T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÖZYÜZ KULLANILARAK YÜZ TANIMA

YÜKSEK LİSANS TEZİ

Bilg.Müh. Bahattin YAMAN

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜH

Tez Danışmanı : Yrd. Doç. Dr. Kürşat AYAN

Temmuz 2006

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÖZYÜZ KULLANILARAK YÜZ TANIMA

YÜKSEK LİSANS TEZİ

Bilg.Müh. Bahattin YAMAN

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜH

Bu tez 06 / 07 /2006 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Jüri Başkanı	Üye	Üye
Yrd. Doç. Dr.	Prof. Dr.	Yrd. Doç. Dr.
Kürşat AYAN	Uğur ARİFOĞLU	Cemil ÖZ

TEŞEKKÜR

Tezin hazırlanması aşamasında bana her türlü desteği veren danışman hocam sayın Yrd. Doç. Dr. Kürşat AYAN beye teşekkürü bir borç bilirim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

RCER	: Kaba kontur tahmin rutini
E	: Enerji fonksiyonu
N_{siyah}	: Siyah noktaların sayısı
N_{beyaz}	: Beyaz noktaların sayısı
Γ	: Yüz görüntüsü eğitim seti
ψ	: Yüz görüntüsü eğitim seti ortalaması
Φ	: Yüz eğitim seti ortalama farkı

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Yüz Tanıma, Özyüz, Görüntü Tanıma Sistemleri, Görüntü Sınıfları

Yüz tanıma, günümüzde önemini hızla arttırmakta olan bir uygulama konusu haline gelmektedir. Mevcut yüzlerden elde edilecek bir veri tabanı üzerinde otomatik olarak yapılacak tanımaların, suçluların teşhisi sırasında, emniyet görevlilerinin işini ne kadar kolaylaştıracağı açıktır.

Bu tezde, yüz tanıma problemi üzerinde araştırma yapılarak, kullanılan temel yöntemler incelendi. Yüz tanımada kullanılan yöntemlerden biri olan, özyüz (eigenface) yöntemi ile bir yüz tanıma sistemi geliştirildi.

Bu tez aşağıdaki şekilde organize edildi: Birinci bölüm de yüz tanıma sisteminin getireceği kolaylıklar ele alındı. Ayrıca özyüz yönteminden kısaca bahsedildi. İkinci bölüm görüntü ve yüz tanımanın temel kavramları ele alındı. Bu bölümde yüz tanıma problemi için yapılmış olan çalışmalardan ve uygulanan metodlardan bahsedildi.

Yüz tanıma problemi için iki ana yaklaşım verildi. Üçüncü bölüm önerilen yüz tanıma sistemini olan özyüz tanıma metodunun detayları ele alındı. Bu bölümde özyüzlerin hesaplanması için gerekli olan tüm işlemler anlatıldı.

Dördüncü bölüm, özyüzler yaklaşımını göstermek için geliştirilmiş olan yüz tanıma yazılımına ayrıldı. Yüz tanıma programı Visual C# ve Visual Basic ortak kullanılarak geliştirildi. Dördüncü bölümde geliştirilen uygulamanın nasıl kurulacağı ve kullanılacağı verildi. Aynı zamanda uygulamada kullanılan önemli bir kaç rutin de verildi.

FACE RECOGNITION USING EIGENFACES

SUMMARY

Keywords: Face recognition, Eigenface, Pattern recognition systems, Pattern class

In this thesis, a research was done to find out the different approaches to the face recognition problem. It has been observed that these different approaches fall into two major categories that are given below:

Feature based recognition, which is based on the extraction of the properties of individual organs located on a face such as eyes, nose and mouth, as well as their relationships with each other. Feature vectors describing the characteristics of face images are evaluated by using deformable templates and active contour models, where excessive geometry and the minimization of energy functions are involved.

Principal component analysis, based on information theory concepts, seek a computational model that best describes a face, by extracting the most relevant information contained in that face. Goal is to find out the eigenvectors (eigenfaces) of the covariance matrix of the distribution, spanned by a training set of face images. Later, every face image is represented by a linear combination of these eigenvectors. Evaluation of these eigenvectors are quite difficult for typical image sizes but, an approximation that is suitable for practical purposes is also presented. Recognition is performed by projecting a new image into the subspace spanned by the eigenfaces and then classifying the face by comparing its position in face space with the positions of known individuals.

A face recognition system, based on the eigenfaces approach is proposed. Eigenfaces approach seems to be an adequate method to be used in face recognition due to its simplicity, speed and learning capability.

BÖLÜM 1. GİRİŞ

Yüz, sosyal ilişkilerde önde gelen dikkat noktamız olup, kimlik ve duyguların aktarılmasında önemli bir rol oynar. Yüz görünümümüzden zeka veya karakterin anlaşılabilmesi şüpheli olmakla birlikte, insanın yüzleri tanıma yeteneği olağanüstüdür. Hayatımız boyunca öğrendiğimiz binlerce yüzü tanıyabilir ve hatta, yıllarca ayrı kaldıktan sonra bile tanıdık yüzleri ayırt edebiliriz. Bu tanıma becerisi, izleme koşulları, ifade, yaşlanma ve gözlük, sakal veya saç stilinde değişiklikler gibi dikkat dağıtan unsurlar nedeniyle görsel uyarıcıdaki büyük değişikliklere rağmen oldukça sağlamdır.

Yüz tanıma olayı, güvenlik sistemleri, kredi kartı doğrulama ve kriminal kimlik tanımlama gibi pek çok uygulamada önemli bir konu haline gelmiştir. Örneğin, belirli bir yüzü modelleme ve onu çok sayıdaki kaydedilmiş yüz modellerinden ayırt etme becerisi, kriminal kimlik tanımlamasının büyük ölçüde iyileştirilmesini mümkün kılacaktır. Hatta yüzleri tanımak değil de, algılama becerisi bile önemli olabilir. Pek çok zenginleştirme ve gürültü azaltma tekniği efekti resmin içeriğine bağlı olduğundan dolayı, renkli film banyosu işleminin otomasyonu için, fotoğraflardaki yüzleri algılama çok faydalı olabilir

Yüzlerin insan beyni tarafından nasıl kodlandığı veya şifresinin çözüldüğü belirgin olmamasına rağmen insanların yüz tanımada iyi olduğu açıktır. İnsan yüzü tanıma konusuna yirmi yılı aşkın bir süredir çalışılmaktadır. Yüzler karmaşık, çok boyutlu görsel uyarılar olduğundan dolayı yüz tanımada bilişimsel bir model geliştirmek oldukça zordur. Bu nedenle yüz tanıma işlemi, son derece yüksek düzeyde bir bilgisayar görmesi görevidir ve bunun içinde pek çok ilk görüş tekniği yer alabilir. İnsan yüzü tanımlamada ilk adım, yüz görüntülerinden ilgili özellikler çıkarmaktır. Bu alandaki araştırmalar öncelikle başka bir insanın bir yüzü doğru bir şekilde tanıması için yeterli olan bilgiyi üretmeyi amaçlar. Soru, doğal olarak yüz

özelliklerinin ne kadar iyi şekilde ölçümlenebileceği konusundadır. Eğer bu tür bir ölçümleme mümkünse bu durumda bir bilgisayar, özellikler seti verilmiş bir yüzü tanıyabilmelidir. Çok sayıda araştırmacının [1, 2, 3] geçmiş yıllarda yapmış oldukları incelemeler, insanlar tarafından yüzleri tanımak için belirli yüz özelliklerinin kullanıldığını göstermiştir.

Yüz tanıma problemine üç farklı yaklaşım öneren üç ana araştırma grubu vardır. En büyük grup [4, 5, 6], bireysel yüzlerin tanınmasında insanlar tarafından kullanılan yüz özellikleri ile ilgilenmişlerdir. İkinci grup [7, 8, 9, 10, 11], profil siluetlerden alınan öznitelik vektörlerine dayalı insan yüzü tanıması gerçekleştirir. Üçüncü grup ise [12, 13] yüzün ön görünümünden alınan öznitelik vektörleri kullanır. Yüz tanıma problemine üç farklı yaklaşım olsa da, bu üç farklı yaklaşımın çıktığı iki temel metot vardır.

Birinci metot bilgi teorisi kavramına, başka bir deyişle, ana bileşen analiz metotlarına dayalıdır. Bu yaklaşımda, bir yüzü en iyi tanımlayan en yakın bilgi tüm yüz görüntüsünden elde edilir. Görüntü tanımadaki Karhunen-Loeve büyümesine dayalı olarak, M Kirby ve L. Sirovich, herhangi bir yüzün "özyüzler" adını verdikleri en iyi koordinat sistemi açısından ekonomik olarak gösterilebileceğini göstermiştir [4, 5]. Bunlar yüz takımlarının ortalama koveryansının öz fonksiyonlarıdır. Daha sonra M. Turk ve A. Pentland, özyüzler yaklaşımına dayalı bir yüz tanıma metodu [14] önerdiler.

İkinci metot bir yüzün gözler, burun, ağız, ve çene gibi temel kısımlarından öz nitelik vektörleri çıkarmaya dayalıdır. Bu yöntemde biçim değiştirebilen şablonlar ve geniş matematik yardımıyla bir yüzün temel kısımlarından önemli bilgiler toplanır ve sonra bir öz nitelik vektörüne dönüştürülür. L. Yullie ve S. Cohen [15], biçim değiştirebilen şablonların yüz görüntülerine ait şekillerin çıkarılmasında büyük bir rol oynamıştır.

1.1. Tanıma İçin Özyüzler

Bu çalışma şekil değiştirebilen şablonlar gibi, aşırı geometriye ve hesaplamalara bağlı olmayan, bir tür denetimsiz görüntü tanıma programı geliştirme konusun da

yapılmıştır. Özyüzler yaklaşımı basitliği, hızı ve öğrenme becerisi nedeniyle yüz tanımasında kullanılacak en uygun metot olarak görülmüştür.

Özyüzlere dayalı önceki bir çalışma M. Turk ve A. Pentland tarafından yapılmıştır. Burada yüzler ilk başta algılanmış ve sonra tanımlanmıştır. Bu tezde M. Turk ve A. Pentland tarafından sunulana benzer bir, özyüzler yaklaşımına dayalı yüz tanıma sistemi önerilir.

Program, yüz görüntülerini, özyüzler adı verilen küçük bir karakteristik öz nitelik görüntülerine ayrıştıran bir bilgi teorisi yaklaşımına dayalıdır ki; bu başlangıçtaki yüz görüntüleri eğitim kümesinin ana bileşenleri olarak düşünülebilir. Tanıma, yeni bir görüntüyü, özyüzler tarafından yayılan alt uzaya yansıtarak ve sonra yüzdeki konumunu, bilinen bireylerin konumları ile karşılaştırıp sınıflandırarak yapılır.

Gerçek sistem hem bilinen bireyleri tanıyabilir ve yeni yüzleri tanımayı öğrenebilir. Bu programda kullanılan özyüz yaklaşımı hız, basitlik, öğrenme becerisi ve yüz görüntüsündeki küçük değişikliklere dayanıklılıkta diğer yüz tanıma metotlarına karşı avantajlara sahiptir.

Özyüzler yaklaşımına dayalı gerçek bir yüz tanıma yazılımı kişisel bir bilgisayarda C# ve Visual Basic programlama dili ile geliştirildi.

BÖLÜM 2. GÖRÜNTÜ VE YÜZ TANIMANIN TEMEL KAVRAMLARI

Bu bölüm görüntü ve yüz tanımanın temel prensiplerine ayrılmıştır. İki ana yüz tanıma yaklaşımları ile birlikte, yüz tanıma tarihi hakkında kısa bir özet, konu hakkında daha fazla bilgi amacıyla sunulur.

2.1. Görüntü Tanımanın Temel Kavramları

2.1.1. Genel bakış

Karar vermede bilgi önemli bir unsur olduğundan, gelişmiş bilgi sistemlerine olan ihtiyaç daha belirgin hale gelmiştir. Modern bilgi sistemlerindeki tasarımın ana problemlerinden biri, otomatik görüntü tanımadır.

Tanıma, insanların ve diğer canlı organizmaların temel bir niteliği olarak kabul edilir. Görüntü, bir nesnenin tarifidir. Kısmen, üstün bir görüntü tanıma becerisine sahip olması nedeniyle, insan oldukça sofistike bir bilgi sistemidir. Tanınacak görüntülerin özelliğine göre tanıma eylemi, iki ana görüntü türüne bölünebilir [16]:

Somut maddelerin tanınması: Buna duyusal tanıma denebilir ve görsel ve işitsel görüntü tanımayı içerir. Bu tanıma işlemi, tanımlama ve uzaysal sınıflandırma ve geçici görüntüleri içerir. Uzaysal görüntülere örnek karakterler, parmak izleri, fiziksel nesneler ve görüntülerdir. Geçici görüntüler arasında, konuşma dalga formları, zaman serileri, elektro kardiyogramlar ve hedef imzalar bulunur.

Soyut maddelerin tanınması: Öte yandan eski bir argüman veya çözüm için problem tanınabilir. Bu süreç soyut maddelerin tanınmasını içerir ve kavramsal tanıma olarak adlandırılabilir.

Somut görüntülerin insanlar tarafından tanınması, bir kişi ile bir fiziksel bir uyarı arasında ilişkiyi bir içeren bir psikofizyolojik problem olarak düşünülebilir. Gerçekte insan tanıması, giriş verilerinin geçmiş deneyimimize dayanan ve ip uçları ile tanıma için öncelik bilgisine bağlı olan, bilinen bir istatistiki insan seti ile ilişkilendirilebilecek olan nispi olasılıkları tahmin etme problemidir. Bu nedenle görüntü tanıma problemi, populasyonlar arasındaki giriş verileri ni, insan üyeleri arasında özellikler veya değişmez niteliklerin araştırılması yoluyla ayırt etme problemi olarak değerlendirilebilir.

2.1.2. Görüntü sınıfları ve görüntüler

Görüntü tanıma, ilgisiz ayrıntılar arka planından önemli özelliklerin veya niteliklerin çıkarılması yoluyla, giriş verilerinin tanımlanabilir sınıflara kategorize edilmesi olarak tanımlanabilir.

Bir görüntü sınıfı, bazı verilen ortak nitelikler veya özellikler tarafından belirlenen bir kategoridir. Bir görüntü sınıfının özellikleri, bu sınıfa ait tüm görüntüler için ortak olan karakterize edici niteliklerdir. Bu özelliklere genellikle, set içi özellikler denir. Görüntü sınıfları arasındaki farkları temsil eden özelliklere de set arası özellikler denebilir.

Bir görüntü, bir görüntü sınıfını temsil eden bir kategorinin herhangi bir üyesinin tarifidir. Kolaylık açısından görüntüler genellikle aşağıdaki gibi bir vektör tarafından temsil edilir.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ \dots \\ x_n \end{bmatrix} \tag{2.1}$$

Burada her x_j elemanı, bu görüntünün bir özelliğini temsil eder. Genellikle bir görüntü vektörünü, bir n-boyutlu öklid uzayında, genellikle bir nokta olarak düşünmek faydalıdır.

2.1.3. Görüntü Tanıma Sistem Tasarımındaki Temel Problemler

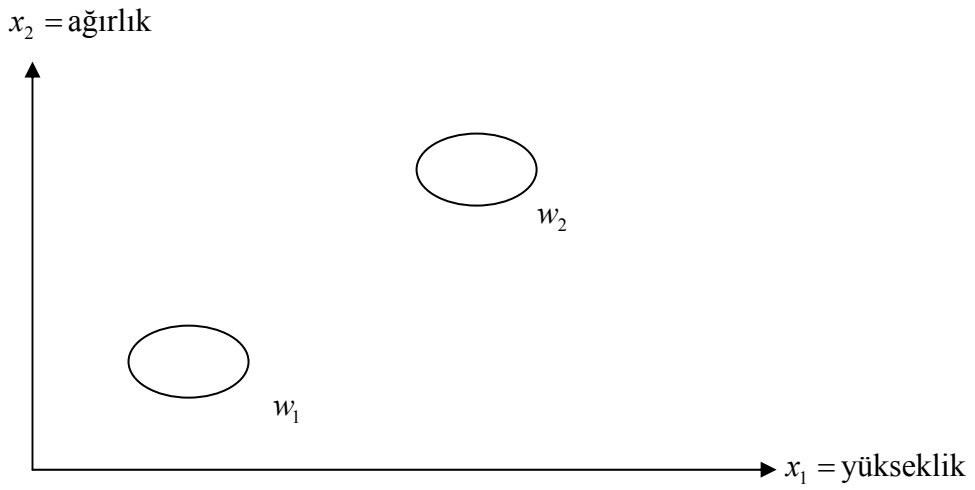
Bir otomatik görüntü tanıma sisteminin tasarımı, genelde birkaç ana problem alanını içerir:

- Öncelikle tanınacak olan nesnelerden, ölçülebilecek giriş verilerinin temsilini ele almalıyız. Bu bir algılama problemidir. Ölçülen her miktar, görüntü veya cismin bir özelliğini tarif eder. Başka bir deyişle, giriş verilerini tarif eden bir görüntü vektörünün oluşturulması gerekir. Görüntü vektörleri, görüntüler hakkında mevcut bulunan tüm ölçülmüş bilgileri içerir. Aynı sınıfa ait görüntüler seti, ölçüm uzayının bir bölgesi içinde dağıtılmış bir noktalar takımına karşılık gelir. Buna basit bir örnek w_1 ve w_2 olarak gösterilen iki görüntü sınıfı için şekil 2.1. de verilmiştir.Yükseklik ve ağırlıkdan dolayı görüntü vektörü $X = \{X_1, X_2\}^T$ şeklindedir.

- Görüntü tanımadaki ikinci problem, ele alınan giriş verilerinden karakteristik özelliklerin veya niteliklerin çıkarılması ve görüntü vektörlerinin boyutsallığının azaltılması ile ilgilidir. Genellikle buna, işlem öncesi ve özellik çıkarma problemi denir. Set içi özellikler, söz konusu tüm görüntü sınıfları için ortak olan unsurları ayırt edici bilgi içermez ve ihmal edilebilir. Eğer ölçülmüş verilerden her görüntü sınıfı için komple bir ayırt edici özellikler takımı belirlenilebilirse, görüntülerin tanınması ve sınıflanması çok az zorluk içerecektir. Otomatik algılama, basit bir eşleşme sürecine veya tablo arama programına indirgenebilir. Ancak pratikte ortaya çıkan pek çok görüntü tanıma problemin de, ayırt edici özelliklerin komple bir takımın belirlenmesi imkansız olmasa da son derece zordur.

- Sistem tasarımında üçüncü problem, tanımlama ve sınıflandırma sürecinde gerekli olan görüntü algılama optimum karar prosedürlerinin belirlenmesini içerir. Gözlemlenen veriler, tanınacak görüntülerden, görüntü uzayında görüntü noktaları

veya ölçüm vektörleri şeklinde ifade edildikten sonra, bu verilerin hangi görüntü sınıfına ait olduğuna dair makinanın karar vermesini isteriz. Sistemin, M farklı görüntü sınıfını tanıyabildiğini düşünelim; ki bu durumda görüntü uzayı her biri bir sınıfın görüntü noktalarını kapsayan M adet bölgeden oluştuğu şeklinde düşünülebilir. Bu durumda tanıma problemi, gözlemlenen ölçüm vektörleri bazında M adet görüntü sınıfını ayıran karar sınırlarını üreten bir problem olarak görülebilir. Bu karar sınırları, genel olarak karar fonksiyonları tarafından belirlenir.



Şekil 2.1. Her görüntünün iki ölçüm ile karakterize edildiği iki ayrılmış görüntü sınıfı

2.1.4. Eğitim ve öğrenim

Karar fonksiyonları çeşitli şekillerde üretilebilir. Tanımlanacak görüntüler hakkında komple bir öncelik bilgisi mevcut olduğunda, karar fonksiyonu bu bilgi ışığında hassasiyetle belirlenebilir. Görüntüler hakkında sadece niceliksel bilgi mevcut olduğunda, karar fonksiyonlarının şekilleri hakkında makul tahminler yapılabilir. Bu durumda karar sınırları doğruluktan uzak olabilir ve bazı ayarlamalarla tahmin edici bir performansa ulaşması için makinenin tasarlanması gerekir.

Daha genel bir durum ise tanınacak görüntüler hakkında çok az öncelik bilgisinin bulunmasıdır. Bu durumlarda görüntü tanıma makineleri en iyi bir eğitim veya öğrenim prosedürü kullanarak tasarlanır. Başlangıçta isteğe bağlı karar fonksiyonları

var sayılır ve tekrarlanan eğitim adımları dizisi ile bu fonksiyonlar optimum veya tatmin edici şekillere yaklaştırılır.

Eğitim veya öğrenimin, sadece bir görüntü tanıma sisteminin tasarım (veya güncelleme) aşamasında gerçekleştiğini akılda tutmak önemlidir. Eğitim görüntü setleri ile kabul edilebilir sonuçlar alındığında, sistem içinde çalışması beklenen ortamdan alınan örnekler üzerinde tanıma işlemini yapma görevine uygulanır. Tanıma performansının kalitesi, büyük ölçüde eğitim görüntülerinin, sistemin normal işletim sırasında karşılaşacağı gerçek verilerle ne kadar yakın bir şekilde ilişkili olması ile belirlenecektir.

