KOCAELİ ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

LOCAL PATH PLAN

ROBOTİK SİSTEMLERE GİRİŞ

Rapor

Şüheda ARAS – Kübra GÜZEL

170207050 - 170207026

Bölümü: Elektronik ve Haberleşme Mühendisliği

Danışman: Doç. Dr. Mustafa ÇAKIR

KOCAELİ, 2020

İÇİNDEKİLER

1.GİRİŞ 5
2.LOCAL PATH PLANNING (Lokal Yol Planma) 6
2.1 Bug Algoritmaları 6
2.1.1 Bug0 algoritması 7
2.1.2 Bug1 algoritması 8
2.1.3 Bug2 algoritması 9
2.1.4 Alg1 algoritması 11
2.1.5 Alg2 algoritması 12
2.1.6 Diğer Bug Algoritmaları 14
3.SİMÜLASYON SONUÇLARI 15
4.SONUÇ 18
KAYNAKÇA 19
EKLER 20
Ek-1 Global Path Planning 20
Ek-2 Local Path Planning 20

ŞEKİLLER DİZİNİ

Şekil 1: Sürücüsüz araçların genel çalışma döngüsü 5
Şekil 2. 1: A ortamı 6
Şekil 2. 2: B ortamı 6
Şekil 2. 3: İki farklı 2B ortam için Bug-0 algoritmasının üreteceği güzergâhlar 7
Şekil 2. 4: A ortamında Bug1 algoritması 9
Şekil 2. 5: B ortamında Bug1 algoritması 9
Şekil 2. 6: A ortamında Bug2 algoritması 10
Şekil 2.7: B ortamında Bug2 algoritması 10
Şekil 2.8: A ortamında Alg1 algoritması 12
Şekil 2.9: B ortamında Alg1 algoritması 12
Şekil 2.10: A ortamında Alg2 algoritması 13
Şekil 2.11: B ortamında Alg2 algoritması 14
Şekil 3. 1: Simülasyon sonuçları 15
Şekil 3. 2:Algoritmanın izlediği yol 17

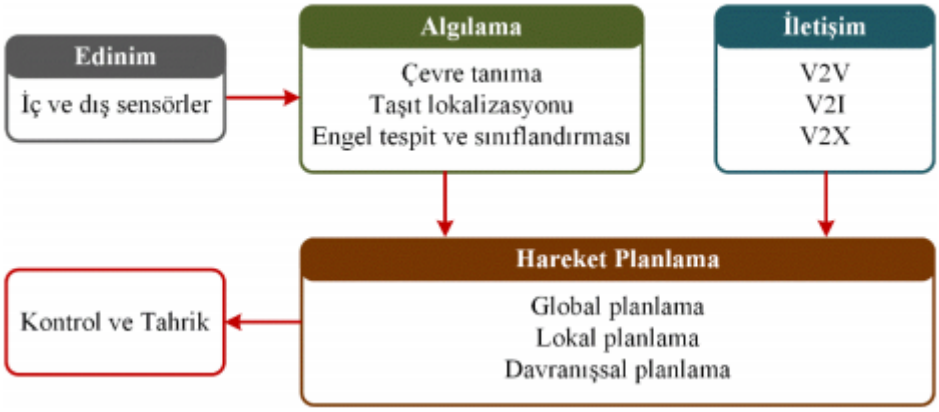
ÖZET

Gerçekleştirdiğimiz projede lokal planlama algoritmalarından birini kullanarak engellere çarpmadan hedef noktaya ulaşabilmek, hedeflenmektedir. Bug 1 algoritması kullanılmıştır. Algoritmanın genel çalışma mantığı:

Hedefin etrafından tamamen dolaşıp hedefe en yakın noktayı tespit etmektir. Bu yüzden Bug 0 algoritmasından daha doğru sonuç verir. Proje Matlab üzerinde gerçekleştirilmiş olup sonuçlar simülasyonda gözlemlemiştir.

1.GİRİŞ

Hareket planlaması aşaması literatürde global, lokal ve davranışsal olmak üzere üç sınıfta incelenmektedir. Global yörünge planlamada çalışma uzayı tamamen bilinmeli ve statik olmalıdır, yani çevresel bilgiler önceden sağlanmalıdır. Global planlama aracın izleyeceği yolu rota ve görev planlaması açısından inceler, başlangıç ve bitiş noktaları arasındaki en kısa yolu bulmaya odaklanır. Lokal planlama dönemeçler ve engellerden kaçınma gibi daha küçük çaplı görevleri üstlenir. Son yıllarda bilgi işleme hızının da gelişmesiyle birlikte üzerine daha çok çalışma yapılabilen davranışsal planlama ise anlık oluşan durumlarda araca manevra kabiliyeti gibi yetenekler kazandırmayı hedeflemektedir. Çalışma döngüsündeki son aşama olan kontrol ve tahrik aşamasında ise aracın planlanan şekilde hareket etmesi için gereken şekilde tahrik edilmesi sağlanır. Bütün bu çalışma döngüsü Şekil 1'de görselleştirilmiştir.

