



BALIKESİR ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

2016-2017 GÜZ

LİSANS ARAŞTIRMA PROJESİ
YÜZ TANIMA SİSTEMİ GELİŞTİRİLMESİ

YRD.DOÇ.DR. GÜLTEKİN KUVAT

201220407039 – AHMET YAYLALIOĞLU

İÇİNDEKİLER

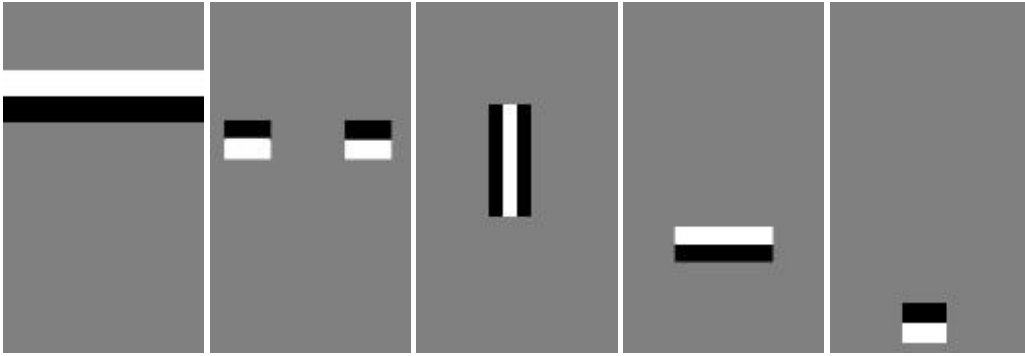
Giriş.....	3
Yüz Algılama	3
Algılanan Yüzü İşleme.....	5
- Gözlerin konumunu dikkate alarak görüntüyü kırpma ve gri tonlama	6
- Histogram Eşitleme (Histogram Equalization) Tekniği	7
- Bilateral Filtresi ile Görüntüyü Yumuşatma	11
Yüz Tanıma	13
OpenCV ile Yüz Tanıma.....	20

Giriş

Yüz tanıma işlemini üç aşama ile gerçekleştirebiliriz. Bunlar sırasıyla; görüntüdeki insan yüzünün algılanması, algılanan yüzün belirli dönüşümlere uğraması ve son olarak veri setleri ile öğrenme ile birlikte yüz tanıma tahmininin gerçekleştirilmesidir. İlk olarak teorik anlatımlar yapılacak olup, daha sonra programın testleri ile ilgili bilgi verilecektir.

Yüz Algılama

Yüz tanıma işleminden önce, gerçek zamanlı görüntüdeki insan yüzleri algılanmalıdır. Sağlıklı insanların yüz özellikleri (gözlerin simetrikliği, burnun, alnın ve çenenin yüz üzerindeki konumları vb.) temel olarak aynıdır. Bu yüzden, genel anlamda sağlıklı bir insanın yüzünü temsil edecek parametreler (burun, gözler, ağız, çene ve alın) şekil 1'deki gibi tanımlanabilir.



Şekil 1: insan yüzünü temel anlamda temsil edecek parametreler (Arubas, 2013)

İnsan yüzünü temsil etmek için kullandığımız bu parametreler, yüz üzerinde bulunan göz, burun, ağız gibi organlarımızın, ve bunlar dışında kalan yerlerin birbirlerine göre oluşan zıtlığından (kontrast farkından) faydalanılarak tasarlanmıştır. Bu parametreleri tek bir görüntüde birleştirirsek, şekil 2'deki görüntü meydana gelir.



Şekil 2:insan yüzünü temsil eden parametrelerin tek görüntüde birleştirilmesi (Arubas, 2013)

Yüzün belirli kısımlarını bu şekilde küçük parçalar halinde tanımladıktan sonra,görüntü işlemede kullanılan bir teknik olan “şablon eşleme (template matching)” tekniği ile bu küçük parçaların,görüntü üzerinde belirli kısımlara denk gelip gelmediğini incelenerek,görüntüde bir insan yüzü olup olmadığına karar verilir.Şekil 3’de binary görüntü (siyah piksel noktaları 0,beyaz piksel noktaları 1 değerli) üzerine yüz parametrelerinin uygulanması görülmektedir.

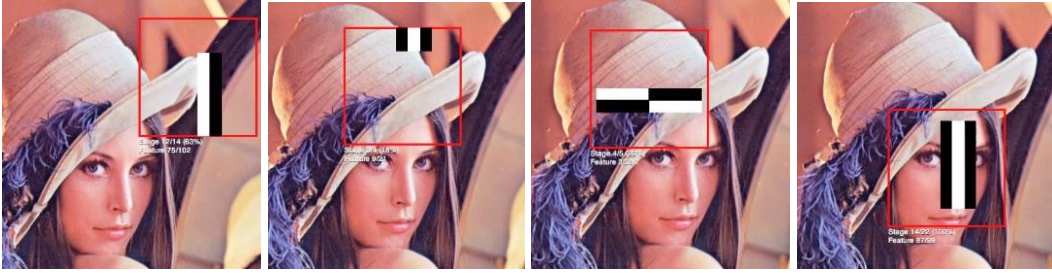


Şekil 3:Yüz parametrelerinin ikili görüntü üzerine uygulanışı

Bu yüz parametrelerinin yüzleri oluşturması hakkında istatistikler toplanırsa,kuracağımız algoritmalar doğru özelliklerin doğru konumlarda kullanılması için eğitilebilir.Projemde kullandığım OpenCV kütüphanesi içinde bu istatistiksel veriler, özel xml formatlı dosyalar biçiminde gelmektedir.

Şablon eşleme tekniği; şablon görüntüde var olan belirlediğimiz özel kısımları, gerçek görüntü üzerinde eşleştiren sayısal görüntü işleme tekniklerinden biridir.Görüntüdeki objelerin kenarlarını algılama,hareket eden robotların ilerlemesinde ve üretimde kalite kontrol aşamasında kullanılır.

Şekil 3’de belirtilen işlemin gerçek anlamda işe yarar şekilde uygulanması için ,elimizde bulunan görüntünün insan yüzü içerip içermediğini hızlı bir şekilde kontrol etmek gerekir.Algoritmanın,görüntü içerisinde incelediği bölünmüş kısımların insan yüzü içerip içermediğini tek tek kontrol etmek yerine; yüz parametreleri, görüntünün incelenen kısmında insan yüzünü temsil edecek şekilde konumlandırılmadığı tespit edildiği anda bir sonraki kısmı taramak daha hızlı bir sonuç almamızı sağlayacaktır.Bu işleme basamaklı süreç (cascading process) adı verilir.Bu yöntem,OpenCV kütüphanesinin içerdiği ‘Haar Cascade’ olarak bilinen Viola-Jones tekniğinin temel yapısıdır.



Şekil 4:Yüz tanıma algoritmasının görsel olarak uygulanışı(bu görüntüleri alıntı yaptığım Adam Harvey’in videosunu izlemenizi öneririm; <https://vimeo.com/12774628>)

Algılanan Yüzü İşleme

Algılanan yüzü,daha sonra yüz tanıma (kimliklendirme) işlemi için kullanmak üzere işleyip veritabanına kaydetmemiz gerekir.Bu işlemi görüntünün gürültüden arındırılması ve yüzün belirginleşmesi için yapmaktayız.Bu sayede daha verimli bir yüz tanıma işlemi gerçekleştirilebilir.

Yüz işleme işlemini 3 aşamada ele alabiliriz;

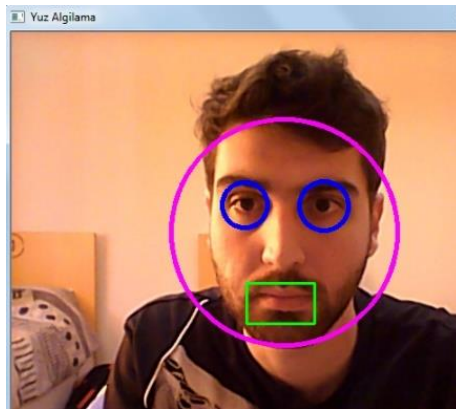
- 1)Gözlerin konumunu dikkate alarak görüntüyü kırpma işlemi ve gri tonlama işlemi
- 2)Histogram Eşitleme tekniğinin görüntüye uygulanması (histogram equalization)
- 3)Bilateral Filtresi kullanarak görüntüyü yumuşatma

Bu 3 adım algılanan yüze uygulandıktan sonra,oluşan yeni görüntümüz daha sonra kullanılmak üzere veritabanına kaydedilmektedir.

