EL ÇİZİMLERİNİN BİYOMETRİK BİLGİYLE TANINMASI İÇİN İYİLEŞTİRİLMİŞ BİR YÖNTEM GELİŞTİRİLMESİ

Bilal Bektaş
131450201

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Doktora Programı
Danışman: Doç Dr. Turgay Tugay Bilgin

İstanbul
T.C. Maltepe Üniversitesi
Fen Bilimleri Enstitüsü
Ağustos, 2019

JÜRİ VE ENSTİTÜ ONAYI

BİLAL BEKTAŞ'ın "El Çizimlerinin Biyometrik Bilgiyle Tanınması İçin İyileştirilmiş Bir Yöntem Geliştirilmesi" başlıklı tezi 25.09.2019 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Maltepe Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği" nin ilgili maddeleri uyarınca Bilgisayar Mühendisliği Anabilim Dalı ~~Yüksek Lisans~~/Doktora tezi oy birliğiyle/~~oy çokluğuyla~~, başarılı/~~başarısız~~ olarak kabul edilmiştir.

	Unvanı. Adı ve Soyadı	İmza
Üye (Tez Danışmanı)	Doç. Dr. Turgay Tugay BİLGİN Bursa Teknik Üniversitesi	.....
Üye	Prof. Dr. Emin Murat ESİN	.....
Üye	Doç. Dr. Sinan APAK	.....
Üye	Dr. Öğr. Üyesi Mehmet Ali Aksoy TÜYSÜZ	.....
Üye	Dr. Öğr. Üyesi H. F. Selim BAYRAKLI Milli Savunma Üniversitesi	.....

Prof. Dr. İlter BÜYÜKDIĞAN
Enstitü Müdürü

Revizyon Takip Tablosu

REVİZYON NO	TARİH	AÇIKLAMA
00	01.03.2018	İlk yayın.

ETİK İLKE VE KURALLARA UYUM BEYANI

25/09/2019

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarından bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilmeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; çalışmanın Maltepe Üniversitesinde kullanılan "bilimsel intihal tespit programı" ile tarandığını ve öngörülen standartları karşıladığını beyan ederim.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

Bilal Bektaş

El Çizimlerinin Biyometrik Bilgiyle Tanınması için İyileştirilmiş bir Yöntem Geliştirilmesi

ORIJINALLIK RAPORU

%2	%1	%1	%1
BENZERLIK ENDEKSI	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCI ÖDEVLERI

BIRINCIL KAYNAKLAR

1	tr.wikipedia.org İnternet Kaynağı	<%1
2	lixinlijing.blogspot.com İnternet Kaynağı	<%1
3	"Learning and Intelligent Optimization", Springer Nature, 2010 Yayın	<%1
4	Calik, Nurullah, Lutfiye Durak Ata, Ahmet Serbes, Bulent Bolat, and Emrah Yavuz. "Performance analysis of feature extraction methods in indoor sound classification", 2015 23nd Signal Processing and Communications Applications Conference (SIU), 2015. Yayın	<%1
5	Costa, L.. "Cohomological characterization of vector bundles on multiprojective spaces", Journal of Algebra, 20051201 Yayın	<%1

28.08.2019
Tarafımdan Görülmüştür.
Doç.Dr. Turgay Tugay Bilgin
Danışman

TEŞEKKÜR

Tez çalışmam süresi boyunca değerli zamanını ve tecrübesini paylaşarak destek olan ve yardımlarını esirgemeyen danışmanım Sayın Doç. Dr. Turgay Tugay Bilgin'e, tez izleme sunumlarında olumlu ve yapıcı eleştiriler ile ilerlememde katkıları bulunan Sayın Prof. Dr. Emin Murat Esin'e ve Sayın Doç. Dr. Sinan Apak'a teşekkür ederim.

Eğitim hayatım boyunca her türlü desteği eksik etmeyen, her zaman sevgi ve sabırlarıyla yanımda bulunan, değerli zamanlarını paylaşan ve bugünlere gelmemi sağlayan değerli aileme en içten teşekkürlerimi sunarım.

Bilal Bektaş

Ağustos 2019

ÖZ

EL ÇİZİMLERİNİN BİYOMETRİK BİLGİYLE TANINMASI İÇİN İYİLEŞTİRİLMİŞ BİR YÖNTEM GELİŞTİRİLMESİ

Bilal Bektaş
Doktora Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Doktora Programı
Danışman: Doç. Dr. Turgay Tugay Bilgin
Maltepe Üniversitesi Fen Bilimleri Enstitüsü, 2019

Bu tez çalışmasında, el çizimlerinin biyometrik bilgiyle tanımlanması için yeni bir yöntem önerilmektedir. Veri toplama işlemi tablet üzerine dijital kalem ile sağlanmıştır. Gönüllülerden toplanan yazılar, rakamlar ve basit şekil çizimleri veri seti olarak kullanılmıştır. Bu veri setinden polyline (bir çizimde el kaldırmadan çizilen her bir alt çizim) adetleri, çizim açıları, çizim hızı ve çizgisel basınç verileri gibi biyometrik bilgi çıkarımı gerçekleştirilmiştir. Yazım esnasında el titremesi sebebiyle oluşan sıçramaları düzeltmek için sıfır faz filtresi kullanılmıştır. Çizim algılama için K-en yakın komşu algoritması (K-NN) kullanılmıştır. Her bir çizim verisi bir zaman serisi olduğu için veri nesneleri arası benzerliği bulmak için Dinamik Zaman Atlaması (DTW) yöntemi kullanılmıştır. Test işlemleri "Genel Çizim Tipleri" ve "İmza" olmak üzere iki ana başlık altında toplanmaktadır. Öncelikle sayılar, rakamlar, çeşitli karakterler ve basit şekiller içeren heterojen bir veri seti kullanılarak algılama testi gerçekleştirilmiştir. Sonrasında, aynı türden çizimler içeren homojen bir veri seti üzerinde testler gerçekleştirilmiştir. İmza ile ilgili test işlemleri "Gerçek İmza" ve "Sahte İmza" olmak üzere iki alt başlık altında toplanmıştır. Özellikle, biyometrik verilerin daha belirgin bir şekilde ortaya çıktığı imza gibi kişiye özel çizimlerde biyometrik bilginin algılama doğruluğunu önemli oranda arttırdığı görülmüştür. Bu bilgiler ışığında, çizen kişiyi tespit etmede en fazla bilgi sağlayan biyometrik bileşenlerin çizimdeki polyline sayısı ve çizim açıları olduğu görülmüştür. Bunların ardından sırasıyla hız ve basınç bilgileri gelmektedir. Test sonuçları, kullanıcının kişisel biyometrik bilgilerinin çizim tanımlamaya %3 ile %11 arasında katkıda bulunduğunu göstermiştir.

Anahtar Sözcükler: Çizim Tanıma; Biyometrik Bilgi; Hız; Basınç; İmza; DTW; kNN

ABSTRACT

DEVELOPING AN IMPROVED METHOD FOR RECOGNITION OF SKETCH WITH BIOMETRIC INFORMATION

Bilal Bektaş
PhD Thesis
Department of Computer Engineering
Computer Engineering Programme
Advisor: Assoc. Prof. Dr. Turgay Tugay Bilgin
Maltepe University Science and Engineering Graduate School, 2019

In this thesis, a new method for recognition of sketch with biometric information is proposed. For data collection, a tablet and a digital pen have been used. Letters, numbers and basic figure sketches, which are collected from volunteers, have been used for data set. From this data set, biometric information has been extracted such as sketch angle, sketch velocity, sketch pressure data and number of polylines (polylines are sub sketches begins with the pen touches the surface and continues until pen lifted). A zero phase filter has been used for correcting spikes which occur because of hand tremors during sketching. K-Nearest Neighbor (K-NN) algorithm has been used for sketch recognition. Dynamic Time Warping (DTW) algorithm has been used in order to find similarity between data objects because every sketch data is transferred into a time series format. Experiments are carried out under two main headings as “General Drawing Types” and “Signature”. Firstly, the recognition tests were carried out using a heterogeneous data set containing numbers, various characters and simple figures. Likewise, tests were performed on a homogeneous data set containing sketching of the same type. The test procedures regarding the signature are grouped under two sub-headings as “Real Signature” and “Fake Signature". Especially, it has been observed that biometric information significantly increases detection accuracy. As a result, it has been observed that number of polylines and sketch angles are the major biometric components to identify sketches of a person. The speed and pressure information are minor components for identification. Test results show that the user's personal biometric information increases the accuracy of sketch recognition from 3% to 11% depending to the complexity of the sketch.

Keywords: Sketch Recognition; Biometric Information; Velocity; Pressure; Signature; DTW; kNN

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

ALGORİTMALAR LİSTESİ

DENKLEMLER LİSTESİ

KISALTMALAR

Kısaltma	İngilizcesi	Türkçesi
DTW	Dynamic Time Warping	Dinamik Zaman Bükme
FAR	False Acceptance Rate	Hatalı Kabul Oranı
FRR	False Rejection Rate	Hatalı Red Oranı
JSON	JavaScript Object Notation	JavaScript Nesnesi Gösterimi
K-NN	K - Nearest Neighbors	K En Yakın Komşu
NoSQL	Not Only SQL	SQL ve Daha Fazlası
SQL	Structured Query Language	Yapılandırılmış Sorgu Dili
SVG	Scalable Vector Graphics	Ölçeklenebilir Vektör Grafikleri
XML	eXtensible Markup Language	Genişletilebilir İşaretleme Dili
W3C	World Wide Web Consortium	Dünya Çapında Ağ Konsorsiyumu

ÖZGEÇMİŞ

Bilal Bektaş

Bilgisayar Mühendisliği Anabilim Dalı

Eğitim

Derece	*Yıl*	*Üniversite, Enstitü, Anabilim/Anasanat Dalı*
Yüksek Ls.	2014	Boğaziçi Üniversitesi, Fen Bilimler Enstitüsü Hesaplamalı Bilim ve Mühendislik Anabilim Dalı
Lisans	2008	İstanbul Teknik Üniversitesi, Fen Edebiyat Fakültesi Matematik Mühendisliği Anabilim Dalı
Lise	2003	Hüseyin Avni Sözen Anadolu Lisesi

İş/İstihdam

Yıl	*Görev*
2018 –	Büyük Veri Danışmanı, OBASE
2018 – 18	Kıdemli İş Zekası Danışmanı, IBSS Danışmanlık
2016 – 17	Kıdemli İş Zekası Danışmanı, Itelligence
2012 – 16	İş Zekası Analisti, TÜBİTAK BİLGEM
2010 – 12	İş Zekası Danışmanı, FIT Danışmanlık
2008 – 10	Proje Asistanı, Boğaziçi Üniversitesi

Yayınlar ve Diğer Bilimsel/Sanatsal Faaliyetler

C. Özturan, B. Bektaş, M. Yılmazer, "Seismic Data Server Application Service for SEEGRID Seismology Virtual Organization", *Earth Science Informatics*, Vol 3, No 4, p. 219-228, 2010.

M. Yılmazer, B. Bektaş, M. Kozlovszky, "Gridification of regional Fault Plain Solution (FPS)", *Earth Science Informatics*, Vol 3, No 4, p. 234-245, 2010.

C. Özturan, B. Bektaş, M. Yılmazer, "Seismic Data Server Application Service and Web Interface", *SEE-GRID-SCI User Forum 2009*, Boğaziçi Üniversitesi, İstanbul, Türkiye, 9-10 Aralık 2009, pp. 187-192.

M. Yılmazer, B. Bektaş, M. Kozlovszky, "Gridification of Fault Plain Solution (FPS) Application", *SEE-GRID-SCI User Forum 2009*, Boğaziçi Üniversitesi, İstanbul, Türkiye, 9-10 Aralık 2009, pp. 181-185.

M. Yılmazer, B. Bektaş, C. Özturan, M. S. Geden, R. Arıkan , "Parallelization of Earthquake Location Finding (ELF) Application", *SEE-GRID-SCI User Forum 2009*, Boğaziçi Üniversitesi, İstanbul, Türkiye, 9-10 Aralık 2009, pp. 177-180.

C. Ozturan, B. Bektas, M. Yilmazer, C. Sener, "Challenges Faced in Building a Virtual Organization for Seismology in Southeastern Europe", *E-Challenges Conference 2009*, İstanbul, Türkiye, 21 - 23 Ekim 2009.

Á. Balaskó , M. Kozlovszky, B. Süle, B. Bektaş , "Enabling Numerical Modeling of Mantle Convection on the Grid", *(MIPRO 2009) 32nd INTERNATIONAL CONVENTION MIPRO and 6th ALADIN (ALPEADRIA UNIVERSITIES INITIATIVE)*, Rijeka, Hırvatistan, Mayıs 2009.

M. Yılmazer, C. Özturan, M. S. Geden, R. Arıkan, B. Bektaş, "Deprem Merkezi Bulma Uygulamasının JDL İş Akışları ile Paralelleştirilmesi", *(BASARIM09) I. Ulusal Yüksek Başarım ve Grid Konferansı*, METU, Ankara, Türkiye, Nisan 2009, pp. 49-52.

BÖLÜM 1. GİRİŞ

Teknolojinin hızlı bir şekilde ilerlemesi ile beraber dijital dünya daha çok hayatımızın içinde var olmaya başlamıştır. Bu dijitalleşme neticesinde kullanılan defterlerin yerini dokunmatik yüzeyli telefonlar ve tabletler, kurşun ve mürekkepli kalemlerin yerini dijital kalemler almaya başlamıştır. Kâğıt üzerine yapılan çizimlerin tanınması veya bu çizimlerin kategorize edilmesi, çizimlerin detaylarına bakılarak uzmanlar tarafından yapılabilmektedir ya da çizimlerin dijital ortama çevrilmesi ile dijital ürünler üzerinden yapılabilmektedir. Dokunmatik yüzeye sahip olan teknolojik aletlerin üzerinde yapılan çizimler için de benzer çözümler üretilebilmektedir.

Bu tez çalışması içerisinde, dokunmatik ekran üzerine yapılan el çiziminin tanımlanabilmesi için çizimdeki biyometrik bilgileri kullanan iyileştirilmiş bir yöntem geliştirilmiştir.

1.1 Problem

Bir çizimin dijital olarak tanınması, karmaşık bir işlem veya çözülmesi zor olan bir problem olması sebebiyle bu konu hakkında birçok çalışma yapılmıştır. Bu çalışmalar genel olarak ikiye ayrılmaktadır. Birincisi, çizim yapıldıktan sonra çizimin dijitalleştirmesi ile yapılan çalışmalardır. Örneğin, kâğıt üzerine yapılan çizim tamamlandıktan sonra çizim taranarak fotoğrafik olarak dijitalleştirilmektedir. Sonrasında optik karakter tanıma (Optical Character Recognition – OCR) yöntemleri ile çizim ile ilgili tanımlama yapılabilmektedir. İkincisi, çizim yapılırken çizim ile ilgili bilgilerin dijital hale getirilerek kaydedilmesidir. Örneğin; dokunmatik bir tablet üzerine dijital bir kalem ile yapılan çizimin her bir milisaniyesinde, çizimin tablet üzerindeki koordinat bilgisi ve basınç bilgisi zaman bazlı olarak alınıp kaydedilebilmektedir. Alınan bu bilgilere göre de çizim ile ilgili tanımlamalar yapılabilmektedir.

Çizimin tanınması konusunda diğer önemli bir konu ise yapılan çizimin gerçekten çizimi yapan kişiye ait olup olmamasıdır. Sadece görsel olarak yapılan karşılaştırmalarda çizimin o kişiye ait olduğu tespit edilebilmektedir ama benzer çizimi başka birisi de taklit edebilmektedir.

1.2 Amaç

Dokunmatik yüzey üzerine yapılan çizim üzerinden çizimi yapan kişi ile ilgili çeşitli biyometrik bilgiler elde edilebilmektedir ve bu bilgilere göre de tanımlama işlemi yapılabilmektedir. Bu tanımlama çeşitlerinden bir tanesi de yapılan çizimin gerçekten o kişiye ait olup olmamasının tespitidir.

Bu çalışma içerisinde, çizimden alınan biyometrik bilgilerin kullanılmasıyla çizim tanımlanmasının daha iyi bir konuma getirilmesi hedeflenmiştir. Ayrıca kimlik doğrulama işlemlerinde önemli bir yeri bulunan sahte imzaların ya da çizimlerin engellenmesi özelliğini sağlayan ve sahte çizimler engellenirken de çizimi yapan kişinin doğru bir şekilde tanınmasını sağlayan bir kimlik doğrulama yönteminin geliştirilmesi hedeflenmiştir.

1.3 Tezin Yapısı

Bu tez çalışması beş adet bölümden oluşmaktadır. Birinci bölüm, tezin amacını ve literatüre katkısını içermektedir. İkinci bölüm, literatür araştırması ile ilgili detaylı bilgiler vermektedir. Üçüncü bölüm, önerilen yöntemin tüm geliştirmelerini içermektedir. Dördüncü bölümde, önerilen yöntemin deneysel sonuçları değerlendirilmektedir. Beşinci ve son bölümde, yapılan çalışmanın sonuçları verilmektedir.

BÖLÜM 2. LİTERATÜR ARAŞTIRMASI

Literatür araştırmasının detayları tanıma teknikleri, kimlik doğrulama yöntemleri, grafiksel doğrulama ve el çizim temelli parolaların tanınması ana başlıkları altında verilecektir.

2.1 Çizim Tanıma Teknikleri

Çizim tanıma teknikleri, çizimin okunma biçimi ve çizimin tanınma yöntemi başlıkları altında açıklanacaktır. Çizimin okunma biçimi başlığı altında, çizimin hangi zamanda ve hangi yöntem ile dijital hale getirilebildiği konusunda bilgi verilecektir. Çizimin tanınma yöntemi başlığı altında, çizimin bulunduğu duruma göre nasıl anlamlandırıldığı ve bu anlamlandırma yapılırken hangi yöntemlerin kullanılabildiği ile ilgili çeşitli bilgiler verilecektir.

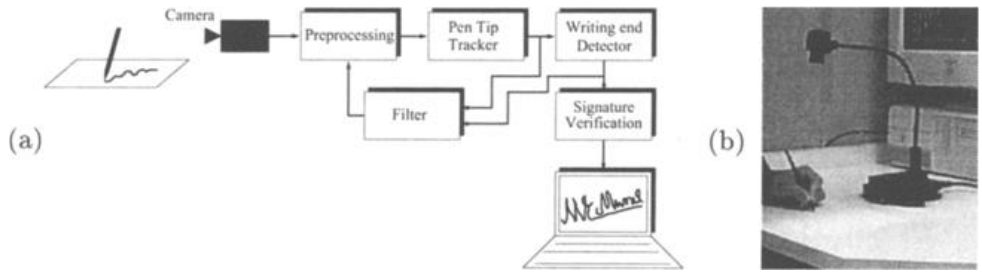
2.1.1 Çizimin Okunma Biçimi

El çizimleri, parmak izi veya el damar haritası gibi her insanın kendine özgü biyometrik bilgilerinden biridir. Dijital ortamda veya kâğıt gibi bir yüzeyde oluşturulan el çizimlerinin tanınması noktasında birçok çalışma yapılmıştır. Bilginin okunma biçimine göre tanıma teknikleri iki ana başlıktan oluşmaktadır [1].

- Statik teknikler (off-line/static techniques) : Çizim tamamlandıktan sonra, çizimin hızı ve sırası önem arz etmeksizin yapılan tanıma tekniği çeşididir. Genellikle bir tarayıcı veya bir kamera ile çizim sisteme aktarılır.
- Dinamik teknikler (on-line/dynamic techniques): Çizimin hızı, çizimin konum koordinatları, konumsal basınç değerleri, kalemi kaldırma ve kalemi yeniden indirme gibi bilgiler kullanılarak gerçekleştirilen tanıma tekniği çeşididir. Bu tür teknikler, dijital tabletler ve akıllı tahtalardan alınan verilerde kullanılabilir.

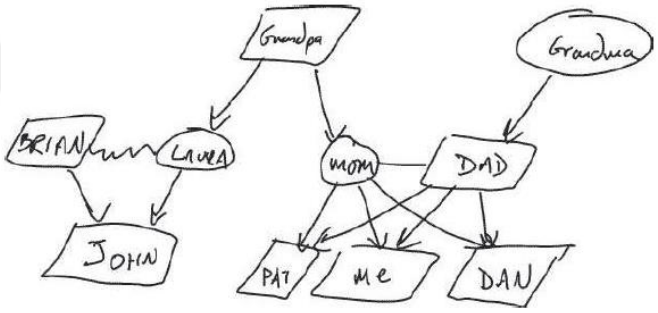
Statik tanıma tekniklerine örnek olarak, Munich ve Perona'nın [2] yaptığı çalışma verilebilir. Bu çalışmada, kalem ile düz bir kâğıda atılan imza kamera yolu ile dijital hale getirilir. Kalemin duruşu, kameranın açısı gibi bilgiler kullanılarak filtreleme işlemi

yapıldıktan sonra sisteme kaydedilir. Doğrulama sırasında da benzer şekilde kamera kullanılarak, sisteme kaydedilen imza ile karşılaştırma yapılır.

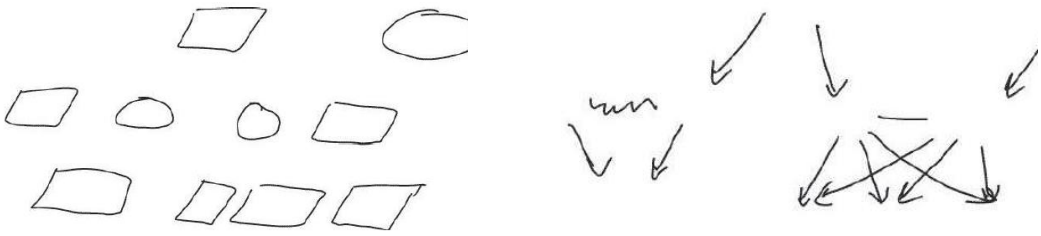


Şekil 2.1. İmza doğrulama sırasında yapılan işlemler [2]

Dinamik tanıma tekniklerine, Peterson'ın [3] doktora tezinde el ile çizilen bir çizim üzerindeki nesneleri gruplara ayırmak için yapmış olduğu çalışma örnek olarak verilebilir. Şekil 2.2'de, Tablet PC üzerinde el çizimi ile yapılmış örnek bir aile diyagramı ve bu aile diyagramının parçalanmış hali Şekil 2.3'de görülebilir.



Şekil 2.2. Örnek bir aile diyagramı çizimi [3]



(a) İnsan nesneleri (b) Bağlantı nesneleri

(c) Metin nesneleri

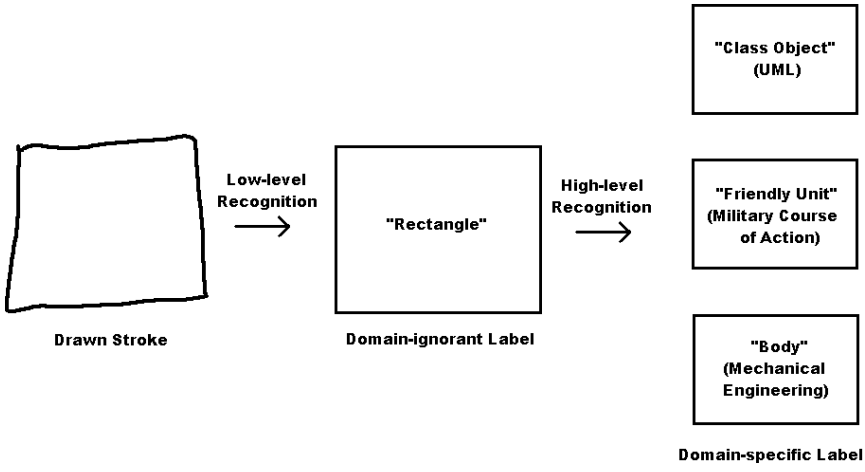
Şekil 2.3. Örnek aile diyagramının parçalanmış hali [3]

2.1.2 Çizimin Tanınma Yöntemi

Tanıma seviyelerine göre çizim tanınma yöntemleri iki ana başlık altında toplanabilir [4]:

- Üst seviye tanıma (High level recognition)
- Alt seviye tanıma (Low level recognition)

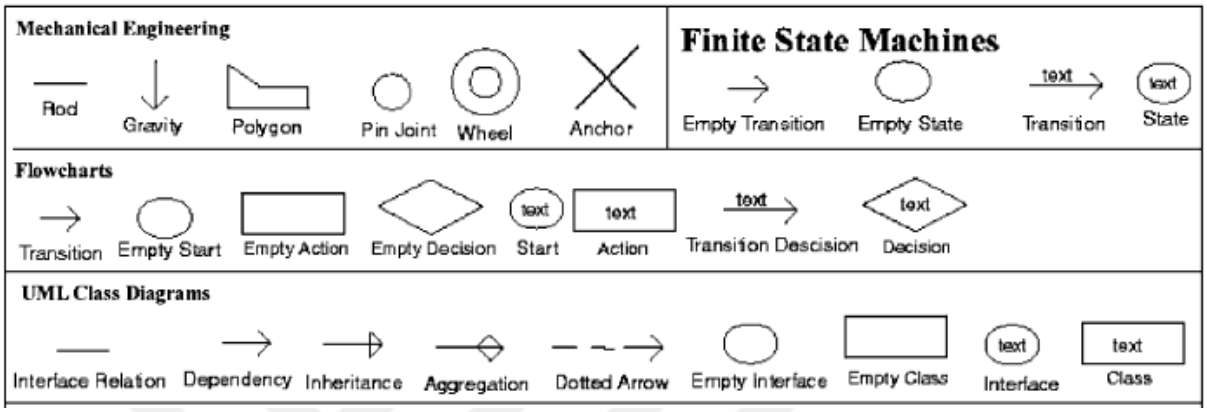
Şekil 2.4'de düşük düzey tanıma ile yüksek düzey arasındaki tanımanın farkı gösterilmektedir. Düşük düzey tanımada, çizilen nesnenin sadece bir dikdörtgen olduğu söylenirken; yüksek düzey tanıma sırasında bu dikdörtgene isimlendirilme yapılırken bulunmuş olduğu alana göre isimlendirme yapılmaktadır. Örnek olarak; makine mühendisliği alanında bu tanıma işlemi yapıldı ise, dikdörtgen vücut olarak isimlendirilmektedir.



