SOM ALGORİTMASIYLA PARMAK İZİ SINIFLANDIRILMASININ FPGA UYGULAMASI

FINGERPRINT CLASSIFICATION APPLICATION USING SOM ALGORITHM IN FPGA

YASEMİN CAN

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2006

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **ELEKTRİK ve ELEKTRONİK ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan	:….. Prof.Dr.Hüseyin Selçuk GEÇİM
Üye(Danışman)	:….. Yrd. Doç. Dr. Ali Ziya ALKAR
Üye	:….. Yrd. Doç. Dr. Atila YILMAZ
Üye	:….. Yrd. Doç. Dr. Derya ALTUNAY
Üye	:….. Yrd. Doç. Dr. Harun ARTUNER

ONAY

Bu tez, ... /..... / ..... tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

Prof. Dr. Ahmet R. ÖZDURAL
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

SOM ALGORİTMASIYLA PARMAK İZİ SINIFLANDIRILMASININ FPGA UYGULAMASI

Yasemin Can

ÖZ

Bu tez çalışmasında, parmak izi sınıflandırılması yapay sinir ağı algoritmalarından öz-düzenlemeli haritalama yöntemiyle gerçekleştirilmiştir. Öz-düzenlemeli haritalama algoritması ve biyolojik nöron davranışının FPGA'de sayısal devrelerle gerçekleştirilmesi amaçlanmıştır. Buna ek olarak, insan beyninin paralel işleme özelliği modellenerek algoritmanın işleme zamanının kısaltılması hedeflenmiştir.

Öz düzenlemeli haritalama kullanılmasıyla sınıflandırma danışmansız öğrenme ile gerçekleştirilmiştir. Öz düzenlemeli haritalama yönteminde sınıf sayısı uygulamaya göre düzenlenebildiğinden, veritabanının en uygun yapıda düzenlenmesi hedeflenmiştir.

Danışmansız öğrenme ile daha önce eğitim aşamasında kullanılmayan parmak izlerinin de doğru sınıflandırılması hedeflenmiştir.

Anahtar Kelimeler: FPGA, Sinir ağları, Parmak izi.

Danışman: Yrd. Doç Ali Ziya ALKAR, Hacettepe Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü, Elektronik Anabilim Dalı

FINGERPRINT CLASSIFICATION APPLICATION USING SOM ALGORITHM IN FPGA

Yasemin Can

ABSTRACT

In this work the fingerprint classification is performed using the self-organizing mapping neural network algorithm. By modelling the self-organizing mapping algorithm the biological neuron behaviour is implemented in the FPGA as digital circuits. In addition, the processing time is aimed to be shortened by modelling the paralel processing characteristic of the human brain.

Classification is performed by unsupervised learning making use of self organizing mapping. Since the number of classes can be arranged according to the application the database is aimed to be organized in the best possible structure.

Correct classification of the non-trained fingerprints is also aimed with unsupervised learning in this work.

Keywords: FPGA, Neural Network, Fingerprint.

Advisor: Asst. Prof. Dr. Ali Ziya ALKAR, Hacettepe University, Department of Electrical and Electronics Engineering, Electronics Section

TEŞEKKÜR

Sayın Yrd. Doç. Dr. Ali Ziya ALKAR ’ a bu çalışmayı yürütürken göstermiş olduğu ilgi ve ayrıca tezin gerçekleştirilmesi için gerekli cihazların ve laboratuar olanaklarının sağlanmasındaki katkıları nedeniyle teşekkürlerimi sunarım.

Mehmet Ali YILDIRIM ‘a ve İrfan OKŞAR ‘a gerçekleştirdiğim bu çalışma sırasında farklı ve yeni bakış açıları sundukları, fikir ve görüşlerini paylaştıkları için çok teşekkür ederim.

Ece YÜKSEL’ e ve Melike GÜRÜN’ e tez çalışmalarımdaki manevi destekleri sebebiyle teşekkürlerimi sunarım.

Hiçbir zaman desteklerini esirgemeyen aileme özellikle de kardeşim Sezai CAN’ a ayrıca teşekkür ederim.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

TERİMLER DİZİNİ

Arch	Kemer
Back propagation	Geri yayılım
Bifurcation	Çatallanma
Biometric authentication system	Biyometrik doğrulama sistemi
Core	İç nokta
Correlation Classifier	İlinti Sınıflandırıcısı
Enclosure	Halka
Feature extraction	Öznitelik bulma
Fork	Çatal
Gradient	Gradyan
Hand geometry	El geometrisi
Hand vein	El damarı
Island	Ada
Kernel Fitting	Öz yaklaştırma
Least square estimator	En küçük karesel kestiricisi
Left loop	Sol döngü
Minutiae	Tanıma ayrıntıları
Neighborhood Classifier	Komşuluk Sınıflandırıcısı
Papilla	Kabarcık
Ridge	Sırt
Ridge ends	Sırt sonu
Right loop	Sağ döngü
Self organizing mapping	Öz düzenlemeli haritalama
Spatial tessellation	Uzamsal mozaik
Tented arch	Çadırlı kemer
Valley	Yarık
Voice print	Ses izi
Whorl	Sarmal
Continuous random processes	Sürekli rastgele süreçler
Discrete random processes	Kesikli rastgele süreçler
Basis vector	Taban vektörü
Orthonormal	Birim boylu

FINGERPRINT CLASSIFICATION APPLICATION USING SOM ALGORITHM IN FPGA

Yasemin Can

ABSTRACT

In this work the fingerprint classification is performed using the self-organizing mapping neural network algorithm. By modelling the self-organizing mapping algorithm the biological neuron behaviour is implemented in the FPGA as digital circuits. In addition, the processing time is aimed to be shortened by modelling the paralel processing characteristic of the human brain.

Classification is performed by unsupervised learning making use of self organizing mapping. Since the number of classes can be arranged according to the application, the database is aimed to be organized in the best possible structure.

Correct classification of the non-trained fingerprints is also aimed with unsupervised learning in this work.

Keywords: FPGA, Neural Network, Fingerprint.

Advisor: Asst. Prof. Dr. Ali Ziya ALKAR, Hacettepe University, Department of Electrical and Electronics Engineering, Electronics Section

SOM ALGORİTMASIYLA PARMAK İZİ SINIFLANDIRILMASININ FPGA UYGULAMASI

Yasemin Can

ÖZ

Bu tez çalışmasında, parmak izi sınıflandırılması yapay sinir ağı algoritmalarından öz-düzenlemeli haritalama yöntemiyle gerçekleştirilmiştir. Öz-düzenlemeli haritalama algoritmasının ve biyolojik nöron davranışının FPGA' de sayısal devrelerle gerçekleştirilmesi amaçlanmıştır. Buna ek olarak, insan beyninin paralel işleme özelliği modellenerek algoritmanın işleme zamanının kısaltılması hedeflenmiştir.

Öz düzenlemeli haritalama kullanılmasıyla sınıflandırma danışmansız öğrenme ile gerçekleştirilmiştir. Öz düzenlemeli haritalama yönteminde sınıf sayısı uygulamaya göre düzenlenebildiğinden, veritabanının en uygun yapıda düzenlenmesi hedeflenmiştir.

Danışmansız öğrenme ile daha önce eğitim aşamasında kullanılmayan parmak izlerinin de doğru sınıflandırılması hedeflenmiştir.

Anahtar Kelimeler: FPGA, Sinir ağları, Parmak izi.

Danışman: Yrd. Doç Ali Ziya ALKAR, Hacettepe Üniversitesi, Elektrik ve Elektronik Mühendisliği Bölümü, Elektronik Anabilim Dalı

1. GİRİŞ

Günümüzde parmak izi tanıma sistemi güvenlik birimleri uygulamalarının yanı sıra personel tanıma sistemleri gibi yaygın amaçlarla da kullanılmaya başlanmıştır. Tanınması gereken parmak izi sayısının az olduğu durumlarda kişisel bilgisayar tabanlı sistemler yeterli olurken profesyonel düzeydeki sistemlerde paralel işlem mimarileri kullanılır [1]. Her iki sistemde de veri tabanı sınıflandırma yöntemleriyle bölümlenerek arama süresi kısaltılmaya çalışılmıştır.

Kimlik belirleme çalışmalarında kullanılan biyometrik tanıma sistemleri arasında parmak izi tanıma sistemi oldukça güvenilir sonuçlar veren bir yöntemdir. Yüz tanıma, el geometrisi, retina, imza ve ses izi gibi biyometrik yöntemlerle parmak izi tanıma sistemi değerlendirilirse evrensellik, benzersizlik, değişmezlik, toplanabilirlik ve doğruluk gibi özelliklerde parmak izi tanımanın oldukça başarılı olduğu görülür [2].

Parmak izi sınıflandırmanın tarihi gelişimine bakıldığında ilk olarak, belirleyici faktör insan olmuştur. Belirleyici faktörün insan olduğu durumlarda sınıflandırma görsel olarak birbirine benzeyen parmak izlerinin aynı sınıflara dahil edilmesiyle gerçekleştirilmiştir. Daha sonra geliştirilen bilgisayar tabanlı sistemlerin bir kısmı da kullanılan bu görsel özelliklerin algoritmaya yansıtılmasıyla tasarlanmıştır.

Sınıflandırma yapay sinir ağları tarafından gerçekleştirildiğinde ve Danışmanlı Öğrenme algoritmaları kullanıldığında (Geri Yayılım gibi) yine görsel farklılıklar esas alınır. Bu tür sistemlerde sınıf sayısı sabittir ve belirlenen bu sınıflara göre eğitim aşaması gerçekleştirilir. Genellikle beş sınıfa ayrılan veritabanında arama gerçekleştirilir.

Öz Düzenlemeli Haritalama yöntemi gibi Danışmansız Öğrenme kullanılırsa, sınıf sayısı ağ ve sistem yapısına göre şekillenir. Sinir ağlarıyla gerçekleştirilmiş sınıflandırma sistemlerine göre, Öz Düzenlemeli Haritalama iki konuda diğer sistemlere göre avantaj sağlar. Bunlardan birincisi veri tabanında oluşturulabilecek sınıf sayısı sınırlı değildir. Sınıf sayısı sistemin özelliklerine göre düzenlenebilir.

Böylece hacimli veri tabanlarını beş sınıfa bölmek yerine daha çok sınıfa bölerek arama zamanı kısaltılabilir. İkinci avantajı ise sistemin eğitilmesi sırasında danışmana ihtiyaç yoktur. Sınıflandırma işlemi gerçekleştirilirken algoritma kendi sınıflarını kendisi oluşturur.

Bu tezde, parmak izi sınıflandırılması Danışmansız Öğrenme algoritmalarından Öz Düzenlemeli Haritalama ile yapılacaktır. Sınıflandırma öncesi parmak izi işlenerek sınıflandırma için giriş vektörleri oluşturulacaktır. Oluşturulan bu vektörlerle algoritma işleme alınacak ve sınıflandırma gerçekleştirilecektir. Yapay sinir ağları ile gerçekleştirilecek olan ve Danışmansız Öğrenme algoritmasının kullanıldığı sistemin sınıflandırmadaki başarısı belirlenecektir.

Sistemin tasarımında parmak izinin işlenmesi, sınıflandırıcı için gerekli giriş vektörlerinin oluşturulması MATLAB kullanılarak gerçekleştirilmiştir. Öz düzenlemeli haritalama yöntemi ile sınıflandırma işlemi ise FPGA kullanılarak gerçekleştirilecektir. FPGA kullanılarak öz düzenlemeli haritalama yönteminin sayısal devreler kullanılarak paralel olarak gerçekleştirilmesi amaçlanmıştır. Hacimli veritabanlarının kullanıldığı uygulamalarda sistemde sınıflandırılacak tüm yapıların eğitimden geçirilip karşılaştırma işlemlerinin yapılması doğru sınıflandırma oranını arttırır. Öz düzenlemeli haritalama yönteminde de ağ yapısının karmaşık olduğu durumlarda ve eğitim kümesi çok eleman içerdiğinde işlem süresi uzun zaman alır. Bu uygulamada ve buna benzer veritabanı sınıflandırma uygulamalarında sayısal devre kullanılarak paralel işlem gerçekleştirilerek zamanın kısaltılması amaçlanmıştır.

İkinci bölümde parmak izinin biyolojik yapısı, parmak izlerinin sınıflandırılıp karşılaştırılması hakkında genel bilgiler ve parmak izi çalışmalarının tarihçesi anlatılmaktadır.

Üçüncü bölümde, parmak izi sınıflandırılması detaylı bir biçimde anlatılmaktadır. Sınıflandırma için hazırlanan öznitelik bulma yöntemleri açıklandıktan sonra parmak

izi sınıflandırıcıları açıklanmıştır. Ayrıca bu bölümde bu çalışmada uygulanacak olan öz düzenlemeli haritalamanın algoritması ve uygulanışı anlatılmaktadır.

Dördüncü bölüm, bu tezde uygulanan sistemi anlatmaktadır. Öznitelik bulmanın hangi yöntemle ve nasıl yapıldığı, sınıflandırmanın gerçekleştirilmesi MATLAB ve FPGA simülasyon sonuçları bu bölümde anlatılmaktadır.

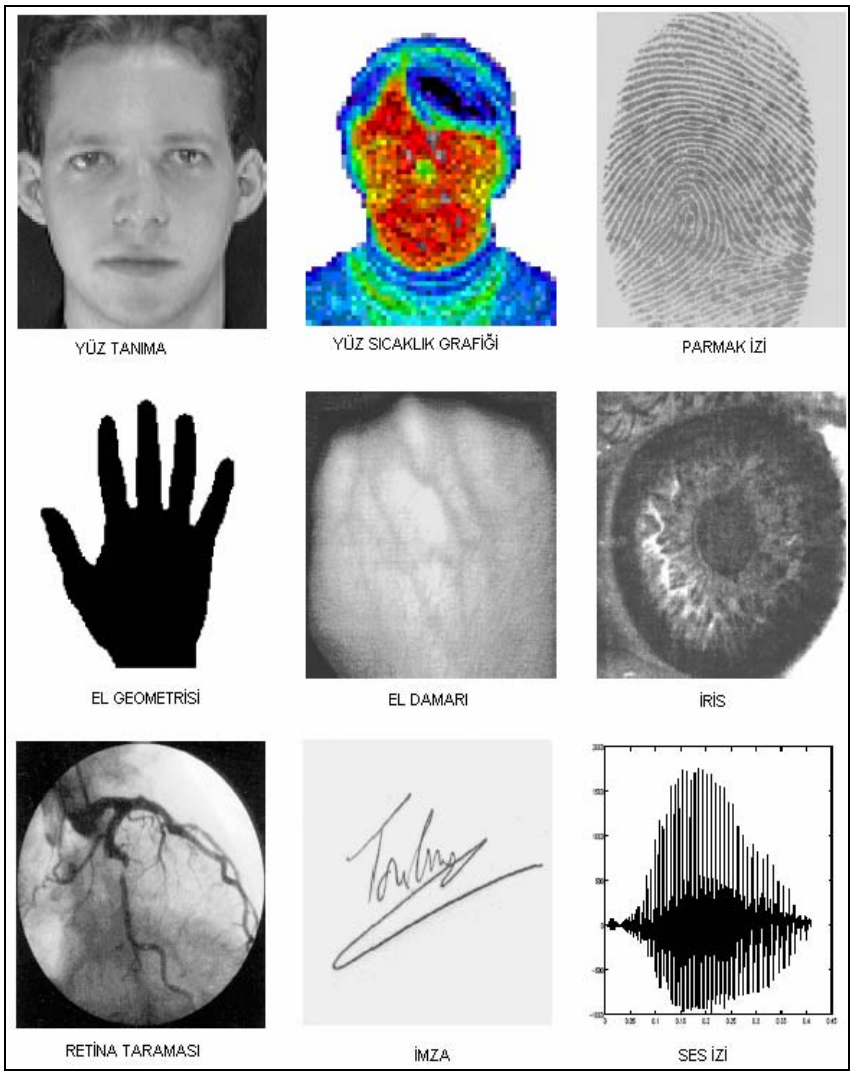
2. PARMAK İZİ HAKKINDA GENEL BİLGİLER

Çok eski çağlardan beri parmak izi bilinmektedir. Bazı arkeolojik kalıntılarda da rastlanan parmak izinin, teşhis amaçlı kullanılma fikri ise daha sonra keşfedilmiştir. Parmak izinin teşhiş amaçlı kullanılmasında biyolojik yapısı ve kendine has özellikleri önemli rol oynar. Kişiye özel olması doğru kimlik belirleme sağladığı gibi sınıflandırılabilr olması bilgisayarlı sistemlerde ve veri tabanı uygulamalarında kolaylık sağlar. 16. yüzyıldan bu yana parmak izi üzerine çalışmalar yapılmıştır. Sınıflandırma ve tanıma alt başlıklarıyla yürütülen bu çalışmalarda elde edilen başarı, parmak izini personel tanıma gibi günlük uygulamalarda karşımıza çıkarmaktadır.

Bilimsel anlamda parmak izi tekniklerinin temelleri 16. yüzyılda atılmıştır [4]. 1788 yılında Mayer tarafından parmak izinin anatomik oluşumu üzerine çalışmalar yayınlanmıştır [5]. Bu çalışmada parmak izinde oluşmuş sırt eğrileri incelenmiştir.

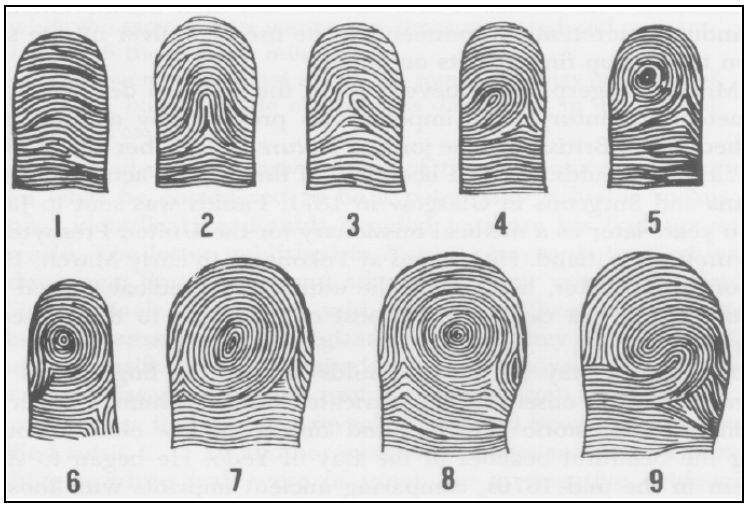
1823 yılında Purkinje parmak izlerini sırt-sağrı dizilimlerini referans alarak 9 temel sınıfa ayırarak ilk sınıflandırma şemasını oluşturmuştur [5]. Şekil 2.2' de bu sınıflar görülmektedir. İngiliz bitki morfolojisi uzmanı olan Nehemiah Grew 1864 yılında bu konuda ilk bilimsel makaleyi yayınlamıştır. Makalede sırt, yarık, por yapıları açıklanmıştır [6].

1809 yılında Thomas Bewick parmak izini ilk kez kendi markası olarak kullanmıştır [5]. Parmak izlerinin birbirine benzememe özelliğini ilk kez Henry Fault 1880 yılında, gözlemlerine dayanarak, öne sürmüştür [6]. Francis Galton 1888 yılında parmak izi sınıflandırılmasında kullanılmak üzere tanıma ayrıntıları kavramını açıklamıştır [4]. 1899 da Edward Henry tarafından "Henry System" adı verilen sınıflandırma oluşturulmuştur [6]. Bu sisteme göre beş sınıf belirlenmiştir. Bu beş sınıf şekil 2.3' de gösterilmiştir. Sınıflandırma işlemi için bu beş sınıf günümüz çalışmalarının pek çoğuna standart oluşturmuştur.

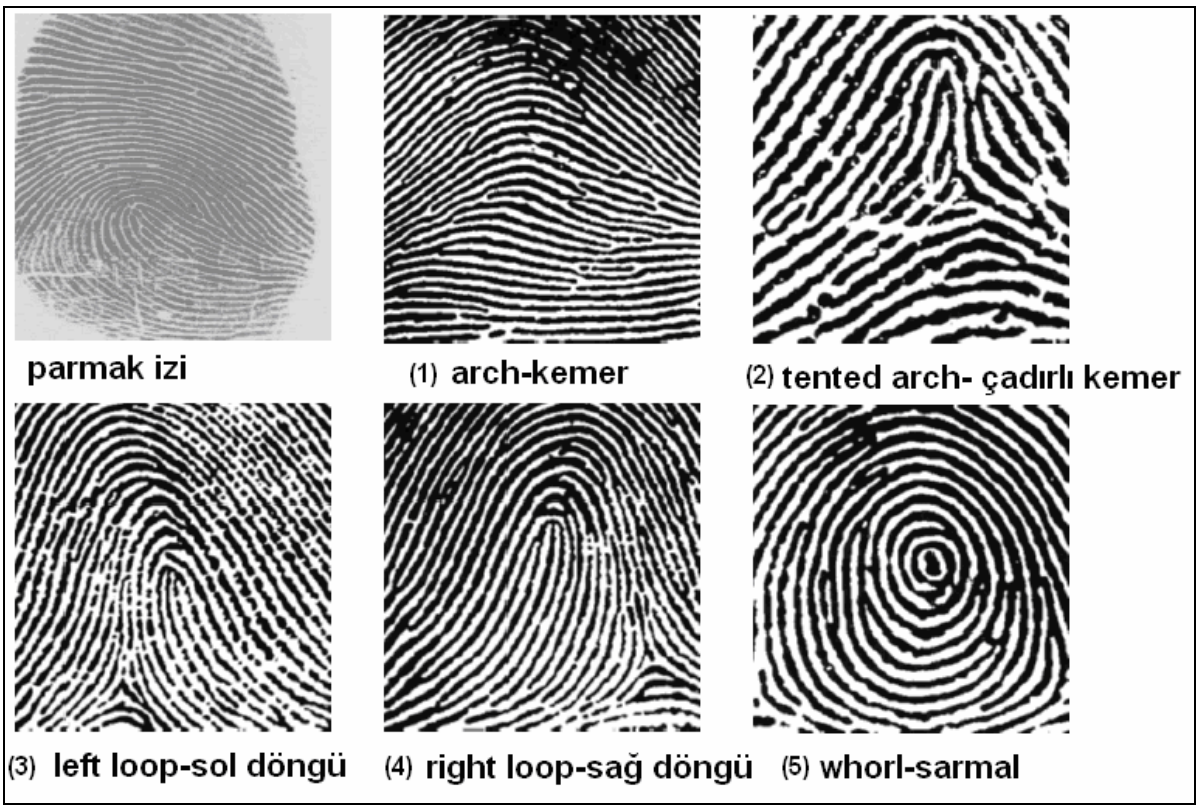


Şekil 2.1 Biyometrik yöntemler [2]

20 yüzyılda parmak izinin kimlik belirleme de oldukça güvenilir olduğu fikri yaygınlaşmıştır. Böylece parmak izi tanıma sistemleri oluşturmaya çalışan kuruluş sayısı artmıştır. Dünya çapında suçlulara ait parmak izi veri tabanları oluşturulmuştur. FBI 1924 yılında parmak izi tanıma bölümü kurmuştur. Bu bölümde 810.000 parmak izinden oluşan bir veritabanı mevcuttur [7].



Şekil 2.2 Purkinje sınıfları [5]



Şekil 2.3 Henry sınıfları [2]

1960 yıllarında FBI ve Paris Polis Teşkilatı otomatik parmak izi tanıma sistemi oluşturmak için çalışmalara başladı [7]. Bilgisayarların gelişimi ve yaygınlaşmasıyla birlikte parmak izi tanıma algoritmaları bilgisayarlara uyarlanmış kısa sürede sonuç alınacak duruma gelinmiştir.

Bilgisayar sistemleri kullanılarak yapılan sınıflandırma çalışmaların çoğunda veri tabanı olarak NIST4 (National Institute of Standarts and Technology) olarak isimlendirilmiş Amerikan standartlarını oluşturan enstitü tarafından kullanılan veritabanı kullanılmıştır.

Farklı yöntemlerle gerçekleştirilen bu çalışmalardan kural tabanlı sistemler üzerine 1984 yılında Kawagoe ve Tojo, 94 parmak izinin kullanıldığı bir veri tabanı ile %91,5 performans elde etmişlerdir [8]. Yine kural tabanlı uygulama ile Karu ve Jain 1996 yılında %85 [9] , Cang ise, 89 parmak izi içeren veritabanıyla, 1997 yılında %96,5 başarı elde etmiştir [10]. Hang ve Jain aynı yöntemle 1999 yılında %87,5 başarıya ulaşmıştır [11]. Aynı sınıflandırma yöntemi ile farklı performansların elde edilmesi sınıflandırma giriş vektörlerinin farklı yöntemlerle oluşturulmasından kaynaklanmaktadır. Komşuluk sınıflandırıcısı kullanıldığında Fritz ve Green, 40 parmak izinden oluşan sistemlerinde, 1996 yılında %85 başarı elde etmişlerdir [12].

Hidden Markov modelini sınıflandırmada kullanan Senior 1997 yılında %90 başarı elde etmiştir [9]. Yapay sinir ağları ile Wilson 1994 yılında %90,2 [13] , Candela 1995 yılında %92,2, Pal ve Mitra 1996 yılında %82 başarı sağlamıştır [9].

Sınıflandırma yöntemlerini birleştirerek Cappelli 2000 yılında %99 başarıya ulaşmıştır. Ancak bu sistem veritabanının %20 gibi bir oranını reddederek sınıflandırmaya almamıştır [14].

2.1 Parmak İzinin Biyolojik Yapısı

Vücudumuzu kaplayan deri, bilhassa el ve ayak parmaklarının iç yüzeylerinde bir kısım çizgiler meydana getirir. Muntazam aralıklarla dizilmiş olan bu çizgiler sıra ile bir alçak bir yüksek olmak üzere sıralanırlar. Kabartma bir çizgi şeklinde olan bu hatlara kabarcık hatları denir. Kabarcık hatları alt deri tabakalarında sıralanmış olan sinir ucu yumakları ve ter bezlerinin üst deriye kadar çıkmış olan ter çıkış deliklerinin yan yana bir yükseklik halinde dizilmelerinden ibarettir [3]. Ayrıca derideki gözenekler de parmak uçlarından ter, yağ ve sıvıların çıkmasını sağlayarak yüzeylere bu izlerin çıkmasına yardımcı olurlar, bu sıvılar nedeniyle parmak uçları devamlı nemlidir.

2.2 Parmak İzinin Özellikleri

Parmak izi biyometrik yöntemler arasında güvenilir sonuçlar veren bir tanıma yöntemidir. Diğer kimlik belirleme sistemlerine oranla evrensellik, benzersizlik, süreklilik, toplanabilirlik ve doğruluk gibi özellikler de parmak izini kullanmak iyi bir seçimdir. Diğer biyometrik yöntemlerden bir kısmı yüz tanıma, yüz sıcaklık grafiği, el geometrisi, el damarı, iris, retina taraması, imza ve ses izi Şekil 2.1' de gösterilmiştir. Bu yöntemlerin karşılaştırılması Çizelge 2.1' de gösterilmektedir.

Karşılaştırma çizelgesi incelenirse özelliklerin güvenilirliğinin yüksek, orta ve düşük olarak derecelendirildiği görülür. Kimlik belirleme işlemi için parmak izi kullanımının en uygun yöntemlerden biri olduğu anlaşılır.

Parmak izini uygulamalarda kullanılmasının ana sebepleri alt başlıklarda açıklanan değiştirilemez, birbirine benzemez ve sınıflandırılabilir olma özelliklerden ileri gelir.

2.2.1 Değiştirilemezlik özelliği

Alt deri tabakasında oluşmuş kabarcık hatları üst deri tabakasına aynen alt deri görünümünü verir. Herhangi bir şekilde üst deri tahrip olsa bile yeniden yapılanan üst deri aynı görünümü geri kazanır. Ancak alt deri tahribatları izin yok olmasına sebep

olabilir. Alt deri tahribatı ise oldukça nadir görülür ayrıca alt deride oluşup üst deriye yansıyan izler parmak izinin daha kolay tanınmasına yol açar [3].

YÖNTEM	BENZERSİZLİK	DEĞİŞMEZLİK	TOPLANABİLİRLİK	PERFORMANS
YÜZ TANIMA	*düşük*	*orta*	*yüksek*	*düşük*
EL GEOMETRİSİ	*orta*	*orta*	*yüksek*	*orta*
PARMAK İZİ	*yüksek*	*yüksek*	*orta*	*yüksek*
EL DAMARI	*orta*	*orta*	*orta*	*orta*
İMZA	*düşük*	*düşük*	*yüksek*	*düşük*
SES İZİ	*düşük*	*düşük*	*orta*	*düşük*
RETİNA TARAMASI	*yüksek*	*orta*	*düşük*	*yüksek*
İRİS	*yüksek*	*yüksek*	*orta*	*yüksek*
YÜZ SICAKLIK GRAFİĞİ	*yüksek*	*düşük*	*yüksek*	*orta*

Çizelge 2.1 Biyometrik sistemlerin karşılaştırılması [2]

2.2.2 Birbirine benzemezlik özelliği

Aynı eldeki parmaklardaki hatta ikiz kardeşlerdeki parmak izleri dahi birbirinden farklıdır.

