YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ROBOT SİMÜLASYONU
ve
KUMANDA ORTAMI

Bilgisayar Müh. Ersin DENİZ

FBE Bilgisayar Mühendisliği Anabilim Dalında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Sırma YAVUZ

İSTANBUL, 2009

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

A	Robotun ön teker orta noktasının konumu
A'	Robotun hareketi sonrasındaki ön teker orta noktasının konumu
B	Robotun arka tekerlerinin orta noktasının konumu
B'	Robotun hareketi sonrasındaki arka tekerlerinin orta noktasının konumu
d	Robotun ön teker orta noktası ve arka tekerlerinin orta noktası arasındaki mesafe
ds_A	Robotun hareketi esnasında ön tekerinin çizdiği yayın uzunluğu
ds_B	Robotun hareketi esnasında arka tekerlerinin orta noktasının çizdiği yayın uzunluğu
gs_A	Robotun ön teker orta noktasının hareket öncesi ve sonrasındaki konumlarını birleştiren doğru parçasının uzunluğu
gs_B	Robotun arka tekerlerinin orta noktasının hareket öncesi ve sonrasındaki konumlarını birleştiren doğru parçasının uzunluğu
k	En küçük kareler yöntemi ile doğru denklemi oluşturmak için kullanılacak nokta sayısı
m_{robot}	Robotun eğim açısının değeri
O	Ani dönme merkezi
r_A	Robotun ön teker orta noktası ile ani dönme merkezini birleştiren doğru parçasının uzunluğu
r_B	Robotun arka tekerlerinin orta noktası ile ani dönme merkezini birleştiren doğru parçasının uzunluğu
thr	Bir noktanın, denklemi bilinen doğrusal engel üzerinde olduğu kararının alınması için nokta ile doğru arasında olması gereken en fazla mesafe değeri
α	Robotun ön tekerinin dönme açısının değeri (*alfa*)
γ	Robotun hareketi esnasında ön tekerinin çizdiği yayı gören merkez açı değeri (*gama*)

KISALTMA LİSTESİ

CCITT	Comité Consultatif International Télégraphique et Téléphonique (Uluslararası Telgraf ve Telefon Danışma Komitesi)
CRC	Cyclic Redundancy Check (Dönüşsel Artıklık Denetimi)
DC	Direct Current (Doğru Akım)
dll	Dynamic-Link Library
ERSIM	Easy Robot SIMulation
GB	Gigabyte
GHz	Gigahertz
GNU	GNU's Not Unix
GPL	General Public License (Genel Kamu Lisansı)
hex	Hexadecimal (Onaltılı Sayı Sistemi)
IDE	Integrated Development Environment (Tümleşik Geliştirme Ortamı)
ITU-T	Telecommunication Standardization Sector (Telekomünikasyon Standardizasyon Sektörü)
jnilib	Java Native Interface Library
JRE	Java Runtime Environment (Java Çalışma Ortamı)
LGPL	Lesser General Public License (Kısıtlı Genel Kamu Lisansı)
MSB	Most Significant Bit (En Anlamlı Bit)
NiMH	Nickel-Metal Hydride
RAM	Random Access Memory (Rastgele Erişimli Bellek)
RF	Radio Frequency
SFD	Start Frame Delimiter
SLAM	Simultaneous Localization and Mapping (Eş Zamanlı Konum Belirleme ve Haritalama)
so	Shared Object Library
SYNCH	Synchronization
TCP/IP	Transmission Control Protocol / Internet Protocol
TM	Trade Mark (Ticari Marka)
USB	Universal Serial Bus (Evrensel Seri Yol)
V	Volt
XOR	Exclusive OR (Dışlayıcı VEYA)

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Gerçekleştirilen çalışmanın konusunu, Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde geliştirilmiş olan ve "Eş Zamanlı Konum Belirleme ve Haritalama" (SLAM-Simultaneous Localization and Mapping) araştırmalarında kullanılan gezgin robotun çalışmasının modellenerek bilgisayar ortamına aktarılması ve bu robotu yapılma gayesine uygun olarak geliştirilen algoritmalar ile kontrol edebilecek gerekli kumanda biriminin gerçekleştirilmesine yönelik çalışmalar oluşturmaktadır.

Her zaman ve bu çalışmayı gerçekleştirme sürecinde bana destek olan aile bireylerime, bilgisini ve zamanını sonuna kadar paylaşan değerli hocam ve tez danışmanım Yrd. Doç. Dr. Sırma YAVUZ'a en içten duygularımla teşekkür ediyorum.

Ayrıca, yüksek lisans eğitim hayatıma yaptıkları katkılardan dolayı Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı öğretim üyelerine, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu'na (TÜBİTAK), Türkiye Bilişim Derneği'ne (TBD) ve Turkcell İletişim Hizmetleri A.Ş.'ye teşekkürlerimi sunuyorum.

ÖZET

Bu çalışmanın amacı, Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde geliştirilmiş olan ve SLAM (Eş Zamanlı Konum Belirleme ve Haritalama) çalışmalarında kullanılan gezgin robotun bilgisayar ortamında simüle edilmesi ve kullanım amacına uygun olarak robotu ve simülasyonu kontrol edebilecek kumanda ortamının oluşturulmasıdır. Simülasyon uygulaması ve kumanda ortamı, NetBeans tümleşik geliştirme ortamı kullanılarak Java programlama dilinde geliştirilmiştir.

Simülasyon kullanıcıya kolay bir şekilde harita oluşturabilme, değişik konfigürasyonlarda robotlar tanımlayabilme, birden fazla aynı veya farklı tip robotu bir arada kullanabilme ve robot aygıtları tanımlayıp bunlara gürültü ekleyebilme imkânlarını sunmaktadır. Böylece, geliştirilmiş olunan bu ortam ile gerçek robotun tüm özellikleri ve davranışları simüle edilmiş, yeni algoritmaların geliştirilmesine ve karşılaştırılarak problemlerin tespitine olanak sağlanmıştır. Ayrıca kumanda ortamı ile de simülasyon üzerinde çalışacak olan araştırmacılara, kendi kontrol algoritmalarını kodlayabilmeleri ve simülasyon uygulaması üzerinde sonuçlarını gözlemleyebilmeleri imkânı verilmiştir.

Anahtar kelimeler: Çoklu robot simülasyonu, Gezgin robot kontrolü, SLAM (Eş Zamanlı Konum Belirleme ve Haritalama).

ABSTRACT

The aim of this study is to simulate a mobile robot at computer which is developed in Yıldız Technical University Computer Engineering Department and used for SLAM (Simultaneous Localization and Mapping) research and develop a command environment which can be used to control robot and simulation according to goal of usage. Simulation application and command environment are developed using NetBeans IDE in Java programming language.

Simulation allows users to create map easily, define various configuration robots, use more than one same or different type of robots with together, able to define robot devices and add noise to them. In this way, with the developed environment all the features and behaviors of real robots are simulated and possibility of developing new algorithms and detection of problems with comparing these algorithms are provided. Furthermore, also with the command environment opportunity of coding own algorithms and observing themselves studies in simulation are given to researchers who will study on simulation.

Keywords: Multi robot simulation, Mobile robot control, SLAM (Simultaneous Localization and Mapping).

1. GİRİŞ

Simülasyon (*benzetim*), gerçek hayattaki bir sistemin veya sürecin çalışmasının zaman üzerinden taklit edilmesi olarak tanımlanmaktadır (Banks vd., 2004). Simülasyonlar, sistemin yapay geçmişinin üretilmesine ve gerçek sistemin karakteristik özelliklerine dair çıkarımlar yapmak üzere bu geçmişin gözlemlenmesine olanak vermektedirler.

Simülasyonlar çoğunlukla bilgisayar üzerinde, gerçek sistemlerin modellenmesi şeklinde hayata geçirilmektedirler. Simülasyon yazılımları sistemin davranışını tanımlamak, analiz etmek ve "... olursa ne olur?" sorularına cevap vermek için kullanılmaktadır.

Simülasyonlar sağladıkları kazanımlardan dolayı havacılık, ekonomi, işletme, eğitim, tıp gibi birçok alanda yaygın olarak kullanılmaktadırlar. Bu kazanımlar genel olarak şu şekilde sıralanabilir:

- Yeni tasarım ve karar alma gibi süreçlerin sonuçlarının önceden kestirilerek daha iyi bir planlamanın yapılabilmesi,
- Zamanın daraltılması veya genişletilmesi şeklindeki uygulamalarla çok uzun süreler alabilecek olayların hızlı bir şekilde veya kısa süreli olayların yavaşlatılarak gözlemlenebilmesi,
- Gerçek uygulamaların insanlar üzerinde tehlikeli sonuçlar doğurabileceği durumlarda uygulanabilmesi.

Bu kazanımların yanında simülasyonların bazı durumlarda uygulanmaları etkin değildir. Eğer, gerçek sistemdeki problem analitik yöntemlerle çözülebiliyorsa, gerçek sistem üzerinde değişiklik ve deney yapmak daha kolaysa veya simülasyonun sağlayacağı kazanım, simülasyonun maliyetinden daha yüksek ise bu gibi durumlarda simülasyonun gerçekleştirilmesi uygun olmayacaktır.

Kısaca simülasyon, bir sistemi temsil edebilecek bir model oluşturma işlemi olarak da tanımlanabilir (Erkut, 1992). Oluşturulan model, temsil ettiği sistem üzerinde maliyetli olabilecek ve zaman alabilecek işlemlerin yapılabilmesine imkân sağlayacaktır.

2. AMAÇ

Bu çalışmanın amacı, Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde geliştirilmiş olan ve "Eş Zamanlı Konum Belirleme ve Haritalama" çalışmalarında kullanılan gezgin robotun bilgisayar ortamında simüle edilmesi ve yapılma amacına uygun olarak robotu ve simülasyonu kontrol edebilecek kumanda ortamının oluşturulmasıdır. Kumanda ortamı, simülasyon üzerinde çalışacak araştırmacıların kendi kontrol algoritmalarını kodlayarak simülasyon uygulaması üzerinde sonuçlarını gözlemleyebilecekleri şekilde tasarlanmıştır.

2.1 Neden Robot Simülasyonu

Projeye konu olan, eş zamanlı konum belirleme ve haritalama amaçlı olarak kullanılan gezgin robot üzerindeki çalışmalar, bölüm bünyesinde tahsis edilmiş olan bir laboratuvarda, tahta veya karton gibi yapay engellerle oluşturulmuş olan bir ortamda gerçekleştirilmektedir. Robotları geliştiren "Olasılıksal Robotik Grubu" ve bölüm öğrencilerinden oluşan birçok proje ekibi de robot üzerindeki çalışmalarına devam etmektedirler.

Çok sayıda kişinin kısıtlı sayıdaki robot üzerinde çalışma gereksinimi, çalışmaların sadece üniversitenin açık olduğu zamanlarda yapılabilmesi, bu çalışmalar sırasında robotlarda oluşabilecek donanımsal problemlere müdahale edebilecek birisinin çalışmalar esnasında bulunma zorunluluğu, oluşturulmuş olan yapay ortam üzerinde değişiklik yapmanın zorluğu ve robot üzerinde yapılması gereken veya yapıldığında oluşabilecek değişimleri gözlemlemek amacıyla yapılmak istenen donanımsal değişiklerin zaman alıcı olması gibi sebeplerle, gerçekleştirilmekte olan robot projesinde çeşitli aksaklıklar ve yavaşlamalar meydana gelebilmektedir. Tüm bu sorunlara çözüm üretebilecek, daha konforlu bir çalışma ortamı sunabilecek ve çalışmaları hızlandırabilecek bir çözüme ihtiyaç duyulmaktaydı. Bu noktada, oluşan bu ihtiyacı giderebilecek olan bu çalışmanın gerçekleştirilmesine karar verilmiştir. Bu çalışmanın temel amacı, gerçek robota ihtiyaç duymadan robotun tüm özelliklerini ve davranışlarını simüle edebilecek, yeni algoritmaların geliştirilmesine ve karşılaştırılarak problemlerin tespitine olanak sağlayacak bir ortamın oluşturulmasıdır.

Geliştirilen yazılımın hayata geçmesiyle elde edilecek faydalar şu şekilde sıralanabilir:

- Robotun çalışma ortamındaki engeller (*duvarlar*), kolay bir şekilde oluşturulabilecek ve değiştirilebilecektir. Bu sayede robotların çok farklı ortamlardaki davranışları gözlemlenebilecek ve kıyaslanabilecektir.
- Robotun gerçek çalışma anında karşılaşılabilecek olan ve elde edilen sonuçları

etkileyebilecek aygıt ölçümlerinin gürültülü veya çalışma zeminin kaygan olması gibi etmenlerin olmadığı ideal ortamda veya bu etmenlerin değişik oranlarda etkili olduğu şartlarda, robotların davranışları gözlemlenerek kıyaslamalar yapılabilecektir.

- Gezgin robota ihtiyaç duyulmadığı için istenilen zamanda, istenilen ortamda ve değişik sayıda ve tipte robot için çalışmalar gerçekleştirilebilecektir.
- Robot üzerindeki değişikler kolaylıkla yapılabileceği için geliştirilen algoritmaların hangi tipte, hangi sayıda robotla, hangi ortam koşullarında ne gibi sonuçlar doğurduğu gözlemlenebilecektir.
- Gezgin robot üzerinde gerçekleştirilmek istenilen değişikler önce simülasyon ortamında denenerek istenilen kazanımların elde edilip edilemediği belirlenebilecektir.

2.2 Gezgin Robot Simülasyonu Alanındaki Diğer Çalışmalar

Gezgin robot simülasyonu konusunda akademik ve ticari olarak çeşitli çalışmalar yapılmış ve yapılmaya devam etmektedir. Bu çalışmalardan en önemlileri ve dikkat çekenleri şunlardır:

2.2.1 Webots™

Cyberbotics şirketi tarafından geliştirilen, çoğunlukla akademik çevrelerce gezgin robotları modelleme, programlama ve simüle etme amaçlı olarak kullanılan profesyonel bir geliştirme ortamıdır. Karmaşık robotik tasarımlar yapılabilmekte, bir veya daha fazla, benzer veya farklı robotların aynı ortamı paylaşabildiği 3 boyutlu ortamlar oluşturulabilmektedir. Simüle edilmiş çok sayıdaki duyargalar ve sürücüler de robot tasarımında kullanılabilmektedir. Windows, Mac ve Linux işletim sistemlerinde çalışabilmektedir. Robot kontrolü C, C++, Java, Python ve MATLAB programlama dillerinde veya TCP/IP protokolünü kullanabilen üçüncü parti yazılımlarla gerçekleştirilebilmektedir (Michel, 2004; [2]).

2.2.2 Carmen

Carmen (*Carnegie Mellon Robot Navigation Toolkit*) açık kaynak kodlu, gezgin robot kontrolü için geliştirilmiş bir yazılımdır. 2 boyutlu robot ve duyarga simülasyonu, hareket planlama ve konum belirleme modülleri gibi birçok özelliğe sahiptir. Linux (Red Hat ve Suse) işletim sisteminde ve GNU GPL lisansı altında kullanılabilmektedir. C ve Java desteği bulunmaktadır [1].

2.3 Geliştirilen Simülasyonun Özellikleri ve Alternatifleriyle Karşılaştırılması

Gerçekleştirilen simülasyon ortamı, NetBeans IDE 6.5 kullanılarak Java programlama dilinde geliştirilmiş ve ERSIM (*Easy Robot SIMulation*) olarak adlandırılmıştır. Uygulama, Java'da geliştirilmesi sayesinde yaygın olarak kullanılan tüm işletim sistemlerinde (Linux, Windows, Mac ve Solaris) çalıştırılabilmektedir. Simülasyonun kontrolü, gezgin robotun haberleşme protokolüne uygun olarak yapılan iletişim ile gerçekleştirilmektedir. Böylece, bu protokole uygun olarak geliştirilmiş olan herhangi bir kumanda kodu ile simülasyon kontrol edilebilmektedir.

Alternatif uygulamalar incelendiğinde, esnek bir yapıya sahip ancak çok büyük boyutlu oldukları görülmektedir. Bu uygulamaların farklı tüm durumları karşılayabilecek genel amaçlı sistemler olmaları sebebiyle, gezgin robotun elektronik ve mekanik özelliklerinin bu ortamlara tanıtılması zorunluluğu oluşmaktadır. Gerçekleştirilen çalışmada, robotun kendine has özellikleri doğrudan tanımlandığı için bu özelliklerin oluşturulmasına ihtiyaç duyulmadan, sadece temel kullanım parametreleri ayarlanmak suretiyle simülasyon ortamı kullanılabilir hale getirilmiş olacaktır. Bu sayede fazla zaman kaybı yaşamadan ve yapılan işe odaklanılarak çalışmalar hızlı bir şekilde gerçekleştirilebilecektir. Ayrıca uygulama, birebir gerçek robot haberleşmesini ve protokol yapısını kullandığı için simülasyon kontrolü amacıyla geliştirilen yazılımlar aynı zamanda gerçek robot üzerinde de kullanılabilecektir.

Geliştirilen simülasyon ortamı, robotun elektronik aksamı üzerinde oldukça fazla değişiklik yapma imkânı sunarak büyük bir esneklik sağlamaktadır. Robotun mekanik tasarımı ise doğrudan tanımlandığı için farklı bir mekanik tasarımdaki (örneğin 4 tekerleği olan) bir robot simülasyonda tanımlanamayacaktır.

Simülasyonun temel amaçlarından birisi, farklı kontrol algoritmalarının değişik parametrelerle denenerek birbirleriyle kıyaslanabilmesidir. Bu amaç doğrultusunda iki boyutlu gösterim tercih edilmiş, uygulamanın çalışmasını yavaşlatacak ve karmaşıklaştıracak olan üç boyutlu gösterime gerek görülmemiştir.

3. GEZGİN ROBOT

Simüle edilen gezgin robot, Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nde "Olasılıksal Robotik Grubu" tarafından geliştirilmiştir. Gezgin robot, geliştirilen bir algoritma vasıtası ile çevrenin ön topolojik bilgilerine veya birtakım referans nesnelerinin yer bilgisine sahip olmadan ve ihtiyaç duymadan "Eş Zamanlı Konum Belirleme ve Haritalama" yapma amacı taşımaktadır. Robot bilinmeyen bir ortamda bilinmeyen bir noktadan hareketine başlayarak bir taraftan bu ortamın haritasını çıkarırken, diğer taraftan da kendi konumunu tahmin etme işlemini gerçekleştirmektedir (Thrun vd., 2005; Kurt, 2007).

Robot bahsedilen bu işlemi bir sistem dâhilinde yerine getirebilmektedir. Robot, elde ettiği verileri kablosuz olarak haberleştiği bilgisayara göndermekte, bilgisayar da aldığı bu verileri geliştirilmiş olan algoritma vasıtası ile değerlendirerek yapılacak olan harekete karar vermekte ve ürettiği kontrol işaretini robota iletmektedir. Üzerinde algoritmanın çalıştırıldığı bilgisayar ve kendine has özelliklere sahip olan robotun birlikte oluşturdukları sistem otonom olarak tanımlanmaktadır (Yavuz vd., 2006). Herhangi bir otonom sistemin en önemli görevlerinden birisi çevresi hakkında bilgi elde etmesidir (Siegwart ve Nourbakhsh, 2004). Bu bilgi edinimi, duyargalar aracılığı ile ölçüm yapılarak ve yapılan bu ölçümlerden anlamlı bilgiler çıkarılarak gerçekleştirilmektedir. Otonom bir sistem oluşturan bilgisayar ve robot birlikteliğinde, robot tarafı duyarga ölçümünü, bilgisayar tarafı ise bilgi çıkarımını gerçekleştirerek çevre hakkında bilgi elde edebilmektedirler (Şekil 3.1).

Şekil 3.1 Bilgisayar ve gezgin robotun oluşturduğu otonom sistem

Robot, algoritmanın ihtiyaç duyduğu verileri toplayabilecek ve bu verileri bilgisayara aktarabilecek şekilde tasarlanmış ve gerçeklenmiştir. Alt bölümlerde gezgin robot hakkında bilgiler verilecek, robotun kontrol edilişi 5. bölümde ele alınacaktır.

3.1 Robotun Mekanik Yapısı

Geliştirilen gezgin robot, önde bir ve arkada iki adet olmak üzere toplamda üç adet tekerlek üzerine yerleştirilen alüminyum malzemeden yapılmış bir platformdan oluşmaktadır. Önde yön vermeye yarayan ve çekme görevini yerine getiren bir tekerlek, arkada ise birbirinden

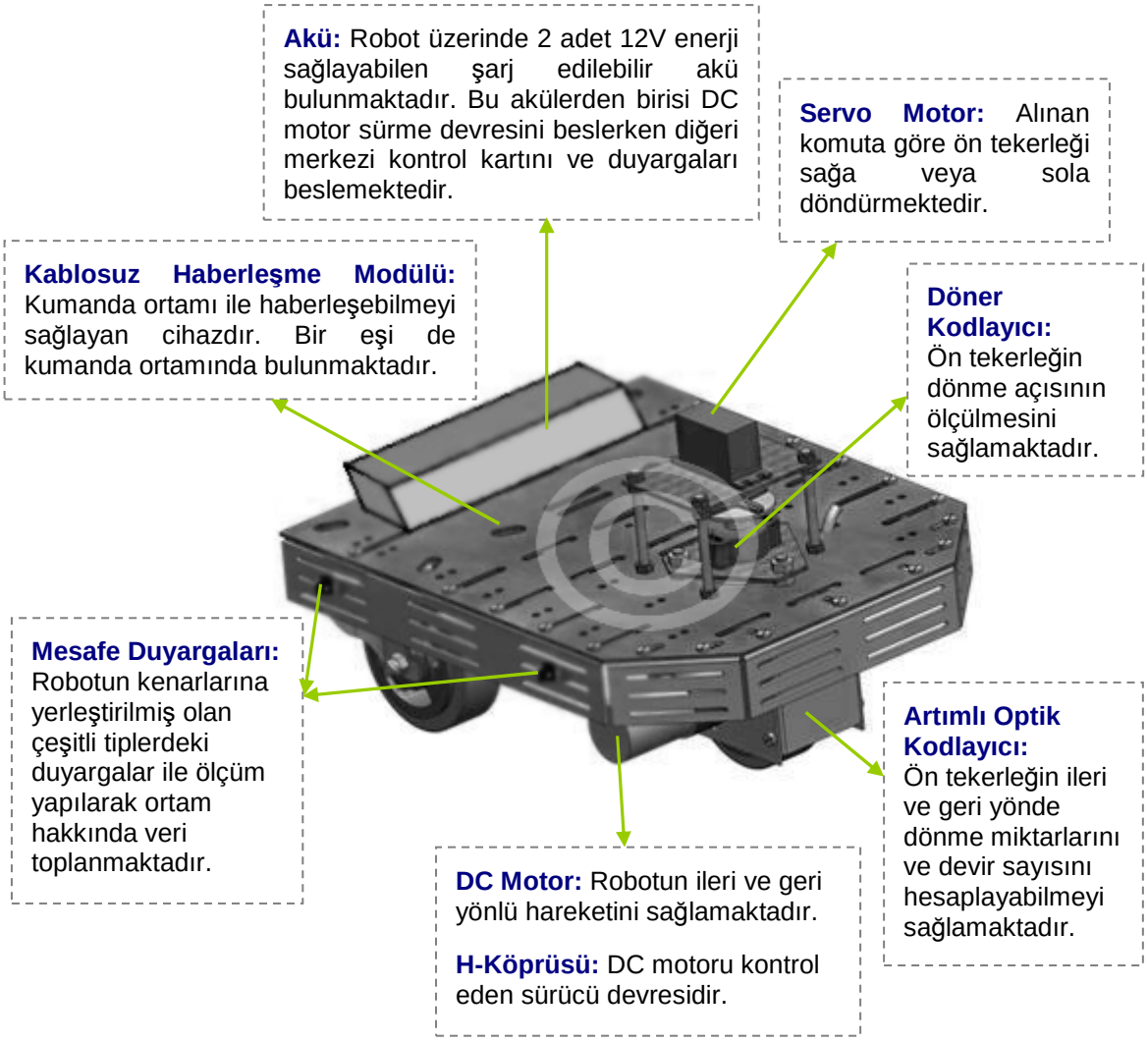
bağımsız iki adet tekerlek yer almaktadır. Ön tekerleğin, verilen kontrol işareti ile sağa ve sola döndürülme imkânı vardır. Arkadaki iki tekerlek ise, robot gövdesini ön tekerleğin çektiği yöne götürmekle görevlidir. Robotun tasarımına ait teknik çizimler Şekil 3.2 ve Şekil 3.3'de görülmektedir.

Şekil 3.2 Robotun önden görünümüne ait teknik çizim

Şekil 3.3 Robotun alttan görünümüne ait teknik çizim

3.2 Robotun Sahip Olduğu Mekanizmalar

Geliştirilen simülasyonda modellenmiş olan gerçek robota ait mekanizmaların bilinmesi hem simülasyonun daha iyi ve anlaşılır bir biçimde kullanılabilmesi hem de robotun kabiliyetlerinin anlaşılabilmesi açısından önemlidir. Şekil 3.4'de robotu oluşturan temel parçalar ve bunların fonksiyonları görülmektedir (Yavuz vd., 2009).



Şekil 3.4 Robotu oluşturan temel parçalar ve fonksiyonları

Robotu oluşturun parçaların yaptıkları görevler ve temel çalışma prensipleri şu şekildedir:

3.2.1 Merkezi Kontrol Kartı

Duyargalardan toplanan verilerin ve bilgisayardan gelen kontrol işaretlerinin işlendiği ana kontrol devresidir.

3.2.2 Hareket Mekanizması

Ön tekerlek mili üzerinde yerleştirilmiş olan bir adet DC motor ile robotun ileri ve geri yönlü hareketi sağlanmaktadır. Motor, ön tekerleğe göre, yatay eksende eş merkezli olarak konumlandırılmıştır. Kullanılan motor 4.5V-15V ile sürülebilen (mevcut robotlarda 12V), yüksek torklu (anlık 9000 g-cm) ve 47:1 dişli oranı özelliklerine sahiptir.

3.2.3 H-Köprü (H-Bridge) Devresi

DC motorun ileri, geri dönme ve fren kontrollerini sağlayan sürücü devresidir. Merkezi kontrol biriminden aldığı sinyale göre motorun hızını ve yönünü ayarlamaktadır.

3.2.4 Yön Mekanizması

Robot, mekanik özelliği gereği ön tekerleğin sağa veya sola dönüşü ile yönlendirilmektedir. Robotta yön mekanizması bir adet servo motor ile sağlanmaktadır. Motor, ön tekerleğe göre, düşey eş merkezli olarak konumlandırılmıştır. Kullanılan motor 6 Volt'ta 13.8 kg-cm tork ve 0.13 saniyede 60 derece dönme hız teknik özelliklerine sahip olan Futaba RS403PR Digital motorudur.

3.2.5 Hareket Yönünün Algılanması

Ön tekerleğe göre düşey eksende eş merkezli olarak konumlandırılan bir adet döner kodlayıcı (*rotary encoder*) ile ön tekerleğin dönme açısı ölçülebilmektedir. Bu şekilde gidilen yön bilinebilmekte, ön tekerlekte sürtünme veya boşluklar dolayısıyla kaynaklanabilecek sapmalar belirlenebilmektedir.

3.2.6 Çevrenin Algılanması

Robotun kenarlarına yerleştirilmiş olan mesafe duyargaları (*distance sensors / range finders*) ile ölçüm yapılarak çevrede bir cisim olup olmadığı algılanabilmektedir. Kullanılan duyargalar, kızılötesi (*infrared*) veya sesötesi (*ultrasonic*) tiptedir. Sayısı değişmekle birlikte genel olarak ön ve arkada bir, yanlarda ise ikişer tane olmak üzere, toplam altı adet duyarga robot üzerine konumlandırılmaktadır.

3.2.7 Alınan Yolun Ölçülmesi

Ön tekerlek mili üzerine yerleştirilmiş olan bir adet artımlı optik kodlayıcı (*incremental encoder*) ile alınan yol miktarı hesaplanabilmektedir. Artımlı kodlayıcının mekanik bağlantısı, ön tekerleğe göre yatay eksende eşmerkezli olarak yapılmıştır. Kodlayıcıya ait, 2 bayt ileri devir sayısı bilgisi, 2 bayt geri devir sayısı bilgisi ve 1 bayt da tekerlek devir bilgisi tutulmaktadır. İleri ve geri devir sayısı tekerin 1 tam turunda 512 artım yapmaktadır.

3.2.8 Robot ile Kumanda Birimi Arasındaki Haberleşmenin Sağlanması

Kumanda biriminden robota gönderilen komutlar veya robottan kumanda birimine gönderilen

verilerin iletimi Bölüm 3.3'de anlatılacak olan protokol yapısına göre RF veya Bluetooth alıcı verici modüller üzerinden kablosuz olarak sağlanmaktadır.

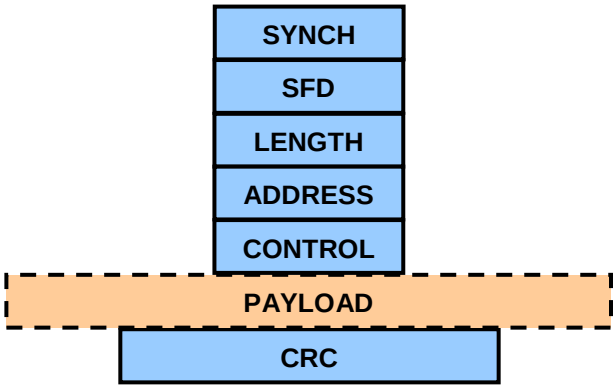
3.2.9 Enerjinin Sağlanması

Sisteme enerji sağlamak üzere, gezgin platform üzerine yerleştirilmiş olan şarj edilebilir NiMH pillerden yararlanılmaktadır.

3.3 Robot Haberleşme Protokolü ve Kontrolü

Robot, hareketini aldığı kontrol işaretine göre gerçekleştirmektedir. Robottan alınan duyarga ölçüm verilerine göre bir karar verilerek robota gerekli kontrol işareti, kontrol birimi tarafından gönderilmektedir. Robot ile kontrol birimi arasındaki iletişim, RF veya RF'in daha özelleşmiş bir biçimi olan Bluetooth (2.4-GHz) teknolojisi ile kablosuz olarak gerçekleştirilmektedir. Bu iletişimi sağlayacak cihazlar robot veya kontrol biriminin herhangi bir bağlantı noktasına (seri port veya USB) takılarak kullanılmaktadır.

İletişim için robotu geliştiren grup tarafından tanımlanmış olan bir protokol kullanılmaktadır. Bu protokoldeki paket yapısının içeriği şu şekildedir (Şekil 3.5):



Şekil 3.5 Haberleşme protokolü paket yapısı

SYNCH (*Synchronization*): Eş zamanlama baytıdır. Haberleşmenin başladığını belirtmektedir. Değeri, 55_{hex} (01010101_2) olarak tanımlanmıştır.

SFD (*Start Frame Delimiter*): Paket sınırlayıcısı başlangıç baytıdır. Verinin başladığını belirtmektedir. Değeri, $7E_{hex}$ (01111110_2) olarak tanımlanmıştır.

LENGTH: Kendisinden sonra gelen verinin uzunluğunu tanımlayan bayttır. Address, Control, Payload ve CRC baytlarının toplam uzunluğunu bayt cinsinden ifade etmektedir. Pozitif tamsayı değerler alabilmektedir ve ifade aralığı 0-255 arasındadır.

ADDRESS: İletişimin taraflarını tanımlayan bayttır. İlk dört bit gönderen tarafın, sonraki dört bit ise alıcı tarafın adresini tanımlamaktadır. Örneğin paketi gönderen tarafın adresi 3_{hex}, paketi alacak tarafın adresi de C_{hex} ise adres baytı $3C_{hex}$ olarak tanımlanmaktadır.

CONTROL: Kontrol verisini içeren bayttır ve paketin içeriğini tanımlamaktadır. Alabileceği değerler ve anlamları Çizelge 3.1'de görüldüğü gibidir.

Çizelge 3.1 Haberleşme protokolünde kullanılan kontrol baytları ve anlamları

Kontrol Baytı	**Değer**	**Açıklama**
Acknowledge	00_{hex}	Son paket başarıyla alındı
Negative-Acknowledge	01_{hex}	Son paket alınırken zaman aşımı oldu
Collision	02_{hex}	Robot bir yere çarptı
Error	03_{hex}	Hatalı veri paketi
Hello	04_{hex}	Bağlantı kuruldu onayı (*Handshaking*)

PAYLOAD: Pakette asıl verinin bulunduğu bölümdür ve paketin esnek bir yapı kazanmasını sağlamaktadır. Bu bölümde, robotta bulunan her aygıta verilmiş olan ve o aygıta has olarak tanımlanan bir baytlık adres değerleri kullanılmaktadır. Robottaki aygıtlar iki temel gruba ayrılmaktadırlar ve payload bölümünde adreslerin kullanımı bu ayrıma göre ve paketi gönderen ve alan tarafların görevlerine göre yapılmaktadır.

Okunabilir aygıtlar (*readable devices*), en anlamlı biti (*MSB*) 0 olan adres değerleriyle tanımlanmaktadırlar. Bu aygıtların görevi, kontrol birimine veri sağlayacak çeşitli boyutlarda olabilen ölçümler yapmaktır. Kontrol birimi, okunabilir bir aygıtın ölçüm değerini, robota gönderdiği paketteki payload bölümüne bu aygıtın adres bilgisini koyarak istemekte, robot ise bu aygıtın ölçüm değerini kontrol birimine gönderdiği paketin payload bölümüne aygıtın adres bilgisini ve ardından ölçüm değerini koyarak iletmektedir. Robot üzerindeki okunabilir aygıtlar mesafe duyargaları, döner kodlayıcı ve artımlı optik kodlayıcıdır. Mesafe duyargaları, ölçüm sonucu olarak 2 bayt boyutunda bir pozitif tamsayı değer üretmektedirler. Bu değer, duyarganın kendisine has ölçüm şekline göre bir denkleme bağlı olarak algılanan engele olan mesafeyi ifade etmektedir. Döner kodlayıcıda duyargalar gibi ölçüm sonucu olarak 2 bayt boyutunda bir pozitif tamsayı değer üretmektedir. Bu değer de bir denkleme bağlı olarak ön tekerleğin dönme açısını ifade etmektedir. Artımlı optik kodlayıcı için ise ölçüm sonucu olarak 5 bayt uzunluğunda bir sonuç üretilmektedir Bu sonucun ilk iki baytı ileri devir miktarını, sonraki iki baytı geri tur miktarını ve en sondaki bir baytı ise tekerleğin toplam devir sayısını ifade etmektedir. Sonuçların tamamı pozitif tamsayı değerlerdir.

Diğer tür aygıtlar ise yazılabilir aygıtlardır (*writable devices*). Bu tür aygıtlar, en anlamlı biti (*MSB*) 1 olan adres değerleriyle tanımlanmaktadırlar. Bu aygıtların görevi, alınan kontrol işaretine göre çalışmasını değiştirerek robotun belirlenmiş bir hareketi yapabilmesini sağlamaktır. Kontrol birimi, yazılabilir bir aygıtın değerini, robota gönderdiği paketteki payload bölümüne bu aygıtın adres bilgisini ve ardından kontrol işaretinin değerini koyarak değiştirebilmektedir. Robot üzerindeki yazılabilir aygıtlar DC ve servo motorlardır. DC motor, gönderilen 1 baytlık kontrol işaretine bağlı olarak ön tekerleği ileriye veya geriye doğru belirli bir hızda döndürmekte, servo motor ise gönderilen 1 baytlık kontrol işaretine bağlı olarak ön tekerleği sağa veya sola doğru belirli bir açıyla konumlandırmaktadır.

CRC (*Cyclic Redundancy Check*): Veri paketinin iletimi esnasında oluşabilecek hataların tespiti amacıyla kullanılan 2 baytlık hata tespit kodudur. Farklı boyutlarda olabilen ve değişik polinomlar kullanılarak hesaplanabilen CRC kodu için ITU-T tarafından V.41 tavsiye kararında tanımlanan ve CRC-16-CCITT ismiyle bilinen $x^{16} + x^{12} + x^5 + 1$ (1021_{hex}) polinomu kullanılmaktadır. CRC kodu, haberleşme paketindeki Address, Control ve Payload baytları için hesaplanmaktadır.

Çizelge 3.2'de örnek bir CRC kodunun oluşturulması görülmektedir. 11010011101100_2 kodu için 1011_2 polinomu ile CRC'nin üretilişi adım adım gösterilmektedir. Hesaplamada, giriş dizisi ile CRC polinomu XOR'lanarak çıkış dizisi elde edilmekte ve CRC polinomu soldaki ilk 1 girişinin altına getirilerek işlem tekrarlanmaktadır.

