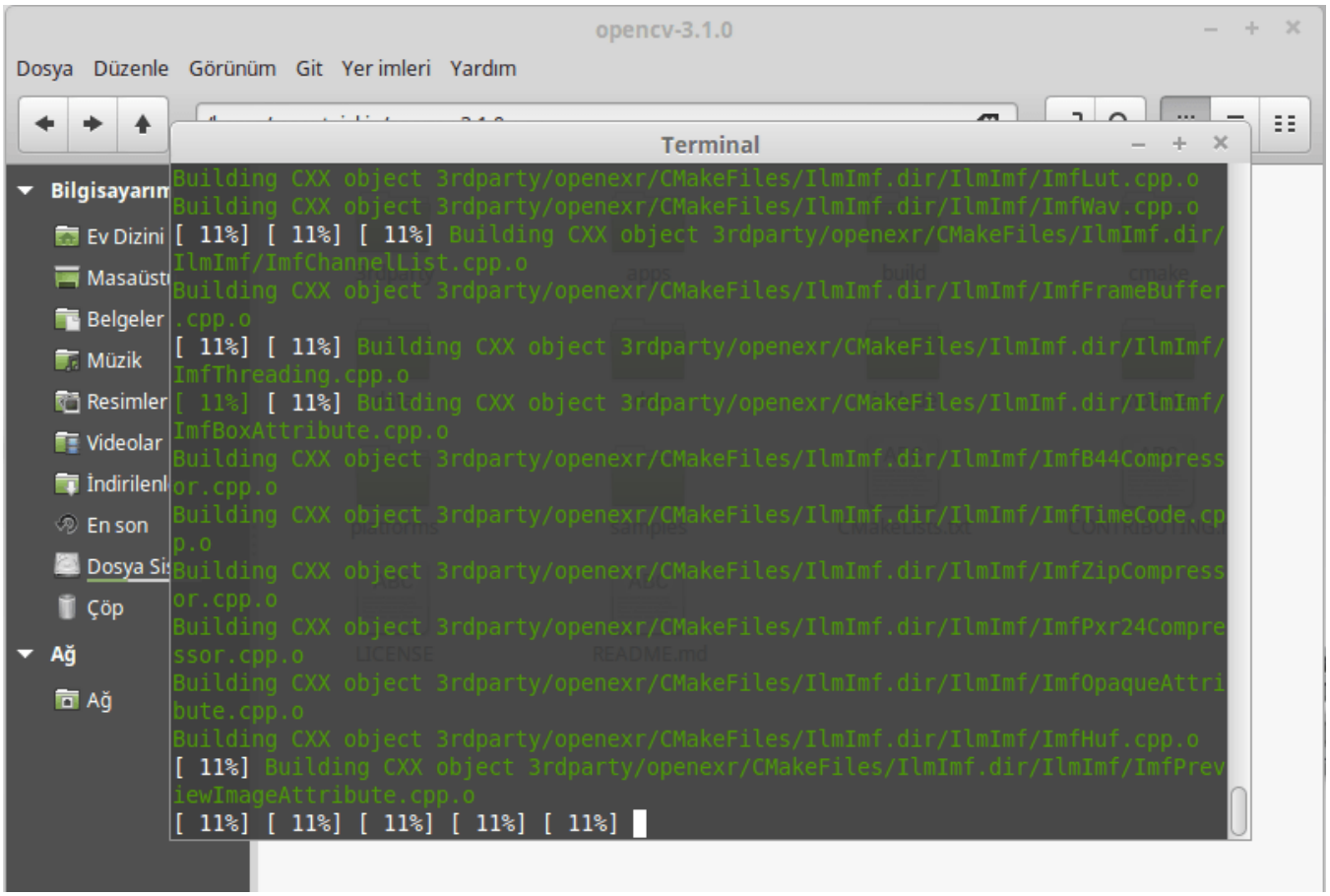
Bu komutun ardından cmake buil klasörü içerisine bazı dosyalar oluşturacak ve derlemeye başlamadan yapılacak olan işlemler için bir çıktı gösterecektir. Burada dikkat etmemiz gereken nokta java kısmı, eğer java başlığı altında ant ve JNI dizinleri karşısında NO ibaresi yer alıyor ise derleme işlemine başlamayınız, jar dosyasını ve sistem kütüphanesini oluşturmayacaktır. Bunun nedeni ant veya JDK kurulumlarının eksik veya java yolunun verilmemiş olmasıdır yukarıdaki adımları tekrardan yapınız. Eğer java başlığı altında aşağıdaki gibi dosya dizinleri görünüyor ise derleme işlemine geçebilirsiniz.



Hata ile karşılaşırsanız build/makefiles içerisinde yer alan error log dosyasına bakabilirsiniz. Derleme işlemini make komutu ile başlatacağız. Bu işlem sisteminizin donanıma göre değişiklik göstermektedir. make –j komutu ile derleme işlemini farklı threadlere bölerek bu süreyi hızlandırabilirsiniz, örneğin make –j4 ile derleme yaparsanız toplam işlemi 4 e bölerek, 4 ayrı iş parçası olarak çalıştırır. Eğer make –j ile derleme esnasında Process Kill hatası alırsanız sadece make komutunu kullanarak derleme işlemini yapın. Raspberry Pi gibi geliştirme kartları üzerinde OpenCV derleme işlemi yapacaksanız tamamlanma süresi biraz uzun olacaktır.

make –j4

Komutu ile derleme işlemini başlatıyoruz.



Derleme işleminin ardından build/bin içerisine opencv-3.jar gibi kurduğunuz sürüme göre ismi değişecek bir jar dosyası olacaktır. Yine build/lib dizininde linopencv_java3.so (kurulan sürüme göre ismi farklılık gösterir) olarak native kütüphane dosyası olacaktır. Bu jar dosasını ve kütüphaneyi uygulama geliştirmek için kullanacağız.

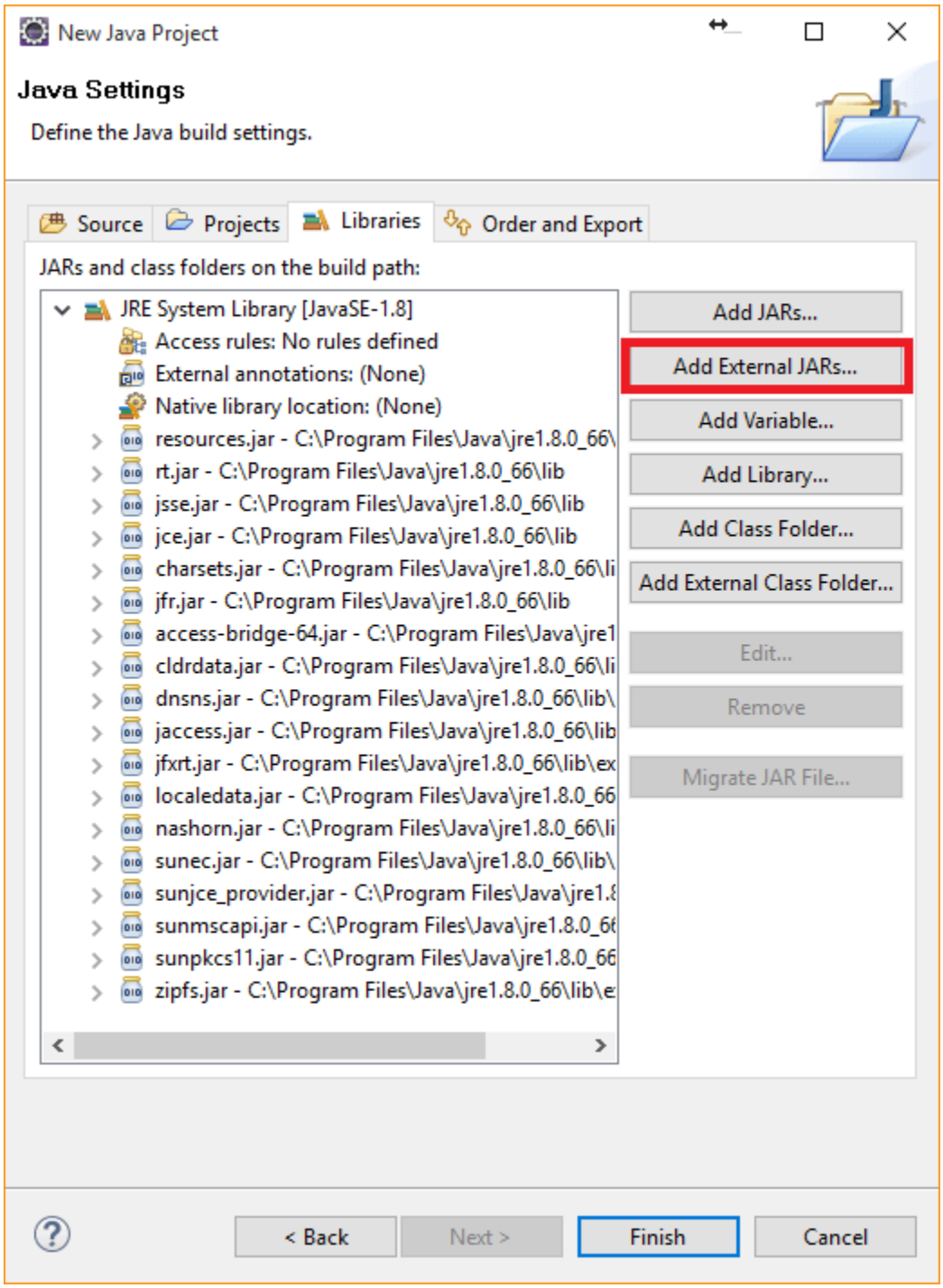
Blogdaki örnek projeler için Eclipse ve Netbeans kullanılmıştır. Java uygulamaları için en çok tercih edilen iki geliştirme ortamı içinde kurulum ve yapılandırma işlemi yapacağız. Siz tercih ettiğiniz ortam ile uygulamaları geliştirebilirsiniz.

ECLİPSE IDE İÇİN OPENCV YAPILANDIRMASI

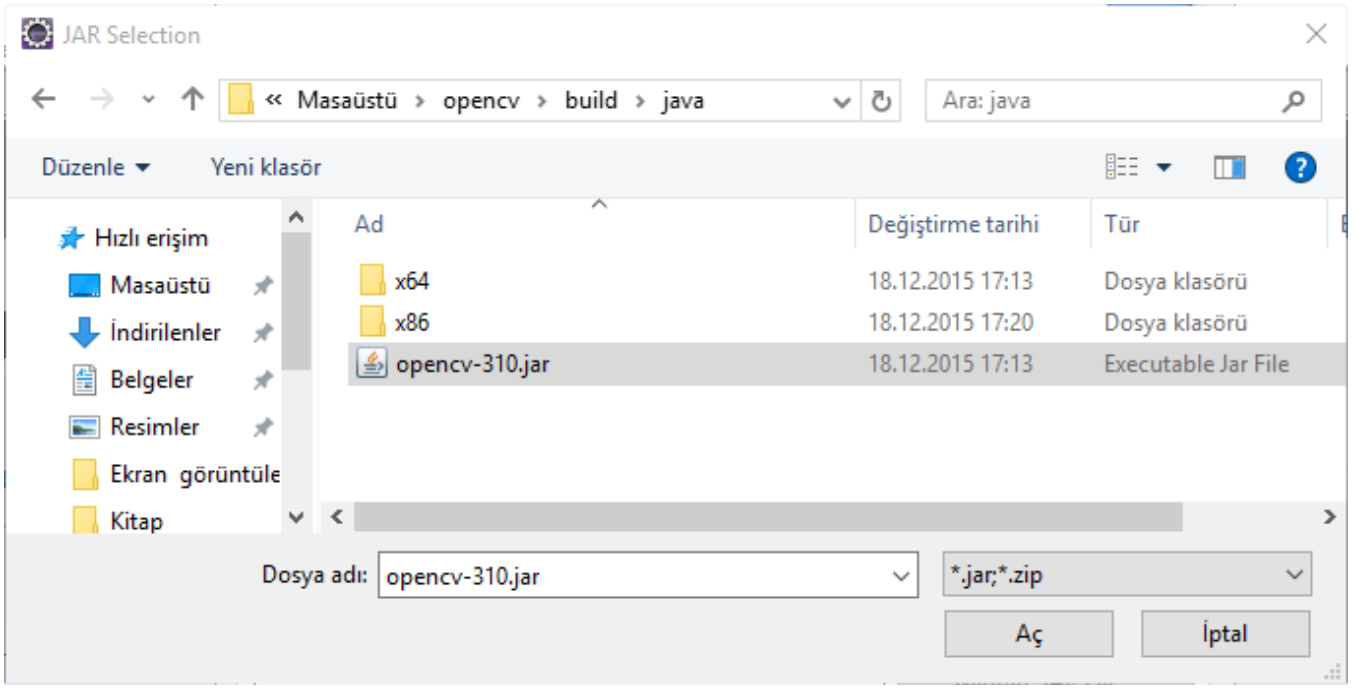
Eclipse ücretsiz bir geliştirme ortamıdır. Eklenti desteği sayesinde esnek ve hızlıdır, java projelerinde en çok tercih edilen IDE dir. Eclipse Mars 1 sürümünü kullanacağım tercih ettiğiniz bir sürümü kullanabilirsiniz. Eclips'i https://eclipse.org/downloads/ adresinden Eclipse IDE for Java Developers veya Eclipse IDE for JavaEE Developers paketini kullandığınız işletim sistemine göre indirebilirsiniz.

OpenCV kütüphanesini kullanabilmek için jar dosyasına ve sistem kütüphanesine ihtiyacımız var. Bu dosyalar Windows kullanıyorsanız daha önce indirdiğiniz OpenCV klasörü içerisindeki build\java dizininde jar dosyası ve işlemci mimarisine göre x86 ve x64 olarak sistem kütüphanesi bulunmaktadır. Linux ile OpenCV derleme işlemi yaptıysanız Opencv klasörü içerisinde build\bin dizininde jar dosyası, build\lib içerisinde ise sistem kütüphanesi yer almaktadır.

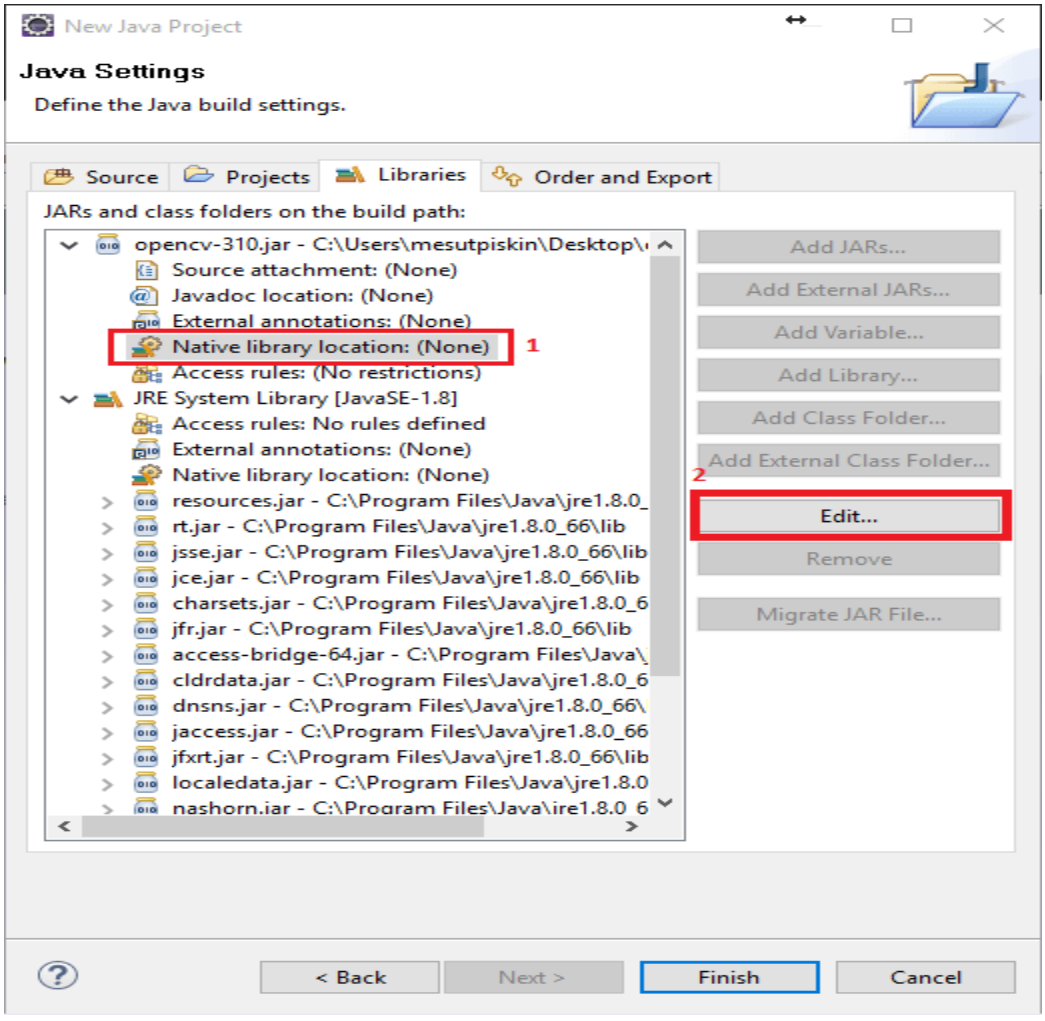
Eclipse ile yeni bir proje oluşturalım ve jar dosyasını kütüphanelere ekleyelim.



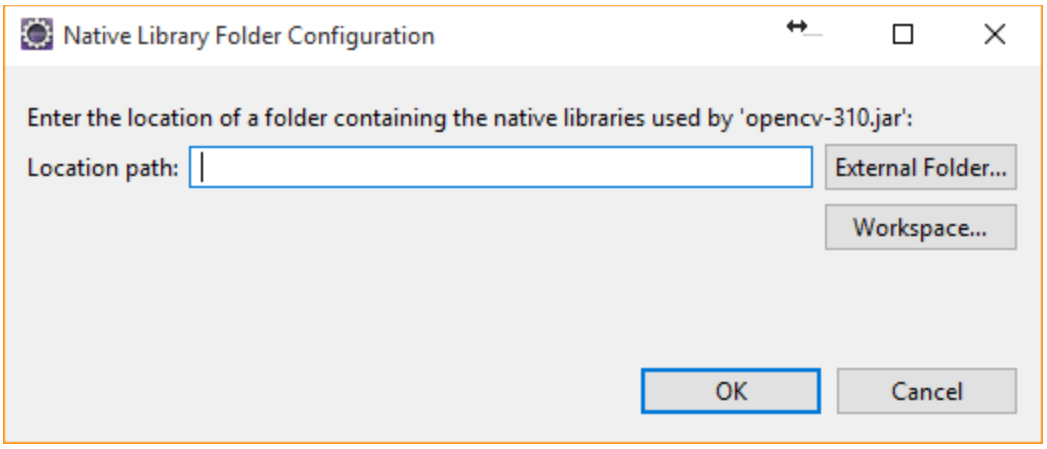
Add External JARs butonu ile OpenCV jar dosyasını seçiyoruz. Bir user library oluşturup, jar dosyasını ve sistem kütüphanesini proje içerisine ekleyebilirsiniz.



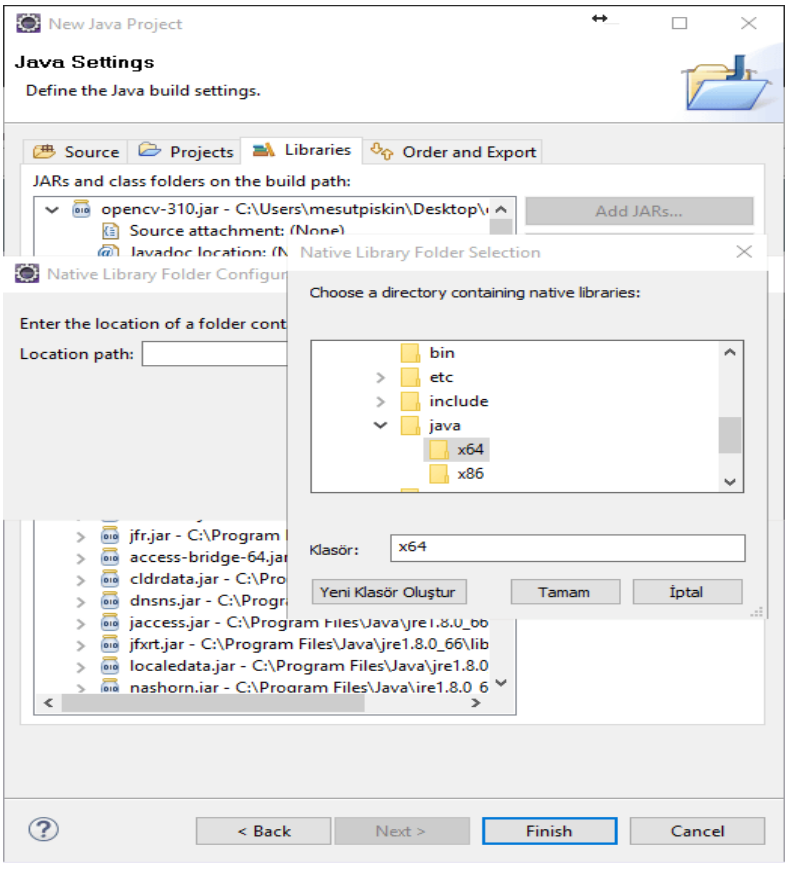
OpenCV'nin bulunduğu dizinden jar dosyasını seçerek devam ediyoruz.



Bir sonraki aşamada işletim sistemine mimarisine göre sistem kütüphanesini, jar dosyasına native library olarak eklememiz gerekmektedir. Windows kullanıyorsanız x64 ve x86 olarak işlemci mimarisine göre farklı kütüphaneler bulunmaktadır kullandığınız mimariye göre seçmeniz gerekmektedir. Linux kullanıyorsanız derleme işlemini kendi bilgisayarımızda yaptığımız için, kullanılan bilgisayar mimarisine göre derlemiştir bu yüzden so uzantılı kütüphaneyi lib klasöründen seçmeniz yeterli olacaktır. Jar dosyasını seçtik ve library kısmına Opencv Kütüphanesi geldi. Native library location seçerek edit butonuna tıklıyoruz.



Açılan pencereden External Folder diyerek sistem kütüphanenin bulunduğu klasörü seçiyoruz. Linux kullanıyorsanız build içerisindeki lib klasörünü seçmeniz yeterlidir.

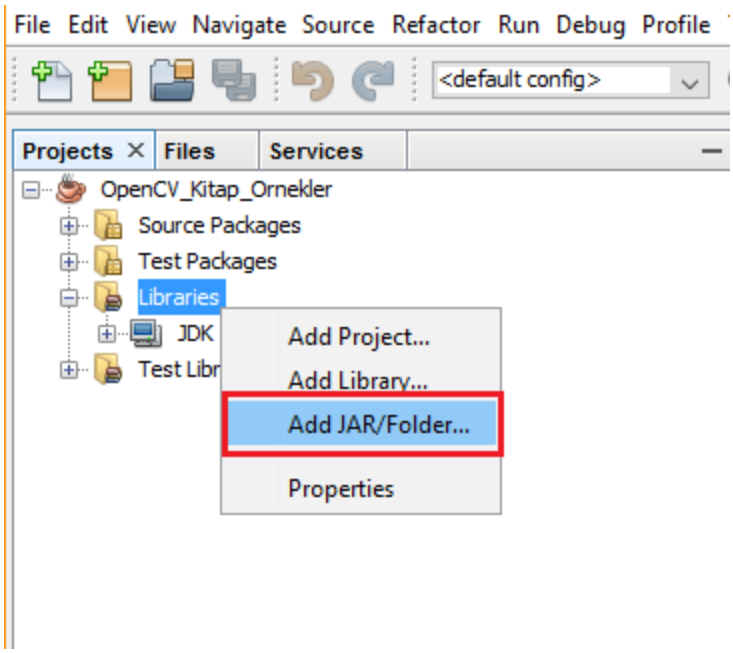


Bu işlemlerin ardından OpenCV kullanıma hazır olacaktır. Finish butonu ile projeyi oluşturabilirsiniz.

