KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

GEZİNGE (TRAJECTORY) VERİLERİNİN BENZERLİK ANALİZLERİ VE KULLANIM ALANLARI AÇISINDAN DEĞERLENDİRİLMESİ

ADEM ULU

KOCAELİ 2019

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

GEZİNGE (TRAJECTORY) VERİLERİNİN BENZERLİK ANALİZLERİ VE KULLANIM ALANLARI AÇISINDAN DEĞERLENDİRİLMESİ

ADEM ULU

Doç. Dr. Ahmet SAYAR
Danışman, Kocaeli Üniversitesi

Dr. Öğr. Üyesi Alev MUTLU
Jüri Üyesi, Kocaeli Üniversitesi

Dr. Öğr. Üyesi Seçkin ARI
Jüri Üyesi, Sakarya Üniversitesi

Tezin Savunulduğu Tarih: 16.07.2019

ÖNSÖZ VE TEŞEKKÜR

Teknoloji hızla gelişmekte ve hayatımızın çok fazla noktasına temas etmektedir. Bu yüzden teknolojik çalışmalar artmakta ve yüksek önem değerine sahip olmaktadır. Yapılan bu çalışmalar sayesinde insanların zamanını iyi değerlendirmesi ve çok daha farklı avantajlar elde etmesi amaçlanmaktadır. İnsanların hayatında teknoloji içerisinde çok bilinmeyen ama ileride büyük öneme sahip olacağını düşündüğümüz için tez çalışması kapsamında gezinge (trajectory) verileri üzerinde çalışılmıştır.

Tezimin konusunun belirlenmesinde, araştırma aşamasında, yön tayininde ve tamamlanmasında destek olan değerli öğretim üyem ve tez danışmanım sayın Doç. Dr. Ahmet Sayar ve Arş. Gör. Dr. Süleyman Eken'e bana ayırdığı değerli zamanı ve sağladığı destekler için minnettarım.

Haziran-2019 Adem ULU

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iii
TABLOLAR DİZİNİ v
SİMGELER VE KISALTMALAR DİZİNİ vi
ÖZET vii
ABSTRACT viii
GİRİŞ 1
1. GENEL BİLGİLER 5
1.1. Proje Kapsamı 5
1.2. Proje Amaçları 6
1.3. Proje Hedefleri 6
2. MALZEME VE YÖNTEM 8
2.1. Uygulama Algoritma ve Metodları 8
2.1.1. Çözgü tabanlı mesafe hesabı (Warping based distance) 8
2.1.1.1. En uzun ortak alt sıra (LCSS) 8
2.1.1.2. Dinamik zaman çarpması (DTW) 13
2.1.1.3. Gerçek sıralamadaki düzenleme mesafesi (EDR) 18
2.1.2. Şekil tabanlı mesafe hesabı (Shape based distance) 22
2.1.2.1. Simetrik segment yolu mesafesi (SSPD) 22
2.2. Uygulama Tasarım ve Geliştirme 24
2.2.1. NetBeans ide 24
2.2.2. Java programlama dili 25
2.2.3. Jar dosyaları 25
2.2.4. Uygulamada kullanılan veri setleri 25
2.2.4.1. Görüntü'den gezinge (trajectory) verisi oluşturma 26
2.2.5. Harita verileri ile karşılaştırma işlemi 35
2.2.6. Geometrik şekillerin karşılaştırma işlemi 39
2.3. Uygulama Bölümleri 40
2.3.1. Uygulama giriş ekranı 41
2.3.2. Seçimli arayüz ekranı 41
2.3.3. Çizimli arayüz ekranı 51
3. İŞ-ZAMAN PLANI 52
3.1. Proje İş Paketleri ve Tanımlamaları 52
3.2. Proje Yaşam Döngüsü 52
3.3. İş Zaman Planı ve Gannt Şeması 54
4. SONUÇLAR VE ÖNERİLER 58
KAYNAKLAR 67
KİŞİSEL YAYIN VE ESERLER 69
ÖZGEÇMİŞ 70

ŞEKİLLER DİZİNİ

Şekil 2.1. İki nokta arasındaki uzaklığın iki boyutlu uzayda gösterilmesi 12
Şekil 2.2. DTW sonucunda elde edilen tablo görüntüsü 16
Şekil 2.3. S karakterinin ilk görüntüsü 27
Şekil 2.4. Görüntüden kenar tespiti işlemi 27
Şekil 2.5. Görüntü işleme ile elde edilen görüntü 28
Şekil 2.6. S karakterinin ilk görüntüsü 28
Şekil 2.7. Görüntüden kenar tespiti işlemi 29
Şekil 2.8. Görüntü işleme ile elde edilen görüntü 29
Şekil 2.9. Minutiae ayırt edilen özellik noktaları 30
Şekil 2.10. Elde edilen gezinge (trajectory) S veri koordinatlarının son hali 31
Şekil 2.11. Z gezinge (trajectory) verisi 31
Şekil 2.12. S'in minimal gezinge (trajectory) veri koordinatlarının son hali 32
Şekil 2.13. Matplotlib ile çizilmiş bir grafik çizimi 34
Şekil 2.14. İzmit-İstanbul arası en kısa iki rotanın harita görüntüsü 35
Şekil 2.15. İzmir-Ankara arası en kısa iki rotanın harita görüntüsü 36
Şekil 2.16. İzmit-İstanbul arası harita üzerine yolların çizilmesi 36
Şekil 2.17. İzmir-Ankara arası harita üzerine yolların çizilmesi 37
Şekil 2.18. İzmit-İstanbul arası temizlenmiş yol görüntüsü 37
Şekil 2.19. İzmir-Ankara arası temizlenmiş yol görüntüsü 38
Şekil 2.20. İzmit-İstanbul üst kısımdaki yol bilgisinin görüntüsü 38
Şekil 2.21. İzmit-İstanbul alt kısımdaki yol bilgisinin görüntüsü 39
Şekil 2.22. Daire ve elips şekilleri görüntüsü 39
Şekil 2.23. Kare ve dikdörtgen şekilleri görüntüsü 40
Şekil 2.24. Eşkenar üçgen ve ikizkenar üçgen şekilleri görüntüsü 40
Şekil 2.25. Uygulama giriş ekranı 41
Şekil 2.26. Gerçek iki verinin karşılaştırılması sonucu elde edilen görüntü 42
Şekil 2.27. Gerçek ve gürültü eklenmiş verilerin karşılaştırılmasının sonucu 43
Şekil 2.28. S ve Ş harfleri kullanılarak elde edilen verilerin karşılaştırılması 44
Şekil 2.29. Minimal S ve Ş harfleri kullanılarak iki verinin karşılaştırılması 44
Şekil 2.30. S ve Z harflerinden elde edilen sentetik verilerin karşılaştırılması 45
Şekil 2.31. Minimal S ve Z harfleri kullanılarak iki verinin karşılaştırılması 46
Şekil 2.32. S ve s harfleri kullanılarak sentetik iki verinin karşılaştırılması 46
Şekil 2.33. Minimal S ve s harfleri kullanılarak iki verinin karşılaştırılması 47
Şekil 2.34. S ve s harfleri kullanılarak sentetik iki verinin karşılaştırılması 48
Şekil 2.35. Minimal Ş ve s harfleri kullanılarak iki verinin karşılaştırılması 48
Şekil 2.36. İzmit-İstanbul arası iki yolun benzerlik analizi 49
Şekil 2.37. İzmir-Ankara arası iki yolun benzerlik analizi 49
Şekil 2.38. Kare ve dikdötrgen arası benzerlik analizi 50
Şekil 2.39. Elips ve daire arası benzerlik analizi 50
Şekil 2.40. Uygulama çizimli arayüz ekranı görüntüsü 51
Şekil 3.1. Artımlı Geliştirme Modeli görsel anlatımı 53

Şekil 3.2. Gannt şeması çalışma 1. kısım ayrıntılı gösterimi 56
Şekil 3.3. Gannt şeması çalışma 2. kısım ayrıntılı gösterimi 56
Şekil 3.4. Gannt şeması çalışma son kısım ayrıntılı gösterimi 57
Şekil 4.1. LCSS ve EDR benzerlik sonuçları grafiği 61

TABLOLAR DİZİNİ

Tablo 2.1. A ve B kümelerinin LCSS tablo ile mantık çözümü 10
Tablo 2.2. A ve B kümelerinin LCSS tablo ile sayısal çözümü..... 11
Tablo 2.3. data1 ve data2 gezinge verilerinin LCSS tablo ile çözümü..... 13
Tablo 2.4. data1 ve data2 gezinge verilerinin DTW tablo ile çözümü 15
Tablo 2.5. DTW matris değer hesaplama prosedürü..... 16
Tablo 2.6. data1 ve data2 verilerinin DTW tablo ile çözümü..... 17
Tablo 2.6. data1 ve data2 verilerinin DTW tablo ile çözümü (Devam)..... 18
Tablo 2.7. EDR için örnek çözümler 19
Tablo 2.8. EDR hesaplama için gerekli olan parametreler ve açıklamaları..... 19
Tablo 2.9. data1 ve data2 gezinge verilerinin EDR tablo ile çözümü..... 21
Tablo 2.10. Maskeleme matrisi..... 30
Tablo 2.11. CN belirleyici değerleri 30
Tablo 3.1. Proje iş paketleri ve tanımlamaları 52
Tablo 3.2. İş zaman planı ayrıntılı gösterimi 54
Tablo 3.2. İş zaman planı ayrıntılı gösterimi (Devam) 55
Tablo 4.1. Elde edilen sonuçların karşılaştırılması 59
Tablo 4.2. Uygulamanın çalıştırılması için gereken adımların sözde kodu..... 62
Tablo 4.3. Öklit mesafe hesaplama işlemi 62
Tablo 4.4. Hesaplamada kullanılan LCSS algoritmasının sözde kodu 63
Tablo 4.5. Hesaplamada kullanılan DTW algoritmasının sözde kodu..... 63
Tablo 4.6. Hesaplamada kullanılan EDR algoritmasının sözde kodu..... 64
Tablo 4.7. Hesaplamada kullanılan SSPD algoritmasının sözde kodu 64
Tablo 4.8. SSPD hesaplamada kullanılan SPD algoritmasının sözde kodu..... 65
Tablo 4.9. SPD hesaplama için point_to_trajectory metodunun sözde kodu 65
Tablo 4.10. Point_to_segment metodunun sözde kodu 65
Tablo 4.10. Point_to_segment metodunun sözde kodu (Devam) 66

SİMGELER VE KISALTMALAR DİZİNİ

A	:Adenin
O	:Büyük O notasyonu
Ɛ	:Epsilon
G	:Guanin
P	:Maskeleme matrisi
T	:Timin
∑	:Toplam sembolü
C	:Sitozin
∞	:Sonsuz

Kısaltmalar

AMD	:Advanced Micro Devices (Gelişmiş Mikro Cihazlar)
CIMG	:Cool Image (Havalı Görüntü)
CN	:Crossing Number (Geçiş Numarası)
DPL	:Dynamic Path Length (Dinamik Yol Uzunluğu)
DTW	:Dynamic Time Warping (Dinamik Zaman Çarpması)
DPT	:Distance Point to Trajectory (Yörüngeye Uzaklık Noktası)
ED	:Edit Distance (Düzenleme Mesafesi)
EDR	:Edit Distance on Real Sequence (Gerçek Sırayla Mesafeyi Düzenle)
ERP	:Edit Distance with Real Penalty (Gerçek Ceza ile Mesafeyi Düzenle)
GPS	:Global Positioning System (Küresel Yer Belirleme Sistemi)
HTML5	:Hyper Text Markup Language (Köprü Metni Biçimlendirme Dili)
IDE	:Integrated Development Environment (Bütünleşik Geliştirme Ortamı)
LCSS	:Longest Common Subsequence (En Uzun Ortak Alt Küme)
OPENCV	:Open Source Computer Vision (Açık Kaynak Bilgisayar Görme)
OWD	:One-Way Distance (Tek Yönlü Mesafe)
PHP	:Hypertext Preprocessor (Köprü Metni Ön İşlemcisi)
SSPD	:Symmetrized Segment-Path Distance (Simetrik Segment Uzunluğu)
XML	:Extensible Markup Language (Genişletilebilir İşaretleme Dili)

GEZİNGE (TRAJECTORY) VERİLERİNİN BENZERLİK ANALİZLERİ VE KULLANIM ALANLARI AÇISINDAN DEĞERLENDİRİLMESİ

ÖZET

Son yıllarda GPS ve sensörler çok yaygın olarak kullanılmaya başlanmıştır. Bu teknolojilerin günlük hayatımızda yaygın olarak kullanılması ile (özellikle akıllı telefonlar) hareket halindeki nesnelerin takibi ile ilgili çok büyüt boyutlarda veriler toplanmaya başlanmıştır. Bu verilerin analizi zaman ve konumsal olarak bilgiler de taşıdığı için analizlerinde yeni yaklaşımların geliştirilmesine ihtiyaç duyulmuştur. Bu veriler gezinge (Eng. trajectory) olarak adlandırılmaktadır. Bu tez çalışması kapsamında, gezinge (trajectory) verilerinin benzerliklerinin nasıl bulunacağı problemine çözüm yaklaşımları incelenmiştir. Bu analizler gezinge verilerinde aykırılık (Eng. anomaly) tespitleri, benzer hareketli nesnelerin kümelenmesi, bu hareketlere bağlı olarak akıllı trafik sistemleri ve benzeri birçok çalışmalarda kullanılmaktadır. Bu çalışmada genel olarak gezinge (trajectory) verilerinin benzerlikleri için en yaygın kullanılan 4 adet algoritma incelenmiştir. Bu algoritmalar: (1) Longest Common Subsequence – (LCSS), (2) Dynamic Time Warping (DTW), (3) Edit Distance with Real Penalty (EDR) ve (4) Symmetrized Segment-Path Distance (SSPD). Çalışma genelinde sentetik gezinge (trajectory) verileri kullanılarak uygulama düzenlenmeye çalışılmıştır. Benzerlik terimi gezinge verileri için, genelde farkı uygulamalarda farklı anlamlara gelebilmektedir. Bahsi geçen algoritmaların gezinge verilerinin benzerliklerini bulmadaki başarılarını analiz edilmesinden sonra, bu algoritmalardan hangilerinin hangi tür uygulamalarda kullanılabileceğine ya da daha faydalı sonuçlar verebileceğine dair çıkarımlar yapılmaya çalışılmıştır.

Anahtar Kelimeler: DTW, EDR, Gezinge Verilerinin Benzerlikleri, LCSS ve SSPD.

EVALUATION OF TRAJECTORY DATA IN TERMS OF SIMILARITY ANALYSIS AND USAGE AREAS

ABSTRACT

In recent years, GPS and sensors have been widely used. With the widespread use of these technologies in our daily lives (especially smart phones), a large scale of data has been collected for tracking objects in motion. Since the analysis of this data also carries time and spatial information, new approaches were needed to be developed in their analyzes. This data is called as Trajectory (Tr. Gezinge). Within the scope of this thesis, the approaches to the solution of the problem of how to find the similarities of the trajectory data are examined. These analyzes are used in detecting anomaly in navigation data, clustering of similar moving objects, intelligent traffic systems and similar studies related to these movements. In this study, the most commonly used algorithms for the similarity of trajectory data were analyzed. These algorithms are: (1) Longest Common Subsequence - (LCSS), (2) Dynamic Time Warping (DTW), (3) Edit Distance with Real Penalty (EDR) and (4) Symmetrized Segment-Path Distance (SSPD). Throughout the study, it was tried to organize the application by using synthetic trajectory data. The term similarity for navigation data can often mean different applications in different applications. After analyzing the success of these algorithms in finding similarities of navigation data, it was tried to infer which of these algorithms can be used in which applications or give more useful results.

Keywords: DTW, EDR, Similarity of Trajectory Data, LCSS and SSPD**.**

GİRİŞ

Böyle bir çalışma yapılmak istenmesinin birçok nedeni bulunmaktadır. Bunlardan en önemlisi tek bir konum verisinin bile günlük hayatta çok fazla bir öneme sahip olması ve birçok uygulamanın artık konum özelliğini kullanması. Tek bir veri bile bu kadar önemli iken çok daha fazlasının kullanılması ile elde edilecek sonuçları araştırmak o derece önemlidir. Bu yüzden bazı algoritmaları kullanarak sonuçlar elde edilmeye ve elde edilen bu sonuçların analizi yapılmaya çalışıldı.

Gezinge (trajectory) verileri, hareketli bir nesne için sahip olunan konum bilgilerinin zamana göre sıralanması sonucu oluşur. Mobil hesaplamanın ve bilgisayar görme tekniklerinin gelişmesiyle birlikte, hareketli nesnelerin yörüngelerini gerçek hayatta ve videolarda izlemek mümkün hale geldi. Bu verilerin kullanımı birçok alanda ve uygulamada mevcuttur. Örneğin hayvanlara ait göç yollarını inceleyip sonuçlar çıkarmak, kasırganın izlediği yol bilgisi sonucu önlem almak, spor branşlarında elde edilen verilerin analizi sonucu performans araştırmaları yapmak. Bir mağaza gözetim video izleme sisteminde, müşterilerin hareket modellerini bulmak malların düzenlenmesinde yardımcı olabilir. Tüm bu uygulamaların temeli, yörüngeler arasındaki benzerliği belirlemek için sağlam ve doğru yöntemlerdir.

Veri toplama için GPS alıcıları ve kablosuz iletişim içeren mobil cihazlar kullanılabilir. GPS ve kablosuz iletişim gibi teknolojilerin hızlı gelişimi ile gerçek hayattaki haraketli nesneler etkili bir şekilde izlenebilmektedir. Bu cihazlar sayesinde analiz etmek için kullanılabilecek çok büyük miktarda veri elde edilebilir. Hareketli nesnelerin sayısı arttıkça daha fazla gezinge (trajectory) verisi ortaya çıkmaktadır.

GPS karada, havada ve denizde birçok kullanım alanı vardır. Basit bir anlatımla, GPS size bulunduğunuz yerleri işaretleme ve belirlediğiniz noktaya geri dönme imkanı sağlar. GPS, kapalı alanlar ve su altı gibi sinyallerin alınmasının güçleştiği yerler dışında dünya üzerinde her yerde çalışır.

Burada gezinge (trajectory) verileri S, çiftler dizisi olarak tanımlanır, S = [(t1, s1). . ., (tn, sn)], hareketli nesnenin belli bir süre boyunca art arda pozisyonlarını gösteren konum değerleridir. Burada, n, S'deki örnek zaman damgası sayısı, S'nin uzunluğu olarak tanımlanır ve si, zaman damgası ti'de örneklenen bir d aritmetik vektörüdür (d, genellikle 2 veya 3'e eşittir). Bu nedenle, yörüngeler iki (x-y düzlemi) veya üç (x-y-z düzlemi) boyutlu zaman serisi verileri olarak kabul edilebilir.

Veri analizinin tipik amacı, benzer yörüngeleri ve hareketli nesnelerdeki hareket kalıplarını kümelemektir. Kümeleme analizi, veri nesnelerinin gruplandırılmış olması, aynı nesne grubunun yüksek derecede benzerliğe sahip olması, farklı gruplardaki nesnelerin daha düşük derecede benzerlik göstermesidir. Yörüngeler arasındaki mesafenin ölçülmesi büyük araştırma önemine sahiptir.

Gezinge (trajectory) verilerinin mesafe hesabını üç ana tür altında inceleyebiliriz. İlk olarak mesafe hesabı için Ağ Zorunlu Mesafe kullanılmaktadır. Bu mesafe hesabı yol ağının bilindiğini ve verilerin çok iyi bir şekilde dağıldığını varsaydığı için çok tercih edilmemektedir. Geriye kalan diğer iki mesafe hesabı Çözgü Tabanlı Mesafe ve Şekil Tabanlı Mesafe hesabıdır. Bu hesaplama şekilleri ayrıntılı olarak ele alınacaktır.

Gezinge (trajectory) verilerinin işlenmesinde ise birçok algoritma mevcuttur. Bu algoritmaların bir kaçı; SSPD (Symmetric Segment-Path Distance), OWD (One-Way Distance), Hausdorff, Frechet, Discret Frechet, DTW (Dynamic Time Warping), LCSS (Longuest Common Subsequence), ERP (Edit distance with Real Penalty), EDR (Edit Distance on Real sequence) şeklindedir.

Bu algoritmalar benzer temel yaklaşımlara sahiptir. Biz bu algoritmalardan LCSS, DTW, EDR ve SSPD algoritmaları kullanıldı. LCSS orijinal olarak metin benzerliğini hesaplamak için kullanılan bir mesafe algoritmasıydı, daha sonra ise bir boyutlu zaman dizisinin benzerliğini hesaplamak için kullandı. LCSS eşleşme noktası çiftlerini tanımlamak için bir eşik değeri benimsemiştir, böylece gezinge (trajectory) verileri arasındaki benzerliği hesaplayabilir. EDR algoritması, düzenleme mesafesine dayanır, aradaki uzaklık eşik değerinden farklı olan iki nokta arasındaki uyumluluğu belirlemek içindir, EDR, sıradaki gürültüye karşı daha yüksek

toleranslıdır. Gezinge (trajectory) verilerini kullanan birçok çalışma yapılmış ve bununla ilgili yöntemler geliştirilmiştir. Bu çalışmalardan bazıları:

Besse, Guillouet ve ark. [1] gözlemler arasındaki mesafenin seçimine dayalı olarak kümeleme tekniklerini kullanarak, önce yörüngeleri karşılaştırmak için literatürde kullanılan farklı mesafelerin kapsamlı bir incelemesini yapmışlar. Daha sonra, bu yöntemlerin sınırlamalarına dayanarak, yeni bir mesafe hesaplama yöntemi olarak: Simetrik Segment Yolu Mesafesi (SSPD) yöntemini açıklamışlar.

Lin ve Su [2] çalışmasında hareketli nesne yörüngelerinin mekânsal şekillerini karşılaştırmak için bir yol önermişler. "Tek yön mesafeli" (OWD) tabanlı yeni bir mesafe fonksiyonu sunmuşlar. Hem sürekli (lineer) hem de ayrı (grid gösterimi) vakalarda OWD'yi değerlendirmişler ve bunun için algoritmalar geliştirilmişler.

Berndt ve Clifford [3] çalışmasında veri akışlarındaki veya zaman serilerindeki kalıpları tespit etmek için bir yöntem önermişler. Çalışmalarında, problem için dinamik bir programlama yaklaşımı ile bazı ön deneyler açıklanmışlar. Konuşma tanıma alanında kullanılan dinamik zaman atlama tekniğine dayanan Desen algılama algoritmasını tanımlamışlar.

Vlachos, Kollios and Gunopulos [4] çalışmasında, gürültü için çok güçlü olan ve ayrıca dizilerin benzer kısımlarına daha fazla ağırlık vererek, yörüngeler arasında sezgisel bir benzerlik hissi sağlayan En Uzun Ortak Alt Kümeye (LCSS) dayanan metrik olmayan benzerlik fonksiyonlarını resmileştirmişler. Çalışmalarında sekansların zaman içinde gerilmesi, ayrıca dizilerin uzayda global olarak çevrilmesine izin verilen benzerlik ölçümlerini hesaplayan verimli yaklaşık algoritmalar da sağlanmışlar.

Lee, Han ve Whang [5] çalışmasında bir yörüngeyi bir dizi parçaya bölen ve ardından benzer çizgi parçalarını birlikte bir kümeye ayıran yörüngelerin kümelenmesi için yeni bir bölüm ve grup çerçevesi önermişler. Bu çerçevenin temel avantajı olarak, bir yörünge veri tabanındaki ortak alt yörüngeleri keşfetmek olduğunu ileri sürmüşler. Bu bölüm ve grup çerçevesinde, bir yörünge kümesi algoritması TRACLUS geliştirmişler.

Bu çalışmanın diğer çalışmalara göre farkları ve literatüre kazandırdıkları ise şunlardır. Yaygın olarak kullanılan benzerlik algoritmalarının görsel ve daha anlaşılır şekilde test edilmesi. Test edilen bu algoritmaların ne tür uygulamalarda kullanılabileceğinin analizi ve elde edilen sonuçların karşılaştırılması olarak söylenebilir. Sonuç elde edilirken farklı türlerde verilerin kullanılmaya çalışılması, yapılan çalışmanın bir önemli noktasıdır. Burada kullanılan verilerin farklı yöntemler ile elde edilmeye çalışılması da önemli bir noktadır.

Çalışmanın geri kalanında yer alanlar şu şekildedir. Birinci bölümde uygulamanın tanımı ve çalışma amacı yer almaktadır. Bu kısımda yapılan çalışma ile ilgili genel bilgiler yer almaktadır ve neden bu şekilde bir çalışma yapıldığına dair açıklamalara yer verilmiştir. İkinci bölümde ise uygulamada kullanılan yöntem ve metodlar yer almaktadır. Bu kısımda ise uygulamada kullanılan algoritmalara ve ayrıntılarına yer verilmiştir. Ayrıca uygulama tasarımı ve geliştirme aşaması ile ilgili bilgilendirme yapılmıştır. Bu kısımda son olarak uygulama arayüz bölümlerine yer verilmiştir. Üçüncü bölümde çalışmanın iş akışı ve zaman planı yer almaktadır. Bu kısımda yapılan çalışmanın planı ve ne kadar sürede tamamlandığına ilgin ayrıntılar yer almaktadır. Diğer kalan kısımlarda ise çalışma sonucunda elde edilenler ve bunun üzerine yapılan öneriler ve son olarakta çalışmada kullanılan algoritmaların sözde kodları yer almaktadır.

1. GENEL BİLGİLER

Konum verilerinin önemi, kullanım alanlarının çokluğu ve daha ayrıntılı incelenebilir olduğu çalışmaya başlanmıştır. Çalışmamız gezinge (trajectory) verileri kullanılarak benzerlik analizi yapılması ve kullanım alanları açısından değerlendirilmesi hedeflenmek üzere geliştirilmiştir.

Günlük hayatta birçok uygulama konum özelliğine erişim sağlayıp kullanarak çok farklı şekilde uygulamalarında kullanmaktadır. Bu sayede çok daha fazla kişisel veriye ve duruma ulaşılmış olmaktadır. Bu durumların ve verilerin analizleri sonucunda önemli çıkarımlar yapmak mümkün olabilmektedir.