Çizelge 3.2 CRC kodu oluşturma örneği

	1	1	0	1	0	0	1	1	1	0	1	1	0	0
⊕	1	0	1	1	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	0	1	1	0	0	0	1	1	1	0	1	1	0	0
⊕		1	0	1	1	↓	↓	↓	↓	↓	↓	↓	↓	↓
	0	0	1	1	1	0	1	1	1	0	1	1	0	0
⊕			1	0	1	1	↓	↓	↓	↓	↓	↓	↓	↓
	0	0	0	1	0	1	1	1	1	0	1	1	0	0
⊕				1	0	1	1	↓	↓	↓	↓	↓	↓	↓
	0	0	0	0	0	0	0	1	1	0	1	1	0	0
⊕								1	0	1	1	↓	↓	↓
	0	0	0	0	0	0	0	0	1	1	0	1	0	0
⊕									1	0	1	1	↓	↓
	0	0	0	0	0	0	0	0	0	1	1	0	0	0
⊕										1	0	1	1	↓
	0	0	0	0	0	0	0	0	0	0	1	1	1	0
⊕											1	0	1	1
	0	0	0	0	0	0	0	0	0	0	0	1	0	1

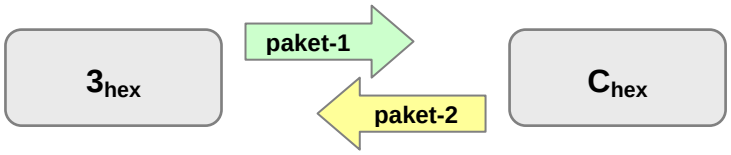
Sonuç olarak 4 bitlik CRC kodu 0101_2 olarak bulunmaktadır.

Gerçekleştirilen iletişimde kural olarak, bir bayttan büyük değerlerde öncelik yüksek anlamlı bayta verilmiştir ve iletişimde önce yüksek anlamlı bayt gönderilmektedir (*big-endian*).

3.3.1 Haberleşme Paketi Örnekleri

Haberleşmenin başlangıcında iletişimin tarafları birbirlerine Hello paketi göndererek iletişim için hazır olunduğunu onaylamaktadırlar (*handshaking*). Hello paketini alan taraf yine Hello paketi ile yanıt vermektedir. Bu paket yapısında, payload bölümünde hiçbir veri yer almamaktadır.

3_{hex} ve C_{hex} adresli iki tarafın birbirleri ile bu türde haberleştiğinde oluşturulacak paket yapıları Şekil 3.6'da görüldüğü gibidir.

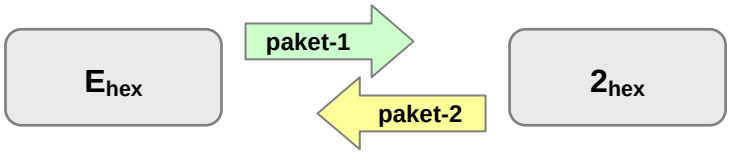


paket-1	
SYNCH	55_{hex}
SFD	$7E_{hex}$
LENGTH	04_{hex}
ADDRESS	$3C_{hex}$
CONTROL	04_{hex}
CRC	$1D_{hex}$
	73_{hex}

paket-2	
SYNCH	55_{hex}
SFD	$7E_{hex}$
LENGTH	04_{hex}
ADDRESS	$C3_{hex}$
CONTROL	04_{hex}
CRC	$1E_{hex}$
	$8C_{hex}$

Şekil 3.6 Hello paketi gönderilmesi örnek paket yapısı

Okunabilir aygıtların ölçüm değerleri bu aygıtların adresleri gönderilerek istenmekte, karşılığı ise okunabilir aygıtın adresinin ardından ölçüm değeri gönderilerek iletilmektedir. Bu tür iletişime örnek olarak E_{hex} adresli tarafın, 2_{hex} adresinden 01_{hex} ve 02_{hex} adresli aygıtların ölçüm değerlerini istediğini ve 2_{hex} adresli tarafın ise 01_{hex} adresli aygıtın ölçüm değerini 613 (265_{hex}), 02_{hex} adresli aygıtın ölçüm değerini ise 556 ($22C_{hex}$) olarak ölçtüğünü varsayarsak, haberleşmede oluşturulacak paket yapıları Şekil 3.7'de görüldüğü gibi olacaktır.

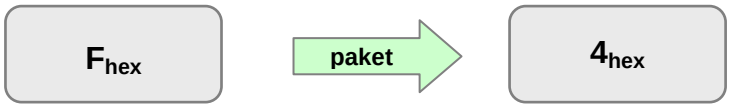


paket-1	
SYNCH	55_{hex}
SFD	$7E_{hex}$
LENGTH	06_{hex}
ADDRESS	$E2_{hex}$
CONTROL	00_{hex}
PAYLOAD	01_{hex}
	02_{hex}
CRC	FE_{hex}
	31_{hex}

paket-2	
SYNCH	55_{hex}
SFD	$7E_{hex}$
LENGTH	$0A_{hex}$
ADDRESS	$2E_{hex}$
CONTROL	00_{hex}
PAYLOAD	01_{hex}
	02_{hex}
	65_{hex}
	02_{hex}
	02_{hex}
	$2C_{hex}$
CRC	DD_{hex}
	$1F_{hex}$

Şekil 3.7 Okunabilir aygıt ölçüm değeri istenmesi örnek paket yapısı

Yazılabilir aygıtların değerleri ise bu aygıtların adresleri ve ardından 1 baytlık kontrol işareti gönderilerek değiştirilebilmektedir. Bu tür iletişime örnek olarak F_{hex} adresli tarafın, 4_{hex} adresinden 80_{hex} adresli yazılabilir aygıtı 135 (87_{hex}) değerine ayarlamasını istediğini varsayalım. Bu haberleşme gerçekleştiğinde oluşturulacak paket yapıları Şekil 3.8'de görüldüğü gibi olacaktır.



paket	
SYNCH	55_{hex}
SFD	$7E_{hex}$
LENGTH	06_{hex}
ADDRESS	$F4_{hex}$
CONTROL	00_{hex}
PAYLOAD	80_{hex}
	87_{hex}
CRC	$2B_{hex}$
	$8B_{hex}$

Şekil 3.8 Yazılabilir aygıt değerinin değiştirilmesi örnek paket yapısı

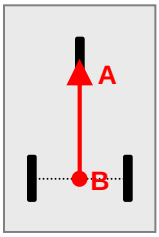
3.4 Gezgin Robot Kinematiği

Kinematik, genel olarak mekanik sistemlerin davranışlarını incelemektedir (Manseur, 2006). Gezgin robot kinematiği ise hareket halinde olan robotun konumunun bulunması ile ilgilenmektedir. Geliştirilen robotun kinematiğinin bilinmesi, robotun pozisyonunun bulunabilmesine buda robotun yapılış gayesi olan konum belirleme ve haritalama işleminin gerçekleştirilmesine olanak sağlamaktadır.

Robotun bilinen bir pozisyondan, verilen teker dönme açısı ve mesafe bilgisi ile hangi konuma gideceğinin bilinmesi düz kinematiği, yine robotun bilinen bir pozisyondan istenilen bir konuma gitmek için tekerleğinin dönme ve gidilmesi gereken mesafe bilgisinin bilinmesi ise ters kinematiği oluşturmaktadır (Siegwart ve Nourbakhsh, 2004).

Geliştirilen robotun düz kinematiğinin bilinmesi ile robotu modellemek için gerekli bilgi sağlanmış olunacaktır. Robotun ilk konumu, ön tekerleğin dönme miktarı (döner kodlayıcıdan elde edebilir) ve gidilen mesafe (artımlı optik kodlayıcıdan elde edebilir) bilindiğinde robotun sonraki konumunun nasıl bulunabileceği ilerleyen bölümlerde açıklanacaktır.

Gezgin robotun kinematiği, robot üzerinde olduğu varsayılan bir vektör referans alınarak çıkarılacaktır. Robotun arka tekerlerinin orta noktası (*B noktası*) vektörün başlangıç noktasını, robotun ön teker orta noktası (*A noktası*) ise vektörün bitiş noktasını oluşturmaktadır. Ayrıca vektörün yönü robotun yönünü göstermektedir (Şekil 3.9).



Şekil 3.9 Robotun kinematik hesaplamalarında referans alınan vektör

Robot, ön tekerleğinin bir yöne döndürülmesi ile gövdesi ile birlikte bir doğrultuya yönlendirilmiş olmaktadır. Bu hareketin yapılması ile geometrik olarak ön teker düzlemine dik olan yatay bir hareket doğrusu (r_A) çizilir. Hareket doğrusu ile arka tekerlerin düzleminin kesiştiği nokta herhangi bir anda yapılan dönme hareketinde arabanın çizeceği yayın merkez noktasıdır (*O noktası*). Bu noktaya da ani dönme merkezi denilmektedir (Şekil 3.10).

Şekil 3.10'daki gezgin robotun I konumundan II konumuna olan hareketi incelenerek robotun kinematiği çıkarılacaktır. Bu ve sonraki hareket tanımlayan durumlarda kullanılan simgeler ve anlamları şu şekildedir:

A : Robotun ön teker orta noktasının konumu

B : Robotun arka tekerlerinin orta noktasının konumu

d : Robotun ön teker orta noktası ve arka tekerlerinin orta noktası arasındaki mesafe, $|AB|$

A' : Robotun hareketi sonrasındaki ön teker orta noktasının konumu

B' : Robotun hareketi sonrasındaki arka tekerlerinin orta noktasının konumu

O : Ani dönme merkezi

α : Robotun ön tekerinin dönme açısının değeri (*alfa*)

ds_A : Robotun hareketi esnasında ön tekerinin çizdiği yayın uzunluğu, $|\widehat{AA'}|$

ds_B : Robotun hareketi esnasında arka tekerlerinin orta noktasının çizdiği yayın uzunluğu, $|\widehat{BB'}|$

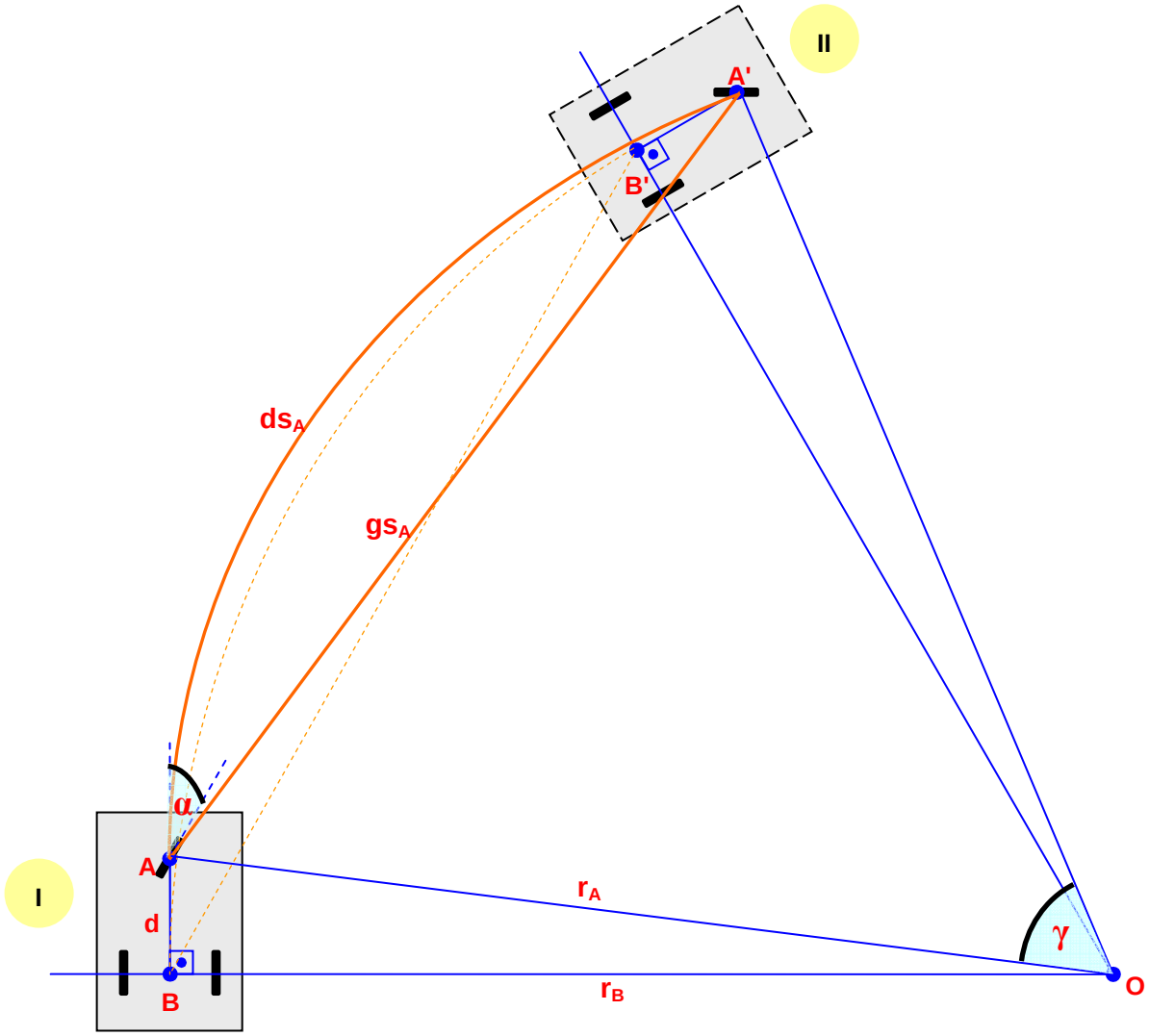
γ : Robotun hareketi esnasında ön tekerinin çizdiği yayı gören merkez açı değeri $\gamma(gama) = m(A\hat{O}A')$ (aynı zamanda hareket anındaki açısal hızdır)

gs_A : $\widehat{AA'}$ yayını gören kirişin uzunluğu, $|AA'|$

gs_B : $B\widehat{B'}$ yayını gören kirişin uzunluğu, $|BB'|$

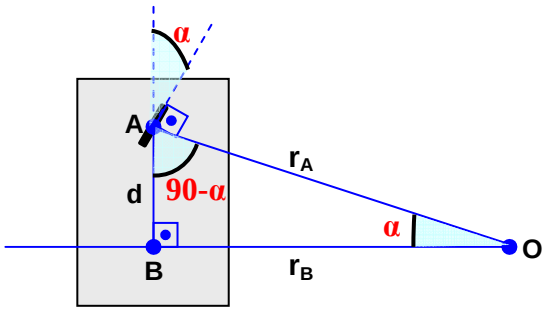
r_A : $|OA|$ uzunluğu

r_B : $|OB|$ uzunluğu



Şekil 3.10 Gezgin robotun dik konumdan ileri yönlü hareketi

r_A ve r_B mesafelerinin hesaplanabilmesi için *AOB* üçgeninden yararlanılabilir (Şekil 3.11). *AOB* üçgeninin özelliklerinden yararlanılarak (3.1), üçgenin iç açılarının toplamından da (3.2) denklemi yazılabilir.



Şekil 3.11 Robotun hareket öncesi konumundaki geometrik durum

$$m(B\hat{A}O) = 90 - \alpha \tag{3.1}$$

$$m(A\hat{O}B) = \alpha \tag{3.2}$$

Pisagor teoreminden yararlanılarak (3.3) yazılabilir.

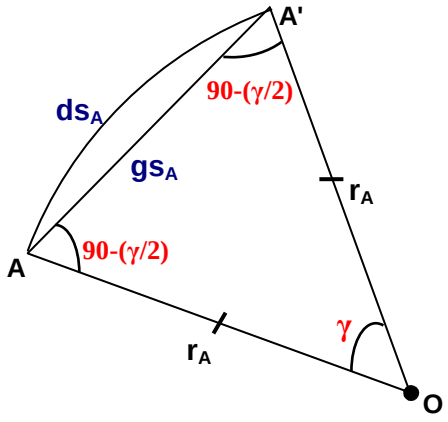
$$r_B{}^2 + d^2 = r_A{}^2 \tag{3.3}$$

Şekildeki üçgenin trigonometrik özelliklerinden yararlanılarak r_A ve r_B, (3.4) ve (3.5)'deki gibi hesaplanabilir.

$$r_A = \frac{d}{\sin\alpha} \tag{3.4}$$

$$r_B = \frac{d}{\tan\alpha} \tag{3.5}$$

$\widehat{AA'}$ yayından yararlanılarak γ değeri, AOA' ikizkenar üçgeninden yararlanılarak da gs_A bulunabilir (Şekil 3.12).



Şekil 3.12 Robotun hareketi öncesi ve sonraki konumlarında ön teker orta noktasının geometrik durumu

Çember özelliklerinden yararlanılarak (3.6) yazılabilir ve (3.7) bulunur.

$$\frac{ds_A}{2\pi r_A} = \frac{\gamma}{360} \tag{3.6}$$

$$\gamma = \frac{ds_A}{2\pi r_A} \times 360 \tag{3.7}$$

Kosinüs teoreminden yararlanarak (3.8) yazılabilir ve (3.9) bulunur.

$$gs_A{}^2 = r_A{}^2 + r_A{}^2 - 2 \times r_A{}^2 \times \cos\gamma \tag{3.8}$$

$$gs_A = \sqrt{2} \times r_A \times \sqrt{1 - \cos\gamma} \tag{3.9}$$

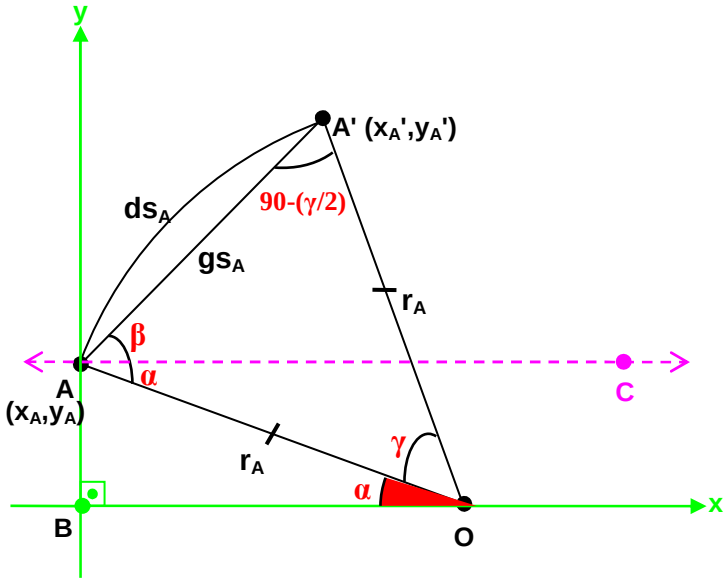
Ayrıca gs_A daha farklı bir yöntemle (3.11)'deki gibi de bulunabilir.

Sinüs teoreminden yararlanarak (3.10) yazılabilir ve (3.11) bulunur.

$$\frac{gs_A}{\sin\gamma} = \frac{r_A}{\sin\left(90 - \frac{\gamma}{2}\right)} \tag{3.10}$$

$$gs_A = \frac{r_A \times \sin\gamma}{\sin\left(90 - \frac{\gamma}{2}\right)} \tag{3.11}$$

Şekil 3.13'de görüldüğü gibi apsis eksenine paralel ve $A(x_A,y_A)$ noktasından geçecek şekilde bir doğru çizildiğinde OAC ve AOB açıları içters açılar olacağından değerleri birbirlerine eşit ve α olmaktadır. Bundan yararlanılarak β açısı aşağıdaki denklemlerdeki gibi hesaplanabilir.



Şekil 3.13 Robotun dik konumdan ileri yönlü hareketi sonrasındaki ön teker orta noktasının konumunun bulunması

$$m(O\hat{A}A') = 90 - \frac{\gamma}{2} \tag{3.12}$$

(3.12)'den yararlanılarak (3.13) yazılabilir.

$$\alpha + \beta = 90 - \frac{\gamma}{2} \tag{3.13}$$

(3.13)'den (3.14) elde edilir.

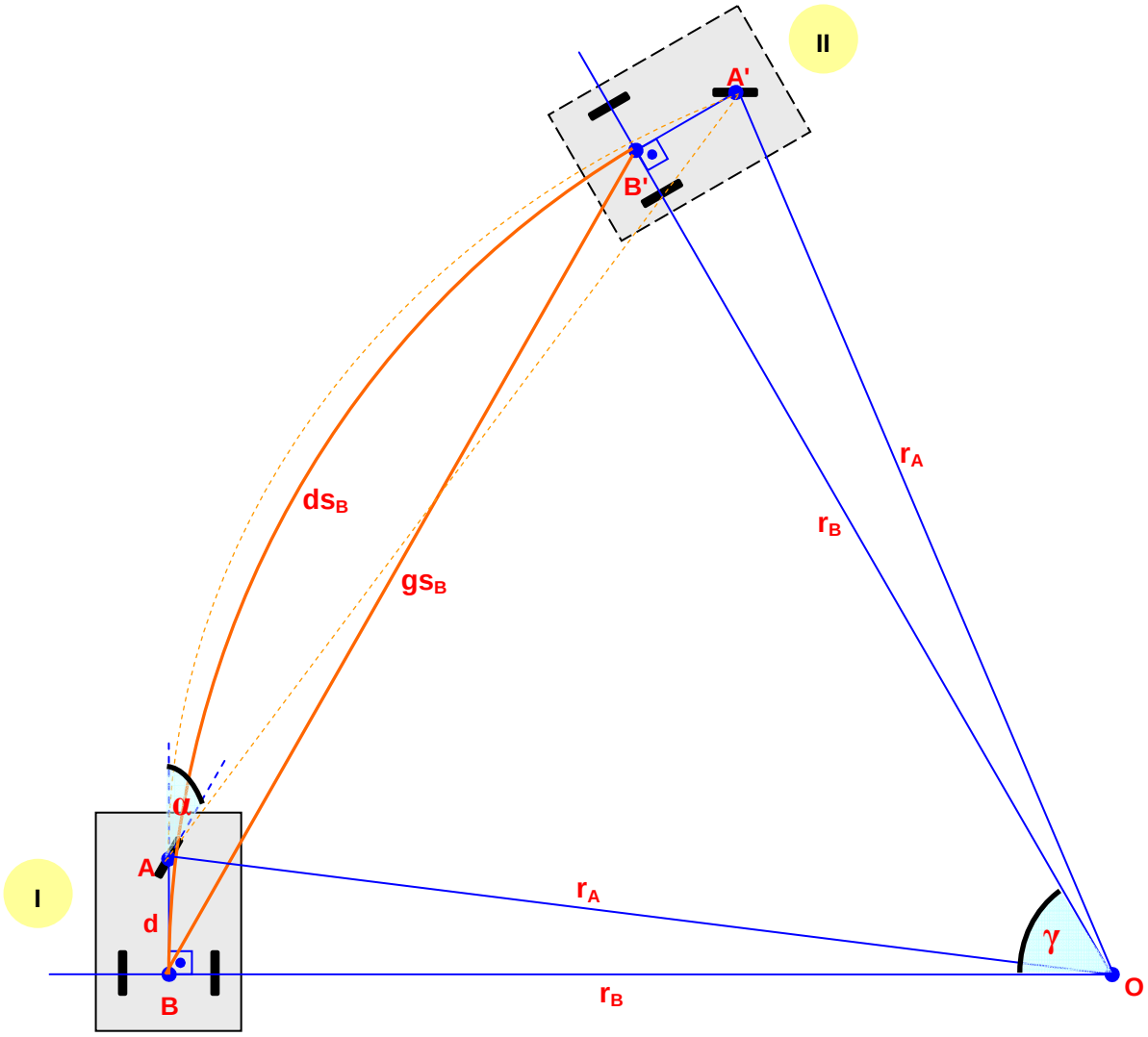
$$\beta = 90 - \frac{\gamma}{2} - \alpha \tag{3.14}$$

Robotun hareketi öncesindeki ön teker orta noktasının konumu $A(x_A,y_A)$, gs_A uzunluğu ve β açısı bilindiğinde robotun hareketi sonrasındaki ön teker orta noktasının konumu olan $A'(x_A',y_A')$, trigonometriden yararlanılarak (3.15) ve (3.16) denklemlerindeki gibi hesaplanabilir.

$$x_A' = x_A + gs_A \times \cos\beta \tag{3.15}$$

$$y_A' = y_A + gs_A \times \sin\beta \tag{3.16}$$

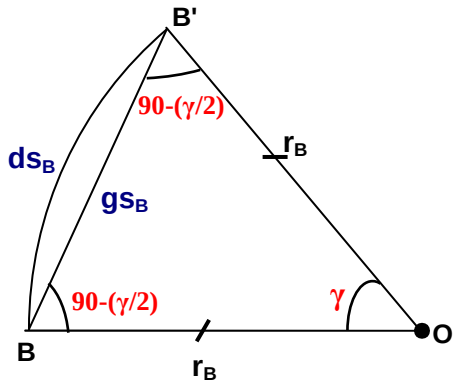
Robotun hareketi ile arka tekerlerinin ortası olan *B* noktası da, *A* noktasına benzer bir biçimde *O* merkezli ve r_B yarıçaplı $B\overset{\frown}{B'}$ yayını çizer (Şekil 3.14). *A* ve *B* noktalarının çizgisel hızları farklı olsa da açısal hızları aynı ve γ'dır.



Şekil 3.14 Robotun arka tekerlerinin orta noktasının, robotun dik konumdan ileri yönlü hareketi ile yer değiştirmesi

r_B değeri, (3.5) denklemi ile hesaplanmıştı.

BOB' üçgeninden yararlanılarak gs_B bulunabilir (Şekil 3.15).



Şekil 3.15 Robotun hareketi öncesi ve sonraki konumlarında arka tekerlerinin orta noktasının geometrik durumu

Kosinüs teoreminden yararlanarak (3.17) yazılabilir ve (3.18) bulunur.

$$gs_B^{\,2} = r_B^{\,2} + r_B^{\,2} - 2 \times r_B^{\,2} \times \cos\gamma \tag{3.17}$$

$$gs_B = \sqrt{2} \times r_B \times \sqrt{1 - \cos\gamma} \tag{3.18}$$

Ayrıca gs_B daha farklı bir yöntemle (3.20)'deki gibi de bulunabilir.

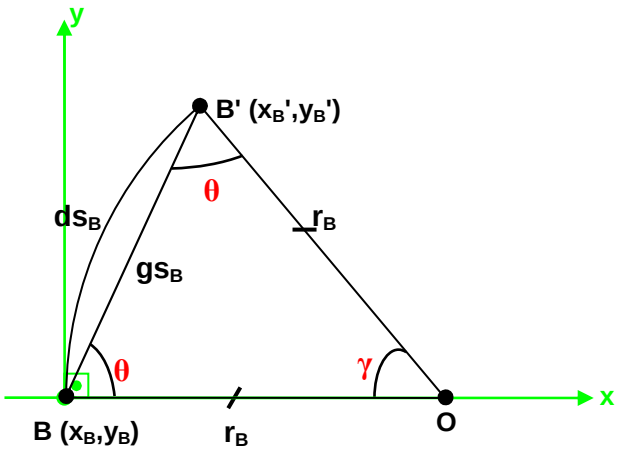
Sinüs teoreminden yararlanarak (3.19) yazılabilir ve (3.20) bulunur.

$$\frac{gs_B}{\sin\gamma} = \frac{r_B}{\sin\left(90 - \frac{\gamma}{2}\right)} \tag{3.19}$$

$$gs_B = \frac{r_B \times \sin\gamma}{\sin\left(90 - \frac{\gamma}{2}\right)} \tag{3.20}$$

BOB' üçgeni ikizkenar üçgendir ve θ değeri (3.21) denklemindeki gibi bulunabilir (Şekil 3.16).

$$\theta = 90 - \frac{\gamma}{2} \tag{3.21}$$



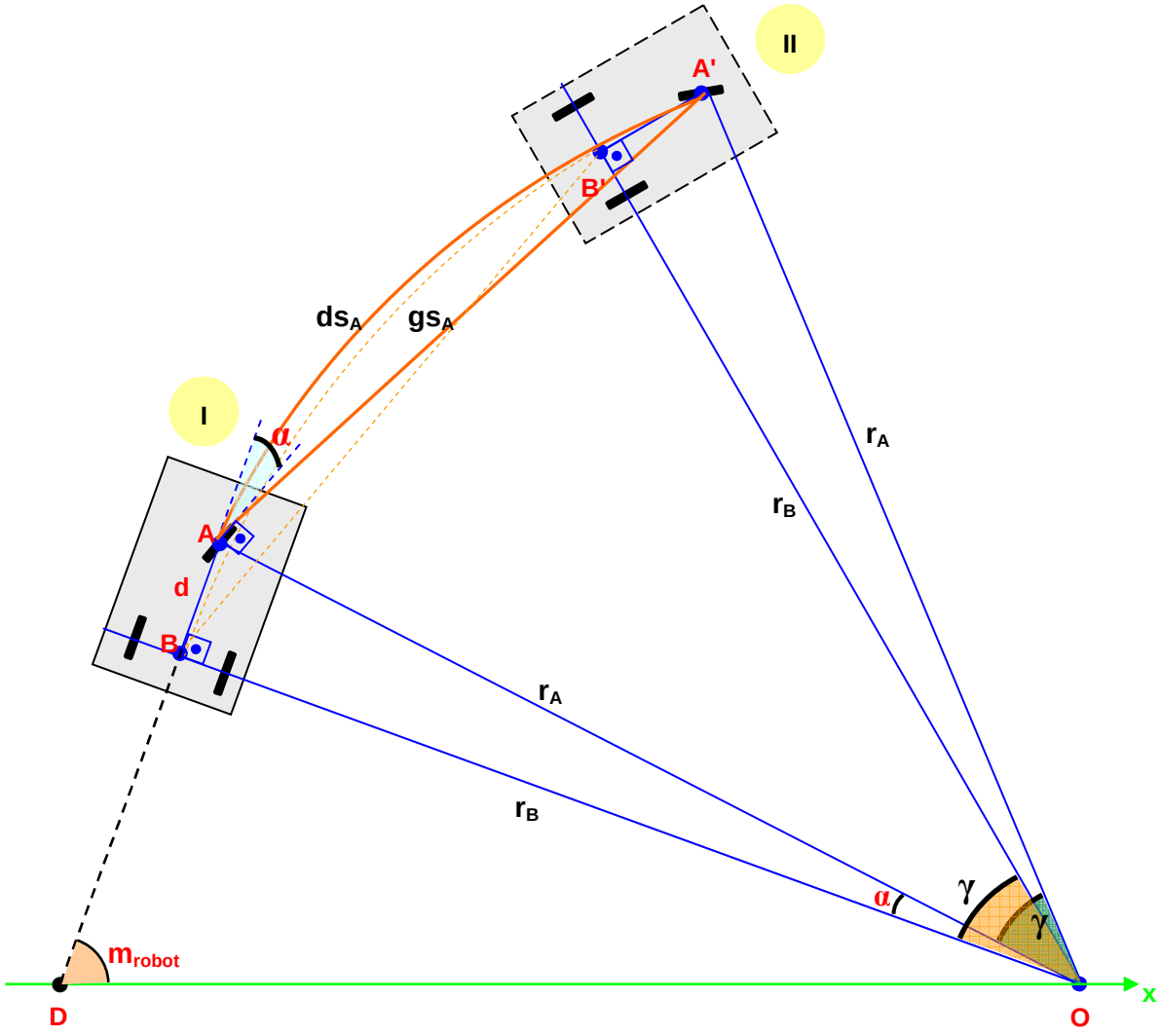
Şekil 3.16 Robotun dik konumdan ileri yönlü hareketi sonrasındaki arka tekerlerinin orta noktasının konumunun bulunması

Robotun hareketi öncesindeki arka tekerlerinin orta noktasının konumu $B(x_B,y_B)$, gs_B uzunluğu ve θ açısı bilindiğinde robotun hareketi sonrasındaki arka tekerlerinin orta noktasının konumu olan $B'(x_B',y_B')$, trigonometriden yararlanılarak (3.22) ve (3.23) denklemlerindeki gibi hesaplanabilir.

$$x_B' = x_B + gs_B \times \cos\theta \tag{3.22}$$

$$y_B' = y_B + gs_B \times \sin\theta \tag{3.23}$$

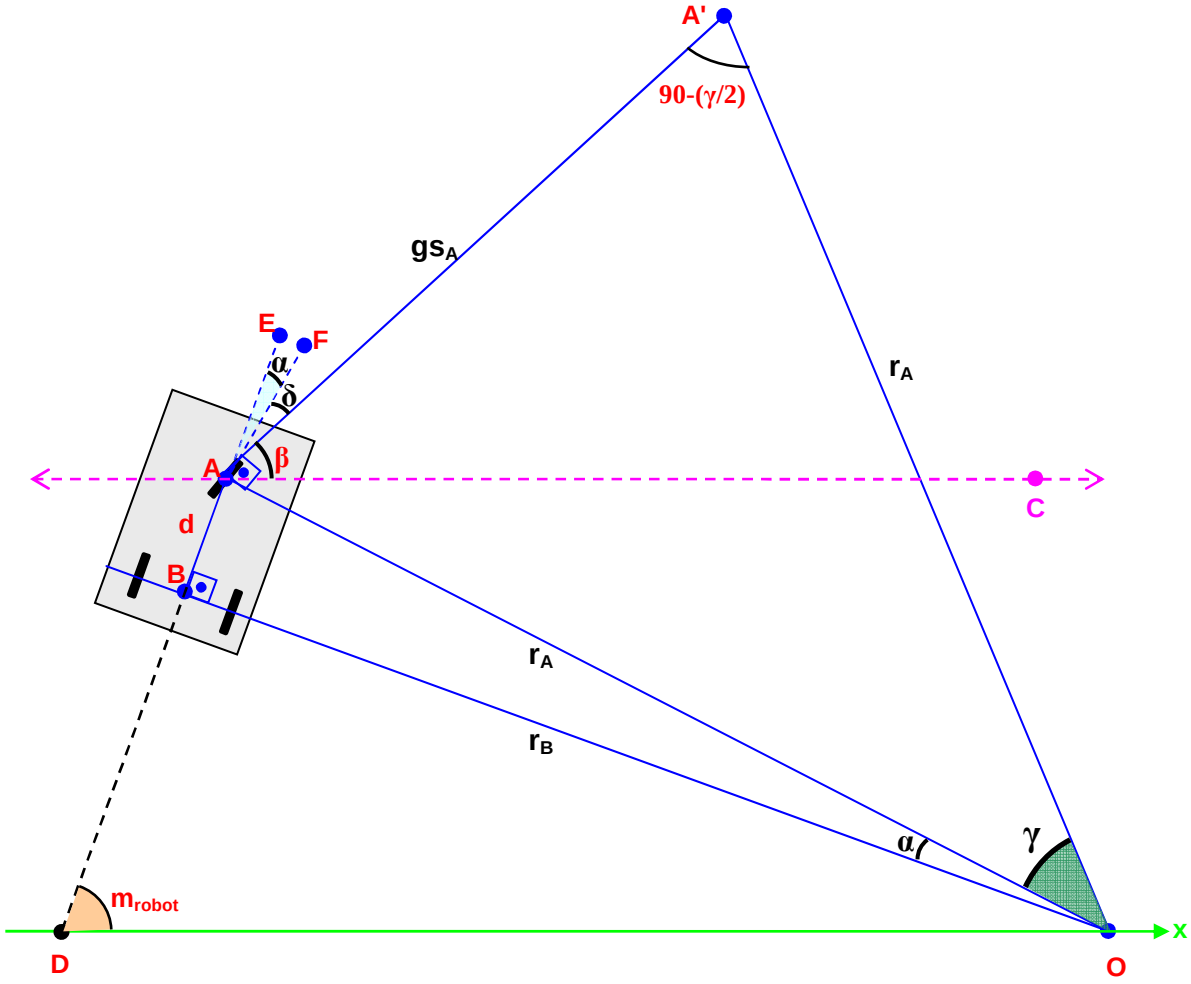
Önceki hesaplamalarda robotun başlangıç konumunda, ordinat ekseninin artım yönünde yani eğimi 90° iken hareketine başladığı varsayılmaktaydı. Eğer robot hareketine farklı bir m_{robot} eğimi ile başlar ise ve bu hareketin denklemeleri bulunursa bunlar robotun kinematiği için daha genel sonuçlar olacaktır (Şekil 3.17).



Şekil 3.17 Gezgin robotun eğimli bir ilk konumdan ileri yönlü hareketi

Gama açısı (3.7)'deki, r_A (3.4)'deki, r_B (3.5)'deki ve gs_A (3.9) veya (3.11)'deki gibi hesaplanabilir.

Şekil 3.18'de görüldüğü gibi apsis eksenine paralel ve *A* noktasından geçecek şekilde bir doğru çizildiğinde



Şekil 3.18 Robotun eğimli bir ilk konumdan ileri yönlü hareketi sonrasındaki ön teker orta noktasının konumunun bulunması

AOA' ikizkenar üçgeninden yararlanılarak (3.24) elde edilebilir.

$$m(O\hat{A}'A) = m(A'\hat{A}O) = 90 - \frac{\gamma}{2} \tag{3.24}$$

(3.25) ve (3.26) şekilde görülebilmektedir.

$$m(F\hat{A}O) = 90 \tag{3.25}$$

$$m(F\hat{A}A') = \delta \tag{3.26}$$

Şekilden (3.27) yazılabilir.

$$m(F\hat{A}O) = \delta + m(A'\hat{A}O) \tag{3.27}$$

(3.28), (3.27)'den elde edilebilir.

$$\delta = m(F\hat{A}O) - m(A'\hat{A}O) \tag{3.28}$$

(3.28)'de (3.24) ve (3.25) yerine yazılırsa (3.29) elde edilir.

$$\delta = 90 - \left(90 - \frac{\gamma}{2}\right) = \frac{\gamma}{2} \tag{3.29}$$

EDO ve *EAC* açıları yöndeş açılar olduğu için değerleri birbirine eşittir (3.30).

$$m(E\hat{D}O) = m(E\hat{A}C) = m_{robot} \tag{3.30}$$

Şekilden (3.31) yazılabilir.

$$m(E\hat{A}C) = \alpha + \delta + \beta = m_{robot} \tag{3.31}$$

(3.32) ise (3.31)'den elde edilir.