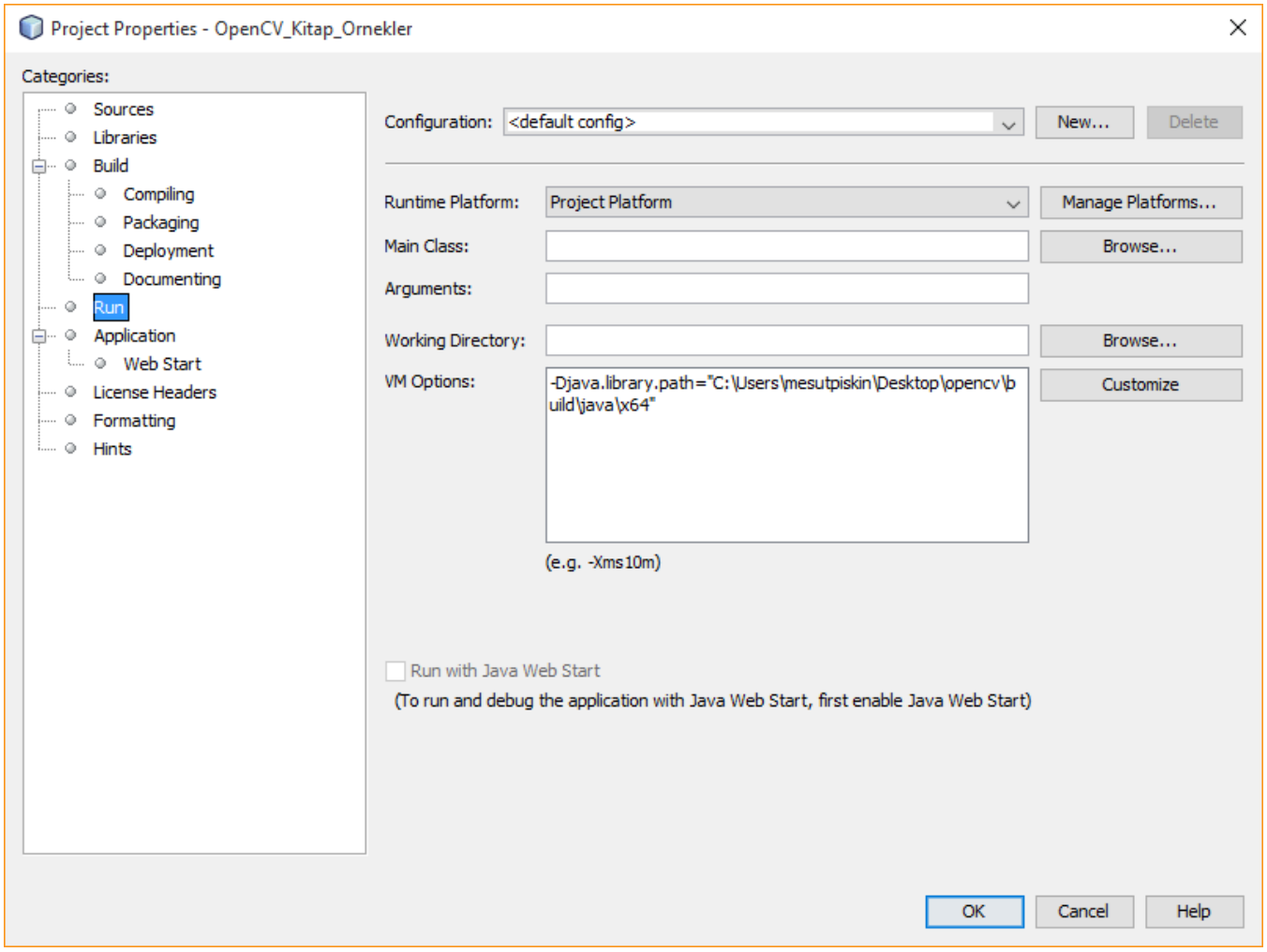
NETBEANS IDE İÇİN OPENCV YAPILANDIRMASI

Netbeans Oracle şirketi tarafından geliştirilen ve ücretsiz olarak dağıtılan bir yazılım geliştirme ortamıdır. Özellikle Java GUI projeleri için sıklıkla tercih edilir. Netbeans IDE içinde OpenCV kütüphanesi kurulumu yapacağız. Netbeans IDE'yi https://netbeans.org/ adresinden indirip kurabilirsiniz.

Netbeans ile yeni bir Java projesi oluşturalım ve projemize OpenCV kütüphanesini ekleyelim. Bunun için Project paelinden libraries üzerine sağ tıklayalım ve Add JAR/Folder ile build/java dizini içerisindeki jar dosyasını seçerek yükleyelim. Yeni bir library oluşturarak jar dosyasını proje içerisine yerleştirebilirsiniz.



Jar dosyasını ekledikten sonra, run menüsünden run project veya F6 tuşu ile projeyi çalıştıralım. Son olarak native sistem kütüphanesinin yolunu göstermemiz gerekiyor. File menüsünden project propertis seçeneği ile proje özellikleri menüsünü açalım. Sol tarafta yer alan menüden run diyerek VM Options kısmına Windows için dll Linux için so uzantılı olan sistem kütüphanesininin yolunu -Djava.library.path="Dosya yolu" parametresi ile verelim.



Bu işlemin ardından OpenCV uygulaması geliştirmek için Netbeans ortamı hazır.

ANDROİD STUDİO İÇİN OPENCV YAPILANDIRMASI

OpennCV Android platformu için de destek vermektedir. Bu sayede Java ile android uygulamaları geliştirirken OpenCV kütüphanesinden yararlanabilirsiniz. OpenCV web sitesi üzerinden "OpenCV for Android" linkine tıklayarak kitabın yazıldığı tarih itibariyle en güncel olan 3.1 sürümünü indiriyoruz. İndirdiğimiz zip dosyasını çıkartıyoruz.

Android stuido üzerinde proje oluşturalım, oluşturulan projeye opencv kütüphanesini ekleyeceğiz. File menüsünden New/ Import Module diyerek New Modüle penceresini açıyoruz. Bu pencereden daha önce indirdiğimiz OpenCV klasörü içerisinde SDK/Java klasörünü seçerek devam ediyoruz. Derleme işlemi ardından hata mesajı ile karşılaşırsanız bunun nedeni OpenCV kütüphanesinin derlendiği sdk versiyonunun sisteminizde kurulu olmamasıdır. SDK Manager ile istenilen sürümünü kurabilirsiniz.

Kullandığınız SDK sürümü ile tekrardan derleyerek te bu hatayı ortadan kaldırabilirsiniz. Bunun için yapmanız gereken, Project menüsünden projenizin altında OpenCVLibrary altındaki (görünümü Project Files olarak değiştirmelisiniz) build.grandle dosyasını açınız. Android altında derleme için kullanacağınız sdk sürümünü ve buildToolVersion'ı ekleyiniz. Tekrar bir derleme işleminin ardından hata ortadan kalkacaktır.

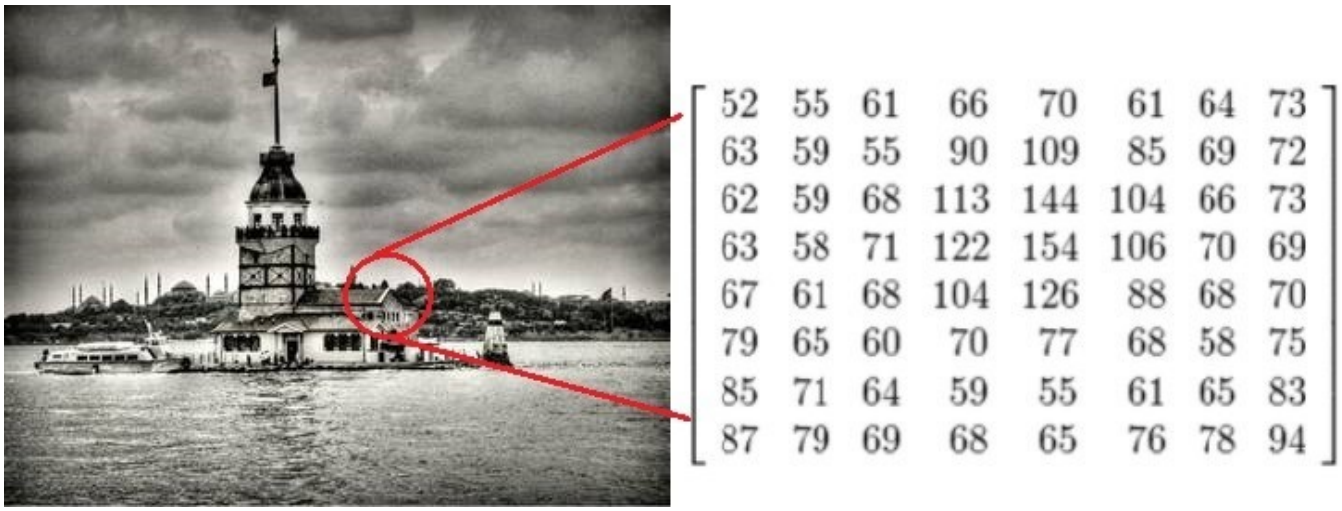
Daha sonra File menüsünden Project Structure'a tıklayarak veya Ctrl+Alt+Shift+S kısa yolu ile Project Structure penceresini açıyoruz. Dependencies sekmesine gelerek sağ tarafta yer alan + butonuna tıklıyoruz, buradan Module dependency seçeneği ile açılan pencereden daha önce eklediğimiz openCVLibrary modulünü seçiyoruz tamam diyerek pencereleri kapatıyoruz. Tekrarlanan bir derleme işlemi ardından OpenCV paketleri kullanım için hazır olacaktır.

TEMEL DİJİTAL GÖRÜNTÜ İŞLEME KAVRAMLARI

Dijital görüntü işleme ile ilgili birçok kavrama aşina olabilirsiniz fakat bu kavramların çok iyi bir şekilde bilinmesi önem arz etmektedir. Bu nedenle bazı görüntü işleme kavramların ne olduğunu açıklamaya çalışacağım. Bu tanımlar ansiklopedik olarak değil, görüntü işlemede neyi ifade ettiklerine göre olacaktır.

Dijital Görüntü: Gerçek yaşamdaki analog bir verinin kamera gibi donanımlar kullanılarak dijital bir hale getirilmesiyle oluşmaktadır. Bu veri 1 ve 0 ile dijital olarak tanımlanıp, analog karşılığına denk gelmektedir. Dijital görüntü 2 boyutlu bir dizi yani satır ve sütun veya renk uzayına göre farklı boyutlara sahip matris olarak depolanabilmektedir.

Piksel: Dijital bir görüntünün satır ve sütun olarak barındırılabileceğini söylemiştik, bu dijital görüntüde satır ve sütunların kesiştiği noktalar piksel olarak adlandırılmaktadır yani her bir hücreye piksel denilmektedir. Bu durumda piksel dijital görüntüyü oluşturan en küçük birimdir.



$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Yukarıdaki görselde göründüğü üzere bir dijital görüntünün seçili bölgesinden kesit alınmış ve bu kesitin dijital olarak nasıl tutulduğu gösterilmiştir. Alınan kesit 8×8 boyutunda bir dizidir, bu dizinin her bir elamanının tuttuğu veri ise dijital görüntünün o pikselindeki renk bilgisine karşılık gelmektedir.

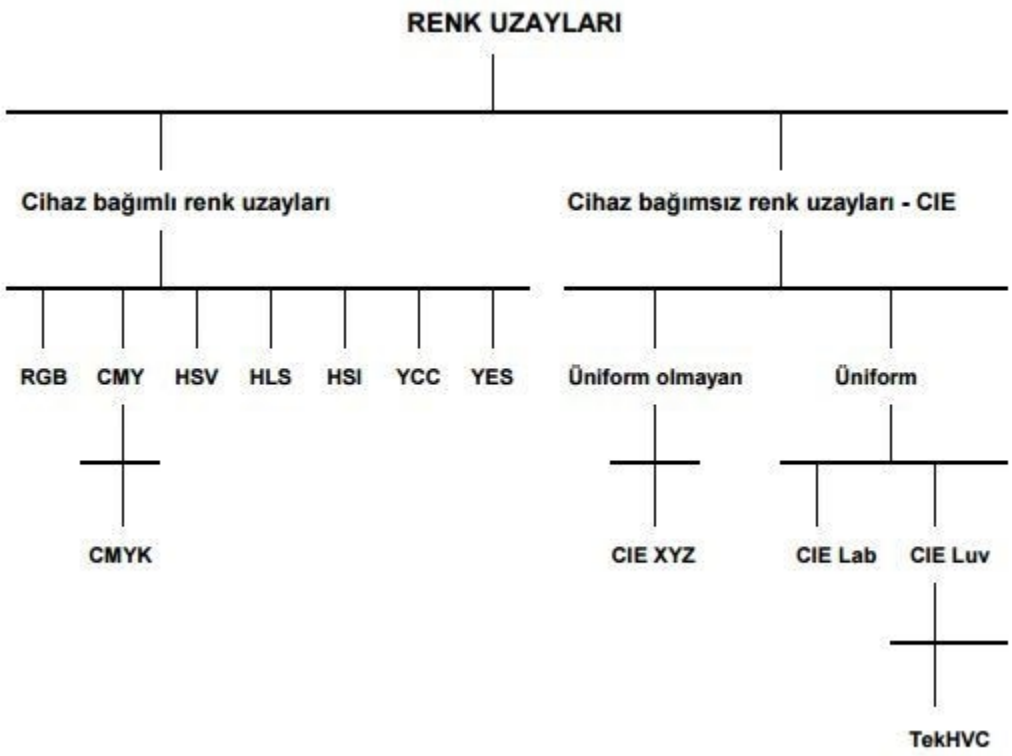
FPS (Frame Per Second): Saniyedeki çerçeve sayısı olarak tanımlanır. Video aygıtlarında ve grafik kartlarında ayırt edici bir kriterdir. Bu görüntüleme aygıtlarının bir saniyede ürettiği görüntü sayısını ifade eder. Fps değeri ne kadar yüksek ise saniyede yakalanan görüntü sayısı artacak ve en küçük değişiklikler bile yakalanabilecektir. Görüntü işleme projelerinizde kullandığınız kameraların fps değerinin yüksek olması önemlidir. Şöyle örnekleyebiliriz: Bir fabrikada üretim bandından saniyede 10 adet ürün üretilerek geçtiğini düşünelim, geliştirdiğimiz yazılım ise geçen bu ürünleri yakalayıp analiz edecek ve kalite

kontrolü yapacak olsun. Seçtiğimiz kamera aygıtının fps değeri düşük olur ise geçen ilk 10 ürünü yakalasak bile ardından gelecek 10 ürünü kamera aygıtı kaçırabilir, çünkü üretim bandının hızı kamera aygıtının saniyede yakalayabildiği görüntü âdetinden fazladır.

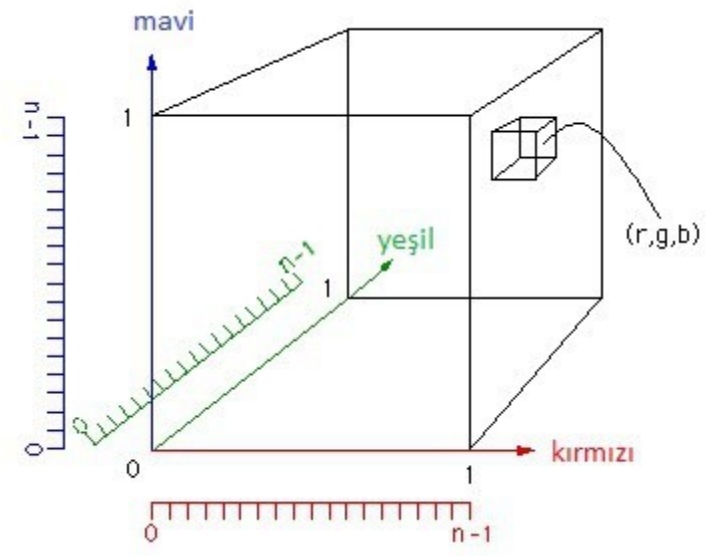
RENK UZAYLARI

Renk Uzayı

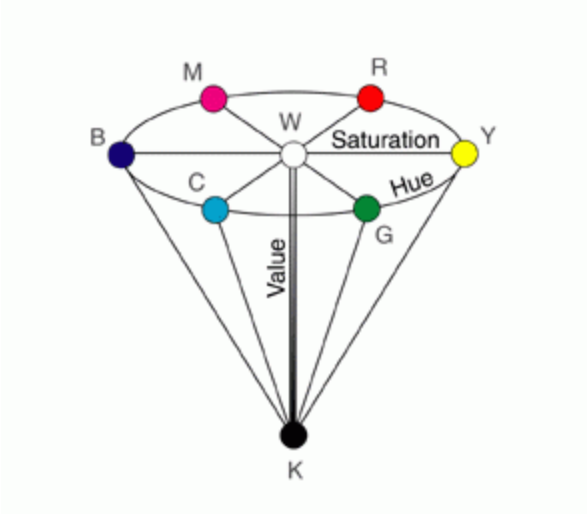
Renk çeşitliliğinin fazla olması nedeniyle bu renkleri gruplama ihtiyacı doğmuştur bu renkleri gruplamak ve standartlaştırmak için renk uzayı (color space) kavramı ortaya çıkmıştır. Her renk uzayı, renk kümesini tanımlamak için kendine özgü bir yapıya sahiptir. Örneğin siyah beyaz bir görüntüyü dijitalleştirmek için çok fazla kavrama gerek yoktur. Görüntü siyah ve beyaz olmak üzere 2 adet değişkene sahiptir. 300×300 boyutunda dijital siyah beyaz bir görüntü dijitalleştirilip renklendirilirken, 300×300 boyutunda bir dizi oluşturulur. Renklendirme işlemi için ise 2 adet değişken olduğu için 1 ve 0 yeterlidir. Fakat renkli bir resim üzerinde farklı renk tonları olacağı için 1 ve 0 ile bu görüntüyü tanımlamak yetersiz olacaktır. Bu farklı durumlar için çeşitli renk uzayları belirlenmiştir. En çok kullanılan ve kitap boyunca yer alacak örneklerde de kullanılan renk uzaylarına göz atalım.



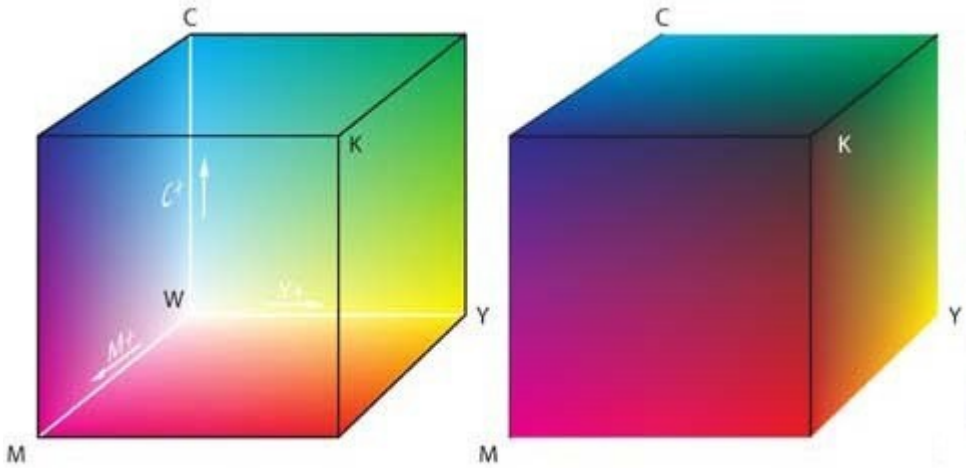
RGB Renk Uzayı: Bu renk uzayı Red Green Blue yani kırmızı, yeşil ve mavi renklerin baş harfi ile adlandırılmıştır. Renkler bir küp olarak tanımlanır bu tanımla sayesinde 3 değişkenli bir dizi elde edilir. Bu dizi elemanları olan hücreler yani pikseller, bir rengi tutabilmek için 3 renk olan kırmızı, yeşil ve mavinin belirli yoğunlukta karıştırılması ile elde edilen renk kodunu tutarlar.



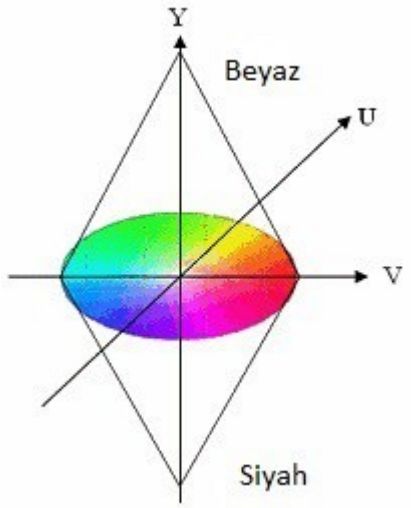
HSV Renk Uzayı: HSV Hue, Saturation, Value yani renk özü, doygunluk ve parlaklık olarak adlandırılmıştır. Anlaşıldığı üzere renk tanımlamalarını bu üç kavrama göre gerçekleştirir.



CMYK Renk Uzayı: Cyan, Magenta, Yellow, Key rengin kısaltmasıdır. Buradaki key siyah rengi temsil etmektedir. CMYK renk uzayı, dijital renk tanımlamaları için belirtilen bu dört rengi karıştırarak yapmaktadır.



YUV Renk Uzayı: Y Luminance, U Chrominance1, V Chrominance2 kısaltmasıdır. Y siyah – beyaz U ve V ise mavi tabanlı renklilik ve kırmızı tabanlı renkliliği temsil eder. Renkler bu üç kavram ile temsil edilerek oluşturulurlar.



Bizim için dijital görüntü işlemede en önemli olan renk uzaylarını tanımladık. Diğer renk uzaylarına ise OpenCV ile renk uzayları arası dönüşümü konusunda kullandıkça değineceğim.

DOSYA SİSTEMİNDEN GÖRÜNTÜ OKUMA

OpenCV ile dosya sisteminden görüntü dosyası okumak için Imgcodecs paketi içerisindeki imread metodu kullanılmaktadır. Bu konu öncesinde önemli bir sınıf olan Mat sınıfına göz atalım. Mat tipi olarak da adlandırabileceğimiz bu sınıf, görüntü barındırma işlevine sahiptir. Okunan görüntü dosyaları OpenCV kütüphanesin de mat sınıfında tutulmaktadır. Alınan görüntü renk uzayına göre farklı boyutlarda olabilecek dizi haline getirilerek bu tip içerisinde tutulur. Bir örnek yapalım ve dosya sisteminden görüntü okuyarak bunu mat tipi içerisinde tutalım.

