

**BİR ROBOT BALIK İÇİN İKİ BOYUTLU YOL
PLANLAMA ALGORİTMALARININ
UYGULANMASI**

Seda YETKİN

Yüksek Lisans Tezi

Mekatronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Gonca ÖZMEN KOCA

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİR ROBOT BALIK İÇİN İKİ BOYUTLU YOL PLANLAMA
ALGORİTMALARININ UYGULANMASI

YÜKSEK LİSANS TEZİ

Seda YETKİN

(141134105)

Anabilim Dalı: Mekatronik Mühendisliği

Programı: Mekanik Sistemler

Tezin Enstitüye Verildiği Tarih: 06 ARALIK 2016

Tezin Savunulduğu Tarih: 19 ARALIK 2016

Tez Danışmanı: Yrd. Doç. Dr. Gonca ÖZMEN KOCA (F.Ü.)

Diğer Jüri Üyeleri: Prof. Dr. Z. Hakan AKPOLAT (F.Ü.)

Yrd. Doç. Dr. Mehmet ÜSTÜNDAĞ (B.Ü.)

ARALIK – 2016

ÖNSÖZ

Yüksek lisans ve proje çalışmalarım sürecinde maddi ve manevi destek sağlayan kurum ve kişilere teşekkürlerimi sunmayı bir borç bilirim.

Bu çalışmanın planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Yrd. Doç. Dr. Gonca ÖZMEN KOCA' ya sonsuz teşekkürlerimi sunarım.

Çalışmalarında bilgi ve tecrübelerini benden hiçbir zaman esirgemeyen, tezimin oluşmasında bana fazlasıyla yardımcı olan Sayın hocalarım Yrd. Doç. Dr. Mustafa AY' a ve Yrd. Doç. Dr. Cafer BAL' a teşekkürü bir borç bilirim.

Ayrıca çalışma sırasında her türlü sorumu yılmadan cevaplayan ve hiçbir yardımını esirgemeyen lisans ve lisanüstü arkadaşım Arş. Gör. Mustafa Can BİNGÖL' e çok teşekkür ederim. Yüksek lisans eğitimim boyunca yardımlarını esirgemeyen Arş. Gör. Deniz KORKMAZ' a ve Arş. Gör. Halil İbrahim YAMAÇ' a manevi destekleri için çok teşekkür ederim.

Hayatımın her anında maddi ve manevi desteğini benden esirgemeyen, sabır ile eğitim hayatımda bana yol gösteren aileme tüm kalbimle teşekkür ediyorum.

Ayrıca bu tez çalışması, TÜRKİYE BİLİMSEL ve TEKNOLOJİK ARAŞTIRMA KURUMU (TÜBİTAK) tarafından 114E652' nolu proje ile desteklenmiştir.

Seda YETKİN

ELAZIĞ – 2016

İÇİNDEKİLER

Sayfa No

ÖNSÖZ	I
İÇİNDEKİLER.....	II
ÖZET	IV
SUMMARY	V
ŞEKİLLER LİSTESİ	VI
TABLolar LİSTESİ	VIII
SEMBOLLER LİSTESİ	IX
KISALTMALAR LİSTESİ	XII
1. GİRİŞ	1
1.1. Mobil robot uygulamaları	1
1.2. Otonom sualtı araç uygulamaları.....	5
1.3. Robot balık uygulamaları	6
2. BALIKLARIN ANATOMİSİ VE HAREKETLERİ	9
2.1. Robot balığın matematiksel modeli	10
2.2. Robot balığın doğrusal modeli	13
3. YOL PLANLAMA ALGORİTMALARI	18
3.1. A* yol planlama algoritması.....	19
3.2. Alt hedef graf (subgoals graphs) algoritması	23
4. BENZETİM ÇALIŞMALARI	25
4.1. A* Uygulamaları	25
4.2. Alt hedef graf uygulamaları.....	30
4.3. Yol bulma algoritmalarının dinamik model uygulamaları	35
5. SONUÇLAR ve TARTIŞMA.....	41
6. ÖNERİLER	42

KAYNAKLAR	43
EKLER.....	47
ÖZGEÇMİŞ.....	55

ÖZET

Son dönemlerde teknolojinin gelişimine paralel olarak robotik sistemlerle ilgili çalışmalar hız kazanmıştır. Bu robotik sistemlerin çoğu tabiatta yaşamını sürdüren hayvanlardan ilham alınarak oluşturulmuştur.

Bu çalışmada, balıklardan esinlenerek ve gerçek bir balığın hareketlerini sergileyebilen, Carangiform yüzüş şekline bağlı bir robot balığın doğrusal modeli MATLAB benzetim ortamında oluşturulmuştur. Bu robot balığın otonom olarak bir yerden başka bir yere hareket edebilmesini sağlamak için farklı yol planlama algoritmaları oluşturulmuştur. Bu planlama algoritmalarından olan A* ve Alt hedef graf algoritmaları kullanılarak MATLAB ortamında 2 boyutlu çalışma alanları için birçok uygulama gerçekleştirilmiştir. Robot balığın en kısa yoldan hedefe ulaşabilmesi ve engeller arasından engellere çarpmadan ilerleyebilmesi için A* ve Alt hedef graf algoritmaları kullanılmıştır. Benzetim ortamında oluşturulan robot balığın doğrusal modeline A* ve Alt hedef graf algoritmalarının belirlemiş olduğu konum değerleri uygulanmıştır. Robot balık modelinin A* ve Alt hedef graf algoritmalarıyla belirlenen rotayı takip ettiği görülmüştür.

Sonuç olarak, robot balık modelinin A * ve Alt hedef graf algoritmaları tarafından belirlenmiş rota ile başlangıç noktasından bitiş noktasına ulaşabileceği çeşitli uygulamalar ile gösterilmiştir.

Anahtar Kelimeler: Robot Balık, Yol Planlama, A* Algoritması, Alt Hedef Graf Algoritması

SUMMARY

Application of Two-Dimensional Path Planning Algorithms for the Robot Fish

In recent years, researches about robotics systems have gained speed in parallel with the development of technology. Most of these robotic systems are inspired by animals living in the nature.

In this study, a linear model of a robot fish based on a Carangiform swimming mode, inspired by a real fish and capable of exhibiting movements of a real fish, has developed in the MATLAB simulation environment. Various path planning algorithms have created to ensure autonomously movements of the robot fish from one place to another. Many applications have been realized for 2D study areas by using A* and Subgoal graph algorithms in the MATLAB environments. A * and Subgoal graph algorithms have been used in order to reach the target from the shortest path and move between obstacles, without hitting them. The location values obtained from A * and Subgoal graph algorithms have applied to the linear model of robot fish. The robot fish model has been seen to follow path determined by the A * and Subgoal graph algorithms.

As a result, it has shown with various applications that the robot fish model can reach the target point from the starting point with the path determined by the A* and Subgoal graph algorithms.

Key Words: Robot Fish, Path Planning, A * Algorithm, Subgoal Graph Algorithm

ŞEKİLLER LİSTESİ

Sayfa No

Şekil 1. 1. Tek eklemlı robot balık [12].....	6
Şekil 1. 2. G9 isimli robot balık [19].....	7
Şekil 2. 1. Balık anatomisi.....	9
Şekil 2. 2. Kuyruk hareket sinyali	10
Şekil 2. 3. Robot balığın hareketleri ve düzlemleri	14
Şekil 2. 4. Elips şeklindeki bir kütle.....	15
Şekil 3. 1. 6X6' lık A* algoritmasının çalışma ortamı	20
Şekil 3. 2. A* algoritmasının G, H ve F değerleri	21
Şekil 3. 3. Alt hedef graf algoritmasının a.) Uygulama alanı b.) Birinci adımı	23
Şekil 3. 4. Alt hedef graf algoritmasının a.) İkinci adımı b.) Üçüncü adımı ve en kısa yol	24
Şekil 4. 1. A* algoritmasının birinci uygulama sonucu	26
Şekil 4. 2. A* algoritmasının github sonucu	26
Şekil 4. 3. A* algoritmasının ikinci uygulama	27
Şekil 4. 4. A* algoritmasının üçüncü uygulama.....	28
Şekil 4. 5. A* algoritmasının dördüncü uygulaması	28
Şekil 4. 6. A* algoritmasının beşinci uygulaması	29
Şekil 4. 7. A* algoritmasının altıncı uygulaması	29
Şekil 4. 8. Alt hedef graf algoritması birinci uygulama	31
Şekil 4. 9. Alt hedef graf algoritması ikinci uygulama.....	31
Şekil 4. 10. Alt hedef graf algoritması üçüncü uygulama	32
Şekil 4. 11. A* ve Alt hedef graf algoritmasının birinci uygulaması	33
Şekil 4. 12. A* ve Alt hedef graf algoritmasının ikinci uygulaması	34
Şekil 4. 13. A* ve Alt hedef graf algoritmasının üçüncü uygulaması.....	34
Şekil 4. 14. Robot balığın doğrusal olmayan ve doğrusal modellerinin algoritma sonucu	36
Şekil 4. 15. Doğrusal model ve A* algoritmasının birinci uygulaması.....	37
Şekil 4. 16. Doğrusal model ve A* algoritmasının ikinci uygulaması	37
Şekil 4. 17. Doğrusal model ve A* algoritmasının üçüncü uygulaması	38
Şekil 4.18. Doğrusal model ile birinci uygulama a.)Alt hedef graf algoritması b.)A* algoritması	39
Şekil 4. 19. Doğrusal model ile ikinci uygulama a.)Alt hedef graf algoritması b.)A* algoritması	39

Şekil 4. 20. Doğrusal model ile üçüncü uygulama a.)Alt hedef graf algoritması b.)A* algoritması	40
---	----

TABLÖLAR LİSTESİ

	<u>Sayfa No</u>
Tablo 1. Algoritmaların özelliklerinin değerlendirilmesi [9]	3
Tablo 2. PID parametreleri	14
Tablo 3. İki eklemlı robot balığın benzetim çalışma parametreleri	17
Tablo 4. A* algoritmalarının yol maliyeti ve çalışma zamanı	30
Tablo 5. Alt hedef algoritmalarının yol maliyeti	32
Tablo 6. A* ve Alt hedef graf algoritmasının yol uzunlukları	35

SEMBOLLER LİSTESİ

$T(n)$	: CPU süresi (sn)
n	: Karesel bir alanda köşe sayısı
$f(n)$	: Toplam maliyet
$g(n)$	: Bir pozisyonun başlangıç pozisyonuna olan maliyeti
$h(n)$	: Bir pozisyonun hedef pozisyonuna olan maliyeti
$r(n)$	: Bir pozisyonun hareket yönüne doğru döndürmek için gereken zamanı
s	:İlk düğüm
s'	:Sonraki düğüm
$c(s,s')$	: s düğümünden s' düğümüne giderken oluşan maliyeti
y_{body}	: Kuyruğun enine hareket sinyali
x	: Ana eksenindeki yer değişim (m)
t	: Zaman (s)
c_1	: Doğrusal dalga genlik
c_2	: Karesel dalga genlik
k	: Dalga katsayısı
w	: Kuyruk hareketi sinyalinin açısal hızı (rad/s)
f	: Kuyruğun çırpınma frekansı (Hz)
λ	: Dalga boyu (s)
M	: Ayırıklaştırılmış enine hareket sinyalinin örnek sayısı
i	: Döngü sayısı
m	: Robot balığın ağırlığı (kg)
M	: Kütle matrisi
$C(v)$	: Coriolis matrisi
$D(v)$	: Hidrodinamik sönümleme matrisi

$g(\eta)$	: Yerçekimi vektörü
τ_f	: Hidrodinamik vektör
M_{RB}	: Gövde matrisi
M_A	: Ek-kütle matrisi
$C_{RB}(v)$	: Gövde coriolis matrisi
$C_A(v)$	: Hidrodinamik coriolis matrisi
D_L	: Doğrusal hidrodinamik sönümleme matrisi
D_{NL}	: Doğrusal olmayan hidrodinamik sönümleme matrisi
W	: Yerçekimi kuvveti
B	: Kaldırma kuvveti
$c()$	: Açılarının kosinüs değeri
$s()$	: Açılarının sinüs değeri
K_p	: Oransal kazancı
K_i	: İntegral kazancı
K_d	: Türevsel kazancı
$e(t)$	: Hata değeri
$v(t)$	: Giriş sinyali
$y(t)$	: Çıkış sinyali
u_s	: Doğrusal sönümleme katsayısı
αT	: İtme kuvvetinin kazancı
X_u	: Dalgalanma hareketindeki ek kütlesi
e	: Merkez kaçıklığı
b_r	: Dönme kazançları

b_v : Sallanma kazançları

ψ : Sapma açısı

r : Sapma açısal hızı

KISALTMALAR LİSTESİ

A*	: A-Yıldız Algoritması
RUR	: Rossum's Universal Robots
GA	: Genetik Algoritma
PSO	: Parçacık Sürü Optimizasyonu
JPS(A*)	: Dğümleri Atlayan A-Yıldız
LSPI	: En Küçük Kareler Yöntemi
CPU	: Merkezi İşlem Birimi
BFS	: Sığ Öncelikli Arama Algoritması
D*	: D-yıldız Algoritması
LPM	: Legendre Pseudospectral Metot
BM	: Bulanık Mantık
DC	:Doğru Akım
BCF	: Vücut ve/veya Kuyruk ile Hareket
MPF	: Orta ve/veya Çift Yüzgeçler ile Hareket
PID	:Oransal-İntegral-Türev Denetleyici

1. GİRİŞ

Son dönemlerde teknolojinin gelişimine paralel olarak yapay zeka, kontrol ve robotik konularda gelişmeler görülmektedir. Robotik sistemler, insanların işlerini kolaylaştırması için ve daha çok insanlar için tehlikeli olabilecek ortamlarda kullanılması gereksinimi ile ortaya çıkmıştır. Bu gereksinimlerden dolayı robotlar birçok alanda karşımıza çıkmaya başlamıştır. “Robot” kavramını ilk olarak Çekoslovak yazar Karel Capek, 1921 yılında RUR (Rossum’s Universal Robots) adlı tiyatrodan kullanmıştır. Bu tiyatrodan robotlar insani duygulardan arındırılmış mekanik ve otonom varlıklar olarak tanıtılmıştır. Fakat robot fikrinin geçmişi 3000 sene öncesine dayanmaktadır. Homeros, İlyada adlı eserinde üçayaklı hareket edebilen varlıklardan ve bir hint efsanesinde hareket edebilen mekanik fillerden bahsetmiştir [1]. Günümüzde ise robotlar; ev, hastane, okul, havaalanı ve denizaltı gibi yerlerde karşımıza çıkmaktadır.

Bir cihazın robot olarak ifade edilmesi için 3 özelliğe sahip olması gerekmektedir. Bu özelliklerden birincisi robotun çevre ile iletişime geçmesi ve çevreyi algılama yeteneğine sahip olmasıdır. Robot, bu algılama işlemini çeşitli sensörler yardımıyla gerçekleştirebilmektedir. İkinci olarak algıladığı karmaşık problemleri çözebilmeli, karar vermeli ve uygun sonuçları bulabilmelidir. Son olarak ise robotun bulduğu uygun sonuçları çevreye geri vermesidir. Kısaca robot, otonom olarak ya da önceden programlanmış görevleri yerine getirebilen elektro-mekanik cihaz olarak da tanımlanabilmektedir [2]. Robotların otonom çalışabilmeleri için yol planlama algoritmalarının geliştirilmesi gerekmektedir. Robotların hedeflerini engeller arasından engellere çarpmadan ve en kısa yoldan gerçekleştirebilmeleri için yol planlama algoritmalarını kullanması gerekir. Bu yolların bulunmasında, sistemlerin çalıştığı ortamlara uygun farklı yol bulma algoritmaları kullanılmış ve geliştirilmiştir.

1.1. Mobil Robot Uygulamaları

Bu bölümde, farklı ortamlarda çalışabilen mobil robotların yol bulma algoritmalarıyla yapılan çalışmaları hakkında detaylı bir şekilde bilgi verilmektedir.

Taharwa ve ark. [3], bir mobil robotun 3 farklı ortamda başlangıç ve bitiş noktası arasındaki en uygun yolu bulması için genetik algoritmayı (GA) kullanmışlardır. Ayrıca

mobil robot uygulamalarında yol bulma işlemleri için Bulanık Mantıkta (fuzzy lojik) kullanılmıştır [4].

Zhang ve Zhao [5], birden fazla hareket edebilen veya durağan arabaların olduğu bir parkta bir arabanın yolunu bulması için A-star (A*) ve Dijkstra algoritmalarının analizlerini ve karşılaştırmalarını gerçekleştirmişlerdir. A* ve Dijkstra algoritmalarında belirli sınırlandırmalar olduğu için akıllı park sistemleri için yeterli olmadığı sonucuna varılmıştır. Bu sebepten dolayı her iki algoritmanın avantajları ve dezavantajları dikkate alınarak yeni bir algoritma oluşturulmuştur. Oluşturulan bu algoritma diğer iki algoritmanın optimize edilmiş halidir. Sonuç olarak optimize edilmiş algoritma diğer algoritmalara göre daha verimli, daha az maliyetli, daha güvenli, daha esnek ve taşınabilir özelliğe sahiptir.