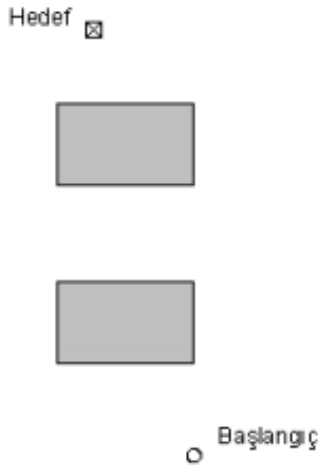


Şekil 1: Sürücüsüz araçların genel çalışma döngüsü [3]

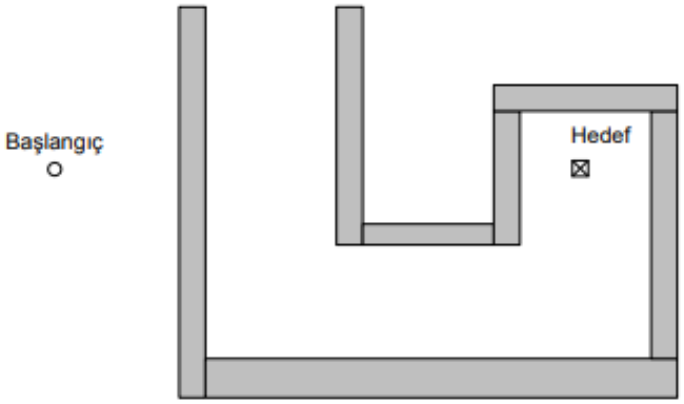
Bu planlama çeşitlerinin yaptıkları görevleri gündelik sürücü görevleriyle eşleştirmek gerekirse, global planlama başlangıç ve bitiş noktaları arasındaki yolu genel hatlarıyla belirleme görevini, lokal planlama kavşaklarda ve dönemeçlerde aracın şeridi gibi mikro görevlerini, davranışsal planlama ise beklenmedik ani araç çıkışı gibi anlık görevleri gerçekleştirmektedir.

2.LOCAL PATH PLANNING (Lokal Yol Planma)

Dar limitli yol planlamak, çarpışmadan kaçınmak için lokal planlamayı tercih ederiz. Lokal planlama da en yaygın Bug algoritmaları kullanılır. Başlangıçta harita bilgisine ihtiyaç yoktur, sensörleriyle anlık ortam haritasına göre yol planlaması yapar.



Şekil 2.1: A ortamı



Şekil 2.2: B ortamı

2.1 Bug Algoritmaları

Lokal yol planlama algoritmalarından olan bug(böcek) algortimaları dokunma sensörüne sahip noktasal robotların yol planlama problemlerinin çözümü amacıyla böceklerden ilham alınarak geliştirilmişlerdir. Bug algoritmaları robotun bulunduğu ortamın tüm bilgisine ihtiyaç duymazlar. Hedef noktanın bulundukları konuma göre konumu bilgisi ve yerel ortam bilgisi ile hedef noktaya ulaşabilirler. Bug algoritması ile çalışan bir robotun iki temel durum arasında geçiş yaparak hareket eder. Bu iki durum; Herhangi bir engel önlerine çıkmadığı sürece bir doğru üzerinde hedef noktaya doğru ilerlemek veya önlerine çıkan engeli, engelden ayrılma durumu oluşana kadar takip etmek olarak tanımlanabilir.

2.1.1 Bug0 algoritması

Bug-0 algoritması en ilkel planlayıcı olup herhangi bir hafıza ihtiyacı bulunmamaktadır. Gezgin robot (MR), başlangıç noktası $SMR = (xS\ ,\ yS\ )$'yi ve varılmak istenen hedef noktayı $GMR\ =\ (xG,\ yG\ )$'yi bilmektedir. Noktalara ait koordinatlar 2B Kartezyen Koordinat Sistemi'nde verilmektedir. Bug-0 algoritmasında:

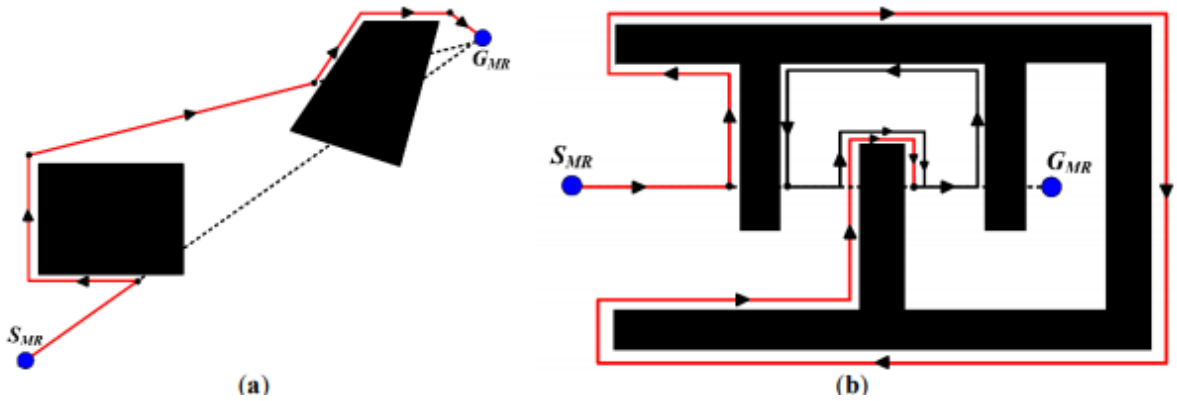
1) Aşağıdaki koşullar sağlanmadıkça, direkt olarak hedef noktası GMR'ye doğru ilerle.

- Hedefe ulaşıldı. Prosedürü durdur.
- Engel ile karşılaşıldı. Adım 2'ye geç.

2) Aşağıdaki koşullar sağlanmadıkça, kabul edilen yönde (varsayılan) engelin sınırlarını takip et .

- Hedefe ulaşıldı. Prosedürü durdur.
- Gezgin robotun baş açısı hedefe doğru bakıyor ve önü açık. Adım 1'e geç.