2.2.3 Sınıflandırılabilirlik özelliği

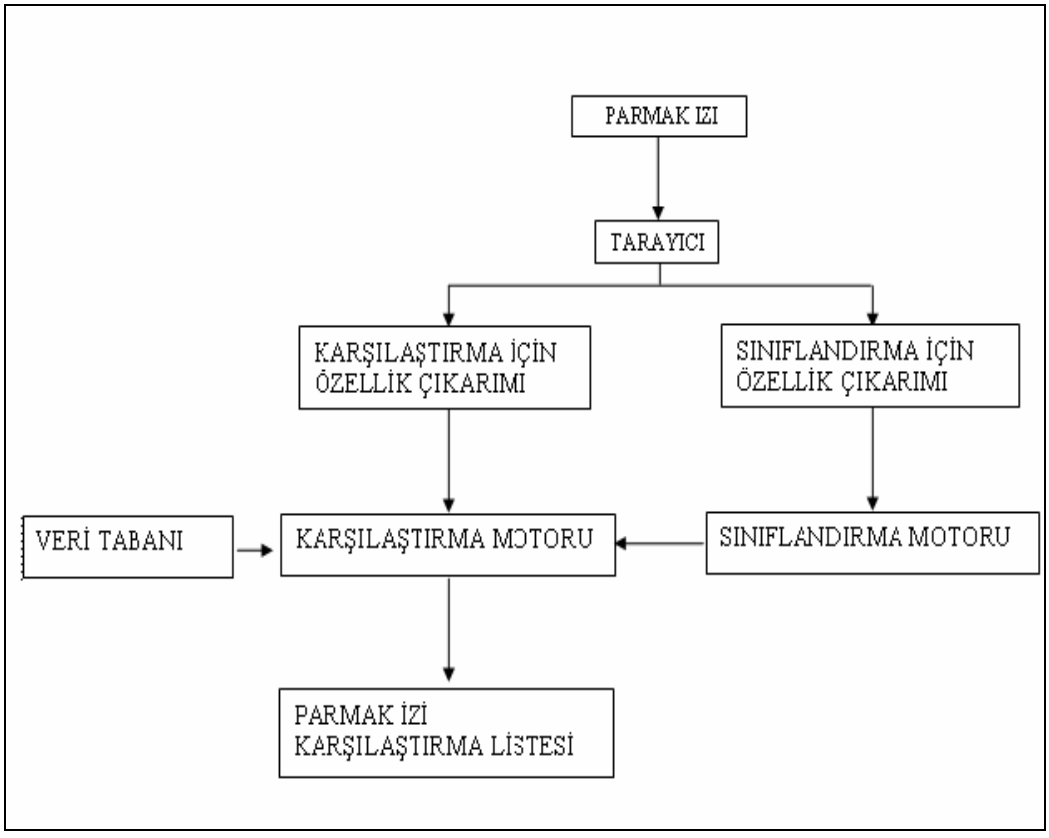
Parmak izleri birbirine benzememesine rağmen bazı temel benzerlikler sebebiyle sınıflandırılabilir. Doğru bir sınıflandırma ya da karşılaştırma yapmak için en az 12 nokta gerekir [2,3].

2.3. Parmak İzi Tanıma Aşamaları

Tarayıcıdan alınan parmak izinin sınıflandırma ve karşılaştırma için öznitelikleri bulunur. Öznitelik bulma farklı yöntemler kullanılarak gerçekleştirilir. Sınıflandırma öznitelik bulma yöntemleri üçüncü bölümde anlatılacaktır.

Parmak izleri öncelikle temel benzerliklerine göre sınıflandırılır, daha sonra bu sınıflandırmaya göre oluşturulmuş veri tabanında karşılaştırma yapılır. Karşılaştırma sonucunda olası sonuçlar listelenir.

Parmak izi tanıma sistemlerinin genel olarak gerçekleştirdikleri işlemler Şekil 2.4' de gösterilmiştir.



Şekil 2.4 Parmak izi tanıma işlem basamakları [1]

2.3.1 Parmak izi sınıflandırma yöntemi

Parmak izi teşhis amaçlı kullanılmaya başlanınca ve parmak izi sayısı çoğaldıkça veri tabanları oluşturulmuş, bazı karakteristik özelliklere göre sınıflandırma işlemi gerçekleştirilmiştir. Böylece çok sayıda parmak izi içeren veritabanları bölümlenerek arama zamanı kısaltılmıştır.

Farklı yöntemlerle sınıflandırılan veritabanları kullanılan yönteme göre farklı performanslar sergilemiştir. Örneğin yön dizilim vektörlerinin sınıflandırmaya giriş olarak kullanılmasıyla İlinti sınıflandırıcısı, Komşuluk sınıflandırıcısı ve Yapay Sinir Ağı sınıflandırıcı performansları sırasıyla %90, %85 ve %90 olarak bulunmuştur [1]. Bu teknikler sınıflandırma yaparken genellikle Henry sınıflandırıcısını esas alırken Yapay sinir ağı sınıflandırıcılarının danışmansız öğrenme algoritmalarından olan öz düzenlemeli haritalama, kendi sınıf sayısını algoritma özelliklerine göre oluşturur.

2.3.1.1 Henry sınıflandırması

Parmak izi tanıma sistemlerinin çoğunluğunda Henry tarafından öne sürülen bu sınıflandırma kullanılmıştır. Buna göre parmak izleri daha önce tarihçe kısmında anlatıldığı gibi beş sınıfa ayrılmıştır. Bu sınıflar kemer, çadırlı kemer, sağ döngü, sol döngü ve sarmal olarak adlandırılmıştır [15].

Bu sınıflardan kemer ve çadırlı birbirine çok yakın olduğu için, zaman zaman dört sınıf olarak da değerlendirmeler yapılmıştır. Bazı durumlarda dört sınıf uygulaması, danışmanlı sinir ağı algoritmaları tarafından gerçekleştirilen sınıflandırma performansını arttırmıştır.

İnsanların yaklaşık %60 – 65 gibi bir oranı döngü parmak izi karakteristiğine sahiptir, sarmal karakteristiği insanların %30 – 35 gibi bir oranında bulunur. %5 gibi bir oranı da kemer parmak izi karakteristiğine sahiptir [5].

Belirlenmiş bu beş sınıfa ait sınıflandırma giriş vektörleri sınıflandırıcılarda işlenerek, veritabanları bölümlenmiştir. Örneğin Yapay Sinir Ağlarından Danışmanlı Öğrenme kullanıldığında sınıflandırma giriş vektörleriyle ağ eğitimi gerçekleştirilir. Eğitim aşaması biten sistem eğitim aşamasında kullanılmamış parmak izlerini sınıflandırır.

2.3.1.2 Öz Düzenlemeli haritalama sınıflandırması

Yapay sinir ağı algoritmalarından Öz düzenlemeli haritalama algoritması sınıflandırma yaparken Henry sınıflarını kullanmaz kendi sınıflarını eğitim aşamasında oluşturur.

Sabit sayıda sınıf yoktur, sistem düzenlenmesi danışmana ihtiyaç duymadan gerçekleştirilir. Bu yöntemle sınıflandırma dördüncü bölümde detaylı olarak incelenecektir.

2.3.2 Parmak izi karşılaştırma yöntemi

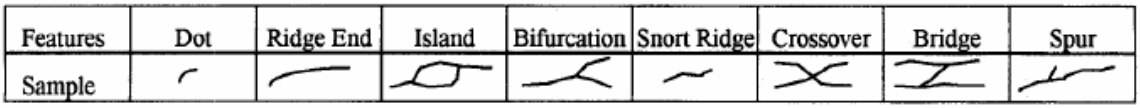
Parmak izini asıl tanınması yani benzersizlik özelliği kullanılarak test edilmesi işlemine karşılaştırma denir. Bu işlem uygulanırken yerel sırt yapıları ve bunların birbirleriyle olan bağlantıları temel alınır.

Parmak izi karşılaştırmada üç yöntem dikkati çeker. Sağrı yönüne göre, sağrı dizilimlerine göre ve “Karhaunen-Loeve” dönüşümüne göre karşılaştırma yapılır [1].

Parmak izlerinin asıl karşılaştırılması ise Galton Özellikleri adı verilen kriterlere göre yapılır [4]. Parmak izinde kabarcıkların oluşturduğu sürekli ve kesik çizgilere sırt (ridge) denir. Sağrılardan oluşan geçiş ve bitiş noktalarına tanıma ayrıntıları

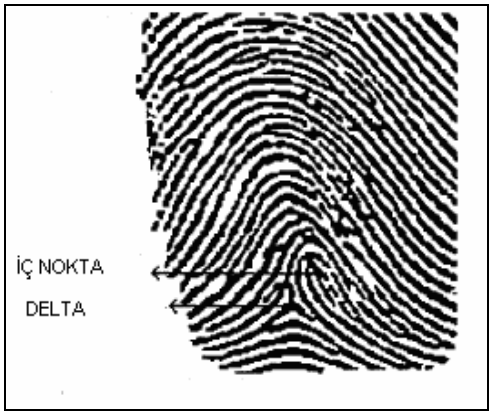
(minutiae) denir. Bu tanıma ayrıntıları kullanılarak yapılan karşılaştırma işlemine Galton tekniği adı verilir.

Temel olarak dört tane tanıma ayrıntısı belirlenmiştir. Bunlar, sırt sonu (ridge end), çatal (fork), ada (island) ve halka (enclosure) olarak adlandırılır [1]. Daha sonraki çalışmalarda bu belirleyici noktalar arttırılmıştır ancak bunların pek çoğu karşılaştırma işlemlerinde kullanılmaz. Belirleyici noktaların tümü Şekil 2.5' de gösterilmiştir.



Şekil 2.5 Parmak izi karşılaştırmasında kullanılan belirleyici noktalar [1]

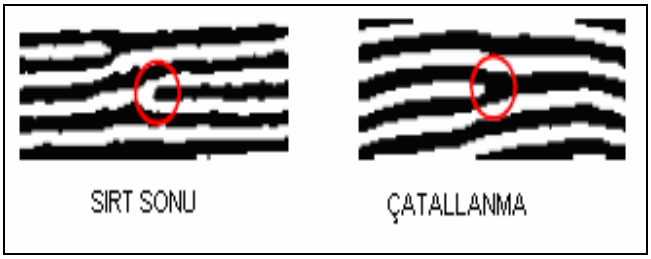
Karşılaştırmada ayrıca iç nokta (core) ve delta noktaları denilen Şekil 2.6' da gösterilen noktalar da belirleyici olur. İç nokta genellikle referans noktası olarak kullanılır. Tanıma ayrıntıları kodlamasında en içteki tepe noktası gibi algılanır.



Şekil 2.6 Merkez noktası [1]

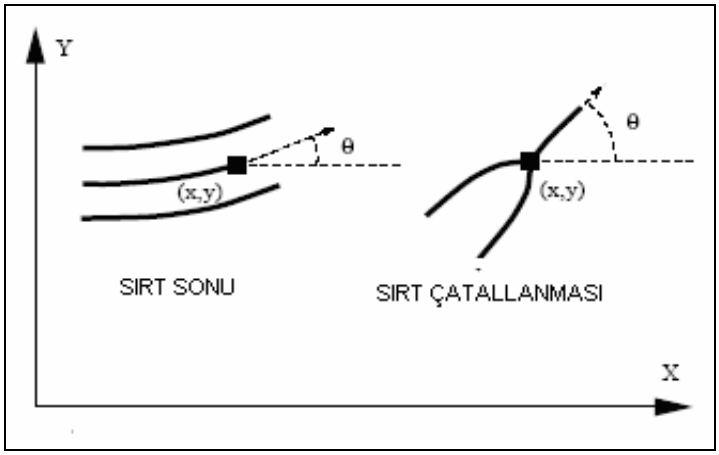
Delta noktası ise birbirine yakınlaşan sırt çizgilerinin birbirinden ayrılmaya başladığı nokta olarak tanımlanabilir.

Sırt sonu ve çatallanma karşılaştırmada en çok kullanılan iki özelliktir. Bu özellikler Şekil 2.7' de gösterilmiştir.



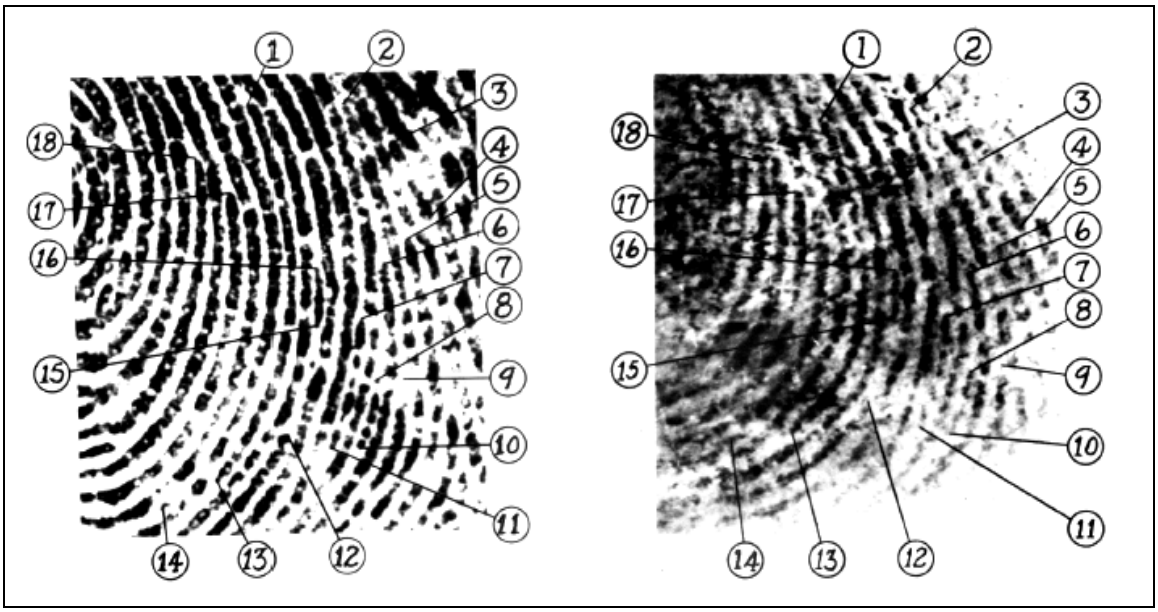
Şekil 2.7 Karşılaştırmada en çok kullanılan iki özellik [2]

Bu özelliklerin karakterize edilişi aşağıdaki şekilde görüldüğü gibi koordinatlar ve açı değerleriyle tanımlanabilir. Şekil2.8' de özelliklerin kartezyen ve açısal tanımlanması gösterilmektedir.



Şekil 2.8 Karakteristik özelliklerin kartezyen ve açısal tanımlanması [2]

Eğer iki parmak izi aynı sınıftaysa ve yeterince tanıma ayrıntıları noktası birbiriyle eşleşiyorsa bu iki parmak izinin aynı kişiye ait olduğu kabul edilir. Eşleştirilmiş iki parmak izi Şekil 2.9' da gösterilmektedir.



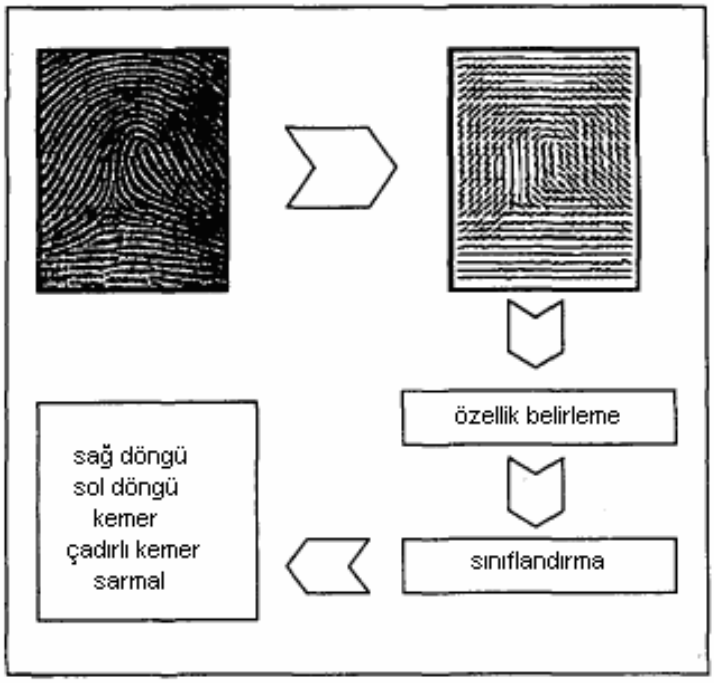
Şekil 2.9 İki parmak izinin karşılaştırılması [5]

3. PARMAK İZİ SINIFLANDIRMA

Parmak izlerinin sınıflandırılmasında öncelikle sınıflandırmayı yapmak için kullanılacak olan öznitelik bulunmalıdır. Ardından bulunan öznitelikler sınıflandırma işlemine giriş olarak uygulanır ve sınıflandırma yapılır. Şekil 3.1' de sınıflandırma işlem basamakları gösterilmektedir.

Sınıflandırmanın doğru yapılabilmesi kullanılacak giriş özelliklerine de bağlıdır. Bu nedenle iyi bir özellik belirleyici kullanılması sistemin performansını arttırır.

Özellik belirleyici yönteminin yanı sıra kullanılan sınıflandırıcı da sistemin performansını etkiler. Yapılan çalışmalarda zaman zaman birden fazla sınıflandırıcı yöntemi birleştirilerek zaman kısaltılmaya çalışılmıştır. Shan ve Tang 2004 yılında böyle bir girişimde bulunarak önce beş sınıfa ayırdıkları veritabanını daha sonra filtre tabanlı yaklaşımla tarayarak arama süresini %95 kısalttıklarını açıklamışlardır [16].



Şekil 3.1 Sınıflandırma işlemi

Parmak izi sınıflandırma üzerine bu güne kadar yapılan çalışmalarda öznitelik bulma yöntemlerini üç gruba ayırabiliriz. Bunlar filtre tabanlı, yapısal tabanlı ve istatistiksel tabanlı yaklaşımlardır [22].

3.1 Öznitelik Bulma

3.1.1 Filtre tabanlı yaklaşım

Bu yöntemle parmak izine ait genel ve yerel özellikler bulunabilir. Küçük boyutta taranmış parmak izlerinde, merkez noktası tam olarak belli olmayan izlerde başarılı sonuçlar elde edilmiştir [16].

Bu yöntemde genellikle tercih edilen yön dizilimi yerine, filtreleme yöntemi kullanılır. İlk etapta bir referans noktası belirlenir ve bu referans noktası merkez alınarak sektör ve bantlardan oluşan bir uzamsal mozaik oluşturulur. Referans noktası kemerli yapı için en içteki eğridir. Sol döngü, sağ döngü, çadırlı kemer için en içteki daire, iki merkezli sarmal yapılarda ise bu iki merkezin ortası olarak belirlenir. Giriş görüntüsünün 0, 45, 90 ve 135 derece dizilimli Gabor filtresiyle işlenmesi sonucu dört bileşenden oluşan görüntüleri elde edilir. Her sektör için hesaplanan gri düzeyi standart sapması özellik vektörü olarak belirlenir ve sınıflandırıcıya giriş olarak uygulanır. Bu şekilde oluşturulmuş özellik vektörüne FingerCode adı verilir. FingerCode oluşturma işlem basamakları aşağıdaki şekilde özetlenebilir.

1. Referans noktası ve bu noktaya bağlı olarak ilgili alan belirlenir.
2. Merkez noktası etrafında eş merkezli, bölümlenmiş bantlarla ilgili alan taranır.
3. Her sektör belirli bir sabit ortalama ve varyans değerine normalize edilir.
4. İlgili alana farklı açılarla Gabor filtresi uygulanır.
5. Her sektör için ortalama mutlak standart sapma hesaplanır.

Şeki l3.2’ de işlem basamakları şema ile açıklanmıştır.

Filtreleme işleminden sonra oluşturulan gri düzey 192 boyutlu FingerCode değişik sınıflara ait referans FingerCode değerleri ile karşılaştırılarak sınıflandırma yapılır. Sınıflara ait FingerCode şemaları Şekil 3.3' de gösterilmektedir.

Parmak izine Gabor filtresi uygulandığında sırt ve yarık bölgeleri vurgulanmış olur.
Filtre tabanlı yaklaşım basamakları incelenirse, öncelikle merkez noktası bulma algoritmalarından birisi kullanılarak görüntünün ilgili alanını oluşturmak için referans noktası belirlenir.

Merkez noktası bulunduktan sonra uzamsal mozaik alanının anlamlı olması için merkez noktasının altında bir nokta taranacak alanın merkezi olarak belirlenir. MxN boyutlu bir görüntüde (x,y) koordinatlarıyla belirtilen piksel değeri $I(x,y)$ ile merkez noktası (x_c, y_c) ile ifade edilirse uzamsal mozaik bölgesinin r ve θ değerlerine bağlı olarak tanımı eşitlik (3.1) ile gösterilmiştir. S_i Uzamsal mozaik sektörlerini ifade eder.

$$S_i = \{(x,y) \mid b(T_i+1) \le r < b(T_i+2), \qquad \theta_i \le \theta < \theta_{i+i}, \quad 1 \le x \le N, \quad 1 \le y \le M \qquad (3.1)$$

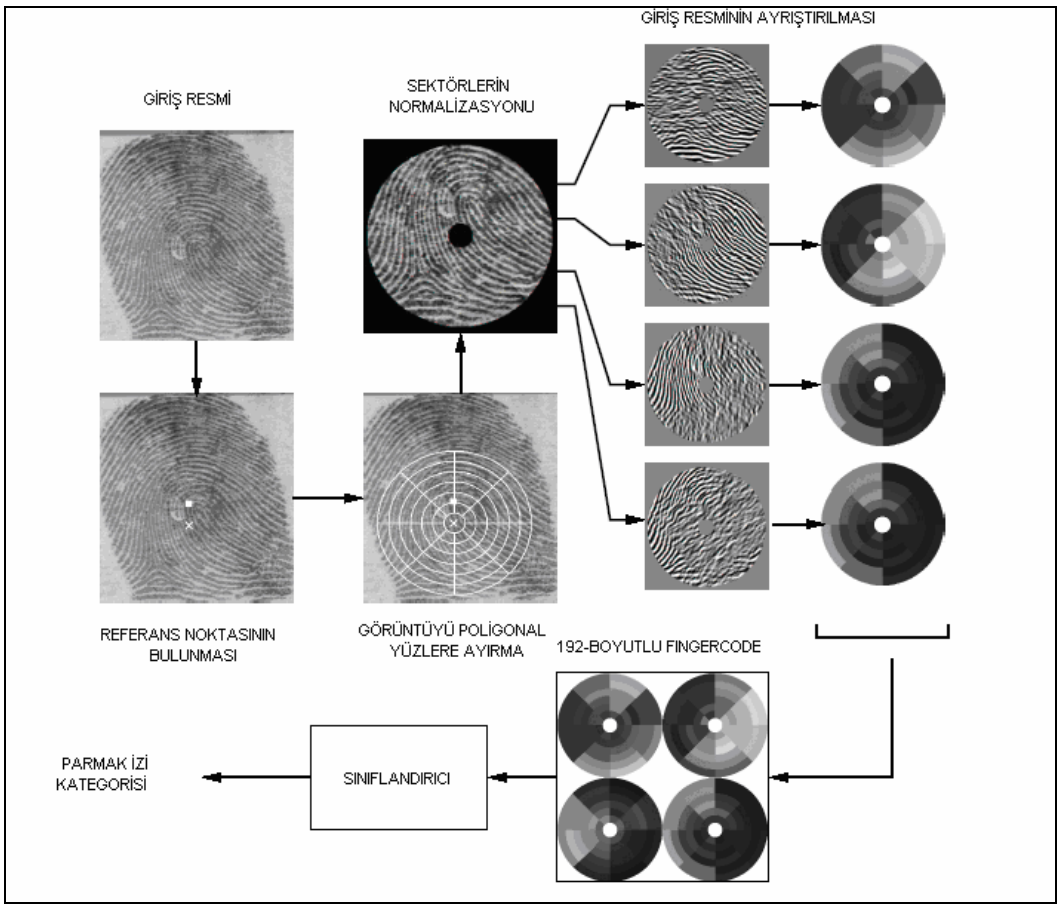
$$T_i = i \quad b\ddot{o}l\ddot{u}m \quad k \qquad (3.2)$$

$$\theta_i = (i \quad \bmod \quad k)(\frac{2\pi}{k}) \qquad (3.3)$$

$$r = \sqrt{(x-x_c)^2 + (y-y_c)^2} \qquad (3.4)$$

$$\theta = \tan^{-1}\left(\frac{y-y_c}{x-x_c}\right) \qquad (3.5)$$

b bant genişliğini, k sektör sayısını ifade eder.



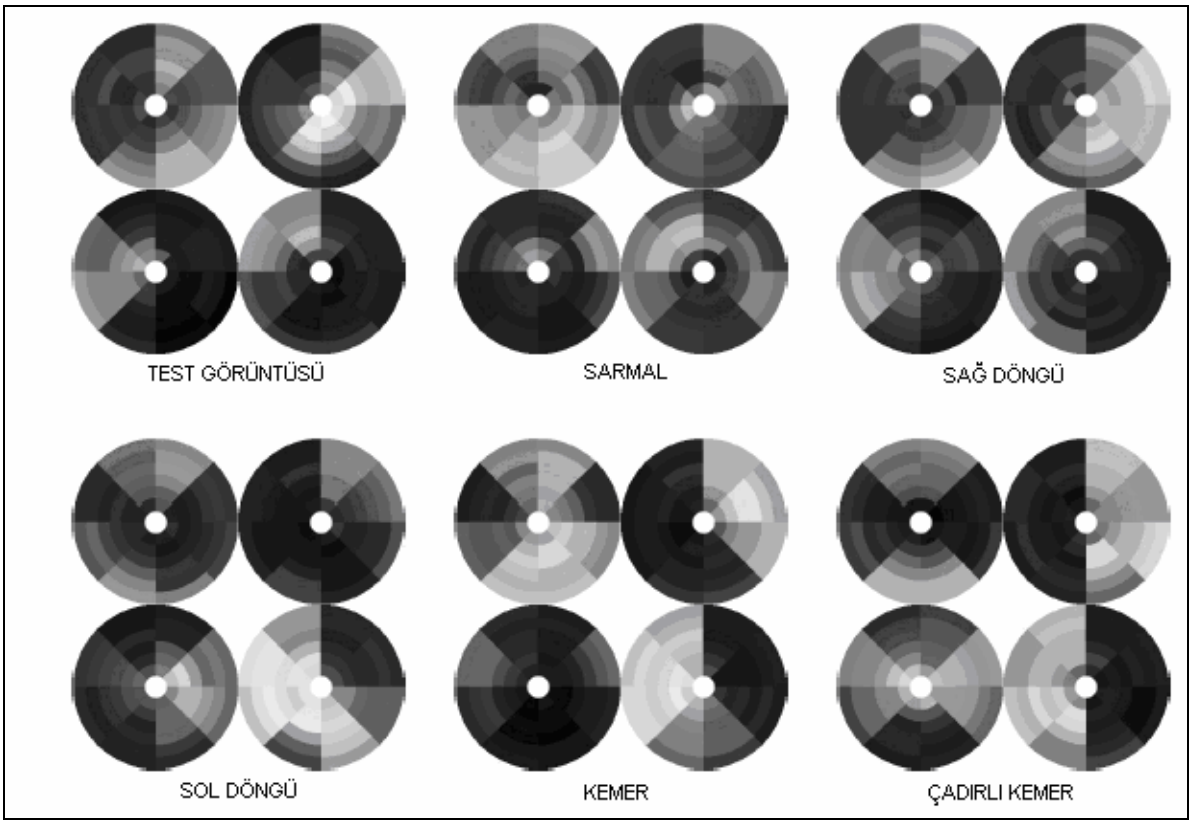
Şekil 3.2 Filtre tabanlı yaklaşım işlem basamakları [9]

Gabor filtrelerinin uzamsal tanım kümesindeki genel gösterimi eşitlik (3.6), (3.7) ve (3.8) ile gösterilmiştir.

$$G(x,y;f,\theta) = \exp\left\{\frac{-1}{2}\left[\frac{x'^2}{\delta_x^{\,2}} + \frac{y'^2}{\delta_y^{\,2}}\right]\right\}\cos(2\pi f x') \tag{3.6}$$

$$x' = x\sin\theta + y\cos\theta \tag{3.7}$$

$$y' = x\cos\theta - y\sin\theta \tag{3.8}$$



Şekil 3.3 Parmak izi sınıflarına göre FingerCode gri düzey görüntüleri [9]

f, θ değerinden x-eksenine olan sinüs düzlem dalgasının frekans değeridir.
f filtrenin frekansını belirler. K sırt çizgileri arasındaki piksel bazında uzaklık olarak tanımlanırsa f için ideal değer 1/K olarak alınır.

δ_x ve δ_y ise x ekseni ve y ekseni için Gaussian zarfı ifade eder, bu değerler aynı zamanda Gabor filtresinin bant genişliğini belirler.

δ_x ve δ_y değerleri çok büyük olursa gürültü faktörü azaltılmış olur, ancak sırt ve yarıklardaki detaylar kaybolur. Bu değerler küçük seçilirse gürültü nedeniyle resim net olmaz. Parmak izi x eksenine 0, 45, 90 ve 135 derecelik açılarla dört bileşenden oluşan Gabor filtreleriyle işlenir. Resim x ekseni ile 0 derece açı yapan filtre ile işlendiğinde resmin x eksenine paralel sırt çizgileri vurgulanır. Diğer filtrelerde de yine

aynı şekilde x eksenine yaptıkları açıya göre bileşenin belli kısımları vurgulanır. Daha sonra bu dört görüntünün birleştirilmesiyle yeniden yapılanan resim netleştirilmiş olur.

Görüntünün ilgili alanı belirlenirken her sektörün ayrı ayrı sabit ortalaması ve varyansı hesaplanır. Normalizasyon işlemiyle sensörden kaynaklanan gürültü faktörü azaltılır. $I(x,y)$ (x,y) koordinatlarıyla belirtilen piksel değeri M_i tahmin edilen gri düzey ortalaması ve V_i tahmin edilen varyans olarak kabul edilirse. S_i İlgili sektör , $N_i(x,y)$ (x,y) koordinatlarıyla belirtilen piksele ait gri düzey değeri olduğunda, M_0 istenen ortalama V_0 istenen varyans ise her sektör için normalize değeri eşitlik (3.9) ile gösterilir.

$$N_i(x,y)=\begin{cases} M_0+\sqrt{\dfrac{(V_0)x(I(x,y)-M_i)^2}{V_i}} & eğer \quad I(x,y)>M_i \\ M_0-\sqrt{\dfrac{(V_0)x(I(x,y)-M_i)^2}{V_i}} & diger \quad durumlarda \end{cases} \qquad (3.9)$$

Normalizasyon sırt ve yarık yapılarını değiştirmeyen piksel düzeyinde bir işlemdir. Normalizasyon işleminin tüm ilgili alana aynı anda değil de her sektör için ayrı ayrı uygulanmasının sebebi görüntüdeki farklı bölgelerdeki koyuluk açıklık farkının tüm resim için hesaplandığında yanlış değerlendirmelere yol açabilmesidir.
İlgili alandaki komşuluk değişimleri yerel sırt ve yarık yapılarının karakterleri hakkında bilgi içerir bu bilgi standart sapma değeriyle tanımlanır. Özellik vektörünü standart sapma değerleri oluşturur. S_i İle tanımlanan sektörlerin θ değerine göre bileşeni $C_{i\theta}(x,y)$ ile ifade edilirse ve standart sapma $F_{i\theta}$ ile gösterilirse standart sapma hesaplaması eşitlik (3.10) ile gösterilir.

$$F_{i\theta} = \sqrt{\sum_{Ki}(C_{i\theta}(x,y) - M_{i\theta})^2} \qquad (3.10)$$

$\forall \quad i \quad için \quad i = 0,1,.....47$

$\theta \quad \in \quad [0°,45°,90°,135°]$

K her sektördeki piksel sayısı

$M_{i\theta}$ Ortalama piksel değeri

Bu hesaplamaların sonucunda FingerCode oluşturularak sınıflandırıcıya giriş vektörü hazırlanmış olur.