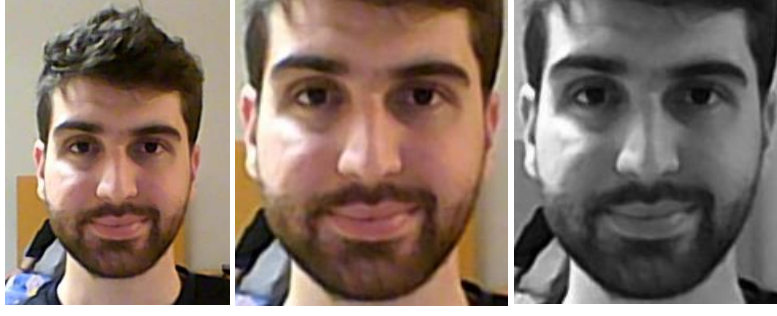
- Gözlerin konumunu dikkate alarak görüntüyü kırpma ve gri tonlama

Algılanan yüzü işlemeye gözlerin algılanmasından başlamalıyız. Aslında yüz görüntüsünde kullanabileceğimiz birçok parametre (ağız, burun, gözler, çene vb.) vardır. Ancak bu özelliklerin çoğu düz bir ten görünümünde olduğundan bunları sanal ortamda tanımlamak zorlaşır. Bu yüzden yüz tanıma (kimliklendirme) işlemi için bize en faydalı olacak olan yüz parametresi gözlerdir. Gözlerin yuvarlak bir yapısı olması gözlerin bulunduğu noktayı tespit etmeyi büyük ölçüde kolaylaştırır. Ayrıca yüz görüntüsünü dikey bir biçimde ortadan ikiye ayırdığımızda gözlerin bu dikey çizgiye göre simetrik oluşu kimlik tespitinde büyük rol oynar. Bu işlem sayesinde gözlerin olduğu konuma göre görüntüyü kırptığımızda, bize lazım olmayan arka plan öğelerinden ve yüz üzerindeki kullanmayacağımız unsurlardan (saçlar, kulaklar, çenenin alt kısmı gibi) kurtulabiliriz. Kırpma işleminden sonra yüz özelliklerini daha ön plana çıkarmak için görüntü gri tonlamalı bir hale dönüştürülür.

OpenCV kütüphanesi içerisinde yüzü algılama kısmında olduğu gibi, gözleri algılamak için de istatistiksel veri dosyası (xml formatlı) bulunmaktadır. Bu sayede gözlüklü insanlar da dahil olmak üzere, insan yüzleri üzerinde gözlerin bulunduğu alanlar tespit edilebilmektedir (Şekil 5). Bu kırpma işleminden sonra elde edilen ve gri tonlama yapılan görüntü ise Şekil 6'da görülmektedir.



Şekil 5: OpenCV istatistiksel veri dosyaları kullanarak, görüntüdeki yüz, göz ve ağızın tespiti



Şekil 6: Görüntünün göz konumlarına göre belirli ölçüde kırılması ve gri tonlama uygulanması

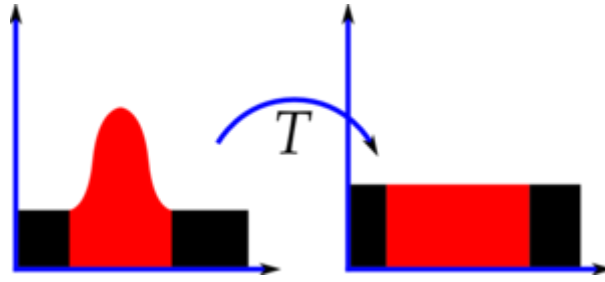
Projemde, görüntüyü göz konumlarına göre kırma işlemini 3 farklı yol ile denedim. İlki programın kendisi içinden algılanan yüzün geometrik bir algoritma kurularak kırılmasıydı. İkinci kullandığım yöntem ise, iki gözün dış sınır noktalarının pixel koordinatlarını Paint programı kullanarak tespit edip, bu pixel noktalarını bir python scriptine girerek resmi kırpmaktı. Son olarak kullandığım yöntem ise gri tonlamalı olarak kaydedilen görüntüyü bir fotoğraf editörü ile (photoscape, photoshop) gerekli noktalarından kırpmaktı. Projenin uygulanış alanına göre bu 3 yoldan biri seçilebilir. Daha sonra ayrıntılı olarak açıklayacağım EigenFaces yüz tanıma algoritmasını verimli kullanmak için veritabanına kayıtlı olan kırpılmış yüz görüntüleri eş ölçekli ve olabildiğince küçük olmalıdır. Ben projemde 70 x 70 ölçülerinde veri setleri kullandım.

- Histogram Eşitleme (Histogram Equalization) Tekniği

Gerçek zamanlı yüz tanıma işleminde en büyük sorunlardan biri ortam şartlarıdır. Bu ortam şartlarının en önemlisi de ortam ışığıdır. Gürbüz (robust) bir kontrol sistemi oluşturabilmek için kullanacağımız teknikler ile ortam ışığını en aza indirmeyi hedeflemeliyiz. Ortam ışığı yüzün bir tarafına çok fazla gelebilir bu yüzden yüzün diğer yarısı gölgeli olabilir ya da ortamda çok az ışık bulunabilir. Bu gibi durumların programımıza en az şekilde etki etmesi için histogram eşitleme tekniğinden yararlanabiliriz. (Daniel Lélis Baggio, 2012)

Histogram eşitleme; görüntüdeki renk frekanslarını kullanarak kontrast (zıtlık) düzeltilmesi yapmaya yaran görüntü işleme tekniklerinden biridir. Bu teknik ,görüntüdeki kullanılabilir (işimize yarayan) kontrast değerleri birbirlerine yakın olduğu zaman, görüntünün

genel kontrast deęerini artırmıř olur.Histogram eřitleme teknięi grntde belirli bir noktada yksek olan renk yoęunluęunu grntnn geneline yayar ve histogram grafięinde bu yoęunluk deęeri genele daęıtılır.Bu sayede dřk yerel kontrast alanına sahip blgelerin kontrast deęerleri ykselir.řekil 7’de grldę gibi renk yoęunluęu yksek olan kısımların yoęunluęu genele daęıtıldığında kontrast deęerleri eřitlenip,grntnn genel kontrast deęeri de belirli bir miktar ykselmiřtir.

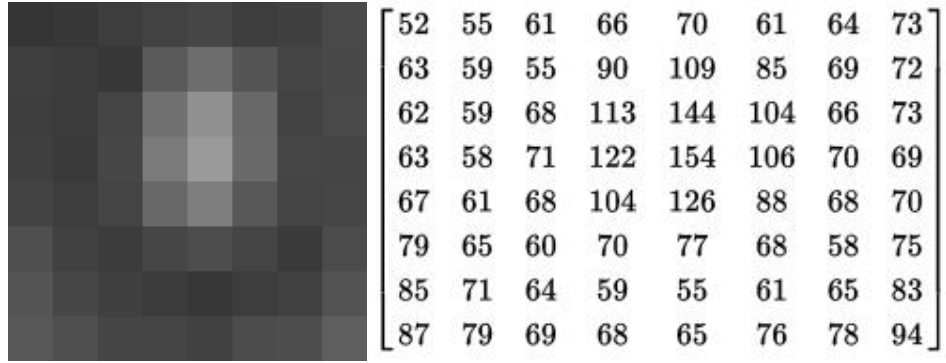


řekil 7:ilk grafik grntnn normal renk yoęunluk deęerleri,ikinci grafik histogram eřitlięi teknięinin uygulanmasından sonraki yoęunluk deęerleri (Wikipedia-Histogram equalization)

Bu yntem ařırı parlak ortamlarda veya eřit ve dřk arkaplan-nplan deęerlerine sahip grntlerde olduka yararlıdır.zellikle Xray grntleme sistemlerinde kemik yapılarının daha belirgin řekilde ortaya ıkmasında,detayların daha iyi grntlenmesinde byk rol oynar.Bu metodun en nemli avantajı uygulanabilirlięinin olduka basit olması ve geri dndrlebilir bir operatr olmasıdır.Teoride,histogram eřitleme fonksiyonu biliniyorsa,orijinal histogram grafięi geri elde edilebilir.Yntemin dezavantajı ise tm grnty ele alarak iřlem yapmasıdır.Yani grntnn, grlt deęerlerinin kontrastını da artıracaktır.Bu yzden bu grlty azaltmak,grntye yumuřatma uygulamak iin bilateral filtresi uygulayacaęız.