Çalışma genelinde veri elde edilmesinde üç tür yaklaşım kullanılmaktadır. Bu yaklaşımlardan ilki gerçek hayatta elde edilen araç GPS verilerinin bir parçasından oluşan verilerdir. Bir diğer yaklaşım ise bazı benzer karakterlerden elde edile sentetik verilerdir. Son yaklaşım ise anlık olarak kişinin koordinat düzlemi üzerinde yaptığı işaretlemeler sonucu elde edilen verilerdir. Bu farklı türdeki veriler kullanılarak analiz yapılmış ve sonuçlar elde edilmiştir.

1.1. Proje Kapsamı

Çalışma sayesinde birçok gezinge (trajectory) verisinin görselleştirilmesi üzerine işlemler yapılmıştır. Gezinge (trajectory) verileri en küçük birimi baz alındığında iki adet enlem ve boylam çiftinden oluşan konum değerlerinin bütününün oluşturmuş olduğu bir kümedir. Ancak bu sayısal değerler bütünü kişiler için çok bir anlam ifade etmemektedir. Bu yüzden çalışma sayesinde anlamayı kolaylaştırmak için görselleştirmeye de önem verilmiştir. Bu sayede çoğu kişi gerçek anlamda gezinge (trajectory) verisi olarak neden bahsedildiğini daha rahat bir şekilde anlamaktadır.

Sistemden önceki durum;

- Gezinge (trajectory) verisi olarak kastedilen yapı tam olarak anlaşılamamaktaydı ve verilerin elde ediliş şekilleri hakkında bu kadar bilgi bilinmemekteydi.
- Bu verilin nasıl işlenip analiz edileceği hakkında net yargılar mevcut değildi.

- Temel anlamda bahsettiğimiz algoritmaların gerçek kullanım alanları bilinmekte ancak bu algoritmaların gezinge (trajectory) verileri üzerinde nasıl kullanılacağı ve sonuçları bilinmemekteydi.
- Kişiler gezinge (trajectory) verilerinin görselliği ve oluşturulması hakkında bu kadar bilgiye sahip değildi.

Sistemden sonra hedeflenenler;

- Gezinge (trajectory) verisi yapısının ve görselliğinin tam olarak anlaşılabilir hale gelmesi.
- Analizlerin ve kullanım alanlarının net bir şekilde anlaşılabilir hale gelmesi.
- Bahsedilen algoritmaların tam anlamıyla öğrenilmesi ve kullanım amaçlarının kavranabilir hale gelmesi.
- Görselleştirme ve anlık veri elde edilmesi ile gezinge (trajectory) verisi kavramının yabancılıktan çıkarılıp bilinen bir hale gelmesi.

1.2. Proje Amaçları

Bu çalışmada gaye insanların gezinge (trajectory) verilerine olan bilgisini arttırmak ve bunun yanında algoritmalar ile birlikte bu verilerin benzerlik analizini ve kullanım alanları açısından değerlendirmesini yapmaktır. Amacımız, ilk olarak görselleştirme ve anlık veri üretimi sayesinde verinin yapısını kavratmaktır. Daha sonrasında ise bazı algoritmaların temel yapısı ve işleyişi ile ilgili ilgilendirme yapmaktır.

Bu yapılan ön hazırlıklar sonucu farklı türdeki veriler üzerine seçilen algoritmalar uygulanacak ve uygulama sonucunda elde edilen sonuçlar karşılaştırılacaktır. Yani çalışmamız sayesinde hem temel bilgi verilip teori kısmının anlaşılması hem de görsel bilgi ve gündelik hayatta kullanılacak bilgi verilmesi amaçlanmaktadır.

1.3. Proje Hedefleri

Gezinge (trajectory) verileri kullanılarak yapılan çalışmanın ana hedefleri şu şekilde sıralanabilir.

- Gerekli olan tüm işlemlerin tanımlanması.
- Gezinge (trajectory) verilerinin kavram ve yapı olarak anlaşılması.
- Gezinge (trajectory) verilerin görselleştirilmesi

- Anlık elde edilen veriler ve diğer veri türlerinin arasındaki farkların anlaşılması.
- Bazı algoritmaların temel yapısının ve çalışma mantığının kavranması.
- Algoritmaların gezinge (trajectory) verileri üzerinde kullanılması.
- Kullanılan bu algoritmaların sonuçlarının karşılaştırılması.
- Gezinge (trajectory) verilerinin benzerlik analizi yapılması.
- Gezinge (trajectory) verilerinin uygulama alanlarının değerlendirilmesi.

Şeklinde bunlar ve birçok benzeri hedef gösterilebilir. Genel olarak ana hedef gezinge (trajectory) verisi kavramının tam olarak anlatılması, benzerlik analizinin yapılması ve uygulama alanlarının değerlendirilmesidir.

2. MALZEME VE YÖNTEM

2.1. Uygulama Algoritma ve Metodları

Gezinge (trajectory) verilerinin benzerlik analizleri yapılırken iki tip mesafe tespit yöntemi kullanılmaktadır. Bunlardan ilki sadece yörünge şekli ile ilgilendiği için Şekil Tabanlı Mesafe Hesabı olarak ikincisi ise aynı zamanda zamansal boyutu da hesaba kattığı için Çözgü Tabanlı Mesafe Hesabı olarak adlandırılır.

2.1.1. Çözgü tabanlı mesafe hesabı (Warping based distance)

Elde edilen verilerdeki konumlar yörünge hızlarına göre çok farklı olabilir, bu yüzden veriler arasında karşılaştırma yaparken bu bilgileri de hesaba katmak gerekir. Bu şekilde zaman serilerinin analizinde Çözgü Tabanlı Mesafe Hesabı kullanılır. Bu hesaplama yönteminde amaç farklı yörüngelere ait konumların farklı indekslerle eşleştirilmesini sağlamaktır.

Bu şekilde hesap yapan yöntemlere, Dinamik Zaman Çarpması (DTW), En Uzun Ortak Alt Sıra (LCSS) ve Gerçek Sıradaki Mesafeyi Düzenle (EDR) örnek olarak verilebilir. Bu mesafeler aynı şekilde tanımlanır, ancak farklı maliyet fonksiyonları kullanırlar.

2.1.1.1. En uzun ortak alt sıra (LCSS)

Düzenleme mesafesinin bir varyasyonudur. Temel fikir, iki diziyi, elemanların sırasını yeniden düzenlemeden, ancak bazı elemanların eşsiz olmasına izin vermeden, esnemelerine izin vererek uydurmaktır. LCSS yönteminin avantajları iki yönlüdür:

- Euclidean ve DTW'de tüm elemanların eşleşmesi gereken, hatta aykırı olan bazı elemanlar eşsiz olabilir.
- LCSS modeli, daha sonra gösterileceği gibi daha verimli bir hesaplamaya olanak sağlar.

LCSS temel mantıkta iki küme arasındaki sıralı ortak elemanların ortak olan en uzun parçasının elde edilmesi. A kümesinin elemanları {X,M,J,Y,A,U} olsun. Diğer bir B kümesinin elemanları ise {M,Z,J,A,W,X,U} olsun. Bu iki küme LCSS algoritmasına girdi olarak verilirse, burada elde edilecek olan sıralı, ortak olan en uzun alt küme {M,J,A,U} olarak bulunur [6].

LCSS çalışma mantığı olarak ayrıntılı bir şekilde incelendiğinde dinamik programlamanın anlaşılması için ideal bir örnektir [6].

Yukarıda bahsettiğimiz örnek kümelerde LCSS uygulanmasını daha ayrıntılı olarak ele almak gerekirse. Problemin ilk akla gelen ve en basit çözümü kümelerden birisini ana küme seçerek, bu kümedeki elemanları teker teker diğer küme ile sınamak olabilir [6].

Örneğin A kümesi seçilmiş olsun. A kümesinin ilk elemanından son elemanına kadar ortak elemanlara bakılacaktır. Bu durumda ilk eleman X, B kümesinde aranacaktır, daha sonra X ile birlikte U aranacaktır. İhtimallerin sonuna gelindiğinde bu ikili bırakılıp yeni bir ihtimal denenecektir:

Örnek çalışma:

A kümesinden X seçilir. B kümesinde sınanır. Karşılığı bulununca LCSS1 olarak kaydedilir. Devamı olan harflerden uygun olan seçilir (bu örnekte U) LCSS1'e eklenir [6].

- LCSS1 -> {X,U}
- LCSS2 -> {M,J,A,U}
- LCSS3 -> {J,A,U}
- LCSS4 -> Y'nin eşi diğer kümede yok
- LCSS5 -> {A,U}
- LCSS6 -> {U}

Yukarıdaki yönteme göre A, kümesinin bütün elemanları başlangıç elemanı olarak denenmiştir. Fakat dinamik programlama yaklaşımı incelenecek olursa bu işlemin çok daha kısa zamanda yapılabileceği görülür:

Aşağıda bir çözüm yöntemi olarak tablo kullanılması önerilmiştir. Buna göre şekilde her hücre, ilgili satır ve sütün başlığına kadar olan en uzun ortak kümeyi tutmaktadır.

Tablo 2.1. A ve B kümelerinin LCSS tablo ile mantık çözümü

	0	***1,X***	***2,M***	***3,J***	***4,Y***	***5,A***	***6,U***
0	<>	<>	<>	<>	<>	<>	<>
1,M	<>	<>	<M>	<M>	<M>	<M>	<M>
2,Z	<>	<>	<M>	<M>	<M>	<M>	<M>
3,J	<>	<>	<M>	<M,J>	<M,J>	<M,J>	<M,J>
4,A	<>	<>	<M>	<M,J>	<M,J>	<M,J,A>	<M,J,A>
5,W	<>	<>	<M>	<M,J>	<M,J>	<M,J,A>	<M,J,A>
6,X	<>	<X>	<M>,<X>	<M,J>	<M,J>	<M,J,A>	<M,J,A>
7,U	<>	<X>	<M>,<X>	<M,J>	<M,J>	<M,J,A>	<M,J,A,U>

Buna göre verilen problemdeki en uzun ortak kümeyi bulmak hedeflenmektedir. Tablonun sağ alt köşesinde bulunan bu küme sorunun cevabıdır. Sorunun çözümünde tablo oluşturulurken her hücre kendisinin bir üstündeki veya bir solundaki hücreyi aynen kopyalamakta ve üzerine satır ve sütün ismini şayet ortaksa yazmaktadır.

Örneğin son oluşturulan sağ alt köşedeki küme, solundaki ve tepesindeki kümelerin kopyalanması ile oluşur. Bu kopyalama işlemine U elemanı ilave edilir çünkü bu hücrenin bulunduğu sütun ile satır ismi aynıdır [6].

Bu işlem yapılırken karşılaşılabilecek bir problem de solundaki bilgi ile tepesindeki bilginin aynı olmamasıdır. Bu durum kırmızı çember içerisinde gösterilmiştir. Bu durumda daha uzun bir küme elde edilince kısa kalan küme bırakılarak, uzun küme ile yola devam edilir.

Bu işlem sonunda aşağıdaki ortak küme uzunluklarını gösteren Tablo 2.2. elde edilir.

Tablo 2.2. A ve B kümelerinin LCSS tablo ile sayısal çözümü

	0	***1,M***	***2,Z***	***3,J***	***4,A***	***5,W***	***6,X***	***7,U***
0	0	0	0	0	0	0	0	0
1,X	0	0	0	0	0	0	1	1
2,M	0	1	1	1	1	1	1	1
3,J	0	1	1	2	2	2	2	2
4,Y	0	1	1	2	2	2	2	2
5,A	0	1	1	2	3	3	3	3
6,U	0	1	1	2	3	3	3	4

Buna göre her seferinde kendinden önceki elemanlara bakılması önlenmiş olur. Yani örneğin tablonun 4,5 hücresine bakıldığında A harfleri kontrol edilmektedir. Bu kontrol sırasında kendisinden önceki harflerin kontrolüne gerek kalmaz. (tekrar B kümesinin MZJ harflerine bakılmaz), bu sayede ortak yapılan işlemler, dinamik programlama ilkesine uygun olarak elenmiş olur [6].

LCSS algoritmasının gezinge (trajectory) verileri üzerindeki çalışma mantığını ele alacak olursak. Elimizde iki adet gezinge (trajectory) verisi bulunmaktadır. Bu örnek veriler daha kolay anlaşılabilir olması için sentetik verilerdir ve değerler tam sayı olarak seçilmiştir.

- data1 = {(1,2),(2,4),(1,5),(2,5),(3,4)}
- data2 = {(1,2),(2,4),(2,6),(2,5),(3,4)}

Gezinge (trajectory) verileri yukarıdaki şekildedir ve hesaplama sonuçlarının karşılaştırılmasında ihtiyaç olan epsilon (Ɛ) değeri ise 0,00001 olarak alınmıştır. Karşılaştırma sırasında noktalar arası benzerlik hesaplaması yapıldığı için öklid mesafe hesaplamasına ihtiyaç duyulmaktadır.

Öklit Mesafesi**,** matematikte pisagor bağlantısı kullanılarak bulunan iki nokta arasındaki mesafe ölçüm birimidir. Buna göre iki boyutlu düzlemde iki nokta arasındaki mesafe basitçe iki noktanın x ve y koordinatlarının ayrı ayrı farklarının hipotenüs'üne eşittir [7].

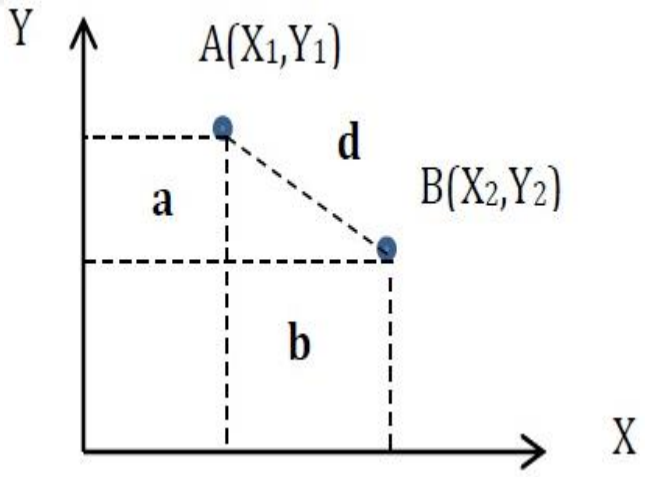
Örneğin birinci nokta $A(X_1,Y_1)$ olsun (A noktasının koordinatları X_1,Y_1) ve ikinci nokta B(X2,Y2) (yani B noktasının koordinatları X2,Y2 olsun) bu durumda iki nokta arasındaki mesafe Denklem (2.1) ye göre hesaplanmaktadır.

$$d(A,B)=\sqrt{(X_1-X_2)^2+(Y_1-Y_2)^2} \tag{2.1}$$

Şeklindedir. Bu bağıntı genelleştirilecek olursa, Denklem (2.2) gibi bir bağıntıya ulaşılır.

$$d(i,j)=\sqrt{\sum_{k=1}^{p}(X_{ik}-X_{jk})^2} \tag{2.2}$$

İki nokta arasındaki uzaklığın iki boyutlu uzayda gösterimi Şekil 2.1.'dedir.



Şekil 2.1. İki nokta arasındaki uzaklığın iki boyutlu uzayda gösterilmesi [7]

Şayet uzay 3 boyutlu olursa bu defa öklit mesafesi noktaların 3 boyuttaki koordinat değerlerinin farklarının karelerinin toplamlarının karekökü olarak hesaplanır [7].

Burada daha önceden bahsedilen epsilon eşik değerine göre noktalar arasındaki uzaklık bu eşik değerinden küçük ise bu iki nokta benzerdir veya aynıdır, şeklinde

yorumlanır. Daha sonra önceden bahsedilen dinamik tablo mantığı uygulanarak C matrisi elde edilir. Bu matris ise Tablo 2.3.'de gösterildiği şekildedir.

Tablo 2.3. data1 ve data2 gezinge verilerinin LCSS tablo ile çözümü

C	***0***	***1***	***2***	***3***	***4***	***5***
0	0	0	0	0	0	0
1	0	1	1	1	1	1
2	0	1	2	2	2	2
3	0	1	2	2	2	2
4	0	1	2	2	3	3
5	0	1	2	2	3	**4**

Elde edilen bu C tablosu kullanılarak benzerlik analiz oranı elde edilir. Bu matris ve gezinge (trajectory) verilerinin boyutlarının en küçüğü kullanılarak hesaplama yapılır. Hesaplama için kullanılan formül ise Denklem (2.3) de verilmiştir.

$$LCSS=1-float(\frac{C[n_0][n_1]}{\min(n_0,n_1)}) \quad (2.3)$$

Yukarıdaki formül kullanılarak hesaplama yapılır. Bu formüldeki n_0 ve n_1 değerleri data1 ve data2 değerlerinin eleman sayılarına karşılık gelmektedir. Buna göre C[5][5] = 4 olarak bulunur ve min(5,5) = 5 olarak bulunur. Bu değerlere göre bu örnekteki data1 ve data2 benzerlik analizi yapıldığında sonuç olarak 0,2 değeri elde edilmektedir.

2.1.1.2. Dinamik zaman çarpması (DTW)

Zaman serilerini ve belirli bir kelime şablonunu hizalamak için bir dinamik programlama yaklaşımı kullanır, böylece bir mesafe ölçümü minimize edilir. Makul bir uyum sağlamak için zaman ekseni gerildiği (veya sıkıştırıldığı) için, bir şablon çok çeşitli gerçek zaman serileri ile eşleşebilir. Bir mesafe ölçüsü tanımlandıktan

sonra, dinamik zaman atlama problemini, her yolun kümülatif mesafesine dayanan potansiyel çarpma yolları üzerinde bir minimizasyonu tanımlayabiliriz.

DTW, hızda değişebilen iki geçici sekans arasındaki benzerliği ölçmek için bir algoritmadır. Örneğin, bir kişi diğerinden daha hızlı yürüdüğü halde veya bir gözlem süresince hızlanma ve yavaşlama olsa bile DTW ile yürüyüş benzerlikleri tespit edilebilir. Kişi önceden kaydedilmiş örnek sesinden daha hızlı veya daha yavaş konuşsa bile, bir örnek ses komutunu başkaları komutuyla eşleştirmek için kullanılabilir. DTW, geçici video, ses ve grafik verilerinin dizilerine uygulanabilir, hatta doğrusal bir diziye dönüştürülebilen veriler DTW ile analiz edilebilir [8].

Genel olarak, DTW, verilen kısıtlamalar ile verilen iki sıra arasındaki en uygun eşleşmeyi hesaplayan bir yöntemdir. Ama burada daha basit noktalara sadık kalalım. Diyelim ki iki tane ses dizimiz var Örnek ve Test ve bu iki dizinin uyuşup uyuşmadığını kontrol etmek istiyoruz. Burada ses dizisi, sesinizin dönüştürülmüş dijital sinyalini ifade eder. Söyleyeceğiniz kelimeleri belirten, sesinizin genliği veya sıklığı olabilir [8]. Diyelim ki:

- Örnek = {1, 2, 3, 5, 5, 5, 6}
- Test = {1, 1, 2, 2, 3, 5}

Bu iki dizi arasındaki en uygun eşleşmeyi bulmak istiyoruz. İlk önce, iki nokta arasındaki mesafeyi tanımladık, d (x, y), burada x ve y iki noktayı temsil eder. Denklem (2.4) deki şekildedir.

$$d(x,y)=| x-y| \tag{2.4}$$

Bu iki diziyi kullanarak bir 2D matris tablosu oluşturalım. Her Test noktasıyla her Örnek noktası arasındaki uzaklığı hesaplar ve aralarındaki en uygun eşleşmeyi buluruz.

Burada, Tablo [i] [j], daha önce gözlemlediğimiz tüm optimal mesafeleri göz önünde bulundurarak Örnek [i] ve Test [j] 'e kadar olan sırayı dikkate alırsak, iki dizi arasındaki en uygun mesafeyi temsil eder.

İlk satırda, Örnek'ten değer almazsak, bununla Test arasındaki mesafe sonsuz (∞, inf) olacaktır. Bu yüzden ilk sıraya sonsuzluğu koyduk. Aynı ilk sütun için de geçerli. Test'ten herhangi bir değer almazsak, bununla Örnek arasındaki mesafe de sonsuz olacaktır. Ve 0 ile 0 arasındaki mesafe sadece 0 olacaktır.

Şimdi, her adım için, söz konusu her nokta arasındaki mesafeyi göz önünde bulunduracağız ve şimdiye kadar bulduğumuz minimum mesafeye ekleyeceğiz. Bu bize, bu diziye kadar iki dizinin optimal mesafesini verecektir. Formülümüz ise şu şekilde olacaktır.

Tablo[i][j]=d(i,j)+min(Tablo[i-1][j], Tablo[i-1][j-1], Tablo[i][j-1]) (2.5)

İlki için d (1, 1) = 0, Tablo[0] [0] minimum değeri temsil eder. Bu yüzden Tablo[1][1] değeri 0 + 0 = 0 olacaktır. İkincisi için d (1, 2) = 0 olur. Tablo[1][1] minimum değeri temsil eder. Değer şöyle olacaktır: Tablo[1][2] = 0 + 0 = 0. Bu şekilde devam edersek, bitirdikten sonra sonuç Tablo 2.4.'deki gibi olacaktır.

Tablo 2.4. data1 ve data2 gezinge verilerinin DTW tablo ile çözümü

	0	**1**	**1**	**2**	**2**	**3**	**5**
0	0	inf	inf	inf	inf	inf	inf
1	inf	0	0	1	2	4	8
2	inf	1	1	0	0	1	4
3	inf	3	3	1	1	0	2
5	inf	7	7	4	4	2	0
5	inf	11	11	7	7	4	0
5	inf	15	15	10	10	6	0
6	inf	20	20	14	14	9	**1**

Tablo[7][6] 'daki değer verilen bu iki sekans arasındaki maksimum mesafeyi temsil eder. Burada 1, Örnek ve Test arasındaki maksimum mesafeyi 1'dir.

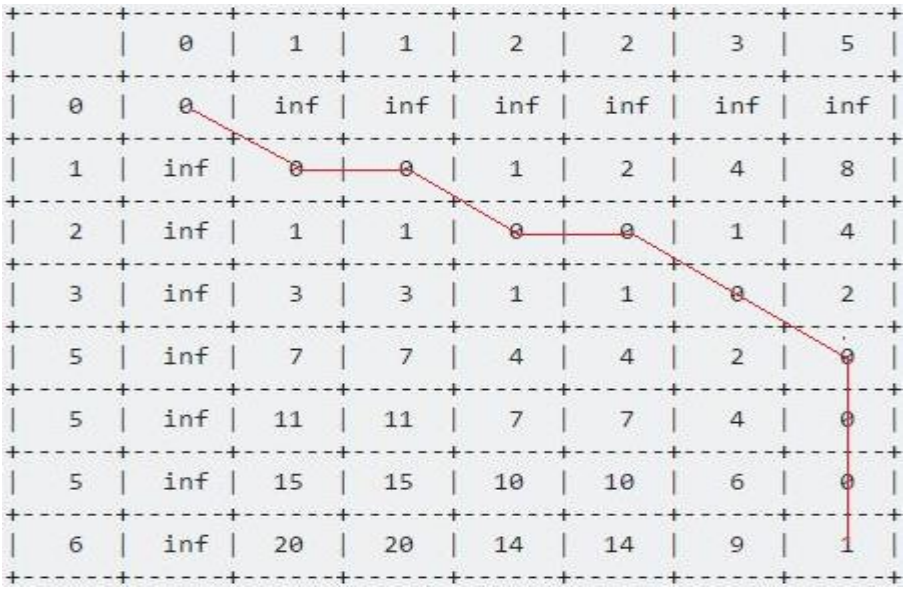
Şimdi son noktadan başlayarak (0, 0) başlangıç noktasına kadar geriye doğru gidersek yatay, dikey ve çapraz olarak hareket eden uzun bir çizgi alırız. Bu işlem için kullanılan prosedür şu şekildedir:

Tablo 2.5. DTW matris değer hesaplama prosedürü

```
if  Tablo[i-1][j-1] <= Tablo[i-1][j] and Tablo[i-1][j-1] <= Tablo[i][j-1]
    i := i - 1
    j := j - 1
else if  Tablo[i-1][j] <= Tablo[i-1][j-1] and Tablo[i-1][j] <= Tablo[i][j-1]
    i := i - 1
else
    j := j - 1
end if
```

Buna ulaşana kadar devam edeceğiz (0, 0). Her hareketin kendi anlamı vardır. Yatay bir hareket silinmeyi temsil eder. Bu, Test sırasının bu aralıkta hızlandırıldığı anlamına gelir.Dikey bir hareket, yerleştirmeyi temsil eder. Bu, Test sırasının bu aralık boyunca yavaşladığı anlamına gelir.

Çapraz hareket eşleşmeyi temsil eder. Bu süre zarfında Test ve Örnek aynıydı. Bu prosedürler sonucunda elde edilen tablo ve çizilmiş hali aşağıdaki şekilde olmaktadır.



		0	1	1	2	2	3	5
0	0	inf	inf	inf	inf	inf	inf	
1	inf	0	0	1	2	4	8	
2	inf	1	1	0	0	1	4	
3	inf	3	3	1	1	0	2	
5	inf	7	7	4	4	2	0	
5	inf	11	11	7	7	4	0	
5	inf	15	15	10	10	6	0	
6	inf	20	20	14	14	9	1	

Şekil 2.2. DTW sonucunda elde edilen tablo görüntüsü [8]

Hesaplama DTW'sinin karmaşıklığı O (m * n) 'dir ve burada m ve n her dizinin uzunluğunu temsil eder. DTW'nin hesaplanması için daha hızlı teknikler arasında PrunedDTW, SparseDTW ve FastDTW sayılabilir [8].

DTW algoritmasının gezinge (trajectory) verileri üzerindeki çalışma mantığını ele alacak olursak. Elimizde iki adet gezinge (trajectory) verisi bulunmaktadır. Bu örnek veriler daha kolay anlaşılabilir olması için sentetik verilerdir ve değerler tam sayı olarak seçilmiştir.