2.1.5. Denetimli ve denetimsiz görüntü tanıma

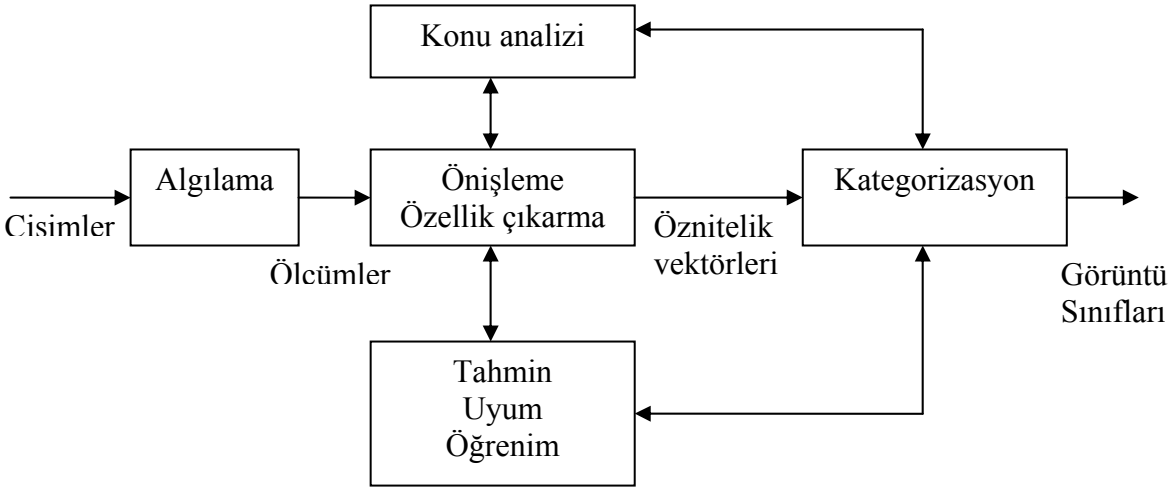
Çoğu durumda, söz konusu her sınıftan örnek görüntüler mevcuttur. Bu durumlarda denetimli görüntü tanıma teknikleri uygulanabilir. Denetimli bir öğrenim ortamında sistemin çeşitli uyumlu programlar aracılığıyla görüntüleri tanıdığı düşünülür. Bu yaklaşımın önemli noktası, bilinen sınıflandırmanın eğitim görüntülerinin kümesi ve uygun bir öğrenim prosedürünün uygulanmasıdır.

Bazı uygulamalarda, sadece bilinmeyen sınıflandırmanın öğrenim görüntülerinden oluşan bir takım mevcut olabilir. Bu durumlarda denetimsiz görüntü tanıma teknikleri uygulanır. Yukarıda bahsedildiği gibi, denetimli görüntü tanıma her eğitim görüntüsünün doğru sınıflandırılmasının bilinmesi ile karakterize edilir. Ancak denetimsiz durumda verilen veriler içinde bulunan görüntü sınıflarını öğrenmek problemi ile karşı karşıya kalınır. Bu problem ayrıca “bir öğretmen olmadan öğrenmek” olarak da bilinir.

2.1.6. Tipik bir görüntü tanıma sisteminin planı

Şekil 2.2. de uyumlu bir görüntü tanıma sisteminin işlevsel blok diyagramı gösterilir. Optimum karar ve önişleme veya özellik çıkarma arasındaki ayrım önemli olmasa

da, işlevsel döküm kavramı görüntü tanıma probleminin anlaşılması için net bir resim sağlar.



Şekil 2.2. Uyumlu bir görüntü tanıma sisteminin işlevsel blok diyagramı

Doğru tanıma, ölçümlerden bulunan ayırt edici bilginin miktarına ve bu bilginin doğru şekilde kullanımına bağlı olacaktır. Bazı uygulamalarda konu bilgisi, doğru tanımaya ulaşmada çok önemlidir. Örneğin, el yazısı karakterlerinin tanınması ve parmak izlerinin sınıflanmasında konu bilgisi son derece önemlidir. Bozulmalara karşı dayanıklı, büyük görüntü sapmaları altında esnek ve kendi kendine ayar yapabilen bir görüntü tanıma sistemi tasarlamak istediğimizde, uyum problemi ile karşı karşıya kalırız.

2.2. Yüz Tanıma

Yüz tanıma, özel olarak yüz üzerinde yapılan bir görüntü tanıma görevidir. Kayıtlı bilinen bireylerle karşılaştırdıktan sonra, bir yüzü “bilinen” veya “ bilinmeyen” olarak sınıflayan bir görev olarak tarif edilebilir. Ayrıca, bilinmeyen yüzleri tanımayı öğrenme kabiliyetine sahip bir sisteme sahip olmak da istenir.

Yüz tanımanın sayısal modelleri, pek çok zor modelleri ele almak zorundadır. Bu zorluk yüzlerin, özel bir yüzü, diğer tüm yüzlerden ayırt etmek için mevcut yüz

bilgisini en iyi kullanacak şekilde temsil edilmesi zorunluluğundan kaynaklanır. Tüm yüzler, gözler, burun, ağız gibi aşağı yukarı aynı tarzda düzenlenmiş, aynı öz niteliklere sahip olmaları noktasında bir birlerine benzedikleri için zor bir problem oluşturur.

2.2.1. Arka plan ve ilgili çalışma

Yüzlerin bilgisayarca tanınmasındaki pek çok çalışma, gözler, burun, ağız ve kafa planı gibi bireysel özellikleri algılama ve konum, büyüklük ve bu özellikler arasındaki ilişkiler ile bir yüzü tanımlama üzerine odaklanmıştır. Bu tür yaklaşımlar, çoklu görünümleri genişletmek için zor olmuştur. Başarılı olması için başlangıçta iyi bir tahmin gerektirirdiğinden dolayı çoğunlukla çok kırılgan olmuşlardır. Yüz tanımak için insan stratejilerindeki araştırmalarda, bireysel özellik ilişkileri yetişkin insan yüzü tanımlaması performansı için yetersiz bir temsil içerdiği görülmüştür. [17]. Yine de yüz tanıması ile ilgili bu yaklaşım bilgisayar görüş literatüründe en popüler olanıdır.

Fotoğraflara elle girilen ölçümsel işaretler bazında yüzleri sınıflandırarak, Bledsoe [18,19] tarafından bir melez insan bilgisayar sistemi ile yarı matematik yüz algılaması denenmiştir. Sınıflandırma için parametreler, göz köşeleri, ağız köşeleri, burun ucu ve çene noktası gibi noktalar arasında normalleştirilmiş mesafeler ve oranlardı. Daha sonra Bell laboratuarındaki çalışmalar da, yaklaşık yirmi özelliğe sahip bir vektör geliştirdi ve standart görüntü sınıflandırma teknikleri kullanarak yüzleri tanıdı.

Fischler ve Elschlager [20], benzer özellikleri otomatik olarak ölçmeye çalıştılar. Yüz özelliklerini bulmak ve ölçmek için lokal özellik şablonu eşlemesini kullanan lineer bir yerleşik algoritma ve global bir uyum ölçümü tarif ettiler. Bu şablon eşleme yaklaşımı, Yuille ve Cohen'in [15] yakın zamandaki çalışması tarafından devam ettirildi ve geliştirildi. Stratejileri parametre değerlerinin yüz görüntüsü ile etkileşimler tarafından belirlendiği yüz ve özelliklerinin parametreleştirilmiş modelleri olan şekil değiştirilebilen şablonlara dayalıdır.

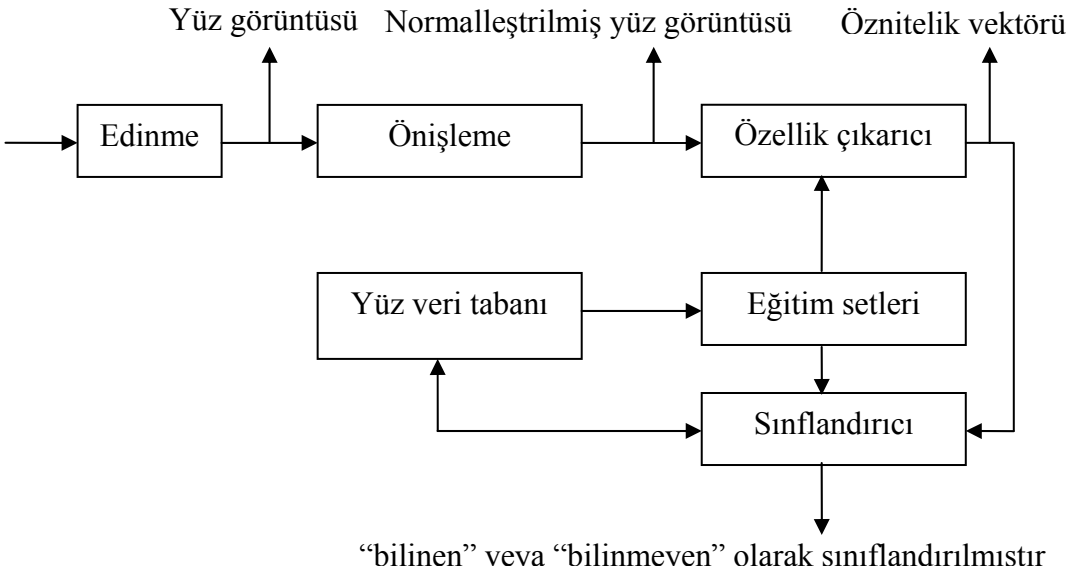
Yüz tanıma ile ilgili bağlantıcı yaklaşımlar, görevin yapısal özelliğini yakalamaya çalışır. Kohenen [21] ve Konenen ve Lehtio [22], yüz görüntülerini tanıyabilen ve eksik veya gürültülü giriş versiyonundan bir yüz görüntüsünü hatırlayabilen basit bir öğrenim algoritması ile asosyatif bir şebekeyi tarif eder. Fleming ve Cottrell [23] geriye yayma ile sistemi eğiterek ve lineer olmayan birimler kullanarak bu fikirleri yaydılar.

Diğerleri de, otomatikleştirilmiş yüz tanıma konusuna bir geometrik parametreler seti ile yüzü karakterize ederek ve parametrelere dayalı görüntü algılanması gerçekleştirerek konuya yaklaştılar. Kanade'nin [24] yüz tanıma sistemi, tanıma sürecinin tüm adımlarının otomatikleştirildiği beklenen öz nitelik karakteristiklerinin, bir jenerik modeli tarafından yönetilen, bir yukarı-aşağı kontrol stratejisi kullanan ilk sistemdir. Bu sistem, tek bir yüz görüntüsünden yüz parametreleri grubunu hesaplar ve yüzü bilinen bir setten eşleştirmek için büyük ölçüde lokal histogram analizine ve mutlak gri skala değerlerine bağlı tamamen istatistiki bir yaklaşım olan bir görüntü sınıflandırma tekniği kullanır.

Burt tarafından [25] yapılan yakın zamandaki bir çalışma, çok çözünürlülüklü şablon eşlemeye dayalı bir akıllı algılama yaklaşımı kullanır. Bu ince strateji çok çözünürlüklü piramit görüntülerini hızlı bir şekilde hesaplamak üzere inşa edilen özel amaçlı bir bilgisayar kullanır ve insanlar gerçek zamana yakın bir şekilde tanımlanarak gösterilmiştir.

2.2.2. Tipik bir yüz tanıma sisteminin planı

Şekil 2.3. de tipik bir yüz tanıma sisteminin planı verilmiştir. Bu plan şekil 2.2. de sunulan tipik görüntü tanıma sisteminin özelliklerini taşır.



Şekil 2.3. Tipik bir yüz tanıma sisteminin planı

Altı adet ana işlevsel blok vardır ve bunların sorumlulukları aşağıda verilmiştir.

- Edinme modülü. Bu, yüz tanıma sürecinin giriş noktasıdır. Söz konusu yüz görüntüsünün sisteme sunulduğu modüldür. Başka bir deyişle, bu modülde, kullanıcıdan yüz tanıma sistemine bir yüz görüntüsü sunması istenir. Edinme modülü çok farklı ortamlarda bir yüz görüntüsü talep edebilir. Yüz görüntüsü manyetik bir diskte bulunan bir görüntü olabilir, bir görüntü yakalayıcı tarafından alınabilir veya bir tarayıcı yardımıyla kağıttan taranabilir.

- Ön işleme modülü. Bu modülde erken görüş teknikleri aracılığıyla yüz görüntüleri normalleştirilir ve istenirse sistemin tanıma performansını iyileştirmek üzere zenginleştirilir. Aşağıdaki ön işleme adımlarının bazıları veya tamamı, bir yüz tanıma sisteminde uygulanabilir.

- Görüntü boyutu normalleştirme. Elde edilen görüntü boyutunu, yüz tanıma sisteminin çalıştığı örneğin 128x128 gibi var sayılan bir görüntü boyutuna değiştirmek için kullanılır. Bu durum, bu tezde önerilen yüz tanıma biçiminde sıklıkla karşılaşılır.

- Histogram eşitleme. Genellikle çok koyu veya çok parlak görüntüler üzerinde görüntü kalitesini artırmak ve yüz tanıma performansını geliştirmek için yapılır. Görüntünün dinamik aralığını (kontrast aralığını) değiştirir ve sonuç olarak bazı önemli yüz özellikleri daha belirgin hale gelir.

- Medyan filtreleme. Özellikle bir kameradan veya bir görüntü yakalayıcıdan elde edilen gürültülü görüntüler için medyan filtreleme, bilgi kaybetmeden görüntüyü temizleyebilir.

- Yüksek geçişli filtreleme. Yüz hatlarına dayalı özellik çıkarıcılar, bir kenar algılama programından elde edilen sonuçlardan istifade edebilir. Yüksek geçişli filtreleme, kenar algılama performansını büyük ölçüde geliştirebilen konturlar gibi, bir görüntünün ayrıntılarını vurgular.

- Arka plan kaldırma. Özellikle yüz bilgisini ele almak için yüz arka planı kaldırılabilir. Tüm bilginin kullanılan görüntüde yer aldığı yüz tanıma sistemleri için bu durum özellikle önemlidir. Arka plan kaldırma için, ön işleme modülünün yüz planını belirleyebilir olması gerektiği açıktır.

- Çevrimsel ve rotasyonel normalleştirmeler. Bazı durumlarda, kafanın bir şekilde kaymış veya dönmüş olduğu bir yüz görüntüsü üzerinde çalışmak mümkündür. Kafa, yüz özelliklerinin belirlenmesinde önemli rol oynar. Özellikle, yüzlerin ön görünümlerine dayalı yüz tanıma sistemleri için kafa konumundaki kaymaları ve dönmeleri ön işleme modülünün belirlemesi ve mümkünse normal hale getirilmesi istenebilir.

- Aydınlatma normalleştirme. Farklı aydınlatmalar altında alınan yüz görüntüleri, özellikle, tüm yüz bilgisinin tanıma için kullanıldığı ana bileşen analizine dayalı yüz tanıma sistemleri için tanıma performansını düşürebilir. Bir resim yansıtırlık dizisi *r(X)* ‘e eşit gibi görülebilir. Bu nedenle aynı aydınlatma *I* altında ilgili resim şu formülle verilir.

$$\Phi(X) = Ir(X) \tag{2.2}$$

Normalleştirme işlemi resim üzerindeki bir X_0 referans noktasında sabit bir I_0 aydınlatma düzeyini empoze ederek yapılır. Normalleştirilmiş resim şu formülle verilir.

$$\Phi(X) = \frac{I_0 \Phi(X)}{I(X_0)} \tag{2.3}$$

Gerçek uygulamada, her gözün altında bir tane olmak üzere 2x2 piksel dizisinden oluşan iki referans noktasının ortalaması kullanılabilir.

- Özellik çıkarma modülü. Bir miktar ön işleme yaptıktan sonra (eğer gerekliyse), normalleştirilmiş yüz görüntüsü, sınıflandırma için kullanılacak ana özellikleri bulmak için özellik çıkarma modülüne sunulur. Başka bir deyişle, bu modül yüz görüntüsünü temsil edecek kadar iyi bir öz nitelik vektörünün oluşturulmasından sorumludur.

- Sınıflandırma modülü. Bu modülde, bir görüntü sınıflandırıcısının yardımı ile yüz görüntüsünün çıkarılmış özellikleri, bir yüz arşivinde (veya yüz veri tabanı) kayıtlı olanlarla karşılaştırılır. Bu karşılaştırmayı yaptıktan sonra yüz görüntüsü, bilinen veya bilinmeyen olarak sınıflandırılır.

- Eğitim seti. Eğitim setleri yüz tanıma sürecinin "öğrenme aşaması" sırasında kullanılır. Özellik çıkarma ve sınıflandırma modülleri, eğitim setlerini kullanarak optimum tanıma performansına ulaşmak için parametrelerini ayarlar.

- Yüz arşivi veya yüz veri tabanı. "bilinmeyen" olarak sınıflandırıldıktan sonra yüz görüntüleri, daha sonra karşılaştırmak için öz nitelik vektörleri ile birlikte bir arşive (veya bir veri tabanına) eklenebilir. Sınıflandırma modülü yüz arşivini doğrudan kullanır.

2.2.3. Yüz tanıma sırasında meydana gelebilecek problemler

Yüz görüntülerinin dinamik özelliği nedeniyle, bir yüz tanıma sistemi işlem sırasında çeşitli problemlerle karşılaşır. Bu şartlar altında bir yüz tanıma sistemini tanıma performansına dayalı olarak "sağlam" veya "zayıf" olarak sınıflandırmak mümkündür. Sağlam bir yüz tanıma sisteminin hedefleri aşağıda verilmiştir:

- Oran değişmezliği. Aynı yüz, şekil 2.4-b'de gösterildiği gibi sisteme farklı oranlarda sunulabilir. Bu yüz ve kamera arasındaki fokal mesafe nedeniyle olabilir. Bu mesafeye yakınlaştıkça yüz görüntüsü büyür.

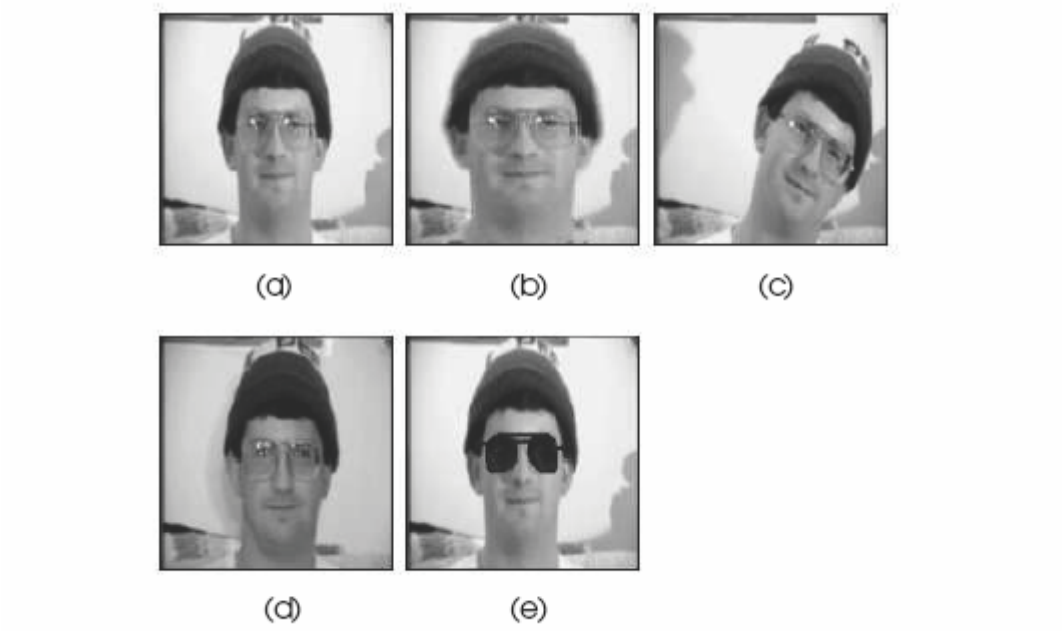
-Kayma değişmezliği. Aynı yüz, şekil 2.4-c'de gösterildiği gibi sisteme farklı perspektiflerde ve yönelimlerde sunulabilir. Örneğin, aynı kişinin yüz görüntüleri, ön ve profil görünümlerden alınabilir. Ayrıca kafa yönelimi, çevrimler ve rotasyonlar nedeniyle değişebilir.

- Aydınlatma değişmezliği. Aynı kişinin yüz görüntüleri, konum gibi farklı aydınlatma koşullarında alınabilir ve ışık kaynağının gücü şekil 2.4-d'de gösterildiği gibi değiştirilebilir.

- Duygusal ifade ve ayrıntı değişmezliği. Aynı kişinin yüz görüntüleri gülerken veya ağlarken ifade olarak farklı olabilir. Ayrıca şekil 2.4-e'de gösterildiği gibi koyu gözlükler sakal veya bıyık gibi bazı detaylar bulunabilir.

- Gürültü değişmezliği. Sağlam bir yüz algılama sistemi, görüntü yakalayıcılar veya kamera tarafından üretilen görüntüye duyarlı olmamalıdır. Ayrıca kısmen kapalı görüntüler ile de çalışabilmelidir.

Sağlam bir yüz tanıma sistemi, eğer yüz, veri tabanında kayıtlı ise, yukarıdaki şartlar altında dahi bir yüz görüntüsünü "bilinen" olarak sınıflandırabilmelidir.



Şekil 2.4. Yüz görüntülerinin farklı durumları; (a) orijinal yüz görüntüsü; (b) oran değişmesi; (c) oryantasyon değişmesi; (d) aydınlatma değişmesi; (e) ayrıntı bulunması.

2.2.4. Özellik tabanlı yüz tanıma

Daha önce yüz tanıma probleminde iki temel yaklaşım olduğundan bahsedilmişti. Bunlar özellik tabanlı yüz tanıma ve ana bileşen analiz metotlarıdır. Özellik tabanlı yüz tanıma, ön görünümlere ve profil siluetlere dayalı olarak iki farklı kategoriye ayrılabilse de, bazı ortak özelliklerinden dolayı bunlar bir bütün olarak ele alınacaktır. Bu bölümde, ön görünümlerden [26] özellik tabanlı yüz tanımanın temel prensipleri sunulmaktadır.

2.2.4.1. Giriş

İnsan yüzü tanımanın ilk adımı, yüz görüntülerinden özellikleri çıkarmaktır. Özellik seçimi alanında sorun, gözler, ağız, çene ve burun gibi ayrı özelliklerin, yüzlerin ayrımı ve tanınması için önemli ipuçları olarak bulunduğu dikkat çekici işaretler üzerinde çalışılmak suretiyle ele alınmasıdır.

Yüz tanıma için etkin özelliklerin ne olduğunu bildikten sonra, gözlerin, kaşların, ağız, burun ve yüzün konturlarını almak için bazı metotlar kullanılmalıdır. Farklı yüz

konturları için, orjinal portreden çıkarmak için farklı modeller kullanılmalıdır. Gözlerin ve ağzın şekli, bazı geometrik şekillere benzediği için değiştirilebilir şablon modeli [15] açısından çıkarılabilir.

