KTO KARATAY
ÜNİVERSİTESİ

T.C.
KTO Karatay Üniversitesi
Fen Bilimleri Enstitüsü

ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
TEZLİ YÜKSEK LİSANS PROGRAMI

KOMPLEKS DIŞ ORTAMLAR İÇİN SEZGİSEL GÜZERGAH PLANLAMA ALGORİTMALARI

İbrahim YAYLALI

KONYA

Eylül 2018

Dr. Öğr. Üyesi H. Oktay Altun H. Oktay Altun
Dr. Öğr. Üyesi Rahime CEYLAN
Dr. Öğr. Üyesi Semih Yumaşak

KOMPLEKS DIŞ ORTAMLAR İÇİN SEZGİSEL GÜZERGAH PLANLAMA ALGORİTMALARI

İbrahim YAYLALI

KTO Karatay Üniversitesi Fen Bilimleri Enstitüsü

Elektrik ve Bilgisayar Mühendisliği Ana Bilim Dalı
Yüksek Lisans Programı

Yüksek Lisans Tezi

Eylül, 2018

H.2.4

Fen Bilimleri Enstitü Onayı

Fen Bilimleri Enstitüsü Müdür Vekili

Prof. Dr. H. Bekir Yıldız

Bu yüksek lisans tezinin yapılması gereken bütün gerekliliklerinin yerine getirdiğini onaylıyorum.

Anabilim Dalı Başkanı

Dr. Öğr. Üyesi H. Oktay Altun

İbrahim YAYLALI tarafından hazırlanan KOMPLEKS DIŞ ORTAMLAR İÇİN SEZGİSEL GÜZERGAH PLANLAMA ALGORİTMALARI başlıklı bu çalışma 17.08.2018 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jüri tarafından tezli yüksek lisans tezi olarak kabul edilmiştir.

Tez Danışmanı

Dr. Öğr. Üyesi H. Oktay Altun

Tez Yardımcı Danışmanı

Dr. Öğr. Üyesi Amir Yavariabdi

Jüri Üyeleri

Başkan : Dr. Öğr. Üyesi Rahime Ceylan

Üye : Dr. Öğr. Üyesi H. Oktay Altun

Üye : Dr. Öğr. Üyesi Semih Yumuşak

Tez Bildirimi

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orjinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu tezde sunduğum çalışmanın özgün olduğunu bildirir aksi bir durumda aleyhime doğabilecek tüm hak ve kayıplarını kabullendiğimi beyan ederim.

İbrahim YAYLALI, Eylül-2018

Özet

KOMPLEKS DIŞ ORTAMLAR İÇİN SEZGİSEL GÜZERGAH PLANLAMA ALGORİTMALARI

İbrahim YAYLALI

KTO Karatay Üniversitesi Fen Bilimleri Enstitüsü
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Tez Danışmanı: Dr. Öğr. Üyesi H. Oktay ALTUN
Tez Yardımcı Danışmanı: Dr. Öğr. Üyesi Amir YAVARİABDİ

Eylül 2018

Bu tezde a) A* algoritmasının kompleks harita kurguları için genellenmesi ve b) A* algoritmasının geliştirilerek düğüm keşfeden seviye tabanlı yeni bir versiyonunun (DS-A*) gene kompleks dış ortamlar için önerilmesi konularında çalışmalar yapılmıştır. Gerçek hayatta karşımıza çıkan dış mekanlarda güzergah belirleme probleminde arazi yapıları önemli ölçüde çeşitlilik göstermektedir. A* algoritması vasıtasıyla bu tip karmaşık ortamlarda en uygun güzergahın bulunabilmesi amacıyla bu algoritmanın tadil edilmesi önemlidir. Öte yandan A* algoritmasının en önemli handikaplarından birisi ölçeklendirilebilir olmamasıdır. Haritalar büyüdükçe A* algoritmasının güzergah hesaplama yükü ve haliyle hesaplama için gereken zaman üstel şekilde artmaktadır. Seviye kümesi algoritmasının seviye özelliği, Dijkstra algoritmasının yeni düğümleri keşfetme yöntemi ve A* algoritmasının sezgiselliği hibritlenerek bu tezde önerilen DS-A* algoritması, A* algoritmasının performansında önemli iyileştirmeler sağlamıştır.

Anahtar kelimeler: A* Algoritması, Dijkstra Algoritması, DS-A* Algoritması, Engelden Kaçınma, Güzergah Planlama, Hızlı Yürüyen Algoritması, Otonom Araç, Seviye Kümesi Algoritması

Abstract

HEURISTIC PATH PLANNING ALGORITHMS FOR
COMPLEX OUTDOOR ENVIRONMENT

İbrahim YAYLALI

KTO Karatay University
The Graduate School of Natural and Applied Sciences
Master of Science Thesis in Electrical and Computer Engineering

Advisor: Asst. Prof. H. Oktay ALTUN
Co-Advisor: Asst. Prof. Amir YAVARİABDİ

September 2018

In this thesis, we study a) generalization of A* algorithm for complex environments and b) a new version of A* node detecting and level based version is proposed for quicker route planning for complex map scenarios. In real world, there is a significant variation in landscapes of maps. It is important to modify this algorithm in order to find the most suitable route in such complex environments via A* algorithm. On the other hand, one of the disadvantages of A* algorithm is that it is not scalable. As the maps grow, the calculation load of A* algorithm and the time required for route estimation increases rapidly. The level feature of the level set algorithm, the method of discovering new nodes of Dijkstra algorithm and the intuition behind A* algorithm are adopted and the proposed DS-A* algorithm provided significant improvements over the performance of A* algorithm.

Keywords: Autonomous vehicle, A* Algorithm, Dijkstra Algorithm, DS-A* Algorithm, Fast Marching Algorithm, Level Set Algorithm, Obstacle Avoidance, Path Planning

Teşekkür

Çalışmalarım boyunca her türlü fedakarlığı gösteren ve daima sabırla destek olan eşim Şerife YAYLALI'ya, maddi ve manevi olarak desteklerini hiç esirgemeyen aileme, engin bilgi ve tecrübeleriyle bana yön veren eş danışmanım Dr. Öğr. Üyesi Amir YAVİABDİ hocama ve tanıdığım günden beri bir hoca olmanın dışında aynı zamanda bir abi gibi davranan ve işleri kolaylaştıran yaklaşımıyla tez danışmanım Dr. Öğr. Üyesi H. Oktay ALTUN'a teşekkürü bir borç bilirim.

İbrahim YAYLALI
Eylül-2018

İçindekiler

Şekil Listesi

Tablo Listesi

Algoritma Listesi

Kısaltmalar

Kısaltmalar	Açıklamalar
A*	A yıldız algoritması
D*	D yıldız algoritması
DEM	*Digital elevation model*
DS-A*	Düğüm keşfeden A yıldız algoritması
GIS	*Geographic information system*
JPEG	*Joint photographic experts group*
MCA	*Multi-layered cellular automata*
PNG	*Portable network graphics*
Theta*	Teta yıldız algoritması
QGIS	*Quantum geographic information system*

Semboller

Semboller	Açıklamalar
$f(n)$	Hesaplama yapan sezgisel fonksiyon (toplam maliyet)
$g(n)$	Başlangıç düğümünden mevcut düğüme kadar gelmenin maliyeti
$h(n)$	Mevcut düğümden hedef düğüme varmak için tahmin edilen mesafe

1 Giriş

Başlangıç noktasından hedef noktasına ulaşmak için geçilecek yolun gerekli hesaplamalar yapılarak belirlenmesine güzergah planlama denilmektedir. Bu hesaplamalar gidilecek yolun hangi tür araçla ve hangi özelliklerdeki bir ortamda katedileceğiyle yakından ilgilidir. Günümüzde otonom araçların geliştirilmesiyle ilgili çalışmalar hız kazanmıştır. Bu doğrultuda insansız hava araçları, denizaltılar, otonom demiryolu araçları, askeri arazi keşif araçları ve otomobiller vb. olmak üzere, farklı alanlarda otonom araçlar geliştirilmektedir. Geliştirilen bu araç türlerinin güzergahlarının planlanmasında her bir araç türü için çalışma ortamına göre farklı algoritma yapıları kullanılması gerekmektedir. Bu tez çalışmasında otonom kara araçları için güzergah planlama çalışması yapılmıştır ve çalışma öncesinde literatür bunun gereklerine göre incelenmiştir.

Bu doğrultuda Javier V. Gomez ve arkadaşları kesin olmayan ortamlarda oluşturulan güzergahlarda oluşan keskin yön değiştirmelerini ortadan kaldırmak ve pürüzsüz bir akışla güzergah oluşturmak için *fast marching square* algoritması ile sanal ya da fiziksel liderler eşliğinde güzergah planlama çalışması yapmışlardır [1]. Santiago Garrido ve arkadaşları dış ortamlar için *fast marching* algoritması ile yeni bir güzergah planlama yöntemi önermektedirler [2]. Bu yönteme göre *fast marching* algoritması iki boyutlu kare varlık ızgarasına uygulanmak yerine üç boyutlu ortamda üçgen ağ yapısına uygulanmaktadır [2]. Bu yöntemin getirdiği yenilik ise güzergah planlama işlemine başlamadan önce üç boyutlu yüzey özelliklerinden çıkarılan bilgiye dayanarak bir ağırlık matrisi hesaplanmasıdır [2]. Bu ağırlık matrisinin hesaplanmasında yüzey pürüzlülüğü, yükseklik ve küresellik gibi özellikler dikkate alınmış olup çıkarılan sonuç ile en az enerjinin tüketileceği yollar, en düz yollar vb. çıkarımlarla *fast marching* algoritmasının yayılma hızının ve güzergahının belirlenmesinde kullanılmaktadır [2]. David Gonzalez Bautista ve arkadaşları hareket planlama ile ilgili bir derleme makalesi yazarak burada literatürdeki güzergah planlama uygulamalarını çok güzel bir şekilde

özetlemişlerdir [3]. Bunlardan dikkat çeken bazılarını sıralayacak olursak F. Marchese uluslararası bir konferansta çoklu robotlar için Dijkstra algoritması ile güzergah planlama yönteminden bahsetmiştir [4]. Bu yöntemde çok katmanlı hücresel otomata (*MCA*) mimarisi kullanılmıştır ve yönlü (*anizotropik*) yayılmaya dayanmaktadır [4]. Joo Young Hwang ve arkadaşları hem verimlilik hem de doğruluk için grafik optimizasyonu ile hızlı güzergah planlama yöntemini önermişlerdir, bu yönteme göre bir yandan geleneksel karesel tabanlı güzergah planlama yaklaşımındaki bir çok kare kullanımının algoritmayı yavaşlatması önlenmeye çalışılırken diğer yandan ise güzergahın geniş alanlarda kullanılan büyük karelerin içlerinden geçememesi nedeniyle oluşacak güzergahtaki uzamaların önüne geçilmeye çalışılmıştır [5]. Anthony Stentz kısmen tanımlanabilmiş ortamlarda çalışan algoritmaların sanki tam bilinen bir ortamda çalışıyormuş gibi varsayımlar ile çalıştırılmasından kaynaklı problemleri ortadan kaldırmak için, tanımlanmamış, kısmen tanımlanmış veya tanımlanmış ortamlarda çalışabilen ve tanımlanamayan kısımlarda kendini optimize edebilen D* algoritmasını önermiştir ki bu algoritma A* algoritmasından geliştirildiği için bu ismi almıştır [6]. Dave Ferguson ve arkadaşları düşük maliyetli algoritma geliştirmek için düzenli varlık ızgarası ve düzensiz varlık ızgarası ile yapılan güzergah planlama yöntemlerini lineer interpolasyon yöntemi ile geliştirerek "Alan D* Algoritmasını" geliştirmişlerdir [7]. Alex Nash ve arkadaşları varlık ızgarası yönteminde varlık hücrelerinin kenarlarını yol kabul eden güzergah planlamasında bu yöntemin algoritmayı kısıtlaması nedeni ile ortaya çıkabilecek optimal olamayan ya da gerçek dışı güzergah desenlerine çözüm olarak A* algoritmasının bir varyantı olan Theta* algoritmasını önermişlerdir ve yine A* agoritmasının bir varyantı olan D* algoritması ile kıyaslamışlardır [8]. Maxim Likhachev ve arkadaşları gerçek dünyadaki bir sonraki eylemin ne olacağına karar vermek için kısıtlı bir zaman olması, dinamik olarak değişen çevre koşulları ve kısa zamanda üretilen güzergahın hatalı kısımlarının bulunabilmesi gibi durumlara çözüm üretebilmek amacı ile ilk başta yaklaşık bir güzergah üretip sonra o güzergahı eyleme dökerken daima iyileştirmeye devam eden dinamik grafikte daima arama yöntemini önermişlerdir [9].

Güzergah planlama algoritmalarını ortamına göre düşündüğümüz zaman ise biri iç ortam diğeri ise dış ortam olmak üzere iki farklı ortam türü bulunmaktadır. İç ortamlara örnek verecek olursak, bir tarafta hastane, okul, alışveriş merkezi ve ev gibi sosyal alanlar karşımıza çıkarken diğer taraftan fabrika ve kargo depoları gibi teknik alanlar karşımıza çıkmaktadır. Dış ortamlar ise işlenmişlik durumuna göre yapılandırılmış dış ortam ve yapılandırılmamış dış ortam olarak ikiye ayrılmaktadır. Yapılandırılmış dış ortamlara örnek verecek olursak yollar, otoparklar ve şehir parkları gibi alanlar öne çıkarken

yapılandırılmamış dış ortamlara ise dağlık araziler düzlük araziler, ormanlar, bataklıklar ve bunların arasında ve kenarında kalan deniz, göl ve okyanus gibi alanlar karşımıza çıkmaktadır. Biz bu tez çalışmasında dış ortam için güzergah planlama uygulaması üzerine yoğunlaştık ve dış ortamlardan yapılandırılmış ve yapılandırılmamış dış ortamları hibrit olarak ele aldık. Kısaca özetleyecek olursak deniz, engebeli arazi, ormanlık ve bataklık alanlar girilmez alan olarak (engel) sınıflandırılırken kalan düz arazi yavaş ilerlenebilir arazi olarak sınıflandırılmıştır. Bunların yanında ise yapılandırılmış dış ortam olarak yollar eklenmiş ve yollar serbest ilerlenebilir dış ortam olarak değerlendirilmiştir.

Dış ortamlarda güzergah planlamanın çeşitli zorlukları bulunmaktadır. Bunlara örnek verecek olursak ortamın yeterince tanınmaması, engel çeşitliliğinin fazla olması dolayısıyla hepsini tanımanın oldukça zor olması ve farklı ortamlarda aracın farklı seviyelerde zorluklarla karşılaşacak olması gösterilebilir. İkili varlık ızgarası haritası yönteminde ortamlar ya engel ya da gezilebilir ortam olarak tanımlanmaktadır. Fakat her gezilebilir ortam aynı gezilebilirlik seviyesinde olmadığı için planlanan güzergahın verimliliği her farklı ortamda değişmektedir. Burada aracın karşılaştığı farklı seviyelerdeki zorluklar çok kritik bir öneme sahiptir ve ikili varlık ızgarası haritası yönteminin iyi bir seçenek olmadığını ortaya koymaktadır.