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;

public class Giris {
    public static void main(String[] args) {
        //sistem kütüphanesini yükleme
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        //görüntüyü barındıracak nesne
        Mat imageArray;
        //görüntü dosyasını oku
        imageArray=Imgcodecs.imread("C:\\Users\\mesutpiskin\\resim.jpg");
        //mat nesnesinin satır ve sutun sayısı
        System.out.println(imageArray.rows());
        System.out.println(imageArray.cols());
    }
}
```

OpenCV kütüphanesini kullanmak için native kütüphaneyi yüklememiz gerekmekte, bunun için loadLibrary metotu ile bu işlemi yapıyoruz. Bir adet mat nesnesi tanımlıyoruz bu nesne ile okunacak olan görüntüyü üzerinde işlem yapmak için barındırabileceğiz. Imgcodecs.imread ile parametre olarak bulunduğu dosya adresi verilen resim.jpg resmini okuyoruz. Okunan bu resim dosyasını dijital olarak imageArray mat nesnesi tutmaktadır, artık bu nesne üzerinden o görüntüye ulaşabiliriz. Resim dosyası okunduğunda bu resmin eni ve boyu kadar mat nesnesi boyutlandırılır ve satır, sütün sayısını yazdırdığımızda resmin satır sütun sayısını elde edebiliriz, satır ve sütun çarpımı ise kaç adet piksele (hücreye) sahip olduğunu gösterecektir.

Yukarıdaki örneği OpenCV 3.x sürümü yerine OpenCV 2.x sürümü ile gerçekleştirecek olursak değişen tek şey imread metodunu çağırdığımız sınıf olacaktır. OpenCV 2.x sürümü görüntü okuma, yazma işlemleri Highgui ile yapılmaktaydı, 3.x sürümü ile birlikte bunlar Imgcodecs içerisine taşınmıştır.

```
import org.opencv.highgui.*;
...
imageArray=Highgui.imread("C:\\Users\\mesutpiskin\\resim.jpg");
...
```

VİDEO AYGITLARINDAN GÖRÜNTÜ OKUMA

Usb kameradan, ip kameradan, video dosyasından veya bir video kaynağından kayıt almak için videoio paketi içerisindeki VideoCapture sınıfı kullanılmaktadır. Bir örnek ile nasıl kullanıldığına göz atalım.

```java
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.videoio.VideoCapture;

public class Kamera {
    public static void main(String[] args) {
        // sistem kütüphanesini yükleme
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        // görüntüyü barındıracak nesne
        Mat imageArray = new Mat();
        // video aygıtlarına erişim sağlar
        VideoCapture videoDevice = new VideoCapture();
        // varsayılan video aygıtını başlatır
        videoDevice.open(0);
        // video aygıtı bağlantısı yapıldı mı?
        if (videoDevice.isOpened()) {
        // video aygıtından bir kare oku ve mat nesnesine yerleştir
            videoDevice.read(imageArray);
            // alınan dijital görüntü bilgileri
            System.out.println(imageArray.toString());
            // video aygıtını setbest bırak
            videoDevice.release();
        } else {
            System.out.println("Video aygıtına bağlanılamadı.");
        }
    }
}
```

VideoCapture sınıfından oluşturulan nesne, video aygıtını başlatmaktadır. Bu nesnenin open() metodu string ve int olarak parametre almaktadır. string olarak verilen parametre video dosyasının yolunu, örneğin C:\video.avi gibi veya yayın yapan bir ip kameranın adresini (http://192.168.1.2/mjpg/video.cgi) alabilir.

int tipinde verilen parametreler ise örnekte de olduğu gibi sisteme takılı olan kameraların numaralarıdır. Parametre olarak 0 verildiğinde sistemdeki varsayılan video aygıtına bağlanır. Sistemdeki başka bir kameraya erişim sağlanılmak isteniyorsa int olarak 1, 2, 3 gibi numarası verilerek bağlanılabilinir. isOpen() metodu ile de bağlantı durumu kontrol edilebilir. VideoCapture sınıfından oluşturulan nesnenin read() metodu ile de aygıttan bir kayıt alınır ve mat nesnesi içerisinde barındırılır. Kamera bağlantısını kapatmak için ise release() metodu kullanılır, bu metot kullanılmaz ve video aygıtı bağlantısı kesilmez ile

sürekli olarak bir görüntü akışı gerçekleştirilebilir. imageArray olarak adlandırılan mat nesnesinin toString() metodu ile satır, sütun veya adresi gibi temel bilgilerine erişilebilir. Video aygıtından alınan görüntü imageArray içerisine aktarılır ve bu nesneye ilişkin bilgiler ekranda gösterilir.

OpenCV Javada Resim Görüntüleme imshow Metodu

OpenCV içerisinde yer alan imshow() metodu parametre olarak verdiğiniz bir mat nesnesini resim tipine dönüştürerek bir pencere içerisinde ekranda gösterir. Bu metot, C++ ve Python dilleri tarafından desteklenmektedir. Yeni başlayanlar örneklerde sıklıkla gördüğü bu metodu Java da denediğinde metodun olmadığını görmüştür. Sıklıkla bu imshow() yerine hangi metot var sorusu gelmektedir, bu yüzden mat tipinin nasıl image tipine dönüştürüleceğini ve frame içerisinde gösterilebileceğini anlatacağım. Tabi bunları swing kütüphanesi yardımıyla gerçekleştiriyor olacağız. JavaCV kütüphanesini kullanırsanız imshow metodu burada yer almaktadır.

Mat tipi bildiğiniz üzere görüntünün renk değerlerinin, renk uzayı ile birlikte sayısal olarak ifade edilip matris olarak tutulduğu nesnelerdir. Öncelikli olarak bu değerleri okuyup image nesnesine daha sonrada bu image nesnesini frame nesnesi içerisinde görüntüleyeceğiz.

Öncelikle mat nesnesini buffered image nesnesine çevirelim

```
private static BufferedImage ConvertMat2Image(Mat kameraVerisi) {

      MatOfByte byteMatVerisi = new MatOfByte();
      // Ara belleðe verilen formatta görüntü kodlar
      Imgcodecs.imencode(".jpg", kameraVerisi, byteMatVerisi);
      // Mat nesnesinin toArray() metodu elemanlarý byte dizisine çevirir
      byte[] byteArray = byteMatVerisi.toArray();
      BufferedImage goruntu = null;
      try {
          InputStream in = new ByteArrayInputStream(byteArray);
          goruntu = ImageIO.read(in);
      } catch (Exception e) {
          e.printStackTrace();
          return null;
      }
      return goruntu;
  }
```

Daha sonra frame içerisinde görüntüleyelim. Eğer gerçek zamanlı bir iş yapacaksanız ve sürekli olarak ekrana yakalanan görüntüyü basacaksanız bu metodu her çağırdığınızda yeni bir frame içerisinde açacaktır bu durumda uygulamanın kilitlenmesi ile sonuçlanacaktır. Bu sorunu aşmak için aşağıdaki

yöntemi uygulayacağız ve bir kurucu metot yazarak gerekli nesnelerin tek bir defa oluşturulmasını sağlayacağız.

```
// Bir frame (çerçeve) oluşturur
    public static void PencereHazirla() {
        frame = new JFrame();
        frame.setLayout(new FlowLayout());
        frame.setSize(700, 600);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    // Resmi gösterecek label oluşturur
    public static void PushImage(Image img2) {
        // Pencere oluşturulmamış ise hazırlanır
        if (frame == null)
            PencereHazirla();
        // Daha önceden bir görüntü yüklenmiş ise yenisi için kaldırır
        if (lbl != null)
            frame.remove(lbl);
        icon = new ImageIcon(img2);
        lbl = new JLabel();
        lbl.setIcon(icon);
        frame.add(lbl);
        // Frame nesnesini yeniler
        frame.revalidate();
    }
```

Örnek kullanım;

```
public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        VideoCapture capture = new VideoCapture(0);
        Mat camImage = new Mat();
        if (capture.isOpened()) {
            while (true) {
                capture.read(camImage);
                PushImage(ConvertMat2Image(camImage));
            }} }
```

GÖRÜNTÜ STREAM ETME

Daha öncelerde opencv kurulumuna ve eclipse için yapılandırmasına bakmıştık ,şimdi sistemdeki kameralar üzerinden görüntü alma işleminin nasıl yapıldığına göz atacağız.Opencv 3.0 sürümünü kullanırsanız 3.0 ile bazı paketler ve sınıflar değişikliğe uğradı için problemler ile karşılaşabilirsiniz bu yüzden bu örnek için 2.x sürümlerini kullanmanızı öneririm.

3 Adet sınıfımız olacak bir tanesi kameraya bağlanmak bir tanesi resimleri yakalamak diğeri ise frame sınıfı webcam görüntüsünü form üzerinde görüntülemek için.

Mat2Image sınıfımız:

```
import java.awt.image.BufferedImage;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.highgui.Highgui;
import org.opencv.imgproc.Imgproc;

public class Mat2Image {
    Mat mat = new Mat();
    BufferedImage img;
    byte[] dat;
    public Mat2Image() {
    }
    public Mat2Image(Mat mat) {
        getSpace(mat);
    }
    public void getSpace(Mat mat) {
        this.mat = mat;
        int w = mat.cols(), h = mat.rows();
        if (dat == null || dat.length != w * h * 3)
            dat = new byte[w * h * 3];
        if (img == null || img.getWidth() != w || img.getHeight() != h
            || img.getType() != BufferedImage.TYPE_3BYTE_BGR)
                img = new BufferedImage(w, h,
                            BufferedImage.TYPE_3BYTE_BGR);
        }
        BufferedImage getImage(Mat mat){
            getSpace(mat);
            mat.get(0, 0, dat);
            img.getRaster().setDataElements(0, 0,
```

```
        mat.cols(), mat.rows(), dat);
        return img;
    }
    static{
         //Open cv native library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }
}
```

VideoCap sınıfı web kameraları yakalayacağımız sınıf

```
import java.awt.image.BufferedImage;
import org.opencv.core.Core;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;

public class VideoCap {
    static{
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }

    VideoCapture cap;
    Mat2Image mat2Img = new Mat2Image();

    VideoCap(){

        cap = new VideoCapture();
        //Bazı kameralarda sorun çıkartabileceği için boyutları oldukça küçük
olarak ayarladım kameranıza göre hd olarak değiştirebilirsiniz
        cap.set(Highgui.CV_CAP_PROP_FRAME_WIDTH, 50);
        cap.set(Highgui.CV_CAP_PROP_FRAME_HEIGHT, 50);
        /*0 yaparsanız varsayılan kamera çalışır birden fazla kamera var ise 1 , 2
,3 şeklinde numaları devam edektir. linux kullanıyorsanız lsusb ile bağlı usb
kameraları görüntüleyebilirsiniz.*/
        cap.open(0);
    }

    BufferedImage getOneFrame() {
```

```
        cap.read(mat2Img.mat);
        return mat2Img.getImage(mat2Img.mat);
    }
}
```

JFrame den kalıtılmış sınıfımızda frame üzerinde kamera görüntüsünü oynatacağız.

```
import java.awt.EventQueue;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

public class MyFrame extends JFrame {
    private JPanel contentPane;

  /**
  * Launch the application.
  */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    MyFrame frame = new MyFrame();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

  /**
  * Create the frame.
  */
    public MyFrame() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 650, 490);
```

```
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        new MyThread().start();
    }

    VideoCap videoCap = new VideoCap();

    public void paint(Graphics g){
        g = contentPane.getGraphics();
        g.drawImage(videoCap.getOneFrame(), 0, 0, this);
    }

    class MyThread extends Thread{
        @Override
        public void run() {
            for (;;){
                repaint();
                try { Thread.sleep(30);
                } catch (InterruptedException e) {    }
            }
        }
    }
}
```

İP KAMERADAN GÖRÜNTÜ OKUMA

İp kameralar sağladığı kullanım kolaylığı, sunucu üzerinde görüntü işlemek veya kamera sisteminin kurulduğu alan dışında görüntüleri yorumlamak amacıyla sıklıkla tercih edilirler. Bazı kamera aygıtları üzerinde kendi mikro işlemcisi olabilir ve görüntü aktarmaya ihityac duymadan burada işlenebilir. Bizim konumuz bu kameralar değil, yakaladığı görüntüyü üzerinde bulunan web sunucu aracılığıyla ağ üzerinde yayınlayabilen kameralar. Bu kameralar sabit ip adresi üzerinden yayın yaparlar. Benim kullandığım ip kamera 192.168.1.51 ip adresinden yayın yapıyor, bu adrese tarayıcı aracılığıyla bağlantığımızda 8080 veya sizin atadığınız varsayılanın dışında bir portt ile üzerindeki web sunucusuna erişiriz. Burada görüntü ayarları içerisinde format kısmını mjpg olarak değiştiriyoruz.

Birçok yüksek çözünürlüklü kamera farklı formatlarda yayın yapıyor ve bu formatları OpenCV aracılığıyla okumak zor olduğu için aygıtınız destekliyorsa mjpg yapmanızda fayda var. Kamera yayın için bir adres vercektir benim kameram http://192.168.1.51/mjpg/stream.cgi adresinden yayın yapıyor. Bu kısımdan sonra bu görüntüyü dosya sisteminden okur gibi okuyabiliyoruz. Şimdi ip kameraya bağlanalım ve bir kare yakalayalım.

```
public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        Mat imageArray = new Mat();
        VideoCapture videoDevice = new VideoCapture();
        // verilen adresteki yayını yakalar
        videoDevice.open("http://192.168.1.51/mjpg/stream.cgi");
        // video aygıtı bağlantısı yapıldı mı?
        if (videoDevice.isOpened()) {
        // video aygıtından bir kare oku ve mat nesnesine yerleştir
            videoDevice.read(imageArray);
            videoDevice.release();
        } else {
            System.out.println("IP kamera aygıtına bağlanılamadı.");
        }
    }
```

VideoCapture sınıfında bulunan set() metodu ile video aygıtları için bazı ayarları yapılandırabiliriz. int tipinde id ve paramtere değerini alır. get() metodu ile de değerleri görebilirsiniz. propId olarak tanımlanan int tipinde verilecek atanacak olan özelliğin id değeridir. Doğrudan int tipinde id değeri verilebileceği gibi 3.x öncesi sürümlerde Highgui 3.x sürümlerde ise Videoio içerisindeki int değişkenleride doğrudan kullanabilirsiniz. VideoCapture sınıfında kullanılabilecek bazı parametrelere göz atalım.

```
VideoCapture videoDevice = new VideoCapture(0);

//Özellik id ve değer olarak atanır
videoDevice.set(propId, value);

//Doğrudan değişlen ismi kullanarak da atanabilir
videoDevice.set(Videoio.CV_CAP_PROP_FRAME_WIDTH, 1024);
```

Kullandığınız IDE ile OpenCV paketleri içerisinde yer alan Videoio sınıfına eğerki eski bir OpenCV sürümü kullanıyorsanız Highgui sınıfına giderek buradaki parametrelere göz atabilirsiniz. Kullanabileceğiniz tüm parametreler ve id değerleri burada yer almaktadır. Veya aşağıdaki bağlantıdan direk olarak videoio sınıfına göz atabilirsiniz.

http://docs.opencv.org/java/3.0.0/org/opencv/videoio/Videoio.html

NOT: Paket dâhilindeki sınıflara göz atmak için sınıf üzerine sağ tıklayarak Open Declaration demeniz yeterli olacaktır.

CV_CAP_PROP_POS_MSEC

Video dosyasından okuma yaparken mili saniye cinsinden o anki video zamanını yönetir.

CV_CAP_PROP_FRAME_WIDTH

Okunan görüntünün genişliğini yönetir. Görüntü genişliği değiştirilebilir yada okunan görüntünün genişliği öğrenilebilir.

CV_CAP_PROP_FRAME_HEIGHT

Okunan görüntünün yüksekliğini yönetir. Görüntü yüksekliği değiştirilebilir yada okunan görüntünün yüksekliği öğrenilebilir.

CV_CAP_PROP_FPS

Görüntünün saniyedeki kare hızını ifade eden FPS değeri değiştirilebilir.

CV_CAP_PROP_FOURCC

Kodek değiştirilebilir. Kodek dört karakterlik kod ile ifade edilir.

CV_CAP_PROP_FRAME_COUNT

Görüntünün çerçeve (frame) sayısına erişilebilir.

CV_CAP_PROP_FORMAT

Mat nesnesinin formatı değiştirilebilir.

CV_CAP_PROP_BRIGHTNESS

Görüntünün parlaklık değeri değiştirilebilir. Bu parametre sadece kamera aygıtları için geçerlidir.

CV_CAP_PROP_CONTRAST

Görüntünün kontrast değeri değiştirilebilir. Bu parametre sadece kamera aygıtları için geçerlidir.

CV_CAP_PROP_SATURATION

Görüntünün doygunluk değeri değiştirilebilir. Bu parametre sadece kamera aygıtları için geçerlidir.

CV_CAP_PROP_HUE

Görüntünün renk tonu değiştirilebilir. Bu parametre sadece kamera aygıtları için geçerlidir.

CV_CAP_PROP_CONVERT_RGB

Görüntünün RGB renk uzayına dönüştürülmesi gerekiyor ise kullanılır.

CV_CAP_PROP_WHITE_BALANCE

Görüntünün beyaz dengesi değiştirilebilir.

CV_CAP_PROP_ISO_SPEED

Kamera aygtının ISO hızı değiştirilebilir. Tüm aygıtlar için desteklenmemektedir.

CV_CAP_PROP_BUFFERSIZE

Ara belleğe alınacak olan çerçeve (frame) miktarı değiştirilebilir.

GÖRÜNTÜ YAZMA (VİDEOWRİTE)

Daha önceki örneklerde dosya sisteminden ve video aygıtlarından nasıl görüntü okunabileceğine ve bu görüntüleri nasıl barındırabileceğimize bakmıştık. Okunan görüntünün tekrardan yazılması işlemi ise Imgcodecs sınıfı içerisindeki imwrite() metodu ile yapılmaktadır. Bu metot parametre olarak dosya adı ve mat tipinde görüntü dizisi almaktadır.

```
Imgcodecs.imwrite("dosyadi.jpg", matArray);
```

Video olarak yazma işlemi için ise VideoWrite sınıfı kullanılmaktadır. Open metodu kullanımında filename dosya dizini ve adı, fourcc yerine codec, fps yerine fps oranı, frameSize yerine Size olarak en boy oranı verilmektedir. Write metodu ise bir mat tipinde nesne almaktadır. Release metodu ile ise işlem sonlandırılabilmektedir.

```
VideoWriter videoYazici=new VideoWriter();
videoYazici.open(filename, fourcc, fps, frameSize);
videoYazici.write(matImage);
videoYazici.release();
```

PİKSEL İŞLEMLERİ

Piksel kavramını daha önce açıklamıştık, pikseller mat nesnesi içerisindeki dizi elemanlarına karşılık gelmektedir. Bir görüntü üzerinde işlem yapmak istediğimizde dizideki elemanları kullanmamız gerekmektedir. OpenCV içerisinde yer alan birçok metot piksel işlemlerini kendisi yapmaktadır. Örneğin bir görüntüyü kopyalamak istediğimizde copy metodunu kullanabiliriz fakat bu metotların nasıl çalıştığını anlamak için veya kendi algoritmanızı geliştirmek zorunda kaldığınızda bu bilgiler işinize yarayacaktır. Basit bir uygulama yazalım ve bu uygulama kameradan okunan görüntüyü mat nesnesi içerisinde tutalım ve bu nesneyi bir başka mat nesnesi içerisine kopyalayarak dosya sistemine kaydedelim. Bu örnek ile bir piksele nasıl ulaşabileceğimizi de öğrenmiş olacağız.

ÖN BİLGİ: Her görüntü renk uzayına göre çeşitli formatlarda, dijital olarak dizi şeklinde tutulur. Her piksel bir dizi elamanıdır ve görüntü üzerinde yapılan her işlem için dizi elemanları arasında dönülür ve istenilen işlem piksel bazında gerçekleştirilir. Görüntü işlemenin temelinde bu vardır.

```
public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        Mat imageArray = new Mat();
        VideoCapture videoDevice = new VideoCapture();
        videoDevice.open(0);
        if (videoDevice.isOpened()) {
            videoDevice.read(imageArray);
            videoDevice.release();
        } else {
            System.out.println("Video aygıtına bağlanılamadı.");
            return;
        }
        //Okunan görüntüyü kopyalamak için yeni bir mat nesnesi
        Mat newimageArray=new Mat();
        /*
         * Yeni nesnenin satır ve sutun sayısını alınan görüntünün satır sutun
sayısına eşitliyoruz
         * create  metodu ile oluşturakacaj nat nesnesinin satır ve sütününu
veriyoruz. */
        newimageArray.create(imageArray.rows(), imageArray.cols(), CvType.CV_8UC1);
        //Döngü ile alınan görüntü dizisinin tüm elemanları arasında dönüyoruz
        for(int i=0; i<imageArray.rows();i++)
        {
            for(int j=0;j<imageArray.cols();j++)
            {
            /* put ile  verilen indise yani piksele değer atanır
             * get ile  verilen indisdeki piksel değeri okunur
             * /
                newimageArray.put(i, j, imageArray.get(i, j));
            }
        }
        Imgcodecs.imwrite("KopyaGoruntu.jpg", newimageArray);
        System.out.println("Görüntü dosya sistemine yazıldı.");
    }
```

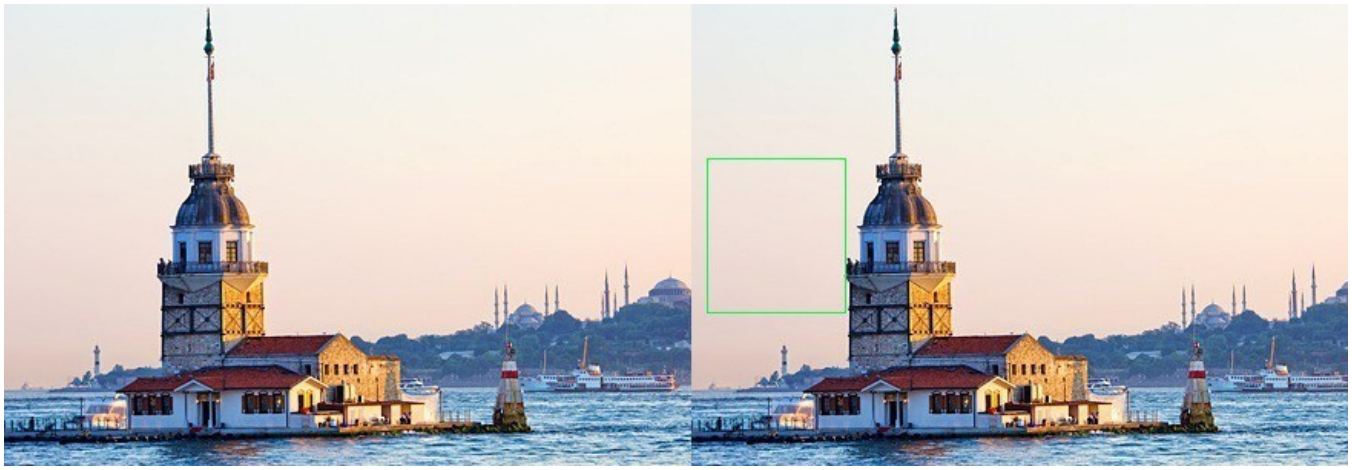
Bir piksele yani dizi elemanına erişmek için get() metodu, belirli indisteki bir piksele değer yazmak için ise set() metodu kullanılır.

MATRİS ÜZERİNDE ÇİZİM İŞLEMLERİ

Mat nesnesi üzerinde çizim yapmak yani dikdörtgen, daire, kare, çizgi çizmek veya metin yazdırmak gibi işlemler için imgproc sınıfı kullanılmaktadır. Bu sınıf içerisinde yer alan metotlardın farklı parametrelere sahip aşırı yüklenmiş (override) alternatifleri bulunmaktadır.

Dikdörtgen çizmek için imgproc içerisindeki rectangle metodunu kullanacağız. Aşağıdaki örnekte dosyadan bir resim dosyası okunuyor ve bu resim dosyası üzerine belirlenen konuma belirlenen renk ile bir kare çiziliyor. İşlemin ardından düzenlenen görüntü tekrardan dosya dizinine yazılıyor.