Histogram eřitlemesi genellikle fotoęraflarda gereki olmayan efektler oluřmasına sebep olur.Ancak termal,uydu ve X-Ray grntleri gibi bilimsel grntlerin kullandığı sahte-renk (false-color) grnt sınıfı iin ok yararlıdır.Ayrıca histogram eřitlemesi dřk renk derinlięine sahip grntlerde de istenmeyen etkiler oluřturur.Bu yzden yntemden verim alabilmek iin, srekli veri deęerli veya en az 16-bit’lik gri tonlamalı deęerli renk derinlięi olan grntlere uygulanmalıdır.

Histogram eşitliğini 8 X 8'lik gri tonlamalı bir görüntü ile örneklendirecek olursak;



Şekil 8:8 bitlik gri tonlamalı pikseller ve görüntünün matris (0-255 değer arası) gösterimi (Wikipedia-Histogram equalization)

Histogram eşitliği'ni uygulamak için kümülatif dağılım fonksiyonundan(Cumulative distribution function – CDF) yararlanılır.Önce şekil 8'deki piksel değerlerinin varyansları bir tabloda gösterilir.Daha sonra CDF uygulanarak bu değerler piksellere atanır.

52	1	64	2	72	1	85	2	113	1
55	3	65	3	73	2	87	1	122	1
58	2	66	2	75	1	88	1	126	1
59	3	67	1	76	1	90	1	144	1
60	1	68	5	77	1	94	1	154	1
61	4	69	3	78	1	104	2		
62	1	70	4	79	2	106	1		
63	2	71	2	83	1	109	1		

Şekil 9:piksel değerleri varyans tablosu (Wikipedia-Histogram equalization)

CDF piksel değerlerinin birbirlerine göre öneminin ,piksel sayısına oranı .(rank/pixelcount) ile hesaplanır.Histogram eşitliği değerlerini hesaplamak için aşağıdaki formül kullanılır.

$$h(v) = \text{round} \left(\frac{cdf(v) - cdf_{min}}{(M \times N) - 1} \times (L - 2) \right) + 1$$

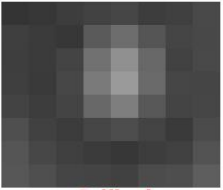
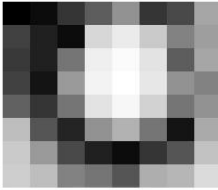
Burada M X N görüntünün aktarıldığı matrisin boyutları, L (8-bitlik görüntü olduğu için) 256, cdf'in minimum değeri (52 değerlikli piksel için) 1, maksimum değeri (154 değerlikli piksel için) 64 tür.

v, Pixel Intensity	cdf(v)	h(v), Equalized v			
52	1	0	71	39	154
55	4	12	72	40	158
58	6	20	73	42	166
59	9	32	75	43	170
60	10	36	76	44	174
61	14	53	77	45	178
62	15	57	78	46	182
63	17	65	79	48	190
64	19	73	83	49	194
65	22	85	85	51	202
66	24	93	87	52	206
67	25	97	88	53	210
68	30	117	90	54	215
69	33	130	94	55	219
70	37	146	104	57	227
			106	58	231
			109	59	235
			113	60	239
			122	61	243
			126	62	247
			144	63	251
			154	64	255

Şekil 10: Hesaplanan cdf değerlerini ve $h(v)$ formülüne göre hesaplanan histogram eşitlik değerlerini (Wikipedia-Histogram equalization)

Bulunan histogram eşitlik değerlerini matrisin önceki değerlerinin yerine koyarsak, matrisin son hali ve 8 bitlik gri tonlamalı görüntümüzün ilk ve son hali şekil 11'deki gibi olur.

0	12	53	93	146	53	73	166
65	32	12	215	235	202	130	158
57	32	117	239	251	227	93	166
65	20	154	243	255	231	146	130
97	53	117	227	247	210	117	146
190	85	36	146	178	117	20	170
202	154	73	32	12	53	85	194
206	190	130	117	85	174	182	219

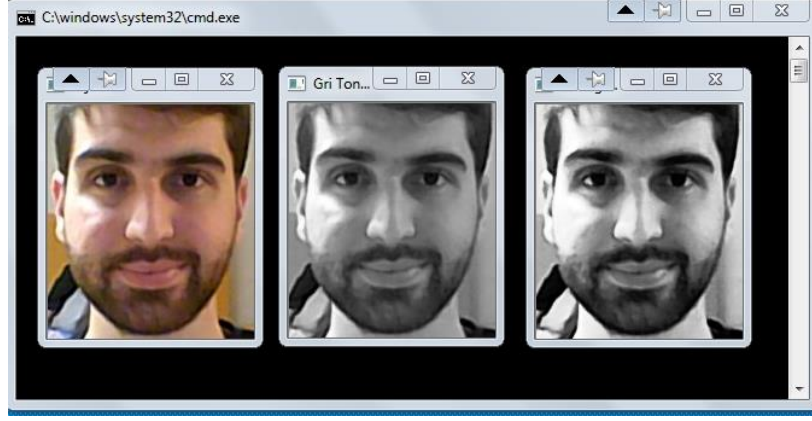



Orijinal
Histogram Eşitliği Uygulanmış

Şekil 11: Yeni oluşan görüntü matrisi ve orijinal görüntü ile yeni görüntünün karşılaştırılması (Wikipedia-Histogram equalization)

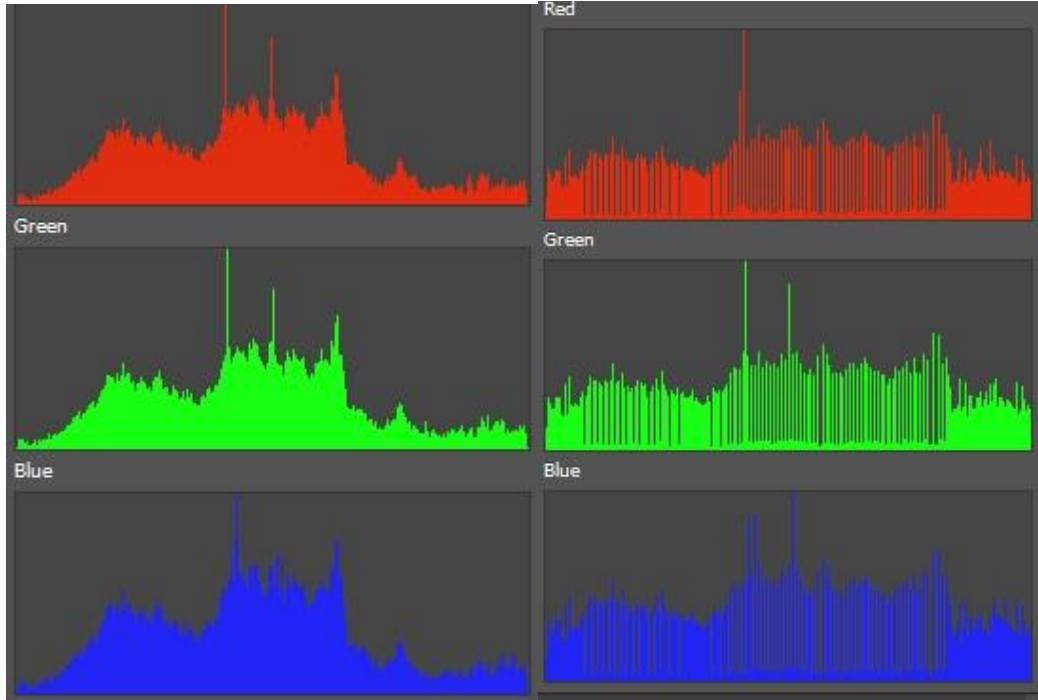
Ayrıca merak edilen bir noktayı da burada açıklamak faydalı olacaktır. Hatırlarsanız histogram eşitlemeden önce görüntüyü gri tonlamalı bir hale dönüştürmüştük. Bunun en önemli sebebi histogram eşitliğini RGB(kırmızı, yeşil, mavi) görüntülere uygulayamıyor

olmamızdır.Üç ayrı renk kanalı olduğu için histogram eşitliği RGB görüntülerde istenmeyen,verimli olmayan sonuçlar meydana getirir. Zaten OpenCV ile bunu uygulamak istediğimizde program debug hatası verecektir.