Das ve ark. [6], bilinmeyen bir çevrede Khepera 2 adlı bir mobil robot için online olarak yol bulma işlemini gerçekleştirmişlerdir. Bir başlangıç noktasından bitiş noktasına en kısa yolu engellerden kaçarak bulmak ve sabit engellerin çevresinden hareket ederek yolu bulması için iyi bilinen sezgisel algoritmalarından biri olan A* algoritması geliştirilmiştir. Bu geliştirme işleminde geleneksel A* algoritmasındaki toplam maliyet $f(n)$ değeri hesaplanırken $f(n) = g(n) + h(n) + r(n)$ şeklinde bir formül kullanılmıştır. $g(n)$, $h(n)$ ve $r(n)$ sırasıyla bir pozisyonun başlangıç pozisyonuna olan maliyetini, bir pozisyonun hedef pozisyonuna olan maliyetini ve mobil robot için hareket yönüne doğru döndürmek için gereken zamanı ifade etmektedir. Sabit bilinmeyen engeller ile bilinmeyen bir çevre oluşturulmuş ve bu çevre ızgara-haritada dönüştürülerek yol bulma işlemi sağlanmıştır. Bu çalışmanın sonucunda zaman, yol ve uzaklık gibi maliyetlerin optimum olduğu yol bulunmuştur.

Yuan vd. [7] yaptıkları çalışmada, MAKLING grafiği ile mobil robotun çalışma uzayını tanımlamışlardır. Başlangıç noktasından bitiş noktasına kadar en kısa yolu elde etmek için bilgisiz arama algoritmalarından Dijkstra algoritmasını kullanmışlardır. Yeni bir algoritma olan Parçacık Sürü Optimizasyonu (PSO) algoritması çok fazla araştırmacının ilgisini çekmiştir. Bundan dolayı PSO'nun avantajları ve dezavantajları gösterilmiş ve birkaç uygulama gerçekleştirilmiştir. Sonuç olarak, PSO algoritmasının teorisinin temeli iyi bilinmediği ve yerel minimum gibi eksikliklerin üstesinden gelmek için daha çok geliştirilmesi gerektiği ortaya koyulmuştur. Fakat Kang ve ark. [8], engelleri bilinen bir

alanda Dijkstra algoritması, geliştirilmiş Dijkstra algoritması ve geliştirilmiş Dijkstra algoritmasına PSO' nun uygulanması ile optimum yolun bulunması üzerine çalışmışlardır. Çalışma sonucunda uygulanan algoritmaların yol uzunlukları hesaplanmış ve en kısa yolun geliştirilmiş Dijkstra algoritmasına PSO' nun uygulanması ile bulunduğu görülmüştür.

Duchon vd. [9] çalışmalarında, ızgara harita üzerine yerleştirilmiş, navigasyon özelliği bulunan bir mobil robotun yol planlamasını deneysel olarak gerçekleştirmişlerdir. Yol planlaması yapılırken A*, Basic Theta* ve Phi* algoritmaları kullanılmıştır. Ayrıca A* algoritmasının da iyileştirilmeler yapılarak atlama noktası arama (JPS A*) algoritması geliştirilmiştir. Bu algoritmalar arasında karşılaştırmalar yapılmıştır. Sonuç olarak JPS A* algoritmasının avantajının diğer algoritmalara göre yolların daha hızlı bulunması olduğu sonucuna varılmıştır. Ayrıca bu algoritma çeşitli ortamlarda en iyi yolu bulabilir. A* algoritmasının dezavantajı ise uzun yollarda kullanılamamasıdır. Basic Theta* ve Phi * algoritmaları ise uzun yollarda kullanılmakta fakat yolları daha uzun zamanda hesaplamaktadırlar.

Bu çalışmada mobil robota uygulanan algoritmaların birbirlerine göre avantajları ve dezavantajları aşağıdaki tabloda verilmiştir:

Tablo 1. Algoritmaların özelliklerinin değerlendirilmesi [9]

Algoritmalar	Avantajlar	Dezavantajlar
A*	Kolay çalışır. Phi, Basic Theta algoritmalarına nispeten daha hızlıdır. Değiştirilebilir.	Her açıda arama yapamayabilir.
Basic Theta*	Her açıda arama yapabilir. Noktalar kümesinde istenilen yol oluşturulur.	Hesaplama zamanı uzundur.
Phi	Her açıda arama yapabilir. Noktalar kümesinde istenilen yol oluşturulur. Hızlı bir şekilde yeniden planlanabilir.	Hesaplama zamanı uzundur.
JPS(A*)	Hızlıdır. Noktalar kümesinde istenilen yol oluşturulur.	Her açıda arama yapamayabilir.

Zuo vd. [10] yaptıkları çalışmada, karmaşık bir çevrede mobil robotun yörüngesi için yeni bir hiyerarjik yol bulma yaklaşımı sunmuşlar. Bu yaklaşım 2 seviyeli bir yapıdan oluşmaktadır. Birinci seviyede engellere çarpmadan geometrik bir yolun bulunması için birkaç yol noktaları A* algoritması ile bulunmuştur ve bu noktalara alt-hedef adını vermişlerdir. Bu alt-hedefler bulunurken robotun kinematik ve dinamik sınırlamalar dikkate alınmadan hesaplanmaktadır. İkinci seviyede ise robotun kinematik karakteristikleri dikkate alınarak uygun yörüngenin bulunması için En küçük kareler yöntemi (LSPI) kullanılmıştır. İkinci seviyede birinci seviyede bulunan alt-hedeflere yaklaşarak en uygun yol bulunmuştur.

Okutkan [11] yaptığı çalışmada, otonom bir mobil robotun en önemli özelliklerinden biri olan yol bulma üzerine yapılan araştırmalar sonucunda A* ve Dijkstra algoritmaları bilgisayar ortamında gerçekleştirmiştir. Bu yol bulma algoritmalarından hareketle daha iyi bir işlem performansı sunan bir algoritma MS Visual Basic'te çalışacak şekilde hazırlanmıştır. 100x100 ebadındaki aşırı karmaşık bir haritada başlangıç ve bitiş noktaları haritanın karşılıklı köşelerine yerleştirilmesi durumunda bile toplam işlem süresi 17 saniyenin altındadır. Bu süre aynı harita üzerinde A* algoritması kullanılarak Ms Visual Basic'te uygulaması yapıldığında 20 saniyenin üzerine çıkmıştır.

Soltani vd. [12] yaptıkları çalışmada, şantiyelerdeki yol planlama uygulamalarını anlatmışlardır. Taşıma merkezli şantiyelerde site yöneticisinin site operatörleri yada araçları arasında en kısa yolları seçmesi gerekir. Bu yolları seçerken en kısa mesafenin bulunması sırasında riskin düşük olması ve görünürlüğün belirgin olması gerekmektedir. Şantiyelerdeki performans değerlerinde 3 optimizasyon algoritması kullanmıştır. Bunlar Dijkstra, A* ve Genetik algoritma algoritmalarıdır. Karmaşık alanlara uygulanan Dijkstra algoritmasının merkezi işlem birimi (CPU) süresini hesaplamak için (1.1) nolu denklem kullanılırken, A* algoritması için (1.2) nolu denklem kullanılmıştır.

$$T(n) = 2 \times (n-1)^2 \in \theta n^2 \quad n = \text{Karesel bir alanda köşe sayısı} \quad (1.1)$$

$$T(n) = (n)^2 \in \theta(n^2) \quad n = \text{Karesel bir alanda köşe sayısı} \quad (1.2)$$

CPU sürelerine karşılaştırılması ve yol sonuçlarının doğruluğu bu optimizasyon algoritmalarıyla kesin olarak belirlenmiştir. Bu algoritmalar arasında CPU süreleri arasında karşılaştırmalar yapılmıştır. Bu karşılaştırmaların sonucunda ise Dijkstra algoritması bir başlangıca göre bir bitiş üreterek doğru sonuç bulabilmiştir. Dijkstra algoritmasının büyük ölçekli problemlerde doğru sonuç verebildiği ancak CPU süresinin çok uzun olduğu görülmüştür. A* algoritması, Dijkstra algoritmasına göre daha kısa CPU süresinde cevap vererek sonuçlar bulabilmektedir.

Vacariu vd.[13] yaptıkları çalışmada, mobil robotların hareketlerinde görülen engelden kaçma ve yol bulma işlemlerinde sıg öncelikli arama (Breadth First Search-BFS) algoritmasıyla kullanmışlardır.

Chen vd.[35] yaptıkları çalışmada, manipülatör, katı cisimler ve çok yönlü katı cisimlerin hareket planlanması için SANDROS adında bir genel bir araştırma stratejisi sunmuşlardır. SANDROS bir dinamik grafik arama algoritmasıdır ve bir alanda sezgisel olarak en kısa hareketi bulmak için en iyi ilk arama olarak tanımlanabilir.

1.2. Otonom Sualtı Araç Uygulamaları

Son dönemlerde sualtı araçlarıyla yapılan yarışlarda sualtı araçlarının yarışı kazanabilmeleri için en kısa yolu en kısa zamanda bulması gerekir. Bundan dolayı sualtı araçlarında önemli problemlerden biride yol bulma işlemidir. Yol bulma işlemini gerçekleştirmek için yol bulma algoritmaları; küresel planlama ve yerel yeniden planlama olarak genellikle 2 ana katagoride incelenmektedir. Eğer durağan ve bilinen engellerin olduğu bir alanda yol bulma işlemi gerçekleştiriliyorsa sezgisel algoritmalara dayanan A* ve D-yıldız (D*) gibi algoritmalar [19] ve yapay potansiyel alan yaklaşımları [14] kullanılmıştır. Engellerin bilinmediği ve engellerin hareket ettiği bir ortamda yol bulma işlemi gerçekleşirse GA ve PSO gibi evrimsel algoritmalar kullanılmıştır [15].

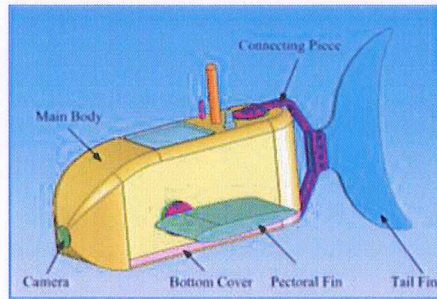
Zhuang vd. [15] yaptıkları çalışmada, otonom sualtı araçlarının dinamik bir çevrede en iyi yörüngeyi bulabilmesi için PSO ve legrendre pseudospectral metod (LPM) birleşimi bir algoritma sunmuşlardır. Bu birleştirilmiş algoritma yolu bulma işlemini 2 aşamada gerçekleştirmektedir. Birinci aşamada PSO algoritması ilk yolu bulmak için başlangıç motoru olarak hizmet vermektedir. Diğer aşamada ise ilk aşamadaki en iyi sonuç LPM yol olarak yüklenir ve hedefe ulaşınca kadar on-line olarak yenilenir.

Sun ve Zhu [16] yaptığı çalışmada, bilinen ve bilinmeyen engellerin bulunduğu su ortamında otonom sualtı araçlarının 2 boyutlu ve 3 boyutlu yol planlamaları için D* Lite algoritmasını geliştirmişlerdir. D* Lite algoritması, Lifelong Planning A* (LPA*) ve D* algoritmalarının avantajları göz önüne alınarak geliştirilmiş bir algoritmadır. D* Lite algoritmasında $g(s)$ ve $rhs(s)$ parametrelerine göre yol bulunmaktadır. $g(s)$ değeri başlangıç düğümünden bir s düğümüne giderken gerekli olan maliyeti ifade etmektedir. $rhs(s)$ ise $g(s)$ değerinin bir önceki değerine bağlıdır ve s düğümü hedef düğüme ulaşmış ise 0, ulaşmamış ise $\min(c(s,s') + g(s'))$ şeklinde hesaplanmaktadır. $c(s,s')$ ise s düğümünden s' düğümüne giderken oluşan maliyeti ifade etmektedir. Bu değerler dikkate alınarak 2 boyutlu çevre olarak $20 \times 20m$, 3 boyutlu çevre olarak $10 \times 10 \times 10m$ boyutunda simulasyon sonuçları incelenmiştir.

1.3. Robot Balık Uygulamaları

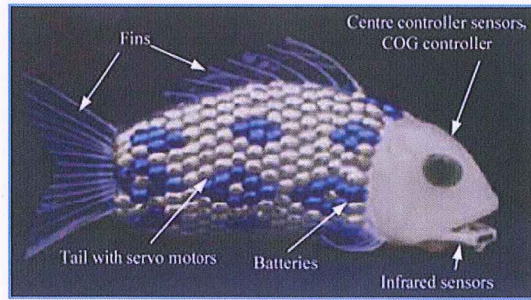
Biomimetik robot balıklar, biyoloji ve robot teknolojisinde yeni bir araştırma alanı olarak son yıllarda araştırmacıların ilgisini çekmiş ve robot balık ile birçok çalışma gerçekleştirilmiştir. Aynı zamanda, yapılan birçok çalışma farklı tasarlanmış robot balıklar ile gerçekleştirilmiştir.

Hu vd. yaptıkları çalışmada [17], tek eklemlili bir robot balığın kapalı çevrim konum kontrolünü gerçekleştirmişlerdir (Şekil 1.1). Geri besleme için kamera görüntüleri ve kontrolör olarak Bulanık Mantık (BM) kullanılmıştır. Hu vd. [18] bir başka çalışmasında ise bir robot balık için üzerinde bulunan kamera yardımı ile nesne takip algoritmasını geliştirmişlerdir.



Şekil 1. 1. Tek eklemlili robot balık [12]

Liu ve Hu [19] yaptıkları çalışmada, çok eklemlili bir robot balığın vücut hareketlerinin oluşturulması için göreceli bir kuyruk fonksiyonu oluşturmuştur. Bu fonksiyon ile kuyruk pozisyonları bir dizi haline getirilmiştir. Daha sonra, eklemlerin dönüş açılarını hesaplamak için sayısal bir yaklaşım metodu kullanılmıştır. Deneysel çalışmalarda Şekil 1.2' de gösterilen 52 cm uzunluğunda G9 isimli robot balık kullanılmıştır. Robot balığın kuyruk hareketini sağlamak için kuyrukların bağlantı yerlerinde 3 yada 4 adet servo motor kullanılmıştır. Ayrıca 2 adet DC motor kullanılmıştır. Bu DC motorlardan biri ağırlık merkezini değiştirmek için balığın baş kısmına sabitlenmiştir. Diğer DC motor ise su pompalanarak robotun ağırlığını ayarlamak için micro-pompayı kontrol etmede kullanılmıştır. Robot balığın merkezi yönetimi 400MHz Gumstix bilgisayar tarafından yürütülmüştür. Gumstix; servo motor, DC motor, sensör bordlarını içeren birçok PIC micro-kontrol sinyali üretmek için kullanılmıştır.



Şekil 1. 2. G9 isimli robot balık [19]

Yol planlama problemi, biomimetik robot balıklar ile yapılan araştırmalarda en önemli konulardan biri haline gelmiştir.

Oiang [20] yaptığı çalışmada, laboratuvar ortamında biomimetik robot balık sistemini tanıtmış ve daha sonra bunu dönme kontrolün odaklanmak için B-spline kullanarak A* algoritması ve yol planlaması geliştirmiştir. İlk olarak, biomimetik robot balık karakterlerini belirlemek ve saptamak için sınıflandırma gerçekleştirilmiştir. İkinci olarak da biomimetik robot balık donanım ve yazılımı tanıtılmıştır. Özellikle, RGB renk sistemi ve HLS renk sistemini içeren platformun görseli üzerine odaklanılmıştır. Ayrıca, robot balık performansını ve hız dönüş karakterlerini belirleyen kuyruk yüzgeci frekans ve genliğini içeren, dönme yarıçapı, hız vb. hayati parametreler tanıtılmıştır. Ayrıca, zaman

karmaşıklığı ve eski bir uzay karmaşıklığını azaltmak için yeni bir A* algoritması önerilmiştir.