Kaşlar, burun ve yüz gibi diğer ilgili özellikler o kadar değişkendir ki aktif kontur modeli ile çıkarılmaları gerekir [27, 28]. Bu iki model aşağıdaki şekilde gösterilebilir:

- Değiştirilebilir şablon modeli. Değiştirilebilir şablonlar, kontur deformasyon işlemine kılavuzluk etmek üzere özelliklerin, beklenen şekli hakkında öncelik bilgisini kullanan bir parametreler seti tarafından belirtilir. Şablonlar, boyutlarını ve diğer parametre değerlerini değiştirmeye yetecek kadar esnektir ve bu şekilde kendilerini veriye uydururlar. Bu parametrelerin nihai değerleri, özellikleri tarif etmek için kullanılabilir. Bu metot, kafanın ölçeği, eğimi ve rotasyonlarındaki değişimlere bakmaksızın iyi çalışır. Parametrelerdeki değişiklikler şablonun özelliğinin her hangi bir durumuna uymasına izin vermelidir. Değiştirilebilir şablonlar, görüntü ile dinamik bir tarzda etkileşimde bulunurlar. Şablonun görüntü yoğunluğundaki, kenarlardaki ve yoğunluğun kendisindeki üst ve alt noktalar gibi, dikkat çekici özelliklere çeken şartları içeren bir enerji fonksiyonu tanımlanır. En küçük enerji fonksiyonu görüntü ile en iyi uyuma karşılık gelir. Şablonun parametreleri daha sonra en dik düşüşle güncellenir.

- Aktif kontur modeli (yılan). Aktif kontur veya yılan, harici kısıtlayıcı güçler tarafından yönlendirilen ve çizgiler ve kenarlar gibi özelliklere çeken görüntü kuvvetleri tarafından etkilenmiş enerji küçültücü bir eğridir. Yılanlar yakın kenarlara kitlenir ve onları doğru bir şekilde lokalize eder. Yılan, bir enerji küçülten eğri olduğu için en küçük lokal değerleri, daha yüksek düzeydeki süreçlere alternatif çözümler içeren enerji fonksiyonları tasarlamalıdır. Bu setten bir yanıtın seçilmesi, modeli, istenen çözüme iten enerji şartlarının eklenmesi ile yapılır. Sonuç olarak, yakınına yerleştirildiğinde istenen çözüme karşılık gelen bir aktif model elde edilir. Aktif kontur modelinde, konturların bağlantısallık ve köşelerin varlığı gibi konular, enerji fonksiyonunu ve bu nedenle lokal olarak optimal konturların yapısını etkiler. Bu konular son derece yüksek düzeyde hesaplamalarla çözülebilir.

2.2.4.2. Etkin özellik seçimi.

Yüz özellik çıkarma prosedürlerinden bahsetmeden önce, aşağıdaki iki konuya değinmek istiyoruz.

- İyi bir enstantane elde etmek için resim çekme ortamı sabitlenmelidir.

- Bir yüzü, verimli bir şekilde tanımlamak üzere kullanılabilecek etkin özellikler bilinmelidir.

Uzaysal görüntüler gibi, yüzlerin işaretlenmiş benzerliklerine rağmen, potansiyel olarak sınırsız sayıda yüzü ayırt edebilir ve hatırlayabiliriz. Yeterli bir aşinalıkla, her hangi iki insanın yüzleri birbirinden ayrılabilir. Beceri, saç stilleri, duygusal ifade ve yüz hareket etkisi gibi yüzün geçici durumundan, değişmez yapısal bilgileri çıkarabilme yeteneğine bağlıdır.

Özellikler, nesne tanımının temel unsurlarıdır. Bu nedenle, bir yüzü tanımlamak için yüz tanıma sürecinde, hangi özelliklerin verimli bir şekilde kullanıldığını bilmemiz gerekir. Yüz tanıma süreci ile ilgili her özelliğin değişkenliği nispeten büyük olduğu için, özellikler üç ana türe ayrılmıştır.

- Birinci sırada özellik değerleri. Gözler, kaşlar, ağız, çene ve burun gibi özellikler yüz tanımasında önemli bulunmuş olup, diğer yüz özelliklerine atıfta bulunmadan belirtilmiş özelliklere, birinci sıra özellikler denir. Önemli birinci sıra özellikler tablo 2.1. 'de verilmiştir.

- İkinci sıra özellik değerleri. Birinci sıra özelliklerinin konumları ile yüzün şekli hakkındaki bilgi arasında uzaysal ilişkileri karakterize eden diğer bir yapısal özellik setine, ikinci sıra özellikler adı verilir. Önemli ikinci sıra özellikleri tablo 2.2. ’de verilmiştir. Burun fark edilebiliyorsa, burunla ilgili ikinci sıra özellikler tablo 2.3. ’de verilmiştir.

- Daha yüksek sırada özellik değerleri. Ayrıca değerleri karmaşık bir özellik değerleri setine bağlı olan daha yüksek düzeyde özellikler bulunmaktadır. Örneğin, yaş, saç kapsamı, saç rengi, deri gerginliği, kırışıklık, ben bulunması ve azalan saç nedeniyle değişen alın yüksekliği vs. gibi bir fonksiyonu olabilir.

Duygusal ifade veya cilt gerilimi gibi değişkenler, yüksek sıralı özelliklerde bulunur ve birinci sıra ve ikinci sıra özelliklerinin fonksiyonu olan karmaşıklığı tahmin etmek çok zordur. Daha yüksek özelliklere ait olan daimi bilgiler, birinci ve ikinci sıra özelliklerini kullanarak bulanamaz. Bu nedenle bu özellikler sadece birinci sıra ve ikinci sıradakileri içerebilirler. Bu etkin özellik değerleri, bir portreden hemen hemen elde edilebilir tüm bilgileri kapsar. Bunlar yüz tanıma işlemi için yeterlidir.

İkinci sıranın özellik değerleri birinci sıradan daha önemlidir ve öz nitelik vektöründe baskındır. Yüz öz nitelik çıkarma işleminden bahsetmeden önce, iki ön işleme adımını öne almak gerekir.

- Eşik atama. Parlaklık eşiği, yüzün öz niteliğini ve diğer alanlarını ayırt etmek için bilinmelidir. Genel olarak kaşlar, gözler, ağız, burun ve yüz için resmin parlaklığına göre farklı eşikler kullanılır.

- Kaba kontur tahmin rutini (RCER). Sol kaş çıkarılması gereken ilk özelliktir. Birinci adım, sol kaşın kaba konturunu tahmin etmek ve kontur noktalarını bulmaktır. Bu öncelik bilgisine sahip olarak sol kaşın kaba konumu bulunabilir ve kaba konturu yakalanabilir. Sol kaşın kaba konturu oluşturulduğunda sol göz, sağ kaş, ağız veya burun gibi diğer yüz özelliklerin kaba konturları RCER tarafından tahmin edilebilir [29]. Kaba kontur elde edildikten sonra net kontur, değiştirilebilir şablon modeli veya aktif kontur modeli tarafından çıkarılacaktır.

Tablo 2.1. Birinci sıra özellikleri

Ölçüm	Yüz Lokasyonu
Alan, Açı	sol kaş sağ kaş sol göz sağ göz ağız yüz
Mesafe	sol kaşın uzunluğu sağ kaşın uzunluğu sol gözün uzunluğu sağ gözün uzunluğu ağzın uzunluğu yüzün uzunluğu yüzün yüksekliği

Tablo 2.2. İkinci sıra özellikleri

Ölçüm	Yüz Lokasyonu
Mesafe	sol kaş <–> sağ kaş sol göz <–> sağ göz sol kaş <–> sol göz sağ kaş <–> sağ göz sol kaş <–> ağız sağ kaş <–> ağız sol göz <–> ağız sağ göz <–> ağız kaş <–> yüzün yanı göz <–> yüzün yanı ağız <–> yüzün yanı ağız <–> yüzün alt tarafı
Açı	sol kaş – sol göz – sol kaş sağ kaş – sağ göz – sağ kaş sol göz – sol kaş – sol göz sağ göz – sağ kaş – sağ göz sol kaş – ağız – sağ kaş sol göz – ağız – sağ göz sol kaş – sol göz – ağız sağ kaş – sağ göz – ağız

Tablo 2.3. Eğer burun fark edilebiliyorsa burun ile ilgili özellikler

Ölçüm	Yüz Lokasyonu
Mesafe	Sol burun <–> sağ burun Sol kaş <–> sol burun Sağ kaş <–> sağ burun Sol göz <–> sol burun sağ göz <–> sağ burun sol burun <–> ağız sağ burun <–> ağız
Açı	Sol kaş – burunun ortası – sağ kaş Sol göz – burunun ortası – sağ göz Sol burun – ağız – sağ burun Sol kaş – sol göz – sol burun Sağ kaş – sağ göz – sağ burun

2.2.4.3. Değiştirilebilir şablonları kullanarak özellik çıkarma

Kaba kontur elde edildikten sonra yüz algılamadaki diğer adım, her özelliğin fiziksel konturunu bulmaktır. Lokal bilgileri makul bir global algılama şeklinde organize edemediği için klasik kenar detektörleri, göz veya ağız konturları gibi yüz özelliklerini lokal kenar delillerinden doğru bir şekilde bulamazlar. Gözün konturunu, şekil değiştirilebilir şablon ile algılamak için Yullie [15] tarafından önerilmiş olan bir metot vardır. Buna göre çıkarılan konturun hassasiyeti azaltılarak hesaplamaları azaltmak mümkündür.

2.2.4.4. Göz şablonu

Şekil değiştirilebilen şablon, görüntünün üç temsilinde ve kendisinde çalışır. İlk iki sunum, görüntü yoğunluğundaki üst ve alt noktalardır ve üçüncüsü görüntü yoğunluğunun değiştiği yerdir. Yullie ve arkadaşları tarafından geliştirilen göz şablonu aşağıdaki özelliklerden oluşur.

- r yarı çaplı bir daire, irise karşılık gelen bir (x_c, y_c) noktasına, irisin sınırları ve gözün beyazları ise, göz yoğunluğundaki kenarlara çekilir. Dairenin içi çukurlara veya görüntü yoğunluğundaki düşük değerlere çekilir.

- Gözün bağlayıcı bir konturu kenarlara çekilir. Bu kontur, sınırın üst ve alt kısımlarını temsil eden iki parabolik bölüm tarafından modellenir ve (x_c, y_c) merkez noktasına sahiptir. 2w ile beraber merkezin üzerindeki sınırın maksimum yüksekliği h_1, merkezin altındaki sınırın maksimum yüksekliği h_2 ve rotasyon açısı Ø'dır.

- Gözün beyazları için merkezlere karşılık gelen görüntü yoğunluğunda zirvelere çekilen iki nokta vardır.

- Sınır konturu ve iris arasında gözün beyazlarına karşılık gelen bölgeler. Bunlar büyük yoğunluk değerlerine çekilecektir.

Orijinal göz şablonu basitlik amacıyla değiştirilebilir ve burada çıkarılan konturun doğruluğu çok önemli değildir. Bir dairenin olmaması, sınıflandırılmış sonuçları etkilemez. Çünkü özellik değerleri diğer bilgiden alınır. Üst ve alt parabol tanıma işlemi için yeterli olacaktır. Bu nedenle göz şablonu için enerji fonksiyonu, kenarın enerji fonksiyonlarının bir fonksiyonu olarak, siyah ve beyaz noktalar şeklinde tanımlanılabilir.

Toplam enerji fonksiyonu aşağıdaki şekilde tanımlanır:

$$E_{toplam} = E_{kenar} + E_{beyaz} + E_{siyah} \tag{2.6}$$

Burada $E_{kenar,}$ E_{beyaz} ve E_{siyah} (2.7) ve (2.8) denklemleri ile tanımlanır:

- Kenar potansiyelleri, uzunluklarına bölünen üst ve alt parabolün eğimleri entegral ile verilir.

$$E_{kenar} = -\frac{W_1}{en\_üst\_uzunluk} \int_{en\_üst\_sınır} \Phi_{kenar}(x,y)dS - \frac{W_2}{en\_alt\_uzunluk} \int_{en\_alt\_sınır} \Phi_{kenar}(x,y)dS \tag{2.7}$$

Burada üst ve alt sınır gözün üst ve alt kısımlarını temsil eder ve Φ_{kenar} (x, y) noktasının kenar tepkisini temsil eder. W_1 ve W_2 ağırlıkları temsil eder.

- Siyah ve beyaz noktaların potansiyeli, üst ve alt parabol tarafından bölünen sınırlandırılmış alan üzerinden entegrali ile tanımlanır:

$$E_{b,s} = -\frac{1}{Alan} \iint_{para\_alan} (-w_s N_{siyah}(x,y) + w_b N_{beyaz}(x,y))dA \tag{2.8}$$

Burada N_{siyah} *(x, y)* ve N_{beyaz} *(x, y),* siyah ve beyaz noktaların sayısını temsil eder. w_s, w_b siyah ve beyaz noktalarla ilgili ağırlıklardır.

Doğru olmayan eşik tarafından etkilenmemek için 2.8 denklemindeki siyah ve beyaz noktalar şu şekilde tanımlanır.

P(x,y) bir siyah noktadır, eğer I(x,y) ≤ (eşik - tolerans),
P(x,y) bir beyaz noktadır, eğer I(x,y) ≥ (eşik + tolerans), (2.9)
P(x,y) bir belirsiz noktadır, eğer I(x,y) arada ise

Burada I(x,y), (x,y) noktasındaki görüntü yoğunluğudur.

Yukarıda tanımlanan enerji fonksiyonları ile, 2w, h_1, h_2 ve Φ 'da yapılan ufak modülasyonlarla enerjiyi hesaplayabiliriz.

2.2.4.5. Ağız şablonu

Yüzün ön görünümünün tam özelliklerinde ağzın rolü nispeten önemlidir. Ağız konturunun özellikleri, yüz algılama işleminde yoğun bir şekilde yer alır. Kenarın görüntü alanlarına (burada yoğunluk hızlı bir şekilde değişir) ve siyah ve beyaz noktalara geldiğinde, şekil değiştirebilen ağız şablonu kendi şeklini değiştirir. Genel olarak, orta dudaklar (dudaklar arasındaki boşluk), üst ve alt dudaklar ile ilgili özellikler çıkarılır. Resim çekme sürecindeki parlaklık etkisi nedeniyle, alt dudağın ortası belirgin olmayabilir. RCER, alt dudağın yaklaşık yüksekliğini bulamaz. Ancak, ağzın uzunluğu RCER tarafından bulunabilir. Genel olarak alt dudağın yüksekliği, ağzın uzunluğunun dörtte biri ve altıda biri arasındadır.

Ağız konturu enerji fonksiyonu E_{kenar} kenar terimi ve E_{siyah} siyah teriminden oluşur. Kenar terimi kenar alanına hakimdir, siyah terimi ağza ait mümkün olduğunca fazla siyah nokta içerir.

$$E_{toplam}=E_{kenar}+E_{siyah} \tag{2.10}$$

- Kenar enerji fonksiyonu üç kısımdan oluşur. Orta dudak, alt dudak ve üst dudaktaki oluktan ayrılan üst dudak. Orta dudak kısmının denklemi (2.11) 'de gösterilmektedir.

$$E_{kenar} = \begin{aligned} &-\frac{w_{alt}}{alt\_uzunluk}\int_{alt}\Phi_{kenar}(x,y)dS-\frac{w_{sol}}{sol\_uzunluk}\int_{sol}\Phi_{kenar}(x,y)dS \\ &-\frac{w_{sağ}}{sağ\_uzunluk}\int_{sağ}\Phi_{kenar}(x,y)dS \end{aligned} \tag{2.11}$$

Burada *alt* ağzın alt sınırını, *sol* üst dudağın sol kısmını, *sağ* üst dudağın sağ kısmını ve Φ_{kenar} (x, y), (x, y) noktasının kenar tepkisini temsil eder.

- Siyah enerji fonksiyonu kenar enerjinin ağza ait siyah noktaları kapsamasına yardımcı olur ve (2.12) denkleminde olduğu gibi tanımlanır.

$$E_{siyah}=\frac{1}{Alan}\int\int_{\text{Altsinir}}^{\text{Ustsinir}} -w_{siyah}N_{siyah}(x,y)dA+\frac{1}{orta\_uzunluk}\int_{orta}-w_{orta}N_{siyah}(x,y)dS \tag{2.12}$$

Burada *Altsinir* alt dudağı, *Ustsinir* üst dudağı, ve *orta* siyah noktaların sayısını temsil eder. (2.12) denklemindeki siyah noktalar (2.9) denklemi tarafından tanımlanır. w_{siyah}, w_{orta}, w_{alt}, w_{sol} ve $w_{sağ}$ ağırlıkları deneysel olarak belirlenir.

2.2.4.4. Aktif konturu kullanarak özellik çıkarma

Kaş, burun deliği ve yüzün şekilleri, göz ve ağzın aksine farklı insanlar için daha farklıdır ve konturları şekil değiştirilebilen şablon kullanarak yakalanılamaz. Bu durumda aktif kontur modeli veya "yılan" kullanılır. Yılan, harici kısıtlayıcı güçler

tarafından yönlendirilen ve çizgiler ve kenarlar gibi özelliklere doğru çeken görüntü kuvvetleri tarafından etkilenmiş, enerji küçültücü bir eğridir. Bu yaklaşım, kenarları algılayan ve sonra bağlayan geleneksel yaklaşımlardan farklıdır. Aktif kontur modelinde, resim ve konturların bağlantısallığı ve köşelerin varlığı ile birlikte dış güçler, enerji fonksiyonunu ve lokal optimal konturun yapısını etkileyecektir. Aktif kontur modelinin enerji fonksiyonu, (2.13) denkleminde ki gibi tanımlanır [28].

$$E_{\text{yilan}} = \int_0^1 E_{\text{yilan}}(v(s))dS = \int_0^1 E_{\text{dahili}}(v(s)) + E_{\text{resimler}}(v(s)) + E_{\text{harici}}(v(s))dS \qquad (2.13)$$

Burada v(s), yılanın konumunu, E_{dahili} eğilme nedeniyle konturun dahili enerjisini temsil eder, $E_{resimler}$ görüntü kuvvetlerini artırır, E_{harici} ise harici enerjiyi temsil eder.

2.2.4.5. Değiştirilmiş aktif kontrol modeli

Orijinal aktif kontrol modeli kullanıcı etkileşimlidir. Bu şekilde olmasının avantajı, yılanın son formunun, daha yüksek seviyeli işlemlerden bir geri besleme ile etkilenebilir olmasıdır. Algoritma tekrar ettikçe, enerji terimleri bu işleme en faydalı görülen en küçük bir lokal değer elde etmek için, daha yüksek düzeyde işlemler tarafından ayarlanabilir. Ancak küçültme prosedürü ile bazı problemler vardır. Amini ve arkadaşları [30], kararsızlık ve noktaların, bir kenarın kuvvetli kısmında toplanması eğilimi gibi bazı problemleri işaret etmişlerdir. Bu noktada enerji fonksiyonunu minimize etmek için, dinamik bir programlama algoritması önerdiler. Onların yaklaşımı, noktaların ayrı bir tabloda kullanılması avantajına sahiptir ve yakınsama çok yavaş olmakla beraber sayısal olarak kararlıdır.

Aktif kontur için daha hızlı bir algoritma bulmak mümkündür [29]. Bu model global bir en küçük değer garanti edememe dezavantajına sahip olmakla birlikte aktif konturda, kuvvetli bir kısımda kümelenme problemini çözebilir. Bu problem, tekrar eden işlem sırasında meydana gelir ve kontur noktaları bu sırada, aktif konturun belirli güçlü kısımlarında toplanacaktır. Ayrıca hesaplama hızı daha fazladır ve bu nedenle yüz tanıma için daha uygundur. Aktif kontur enerjisi (2.14) denkleminde olduğu gibi yeniden tanımlanabilir [26].

$$E_{toplam} = \int_0^1 (\alpha(s)E_{devamlilik}(v(s)) + (\beta(s)E_{egrilik}(v(s)) + (\delta(x)E_{resim}(v(s))dS \qquad (2.14)$$

v(s) tanımı denklem 2.13'e benzemektedir ve (2.15) denkleminde ki tahminler kullanılır.

$$\left|\frac{dv_i}{dS}\right| \approx \left|v_i - v_{i-1}\right|^2 \text{ ve } \left|\frac{d^2v_i}{dS^2}\right| \approx \left|v_{i-1} - 2v_i + v_{i+1}\right|^2 \qquad (2.15)$$

Bu formüller, konturdaki noktaların, birim aralıklarında bulunduğunu ve parametrenin yay uzunluğu olduğunu varsayar. Bunun bir etkisi eşit olmayan şekilde yerleştirilen noktaların daha yüksek eğime sahip olacağıdır. Değiştirilmiş enerji fonksiyonunda yer alan güçlerin özellikleri, aşağıdaki şekilde tarif edilebilir:

- Süreklilik kuvveti. Birinci türev $\left|v_i - v_{i-1}\right|^2$, eğimin küçülmesine neden olur. Noktalar arasındaki mesafeyi küçültür. Ayrıca, konturun güçlü kısımlarında kümelenme problemine katkıda bulunur. Noktaların eşit aralıkla yerleştirilmesini teşvik eden bir terimin, küçülme etkisi olmadan, birinci sıra devamlılık hedefini karşılayacağına karar verilir. Burada olan bu terim, ortalama mesafe noktaları *d* ve söz konusu iki nokta arasındaki mesafe $\left|d - \left|v_i - v_{i-1}\right|\right|$ farkını kullanır. Bu nedenle, ortalamaya yakın bir mesafesi olan noktalar, en az değere sahip olacaktır. Her tekrar sonunda yeni bir *d* değeri hesaplanır.

- Eğim kuvveti. Süreklilik teriminin formülasyonu, noktaların nispeten eşit aralıkla yerleştirilmesine neden olduğundan, $\left|v_{i-1} - 2v_i + v_{i+1}\right|^2$ terimi eğimin makul ve çabuk bir tahminini verir. Bu terim süreklilik terimi gibi, bölgedeki en yüksek değer bölünerek normalleştirilir ve 0-1 arasında bir sayı verir.

- Görüntü kuvveti. $E_{resimler}$ terimi, aşağıdaki işlemler tarafından tanımlanan görüntü kuvvetidir. Sekiz bölge için sekiz görüntü enerjisi ölçümümüz (mag) vardır. Görüntü

enerji ölçümlerini normalleştirmek için bu sekiz ölçümden minimum (min) ve maksimum (max) terimleri seçip, daha sonra görüntü kuvvetini elde etmek için (min– mag) / (max – min) hesabı yapılır.