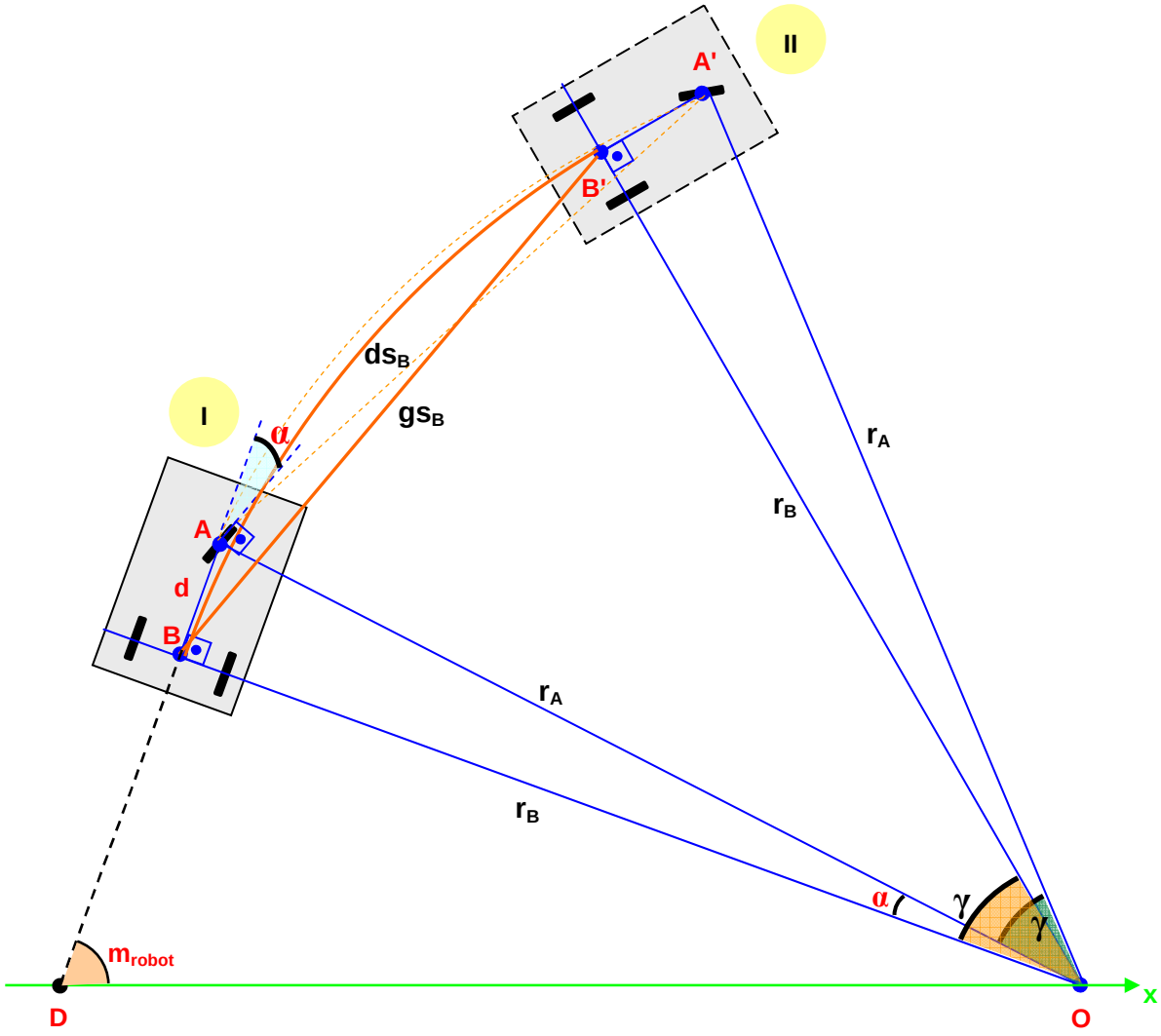
$$\beta = m_{robot} - \alpha - \delta \tag{3.32}$$

(3.32)'de (3.29) yerine yazılırsa (3.33) elde edilir.

$$\beta = m_{robot} - \alpha - \frac{\gamma}{2} \tag{3.33}$$

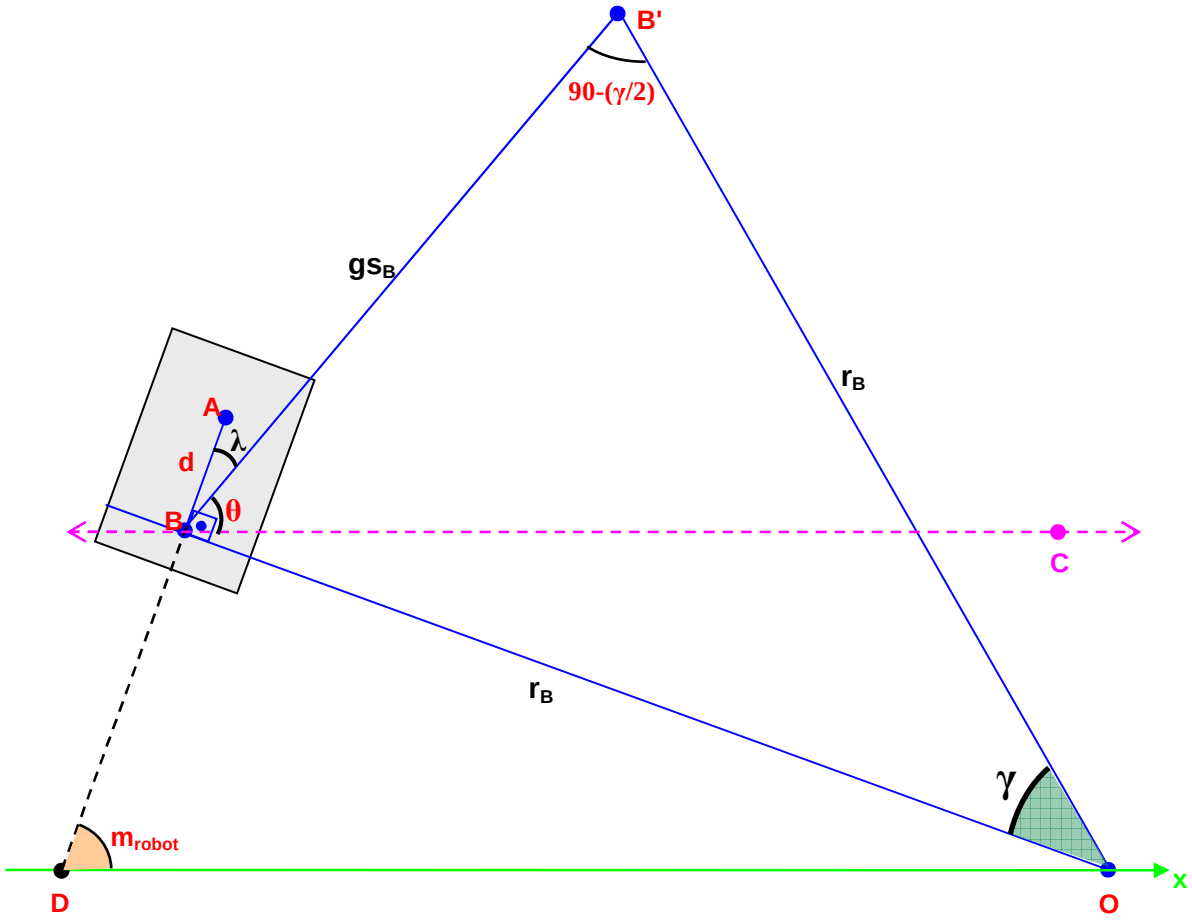
Robotun hareketi öncesindeki ön teker orta noktasının konumu $A(x_A,y_A)$, gs_A uzunluğu ve β açısı bilindiğinde robotun hareketi sonrasındaki ön teker orta noktasının konumu olan $A'(x_A',y_A')$, (3.15) ve (3.16) denklemlerindeki gibi hesaplanabilir.

B noktası için daha önce yapılan hesaplamalar tekrarlanırsa r_B (3.5)'deki, gs_B ise (3.18) veya (3.20)'deki gibi hesaplanabilir (Şekil 3.19).



Şekil 3.19 Robotun arka tekerlerinin orta noktasının eğimli bir ilk konumundan hareketle yer değiştirmesi

Şekil 3.20'de görüldüğü gibi apsis eksenine paralel ve *B* noktasından geçecek şekilde bir doğru çizilmiş olsun.



Şekil 3.20 Robotun eğimli bir ilk konumdan ileri yönlü hareketi sonrasındaki arka tekerlerinin orta noktasının konumunun bulunması

BOB' ikizkenar üçgeninden yararlanılarak (3.34) elde edilebilir.

$$m(O\hat{B}'B) = m(B'\hat{B}O) = 90 - \frac{\gamma}{2} \tag{3.34}$$

(3.35) ve (3.36) şekilde görülebilmektedir.

$$m(A\hat{B}O) = 90 \tag{3.35}$$

$$m(A\hat{B}B') = \lambda \tag{3.36}$$

Şekilden (3.37) yazılabilir.

$$m(A\hat{B}O) = \lambda + m(B'\hat{B}O) \tag{3.37}$$

(3.37)'den (3.38) elde edilir.

$$\lambda = m(A\hat{B}O) - m(B'\hat{B}O) \tag{3.38}$$

(3.38)'de (3.34) ve (3.35) yerine yazılırsa (3.39) elde edilir.

$$\lambda = 90 - \left(90 - \frac{\gamma}{2}\right) = \frac{\gamma}{2} \tag{3.39}$$

ADO ve *ABC* açıları yöndeş açılar olduğu için değerleri birbirine eşittir (3.40).

$$m(A\hat{D}O) = m(A\hat{B}C) = m_{robot} \tag{3.40}$$

(3.41) şekilden yazılabilir.

$$m(A\hat{B}C) = \lambda + \theta = m_{robot} \tag{3.41}$$

(3.41)'den (3.42) elde edilir.

$$\theta = m_{robot} - \lambda \tag{3.42}$$

(3.42)'de (3.39) yerine yazılırsa (3.43) elde edilir.

$$\theta = m_{robot} - \frac{\gamma}{2} \tag{3.43}$$

Robotun hareketi öncesindeki arka tekerlerinin orta noktasının konumu $B(x_B,y_B)$, gs_B uzunluğu ve θ açısı bilindiğinde robotun hareketi sonrasındaki arka tekerlerinin orta noktasının konumu olan $B'(x_B',y_B')$, (3.22) ve (3.23) denklemlerindeki gibi hesaplanabilir.

Önceki hesaplamaların tümünde robotun ileri yönlü hareketini sağa doğru dönerek yaptığı varsayılmıştı. Eğer, robot hareketini sola doğru dönerek yapacak olursa hesaplamalar değişecektir. Şu ana kadar bulunan denklemler, aşağıdaki gibi genelleştirilerek robotun hem sola hem de sağa doğru dönüşü için kullanılabilir.

Robot dönme hareketini sola doğru yaptığında alfa açısı, $\left[\frac{3}{2}\pi,2\pi\right)$ aralığında değerler alabilecektir. Bu durumda, önceki hesaplamalardaki denklemler kullanıldığında gs_A ve gs_B değerleri negatif olarak hesaplanacaktır. Hesaplamalarda mutlak değerler alınarak (3.44), (3.45), (3.46) ve (3.47) denklemlerindeki gibi yazılırlarsa, gs_A ve gs_B değerlerinin hesabında iki farklı durum içinde kullanılabilecek daha genel ifadeler elde edilmiş olacaktır.

$$gs_A = \left|\sqrt{2}\times r_A \times \sqrt{1-\cos\gamma}\right| \tag{3.44}$$

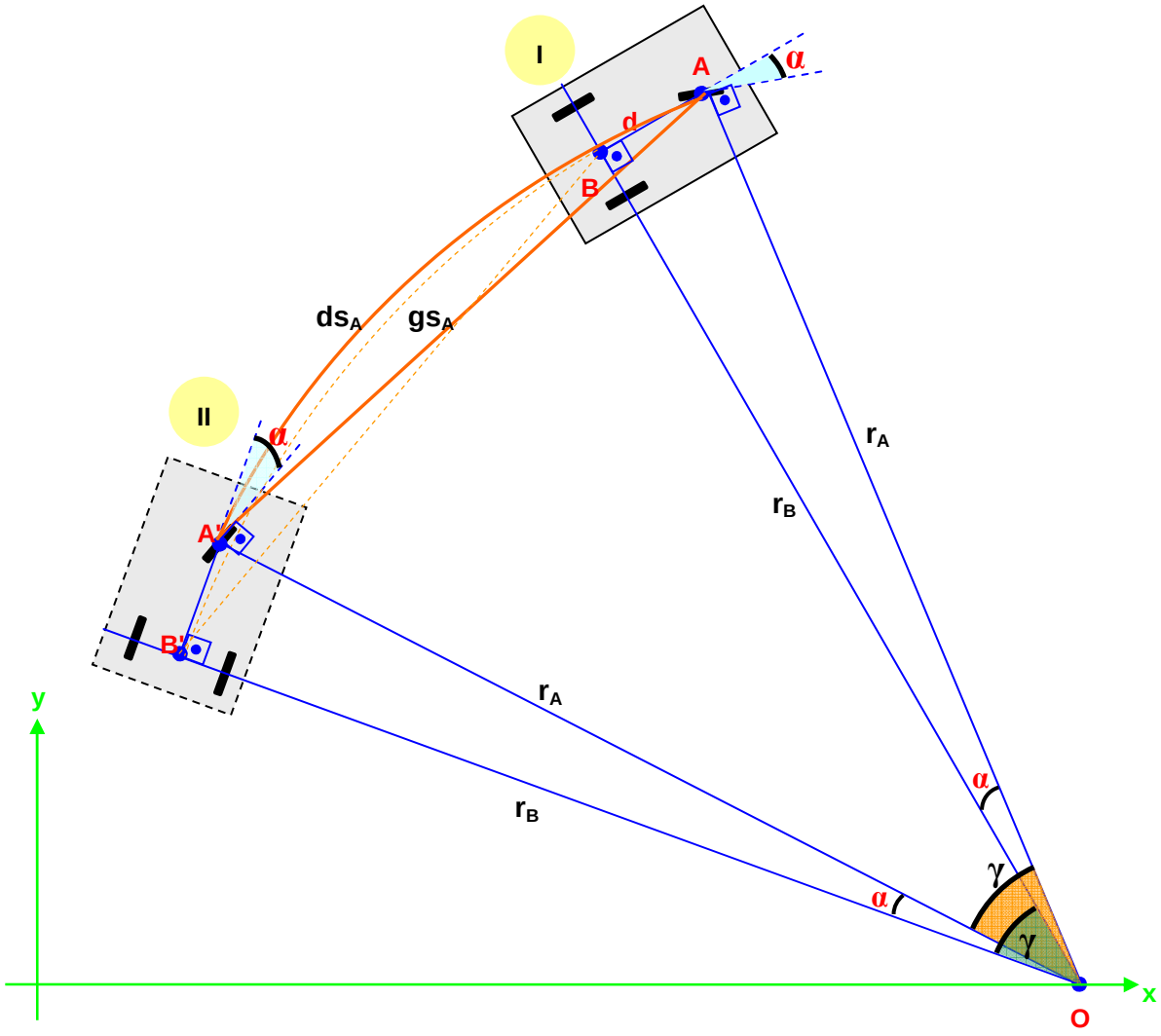
$$gs_A = \left|\frac{r_A \times \sin\gamma}{\sin\left(90-\frac{\gamma}{2}\right)}\right| \tag{3.45}$$

$$gs_B = \left|\sqrt{2}\times r_B \times \sqrt{1-\cos\gamma}\right| \tag{3.46}$$

$$gs_B = \left|\frac{r_B \times \sin\gamma}{\sin\left(90-\frac{\gamma}{2}\right)}\right| \tag{3.47}$$

Denklemlerin bu şekilde kullanılmasıyla, $A'(x_A',y_A')$ noktası (3.33), (3.44), (3.15) ve (3.16) denklemleri kullanılarak, $B'(x_B',y_B')$ noktası ise (3.43), (3.46), (3.22) ve (3.23) denklemleri ile hesaplanabilir.

Gezgin robot ayrıca bazı koşullarda geri yönlü harekette yapmaktadır. Bu durumda, robotun ileri yönlü hareketinde kullanılan kinematik denklemelere benzer hesaplamalar yapılarak robotun pozisyonu bulunabilmektedir.

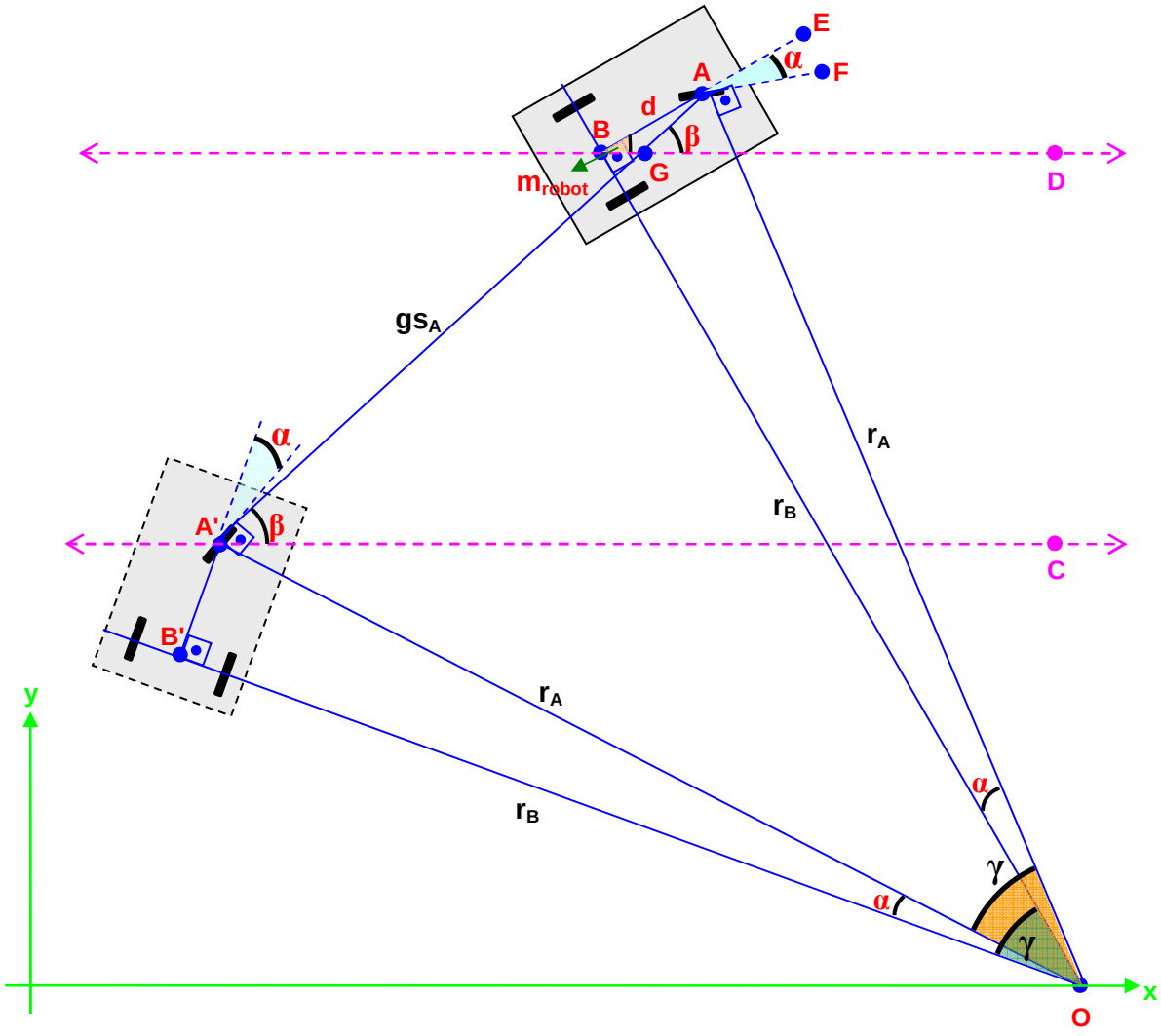


Şekil 3.21 Gezgin robotun geri yönlü hareketi

Yapılan hesaplamalarda, gezgin robotun I konumundan II konumuna hareket ettiği varsayılmaktadır (Şekil 3.21). Simgeler yine robotun ileri gitmesi durumundaki gibi kullanılarak hesaplamalar yapılacaktır. Robotun ön tekeri, ileri yönlü harekette olduğu gibi ds_A yayını çizecektir ve açısal hız γ'dır.

Gama açısı (3.7), r_A (3.4), r_B (3.5), gs_A (3.44) veya (3.45) ve gs_B (3.46) veya (3.47)'deki gibi hesaplanabilir.

Apsis eksenine paralel olacak ve *A'* ve *B* noktalarından geçecek şekilde iki doğru çizildiğinde *AA'C* ve *AGD* açılarının yöndeş açılar olduğu görülmektedir (Şekil 3.22). Bu açıların değerlerinin β olduğu kabul edilmiştir.



Şekil 3.22 Robotun geri yönlü hareketi sonrasındaki ön teker orta noktasının konumunun bulunması

ABD açısının değeri robotun eğimidir ve (3.48) elde edilebilir.

$$m(A\hat{B}D) = m_{robot} \tag{3.48}$$

AOA' ikizkenar üçgeninden yararlanılarak (3.49) elde edilebilir.

$$m(O\hat{A}'A) = m(G\hat{A}O) = 90 - \frac{\gamma}{2} \tag{3.49}$$

Doğrusal açıların toplamının 180 olmasından ve *BE* doğrusundan yararlanılarak (3.50)

bulunabilir.

$$m(B\hat{A}O) = 90 - \alpha \tag{3.50}$$

Şekilden (3.51) yazılabilir.

$$m(B\hat{A}O) = m(B\hat{A}G) + m(G\hat{A}O) \tag{3.51}$$

(3.51)'den (3.52) elde edilir.

$$m(B\hat{A}G) = m(B\hat{A}O) - m(G\hat{A}O) \tag{3.52}$$

(3.52)'de (3.49) ve (3.50) yerine yazılırsa (3.53) elde edilir.

$$m(B\hat{A}G) = (90 - \alpha) - \left(90 - \frac{\gamma}{2}\right) = -\alpha + \frac{\gamma}{2} \tag{3.53}$$

Üçgen özelliklerinden (3.54) yazılabilir.

$$m(A\hat{G}D) = m(A\hat{B}D) + m(B\hat{A}G) \tag{3.54}$$

(3.54)'de (3.48) ve (3.53) yerine yazıldığında (3.55) elde edilir.

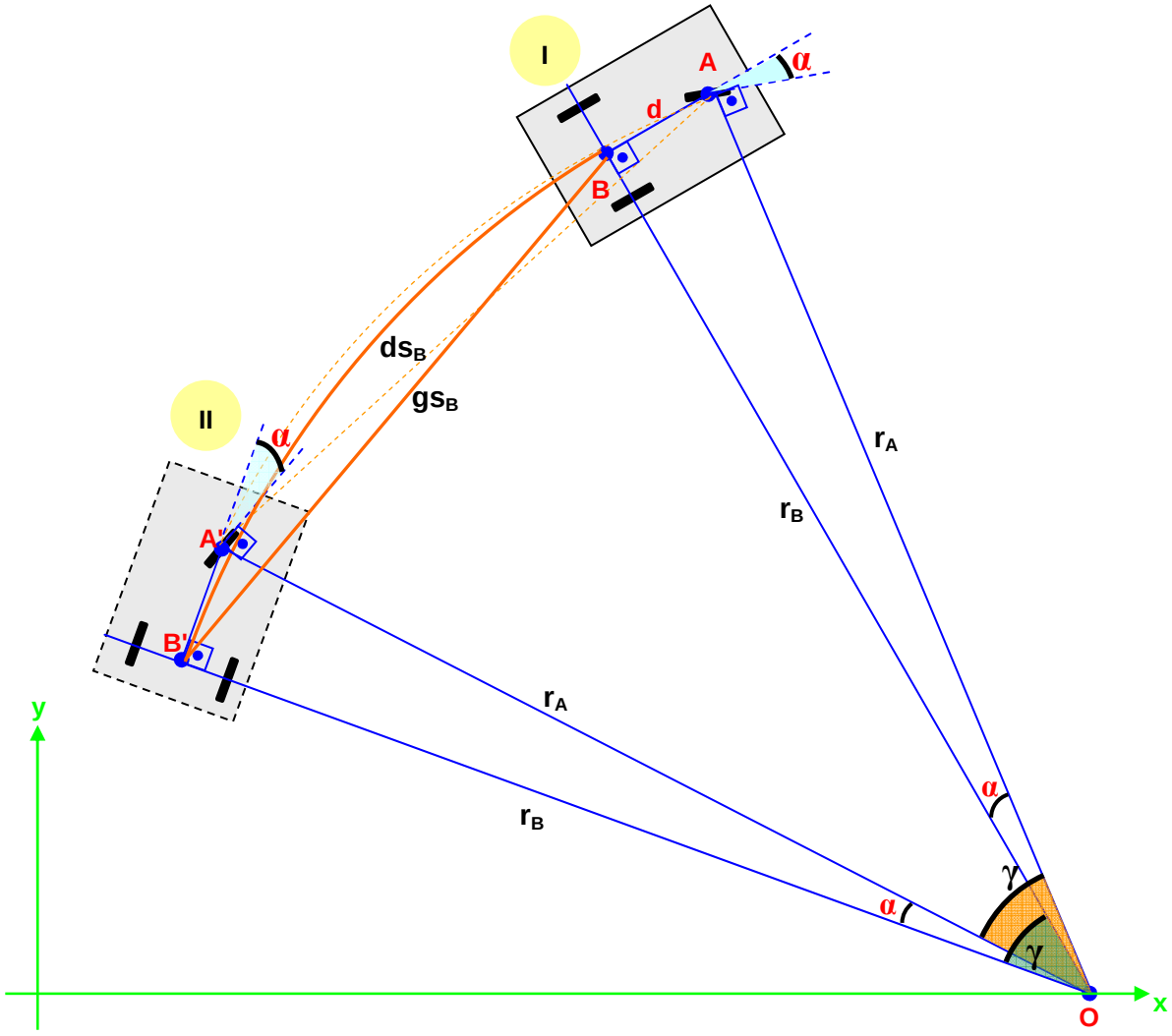
$$m(A\hat{G}D) = \beta = m_{robot} - \alpha + \frac{\gamma}{2} \tag{3.55}$$

Robotun hareketi öncesindeki ön teker orta noktasının konumu $A(x_A,y_A)$, gs_A uzunluğu ve β açısı bilindiğinde robotun hareketi sonrasındaki ön teker orta noktasının konumu olan $A'(x_A',y_A')$, trigonometriden yararlanılarak (3.56) ve (3.57) denklemlerindeki gibi hesaplanabilir.

$$x_A' = x_A - gs_A \times \cos\beta \tag{3.56}$$

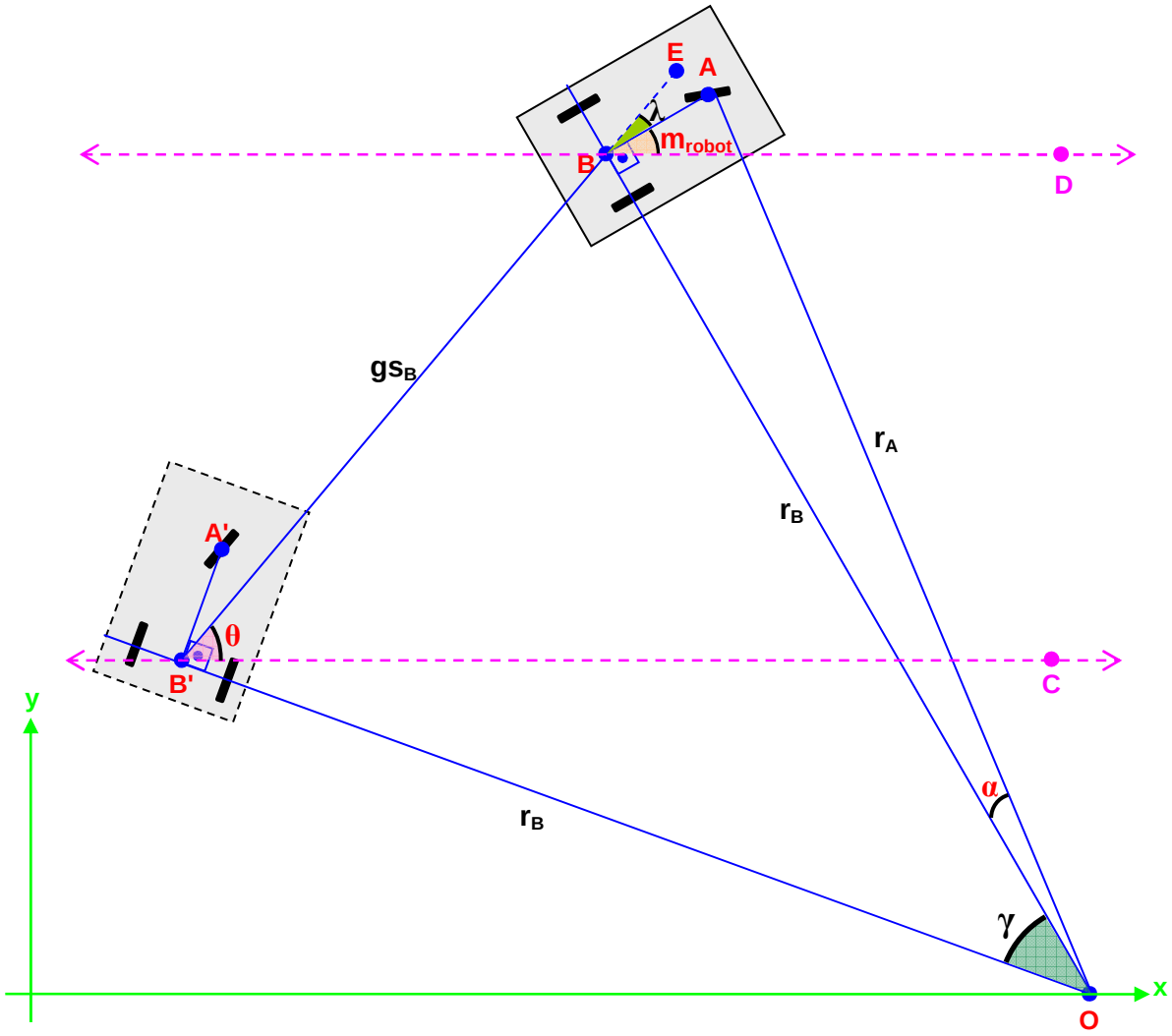
$$y_A' = y_A - gs_A \times \sin\beta \tag{3.57}$$

Robotun geri yönlü hareketinde *B* noktası da *A* noktasına benzer biçimde ds_B yayını çizmektedir (Şekil 3.23). Bu yayı gören kirişte gs_B'dir ve açısal hız γ'dır.



Şekil 3.23 Robotun geri yönlü hareketi ile arka tekerlerinin orta noktasının yer değiştirmesi

Apsis eksenine paralel olacak şekilde *B′* ve *B* noktalarından geçecek iki doğru çizildiğinde *EB′C* ve *EBD* açılarının yöndeş açılar olduğu görülmektedir (Şekil 3.24). Bu açıların değerleri *θ*'dır.



Şekil 3.24 Robotun geri yönlü hareketi sonrasındaki arka tekerlerinin orta noktasının konumunun bulunması

ABD açısının değeri robotun eğimidir (3.58).

$$m(A\hat{B}D) = m_{robot} \tag{3.58}$$

BOB′ ikizkenar üçgeninden yararlanılarak (3.59) elde edilebilir.

$$m(O\hat{B}'B) = m(B'\hat{B}O) = 90 - \frac{\gamma}{2} \tag{3.59}$$

Şekilden (3.60) yazılabilir.

$$m(E\hat{B}A) + m(A\hat{B}O) + m(B'\hat{B}O) = 180 \tag{3.60}$$

(3.60)'dan (3.61) elde edilir.

$$m(E\hat{B}A) = 180 - m(A\hat{B}O) - m(B'\hat{B}O) \tag{3.61}$$

ABO açısının değeri 90'dır ve (3.59) ile birlikte (3.61)'de yerine yazılırsa (3.62) elde edilir.

$$m(E\hat{B}A) = \lambda = 180 - 90 - \left(90 - \frac{\gamma}{2}\right) = \frac{\gamma}{2} \tag{3.62}$$

Şekilden (3.63) yazılabilir.

$$m(E\hat{B}D) = m(A\hat{B}D) + m(E\hat{B}A) \tag{3.63}$$

(3.63)'de (3.58) ve (3.62) yerine yazılırsa θ açısı (3.64) deki gibi bulunur.

$$\theta = m_{robot} + \frac{\gamma}{2} \tag{3.64}$$

Robotun hareketi öncesindeki arka tekerlerinin orta noktasının konumu $B(x_B,y_B)$, gs_B uzunluğu ve θ açısı bilindiğinde robotun hareketi sonrasındaki arka tekerlerinin orta noktasının konumu olan $B'(x_B',y_B')$, trigonometriden yararlanılarak (3.65) ve (3.66) denklemlerindeki gibi hesaplanabilir.

$$x_B' = x_B - gs_B \times \cos\theta \tag{3.65}$$

$$y_B' = y_B - gs_B \times \sin\theta \tag{3.66}$$

Tüm bu denklemleri kullanarak genel bir sonuç çıkarılabilir. Gezgin robotun başlangıç konumu ($A(x_A,y_A)$ ve $B(x_B,y_B)$ noktaları), robotun başlangıç konumundaki eğimi olan m_{robot} değeri, robotun ön teker orta noktası ve arka tekerlerinin orta noktası arasındaki *d* mesafesi, robotun ön tekerinin dönme miktarı olan alfa açısı ve robotun hareketi esnasında ön tekerinin almış olduğu ds_A mesafesi bilindiğinde robotun hareketi sonrasındaki son konumunun ($A'(x_A',y_A')$ ve $B'(x_B',y_B')$ noktaları) bulunabilmesi için kullanılabilecek yöntemin sözde kodu şu şekilde olacaktır:

▪ eğer <*robot düz gidiyorsa*> ($\alpha = 0°$)

$\gamma = 0$, $gs_A = ds_A$, $gs_B = ds_A$

▪ eğer <*robot dönüyorsa*> ($\alpha \neq 0°$)

$$r_A = \frac{d}{\sin\alpha} \text{ , } r_B = \frac{d}{\tan\alpha} \text{ , } \gamma = \frac{ds_A}{2\pi r_A} \times 360$$

$$gs_A = \left|\sqrt{2} \times r_A \times \sqrt{1-\cos\gamma}\right| \text{ veya } gs_A = \left|\frac{r_A \times \sin\gamma}{\sin\left(90 - \frac{\gamma}{2}\right)}\right|$$

$$gs_B = \left|\sqrt{2} \times r_B \times \sqrt{1-\cos\gamma}\right| \text{ veya } gs_B = \left|\frac{r_B \times \sin\gamma}{\sin\left(90 - \frac{\gamma}{2}\right)}\right|$$

▪ eğer <*robot ileri doğru gidiyorsa*>

$$\beta = m_{robot} - \alpha - \frac{\gamma}{2}$$

$$x_A' = x_A + gs_A \times \cos\beta$$

$$y_A' = y_A + gs_A \times \sin\beta$$

$$\theta = m_{robot} - \frac{\gamma}{2}$$

$$x_B' = x_B + gs_B \times \cos\theta$$

$$y_B' = y_B + gs_B \times \sin\theta$$

▪ eğer <*robot geri doğru gidiyorsa*>

$$\beta = m_{robot} - \alpha + \frac{\gamma}{2}$$

$$x_A' = x_A - gs_A \times \cos\beta$$

$$y_A' = y_A - gs_A \times \sin\beta$$

$$\theta = m_{robot} + \frac{\gamma}{2}$$

$$x_B' = x_B - gs_B \times \cos\theta$$

$$y_B' = y_B - gs_B \times \sin\theta$$

4. SİMÜLASYON ORTAMI

Tasarlanan simülasyon ortamı, NetBeans IDE 6.5 [4] kullanılarak Java programlama dilinde geliştirilmiştir. Java'nın kullanılmış olmasının sebebi nesne yönelimli olması, dilin sağladığı kolaylıklar, geniş kütüphane seçeneği ve taşınabilirlik sayesinde platformdan bağımsızlık olmuştur (Schildt, 2003; Flanagan, 2004; [3]). Uygulamaya, kullanıcıya sağlamış olduğu kolaylıklar göz önünde tutularak ERSIM (*Easy Robot SIMulation*) ismi verilmiştir.