Gri tonlamalı görüntü histogramı

Histogram eşitliği uygulanmış halı



Şekil 12:OpenCV ile histogram eşitliği uygulanması ve bunlara ait histogram grafikleri

- Bilateral Filtresi ile Görüntüyü Yumuşatma

Histogram eşitliğinin görüntünün her yerine uygulanmasından dolayı görüntünün arkaplanı gibi dikkati istenmeyen kısımlarının da kontrastının artacağını ve bu yüzden

gürültünün artacağını yukarıda belirtmiştik. Bilateral filtresi ile bu sinyal gürültüsünü azaltarak görüntüyü yumuşatmayı hedefliyoruz.

Bilateral filtre, görüntüler için doğrusal olmayan, kenarları koruyan ve gürültüyü azaltan yumuşatma filtresidir. Bir görüntüdeki her pikseldeki yoğunluk değeri, yakındaki piksellerin yoğunluk değerlerinin ağırlıklı ortalamasıyla değiştirilir. Bu ağırlık Gauss dağılımına dayanabilir. Ağırlıklar, ağırlıkların yalnızca Öklid uzaklıklarına değil, aynı zamanda radyometrik farklara (örn. renk yoğunluğu, derinlik mesafesi, vb. gibi uzaklık farklarına) bağlıdır. Bu, sistematik olarak her pikselden geçerek ve ağırlıkları bitişik piksellere göre ayarlayarak keskin kenarları korur.

Bilateral filtresi aşağıdaki matematiksel formül ile tanımlanır;

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

(Wikipedia-Bilateral Filter)

Burada;

W_p , normalleştirme ifadesidir ve aşağıdaki şekilde tanımlanır.

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

(Wikipedia-Bilateral Filter)

I: filtrelenecek olan orijinal görüntü

X : O an filtre uygulanan pikselin koordinatı

Fr : Yoğunluk farklarını yumuşatmak için kullanılan aralık sistemi. Bu bir Gauss fonksiyonu olabilir.

Gs : Koordinatlardaki farklılıkları düzeltmek için kullanılan uzaysal sistemdir.Bu bir Gauss fonksiyonu olabilir.



Şekil 13:Histogram eşitliği uygulanan görüntüye Bilateral filtre uygulanmasının sonucu

Yüz Tanıma

Yüz algılama kısmında,elimizdeki görüntüde bir yüz olup olmadığına karar verdikten sonra yüz tanıma aşamasında bu yüzün veritabanındaki hangi yüz ile eşleştiğini bulmaya çalışırız.

OpenCV kütüphanesi,yüz tanıma işlemleri yapabilmemiz için bize 3 yöntem sunar.Bunlar Eigenfaces,Fisherfaces ve Local Binary Patterns Histograms (LBPH) yöntemleridir.Ben projemde Eigenfaces ile yüz kimliklendirmeye çalıştım.

Her üç yöntem de anlaşılır hale getirilmiş olan yüz görüntüsünü,bilinen ve veri tabanına kaydedilen yüzlerin eğitim veri setleriyle (training data set) karşılaştırarak tanımayı gerçekleştirir.Eğitim veri setlerini oluştururken ya programımız anlık görüntü alır ve biz bu görüntünün kime ait olduğunu programa bildiririz ya da veri setlerini kendimiz oluşturup veri tabanı dosyasına görüntünün kime ait olduğunu yazarız.

Bu yöntemler eğitim setlerini farklı şekilde kullanırlar.Eigenfaces ve fisherfaces yöntemleri eğitim veri setinin en baskın özelliklerini bir bütün olarak ele alıp buna uygun

matematiksel bir tanım çıkartırken, LBPH eğitim setindeki her yüzü ayrı ayrı ve birbirinden bağımsız olarak analiz eder.

Görüntüleri sadece bir dizi sayıdan oluşan matris şeklinde ifade edebilmenin yanında, görüntüleri vektör olarak da temsil edebiliriz. Her vektör alanı orthogonal (tüm vektör çiftlerinin birbirine dik olduğu –yani skaler çarpımlarının sonucu sıfır olan vektör kümesi) bir temele sahiptir. Bu temel unsurlar birleştirilerek bu vektör uzayındaki her vektör oluşturulabilir ve tersi olarak vektör uzayındaki her vektör de temel unsurlarına ayrılabilir.

Görüntüleri bir dizi renk yoğunluğundan oluşan matrisler olarak ifade etmek yerine vektörler olarak ifade edebileceğimizi belirtmiştik. Örneğin 70 x 70 piksel boyutunda bir yüz eğitim veri setimiz varsa, bu eğitim setindeki görüntülerin her biri 4900 ($70 * 70$) boyutlu bir vektör olarak düşünülebilir. Artık bu vektörlerin bulunduğu vektör alanları üzerine konuşabilir, ve görüntülerin vektör alanını oluşturan özvektörlerini elde edebilmek için Ana Bileşenler Analizi (Principal Components Analysis) tekniğinden faydalanabiliriz. Kullandığım Eigenfaces yöntemi, eigen vektörleri üzerinden işlem yaptığı için, ana bileşenler analizinin matematiksel incelemesinden önce matlab’da eigenfaces yönteminin ana bileşenler analizini kullanarak nasıl bir sonuç meydana getirdiğini inceleyelim. Şekil 16’da ana bileşenler analizi (PCA) kullanılarak oluşturulmuş yeni veri setini görebilirsiniz.



Şekil 14: Kullanılan eğitim veri seti

```

baslangic = 0;
son       = 10;
ayrac     = '';
resim_uzantisi = '.jpg';
veriseti_klasoru = 'giris';
sonuc_klasoru = 'cikiss';
goruntuler = [];

for i=baslangic:son
    goruntu = imread(strcat(veriseti_klasoru, '/', ayrac, num2str(i),
resim_uzantisi));
    goruntu = double(rgb2gray(goruntu));
    [rows, cols] = size(goruntu);
    goruntu = goruntu(:)';
    goruntuler = vertcat(goruntuler, goruntu); %goruntuler dizisine o anki
goruntu matrisi ekleniyor
end

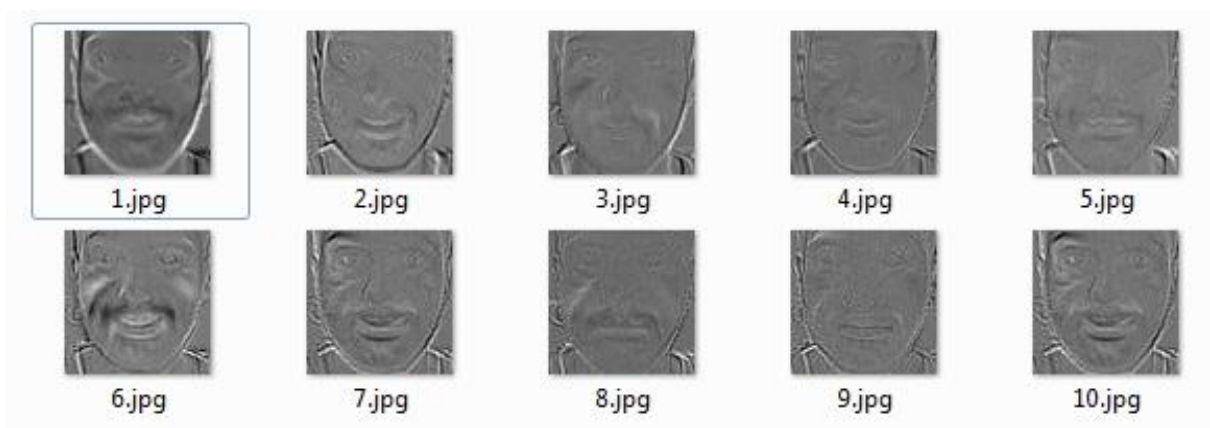
images_mean = mean(goruntuler);
goruntuler = goruntuler - ones(son - baslangic + 1, 1) * images_mean;

%Goruntulere Ana Bileşen analizi uygulanması:
[C, S, L] = princomp(goruntuler, 'econ');

%Goruntulerin Eigen vektörleri olarak kaydedilmesi
for i=1:size(C,2)
    goruntu = C(:,i);
    goruntu = reshape(goruntu,rows,cols);
    goruntu = ( goruntu - min(goruntu(:)) ) ./ ( max(goruntu(:)) -
min(goruntu(:)) );
    imwrite(goruntu, strcat(sonuc_klasoru, '/', num2str(i), resim_uzantisi));
end