Şekil 2.4. Düşük düzey tanıma ile yüksek düzey tanıma arasındaki fark [4]

2.1.2.1 Üst Seviye Tanıma

Üst düzey tanıma sistemlerine örnek olarak LADDER [5] algoritması verilebilir. Önceden etiketlenmiş şekiller ile çizimdeki şekiller kıyaslanarak çizilen nesneler sınıflandırılır. Şekil 2.5'de önceden tanımlanmış olan şekiller görülebilir.



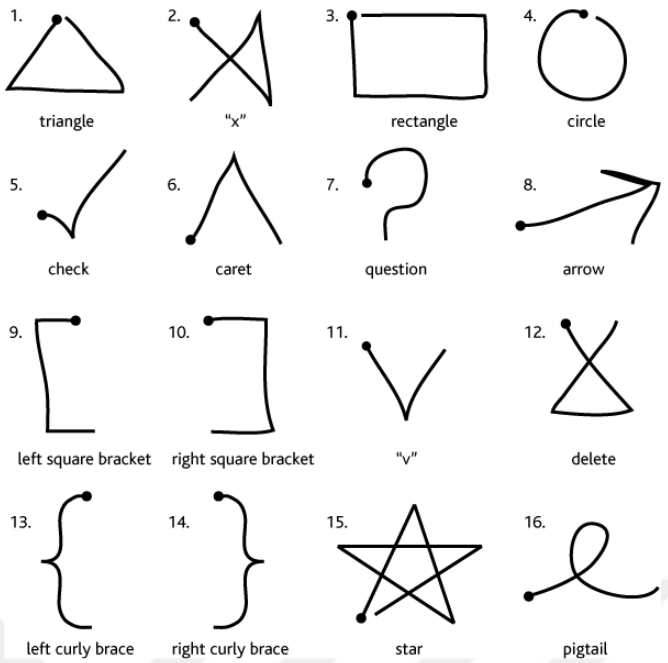
Şekil 2.5. Önceden tanımlanmış olan alan şekilleri [5]

2.1.2.2 Alt Seviye Tanıma

Alt seviye tanımada, çizilen nesnenin sadece bir dikdörtgen olduğu söylenirken; üst seviye tanımada ise bu dikdörtgen kullanım alanına göre etiketlenir. Örneğin; makine mühendisliği alanında tanıma yapılırken, "dikdörtgen" şekli "vücut" olarak etiketlendirilebilir. Alt seviye tanıma üç alt kategoriye ayrılabilir [4]:

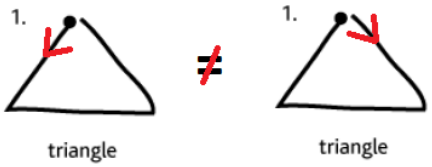
- Harekete dayalı tanıma (Motion-based Recognition)
- Görünüme dayalı tanıma (Appearance-based Recognition)
- Geometrik tabanlı tanıma (Geometric-based Recognition)

Harekete dayalı tanımada çizimin başlangıç noktası ve bitiş noktası önemlidir. Wobbrock ve arkadaşlarının [6] yapmış olduğu çalışmada Şekil 2.6'da görülen semboller ile çizim sırasındaki semboller karşılaştırılarak tanımlama yapılmaktadır. Bu yönteme göre, soldan sağa doğru çizilen bir üçgen tanımlanabilirken, kişi çizimi yaparken sağdan sola doğru çizim yaparsa bu çizim üçgen olarak tanımlanmamaktadır.



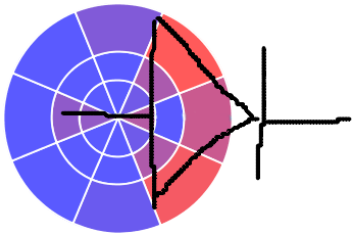
Şekil 2.6. Tahmin yapabilmek için önceden tanımlanmış olan semboller [6]

Şekil 2.7'de görüntü olarak benzeyen iki üçgen hareket dayalı tanımlama yapıldığı için ikisi de farklı üçgeni ifade etmektedir.



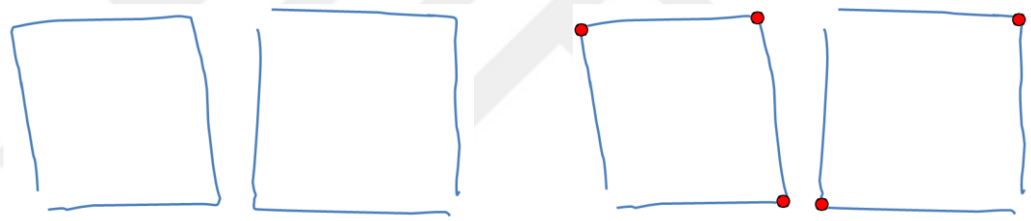
Şekil 2.7. Üçgenlerin çizim yönü

Görünüme dayalı tanımaya örnek olarak, Oltmans'ın [7] doktora tezindeki yaklaşım verilebilir. Oltmans çizimleri küçük parçalara bölerek ve Şekil 2.8'de görülen kendi oluşturmuş olduğu dairesel alanlar içerisinde bulunan noktaları sayarak histogram değerlerini çıkarmaktadır. Sonrasında çıkarttığı histogram değerlerini, önceden düzgün olarak verilmiş olan histogram değerleri ile kıyaslayarak bu küçük parçaların ne olduğuna karar vermektedir.



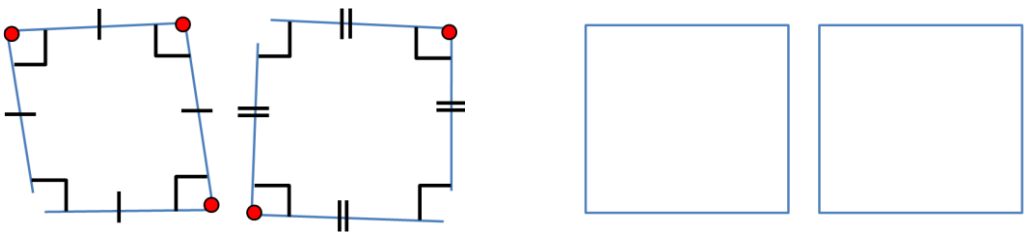
Şekil 2.8. Bir diyot sembolü üzerinden hesaplanan histograma örnek [7]

Geometrik tabanlı tanımaya örnek olarak, MergeCF [8] algoritması verilebilir. Şekil 2.9'da görüleceği üzere, algoritma ilk olarak çizimlerin köşe noktalarını bulmakta (b) ve bu bulunan noktalara göre çizgi sayısını (c) bulmaktadır. Bulunan çizgilerin boyutlarını geometrik olarak kıyaslayarak (c) , 4 adet çizginin eşit olması sebebiyle bu şekillere kare tanımlaması (d) yapmaktadır.



(a) Kullanıcı tarafından çizilmiş iki farklı kare

(b) Köşe noktaları



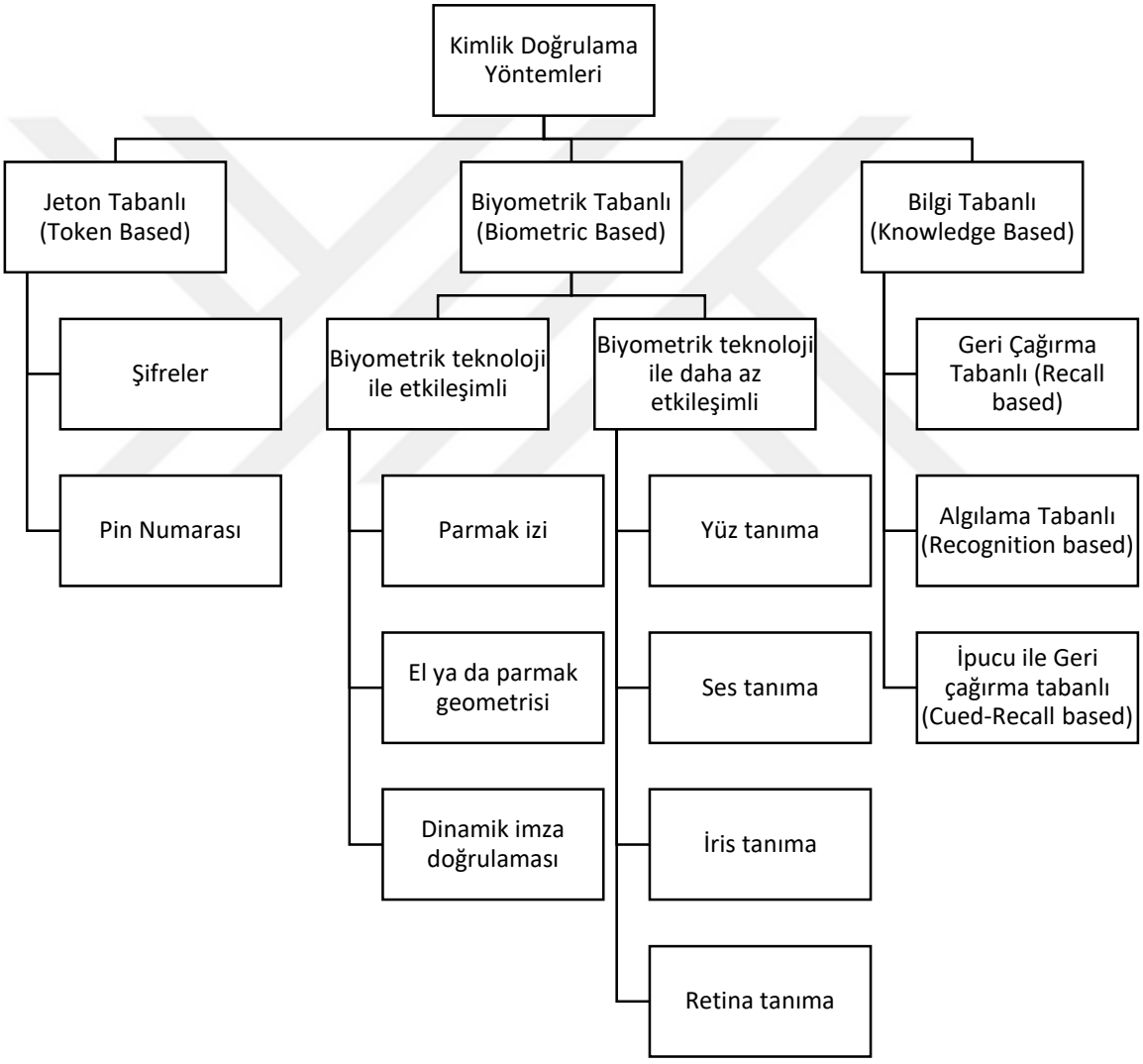
(c) Kare olması için çizgilerin aynı olması

(d) Her ikisinin de kare olarak tanımlanması

Şekil 2.9. MergeCF algoritmasının iki adet çizimi kare olarak algılamasına örnek [8]

2.2 Kimlik Doğrulama Yöntemleri

Yüzyıllar boyunca bilgi güvenliği önemli bir konu oldu ve güvenliği sağlamak için çeşitli araştırmalar ve geliştirmeler yapıldı. Bu araştırmalar ile bilginin doğru kişiye ulaştığını teyit edebilmek için ya da yetkili kullanıcıların çeşitli sistemlere veya mekânlara ulaşabilmesini sağlamak için kimlik doğrulama yöntemleri geliştirildi. Şekil 2.10'da görülebileceği gibi; kimlik doğrulama yöntemleri üç ana başlık altında toplanabilir [9]: Jeton tabanlı, Biyometrik tabanlı ve Bilgi tabanlı.



Şekil 2.10. Kimlik Doğrulama Yöntemlerinin Sınıflandırılması [9]

2.2.1 Jeton Tabanlı Doğrulama

Jeton tabanlı doğrulama; belirli bir zaman aralığında kullanılabilecek bir nesne olarak düşünülebilir. Bu nesne yeri geldiği zaman bir üniversite giriş kartı ya da yeri geldiği zaman bir internet sitesine şifresiz girişi sağlayan bir sanal id kart olabilir. Bir web sitesine erişim için ilk başta kullanıcı adı ve şifre ile belirli bir zaman aralığı için bir jeton (token) oluşturabilir. Oluşturulan jeton sisteme aktarıldıktan sonra kullanıcı web sitesine yetkili olarak ve şifresiz bir şekilde erişebilir [10]. İkinci bir örnek olarak da, grafik bir ara yüz üzerinden seçilen noktaları kullanarak bir jeton üreten sistem [11] verilebilir.

2.2.2 Biyometrik Tabanlı Doğrulama

Biyometrik tabanlı doğrulama; kişinin biyolojik bilgilerini kullanarak bir sisteme ya da bir mekâna giriş yapmasını sağlayan bir yöntemdir. Bu yöntemde; kişi, biyometrik sistemler çeşitli derecelerde etkileşim içerisinde olabilir. Biyometrik teknoloji ile direk etkileşimli olan yöntemlere örnek olarak parmak izi verilebilir. İlk başta biyometrik teknoloji ile parmak izinin görüntüsü alınır. Görüntü üzerinde düzeltme işlemleri yapıldıktan sonra parmak izi üzerindeki nokta, çökme, köşe gibi ayrıntılar üzerinden parmak izinin karakteristik özellikleri çıkarılır ve sisteme kaydedilir [12]. Mesela, bu özellikler ile kişi bir odaya girmek istediğinde yetkili kişi olup olmadığı önceden kaydedilmiş sistemdeki parmak izi ile verilen parmak izi özellikleri karşılaştırılarak yapılır. İkinci bir örnek olarak; parmak izi yöntemine benzer şekilde, iris tanıma yöntemi de gözdeki iris ve gözbebeği sınırlarını bularak kişiye özgü bir biyometrik doğrulama sistemi oluşturmaktadır [13].

2.2.3 Bilgi Tabanlı Doğrulama

Bilgi tabanlı doğrulama; kişinin önceden bildiği ve sisteme kaydettiği bir bilgiyi yeniden sisteme girerek yetkili olup olmamasının anlaşıldığı bir yöntemdir. Bu bilgi metin bir şifre ya da grafik ara yüzlü bir şifre olabilir. Örneğin; bir e-mail adresine ulaşım sağlarken kullanıcı adı ile beraber kullanılan bir şifre olabilir veya mail adresinizin unuttuğunuzda şifreyi sıfırlamak için kullanılan özel bir soru olabilir. Tabi bu soru, herkes tarafından cevap verilebilecek ve kolay hatırlanabilecek nitelikte olmalıdır. Ayrıca

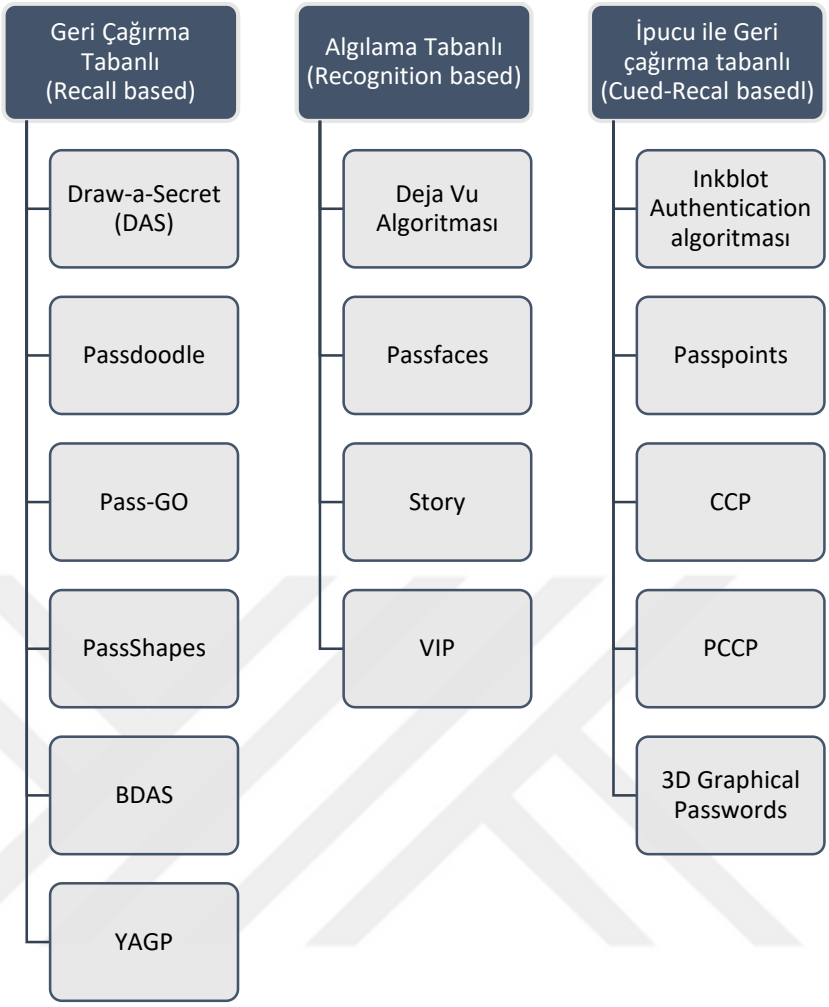
cevabın tek bir doğru cevabı olması ve başka kişiler tarafından tahmin edilemez olması gerekir [14].

2.3 Grafiksel Doğrulama

Kimlik doğrulama yöntemlerine genel olarak baktığımızda, koruma yöntemi olarak metin tabanlı şifrelemenin daha çok kullanıldığı görülse de son zamanlarda metin tabanlı şifreleme dışında farklı yöntemler de geliştirilmekte ve kullanılmaktadır. Metin tabanlı olmayan doğrulama yöntemlerinin en bilinen bir türü çizim tabanlı doğrulamadır. Çizim tabanlı doğrulama üç ana kategoriye ayrılabilir [15] .

1. Geri Çağırma Tabanlı (Recall based)
2. Algılama Tabanlı (Recognition based)
3. İpucu ile Geri çağırma tabanlı (Cued-Recall based)

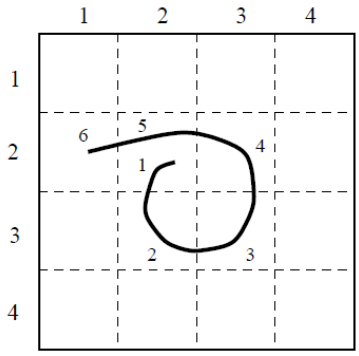
Geri çağırma tabanlı doğrulama yöntemlerine örnek olarak verilebilecek algoritmalar: Draw-a-secret (DAS) [16], Passdoodle [17], Pass-GO [18], PassShapes [19], BDAS [20], YAGP [21]. Algılama tabanlı doğrulama yöntemlerine örnek olarak verilebilecek algoritmalar: Deja Vu Algoritması [22], Passfaces [23], Story [24], VIP [25]. İpucu ile geri çağırma tabanlı doğrulama yöntemlerine örnek olarak verilebilecek algoritmalar: Inkblot Authentication algoritması [26], Passpoints [27], CCP [28], PCCP [29], 3D Graphical Passwords [30]. Bu algoritmaların görsel olarak sınıflandırılması Şekil 2.11’de verilmiştir.



Şekil 2.11. Grafik ara yüz tabanlı doğrulamanın sınıflandırılması

2.3.1 Geri Çağırma Tabanlı

Geri çağırma tabanlı sistemlerde, kullanıcı önceden çizmiş olduğu grafiksel şifresini hatırlamalı ve kimlik doğrulama sırasında bu görseli yeniden yapabilmelidir. Bu yönteme örnek olarak, Draw-a-Secret (DAS) algoritması [16] verilebilir. Algoritma şifre belirleme sırasında, kullanıcıdan 4x4 lük bir ızgara içerisine bir çizim yapmasını istemektedir ve kimlik doğrulama sırasında doğru ızgaralar kullanılarak aynı çizimin sırası ile yapılması beklenmektedir. Şekil 2.12'de DAS algoritmasının giriş ekranı Şekil 2.13'de DAS algoritmasının çıktıları görülebilir.

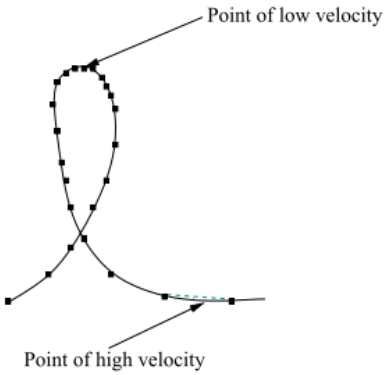


Şekil 2.12. Draw-a-Secret (DAS) algoritmasının 4x4 lük giriş ekranı [16]

$$(2,2),(3,2),(3,3),(2,3),(2,2),(2,1),(5,5)$$

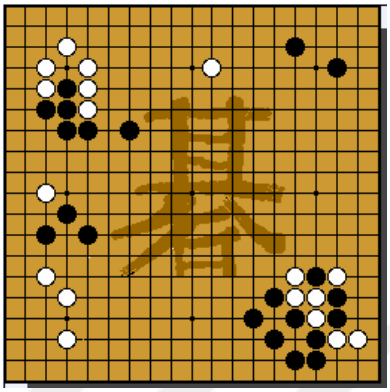
Şekil 2.13: Draw-a-Secret (DAS) algoritmasının çıktısı [16]

Passdoodle [17] algoritması da geri çağırma tabanlı sistemlere örnek olarak verilebilir. DAS algoritmasına benzer olarak 100x100 lük bir ızgara kullanılmaktadır. Grid yapısına ek olarak çizim yapılırken oluşturulan noktalar arasındaki hız belirlenmektedir. Bu sayede; kimlik doğrulama esnasında, görsel olarak çizimler aynı olması yeterli olmamakta ve çizim esnasındaki hızlar da karşılaştırılmaktadır. Şekil 2.14 üzerinde, çizim sırasında örnek alınan noktalar üzerinden hızların nasıl bulunduğu görülebilir.



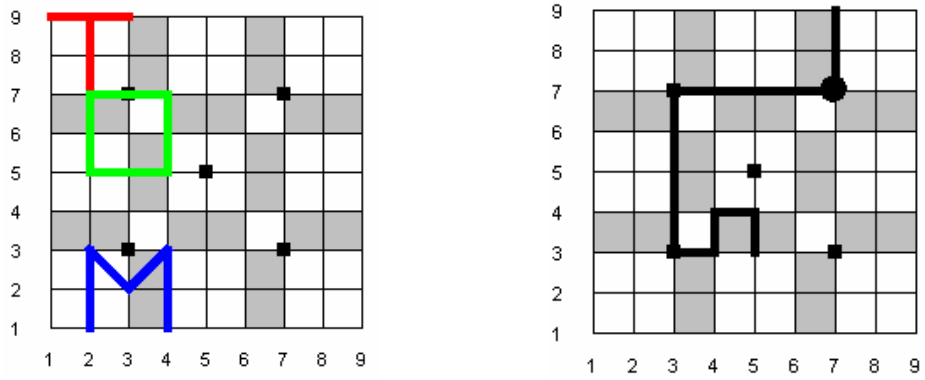
Şekil 2.14. Passdoodle algoritmasında hızların belirlenmesi [17]

Geri çağırma tabanlı sistemlere örnek olarak Pass-GO [18] algoritması da verilebilir. Yazarlar bu algoritmayı DAS [16] algoritmasını temel alarak daha da geliştirmiş ve daha kullanabilir hale getirmişlerdir. Geliştirme sırasında, Şekil 2.15'de görülen GO oyunundan esinlenmişlerdir.