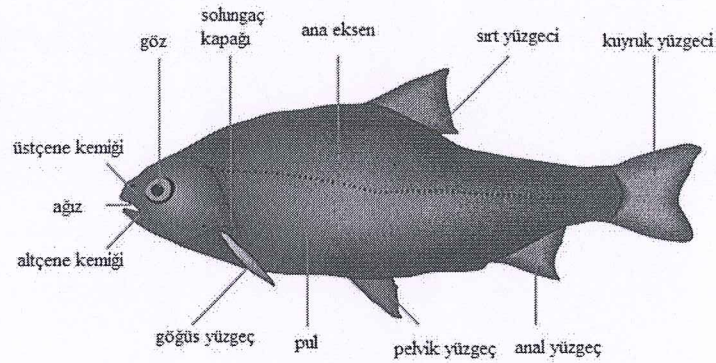
Zhou vd. [21] yaptıkları çalışmada, Thunniform yüzüş moduna sahip bir robot balığın kızılötesi ve ışığa duyarlı sensörler ile havuzdan bilgi alarak engelden kaçma algoritmalarını uygulamışlardır. Bu algoritmalar ışık kaynağı arama ve dinamik ışık kaynağı izleme algoritmalarıdır. Bu algoritma için robot balığın baş kısmına 7 adet ışık sensörü koyulmuştur. Sensörler engeller ile karşılaştığında zaman ışık miktarını ölçer ve sensörlerin ölçtüğü ışık miktarı çevredeki ışık miktarına yakın olan sensöre doğru yönelerek robot balık engeli aşarak hareketini sağlamış olur. Çalışmanın deneysel sonuçlarında robot balığın engelden kaçarak hareket edebilmesi önerilmiş algoritmalar ile sağlanmıştır.

Shao vd. [22] yaptıkları çalışmada, robot balık havuz içerisinde yüzme hareketini gerçekleştirdiği zaman engellerin çevresinde potansiyel bir akış oluşmaktadır. Engellerden kaçmak ve uygun olabilecek yolları belirlemek için bu akış çizgilerini kullanmışlardır.

Wang ve Jiang [23] yaptıkları çalışmada, 2 boyutlu ve kompleks bir çevrede çok fazla engel olduğu için çoğu uygulamalarda kullanılan geleneksel A* algoritması kullanmışlardır. Bu çalışmada A* algoritmasına dayanan yeni bir divide-and-rule yol bulma metodunu robot balık için yol bulma işleminde kullanmışlardır. Robot balığın baş kısmının hedefe yönelmesi için uygun bir yörünge tasarlanmıştır. Robot balığa uygulanan bu yeni yaklaşım benzetim ortamında gösterilmiştir.

2. BALIKLARIN ANATOMİSİ VE HAREKETLERİ

Balıkların anatomisi incelendiğinde balıkların vücutlarının baş, gövde ve kuyruk bölümlerinden oluştuğu görülmüştür. Şekil 2.1’ de görüldüğü üzere balıkların baş bölümü üstçene kemiği, altçene kemiği, ağız, göz ve solungaç kapağını içerirken balıkların gövde bölümü sırt, pelvik, göğüs ve anal yüzgeçleri içermektedir. Ayrıca kuyruk bölümünde kuyruk yüzgeci bulunmaktadır [27].



Şekil 2. 1. Balık anatomisi

Balıklar çok farklı ortamlarda yaşadıkları için vücut ve yüzme şekilleri birbirinden farklıdır. Birçok balık türü sudaki hareketlerini vücutlarının ve kuyruklarının kıvrılmasıyla (Body Caudal Fin–BCF) gerçekleştirirken, bazıları ise orta ve çift yüzgeçlerinin (Median Paired Fin-MPF) hareketleriyle sağlamaktadır [24]. Balıklar suyun içerisindeki hareketini BCF ve MPF yüzme hareketlerini birlikte gerçekleştirerek sağlarlar. Balıkların ileri yönde hareket etmesi sırasında yer değiştirme özelliği dönüş esnasındaki manevra yapma özelliğinden daha etkili olduğu için BCF yüzme şekli çok daha fazla sayıda balıkta görülmektedir. BCF türü yüzme şekli Anguilliform, Subcarangiform, Carangiform ve Thunniform olmak üzere 4 farklı yüzüş modunu içermektedir [25]. Bu çalışmada kullanılan robot balık Carangiform yüzüş moduna sahip olan bir balık model alınarak oluşturulmuştur.

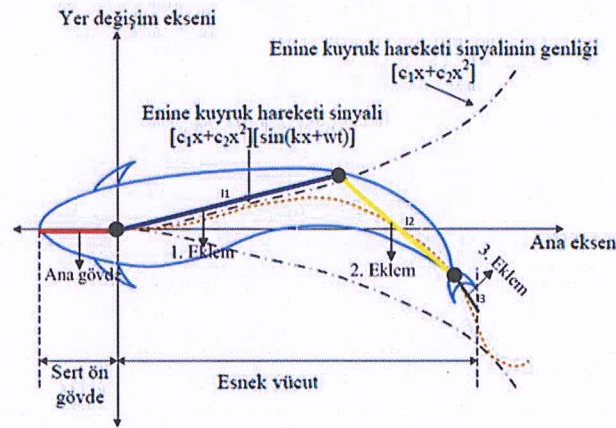
Balıklar 3 ana eksen üzerinden 3 hareket gerçekleştirirler. Bunların birincisi balığın ağırlık merkezinden geçen ve burnundan kuyruğa doğru uzanan eksen etrafında yaptığı yalpalama (Roll) hareketidir. İkincisi yine ağırlık merkezinden geçen ve balığın sağından soluna doğru uzanan eksen etrafında yaptığı yunuslama (Pitch) hareketidir. Sonuncusu ise

yine ağırlık merkezinden geçen ve balığın üst kısmından alt kısmına uzanan eksen etrafında yaptığı sapma (Yaw) hareketidir.

2.1. Robot Balığın Matematiksel Modeli

Bir robot balığın zamana bağlı olarak yaptığı hareketi anlayabilmek için ilk olarak robot balığın matematiksel modelinin oluşturulması gerekir. Robot balığın matematiksel modeli robot balığın ağırlığı, akışkanın robot balığa gösterdiği direnç ve robot balığın kuyruğunun çırpınma hareketleri dikkate alınarak elde edilmelidir [29].

Şekil 2.2' de gösterildiği gibi Carangiform türü bir robot balık bir ana gövde ve üç eklemi içeren esnek kuyruktan oluşmaktadır. Robot balığın düz yüzüş hareketinin büyük bir bölümünü kuyruk yüzgecinin çırpınma hareketi sağlamaktadır. Bu hareket sinyali Şekil 2.2' de verilmiştir.



Şekil 2. 2. Kuyruk hareket sinyali[29]

Sinyalin genliği burundan başlayarak kuyruk boyunca artarak oluşmaktadır. Bu enine hareket sinyali Denklem (2.1) ile ifade edilir [29,30].

$$\left\{ \begin{array}{l} y_{body}(x,t) = (c_1x + c_2x^2) \sin(kx + wt) \\ w = 2\pi f \\ k = 2\pi / \lambda \end{array} \right\} \quad (2.1)$$

Burada; y_{body} kuyruğun zamana (t) ve x bağlı olarak değişen enine hareketi, x ana eksen boyunca yer değiştirme miktarı, t zaman, c_1 doğrusal dalga genliği, c_2 karesel dalga

genliği, w hareket sinyalinin açısal hızı ve k dalga katsayısını ifade etmektedir. Hareket sinyalinin açısal hızı balık kuyruğunun çırpınma frekansına (f) bağlıdır ve dalga sayısı ise hareket sinyalinin dalga boyuna (λ) bağlıdır.

Enine hareket sinyalinin gerçekleştirilmesi için robot balık 3 eklemli tasarlanmış ve eklemleri hareket ettirebilmek için servo motorlar kullanılmıştır. Servo motorlara uygun açı değerleri verilerek robot balığın enine hareketi sağlanmış olacaktır. Bundan dolayı, 3 eklem için gerekli olan açılar enine hareket sinyali ayrık formda tekrar düzenlenerek elde edilir [30]. Denklem (2.2)' de ayrıklaştırılmış enine hareket sinyali verilmiştir.

$$y_{body}(x, t) = (c_1x + c_2x^2) \sin(kx + \frac{2\pi}{M}i) \quad (2.2)$$

Burada; M ayrıklaştırılmış kuyruk hareketine ait örnek sayısını temsil etmektedir.

Fossen' nin yaptığı çalışmalara [32-33] göre; sualtında hareket edebilen sistemlerin 3 boyutlu denklemlerinin genel ifadesi Denklem (2.3)' deki gibi verilmektedir.

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau_f \quad (2.3)$$

Burada; M , $C(v)$, $D(v)$, $g(\eta)$ ve τ_f sembolleri sırasıyla kütle matrisini, coriolis matrisini, hidrodinamik sönümleme matrisini, yerçekimi vektörünü ve hidrodinamik vektörü belirtmektedir. M değeri, gövde matrisi (M_{RB}) ve ek-kütle matrisinin (M_A) toplanmasıyla bulunmaktadır. M_{RB} ve M_A değerleri Denklem (2.4), (2.5)' de gösterildiği gibi hesaplanmaktadır [32].

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & 0 \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & 0 & -mz_G & 0 \\ 0 & -mz_G & 0 & I_{xx} & -I_{xy} & -I_{xz} \\ mz_G & 0 & -mx_G & -I_{yx} & I_{yy} & -I_{yz} \\ 0 & mx_G & 0 & -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \quad (2.4)$$

$$M_A = \begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & Y_{\dot{r}} \\ 0 & 0 & Z_{\dot{w}} & 0 & Z_{\dot{q}} & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & M_{\dot{w}} & 0 & M_{\dot{q}} & 0 \\ 0 & N_{\dot{v}} & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \quad (2.5)$$

$C(v)$ değeri, gövde coriolis matrisi ($C_{RB}(v)$) ve hidrodinamik coriolis matrisinin ($C_A(v)$) toplanmasıyla bulunmaktadır. Denklem (2.6), (2.7)' de gösterildiği gibi sırasıyla $C_{RB}(v)$ ve $C_A(v)$ değerleri hesaplanmaktadır [33].

$$C_{RB}(v) = \begin{bmatrix} 0_{3 \times 3} & -S(M_{11}v_1 + M_{12}v_2) \\ -S(M_{11}v_1 + M_{12}v_2) & -S(M_{21}v_1 + M_{22}v_2) \end{bmatrix} \quad (2.6)$$

$$C_A(v) = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \quad (2.7)$$

$D(v)$, doğrusal (D_L) ve doğrusal olmayan (D_{NL}) sönümleme etkilerini içeren hidrodinamik sönümleme matrisini ifade eder ve Denklem (2.8), (2.9)' daki gibi hesaplanmıştır [33].

$$D_L = \begin{bmatrix} u_s & 0 & 0 & 0 & 0 & 0 \\ 0 & v_s & 0 & 0 & 0 & 0 \\ 0 & 0 & w_s & 0 & 0 & 0 \\ 0 & 0 & 0 & p_s & 0 & 0 \\ 0 & 0 & 0 & 0 & q_s & 0 \\ 0 & 0 & 0 & 0 & 0 & r_s \end{bmatrix} \quad (2.8)$$

$$D_{NL} = \begin{bmatrix} u_D & 0 & 0 & 0 & 0 & 0 \\ 0 & v_D & 0 & 0 & 0 & 0 \\ 0 & 0 & w_D & 0 & 0 & 0 \\ 0 & 0 & 0 & p_D & 0 & 0 \\ 0 & 0 & 0 & 0 & q_D & 0 \\ 0 & 0 & 0 & 0 & 0 & r_D \end{bmatrix} \quad (2.9)$$

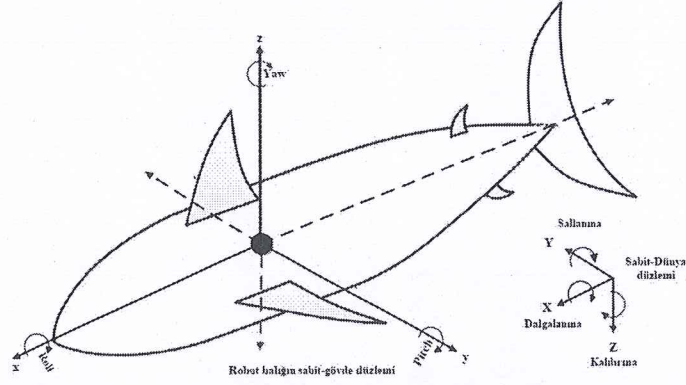
Bu modelde yerçekimi ve kaldırma kuvveti merkezlerinin konumları farklı olduğundan dolayı hidrostatik etki $g(\eta)$ oluşur ve $g(\eta)$ değeri Denklem (2.10)'daki gibi hesaplanır [32].

$$g(\eta) = \begin{bmatrix} (W - B)s(\theta) \\ -(W - B)c(\theta)s(\phi) \\ -(W - B)c(\theta)c(\phi) \\ -(y_g W - y_b B)c(\theta)c(\phi) + (z_g W - z_b B)c(\theta)s(\phi) \\ -(z_g W - z_b B)s(\theta) + (x_g W - x_b B)c(\theta)c(\phi) \\ -(x_g W - x_b B)c(\theta)s(\phi) - (y_g W - y_b B)s(\theta) \end{bmatrix} \quad (2.10)$$

Burada; W , B , $c()$, ve $s()$ sembolleri sırasıyla yerçekimi kuvveti, kaldırma kuvveti, açılarının kosinüs değeri ve açılarının sinüs değerini belirtmektedir.

2.2. Robot Balığın Doğrusal Modeli

Bu çalışmada robot balığın yönlendirilmesini kontrol etmek için 6 serbestlik dereceli gövde hareketi denklemleri kullanılmıştır. Bu denklemler robot balığın, iki koordinat düzlemindeki dinamik ve kinematik özellikleri göz önüne alınarak hesaplanmıştır. Hareketli koordinat düzlemi, robot balığın gövdesine sabitlenmiş ve sabit-gövde düzlemi olarak ifade edilmiştir. Hareketli koordinat düzleminin başlangıcı, yerçekiminin merkezi olarak belirlenmiştir. Ayrıca, sabit-gövde düzlemi harekesiz düzlem olan sabit-dünya düzlemine bağlıdır. Şekil 2.3' de robot balığın dünya düzlemindeki 6 serbest dereceli hareketi gösterilmiştir.



Şekil 2. 3. Robot balığın hareketleri ve düzlemleri

Robot balığın kontrolü 2 kısımda gerçekleştirilmiştir. Bunlar; hız kontrolü ve yön kontrolüdür. Hız kontrolü robot balığın hızını kontrol etmek için kullanılırken yön kontrolü robot balığın yaw hareketinin kontrolünü sağlamak için kullanılır. Bunların kontrolü oransal-integral-türev denetleyici (PID) kontrol döngüsü yöntemi ile sağlanmıştır. Bir PID denetleyici Denklem (2.11)' daki gibi tanımlanır.

$$\begin{cases} u(t) = K_p \left\{ e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \right\} \\ e(t) = v(t) - y(t) \end{cases} \quad (2.11)$$

Burada, K_p , çıkış sinyalinin genliğini kontrol eden fakat ofset hatasını neden olan oransal kazancı ifade etmektedir. Ofset hatasını giderilmesi için uygun bir integral kazancı (K_i) ve aşmaları azaltmak için ise türevsel kazancı (K_d) ifade etmektedir. $e(t)$, giriş sinyali ile çıkış sinyali arasındaki hatayı ifade etmektedir. $v(t)$ ve $y(t)$ sırasıyla giriş sinyalini ve çıkış sinyalini belirtir. Hız ve yön kontrolü için en uygun PID katsayıları Tablo 2.' de verilmiştir.

Tablo 2. PID parametreleri

PID katsayıları	Hız Kontrolü	Yön kontrolü
K_p	50.775	104.765
K_i	46.365	164.553
K_d	3.0373	9.6990

PID parametrelerinin optimum değerlerinin bulunabilmesi için Ziegler-Nichols yöntemi kullanılmıştır.

Robot balığın hız kontrolü için oluşturulan model: Robot balığın hızının doğrusallaştırılmış modeli Denklem (2.12)' deki gibi gösterilmektedir [36].

$$\begin{cases} \dot{u} = a_u u + b_u A_T \\ y_u = c_u u \end{cases} \quad (2.12)$$

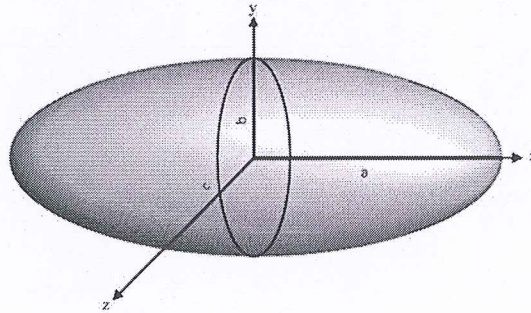
Burada; $a_u = -u_s / (m + X_u)$, $b_u = \alpha T / (m + X_u)$ ve $c_u = 1$ şeklinde hesaplanmıştır. u_s , m , αT ve X_u sembolleri sırasıyla doğrusal sönümleme katsayısı, balığın ağırlığı, itme kuvvetinin kazancı ve dalgalanma hareketindeki ek kütesini ifade etmektedir. Ek kütle matrisindeki X_u değeri Denklem (2.13), Y_v ve Z_w değerleri Denklem (2.14) ve N_r ve M_q değerleri Denklem (2.15) gibi hesaplanmıştır.

$$X_u = \frac{-a_0}{2 - a_0} m \quad (2.13)$$

$$Y_v = Z_w = -\frac{\beta_0}{2 - \beta_0} m \quad (2.14)$$

$$N_r = M_q = -\frac{1}{5} \frac{(b^2 - a^2)^2 (a_0 - \beta_0)}{2(b^2 - a^2) + (b^2 + a^2)(\beta_0 - a_0)} m \quad (2.15)$$

Robot balık Şekil 2.4' deki gibi elips şeklindeki bir kütle olarak kabul edilmiştir.