Shan and Tang 2004 yılında birleştirilmiş bir yöntem uygulamışlardır [16], önce beş sınıfa ayırdıkları veritabanını daha sonra filtre tabanlı yaklaşımla tarayarak, arama süresini %95 kısalttıklarını açıklamışlardır.

Bu yöntemle öznitelik bulma çıkarımını ayrıca Jain ve Salil çalışmalarında kullanmışlardır [10,20]. Bu çalışmada FingerCode sonuçları yapay sinir ağlarıyla sınıflandırılmıştır.

Dinov Batra, Grish Singhol va Santau Chaudrey yine bu yöntemle özellik çıkarımı yapmışlar ve komşuluk sınıflandırıcısıyla FingerCode değerlerini işlemişlerdir. 257 parmak izinden oluşan veritabanını doğru sınıflandırmada %98.81 başarı elde edilmiştir [18].

3.1.2 Yapısal Tabanlı Yaklaşım

Bu metodun temeli örüntülerin yapısını modelleyen ilişkisel grafiklere dayanır. Bu metotta resim önce bölümlenir, ilişkisel grafik oluşturulur ve sonra oluşturulmuş bu grafikle karşılaştırma yapılır.

Yön dizilim grafiği de bu yöntemde kullanılır. Önce bölümlenen görüntü de daha sonra her bölüm için yön dizilimi oluşturulur ve elde edilen grafik başlangıç

görüntüsünü yapısal olarak modellemiş olur. Bölümleme yapılırken aynı yöndeki vektörler bir bölüm oluşturacak şekilde yapılır. Bölümleme yapılırken dinamik sınıflandırma algoritması kullanılır. Bu algoritmada yön vektörlerine ait varyans değerlerinin minimum olduğu durumlar hesaplanır. Her bölüm için bir düğüm oluşturulur. Her bölüm bir diğerine bir ark ile birleştirilir.

Bu model parmak izinin topolojik yapısının bir özetidir ve farklı açılardan alınmış aynı parmak izlerinin tanınmasına olanak taşır. İlişkisel grafiği oluşturulmuş parmak izleri değişik sınıflara ait ilişkisel grafik şablonlarıyla karşılaştırılarak da sınıflandırma gerçekleştirilebilir

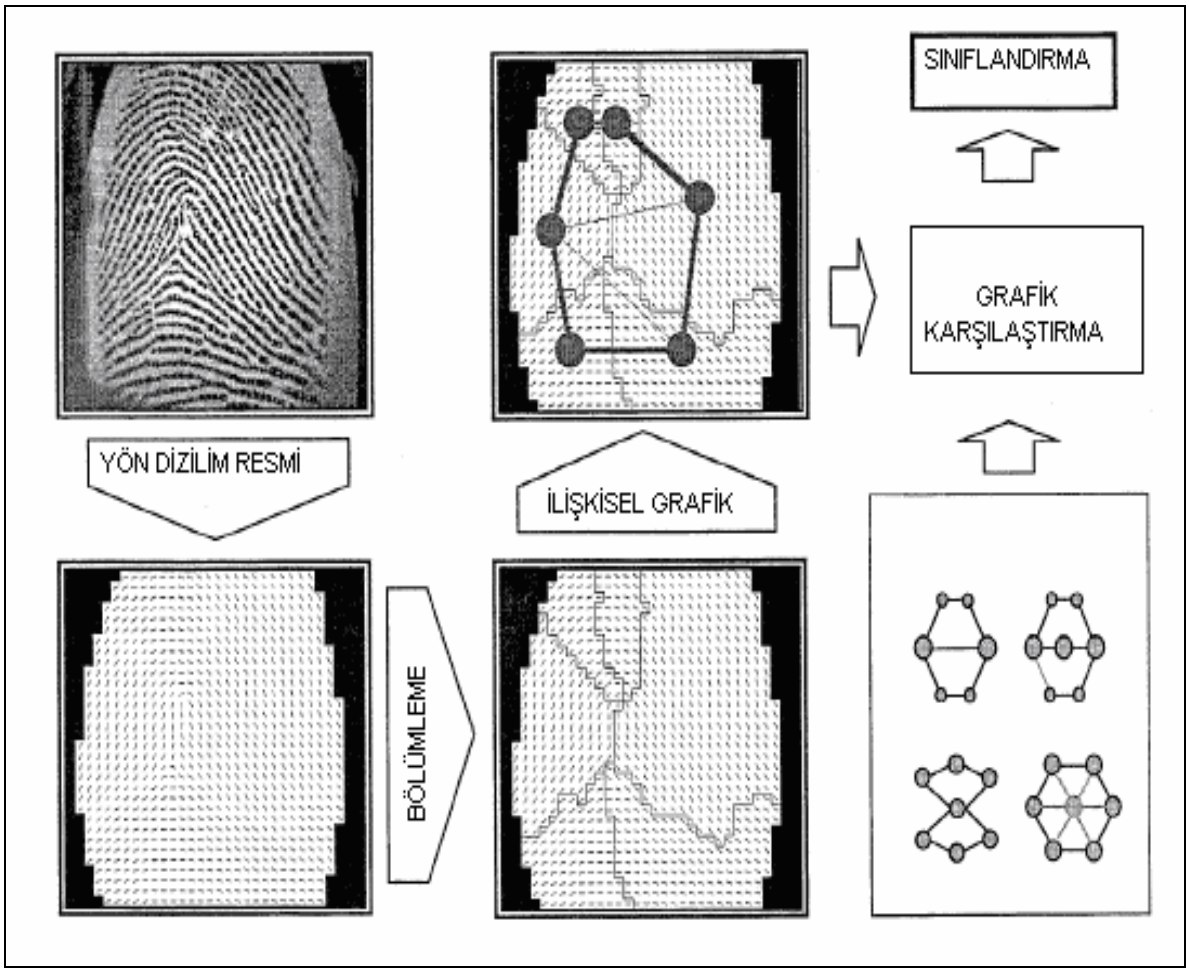
Karşılaştırma yapılırken grafikten elde edilen yapı ile şablon uzaklıkları kıyaslanır. Yapısal tabanlı yaklaşım bu şekilde sınıflandırma işleminde direk kullanılabilir. Gelişmiş yapısal tabanlı yaklaşımda ise şablonlara ait maliyet hesaplamaları yapılarak, hesaplanmış bu değerler sınıflandırıcıya giriş vektörü olarak kullanılır [19]. Şekil 3.4' de yapısal tabanlı yaklaşımın aşamaları gösterilmektedir.

Bu yaklaşım ilk kez Grasselli tarafından kullanılmıştır [20]. Daha sonra bu yöntemi Dario Maio ve Davide Maltoni sınıflandırma için kullanmışlardır [19].

Şablonların ve işlenen parmak izinin kıyaslanmasını Bunke ve Allermen yapısal bölümlenmiş grafik üzerinde tarama yaparak gerçekleştirmişlerdir [21].

Shapiro ve Horalick bu kıyaslamayı ilişkisel yapı yöntemiyle uygulamışlardır [22].

Kaul ise çalışmalarında uzaklık hesaplamalarını kullanarak şablonlarla ilişkisel grafik sonuçlarını karşılaştırmıştır [23].



Şekil 3.4 Yapısal tabanlı yaklaşıma örnek [19]

3.1.3 İstatiksel Tabanlı Yaklaşım

Bu yaklaşım istatistiksel anlamda toplanan bilgilerden öznitelik bulma vektörünün oluşturulmasını içerir.

3.1.3.1 Karhunen-Loeve (KL) Dönüşümü

Yapay sinir ağları ile oluşturulmuş sınıflandırma sistemlerinin çoğu öznitelik bulma işleminde KL (Karhunen-Loeve) dönüşümünü kullanır.

KL dönüşümü Karhunen ve Loeve tarafından sürekli rastgele süreçler için tanımlanmış seri açılımıdır [37]. Kesikli rastgele süreçler için Hotelling KL dönüşümünün kesikli seri açılımını yapmıştır. Gerçek bir MxN görüntüsü için 'U' KL

dönüşümü için taban vektörüdür. Kl dönüşümü görüntünün otokolerasyon matrisinini birim boylu öz vektörleri olarak ifade edilir.

U_{MxN} : Görüntü matrisi

u_n: Görüntünün n. Kolon vektörü U.

V_{MxN} : Dönüştürülmüş görüntü matrisi

v_n: Dönüştürülmüş görüntü matrisinin n. kolonu

R_u: Görüntünün otokolerasyon matrisi

R_v: Dönüştürülmüş görüntünün otokolerasyon matrisi

U görüntüsünün KL dönüşümü eşitlik (3.11) ile ifade edilir,

$$V_n = F^T u_n \tag{3.11}$$

F KL dönüşüm matrisi olarak tanımlanırsa ters KL dönüşümü eşitlik (3.12) ile ifade edilir.

$$U_n = \Phi v_n \tag{3.12}$$

Parmak izinin işlenmesi sonucu elde edilen istatistiksel bilgi işlenmeden önce bu yöntemle azaltılır. Bu yöntemle parmak izi görüntüsü hakkında edinilen anlamlı bilgi oranı yüksektir [22].

Yön dizilim vektöründen oluşturulacak öznitelik bulma vektörünü ele alırsak, her yön elemanı alt uzayda bir giriş olarak atanır bu yöntemle aynı bilgiyi veren elemanlar çıkarılarak boyut azaltılması sağlanmış olur [28,29]. Boyutta yapılan bu azaltma işlem süresini kısaltmaya yöneliktir.

3.1.3.2 Çok uzaylı Karhunen-Loeve (MKL) dönüşümü

KLT yönteminin genellenmiş ifadesidir. Örüntülerin ifade edilmesinde ve sınıflandırılmasında çok uzaylı yapı kullanılır [11]. KLT verinin yalnızca bir açıdan bilgisini taşırken MKL ile veriye ait birçok bilgi taşınabilir. Bu metot parmak izi yapılarında ayrımın daha net olmasını sağlar. Bu dönüşüme örnek olarak geliştirilmiş öz düzenlemeli haritalama çalışmasını verebiliriz [1]. Bu çalışmada yön vektörlerine ait açı bilgilerinin yanı sıra şiddet değerleri de sınıflandırıcıda kullanılır.

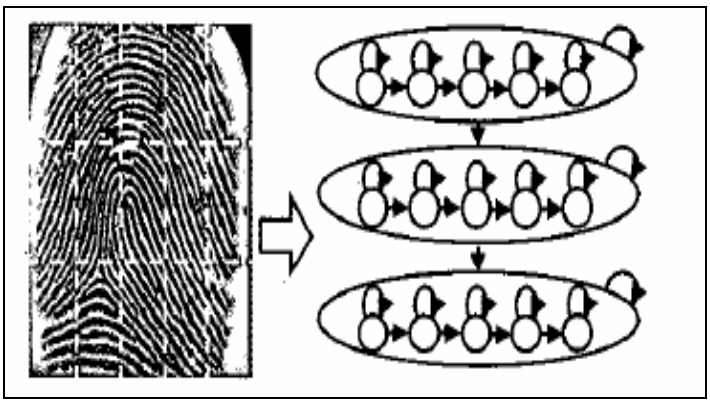
3.1.3.3 Kesişim yöntemi

Parmak izine ait görüntüdeki sırt eğrileriyle görüntünün iskeleti oluşturulur. Oluşturulan bu resimde sırt eğrisiyle kesişen her bir çizgi bir özellik olarak algılanır. Bu özelliklere ait bazı ölçümler belirleyici olarak kullanılır. Bunlar:

a) Bulunan son kesişim noktası ve anlık kesişim noktası arasındaki uzaklık
b) Anlık kesişim açısı
c) Son kesişim ve anlık kesişim arasındaki açı farkı
d) Anlık kesişim noktasındaki sırt eğrisinin kıvrımı

Bu yöntem toplam kesişim noktalarını bulur. Bu model Hidden Markov sınıflandırıcısında kullanılır [22]. Hidden Markov model veri dizilerindeki istatistiksel gözlemlere olanak sağlayan bir yöntemdir. Bu model bir durumdan diğerine geçişi ve gözlemlenen veri ile durumları arasındaki olasılıkları tanımlar. Hidden Markov Modeli istatistiksel olarak modellemek için, geçiş olasılık matrisi, başlangıç durumu olasılık dağılımı ve her durum için olasılık yoğunluk fonksiyonuna ihtiyaç vardır.

Parmak izi görüntülerinde bu model uygulanırken resim yatayda üç bölüme ayrılır. Yatay üç bölüm kendi içinde dikey olarak beş bölüme ayrılır. Her bölümün yön dizilim modeli çıkarıldıktan sonra bloklar için açı değerini içeren bir gözlem matrisi oluşturulur. Bu gözlem matrislerinden oluşan istatistiksel bilgiler Hidden Markov yöntemiyle sınıflandırma işleminde kullanılır. Şekil 3.5’ de parmak izinden çıkarılan istatistiksel bilgilerin oluşturulması gösterilmektedir.



Şekil 3.5 Hidden Markov modeli [10]

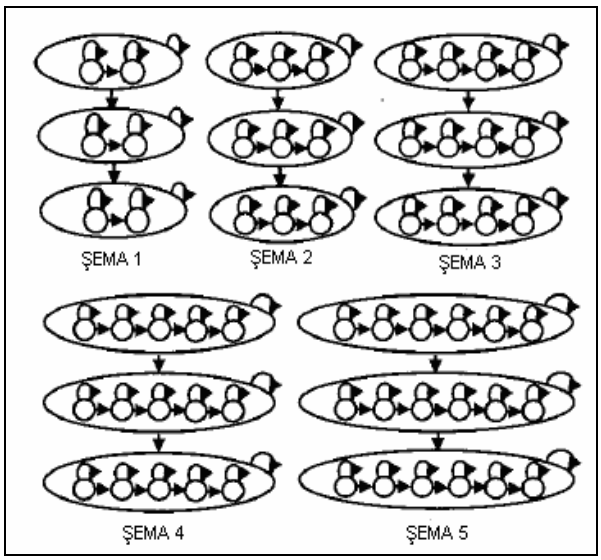
Parmak izinin işlenmesiyle elde edilen bilgiler Hidden Markov yöntemiyle oluşturulmuş Şekil 3.6' da gösterilen şablonlarla karşılaştırılır ve sınıflandırma yapılır.

3.1.4. Yaklaşımların Değerlendirilmesi

Filtre tabanlı yaklaşımda, parmak izine ait yerel ve genel özellikler ayrıntılarıyla belirlenebilmektedir. Bu yaklaşım gürültü faktöründen az etkilenmektedir. Gabor Filtrelerinin uygulanmasıyla resimde gürültü faktörü azaltılır böylece parmak izi yapısının netliği artar. Merkez noktası bulma bu yaklaşımda kullanılmasına rağmen kritik bir faktör değildir. Böylece merkez noktası belli olmayan parmak izlerinde ya da çok küçük boyutlardaki izlerde bu yaklaşımla özellik vektörü oluşturma sınıflandırıcı için avantaj sağlar. Diğer yaklaşımlara oranla daha fazla işlem gerektirir.

Yapısal tabanlı yaklaşımda tekillik noktalarının yeri ve özellikleri önemlidir. Resim kalitesinin iyi olduğu durumlarda bu yaklaşım sınıflandırıcı için avantaj sağlar. Yön dizilim görüntüsü oluşturularak özellik vektörleri oluşturulur. Gerekli işlem basamakları filtre tabanlı yaklaşıma göre daha azdır.

İstatistiksel yaklaşımda ise parmak izinden çıkarılan istatistiksel bilgiler kullanılır. Kullanılacak bilginin özelliğine ve belirleyiciliğine bağlı olarak sınıflandırma performansı değişir.



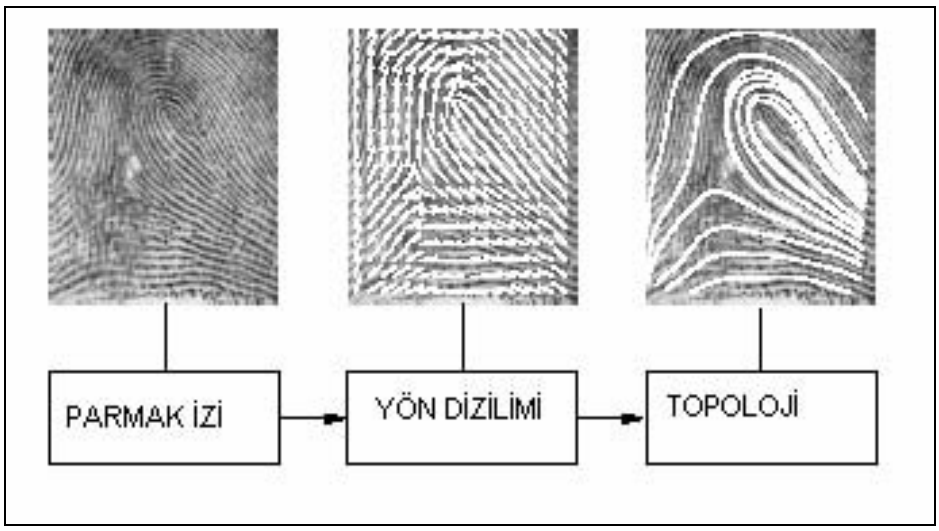
Şekil 3.6 Hidden Markov model şablonları [10]

3.2 Sınıflandırıcılar

3.2.1 Sözdizimsel sınıflandırıcılar

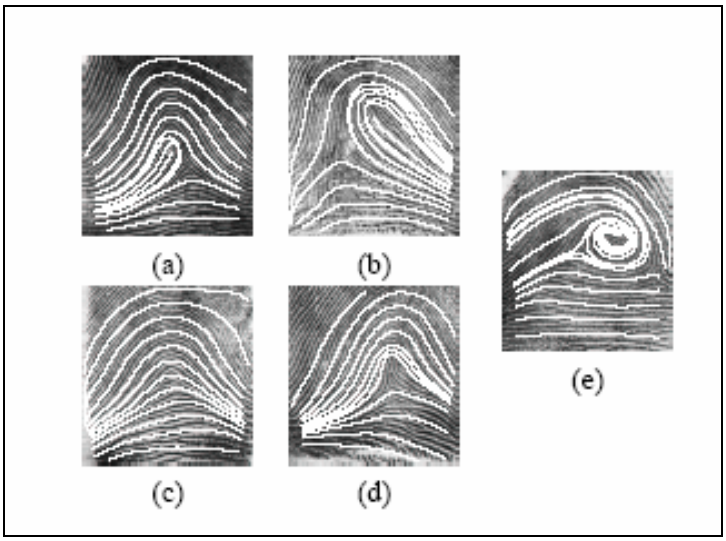
Geleneksel bir yöntemdir. Örüntüler bazı kurallara göre belirlenmiş geçiş sembolleriyle gösterilir. Bu geçiş sembolleri parmak izi için oluşturulmuş yön dizilim vektöründeki elemanlarla ilgilenir. Bu sembollerle oluşturulmuş görüntü, her sınıf için önceden oluşturulmuş görüntü şablonlarıyla karşılaştırılarak sınıflandırma yapılır.
Sarat ve Jain tarafından gerçekleştirilen yapısal-sözdizimsel uygulama ile sınıflandırmada %94,4 performans elde edilmiştir [26].

Bu uygulamada dört ana Henry sınıfı kullanılmıştır. Bu sınıflar sağ döngü, sol döngü, sarmal ve kemer yapılardır. Parmak izinin yön vektör görüntüsü oluşturulur. Daha sonra çeşitli algoritmalar kullanılarak parmak izinin sırt eğrileri, izin topolojik yapısını modelleyecek şekilde çıkarılır. Şekil 3.7' de bu işlem gösterilmektedir.



Şekil 3.7 Parmak izinin topolojik yapısının çıkarılması

Oluşturulan topolojik yapı önceden hazırlanmış şablonlarla karşılaştırılarak sınıflandırma gerçekleştirilir.



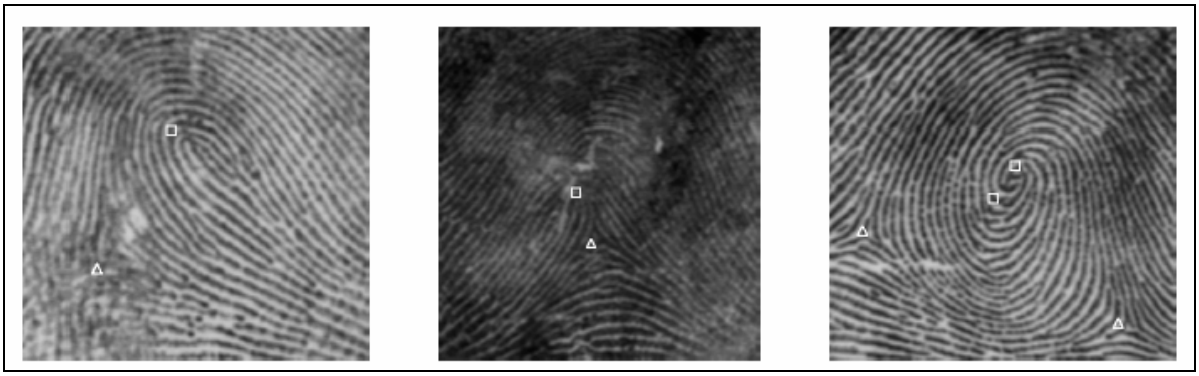
Şekil 3.8 Sınıflara göre oluşturulmuş şablonlar

3.2.2. Tekillik tabanlı sınıflandırıcılar

Tekil noktalarının sayılarına ve bulunma yerlerine bağlı olarak sezgisel yöntemlerle sınıflandırma yapılır. Ayrıca sırt eğrileri ve yerel yön dizilimleri, performansı arttırmak için sisteme ek olarak uygulanabilir.

Tekil tabanlı sınıflandırıcıların başarısı tekil noktaları denilen, delta ve iç noktaların konumlarının, doğru bulunmasına bağlıdır. Bu sebeple incelenecek parmak izinin görüntü kalitesi sistemin performansını doğrudan etkiler.

Resim kalitesinin düşük olması sebebiyle oluşabilecek hataları önlemek için tekil noktalarını sayan bir teknik geliştirilmiştir. Tekil noktaları sayısı sınıflandırma için anlamlı olana dek resim içinde tekil noktaları belirleme algoritması işlenir. Şekil 3.7' de iç noktalar kare, delta noktaları üçgenle işaretlenerek tekil noktaları gösterilmiştir.



Şekil 3.9 Tekil noktaları [2]

Tekil noktaları bulunduktan sonra Şeki l3.8' de gösterilen eğri çeşidi, delta ve iç nokta sayısı gibi kriterler parmak izinin hangi sınıfa ait olduğunu belirler.

Sınıflandırmayı belirleyen kriterler:

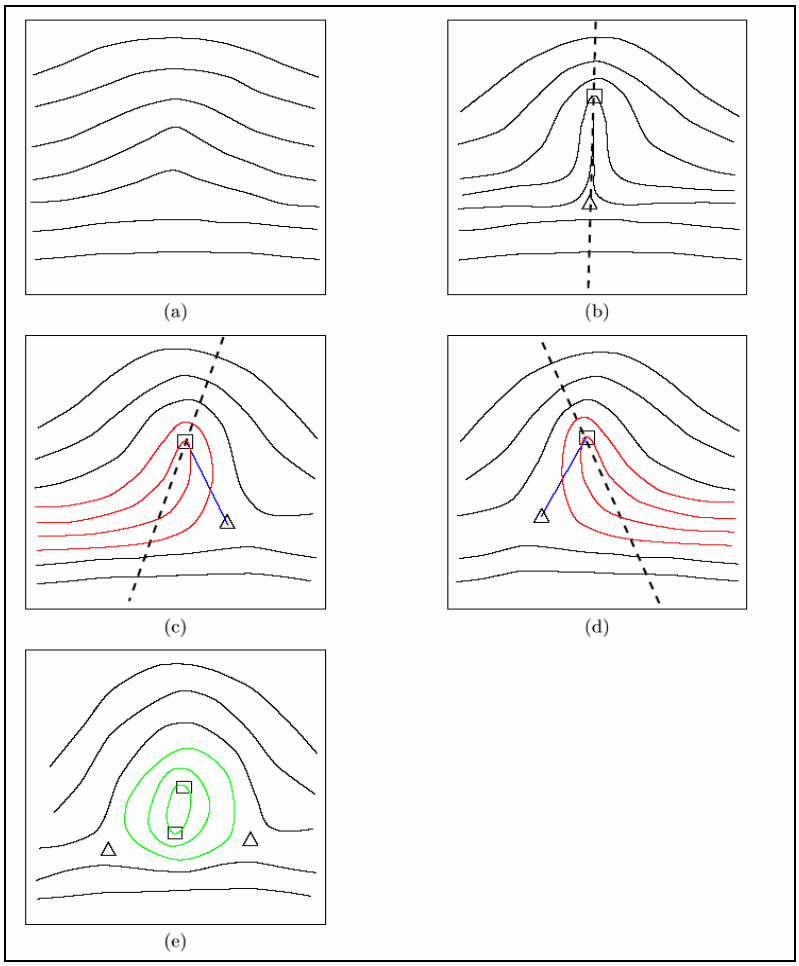
1. Daire sayısı >0, iç nokta sayısı =2, delta sayısı =2 olursa parmak izi sarmal yapıdadır.
2. Kapalı eğri sayısı=0, Daire sayısı=0, iç nokta sayısı=0, delta sayısı =0 olursa parmak izi kemer yapıdadır.
3. Kapalı eğri sayısı>0, Daire sayısı =0, iç nokta sayısı=1, delta sayısı=1 olursa iç nokta-delta değerlendirilmesi yapılacaktır.
4. İç nokta sayısı=2, delta sayısı =0 olursa parmak izi sarmal yapıdadır.

İç nokta-delta değerlendirilmesi:

a) İç noktadan geçen ve dizilimle aynı yönde uzanan bir eksen çizilir N_1 olarak tanımlanır.
b) İç nokta ve delta ikinci bir eksenle birleştirilir (N_2). İki eksen arasındaki açı belirlenir α olarak tanımlanır.
c) Yerel sırt yönü ile N_2 arasındaki açı hesaplanır β olarak tanımlanır.
d) N_2 ekseninden geçen kapalı eğri sayısı γ olarak tanımlanır.
e) Eğer (α>10°) ya da (β<15°) ve (γ=0) ise parmak izi çadırlı kemer yapıdadır.
f) Eğer delta noktası N_1 ekseninin sağ tarafında ise sağ döngü parmak izidir.
g) Delta noktası N_1 ekseninin sol tarafında ise sol döngü parmak izidir

Yukarıdaki algoritmaya göre sınıflandırma işlemi yapılır, bu yöntem herhangi bir eğitim aşaması içermez.

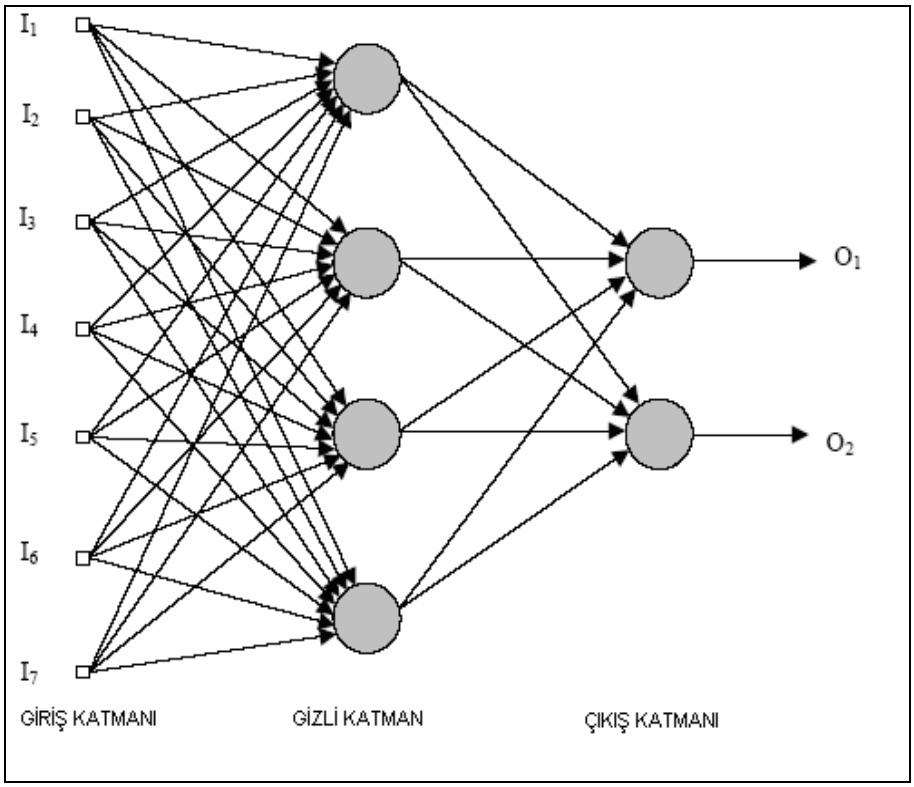
Tekillik tabanlı sistemler üzerine 1984 yılında Kawagoe ve Tojo 94 adet parmak izi ile %91,5 performans elde etmişlerdir [8]. Karu ve Jain 1996 yılında %85 [9], Cang ise, 89 parmak izi içeren veritabanıyla, 1997 yılında %96,5 sınıflandırma başarısı elde etmiştir [10]. Hang ve Jain aynı yöntemle 1999 yılında %87,5 başarıya ulaşmıştır [11].