```
public class Dikdortgen {

    public static void main(String[] args) {

    System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    Mat goruntuDizisi=new Mat();
    goruntuDizisi=Imgcodecs.imread("C:\\kiz_kulesi.jpg");
    /* rectangle metodu paramatetre olarak, üzerinde çizim yapılacak bir mat
nesnesi
     * dikdörtgen çizimi için gerekli olan 4 köşenin koordinatı ve rengini
almaktadır.
     * */
    Imgproc.rectangle(goruntuDizisi, new Point(10,100), new Point(100,200),new
Scalar(76,255,0));
    Imgcodecs.imwrite("C:\\Yeni_kiz_kulesi.jpg", goruntuDizisi);
    System.out.println("Düzenlenen görüntü dosya sistemine yazıldı.");
    }
}
```



Point tipinde dikdörtgenin köşe koordinatlarını (x,y), scalar ile de dikdörtgenin rengini veriyoruz. Buradaki önemli nokta renkleri belirtmek için kullandığımız scalar sınıfı. Bu sınıf core paketi içerisinde bulunmaktadır ve RGB renkleri belirtmek için kullanılabilir. Scalar üç parametre almaktadır ve bu parametreler tahmin edebileceğiniz üzer RGB (Kırmızı Yeşil Mavi) değerleridir. Fakat RGB değerlerini BGR (Mavi Yeşil Kırmızı)olarak tersten vermemiz gerekmektedir.

Çizgi çizmek için ise line() metodu bulunmaktadır. Üzerinde işlem yapılmak için bir mat nesnesi ve point tipinde x, y koordinatları ve rengini parametre olarak almaktadır.

Imgproc.line(mat, point, point, color);

Daire çizmek için circle() metodu bulunmaktadır. Parametre olarak işlem yapılacak mat tipinde görüntü, point tipinde merkez koordinat, int tipinde yarıçap ve renk almaktadır.

Imgproc.circle(mat, point, int, color);

Çokgen (Poligon) çizmek için polylines() metodu bulunmaktadır. Parametre olarak işlem yapılacak mat tipinde görüntü nesnesi, MatOfPoint tipinde liste olarak köşe koordinatları, bool tipinde açıklık kapalılık durumu ve renk almaktadır. Buradaki isClosed kapalı olarak atanırsa her eğrinin son köşesine bir çizgi çizer.

Imgproc.polylines(mat, point, isClosed, color);

Yazı yazmak için ise putText() metodu bulunmaktadır. Parametre olarak işlem yapılacak mat nesnesi, yazılacak olan string metin, point olarak koordinat, metinsel ifadenin fontu, double tipinde fontun ölçeği ve renk almaktadır.

Imgproc.putText(mat, text, point, fontFace, fontScale, color);

Ok çizmek için ise yine çizgi çizmek için kullanılana benzer arrowedLine() metodu bulunmaktadır. Bu metot bir mat nesnesi, point tipinde 2 adet koordinat ve renk almaktadır.

Imgproc.arrowedLine(mat, point1, point2, color);

GÖRÜNTÜ KIRPMA

Bir önceki yazı olan matris üzerinde çizim işlemlerinde kız kulesi üzerine bir dikdörtgen çizmiştik, şimdi bir örnek yapalım ve bu geometrik şekillerin kullanım alanlarını daha iyi kavrayalım. Bir dikdörtgen nesnesi oluşturacağız ve okunana görüntüyü bu dikdörtgen boyutlarında kırparak görüntü içerisinden çıkartacağız.

```
public static void Kirp(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        Mat goruntuDizisi = new Mat();
        goruntuDizisi = Imgcodecs.imread("C:\\kiz_kulesii.jpg");
        //Dikdörtgen olluşturuyoruz ve koordinatlarını belirliyoruz
        Rect dikdortgen=new Rect(new Point(10,100),new Point(100,200));
        //Yeni bir mat nesnesi oluşturuyoruz ve okunan görüntüye dikdörtgen
ebatlarında kırpma işlemi uyguluyoruz
Mat yeniGoruntu=new Mat(goruntuDizisi,dikdortgen);
Imgcodecs.imwrite("C:\\Yeni_kiz_kulesi.jpg", yeniGoruntu);

    }
```

JAVA GUI UYGULAMALAR

Uygulamalarda görsellik oldukça önemlidir, geliştirmekte olduğunuz OpenCV projesi müşteri odaklı bir yazılım olabilir ve bu yazılımda yapılan işlemlerin yani işlenen görüntünün, uygulanan filtrelerin veya üzerine çizim yapılan görüntünün anlık olarak kullanıcıya gösterilmesi gerekebilir. Java da GUI (Grafiksel

kullanıcı arabirimi) uygulamalar için genellikle swing tercih edilir fakat swing kullanmak zorunda değilsiniz bu bölümde öğreneceklerinizi diğer kütüphaneler ile de kullanabilirsiniz bu konuda bir sınırlama yoktur.

EK BİLGİ: Netbeans kullanıyorsanız ayrıca bir swing kurulumuna ihtiyacınız yoktur projenize new/frame diyerek bir çerçeve ekleyerek tasarım aracını kullanarak grafiksel ara yüzler geliştirebilirsiniz. Eclipse kullanıyorsanız Help menüsünden Install new software.. seçeneğine tıklayarak açılan penceredeki Work with kısmından –All Available Sites– seçiniz, ardından General Purpose Tools seçeneği altından swing bileşenini seçerek kurabilirsiniz.

OpenCV içerisinde GUI işlemleri için bazı sınıflar bulunmaktadır, daha çok C++ geliştiricilerine yönelik olarak eklenen bu sınıflar ve metotlar ile pencere oluşturma ve resim görüntüleme işlemleri yapılabilmektedir. Java ile GUI uygulamalar için çok fazla kütüphaneler olduğu için projelerinizde bunları kullanmanız doğru bir seçim olacaktır zira uygulamalar sadece video oynatmak ve resim görüntülemekten ibaret olmayacaktır.

OpenCV de görüntüleri dijital olarak mat nesneleri içerisinde barındırıyoruz. Bu dijital görüntüyü kullanıcıya göstermek için bu nesne içerisindeki verileri (pikselleri) okuyup, yorumlayıp bir resim haline getirmemiz gerekiyor. Oluşan bu resmi ise bir frame veya panel ile kullanıcıya gösterebiliriz veya okunan görüntüler dosya sistemine yazılır ve Java'daki io sınıfı ile bu görüntü okunarak görüntülenebilir.

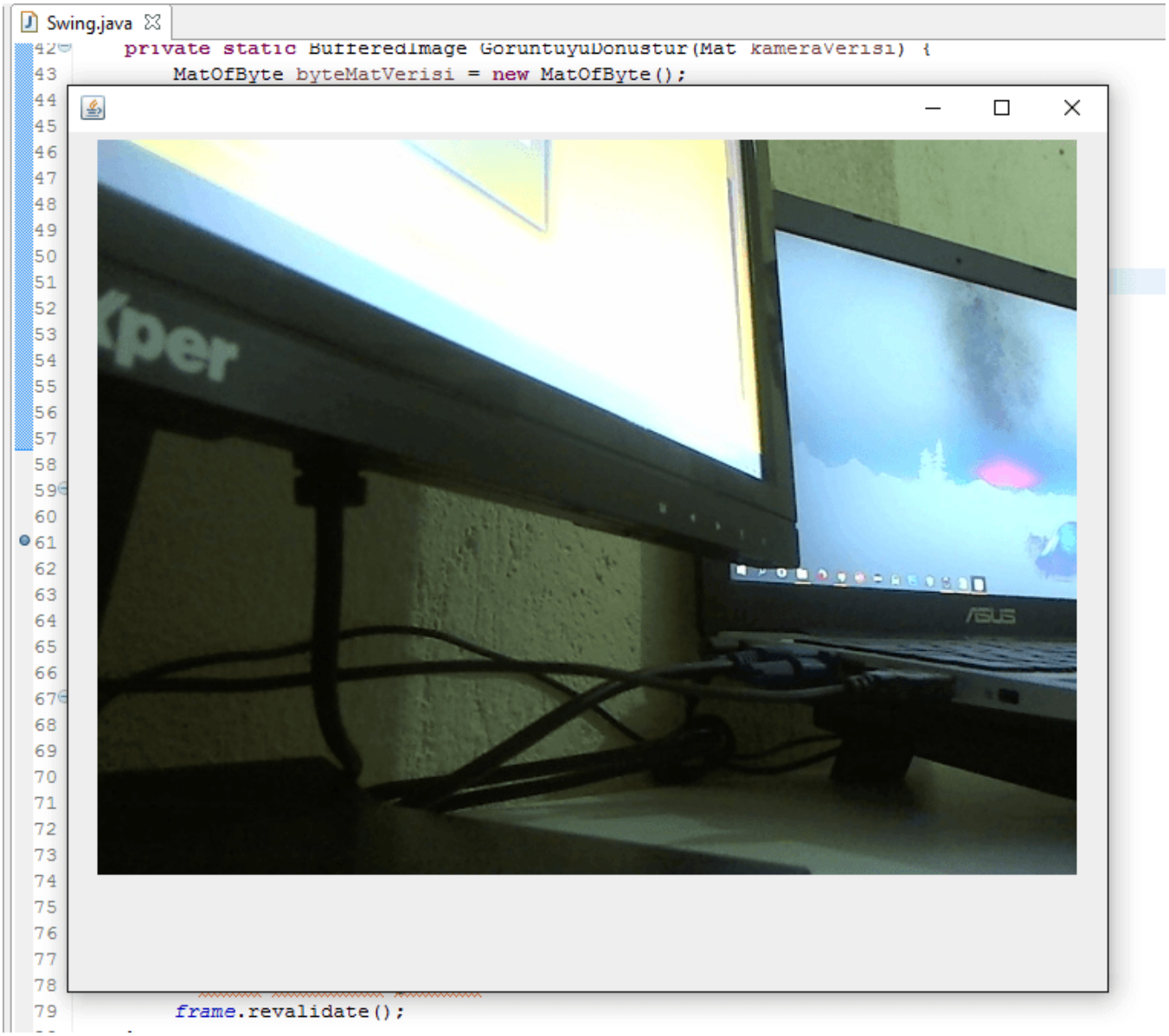
Bir örnek yapalım ve kameradan alınan görüntüleri anlık olarak frame üzerinde görüntüleyelim.

```
import java.awt.FlowLayout;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.InputStream;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.videoio.VideoCapture;

public class Swing {

    static JFrame frame;
    static JLabel lbl;
    static ImageIcon icon;
```

```
public static void main(String[] args) {
    System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    Mat kameraVerisi = new Mat();
    VideoCapture videoDevice = new VideoCapture();
    videoDevice.open(0);
    if (videoDevice.isOpened()) {
    //Sonsuz bir döngü ile sürekli olarak görüntü akışı sağlanır
        while (true) {
            videoDevice.read(kameraVerisi);
    //Yakalanan görüntüyü önce dönüştür ve frame içerisine yükle
            GoruntuYukle(GoruntuyuDonustur(kameraVerisi));
        }
    } else {
        System.out.println("Video aygıtına bağlanılamadı.");
        return;
    }
}
//Mat nesnesini image tipine dönüştür
private static BufferedImage GoruntuyuDonustur(Mat kameraVerisi) {
    MatOfByte byteMatVerisi = new MatOfByte();
    //Ara belleğe verilen formatta görüntü kodlar
    Imgcodecs.imencode(".jpg", kameraVerisi, byteMatVerisi);
    //Mat nesnesinin toArray() metodu elemanları byte dizisine çevirir
    byte[] byteArray = byteMatVerisi.toArray();
    BufferedImage goruntu = null;
    try {
        InputStream in = new ByteArrayInputStream(byteArray);
        goruntu = ImageIO.read(in);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return goruntu;
}
```

```
//Bir frame (çerçeve) oluşturur
    public static void PencereHazirla() {
        frame = new JFrame();
        frame.setLayout(new FlowLayout());
        frame.setSize(500, 600);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    //Resmi gösterecek label oluşturur
    public static void GoruntuYukle(Image img2) {
        //Pencere oluşturulmamış ise hazırlanır
        if (frame == null)
            PencereHazirla();
        //Daha önceden bir görüntü yüklenmiş ise yenisi için kaldırır
        if (lbl != null)
            frame.remove(lbl);
        icon = new ImageIcon(img2);
        lbl = new JLabel();
        lbl.setIcon(icon);
        frame.add(lbl);
        //Frame nesnesini yeniler
        frame.revalidate();
    }
}
```

Daha önce yaptığımız gibi video aygıtından bir görüntü aldık, anlık olarak akış sağlamak için bu görüntü alma işlemini döngü içerisine aldık bu sayede sürekli olarak video aygıtından görüntü akışı sağlandı. Alınan görüntü mat nesnesinden belleğe alındı ve image olarak dönüştürüldü, bir dönüşüm işlemi yapmadan mat tipi olarak swing veya diğer Java GUI kütüphaneleri bu görüntüyü işleyemezler.



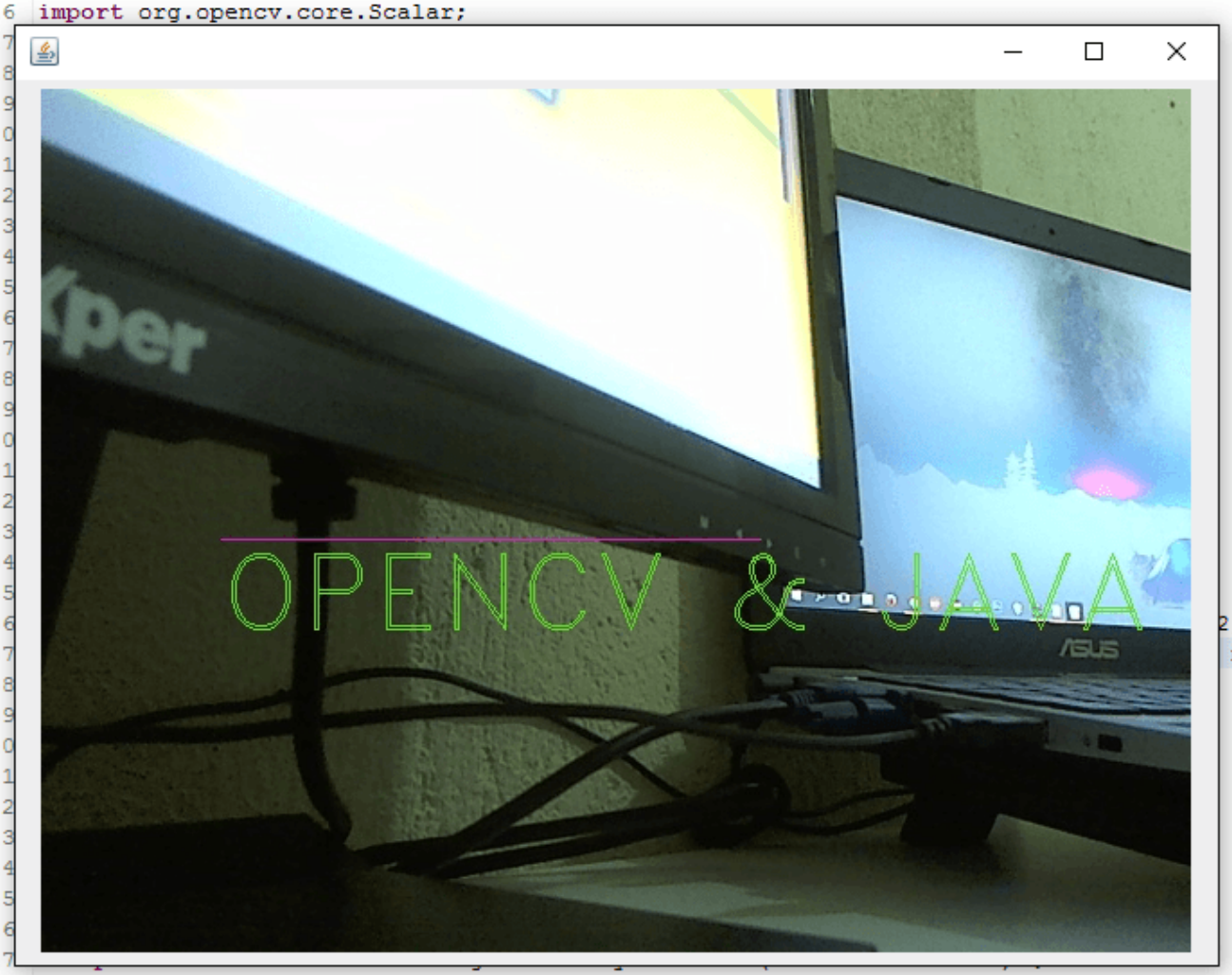
Anlık olarak bir görüntü akışı elde ettik. Bu akış bir biri ardına devam eden görüntülerin birleşmesi ile elde edildi ve bu sayede görüntü üzerinde istediğimiz değişikliği yapabiliriz. Daha önce gördüğümüz, görüntü üzerine şekil çizme veya metin yazdırma işlemlerini de uygulayabiliriz. Bir örnek yapalım ve akıştaki görüntülere müdahale ederek bazı değişiklikler yapalım.

```
public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        Mat kameraVerisi = new Mat();
        VideoCapture videoDevice = new VideoCapture();
        videoDevice.open(0);
        if (videoDevice.isOpened()) {
        //Sonsuz bir döngü ile sürekli olarak görüntü akışı sağlanı
            while (true) {
```

```
                videoDevice.read(kameraVerisi);
                //Metin yaz
                Imgproc.putText(kameraVerisi, "OPENCV & JAVA", new
Point(100,300),2, 2, new Scalar(27,228,94));
                //Çizgi çiz
                Imgproc.line(kameraVerisi, new Point(100,250), new Point(400,250),
new Scalar(207,28,228));
        //Yakalanan görüntüyü önce dönüştür ve frame içerisine yükle
                GoruntuYukle(GoruntuyuDonustur(kameraVerisi));
            }
        } else {
            System.out.println("Video aygıtına bağlanılamadı.");
            return;
        }
    }
```

Swing2.java

```
import org.opencv.core.Scalar;
```



Video dosyasından ve ip kameradan görüntü oynatmak için ise daha öncede gördüğümüz gibi VideoCapture sınıfından oluşturulan nesnenin, kurucu metodunda veya Open metodunda video dosyasının veya ip kameranın adresini vermeniz yeterli olacaktır.

RENK UZAYLARI ARASI DÖNÜŞÜMLER

Daha önce temel dijital görüntü işleme kavramları bölümünde renklere ve renk uzaylarına değinmiştir. OpenCV'de birçok renk uzayı desteklenmektedir ve bunlar arasında dönüşüm yapılabilmektedir. Bu bölümde OpenCV ile bu renk uzayları arasında dönüşüm işlemleri için Imgproc sınıfı içerisinde cvtColor() metodu bulunmaktadır. cvtColor metodu parametre olarak iki adet mat nesnesi ve dönüşüm yapılacak olan renk uzayını almaktadır.

```
Imgproc.cvtColor(srcMat, dstMat, code);
```

srcMat: kaynak bir mat nesnesi yani dönüşümü yapılacak olan görüntü, dstMat: hedef mat nesnesi yani dönüşüm sonucunda oluşacak yeni renk uzayına sahip görüntü, code ise hangi renk uzayları arasında dönüşüm yapılacağıdır. Desteklenen bazı renk uzayları aşağıdaki tabloda yer almaktadır.

Kaynak Renk Uzayı 2 Hedef Renk Uzayı

- COLOR_RGB2BGR
- COLOR_RGB2BGRA
- COLOR_RGB2GRAY
- COLOR_GRAY2RGB
- COLOR_RGB2HLS
- COLOR_HSV2RGB
- COLOR_RGB2HSV
- COLOR_RGB2Luv
- COLOR_HSV2RGB
- COLOR_RGB2YUV
- COLOR_RGB2Lab

RGB HSV Renk Dönüşümü

RGB (Red Greeb Blue – Kırmızı Yeşil Mavi) renk uzayından HSV (Hue Saturation Value – Renk tonu Doygunluk Değer) renk uzayına dönüşüm;

```
public static void main(String[] args) {
    System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    Mat kaynakGoruntu=new Mat();
    kaynakGoruntu=Imgcodecs.imread("C:\\kizkulesi.jpg");
    Mat hedefGoruntu=new Mat();
```

```
      //Okunanan RGB görüntüyü HSB renk uzayına çevirerek hedefGoruntu mat
nesnesine atar
    Imgproc.cvtColor(kaynakGoruntu, hedefGoruntu, Imgproc.COLOR_RGB2HSV);
        Imgcodecs.imwrite("C:\\ kizkulesiHSV.jpg", hedefGoruntu);
}
```

RGB GRAY Renk Dönüşümü

GRAY renk uzayı ile renkler, siyahın ve beyazın tonlarında yani gri olarak tanımlanır.

