UZMAN SİSTEM KULLANILARAK TANITILAN FİGÜRLERİN ÖĞRETİMİNİ SAĞLAYAN YAZILIM TASARIMI

2012
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Alperen KETHUDAOĞLU

UZMAN SİSTEM KULLANILARAK TANITILAN FİGÜRLERİN ÖĞRETİMİNİ SAĞLAYAN YAZILIM TASARIMI

Alperen KETHUDAOĞLU

Karabük Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalında

Yüksek Lisans Tezi

Olarak Hazırlanmıştır

KARABÜK

Eylül 2012

Alperen KETHUDAOĞLU tarafından hazırlanan "UZMAN SİSTEM KULLANILARAK TANITILAN FİGÜRLERİN ÖĞRETİMİNİ SAĞLAYAN YAZILIM TASARIMI" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Baha ŞEN .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 21/ 09/ 2012

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan :Doç.Dr. Raif BAYIR (KBÜ)	----------
Üye :Yrd.Doç.Dr. Baha ŞEN (KBÜ)	----------
Üye :Yrd.Doç.Dr. Salih GÖRGÜNOĞLU (KBÜ)	----------

...../...../2012

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Nizamettin KAHRAMAN .....
Fen Bilimleri Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Alperen KETHUDAOĞLU

ÖZET

Yüksek Lisans Tezi

UZMAN SİSTEM KULLANILARAK TANITILAN FİGÜRLERİN ÖĞRETİMİNİ SAĞLAYAN YAZILIM TASARIMI

Alperen KETHUDAOĞLU

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Yrd. Doç. Dr. Baha ŞEN

Eylül 2012, 92 sayfa

Yazı teknikleri gelişirken bunla birlikte şekil, imge, grafik, resim ve tasarım teknikleri de gelişmektedir. Günümüzde insanoğlunun öğrenmesi gereken çok fazla şekil, imge, sembol vardır ve bu çeşitlilik her geçen gün daha da artmaktadır. Az zamanda çok ve etkili bir öğrenme için interaktif ortamlardan faydalanmanın önemi bu çeşitliliğe paralel şekilde gün geçtikçe artmaktadır. Bu çalışma ile bir çevrimiçi figür öğretme sistemi geliştirilmiş, etkin sonuçlar alabilmek için karakter öğretimi üzerinde durulmuştur. Uzman tarafından bilgisayara girilen figürleri, hedef kitleye öğretmek için deterministik yöntemler ve bulanık işlemcileri kullanan bir uzman sistem tasarlanmıştır. Çalışma ile interaktif bir öğrenme modeli oluşturularak, yeni bir öğretim materyali geliştirilmiştir.

Çalışmanın araştırmacılara yeni bir ufuk ve inceleme alanı oluşturması ve bu alanda özgün çalışmalar geliştirilmesi beklenmektedir.

Anahtar Sözcükler **:** Çevrimiçi tanıma, karakter öğretimi, uzman sistem, interaktif öğrenme, bulanık işlem.

Bilim Kodu **:** 902.1.014

ABSTRACT

M.Sc. Thesis

SOFTWARE DESIGN SUPPLYING THE EDUCATION OF ADVERTISED FIGURES BY USE OF EXPERT SYSTEM

Alperen KETHUDAOĞLU

Karabük University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Baha ŞEN
September 2012, 92 pages

While writing techniques are progressing,in parallel with this image,graphics,art and design techniques are also progressing .In today's world, there are a lot of images and symbols for human beings to learn. For a very and effective learning in a limited time, the importance of use from the interactive conditions is increasing in parallel with to this range. Through this study, an on-line figure teaching system has been built up and character teaching has been emphasized to get effective results. To teach the figures which are entered to the computer by the expert to the target group, an expert system which uses the deterministic methods and fuzzy operators has been designed. Having an interactive teaching model through this study, a new teaching material has been built up.

It is supposed to have a new horizon and working area for researchers and to develop original works in this field.

Key Word : Online recognition, character teaching, expert system, interactive learning, fuzzy operation.

Science Code : 902.1.014

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Yrd. Doç. Dr. Baha ŞEN'e sonsuz teşekkürlerimi sunarım.

Sevgili eşim Fatma KETHUDAOĞLU'na gerek tez yazımı konusunda gerekse manevi hiçbir yardımı esirgemeden yanımda olduğu için tüm kalbimle teşekkür ederim.

Sevgili arkadaşım Uğur ÖZDEMİR'e tez yazımı konusundaki yardımlarından ve her türlü desteğinden dolayı tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

$\sum$: n adet değerin toplamı

$\sum$: n adet değerin koleksiyonu

ß : sıralı açılar açı değeri

α : P_i noktası açı değeri

P_i : x ve y koordinat değerleri ile z basınç değerini tutan nesne

$\mu_A(x)$: üyelik fonksiyonu

yp(x) : interpolasyon hesabı

RN1x : referans nokta 1'in x değeri

RN1y : referans nokta 1'in y değeri

RN2x : referans nokta 2'nin x değeri

RN2y : referans nokta 2'nin y değeri

RN3x : referans nokta 3'ün x değeri

RN3y : referans nokta 3'ün y değeri

SO : referans noktalar dizisi

SOB : bulanık referans noktalar dizisi

$\sim, \neg$: bulanık değilleme - tümleme

$\wedge$: bulanık kesişim (ve)

$\vee$: bulanık birleşim (veya)

$\bigcap$: kümelerde bulanık kesişim

$\bigcup$: kümelerde bulanık birleşim

KISALTMALAR

PDL	: Picture Description Language (Resim Tanımlama Dili)
OHCR	: On-line Handwriting Recognation (Çevrimiçi El Yazısı Tanıma)
OFHRCR	: Off-line Handwriting Recognation (Çevrimdışı El Yazısı Tanıma)
ORM	: Object Relational Mapping (Nesne İlişkisel Haritalama)
JPA	: Java Persistence Api (Java Kalıcılık Birimi)
JPQL	: Java Persistence Query Language (Java Kalıcılık Sorgulama Dili)
POM	: Project Object Model (Proje Nesne Modeli)
LPI	: Lines Per Inch (İnç başına çigi)

BÖLÜM 1

GİRİŞ

İnsanoğlu yüzyıllar boyunca birbirleri ile bilgi paylaşımı için çeşitli teknikler kullanmıştır. Bunu kimi zaman Amerika yerlilerinin yaptığı gibi duman ile; kimi zaman Mısırlıların kullandığı Hiyeroglifler ile; kimi zaman da günümüz insanlarının kullandığı gibi harfler ile anlaşarak sağlamışlardır. Yazı teknikleri gelişirken şekil, imge, grafik, resim ve tasarım teknikleri de gelişmiş, günümüzde oldukça karmaşık bir aşamaya ulaşmış ve her geçen gün bu durum daha da karmaşık bir hal almaktadır. Artık insanoğlunun öğrenmesi gereken çok fazla şekil, imge ve sembol vardır. Az zamanda çok ve etkili bir öğrenme için interaktif ortamlardan faydalanmak önem kazanmıştır. Uzman tarafından bilgisayara girilen figürleri, hedef kitleye öğretmek için deterministik yöntemler ve bulanık işlemcileri kullanan bir uzman sistem tasarlanmıştır. Bu projede interaktif bir öğrenme modeli oluşturularak bu alandaki boşluğun doldurulmasına katkı sağlanacaktır. Bu proje ile araştırmacılar için yeni bir ufuk ve inceleme alanına dikkat çekilerek bu alanın yeni çalışmalarla geliştirilmesi ümit edilmektedir. Bu çalışmada daha etkin sonuç alabilmek için karakter öğretimi üzerinde durulacaktır.

Bölüm 2'de Örüntü tanımlama ve alt alanları incelenmiş Karakter tanıma tekniklerinin sınıflandırılması üzerinde durulmuş, çevirim içi ve çevirim dışı karakter tanıma teknikleri ayrıntılı olarak incelenmiştir. Bölüm 3 de Yapay Zeka kavramı tanıtılmış, Bulanık Mantık ve Uzman Sistem hakkında ayrıntılı bilgi verilmiştir. Bölüm 4 de Hata kaynaklarını en aza indirme ve uygulama için en iyi interpolasyon tekniğiniz seçimi üzerinde durulmuştur. Bölüm 5'de geliştirilen uygulama ayrıntılı olarak açıklanmıştır. Bölüm 6'da sonuçlar tartışılmıştır.

BÖLÜM 2

ÖRÜNTÜ TANIMA

Görme olayında, nesnelerin ne olduklarının belirlenmesine *örüntü tanıma süreci* ya da kısaca *tanıma* denir [1].

Aralarında ortak özellik bulunan ve aralarında bir ilişki kurulabilen karmaşık işaret örneklerini veya nesneleri bazı tespit edilmiş özellikler veya karakterler vasıtası ile tanımlama veya sınıflandırma olayına "örüntü tanıma" denmektedir [2]. Bu nesneler uygulamaya göre ses, görüntü ya da sınıflandırılması istenen işaret olabilir.

Örüntü tanımlama teknikleri mühendislik, tıp, askeri ve çeşitli bilim alanlarında yaygın olarak kullanılmaktadır.

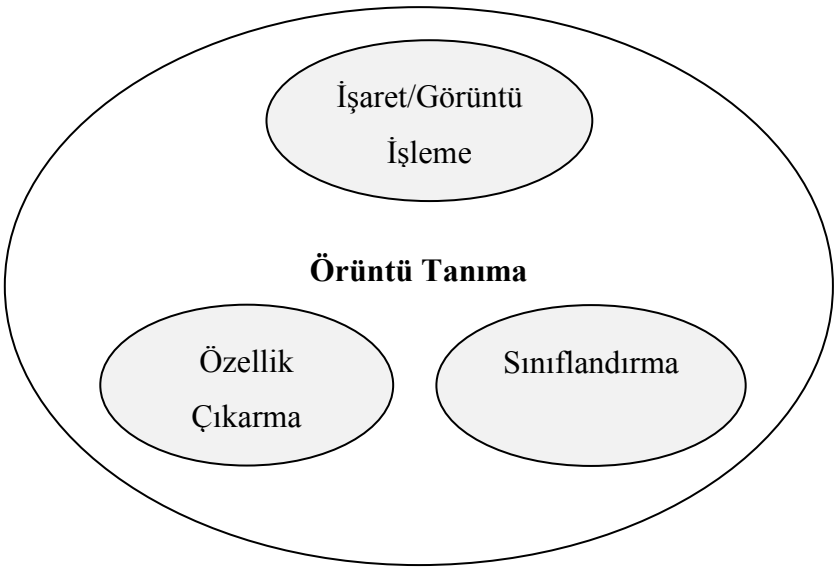
Örüntü tanımlamanın kullanıldığı bazı alanlar aşağıda listelenmiştir.

1. Resim tanıma
2. Karakter tanıma
3. Yüz tanıma
4. Parmak izi tanıma
5. Kenar tanıma
6. Plaka tanıma

Örüntü tanıma kavramı 3 önemli birimden oluşur [3]. Şekil 2.1'de örüntü tanıma kavramı resmedilmiştir.

1. İşaret / Görüntü İşleme: İşaret veya görüntünün filtre edildiği, çeşitli dönüşüm ve gösterim teknikleri ile işlendiği, birleşenlerine ayrıldığı veya modellendiği kısımdır.

2. Özellik Çıkarma: İşaret ve görüntünün veri boyutunun indirgendiği ve tanımlayıcı anahtar özelliklerinin tespit edildiği ve aynı zamanda normalizasyona tabi tutulduğu aşamadır. Sistemin başarımında çok etkili rol oynar.
3. Sınıflandırma: Çıkarılan özellik kümesinin indirgendiği ve formüle edildiği tanımlayıcı karar aşamasıdır.



Şekil 2.1. Örüntü tanıma kavramı.

2.1. KARAKTER TANIMA

Karakter tanıma örüntü tanıma disiplinin bir uygulamasıdır. Karakter tanıma problemi, en genel anlatımla, bir takım semboller içeren bir belgeyi bilgisayara görüntü olarak aktarma ve belgeyi oluşturan karakterleri ön işleme vasıtasıyla tespit ettikten sonra önceden bilinen ya da tanınan karakterlerle eşleştirme işlemi olarak tanımlanabilir [4].

Karakter tanıma genel olarak ön işlemler gerektirmektedir. Bu önişlemlerden bazıları aşağıda listelenmiştir.

1. Eşikleme
2. Karakterlerin çerçevelenmesi(Framing)

3. Genişletme(Dilation)
4. Nomalize Etme
5. İnceltme veya iskelet çıkarma(Thinning)
6. Çizgi parçalarına ayrıştırma
7. Çizgi birleştirme
8. Çizgi tanımlama
9. Çizgi gruplama

2.1.1. Karakter Tanıma Sistemlerinin Sınıflandırılması

Karakter tanıma sistemleri verinin sisteme sunum şekline göre ve verinin kullanıcı ile ilişkisine göre iki şekilde sınıflandırılabilir.

2.1.1.1. Verinin Sisteme Sunum Şekline Göre Sınıflandırma

El yazsısı tanıma verinin sisteme sunumuna göre 2 alana ayrılır [5].

Çevrimdışı (off-line handwriting recognition) el yazısı tanıma**,** yazma işlemi tanıma işleminden önce bitmiştir. Tarayıcı yardımı ile belgenin dijital imgesi çıkartılar ve daha sonra tanıma işlemine geçilir.

Çevrimiçi (online handwriting recognition) el yazısı tanıma**,** dijital tablet benzeri araçlarla özel zaman aralıklarında yakalanan x,y ve z(basınç) değerlerinin anlık olarak tutulduğu sistemler. Tanıma işlemi yazma süresince devam eder.

2.1.1.2. Verinin Kullanıcı İle İlişkisine Göre Sınıflandırma

El yazısı tanımlama sistemleri kullanıcı ile ilişkisine göre de 2 ye ayrılabilir [6].

Yazar- Bağımlı (Writer- Dependent) El Yazısı Tanıma

Yazar bağımlı sistemler daha küçük değişkenlikteki veriler üzerinde çalışması sebebiyle yazar-bağımsız sistemlere göre daha yüksek doğrulukta sonuçlar vermektedir. Kullanıcıya hastır.

Yazar- Bağımsız (Writer- Independent) El Yazısı Tanıma

Yazar- Bağımsız sitem, hedef gurubun farklı yazı sitillerinden dolayı büyük miktarda çeşitlilik içermektedir. Yazar-bağımsız sistemin, çevirim içi ortamda tanınması bile benzer bir şeklin veya harfin temsili farklı sıralı özellikler içerebilmesinden dolayı oldukça zordur.

2.1.2. Çevrimdışı (Offline) Karakter Tanıma

Önceden tarayıcı ile taranarak bilgisayara aktarılan belgeler çeşitli tanıma teknikleri kullanılarak bilgisayar tarafından tanınmaktadır.

2.1.2.1. Çevrimdışı Karakter Tanıma Teknikleri

Karakter tanımada birçok teknik kullanılmaktadır. Bu tekniklerden en çok kullanılanları aşağıda listelenmiştir.

Ağırlık Merkezine Dayalı Tanıma

Bu yöntem karakterlerin ağırlık merkezinin hesaplanmasına dayalıdır. Şekil 2.2'de bazı karakterlerin ağırlık merkezlerine (G(x,y)) işaret edilmiştir. Piksel bazında incelendiğinde ağırlık merkezlerinin aynı noktada olduğu söylenebilir. Oysaki oransal temelde bakıldığında bu değerler farklı olacaktır. Ağırlık merkezine dayalı sınıflandırma sabit karakter boyutlarında (bazı çakışmalar göz ardı edilirse) iyi sonuçlar vermektedir [1].

Şekil 2.2. Bazı harflerin ağırlık merkezi.

Karakterlerin büyüklükleri değiştiğinde ağırlık merkezleri de kaymaktadır. Aynı yazı tipinin farklı büyüklüklerinde de ağırlık merkezi kaymakta ve farklı özellik vektörlerine ihtiyaç vardır.

Korelasyon

Karakter tanıma alanında ilk yaklaşımlardan biri basit optik şablon eşlemedir. Korelasyon en basit anlatımı ile iki örüntü arasındaki Hamming uzaklığının belirlenmesi olarak açıklanabilir. Hamming dizisi aynı uzunluktaki iki dizenin farklı elemanlarının sayısıdır [1].

Kontur Çıkarma

Bu algoritma çerçevelenmiş karakterin 4 yönlü kesit (sol, üst, sağ, alt) eğrilerinin yapılarının incelenmesine dayanmaktadır. Karakter tanıma işlemleri bu 4 yön izdüşümü bilgilerinin karşılaştırılması ile gerçekleştirilmektedir [7].

Aramayı hızlandırmak amacı ile her yönde elde edilen uzaklık bilgileri önce toplanarak toplam izdüşümü değerleri elde edilir. Aynı değerli birkaç karaktere rastlandığında ise oluşturulmuş yön bilgilerini içeren 4 vektörün farkları değerlendirilir. Her yönde normalize edilmiş karakterlerin izdüşümü farklarının minimum olanı muhtemel sonuç olarak verilir.

Tekleştirme Modeline Dayalı Tanıma

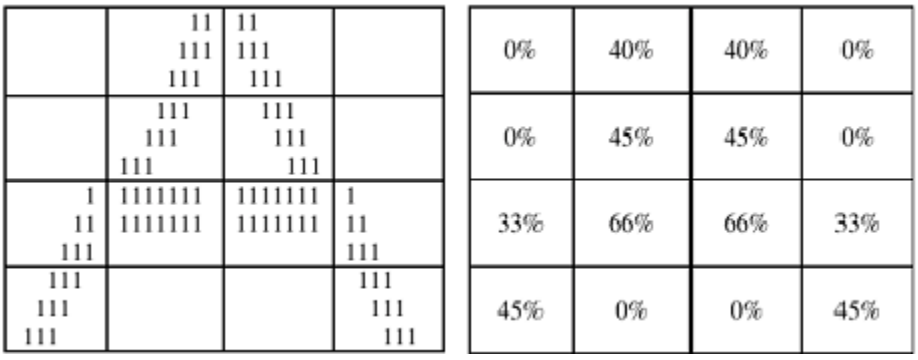
Bu yöntemde karakterlerin iskeletleri modellenmeye çalışılmaktadır. Bu modelleme bir çeşit kontur çıkarmadır [1]. Yalnız bu kontur görüntüdeki dikey ve yatay yönden ayrı ayrı sıkıştırılan değişimleri ifade etmektedir.

Kalıp Kıyaslama

Kalıp kıyaslama, kavramsal olarak çok basit işlemdir. Kalıp, her bir karakterin, sisteme önceden tanımlı imgesidir [8]. Kıyaslama, kalıp imge ile girişe verilen karakter imgesinin üst üste gelen noktaların sayımı ile gerçekleştirilir. Bu işlem tüm kalıplar için gerçekleştirilir. En fazla benzerlik gösteren kalıp, kıyaslamanın sonucu olarak ele alınır.

Desen Tanıma

Şekil 2.3'te görülen bu yöntemde örüntünün desen vektörü kullanılır. Her bir karakterin imgesi matrise dönüştürüldükten sonra bu matrisin her bir hücresindeki pikseller sayılır. Değerler bir vektör halinde birleştirilir [1].

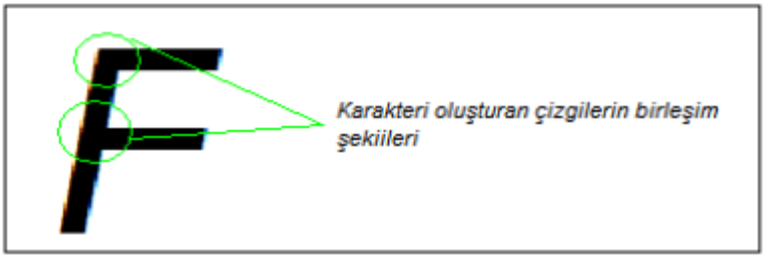


(a) (b)

Şekil 2.3. Desen tanıma örneği, a) Karakter imgesinin bit haritası, b) Her hücredeki siyah bitlerin oranı.

Yapısal Analiz

Bu yöntemde karakterlerin yapısal özelliklerine göre kıyaslama yapılır. Karakterler farklı çizgilerden oluşur. Bu çizgiler farklı açılar altında birleştirilir. Karakter sınıflandırması için bu çizgilerin bağlanma şekilleri ve açıları kullanılabilir [9]. Şekil 2.4'te karakter imgesinin yapısal analizi gösterilmektedir.



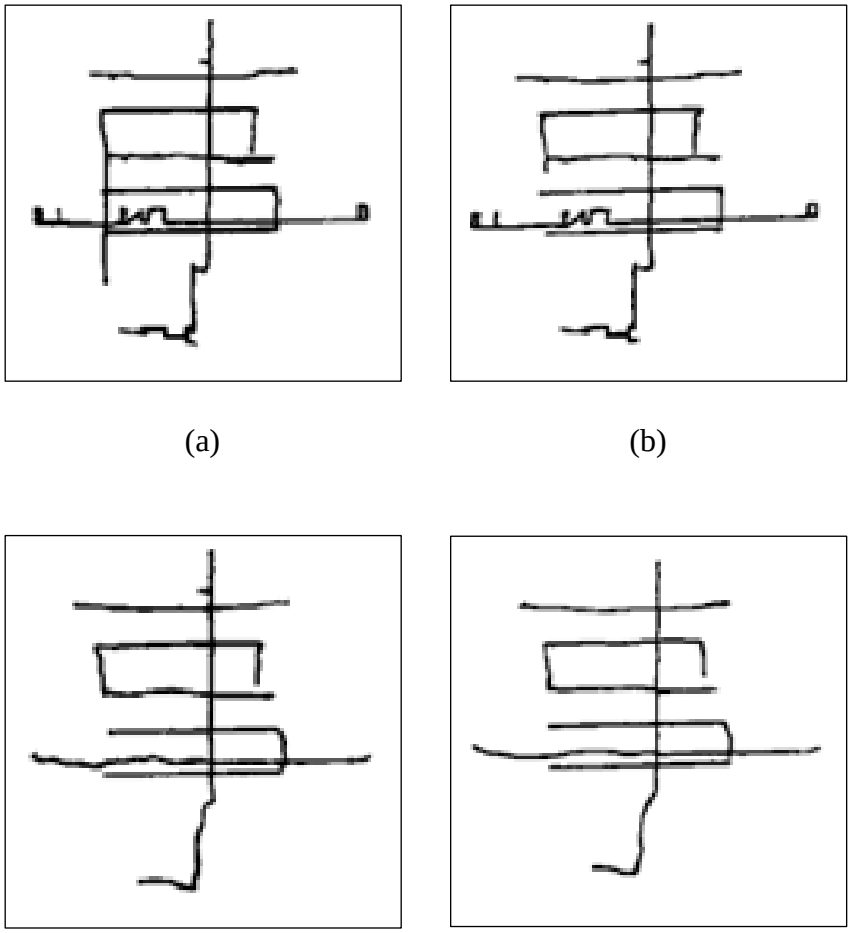
Şekil 2.4. Karakter imgesinin yapısal analiz sınıflandırma yönteminde kullanılan özellikleri.

2.1.3. Çevrimiçi (Online) Karakter Tanıma

Dijital kalem ile çizim yapma esnasında karakterlerin tanınmasını amaçlayan çevrimiçi karakter tanımada literatürde birçok ön işlem ve tanıma tekniği ile karşılaşılmıştır. Aşağıda bunların bir kısmı açıklanmaktadır.

2.1.3.1. Ön İşlemler (Preprocessing)

Karakter tanımada ön işlemler şekil tanıma algoritması uygulamasından önce yapılır. Bölümlemenin yanı sıra, genellikle çizim yumuşatma (smoothing) ve temizleme (cleaning) içerir. Şekil'2.5. te bir figürün ön işlemeye girmeden ve girdikten sonraki hali gösterilmektedir [10].



(a) (b)

(c) (d)

Şekil 2.5. Ön işleme örneği, a) Giriş verisi, b) Sahte çizgi düzeltme, c) Yumuşatma, d) Filtreleme [10].

Bölümleme, tanıma işleminden önce çeşitli yazım veya birleşik kelimelerin yapı parçalarına ayırılması işlemidir. Harici bölümleme ve Dahili bölümleme olmak üzere iki sınıfa ayrılabilir [10].

Harici Bölümleme (External Segmentation) **:** Harici bölümleme, tanıma işleminden önce çeşitli yazım birimlerini ayırmaktır. Örnek olarak yazıyı karakterlere veya kelimelere ayırmaktır.

Dahili Bölümleme (Internal Segmentation) **:** Birçok belgede kelimeler tek bir çizim halinde bileşik el yazısı ile yazılabilir ve bunların tanıma işleminden önce karakterlere ayrılması gerekebilir.

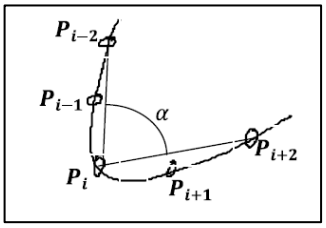
Gürültü Azaltma (Nois Reduction)

Tablet verisindeki gürültüyü azaltabilmek için kullanılan birçok teknik sinyal işleme algoritmalarını içerir. Gürültünün kaynakları tabletin doğruluk sınırları, sayısallaştırma sürecinde düzensiz el hareketleri ve sayısal kalem basıncının ölçümünde meydana gelen hatalardır. Gürültü azaltma yumuşatma, filtreleme, inceltme, sahte çizgi düzeltme, kanca, nokta azaltma ve inme bağıntıları oluşturma algoritmalarını içerir. Aşağıda bu teknikler kısaca açıklanmaktadır.

Yumuşatma (Smoothing)

Genellikle bir noktanın komşuları ile ortalanmasıdır. Yaygın kullanılan bu teknik, her bir noktayı yalnızca bir önceki noktayla ortalar. Bu teknik aynı zamanda alınan her bir noktanın hesaplanmasına da izin verir.

El yazısında veri giriş araçlarına ilişkin donanımsal sınırlar ve bireysel el yazısı stilleri sebebiyle titreşimler oluşmaktadır. A Sharma 2009 yılındaki Doktora tezinde k'nın komşuları anlamına gelen bir yöntem kullanmıştır [11]. Bu titreşimler, her birinin k'ıncı pozisyonunun açısı olarak tanımlanan ve k'nın komşuları (k-neighbors) anlamına gelen listenin her bir noktasının düzeltilmesiyle silinebilir.



Şekil 2.6. P_i noktasında α açısının biçimi [11].

Şekil 2.6'da her iki yandaki iki komşu açıyla nokta düzeltmenin nasıl düşünüldüğü gösterilmektedir. Bu görüntüde bir önceki adımda oluşturulan listede yer alan 5 nokta, çizimin yumuşatılması için kullanılmıştır. P_i Noktasının düzeltilmesi için, P_{i+1}, P_{i+2}, P_{i-1}, P_{i-2} noktalarından faydalanılmaktadır. Aşağıda A Sharma'nın kullandığı algoritma gösterilmektedir [11].

Algoritma:

1. t Ataması: Listedeki çizimlerin sayısı ve (k = 1) atanmaktadır.
2. Her bir çizim için ($k \leq t$) olana kadar 3 aşama tekrar edilecektir.
3. 3.1. k'ıncı çizimdeki noktaların toplam sayısı ile m bulunacaktır.
   3.2. Her P_i noktası için 3.3. ve 3.4. tekrar edecektir. (i= 3,4, ……. , m-2.)
   3.3. P_i'nin α açısı, (P_{i-2}), (P_{i+2}) kullanılarak hesaplanacaktır.
   3.4. $P_{ix}=(P_{(i-2)x} + P_{(i-1)x} + \alpha * P_{ix} + P_{(i+1)x} + P_{(i+2)x}) / (2 * 2 + \alpha)$
   $P_{iy}=(P_{(i-2)y} + P_{(i-1)y} + \alpha * P_{iy} + P_{(i+1)y} + P_{(i+2)y}) / (2 * 2 + \alpha)$
4. k=(k+1)
5. Çık.