Şekil 3.10 Tekil Noktalarını sayı ve konumlarına göre sınıflandırma [2]

3.2.3 Yapay sinir ağı sınıflandırıcılar

Yapay sinir ağları ve algoritmaları kullanılarak sınıflandırma yapılır. Danışmalı ve danışmansız öğrenme algoritmalarının her ikisiyle de yapılan çalışmalar mevcuttur. (MLP) Çok Katmanlı Perseptron ve (BP) Geri Yayılım yöntemleriyle danışmanlı öğrenme algoritmaları sınıflandırılmada kullanıldığı gibi, Öz düzenlemeli haritalama yöntemiyle danışmansız öğrenme algoritması da kullanılmıştır. Şekil 3.9' da iki katmanlı bir sinir ağı yapısı gösterilmektedir.



Şekil 3.11 Yapay sinir ağı modeli [13]

Danışmanlı öğrenme yönteminde Henry sınıfları kullanılarak ağ eğitilir. Eğitim aşamasından sonra veritabanındaki diğer parmak izleri eğitilmiş ağ ile sınıflandırılır.

Danışmansız öğrenme sınıflarını ağ yapısıyla belirler. Eğitim aşamasında ağırlık değerleri belirlenir. Belirlenen bu ağırlık değerlerine göre veritabanındaki diğer parmak izleri Euclid uzaklığı kriterine göre sınıflandırılır.

Parmak izi sınıflandırma çalışmalarında yapay sinir ağları ile yapılmış çalışma sayısı diğer yöntemlere göre fazladır.

Anil K. Jain ve Salil Prabrakar iki aşamalı sınıflandırıcı kullanarak, ikinci aşamayı MLP ile gerçekleştirerek, 1999 yılında dört sınıf için %97,8 beş sınıf için %96 başarı sağlamışlardır. Bu çalışmada veritabanının %32,5' i reddedilerek sınıflandırmaya tabi tutulmamıştır [17]. Dubravko Krasnjak ve Vuk Krivec MLP yöntem ile sınıflandırma

işlemini gerçekleştirmişler ve çalışmalarında MLP yapısının sonuçlara etkisini incelemişlerdir [27]. Wilson paralel yapıda MLP kullanarak %99,7 başarı elde etmiştir [28].

H.Paurghassem ve H. Ghassemian olasılık tabanlı yapay sinir ağı kullanarak altı sınıftan oluşan sınıflandırma çalışmalarında %95,5 başarı sağlamışlardır [29].

Masayashi Kamujo dört katmanlı sinir ağı ile 500 parmak izini sınıflandırmıştır. İki aşamalı öğrenme uygulanan çalışmada %99 başarı sağlanmıştır [30].

3.2.4 Parmak izi sınıflandırma çalışmaları ve performansları

Günümüze değin parmak izi sınıflandırma işlemi birçok farklı yaklaşımla gerçekleştirilmiştir. Sözdizimsel sınıflandırıcılar, tekillik sınıflandırıcılar ve yapay sinir ağı sınıflandırıcıları başlıca ana yöntemledir. Bunun yanı sıra fraktal analiz, AM-FM modelleri gibi farklı yapıların geliştirildiği çalışmalarda mevcuttur [38,39].

Genel olarak bu yöntemler Çizelge 3.1 ile özetlenebilir [21].

Yapılan çalışmaların öznitelik bulma yöntemleri, sınıflandırıcı metot ve performansları ise Çizelge 3.2 ile özetlenebilir [9].

Oluşturulan bu sistemlerin başarısı kullanılan özellik vektörü, sınıflandırma yöntemi ve kullanılan veritabanlarına göre şekillenmiştir. Öznitelik bulma için genellikle çalışmalar yön dizilim yöntemiyle yapılmış ve yüksek performanslar elde edilmiştir. Sınıflandırma yöntemlerinde ise en yüksek performans birleştirilmiş yöntemle elde edilmiş. Öz düzenlemeli haritalama ise ikinci sırada yer almıştır.

ÇALIŞMAYI YAPAN	ÇALIŞMA KARAKTERİSTİĞİ
Kamijo -1993 [30]	4 katmanlı sinir ağı yöntemi
Candela -1995 [13]	Olasılık yapay sinir ağı (PNN)
Karu and Jain -1996 [12]	Kural tabanlı (yapısal) yöntem
Maio and Maltoni -1996 [19]	Yön dizilim görüntüsünün bölümlenmesi ile sınıflandırma
Halıcı and Ongun -1996 [1]	Öz düzenlemeli haritalama
Q. Yuan -1998 [25]	Genetik algoritma, geri besleme ve olasılık sinir ağı kullanılarak sınıflandırma
Jain -1999 [17]	K-yakınlık komşuluğu, Gabor filtrelerinin yapay sinir ağında gerçeklenmesi ile sınıflandırma
Cappeli -1999 [14]	Çok uzaylı PCA
Sun -2000 [32]	Fraktal analiz
Pattichis – 2001 [31]	Olasılık yapay sinir ağı ve parmak izi AM – FM özelliklerinin birleştirilmesiyle yapılan sınıflandırma
Bernardi -2001 [22]	Kohonen topolojik yöntemi
Senior -1997 [23]	Hidden Markov + PCASYS + Karar ağacı yöntemi
Jain and Minut -2002 [21]	Model tabanlı hiyerarşik öz yaklaştırma yöntemi
Mohamed and Nyongesa -2002 [33]	Bulanık mantık yaklaşımı ile sınıflandırma
Yao -2003 [21]	Tekrarlanan yapay sinir ağ yöntemi

Çizelge 3.1 Sınıflandırma çalışmaları

ÇALIŞMAYI YAPAN	SINIF SAYISI	ÖZELLİK BELİRLEME	METHOD	BAŞARI %
Kawagoe and Tojo -1984	7	Tekillik noktası	Tekillik Tabanlı	91.5
Blue -1994	5	Yapısal Tabanlı	Sinir ağı	92.8
Wilson -1994	5	Yapısal Tabanlı	Sinir ağı	90.2
Candela -1995	6	Yapısal Tabanlı	Sinir ağı	92.2
Pal and Mitra -1996	5	Yapısal Tabanlı	Sinir ağı	82
Fitz and Gren	3	Forier Transform	En yakın Komşuluk	85
Karu and Jain-1996	5	Tekillik noktası	Tekillik Tabanlı	85
Halıcı and Ongun -1996	Değişken	Yapısal Tabanlı	Sinir ağı	96.8
Senior -1997	4	Sırt çizgileri	Hidden Markov	90
Chong -1997	5	Sırt çizgileri	Tekillik Tabanlı	96.5
Hong and Jain -1999	5	Sırt çizgileri ve Tekillik noktası	Tekillik Tabanlı	87.5
Proposed -1999	5	Gabor filtreleri	Birleşik yöntem*	90
Capelli -2000	5	Yapısal Tabanlı	Birleşik yöntem*	99

Çizelge 3.2 Yapılan çalışmaların metot ve performansları

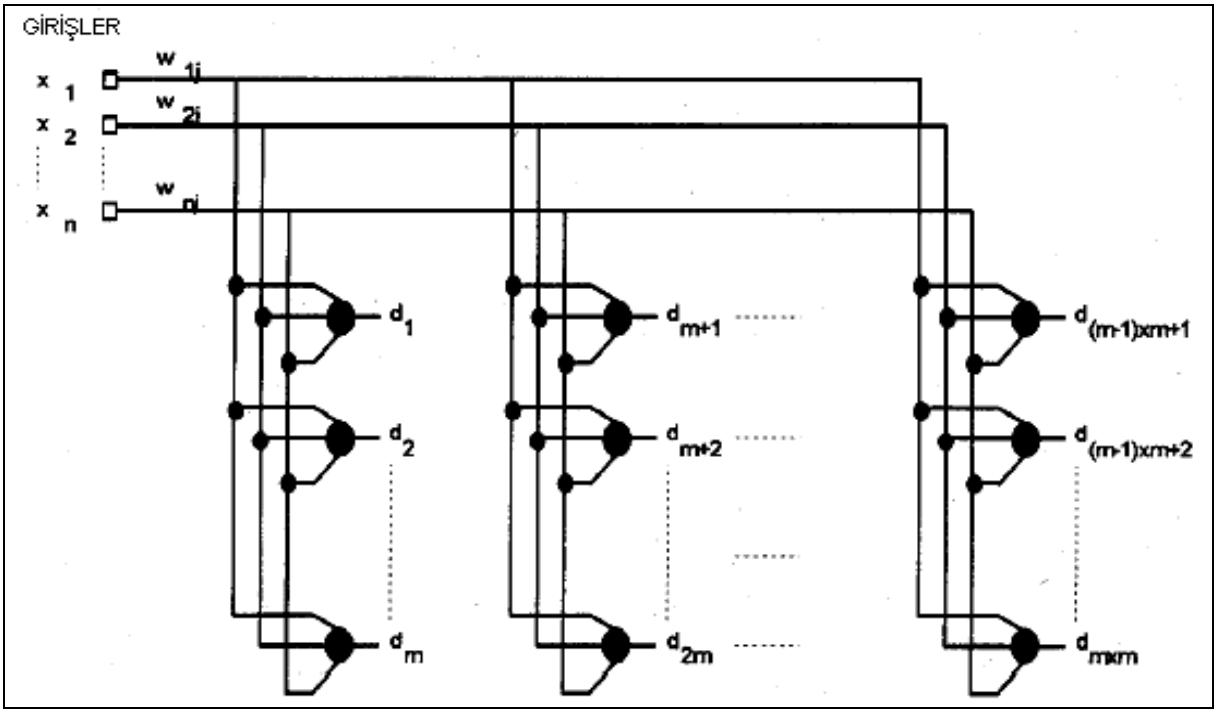
*Birleşik yöntem uygulamasında algoritma sınıflandırıcı yöntemlerinden iki veya daha fazlasını kullanarak sınıflandırmayı gerçekleştirir.

3.3 Öz Düzenlemeli Haritalama İle Sınıflandırma

3.3.1 Öz düzenlemeli haritalama algoritması

İnsan beyninin kullandığı yöntemlerden biridir. Beyinde her alıcını işlendiği bölge ayrı bir alanda yapılandırılarak tüm bilgi düzgün bir biçimde saklanır ve aynı bilgiyi işleyen nöronlar birbirine yakındır. Yarışmacı öğrenmeye dayanan bu algoritmada belli girişler belli nöronlara eşleştirilir. Algoritmanın simülasyonu yapıldığında her giriş için kazanan bir nöron ya da düğüm olduğu görülür.

Öz düzenlemeli haritalama algoritması n-boyutlu bir giriş vektörünü daha az boyutlu bir çıkış düzlemine atayan bir sinir ağı yapısıdır. N girişli ve mxm çıkışlı öz düzenlemeli haritalama modeli Şekil3.12' de gösterilmiştir.



Şekil 3.12 Öz düzenlemeli haritalama ağ yapısı [1]

Tüm çıkış düğümleri (j) tüm giriş düğümleriyle bağlıdır (i) ve bu düğümlerin ağırlıkları w_{ij} ile ifade edilir. Öz düzenlemeli haritalama için iki aşama söz konusudur bunlar eğitim aşaması ve sınıflandırma aşamasıdır.

Öz düzenlemeli haritalama algoritması eğitim basamakları aşağıdaki gibidir:

1. w_{ij} ağırlıklarına başlangıç değeri olarak rastgele küçük sayılar atanır.
2. Örnek kümesinden bir x vektörü seçilerek girişe uygulanır.
3. Kazanan düğüm d_{win}

$$d_{win} = \min_{j} \left\{ \left\| x - w_j \right\| \right\} \qquad (3.13)$$

İfadesine göre belirlenir.

||.|| euclid uzaklığını,

W_j giriş vektörlerini çıkışa bağlayan ağırlık değerlerini sembolize eder.

4. Ağırlıkların yeniden güncellenmesi için

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left[x_i(t) - w_{ij}(t)\right]N(j,i) \qquad (3.14)$$

Eşitliği kullanılır.

W_{ij}, w_j ağırlık vektörünün i'inci bileşenini
$\eta(t)$ öğrenme katsayısı
N(j,t) komşuluk fonksiyonunu ifade eder.

5. Ağırlık değişimlerindeki fark en aza indirilene kadar 2. ve 4. basamaklar arasında döngü devam eder.

$\eta(t)$ öğrenme katsayısı başlangıçta büyük bir değerdir ancak zamanla öğrenme gerçekleştikçe bu değer küçülür.

N(j,t) komşuluk fonksiyonunu kazanan birimi ortalayan bir filtre gibidir ve zamanla yarıçapı azalır. Komşuluk fonksiyonu j çıkış nöronunun eğitim esnasında derecesini belirler. Bu fonksiyon öyle seçilir ki j çıkış nöronu kazanan nörona uzaklaştıkça ağırlık değişim oranı azalır. Bu uzaklık ağın çıkış katmanındaki topolojiye göre belirlenir. Komşuluk fonksiyonu "gaussian", "Mexican hat" ya da "rectangular" olarak seçilebilir.

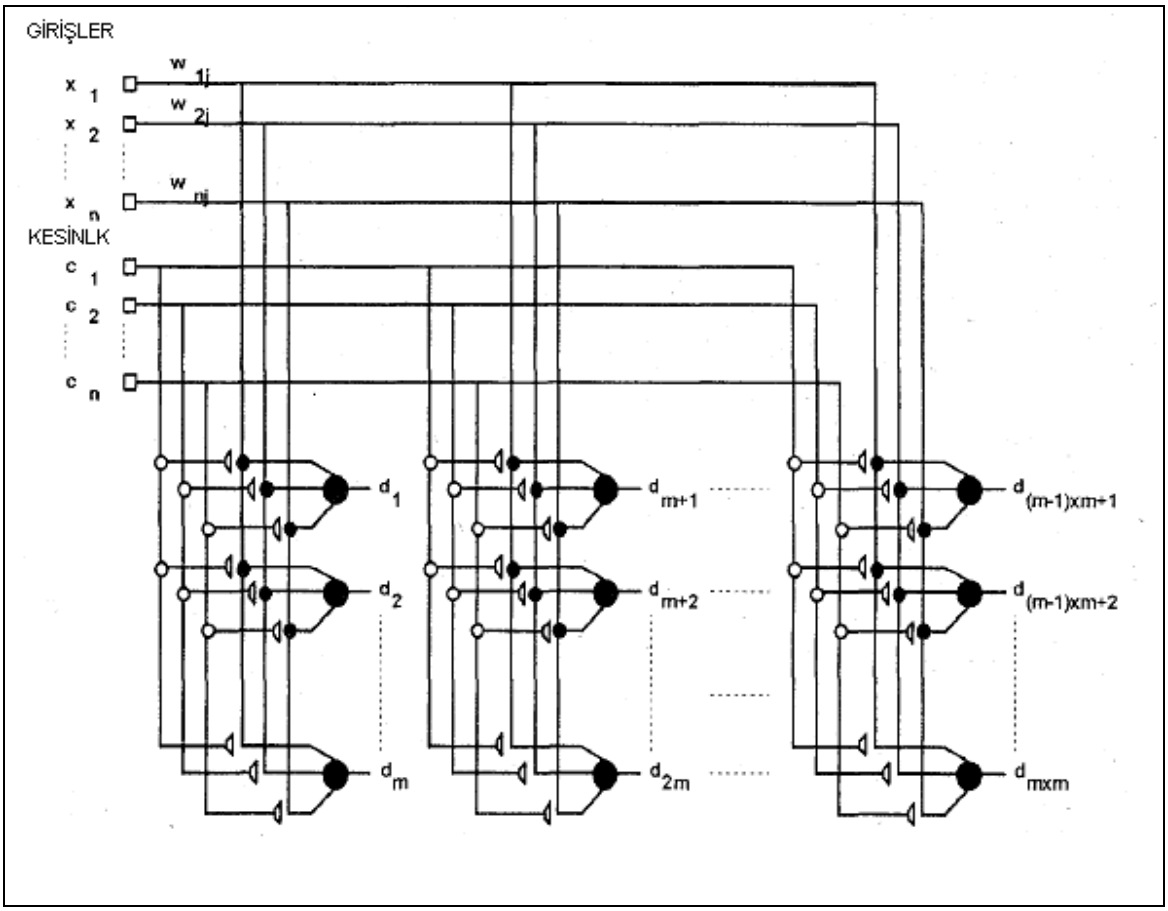
Eğitim aşaması başarıyla tamamlandıktan sonra sınıflandırma kısmına geçilir. Ağ giriş kümesiyle ağırlık kümesi arasındaki minimum Euclid uzaklığına sahip çıkış düğümünü bulmaya çalışır. Bu düğüme kazanan düğüm denir. Kazanan düğüme komşu olan ağırlıkların katsayıları arttırılarak ağırlıklar o anki giriş vektörüne yaklaştırılır. Bu durum tüm giriş vektörlerine ağırlıklar sabitlenene kadar uygulanır. Komşuluk fonksiyonu, öğrenme katsayısı, döngünün durdurulacağı an probleme göre değişiklik gösterir.

Eğitim tamamlandıktan sonra vektörün sınıflandırılması için eşitlik 3.13 kriter olarak kullanılır. Kazanan çıkış düğümü girişe uygulanan vektörün sınıfını belirler.

3.3.2 Öz Düzenlemeli haritalama ile sınıflandırma çalışmaları

Öz düzenlemeli haritalama kullanılarak yapılan sınıflandırma çalışmalarından birincisinde [1] sınıflandırma işleminde başarıyı artırmak için kesinlik değeri parametre olarak kullanılmıştır. Ağ eğitilirken kesinlik değerleri de göz önünde bulundurularak çıkış için, giriş düğümleri ve o anki giriş vektörü arasındaki, en kısa mesafeyi seçer. Kesinlik değerleri yön dizilim vektörünün oluşturulması sırasında bloklara ait şiddet

değerlerinden oluşur. Giriş değerlerine uygulanan kesinlik değerleri Şekil3.13' de gösterilmiştir.



Şelkil 3.13 Geliştirilmiş öz düzenlemeli haritalama ağ yapısı [1]

Geliştirilmiş öz düzenlemeli haritalama eğitim algoritması aşağıdaki gibidir.

1. w_{ij} ağırlıklarına başlangıç değeri olarak rastgele küçük sayılar atanır.
2. Örnek kümesinden bir x vektörü seçilir. Ve bu vektöre kesinlik değeri de eklenerek x^c vektörü elde edilir.

$$x_i^c = c_i x_i + (1 - c_i) x_i^{avg} \qquad (3.15)$$

x_i^{avg}, x_i 'nin örnek kümesindeki ortalama değerini gösterir

3. Kazanan düğüm d_{win}

$$d_{win} = \min_{j}\left\{\left\|x^c - w_j\right\|\right\} \qquad (3.16)$$

İfadesine göre belirlenir.

||.|| euclid uzaklığını,

W_j giriş vektörlerini çıkışa bağlayan ağırlık değerlerini sembolize eder.

4. Ağırlıkların yeniden güncellenmesi için

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left[x_i^c(t) - w_{ij}(t)\right]N(j,i).c_i \qquad (3.17)$$

eşitliği kullanılır.

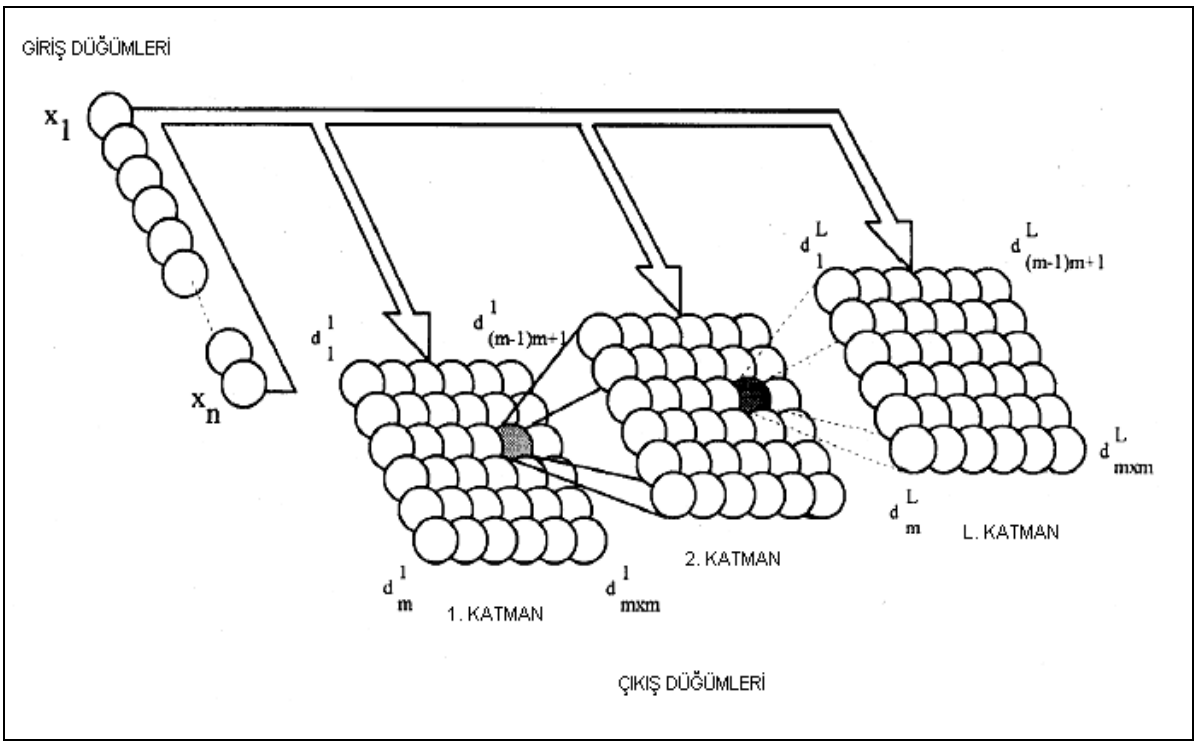
W_{ij}, w_j ağırlık vektörünün i'inci bileşenini

$\eta(t)$ öğrenme katsayısı

N(j,t) komşuluk fonksiyonunu ifade eder.

5. Ağırlık değişimlerindeki fark en aza indirilene kadar 2. ve 4. basamaklar arasında döngü devam eder.

Öz düzenlemeli haritalama ağ yapısında bazı sınıfları göstermek diğerlerine göre daha çok nöron gerektirir**.** Ağ yapısını çoğaltarak sınıfları arttırmak bir çözüm olabilir ancak bu defada birçok sınıf çok az veriye sahip olurken bazı sınıflar çok kalabalık bir veri taşıyacağından verim azalır. Ağ yapısını büyütmek yerine bazı sınıflara katman eklemek verimi arttırır. Algoritmaya katman eklendiğinde yeni katman bir alt katmanın kazanan sınıf örnekleriyle eğitilir. Gerekli görüldüğünde yeni katmanlarla bu işlem devam ettirilebilir böylece çok kalabalık sınıflar bölümlere ayrılmış olur. Çok katmanlı ağ yapısı Şekil 3.14' de gösterilmiştir.



Şekil 3.14 Çok katmanlı ağ yapısı [1]

Her parmak izi katmandaki ilgili nöronun gösterdiği hiyerarşik sınıf numarasına göre eşleştirilir. Bu parmak izinin farklı açıdan bir görüntüsü sisteme girildiğinde arama sonucu son katmandaki kazanan nöron ile ilişkilendirilir. Eğer bulunamazsa ikinci kazanan nöron ile aynı işlem yapılır. En üst katmandaki kazanan nöronlardan hiçbirinde bulunamazsa bir alt katmanın kazanan ikinci nöronunda arama yapılır. Sistem performansı Öz düzenlemeli haritalama için %92,8, geliştirilmiş öz düzenlemeli haritalama için %96,8 olarak bulunmuştur.

2000 yılında bu çalışma üzerinde yapılan deneyler sonucu [34] yön dizilim grafiğindeki kalite farkının ve merkez noktası bulma performansının bu konuda 1995 yılında yapılan [1] çalışmaya etkisi araştırılmıştır. Bu araştırma sonucu bu iki faktörün sistem üzerinde etkili olduğunu göstermiştir. Yön dizilim vektörü gradyan yöntemiyle bulunursa, merkez noktası sistem tarafından bulunduğunda sınıflandırma performansı

%79,8 olarak bulunurken eğer merkez uzman tarafından sisteme doğru olarak belirtilirse başarı %85,7 olarak bulunur. Yön vektörlerinde temel bileşen analizi kullanılırsa, merkezi sistem otomatik bulduğunda performans %83, uzman tarafından doğru olarak belirlendiğinde ise %86,9 başarı sağlanır. Bu çalışmada ayrıca ağ yapısının boyutu arttığında performansın arttığı belirlenmiştir.

Bu konuda ikinci çalışma 2000 yılında yapılmıştır [25] ve ilk çalışmanın geliştirilmesi amaçlanmıştır. Kullanılan sistem ilk sistemin aynısıdır. Farklı olarak bu çalışmada, öznitelik bulma aşamasından önce resimlerdeki bozulmanın sisteme etkisinin azaltılması amacıyla, resimlere ters filtreleme tekniği uygulanmıştır. Bu tekniğin uygulanmasıyla uygulanan giriş resimlerinden bir kısmı reddedilerek sınıflandırılmamıştır. Sistem performansı %91 (tanımlanmayan resim oranı %0), %98 (tanımlanmayan resim oranı %8,1) olarak belirlenmiştir.

2001 yılında yapılan üçüncü çalışmada [22] yine yön vektörleriyle özellik belirlenmiş ve iki boyutlu öz düzenlemeli haritalama yapısı kullanılarak %88 performans elde edilmiştir. Bu çalışmanın farkı ise öz düzenlemeli haritalama algoritması ile forgy algoritmasının birleştirilmesidir [22]. Ancak elde edilen performans diğer çalışmalara oranla düşüktür.

4.SİSTEMİN TASARIMI

Parmak izi sınıflandırma çalışmaları göz önüne alınarak sistemin tasarımı için gerekli iki aşamada kullanılacak yöntemler belirlenmelidir. Öncelikle öznitelik bulma vektörü için yaklaşımlardan biri seçilmiş ve daha sonra sınıflandırıcı yöntemi belirlenmiştir.

Sistem öznitelik bulma yöntemi olarak yön dizilim görüntüsü sistemin tasarımında kullanılmıştır. Yön dizilim görüntü öznitelik bulma yaklaşımlarının tümünde kullanılır. Filtre tabanlı yaklaşımda merkez noktası yön dizilim vektörleriyle belirlenir. Yapısal tabanlı yaklaşımda görüntünün modeli yön dizilim vektörlerindeki değişime göre oluşturulur. İstatistiksel tabanlı yaklaşımda yine bu vektörlerden elde edilen bilgi istatistiksel veri olması açısından kullanılabilir.

Bu çalışmada yön dizilim görüntüsü sonuçları istatistiksel veri bazında ele alınıp öznitelik bulma vektörü olarak kullanılmıştır. İşlem basamakları azaltılarak sadece yön dizilim görüntüsü kullanılarak sınıflandırma performansının incelenmesi bu çalışmanın hedeflerinden birisidir.

Yön dizilim vektörü daha önce yapılan çalışmalarda 16x16 boyutlarında bloklanmış alan kullanılarak 256x1 boyutlarında oluşturulmuştur [1,25]. Bu konudaki yapılan başka bir çalışmada ise 32x32 bloklanmış alan kullanılmıştır [22]. Bu çalışmada 100x1 vektör 10x10 boyutlarında bloklanmış alandan elde edilmiştir. Boyut azaltılmasının sistemin performansına olan etkilerinin incelenmesi bu çalışmada amaçlanan diğer etkendir.

Sınıflandırıcı yöntemlerinden yapay sinir ağları kullanılmıştır. Yapay sinir ağı algoritmalarından öz düzenlemeli haritalama algoritması kullanılmaktadır. Bu yöntemin kullanılmasıyla sınıflandırma nöron davranışlarıyla gerçekleştirilmiştir. Ayrıca öz düzenlemeli haritalama parmak izi veritabanının Henry sınıfları dışında da bölümlenmesine olanak tanır. Böylece daha fazla sınıf oluşturulabilir.

Sistemin aşamaları

a) Veri tabanı

b) Yön dizilim görüntüsü

c) Merkez noktası belirleme

d) Sınıflandırma Giriş Vektörünün oluşturulması

e) Sınıflandırma

ana başlıklarıyla incelenecektir.

4.1 Veri Tabanı

Sisteme uygulanacak parmak izi veri tabanı “Fingerpirint Verification Competition” adıyla bilinen, parmak izi tanıma sistemlerinin performansını ölçen kuruluştan alınmıştır.

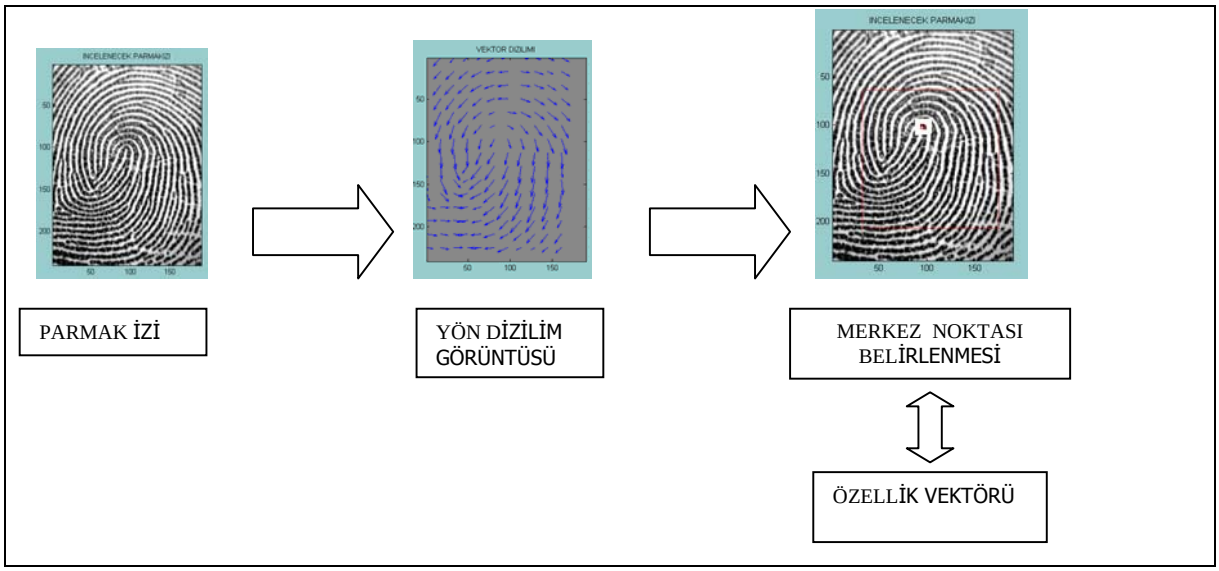
Veri tabanı (VT) iki farkı sensörle alınmış 10 farklı parmak izinin, farklı açılarda alınan 8’er görüntüsünden oluşur. Toplam 10 farklı parmak izinden alınmış 160 (80 VT1, 80 VT2) görüntü ile çalışılmıştır.