```
Imgproc.cvtColor(kaynakGoruntu, hedefGoruntu, Imgproc.COLOR_GRAY2RGB);
```

Sistemdeki kameradan alınan görüntü üzerine de aynı işlemi uygulayarak, akış halindeki bir görüntünün renk uzayı değiştirilebilir. Bazı renk uzayları arasındaki dönüşüm kaliteli bir sonuç vermeyebilir, her renk uzayı farklı geometrik biçimlerde ifade edilmektedir, bu ifade şekli matematiksel olarak bazı renk uzayları arasındaki dönüşüme engel teşkil etmektedir.

MORFOLOJİK OPERATÖRLER VE FİLTRELER

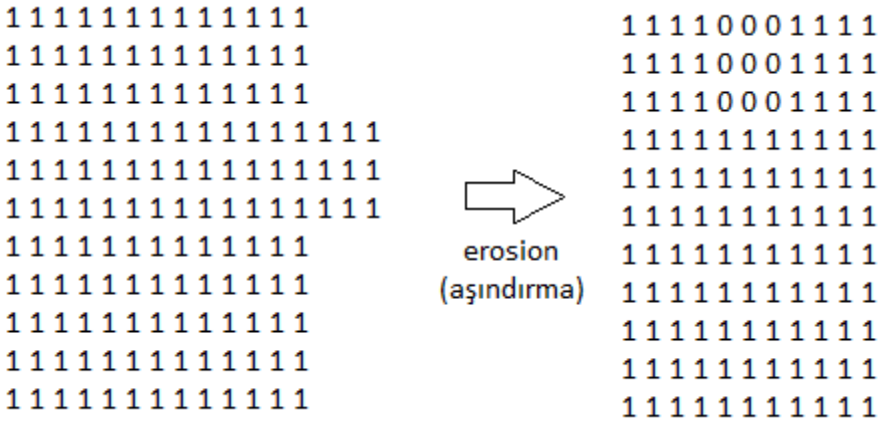
Morfoloji İngilizce tabiriyle morphology şekil bilim olarak tanımlanmaktadır. Başlı başına bilim olan bu alanı tüm yöntemleri ile OpenCV Kütüphanesi içerisine taşımak elbette ki mantıklı bir seçim değildir bu yüzden ihtiyaç duyulabilecek bazı teknikler aktarılmıştır. OpenCV içerisinde morfolojik işlem operatörleri Imgproc içerisinde bulunmaktadır.

Morfoloji'nin bir şekil bilimi olduğunu söylemiştik, çalışılan görüntü üzerindeki şekillerin yorumlanması, analiz edilmesi, istenilen bilginin çıkartılması, inceltme, görüntü sıkıştırma, köşe analizi, bozuk görüntü onarma (eksik veya fazla piksellerin çıkarılması, eklenmesi), dokuların tespiti gibi işlemlerde sıklıkla başvurulmaktadır.

- Erosion (Aşındırma)
- Dilation (Yayma – Genişletme)
- Opening (Açınım)
- Closing (Kapanım)
- Morphological Gradient
- Top Hat
- Black Hat

Erosion (Aşındırma) Morfolojik Operatör

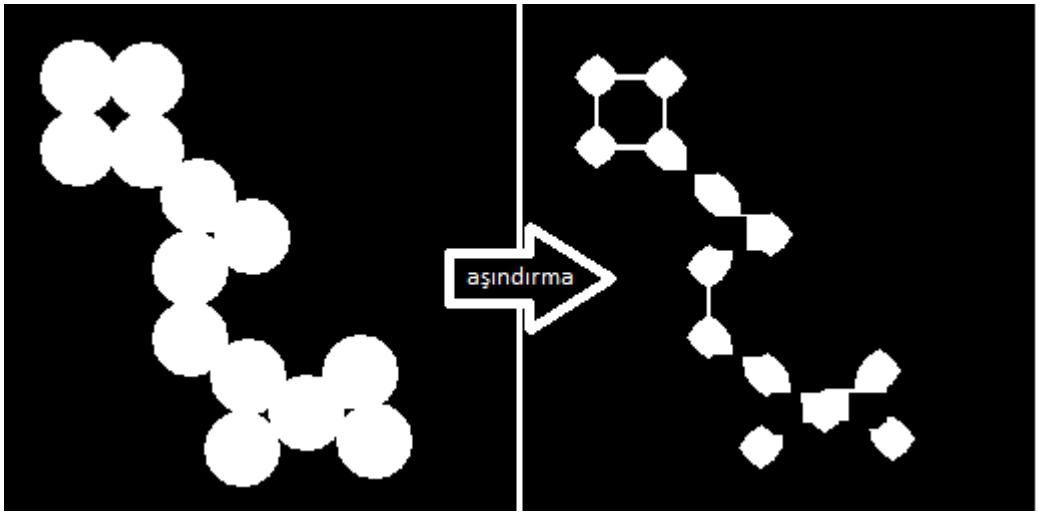
Bu operatör görüntü üzerinde bir aşındırma işlemi uygular. Parametrelere göre belirtilen alan içerisindeki pikseller aşındırılır ve gürültülü olarak adlandırılan bozuk olan görüntü, gürültüden arındırılarak temizlenir. Bütün bu olaylar matematiksel olarak tanımlanmıştır ve diziler üzerinde gerçekleştirilir. Aşağıdaki görseller yardımı ile nasıl çalıştığını somut olarak görelim. İlk görüntü dizisi aşındırma ile gürültüden arındırılmaktadır.



Erosion işlemi için kullanacağımız metot erode(), bu metot imgproc içerisinde yer almaktadır. Erode metodunun üç adet overloadı bulunmaktadır. Kullandığımız parametre olarak giriş mat nesnesi, işlem sonucunu atamak için çıkış mat nesnesi ve yapılandırma için bir nesne almaktadır. Bu nesne yapısal element olarak adlandırılır (Structuring Element) ve yayma işleminin şeklini belirler. Yapısal elementin merkez noktası üzerine giriş görüntüsünün pikselleri bu noktaya oturtularak oluşturulur. Bu şekiller Imgproc içerisinde tanımlanmışlardır, aşağıdaki örnekte MORPH_ERODE kullandık.

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

public class Erosion {
    public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        Mat girisGoruntu=new Mat();
        girisGoruntu=Imgcodecs.imread("C:\\1.png");
        Mat cikisGoruntu=new Mat();
           //Aşındırma işlemi
Imgproc.erode(girisGoruntu, cikisGoruntu,
Imgproc.getStructuringElement(Imgproc.MORPH_ERODE, new Size(15,15)));

        Imgcodecs.imwrite("C:\\2.png", cikisGoruntu);
    }
}
```

Verilen parametreler doğrultusunda giriş görüntüsü üzerinde aşındırma operatörü kullanılmıştır. 1.png olarak adlandırılan görüntüyü aşağıda solda görmektesiniz bu input olarak tanımlanan girisGoruntu mat nesnesi içerisinde tutulan, cikisGoruntu olarak tanımlanan mat nesnesini oluşturulan ve 2.png olarak adlandırılan görüntüyü ise sağda görmektesiniz.

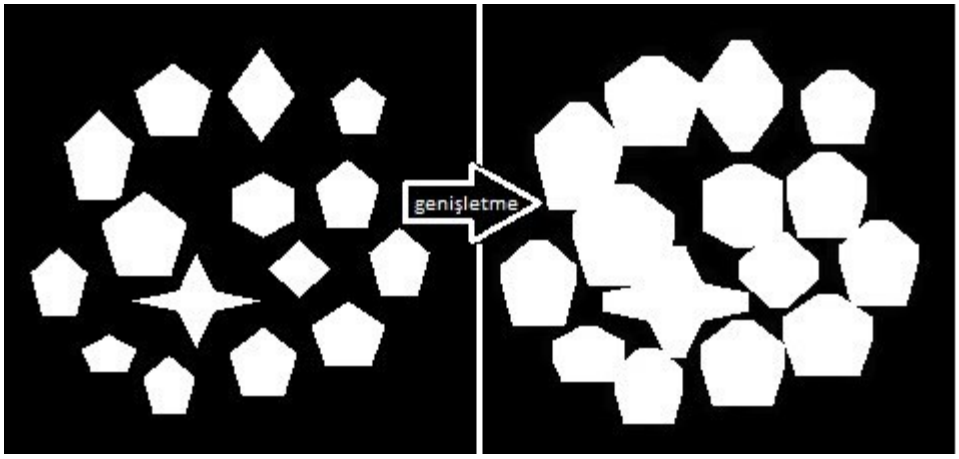


Dilation (Yayma – Genişletme) Morfolojik Operatör

Bu operatör giriş olarak verilen görüntü üzerinde parametreler ile verilen alan içerisindeki sınırları genişletmektedir, bu genişletme sayesinde piksel gurupları büyür ve pikseller arası boşluklar küçülür. OpenCV dilation operatörü için Imgproc içerisinde dilate() operatörü bulunmaktadır. Bu metot parametre olarak giriş görüntüsü olacak bir mat nesnesi, çıkış görüntüsü için ikinci bir mat nesnesi ve yapısal element almaktadır.

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

public class Dilation {
    public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        Mat girisGoruntu=new Mat();
        girisGoruntu=Imgcodecs.imread("C:\\ 1.jpg");
        Mat cikisGoruntu=new Mat();
        //dilate (genişletme) operatörü
                Imgproc.dilate(girisGoruntu,
cikisGoruntu,Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(25,25)));
        Imgcodecs.imwrite("C:\\2.jpg", cikisGoruntu);
    }
}
```

Aşağıdaki çıktıda sol tarafta bulunan görüntü 1.jpg olarak adlandırılan giriş mat nesnesi, sağındaki ise 2.jpg olarak işlem sonucunda oluşturulan çıktı görüntü. Gördüğünüz üzere giriş görüntüsünde bulunan beyaz şekiller dilation operatörü uygulandığında bir birlerine yaklaşmışlardır. Burada önemli nokta zeminin, siyah nesneler beyaz olması ve yapısal element. Yapısal element üzerindeki değişikler ile aralarındaki mesafe daha da azaltılıp birleştirilebilirdi.



Diğer operatörleri kullanmak için daha önceki örneklerde yaptığımız gibi erode ve dilate metotlarını kullanarak gerçekleştirebilirsiniz fakat OpenCV içerisinde morfolojik operatörleri yönetmek için yazılmış bir metot hazırda bulunmaktadır.

Imgproc içerisinde yer alan morphologyEx() ile operatörler yönetilebilmektedir. Bu metot parametre olarak giriş görüntüsü için mat nesnesi, işlem sonucu için bir mat nesnesi, uygulanacak olan operatör ve yapısal element almaktadır.

Opening (Açınım) Morfolojik Operatör

(Imgproc.MORPH_OPEN)

Erosion ve dilation operatörlerinin görüntü üzerine birlikte uygulanması ile gerçekleşir. Öncelikli olarak erosion operatörü uygulanır ve ardından dilation operatörü uygulanır.

```
Imgproc.morphologyEx(girisGoruntu, cikisGoruntu, Imgproc.MORPH_OPEN,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(25,25)));
```

Closing (Kapanım) Morfolojik Operatör

(Imgproc.MORPH_CLOSE)

Görüntüye dilation operatörü uygulanır ve ardından Erosion operatörü uygulanır.

```
Imgproc.morphologyEx(girisGoruntu, cikisGoruntu, Imgproc.MORPH_CLOSE,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(25,25)));
```

Morphological Gradient Morfolojik Operatör

(Imgproc.MORPH_GRADIENT)

Dilation ve Erosion operatörü arasındaki farktır. Nesnelerin ana hatlarını belirlemek için kullanılır. Sınır çizgilerini tam hatlarıyla belirlemek için yapısal element, görüntüye göre özelleştirilmelidir.

```
Imgproc.morphologyEx(girisGoruntu, cikisGoruntu, Imgproc.MORPH_GRADIENT,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(25,25)));
```

Top Hat Morfolojik Operatör

(Imgproc.MORPH_TOPHAT)

Bu operatör giriş olarak verilen görüntüden, opening (açınım) operatörü uygulanmış halini çıkarır.

```
Imgproc.morphologyEx(girisGoruntu, cikisGoruntu, Imgproc.MORPH_TOPHAT,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(25,25)));
```

THRESHOLDİNG (EŞİKLEME)

Giriş olarak verilen görüntüyü ikili görüntüye çevirmek için kullanılan bir yöntemdir. İkili görüntü (binary), görüntünün siyah ve beyaz olarak tanımlanmasıdır. Morfolojik operatörler gibi görüntü üzerindeki gürültüleri azaltmak veya nesne belirlemek gibi farklı amaçlar için kullanılır. Giriş olarak verilen görüntü üzerinde uygulanan thresholding tipine bağlı olarak, pikselleri verilen eşik değerine göre siyah ya da beyaz olarak günceller.

OpenCV içerisindeki sık kullanılan threshold tipleri:

- THRESH_BINARY
- THRESH_BINARY_INV
- THRESH_TRUNC
- THRESH_TOZERO
- THRESH_TOZERO_INV

Thresholding işlemi için Imgproc içerisindeki threshold()metodunu kullanacağız. Bu metot beş adet parametre almaktadır. Kaynak mat nesnesi yani giriş görüntüsü, hedef olarak ikinci bir mat nesnesi bu hedef nesne işlem sonucunu tutmak için, thresh olarak adlandırılan parametre eşik değeri, THRESH_BINARY ve THRESH_BINARY_INV gibi tipler için kullanılmak üzere maksimum değer ve yukarıda belirtilenler gibi threshold tipini parametre olarak almaktadır.

```
Imgproc.threshold(kaynakMat,hedefMat,esikDegeri,maksDeger,threshoidngTipi);
```

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.

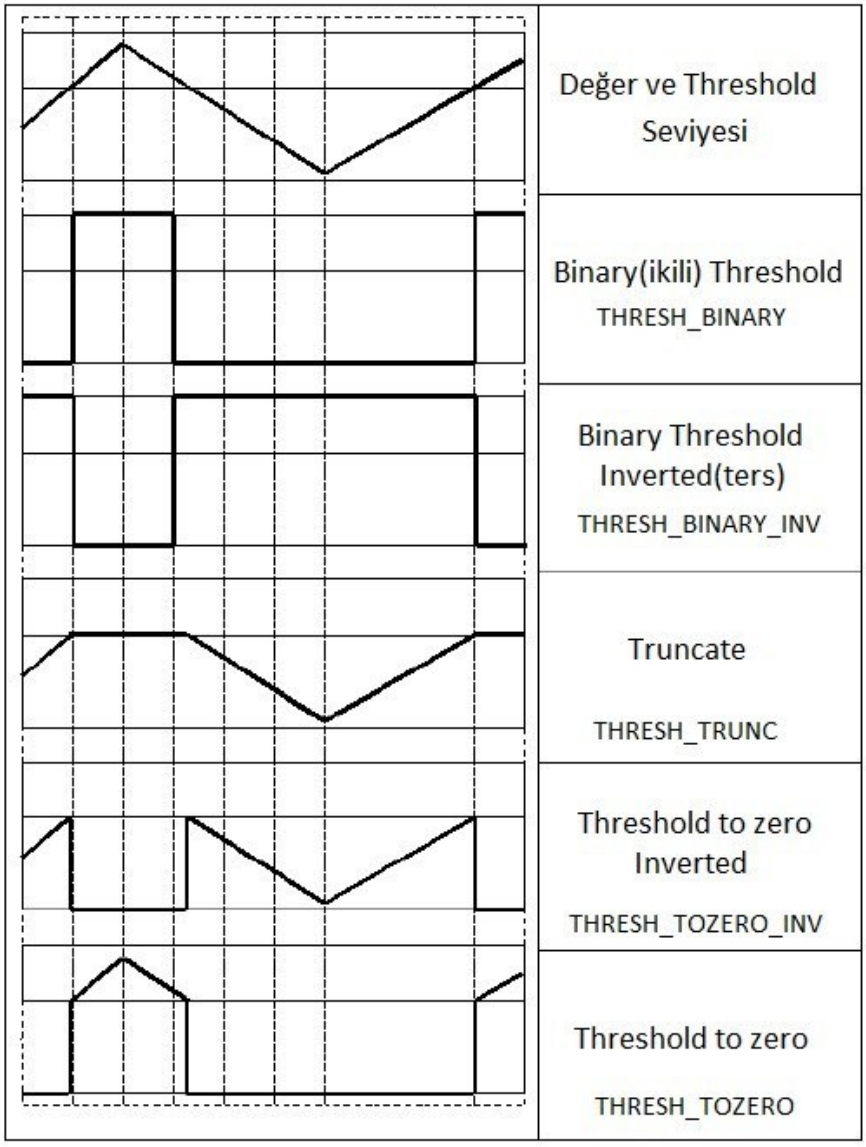
THRESH_TOZERO

Kaynak olarak alınan görüntü üzerindeki piksel, sınır olarak verilen değerden büyük olması durumunda piksel değeri korunacak, küçük olması durumunda ise piksel siyah olarak atanacaktır.

THRESH_TOZERO_INV

Kaynak olarak alınan görüntü üzerindeki piksel, sınır olarak verilen değerden küçük olması durumunda piksel değeri korunacak, büyük olması durumunda ise piksel siyah olarak atanacaktır.

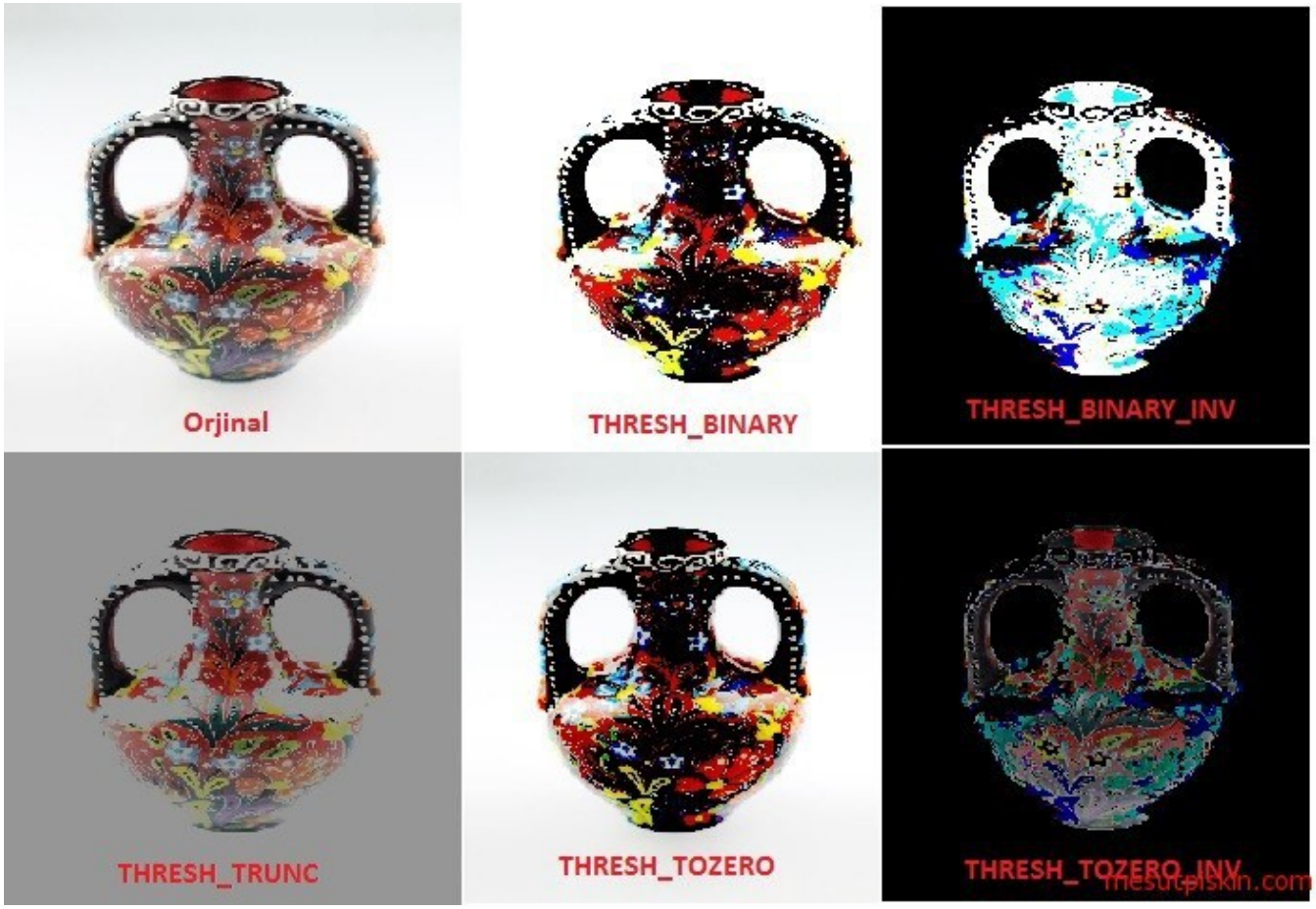
Aşağıdaki görselde kaynak üzerine etki eden threshold tipleri grafiksel olarak ifade edilmiştir.



Kaynak:http://docs.opencv.org

```
System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
Mat kaynakGoruntu=new Mat();
kaynakGoruntu=Imgcodecs.imread("C:\\1.jpg");
Mat hedefGoruntu=new Mat();
intthresh=150;
intmaxDeger=255;
Imgproc.threshold(kaynakGoruntu, hedefGoruntu, thresh, maxDeger,
Imgproc.THRESH_BINARY);
Imgcodecs.imwrite("C:\\2.jpg", hedefGoruntu);
System.out.println("Thresholding uygulandı.");
```

Yukarıdaki örnekte okunan görüntü üzerine threshold binary uygulanmıştır. Imgproc. THRESH_BINARY parametresini uygulamak istediğiniz threshold tipine göre değiştirebilirsiniz. Threshold metoduna parametre olarak diğer threshold tiplerini verelim ve kaynak görüntü üzerindeki farklılıklara bakalım.



FİLTRELER

Filtreler genellikle morfolojik işlemlerden veya thresholding işlemlerinden önce yapılır. Mobil fotoğraf düzenleme uygulamaları ve profesyonel görüntü düzenleme programlarında filtreler oldukça sık kullanılır.

Bundan önceki ilk iki konuda ele alınan örneklere dikkat ederseniz çıktı olarak oluşturulan görsellerde bazı piksellerin kaydığını, silik çıktığını veya tam olarak temizlenemediğini görürsünüz. Farklı görseller ile bu örnekleri yaptıysanız benzer sonuçlarla karşılaşmışsınızdır. Bunun nedeni kaynak olarak alınan görüntünün gürültülü olması veya ışıl dengesinin bozuk olması gibi birçok durumdur. Bu sorunları aşmak için kaynak görüntüye öncelikle bir filtre uygulanır ve görüntünün işleme için en verimli hale getirilmesi sağlanır ve bu durum ön işleme olarak adlandırılır.

Filtreler org.opencv.imgproc paketi içerisinde yer almaktadırlar.

Blur

Blur filtresi görüntüyü bulanıklaştırmak için kullanılır. Uygulamak için ise blur() metodu kullanılır. Bu metot parametre olarak kaynak görüntü mat nesnesi tipinde, mat tipinde bir sonuç ve Size tipinde uygulanacak olan bulanıklık değerini almaktadır.(çekirdek boyutu olarak da adlandırılır).

```
Imgproc.blur(kaynakGoruntu, hedefGoruntu, new Size(50,50));
```

GaussianBlur

GaussianBlur filtresi görüntü üzerinde düzleştirme işlemi uygular. Uygulamak için GaussianBlur() metodu kullanılır. Bu metot parametre olarak kaynak görüntü mat nesnesi tipinde, mat tipinde bir sonuç ve Size tipinde uygulanacak olan bulanıklık değerini (çekirdek boyutu olarak da adlandırılır) ve SigmaX olarak adlandırılan çekirdek standart sapmasıdır almaktadır.

```
Imgproc.GaussianBlur(kaynakGoruntu, hedefGoruntu, new Size(100,100),0);
```

Laplace

Görüntü üzerinde nesnelerin sınır çizgilerini belirlemek için kullanılır. Piksellerin renk farklılıklarından yararlanır ve bu sayede nesnelerin sınır çizgileri tespit edilmiş olur. Uygulamak için Laplacian() metodu kullanılır. Bu metot parametre olarak kaynak görüntü mat nesnesi tipinde, mat tipinde bir sonuç ve int tipinde derinlik değeri almaktadır.

```
Imgproc.Laplacian(kaynakGoruntu, hedefGoruntu,20);
```

Sobel

Görüntü üzerindeki kenarları elde etmek için kullanılır. Görüntü üzerindeki nesneleri kenarları belirleyerek ayrıştırmak istendiğinde bu filtreden yararlanılır. Uygulamak için Sobel() metodu kullanılır. Bu metot parametre olarak kaynak görüntü mat nesnesi tipinde, mat tipinde bir sonuç, int olarak çıkış görüntü nesnesi için derinlik ve int tipinde türev olarak adlandırılan x, y değeri.