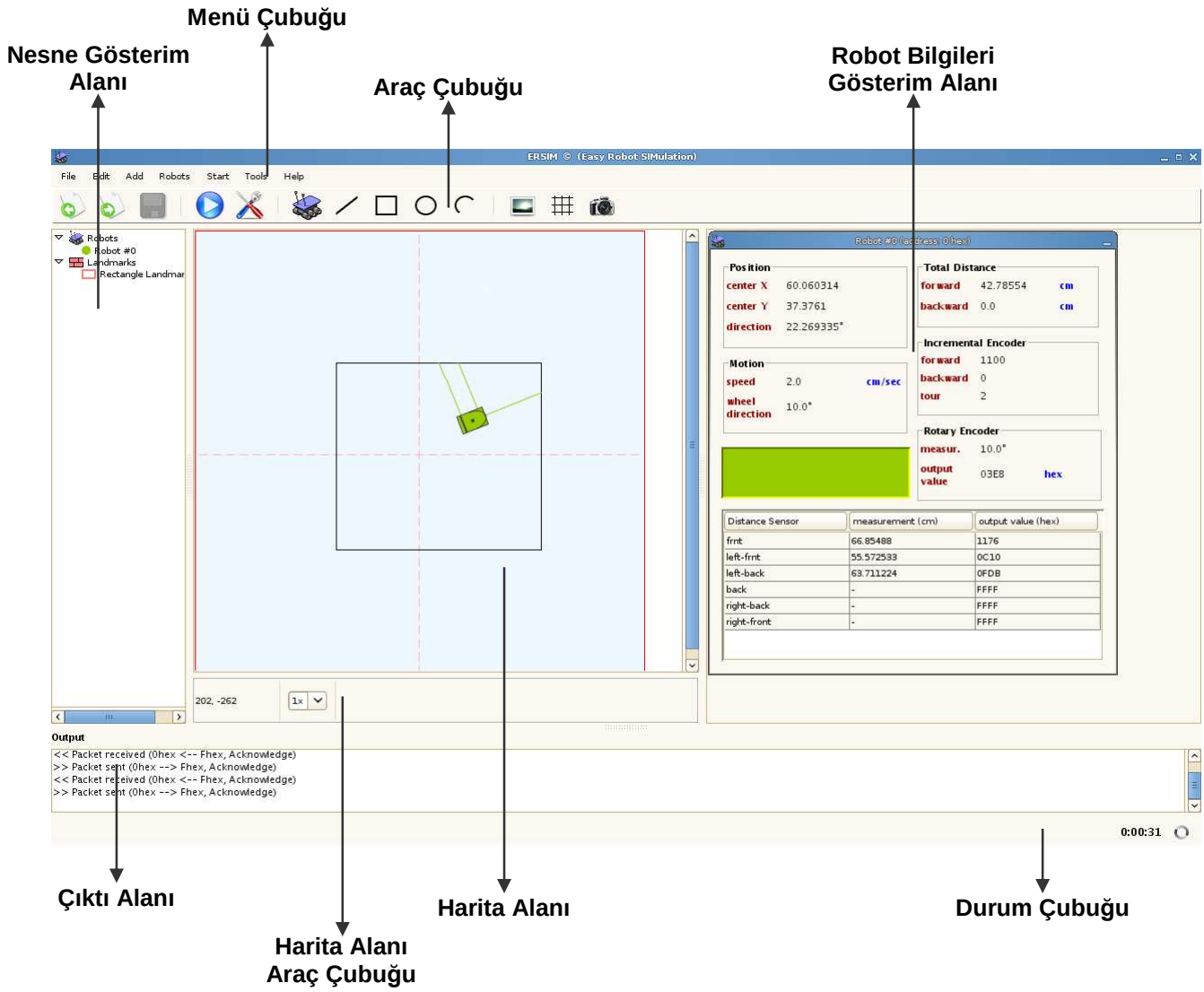
Simülasyon, robotu birebir modellediği için uygulamanın kontrolü amaçlı gerçekleştirilen haberleşmede Bölüm 3.3'de açıklanan protokol yapısı kullanılmaktadır. İletişim, seri port üzerinden gerçekleştirilmekte ve simülasyonun kontrolü bu şekilde sağlanmaktadır. Seri port erişiminde, GNU LGPL lisansı altında dağıtımı yapılan RXTX kütüphanesi kullanılmaktadır [5]. Seri port erişimi, yapısı gereği işletim sistemi bağımlı bir işlem olduğu için RXTX kütüphanesi seri port erişimi için her işletim sisteminde, o sistem için özel hazırlanmış dinamik erişimli bir kütüphaneye çalışma anında ihtiyaç duymaktadır. Farklı dosya türlerinde olabilen (so, dll veya jnilib) bu kütüphane, uygulama ilk çalıştırıldığında kontrol edilmekte bulunamaması halinde ise işletim sistemine göre gerekli dosya JRE'nin kullanabileceği uygun konuma yerleştirilmektedir. Böylece JRE'nin yüklü olduğu ve seri portu bulunan herhangi bir bilgisayarda uygulamanın çalıştırılması mümkün olmaktadır.

4.1 Simülasyon Arayüzü

Simülasyonun mümkün olduğunca kolay kullanımlı ve anlaşılır bir yapıda olmasına gayret gösterilmiştir. Uygulama arayüzüde buna paralel olarak tasarlanmıştır (Şekil 4.1).

Arayüzdeki temel bölümler ve fonksiyonları şu şekildedir:

- **Menü Çubuğu:** Simülasyonun tüm özelliklerine erişebilmek için gerekli olan menülerin yer aldığı bölümdür. File (*Dosya*) menüsü ile harita kaydedip açma, konfigürasyon ayarlarını kaydetme ve haritanın anlık görüntüsünü alma işlemleri; Edit (*Düzen*) menüsü ile engel ve robot nesnelerini kesme, kopyalama, yapıştırma ve silme işlemleri; Add (*Ekle*) menüsü ile yeni engel veya robot oluşturma işlemleri; Robots (*Robotlar*) menüsü ile robot aygıtları oluşturma, düzenleme, silme işlemleri ve bu oluşturulan aygıtları kullanarak yeni robotlar tanımlayabilme işlemi; Start (*Başlat*) menüsü ile simülasyonun parametre ayarlama ve simülasyonu çalıştırıp durdurma işlemleri ve Tools (*Araçlar*) menüsü ile de harita alanı üzerinde değişiklik yapma işlemleri gerçekleştirilebilmektedir.



Şekil 4.1 Simülasyon uygulaması arayüzü

- **Araç Çubuğu:** Bu alanda, menü seçeneklerine daha hızlı erişimi sağlayan kısayol tuşları bulunmaktadır. Harita kaydedip açma, konfigürasyon ayarlarını kaydetme, simülasyonu başlatıp sonlandırma, simülasyon parametrelerini ayarlama, robot veya engel ekleme, harita alanı üzerinde değişiklik yapma ve anlık görüntü alma işlemlerini yapacak kısayol tuşları bulunmaktadır.
- **Nesne Gösterim Alanı:** Harita alanına eklenilen robotların, engellerin ve engel tiplerinin ağaç yapısı biçiminde görülebileceği bölümdür.
- **Robot Bilgileri Gösterim Alanı:** Çalışma anında, ortamda bulunan tüm robotların kendi durumlarının ayrı bir iç pencerede (*internal frame*) gösterildiği bölümdür.
- **Çıktı Alanı:** Uygulamanın gerçekleştirdiği işlemlerin ve yapılan iletişim bilgilerinin anlık olarak görülebildiği bölümdür.

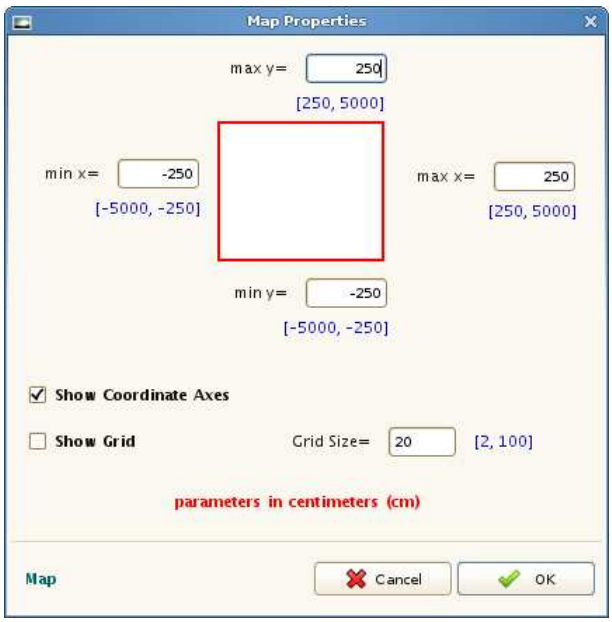
- **Harita Alanı:** Boyutları ayarlanabilen, üzerine çeşitli tiplerde (doğru, dikdörtgen, elips veya yay) engel ve farklı tiplerde ve/veya sayıda robot eklenebilen, 2 boyutlu görsellik sağlayan bölümdür.
- **Harita Alanı Araç Çubuğu:** Harita alanı üzerinde iken fare konumunun gösterildiği, harita alanına yaklaşma veya uzaklaşmayı sağlayan nesne kutusunun bulunduğu ve seçili olan nesneye ait tanım bilgilerinin gösterildiği bölümdür.
- **Durum Çubuğu:** Simülasyonun ilk ayarlarının yapılıp çalıştırıldığı andan itibaren gösterilmeye başlanan ve simülasyonun çalıştığı süreyi gösteren zaman sayacının bulunduğu bölümdür.

4.2 Simülasyon Ölçü Birimleri

Simülasyonda, görsellik iki boyutta ve yeryüzüne dik açılı olarak sunulmaktadır. Gösterim ve çalışma uzayı kartezyen koordinat sistemindedir. Uzunluk ölçü birimi olarak santimetre (*cm*) ve açı ölçü birimi olarak derece (°) kullanılmaktadır.

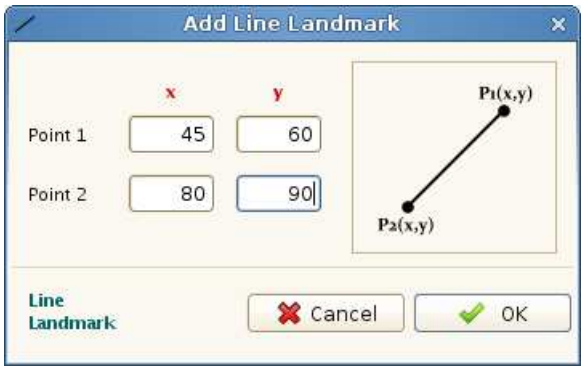
4.3 Simülasyon Harita Alanı

Engellerin yer aldığı ve robotların çalıştığı harita alanı, simülasyon arayüzünün orta kısmında gösterilmektedir. Harita alanının boyutları ve görünümü, Map Properties (*Harita Özellikleri*) penceresinden değiştirilebilmektedir (Şekil 4.2).

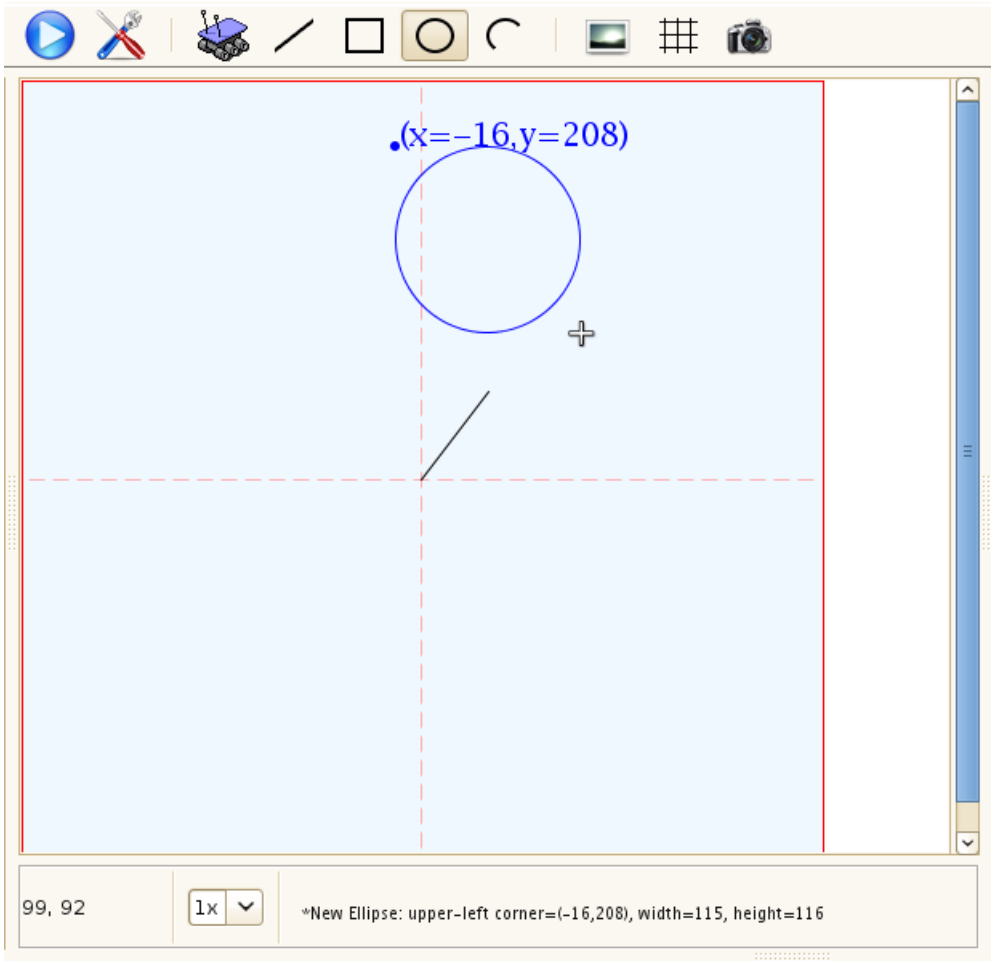


Şekil 4.2 Harita özellikleri penceresi

Harita alanında doğru, dikdörtgen, elips veya yay biçimli engeller (*duvarlar*) oluşturma imkânı bulunmaktadır. Oluşturulan engellerin kalınlıkları sıfır olarak kabul edilmektedir. Haritada yeni engel oluşturmak için, Add (*Ekle*) → Landmark (*Engel*) → <engel tipi> seçilerek açılan pencerede engelin boyutları belirtilebilir (Şekil 4.3) veya araç çubuğundan engel tipi seçildikten sonra fare tuşları kullanılarak harita alanına çizilebilir (Şekil 4.4).

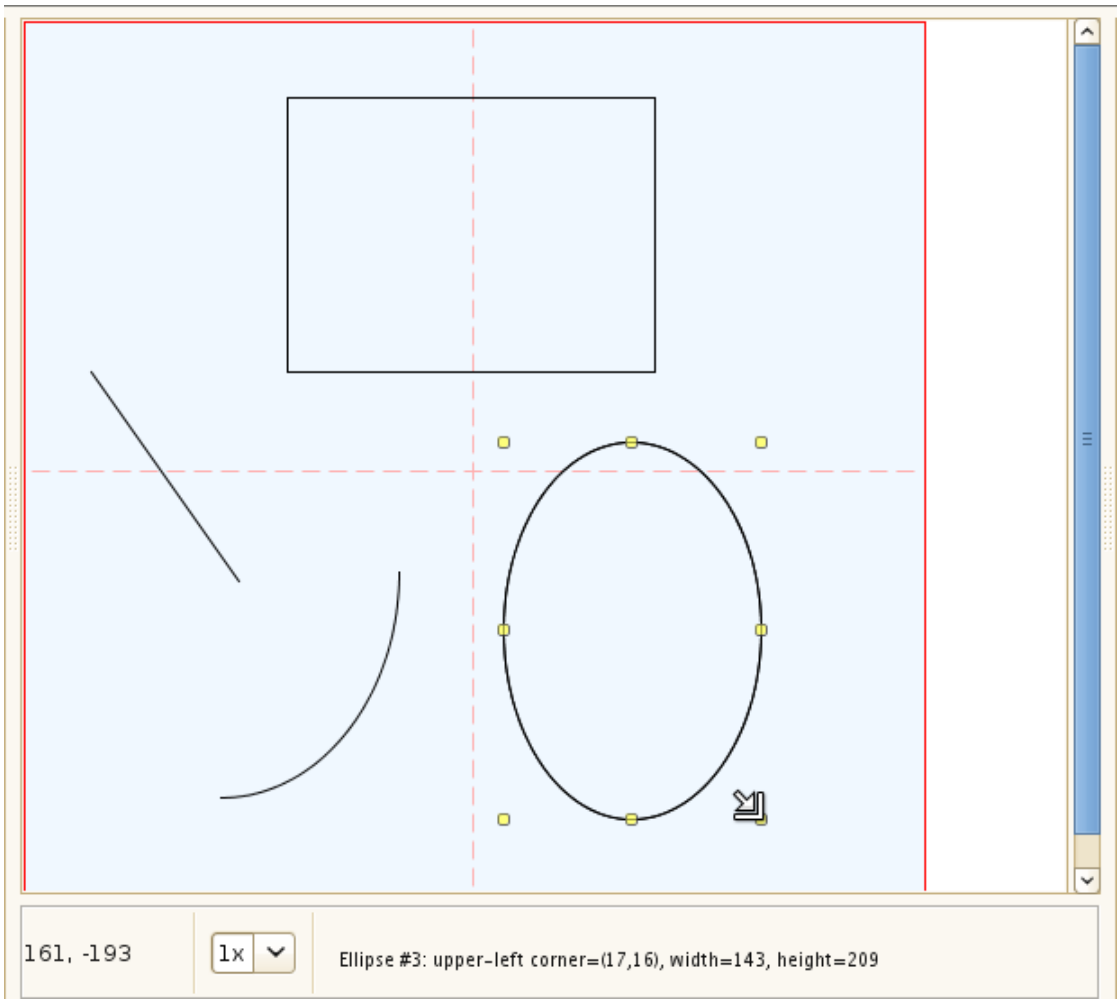


Şekil 4.3 Doğru biçimli engel ekleme penceresi



Şekil 4.4 Elips biçimli engelin fare ile çizimi

Simülasyonda doğru biçimli engel başlangıç ve bitiş noktası ile, dikdörtgen ve elips biçimli engeller sol-üst köşelerinin konumu, en ve boy uzunlukları ile ve yay biçimli engel ise sol-üst köşesinin konumu, en ve boy uzunluğu, başlangıç noktasının ve yayın genişliğinin açı cinsinden değeri ile ifade edilmektedir. Engeller oluşturulduktan sonra seçilebilmekte, üzerine çift tıklanarak açılan özellik penceresinden, köşelerinde oluşan boyutlandırma işaretçileri fare ile kontrol edilerek veya klavyenin yön tuşları kullanılarak konumları veya boyutları değiştirilebilmektedir (Şekil 4.5).



Şekil 4.5 Oluşturulan engelin seçildikten sonra fare ile yeniden boyutlandırılması

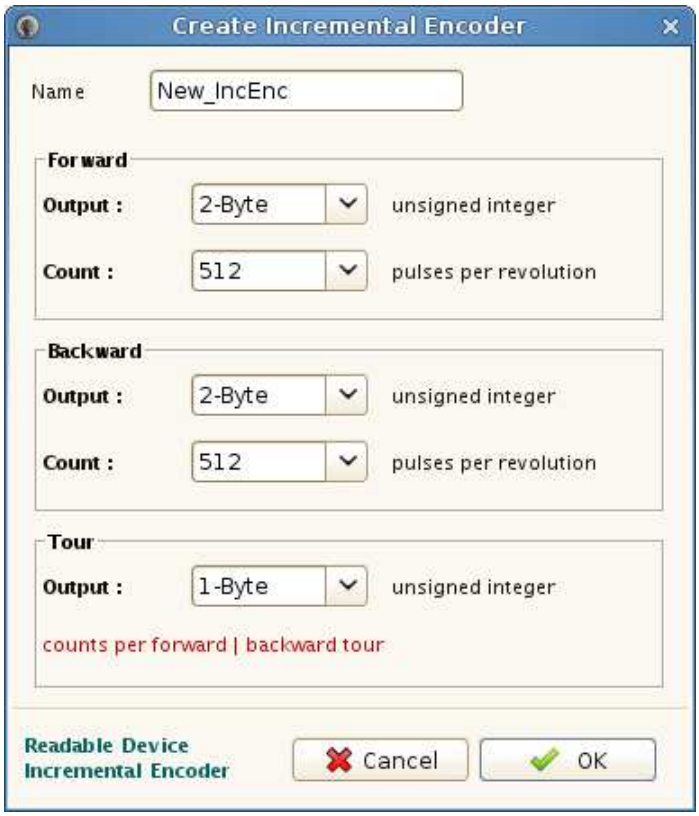
4.4 Simülasyonda Robot Aygıtlarının Temsili

Simülasyonda, robotun temel parçalarını oluşturan ve haberleşme protokolünde bir adres ile belirtilen üçü okunabilir ve ikisi yazılabilir türde olmak üzere beş adet aygıt türünün tanımlanmasına olanak sağlanmıştır. Bu aygıtlar ve tanımlamalarının nasıl yapılacağı şu şekildedir:

4.4.1 Artımlı Optik Kodlayıcı Tanımı

Artımlı optik kodlayıcı, robotun ön tekerleğinin dönme miktarının bulunmasına imkân tanıyarak robotun aldığı yol miktarının hesaplanabilmesini sağlamaktadır. Kodlayıcı, bağlı olduğu ön tekerleğin dönüşü ile darbeler (*pulses*) üretmekte ve bu darbelerin sayısı kodlayıcı bilgisi olarak tutulmaktadır. Mevcut robotlarda ön tekerleğin her 1 tam turunda 512 artım yapan 2 bayt ileri ve 2 bayt da geri dönüş sayısı bilgisi ve tekerleğin 1 tam turunu sayan 1 bayt tekerlek devir sayısı bilgisi tutulmaktadır.

Robots (*Robotlar*) → Devices (*Aygıtlar*) → Incremental Encoders (*Artımlı Kodlayıcılar*) menüleri seçilerek açılan pencereden daha önce tanımlanmış olan bu türdeki aygıtları görebilme, bu aygıtları düzenleyip silebilme ve yeni tip aygıt oluşturabilme işlemleri yapılabilmektedir. Bu pencereden, Add (*Ekle*) düğmesi seçilerek açılan pencereden yeni artımlı optik kodlayıcı tanımı yapılabilmektedir (Şekil 4.6).



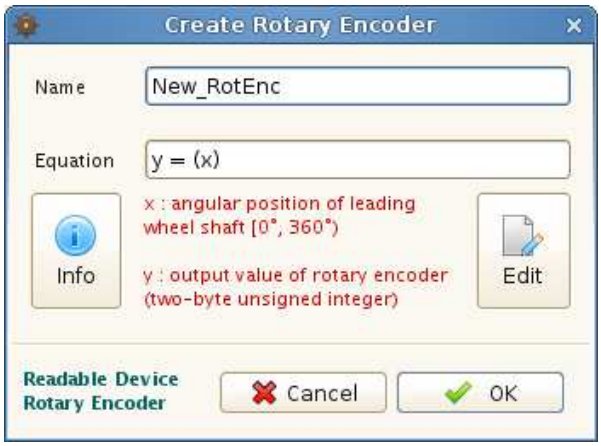
Şekil 4.6 Artımlı optik kodlayıcı tanımlama penceresi

Yeni artımlı optik kodlayıcı tanımı yapılırken aygıta bir isim verilmeli; ileri, geri dönüş ve devir bilgileri için tutulacak verinin boyutları bayt cinsinden girilmeli, ileri ve geri dönüşün her 1 tam turunda üretilecek darbelerin sayısı (*pulses per revolution*) tanımlanmalıdır.

4.4.2 Döner Kodlayıcı Tanımı

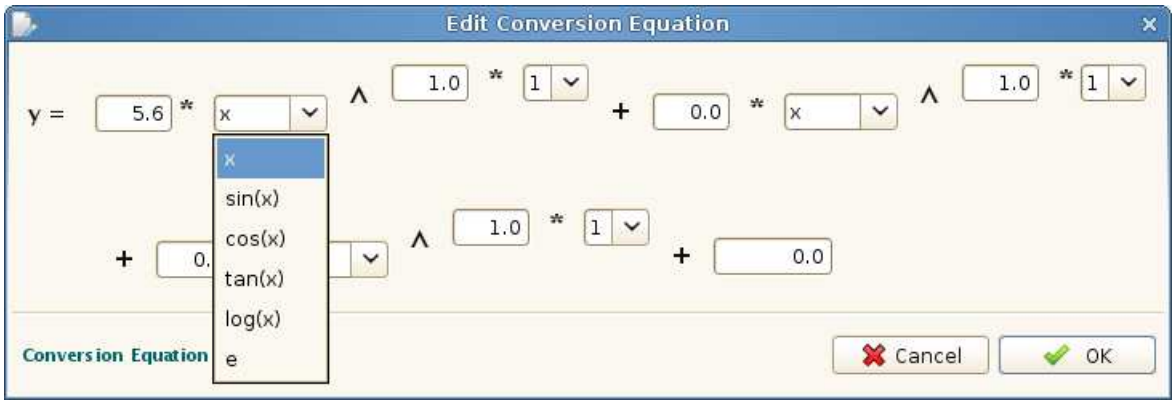
Döner kodlayıcı, robotta hareket yönünün algılanmasına olanak tanıyan parçadır. Ön tekerleğin dönme miktarını ölçen okunabilir türdeki aygıttır. Mevcut robotlardaki aygıt, 2 baytlık bir değerle ölçüm sonucu üretmektedir. Tekerleğin dönme miktarı ve aygıtın ürettiği ölçüm sonucu arasında denklemle ifade edilebilen bir oran bulunmaktadır.

Robots (*Robotlar*) → Devices (*Aygıtlar*) → Rotary Encoders (*Döner Kodlayıcılar*) menüleri seçilerek açılan pencereden daha önce tanımlanmış olan bu türdeki aygıtları görebilme, bu aygıtları düzenleyip silebilme ve yeni tip aygıt oluşturabilme işlemleri yapılabilmektedir. Bu pencereden, Add (*Ekle*) düğmesi seçilerek açılan pencereden yeni döner kodlayıcı tanımı yapılabilmektedir (Şekil 4.7).



Şekil 4.7 Döner kodlayıcı tanımlama penceresi

Yeni döner kodlayıcı tanımı yapılırken aygıta bir isim verilmeli ve tekerlek dönme açısı - kodlayıcı ölçüm sonucu dönüşümünü ifade eden denklem tanımlanmalıdır. Bu denklem, Edit (*Düzenle*) düğmesi seçilerek açılan pencereden tanımlanabilmektedir (Şekil 4.8).

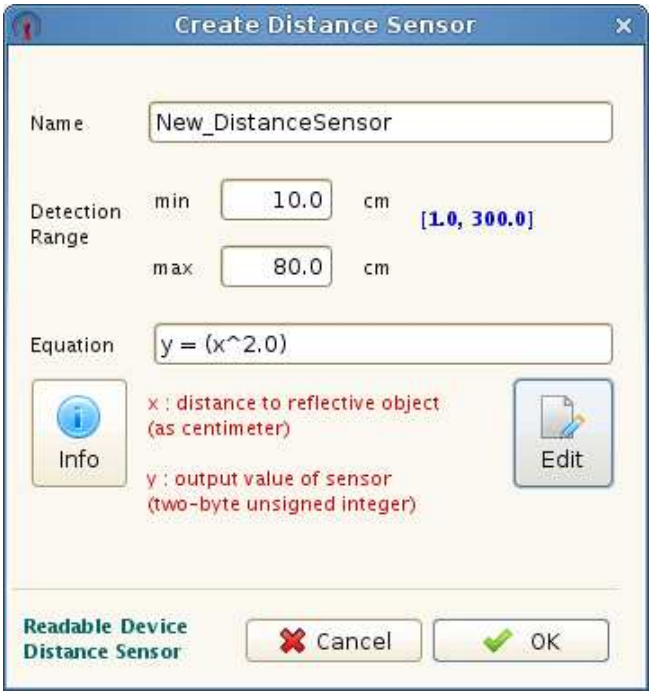


Şekil 4.8 Dönüşüm denklemi tanımlama penceresi

4.4.3 Mesafe Duyargası Tanımı

Mesafe duyargaları, robotun bulunduğu ortamdaki cisimlerin algılanabilmesini sağlamaktadırlar. Duyargalar belirli bir mesafeye kadar olan cisimlerin uzaklıklarını tespit edebilen ve çeşitli tiplerde olabilen okunabilir türdeki aygıtlardır. Mevcut robotlardaki aygıtlar, 2 baytlık bir değerle ölçüm sonucu üretmektedirler. Tespit edilen cisme olan uzaklık ve duyarga ölçüm sonucu arasında denklemle ifade edilebilen bir oran bulunmaktadır.

Robots (*Robotlar*) → Devices (*Aygıtlar*) → Distance Sensors (*Mesafe Duyargaları)* menüleri seçilerek açılan pencereden daha önce tanımlanmış olan bu türdeki aygıtları görebilme, bu aygıtları düzenleyip silebilme ve yeni tip aygıt oluşturabilme işlemleri yapılabilmektedir. Bu pencereden Add (*Ekle*) düğmesi seçilerek açılan pencereden yeni mesafe duyargası tanımı yapılabilmektedir (Şekil 4.9).



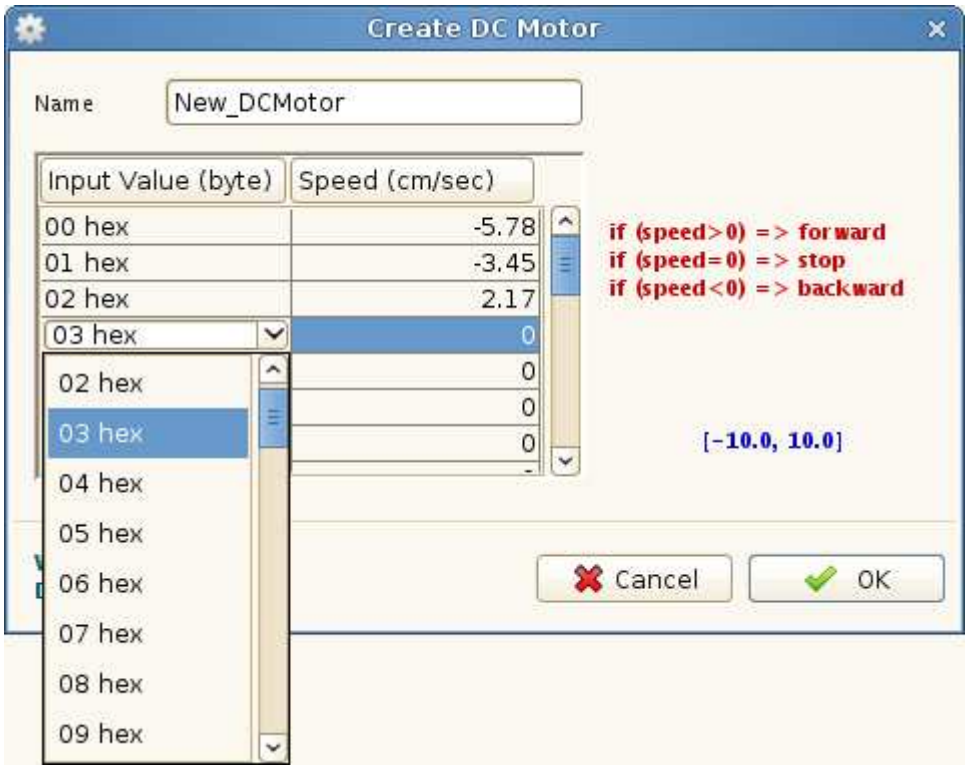
Şekil 4.9 Mesafe duyargası tanımlama penceresi

Yeni mesafe duyargası tanımı yapılırken aygıta bir isim verilmeli, duyarganın tespit edebileceği en yakın ve en uzak mesafe değerleri santimetre cinsinden girilmeli ve cisim uzaklığı - duyarga ölçüm sonucu dönüşümünü ifade eden denklem daha önce açıklandığı gibi Edit (*Düzenle*) düğmesi seçilerek açılan pencereden tanımlanmalıdır.

4.4.4 DC Motor Tanımı

DC motor, robotun ileri ve geri yönlü hareketini sağlamakla görevlidir. Haberleşme paketi ile gönderilen 1 baytlık kontrol işaretine bağlı olarak motor sürücü devresi (*H-Bridge*) tarafından kontrol edilmekte ve robotun belirli bir hızla hareket etmesini sağlamaktadır.

Robots (*Robotlar*) → Devices (*Aygıtlar*) → DC Motors (*DC Motorlar*) menüleri seçilerek açılan pencereden daha önce tanımlanmış olan bu türdeki aygıtları görebilme, bu aygıtları düzenleyip silebilme ve yeni tip aygıt oluşturabilme işlemleri yapılabilmektedir. Bu pencereden Add (*Ekle*) düğmesi seçilerek açılan pencereden yeni DC motor tanımı yapılabilmektedir (Şekil 4.10).



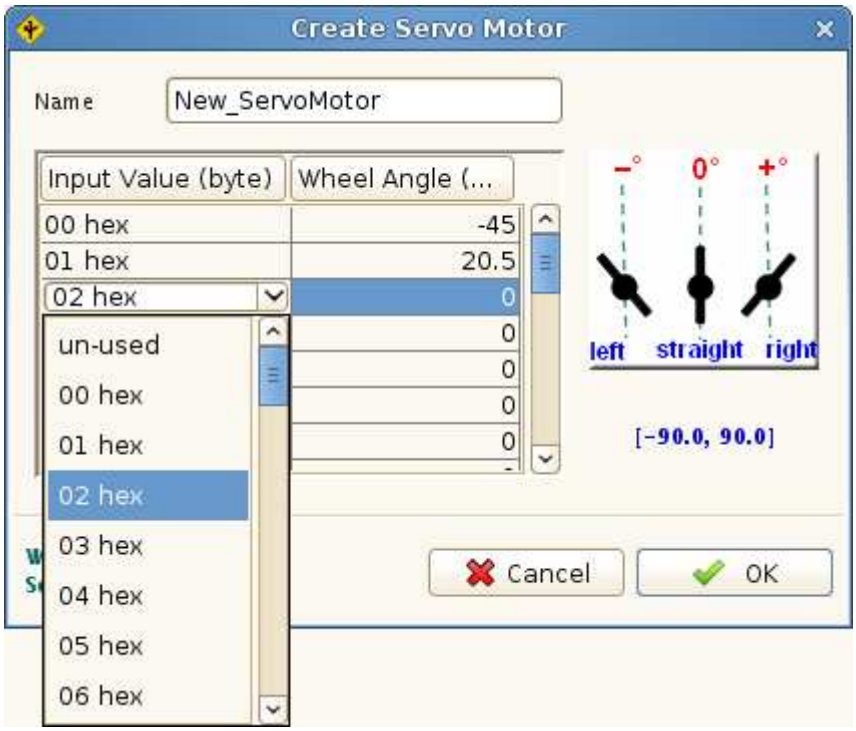
Şekil 4.10 DC Motor tanımlama penceresi

Yeni DC motor tanımı yapılırken aygıta bir isim verilmeli, motorun hangi kontrol işaretinde ön tekerleği hangi hızla ve yönde kontrol edeceği tanımlanmalıdır. Hız tanımlamaları santimetre/saniye cinsinden verilmelidir. Tanımlaması yapılmayan bir kontrol işareti geldiğinde hızda herhangi bir değişiklik yapılmayacaktır ve bu durumun kontrolü kullanıcıya bırakılmıştır. Hız tanımlamasında sıfırdan büyük değerler ileri yönlü hareketi, sıfırdan küçük değerler geri yönlü hareketi, sıfır ise durmayı ifade etmektedir.

4.4.5 Servo Motor Tanımı

Servo motor, ön tekerleğin sağa veya sola dönüşünü sağlayarak robotun yönlendirilmesine imkân tanımaktadır. Haberleşme paketi ile gönderilen 1 baytlık kontrol işareti ile servo motorun ön tekerleği belirli bir açı ile döndürmesi sağlanmaktadır.

Robots (*Robotlar*) → Devices (*Aygıtlar*) → Servo Motors (*Servo Motorlar*) menüleri seçilerek açılan pencereden daha önce tanımlanmış olan bu türdeki aygıtları görebilme, bu aygıtları düzenleyip silebilme ve yeni tip aygıt oluşturabilme işlemleri yapılabilmektedir. Bu pencereden Add (*Ekle*) düğmesi seçilerek açılan pencereden yeni servo motor tanımı yapılabilmektedir (Şekil 4.11).