Filtreleme – İnceltme (Filtering- Thinning)

Filtreleme bazen inceltme olarak ta anılmaktadır. Çift veri noktalarının elimine edilmesi ve nokta sayısının azaltılmasıdır. Filtreleme biçimi tanıma metoduna bağlı olarak değişmektedir. Filtreleme tekniği, ardışık noktalar arasındaki uzaklığı minimuma zorlamaktır. Bu da çizim noktalarının eşit uzaklıkta olmasını sağlamaktadır. Yazar, çizimi hızlı yaptığında ardışık noktalar arasındaki mesafe minimum uzaklığı aşabilir. Bu durumda çizim noktalarını eşit aralığa getirmek için interpolasyon tekniklerinden faydalanılır.

Diğer bir filtreleme tekniği, çizimdeki ardışık noktaların teğet yönlerini minimum bir değişime zorlamaktır.

Karakter tanımada ilk operasyon olarak Yumuşatma (Smoothing) ve İnceltme (thinnigh) uygulanabilir.

Sahte Çizgi Düzeltme (Wild Point Correction)

Sahte Çizgi Düzeltme genellikle donanımsal problemlerden kaynaklanan sahte noktaların elimine edilmesi, değiştirilmesi veya düzeltilmesidir. Kas yapısının gücü

ile sınırlı el hareketlerinin ivmelenmesi ile el ve kalemin yüksek hızla ivmelenmesi veya yazım hızı Sahte Nokta (Wild Point) oluşmasına sebep olabilir. (Bkz. Şekil 2.5)

Kanca (Dehooking)

Kanca algoritmaları ile çizimin başlangıcında ve daha çok sonunda beliren kancalar elimine edilebilir. Kancalar kalem basıncı yakalama, hızlı yazma veya kalem ucunun düzensiz hareketlerinden ve kalem tablet üzerinden kaldırılırken meydana gelebilen yanlışlıklardır. (Bkz. Şekil 2.5)

Nokta Azaltma (Dot Reduction)

Ardışık olarak tekrar eden noktaları tek noktalara indirmektedir.

İnme Bağlantıları (Stroke Connection),

Harici kalem kaldırışlarını elimine etmeye yarayan bir yöntemdir. Kalemin aşağı ve yukarı ucu arasındaki mesafe karakterin ölçüsüne göre küçük olduğunda, çizimleri bağlayan bir metottur.

Normalizasyon

C.C. Tappert at al (1980), normalizasyonu 4 başlık altında incelemiştir [10].

1. Eğrilik Giderme Algoritmaları (Deskewing Algorithms), karakter eğimlerini düzeltmektedir. Karakter ve kelime eğimlerini düzeltmek için çok sayıda algoritma kullanılmaktadır.
2. Taban Kaydırma Düzeltmesi (Baseline Drift Correction), karakter veya kelimeleri, yatay çizgilere göre veya tabana göre doğrultmaktadır.
3. Boyut Normalizasyonu (Size Normalization), karakteri standart boyuta ayarlamaktadır. Bu yöntem aynı zamanda karakteri merkeze veya sol alt köşeye taşıyarak normelleştirmeyi de içermektedir.

4. Çizim Uzunluğu Normalizasyonu (Stroke Lenght Normalization), bir çizimdeki nokta sayılarını model veri tabanındaki nokta sayılarına daha kolay sınıflandırmak ve sıralamak için eşitlemeyi içermektedir.

2.1.3.2. Çevrimiçi (Online) Karakter Tanıma Teknikleri

Bellegarda et al. 1990 yılındaki bir makalesinde çevrimiçi el yazısı tanıma tekniklerini 5 grupta tanımlamıştır [5].

1. İlkel Ayrıştırma (Primitive Decompisiton)
2. Motor Modeller (Motor models)
3. Elastik Eşleştirme (Elastic Matching)
4. Olasılıksal Modeller (Stochastic Model)
5. Sinir Ağları (Neural Network)

İlkel Ayrıştırma (Primitive Decomposition)

Şekillerin alt parçalarını tanımlar. Bu alt parçalar karakterler için ortak yapı bloklarıdır. Örnek olarak; döngüler, noktalar, geçitler, kavisler, harfin üst çıkıntısı, harfin alt çıkıntısı vs.

Bu metot genellikle çizimi alt çizimlere parçalamak için ön işlem yapmayı gerektirir. Kelime tanımlama daha önce gözlemlenen kelime dizilerini uygun alt çizim dizilerini uygulamak için Dictionary lookup veya hidden markov model gibi birçok model kullanır.

Motor Modeller

Yaygın kullanılan tekniklerdir. Analiz sentezleme olarak bilinir. Çizim segment modelleri karakterleri bağlamak için kurallar ile birlikte oluşturulur. Motor modeller bu çizim bölümlerini kalem ucu hareketlerinin parametreli modeli, insan el hareketlerinin fiziksel özelliklerinin taklidini temsil eder.

Elastik Eşleştirme – Dizi Eşleştirme(Elastic Matching – String Matching)

Dizi Eşleştirme tekniği, karakter desen çiftleri arasındaki mesafeyi ölçen bir yaklaşım kullanmaktadır. Yapılan çizimler, uygun çizim noktaları dizisi, olaylar dizisi gibi bilgileri temsil etmektedir. Bu dizi biçimleri değişken uzunluklarda olabilir. İki dizi arasındaki mesafe A ve B şeklinde adlandırılırsa, A ve B, $e_i^{A'}$ inci ve $e_i^{B'}$ inci uygun olay çiftleri arasındaki uzaklık hesaplarını içerir. Bu işlem, iki olay dizisi arasında bir hizalama gerektirir, hizalanmış benzer olay çiftleri arasındaki mesafe ve diziler arasındaki mesafenin ölçümü, bu hizalamaya dayanmaktadır. Uzaklık hesaplamasında V.Kumar 2011 yılındaki bir makalesinde eşitlik 2.1 de gösterilen Öklidyen (Euklidian) Hesaplaması kullanmıştır [6].

$$d_E(i,j) = \sqrt{(x_i^A - x_j^B)^2 + (y_i^A - y_j^B)^2} \qquad (2.1)$$

S. D. Connel , A. K. Jain 2001 yılındaki bir makalesinde dizi eşleştirme işleminde uzaklık hesaplaması için aşağıda verilen eşitlikler kullanılmıştır [5].

$$d_x(i,j) = |(x_i^A - x_1^A) - (x_j^B - x_1^B)|, \qquad (2.2a)$$

$$d_y(i,j) = |(y_i^A - y_1^A) - (yj^B - y_1^B)|, \qquad (2.2b)$$

$$d_Q(i,j) = \text{Min}(| Q_i^A - Q_j^B | , 360 - | Q_i^A - Q_j^B |), \qquad (2.2c)$$

$$d_E(i,j) = \alpha_d^x(i,j) + \beta_d^y(i,j) + \gamma_d^Q(i,j), \qquad (2.2d)$$

Dizi Eşleştirme uygulaması karakter tanımada Elastik Eşleştirme olarak bilinmektedir [6, 12]. Model ile bilinmeyen arasındaki mesafeyi en aza indirmeye amaçlayan bir veri tabanı modeli şeklinde tanımlanabilen bu model, model veri tabanında bulunan bilinen bir desen ile bilinmeyen bir tanesini eşleştirir.

Eşitlik 2.3'te bilinmeyen noktaların hepsini modelin bir noktası ile karşılaştırmayı sağlayan Scatt'ın [12] kullandığı eşitlikler verilmiştir.

$$D(i,j) = d(i,j) + \begin{cases} \min \begin{cases} D(i-1,j) \\ D(i-1,j-1) \\ D(i-1,j-2) \end{cases} j > 2 \\ \\ \min \begin{cases} D(i-1,j) \\ D(i-1,j-1) \end{cases} j = 2 \\ \min \{D(i-1,j)\}\ j = 1 \end{cases} \quad (2.3)$$

Eğer

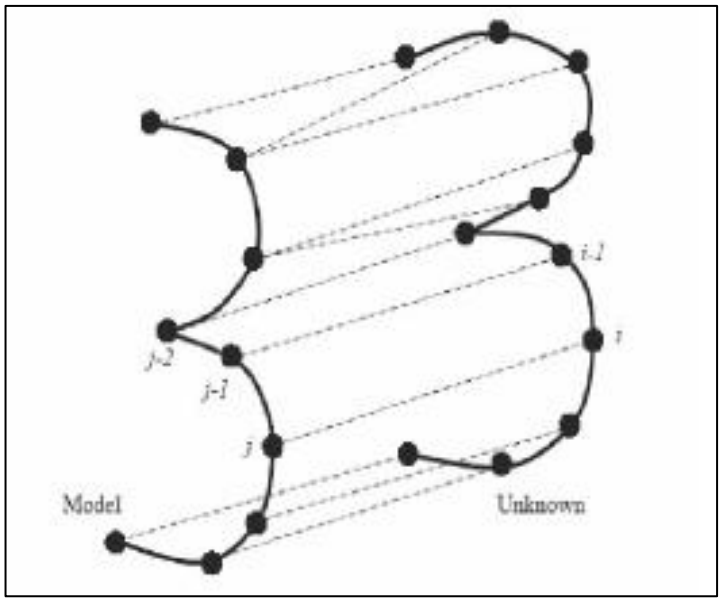
$1 \leq i \leq N_A, 1 \leq j \leq N_b$

Dist(A,B) = $D(N_A, N_B)$

V. Kumar, 2011 [6] tarihli çalışmasında Iteratif Elastik Eşleştirmeyi Elastik Eşleştirmedeki karmaşıklığını kolaylaştıran ve rikörsiflikten kurtaran neredeyse özdeş bir yaklaşım olarak ortaya koymuştur. Aşağıda İteratif Elastik Eşleştirmenin eşitliği verilmiştir. Şekil 2.4 de ise Elastik Eşleştirme gösterilmiştir.

$$DI(i,j) = DI(i-1,j) + \begin{cases} \min \begin{cases} d(i,j) \\ d(i,j+1) \\ d(i,j+2) \end{cases} j < m-1 \\ \\ \min \begin{cases} d(i,j) \\ d(i,j+1) \end{cases} j = m-1 \\ \min \{d(i,j)\}\ j \approx m \end{cases} \quad (2.4)$$

Eğer DI (1,1) = d (1,1) ve j {j, j+1, j+2}'den
{d (i, j), d (i, j+1), d (i, j+2)}'ye bağlı olarak minimum olan değer seçilir.

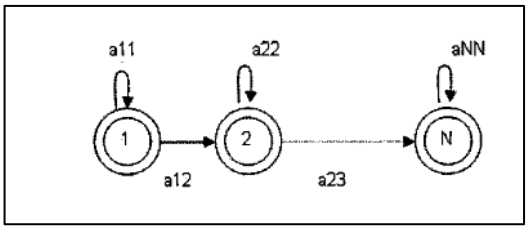


Şekil 2.7. Elastik karşılaştırmanın gösterimi [6].

Olasılıksal Model (Stochastic Model)

Olasılıksal modeller zamansal dizilimi bakımından verileri temsil eden elastik eşleştirmeye'e benzer şekilde sıklıkla kullanılırlar. Birçok yaygın metot, temsil ettiği her bir sınıf için Gizli Markov modelini kullanır. Bu model sıklıkla örnek noktalardan çıkarılan özellikler yada örnek nokta dizileri boyunca ardışık noktalardan oluşturulmaktadır [5, 13].

E. Vural ve arkadaşları (2004) [14], makalelerinde gizli markov modelini kullanarak Türkçe için çevirim içi yazı tanıma sistemi geliştirmişlerdir. Öznitelik olarak x ve y değerlerinin birincil ve ikincil türevlerini ve yüzde olarak basınç değerlerini kullanmışlardır. Eğitim modeli olarak kelime bazlı ve harf bazlı modelleri denemişlerdir. Harf modeli kullanıldığında bütün harfler için, kelime modeli kullanıldığında bütün kelimeler için sabit sayıda durum kullanılmıştır. Gözlem olasılıkları için Gauss dağılımı kullanılmıştır. Harf modeli kullanıldığını zaman, bir kelimenin bütün harflerine karşılık gelen harf modelleri peş peşe dizilerek kelime modeli yaratılmıştır. Şekil 2.8'de E. Vural ve arkadaşlarını çalışmalarında gösterdiği soldan sağa yapıya sahip Gizli Markov Modeli gösterilmektedir.



Şekil 2.8. Soldan sağa yapıya sahip GMM gösterimi [15].

Sinir Ağları (Neural Network)

Karakterlerin yada karakter bölümlerinin tanımlanmasında onların örneklendirilmiş dizileri boyunca ardışık noktaları göstererek sıklıkla kullanılır. Ardışık örnek noktalardan çıkartılan özellikler, ileri beslemeli sinir ağının girdi katmanına geçer.

Bunların dışında literatürde aşağıda açıklanan tanıma teknikleri ile karşılaşılmıştır.

Resim Tanımlama Dili (Picture Description Language)

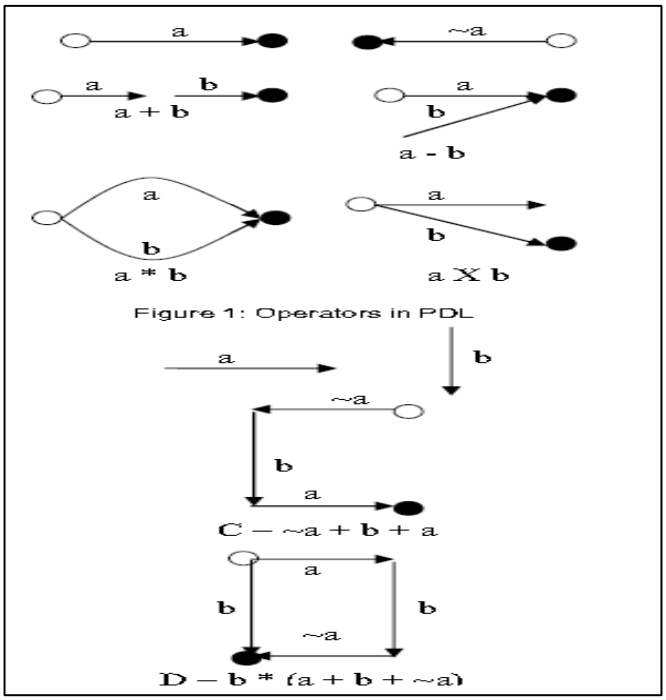
Picture Description Language (PDL) ilk OLCR çalışmalarındandır. Head(Baş) ve Tail (kuyruk) olmak üzere iki bağlantı notası ile düşünülmüştür PDL'nin bağlantı yapısı 4 binary operatör ile tanımlanmıştır. Bunlar (+,-,X,*) operatörleridir.

"~" operatörü ise ters yönü göstermek için kullanılmıştır [15].

Bu yaklaşımın sakıncaları aşağıda listelenmiştir.

1. Bağlantılar yalnız 2 nokta kabul eder.
2. Aynı karakter için PDL yapısı birden fazla olabilir.
3. Yeniden kullanılamazlar, Karakter D, Karakter C ve ilkel 'b' 'nin birleşimi gibi ifade edilemez.

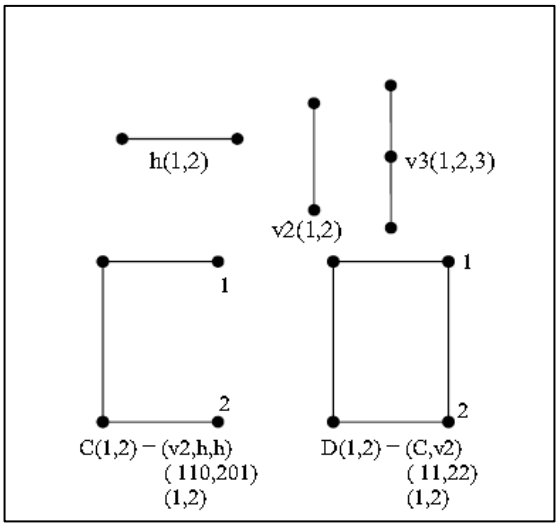
Şekil 2.9'da PDL operatörleri ve C ve D Harflerinin tanımlanması resmedilmiştir.



Şekil 2.9. PDL ifadesi örneği [16].

İkili Gramer (Plex Grammer)

Plex Grammer PDL'nin geliştirilmiş halidir ve onun eksikliklerini gidermek için tasarlanmıştır. Plex Grammer 2 noktadan daha fazla bağlantıyı kabul eder. Plex Grammer birimi basitçe bir n-ilişkili nokta varlığı (n-attachin point entry) veya basitçe nape'ten (boyun) oluşur. Yapıları bağlantı boyunları ile birlikte oluşturulur ve plex structured (ikili yapı) olarak adlandırılır. Bir plex structure 3 komponen içerir, bir liste halinde boyunlar, boyunlar arasındaki bağlantıların dahili bir listesi, ve bir ilişkili noktaların listesi ki diğer boyunlar veya plex structur'lar ile birlikte daha fazla olabilir. Aynı desenler için birden daha fazla plex structur olma olasılığı dez avantajıdır. Plex grammer kullanılarak çizilen bazı karakterler şekil 2.10'da gösterilmektedir.



Şekil 2.10. Plex gramer örneği [17].

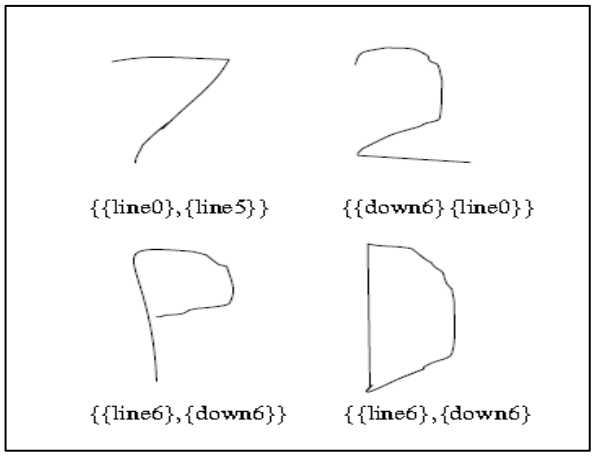
Esnek – Elastik Yapısal Yaklaşım (Flexible Structural Approach)

Flexible Structural Yaklaşımda karakterler 5 ilkel yapı yardımı ile tanımlanır.

1. Line (Çizgi)
2. Up (Yukarı), (Saat yönü tersine giden eğri)
3. Down(Aşağı) (Saat yönünde eğri)
4. Loop (döngü)(Bir noktada kendisi ile bağlanan eğri)
5. Dot (Nokta)(Çok kısa bölüm)

Freeman'ın zincir kodları olarak ta temsil edilen yön ilkel(primitive) yapıları da vardır.

Bu yaklaşımda karakter yapıları primitive dizileri olarak temsil edilir. Bu yaklaşımın dez avantajı bazı karakterler aynı primitive yapı dizilerine sahip olabilmesidir. Şekil 2.11'de P ve D harflerinin temsili gösterilmiştir.



Şekil 2.11. Elastik yapısal yaklaşım örneği [16].

Dr. Suneeta Agarwal ve Vikas Kumar 2005 yılındaki bir makalelerinde PDL, Plex Grammer, Flexible Structural Aproch yaklaşımlarına benzer karakter primitive dizilerine bağlantı bilgilerini de ekleyerek belirsizliği gidermiştir [16].

Bu yaklaşım 3 bilgi alanından oluşmaktadır.

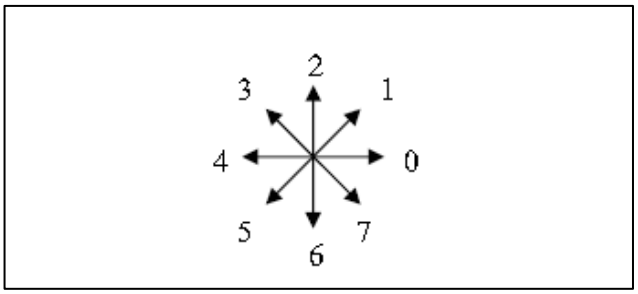
1. Primitive

   Bu Primitive tipleri 4 özellik içerir.

   1.1. Line (Çizgi)
   1.2. Up (Yukarı), (Saat yönü tersine giden eğri)
   1.3. Down(Aşağı) (Saat yönünde eğri)
   1.4. Dot (Nokta)(Çok kısa bölüm)

2. Yönsel Bilgi (Directional Information)

   360 derecelik primitive bilgileri 0 dan 7 toplam 8 eşit parçalı yön bilgileri olarak kullanılmıştır. Şekil 2.12 bu yön bilgilerini resmetmektedir.



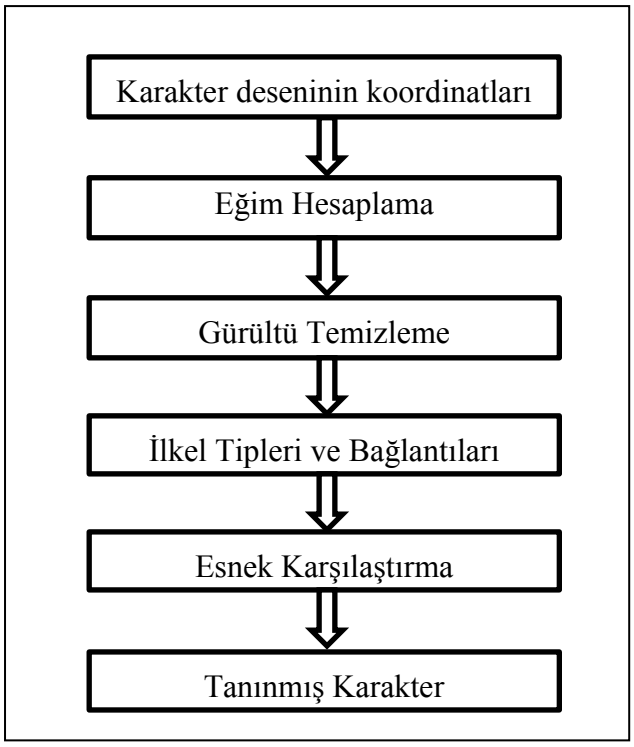
Şekil 2.12. Freeman'ın chain code yön değerleri [16].

3. Göreceli Bağlantı (Relatively Connectivity)

Relatively Connectivity 4 bitlik binary vektörü ifade eder. Bu vektör aşağıda belirtilen 4 nokta için ardışık primitive olup olmadığı bilgisini temsil eder.

3.1. i'inci primitiv'in ilk noktası ile (i+1)'inci primitive'nin ilk noktası
3.2. i.'inci primitve'nin ilk noktası ile (i+1)'inci primitive'nin son noktası
3.3. i'inci primitive'nin son noktası ile (i+1)'inci primitive'nin ilk noktası
3.4. i'inci primitive'nin son noktası ile (i+1)'inci primitive'nin son noktası

Şekil 2.13. de bu tanıma işleminin ana adımları gösterilmektedir. Sürecin başında karakter deseninin koordinatları alınır ve eğimler hesaplanır. Bu koordinatlar kullanıcı karakteri dijital tablete çizerken alınır.



Şekil 2.13. Tanımlama sürecinin aşamaları [16].

Koordinatlar alındıktan sonra ikinci aşama bu çıkartılan koordinat dizilerinden eğimlerin hesaplanması ve bunların yönsel değerler (0-7) dönüştürülmesidir.

Eğim hesaplandıktan sonraki diğer adım koordinatların temsilindeki gürültülü verilerin temizlenmesidir. Bu aşamada belli bir eşik değerinin altındaki eğimler ve buna bağlı nokta çiftleri gibi diğer bilgilerin giriş verisinden silinmesi işlemidir.

En son olarak String Matching işlemi ile sonuca ulaşılmıştır.

2.1.3.3. Online Karakter Tanımanın Avantajları

Online karakter tanıma işlemlerinin ofline karakter tanıma işlemine göre avantajları H. Shing ve Dr. R. K. Sharma tarafından 2007 yılındaki bir makalede 5 başlık altında toplanmıştır [17].

1. Gerçek zamanlı (real-time) bir süreçtir. Verilerin sayısallaştırılması yazma işlemi sırasında yapılır.
2. Gerçek zamanlı olarak uyarlanabilme. Yazar anlık olarak alabileceği geri beslemeler ile yazdığı karakterin tanınma oranını görebilir hatasını düzeltebilir
3. Kelem yörüngesinin geçici ve dinamik bilgilerini yakalar. Bu bilgiler, kalem çizimlerinin/darbelerinin sıra numaraları, her kalem çizim/darbesinin yön bilgileri ve çizim hızı bilgileri sayılabilir.
4. Daha az ön işleme gerektir.
5. Bölümleme kolaydır. Özellikle el yazısı karakterleri için kalem-kaldırma kalem bilgileri ve geçici bilgilerin kullanılması bölümleme operasyonunu kolaylaştırır.

BÖLÜM 3

YAPAY ZEKÂ

Yapay zeka konusuna geçmeden önce, zeka kavramı üzerinde durmak faydalı olacaktır. Fransız felsefeci Taine'ye göre zeka; zihinsel hayatı kurmayı amaçlayan bir araçtır. Binet'e göre ise zeka, insanın sahip olduğu dikkat, bellek, yargılama, akıl yürütme, soyutlama, vs. yetilerin topluluğudur. Amerikan psikolog, Wechsler'e göre zeka, bireyin amaçlı bir şekilde hareket edebilme, mantıklı düşünebilme ve çevresindekilere uyum gösterme yetilerinin tamamıdır [1].

Zekayı tanımladıktan sonra artık insanların zekice olarak kabul ettikleri davranışlara sahip bilgisayarların yapılmasıyla ilgili bilgisayar bilimleri alt alanı olan Yapay Zekaya geçebiliriz.

Yapay Zekanın çağdaş bir bilim dalı olarak gelişmesi 1956 yılında C. Shannon, M.Minsky ve J. McCharty'nin çabaları ve katkıları ile başlamıştır. Yapay Zekâ adı ilk defa 1956 yılında ABD'de "Makine Zekâsı" konferansında ortaya konmuş bir kavramdır [18].

Yapay zeka çok kapsamlı bir konu olup tanımını yapmak oldukça zordur. Birçok alt alanı içermekte ve birçok alanla da ilintilidir.

Literatürde, *yapay zekâ* terimine uygun çok sayıda tanım yapıldığı görülmektedir. Bunlardan bir kısmı aşağıda listelenmiştir.

Slage'ye göre yapay zekâ; sezgisel programlama temelinde olan bir yaklaşımdır [1].

Yapay zekâ, bir makine ya da insan eliyle üretilmiş otonom bir sistem kullanarak insan zekâsının benzetimini yapmaya çalışan bir araştırma alanıdır.

Yapay zekâ, insanın düşünme yöntemlerini analiz ederek bunların benzeri yapay yönergeleri geliştirmeye çalışan araştırma alanıdır.

Yapay zekâ, canlılarda (özellikle insanlarda) bulunan algılama, öğrenme, çoğul kavramları bağlama, düşünme, fikir yürütme, sorun çözme, iletişim kurma, çıkarım yapma ve karar verme gibi yüksek bilişsel fonksiyonları ve otonom davranışları sergilemesi beklenen yapay bir sistemdir (donanım+yazılım) [7].