Şekil 2.15: GO Oyunu [18]

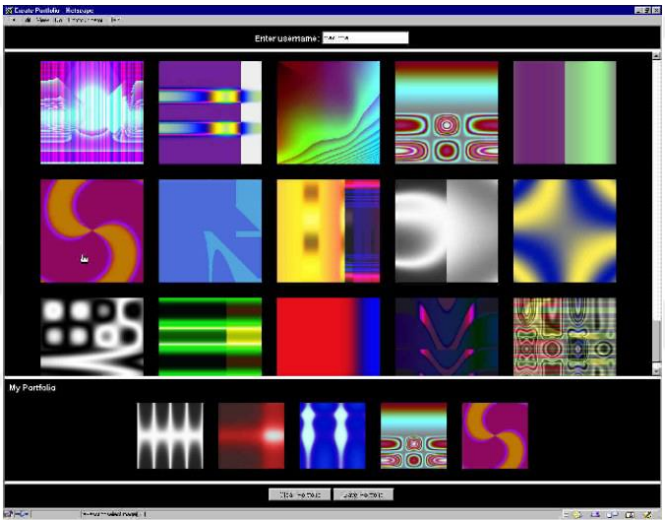
Renkli ve renksiz olmak üzere iki adet giriş yöntemi belirlenmektedir. Kullanıcıdan, 9x9'luk ızgara üzerinde bulunan duyarlı alanlar üzerinde çizgi ya da nokta yapması istenmektedir. Doğrulama sırasında, Şekil 2.16'daki gibi kullanıcının önceden oluşturmuş olduğu renkli veya renksiz şifreleri yeniden yapması istenmektedir.



Şekil 2.16. Pass-GO algoritmasının renkli ve renksiz şifrelerine örnek [18]

2.3.2 Algılama Tabanlı

Algılama tabanlı sistemlerde, kimlik doğrulama sırasında grafiksel bilgiler verilir ve kullanıcının önceden seçmiş olduğu görseli yeniden seçmesi beklenir. Bu sisteme örnek olarak Deja Vu [22] algoritması verilebilir. Algoritma şifre belirleme sırasında, oluşturulmuş olan resim veri tabanından, 1000 ve üzerinde resmin bulunduğu, 5 adet resim seçilmesini istemektedir. Kimlik doğrulama sırasında da seçilmiş resimlerde dâhil olmak üzere kullanıcıya Şekil 2.17'deki gibi 15 adet resim seçeneği sunulmaktadır. Bu resimler içerisinden önceden seçmiş olduğu resimleri seçmesi kullanıcıdan beklenmektedir.



Şekil 2.17. Deja Vu algoritmasının şifre giriş ekranı [22]

Passface [23] algoritması, algılama tabanlı sistemlere örnek olarak verilebilir. Şifre giriş ekranında herhangi bir görüntü ya da metin tabanlı şifre yerine yüz resimleri kullanılmakta. Yüz resimlerinden de daha rahat akılda kalabildiği için gülümseyen ve mutlu resimler seçtirilmekte. Şifre belirleme aşamasında kullanıcıya belirli yüz resimleri verilmekte ve bunlar arasından 5 adet seçmesi istenmekte. Şekil 2.18'de seçilmiş olan yüzler görülebilir.

Şekil 2.18. Passface için seçilmiş olan yüzler [23]

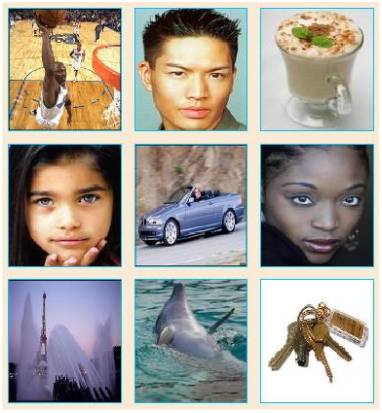
Şekil 2.19'da görüldüğü gibi, 3x3 lük şifre giriş ekranında önceden seçilmiş olan resimlerden bir tanesi ve geriye kalan 8 tane farklı resim konulmakta ve bunlardan doğru olanının kullanıcı tarafından seçilmesi istenmektedir. Her bir doğru yanıt verildikten sonra yeni bir ekranda sırası ile önceden seçilmiş olan yüzlerden bir tanesi ve 8 tane hatalı resim konulmaktadır. Bu örnekte, toplamda 5 resim doğru seçildikten sonra kimlik doğrulama işlemi tamamlanmaktadır.

Şekil 2.19: Passface şifre giriş ekranı [23]

Story [24] algoritması da algılama tabanlı sistemlere örnek olarak verilebilir. Bu algoritma, Passface [23] algoritmasını bir miktar daha geliştirilmiş hali denilebilir. Story algoritması, “Face scheme” ve “Story scheme” olmak üzere iki fazdan oluşmaktadır.

Şekil 2.20. Story algoritmasının "Face scheme" fazı [24]

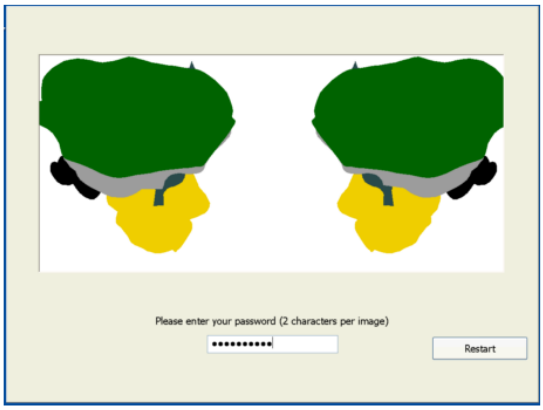
İlk fazda Passface algoritması gibi yüzlerin seçilmesini istemektedir ama Passface den farklı olarak yüzler toplamda on iki adet kategoriye (asyalı, erkek, bayan, erkek model, bayan model gibi) ayrılmıştır. Şekil 2.20’de görülebileceği gibi on iki adet kategoriden 3x3 lük yüz resimleri seçim ekranını oluşturmak için kategorileri farklı olmak üzere 9 adet resim seçilmektedir. İkinci faz da -Story-, birincisine benzer şekilde 9 adet kategori (hayvanlar, arabalar, erkek, bayan gibi) içermektedir. Bu kategorilere uygun olarak Şekil 2.21’de görüldüğü gibi 3x3 lük seçim ekranı oluşturulmaktadır. İki fazlı olan kimlik doğrulama aşamasında seçilen resimlerin sırası ile doğru seçilmesi beklenmektedir.



Şekil 2.21. Story algoritmasının "Story scheme" fazı [24]

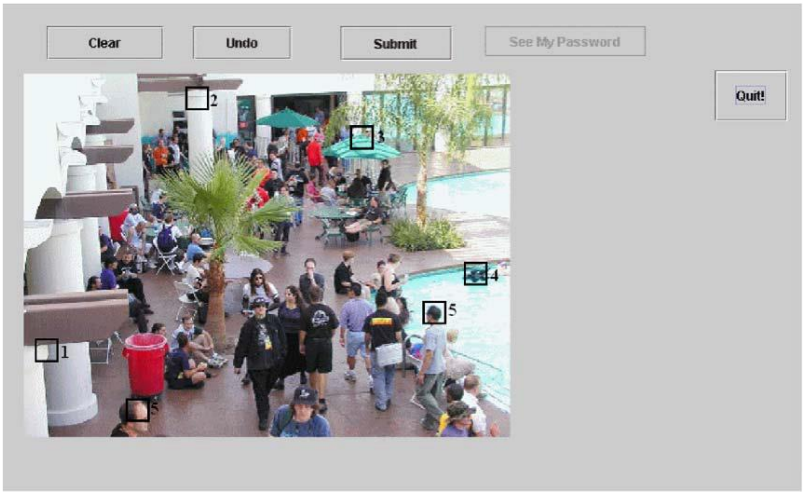
2.3.3 İpucu ile Geri çağırma tabanlı

İpucu ile geri çağırma tabanlı sistemleri için geri çağırma tabanlı ve algılama tabanlı sistemlerin birleşimi olarak söylenebilir. Bu sistemi Inkblot Authentication [26] algoritması üzerinde anlatırsak; algoritma şifre belirleme aşamasında, kullanıcıya rastgele olarak inkblots (mürekkep lekeleri) göstermekte ve kullanıcıdan bu imge neyi ifade etti ise, ifade eden kelime veya kelime tamlamasının ilk ve son harfini yazmasını istemektedir. Şekil 2.22'de gösterildiği gibi bu işlem on kez yapılarak şifre oluşturulması sağlanmaktadır. Kimlik doğrulama sırasında, kullanıcıya sırası ile gösterilen mürekkep lekelerine karşılık şifre belirlerken yazılan harflerin yazılması beklenmektedir.



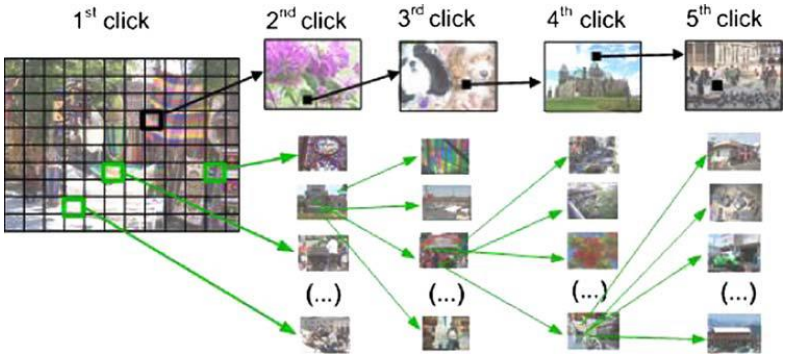
Şekil 2.22. Inkblot Authentication şifre oluşturma ekranı [26]

Passpoints [27] algoritması da bir resim üzerinde seçilen ardışık noktaların şifre olarak atanması şeklinde gerçekleşmektedir. Şekil 2.23'de görüldüğü şekilde, kullanıcı sisteme girmek istediğinde önceden seçmiş olduğu noktaların belirli bir mesafesi de doğru kabul edilerek, o resim alanı içerisindeki herhangi bir noktayı seçmelidir. Ayrıca, Şekil 2.23'de görüldüğü gibi seçimler sırası ile olması gerekmektedir; 1. resim alanı, 2. resim alanı şeklinde.



Şekil 2.23. Passpoint algoritmasının şifre seçim ekranı [27]

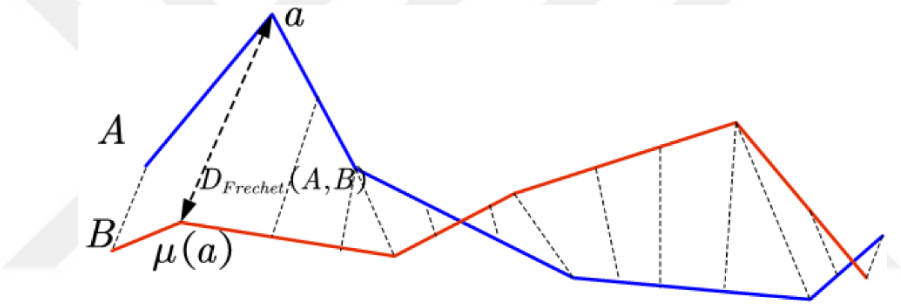
CCP [28] algoritması temel olarak Passpoints [27] algoritmasına benzemektedir ama şifre oluşturma ekranında birden fazla resim gösterilmektedir. Passpoints algoritmasındaki gibi seçilen ilk noktanın etrafındaki belirli bir alan doğru olarak kabul edilmekte ve her seçimden sonra yeni bir resim çıkarak; toplamda beş adet farklı resimde beş adet farklı nokta seçilmiş olmaktadır. Şekil 2.24 üzerinde CCP algoritmasının giriş ekranı gösterilmektedir.



Şekil 2.24. CCP algoritmasının şifre giriş ekranı [28]

2.4 El Çizim Temelli Parolaların Tanınması

Geri çağırma tabanlı sistemler kategorisine giren el çizim temelli parolaları tanımlamak için çeşitli algoritmalar kullanılmaktadır. Bunlardan bir tanesi Fréchet Distance [31] algoritmasıdır. Temel mantık olarak, iki adet eğri arasındaki benzerliği ölçmektedir. J. Zheng ve arkadaşları [32] makalelerinde, Fréchet Distance algoritmasını temel alarak atılan imzaların benzerliğini hesaplayarak sisteme giriş yapılıp yapılamayacağına karar vermektedirler. Bu çalışmada, önceden kayıtlı olan imza ile yetkilendirme sırasında atılan imzanın aynı olup olmadığı, imzaların tepe noktalarını, pik noktalarını ve çukur noktalarını Fréchet Distance algoritması ile karşılaştırarak tespit edilmektedir. Şekil 2.25 üzerinde, A ve B eğrisi üzerinde belirlenen noktaların Fréchet Distance algoritması ile nasıl benzerliklerinin hesaplandığı görülebilir.



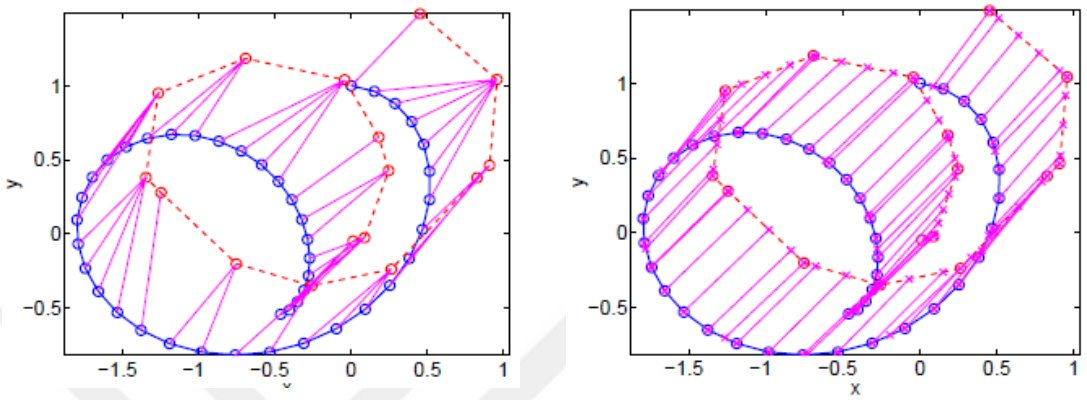
Şekil 2.25. A ve B Eğrisi üzerindeki belli noktaların, Fréchet Distance ile benzerliklerinin hesaplanması [33]

Dynamic time warping (DTW), belirli kısıtlamalar altında zamana bağlı diziler arasındaki optimum hizalamayı bulan bir tekniktir [34]. Bu teknik ilk başta, otomatik konuşma tanımada farklı konuşma kalıplarını karşılaştırmak için kullanılmıştır [35]. Sonraki zamanlarda DTW tekniği, el çizim temelli parolaların tanınması için çeşitli makalelerde de kullanılmıştır. S. K. Chan ve arkadaşı [36] ve R. Martens ve arkadaşı [37] makalelerinde çevrimiçi olarak imzaların doğrulanması için Dynamic Time Warping (DTW) tekniğini kullanılmışlardır. Munich ve Perona kamera üzerinden almış oldukları imza görüntülerini DTW tekniğini kullanarak doğrulamaya çalışmışlardır [38].

Continuous Dynamic Time Warping (CDTW) [39] tekniği, DTW tekniğinin geliştirilmiş bir türüdür. Şekil 2.26'de DTW ve CDTW tekniklerinin iki eğriyi

ilişkilendirmesi görülmektedir. CDTW belirli doğrulama noktalarında DTW kullanımına göre bir miktar öne geçmesine rağmen hesaplama zamanı olarak üç kata kadar daha yavaştır.

DTW eşleştirmesi CDTW eşleştirmesi

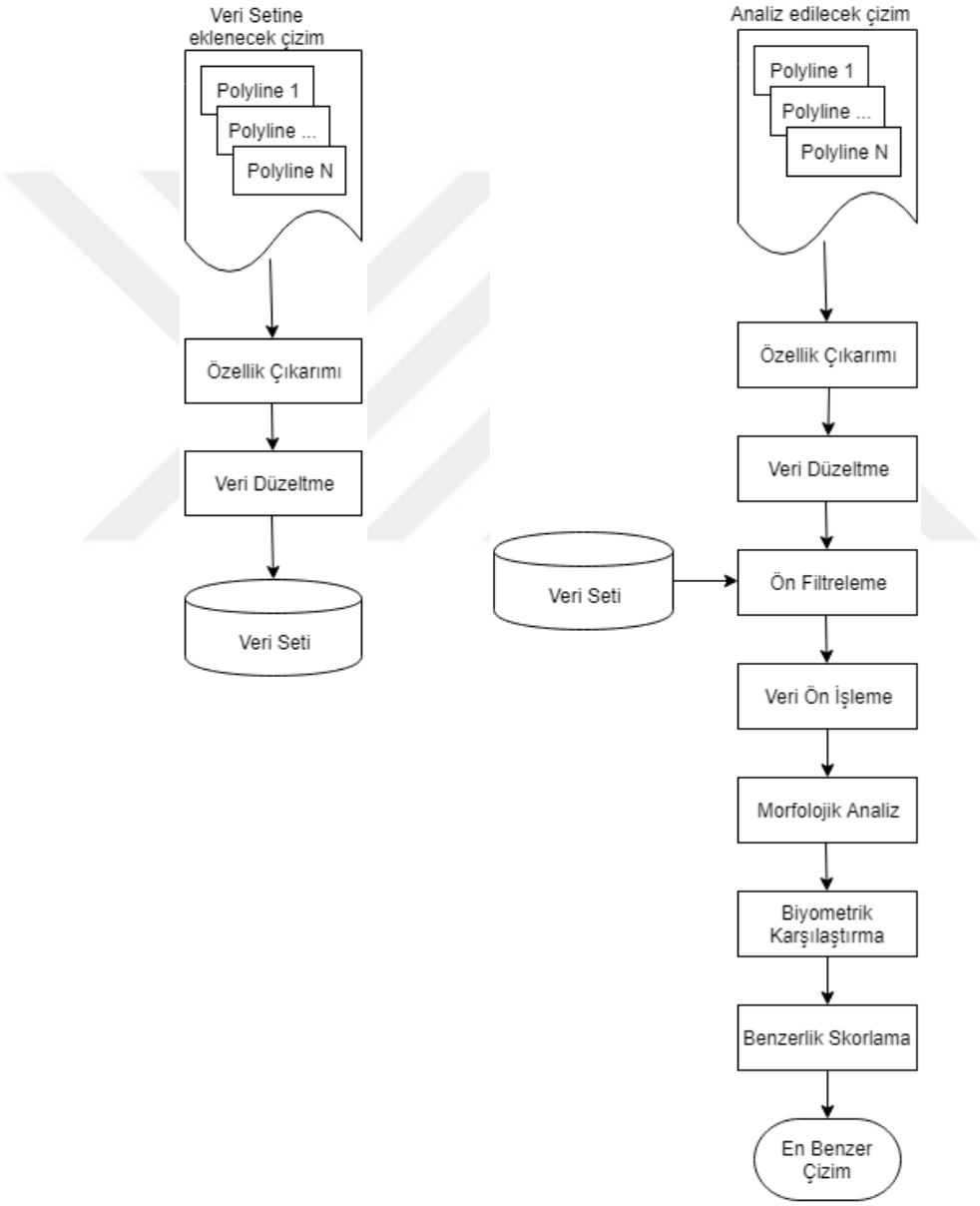


Şekil 2.26. DTW ve CDTW tekniklerinin iki eğriyi ilişkilendirmesi [39]

Simple K-Space (SKS) algoritması da eşleştirme ve tanıma uygulamalarında kullanılmaktadır. SKS algoritmasının tek parametreli sürümü [40]'de, çift parametreli sürümü [41]'da kullanılmıştır. Riggan [42] yaptığı çalışmada SKS algoritmasını alt yapı olarak kullanmıştır. Bu çalışmada, biyometrik bilgi olarak basınç bilgisini eklemiş ve bu eklemenin normal SKS algoritmasına göre daha iyi eşleşme sonuçları ürettiğini göstermiştir.

BÖLÜM 3. YÖNTEM

Biyometrik bilgileri kullanarak çizim tabanlı kimlik doğrulama için önerilen yöntemin bileşenleri, Şekil 3.1'de görülen özellik çıkarımı, veri düzeltme, ön filtreleme, veri ön işleme, morfolojik analiz, biyometrik karşılaştırma ve benzerlik skorlama ana başlıklarından oluşmaktadır. Bu modüllerin detayları ileriki bölümlerde anlatılacaktır.



(a) (b)

Şekil 3.1. Veri Seti Oluşturma (a), Veri Analizi Süreci (b) akış diyagramları

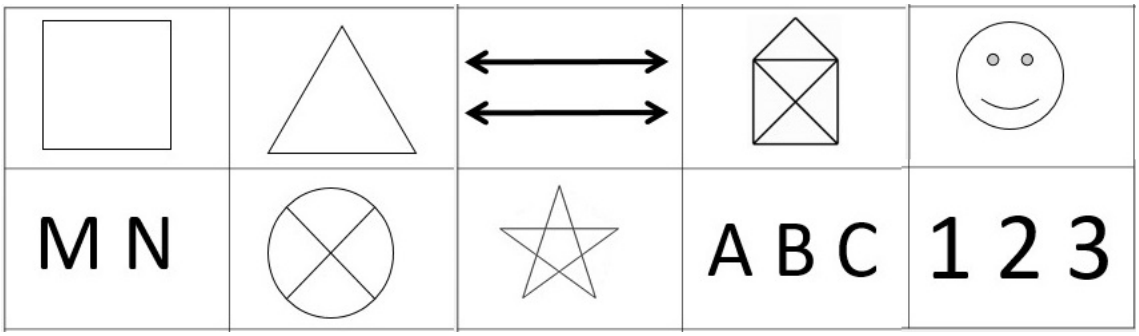
3.1 Veri Toplama

Dijital veri toplama işlemini yapabilmek için gönüllü kişiler ile çalışılmıştır. Veri toplama işlemi tablet üzerine dijital kalem ile çizilen şekiller ve imzalar üzerinden sağlanmıştır. Veri toplama yazılımı olarak Turgay Tugay Bilgin'in 114E427 no'lu "Yazı Yazma Güçlüğü Çeken Okul Öncesi Ve İlköğretim Öğrencileri İçin El Yazısı Özellik Çıkarımı ve Veri Analiz Platformu" adlı TÜBİTAK Projesi kapsamında geliştirdiği Mobil uygulama kullanılmıştır.

Şekil 3.2. Veri Toplama için kullanılan yazılım

Veri toplama sürecinde kullanılan mobil yazılımın kullanımına ait bir ekran görüntüsü Şekil 3.2'de görülmektedir. Uygulama hem dijital kalem ile hem de parmak ile kullanılabilmektedir. Bu çalışma kapsamında toplanan veriler için dijital kalem kullanılmıştır. Ayrıca kullanılan tablet, Z ekseninde basınç bilgisini de kaydedebilecek donanıma sahiptir.

Gerekli verinin toplanabilmesi için 10 kişiden Şekil 3.3'de görülen 10 adet çizim tipi (Kare, Üçgen, Ok, Ev, Gülen yüz, MN harfleri, Daire, Yıldız, ABC harfleri, 123 rakamları) ve imzalarını üçer kez yapması istenmiştir. Bu sayede 330 adet dijital çizim elde edilmiştir. Bunların 110 adedi test verisi olarak ayrılmış ve 220 adedi bilgi tabanı veri seti olarak kullanılmıştır.



Şekil 3.3. Çizim Tipleri – Sırasıyla: Kare, Üçgen, Ok, Ev, Yüz, Mn, Daire, Yıldız, Abc, Rakam

3.2 Çizim Dosyası Veri Biçimi

Ölçeklenebilir Vektör Grafikleri (Scalable Vector Graphics – SVG), 1999 yılından bu yana W3C Konsorsiyumu tarafından geliştirilen açık standartlara sahip XML tabanlı bir vektörel grafik biçimidir. SVG dosyalarının oluşturulmasında XML işaretleme dili kullanılır ve görüntüler ile davranışları XML metin dosyasında tanımlıdır. SVG, bir metin editörü veya grafik çizim yazılımı ile oluşturulabilir. Şekil 3.4'de görülen örnek bir "B" harfi için oluşturulan SVG dosyasının içeriği görünmektedir. <polyline> bilgileri ve <pressure> bilgileri XML formatında tutulmaktadır. <polyline> öğesi, ardışık noktalardan oluşan herhangi bir eğri oluşturmak için kullanılır. <pressure> öğesi ise, W3C tarafından belirlenen SVG standardında bulunmayan bir öğedir. Basınç önemli bir biyometrik bilgi olduğu için bu bilgiyi depolayabilmek adına SVG belgesine bizim tarafımızdan <pressure> niteliği eklenmiştir.