- data1 = {(1,2),(2,4),(1,5),(2,5),(3,4)}
- data2 = {(1,2),(2,4),(2,6),(2,5),(3,4)}

Karşılaştırma sırasında noktalar arası benzerlik hesaplaması yapıldığı için yine öklit mesafe hesaplamasına ihtiyaç duyulmaktadır. Bu hesaplama türünden yukarıda ayrıntılı olarak bahsetmiştik. Öklit mesafe hesaplamasında kullanılan formül şu şekildeydi:

$$d(A,B)=\sqrt{(X_1-X_2)^2+(Y_1-Y_2)^2} \quad (2.6)$$

Dinamik tablo mantığı göre C matrisi elde edilir. Bu matris değerleri elde edilirken kullanılan DTW formülü ise şu şekildedir.

$$C[i][j]=oklit(data[i], data2[j])+min(C[i-1][j], C[i-1][j-1], C[i][j-1]) \quad (2.7)$$

Bu formül ve gezinge (trajectory) verileri kullanılarak aşağıdaki C matris tablosu elde edilir. Burada son değer olan C[5][5] bizim gezinge (trajectory) verilerinin benzerlik analiz sonucunu verir. Bu sonucun hesaplanması ise şu şekildedir.

$$C[5][5]=0+\min(2.8, 1.4, 2.8)=1.4 \quad (2.8)$$

Tablo 2.6. data1 ve data2 verilerinin DTW tablo ile çözümü

C	***0***	***1***	***2***	***3***	***4***	***5***
0	0	inf	inf	inf	inf	inf

Tablo 2.6. data1 ve data2 verilerinin DTW tablo ile çözümü (Devam)

1	inf	0	2,2	6,3	9,5	12,3
2	inf	2,2	0	2	3	4
3	inf	5,2	2,4	1,4	2,4	4,6
4	inf	8,3	2,4	2,4	1,4	2,8
5	inf	11,2	3,4	4,6	2,8	**1,4**

2.1.1.3. Gerçek sıralamadaki düzenleme mesafesi (EDR)

İki yörünge arasındaki benzerliğin ölçülmesinde mevcut olanlardan daha sağlam ve doğrudur. EDR'de eşleşme eşiği, bir çift eleman arasındaki mesafeyi 0 ve 1'e iki değerle niceleyerek gürültünün etkilerini azaltır. Yerel zaman kaydırmasını işleme yeteneğini sunar.

EDR, iki dizi arasındaki benzerliği ölçmek için yaygın olarak kullanılan Düzenleme Mesafesi (ED) [9] 'ye dayanmaktadır. Düzenleme Mesafesi sayesinde biyo-bilişim ve konuşma tanıma üzerine önemli çalışmalar yapılmıştır.

Düzenleme Mesafesi(ED), hesaplamalı dilbilim ve bilgisayar bilimlerinde düzenleme mesafesi, bir dizgeyi diğerine dönüştürmek için gereken minimum işlem sayısını sayarak, birbirine benzemeyen iki dizginin (örneğin sözcükler) birbirine ne kadar olduğunu ölçmenin bir yoludur.

Mesafeleri düzenleme, doğal yazım işlemlerinde, otomatik yazım düzeltmesinin yanlış yazılan bir sözcük için aday düzeltmelerini söz konusu kelimeye uzak mesafeli bir sözlükten kelimeler seçerek belirleyebildiği uygulamalar bulur. Biyoinformatikte, A, C, G ve T harflerinin dizileri olarak görülebilen DNA dizilerinin benzerliğini ölçmek için kullanılabilir [10].

Bir düzenleme mesafesinin farklı tanımları farklı dizi işlemlerini kullanır. Levenshtein mesafe işlemleri dizedeki bir karakterin çıkarılması, eklenmesi veya

ikame edilmesidir. En yaygın metrik olan Levenshtein mesafesi genellikle "mesafeyi düzenle" ile kastedilen şeydir [10]. Örnekler Tablo 2.7.'de yer almaktadır.

Tablo 2.7. EDR için örnek çözümler

Girdi: str1 = "geek", str2 = "gesek" Çıktı: 1 Bir 's' ekleyerek str1'i str2'ye dönüştürebiliriz
Girdi: str1 = "cat", str2 = "cut" Çıktı: 1 'a' yerine 'u' yazarak str1'i str2'ye çevirebiliriz.
Girdi: str1 = "sunday", str2 = "saturday" Çıktı: 3 Son üç ve ilk karakterler aynı. 'n' yi 'r' ile değiştir, t ekle, a ekle

EDR için iki tanım mevcuttur. Öncelikle bir çift gezinge (trajectory) verisi birbiri ile eşleştirilir. Sırasıyla n ve m uzunluğunda R ve S iki gezinge (trajectory) verildiğinde, R ve S arasındaki Gerçek Sıralamadaki Düzenleme Mesafesi (EDR), R'yi S'ye değiştirmek için gereken işlemleri ekleme, silme veya değiştirme sayısıdır. EDR hesaplamasında 4 parametre mevcuttur. EDR(x, y, epsilon, sigma), bu parametrelerin detayları Tablo 2.8.'de gösterildiği şekildedir.

Tablo 2.8. EDR hesaplama için gerekli olan parametreler ve açıklamaları

x	İlk zaman serilerini içeren sayısal vektör.
y	İkinci zaman serisini içeren sayısal vektör.
epsilon	Mesafeyi tanımlayan pozitif bir eşik değeri.
sigma	Arzu edilirse, pencere boyutunu temsil eden pozitif bir tamsayı belirtilerek bir Sakoe-Chiba pencereleme kısıtlaması eklenebilir.

Diğer hesaplama yöntemleri ile karşılaştırıldığında EDR sahip oldukları:

- EDR'de eşleşme eşiği, bir çift eleman arasındaki mesafeyi 0 ve 1 olan iki değere ölçerek gürültünün etkilerini azaltır (LCSS de aynı nicelemeyi gerçekleştirir) [10].

- Bir gezinge (trajectory) değerini diğerine değiştirmek için gereken minimum düzenleme işleminin aranması, EDR'ye yerel zaman kaymasını idare etme yeteneği sunar [10].
- LCSS'nin aksine, EDR, boşlukların uzunluklarına göre eşleşen iki alt yörünge arasındaki boşluklara ceza atar ve bu da LCSS'den daha doğru olmasını sağlar [10].

EDR algoritmasının gezinge (trajectory) verileri üzerindeki çalışma mantığını ele alacak olursak. Elimizde iki adet gezinge (trajectory) verisi bulunmaktadır. Bu örnek veriler daha kolay anlaşılabilir olması için sentetik verilerdir ve değerler tam sayı olarak seçilmiştir.

- data1 = {(1,2),(2,4),(1,5),(2,5),(3,4)}
- data2 = {(1,2),(2,4),(2,6),(2,5),(3,4)}

Dinamik tablo mantığı göre C matrisi elde edilir. Bu matris değerleri elde edilirken kullanılan EDR formülü ise şu şekildedir.

$$C[i][j]=\min(C[i][j-1]+1, C[i-1][j]+1, C[i-1][j-1]+\text{maliyet}) \quad (2.9)$$

Formülde görüleceği gibi bir maliyet değerinin hesaplanması gereklidir. Bu maliyet değeri 0 veya 1 olmaktadır. Bu değerin kaç olacağı epsilon değeri ve öklit mesafesinin karşılaştırılması sonucu belirlenir. Eğer öklit mesafesi epsilon değerinden küçük ise maliyet 0 aksi durumda ise maliyet 1'dir. Bu işlem sırasında öklit mesafe hesaplamasına ihtiyaç duyulmaktadır. Bu hesaplama türünden yukarıda ayrıntılı olarak bahsetmiştik. Öklit mesafe hesaplamasında kullanılan formül şu şekildeydi:

$$d(A,B)=\sqrt{(X_1-X_2)^2+(Y_1-Y_2)^2} \quad (2.10)$$

Bu formül ve gezinge (trajectory) verileri kullanılarak aşağıdaki C matris tablosu elde edilir. Burada son değer olan C[5][5] bizim gezinge (trajectory) verilerinin benzerlik analiz sonucunu verir. Karşılaştırma için kullanılan epsilon değeri ise 0,00001 olarak alınmıştır. Bu sonucun hesaplanması ise şu şekildedir.

$$C[5][5] = \min(3, 3, 1) = 1 \quad (2.11)$$

Burada formüle bakıldığında bir noktanın değer hesaplaması şu şekildedir:

$$C[i][j] = \min(sol+1, üst+1, çapraz+maliyet) \quad (2.12)$$

Hesaplamalar sonucunda C matrisi elde edilir. Kullanılan data1 ve data2 için oluşturulan C dinamik programlama matrisi ise şu şekildedir. Bu matris kullanılarak EDR değeri hesaplanacaktır.

Tablo 2.9. data1 ve data2 gezinge verilerinin EDR tablo ile çözümü

C	***0***	***1***	***2***	***3***	***4***	***5***
0	0	0	0	0	0	0
1	0	0	1	1	1	1
2	0	1	0	1	2	2
3	0	1	2	2	2	3
4	0	1	2	2	1	2
5	0	1	2	3	2	**1**

Bu tablo kullanılarak hesaplanacak EDR değerinin formülü ise şöyledir:

$$EDR = \text{float}\left(\frac{C[n_0][n_1]}{\max(n_0, n_1)}\right) \quad (2.13)$$

Formül içerisindeki n_0 ve n_1 değerleri kullanılan data1 ve data2 gezinge (trajectory) verilerinin eleman sayılarına denk gelmektedir. Bu C matrisi ve formül kullanılarak elde edilen data1 ve data2 gezinge (trajectory) verilerinin benzerlik analizi sonucu ise şu şekildedir.

$$EDR = \text{float}\left(\frac{1}{\max(5,5)}\right) = 0.2 \quad (2.14)$$

2.1.2. Şekil tabanlı mesafe hesabı (Shape based distance)

Yörüngelerin geometrik özelliklerini ve şekillerini yakalamaya çalışan mesafe hesaplama algoritmaları bu tür içerisinde yer almaktadır. En çok bilinenler Hausdorff mesafesi ve Frechet mesafesidir. Biz ise çalışmamız da bu türdeki diğer mesafe hesaplama yöntemi olan Simetrik Segment Yolu Mesafesi (SSPD) algoritmasını kullanmaktayız. Diğer hesaplama yöntemlerinden kısaca bahsedecek olursak.

Hausdorff, mesafe hesaplama türü metrik bir yöntemdir. İki metrik uzay arasındaki mesafeyi ölçmemizi sağlar. İki veri kümesinde her nokta için hesaplanan en büyük mesafelerin supremumu sonucu elde edilen değer Hausdorff mesafesi olarak adlandırılır. Bu yöntemin hesaplamaları çok zor ve karmaşık olabilmektedir. Ancak yapılacak bazı basitleştirme işlemleri sonucu kullanılabilir duruma gelmektedir [11].

Frechet ve Ayrık Frechet, Frechet mesafesi, eğriler arasındaki benzerliği ölçer. Genellikle "yürüyüş-köpek mesafesi" olarak bilinir. Bir köpeği ve sahibini, bir uç noktadan diğerine geri dönmeden iki ayrı yolda yürüyor düşünün. Frechet mesafesi, bir köpeği ve sahibini bağlamak için gereken minimum uzunluktur. Hausdorff mesafesi keyfi nokta arasındaki mesafeyi alırken, Frechet metriği iki eğrinin akışını dikkate alır [11].

Tek Yön Mesafesi (OWD), ızgara temsili lokal minimum noktalarını kullanarak bilinen kuadratik algoritmalardan daha iyi bir karmaşıklık sağlar. Farklı ızgara hücre boyutu, seslerin doğruluğunu, performansını ve hassasiyetini ortadan kaldırmak için seçilebilir [11].

2.1.2.1. Simetrik segment yolu mesafesi (SSPD)

Bu mesafe şekil tabanlı bir mesafe hesaplama yöntemidir. Tüm yörüngeleri hesaba katar ve gürültüden daha az etkilenir. Noktadan-Dönüş mesafesinin ve SPD mesafesinin ortalaması alınarak hesaplanır. Ortalama yerine maksimum kullanılırsa, iki yörünge arasında Hausdorff işlevini kurtarır. İki konum arasında sadece bir mesafeyi hesaplamak işlemi çok hassas hale getirir.

SSPD algoritmasında hesaplama yapmak için bazı alt basamaklar mevcuttur. Temel olarak hesaplama için gerekli olan metodlar şunlardır. SPD (Segment Yolu

Mesafesi), point_to_trajectory fonksiyonu, point_to_segment fonksiyonu, öklit uzaklığı ve epsilon değerleri kullanılır. Bu fonksiyonlar ve kısımlar aşağıda ayrıntılı olarak yer alacaktır.

İlk olarak temel SSPD algoritmasının iki adet gezinge (trajectory) verisinin benzerlik analizi için kullanılacak formül şu şekildedir:

$$SSPD=(SPD(data1,data2)+SPD(data2,data1))/2 \quad (2.15)$$

Buradaki fonksiyon çift yönlü olarak hesaplama yapıp bu değerlerin ortalamasını alarak daha optimum bir sonuç elde etmemizi sağlar. Burada kullanılan SPD metodunun formülü ise şu şekildedir.

$$SPD=\frac{sum(map(lambda\ p:point\_to\_trajectory(p,data2),data1))}{len(data1)} \quad (2.16)$$

Bu formül içerisinde kullanılan p değeri nokta yerine kullanılmıştır. Burada ilk gezinge (trajectory) verisinin tüm noktaları tek tek işleme alınmaktadır. Burada hesaplama yapılırken kullanılan point_to_trajectory metodu ile bu noktalar diğer gezinge (trajectory) verisi ile karşılaştırma yapmamıza olanak sağlar. Bu değere nokta ile gezinge (trajectory) arasındaki hesaplama nedeni ile DPT (Distance Point to Trajectory) adını vermekteyiz. Bu metodun fonksiyonu ise şu şekildedir:

$$DPT=\min(lambda\ S1,S2:point\_to\_segment(p,S1,S2),t[:-1],t[1:]) \quad (2.17)$$

Bu hesaplama sırasında gezinge (trajectory) verileri kullanılarak oluşturulan segmentler kullanılmaktadır. Bu metod içerisinde epsilon ve öklit uzaklık değerleri kullanılarak hesaplama yapılır. Bu işlem sonrasında geriye dönük olarak bakıldığında SSPD değeri hesaplanmış olunur.

SSPD algoritmasının gezinge (trajectory) verileri üzerindeki çalışma mantığını bir örnek ile ele alacak olursak. Elimizde iki adet gezinge (trajectory) verisi bulunmaktadır. Bu örnek veriler daha kolay anlaşılabilir olması için sentetik verilerdir ve değerler tam sayı olarak seçilmiştir.

- data1 = {(1,2),(2,4),(1,5),(2,5),(3,4)}
- data2 = {(1,2),(2,4),(2,6),(2,5),(3,4)}

Bu hesaplama işlemi için öncelikle SPD(data1,data2) ve SPD(data2,data1) değerlerini hesaplamamız gerekir. Bu değerlerin hesaplanmasında kullanılan genel formül ise şu şekildedir.

$$SPD= \sum p:point\_to\_trajectory(p,t_2) \quad (2.18)$$

Burada p olarak ifade edilen değer data1 in tüm noktaları t_2 olarak ifade edilen ise data2 gezinge (trajectory) verisidir. Bu formül baz alınarak hesaplanan SPD(data1,data2) değerinin hesaplama sonucu 0,2'dir ve şu şekilde hesaplanmıştır.

$$SPD= \frac{\sum(0,0,1,0,0)}{5}=0,2 \quad (2.19)$$

Aynı şekilde SPD(data2,data1) değerinin sonucu da 0,2 olarak hesaplanır. Bu iki değerinin ortalamasını aldığımızda ise SSPD değerinin sonucunu elde etmiş oluruz. Bu hesaplama sonucuna göre ise SSPD değeri 0,2 olarak bulunmuş olunur.

2.2. Uygulama Tasarım ve Geliştirme

Uygulama tasarım ve geliştirme sırasında bir çeşit java uygulama geliştirme platformu olan NetBeans IDE üzerinde çalışılmıştır. Uygulama çalıştırma sırasında farklı algoritmalar ve veri seçim yöntemleri kullanılmıştır. Kullanılan bu algoritmalar LCSS, DTW, EDR ve SSPD şeklindedir. Kullanılan veri seçim yöntemleri ise otomatik veya manuel(el ile) şeklindedir. Uygulama Java Programlama Dili kullanılarak geliştirilmiştir.

2.2.1. Netbeans ide

NetBeans IDE, Oracle firması tarafından geliştirilen bir Integrated Development Environment (IDE: Bütünleşik Geliştirme Ortamı)'dır. Java, C/C++, PHP ve HTML5 diliyle profesyonel masaüstü, kurumsal, Web ve Mobil uygulamaları geliştirmek için kullanılabilir. Windows, Linux, Mac OS X ve Solaris de dahil olmak üzere birçok platformda çalışır. 2010 yılında ise NetBeans 1.000.000 aktif kullanıcı kitlesine ulaşmıştır [12]. Avantajları:

- Netbeans IDE'nin kurulumu kolaydır.
- Hızlı kullanıcı arayüzü ve birçok dilde desteklenebilme özelliği vardır.

- Geniş bir kullanıcı kitlesi, dünya çapında yüze yaklaşan ortakları olan başarılı bir açık kaynak kod projesidir.
- NetBeans IDE'yi genişletmek için çok sayıda modül bulunur.
- NetBeans IDE 6.5 sürümünden sonra zengin JavaScript özellikleri, Spring web framework kullanım gibi bir kısım yeni özellikler gelmiştir.
- Ücretsiz bir üründür.
- Kolay ve verimli proje yönetimi sağlar [12].

2.2.2. Java programlama dili

Java programlama dili Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış gerçek nesneye yönelik, platformdan bağımsız, yüksek performanslı, çok işlevli, yüksek seviye, adım adım işletilen bir dildir. Java programlama dilinin başlıca nitelikleri; basittir, nesne yönelimlidir, dağıtıktır, çoklu iş yeteneği sağlar, dinamiktir, mimari yapıdan bağımsızdır, taşınabilir, yüksek performansı vardır, sağlamdır ve güvenlidir. Java programlama dili ile yazılan çalışmada, kodlar ilk olarak java derleyicisi ile derlenir. Sonuçta bytecode adı verilen bir tür makine kodu ortaya çıkar. Platform bağımsızlığını sağlayan şey bytecode'dur. Çünkü bir kere bytecode oluştuktan sonra yazılım tüm işletim sistemlerinde çalışabilir. Bu bytecode Java Virtual Machine (Java Sanal Makinesi) tarafından adım adım işletilir.

2.2.3. Jar dosyaları

Çalışma sırasında uygulama geliştirilirken birkaç tane gelişmiş jar dosyası kullanılmıştır. Bu jar dosyaları; gnujaxp.jar, jcommon.jar, jfreechart.jar, junit.jar ve servlet.jar şeklindedir. Bu dosyalar yazılım sonucunda birçok amaca hizmet etmektedir. Bu dosyalardan jcommon.jar ve jfreechart.jar grafik çizim için gerekli olan dosyalardır. Junit.jar ise oluşturulan Java kodu için bir birim test çerçevesidir. Diğer dosyalar ise uygulamanın daha stabil çalışması için gereklidir.

2.2.4. Uygulamada kullanılan veri setleri

Çalışma sırasında gezinge (trajectory) veri seti için üç tür mevcuttur. Bu türler; gerçek gps verilerinden elde edilen veri seti parçaları, sentetik şekillerden elde edilen

veri seti ve anlık olarak koordinat düzleminden seçim yolu ile elde edilen veri setleridir. Bu üç tür veri seti arasındaki farklar görselleştirme sonucu ile net bir şekilde görülmektedir.

2.2.4.1. Görüntü'den gezinge (trajectory) verisi oluşturma

Görüntü üzerinde görüntü işleme işlemleri yapılıp sentetik gezinge (trajectory) verileri elde edilmiştir. Gezinge (trajectory) verilerinin koordinat değerleri aynı değildir, algoritmaların benzerlik analiz hesabı yapabilmesi için benzer karakterlerin resimleri kullanılarak gezinge (trajectory) verileri, farklı işlemler ile indirgeme yapılarak elde edilmiştir.

Görüntüler üzerinde yapılan işlemler benzerlik analizi için kullanılacak verilerin elde edilmesi için seçilen iki resim üzerinde de uygulanmıştır. İşlemler sonucunda elde edilen gezinge (trajectory) verileri [x, y] şeklinde koordinat değerleridir.

Burada değerler gerçek enlem-boylam değerlerine tam uymadığı için bu verileri sentetik gezinge (trajectory) verileri olarak ele aldık. Örnek veri kümesi:

[…[366, 36], [216, 275], [171, 86],

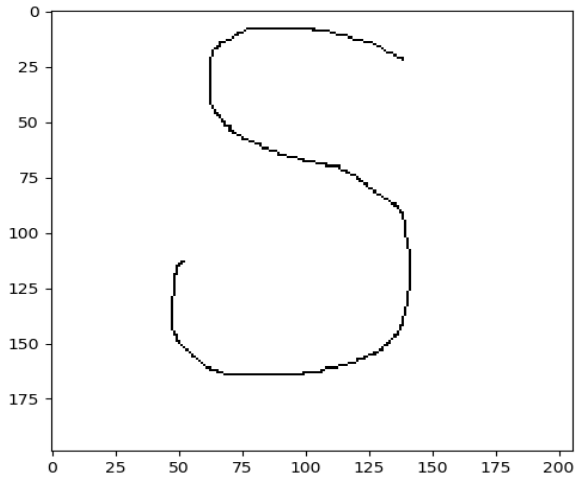
[206, 318], [191, 155], [41, 246],

[141, 264], [186, 176], [211, 60],

[221, 330], [251, 85], [331, 329]…] şeklindedir.

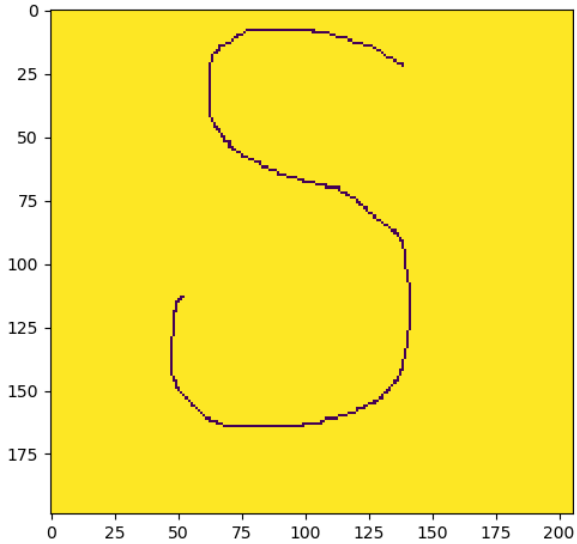
Görüntüden gezinge (trajectory) verisi elde edilmesi için yapılması gereken bir dizi işlem vardır. Bu işlemler, görüntü okuma, görüntü üzerinde kenar algılama, gri tonlama, ikili görüntü elde etme şeklindedir.

İlk olarak dosyadan benzerlik analizi için kullanılacak karakterlerin görüntü verisi gri tonlama olarak okunur, bölgesel segmentasyona hazırlık amacı ile RGB formatındaki görüntü, gri tonlamalı hale getirilmiştir. İlk okunan veri şu şekildedir.



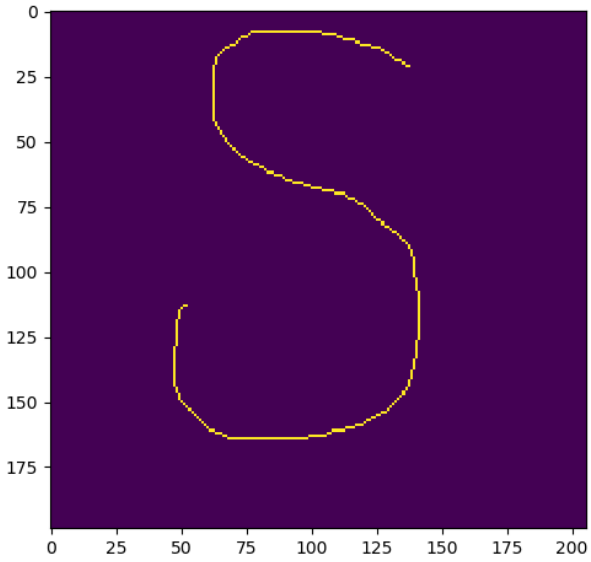
Şekil 2.3. S karakterinin ilk görüntüsü

Okuma işlemi yapıldıktan sonra görüntü üzerinde kenar algılama işlemi yapılır. Bu işlem sırasında görüntü birbirini izleyen geçişler yaparak iskeletleştirmeye çalışılır. Her geçişte, kenarlık pikselleri, karşılık gelen nesnenin bağlantısını kesmemeleri şartıyla tanımlanır ve kaldırılır. Bunun sonucunda görüntüyü temsil eden kenarlar belirlenmiş olur.



Şekil 2.4. Görüntüden kenar tespiti işlemi

Kenar tespiti yapıldıktan sonra elde edilen veriler ikili görüntüye çevrilir ve görüntü işleme adımları tamamlanmış olunur. Bu işlemler sonucunda elde edilen gezinge (trajectory) verisi oluşturma işlemleri için girdi olarak kullanılacaktır. Görüntü üzerinde yapılan diğer işlemler sonucunda elde edilen sonuç aşağıdaki şekildedir.

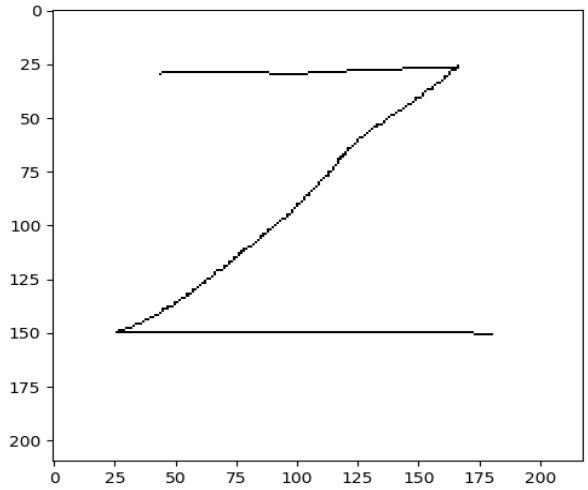


Şekil 2.5. Görüntü işleme ile elde edilen görüntü

İşlenen görüntü sonucu elde edilen ikili görüntü üzerinde gezinge (trajectory) veri değerleri bulunmaya çalışılır. Gezinge(trajactory) verisi oluşturmak için elde edilen veri üzerinden, piksellerin başlangıç ve bitiş noktaları ile kesişim noktaları belirlenir ve bunlar gezinge (trajectory) verisinin koordinat değerleri olarak işaretlenir.