Tester'e göre yapay zekâ, şu ana kadar yapılamayanlardır [1].

Yapay zekâ sistemlerinin en önemli özelliği olaylara çözümler üretirken bilgiye dayalı karar mekanizmalarını çalıştırabilmeleri ve eldeki bilgilerle olayları öğrenerek, sonraki olaylar hakkında karar verebilmeleridir.

3.1. YAPAY ZEKÂNIN ALT ALANLARI

İnsanların problem ve/veya olayları algılamadaki, çözümlemedeki farklılıklar yapay zeka çalışmalarında da farklı teknolojilerin doğmasına sebep olmuştur. Bu teknolojilerin bir kısmı aşağıda listelenmiştir.

1. Bulanık Mantık (Fuzzy Logic)
2. Dağıtık Zeka (Distributed Intelligence)
3. Doğal Dil İşleme (Natural Language Processing)
4. Genetik Algoritma (Genetik Algorithms)
5. Görüntü İşleme (Image Processing)
6. Konuşma İşleme (Speech Processing)
7. Makine Öğrenmesi (Machine Learning)
8. Örüntü Tanımlama (Pattern Recognition)
9. Robotik (Robotics)
10. Uzman Sistemler (Expert Systems)
11. Veri Madenciliği (Data Mining)
12. Yapay Sinir Ağları (Artificial Neural Networks)

3.2. BULANIK MANTIK

Bulanık Mantık konusu anlatılmadan önce mantık, klasik mantık, sembolik mantık ve çok değerli mantık terimlerinin kısaca açıklanmasında fayda görülmüştür.

Mantık Arapça nutk kelimesinden Türkçemize geçmiştir [19]. N. Baykal ve T. Beyan(2004) [20] Bulanık Mantık İlke ve Temelleri adlı kitabında bulanık mantığı "Doğru öncülerden doğru sonuçlar çıkarma biçimlerini inceleyen bilim dalıdır" diye tanımlamışlardır.

Önermelerden geçerli çıkarımla bulmayı amaçlayan klasik mantık, Aristo mantığı, iki değerli mantık, siyah beyaz mantık olarakta bilinmektedir. Günlük dile dayalıdır ve kısmen semboliktir. İki değerli mantıkta akıl yürütme özdeşlik, çelişmezlik, ve üçüncünün olmazlığı ilkelerine dayanır. Aşağıda Aristoles mantığındaki akıl yürütme ilkeleri açıklanmaktadır.

Özdeşlik :Bir şey ne ise odur.
Çelişmezlik :Bir şey hem kendi kemde başka bir şey olamaz.
Üçüncünün olmazlığı :Bir şey ya A'dır ya da A- olmayandır. Üçüncü bir durum yoktur.

Sembolik mantık ise matematiksel bir yapısı olan mantık türüdür. Sembolik mantık Lojik, Matematiksel Mantık ve Modern Mantık gibi adlarla da anılmaktadır.

Sembolik mantık özdeşlik, çelişmezlik ve üçüncünün olmazlığı ilkelerine dayanan iki değerli mantıktır. Sembolik mantığın amacı önermelerden geçerli çıkarımlara ulaşmaktır. Ama bunu yaparken klasik mantıktaki gibi içeriksel değil biçimsel doğruluk aranarak yapar.

Çok değerli mantık ise ikiden fazla değeri kabul eden mantık olarak tanımlanabilir [20].

Bulanık mantık bulanık küme teorisin dayanır. Bulanık küme teorisi ile sonsuz değerli mantık arasında bir benzerlik vardır. Bulanık mantığa bulanık kümeleri ve bulanık bağıntıları kullanan sonsuz değerli mantık bile demek mümkündür [20].

Sembolik mantık kuralları ile sadece somut değil, soyut düşüncelere dayalı önermelerde yapılarak genel çıkarımların elde edilmesi mümkündür. Ancak bunu yaparken kavram ve terimlerdeki belirsizlik ve bulanıklıkların işin başında durulaştırılarak kesinlik kazandırılması gerekir. Bundan dolayı, sembolik mantık idealleştirilmiş kavram ve terimlerle önermelerden çıkarılacak ideal sonuçları içerir. Oysa gerçek dünyada bulanıklık ve belirsizlik kaçınılmazdır. İşte bu esnada bulanık mantık devreye girer. Bulanık mantığın diğer mantık sistemlerinden ayıran en önemli özelliklerden birisi, üçüncünün olmazlığı ilkesi ve çelişmezlik ilkesi olarak adlandırılan ve diğer mantık sistemlerin için oldukça önemli olan, hatta temel kural denebilecek iki özelliğin bulanık mantık için geçerli olmamasıdır [20].

Zadeh[1] bulanık mantığın genel özelliklerini şu şekilde tarif etmiştir [31].

1. Bulanık mantıkta kesin değere dayalı düşünme yerine yaklaşık değerlere dayanan düşünme kullanılır.
2. Bulanık mantıkta her şey [0,1] aralığında belirli bir derece ile gösterilir.
3. Bulanık mantıkta bilgi büyük, küçük, çok az gibi sözel ifadelerle şekillendirilir.
4. Bulanık çıkarım işlemi sözel ifadeler arasında tanımlanan kurallar ile yapılır.
5. Her mantıksal sistem bulanık olarak ifade edilebilir.
6. Bulanık mantık matematiksel modeli çok zor elde edilen sistemler için çok uygundur.

Bulanık kümelere dayalı olan bulanık mantık genelde, insan düşüncesine özdeş işlemlerin gerçekleşmesini sağlamakla, gerçek dünyada sık sık meydana gelen belirsiz ve kesin olmayan verileri modellemede yardımcı olmaktadır. Örnek olarak "Top kalenin yanından geçti" veya "Top kalenin çok yakınından geçti". Burada

[1] Lotfi A.Zadeh Bulanuk Mantığın kurucusudur [20].

kalenin (yakınının) ve (çok yakınının) kaç cm veya metre olduğuna değinilmemiştir. Diğer bir örnek olarak havayı tanımlarken, bulutluluk yüzdesi vermek yerine genellikle güneşli, yağmurlu veya kapalı demekteyiz [1].

3.2.1. Bulanık Deyim

Bulanık deyim fonksiyonu [0,1] aralığından [0,1] aralığına olan bir fonksiyon olup f:[0,1]→[0,1] olarak gösterilir. N-boyutlu bir tanım kümesi için ise fonksiyonumuz $f:[0,1]^n \rightarrow [0,1]$ şeklinde ifade edilir. Klasik mantıkta kullanılan, değilleme (~, ¬), kesişim (ve, ∧) ve birleşim (veya, ∨) gibi klasik mantıkta kullanılan işlemcileri kullanır.

Bulanık deyimlerin özellikleri [20];

1. Doğruluk değerleri, 0 ve 1 arasındadır.
2. F bir bulanık deyim ise , ~f de bir bulanık deyimdir.
3. F ve g bulanık deyim ise, f ∧g ve f ∨g de bulanık deyimlerdir.

3.2.2. Sözel Değişkenler

Bir değişkene sözel terim atanırsa sözel değişken adını alır. Bulanık sözel değişkenler bulanık yüklem ve bulanık sıfat olarak iki parçadan oluşur.

Bulanık yüklem birincil terimdir. Bulanık sıfatlar da bunu niteleyen çok, olası, hemen hemen imkansız, aşırı, olası olmayan gibi kelimlerdir. Sıfatlar öncülün anlamını değiştirmede kullanılır ve iki sınıfta toplanabilir. Bir kısım sıfatlar bulanık doğruluk niteleyici veya bulanık doğruluk değerleri olarak tanımlanır. Bunlara örnek olarak kesinlikle doğru, çok doğru, daha az veya doğru, çoğunlukla yanlış kelimeleri sayılabilir. Bulanık niteleyiciler ise ,çok birkaç, hemen hemen, tüm, genellikle gibi kelimlerdir.

3.2.3. Bulanık Kümeler

Klasik küme, kümeye kesinlikle ait (üye) veya kesinlikle ait değil (üye değil) biçiminde iki grubun oluşturulması ile anlamlıdır. Klasik kümede üye olanlarla olmayanlar arasında kesin bir fark vardır. Bulanık küme, kesin geçişleri elimine ederek belirsizlik kavramının tanımını yeniden verir ve evrendeki bütün bireylere üyelik derecesi değerini atayarak matematiksel olarak tanımlar.

Klasik kümenin karakteristik fonksiyonu, evrensel kümede her bireye ya 1 ya da 0 değerini atar. Böylece bu fonksiyon, evrensel kümenin elemanlarının 0 ve 1'den oluşan bir kümeye çerçeveler [1].

$$\mu_A = X \rightarrow \{0,1\} \tag{3.1}$$

Bu fonksiyon, elemanın küme içerisindeki üyelik derecesini veren özel bir aralıkla evrensel kümenin elamanlarına 0 ve 1 arasında değer atayan fonksiyon olarak genelleştirilirse bulanık küme üyelik fonksiyonu elde edilir.

$$\mu_A = X \rightarrow [0,1] \tag{3.2}$$

3.2.3.1. Bulanık Kümelerin Gösterimi

Bulanık kümeler, küme elemanlarının üyelik derecelerine göre sıralanarak veya bir üyelik fonksiyonu tanımlayarak gösterilirler.

$$A=\{0.1, 0.5, 0.7\} \tag{3.3}$$

veya

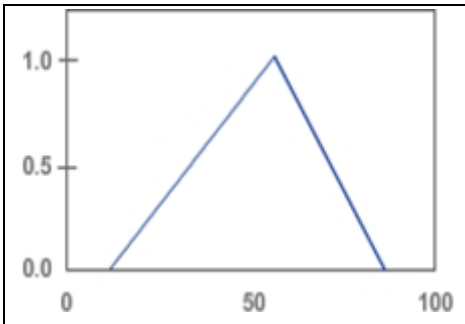
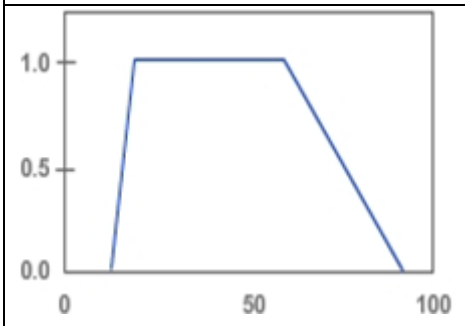
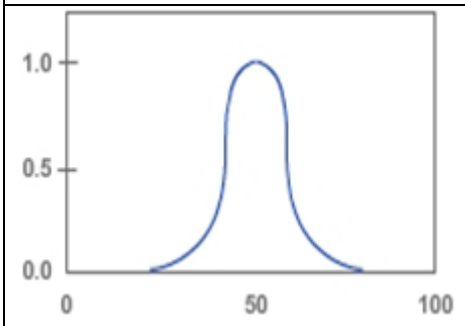
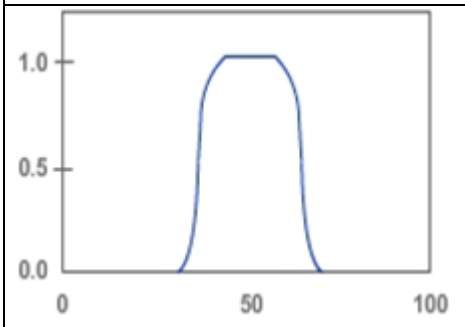
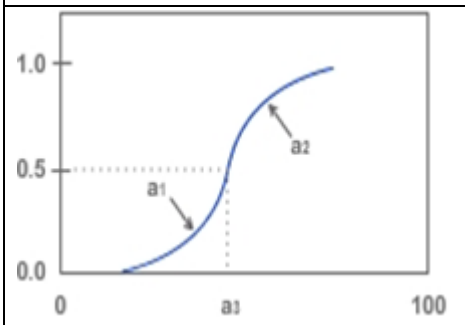
$$A=\{\sum \mu_A(x_i)/x_i\} \tag{3.4}$$

Açıklama: Buradaki $\sum$ işareti küme öğelerinin topluluğunu, / (bölme) işareti ise üyelik derecelerinin sırasını belirtir.

3.2.3.2. Üyelik Fonksiyonu Tipleri

Pratikte çok sayıda üyelik fonksiyonu vardır. Bunlardan en çok kullanılanları (x), üçgen, yamuk, çan eğrisi, Gaussian ve sigmoidal fonksiyonlardır. Bunların dışında S, π_1 ve π_2 gibi spesifik üyelik fonksiyonları da vardır. Çizelge 3.1 de çeşitli üyelik fonksiyonları ve formülleri verilmiştir.

Çizelge 3.1. Bulanık mantıkta kullanılan çeşitli üyelik fonksiyonları.

	**Üçgen Üyelik Fonksiyonu** Bir üçgen üyelik fonksiyonu a_1, a_2 ve a_3 olmak üzere üç parametre ile tanımlanır. $\mu_A(x;a_1, a_2, a_3)=\begin{cases} a_1 \le x \le a_2 \; ise \; (x-a_1)/(a_2-a_1) \\ a_1 \le x \le a_2 \; ise \; (a_3-x)/(a_3-a_2) \\ x > a_3 \; veya \; x < a_1 \; ise \; 0 \end{cases}$
	**Yamuk Üyelik Fonksiyonu** Yamuk üyelik fonksiyonu a_1, a_2 , a_3 ve a_4 olmak üzere dört parametre ile tanımlanır. $\mu_A(x;a_1, a_2, a_3, a_4)=\begin{cases} a_1 \le x \le a_2 \; ise (x-a_1)/(a_2-a_1) \\ a_2 \le x \le a_3 \; ise \; 1 \\ a_3 \le x \le a_4 \; ise \frac{a_4-x}{a_4-a_3} \\ x > a_4 \; veya \; x < \; a_1 \; ise \; 0 \end{cases}$
	**Gaussian Üyelik Fonksiyonu** Gaussian üyelik fonksiyonu m ve σ parametreleri ile tanımlanır. Bu fonksiyonda m fonksiyon merkezini ve σ da genişliği ifade eder. σ değerini değiştirerek, fonksiyonun biçimini değiştirebiliriz. σ küçük olursa fonksiyon ince büyük olursa daha yayvan olacaktır. $\mu_A(x;m,\sigma) = \left\{ exp\left\{ \frac{-(x-m)^2}{2\sigma^2} \right\} \right.$
	**Çan Üyelik Fonksiyonu** Çan üyelik fonksiyonu a1, a2 ve a3 olmak üzere üç parametre ile tanımlanır. $\mu_A(x;a_1, a_2, a_3) = \left\{ \frac{1}{1+\left\lvert \frac{x-a_3}{a1} \right\rvert^{a2}} \right\}$
	**Sigmoidal Üyelik Fonksiyonu** Sigmoidal üyelik fonksiyonu a1 ve a2 parametreleri ile tanımlanır. $\mu_A(x;a_1, a_2) = \left\{ \frac{1}{1+e^{-a1(x-a2)}} \right\}$

3.2.3.3. Üyelik Fonksiyonunun Kısımları

Bir bulanık alt kümede üyelik derecesi 1'e eşit olan elemanlara **öz,** bir alt kümenin tüm elemanlarını içeren aralığa **dayanak** ve üyelik dereceleri 1 ve 0'a eşit olmayanların oluşturduğu kısımlara ise üyelik fonksiyonunun sınırları veya geçiş bölgeleri denir. Genel olarak tüm üyelik fonksiyonlarında biri sağda ve biri solda olmak üzere iki geçiş bölgesi bulunmaktadır. Üyelik fonksiyonunun kısımları aşağıda gösterilmiştir.

$$\mu_A(x) = 1 \longrightarrow \textbf{öz}$$

$$\mu_A(x) > 0 \longrightarrow \textbf{dayanak}$$

$$0 < \mu_A(x) < 1 \longrightarrow \textbf{sınırlar}$$

$$\textbf{Dayanak (A)} = \{ x \in \boldsymbol{E} \mid \boldsymbol{\mu_A(x) > 0} \}$$

Üyelik fonksiyonu aynı zamanda normallik ve dışbükey özelliğine sahip olmalıdır. Normal bulanık küme en az bir tane elemanının derecesi 1'e eşit olmalıdır. Dışbükey bulanık küme ise üyelik fonksiyonunun sürekli artan, sürekli azalan veya üçgen gibi olmasıdır. Bir kümedeki herhangi iki noktayı birleştiren çizgideki her nokta bu kümenin elemanı ise küme dışbükeydir.

Bir üyelik fonksiyonu x=c noktası için simetrik ise bulanık küme simetrik olarak tanımlanır. Aşağıda simetrik olma özelliği gösterilmektedir.

$\forall x \in E \; için;$

$$\mu_A(x + c) = \mu_A(c - x) \tag{3.5}$$

Geçiş noktası ise üyelik derecesi 0.5'e eşit olma durumudur.

Öğe değerleri ile üyelik dereceleri arasındaki bire bir karşılıklılığa bulanık tekillik denir [20].

3.2.3.4. Bulanık Mantık İşlemcileri

Bulanık mantığın doğruluk tabloları ve çıkarım kuralları belirsizlik içerir. Doğruluk dereceleri doğru ve yanlışa yüklenen anlamlara olduğu kadar, bu anlamları güçlendirmek ya da zayıflatmakta kullanılan niceleyicilere de bağlıdır.

Bulanık deyimlerde ve kümlerde, değilleme (~, ¬), tümel evetleme (ve, ∧) ve tikel evetleme (veya, ∨) ve içerme gibi çeşitli işlemciler kullanmaktadır. Literatürde bu işlemcileri açıklayan farklı tanımlar bulunmaktadır [1,20]. Aşağıda temel bulanık mantık işlemcileri açıklanmıştır.

Değilleme, Bulanık Tümleme (Lukasiewicz)

$$\sim a = 1-a \quad , \quad \neg\mu_a(x) = 1-\mu_a(x) \tag{3.6}$$

Tümel Evetleme, Bulanık Kesişim (Lukasiewicz)

$$a \wedge b = EK(a,b) \quad , \quad I[\mu_A(x),\mu_B(y)] = EK[\mu_A(x), \mu_B(x)] \tag{3.7}$$

Tikel Evetleme, Bulanık Birleşim (Lukasiewicz)

$$a \vee b = EB(a,b) \quad , \quad U[\mu_A(x),\mu_B(y)] = EB[\mu_A(x), \mu_B(x)] \tag{3.8}$$

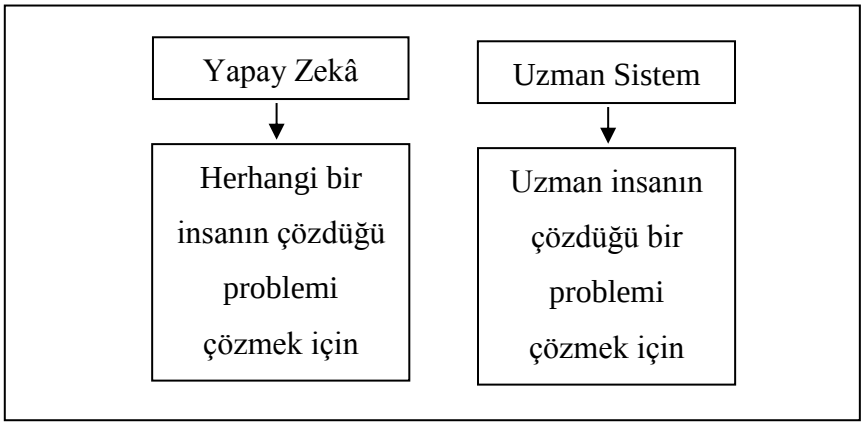
İçerme :

$$a \Longrightarrow b = EK(1,1+b-a) \tag{3.9}$$

3.3. UZMAN SİSTEMLER

Uzman sistem yapay zekânın bir uygulamasıdır. Belirli bir problem kümesi için bir uzman gibi davranan programlara Uzman Sistem (US) denir. Diğer yapay zekâ sistemleri ile uzman sistem arasındaki en büyük fark diğer yapay zekâ sistemleri

herhangi bir insanın çözebileceği bir problemi çözmeyi hedeflerken uzman sistem ise uzman bir kişinin çözebileceği bir problemi çözmeyi hedeflemektedir [18].



Şekil 3.1. Yapay zekâ ve uzman sistemlerin amaçları [18].

Veri işlemeden bilgi işlemeye bir geçiştir. Yani geleneksel programlama ile aralarındaki fark Şekil 3.2.'deki gibi gösterilebilir.

Geleneksel Programlar = Algoritmalar + Veritabanı
Uzman Sistem = Çıkarım Mekanizması + Bilgi Tabanı

Şekil 3.2. Geleneksel programlar ve uzman Sistem.

Uzman Sistemin tasarımı işlemi genellikle "bilgi mühendisliği" olarak adlandırılır.

Bir uzman sistemin tasarlanması sürecinde;

1. Sistemin çözeceği problemin belirlenmesi,
2. Gereken uzmanların bulunması ve ondan bilginin elde edilmesi,
3. Sistemin analizi ve tasarımı,
4. Bir prototipinin oluşturulması,
5. Uygulama

adımları gerçekleştirilir [18].

Bir uzman sistemin 4 temel elemanı vardır [21].

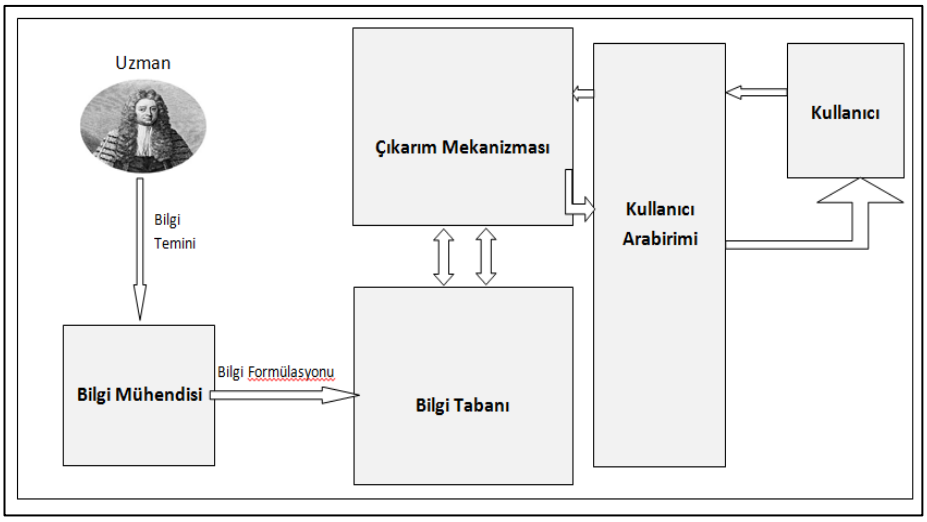
1. Bilginin temin edilmesi: Uzman sistemin uzmanlık alanı ile ilgili bilgilerin toplanması, derlenmesi ve bilgisayarın anlayacağı şekle dönüştürülmesi aşamalarını kapsar.
2. Bilgi tabanı: Toplanan bilgilerin saklandığı yerdir. Bilgiler genellikle kurallar(if.. else..if..), bilgi çatıları(çerçeveleri), bilgi sınıfları ve prosedürlerden oluşur.
3. Çıkarım mekanizması: Bilgi tabanında bulunan bilgileri arayan, filtreleyen, yorumlayan ve sonuçlar çıkaran mekanizmadır. Sonuç üretir. Genel olarak *"İleri Doğru Zincirleme", "Geri Doğru Zincirleme"* olmak üzere iki türlü çıkarım vardır.

   İleri Doğru Zincirleme: İlgili problem hakkındaki gerçeklerden hareket edilerek sonuca gidilir.

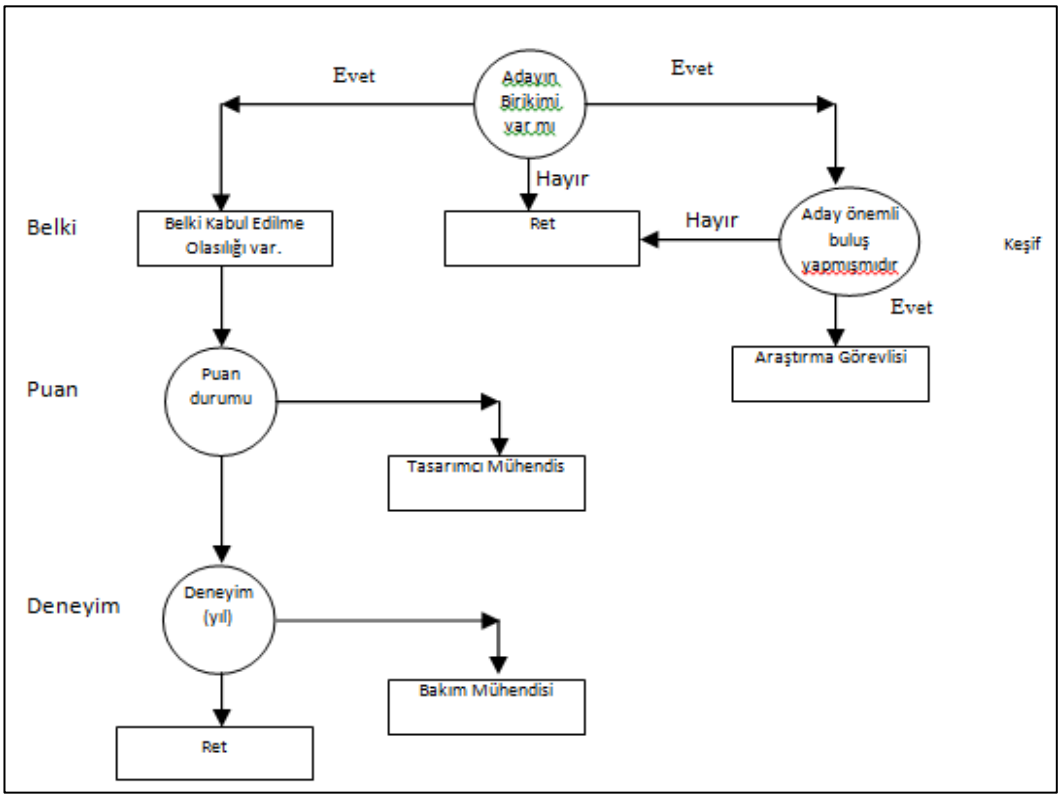
   Geri Doğru Zincirleme: Bu durumda ise bir sonuç ele alınarak o sonucu destekleyen gerçekler var mıdır? Sorusuna cevap aranır.

4. Kullanıcı arabirimi: Kullanıcı ile uzman sistemin iletişimini sağlar. Üretilen sonuçların nasıl üretildiğini ve niçin o sonuçlara varıldığını açıklar.

Uzman sistem elemanları ve bilgi akışı Şekil 3.3.'te gösterilmiştir. Şekil 3.4.'de ise kural tabanlı bir uzman sistem örneği verilmiştir.



Şekil 3.3. Bir uzman sistem elemanları ve bilgi akışı.



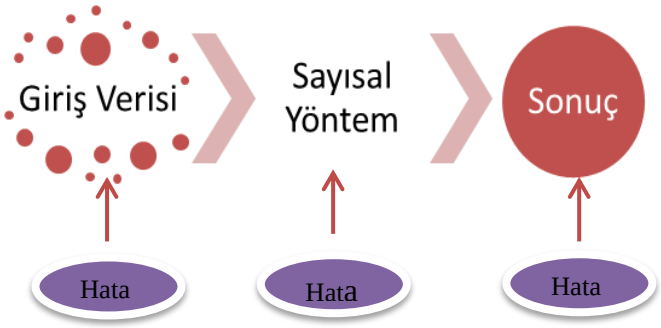
Şekil 3.4. Kural tabanlı bir uzman sistem örneği.