Şekil 2. 4. Elips şeklindeki bir kütle

e sembolü merkez kaçıklığı ifade etmektedir ve Denklem (2.16) gibi hesaplanmaktadır. a_0 ve β_0 sembolleri robot balığın şekline göre değişen katsayılarını belirtmektedir. a_0 ve β_0 değerleri merkez kaçıklığına bağlıdır ve sırasıyla Denklem (2.17), (2.18)' deki gibi hesaplanır.

$$e = 1 - (b/a)^2 \quad (2.16)$$

$$a_0 = \frac{2(1-e^2)}{e^3} \left(\frac{1}{2} \ln \frac{1+e}{1-e} - e \right) \quad (2.17)$$

$$\beta_0 = \frac{1}{e^2} - \frac{1-e^2}{2e^3} \ln \frac{1+e}{1-e} \quad (2.18)$$

Yukarda verilen denklemler kullanılarak robot balığın hız kontrolü sağlanmıştır. Robot balığın yön kontrolü için oluşturulan MATLAB/Simulink modeli EK-4' de verilmiştir.

Robot balığın yön kontrolü için oluşturulan model: Robot balığın yönünün doğru-sallaştırılmış modeli, Denklem (2.19)' daki gibi durum uzay modeli şeklinde belirtilmektedir.

$$\begin{cases} \dot{d} = A_r d + B_r \phi_r \\ y_r = C_r d \end{cases} \quad (2.19)$$

Burada; A_r , B_r ve C_r değerleri Denklem (2.20), (2.21) ve (2.22)' deki gibi hesaplanmıştır.

$$A_r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & I_{zz} + N_{\dot{r}} & N_{\dot{v}} \\ 0 & Y_{\dot{r}} & m + Y_{\dot{v}} \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & 0 \\ 0 & -r_s & 0 \\ 0 & 0 & v_s \end{bmatrix} \quad (2.20)$$

$$B_r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & I_{zz} + N_{\dot{r}} & N_{\dot{v}} \\ 0 & Y_{\dot{r}} & m + Y_{\dot{v}} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ b_r \\ b_v \end{bmatrix} \quad (2.21)$$

$$C_r = [c_r \quad 0 \quad 0] \quad (2.22)$$

$$d = [\psi \quad r \quad v]^T \quad (2.23)$$

A_r ve B_r hesaplanmasında kullanılan I_{zz} , $N_{\dot{r}}$, $N_{\dot{v}}$, $Y_{\dot{r}}$, $Y_{\dot{v}}$ değişkenleri Denklem (2.4), (2.5) de belirtilen kütle matrislerindeki değişken değerleri kullanılmıştır ve Denklem (2.20), (2.21) ve (2.22)' deki denklemler ile hesaplanmıştır. Denklem (2.2)' deki c_r değeri 1 olarak kabul edilmiştir. r_s ve v_s hidrodinamik doğrusal sönümleme matrisindeki değişkenlerdir ve Denklem (2.8)' deki matris değerleri kullanılmıştır. Denklem (2.23)' de b_r , b_v , ψ ve r sırasıyla dönme kazançlarını, sallanma kazançlarını, yaw açısını ve yaw açısal hızını belirtmektedir. Robot balığın yön kontrolü için oluşturulan MATLAB/Simulink modeli EK-5' de verilmiştir. Ayrıca dinamik modelini oluşturduğumuz robot balığın, benzetim çalışmaları için kullanılan parametreler Tablo 3' de verilmiştir.

Tablo 3. İki eklemli robot balığın benzetim çalışma parametreleri

<i>Parametre</i>	<i>Değer</i>
Ön Gövde Ağırlığı	0.28826 kg
1. Kuyruk Eklemi Ağırlığı	0.19552 kg
2. Kuyruk Eklemi Ağırlığı	0.19552 kg
Kuyruk Yüzgeci Ağırlığı	0.15 kg
Ön Gövde Uzunluğu	0.210 m
1. Kuyruk Eklemi Uzunluğu	0.170 m
2. Kuyruk Eklemi Uzunluğu	0.170 m
Kuyruk Yüzgeci Uzunluğu	0.100 m
Su Yoğunluğu	998 kg/m ³
Çarpınma Frekansı	2 Hz

Bütün bu alınan değerler ve hesaplamalar sonucunda robot balığın dinamik modeli ve kontrolü MATLAB programında oluşturulmuştur ve EK-3' de verilmiştir.

3. YOL PLANLAMA ALGORİTMALARI

Yapısal olarak arama algoritmalarını iki grupta toplamak mümkündür. Bu arama algoritmalar aşağıda kısaca açıklanmaktadır.

Bilgili arama algoritmaları (Informed Search): Arama işleminin bilgili olması algoritmanın çalışma alanına ait bazı özellikleri bünyesinde bulundurması demektir ve bu özelliklerden dolayı bilgili arama algoritmaları çalışma ortamlarına göre değişiklik göstermektedir. Bu algoritmaların bazıları aşağıdaki gibi listelenebilir:

- Minimax ağaçları
- Tepe tırmanma algoritması (Hill Climbing Algorithm)
- Arı sürüsü arama algoritması (Bees Search Algorithm)
- Geri izleme algoritması (Backtracking)
- Işın arama algoritması (Beam Search)
- A* araması algoritması (Astar Search)

Bu çalışmada yukarıdaki algoritmalar içerisinde robot balıkların en kısa yolu bulması için kullanılan sezgisel algoritmalarından biri olan A* algoritması Bölüm 3.1' de detaylı bir şekilde açıklanmıştır.

Bilgisiz arama algoritmaları (Uninformed Search): Bu tür arama algoritmaları çalışma alanı için herhangi bir özelliğe ihtiyaç duymaz ve bütün çalışma alanlarında aynı şekilde çalışır. Bilgisiz arama algoritmaları aşağıdaki gibi sıralayabiliriz [26]:

- Listeler (arrays) üzerinde çalışan arama algoritmaları
- Metin arama algoritmaları (bir yazı içerisinde belirli bir karakter dizisini (string) arayan algoritmalar)
- Şekiller (graphs) üzerinde çalışan algoritmalar

Dizi, içerisinde birden fazla veriyi depolayabilen değişkenlerdir. Diziler üzerinde çalışan algoritmalar tek boyutlu, iki boyutlu veya üç boyutlu şeklinde tanımlayabiliriz. Metin arama algoritmaları bir yazı içerisinde belirli bir diziye (string) arayan algoritmalarlardır. Graf bir olay veya ifadenin düğüm ve çizgiler kullanılarak gösterilmesine olanak sağlayan bir veri modelidir. Graftaki her bir noktaya düğüm ve çizgilere kenar adı verilmektedir. Bir graf üzerindeki iki düğümü birbirine bağlayan birden fazla yol bulunabilir. Bu yolların her birinin kendine ait birer maliyet değeri bulunmaktadır. Böyle

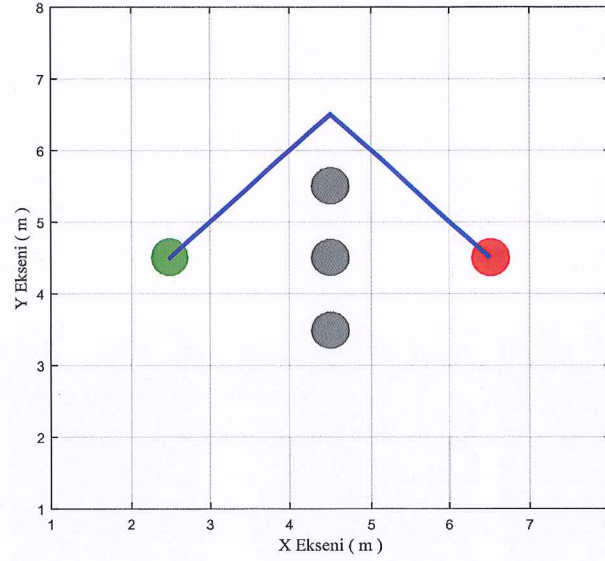
bir yapı içerisinde bir düğümden bir başka düğüme en kısa yolu kullanarak ulaşmak önemli konulardan biridir. En kısa yola ulaşmak için maliyet değeri olarak en küçük olan kullanılır. Graflar üzerinde çalışabilen arama algoritmalarından bazıları aşağıdaki şekilde sıralanabilir [26]:

- Floyd-Warshall algoritması
- Bellman Ford algoritması
- Prim's algoritması
- İkili arama ağacı (Binary Search Tree) algoritması
- Kruskal algoritması
- Dijkstra algoritması
- Alt hedef graf algoritması

Bu çalışmada yukarıdaki algoritmalar içerisinde en çok kullanılan algoritmalarından biri olan Alt hedef graf algoritması Bölüm 3.2' de detaylı bir şekilde incelenmiştir.

3.1. A* Yol Planlama Algoritması

A* algoritmasının temelini, 1964 yılında Dijkstra algoritmasının hızını artırmak için sezgisel bir algoritma bulan Nils Nilsson atmıştır. Nilson'un bu algoritması A1 algoritması olarak bilinmektedir. 1967 yılında Bertram Raphael, A1 algoritmasının maliyetini azalmak için çalışmalar yapmış fakat başarısız olmuştur. Raphael, başarısız olduğu algoritmasına A2 algoritması ismini vermiştir. 1968 yılında ise Peter Hart, A2 algoritmasında küçük değişiklikler yaparak maliyetini azalmıştır ve bu algoritmaya A* algoritması adı verilmiştir [27]. A* algoritması, 2 boyutlu bir düzlemde iki nokta arasında engelleri aşarak en kısa mesafeyi bulmada kullanılan sezgisel bir yol planlama algoritmasıdır. Şekil 3.1' de gösterildiği gibi başlangıç noktası yeşil (●), bitiş noktası kırmızı (●) ve engeller gri (●) renk ile temsil edilmektedir.



Şekil 3. 1. 6X6' lık A* algoritmasının çalışma ortamı

A* algoritmasının birinci aşaması arama yapılacak alanın hürelere bölünmesidir. Bu hücre şekilleri arama yapılan alana göre kare, üçgen, altıgen veya yamuk şeklinde seçilebilmektedir. Yaptığımız çalışmada arama alanı, 1×1 ' lik kare şeklindeki hürelere bölünerek gerçekleştirilmiştir. Bu kare hücrelerin merkez noktalarına (düğüm) adı verilmektedir. Ayrıca A* algoritması sezgisel bir algoritma olduğu için her bir düğümün bitiş noktasına olan uzaklığının hesaplanmasıyla bulunan tahmini değer (H) farklı fonksiyonlar ile hesaplanabilmektedir [31]. Bunlar Manhattan, Euclidean, Chebyshev ve Octile fonksiyonlardır ve bu fonksiyonlar sırasıyla aşağıdaki Denklemler (3.1), (3.2), (3.3), (3.4) ile hesaplanmaktadır.

$$H(n) = (|x_n - x_t| + |y_n - y_t|) * 10 \quad (3.1)$$

$$H(n) = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2} \quad (3.2)$$

$$H(n) = \max(|x_n - x_t|, |y_n - y_t|) \quad (3.3)$$

$$H(n) = \max(|x_n - x_t|, |y_n - y_t|) + (\sqrt{2} - 1) * \min(|x_n - x_t|, |y_n - y_t|) \quad (3.4)$$

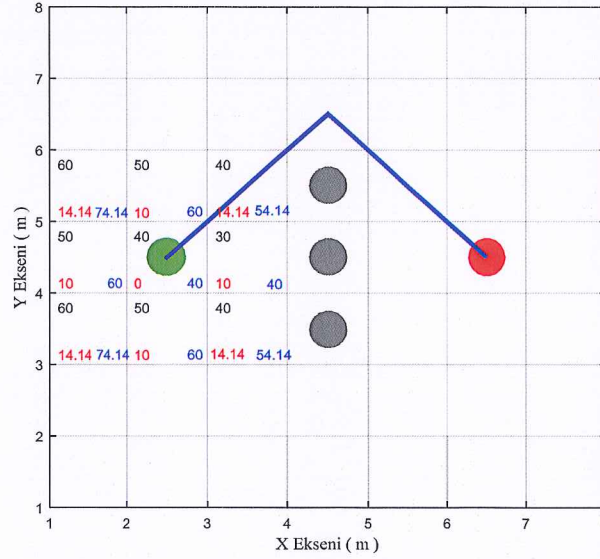
Bu çalışmada, tahmini değer hesaplanırken Manhattan fonksiyonu kullanılmıştır. Daha sonra başlangıç düğümünün komşuları olan 8 düğüme uzaklıkları Denklem (3.5) kullanılarak hesaplanır ve G değerleri elde edilir. Denklem (3.1) kullanarak da H değeri

hesaplanır. Bu H ve G değerleri hesaplandıktan sonra bu değerlerin toplanması sonucunda Denklem (3.6)'daki gibi F değeri bulunmaktadır.

$$G(n) = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2} * 10 \quad (3.5)$$

$$F(n) = G(n) + H(n) \quad (3.6)$$

Şekil 3.1.'de kırmızı renkli sayılar G değerlerini, siyah renkli sayılar H değerlerini ve mavi renkli sayılar F değerini göstermektedir.



Şekil 3. 2. A* algoritmasının G, H ve F değerleri

Başlangıç düğümünün çevresindeki engel olmayan komşu düğümler açık listeye eklenir. G ve H değerlerine göre en küçük F değerine sahip düğüm tekrardan 8 komşu düğüme bakılır ve eğer bu komşu düğümler açık listede yoksa eklenir. Eğer var ise bu düğümün açık listedeki düğümlere uzaklığı tekrardan Denklem (3.5)'e göre hesaplanır ve kendi G değeri ile toplanarak bir sayısal değer bulunur. Bu değer komşu düğümün G değerinden küçükse yada eşitse bu düğüm kapalı listeye eklenir ve bu düğüm üzerinden yola devam edilir. Değilse daha sonraki en küçük F değerine sahip olan düğüm seçilerek tekrar işlem yapılır. Bu işlemler bitiş düğüme ulaşınca kadar devam eder ve kapalı listedeki düğümler takip edilerek en kısa yol hesaplanmış olur. Algoritmanın çalışma adımları aşağıda sunulmuştur:

- Adım 1:**Algoritmanın çalıştırılacağı alana göre $n \times n$ ' lik bir matris oluşturulur.
- Adım 2:**Kullanıcı tarafından belirlenen bitiş, engel ve başlangıç düğümleri çalışma alanı matrisinde sırasıyla 1,-1 ve 0 ile gösterilir.
- Adım 3:**Kapalı liste ve açık liste adında $n \times n$ ' lik matrisler oluşturulur.
- Adım 4:**Başlangıç düğümü açık listeye eklenir.
- Adım 5:**Başlangıç düğümünden bitiş düğümüne olan uzaklığın tahmini değeri $H(n)$ hesaplanır.
- Adım 6:**Başlangıç düğümünün etrafında engelin olmadığı düğümler açık listeye eklenir.
- Adım 7:**Eklenmiş bu düğümlerin başlangıç düğümüne olan uzaklığı $G(n)$ hesaplanır. Başlangıç düğümü kapalı listeye eklenir.
- Adım 8:**Dizide açık listede bulunan düğümlerin $H(n)$, $G(n)$ ve $F(n)$ değerleri hesaplanarak yeni bir matriste değerleri tutulur. $F(n)$ değeri hesaplanır.
- Adım 9:**Açık listedeki düğümlerin etrafında engel olmayan düğümlerin $H(n)$, $G(n)$ ve $F(n)$ değerleri başka bir matriste tutulur. $F(n)$ değeri önceki bulunan değere eklenerek hesaplanır.
- Adım 10:** $F(n)$ değerlerinin bulunduğu matrislere bakılarak en küçük değerli olan düğüm açık listeden çıkarıp kapalı listeye eklenir. Bu döngü bulunan düğümler hedef düğüme ulaşmaya kadar devam etmektedir.
- Adım 11:** Başlangıç ve bitiş düğümleri arasındaki yolun maliyeti hesaplanır.

Yukarıdaki adımlar dikkate alınarak MATLAB R2015a programında A* algoritmasının kodu EK-1' de gösterildiği gibi yazılmıştır. Yazılan algoritmanın doğrulanması için birçok uygulama gerçekleştirilmiştir. Bu uygulamalardan birkaçı Bölüm 4.1' de detaylı olarak verilmiştir.