XML formatındaki bilgiler girdi olarak alındıktan sonra XML Parser ile aşağıdaki dosyalar çıktı olarak üretilmektedir:

a) Her bir <polyline> nesnesine ait noktaların koordinat verileri
b) Bir <polyline> nesnesinin tüm noktalarındaki basınç (pressure) bilgileri

```
<!DOCTYPE html>

<svg xmlns="http://www.w3.org/2000/svg" width="2560"
height="1504" viewBox="0 0 2560 1504">

<rect width='2560' height='1504' x='0' y='0'
fill='transparent' />

<polyline id="0" fill="none" stroke-linecap="round"
stroke-linejoin="round" stroke="#060628" stroke-
width="20" points="1177.7876,279.77045
1176.8407,274.74023 1174.0366,279.84967 ...
1166.3789,959.91516 1165.9097,965.1008 " />

<pressure point="0.48 0.525 0.57 0.57 ... 0.465 0.435
"/>

<polyline id="1" fill="none" stroke-linecap="round"
stroke-linejoin="round" stroke="#060628" stroke-
width="20" points="1197.7925,286.8034
1191.5409,279.12872 1190.9158,275.0818 ...
1156.369,969.7827 1151.3916,968.95764 " />

<pressure point="0.39 0.45 0.48 0.49499997 ... 0.345
0.345 0.345 0.32999998 "/>

</svg>
```

Şekil 3.4. Örnek SVG Dosyası

3.3 Veri Seti Depolama Biçimi

İleriki aşamalarda tanımlama işlemleri için kullanılacak olan kullanıcıların önceden çizmiş olduğu el çizimlerinin bütüncül olarak tutulduğu tekil dosyaya veri seti denilmektedir. Bu dosya içerisinde veriler JavaScript Nesnesi Gösterimi (JavaScript Object Notation – JSON) formatında tutulmaktadır. JavaScript uygulamaları için geliştirilmiş olan JSON veri formatı, anlaşılması kolay ve esnekliği sebebiyle birçok yazılım geliştirme aracında kullanılmaktadır. Şekil 3.5'te görüldüğü gibi bir JSON dosyası anahtar (key) ve değer (value) olmak üzere iki parçadan oluşmaktadır. Örnek üzerinden anlatmak gerekirse, Şekil 3.5'deki "type" anahtar alanı ve "kare" değer alanını ifade etmektedir. Veri seti formatı olarak JSON dosya formatının seçilmesi, programlama dili olarak esneklik sağlamasının yanında standart veri tabanları için olan veri tipindeki sınırları tamamen ortadan kaldırması ve sınırsız bir şekilde veriyi kaydedebilme olanağını sağlamasından kaynaklanmaktadır. Bu sayede, kişinin tablet üzerinde elini hiç

kaldırmadan çizebileceği tüm çizim herhangi bir kısıt olmadan veri seti içerisine kaydedilebilmektedir. Ayrıca, herhangi bir NoSQL veri tabanına geçiş için gerekli alt yapı hazırlanmış olmaktadır.


```
[
  {
    "ID": "1",
    "name": "TestDosyasi_2019_01_18-20_55_39_1Kare2",
    "type": "kare",
    "count": 2,
    "coords": {
      "polyline1": "1052.132,334.08057 ... 1060.4827,699.89355 ",
      "polyline2": "1034.0024,333.68988 ... 1056.5499,708.7552 "
    },
    "angles": {
      "polyline1": "278.80595609869357, ..., 291.8966766422092",
      "polyline2": "10.124494337760426, ..., 23.731142981301904"
    },
    "speed": {
      "polyline1": "11.7482238991394, ..., 7.318751258080377",
      "polyline2": "13.108046443442486, ..., 4.823142539769926"
    },
    "pressure": {
      "polyline1": "0.24000020398837427, ..., 0.18000005673155423",
      "polyline2": "0.2550000802970084, ..., 0.28499999108845203"
    }
  }, ...
```


Şekil 3.5. Örnek veri seti dosyası içeriği

Şekil 3.5'de JSON formatında görünen bilgiler JSON Parser ile bu modül içerisinde okunmaktadır. Çizimler ile ilgili aşağıdaki bilgiler çıktı olarak üretilmektedir:

1. Metin biçiminde "İsim"
2. Metin biçiminde "Tipi"
3. Tamsayı biçiminde <polyline> sayısı
4. Dictionary biçiminde "Koordinatlar"
5. Dictionary biçiminde "Açılar"

6. Dictionary biçiminde “Hızlar”
7. Dictionary biçiminde “Basınçlar”

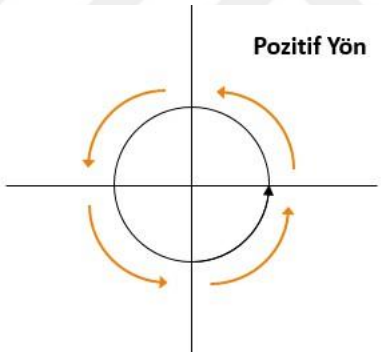
3.4 Özellik Çıkarımı

XML belgelerden gelen bilgiler ayrıştırıldıktan sonra bu bilgiler ile karşılaştırılan çizimin özellikleri (Polyline Sayısı, Açı, Hız ve Basınç) hesaplanmaktadır.

Girdi olarak, liste halindeki nokta ve metin halindeki basınç bilgileri alındıktan sonra liste halinde açı, hız ve basınç bilgileri üretilmektedir. Bu özelliklerin hesaplama işlemlerinin detayı bir alt başlıkta açıklanacaktır:

3.4.1 Açılar

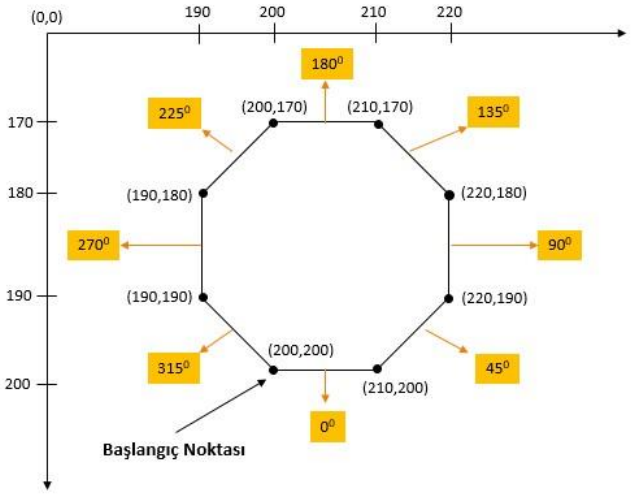
Açıları bulmak için kutupsal koordinat sistemi kullanılmıştır. Şekil 3.6’da gösterildiği gibi pozitif yön, saatin akış yönünün tersi olarak kabul edilmiştir.



Şekil 3.6. Kutupsal koordinat sistemi yönü

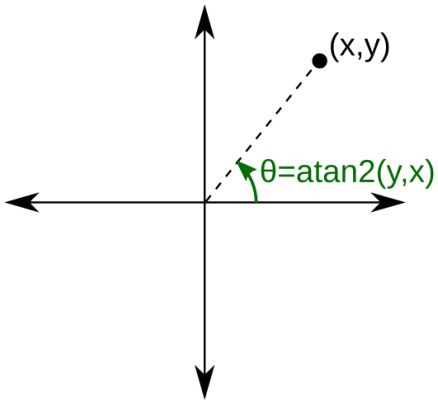
Şekil 3.7’te görüldüğü şekilde kutupsal koordinat sistemi üzerinden açılar hesaplanırken, arka arkaya gelen iki nokta arasındaki ve Şekil 3.8’de görülen ark tanjant bilgisi esas alınmaktadır. Şekil 3.7 üzerinde görülen örnek noktalar üzerinden daha detaylı açıklamak gerekirse; sekizgenin başlangıç noktasının x_1 ve y_1 koordinatları (200,200) şeklinde ve hemen ardından gelen x_2 ve y_2’nin koordinatları (210,200) şeklindedir. Şekil 3.7’de genel olarak verilen $atan2$ formülünde y_2 yerine 200, y_1 yerine 200, x_2 yerine 210 ve x_1 yerine 200 bilgilerini koyarsak $atan2(200-200, 210-200)$ işleminin sonucu sıfır olarak çıkmaktadır.

$$atan2(y_2 - y_1, x_2 - x_1)$$



Şekil 3.7. Kutupsal koordinat sistemi

$$\text{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & Eğer\ x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & Eğer < x\ ve\ y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & Eğer < x\ ve\ y < 0, \\ +\frac{\pi}{2} & Eğer\ x = 0\ ve\ y > 0, \\ -\frac{\pi}{2} & Eğer\ x = 0\ ve\ y < 0, \\ undefined & Eğer\ x = 0\ ve\ y = 0. \end{cases}$$



Şekil 3.8. atan2(y,x), (-π, π] aralığında sınırlandırılmış olan (x,y) noktası ile pozitif x ekseni arasındaki θ açısını getirir.

Kutupsal koordinat sisteminin kullanılması sebebiyle bazı noktalarda ani iniş ve çıkışlar olabilir. Bu durum Şekil 3.9.a'da görülmektedir. Bunları düzeltmek için, Algoritma 1'de sözde kodu (pseudo code) verilen aşağıdaki iki farklı yöntem uygulanır:

1. Ani sıfıra düşüşler:
    - Ani sıfıra düşme noktasının bir önceki açısı ile bir sonraki açısının ortalaması alınarak bu noktaya yazılır. Örneğin; bir önceki açı 90 ve bir sonraki açı 92 ise, 0 olan açı (90+92)/2=91 olarak değiştirilir: (90,0,92) -> (90,91,92)
2. Sıfırdan ani yükselişler:
    - Ani yükseliş noktasının bir önceki açısı ile bir sonraki açısının 0 olması durumunda, bu nokta sıfıra eşitlenir. Örneğin, bir önceki açı 0 ve bir sonraki açı 0 ise, 350 olan açı 0 olarak yazılır: (0,350,0) -> (0,0,0)

Algoritma 1. Açı Düzeltme İşlemi

$\theta = atan2(\vec{V})$

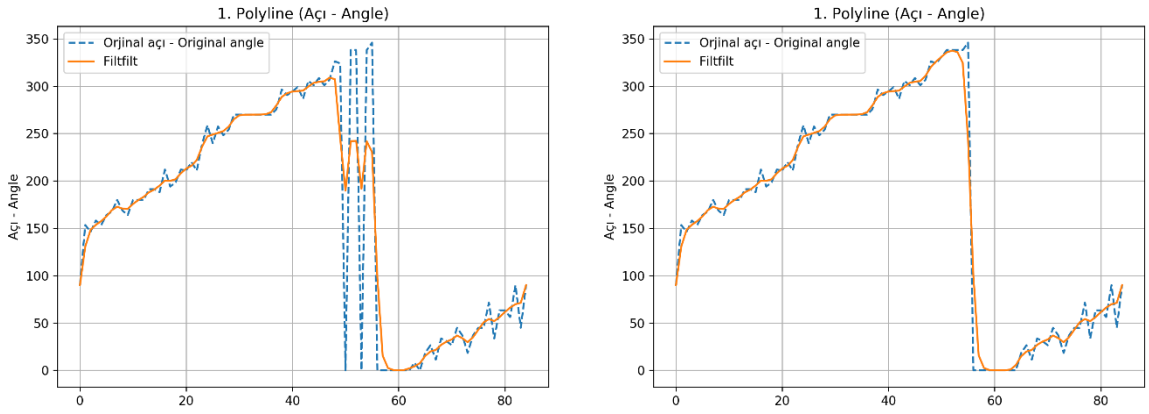
if $\theta_x = 0$ **and** $\theta_{x+1} != 0$ **and** $\theta_{x-1} != 0$ **then**

$\theta_x = (\theta_{x+1} + \theta_{x-1})/2$

else if $\theta_x != 0$ **and** $\theta_{x+1} = 0$ **and** $\theta_{x-1} = 0$ **then**

$\theta_x = 0$

end if



a) Düzeltme yapılmadan önce

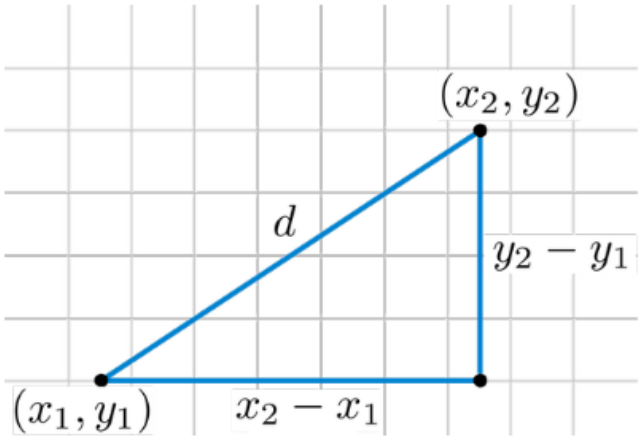
b) Düzeltme Yapıldıktan Sonra

Şekil 3.9. Açı düzeltme işlemi

Şekil 3.9.a'da açı düzeltme yapılmadan önceki durum görünmektedir ve kutupsal koordinat sistemi kullanılması sebebiyle oluşan ani iniş çıkışların eğrilere nasıl etki ettiği görünmektedir. Şekil 3.9.b'de ise açı düzeltme işlemi ile ani iniş çıkışların etkisi giderildikten sonraki durum görünmektedir.

3.4.2 Hızlar

Hız bilgisi bulunurken, Şekil 3.10'da görülen Öklid uzaklığı kullanılmaktadır. Dokunmatik cihazlar dokunma ile ilgili koordinat bilgisini alırken sabit bir zaman aralığında bilgi aldığı için iki nokta arasındaki mesafe direk hız bilgisini vermektedir. Örneğin; x_1 ve y_1 koordinat noktasını (0,0) kabul edilirse ve x_2 ve y_2 koordinat noktası (3,4) olarak kabul edilirse, hız bilgisi 5 olarak bulunur.



$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Şekil 3.10. Öklid uzaklığı

3.4.3 Basınçlar

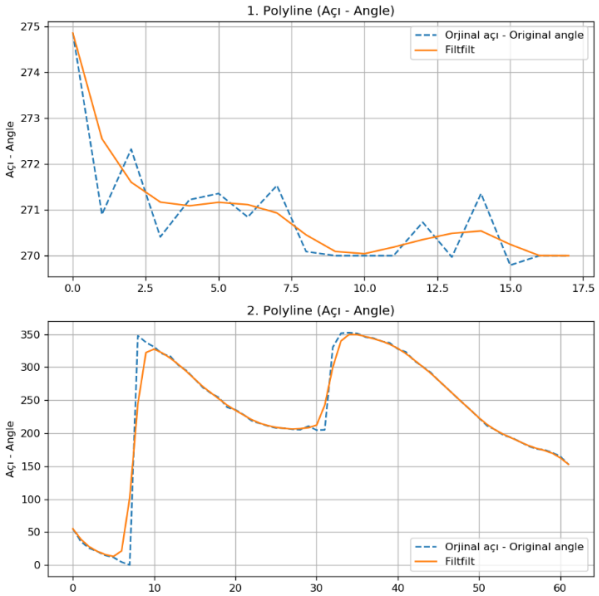
Basınç bilgisi doğrudan dokunmatik cihazı ürettiği bilgi üzerinden okunur. Algoritma içerisinde gelen string bilgisi boşluk (' ') karakterine göre bölünür ve eğer içerisinde basınç bilgisi var ise float veri tipine çevrilerek basınç listesine eklenir.

3.5 Veri Düzeltme

Özellikler bulunduktan sonra, yazım esnasında el titremesi sebebiyle oluşan ufak sıçramaları (spike) düzeltmek için doğrusal faz filtresinin (linear phase filter) özel bir durumu olan sıfır faz dijital filtresi (zero phase digital filter) kullanılır. Bu filtre, veri üzerinde bir kez ileri ve bir kez de geri olmak üzere iki kez işlem uygulamaktadır ve sıfır faz çarpıtma özelliğine sahiptir. Bu filtre içerisinde de aşağıda transfer fonksiyonunun en genel hali verilmiş olan Butterworth [43] filtresinin "low" tipi kullanılmaktadır.

$$G^2(\omega) = |H(j\omega)|^2 = \frac{G_0^2}{1+\left(\frac{j\omega}{\omega_c}\right)^{2n}} \quad (1)$$

Burada; n filtrenin derecesini, ω_c kesim frekansını (cutoff frequency) ve G_0 sıfır frekansındaki kazancı ifade etmektedir. Testler sonucunda her bir özellik için farklı filtreleme derecesi ve kesim frekansı belirlenmiştir. Kutupsal koordinat sistemi açıları için filtreleme derecesi 1 ve kesim frekansı 0,4; hız bilgisi için filtreleme derecesi 1 ve kesim frekansı 0,1; basınç bilgisi için filtreleme derecesi 1 ve kesim frekansı 0,4 olarak belirlenmiştir. Bu filtreleme işlemleri sonucunda; Şekil 3.11'de görüleceği gibi açılar, hızlar ve basınçlar üzerinde daha yumuşak ve daha karşılaştırılabilir bir veri oluşturulmaktadır.



Şekil 3.11. Açı filtrelemesi

3.6 Ön Filtreleme

Bu modül içerisinde biyometrik bilgi olan polyline sayıları karşılaştırılmaktadır. Karşılaştırma yapılmasının sebebi, aynı gibi görünen çizimler için kişilerin tablet üzerinde farklı sayıda el kaldırışları yapmasından kaynaklanmaktadır. Örnek üzerinden açıklamak gerekirse; bir üçgeni çizmek için bir kişi üçgenin tüm kenarlarını üç hamlede teker teker çizerken, diğer bir kişi tek bir hamlede elini hiç kaldırmadan üçgenin tüm kenarlarını çizebilmektedir. Buradaki hamle sayısı, diğer bir ifade ile polyline sayısı kişinin biyometrik özelliğini yansıtmaktadır. Bu modül içerisinde karşılaştırma işleminin yapılabilmesi için girdi olarak aşağıdaki bilgiler alınmaktadır:

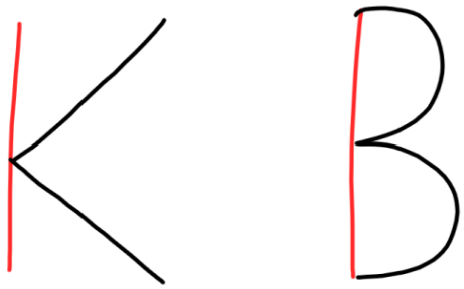
1. Veri setini okuma modülü üzerinden üretilmiş olan veri setindeki çizimlerin listesi
2. Tamsayı biçiminde olan veri setindeki çizimlerin polyline sayısı
3. SVG dosyasını ayrıştırma modülünün üretmiş olduğu tamsayı biçimindeki karşılaştırılan çizimin polyline sayısı

Yukarıdaki bilgiler alındıktan sonra karşılaştırılan çizimin polyline sayısı ile veri setindeki çizimlerin polyline sayısı eşleşmediği zaman veri seti içeresindeki çizim silinmekte ve çıktı olarak filtrelenmiş veri seti çizimleri listesi verilmektedir. Örnek vermek gerekirse, karşılaştırılan çizimin içerisindeki polyline sayısı iki olduğu zaman ve veri seti içerisinden gelen çizimin polyline sayısı üç olduğu zaman, bu çizim veri seti listesinden çıkarılmaktadır.

3.7 Veri Ön-işleme

Her bir polyline bilgisi için oluşturulan açı bilgileri tek bir polyline olacak şekilde birleştirilir ve hızlı bir işlem yapabilmek için toplam açı sayısı 1/3 oranına düşürülür. Buradaki amaç, görüntüleri büyük bir bakış açısı ile bakarak detaya inmeden karşılaştırma yapılabilmesini sağlamak ve uymayan çizimi baştan elemektir. Şekil 3.12 üzerinden örnek vermek gerekirse; “K” harfinin kırmızı ile işaretlenmiş olan kısmı ile “B” harfinin kırmızı ile işaretlenmiş olan kısmı birbirlerine çok benzemektedir ve bu benzerlik ileriki hesaplamalarda hatalı sonuçlar ürettirebilmektedir. Karşılaştırılan çizim olarak “B” harfinin girdi olarak geldiğini varsayarsak ileriki hesaplamalara veri seti

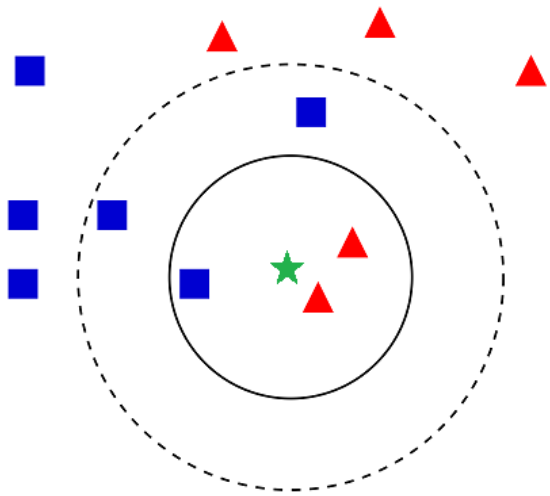
üzerinden gelen “K” harfinin detay hesaplamalara girmemesi gerekmektedir ve açıların birleşimi ile bu durum ortadan kaldırılmaktadır.



Şekil 3.12. Açıları birleştirme durumuna bir örnek

3.8 Morfolojik Karşılaştırma

Bu modül içerisine girdi olarak; veri seti üzerinden filtrelenerek gelen çizimlerin açıları ve karşılaştırılan çizimin birleştirilmiş açıları alınmaktadır. Bu aşamada, çizimler karşılaştırılırken herhangi bir biyometrik bilgi kullanılmamakta olup sadece biçimsel (morfolojik) benzerlik tespit edilmektedir. Çizimlerin polyline bilgileri bu modül içerisinde birleştirilmektedir. En benzer çizimleri bulmak için veri madenciliği sınıflandırma algoritmalarından K en yakın komşu (k-NN) algoritması kullanılmıştır.

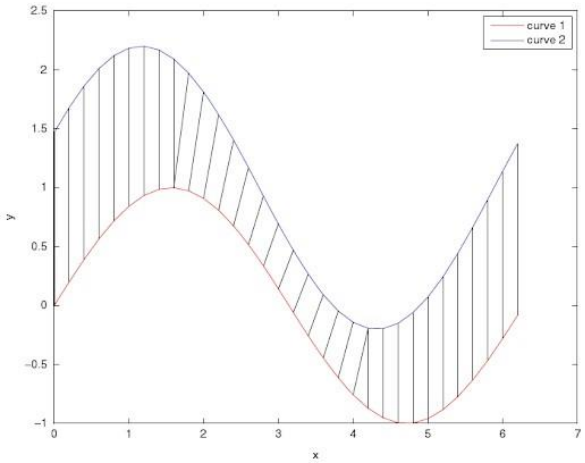


Şekil 3.13. K en yakın komşu (k-NN) algoritma örneği [44]

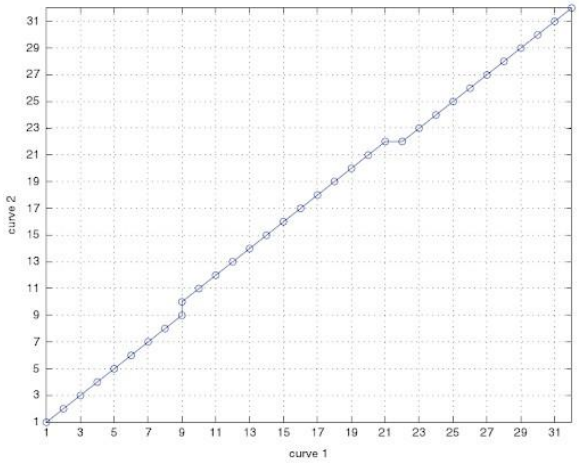
Şekil 3.13’de K en yakın komşu (k-NN) algoritması sınıflandırma yöntemine bir örnek gösterilmiştir. Burada, hangi sınıfa ait olduğu bilinmeyen ve karşılaştırılmaya

çalışılan eleman yıldız olarak ve yeşil renk ile gösterilmiştir. Etrafında görünen mavi renkli kareler ve kırmızı renkli üçgenler yıldıza belli bir mesafede konumlanmıştır. Yıldıza en yakın üç adet elemanı bulmak için K=3 olacak şekilde daire çizilir. Örnekte göründüğü şekilde, düz çizgili daire iki adet üçgen ve bir adet kare olmak üzere üç adet eleman içermektedir. İki adet üçgen, bir adet kareye göre miktar olarak daha fazla olması sebebiyle yeşil renkli yıldız üçgen olarak sınıflandırılır. Benzer şekilde, K=5 olacak şekilde aynı örneğe bakarsak. Kesik çizgili daire iki üçgen ve üç adet kare olmak üzere beş adet eleman içermektedir. Bu durumda ise, kare miktar olarak üçgenden daha fazla olduğu için yıldız kare olarak sınıflandırılır.