Sistem farklı boyutlardaki resimlere algoritmaları uygulayabilecek şekilde tasarlanmıştır. Birinci sensör parmak izlerini 300x300 piksel boyutunda tararken, ikinci sensör 256x364 piksel boyutunda taramıştır.

4.2 Yön Dizilim Görüntüsü

Parmak izleri bloklara ayrılıp her blok için yön değerleri ve açıları belirlenirse oluşan model ile farklı açılarda alınan görüntülerin modelleri karşılaştırılabilir. Parmak izinin oluşan modelle sınıflandırılması mümkündür.

Veritabanında alınan parmak izi önce filtrelenerek piksel koyuluğu azaltılır böylelikle işleme süreci kısaltılmış olur. Filtreleme işleminde iki boyutlu gürültü giderici bir filtre olan Wiener alçak geçiren filtresi kullanılır.



Şekil 4.1 Oluşturulan sistemin şeması

Wiener filtresi gürültüyü azaltma ve ters filtreleme arasındaki seçimde uygun sonuçlar verir [35]. Bulanıklığı ve gürültü faktörünü hızlı bir biçimde giderebilir. Ortalama karesel hata hesaplaması açısından da iyi bir filtredir. Görüntüyü işlediği esnada ortalama karesel hata en düşük değerde olur. Wiener filtresiyle işlenmiş resim orijinal görüntünün doğrusal tahminidir [36]. Wiener filtresinin Fourier sistemde ifadesi aşağıdaki formülle gösterilebilir.

$$W(f_1,f_2)=\frac{H^*(f_1,f_2)S_{xx}(f_1,f_2)}{\left|H(f_1,f_2)\right|^2 S_{xx}(f_1,f_2)+S_{\eta\eta}(f_1,f_2)} \qquad (4.1)$$

$S_{xx}(f_1,f_2)$, $S_{\eta\eta}(f_1,f_2)$ Orijinal resmin ve toplam gürültünün güç bandıdır, $H(f_1,f_2)$ Resmi bulanıklaştıran filtredir.

Şekil 4.2 Görüntünün Wiener filtresiyle işlenmesi

Şekil4.2' deki resim alçak geçiren filtresi ile görüntü kalitesi düşürülmüş ardından Wiener filtresiyle işlenerek orijinal resme yaklaştırılmaya çalışılmıştır.

Filtrelenen resim "yinelenen en küçük ortalama karesel yön kestirimi" adı verilen yöntemle yön vektörlerine ayrıştırılır.

Kullanılan algoritmanın temel basamakları;

1. Resmin çözünürlüğüne göre belirlenen bir değerde resim bloklara ayrılır. Bloklama matris boyutları resim üzerinde denenerek belirlenebilir. Resmin bloklanmasında fazla blok sayısı işlemlerin artmasına yol açarken, gerekenden az blok sayısı da yönlerin tam olarak modellenmesini engeller.

2.Resim matrisinde (i,j) olarak tanımlanan her pikselin yatay $\partial_x(i,j)$ ve düşey $\partial_y(i,j)$ gradyan değerleri hesaplanır. Gradyan değerleri şekil 4.3' de gösterilmektedir.

Şekil 4.3 Yatay gradyan Dikey gradyan

3. Merkezi piksel (i,j) ile tanımlanan bloğun yön vektörünü elde etmek için

$$V_x(i,j) = \sum_{u=i-\frac{w}{2}}^{i+\frac{w}{2}} \sum_{v=j-\frac{w}{2}}^{j+\frac{w}{2}} 2\partial_x(u,v)\partial_y(u,v) \tag{4.2}$$

$$V_y(i,j) = \sum_{u=i-\frac{w}{2}}^{i+\frac{w}{2}} \sum_{v=j-\frac{w}{2}}^{j+\frac{w}{2}} (\partial_x{}^2(u,v) - \partial_y{}^2(u,v)) \tag{4.3}$$

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{V_y(i,j)}{V_x(i,j)}\right) \tag{4.4}$$

eşitlikleri kullanılır [31].

θ (i,j) burada en küçük karesel hesaplayıcı olarak kullanılır.

Yukarıdaki işlemlerle resmin yön vektörleri oluşturulduktan sonra resmin işlenmesinde kullanılacak ilgili alanın belirlenmesi için yerel kesinlik değerleri belirlenir.

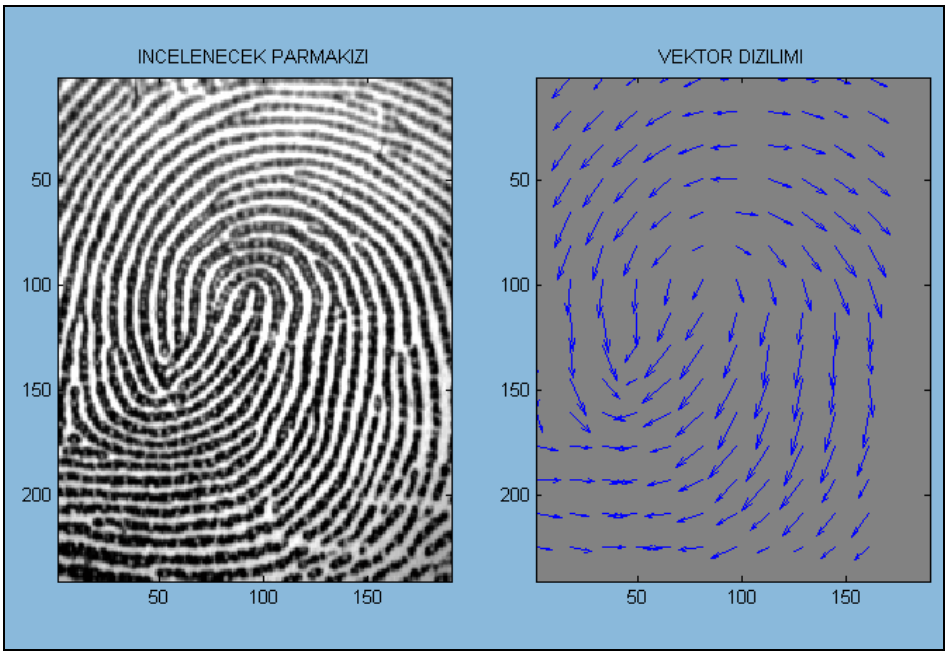
Bloğun kesinlik değeri

$$\varepsilon(i,j) = \sqrt{\frac{1}{wxw} \cdot \frac{(V_x(i,j)^2 + V_y(i,j)^2)}{V_e(i,j)}} \tag{4.5}$$

$$V_e(i,j) = \sum_{u=i-\frac{w}{2}}^{i+\frac{w}{2}} \sum_{v=j-\frac{w}{2}}^{j+\frac{w}{2}} (\partial_x{}^2(u,v) - \partial_y{}^2(u,v)) \qquad (4.6)$$

Eşitlik (4.5) ve (4.6) ile hesaplanır. Bloktaki piksel değerleri belirlenmiş bir eşik değerin altındaysa arka fon piksel olarak algılanır. Kesinlik değeri bulmak için $\varepsilon(i,j)$ eşitliğinin kullanılmasının iki sebebi vardır. Birincisi tahmin edilen yön vektörlerinin çarpımıyla bulunur dolayısıyla bulunması daha kolaydır, ikincisi ilgili işlem alanının bulunmasında iyi bir performans sergiler.

Resmin yön dizilimi için gerekli açı ve şiddet değerleri yukarıdaki algoritma ile bulunduktan sonra, bu değerlerle yön görüntüsü çizdirilebilir. Şekil 4.4' de yön dizilim görüntüsü gösterilmektedir.



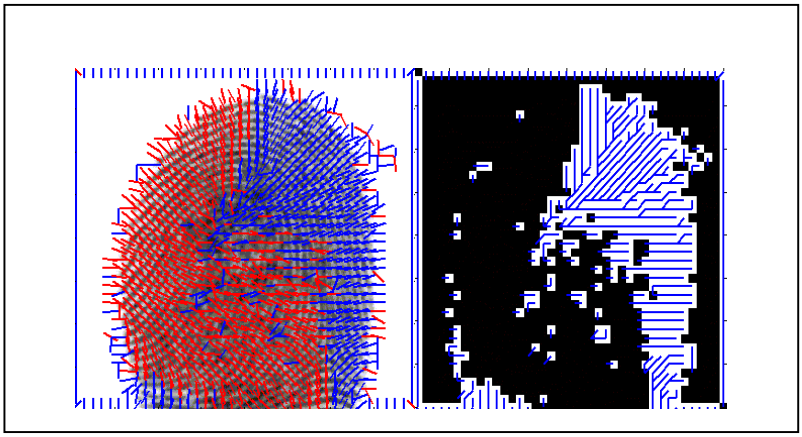
Şekil 4.4 Yön dizilim görüntüsü

4.3 Merkez Noktası Belirleme

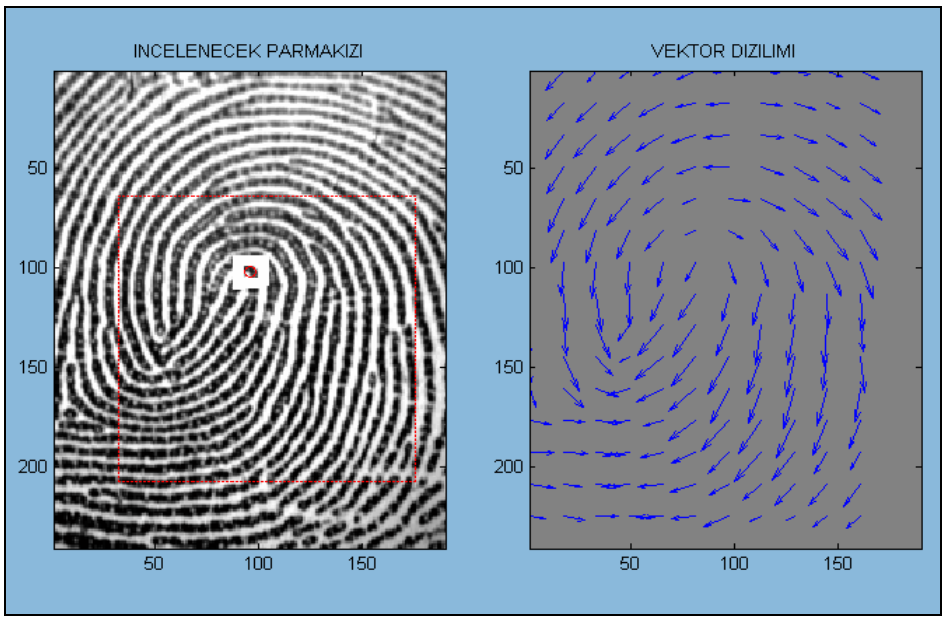
Parmak izi üzerinde bir merkez noktası parmak izlerini karşılaştırırken referans noktası olarak kullanılır. Parmak izleri alınırken her zaman aynı açıda alıp yapılması mümkün olmadığından, büyüklük açı değişimi gibi faktörlerden etkilenilmemesi için resim kendi içinde bir çeşit kodlama ile ifade edilmelidir. Merkez noktası belirlemede bu kodlamayı sağlarken başlangıç noktasına karşılık gelir [1].

Merkez noktası, iç içe geçmiş sırt çizgilerinin en içteki eğrinin orta noktasıdır ve bu nokta blokların eğimleri karşılaştırılarak bulunur. Merkez noktası belirlendikten sonra bloklanmış görüntü ile kesinlik değerleri, merkez noktası referans alınarak yeniden düzenlenir. Düzenlenen bu vektörle sınıflandırma yapılır.

Merkez noktasını bulmak için resim bloklarındaki açılara bakılır. Açıların 0-π aralığında değer aldığı bloklar 1 ile diğerleri 0 ile işaretlenir [35]. Yön dizilim görüntüsü oluşturulurken bulunan açı değerleri yerel yöne dik olarak ifade edildiğinden 0-π aralığı 1 ile işaretlenmiştir. İşaretlenen matristeki 1 değerleri sayılarak matris aşağı yönde takip edilir. Eğim değerinin değiştiği noktaya yani 0 noktasına gelindiğinde merkez noktası bulunmuş olur. Şekil 4.5' de merkez noktasının bulunuşu, Şekil 4.6' de ise merkez noktası bulunmuş bir parmak izi gösterilmektedir.



Şekil 4.5 Merkez noktası bulma



Şekil 4.6 Merkez noktası gösterimi

Merkez noktası bulma işleminin veritabanına uygulanması sonucu oluşan performans değerleri Çizelge 4.1' de özetlenmiştir.

	TÜM RESİMLER	**İŞLENEBİLİR RESİMLER ***
VERİTABANI1 (VT1)	%70	%89.2
VERİTABANI2 (VT2)	%71.25	%95
TOPLAM	%70.62	%92.2

* Aşırı koyu aşırı silik ya da alınış açısı sebebiyle merkez noktası olmayan resimlerin veri tabanına dahil edilmediği durum

Çizelge 4.1 Merkez noktası bulma performansı

Merkez noktası bulunacak parmak izi görüntüsünün, çok koyu ya da silik olup sırt ve yarıkların belli olmaması durumunda, bunun yanı sıra parmak izinin alınma açısından

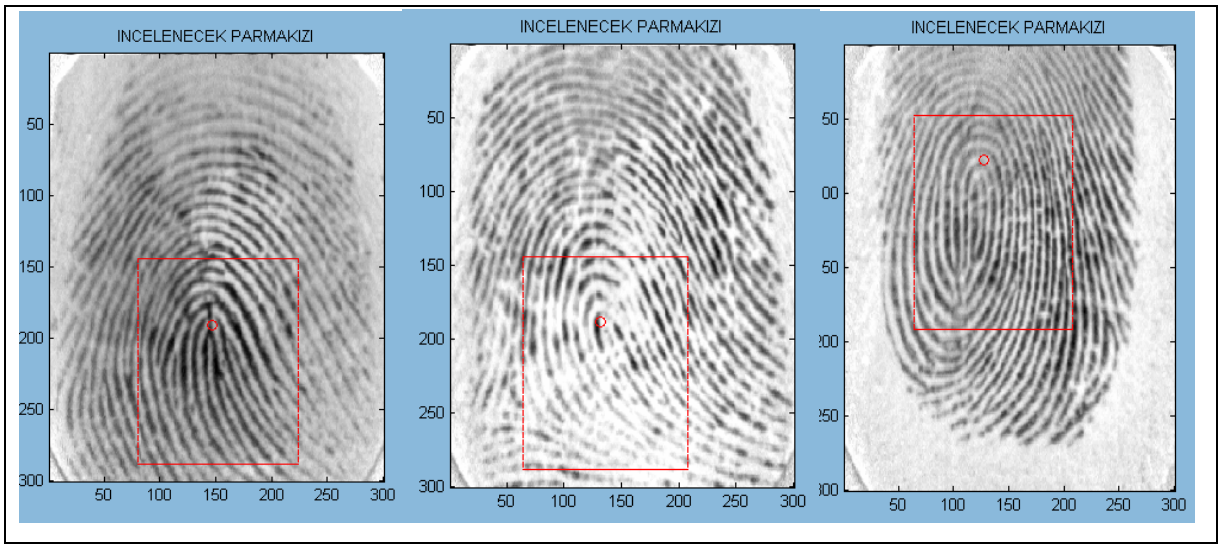
kaynaklanan merkezin olmaması ya da tam belirgin olmaması gibi durumlarda, merkez noktası belirlenemez. Örnekler Şekil 4.7' da gösterilmektedir.

Şekil 4.7Merkez noktasının belirlenemediği durumlar

Uygulanan yöntemin performansı veritabanı tipine (sensör farklılığına) göre değişim göstermiştir. Sisteme uygulanacak görüntü kalitesi ve parmak izi alım açısı merkez noktası bulma performansını etkiler.

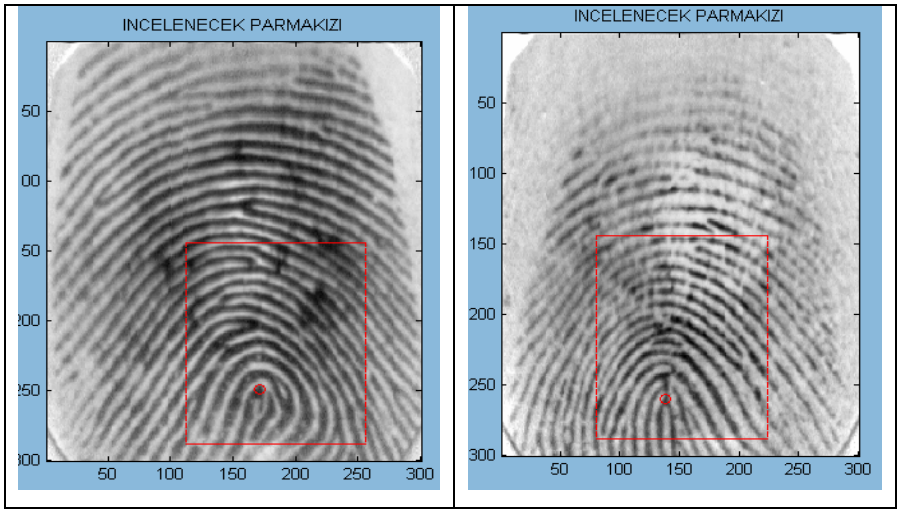
4.4 Sınıflandırma Vektörlerinin Oluşturulması

Sınıflandırma için giriş vektörleri, merkez noktası etrafındaki 160x160 pikselden oluşan bir alanın açı değerlerinin alınmasıyla oluşur. Bu değerler 10x10 boyutunda bir matris ile ifade edilir. 10x10 matrisi birinci sütundan son sütuna sırayla taranarak 100x1 matris oluşturulur. Bu matris öz düzenlemeli haritalama için sınıflandırma giriş vektörü olarak kullanılır. Aşağıdaki şekilde vektör oluşumu için kullanılacak bölgeler görülmektedir. Merkez noktasının altında kalan bölge sınıf karakteristik bilgisini taşıdığından seçilen alanın büyük bir kısmı Şekil4.8' de görüldüğü gibi merkez noktasının altındadır.



Şekil 4.8 Sınıflandırma vektörü bölgesi

Sınıflandırma vektörünün sabit eleman sayısına sahip olması gerekir. Bu nedenle vektörün eleman sayısı sabit olacak şekilde bölge belirlenir. Merkez noktasının kenarlara yakın olması durumunda ise Şekil 4.9' deki gibi seçilen bölge anlamlı bilgi taşımayabilir. Bu durum merkez noktası doğru belirlenmiş olsa bile sınıflandırma performansını olumsuz etkiler.



Şekil 4.9 Sınıflandırma performansını olumsuz etkileyen durumlar

4.5 Sınıflandırma

Bu bölümde Öz düzenlemeli haritalama yöntemi veri tabanına uygulanarak sınıflandırma başarısı ölçülecektir.

Öz düzenlemeli haritalamanın uygulamada iki aşamadan oluştuğu, işlem basamaklarında açıklanmıştı. Bu aşamalar eğitim ve simülasyon aşamalarıdır. Eğitim aşamasında sınıflandırılacak veriler algoritmaya uygulanır. Uygulanan bu verilerle algoritma sınıfları oluşturur. Tüm verilerin atandığı bir sınıf oluşur ancak farklı iki veri aynı sınıfta da olabilir. 10 farklı parmak izinden birer tanesi eğitim aşaması için seçilir ve eğitim kümesini oluşturur. Eğitim kümesi algoritmada tanımlanan yapıya uygulandıktan sonra sınıflar ve sınıflara ait olan veriler belli olur. Ardından simülasyon aşaması gelir. Daha önce eğitimde kullanılmayan farklı açılardan alınmış izler algoritmaya uygulanarak sınıflara atanır.

Veri tabanında daha önce belirtildiği gibi 10 farklı parmak izi kullanılmıştır. Herhangi bir parmak izine ait birinci veritabanında sekiz, ikinci veritabanında sekiz olmak üzere toplam on altı adet resimle simülasyonlar gerçekleştirilmiştir.

Simülasyonlarda kazanan ilk nöron performanslarına ve kazanan ikinci nöron performanslarına bakılmıştır. Sadece bir kazanan nöron olduğunda sistemin performansı oldukça düşüktür, sınıflandırma için kazanan ilk nöron ve ikinci nöron belirlenir aranan parmak izi karşılaştırma işlemi için, önce ilk sınıfta daha sonra belirlenen ikinci sınıfta aranır.

Sınıflandırma işlemi kişisel bilgisayarda ve FPGA' da gerçekleştirilmiştir. Kişisel bilgisayarda yazılım olarak MATLAB 7, donanım olarak da İntel Celeron 1.7GHz işlemci ve 512 MB ram kullanılmıştır. FPGA ile gerçekleştirilen sınıflandırma işleminde Leonardo Spectrum yazılımı ile yapılan simülasyonlarda farklı FPGA tipleri kullanılmıştır. FPGA uygulaması ise Xilinx Spartan2E XC2S200E-PQ208 kullanılarak gerçekleştirilmiştir.

4.5.1 MATLAB ile sınıflandırma

MATLAB 7 kullanılarak gerçekleştirilen simülasyonlarda başlangıçta nöron dağılımı rastgele topolojik yapıda seçilmiştir. Kazanan nöron belirleme kriteri olarak öklid uzaklığı, öğrenme katsayısı olarak da 0.8 sabiti seçilmiştir. Öğrenme işleminin ikinci aşamasında ise, daha küçük bir öğrenme katsayısıyla ağırlıklar son değerlerine yaklaştırılır. Bu deger 0.02 sabitidir. Komşuluk fonksiyonu için kazanan nöron ve komşu nöronlar arasındaki uzaklık sabit öklid uzaklığı olarak alınmıştır.

Simülasyonda 10 farklı parmak izinden bir tanesi eğitim vektöründe kullanılmıştır. Kullanılan bu parmak izine ait sınıflandırma vektörünün simülasyon sonucunda atandığı nöron bu parmak izinin sınıfını belirtir. Nöronlar arasındaki uzaklıklara bakılarak komşu nöronlar belirlenmiştir. Aynı sınıfa dahil nöronlar ve kazanan ikinci nöron, nöronlar arasındaki uzaklıklar göz önünde bulundurularak belirlenmiştir.

Eğitim aşamasındaki döngü sayısı az olursa ya da çok fazla olursa sınıflandırmada yanlışlıklar oluşabilmektedir. Ek 5 ‘de [3 3] yapıdaki bir SOM ağının eğitim aşamasında seçilen farklı döngü sayılarına göre nöron dağılımları gösterilmiştir.

Merkez noktasının bulunmasında veritabanındaki uygulanabilir resimler belirlenmişti. Uygulanabilir resimleri içeren veritabanı ile SOM algoritmasıyla sınıflandırılma sonuçları Çizelge 4.2’ de gösterilmiştir. Çizelge 4.2‘ de iki boyutlu SOM ağ yapısı ile simülasyonlar gösterilirken, Çizelge 4.3’ de tek boyutlu SOM ağ yapısı ile gerçekleştirilmiş simülasyonlar gösterilmektedir.

Simülasyonlar aynı sayıda nöron içeren iki boyutlu ve tek boyutlu SOM ağ yapıları ile gerçekleştirilmiştir. İki boyutlu yapılarda sistemin kararlı duruma geçmesi için daha fazla döngü sayısı gerekmektedir.

YAPI	SINIF SAYISI	DÖNGÜ SAYISI	%BAŞARI (1.S / 2.S)	İŞLEM SÜRESİ(s)*
3x3	7	500	% 47,71 / % 57,02	11,33
	7	1000	**% 72,72 / % 77,68**	22,90
	7	2000	% 52,02 / % 66,11	48,29
	7	4000	% 43,80 / % 59,50	88,37
4x4	8	500	% 47,79 / % 66,11	13,20
	8	1000	% 52,89 / % 63,63	25,97
	8	2000	% 44,14 / % 58,67	47,41
	8	4000	% 47,10 / % 64,67	91,78
5x5	8	500	% 23,96 / % 36,36	14,50
	8	1000	% 33,88 / %34,71	27,69
	8	2000	% 48,76 / %53,71	55,03
	8	4000	% 54,54 / %61,15	111,66

* İşlem süresi algoritmanın eğitim aşaması süresidir. MATLAB 7 de (cputime) tanımlı fonksiyonu kullanılarak bulunmuştur.

Çizelge 4.2 İki boyutlu SOM ağ yapısı simülasyonları

Çizelge 4.2 incelendiğinde, iki boyutlu 3x3 yapının performansının en yüksek olduğu görülür. Bu sistem için sınıflandırma işlemi veritabanının bölümlenerek numaralandırılması anlamına gelmektedir. Buna göre çizelgeyi değerlendirdiğimizde, 3x3 yapı veri tabanını toplam 7 sınıfa ayırmaktadır. Kazanan ilk nöron ele alındığında performans %72.72 bulunur. Bunun anlamı, veritabanı bölümlendirilmesi sonucu sisteme girişi yapılan bir parmak izinin, veritabanı arama işleminde ilk bölümde bulunma olasılığının %72.72 olmasıdır. Veritabanı kazanan ikinci nörona, üçüncü ve diğerlerine bakılarak taranır. Böylece SOM yapısı kullanılarak veri tabanı taraması verimli bir şekilde yapılabilir. Çizelgeye göre sistemin ikinci bölümde, yani ikinci nöronda aranan parmak izine ulaşma şansı %77.68'dir.

YAPI	SINIF SAYISI	DÖNGÜ SAYISI	%BAŞARI (1.S / 2.S)	İŞLEM SÜRESİ(s)*
9	5	100	%47,93 / %62,80	1,51
	5	200	% 48,76 / %66,94	2077
	5	400	% 52,89 / %71,90	4,93
	5	600	% 53,71 / %74,38	9,37
16	8	100	% 49,8 / %56,19	3,17
	8	200	% 47,94 / % 59,50	5,61
	6	400	% 45,45 / % 70,24	10,71
	5	600	% 50,41 / % 71,33	15,79
25	6	100	% 42,14 / %64,46	3,71
	5	200	% 72,72 / % 79,33	6,00
	4	400	**% 77,68 / % 83,47**	12,69
	4	600	% 73,55 / % 81,81	16,92

Çizelge 4.3 Tek boyutlu SOM ağ yapısı simülasyonları

Çizelge 4.3 incelendiğinde ise tek boyutlu ağ yapılarından 25 nöron kullanılarak oluşturulan yapının performansı, kazanan ilk nöronda %77.68, ikinci nöronda %83.47 bulunmuştur. İki çizelge karşılaştırıldığında tek boyutlu 25 nörondan oluşan sistemin en yüksek performansı sergilediği görülmektedir.

4.5.1.1 Simülasyon sonuçlarının değerlendirilmesi

Sistem 100 elemanlı giriş vektörlerinden oluşmaktadır. Giriş vektörleri parmak izinin merkez noktası etrafındaki çerçevenmiş alanın açı bilgilerini içermektedir. SOM algoritması ile sınıflandırılan bu vektörler, incelenen parmak izinin hangi sınıfa ait olduğunu göstermektedir. Sınıflandırma işlemi, bu uygulamada veritabanı

bölümlenmesinde kullanılır. SOM algoritması ile bu işlemi gerçekleştirmek, parmak izinin hangi sınıfa ait olduğunu bulmaya yardımcı olduğu gibi kazanan nöron kavramı sayesinde veritabanını verimli bir biçimde taramaya da olanak sağlar. Sınıflandırma sistemleri genellikle parmak izlerini sınıflandırır ve veritabanında aranacak parmak izini bu sınıflandırma sonucuna göre arar. Sistemin belirlediği sınıfta iz bulunamazsa bakılacak ikinci veritabanı bölümünü tanımlamak gerekebilir. Ancak bu uygulamada kullanılan SOM algoritması, sisteme müdahale edilmesine gerek duymadan kazanan nöronları sıralayarak veritabanının taranmasını verimli bir şekilde gerçekleştirebilir.

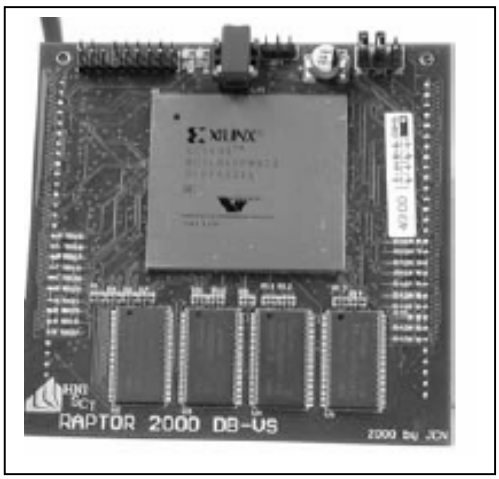
Sınıflandırma performansı kazanan ilk nöron ele alındığında çok yüksek değildir ancak ikinci, üçüncü nöronlarda ele alındığında performans artmaktadır. Bu uygulamada amaç veritabanı taramasının kısaltılması olduğundan bulunan performans değerleri kabul edilebilir ölçülerdedir.

4.5.2 Som algoritmasının VHDL ile gerçekleştirilmesi

Yapay sinir ağları oldukça işlem gerektiren bir yapı içerir bu nedenle paralel işlemciler ya da paralel bilgisayarlar kullanılarak zaman kısaltmaları üzerine çalışmalar gerçekleştirilmektedir.

Özellikle görüntü işleme uygulamaları için karmaşık yapılı sinir ağları gerekir ve bu da genellikle bilgisayarların paralel işlenmesiyle oluşturulur [39]. Ancak matematiksel ifadelerle oluşturulan algoritmalar bazen paralel yapılara uyum sağlayamadığı gibi, paralel yapılarda her zaman matematiksel olarak ifade edilemez [40]. Bu sebeple FPGA için yazılan kodlarla her zaman bir donanım oluşturmak mümkün olmayabilir. Sentezlenebilir bir mimari ile algoritma ifade edilerek FPGA içine gömülü yazılım geliştirmek mümkündür.