Bug-0 algoritmasına yönelik gezgin robotun davranışı Şekil 2.3'de örneklenmiştir. Şekil içerisindeki kırmızı yol gezgin robotun engelin sınırlarını takip ettiği yolu göstermektedir. Siyah renkli yol ise Bug0 algoritmasının yerel bir çevrimde sonsuz döngüye girdiği yolu göstermektedir. Yerel çevrim, iç bükey bir nesnenin içerisinde gezgin robotu bir döngüye sokacak bir yapı olarak düşünülebilir. Şekil 1.(b)'de görüldüğü üzere Bug-0 algoritması, Pledge Algoritmasında olduğu gibi yerel bir çevrimden dolayı kolayca sonsuz bir döngüye girebilmektedir.[1]



Şekil 2.3: İki farklı 2B ortam için Bug-0 algoritmasının üreteceği güzergâhlar

2.1.2 Bug1 algoritması

Bug1 algoritması ilk defa Lumelsky ve Stepanov tarafından önerilmiştir. Bug 1 algoritması adımları aşağıdaki gibi tanımlanabilir. Şekil 2.4'te ve Şekil 2.5'te anlatılan Bug1 algoritmasının adımları aşağıda açıklanmıştır.

1) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar hedefe doğru bir doğru üzerinde ilerler.

- Robot hedef ulaşırsa durur.
- Robot ilerlediği doğru üzerinde bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 2. Adıma geçilir.

2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engel takip eder. Bu adım devam ederken robotun hedef noktaya en yakın olduğu nokta L olarak adlandırılır.

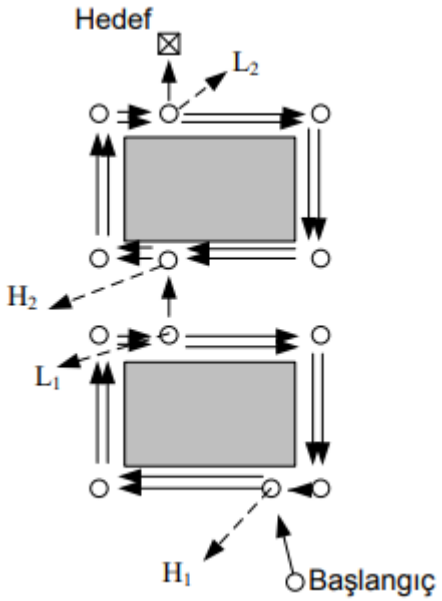
- Robot hedef ulaşırsa durur.
- Robot H noktasına ulaşırsa 3. Adıma geçilir.

3) Öncelikle hedefin ulaşılabilir olup olmadığı kontrol edilir. Bunun için H noktası ile L noktasının aynı olup olmadığı kontrol edilir.

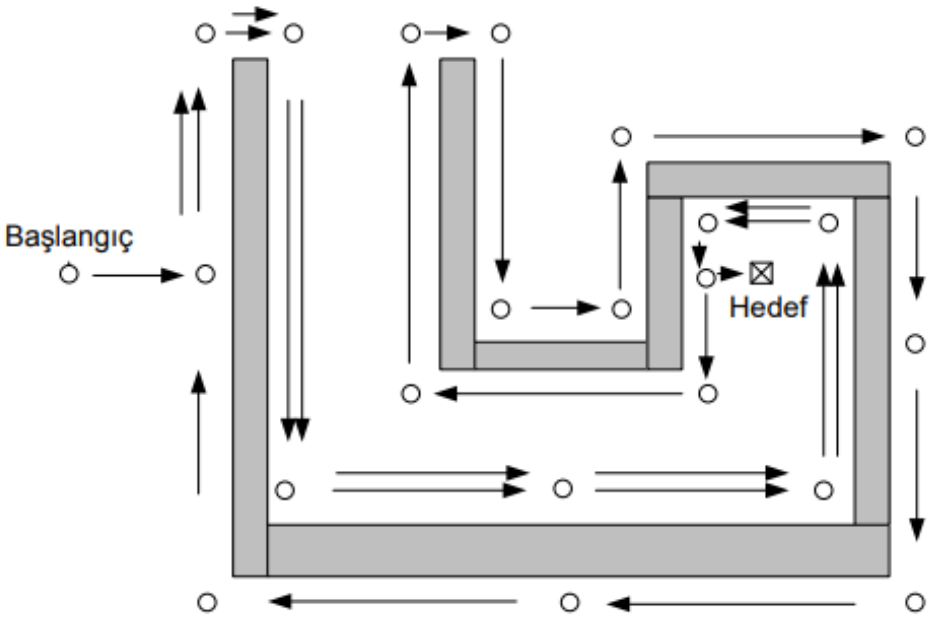
- Eğer H ve L noktası aynı ise hedef ulaşılabilir değildir. Robot durur.
- Eğer H ve L noktası aynı değilse robot yine engeli takip ederek L noktasına gider. L noktasına ulaştığında 1. Adıma geçilir.

3) Öncelikle hedefin ulaşılabilir olup olmadığı kontrol edilir. Bunun için H noktası ile L noktasının aynı olup olmadığı kontrol edilir.

- Eğer H ve L noktası aynı ise hedef ulaşılabilir değildir. Robot durur.
- Eğer H ve L noktası aynı değilse robot yine engeli takip ederek L noktasına gider. L noktasına ulaştığında 1. Adıma geçilir.