K en yakın komşu (k-NN) algoritması içerisinde mesafe hesaplama yöntemi olarak Manhattan uzaklığı (Manhattan distance) veya Öklid uzaklığı (Euclidean Distance) kullanılabilmektedir ama zamansal bir diziye sahip olunması sebebiyle; veri setinden gelen her bir çizimin açıları ile karşılaştırılan çizimin açıları arasındaki uyum için Dynamic time warping (DTW) algoritması kullanılmıştır [34]. DTW, belirli kısıtlamalar altında zamana bağlı diziler arasındaki optimum hizalamayı bulan bir tekniktir. Şekil 3.14'de, DTW ile örnek iki tane eğrinin hizalanması gösterilmektedir. Şekil 3.15'de, DTW ile eşleşen noktaların indeksi gösterilmektedir.



Şekil 3.14. DTW'de örneklerin hizalanması [45]



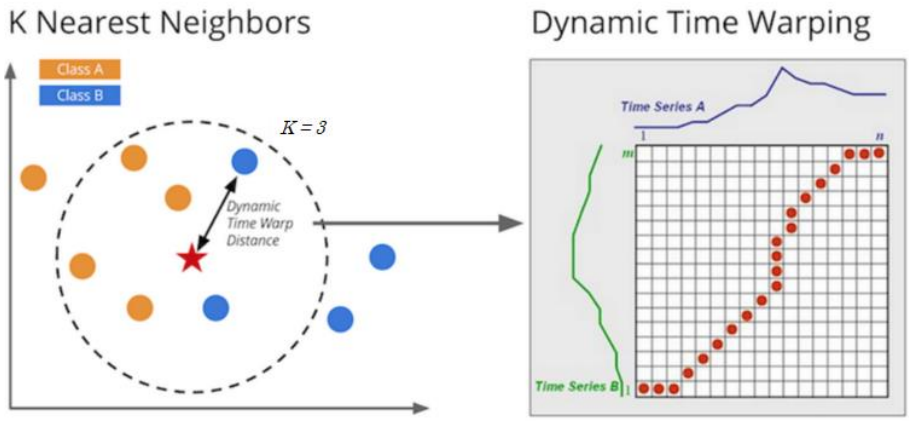
Şekil 3.15. DTW'de eğrilik yolu [45]

DTW ile her bir çizim için hesaplanan değerler K en yakın komşu (k-NN) algoritmasında K=3 olacak şekilde en benzer üç adet çizim seçilir. Matematiksel olarak ifade etmek gerekirse, karşılaştırılan çizimin açıları $X = \{x_1, x_2, \dots, x_n\}$ şeklinde ve veri setinden gelen her bir çizimin açıları $Y_i = \{y_{i1}, y_{i2}, \dots, y_{im}\}$ şeklinde tanımlanmış olsun. K en yakın komşu (k-NN) algoritması içerisinde mesafe hesaplama işlemi aşağıdaki şekilde olur:

$$kNN\_mesafe(X, Y_i) = DTW(X, Y_i) \quad (2)$$

Her bir çizim için hesaplanan mesafe sonuçlarına göre K=3 olacak şekilde veri seti içerisinden karşılaştırılan çizime en yakın üç adet çizim seçilir.

Şekil 3.16'da, yıldız ile işaretli yer karşılaştırılan çizimin kendisini ifade etmektedir. Etrafındaki her bir renkli yuvarlak veri setindeki çizimleri ifade etmektedir. Çizimler ile karşılaştırılan çizim arasındaki uzaklık DTW ile hesaplanmaktadır. Dynamic Time Warping başlıklı resimde iki adet zaman bağımlı dizi görünmektedir. Bunlardan bir tanesi karşılaştırılan çizimi, diğeri ise veri setinden bir çizimi ifade etmektedir. DTW yöntemine göre karşılaştırılan çizimin noktaları ile veri setindeki çizimin noktaları arasındaki en kısa mesafe, kırmızı noktalar ile gösterilmektedir. Kırmızı noktalardaki hesaplanmış olunan değerlerin toplamı da iki çizim arasındaki DTW değerini vermektedir.



Şekil 3.16. En benzer çizimleri bulma [46]

3.9 Polyline Seviyesinde Biyometrik Karşılaştırma

Bu modül içerisinde, en benzer olarak gelen N adet çizimdeki tüm toplam polyline'lar benzerlik tespiti yapılacak çizimin tüm polyline'ları ile kıyaslanmaktadır. Bu işlemin sözde kodu (pseudo code) Algoritma 2'de görülmektedir ve girdi olarak aşağıdaki bilgiler alınmaktadır:

1. Liste halinde en iyi N adet çizim
2. Liste halinde karşılaştırılan çizimin filtrelenmiş açıları
3. Liste halinde karşılaştırılan çizimin filtrelenmiş hızları
4. Liste halinde karşılaştırılan çizimin filtrelenmiş basınçları

Karşılaştırılan çizimin içerisindeki her bir polyline, sıralarına özen gösterilerek veri seti içerisindeki polyline'lar ile karşılaştırmaktadır. Yapılan işlemi örnek üzerinden anlatmak gerekirse; karşılaştırılan çizimin birinci polyline, veri seti içerisindeki çizimlerin birinci polyline'ları ile karşılaştırılmaktadır. Bu karşılaştırma sırasında; her bir polyline için hem açı hem hız hem de basınç için DTW [47] hesaplaması yapılmaktadır. Her bir özellik (açı, hız ve basınç) için çıkan DTW sonuçları farklı katsayılar ile çarpılıp toplanarak kümüle bir DTW değeri (Algoritma 2'de görünen *toplam_maliyet* değişkeni) bulunmaktadır. Veri seti içerisindeki çizimler arasında en düşük toplam DTW değerine sahip olan çizim, bulunmuş olduğu polyline için en benzer çizim olarak seçilir. Bu sayede her bir polyline için ayrı ayrı en benzer çizim bulunur ve bu bilgiler liste halinde bir sonraki modüle aktarılır.

Algoritma 2'de görünen *toplam_maliyet* değişkeninin hesaplanma yöntemi Denklem 3'de verilmiştir. Bu denklem içerisinde, DTW_a açı benzerliği için hesaplanan DTW değerini, DTW_v hız benzerliği için hesaplanan DTW değerini, DTW_p basınç benzerliği için hesaplanan DTW değerini, C_a açı için kullanılan sabit katsayıyı, C_v hız için kullanılan sabit katsayıyı, C_p basınç için kullanılan sabit katsayıyı ifade etmektedir. En ideal sabit katsayı kombinasyonunu bulabilmek için yapılan 244.552 adet test sonucunda; C_a açı katsayısı 1 olarak, C_v hız katsayısı 10 olarak, C_p basınç katsayısı 100 olarak belirlenmiştir.

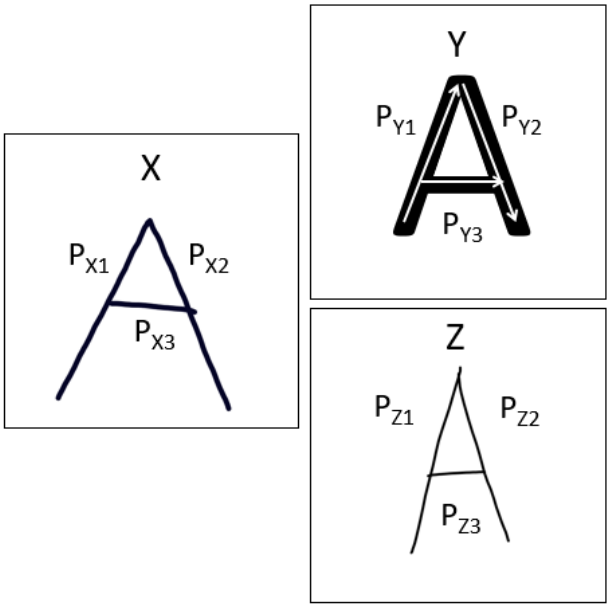
$$WSM\_DTW = DTW_a * C_a + DTW_v * C_v + DTW_p * C_p \quad (3)$$

Algoritma 2. Polyline Seviyesinde Biyometrik Karşılaştırma

```
Data: TopN_Sketch - Veri seti içerisindeki En Benzer Çizim Listesi ve Özellikleri;
Dosya – Karşılaştırılan dosya ve özellikleri;
T - Polyline Sayısı
for i in 1:T do
    best_list = []
    for sketch in TopN_Sketch do
        Sketch_ismi := sketch[isim]
        Maliyet_Açı, Path_Açı := DTW(Dosya[i][Açı], sketch[i][Açı])
        Maliyet_Hız, Path_Hız :=  DTW(Dosya[i] [Hız], sketch[i][Hız])
        Maliyet_Basınç, Path_Basınç := DTW(Dosya[i] [Basınç], sketch[i][Basınç])
        best_list.append(Sketch_ismi, Maliyet_Açı, Maliyet_Hız, Maliyet_Basınç)
    end for
    minimum_toplam_maliyet := 99999
    for best_l in best_list do
        toplam_maliyet := best_l [Maliyet_Açı] + best_l[Maliyet_Hız])*10 +
best_l[Maliyet_Basınç]*100
        if ( toplam_maliyet < minimum_toplam_maliyet) then
            if genel_karsilastirma_etiketi != DOĞRU and toplam_maliyet >
toplam_maliyet_esigi then
                continue
            end if
            minimum_toplam_maliyet := toplam_maliyet
            best := best_l
        end if
    end for
end for
```

3.10 Skorlama Yöntemi ile Benzerlik Tespiti

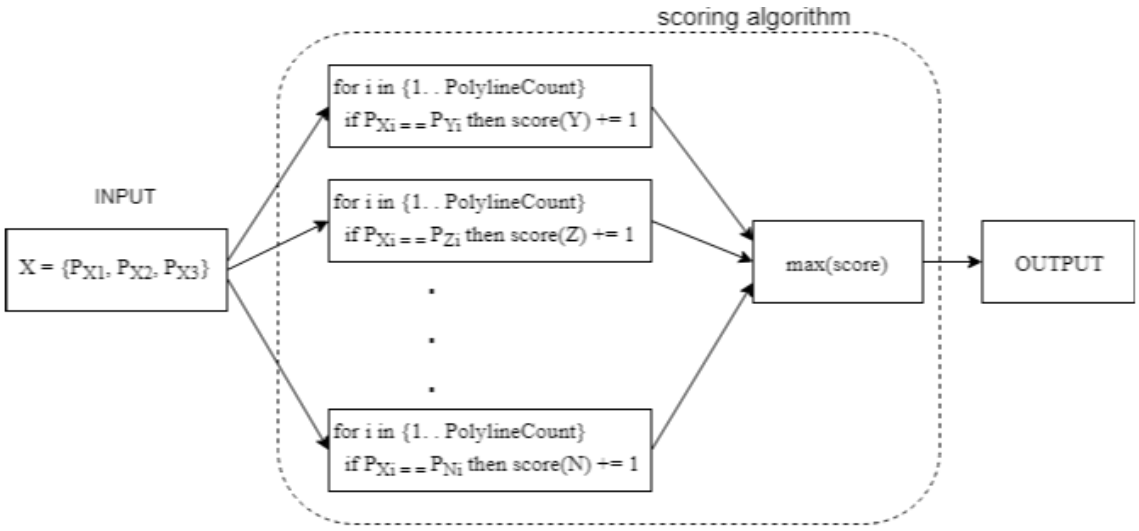
Bu modül içerisinde, en benzer çizimlerin benzeşen polyline sayıları karşılaştırılmaktadır. Örnek üzerinden anlatmak gerekirse; Şekil 3.17'daki X = {P_{X1}, P_{X2}, P_{X3}} olmak üzere benzerlik araması yapılacak çizim ve Y = {P_{Y1}, P_{Y2}, P_{Y3}} ve Z = {P_{Z1}, P_{Z2}, P_{Z3}} olmak üzere veri setindeki iki adet çizim olsun. Y çizimi üzerinden görüleceği gibi P ifadesi polyline'ın kısaltmasını ve A harfi çizerken oluşan polyline'ların yönlerini göstermektedir. Benzerlik karşılaştırılması yapılırken her bir polyline bağımsız olarak değerlendirilmektedir. X e ait olan P_{X1} polyline'ına en çok benzeyen polyline P_{Y1} olsun. Benzer şekilde, P_{X2} polyline'ına en çok benzeyen P_{Y2} olsun. P_{X3} polyline'ına en çok benzeyen polyline ise P_{Z3} olsun. Bu durumda; X çizimine benzeyen polyline sayılarına bakıldığında, Y çizimi için toplam 2 adet polyline bulunmaktadır ve Z çizimi için 1 adet polyline bulunmaktadır. Y çizimi en fazla benzeyen polyline sayısına sahip olduğu için ve karşılaştırılan çizime göre en fazla 1 adet eksik polyline sayısına (3-1=2) sahip olduğu için skoru en yüksek çizimdir ve dolayısıyla en benzer çizim olarak seçilmektedir.



Şekil 3.17. X, karşılaştırılan çizim; Y ve Z, veri setindeki çizimler

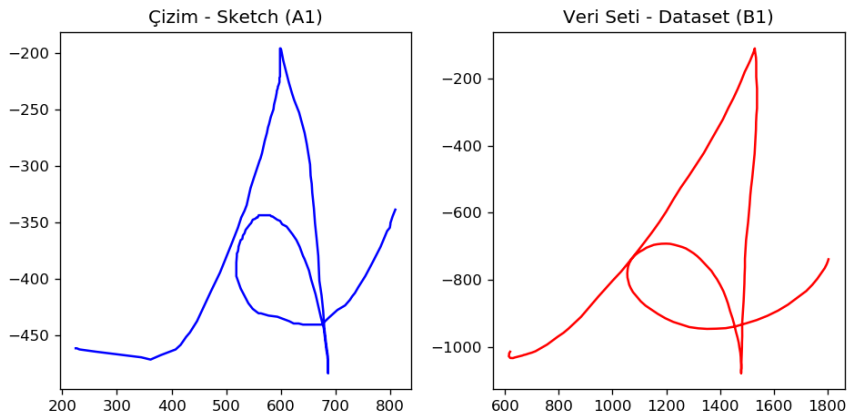
Bu modül içerisinde kullanılan skorlama algoritması Şekil 3.18'de görünmektedir. Örnek üzerinden anlatmak gerekirse, Şekil 3.17'de görünen X çizimi

Şekil 3.18'de INPUT olarak kabul gösterilmiştir. X çiziminin polyline'ları tek tek benzerlik testi yapılan çizimlerin (Y ve Z) polyline'ları ile karşılaştırılmaktadır. X çizimi ile eşleşen polyline sahip olan çizime 1 puan eklenmektedir. Yukarıdaki örneğe göre Y çizimi 2 puan ve Z çizimi 1 puan toplamaktadır. En fazla puanı toplayan Y çizimi, skorlama algoritması tarafından OUTPUT olarak üretilmektedir.



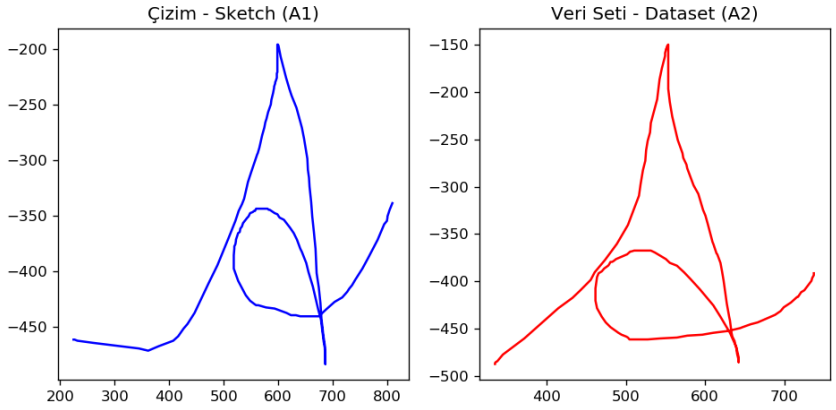
Şekil 3.18. Skorlama Algoritması

Şekil 3.19'da sadece koordinat sistemi bilgisi kullanılarak yapılan bir eşleştirme gösterilmektedir. Bu eşleştirme yöntemi ile veri seti içerisinden benzer çizim eşleşmesi yapılabilmektedir. Örnekten görüleceği gibi, "A1" ismi ile tanımlanan "A" harfi, veri seti içerisinde bulunan "B1" ismi ile tanımlanan "A" harfi ile eşleşmektedir.



Şekil 3.19. Sadece koordinat sistemi kullanılarak yapılan benzerlik tespiti

Şekil 3.20’de koordinat sistemine ek olarak hız bilgisinin de hesaplamalara katılmasıyla yapılan benzerlik tespiti gösterilmektedir. Veri seti içerisinde “B1” ismi ve “A2” ismi ile tanımlanan “A” harfi benzer görüntülere sahip olsa da farklı biyometrik bilgilere sahiptir. Şekil 3.19 ve Şekil 3.20’da yapılan benzerlik tespitinden, asıl doğru olan sonuç Şekil 3.20’deki “A2” ile isimlendirilmiş “A” harfidir. Çünkü “A1” çizimini yapan kişi ile “A2” çizimini yapan kişi aynı kişidir. Bu örnekten anlaşıldığı gibi biyometrik bilgilerin eşleşmeye etkisi net bir şekilde görülmektedir.



Şekil 3.20. Hem koordinat sistemi hem de hız bilgisi kullanılarak yapılan benzerlik tespiti

BÖLÜM 4. DENEYSEL SONUÇLAR

Dijital veri toplama işlemini yapabilmek için gönüllü kişiler ile çalışılmıştır. Veri toplama işlemi tablet üzerine dijital kalem ile çizilen şekiller ve imzalar üzerinden sağlanmıştır. Test işlemleri "Genel Çizim Tipleri" ve "İmza" olmak üzere iki ana başlık altında toplanmaktadır ve test işlemlerinin detayları alt başlıklarda verilecektir.

4.1 Genel Çizim Tipleri

Bu bölümde gönüllülerden toplanan yazılar, rakamlar ve basit şekil çizimleri veri seti olarak kullanılmıştır. Bölüm 4.1.1'de sayılar, rakamlar, çeşitli karakterler ve basit şekiller içeren heterojen bir veri seti kullanılarak benzerlik arama işlemi gerçekleştirilmesi açıklanmıştır. Bölüm 4.1.2'de ise benzerlik arama için aynı türden çizimler içeren homojen bir veri seti kullanılmıştır.

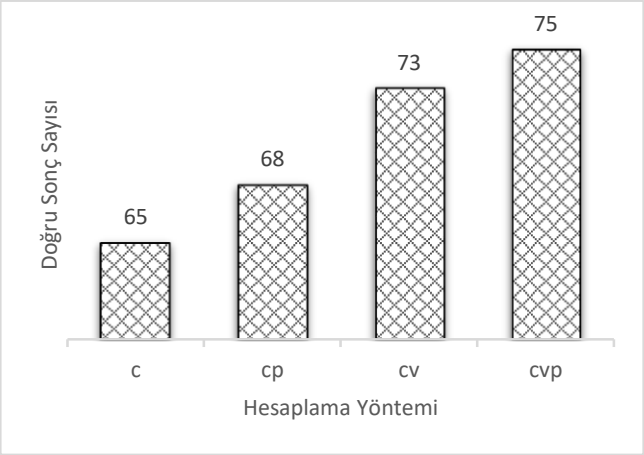
4.1.1 Benzerlik aramanın heterojen veri setinde gerçekleştirilmesi

Bu test işlemi sırasında, çizim tipi dikkate alınmayarak, çizime ait koordinat, hız ve basınç değerlerine göre eşleştirme işlemi yapılmaktadır. Örnek üzerinden anlatmak gerekirse, kare tipindeki bir çizim veri seti içerisinde bulunan tüm çizim tipleri ile kıyaslanmaktadır ve buna göre sonuç üretilmektedir. Ayrıca, doğru sonucun üretilebilmesi için, x kişisinin kare çiziminin, veri setindeki kendisine ait olan kare çizimleri ile eşleşmesi gerekmektedir. Daha farklı şekilde ifade etmek gerekirse, x kişisinin çizdiği kare çizimi ile y kişisinin kare çizimi eşleşirse hatalı olarak kabul edilmektedir. Örnek ile anlatılan doğru eşleşme ve yanlış eşleşme işleminin sözde kodu (pseudo code) Algoritma 3'de verilmiştir.

Algoritma 3. Heterojen Veri Setinde Karşılaştırma

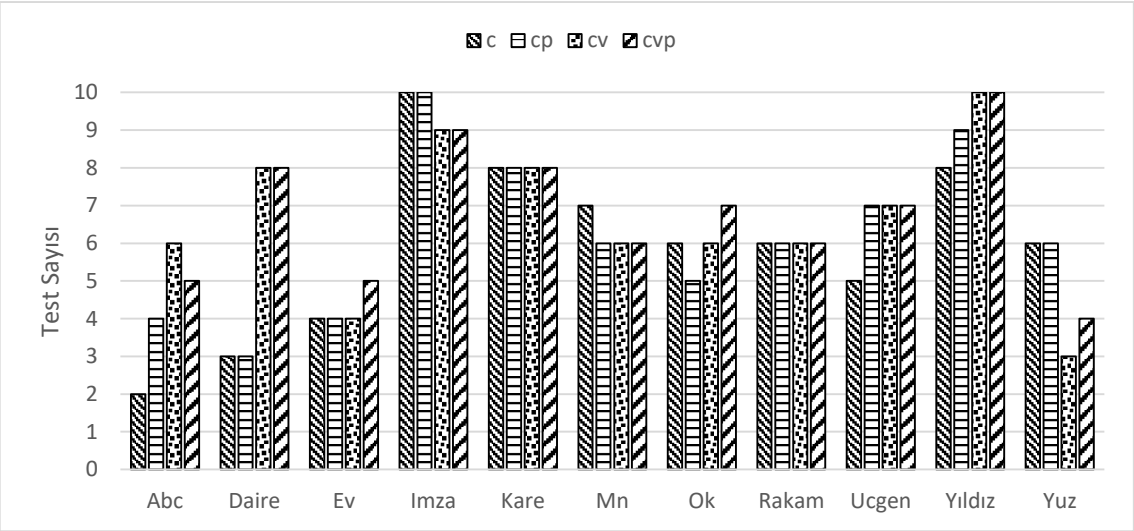
```
Data: DataSet - Veri seti içerisindeki Tüm Çizim Listesi ve Özellikleri;
Çizim – Karşılaştırılan çizim ve özellikleri;
Eslesen _Cizim := Eslesen_Cizimi_Bul(Çizim, DataSet)
if  Cizim[Tipi] == Eslesen _Cizim[Tipi] and Cizim[Kisi] == Eslesen _Cizim[Kisi]
then
   sonuc := "Doğru Eşleştirme"
else
   sonuc := "Hatalı Eşleştirme"
end if
```

Şekil 4.1'de bulunan kısaltmalar şunları ifade etmektedir: "c" koordinat bilgisini, "p" basınç bilgisini ve "v" hız bilgisini. Hesaplama yöntemi alanında bulunan "c" ile sadece koordinat bilgisine dayanılarak yapılan eşleşme sonuçlarını, "cp" ile koordinat bilgisi ve basınç bilgisinin beraberce hesaplamaya katılması sonucu oluşan eşleşme sonuçlarını; "cv" ile koordinat bilgisi ve hız bilgisinin beraberce hesaplamaya katılması sonucu oluşan eşleşme sonuçlarını; "cvp" ile koordinat bilgisi, hız bilgisi ve basınç bilgisinin beraberce hesaplamaya katılması sonucu oluşan toplam eşleşme sonuçlarını göstermektedir. Grafikte, her bir biyometrik bilginin (hız ve basınç) hesaplamaya eklenmesiyle beraber doğru sonuç değerleri arttığı görülmektedir. Örneğin, sadece koordinat sistemi (c) hesaplamaya katıldığında toplam doğru sonuç sayısı 65 iken, tüm biyometrik bilgiler (cvp) hesaplamaya katıldığında toplam doğru sonuç sayısı 75'e çıkmaktadır.