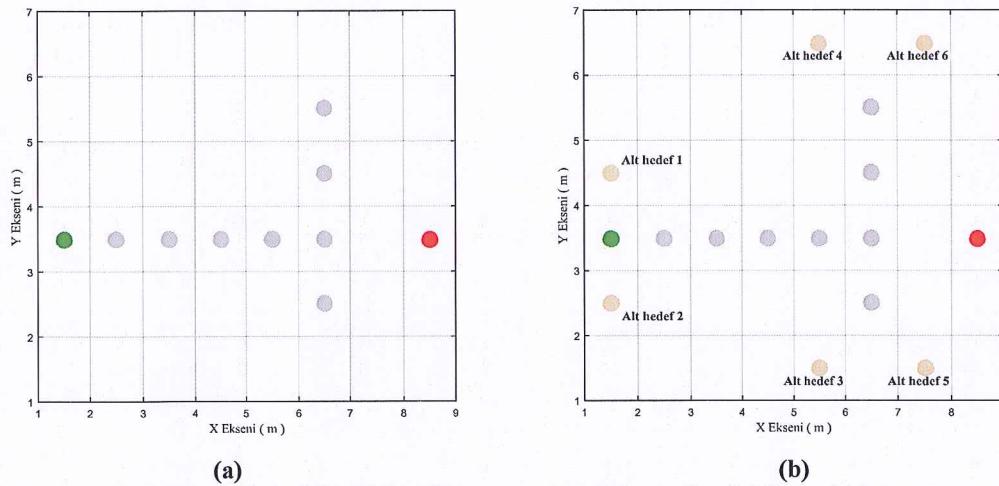
3.2. Alt Hedef Graf (Subgoals Graphs) Algoritması

Son yıllarda yol planlama için Alt hedef graf algoritması kullanılmaya başlanmıştır. Uras ve ark.[37] yaptıkları çalışmada, 2012 ve 2013 yılları arasında düzenlen ızgara tabanlı yol planlama yarışmasında en uygun yolu 2 seviyeli alt hedef algoritmasıyla bulmuşlardır. 2 seviyeli alt hedef algoritması, StarCraft oyununda 1024×1024 harita üzerinde en kısa yolu, A* algoritmasından 25 kat daha hızlı bir şekilde bulmaktadır. Ayrıca A* algoritmasıyla bulunan gereksiz noktaları atlayarak en kısa yolu bulan bir algoritmadır. Bir başka çalışmalarında ise [38], en kısa yolu bulmak için hiyerarşi bir yapıya sahip N-seviyeli graf algoritmasını kullanmışlardır.

Robotik sistemlerin başlangıç noktasından bitiş noktasına engelleri aşarak gitmesi esnasında sistem dinamiğinin dikkatte alınması gerekmektedir. Bu yüzden optimum yol hesaplanırken alt hedef graf algoritmaları kullanılmaktadır.

Şekil 3.3.a' da verilen başlangıç (●), bitiş (●) ve engellerin (●) bulunduğu bir alan üzerinde Alt hedef graf algoritması adımları aşağıda anlatılmaktadır.

Adım 1: Şekil 3.3.b' de gösterilen engellerin bulunduğu duvarların köşe noktaları belirlenir ve alt hedef adı verilir.

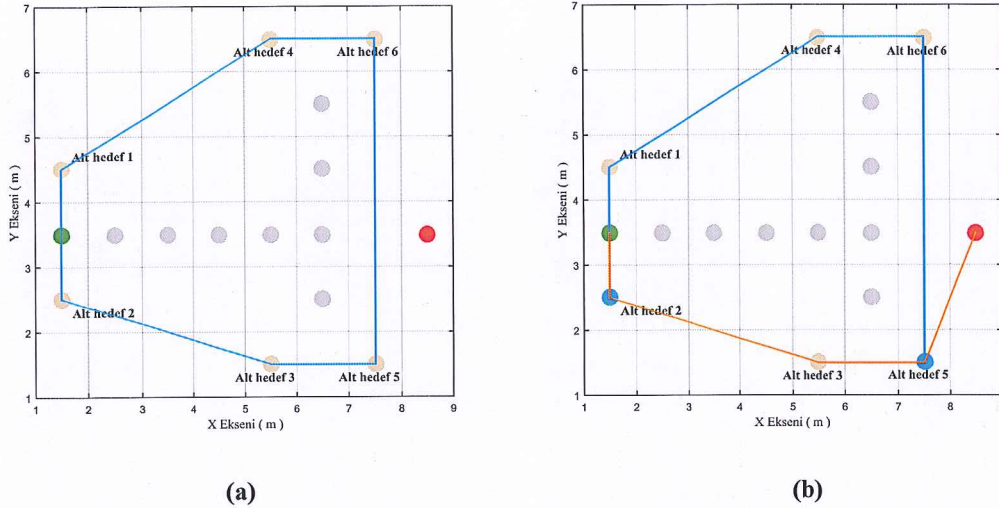


Şekil 3. 3. Alt hedef graf algoritmasının a.) Uygulama alanı b.) Birinci adımı

Adım 2: Alt hedeflerin birbiriyle direk ulaşılacak düğümler arası Şekil 3.4.a' daki gibi bağlantıları sağlanır.

Adım 3: Başlangıç ve bitiş noktaları kendilerine en yakın alt hedefler seçilir ve bağlantıları yapılır. Alt hedefler arasında hareket edildiğinde yolun maliyet değeri Denklem 3.7’ deki gibi hesaplanır.

$$subgoldist = (\sqrt{2} * (\min(|x1 - x2|, |y1 - y2|)) + (|x1 - x2| + |y1 - y2|)) \quad (3.7)$$



Şekil 3. 4. Alt hedef graf algoritmasının a.) İkinci adımı b.) Üçüncü adımı ve en kısa yol

Böylece sistemlerin dinamiğine uygun olarak en kısa yol Şekil 3.4.b’ de gösterilen turuncu çizgiler ile hesaplanır. Yukarıdaki adımlar dikkate alınarak MATLAB programında Alt hedef graf algoritmasının kodu EK-2’ de gösterildiği gibi yazılmıştır. Yazılan algoritmanın doğrulanması için birçok uygulama gerçekleştirilmiştir. Bu uygulamalardan birkaçı Bölüm 4.2’ de detaylı olarak verilmiştir.

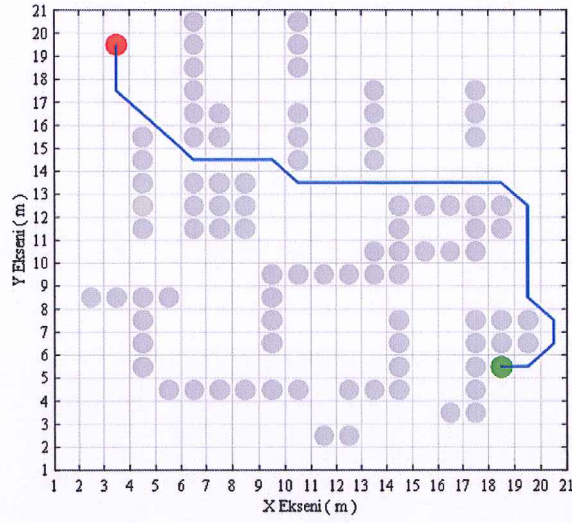
4. BENZETİM ÇALIŞMALARI

Bu bölümde arama alanının robot balığın hareket edeceği havuz olarak kullanılması planlanmıştır. 2 boyutlu bir düzlem içerisinde ve bir başlangıç noktası ve bitiş noktası olmak üzere iki nokta vardır ve bu iki noktayı birbirine bağlayan birden fazla yol bulunabilmektedir. Bu yolların her birinin birer maliyet değeri de bulunmaktadır. Böyle bir yapı içerisinde bir noktadan bir başka noktaya en kısa yolu kullanarak ulaşmak önemli bir problemdir. Bu amaca ulaşmak için maliyet değeri olarak en küçük olan yolların kullanımı tercih edilir. Bu bölümde literatürde en çok adı geçen bazı en kısa yol algoritmalar ile ilgili bazı uygulamalar MATLAB programında gerçekleştirilmiştir.

4.1. A* Uygulamaları

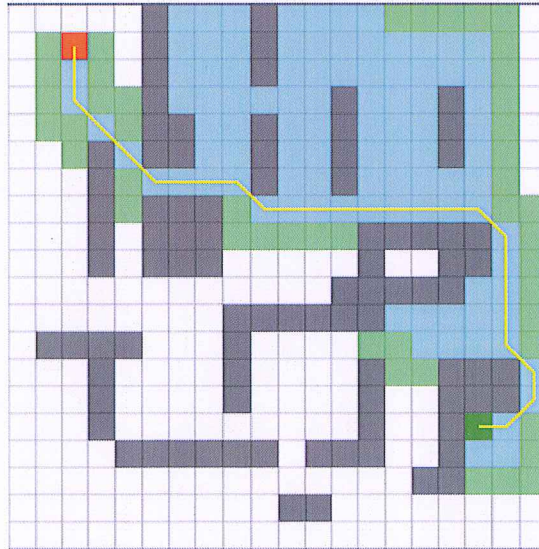
Bu bölümde robot balığın engellerden kaçarak en kısa yolu bulabilmesi için MATLAB2015a programında A* algoritması yazılmıştır ve bu algoritma ile 6 uygulama gerçekleştirilmiştir. Bu uygulamalarda başlangıç noktasını (●), bitiş noktasını (●) ve engelleri (●) temsil etmektedir. Bu uygulamaların tamamı 2 boyutlu düzlemde gerçekleştirilmiş ve çalışma alanları $1x1$ ' lik kare hücrelere bölünmüştür. 6 uygulama aşağıda detaylı bir şekilde açıklanmıştır.

Birinci ve ikinci uygulamalarda başlangıç noktası, bitiş noktasının sağında bulunmaktadır. Birinci uygulamada çalışma ortamı olarak $20x20$ ' lik bir alan seçilmiş. Bu ortamda başlangıç noktasının koordinatı $(18,5)$ ve bitiş noktasının koordinatı $(3,19)$ olarak kabul edilmiştir. Şekil 4.1' deki gibi engeller konularak A* algoritması uygulanmıştır.



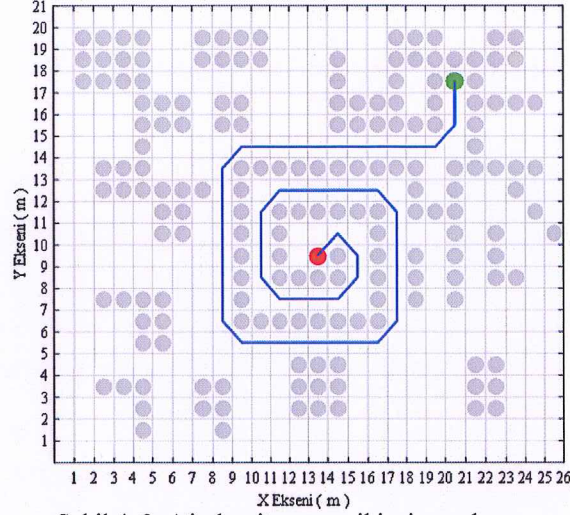
Şekil 4. 1. A* algoritmasının birinci uygulama sonucu

A* algoritması birinci uygulaması çalıştırıldıktan sonra algoritmanın en kısa yol olarak Şekil 4.1.'deki mavi çizgiyi takip ederek ulaşılacağı belirlenmiştir. Bu algoritmanın doğruluğunu ispatlamak için Github programında [33] aynı başlangıç (yeşil kare), bitiş (kırmızı kare) ve engel noktaları (gri kare) seçilerek A* algoritması çalıştırıldığında Şekil 4.2' deki gibi bir sarı renkli yol izlemektedir. Böylece Şekil 4.1 ve Şekil 4.2' deki en kısa yollara bakıldığında aynı yol maliyeti ile aynı yollu izlediği görülmüştür. Sonuç olarak MATLAB programında yazdığımız A* algoritmasının doğru yolu bulduğu belirlenmiştir.



Şekil 4. 2. A* algoritmasının github sonucu

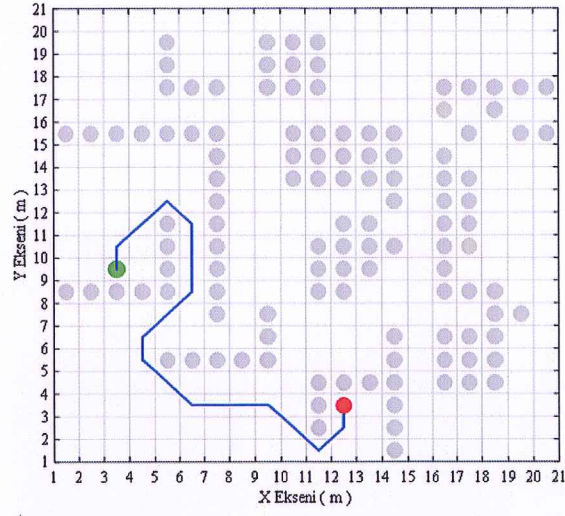
İkinci uygulamada çalışma ortamı olarak 25x20' lik bir alan seçilmiştir. Bu alanda başlangıç noktasının koordinatı (20,17) ve bitiş noktasının koordinatı (13,9) olarak kabul edilmiştir. Şekil 4.3' deki gibi engeller konularak A* algoritması denenmiştir.



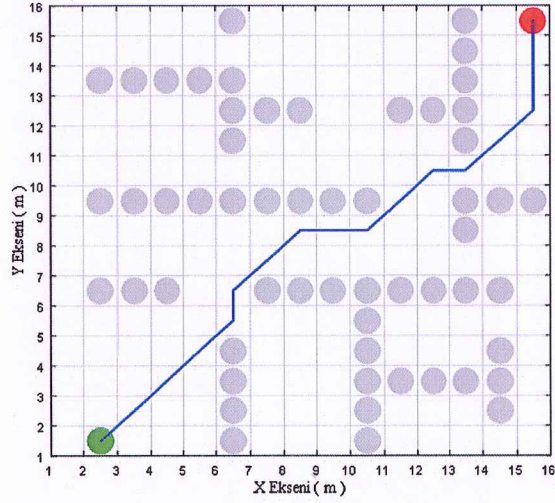
Şekil 4. 3. A* algoritmasının ikinci uygulama

İkinci uygulama sonucunda en kısa yoldan hedefe Şekil 4.3' deki mavi çizgiyi takip ederek ulaşılacağı belirlenmiştir.

Üçüncü ve dördüncü uygulamalarda başlangıç noktası, bitiş noktasının solunda bulunmaktadır. Üçüncü uygulamada çalışma alanı olarak 20x20' lik bir alan seçilmiştir. Başlangıç noktasının koordinatı (3,9) ve bitiş noktasının koordinatı (12,3) olarak kabul edilmiştir. Dördüncü uygulamada ise çalışma ortamı olarak ise 15x15' lik bir alan seçilmiştir. Başlangıç noktasının koordinatı (2,1) ve bitiş noktasının koordinatı (12,3) olarak kabul edilmiştir. Üçüncü ve dördüncü uygulamalara sırasıyla Şekil 4.4 ve Şekil 4.5' deki gibi engeller konularak A* algoritması denenmiştir.



Şekil 4. 4. A* algoritmasının üçüncü uygulama



Şekil 4. 5. A* algoritmasının dördüncü uygulaması

Üçüncü ve dördüncü uygulama sonuçlarında Şekil 4.4 ve Şekil 4.5’ deki gibi en kısa yolu bulmak için mavi çizgilerin takip edileceği görülmüştür.

Beşinci uygulamada ise başlangıç noktasının ve bitiş noktasının değerleri Y ekseninde aynı olup X ekseninde farklı değerler de bulunmaktadır. Çalışma alanı olarak 30x25’ lik bir alan seçilmiştir. Başlangıç noktasının koordinatı (27,15) ve bitiş noktasının koordinatı (14,15) olarak seçilmiştir. Beşinci uygulama için engeller ise Şekil 4.6’ daki gibi yerleştirilerek A* algoritması denenmiştir.

A* algoritmasıyla gerçekleştirilmiş olan altı uygulamanın toplam yol maliyetleri ve çalışma zamanları Tablo 4.' deki gibi bulunmuştur.