```
Imgproc.Sobel(girisGoruntu, cikisGoruntu, ddepth, dx, dy);
```

Diğer OpenCV içerisinde bulunan filtreleri ise aşağıda yer almaktadır.

- pyrUp()
- pyrDown()
- pyrMeanShiftFiltering()
- boxFilter()
- filter2D()
- Scharr()
- sepFilter2D()
- buildPyramid()

ARKA PLAN TEMİZLEME

OpenCV ile arka plan temizleme işlemini absdiff() metoduile yapılmaktadır. Absdiff metodu parametre olarak verilen iki mat nesnesi yani matris arasında çıkarma işlemi yapar bu çıkarma işlemi sonucunda değişen kısımlar (hareketli kısımlar) sonuç olarak gösterilir ve çıkarma işlemi sonucu mutlak değer olarak döndürülür.

Arka plan temizleme, genellikle nesnelerin belirlenmesi, sayılması veya karşılaştırılması gibi işlemler için tercih edilir. Örneğin kapı girişlerine yerleştirilen bir kamera ile içeri giriş yapan kişi sayısı hesaplanabilir. Kamera yerleştirildikten sonra bir görüntü alınır ve arka plan olarak saklanır, daha sonraki her görüntü ile arka plan arasında bir çıkarma işlemi yapılır, çıkarma işlemi sonucunda oluşan görüntüye morfolojik operatörler ve thresholding uygulanarak fark sonucunda görüntü üzerindeki nesne belirlenir ve sayılır. Bu sayede giriş yapan kişi sayısı elde edilebilir. Bu algoritma kullanım alanı için basit bir örnek teşkil etmektedir.

Absdiff metodu Core içerisinde yer almaktadır.

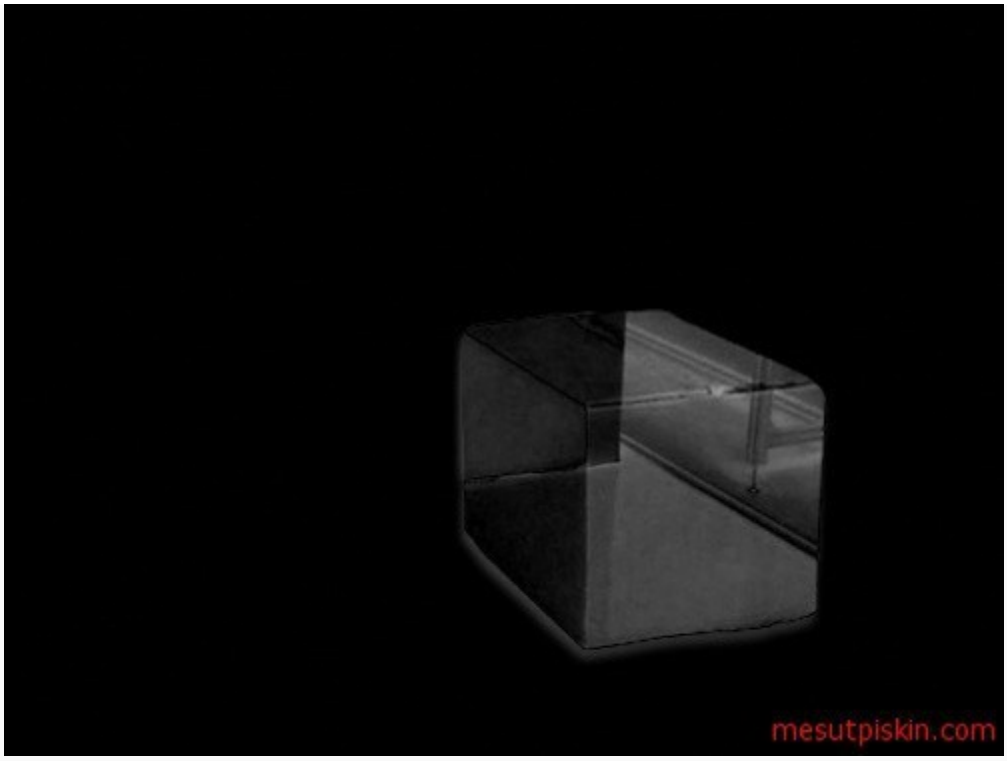
```
Core.absdiff(src1, src2, dst);
```

Parametre olarak iki adet kaynak görüntüleri barındıran mat nesneleri ve işlem sonucu için hedef mat nesnesi almaktadır. src1 nesnesi karşılaştırma için kullanılacak anlık görüntü src2 ise arka planı temsil etmektedir. dst nesnesi ise işlem sonucunu barındıracaktır. Aşağıdaki örnekte parametre olarak verilen 2 görüntüye çıkartma işlemi uygulanacak ve sonuç yeni bir görüntü olarak yazılacaktır.

```
importorg.opencv.core.Core;
importorg.opencv.core.Mat;
importorg.opencv.imgcodecs.Imgcodecs;
importorg.opencv.imgproc.Imgproc;
```

```
publicclass ArkaplanTemizleme {

    publicstaticvoid main(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        Mat kaynakMatris = Imgcodecs.imread("/home/mesutpiskin/Goruntu1.jpg");
        Mat kaynakMatrisGray = new Mat();
        // Rgb renk uzayından gri renk uzayına çevirme
        Imgproc.cvtColor(kaynakMatris, kaynakMatrisGray, Imgproc.COLOR_RGB2GRAY);

        Mat hedefMatris = Imgcodecs.imread("/home/mesutpiskin/ Goruntu2.jpg");
        Mat hedefMatrisGray = new Mat();
        // Rgb renk uzayından gri renk uzayına çevirme
        Imgproc.cvtColor(hedefMatris, hedefMatrisGray, Imgproc.COLOR_RGB2GRAY);
        // Arka plan temizlendikten sonraki veri
        Mat islemSonucu = new Mat();
        Core.absdiff(hedefMatrisGray, kaynakMatrisGray, islemSonucu);
        Imgcodecs.imwrite("/home/mesutpiskin/yeni.jpg", islemSonucu);
    }
}
```

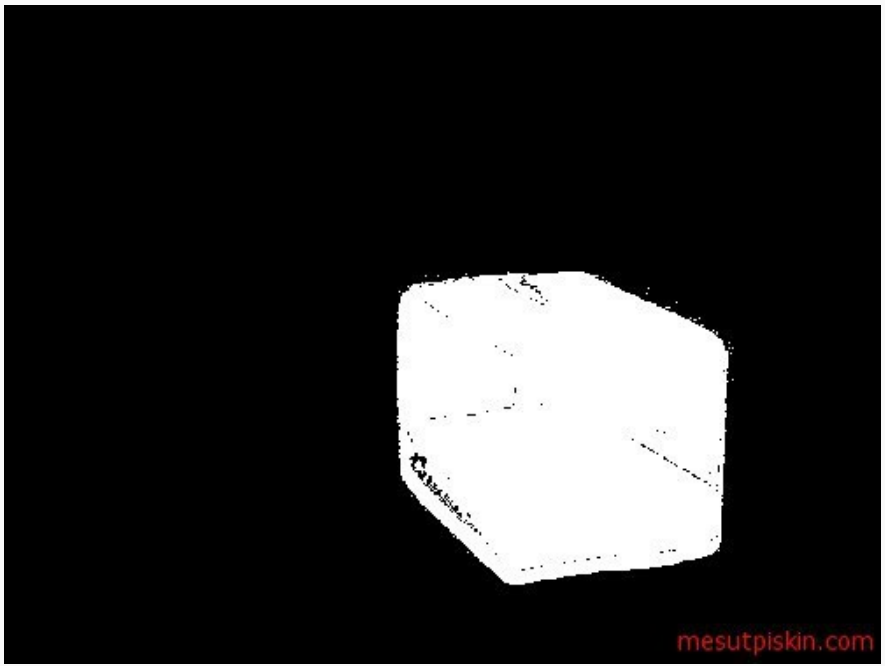
Soldaki görüntüyü arka plan olarak verdik ve çıkarma işlemi için ise sağdaki görüntüyü verdik, sonuç olarak ise ikinci resimde yer alan ve görüntüye sonradan dâhil olan arkadaki koli kaldı. Bu fonksiyon sayesinde görüntü üzerindeki değişiklikleri tespit ettik.



Çıktı olarak yakaladığımız görüntüyü daha net bir hale getirmek için threshold işlemi uygulayalım. Kenar çizgilerini daha net belirlemek ve görüntü üzerinde ki gürültüleri temizlemek için resim filtreleri uygulanabilir.

```
Mat thresholdCikti=new Mat();
Imgproc.threshold(yeni, thresholdCikti, 2, 255, Imgproc.THRESH_BINARY);
```

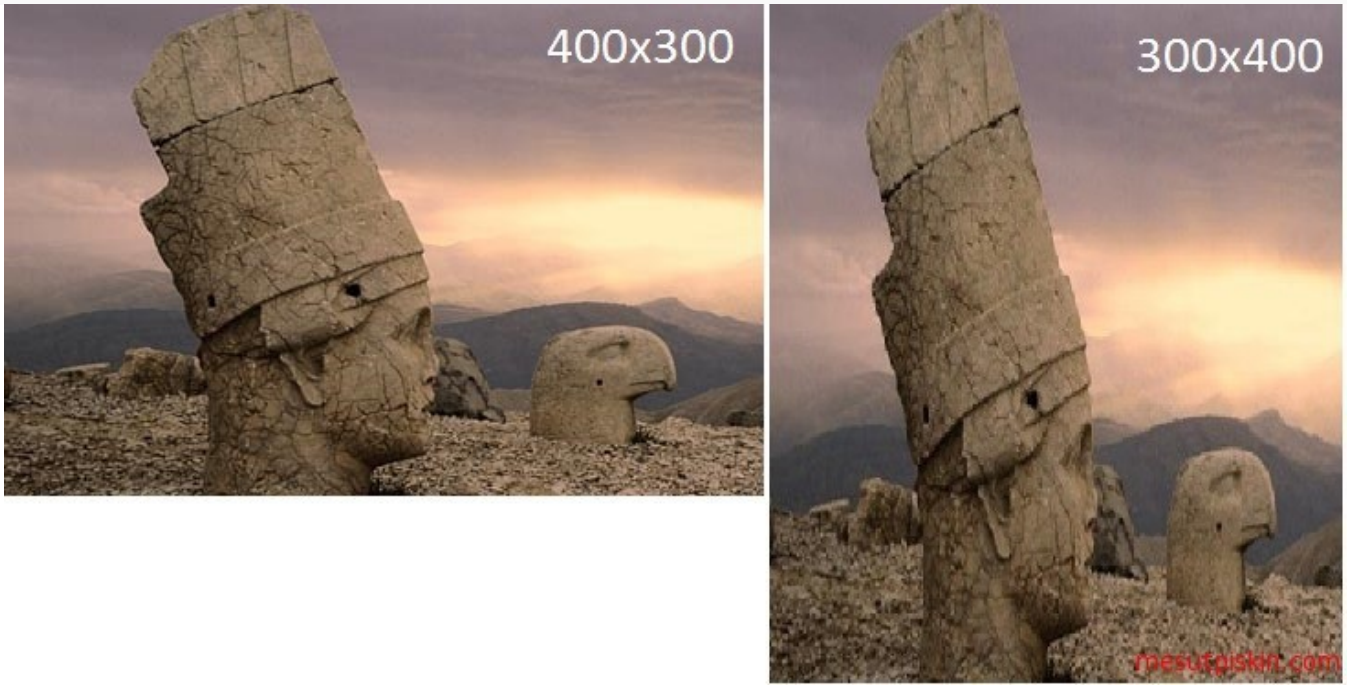
Yukarıdaki gibi thresh binary uyguladığımızda daha net bir sonuç elde ettik.



YENİDEN BOYUTLANDIRMA (RESİZE)

Bazı durumlarda okunan görüntünün tekrardan boyutlandırılması istenilebilir bunun için imgproc sınıfı içerisinde resize() metodu bulunmaktadır. Parametre olarak giriş mat nesnesi, çıkış mat nesnesi ve size olarak boyut almaktadır.

```
System.loadLibrary(Core.NATIVE_LIBRARY_NAME );
Mat kaynak = Imgcodecs.imread("C:\\1.jpg");
Mat hedef = new Mat();
Imgproc.resize(kaynak, hedef, new Size(300,400));
Imgcodecs.imwrite("C:\\2.jpg", hedef);
```



NESNE TESPİT VE TANIMA YÖNTEMLERİ

Nesne tanıma, görüntü işlemede büyük önem taşımaktadır. Bu ihtiyaç üzerine OpenCV de geliştirilmiş birçok yöntem bulunmaktadır. Bu yazımda nesne tanıma için kullanabileceğiniz yöntemleri aktaracağım.

Nesne tespiti için sık kullanılan dört yöntem mevcuttur. Bu yöntemler;

- Template Matching (Şablon Eşleştirme)
- HAAR Cascade
- LBP – Local Binary Pattern
- HOG – Histogram of Oriented Gradients

Şablon Eşleştirme (Template Matching) yöntemi dışındaki diğer yöntemler sınıflandırıcı olarak tanımlanan Machine Learning algoritmalarıdır. Bu algoritmalar öğretilmiş bir nesneyi tanımak için kullanılır, nesne tanıma için en çok kullanılan yöntemler bu algoritmalardır. Sınıflandırıcılar için nesneler Deep Learning yöntemleri ile öğretilebilir. Şablon eşleştirme yöntemi ise, aranan görüntü şablonunu kaynak görüntü üzerindeki tüm piksellerde dolaşarak eşleştirme işlemi yapar.

TEMPLATE MATCHİNG İLE NESNE TESPİTİ

Template Matching (Şablon Eşleştirme) yöntemi ile nesne tanıma daha çok kaynak bir görüntü üzerinde bir şablonu aramak için kullanılır. Nesneleri ayırt etmede çok fazla başarılı değildir. Örneğin, bir meyve sepeti bulunan görüntü üzerinde elmayı aramak için kullanılabilir. Aranan kaynak üzerinde verdiğiniz şablon birebir olarak aranır, başarılı bir sonuç için aradığınız elma görüntüsünün, meyve sepeti görselinden kırpılmış olması gerekebilir. Kırmızı bir elmayı şablon olarak tanımladınız ve meyve sepetinde aradınız, eğer meyve sepetinizde yarısı kesilmiş yarım bir elma var ise başarılı sonuç alamayacaksınız çünkü şablonunuzda ki ile kaynak görsel üzerinde yer alan elma aynı ölçülerde değildir.

Template Matching yöntemi ile kaynak görsel üzerinde aranan şablon Sliding window (Kayan,sürgülü pencere) yöntemi ile aranır. Kaynak üzerinde şablon (1,1) koordinatlarına oturtulur ve tüm pikseller üzerinde dönülür, kullandığınız benzerlik yöntemine göre bir benzerlik oranı oluşturulur ve şablonunuz ile o anki dönülen şablon benzer ise sonuç olarak size o pikselleri döndürür.

Yukarıdaki görselde, solda yer alan hayvanlar kaynak görüntü olarak alınmış buradaki bir köpeğin yüzü kırpılarak alınmış ve şablon olarak kullanılmış işlem sonucunda ise verilen şablon ile aynı ölçüde bir sonuç çıkmıştır. Kaynak değiştirilmiş olsaydı ve aynı köpek farklı ışık açısı, farklı poz veya farklı bir zemin üzerinde olsaydı sonuç yukarıdaki kadar başarılı olmayacaktı.

Kaynak üzerinde şablon aranırken, yukarıda olduğu gibi 0,0 koordinatlarına istenilen şablon oturtulacak soldan sağa ve yukarıdan aşağıya doğru tüm matris elemanları yani pikseller üzerinde dönülmektedir.

Template Matching yönteminde kaynak ile şablonu eşleştirirken kullanılan farklı yöntemler vardır. Bu yöntemler aşağıdaki gibidir.

- TM_CCOEFF
- TM_CCOEFF_NORMED
- TM_CCORR
- TM_CCORR_NORMED
- TM_SQDIFF
- TM_SQDIFF_NORMED

Bu yöntemlerin bir birleri arasındaki farkları örnek görseller ile anlayabilir. Bu yöntemlerin her birinin farklı bir matematiksel formül olduğunu unutmayalım. Bu formüller aşağıdaki gibidir.

1. method=CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

2. method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

3. method=CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

4. method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

5. method=CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

6. method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

Bir örnek yapalım ve sonuçlarını gözlemleyelim.

```
package com.mesutpiskin.templatematcing;
import org.opencv.core.Core;
import org.opencv.core.Core.MinMaxLocResult;
import org.opencv.core.Mat;
import org.opencv.core.Point;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

public class TemplateMatching {

    public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        Mat source=null;
        Mat template=null;
        String filePath="C:\\Users\\mesutpiskin\\Desktop\\Object
Detection\\Template Matching\\Sample Image\\";
        source=Imgcodecs.imread(filePath+"kapadokya.jpg");
        template=Imgcodecs.imread(filePath+"balon.jpg");

        Mat outputImage=new Mat();
        int machMethod=Imgproc.TM_CCOEFF;

        Imgproc.matchTemplate(source, template, outputImage, machMethod);
        MinMaxLocResult mmr = Core.minMaxLoc(outputImage);
        Point matchLoc=mmr.maxLoc;

        Imgproc.rectangle(source, matchLoc, new Point(matchLoc.x + template.cols(),
                matchLoc.y + template.rows()), new Scalar(255, 255, 255));

        Imgcodecs.imwrite(filePath+"sonuc.jpg", source);
        System.out.println("İşlem tamamlandı.");
    }

}
```

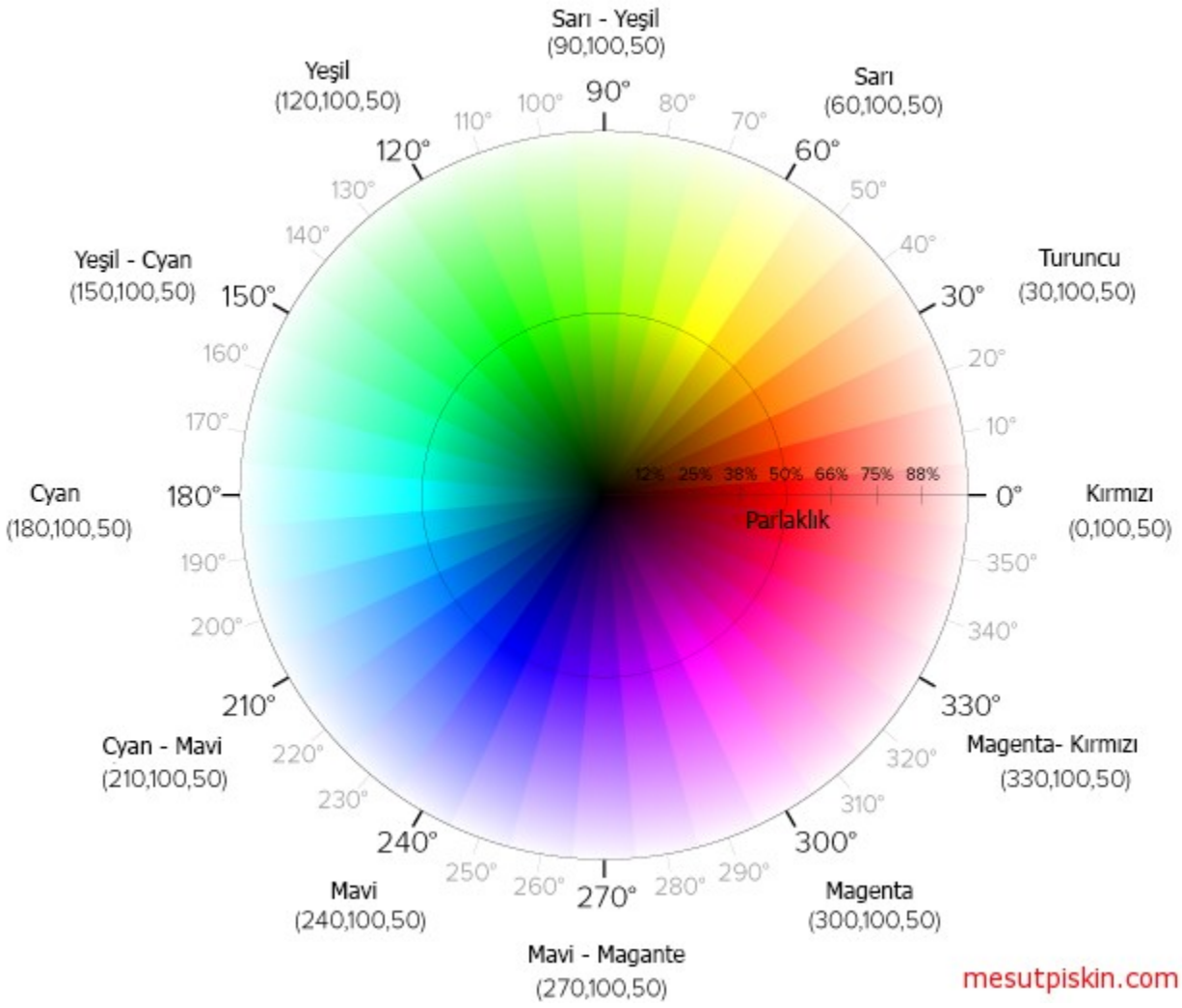
matchTemplate metodu parametre olarak mat tipinde kaynak görsel, şablon görsel ve çıktı için kullanacağı mat nesnesini, int tipinde ise eşleştirme yöntemini almaktadır. Örnekte bir kaynak görsel yükledik bu görsel üzerinden kırpılmış bir görsel şablon olarak eklendi. Sonuç için bir mat nesnesi tanımlandı. Bu çıktı matrisi şablonun ölçüleri kullanılarak boyutlandırıldı. Kaynak görsel üzerinde sonuç nesnesi boyutları kullanılarak bir kare çizildi, kare için bir scalar yani renk tanımlandı (255,255,255 rgb renk kodları) ve sonuç aynı dizine yazıldı.

Sonuç;

RENK TESPİTİ OBJE TAKİBİ

Bu projede kamera aygıtından alınan görüntü üzerinde belirlediğimiz bazı renkleri tespit edecek ve bu renge sahip objeleri işaretleyeceğiz. Öncelikle bir renk aralığı belirleyeceğiz daha sonra kamera aygıtından yakalanan RGB renk uzayına sahip görüntüyü HSV renk uzayına çevireceğiz. Görüntü üzerinde eşikleme, aşındırma ve genişletme operatörlerini uygulayacağız. Kenar bulma gibi metotları kullanarak renklerin ayrımını tespit edip nesneleri işaretleyeceğiz.

Renk tespiti için HSV uzayını kullanacağız, RGB renk uzayında yapılacak olan threshold HSV renk uzayında yapılacak threshold a göre yetersizdir. HSV de H (HUE) değeri daha ayırt edilebilir şekilde değiştiği için farklı renkli objelerin resimde tespiti çok daha kolay olmaktadır. HSV uzaydaki renk kodları aralığını aşağıdaki görselden tespit edebilirsiniz. Proje 2 adet sınıftan oluşacak, birincisi Frame'ler ve görüntülenecek olan kamera görüntüsünü yönetecek sınıf, diğer ise renk tespitini ve işaretleme gibi işlemleri yerine getirecek sınıf.