Farklı seviyelerdeki zorlukların çözümü için literatürde çok sayıda yöntem bulunmaktadır. Bu yöntemlerin bir kısmını Tablo 1.1'de olduğu gibi farklı kategoriler halinde özetleyebiliriz. Grafik arama tabanlı planlayıcılarda yaygın olarak kullanılan Dijkstra algoritması ile geliştirilen ilişkili ağırlıklar ile bilinen düğüm/hücre arama uzayı yöntemini [10–13] ve ortama göre varlık ızgarası ve düğüm/hücre ağırlığı hesaplama yöntemini [14, 15] örnek verebiliriz. Grafik arama tabanlı planlayıcılardan A* algoritmasına ise Voronoi maliyet fonksiyonları ile herzaman D* yöntemini [16–18], hibrit sezgisel A* yöntemini [19, 20], Voronoi/kafes ortam temsili ile A* yöntemini [11, 21, 22] ve [23]'teki gibi PAO* yöntemini [24] örnek verebiliriz. Yine grafik arama algoritmalarından durum kafesleri algoritması ile geliştirilen yerel bir değişken ızgarada, temaların karmaşıklığına bağlı olarak, ayrıştırılmış ortam yöntemi [17, 18, 25, 26], ve mekansal ve zamansal kafesler (zaman ve hız boyutlarını dikkate alarak) yöntemini [27–32] örnek verebiliriz. Örnekleme tabanlı planlayıcılardan hızlı keşfedilen rasgele ağaç (*rapidly exploring random tree*) algoritması ile geliştirilen rasgele ağaç üretmek için fiziksel ve mantıksal önyargı kullanımı yöntemini [33–35], *rapidly exploring random tree (RRT)*'nin yakın komşu araması yapan ve ağacı yeniden yapılandıran varyasyonu olan RRT* ile herzaman planlama yöntemini [13, 36–38] ve RRT ile yörünge koordinasyonu yöntemini [39] örnek verebiliriz. İnterpolasyon eğrisi

planlayıcılarından çizgi ve daire algoritması ile geliştirilen yol uydurma ve bilinen yol noktalarının enterpolasyonu yöntemini [40, 41] örnek verebiliriz. İnterpolasyon eğrisi planlayıcılarından ikinci bir algoritma olarak klotoyid eğrileri algoritması ile geliştirilen düz, klotoyid ve dairesel parçalarla parçalı yörünge üretimi yöntemini [42–47] ve çevrim içi değerlendirmede alınacak en iyi klotoyid ilkelerin çevrim dışı üretimi yöntemini [48–50] örnek verebiliriz. Yine interpolasyon eğrisi planlayıcılarından polinom eğrileri algoritması ile geliştirilen kübik düzen polinom eğrileri yöntemini [27, 51] ve yüksek mertebeden polinom eğrileri yöntemini [52–56] örnek verebiliriz. İnterpolasyon eğrileri planlayıcılarından Bezier eğrileri algoritması ile geliştirilen eldeki durum için optimal kontrol noktaları konumunun seçimi yöntemini [55, 57–61] ve rasyonel Bezier eğrileri uygulaması yöntemini [62, 63] örnek verebiliriz. İnterpolasyon eğrileri planlayıcılarından son olarak spline eğrileri algoritması polinom parçalı uygulama yöntemini [14, 52, 64] ve temel kıvrımlar (b-spline) yöntemini [65–67] örnek verebiliriz.

Tez çalışmamızda problemin çözümüne grafik arama algoritmaları ile devam etmeyi tercih ettik. Bu doğrultuda geliştirdiğimiz güzergah planlama algoritmasında hedefi Dijkstra algoritmasında olduğu gibi düğüm keşfederek bulmaya çalıştık. Düğümleri keşfederken ise level set algoritmasındaki gibi başlangıç konumundan etrafa bir su dalgası gibi yayılan keşif yöntemini tercih ettik, fakat matematiksel karmaşıklığı azaltarak algoritmamızı hızlandırmak için *level set* algoritmasındaki gibi sürekli bir form yerine *fast marching* algoritmasındaki gibi kesikli bir form tercih ettik. Bunun dışında ise algoritmamızın arama esnasında öncelikle serbest gezilebilir yolları bulamadığı taktirde ise zor gezilebilir yolları tercih etmesi için araziye farklı gezilebilirlik katsayıları verdik. Fakat bu durumda algoritmanın güzergahı serbest gezilebilir bölgede planlamaya çalışırken yolu uzatmaması için ise A* algoritmasında olduğu gibi algoritmamıza sezgisel hesaplamalı bir yapı ekledik. Bu sezgisel yapıya göre algoritmamız başlangıçta hedefin kuş uçuşu uzaklığını hesaplamaktadır. Sonrasında ise arama işlemi boyunca geçtiği yolun uzunluğuyla hedefe ulaşması için gitmesi gereken yolun kuş uçuşu uzaklığını toplamaktadır. Topladığı bu uzaklık değeri ile başlangıçtaki kuş uçuşu uzaklığın bir güvenlik katsayısı ile çarpımını kıyaslamaktadır. Eğer arama esnasında elde edilen uzaklık değeri başlangıçtaki katsayı ile çarpılmış değere ulaşırsa arazideki katsayı ihmal edilerek güzergah serbest gezilebilir bölgeden zor gezilebilir bölgeye geçiş yapmaktadır. Geliştirdiğimiz bu algoritmanın dışında klasik A* algoritmasını da düzenleyerek farklı arazi katsayıları ile öncelikle serbest gezilebilir bölgeleri sonra zor gezilebilir bölgeleri tercih eder hale getirdik. Tezimizin sonunda ise bu iki algoritmayı performans olarak kıyasladık.

Algoritma grubu	Algoritma	Yöntem açıklaması	Uygulama örnekleri
Grafik arama tabanlı planlayıcılar	Dijkstra algoritması	İlişkili ağırlıklar ile bilinen düğüm/hücre arama uzayı	[10–13]
		Ortama göre varlık ızgarası ve düğüm/hücre ağırlığı hesaplama	[14, 15]
	A* algoritma ailesi	Voronoi maliyet fonksiyonları ile her zaman D*	[16–18]
		Hibrit sezgisel A*	[19, 20]
		Voronoi/kafes ortam temsili ile A*	[11, 21, 22]
		[23]'teki gibi PAO*	[24]
	Durum kafesleri	Yerel bir değişken ızgarada, temaların karmaşıklığına bağlı olarak ayrıştırılmış ortam	[17, 18, 25, 26]
		Mekansal zamansal kafesler (zaman ve hız boyutlarını dikkate alarak)	[27–32]
Örnekleme tabanlı planlayıcılar	RRT	Rasgele ağaç üretmek için fiziksel ve mantıksal önyargı kullanılır	[33–35]
		RRT* ile her zaman planlama	[13, 36–38].
		RRT ile yörünge koordinasyonu	[39]

İnterpolasyon eğrisi planlayıcıları	Çizgi ve daire	Yol uydurma ve bilinen yol noktalarının enterpolasyonu	[40,41]
	Klotoyid eğrileri	Düz, klotoyid ve dairesel parçalarla parçalı yörünge üretimi	[42–47]
		Çevrim içi değerlendirmede alınacak en iyi klotoyid ilkellerin çevrim dışı üretimi	[48–50]
	Polinom eğrileri	Kübik düzen polinom eğrileri	[27,51]
		Yüksek mertebeden polinom eğrileri	[52–56]
	Bezier eğrileri	Eldeki durum için optimal kontrol noktaları konumunun seçimi	[55, 57–61]
		Rasyonel Bezier eğrileri uygulaması	[62,63]
	Spline eğrileri	Polinom parçalı uygulama	[14, 52, 64]
		Temel kıvrımlar (b-spline)	[65–67]
Uyarlanabilen planlayıcılar	Genetik algoritmalar	Genetik algoritma ile entegre planlama ve kontrol sistemi	[68]
		Genetik algoritma ile uyarlanabilen hareket planlama	[69]

Tablo 1.1: Otonom sürüş senaryolarında uygulanan güzergah planlama tekniklerinin sınıflandırılması [3].

Bu tez çalışmamızda Bölüm 2'de kuramsal temellerden bahsederek okuyucuyu tez konusuna hazırlamaya çalıştık ve tezimizin dayanağı olan algoritmaları anlattık. Bölüm 3 ise tezimizde önerdiğimiz algoritma ve yaptığımız çalışmayı detaylandırdık ve Bölüm 3.1'de veri elde etme yöntemlerine, Bölüm 3.2'de algoritmamızın bileşenlerine ve Bölüm 3.3'te ise algoritma uygulamalarımıza değindik. Dördüncü bölümde yaptığımız çalışmanın deneysel sonuçlarına ve karşılaştımalarına yer verdik. Sonuç bölümünde ise tez çalışmamızla ilgili değerlendirme yapıp tez çalışmamızın sonuçlarını yorumladık.

2 Kuramsal Temeller

2.1 Grafik Arama Algoritmaları

Bilgisayar biliminde farklı veri yapıları üzerinde bir bilginin aranması esnasında yararlanılan algoritmaların geneli arama algoritmaları olarak anılmaktadır. Dosyalarda kelimelerin aranması, ağaç yapılarında düğümlerin aranması ya da diziler üzerinde verilerin aranması vb. durumlar bu algoritmanın çalışma alanlarına örnek olarak verilebilir. Grafik arama algoritmaları ise arama algoritmalarının grafik verileri üzerinde özelleşmiş halidir. Örneğin grafik üzerindeki bir rengi veya bir şekli vb. özellikleri arama yöntemidir. Bu arama işlemini gerçekleştirmek için çeşitli veri kaynakları ve işleme yöntemleri kullanılmaktadır. Kaynak olarak kamera görüntüleri gibi hazır veriler kullanılabildiği gibi, grafik tasarım programları ile de istenilen grafikler üretilebilmektedir. İşleme yöntemlerinde ise ham kamera görüntülerinin çeşitli genel amaçlı grafik tasarım programları ve çeşitli matematiksel işlem programları kullanabileceğimiz gibi amaca özel yazılımlar da kullanabilmekteyiz. Amaca özel yazılımlara tezimizdeki çalışmalarda da kullandığımız gibi haritacılıkta yoğun bir şekilde kullanılan *geographic information system (GIS)* yazılımlarını örnek verebiliriz. Bu tür yazılımlar daha önce uydu görüntüleri ile elde edilmiş sayısal veriler ile fiziksel dünyada farklı yöntemlerle elde edilmiş grafiksel verileri bütünleşik olarak işlememize ve gerçek fiziksel ortamlardan uydu ile elde edilen sayısal verilerin karmaşıklığını grafiksel verilerle sadeleştirmemize olanak sağlamaktadır. Grafik arama algoritmaları ise elde edilen grafikselleştirilmiş bu veriler üzerinde bilgi arama işlemini gerçekleştirmektedir.

2.2 Dijkstra Algoritması

Dijkstra algoritması *fast marching* metodu ile çok yakından alakalıdır ve *fast marching* metodu Dijkstra algoritmasının sürekli formülasyonu olarak da adlandırılabilir. Dijkstra algoritması grafik üzerinde verilen başlangıç ve hedef noktaları arasındaki en kısa güzergahı bulan bir grafik arama algoritmasıdır. Dijkstra algoritmasına göre verilen ağırlıklandırılmış grafik üzerinde ilk olarak bütün düğümler keşfedilmedi olarak işaretlenir. Ardından başlangıç düğümü mevcut düğüm olarak işaretlenir. Sonrasında mevcut düğümden tüm komşu düğümlere olan geçici mesafe hesaplanır. Örneğin herhangi bir komşu düğümden başlangıç düğümüne kaydedilen mesafe daha önceden aynı iki düğüm arasında kaydedilen mesafeden küçük ise küçük olan mesafe eskisinin üzerine yazılır ve eski uzun mesafe silinir. Mevcut düğümün tüm komşularla arasındaki mesafeler kaydedildikten sonra mevcut düğüm keşfedildi olarak etiketlenir ve bu düğüm bir daha hiç bir zaman tekrar keşfedilmez. Kaydedilen mesafe ise nihai ve optimum mesafe olarak kalır. Eğer bütün düğümler keşfedildi olarak etiketlenmişse işlem biter değilse sıradaki en yakın düğüm mevcut düğüm olarak etiketlenerek ikinci adıma geçilir [70].

2.3 Sezgisel Optimizasyon

Optimizasyon kısaca eldeki kısıtlı imkanları kullanarak mümkün olan maksimum faydayı ya da en iyi sonucu elde etme çabasıdır. Sezgisel optimizasyon ise gerçek hayatta net bir matematiksel çözüm formülü olmayan karmaşık problemlerin çözümünde problemin matematiksel modellemesi sonrası istatistikler ve algoritmalar vb. gibi bilimsel yöntemlerden de faydalanarak çözülmeye çalışılmasıdır. Sezgisel optimizasyon problemin tam çözümünü garanti etmez, tam çözüme yaklaşmaya çalışır. Sezgisel optimizasyon algoritmasının performansı çözüme yaklaşma oranı ile belirlenir [71].

2.4 Aç Gözlü Önce-En-İyi Arama Algoritması

Önce-en-iyi arama (*best first search*) algoritması bir ya da birden fazla sezgisel fonksiyondan oluşabilmektedir, ve hangi sezgisel fonksiyonun daha

iyi sonuç vereceğine ya da gerekliyse birden fazla sezgisel fonksiyonun kullanılması gerekip gerekmediğine iyi karar verilmesi gerekmektedir [72]. Ayrıca burada dikkat edilmesi gereken önemli konulardan birisi de sezgisel fonksiyonun kabul edilebilir olup olmamasıdır. Eğer sezgisel fonksiyonun maliyeti gerçek maliyetten daha düşük ise bu kabul edilebilir bir fonksiyondur fakat sezgisel fonksiyonun maliyeti gerçek maliyetten daha yüksek ise bu fonksiyon kabul edilebilir bir fonksiyon değildir [72]. Önce-en-iyi arama algoritması aynı zamanda aç gözlü önce-en-iyi arama (*greedy best first search*) algoritması olarakta isimlendirilmektedir. Bunun sebebi algoritmanın problem çözümündeki adımlarda sadece bulunduğu adımdaki sezgisel maliyeti en düşük olan yolu seçip sonraki adımları ve gerçek maliyetleri göz önünde bulundurmamasıdır. Bu durum algoritmanın o adımda dezavantajlı fakat problemin tam çözümünde avantajlı olan yolları seçmesini önlediği için algoritma aç gözlü önce-en-iyi arama algoritması olarak isimlendirilmektedir.

2.5 A* Algoritması

A* algoritması, derinlik öncelikli bir arama yöntemidir. Bu nedenle kısa sürede başlangıç noktasından uzaklaşıp hedefe yaklaşmaya çalışır. A* algoritması aynı zamanda gerçek maliyet öncelikli arama yöntemi ile sezgisel maliyet öncelikli arama (önce-en-iyi arama) yönteminin hibritleşmesi ile oluşmuş maliyet öncelikli bir arama yöntemidir [72]. A* algoritmasında kullanılan sezgisel fonksiyon mevcut konumun hedef konuma kuş uçuşu uzaklığından hesaplanmaktadır [73].

A* algoritmasında mevcut düğüme kadarki gerçek maliyet $g(n)$ ve mevcut düğümden sonraki seçeneklerdeki sezgisel maliyetler $h(n)$ toplanır ve toplam maliyetler $f(n)$ elde edilir, bu seçeneklerden en düşük maliyetli olandan ilerlenir ve diğer seçeneklerin maliyetleri de hafızada tutulur [73]. Bu işlem hedefi buluncaya kadar devam eder [73]. Fakat henüz daha hedefe ulaşmadan önce herhangi bir adımdaki $f(n)$ önceki adımlardaki $f(n)$'lerden daha yüksek çıkarsa geri adım atılır ve düşük maliyetli yoldan devam edilir [73].