Her tekrardan sonra eğim, yeni konturun her noktasında belirleniyor. Eğer değer daha büyük ise, β eşiği bir sonraki tekrar için sıfır olarak ayarlanır. Hızlı yakınsama için greedy algoritması [29] uygulanır. Enerji fonksiyonu, v_i 'nin ve her bir komşusunun mevcut lokasyonu için hesaplanır. En küçük değere sahip komşu v_i 'nin yeni konumu olarak seçilir. Daha sonra kaş, burun deliği ve benzeri özellikleri çıkarmak için greedy algoritması uygulanabilir. Yılanın yanlış lokal minimada hataya düşmesini engellemek için, kontur tahmini, yılan tekrarlamadan önce RCER üzerinden yapılır. RCER, yılan için bir başlangıç noktası olarak kaba bir kontur bulmak üzere bir öncelik bilgisi kullanır.

2.2.4.6. Bir yüzün sınır çıkarması

Aktif kontur modellerinin kullanımını göstermek için, bu bölümde, bir yüzün sınır çıkarması ile ilgili enerji fonksiyonu verilmiştir.

Yüzün kaba konturu RCER tarafından tahmin edilemediği için kaş çıkarmasının aksine, bir yüzün sınır çıkarması daha zaman alıcıdır. Ancak kaba konturun mümkün olduğunca doğru bir şekilde tahmin edilmesi gerekir. Bir yüzün sınır çıkarması ile ilgili enerji fonksiyonu (2.16) denklemindeki gibi tanımlanır.

$$E_{yuz} = \sum_{i=1}^{n} (\alpha_i E_{devamlilik} + \beta_i E_{egrilik} + \delta_i E_{resimler}) \tag{2.16}$$

Buradaki $E_{\text{devamlilik}}$ ve E_{egriler} bölüm 2.4.4.1'de daha önce tanımlanmıştır ve E_{resimler} terimi (2.17) denklemindeki gibi tanımlanır.

$$E_{resimler} = w_{cizgiler} E_{cizgiler} + w_{kenar} E_{kenar} \tag{2.17}$$

E_{cizgiler} teriminin amacı, aktif kontur içinde daha fazla beyaz noktalar ve daha az koyu

noktalar çekmektir, E_{kenar} teriminin amacı ise aktif kontur sınırında daha fazla nokta çekmektir.

$E_{cizgiler}$ ve E_{kenar} (2.18) denkleminde olduğu gibi tanımlanır.

$$E_{cizgiler}= \begin{cases} -Mag_{beyaz} & \text{, eğer } \beta(x,y) = 0 \text{ ve } I(x,y) \geq \text{eşik} + \text{tolerans} \\ -Mag_{koyu} & \text{, eğer } \beta(x,y) = 0 \text{ ve } I(x,y) < \text{eşik} - \text{tolerans} \\ -Mag_{beyaz} & \text{, eğer } \beta(x,y) = 1 \text{ ve } I(x,y) \geq \text{eşik} + \text{tolerans} \\ -Mag_{koyu} & \text{, eğer } \beta(x,y) = 1 \text{ ve } I(x,y) < \text{eşik} - \text{tolerans} \\ -Mag_{kontur} & \text{, eğer } \beta(x,y) = 255 \text{ ve } I(x,y) \geq \text{eşik} + \text{tolerans} \\ -Mag_{kontur} & \text{, eğer } \beta(x,y) = 255 \text{ ve } I(x,y) < \text{eşik} - \text{tolerans} \\ -0 & \text{, diğer türlü} \end{cases} \quad (2.18)$$

Burada *I(x,y,)* (*x,y)* noktasının yoğunluk değerini temsil eder ve *eşik* yüz eşiği ile aynıdır, *tolerans* deneysel olarak belirlenir.

$E_{yüz}$ sürekli olarak azaldığından sakalsız bir yüzün sınırını çıkarmak için, yılan tekrar eder ve çeneye doğru ilerler. Yılanın yakınsama işlemi greedy algoritmasına dayalıdır. Tekrar eden işlem durduğunda, tekrarlara sonraki yakınsak yerini bulmak için yeniden başlanabilir. Eğer aynı çıkarsa, konum doğrudur.

2.2.5. Temel bileşen analizine dayalı yüz tanıma

Bölüm 2.2.4'de, öz nitelik çıkarmaya dayalı bir yüz tanıma metodunu gözden geçirildi. Geniş oranda geometri kullanarak göz, kaş, burun, ağız ve hatta yüzün kendisinin konturlarını bulmak mümkündür.

Yüz tanıma için temel bileşen analizi, bilgi teorisi yaklaşımına dayalıdır. Burada bir yüz görüntüsündeki ilgili bilgi mümkün olduğunca etkin bir şekilde çıkarılır ve kodlanır. Tanıma benzer şekilde kodlanan bir yüz veritabanından yapılır.

Bir görüntüyü matematiksel açıdan, son derece yüksek boyutsal yüz boşluğunda bir nokta (vektör) olarak ele alan, yüzlerin dağıtımının temel bileşenleri veya yüz

görüntüleri setinin kovaryans matrisinin özyüzleri aranır. Temel bileşen analiz metodu üçüncü bölümde daha ayrıntılı olarak sunulacaktır.

BÖLÜM 3. ÖZYÜZ KULLANILARAK YÜZ TANIMA

Bu bölüm, yüz tanımada kullanılan, bir ana bileşen analizinin açıklamasını içerir. Önerilen yüz tanıma sistemi bu metoda dayalıdır.

3.1. Giriş

Otomatik yüz tanıma üzerinde yapılan önceki çalışmaların bir çoğu, yüz uyarıcısının hangi yönlerinin, yüz tanıma için önemli olduğu konusunu ihmal etmiştir. Buna göre, önemli lokal ve global özellikleri vurgulayan yüz görüntülerinin kodlanması ve şifrelerinin çözülmesi için, bir bilgi teorisi yaklaşımının kullanımı önerilir. Bu özellikler, gözler, burun, dudaklar ve saç gibi yüz özellikleri hakkındaki sezgisel nosyonumuz ile doğrudan ilgili olabilir yada olmayabilir.

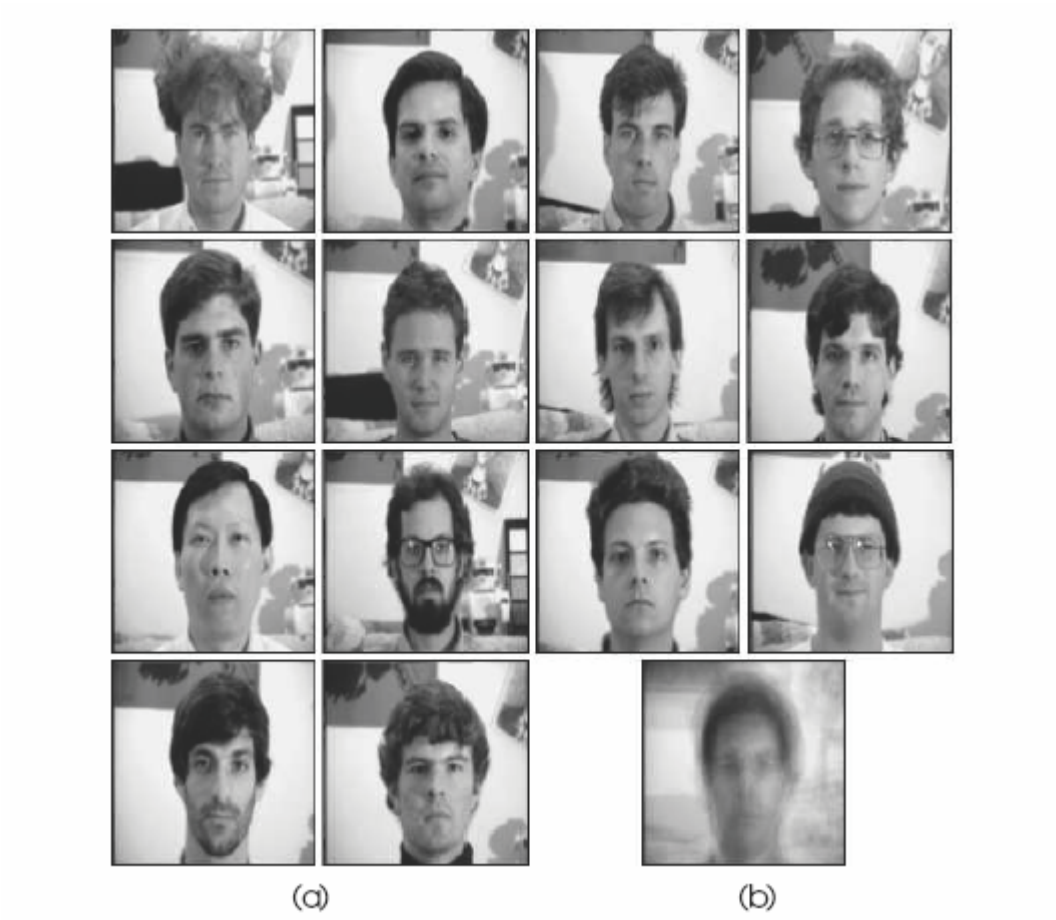
Bilgi teorisi dilinde, bir yüz görüntüsünün ilgili bilgisi çıkarılır, mümkün olduğunca verimli bir şekilde kodlanır ve sonra benzer şekilde kodlanan bir model, veri tabanı ile karşılaştırılır. Bir yüz görüntüsü içerisinde yer alan basit bir yaklaşım, yüz görüntüleri grubundaki değişikliği, her hangi bir özellik yargısından bağımsız bir şekilde yakalamak ve bu bilgiyi ayrı ayrı yüz görüntülerini kodlamak ve karşılaştırmak üzere kullanmaktır.

Matematiksel açıdan, yüzlerin dağıtımının ana bileşenleri veya yüz görüntüleri setinin koveryans matrisinin öz vektörleri, çok yüksek boyutlu bir uzayda görüntüyü nokta (veya vektör) olarak ele alacak şekilde aranır.

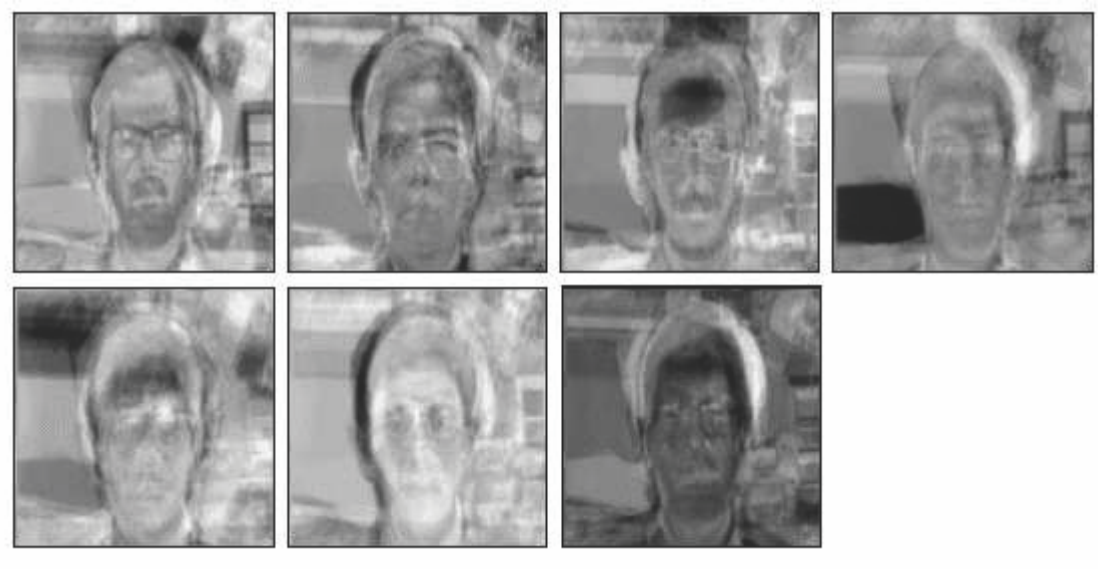
Öz vektörler düzenlenir ve bunların her biri yüz görüntüleri arasında farklı bir değişiklik tutarına karşılık gelir. Bu öz vektörler, yüz görüntüleri arasında varyasyonu karakterize eden bir özellikler seti olarak düşünülebilir. Her görüntü

lokasyonu aşağı yukarı her öz vektöre katkıda bulunarak, bu öz vektörlerin "özyüz" adı verilen bir tür hayalet yüz şeklinde görüntülenmesi mümkün olur.

Örnek yüz görüntüleri ve bunlara karşılık gelen özyüzler sırasıyla şekil 3.1-a ve şekil 3.2. 'de gösterilmiştir. Her özyüz, bazı yüz özelliklerinin eğitim yüzleri setinden farklı olduğu tek düze griden farklılık gösterir.



Şekil 3.1. Örnek, eğitim seti yüz görüntüleri ve eğitim setinin ortalama yüz görüntüsü



Şekil 3.2. En yüksek öz değerlere sahip 7 özyüz, şekil 3.1.'de verilen örnek eğitim setinden hesaplanmışlardır

Her bir yüz, özyüzlerin lineer bir kombinasyonu açısından, kesin bir şekilde temsil edilebilir. Her yüz, ayrıca sadece "en iyi" özyüzleri kullanarak tahmin edilebilir. Bu en iyi özyüzler en büyük öz değerlere sahiptirler ve bu nedenle yüz görüntüleri seti içinde en fazla değişime karşılık gelirler. En iyi M özyüzleri, tüm muhtemel görüntülerin "yüz uzayı" adı verilen bir M boyutlu alt uzayı kapsar.

Kirpy ve Sirovich [4, 5], temel bileşen analizini kullanarak, yüzlerin resimlerinin verimli bir şekilde temsil edilmesi için bir teknik geliştirmiştir. Orijinal yüz görüntülerinin bir takımı ile başlayarak görüntü sıkıştırmak için, en iyi koordinat sistemini hesaplamışlardır ve burada her koordinat "özresim" adını verdikleri bir görüntüdür. En azından prensipte yüz görüntülerinin her hangi bir grubunun, her yüz için küçük bir ağırlıklar grubu ve küçük bir standart resimler (özresimler) seti kaydederek, yaklaşık bir şekilde yeniden yapılandırılabileceğini ileri sürdüler. Her yüzü tarif eden ağırlıklar, yüz görüntüsünü her öz resmine yansıtarak bulunur.

Bu çalışmada, özyüzler yaklaşımına dayalı olarak, bir yüz tanıma sistemi geliştirmek üzere M. Turk ve A. Pentland [14] tarafından önerilen metod takip edildi. Bu kişiler, eğer bir yüz görüntüleri grubu, küçük bir karakteristik özellikler veya özyüzler toplamının ağırlıklı toplamı tarafından yeniden oluşturulabilirse, belki de yüzleri

öğrenmek ve tanımak için verimli bir yöntemin zamanla elde edilen deneyimle, karakteristik özellikleri oluşturmak ve yaklaşık olarak yeniden oluşturulmaları için gerekli olan özellik ağırlıklarını, bilinen kişilerle bağlantılı ağırlıklarla karşılaştırarak, belirli yüzleri tanımak olabileceğini ileri sürdüler. Bu nedenle, her bir bireyi tarif etmek ve yeniden oluşturmak için, gerekli küçük bir özellik veya öz resim ağırlıkları grubu ile karakterize edilir. Bu görüntülerin kendileri ile karşılaştırıldığında son derece kompakt bir sunumdur.

3.2. Önerilen Yüz Tanıma Sisteminin Planı

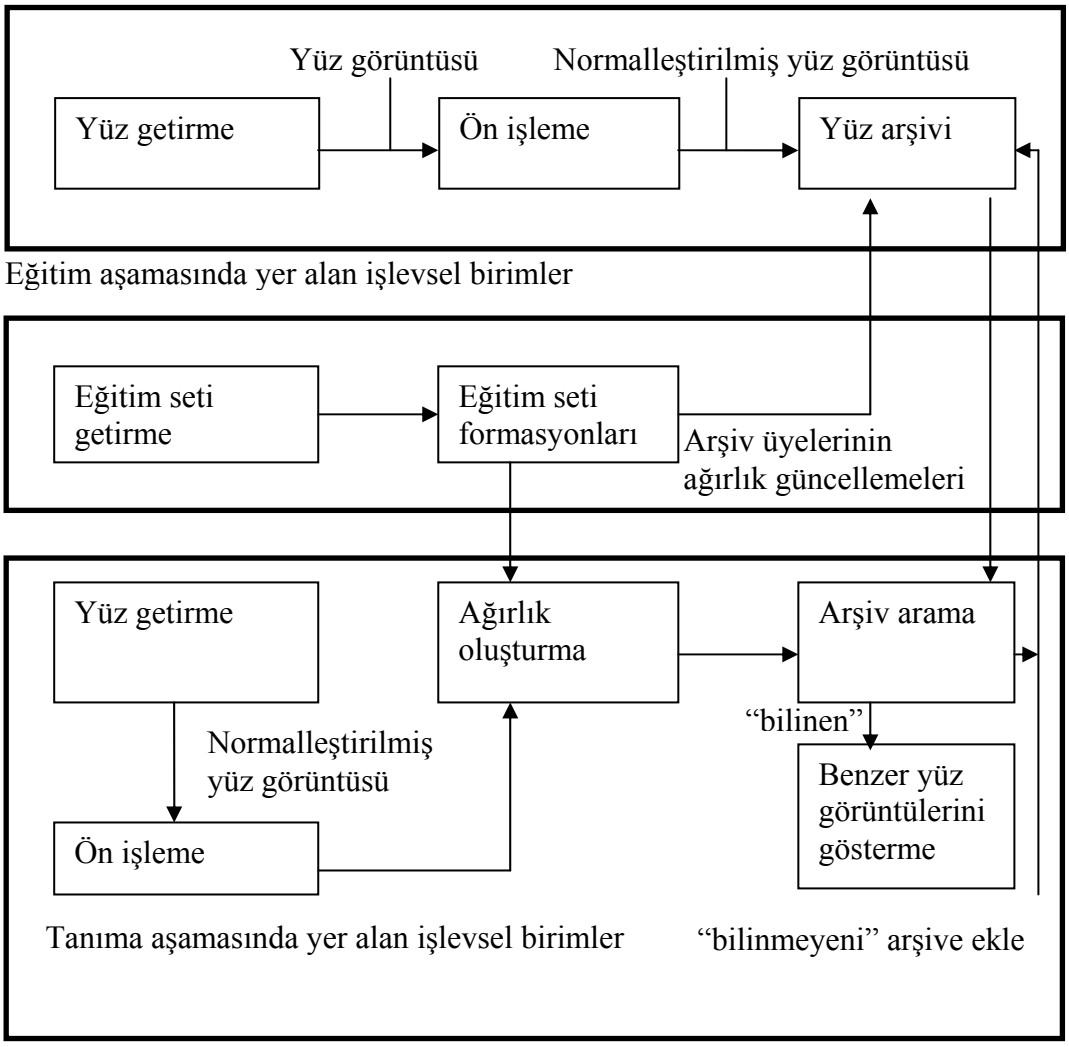
Önerilen yüz tanıma sistemi, bir yüz tanıma işlemi sırasında üç ana aşamadan geçer. Bu aşamalarda üç ana fonksiyonel birim yer alır ve bunlar şekil 3.3. 'de gösterilmiştir. Üç işlevsel birim ile bağlantılı olarak bu aşamaların özellikleri aşağıda verilmiştir:

- Yüz arşivi oluşturma aşaması. Bu aşamada, yüz arşivine eklenecek olan yüz görüntülerinin elde edilmesi ve ön işlemi gerçekleştirilir. Yüz görüntüleri sistemde bir yüz arşivinde kayıtlıdır. Bu yüz veri tabanına "yüz arşivi" denir. Eğitim seti veya özyüz formasyonu gibi her eylem bu yüz arşivinde gerçekleştirilir. Yüz arşivi başlangıçta boştur. Yüz tanıma sürecine başlamak için, bu başlangıçtaki boş yüz arşivinin yüz görüntüleri ile doldurulması gerekir. Şu anda, tarayıcı veya kamera desteği yoktur. Görüntü boyutu dönüşümleri ve yüz görüntülerinde iyileştirmeler gerçekleştirmek için "ön işleme" modülü mevcuttur. Bu modül her yüz görüntüsünü istenilen ölçülere (gerekliyse) getirir. Edinme ve ön işlemeden sonra, söz konusu yüz görüntüsü yüz arşivine eklenir.

- Eğitim aşaması. Yüz görüntülerini başlangıçta boş olan yüz arşivine ekledikten sonra sistem eğitim seti ve yüz formasyonlarını gerçekleştirmeye hazırdır. Eğitim setinde bulunacak olan yüz görüntüleri tüm yüz arşivinden seçilir. Yüz arşivi girişleri normalleştirildiği için bu adımda başka ön işleme gerekli değildir. Eğitim setini seçtikten sonra özyüzler oluşturulur ve daha sonra kullanmak üzere saklanır. Özyüzler eğitim setinden hesaplanır ve sadece en yüksek öz değerlere karşılık gelen M görüntüleri tutulur. Bu M özyüzleri M boyutlu "yüz uzayını" tanımlar. M boyutlu

ağırlık uzayındaki ilgili dağılım, özyüzler tarafından kapsanan "yüz uzayına" yüz görüntüsünü yansıtarak, her yüz arşivi üyesi için hesaplanır. Bu durumda her yüz arşivi üyesinin ilgili ağırlık vektörü başlangıçta boş iken güncellenmiştir. Sistem şu anda tanıma işlemine hazırdır. Bir eğitim seti seçildiğinde, yüz arşivine "birinci aşamada" sunulan klasik metot ile yeni üyeler eklemek mümkün değildir. Çünkü sistem bu maddenin yüz arşivinde zaten bulunup bulunmadığını bilmemektedir. Bir arşiv araması yapılmalıdır.

- Tanıma ve öğrenme aşaması. Bir eğitim setini seçtikten ve yüz arşivi üyelerinin ağırlık vektörlerini oluşturduktan sonra, sistem tanıma işlemi için hazırdır. Kullanıcı bir yüz görüntüsü seçerek tanıma sürecini başlatır. Kullanıcı talebine ve elde edilen görüntü boyutuna bağlı olarak bu elde edilen görüntüyü yüz arşivi özelliklerine normalleştirmek için (gerekliyse) ön işleme adımları uygulanır. Görüntü normalleştiğinde, ağırlık vektörü eğitim aşamasında kaydedilmiş bulunan özyüzlerin yardımı ile oluşturulur. Ağırlık vektörünü elde ettikten sonra, kullanıcı tanımlı bir "eşik" içinde her yüz arşivi üyesinin ağırlık vektörü ile karşılaştırılır. Bu eşik içindeki elde edilen görüntüye benzer en az bir yüz arşivi üyesi varsa, bu durumda yüz görüntüsü "bilinen" olarak sınıflandırılır. Aksi halde bir kayıp gerçekleşmiştir ve yüz görüntüsü "bilinmeyen" olarak sınıflandırılır. Bilinmeyen olarak sınıflandırıldıktan sonra, bu yeni yüz görüntüsü daha sonra kullanmak için (tanımayı öğrenmek) ilgili ağırlık vektörü ile birlikte yüz arşivine eklenebilir.