BÖLÜM 4

HATA KAYNAKLARINI EN AZA İNDİRME VE UYGUN İNTERPOLASYON TEKNİĞİNİN SEÇİMİ

Bu bölümde hata kavramı üzerinde durularak daha çok programcıların hassas matematiksel işlemleri bilgisayara yaptırırken karşılaştıkları hata kaynaklarını azaltmaya yönelik bir çalışma yapılmıştır. Ayrıca dijital kalem ile veri girişi esnasında aşırı hızlı el hareketleri veya bir takım donanımsal sebeplerden yakalanamayan noktaları doldurmak için hangi interpolasyon tekniğinin daha etkili sonuç verdiği konusunda yaptığım araştırmamdaki bulgular gösterilmektedir.

Bir denklem veya problemin çözümünde kullanılan sayısal yöntem belli bir giriş verisini işleme tabi tutarak çözüme ulaşır. Sayısal yöntemler, analitik çözümlerden farklı olarak sayıları kullanarak işlem yapar ve belli bir hata payı içerir. Giriş verisi belirli deneyler, gözlemler veya hesaplamalar sonucu bulunuyorsa işlemlerin hassasiyetine göre belli bir hata payı içerecektir. Giriş verisinin ve kullanılan yöntemin hata içermesi, bulunacak sonuçların da belli bir hata içereceğini, yani sonuçların kesin değil yaklaşık değerler olacağını göstermektedir.



Şekil 4.1. Sayısal çözümün blok şeması ve hata etkileşimi [23].

Hatalar, sonuç üzerinde ve yöntemin geçerliliği üzerinde oldukça etkilidir. Sonuçların hata içermesinden ziyade bu hatanın kabul edilebilir tolerans değerleri sınırları içerisinde olması önemlidir.

Hata esas olarak gerçek değer ile hesaplanan yaklaşık değer arasındaki farktır. Bu şekilde tanımladığımız hata iki şekilde ifade edilmektedir [22].

1. Mutlak Hata (Absolute Error) Gerçek değer (x_r) ile hesaplama sonucu bulunan yaklaşık değer (x_n) arasındaki farka

$$\boldsymbol{e_n = x_r - x_n} \tag{4.1}$$

mutlak hata denilmektedir.

2. Bağıl (izafi) Hata (Relative Error) Daha anlamlı olması açısından hatanın mutlak olarak değil gerçek değere göre büyüklüğünün bilinmesi gerekir. Mutlak hatanın gerçek değere oranı bağıl hatayı vermektedir.

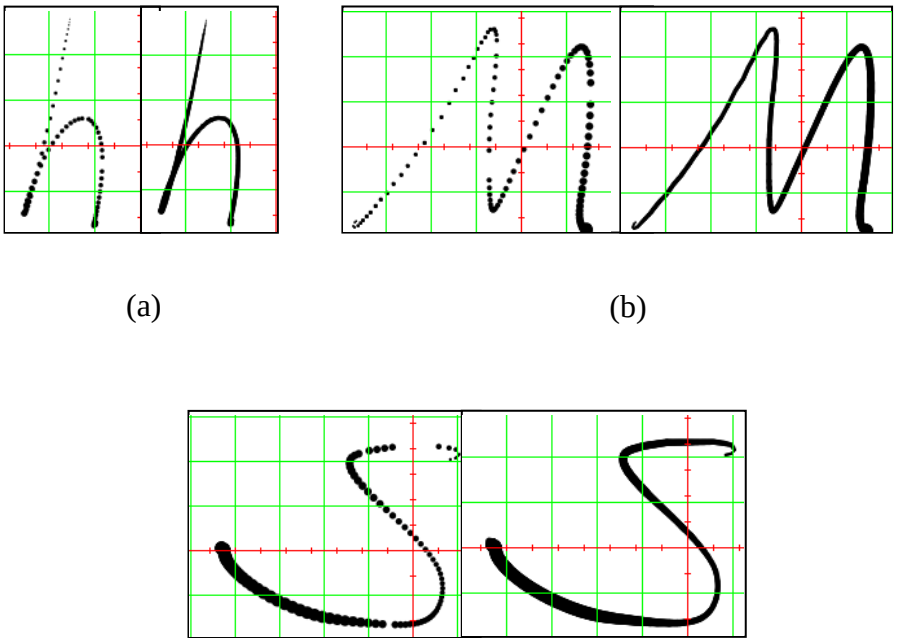
$$\boldsymbol{e_b = \frac{e_n}{x_r}} \tag{4.2}$$

4.1. HATA KAYNAKLARI VE ÇÖZÜM YOLLARI

4.1.1. Giriş Verisi Hatası

Ölçülen veri ile gerçek veri arasındaki farka giriş verisindeki hata denmektedir. Ölçü aletinin doğruluk ve hassasiyeti bilinmeli, gerekli tedbirler alınmalıdır.

Giriş verisi hatasını en aza indirmek için Bölüm 2'de bahsedilen Yumuşatma ve Nokta Azaltma teknikleri kullanılmıştır. Deneylerimizde dijital tablet aracı ile hızlı el yazımı esnasında çok fazla veri noktasının kaçırıldığı gözlemlenmiş, doğal yazıma en uygun noktaların bulunabilmesi için en uygun interpolasyon tekniğinin kullanımının sistemin başarısında çok önemli olduğu tespit edilmiştir. Şekil 4.2'de ham veri ve interoplasyon işlemine sokulan işlenmiş veri gösterilmektedir.



(a) (b)

(c)

Şekil 4.2. Ham veri ve interpolasyon uygulanmış veri, a) h harfi için interpolasyon uygulaması, b) m harfi için interpolasyon uygulaması, c) s harfi için interpolasyon uygulaması

Bu kısımda çeşitli fonksiyonlarla çizilen gerçek eğriler ile bunlardan nokta eksiltmeleri ile oluşan eksik eğrileri, farklı interpolasyon teknikleri ile doldurulmuş; bağıl ve mutlak hata araştırması yapılarak en uygun interpolasyon tekniği irdelenmiştir. Çizim esnasında daha önce belirlenen sonuçlara göre en doğru interpolasyon tekniğini seçen basit bir algoritma geliştirilmiştir.

4.1.1.1. İnterpolasyon

Bir fonksiyonun x_i(i=0, 1 ,2…n) noktalarında bilinen f_i(i=0, 1, 2…n) değerlerinden hareketle herhangi bir x ara noktasında bilinmeyen f(x) bulunması anlamına gelen interpolasyon teknikleri aynı zamanda sayısal türev ve integrasyon, adi ve kısmı türevli diferansiyel denklemlerin sayısal çözümü gibi başka sayısal yöntemlerin de temelini teşkil eder.

İnterpolasyon polinomları genellikle mevcut (x_1, f_i) veri noktalarına eğri veya eğriler uydurulması yolu ile uygulanmaktadır. Bu amaçla kullanılan fonksiyonlara interpolasyon fonksiyonları denir [23].

İnterpolasyon fonksiyonları olarak çoğu zaman çeşitli mertebeden polinomlar kullanılmaktadır. Ancak bazı hallerde logaritmik, ekponansiyel, hiperbolik gibi daha özel fonksiyonlar ile periyodik veri değerleri için trigonometrik fonksiyonlar da kullanılmaktadır.

Veri noktaları eşit aralıklı olarak dağılmışsa sonlu farklı esaslı interpolasyon yöntemleri, eşit aralıklı değilse Lineer interpolasyon, İkinci Dereceden (Kuadratik) İnterpolasyon, Newton - Gregory Polinomları, Lagrange İnterpolasyon, Parça Parça (Spline) İnterpolasyon vb. yöntemler uygulanmaktadır.

Lineer İnterpolasyon

Bir f(x) fonksiyonunun x_k, x_{k+1} gibi farklı iki noktadaki değeri biliniyorsa, bu durumda $x \in [x_k, x_{k+1}]$ noktasındaki değerinin hesaplanması bu iki noktadan geçen birinci dereceden bir polinom yardımıyla yapılabilmektedir [24]. Lineer interpolasyon aşağıdaki eşitlik ile ifade edilmektedir:

$$y = f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} (x - x_k) \tag{4.3}$$

İkinci Dereceden (Kuadratik) İnterpolasyon

f(x) fonksiyonunun (x_0, y_0), (x_1, y_1) ve (x_2, y_2) gibi 3 noktası belli ise [x_0, x_2] aralığındaki herhangi bir x noktasındaki fonksiyonun değeri, bu üç noktadan geçen parabole eşdeğer yaklaşım polinomu seçilerek bulunmaktadır [24]. Kuadratik interpolasyon aşağıdaki eşitlik ile ifade edilmektedir.

$$\boldsymbol{p(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + \left(\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}\right)\frac{(x - x_0)(x - x_1)}{(x_2 - x_0)}} \tag{4.4}$$

Newton - Gregory Polinomları

İkinci dereceden interpolasyon hesaplarken kullanılan yöntemde mertebe arttırılarak, daha yüksek mertebeli polinomlar elde edilebilmektedir.

Lagrange İnterpolasyonu

Eşit aralıklı olmayan n adet (x_i,y_i) noktalarından (i=1,2,3, ... n) geçen (n-1). Dereceden Langrange interpolasyon polinomu;

$$yp(x) = \frac{(x-x_2).(x-x_3).....(x-x_n)}{(x_1-x_2).(x_1-x_3).....(x_1-x_n)} y_1 + \frac{(x-x_1).(x-x_3).....(x-x_n)}{(x_2-x_1).(x_2-x_3).....(x_2-x_n)} y_2 \ldots + \frac{(x-x_1).(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1).(x_n-x_2)\ldots(x_n-x_{n-1})} y_n \quad (4.5)$$

Formundadır [23]. Bu genel ifade daha kısa olarak;

$$yp(x) = \sum_{i=1}^{n} L_i(x) y_i \quad (4.6)$$

şeklinde yazılabilir. Burada

$$Li(x) = \frac{(x-x_1).(x-x_2).....(x-x_{i-1}).(x-x_{i+1}).(x-x_n)}{(x_i-x_1).(x_i-x_2).....(x_i-x_{i-1})(x_i-x_{i+1})(x_i-x_n)} \quad (4.7)$$

$$Li(x) = \prod_{\substack{j=1 \\ j\neq 1}}^{n} \frac{x-x_j}{x_i-x_j} \quad (4.8)$$

çarpım terimini göstermektedir.

Parça Parça Spline İnterpolasyon

İnterpolasyon polinomlarının derecesi arttıkça, uzun ve karmaşık hale gelmekte, noktalar arasında gerçek eğriden büyük oranda sapmalar olmaktadır. Bunun nedenleri, bulunan polinomun noktalarda fonksiyonun eğimleri hakkında bilgi

içermemesi, derece arttıkça kötü şartlanmış denklem sistemlerinin oluşması ve yuvarlama hatalarıdır.

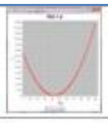
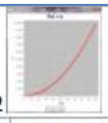
Gerçek fonksiyona en uygun, yuvarlama hataları az, bütün noktaları kullanılan ama derecesi düşük interpolasyon polinomları kullanılması daha uygundur. Bu şekilde noktalar gruplandırılarak interpolasyon tekniklerinin uygulanmasına Spline İnterpolasyon denir.

Spline interpolasyonu çeşitli 2 boyutlu ve 3 boyutlu bilgisayar animasyonlarında sıklıkla kullanılmaktadır [25] .

4.1.1.2. Uygulama İçin En İyi İnterpolasyon Tekniğinin Seçimi

Giriş verisi hatasını en aza indirmek için Lineer İnterpolasyon, Kuadratik Interpolasyon, Newton-Gregory İlerleme Polinomu, Lagrange Lineer, çeşitli derecede Lagrenge Interpolasyon teknikleri kullanılarak en iyi sonuç alınmaya çalışılmıştır. İnterpolasyon teknikleri Spline (parça, parça) Interpolasyon şeklinde uygulanmıştır.

Seçilen 5 farklı fonksiyon ile oluşturulan şekillerden nokta eksiltme ile oluşturulan yeni şekil interpolasyon tekniklerine tabi tutulmuş mutlak hata ve bağıl hata analizi ile Şekil 4.3'de gösterilen sonuçlar elde edilmiştir.

İnterpolasyon Teknikleri	Fonksiyon 1 		Fonksiyon 2 		Fonksiyon 3 	
	Mutlak Hata *1.000.000	Bağıl Hata *1.000.000	Mutlak Hata *1.000.000	Bağıl Hata *1.000.000	Mutlak Hata *1.000.000	Bağıl Hata *1.000.000
Lineer	56266654,968262	1417812,798599	262133300,781250	177846,595239	2933303,833008	20922,388199
Quadratik	56266654,968262	1417812,798599	262133300,781250	177846,595239	2933303,833008	20922,388199
Lagrange Lineer	24000000,000000	872151,597375	12000000,000000	54851,475541	0,000000	0,000000
Lagrange 2	61,035156	0,062729	0,000000	0,000000	7,629395	0,070643
Lagrange 3	13,351440	0,348902	640,869141	0,376997	76,293945	0,545179
Lagrange 4	152,587891	0,216495	263,214111	0,329790	7,629395	0,070643
Lagrange 5	1697,063446	3,688979	3725,051880	2,202660	267,028809	2,051762
Lagrange 6	3812,313080	6,931371	6574,630737	6,753831	572,204590	4,746828
Oto Seçim	61,035156	0,062729	0,000000	0,000000	0,000000	0,000000

İnterpolasyon Teknikleri	Fonksiyon 4 		Fonksiyon 5 	
	Mutlak Hata *1.000.000	Bağıl Hata *1.000.000	Mutlak Hata *1.000.000	Bağıl Hata *1.000.000
Lineer	2933322,906494	65393,987739	0,000000	0,000000
Quadratik	2933322,906494	65393,987739	0,000000	0,000000
Lagrange Lineer	0,000000	0,000000	0,000000	0,000000
Lagrange 2	3,814697	0,071975	0,000000	0,000000
Lagrange 3	22,888184	0,523354	2,861023	0,572205
Lagrange 4	3,814697	0,077851	0,000000	0,000000
Lagrange 5	85,353851	2,560209	7,629395	1,525879
Lagrange 6	174,999237	5,967793	20,980835	4,196167
Oto Seçim	0,000000	0,000000	0,000000	0,000000

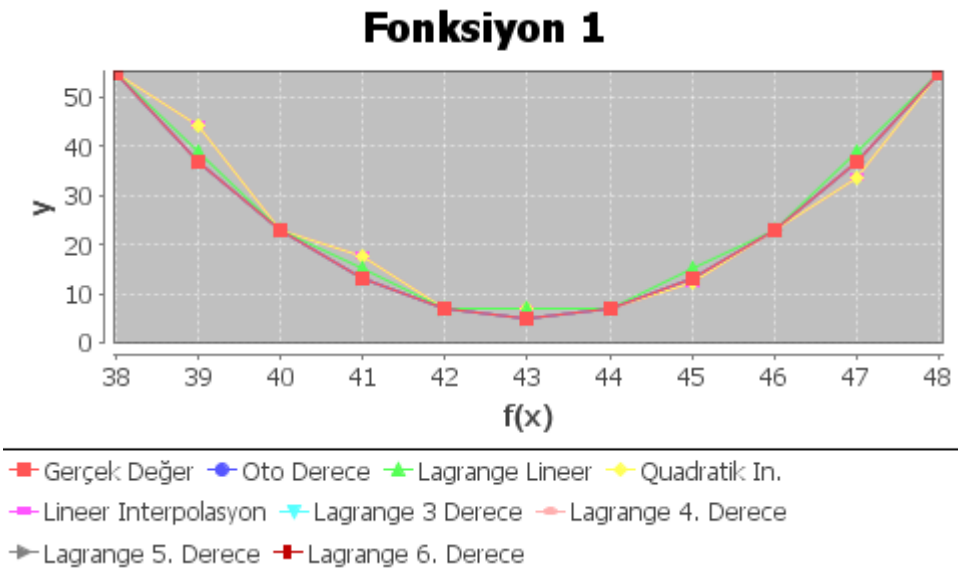
Şekil 4.3. İnterpolasyon hesaplaması sonuçları.

Elde edilen sonuçlara göre eşit eğimli noktalar arasında ara değerler bulunurken Lagrenge Lineer interpolasyon hesabı daha iyi sonuçlar vermekte, farklı eğimli noktalar arasındaki ara değerlerin bulunmasında ise Lagrange 2. Derece interpolasyon hesabı daha iyi sonuçlar vermektedir.

Lagrange Interpolasyon hesabında polinom derecesi arttıkça interpolasyon eğrilerinin gerçek eğriden saptıkları görülmüş bu yüzden 6. Dereceden daha yüksek derecelere çıkılmamıştır.

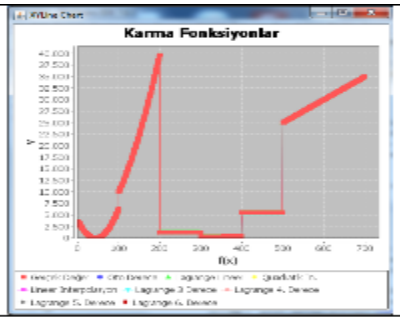
Kuadratik İnterpolasyon ve Lineer İnterpolasyan uygulamasında bağıl ve mutlak hata oranlarının çok çıkması nedeni ile kullanılamayacağı gözlenmiş, aynı teknikle hesaplanan Newton Gregory İlerleme Polinomu interpolasyonu tekniğinden vazgeçilmiştir.

Her zaman en iyi sonuca ulaşmak için eğime göre Lagrange Lineer ve Lagrange 2. Derece interpolasyonları arasında seçim yapan basit bir algoritma oluşturulmuştur. Şekil 4.4.'te de görüleceği üzere Oto Seçim ile yapılan hesaplar neredeyse gerçek değerle çakışmaktadır. (Gerçek eğrinin altında görünmüyor.)



Şekil 4.4. Otomatik seçim hesabı ve diğer interpolasyon teknikleri sonuçları.

Test için oluşturulan 5 fonksiyonun birlikte çalıştırılması ile oluşturulan grafik interpolasyon teknikleri ve oto seçim algoritması ile doldurularak Şekil 4.4'teki sonuçlar elde edilmiştir.

	Tüm Fonksiyonlar Karma	
	Toplam Mutlak Hata	Toplam Bağıl Hata
İnterpolasyon Teknikleri	Ölçek : * 1.000.000	Ölçek : * 1.000.000
Lineer İnterpolasyon	1057596549,987793	1474386,421007
Quadratik (2. Derece) İnterpolasyon	1057596549,987793	1474386,421007
Lagrange Lineer İnterpolasyon	36000000,000000	872835,950196
Lagrange 2 Derece İnterpolasyon	2990,722656	0,214184
Lagrange 3 Derece İnterpolasyon	15699,386597	2,666426
Lagrange 4 Derece İnterpolasyon	2410,888672	0,764368
Lagrange 5 Derece İnterpolasyon	116076,946259	14,033769
Lagrange 6 Derece İnterpolasyon	283170,223236	32,544848
Otomatik Seçim İnterpolasyon	**1037,597656**	**0,139208**

Şekil 4.5. Karma fonksiyonlar ile interpolasyon hesapları sonuçları.

Yapılan çalışma sonucunda aşağıdaki bulgulara ulaşılmıştır:

1. Lineer, Quadratik ve Newton Gregory İnterpolasyon tekniklerinde mutlak ve bağıl hata hesaplama sonuçları yüksek çıkmaktadır.
2. Eşit Eğimlerde ; Lagrange Lineer interpolasyonu en iyi sonucu vermektedir.
3. Bazı durumlarda Lagrange 4. Derece interpolasyonunda olduğu gibi değişik dereceli Lagrange interpolasyon hesaplamalarında daha düşük hata değerleri ile karşılaşılsa bile bunlar arasında en kullanılabilir olanı her eğimde kararlı bir şekilde hata düzeyini düşük tutabilen sadece Lagrange 2. Derece interpolasyonudur.
4. Bulunan bu bulgular üzerine eğime göre Lagrenge Lineer ve Lagrange 2. Derece interpolasyonları arasında seçim yapabilen basit bir Oto Seçim fonksiyonu oluşturularak giriş verisi hatası en düşük seviyede tutulabilir.

4.1.2. Yuvarlama Hatası

Hesaplarda rakamların hane sayısının sonlu tutulmasından kaynaklanan hataya *yuvarlama hatası(round-off error)* denmektedir [23]. Bilgisayarlı hesaplamalarda art

arda yapılan hesaplamalar oluşan hatayı taşır, bu şekilde hata birikerek büyüyebilmekte ve sonuçta önemli değişikliklere sebebiyet verebilmektedir. Hesaplamalar süresince eğer hata azalıyorsa kullanılan yöntem *"kararlı",* hata büyüyorsa *"kararsız"* dır.

4.1.3. Kesme Hatası

Seriler çok veya sonsuz sayıda terimden oluşur. Serinin belirli bir sayıda terimini alıp diğer terimlerin atılmasından bir miktar hata oluşur. Bu hataya kesme hatası *(truncation error)* denir. Bu hatanın başka bir sebebi de iterasyon sayısının sınırlı tutulmasıdır.

4.1.4. İnsan Hatası

İnsanın kendisinden kaynaklanan hatalardır. Fiziksel veya matematiksel modelin oluşturulmasında, işlem yaparken veya program yazarken yapabileceği hatalardır.

4.1.5. Binary Aritmetiğin Kötü Sonuçları ve Bunları En Aza İndirme

Floating-point hesaplamalarında bilgisayar temelde ikili (binary) olarak çalışmaktadır [26]. Ondalık kesirler "0.1" gibi aslında tam olarak 2'li sistemde tanımlanamaz. Bu "0,00011001100110…" gibi 2'li sistemde tekrarlayan kesir onluk sistemdeki "1/3=0,3333…" gibi bir kesre benzer. Eğer "0.33.." ve "0.66…" gibi iki sayı toplanırsa 1'den ziyade "0.99.." sayısı elde edilir. Oysa 1/3 ile 2/3 toplandığında 1 sayısı elde edilir.

Floating-point hesaplamalarında Java, ANSI/IEEE Std 754-1985 (ANSI/IEEE Std 754-1985, An American National Standard IEEE Standard for Binary Floating-Point Arithmetic) standartlarına göre işlem yapar.

Floating-point hesaplamalarında Java'da bazen 0,1 ile 0.1 toplandığında 0.2 den farklı sonuçlar almak mümkündür. Benzer hesaplamalarda da buna rastlanılmaktadır. İkili sistemde 0,1 gibi bir kesri Java'nın mükemmel olarak yapamaması bir hatadır.

Yeni insan-yaklaşımlı java alt yapısı kullanılarak yazılan bir programlama dili olan Netrexx'in [28] yaratıcısı Mike Cowlishaw, floating-point hatası için bilgisayar donanımı yapıcılarını suçlamıştır. O, bilgisayar chiplerinin insanlar gibi 10'luk sistemde hesaplama yapmasını önerir. 1/3 kesri aynı zamanda ikili sistemde 0.01110011.. gibi bir tekrar edici değer olması sebebiyle Java'da (ve diğer benzer dillerde) 10'luk aritmetik işlemlerde kullanılması çok uygun olmayacaktır.

Bu yapısal hatalardan kurtulmak için aşağıda açıklanan yöntemler denenebilir;

1. Approximate Comparison (Karşılaştırma Yaklaşımı)

   if (f = = 100.00) yerine if(f > 99.995)

2. Equality Testing (Eşitlik Testinde)

   if (Math.abs(value - target) < epsilon)

   veya ;

   if (value >= target - epsilon && value <= target + epsilon)

   Bilinmeyen büyüklükler sunulduğunda, daha yavaş çalışan aşağıdaki kodda kullanılabilir;

   *if(Math.abs(value - target< epsilon *target)*

   veya;

   *if(value >= target * (1 - epsilon) && value <= target * (1+ epsilon))*

3. Loop (Döngü)

   Döngülerde int ve float/double karışımları kullanıma uygundur. Floating-point artırımlar yerine int ve float karışık artırımlar kullanılmalıdır.

   Şöyle ki; *f+=0.001F ;* yerine *f+=i*0.001F;*

4. Double and Float

   Float yerine Double kullanılmalıdır. Bu işlem problemi tam olarak çözmez. Fakat hassasiyet arttırılarak taşmalar bir nebze olsa önlenebilir.

5. Perfect Accurracy (Para Tipleri)

   Tam çözüme ulaşmak için int, long, BigDecimal ve BigInteger kullanılmalıdır. Currency tipleri floating-point para değerlerinin tutmanın en iyi yoludur. Javada böyle bir tip bulunmamaktadır.

6. Binary Formats

   Float-double sayı binary formata çevirip o şekilde eşitlik karşılaştırması yapılabilir. Nitekim Java saklı yordamlarından biri olan Float.compare() fonksiyonunun orijinal tanımı Şekil 4.6'de belirtildiği gibidir;

```
public static int compare(float f1, float f2)
{
    if (f1 < f2)
        return -1;              // Neither val is NaN, thisVal is smaller
    if (f1 > f2)
        return 1;               // Neither val is NaN, thisVal is larger

    int thisBits = Float.floatToIntBits(f1);
    int anotherBits = Float.floatToIntBits(f2);

    return (thisBits == anotherBits ? 0 :  // Values are equal
          (thisBits < anotherBits ? -1 :       // (-0.0, 0.0) or (!NaN, NaN)
           1));                                // (0.0, -0.0) or (NaN, !NaN)
}
```

Şekil 4.6. Java float.compare() fonksiyonu [22].

4.2. MAKİNE EPSİLONUNUN HESAPLANMASI

Makine Epsilonu, bir sayıya eklendiğinde değişim yapabilen en küçük birimdir. (1 + € > 1) şeklinde ifade edilebilir.

Eşit aralıklı olmayan interpolasyon hesabı yaparken aralık indisleri eşit olduğunda yani aynı y=f(x) noktası işaret edildiğinde formül gereği 0 bölme hatası ile karşılaşılabilmektedir.

Şöyle ki;

$$y_i=f(x_i) == y_{i+1}=f(x_{i+1}) \tag{4.9}$$

Lagrange interpolasyon formülünde indis numaralarını bulmak için bir sonraki değer ile eldeki değer arasındaki farkı indis numarası olarak kabul edilmiştir. İki değerin eşit çıkması durumunda indis değeri aynı olacak ve formül gereği '0' bölme hatası ile karşılaşılacaktır. Bunu önlemek ve interpolasyon hesabında ara nokta bulunurken eşit değerli noktaların '0' indis numaralarının etkisini en hassas şekilde almak için Şekil 4.7'da belirtilen makine epsilon değerini bulan algoritma ile makine epsilonu bulunmuş ve indis numarası epsilon değeri kadar arttırılmıştır. Aynı şekilde bu, bizim aritmatik işlemlerimizde hata düzeltme tekniklerinde kullanacağımız epsilon değeridir [22].

```
package numbercruncher.mathutils;

/**
 * Compute the machine epsilon for the float and double types,
 * the largest positive floating-point value that, when added to 1,
 * results in a value equal to 1 due to roundoff.
 */
public final class Epsilon
{
private static final float  floatEpsilon;
private static final double doubleEpsilon;

    static {

      // Loop to compute the float epsilon value.
         float fTemp = 0.5f;
         while (1 + fTemp > 1) fTemp /= 2;
         floatEpsilon = fTemp;

         // Loop to compute the double epsilon value.
         double dTemp = 0.5;
         while (1 + dTemp > 1) dTemp /= 2;
         doubleEpsilon = dTemp;
     };

   /**
    * Return the float epsilon value.
    * @returns the value
    */
public static float floatValue(){return floatEpsilon;}

   /**
    * Return the double epsilon value.
    * @returns the value
    */
public static double doubleValue(){ return doubleEpsilon; }
}
```

Şekil 4.7. Java makine epsilonu değeri hesabı [22].