2.6 A* Algoritmasının Dezavantajları ve Çözüm Önerisi

A* algoritmasında yol ya yoktur ya da var ise hepsi aynıdır mantığı ile gelecekteki tüm arazi koşullarının sezgisel maliyeti eşit kabul edilerek ilerlenir. Yani önümüzdeki yol seçenekleri nereden geçerse geçsin maliyet sadece kuş uçuşu mesafeye göre hesaplanır. Mesela yollardan biri 9.000 metre taşlık bir yol diğeri ise 9.001 metre otoban olduğunu düşünürsek algoritmamız 1 metre daha kısa olan taşlık yolu seçer ve otobandan gitmez. İkinci bir örnek olarak ise taşlık yolu engel, yani yol yok olarak sınıflandırdığımızı düşünelim, ve hedefe ulaşmak için taşlık bir yoldan 10 metre ilerleyerek karşı şeritteki yola geçmemiz gerektiğini düşünelim ve alternatif olarak ise 10.000 metre ilerideki kavşaktan dönüş yapıp gelmemiz gerektiğini düşünelim, eğer taşlık yolu yol yok olarak sınıflandırırsak 10.000 metre ilerideki yolu gidip dönme esnasında 20.000 metre yol kat etmek zorunda kalacağız. Normal koşullarda performansı çok iyi olan A* algoritması bu tür durumlarda başarısız kalmaktadır. Bu problemin çözümü için önerimiz farklı arazi yapılarının farklı arazi katsayıları ile etiketlenmesidir. Etiketlenmiş bu farklı katsayılı arazilerde algoritma arama yaparken ise güvenlik ve maliyet optimizasyonu ile önce en kolay arazi bölgelerini sonra daha zor olanlarını tercih etmelidir. Bu doğrultuda tezimizde iki farklı öneride bulunduk, bunlardan birisi kompleks ortamda A* algoritması diğeri ise kompleks ortamda DS-A* algoritmasıdır. Bunlardan kompleks ortamda A* algoritmasına Bölüm 2.7'de değindik, kompleks ortamda DS-A* algoritmasına ise Bölüm 3'te detaylı bir çalışma yaptık. Bu iki algoritmanın değerlendirmesini ve karşılaştırılmasını ise Bölüm 4'te yaptık.

2.7 Kompleks Ortamda A* Algoritması

A* algoritmasının ikili ortamlardaki bütün yolları eşit özellikte görmesinden kaynaklanan başarısızlığını ortadan kaldırmak ve algoritmanın performansını yükseltmek için bu tez çalışmamızdaki önerilerimizden birisi de "Kompleks Ortamda A* Algoritması" yöntemidir. Bu yönteme göre haritamızdaki alanları klasik A* algoritmasındaki gibi gezilebilir ya da gezilemez olarak sınıflandırmak yerine gezilemez, yavaş gezilebilir, orta hızda gezilebilir, yüksek hızda gezilebilir ve aşırı hızda gezilebilir gibi çok daha fazla sınıfa ayırabiliriz. Burada kaç sınıfa ayıracağımız algoritmanın kullanım yeri ve amacına göre değişmektedir. İstediğimiz sayıda sınıflandırma yapabiliriz, bu durum ihtiyaca bağlıdır. Biz bu tez çalışmasında engel (hiç gezilemez),

arazi (yavaş gezilebilir) ve yol (hızlı gezilebilir) olarak üç farklı sınıflandırma yaptık.

3 Metod, Veri ve Algoritmalar

3.1 Veri Elde Etme Yöntemleri

Bu tez çalışmamızdaki güzergah planlama algoritmalarında harita, algoritmalara hazır olarak verilmektedir. Algoritma ise haritayı kullanabileceği hâle dönüştürdükten sonra kullanmaktadır. Hazır olarak verdiğimiz haritalar grafik tabanlı haritalardır, yapısal olarak PNG ve JPEG vb. formatlarda olabilmektedir. Bu haritaların elde edilmesinde çeşitli yöntemler kullanılabilmekle birlikte genellikle *GIS* yazılımı çıktıları ve grafik çizim programı çıktıları tercih edilmektedir.

3.1.1 GIS Yazılımı Çıktıları

GIS (*Geographic Information System*) in kısaltılmışı olup haritacılık ve istatistik alanlarında yaygın olarak kullanılan programlardır. NASA'nın *EARTHDATA* sitesi üzerinden *Digital Elevation Model* (*DEM*) verileri indirilerek *GIS* yazılımları ile işlenebilir ve grafiksel bir haritalar elde edilebilir ve bu grafiksel haritalar üzerinde güzergah planlama algoritmaları çalıştırılabilir. Fakat tezimizde asıl amacımız grafiksel harita üretmek değil, güzergah planlama algoritması geliştirmektir. Bu nedenle *GIS* yazılımları üzerinde grafiksel harita üretmek yerine, zaten üretilmiş ve paylaşılmış grafiksel harita verilerini *GIS* yazılımları üzerinde derleyip kullanmak daha hızlı bir çözüm yöntemidir. Yaptığımız çalışmada hazır olarak erişebildiğimiz harita verilerinin aşırı büyük boyutları nedeniyle işlenebilirlik bakımından istediğimiz özelliklerde olmadığını gözlemledik. Bu verileri işleyip istediğimiz özelliklere getirmek ise çok zaman gerektirmekteydi. Hazır harita kullanmanın bu dezavantajları nedeni ile tezimizde farklı bir veri elde etme yöntemi bulmaya karar verdik.

3.1.2 Grafik Çizim Programı Çıktıları

GIS çıktıları yöntemine göre çok daha hızlı işleyebileceğimizi düşündüğümüz haritanın kendi ürettiğimiz test haritası olduğuna karar verdik. Geliştirdiğimiz algoritmanın hızlı sonuç verebilmesi için haritanın çok hızlı işlenebilir olması gerekmektedir. Bu doğrultuda test amaçlı kullanacağımız haritamızı grafik tasarım programı ile basitleştirdik. Bu haritada yollar beyaz ile arazi gri ile ve engeller ise siyah ile gösterilmektedir.

Şekil 3.1: Grafik çizim programı ile elde edilen 100 × 100 piksel boyutunda üç ortamlı test haritası: Siyah gezilemez ortam, beyaz serbest gezilebilir ortam ve gri zor gezilebilir ortam

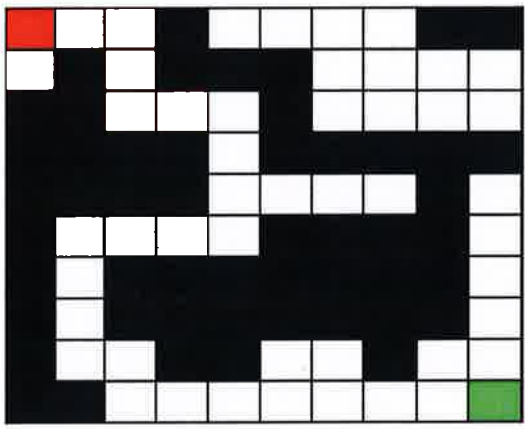
3.2 Algoritma Bileşenleri

Tez çalışmamızda kullandığımız güzergah planlama algoritmaları, problemin farklı kısımlarını farklı alt bileşenler ile çözmektedir. Problemin çözümünü kolaylaştıran bu alt bileşenleri varlık ızgarası, komşu tanıma, spiral keşif, seviye, etiketleme, listeleme, analiz ve sentez bileşeni olarak sıralayabiliriz.

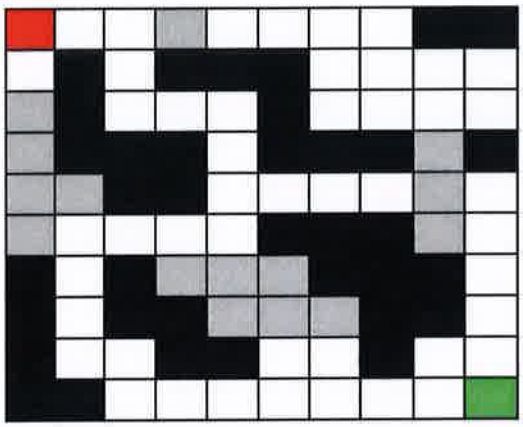
3.2.1 Varlık Izgarası Bileşeni

Tezimizde kullandığımız güzergah planlama algoritmalarında haritayı daha basit bir şekilde analiz etmek amacı ile algoritmalara varlık ızgarası bileşeni eklenmiştir. Varlık ızgarası yöntemine göre iki boyutlu bir harita x ve y eksenleri doğrultusunda küçük karelere bölünür ve engel olan bölgeler 1 (siyah,

engel var) ile temsil edilirken serbest gezinme bölgeleri 0 (beyaz, engel yok) ile temsil edilmektedir. Algoritma bu bilgileri kullanarak engellerin olduğu varlık hücrelerine uğramadan sadece engellerin olmadığı varlık hücrelerine uğrayarak hedefine ulaşmaya çalışır. Fakat biz tezimizde algoritmanın verimliliğini artırabilmek ve gerçek dünyaya daha yakın bir çalışma ortamı oluşturabilmek için fazi (*fuzzy*) mantığından da esinlenerek kısmi serbestlikteki gezinme bölgeleri oluşturmak istedik. Burada hücreleri ikilik (*binary*) olarak kodlamak yerine 8 bit olarak kodladık. Böylelikle alanımızda 256 adete kadar farklı serbestlik bölgesine sahip olma imkanı elde ettik. Fakat bu çalışmada algoritmamız henüz deneme aşamasında olduğu için 0 (beyaz, engel yok), 1-254 (gri, engel kısmen var) ve 1 (siyah, engel var) 3 farklı serbestlik bölgesini yeterli bulduk. Bu doğrultuda *photo image editor* isimli grafik tasarım programında test amaçlı yapay olarak 100 × 100 piksel boyutunda harita ürettik. Bu harita 3 farklı alan içermektedir, bu alanlar siyah engel bölgeleri, gri kısmen engelli yavaş ilerleme bölgeleri ve beyaz serbest gezinti bölgelerinden oluşmaktadır.



Şekil 3.2: İkili ortam için varlık ızgarası yapısı: Siyah var, beyaz yok, kırmızı(sol-üst) başlangıç düğümü ve yeşil(sağ-alt) hedef düğümü



Şekil 3.3: Üçlü ortam için varlık ızgarası yapısı: Siyah var, beyaz yok, gri kısmen var, kırmızı(sol-üst) başlangıç düğümü ve yeşil(sağ-alt) hedef düğümü

Şekil 3.4: Grafik çizim programı ile elde edilen 100 × 100 piksel boyutunda üç ortamlı test haritası: Siyah gezilemez ortam, beyaz serbest gezilebilir ortam ve gri zor gezilebilir ortam

3.2.2 Komşu Tanıma Bileşeni

Güzergah planlama algoritmalarında kullandığımız ikinci bileşen ise komşu tanıma bileşenidir. Bu bileşen sayesinde algoritmanın mevcut incelediği varlık hücresinin kullanılan algoritmaya göre tanımlanmış komşu hücreleri karakteristik olarak incelenir ve incelenen komşunun serbest gezinme hücresi mi, kısmi gezinme hücresi mi yoksa bir engel hücresi mi olduğu belirlenir ve ilgili dizi üzerinde kayıt edilir, algoritmanın ileriki adımlarında ise bu bilgiler algoritma tarafından kullanılır.

3.2.3 Spiral Keşif Bileşeni

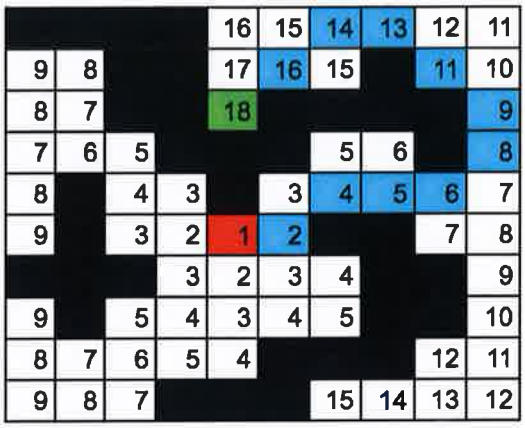
Dijkstra algoritmasında daha önce bahsedildiği üzere genişlik öncelikli arama (*breadth-first search*) yapılmaktadır. Bazı arama ağaçlarında sonraki arama katmanları belirli bir yönde ilerler iken bazı arama ağaçlarında başlangıç düğümüne göre 360° genişleyerek ilerlemektedir. 360° genişleyen arama ağaçlarında önemli konulardan birisi de aramanın ne şekilde genişleyeceğidir. Eğer düzenli bir genişleme planı oluşturulmaz ise algoritma ya bazı düğümleri kaçırarak ya da bazı düğümleri tekrar tekrar arayarak verimsizliğe veya hataya neden olabilmektedir. Bu tür durumlara önlem olması amacı ile DS-A* algoritmasında spiral genişleme bileşenini kullanmayı tercih ettik.

			62	59	56	54	51	48
44	37		63	61	58		49	46
38	31		64					40
32	26	22			17	23		33
39		16	9		5	11	18	24	27
45		10	4	1	2			28	34
			8	3	6	12			41
43		21	15	7	13	19			47
36	30	25	20	14				53	50
42	35	29				60	57	55	52

Şekil 3.5: Spiral keşif yapısı: Siyah engel, beyaz serbest gezilebilir bölge, kırmızı (1.kare) başlangıç düğümü, yeşil (64.kare) hedef düğümü, sayılar 1'den 64'e kadar keşif sırası

3.2.4 Seviye Bileşeni

DS-A* algoritmasında kullanılan diğer bir bileşen de seviye bileşenidir. Aslında DS-A* algoritmasını kısmen seviye bileşeni algoritmasına benzetmektedir. Seviye bileşenine göre arama işlemi ilk düğümden başladıktan sonra ilk bileşenin seviyesi 1 olarak etiketlenir ve spiral keşif bileşenindeki kurallara göre arama işlemi devam ederken her bir düğüm keşfedilirken o düğüme yatayda ve dikeyde komşu olan düğümlerin seviyesi mevcut düğümün seviyesinin bir fazlası olarak etiketlenir ve ilgili matrise kaydedilir. Bu şekilde arama işlemi hedefi buluncaya kadar devam edilir ve en büyük seviye olarak hedef düğümü etiketlenir.



Şekil 3.6: Seviye yapısı: Siyah engel, beyaz serbest gezilebilir bölge, kırmızı başlangıç düğümü, yeşil hedef düğümü, sayılar keşif esnasında oluşturulan 1'den 18'e kadar seviye değeri

3.2.5 Etiketleme Bileşeni

Algoritmalarımızda kullanılan bileşenler vasıtası ile analiz edilen ve keşfedilen düğümlerin bilgileri kullanım amacına göre sınıflandırılır ve sınıflandırılan bu bilgiler güzergahın bu bilgilerden faydalanılarak sentezi için sentez aşamasında kullanılmak üzere ilgili dizilere etiketlenerek kayıt edilir.

3.2.6 Listeleme Bileşeni

Listeleme bileşeni etiketleme bileşeninde etiketlenen bilgilerin güzergahın sentezi esnasında verimli ve kolay bir şekilde kullanılabilmesi için verileri kullanım amacına göre listeler.

3.2.7 Yönelim Matrisi Bileşeni

Yönelim matrisi bileşeni analiz sonucu hedef bulunduktan sonra, sentez esnasında güzergah üretilirken işlev gören bir bileşendir. Bu bileşen, boyutu tek sayılardan oluşan ve minimum 3×3 piksel boyutundan başlayan matrislerden oluşur. Matrislerin merkezi mevcut konuma denk gelecek şekilde her adımda kaydırılır ve matrislerin kalan hücreleri komşu olarak incelenir. Komşuların en küçük değerli olanı güzergahın bir sonraki düğümünü belirler ve güzergah bu düğüm doğrultusunda yönelerek ilerler.