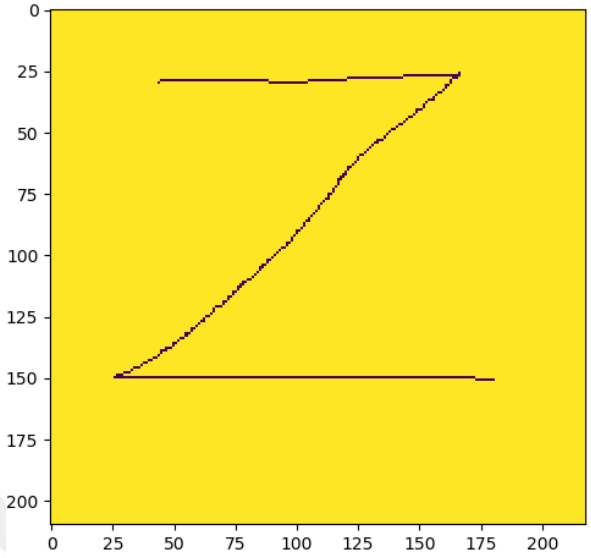
Başka bir gezinge (trajectory) verisi olarak ise Z karakterinin görüntüsünü kullanarak ikinci bir veri elde etmeye çalıştık.

Burada ilk olarak orijinal görüntümüz ise Şekil 2.6.'da göründüğü gibidir.



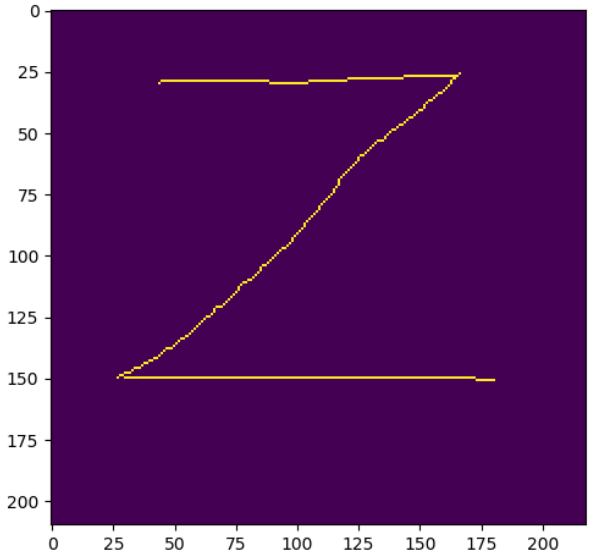
Şekil 2.6. S karakterinin ilk görüntüsü

Okuma işlemi yapıldıktan sonra görüntü üzerinde kenar algılama işlemi yapılır. Bu işlemin ise Z görüntüsü üzerindeki sonucu ise Şekil 2.7.'de göründüğü gibidir.



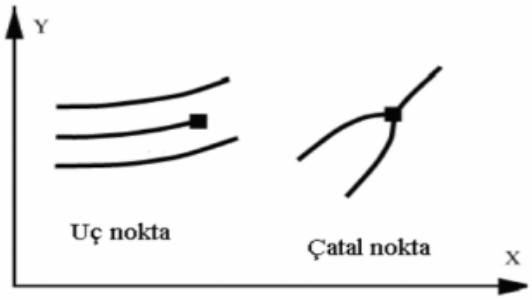
Şekil 2.7. Görüntüden kenar tespiti işlemi

Kenar tespiti yapıldıktan sonra elde edilen veriler ikili görüntüye çevrilir ve görüntü işleme adımları tamamlanmış olunur. Buradaki işlemlerden sonra ise Z görüntüsünün yeni hali Şekil 2.8.'deki gibidir.



Şekil 2.8. Görüntü işleme ile elde edilen görüntü

Gezinge verisi koordinat noktalarını oluşturmak için Minutiae Yöntemi kullanıldı. Minutiae Yöntemi iskelet resim üzerinde ki hatlarda bulunan çizgilerin uç nokta veya çatallaşmasının teşhisinde bulunmaya yardım eder [13] .



Şekil 2.9. Minutiae ayırt edilen özellik noktaları [14]

Minutiae Yöntemi'nde en çok kullanılan tekniklerden biri olan Crossing Number (CN) yöntemini kullandık. Bu yöntemde 3x3 komşu piksel belirleme maskesi matrisi kullanarak her piksel maskelenir ve bir CN değeri elde edildi. Sonuç olarak bu CN değerine göre gezinge (trajectory) verisinin koordinat değerleri oluşturuldu. Bu işlemler sırasında kullanılan P maskeleme matrisi oluşturulurken hedef piksel değeri ve bu noktanın birinci dereceden komşu piksel değerleri kullanılmıştır. Bu işlem Tablo 2.10.'da yer almaktadır.

Tablo 2.10. Maskeleme matrisi

P4	P3	P2
P5	P	P1
P6	P7	P8

Tablo 2.11. CN belirleyici değerleri

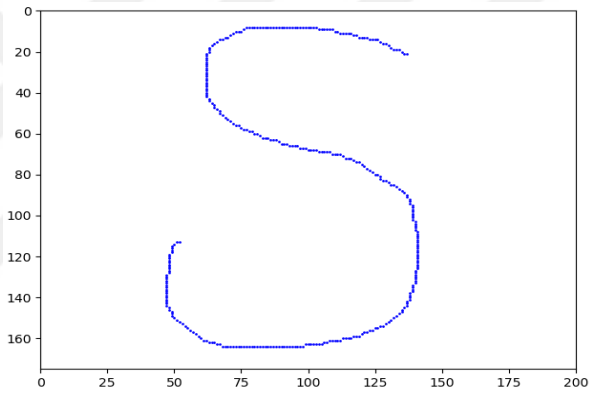
CN	Açıklama
0	Tekil nokta
1	Çatallanma bitiş noktası
2	Çatallanma devamı noktası
3	Çatallanma noktası
4	Kesişme noktası

CN Maskeleme hesaplama formülü ise şu şekildedir.

$$CN=\frac{1}{2}\sum_{i=1}^{8}|P_i-P_{i+1}| \quad (2.20)$$

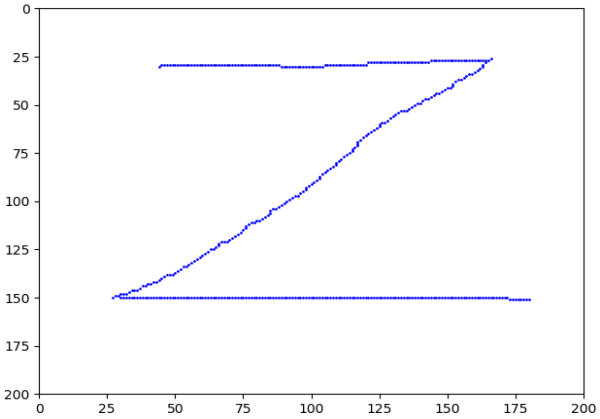
Crossing Number yönteminde görüntüde bulunan her piksel, maskeleme matrisine Formül 1 deki gibi uygulandı. Bu işlemin Tablo 2 deki gibi [0,4] aralığında belirleyici değer elde edildi. Bu belirleyicilerden değeri 4 olanlar (Kesişme noktası-Cross) gezinge (trajectory) verisi için koordinat noktası seçildi [15]

Minutiae Yöntemi görüntü üzerindeki her piksel için çağırıldığı için mxn piksellik bir görüntü için zaman karmaşıklığı O(m*n) olur. İşlemler sonucunda görüntüden elde edilen graf düğümleri Şekil 2.10.'daki gibidir.



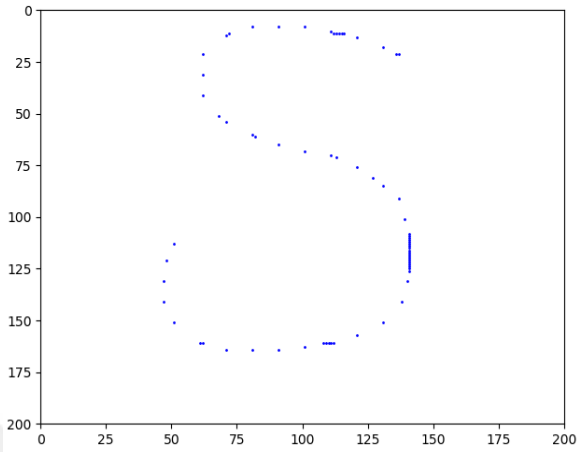
Şekil 2.10. Elde edilen gezinge (trajectory) S veri koordinatlarının son hali

Uygulama genelinde bu mantık ile farklı karakterlerin (S, Ş, Z, s, ş, z) resimleri kullanılarak sentetik gezinge (trajectory) verileri oluşturuldu.



Şekil 2.11. Z gezinge (trajectory) verisi

Bu işlem sırasında daha önceden bahsettiğimiz gibi verilerin minimal ve daha kararlı hale getirilmesi için ek olarak yapılan işlem sonucu elde edilen minimal gezinge (trajectory) verisi ise Şekil 2.12.'deki gibidir.



Şekil 2.12. S'in minimal gezinge (trajectory) veri koordinatlarının son hali

Görüntüden gezinge (trajectory) verisi elde edilmesi işlemi Python programlama dili ile yapıldı. Bu sırada çok fazla python kütüphanesi kullanıldı. Bu Python kütüphaneleri şu şekildedir:

- Pillow
- opencv-python
- matplotlib
- scikit-image
- networkx

Burada bu kütüphanelerden de biraz bahsetmenin doğru olacağını düşündüğümüz için ekleme yaptık.

Pillow**,** Python Resim Kütüphanesi, Python'da resim işlemlerini kolayca yapabilmek için geliştirilmiş kütüphanedir [16]. Pillow ile yapılabilecek bazı görüntü işlemleri:

- Resim boyutlandırma ve belirli bir formatta "RGB" çizdirme.
- Resme en büyük etki olarak ImageFilter ile özellik verme

ImageFillter ile verilebilecek yeni özellikler için parametreler;

- BLUR
- CONTOUR
- DETAIL
- EDGE_ENHANCE
- EDGE_ENHANCE_MORE
- EMBOSS
- FIND_EDGES
- SMOOTH
- SMOOTH_MORE
- SHARPEN [16]

Opencv-python**,** OpenCV (Open Source Computer Vision) açık kaynak kodlu görüntü işleme kütüphanesidir. 1999 yılında İntel tarafından geliştirilmeye başlanmış daha sonra Itseez, Willow, Nvidia, AMD, Google gibi şirket ve toplulukların desteği ile gelişim süreci devam etmektedir. İlk sürüm olan OpenCV alfa 2000 yılında piyasaya çıkmıştır. İlk etapta C programlama dili ile geliştirilmeye başlanmış ve daha sonra birçok algoritması C++ dili ile geliştirilmiştir [17].

OpenCV platform bağımsız bir kütüphanedir, bu sayede Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında çalışabilmektedir. C++, C, Python, Java, Matlab, EmguCV kütüphanesi aracılığıyla da Visual Basic.Net, C# ve Visual C++ dilleri ile topluluklar tarafından geliştirilen farklı wrapperlar aracılığıyla Perl ve Ruby programlama dilleri ile kolaylıkla OpenCV uygulamaları geliştirilebilir [17].

OpenCV Bileşenleri;

- Core, OpenCV'nin temel fonksiyonları ve matris, point, size gibi veri yapılarını bulundurur. Ayrıca görüntü üzerine çizim yapabilmek için kullanılabilecek metotları ve XML işlemleri için gerekli bileşenleri barındırır [17].
- HighGui, resim görüntüleme, pencereleri yönetme ve grafiksel kullanıcı arabirimleri için gerekli olabilecek metotları barındırır. 3.0 öncesi sürümlerde dosya sistemi üzerinden resim dosyası okuma ve yazma işlemlerini yerine getiren metotları barındırmaktaydı [17].
- Imgproc, filtreleme operatörleri, kenar bulma, nesne belirleme, renk uzayı yönetimi, renk yönetimi ve eşikleme gibi neredeyse tüm fonksiyonları içine alan

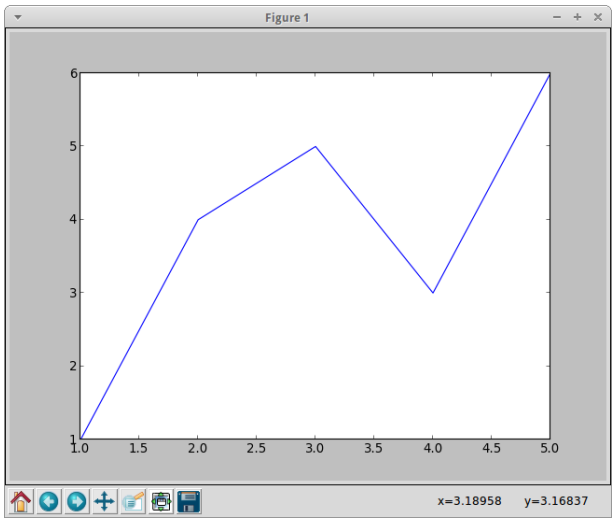
bir pakettir. 3 ve sonra sürümlerde bazı fonksiyonlar değişmiş olsada 2 ve 3 sürümünde de birçok fonksiyon aynıdır [17].

- Imgcodecs, dosya sistemi üzerinden resim ve video okuma/yazma işlemlerini yerine getiren metotları barındırmaktadır [17].
- Videoio, kameralara ve video cihazlarına erişmek ve görüntü almak ve görüntü yazmak için gerekli metotları barındırır [17].

OpenCV alternatif kütüphanelerine; Matlab, Halcon, OpenFrameworks, CIMG ve Fiji örnek olarak verilebilir.

Matplotlib, grafik çizim paketi Python'la bilimsel programlamanın en önemli araçlarından birisidir. Çok kuvvetli bir paket olan Matplotlib ile verileri etkileşimli olarak görselleştirebilir, yayınlamaya uygun yüksek kalitede çıktılar hazırlayabiliriz. Hem iki boyutlu hem de üç boyutlu grafikler üretilebilir [18].

Matplotlib grafikleri çok incelikli şekilde düzenlenebilir, aşağıdaki Şekil 2.13.'de çizilmiş olan bir adet grafik görülebilir. Burada ilk data [1,2,3,4,5] ikinci data ise [1,4,5,3,6] şeklindedir.



Şekil 2.13. Matplotlib ile çizilmiş bir grafik çizimi [18]

Scikit-image, Python programlama dili için açık kaynaklı bir görüntü işleme kütüphanesidir. Segmentasyon, geometrik dönüşümler, renk alanı manipülasyonu, analiz, filtreleme, morfoloji, özellik tespiti ve daha fazlası için algoritmalar içerir [19].

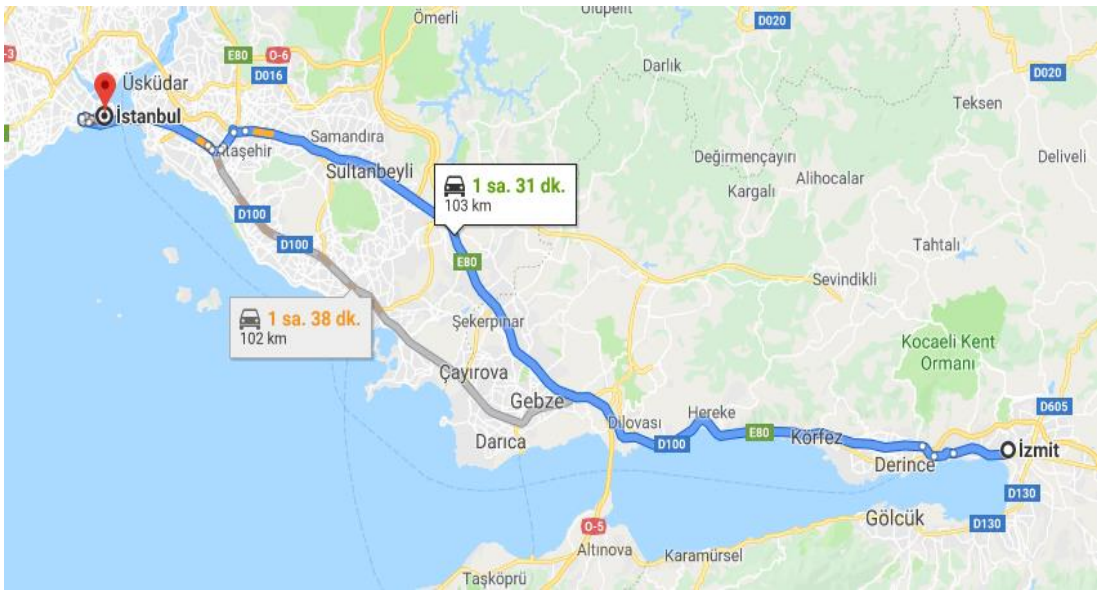
Python sayısal ve bilimsel kütüphaneleri NumPy ve SciPy ile birlikte çalışmak üzere tasarlanmıştır [19].

Networkx, Karmaşık ağların yapısını, dinamiklerini ve fonksiyonlarını oluşturma, değiştirme ve inceleme için bir Python paketidir [20]. Özellikleri;

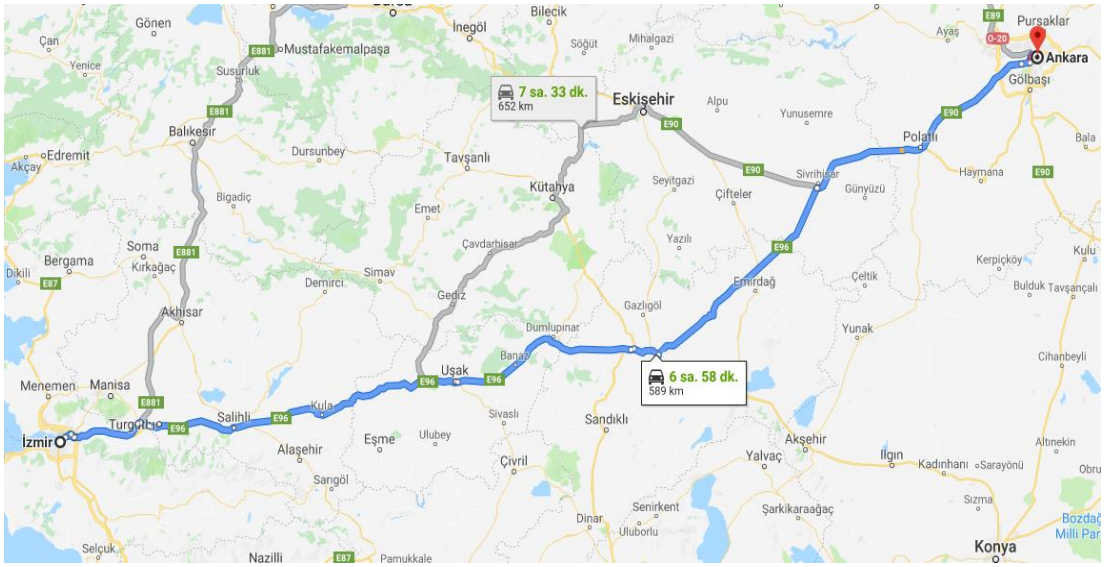
- Grafikler, dipnotlar ve çoklu belgeler için veri yapıları
- Birçok standart grafik algoritması
- Ağ yapısı ve analiz önlemleri
- Klasik grafikler, rastgele grafikler ve sentetik ağlar için jeneratörler
- Düğümler "her şey" olabilir (ör. Metin, görüntüler, XML kayıtları)
- Kenarlar isteğe bağlı verileri tutabilir (örneğin ağırlıklar, zaman serileri)
- Açık kaynak kodlu 3-maddeli BSD lisansı
- % 90'dan fazla kod kapsamı ile iyi test edildi
- Python'un ek avantajları arasında hızlı prototip oluşturma, öğretmesi kolay ve çok platformlu [20]

2.2.5. Harita verileri ile karşılaştırma işlemi

Harita kullanılarak elde edilen veriler üzerinde karşılaştırma yapılmaya çalışıldı. Burada ilk olarak belirlenen iki şehir arasında en kısa iki rota belirlenip bu iki rotanın benzerliği hesaplamaya çalışıldı.

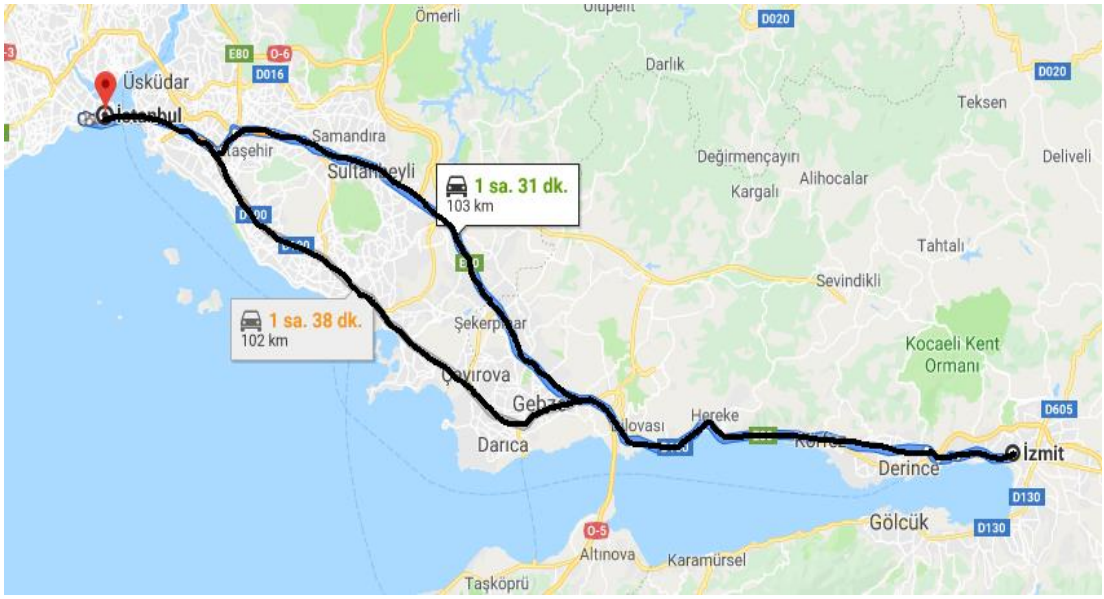


Şekil 2.14. İzmit-İstanbul arası en kısa iki rotanın harita görüntüsü

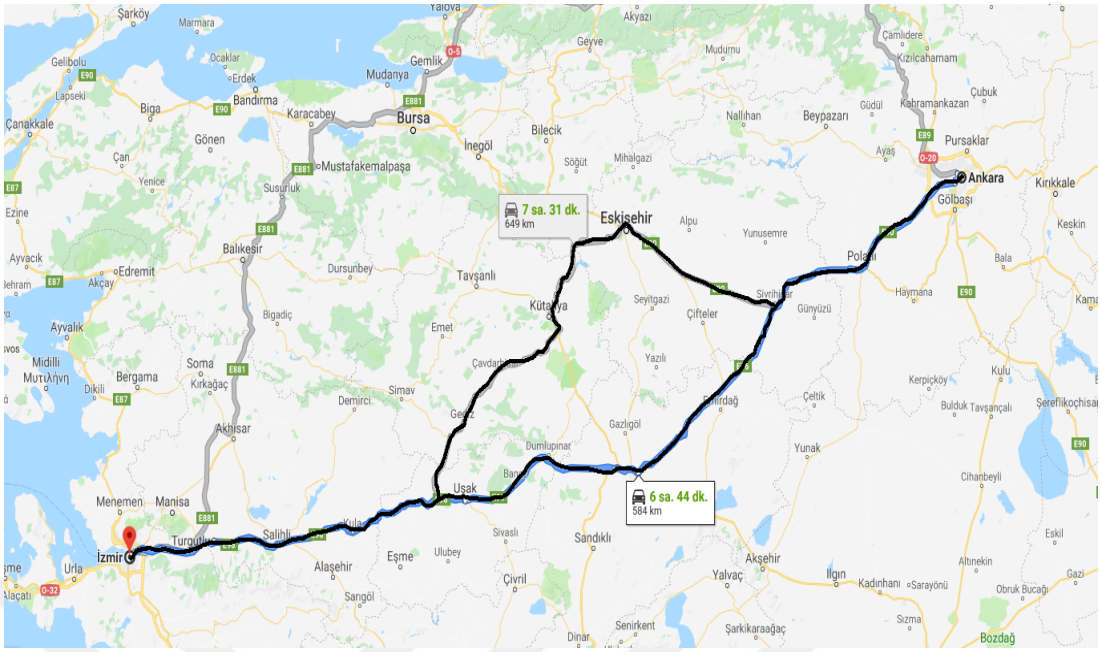


Şekil 2.15. İzmir-Ankara arası en kısa iki rotanın harita görüntüsü

Yukarıdaki Şekil 2.14. daha ayrıntılı olarak ele alınmak istenirse, ilk olarak iki konum belirlendi. Bu konumlardan ilki İzmit ili ikincisi ise İstanbul ilidir. Bu iki ilin konumları başlangıç ve bitiş noktaları olarak ele alınmaktadır. Bu iki nokta arasında harita yardımı ile bir yol tarifi alınır ise bu iki yol ortaya çıkmaktadır. Bu yollardan ilki 102 km olup süre önerisi olarak 1 saat 38 dakika olarak verilmektedir. Diğer bir yol ise 103 km olup 1 saat 31 dakika süreceği belirtilmiştir. Bu bilgiler Google haritalar yardımı ile elde edilmektedir.