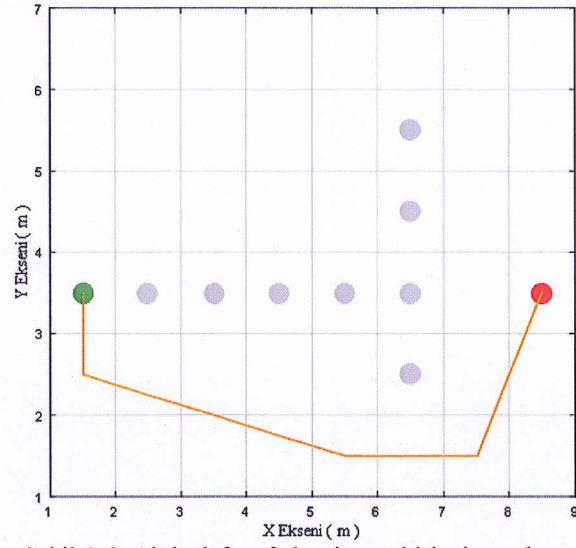
Tablo 4. A* algoritmalarının yol maliyeti ve çalışma zamanı

Uygulama No:	Çalışma Alanı	Başlangıç düğümü	Bitiş düğümü	Yol maliyeti	Zaman (sn)
1	20x20	(18,5)	(3,9)	288,99	8.144665
2	25x20	(20,17)	(13,9)	571.42	16.753765
3	20x20	(3,9)	(12,3)	231.42	5.652438
4	15x15	(2,1)	(15,15)	211.42	4.910316
5	30x25	(27,15)	(3,15)	341.42	8.738101
6	43x36	(1,2)	(21,33)	780.12	19.757767

Yol maliyeti hesaplanırken başlangıç noktasından bitiş noktasına ulaşınca kadar hareket edilen düğüm sayılarına bağlı olduğu belirlenmiştir. Ne kadar fazla düğüm takip edilirse yol maliyeti o kadar yüksektir. Ayrıca çalışma zamanı yol maliyeti ile doğru orantılı olarak artmaktadır.

4.2. Alt Hedef Graf Uygulamaları

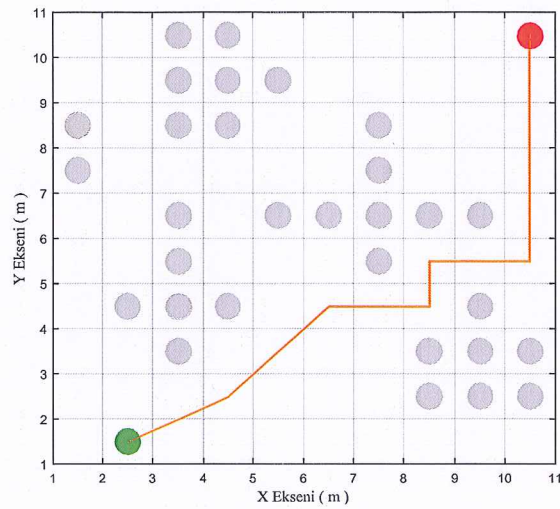
Bu bölümde; birçok sistemin dinamik modelline uygun olan ve yol bulma problemleri için kullanılan Alt hedef graf algoritması MATLAB2015a programında yazılmış ve 3 uygulama gerçekleştirilmiştir. Bu uygulamalardan birincisin de çalışma alanı olarak 8x6' lık küçük bir alan seçilmiştir. Başlangıç konumu (1,3), bitiş konumu (8,3) ve engeller Şekil 4.8' de gösterildiği gibi yerleştirilmiştir.



Şekil 4. 8. Alt hedef graf algoritması birinci uygulama

Şekil 4.8’ de bir sistemin hareket esnasında engellerin köşe noktalarına çarpmaması için engellerin köşelerinden bir düğüm daha uzaktan hareket ettiği turuncu renkli çizgi ile görülmüştür. Böylece küçük alanlarda gerçek zamanlı çalışan sistemler için uygun en kısa yolu bulunmaktadır.

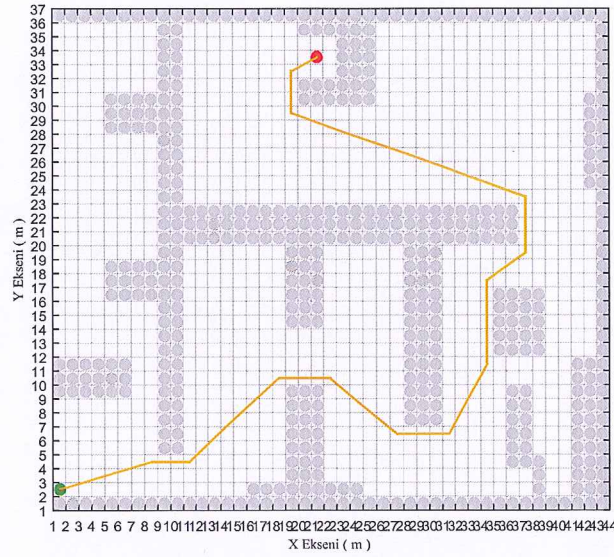
İkinci uygulamada 10×10 ’ luk bir çalışma alanı içerisinde başlangıç konumu $(2,1)$, bitiş konumu $(10,10)$ olarak seçilmiştir. Daha sonra Şekil 4.9’ daki gibi engeller yerleştirilmiştir.



Şekil 4. 9. Alt hedef graf algoritması ikinci uygulama

Böylece Şekil 4.9’ daki gibi turuncu renkli çizginin, yerleştirilen engellerin birbirine yakın olduğu için en kısa yol olduğu ve bu yolun çoğu gerçek sistemin dinamiğine uygun olduğu belirlenmiştir.

Üçüncü uygulamada ise, 43x36 büyük bir alan seçilmiştir. Başlangıç konumu (1,2), bitiş konumu (21,33) ve engeller Şekil 4.10’ da gösterildiği gibi yerleştirilmiştir.



Şekil 4. 10. Alt hedef graf algoritması üçüncü uygulama

Böylece Şekil 4.10’da gösterildiği gibi çalışma alanının büyük olması ve engeller arası mesafenin büyük olması durumunda bulunan en uygun yol turuncu renkli çizgi ile görülmüştür. Ayrıca Alt hedef graf algoritmasıyla yapılan 3 uygulamanın yol maliyet değerleri Tablo 5.’de gösterilmiştir.

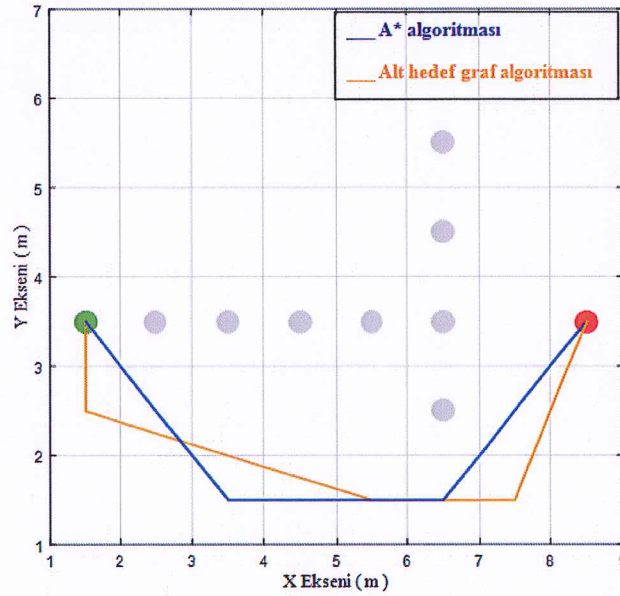
Tablo 5. Alt hedef algoritmalarının yol maliyeti

Uygulama No:	Çalışma Alanı	Başlangıç düğümü	Bitiş düğümü	Yol maliyeti
1	8x6	(1,3)	(8,3)	13.8284
2	10x10	(2,1)	(10,10)	21.2426
3	43x36	(1,2)	(21,33)	129.9411

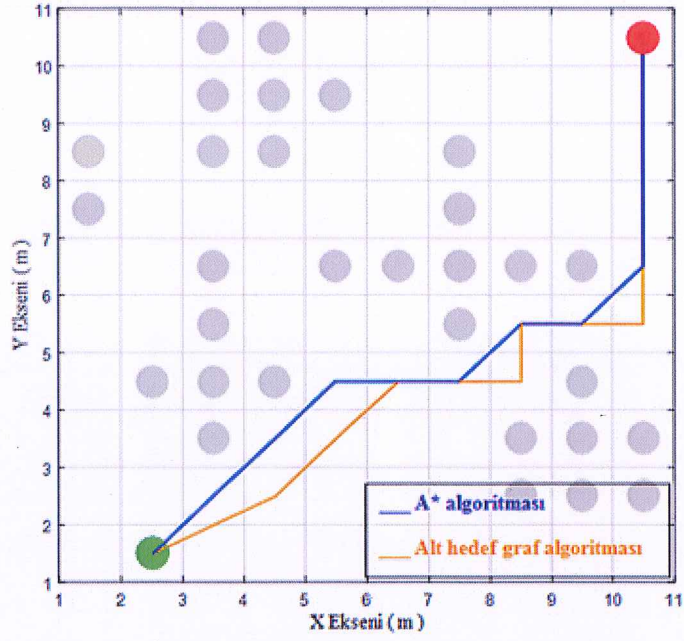
Yol bulma algoritmalarından biri olan A* algoritması başlangıç konumundan hedef konuma en kısa yolu bulmak için kullanılmıştır. Fakat A* algoritmasıyla yol bulma işlemi gerçekleştirilirken robot balığın sistemini dikkate almadan bir yol bulmaktadır. Alt hedef

graf algoritması ise gerçek sistemlerin dinamiğinden dolayı hareket esnasında engellere çarpmadan hedef noktasına ulaşabilsin diye engellerin köşe noktalarının 1 birim uzağındaki düğümlerden hareket ederek sistemin daha güvenli yol olarak hedef noktasına ulaşımı sağlanmıştır.

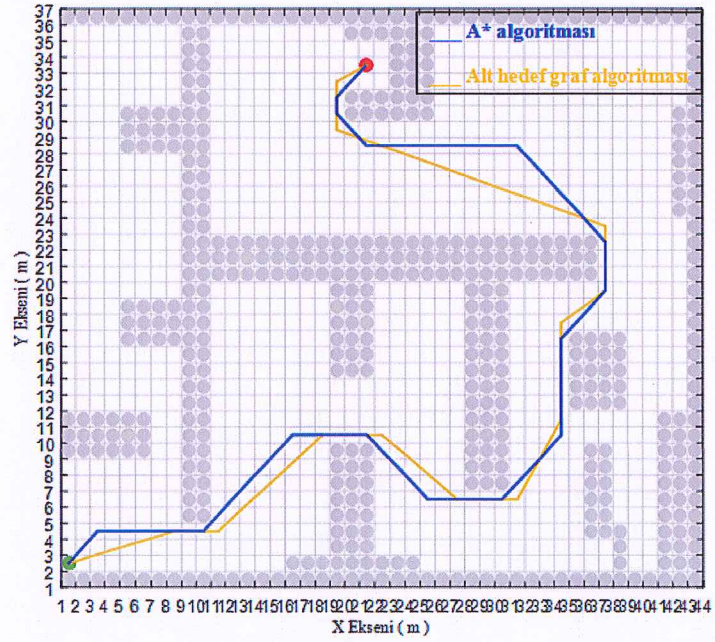
Alt hedef graf algoritması ile yapılan 3 uygulamadaki aynı veriler A* algoritmasına verildiğinde Şekil 4.11, 4.12 ve 4.13’ deki gibi mavi renkli çizgiyi takip ettiği görülmektedir. A* algoritması robot balık için en kısa yolu bulmaktadır fakat robot balık bir düğümden diğer düğüme gidişi esnasında köşegenlerden geçerken çarpma olasılığı bulunduğu için Alt hedef graf algoritması tercih edilmektedir.



Şekil 4. 11. A* ve Alt hedef graf algoritmasının birinci uygulaması



Şekil 4. 12. A* ve Alt hedef graf algoritmasının ikinci uygulaması



Şekil 4. 13.A* ve Alt hedef graf algoritmasının üçüncü uygulaması

A* ve Alt hedef graf algoritmasıyla yapılan 3 uygulamanın en kısa yol uzunluklarının değerleri Tablo 6.'da verilmiştir.

Tablo 6. A* ve Alt hedef graf algoritmasının yol uzunlukları

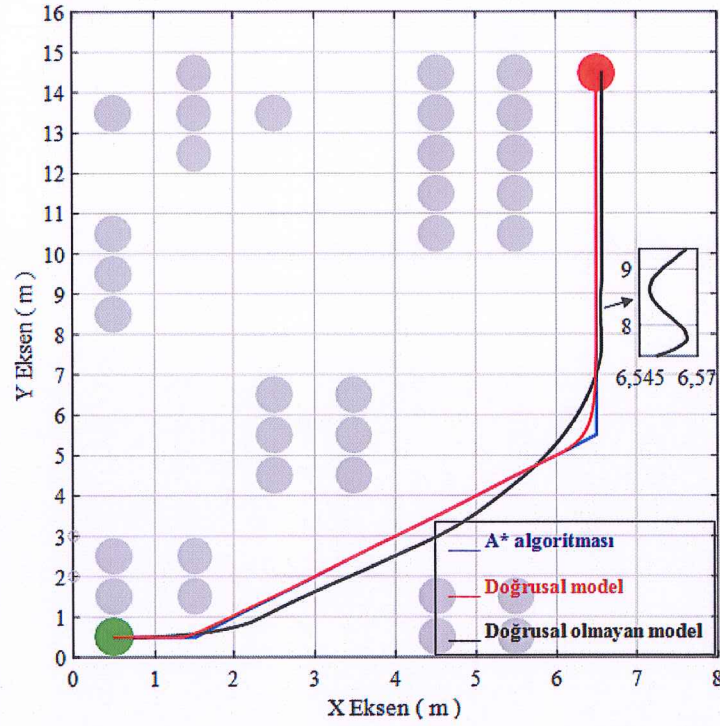
Uygulama no:	Çalışma alanı	Başlangıç düğümü	Bitiş düğümü	A* algoritması yol uzunluğu	Alt graf algoritması yol uzunluğu
1	8x6	(1,3)	(8,3)	8.656	9.3592
2	10x10	(2,1)	(10,10)	14.071	15.0645
3	43x36	(1,2)	(21,33)	78.012	74.4753

Sonuç olarak engellerin birbirine yakın olduğu küçük çalışma alanlarında Alt hedef graf algoritması A* algoritmasından daha uzun yol bulmaktadır fakat robot balığın dinamiği göz önüne alındığında daha güvenli bir yol bulmaktadır. Ayrıca engeller arasındaki mesafe arttığı zaman Alt graf algoritması ile bulunan yol hem A* algoritması ile bulunan yoldan daha kısa bir yol hem de çoğu dinamik sistemin matematiksel modeline uygun güvenli bir yoldur.

4.3. Yol Bulma Algoritmalarının Dinamik Model Uygulamaları

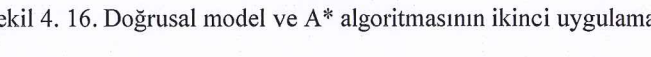
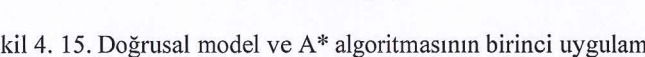
Bu bölümde MATLABR2015a programında robot balığın doğrusal olmayan ve doğrusal modelleri oluşturulmuş ve robot balığın hız ve yön kontrolü sağlanmıştır. Robot balığın kontrolü için oluşturulan hız ve yön kontrolünün MATLAB/Simulink modelleri sırasıyla EK-4 ve EK-5' de verilmiştir. Matematiksel modelinin oluşturulduğu robot balığa, A* algoritmasıyla bulunan en kısa yolun koordinat değerleri referans olarak verildiğinde robot balığın 2 boyutlu benzetim ortamında nasıl hareket ettiği incelenmiştir.

Dinamik model ile yapılan ilk uygulamada çalışma alanı olarak 15x7' lik bir alan seçilmiş ve başlangıç konumu (0,0), bitiş konumu (6,14) ve engeller Şekil 4.14' deki gibi kabul edilmiştir.

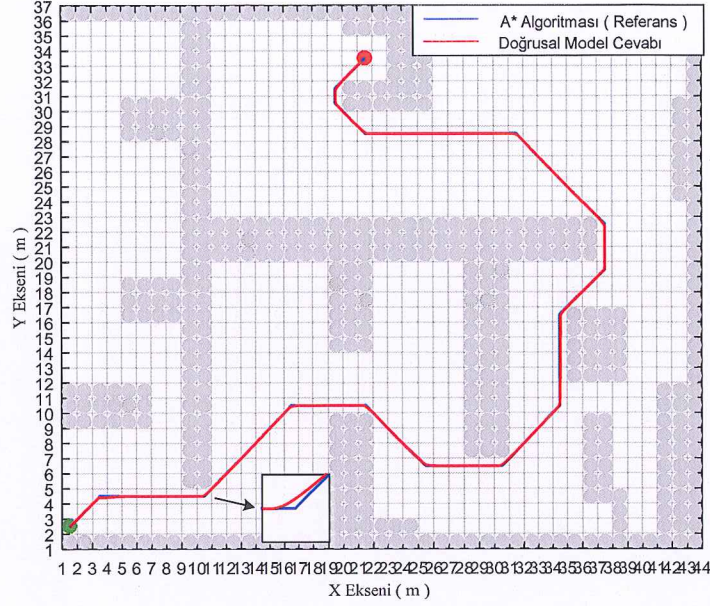


Şekil 4. 14. Robot balığın doğrusal olmayan ve doğrusal modellerinin algoritma sonucu

Şekil 4.14’ de görüldüğü gibi A* algoritması, başlangıç noktasından bitiş noktasına en kısa yoldan ulaşılması için hangi koordinatları takip etmesi gerektiğini (referansı) belirlemiştir. A* algoritmasının belirlemiş olduğu bu koordinatlar takip edilerek en kısa yoldan hedefe mavi renkli çizgi ile ulaşılacağı belirlenmiştir. Carangiform yüzüş moduna sahip bir robot balığın Bölüm 2.1’ de elde edilen doğrusal olmayan modeline bu koordinat değerleri verildiğinde robot balığın hareketi siyah renkli çizgi ile Şekil 4.14’ de gösterilmiştir. Daha sonra robot balığın doğrusal olmayan modeli çok karmaşık bir yapıya sahip olduğu ve çalışma yükünün fazla olması nedeniyle modeldeki bazı özellikler 0 kabul edilerek doğrusal modeli oluşturulmuştur. Oluşturulan robot balığın doğrusal modeline A* algoritmasının belirlemiş olduğu aynı koordinat değerleri verildiğinde kırmızı renkli çizgi ile hareket ettiği Şekil 4.14’ de gösterilmektedir. Robot balığın doğrusal olmayan ve doğrusal modeline A* algoritmasının belirlemiş olduğu yolun koordinat değerleri verildiğinde yaklaşık olarak aynı hareketi sağladığı belirlenmiştir. Bu yüzden daha sonraki yapılan uygulamalarda robot balığın daha az çalışma yüküne sahip olan doğrusal modeli kullanılmıştır.