Şekil 3.3. Önerilen yüz tanıma sisteminin işlevsel blok diyagramı

3.3. Özyüzlerin Hesaplanması

I(x, y) yüz görüntüsünün, 8 bit yoğunluk değerlerine sahip N x N dizi olduğunu düşünelim. Bir görüntü, ayrıca N^2 boyutunun bir vektörü olarak düşünülebilir. Böylece 256x256 boyutundaki tipik bir görüntü 65,536 boyutunda bir vektör veya eşit şekilde 65,536 boyutlu uzayda bir nokta olur. Sonra bir görüntü takımı, bu büyük alanda bir noktalar grubuna eşlenir.

Genel yapılandırmada aynı olan yüz görüntüleri, bu büyük görüntü uzayında rast gele bir şekilde dağıtılmayacaktır ve bu nedenle nispeten düşük boyutlu alt uzayda tarif edilebilir. Temel bileşen analizinin ana fikri (veya Karhunen – Loeve

genişlemesi), tüm görüntü uzayında yüz görüntülerinin dağıtılmasını en iyi açıklayan vektörleri bulmaktır.

Bu vektörler, yüz görüntülerinin alt uzayını tanımlarlar ve buna "yüz uzayı" denir. Her vektör N^2 uzunluğundadır, bir N x N görüntüsünü tarif eder ve orijinal yüz görüntülerinin lineer bir kombinasyonudur. Bu vektörler, orijinal yüz görüntülerine karşılık gelen koveryans matrisinin öz vektörleri olduğundan ve görünüm olarak yüze benzediklerinden bunlara "özyüzler" adı verilir. Bazı özyüz örnekleri şekil 3.2.'de gösterilmiştir.

Tanımlar:
Eğer

$$AX = \lambda X \tag{3.1}$$

ise, bir N x N matris A'nın λ öz değerine karşılık gelen bir X öz vektörüne sahip olduğu söylenir.
Açıktır ki eğer,

$$\det\left|A - \lambda I\right| = 0 \tag{3.2}$$

ise denklem 3.1 geçerli olabilir. Eğer genişletilirse, kökleri öz değerler olan λ içinde N dereceli bir polinom olur. Bu gösterir ki, her zaman N (ayrı olması gerekmez) öz değerleri vardır. Çoklu köklerden gelen eşit öz değerlere "dejenere" denir.
Eğer,

$$A = A^T \text{ veya } a_{ij} = a_{ji} \tag{3.3}$$

devriğine eşitse bir matrise *simetrik* denir.

Eğer devriği evriğine eşitse *dikgen* adını alır.

$$A^T A = AA^T = I \tag{3.4}$$

Son olarak eğer devriği ile değiştirilirse bir gerçek matrise *normal* denir.

$$AA^T = A^T A \tag{3.5}$$

Teorem: Gerçek bir simetrik matrisin öz değerlerin hepsi gerçektir. Tersine gerçek bir simetrik olmayan matrisin öz değerleri gerçek değerler içerebilir, fakat aynı zamanda karmaşık çekim değerleri çiftleri içerebilir. Normal bir matrisin öz değerleri, dejenere olmayan öz değerlerle birlikte tam ve dikgendir ve N boyutlu vektör uzayına dağılır.

Özyüzlerin değerlendirilmesinde kullanılacak olan terimler hakkında bazı açıklamalar verdikten sonra bu yüzlerin bulunması işlemine geçilebilir.

Yüz görüntülerinin eğitim setinin $\Gamma_1\Gamma_2,...,\Gamma_M$ olduğunu düşünelim. Bu durumda setin ortalaması

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n \tag{3.6}$$

(3.6) denkleminde olduğu gibi tanımlanır. Her yüz ortalamadan

$$\Phi_i = \Gamma_i - \Psi \tag{3.7}$$

(3.7) denkleminde gösterilen vektör ile ayrılır.

Bir örnek eğitim seti şekil 3.1.a'da gösterilmektedir ve ortalama yüz ψ şekil 3.1.b'de gösterilmektedir.

Verinin dağılımını en iyi tarif eden bir m ortalama u_n vektörleri setini aramak için, bu çok büyük vektörler seti ana bileşen analizine tabi tutulur. Maksimum olacak şekilde k'nıncı u_k vektörü (3.8) ve (3.9) denklemlerinde olduğu gibi seçilir.

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^T \Phi_n)^2 \tag{3.8}$$

$$u_i^T u_k = \delta_{ik} = \begin{cases} 1, I = k \quad ise \\ 0, \quad aksi\_halde \end{cases} \tag{3.9}$$

Bu u_k vektörleri ve λ_k sayısalları öz vektörler ve öz değerlerdir ve sırasıyla (3.10) denklemindeki koveryans matrisine aittirler.

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n \Phi_n^T = AA^T \tag{3.10}$$

Burada, $A = [\Phi_1 \Phi_2 ... \Phi_M]$ koveryans matrisi, C ise N^2 x N^2 boyutlu gerçek simetrik matris olup, N^2 x N^2 öz vektörlerini ve öz değerlerini belirlemek, tipik görüntü boyutları için kontrol edilemeyen bir görevdir. Bu öz vektörleri bulmak için, sayısal olarak fizibil bir metoda ihtiyacımız vardır.

Eğer görüntü boyutundaki veri noktalarının sayısı uzaydan küçükse ($M < N^2$) sadece M – 1 anlamlı öz vektörleri olacaktır, N^2 olmayacaktır. Kalan öz vektörlerin ilgili öz değerleri olmayacaktır. Bu durumda, N^2 boyutlu öz vektörlerini önce bir M x M matrisinin öz vektörlerini çözüp, örneğin 16,384 x 16,384 matrisinden çok 16 x 16 matrisini çözerek, sonra yüz görüntülerinin Φ_i uygun lineer kombinasyonlarını alarak çözülebilir.

A^TA'nın v_i öz vektörlerini

$$A^T A v_i = \mu_i v_i \tag{3.11}$$

olarak düşünürsek. Her iki tarafı A ile önceden çarparak

$$AA^T Av_i = \mu_i Av_i \tag{3.12}$$

elde edilir. Burada Av_i, $C=AA^T$ 'nin öz vektörleri olduğunu görülür.

Bu analizi takip ederek M X M matris $L=A^TA$ oluşturulur. Burada, $L_{mn} = \Phi_m^T \Phi_n$ ve L'nin M öz vektörleri olan v_i bulunur. Bu vektörler, u_i özyüzlerini oluşturmak için M eğitim seti yüz görüntülerinin lineer kombinasyonlarını belirler.

$$u_i = \sum_{k=1}^{M} v_{ik} \Phi_k, \qquad I = 1,...,M \tag{3.13}$$

Bu analizle hesaplamalar büyük ölçüde azaltılır. Pratikte yüz görüntülerinin eğitim seti nispeten küçük olacaktır ($M<<N^2$) ve hesaplamalar oldukça kolay hale gelecektir. İlgili öz değerler, görüntüler arasındaki değişikliği karakterize etmede ki faydalarına göre, öz vektörleri sıralama imkanı verir.

Bu algoritmanın başarısı, görüntülerinin eğitim setinden oluşturulan gerçek simetrik matris L'nin öz değerlerin ve öz vektörlerin değerlendirilmesine dayalıdır. Karakteristik denklem (3.2)'de kök arama, öz değerlerin bulunması için genellikle oldukça basit bir hesaplama yöntemidir. Yukarıdaki algoritmanın programlama aşamasında, öz değerleri ve öz vektörleri değerlendirmek için daha verimli bir metot [31] kullanılmıştır. Başlangıçta, gerçek simetrik matris "householder" algoritmasının yardımı ile üç köşegenli şekle dönüştürülür. Householder algoritması, N-2 dikgen transformasyonlarla bir N x N simetrik A matrisini bir üç köşegenli şekle indirger. Her transformasyon tüm sütunun ve tüm ilgili sıranın ilgili kısmını sıfırlar. Bundan sonra, öz değerler ve öz vektörler QR transformasyonlarının yardımı ile elde edilir. QR algoritmasının arkasındaki temel fikir, her hangi bir gerçek simetrik matrisin A=QR şeklinde ayrıştırılabilmesidir. Burada Q dikgen ve R üst üçgendir. QR algoritmasındaki iş yükü, yasaklayıcı olan genel bir matrisin her tekrarı için $O(N^3)$

dir. Ancak, bunu oldukça verimli hale getiren üç köşegenli bir matris için her tekrarda iş yükü sadece O(N) dir.

3.4. Bir Yüz Görüntüsünü Sınıflandırmak İçin Özyüzlerin Kullanılması

L öz vektörlerinden hesaplanan özyüz görüntüleri, yüz görüntülerini tarif etmek üzere temel bir set oluşturur. Sirovich ve Kirby, kontrollü bir tarzda sayısallaştırılan, Kafkas erkeklerinin M=115 görüntü takımında, bu çerçevenin sınırlı bir versiyonu değerlendirdiler ve yüz görüntülerinin çok iyi bir tarifi için, 40 özyüzün yeterli olduğunu buldular. M'=40 özyüz ile, yüz görüntülerinin kesilmiş versiyonlarını temsil etmede, RMS piksel hataları yaklaşık %2 dir.

Bu çerçevede tanımlama bir görüntü tanıma görevi haline gelir. Özyüzler, orijinal N^2 görüntü uzayının bir M' boyutlu alt uzayını dağıtır. L matrisinin M' önemli öz vektörleri, en büyük ilgili öz değerlere sahip olanlar olarak seçilirler. Yeni bir yüz görüntüsü (Γ), basit bir operasyonla özyüz bileşenlerine dönüştürülür ("yüz uzayı"na yansıtılır).

$$w_k = u_k^T(\Gamma - \Psi) \tag{3.14}$$

K=1,…..,M' olduğu durum için $O(N^4)$ hesap karmaşıklığı ile birlikte nokta nokta görüntü çoğaltmaları ve toplanmaları setini, mevcut görüntü işlem donanımında yaklaşık görüntü hızında gerçekleştirilen işlemleri tarif eder.

Bu bir özellik vektöründen ortaya çıkar.

$$\Omega^T = \left[w_1 w_2 ... w_M\right] \tag{3.15}$$

Bu vektör giriş yüzü görüntüsünün temsil edilmesinde her özyüzün katkısını yüz görüntüleri için belirlenmiş bir temel set olarak özyüzleri ele almak suretiyle tarif eder. Daha sonra, öz nitelik vektörü, önceden belirlenmiş yüz sınıflarından hangisinin yüzü daha iyi tarif ettiğini bulmak için, standart bir görüntü tanıma

algoritmasında kullanılır. Yüz sınıfları Ω_i, özyüz sunumları sonuçlarını her bireyin az sayıdaki yüz görüntüsü (1 adet olabilir) üzerinden ortalayarak hesaplanabilir. Önerilen yüz tanıma sisteminde, yüz sınıfları her bireyin sadece bir temsilini içerir. Sınıflandırma, yüz arşiv üyelerinin öz nitelik vektörlerini giriş yüzü görüntüsünün öz nitelik vektörü ile karşılaştırarak yapılır. Bu karşılaştırma, kullanıcı tanımlı eşikten ε_k daha küçük olması için iki üye arasındaki öklid mesafesine dayalıdır. Bu denklem (3.16)'da verilmiştir. Eğer karşılaştırma kullanıcı tanımlı eşik içinde gerçekleşirse sonra yüz görüntüsü "bilinen" olarak sınıflanır. Eğer eşik dışında gerçekleşirse "bilinmeyen" olarak sınıflandırılır ve daha sonraki kullanımlar için öz nitelik vektörü ile birlikte yüz arşivine eklenebilir ve sistemin yeni yüz görüntülerini öğrenmesi sağlanabilir.

$$\frac{\left\|\Omega - \Omega_k\right\|}{\left\|\Omega_k\right\|} \leq \varepsilon_k \tag{3.16}$$

3.5. Özyüzlerle Bir Yüz Görüntüsünün Yeniden Oluşturulması

Bir yüz görüntüsü öz nitelik vektörünü

$$\Gamma' = \Psi + \Phi_i \tag{3.17}$$

(3.17) denkleminde olduğu gibi özyüzlerini kullanarak yaklaşık şekilde yeniden oluşturulabilir. Burada,

$$\Phi_f = \sum_{i=1}^{M'} w_i u_i \tag{3.18}$$

yansıtılan görüntüdür.

(3.17) denklemi, söz konusu yüz görüntüsünün sadece w_i 'nin katkısına sahip her özyüzü, eğitim seti görüntülerinin ortalamasına ekleyerek, yeniden yapıldığını gösterir. Uyum derecesi veya "yeniden oluşturma hata oranı", denklem (3.19)

verildiği gibi orijinal ve yeniden yapılan yüz görüntüsü arasındaki öklid mesafesi aracılığıyla ifade edilebilir.

$$\text{yeniden oluşturma oranı} = \frac{\left\|\Gamma' - \Gamma\right\|}{\left\|\Gamma\right\|} \tag{3.19}$$

Yeniden oluşturma hata oranının, eğitim seti üyelerinin birbirlerinden yoğun bir şekilde farklılık gösterirken arttığı gözlenmiştir. Bunun nedeni, ortalama yüz görüntüsünün eklenmesidir. Üyeler birbirinden farklıyken (özellikle görüntü arka planında) ortalama yüz görüntüsü daha karmaşık hale gelir ve bu yeniden oluşturma hata oranını artırır.

Bir giriş görüntüsü ve görüntü vektörü için 4 olasılık vardır:
1. Yüz uzayına yakın ve bir yüz sınıfına yakın.
2. Yüz uzayına yakın fakat bilinen bir yüz sınıfına yakın değil.
3. Yüz uzayından uzak ve bir yüz sınıfına yakın.
4. Yüz uzayından uzak ve bilinen bir yüz sınıfına yakın değil.

Birinci durumda, bir birey tanınır ve kimliği belirlenir. İkinci durumda, bilinmeyen bir kişi gösterilir. Son iki durum, görüntünün bir yüz görüntüsü olmadığını gösterir. Üçüncü durum tipik olarak yanlış bir sınıflandırma gösterir. Bu sistemdeki yanlış sınıflandırmadan (3.18) denklemi (3.20) denklemindeki gibi kullanılarak kaçınmak mümkündür.

$$\frac{\left\|\Phi - \Phi_f\right\|}{\left\|\Phi_f\right\|} \leq \Phi_k \tag{3.20}$$

Burada Φ_k k yüz sınıfına ait giriş yüzü görüntülerinin yüzleri için kullanıcı tanımlı bir eşiktir.

3.6. Özyüz Tanıma Prosedürünün Özeti

Yüz tanıma için özyüzler yaklaşımı aşağıdaki adımlarda özetlenebilir:

- Bilinen bireylerin yüz görüntülerinden oluşan bir yüz arşivi oluşturun.

- İfadede ve aydınlatmada bazı değişikliklerle her kişi için belli bir miktarda görüntü (M) içeren bir eğitim seti seçin

- M x M boyutlu L matrisini hesaplayın, öz vektörlerini ve öz değerlerini bulun ve en yüksek ilgili öz değerleri ile M' öz vektörlerini seçin.

- M' özyüzlerini oluşturmak için Denklem (3.13)'e göre normalleştirilmiş görüntüler eğitim setini birleştirin. Bu özyüzleri daha sonra kullanmak üzere saklayın.

- Yüz arşivindeki her üye için denklem (3.15)'e göre bir öz nitelik vektörünü hesaplayın ve kaydedin.

- Her hangi bir yüz sınıfından izin verilen maksimum mesafeyi tanımlayan denklem (3.16)'a göre bir eşik ε seçin. İsteğe bağlı olarak yüz uzayından izin verilen maksimum mesafeyi tanımlayan denklem (3.20)'ye göre bir eşik Φ seçin.

- Tanımlanacak her yeni yüz görüntüsü için, denklem (3.15)'e göre öz nitelik vektörünü hesaplayın ve yüz arşivi üyelerinin kaydedilmiş öz nitelik vektörleri ile karşılaştırın. Eğer karşılaştırma denklem (3.16)'da verilen şartı en az bir üye için karşılarsa, bu durumda bu yüz görüntüsünü "bilinen" olarak sınıflayın, aksi halde bir kayıp meydana gelir ve bunu "bilinmeyen" olarak sınıflandırın ve öz nitelik vektörü ile yüz arşivine ekleyin.

3.7. Özyüzler Yaklaşımının Özellik Tabanlı Yüz Tanımı İle Karşılaştırılması

Bu iki farklı yaklaşım yüz tanımanın aşağıdaki yönlerine göre karşılaştırılmıştır:

- Hız ve basitlik. Özellik tabanlı yüz tanıma, şekil değiştirilebilen şablonlar ve aktif kontur modelleri gibi karmaşık hesaplamalar içerir. Bu parametrelerin değerlendirilmesi, bu günkü bilgisayarlarda bile çok zaman alır. Özyüzler yaklaşımı, hızı ve makul şekilde basit uygulaması ile üstündür. Denklem (3.14)'de bir öz nitelik vektörünün değerlendirmesinin sadece ilaveler ve çoğaltmalardan oluştuğu görülmüştür.

- Öğrenme becerisi. Özellik tabanlı yüz tanıma sistemi, genel olarak denetimli bir şekilde parametrelerini optimize etmek üzere eğitilir. Yani sistem tasarımcısı, bilinen bireyleri sisteme sunar ve sistem tepkisini kontrol eder. Özyüzler yaklaşımında, eğitim denetimsiz bir tarzda yapılır. Kullanıcı yüz görüntülerinin kalanını temsil eden bir eğitim seti seçer. Özyüzler, eğitim seti üyelerinden elde edilir ve öz nitelik vektörleri oluşturulur.

- Yüz arka planı. Yüz görüntülerinin arka planı, özyüz yaklaşımında son derece önemlidir. Özyüzler ve öz nitelik vektörleri görüntü çoğaltma ve eklemelerle değerlendirilir. Bunun bir sonucu olarak, yüz görüntüsünde bulunan tüm görüntü kullanılır. Yüzün arka planı nedeniyle bu bilgi değişirse, tanıma performansı büyük ölçüde azalır. Bundan kaçınmak için, bir ön işleme adımında "arka plan kaldırma" algoritması uygulanabilir. Özellik tabanlı yüz tanıma algoritmaları, yüz arka planına daha az hassas olabilir ve genel olarak yüz özelliklerini çıkarmak için yüz konturlarının yerini belirlerler.

- Ölçek ve oryantasyon. Giriş görüntüsündeki kafa boyutu ve oryantasyon sistemin iyi çalışması için özyüzlerinkine yakın olmalıdır. Bu problemi aşmak için, çok ölçekli özyüzler kullanılabilir veya kafa görüntünün geri kalanından çıkartılabilir. Sonra özyüzlerin özelliklerini karşılamak için ölçeklenir ve döndürülür. Burada da özellik tabanlı yüz tanıma algoritmaları bu karşılaştırmada daha iyi not alabilir. Ölçek ve oryantasyona daha az hassas olan şekil değiştirebilen şablonlar ve aktif kontur modelleri kullanarak yüz özelliklerini bulmaktadırlar.

- Küçük ayrıntıların varlığı. Özellik tabanlı yüz tanıma algoritmaları, koyu gözlükler veya sakal gibi yüz görüntüsünde bazı detaylar bulunduğunda sorun yaşayabilir. Bir

özellik tabanlı yüz tanıma sistemi için, yüzde koyu bir gözlük varken göz ile ilgili özellikleri çıkarmak neredeyse imkansızdır. Ayrıca aktif kontur modelleri, yüz konturunu belirlerken yüzde sakal olması durumunda sorun yaşayabilir. Özyüzler yaklaşımı bu yönde çok iyidir. Gözlükler, sakal veya bıyık gibi yüz görüntülerindeki küçük değişiklikler yüz tanıma performansında bir azalmaya neden olmaz. Yüz görüntüsünün geri kalan kısmında mevcut bulunan bilgi doğru sınıflandırma için yeterlidir.

BÖLÜM 4. YÜZ TANIMA PROGRAMI

Bu bölümde, önerilen özyüz yöntemi kullanılarak geliştirilen basit bir yüz tanıma programı tanıtılacaktır.

Program Microsoft Visual Studio 2003 kullanılarak C# dilinde geliştirilmiştir. Ayrıca bazı rutinler Visual Basic 6.0 programlama dili kullanılarak yapılmıştır. Raporlama aracı olarakta ActiveReport kullanılmıştır.

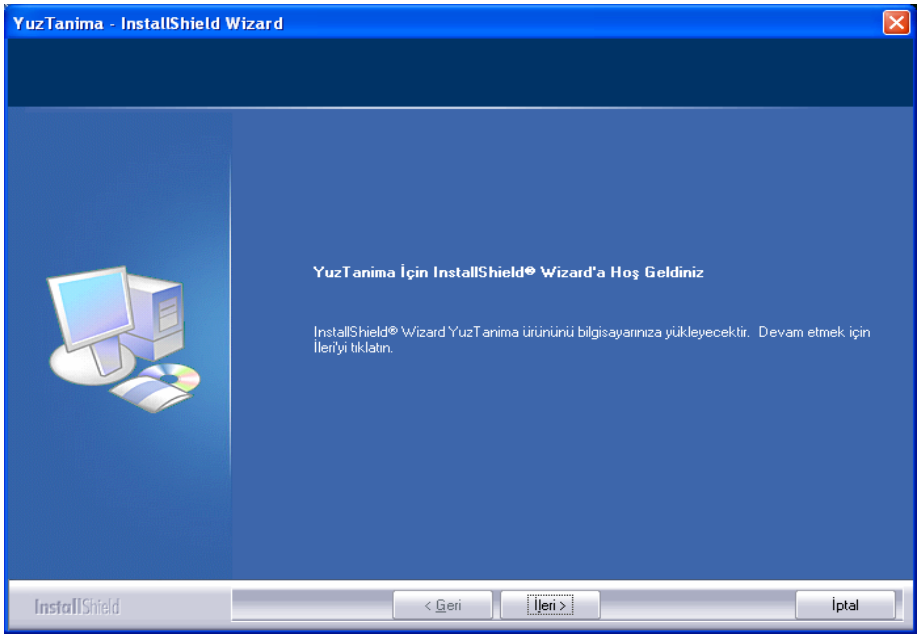
4.1. Sistem İhtiyaçları

Geliştirilen programın çalışabilmesi için gerekli olan sistem aşağıdaki gibidir.