Şekil 4.1. Toplam doğru sonuç dağılımı

Şekil 4.2'de toplam doğru sonuç dağılımının çizim tiplerine göre dağılımı gösterilmektedir. Hemen hemen tüm çizim tiplerinde cvp hesaplama yönteminin doğru sonuca olumlu katkısı görünmektedir. Örneğin, daire çizim tipine bakıldığında sadece koordinat sistemine (c) bakılarak yapılan hesaplamaya göre doğru sonuca ulaşım değeri 3 iken, koordinat, hız ve basınç bilgilerinin (cvp) hepsi birden hesaplamaya katıldığında doğru sonuca ulaşım değeri 8'e ulaşmaktadır.

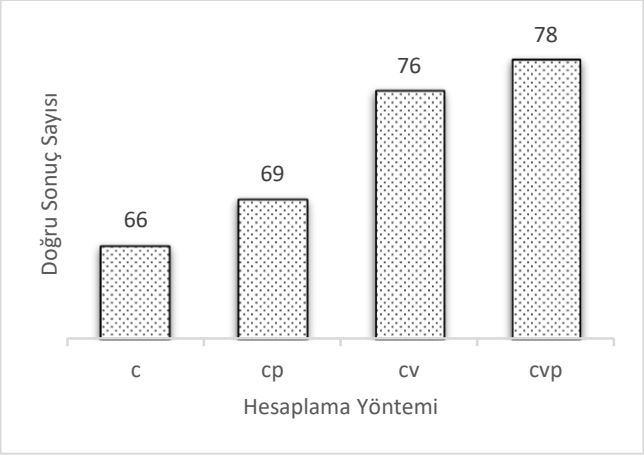


Şekil 4.2. Çizim tiplerine göre doğru sonuç dağılımı

Şekil 4.2'deki İmza, Mn ve Yüz çizim tiplerine bakıldığında, tüm biyometrik bilgiler hesaplamaya katıldığı zaman doğru sonuç değerinin bir miktar düştüğü görünmektedir. İmzadaki düşüşün detayına inildiğinde, doğruluk için gereken yeterli polyline sayısına erişilemediği görülmektedir. Mn ve Yüz deki düşüşlerin detayına inildiğinde, veri seti içerisinde bulunan kişiye ait olan iki adet çizim arasında polyline detayında doğru eşleşmeler olduğu görünmektedir ama iki çizimde doğruluk için gereken yeterli sayıda polyline sayısına erişemediği için algoritma sonuç üretememektedir.

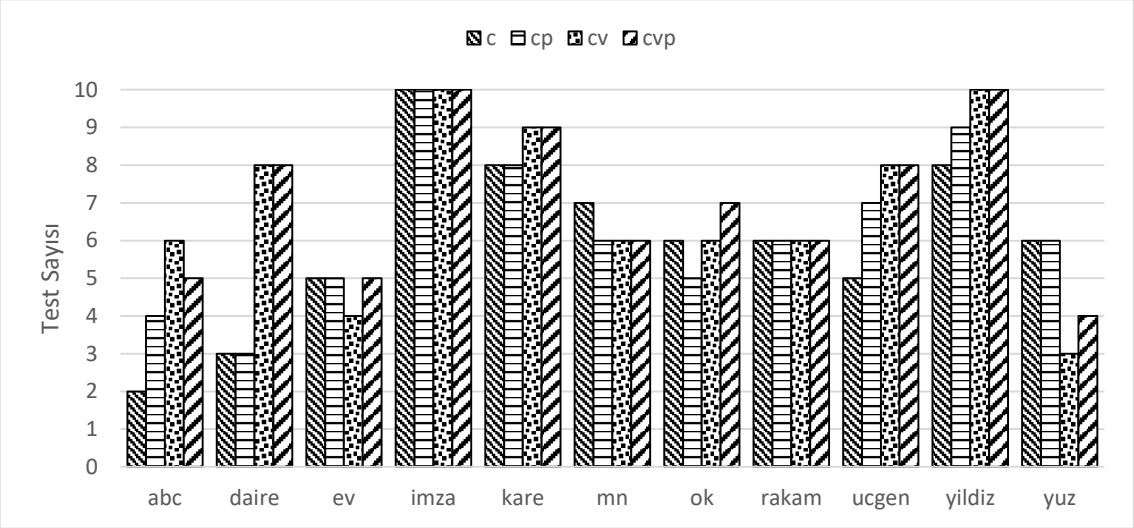
4.1.2 Benzerlik aramanın homojen veri setinde gerçekleştirilmesi

Bu testte, bir önceki testten farklı olarak çizim tipleri hesaplama yöntemi içerisine katılmaktadır. Bu test işlemine giren çizimler, sadece kendi çizim tipleri içerisinde eşleşmeye tabi tutulmaktadır. Örneğin, x kişisinin daire çizimi, veri seti içerisinde bulunan sadece daire çizimleri ile karşılaştırılır. Diğer çizim tipleri hesaplamaya katılmaz. Bu şekilde yaklaşım, çizim tipinin verilmemesine göre daha doğru sonuçlar üretmektir. Şekil 4.1'deki cvp toplam değeri 75'den Şekil 4.3'de 78'e çıkmıştır. Ayrıca, Şekil 4.3'de biyometrik bilgilerin hesaplamaya katılmasının etkileri net bir şekilde görülmektedir. Sadece koordinat sistemine (c) göre eşleşme sonuçları 66 iken, koordinat ve basınç bilgisi (cp) hesaplamaya katıldığında toplam doğru sonuç değeri 69'a, koordinat ve hız bilgisi (cv) hesaplamaya katıldığında toplam doğru sonuç değeri 76'ya, tüm biyometrik bilgiler (cvp) hesaplamaya katıldığında toplam doğru sonuç değeri 78'e çıkmıştır.



Şekil 4.3. Toplam doğru sonuç dağılımı

Şekil 4.4’de toplam doğru sonuç dağılımının çizim tiplerine göre dağılımı gösterilmektedir. Hemen hemen tüm çizim tiplerinde cvp hesaplama yönteminin doğru sonuca olumlu katkısı görünmektedir. Örneğin, üçgen tipindeki çizime bakıldığında; sadece koordinat sistemi (c) hesaba katıldığı zaman doğru sonuç değeri 5 iken, tüm biyometrik (cvp) bilgiler hesaplama yöntemine katıldığı zaman doğru sonuç değeri 8’e yükselmektedir.



Şekil 4.4. Çizim Tiplerine Göre Doğru Sonuç Dağılımı

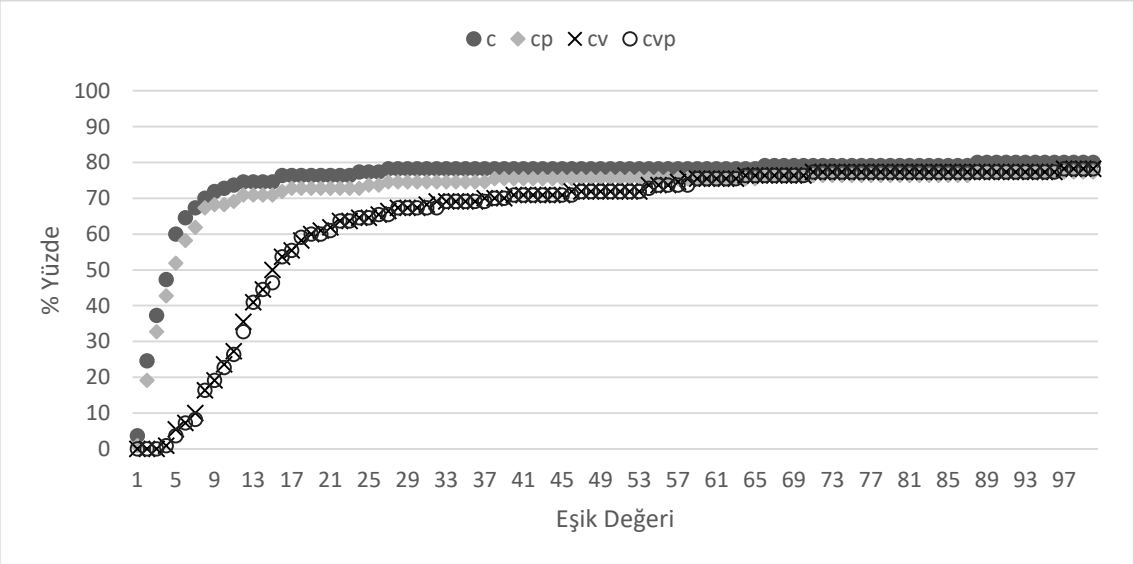
Şekil 4.4’deki Mn ve Yüz çizim tiplerine bakıldığında, tüm biyometrik bilgiler hesaplamaya katıldığı zaman doğru sonuç değerinin bir miktar düştüğü görünmektedir. Mn ve Yüz deki düşüşlerin detayına inildiğinde, veri seti içerisinde bulunan kişiye ait olan iki adet çizim arasında polyline detayında doğru eşleşmeler olduğu görünmektedir ama iki çizimde doğruluk için gereken yeterli sayıda polyline sayısına erişemediği için algoritma sonuç üretememektedir.

4.2 İmza

İmza ile ilgili test işlemleri “Gerçek İmza” ve “Sahte İmza” olmak üzere iki alt başlık altında toplanmıştır.

4.2.1 Gerçek İmza

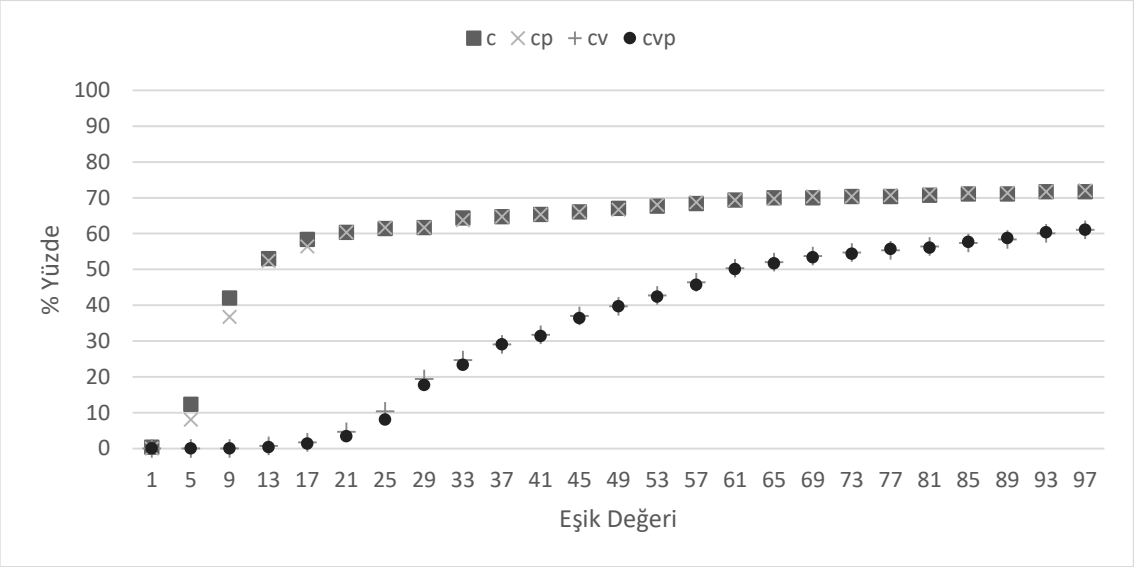
Bu test işleminde, kişinin attığı imzanın veri setinde bulunan tüm imzalar arasından kendi imzası ile doğru bir şekilde eşleşmesi hedeflenmiştir. Bu amaçla karşılaştırma işlemi sırasında bir eşik değer belirlenmesi gerekmektedir. Her bir hesaplama yöntemi ve eşik değeri için 110 adet test işlemi gerçekleştirilmiş olup, Şekil 4.5’deki yüzdesel sonuçlara ulaşmak için toplam 44.000 adet test işlemi gerçekleştirilmiştir. Bu sonuçlara göre, sadece koordinat sisteminin (c) hesaplama sistemine katılması ile doğru sonuca ulaşmak, düşük eşik değerlerinde mümkün olmaktadır ve tüm biyometrik bilgilerin (cvp) hesaplama sistemine katılmasına göre daha yüksek doğru sonuç üretmektedir. Buna rağmen, düşük eşik değerlerinde biyometrik bilgilere göre daha yüksek doğru sonuç üretmenin tek başına yeterli bir ölçüt olmadığı “Sahte İmza” başlığı altında açıklanacaktır.



Şekil 4.5. Eşik değerinin gerçek imzadaki doğru sonuç üretimine etkisi

4.2.2 Sahte İmza

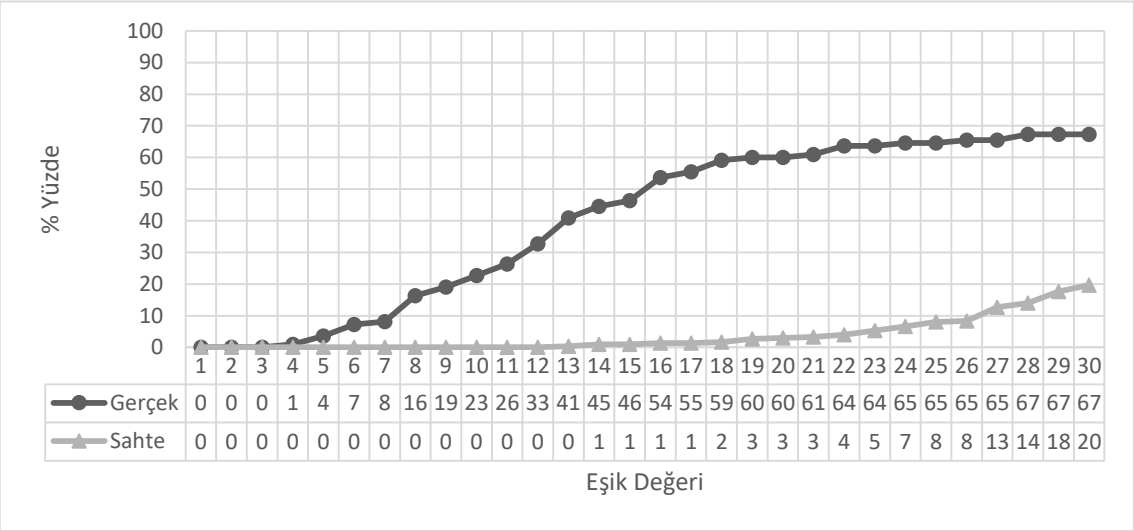
Bu test işlemini gerçekleştirebilmek için genel çizim tiplerinin yapan 10 kişiden diğer 9 kişinin imzasını taklit etmesi istenmiştir. Her bir gerçek imzaya karşılık olarak üç adet sahte imza alınmıştır. Ayrıca bir kişide ilk aşamadaki teste katılan 10 kişinin imzasını taklit etmiştir. Böylece, veri seti olarak toplam 300 adet sahte imza elde edilmiştir. Her bir hesaplama yöntemi ve eşik değeri için 300 adet test işlemi gerçekleştirilmiş olup, Şekil 4.6'daki yüzdesel sonuçlara ulaşmak için toplam 120.000 adet test işlemi gerçekleştirilmiştir. Bu sonuçlara göre, sadece koordinat sistemi (c) hesaplamaya katıldığında düşük eşik değerlerinde sahte imza gerçek imza ile eşleşme yapabilmesine rağmen tüm biyometrik bilgiler (cvp) hesaplamaya katıldığında eşleşme işlemi daha yüksek eşik değerlerinde olmaktadır. Buradan çıkarılacak sonuç ise sahte imzayı yapan kişi imzayı görsel olarak taklit edebilmektedir ama imza atılırken elin yaptığı çizim hızı ve dokunmatik yüzeye uyguladığı basınç bilgisini genellikle belirli bir eşik değerine kadar taklit edememektedir.



Şekil 4.6. Eşik değerinin sahte imzadaki doğru sonuç üretimine etkisi

Şekil 4.7'de cvp hesaplama yöntemine göre gerçek imzayı atan ile sahte imzayı atan kişilerin doğru sonuç üretme yüzdelerinin karşılaştırılması gösterilmektedir. Bu grafikte; eşik değeri 13 olana kadar, sahte imzayı atan kişinin eşleşme yüzdesi 0 iken,

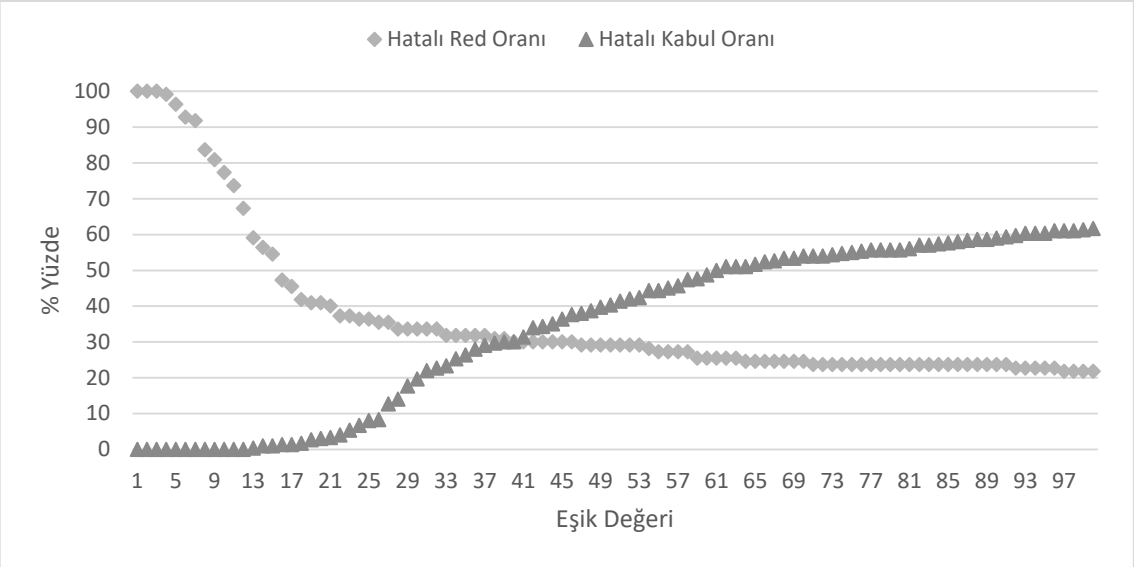
gerçek kişinin veri setindeki imzası ile eşleşme oranı % 41'dir. Eşik değeri 26'dan sonra sahte imzanın eşleşme yüzdesi hızlı bir şekilde artmaya başlamaktadır.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Gerçek	0	0	0	1	4	7	8	16	19	23	26	33	41	45	46	54	55	59	60	60	61	64	64	65	65	65	65	67	67	67
Sahte	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2	3	3	3	4	5	7	8	8	13	14	18	20

Şekil 4.7. Eşik değerlerine göre gerçek ve sahte imzaların doğru sonuç üretme yüzdesinin karşılaştırılması

Şekil 4.8'de hatalı red oranı (False Rejection Rate – FRR) ile hatalı kabul oranının (False Acceptance Rate – FAR) karşılaştırılması yapılmaktadır. İmzayı atan gerçek kişinin imzası veri setinde bulunan imzası ile eşleşmesi gerekmektedir ama belirlenen eşik değeri sebebiyle, atılan imza hatalı bir şekilde reddedilmektedir. Bu hatalı durum, hatalı red oranı olarak ifade edilmektedir. Bu yaklaşımın zıttı olarak da; hatalı kabul oranı, sahte imza atan kişinin belirlenen eşik değerine göre veri setindeki gerçek imza ile eşleşmemesi gerekirken eşleşmesini ifade etmektedir. Şekil 4.8'daki sonuçlara göre eşik değeri olarak 40 önemli bir noktadır. Bu değerden sonra sahte imzalar hatalı bir şekilde daha fazla kabul olmaya, gerçek imzalar da hatalı bir şekilde daha fazla reddedilmeye başlamıştır.



Şekil 4.8. Hatalı red oranı ile hatalı kabul oranın karşılaştırılması

BÖLÜM 5. SONUÇ

5.1 Yargı

Bu çalışma içerisinde, çizim esnasında alınan x ve y koordinatlarına ek olarak alınan elin yaptığı çizim hızı bilgisinin ve elin yüzeye yaptığı basınç bilgisinin hesaplama yöntemine katılarak daha iyi sonuçlar üretmesi hedeflenmiştir. Yapılan geliştirmenin sadece koordinat sistemine göre yapılan eşleşmeye göre daha iyi sonuçlar ürettiği, gönüllülerden toplanan ve çeşitli çizimleri içeren veri seti üzerinde yapılan test sonuçlarında görünmektedir. Ayrıca, biyometrik bilgilerin daha belirgin bir şekilde ortaya çıktığı imza gibi kişiye özel çizimlerde daha doğru sonuçlar ürettiği de görünmektedir. Bu bilgiler ışığında, doğru sonuç üretimini polyline sayısı ve kutupsal koordinat sistemi diğer biyometrik bilgilere göre daha fazla etkilemektedir. Sonrasında, sırasıyla hız ve basınç bilgileri gelmektedir.

5.2 Öneriler

Gelecekte polyline sayısının sonuçlara etkisi araştırılıp yapılan geliştirmenin daha iyi bir konuma gelmesi sağlanabilir. İkinci olarak, farklı filtreleme metotları kullanılarak daha iyi sonuçlar elde edilip edilemeyeceği araştırılabilir. Üçüncü olarak da, el ile çizilen dijital imzanın bir sisteme giriş mekanizması veya sistemde onay mekanizması olması için bu çalışma içerisinde yapılan geliştirmenin alt yapısı kullanılabilir.

BÖLÜM 6. EK-1

Yapılan geliştirmenin kaynak kodları, kütüphane isimleri esas alınarak aşağıdaki alt başlıklarda verilmiştir. Geliştirme ortamı işletim sistemi olarak Windows 10, programlama dili olarak Python 3.7.0 ve geliştirme aracı olarak Spyder 3.3.2 kullanılmıştır. Program hem Windows ortamlarda hem de Linux ortamlarda çalışabilecek şekilde kodlanmıştır.

6.1 Ana Program (Run_Program)

Program bu kaynak kod üzerinden yönlendirilmektedir ve veri setine dosya eklenmesi ve karşılaştırma işleminin yapılması şeklinde iki çeşit çalışma yöntemi bulunmaktadır.