Şekil 2.4: A ortamında Bug1 algoritması



Şekil 2.5 : B ortamında Bug1 algoritması

2.1.3 Bug2 algoritması

Bug2 algoritması Lumelsky ve Stepanov tarafından önerilmiştir. Bug1 algoritması ile Bug2 algoritması arasındaki en büyük fark engel takibi durumundan ayrılma koşulu olarak getirilen M doğrusu kavramıdır. adımları aşağıdaki gibi tanımlanabilir. Şekil 2.6'da anlatılan Bug2 algoritmasının adımları aşağıda açıklanmıştır. Şekil 2.7'de Hedefe ulaşamayan bir Bug2 algoritması süreci görülmektedir.

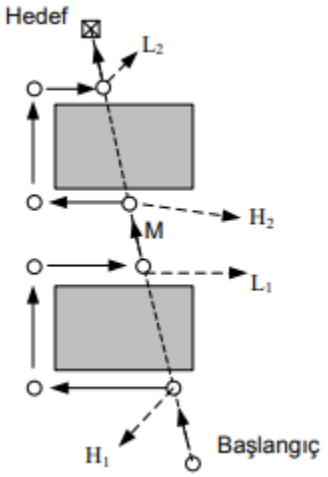
1) Başlangıç olarak robot başlangıç konumu ile hedef nokta arasında hayali bir M doğrusu tanımlanır. 2. Adıma geçilir

2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar M doğrusu üzerinde hedefe doğru ilerler.

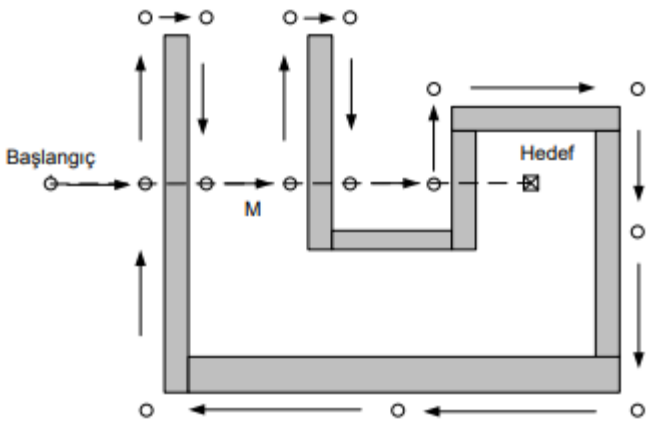
- Robot hedef ulaşırsa durur.
- Robot ilerlediği bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 3. Adıma geçilir.

3) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engeli takip eder.

- Robot hedef ulaşırsa durur.
- Robot engel takip ederken M doğrusu üzerindeki bir noktaya ulaşırsa bu nokta L noktası olarak adlandırılır. Robot engel takibini sonlandırır. 1. Adıma geçilir.
- Robot H noktasına ulaşırsa hedef ulaşılabilir değildir Robot durur.



Şekil 2.6 : A ortamında Bug2 algoritması



Şekil 2.7 : B ortamında Bug2 algoritması

2.1.4 Alg1 algoritması

Alg1 algoritması, Bug2 algoritması geliştirilmiş hali olarak Sankaranarayanan ve Vidyasagar tarafından önerilmiştir. Bug2 algoritmasının Şekil 2.8'de görüldüğü gibi aynı yolu tekrar tekrar takip etme olasılığı bulunmaktadır. Bunun önüne geçmek için Alg bir algoritması önceki H engelle ile karşılaşma noktaları ve L engel takibinden ayrılma noktalarını hatırlar ve bu noktaları daha kısa bir yol üretmek için algoritması içerisinde kullanır. Alg1 algoritmasının adımları aşağıda açıklanmıştır. Şekil 2.9'da Alg1 algoritmasının A ortamında Bug 2 algoritması ile aynı yolu izlediği görülebilmektedir. Şekil 2.7'de B ortamında Bug2 algoritmasının başarısız olduğu ortamda Alg1 algoritmasının nasıl başarılı olduğu görülebilmektedir. Alg1 algoritması 3. Adımındaki değişiklik ile H noktasına engeli soluna alarak engel takibi yaparak hedef noktaya ulaşabilmiştir.

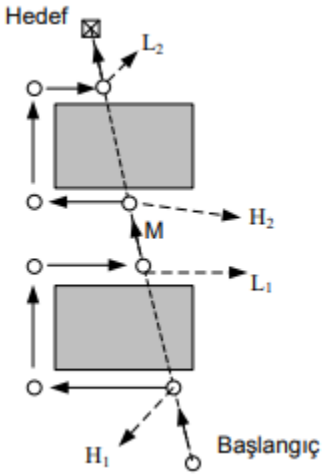
1) Başlangıç olarak robot başlangıç konumu ile hedef nokta arasında hayali bir M doğrusu tanımlanır. 2. Adıma geçilir

2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar M doğrusu üzerinde hedefe doğru ilerler.

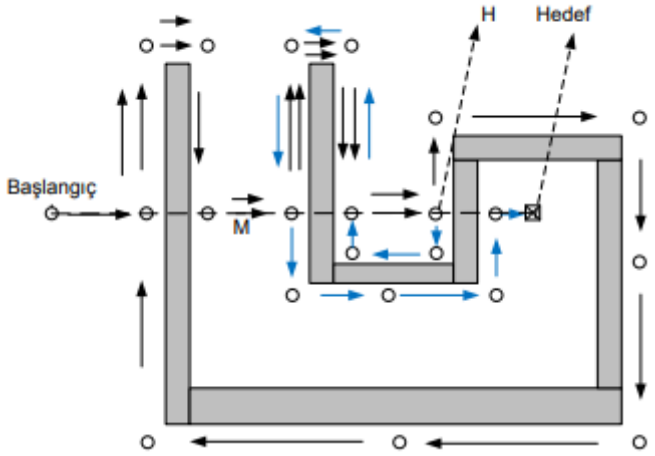
- Robot hedef ulaşırsa durur.
- Robot ilerlediği bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 3. Adıma geçilir.

3) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engeli takip eder.

- Robot hedef ulaşırsa durur.
- Aşağıdaki koşulları sağlayan bir Y noktası bulunması durumunda bu Y noktası L olarak tanımlanır. 2. Adıma gidilir.
  - ❖ Y noktası M doğrusu üzerindedir.
  - ❖ Y noktasının hedefe olan uzaklığı M doğrusu üzerindeki daha önce robotun bulunduğu tüm noktalardan daha azdır.
  - ❖ Y noktasında iken robot hedef noktaya doğru doğrusal olarak ilerleyebilmektedir.
- Robotun daha önceden tanımlanmış bir L veya H noktasına ulaşması durumunda robot H noktasına gidip engel robota göre solda kalacak şekilde engeli takip eder. Bu adım yeni bir L noktası bulunana kadar tekrarlanamaz.
- Robot bu adımdaki H noktasına ulaşırsa hedef ulaşılabilir değildir. Robot durur.



Şekil 2.8 : A ortamında Alg1 algoritması



Şekil 2.9 : B ortamında Alg1 algoritması

2.1.5 Alg2 algoritması

Alg2 algoritması, Alg1 algoritmasının geliştirilmiş hali olarak Sankaranarayanan ve Vidyasagar tarafından önerilmiştir. Alg2 algoritması M doğrusu kavramı yerine yeni bir engelden ayrılma kavramı getirmiştir. Şekil 2.10'da ve Şekil 2.11'de A ve B ortamlarında Alg2 algoritmasının yeni engelden ayrılma yöntemi sayesinde engeli sola alarak engeli takip

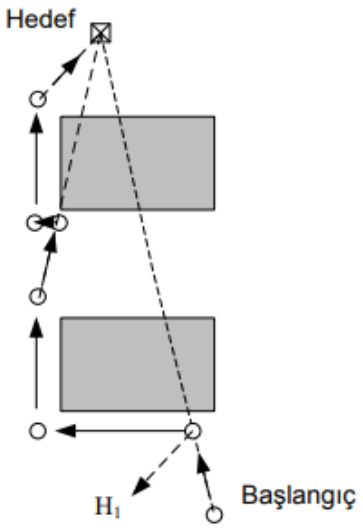
etme aşamasına Alg1 algoritmasına kıyasla daha kısa bir yol aldıktan sonra vardığı açıkça görülmektedir.

1) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar hedefe doğru bir doğru üzerinde ilerler. Hedefe doğru ilerlerken hedefe en yakın olduğu konumları Q noktası olarak tanımlar. Q robotun izlediği yol üzerinde robota en yakın olduğu noktadır.

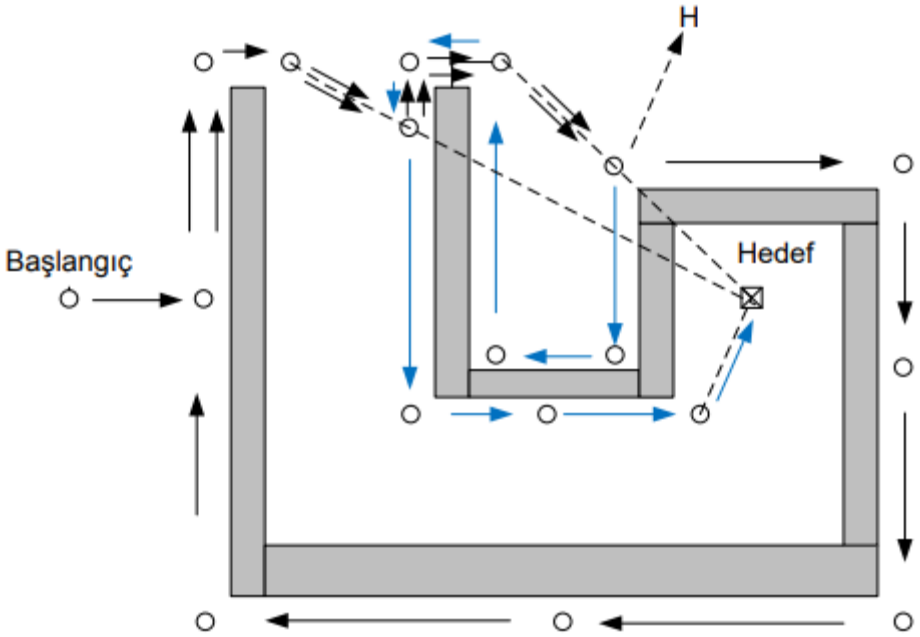
- Robot hedef ulaşırsa durur.
- Robot ilerlediği bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 2. Adıma geçilir.

2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engeli takip eder. Bu arada Q noktasını hedefe yaklaştıkça yeniden tanımlar.

- Robot hedef ulaşırsa durur.
- Aşağıdaki koşulları sağlayan bir Y noktası bulunması durumunda bu Y noktası L olarak tanımlanır. 2. Adıma tekrar gidilir.
  - ❖ Y noktası hedefe Q noktasında daha yakındır.
  - ❖ Y noktasında iken robot hedef noktaya doğru doğrusal olarak ilerleyebilmektedir.
- Robotun daha önceden tanımlanmış bir L veya H noktasına ulaşması durumunda robot H noktasına gidip engel robota göre solda kalacak şekilde engeli takip eder. Bu adım yeni bir L noktası bulunana kadar tekrarlanamaz.
- Robot bu adımdaki H noktasına ulaşırsa hedef ulaşılabilir değildir. Robot durur.