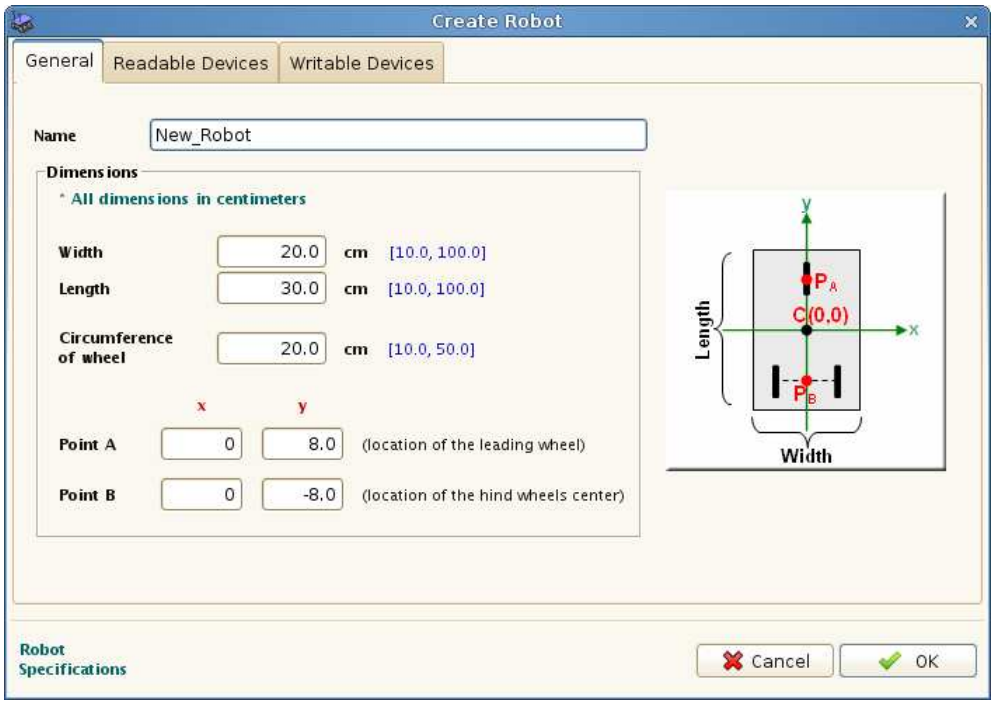
Şekil 4.11 Servo motor tanımlama penceresi

Yeni servo motor tanımı yapılırken aygıta bir isim verilmeli, motorun hangi kontrol işaretinde tekerleği hangi yöne ve hangi açıyla ne kadar döndüreceği tanımlanmalıdır. Yön tanımlamaları derece cinsinden verilmelidir. Tanımlaması yapılmayan bir kontrol işareti geldiğinde teker yönlendirilmesinde herhangi bir değişiklik yapılmayacaktır ve bu durumun kontrolü kullanıcıya bırakılmıştır. Yön tanımlamasında sıfırdan büyük değerler tekerleğin sağa dönüşünü, sıfırdan küçük değerler tekerleğin sola dönüşünü, sıfır ise tekerleğin düz durumda olduğunu ifade etmektedir.

4.5 Simülasyonda Robot Tanımlanması

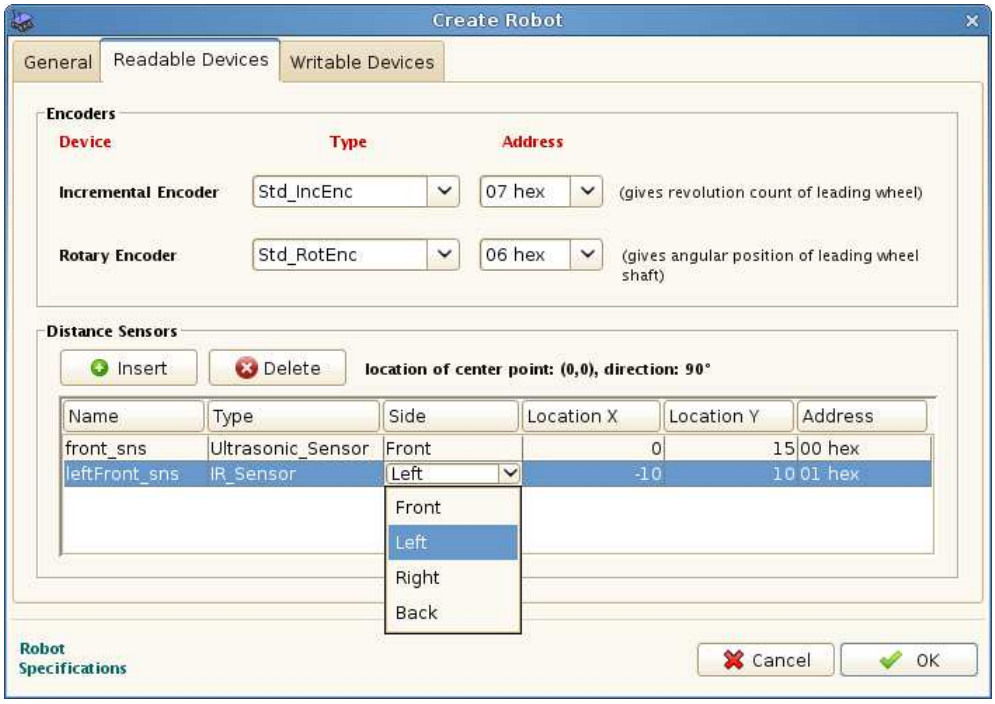
Simülasyonda, farklı tiplerde robotların tanımlanmasına ve aynı anda kullanılmasına olanak sağlanmıştır. Robots (*Robotlar*) → Robot Types (*Robot Tipleri*) menüleri seçilerek açılan pencereden daha önce tanımlanmış robot tiplerini görebilme, bu robot tiplerini düzenleyip silebilme ve yeni robot tipleri oluşturabilme işlemleri yapılabilmektedir. Bu pencereden Add (*Ekle*) düğmesi seçilerek açılan pencereden yeni ve istenilen konfigürasyonda robotların tanımlanması mümkün olmaktadır.

Robot tanımlama penceresinin General (*Genel*) sekmesinde, robotun temel özelliklerinin tanımlaması yapılmaktadır (Şekil 4.12). Bu sekmede robota bir isim verilmeli, robotun en ve boy uzunlukları, tekerleğinin çevre uzunluğu ve robotun kinematik hesaplamaları için gerekli olan robotun ön teker orta noktası (*A noktası*) ve arka tekerlerinin orta noktasının (*B noktası*) konum bilgileri tanımlanmalıdır.



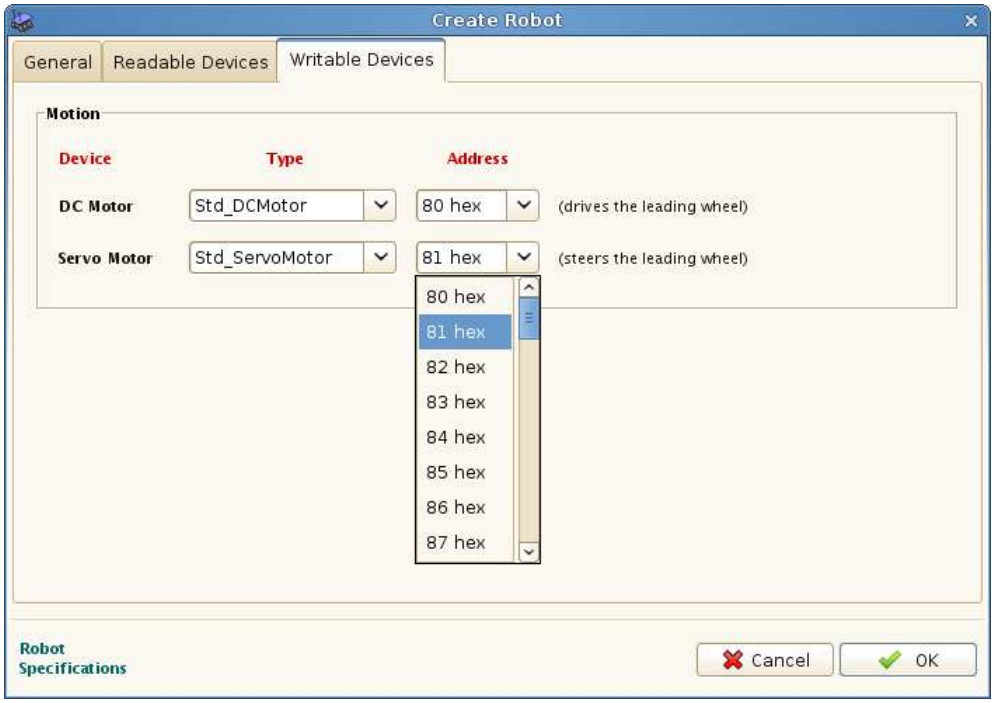
Şekil 4.12 Robot tanımlama penceresi genel sekmesi

Robot tanımlama penceresinin Readable Devices (*Okunabilir Aygıtlar*) sekmesinde, okunabilir aygıtlarla ilgili tanımlamalar yapılmaktadır (Şekil 4.13). Okunabilir aygıtlardan artımlı optik kodlayıcı ve döner kodlayıcı için tip ve adres seçimi yapılmalı ve mesafe duyargaları isim, tip, yön, konum ve adres bilgileri tanımlanarak istenilen sayıda oluşturulmalıdır.



Şekil 4.13 Robot tanımlama penceresi okunabilir aygıtlar sekmesi

Robot tanımlama penceresinin Writable Devices (*Yazılabilir Aygıtlar*) sekmesinde, yazılabilir aygıtlarla ilgili tanımlamalar yapılmaktadır (Şekil 4.14). Yazılabilir aygıtlardan DC ve servo motorlar için tip ve adres seçimleri yapılmalıdır.



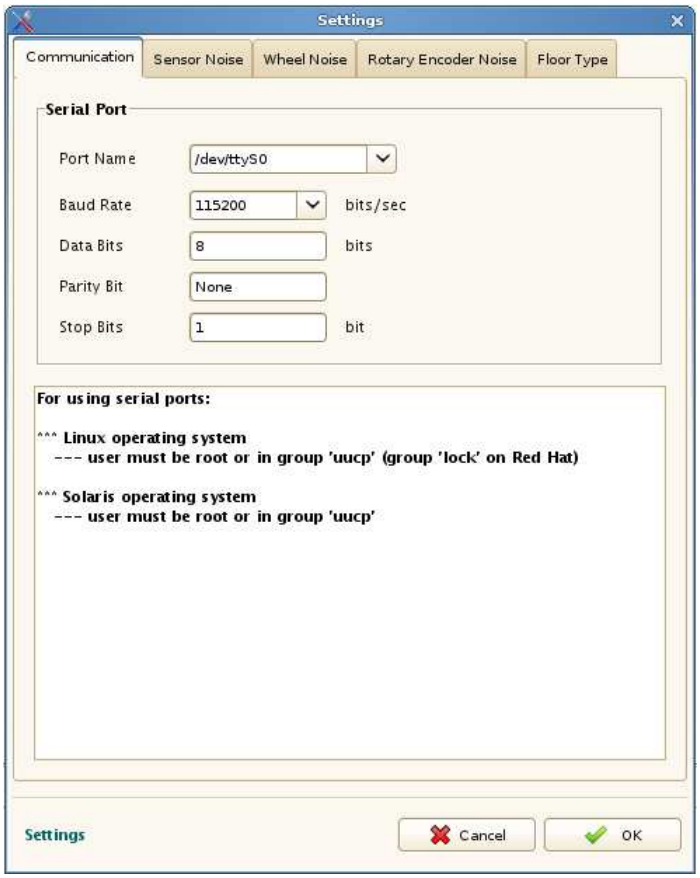
Şekil 4.14 Robot tanımlama penceresi yazılabilir aygıtlar sekmesi

4.6 Simülasyon Parametre Ayarları

Simülasyonun çalışma zamanı için bazı ayarların yapılması gerekmektedir. Start (*Başlat*) → Settings (*Ayarlar*) menüleri seçilerek açılan pencerede yapılabilecek parametre ayarları şu şekildedir:

4.6.1 İletişim Ayarları

Settings (*Ayarlar*) penceresi, Communication (*İletişim*) sekmesinde simülasyonun kontrolü için gerekli olan seri port haberleşmesinin parametre ayarları yapılabilmektedir (Şekil 4.15). Sekmede, iletişimin yapılacağı port ve iletişimin gerçekleştirileceği hız seçimleri yapılmaktadır. Seri iletişim 8 veri biti, eşlik bitsiz ve 1 durma biti (8/N/1) ile gerçekleştirilmektedir ve bu ayarlar değiştirilememektedir.

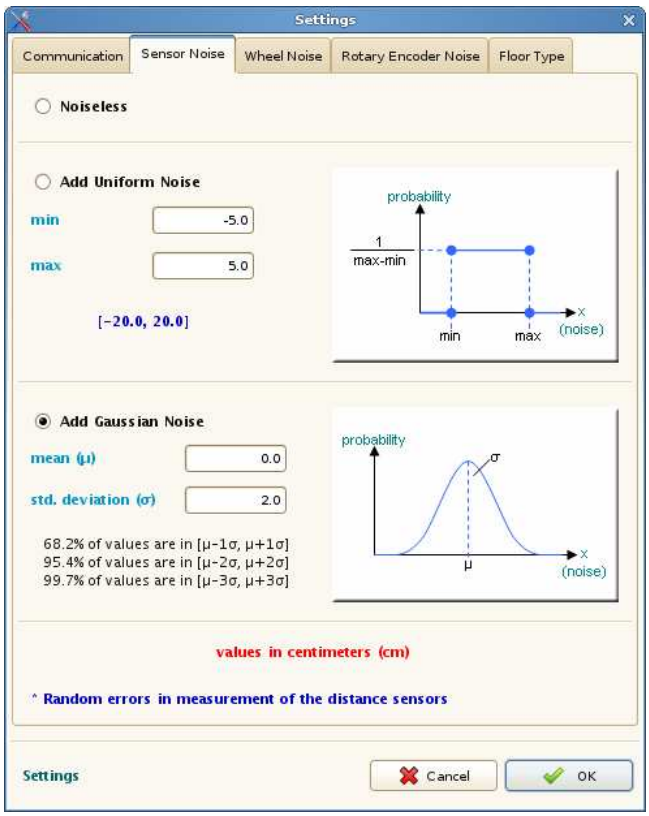


Şekil 4.15 İletişim parametreleri ayarlama sekmesi

Linux ve Solaris işletim sisteminde seri portu kullanabilmek için sistem kullanıcısının bu hakkı elde etmiş olması gerekmektedir. Bu yüzden Linux veya Solaris işletim sisteminde kullanıcının root olması veya ‘uucp’ (Red Hat’ta ‘lock’) grubunda yer alması gerekmektedir.

4.6.2 Mesafe Duyargalarına Gürültü Ekleme

Mesafe duyargaları, gerçek çalışmalarında ölçümlerini rastgele meydana gelen bir hata oranıyla gerçekleştirebilmektedirler. Bu durumu simülasyon ortamında oluşturabilmek için duyarga ölçümlerine gürültü eklenmesine olanak sağlanmıştır. Eklenecek gürültünün oranı, parametreleri belirlenebilen düzgün dağılım (*uniform distribution*) veya normal dağılıma (*Gaussian distribution*) uyacak biçimde ayarlanabilmektedir. Bu işlem, Settings (*Ayarlar*) penceresinin Sensor Noise (*Duyarga Gürültüsü*) sekmesinden yapılabilmektedir (Şekil 4.16).



Şekil 4.16 Duyarga gürültüsü ekleme sekmesi

4.6.3 Tekerlek Gürültüsü Ekleme

Gezgin robot, servo motora gönderilen kontrol işareti ile yönlendirilmektedir. Gerçek çalışma anında çeşitli sebeplerden dolayı tekerleğin döndürülmesinde rastgele meydana gelen hata oranlarında sapmalar meydana gelebilmektedir. Bu durumu simülasyon ortamında oluşturabilmek için servo motorun ön tekerleği kontrolü esnasında tekerleğin dönüş miktarında belirli bir oranda sapma meydana getirilmesine olanak sağlanmıştır. Bu işlem, Settings (*Ayarlar*) penceresi Wheel Noise (*Tekerlek Gürülüsü*) sekmesinden yapılabilmektedir. Sapma miktarı, parametreleri belirlenebilen düzgün veya normal dağılıma uyacak biçimde ayarlanabilmektedir.

4.6.4 Döner Kodlayıcıya Gürültü Ekleme

Döner kodlayıcı, gerçek çalışmasında mesafe duyargalarında olduğu gibi ölçümünü rastgele meydana gelen bir hata oranıyla gerçekleştirebilmektedir. Bu durumu simülasyon ortamında oluşturabilmek için kodlayıcının ölçümüne gürültü eklenmesine olanak sağlanmıştır. Bu işlem, Settings (*Ayarlar*) penceresi Rotary Encoder Noise (*Döner Kodlayıcı Gürültüsü*) sekmesinden yapılabilmektedir. Gürültü miktarı, parametreleri belirlenebilen düzgün veya normal dağılıma uyacak biçimde ayarlanabilmektedir.

4.6.5 Zemin Tipinin Ayarlanması

Gezgin robot, çalışması esnasında zeminin özelliklerinden dolayı patinaj veya kayma ile karşılaşabilmektedir. Böyle bir durumun sonucu olarak artımlı optik kodlayıcıdan elde edilen değerler, alınan gerçek mesafeden daha farklı olmaktadır. Bu sebeple de robotun konumunun tahmininde ve dolayısıyla çevrenin haritasının çıkarılmasında zamanla artış gösteren bir hata meydana gelmektedir. Bu durumu simülasyon ortamında oluşturabilmek için zemin tipinin ayarlanmasına olanak sağlanarak, parametreleri ayarlanabilecek biçimde düzgün veya normal dağılım oranlarında robotun patinaj veya kayma olaylarına maruz kalması sağlanabilir. Bu işlem, Settings (*Ayarlar*) penceresi Floor Type (*Zemin Tipi*) sekmesinden yapılabilmektedir.

4.7 Simülasyonun Çalışma Mantığı

Harita alanında engeller oluşturulup, istenilen sayıda ve tipte robot harita alanına konumlandırılarak önceki bölümde açıklanan parametre ayarları yapıldıktan sonra simülasyon çalıştırılabilir hale gelmektedir. Uygulama çalıştırıldıktan sonra belirlenen port dinlenmeye başlanmakta, alınan komutlar ilgili robot için uygulanıp istenilen robot bilgileri bu porttan daha önce açıklanan protokol yapısına göre gönderilmektedir.

Simülasyonun güncellenmesi insanın algılayamayacağı sıklıkla, gerçek zamanlı olarak gerçekleştirilmektedir. Güncelleme için, en son güncelleme ve o andaki zaman arasında oluşan farktan yararlanılmaktadır. Belirlenen zaman aralığında her bir robot için, robotun ne kadar hareket edeceği hesaplanmakta ve robotun kinematik özelliklerinden yararlanılarak robotun son konumu bulunmaktadır. Daha sonra bu konumda robot üzerideki duyargaların yapacakları ölçümler hesaplanmakta, robottaki diğer aygıtların ölçüm değerleri de hesaplanarak ekranda gösterilmektedir. Yapılan bu güncellemenin süresini robot sayısı, ortamdaki engel sayısı ve robotlardaki duyarga sayısı belirlemektedir. Bu etmenlerin güncelleme süresine etkileri Çizelge 4.1’de görülmektedir.

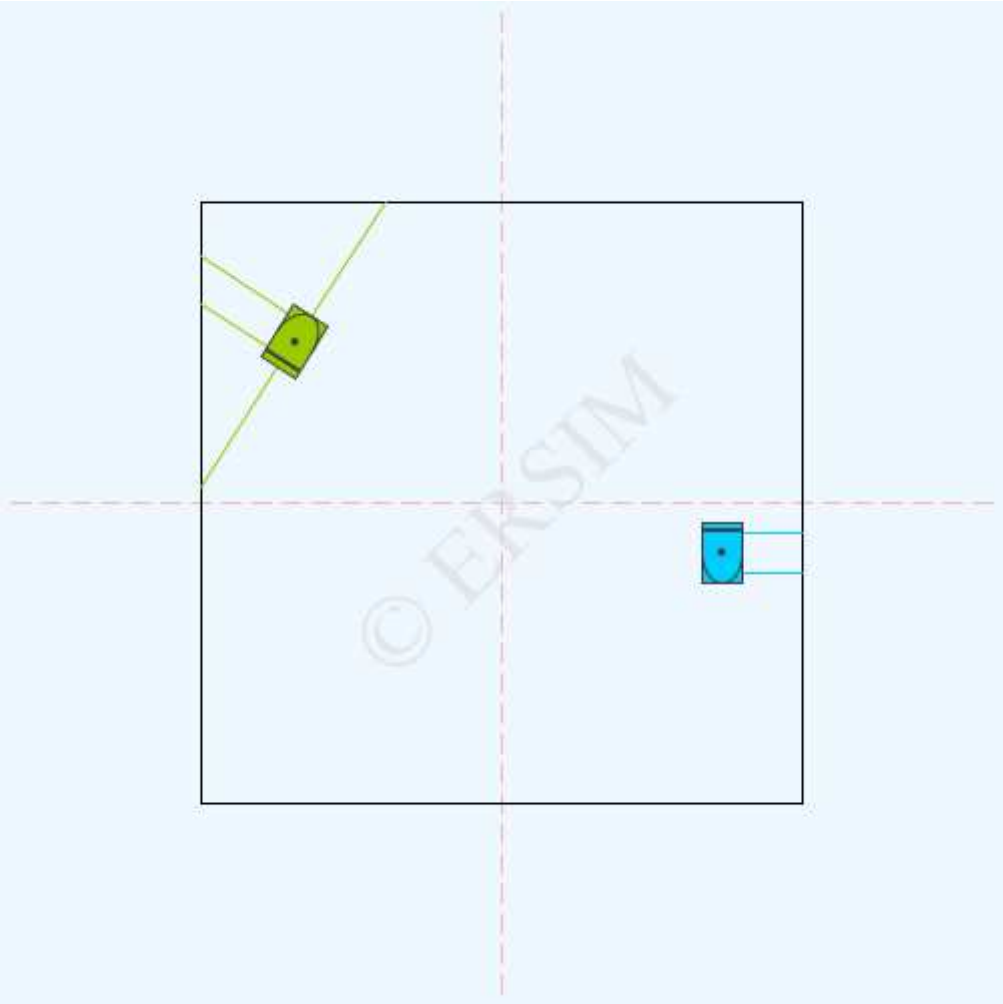
Microsoft Windows XP Home Edition işletim sistemi, 2×1.66 GHz işlemci ve 3 GB RAM özelliklerinde bir donanımda, değişik konfigürasyonlarda 500 güncelleme süresinin ortalamaları Çizelge 4.1'de görülebilmektedir.

Çizelge 4.1 Farklı durumlarda simülasyonun güncellenmesi için geçen süreler

Robot Sayısı	Robotlardaki Duyarga Sayısı	Haritadaki Engel Sayısı	Güncelleme Süresi (milisaniye)
1	5	10	**0.332**
1	5	20	**0.421**
1	5	30	**0.478**
1	10	10	**0.412**
1	10	20	**0.526**
1	10	30	**0.608**
2	5	10	**0.596**
2	5	20	**0.696**
2	5	30	**0.790**
2	10	10	**0.746**
2	10	20	**0.934**
2	10	30	**1.101**
5	5	10	**1.663**
5	5	20	**1.899**
5	5	30	**2.181**
5	10	10	**2.392**
5	10	20	**2.828**
5	10	30	**3.278**
10	5	10	**4.835**
10	5	20	**5.451**
10	5	30	**5.975**
10	10	10	**7.412**
10	10	20	**8.302**
10	10	30	**9.076**
16	5	10	**11.187**
16	5	20	**12.010**
16	5	30	**12.912**
16	10	10	**17.113**
16	10	20	**18.726**
16	10	30	**19.904**

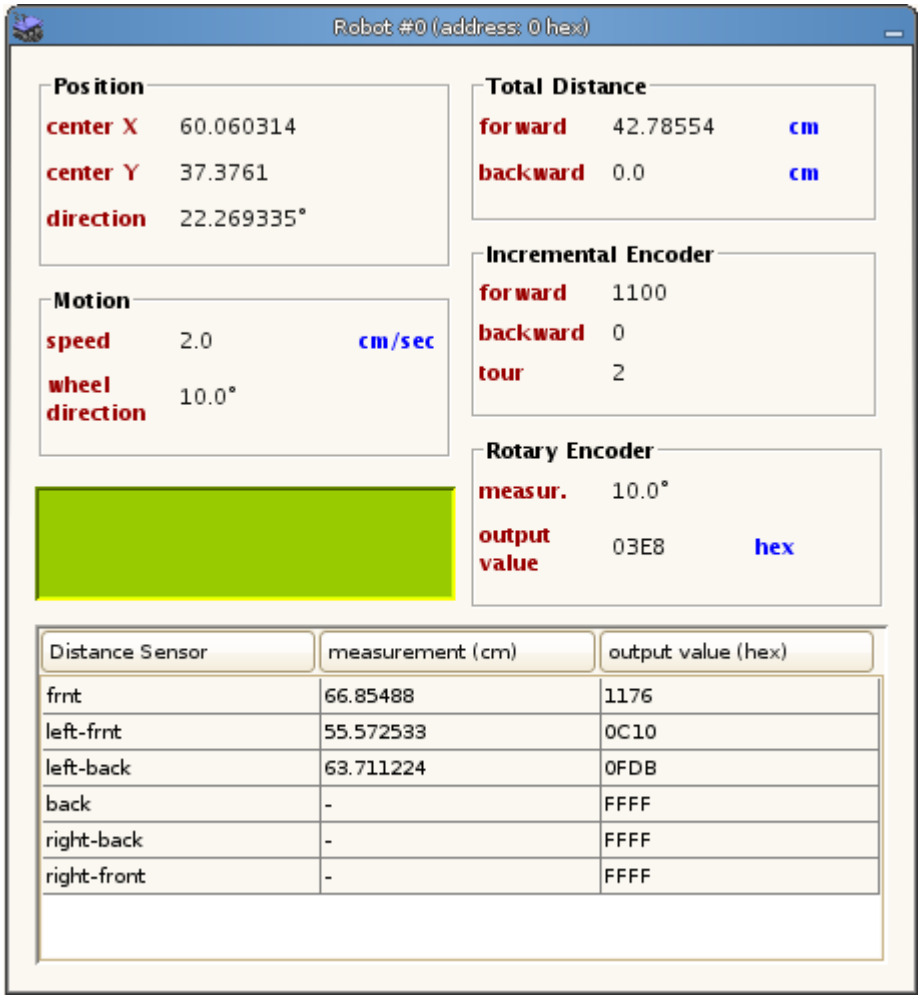
Çizelge incelendiğinde, güncelleme süresi üzerindeki en büyük etkiyi robot sayısının yaptığı görülmektedir. Bu sürelerin güncellemenin ekrana yansıtılmasını da içerdiği düşünüldüğünde elde edilmiş olan sürelerin kabul edilebilir düzeyde olduğu söylenebilir.

Simülasyon güncellendikten sonra robotların yeni konumları, engeller ve duyarga ölçümleri uygulamanın ortasında yer alan harita alanında gösterilmektedir (Şekil 4.17).



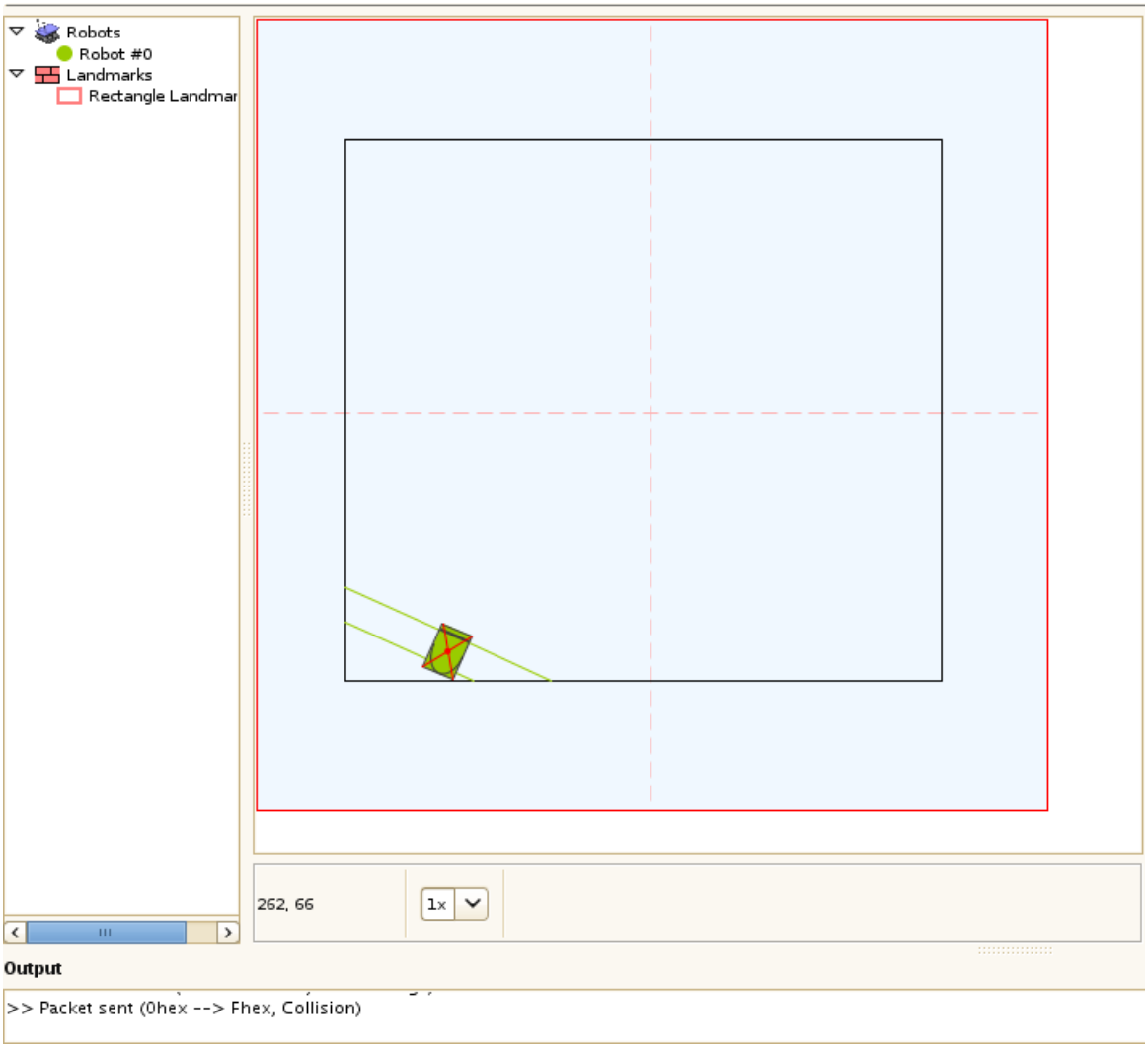
Şekil 4.17 Çalışma anında simülasyon harita alanının görünümü

Ayrıca çalışma esnasında ekranın sağındaki robot bilgileri gösterim alanında her robot için açılan ayrı bir iç pencerede robotun konumunu, yönünü, kat ettiği toplam yolu, hızını, ön tekerleğin dönüş açısını, artımlı optik kodlayıcının ve döner kodlayıcının ölçüm değerlerini ve sahip olunan duyargaların algıladığı cisimlerin mesafe ve ölçüm değerlerini görebilme imkânı sağlanmaktadır (Şekil 4.18).



Şekil 4.18 Çalışma anında robot bilgileri gösterim penceresinin görünümü

Simülasyonun güncellenmesinde robotların engellere veya birbirlerine çarpma ihtimalleri değerlendirilmekte, böyle bir durumun oluşması halinde robot durdurulmakta ve bu oluşan durumu bildirmek üzere F_{hex} adresine (kumanda) Collision paketi gönderilmektedir (Şekil 4.19).



Şekil 4.19 0_{hex} adresli robotun engele çarpma durumu

5. ROBOT KONTROLÜ

Robot kontrolü, belirlenen işin yapılabilmesi için robota gerekli komutların verilmesidir. Bu işlem, Gifford (2008) tarafından şöyle açıklanıyor: “Robotlara ne yapacağının söylenmesi gerekir; ancak bu ‘Git oradaki vidayı al!’ gibi bir komutla olmaz. Robotun istenen işi yapabilmesi için bütün parçalarını nasıl hareket ettirmesi gerektiğini belirten bir dizi kesin komuta ihtiyacı vardır. Robotların çoğu bu komutları bir bilgisayar veya mikroişlemciden alır. Bilgisayar programları robotun denetim sistemine veya doğrudan, hareket ettirici adı verilen tahrik sistemi elemanlarına elektrik sinyalleri göndererek robot parçalarının hareketini sağlar. Bu sinyaller robotun istenen işi yapmasına olanak tanır.”.

Gezgin robotun kontrolü, haberleşme protokolüne göre hazırlanmış olan uygun içerikteki veri paketlerinin robota iletilmesiyle gerçekleştirilmektedir. Paketler alındıktan sonra robotun merkezi kontrol kartı tarafından çözümlenmekte ve alınan komutlara göre hareket ettiriciler uygun şekilde kontrol edilmektedir. İlerleyen bölümlerde, simülasyonu veya robotu bu şekilde kontrol edebilecek olan kumanda yapılarından bahsedilecektir.

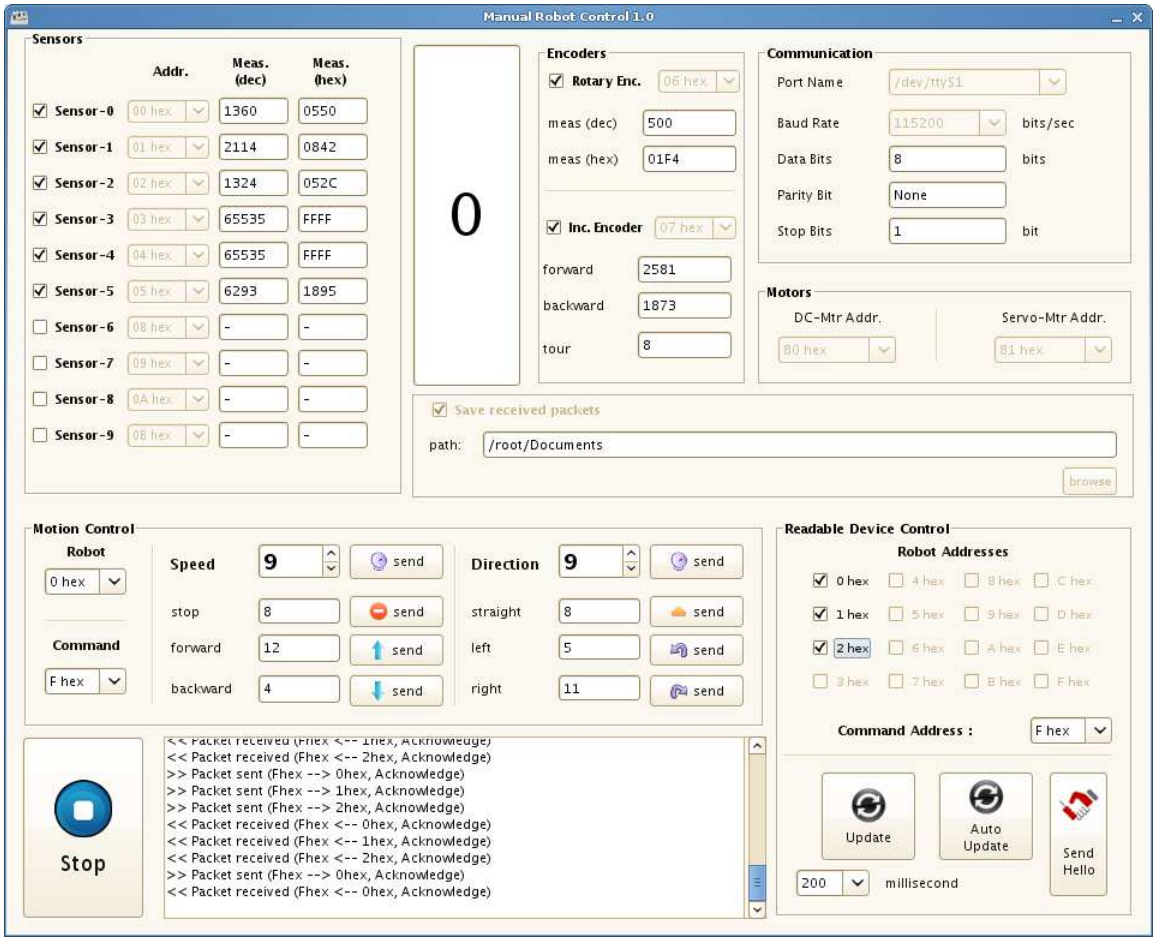
Uygulanan tüm yöntemlerde iletişim seri port üzerinden sağlanmaktadır. Robotun kontrolü için seri porta bağlanacak RF veya Bluetooth cihazı ile, farklı bir bilgisayarda bulunan simülasyon ortamının kontrolü için bilgisayarın seri portlarının kablosuz veya kablolu olarak bağlanması ile veya aynı bilgisayar üzerinde bulunan simülasyon ortamının kontrolü için sanal seri port (*virtual serial port*) kullanılması ile kumanda ortamının karşı taraf ile haberleşebilmesi sağlanabilir.

5.1 Manuel Robot Kontrolü

Manuel robot kontrolü, gerçek robotları veya simülasyon uygulamasını elle kontrol etmeyi amaçlamaktadır. Bu doğrultuda, robotlara komut gönderip veri alabilecek bir kumanda uygulaması geliştirilmiştir (Şekil 5.1). Uygulama çalıştırılmadan önce robot aygıtlarının adreslerinin belirlenmesi ve kontrol edilecek olan robotların adres seçilmelerinin yapılması gerekmektedir. Kumanda ile artımlı optik kodlayıcı, döner kodlayıcı ve 10 adete kadar mesafe duyargası ölçüm verisi alınabilip, DC ve servo motorlara kontrol işareti gönderilebilmektedir.