Paralel işlemcilerle tasarlanan sinir ağları tekrarlanan işlemlerin paralel yapılması nedeniyle zamandan kazanç sağlar. Dikkat edilmesi gereken yapay sinir ağları işlenirken sadece aynı katmandaki nöronlar paralel işlenebilir [38]. Buna göre nöron sayısı çok olan yapılar için zaman kazancı artacaktır.



Şekil 4.10 RAPTOR2000

Ayrıca beş FPGA' den oluşan, PCI bus içeren RAPTOR2000 adındaki tasarımda SOM algoritması işlenmiştir [41]. 24 milyon lojik kapı kullanılarak oluşturulmuş tasarımda kullanılan FPGA' ler Virtex olup 128 MB SRAM içermektedir. Algoritma değişiklikleri uygulanarak performans arttırılmıştır. Örneğin Öklid uzaklığı hesaplamaları daha karmaşık olduğundan uzaklık fonksiyonu olarak Manhatten uzaklığı kullanılmıştır. Bu tasarımda saniyede 5GB (gigabits) bağlantı güncellenebilmiştir.

4.5.2.1 Sistemin yapısı ve çalışması

Oluşturulan sistemin FPGA içine yüklenebilmesi için öncelikle algoritmanın VHDL kodu Modelsim kullanılarak simule edilmiştir. Modelsim, yazılan kodun doğru çalışıp çalışmadığını test eder.

UART yapısı sebebiyle Modelsim simülasyonlarında vektörlerin tek tek sisteme giriş olarak verilmesi zaman alıcı olduğu için ve ayrıca sistem kullanılan deneme kartlarıyla da kontrol edileceğinden sadece Modelsim hataları göz önüne alınmıştır. Oluşturulan dizi ve sinyallere bakılarak hatasız VHDL kod oluşturulmuştur. EK7' de Modelsim sonuçları verilmiştir.

VHDL kod ile ifade edilebilen ve doğru çalışan bir sistem FPGA yapısı içinde donanım oluşturamayabilir. Modelsim başarılı olursa, oluşturulan kodun FPGA içinde bir donanım oluşturup oluşturamayacağı Leonardo Spectrum ile test edilir.

Sentezlemede sorun çıkmasına neden olan ya da sentezlenmesi mümkün olmayan yapılar düzenlenmezse sentezleme işlemi gerçekleşmez. Bu sistem için sentezlemede problem oluşturan noktalar aşağıda maddelenmiştir:

1. Gerçel sayılar
2. While döngüsü
3. Rastgele atanan sayılar
4. Döngü yapıları

Sentezleme yapılırken Modelsimden başarıyla geçen VHDL SOM algoritmasında yapılan değişiklikler gerekçeleri ve sisteme olan etkileriyle birlikte bölüm 4.5.2.3' de açıklanacaktır.

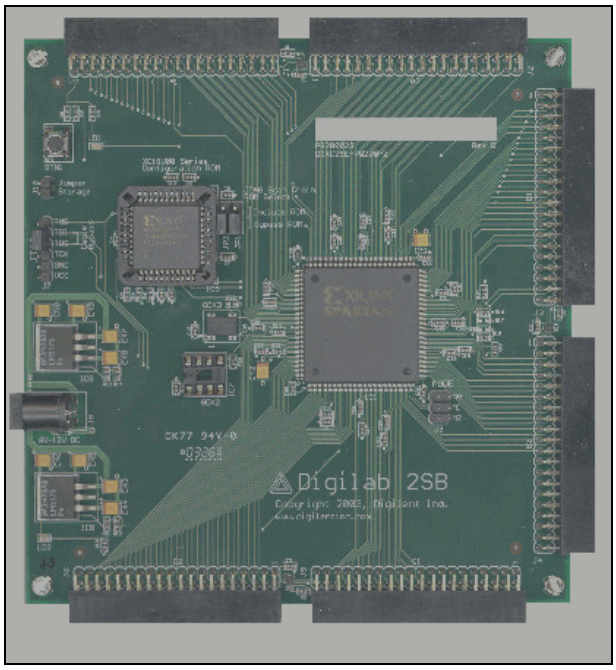
Leonardo spectrum da donanım SPARTAN2E XC2S200E Xilinx FPGA için oluşturulmuş ve Digilab2SB kartı üzerinde kontrol edilmiştir. Oluşturulan sistemde kullanılan FPGA kartı Şekil 4.11' da, çıkışları görmeyi sağlayan I/O kartı Şekil 4.12' de gösterilmiştir. Şekil 4.13' de Digilab 2SB iç blokları gösterilmiştir.

4.5.2.2 Algoritmanın VHDL Basamakları

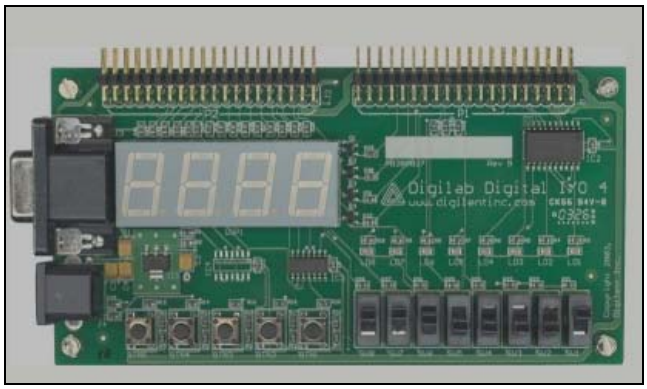
1. MATLAB7' de oluşturulmuş parmak izinin, çerçevelenmiş belli bir bölgesinin açı bilgileri, seri iletişim yoluyla FPGA içerisine aktarılır. FPGA içinde oluşturulmuş bir hafıza dizisinde saklanan bu değerler algoritmada kullanılacak olan giriş vektorlerinin her elemanının 2 byte sabit nokta formatı ifadeleridir.

Hafıza dizisinde saklanan her byte bir sonraki byte ile birlikte bir alt fonksiyon yardımıyla reel değerlere çevrilir. Hafıza dizisindeki 200 byte bir giriş vektörüne denk

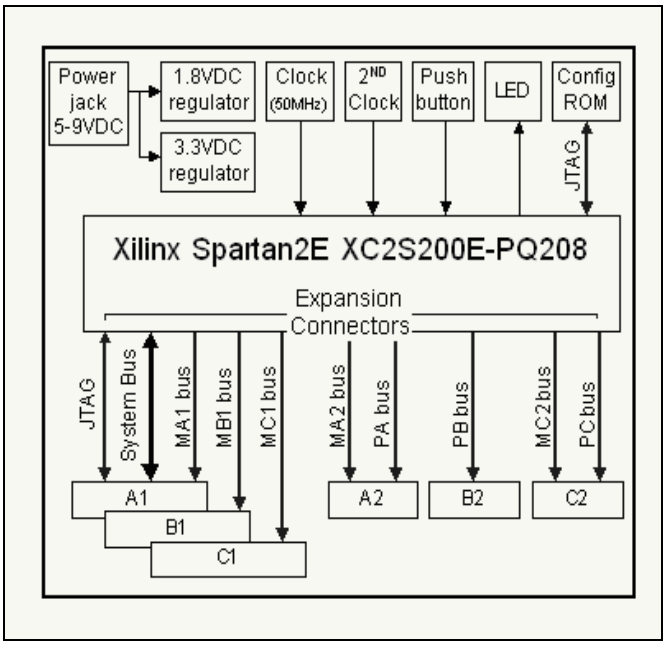
gelir. Kullanılan sabit nokta formatı formatında 11 bit ondalık kısmı, beş bit ise tam değerleri ifade eder



Şekil 4.11 Digilab 2SB

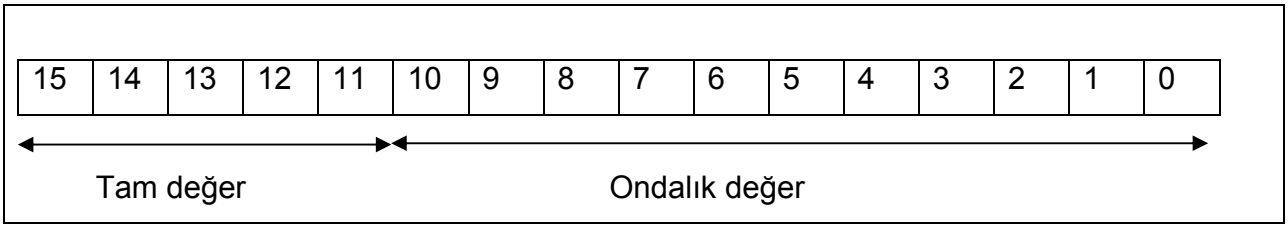


Şekil 4.12 I/O Kartı



Şekil 4.13 Digilab 2SB blokları

Sentezleme Değişikliği: Gerçel sayılar Leonardo Spectrum tarafından sentezlenemediği için integer sayı sistemi kullanılmıştır. Bu sistem için kullanılan gerçel sayılar sabit nokta gösteriminde olduğu için reel ya da integer kullanmak sadece girişlerin katsayısını attırmak anlamına gelir.



Şekil 4.14 Sabit nokta formatı formatı bit tanımları

2. Nöronlara ait ağırlık vektörlerine rastgele değerler atanarak ağırlık vektörleri için başlangıç değerleri belirlenmiş olur.

Sentezleme Değişikliği: Rastgele sayılar oluşturmak için kullanılabilecek donanım yapılarından birisi kullanılabilir. Ya da degişkenlere bağlı bir eşitlikle rastgele sayılar üretilebilir. Bu sistem için değişkenlere bağlı bir yapı kulanılmıştır.

3. Nöronların yarışmacı bir yöntemle giriş vektörlerine atanmasında, giriş vektörleri ile nöronlar arasındaki Öklid uzaklığı belirleyici bir faktördür. Öklid uzaklığı eşitlik (4.7) ile ifade edilir.

$$d_{win} = \min_{j}\left\{\left\|x - w_j\right\|\right\} \qquad (4.7)$$

Her nöron için her girişin Öklid uzaklığı hesaplattırılır ve en küçük Öklid uzaklığı hangi girişe aitse ağırlık vektörü eğitilerek o girişe yaklaştırılır.

Sentezleme Değişikliği: Öklid uzaklığı hesaplanırken karekök kullanılır. Sentezleme esnasında sorun çıkmaması için Öklid uzaklığı almak yerine iç çarpımlara bakılmıştır. İç çarpımı en büyük olan iki nokta, Öklid uzaklığı en küçük iki nokta olduğu için sentezleme kriteri olarak iç çarpım kullanılmıştır.

4. Ağırlıklar eşitlik (4.8) ile güncellenir. Ağırlıkların güncellenmesinde öğrenme katsayısı önemli bir faktördür. VHDL kodun işlem süresini belirleyen en önemli kısım öğrenme katsayısıdır.

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left[x_i(t) - w_{ij}(t)\right]N(j,i) \qquad (4.8)$$

Bu katsayının çok küçük seçilmesi algoritmanın işlem süresini uzatabileceği gibi çok büyük seçilmesi de hedef değere giderken salınımlara sebep olabileceğinden uygun bir değerde seçilmesi önemlidir.

Ayrıca komuşuluk fonksiyonu kullanılarak komşu nöronların ağırlıkları da güncellenir aynı sınıfa ait nöronlar belirlenir.

Sentezleme Değişikliği: Ağırlık güncellemesi while döngüsü ile mümkündür. Sentezleme esnasında döngüden ne zaman çıkılacağı bilinmediği için donanım oluşturulamaz. Döngü sayıları girişte belirlenerek sistem simule edilmiştir.

5. Eğitim aşaması bittikten sonra simülasyonda kullanılacak girişlerin Öklid uzaklıkları belirlenerek hangi sınıfa ait oldukları belirlenir.

4.5.2.3 Algoritmanın Sentezlenmesi ve Simülasyon Sonuçları

Sistem tasarımında Modelsim 5.7, Leonardo Spectrum 2003b kullanılmıştır. Leonardo ile yapılan simülasyonlarda ASIC, Virtex ve Spartan seçenekleriyle sonuçlar nöron sayıları değiştirilerek elde edilmiştir.

SOM algoritması ve sistemde kullanılan seri iletişim protokolü sentezlendiğinde, farklı FPGA tipleri için çalışma frekansları bulunur. FPGA tipi çalışma frekansını yapısı sebebiyle etkiler.

Algoritmada bazı işlemler paralel olarak yürütüldüğü için nöron sayısı ya da giriş sayısı çalışma frekansını etkilememektedir. Ancak çalışma süresini etkiler. Modelsim de oluşturulan gösterimde döngü yapıları sistem kristalinin bir kez tetiklenmesinde birçok işlemi gerçekleştirecek şekilde tasarlanmıştır. Toplam süre ise bulunan çalışma frekansından elde edilen saat sinyalinin üçle çarpılmasıyla bulunur.

Ancak bu kod sentezleme araçlarındaki problemler nedeniyle değişikliğe uğratılarak sentezlenmiştir. Sentezleme işleminde döngü yapıları saat tetiklemesine bağlı olarak seri bir yapıya dönüştürülmüştür. Seri yapıya dönüştürme işlemi tek tetiklemede oluşturulacak donanıma göre daha basit olduğundan çalışma frekansını arttırarak

işlemlerin süresinin seri yapıya rağmen yine de kısa olmasını sağlamıştır. Bunun yanında paralel yürütülen işlemler de süre kısaltılmasında rol oynar.

Çizelge 4.4' de farklı FPGA tiplerine göre çalışma frekansları gösterilmiştir.

FPGA TİPİ	**2 NÖRON**	**5 NÖRON**	**10 NÖRON**
SPARTAN2E (XC2S200EPQ208)	152.9MHz	152.9MHz	152.9MHz
SPARTAN3 (XC3S5000FG1156)	231.1MHz	231.1MHz	231.1MHz
VIRTEX (2VPX70FF1704)	217MHz	217MHz	217MHz
ASIC	217.4MHz	217.4MHz	217.4MHz

Çizelge 4.4 Kullanılan FPGA tipine göre çalışma frekansı

İşleme zamanı frekans, nöron sayısı ve giriş sayısına bağlı olarak değişir. Algoritmadan yola çıkılarak toplam süre eşitlik (4.9) ve (4.10) ile bulunur.

$$toplamclk = \max|giriş, nöron|x100 + (nöronxgirişx100) + (nöronxgiriş) + döngü\_sayisi \quad (4.9)$$

$$toplam\_zaman = \frac{1}{frekans} xtoplamclk \quad (4.10)$$

MATLAB ve FPGA işlem süreleri Çizelge 4.5' de karşılaştırılmıştır. Çizelge oluşturulurken 4000 döngüde sistemin ağırlık güncellemesinin tamamlandığı varsayılmıştır.

NÖRON SAYISI	9	16	25
MATLAB	88.37s	91.78s	111.66s
FPGA(SPARTAN2E)	142.5μs	468.54μs	850.43μs

Çizelge 4.5 MATLAB ve FPGA işlem süreleri

4.5.2.4 Simülasyon Sonuçlarının Değerlendirilmesi

Oluşturulan sistemde nöron sayısı ya da giriş sayısı sistemin çalışma frekansını etkilememektedir. Çalışma frekansını kritik zaman sonucu belirler. Kritik zaman işlem süresi en uzun donanım birimidir. Giriş, döngü saıyısı ve nöron sayısı toplam işleme zamanını etkiler. Bu degerler arttıkça işleme süresi de artar.

Kullanılan FPGA yapısı da sistemin çalışma frekansını etkilemektedir. FPGA' nın yapısındaki donanım birimleri algoritmanın çalışma yapısına uygun olduğunda çalışma frekansı artar. Çalışma frekansının artmasıyla bazı FPGA tiplerinde SOM algoritmasının işlenmesi daha az zaman alır.

5. SONUÇ

Yapılan bu çalışmada iki temel amaç gerçekleştirilmeye çalışılmıştır. Parmak izlerinin sınıflandırılmasında SOM algoritmasının uygulanması ve SOM algoritmasının FPGA kullanılarak gerçekleştirilmesi.

Parmak izi sınıflandırılmasının ve kullanılan yöntemlerin ayrıntılı incelenmesinden sonra sınıflandırma için parmak izinin işlenmesi gerçekleştirilmiştir. Parmak izleri Matlab kullanılarak görüntü işleme teknikleriyle işlenmiştir.

Matlab'da arayüz birimi oluşturularak sırasıyla görüntünün yön dizilim vektörü oluşturulmuş, ardından merkez noktası belirlenmiştir. Belirlenen merkez noktası etrafında parmak izinin karakteristik yapısını içeren alan çerçevelendirilmiştir.

Sınıflandırma işlemi için öznitelik belirleme işlemi yön dizilim vektörleri kullanılarak yapılmıştır. Öz nitelik belirleme işlemi bittikten sonra oluşturulan 100x1 vektörler SOM algoritması kullanılarak sınıflandırılmıştır. Parmak izi uygulaması için SOM performans değerleri MATLAB 7 kullanılarak bulunmuştur.

Performans değerleri kazanan ilk nöronda çok yüksek olmamasına rağmen uygulamanın yapısı sebebiyle kabul edilebilir değerlerdedir. Çünkü parmak izi sadece belirlenen tek bölümde değil veritabanının taranması sonucu sıra ile tüm bölümlerde kazanan nöron kavramıyla aranmaktadır. Veritabanını bölümlere ayıran SOM algoritması ayrıca veri tabanı tarama sırasını da belirlemektedir.

Buna göre yapılan simülasyonlarda parmak izi sınıflandırılması için tek boyutlu 25 nörondan oluşan yapı en yüksek performansı sergilemiştir. Aranan parmak izi bu sonuca göre %77.68 ihtimalle belirlenmiş birinci veritabanı bölümündedir. Eğer bu bölümde bulunamazsa sistemin ikinci bölümde bu izi bulma şansı %83.47 dir.

Çalışmanın ikinci hedefi ise daha öncede belirtildiği gibi SOM algoritmasını FPGA kullanarak gerçekleştirmektir. Algoritmanın VHDL kodlarla ifade edilmiştir. Modelsim hatası içermeyen VHDL ifade uygun bir sentezleme aracı ile FPGA içine gömülü yazılım haline dönüştürülebilir. Bu çalışmada Leonardo Spectrum 2003b kullanılmıştır.

Teorik olarak FPGA içerinde donanım oluşturulabilecek yapılar Modelsimde hata vermez ancak zaman zaman sentezlenmeyebilir. Bu uygulamada da sentezleme işlemini gerçekleştirmek için VHDL kod değişikliğe uğratılmıştır. Paralel tasarlanan bazı yapılar seri olarak işlenmiştir. Paralel yapıların seri yapılara dönüştürülmesi zaman kazancını normal şarlarda azaltır. Ancak paralel yapılar seriye dönüştürüldüğünde oluşturulan donanım ifadesi daha uygulanabilir olduğundan çalışma frekansının artmasıyla bu durum dengelenmiştir.

Kullanılan nöron ve giriş sayısına ve de algoritmanın uygulanacağı döngü sayısına bağlı olarak toplam geçen zaman hesaplanmıştır. Matlab ve FPGA SOM algoritması işleme zamanları karşılaştırılarak, FPGA' nın zamandan kazanç sağlayıp sağlamadığı gözlemlenmiştir.

Yapılan simülasyonlar ve hesaplamalar sonucunda FPGA ile işlemlerin Matlab a oranla oldukça hızlı olduğu görülmüştür.

KAYNAKLAR

[1] Halici U., Ongun G. ,Fingerprint Classification through Self Organization Maps Modified to Treat Uncertainties, Proceedings of the IEEE, Vol84, No10, pp1497-1512, Ekim 1996

[2] Lin Hong. , Automatic Personal Identification Using Fingerprints, Ph.D. Thesis,1998.

[3] Dönmez Mustafa, polis akademisi ders notları, http://geocities.com/mdonmez1/finger2.doc

[4] H. Cummins and C. Mildo. Fingerprints, Palms and Soles. Dover Publication Inc., New York,1961.

[5] A. Moenssens. Fingerprint Techniques. Chilton Book Company, London,1971.

[6] H. C. Lee R.E. Gaensslen. Advances in Fingerprint Technology. Elsevier,New York,1991 pp. 68-70; 142-144

[7] Federal Bureau of Investigation the Science of Fingerprints Classification and Uses. U.S. Government printing Office Washington, D.C. 1984

[8] M. Kawagoe and A. Tojo, “Fingerprint Pattern Classification,” Pattern Recognition, Vol. 17, No. 3, pp. 295-303, 1984.

[9] Prabhakar Salil , 2001, Fingerprint Classification and Matching Using a Filterbank, Doktora tezi , Michigan State University, 259 s.

[10] M. S. Chong, T. H. Ngee, L. Jun, and R. K. L. Gay, “Geometric framework for Fingerprint Classification,” Pattern Recognition, Vol. 30, No. 9, pp. 1475-1488, 1997.

[11] L. Hong and A. K. Jain, “Classification of Fingerprint Images,” 11th Scandinavian

Conference on Image Analysis, June 7-11, Kangerlussuaq, Greenland, 1999.

[12] K. Karu and A. K. Jain, “Fingerprint Classification,” Pattern Recognition, Vol.29, No. 3, pp. 389-404, 1996.

[13] C. L. Wilson, G. T. Candela, and C.I. Watson, “Neural Network Fingerprint Classification,” J. Artificial Neural Networks, Vol. 1, No. 2, pp. 203-228, 1993.

[14] R. Cappelli, D. Maio, and D. Maltoni, “Fingerprint Classification based on Multi-space KL”, Proc. Workshop on Automatic Identification Advances Technologies (AutoID’99), Summit (NJ), pp. 117-120, Ekim 1999.

[15] E. R. Henry, Classfiication and Uses of Finger Prints, London:Routledge, 1900.

[16]Liferig Sahn Atid Xinoori Thang, Combining Exclusive And Continuous Fingerprint Classification, International Conference on Image Processing ,2004 Volume 2, 24-27 Ekim. 2004 s:1245 - 1248

[17] Anil K. Jain, Salil Prabhakar,A Multichannel Approachto Fingerprint ClassificationIEEE Transactıons On Pattern Analysıs And Machine Intelligence, Vol. 21, No. 4, Nisan 1999

[18] Dhruv Batra, Girish Singhal and Santanu Chaudhury, Gabor Filter based Fingerprint Classification using Support Vector Machines, IEEE India Annual Conference 2004, Indicon 2004

[19] Dario Maio and Davide Maltoni, A Structural Approach to Fingerprint Classification,Corso di Laurea in Scienze dell’Informazione, Universith di Bologna, via Sacchi 3,47023 Cesena - Italy.

[20] A. Grasselli, “On the Automatic Classification ofFingerprint - Some consideration of the Linguistic Interpretation of Pictures” in Methodologies of Pattern Recognition, S.

[21] H. Bunke and G. Allermann, “Inexact Graph Matching for Structural Pattem Recognition”, Pattern Recognition Letters, v. 1, pp. 245-253, (1983).

[22] L. G. Shapiro and R.M. Haralick, “Structural Descriptions and Inexact Matching”, IEEE Trans. PAMI, pp. 504-529, (1981).

[23] M. Kaul, “Computing the Minimum Error Distance ofGraphs in O(n’) Time with Precedence Grammars”, in Syntactic and Structural Pattern Recognition, G. Ferrate et. al. eds., NATO AS1 Series, v. F45, Springer Verlag,(1988).

[24] G.T. Candela, P.J. Grother, C.I. Watson, R.A.Wilkinson and C.L. Wilson, “PCASYS – A Pattern-Level Classification Automation System for Fingerprints,” Technical Report NISTIR 5647,1995.

[25] Q. Yuan, T. Jie and R.W. Dai, “Fingerprint Classification System with Feedback Mechanism Based on Genetic Algorithm,” Proc. 14th International Conference on Pattern Recognition, vol. 1, pp. 163-165, 1998.

[26] Sarat C. Dass, Anil K. Jain, Fingerprint Classification Using Orientation Field Flow Curves, Proc. of Indian Conference on Computer Vision, Graphics and Image Processing, (Kolkata), pp. 650-655, Aralık 2004

[27] Dubravko Krašnjak, Vuk Krivec, Fingerprint Classification Using a Homogeneity Structure of Fingerprint’s Orientation Field and Neural Net Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on 15-17 Eylül 2005 s:7 - 11

[28] C. L. Wilson, Masively paralel neural network recognition**,** Neural Networks, 1992. IJCNN., International Joint Conference on Volume 3, 7-11 Haziran1992 s:227 – 232

[29] H. Pourghassem ve H. Ghassemen, Fingerprint Classification with PNN, Signal Processing and Its Applications, 2005. Proceedings of the Eighth International Symposium on Volume 2, Ağustos 28-31, 2005 s:663 - 666

[30] Masayoshi Kamıjo , Classifing Fingerprint Images using Neural Network Deriving the Classification State, Neural Networks, 1993., IEEE International Conference on 28 March-1 April 1993 s:1932 - 1937 vol.3IEEE 1993

[31] Marios S. Pattichis, George Panayi, Alan C. Bovik, and Shun-Pin Hsu, Fingerprint Classification Using an AM–FM Model, IEEE Transactions On Image Processing, VOL. 10, NO. 6, Haziran 2001

[32] FeiSn Jingao Sun AnniCai, Fingerprint Classification Based on Fractal Analysis Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5t International Conference on Volume 3, 21-25 Aug. 2000 s:1471 - 1474

[33] A. Senior, “A Hidden Markov ModelFingerprint Classifier,” Proc. 3 1‘‘ Asilomar Conf.Signals, Systems, and Computers, pp. 305-3 10,1997.

[34] Ali Erol, Ugur Halici, The Effect Of Orintation Map Accuracy On Fingerprint Classification Usıng Pca-Msom , Fourth International Conference on knowledge-Based Intelligent Engineering Systems 6 Allied Technologies, 3 Ağustos 2000, Brighton,UK

[35] Markus Adhiwiyogo, Samuel Chong, Joseph Huang, Weechoon Teo Final Report 18-551 (Bahar yarıyılı 1999)

[36] *Cherry Wang* ,*Huipin Zhang* ,*Andrew Doran* Image Restoration ,1999

http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/Wiener.html

[37] Suliman M Mohammed and Henry O Nyongesa, Automatic Fingerprint Classification System Using Fuzzy Neural Techniques, School of Computing and Management Sciences Sheffield Hallom University , IEEE 2002

[38] Cyprian Grassmann, Tim Schoenauer, Carsten Wolff, PCNN Neurocomputers – Event Driven and Parallel Architectures, ESANN'2002 proceedings – European Symposium on Artificial Neural Networks Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 331-336

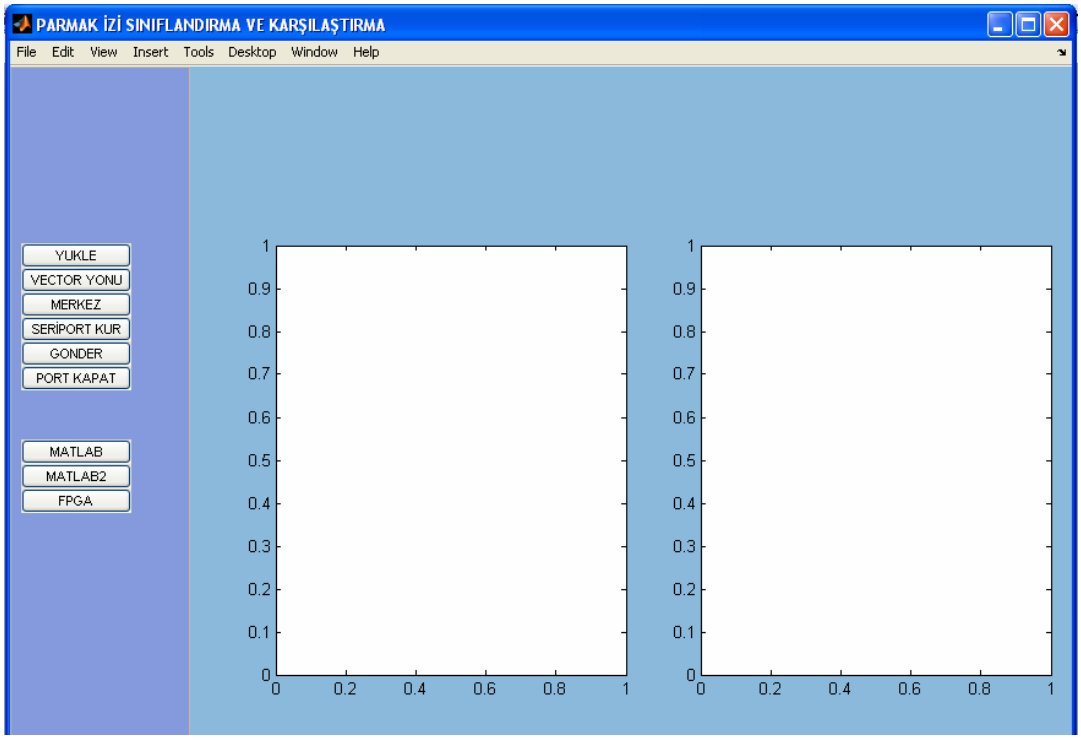
[39] 5. A. Jahnke, U. Roth, H. Klar, “A SIMD/Dataflow Architecture for Neurocomputer for Spike-Processing Neural Networks (NESPINN)”,MicroNeuro'96, p. 232-237, (1996).