Şekil 2.10 : A ortamında Alg2 algoritması



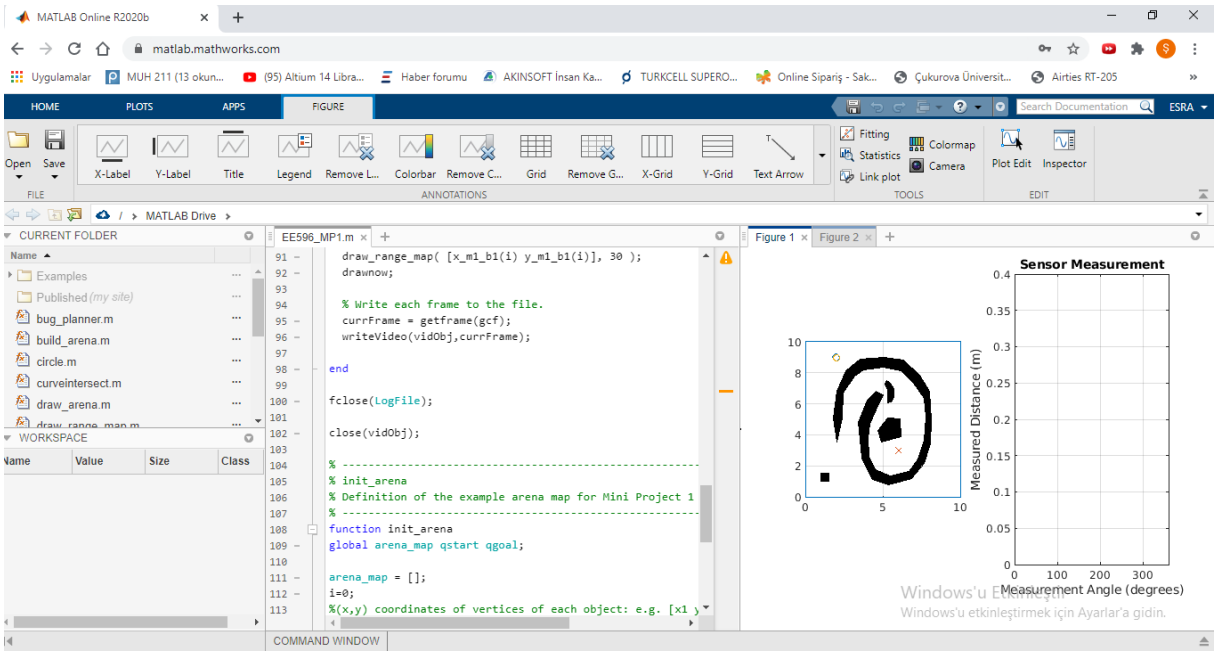
Şekil 2.11: B ortamında Alg2 algoritması

2.1.6 Diğer Bug Algoritmaları

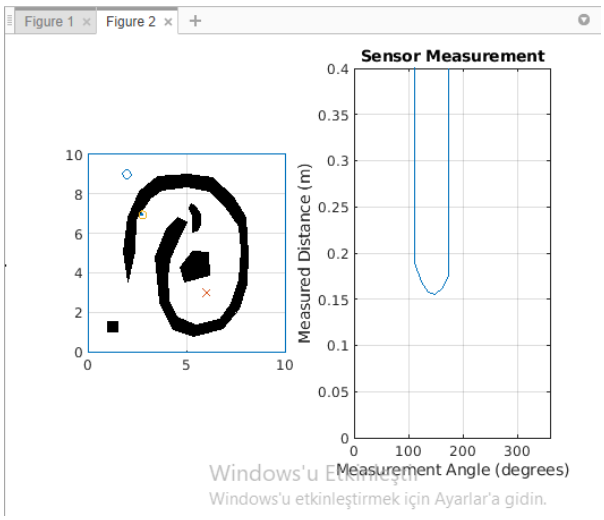
Bug algoritmalarının kolay programlanmaları ve geliştirilmeye açık olmaları nedenle pek çok farklı türü vardır. Buraya kadar gördüğümüz örneklerde olduğu gibi bu algoritmaların arasındaki fark engel takibinin bırakılacağı H noktasının belirlenmesi üzerinedir[2]

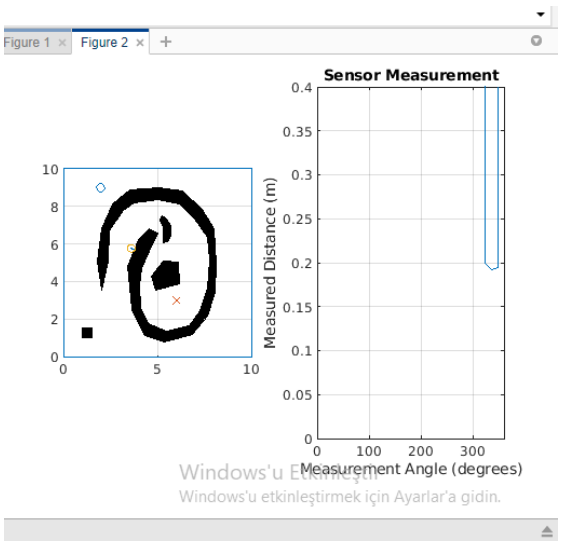
3.SİMÜLASYON SONUÇLARI

Projede Bug1 algoritması kullanılmıştır. Lisanslı Matlab üzerinde gerçekleştirilmiştir.