- İşletim Sistemi: Ms Windows 2000 veya üzeri
- Ram : 256 veya üzeri
- Ekran Bağdaştırıcısı : 32 MB veya üzeri hafızası olan ekran kartı
- Klavye
- Fare
- Intel Pentium 3 veya eşdeğer seviyede bir işlemci

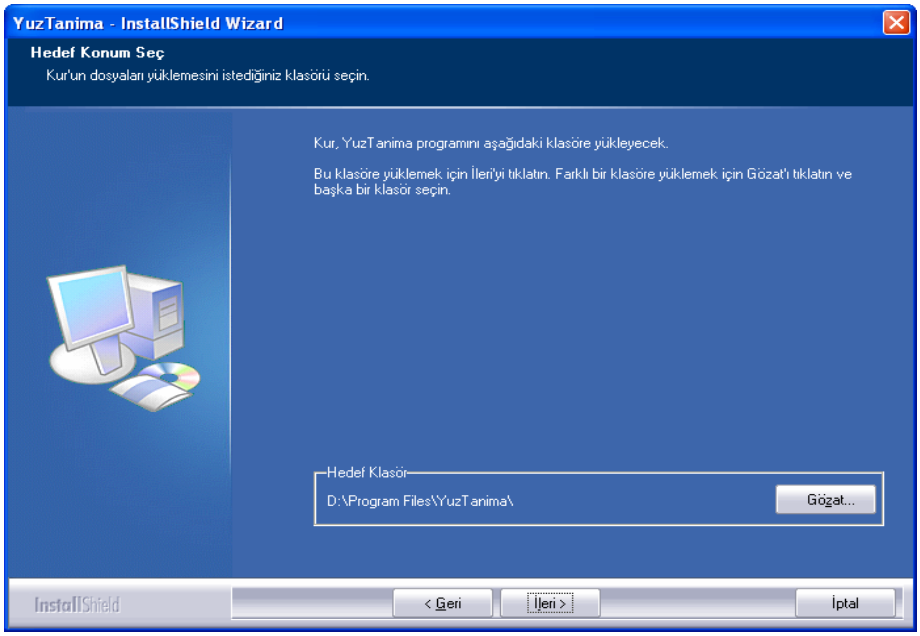
4.2. Kurulum İşlemleri

YüzTanıma yazılımının kurulması işlemi oldukça basittir. Kurulum dosyalarından Setup.exe dosyası çalıştırılır. Kurulum için gerekli işlemleri otomatik olarak gerçekleştirdikten sonra şekil 4.1. görüldüğü gibi bir ekran gelecektir. Bu ekranda İleri butonuna basılarak kurulum işlemi için sonraki adıma geçilebilir. İptal butonu kullanılarakta kurulum işleminden vazgeçilebilir.

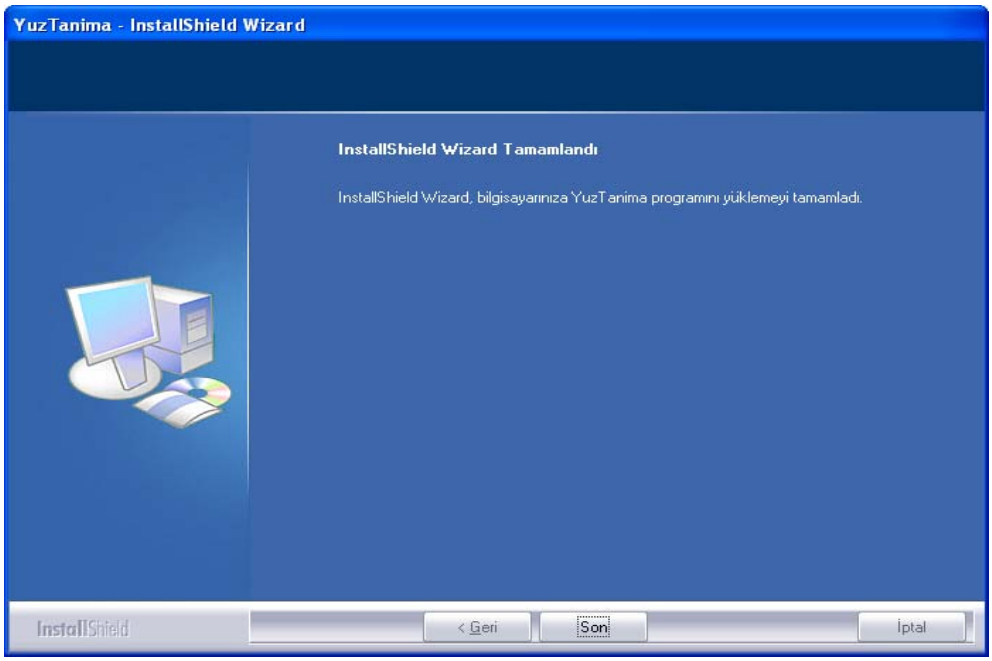


Şekil 4.1. YüzTanıma kurulum işlemi ilk ekran

Daha sonra şekil 4.2. görüldüğü gibi kurulum işleminin bilgisayarınızda nereye yapılacağını seçmek için bir ekran gelecektir. Bu ekranda Gözat butonunu kullanarak bilgisayarınızda ki herhangi bir klasörü hedef olarak belirtebilirsiniz. Eğer her hangi bir yer belirtilmezse setup programı varsayılan olarak Program Files\YuzTanima olarak kabul edecektir. Hedef klasörü belirlendikten sonra İleri butonu ile kurulum işlemine devam edilebilir veya İptal butonu ile kurulum işleminden vazgeçilebilir. Kurulum işleminde İleri butonuna tıklanıldığında setup programı YuzTanima programı için gerekli olan program dosyalarını ve örnek bir veritabanı yükleyecektir. Gerekli dosyalar sisteminize kopyalandıktan sonra şekil 4.3. görüldüğü gibi kurulum işlemi tamamlama ekranı gelecektir. Bu ekranda Son butonuna basılarak işlem tamamlanır.



Şekil 4.2. Kurulum işlemi hedef klasörü seçimi



Şekil 4.3. Kurulum işlemi tamamlanması

4.3. Programın Tanıtılması

Program şekil 4.4. görüldüğü gibi sadece bir ana ekrandan oluşur. Bu ekran kullanılarak tüm işlemler yapılmaktadır.

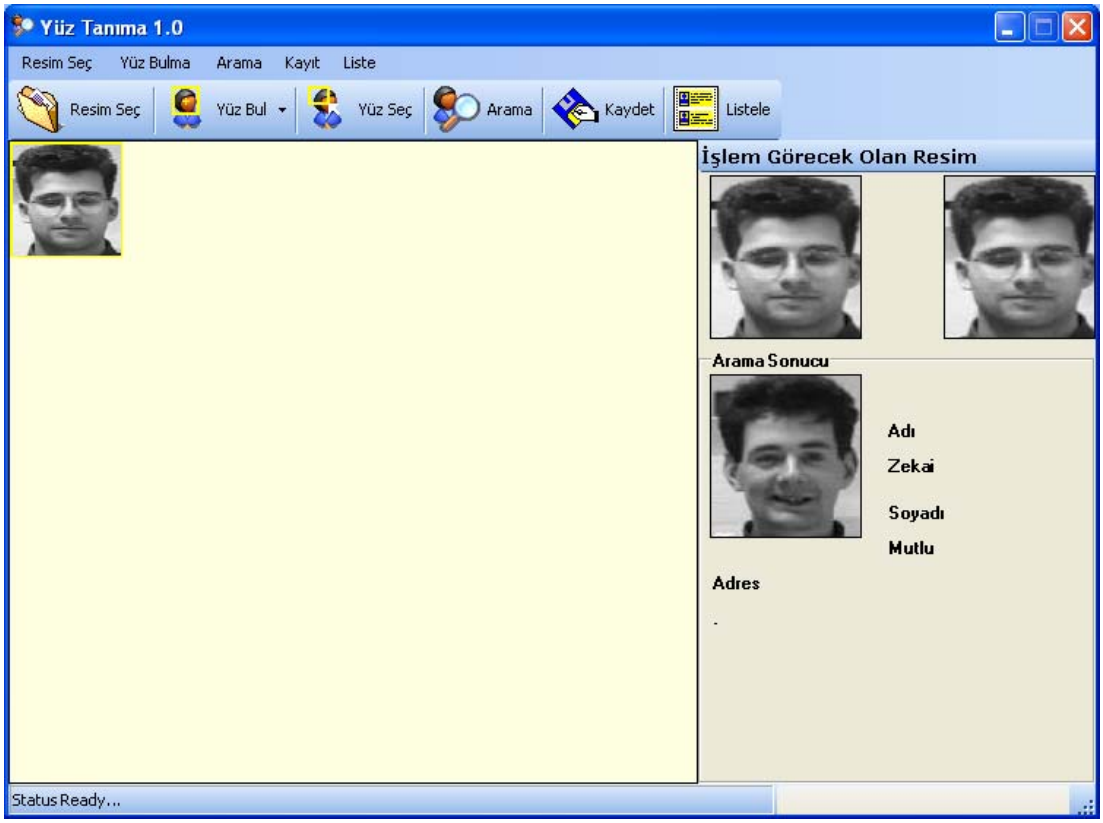
Bir yüzün aranması işlemi bir kaç adımdan oluşmaktadır.

1. Resim Seç menüsünü kullanarak aranmasını istediğimiz resim dosyasını seçeriz.

2. Yüz Bul menüsünü kullanarak seçilen resmin içindeki yüzün otomatik olarak tespit edilmesi sağlanabilir. Yüz bulma işlemi başarısız olursa resmin tamamını işaretleyecektir. Yüz bulma menüsünün altındaki Ayrıntıları Göster menüsü işaretlenirse yüz tespit işlemi sırasında resme uygulanan işlemin son halini görebilirsiniz.

3. Eğer yüz bulma işlemi başarısız olursa Yüz Seç menüsünü kullanarak elle resimdeki yüz bölgesininin belirtilmesi gerekir. Yüz Seç menüsüne tıkladıktan sonra fare ile sol üst köşeden başlayarak sağ alt köşeye kadar işaretleme işlemi yapılmalıdır.

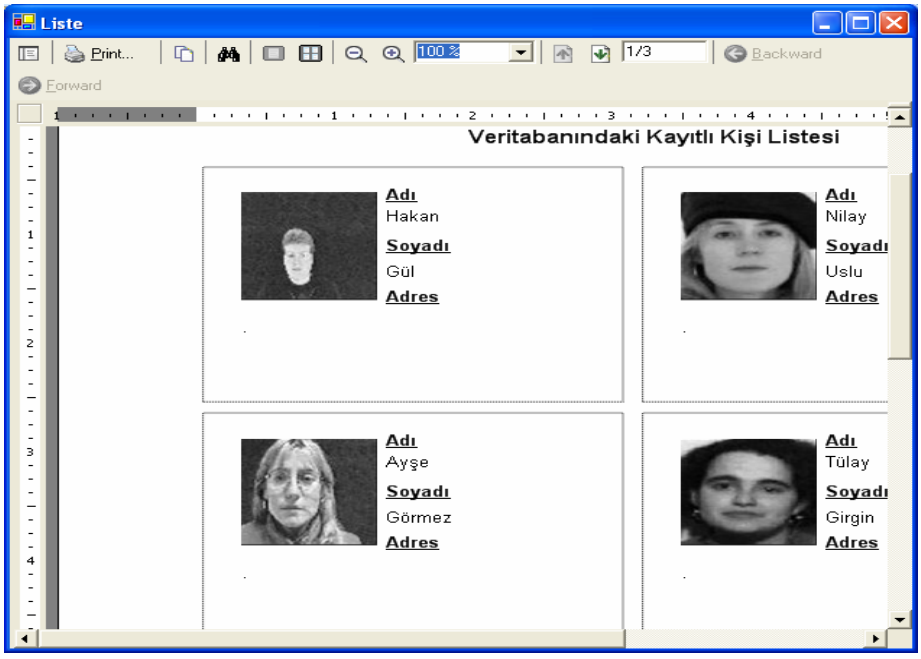
4. Yüz seçme işlemi tamamlandıktan sonra Arama menüsüne tıklayarak seçilen resmin veritabanında bulunan diğer resimler ile karşılaştırılması yapılabilir. Seçilen resim veritabanında kayıtlı değilse alakasız bir resim bulacaktır. İstenildiği taktirde, seçilen resim daha sonraki aramalarda kullanılmak üzere Kaydet menüsü kullanarak kaydedilmesi sağlanabilir. Şekil 4.5. de gösterildiği gibi kaydet menüsünde resim kaydedilirken ilgili şahsın adı, soyadı ve adreside kaydedilebilmektedir.



Şekil 4.4. Yüz tanıma programı ana ekranı

Listele menüsü kullanılarak veritabanında kayıtlı olan tüm resimler alınabilmektedir. Şekil 4.6. te görüldüğü gibi ön izleme ekranında gösterilerek, ihtiyaç duyulduğunda doğrudan yazıcıya gönderilebilmektedir.

Şekil 4.5. Yeni resim ekle



Şekil 4.6. Veritabanı resim listesi

4.4. Programda Kullanılan Önemli Fonksiyonlar

4.4.1. Renkli bir resimden yüz bölgesinin bulunması

```
bool isSkinColor(Color cColor)
{
        float g, b, r;
        if(cColor.R==0)
        {
                r = (float) cColor.R;
                g=0;
                b=0;
        }
        else
        {
                r = (float) cColor.R;
                g = (float) (cColor.G  / (float)cColor.R);
                b = (float) (cColor.B  / (float)cColor.R);
        }
        if ( (r > 90) && (g  < .9) && (g > .2) && (b < .8)
        && (b > .2))
                return true;
        else
                return false;
}



private void YuzBul()
{
        if(bmpSelectedBitmap==null)
```

```
{
        MessageBox.Show("Önce Bir Resim
        Seçmelisiniz.","Dikkat",MessageBoxButtons.OK,MessageBoxIcon.W
        arning);
        return;
}
Bitmap bmp=new Bitmap(bmpSelectedBitmap);
Bitmap newbmp=new Bitmap(bmp.Width,bmp.Height);
#region Yüz Kısmını Siyah Gerisini Beyaz Yap
for(int x=0;x<bmp.Width;x++)
{
        for(int y=0;y<bmp.Height;y++)
        {
                Color r=bmp.GetPixel(x,y);
                if(isSkinColor(r)==true)
                {
                        newbmp.SetPixel(x,y,Color.Black);
                }
                else
                {
                        newbmp.SetPixel(x,y,Color.White);
                }
        }
}
#endregion

#region En Yoğun  Sütunu Bul
//Yatayda En yoğun Çizgiyi bul
int intMaxX=0;
int intMaxLength=0;
for(int x=0;x<bmp.Width;x++)
{
        int intMax=0;
```

```
            for(int y=0;y<bmp.Height;y++)
            {
                Color r=newbmp.GetPixel(x,y);
                if(r.R==0 && r.G ==0 && r.B==0)
                {
                    intMax++;
                }
            }
            if(intMaxLength<intMax)
            {
                intMaxLength=intMax;
                intMaxX=x;
            }
        }
        #endregion

        #region En Yoğun Satırı Bul
        int intMaxY=0;
        intMaxLength=0;
        for(int y=0;y<bmp.Height;y++)
        {
            int intMax=0;
            for(int x=0;x<bmp.Width;x++)
            {
                Color r=newbmp.GetPixel(x,y);
                if(r.R==0 && r.G ==0 && r.B==0)
                {
                    intMax++;
                }
            }
            if(intMaxLength<intMax)
            {
                intMaxLength=intMax;
```

```
                intMaxY=y;
            }
        }
        #endregion

        int intX1=intMaxX;
        for(int x=intMaxX;x>0;x--)
        {
            Color r=newbmp.GetPixel(x,intMaxY);
            if(r.R==0 && r.G ==0 && r.B==0)
            {
                intX1=x;
            }
            else
            {
                break;
            }
        }
        int intX2=intMaxX;
        for(int x=intMaxX;x<bmp.Width;x++)
        {
            Color r=newbmp.GetPixel(x,intMaxY);
            if(r.R==0 && r.G ==0 && r.B==0)
            {
                intX2=x;
            }
            else
            {
                break;
            }
        }
        int intY1=intMaxY;
        for(int y=intMaxY;y>0;y--)
```

```
{
        Color r=newbmp.GetPixel(intMaxX,y);
        if(r.R==0 && r.G ==0 && r.B==0)
        {
                intY1=y;
        }
        else
        {
                break;
        }
}
int intY2=intMaxY;
for(int y=intMaxY;y<bmp.Height;y++)
{
        Color r=newbmp.GetPixel(intMaxX,y);
        if(r.R==0 && r.G ==0 && r.B==0)
        {
                intY2=y;
        }
        else
        {
                break;
        }
}
int intFaceWidth=intX2-intX1;
int intFaceHeight=intY2-intY1;
intX1=intX1- (intFaceWidth/2);
if(intX1<0)
        intX1=0;
if((intFaceWidth+intX1)>bmp.Width)
        intFaceWidth=bmp.Width- intX1;

intFaceWidth=intFaceWidth+(intFaceWidth/2);
```

```
intY1=intY1-(intFaceHeight/2);
intFaceHeight=intFaceHeight+(intFaceHeight/2);
if(intY1<0)
        intY1=0;
if((intFaceHeight+intY1)>bmp.Height)
        intFaceHeight=bmp.Height- intY1;
if(intFaceWidth<20 || intFaceHeight<20)
{
        intX1=1;
        intY1=1;
        intFaceWidth=bmp.Width-1;
        intFaceHeight=bmp.Height-1;
}
DevComponents.DotNetBar.ButtonItem item=
(DevComponents.DotNetBar.ButtonItem)
dotNetBarManager.GetItem("tlbMenuDetay",true);
if(item.Checked==true)
{
        Graphics G=Graphics.FromImage(newbmp);
        G.DrawRectangle(new Pen(Color.Yellow,2)
        ,new Rectangle(intX1,intY1,intFaceWidth,intFaceHeight));
        rectFaceRectangle=new
        Rectangle(intX1,intY1,intFaceWidth,intFaceHeight);
                        picResim.Image=newbmp;
                        picResim.Refresh();
}
else
{
        Graphics G=Graphics.FromImage(bmp);
        G.DrawRectangle(new Pen(Color.Yellow,2),new
        Rectangle(intX1,intY1,intFaceWidth,intFaceHeight));
        rectFaceRectangle=new
        Rectangle(intX1,intY1,intFaceWidth,intFaceHeight);
```

```
                picResim.Image=bmp;
                picResim.Refresh();
        }
        YuzAyirveGrayScale();
}
```