```

Şekil 15: Ana bileşenler analizi ile eigen vektörlerinin elde edilmesi matlab kodu (Arubas, 2013)



Şekil 16: Ana bileşenler analizinden sonra veri eğitim setinin son hali

Eigenfaces yönteminin , veri setlerini bir bütün olarak ele aldığını ve bunların doğrusal bir kombinasyonu olduğunu belirtmiştik.Şimdi ise yüz tahmin işleminin nasıl yapıldığına basitçe bakalım.



Şekil 17:Öğrenmenin gerçekleştirilmesi

Bu şekilde bir öğrenme metodu ile katsayılar düzenlenip yüzün tahmin edilmesi gerçekleştirilmektedir.

Eigen değerleri ve vektörlerinin matematiksel olarak incelenmesine basit bir örnek üzerinden bakacak olursak,bu işlemleri 5 adım ile sınıflandırabiliriz.Örnek olarak aşağıda belirtilen 2 x 2 'lik matris için Eigen değerlerini ve bu değerlere karşılık gelen vektörleri bulalım.

$$A = \begin{pmatrix} 7 & 3 \\ 3 & -1 \end{pmatrix}$$

Çözümümüzün ilk adımı olarak eigen değerlerini temsil edecek olan λ ile birim 2 x 2'lik birim matrisi çarpalım.

$$\lambda * I = \lambda * \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$$

İkinci adım olarak λ ile çarpılmış birim matrisi,A matrisinden çıkaralım.

$$A - \lambda * I = \begin{pmatrix} 7 & 3 \\ 3 & -1 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = \begin{pmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{pmatrix}$$

Üçüncü adım olarak bu yeni matrisin determinantını alalım.

$$\begin{aligned} \det \begin{pmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{pmatrix} &= (7-\lambda) * (-1-\lambda) - (3) * (3) \\ &= -7 - 7\lambda + \lambda + \lambda^2 - 9 \\ &= \lambda^2 - 6\lambda - 16 \end{aligned}$$

Dördüncü adım olarak da, ikinci derece denklemi çözerek λ (eigen değerlerini) değerlerini bulalım.

$$\begin{aligned} \lambda^2 - 6\lambda - 16 &= 0 \Rightarrow (\lambda - 8) * (\lambda + 2) = 0 \\ \lambda_1 &= 8, \lambda_2 = -2 \end{aligned}$$

Beşinci ve son adım olarak da bu eigen değerlerine karşılık gelen eigen vektörlerini bulalım.

λ 'nın ilk değeri olan 8 'i ,ikinci adımdaki matriste yerine yerine koyduktan sonra sonra elde ettiğimiz matris ve bu matrisi X vektör matrisi ile çarpıp sıfır matrisine eşitledikten sonra, $\lambda=8$ değerine karşılık gelen eigen vektörü aşağıdaki gibi bulunur. λ 'nın diğer değeri olan -2 için de benzer işlemler tekrarlanır.

$$B = \begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix}$$

$$B * \bar{X} = \bar{0}$$

$$\Rightarrow \begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix} * \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow \begin{matrix} s_1 \\ s_2 \end{matrix} \begin{pmatrix} -1 & 3 \\ 3 & -9 \end{pmatrix} \Big| \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$3 * S_1 + S_2 \rightarrow S_2 \Rightarrow \begin{pmatrix} -1 & 3 \\ 0 & 0 \end{pmatrix} \Big| \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

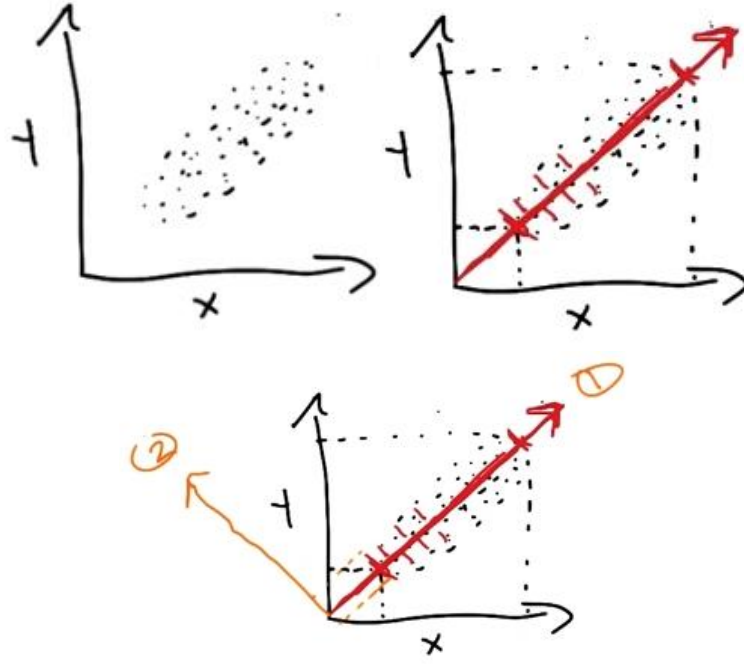
$$-X_1 + 3X_2 = 0 \Rightarrow X_2 = 1, X_1 = 3$$

Böylece 8 olan eigen değerine karşılık gelen eigen vektörü şöyle olur,

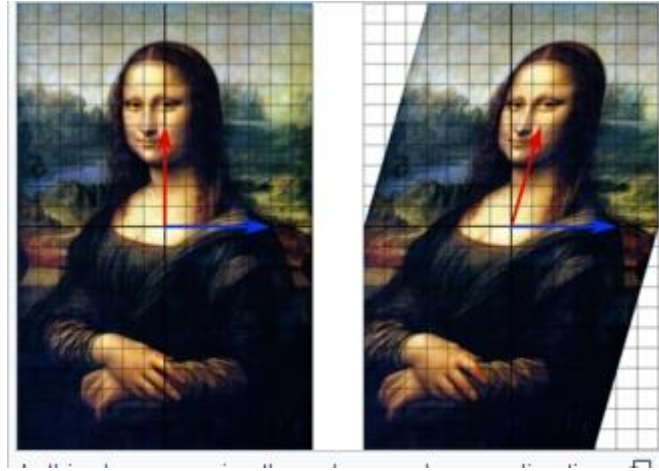
$$\begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Ana bileşenler analizinden kısa bir şekilde inceleyecek olursak; Bu analiz doğrusal dönüşüm algoritmalarında ilk ele alınan analizlerden biridir. Ana bileşenler analizi için elimizde iki boyutta inceleyeceğimiz bir veri setinin olduğunu varsayalım. Ana bileşenler analizi ilk olarak bu verilerin varyansının maksimum olduğu doğrultuyu ve aralığı belirler.

Şekil 18’de iki boyutta incelenen örnek veri setini (siyah noktalar), maksimum varyansın bulunduğu aralığın ve doğrultunun bulunmasını ve bu doğrultuya dik olan bir doğrultunun çizilmesini görebilirsiniz.



Şekil 18:İki boyutta ana bileşenler analizinin incelenmesi (Georgia)



Şekil 19:Ana Bileşenler Analizi sembolik gösterimi (Wikipedia-Eigenvalues and eigenvectors)

Şekil 19’da Ana bileşenler analizinin 2 boyutta uygulanması basit şekilde gösterilmiştir.Burada mavi renkli vektör yönü değişmediğinden,değeri 1 olan eigen vektörüdür.