Şekil 3. 1: Simülasyon sonuçları





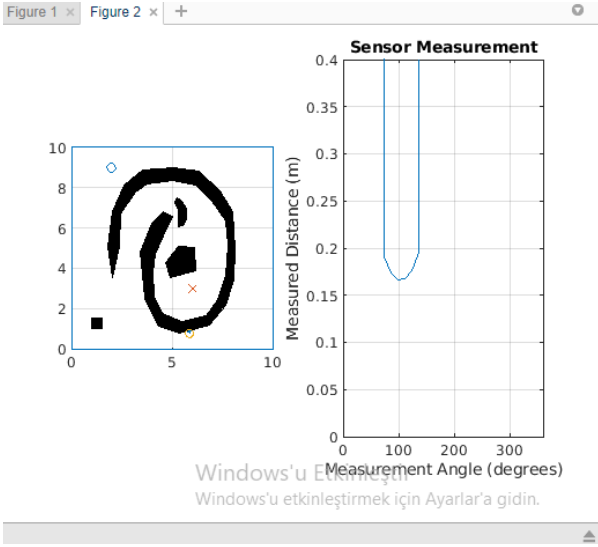


Figure 1
Figure 2
Sensor Measurement
Measured Distance (m)
Measurement Angle (degrees)
Windows'u etkinleştirmek için Ayarlar'a gidin.

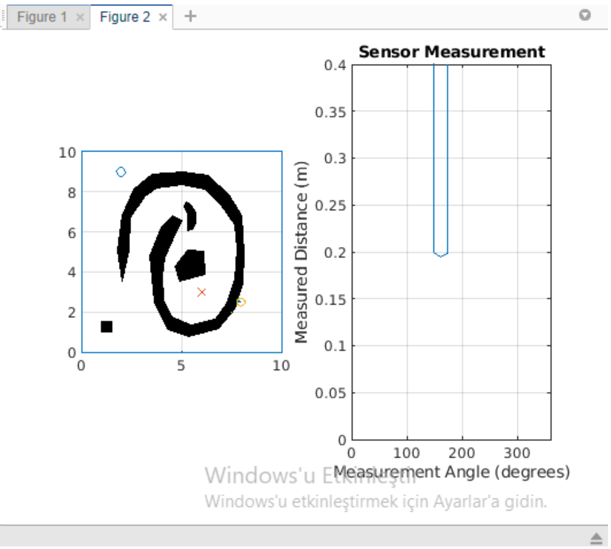


Figure 1
Figure 2
Sensor Measurement
Measured Distance (m)
Measurement Angle (degrees)
Windows'u etkinleştirmek için Ayarlar'a gidin.

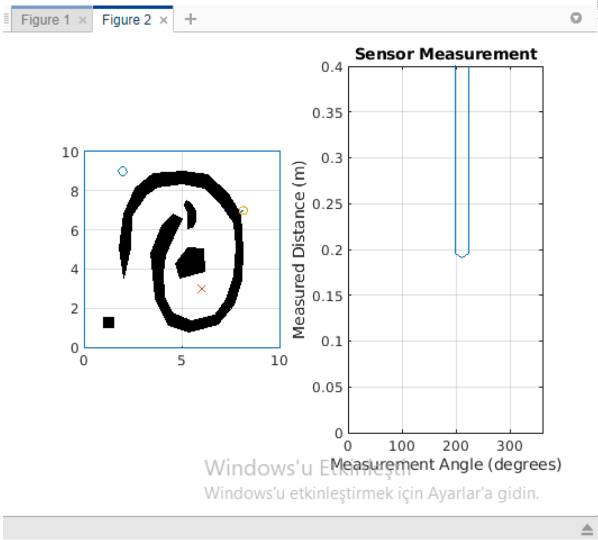


Figure 1
Figure 2
Sensor Measurement
Measured Distance (m)
Measurement Angle (degrees)
Windows'u etkinleştirmek için Ayarlar'a gidin.