4.3. ARİTMETİK İŞLEMLERDE HATALARIN ETKİSİ

Hatalar yayılıp büyüyerek sonuçların anlamsız hale gelmesine sebep olabilir. Bu durumu incelemek için, x_r ve y_r gibi iki gerçek sayı ile yapılan basit aritmetik işlemlerde hataların etkisi irdelenecektir. Bu sayıların bilgisayarda gösterimi ile oluşan yuvarlama hataları sırası ile $\boldsymbol{e_x}$ ve $\boldsymbol{e_y}$, sayılarının bilgisayarda gösterimi x ve y denilirse;

$$\boldsymbol{e_x = x_r - x} \quad , \quad \boldsymbol{e_y = y_r - y} \tag{4.10}$$

yazılabilir. Bu iki sayının bilgisayarda toplanması;

$$x+y=(x_r+e_x)+(y_r+e_y) \tag{4.11}$$

sonucunda oluşan hata,

$$e_+=(x_r+y_r)-(x+y)=e_x+e_y \tag{4.12}$$

olacaktır. Benzer şekilde çıkarma işleminde hata,

$$e_-=e_x-e_y \tag{4.13}$$

çarpma işleminde oluşan hata,

$$e_*=ye_x+xe_y+e_xe_y \tag{4.14}$$

ve bölme işleminde hata da;

$$e_+=(ye_x-xe_y)/y(y+e_y) \tag{4.15}$$

olarak elde edilir.

Bu sayıların nasıl etkilediğini basitçe şu şekilde sıralayabiliriz.

1. Bölme işleminde bulunan hata bölen arttıkça azalmaktadır.
2. Birbirine çok yakın olan iki sayının çıkarılması işleminde bağıl (izafi) hata büyük olabilmektedir.

BÖLÜM 5

UYGULAMA

5.1. UYGULAMA GELİŞTİRME ORTAMI

5.1.1. Donanım Ortamı

Aşağıda çizelge 5.1 ve 5.2'de belirtilen donanım özellikleri ile gerçekleştirilen uygulama her donanım ortamında çalışacak şekilde tasarlanmıştır.

Çizelge 5.1. Uygulamanın geliştirildiği ve test edildiği bilgisayar ortamları.

Bilgisayar Donanımı 1	
İşlemci	AMD Athlon (tm) 64 X2 Dual Core Processor 4200+
İşlemci Hızı (GHz)	2.21
Bellek (GB)	2
Bilgisayar Donanımı 2	
İşlemci	Intel (R) Core (tm) 2 Duo CPU T9300
İşlemci Hızı (GHz)	2.5
Bellek (GB)	3

Çizelge 5.2. Uygulamanın geliştirildiği grafik tabletin teknik özellikleri.

Grafik Tablet	
Marka – Model	UC-Logic Lapazz TWH850 Grafik tablet.
Toplam alan (mm)	315 x 226,3
Çalışma alanı genişliği (mm)	203,2 x 127
Basınç hassasiyeti	1024 kademe
Çözünürlük (lpi)	4000
Rapolama hızı (rpm)	200
Yüzey malzemesi	FR-4 cam epoksi
Maksimum kalem algılama yüksekliği (mm)	10
Kalem eğim algılama	Maksimum 60^0 dikey
Doğruluk (mm)	0/ -0,25
Arayüz	USB

5.1.2. Yazılım Geliştirme Ortamı

Uygulama Windows 7 Professionel Service Pack 1 64 bit işletim sistemi üzerinde Java Swing, Cellosoft JTablet, Java Persistence Api, Hibernate ve MySQL teknolojileri kullanılarak NetBeans geliştirme ortamı üzerinde Maven Proje yönetim aracı ile geliştirilmiştir.

Swing Java ile görsel arayüzler oluşturabilmek için tasarlanmış bir teknolojidir. Ücretsiz olması nedeniyle özelikle A.B.D gibi ülkelerde en çok kullanılan arayüz geliştirme teknolojilerinden bir tanesidir [28]. Swing tamamen Java dili ile yazılmış ve tüm platformlarda çok yüksek derecede aynı görüntüyü veren bir bileşenler kütüphanesidir. Swing paketi javax.swing dizini içinde yer almaktadır [29].

Cellosoft JTablet kütüphanesi, tablet giriş verisine erişmek için geliştirilmiş olay tabanlı ve açık kaynak kodlu bir arayüzdür. Windows, Mac OS X desteği vardır. Linux desteği çok yakında hizmete girecektir [30].

Nesne İlişkisel Haritalama (Object Relational Mapping) kavramı 1970'li yıllardan itibaren üzerinde akademik olarak çalışılan ve 1990'ların sonlarından itibaren uygulamaya geçirilmiş ve günümüzde büyük çaplı projelerde sıklıkla kullanılan bir kavramdır. ORM nesneler ile veritabanındaki tabloların eşleştirilmesidir. Hemen hemen bütün programlama dilleri ORM çözümleri sunar. Java bu çözümlerin uygulandığı ilk güçlü dildir [28]. Aşağıda ORM araçlarının bazıları listelenmiştir.

1. Java
   1.2. Hibernate
   1.3. EclipseLink
   1.4. Toplink
   1.5. JPA (Java Persistence Api)
2. .Net Dilleri (C,# VB.Net, J#)
   2.2. NHibernate
   2.3. ADO.Net Entities
3. PHP
   3.2. Doctrine
   3.3. Propel

Java Persistence Api, genellikle JPA olarak anılan, Java Standart Sürüm ve Java Kurumsal Sürümleri kullanılarak geliştirilen uygulamalarda ilişkisel verileri yönetmek için geliştirilmiş Java Programlama Dili Çerçevesidir [31]. JPA ilişkisel veritabanları ile programsal nesneler arasındaki ilişkilerin tanımlandığı güçlü bir araçtır [28].

JPA, Hibernate , TopLink, JDO gibi kalıcılık teknolojilerinin en iyi yönleri ve son zamanlardaki EJB(Enterprise Java Bean) kap (Container) yönetimine dayalı kalıcılık üzerine kurulmuştur. JPA var olan tek bir yapı iskeletine dayanmaz aksine var olan pek çok güncel yapı iskeletinin desteklediği fikirleri içine alır, beraber çalışır ve geliştirir. Bu sayede uygulama geliştiriciler artık nesne ilişkisel modeli gerçekleştirmede uyumsuz ve standart olmayan kalıcılık modelleriyle yüzleşmek durumunda kalmamaktadır. Bununla beraber Java Persistence API daha çok uygulama geliştiriciye standart bir kalıcılık aracına sahip olma şansını vererek hem

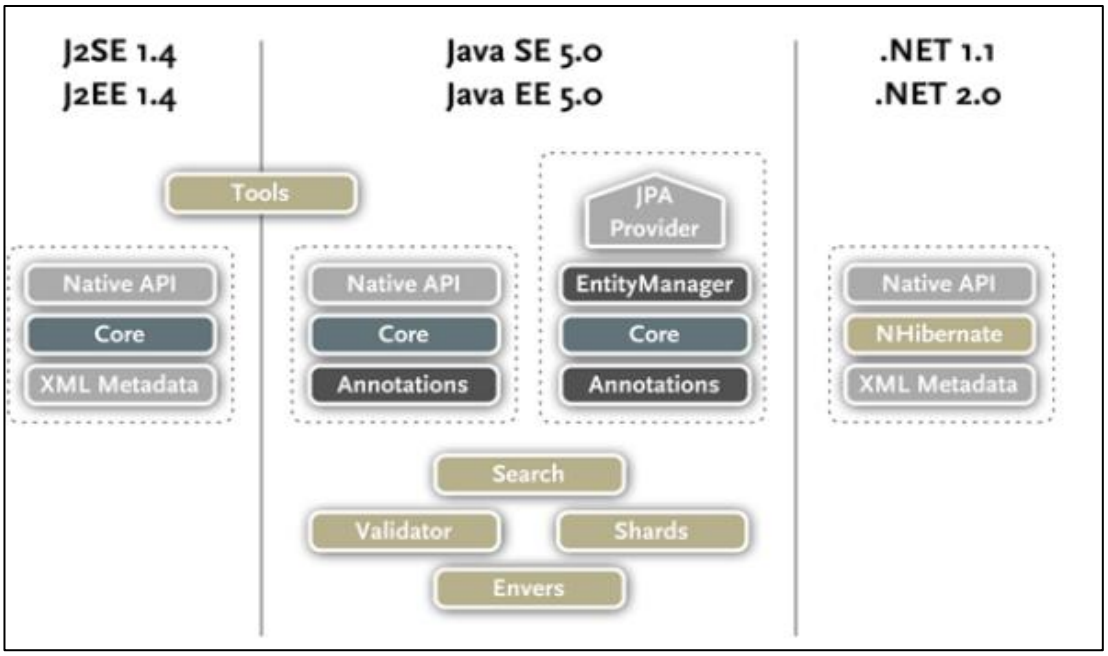
Java SE hem de Java EE içinde kullanılabilmektedir [32]. JPA nın içinde JPQL (Java Persistence Query Language) denilen SQL benzeri çok güçlü bir sorgulama dili bulunmaktadır. Bu sayede veritabanından bağımsız bir şekilde sorgular oluşturulabilmektedir. JPQL doğrudan ilişkisel veritabanı ile değil entity denilen varlık nesneleri üzerinde işlemlerini yürütür. Varlık nesneleri POJO denilen yani getter ve setter metotları bulunan basit Java nesneleridir. Şekil 5.1'de geliştirilen uygulamadaki User veritabanı tablosunu temsil eden POJO tarzındaki User varlık sınıfı gösterilmektedir.

```
@Entity
@Table(name = "user", catalog = "tablet", schema = "", uniqueConstraints
    @UniqueConstraint(columnNames = {"username"})})
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "User.findAll", query = "SELECT u FROM User u"),
    @NamedQuery(name = "User.findById", query = "SELECT u FROM User u WH
    @NamedQuery(name = "User.findByUsername", query = "SELECT u FROM Use
    @NamedQuery(name = "User.findByAd", query = "SELECT u FROM User u WH
    @NamedQuery(name = "User.findBySoyad", query = "SELECT u FROM User u
    @NamedQuery(name = "User.findByTcno", query = "SELECT u FROM User u
    @NamedQuery(name = "User.findByYetki", query = "SELECT u FROM User u
    @NamedQuery(name = "User.findByUserParola", query = "SELECT u FROM U
public class User implements Serializable {
    @OneToMany(mappedBy = "uzmanID")
    private Collection<Grupparametreleri> grupparametreleriCollection;
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id", nullable = false)
    private Integer id;
    @Column(name = "username", length = 11)
    private String username;
    @Column(name = "ad", length = 50)
    private String ad;
    @Column(name = "soyad", length = 100)
```

Şekil 5.1. User varlık nesnesinden bir görünüş.

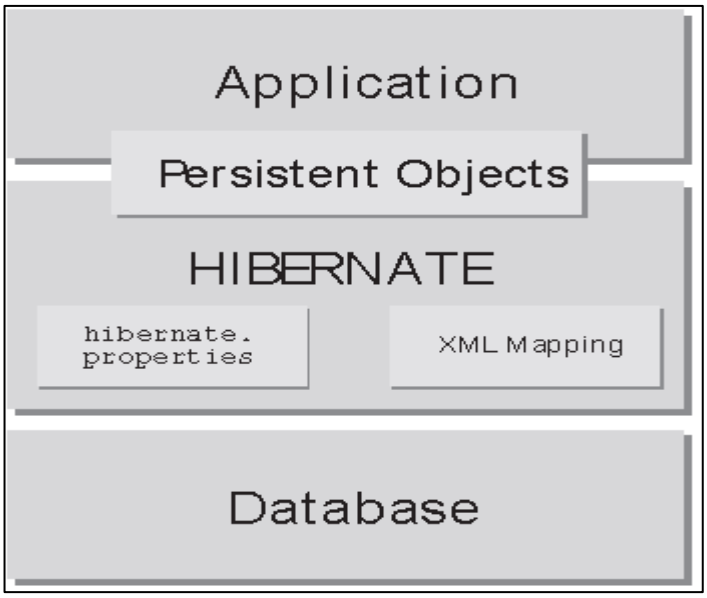
Hibernate, Java platformunda yazılmış bir ORM aracıdır. Hibernate gibi ORM araçlarının en önemli faydası, kod yazımını kısaltmak veya kolaylaştırmaktan öte, yazılımın bakımını kolaylaştırmasıdır [33]. Hibernate bedava bir yazılımdır. GNU lisansı ile dağıtılmaktadır. Red Hat tarafından geliştirilmektedir. Java Virtual Machine üzerinde çalıştığı için platformdan bağımsız tüm işletim sistemlerinde ve 12

databese üzerinde çalışabilmektedir [34]. ORM araçları sayesinde kod yazarı için hangi veritabanının kullanıldığının önemi kalmamıştır. 2 satır kod ile istenilen veritabanına uygulama taşınabilmektedir. Hibernate'nin gelişim evreleri Şekil 5.2'de gösterilmektedir.



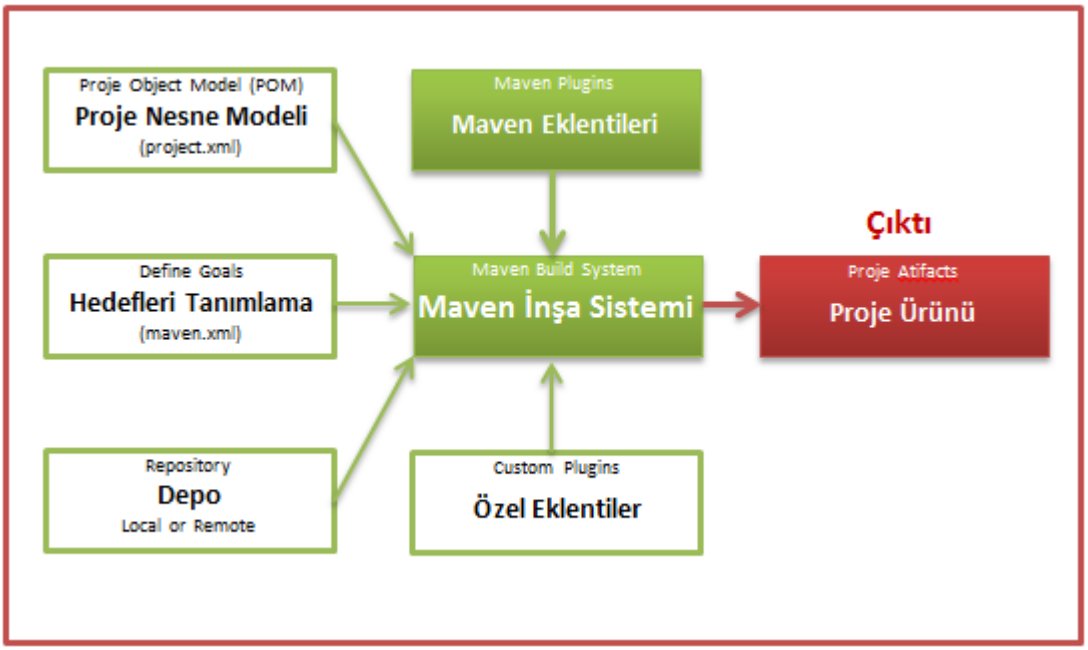
Şekil 5.2. Hibernate'nin gelişim tarihçesi [35].

Şekil 5.3 'de Hibernate mimarisinin yüksek katmanlı görümü gösterilmektedir.



Şekil 5.3. Hibernate mimarisinin yüksek katmanlı görünümü [36].

Maven**,** bir proje yönetimi ve anlama aracıdır. Bir Proje Nesne Modeli (Project object model-POM) kavramına dayanarak, Maven bilgi merkezinin bir parçası olarak, proje oluşturma, raporlama ve dokümantasyonu yönetebilmektedir [37]. Şekil 5.4 'de Maven'nin yapı taşları resmedilmektedir.



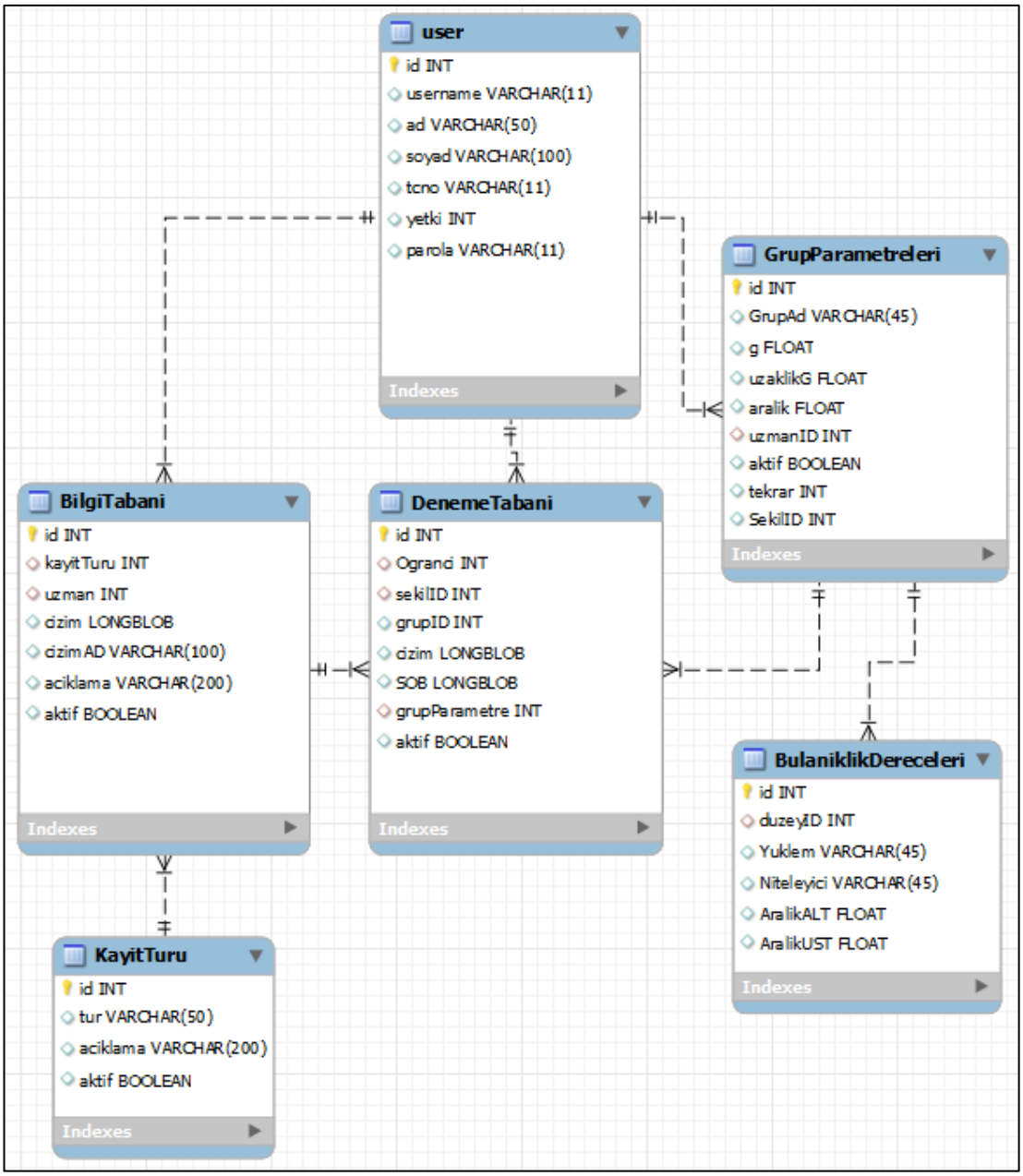
Şekil 5.4. Maven'nin yapısı.

Proje Nesne Modeli (POM), tüm ayarlarının yapıldığı kısımdır. Genel olarak bu ayarlar, projenin ismi, sahibi ve onun bağlantılı olduğu diğer projelerdir.

Eklentiler (Plugins), Maven'nin birçok işlevsel eklentileri bulunmaktadır. Bu eklentiler, derleme, test , kaynak kontrol yönetimi, web sunucuda çalışma, Eclipse proje dosyası oluşturmak için vs.

Depo (Repository), projelerde kullanılan birçok kaynak paket Maven depolarında kolaylıkla indirilebilir. Ayrıca harici bir depodan da proje kod paketleri çekilebilir.

Hedef Tanımlama, ile Maven yaşam döngüsü ihtiyaçlar doğrultusunda yeniden tanımlanabilir. Örnek olarak, derleme, test etme, yayınlama gibi.[38]
MySQL veri tabanı açık kaynak kodlu dünyanın en popüler veritabanıdır [39]. Tamamen ücretsiz olan bu veritabanının ücretli olan Enterprise sürümüde mevcuttur. Uygulama için geliştirilen veritabanı diyagramı Şekil 5.5 'te gösterilmektedir.



Şekil 5.5. Uygulama için geliştirilen veritabanı tabloları diyagramı.

5.2. UYGULAMA SÜRECİ

Uygulama, öğretilecek figürün bilgisayara tanıtımı ve figürün kullanıcıya öğretimi olmak üzere iki ana bölümde geliştirilmiştir.

Programın çalışma mantığı, öncelikle Yönetici kullanıcısının öğretilecek figürü bilgisayara tanıtması için bir uzmana, Uzman Yetkili kullanıcı hesabı açması ile başlar. Uzman kendisine verilen kullanıcı hesabı ile öğreteceği figürleri bilgisayara girer. Tecrübelerine göre tanıma düzeyi ve düzey ile ilintili olan çıkarım mekanizmasının bulanıklık derecelerine uygun sözel ifadeleri tanımlar. Figürün öğretileceği öğrenci ise kendisine Yönetici kullanıcısı tarafından verilen Öğrenci yetkili kullanıcı adı ile sisteme girer. Düzeyine göre el alıştırma veya doğrudan figür öğretimi ekranına geçerek denemeler yapar. Belli bir ortalamanın üzerindeki başarı oranına eriştiğinde figürü öğrenmiş sayılır.

Figürün bilgisayara tanıtımı basamakları çizelge 5.3'te, figür öğretimi basamakları çizelge 5.4'te kısaca özetlenmiştir.

Çizelge 5.3. Öğretilecek figürün bilgisayara tanıtımı.

İşlem Basamakları Ana Başlık ve Alt Başlık			**Kullanılan Teknikler**
Ana Başlık	**Alt Başlıklar**		
Program ön işlem			- Makine epsilonunu hesaplama
Veri koleksiyonu alma	Ön İşlemler	İnceltme – Filtreleme	- Çift noktayı silme - Oto Seçim (Lagrange Lineer ve Lagrange 2. Mertebe Interpolasyon)
		Yumuşatma	- A. Sharma K'nın komşuları algoritması
		Gürültü Silme	- Donanımsal ve yazılımsal gürültüyü silme
	Bilgi Tabanına Kaydetme		- Kaydetme

Çizelge 5.4. Figür çizimi öğretimi basamakları.

İşlem Basamakları			**Kullanılan Teknikler**
Ana Başlıklar	**Alt Başlıklar**		
Program Ön İşlem			- Makine epsilonunu hesaplama
Veri koleksiyonu alma	Ön İşlemler	İnceltme – Filtreleme	- Çift noktayı silme - Oto Seçim (Lagrange Lineer ve Lagrange 2. Mertebe Interpolasyon)
		Yumuşatma	- A. Sharma K'nın komşuları algoritması
		Gürültü Silme	- Donanımsal ve yazılımsal gürültüyü silme
Özellik Çıkartma	3 Noktadan sıralı açı alma		- Referans noktaları belirleme - 3 Noktadan sıralı açıları alma
Normalleştirme	Çıkarılan özellik boyutlarını eşitleme		- Lineer Interpolasyon ve ya nokta eksiltme ile eşitleme
Bulanık İşlem	Bulanık Küme Oluşturma		- Gaussian Üyelik fonksiyonu ile bulanık küme oluşturma
	Elastik Karşılaştırma		- Elastik Karşılaştırma - Lukasiewicz Bulanık Birleşim
	Netleştirme		- Bulanık kümelerin Aritmetik ortalaması - Sözel değişkene çevirme
Deneme Tabanına Kaydetme			- Deneme tabanına kaydetme

Figür giriş ve gösterme ekranları 500*300 piksel boyutlarında olup her bir piksel Float en ve yüksekliğinde bir değerden oluşmaktadır. Tüm hesaplamalar donanıma özgü sınırlarda gerçekleşmektedir. Bu yüzden önce makine epsilonu (yazılımın hesaplayabileceği en küçük birim) bulunmuş ve bulanan epsilon hassasiyetine göre işlemler yapılmıştır. Geliştirilen uygulama Java'nın çalıştığı her platformda (Linux ve türevleri hariç) donanım sınırlarında çalışacak şekilde tasarlanmıştır.

5.2.1. Öğretilecek Figürün Bilgisayara Tanıtımı

5.2.1.1. Veri Koleksiyonu Alma

Dijital tablet üzerinde kalem ile yazı yazma esnasında x, y, z (basınç) ve iki nokta yazımı arasında geçen süre değerleri anlık olarak donanım ve yazılım sınırları altında tutulur. Veriler girilirken ardışık gelen aynı noktalar göz ardı edilir. Aşağıda ham veri toplama işlemi ifade edilmektedir.

x = yazım alanının x koordinatı üzerindeki t anındaki kalem izi (nokta)
y = yazım alanının y koordinatı üzerindeki t anındaki kalem izi (nokta)
z = t anındaki basınç değeri
sure= iki nokta yazımı arasında geçen zaman
P = {x,y,z,sure}
S = kalem basma ve kaldırma arasında yakalanan tüm ardışık gelen eşit x ve y değerleri hariç P değerler kümesi.

$$P = \left\{ \begin{array}{l} x \mid 0 <= x <= 500 \text{ ve} \\ y \mid 0 <= y <= 300 \text{ ve} \\ z \mid 0 < z <= 1 \\ sure \mid 0 <= sure \end{array} \right\}$$

$$S = P_{i-1} \neq P_i \Longrightarrow \sum P \quad (5.1)$$

Uygulamada Pen isimli bir sınıf tanımlanmış ve LinkedList nesnelerinde tutulmuştur.

5.2.1.2. Ön İşlemler

Yapılan çalışmada yumuşatma ve inceltme başlıkları altında iki ön işlem yapılmıştır.

[2] ∑ işareti ardışık zaman dilimlerinde tutulan x,y ve z (basınç) değerlerini ifade eden "P" noktalar koleksiyonunu ifade eder.