Kumanda ile yön veya hız kontrolü için istenilen robotun adresi seçilmekte ve belirlenen kontrol işareti gönderilebilmektedir. Ayrıca seçilen robotlardan işaretlenen okunabilir aygıtların ölçüm değerleri istenebilmektedir. Otomatik güncelleme seçeneği ile de belirli aralıklarla seçilen robotlardan ölçüm değerleri otomatik olarak istenilebilmektedir.

Başlangıçta seçimi yapılan tüm robotlar için ayrı metin dosyalarına, okunabilir aygıtlardan alınan ölçüm değerleri ve yazılabilir aygıtlara gönderilen kontrol işaretleri zaman bilgisiyle birlikte kaydedilmektedir. Bu şekilde, elde edilen veriler çevrimdışı işleme için saklanmış olunmaktadır.



Şekil 5.1 Manuel robot kumandası arayüzü

5.2 Otomatik Robot Kontrolü

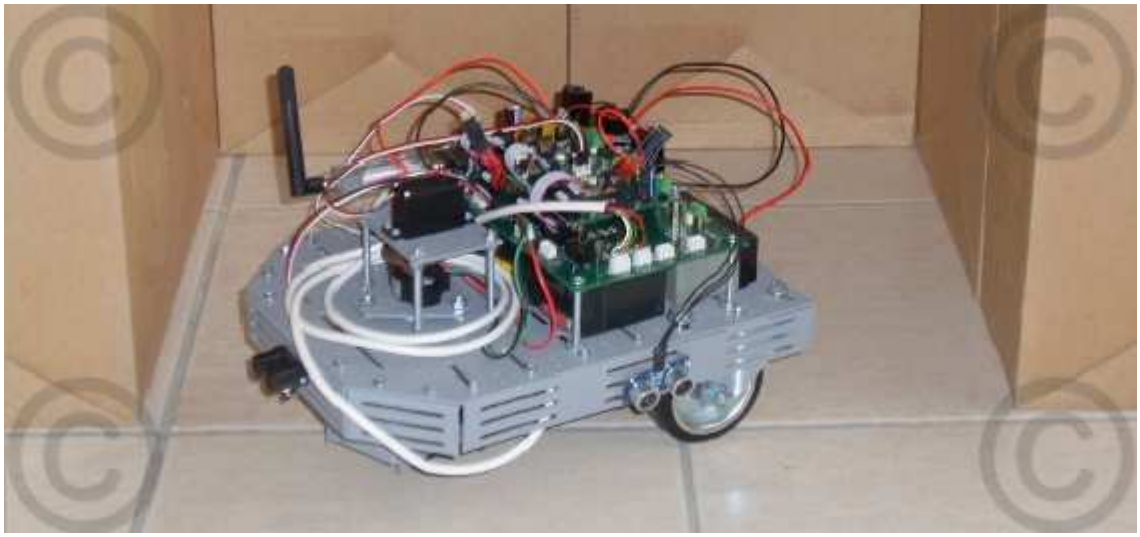
Bahsi geçen gezgin robotların geliştirilmelerinin sebebi, literatürde SLAM (*Simultaneous Localization and Mapping / Eş Zamanlı Konum Belirleme ve Haritalama*) olarak bilinen tekniğe yönelik uygulamalarda kullanılma amaçlıdır. Bu teknikte amaç, bilinmeyen bir ortamın haritasının gezgin araçlar kullanılarak çıkarılması ve bu işlem esnasında aracın konumunun tahmin edilmesine dayanmaktadır (Durrant-Whyte ve Bailey, 2006).

Bu yöntem, uygulama aşamasında değişik algoritmalar kullanılarak farklı biçimlerde hayata geçirilebilmektedir. Bu aşamada, bahsi geçen robotun bu amaç doğrultusunda nasıl kontrol edileceğine dair temel oluşturabilecek olan bir kontrol mekanizması geliştirilmeye

çalışılmıştır. Çeşitli algoritmalarla ve farklı şekillerde oluşturulan bu yöntemler ilerleyen bölümlerde ayrıntılı olarak açıklanacaktır.

5.2.1 Tek Robot Kontrolü

SLAM uygulamaları çoğunlukla tek robot kullanılarak gerçekleştirilmektedir. Robot, herhangi bir konumdan hareketine başlatılmakta ve üzerindeki donanımlar aracılığı ile elde ettiği ölçüm sonuçlarının değerlendirilmesi ile hareketine karar verilmektedir. Robotun ilk olarak konumu tahmin edilmeye çalışmakta daha sonra da elde edilen bu sonuç ve duyarga ölçümleri bir arada kullanılarak ortamın haritası çıkarılmaya çalışılmaktadır. Bu türde bir çalışmanın gezgin robot üzerinde uygulanması esnasında kaydedilmiş olan bir fotoğraf Şekil 5.2'de görülmektedir (Şahin ve Yavuz, 2009).



Şekil 5.2 Gezgin robot üzerinde yapılan çalışmalardan bir görünüm

Robotun konumu, Bölüm 3.4'te açıklanan kinematik denklemeler kullanılarak hesaplanmaktadır. Bu hesaplamalar için ilk olarak robotun kat ettiği yol miktarına ihtiyaç duyulmaktadır. Bu değeri belirlemek için artımlı optik kodlayıcıdan elde edilen veriler kullanılmaktadır. Artımlı optik kodlayıcı tekerleğin her 1 tam turunda 512 darbe üretmektedir ve kodlayıcı bilgisi bu darbeler sayılarak oluşturulmaktadır. Ölçüm anındaki artımlı optik kodlayıcıdan elde edilen değere *encoder(t),* bir önceki ölçümün alındığı andaki artımlı optik kodlayıcı değerine *encoder(t-1)* denilirse bu iki zaman aralığında robotun aldığı yol miktarı (5.1) bağıntısı kullanılarak elde edilebilir.

$$gidilen\_mesafe = \frac{encoder(t) - encoder(t-1)}{512} \times tekerlek\_çevre \tag{5.1}$$

Robotun ön tekerinin dönme miktarı da döner kodlayıcının ölçüm sonucu kullanılarak elde edilebilmektedir. Robotun bir önceki hesaplamadaki konumu ve bu iki değer kullanılarak robotun ölçüm anındaki pozisyonu kinematik denklemler vasıtası ile hesaplanabilmektedir. Konum hesabından sonra duyarga verileri kullanılarak ortamın haritası çıkarılmaya çalışılmaktadır.

Tüm bu hesaplamalar ideal şartlarda doğru sonuç üretecektir. Ancak donanımların yapıları gereği çalışmalarında birtakım hatalar meydana gelebilmektedir. Oluşabilen gürültülerden dolayı döner kodlayıcı veya duyarga ölçümleri belirli oranlarda hatalar içerebilecek, patinaj veya kayma gibi sebeplerden dolayı artımlı optik kodlayıcıdan elde edilen değerler alınan gerçek mesafeyi veremeyebilecek veya robotu yönlendirmek için gönderilen kontrol işareti ile robot tekerleği istediğimiz oranda döndürülemeyebilecektir. Bu gibi sebeplerden dolayı gerçek çalışma anında, kinematik hesaplamalar ile tahmin edilen robot konumunda sapmalar meydana gelebilecek ve oluşturulan harita gerçek ortamı tam olarak yansıtmayacaktır. Bu durum gösteriyor ki ideal şartlar gerçek çalışmada hiçbir zaman elde edilemeyecek, sadece simülasyon ortamında oluşturulabilecektir.

Tek robot kontrolü için iki farklı yöntem uygulanmıştır.

5.2.1.1 Engelden Sakınma Hareketi ile Robot Kontrolü

Tek robotun otomatik kontrolü için uygulanan ilk yöntem engelden sakınma hareketi ile robot kontrolü olmuştur. Bu yöntemde ilk olarak robot hareketine başlatılmakta, duyargaları aracılığı ile bir engele yaklaştığı tespit edildiğinde, engelden sakınma davranışı doğrultusunda tanımlanmış olan hareketi yaptıracak komut robota gönderilerek robotun engelden uzaklaşması sağlanmaktadır. Robotun ortamı dolaşması esnasında konumu hesaplanmakta ve duyarga ölçüm sonuçları kullanılarak ortamın haritası çıkarılmaya çalışılmaktadır.

Kontrol edilen robot, 10 ile 80 cm arasında ölçüm yapabilme kabiliyetine sahip, ön orta ve arka orta noktalarında 1’er, sağ ve sol yanlarında ise 2’şer adet olmak üzere toplam 6 adet mesafe duyargasına sahiptir.

Engelden sakınma hareketinin kararında duyarga ölçüm sonuçları kullanılmaktadır. Belirlenmiş eşik değerlerine göre robotun önünde bir engel tespit edildiğinde engelin bulunmadığı sağ veya sol yana doğru robot hareket ettirilmekte, sağ veya sol tarafında bir engel tespit edildiğinde ise tersi yöne robotun hareket etmesi sağlanmaktadır.

Harita çıkarılırken, duyarga ölçüm sonuçları kullanılarak ortamda bulunan cisimler üzerinde

tespit edilen noktaların konumlarından yararlanılmaktadır. Çalışma anında hesaplanan noktalar bir dosyaya kaydedilmekte ve daha sonra elde edilen bu noktalar grafiksel olarak çizdirilerek harita amaçlı kullanılmaktadır.

Yöntemde kullanılan algoritmanın adımları şu şekildedir:

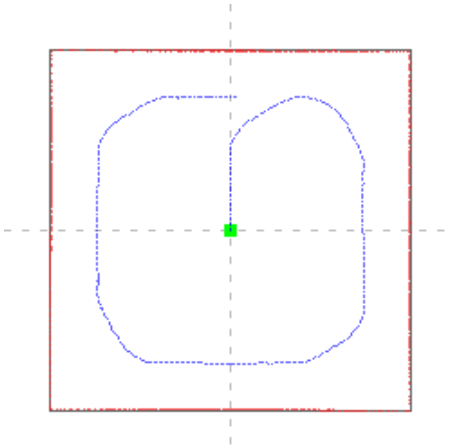
- Robotun başlangıç konumuna bir değer atanır.
- Sonraki adımlar sürekli olarak veya başlangıç konumuna dönene kadar tekrarlanır
  - Robot aygıtlarının ölçüm değerleri alınır.
  - Önceki konum, alınan yol miktarı ve tekerlek dönme ölçüm verileri kullanılarak kinematik denklemler ile robotun yeni konumu hesaplanır.
  - Duyarga ölçüm değerleri ile ortamdaki nesnelerin konumları hesaplanır (harita çıkarılır).
  - Duyarga ölçüm sonuçlarına göre gerekiyorsa robota engelden sakınma hareketi yaptırılır.

5.2.1.2 Engelden Sakınma Hareketi ile Elde Edilen Sonuçlar

Bu bölümde engelden sakınma hareketi ile robot kontrolünün farklı ortamlarda denenerek elde edilen harita sonuçları incelenecektir. Gösterilen haritalarda kesikli çizgiler koordinat eksenini, siyah çizgiler gerçek engelleri, kırmızı noktalar tespit edilen engeller üzerindeki noktaları, mavi noktalar robotun tahmin edilen yörüngesini, yeşil renkli kare ise robotun hareketine başladığı ilk konumu göstermektedir.

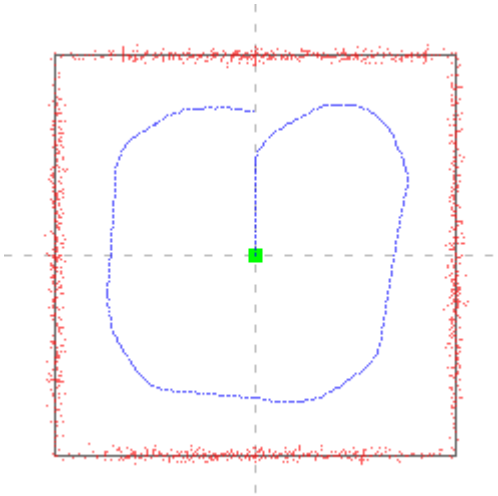
İlk olarak robot kenarları 2 m olan kare bir ortamda, karenin merkezinden ve 90 derece eğim ile ilk hareketine başlatılarak ortam farklı koşullarda dolaştırılmıştır. Robotun ileri yönlü hızı 7 cm/sn'dir. Değişik ortam şartlarında elde edilen sonuçlar şu şekildedir:

- İdeal şartlarda, ortam 1 dakika 27 saniyede dolaşılmış ve Şekil 5.3'deki sonuç elde edilmiştir.



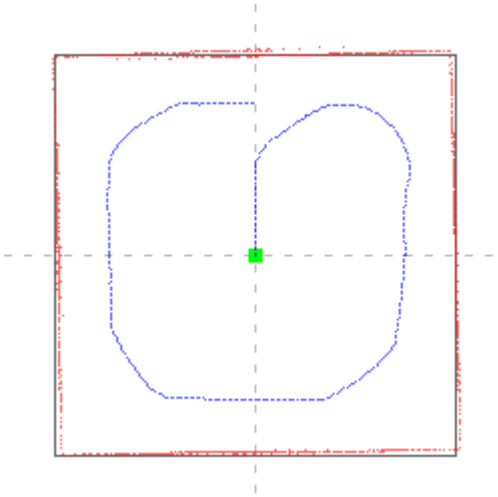
Şekil 5.3 Kare şeklindeki bir ortamın ideal şartlarda engelden sakınma hareketi ile tek robot tarafından çıkarılan haritası

● Duyargalara gürültü eklendiğinde (ortalaması 0 ve standart sapması 2 cm olan normal dağılıma uyacak biçimde rastgele olarak) ortam 1 dakika 25 saniyede dolaşılmış ve Şekil 5.4'deki sonuç elde edilmiştir.



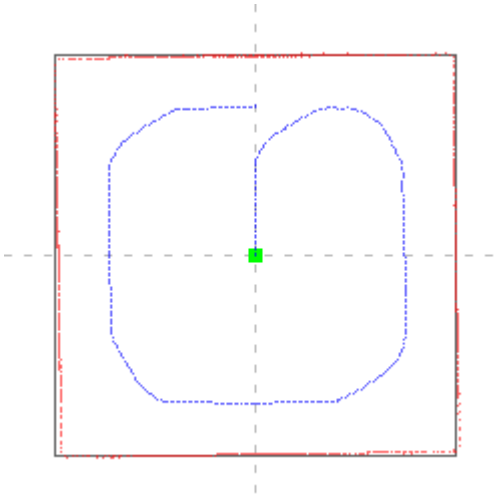
Şekil 5.4 Kare şeklindeki bir ortamın duyargalara gürültü eklenerek engelden sakınma hareketi ile tek robot tarafından çıkarılan haritası

● Tekerleğe ve döner kodlayıcıya gürültü eklendiğinde (ortalaması 0 ve standart sapması 2° olan normal dağılıma uyacak biçimde rastgele olarak) ortam 1 dakika 25 saniyede dolaşılmış ve Şekil 5.5'deki sonuç elde edilmiştir.



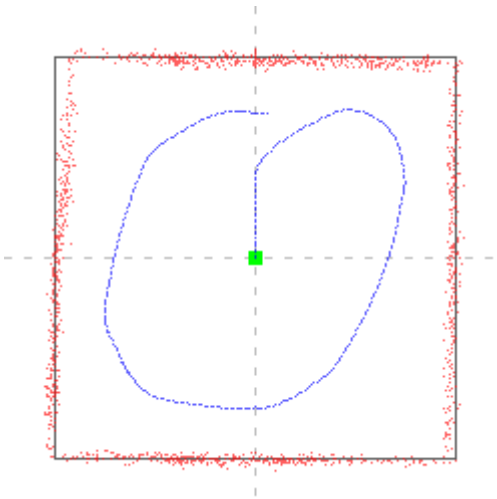
Şekil 5.5 Kare şeklindeki bir ortamın tekerlek ve döner kodlayıcıya gürültü eklenerek engelden sakınma hareketi ile tek robot tarafından çıkarılan haritası

● Çalışma zemini patinaj ve kayma yaptıracak biçimde seçildiğinde (ortalaması 0 ve standart sapması 5% olan normal dağılıma uyacak biçimde rastgele olarak patinaj veya kayma) ortam 1 dakika 27 saniyede dolaşılmış ve Şekil 5.6'daki sonuç elde edilmiştir.



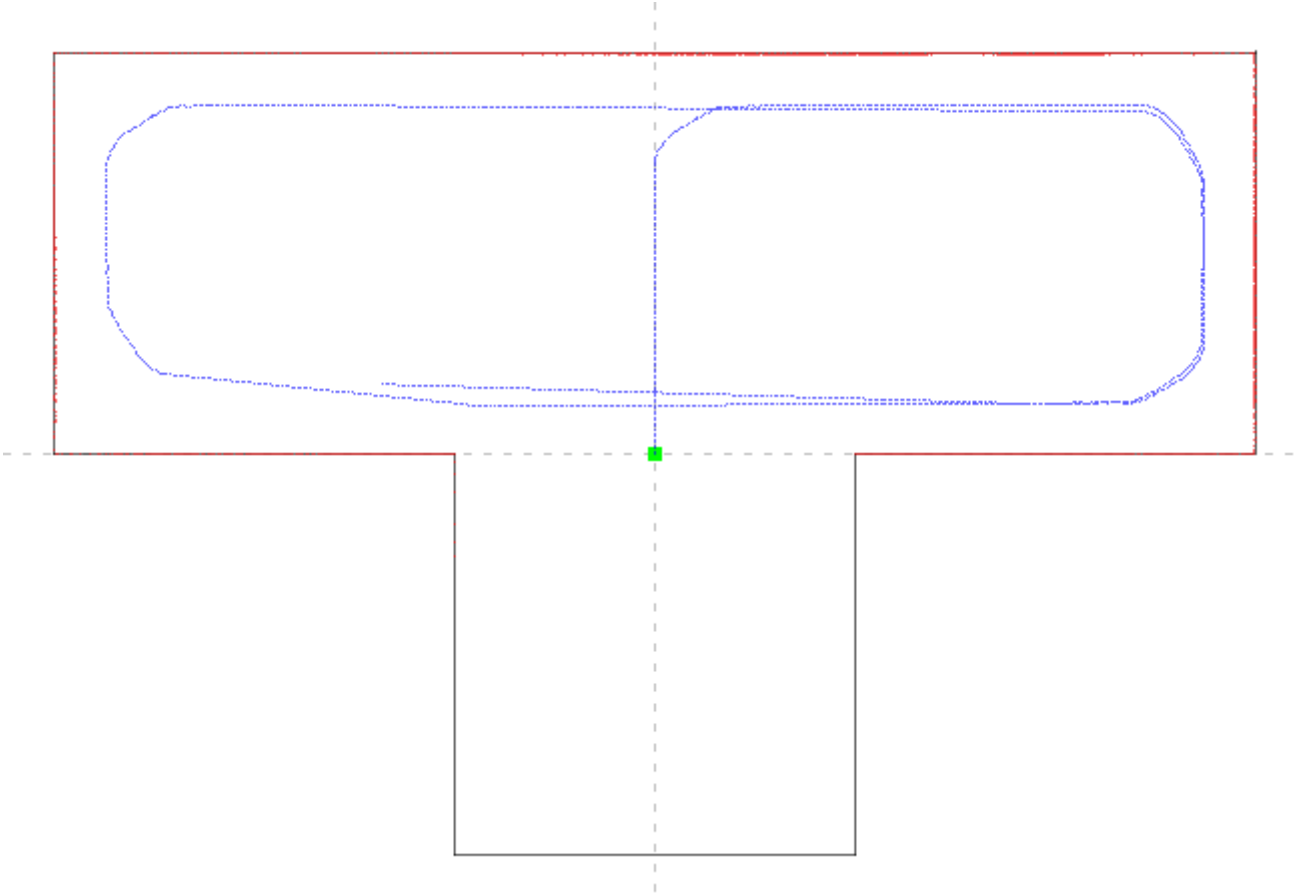
Şekil 5.6 Kare şeklindeki bir ortamın kaygan bir zeminde engelden sakınma hareketi ile tek robot tarafından çıkarılan haritası

● Duyargalara (ortalaması 0 ve standart sapması 2 cm olan normal dağılıma uyacak biçimde rastgele olarak), tekerleğe ve döner kodlayıcıya (ortalaması 0 ve standart sapması 2° olan normal dağılıma uyacak biçimde rastgele olarak) gürültü eklendiğinde ve kaygan bir zeminde (ortalaması 0 ve standart sapması 5% olan normal dağılıma uyacak biçimde rastgele olarak patinaj veya kayma) ortam 1 dakika 22 saniyede dolaşılmış ve Şekil 5.7'deki sonuç elde edilmiştir.



Şekil 5.7 Kare şeklindeki bir ortamın robot aygıtlarına gürültü eklenerek ve kaygan bir zeminde engelden sakınma hareketi ile tek robot tarafından çıkarılan haritası

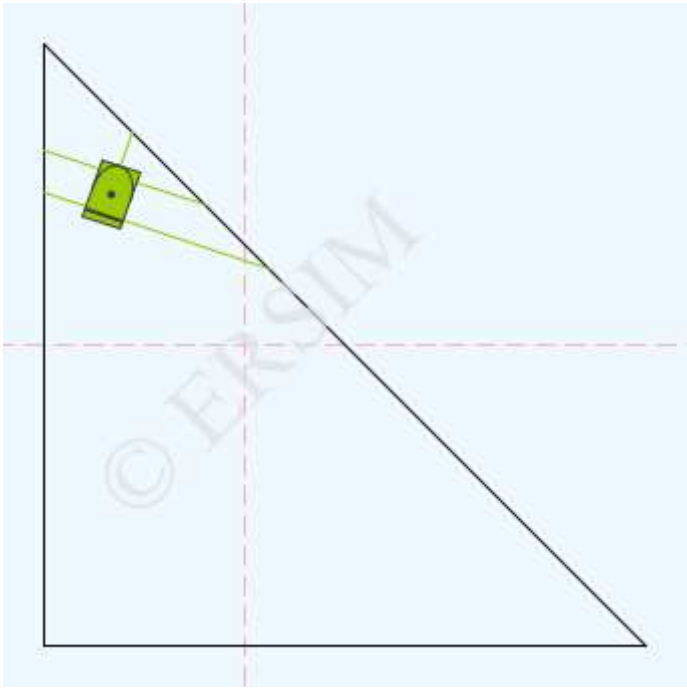
Gezgin robot kısa kenarları 2 m, uzun kenarı 6 m olan T şeklindeki bir ortamda, ideal şartlarda ve alanın merkezinden 90 derece eğim ile ilk hareketine başlatılarak ortam dolaştırıldığında Şekil 5.8'deki sonuç elde edilmiştir.



Şekil 5.8 T şeklindeki bir ortamın ideal şartlarda engelden sakınma hareketi ile tek robot tarafından çıkarılan haritası

Deneme sonucunda robotun ortamın sadece üst kısmını gezdiği, alt kısma hiç gitmediği görülmektedir. Bu durumun sebebi robota engelden sakınma hareketi yaptırılması ve ortamın alt kesimlerine gitmesi kararını aldıracak bir durumla karşılaşılmamasıdır.

Gezgin robotla ayrıca eşit kenarları 3 m olan ikizkenar dik üçgen şeklindeki bir ortamda deneme yapılmıştır. Robot üçgenin dik köşesinden ilk hareketine başlatılarak ortam dolaştırıldığında, robotun hipotenüs kenarına eşik değeri kadar yaklaştığında dönme hareketine başlatıldığı, ancak dönme hareketinin tamamlanamayacağı bir durumla karşılaşıldığı için (duvarın fazla eğimli olması sebebiyle) durma kararının alındığı görülmektedir (Şekil 5.9).



Şekil 5.9 Robotun ikizkenar dik üçgen şeklinde bir ortamda engelden sakınma hareketi ile gösterdiği davranış

5.2.1.3 Engel Takip Etme Algoritması ile Robot Kontrolü

Bir önceki yöntemin eksikliklerini gidermek ve daha iyi bir kontrol mekanizması uygulamak amacıyla robotun kontrolü için engel takip etme algoritması kullanılmıştır. Bu yöntemde robot ilk hareketine başladıktan sonra ortamda bir engel bulmasına çalışılmakta, engel bulunduktan sonrada robotun konumu bu engele paralel hale getirilerek, robotun bulunan ve peşi sıra gelen engelleri takip ederek hareketine devam etmesi sağlanmaya çalışılmaktadır.

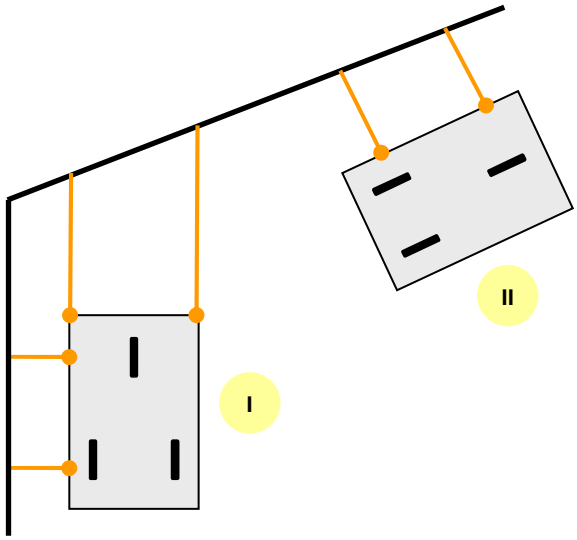
Kontrol edilen robot 10 ile 80 cm arasında ölçüm yapabilme kabiliyetine sahip, ön sağ ve sol köşelerde 1’er adet, sağ ve sol yanlarında 2’şer adet ve arka ortada 1 adet olmak üzere toplam 7 adet mesafe duyargasına sahiptir. Ayrıca ortamdaki engellerin doğrusal yapıda oldukları kabul edilmektedir.

Kullanılan algoritmanın adımları şu şekildedir:

- Robotun başlangıç konumuna bir değer atanır.
- Ortamda bir engel bulunup robot bu engele paralel konuma getirilir.
- Sonraki adımlar sürekli olarak veya başlangıç konumuna dönene kadar tekrarlanır
  - Robot aygıtlarının ölçüm değerleri alınır.
  - Önceki konum, alınan yol miktarı ve tekerlek dönme ölçüm verileri kullanılarak kinematik denklemler ile robotun yeni konumu hesaplanır.
  - Duyarga ölçüm değerleri ile ortamdaki nesnelerin konumları hesaplanır (harita çıkarılır).
  - İç köşe bulunmuş ise karşıdaki engelin eğimi hesaplanır ve robota dönme hareketi yaptırılarak bu engele paralel hale gelmesi sağlanır.
  - Dış köşe bulunmuş ise robot takip edilen duvar yönüne döndürülerek bulunacak ilk engele paralel hale getirilmeye çalışılır.
  - Hiçbir köşe tespit edilememiş ise robot takip edilen duvara paralel hale getirilmeye çalışılır.

Bu algoritmada robotun engel takip edebilmesi için yapabileceği 3 temel hareket bulunmaktadır. Bu hareketler, hangi durumda ve nasıl uygulanacakları şu şekildedir:

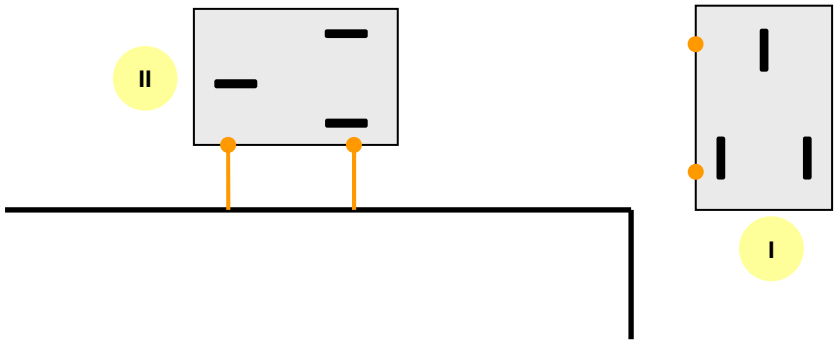
İç Köşe Bulunması Durumu: Robotun öndeki duyargalarından elde edilen ölçümlerde, belirlenmiş bir eşik değerinin altına düşüldüğünde iç köşe bulunduğuna karar verilir. Öndeki iki duyarganın ölçüm değerinden yararlanılarak karşıdaki engelin eğim değeri tespit edilir ve robotun eğimi bulunan bu eğim değeri oluncaya kadar takip edilen duvarın tersi yönünde dönme hareketi yaptırılarak karşıdaki engele paralel hale gelinmiş olunur. Ayrıca robot dönmesini tek harekette yapamayacaksa, robot bir miktar geri yönlendirilerek dönme hareketinin tamamlanması için gerekli açı elde edilmiş olunur.



Şekil 5.10 Robot için iç köşe bulunması durumunda yaptırılacak hareket

Böyle bir durumun gösterildiği Şekil 5.10'da sol duvarı takip eden robot için iç köşe tespitinde bulunulmakta, karşı duvarın eğimi hesaplandıktan sonra bu eğim değerini elde edinceye kadar robot sağa doğru döndürülmektedir.

Dış Köşe Bulunması Durumu: Robotun takip ettiği duvara bakan kenarındaki iki duyargasından da ölçüm alınamadığında dış köşe bulunduğuna karar verilmektedir. Bu durumda takip edilen duvar yönünde, robotun bu kenarındaki duyargalarından ölçüm alınıncaya dek robota dönme hareketi yaptırılmaktadır. Böylece takip edilen engelden sonra gelen engele paralel hale gelinmiş olunmaktadır.

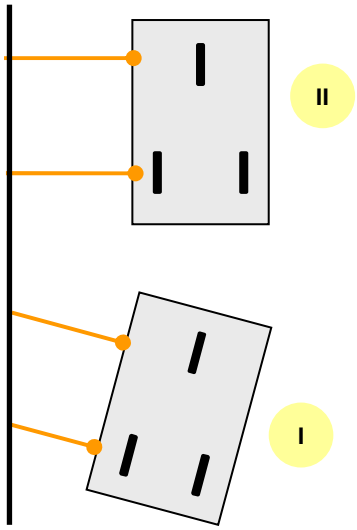


Şekil 5.11 Robot için dış köşe bulunması durumunda yaptırılacak hareket

Böyle bir durumun gösterildiği Şekil 5.11'de sol tarafındaki duvarı takip eden robot için, I konumunda iken sol kenar duyargalarından ölçüm alınamadığında dış köşe bulunduğuna karar verilmekte ve bu kenardaki duyargalardan ölçüm alınıncaya dek sola dönme hareketi yaptırılarak II konumuna getirilmektedir.

Engele Paralel Hale Gelme Durumu: Robot duvar takip ediyor iken, iç veya dış köşe bulunamamış ise robotun takip ettiği duvara paralel hale getirilmesine çalışılmaktadır. Bu durumda takip edilen duvar yönündeki kenarın ön tarafında bulunan duyarga, o kenarın arka tarafında bulunan duyargaya göre daha fazla ölçüm yapmakta ise engelden uzaklaşıldığına karar verilip takip edilen duvar yönünde, aksi durumda engele yaklaşıldığına karar verilip takip edilen duvarın tersi yönde dönme hareketi yaptırılarak takip edilen engele paralel olma işlemi yerine getirilmeye çalışılmaktadır.

Böyle bir durumun gösterildiği Şekil 5.12'de sol duvarı takip etmekte olan robot I konumunda iken ölçüm değerleri alınmakta, sol-ön duyarganın sol-arka duyargaya göre daha fazla ölçüm yaptığı tespit edilerek engelden uzaklaşıldığına karar verilip sola dönme hareketi yaptırılmaktadır. Böylece robotun II konumundaki gibi duvara paralel hale getirilmesine çalışılmaktadır.



Şekil 5.12 Robotun takip ettiği engele paralel hale getirilmesi için yaptırılacak hareket

Harita çıkarılırken engeller noktalar şeklinde tutulmak yerine başlangıç ve bitiş noktaları bilinen doğru parçaları halinde değerlendirileceklerdir. Ortamdaki engellerin doğrusal nitelikte olduğu varsayımından yola çıkılarak engellerin hafızada doğru parçaları şeklinde tutularak hem daha iyi bir haritanın çıkarılması hem de bellek alanından tasarruf edilmesi sağlanılmaya çalışılmıştır. Bu doğru parçaları bulunurken ilk olarak engel üzerinde tespit edilen noktalardan geçecek bir doğru denklemi oluşturulmuş daha sonrada bu denklemi doğrulayacak olan noktalarla doğru parçasının başlangıç ve bitiş noktaları bulunmaya çalışılmıştır.

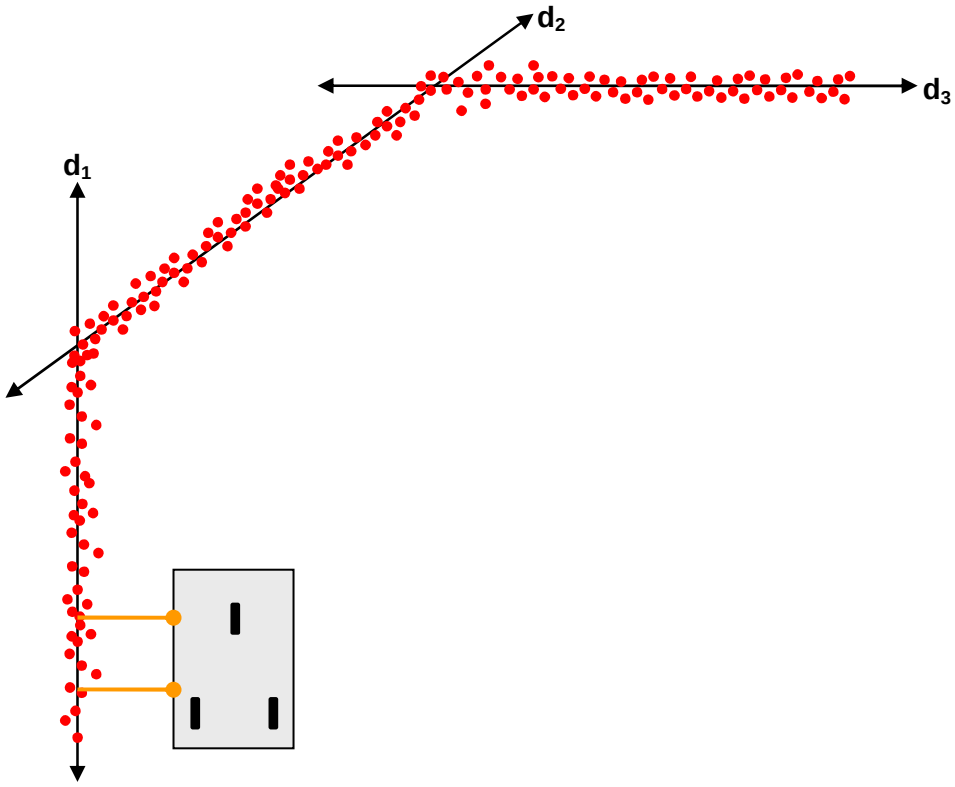
Bu işlem sırasında ilk olarak takip edilen engel üzerinde duyarga ölçüm sonuçları kullanılarak *k* adet nokta tespit edilecektir. Daha sonra en küçük kareler (*least squares*) yöntemi kullanılarak bu noktalardan geçen doğrunun denklemi hesaplanacaktır. Bundan sonraki amaç, denklemi hesaplanmış olan doğrusal engelin ne zamana kadar takip edildiğinin bulunmasıdır. Bunun için sonraki adımlarda, takip edilen engel üzerinde yine *k* adet nokta tespit edilecek ve tespit edilen bu noktaların takip edildiği varsayılan ve denklemi hesaplanmış olunan doğrusal engel üzerinde olup olmadıkları kontrol edilecektir. Bu kontrol işlemi için tespit edilen noktaların, denklemi bilinen doğruya olan uzaklıkları hesaplanacak, eğer bu uzaklık belirlenen bir eşik değerinden (*thr*) daha az ise bu noktaların doğru üzerinde oldukları daha fazla ise olmadıkları kabul edilecektir. Sonuçta bu *k* adet noktanın yarısı veya daha fazlası doğru üzerinde ise doğrusal engelin takip edildiğine karar verilip tekrar *k* adet nokta tespit edilerek doğrulama işlemi tekrar edilecek, doğrulamıyor ise yeni bir doğrusal engelin takibine

başlandığına karar verilip bu engel için yeni bir doğru denklemi bulunmaya çalışılacaktır.

Engel tespitinde kullanılan bu yöntemin adımları şu şekildedir:

1) Robot doğrusal bir engeli takip etmeye başlar.
2) Robotun takip ettiği doğrusal engel üzerinde *k* adet nokta tespit edilir.
3) *k* adet noktadan geçecek doğru denklemi en küçük kareler yöntemi kullanılarak bulunur.
4) Takip edilen engel üzerinde *k* adet nokta tespit edilir.
5) Bu noktaların kaç tanesinin denklemi bulunan doğru üzerinde olduğu hesaplanır.
6) Eğer *k* değerinin yarısı kadar veya daha fazla nokta doğru üzerinde ise 4 adımına, değilse 2 adımına dönülür.