4.4.2. Resim işlemek için gerekli kütüphane

```
Public width As Integer
Public height As Integer
Dim image() As Byte

Dim edgeTraced() As Boolean
Dim temp() As Boolean
Public TraceEdgesThresh As Integer
Public minEdgeLength As Integer
Dim traceDirection As Single
Dim traceRadius As Integer
Dim traceX As Single
Dim traceY As Single

Dim angleHistogram(18) As Integer

Public edgesWidth As Integer
Public edgesHeight As Integer
Dim Edges() As Byte

Public processType As Integer

Public EdgeThreshold As Single
Dim averageContrast As Double
```

```
Const IMAGE_RAW = 0
Const IMAGE_RED = 1
Const IMAGE_GREEN = 2
Const IMAGE_BLUE = 3
Const IMAGE_EDGES = 4
Const IMAGE_MOVEMENT = 5


'masks used for edge detection
Const NO_OF_EDGE_MASKS = 14
Dim EdgeMask(NO_OF_EDGE_MASKS)
Const NO_OF_EDGE_TYPES = 5
Dim EdgeHistogram(NO_OF_EDGE_TYPES) As Integer

Const EDGE_VECTOR_LENGTH = 200
Dim EdgeVector(5, EDGE_VECTOR_LENGTH) As Single
Dim currEdgeVector As Integer
Dim maxEdgeVectorIntensity As Integer



Private Function traceSearch(Optional beginSearch As Boolean) As Boolean
'move the trace point in a curcular motion until a new feature is found
'returns TRUE when a new feature is located

  Dim tx As Integer
  Dim ty As Integer

  traceSearch = False

  If (beginSearch) Then
    traceDirection = 0
```

```
  traceRadius = 90
End If

traceX = traceX + Cos((traceDirection / 180) * 3.14)
traceY = traceY + Sin((traceDirection / 180) * 3.14)
traceDirection = traceDirection + traceRadius
If (traceDirection > 360) Then
  traceDirection = 0
  traceRadius = traceRadius - 1
  If (traceRadius < 0) Then
    traceRadius = 0
  End If
End If

If (traceX < 0) Then
  traceX = 0
End If
If (traceX >= width) Then
  traceX = width - 1
End If
If (traceY < 0) Then
  traceY = 0
End If
If (traceY >= height) Then
  traceY = height - 1
End If

tx = Int(traceX)
ty = Int(traceY)

If ((image(tx, ty) > TraceEdgesThresh) And (Not edgeTraced(tx, ty))) Then
  traceSearch = True
End If
```

```
End Function


Private Sub calcAngleHistogram()
'calculates a histogram from the angles of edge traces
  Dim i As Integer
  Dim dx As Integer
  Dim dy As Integer
  Dim length As Integer
  Dim angle As Single
  Dim intensity As Single


  For i = 0 To 17
    angleHistogram(i) = 0
  Next

  For i = 0 To currEdgeVector - 1
    dx = EdgeVector(0, i) - EdgeVector(2, i)
    dy = Abs(EdgeVector(1, i) - EdgeVector(3, i))
    length = Sqr((dx * dx) + (dy * dy))
    If (length > 0) Then
      angle = (Acos(dy / length) / 3.14) * 180
      If (dx < 0) Then
        angle = 180 - angle
      End If
      angle = Int(angle / 10)
      intensity = 1 'EdgeVector(4, i) / 255
      angleHistogram(angle) = angleHistogram(angle) + (length * intensity)
    End If
  Next

End Sub
```

```
Private Sub initEdgeMasks()
'defines edge masks
' 1 = horizontal
' 2 = vertical
' 3 = diagonal left
' 4 = diagonal right
' 5 = cross

  Dim mask
  Dim i As Integer
  Dim mstr As String

  'Lines -
  EdgeMask(0) = Array(1, 1, 1, _
                      0, 0, 0, _
                      0, 0, 0, _
                      1)
  EdgeMask(1) = Array(0, 0, 0, _
                      1, 1, 1, _
                      0, 0, 0, _
                      1)
  EdgeMask(2) = Array(0, 0, 0, _
                      0, 0, 0, _
                      1, 1, 1, _
                      1)
  'Lines |
  EdgeMask(3) = Array(1, 0, 0, _
                      1, 0, 0, _
                      1, 0, 0, _
                      2)
  EdgeMask(4) = Array(0, 1, 0, _
```

```
                0, 1, 0, _
                0, 1, 0, _
                2)
EdgeMask(5) = Array(0, 0, 1, _
                0, 0, 1, _
                0, 0, 1, _
                2)
'Diagonals
EdgeMask(6) = Array(0, 0, 1, _
                0, 1, 0, _
                1, 0, 0, _
                3)
EdgeMask(7) = Array(0, 1, 0, _
                1, 0, 0, _
                0, 0, 0, _
                3)
EdgeMask(8) = Array(0, 0, 0, _
                0, 0, 1, _
                0, 1, 0, _
                3)
EdgeMask(9) = Array(1, 0, 0, _
                0, 1, 0, _
                0, 0, 1, _
                4)
EdgeMask(10) = Array(0, 1, 0, _
                0, 0, 1, _
                0, 0, 0, _
                4)
EdgeMask(11) = Array(0, 0, 0, _
                1, 0, 0, _
                0, 1, 0, _
                4)
'Crosses
```

```
  EdgeMask(12) = Array(1, 0, 1, _
                  0, 1, 0, _
                  1, 0, 1, _
                  5)
  EdgeMask(13) = Array(0, 1, 0, _
                  1, 1, 1, _
                  0, 1, 0, _
                  5)
  'nothing
  'EdgeMask(14) = Array(0, 0, 0, _
  '                 0, 0, 0, _
  '                 0, 0, 0, _
  '                 0) 'last number indicates edge type
  'EdgeMask(15) = Array(1, 1, 1, _
  '                 1, 1, 1, _
  '                 1, 1, 1, _
  '                 0)
End Sub


Private Sub initEdgeMasks_old()
'defines edge masks
' 0 = horizontal
' 1 = vertical
' 2 = diagonal left
' 3 = diagonal right
' 4 = cross

  Dim mask
  Dim i As Integer
  Dim mstr As String

  'Lines -
```

```
EdgeMask(0) = Array(1, 1, 1, _
                0, 0, 0, _
                0, 0, 0, _
                1)
EdgeMask(1) = Array(0, 0, 0, _
                1, 1, 1, _
                0, 0, 0, _
                1)
EdgeMask(2) = Array(0, 0, 0, _
                0, 0, 0, _
                1, 1, 1, _
                1)
'Lines double -
EdgeMask(3) = Array(1, 1, 1, _
                1, 1, 1, _
                0, 0, 0, _
                1)
EdgeMask(4) = Array(0, 0, 0, _
                1, 1, 1, _
                1, 1, 1, _
                1)
'Lines |
EdgeMask(5) = Array(1, 0, 0, _
                1, 0, 0, _
                1, 0, 0, _
                2)
EdgeMask(6) = Array(0, 1, 0, _
                0, 1, 0, _
                0, 1, 0, _
                2)
EdgeMask(7) = Array(0, 0, 1, _
                0, 0, 1, _
                0, 0, 1, _
```

```
                2)
EdgeMask(8) = Array(1, 1, 0, _
                1, 1, 0, _
                1, 1, 0, _
                2)
EdgeMask(9) = Array(0, 1, 1, _
                0, 1, 1, _
                0, 1, 1, _
                2)
'Diagonals
EdgeMask(10) = Array(0, 0, 1, _
                0, 1, 0, _
                1, 0, 0, _
                3)
EdgeMask(11) = Array(0, 0, 1, _
                0, 1, 1, _
                1, 1, 0, _
                3)
EdgeMask(12) = Array(0, 1, 1, _
                1, 1, 0, _
                1, 0, 0, _
                3)
EdgeMask(13) = Array(1, 0, 0, _
                0, 1, 0, _
                0, 0, 1, _
                4)
EdgeMask(14) = Array(1, 1, 0, _
                0, 1, 1, _
                0, 0, 1, _
                4)
EdgeMask(15) = Array(1, 0, 0, _
                1, 1, 0, _
                0, 1, 1, _
```

```
                4)
  'Crosses
  EdgeMask(16) = Array(1, 0, 1, _
                0, 1, 0, _
                1, 0, 1, _
                5)
  EdgeMask(17) = Array(0, 1, 0, _
                1, 1, 1, _
                0, 1, 0, _
                5)
  'nothing
  EdgeMask(18) = Array(0, 0, 0, _
                0, 0, 0, _
                0, 0, 0, _
                0) 'last number indicates edge type
  EdgeMask(19) = Array(1, 1, 1, _
                1, 1, 1, _
                1, 1, 1, _
                0)
End Sub


Public Sub traceEdges()
'traces edges within the image
  Dim finished As Boolean
  Dim x As Integer
  Dim y As Integer
  Dim traced As Boolean

  finished = False
  traced = False
  x = 0
  y = 0
```

```
  While (Not finished)
    x = x + 1
    If (x = width) Then
      y = y + 1
      x = 0
    End If
    If (y < height) Then
      If ((edgeTraced(x, y) = False) And (image(x, y) > TraceEdgesThresh)) Then
        traced = traceEdgesFromPoint(x, y, 0)
      End If
      Else
      x = 0
      y = 0
      If (Not traced) Then
        finished = True
      End If
      traced = False
    End If
  Wend

  Call sortEdgeVector
  Call calcAngleHistogram

End Sub


Public Sub traceEdges_old()
'traces edges within the image

  Dim x As Integer
  Dim y As Integer

  traceX = 0
```

```
  traceY = 0
  Call traceSearch(True)
  While (traceRadius > 0)
    If (traceSearch()) Then
      traceRadius = 90
      x = Int(traceX)
      y = Int(traceY)
      If (traceEdgesFromPoint(x, y, 0)) Then
        traceX = x
        traceY = y
      End If
    End If
  Wend

  'Call sortEdgeVector
  Call calcAngleHistogram

End Sub



Private Sub diffuseEdges()
'diffuses edges information
'this allows edge tracing to be more noise tollerant
  Dim x As Integer
  Dim y As Integer
  Dim i As Integer
  Dim value As Integer

  For i = 0 To 1
    For x = 1 To width - 2
      For y = 1 To height - 2
        If (image(x, y) > TraceEdgesThresh) Then
          image(x, y) = 255
```

```
        'value = image(x - 1, y - 1)
        'value = value + image(x - 1, y)
        'value = value + image(x - 1, y + 1)
        'value = value + image(x + 1, y - 1)
        'value = value + image(x + 1, y)
        'value = value + image(x + 1, y + 1)
        'value = value + image(x, y + 1)
        'value = value + image(x, y - 1)
        'value = value / 8
        'image(x, y) = value
       End If
     Next
   Next
 Next

End Sub


Public Function traceEdgesFromPoint(ByRef x As Integer, ByRef y As Integer,
ByRef edgeLength As Integer) As Boolean
'traces along edges starting at the given point
  Dim i As Integer
  Dim j As Integer
  Dim sx As Integer
  Dim sy As Integer
  Dim xx As Integer
  Dim yy As Integer
  Dim pathFound As Boolean
  Dim initialEdgeLength As Integer
  Dim mindirection As Single
  Dim maxdirection As Single
  Dim initialX As Integer
```

```
Dim initialY As Integer
Dim max As Integer
Dim value As Integer
Dim intensity As Single
Dim direction As Integer
Static averagedirection As Single
Dim directionDifference As Integer
Dim thresh As Integer

initialX = x
initialY = y
xx = initialX
yy = initialY
initialEdgeLength = edgeLength
intensity = 0
thresh = 0 ' TraceEdgesThresh / 2

If (initialEdgeLength = 0) Then
  For i = 0 To width - 1
    For j = 0 To height - 1
      temp(i, j) = False
    Next
  Next
End If

averagedirection = 0
traceEdgesFromPoint = False
While ((image(xx, yy) > thresh) And (temp(xx, yy) = False))
  sx = xx
  sy = yy
  temp(xx, yy) = True
  edgeLength = edgeLength + 1
  If (edgeTraced(xx, yy) = False) And (edgeLength > minEdgeLength) Then
```

```
    traceEdgesFromPoint = True
   End If
   pathFound = False
   max = 0

   If (sy > 0) Then
    value = image(sx, sy - 1)
    If ((value > thresh) And (temp(sx, sy - 1) = False)) Then
     If (value > max) And ((averagedirection > 270) Or (averagedirection < 90))
Then
      max = value
      xx = sx
      yy = sy - 1
      direction = 0
     End If
    End If
   End If

   If (sx < width - 1) Then

    If (sy > 0) Then
     value = image(sx + 1, sy - 1)
     If ((value > thresh) And (temp(sx + 1, sy - 1) = False)) Then
      If (value > max) And ((averagedirection > 315) And (averagedirection < 135))
Then
       max = value
       xx = sx
       yy = sy - 1
       direction = 45
      End If
     End If
    End If
```

```
      value = image(sx + 1, sy)
      If ((value > thresh) And (temp(sx + 1, sy) = False)) Then
        If (value > max) And ((averagedirection > 0) And (averagedirection < 180))
Then
          max = value
          xx = sx + 1
          yy = sy
          direction = 90
        End If
      End If


      If (sy < height - 1) Then
        value = image(sx + 1, sy + 1)
        If ((value > thresh) And (temp(sx + 1, sy + 1) = False)) Then
          If (value > max) And ((averagedirection > 45) And (averagedirection < 225))
Then
            max = value
            xx = sx
            yy = sy + 1
            direction = 135
          End If
        End If
      End If

    End If

    If (sy < height - 1) Then
      value = image(sx, sy + 1)
      If ((value > thresh) And (temp(sx, sy + 1) = False)) Then
        If (value > max) And ((averagedirection > 90) And (averagedirection < 270))
Then
          max = value
```

```
          xx = sx
          yy = sy + 1
          direction = 180
        End If
      End If
    End If

    If (sx > 0) Then

      If (sy < height - 1) Then
        value = image(sx - 1, sy + 1)
        If ((value > thresh) And (temp(sx - 1, sy + 1) = False)) Then
          If (value > max) And ((averagedirection > 135) And (averagedirection < 315))
Then
            max = value
            xx = sx - 1
            yy = sy + 1
            direction = 225
          End If
        End If
      End If

      value = image(sx - 1, sy)
      If ((value > thresh) And (temp(sx - 1, sy) = False)) Then
        If (value > max) And ((averagedirection > 180) Or (averagedirection = 0)) Then
          max = value
          xx = sx - 1
          yy = sy
          direction = 270
        End If
      End If

      If (sy > 0) Then
```

```
        value = image(sx - 1, sy - 1)
        If ((value > thresh) And (temp(sx - 1, sy - 1) = False)) Then
          If (value > max) And ((averagedirection > 225) Or (averagedirection < 45))
Then
            max = value
            xx = sx - 1
            yy = sy - 1
            direction = 315
          End If
        End If
      End If
    End If

    If (averagedirection > 0) Then
      intensity = (intensity + max) / 2
      directionDifference = Abs(averagedirection - direction)
      If (directionDifference > 180) Then
        directionDifference = 360 - directionDifference
      End If
      averagedirection = averagedirection - (directionDifference / 2)
      If (averagedirection < 0) Then
        averagedirection = 360 + averagedirection
      End If
      If (averagedirection > 360) Then
        averagedirection = averagedirection - 360
      End If

      If ((edgeLength > 3) And (directionDifference > 20) And
(traceEdgesFromPoint)) Then
        Call addEdgeVector(initialX, initialY, xx, yy, intensity)
        initialX = xx
        initialY = yy
      End If
```

```
    Else
    intensity = max
    averagedirection = direction
   End If

  Wend

  If (traceEdgesFromPoint = True) Then
   Call addEdgeVector(initialX, initialY, xx, yy, intensity)
  End If

  If (initialEdgeLength = 0) Then
   'If (edgeLength > minEdgeLength) Then
    For i = 0 To width - 1
     For j = 0 To height - 1
      If (temp(i, j) = True) Then
       edgeTraced(i, j) = True
      End If
     Next
    Next
   'End If
  End If

  x = xx
  y = yy
  traceDirection = direction

End Function


Private Sub addEdgeVector(x1 As Integer, y1 As Integer, x2 As Integer, y2 As
Integer, intensity As Single)
```

```
'adds a new edge vector

  If (currEdgeVector < EDGE_VECTOR_LENGTH) Then
    EdgeVector(0, currEdgeVector) = x1
    EdgeVector(1, currEdgeVector) = y1
    EdgeVector(2, currEdgeVector) = x2
    EdgeVector(3, currEdgeVector) = y2
    EdgeVector(4, currEdgeVector) = intensity
    If (intensity > maxEdgeVectorIntensity) Then
      maxEdgeVectorIntensity = intensity
    End If
    currEdgeVector = currEdgeVector + 1
  End If
End Sub



Private Sub sortEdgeVector()
'sorts the edge vector by distance

  Dim dx As Integer
  Dim dy As Integer
  Dim length As Long
  Dim mindist As Long
  Dim closest As Integer
  Dim vect As Single
  Dim i As Integer
  Dim j As Integer

  For i = 0 To currEdgeVector - 2
    mindist = 99999
    closest = 0
    For j = i + 1 To currEdgeVector - 1
      dx = EdgeVector(2, i) - EdgeVector(0, j)
```

```
    dy = EdgeVector(3, i) - EdgeVector(1, j)
    length = (dx * dx) + (dy * dy)
    If (length < mindist) Then
      mindist = length
      closest = j
    End If
  Next
  If ((closest > 0) And (closest <> i + 1)) Then
    'swap
    For j = 0 To 4
      vect = EdgeVector(j, i + 1)
      EdgeVector(j, i + 1) = EdgeVector(j, closest)
      EdgeVector(j, closest) = vect
    Next
  End If
Next



'For i = 0 To currEdgeVector - 1
'  For j = 0 To currEdgeVector - 1
'    If (i <> j) Then
'      dx = EdgeVector(0, i) - EdgeVector(0, j)
'      dy = EdgeVector(1, i) - EdgeVector(1, j)
'      length = ((dx * dx) + (dy * dy))
'      If (length < 3 * 3) Then
'        EdgeVector(0, i) = EdgeVector(0, i) - (dx / 2)
'        EdgeVector(1, i) = EdgeVector(1, i) - (dy / 2)
'        EdgeVector(0, j) = EdgeVector(0, j) + (dx / 2)
'        EdgeVector(1, j) = EdgeVector(1, j) + (dy / 2)
'      End If
'      dx = EdgeVector(2, i) - EdgeVector(2, j)
'      dy = EdgeVector(3, i) - EdgeVector(3, j)
```

```
'      length = ((dx * dx) + (dy * dy))
'      If (length < 3 * 3) Then
'        EdgeVector(2, i) = EdgeVector(2, i) - (dx / 2)
'        EdgeVector(3, i) = EdgeVector(3, i) - (dy / 2)
'        EdgeVector(2, j) = EdgeVector(2, j) + (dx / 2)
'        EdgeVector(3, j) = EdgeVector(3, j) + (dy / 2)
'      End If
'    End If
'  Next
 'Next

End Sub


Private Function dist(x1 As Single, y1 As Single, x2 As Single, y2 As Single) As
Single
  Dim dx As Single
  Dim dy As Single

  dx = x1 = x2
  dy = y1 - y2
  dist = Sqr((dx * dx) + (dy * dy))
End Function



Public Sub getEdges()
'updates the edges

  Dim mask
  Dim i As Integer
  Dim j As Integer
  Dim x As Integer
```

```
Dim y As Integer
Dim xx As Integer
Dim yy As Integer
Dim diff As Long
Dim thresh As Integer
Dim diff2 As Long
Dim estr As String
Dim minDiff As Long
Dim winner As Integer
Dim ex As Integer
Dim ey As Integer
Dim av As Integer

thresh = 100

For i = 0 To NO_OF_EDGE_TYPES - 1
  EdgeHistogram(i) = 0
Next

x = 0
ex = 0
While (x < width - 2)
  y = 0
  ey = 0
  While (y < height - 2)
    Edges(ex, ey) = 0
    minDiff = 9999999
    winner = -1
    For i = 0 To NO_OF_EDGE_MASKS - 1
      mask = EdgeMask(i)
      diff = 0
      j = 0
      av = 0
```

```
  For yy = y To y + 2
    For xx = x To x + 2
      av = av + image(xx, yy)
      diff2 = Abs((mask(j) * 255) - image(xx, yy))
      diff = diff + diff2
      j = j + 1
    Next
  Next
  If (av / 9 > 30) Then

    'edge

    diff = diff / 9
    If (diff < minDiff) And (diff < thresh) Then
      winner = mask(9)
      minDiff = diff
      Edges(ex, ey) = winner
    End If

    Else

    'blank
    winner = 0
    Edges(ex, ey) = winner

  End If
Next
'Edges(ex, ey) = Rnd * 5  'test
If (winner > 0) Then
  EdgeHistogram(winner - 1) = EdgeHistogram(winner - 1) + 1
End If
ey = ey + 1
y = y + 2
```

```
    Wend
    ex = ex + 1
    x = x + 2
  Wend

  'fill in the gaps
  Call getEdges_secondary

End Sub


Public Sub getEdges_secondary()
'fills in edges where they "should" appear
  Dim x As Integer
  Dim y As Integer

  For x = 1 To edgesWidth - 1
    For y = 1 To edgesHeight - 1
      'horizontal
      If ((Edges(x - 1, y) > 0) And (Edges(x + 1, y) > 0)) Then
        Edges(x, y) = 1
        Else
        'vertical
        If ((Edges(x, y - 1) > 0) And (Edges(x, y + 1) > 0)) Then
          Edges(x, y) = 2
          Else
         'diagonal
         If ((Edges(x - 1, y - 1) > 0) And (Edges(x + 1, y + 1) > 0)) Then
           'Edges(x, y) = 4
           Else
           'diagonal
           If ((Edges(x + 1, y - 1) > 0) And (Edges(x - 1, y + 1) > 0)) Then
             'Edges(x, y) = 3
```

```
          End If
        End If
       End If
      End If

      If ((Edges(x + 1, y) <> 1) And (Edges(x + 1, y) = Edges(x, y))) Then
        Edges(x, y) = 0
      End If
      If ((Edges(x, y + 1) <> 2) And (Edges(x, y + 1) = Edges(x, y))) Then
        Edges(x, y) = 0
      End If

      'surrounded by edges
      If ((Edges(x - 1, y - 1) > 0) And (Edges(x - 1, y) > 0) And (Edges(x - 1, y + 1) >
0) And (Edges(x, y - 1) > 0) And (Edges(x, y + 1) > 0) And (Edges(x + 1, y - 1) > 0)
And (Edges(x + 1, y) > 0) And (Edges(x + 1, y + 1) > 0)) Then
        Edges(x, y) = 0
      End If

    Next
  Next

End Sub


Public Sub init(imageWidth As Integer, imageHeight As Integer)
  width = imageWidth
  height = imageHeight
  ReDim image(width, height)

  ReDim edgeTraced(width, height)
  ReDim temp(width, height)
  minEdgeLength = 10
```

```
  edgesWidth = width / 2
  edgesHeight = height / 2
  ReDim Edges(edgesWidth, edgesHeight)
  EdgeThreshold = 0
  processType = 0
  Call initEdgeMasks
  averageContrast = 1
  ReDim picked(width, height)
End Sub


Private Sub calcEdgeVector()
'calculates the edge vector for the image
  Dim i As Integer

  For i = 0 To EDGE_VECTOR_LENGTH - 1

  Next

End Sub


Public Sub whiteNoise()
  Dim x As Integer
  Dim y As Integer

  For x = 0 To width - 1
    For y = 0 To height - 1
      image(x, y) = Rnd * 255
    Next
  Next
End Sub
```

```
Public Function getPoint(x As Integer, y As Integer) As Byte
  getPoint = image(x, y)
End Function


Public Function setPoint(x As Integer, y As Integer, value As Byte)
  image(x, y) = value
End Function


Public Sub update(canvas As PictureBox, Optional left As Variant, Optional top As
Variant, Optional wdth As Variant, Optional hght As Variant)
'import a picture
'processtype = 0   greyscale
'             1   red
'             2   green
'             3   blue
'             4   edges
'             5   movement

  Dim x As Integer
  Dim y As Integer
  Dim screenX As Integer
  Dim screenY As Integer
  Dim w As Integer
  Dim h As Integer
  Dim xx As Integer
  Dim yy As Integer
  Dim value As Double
  Dim RGBval As Long
  Dim pixels As Double
```

```
Dim maxCol As Long
Dim edgeValue As Single
Dim screenWidth As Single
Dim screenHeight As Single
Dim screenLeft As Single
Dim screenTop As Single

If (Not IsMissing(left)) And (Not IsMissing(top)) Then
  screenLeft = left
  screenTop = top
  screenWidth = wdth
  screenHeight = hght
  Else
  screenLeft = 0
  screenTop = 0
  screenWidth = canvas.ScaleWidth
  screenHeight = canvas.ScaleHeight
End If

w = CInt(screenWidth / width)
If (w < 1) Then
  w = 1
End If
h = CInt(screenHeight / height)
If (h < 1) Then
  h = 1
End If
pixels = w * h
maxCol = RGB(255, 255, 255)
For x = 0 To width - 1
  For y = 0 To height - 1
    edgeTraced(x, y) = False
    screenX = screenLeft + ((x / width) * screenWidth)
```

```
      screenY = screenTop + ((y / height) * screenHeight)
      value = 0
      For xx = screenX To screenX + w - 1
        For yy = screenY To screenY + h - 1

          RGBval = canvas.Point(xx, yy)
          Select Case processType
            Case 0  'greyscale
              value = value + (RGBval / maxCol)
            Case 1  'red
              value = value + ((RGBval And 255) / 255)
            Case 2  'green
              value = value + ((RGBval And 65280) / 65280)
            Case 3  'blue
              value = value + ((RGBval And 16711680) / 16711680)
          End Select

        Next
      Next
      value = (value / pixels) * 255
      image(x, y) = value
    Next
  Next

End Sub


Public Sub getImageEdges(rawImage As classImageProcessing)
'extracts edges from the given image
  Dim x As Integer
  Dim y As Integer
  Dim value As Single
  Dim scalex As Single
```

```
Dim scaley As Single
Dim xx As Integer
Dim yy As Integer
Dim p1 As Integer
Dim p2 As Integer
Dim avContrast As Double

scalex = rawImage.width / width
scaley = rawImage.height / height

currEdgeVector = 0
maxEdgeVectorIntensity = 0

avContrast = 0
For x = 1 To width - 1
  For y = 1 To height - 1
    edgeTraced(x, y) = False
    xx = x * scalex
    yy = y * scaley
    If ((xx >= 1) And (yy >= 1)) Then
      p1 = rawImage.getPoint(xx, yy)
      p2 = rawImage.getPoint(xx - 1, yy)
      value = Abs(p1 - p2)
      p2 = rawImage.getPoint(xx, yy - 1)
      value = value + Abs(p1 - p2)
      value = value / (255 * 2)
      avContrast = avContrast + value
      'If (Abs(value - averageContrast) < EdgeThreshold) Then
      If (value < EdgeThreshold) Then
        value = 0
        Else
        value = 255 * value
      End If
```

```
      image(x, y) = value
    End If
  Next
 Next

 'calc average contast
 avContrast = avContrast / (width * height)
 averageContrast = avContrast
 If (averageContrast < 0.01) Then
  averageContrast = 0.01
 End If

 'calc threshold used for tracing along edges
 TraceEdgesThresh = (averageContrast * 255) * 0.1

 'Call diffuseEdges

 'Call getEdges
 Call traceEdges

End Sub


Public Sub getImageContours(rawImage As classImageProcessing)
'extracts edges from the given image
 Dim x As Integer
 Dim y As Integer
 Dim value As Single
 Dim scalex As Single
 Dim scaley As Single
 Dim xx As Integer
 Dim yy As Integer
 Dim p1 As Integer
```

```
Dim p2 As Integer
Dim value2 As Single
Dim max As Single

scalex = rawImage.width / width
scaley = rawImage.height / height

currEdgeVector = 0
maxEdgeVectorIntensity = 0
max = 1 - EdgeThreshold

For x = 1 To width - 1
  For y = 1 To height - 1
    edgeTraced(x, y) = False
    xx = x * scalex
    yy = y * scaley
    If ((xx >= 1) And (yy >= 1)) Then
      p1 = rawImage.getPoint(xx, yy)
      p2 = rawImage.getPoint(xx - 1, yy)
      value = Abs(p1 - p2)
      p2 = rawImage.getPoint(xx, yy - 1)
      value = value + Abs(p1 - p2)
      value = value / (255 * 2)
      value2 = value - EdgeThreshold
      If (value2 < 0) Then
        value = 0
        Else
        value = 255 - (255 * (value2 / max))
      End If
      image(x, y) = value
    End If
  Next
Next
```

```
End Sub



Public Sub show(canvas As PictureBox)
  Dim x As Integer
  Dim y As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim value As Byte
  Dim c As Long
  Dim i As Integer

  If (processType <> 4) Then

  canvas.FillStyle = 0
  For x = 0 To width - 1
    For y = 0 To height - 1
      value = image(x, y)
      Select Case processType
        Case 1 'red
          c = RGB(value, 0, 0)
        Case 2 'green
          c = RGB(0, value, 0)
        Case 3 'blue
          c = RGB(0, 0, value)
        Case 4 'edges
          value = 255 - value
          c = RGB(value, value, value)
        Case Else
          c = RGB(value, value, value)
```

```
      End Select
      canvas.FillColor = c
      screenX(0) = (x / width) * canvas.ScaleWidth
      screenY(0) = (y / height) * canvas.ScaleHeight
      screenX(1) = ((x + 1) / width) * canvas.ScaleWidth
      screenY(1) = ((y + 1) / height) * canvas.ScaleHeight
      canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c, B
    Next
  Next

  Else

    'Call showEdges(canvas)
    canvas.Cls
    Call showEdgeTraces(canvas)

  End If

End Sub



Public Sub showEdgeTraces(canvas As PictureBox)
  Dim x As Integer
  Dim y As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim value As Byte
  Dim c As Long
  Dim i As Integer

  'canvas.Cls
  canvas.FillStyle = 0
  For x = 0 To width - 1
```

```
    For y = 0 To height - 1
      If (edgeTraced(x, y) = True) Then
        c = RGB(230, 230, 230)
        canvas.FillColor = c
        screenX(0) = (x / width) * canvas.ScaleWidth
        screenY(0) = (y / height) * canvas.ScaleHeight
        screenX(1) = ((x + 1) / width) * canvas.ScaleWidth
        screenY(1) = ((y + 1) / height) * canvas.ScaleHeight
        canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c, B
      End If
    Next
  Next

  Call showEdgeVector(canvas)

End Sub

Public Sub showEdgeVector(canvas As PictureBox)
  Dim x1 As Integer
  Dim y1 As Integer
  Dim x2 As Integer
  Dim y2 As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim value As Byte
  Dim c As Long
  Dim i As Integer
  Dim radius As Integer

  'canvas.Cls
  canvas.FillStyle = 0
  canvas.DrawWidth = 1
  radius = (canvas.ScaleWidth / width) / 2
```

```
  For i = 0 To currEdgeVector - 1
    x1 = EdgeVector(0, i)
    y1 = EdgeVector(1, i)
    x2 = EdgeVector(2, i)
    y2 = EdgeVector(3, i)

    'c = RGB((EdgeVector(4, i) / maxEdgeVectorIntensity) * 255, 0, 0)
    c = RGB(i, 0, 0)
    canvas.FillColor = c
    screenX(0) = (x1 / width) * canvas.ScaleWidth
    screenY(0) = (y1 / height) * canvas.ScaleHeight
    screenX(1) = (x2 / width) * canvas.ScaleWidth
    screenY(1) = (y2 / height) * canvas.ScaleHeight
    If (i > 0) Then
      canvas.Line -(screenX(0), screenY(0)), c
    End If
    canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c
    'canvas.Circle (screenX(0), screenY(0)), radius, c
    'canvas.Circle (screenX(1), screenY(1)), radius, c
  Next

End Sub



Public Sub showEdges(canvas As PictureBox)
  Dim x As Integer
  Dim y As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim edgeType As Byte
  Dim c As Long
  Dim i As Integer
```

```
 canvas.Cls
 canvas.FillStyle = 0
 c = RGB(0, 0, 0)
 For x = 0 To edgesWidth - 1
  For y = 0 To edgesHeight - 1

    screenX(0) = (x / edgesWidth) * canvas.ScaleWidth
    screenY(0) = (y / edgesHeight) * canvas.ScaleHeight
    screenX(1) = ((x + 1) / edgesWidth) * canvas.ScaleWidth
    screenY(1) = ((y + 1) / edgesHeight) * canvas.ScaleHeight

    edgeType = Edges(x, y)
    Select Case edgeType
     Case 1 'horizontal line
       canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(0)), c
     Case 2 'vertical line
       canvas.Line (screenX(0), screenY(0))-(screenX(0), screenY(1)), c
     Case 3 'diagonal /
       canvas.Line (screenX(0), screenY(1))-(screenX(1), screenY(0)), c
     Case 4 'diagonal \
       canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c
     Case 5 'cross
       canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(0)), c
       canvas.Line (screenX(0), screenY(0))-(screenX(0), screenY(1)), c
    End Select
  Next
 Next

End Sub


Public Sub showEdgeHistogram(chart As Object)
```

```
'displays edge histogram using MS chart control

  Dim i As Integer
  Dim estr As String

  chart.chartType = 7
  chart.RowCount = NO_OF_EDGE_TYPES
  chart.ColumnCount = 1

  estr = ""
  For i = 0 To chart.RowCount - 1
    chart.Row = i + 1
    chart.Data = EdgeHistogram(i)
    estr = estr & EdgeHistogram(i) & ", "
  Next
  chart.Refresh
  'MsgBox estr

End Sub


Public Sub showAngleHistogram(chart As Object)
'displays angle histogram using MS chart control

  Dim i As Integer
  Dim estr As String

  chart.chartType = 7
  chart.RowCount = 18
  chart.ColumnCount = 1

  estr = ""
  For i = 0 To chart.RowCount - 1
```

```
    chart.Row = i + 1
    chart.Data = angleHistogram(i)
    estr = estr & angleHistogram(i) & ", "
  Next
  chart.Refresh
  'MsgBox estr

End Sub
```