[40]R. Greenlaw, H.J. Hoover, and W.L. Ruzzo, *Limits to Parallel Computation* (Oxford University Press, 1995)

[41]A Reconfigurable SOM Hardware Accelerator M. Porrmann, M. Franzmeier, H. Kalte, U. Witkowski, U. RückertESANN'2002 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 337-3

EKLER

EK1 MATLAB ARAYÜZ TASARIMI

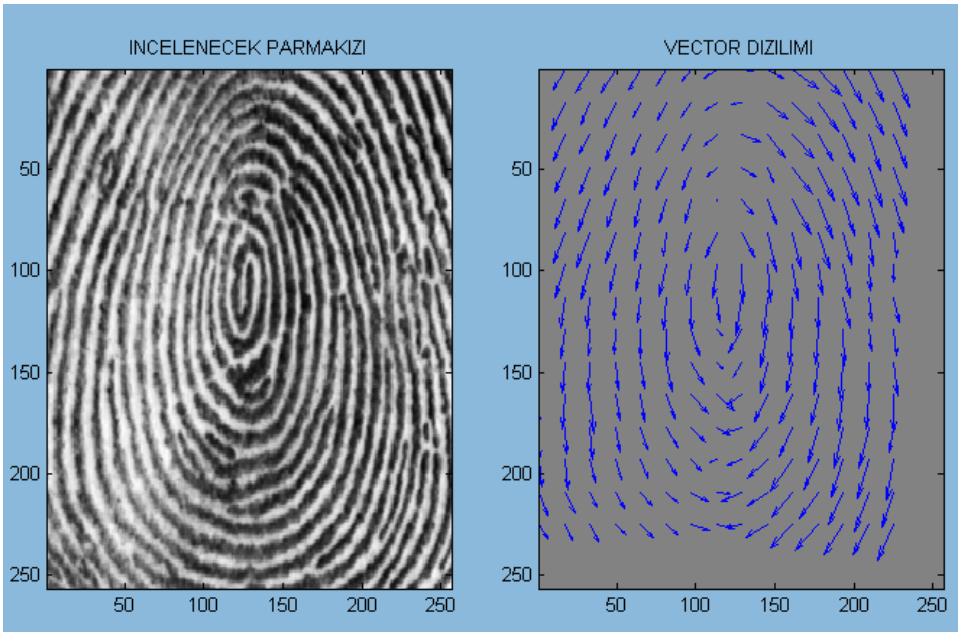



```
clear
FigWin = figure('Position',[250 40 850 600],...
   'Name','PARMAK İZİ SINIFLANDIRMA VE KARŞILAŞTIRMA',...
   'NumberTitle','off',...
   'Color',[0.541176470588235 0.725490196078431 0.858823529411765  ]);
pencere1 = axes('Position',[0.25 0.15 0.33 0.6],...
   'Box','on');
pencere2 = axes('Position',[0.65 0.15 0.33 0.6],...
   'Box','on');
BackColor = get(gcf,'Color');
FrameBox = uicontrol(FigWin,...
   'Units','normalized', ...
   'Style','frame',...
   'BackgroundColor',[ 0.517450980392157 0.605686274509804 0.866470588235294
],...
   'ForegroundColor',[ 0.741176470588235 0.725490196078431 0.658823529411765
],...
   'Position',[0 0 0.17 1]);
```

```
w=16;
baslık='INCELENECEK PARMAKIZI';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,420,90,20],...
   'String','YUKLE',...
   'Callback',...
   ['image1=yukle;'...
    'subplot(pencere1);'...
    'imagesc(image1);'...
    'title(baslık);'...
          'colormap(gray);']);

 Baslık2='VECTOR DIZILIMI';
h=uicontrol(FigWin,...
   'Style','pushbutton',...
          'Position',[10,400,90,20],...
          'String','VECTOR YONU',...
          'Callback',...
   ['subplot(pencere2);[yondizilim,magnitude,image1] =
centralizingyon(image1,1);title(baslık2);']);
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,380,90,20],...
   'String','MERKEZ',...
   'Callback',...
   [  'subplot(pencere1);'...
      '[finalMap,finalCMap,image2] = centralizing2(image1,yondizilim,magnitude,1);']);
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,340,90,20],...
   'String','GONDER',...
   'Callback',...
['[Dmap,Cmap]=gonder(finalMap,finalCMap,s);']);

h=uicontrol(FigWin,...
   'Style','pushbutton',..
   'Position',[10,360,90,20],...
   'String','SERİPORT KUR',...
   'Callback',...
   ['s=kur;'...
          'colormap(gray);']);
 h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,320,90,20],...
   'String','PORT KAPAT',...
   'Callback',.
```

```
   ['s=kaldir(s);'...
         'colormap(gray);']);

h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,260,90,20],...
   'String','MATLAB',...
   'Callback',...
   ['matlab_isleme2;']);

h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,240,90,20],...
   'String','MATLAB2',...
   'Callback',...
   ['som_isleme3;']);
h=uicontrol(FigWin,...
   'Style','pushbutton',...
   'Position',[10,220,90,20],...
   'String','FPGA',...
   'Callback',...
   ['fpga_isleme;']);
```

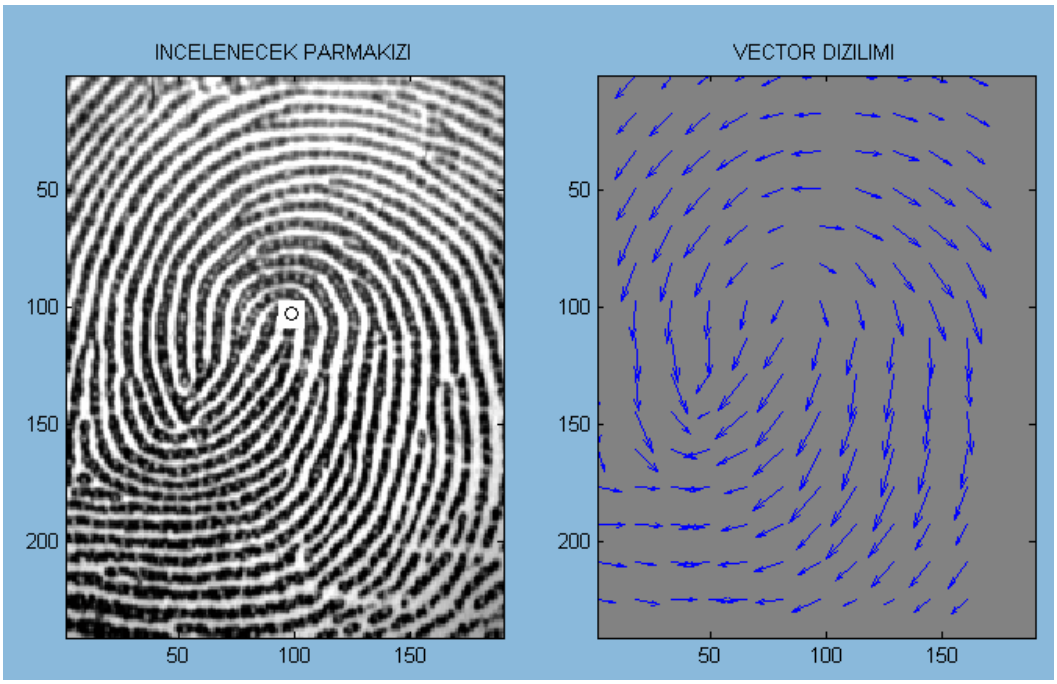
EK2 YÖN DİZİLİM UYGULAMASI




```
function [yondizilim,magnitude,image1] = centralizingyon(image1,ctrl,noShow)
[imgNy,imgNx]=size(image1);
%lowpass filtre olan wiener ile resmin filtrelenmesi
image2 = wiener2(image1,[3 3]);
%resmin gradientinini alınması
[Gx,Gy] = gradient(image2);
%yonvektoru için komponent hesaplanması
orientnum = wiener2(2.*Gx.*Gy,[3 3]);
orientden = wiener2((Gx.^2) - (Gy.^2),[3 3]);
%kesinlik vektorü
certainityden =wiener2((Gx.^2) + (Gy.^2),[3 3]);
W =16;
ll = 6;
yon_deger=zeros(imgNy,imgNx);
orient=[];
kesinlik=[];
anglecizim=[];
[ny,nx]=size(orient);
numblock=[];
denblock=[];
```

```
cer_denblok=[];
% blok işlemleri
for x= 1:W:(imgNx-W)
    for y=1:W:(imgNy-W)
   numblock = orientnum(y:y+W-1,x:x+W-1);
   denblock = orientden(y:y+W-1,x:x+W-1);
   cer_denblok=certainityden(y:y+W-1,x:x+W-1);
   inside_yon = sum(sum(numblock))/sum(sum(denblock));
   angle = 0.5*atan(inside_yon); %yön vektör değerleri bulunmuş olur
inside_certainity=((sum(sum(numblock)))^2+(sum(sum(denblock)))^2)/sum(sum(cer_de
nblok));
   last_certainity= sqrt(inside_certainity*(1/(W*W)));
   anglecizim= 0.5*atan(inside_yon);
   if angle < 0
      if sum(sum(numblock)) < 0
         angle = angle + pi/2;
      else
         angle = angle + pi;
      end
   else
      if sum(sum(numblock)) > 0
         angle = angle + pi/2;
      end
   end

   orient(1+(y-1)/W,1+(x-1)/W) = angle;
   kesinlik(1+(y-1)/W,1+(x-1)/W)=last_certainity;
   anglecizim(1+(y-1)/W,1+(x-1)/W)=anglecizim;
    end
end
yondizilim=orient;
magnitude=kesinlik;
x1=[1:W:(imgNx-W)];
y1=[1:W:(imgNy-W)];
if nargin==2
   imagesc(yon_deger); %boş resim
hold on;
   [u,v] = pol2cart(orient,kesinlik*4);
    % vektörler çizdirilir
   quiver(x1,y1,u,v,1,'b');
    hold off;
end
```

EK3 MERKEZ NOKTASI BULMA UYGULAMASI

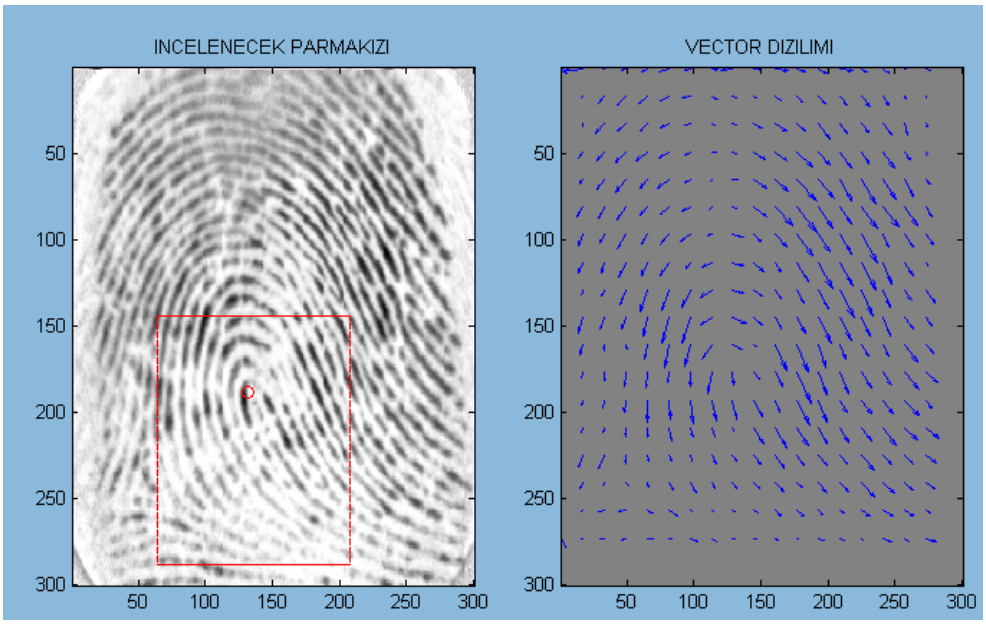



```
function [finalMap,finalCMap,image2] = centralizing2(image1,yondizilim,magnitude,ctrl)
[imgNy,imgNx]=size(image1);
%lowpass filtre olan wiener ile resmin filtrelenmesi
image2 = wiener2(image1,[3 3]);
%resmin gradientinini alınması
[Gx,Gy] = gradient(image2);
%yonvektoru için komponent hesaplanması
orientnum = wiener2(2.*Gx.*Gy,[3 3]);
orientden = wiener2((Gx.^2) - (Gy.^2),[3 3]);
% bloklara ayırma
W = 3;
ll =4;
%başlangıç matrisleri oluşturulur
orient = zeros((floor(imgNy/W)),(floor(imgNx/W)));
[ny,nx]=size(orient);
numblock=[];
denblock=[];

%--------------------------------------------------------------------------
```

```
% YÖN DİZİLM MODELİNİN AÇI VE ŞİDDETLERİ OLŞTURULUR
%-------------------------------------------------------------------------------
for x= 1:W:(imgNx-W)
    for y=1:W:(imgNy-W)

   numblock = orientnum(y:y+W-1,x:x+W-1);
   denblock = orientden(y:y+W-1,x:x+W-1);
   inside = sum(sum(numblock))/sum(sum(denblock));
   angle = 0.5*atan(inside); %yön vektör değerleri bulunmuş olur

   if angle < 0
      if sum(sum(numblock)) < 0
         angle = angle + pi/2;
      else
         angle = angle + pi;
      end
   else
      if sum(sum(numblock)) > 0
         angle = angle + pi/2;
      end
   end
orient(1+(y-1)/W,1+(x-1)/W) = angle;%yöndegerlerinin tamamı pozitif yapılır
    end
end
%-------------------------------------------------------------------------------------------------------------
%%%%%%%%%MERKEZ NOKTASINI BULDURMA%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------------------------------------------------------------

%AÇI DEĞERİ pi/2 den KÜÇÜK OLAN BLOKLAR 1 İLE işaretlenir
binarize = (orient < pi/2);%pi/2 den küçük degerler 1 ile işaretlenir

[bi,bj] = find(binarize);%1 ile işaretlenen noktaların koordinatları
xdir=[];
ydir=[];
for k = 1:1:size(bj,1)%pi/2 değerinin
   i = bj(k);% i ye x degerlerinin atanması
   j = bi(k);% j ye y degerlerinin atanması
 if orient(j,i) < pi/2
      x = fix(ll*cos(orient(j,i)-pi/2)/(W/2));
      y = fix(ll*sin(orient(j,i)-pi/2)/(W/2));
      xdir(j,i) = i-x;
      ydir(j,i) = j-y;

   end
end,
%-----------------------------------------------------------------------
binarize2=zeros(ny,nx);
```

```
for i = 1:1:size(bj,1)
   x = bj(i);
   y = bi(i);
   if ~(xdir(y,x) < 1 | ydir(y,x) < 1 | xdir(y,x) > nx | ydir(y,x) > ny)

     while binarize (ydir(y,x),xdir(y,x)) > 0
        xtemp = xdir(y,x);
        ytemp = ydir(y,x);
        if xtemp < 1 | ytemp < 1 | xtemp > nx | ytemp > ny
           break;
        end
        x = xtemp;
        y = ytemp;
        if xdir(y,x) < 1 | ydir(y,x) < 1 | xdir(y,x) > nx | ydir(y,x) > ny
                        if x-1 > 0
                                while binarize(y,x-1) > 0
                                        x = x-1;
                                        if x-1 < 1
                                                break;
                                        end                     end
                        end
                        break;
        end
      end
   end
   binarize2(y,x) = binarize2(y,x)+1;
end
%---------------------------------------------------------------------
[temp,y] = max(binarize2(1:end-7,:));
[temp2,x] = max(temp);
angle = orient(y(x),x)-pi/2;

%Merkez noktası koordinatları
XofCenter=round(x*W-(W/2)-(ll/2)*cos(angle));
YofCenter=round(y(x)*W-(W/2)-(ll/2)*sin(angle));

%Merkez gösterimi
hold on
plot(XofCenter,YofCenter,'ro');
hold off
```

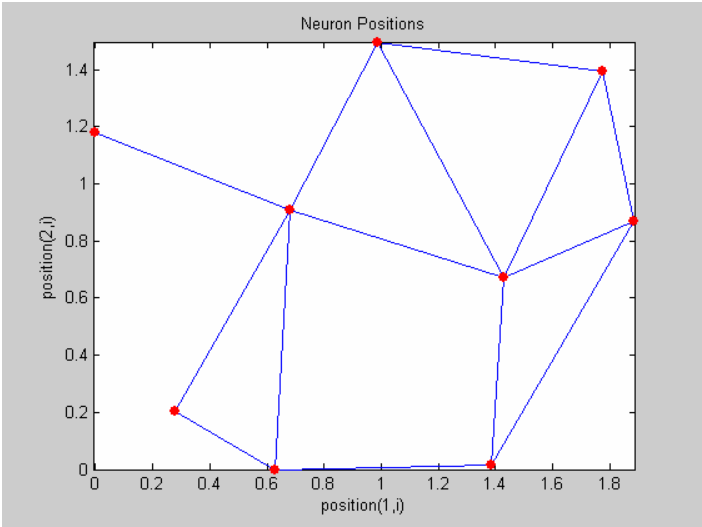
EK4 ÖZELLİK VEKTÖRLERİNİ OLUŞTURMA



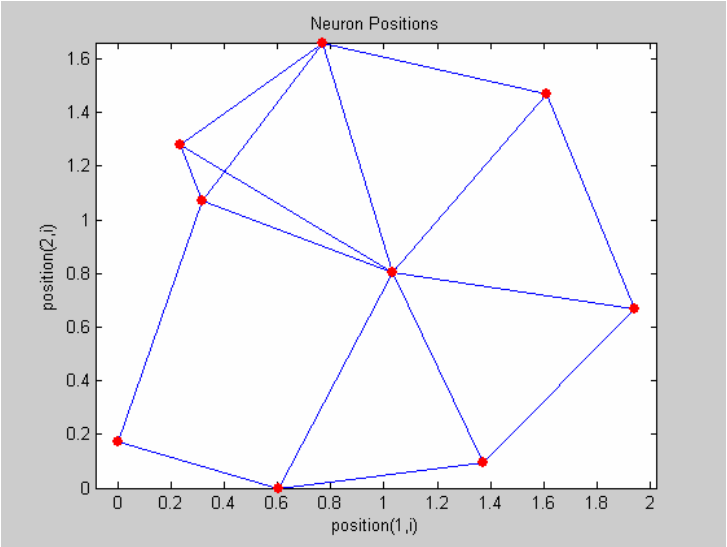

```
%merkez noktasının bloğunun belirlenmesi
xpoint=round(XofCenter/16)
ypoint=round(YofCenter/16)

% yön dizilim bloklarının seçilmesi
[fr,kr]=size(yondizilim)
left = max((xpoint-4),1);
right = min((xpoint+5),kr);

up = max((ypoint-2),1);
bottom = min((ypoint+7),fr);

while (bottom-up)<= 8
   if   bottom == fr
      up = up-1;
   end

   if up<1
      bottom=bottom+1;
   end
 end

while (right-left)<= 8
```

```
  if right==kr
       left=left-1;
    end
    if left<1
       right=right+1;
    end
end
left1=left*16;
right1=right*16;
bottom1=bottom*16;
up1=up*16;
plot(left1:right1,bottom1,'r')
plot(left1:right1,up1,'r')
plot(left1,up1:bottom1,'r')
plot(right1,up1:bottom1,'r')
hold off
%finalMap (up:bottom ,left:right) = yondizilim(up:bottom ,left:right)
%finalCMap(up:bottom ,left:right) = magnitude(up:bottom ,left:right)
finalMap  = yondizilim(up:bottom,left:right);
finalCMap = magnitude(up:bottom,left:right);
[a,b]=size(finalMap);
 finalMap  = reshape(finalMap,a*b,1)
 finalCMap  = reshape(finalCMap,a*b,1);

[k1,k2]=size(finalCMap);
```

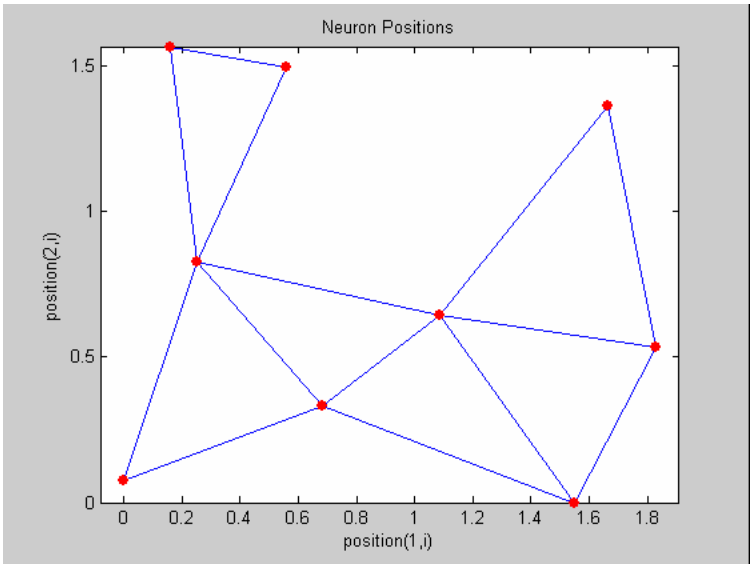
EK 5 MATLAB SOM ALGORİTMASI SİMULASYONU NÖRON DAĞILIMI



2000 döngü kullanılarak oluşan nöron dağılımı



3000 kullanılarak oluşan nöron dağılımı



5000 kullanılarak oluşan nöron dağılımı

EK 6 VHDL SOM ALGORİTMASI

```
----------------------------------------------------------------------------------
--
--                  SOM ALGORİTMASININ VHDL İFADESİ
--
--                          Yasemin Can
--
--                            2006
----------------------------------------------------------------------------------
--kullanılan kütüphanaler
----------------------------------------------------------------------------------
library IEEE;
library work;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.rng_lib.all;
use work.math_lib.all;
use std.textio.all;
use ieee.std_logic_textio.all;

----------------------------------------------------------------------------------
--sistemin en üst seviye girişleri

--sistem uart yapısıyla giriş vektörlerini Matlab dan almaktadır.
--Ledler işlemleri kontrol etmel için kullanılmıştır.
----------------------------------------------------------------------------------
entity Main is
            Port ( TXD          : out std_logic := '1';
                        RXD     : in std_logic := '1';
                        CLK     : in std_logic;
                        led: out std_logic:='0';
                        IO4_ledg    : out std_logic;
                        IO4_led : out std_logic_vector(7 downto 0);
                        RST     : in std_logic        := '0');
end Main;


architecture Behavioral of Main is
----------------------------------------------------------------------------


----------------------------------------------------------------------------
 --Uart için  Dan Peterson tarafından yazılmış kod değiştirilerek
 --sisteme uygulanmıştır
component UARTcomponent

   Port (TXD         : out std_logic      := '1';
                        RXD  : in  std_logic;
                        CLK  : in  std_logic;
                        DBIN            : in  std_logic_vector (7 downto 0);
                        DBOUT           : out std_logic_vector (7 downto 0);
                        RDA             : inout std_logic;
                        TBE             : inout std_logic   := '1';
                        RD              : in  std_logic;
                        WR              : in  std_logic;
                       vector   : in std_logic_vector (7 downto 0);
                        RST             : in  std_logic       := '0');
```

```
end component;


--------------------------------------------------------------------------
--
--------------------------------------------------------------------------
-- Fixed point gösterimi ile Matlab'dan FPGA 'ya aktarılan değerlerin
--gerçel sayılara çevrilmesi

 function Fract2Real(tempvector: in std_logic_vector(15 downto 0)) return real IS
    VARIABLE toadd, sum,sum2,sumson,toadd2 : real;
                                variable fraction_val,TAM_VAL:integer;
                        variable temp_vector: std_logic_vector(15 downto 0);
      BEGIN

      fraction_val:= 11;
           TAM_VAL:=0;
 sum := 0.0;
           sum2:=0.0;

        FOR I IN 0 TO (fraction_val-1) LOOP
         IF tempvector(I)='1' THEN
           sum := sum + toadd;
         END IF;
         toadd := 1.0 / (2.0 ** fraction_val);

        END LOOP;

           FOR J IN 11 TO 15          LOOP
         IF tempvector(J)='1' THEN
                        sum2 := sum2 + toadd2;
                        TAM_VAL:=TAM_VAL+1;
                                        END IF;
          toadd2 := toadd2*2.0;
        END LOOP;
                        sumson:=sum+sum2;
       RETURN sumson;
      END Fract2Real;

--------------------------------------------------------------------------
--
--------------------------------------------------------------------------

           type mainState is (
                        stReceive,
                        stSend);

--------------------------------------------------------------------------
--
--------------------------------------------------------------------------
 constant neuron_sayisi:integer:= 2;
 constant giris_sayisi:integer:=3;
 constant epsilon:real:= 0.2;
 constant ogrenme_katsayisi:real:=0.5;
 constant epsilon_nron:real:= 1.0;

           signal dbInSig     :          std_logic_vector(7 downto 0);
           signal dbOutSig:  std_logic_vector(7 downto 0);
           signal rdaSig      :          std_logic;
           signal tbeSig      :          std_logic;
           signal rdSig       :          std_logic;
           signal wrSig       :          std_logic;
```

```
        signal vector_main :        std_logic_vector(7 downto 0);

        signal stCur       :        mainState := stReceive;
        signal stNext      :        mainState;



type memory1 is array(0 to neuron_sayisi-1 ,1 to giris_sayisi) of real;
signal euclid_uzaklik : memory1;

type memory7 is array(0 to neuron_sayisi-1 ,0 to neuron_sayisi-1) of real;
signal euclid_uzaklik_nron : memory7;

type memory3 is array(0 to neuron_sayisi-1) of integer;
signal neuron : memory3;

type memory2 is array(1 to giris_sayisi) of real;
signal neuron_euclid_array : memory2;

type memory4 is array(1 to giris_sayisi) of real;
signal vectorler:memory4;

type memory8 is array(0 to neuron_sayisi-1 ,1 to 100) of real;
signal agirlik_data_den : memory8;
signal agirlik_data_den_v : memory8;

type memory5 is array(0 to neuron_sayisi-1 ,1 to 100) of real;
signal agirlik_data_den2 : memory5;

type memory9 is array(1 to giris_sayisi,1 to 100) of real;
signal input_data_den : memory9;

type memory6 is array(1 to 600) of std_logic_vector(7 downto 0);
signal sram_data : memory6;

signal g :integer range 1 to giris_sayisi:=1;
signal h : integer range 1 to 200*giris_sayisi:=1;
signal m : integer range 1 to 200*giris_sayisi:=1;
signal i : integer range 1 to 100:=1;
signal j : integer range 0 to neuron_sayisi-1;
signal k,n,o : integer range 1 to 100:=1;
signal initial:integer:=1;
signal hesapla,donguler,gonder:integer:=0;
signal giris_sec:std_logic_vector (31 downto 0):="00000000000000000000000000000000";

-------------------------------------------------------------------------
--
-------------------------------------------------------------------------

begin
-------------------------------------------------------------------------------
---uart
UART: Uartcomponent port map (TXD         => TXD,
                                          RXD       => RXD,
                                          CLK       => CLK,
                                          DBIN => dbInSig,
                                          DBOUT=> dbOutSig,
                                          RDA=> rdaSig,
                                          TBE=> tbeSig,
                                          RD        => rdSig,
                                          WR        => wrSig,
```

```
                                                vector=> vector_main,
                                                 RST => RST
                                                );

        process (CLK, RST)
                begin
                        if (CLK = '1' and CLK'Event) then
                                if RST = '1' then
                                        stCur <= stReceive;
                                else
                                        stCur <= stNext;
                                end if;
                        end if;
                end process;


        process (stCur, rdaSig, dboutsig)
                begin
                case stCur is
                                        when stReceive =>
                                        rdSig <= '0';
                                        wrSig <= '0';

                                        if rdaSig = '1' then
                                                dbInSig <= dbOutSig;
                                                stNext <= stSend;
                                        else
                                                stNext <= stReceive;
                                        end if;
                                when stSend =>
                                        rdSig <= '1';
                                        wrSig <= '1';
                                        stNext <= stReceive;
                        end case;
                end process;

        ------------------------------------------------------------------
        --
        ------------------------------------------------------------------
        --alınan degerlerin hafızaya kaydedilmesi

process(CLK,rdaSig)
  begin
    if rdaSig='1' then
              sram_data(h)<= dbOutSig;
              if h=200*giris_sayisi then
                h<=1;
              else
                        h<=h+1;
                 end if;
                end if;
end process;
------------------------------------------------------------
--ağırlık vektörleri için rastgele sayıların oluşturulması

process(CLK)

 variable r_uni, r_gauss, r_exp : rand_var;
    begin
      r_gauss := init_gaussian(0,0,0, 0.5, 0.5);  -- mean=0, stdev=1
      r_gauss := rand(r_gauss);
```