İnceltme – Filtreleme (Filtering - Thining)

Bazen inceltme olarak ta anılan filtreleme, çift veri noktalarının elimine edilmesi, nokta sayılarının azaltılması, ardışık noktalar arasındaki uzaklığı minimumda tutulması anlamına gelmektedir. Kullanıcı dijital kalem ile çizimini hızlı yaptığında ardışık noktalar arasındaki mesafe minimum uzaklığı aşmaktadır. Bölüm 4'te anlatılan oto seçim algoritmasını kullanan karma derece interpolasyon tekniği ile bu ardışık noktalar arasındaki mesafe minimum seviyeye çekilmektedir. Aşağıda inceltme işlemi için kullanılacak olan Bölüm 4'te anlatılan oto seçim algoritmasının kullandığı Lagrange İnterpolasyon formülü verilmektedir.

$$Li\ (x)\ =\ \frac{(x-x_1)\,.(x-x_2)\,.....(x-x_{i-1}).(x-x_{i+1}).(x-x_n)}{(x_i-x_1).(x_i-x_2).....(x_i-x_{i-1})\,(x_i-x_{i+1})\,(x_i-x_n)} \tag{5.2}$$

$$Li\ (x)\ =\ \prod_{\substack{j=1\\ j\neq 1}}^{n} \frac{x-x_j}{x_i-x_j} \tag{5.3}$$

$$yp(x)\ =\ \sum_{i=1}^{n} L_i\,(x)y_i \tag{5.4}$$

Aşağıda İnceltme işlemi yapabilmek için kullanılan otomatik seçim algoritması gösterilmektedir.

N, S özellik vektöründeki nokta sayısı;

1. Eğer (P_i - P_{i+1}) > 1 den ve i < N-3 den; 2 ve 4 arasındaki işlem adımları yapılır. Küçük ise i'nin değeri 1 arttırılır ve 1 adım tekrarlanır.
2. Ardışık 3 nokta (P_i, P_{i+1}, P_{i+2}) alınır.
3. Eğim1 ($P_{i-1} - P_i$) için, Eğim2 ($P_{i-2} - P_i$) noktaları için hesaplanır.

   3.1.Eğim1 == Eğim2 ise

$$yp(x)\ =\ \sum_{i=1}^{n=2} L_i\,(x)y_i \tag{5.5a}$$

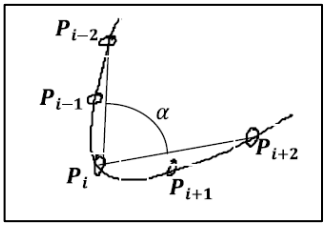
3.2.Eğim1 ≠ Eğim2 ise

$$yp(x) = \sum_{i=1}^{n=3} L_i(x) y_i \quad (5.5b)$$

4. Pi ve Pi+1 arasına hesaplanan değer eklenir. Birinci adıma atlanır.

Yumuşatma (Smoothing)

Donanımsal ve bireysel sebeplerden dolayı çizimlerde titreşimler oluşabilmektedir. Bu titreşimler çizimi oluşturan k'ıncı bir nokta için k'nın (k-neighbors) komşuları anlamına gelen listenin her bir noktasının işleme sokularak düzeltilmesi ile giderilmeye çalışılmıştır. (A. Sharma 2009 [11])



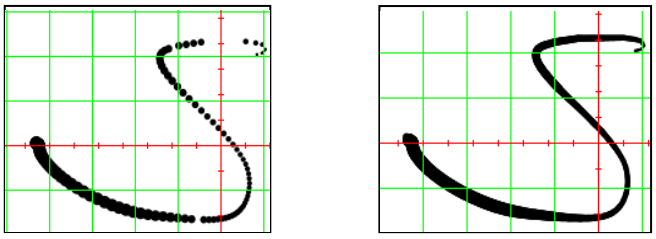
Şekil 5.6. P_i noktasında α açısının biçimi [11].

Şekil 5.6'da her iki yandaki iki komşu nokta ve bunların merkez noktaya göre oluşturduğu α açısıyla nokta düzeltmenin nasıl düşünüldüğü gösterilmiştir. Bu görüntüde bir önceki adımda oluşturulan S listesinde yer alan 5 nokta, çizim yumuşatılması için kullanılmıştır. Pi noktasının düzeltilmesi için, P_{i-2}, P_{i-1}, P_i, P_{i+1}, P_{i+2} noktalarından faydalanılmaktadır. Aşağıda çizim yumuşatma algoritması gösterilmektedir.

1. t Ataması: Listedeki çizimlerin sayısı ve (k = 1) atanmaktadır.
2. Her bir çizim için (k ≤ t) olana kadar 3 aşama tekrar edilecektir.
3. 3.1. k'ıncı çizimdeki noktaların toplam sayısı ile m bulunacaktır.
   3.2. Her P_i noktası için 3.3 ve 3.4 tekrar edecektir. (i= 3,4, ……. , m-2.)
   3.3. P_i'nin α açısı, (P_{i-2}), (P_{i+2}) kullanılarak hesaplanacaktır.
   3.4. $P_{ix}=(P_{(i-2)x} + P_{(i-1)x} + \alpha * P_{ix} + P_{(i+1)x} + P_{(i+2)x}) / (2 * 2 + \alpha)$
   $P_{iy}=(P_{(i-2)y} + P_{(i-1)y} + \alpha * P_{iy} + P_{(i+1)y} + P_{(i+2)y}) / (2 * 2 + \alpha)$

4. k=(k+1)
5. Çık.

Şekil 5.7'de "M" harfi için inceltme ve yumuşatma işlemi uygulanmadan önceki ve sonraki halleri gösterilmektedir.



(a) (b)

Şekil 5.7. Ham veri ve ön işlemlerden geçirilmiş veri, a) Ham veri, b) İnceltme ve yumuşatma ön işlemlerinden geçirilmiş veri.

5.2.1.3. Bilgi Tabanına Kaydetme

Uzman tarafından oluşturulan şekil Java nesnesi olarak MySQL veritabanına kaydedilir. Kaydedilmeden önce donanımsal ve yazılımsal sebeplerden ötürü meydana gelen gürültüler (oluşan boş LinkedList nesneleri) temizlenir. Şekil 5.8' de bilgisayara öğretilen figürün gürültüsünü silen ve bilgi tabanına kaydeden kod bloğu gösterilmektedir.

```
    private void kaydetActionPerformed(java.awt.event.ActionEvent evt) {
         LinkedList <LinkedList<Pen>> figur=g.getFigur();
            for (int i = 0; i <figur.size(); i++) {
                if (figur.get(i).size()==0){
                figur.remove(figur.get(i));
                i--;
                }
            }
        em=JpaUtil.getEMF();
        Kayitturu kt= em.find(Kayitturu.class, (tur.getSelectedIndex()+1));
        em.getTransaction().begin();
        Bilgitabani bt=new Bilgitabani();
        bt.setCizimAD(figurAd.getText());
        bt.setAciklama(aciklama.getText());
        bt.setKayitTuru(kt);
        bt.setSekil(g.figur);
        bt.setUzman(uygulama.us);
        em.persist(bt);
        em.getTransaction().commit();
        em.close();
    }
```

Şekil 5.8. Öğretilen figürün bilgi tabanına kaydedilmesi.

5.2.2. Figür Çizimi Öğretimi

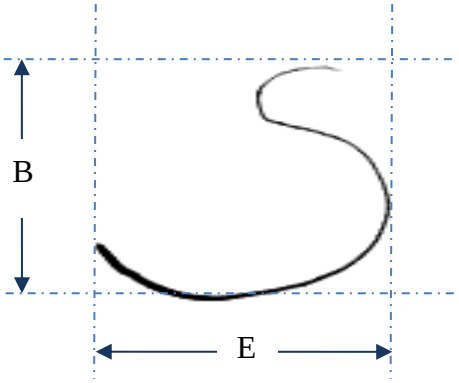
Öğretilecek figürün bilgisayara tanıtımında kullanılan veri koleksiyonu alma ve ön işlemler aynen figür çizimi öğretiminde uygulanmıştır.

5.2.2.1. Özellik Çıkartma

Bilgi teknolojileri çeşitlerindeki hızlı gelişme, karar alma sürecinde kullanılan ve biriktirilen bilgi miktarını da artırmıştır. Toplanan verilerin büyüklük ve karmaşıklığındaki artış sebebiyle, çalışmalar bu verileri saklama, yönetme ve analiz etme üzerine yoğunlaşmaktadır [40]. Bütün bu işlemler doğrultusunda depolanan verilere, figürlere ve karakterlere sistemli bir özellik çıkartma işlemi uygulandığı takdirde, bu veriler azaltılıp, anlamlı hale getirilecek ve sistematik şekilde sınıflandırılabilecektir.

Özellik çıkartma karakter tanımada en önemli noktalardan biridir. Sistemin performansına çok büyük etkisi vardır. Çalışmamızda bir figürü oluşturan her çizime 3 noktadan bakarak ardışık olarak ilerleyen noktalara bakış noktalarından bir doğru

çizerek bu doğrunun 0-360 derece aralığındaki açı derecesini özellik vektörü olarak kullanılmıştır. Bu şekilde 3 noktadan açı değişimlerini tutan bir sıralı açı vektörü oluşturulmuştur. Aşağıda referans noktaların belirlenmesi ve özellik çıkartma tekniği anlatılmıştır.



B = y koordinatlarının en küçüğü ve en büyüğü arasındaki mutlak fark

E = x koordinatlarının en küçüğü ve en büyüğü arasındaki mutlak fark

Şekil 5.9. Referans noktalar için enin ve yüksekliğin belirlenmesi.

En ve yükseklik belirlendikten sonra sırasıyla aşağıdaki işlemler yapılır.

1. Birinci referans nokta için (Sağ Üst Köşe);

RN1x = EnBüyük(E,B) * 1,5+ (en büyük ve en küçük x değerlerinin tam ortası)
RN1y = (en büyük ve en küçük y değerlerinin tam ortası) – (EnBüyük(E,B) * 1,5)

2. İkinci referans nokta için (Şeklin altı);

RN2x = en büyük ve en küçük x değerlerinin tam ortası
RN2y = EnBüyük(E,B) * 1,5 + (en büyük ve en küçük y değerinin tam ortası)

3. Üçüncü referans nokta için (Soldan bakış)

RN3x = (en büyük ve en küçük x değerlerinin tam ortası) – (EnBüyük(E,B) * 1,5)
RN3y = en küçük y değeri

Referans noktaları belirlendikten sonra S dizisindeki her bir noktanın referans noktalara göre açıları belirlenir ve bu değerler bir önce alınan açı değerinden en az 2

derece farklı ise bu açı değeri ve diğer referans noktaların açı değeri hesaplanarak özellik listesine (SO) eklenir. Bu çalışmada 2 derecelik bir değişiklikte açılar alınmıştır hassasiyetin önemine göre bu değer değiştirilebilir.

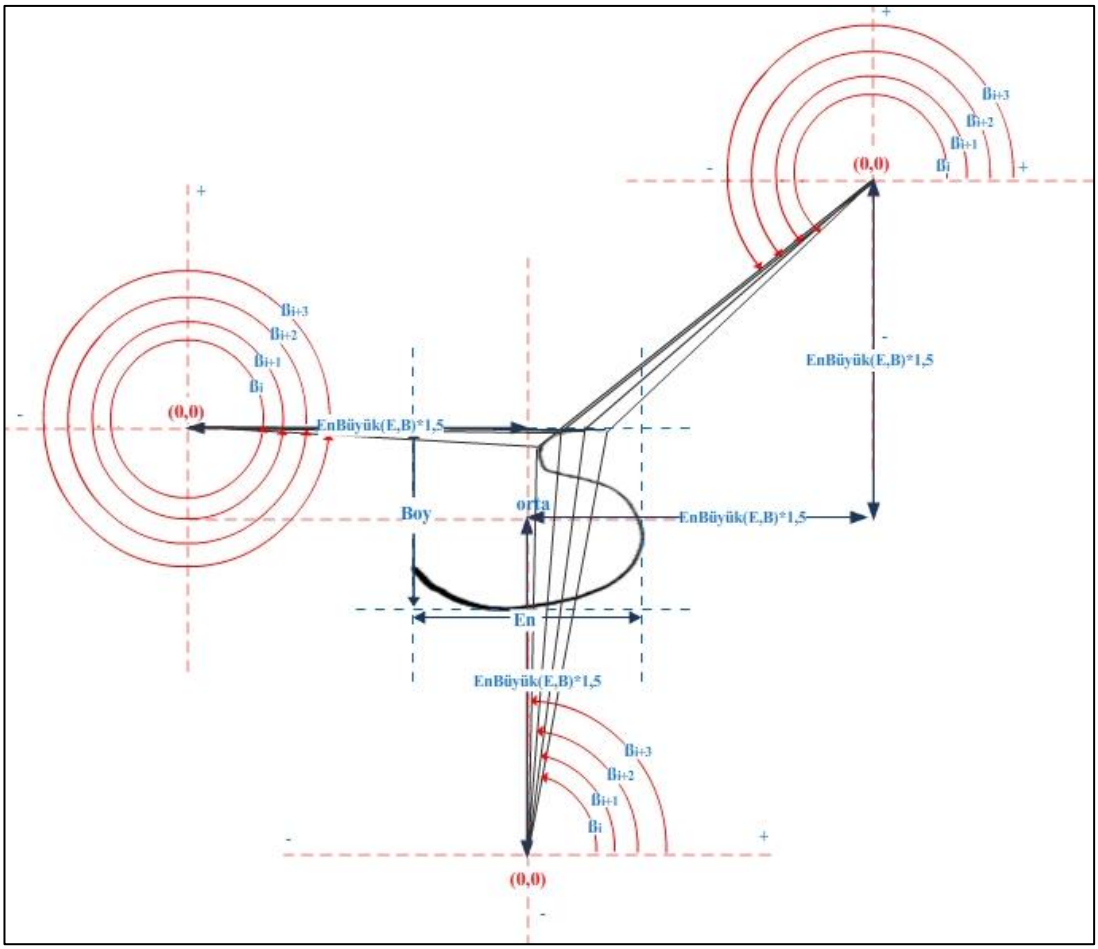
Her bir anlık görüntü 3 referans noktaya göre açıları barındıran SAOzellik adlı Java sınıflarında tutulmuştur. Eğer her bir referans noktayı sırasıyla x, y ve z olarak adlandırırsak ve ß x, y, z açıların i sırasındaki durumları ise;

$$ß = \{x, y, z\}$$

$$\forall(x_i - x_{i-1}) \geq 2 \vee \forall(y_i - y_{i-1}) \geq 2 \vee \forall(z_i - z_{i-1}) \geq 2 \Rightarrow \sum ß \quad (5.6)$$

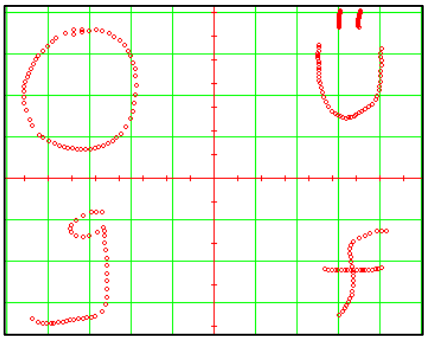
şeklinde formüle edilebilir.

Şekil 5.10'da referans noktaların belirlenmesi ve sıralı açıların alınması gösterilmektedir.

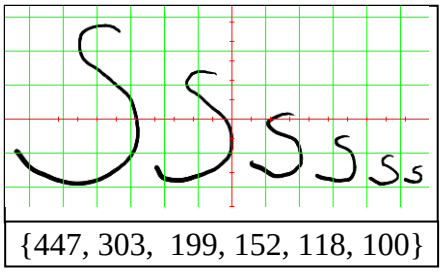


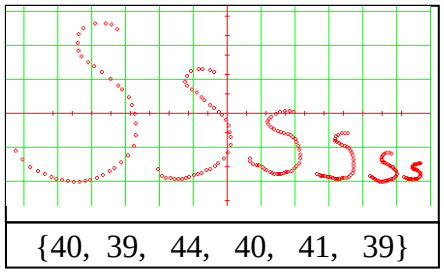
Şekil 5.10. Referans noktaların belirlenmesi ve sıralı açıların alınması.

Eğer figür birden fazla parçadan oluşuyorsa her parça için ayrı ayrı özellik çıkartma işlemi uygulanır. Daha sonra tüm parçaların birleşimi olan tam figür içinde özellik çıkartma işlemi uygulanır. Bu yöntemle çizilen figürün tamamı ve parçaları hakkında bilgisayarın yorum yapması ve daha doğru bir tanıma yapması sağlanmış olmaktadır. Şekil 5.11'de özellik çıkartma işleminden geçmiş bazı harflerin sıralı açılarının alındığı noktaları gösterilmektedir.



Şekil 5.11. Bazı harflerin sıralı açılarının alındığı noktalar.





(a) (b)

Şekil 5.12. Çeşitli büyüklüklerdeki "S" harfi için özellik çıkartma işlemi uygulaması, a) Sırasıyla soldan sağa her "S" harfi için toplam nokta "P" sayısı, b) Şekil a'daki S harflerinden çıkartılan özeliklerin temsil ettiği noktalar ve sırasıyla soldan sağa her "S" harfi için çıkartılan özellik sayıları.

Şekil 5.12'de de görüldüğü gibi 3 noktadan bakarak alınan sıralı açıların farklı boyutlardaki benzer şekillerde yakın sayılarda özellikler çıkardığı saptanmıştır. Bu yöntem şeklin boyutları ile değil doğrudan şeklin kendisi ile ilgili bilgiler toplamaktadır. Bu sayede taban kaydırma düzeltmesi ve boyut normalizasyonu işleminin kullanımını gereksiz kılmaktadır.

Bu işlem hem Bilgi Tabanındaki figür hem de öğrencinin girdiği figür için yapılır.

5.2.2.2. Normalizasyon

Eğer Bilgi Tabanındaki figürün çıkartılan özellik sayısı ile öğrencinin girdiği figürün özellik sayısı eşit değil ise Bilgi Tabanındaki figür baz alınarak ya Lineer

İnterpolasyon ile orantılı aralıklarla deneme figürüne veriler eklenir ya da belli orantılı aralıklarla özellik eksiltme yapılarak iki figürün özellik sayıları eşitlenir.

$$y = f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} (x - x_k) \quad (5.7)$$

Çizelge 5.5' te Deneme harf girişinin Bilgi Tabanındaki model harfin özellik sayısına eşitlenmiş hali gösterilmektedir.

Çizelge 5.5. Özelik listesinde 35 adet özellik barındıran "S" denek harfinin normalizasyon işlemine tabi tutulduktan sonra eklenen sahte özellikler.

	S Denek Harfi				S Model Harfi		
	144	0	273		139	1	279
	144	2	272		141	0	278
	143	4	271		142	0	276
	143	5	269		142	2	274
	143	7	268		142	4	272
	142	9	268		142	6	271
	141	11	268		141	8	270
Sahte özellik	140	12	268		140	10	270
	139	13	269		138	12	271
	137	14	270		136	12	273
	136	14	272		135	13	275
	135	14	274		133	14	277
	133	15	275		131	14	279
	132	15	277		129	15	281
Sahte özellik	131	15	277		128	16	283
	130	16	278		126	17	284
	129	18	279		124	18	286
	128	18	281		123	20	287
	127	20	282		122	22	289
	125	21	283		121	24	289
	125	23	283		121	26	288
Sahte özellik	124	24	283		121	28	287
	124	25	284		122	28	285
	123	27	284		123	30	283
	123	29	283		123	30	281
	123	30	281		124	31	279
	124	31	279		124	32	277

Çizelge 5.5. (devam ediyor).

	125	32	277		125	32	275
Sahte özellik	125	32	276		126	33	273
	125	32	275		127	33	271
	126	33	273		127	34	269
	127	33	271		128	34	267
	127	34	269		129	34	265
	128	34	267		130	34	263
	129	35	265		131	35	261
Sahte özellik	129	35	264		132	35	259
	129	35	263		133	34	257
	130	36	261		135	34	256
	132	35	259		136	34	254
	133	35	257		138	33	253

5.2.2.3. Bulanıklaştırma

Bulanıklaştırma aşamasında SO özellik vektöründeki açı değerleri ve Bilgi Tabanında bulunan açı değerleri Gausian Üyelik fonksiyonu ile işleme sokulmaktadır. Gaussiyan üyelik fonksiyonu eşitlik 5.8' de görüldüğü gibidir.

$$\mu_A(x; m, \sigma) = \exp[\frac{-(x-m)^2}{2\sigma^2}] \quad (5.8a)$$

$$A = \sum \mu_A(x; m, \sigma) \quad (5.8b)$$

Bu fonksiyonda **m** fonksiyonun merkezini **q** ise genişliğini ifade eder. Genişlik değerini hassasiyete göre Uzman tarafından belirlenir. Fonksiyonun merkezini ifade eden **m** ise Bilgi Tabanındaki SO dizisine uygun sıradaki elemanıdır. Bu işlemde Elastik Karşılaştırma tekniği kullanılmıştır.

5.2.2.4. Elastik Karşılaştırma – Bulanık İşlem

Scatt' ın (2000) elastik karşılaştırma tekniğini biraz değiştirip bulanık mantık Gaussian üyelik fonksiyonuna uyarlanmıştır. Elastik karşılaştırma yöntemi ile SOB bulanık kümesini oluşturan formül Eşitlik 5.9'da görüldüğü gibidir.

N | SO dizisinin eleman sayısı;

$$\textbf{SOB}=\sum_{i=0}^{N}\begin{cases}\text{EnBüyük}\begin{Bmatrix}\mu_A(SO_i,SM_i,\sigma)\\ \mu_A(SO_i,SM_{i+1},\sigma)\\ \mu_A(SO_i,SM_{i+2},\sigma)\end{Bmatrix} & i=1\ ve\ 2<N\\ \text{EnBüyük}\begin{Bmatrix}\mu_A(SO_i,SM_{i-1},\sigma)\\ \mu_A(SO_i,SM_i,\sigma)\\ \mu_A(SO_i,SM_{i+1},\sigma)\end{Bmatrix} & i>1\ ve\ i<N\\ \text{EnBüyük}\begin{Bmatrix}\mu_A(SO_i,SM_{i-2},\sigma)\\ \mu_A(SO_i,SM_{i-1},\sigma)\\ \mu_A(SO_i,SM_i,\sigma)\end{Bmatrix} & i=N\ ve\ 2<N\\ \{\mu_A(SO_i,SM_i,\sigma)\}\ N<3\end{cases} \quad (5.9)$$

Bu işlem sonunda her biri en fazla 3 elamandan oluşan N adet bulanık küme oluşturulur. Daha sonra her bir bulanık küme Lukasiewicz Bulanık Birleşim işleminden geçirilerek tek elemanlı kümeye indirilir.

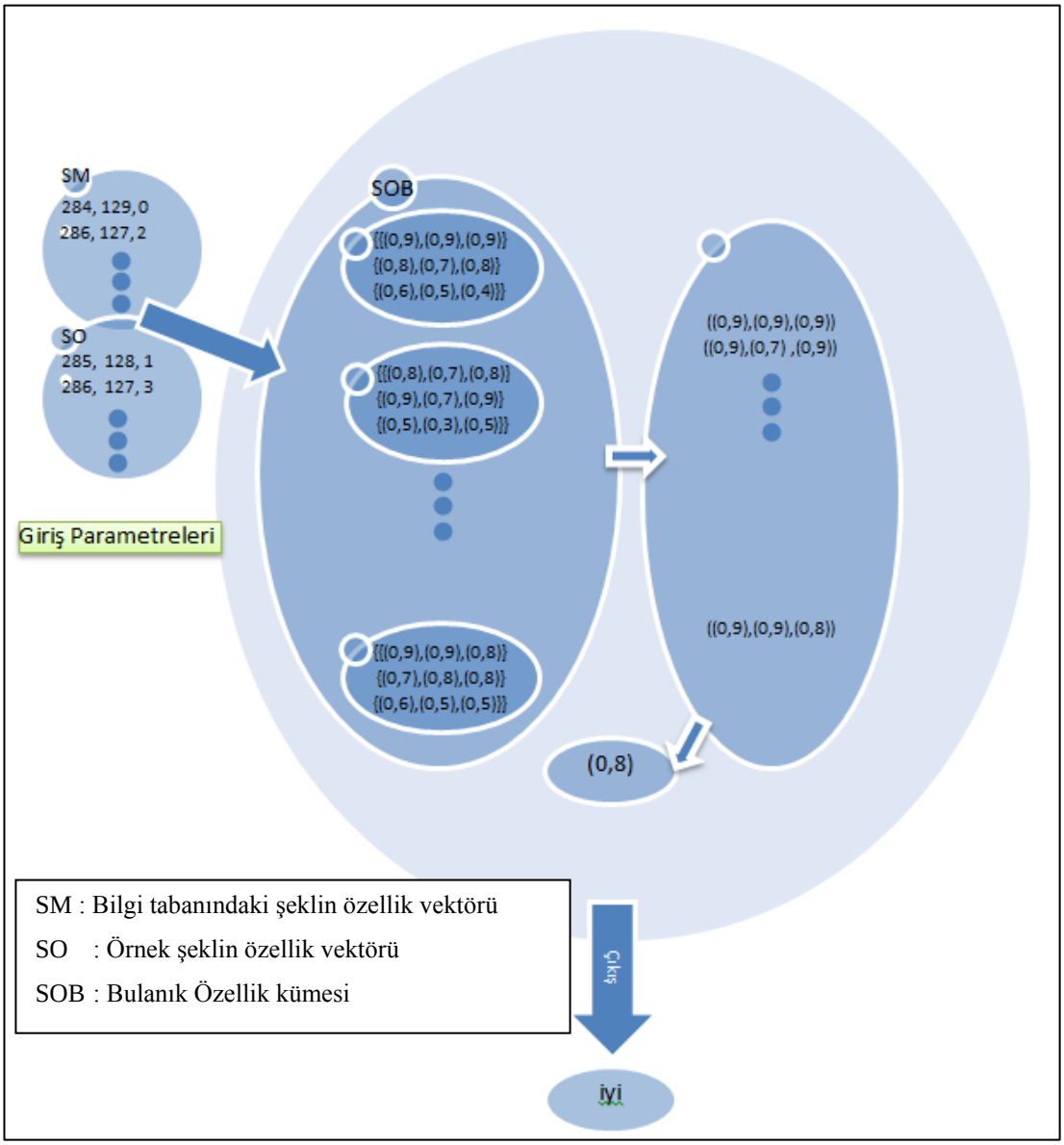
5.2.2.5. Netleştirme - Tanıma – Yorumlama

Elastik karşılaştırma ile oluşturulan üyelik derecelerini tutan SOB özellik dizisinin aritmetik ortalaması bize çizilen figürün başarım oranını vermektedir. Eşitlik 5.10'da aritmetik ortalama ile bulunan üyelik derecesini göstermektedir.