4.4.3. Yüz tanıma için kullanılan fonksiyonlar

```
Private imageWidth As Integer
Private imageHeight As Integer

Private NoOfFaces As Integer
Dim Face(1000) As classImageProcessing
Dim EigenFace(1000) As classImageProcessing
Dim NameOfFace(1000) As String

Dim faceTemplate() As Single

Dim testFace As classImageProcessing
Dim testEigenFace As classImageProcessing

Private Identity As classImageProcessing
'To fire this event, use RaiseEvent with the following syntax:
'RaiseEvent IdentifyImageChange[(arg1, arg2, ... , argn)]
Public Event IdentifyImageChange(ByVal ImageName As String)
```

```
Public Function ImageCount() As Long
    ImageCount = NoOfFaces
End Function

Public Function Identify(ByVal PicFile As String) As String
    Dim pic As PictureBox
    Load frmImage
    Set frmImage.picImage.Picture = LoadPicture(PicFile)
    Identify = LocalIdentify(frmImage.picImage)
    Unload frmImage
End Function

Public Sub AddFace(ByVal PicFile As String)
    Load frmImage
    Set frmImage.picImage.Picture = LoadPicture(PicFile)
    LocalAddFace frmImage.picImage, PicFile
    Unload frmImage
End Sub


Public Sub init(image_Width As Integer, image_Height As Integer)
  imageWidth = image_Width
  imageHeight = image_Height
  ReDim faceTemplate(imageWidth, imageHeight)
End Sub


Private Sub LocalAddFace(facePicture As PictureBox, faceName As String)
  Set Face(NoOfFaces) = New classImageProcessing
  Set EigenFace(NoOfFaces) = New classImageProcessing
  Set testFace = New classImageProcessing
  Set testEigenFace = New classImageProcessing
```

```
  Call Face(NoOfFaces).init(imageWidth, imageHeight)
  Call EigenFace(NoOfFaces).init(imageWidth, imageHeight)
  Call testFace.init(imageWidth, imageHeight)
  Call testEigenFace.init(imageWidth, imageHeight)
  Call Face(NoOfFaces).update(facePicture)
  NameOfFace(NoOfFaces) = faceName
  NoOfFaces = NoOfFaces + 1

  Call updateFaceTemplate
  Call updateEigenFaces
End Sub


Private Sub updateFaceTemplate()
'calculates an average face template
  Dim i As Integer
  Dim x As Integer
  Dim y As Integer

  For i = 0 To NoOfFaces - 1
    For x = 0 To imageWidth - 1
      For y = 0 To imageHeight - 1
        If (i > 0) Then
          faceTemplate(x, y) = faceTemplate(x, y) + Face(i).getPoint(x, y)
          Else
          faceTemplate(x, y) = Face(i).getPoint(x, y)
        End If
      Next
    Next
  Next

  For x = 0 To imageWidth - 1
    For y = 0 To imageHeight - 1
```

```
      faceTemplate(x, y) = Int(faceTemplate(x, y) / NoOfFaces)
    Next
  Next
End Sub


Private Sub updateEigenFaces()
'updates all the eigenfaces
  Dim i As Integer
  Dim x As Integer
  Dim y As Integer
  Dim df As Integer

  For i = 0 To NoOfFaces - 1
    For x = 0 To imageWidth - 1
      For y = 0 To imageHeight - 1
        df = Face(i).getPoint(x, y) - faceTemplate(x, y)
        If (df < 0) Then
          df = 0
        End If
        Call EigenFace(i).setPoint(x, y, CByte(df))
      Next
    Next
  Next
End Sub


Private Function LocalIdentify(facePicture As PictureBox) As String
'identifies the given image
  Dim i As Integer
  Dim x As Integer
  Dim y As Integer
  Dim df As Integer
```

```
Dim Distance As Long
Dim minDistance As Long
Dim retval As String
Dim a As Integer
Dim B As Integer

retval = ""
Call testFace.update(facePicture)

'calculate the eigenface
For x = 0 To imageWidth - 1
  For y = 0 To imageHeight - 1
    df = testFace.getPoint(x, y) - faceTemplate(x, y)
    If (df < 0) Then
      df = 0
    End If
    Call testEigenFace.setPoint(x, y, CByte(df))
  Next
Next

'compare it to other eigenfaces
minDistance = 99999999#
For i = 0 To NoOfFaces - 1
  RaiseEvent IdentifyImageChange(NameOfFace(i))
  Distance = 0
  For x = 0 To imageWidth - 1
    For y = 0 To imageHeight - 1
      a = EigenFace(i).getPoint(x, y)
      B = testEigenFace.getPoint(x, y)
      df = Abs(a - B)
      Distance = Distance + df
    Next
  Next
```

```
    If (Distance < minDistance) Then
      minDistance = Distance
      retval = NameOfFace(i)
      Set Identity = Face(i)
    End If
  Next
  LocalIdentify = retval
End Function
```

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Bu bölüm, tezde anlatılan konunun bir özetini vermektedir. Ayrıca gelecekte yapılabilecek olan geliştirmeler için neler yapılabileceğini içermektedir.

5.1. Sonuçlar

Bu araştırmada, yüz tanıma problemi için iki ana yaklaşım üzerinde araştırma yapılmıştır ve özyüzler yaklaşımına dayalı bir yüz tanıma sistemi önerilmiştir. Bu iki yaklaşımın ana özelliklleri :

- Özellik tabanlı yüz tanıma: Yüzün göz, ağız ve burun gibi organlarının bireysel özellikleri ve bunların birbirleri ile ilişkileri kullanılarak, tanıma işlemi yapılmaktadır. Bu özellikleri hesaplamanın en yaygın kullanılan yöntemi şekil değiştirebilen şablonlar ve aktif kontur modelleridir. Yüz ile ilgili özellikler, öncelikle kaba kontur tahmin metodu ile tespit edilir ve bazı enerji fonksiyonları minimize edilerek kesin sonuçlar çıkarılır. Bu yaklaşımın temel karakteristiği geometriye bağımlı olmasıdır.

- Temel bileşen analizi: Bu yaklaşımda yüz tanıma problemi bilgi teorisi kavramına dayalıdır. Yüz resminden en çok anlamlı olan bilgiler çıkarılır. Özyüzler metodu, resmin karakteristik özelliklerinin küçük bir kümesinin kovaryans matrisinin özvektörlerinin kullanıldığı temel bileşen analizidir. Bu özvektörler, yüz resimlerine benzerliklerinden dolayı özyüz olarak adlandırılırlar. Tanıma işlemi, özyüzlerin yüz uzayındaki katkılarıyla, yüz resimlerine atanan ağırlık vektörleri ile gerçekleştirilir. Bu yaklaşım hızı, basitliği ve öğrenme kabiliyeti ile ön plana çıkar.

Güçlü bir yüz tanıma sistemi aşağıdaki özelliklerden etkilenmemelidir:

- Işık kaynağındaki değişimler
- Başın açısından ve oranından
- Yüzde bulunan gözlük ve sakal gibi detaylardan
- Yüz arka zemininden

Bu çalışmada önerilen yüz tanıma sistemi, yüz arka zemininden ve kafa açısından oldukça fazla miktarda etkilendiği deneyimler sonucu görülmüştür. Işık kaynağındaki değişiklikler büyük problem teşkil etmemektedir. Ayrıca, yüzde bulunan gözlük, sakal ve maske gibi ufak detaylar sistemi fazla etkilememektedir. Doğru tanıma oranı ve eşik değeri arasında bir ilişki vardır. Eşik değeri arttırıldığında ıska olanların sayısı azalmaktadır. Fakat yanlış sınıflandırma oranı artmaktadır. Tersi durumda, ıska sayısı artmasına karşın doğru sınıflandırma sayısı artmaktadır.

5.2. Öneriler

Bu tezde yapılan çalışmanın geliştirilmesi için aşağıdaki işlemler yapılabilir:

- Arka zemin çıkarma algoritması geliştirilmesi: Tanıma performasına etkisinin en aza indirgenmesi için resmin arka planının temizlenmesi ve kafa pozisyonun düzeltilmesi yapılabilir.

- Farklı görünümlerden tanıma: Bu tezde yapılan çalışma, yüzün ön yüz görünümü için çalışmaktadır. Yapay sinir ağı mimarisi (özellik tabanlı yaklaşımla beraber kullanılabilir) kullanılarak yüzün yönü tespit edilerek en uygun tanıma sistemi seçilebilir.

- Kamera ve tarayıcı desteği: Şu anda yüz resimleri bilgisayardaki dosyalardan alınmaktadır. Tarayıcı yardımıyla herhangi bir resim taranarak tanıma işlemi yapılabilmesi büyük kolaylık sağlayacaktır. Ayrıca güvenlik kameralarının görüntülediği ortamlarda sistemde tanımlı bir kişi görüntüye girdiğinde uyarı verilmesi sağlanabilir.

KAYNAKLAR

[1] Goldstein, A. J., Harmon, L. D., and Lesk, A. B., Identification of human faces", Proc. IEEE 59, pp. 748-760, (1971).

[2] Haig, N. K., "How faces differ - a new comparative technique", Perception 14, pp. 601-615, (1985).

[3] Rhodes, G., "Looking at faces: First-order and second order features as determinants of facial appearance", Perception 17, pp. 43-63, (1988).

[4] Kirby, M., and Sirovich, L., "Application of the Karhunen-Loeve procedure for the characterization of human faces", IEEE PAMI, Vol. 12, pp. 103-108, (1990).

[5] Sirovich, L., and Kirby, M., "Low-dimensional procedure for the characterization of human faces", J. Opt. Soc. Am. A, 4, 3,pp. 519-524, (1987).

[6] Terzopoulos, D., and Waters, K., "Analysis of facial images using physical and anatomical models", Proc. 3rd Int. Conf. on Computer Vision, pp. 727-732, (1990).

[7] Manjunath, B. S., Chellappa, R., and Malsburg, C., "A feature based approach to face recognition", Trans. of IEEE, pp. 373-378, (1992).

[8] Harmon, L. D., and Hunt, W. F., "Automatic recognition of human face profiles", Computer Graphics and Image Processing, Vol. 6, pp. 135-156, (1977).

[9] Harmon, L. D., Khan, M. K., Lasch, R., and Ramig, P. F., "Machine identification of human faces", Pattern Recognition, Vol. 13(2), pp. 97-110, (1981).

[10] Kaufman, G. J., and Breeding, K. J, "The automatic recognition of human faces from profile silhouettes", IEEE Trans. Syst. Man Cybern., Vol. 6, pp. 113-120, (1976).

[11] Wu, C. J., and Huang, J. S., "Human face profile recognition by computer", Pattern Recognition, Vol. 23(3/4), pp. 255-259, (1990).

[12] Kerin, M. A., and Stonham, T. J., "Face recognition using a digital neural network with self-organizing capabilities", Proc. 10th Int. Conf. on Pattern Recognition, pp.738-741, (1990).

[13] Nakamura, O., Mathur, S., and Minami, T., "Identification of human faces based on isodensity maps", Pattern Recognition, Vol. 24(3), pp. 263-272, (1991).

[14] Turk, M., and Pentland, A., "Eigenfaces for recognition", Journal of Cognitive Neuroscience, Vol. 3, pp. 71-86, (1991).

[15] Yuille, A. L., Cohen, D. S., and Hallinan, P. W., "Feature extraction from faces using deformable templates", Proc. of CVPR, (1989).

[16] Gonzalez, R. C., and Tou, J. T., "Pattern recognition principles", Addison-Wesley Publishing Company, (1974).

[17] Carey, S., and Diamond, R., "From piecemeal to configurational representation of faces", Science 195, pp. 312-313, (1977).

[18] Bledsoe, W. W., "The model method in facial recognition", Panoramic Research Inc. Palo Alto, CA, Rep. PRI:15, (August 1966).

[19] Bledsoe, W. W., "Man-machine facial recognition", Panoramic Research Inc. Palo Alto, CA, Rep. PRI:22, (August 1966).

[20] Fischler, M. A., and Elschlager, R. A., "The representation and matching of pictorial structures", IEEE Trans. on Computers, c-22.1, (1973).

[21] Kohonen, T., "Self-organization and associative memory", Berlin: Springer-Verlag, (1989).

[22] Kohonen, T., and Lehtio, P., "Storage and processing of information in distributed associative memory systems", (1981).

[23] Fleming, M., and Cottrell, G., "Categorization of faces using unsupervised feature extraction", Proc. of IJCNN, Vol. 90(2), (1990).

[24] Kanade, T., "Picture processing system by computer complex and recognition of human faces", Dept. of Information Science, Kyoto University, (1973).

[25] Burt, P., "Smart sensing within a Pyramid Vision Machine", Proc. of IEEE, Vol. 76(8), pp. 139-153, (1988).

[26] Ching, C. W., and Huang, L. C., "Human face recognition from a single front view", Int. J. of Pattern Recognition and Artificial Intelligence, Vol. 6(4), pp. 570-593, (1992).

[27] Berger, M. O., and Mohr, R., "Towards autonomy in active contour models", Proc. 10th Int. Conf. on Pattern Recognition, pp. 847-851, (1990).

[28] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active contour models", Int. J. of Computer Vision, pp. 321-331, (1987).

[29] Williams, D. J., and Shah, M., "A fast algorithm for active contours", Proc. 3rd Int. Conf. on Computer Vision, pp. 592-595, (1990).

[30] Amir, A. A., Weymouth, T. E., and Jain, R. C., "Using dynamic programming for solving variational problems in vision", IEEE PAMI, Vol. 12(9), pp. 855-867, (1990).

[31] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., "Numerical recipes in C", Cambridge University Press, pp. 353-380, (1988).

ÖZGEÇMİŞ

1976 yılında Bursa ilinin İnegöl ilçesinde doğdu. İlköğrenimini Hacıkara Köyü ilkokulunda tamamladı. Orta öğrenimini Tahtaköprü ortaokulunda tamamladı. Lise eğitimini İnegöl Endüstri Meslek Lisesi Elektrik bölümünde tamamladı.

Üniversite eğitimine 1995 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümüne başladı. 2000 yılında Sakarya Üniversitesinden mezun oldu. 2002 yılında Sakarya Üniversitesi Fen Bilimleri Enstütüsünde yüksek lisans eğitimine başladı.

Lisans eğitiminden mezun olduktan sonra yazılım mühendisliği alanında proje yöneticisi olarak iş yaşamına devam etmektedir.