3.2.8 Analiz ve Sentez Bileşenleri

Güzergah planlama amacı ile kullandığımız A* ve DS-A* algoritmalarında önemli iki bileşen daha mevcuttur. Bu bileşenlerden birisi analiz bileşeni diğeri ise sentez bileşenidir. Analiz bileşeninde başlangıç düğümünden hedef düğümüne kadar arama yapılır ve veriler sınıflandırılarak ve etiketlenerek kayıt edilir. Böylece harita hedefi bulacak şekilde tanınmış olur. Aslında buraya kadar anlattığımız kısımdaki varlık ızgarası bileşeni, komşu tanıma bileşeni, spiral keşif bileşeni, seviye bileşeni, etiketleme bileşeni, yönelim matrisi bileşeni ve listeleme bileşeni de analiz bileşeninin birer parçasıdır. Sentez bileşeninde ise analiz bileşeninde üretilen ve kayıt edilen veriler kullanılarak optimum güzergah üretilmeye çalışılır.

3.3 Güzergah Planlama Algoritması Uygulamaları

3.3.1 İkili Ortamlar İçin A* Algoritması

Algoritmayı detaylı bir şekilde inceleyecek olursak şu şekildedir: Güzergah planlama algoritmamız *MATLAB* üzerinde programlanıp, kullanıcı arayüzü oluşturulmuştur. Algoritmaya girdi verisi olarak PNG ve JPEG vb. yapıda bir harita verilmesi gerekmektedir. Bunun dışında harita boyutu x ve y koordinatı olarak kod üzerinden girilmelidir ve arayüz üzerinden başlangıç koordinatı, bitiş koordinatı, arazi katsayısı ve zıplama katsayısı girilmelidir. Burada bahsedilen arazi katsayısı harita üzerinde ara gezinme bölgesi olarak ifade edilen bölgenin serbest gezinme bölgesine kıyasla kaç kat yavaşlatıldığını göstermektedir. Zıplama katsayısı ise arazi gezinilirken kaç *grid*(kafes) hücresi atlayarak güzergah oluşturulacağını göstermektedir. Burada atlama katsayısı ne kadar büyük olursa algoritma o kadar hızlı çalışmaktadır ve aynı zamanda o kadar pürüzsüz bir güzergah oluşturmaktadır. Fakat bu katsayı fazla artırılırsa aşamayacağı engellerin üzerinden güzergah oluşturmasına neden olmaktadır. Bu nedenle bu katsayının uygulanacağı arazi, harita ölçeği ve araç özellikleri göz önünde bulundurularak iyi belirlenmesi gerekmektedir [74].

```
Result: Güzergah
if arayüz başlatılmamışsa then
    (manuel) Arayüzü başlat;
else
    if başlangıç değerleri girilmemişse then
        (manuel) başlangıç-x,y bitiş-x,y ve birinci, ikinci,
         üçüncü güzergahlar için zıplama katsayı değerlerini gir;
    else
        if bul butonuna basılmamışsa then
            (manuel) bul butonuna bas;
        else
            for güzergah 1 → 3 do
                haritayı oku, hedef kayıtçı matris'ini oluştur, hedef
                 noktasının değerini işaretle mevcut adımdaki güzergah için
                 zıplama katsayısını işleme al;
                optimum güzergah bulucuyu başlat;
                gerekli kayıtçı matrislerin oluşturulması;
                gidilmesi gereken mesafenin başlangıç düğümünü belirle;
                while 1 do
                    düğümü keşfedilmeyenler listesinden keşfedilenler listesine
                     taşı;
                    komşuları kontrol etmek için for döngüsüne gir;
                    eğer keşfedilmeyen komşu bulunduysa;
                    yolun engelden geçip geçmediğini kontrol et;
                    yol engelden geçiyorsa bu düğümü listeden kaldırarak bir
                     önceki adımdan farklı bir komşuya git;
                    yol engelden geçmiyorsa maliyeti hesapla ve yolu yeniden
                     yapılandır;
                end
                eğer optimum güzergahın boyutu 1'den büyükse haritayı
                 ekranda göster;
                eğer optimum güzergahın boyutu 1'den büyük değilse
                 güzergah bulunamadı mesajı ver;
            end
        end
    end
end
```

Algorithm 1: İkili ortamlar için A* algoritması sözde kodu

3.3.2 Kompleks Ortamlar İçin A* Algoritması

Bu yöntem ikili ortamlar için geliştirilen A* algoritması ile güzergah planlama uygulamasının genişletilmiş halidir. Bu yöntem algoritmamızın birden fazla gezilebilirlik bölgesinde arama yapabilir hâle gelmesini sağlamaktadır. Böylelikle gezilebilir alanın her bir kısmını aynı verimle gezilebileceği varsayımından kaynaklanacak problemlerin önüne geçilmiş olacaktır. Bu problemlere örnek verecek olursak en basit olanı yol ve taşlık arazi örneği olur. Bu örneğe göre önümüzde 2 farklı güzergah seçeneği olduğunu varsayalım biri 120 km/saat hızla gidebileceğimiz 1.200 metre uzunlukta otoban diğeri ise 15 km/saat hızla gidebileceğimiz 1.199 metre taşlık bir arazi. Bu iki seçenek eğerki ikili mantıkla işleyen yöntemle değerlendirilirse algoritmamız 1.199 metre taşlık araziyi tercih edecek ve yol çok uzun sürecek, aracımız bu yolculukta daha fazla yıpranacak ve daha fazla enerji tüketimi gerçekleşecektir. Aynı iki durum çoklu mantıkla işleyen yöntemle değerlendirildiğinde ise algoritmamız taşlık arazide daha yavaş gideceğini hesaba katacak ve 1.200 metre uzunluktaki otobanı tercih edecektir. Böylelikle hem enerjiden, hem zamandan ve hem de aracın daha az yıpranmasından dolayı tasarruf sağlayacaktır.

```
Result: Güzergah
if arayüz başlatılmamışsa then
    (manuel) arayüzü başlat;
else
    if başlangıç değerleri girilmemişse then
        (manuel) başlangıç-x ve başlangıç-y değerlerini gir;
        (manuel) bitiş-x ve bitiş-y değerlerini gir;
        (manuel) birinci, ikinci ve üçüncü güzergahlar için arazi ve zıplama
         katsayılarını gir;
    else
        if bul butonuna basılmamışsa then
            (manuel) bul butonuna bas;
        else
            for guzergah 1 → 3 do
                haritayı oku, hedef kayıtçı matris'ini oluştur, hedef
                 noktasının değerini işaretle mevcut adımdaki güzergah için
                 zıplama katsayısını işleme al;
                optimum güzergah bulucuyu başlat;
                gerekli kayıtçı matrislerin oluşturulması;
                gidilmesi gereken mesafenin başlangıç düğümünü belirle;
                while 1 do
                    düğümü keşfedilmeyenler listesinden keşfedilenler listesine
                     taşı;
                    komşuları kontrol etmek için for döngüsüne gir;
                    eğer keşfedilmeyen komşu bulunduysa;
                    yolun engelden geçip geçmediğini kontrol et;
                    yol engelden geçiyorsa bu düğümü listeden kaldırarak bir
                     önceki adımdan farklı bir komşuya git;
                    yol engelden geçmiyorsa arazi sınıfını belirle, maliyeti
                     hesapla ve yolu yeniden yapılandır;
                end
                eğer optimum güzergahın boyutu 1'den büyükse haritayı
                 ekranda göster;
                eğer optimum güzergahın boyutu 1'den büyük değilse
                 güzergah bulunamadı mesajı ver;
            end
        end
    end
end
```

Algorithm 2: Kompleks ortamlar için A* algoritması sözde kodu

3.3.3 İkili Ortamlar İçin DS-A* Algoritması

Bu bölümde Bölüm 2.2'de bahsedilen Dijkstra algoritmasına eklediğimiz özellikleri de içeren ve kısmen *level set* algoritmasına benzerlik gösteren DS-A* algoritması ile *MATLAB* üzerinde oluşturduğumuz güzergah planlamasına değinilmiştir. Buradaki en önemli özellik algoritmamızın bir yandan Dijkstra algoritmasının kesikli özelliğini kullanması diğer yandan ise seviye kümesi algoritması mantığı ile seviye seviye arama yaparak ilerleyip geri dönüşte de yine *level set* algoritmasındaki gibi seviyeleri tersten takip ederek güzergahı üretmesidir. Bu durum algoritmamızın hızını tatmin edici düzeyde artırmaktadır.

```
if başlangıç verileri girilmemişse then
    başlangıç ve bitiş pozisyonunu, harita boyutunu, görev ve seviye
     sayacını gir, kayıtçı matrisleri oluştur
else
    haritayı oku ve ön işlemden geçir;
    while 1 do
        if yol bulunmamışsa then
            if yol ortamında bekleyen bir görev yok ve mevcut konum hedef
             değil ise then
                komşuları tanımla;
                if eğer komşuların arasında hedef yok ise then
                    tanımlı ama henüz keşfedilmemiş komşuları yol ortamı
                     için görev listelerine ekle ve seviyelerini etiketle;
                else if eğer komşulardan birisi hedef ise then
                    hedefin ve mevcut konumun seviyesini etiketle, mevcut
                     konumu keşfedildi ve hedefi bulundu olarak işaretle;
            else if eğer yol ortamında bekleyen bir görev var ve mevcut
             konum hedef değil ise then
                yol ortamında ilk sırada bekleyen görevi mevcut konum
                 olarak güncelle, kullanılan görevi görev listesinden sil, yeni
                 komşuları tanımla;
                if yeni komşuların arasında hedef yok ise then
                    tanımlı ama henüz keşfedilmemiş komşuları görev
                     listesine ekle ve seviyelerini etiketle;
                else if komşulardan birisi hedef ise then
                    hedefin ve mevcut konumun seviyesini etiketle, mevcut
                     konumu keşfedildi ve hedefi bulundu olarak işaretle;
        else if hedef bulunduysa then
            yol matrisini oluştur ve ilk vektörüne hedef noktası bilgilerini
             kaydet, harita sınır koruması için gerekli diziyi oluştur;
            for güzergah hedefseviyesi → baslangicseviyesi do
                mevcut komşuların seviyesini kayıt eden matrisi oluştur, en
                 küçük seviyeli komşuyu bul ve mevcut konum olarak
                 güncelle, bilgileri yol matrisinin sıradaki vektörüne kaydet;
            end
        else if hedefe ulaşıldıysa then
            yolu harita üzerinde çizdir ve ekranda göster;
    end
end
```

Algorithm 3: İkili ortamlar için DS-A* algoritması sözde kodu

3.3.4 Kompleks Ortamlar İçin DS-A* Algoritması

İkili ortamlar için geliştirdiğimiz DS-A* algoritmasının genişletilmesi ile elde edilen bu yöntem algoritmamızın birden fazla gezilebilirlik bölgesinde arama yapabilir hâle gelmesini sağlamaktadır. Böylelikle gezilebilir alanın her bir kısmını aynı verimle gezilebileceği varsayımından kaynaklanacak problemlerin önüne geçilmiş olacaktır. Bu problemlere örnek verecek olursak en basit olanı yol ve taşlık arazi örneği olur. Bu örneğe göre önümüzde 2 farklı güzergah seçeneği olduğunu varsayalım biri 120 km/saat hızla gidebileceğimiz 1.200 metre uzunlukta otoban diğeri ise 15 km/saat hızla gidebileceğimiz 1.199 metre taşlık bir arazi. Bu iki seçenek eğerki ikili mantıkla işleyen yöntemle değerlendirilirse algoritmamız 1.199 metre taşlık araziyi tercih edecek ve yol çok uzun sürecek, aracımız bu yolculukta daha fazla yıpranacak ve daha fazla enerji tüketimi gerçekleşecektir. Aynı iki durum çoklu mantıkla işleyen yöntemle değerlendirildiğinde ise algoritmamız taşlık arazide daha yavaş gideceğini hesaba katacak ve 1.200 metre uzunluktaki otobanı tercih edecektir. Böylelikle hem enerjiden, hem zamandan ve hem de aracın daha az yıpranmasından dolayı tasarruf sağlayacaktır.

```
if başlangıç verileri girilmemişse then
    başlangıç ve bitiş pozisyonunu, harita boyutunu, arazi katsayısını, görev
     ve seviye sayacını gir, kayıtçı matrisleri oluştur
else
    haritayı oku ve ön işlemden geçir;
    while 1 do
        if yol bulunmamışsa then
            if yol ortamında bekleyen bir görev yok ve mevcut konum hedef
             değil ise komşuları tanımla then
                if eğer komşuların arasında hedef yok ise then
                    tanımlı ama henüz keşfedilmemiş komşuları arazi veya yol
                     ortamı için görev listelerine ekle ve seviyelerini katsayısı
                     ile etiketle;
                else if eğer komşulardan birisi hedef ise then
                    hedefin ve mevcut konumun seviyesini katsayısı ile
                     etiketle, mevcut konumu keşfedildi ve hedefi bulundu
                     olarak işaretle;
            else if öncelikle yol ortamında, yoksa arazi ortamında bekleyen
             bir görev varsa ve mevcut konum hedef değil ise then
                yol ya da arazi ortamında ilk sırada bekleyen görevi mevcut
                 konum olarak güncelle, kullanılan görevi görev listesinden
                 sil, yeni komşuları tanımla;
                if yeni komşuların arasında hedef yok ise then
                    tanımlı ama henüz keşfedilmemiş komşuları görev
                     listesine ekle ve seviyelerini katsayısı ile etiketle;
                else if komşulardan birisi hedef ise then
                    hedefin ve mevcut konumun seviyesini katsayısı ile
                     etiketle, mevcut konumu keşfedildi ve hedefi bulundu
                     olarak işaretle;
        else if hedef bulunduysa then
            yol matrisini oluştur ve ilk vektörüne hedef noktası bilgilerini
             kaydet, harita sınır koruması için gerekli diziyi oluştur;
            for güzergah hedef seviyesi → baslangic seviyesi do
                mevcut komşuların seviyesini kayıt eden matrisi oluştur, en
                 küçük seviyeli komşuyu bul ve mevcut konum olarak
                 güncelle, bilgileri yol matrisinin sıradaki vektörüne kaydet;
            end
        else if hedefe ulaşıldıysa then
            yolu harita üzerinde çizdir ve ekranda göster;
    end
end
```

Algorithm 4: Kompleks ortamlar için DS-A* algoritması sözde kodu

4 Deneysel Sonuçlar

Bu bölümde geliştirdiğimiz güzergah planlama algoritmalarını çeşitli durumlarda çalıştırdık ve elde ettiğimiz sonuçları karşılaştırarak değerlendirdik. Öncelikle A* ve DS-A* algoritmalarımızın hem ikili ortamlar için olan versiyonlarını hem de kompleks ortamlar için olan versiyonlarını aynı koşullarda çalıştırdık. Burada test amaçlı oluşturduğumuz 100×100 piksel boyutunda yapay bir harita kullandık bu haritada üç farklı ortam bulunmaktadır, bu ortamlar siyah yani gezilemez alan, beyaz ya da açık gri yani serbest gezilebilir alan ve gri yani zor gezilebilir alan.

Test haritasındaki ilk deneyde A* algoritması için zıplama katsayısını 1, DS-A* algoritması için güvenlik katsayısını 3 ve arazi katsayısını ise her iki algoritmada da 3 verdik. Her iki algoritmanın her iki versiyonu için başlangıç konumunu $(1, 1)$ bitiş konumunu ise $(100, 98)$ verdik ve aldığımız sonuçlar A* algoritması için Şekil 4.1'deki gibi, DS-A* algoritması için ise Şekil 4.2'deki gibidir. Güzergahı A* ikili ortam algoritması Şekil 4.1(a)'daki gibi $0,66$ saniyede, A* kompleks ortam algoritması 4.1(b)'deki gibi 1,23 saniyede, DS-A* ikili ortam algoritması 4.2(a)'daki gibi $0,93$ saniyede ve DS-A* kompleks ortam algoritması ise 4.2(b)'deki gibi $0,91$ saniyede üretmiştir.