```
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import javax.swing.JPanel;
import org.opencv.core.Mat;

//Kameradan alınan ve işlenen görüntüler jpanel üzerinde görüntülenecek
class Panel extends JPanel {
    private static final long serialVersionUID = 1L;
    private BufferedImage image;

    public Panel() {
        super();
    }
    private BufferedImage getimage() {
        return image;
```

```
    }
    public void setimage(BufferedImage newimage) {
        image = newimage;
        return;
    }

    public void setimagewithMat(Mat newimage) {
        image = this.ConvertImage(newimage);
        return;
    }

    /* Mat nesnesini frame içerisinde göstermek için BufferedImage tipine
    çeviriyoruz*/
    public BufferedImage ConvertImage(Mat matrix) {
        // Mat nesnesinin sütun, satır ve boyutunu BufferedImage nesnesi
içintutuyoruz
        int cols = matrix.cols();
        int rows = matrix.rows();
        int elemSize = (int) matrix.elemSize();
        byte[] data = new byte[cols * rows * elemSize];
        int type;
        matrix.get(0, 0, data);
        // Mat nesnesinin kaç kanallı, hangi renk uzayında olduğunu tespit
ediyoruz.
        switch (matrix.channels()) {
        // Tek kanallı gri renk uzayına sahip matris
        case 1:
            type = BufferedImage.TYPE_BYTE_GRAY;
            break;
        // Üç kanallı BGR renk uzayına sahip matris
        case 3:
            type = BufferedImage.TYPE_3BYTE_BGR;
    /*
     * Opencv rgb renk uzayını bgr olarak tuttuğu için görüntülemede
     * düzgün bir görüntü elde etmek amacıyla rgb uzayına çeviriyoruz
     */
            byte b;
            for (int i = 0; i<data.length; i = i + 3) {
```

```
                b = data[i];
                data[i] = data[i + 2];
                data[i + 2] = b;
            }
            break;
        default:
            return null;
        }
        BufferedImage image2 = new BufferedImage(cols, rows, type);
        image2.getRaster().setDataElements(0, 0, cols, rows, data);
        return image2;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        BufferedImage temp = getimage();

        if (temp != null)
            g.drawImage(temp, 10, 10, temp.getWidth(), temp.getHeight(), this);
    }
}
```

Renk tespitini yapacak olan ikinci sınıf.

```
import java.util.ArrayList;
import java.util.List;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;
```

```
public class Detector {

    public static void main(String arg[]) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        // Anlıkolarakyakalanankameragörüntülerinigöstereceğimiz frame ve panel
        JFrame cameraFrame = new JFrame("Anlık kamera görüntüsü");
        cameraFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        cameraFrame.setSize(640, 480);
        cameraFrame.setBounds(0, 0, cameraFrame.getWidth(),
cameraFrame.getHeight());
        Panel panelCamera = new Panel();
        cameraFrame.setContentPane(panelCamera);
        cameraFrame.setVisible(true);

        // İşlenecekgörüntünün threshold uygulandıktan sonraki halini
        // göstereceğimiz frame ve panel
        JFrame thresholdFrame = new JFrame("Threshold görüntü");
        thresholdFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        thresholdFrame.setSize(640, 480);
        thresholdFrame.setBounds(0, 0, cameraFrame.getWidth(),
cameraFrame.getHeight());
        Panel panelThreshold = new Panel();
        thresholdFrame.setContentPane(panelThreshold);
        thresholdFrame.setVisible(true);

        // Video akışıiçin, 0 ilevarsayılankamerayıbaşlatacağız
        VideoCapture capture = new VideoCapture(0);
        // Parametreleri atıyoruz
        /*capture.set(3, 1366);
        capture.set(4, 768);
        capture.set(15, -2);*/
        // Saf kameragörünrüsü
        Mat webcam_image = new Mat();
        // Hsv renk uzayında görüntüsü
        Mat hsv_image = new Mat();
        // 1. ve 2. threshold
```

```
        Mat thresholded = new Mat();
        Mat thresholded2 = new Mat();
        // Kameradan görüntü oku
        capture.read(webcam_image);
        // Kameradan alınan görüntüleri gösterecek olduğumuz frame boyutları
        // kameradan okunang örüntüye göre ayarlanıyor.

        cameraFrame.setSize(webcam_image.width() + 50, webcam_image.height() + 50);
        thresholdFrame.setSize(webcam_image.width() + 50, webcam_image.height() +
50);

        Mat array255 = new Mat(webcam_image.height(), webcam_image.width(),
CvType.CV_8UC1);
        array255.setTo(new Scalar(255));
        Mat distance = new Mat(webcam_image.height(), webcam_image.width(),
CvType.CV_8UC1);

        List<Mat> lhsv = new ArrayList<Mat>(3);
        Mat circles = new Mat();
        // Renktespitiburadabelirttiğimizminve max değerleregöreyapılacak
        // hsvuzayındaverdiğimizrenktonlarıarasındaki her renktespitedilecektir.
        // Renkaralıklarıiçinhsvrenktablolarınagözatabilirsiniz.

        Scalar minColor = new Scalar(5, 100, 100, 0);
        Scalar maxColor = new Scalar(10, 255, 255, 0);
        // Kamera aygıtı çalışıyor ise
        if (capture.isOpened()) {
            while (true) {
                capture.read(webcam_image);
                // Bir görüntü okunmuş ve boşdeğilse
                if (!webcam_image.empty()) {
                    // Kameradan okunan görüntü hsv renk uzayına dönüştürülür
                    Imgproc.cvtColor(webcam_image, hsv_image,
Imgproc.COLOR_BGR2HSV);
                    Core.inRange(hsv_image, minColor, maxColor, thresholded);

                    // Erode ve dilate işlemi uygulanır yapısal element ölçüleri
                    // iki işlemdede aynıdır
```

```
                    Imgproc.erode(thresholded, thresholded,
                            Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new
Size(8, 8)));
                    Imgproc.dilate(thresholded, thresholded,
                            Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new
Size(8, 8)));
                    /*
                     * Split metodu
                     * ilegörüntüüzerindekihistogramlarıparçalıyoruz. Matrisin
                     * her boyutu ayrı ayrı nesnelere atanıyor.
                     */
                    Core.split(hsv_image, lhsv);
                    Mat S = lhsv.get(1);
                    Mat V = lhsv.get(2);
                    // dizilerarası element farklarıhesaplıyoruz
                    Core.subtract(array255, S, S);
                    Core.subtract(array255, V, V);
                    S.convertTo(S, CvType.CV_32F);
                    V.convertTo(V, CvType.CV_32F);
                    // 2 boyutluvektörlerimizinbüyüklüğünühesaplıyoruz
                    Core.magnitude(S, V, distance);
                    /*
                     * Verilendeğerlerarasında thresholding uyguluyor.
                     * pikselindeğeriverilendeğerlerarasındaise o
                     * piksel,beyazdeğilsesiyahyapılıyor.
                     */
                    Core.inRange(distance, new Scalar(0.0), new Scalar(200.0),
thresholded2);
                    Core.bitwise_and(thresholded, thresholded2, thresholded);
                    // thresholded içingaussian blur filtresiuyguluyoruz
                    Imgproc.GaussianBlur(thresholded, thresholded, new Size(9, 9),
0, 0);
                    // Şeklindışhatları
                    List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
                    Imgproc.HoughCircles(thresholded, circles,
Imgproc.CV_HOUGH_GRADIENT, 2, thresholded.height() / 8,
                            200, 100, 0, 0);
                    // thresholding sonrasınesnenin binary
```

```
                    // haliüzerindebağlınoktalarıtesiğediyoruz
                    Imgproc.findContours(thresholded, contours, thresholded2,
Imgproc.RETR_LIST,
                            Imgproc.CHAIN_APPROX_SIMPLE);
                    // Nesneninkonumunaaşağıdakirenkileçiziyoruz
                    Imgproc.drawContours(webcam_image, contours, -2, new Scalar(10,
0, 0), 4);


                    /*
                     * Panellerüzerinegörünrüleriatayıp frame'lerin
                     * tekrardançizilmesinisağlıyoruz
                     */
                    panelCamera.setimagewithMat(webcam_image);
                    panelThreshold.setimagewithMat(thresholded);
                    cameraFrame.repaint();
                    thresholdFrame.repaint();

                } else {
                    JOptionPane.showMessageDialog(null, "Kamera aygıtına
bağlanılamadı!");
                    break;
                }
            }
        }

    }

}
```

RENGİNİ KULLANARAK NESNENİN TESPİTİ

Opencv ile kamerada yakalanan objelerin tespit edilmesinin farklı yolları vardır, bu yazıda nesnelerin renklerine göre nasıl yakalanabileceğine bakacağız. Tek bir sınıfımız olacak bu sınıf ile ilk olarak sistemdeki kameraya bağlanıp ardından yakalanan görüntüde istediğimiz renge sahip nesneyi yakalayarak bu nesnenin etrafını çizeceğiz. Bu işlemlerin hepsi için pratik opencv sınıf ve metotları mevcuttur. Renkleri karşılaştırırken R G B kodlarına göre karşılaştırma yapabilirsiniz. Projede renklerin kameradaki farklı parametrelere göre farklı görünebileceğini düşünerek minumum ve maksimum diye nitelendirdiğimiz o rengin açığı ve koyusunu belirterek istediğimiz rengi yakalayacağız yakalanan fotoğrafa thresholding uygulayarak gereksiz yerleride atacağız.

Proje örnek olarak kullanacağım renk sarı olacak. Aynı anda bir den fazla objeyi yakalamakta mümkün bunun için gerekli açıklamaları proje kodları içerisinde bulabilirsiniz.

Kullandığım opencv versiyonu 2.4.11 şu anda 3 sürümü mevcut fakat birçok sınıf ve metot değiştiği için 3x sürümlerinde sıkıntı çıkabileceğini unutmayın.Kullanmak istiyorsanız da opencv.org üzerindeki 3 dokümanına göz atabilirsiniz,uyarlamanız için size yardımcı bilgiler içermektedir.

```
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JFrame;
import javax.swing.JPanel;
//OPENCV 2.4.11
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.Point;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.highgui.*;
import org.opencv.imgproc.*;
import org.opencv.core.CvType;

class Panel extends JPanel {

    private static final long serialVersionUID = 1L;
    private BufferedImage image;


    public Panel() {
        super();
    }

    private BufferedImage getimage() {
        return image;
    }

    public void setimage(BufferedImage newimage) {
```

```
        image = newimage;
        return;
    }

    public void setimagewithMat(Mat newimage) {
        image = this.matToBufferedImage(newimage);
        return;
    }

    public BufferedImage matToBufferedImage(Mat matrix) {
        int cols = matrix.cols();
        int rows = matrix.rows();
        int elemSize = (int) matrix.elemSize();
        byte[] data = new byte[cols * rows * elemSize];
        int type;
        matrix.get(0, 0, data);
        switch (matrix.channels()) {
        case 1:
            type = BufferedImage.TYPE_BYTE_GRAY;
            break;
        case 3:
            type = BufferedImage.TYPE_3BYTE_BGR;
            //renk dönüşümü
            byte b;
            for (int i = 0; i < data.length; i = i + 3) {
                b = data[i];
                data[i] = data[i + 2];
                data[i + 2] = b;
            }
            break;
        default:
            return null;
        }
        BufferedImage image2 = new BufferedImage(cols, rows, type);
        image2.getRaster().setDataElements(0, 0, cols, rows, data);
        return image2;
    }
```

```
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        BufferedImage temp = getimage();

        if (temp != null)
            g.drawImage(temp, 10, 10, temp.getWidth(), temp.getHeight(), this);

    }
}

public class ColorTracking {

    public static void main(String arg[]) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        try {
            //Yakalamak istediğim renklerin kodları min ve max olarak verilmiştir
            Scalar hsv_min = new Scalar(21, 100, 100);
            Scalar hsv_max = new  Scalar(31, 255, 255);
            //Varsayılan webcam 0 dır sistemdeki diğer kameralar 1,2,3,4 şeklinde
indexlenmektedirler
            trackColor(1, hsv_min, hsv_max);
        } catch (AWTException e) {

            e.printStackTrace();
        }
        return;
    }
    //Renge göre objeyi takip eder
    private static void trackColor(int webcam, Scalar colorMin, Scalar colorMax)
throws AWTException {

        //Kamera frame orjinal görüntüye sahiptir
        JFrame frmeaCamera = new JFrame("Kamera");
        frmeaCamera.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        frmeaCamera.setSize(640, 480);
        frmeaCamera.setBounds(0, 0, frmeaCamera.getWidth(),
frmeaCamera.getHeight());
        Panel panel1 = new Panel();
        frmeaCamera.setContentPane(panel1);
        frmeaCamera.setVisible(true);

        //HSV görüntüye sahiptir
        JFrame frameHsv = new JFrame("HSV");
        frameHsv.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frameHsv.setSize(640, 480);
        frameHsv.setBounds(300, 100, frameHsv.getWidth() + 300, 100 +
frameHsv.getHeight());
        Panel panel2 = new Panel();
        frameHsv.setContentPane(panel2);
        frameHsv.setVisible(true);

        //Yakalanan objelerin görüntüsüne sahiptir
        JFrame frameThreshold = new JFrame("Threshold");
        frameThreshold.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frameThreshold.setSize(640, 480);
        frameThreshold.setBounds(900, 300, frameHsv.getWidth() + 900, 300 +
frameHsv.getHeight());
        Panel panel4 = new Panel();
        frameThreshold.setContentPane(panel4);
        frameThreshold.setVisible(true);
        // Vivdeo akışı
        VideoCapture capture = new VideoCapture(webcam);
        //Video çözünürlülükleri
        capture.set(3, 1024);
        capture.set(4, 768);


        Mat webcam_image = new Mat();
        Mat hsv_image = new Mat();
        Mat thresholded = new Mat();
        Mat thresholded2 = new Mat();
        capture.read(webcam_image);
```

```
        frmeaCamera.setSize(webcam_image.width() + 40, webcam_image.height() + 60);
        frameHsv.setSize(webcam_image.width() + 40, webcam_image.height() + 60);

        frameThreshold.setSize(webcam_image.width() + 40, webcam_image.height() +
60);
        Mat array255 = new Mat(webcam_image.height(), webcam_image.width(),
CvType.CV_8UC1);
        array255.setTo(new Scalar(255));
        Mat distance = new Mat(webcam_image.height(), webcam_image.width(),
CvType.CV_8UC1);

        List<Mat> lhsv = new ArrayList<Mat>(3);
        Mat circles = new Mat();

        double[] data = new double[3];

        //Webcam açılmış ,çalışıyor ise
        if (capture.isOpened()) {
            while (true) {
                capture.read(webcam_image);
                if (!webcam_image.empty()) {

                    Imgproc.cvtColor(webcam_image, hsv_image,
Imgproc.COLOR_BGR2HSV/* COLOR_BGR2RGB */);

                    //Min ve max renk uzaylarına göre karşılaştırma
                    Core.inRange(hsv_image, colorMin, colorMax,thresholded);
                    //birden fazla renk yakalamak için
                    //Core.inRange(hsv_image, colorMin2, colorMax2,thresholded2);
                    Imgproc.erode(thresholded, thresholded,
                            Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new
Size(8, 8)));
                    Imgproc.dilate(thresholded, thresholded,
                            Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new
Size(8, 8)));

                    Core.split(hsv_image, lhsv);
                    Mat S = lhsv.get(1);
```

```
                    Mat V = lhsv.get(2);
                    Core.subtract(array255, S, S);
                    Core.subtract(array255, V, V);
                    S.convertTo(S, CvType.CV_32F);
                    V.convertTo(V, CvType.CV_32F);
                    Core.magnitude(S, V, distance);
                    Core.inRange(distance, new Scalar(0.0), new Scalar(200.0),
thresholded2);
                    Core.bitwise_and(thresholded, thresholded2, thresholded);

                    Imgproc.GaussianBlur(thresholded, thresholded, new Size(9, 9),
0, 0);
                    List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
                    Imgproc.HoughCircles(thresholded, circles,
Imgproc.CV_HOUGH_GRADIENT, 2, thresholded.height() / 8,
                            200, 100, 0, 0);

                    Imgproc.findContours(thresholded, contours, thresholded2,
Imgproc.RETR_LIST,
                            Imgproc.CHAIN_APPROX_SIMPLE);
                    //Yakalanan nesneyi nokta çizerek belirtir,scalar çizim rengi,
2  ise kalınlıktır
                    Imgproc.drawContours(webcam_image, contours, -1, new
Scalar(255, 0, 0), 5);

                    System.out.println(contours.size());



                            // -- Resimi panellere ekler

                    //Data dizisi renk kodlarını tutmaktadır
                    panel1.setimagewithMat(webcam_image);
                    panel2.setimagewithMat(hsv_image);
                    panel4.setimagewithMat(thresholded);
                    frmeaCamera.repaint();
                    frameHsv.repaint();
                    // frame3.repaint();
```

```
                    frameThreshold.repaint();

                } else {
                    System.out.println("GÖRÜNTÜ ALINAMADI");
                    break;
                }
            }
        }

    }
}
```

HAAR CASCADE CLASSİFİER YÜZ VE GÖZ TESPİTİ

Nesneyi tespit etmek için öncelikle nesneyi sisteme tanıtmamız ve daha sonra bu tanımlanmış veri setini kullanarak görüntü üzerinde arama yapmamız gerekir. Haar cascade sınıflandırıcı bizden xml dosyası alır bu xml dosyaları bir nesnenin binlerce negatif ve pozitifi ile hazırlanmış veri setidir. Pozitif olarak tanımlanan görüntüler istenilen nesnenin bulunduğu negatif olarak tanımlananlar ise bulunması istenilen nesnenin bulunmadığı görüntülerdir. Daha detaylı bilgi için Haar cascade algoritmasına göz atabilir ve Haar cascade sınıflandırıcı eğitimleri ile de nasıl nesne tanımlanır öğrenebilirsiniz.

OpenCV içerisinde birçok nesne hali hazırda öğretilmiş olarak gelir bunlardan yüz ve gözü kullanarak anlık olarak bir yüz ve göz tespiti yapacağız. Açıklamaları kaynak kod üzerinde yapacağım adım adım giderek kurguyu anlayabilirsiniz. Koda geçmeden önce aşağıdaki diğer yazılarımı okumanızı öneririm bu sayede çizim işlemlerini ve gui uygulama geliştirme hakkında bilgi edinebilirsiniz.

```
import java.awt.FlowLayout;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.InputStream;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
```

```
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;
import org.opencv.videoio.VideoCapture;
/*
 * OpenCV version 3.1
*/
public class DetectFace {

    static JFrame frame;
    static JLabel lbl;
    static ImageIcon icon;

    public static void main(String[] args) {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        /*Cascade Classifier için öðretilmiþ veri kümesi, opencv
build/etc/haarcascades/ içerisinde yer almaktadýr
          Daha fazla bilgi için Haar Cascade sýnýflandýrýcýlarýna bakabilirsiniz.
        */

        CascadeClassifier cascadeFaceClassifier = new CascadeClassifier(

"D:/Programlar/Opencv/3.1.0/opencv/build/etc/haarcascades/haarcascade_frontalface_d
efault.xml");
        CascadeClassifier cascadeEyeClassifier = new CascadeClassifier(

"D:/Programlar/Opencv/3.1.0/opencv/build/etc/haarcascades/haarcascade_eye.xml");

        CascadeClassifier cascadeNoseClassifier = new CascadeClassifier(

"D:/Programlar/Opencv/3.1.0/opencv/build/etc/haarcascades/haarcascade_mcs_nose.xml"
);
        //CascadeClassifier cascadeMouthClassifier = new
CascadeClassifier("OpenCV/haarcascades/haarcascade_mcs_mouth.xml");
haarcascade_mcs_mouth on 2.4.11
        //Varsayýlan kamera aygýtýný baþlat
```

```
        VideoCapture videoDevice = new VideoCapture();
        videoDevice.open(0);
        if (videoDevice.isOpened()) {
        //Sonsuz bir döngü ile sürekli olarak görüntü akýþý saðlanýr
            while (true) {
                Mat frameCapture = new Mat();
                videoDevice.read(frameCapture);

                //Yakalanan görüntüyü önce dönüþtür ve frame içerisine yükle
                MatOfRect faces = new MatOfRect();
                cascadeFaceClassifier.detectMultiScale(frameCapture, faces);
                //Yakalanan çerçeve varsa içerisinde dön ve yüzün boyutlarý
ölçüsünde bir kare çiz
                for (Rect rect : faces.toArray()) {
                    //Sol üst köþesine metin yaz
                    Imgproc.putText(frameCapture, "Face", new Point(rect.x,rect.y-
5), 1, 2, new Scalar(0,0,255));
                    Imgproc.rectangle(frameCapture, new Point(rect.x, rect.y), new
Point(rect.x + rect.width, rect.y + rect.height),
                            new Scalar(0, 100, 0),3);
                }

                //Gözleri bul ve bulunan array içerisinde dönerek kare çiz
                MatOfRect eyes = new MatOfRect();
                cascadeEyeClassifier.detectMultiScale(frameCapture, eyes);
                for (Rect rect : eyes.toArray()) {
                    //Sol üst köþesine metin yaz
                    Imgproc.putText(frameCapture, "Eye", new Point(rect.x,rect.y-
5), 1, 2, new Scalar(0,0,255));
                    //Kare çiz
                    Imgproc.rectangle(frameCapture, new Point(rect.x, rect.y), new
Point(rect.x + rect.width, rect.y + rect.height),
                            new Scalar(200, 200, 100),2);
                }

                //Burunlarý bul ve bulunan array içerisinde dönerek kare çiz
                MatOfRect nose = new MatOfRect();
                cascadeNoseClassifier.detectMultiScale(frameCapture, nose);
```

```
                for (Rect rect : nose.toArray()) {
                    //Sol üst köþesine metin yaz
                    Imgproc.putText(frameCapture, "Nose", new Point(rect.x,rect.y-
5), 1, 2, new Scalar(0,0,255));
                    //Kare çiz
                    Imgproc.rectangle(frameCapture, new Point(rect.x, rect.y), new
Point(rect.x + rect.width, rect.y + rect.height),
                            new Scalar(50, 255, 50),2);
                }

                //Aðýz bul ve bulunan array içerisinde dönerek kare çiz
               /*MatOfRect mouth = new MatOfRect();
                cascadeMouthClassifier.detectMultiScale(frameCapture, mouth);
                for (Rect rect : mouth.toArray()) {

                    Imgproc.rectangle(frameCapture, new Point(rect.x, rect.y), new
Point(rect.x + rect.width, rect.y + rect.height),
                            new Scalar(129, 90, 50),2);
                }

                */
                //Resmi swing nesnesinde gösterebilmek için önce image haline çevir
ve ekrana bas
                PushImage(ConvertMat2Image(frameCapture));
                System.out.println(String.format("%s yüz(FACES) %s göz(EYE) %s
burun(NOSE) detected.",
faces.toArray().length,eyes.toArray().length,nose.toArray().length));
            }
        } else {
            System.out.println("Video aygýtýna baðlanýlamadý.");
            return;
        }
    }
    //Mat nesnesini image tipine dönüþtür
    private static BufferedImage ConvertMat2Image(Mat kameraVerisi) {


        MatOfByte byteMatVerisi = new MatOfByte();
```

```
        //Ara belleðe verilen formatta görüntü kodlar
        Imgcodecs.imencode(".jpg", kameraVerisi, byteMatVerisi);
        //Mat nesnesinin toArray() metodu elemanlarý byte dizisine çevirir
        byte[] byteArray = byteMatVerisi.toArray();
        BufferedImage goruntu = null;
        try {
            InputStream in = new ByteArrayInputStream(byteArray);
            goruntu = ImageIO.read(in);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        return goruntu;
    }
//Bir frame (çerçeve) oluþturur
    public static void PencereHazirla() {
        frame = new JFrame();
        frame.setLayout(new FlowLayout());
        frame.setSize(700, 600);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    //Resmi gösterecek label oluþturur
    public static void PushImage(Image img2) {
        //Pencere oluþturulmamýþ ise hazýrlanýr
        if (frame == null)
            PencereHazirla();
        //Daha önceden bir görüntü yüklenmiþ ise yenisi için kaldýrýr
        if (lbl != null)
            frame.remove(lbl);
        icon = new ImageIcon(img2);
        lbl = new JLabel();
        lbl.setIcon(icon);
        frame.add(lbl);
        //Frame nesnesini yeniler
        frame.revalidate();
    }
}
```

HAAR CASCADE İLE YÜZ TESPİTİ (PYTHON)