Ana bileşenler analizi ile veri seti çok az bir hata oranı ile 2. bir düzleme aktarılıp tekrar incelenebilir.İlk veri setimizi bu analizi kullanarak ,2 numaralı düzlemde(1 ve 2 numaralı düzlemler birbirlerine diktir) yeniden yapılandırabiliriz.Anna bileşenler analizi temel olarak bize bu imkanları sağlayan,makine öğrenmesi alanının önemli konularındandır.

OpenCV ile Yüz Tanıma

Program C++ ile yazılmış olup konsol ekranı üzerinden kontrol edilmektedir. Tanımlama kısmında gerekli opencv ve c++ kütüphaneleri deklare edilmiştir. Ayrıca insan yüzünü algılama için gerekli olan istatistiksel veri dosyasının yolu da burada tanımlanmıştır.

```
string haar_cascade_dosyasi =  
"C:\\opencv\\sources\\data\\haarcascades\\haarcascade_frontalface_default.xml";
```

OpenCV (Open Computer Vision) kütüphanesinin en temel nesne tipi 'Mat' ile tanımlanır. Mat nesneleri oluşturarak kullanacağımız görüntüleri 2 boyutlu bir matris şeklinde tutabiliriz. Bu programda veri setlerimizin olduğu veritabanımız bir dosya klasörüdür. Bu dosya klasöründeki görüntü verilerinin dosya yol adresleri bir text tabanlı dosyada tutulup (ext uzantılı) program içerisinde dosya yolları kontrol edilmektedir. Bu ext uzantılı dosya içerisinde 2 kişiye ait 10ar adet örnek görüntünün dosya yolu bulunmaktadır ve birinci kişiye '0', ikinci kişiye '1' etiketleri yine bu dosya içerisinde verilmiştir. Bu dosyanın program içindeki kontrolü csvDosya_oku fonksiyonu ile yapılmaktadır.

```
static void csvDosya_oku(const string& dosyaadi, vector<Mat>& goruntuler, vector<int>&  
etiketler, char ayirici = ';'){  
    std::ifstream dosya(dosyaadi.c_str(), ifstream::in);  
    if (!dosya){  
        string hata_mesaji = "Dosya bulunamadi";  
        CV_Error(CV_StsBadArg, hata_mesaji);  
    }  
    string satir, dosya_yolu, dosya_etiketi;  
    while (getline(dosya, satir)){  
        stringstream satirlar(satir);  
        getline(satirlar, dosya_yolu, ayirici);  
        getline(satirlar, dosya_etiketi);  
        if (!dosya_yolu.empty() && !dosya_etiketi.empty()){  
            goruntuler.push_back(imread(dosya_yolu, 0));  
            etiketler.push_back(atoi(dosya_etiketi.c_str()));  
        }  
    }  
}
```

```
}
```

csvDosya_oku fonksiyonu daha sonra main() fonksiyonu içerisinde kullanılacaktır. Bu sayede veritabanındaki goruntuler ve etiketler, Mat (görüntüler matris şeklinde bellekte tutulacak) ve int (etiketler 0 ve 1 şeklinde tamsayı olarak bellekte tutulacak) türünde vector<> nesneleri şeklinde kullanılarak programın çalışmasına dahil edilecektir.

Daha önce belirttiğim gibi veritabanını programı çalıştırmada önce de oluşturabiliriz. Ancak programın çalışması sırasında sıfırdan bir veritabanı oluşturmak istersek, bunun için YuzuAl() fonksiyonunu kullanabiliriz. Bu fonksiyon yüzü algıladıktan sonra görüntüsünü ara belleğe kaydeder, daha sonra belirli oranda ana görüntüden yüzün olduğu alanı keser ve bu kestiği görüntüye gri tonlama, histogram eşitlemesi ve bilateral filtre uygulayarak veritabanına kaydeder. Ayrıca bu fonksiyon o an kamera karşısındaki en büyük yüzü algılayarak kameraya en yakın olan kişiyi dikkate almaktadır. Bu sayede arka plandaki insanlar veritabanına kayıta dikkate alınmaz.

```
void YuzuAl(Mat goruntu){
    string pencere_ismi = "Elde edilen yüz";
    std::vector<Rect> yuz_alani;
    Mat gri_cerceve;
    Mat kirp;
    Mat res;
    Mat gri_yuz;
    string text;
    stringstream sstm;

    cvtColor(goruntu, gri_cerceve, COLOR_BGR2GRAY);
    equalizeHist(gri_cerceve, gri_cerceve);
    bilateralFilter(gri_cerceve, gri_cerceve, 5 , 80, 80);

    //Yuzu algıla
    yuzAlgila.detectMultiScale(gri_cerceve, yuz_alani, 1.1, 2, 0 |
    CASCADE_SCALE_IMAGE, Size(30, 30));

    // Bize lazım olan alan
    cv::Rect roi_b;
    cv::Rect roi_c;

    size_t ic = 0; // o anki nesnenin index nosu
    int ac = 0; // o anki nesnenin alanı

    size_t ib = 0; // en büyük nesnenin index nosu
    int ab = 0; // en büyük nesnenin alanı

    for (ic = 0; ic < yuz_alani.size(); ic++) // Algılanan yuzlere uygula
    {
        roi_c.x = yuz_alani[ic].x;
        roi_c.y = yuz_alani[ic].y;
        roi_c.width = (yuz_alani[ic].width);
        roi_c.height = (yuz_alani[ic].height);

        ac = roi_c.width * roi_c.height; // Algılanan yuzun alanı

        roi_b.x = yuz_alani[ib].x;
        roi_b.y = yuz_alani[ib].y;
        roi_b.width = (yuz_alani[ib].width);
        roi_b.height = (yuz_alani[ib].height);

        ab = roi_b.width * roi_b.height; //O anki en büyük objenin alanı
```

```

if (ac > ab)
{
    ib = ic;
    roi_b.x = yuz_alani[ib].x;
    roi_b.y = yuz_alani[ib].y;
    roi_b.width = (yuz_alani[ib].width);
    roi_b.height = (yuz_alani[ib].height);
}

kirp = goruntu(roi_b);
resize(kirp, res, Size(128, 128), 0, 0, INTER_LINEAR);
cvtColor(kirp, gri_yuz, CV_BGR2GRAY); //kırpılmış resim siyah beyaza
çevriliyor

dosyaadi = "";
stringstream ssfn;
ssfn << dosyanumarasi << ".jpg";
dosyaadi = ssfn.str();
dosyanumarasi++;

imwrite("../kisi1gri/" + dosyaadi, gri_yuz);

Point pt1(yuz_alani[ic].x, yuz_alani[ic].y); // Ana ekranda algılanan
yuzleri goster
Point pt2((yuz_alani[ic].x + yuz_alani[ic].height), (yuz_alani[ic].y +
yuz_alani[ic].width));
rectangle(goruntu, pt1, pt2, Scalar(0, 255, 0), 2, 8, 0);
}

// resmi goster
sstm << "Kırpılan Alan Boyutu: " << roi_b.width << "x" << roi_b.height << "
Dosya Adi: " << dosyaadi;
text = sstm.str();

putText(goruntu, text, cvPoint(30, 30), FONT_HERSHEY_COMPLEX_SMALL, 0.8,
cvScalar(0, 0, 255), 1, CV_AA);
imshow("ORIJINAL", goruntu);

if (!kirp.empty())
{
    imshow("ALGILANDI", kirp);
}
else
    destroyWindow("ALGILANDI");
}

```

Ana fonksiyonda ise FaceRecognizer işaretçisi oluşturularak yuzTanima isminde EigenFaceRecognizer nesnesi oluşturulmaktadır. Bu nesne createEigenFaceRecognizer isimli opencv fonksiyonuyla oluşturulmaktadır. Bu fonksiyonun içine aldığı parametreler yüzünü algılamak istediğimiz veritabanına kayıtlı kişi sayısı ve yüzü algılamada kullanacağı eşik değeridir. Denemelerim sonucunda, c++ içinde var olan DBL_MAX sabit değerini eşik değeri olarak kullandım. Bu sayede program ayrıntıları dikkate alarak yüzleri en verimli şekilde tahmin etmeye çalışıyor. Eşik değerini DBL_MIN olarak değiştirdiğimde gerçek zamanlı görüntüdeki

tüm yüzlere “Bilinmiyor” etiketi verildiğini gözlemledim. Bu yüzden kullanacağımız eşik değeri bizim için önemli.