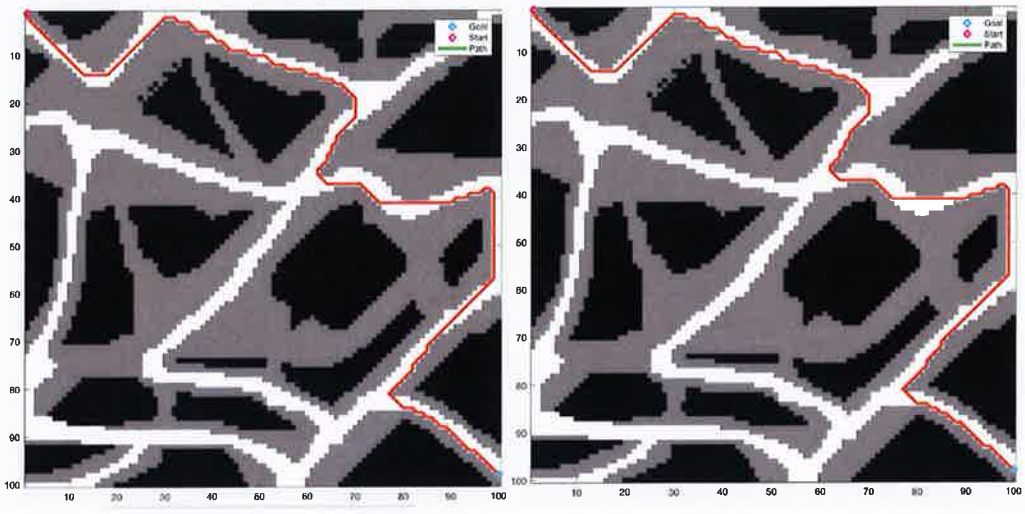
Test haritasında yaptığımız bu ilk deneyde A* algoritmasının ikili ortam algoritması en iyi performansı gösterirken, A* kompleks ortam algoritması ise en kötü sonucu vermiştir. DS-A* algoritması ise her iki ortam türü için de benzer sonuçlar vermiştir ve diğer iki algoritmanın birinden daha iyi diğerinden ise daha kötü bir sonuç vermiştir. Bunun sebebi şu şekilde açıklanabilir: A* algoritmasının ikili ortam için olanı sadece serbest gezilebilir kısımlar için değerlendirme yapmış ve gezilemez ve zor gezilebilir kısımları değerlendirmeye dahil etmediği için o kısımlara zaman harcamamıştır. A* kompleks ortam algoritması ise arama yaparken her bir adımda bulunduğu düğüm için serbest gezilebilir bölge de olsa zor gezilebilir bölge de olsa maliyet hesabı yapmaktadır ve bu maliyeti kayıt etmektedir, bu nedenle harcadığı süre artmaktadır.

DS-A* algoritmasında ise daha farklı bir yöntem kullanılmaktadır, öncelikle başlangıç konumu ile hedef konumu arasındaki kuş uçuşu uzaklığı "uzaklık" olarak kaydetmekte. Ardından tüm haritayı hızlı bir şekilde her bir adımda seviye değerini serbest gezilebilir bölgede birer birer zor gezilebilir bölgelerde ise bölgenin arazi katsayısı kadar artırarak etiketlemektedir. Burada her bir adımda seviye değerindeki artış miktarını o adımda karşılaşılan direnç olarak düşünebiliriz. Böylece serbest gezilebilir bölgede karşılaşılan direnç diğer bölgelerden az olacağı için hareketin akışı serbest gezilebilir bölgede kalacaktır.

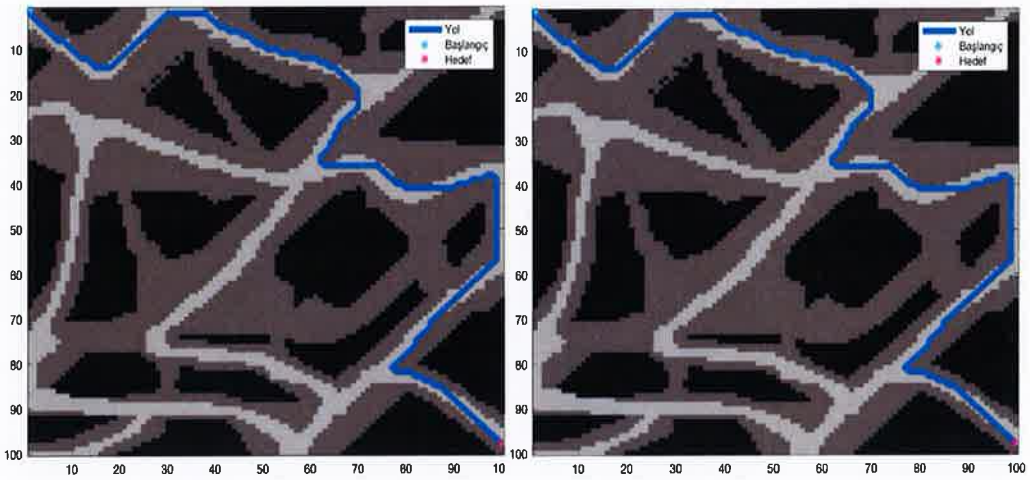
Algoritma etiketleme işlemiyle eş zamanlı olarak bir yandan da güzergahın başlangıç konumundan muvcut konuma kadarki geçtiği yolun uzaklığı ile mevcut konumdan hedef konumuna kadarki kuş uçuşu uzaklığın toplamını alıp "mevcut uzaklık" olarak kaydetmektedir. Her bir düğüm ilerleme sonrasında ise uzaklık değerinin güvenlik katsayısı ile çarpımını mevcut uzaklık değeri ile kıyaslamaktadır. Ne zaman mevcut uzaklık değeri uzaklık değeriyle güvenlik katsayısı çarpımına ulaşırsa orada artık serbest gezilebilir bölgedeki güzergahın uzun olacağına karar vermektedir. Bu karar sonrası zor gezilebilir bölgede de serbest gezilebilir bölgede olduğu gibi her bir adımda seviye değerini birer birer artırmaya devam etmektedir. Böylece karşılaşılan direnç her iki bölgede de eşitlenmektedir. Karşılaşılan direncin eşitlenmesiyle eğer zor gezilebilir bölgeden geçebilecek daha kısa bir güzergah varsa hareketin akışı o yöne yönelmektedir.

Akış yönü belirlendikten sonra ise algoritma harita üzerinde hedeften başlangıç konumuna doğru yani güzergahı tersten başlatarak akışın dışına çıkmadan güzergah adımlarını belirlemekte ve maliyet te bu aşamada çıkarılmaktadır. Böylelikle güzergaha dahil edilmeyecek bir düğüm için maliyet hesabı yapılmamakta ve ekstradan tüm düğümler için etiketleme maliyeti olmasına rağmen A* kompleks ortam algoritmasında harcadığından daha az zaman harcanmakta ama yine etiketleme maliyetinden dolayı A* ikili ortam algoritmasında harcadığı zamandan daha fazla zaman harcanmaktadır.

Burada akla gelecek sorulardan bir tanesi ise hepsinin aynı güzergahı belirleme sebebidir. Hepsinin aynı güzergahı belirlemesinin sebebi aralarında kıyaslama yapılabilmesi için aynı koşulların oluşturulmasıdır ve aynı koşulların oluşturulabilmesi için DS-A* algoritmasında güvenlik katsayısı 3 alınarak ve her iki algoritmada da arazi katsayıları 3 alınarak güzergahların hepsinin serbest gezilebilir bölgede kalması sağlanmıştır.



Şekil 4.1: (a) A* ikili ortam algoritmasının kompleks ortam haritasındaki test sonucu harcanan süre: 0,66 sn, (b) A* kompleks ortam algoritması test sonucu harcanan süre: 1,23 sn

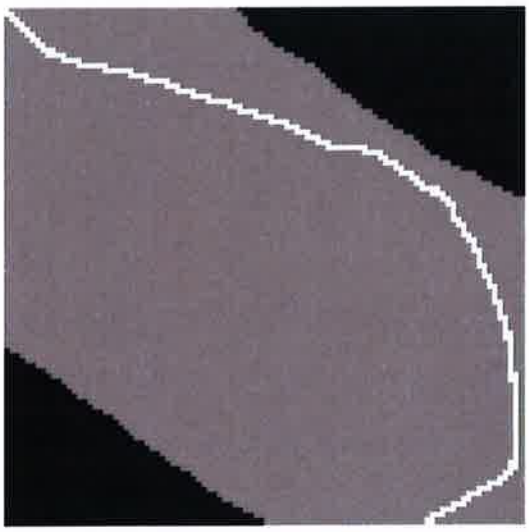


Şekil 4.2: (a) DS-A* ikili ortam algoritmasının kompleks ortam haritasındaki test sonucu harcanan süre: 0,93 sn, (b) DS-A* kompleks ortam algoritması test sonucu harcanan süre: 0,99 sn

Şimdi ise gerçek hayatta kullanım alanı oldukça kısıtlı olan ikili ortam algoritmalarımızı bir kenara bırakarak A* kompleks ortam algoritması ve DS-A* kompleks ortam algoritması ile çalışmaya devam edeceğiz. Burada çalışmamıza Şekil 4.3'teki gibi 100×100 piksel boyutunda; engel ortamı, serbest gezilebilir ortam ve zor gezilebilir ortam içeren yapay bir test haritası ile devam edeceğiz.

Harita ortamı	A*	DS-A*
İkili ortam	0,66 sn	0,93 sn
Kompleks ortam	1,23 sn	0,99 sn

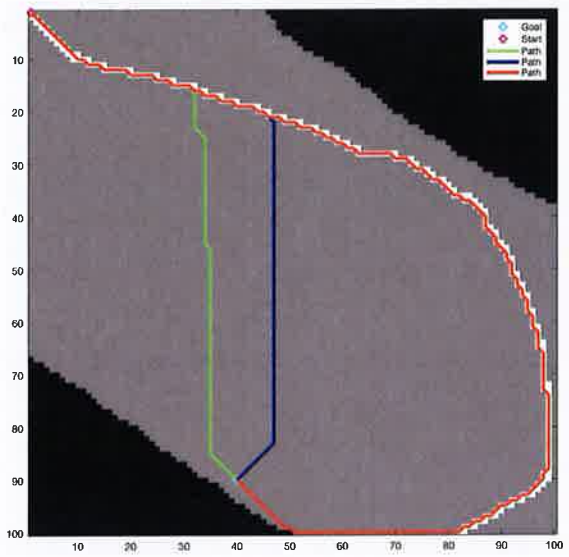
Tablo 4.1: A* ve DS-A* algoritmalarının farklı harita ortamlarında güzergah planlama süreleri



Şekil 4.3: Farklı ortamlar arası geçiş kontrolü için 2. test haritası

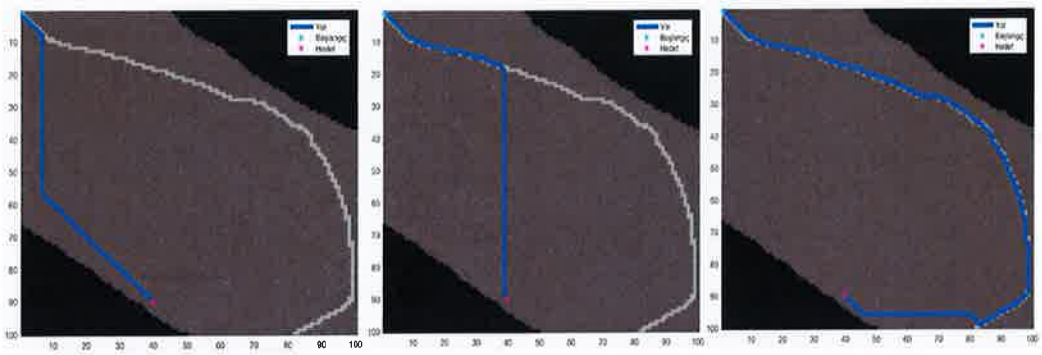
Şekil 4.3'teki haritayı kullanarak hem A* kompleks ortam algoritması ile hemde DS-A* algoritması ile $(1,1)$ düğümöünden $(40,90)$ düğümüne güzergah aramaktayız. Hedef düğümümüz zor gezilebilir alanda bir noktaya denk gelmektedir. A* komplex ortam algoritmamızda bütün güzergahlar için zıplama katsayısını 1 seçiyoruz ve arazi katsayısını yeşil güzergah için 3, mavi güzergah için 5 ve kırmızı güzergah için 10 seçiyoruz. Aramayı başlattığımız zaman Şekil 4.4'deki sonucu 1,42 saniyede elde ediyoruz. Görüldüğü üzere yeşil güzergahta arazi katsayısı 3 olması nedeni ile serbest gezilebilir ortam ile zor gezilebilir ortam arasındaki maliyet farkı mavi ve kırmızı güzergahlara göre daha az olduğu için güzergah serbest gezilebilir ortamdan zor gezilebilir ortama çabuk geçiş yapmıştır. Arazi katsayısı 5

seçilen mavi güzergah hedefe daha yakın bir yerden geçiş yaparken arazi katsayısı 10 olan kırmızı güzergah zor gezilebilir ortama en uç düğümden yani hedefe en yakın düğümden geçiş yapmıştır.



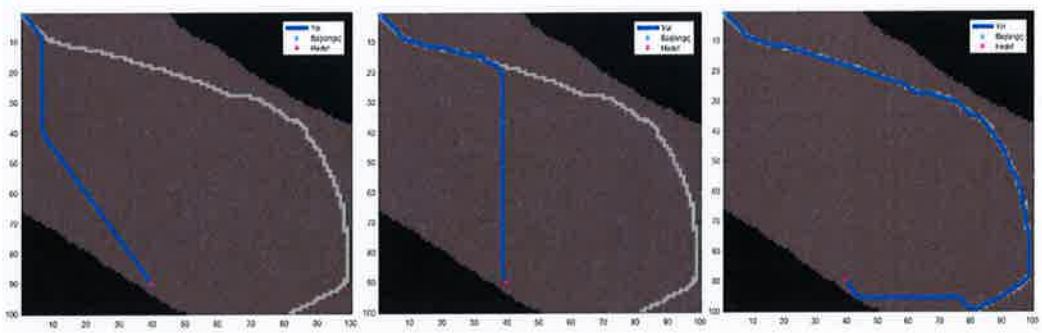
Şekil 4.4: 2. Test haritası A* kompleks ortam test sonucu harcanan süre: 1,42 sn

Şekil 4.5'te ise DS-A* kompleks ortam algoritması ile hedefe ulaşılmaya çalışılmıştır. DS-A* algoritmasının yapısının A* algoritmasından farklı olması nedeni ile serbest gezilebilir ortamdan zor gezilebilir ortama geçiş düğümünü arazi katsayısı değil güvenlik katsayısı belirlemektedir. Bu durumun sebebi DS-A* algoritmasında arazi katsayısının görevinin farklı zorluklardaki zor gezilebilir ortamları birbirinden ayırt etmek olmasıdır. Bu durumda bizim güvenlik katsayısına farklı değerler vererek devam etmemiz gerekmektedir. Buna göre güvenlik katsayıları sırası ile a) 1,05, b) 1,3 ve c) 2,8 seçilmiştir. Görüldüğü üzere arazi katsayısı arttıkça güzergahın serbest gezilebilir bölgeden zor gezilebilir bölgeye geçişi gecikmektedir. Güvenlik katasayısının 1,05 olduğu yerde güzergah oluşturulurken oluşturulan güzergah uzunluğu başlangıç konumundan hedef konumuna kuş uçuşu uzaklığın yüzde beş fazlasını aşınca gidilen yolun uzun olduğuna karar verilmiş ve hareketin akışı serbest gezilebilir bölgeden zor gezilebilir bölgeye geçiş yapmıştır. Aynı şekilde güvenlik katsayısının 1,3 olduğu yerde yüde otuzu aşınca, 2,8 olduğu yerde ise yüzde yüz sekseni aşınca zor gezilebilir alana geçiş yapmıştır.