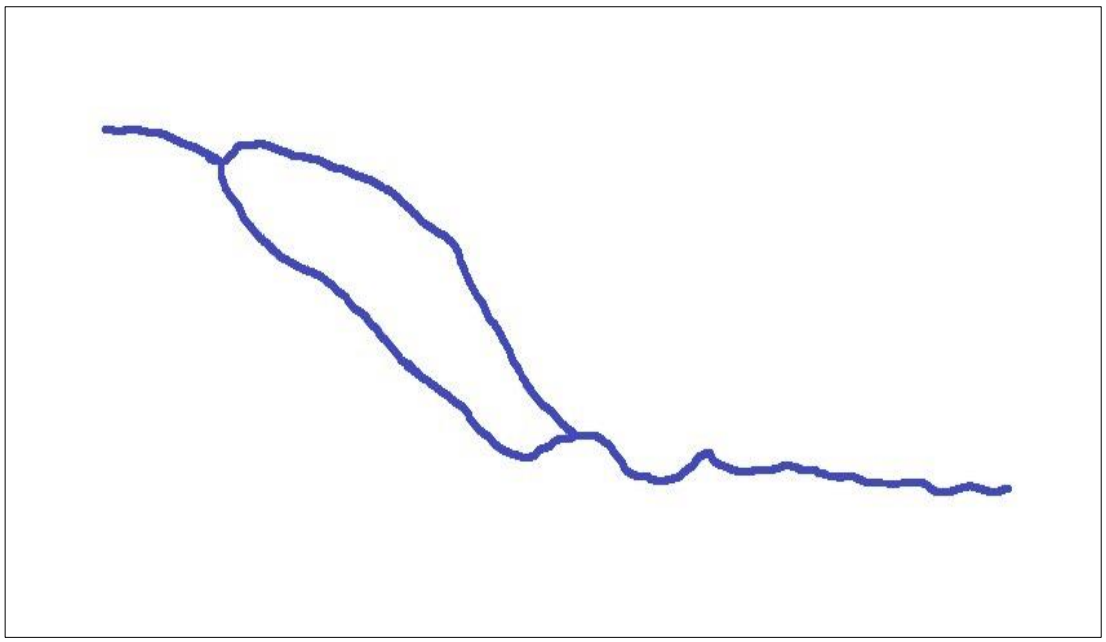
Şekil 2.16. İzmit-İstanbul arası harita üzerine yolların çizilmesi



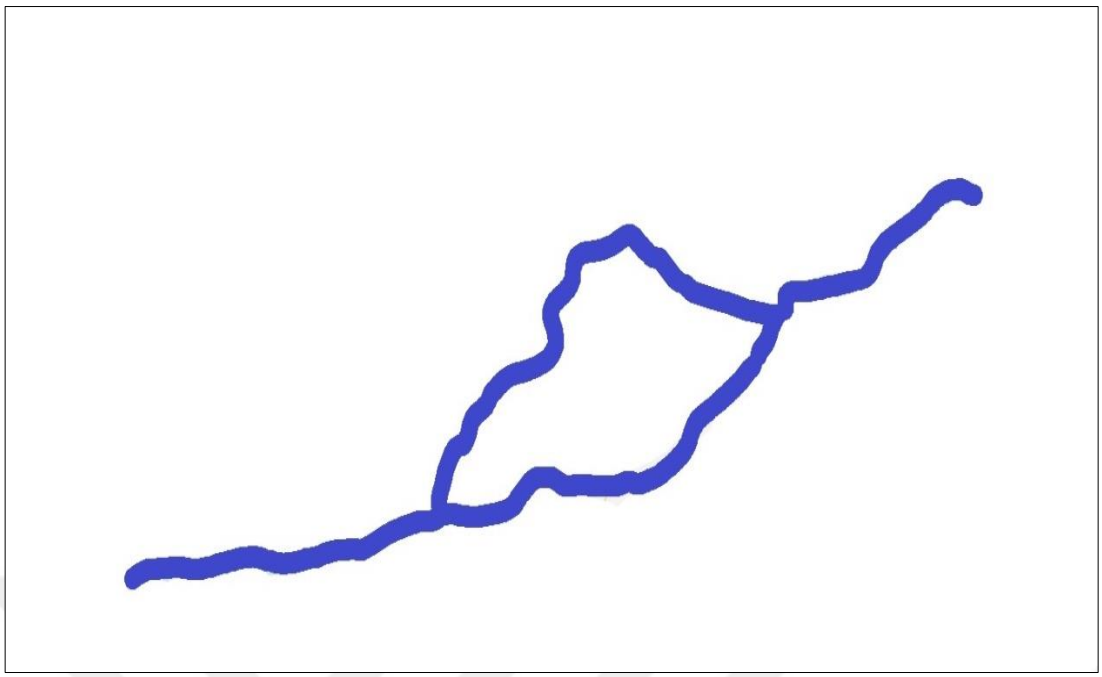
Şekil 2.17. İzmir-Ankara arası harita üzerine yolların çizilmesi

Daha sonra bu yol bilgilerinin haritadan çıkarılması işlemi yapılmaktadır. Bu kısım manuel olarak ele alınıp gerçeklemeye çalışılmıştır. Bu işlem paint üzerinde yapılmıştır. Bu sayede harita üzerinde yollar belirlenmiş olmaktadır.

Yollar belirlendikten sonra gerekli olan yol bilgileri dışındaki diğer noktalar temizlenip sadece gerekli olan değerler elde edilmiştir.

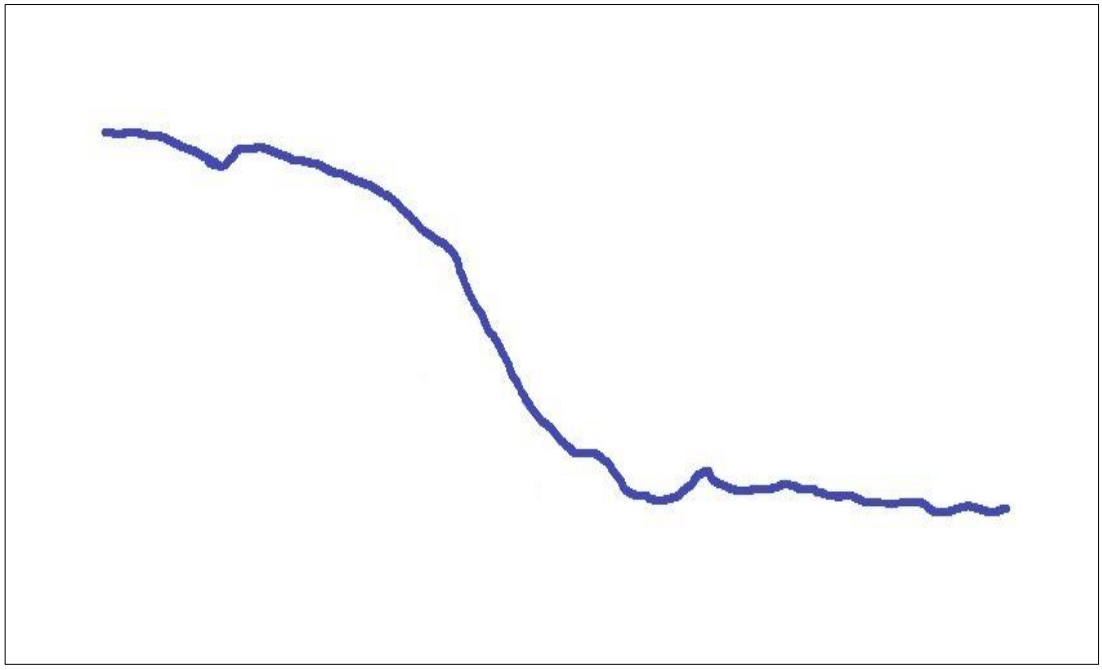


Şekil 2.18. İzmit-İstanbul arası temizlenmiş yol görüntüsü

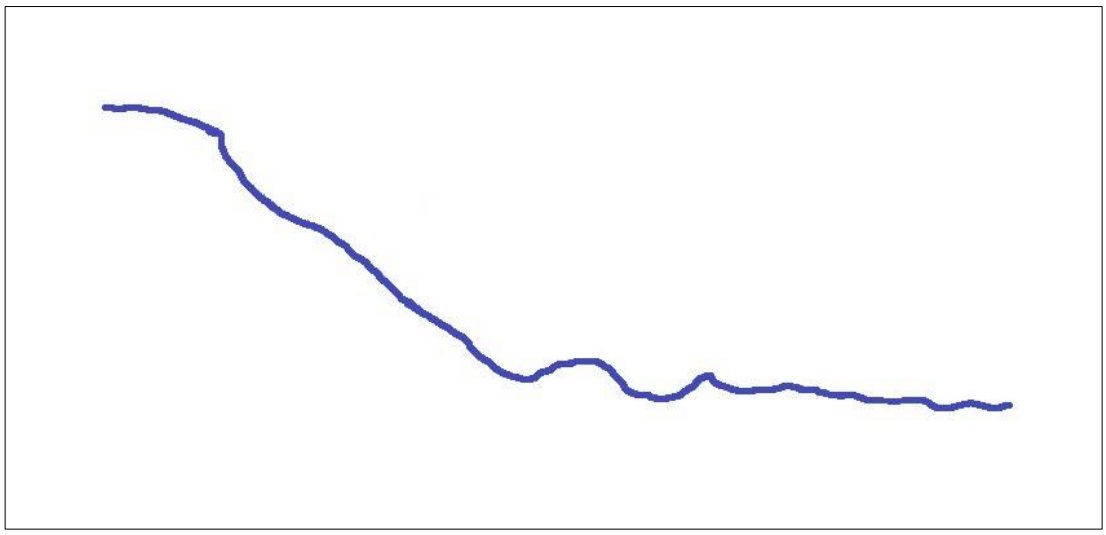


Şekil 2.19. İzmir-Ankara arası temizlenmiş yol görüntüsü

Gerekli temizleme işlemi sonucu elde edilen yol görüntüsünde elimizde toplam yol değerleri bulunmaktadır. Bize gerekli olan şu aşamadan sonra iki adet yol görüntüsünün ayrıştırılması. Bu ayrıştırma sonucunda iki tane yol bilgisine ulaşmış olacağız.



Şekil 2.20. İzmit-İstanbul üst kısımdaki yol bilgisinin görüntüsü

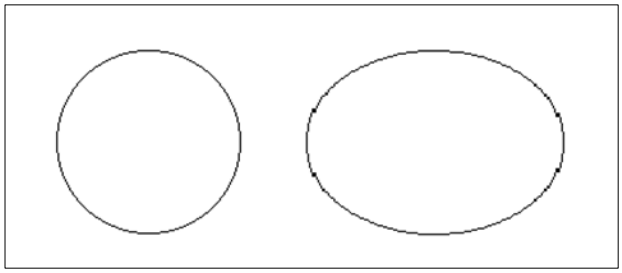


Şekil 2.21. İzmit-İstanbul alt kısımdaki yol bilgisinin görüntüsü

İlk olarak üst kısımdaki yol bilgisi olan Şekil 2.20.'yi toplam yoldan elde ettik daha sonra ise alt kısımdaki yol bilgisi Şekil 2.21.'i elde ettik. Bu sayede karşılaştırma yapılacak iki görüntü de elde edilmiş oldu. Bu görüntülerin karşılaştırmasının yapılabilmesi için görüntüden gezinge (trajectory) verisi elde edilmesi işlemleri uygulanmıştır. Bunun sonucunda iki rota arasında karşılaştırma yapmak için gerekli olan gezinge (trajectory) verisi oluşturulmuş olunur.

2.2.6. Geometrik şekillerin karşılaştırma işlemi

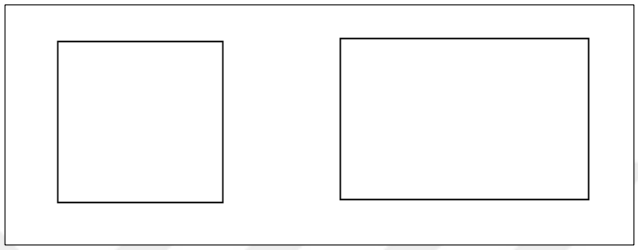
Benzerlik analizi için benzerliği bilinen geometrik şekiller üzerine bir analiz işlemi yapıldı. Bu işlem için benzerliği bilinen geometrik şekil çiftleri seçildi. Bu çiftler şu şekildedir; Daire-Elips, Kare-Dikdörtgen, Eşkenar Üçgen-İkizkenar Üçgen. Burada bu çiftlerin seçilmiş olmasının sebebi bu şekillerin benzerliklerinin yüksek olduğunun düşünülmesi ve birbirinden türetilmiş olmalarıdır. Şöyle ki bir elips dairenin üstten veya yanlardan sıkıştırılması sonucu oluşmaktadır.



Şekil 2.22. Daire ve elips şekilleri görüntüsü

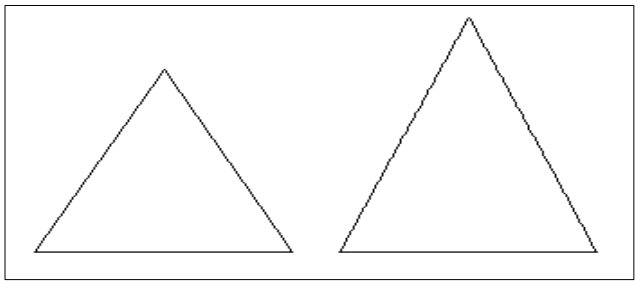
Burada kullanılan elips şekli dairenin sol ve sağ tarafından genişletilmesi sonucu oluşturulmuştur. Bu şekiller kullanılarak benzerlik analizi yapılabilmesi için yine görüntüden gezinge (trajectory) verisi elde edilmesi yöntemi kullanılmıştır.

Farklı geometrik şekillerden olan kare ve dikdörtgenin benzerliğinde de daire ve elipse benzer bir durum bulunmaktadır. Aynı şekilde karenin sağ-sol veya üst-alt genişletme, daraltma işlemleri sonucu dikdörtgen elde edilmektedir.



Şekil 2.23. Kare ve dikdörtgen şekilleri görüntüsü

Farklı geometrik şekillerden olan ikizkenar üçgen ve eşkenar üçgen benzerliğinde de diğerlerine benzer bir durum bulunmaktadır. Aynı şekilde eşkenar üçgenin sağ-sol veya üst-alt genişletme, daraltma işlemleri sonucu ikizkenar üçgen elde edilmektedir.



Şekil 2.24. Eşkenar üçgen ve ikizkenar üçgen şekilleri görüntüsü

2.3. Uygulama Bölümleri

Uygulama içerisinde farklı alt bölüm mevcuttur. Bu bölümler uygulamanın kullanıcı isteğini sınırlama olarak düşünülebilir. Uygulama içerisinde giriş ekranı ve sonrasında yapılan seçimlere göre çıktılar ve sonuçlar elde edilir.

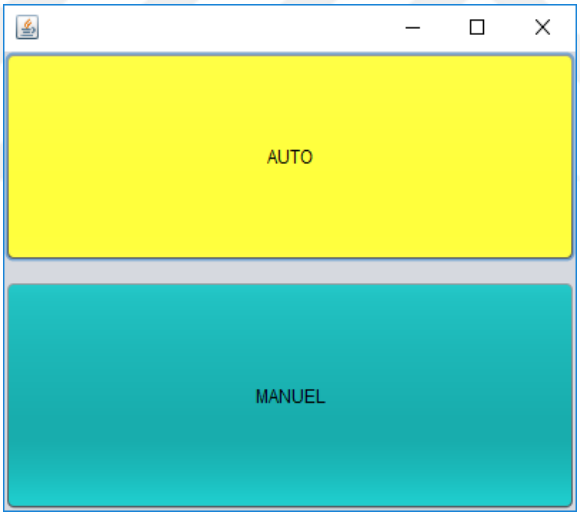
Bu sonuçların kolay karşılaştırılması için düzenlemeler yapılmıştır. Uygulamalar sonucunda elde edilen sonuçlar kullanıcıya toplu olarak veya tekil olarak isteğe bağlı aktarılmaktadır. Uygulama içerisindeki bölümler;

- Uygulama giriş ekranı
- Seçimli arayüz ekranı
- Çizimli arayüz ekranı
- Hesaplama ve sonuç alanı

2.3.1. Uygulama giriş ekranı

Uygulama ilk açıldığında karşımıza gelen seçim ekrandır. Bu ekran sayesinde benzerlik analizinde kullanılacak gezinge (trajectory) verilerinin yapısını seçmiş olacağız. Bu ekranda bize iki seçenek sunulmaktadır.

Bu seçeneklerden birisi önceden hazırlanmış veri setlerini seçerek analiz yapmamızı sağlar. Diğeri ise veri setini kendimizin koordinat düzlemi yardımı ile oluşturmamızı sağlar.



Şekil 2.25. Uygulama giriş ekranı

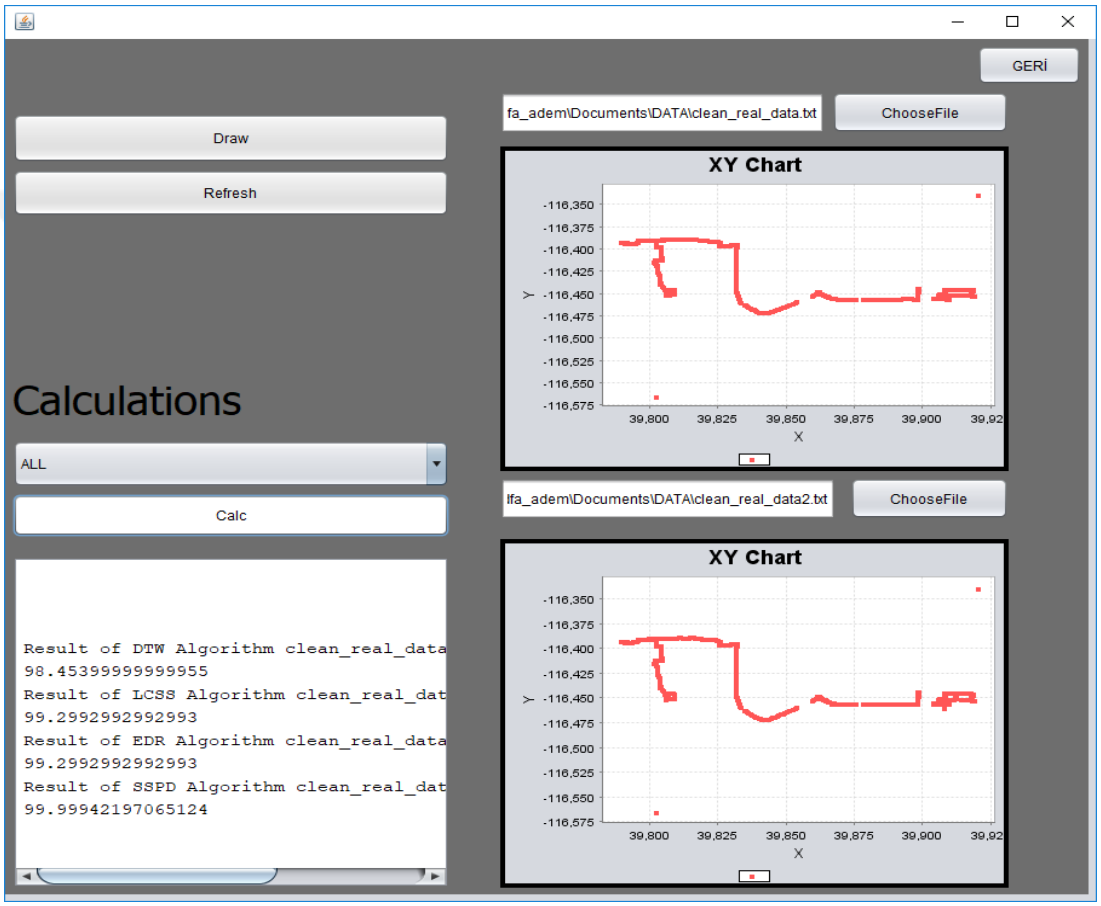
Uygulama Giriş Ekranı üzerinde iki adet buton bulunmaktadır. Bu butonlar yardımı ile seçim yapılmaktadır. Bu ekran bizi ilgili arayüz ekranlarına yönlendirmek için kullanılmaktadır. Bu ekranın java sınıfı javax.swing.JFrame'den extend edilerek oluşturulmuştur.

2.3.2. Seçimli arayüz ekranı

Bu ekran üzerinde kullanıcının işlem yapabileceği birçok arayüz elemanı bulunmaktadır. Sol üstte alt alta iki adet buton bulunmaktadır. Bu butonların

görevleri sırası ile seçilen verilerin görselleştirilmesi ve arayüzün yenilenip temizlenmesidir.

Arayüzün komple sağ bölümü dosya seçimine ve seçilen dosyaların görselleştirilmesine ayrılmıştır. Buradan seçilen iki adet gezinge (trajectory) verisi çizim butonu sayesinde görselleştirilmektedir.



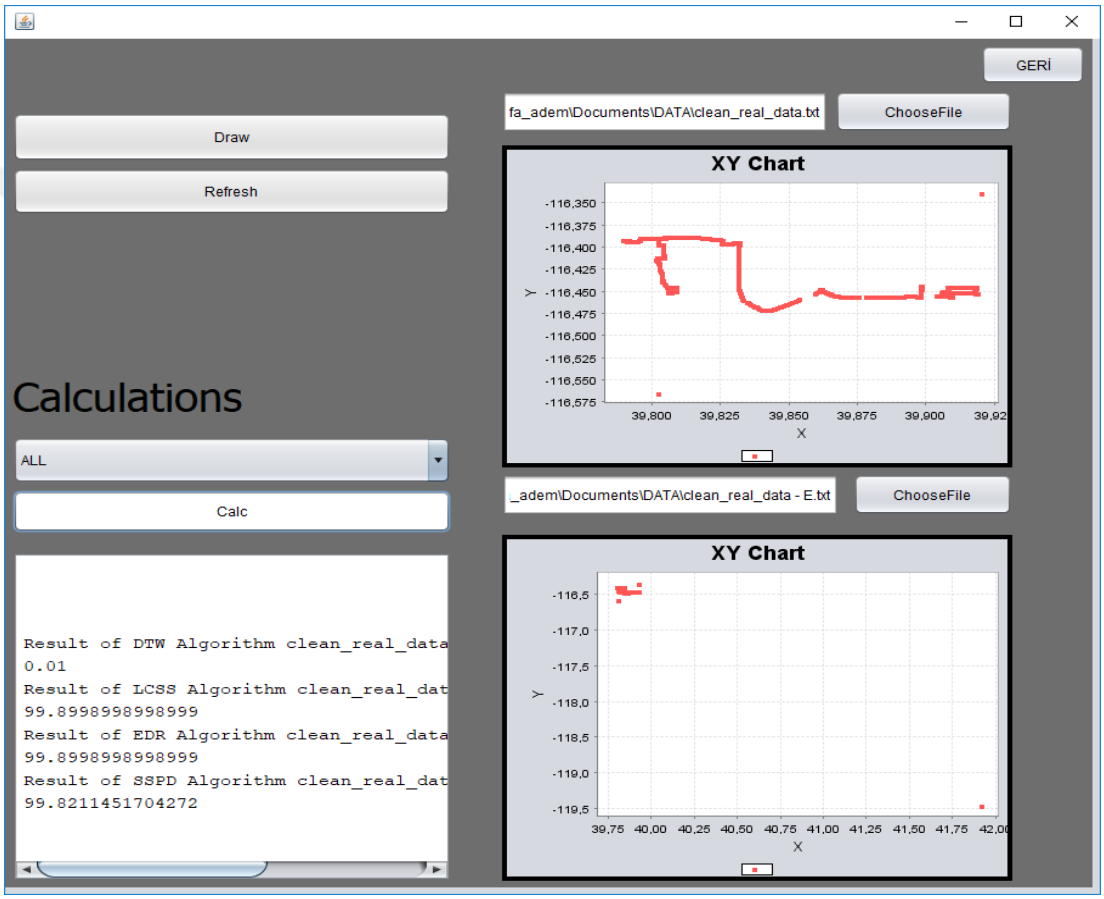
Şekil 2.26. Gerçek iki verinin karşılaştırılması sonucu elde edilen görüntü

Bu ekranda gerçek veya sentetik gezinge (trajectory) verileriden istenilen verinin seçimi ve çizimi yapılabilir. Bu seçim ve çizim işlemlerinden sonra hesaplama işlemine geçilir.

Bu işlem için sol altta yer alan kısım kullanılır. Bu bölümde bir adet açılır kapanır seçim alanı bulunmaktadır. Bu alan sayesinde tek bir algoritma LCSS, DTW, EDR veya SSPD seçilebilir ya da tüm algoritmalar birlikte seçilebilir.

Seçim işlemi sonrası hesaplama butonuna basılır ve istenilen algoritmaların sonuçları sonuç alanında listelenir. Bu işlem sonrası sağ en üstteki geri butonu sayesinde giriş ekranına dönüş sağlanabilir.

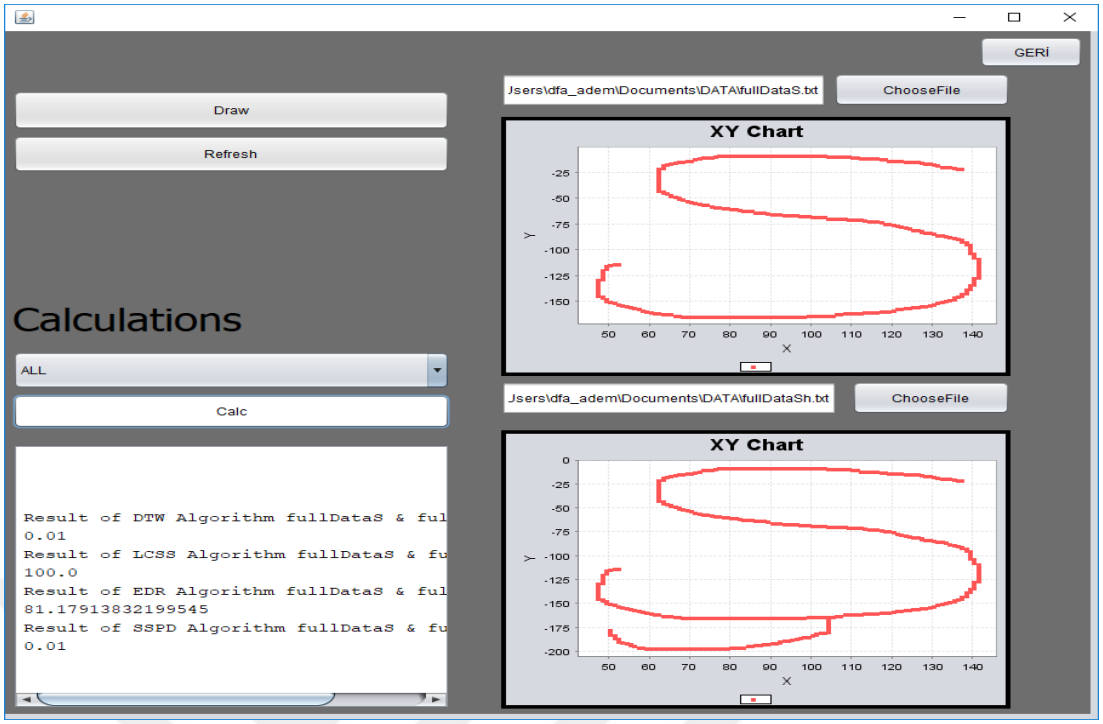
Gerçek gezinge (trajectory) verisine gürültü eklenip elde edilen yeni gezinge (trajectory) verisinin karşılaştırmasını yaparak sonuçları görüntüledik. Bu şekilde yaparak algoritmaların gürültüye vermiş olduğu tepkiyi ölçmüş olduk.



Şekil 2.27. Gerçek ve gürültü eklenmiş verilerin karşılaştırılmasının sonucu

Şekil 2.27.'de verilerin görünümleri çok farklı gibi görünebilir ancak gerçekte çok benzer şekillerdir. Sadece gürültülü, yani farklı ve uzak noktalar eklendiği için elde edilen yeni bir veri setidir. Bu benzerliği sonuçlar üzerinde rahatlıkla görebilmekteyiz.