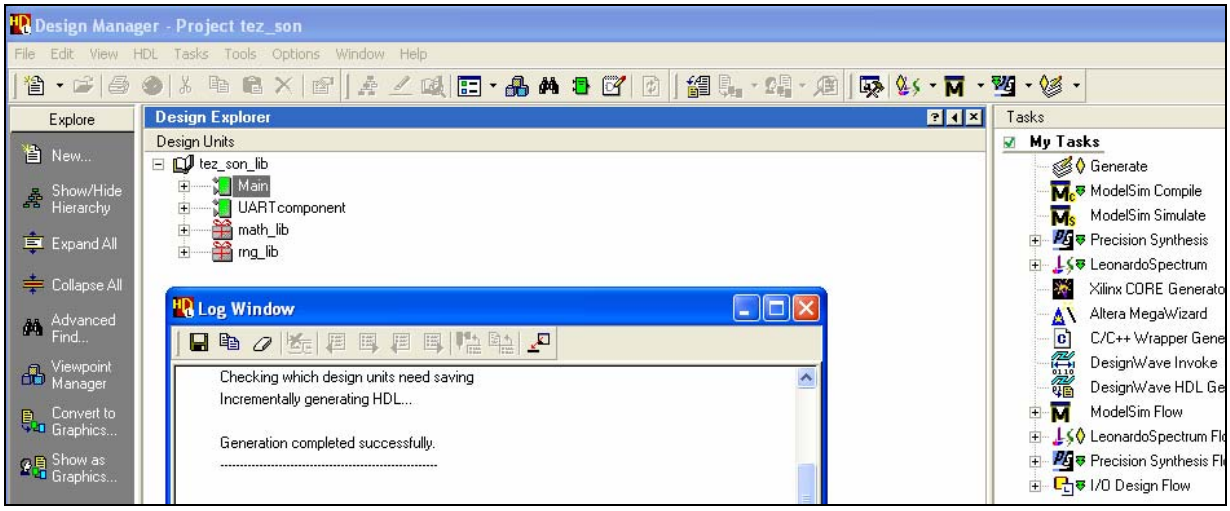
```
 end process;

              led<='1'; --program yüklendi mi kontrol
              IO4_ledg <= '1';
                     IO4_led <=sram_data(h) ; --gonderilen degerler kontrol

--------------------------------------------------------------------------------
 process(CLK,initial)         --başlangıc degerleri
   variable r_uni, r_gauss, r_exp : rand_var;
   type memory is array(0 to neuron_sayisi-1 ,1 to 100) of real;
   variable agirlik_data : memory;
   type memory5 is array(1 to giris_sayisi,1 to 100) of real;
   variable input_data : memory5;

 begin
    if initial=1 then
------ 1. basamak  giriş vektörlerinin hazırlanması

        for g in 1 to giris_sayisi loop
        m<=200*(g-1)+1;
           for k in 1 to 100 loop
           input_data(g,k):= Fract2Real(sram_data(m)& sram_data(m+1));
           input_data_den(g,k)<=input_data(g,k);
           m<=m+2;
           end loop;
        end loop;

-------- 2. basamak=  agirliklara random deger  atama

r_gauss := init_gaussian(0, 0, 0, 0.5, 0.5);  -- mean=0, stdev=1

   for j in 0 to neuron_sayisi-1 loop
      for i in  1 to 100 loop
      r_gauss := rand(r_gauss);
      agirlik_data(j,i):=r_gauss.rnd;
      agirlik_data_den(j,i)<=agirlik_data(j,i);
      end loop;
   end loop;
 hesapla<=1;
end if;
end process;



process(CLK,hesapla)

  variable r,r2,r3,r4,neuron_eslestirme_giris :integer range 0 to neuron_sayisi-1;
  variable t,t2,t3,compare :integer range 1 to giris_sayisi;
  variable s,s2,s3 :integer range 1 to 100;
  variable temp_euclid,temp_euclid2:real;
  variable temp_euclid_nron,temp_euclid2_nron:real;
 begin
    if hesapla=1 then
    initial<=0;
    end if;

--3. basamak euclid uzaklıklarını belirleme
--girişler ve nöronlar arasındaki öklid uzaklıkları

  for r in 0 to neuron_sayisi-1 loop
     for t in 1 to giris_sayisi loop
       for s in 1 to 100 loop
```

```
      temp_euclid :=(input_data_den(t,s)-agirlik_data_den(r,s))**2;
      temp_euclid2:=temp_euclid2 + temp_euclid;
      end loop;
    euclid_uzaklik(r,t)<=sqrt(temp_euclid2);
    end loop;
  end loop;

--nöronlar arasındaki komşuluk fonksiyonu öklid uzaklığı=1 tanımlanmıştır
--nöronlar arasındaki öklid uzaklığı

   for r3 in 0 to neuron_sayisi-1 loop
     for r4 in 0 to neuron_sayisi-1 loop
       for s3 in 1 to 100 loop
       temp_euclid_nron :=(agirlik_data_den(r3,s)-agirlik_data_den(r4,s))**2;
       temp_euclid2_nron:=temp_euclid2_nron + temp_euclid_nron;
       end loop;
     euclid_uzaklik_nron(r3,r4)<=sqrt(temp_euclid2_nron);
     end loop;
   end loop;


--4. basamak girisleri neuronlara atama
--nöronlar öklid uzaklığı en yakın olan nöronlara atanır.

compare:=1;
for r2 in 0 to neuron_sayisi-1 loop
  for t2 in 1 to giris_sayisi loop
    if euclid_uzaklik(r2,compare)<euclid_uzaklik(r2,t2)then
      if giris_sec(t2)='0' then
      compare:=t2;
      end if;
    end if;
  end loop;
giris_sec(compare) <= '1';
neuron(r2)<=compare;
end loop;
donguler<=1;
end process;

--5. basamak agirlik güncelleme
--komşuluk fonksiyonunu sağlayan diğer nöronlarında
--ağırlıkları girişe yaklaştırılır , kalanlar uzaklaştırılır.

process(CLK,donguler)
  variable index:integer range 0 to 100;
  variable r,r1,r2:integer range 0 to neuron_sayisi;
  begin
  if initial=0 then
    agirlik_data_den_v<=agirlik_data_den;
    for r in 0 to neuron_sayisi-1 loop
      for index in 1 to 100 loop
       while (abs(agirlik_data_den_v(r,index)-input_data_den(neuron(r),index)) < epsilon) loop
        agirlik_data_den_v(r,index)<=agirlik_data_den_v(r,index) +
ogrenme_katsayisi*(input_data_den(neuron(r),index)-

agirlik_data_den_v(r,index));
            for r2 in 0 to neuron_sayisi-1 loop
              if (euclid_uzaklik_nron(r,r2) < epsilon_nron) then

              agirlik_data_den_v(r2,index)<=agirlik_data_den_v(r2,index) +
ogrenme_katsayisi*(input_data_den(neuron(r),index)-
```

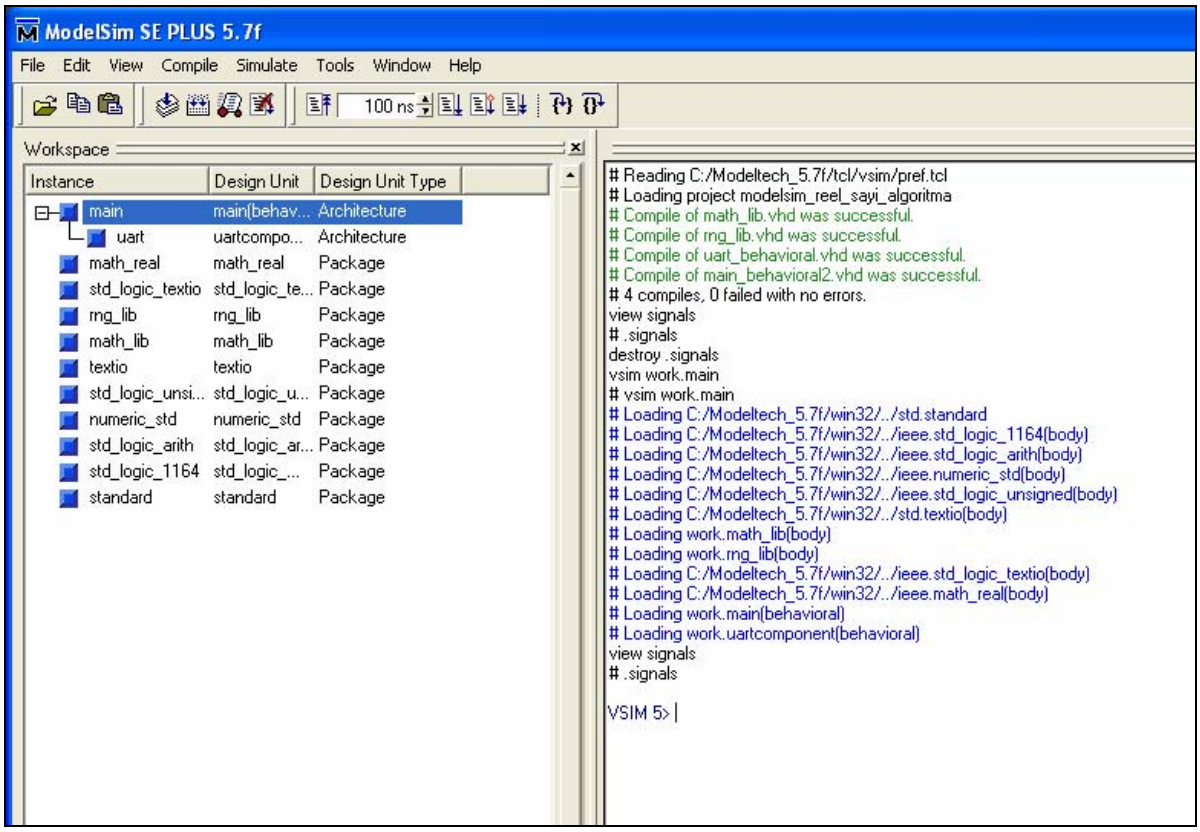
```
agirlik_data_den_v(r2,index));
             else
               agirlik_data_den_v(r2,index)<=agirlik_data_den_v(r2,index) -
ogrenme_katsayisi*(input_data_den(neuron(r),index)-

agirlik_data_den_v(r2,index));
             end if;
           end loop;
       end loop;
      end loop;
 end loop;
 end if;
 gonder<=1;
end process;
--- nöronlara atanan girişleri Matlab a gönderme işlemi
process(CLK,gonder)
  variable r:integer range 0 to neuron_sayisi;
  begin
    if gonder=1 then
      for r in 0 to neuron_sayisi-1 loop
      vector_main<=Std_logic_vector(neuron(r)); --uart tranmit register içine yazılınca
end loop;                                       --matlaba gönderilir.
     end if;
end process;

end Behavioral;
```

EK7 MODELSIM HDL DESIGNER SONUÇLARI

- HDL Designer



- Modelsim



- **EK8** LEONARDO SİMULASYON SONUÇLARI

Sonuçlar Spartan2E XC2S200EPQ208 FPGA kullanılarak,10 nöronlu 20 girişli bir yapı için oluşturulmuştur.

1. LEONARDO SENTEZLEME İŞLEMİ İÇİN OLUŞTURULMUŞ VHDL KOD

```
library IEEE;
library work;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.math_lib.all;
use std.textio.all;
use ieee.std_logic_textio.all;
use ieee.standart.all;

entity Main is
        Port ( TXD          : out std_logic := '1';
                    RXD   : in std_logic := '1';
                    CLK   : in std_logic;
                    led: out std_logic:='0';
                    IO4_ledg   : out std_logic;
                    IO4_led : out std_logic_vector(7 downto 0);
                    RST   : in std_logic        := '0');
end Main;


architecture Behavioral of Main is
---------------------------------------------------------------------------
-- Component Declarations
---------------------------------------------------------------------------
component UARTcomponent
  Port (TXD         : out std_logic      := '1';
                      RXD : in  std_logic;
                      CLK  : in  std_logic;
                      DBIN             : in  std_logic_vector (7 downto 0);
                      DBOUT            : out std_logic_vector (7 downto 0);
                      RDA              : inout std_logic;
                      TBE              : inout std_logic   := '1';
                      RD               : in  std_logic;
                      WR               : in  std_logic;
                     vector   : out std_logic_vector (7 downto 0);
                      RST              : in  std_logic       := '0');

end component;

Component agirlik_guncelleme
  PORT( agirlik_data_den:in integer;
       giris_data_den:in integer;
       CLK           : in std_logic;
       agirlik_data_den2:out integer
      );
END component;
```

```
--------------------------------------------------------------------------------
-- 1.basamak
--------------------------------------------------------------------------------

   function fix_to_int(tempvector: in std_logic_vector(15 downto 0)) return integer IS
VARIABLE toadd, sum,sumson:integer:=0;
-- variable tempvector: std_logic_vector(15 downto 0);
BEGIN

FOR J IN 0 TO 15            LOOP
IF tempvector(J)='1' THEN
sum := sum + toadd;
END IF;
toadd := toadd*2;
END LOOP;
RETURN sum;
END fix_to_int;

--------------------------------------------------------------------------------
--  Local Type Declarations
--------------------------------------------------------------------------------

            type mainState is (
                        stReceive,
                        stSend);

--------------------------------------------------------------------------------
-- Signal Declarations
--------------------------------------------------------------------------------
 constant neuron_sayisi:integer:=10;
 constant giris_sayisi:integer:=20;
 constant epsilon:integer:= 2000;
 constant ogrenme:integer:=5000;

            signal dbInSig     :          std_logic_vector(7 downto 0);
            signal dbOutSig:  std_logic_vector(7 downto 0);
            signal rdaSig      :          std_logic;
            signal tbeSig      :          std_logic;
            signal rdSig       :          std_logic;
            signal wrSig       :          std_logic;
            signal vector_main :          std_logic_vector(7 downto 0);

            signal stCur       :          mainState := stReceive;
            signal stNext      :          mainState;

type memory1 is array(0 to neuron_sayisi-1 ,1 to giris_sayisi) of integer;
signal euclid_uzaklik : memory1;

type memory3 is array(0 to neuron_sayisi-1) of integer;
signal neuron : memory3;

type memory2 is array(1 to giris_sayisi) of integer;
signal neuron_euclid_array : memory2;

type memory4 is array(1 to giris_sayisi) of integer;
signal vectorler:memory4;

type memory8 is array(0 to neuron_sayisi-1 ,1 to 100) of integer range 0 to 65553;
signal agirlik_data_den : memory8;
signal agirlik_data_den_v : memory8;

type memory5 is array(0 to neuron_sayisi-1 ,1 to 100) of integer;
```

```
signal agirlik_data_den2 : memory5;

type memory9 is array(1 to giris_sayisi,1 to 100) of integer range 0 to 65553;
signal input_data_den : memory9;

type memory6 is array(1 to 600) of std_logic_vector(7 downto 0);
signal sram_data : memory6;

signal g :integer range 1 to giris_sayisi:=1;
signal h : integer range 1 to 200*giris_sayisi;
signal m : integer range 1 to 200*giris_sayisi;
signal i : integer range 1 to 100:=1;
signal j : integer range 0 to neuron_sayisi-1;
signal k,n,o : integer range 1 to 100;
signal initial1,initial2:integer:=1;
signal hesapla1,hesapla2,donguler:integer:=0;
signal giris_sec:std_logic_vector (31 downto 0):="00000000000000000000000000000000";
signal clk_g,clk_n,clk_100:std_logic;
signal clk_say3,clk_say2,clk_say1:integer;
signal r,r2,neuron_eslestirme_giris :integer range 0 to neuron_sayisi-1;
signal t,t2,compare :integer range 1 to giris_sayisi:=1;
signal s,s2 :integer range 1 to 100;
signal temp_euclid,temp_euclid2:integer range 0 to 2147483647;
signal index:integer range 1 to 100;
signal rr:integer range 0 to neuron_sayisi-1;

--------------------------------------------------------------------------
-- Module Implementation
--------------------------------------------------------------------------
begin
--------------------------------------------------------------------------------
---uart function
--------------------------------------------------------------------------------
---uart function
UART: Uartcomponent port map (TXD        => TXD,
                                                RXD      => RXD,
                                                CLK      => CLK,
                                                DBIN => dbInSig,
                                                DBOUT=> dbOutSig,
                                                RDA=> rdaSig,
                                                TBE=> tbeSig,
                                                RD       => rdSig,
                                                WR       => wrSig,
                                                vector=> vector_main,
                                          RST => RST
                                                );

        process (CLK, RST)
                begin
                        if (CLK = '1' and CLK'Event) then
                                if RST = '1' then
                                        stCur <= stReceive;
                                else
                                        stCur <= stNext;
                                end if;
                        end if;
                end process;

        process (stCur, rdaSig, dboutsig)
                begin
                        case stCur is
```

```
                                        when stReceive =>
                                                rdSig <= '0';
                                                wrSig <= '0';

                                                if rdaSig = '1' then
                                                        dbInSig <= dbOutSig;
                                                        stNext <= stSend;
                                                else
                                                        stNext <= stReceive;
                                                end if;


                                        when stSend =>
                                                rdSig <= '1';
                                                wrSig <= '1';
                                                stNext <= stReceive;
                                end case;
                        end process;


 process(CLK,rdaSig)
   begin
     if rdaSig='1' then
                sram_data(h)<= dbOutSig;
                if h=200*giris_sayisi then
                  h<=1;
                else
                        h<=h+1;
                 end if;
                end if;
 end process;

------ 1. basamak

process(CLK)
begin
  clk_say1<=clk_say1+1;
  clk_say2<=clk_say2+1;
  clk_say3<=clk_say3+1;

    if clk_say1=100 then
        clk_100<='1';
        clk_say1<=0;
    end if;

    if clk_say2=giris_sayisi then
        clk_g<='1';
        clk_say2<=0;
    end if;

    if clk_say3=giris_sayisi then
       clk_n<='1';
        clk_say3<=0;
    end if;

end process;



 process(CLK,initial1)        --başlangıc degerleri

    type memory5 is array(1 to giris_sayisi,1 to 100) of integer;
```

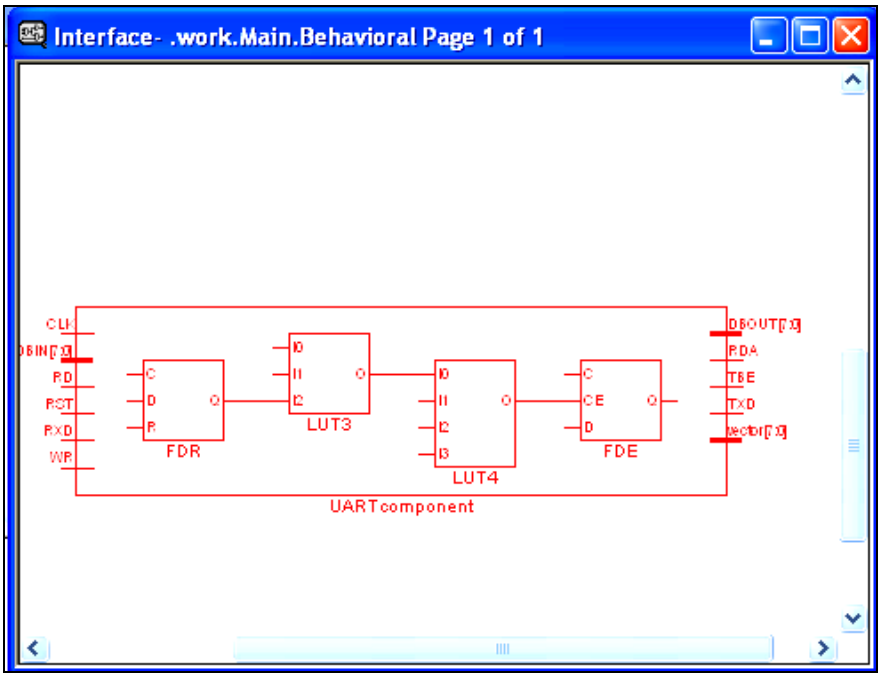
```
   variable input_data : memory5;

  begin
    if initial1=1 then

      input_data(g,k):= fix_to_int(sram_data(m)& sram_data(m+1));
      input_data_den(g,k)<=input_data(g,k);

                if k=100 then
                     k<=1;
        else

                     k<=k+1;
        end if;
     end if;
  end process;


process(clk_100 , initial1)
begin
   if initial1=1 then
      g<=g+1;
      m<=200*(g-1)+1;
   end if;
end process;


process(clk_g,initial1)
   begin
      if initial1=1 then
        hesapla1<=1;
      end if;

end process;
-------- 2. basamak=  agirliklara random deger  atama

process(CLK,initial2)

type memory is array(0 to neuron_sayisi-1 ,1 to 100) of integer;
variable agirlik_data : memory;
begin
    if initial2=1 then
      agirlik_data(j,i):=(i+j)*50;
      agirlik_data_den(j,i)<=agirlik_data(j,i);
              if i=100 then
                     i<=1;
        else

                     i<=i+1;
        end if;
     end if;
  end process;

  process(clk_100 , initial2)
    begin
      if initial2=1 then
       j<=j+1;

       end if;
   end process;

process(clk_n,initial2)
```

```
   begin
     if initial2=1 then
       hesapla2<=1;
     end if;

end process;


process(CLK,hesapla1,hesapla2)


 begin
  if hesapla1=1 then
              if hesapla2=1 then
          initial1<=0;
          initial2<=0;
--                          compare<=1;
--3. basamak euclid uzaklıklarını belirleme

       temp_euclid <=input_data_den(t,s)*agirlik_data_den(r,s);
       temp_euclid2<=(temp_euclid2 + temp_euclid);
         if s=100 then
                        s<=1;
         else

                        s<=s+1;
         end if;

     end if;
 end if;
 end process;

 process(clk_100,hesapla2)
    begin
      if hesapla2=1 then
       euclid_uzaklik(r,t)<=temp_euclid2;

                         if t=giris_sayisi then
                        t<=1;
         else
                        t<=t+1;
         end if;
       end if;
  end process;

  process(clk_g,hesapla2)
    begin
      if hesapla2=1 then
       if r=neuron_sayisi then
                        r<=1;
         else
                        r<=r+1;
         end if;
       end if;
  end process;

--4. basamak girisleri neuronlara atama

process(clk,hesapla1,hesapla2)
begin
  if hesapla1=1 then
    if hesapla2=1 then
```

```
   if euclid_uzaklik(r2,compare)>euclid_uzaklik(r2,t2)then
      if giris_sec(t2)='0' then
       compare<=t2;
       end if;
   end if;
   if t2=giris_sayisi then
                    t2<=1;
        else
                    t2<=t2+1;
        end if;

    end if;
   end if;
end process;

process (clk_g,hesapla1,hesapla2)
begin
if hesapla1=1 then
   if hesapla2=1 then
      giris_sec(compare) <= '1';
      neuron(r2)<=compare;
        if r2=giris_sayisi then
                    r2<=1;
        else
                    r2<=r2+1;
        end if;
     end if;
  end if;
end process;

process(clk_n,hesapla1,hesapla2)
begin
if hesapla1=1 then
   if hesapla2=1 then
   donguler<=1;
    end if;
  end if;
end process;



--5. basamak agirlik güncelleme
process(CLK,donguler,initial1,initial2)

  begin
   if initial1=0 then
            if initial2=0 then
              if donguler=1 then
   agirlik_data_den_v<=agirlik_data_den;
   --         while (abs(agirlik_data_den_v(r,index)-input_data_den(neuron(r),index)) < epsilon) loop
   agirlik_data_den_v(rr,index)<=agirlik_data_den_v(rr,index) + ogrenme*(input_data_den(neuron(rr),index)-
agirlik_data_den_v(rr,index));
   agirlik_data_den_v(rr,index)<=agirlik_data_den_v(rr,index) + ogrenme*(input_data_den(neuron(rr),index)-
agirlik_data_den_v(rr,index));
--            end loop;
   if index=neuron_sayisi then
                    index<=1;
        else
                    index<=index+1;
        end if;
   end if;
           end if;
```

```
            end if;
            end process;

process(clk_n,initial1,initial2,donguler)
begin
if initial1=0 then
              if initial2=0 then
                if donguler=1 then

               if rr=neuron_sayisi then
                        rr<=1;
         else
                        rr<=rr+1;
         end if;
 end if;
            end if;
            end if;
            end process;
end Behavioral;
```

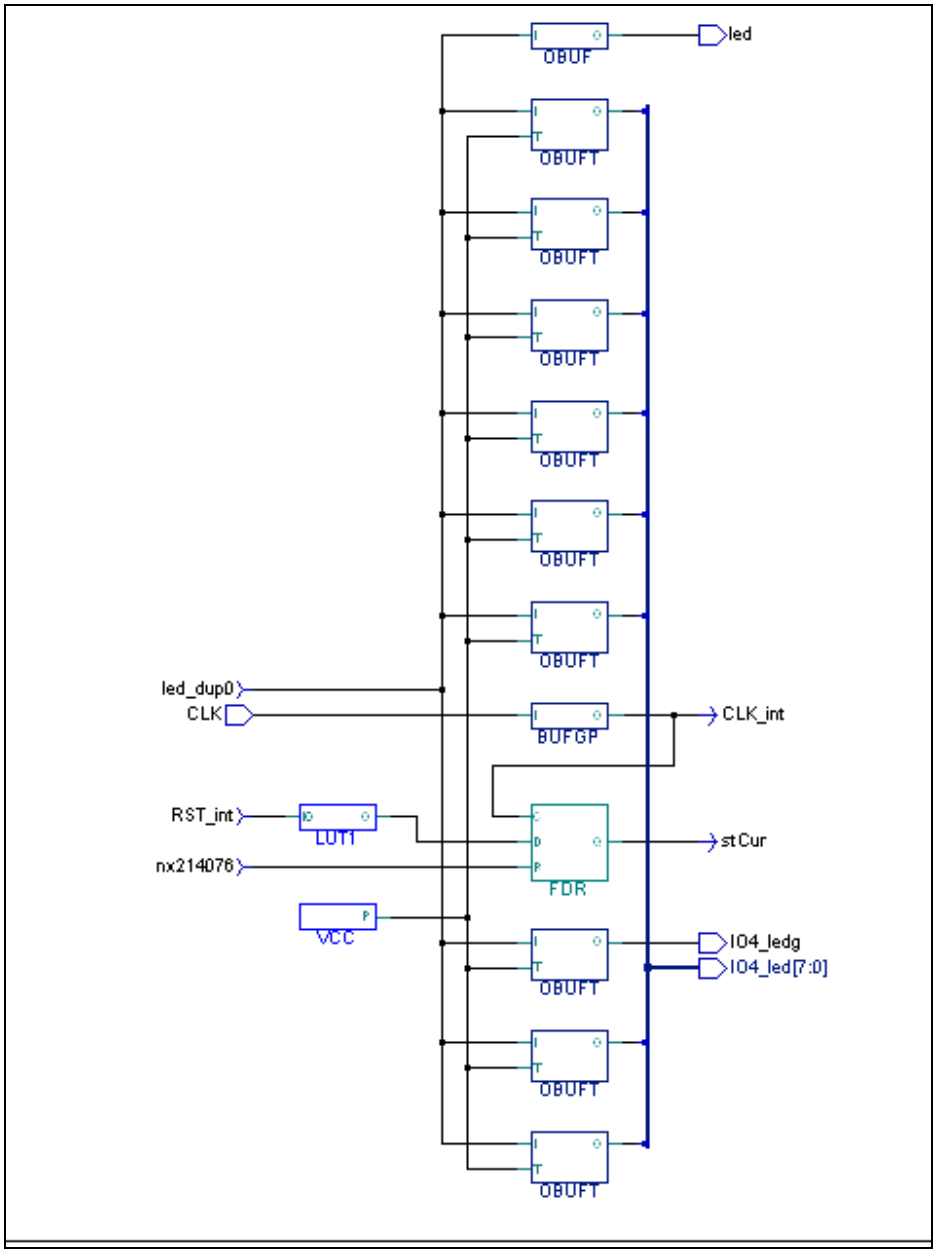
2. KRİTİK ZAMAN ŞEMASI



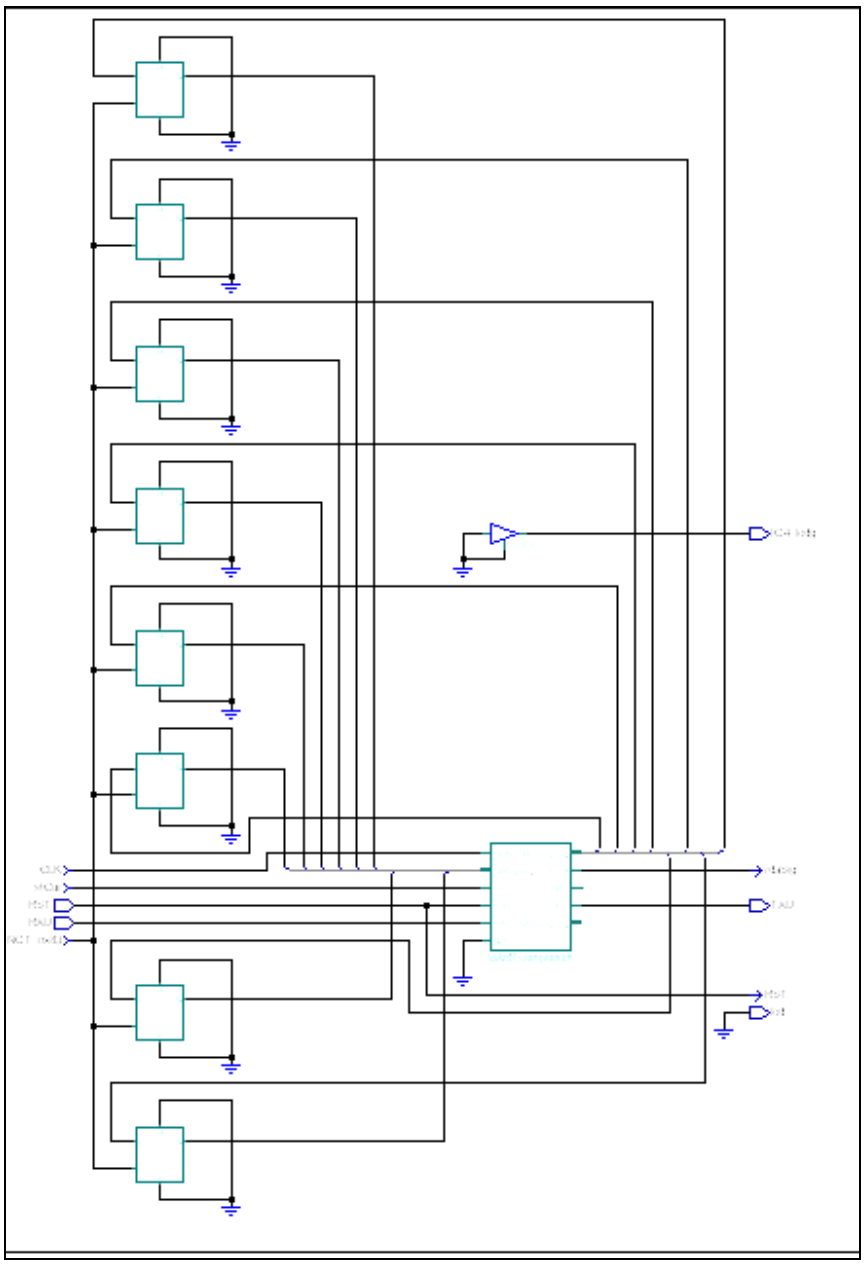
3. KULLANILAN ALAN RAPORU

```
************************************************

Device Utilization for 2s200epq208

************************************************

Resource              Used    Avail   Utilization

------------------------------------------------

IOs                   13      142     9.15%

Function Generators   65      4704    1.38%

CLB Slices            34      2352    1.45%

Dffs or Latches       68      5130    1.33%
```

4.TEKNOLOJİ ŞEMASI



5.RTL ŞEMASI



ÖZGEÇMİŞ

Adı Soyadı : Yasemin Can

Doğum Yeri : Çankırı

Doğum Yılı : 1980

Medeni Hali : Bekar

Eğitim ve Akademik Durumu:

Lise 1995-1998 İnegöl Turgut Alp Anadolu Lisesi

Lisans 1999-2003 Osmangazi Üniversitesi

Yabancı Dil: İngilizce (İleri düzey), Almanca (Orta düzey)

İş Tecrübesi:

Eylül 2003 –Eylül 2004 Merih Asansör Ltd. Şti, Ar-ge Mühendisi.

Temmuz 2006- Elektromed , Yazılım Mühendisi