6.1.1 Veri Setine Dosya Eklenmesi

Veri setine dosya eklenirken "--import" parametresi kullanılmaktadır ve import paramteresi ile dosya veya dizin belirtilerek program çalıştırılabilmektedir. İsteğe göre bu dosyaların kime ait olduğunu belirtmek için "--id" parametresi ve çizimin tipi ile ilgili bilgi eklemek için "--type" parametresi kullanılabilir. Örnek kullanım aşağıda verilmiştir:

```
RUN_PROGRAM.py --import ./Import/ --id 10 --type letter
```

6.1.2 Karşılaştırma işleminin yapılması

Veri seti içerisindeki çizimler ile karşılaştırma yapabilmek için "--file" parametresi ve sonrasında dosyanın dizini belirtilmektedir. Yapılacak karşılaştırmanın hangi biyometrik bilgi ile yapılacağı "c" (koordinat), "v" (hız) ve "p" (basınç) ile belirtilmektedir. Koordinat sabit kalmak şartıyla diğer biyometrik bilgiler karşılaştırma parametresi olarak verilebilmektedir. Örneğin, cv, cp ve cvp gibi. İsteğe bağlı olarak yapılan karşılaştırma işlemine kişi bilgisini belirten "--id" parametresi verilebilmektedir. Böylece, sadece o kişiye ait olan veri seti içerisindeki çizimler arasında karşılaştırma yapılabilmektedir. Benzer şekilde, çizim tipi belirtilmek istendiğinde "--type" parametresi kullanılmaktadır. İsteğe bağlı olarak, kişiye ait olan çizim karşılaştırılırken

eşik değeri “--t” parametresi ile belirlenebilmektedir. Tüm parametrelerin olduğu en genel kullanım aşağıda verilmiştir:

```
      RUN_PROGRAM.py --file ./Input/Test/Kare1.svg -cvp --id 10  -
-t 11 --type letter
```

Programın yardım dokümanına, “--help” parametresi yazılarak ya da her hangi bir parametre verilmeden ulaşılabilir. Örnek kullanımı aşağıda verilmiştir:

```
      RUN_PROGRAM.py --help
```

Ana programın kaynak kodu aşağıda verilmiştir:

```
import matplotlib.pyplot as plt # For Figure
import sys
import os
import time
import InputOutput as io
import config as cnfg
import computations as cmpt
import figure
import check_dtw as dtw
import argparse


######################
# Run Comparison     #
######################
def run_comparison(svg_file_full_path):

    # Read Library Data
    library_data = io.read_library()

    # Preperation for Output Files
    output_file, output_file_figure  =
io.prepare_output_files(svg_file_full_path, 'file')
    print ("Output Figure (PNG): " , output_file_figure)

    # Write Output
    temp_stdout = sys.stdout
    file = open(output_file, "w")
    sys.stdout = file
```

```
    # Parse SVG File
    points_strings, pressure_strings =
io.pars_svg_file(svg_file_full_path)

    # Check Polyline Amount between File and Library
    cmpt.check_polyline_number(points_strings, library_data)


    # Find Feature of Points (Angle, Speed, Pressure)
    polyline_number = 1
    points_x_y_list = []
    points_feature_list = []
    filtfilt_butter_list = []

    file_angle_merge = []

    for x in range(len(points_strings)):
        cmpt.print_polyline(polyline_number)
        points_string = points_strings[x]

        # Check Empty Pressure
        if len(pressure_strings) >= x+1:
            pressure_string = pressure_strings[x]
        else:
            pressure_string = ''

        # Get Points
        points_x_y =
cmpt.get_points(points_string,polyline_number)
        cmpt.print_number_of_points(points_x_y[:,0])
        points_x_y_list.append(points_x_y)

        # Find Feature of Points (Angle, Speed)
        # Return points_feature[angle, speed]
        points_feature = cmpt.find_feature(points_x_y)

        # Get Pressure
        points_feature["pressure"] =
cmpt.get_pressure(pressure_string)

        # Filtration (Angle, Speed, Pressure)
        # Return points_feature[angle_filter, speed_filter,
pressure_filter]
        points_feature, filtfilt_butter =
cmpt.filter_feature(points_feature)
```

```
            points_feature_list.append(points_feature)
            filtfilt_butter_list.append(filtfilt_butter)

            file_angle_merge.extend(points_feature['angle'])

            polyline_number +=1

        # Select TopN in Library (Angle)
        library_data_length =len(library_data)
        cmpt.print_polyline("All")
        cmpt.print_number_of_points(file_angle_merge)
        if (library_data_length <= cnfg.TopN):
            print(f"Omit Select TopN in Library -
length(library_data):{library_data_length}")
        else:
            library_data = dtw.library_check_dtw(library_data,
file_angle_merge)

        # Preparation for Figure
        polyline_number = 1
        figure_plot = len(points_strings)*100 + 11
        if cnfg.figure == True:
            plt.figure(1,figsize=(8, 4*len(points_strings)),
dpi=cnfg.figure_dpi, facecolor='w', edgecolor='k')

        best_list = []

        for x in range(len(points_strings)):
            cmpt.print_polyline(polyline_number)

            points_feature = points_feature_list[x]
            filtfilt_butter = filtfilt_butter_list[x]

            # Check DTW
            # Return best[distance, name, path, angle_distance, speed,
polyline, file_angle_filter, file_speed, file_speed_filter,
file_speed_distance]
            best = dtw.check_dtw(library_data, points_feature,
polyline_number)
            if (best != {}):
                best['filtfilt_Angle'] = filtfilt_butter['Angle']
                best['filtfilt_Speed'] = filtfilt_butter['Speed']
                best['filtfilt_Pressure'] =
filtfilt_butter['Pressure']
                best_list.append(best)
```

```
        # Assignment of Figure Features
        if cnfg.figure == True:

figure.assign_figure_feature(polyline_number,figure_plot,
points_feature, points_x_y)

        polyline_number +=1
        figure_plot +=1

    ##plt.subplots_adjust(top=0.92, bottom=0.08, left=0.20,
right=0.95, hspace=0.45, wspace=0.35)
    ##plt.show(block=False)
    ##plt.ion()
    if cnfg.figure == True:
        plt.savefig(output_file_figure)

    cmpt.find_best(best_list, library_data, points_x_y_list,
temp_stdout)

    file.close() # Close the output file

######################
# Run Import Library #
######################
def run_import_library(svg_file_full_path):

    # Preperation for Output Files
    output_file, output_file_figure =
io.prepare_output_files(svg_file_full_path, 'import')

    # Write Output
    temp_stdout = sys.stdout
    file = open(output_file, "w")
    sys.stdout = file

    # Parse SVG File
    points_strings, pressure_strings =
io.pars_svg_file(svg_file_full_path)

    coords = {}
    angles = {}
    pressures = {}
    speeds = {}
    polyline_number = 1
```

```
    for x in range(len(points_strings)):
        points_string = points_strings[x]

        # Check Empty Pressure
        if len(pressure_strings) >= x+1:
            pressure_string = pressure_strings[x]
        else:
            pressure_string = ''

        # Get Points
        points_x_y =
cmpt.get_points(points_string,polyline_number)
        coords['polyline' + str(polyline_number)] = points_string

        # Find Feature of Points (Angle, Speed)
        # Return points_feature[angle, speed]
        points_feature = cmpt.find_feature(points_x_y)

        # Get Pressure
        points_feature["pressure"] =
cmpt.get_pressure(pressure_string)

        # Filtration (Angle, Speed, Pressure)
        # Return points_feature[angle_filter, speed_filter,
pressure_filter]
        points_feature, filtfilt_butter =
cmpt.filter_feature(points_feature)

        # Assign Angle, Pressure and Speed for JSON
        angles['polyline' + str(polyline_number)] =
str(points_feature['angle_filter'].tolist()).strip('[]')
        speeds['polyline' + str(polyline_number)] =
str(points_feature['speed_filter'].tolist()).strip('[]')
        pressures['polyline' + str(polyline_number)] =
str(points_feature['pressure_filter'].tolist()).strip('[]')

        polyline_number +=1

    io.write_library(coords, angles, pressures, speeds,
polyline_number-1)

    sys.stdout = temp_stdout
    file.close() # Close the output file

if __name__ == '__main__':
```

```
    parser = argparse.ArgumentParser(description='Sketch
Matcher',\

formatter_class=argparse.RawTextHelpFormatter, epilog=('''
Example of Use:
 Compare File: python RUN_PROGRAM.py --file ./Input/Test/Kare1.svg
-cvp --id 10  --t 11 --type letter
 Import Files to Library: python RUN_PROGRAM.py --import ./Import/
--id 10 --type letter'''))
    parser.add_argument('--file', action='store',
dest='file_destination', help='Compare File')
    parser.add_argument('--import', action='store',
dest='import_directory', help='Import Files to Library')
    parser.add_argument('--id', action='store', dest='id',
default=0, help='Import ID & Comparison ID (Optional)')
    parser.add_argument('--type', action='store', dest='type',
default='letter', help='Type (Optional)')
    parser.add_argument('--t', action='store', dest='threshold',
default=10, help='Threshold (Optional)')
    parser.add_argument('-c', action='store_true',
dest='arg_coordinate', default=False, help='For Angle
Computation')
    parser.add_argument('-v', action='store_true',
dest='arg_speed', default=False, help='For Speed Computation')
    parser.add_argument('-p', action='store_true',
dest='arg_pressure', default=False, help='For Pressure
Computation')
    parser.add_argument('--nofigure', action='store_false',
dest='arg_figure', default=True, help='No Figure (Optional)')
    parser.add_argument('--version', action='version',
version='%(prog)s '+ str(cnfg.program_version))
    args = parser.parse_args()

    #Ignore Warnings
    import warnings
    warnings.simplefilter("ignore")

    if args.file_destination is not None or args.import_directory
is not None:
        start = time.time()
        cnfg.id = args.id
        cnfg.type = args.type

        # Run Comparison
        if args.file_destination is not None:
```

```
            cnfg.computation_angle = args.arg_coordinate
            cnfg.computation_speed = args.arg_speed
            cnfg.computation_pressure = args.arg_pressure
            cnfg.figure = args.arg_figure
            cnfg.dtw_sum_threshold = int(args.threshold)

            print("\nAngle Computation:", cnfg.computation_angle)
            print("Speed Computation:", cnfg.computation_speed)
            print("Pressure Computation:",
cnfg.computation_pressure)
            print("Threshold:", cnfg.dtw_sum_threshold)
            print("Comparison ID:", cnfg.id)
            print("Comparison Type:", cnfg.type)
            print("Figure:", cnfg.figure)

            svg_file_full_path =
io.check_windows_backslashes(args.file_destination)
            print ("SVG Input File: " , svg_file_full_path)
            run_comparison(svg_file_full_path)

        # Run Import Library
        else:
            print("Import ID:", cnfg.id)
            svg_file_full_path =
io.check_windows_backslashes(args.import_directory)
            cnfg.gv_output_directory = './Import_Output/'

            # Only File
            if svg_file_full_path.endswith('.svg'):
                run_import_library(svg_file_full_path)
            # Directory
            else:
                for file in os.listdir(svg_file_full_path):
                    if file.endswith('.svg'):
                        run_import_library(svg_file_full_path +
"/" + file )

        end = time.time()
        print(f"\nProgram Execution Time:",end - start)
    else:
        parser.print_help()
```

6.2 config kütüphanesi

Ana program tarafından parametrik bilgilerin alınması için oluşturulmuştur. Kaynak kodları aşağıda verilmiştir:

```
gv_library_data = 'library.json'
gv_output_directory = './Output/'
gv_output_file_prefix = ''
log_file = 'logfile.csv'
TopN = 3
simplify_number = 3
program_version = 11.3
computation_angle = False
computation_speed = False
computation_pressure = False
id = 0
type = 'letter'
dtw_sum_threshold = 1


# Figure Options
figure_dpi = 120
figure = True


# Format Figures
fig_angle_ylabel='Açı - Angle'
fig_angle_legend=['Orjinal açı - Original angle', 'Filtfilt']
```

6.3 InputOutput kütüphanesi

Veri giriş çıkışlarının (IO) yapıldığı kütüphanedir. Gerekli durumlarda ana prorgram tarafından çağrılmaktadır. Kaynak kodları aşağıda verilmiştir:

```
import os,sys
import json
import config as cnfg
import time

# Convert backslashes to forward slashes (For Windows)
def check_windows_backslashes(svg_file_full_path):
    if sys.platform == 'win32' and
svg_file_full_path.find("\\")>0:
        svg_file_full_path = svg_file_full_path.replace(os.sep,
'/')
    return svg_file_full_path


# Read Library Data
def read_library():
    with open(cnfg.gv_library_data) as file:
        library_data = json.load(file)
    file.close()
    return library_data


# Write Library Data
def write_library(coords, angles, pressures, speeds,
polyline_count):

    data = []
    data.append({ 'ID':cnfg.id, 'name':cnfg.gv_output_file_prefix,
'type':cnfg.type, 'count':polyline_count, 'coords':coords,
'angles':angles, 'speed':speeds, 'pressure':pressures })

    if not os.path.isfile(cnfg.gv_library_data):
        with open(cnfg.gv_library_data, mode='w') as file:
            file.write(json.dumps(data, indent=2))
    else:
        library_data = read_library()

        library_data.extend(data)

        with open(cnfg.gv_library_data, mode='w') as file:
            file.write(json.dumps(library_data, indent=2))
```

```
    file.close()


# Preparation for Output Files
def prepare_output_files(svg_file_full_path, import_file):

    svg_file_name = svg_file_full_path.split("/")
    cnfg.gv_output_file_prefix = svg_file_name[-1].split(".")[0] #
Example : cizim_3_adet.svg -> cizim_3_adet

    if import_file == 'file':
        now = time.strftime("%Y%m%d_%H%M%S") # Example output:
20181121_004308
        cnfg.gv_output_directory = cnfg.gv_output_directory +
cnfg.gv_output_file_prefix + "/" + now + "/"


    if not os.path.exists(cnfg.gv_output_directory):
        os.makedirs(cnfg.gv_output_directory)

    output_file = cnfg.gv_output_directory +
cnfg.gv_output_file_prefix + ".csv"
    output_file_figure = cnfg.gv_output_directory +
cnfg.gv_output_file_prefix + ".png"

    print ("\nOutput File (CSV): " , output_file)
    return output_file, output_file_figure


# Parse SVG File
def pars_svg_file(svg_file_full_path):
    from xml.dom import minidom # For Parsing

    doc = minidom.parse(svg_file_full_path)

    points_strings = [path.getAttribute('points') for path
                      in doc.getElementsByTagName('polyline')]

    pressure_strings = [path.getAttribute('point') for path
                      in doc.getElementsByTagName('pressure')]

    doc.unlink()
    print ("Total number of polyline: " , len(points_strings))
    return points_strings, pressure_strings
```

6.4 computations kütüphanesi

Biyometrik bilgilerin hesaplanması, filtrelenmesi gibi hesaplama işlemlerinin yapıldığı kütüphanedir. Gerekli durumlarda ana prorgram tarafından çağrılmaktadır. Kaynak kodları aşağıda verilmiştir:

```
import numpy as np
import math
from scipy.signal import filtfilt, butter # For Filter
import figure
import config as cnfg


# Check Polyline Amount between File and Library
def check_polyline_number(points_strings, library_data):
    print("\n# Check Major Features between File and Library")
    polyline_number_file = len(points_strings)

    for sketch in library_data[:]: # [:] Generate a Copy
        polyline_number_sketch = sketch['count']

        # Check Default ID
        if cnfg.id !=0:
            sketch_id = sketch['ID']
            # Check Skectch ID
            if cnfg.id != sketch_id:
                sketch_name = sketch['name']
                library_data.remove(sketch)
                print(f"The sketch has been removed from library
(ID):{sketch_name} ID:{sketch_id}")
                continue

        # Check Default Type
        if cnfg.type != 'letter':
            sketch_type = sketch['type']
            # Check Skectch Type
            if cnfg.type != sketch_type:
                sketch_name = sketch['name']
                library_data.remove(sketch)
                print(f"The sketch has been removed from library
(Type):{sketch_name} Type:{sketch_type}")
                continue

        # Check Poyline Number
        if (polyline_number_sketch != polyline_number_file):
```

```
            sketch_name = sketch['name']
            library_data.remove(sketch)
            print(f"The sketch has been removed from library
(Polyline):{sketch_name} Poyline Number:{polyline_number_sketch}")


def print_polyline(polyline_number):
    print("\n################")
    print("# " + str(polyline_number)+ ". Polyline #")
    print("################")


def print_number_of_points(points_x_y):
    print("Total number of points:" ,len(points_x_y))


def search_file_type(file):

    search_list = ['kare', 'ucgen', 'mn', 'daire', 'ok', 'ev',
'yildiz', 'abc', 'yuz', 'rakam', 'imza']

    find = False
    for search_type in search_list:
        result = file.lower().find(search_type)
        if result != -1:
            find = True
            break;

    if find == False:
        search_type = 'letter'

    return search_type

# Get Points
def get_points(points_string,polyline_number):
    points_list = points_string.split(" ") # Example
points_string: '216.5,176.5 216.5,174.5'

    points = list() # Create empty list for first insertion
    for point in points_list:
        if len(point) != 0:  # Omit empty point: " "
            points += point.split(",") # Example point:
'216.5,176.5'

    points_x_y = np.asarray(points).reshape(-1,2) # 2D array
conversion
```

```
#    print("Total number of points:" ,len(points_x_y[:,0]))

    return points_x_y


#Get Pressure
def get_pressure(pressure_string):
    pressure_list = pressure_string.split(" ")

    for x in range(len(pressure_list)):
        if len(pressure_list[x]) == 0:
            del pressure_list[x]  # Delete empty pressure: " "
        else:
            pressure_list[x] = float(pressure_list[x])

    return pressure_list


# Find Feature of Points (Angle, Speed)
def find_feature(points_x_y):
    points_feature = {}
    points_feature['angle'] = list()
    points_feature['speed'] = list()

    print ("Coordinates of points(x,y) , radian , angle, speed:")
    range_number = len(points_x_y)-1 # Total number of lines,
minus last line (point)
    for x in range(range_number):

        x1 = float(points_x_y[x][0])
        y1 = float(points_x_y[x][1])

        x2 = float(points_x_y[x+1][0])
        y2 = float(points_x_y[x+1][1])

        # Find Speed
        speed = math.sqrt( (x2 - x1)**2 + (y2 - y1)**2 )

        # Fine Angle
        point_radian = math.atan2(y2-y1, x2-x1)
        point_degree = math.degrees(point_radian)

        # Positive Direction
        if point_degree > 0:
            point_degree = 360 - point_degree
        elif point_degree != 0:
```

```
            point_degree = -1*point_degree

        print(f"{x+1}: ({x1},{y1}) {point_radian}, {point_degree},
{speed}")
        points_feature['angle'].append(point_degree)
        points_feature['speed'].append(speed)

    # Correction of Zeros
    print("\nCorrection of Zeros:")
    range_number = len(points_feature['angle'])-1 # Total number
of degrees, minus last degree
    for x in range(1,range_number):
        if points_feature['angle'][x] == 0 and
points_feature['angle'][x+1] != 0 and points_feature['angle'][x-1]
!= 0: # Example degrees (90,0,92) -> (90,91,92)
            points_feature['angle'][x] =
(points_feature['angle'][x-1] + points_feature['angle'][x+1])/2
            print("Point:",x, "-
Degree:",points_feature['angle'][x])
        elif points_feature['angle'][x] != 0 and
points_feature['angle'][x+1] == 0 and points_feature['angle'][x-1]
== 0: # Example degrees (0,350,0) -> (0,0,0)
            points_feature['angle'][x] = 0
            print("Point:",x, "-
Degree:",points_feature['angle'][x])

    print("\nAngles: ", points_feature['angle'])
    print("\nSpeeds: ", points_feature['speed'])
    return points_feature


# Filtration (Angle, Speed, Pressure)
def filter_feature(points_feature):

    points_angle = points_feature['angle']
    points_speed = points_feature['speed']
    pressure_list = points_feature["pressure"]

    butter_angle = 'butter(1, 0.4)'
    butter_speed = 'butter(1, 0.1)'
    butter_pressure = 'butter(1, 0.4)'

    b, a = eval(butter_angle) # For Angle
    c, d = eval(butter_speed) # For Speed
    e, f = eval(butter_pressure) # For Pressure
```

```
    filtfilt_butter = {}
    filtfilt_butter['Angle'] = butter_angle
    filtfilt_butter['Speed'] = butter_speed
    filtfilt_butter['Pressure'] = butter_pressure

    # Check minimum length for Filter (Angle,Speed)
    if (len(points_angle) > 6):
        points_angle_filter = filtfilt(b, a, points_angle)
        points_speed_filter = filtfilt(c, d, points_speed)

        print("\nAngles (Filter): ", points_angle_filter.tolist())
        print("\nSpeeds (Filter): ", points_speed_filter.tolist())

    else:
        points_angle_filter = np.array(points_angle).astype(float)
        points_speed_filter = np.array(points_speed).astype(float)
        points_pressure_filter =
np.array(pressure_list).astype(float)

        print("\nAngles (Filter): NOT MADE")
        print("\nSpeeds (Filter): NOT MADE")

    # Check minimum length for Filter (Pressure)
    if (len(pressure_list) > 6):
        points_pressure_filter = filtfilt(e, f, pressure_list)
        print("\nPressures (Filter): ",
points_pressure_filter.tolist())
    else:
        points_pressure_filter =
np.array(pressure_list).astype(float)
        print("\nPressures (Filter): NOT MADE")

    points_feature['angle_filter'] = points_angle_filter
    points_feature['speed_filter'] = points_speed_filter
    points_feature["pressure_filter"] = points_pressure_filter

    return points_feature, filtfilt_butter


# Find Best
def find_best(best_list, library_data, points_x_y_list,
temp_stdout):
    import collections
    import sys
    import time
    import os
```

```
    start = time.time()

    # Log File (Header)
    if not os.path.isfile(cnfg.log_file):
        file = open(cnfg.log_file, mode='w')

file.write("Time;File_Name;Poyline;Type;Comparison_ID;Angle;Speed;
Pressure;CVP_ID;Threshold;Result_Flag;Match_ID_Flag;Type_Flag;Corr
ect_Flag;Result_File_Name;File_ID;Poyline\n")
    else:
        file = open(cnfg.log_file, "a")

    polyline_number = len(points_x_y_list)
    best_names = []
    for best in best_list:
        best_names.append(best['name'])
    counter_list = collections.Counter(best_names) # Find polyline
count per skectch

    best_sketch = ''
    for i in counter_list.keys():
        if ( polyline_number-1 <= counter_list[i]): # Check
polyline count

            # Find polyline count for best_sketch
            for lib_sketch in library_data:
                if( lib_sketch['name'] == i):
                    sketch_polyline_count = lib_sketch['count']
                    sketch_id = lib_sketch['ID']
                    sketch_type = lib_sketch['type']

            # Check polyline count
            if (sketch_polyline_count-1 <= polyline_number):
                if polyline_number == 2 and counter_list[i] == 1:
                    # For Single Point
                    if( len(best_list[0]['angle']) or
len(best_list[1]['angle']) ):
                        for best in best_list:
                            if(best['name']==i and
len(best['angle']) > 1):
                                best_sketch = i
                else:
                    best_sketch = i

    if (best_sketch == ''):
```

```
        end = time.time()
        print(f"\nExecution Time for Find Best:",end - start)

        sys.stdout = temp_stdout
        print(f"\nOptimum Solution For
{cnfg.gv_output_file_prefix}: NON")

        # For Log
(Result_Flag;Match_ID_Flag;Type_Flag;Correct_Flag)
        log_result = '0;0;0;0'

    else:
        # Figure
        if cnfg.figure == True:
            for best in best_list:
                if( best['name'] == best_sketch ):

                    # DTW - Figure (Angle, Speed, Pressure)
                    figure.dtw_figure(best)

                    # Filter - Figure (Speed)
                    if cnfg.computation_speed == True:
                        if( len(best['speed']) != 0):
                            figure.feature_figure(best['name'],
'speed', best['polyline'], best['file_speed'],
best['file_speed_filter'], best['speed'],best['filtfilt_Speed'])

                    # Filter - Figure (Pressure)
                    if cnfg.computation_pressure == True:
                        if( len(best['pressure']) > 1):
                            figure.feature_figure(best['name'],
'pressure', best['polyline'], best['file_pressure'],
best['file_pressure_filter'], best['pressure'],
best['filtfilt_Pressure'])

            figure.best_figure(points_x_y_list, best_sketch,
library_data)

        end = time.time()
        print(f"\nExecution Time for Find Best:",end - start)

        sys.stdout = temp_stdout

        # Find ID
        try:
            import re
```

```
            file_id = re.findall(r'-?\d+\.?\d*',
cnfg.gv_output_file_prefix[28:30])[0]
        except:
            file_id = '-'

        print(f"\nOptimum Solution For
{cnfg.gv_output_file_prefix} (Polyline Count:{polyline_number} -
ID:{file_id}):")
        print(f"Skecth Name: {best_sketch} (Polyline Count:
{sketch_polyline_count} - Match Count: {counter_list[best_sketch]}
- ID:{sketch_id})")

        # For Log (Match_ID_Flag)
        if (sketch_id == file_id):
            match_id_flag = '1'
        else:
            match_id_flag = '0'

        # For Log (Type_Flag)
        if (sketch_type == cnfg.type):
            type_flag = '1'
        else:
            file_type =
search_file_type(cnfg.gv_output_file_prefix)
            if (sketch_type == file_type):
                type_flag = '1'
            else:
                type_flag = '0'


        # For Log (Correct_Flag)
        if (match_id_flag == '1' and type_flag == '1'):
            correct_flag = '1'
        else:
            correct_flag = '0'

        # Log
(Result_Flag;Match_ID_Flag;Type_Flag;Correct_Flag;Result_File_Name
;File_ID;Poyline)
        result_flag = '1'
        log_result = result_flag +";"+ match_id_flag +";"+
type_flag +";"+ correct_flag +";"+ best_sketch +";"+
str(sketch_id) +";"+ str(counter_list[best_sketch])

    # For Log (CVP_ID)
    if cnfg.computation_angle == True:
```

```
        cvp_id = "c"
    if cnfg.computation_speed == True:
        cvp_id = cvp_id + "v"
    if cnfg.computation_pressure == True:
        cvp_id = cvp_id + "p"


    # Write Log
(Time;File_Name;Poyline;Type;Comparison_ID;Angle;Speed;Pressure;cv
p_id;Threshold)
    file_data = time.strftime("%Y%m%d_%H%M%S") +";"+
cnfg.gv_output_file_prefix +";"+ str(polyline_number) +";"+
cnfg.type +";" \
        + str(cnfg.id) +";"+ str(cnfg.computation_angle) +";"+
str(cnfg.computation_speed) +";"+ str(cnfg.computation_pressure)
+";" \
        + cvp_id +";"+ str(cnfg.dtw_sum_threshold) +";"+
log_result + "\n"
    file.write(file_data)
```