Kullanılan bir diğer gezinge (trajectory) veri çifti S ve Ş Türkçe karakterleri kullanılarak elde edilen sentetik veriler ile yapılan karşılaştırmadır. Bu tip veriler kullanılmasının sebebi yine benzer olduklarının bilinmesidir.

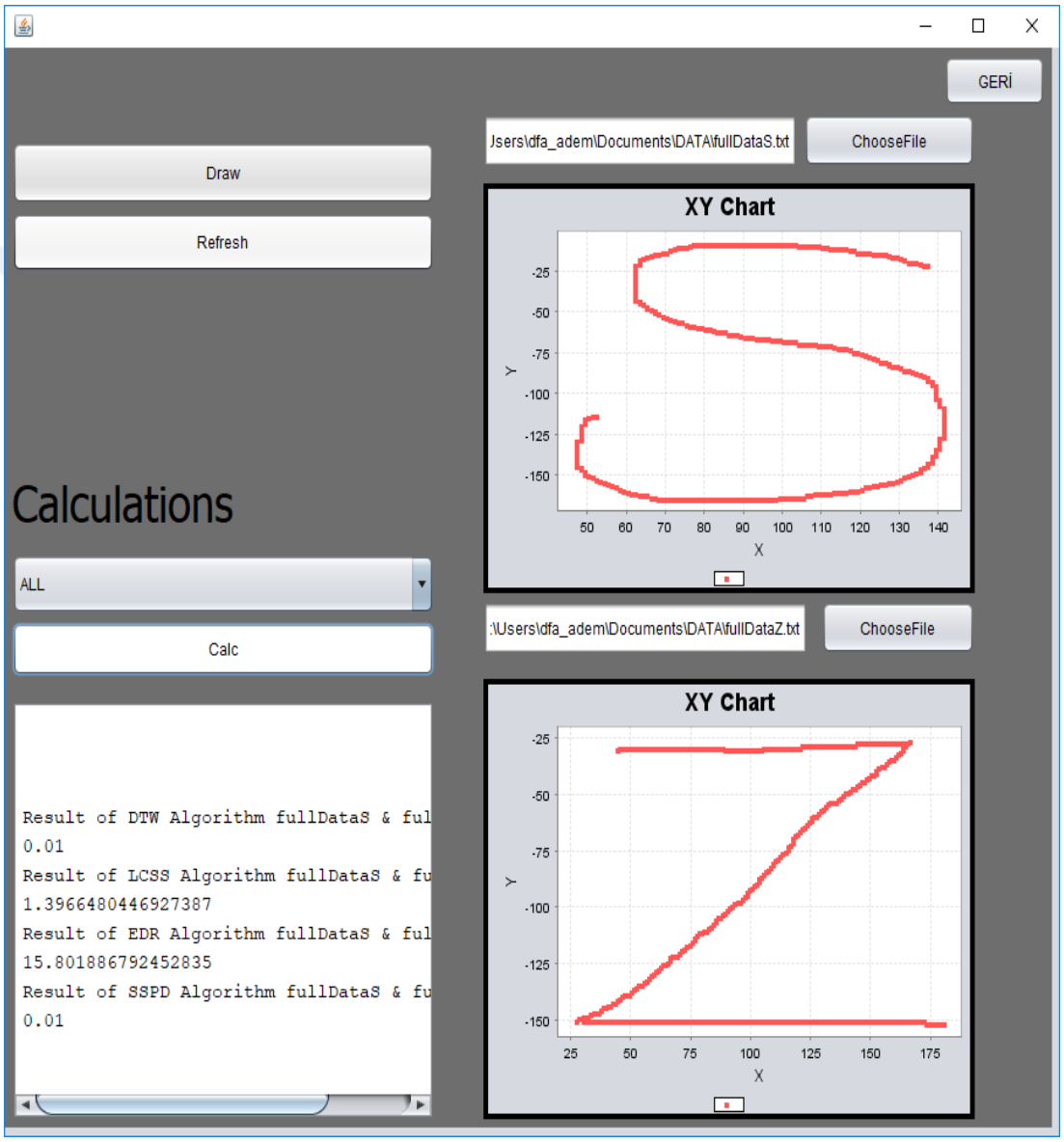


Şekil 2.28. S ve Ş harfleri kullanılarak elde edilen verilerin karşılaştırılması

Bu S ve Ş verilerinden elde edilen sentetik gezinge (trajectory) verilerinin fazla nokta değerine sahip olması çok mantıklı olmadığı için ve zaman kavramını dahil edebilmek için verilerde değer azaltması yaparak sonuçları tekrar analiz etme isteği sonucu yeni ve değer olarak az veriler ile test işlemi yaptık.

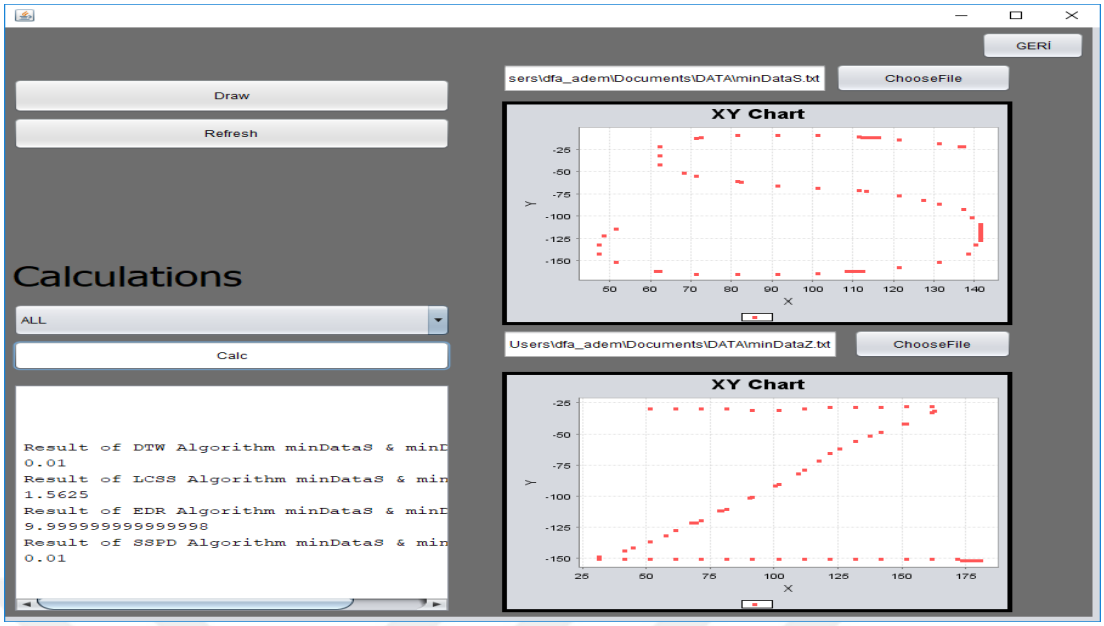
Şekil 2.29. Minimal S ve Ş harfleri kullanılarak iki verinin karşılaştırılması

Bu karşılaştırma sonucu bazı algoritmaların benzerlik değerleri artarken bazıları da aynı kalmıştır. Sentetik veriler üzerinde çalışma yapmaya devam ettik. Mantık olarak belirli bir benzerliği olan ancak gerçekte çok yüksek bir benzerlik değeri olmayan iki karakteri tercih ettik. Bu karakterler S ve Z karakterleridir. Bu karakterlerden elde edilen sentetik verilerin sonuçlarını analiz ettik.



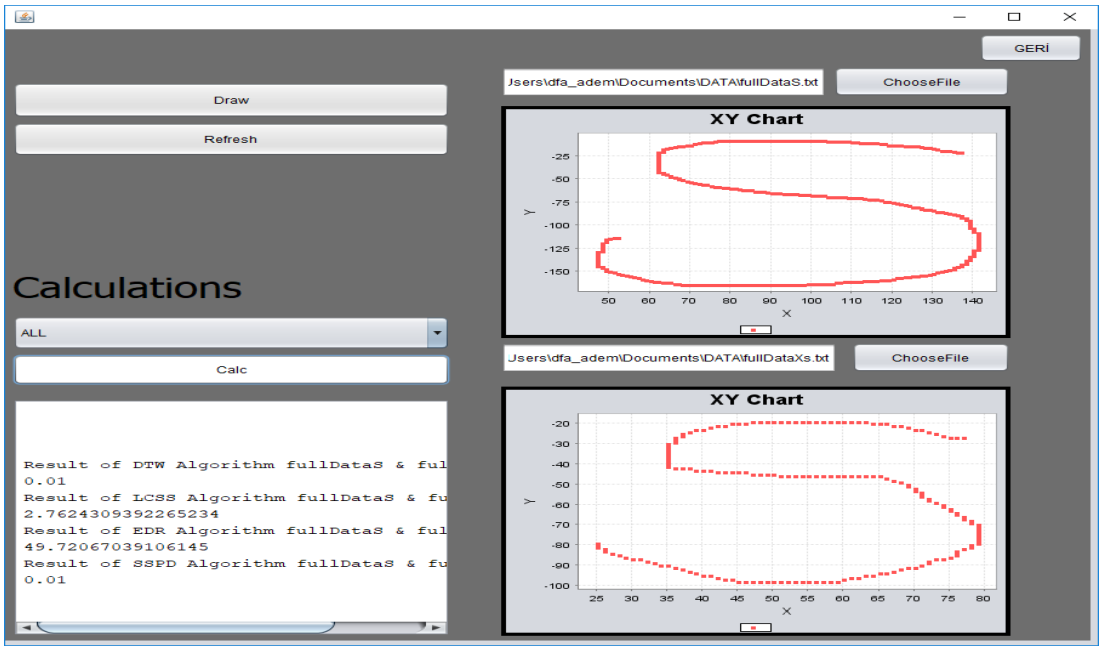
Şekil 2.30. S ve Z harflerinden elde edilen sentetik verilerin karşılaştırılması

Sonuçlar incelendiğinde düşükte olsa bir benzerlik değerinin olduğu gözlemlenmiştir. Bu değeri daha fazla yükseltebilmek için önceden yapmış olduğumuz şekilde veriler üzerinde değer azalması yaptık. Bu azaltma sonucunda elde edilen verilerin hesaplanan sonuçlarını karşılaştırdık.



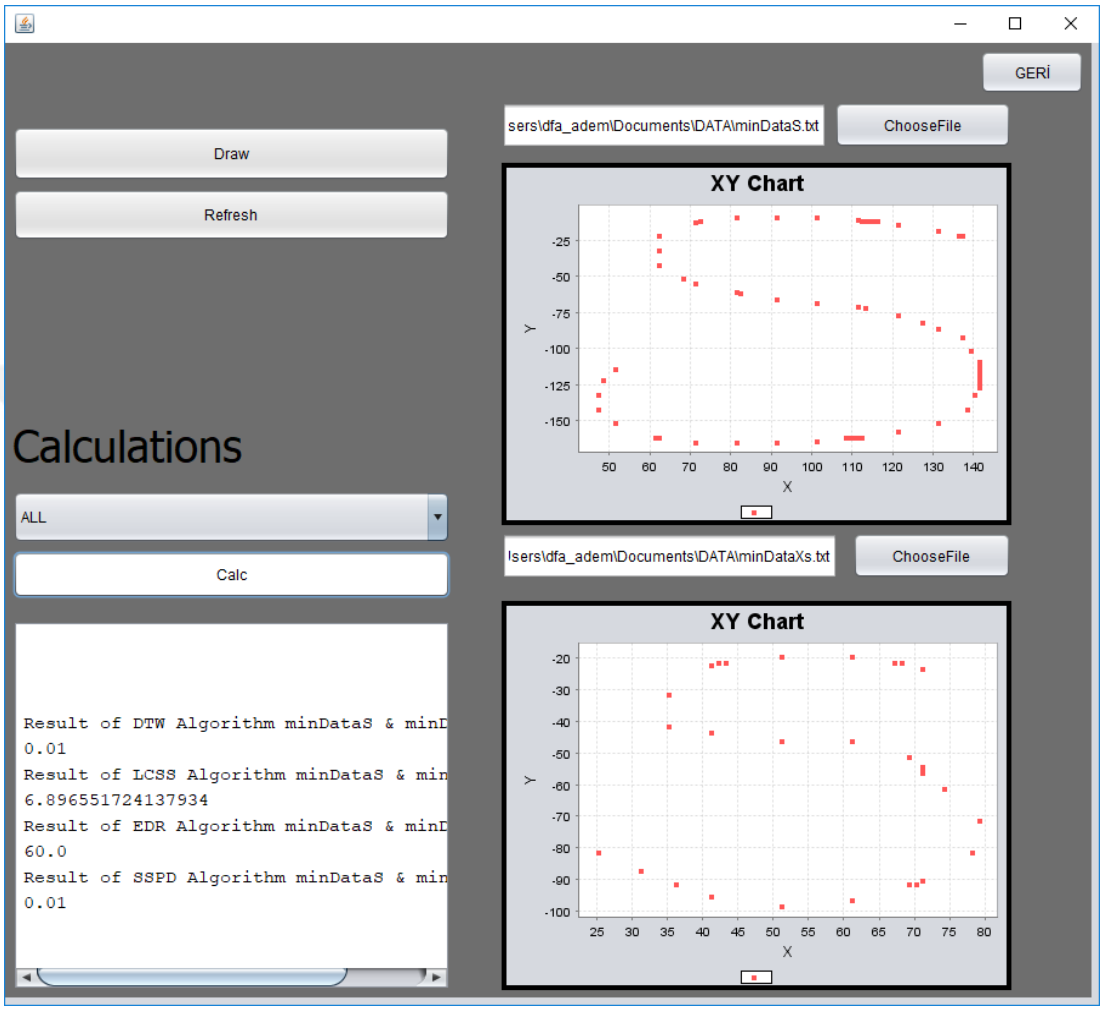
Şekil 2.31. Minimal S ve Z harfleri kullanılarak iki verinin karşılaştırılması

Bu karşılaştırmadan elde edilen sonuçlar incelendiğinde bazı algoritma benzerlik değerle sabit kalırken, bazıları çok az yükselmiş bazıları da çok az düşmüştür. Algoritmaların sınırlarını zorlamaya ve sonuçlarını analiz etmeye devam ediyoruz. Şimdiki sentetik veri üretiminde kullandığımız harfler ise S ve s karakterleridir. Bu iki karakterin gerçekte benzerliği üst düzeydir. Biz bu karakterlerden sentetik veriler üreterek bunların karşılaştırmalarını yaptık.



Şekil 2.32. S ve s harfleri kullanılarak sentetik iki verinin karşılaştırılması

Burada aynı şekilde S ve s karakterlerinden elde edilen gezinge (trajectory) verilerinin minimal halleri kullanılarak benzerlik analizi ve sonuç karşılaştırması yapılmaktadır. Bu karşılaştırmanın görseli ise şu şekildedir.

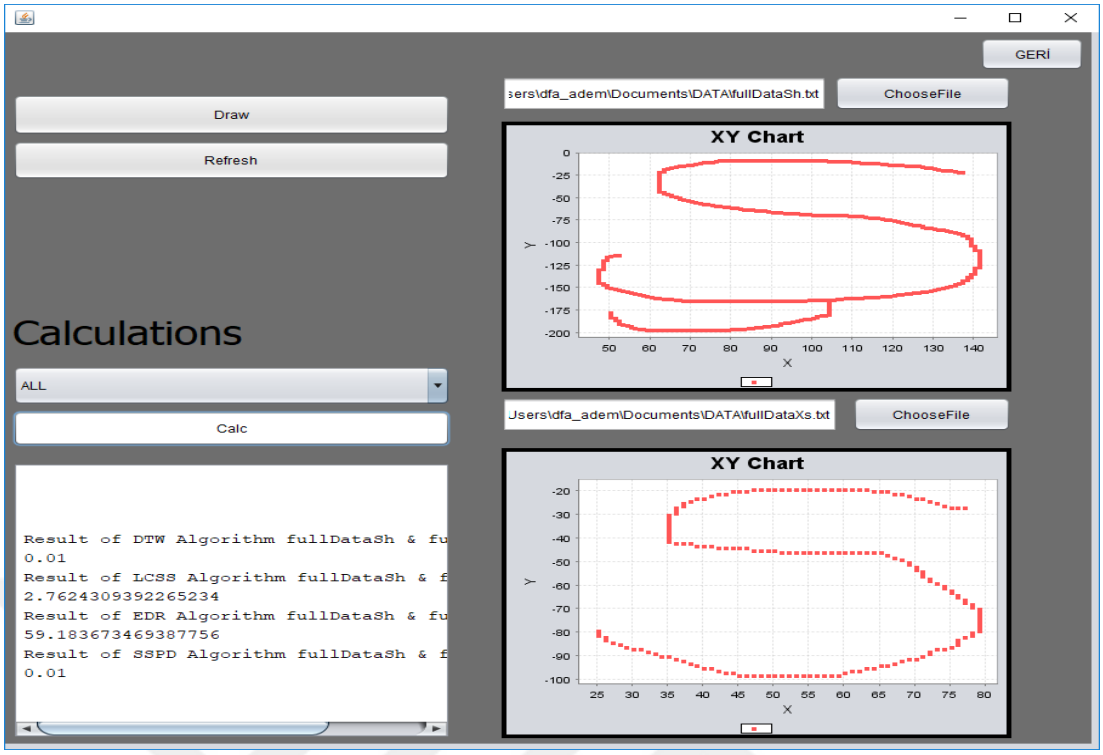


Şekil 2.33. Minimal S ve s harfleri kullanılarak iki verinin karşılaştırılması

Buradan elde edilen sonuçlar incelendiğinde, LCSS ve EDR değerlerinde artış olduğu gözlenmiştir. Bu da beklenen bir sonuçtur.

Bu karşılaştırmadan elde edilen sonuçlar incelendiğinde bazı algoritma benzerlik değerle sabit kalırken, bazıları çok az yükselmiş bazıları da çok az düşmüştür. Algoritmaların sınırlarını zorlamaya ve sonuçlarını analiz etmeye devam ediyoruz.

Şimdiki sentetik veri üretiminde kullandığımız harfler ise S ve s karakterleridir. Bu iki karakterin gerçekte benzerliği üst düzeydir. Biz bu karakterlerden sentetik veriler üreterek bunların karşılaştırmalarını yaptık.



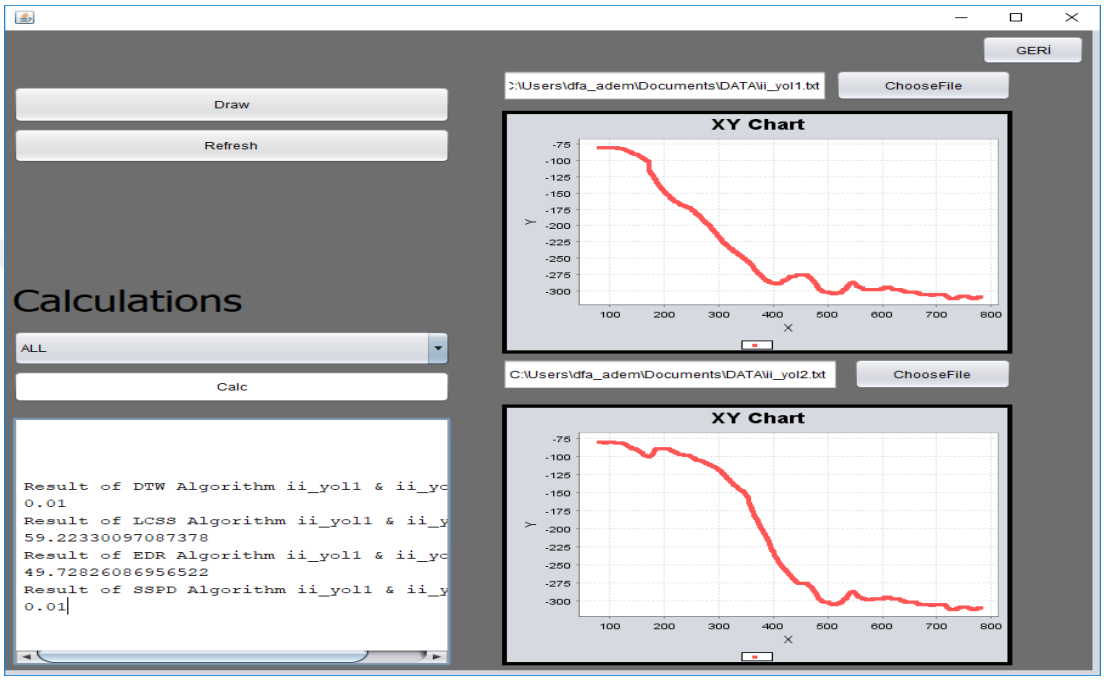
Şekil 2.34. S ve s harfleri kullanılarak sentetik iki verinin karşılaştırılması

Burada aynı şekilde Ş ve s karakterlerinden elde edilen gezinge (trajectory) verilerinin minimal halleri kullanılarak benzerlik analizi ve sonuç karşılaştırması yapılmaktadır. Bu karşılaştırmanın sonucu ise Şekil 2.35.'deki gibidir.

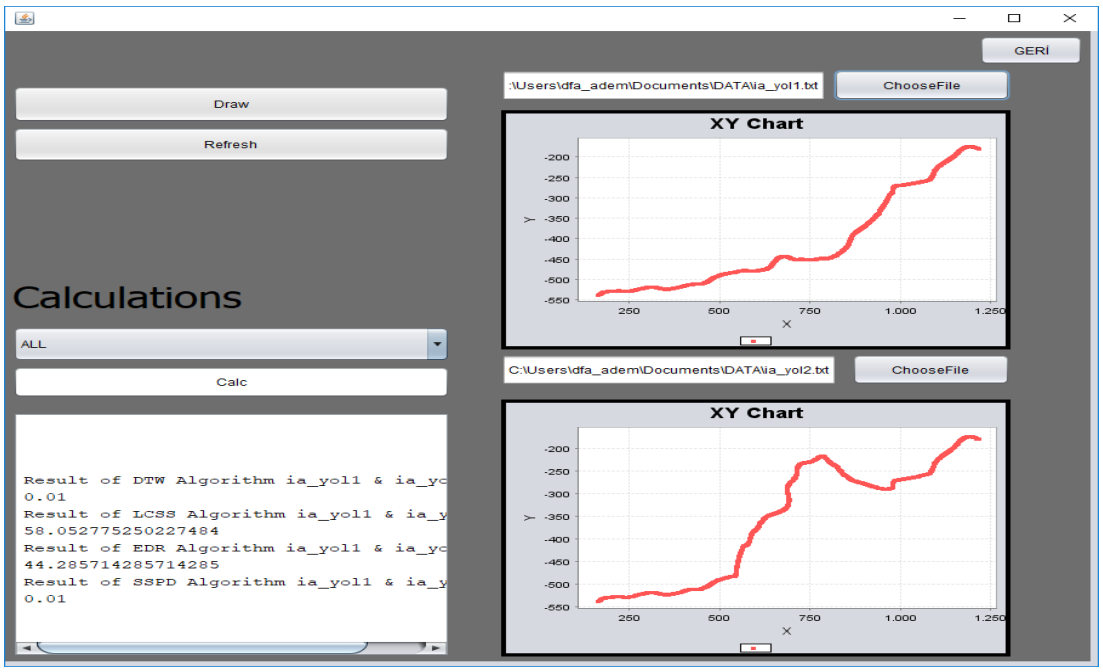
Şekil 2.35. Minimal Ş ve s harfleri kullanılarak iki verinin karşılaştırılması

Buradan elde edilen sonuçlar incelendiğinde, LCSS ve EDR değerlerinde artış olduğu gözlenmiştir. Bu da beklenen bir sonuçtur.

Harita kullanılarak elde edilen şehirlerarası yolların benzerliğinde ise iki farklı karşılaştırma yapılmıştır. Bu karşılaştırma için İstanbul-İzmit arası ve Ankara-İzmir arası yollar belirlenip benzerlik analizi yapılmıştır.

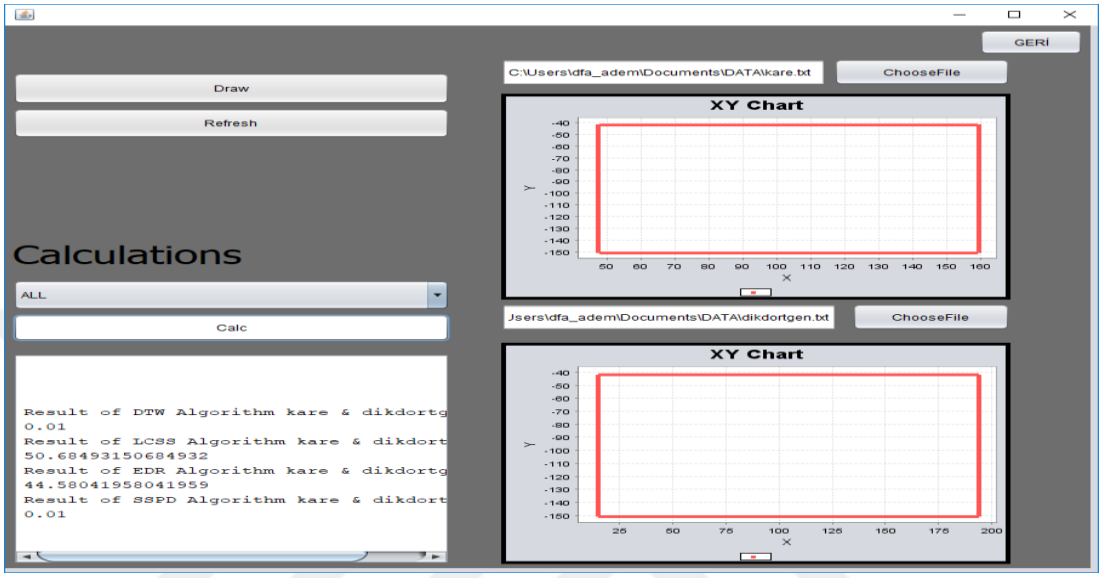


Şekil 2.36. İzmit-İstanbul arası iki yolun benzerlik analizi



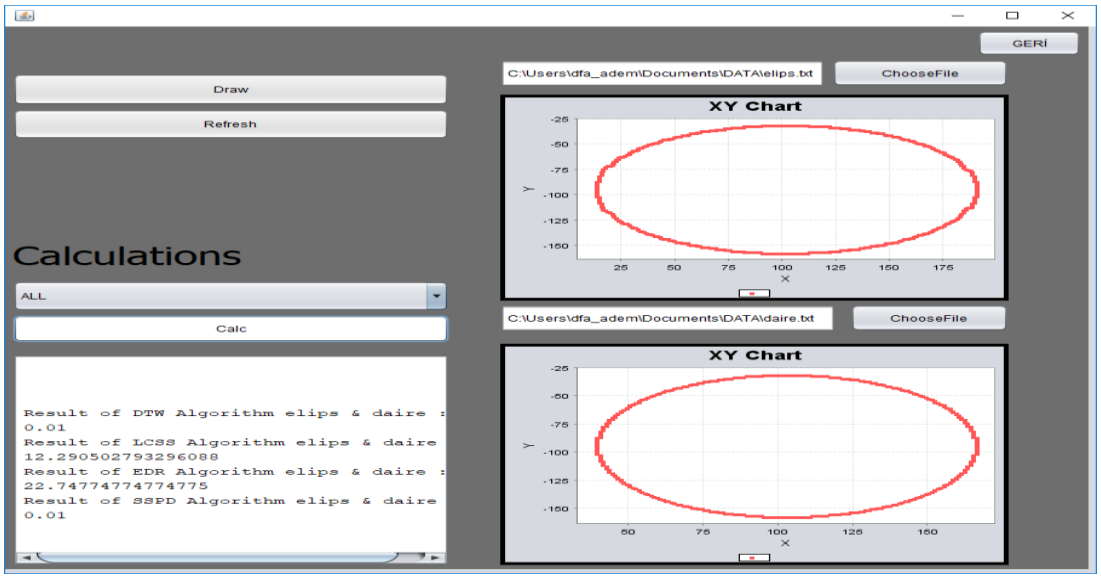
Şekil 2.37. İzmir-Ankara arası iki yolun benzerlik analizi

Geometrik şekil kullanılarak elde edilen gezinge (trajectory) verilerinin benzerliğinde ise üç farklı karşılaştırma yapılmıştır. Bu karşılaştırma için Kare-Dikdörtgen, Daire-Elips ve Eşkenar Üçgen-İkizkenar Üçgen arasında benzerlik analizi yapılmıştır.