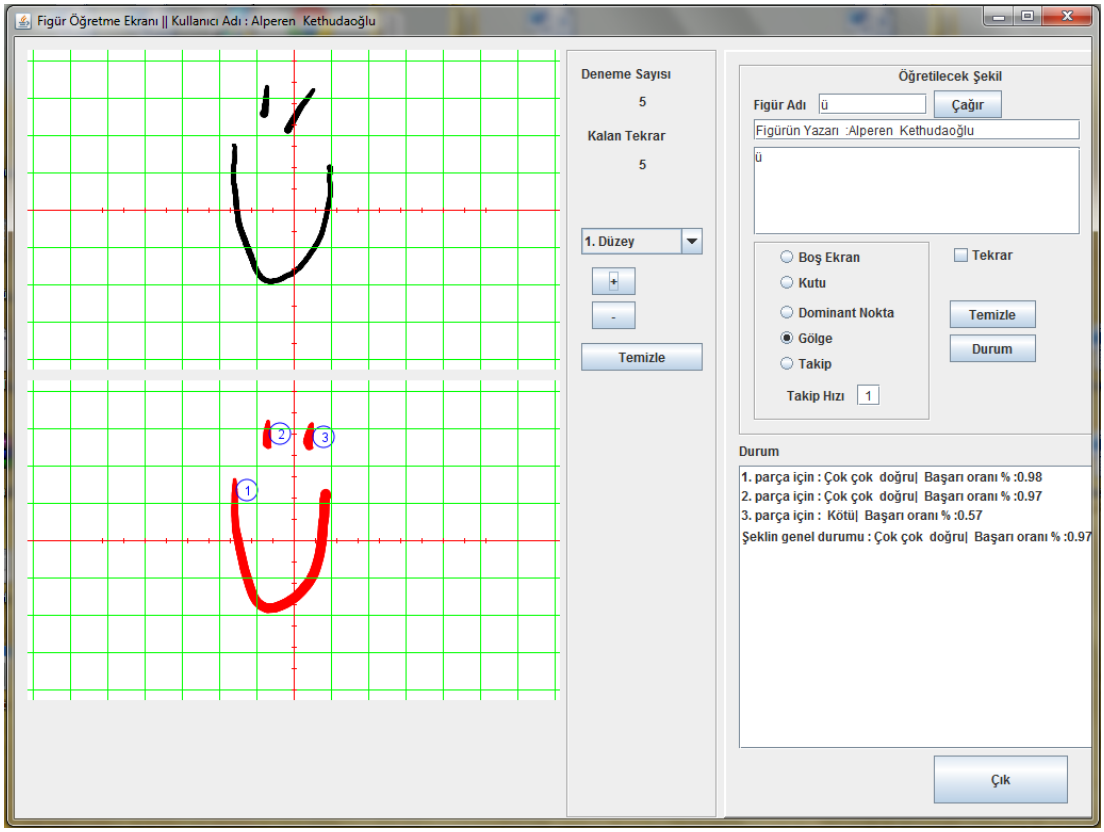
n | SOB dizisinin eleman sayısı.

$$\mu_{aort}=\frac{\sum_{i=1}^{n}(SOB_i)}{n} \quad (5.10)$$

Figürü oluşturan her bir parça için ayrı ayrı çıkartılan üyelik dereceleri Bulanıklık Dereceleri adlı çıkarım tablosuna bakılarak sonuç hakkında figürü oluşturan her bir parça için sözel dönütler verilir. Aynı şekilde figürün tamamı tek parça bir çizim farz edilerek alınan genel üyelik derecesi de çıkarım tablosuna bakılarak öğrenciye Figürün genel başarımı hakkında sözel dönütler verilmesi sağlanmış olur. Şekil 5.13'te Bulanık İşlem gösterilmektedir. Şekil 5.14'te ise örnek bir figür öğrenme durumu gösterilmektedir.



Şekil 5.13. Bulanık işlem giriş ve çıkış parametreleri.



Şekil 5.14. Ü harfi için örnek ekran görüntüsü.

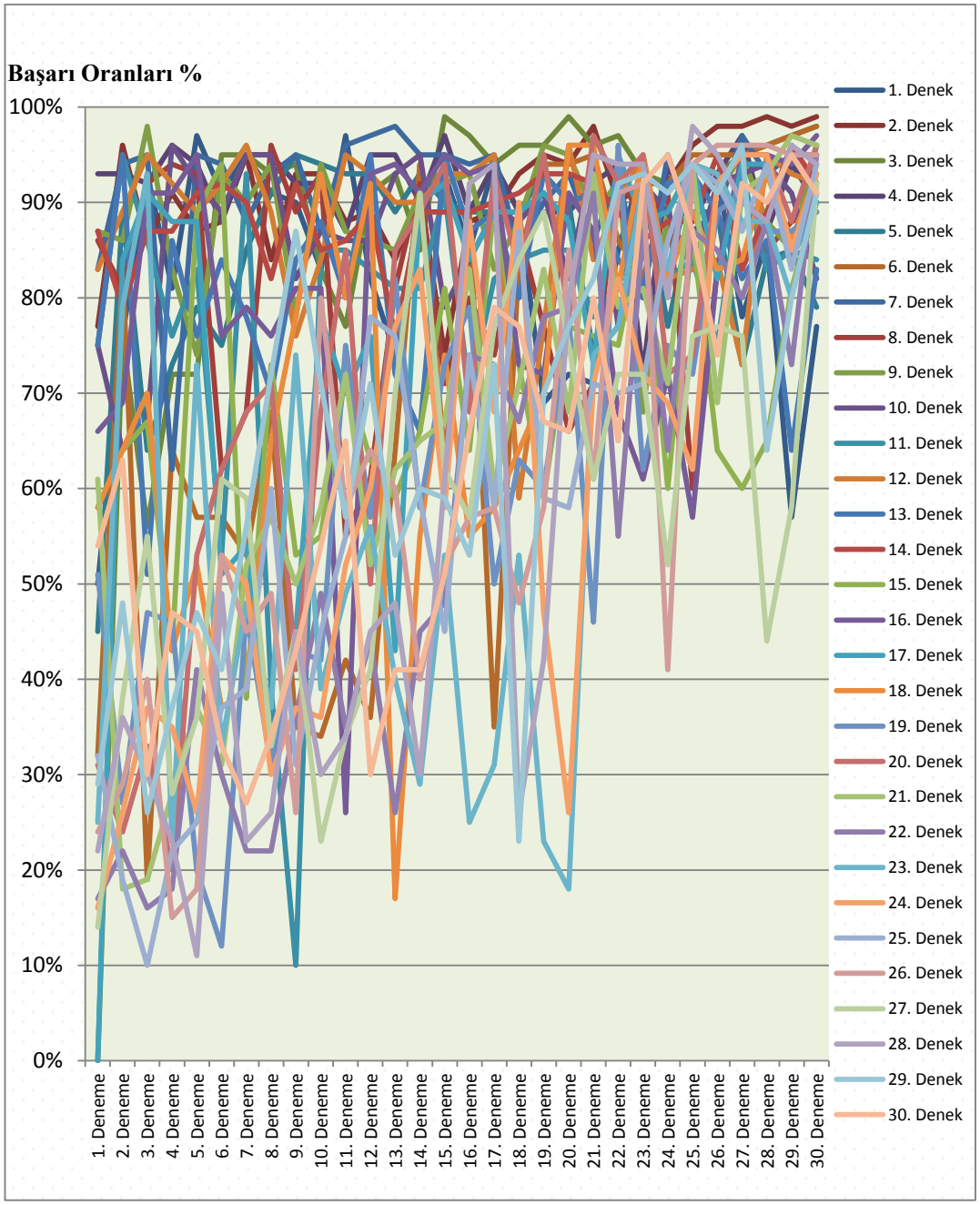
5.2.2.6. Deneme Tablosuna Kaydetme

Çıkarılan özellikler dizisi SO ve ön işlem basamakları yapıldıktan sonraki ham veri dizisi olan S, deneme veritabanına kaydedilmektedir. Bu şekilde öğrencinin gelişim evrelerini takip etmek için elimizde veri olması sağlanmakladır. Şekil 5.15' te SO özellik LinkedList'ler nesnesini veritabanına kaydeden ve okuyan Varlık sınıfı ilgili kod bloğu görünmektedir.

```
public LinkedList<LinkedList<SAOzellik>> getOzellikler() {

ByteArrayInputStream bais;
ObjectInputStream ois;
    try {
            bais=new ByteArrayInputStream(sob);
            ois=new ObjectInputStream(bais);
            ozellikler=(LinkedList<LinkedList<SAOzellik>>) ois.readObject();
        }catch (IOException e){
            e.printStackTrace();
        } catch (ClassNotFoundException e){
            e.printStackTrace();
        }
    return ozellikler;
}

public void setOzellikler(LinkedList<LinkedList<SAOzellik>> ozellikler) {

this.ozellikler = ozellikler;
ByteArrayOutputStream baos;
ObjectOutputStream ous;
baos=new ByteArrayOutputStream();
   try{
            ous=new ObjectOutputStream(baos);
            ous.writeObject(ozellikler);
            ous.close();
      } catch (IOException e){
            e.printStackTrace();
      }
this.sob=baos.toByteArray();
}
```

Şekil 5.15. Varlık sınıfı SO özellik vektörünü veritabanına kaydeden ve okuyan kod bloğu.

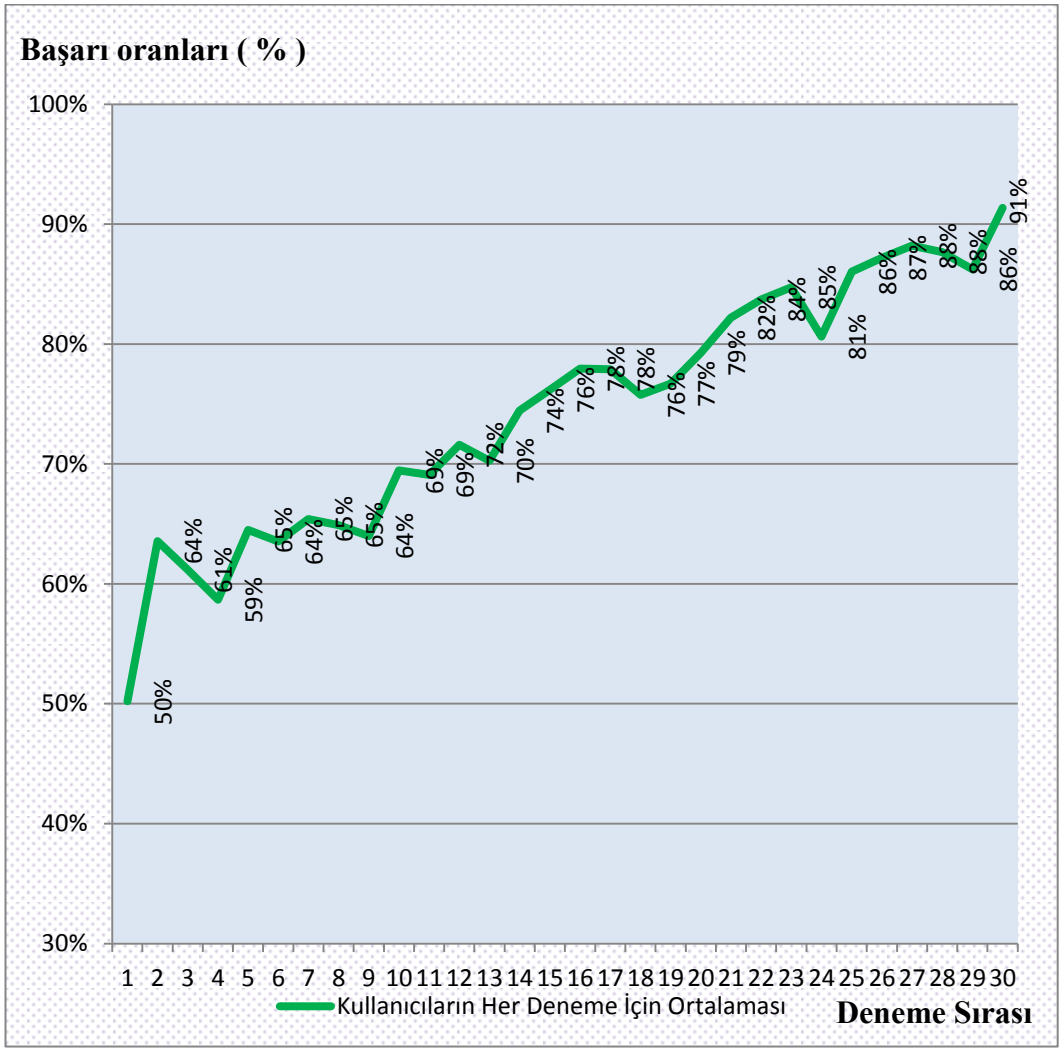
15- 40 yaş arası 30 kişi üzerinde yapılan uygulama sonuçlarına göre Şekil 5.16.'daki grafikte, bütün kullanıcıların deneme sonuçları çakıştırılmış, elde edilen sonuç yazılımın öğretime belirgin şekilde katkı sağladığını göstermiştir. Deneklerin ilk denemelerinde %1 ile %93 gibi oldukça geniş bir aralıkta olan başarı oranları, 30. denemenin sonunda %75 ile %98 aralığında yoğunlaşmış, hata oranında gözle görülür bir azalma meydana gelmiştir.



Şekil 5.16. Kullanıcıların başarı trendi grafiği.

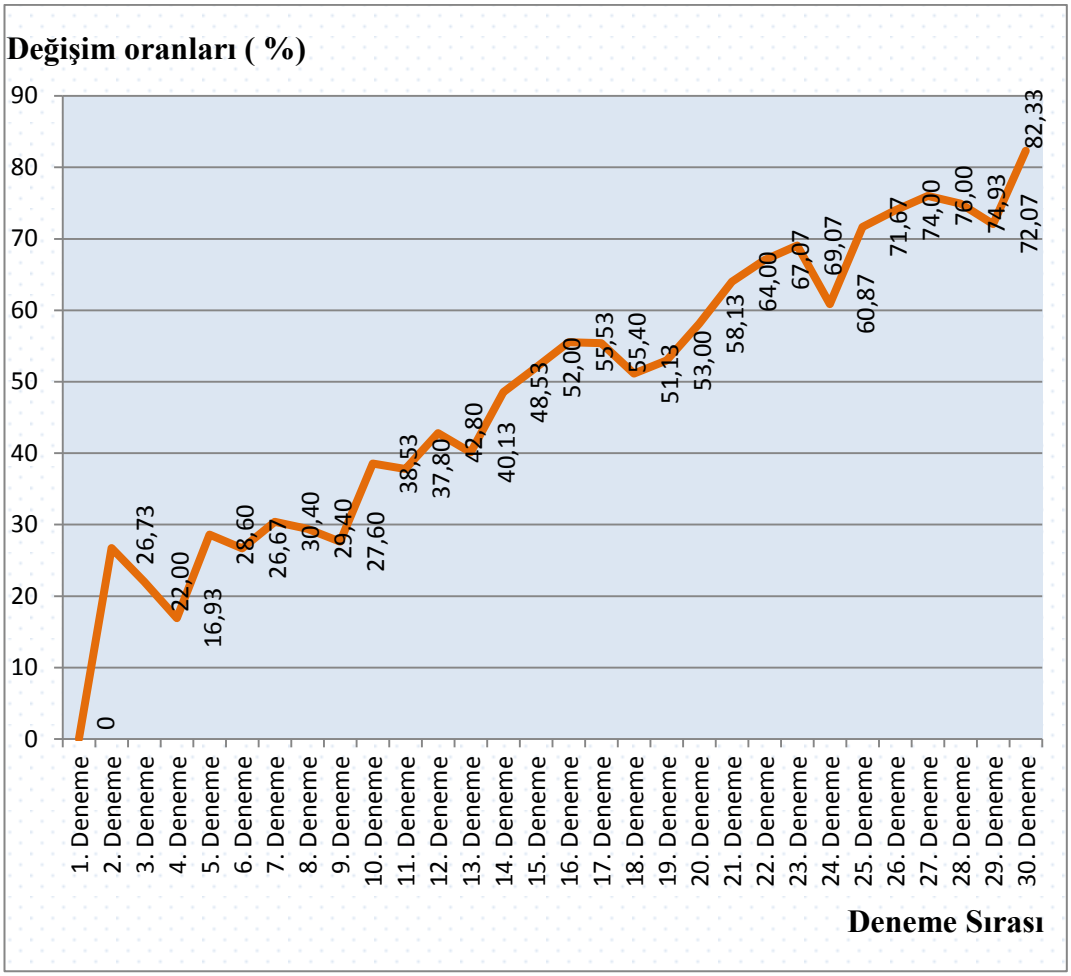
Şekil 5.17.'deki grafikte kullanıcıların deneme sonuçlarından yola çıkılarak, her bir deneme için ortalama değerler hesaplanmış ve bu şekilde öğretim yazılımının genel başarısı gözlenmiştir. Birinci deneme üyelik oranı ortalama %50 iken, 30. denemenin sonunda üyelik oranı ortalama %91'e çıkmıştır. Genel olarak yazılım öğretiminin başarısı incelendiğinde, birinci denemeden sonuncu denemeye kadar kullanıcıların başarı ortalamalarının yükseliş trendinde olduğu gözlenmektedir. Başarı trendinin

yükselişte olması, yazılımın kullanıcılara istenilen şekli öğretebildiğini göstermektedir.



Şekil 5.17. Öğretimin başarı grafiği.

Şekil 5.18.'deki grafikte kullanıcıların başarı ortalamasındaki yüzde değişme incelenmiştir. Buna göre kullanıcıların her denemesinde ilk denemeye göre nasıl bir yüzde değişim meydana geldiği hesaplanmıştır. Son 5 denemede başlangıca göre en yüksek oranda yüzde değişim sağlanmıştır. Bu sonuç, tekrar sayısı arttıkça, el becerilerinin geliştiğini, anlamanın ve öğrenmenin pekiştiğini göstermektedir.



Şekil 5.18. Başlangıç ortalama değerine göre değişme (%).

Yukarıdaki grafiklerden de anlaşılacağı gibi, gerek öğretimin bir bütün olarak başarısında, gerekse kullanıcıların bireysel performanslarının gelişiminde yazılım anlamlı katkı sağlamaktadır. Kullanıcıların bireysel performanslarına ilişkin grafikler ekte verilmiştir. Kullanıcıların öğreniminde başarı düzeyinin düşmesi ve artmasının yapılan denemeler sırasındaki gözlemler neticesinde konsantrasyona bağlı olduğu tespit edilmiştir. Yazılım ödül ve puan mekanizması ile geliştirilirse daha faydalı olacağı düşünülmektedir.

BÖLÜM 6

SONUÇLAR

Bu çalışmada, bir çevrimiçi figür çizimi öğretimi sistemi geliştirilmiştir. Etkin sonuç alabilmek için karakter öğretimi üzerinde durulmuştur. Ayrıca, karşılaşılan hata kaynaklarını en aza indirgeme ve uygun interpolasyon hesabının seçimi için bir çalışma yapılmıştır [41]. Kullanıcılara çizim ve karakter öğretimi için bir uzman sistem tasarlanmıştır. Çalışma sonucunda;

- Okul öncesi eğitiminde doğru çizim senaryoları ile doğru dönütler verilerek öğrenme güdülenmesi arttırılabilir ve daha hızlı ve etkin bir öğrenme sağlanabilir.

- İlköğretimin ilk basamağındaki öğrencilerin yazı yazabilmeleri için gerekli olan ince kas gelişiminin tamamlanmasına yardımcı olmaktadır.

- İlköğretimin ilk basamağındaki öğrencilerin el motor becerilerinin gelişimine katkı sağlamaktadır

- Gardner'in Çoklu Zeka Kuramı'na uygun şekilde öğrenciye farklı öğrenme stillerini sunduğu için farklı öğrenme stiline sahip öğrencilerin öğrenmesine imkan sağlar [42].

- Sayısal kalemi uzun süreler kullanan kişilerin çizdiği karakterlerde tanınma oranlarının yeni kullanmaya başlamış kişilere göre daha iyi sonuçlar verdiği gözlemlenmiştir.

- Harf öğretiminde yerden ve zamandan tasarruf yapmanın çevrimiçi öğretim ile mümkün olduğu gözlemlenmiştir.

- Bilgisayar tarafından doğru şekilde çizilmesi öğretilen harfler ile el yazısı harflerde belli bir standarta ulaşmak mümkündür.

- Bulanık Mantık Gaussian Üyelik fonksiyonuna çeşitli genişlik değerleri verilerek Figür çizimi öğretimi için öğretme ve tanıma hassasiyetinin kontrol edilebileceği saptanmıştır.

- İnteraktif bir ortamda çeşitli şekil, figür veya karakterlerin öğretiminin mümkün olduğu ve bunun hedef kitleye göre daha eğlenceli hale getirilerek iyi sonuçların alınabileceği düşünülmektedir.

- Geliştirilen özellik çıkartma tekniği çevrimiçi ve çevrimdışı karakter tanıma sistemlerinde kullanılabilir.

- Geliştirilen özellik çıkartma tekniği şeklin tabanı veya boyutları ile değil doğrudan şeklin kendisi ile ilgili bilgiler tuttuğu için çevrimiçi ve çevrimdışı karakter tanıma sistemlerinde kullanılan bazı normalizasyon işleminin kullanımını gereksiz kılar.

- Geliştirilen özellik çıkartma tekniği çevrimiçi ve çevrimdışı karakter tanımada kullanılan bazı gürültü azaltma tekniklerinin kullanımını gereksiz kılar.

- Geliştirilen özellik çıkartma tekniği üçgen veya yamuk üyelik fonksiyonu ile bulanıklaştırılmış karakter tanıma sistemlerinde etkin sonuçlar verebileceği düşünülmektedir.

Günümüzde az zamanda çok ve etkili bir öğrenme için interaktif ortamlardan faydalanmak önem kazanmıştır. Bu tezde interaktif bir öğrenme modeli oluşturularak bu alandaki boşluğun doldurulmasına katkı sağlanması hedeflenmiştir.

KAYNAKLAR

1. Nabiyev, V., "Yapay Zeka Problemler – Yöntemler – Algoritma**"**, ***Seçkin Kitapevi***, Ankara, 523-583 (2005).

2. Avcı, E., Türkoğlu, İ. ve Poyraz, M., "An inteligent target recognation system based periodogram for pulsed radar system", ***Gazi Üniversitesi Fen Bilimleri Enstitüsü Dergisi***, 18 (2): 259-272 (2005).

3. Dursun, Ö. O., "Meteorolojik verilerin veri madenciliği ile değerlendirilmesi", Yüksek Lisans Tezi, ***Fırat Üniversitesi Fen Bilimleri Enstitüsü***, Elazığ, Türkiye, 4-8 (2005).

4. Öztürk, A., "Osmanlıca karakterlerin bilgisayar destekli tanınması", Yüksek Lisans Tezi, ***Gebze Yüksek Teknoloji Enstitüsü***, Kocaeli, Türkiye, 5-30 (1998).

5. Connell, S. D. and Jain, A. K., "Template -based online character recognition", ***The Journal Of The Pattern Recognition Sociey***, 34 (1):1-14 (2001).

6. Kumar, V., "Template-based writer - independent online character recognition system using elastic matching", ***MIT International Journal of Computer Science & Information Technology***, 1 (1): 15-18 (2011).

7. Aras, P., "Bilgisayar destekli el yazısı karakterlerini tanıma sistemi tasarımı", ***İstanbul Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, Türkiye, 4-5 (2006).

8. Musayev, E., "Bilgisayar destekli karakter tanıma sistemi tasarımı", ***İstanbul Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, Türkiye, 10 (2004).

9. Teo, K. K., "Low cost number plate recognition", Msc. Thesis, ***Queensland University***, Brisbane, Australia, 1-20 (2003).

10. Tappert, C. C., Suen, C. Y. and Wakahara, T., "The state of art in on-line handwriting recognition", ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, 12 (8): 787-808 (1980).

11. Sharma, A., "Online handwriting gurmukhi character recognition", Ph.D. Thesis, ***Thapar University***, Patiala, India, 68-69 (2009).

12. Kumar, V., and R. Ahuja, ''Online character recognition system using elastic matching'' ***First International Conference on Industrial and Information Systems***, Sri Jayawardenapure-Kotte, Sri Lanka, 8-11 (2006).

13. Scott D. C., "Online handwriting recognition using multiple pattern class models", Ph.D. Thesis, ***Michigan State University***, Michigan, USA, 78-80 (2000).

14. Vural, E., Oflazer, K. ve Yanıkoğlu, B., "Türkçe için tablet pc ortamında Çevrimiçi yazı tanıma sistemi", ***Signal Processing and Communication Applications Conference, Proceedings of the IEEE 12th***, Kuşadası, Türkiye, 607-610 (2004).

15. Shaw, A. C., "A formal picture description scheme as a basis for Picture processing systems", Slac-Sub-402, ***Standford Universty***, California, USA, 7-24 (1968).

16. Agarwal, S. and Kumar, V., "Online character recognition", ***Proceedings of the Third International Conference on Information Technology and Applications***, Sydney, Australia, 698-703 (2005).

17. Singh, H. and Sharma, R. K., "Moment in online handwriting character recognition", ***Proceedings of National Conferance on Challenges & Opportunities in Information Technology***, Punjab, India, 225 (2007).

18. Allahverdi, N., "Uzman Sistemler Bir Yapay Zeka Uygulaması", ***Nobel Yayın Dağıtım***, Ankara, Türkiye, 1-50 (2001).

19. İnternet: Türk Dil Kurumu, "Büyük Sözlük", **http://tdkterim.gov.tr/bts/**, (2012).

20. Baykal, N. ve Beyan, T., "Bulanık Mantık İlke ve Temelleri", ***Bıçaklar Kitapevi***, Ankara, Türkiye, 19 -113 (2004).

21. Öztemel, E., "Yapay Sinir Ağları", ***Papatya Yayıncılık***, İstanbul, Türkiye, 15-16 (2003).

22. Ronald, M., "Java™ Number Cruncher: The Java Programmer's Guide to Numerical Computing", ***Prentice Hall PTR***, New Jersey, USA, 1-480 (2003).

23. Karagöz, İ., "Sayısal Analiz ve Mühendislik Uygulamaları", ***Nobel Yayıncılık***, Ankara, Türkiye, 2-210 (2008).

24. Yüzer, A. H., ''Prostat ameliyatı için μP veya μC temelli insan total kanında veya Plazmasında bulunan Na+ konsantrasyonunu online (ameliyat süresince) sürekli ölçecek sensör tasarımı'', Yüksek Lisans Tezi, ***İnönü Üniversitesi Fen Bilimleri Enstitüsü***, Malatya, Türkiye, 30-32 (2002).

25. Ningping, S., Ayabe, T. and Nishizaki, T., "Efficient spline interpolation curve modeling", ***The Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing,*** Splendor Kaohsiung, Taiwan, 59-62 (2007).

26. İnternet: MindProd, “Floating - Point: JavaGlossary”, **http://mindprod. com/jgloss/floatingpo int .html,** (2011).

27. İnternet: IBM, “Red Books”, **http://www.redbooks.ibm .com/abstracts /sg242216.html,** (2011).

28. Kızılören, T., “Java ve Java Teknolojileri”, ***Kodlab Yayıncılık***, İstanbul, Türkiye, 303-340, 489-532 (2010).

29. İnternet: Oracle, “Swing”, **http://docs.oracle.com/javase/1.4.2/docs/api/javax/ swing/pac kage-summary.html,** (2012).

30. İnternet : Cellosoft, “Jtablet”, **http://jtablet.cellosoft.com/develop.html**, (2012).

31. İnternet: Oracle, “Java Persistence Api”, **http://www.oracle.com /technetwork/java/javaee/tech/persistence-jsp-140049.html,** (2012).

32. İnternet: Hacettepe Üniversitesi, “Java Persistence Api” **http://belgeler. cs.hacettepe .edu.tr/yayinlar/eski/JPA.pdf**, (2012).

33. İnternet: DZone JAVALOBY, “Is Hibernate the best choice”, **http://java.dzone.com/news/hibernate-best-choice**, (2012).

34. İnternet: CandidJava, “Hibernate Introduction”, **http://candidjava.com /hibernate-introduction**, (2012).

35. İnternet: Hibernate, “Hibernate”, **http://www.hibernate.org/**, (2012).

36. İnternet: Jboss, “Hibernate Referance Documentation”, **http://docs.jboss.org /hibernate/core/3.6/reference/en-US/pdf/hibernate_reference.pdf,** (2012).