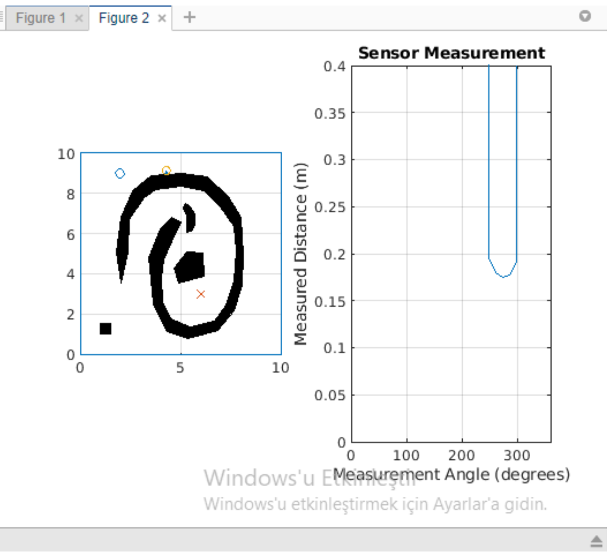
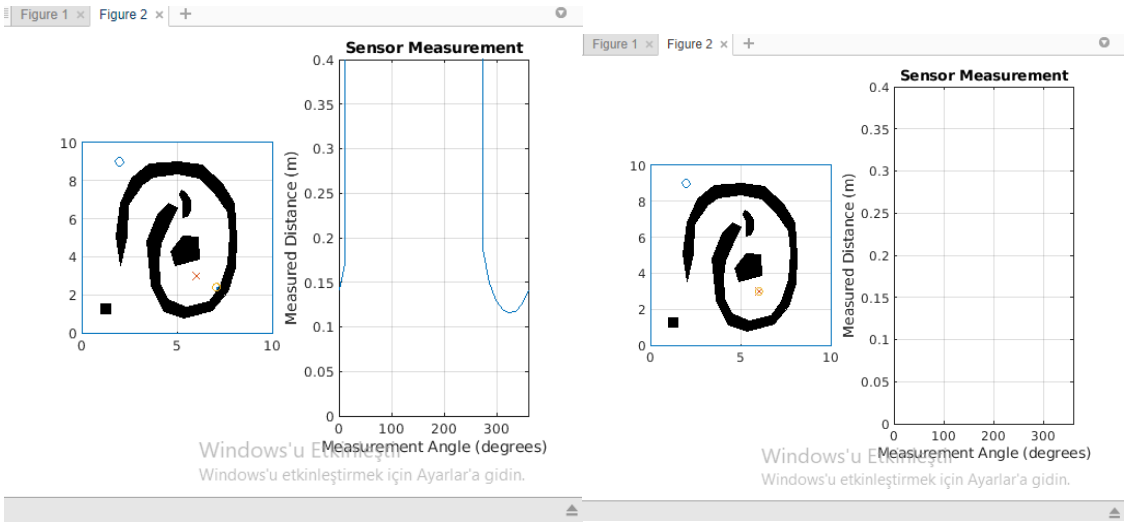
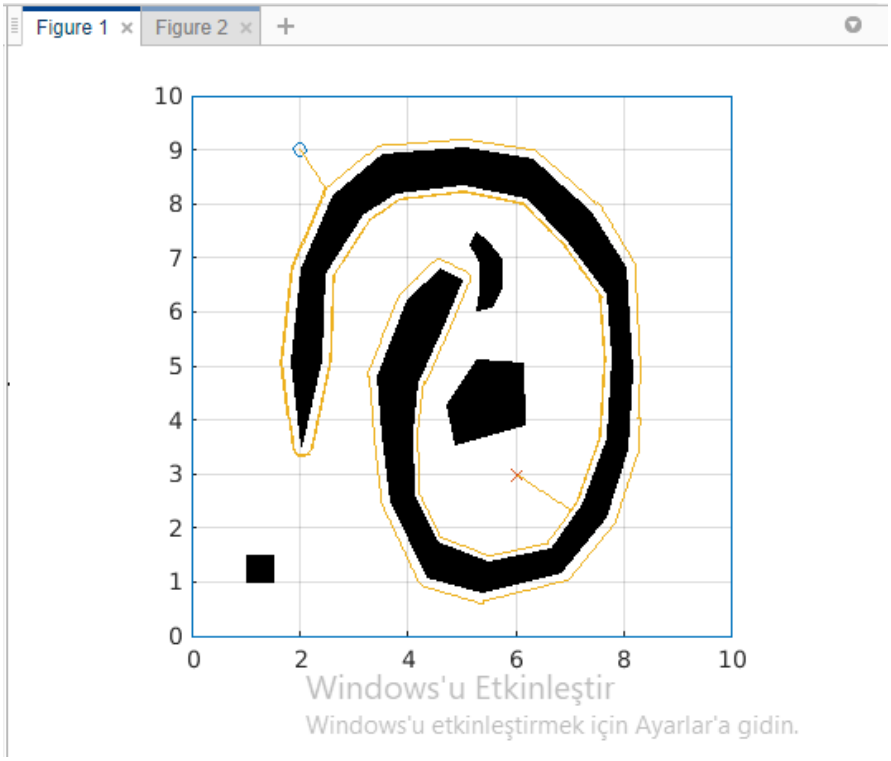


Figure 1
Figure 2
Sensor Measurement
Measured Distance (m)
Measurement Angle (degrees)
Windows'u etkinleştirmek için Ayarlar'a gidin.



Şekillerde de görüldüğü gibi algoritmamız ve kodumuz doğru çalışıp hedef noktaya ulaşılmıştır.



Şekil 3. 2: Algoritmanın izlediği yol

4.SONUÇ

Simülasyonda yaşadığımız tek sorun programın yavaş çalışmasıydı, bunu da çalıştırılan script ve komut sayısına bağlıyoruz. Kodu biraz daha basitleştirerek programı hızlandırabiliriz.

KAYNAKÇA

[1] https://dergipark.org.tr/en/download/article-file/631547

(Ziyaret Tarihi:16.12.2020)

[2]https://polen.itu.edu.tr/xmlui/bitstream/handle/11527/15495/10065452.pdf?sequence=2&isAllowed=y

(Ziyaret Tarihi: 9.12.2020)

[3] https://dergipark.org.tr/en/download/article-file/631547

(Ziyaret Tarihi:12.12.2020)

[4] https://www.energid.com/blog/what-is-global-path-planning-how-does-it-compare-to-local-path-planning

(Ziyaret Tarihi:15.12.2020)

Kodu:

[5] https://github.com/mustafaylmz1995/Bug_Algorithms

(Ziyaret Tarihi:10.12.2020)

EKLER

Global ve lokal planlama arasındaki farkı görebilmek için aşağıdaki videoları izleyebilirsiniz.

Ek-1 Global Path Planning

https://www.youtube.com/watch?v=txvXcQw-4QM&feature=emb_imp_woyt

Ek-2 Local Path Planning

https://www.youtube.com/watch?v=89ijHs1qOiw&feature=emb_logo