6.5 check_dtw kütüphanesi

Dynamic Time Warping (DTW) işlemlerinin yapıldığı kütüphanedir. Gerekli durumlarda ana prorgram tarafından çağrılmaktadır. Kaynak kodları aşağıda verilmiştir:

```
import numpy as np
import time
import dtw
from pylab import norm
from scipy.spatial.distance import euclidean
import config as cnfg

from fastdtw import fastdtw


# Check DTW for Feature (Angle, Speed, Pressure)
def check_dtw_feature( feature_name, name, file_feature_filter,
feature):
    x = file_feature_filter.reshape(-1, 1)
    y = feature.reshape(-1, 1)

    start = time.time()
    distance, cost, acc, path = dtw.fastdtw(x, y, dist=lambda x,
y: norm(x - y, ord=1))
```

```
    end = time.time()

    print(f"{name} - {cnfg.gv_output_file_prefix} ({feature_name}
- Distance): {distance} - DTW Exec. Time:",end - start)

    return distance, path


def check_fastdtw(x, y, lib_sketch_name, gv_output_file_prefix):
    start = time.time()
    distance, path = fastdtw(x, y, dist=euclidean)
    end = time.time()

    print(f"{lib_sketch_name} - {gv_output_file_prefix} (Angle -
Euclidean - Distance): {distance} - DTW Exec. Time:",end - start)

    return distance, path


# Check DTW for Angle
def check_dtw( library_data, points_feature, polyline_number):
    start = time.time()

    print("\n# Check DTW:")
    print("Sketch Name - Distance")

    best = {}
    best_list = []

    for sketch in library_data:
        sketch_name = sketch['name']

        if polyline not in sketch['angles']:
            print(sketch_name, "-",cnfg.gv_output_file_prefix,"
(NON)",polyline)
            continue

        # Check Angle in File (Single Point)
        if len(points_feature['angle_filter']) == 0:
            if len(sketch['angles'][polyline]) == 0:
                best['polyline'] = polyline
                best['name'] = sketch_name
                best['angle'] = sketch['angles'][polyline]
                best['speed'] = sketch['speed'][polyline]
                if len(sketch['pressure'][polyline]) == 0:
```

```
                    best['pressure'] =
sketch['pressure'][polyline]
                else:
                    best['pressure'] =
np.array(sketch['pressure'][polyline].split(",")).astype(float)
                best['file_angle_distance'] = 0
                best['file_speed_distance'] = 0
                best['file_pressure_distance'] = 0
                best_list.append(best.copy())
                print(f"{sketch_name} -
{cnfg.gv_output_file_prefix} (Single Point)")
            else:
                print(f"{sketch_name} -
{cnfg.gv_output_file_prefix} (Single Point): (NON)")
            continue

        # Check Angle in Library (Single Point)
        if len(sketch['angles'][polyline]) == 0:
            continue


        best['polyline'] = polyline
        best['name'] = sketch_name
        # Angle
        if(cnfg.computation_angle == True):
            sketch_raw = sketch['angles'][polyline].split(",")
            best['angle'] = np.array(sketch_raw).astype(float)
            best['file_angle_filter'] =
points_feature['angle_filter']
            best['file_angle_distance'],
best['file_angle_dtw_path'] = \
                check_dtw_feature( 'Angle - Norm', best['name'],
best['file_angle_filter'], best['angle'])
        else:
            best['file_angle_distance'] = 0


        # Speed
        if(cnfg.computation_speed == True):
            best['speed'] =
np.array(sketch['speed'][polyline].split(",")).astype(float)
            best['file_speed'] = points_feature['speed']
            best['file_speed_filter'] =
points_feature['speed_filter']
            best['file_speed_distance'],
best['file_speed_dtw_path'] = \
```

```
                check_dtw_feature( 'Speed', best['name'],
best['file_speed_filter'], best['speed'])
        else:
            best['file_speed_distance'] = 0


        # Pressure
        if(cnfg.computation_pressure == True):
            if (len (sketch['pressure'][polyline]) != 0):
                best['pressure'] =
np.array(sketch['pressure'][polyline].split(",")).astype(float)
            else:
                best['pressure'] = ""

            best['file_pressure'] = points_feature['pressure']
            best['file_pressure_filter'] =
points_feature['pressure_filter']

            # Pressure - Find DTW Features
            if ( len( points_feature['pressure_filter']) != 0):
                best['file_pressure_distance'],
best['file_pressure_dtw_path'] = \
                    check_dtw_feature( 'Pressure', best['name'],
best['file_pressure_filter'], best['pressure'])
            else:
                best['file_pressure_distance'] = 0
                best['file_pressure_dtw_path'] = ""
        else:
            best['file_pressure_distance'] = 0

        print("")
        best_list.append(best.copy())

    best = {}
    if (len(best_list) != 0):
        # Find a best sketch in the best list
        print("# Find a best sketch in the best list")
        best_l = {}
        min_sum_distance = 99999
        for best_l in best_list:
            sum_distance =  best_l['file_angle_distance'] +
best_l['file_speed_distance']*10 +
best_l['file_pressure_distance']*100
            print(f"{best_l['name']} - sum distance:
{sum_distance}")
            if ( sum_distance < min_sum_distance):
```

```
                # Check Default ID and Threshold
                if cnfg.id !=0 and sum_distance >
cnfg.dtw_sum_threshold:
                    continue;

                min_sum_distance = sum_distance
                best = best_l
                best['sum_distance'] = sum_distance


        if (len(best) != 0):
            print(f"\n# Optimum Solution For {polyline}: \nSkecth
Name: {best['name']} - Sum Distance: {best['sum_distance']} -
Angle Distance: {best['file_angle_distance']}")
        else:
            print(f"\n# Optimum Solution For {polyline}: NON")
    else:
        print(f"\n# Optimum Solution For {polyline}: NON")
        best = {}

    end = time.time()
    print(f"Execution Time:",end - start)
    return best


# Simplify Merged Content
def simplify(merged_content):
    merged_content_simple = []
    for x in range(0, len(merged_content), cnfg.simplify_number):
# range(0,10,3) : 0,3,6,9
        merged_content_simple.append(merged_content[x])

    return merged_content_simple


# Select TopN in Library (Angle)
def library_check_dtw(library_data, file_angle_merge):
    import computations as cmpt

    start = time.time()

    # Simplify Merged Content
    file_angle_merge_simple = simplify(file_angle_merge)
    print("# Simplify Content")
    cmpt.print_number_of_points(file_angle_merge_simple)
```

```
    lib_sketch_list = []
    for lib_sketch in library_data:

        # Merge Angles of Library
        lib_sketch_angle_merge = []
        for x in range(len(lib_sketch['angles'])):
            polyline = 'polyline'+str(x+1)

            if len(lib_sketch['angles'][polyline]) != 0:
                lib_sketch_raw =
lib_sketch['angles'][polyline].split(",")
                lib_sketch_angles =
np.array(lib_sketch_raw).astype(float)
                lib_sketch_angle_merge.extend(lib_sketch_angles)


        lib_sketch_angle_merge_simple =
simplify(lib_sketch_angle_merge)

        # Check DTW
        distance, path = check_dtw_feature( 'Angle (Combined)-
Norm', lib_sketch['name'], np.array(file_angle_merge_simple),
np.array(lib_sketch_angle_merge_simple))
#        distance, path =
check_fastdtw(np.array(file_angle_merge),
np.array(lib_sketch_angle_merge), lib_sketch['name'],
gv_output_file_prefix)
        name_dtw_path = (lib_sketch['name'], distance, path)
        lib_sketch_list.append(name_dtw_path)

    lib_sketch_list = sorted(lib_sketch_list, key=lambda distance:
distance[1]) # Sort by distance

    print(f"\n# Select Top {cnfg.TopN} in Library:")
    for x in range(len(lib_sketch_list)):
        print(lib_sketch_list[x][0:2])
        if (x == cnfg.TopN-1):
            print("--------")

    print("\n# The sketches have been removed from library:")
    for lib_sketch in library_data[:]: # [:] Generate a Copy

        lib_check = False
        # Check library sketch Name in TopN List
        for x in range(cnfg.TopN):
```

```
            best_sketch_name = lib_sketch_list[x][0]
            best_sketch_distance = lib_sketch_list[x][1]
            if lib_sketch['name'] == best_sketch_name:
                lib_check = True

                # Check Default ID and Threshold
                if cnfg.id !=0 and best_sketch_distance >
cnfg.dtw_sum_threshold:
                    library_data.remove(lib_sketch)
                    print(f"{lib_sketch['name']} (Threshold)")

                break
            else:
                lib_check = False

        if lib_check == False:
            library_data.remove(lib_sketch)
            print(f"{lib_sketch['name']}")

    end = time.time()
    print(f"Execution Time:",end - start)

    return library_data
```

6.6 figure kütüphanesi

Grafiksel çizimlerin yapıldığı kütüphanedir. Gerekli durumlarda ana prorgram tarafından çağrılmaktadır. Kaynak kodları aşağıda verilmiştir:

```
import matplotlib.pyplot as plt # For Figure
import re
import config as cnfg

# Assignment of Figure Features
def assign_figure_feature(polyline_number,figure_plot,
points_feature, points_x_y):
    plt.figure(1)
    plt.subplot(figure_plot)
    plt.plot(points_feature['angle'],'--')
    plt.plot(points_feature['angle_filter'])

    legend=[cnfg.fig_angle_legend[0], cnfg.fig_angle_legend[1]]
    plt.legend(legend)

    plt_title = str(polyline_number) + ". Polyline (" +
cnfg.fig_angle_ylabel + ")"
    plt.title(plt_title)

    plt.grid(True)
    plt.ylabel(cnfg.fig_angle_ylabel)
    plt.tight_layout()


# Reference: https://pypi.org/project/dtaidistance/
def plot_warping(s1, s2, path, filename=None, xlabel_title=None):
    """Plot the optimal warping between to sequences.

    :param s1: From sequence.
    :param s2: To sequence.
    :param path: Optimal warping path.
    :param filename: Filename path (optional).
    """
    import matplotlib.pyplot as plt
    import matplotlib as mpl
    fig, ax = plt.subplots(nrows=2, ncols=1, sharex=True,
sharey=True, dpi=cnfg.figure_dpi)
    if xlabel_title:
        plt.xlabel(xlabel_title)
    ax[0].plot(s1)
    ax[1].plot(s2)
```

```
    transFigure = fig.transFigure.inverted()
    lines = []
    line_options = {'linewidth': 0.5, 'color': 'orange', 'alpha':
0.8}
    for r_c, c_c in path:
        if r_c < 0 or c_c < 0:
            continue
        coord1 =
transFigure.transform(ax[0].transData.transform([r_c, s1[r_c]]))
        coord2 =
transFigure.transform(ax[1].transData.transform([c_c, s2[c_c]]))
        lines.append(mpl.lines.Line2D((coord1[0], coord2[0]),
(coord1[1], coord2[1]),
                                      transform=fig.transFigure,
**line_options))
    fig.lines = lines
    if filename:
        plt.savefig(filename)
    return fig, ax


# DTW - Figure - Feature (Angle, Speed, Pressure)
def dtw_figure_feature(feature_name, file_feature_dtw_path,
file_feature_distance, file_feature_filter, feature,  name,
polyline):
    if ( len(file_feature_dtw_path) != 0):
        path_figure = list()
        range_number = len(file_feature_dtw_path[0])
        for z in range(range_number):

path_figure.append(tuple((file_feature_dtw_path[0][z],file_feature
_dtw_path[1][z]))) # file_feature_dtw_path[0] = x,
file_feature_dtw_path[1] = y

        filenamew = cnfg.gv_output_directory +
cnfg.gv_output_file_prefix + "_"+ name+"_" + polyline + "_" +
feature_name + "_warp.png"
        xlabel = cnfg.gv_output_file_prefix + " - "+ name+" - " +
polyline + " - Distance: " + str(round(file_feature_distance,2)) +
" ("+feature_name+")"
        try:
            plot_warping(file_feature_filter, feature,
path_figure, filename=filenamew, xlabel_title=xlabel)
        except:
            print(f"\nDTW - Figure - {polyline} ({feature_name}):
Problem of Generate Figure")
```

```
    else:
        print(f"\nDTW - Figure - {polyline} ({feature_name}):
NON")


# DTW - Figure (Angle, Speed, Pressure)
def dtw_figure(best):

    # DTW - Figure (Angle)
    if cnfg.computation_angle == True:
        if( len(best['angle']) != 0):
            dtw_figure_feature('Angle',
best['file_angle_dtw_path'], best['file_angle_distance'],
best['file_angle_filter'], best['angle'],  best['name'],
best['polyline'])

    # DTW - Figure (Speed)
    if cnfg.computation_speed == True:
        if( len(best['speed']) != 0):
            dtw_figure_feature('Speed',
best['file_speed_dtw_path'], best['file_speed_distance'],
best['file_speed_filter'], best['speed'],  best['name'],
best['polyline'])

    # DTW - Figure (Pressure)
    if cnfg.computation_pressure == True:
        if( len(best['pressure']) > 1):

dtw_figure_feature('Pressure',best['file_pressure_dtw_path'],
best['file_pressure_distance'], best['file_pressure_filter'],
best['pressure'],  best['name'], best['polyline'])


# Filter - Figure (Speed, Pressure)
def feature_figure(name, feature, polyline, points_feature,
points_feature_filter, library_feature, filtfilt):
    if ( len(points_feature) != 0):
        plt.figure()
        plt.subplots(num=None, figsize=(8, 4),
dpi=cnfg.figure_dpi, facecolor='w', edgecolor='k')

        # Figure - Left Side (File)
        plt.subplot(121)
        title = cnfg.gv_output_file_prefix + " - "+ polyline
        plt.title(title)
```

```
        plt.plot(points_feature) # Example: best['file_speed']
        plt.plot(points_feature_filter) # Example:
best['file_speed_filter']
        legend=['original ' + feature, 'filtfilt - ' + filtfilt ]
        plt.legend(legend)

        # Figure - Right Side (Library)
        plt.subplot(122)
        plt.title('Veri Seti - Dataset ('+ name +')')
        plt.plot(library_feature) # Example: best['speed']

        plt.tight_layout()
        filenamew = cnfg.gv_output_directory +
cnfg.gv_output_file_prefix + "_" + name + "_" + feature + "_" +
polyline + ".png"

        plt.savefig(filenamew)
    else:
        print(f"Filter - Figure - {polyline} ({feature}): NON")


def best_figure(points_x_y_list, best_sketch, library_data):
    from computations import get_points
    plt.figure()
    plt.subplots(num=None, figsize=(8, 4), dpi=cnfg.figure_dpi,
facecolor='w', edgecolor='k')

    # Figure - Left Side (File)
    plt.subplot(121)
    plt.title('Çizim - Sketch (' + cnfg.gv_output_file_prefix +
')')
    for points_x_y in points_x_y_list:
        plt.plot(points_x_y[:,0].astype(float), -
points_x_y[:,1].astype(float), 'b-')

    # Figure - Right Side (Library)
    plt.subplot(122)
    plt.title('Veri Seti - Dataset (' + best_sketch + ')')
    # Find coordinates for best_sketch
    for lib_sketch in library_data:
        if( lib_sketch['name'] == best_sketch):
            points_x_y_list = []
            for coords in lib_sketch['coords']:
                points_string = lib_sketch['coords'][coords]
```

```
                points_x_y =
get_points(points_string,re.findall("\d+", coords)[0]) # coords:
polyline1 -> 1
                points_x_y_list.append(points_x_y)

    for points_x_y in points_x_y_list:
        plt.plot(points_x_y[:,0].astype(float), -
points_x_y[:,1].astype(float), 'r-')


    plt.tight_layout()
    plt.savefig( cnfg.gv_output_directory +
cnfg.gv_output_file_prefix + '_' + best_sketch + '_result.png')
```

KAYNAKÇA

[1] Gabriel Goehring Johnson, "Sketch-based Interaction for Designing Precise Laser Cut Items," Carnegie Mellon University, 2012.

[2] M. E. Munich and P. Perona, "Camera-based ID verification by signature tracking," in *Computer Vision --- ECCV'98*, 1998, pp. 782–796.

[3] E. J. Peterson, "An Efficient and General-Purpose Technique for Grouping Hand-Drawn Pen Strokes into Objects," University Of California Riverside, 2010.

[4] B. C. Paulson, "Rethinking Pen Input Interaction : Enabling Freehand Sketching Through Improved Primitive Recognition," Texas A&M University, 2010.

[5] T. Hammond and R. Davis, "LADDER, a sketching language for user interface developers," *Comput. Graph.*, vol. 29, no. 4, pp. 518–532, 2005.

[6] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," *Proc. 20th Annu. ACM Symp. User interface Softw. Technol. - UIST '07*, p. 159, 2007.

[7] M. Oltmans, "Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches," Massachusetts Institute Of Technology, 2007.

[8] A. Wolin, B. Paulson, and T. Hammond, "Sort, merge, repeat: an algorithm for effectively finding corners in hand-sketched strokes," *Sketch Based Interfaces Model.*, vol. 1, no. January, pp. 93–99, 2009.

[9] W. Z. Khan, M. Y. Aalsalem, and Y. Xiang, "A Graphical Password Based System for Small Mobile Devices," *Int. J. Comput. Sci.*, vol. 8, no. 5, pp. 145–154, 2011.

[10] "Token Based Authentication." [Online]. Available: https://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/token_based_authentication/. [Accessed: 07-Jan-2018].

[11] J. C. Gyorffy, A. F. Tappenden, and J. Miller, "Token-based graphical password authentication," *Int. J. Inf. Secur.*, vol. 10, no. 6, pp. 321–336, 2011.

[12] S. Bhattacharya and K. Barua, "Fingerprint Identification," *Glob. J. Comput. Sci. Technol.*, vol. 11, no. 6, 2011.

[13] S. Gupta, "Iris Recognition System using Biometric Template Matching Technology," *Int. J. Comput. Appl.*, vol. 1, no. 2, pp. 24–28, 2010.

[14] M. Rouse, "Knowledge-Based Authentication (KBA)," 2015. [Online]. Available: http://searchsecurity.techtarget.com/definition/knowledge-based-authentication. [Accessed: 01-Aug-2018].

[15] M. Martinez-Diaz, J. Fierrez, and J. Galbally, "Graphical Password-Based User

Authentication with Free-Form Doodles," *IEEE Trans. Human-Machine Syst.*, vol. 46, no. 4, pp. 607–614, 2016.

[16] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "The Design and Analysis of Graphical Passwords (DAS)," *Proc. 8th USENIX Secur. Symp.*, vol. 8, p. 1, 1999.

[17] C. Varenhorst and others, "Passdoodles: A lightweight authentication method," *Res. Sci. Inst.*, 2004.

[18] H. Tao, "Pass-Go, a new graphical password scheme," *PhD diss., Univ. Ottawa*, 2006.

[19] R. Weiss and A. De Luca, "PassShapes - Utilizing Stroke Based Authentication to Increase Password Memorability," *Group*, pp. 18–22, 2008.

[20] P. Dunphy and J. Yan, "Do Background Images Improve 'Draw a Secret' Graphical Passwords?," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 36–47.

[21] X. Y. Liu, H. C. Gao, L. M. Wang, and X. L. Chang, "An enhanced drawing reproduction graphical password strategy," *J. Comput. Sci. Technol.*, vol. 26, no. 6, pp. 988–999, 2011.

[22] R. Dhamija and A. Perrig, "Deja vu: A User Study Using Images for Authentication," *Proc. 9th USENIX Secur. Symp. Denver, CO Usenix, 2000.*, no. 102590, pp. 45–58, 2000.

[23] R. Authentication, "Passface: The Science Behind Passfaces," *White Pap. June*, no. September, pp. 1–5, 2004.

[24] D. Davis, F. Monrose, and M. K. Reiter, "Story: On user choice in graphical password schemes," *Proc. 13th Conf. USENIX Secur. Symp. - Vol. 13*, p. 11, 2004.

[25] A. De Angeli, L. Coventry, G. Johnson, and K. Renaud, "VIP: Is a picture really worth a thousand words? Exploring the feasibility of graphical authentication systems," *Int. J. Hum. Comput. Stud.*, vol. 63, no. 1–2, pp. 128–152, 2005.

[26] A. Stubblefield and D. R. Simon, "Inkblot Authentication," *Test*, pp. 1–16, 2004.

[27] S. Wiedenbeck, J. Waters, J. C. Birget, A. Brodskiy, and N. Memon, "PassPoints: Design and longitudinal evaluation of a graphical password system," *Int. J. Hum. Comput. Stud.*, vol. 63, no. 1–2, pp. 102–127, 2005.

[28] S. Chiasson, "Usable Authentication and Click-based Graphical Passwords," Carleton University, 2008.

[29] S. Chiasson, A. Forget, R. Biddle, and P. C. van Oorschot, "Influencing users towards better passwords: Persuasive Cued Click-Points," *Proc. 22nd Br. HCI Gr. Annu. Conf. People Comput. Cult. Creat. Interact.*, pp. 121–130, 2008.

[30] F. Alsulaiman and A. Saddik, “A Novel 3D Graphical Password Schema,” in *2006 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, 2006, pp. 125–128.

[31] T. Eiter and H. Mannila, “Computing discrete Fréchet distance,” TU Vienna, Austria, 1994.

[32] J. Zheng, X. Gao, E. Zhan, and Z. Huang, “Algorithm of on-line handwriting signature verification based on discrete Fréchet distance,” *Adv. Comput. Intell.*, pp. 461–469, 2008.

[33] N. Guo, M. Ma, W. Xiong, L. Chen, and N. Jing, “An Efficient Query Algorithm for Trajectory Similarity Based on Fréchet Distance Threshold,” *ISPRS Int. J. Geo-Information*, vol. 6, no. 11, p. 326, 2017.

[34] M. Müller, *Information Retrieval for Music and Motion - Dynamic Time Warping*. Springer, 2007.

[35] H. Sakoe and S. Chiba, “Dynamic Programming Algorithm Optimization for Spoken Word Recognition,” *IEEE Trans. Acoust.*, vol. 26, no. 1, pp. 43–49, 1978.

[36] S. Keng Chan and Y. H. Tay, “Online Signature Verification using Dynamic Time Warping,” 2006.

[37] R. Martens and L. Claesen, “On-line signature verification by dynamic time-warping,” in *Proceedings - International Conference on Pattern Recognition*, 1996, vol. 3, pp. 38–42.

[38] M. E. Munich and P. Perona, “Visual signature verification using affine arc-length,” in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999, pp. 180–186.

[39] M. E. Munich and P. Perona, “Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification,” *Proc. Seventh IEEE Int. Conf. Comput. Vis.*, vol. 1, pp. 108–115, 1999.

[40] W. E. Snyder, “A strategy for shape recognition,” *Workshop on Challenges and Opportunities in Image Understanding*, College Park, Jan-2007.

[41] K. Krish, S. Heinrich, W. E. Snyder, H. Cakir, and S. Khorram, “Global registration of overlapping images using accumulative image features,” *Pattern Recognit. Lett.*, vol. 31, no. 2, pp. 112–118, Jan. 2010.

[42] B. S. Riggan, “Recognition of Sketch-based Passwords with Biometric Information Using A Generalized Simple K-Space Model,” North Carolina State University, 2014.

[43] S. Butterworth, “On the Theory of Filter Amplifiers,” *Exp. Wirel. Wirel. Eng.*, vol. 7, pp. 536–541, 1930.

[44] T. Srivastava, “Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm(with implementation in Python),” *Analytics Vidhya*, 2014. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/. [Accessed: 24-Aug-2019].

[45] B. S. Riggan, W. E. Snyder, and C. Wang, *Fundamentals of Sketch-Based Passwords*. Cham: Springer International Publishing, 2014.

[46] A. Tolpygo, “Time-Series Analysis: Wearable Devices using DTW and kNN,” *SFL SCIENTIFIC*, 2017. [Online]. Available: https://sflscientific.com/data-science-blog/2016/6/4/time-series-analysis-fitbit-using-dtw-and-knn. [Accessed: 28-Dec-2018].

[47] K. C. Santosh, B. Lamiroy, and L. Wendling, “DTW – Radon-based Shape Descriptor for Pattern Recognition,” *Int. J. Patt. Recogn. Artif. Intell.*, vol. 27, no. 3, 2013.