Şekil 2.38. Kare ve dikdötrgen arası benzerlik analizi

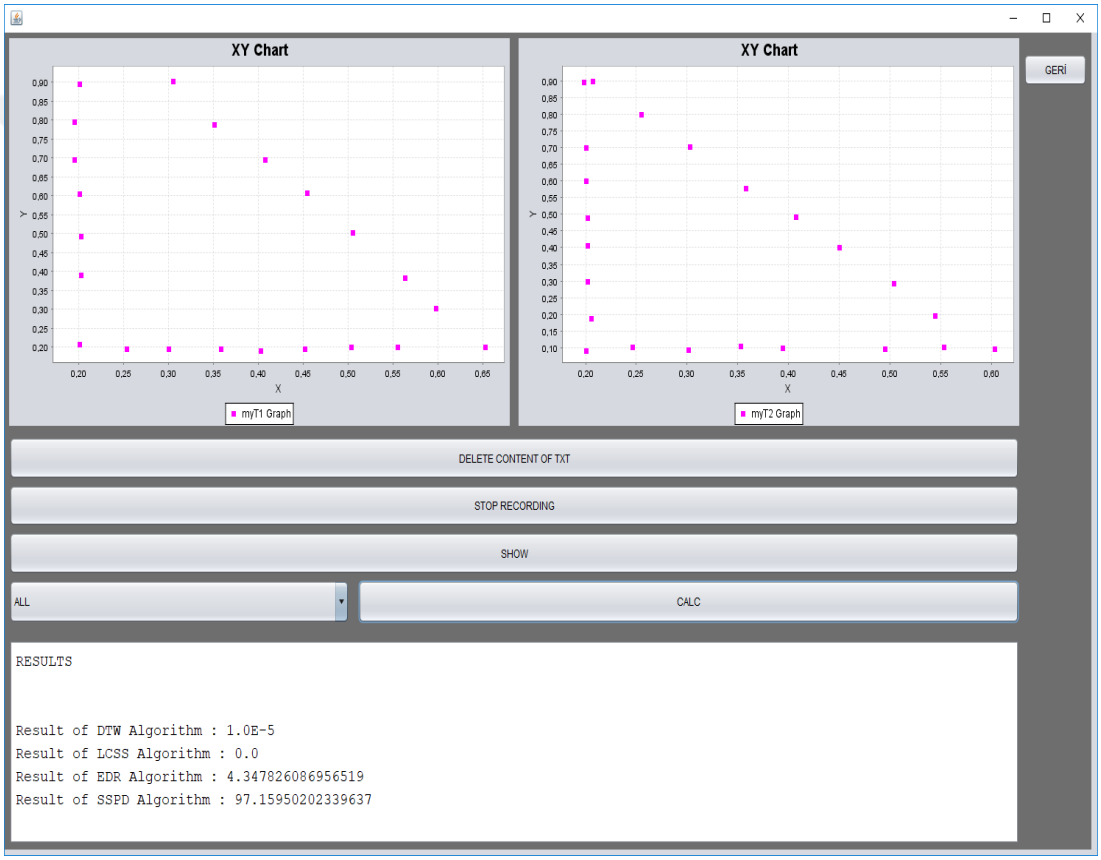
Şekil incelendiğinde iki şeklin dikdörtgen ve aynı olduğu düşünülebilir. Ancak üstteki şekil 110 birim kenar uzunluğuna sahip bir karedir. Alttaki şekil ise üstteki şekilden elde edilen 110 birim yüksekliğe ve 175 birim genişliğe sahip bir dikdörtgendir. Diğer bir benzerlik analiz çifti ise Daire ve Elipstir.



Şekil 2.39. Elips ve daire arası benzerlik analizi

2.3.3. Çizimli arayüz ekranı

Kullanıcı giriş ekranından alt kısımda yer alan manuel butonunu seçer ise bu ekran ile karşılaşır. Bu ekranda üst kısım komple iki tane koordinat düzlemine ayrılmıştır. Bu alanlar sayesinde kendi gezinge (trajectory) verimizi kendimiz oluşturabilmekteyiz. Koordinatlar hem çizim hem görselleştirme işlemlerini birlikte yapmamızı sağlar. Orta kısımda ise çizim işlemini sıfırlama, durdurma ve görselleştirme işlemlerini yapabilmemizi sağlayan butonlar mevcuttur.



Şekil 2.40. Uygulama çizimli arayüz ekranı görüntüsü

Bu çizim ve gösterim işlemlerinden sonra hesaplama işlemine geçilir. Bu işlem için altta yer alan kısım kullanılır. Bu bölümde bir adet açılır kapanır seçim alanı bulunmaktadır. Bu alan sayesinde tek bir algoritma LCSS, DTW, EDR veya SSPD seçilebilir ya da tüm algoritmalar birlikte seçilebilir. Seçim işlemi sonrası hesaplama butonuna basılır ve istenilen algoritmaların sonuçları sonuç alanında listelenir. Bu işlem sonrası sağ en üstteki geri butonu sayesinde giriş ekranına dönüş sağlanabilir.

3. İŞ-ZAMAN PLANI

3.1. Proje İş Paketleri ve Tanımlamaları

Uygulama gerçekleştirilmesinde takip edilecek aşamaların neler olduğu Tablo 3.1.' de belirtilmektedir.

Tablo 3.1. Proje iş paketleri ve tanımlamaları

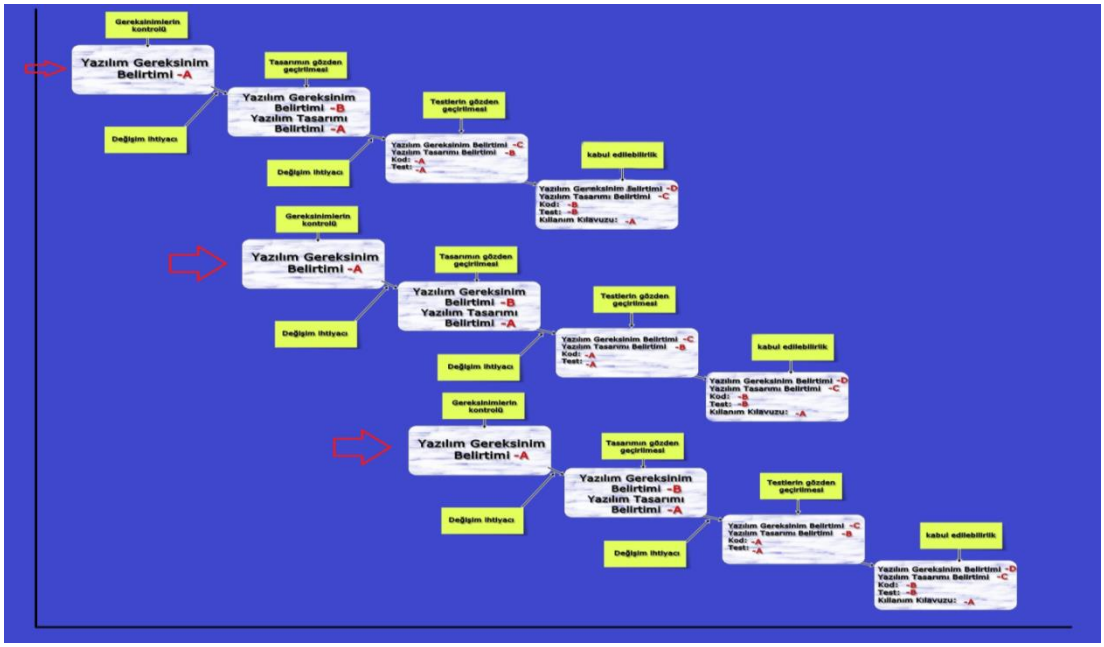
İş Paketi Adı	İş Paketi Tanımı
Uygulama Tanımlanması	Uygulama kapsamı, amacı ve gereksinimlerinin belirlendiği. Önceki sonuçların incelendiği kısım.
Başlangıç Süreci	Projenin başlangıç kapsamının oluşturulduğu, paydaşların belirlendiği, mali kaynakların tahsis edildiği ve bunlara yönelik belgelerin oluşturulduğu projeye hazırlık aşamasıdır.
Tasarım	Projenin tüm fonksiyonlarının tanımlandığı, alt sistem için gereksinimleri ve tüm ihtiyaçların belirlendiği aşamadır.
Uygulama	Uygulama kısmı, kullanıcı ara yüzünün tasarlanması ve gerekli uygulama kısımlarının birleştirilmesi süreçlerini kapsar.
Test	Final ürününün oluşturulup, üzerinde hata ayıklama işlemlerinin ve final testlerinin yapıldığı iş paketidir. Yazılımın son şeklinin kurulumunun bitirilmesi ve kullanım kılavuzuna son halinin verilmesiyle sona erer.
Kapama	Personel aktiviteleri ve proje sürecinin genel bir değerlendirmesi yapılır.

3.2. Proje Yaşam Döngüsü

Bu projede çalışırken kullanılan yaşam döngüsü modeli Artımlı Geliştirme Modelidir. Bu model Çağlayan ve Evrimsel Geliştirme arası bir modeldir. Sistemi tek bir parça olarak en sonda teslim etmektense, sistem, her biri sistemin ayrı bir istenen işlevini yerine getirecek artımlara bölünür.

Kullanıcı gereksinimleri belirlenir ve yüksek öncelikli gereksinimler ilk artımlar arasında gerçekleştirilir. Bu modelde bulunan ana aktiviteler ana hat gereksinimlerini tanımlama, gereksinimleri artımlara atama ve sistem mimarisini tasarlama şeklindedir. Sistem tamamlanıncaya kadar bir döngü halinde yinelenen alt adımlar ise sistem artımları geliştirme, artım geçerleme ve tümleştirme ve sistem geçerleme şeklindedir.

Ana Hat Gereksinimlerini Tanımlama, sistem için gerekli olan gereksinimler müşterilerle belirlenir. Kullanıcı gereksinimleri öncelik belirlemesi yapılır. Gereksinimleri Artımlara Atama, gereksinimlerin önemine göre teslim edilecek artımlar belirlenir. Sistem Mimarisini Tasarlama, öncelikle en önemli gereksinimleri karşılayan çekirdek bir sistem geliştirilir. Kodlama ve test aşaması gerçekleştirilir. Sistem Artımları Geliştirme, bir artımın geliştirilmesine geçilince diğer artımlar için gereksinim gelişimi devam etse bile o artım için olan gereksinimler dondurulur. Artım Geçerleme ve Tümleştirme, erken artımlar prototip gibi davranarak, gereksinimlerin daha iyi anlaşılmasını sağlar. Tüm projenin başarısız olma riskini azaltır. Sistem Geçerleme, yeni artımlarda sisteme eklenerek deneme yapılır. En önemli sistem özellikleri daha fazla sınanma (test edilme) imkanı bulmuş olur. Sistem tamamlanana kadar her artım için işlemler tekrarlanır. Divide and Conquer (Böl ve Yönet) yaklaşımıdır.



Şekil 3.1. Artımlı Geliştirme Modeli görsel anlatımı

3.3. İş Zaman Planı ve Gannt Şeması

Yapılan tez çalışmasının mevcut sürede olabilecek en iyi şekilde tamamlanabilmesi için öncelikle iş-zaman planı yapılmıştır. Çalışma süresi iki dönemden oluşan 2018-2019 eğitim öğretim dönemini kapsamaktadır.

Tablo 3.2. İş zaman planı ayrıntılı gösterimi

GÖREV ADI	SÜRE	BAŞLANGIÇ	BİTİŞ
Tez Çalışma Tanımlamaları	**30 gün**	**20.09.2018**	**20.10.2018**
Çalışma Kapsamının Belirlenmesi	10 gün	20.09.2018	30.09.2018
Çalışma Gereksinimlerinin Belirlenmesi	10 gün	30.09.2018	10.10.2018
Çalışma Konusu ile İlgili Literatür Taraması Yapılması	10 gün	10.10.2018	20.10.2018
Başlangıç Süreci	**21 gün**	**20.10.2018**	**10.11.2018**
Başlangıç Planı Oluşturma	7 gün	20.10.2018	27.10.2018
Gereksinim Temini ve Kurulumlar	7 gün	27.10.2018	03.11.2018
Çalışmada Öncelik Belirleme	7 gün	03.11.2018	10.11.2018
Tasarım	**17 gün**	**10.11.2018**	**25.11.2018**
Giriş Ekranı Tasarımı	2 gün	10.11.2018	11.11.2018
Seçimli Arayüz Ekranı Tasarımı	10 gün	11.11.2018	21.11.2018
Çizimli Arayüz Ekranı Tasarımı	5 gün	21.11.2018	25.11.2018
Uygulama	**104 gün**	**25.11.2018**	**10.03.2019**
Grafikler ile Gösterim	14 gün	25.11.2018	09.12.2018

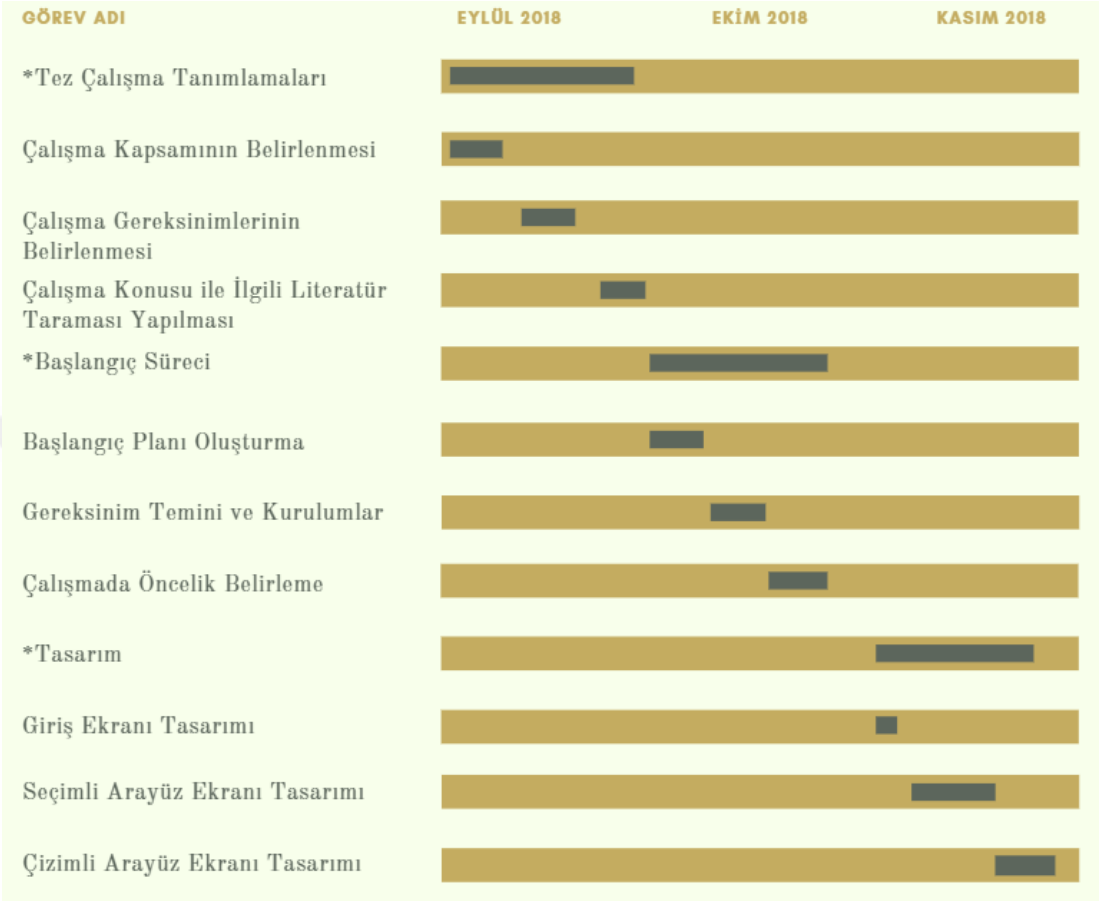
Tablo 3.2. İş zaman planı ayrıntılı gösterimi (Devam)

Koordinatlar ile Çizim	10 gün	09.12.2018	19.12.2018
LCSS Hesaplama	20 gün	19.12.2018	08.01.2019
DTW Hesaplama	20 gün	08.01.2019	28.01.2019
EDR Hesaplama	20 gün	28.01.2019	17.02.2019
SSPD Hesaplama	20 gün	17.02.2019	10.03.2019
Test	**20 gün**	**10.03.2019**	**30.03.2019**
Kullanıcı Testleri	5 gün	10.03.2019	15.03.2019
Sistem İyileştirme	15 gün	16.03.2019	31.03.2019
Kapama	**81 gün**	**31.03.2019**	**21.06.2019**
Sonuçların Benzerlik Analizleri	11 gün	31.03.2019	11.04.2019
İşlemlerin Kayıt Altına Alınması	10 gün	11.04.2019	21.04.2019
Tez Yazılması	60 gün	21.04.2019	21.06.2019

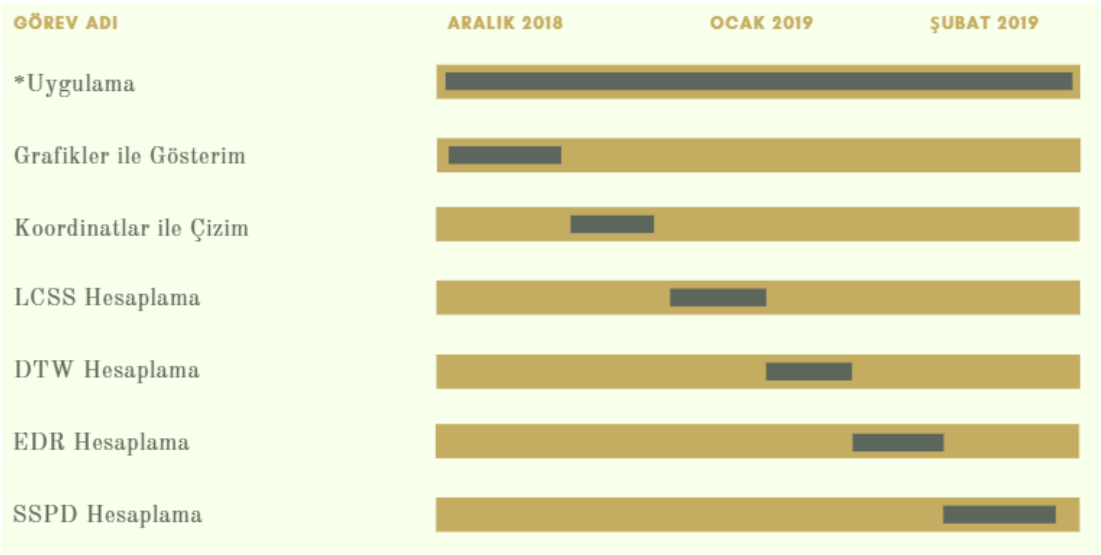
Yukarıdaki şekilde de görülebileceği gibi iş akışımız 6 ana bölümden oluşuyor. Bu bölümlerde projenin geliştirilmesi için uygulama tanımlama, başlangıç süreci, tasarım, uygulama, test ve kapama aşamaları bulunmaktadır. Bu işlemin amacı düzenli bir şekilde çalışmayı sağlayabilmektir.

Öncelikle çalışma konusu belirleyip proje ile ilgili kapsam ve gereksinimlerin belirlenmesinde sonra daha önce üzerinde yapılan çalışmalar ile ilgili literatür taraması yapılıp çalışmaya başlanmıştır. Başlangıç süreci aşamasında ise amacımız belirli bir planda gidebilmek ve sıkıntı olduğunda geri dönüş noktası belirleyebilmektir. Bu aşamada başlangıç planı oluşturma, gereksinimleri temin

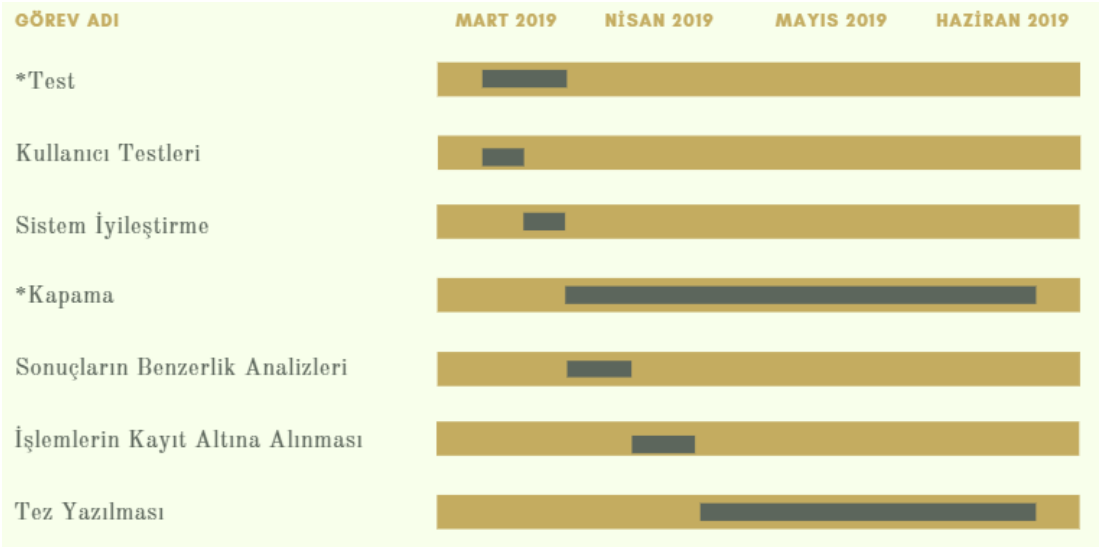
etme, önemli kısımlara öncelik ataması yapılması bulunuyor. Analiz için yapılan Gantt Şeması aşağıda gösterilmiştir.



Şekil 3.2. Gannt şeması çalışma 1. kısım ayrıntılı gösterimi



Şekil 3.3. Gannt şeması çalışma 2. kısım ayrıntılı gösterimi



Şekil 3.4. Gannt şeması çalışma son kısım ayrıntılı gösterimi

Tasarım aşamasına geçildiğinde daha önceki çalışma ile oluşturulan çalışmaların incelenmesi ve yeni çalışma için gerekli olan fonksiyonların belirlenmesi işlemleri yapılmıştır. Sonrasında belirlenen fonksiyonların içerikleri hakkında incelemelerin yapılması işlemleri gerçekleştirilmiştir.

Uygulama aşamasına geçildiğinde ise tasarım aşamasında belirlenen fonksiyonlar için düzenleme ve kodlama işlemleri yapılmıştır. Bu aşama içerisinde benzerlik analizi ve görselleştirme için kullanılacak uygulamanın gerçeklenme işlemi yapılmıştır. İlk olarak gezinge (trajectory) verilerinin görselleştirilmesi ve oluşturulması tamamlanmıştır. Daha sonra ise benzerlik analiz hesaplamasında kullanılan temel uygulama algoritmaları sıra ile geliştirilip uygulamaya entegre edilmiştir.

Test aşamasında uygulama kısımları teker teker denenip sınırları zorlanmıştır. Hata alınan kısımlar üzerinde gerekli iyileştirmeler yapılmaya çalışılmıştır. İyileştirme yapılamayan kısımlar için ise alternatif çözümler bulunmaya çalışılmıştır.

Kapama kısmında yapılan işlemlerin belgelendirilmesi ve çalışma ile ilgili dokümanların hazırlanması işlemleri yapılmıştır. Doküman hazırlanırken tez şablonuna uygun olarak düzenlemeler yapılmıştır. Tez şablonun içerisinde yer alan kısımlar tek tek oluşturulmuştur. Doküman içerisinde istenilen formatlara uygun olarak düzenleme yapılmıştır. İçindekiler kısmı, şekiller kısmı, tablolar kısmı ve kaynakça kısmı açıklandığı gibi düzenlenmiştir.

4. SONUÇLAR VE ÖNERİLER

Bu çalışma ile yapılmak istenen gezinge (trajectory) verilerinin benzerlik analizini ve kullanım alanları açısından değerlendirmesini yapmaktı. Yapılan uygulama sayesinde gerçek, sentetik ve çizim ile oluşturulan veriler üzerinde algoritmalar kullanılarak sonuçlar elde edildi. İlk olarak görsellerde yer alan iki adet gerçek gezinge(tarjectory) verisinin karşılaştırmasını yaptık. Burada kullanılan veriler araç üzerindeki gps yardımı ile elde edilen verilerdir. Bu verilerden ikincisi ilk veri kullanılarak elde edilmiştir. Bu sayede benzerlik oranlarının yüksek olması beklenmektedir.