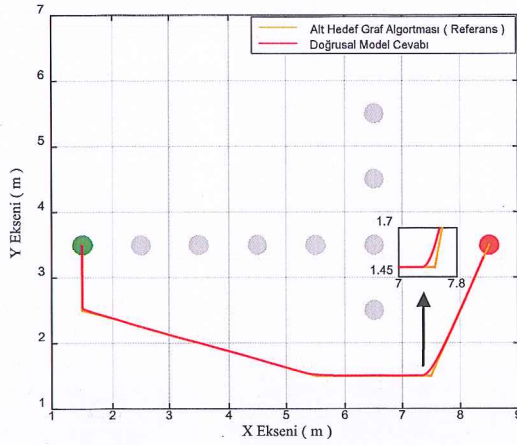
Şekil 4.17’ deki bir alanda A* algoritmasının bulduğu yolu ile robot balığın doğrusal modelinin takip ettiği kırmızı renkli çizgi ile gösterilmiştir.



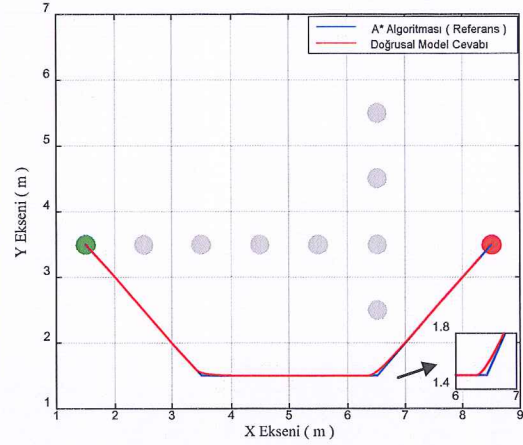
Şekil 4. 17. Doğrusal model ve A* algoritmasının üçüncü uygulaması

Yukarıdaki doğrusal model uygulamalarında robot balığın A* algoritması tarafından verilen konum değerlerine hareket ettiği kanıtlanmıştır. Robot balık için oluşturulmuş olan doğrusal olmayan ve doğrusal modeline A* algoritmanın koordinat değerleri verildiğinde çok az değişim göstererek hareket ettiği sonucuna varılmıştır.

Şekil 4.18’ deki uygulamada robot balığın doğrusal modeline Alt hedef graf ve A* algoritmaları ile gidilmesi gereken düğümlerin konum verileri verildiğinde kırmızı renkli yolu takip ettiği görülmüştür.



(a)

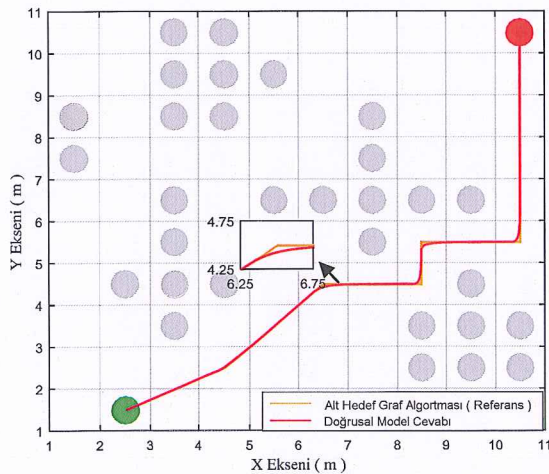


(b)

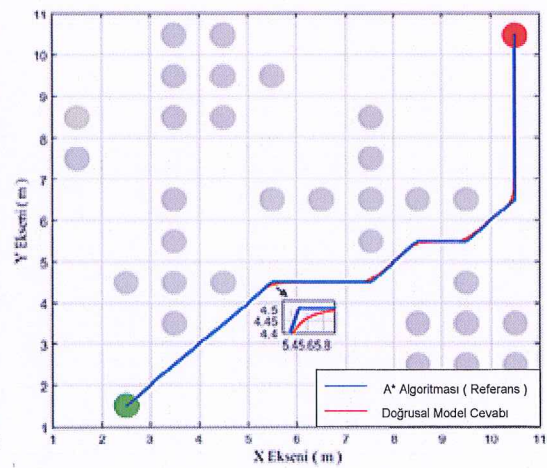
Şekil 4.18. Doğrusal model ile birinci uygulama a.)Alt hedef graf algoritması b.)A* algoritması

Şekil 4.18.a' da gösterilen başlangıç, bitiş ve engel noktalarına göre Alt hedef graf algoritmasıyla bulunan yol ve Alt graf algoritmasıyla bulunana düğüm değerlerine göre robot balığın doğrusal modelinin izledi yol gösterilmiştir. Şekil 4.18.b' de ise aynı çalışma ortamında A* algoritmasının izlediği yol ve doğrusal modelin izlediği yol gösterilmiştir.

Şekil 4.19' daki uygulamada robot balığın doğrusal modeline Alt hedef graf algoritması ile gidilmesi gereken düğümlerin konum verileri verildiğinde kırmızı renkli çizgiyi izleyerek ulaşılacağı belirlenmiştir.



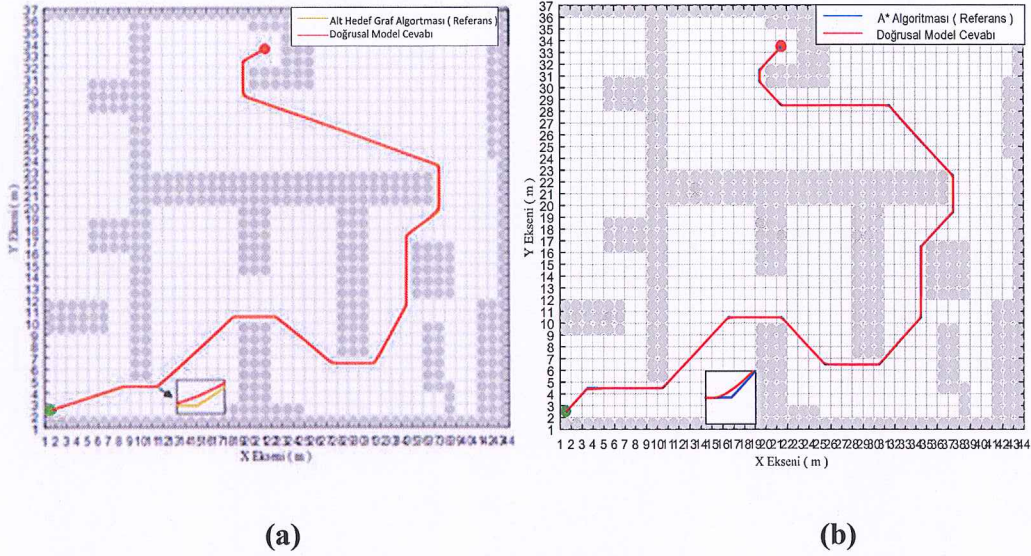
(a)



(b)

Şekil 4. 19. Doğrusal model ile ikinci uygulama a.)Alt hedef graf algoritması b.)A* algoritması

Şekil 4.19.a' da gösterilen başlangıç, bitiş ve engel noktalarına göre Alt hedef graf algoritmasıyla bulunan yol ve Alt graf algoritmasıyla bulunana düğüm değerlerine göre robot balığın doğrusal modelinin izledi yol kırmızı renkli çizgi ile gösterilmiştir. Şekil 4.19.b' de ise aynı çalışma ortamında A* algoritmasının izlediği yol ve doğrusal modelin izlediği yol gösterilmiştir.



Şekil 4. 20. Doğrusal model ile üçüncü uygulama a.)Alt hedef graf algoritması b.)A* algoritması

Şekil 4.20.a' da gösterilen büyük çalışma alanı için Alt hedef graf algoritmasıyla bulunan yol ve Alt graf algoritmasıyla bulunana düğüm değerlerine göre robot balığın doğrusal modelinin izledi yol kırmızı renkli çizgi ile gösterilmiştir. Şekil 4.20.b' de ise aynı çalışma ortamında A* algoritmasının izlediği yol ve doğrusal modelin izlediği yol gösterilmiştir.

5. SONUÇLAR ve TARTIŞMA

Bu çalışmada bir başlangıç noktasından bitiş noktasına en kısa yolu bulmak için yol bulma algoritmalarından olan A* ve Alt hedef graf algoritmaları MATLAB2015a programında yazılmıştır. Yazılan A* algoritması başlangıç noktasının komşu düğümlerine bakar ve minimum F değerine sahip olan düğüm açık listeye eklenir ve bu açık listedeki düğümlerin komşu düğümüne tekrar bakılarak minimum değerli olan düğüm seçilerek kapalı listeye eklenir. Kapalı listedeki düğümler takip edilerek en kısa yol bulunmuş olur. Fakat A* algoritmasıyla bulunan yol, robot balığın sistemi dikkate alınmadan bulunan bir yoldur. Bundan dolayı güvenli yol bulma işlemi için Alt hedef graf algoritması kullanılmıştır. Alt hedef graf algoritması robot balığın sistem dinamiğinden dolayı hareket esnasında engellere çarpmadan bitiş noktasına ulaşabilmesi için engellerin köşe noktalarının 1 birim uzağındaki düğümlere hareket ederek robot balığın daha güvenli bir şekilde ulaşımını sağlamış olur.

6 farklı çalışma ortamı için A* algoritması ile yol bulma işlemi gerçekleştirilmiştir. Yapılan uygulamalar sonucunda A* algoritması en kısa yolu bulmuştur. Ayrıca 3 farklı ortam için hem Alt hedef graf hem de A* algoritması ile yollar bulunmuştur. Bulunan bu yollar için en kısa yolu A* algoritması bulmuştur. Fakat sistemin güvenliği söz konusu olup yolun uzunluğunun önemli olmadığı durumlarda tercih olarak Alt hedef graf algoritması kullanılabilir.

Robot balığın doğrusal olmayan ve doğrusal modelleri MATLAB2015a programında oluşturulmuş ve robot balığın hız ve yön kontrolü sağlanmıştır. Doğrusal matematiksel modelinin oluşturulduğu robot balığa, A* ve Alt hedef graf algoritmalarıyla bulunan en kısa yolun konum değerleri verilerek robot balığın 2 boyutlu benzetim ortamında nasıl hareket ettiği incelenmiştir ve doğrusal model uygulamalarında robot balığın A* ve Alt hedef graf algoritmaları tarafından verilen konum değerlerine çok az değişimle hareket ettiği kanıtlanmıştır.

6. ÖNERİLER

Bu çalışma kapsamında 2 boyutlu ortamlar için yazımı gerçekleştirilen A* ve Alt hedef graf algoritmaları MATLAB benzetim ortamında oluşturulmuş ve robot balığın dinamik modeli ile yol bulma işlemi gerçekleştirilmiştir. Bu çalışmalara ek olarak;

- 2 boyutlu ortamlar için gerçekleştirilen A* algoritması 3 boyut için 3 boyutlu çalışma ortamlarında uygulanabilir.
- Robot balığa A* ve Alt hedef graf algoritmaları benzetim ortamında uygulanmıştır. İleriki çalışmalarda bu algoritmalar TÜBİTAK 114E652 nolu proje kapsamında oluşturulan bir robot balık prototipine havuz ortamında 2 yada 3 boyutlu olarak en kısa yolun bulunması deneylerinde uygulanacaktır. Deneysel ortamda elde edilecek veriler ışığında A* ve Alt hedef graf algoritmalarında iyileştirme çalışmaları yapılacaktır.

KAYNAKLAR

- [1] Barutçuoğlu, E.I. 2001. Robotların Tarihçesi, July 2.
- [2] Şabanović, A. and Yannier, S. 2003. Robotlar: Sosyal Etkileşimli Makineler , *İstisnalar bilimi ve tekniği dergisi*.
- [3] Taharwa, İ.A., Sheta, A., and Weshah M.A. 2008. A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment, *Journal of Computer Science*, 4, 341–344.
- [4] Serrano A.R. and Boumedine M., 1996. Real-time navigation in unknown environments using fuzzy logic and ultrasonic sensing, *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*.
- [5] Zhang, Z. and Zhao, Z., 2014. A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm, *International Journal of Smart Home*, 8, 75–86.
- [6] Das, P.K., Konar, A. and Laishram, R., 2010. Path Planning of Mobile Robot in Unknown Environment, *Special Issue of IJCCT International Conference* .
- [7] Yuan, Q.Q., Debao, S., Ning, L., and Yigang, C., 2004. Path planning for mobile robot using the particle swarm optimization with mutation operator, *Machine Learning and Cybernetics* 4, 2473–2478.
- [8] Kang, H., Lee, B. and Kim, K 2008. Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm, *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*.
- [9] Duchon, F., Babinec, A., Kajan, M., Beno, P., Florek, M., Fico, T., and Jurisica, L., 2014. Path planning with modified A star algorithm for a mobile robot, *Procedia Engineering* 96, 59–69.
- [10] Zuo, L., Gua, Q., Xu, X. and Fu, H. 2015. A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots, *Neurocomputing*, 170, 257–266.

- [11] **Okutkan, O.**, 2006. Yapay zeka ile robot kontrolü, *Yüksek Lisans Tezi.*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.
- [12] **Soltani, A.R., Tawfik, H., Goulernas, J.Y., and Fernando, T.**, 2002. Path planning in construction sites: Performance evaluation of the Dijkstra, A*, and GA search algorithms, *Advanced Engineering Informatics*, **16**, 291–303.
- [13] **Vacariu, L., Roman, F., Timar, M., Stanciu, T., Banabic, R., and Cret, O.**, 2007. Mobile robot path-planning implementation in software and hardware, *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, Greece, February ,2007,16-19
- [14] **Daily, R. and Bevely, D.M.**, 2008. Harmonic potential field path planning for high speed vehicles, *Proceedings of the American Control Conference*. Washington, USA. 4609–4614.
- [15] **Zhuang, Y., Sharma. S., Subudhi, B., Huang, H. and Wan, J.**, 2016. Efficient collision-free path planning for autonomous underwater vehicles in dynamic environments with a hybrid optimization algorithm, *Ocean Engineering*, **127**, 190-199.
- [16] **Sun, B., and Zhu, D.**, 2016. Three dimensional D*Lite path planning for Autonomous Underwater Vehicle under partly unknown environment, *12 th World Congress on Intelligent Control and Automation*, 10.1109/WCICA.2016.7578444.
- [17] **Hu, Y., Zhao, W., Wang, L., and Jia, Y.**, 2009. Underwater target following with a vision-based autonomous robotic fish, *American Control Conference*, June 10-12, 5265-5270.
- [18] **Hu, Y., Zhao, W., Wang, L., and Jia, Y.**, 2009. Vision-Based Target Tracking and Collision Avoidance for Two Autonomous Robotic Fish, *IEEE Transactions on Industrial Electronics*, **56**, 1401-1410.
- [19] **Liu, J., and Hu, H.** 2010. Biological Inspiration: From Carangiform Fish to Multi-Joint Robotic Fish, *Journal of Bionic Engineering*, **7**, 35 -48.