37. İnternet: Hacettepe Üniversitesi, “Maven Proje Yönetim Aracı”, **http://belgeler.cs.hacettepe.edu.tr/yayinlar/eski/Maven2.pdf**, (2012).

38. İnternet: Apache, “Introduction to the Build Lifecycle”, **http://maven.apache. org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Referen ce** , (2012).

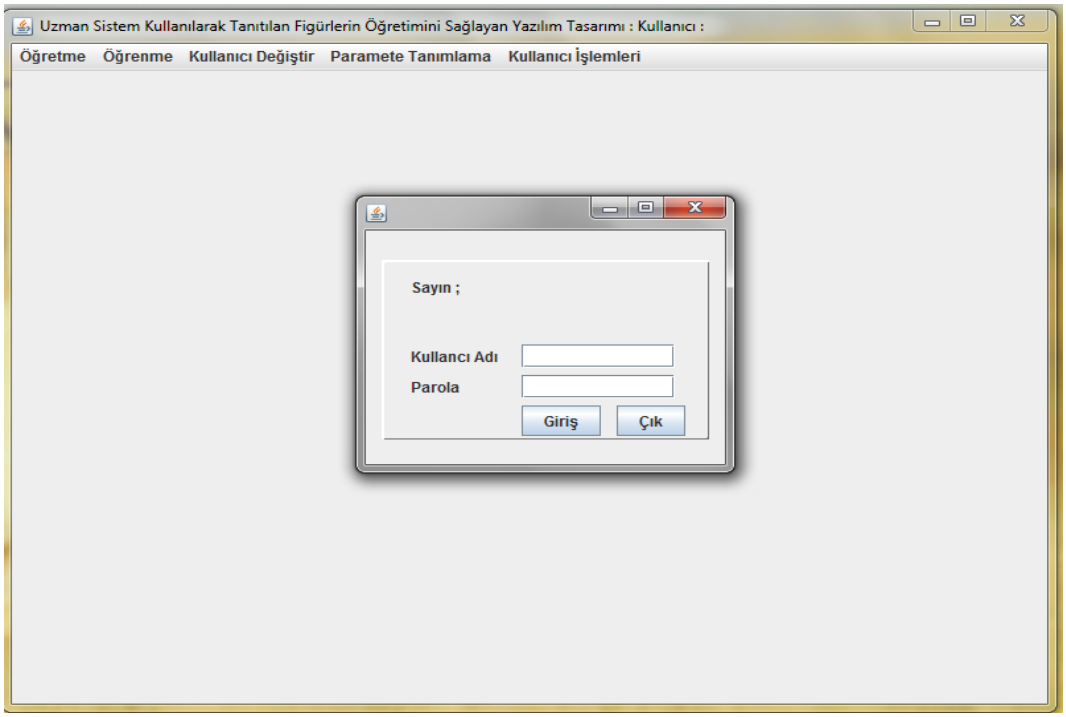
39. İnternet: Mysql “Mysql”, **http://www.mysql.com/**, (2012).

40. Şen, B., Uçar, E. ve Delen, D., “Predicting and analyzing secondary education placement-test scores: a data mining approach”, ***Expert Systems with Applications, Elsevier Publishing***, 39 (1): 9468–9476 (2012).

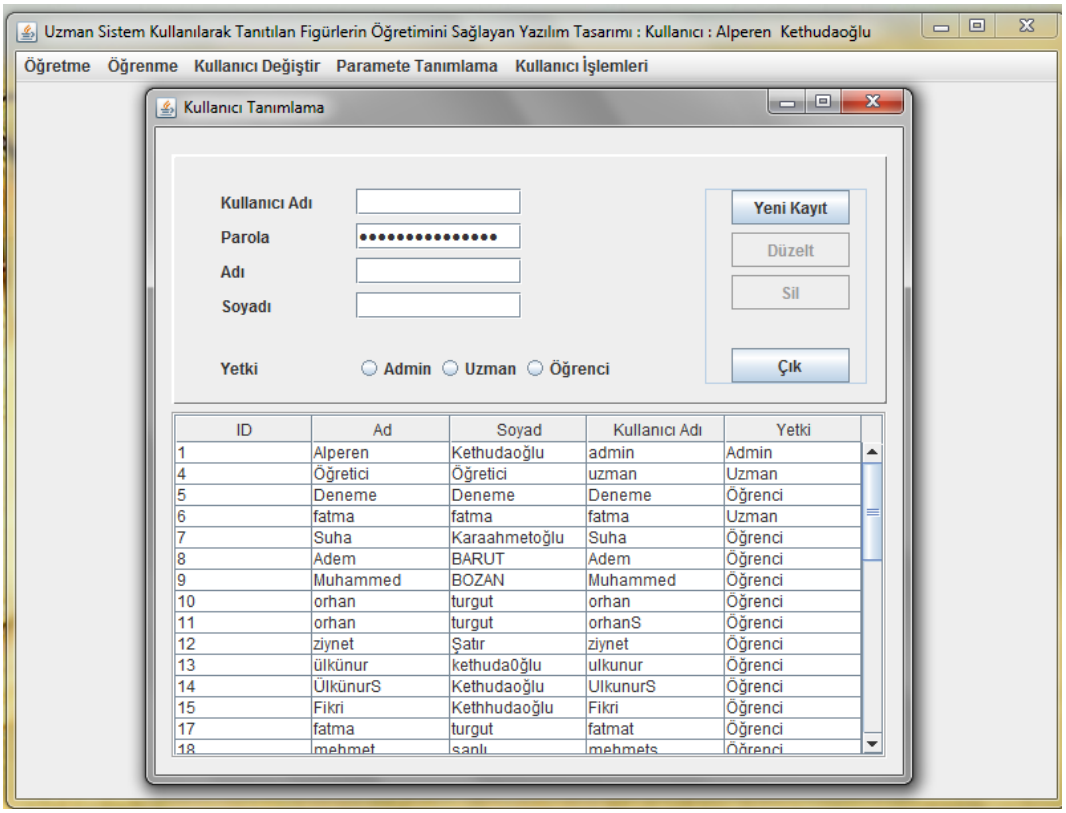
41. Kethudaoğlu, A. ve Şen, B., “Sayısal kalem (dijital pen) ile gerçekleştirilen çizimlerde giriş verisindeki hataların azaltılması”, ***Akademik Bilişim Konferansı***, Uşak, Türkiye, 1-11 (2012).

42. Numanoğlu G. ve Şen, B., "Bilgisayar ve öğretim teknolojileri eğitimi bölümü öğrencilerinin öğrenme stilleri", ***Ahi Evran Üniversitesi Kırşehir Eğitim Fakültesi Dergisi***, 2 (8): 128-149 (2007).

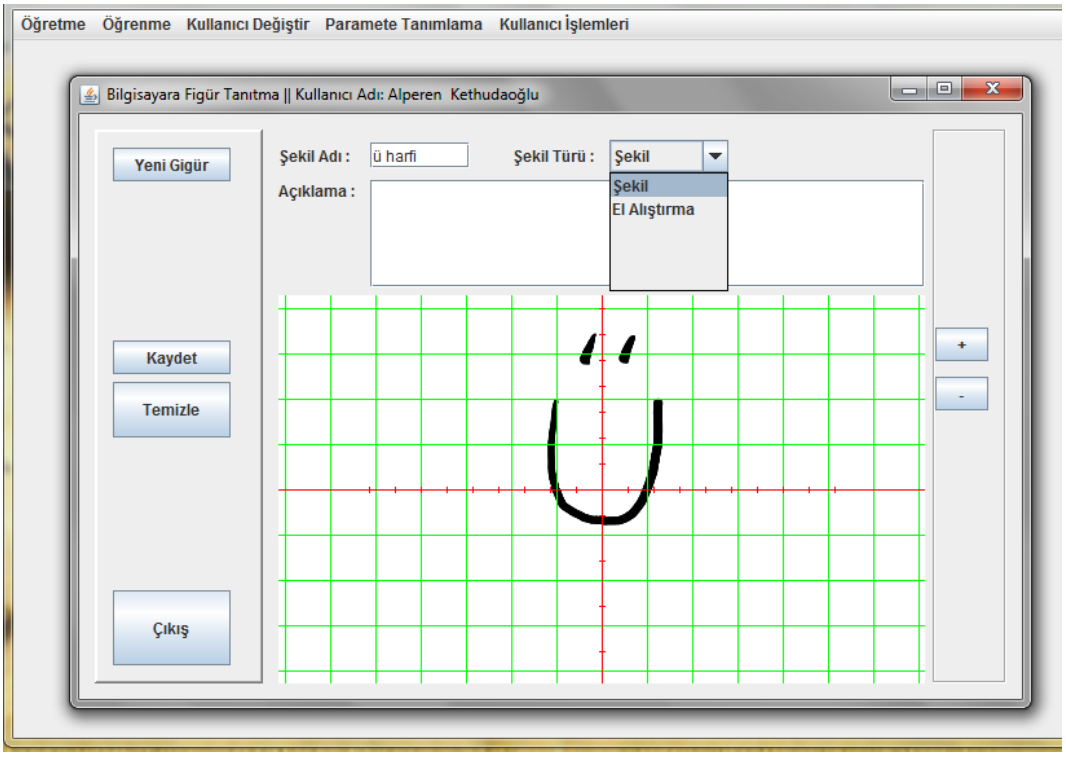
EK AÇIKLAMALAR A.
GELİŞTİRLEN UYGULAMANIN EKRAN GÖRÜNTÜLERİ



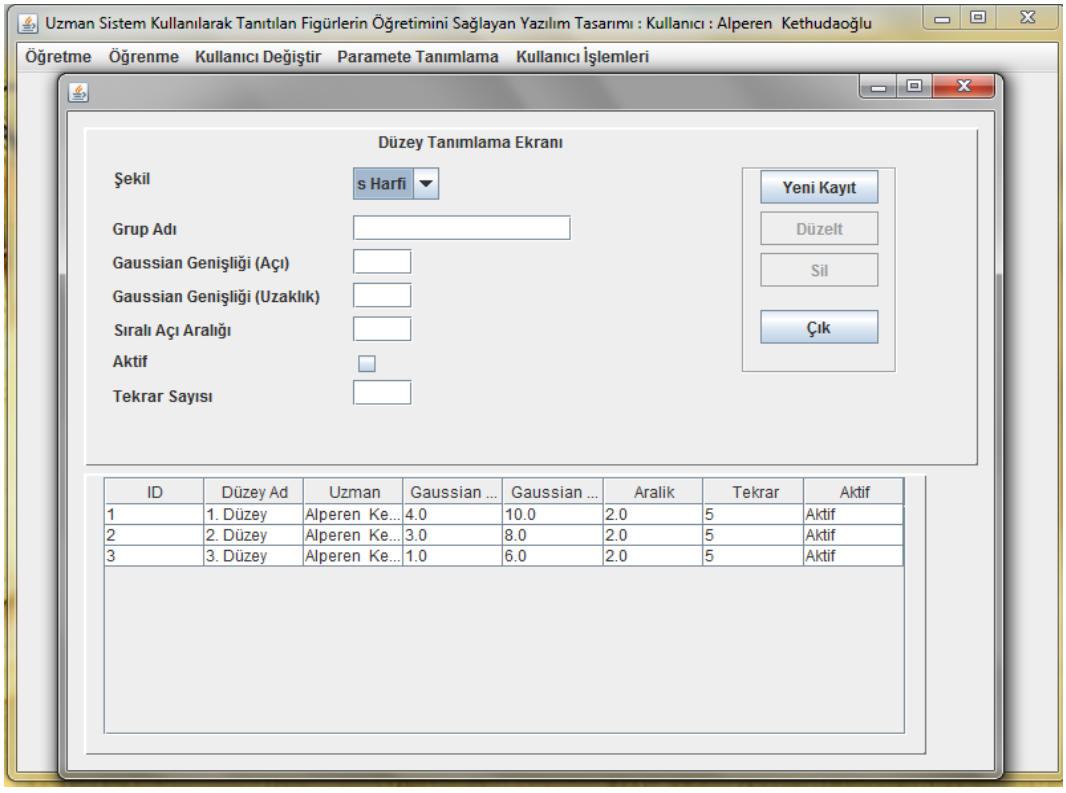
Şekil EK.A.1. Kullanıcı girişi ve kullanıcı değiştirme ekranı.



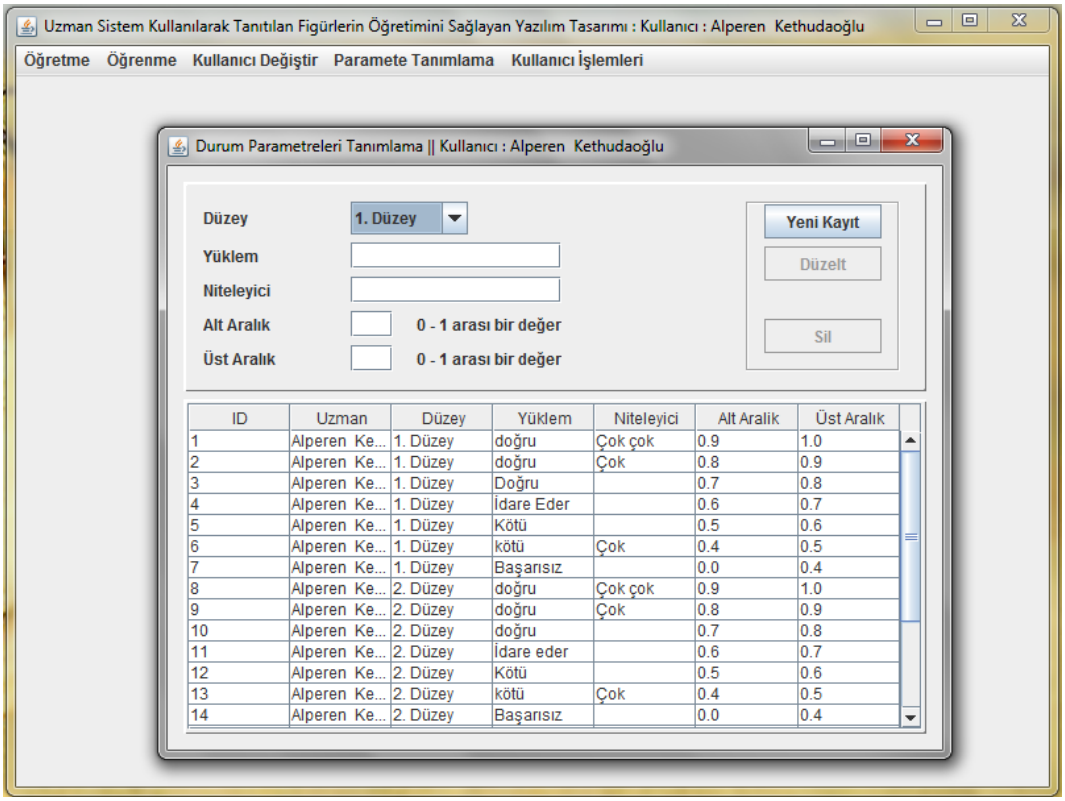
Şekil EK.A.2. Kullanıcı tanımlama ekranı.



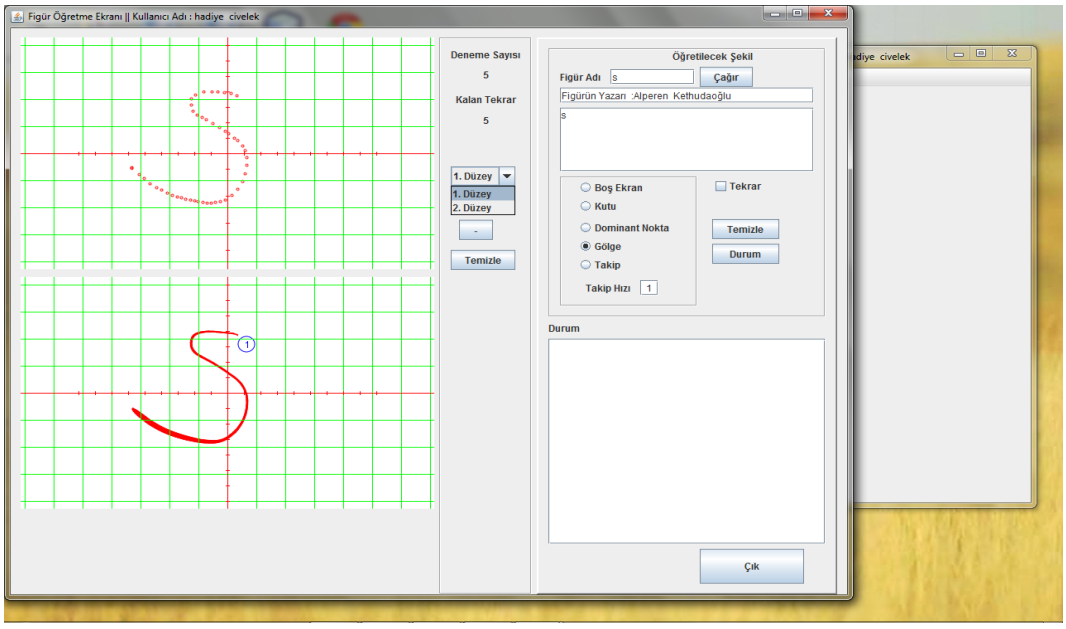
Şekil EK.A.3.Bilgisayara figür tanıtma ekranı.



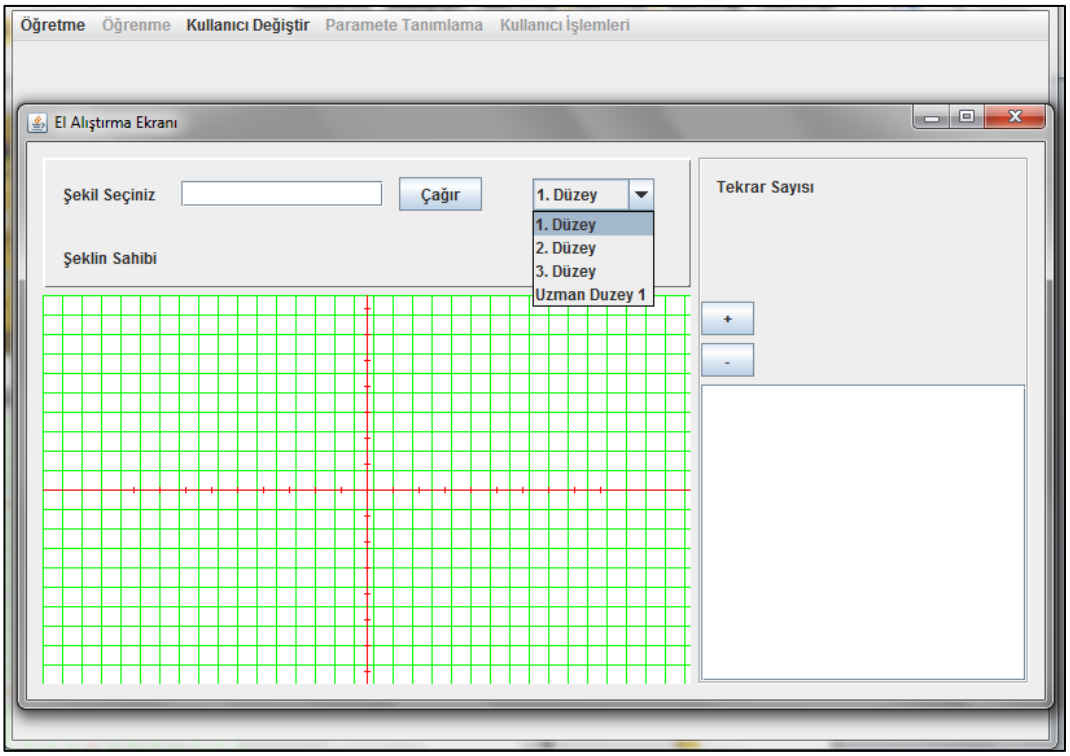
Şekil EK.A.4. Düzey tanımlama ekranı.



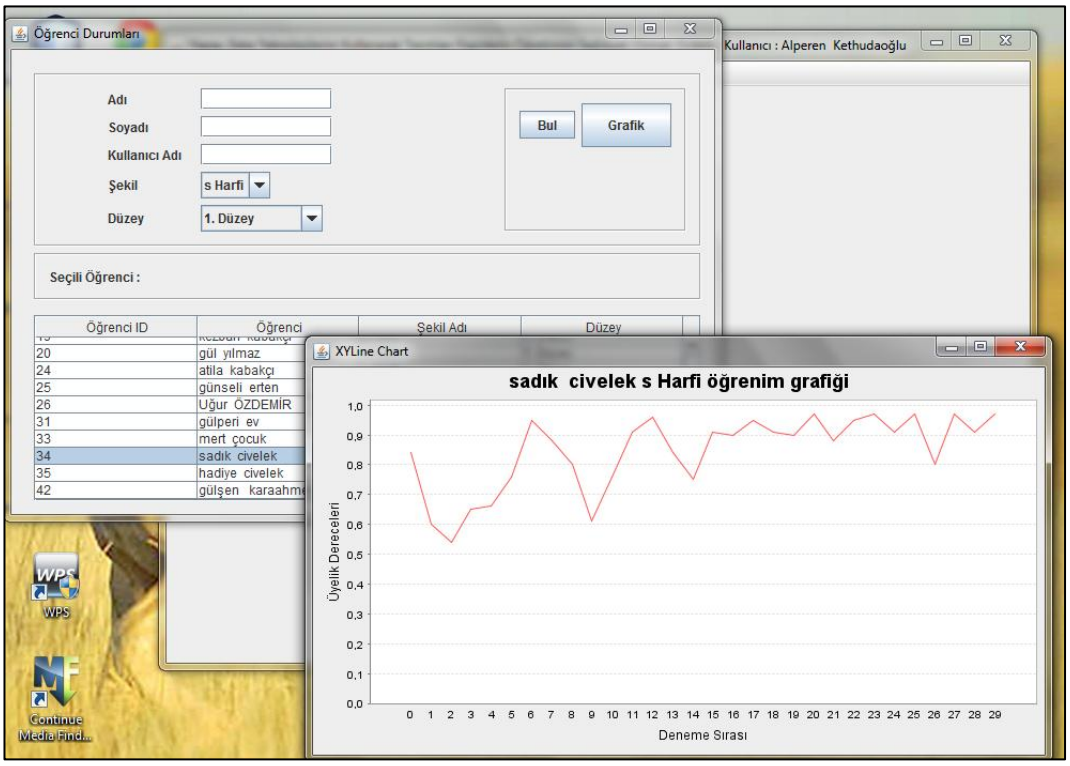
Şekil EK.A.5. Durum parametreleri tanımlama ekranı.



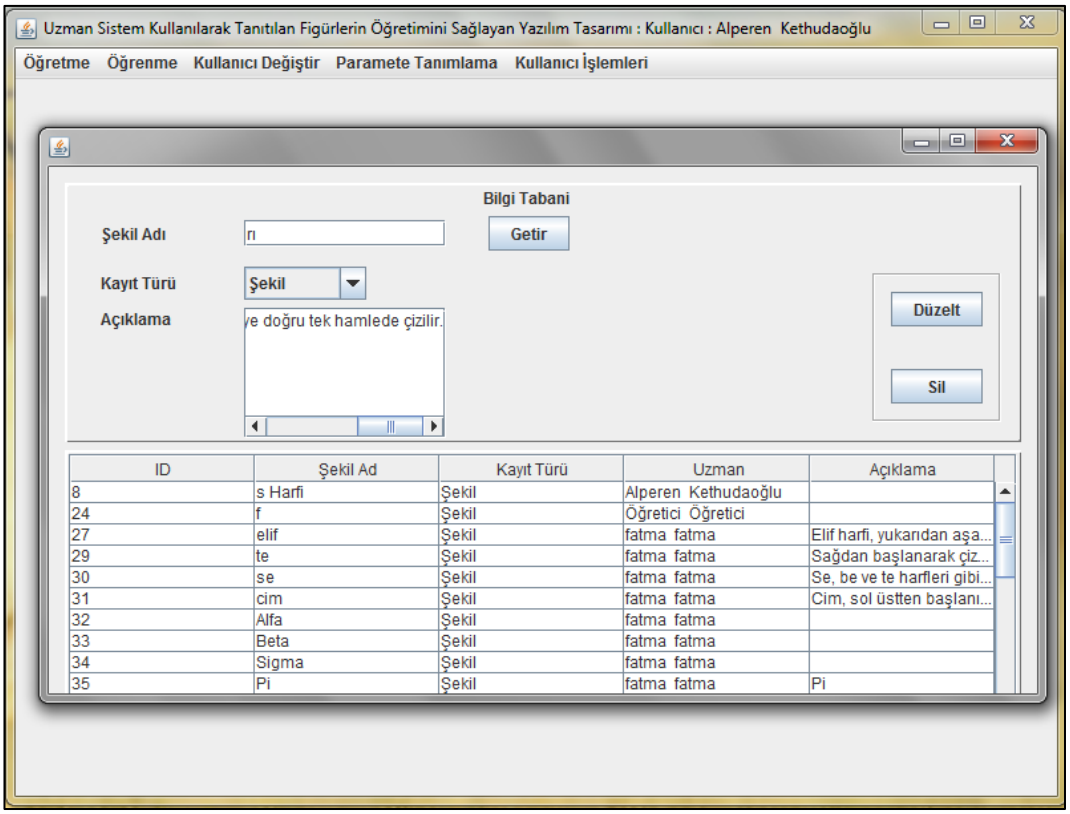
Şekil EK.A.6. Figür öğretimi ekranı.



Şekil EK.A.7. El alıştırma ekranı.



Şekil EK.A.8. Öğrenci başarı durumlarını gösteren ekran.



Şekil EK.A.9. Bilgi tabanında var olan şekilleri gösteren ekran.

Şekil EK.A.10. Yazılımın geliştirildiği sayısal kalem ve grafik tablet aracı.

ÖZGEÇMİŞ

Alperen KETHUDAOĞLU 1980 yılında Artvin'in Arhavi ilçesinde doğdu; ilk ve orta öğrenimini Kastamonu'da tamamladı. Taşköprü Sağlık Meslek Lisesi Sağlık Memurluğu Bölümü'nden mezun oldu. 2002 yılında Anadolu Üniversitesi Eskişehir Meslek Yüksekokulu'ndan mezun oldu. 2 yıl Akgün Yazılım Limited Şirketinde Yazılım Destek Kastamonu bölge sorumlusu olarak çalıştı. 2004 yılında Selçuk Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Sistemleri Öğretmenliği bölümüne başladı ve 2008 yılında mezun oldu. 2005 yılında Selçuk Üniversitesi Bilgi İşlem Daire başkanlığında Devlet görevine başladı ve 2009 yılından beri Kastamonu Üniversitesi Bilgi İşlem Daire Başkanlığında Yazılımcı Geliştirme gurubu bünyesinde görev yapmaktadır. Seçmeli Ders Seçim Sitesi, Ayniyat programı, Akademik Bilgi Sistemi, Kastamonu Üniversitesi Web Sayfaları, Köy-Koop Bölge Birlik Otomasyonu gibi birçok projede proje yürütücüsü ve yazılım geliştiricisi olarak görev yaptı. Halen Kastamonu Üniversitesi Bilgi İşlem Daire Başkanlığında Üniversite Web Sayfaları ve Personel Otomasyonu proje yürütücüsü ve geliştiricisi görevini yürütmektedir.

ADRES BİLGİLERİ

Adres : Kastamonu Üniversitesi
Bilgi İşlem Daire Başkanlığı
Kuzeykent Mahallesi / KASTAMONU

Tel : (505) 793 81 23
E-posta : kethuda37@hotmail.com