```
import cv2

capture = cv2.VideoCapture(0)
cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

while True:

    ret, frame = capture.read()
    faces = cascade.detectMultiScale(frame, 1.5, 3)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,255),2)



    cv2.imshow('Kamera',frame)
    if cv2.waitKey(30) & 0xff ==27:
        break

capture.release()
cv2.destroyAllWindows()
```

YÜZ TANIMAYA GİRİŞ

Yüz tanımaya girmeden önce değinmemiz gereken bir konu var. Görüntü işlemede yüz tespiti için birçok yöntem mevcut bu yöntemlere daha önce gerek video eğitimlerimde gerekse yazılarımda değinmiştim. Yüz tanıma içinde farklı yöntemler mevcut, burada dikkat edilmesi gereken konu yüz tanıma işlemi ile yüz tespit işleminin farklı olmalı. Bazı yöntemler ile görüntülerdeki insan yüzlerini diğer nesnelerden ayırt ederek tespit edebiliriz çünkü insan yüzü geometrik olarak çok fazla farklılık göstermez bu farklılığın az oluşuda yüzü tespit etmeyi kolaylaştırır. Yüz tanıma ise daha önceden tespit edilen bir kaynak yüzün

yeni alınacak yüz ile karşılaştırılıp aradaki benzerliği tespit edebilmekir. Bu bağlamda yüz tanıma tespit etme işlemine göre daha zordur. Ortamdaki ışık veya yüzde meydana gelecek küçük değişiklikler algoritmanızın hatalı sonuç vermesine yol açabilir. Bu durumlardan dolayı tespit ile tanıma işlemini bir birinden iyi ayırt etmek gerekir.

Yüz tanıma insanlar için oldukça kolay bir iştir. Bazı deneyler göstermiştir ki üç günlük bir bebek bile gördüğü yüzü daha sonra ayırt edebilmektedir. Peki, bilgisayarlar için bu durum ne kadar zor olabilir? Bizler bu güne kadar yüz tanıma konusunda çok az şey biliyorduk. Yüz tanıma esnasında gözleri, burnu, ağzı veya kafa şeklini, saçlarımızı kullanıyor muyduk? Beynimiz bunları nasıl analiz ediyor, nasıl kodlanmış olabilir ki? David Hubel ve Torsten Wiesel bize göstermiştir ki beynimiz çizgileri, kenarları, hareketleri, görüntünün belirli özelliklerini belirli sinir hücreleri ile anlayabiliyoruz. Bütün bir görseli parçalayarak veya parçalanmış bir görseli kullanarak oluşturulabilecek bir bütünden anlamlı sonuçlar çıkartabiliyoruz. Yüz tanıma ise bütün bir görüntüden anlamlı özelliklerin ayıklanması ve onların sınıflandırılarak karşılaştırılması ile oluyor.

Bir yüzün geometrik özelliklerine göre yapılacak yüz tanıma işlemi, muhtemelen yüz tanıma için en kolay yaklaşımdır. İlk otomatik yüz tanıma sistemlerinden biri Kanade73: işaretleyici noktaları (gözler, kulaklar ve burun pozisyonu) özellik vektörü (noktaları arasındaki mesafe, bunlar arasındaki açı) oluşturmak için kullanıldı. Özellik vektörünü kullanarak yapılan tanımada kaynak ve referans görüntünün özellik vektörleri arasındaki Öklid mesafe hesaplanarak yaptı. Bu gibi bir yöntem başarılı oldu, doğası gereği parlaklık gibi değişikliklere karşı dayanıklıydı, ancak çok büyük bir dezavantajları davardı. Geometrik yüz tanıma yöntemi ile yapılan bir başka çalışma [Bru92]. A 22 boyutlu özellik vektörü kullandı ve büyük veri setleri üzerinde deneyler yaptı. Tek başına geometrik özelliklerin yüz tanıma için yeterli olmayacağı bu çalışma ile fark edilmiştir.

[TP91], Eigenfaces yöntemi, yüz tanıma için bütünsel bir yaklaşım aldı. Yüz görüntüsünün bir noktasından yüksek boyutlu görüntü alanı ve küçük boyutlu bir temsil alındı ve sınıflandırma kolay hale getirildi. Doğrusal diskriminant analizi ile bir sınıfa özel projeksiyon [BHK97] yöntemi olarak yüz tanımada uygulandı. Temel fikri, sınıflar arasında varyansı maksimize ederken, bir sınıf içinde varyansı en aza indirmekti.

OpenCV 2.4 sürümü ile birlikte bazı yüz tanıma algoritmaları geldi. Mevcut algoritmalar şunlardır:

- Eigenfaces (createEigenFaceRecognizer())
- Fisherfaces (createFisherFaceRecognizer())
- Local Binary Patterns Histograms LBPH (createLBPHFaceRecognizer())

Yüz tanıma için bazı örnek yüz verilerine ihtiyacımız olacaktır. Kendi veri kümenizi oluşturabilir veya hazır olarak oluşturulmuş veri kümesini indirerek kullanabilirsiniz.

- http://face-rec.org/databases/
- http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
- http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
- http://vision.ucsd.edu/content/yale-face-database

Algoritmalar için Veri Setinin Hazırlanması

Elde ettiğimiz verileri programımız tarafından okuyabiliyor olmamız gerekir. Bunun için basit bir yöntem olan CSV dosyasını tercih edeceğiz. Neden CSV? Çünkü CSV platform bağımsız bir dosya formatı. Daha farklı çözümlerde kullanılabilir.

/path/to/image.ext;0

Buradaki adres görüntünün dosya yoludur. Windows kullanıyor iseniz dosya yolunuz muhtemelen C:/faces/person0/image0.jpg şeklinde olacaktır. Dosya adresinden sonra ";" ayracı vardır ve bundan sonra görüntü etiketi 0 olarak atanır. Bu etiket görüntüye eklenen bir veri olarak düşünülebilir örneğin kaydını aldığınız kişinin id bilgisi olabilir.

AT&T Facedatabase'den (http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html) at.txt ye karşılık gelen CSV dosyasını indirin.

```
./at/s1/1.pgm;0
./at/s1/2.pgm;0
...
./at/s2/1.pgm;1
./at/s2/2.pgm;1
...
./at/s40/1.pgm;39
./at/s40/2.pgm;39
```

Dosyaları D:/data/at altına çıkarttım. CSV dosyam artık D:/data/at.txt dizinde. Şimdi tüm dosyada bir arama yaparak ./ olan yerleri D:/data/ şekline getiriyorum böylelikle dizini göstermiş oluyorum. Herhangi bir metin editörü (notepad, notepad++ vb.) ile bu işlemi yapabilirsiniz.

Artık demo uygulamaya dosya dizinini vererek çalıştırabilirsiniz. Örneğin:

facerec_demo.exe D:/data/at.txt

CSV Dosyası Oluşturma

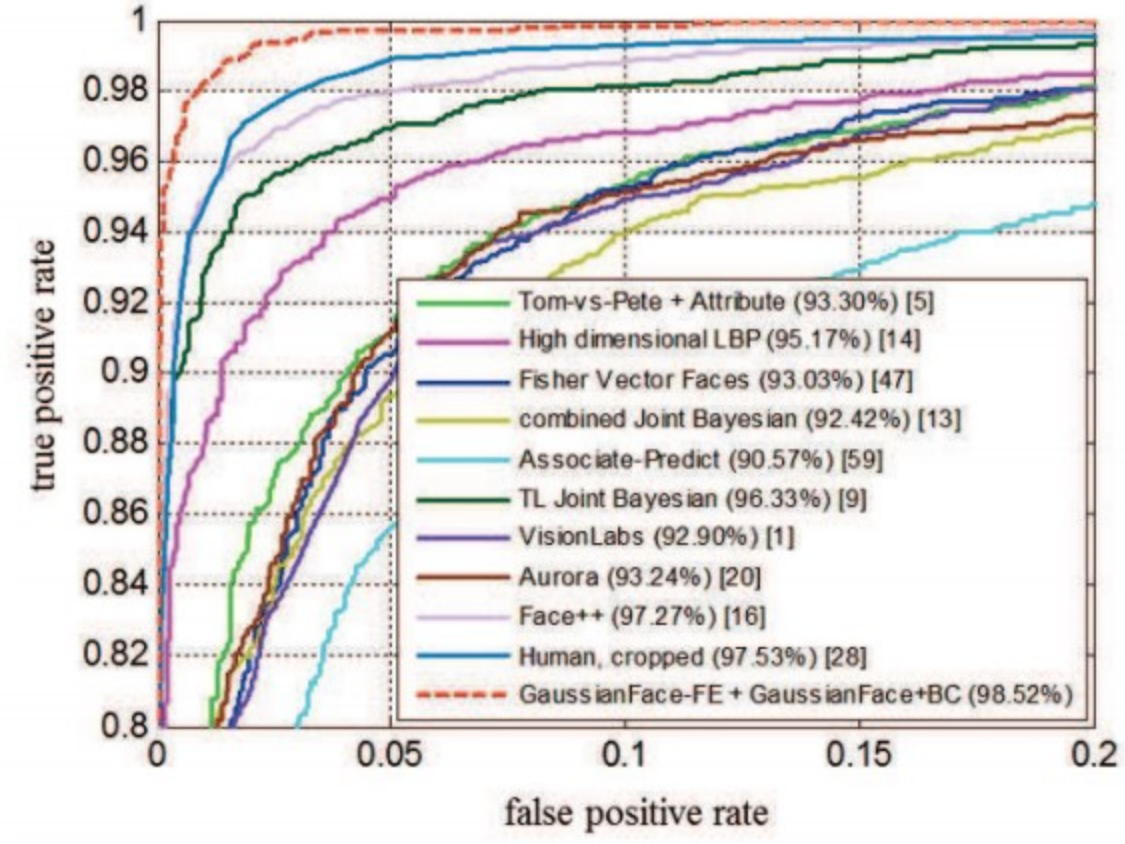
Elle manuel olarak CSV dosyası oluşturmak sıkıcı bir iş. Bunu yapması için create_csv.py adında bir Python scriptimiz var. Bu script src / create_csv.py dizininde mevcuttur, otomatik olarak bizim için CSV dosyası oluşturacaktır.

(/basepath/<subject>/<image.ext>) bu şekilde hiyrerarşik görüntülere sahipseniz:

```
philipp@mango:~/facerec/data/at$ tree
.
|-- s1
|   |-- 1.pgm
|   |-- ...
|   |-- 10.pgm
|-- s2
|   |-- 1.pgm
|   |-- ...
|   |-- 10.pgm
...
|-- s40
|   |-- 1.pgm
|   |-- ...
|   |-- 10.pgm
```

Sadece klasör yolunu vererek create_csv.py scriptini çalıştıralım:

```
philipp@mango:~/facerec/data$ python create_csv.py
at/s13/2.pgm;0
at/s13/7.pgm;0
at/s13/6.pgm;0
at/s13/9.pgm;0
at/s13/5.pgm;0
at/s13/3.pgm;0
at/s13/4.pgm;0
at/s13/10.pgm;0
at/s13/8.pgm;0
at/s13/1.pgm;0
at/s17/2.pgm;1
at/s17/7.pgm;1
at/s17/6.pgm;1
at/s17/9.pgm;1
at/s17/5.pgm;1
at/s17/3.pgm;1
[...]
```



Geliştirilen Bazı Yüz Tanıma Algoritmaların Karşılaştırılması

YÜZ TANIMA EİGENFACES, FİSHERFACES, LBPH

Daha önce yüz tanımaya giriş yapmıştık, bu defa yüz tanıma için OpenCV de üç adet algoritma olduğunu belirtmiştik (Eigenfaces, Fisherfaces, LBPH). Örnek olarak kullanılabilecek veritabanlarına da değinmiştik bu örnekte de att örnek yüz veri tabanını kullanacağız. Örnek uygulamayı JavaCV ile gerçekleştirereceğiz.

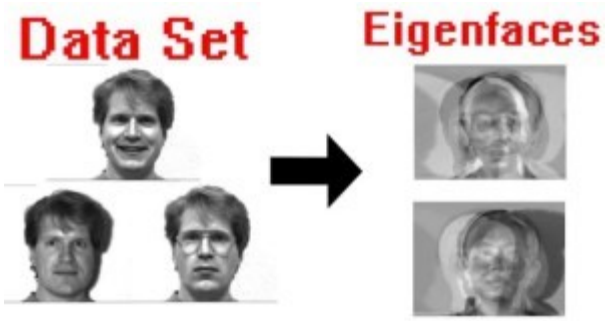
Bu algoritmalar için yapılmış birçok çalışma mevcut daha akademik bir çalışma içerisindeyseniz veya algoritmaların mantığını kavramak istiyorsanız bu konuda bilgi edinebileceğiniz bazı kaynakların linkine yazının altından erişebilirsiniz.

Bu algoritmalar öncelikle bir eğitim gerektirir. Eğitim yüz resimlerinin algoritmaya verilerek bu görüntülerden bir vektör oluşturur bu vektör eğitimde kullanılan yüzlerin belirgin özelliklerinden oluşur. Bu sayede elde edilen vektör veri tabanındaki tüm yüzlerden yararlanarak oluşturulmuş olur. Daha sonra karşılaştırma için gönderilecek yüzler bu vektör ile karşılaştırılır. Her eğitim sonrası, bu vektör değişir ve eğitim için kullanılan veri ne kadar fazla ise başarı oranı da bir o kadar artar.

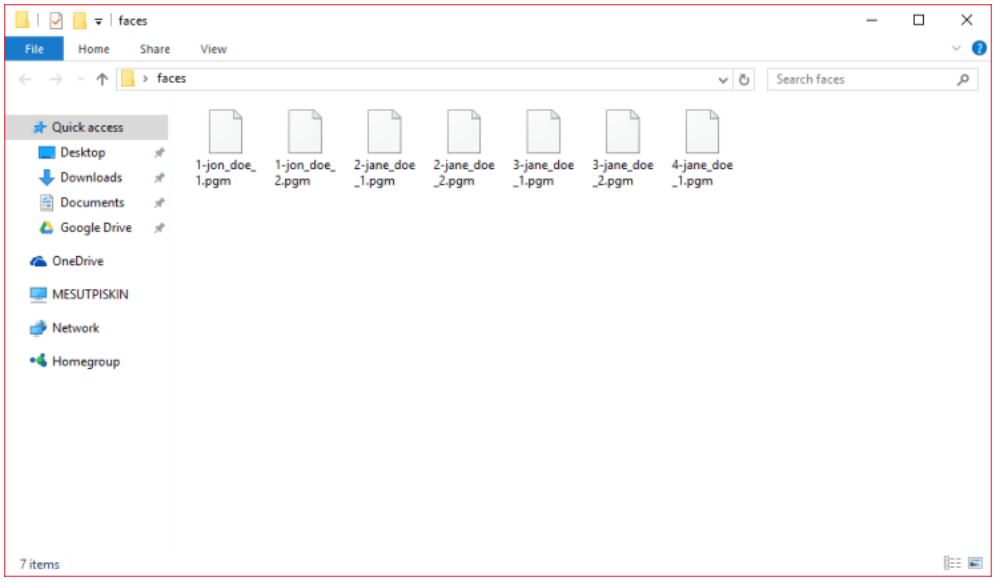
Algoritmanın Eğitilmesi

Eğitim için en az iki adet görüntüye ihtiyaç vardır ve eğitim için kullanılacak her dosyanın genişliği ve yüksekliği aynı boyutta olmalıdır. Eğitim için örnek veritabanlarını indirip kullanacağınız gibi kendinizde çekmiş olduğunuz resimleri kullanabilirsiniz. Kullanılacak görsellerin formatları genellikle jpg, jpeg, png, pmg vb. dir. Att veritabanını indirirseniz, içerisindeki görüntü formatları pmg dir.

OpenCV de eğitim için FaceRecognizer sınıfı altında train() metodu mevcuttur. Bu metot parametre olarak eğitim için kullanılacak mat vektörlerini yani yüzleri ve bu yüzler için kullanılacak etiketleri (label) alır. Etiket kavramını daha önce açıklamıştık fakat tekrardan değinmek gerekirse, eğitilen her yüz görüntüsünün eşleşme sonucunda adlandırılması için verdiğimiz id numaralarıdır. Bu etiketler görüntü dosyasının isimlerinde kullanılır. Örneğin att içerisinde 1.pgm 2.pgm olarak giderken bazı veritabanlarında 1-jon_doe_1.pgm veya 1-jon_doe.jpg gibi olabilir. Bu etiketler yüzün kime ait olduğunu belirlemek için kullanılabilir.



Bizim kullanacağımız att veritabanında 1.pgm olarak adlandırılmış fakat benimde kullanışlı bulduğum 1-jon_doe_1.pgm etiketlendirmesini kullanacağız. 1-jon_doe_1.pgm isimlendirmesinde ilk rakam kişinin id numarası, ikincisi tahmin edebileceğiniz üzere kişinin ismi ve sonuncu rakam ise bu kişinin veritabanında ki yüz sayısına göre verilmiş id si, örneğin jon doe kişisinin veritabanında 2 adet yüzü varsa isimlendirmesi 1-jon_doe_1.pgm ve 1-jon_doe_2.pgm şeklinde olacaktır. Eğitim için kullanacağımız yüz görüntülerini okurken gri renk uzayına çevireceğiz.



Hazırladığım örnek veri seti

Eşleştirme

Öncelikle eğitim için kullanacağımız yüz veri setinin dosya dizinine giderek buradaki tüm görüntüleri alacağız. Bu görüntülerin etiketlerini almak için split edeceğiz, bu işlemin ardından görüntüleri ve etiketleri train() metoduna vererek eğitimi tamamlayacağız. FaceRecognizer sınıfınının predict metodu ile eşleştirme için bir yüz vereceğiz. Bu metot işlem sonucunda aradığımız yüz veri setinde eşleşiyor ise yani yüz veritabanında var ise bu yüzün etiketini döndürecektir. predict metodunun farklı overloadları vardır bunlara buradaki (http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_api.html) bağlantıdan ve şekilden göz atabilirsiniz. Eşleşme değerlerini ve birden fazla kayıt ile eşleşiyor ise, bunların hepsini almak için parametre olarak double ve int tipinde diziler metoda veriler. Bu diziler işlem sonucu etiketler ve eşleşme oranları ile doldurulur. Veya sadece eşleştirme resmini parametre olarak vererek eşleşen yüzün etiketini alabilirsiniz.

```
class FaceRecognizer : public Algorithm
{
public:
    //! virtual destructor
    virtual ~FaceRecognizer() {}

    // Trains a FaceRecognizer.
    virtual void train(InputArray src, InputArray labels) = 0;

    // Updates a FaceRecognizer.
    virtual void update(InputArrayOfArrays src, InputArray labels);

    // Gets a prediction from a FaceRecognizer.
    virtual int predict(InputArray src) const = 0;

    // Predicts the label and confidence for a given sample.
    virtual void predict(InputArray src, int &label, double &confidence) const = 0;

    // Serializes this object to a given filename.
    virtual void save(const string& filename) const;

    // Deserializes this object from a given filename.
    virtual void load(const string& filename);

    // Serializes this object to a given cv::FileStorage.
    virtual void save(FileStorage& fs) const = 0;

    // Deserializes this object from a given cv::FileStorage.
    virtual void load(const FileStorage& fs) = 0;

    // Sets additional information as pairs label - info.
    void setLabelsInfo(const std::map<int, string>& labelsInfo);

    // Gets string information by label
    string getLabelInfo(const int &label);

    // Gets labels by string
    vector<int> getLabelsByString(const string& str);
};
```

Predict Metodunun Overloadları

Aşağıdaki örnek kodda Eigenfaces, Fisherfaces, LBPH algoritmaları kullanılabilir durumdadır, kullanmak istediğiniz algoritmayı seçerek diğerlerini yorum satırı haline getirmeniz yeterli olacaktır. Uygulamaya istediğiniz gibi görsellik katarak projelerinize uyarlayabilirsiniz.

```
import java.io.File;
import java.io.FilenameFilter;
import java.nio.IntBuffer;

import static org.bytedeco.javacpp.opencv_core.CV_32SC1;
import static org.bytedeco.javacpp.opencv_core.CV_8UC1;
import static org.bytedeco.javacpp.opencv_face.createFisherFaceRecognizer;
import static org.bytedeco.javacpp.opencv_face.createEigenFaceRecognizer;
import static org.bytedeco.javacpp.opencv_face.createLBPHFaceRecognizer;
import org.bytedeco.javacpp.BytePointer;
import org.bytedeco.javacpp.opencv_face.FaceRecognizer;
import org.bytedeco.javacpp.opencv_core.Mat;
import org.bytedeco.javacpp.opencv_core.MatVector;
import org.bytedeco.javacpp.opencv_face;
import static org.bytedeco.javacpp.opencv_highgui.cvWaitKey;
import static org.bytedeco.javacpp.opencv_highgui.imshow;
import static org.bytedeco.javacpp.opencv_imgcodecs.CV_LOAD_IMAGE_GRAYSCALE;
import static org.bytedeco.javacpp.opencv_imgcodecs.imread;


public class OpenCVFaceRecognizer {

    public static void main(String[] args) {
        //Eğitim için kullanacağım veri setinin dizini
        String trainingDir = "C:/Users/mesutpiskin/Desktop/faces/";
        //Eşleştirme için kullanacağım diğer yüz
        Mat testImage = imread("C:/Users/mesutpiskin/Desktop/40.pgm",
CV_LOAD_IMAGE_GRAYSCALE);

        File root = new File(trainingDir);

        FilenameFilter imgFilter = new FilenameFilter() {
            public boolean accept(File dir, String name) {
                name = name.toLowerCase();
```

```
                return name.endsWith(".jpg") || name.endsWith(".pgm") ||
name.endsWith(".png");
            }
        };

        File[] imageFiles = root.listFiles(imgFilter);

        MatVector images = new MatVector(imageFiles.length);

        Mat labels = new Mat(imageFiles.length, 1, CV_32SC1);
        IntBuffer labelsBuf = labels.createBuffer();

        int counter = 0;

        for (File image : imageFiles) {
            Mat img = imread(image.getAbsolutePath(), CV_LOAD_IMAGE_GRAYSCALE);

            int label = Integer.parseInt(image.getName().split("\\-")[0]);
            System.out.println("Eğitilen Yüz: "+label);
            images.put(counter, img);
            labelsBuf.put(counter, label);
            counter++;
        }
         FaceRecognizer faceRecognizer = createEigenFaceRecognizer();
        //FaceRecognizer faceRecognizer = createFisherFaceRecognizer();
        //FaceRecognizer faceRecognizer = createLBPHFaceRecognizer();

        faceRecognizer.train(images, labels);


        int predictedLabel =  faceRecognizer.predict(testImage);
        faceRecognizer.predict(testImage);
        System.out.println("Bulunan Yüz ID: " + predictedLabel);
        cvWaitKey(0);
    }
}
```

İşlem Sonucu