Örneğin Şekil 5.13'deki gezgin robot sol tarafında engel takip etmeye başlayarak kırmızı noktaları tespit etmektedir. İlk olarak d_1 doğrusu tespit edilecek, bulunan noktalar d_1'i doğrulamadığında d_2 doğrusu bulunacak ve yine aynı şekilde bu doğruda doğrulanmadığında d_3 doğrusunun denklemi bulunacaktır.

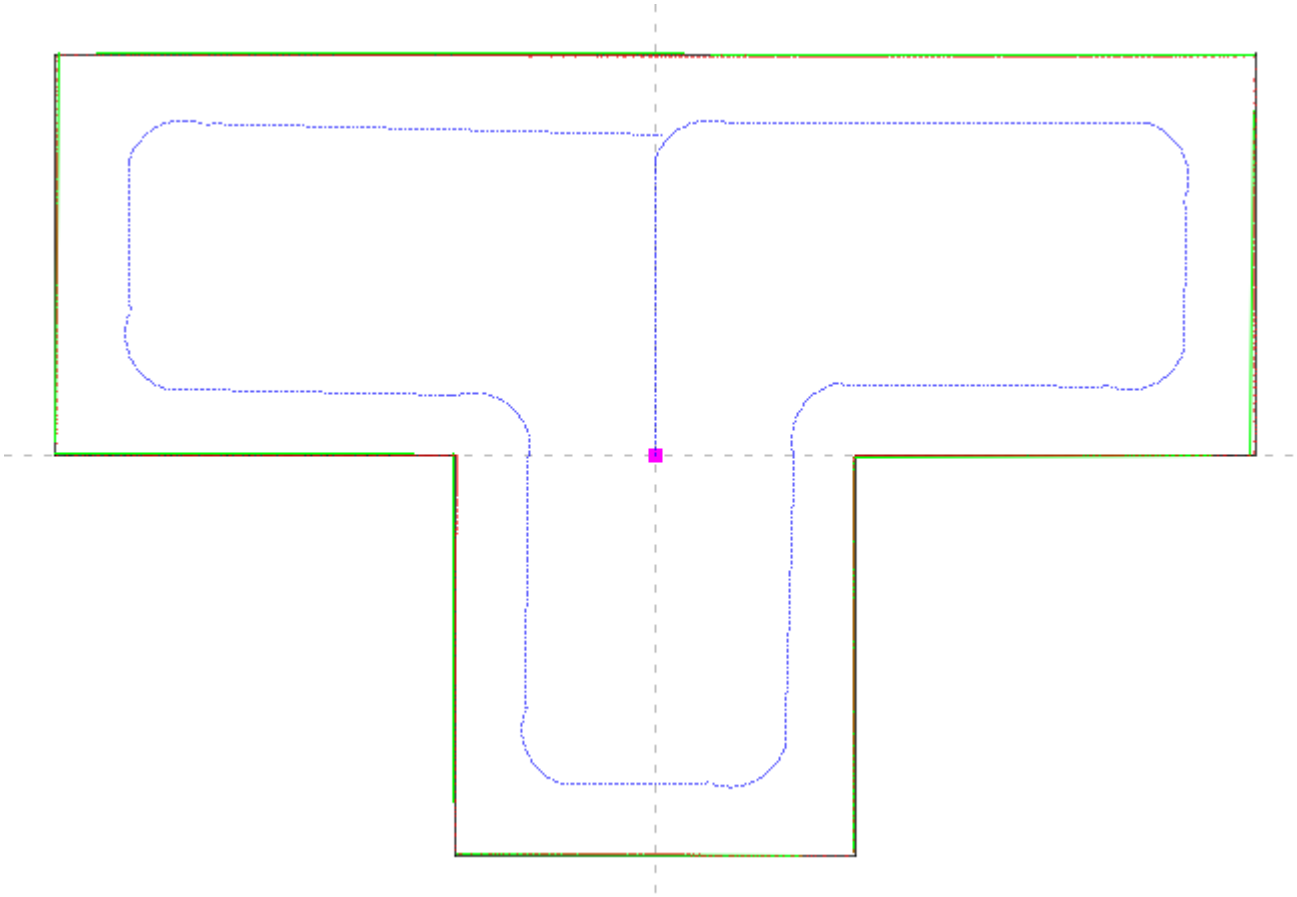


Şekil 5.13 Gezgin robotun tespit ettiği noktaların doğru denklemi ile tanımlanması

5.2.1.4 Engel Takip Etme Algoritması ile Elde Edilen Sonuçlar

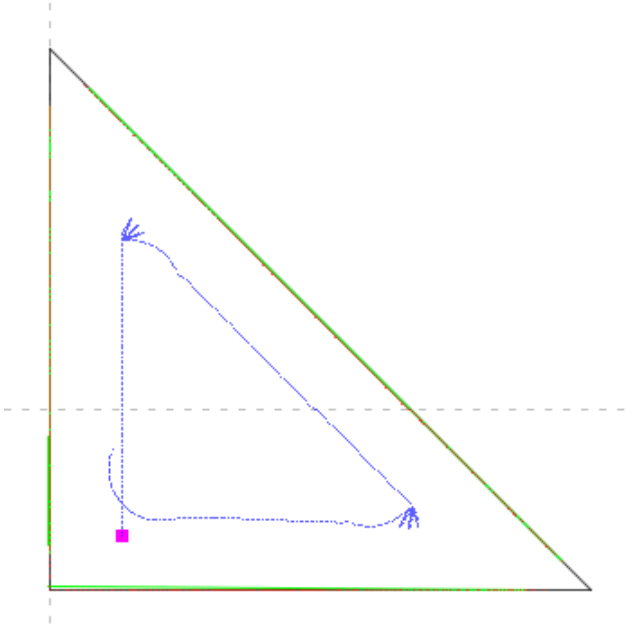
Bu bölümde engel takip etme algoritması ile robot kontrolünün farklı ortamlarda ve farklı parametreler ile denenerek elde edilen harita sonuçları incelenecektir. Gösterilen haritalarda kesikli çizgiler koordinat eksenini, siyah çizgiler gerçek engelleri, kırmızı noktalar ortamdaki engeller üzerinde tespit edilen noktaları, yeşil çizgiler tahmin edilen doğrusal engelleri, mavi noktalar robotun tahmin edilen yörüngesini, mor renkli kare ise robotun hareketine başladığı ilk konumu göstermektedir.

Önceki yöntem ile tamamı dolaşılamayan kısa kenarları 2 m, uzun kenarı 6 m olan T seklindeki ortam, ideal şartlarda ve alanın merkezinden 90 derece eğim ile ilk harekete başlanarak robot tarafından dolaşıldığında Şekil 5.14'deki sonuç elde edilmiştir (k=6, thr=2cm).



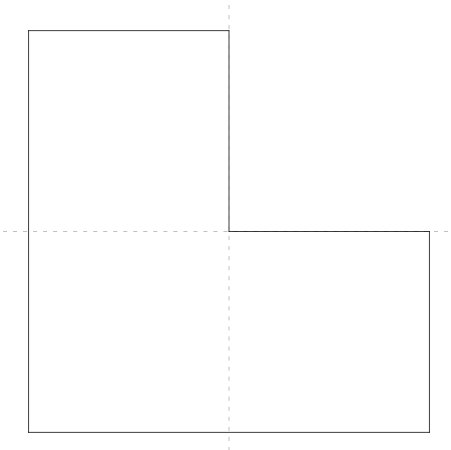
Şekil 5.14 T şeklindeki bir ortamın ideal şartlarda engel takip etme algoritması ile tek robot tarafından çıkarılan haritası

Yine önceki yöntem ile dolaşılamayan eşit kenarları 3 m olan ikizkenar dik üçgen şeklindeki ortam, ideal şartlarda ve üçgenin dik köşesinden ilk harekete başlanarak robot tarafından dolaşıldığında Şekil 5.15'deki sonuç elde edilmiştir (k=6, thr=2cm).



Şekil 5.15 İkizkenar dik üçgen şeklindeki bir ortamın ideal şartlarda engel takip etme algoritması ile tek robot tarafından çıkarılan haritası

Kısa kenarları 2 m ve uzun kenarları 4 m olan L şeklindeki (Şekil 5.16) bir ortam üzerinde, farklı koşullarda ve engel takip etme algoritması farklı parametrelerle kullanılarak denemeler yapılmıştır. Bu denemeler sonucunda farklı haritalar bulunmuş ve haritalarda farklı sayılarda doğrusal engeller tespit edilmiştir. Bu farklılıklar deneme yapılan ortam koşullarına ve uygulanan algoritmadaki parametre değerlerine göre oluşmaktadır.

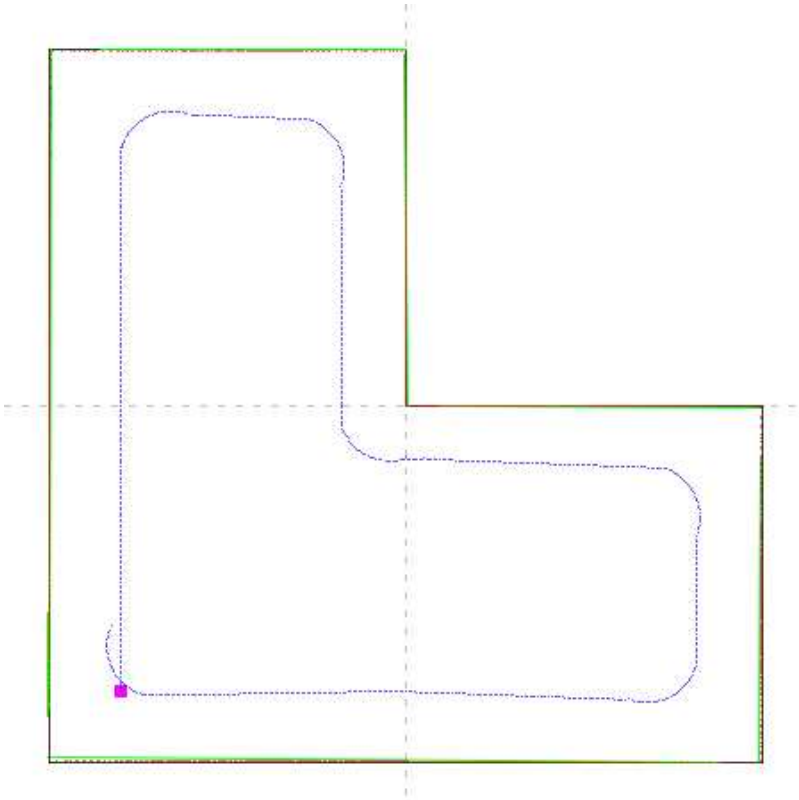


Şekil 5.16 L şeklindeki ortamın görüntüsü

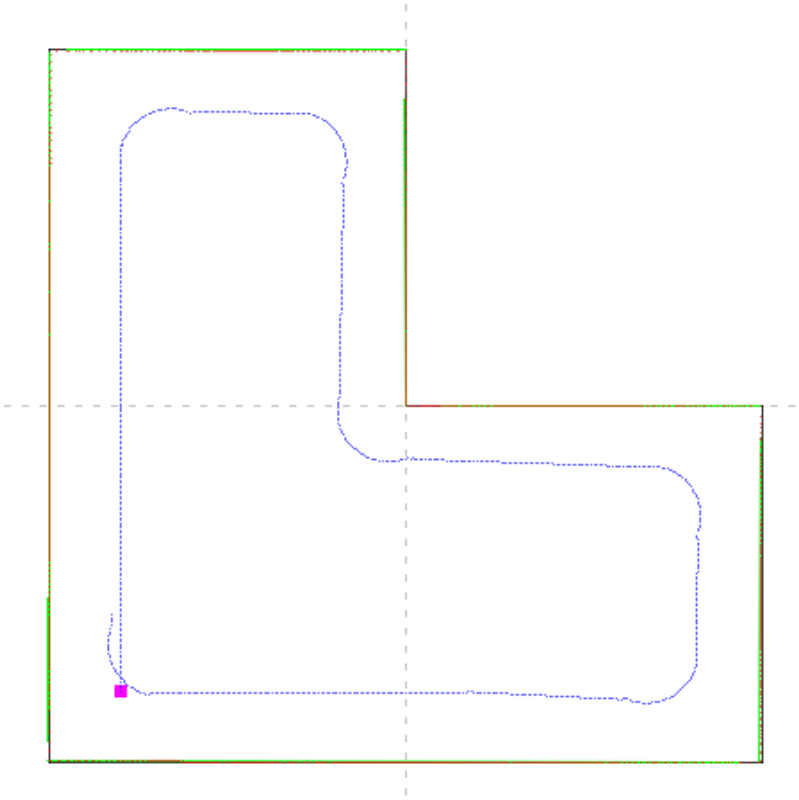
Elde edilen tüm sonuçlar Çizelge 5.1'de görülebilmektedir.

Çizelge 5.1 Engel takip etme algoritması ile farklı ortam koşulları ve algoritma parametreleri ile elde edilen sonuçlar

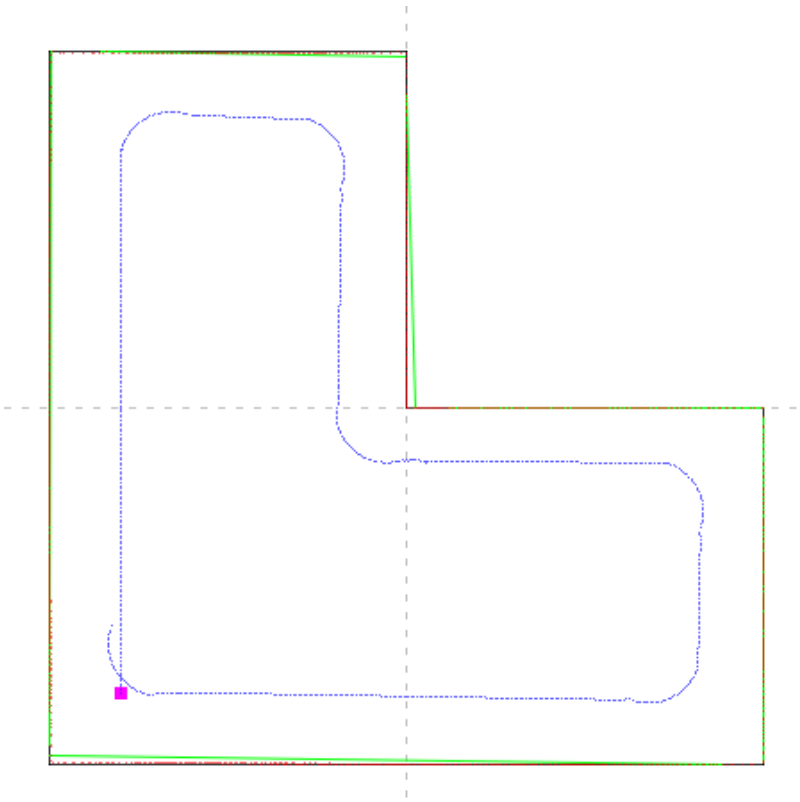
Deneme No	Duyarga Gürültüsü (cm)	Tekerlek Gürültüsü (°)	Döner Kodlayıcı Gürültüsü (°)	Zemin Tipi (%)	k	thr (cm)	Tespit Edilen Engel Sayısı	Elde Edilen Harita
1	-	-	-	ideal	6	2	7	Şekil 5.17
2	-	-	-	ideal	10	2	7	Şekil 5.18
3	-	-	-	ideal	6	5	7	Şekil 5.19
4	-	-	-	ideal	10	5	7	Şekil 5.20
5	Gaussian (μ=0, σ=2)	-	-	ideal	6	2	212	Şekil 5.21
6	Gaussian (μ=0, σ=2)	-	-	ideal	10	2	140	Şekil 5.22
7	Gaussian (μ=0, σ=2)	-	-	ideal	6	5	51	Şekil 5.23
8	Gaussian (μ=0, σ=2)	-	-	ideal	10	5	49	Şekil 5.24
9	-	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	ideal	6	2	14	Şekil 5.25
10	-	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	ideal	10	2	12	Şekil 5.26
11	-	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	ideal	6	5	9	Şekil 5.27
12	-	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	ideal	10	5	9	Şekil 5.28
13	-	-	-	kaygan Gaussian (μ=0, σ=5)	6	2	8	Şekil 5.29
14	-	-	-	kaygan Gaussian (μ=5, σ=5)	6	2	17	Şekil 5.30
15	-	-	-	kaygan Gaussian (μ=-5, σ=5)	6	2	13	Şekil 5.31
16	-	-	-	kaygan Gaussian (μ=0, σ=10)	6	2	8	Şekil 5.32
17	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	kaygan Gaussian (μ=0, σ=5)	6	2	164	Şekil 5.33
18	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	kaygan Gaussian (μ=0, σ=5)	10	2	131	Şekil 5.34
19	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	kaygan Gaussian (μ=0, σ=5)	6	5	45	Şekil 5.35
20	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	Gaussian (μ=0, σ=2)	kaygan Gaussian (μ=0, σ=5)	10	5	37	Şekil 5.36



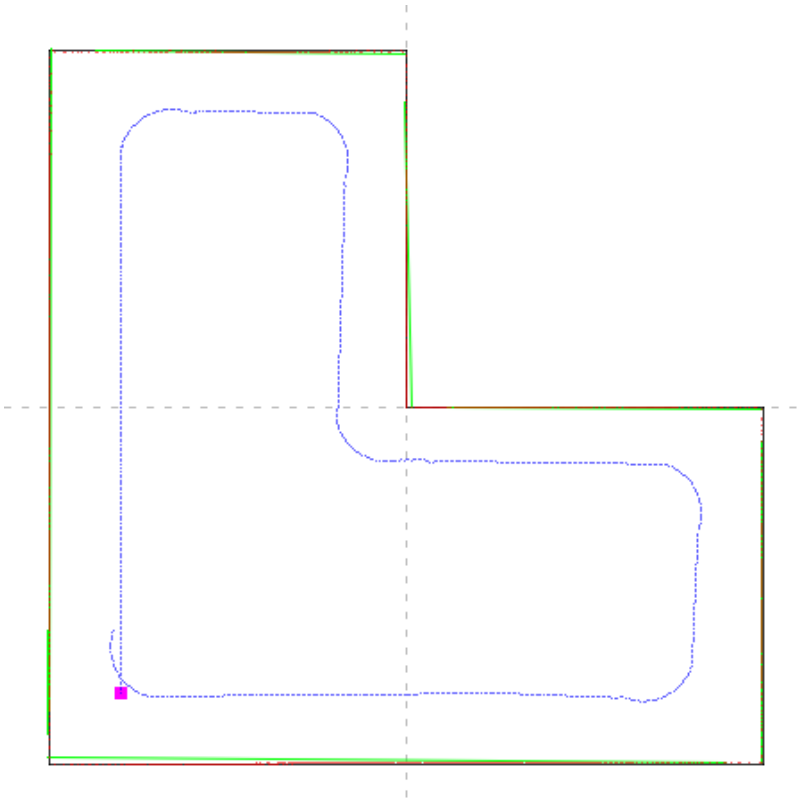
Şekil 5.17 Engel takip etme algoritması ile 1. denemede elde edilen harita



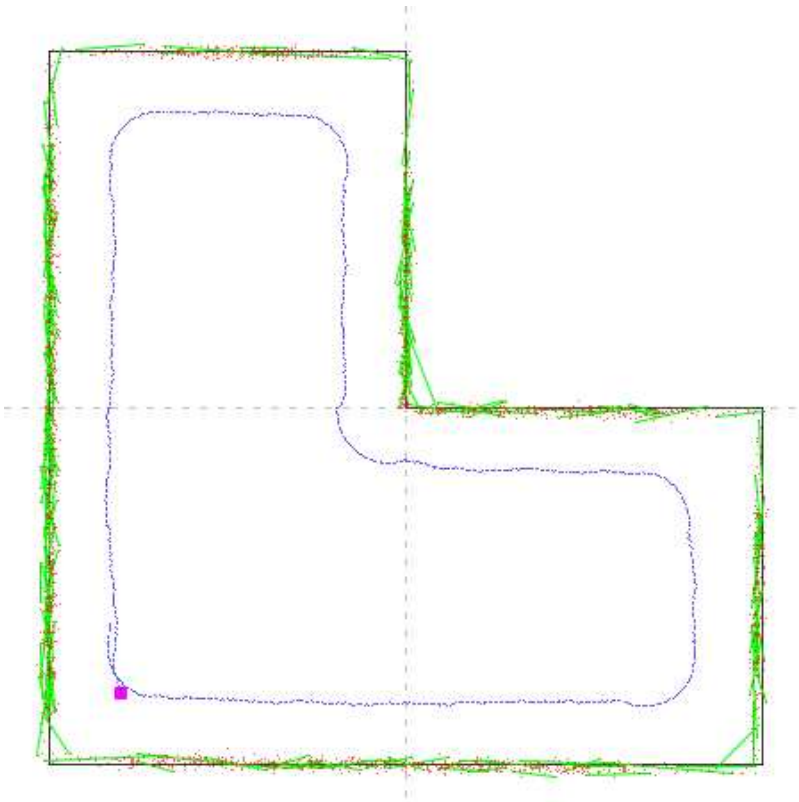
Şekil 5.18 Engel takip etme algoritması ile 2. denemede elde edilen harita



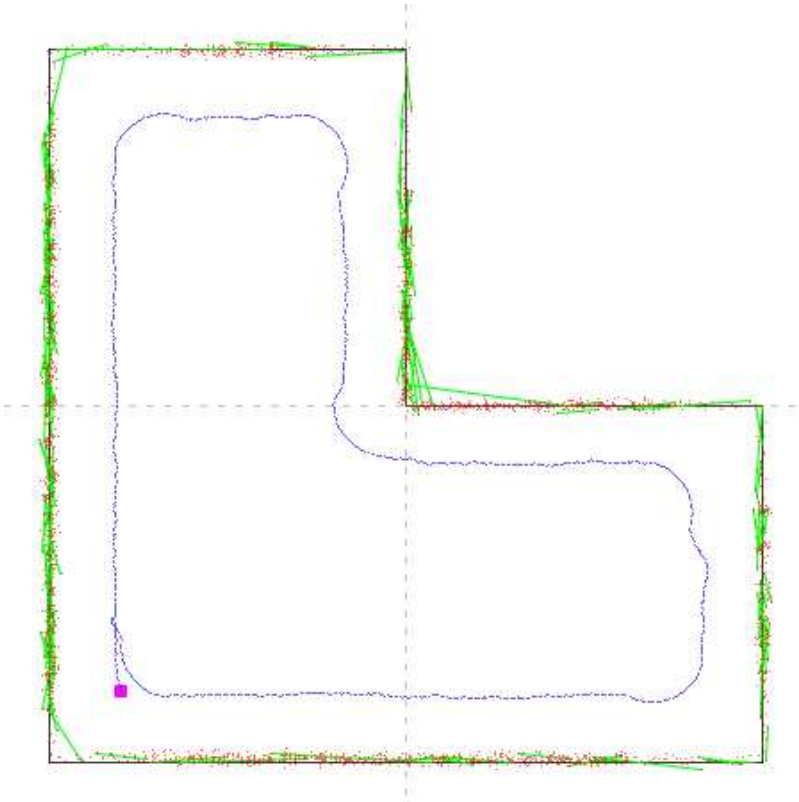
Şekil 5.19 Engel takip etme algoritması ile 3. denemede elde edilen harita



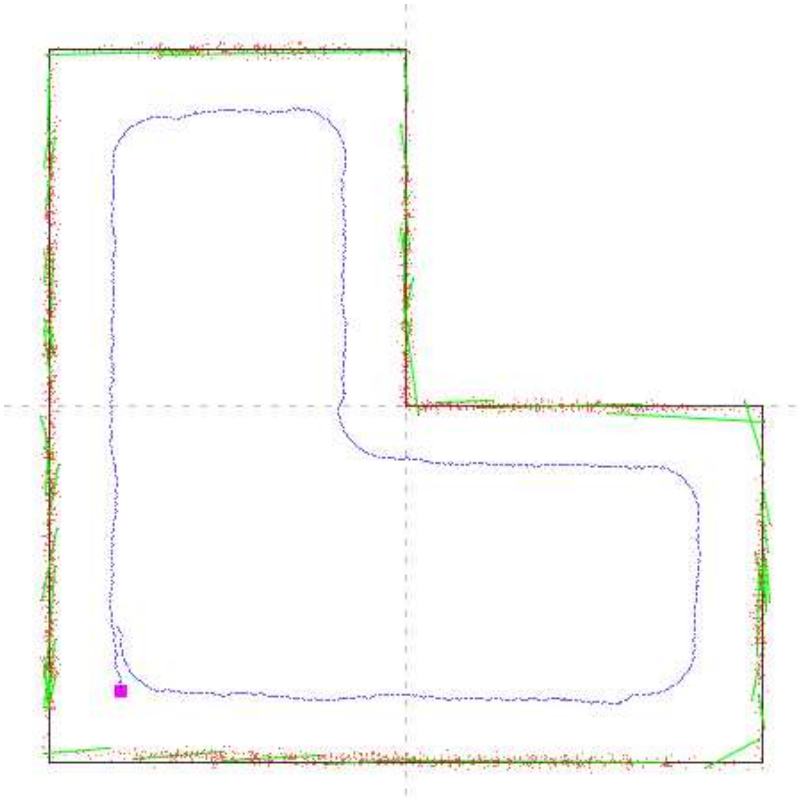
Şekil 5.20 Engel takip etme algoritması ile 4. denemede elde edilen harita



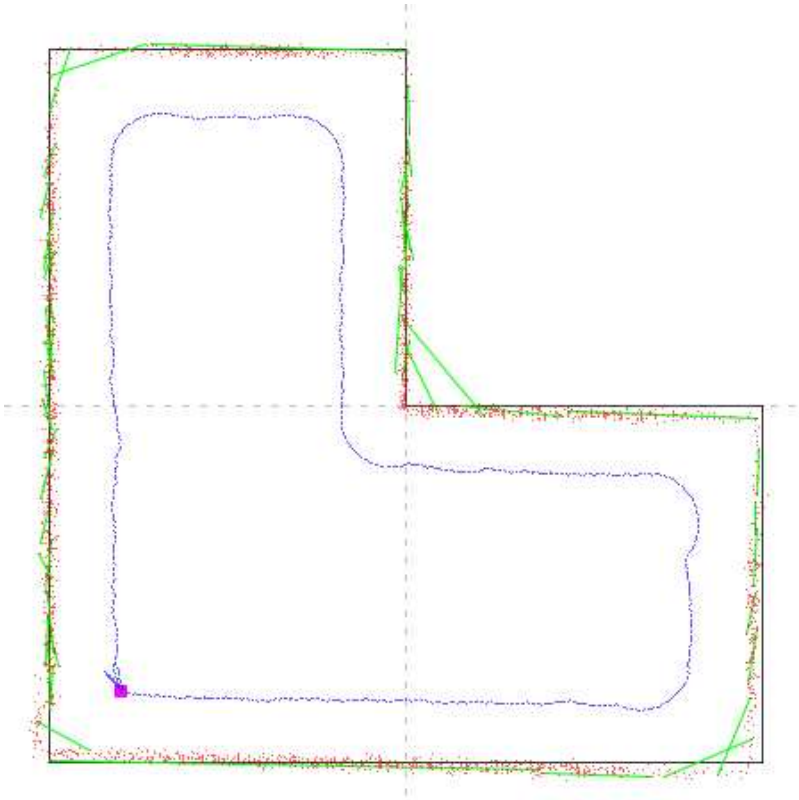
Şekil 5.21 Engel takip etme algoritması ile 5. denemede elde edilen harita



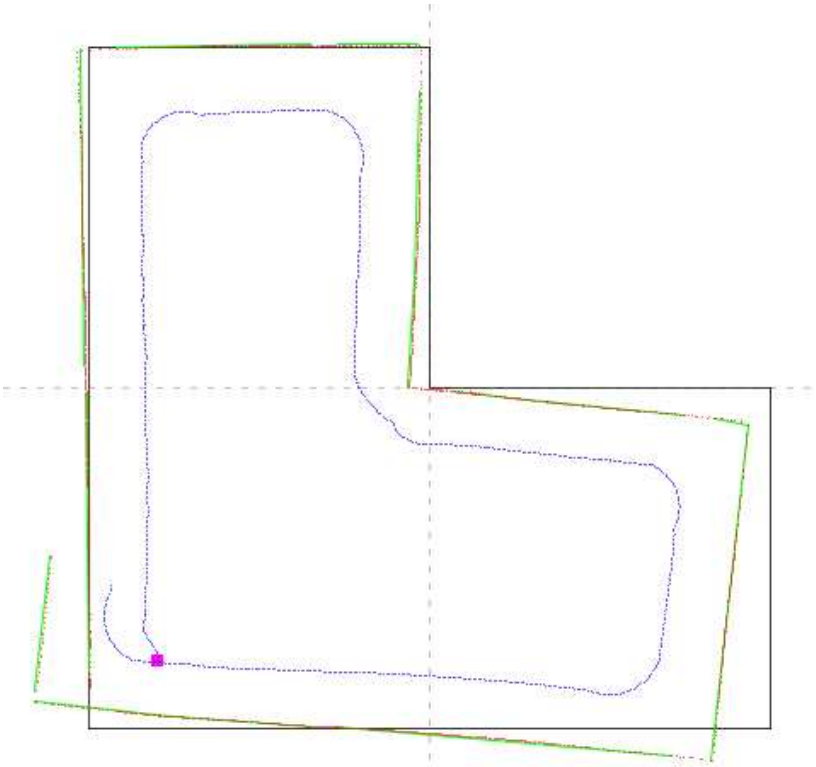
Şekil 5.22 Engel takip etme algoritması ile 6. denemede elde edilen harita



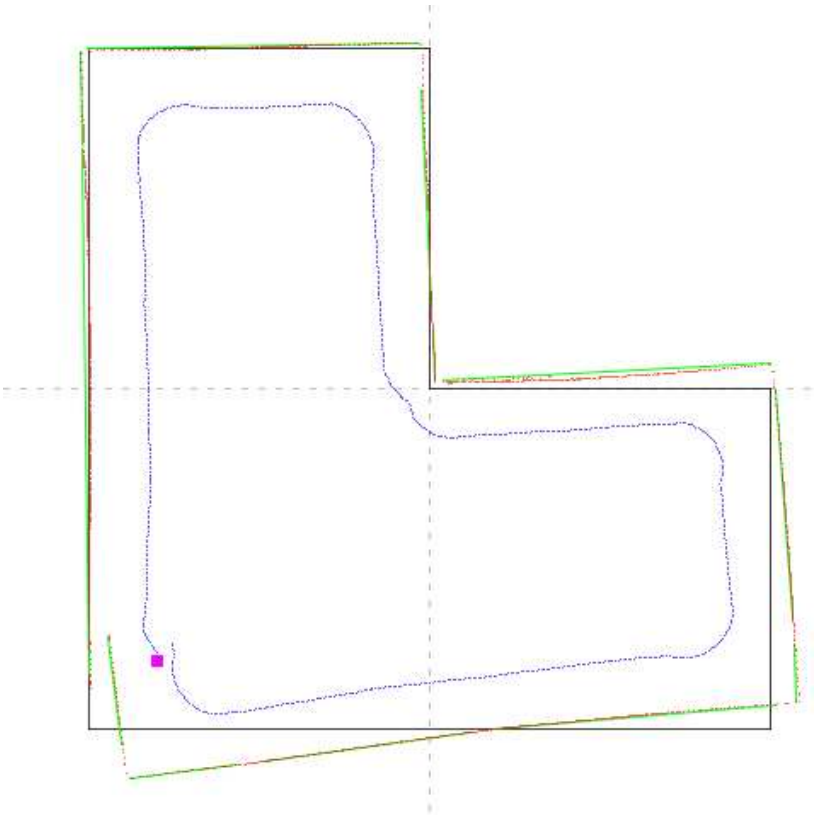
Şekil 5.23 Engel takip etme algoritması ile 7. denemede elde edilen harita



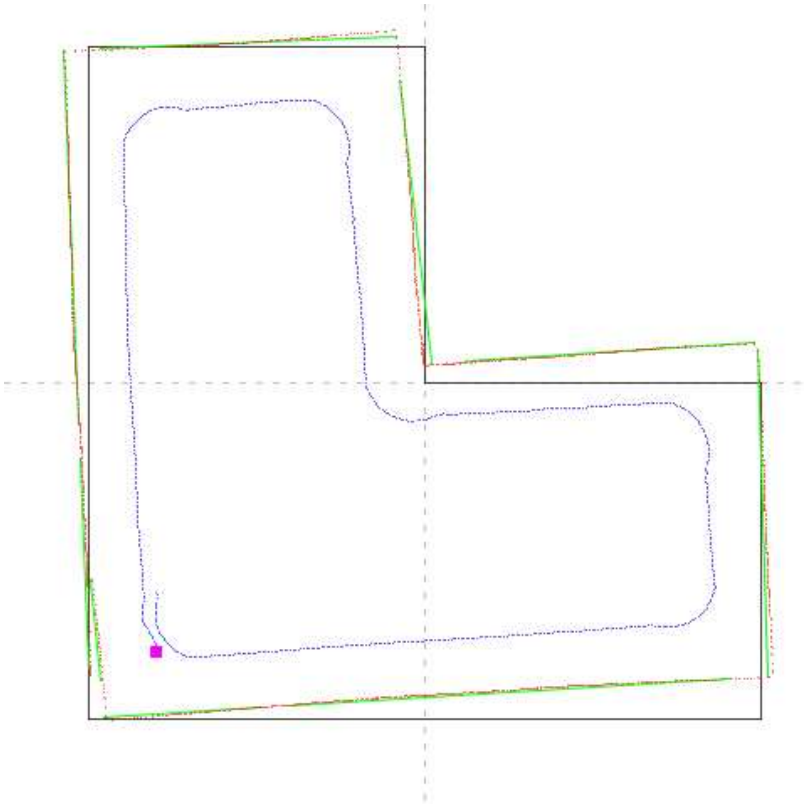
Şekil 5.24 Engel takip etme algoritması ile 8. denemede elde edilen harita



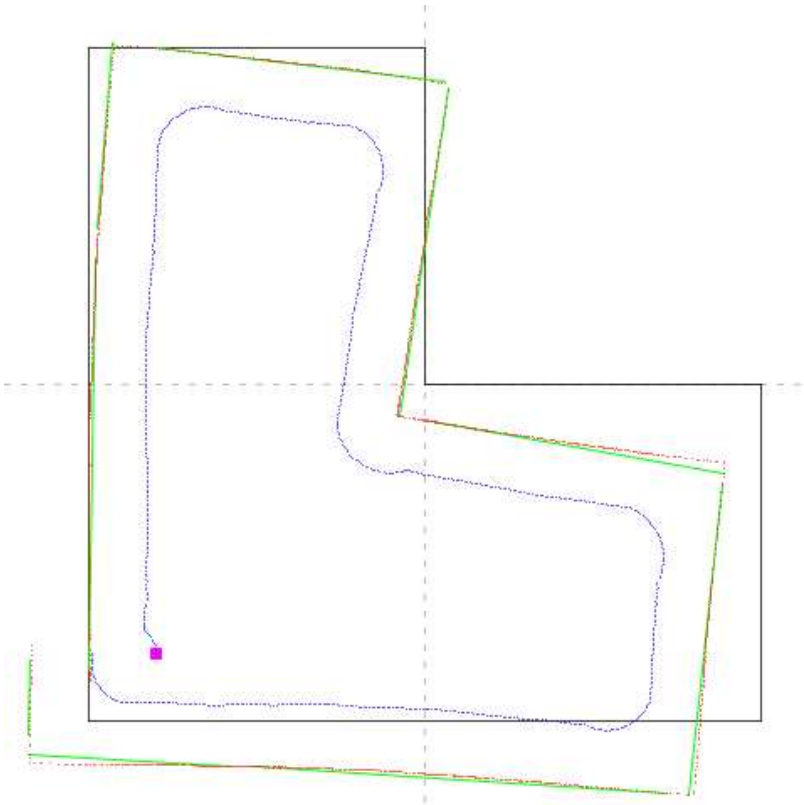
Şekil 5.25 Engel takip etme algoritması ile 9. denemede elde edilen harita



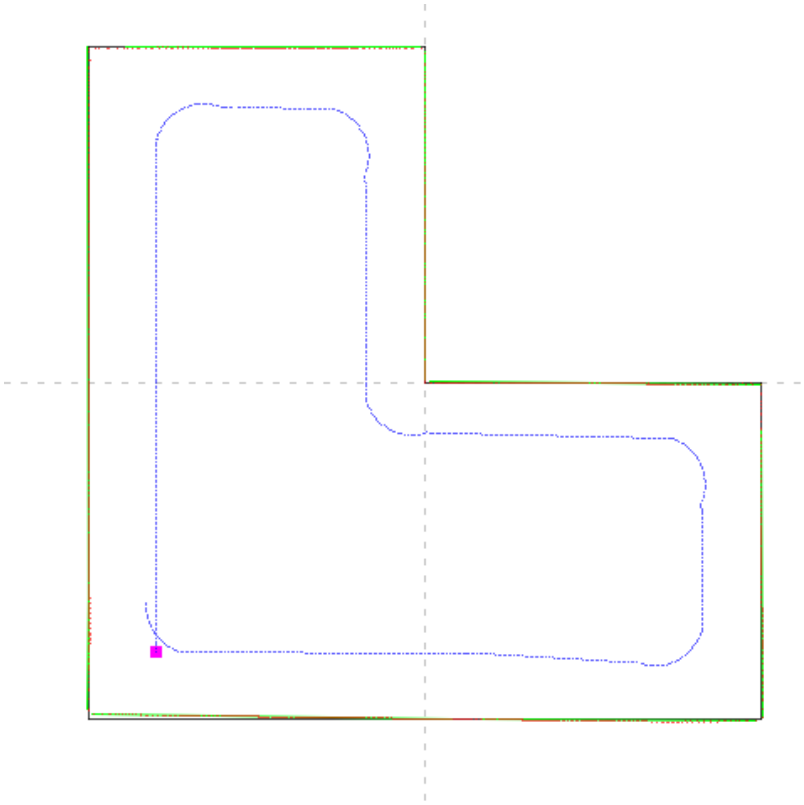
Şekil 5.26 Engel takip etme algoritması ile 10. denemede elde edilen harita