Şekil 4.5: 2. Test haritası DS-A* kompleks ortam algoritması ile 3 × 3 piksel boyutlu yönelim matrisinde test sonucu, (a) harcanan süre: 1,05 sn, (b) harcanan süre: 0,96 sn, (c) harcanan süre: 0,99 sn

Burada dikkat etmemiz gereken bir konu daha var, Şekil 4.5'teki örnekte hedef bulunduktan sonra yol oluşturulurken 3 × 3 piksel boyutunda bir yönelim matrisi kullanılmıştır. Buna göre yol ya çapraza ya da üst, alt, sağ ve sol gibi ana yönlere ilerlemiştir ve Şekil 4.5'teki ilk güzergahtaki gibi çapraz bir yol belirlenmiş ve daha uzun bir güzergah üretilmiştir. Bu problemin çözümü için 5 × 5 piksel ve 7 × 7 piksel boyutlu yönelim matrislerinin kullanıldığı testler gerçekleştirilmiştir. Bu test sonuçlarında matris boyutu büyüdükçe güzergahın farklı açılardaki ara yönlere de ilerleyebilir hale geldiği ve daha iyi sonuçların elde edildiği görülmüştür. Şekil 4.6'da güvenlik katsayıları sırası ile a) 1,05, b) 1,3 ve c) 2,8 seçilerek ve yol oluşturulurken 7 × 7 piksel boyutlu yönelim matrisi kullanılarak elde edilen güzergahlar görülmektedir. Burada Şekil 4.5(a) ve Şekil 4.6(a)'yı kıyasladığımız zaman 7 × 7 piksel boyutlu yönelim matrisi kullanılarak elde edilen güzergahın daha kısa olduğu görülmektedir.

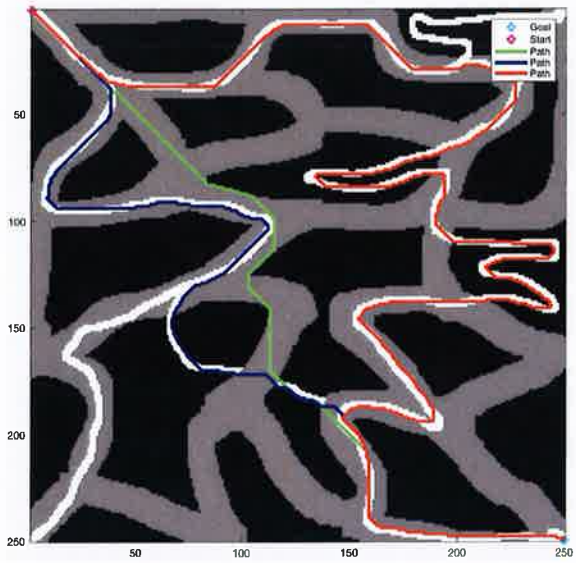


Şekil 4.6: 2. Test haritası DS-A* kompleks ortam algoritması ile 7 × 7 piksel boyutlu yönelim matrisinde test sonucu, (a) harcanan süre: 1,04 sn, (b) harcanan süre: 0,96 sn, (c) harcanan süre: 0,98 sn

Test etmemiz gereken bir başla konu ise A* algoritması ile DS-A* algoritmasının daha büyük boyutlu haritalardaki hızıdır. Bunun için Şekil 4.7'de görüldüğü gibi 250 × 250 piksel boyutlu yeni bir yapay test haritası daha oluşturduk.

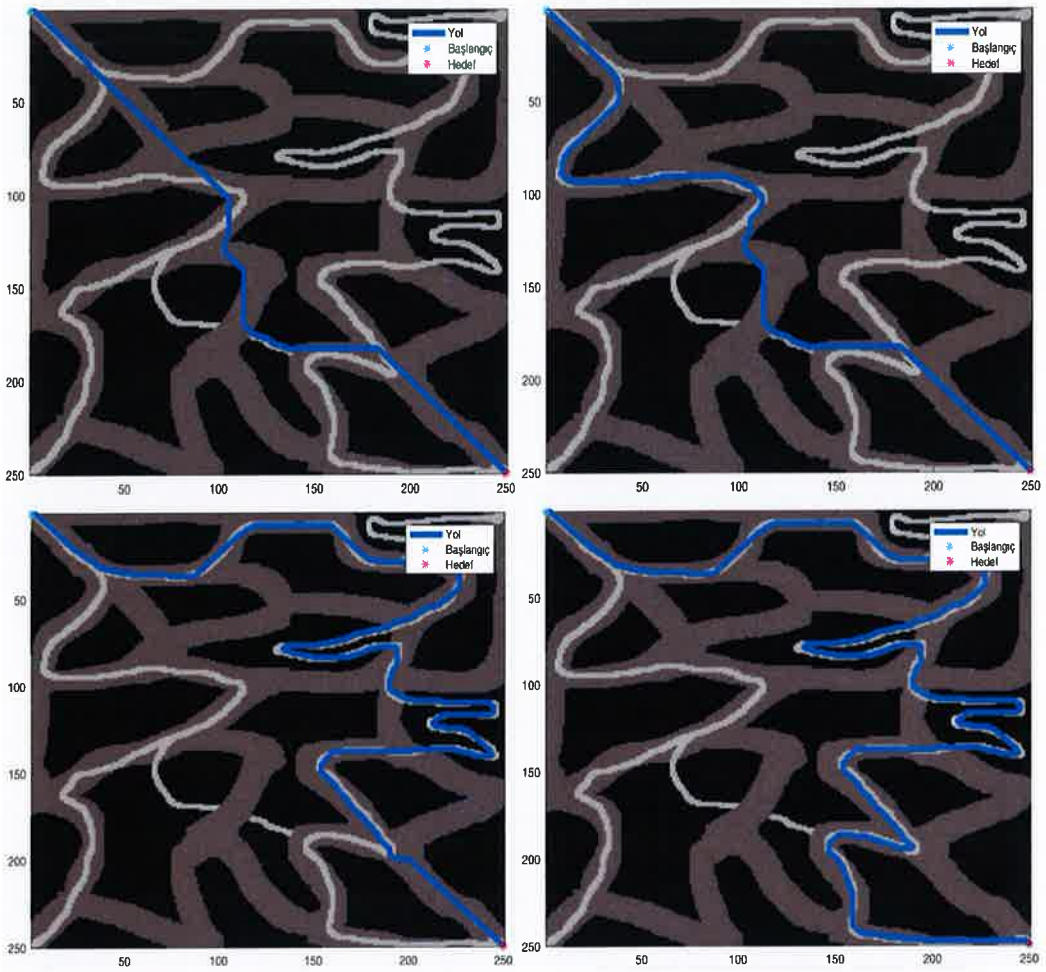
Şekil 4.7: 250 × 250 piksel boyutlu 3. test haritası

Oluşturduğumuz 250 × 250 piksel boyutlu yeni test haritasında öncelikle Şekil 4.8'de görüldüğü üzere A* algoritması ile güzergah planlama yaptık. Burada arazi katsayılarını sırası ile yeşil güzergah için 2, mavi güzergah için 5 ve kırmızı güzergah için 15 seçtik. Planlama süresi 100 × 100 piksel boyutlu haritada 1, 23 sn iken burada 9, 54 sn olmuş ve 100 × 100 piksel boyutlu haritaya kıyasla oldukça yüksek çıkmıştır.



Şekil 4.8: 3. Test haritası, A* kompleks ortam algoritması ile 250 × 250 piksel boyutlu haritada test sonucu, arazi katsayıları yeşil için 2, mavi için 5 ve kırmızı için 15'tir. Harcanan süre: 9,54 sn

Şekil 4.8'de A* algoritması için yapmış olduğumuz testi aynı haritayı kullanarak Şekil 4.9'da DS-A* algoritması için tekrarladık. Burada güvenlik katsayılarını Şekil 4.9(a) için (sol üst) 1,05, Şekil 4.9(b) için (sağ üst) 1,5, Şekil 4.9(c) için (sol alt) 3 ve Şekil 4.9(d) için 5 seçtik. DS-A* algoritması ile 100 × 100 piksel boyutlu haritada daha önce 0.99 saniyede elde ettiğimiz sonucu 250 × 250 piksel boyutlu haritada güzergahın zor gezilebilir bölgeye hemen geçiş yaptığı ilk güzergahta 1,32 saniyede, tamamen serbest gezilebilir bölgede ilerleyen son güzergahta ise 1,03 saniyede elde ettik. Görüldüğü üzere harita boyutunun büyümesi ile DS-A* algoritmasının sonucu elde etme süresi çok fazla etkilenmez iken A* algoritmasının sonucu elde etme süresi büyük oranda etkilenmiştir.

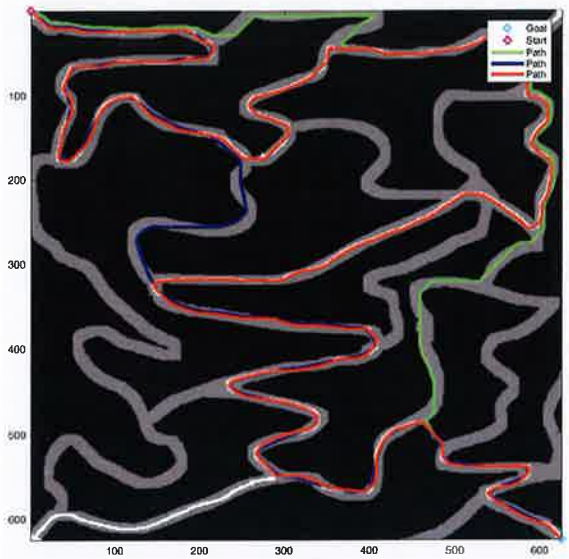


Şekil 4.9: 3. Test haritası DS-A* kompleks ortam algoritması ile 250 × 250 piksel boyutlu haritada test sonucu, (a) harcanan süre: 1,32 sn, (b) harcanan süre: 1,31 sn, (c) harcanan süre: 1,17 sn, (d) harcanan süre: 1,03 sn

Harita boyutunun büyümesi ile DS-A* algoritmasının sonucu elde etme süresi çok fazla etkilenmez iken A* algoritmasının sonucu elde etme süresi büyük oranda etkilenir tezi gerçekten her zaman doğru mudur yoksa tesadüfen mi denk gelmiştir. Bu sorunun cevabı için Şekil 4.10'da görüldüğü gibi 625 × 625 piksel boyutlu yeni bir test haritası daha oluşturduk.

Şekil 4.10: 625 × 625 piksel boyutlu 4. test haritası

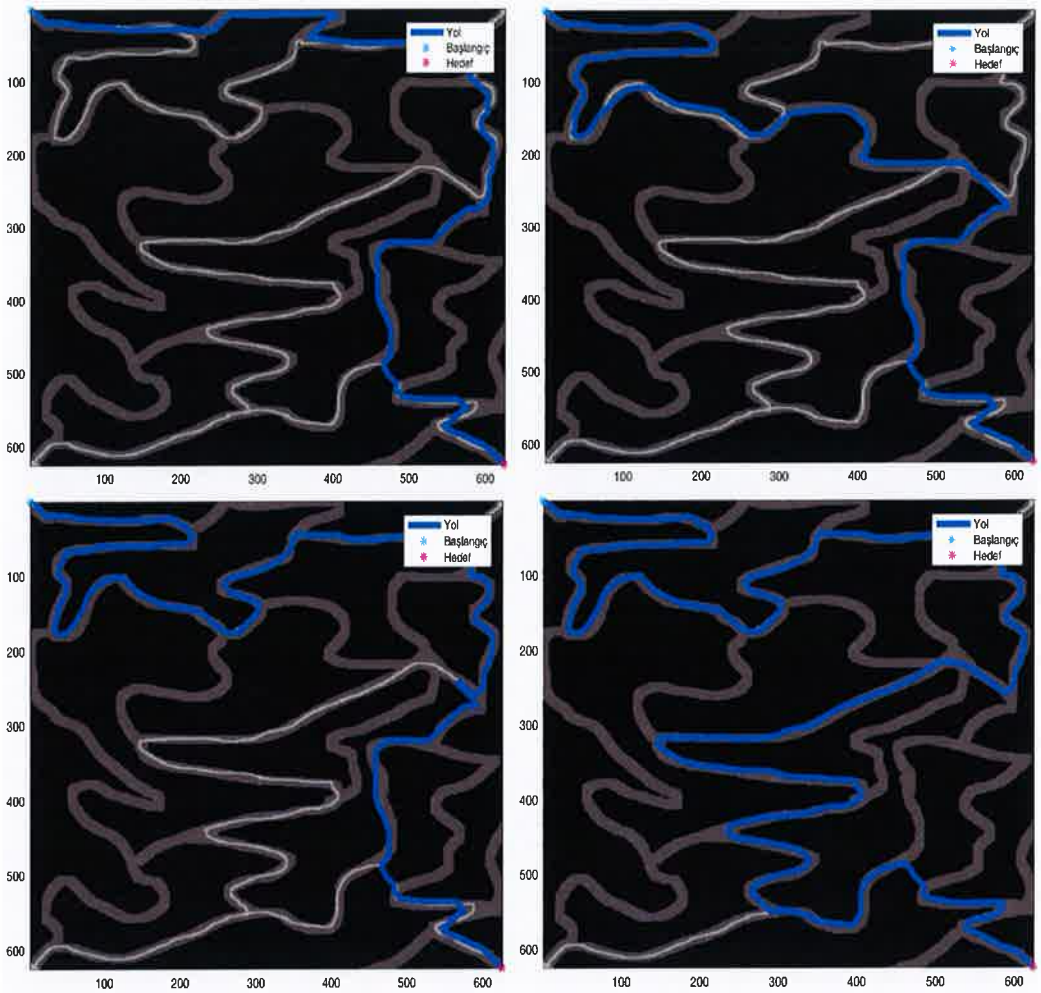
Oluşturduğumuz 625 × 625 piksel boyutlu yeni test haritasında öncelikle Şekil 4.11'de görüldüğü gibi A* algoritması ile güzergah planladık. Bu planlamada arazi katsayılarını sırası ile yeşil güzergah için 2, mavi güzergah için 3 ve kırmızı güzergah için 15 seçtik. Planlama sonucunu ise 100 × 100 piksel boyutlu ve 250 × 250 piksel boyutlu haritalardakine kıyasla aşırı uzun bir sürede olan 153, 44 saniyede elde ettik.



Şekil 4.11: 4. Test haritası, A* kompleks ortam algoritması ile 625 × 625 piksel boyutlu haritada test sonucu, arazi katsayıları yeşil için 2, mavi için 3 ve kırmızı için 15'tir. Harcanan süre: 153, 44 sn

Ardından 625 × 625 piksel boyutlu aynı harita üzerinde DS-A* algoritması ile de güzergah planladık. Bu planlamada ise güvenlik katsayılarını 250 × 250 piksel boyutlu haritada olduğu gibi Şekil 4.12(a) için (sol üst) 1, 05, Şekil 4.12(b) için (sağ üst) 1,5, Şekil 4.12(c) için (sol alt) 3 ve Şekil 4.12(d) için 5 seçtik. Sonuçları ise 250 × 250 piksel boyutlu haritaya kıyasla kısmen daha geç elde etmekle birlikte A* algoritması kadar aşırı bir fark gözlemlemedik. 625 × 625 piksel boyutlu haritada güzergahın zor gezilebilir bölgeye hemen geçiş yaptığı Şekil 4.12(a)'daki güzergahta 2, 32 saniyede, tamamen serbest gezilebilir bölgede ilerleyen Şekil 4.12(d)'deki güzergahta ise 1, 25 saniyede elde ettik.

Tezi pekiştirmek amacıyla Şekil 4.10'dakine benzer bir şekilde Şekil 4.13'teki gibi 1563 × 1563 piksel boyutunda bir test haritası daha oluşturduk. Oluşturduğumuz bu test haritasında arazi katsayılarını yeşil güzergah için 2, mavi güzergah için 5 ve kırmızı güzergah için 15 seçerek A* algoritmasıyla güzergah planladık. Bu güzergah planlama işlemininde Şekil 4.14'teki sonucu 4524,48 saniyede elde ettik. Aynı harita üzerinde DS-A* algoritması ile 1,05 güvenlik katsayılı güzergahı Şekil 4.15(a)'daki gibi 8,24 saniyede, 1,5 güvenlik katsayılı güzergahı Şekil 4.15(b)'deki gibi 8,09 saniyede, 3 güvenlik katsayılı güzergahı Şekil 4.15(c)'deki gibi 2,69 saniyede ve 5 güvenlik katsayılı güzergahı Şekil 4.15(d)'deki gibi 2,71 saniyede elde ettik.

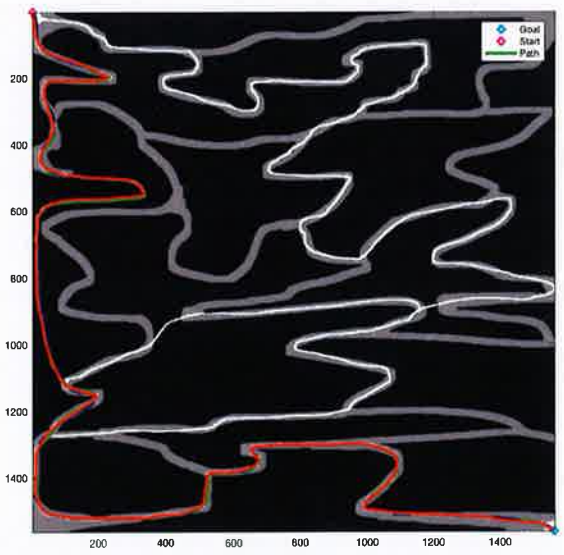


Şekil 4.12: 4. Test haritası DS-A* kompleks ortam algoritması ile 625×625 piksel boyutlu haritada test sonucu, (a) harcanan süre: $2,31$ sn, (b) harcanan süre: $2,32$ sn, (c) harcanan süre: $2,09$ sn, (d) harcanan süre: $2,56$ sn

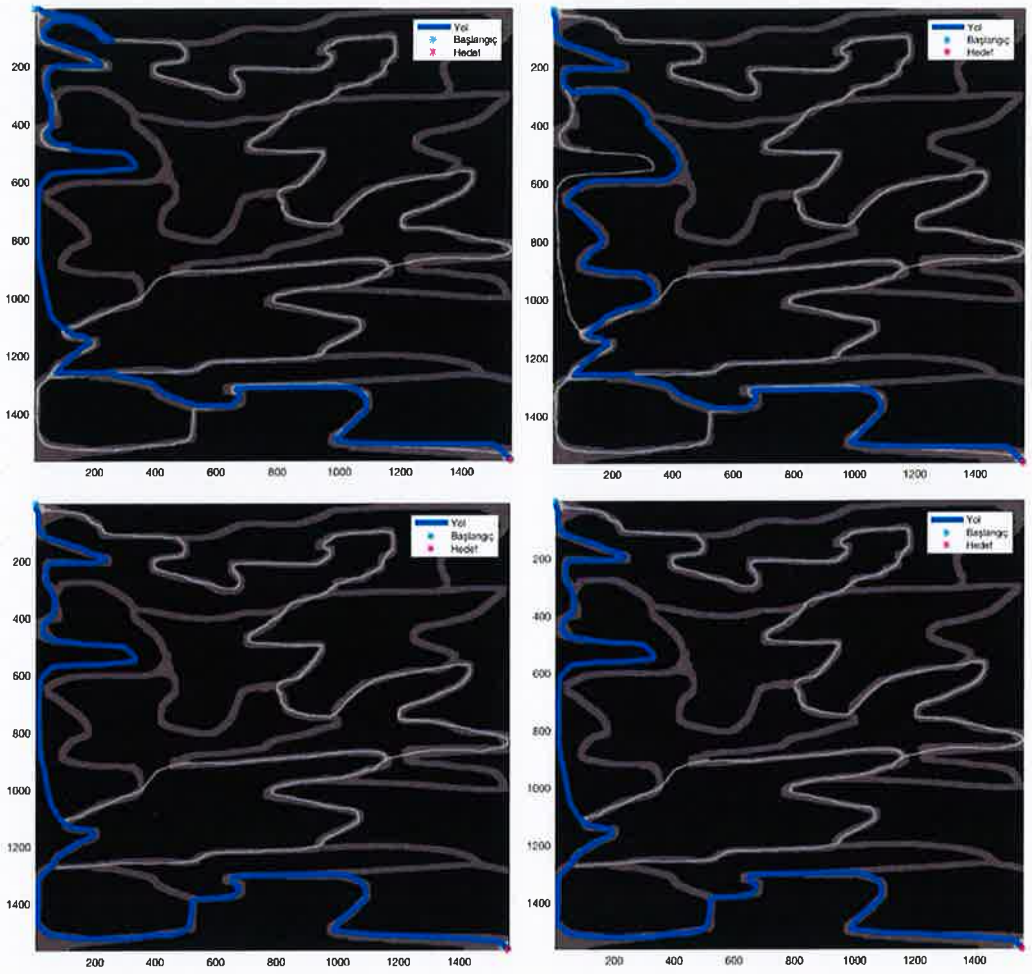
Buraya kadarki elde ettiğimiz sonuçlardan anlaşılacağı üzere A* algoritması harita boyutu büyüdükçe işlem süresi hızlıca artmakta ve artık sonuç elde etmesi aşırı derecede zorlaşmaktadır. Fakat DS-A* algoritmasında haritanın boyutundaki büyümeyle işlem süresindeki artış A* algoritmasına kıyasla çok düşük kalmaktadır. Bu sonuç DS-A* algoritmasının güçlü özelliklerinden biri iken DS-A* algoritmasının zayıf özelliklerinden bir tanesi ise Şekil 4.12(c)'de yaklaşık $(560, 250)$ konumunda ve Şekil 4.15(a)'da yaklaşık $(260, 140)$ konumunda görüldüğü üzere güzergah serbest gezilebilir bölgede ilerlerken ilerlediği yolun uzun olduğuna karar verip zor gezilebilir bölgeye geçiş yapması esnasında bir miktar geri giderek farklı yola sapmasıdır. Burada kısa bir me-

safe yol gereksiz yere gidilip tekrar dönülmüştür. Bu durum seçilen güvenlik katsayısı değeriyle alakalıdır ve her harita yapısında gerçekleşmeyebilir fakat bazı haritalarda özel bazı güvenlik katsayısı değerlerinde bu örnektekinden daha uzun gidip dönme işlemi gerçekleşme ihtimali vardır ve bu hataların giderilmesi için ayrı bir filtre algoritması geliştirilmesi gerekmektedir.

Şekil 4.13: 5. Test haritası, boyut: 1563 × 1563 piksel



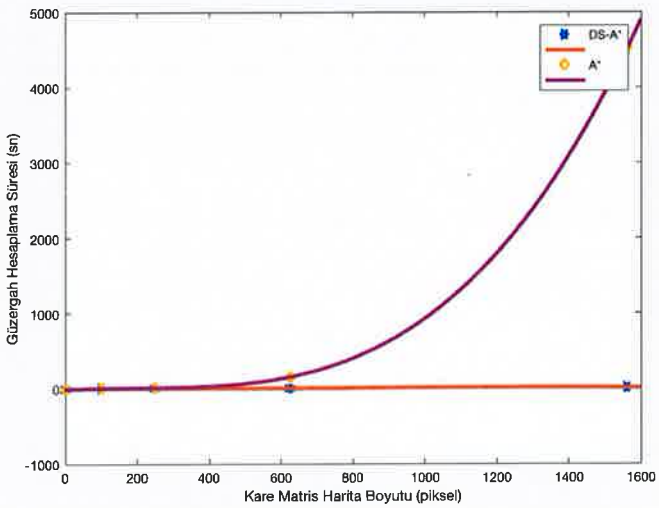
Şekil 4.14: 5. Test haritası, A* kompleks ortam algoritması ile 1563 × 1563 piksel boyutlu haritada test sonucu, arazi katsayıları yeşil için 2, mavi için 5 ve kırmızı için 15'tir. Harcanan süre: 4524, 48 sn



Şekil 4.15: 5. Test haritası DS-A* kompleks ortam algoritması ile 1563×1563 piksel boyutlu haritada test sonucu, (a) harcanan süre: $8,24$ sn, (b) harcanan süre: $8,09$ sn, (c) harcanan süre: $2,69$ sn, (d) harcanan süre: $2,71$ sn

Harita boyutu	A* Kompleks ortam	DS-A* kompleks ortam
100 × 100 piksel	1, 23 sn	0, 93 sn
250 × 250 piksel	9, 54 sn	1, 32 sn
625 × 625 piksel	153, 44 sn	2, 56 sn
1563 × 1563 piksel	4524, 48 sn	8, 24 sn

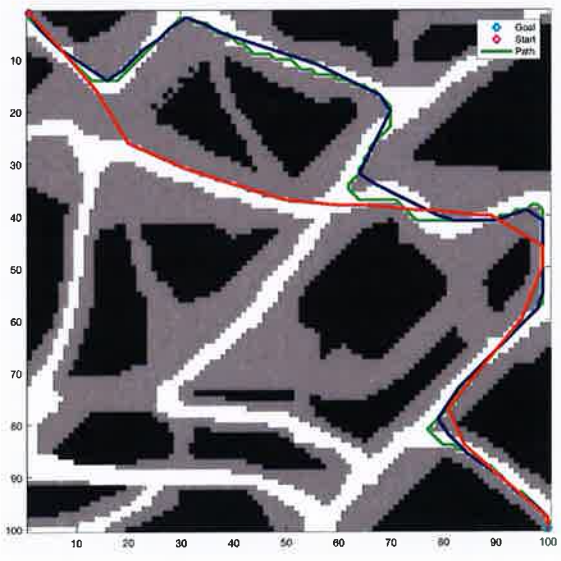
Tablo 4.2: A* ve DS-A* algoritmalarının farklı harita boyutlarında güzergah planlama süreleri



Şekil 4.16: Harita boyutunun artışıyla A* ve DS-A* algoritmalarının hesaplama sürelerindeki değişim

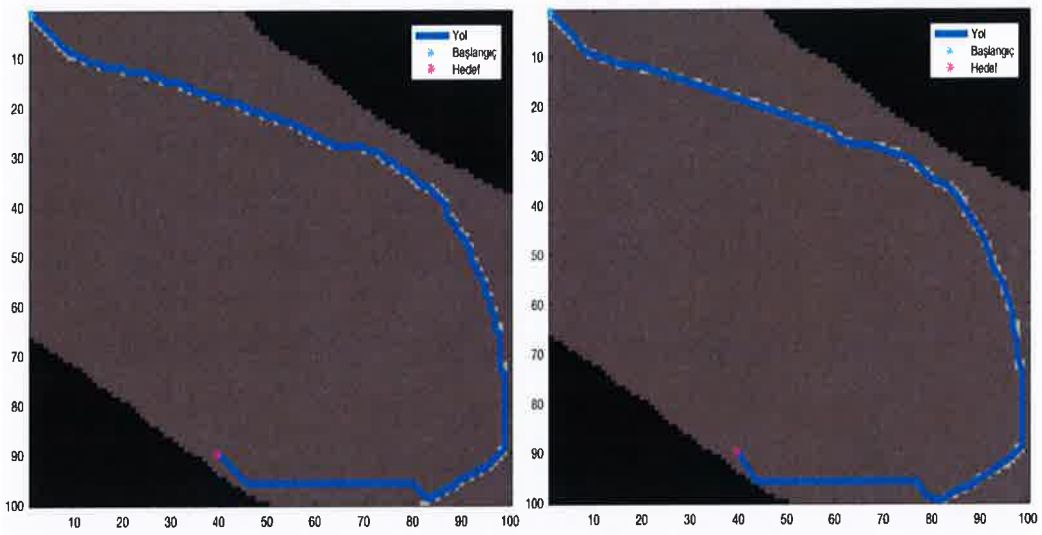
Burada önemli konulardan bir taneside zıplama katsayısıdır. Zıplama katsayısı A* algoritmasında algoritma ile üretilen güzergahın daha düzgün (zik zak yapmayan) bir şekilde üretilmesini sağlayan bir özelliktir. Bu özellik güzergah oluşturulurken incelenen düğümlerin belirli aralıkla atlanarak in-

celenmesi ile elde edilir. Fakat bu katsayı aşırı büyük seçilirse güzergahın engelin üzerinden geçirilmesine neden olur ve oluşturulan güzergahtan gerçek araç geçemeyebilir. Bu nedenle aracın geçebileceği engellerden daha büyük bir zıplama katsayısı tercih edilmemelidir. Optimum seçilen bir zıplama katsayısı daha pürüzsüz bir güzergah üretimine olanak sağlar. Şekil 4.17'te A* kompleks ortam algoritması ile elde edilmiş farklı zıplama katsayılarına sahip 3 güzergah görmekteyiz, bu güzergahların üçü de eşit ve 10 arazi katsayısına sahiptir, zıplama katsayıları ise yeşil için 1, mavi için 3 ve kırmızı için 10'dur. Yeşil ve mavi güzergahı kıyasladığımız zaman yeşil güzergah tamamen serbest gezilebilir ortamda gezinirken zikzaklar yapmış, mavi ise zikzak yapmamış ama bazı köşe bölgelerde zor gezilebilir alanın üzerinden atlamıştır. Yeşil güzergah ile aynı arazi katsayısına sahip olan mavi güzergah aslında burada tamamen serbest gezinme bölgesinde olduğunu zannetmektedir fakat zıplama katsayısının değeri nedeniyle köşelerdeki atlamaları görmemektedir. Diğer iki güzergahla aynı arazi katsayısına sahip olan kırmızı güzergah ise 10 zıplama katsayısına sahiptir ve aşırı büyük bir zıplama katsayısı seçilmesi nedeni ile aslında farkında olmadan gittiği yoldan zor gezilebilir araziden geçerek karşı yola atlamıştır ve bunun farkında değildir. Bu nedenlerle burada düşük bir zıplama katsayısı seçildiği taktirde hiç seçilmediği duruma kıyasla daha iyi bir güzergah ortaya çıkacak ama büyük bir zıplama katsayısı hatalara neden olacaktır.



Şekil 4.17: 1. Test haritası, zıplama katsayısı etkisini gösteren güzergah planlama

A* algoritmasındaki zıplama katsayısının işlevini gören DS-A* algoritmasındaki özellik ise yönelim matrisidir. Çalışmamızda daha önce bahsedildiği üzere 3 × 3 piksel, 5 × 5 piksel ve 7 × 7 piksel boyutlu matrisleri kullanmıştık ve matris boyutu büyüdükçe sadece çapraz ve ana yönlere hareketle sınırlı kalmayıp güzergahın farklı açılardaki ara yönlere de ilerleyebildiğini söylemiştik. Güzergahın farklı açılardaki ara yönlere de ilerleyebilmesiyle DS-A* algoritmasının ürettiği güzergah yönelim matrisinin boyutunun artmasıyla daha da pürüzsüz hale gelmektedir. Şekil 4.18'te sol tarafta 3×3 piksel boyutlu yönelim matrisi sağ tarafta ise 7×7 piksel boyutlu yönelim matrisi kullanılmıştır. Grafikte de görüldüğü üzere sağ taraftaki güzergah soldakine göre daha pürüzsüz üretilmiştir.