Main() fonksiyonu içerisinde ana işlemi gerçekleştiren satırlar aşağıda verilmiştir.

```
int goruntu_genislik = görüntuler[0].cols;
int goruntu_yukseklik = görüntuler[0].rows;
```

Yukarıdaki kod satırlarında veritabanındaki ilk görüntünün satır ve sütunları kullanılarak boyutları hesaplanıyor. Çünkü yüzü boyutlandırmamız bizim için önemli. main() fonksiyonu içindeki resize fonksiyonu ile anlık görüntü arka planda yeniden boyutlandırılıyor.

```
string kisi_listesi[] =
{
    "AHMET",
    "ORKUT"
};

while (1){
    video >> canli_goruntu;
    Mat orjinal = canli_goruntu.clone(); // orjinal görüntüyü tuttuk
    Mat gri;
    cvtColor(orjinal, gri, CV_BGR2GRAY); //görüntüyü siyah beyaz yaptık

    //önce görüntüdeki yüzleri algılayalım
    vector< Rect_<int> > yuzler;
    yuzAlgila.detectMultiScale(gri, yuzler);

    //alt satırlarda elde algılanan yüzlerden tahmin yapılıyor
    for (int i = 0; i < yuzler.size(); i++){

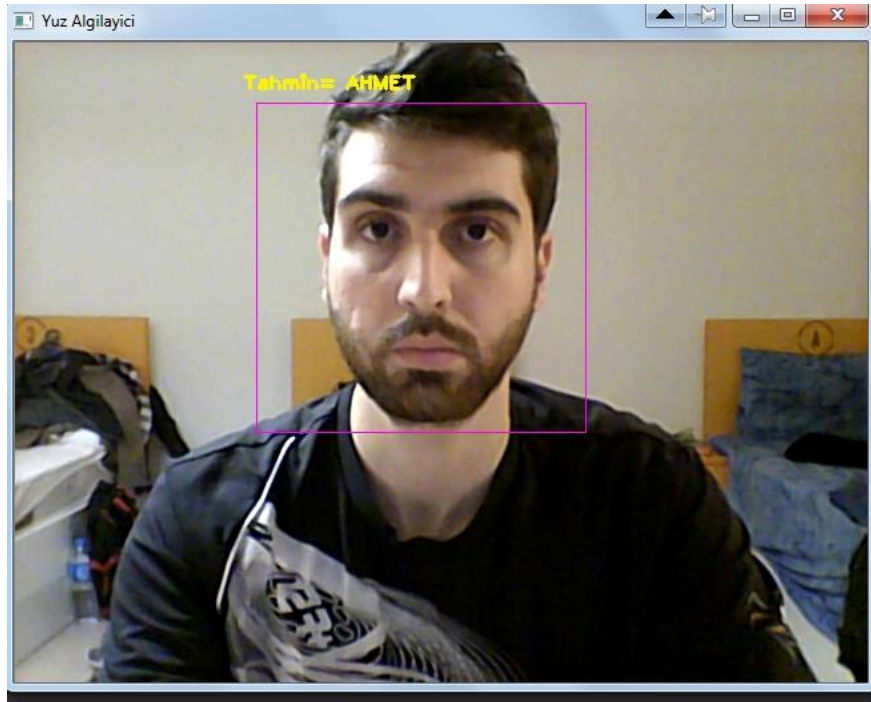
        Rect yuz_i = yuzler[i];
        Mat yuz = gri(yuz_i);

        Mat yuz_resize;
        resize(yuz, yuz_resize, Size(goruntu_genislik, goruntu_yukseklik),
1.0, 1.0, INTER_CUBIC);
        equalizeHist(yuz_resize, yuz_resize);
        int tahmin = yuztanima->predict(yuz_resize);
        rectangle(orjinal, yuz_i, CV_RGB(252, 1, 245), 1);
        string text_box;
        text_box = format("Tahmin= ");
        if ( tahmin >= 0 && tahmin <= 1 ){

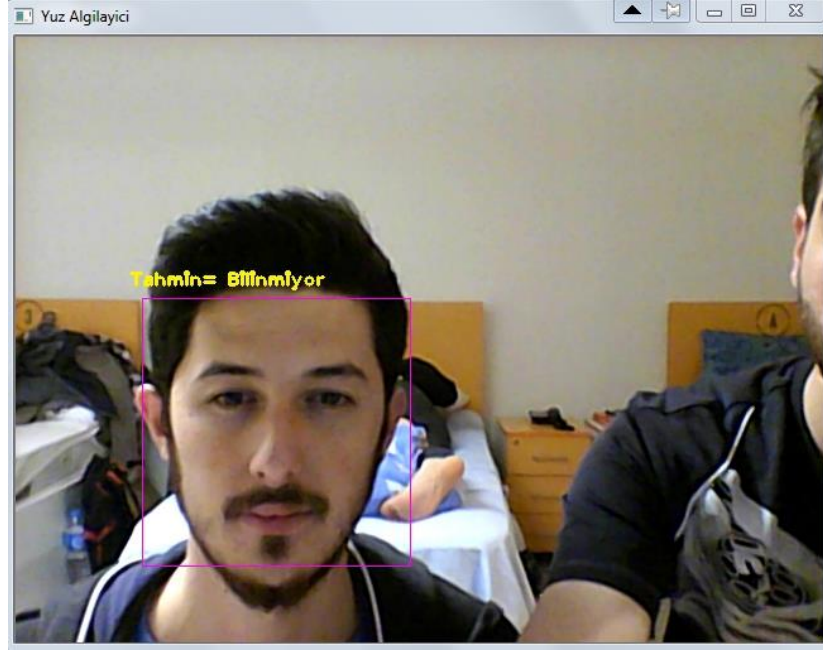
            text_box.append(kisi_listesi[tahmin]);
        }
        else{
            text_box.append("Bilinmiyor");
        }
    }
}
```

Rectangle() fonksiyonu ile algılanan yüz çerçeve içine alınıyor.EigenFaceRecognizer nesnemiz olan yuztanima nesnesi predict yordamı ile yeniden boyutlandırılmış görüntü üzerinden bir tahmin üretiyor.Daha sonra bu tahminin 0 ya 1 olmasına göre veritabanından ilgili etiket çekilip canlı görüntü üzerinde gösteriliyor.

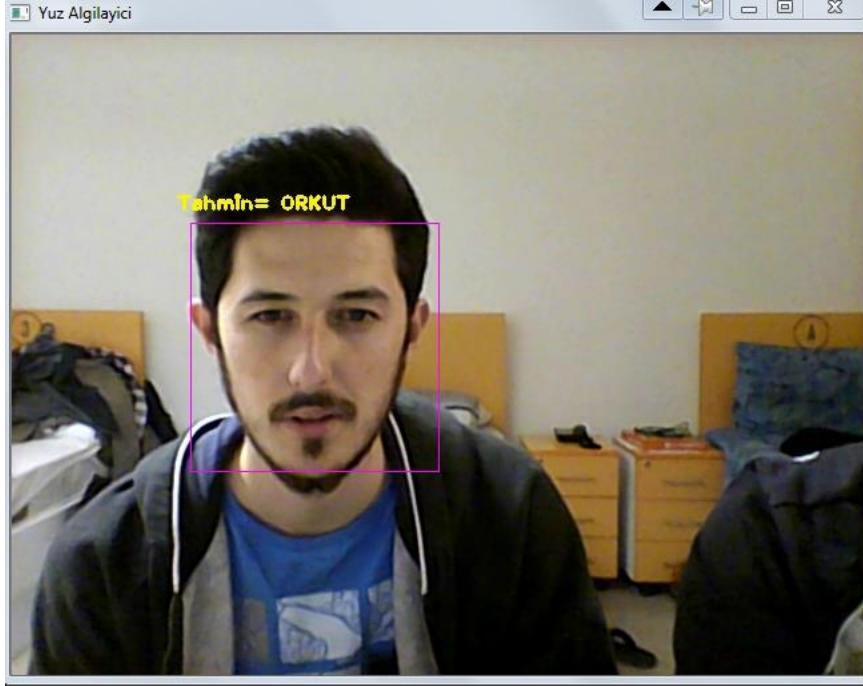
Programın ilk yapım aşamalarında veritabanında olmayan bir kişi için bilinmiyor uyarısı vermiyordu.Bunu çözmek için dünyaca ünlü bir platform olan stackoverflow'da sorunumu anlattım ve bana ikili sınıflandırma(binary classification) yaptığımdan dolayı mutlak bir şekilde döndürülen değerin 0 ya da 1 olacağı(veritabanındaki 2 kişiden biri) söylendi.Bu yüzden veritabanına bilinmeyen kişi olarak boş bir veri seti eklemek durumunda kaldım.