Gerçek gezinge (trajectory) verisine gürültü eklenip elde edilen yeni gezinge (trajectory) verisinin karşılaştırmasını yaparak sonuçları görüntüledik. Bu şekilde yaparak algoritmaların değişime verdiği tepkiyi ölçmüş olduk.

Kullanılan bir diğer gezinge (trajectory) veri çifti S ve Ş Türkçe karakterleri kullanılarak elde edilen sentetik veriler ile yapılan karşılaştırmadır. Bu tip veriler kullanılmasının sebebi yine benzer olduklarının bilinmesidir. Bu şekilde bazı benzer verilerin karşılaştırmasını yaptık. Örneğin S-Ş, S-Z, S-s ve Ş-s şeklinde verileri kullandık. Daha sonra bu verilerin minimize edilen hallerini kullanıp sonuçları karşılaştırdık.

Son olarak çizim arayüz ekranı üzerinde koordinatlar sayesinde kendi gezinge (trajectory) veri çiftimizi kendimiz elde etmiş oluyoruz. Burada elde edilen verilenin sayısal olarak benzerliği çok düşük seviyede olacağı için algoritmalardan elde edilen sonuçlarında düşük olması beklenmektedir.

Elde edilen sonuçlar incelendiğinde kullanılan tüm veri seti çeşitlerinde EDR-LCSS çifti DTW-SSPD çiftine göre daha iyi sonuçlar vermiştir. DTW-SSPD algoritmaları gerçek gezinge (trajectory) verileri dışında kullanılması çok doğru olmayan algoritmalardır. Verilerde gürültünün daha az olduğu durumlarda DTW ve SSPD algoritmaları kullanılabilir. Diğer durumlarda bu algoritmalar çok iyi sonuçlar vermemektedir

Tablo 4.1.'de elde edilen sonuçların toplu listesini görmektesiniz.

Tablo 4.1. Elde edilen sonuçların karşılaştırılması

Veri Çifti	DTW	LCSS	EDR	SSPD
real_1-real_2	98,4540	99,2993	99,2993	99,9994
real_1-real_3	0,01	99,8999	99,8999	99,8211
S-Ş	0,01	100,00	81,1791	0,01
minS-minŞ	001	100,00	84,3373	0,01
S-Z	0,01	1,3966	15,8019	0,01
minS-minZ	0,01	1,5625	9,9999	0,01
S-s	0,01	2,7624	49,7206	0,01
minS-mins	0,01	6,8966	60,00	0,01
Ş-s	0,01	2,7624	59,1837	0,01
minŞ-mins	0,01	6,8966	66,2650	0,01
manuel_1-manuel-2	1,0E-5	0,00	4,3478	97,1595
İzmir-Ankara	0,01	58,0528	44,2857	0,01
İstanbul-Kocaeli	0,01	59,2233	49,7283	0,01
Kare-Dikdörtgen	0,01	50,6849	44,58041	0,01
Daire-Elips	0,01	0,2445	16,2218	0,01
Eşkenar-İkizkenar	0,01	12,2905	22,7477	0,01

Diğer iki algoritma olan EDR-LCSS algoritmaları genellikle iyi ve birbirine çok uzak olmayan sonuçlar ortaya koymuştur. Tüm verilerde en iyi sonuçlar üreten algoritma EDR algoritmasıdır. Çok fazla gürültülü veri olan durumlarda bile doğru sonuçlar verebilmektedir. LCSS algoritması birbirine benzer değerlerin eşitliği durumunda çok iyi sonuçlar vermektedir. LCSS temelinin karakter, katar benzerliğine dayanıyor olması bunda büyük bir etken olmaktadır. Bu iki algoritmanın sonuçlarını daha ayrıntılı olarak ele almak gereklidir.

DTW ve SSPD gerçek gezinge (trajectory) verilerinde iyi sonuçlar vermektedir. Karakterlerden elde edilen görüntü benzerliklerinde ve geometrik şekillerden elde edilen görüntü benzerliklerinde EDR algoritması daha iyi ve doğru sonuçlar vermektedir. Şehirlerarası rota benzerliklerinde ise LCSS algoritması daha doğru sonuçlar vermektedir.

EDR ve LCSS sonuç analizi, gerçek iki veri seti karşılaştırmasında iki algoritmada da aynı benzerlik analizi sonucu çıkmıştır. Daha sonra S ve Ş karakterleri karşılaştırması yapıldığında LCSS %100 benzerlik analizi sonucu vermiş EDR ise %82 benzerlik sonucu vermiştir. Burada S karakterinden elde edilen veriler Ş karakterinin alt kümesi olduğu için benzerlik sonucunun %100 çıkması beklenen bir sonuçtur. Ancak EDR algoritması sonucu elde edilen benzerlik sonucu olması gereken gerçek değere daha yakın olan sonuçtur. Bu yüzden burada EDR ile bulunan sonuç daha doğrudur.

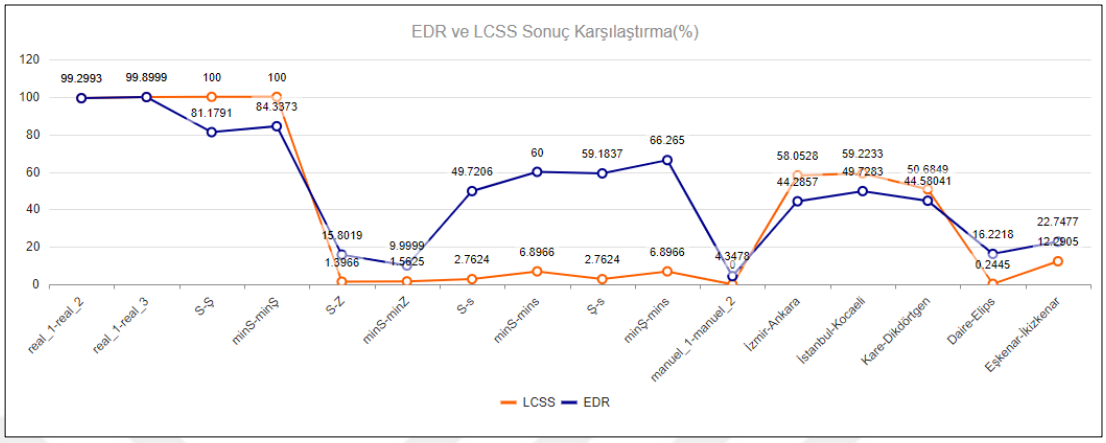
S ve Z karakterleri kullanılarak yapılan analizde ise LCSS %1.35 gibi düşük bir sonuç verirken EDR ise %15 gibi bir sonuç vermektedir. Bu sonuçlara bakıldığında Z-S arasında düşükte olsa bir benzerlik olduğu sonucuna ulaşılır.

Aynı şekilde S-s ve Ş-s arası benzerlik sonuçları incelendiğinde ise yine LCSS bire bir eşleme oranı daha düşük olduğu ve alt kümesi olma durumu bulunmadığı için düşük sonuç üretmiştir. EDR ise %50 dolaylarında sonuç ürettiği için benzerlik olduğunu doğrulamıştır. Burada ki benzerlik oranın daha yüksek beklenmesine rağmen bu değerde çıkması karşılaştırma yapılan iki gezinge (trajectory) verisi arasındaki boyut farkının yüksek olmasıdır. Şöyle ki **S** gezinge (trajectory) verisi 358 ve **Ş** gezinge (trajectory) verisi 441 konum çiftinden oluşurken **s** gezinge (trajectory) verisi 181 konum çiftinden oluşmaktadır. Aradaki yaklaşık 2 katlık farkın sebebi benzerlik oranını %50 dolaylarında olmasına sebep olmaktadır.

İzmir-Ankara ve İzmit-İstanbul şehirleri arasındaki yolların benzerlik analizlerinde ise LCSS'nin %60 dolaylarında değerine karşılık EDR'da %50 civarında sonuçlar elde edilmiştir. Buradan yine birbirine yakın benzerlik sonuçlarının çıkması algoritmaların doğru çalıştığını göstermektedir.

Geometrik şekillerin benzerliğinde Daire ve Elips ele alındığında LCSS 0,25 değerini EDR ise %16 değerini vermektedir. Buradaki oranların bu kadar düşük olmasının sebebi ise ortak nokta azlığıdır. Şöyle ki daireden elde edilen elips ile benzerlikleri

sadece alt ve üst kutup noktalarında bulunmaktadır. Diğer geometrik şekillerin benzerliğinde ise iki algoritma birbirine yakın sonuçlar vermektedir.



Şekil 4.1. LCSS ve EDR benzerlik sonuçları grafiği

Çalışma sayesinde elde edilenler;

- Gezinge (trajectory) verisinin yapısı ve görselliği tam olarak anlaşılabilir hale geldi.
- Analizler ve kullanım alanları daha anlaşılabilir hale geldi.
- Bahsedilen algoritmaların yapıları tam anlamıyla anlaşıldı ve kullanım amaçları daha kavranabilir hale geldi.
- Görselleştirme ve anlık veri elde edilmesi ile gezinge (trajectory) verisi kavramı yabancılıktan çıkarılıp bilinen bir hale geldi.
- Algoritmaların sonuçları karşılaştırıldı.

Gibi sonuçlar elde edilmiştir. Ancak bundan sonra bu veri ve algoritmaların günlük hayatta daha fazla kullanılabilmesi için çalışma yapılabilir. Bu çalışmaya daha fazla özellik ekleyerek daha büyük çapta kitleye ulaşmak amaçlanabilir.

Kullanım alanları açısından değerlendirdiğimizde, Gezinge (trajectory) verileri birçok alanda benzerlik analizi için kullanılabilir. Bu çalışmada ise taşıt yollarının benzerliğinin analizinde, yazı karakterlerinin benzerlik analizinde, koordinat düzlemine çizilen şekillerin benzerlik analizinde, şehirlerarası yolların benzerlik analizinde ve geometrik şekillerin benzerlik analizinde kullanılmıştır. EDR bu analizlerde LCSS ye göre daha iyi ve daha gerçek sonuçlar ortaya koymuştur. İki

algoritmada gerçek gezinge (trajectory) verilerinde %100'e yakın sonuçlar verirken diğer analizlerde ise daha makul sonuçlar vermektedir.

Devam eden kısımda algoritmaların sözde kodları yer alacaktır. Bu kısımda tüm kaynak kodlarına yer verilmeyecektir. Burada benzerlik analizi ve karşılaştırma için kullanmış olduğumuz algoritmalara ve sözde kod içeriklerine adım adım yer vereceğiz.

Tablo 4.2. Uygulamanın çalıştırılması için gereken adımların sözde kodu

Gezinge (trajectory) Benzerlik Değeri Analizi
1. Uygulama başlatma 2. Giriş ekranından auto veya manuel seçimi yapma 3. Eğer auto butonu seçilmiş ise 4. Birinci gezinge (trajectory) verisini seçme 5. Ardından ikinci gezinge (trajectory) verisini seçme 6. Daha sonra Çizim butonunu kullanarak görselleştirme yapma 7. Görselleştirme sonrası istenilen algoritma seçimini seçim alanından yapma 8. Seçim yapıldıktan sonra Hesaplama butonunu kullanarak sonuçları hesaplatma 9. Hesaplanan sonuçlarını görüntülenme alanından incelenmesi 10. Eğer manuel butonu seçilmek isteniyor ise 11. Geri butonu yardımı ile giriş ekranına geri dönme 12. Giriş ekranından manuel seçimi yapma 13. Çizim ekranından ilk koordinat alanından birinci verinin noktalarını belirleme 14. Daha sonra ikinci koordinat alanından diğer verinin noktalarını belirleme 15. Göster butonu ile seçilen noktaların ekranda görünür hale gelmesini sağlama

Tablo 4.3. Öklit mesafe hesaplama işlemi

Öklit Mesafe Hesaplama **Girdi:** A(x1,x2) noktası ve B(y1,y2) noktası
1. İki nokta X koordinat değerlerinin farkını al (x1-x2) 2. Koordinat farkının karesini al 3. İki nokta Y koordinat değerlerinin farkını al (y1-y2) 4. Koordinat farkının karesini al 5. X koordinat farkını Y koordinat farkından çıkar 6.Çıkan değerin karekökünü al 7.Bulunan değeri sonuç olarak ata 8.Sonucu geri döndür 9. Son

Tablo 4.4. Hesaplamada kullanılan LCSS algoritmasının sözde kodu

LCSS Algoritması **Girdi:** t0:ilk gezinge (trajectory) verisi, t1:ikinci gezinge(trajetory) verisi ve epsilon=0.00001 değeri
1. no: birinci gezinge (trajectory) verisinin boyutunu hesapla 2. n1: ikinci gezinge (trajectory) verisinin boyutunu hesapla 3. Boyutu [n1+1,no+1] olan C matrisini 0 değerleri ile doldur 4. Değeri i=1 den j=n0+1 kadar olan döngü başlat 5. Değeri j=1 den j=n1+1 kadar olan döngü başlat 6. Eğer to[i-1] ile t1[j-1] arasındaki öklit mesafesi küçüktür epsilon ise 7. C[i][j] değerine C[i-1][j-1]+1 değerini ata 8. to[i-1] ile t1[j-1] arasındaki öklit mesafesi küçüktür epsilon değil ise 9. C[i][j] değerine (C[i][j-1, C[i-1][j]) çiftinden büyük olanı ata 11. İçteki döngü sonu 12. Dıştaki döngü sonu 13. C matrisinin C[no][n1] değerini (n0,n1) çiftinden küçük olana böl 14. Bölme sonucunu float yap 15. Elde edilen sonucu 1 değerinden çıkar 16. Elde edilen bu sonucu LCSS değişkenine ata 17. Geri dönen LCSS değerinden % hesabını yap 18. Benzerlik analiz değerini sonuç olarak geriye döndür 19.Son

Tablo 4.5. Hesaplamada kullanılan DTW algoritmasının sözde kodu

DTW Algoritması **Girdi:** t0:ilk gezinge (trajectory) verisi, t1:ikinci gezinge(trajetory) verisi
1. no: birinci gezinge (trajectory) verisinin boyutunu hesapla 2. n1: ikinci gezinge (trajectory) verisinin boyutunu hesapla 3. Boyutu [n1+1,no+1] olan C matrisini 0 değerleri ile doldur 4. C matrisinin 1 satır elemanlarına ilk eleman hariç “inf” sonsuz değerini ata 5. C matrisinin 1 sütün elemanlarına ilk eleman hariç “inf” sonsuz değerini ata 6. Değeri i=1 den j=n0+1 kadar olan döngü başlat 7. Değeri j=1 den j=n1+1 kadar olan döngü başlat 8. to[i-1] ve t1[j-1] noktaları arasındaki öklit mesafesini hesapla 9. C[i][j-1], C[i-1][j-1] ve C[i-1][j] değerlerinin en küçüğünü hesapla 10. Öklit mesafesini ve hesaplanan en küçük değerin toplamını bul 11. Bulunan toplam değerini C[i][j] değerine ata 12. İçteki döngü sonu 13. Dıştaki döngü sonu 14 Elde edilen C matrisinin C[n0][n1] değerini DTW değişkenine ata 15. DTW değerinin % hesabını yap geri döndür ve son

Tablo 4.6. Hesaplamada kullanılan EDR algoritmasının sözde kodu

EDR Algoritması **Girdi:** t0:ilk gezinge (trajectory) verisi, t1:ikinci gezinge(trajetory) verisi ve epsilon=0.0001 değeri
1. no: birinci gezinge (trajectory) verisinin boyutunu hesapla 2. n1: ikinci gezinge (trajectory) verisinin boyutunu hesapla 3. Boyutu [n1+1,no+1] olan C matrisini 0 değerleri ile doldur 4. Değeri i=1 den j=n0+1 kadar olan döngü başlat 5. Değeri j=1 den j=n1+1 kadar olan döngü başlat 6. Eğer to[i-1] ile t1[j-1] arasındaki öklit mesafesi küçüktür epsilon ise 7. Maliyet değerine 0 ata 8. to[i-1] ile t1[j-1] arasındaki öklit mesafesi küçüktür epsilon değil ise 9. Maliyet değerine 1 ata 11. C[i][j-1]+1, C[i-1][j]+1 ve C[i-1][j-1]+maliyet değerlerinin en küçüğünü bul 12. Bulunan değeri C[i][j] matris elemanına ata 13. İçteki döngünün sonu 14. Dıştaki döngünün sonu 15. C[n1][n1] bölü en büyük (n0,n1) değerini hesapla 16. Bölme sonucunu float yap 17. Elde edilen bu sonucu EDR değişkenine ata 17. Geri dönen EDR değerinden % hesabını yap 18. Bulunan sonucu benzerlik analiz değeri olarak ata 19.Son

Tablo 4.7. Hesaplamada kullanılan SSPD algoritmasının sözde kodu

SSPD Algoritması **Girdi:** t0:ilk gezinge (trajectory) verisi, t1:ikinci gezinge(trajetory) verisi
1. SPD(t0,t1) değerini hesapla 2. SPD(t1,t0) değerini hesapla 3. Hesaplanan değerleri topla 4. Toplam sonucunu ikiye böl 5. Çıkan sonucu SSPD değişkenine ata 6. SSPD değerini geriye döndür 7. SSPD değerinin maksimum 1 değeri üzerinden % hesabını yap 8. Bulunan % değerini geriye sonuç olarak döndür 9. Son

Tablo 4.8. SSPD hesaplamada kullanılan SPD algoritmasının sözde kodu

SPD Algoritması **Girdi:** t0:ilk gezinge (trajectory) verisi, t1:ikinci gezinge(trajetory) verisi
1. Toplam değişkenine 0 değeri ata 2. İlk parametre olan t0 verisinin değerlerini p noktaları olarak ele al 3. Her p noktasını ve t1 verisini kullanarak point_to_trajectory(p,t1) değerini hesapla 4. Hesaplana bu değerleri toplam değişkenine sıra ile ekle 5. Bu işlem sonucunda bulunan toplam değerini t0 verisini eleman sayısına böl 6. Çıkan sonucu SPD değişkenine ata 7. SPD değişkenini geri döndür 8. Son

Tablo 4.9. SPD hesaplama için point_to_trajectory metodunun sözde kodu

Point_to_trajectory Metodu **Girdi:** p:ilk gezinge (trajectory) verisinin noktaları, t:ikinci gezinge(trajetory) verisi
1. t[:-1] değerlerini S1 segmentine ata 2. t[1:] değerlerini S2 segmentine ata 3. Her p noktasını ile S1 ve S2 segmentlerini kullanarak point_to_segment(p,S1,S2) değerini hesapla 4. Hesaplanan bu değerlerin en küçüğünü bul 5. Bulunan değeri DPT değişkenine ata 6. DPT değişkenini geri döndür 7. Son

Tablo 4.10. Point_to_segment metodunun sözde kodu

Point_to_segment Metodu **Girdi:** p:ilk gezinge (trajectory) verisinin noktaları, gezinge(trajetory) verisinden elde edilen ilk bölüm S1 ve ikinci bölüm S2 segmentleri ve epsilon=0.00001
1. px değişkenine p noktasının X koordinatını ata 2. py değişkenine p noktasının Y koordinatını ata 3. p1x değişkenine S1 segmentinin birinci değerini ata 4. p1y değişkenine S1 segmentinin ikinci değerini ata 5. p2x değişkenine S2 segmentinin birinci değerini ata

Tablo 4.10. Point_to_segment metodunun sözde kodu (Devam)

6. p2y değişkenine S2 segmentinin ikinci değerini ata 7. Eğer p1x eşit p2x ve p1y eşit p2y ise 8. DPL değişkenine p ve S1 noktasının öklit mesafe değerini ata ve işlemi sonlandır 9. p1x eşit p2x ve p1y eşit p2y değil ise 10. Segment değişkenine p ve S1 noktasının öklit mesafe değerini ata 11. U1 değişkenine (((px-p1x)*(p2x-p1x))+((py-p1y)*(p2y-p1y))) sonucunu ata 12. U1 değerinin segment değerinin karesine bölümünü hesapla 13. Bu hesaplama sonucunu U değişkenine ata 14. Eğer U değişkeni küçüktür epsilon veya büyüktür 1 ise 15. ix değerine p ile S1'in öklit mesafesini ata 16. iy değerine p ile S2'in öklit mesafesini ata 17. Eğer ix büyüktür iy ise 18. DPL(Dynamic Path Length) değişkenine iy değerini ata ve işlemi sonlandır 19. ix büyük değil iy ise 20. DPL değişkenine ix değerini ata 21. U değişkeni küçüktür epsilon veya büyüktür 1 değil ise 22. ix değişkenine p1x+U*(p2x-p1x) değerini ata 23. iy değişkenine p1y+U*(p2y-p1y) değerini ata 24. p noktalarının hesaplanan [ix,iy] değerlerine olan öklit mesafesini bul 25. Bulunan bu değeri DPL değişkenine ata 26. DPL değişkenini geri döndür 27. Son

KAYNAKLAR

[1] Besse P., Guillouet B., Loubes J.-M., Francois R., Review and Perspective for Distance Based Trajectory Clustering, 2015.

[2] Lin B. ve Su J., Shapes Based Trajectory Queries for Moving Objects, *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, ACM, Bremen, Germany, 2005, DOI: 10.1145/1097064.1097069.

[3] Berndt D. J., Clifford J., Using Dynamic Time Warping to Find Patterns in Time Series, Seattle, *WA*, 1994.

[4] Vlachos M., Kollios G., Gunopulos D., Discovering Similar Multi-Dimensional Trajectories, *18th International Conference on IEEE*, San Jose, CA, USA, 2002, DOI: 10.1109/ICDE.2002.994784.

[5] Lee J.-G., Han J., Whang K.-Y., Trajectory Clustering: a Partition and Group Framework, *ACM*, 2007.

[6] Şeker S. E., En Uzun Ortak Küme (Longest Common Subsequence, LCS), http://bilgisayarkavramlari.sadievrenseker.com/2007/12/04/en-uzun-ortak-kume-longest-common-subsequence-lcs/ (Ziyaret Tarihi: 21 Haziran 2019).

[7] Çınar A. C., Öklit uzaklığı nasıl hesaplanır?, https://ahmetcevahircinar.com.tr/2017/04/17/oklit-uzakligi-nasil-hesaplanir/, (Ziyaret Tarihi: 21 Haziran 2019).

[8] Dinamik Zaman Eğrilerine Giriş, https://riptutorial.com/algorithm/example/24981/introduction-to-dynamic-time-warping (Ziyaret Tarihi: 21 Haziran 2019).

[9] Düzenleme Mesafesi (Edit Distance), http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvRWRpdF9kaXN0YW5jZQ (Ziyaret Tarihi: 21 Haziran 2019)

[10] Chen L., Ozsu M.T., Oria V., Robust and Fast Similarity Search for Moving Object Trajectories, Baltimore, Maryland, 2005, DOI: 10.1145/1066157.1066213.

[11] Wang H., Su H., Zheng K., Sadiq S., Zhou X., An Effectiveness Study on Trajectory Similarity Measures, Adelaide, Australia, 2013.

[12] Oktay Y.,Netbeans IDE'ye Dair Bilgiler, https://webmaster.kitchen/netbeans-ideye-dair-bilgiler (Ziyaret Tarihi: 21 Haziran 2019).

[13] Sağıroğlu Ş., Beşdok E., Özkaya N., Genel Amaçlı Otomatik Parmakizi Tanıma Sistemi Tasarımı ve Gerçekleştirilmesi, *Politeknik Dergisi*, 8 (3), 239-247, 2005.

[14] Sehgal P., Bedi P., Bansal R., Minutiae Extraction from Fingerprint Images - a Review, *IJCSI International Journal of Computer Science Issues*, 8 (3), 74-86, 2011.

[15] Paindavoine M., Kusuma T. M., Sudiro S. A., Simple Fingerprint Minutiae Extraction Algorithm Using Crossing Number On Valley Structure, *IEEE Workshop on Automatic Identification Advanced Technologies*, 41-44, 2007, DOI: 10.1109/AUTOID.2007.380590.

[16] Python ile Görüntü İşleme - PIL Nedir ?, http://sevimlikodparcaciklari.blogspot.com/2017/07/python-ile-goruntu-isleme-pil-nedir.html (Ziyaret Tarihi: 21 Haziran 2019).

[17] Pişkin M., OpenCV Nedir?, http://mesutpiskin.com/blog/opencv-nedir.html (Ziyaret Tarihi: 21 Haziran 2019).

[18] matplotlib 1: Temel Grafikler, https://pybilim.wordpress.com/2014/01/01/matplotlib-1-temel-grafikler/ (Ziyaret Tarihi: 21 Haziran 2019).

[19] Scikit-Image, http://www.wikizero.biz/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvU2Npa2l0LWltYWdl (Ziyaret Tarihi: 21 Haziran 2019).

[20] Karmaşık Ağlar İçin Yazılım, https://networkx.github.io/ (Ziyaret Tarihi: 21 Haziran 2019).

KİŞİSEL YAYIN VE ESERLER

[1] **Ulu A.**, Sayar A., Çelikhası C., *Graf Eşleme Algoritmalarından FastPFP'nin İncelenmesi*, International Conference on Data Science and Applications (ICONDATA), Yalova Üniversitesi, Yalova, 4-7 Ekim 2018.

[2] Çelikhası C., Sayar A., **Ulu A.**, *DNA Dizilerinin Graf Benzetim Yolu ile Karşılaştırılması*, International Conference on Data Science and Applications (ICONDATA), Yalova Üniversitesi, Yalova, 4-7 Ekim 2018.

ÖZGEÇMİŞ

1994 yılında Manisa ilinin Gördes ilçesinde doğdu. İlk ve orta eğitimini Gördes Cumhuriyet İlköğretim Okulu'nda tamamladı. Lise eğitimini ise Gördes Anadolu Lisesi'nin Fen Bilimleri bölümünde tamamladı. 2012 yılında girdiği lisans yerleştirme sınavında aldığı puanla Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliğine yerleşti. Bir yıl süreyle Kocaeli Üniversitesi Yabancı Diller Yüksekokulu'nda İngilizce hazırlık eğitimi aldı. 08.06.2017 tarihinde Kocaeli Üniversitesi Bilgisayar Mühendisliği'nden 3.17 ortalama ile mezun oldu. Şuan Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği yüksek lisans öğrencisi olup tez aşamasında bulunmaktadır. Aynı zamanda Kocaeli Üniversitesi Teknopark bünyesindeki Yeninova Yazılım Stüdyosu Limited Şirketi'nde Yazılım ve Veri Tabanı Uzmanı olarak çalışmaktadır.