Şekil 4.18: 2. Test haritası, seviye kontrolünde kullanılan matrisin zik-zak yapısına etkisi

5 Sonuç

Otonom araçların hedeflerine mümkün olan en kısa yoldan daha güvenli bir şekilde ulaşabilmesi amacıyla mevcut güzergah planlama algoritmaları gözden geçirilmiştir. Mevcut algoritmalar arasından normalde ikili ortamda çalışan A* algoritması geliştirilerek kompleks ortamda çalışır hale getirilmiştir. Bunun dışında ise mevcut algoritmalardan level set algoritması, Dijkstra algoritması, *fast marching* algoritması ve A* algoritmasının bazı özellikleri toplanarak DS-A* algoritması geliştirilmiştir. Çalışmanın sonunda geliştirilen A* algoritmasının ve DS-A* algoritmasının kompleks ve ikili ortamlar için olan versiyonları karşılaştırılmıştır. Karşılaştırmada her iki algoritmanın da gerçek hayatta kullanım bakımından daha elverişli olan kompleks ortam versiyonları üzerinde daha fazla durulmuştur. DS-A* algoritmasının A* algoritmasına kıyasla dikkat çeken önemli özelliklerinden bir tanesi güzergah üretme hızının haritanın boyutundan çok fazla etkilenmemesi ve kompleks ortamlarda A* algoritmasından daha hızlı hesaplayabilmesidir. Fakat her iki algoritmanında kompleks ortamlar için olan versiyonları farklı arazi koşullarına uyum sağlayabilme yetenekleri nedeni ile mevcut algoritmalardan daha iyi sonuç vermektedir.

Kaynaklar

[1] S. Garrido J.V. Gómez, A. Lumbier and L. Moreno. Planning robot formations with fast marching square including uncertainty conditions. *Robotics and Autonomous Systems*, 61(2):137–152, 2013.

[2] M. Malfaz S. Garrido and D. Blanco. Application of the fast marching method for outdoor motion planning in robotics. *Robotics and Autonomous Systems*, 61(2):106–114, 2012.

[3] V. Milanés D. González, J. Pérez and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[4] F.M. Marchese. Multiple mobile robots path-planning with mca. International Conference on Autonomic and Autonomous Systems (ICAS'06), IEEE, 2006.

[5] S.S. Lim J.Y. Hwang, J.S. Kim and K.H. Park. A fast path planning by path graph optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33(1):121–129, 2003.

[6] A. Stentz. Optimal and efficient path planning for partially-known environments. Proceedings of the 1994 IEEE International Conference on Robotics and Automation, IEEE, 1994.

[7] D. Ferguson and A. Stentz. Using interpolation to improve path planning: The field d* algorithm. 2005.

[8] S. Koenig A.Nash, K.Daniel and A. Felner. Theta*: Any-angle path planning on grids.

[9] G. Gordon A. Stentz M. Likhachev, D. Ferguson and S. Thrun. Anytime search in dynamic graphs.

[10] J. Keller A. Kushleyev D. Lee A. Stewart P. Vernaza J. Derenick J. Spletzer J. Bohren, T. Foote and B. Satterfield. Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

[11] C. Anderson A. Broggi P. Grisleri P. P. Porta P. Zani Y.-L. Chen, V. Sundareswaran and J. Beck. Terramax: Team oshkosh urban robot. *Journal of Field Robotics*, 25(10):841–860, 2008.

[12] R. Pillat G. Stein B. J. Patz, Y. Papelis and D. Harper. A practical approach to robotic design for the darpa urban challenge. *Journal of Field Robotics*, 25(8):528–566, 2008.

[13] S. B. Karumanchi S. J. Anderson and K. Iagnemma. Constraint-based planning and control for safe, semi-autonomous operation of vehicles. *IEEE Intelligent Vehicles Symposium (IV)*, page 383–388, 2012.

[14] R. Faruque M. Fleming C. Terwelp C. Rein-holtz D. Hong A. Wicks T. Alberi D. Anderson et al. A. Bacha, C. Bauman. Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.

[15] R. Kala and K. Warwick. Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization. *Journal of Intelligent Robotic Systems*, 72(3-4):559–590, 2013.

[16] D. Bagnell C. Baker R. Bittner M. Clark-J. Dolan D. Duggins T. Galatali C. Geyer et al. C. Urmson, J. Anhalt. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[17] T. M. Howard D. Ferguson and M. Likhachev. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12):939–960, 2008.

[18] M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8):933–945, 2009.

[19] S. Bhat H. Dahlkamp D. Dolgov S. Et-tinger D. Haehnel T. Hilden G. Hoffmann B. Huhnke et al. M. Montemerlo, J. Becker. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[20] M. Montemerlo D. Dolgov, S. Thrun and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[21] B. Pitzer M. Werling T. Gindele D. Jagzent-J. Schro der M. Thuy M. Goebl F. v. Hundelshausen et al. S. Kammel, J. Ziegler. Team annieway's autonomous system for the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[22] M. Werling J. Ziegler and J. Schroder. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. *IEEE Intelligent Vehicles Symposium*, page 787–791, 2008.

[23] A. Stentz ve S. Thrun D. Ferguson. Pao for planning with hidden state. *EEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 3:2840–2847, 2004.

[24] D. Huttenlocher F.-R. Kline A. Nathan S. Lupashin J. Catlin B. Schimpf P. Moran N. Zych E. Garcia M. Kurdziel I. Miller, M. Campbell and H. Fujishima. Team cornell's skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527, 2008.

[25] M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. International Symposium on Artificial Intelligence, Robotics, and Automation in Space, 2005.

[26] A. Kelly T. M. Howard, C. J. Green and D. Ferguson. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325–345, 2008.

[27] J. Dolan M. McNaughton, C. Urmson and J.-W. Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. *IEEE International Conference on Robotics and Automation (ICRA)*, page 4889–4895, 2011.

[28] J. Dolan T. Gu, J. Snider and J. woo Lee. Focused trajectory planning for autonomous on-road driving. *IEEE Intelligent Vehicles Symposium (IV)*, page 547–552, 2013.

[29] J. Ziegler and C. Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, page 1879–1884, 2014.

[30] J. Ziegler M. Werling, S. Kammel and L. Grll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 2011.

[31] A. Kushleyev and M. Likhachev. Time-bounded lattice for efficient planning in dynamic environments. *IEEE International Conference on Robotics and Automation ICRA'09*, page 1662–1668, 2009.

[32] M. Rufli W. Derendarz H. Grim-mett P. Muhlfellner S. Wonneberger J. Timpner S. Rottmann B. Li B. Schmidt T. Nguyen E. Cardarelli S. Cattani S. Bruning S. Horstmann M. Stellmacher H. Mielenz K. Koser M. Beermann C. Hane L. Heng G. H. Lee F. Fraundorfer R. Iser R. Triebel I. Posner P. Newman L. Wolf M. Pollefeys S. Brosig J. Effertz C. Pradalier P. Furgale, U. Schwesinger and R. Siegwart. Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project. *IEEE Intelligent Vehicles Symposium (IV)*, page 809–816, 2013.

[33] J. Teo E. Frazzoli J. How Y. Kuwata, S. Karaman and G. Fiore. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[34] A. Broggi D. Braid and G. Schmiedel. The terramax autonomous vehicle. *Journal of Field Robotics*, 23(9):693–708, 2006.

[35] S. Bulavintsev H. Kim J.-H. Ryu, D. Ogay and J.-S. Park. Develop- ment and experiences of an autonomous vehicle for high-speed navi- gation and obstacle avoidance. *Frontiers of Intelligent Autonomous Systems.*, page 105–116, 2013.

[36] S. Peters S. Karaman E. Frazzoli P. Tsio-tras J. hwan Jeon, R. Cowlagi and K. Iagnemma. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. *American Control Conference (ACC)*, page 188–193, 2013.

[37] A. Perez E. Frazzoli S. Karaman, M. R. Walter and S. Teller. Anytime motion planning using the rrt*. *IEEE International Conference on Robotics and Automation (ICRA)*, page 1478–1483, 2011.

[38] D. S. Levine G. S. Aoude, B. D. Luders and J. P. How. Threat-aware path planning in uncertain urban environments. *IEEE/RSJ International Conference in Intelligent Robots and Systems (IROS)*, page 6058–6063, 2010.

[39] R. Kala and K. Warwick. "planning of multiple autonomous vehicles using rrt. *IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*, page 20–25, 2011.

[40] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[41] M. F. Hsieh and U. Ozguner. A parking algorithm for an autonomous vehicle. *Intelligent Vehicles Symposium in 2008 IEEE*, 9(5):1155–1160, 2008.

[42] T. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, 2004.

[43] R. Hindiyeh G. Stanek K. Kritataki-rana C. Gerdes-D. Langer M. Hernandez B. Muller-Bessler J. Funke, P. Theodosis and B. Huhnke. Up to the limits: Autonomous audi tts. *IEEE Intelligent Vehicles Symposium (IV)*, page 541–547, 2012.

[44] P. Zani A. Coati A. Broggi, P. Medici and M. Panciroli. Autonomous vehicles control in the vislab intercontinental autonomous challenge. *Annual Reviews in Control*, 36(1):161 – 171, 2012.

[45] S.Glaser andS.Mammar H.Vorobieva, N.Minoiu-Enache. Geometric continuous-curvature path planning for automatic parallel parking. *Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on, April 2013*, page 418–423, 2013.

[46] N. Minoiu-Enache H. Vorobieva, S. Glaser and S. Mammar. Automatic parallel parking with geometric continuous-curvature path planning. *Intelligent Vehicles Symposium Proceedings*, page 465–471, 2014.

[47] Y. Tazaki-B. Levedahl H. Fuji, J. Xiang and T. Suzuki. Trajectory planning for automated parking using multi-resolution state roadmap considering non-holonomic constraints. *Intelligent Vehicles Sympo- sium Proceedings*, page 407–413, 2014.

[48] M. Brezak and I. Petrovic. Real-time approximation of clothoids with bounded error for path planning applications. *Robotics, IEEE Transactions*, 30(2):507–515, 2014.

[49] A. Lacaze D. Coombs, K. Murphy and S. Legowik. Driving autonomously off-road up to 35 km/h. *Intelligent Vehicles Symposium. IV 2000. Proceedings of the IEEE*, page 186–191, 2000.

[50] R. Behringer and N. Muller. Autonomous road vehicle guidance from autobahnen to narrow curves. *IEEE Transactions on Robotics and Automation*, 14(5):810–815, 1998.

[51] P. Petrov and F. Nashashibi. Modeling and nonlinear adaptive control for autonomous vehicle overtaking. *Intelligent Transportation Systems, IEEE Transactions*, 15(4):1643–1656, 2014.

[52] M. Bertozzi A. Fascioli A. Piazzi, C. G. Lo Bianco and A. Broggi. Quintic g2-splines for the iterative steering of vision-based au- tonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3:27 – 36, 2012.

[53] S. Mammar D. Gruyer S. Glaser, B. Vanholme and L. Nouveliere. Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. *Intelligent Transporta- tion Systems, IEEE Transactions*, 11(3):589–606, 2010.

[54] J. woo Lee and B. Litkouhi. A unified framework of the automated lane centering/changing control for motion smoothness adaptation. *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, Sept 2012*, page 282–287, 2012.

[55] A. Simon and J. Becker. Vehicle guidance for an autonomous vehicle. *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference, 1999,*, page 429–434, 1999.

[56] H. Fritz A. Joos C. Rabe C. G. Keller, T. Dang and D. M. Gavrila. Active pedestrian safety by automatic braking and evasive steering. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1292–1304, 2011.

[57] R. Lattarulo J. Perez and F. Nashashibi. Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles. *Intelligent Vehicles Symposium Proceedings*, 9(5):510–515, 2014.

[58] R. Lattarulo V. Milanes D. Gonzalez, J. Perez and F. Nashashibi. Continuous curvature planning with obstacle avoidance capabilities in urban scenarios. *17th International Conference on Intelligent Transportation Systems (ITSC)*, page 1430–1435, 2014.

[59] J. Villagra J. Perez, J. Godoy and E. Onieva. Trajectory generator for autonomous vehicles in urban environments. *IEEE International Conference on Robotics and Automation (ICRA)*, page 409 –414, 2013.

[60] H. T. N. Nejad Q. H. Do L. Han, H. Yashiro and S. Mita. Bezier curve based path planning for autonomous vehicle in urban environ- ment. *Intelligent Vehicles Symposium (IV), 2010 IEEE.*, 9(5):1036–1042, 2010.

[61] G. Zheng Z. Liang and J. Li. Automatic parking path optimization based on bezier curve fitting. *Automation and Logistics (ICAL), 2012 IEEE International Conference*, page 583–587, 2012.

[62] L.Armesto N.Montes, A.Herraez and J.Tornero. Real-time clothoid approximation by rational bezier curves. *Robotics and Automation*, 2008.

[63] M. Mora N. Montes and J. Tornero. Trajectory generation based on rational bezier curves as clothoids. *Intelligent Vehicles Symposium*, page 505–510, 2007.

[64] H. Dahlkamp D. Stavens A. Aron J. Diebel P. Fong J. Gale M. Halpenny G. Hoffmann et al. JS. Thrun, M. Montemerlo. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[65] P. M. Kinney C. Koutsougeras P. G. Trepagnier, J. Nagel and M. Dooner. Kat-5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain. *Journal of Field Robotics*, 23(8):509–526, 2006.

[66] Z. Shiller and "Dynamic motion planning of autonomous vehicles Y.-R. Gwo. Robotics and automation. *IEEE Transactions*, 7(2):241–249, 1991.

[67] H. Jonsson M. Staffanson T. Berglund, A. Brodnik, "Planning smooth I. Soderkvist, and obstacle-avoiding b-spline paths for autonomous mining vehicles. Automation science and engineering. *IEEE Trans- actions on*, 7(1):167–172, 2010.

[68] B.B. Choudhury R.K. Panda. An effective path planning of mobile robot using genetic algorithm. 2015 IEEE International Conference on Computational Intelligence Communication Technology, 2015.

[69] M. Hasanzadeh A.H. Karami. An adaptive genetic algorithm for robot motion planning in 2d complex environments. *Journal of Intelligent Robotic Systems*, 43:317–329, 2015.

[70] K. Yigit. Path planning methods for autonomous underwater vehicles. Master's thesis, Department of Mechanical Engineering Massachusetts Institute of Technology, Cambridge, Massachusetts, 6 2011. Yüksek Lisans Tezi.

[71] B. Saim. Sezgisel optimizasyon algoritmaları (heuristic algorithms). 2017.

[72] Ş. E. Şeker. Yapay zeka 5.3 : Sezgisel: Best first search algoritması. 2017.

[73] Ş. E. Şeker. Yapay zeka 5.5 : A yıldız (a*) sezgisel arama algoritması. 2017.

[74] E. Ueland. A* (astar / a star) search algorithm. easy to use. *IEEE transactions on information forensics and security*, 2017.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: YAYLALI, İbrahim
Uyruğu	: TC
Doğum Yeri ve Tarihi	: 16/10/1986
Medeni Hali	: Evli
Tel	: +90 544 778 42 97
Fax	: -
e-mail	: ibrahimyaylali2011@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Lisans	: KTO Karatay Üniversitesi	Temmuz-2015
Yüksek Lisans	: KTO Karatay Üniversitesi	Ağustos-2018
Doktora	: -	

İş Deneyimi

Yıl	Yer	Görev
2013 - (1 ay)	Teknodrom Robotik Otomasyon	Stajyer
2014 - (1 ay)	ASELSAN	Stajyer
2016 - (halen)	TÜBİTAK BİLGEM Bilişim Teknolojileri Enstitüsü (Radar Teknolojileri Araştırma Grubu)	AR-GE Personeli (Mekatronik Sistem Tasarımı ve Hareket Kontrolü)

Yabancı Dil

İngilizce(İyi), Çince(Başlangıç)

Yayınlar

-