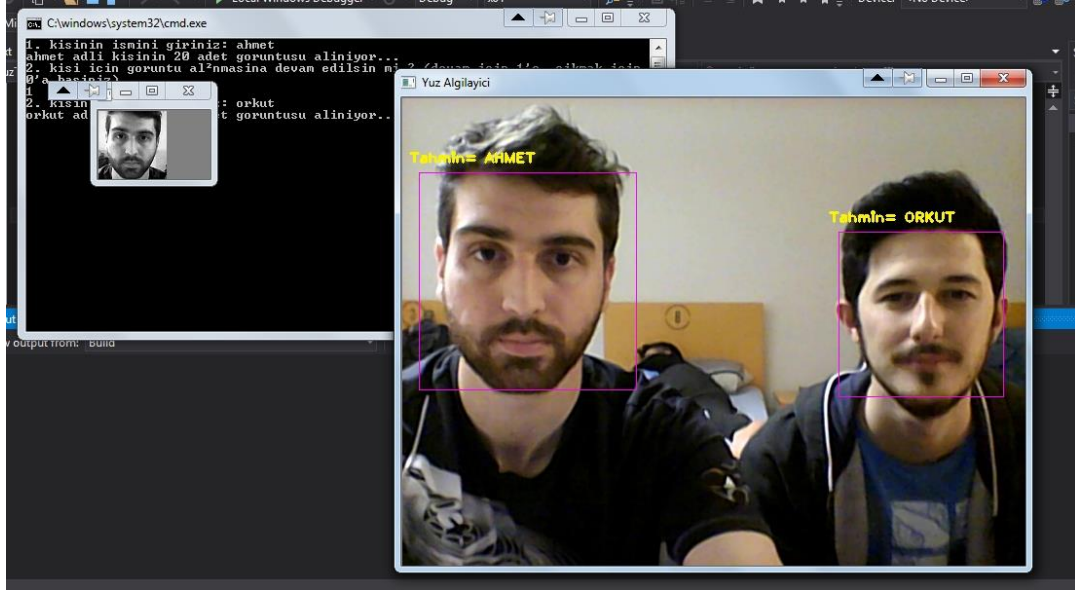
Şekil 20:Program çalıştırıldığında kamera karşısına ilk olarak kendim (Ahmet Yaylaloğlu) geçip programı test ettim.



Şekil 21:Daha sonra, oda arkadaşım Orkut Göküş 'ü veri tabanına kaydetmeden kamera karşısına geçmesini rica edip programı test ettim.



Şekil 22:Bu işlemten sonra veritabanına Orkut'a ait olan 10 adet veri setini ekleyip,tekrar Orkut'un kamera karşısına geçmesini rica ettim.



Şekil 23: Son test aşaması olarak da Orkut ile aynı anda kamera karşısına geçip programı çalıştırdık.

Programın çalışması hakkındaki gözlemlerimi aktaracak olursam;

C++ ile yazdığım bu uygulama gerçek zamanlı olduğundan dolayı sistemi çok aşırı şekilde yordu. Bu yüzden programın kamerayı açmasından yaklaşık 10 saniye sonra kameradan alınan görüntüler, sistemin yorulmasından dolayı geç işlenmeye başladı ve öğrenme bir süre sonra yanlış sonuçlar vermeye başladı. Bu yüzden sistem kaynaklarının çok fazla kullanılmadığı ilk 10-15 saniye içerisinde program doğru tahminler yaparken, daha sonra kısmen yanlış tahminler yapmaya başladı. Bu yüzden .NET platformu üzerinden aynı algoritmaları kullanarak benzer bir uygulama için çalışmaktayım. OpenCV kütüphanesinin .NET platformuna uyarlanmış versiyonu olan EmguCV kütüphanesi ile çok verimli sonuçlar aldım. Ayrıca bu görüntü işleme sistemini robotik ya da web projelerine aktarmak için .NET platformu daha uygun olacaktır.

Şekil 1:insan yüzünü temel anlamda temsil edecek parametreler -----	3
Şekil 2:insan yüzünü temsil eden parametrelerin tek görüntüde birleştirilmesi -----	4
Şekil 3:Yüz parametrelerinin ikili görüntü üzerine uygulanışı-----	4
Şekil 4:Yüz tanıma algoritmasının görsel olarak uygulanışı(bu görüntüleri alıntı yaptığım Adam Harvey'in videosunu izlemenizi öneririm; https://vimeo.com/12774628)-----	5
Şekil 5:OpenCV istatistiksel veri dosyaları kullanarak,görüntüdeki yüz,göz ve ağzın tespiti-	6
Şekil 6:Görüntünün göz konumlarına göre belirli ölçüde kırılması ve gri tonlama uygulanması -----	7
Şekil 7:ilk grafik görüntünün normal renk yoğunluk değerleri,ikinci grafik histogram eşitliği tekniğinin uygulanmasından sonraki yoğunluk değerleri-----	8
Şekil 8:8 bitlik gri tonlamalı pikseller ve görüntünün matris (0-255 deger arası) gösterimi ---	9
Şekil 9:piksel değerleri varyans tablosu-----	9
Şekil 10:Hesaplanan cdf değerlerini ve $h(v)$ formülüne göre hesaplanan histogram eşitlik değerlerini-----	10
Şekil 11:Yeni oluşan görüntü matrisi ve orijinal görüntü ile yeni görüntünün karşılaştırılması -----	10
Şekil 12:OpenCV ile histogram eşitliği uygulanması ve bunlara ait histogram grafikleri ----	11
Şekil 13:Histogram eşitliği uygulanan görüntüye Bilateral filtre uygulanmasının sonucu ----	13
Şekil 14:Kullanılan eğitim veri seti -----	14
Şekil 15: Ana bileşenler analizi ile eigen vektörlerinin elde edilmesi matlab kodu -----	15
Şekil 16:Ana bileşenler analizinden sonra veri eğitim setinin son hali -----	15
Şekil 17:Öğrenmenin gerçekleştirilmesi -----	16
Şekil 18:İki boyutta ana bileşenler analizinin incelenmesi -----	19
Şekil 19:Ana Bileşenler Analizi sembolik gösterimi -----	19
Şekil 20:Program çalıştırıldığında kamera karşısına ilk olarak kendim (Ahmet Yaylalıoğlu) geçip programı test ettim. -----	24
Şekil 21:Daha sonra, oda arkadaşım Orkut Göküş 'ü veri tabanına kaydetmeden kamera karşısına geçmesini rica edip programı test ettim. -----	25
Şekil 22:Bu işlemten sonra veritabanına Orkut'a ait olan 10 adet veri setini ekleyip,tekrar Orkut'un kamera karşısına geçmesini rica ettim. -----	25
Şekil 23:Son test aşaması olarak da Orkut ile aynı anda kamera karşısına geçip programı çalıştırdık. -----	26

Kaynakça

Arubas, E. *Face Detection and Recognition*. <http://eyalarubas.com/face-detection-and-recognition.html> adresinden alındı

Bilateral Filter. Wikipedia: https://en.wikipedia.org/wiki/Bilateral_filter adresinden alındı

Daniel Lélis Baggio, S. E. (2012). *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing.

Eigenvalues and eigenvectors. Wikipedia:
https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors#Principal_components_analysis adresinden alındı

Histogram equalization. (tarih yok). Wikipedia: https://en.wikipedia.org/wiki/Histogram_equalization adresinden alındı

Suarez, O. D. (2014). *OpenCV Essentials*. Packt Publishing.