- [20] Oiang, G., 2012. Path Planning of Bio-mimetic Robot Fish Based on Improved a Star Algorithm and B-Splines Function Çin, *Master's thesis.*, Tianjin University Control Theory and Control Engineering.
- [21] Zhou, C., Inaba, M. and Tasaka, A., 2008. A Miniature Biomimetic Robotic Fish and Its Realtime Path Planning, *Field and Service Robotics*, **42**, 147–156.
- [22] Shao, J., Xie, G., Wang, L. and Zhang, W., 2005. Obstacle Avoidance and Path Planning Based on Flow Field for Biomimetic Robotic Fish, *AI 2005:Advances in Artificial Intelligence.*, **3809**, 857-860.
- [23] Wang, H. and Jiang, Y., 2013. Robotic Fish Path Planning Based on an Improved A* Algorithm, *Applied Mechanics and Materials*, **336-338**, 968-972.
- [24] Özbilgin, H., Kınacıgil, H.T. ve Özbilgin, Y.D., 2004, Balıklarda Yüzme Davranışı ve Trol Operasyonu Açısından Önem, *E.Ü. Su Ürünleri Dergisi*, **21**, 355-359.
- [25] Zhang, H. and Chen, L.L., 2009, The research on the nonlinear control of carangiform biomimetic robotic fish, *2009 International Conference on Mechatronics and Automation*, IEEE, Aug. 9-12, 3207-3211.
- [26] Şeker, Ş. S., Arama Algoritmaları (Search Algorithms) <http://bilgisayarkavramlari.sadievrenseker.com/2009/11/23/arama-algoritmaları-search-algorithms/> 18 Kasım 2016.
- [27] Reddy. H., 2013. PATH FINDING - Dijkstra's and A* Algorithm's, <http://cs.indstate.edu/hgopireddy/algor.pdf> , 10 Kasım 2016.
- [28] Blake. R. W., 1976. Fish locomotion, Cambridge university press, New york.
- [29] Yu, J., Tan, M., Wang, S. and Chen, E., 2004, Development of a Biomimetic Robotic Fish and Its Control Algorithm, *IEEE Transactions on Systems, Man, and Cybernetics.*, **34**, 1798-1810.
- [30] Korkmaz D., 2011. Uzaktan Kontrollü Bir Robot Balığın Tasarımı ve Gerçeklemesi, *Yüksek Lisans Tezi*, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ.

- [31] **Sturtevant, N.**, Video Game AI: Lecture 5 Planning for Pathfinding, <http://web.cs.du.edu/~sturtevant/ai-s11/Lecture05.pdf>, 15 Kasım 2016.
- [32] **Fossen, T.I.**, 1994. Guidance and control of ocean vehicles, *Trondheim University*, Norway.
- [33] **Fossen, T.I.**, 2011. Handbook of Marine Craft Hydrodynamics and Motion Control, *John Wiley & Sons*, New York.
- [34] <https://qiao.github.io/PathFinding.js/visual/> Github Programı. 16 Kasım 2016.
- [35] **Chen, P. J., and Hwang, Y. K.**, 1998, SANDROS: A Dynamic Graph Search Algorithm for Motion Planning, *IEEE Transactions On Robotics And Automation*, **14**, 390 - 403.
- [36] **Suebsaiprom, P., and Lin, C. L.**, 2015, Maneuverability modeling and trajectory tracking for fish robot, *Control Engineering Practice*, **45**, 22-36.
- [37] **Uras, T., Koenig, S., and Hernandez, C.**, 2013, Subgoal graphs for optimal pathfinding in eight-neighbor grids, *In Proceedings of the 23rd International Conference on Automated Planning and Scheduling*.
- [38] **Uras, T. and Koenig, S.**, 2014, Identifying Hierarchies for Fast Optimal Search, *Seventh Annual Symposium on Combinatorial Search*.

EKLER

EK-1 A-Star Algoritmasının MATLAB Kodu

```
% Haritanın alanını oluşturur ve alan 2 olarak alınır.
% Başlangıç noktası ,Engeller ve bitiş noktası belirlenir.
% Bu noktalar grafik üzerine yerleştirilir.
% Engel=-1,Bitiş =0, Robot=1, Alan=2
clear
clc
str=8;
stn=6;
MAP=2*(ones(str,stn));
x_val = 1;
y_val = 1;
axis([1 str+1 1 stn+1])
grid on;
hold on;
n=0;
h=msgbox('Lütfen bitiş noktasını farenin solu seçiniz')
uiwait(h,5);
if ishandle(h) == 1
    delete(h);
end
but=0;
while (but ~= 1)
    [xval,yval,but]=ginput(1);
end
xval=floor(xval);
yval=floor(yval);
xTarget=xval;
yTarget=yval;
MAP(xval,yval)=0;
plot(xval+.5,yval+.5,'ro');
h=msgbox('Engeli seçmek için farenin sol tuşu ile seçiniz ve son engelden sonra farenin sağ tuşu ile bitiriniz.');
```

```
uiwait(h,10);
if ishandle(h) == 1
    delete(h);
end
while but == 1
    [xval,yval,but] = ginput(1);
    xval=floor(xval);
    yval=floor(yval);
    MAP(xval,yval)=-1;
    plot(xval+.5,yval+.5,'ko');
end
h=msgbox('Lütfen başlangıç noktasını farenin solu seçiniz');
uiwait(h,5);
if ishandle(h) == 1
    delete(h);
end
but=0;
while (but ~= 1)
    [xval,yval,but]=ginput(1);
    xval=floor(xval);
    yval=floor(yval);
end
xStart=xval;
yStart=yval;
```

```

MAP(xval,yval)=1;
plot(xval+.5,yval+.5,'go');
%Grafik üzerinde başlangıç bitiş ve engellerin alımının bitimi.
tic
OPEN=[];
CLOSED=[];
%Kapalı liste içerisine engelleri yerleştirmek.
k=1;
for i=1:str
    for j=1:stn
        if(MAP(i,j) == -1)
            CLOSED(k,1)=i;
            CLOSED(k,2)=j;
            k=k+1;
        end
    end
end
CLOSED_say=size(CLOSED,1);
xNode=xval;
yNode=yval;
OPEN_say=1;
yol_mal=0;
goal_distance=distance(xNode,yNode,xTarget,yTarget);
h_distance=distancel(xNode,yNode,xTarget,yTarget);
OPEN(OPEN_say,:)=open_ekleme(xNode,yNode,xNode,yNode,yol_mal,h_distance,g
oal_distance);
OPEN(OPEN_say,1)=0;
CLOSED_say=CLOSED_say+1;
CLOSED(CLOSED_say,1)=xNode;
CLOSED(CLOSED_say,2)=yNode;
NoPath=1;
while((xNode ~= xTarget || yNode ~= yTarget) && NoPath == 1
    dizigens=dizi_gens(xNode,yNode,yol_mal,xTarget,yTarget,CLOSED,str,stn);
    dizi_say=size(dizigens,1);
    for i=1:dizi_say
        flag=0;
        for j=1:OPEN_say
            if(dizigens(i,1) == OPEN(j,2) && dizigens(i,2) == OPEN(j,3) )
                OPEN(j,8)=min(OPEN(j,8),dizigens(i,5));
                if OPEN(j,8)== dizigens(i,5)
                    OPEN(j,4)=xNode;
                    OPEN(j,5)=yNode;
                    OPEN(j,6)=dizigens(i,3);
                    OPEN(j,7)=dizigens(i,4);
                end;
                flag=1;
            end;
        end;

        end;
        if flag == 0
            OPEN_say = OPEN_say+1;

OPEN(OPEN_say,:)=open_ekleme(dizigens(i,1),dizigens(i,2),xNode,yNode,dizi
gens(i,3),dizigens(i,4),dizigens(i,5));
        end;
    end;
    min_node = min_fn(OPEN,OPEN_say,xTarget,yTarget);
    if (min_node ~= -1)
        xNode=OPEN(min_node,2);
        yNode=OPEN(min_node,3);

```



```

        yol_mal=OPEN(min_node,6);
        CLOSED_say=CLOSED_say+1;
        CLOSED(CLOSED_say,1)=xNode;
        CLOSED(CLOSED_say,2)=yNode;
        OPEN(min_node,1)=0;
    else
    end;
end;
i=size(CLOSED,1);
Optimal_yol=[];
xval=CLOSED(i,1);
yval=CLOSED(i,2);
i=1;
Optimal_yol(i,1)=xval;
Optimal_yol(i,2)=yval;
i=i+1;

if ( (xval == xTarget) && (yval == yTarget))
    inode=0;
    parent_x=OPEN(node_index(OPEN,xval,yval),4);
    parent_y=OPEN(node_index(OPEN,xval,yval),5);

    while( parent_x ~= xStart || parent_y ~= yStart)
        Optimal_yol(i,1) = parent_x;
        Optimal_yol(i,2) = parent_y;
        inode=node_index(OPEN,parent_x,parent_y);
        parent_x=OPEN(inode,4);
        parent_y=OPEN(inode,5);
        i=i+1;
    end;
    j=size(Optimal_yol,1);
    Optimal_yol(j+1,1)=xStart
    Optimal_yol(j+1,2)=yStart
    j=j+1
    p=plot(Optimal_yol(j,1)+.5,Optimal_yol(j,2)+.5,'ro');
    j=j-1;
    plot(Optimal_yol(:,1)+.5,Optimal_yol(:,2)+.5,'b');
else
h=msgbox('Hiçbir yol yok','warn');
uiwait(h,5);
end
toc

```

EK-2 Alt Hedef Graf Algoritmasının MATLAB Kodu

```
% Haritanın alanını oluşturur ve alan 2 olarak alınır.
% Başlangıç noktası ,Engeller ve bitiş noktası belirlenir.
% Bu noktalar grafik üzerine yerleştirilir.
% Engel=-1,Bitiş =0, Robot=1, Alan=2
clear
clc
str=8;
stn=6;
MAP=2*(ones(str,stn));
x_val = 1;
y_val = 1;
axis([1 str+1 1 stn+1])
grid on;
hold on;
n=0;
h=msgbox('Lütfen bitiş noktasını farenin solu seçiniz')
uiwait(h,5);
if ishandle(h) == 1
    delete(h);
end
but=0;
while (but ~= 1)
    [xval,yval,but]=ginput(1);
end
xval=floor(xval);
yval=floor(yval)
xTarget=xval;
yTarget=yval;
MAP(xval,yval)=0;
plot(xval+.5,yval+.5,'ro');
h=msgbox('Engeli seçmek için farenin sol tuşu ile seçiniz ve son engelden sonra farenin sağ tuşu ile bitiriniz.');
```

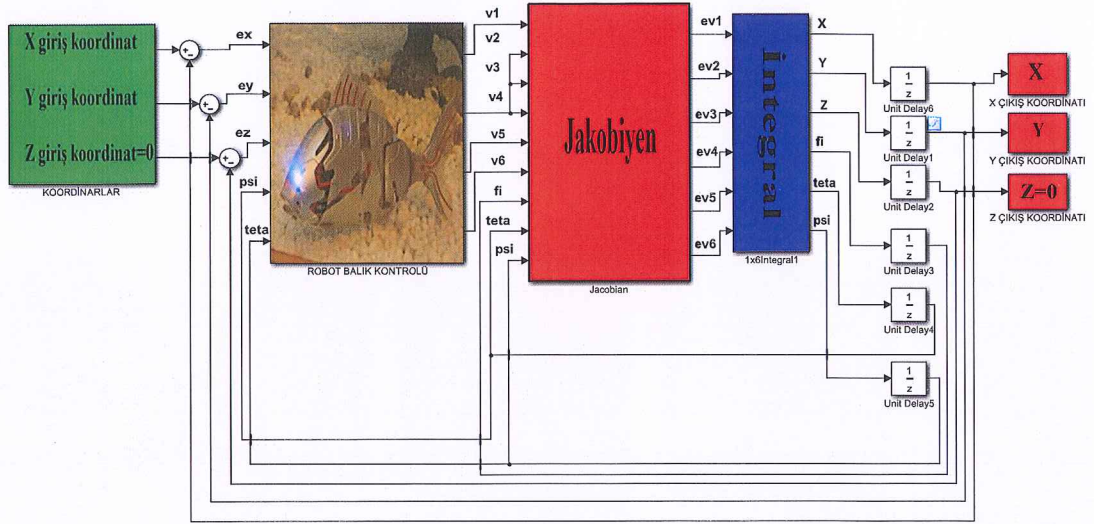
```
uiwait(h,10);
if ishandle(h) == 1
    delete(h);
end
while but == 1
    [xval,yval,but] = ginput(1);
    xval=floor(xval);
    yval=floor(yval);
    MAP(xval,yval)=-1;
    plot(xval+.5,yval+.5,'ko');
end
h=msgbox('Lütfen başlangıç noktasını farenin solu seçiniz');
uiwait(h,5);
if ishandle(h) == 1
    delete(h);
end
but=0;
while (but ~= 1)
    [xval,yval,but]=ginput(1);
    xval=floor(xval);
    yval=floor(yval);
end
xStart=xval;
yStart=yval;
MAP(xval,yval)=1;
```

```

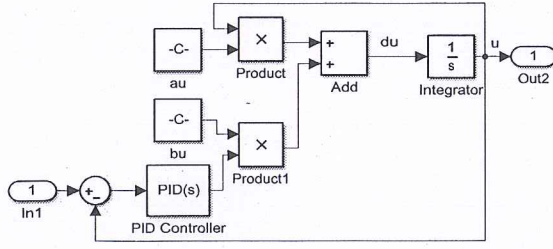
plot(xval+.5,yval+.5,'go');
%Grafik üzerinde başlangıç bitiş ve engellerin alımının bitimi.
tic
OPEN=[];
CLOSED=[];
%Kapalı liste içerisine engelleri yerleştirmek.
k=1;
for i=1:str
    for j=1:stn
        if(MAP(i,j) == -1)
            CLOSED(k,1)=i;
            CLOSED(k,2)=j;
            k=k+1;
        end
    end
end
[C_x,C_y]=size(CLOSED)
yatay = yatay_fn(C_x,CLOSED);
dusey = dusey_fn(C_x,CLOSED);
[e_x,e_y]=size(yatay)
[f_x,f_y]=size(dusey)
u=1;
devamyatay = devam_yatay_fn(e_x,yatay);
devamdusey = devam_dusey_fn(C_x,dusey);
e2=sadel( yatay,devamyatay)
f2=sadel(dusey,devamdusey)
[f2_x,f2_y]=size(f2)
[e2_x,e2_y]=size(e2)
Geriyat=sadel(CLOSED,f)
Geridik=sadel(Geriyat,e)
[Gd_x,Gd_y]=size(Geridik)
son=son_fn(Gd_x,C_x,Geridik,CLOSED) %engellerin son kısımdaki köşe
nktaları bulunur.
yataybas=yatay_bas_fn(e2_x,C_x,e2,CLOSED) %Yatay engellerin baş
kısımdaki köşe nktaları bulunur.
duseybas=dusey_bas_fn(f2_x,C_x,f2,CLOSED) %Düşey engellerin baş
kısımdaki köşe nktaları bulunur.
subgoldist = subgoldis(x1,y1,x2,y2) % maliyet hesaplanır.
yol=min_fon(subgoaldist) % uygun yol bulunur.
plot(yol(:,1)+.5, yol(:,2)+.5,'b'); % uygun yol çizimi gerçekleşir.

```

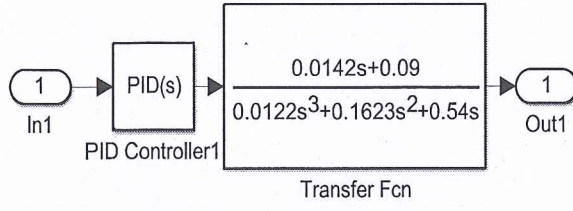

EK-3 Robot Balığın Doğrusal sisteminin MATLAB/Simulink Modeli



EK-4 Robot Balığın Hız Kontrolü için Oluşturulan MATLAB/Simulink Modeli



EK-5 Robot Balığın Yön Kontrolü için Oluşturulan MATLAB/Simulink Modeli



ÖZGEÇMİŞ

Seda YETKİN

sedayetkin23@gmail.com

Fırat Üniversitesi Teknoloji Fakültesi

Mekatronik Mühendisliği Bölümü

ELAZIĞ

03 Mayıs 1992 yılında Elazığ'da doğmuştur. Ortaöğrenimini Mehmet Akif Ersoy Lisesinde tamamlamıştır. Üniversiteye hayatına 2010 yılında Fırat Üniversitesi Mühendislik Fakültesi Mekatronik Mühendisliği bölümünde başlamış ve 2012 yılında Fırat Üniversitesi Mühendislik Fakültesi Makine Mühendisliği bölümünde Çift Anadal yapmaya başlamıştır. 2014 yılında Mekatronik Mühendisliği bölümünü, 2015 yılında Makine Mühendisliği bölümünü bitirmiştir. Eylül 2014'de Fırat Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliği Anabilim Dalında ve Şubat 2015'de Fırat Üniversitesi Mühendislik Fakültesi Makine Mühendisliği Anabilim Dalında yüksek lisansa başlamıştır ve yüksek lisansları devam etmektedir. Mayıs 2015'ten beri 114E652 nolu "Üç Boyutlu Hareket Kabiliyetine Sahip Carangiform Türünde Bir Biyomimetik Robot Balığın Tasarımı ve Gerçeklemesi" konulu 1001 projesinde bursiyer olarak görev almaktadır.