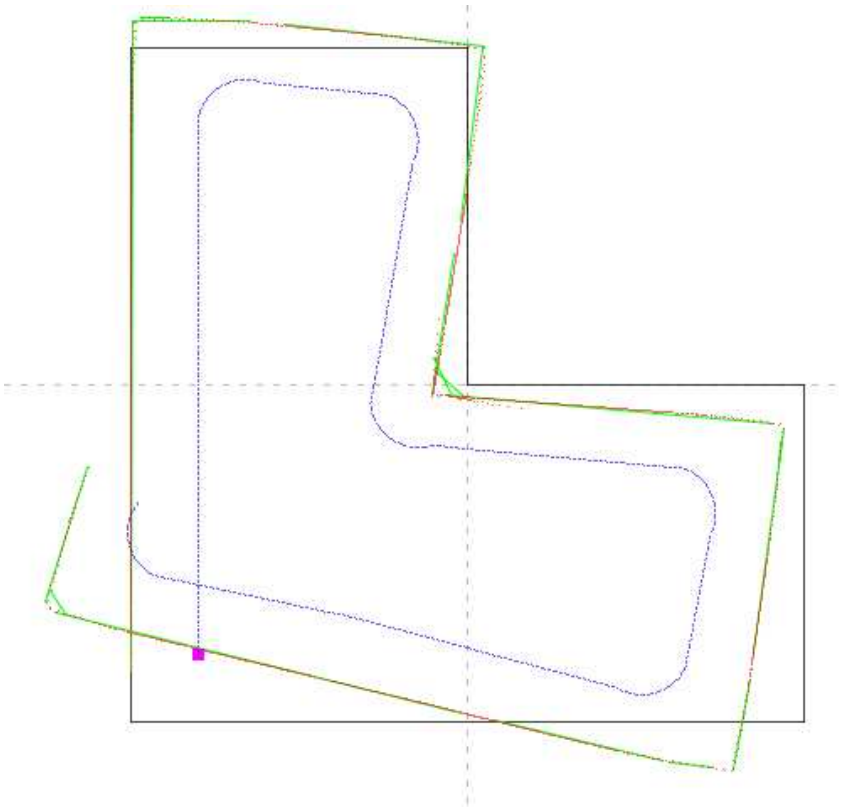
Şekil 5.27 Engel takip etme algoritması ile 11. denemede elde edilen harita



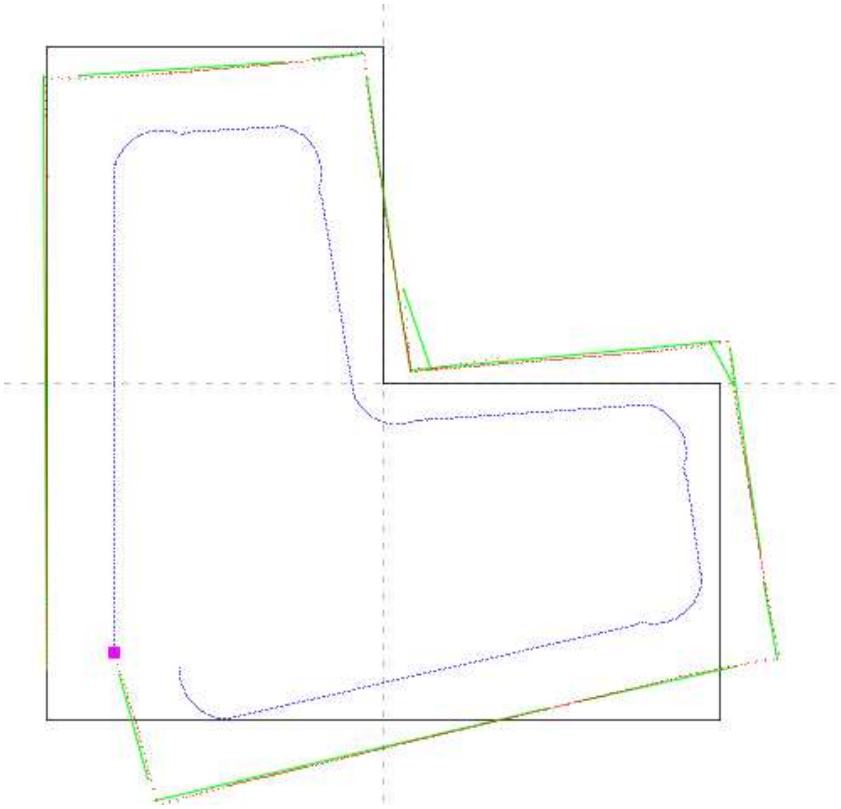
Şekil 5.28 Engel takip etme algoritması ile 12. denemede elde edilen harita



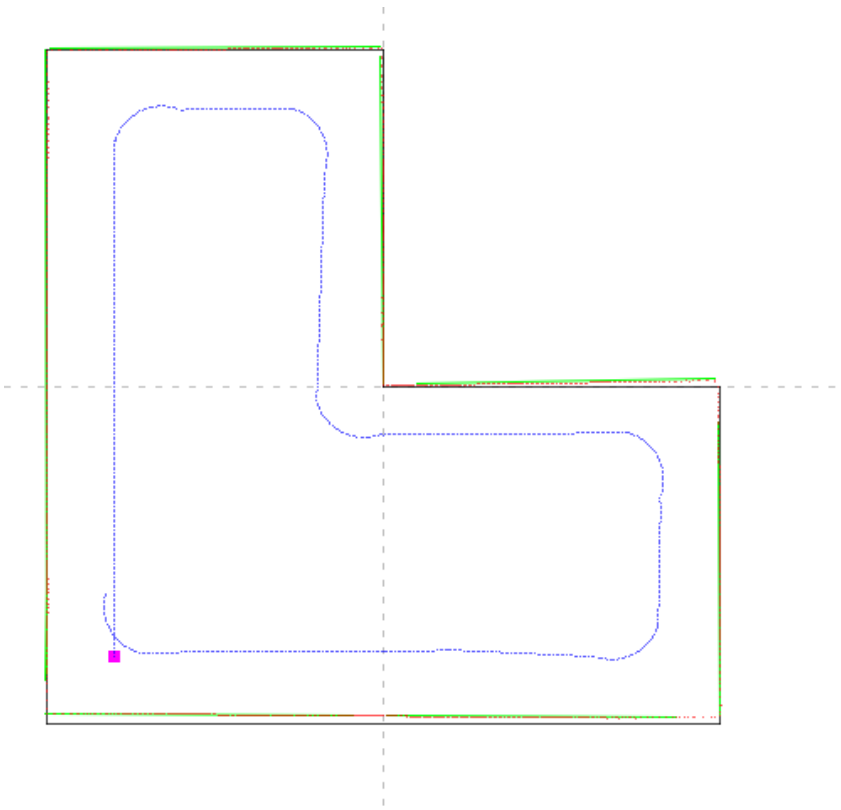
Şekil 5.29 Engel takip etme algoritması ile 13. denemede elde edilen harita



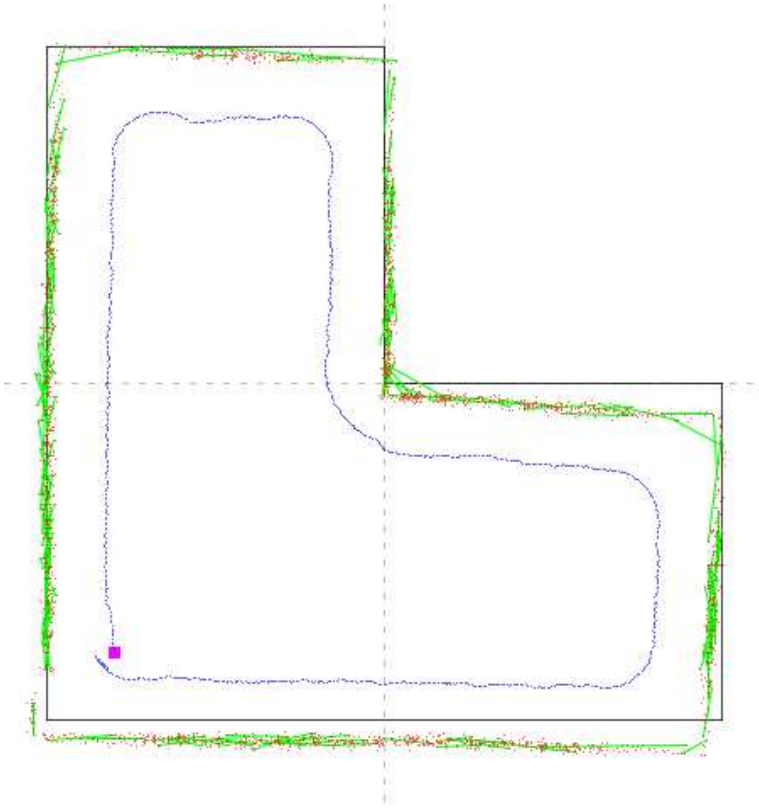
Şekil 5.30 Engel takip etme algoritması ile 14. denemede elde edilen harita



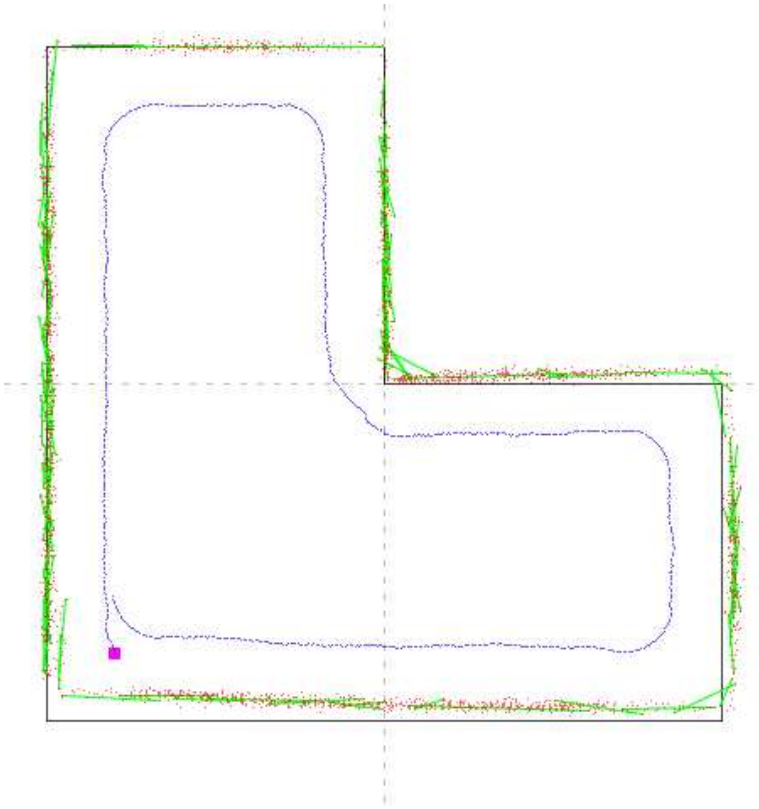
Şekil 5.31 Engel takip etme algoritması ile 15. denemede elde edilen harita



Şekil 5.32 Engel takip etme algoritması ile 16. denemede elde edilen harita

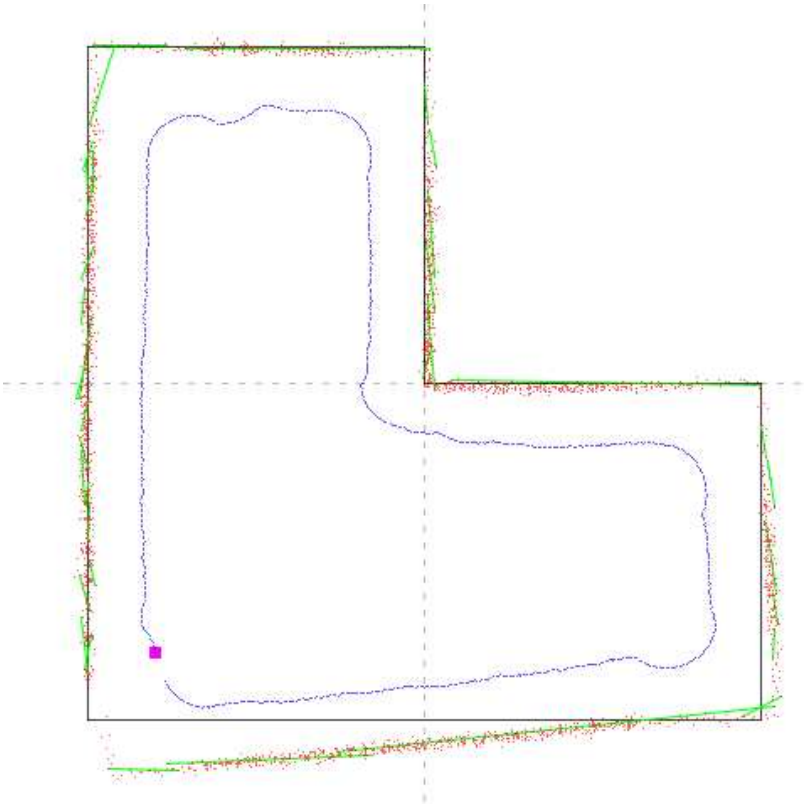


Şekil 5.33 Engel takip etme algoritması ile 17. denemede elde edilen harita



Şekil 5.34 Engel takip etme algoritması ile 18. denemede elde edilen harita

Şekil 5.35 Engel takip etme algoritması ile 19. denemede elde edilen harita



Şekil 5.36 Engel takip etme algoritması ile 20. denemede elde edilen harita

5.2.2 Çoklu Robot Kontrolü

SLAM uygulamalarında genel olarak tek robot kullanılarak nispeten başarılı sonuçlar elde edilebilmiştir. Birden fazla robotun kullanıldığı sistemler ise daha karmaşık bir kontrol yapısını gerektirmektedir. Böyle bir durumda robotların işbirliği içinde çalışmaları, ortamda durağan engellerin varlığının yanı sıra hareketli cisimlerinde olmasıyla (diğer robotlar) daha iyi bir kontrol algoritması ile yönlendirilmeleri ve her robotun oluşturduğu haritaların birlikte değerlendirilmeleri konuları gündeme gelmektedir.

Bu bölümde birden fazla robotun nasıl kontrol edileceğine dair temel oluşturulabilecek bir kontrol yapısı oluşturulmaya çalışılmıştır. Oluşturulan merkezi kontrolde her robot için ayrı bir çalışma kanalı oluşturulmuş ve robotun önceki bölümlerde anlatılan tek robot kontrolünde olduğu gibi hareket etmesi sağlanmıştır.

Çalışılan ortamdaki engellerin doğrusal nitelikte olduğu ve robotların başlangıç konumlarını bildikleri varsayılmıştır. Kontrol edilen robotlar 10 ile 80 cm arasında ölçüm yapabilme kabiliyetine sahip, ön sağ ve sol köşelerde 1'er adet, sağ ve sol yanlarında 2'şer adet ve arka ortada 1 adet olmak üzere toplam 7 adet mesafe duyargasına sahiptirler.

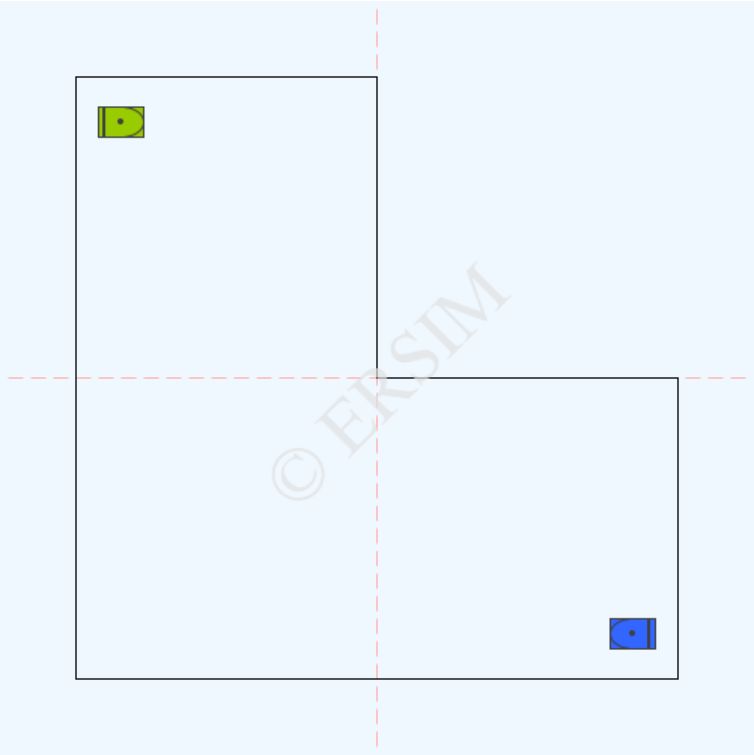
Robot kontrolünde Bölüm 5.2.1.3'de açıklanan engel takip etme algoritması kullanılacaktır. Engeller yine bu yöntemde olduğu gibi başlangıç ve bitiş noktaları bilinen doğru parçaları halinde tutulacaklardır ve tüm robotların elde ettikleri haritalar birleştirilip birlikte değerlendirilecektir.

5.2.2.1 Çoklu Robot Kontrolü ile Elde Edilen Sonuçlar

Bu bölümde çoklu robot kontrolünün farklı ortamlarda ve farklı parametreler ile denenerek elde edilen harita sonuçları incelenecektir. Gösterilen haritalarda kesikli çizgiler koordinat eksenini, siyah çizgiler gerçek engelleri, noktalar ortamdaki engeller üzerinde tespit edilen noktaları, çizgiler tahmin edilen doğrusal engelleri, kare ile başlayan noktalar ise robotun tahmin edilen yörüngesini göstermektedir.

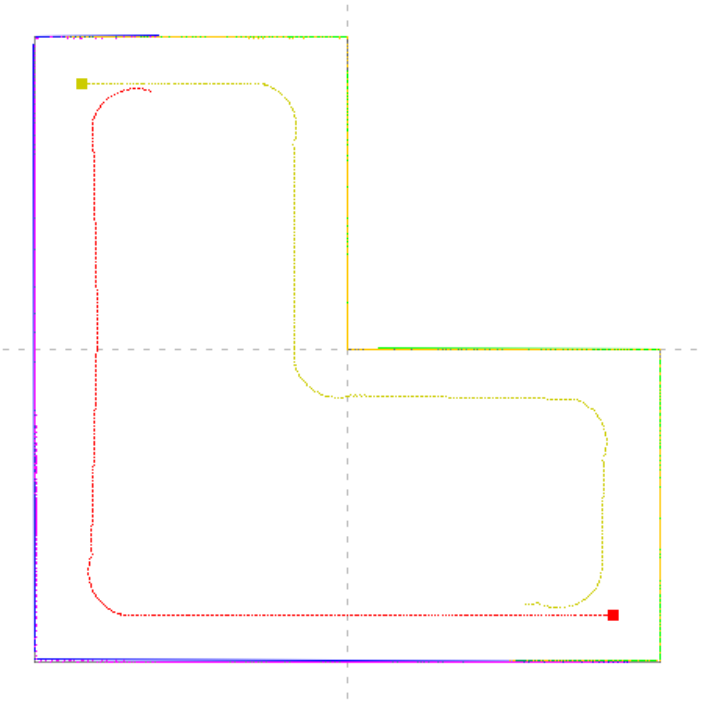
Robotlar kısa kenarları 2 m ve uzun kenarları 4 m olan L şeklindeki ortam üzerinde Şekil 5.37'deki ilk konumlarından hareketlerine başlatılarak denemeler yapılmıştır.

Yeşil renkli robotun elde ettiği sonuçlar yeşil, turuncu ve kahverengi renklerde; mavi renkli robotun elde ettiği sonuçlar mavi, mor ve kırmızı renklerde gösterilmiştir.



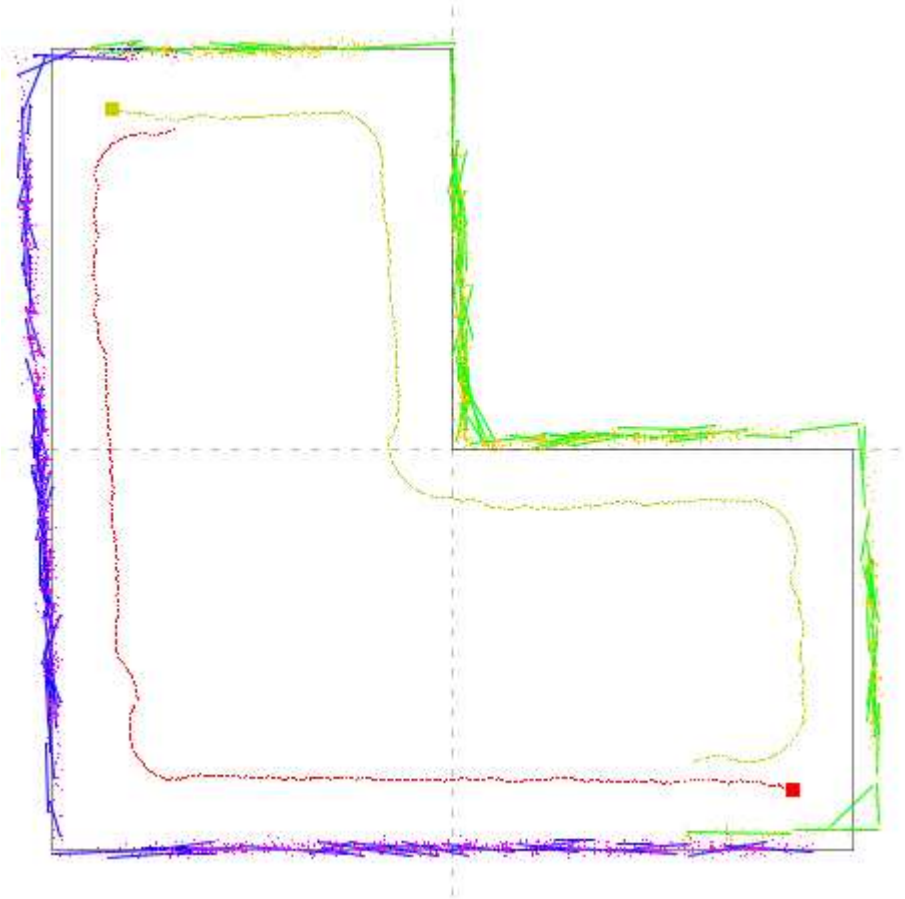
Şekil 5.37 L şeklindeki bir ortamda iki robotun hareketlerine başlamadan önceki konumları

● İdeal şartlarda ortamın iki robot tarafından çıkarılan haritası Şekil 5.38'deki gibidir (k=6 ve thr=2cm).



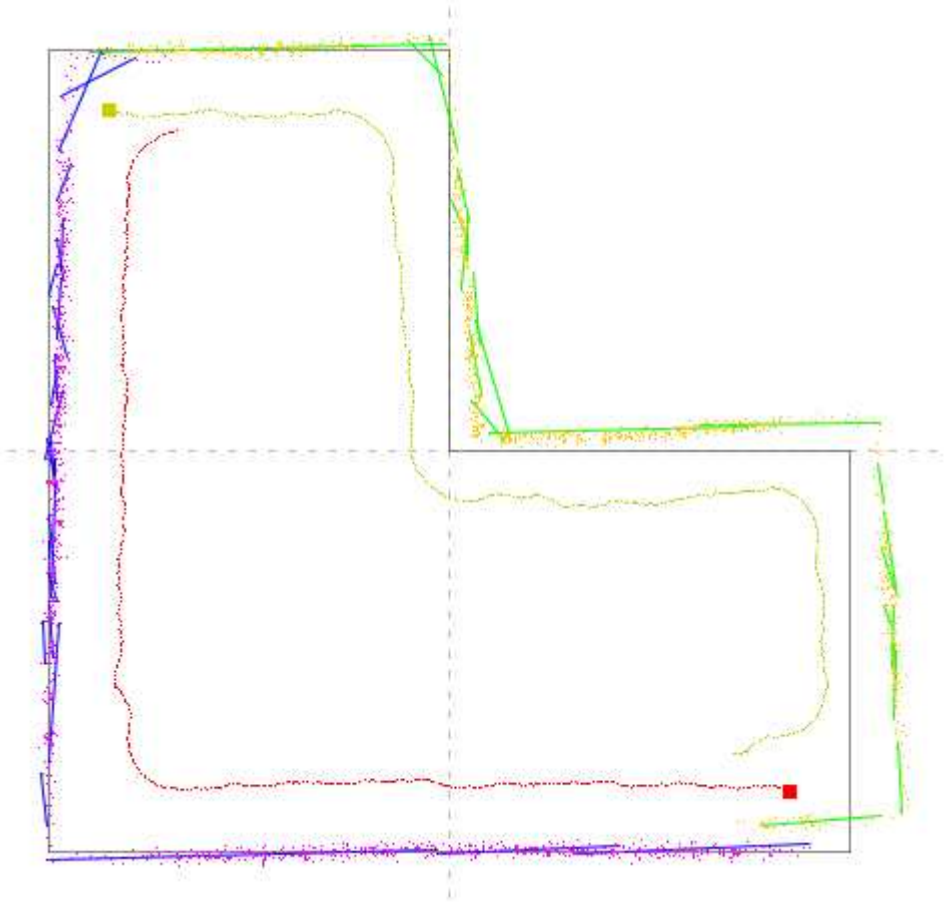
Şekil 5.38 L şeklindeki bir ortamın ideal şartlarda iki robot tarafından çıkarılan haritası

● Duyargalara (ortalaması 0 ve standart sapması 2 cm olan normal dağılıma uyacak biçimde rastgele olarak), tekerleğe ve döner kodlayıcıya (ortalaması 0 ve standart sapması 2° olan normal dağılıma uyacak biçimde rastgele olarak) gürültü eklendiğinde ve kaygan bir zeminde (ortalaması 0 ve standart sapması 5% olan normal dağılıma uyacak biçimde rastgele olarak patinaj veya kayma) ortamın iki robot tarafından k=6 ve thr=2cm parametreleri ile çıkarılan haritası Şekil 5.39'daki gibidir.



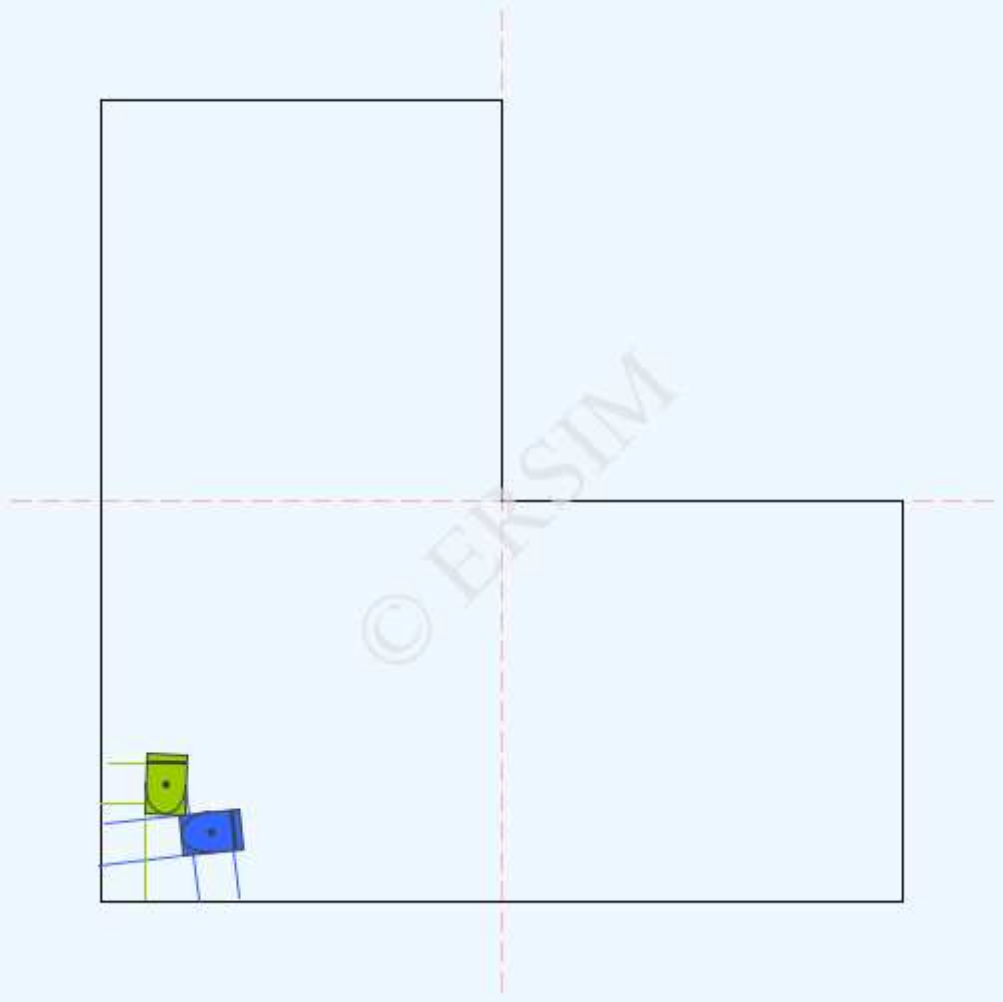
Şekil 5.39 L şeklindeki bir ortamın robot aygıtlarına gürültü eklenerek ve kaygan bir zeminde k=6 ve thr=2 parametreleri ile iki robot tarafından çıkarılan haritası

● Duyargalara (ortalaması 0 ve standart sapması 2 cm olan normal dağılıma uyacak biçimde rastgele olarak), tekerleğe ve döner kodlayıcıya (ortalaması 0 ve standart sapması 2° olan normal dağılıma uyacak biçimde rastgele olarak) gürültü eklendiğinde ve kaygan bir zeminde (ortalaması 0 ve standart sapması 5% olan normal dağılıma uyacak biçimde rastgele olarak patinaj veya kayma) ortamın iki robot tarafından k=10 ve thr=5cm parametreleri ile çıkarılan haritası Şekil 5.40'daki gibidir.



Şekil 5.40 L şeklindeki bir ortamın robot aygıtlarına gürültü eklenerek ve kaygan bir zeminde k=10 ve thr=5 parametreleri ile iki robot tarafından çıkarılan haritası

Yine L şeklindeki ortamda bu sefer yeşil renkli robotun başlangıçtaki eğimi 270° yapılıp robotlar birbirlerine doğru yönlendirilerek bir deneme yapıldığında robotların ortamın köşesinde birbirlerine çarptıkları gözlenmiştir (Şekil 5.41). Bu durumun sebebi, robot kontrolü için kullanılan algoritmanın durağan ortamlar için geliştirilmiş olması ve robotların hareketli olmaları sebebiyle böyle bir durumla karşılaşma durum bilgisinin ve uygulayacakları bir çözümün bulunmamasıdır.



Şekil 5.41 L şeklindeki bir ortamda iki robotun birbirlerine doğru yönlendirilmeleriyle oluşan durum

6. SONUÇLAR

Bu çalışmada, SLAM (*Eş Zamanlı Konum Belirleme ve Haritalama*) uygulamalarında kullanılmak için geliştirilen gezgin robotun tüm özellikleri ve davranışları modellenerek bir simülasyon ortamı oluşturulmuş ve bu robotu SLAM uygulamalarında kullanabilecek şekilde kontrol edebilecek olan kumanda birimi geliştirilmiştir.

Daha sonra oluşturulan simülasyon üzerinde ortamın sunduğu imkanlardan yararlanılarak ve geliştirilen kumanda birimi kullanılarak denemeler yapılmış ve sonuçları gözlemlenmiştir.

Simülasyonda harita alanının oluşturulmasının kolaylıkla yapılabildiği, değişik konfigürasyonlarda robotların oluşturulabildiği ve bu robotların tek veya beraberce kullanılarak simüle edilebildiği görülmüştür. Ayrıca robot aygıtlarının ölçümlerine gürültü eklenerek veya çalıma zemininin özellikleri ayarlanarak robot kontrolünde gerçeğe yakın denemeler yapılabilmiştir.

Oluşturulan kontrol biriminin simülasyonla iletişime geçebildiği ve simülasyon ortamında bulunan robotları kullanılan algoritmaya göre kontrol edebildiği görülmüştür. Tek robot engel takip etme algoritmasının başarılı bir şekilde ortamın haritasını çıkarabildiği tespit edilmiştir. Buna rağmen aynı algoritmanın kullanıldığı çok robot kontrolünde robotların birbirlerine çarpabildikleri ve karar verilemeyen durumlarla karşılaşılabildiği görülmüştür. Oluşan bu durumun sebebi olarak algoritmanın durağan ortamlara göre geliştirilmesi ve hareketli nesnelerin bulunduğu, şartların değişebildiği durumlara uyum sağlayamaması olduğu söylenebilir. Çoklu robot kontrolünde, merkezi karar alma biçiminde tüm robotların birbirlerine göre durumları kontrol edildikten sonra robotların birbirlerinin işleyişlerini bozamayacakları biçimde yönlendirilmelerini sağlayacak olan bir yaklaşımın uygulanmasının daha başarılı olacağı düşünülebilir.

Sonuç olarak simülasyon ortamının gezgin robotu başarılı bir şekilde modelleyebildiği ve yapılan çalışmalarda kullanıldığında olumlu sonuçlar alınabileceği tespit edilmiştir. Kontrol biriminin de robot kontrolünde başarılı olduğu ve değişik algoritmaların denenmesi için kullanılabileceği görülmüştür.

KAYNAKLAR

Banks, J., Carson, J.S., Nelson, B.L. ve Nicol, D.M. (2004), Discrete-Event System Simulation, Prentice Hall, New Jersey.

Durrant-Whyte, H. ve Bailey, T. (2006), "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms", Robotics and Automation Magazine, 13:99-110.

Erkut, H. (1992), Yönetimde Simülasyon Yaklaşımı, İrfan Yayımcılık, İstanbul.

Flanagan, D. (2004), Java Uygulamaları (Çev., B.S. Furat), Pusula, İstanbul.

Gifford, C. (2008), Robotlar (Çev., T.T. Sabuncuoğlu), TÜBİTAK Popüler Bilim Kitapları, Ankara.

Kurt, Z. (2007), "Eş Zamanlı Konum Belirleme ve Haritalamaya Yönelik Akıllı Algoritmaların Geliştirilmesi", Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi.

Manseur, R. (2006), Robot Modeling & Kinematics, Charles River Media, Boston.

Michel, O. / Cyberbotics Ltd (2004), "Webots™: Professional Mobile Robot Simulation", International Journal of Advanced Robotic Systems, 1:39-42.

Schildt, H. (2003), Java 2 (Çev., Ü. Yazıcı), Alfa, İstanbul.

Siegwart, R. ve Nourbakhsh, I.R. (2004), Introduction to Autonomous Mobile Robots, The MIT Press, Cambridge, Massachusetts.

Şahin, O. ve Yavuz, S. (2009), "Autonomous Parallel Parking Robot", International Symposium on INnovations in Intelligent SysTems and Applications (INISTA 2009), 29 June-1 July 2009, KTU, Trabzon.

Thrun, S., Burgard, W. ve Fox, D. (2005), Probabilistic Robotics, The MIT Press, Cambridge, Massachusetts.

Yavuz, S., Amasyalı, M.F., Balcılar, M., Bilgin, G., Dinç, T. ve Kurt, Z. (2006), "Eş Zamanlı Konum Belirleme ve Harita Oluşturma Amaçlı Otonom Bir Robot", ELECO 2006, 6-10 Aralık 2006, Bursa.

Yavuz, S., Biçer, S. ve Kurt, Z. (2009), "Genişletilmiş Kalman Filtresi Yöntemine Dayalı Eş Zamanlı Konum Belirleme ve Haritalama", IEEE 17. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU 2009), 9-11 Nisan 2009, Antalya.

İNTERNET KAYNAKLARI

[1] http://carmen.sourceforge.net/

[2] http://www.cyberbotics.com/products/webots/

[3] http://java.sun.com/docs/books/tutorial/

[4] http://www.netbeans.org/

[5] http://rxtx.qbang.org/

ÖZGEÇMİŞ

Doğum tarihi	27.02.1983	
Doğum yeri	Aksaray	
Lise	1994-2001	Aksaray Hazım Kulak Anadolu Lisesi
Lisans	2002-2007	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fak. Bilgisayar Mühendisliği Bölümü
Yüksek Lisans	2007-2009	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı