Kapalı Alanlarda Kullanılan Otonom Yer Araçları İçin Yer Bulma, Haritalandırma Ve Güzergâh Planlama Yapabilen Cihaz Tasarımı Ve Geliştirilmesi

Program Kodu:3501

Proje No:119E376

Proje Yürütücüsü:

Dr. Öğr. Üyesi PINAR OĞUZ EKİM

Bursiyer(ler)

SERCAN ÇAĞDAŞ TEKKÖK

BEKİR BOSTANCI

MEHMET EMRE SÖYÜNMEZ

TEMMUZ 2022

ANKARA

Önsöz

Bu proje kapsamında, çeşitli algılayıcı teknolojilerinin harmanlanması sayesinde iç mekânda hareket eden otonom yer araçlarının yüksek doğrulukta konumlandırılmasını, çevresinin haritalandırılmasını ve de güzergâh planlamasını yapabilecek cihaz tasarımı ve geliştirilmesi gerçekleştirilmiştir. Bu anlamda çalışmamız çeşitli algılayıcılardan gelen bilgileri işleyebilen donanım ve de belirtilen problemleri çözebilen algoritmaları elde etmiştir. Önemli başarılar ve çıktılar elde ettiğimiz bu projeyi destekleyen TÜBİTAK'a çok teşekkür ederim.

İçindekiler

Tablo Listesi

Şekil Listesi

Özet

Otonom sistemler son yıllarda insan için tehlikeli uygulamalarda veya insanla iş birliği içinde oldukları alanlarda oldukça önem kazanmıştır. Bu uygulamaların pek çoğunda otonom sistemin bulunduğu yeri tespit etmesi, çevresinin haritasını çıkartması ve de bu bilgilere dayanarak görevine uygun olan güzergâhı planlaması önemli alt problemler olarak karşımıza çıkmaktadır. Farklı sensörlerden gelen bilgilerin güvenilir ve verimli bir şekilde birleştirilmesinin, konum bulma ve daha birçok robotik sorununun çözümüne yardımcı olmaktadır. Çünkü bir sensör belirli çevresel koşullar altında ölçüm alamadığında diğeri hata yayılımını azaltmak için kullanılabilir. Buna ek olarak, bir sensörün hata özellikleri, farklı özelliklere sahip başka bir sensörün kullanılması ile düzeltilebilir. Sensör füzyonunun önemi yaygın olarak bilinmesine rağmen, yeni geliştirilen sensör teknolojileri ve uyumlu algoritmalar henüz kapsamlı bir şekilde çalışılmamıştır. Bu nedenle projemizde özellikle Odometre, UWB ve de Lidar sensörlerinden gelen bilgiler harmanlanmıştır. Algoritmalar Python ve C dilleri ile geliştirilmiş ve ROS aracılığı ile gerçek zamanlı olarak donanımlardan gelen bilgileri işleyebilmişlerdir. 5cm-10cm konumlandırma hassasiyetinde oldukça gürbüz bir sistem ortaya çıkarılmıştır.

Anahtar Kelimeler: Yer bulma, haritalandırma, güzergâh planlama, algılayıcı harmanlama

Abstract

In recent years, autonomous systems have gained importance in applications that are dangerous for humans or in areas where they are in cooperation with humans. In many of these applications, determining the location of the autonomous system, mapping its environment, and planning the appropriate route based on this information appear as important sub-problems. Combining information from different sensors reliably and efficiently helps to solve many robotics problems such as positioning and more. Because when one sensor fails to measure under certain environmental conditions, the other can be used to reduce error propagation. In addition, the error characteristics of a sensor can be corrected by using another sensor with different characteristics. Although the importance of sensor fusion is widely known, newly developed sensor technologies and compatible algorithms have not yet been extensively studied. For this reason, information from Odometer, UWB and Lidar sensors has been blended in this project. The algorithms which were developed with Python and C languages, were able to process the information coming from the hardware in real time via ROS. Hence a very robust system with a position accuracy of 5 cm-10 cm has been achieved.

Keywords: Localization, mapping, path planning, sensor fusion

1. GİRİŞ

Otonom yer araçlarının iç mekânlarda aktif ve güvenli bir şekilde kullanılması için öncelikle otonom konumlandırma, haritalandırma ve güzergâh planlamasının yapılması gerekmektedir. Başka bir deyişle; yükü veya ziyaretçiyi bir yerden bir yere götürme gibi verilen görevin başarılması; çevresi hakkında sıcaklık, nem vb. gibi edindiği bilgilerin anlamlandırılması ancak ve ancak otonom yer aracının konumu, çevresinin yapısı ve bunun üzerinde planlama yapabilme kabiliyeti ile mümkün olmaktadır. Bu ihtiyaca rağmen her bir uygulamanın farklı gereksinimleri olması ve iç ortam dinamiklerinin dış ortama göre farklılık göstermesi nedeniyle genel kabul gören bir çözüm ortaya konulamamıştır. Değişik algılayıcılardan gelen bilgilerin verimli bir şekilde birleştirilmesi ise bu problemin temel çözümüdür ve dahası bir algılayıcının doğru ölçüm yapamadığı koşulda diğeri devreye girerek hataları azaltabilir. Ek olarak bir algılayıcının hata karakteristiği başka karakterdeki algılayıcı yardımı ile düzeltilebilir. Bu işlemin önemi bilinmesine rağmen yeni gelişen UWB ve Lidar gibi algılayıcı teknolojileri ve onlarla uyumlu algoritmalar ve de donanımlar yeterince çalışılmamıştır. Projemiz bu yönüyle önemli bir eksikliği gidermiştir.

Otonom yer araçlarının hastaneler, fabrikalar ve benzeri yerlerde servis robotları, müzelerde tur rehberi ya da afet durumlarında yardımcı robot olarak gibi geniş bir yelpazede kullanım alanı mevcuttur. Bu bağlamda çalışmamız, elde edilen algoritmalar ve de donanım sayesinde bu robotların geliştirilmesine ışık tutmuştur.

2. LİTERATÜR ÖZETİ

Konumlandırma sistemlerinin pek çoğu tek tip algılayıcılardan sağlanan bilgilere dayanır. Bu sistemlere örnek olarak sonar algılayıcılar (Tardos vd. 2002), 2D lazerler (Thrun vd. 2005), 3D lazerler (Murtra 2011), robot üzerine monte edilmiş kameralar (Gamallo vd. 2015) verilebilir. Buna rağmen, mükemmel algılayıcı diye bir şeyden söz edemeyiz; her algılayıcının kısıtları vardır ve hiçbir algılayıcı her duruma uygulanamaz. Bu nedenle bahsedilen sistemlerin gürbüz bir şekilde gerçek dünyada karşılaşılabilecek her duruma cevap verebilmesini bekleyemeyiz. Konumlandırma çalışmaları bu tip sorunları yeni tip algılayıcılar ve de giderek karmaşıklığı artan sistemler tasarlayarak aşmaya çalışıyor. Bu projede birçok algılayıcıdan gelen bilginin harmanlandığı tekniklere odaklanıldı (Khaleghi vd. 2013; Canedo-Rodriguez vd. 2016). Farklı karakterlerdeki, uygulamaya ve çevresel faktörlere göre performansları değişen algılayıcılardan gelen bilgileri birbirlerinin kötü yönlerini iyileştirecek biçimde birleştirerek daha gürbüz bir

konumlandırma sistemi elde edildi. Bu konumlandırma sistemi ve güzergâh planlaması ile otonom yer araçlarının amaca yönelik hareket etmesini sağlayacak beyin yapısı sağlandı. Literatürde eş zamanlı konumlandırma ve de haritalandırma işlemi **S**imultaneous **L**ocalization and **M**apping (**SLAM**) olarak adlandırılır ve raporumuzun geri kalanında bu konudan bahsederken SLAM kısaltmasını kullanacağız.

Resim veya video bilgisi veren algılayıcıları kullanarak SLAM problemini çözmek mümkün olabilmektedir. (Hwang vd. 2011) çalışmalarında hareketli bir robotun üstüne tavana bakacak şekilde kameranın yerleştirilmesi ve resimlerin alınıp işlenmesi ile haritalandırma ve konum bulma yapılmaya çalışmışlardır. Fakat tavanda ayırt edici özelliklerin bulunmaması ve de görüntüden direkt uzaklık bilgisinin kolayca elde edilememesi bu metodun kullanılabilirliğini azaltmaktadır. Yapılan başka bir araştırmada ORB-SLAM adlı algoritma ile kamera tarafından kaydedilen videolar kullanılarak hem iç hem dış mekân için SLAM problemi aşılmaya çalışılmıştır (Mur-Artal vd. 2015). Görüntü işlemenin daha karmaşık algoritmalara ihtiyaç duyması ve sis gibi görüntü kalitesini bozan ortamlarda iyi veri toplanamaması otonom sistemler için bir engel oluşturmaktadır. Ek olarak otonom sistemlerde kameraların görüşünden daha hassas biçimde çevrenin yapısını ve de engelleri belirten sistemlere ihtiyacımız var, böylece acil bir durumda daha hızlı önlem alınabilsin. Daha da fazlası Zhang vd. (2015)'de belirtildiği gibi kameradan gelen bilginin işlenmesi yüksek işletim kabiliyeti gerektirir. Sistemimizde ise otonom araçlar için maliyeti düşük ve hassas bir SLAM ve güzergâh planlama sistemi farklı algılayıcılardan gelen bilgilerin harmanlanması ile elde edilmiştir.

Otonom yer aracı ve çevredeki vericiler arasındaki uzaklık ölçümleri ve de kontrol girdileri kullanılarak eş zamanlı konumlandırma ve de haritalandırma yapmak mümkündür (Blanco vd. 2008; Deibler vd. 2010). Bu gibi çalışmalarda ultrasonik algılayıcısından yayılan ses sinyalinin engele çarpıp geri gelme zamanından elde edilen uzaklık bilgisi (Schweinzer, H. ve Kaniak, G. 2010) ile konumlandırma yapılmaya çalışılmıştır. Ses sinyalinin bir koni içerisinde yayılması, çevresel faktörlerden çok fazla etkilenmesi ve de ses sinyalini kullanarak ölçülebilen mesafenin 10 metrenin altında olmasından dolayı, hassas ve geniş alan içeren fabrika içi ortamlar için tasarlanması planlanan otonom yer araçları için uygun bir çözüm değildir. (Liu vd. 2007) Wi-Fi algılayıcılarından gelen sinyalin gücüyle (**R**eceived **S**ignal **S**trength (**RSS**)) uzaklık arasındaki ilişkiden mesafe ölçümlerini kullanmışlardır. RSS'den elde edilen uzaklık ölçümünü tahmin etmek ve sonrasında kullanmak için Fingerprinting (He vd. 2016) olarak adlandırılan emek yoğun bir işlemin yapılması ve iç ortamın düzenli aralıklarla RSS-uzaklık bilgi ilişkisinin kontrol edilmesi

gerekmektedir. Ek olarak sinyal gücü çevresel faktörlerden oldukça etkilendiği için buna bağlı olarak RSS'den elde edilen uzaklık ölçümü de etkilenecek ve bu teknolojiyi kullanan sistemlerin hassasiyetleri düşük olacaktır. UWB sinyallerden elde edilen uzaklık bilgisi oldukça hassastır. Fakat yeni ve son birkaç yıla kadar pahalı bir teknoloji olması nedeniyle sadece UWB kullanarak konumlandırma yapmak isteyen sistemler literatürde oldukça azdır ve araştırılmaya değerdir (Sahinoglu vd. 2008). Fakat sadece UWB algılayıcı kullanarak, çevresel haritalandırma yapmak ya da engel tanımak mümkün değildir.

Literatürdeki (Grisetti vd. 2005)'in ve (Kohlbrecher vd. 2011)'in çalışmaları Lidar tabanlı SLAM araştırmalarını ileri bir noktaya taşımıştır. Gmapping (Grisetti vd. 2005) adı verilen yöntem ızgara (grid) haritalarını uyarlanabilir yaklaşımlarla parçacık süzgeci kullanarak öğrenmektedir. Kullandıkları Rao Blackwellized parçacık süzgeci sayesinde parçacık olasılık dağılımlarını yakınsayan parçacık sayısı oldukça azalmıştır ve işlemleme hızı artmıştır. Hector SLAM algoritması ise (Kohlbrecher vd. 2011) yaklaşık harita eğimlerini ve çoklu çözülümlü ızgaraları kullanarak gerçek zamanda hızlı bir şekilde doluluk ızgara haritalarını öğrenebilir. Gmapping odometre bilgisine ihtiyaç duyarken, Hector SLAM sadece lazer uzaklık bulucudan gelen bilgiyi kullanır. Wachaja vd. (2017) birden fazla lazerle görme engeli ve de yürümesinde problem olan kişiler için akıllı yürüteç sistemi tasarlamıştır. Lazerlerden birisi kullanıcının hareketlerini algılarken diğeri ise bir servo-motor yardımı ile dönen lazerden gelen bilgilerle çevredeki engelleri tanımlamak için kullanılmaktadır. Ortam haritasının daha önceden çıkartıldığı varsayılarak kullanıcının engelleri algılayarak güzergâh planlama algoritması ile ilerlemesi sağlanmıştır. Endüstride de taşıma, bir yerden alıp bir yere koyma gibi uygulamalar için robot sistemlerine ihtiyaç duyulmaktadır. İki tane SICK S300 lazer uzaklık ölçümü algılayıcısı ile donatılmış KUKA omniRob sistemi Monte Carlo konumlandırması ve tarama eşleştirme (scan matching) teknikleri ile statik ve dinamik fabrika içi ortamlarda verilen görevleri yapması için gerekli olan pozisyon bulma problemini çözmek için kullanılır (Rowekamper vd. 2012). Bu uygulamada da haritanın önceden çıkartıldığı varsayılmıştır. Değişen dış ortam koşulları için benzer bir teknik de (Tipaldi vd. 2013) tarafından uygulanmıştır. Lazerin hassasiyetini etkileyecek cam vb. gibi engellerin sıkça karşılaşıldığı iç ortamlarda, sadece Lidar'dan gelen bilgi ile SLAM uygulaması yapmaya çalışmak hassasiyeti azalttığı gibi bu algılayıcının çeşitli donanımsal problemlerden dolayı bozulması ile sistemin geçici olarak kullanılamaz hale gelmesine sebep olacaktır.

Radio **F**requency **Id**entification (RFID) teknolojisi ve lazer uzaklık tarayıcısı kullanarak SLAM yapmak mümkündür (Mota vd. 2018). Fakat çevresel faktörlerin RFID tag'lerin ortamda bulunduğu halde tanınmaması (false negative) ya da ortamda bulunmadığı halde tanınması (false

positive) durumlarına yol açmasından dolayı kesinliği çok da yüksek değildir (Hahnel vd. 2004). Lidar; duman, sis ve toz gibi engellerin olduğu ortamlarda görüş sağlayamaz bu nedenle (Fritsche vd. 2017)'nin yaptığı çalışma Lidar ve Radar bilgisini kullanarak görüşün düşük olduğu ortamlarda SLAM problemini çözmeyi hedeflemektedir. (Wang vd. 2017), UWB ve görsel atalet odometre bilgisini harmanlayarak görsel sürüklenme hatasını UWB uzaklık ölçümleri ile ortadan kaldırarak daha gürbüz bir sistem elde etmişlerdir. Kullandıkları yöntemde sabit referanslara ve iyi bir başlangıç bilgisine ihtiyaçları vardır ama bizim çalışmamızda böyle bir bilgiye ihtiyaç duyulmamaktadır. Benzer şekilde Song vd. (2018), UWB ve oldukça pahalı bir Lidar'dan gelen bilgileri harmanlamışlardır. Odometre kullanmamışlar ve otonom yer aracının sabit, doğrusal hız ile gittiğini varsayarak sadece UWB'den gelen uzaklık ölçümü bilgisi ile yönelim bilgisini de tahmin etmeye çalışmışlardır. Otonom yer aracı her zaman doğrusal ve sabit hızda gitmeyeceği için projemizde odometre bilgisini kullanmak sureti ile daha ucuz bir Lidar'la daha iyi bir SLAM algoritması geliştirilmiştir.

Güzergâh planlama algoritması otonom robotun bir alanda iki nokta arasında engellerden kaçınarak en kısa ya da en iyi güzergâhı bularak ilerlemesini sağlar (Le vd. 2018). Robotik uygulamalarında iyi bir performans elde etmek için yol planlama önemli bir bileşendir (Patle vd. 2019).

SLAM algoritması robotu konum belirsizliğini azaltan bölgelere yönlendirirse aktif SLAM algoritması olarak adlandırılır. Genel olarak robot bilgi kazancının en fazla olacağı daha önce ziyaret edilmiş yolu seçer ve bilinen referanslardan (noktalar, çizgiler, düzlemler vb.) ölçümler alır, bu tekniğe döngü kapama (loop closing) denir (Maurović vd. 2017). Böylece hem konum hem de haritalandırmadaki belirsizlik azaltılır ve sıklıkla keşif birimi ile beraber çalışır (Liu vd. 2007; Leung vd. 2006). Fakat hedefimiz fabrika, ya da bina içi ortamlarda ucuz ve gerçek zamanlı çözümler olduğu için bu projemizde aktif SLAM kullanılmamıştır. Literatürde güzergâh belirleme için A*, D* ve Dijkstra algoritmaları kullanılır (Siciliano B. ve Khatib O., 2016). Bunların içinden, engellerden kaçınan, kat edilen mesafenin ve dönüşün minimuma indirildiği A* algoritması ile güzergâh planlama algoritmaları içinde bir adım öne çıkmaktadır (Hasegawa vd. 2016). A* algoritması sezgisel fonksiyon tabanlıdır; haritayı düğümlerden oluşmuş bir şema olarak düşünürsek, A* algoritması her bir düğüm için sezgisel fonksiyonun değerini hesaplar ve her bir düğüm için birçok komşu düğümü kontrol ederek çarpışma olasılığını sıfıra indirir (Guruj vd. 2016).

Bu proje kapsamında geliştirdiğimiz algoritmalar ve sistemimizin çıktıları yayınladığımız 4 makale (1 SCI (Oguz-Ekim 2020b) ve 3 TR-index (Oguz-Ekim 2020d; Oguz-Ekim 2021; Tatli vd 2021)) ve de 2 tane sözlü sunulmuş bildiri (1 ulusal (Oguz-Ekim vd. 2020c) ve 1 uluslararası (Oguz-Ekim, 2020a)) Bölüm 3 ve Bölüm 4'te detaylı şekilde bahsedilmiştir.

3. GEREÇ VE YÖNTEM

İlklendirme, tamamen otonom navigasyona ulaşmak için çözülmesi gereken önemli sorunlardan biridir. Geçmişte, bu sorunun üstesinden gelmek için görsel-eylemsizlik odometrisi (Huang vd. 2018) kullanılmıştır. GNNS'den gelen konum bilgisini kullanan başka bir yöntem (Atia vd 2015)'te önerilmiştir. Ancak bu iç mekan uygulamaları için uygun değildir. Çünkü GNSS bilgilerine erişemeyebiliriz veya sahip olsak bile çok büyük hatalara neden olur. Önceden tanımlanmış bir haritada bir robotu gezdirmek için, Robot İşletim Sisteminin (ROS) gezinme yığını (navigasyon stack) kullanılır. Genel prosedür, bir başlangıç pozunu (haritaya göre hem konum hem de yönelim) ROSun grafik arayüzü olan Rviz üzerinde manuel olarak tanımlamaktır. Başlatma yeterince doğru değilse, robot hedef konuma ulaşamayabilir.

Bunun yerine, UWB ve LiDAR verilerini kullanan bir başlatma yöntemi önerdik. Yinelemeli En Yakın Nokta (ICP) algoritmalarının (Besl vd. 1992) varyasyonu, önceden elde edilen harita verileri ve mevcut LiDAR taramalarını eşleştirmek için kullanılır. Fakat harita verileri oldukça büyük olabilir ve benzer noktalara sahip olabilir, böylece eşleştirme işlemi de zaman alıcı ve hatalı olabilir. İlklendirme sürecini otomatikleştirmek için, UWB sensörlerinden elde edilen uzaklık ölçümleri (Beck vd 2008) ya da uzaklık ölçümlerinin farkının karelerini kullanan (Oguz-Ekim 2020b) en küçük kareler yöntemi ile cihazın mevcut konumu bulunabilir. UWB verileri, robotun baktığı yönü bulmak için yeterli değildir. Ancak bunlardan elde edilen konum bilgisi, etrafındaki harita verilerini eşleştirmek için bir referans noktası olarak kullanılır ve mevcut LiDAR taramalarının harita ile eşleştirilmesi ile, haritada robotun yönü hesaplanabilir. Bu amaçla, LiDAR tarama verileri adım adım 10 derece döndürülür ve tüm noktalar, her bir yinelemede harita verileri ile karşılaştırılır. Her LiDAR uç noktasının haritadaki en yakın noktaya olan mesafesi hesaplanır ve bu prosedür 360 derecenin tamamı için tekrarlanır. Noktaların mesafelerinin toplamının en düşük olduğu açı değeri robotun açısını verir. UWB'den gelen verilerle çözdüğümüz konumlandırma problemini robot hareket ederken pozisyonunu bulmak için de kullanabilmekteyiz. Yaptığımız simülasyon ve de gerçek testler sonucunda konumlandırma hatamızın 2 cm ile 5 cm arasında olduğunu hesapladık.

Geliştirdiğimiz bir başka konumlandırma ve de ilklendirme algoritmasında (Oguz-Ekim vd 2020c), UWB ve odometre verilerini ilk konumun belirlenmesinde kullandık. Bu problemin çözümü için bizim önerimiz; robot hareket etmeden önce UWB'den uzaklık ölçümleri alıp düz hareket etmesinin ardından tekrar UWB ve odometreden ölçüm alarak, trigonemetrik bağıntılarla robotun bakış açısını ve konumunu saptamaktır. Yine aynı bildiride konumlandırma için UWB ve odometre verilerini girdi olarak alan Genişletilmiş Kalman Filtre (EKF) önermekteyiz. Yaptığımız simülasyon ve de gerçek testlerde bu algoritmaların da 5cm civarında konumlandırma hatalarına sahip olduğunu görmüş bulunmaktayız.

Başka bir konumlandırma algoritmamızda (Oguz-Ekim 2020a) Bayes süzgeç yapılarından olan Parçacık süzgecini kullandık. Bu yapı da yine odometre ve de UWB'den gelen bilgilerin harmanlanması ile çalışmaktadır. Fakat EKF'e göre daha iyi bir performans elde edemedik. Bunun nedeni Parçacık süzgecinde uygulamaya göre parametrelerin seçilmesinin daha önemli olduğu ve konumlandırma algoritmalarının performansını önemli ölçüde etkilediği sonucunu çıkardık. Bu algoritmalardan elde ettiğimiz konumlandırma hassasiyeti 10-15 cm civarindadır.

Son olarak başka bir çalışmamızda odometre verilerini, uyarlanabilir Monte Carlo konumlandırma (AMCL) poz bilgisini ve de UWB uzaklık ölçümlerini harmanlayan Genişletilmiş Kalman Filtresi (EKF) geliştirilmiştir (Oguz-Ekim 2020d). Bu algoritma İzmir Ekonomi Üniversitesinin Tesla laboratuvarında bulunan Turtlebot3'un üzerine eklenen UWB sensörü ve de laboratuvarın çeşitli duvarlarına konulan UWB vericilerin arasında hem direk görüşün olduğu hem de direk görüşün olmadığı diğer durumlar için gerçek ortamda test edilmiştir. Aşağıda bu çalışmamızın detaylarını diğer algoritmalarımızı da kapsayacak şekilde sunduk.

3.1. Konumlandırma algoritmasının formülasyonu (EKF)

Robotik uygulamaların konum bulma problemini çözmek için özel bir Bayes Filtresi olan EKF önerilmektedir. Zaman indeksi t ile tahmin edilecek x_t durum olsun. Kontrol girişleri $u_{1:t} = \{u_t, u_{t-1}, \cdots u_1\}$ ve sensörlerden gelen bilgi veya ölçümler $z_{1:t} = \{z_t, z_{t-1}, \cdots z_1\}$, kullanılarak durum tahmin edilmeye çalışılır. Ölçümler durum hakkında kısmi bilgiye sahiptir veya hata içerebilirler. Ek olarak, durum modelleme ve kontrol girdileri nedeniyle hatalar olabilir. Durumların evrimi ve ölçümlerin hata özellikleri olasılık yasalarına uyar.

Robotik uygulamalarda hareket modeli ve gözlem modeli şu şekilde tanımlanır:

$$x_t = g(x_{t-1}, u_t, v_{t-1})$$
$$z_t = h(x_t, w_t). \qquad (1)$$

Hareket gürültüsü v_{t-1} ve gözlem gürültüsü w_t ile ifade edilir. Hareket ve gözlem modelleri doğrusal ise ve sesler toplamalı, bağımsız ve aynı şekilde dağıtılmış (i.i.d.) Gauss dağılımları ise, Kalman Filtresi (KF) optimum filtredir (Bar-Shalom vd 2004). Diğer bir deyişle,
İlk inanç bel (x_0) bir Gauss dağılımına sahipse: $x_o \sim \mathcal{N}(\mu_o, \Sigma_o)$

$$bel(x_0) = p(x_0) = \det(2\pi\Sigma_o)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_o^{-1}(x_0 - \mu_0)\} \qquad (2)$$

Hareket olasılığı modeli doğrusal ve hareket gürültüsü toplamsal ise i.i.d. Gauss dağılımı ise:

$$x_t = A_t x_{t-1} + B_t u_t + v_{t-1},\ v_t \sim \mathcal{N}(0, R_t)$$
$$p(x_t|x_{t-1}, u_t) = \det(2\pi R_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\} \quad (3)$$

Gözlem olasılık modeli doğrusal ve gözlem gürültüsü toplamsal i.i.d. Gauss dağılımı ise:

$z_t = C_t x_t + w_t,\ w_t \sim \mathcal{N}(0, Q_t)$

$$p(z_t|x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\} \qquad (4)$$

daha sonra arka dağılım, bel (x_t) her zaman Gauss şeklindedir ve KF bunu ortalama μ_t ve t zamanında kovaryans Σ_t parametreleriyle ifade eder.
Tahmini ortalama ve kovaryans, hareket modeli kullanılarak güncellenir ve bu adım, tahmin adımı olarak adlandırılır:

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t. \qquad (5)$$

Öngörülen ortalama ve kovaryans, gözlem modeli kullanılarak güncellenir ve bu adım, güncelleme adımı olarak adlandırılır:

$$K_t = \overline{\Sigma}_t C_t (C_t \overline{\Sigma}_t {C_t}^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t)\overline{\Sigma}_{.t}. \quad (6)$$

Denklem 6'da Kalman kazancı K_t durum güncellemesinde ölçümün hangi oranda kullanıldığını gösterir.

Hareket ve gözlem modeli doğrusal olmadığında, KF'nin yaklaşık formu olan EKF kullanılır. EKF, tahmin ve güncelleme adımlarında yerel doğrusal formlar olarak model işlevlerinin birinci derece Taylor yaklaşımlarından yararlanır.

3.1.1. Hareket modeli: odometri bilgileri

Teknik olarak odometri bir sensördür, ancak robotik uygulamalar için kontrol girdileri sağlar (Dobrev vd 2011). Odometri hareket modeli, tahmin basamağındaki durumlar arasında geçiş olasılığını içerir. Robotun (*t-1, t*] aralığında δ_{trans} hareket ettiği ve δ_{rot} döndüğü varsayılırsa girdi u_t = (δ_{trans}, δ_{rot}) olarak ifade edilir. X_{t-1} = (x, y, θ) ve x_t = (x', y ', θ') önceki ve mevcut robot pozları olmak üzere aşağıdaki ilişkiye sahiptir.

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \delta_{trans} \cos\theta \\ \delta_{trans} \sin\theta \\ \delta_{rot} \end{pmatrix}. \quad (7)$$

Denklemde 4'te kullanılmak üzere hareket modelinin *u* ve x'e göre birinci dereceden türevi şöyledir:

$$A_t = \frac{\partial g(.)}{\partial x} = \begin{bmatrix} 1 & 0 & -\delta_{trans} \sin\theta \\ 0 & 1 & \delta_{trans} \cos\theta \\ 0 & 0 & 1 \end{bmatrix}, B_t = \frac{\partial g(.)}{\partial u} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}. \quad (8)$$

3.1.2. Gözlem modeli: UWB sensörleri

UWB sensörleri, radyo sinyallerinin iki yönlü varış zamanından itibaren verici ve alıcı arasındaki mesafeyi ölçebilir. İç mekân lokalizasyonu sorunu, çoklu yol ve düz olmayan görüş koşulları nedeniyle çok zordur. Bununla birlikte, UWB teknolojisi, sinyalin doğrudan yolu, ultra geniş bant sayesinde çoklu yoldan kolayca ayırt edilebildiğinden daha doğru aralık ölçümleri sağlar (Bregar vd. 2018).

p_t^i ve l^j alıcının robot üzerindeki ve vericinin 2B iç ortamdaki konumları (3B'ye genişletme basittir) olsun, aralarındaki gerçek uzaklık şu şekildedir:

$$r_t^{UWB} = \|p_t^i - l^j\| = \sqrt{(x_t^i - l_x^j)^2 + (y_t^i - l_x^j)^2}. \qquad (9)$$

Çevresel faktörler bağımsız bir Gauss gürültüsü $w_t \sim N(0,\sigma_w^2)$ olarak modellenebilir ve aşağıdaki ifadeye sahip ölçümlere eklenebilir:

$$z_t = r_t^{UWB} + w_t. \qquad (10)$$

Eşitlikteki gözlem modeli fonksiyonunun x'e göre birinci türevi her uzaklık ölçümü için aşağıdaki gibi elde edilir:

$$C_{j,t} = \frac{\partial h(.)}{\partial x} = \left[\frac{x_t^i - l_x^j}{\sqrt{(x_t^i - l_x^j)^2 + (y_t^i - l_x^j)^2}} \quad \frac{y_t^i - l_y^j}{\sqrt{(x_t^i - l_x^j)^2 + (y_t^i - l_x^j)^2}} \quad 0\right]. \qquad (11)$$

3.1.3. Gözlem modeli: AMCL poz bilgisi

AMCL algoritması, robotun pozunu hesaplamak için odometri ve LiDAR bilgilerini kullanan özel bir parçacık filtresidir (Dellaert vd 1999). Bu projede önerilen yerelleştirme yöntemi EKF'nin güncelleme adımında AMCL algoritmasının pozundan yararlanmaktadır. Kullanılan amcl çıktısı aşağıdaki forma sahiptir.

$$p_t^{i,amcl} = p_t^i + \eta_t \qquad (12)$$

AMCL tahmininin gürültüsü η_t'dür.

3.2. İlklendirme Algoritmalarının Formülasyonu

Bilinmeyen robot pozisyonu $\boldsymbol{x} \in R^n$ olsun, $\boldsymbol{a}_i \in R^n, i = 1, \cdots, m$ bilinen sensor pozisyonları (anchors), ve $r_i = \|\boldsymbol{x} - \boldsymbol{a}_i\| + w_i$ kaynak ile i. anchor arasındaki gürültülü uzaklık ölçümü olsun, gürültü terimini ifade eder. Gürültünün i.i.d. varsayıldığı durumda gözlem olasılığının maksimize edildiği kaynak bulma problemi aşağıdakine eşittir:

$$minimize \quad \sum_{i=1}^{m} \left| \|\boldsymbol{x} - \boldsymbol{a}_i\|^p - r_i^p \right|^q. \tag{13}$$

p = 2, q = 2 durumu bu çalışmanın ana ilgi alanıdır ve maliyet fonksiyonu SR-LS algortimasındakullanılana denk gelmektedir (Beck vd 2008). SR-LS'deki maliyet fonksiyonu olasılıksal bir fonksiyon değildir fakat uygulanması kolay ve işlem karmaşıklığı az olup robotic uygulamalarına uygundur:

$$minimize \quad \sum_{i=1}^{m} \left| \|\boldsymbol{x} - \boldsymbol{a}_i\|^2 - r_i^2 \right|^2. \tag{14}$$

Bu problemin kesin çözümü aşağıdaki gibi verilmektedir:

$$\begin{array}{ll} minimize & \|\mathbf{Ay} - \mathbf{b}\|^2 \\ subject\ to & \mathrm{y^T Hy} + 2\mathrm{c^T y} = 0, \end{array} \tag{15}$$

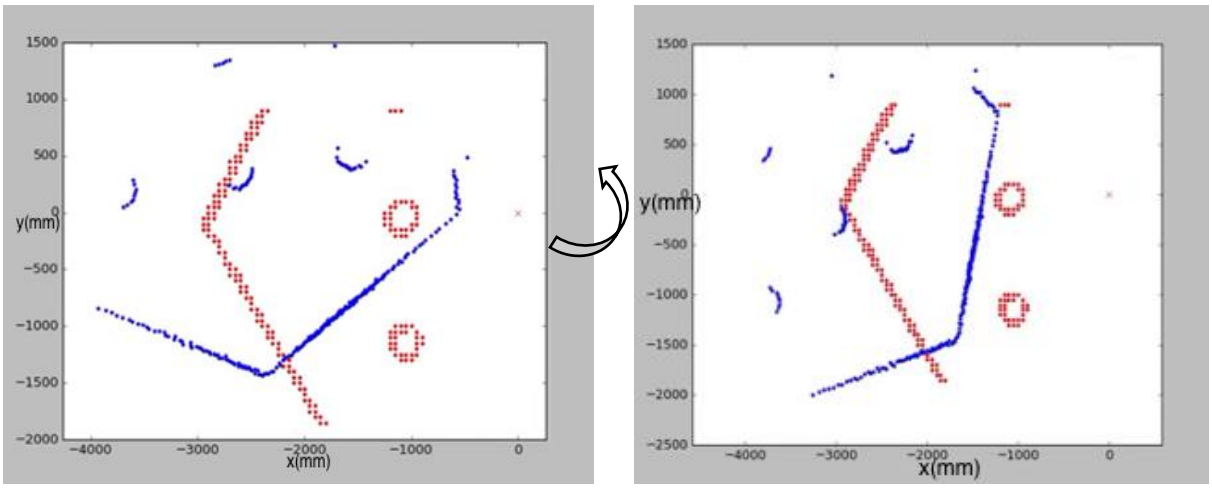
$$\mathrm{A} = \begin{bmatrix} -2a_1^T & 1 \\ \vdots & \vdots \\ -2a_m^T & 1 \end{bmatrix}, \mathrm{b} = \begin{bmatrix} r_1^2 - \|a_1\|^2 \\ \vdots \\ r_m^2 - \|a_m\|^2 \end{bmatrix}, H = \begin{bmatrix} I_n & 0_n \\ 0_1 & 0 \end{bmatrix}, \text{and } c = \begin{bmatrix} 0_n \\ -0.5 \end{bmatrix}.$$

Aşağıdaki polinomun bir kökü α^*'dır.

$$\hat{y}(\alpha)^T H \hat{y}(\alpha) + 2c^T \hat{y}(\alpha) = 0, \quad \text{ve } \alpha \in \left(-\frac{1}{\alpha_1}, \infty\right), \ \hat{y}(\alpha) = (\alpha \mathrm{H} + \mathrm{A}^T\mathrm{A})^{-1}(\mathrm{A}^T\mathrm{b} - \alpha \mathrm{c}).$$

Ve α_1 ise (H, A^TA)'nın genişletilmiş eigen değerlerinin maksimumudur (Golub, G. ve Van Loan, C. 1996).

Önerilen ilklendirme yöntemi UWB ve LiDAR verilerini kullanır. Başlatma işlemini otomatikleştirmede UWB sensörlerinden robotun mevcut konumunu kaynak konumlandırma algoritması aracılığıyla tahmin etmek için yararlanılır. UWB verisi sadece konum bilgisini bulmak için kullanılabilir. Bununla birlikte, bunlardan elde edilen konum bilgisi, etrafındaki harita verilerini eşleştirmek için bir referans noktası olarak kullanılır ve mevcut LiDAR taramaları, böylece robotun oryantasyonu hesaplanabilir. Bunu başarmak için, LiDAR tarama verileri kademeli olarak on derece döndürülür ve tüm noktalar, her yinelemede Şekil 3.1'de gösterildiği gibi harita verileriyle karşılaştırılır. 360 derece üzerinde, her LiDAR uç noktasının haritadaki en yakın noktaya olan mesafesi tekrar tekrar hesaplanır. Noktaların mesafelerinin toplamının en düşük olduğu açı değeri robotun yönünü verir. Başlatma algoritmasının kodu Tablo 3.1'de verilmiştir.



Şekil 3. 1. Her yinelemede LiDAR tarama verilerinin on derece dönüşü. Mavi ve kırmızı noktalar, sırasıyla tarama ve harita verilerine karşılık gelir.

Tablo 3. 1. İlklendirme algoritmasının kodu

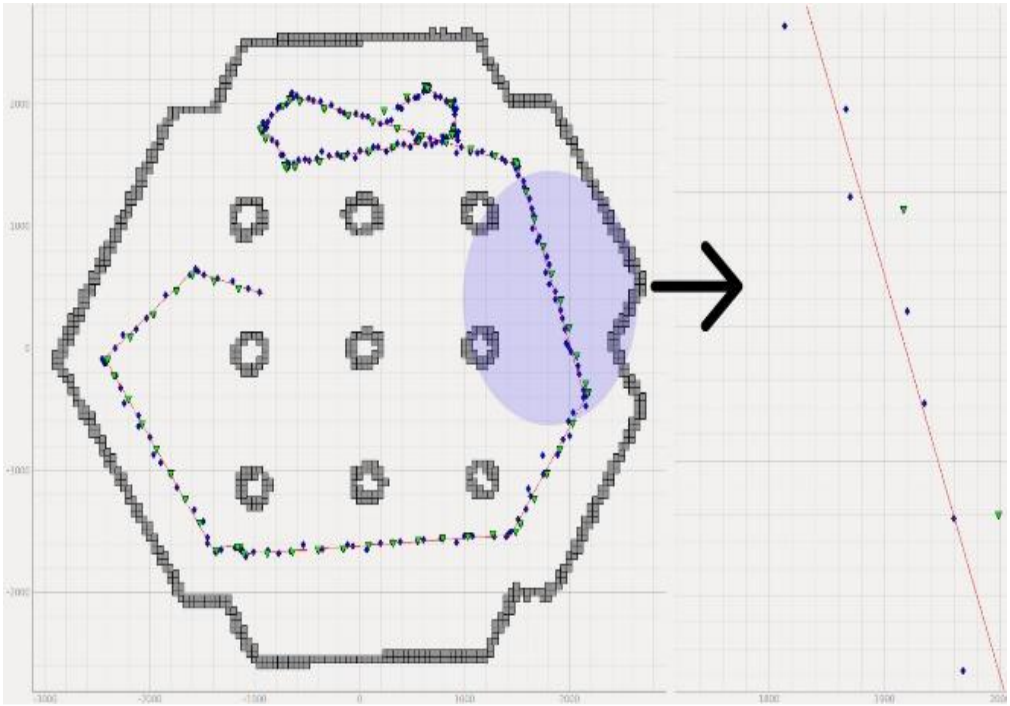
```
Girdi: M = Harita Array , L = Lidar Array
Çıktı A = robot açısı
1.  A = 0
2.  min_hata = sonsuz
3.  for i:=1 to 360
4.  if i % 10 = 0
5.  L[] i açısı kadar dön
6.  Toplam_uzaklık = 0
7.  for j = L[] her eleman
8.  min_uzaklık = sonsuz
9.  for k = M[] her eleman
10. if min_uzaklık > k'nın j'ye
11. min_uzaklık = k'nın j'ye
12. total_uzaklık+= min_uzaklık
13. if min_hata > total_uzaklık
14. A = i
15. min_hata= total_uzaklık
```

3.3. ROS Ekosistemi ve Sistem Akışı

Konum bilgisi, navigasyon sistemleri için en önemli bilgidir. UWB sensörleri, düşük gürültülü bileşenlerle uzaklık bilgisi sağladıkları ve çok yolluluk hatasına dirençli oldukları için robotik konumlandırma uygulamaları için çok iyi bir seçimdir. Dahası, bilgilerinin odometre bilgileriyle birleştirilmesi, zorlu çevre koşulları için daha sağlam bir çözüme sahip olmalarını sağlar. Bununla birlikte, UWB sensörleri ve bunların diğer sensör türleri ile ve farklı iç mekân uygulamaları için birleşmesi ROS ekosisteminde yeterince incelenmemiştir (ROS 2022). Bu nedenle, bu bölümde ROS modüllerinin uygulanma adımları açıklanacaktır.

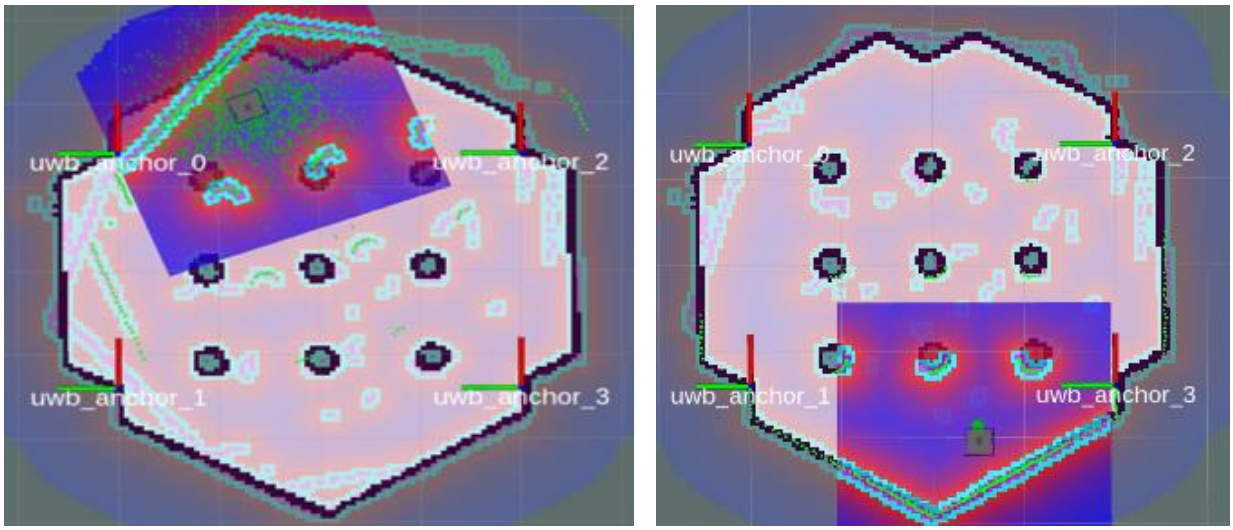
Pozyx UWB sensörleri (pozyx 2022) simülasyonlar ve gerçek testler aracılığıyla kullanılmaktadır. Bu nedenle, ROS'taki her UWB çapasından elde edilen menzil bilgilerinin yayınlanması gerekir. Bununla birlikte, robot konumunu belirlemek için, 2B ve 3B için en az 3 veya 4 farklı sensörden gelen uzaklık bilgisi gereklidir. Ayrıca, sensörler rastgele arızalanabilir ve hatalı sensör bilgilerinin algılanması gerekir. Böylece hedef_id dizisi, mesafe dizisi ve damga dizisini içeren yeni bir mesaj türü oluşturulur. Her sensör için mesaj, sensör kimliği, aralığı ve ölçüm süresini içerir ve "uwb_data_topic" olarak yayınlanır. Geliştirdiğimiz kodlar GitHub web sitesinde bulunabilir

(Bostanci vd. 2021). Ek olarak, simülasyon amacıyla, yeni bir UWB simülasyon düğümü oluşturulur ve "gazebo / model_states" e abone olur ve robotun simülasyon üzerindeki gerçek konumunu alır. Bu adımdan sonra her sensörden gerçek lokasyon ile ilgili uzaklık bilgisi hesaplanır ve üzerine Gauss gürültüsü eklenir. Daha sonra aynı mesaj türü kullanılarak "uwb_data_topic" ile paylaşılır. UWB ve odometri sensörlerinden ve amcl pozundan gelen bilgiler sırasıyla "uwb_data_topic" konusu, "odom" konusu ve "amcl_pose topic" yoluyla elde edilir. "Odom" konusu 30 Hz'de çalışır. Bununla birlikte, "uwb_data_topic" ve "amcl_pose topic" sırasıyla 5 Hz ve 2 Hz ile çalışır, böylece pozu tahmin etmek için her 0,033 s'lik odom bilgisi kullanılır ve bu, her 0,2 saniyede bir UWB verileriyle ve her 0,5 saniyede bir amcl verileriyle doğrulanır. Bundan sonra robotun pozu alınır ve adı "localization_data_topic" olan bir yayıncı ile yayınlanır. Kodlar GitHub web sitesinde bulunabilir (Bostanci vd. 2021). Şekil 3.2 simülasyon sonuçlarını göstermektedir. Eklenen Gauss gürültüsü, standart sapma gerçek mesafelerin %1.5'i olacak şekilde düzenlenmiştir (pozyx 2022). Birkaç yörünge için önerilen yöntemin Ortalama Karekök Hatası (RMSE) 3 cm iken AMCL'nin RMSE'si 6 cm'dir



Şekil 3. 2. Mavi elmaslar, yeşil üçgenler ve kırmızı çizgi, sırasıyla bizim yöntemimize göre tahmini yörünge, amcl'ye göre tahmini yörünge ve gerçek yörüngedir.

UWB node'lu kaynak konumlandırma algoritması konumu sağlar. Bununla birlikte, ilk pozun haritaya göre de bir yönelime ihtiyacı vardır ve sistem tek bir konumla yönlendirmeyi tahmin edemez. Böylece harita ve LiDAR tarama eşleşmesi uygulanır. İlk bakış açısı tahmininden sonra, poz "initial pose" konusunda yayınlanabilir. Şekil 3.3, robotun Rviz'de başlatmadan önceki ve sonraki duruşunu göstermektedir. Otonom başlatmadan önce, gri dikdörtgenle gösterilen robot konumu yanlıştır ve yeşil noktalar ve çizgilerle gösterilen tarama verileri, gösterildiği gibi siyah noktalar ve çizgilerle gösterilen gerçek harita ile eşleşmez (Şekil 3.3a). Başlatma sonrasında, tarama verisi ve harita Şekil 3.3b'de gösterildiği gibi uygun şekilde eşleştirilir.

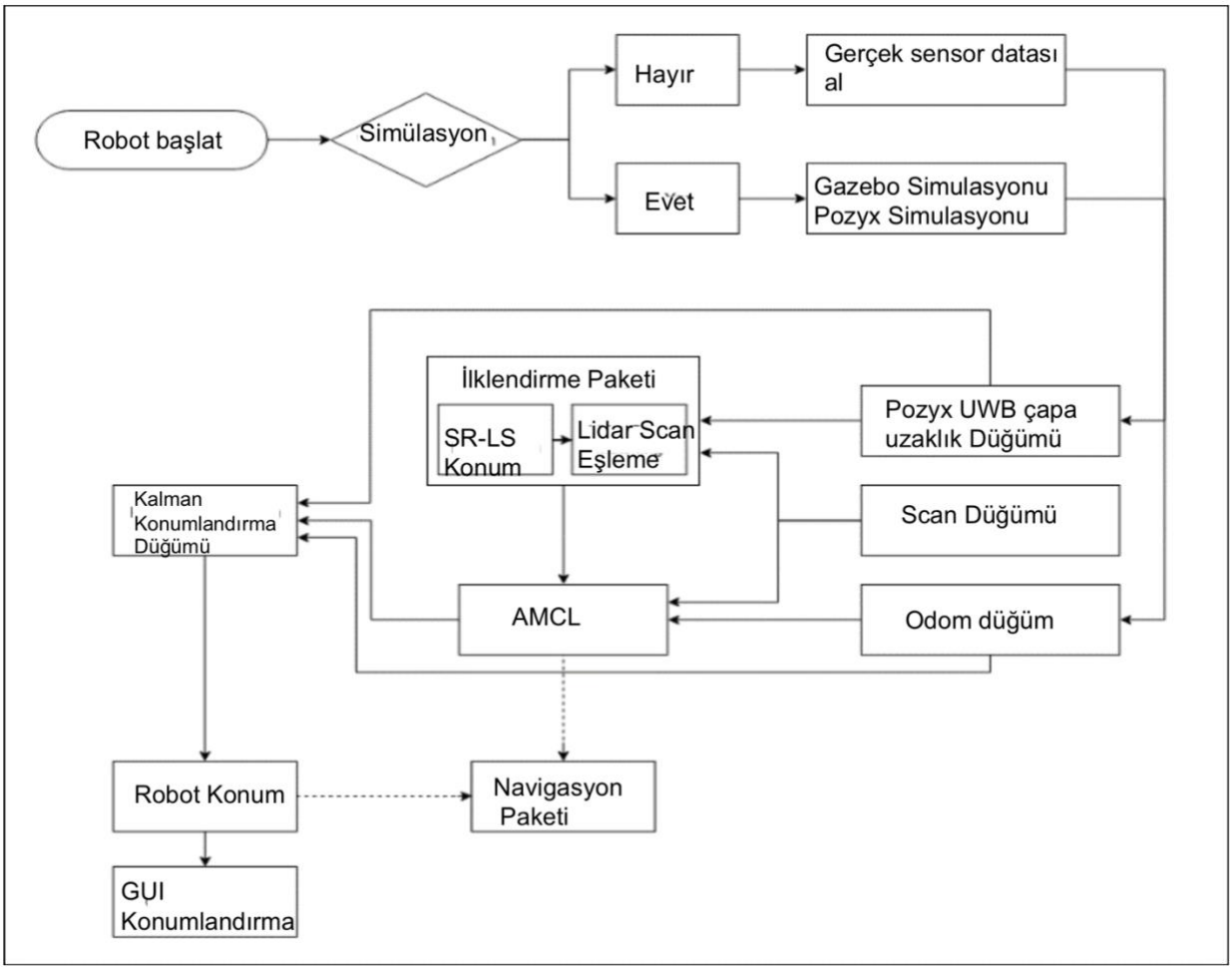


a) b)

Şekil 3. 3. a) Robot poz verir ve otomatik başlatmadan önce LiDAR'dan tarama yapar. b) Robot poz verir ve LiDAR otomatik başlatmadan sonra tarar.

Son olarak, tüm sistemin akış şeması Şekil 3.4'te gösterilmektedir. Farklı görevler için kullanılabilmesi için tam bir simülasyon ve gerçek test ortamları yaratılmıştır. Robot başlar ve ardından simülasyon veya gerçek dünya uygulaması seçilir. Uygulama simülasyonda çalışıyorsa, Gazebo simülasyonu ve Pozyx simülasyonu başlatılır ve bu iki simülatör sentetik sensör verilerini ve harita verilerini sağlar. Uygulama gerçek dünyada çalışıyorsa gerçek sensör verileri elde edilir. Sentetik verileri veya gerçek veri başlatma paketini elde ettikten sonra, otonom başlatmayı tam filtrelemek için UWB menzil verilerini ve LiDAR tarama verilerini kullanır. Daha sonra AMCL algoritması, robot yörüngesi sırasında robot konumunu hesaplamak için bu bilgileri, odometriyi ve tarama verilerini alır. Kalman Filtresi yerelleştirme düğümü, hareket modelini ayarlamak için

odometri verilerini kullanır ve UWB aralık ölçümlerini ve AMCL poz bilgilerini kullanarak poz bilgilerini günceller. Son olarak, GUI haritayı ve mevcut robot konumunu gösterir. Gerekirse, gezinti yığını ya AMCL poz bilgisi ile ya da daha iyi poz bilgisi sağladığı kanıtlanmış Kalman Filtresinden elde edilen poz bilgisi ile beslenebilir.



Şekil 3. 4. Tüm sistemin akış şeması

3.4. Haritalandırma Algoritmaları

Robotların belirli güzergahlar üzerinde hareket etmelerini sağlamak amacı ile daha önceden hazırlanmış haritalar kullanılmaktadır. Günümüzde bu haritalar haritalandırma algoritmaları ile robotlar tarafından oluşturulmaktadır. Haritalandırma algoritmaları genellikle lidar ya da laser range finder sensörleri ile etraftaki nesnelerin robota olan uzaklık bilgileri elde edilmektedir, elde

edilen bu bilgi ile robotun her hareketinde robotun konumu güncellenmekte ve yeni gelen uzaklık bilgileri diğer bilgilerin üzerine eklenmektedir. Sonuç olarak ise robotun hareket ettirildiği alanın haritası oluşmaktadır. Gmapping bugün ROS ile beraber en çok kabul gören veya altın standart olarak düşünülen slam/haritalandırma algoritmalarının başında yer almaktadır. Otonom yer robotlarında en çok tercih edilen slam/haritalandırma algoritmasının gmapping olmasının başlıca sebebi gmapping algoritmasının enkoder verisi ile lidar verisini birleştirerek Rao-Blackwellized partical filtre ile birleştirerek veri kaybını ve robot pozisyonun diğer slam algoritmalarına göre daha doğru tahmin etmesidir. Ayrıca yer araçlarında enkoder verisinin erişilebilir bir veri olması gmapping'in popüler olmasını sağlamaktadır. Hector slam ve enkoder kullanmayan diğer slam/haritalandırma algoritmaları sadece lidar verisine bakarak lidar'dan elde edilen verilerin birleştirilmesi ile haritayı oluşturmaktadır. Bu süreçte robotun ani hareketleri verilerin eşleştirilememesinden dolayı haritalarda ayrılıklar oluşmaktadır. Ancak gmapping'de lidar verileri enkoder ile desteklenmektedir bu yüzden haritalar çok daha verimlidir. Ayrıca enkoder verisinin üzerindeki biriken gürültü gmapping'de partical filtre ile yok edilmektedir. Böylece her adımda enkoder üzerinde biriken hata azaltılmakta aynı zamanda da enkoder ile de harita desteklenmektedir.

Bu algoritmalarımızı ROS sistemini kullanarak test ettik ve ROS üzerinde gmapping'i kullanmak için bazı parametreleri ve de konuları kullandık. Aşağıda bunların açıklamasını bulabilirsiniz.

Abone Olunan Konular

tf (tf/tfMessage)

Robot üzerindeki parçaların birbirlerine olan konumlarını ve açılarının bilgisini elde etmek için tf(transformation) konusu dinlenmektedir. Böylece robotun hareket bilgisi elde edilmektedir.

scan (sensor_msgs/LaserScan)

Lidar ile gelen veriler dinlenmektedir.

Yayınlanan Konular

map_metadata (nav_msgs/MapMetaData)

Haritanın genişlik yükseklik, güncellenme zamanı, çözünürlük ve merkezinin bilgilerini taşıyan konudur

map (nav_msgs/OccupancyGrid)

harita bilgisini içeren haritayı oluşturan gridlerin doluluk bilgisini saklayan konudur

~entropy (std_msgs/Float64)

Robotun pozu üzerindeki dağılımın entropisinin tahmini (daha yüksek bir değer daha büyük belirsizliği gösterir)

Servisler

dynamic_map (nav_msgs/GetMap)

Anlık harita bilgisini elde etmek için konuya bağlanmak yerine anlık olarak almanızı sağlayan servistir.

Parametreler

~throttle_scans (int, default: 1)

Her gelen veri kümesinden n tanesini çıkarmaktadır (daha fazla taramayı atlamak için daha yüksek bir sayıya ayarlayın)

~base_frame (string, default: "base_link")

Robotun bağlı olduğu çerçevenin ismi

~map_frame (string, default: "map")

Haritaya eklenen çerçevenin ismi

~odom_frame (string, default: "odom")

Odometrenin ekli olduğu çerçevenin ismi

~map_update_interval (float, default: 5.0)

Harita güncellemeleri arasındaki süredir (saniye cinsinden). Bu sayının düşürülmesi, işlem sayısının artmasına neden olmakla beraber, haritayı daha sık güncellemektedir.

~maxUrange (float, default: 80.0)

Haritayı oluşturmak için kullanılan maksimum lazer uzaklığıdır

~sigma (float, default: 0.05)

Tarama eşleştirme işlemi için standart sapma

~kernelSize (int, default: 1)

Tarama eşleştirme işlemi için arama penceresi boyutu

~lstep (float, default: 0.05)

Tarama işlemi için verilerin lineer olarak öteleneceği uzaklık

~astep (float, default: 0.05)

Tarama verilerinin açısal olarak döndürülme miktarı

~iterations (int, default: 5)

Tarama eşleşmesindeki iyileştirme adımlarının sayısıdır

~lsigma (float, default: 0.075)

Tarama eşleştirme işlemi için standart sapma

~ogain (float, default: 3.0)

Yeniden örnekleme etkilerini yumuşatmak için olasılığı değerlendirirken kullanılacak kazanç

~lskip (int, default: 0)

Her taramada atlanacak ışın sayısı. Bir eşleşme hesaplamak için yalnızca her (n + 1). Lazer ışınını alın (0 = tüm ışınları alın)

~minimumScore (float, default: 0.0)

Açık alanlarda sınırlı lazer taramasından dolayı yanlış eşleştirmeleri önlemek amacıyla kullanılır

~srr (float, default: 0.1)

Odyometri üzerindeki lineer gürültü bileşeni (rho/rho)

~srt (float, default: 0.2)

Odyometri üzerindeki açısal gürültü bileşeni (rho/theta)

~str (float, default: 0.1)

Odometry error in rotation as a function of translation (theta/rho)

~stt (float, default: 0.2)

Odometry error in rotation as a function of rotation (theta/theta)

~linearUpdate (float, default: 1.0)

Robot, yeni ölçümleri en az bu parametredeki kadar hareket etmişse işler

~angularUpdate (float, default: 0.5)

Robot, yeni ölçümleri en az bu parametredeki kadar dönmüşse işler

~temporalUpdate (float, default: -1.0)

Son işlenen tarama, saniye cinsinden güncelleme süresinden daha eskiyse bir tarama gerçekleştirin. Sıfırdan küçük bir değer, zamana dayalı güncellemeleri kapatır.

~resampleThreshold (float, default: 0.5)

Neff tabanlı yeniden örnekleme eşiği

~particles (int, default: 30)

Filtredeki partikül sayısı

~xmin (float, default: -100.0)

Başlangıçtaki x eksenindeki minimum harita büyüklüğü (metre)

~ymin (float, default: -100.0)

Başlangıçtaki y eksenindeki minimum harita büyüklüğü(metre)

~xmax (float, default: 100.0)

Başlangıçtaki x eksenindeki maksimum harita büyüklüğü(metre)

~ymax (float, default: 100.0)

Başlangıçtaki y eksenindeki maksimum harita büyüklüğü (metre)

~delta (float, default: 0.05)

Harita çözünürlüğü (in metres per occupancy grid block)

~llsamplerange (float, default: 0.01)

Olasılık için çeviri örnekleme aralığı

~llsamplestep (float, default: 0.01)

Olasılık için çeviri örnekleme adımı

~lasamplerange (float, default: 0.005)

Olasılık için açısal örnekleme aralığı

~lasamplestep (float, default: 0.005)

Olasılık için açısal örnekleme adımı

~transform_publish_period (float, default: 0.05)

Dönüşüm(tf) yayınları arasındaki süre (saniye cinsinden). Yayın dönüşümlerini devre dışı bırakmak için 0'a ayarlayın.

~occ_thresh (float, default: 0.25)

Gmapping doluluk değerlerinde eşik. Daha fazla doluluk oranına sahip hücreler dolu kabul edilir (yani sonuçta elde edilen sensor_msgs / LaserScan'de 100'e ayarlanır)

~maxRange (float)

Sensörün maksimum aralığı. Sensörün menzilinde engel bulunmayan bölgeler haritada boş alan olarak görünecekse, maxUrange <gerçek sensörün maksimum aralığı <= maxRange ayarlayın.

3.5. Güzergah Planlama Algoritmaları

Turtlebot3 ile ROS kullanarak yaptığımız denemelerde navigation stack paketinden yararlanılmıştır. Navigation stack paketi ile gelen güzergâh tayin algoritmaları iki şekilde ayrılmaktadır. Bunlar local planners (kısmı planlayıcılar) ve global planners (genel planlayıcılar)dır.

ROS navigation paketi ile genel planlayıcı olarak tanımlanan algoritmalar Dijkstra's ve A* algoritmalarıdır (Peng 2015). Kısmi planlayıcılar ise carrot planner, base local planner ve dwa local plannerdır. Denemelerde kısmı planlayıcı olarak DWA (Dynamic Window Approach) algoritması kullanılmıştır (Fox vd 1997). Ayrıca navigation paketi içerisinde amcl, clear_costmap_recovery, costmap_2d, fake_localization, map_server, move_base, move_base_msgs, move_slow_and_clear, nav_core, navfn, rotate_recovery, voxel_grid paketleri de yer almaktadır.

Genel planlayıcılar daha önceden hazırlanmış grid base (ızgara tabanlı) haritaları kullanarak belirlediğiniz harita çözünürlüğü ölçüsünde robotun takip etmesi gereken rotayı planlamaktadır. A*, Dijkstra algoritmasının özel bir durumudur, tek fark, A* 'nın diğerlerinden daha iyi olduğu varsayılan düğümlere öncelik veren sezgisel bir işlev kullanarak daha iyi bir yol aramaya çalışmasıdır, Dijkstra ise tüm olası yolları keşfetmeye çalışmaktadır ve bu sebepten dolayı Dijkstra'nın genel rotayı tayin etmesi daha uzun zaman almaktadır. Yukarıda bahsedilen nedenlerden dolayı genel planlayıcı algoritması olarak A* algoritması kullanılmaktadır.

Robotların haritaya bağlı olarak gitmesi gereken konumlar hesaplandıktan sonra lidar ve ultrasonic sensörlerlerinden gelen verilere göre dinamik engellere karşı yeni rotalar hazırlamaları gerekmektedir. Kısmi planlayıcıların görevi kısa mesafede ortaya çıkan dinamik engellere göre genel planlayıcıların oluşturduğu güzergaha yakın bir şekilde robotu engellere çarpmadan ortamda hareket ettirmektir. Genel planlayıcılar rotalarını bulunan konumdan bitiş konumuna

kadar hazırlarken, kısmı planlayıcılarda bu mesafe 0.3 ile 0.1 metre arasında değiştirebileceğiniz bir parametre olarak verilmektedir (forward_point_distance). Yukarıda belirtildiği gibi farklı kısmi planlayıcı algoritmaları ROS ile beraber hazırlanmıştır. Ancak diferansiyel sürüş sistemlerinin dinamiklerine en yakın ve uygun algoritma olarak DWA algoritması önerilmektedir. DWA algoritması birçok parametre ile robot hareketlerini kontrol etmektedir. Bunların başlıcaları aşağıda belirttildiği gibidir.

Robot Yapılandırma Parametreleri

~<name>/acc_lim_x (double, default: 2.5) (m/s^2)

robotun x ekseni yönündeki en yüksek ivmesi

~<name>/acc_lim_y (double, default: 2.5) (m/s^2)

Robotun ileri y ekseni yönündeki en yüksek ivmesi

Diferansiyel sürüş sistemine sahip robotlarda buradaki hız ve ivme önemsizdir. Çünkü robotlar yalnızca robota göre x ekseni üzerinde hareket etmektedir.

~<name>/acc_lim_th (double, default: 3.2) (rad/s^2)

Robotun dönüş ivmesini belirtmektedir

~<name>/max_vel_trans (double, default: 0.55) (m/s)

Robotun maximum mutlak hızını belirtmektedir. Diferansiyel sürüş sistemine sahip robotlarda bu hız max_vel_x ile sınırlanmaktadır. Çok yönlü(omnidirectional) robotlarda ise x ve y ekseni üzerindeki hızlarının birleştirilmesi ile oluşturulmaktadır.

Bu parametrenin yüksek değerlerde olması sakıncalıdır.

~<name>/min_vel_trans (double, default: 0.1) (m/s)

Robotun minimum mutlak hızını belirtmektedir. Bu değerin çok yüksek olması aynı zamanda robotunuz kalkış hızının yüksek olmasına sebebiyet verecektir.

~<name>/max_vel_x (double, default: 0.55) (m/s)

Robotunuzun x ekseni yönündeki maximum hızını ifade etmektedir. max_vel_x değerinin max_vel_trans değerinden büyük olması bir anlam ifade etmemektedir.

~<name>/min_vel_x (double, default: 0.0) (m/s)

Robotun kendisine göre x ekseni üzerindeki minimum hızını ifade etmektedir. Geri manevra yapacak robotlar için bu değerin negatif olması gerekmektedir.

~<name>/max_vel_y (double, default: 0.1) (m/s)

Robotun kendisine göre y ekseni üzerindeki maksimum hızını ifade etmektedir.

Diferansiyel sürüş sistemine sahip robotlarda bu parametre bir anlam ifade etmemektedir

~<name>/min_vel_y (double, default: -0.1) (m/s)

Robotun kendisine göre y ekseni üzerindeki minimum hızını ifade etmektedir.

~<name>/max_rot_vel (double, default: 1.0)

Robotun maksimum mutlak dönüş hızını ifade etmektedir. Buradaki değerlerin büyük olması robotunuzun çok yüksek hızlarda dönebilmesine imkan sağlayacağı için dikkatli bir şekilde artırılması gerekmektedir.

~<name>/min_rot_vel (double, default: 0.4)

Robotun minimum mutlak dönüş hızını ifade etmektedir.

Hedef Tolerans Parametreleri

~<name>/yaw_goal_tolerance (double, default: 0.05) (rad)

Robotun hedef noktadaki kafa açısındaki son tolerans değeridir. Encoder ve motor hassasiyetine bağlı olarak 0.1 derece gibi çok küçük hata paylarında robotlar hedef noktada fazlası ile hareket etmektedir.

~<name>/xy_goal_tolerance (double, default: 0.10) (metre)

Robotların hedef noktaya olan uzaklık toleranslıdır. Bu değer hedef konumdan robotun konuma olan hatayı temsil etmektedir.

İleri Simülasyon Parametreleri

~<name>/sim_time (double, default: 1.7) (s)

Rotayı saniye cinsinden ileriye doğru simüle etmek için geçen süredir. Robotun belirlenen hız değerleri ile belirlediğiniz saniye sonunda varacağı noktadır.

~<name>/sim_granularity (double, default: 0.025) (m)

Belirli bir rota üzerindeki noktalar arasında atılacak metre cinsinden adım boyutudur. Robotunuzun path üzerinde takip etmesi gereken noktalar arasındaki mesafedir. Buradaki parametre motor ve encoder hassasiyetine bağlı olarak değiştirilmesi gerekmektedir.

~<name>/vx_samples (integer, default: 3) (adet)

X hız uzayını keşfederken kullanılacak örnek sayısı

~<name>/vy_samples (integer, default: 10) (adet)

Y hız uzayını keşfederken kullanılacak örnek sayısı

3.6. Arayüz Tasarlama

Arayüzler tasarlama iş paketi değişik sensörlerden gelen verileri alabilmek için geliştirilmiş programları kapsamaktadır. Bu amaçla ROS kullanılmıştır. ROS farklı süreçler (nodes) arasında bir ağ üzerinde mesaj aktarım yapısına sahip açık kaynaklı bir işletim sistemidir (ROS 2022). ROS'un sağladığı fayda farklı algılayıcıların ve donanımların kolaylıkla sisteme eklenmesine olanak sağlamasıdır. Bahsedilen faydalar, farklı sensör füzyon tekniklerinin kolay uygulanmasına ve çeşitli donanım bileşenlerinin birleştirilmesine yol açar. Ayrıca ROS, robotik alanında güçlü bir araç olan Gazebo gibi bir simülatöre sahiptir (Yılmaz 2019). ROS, Ubuntu işletim sistemine kurulmuş ve aşağıda belirtilen mesajlar ve de algoritmalar python dilinde yazılıp ROS ile entegre edilmiştir. Bu yazılım desteği ve de geliştirilen arayüzler Jetson Nano veya Raspberry Pi gibi işlemcilerde kullanılabilecek durumdadır. Ve hali hazırda Jetson Nano'ya kurulmuştur.

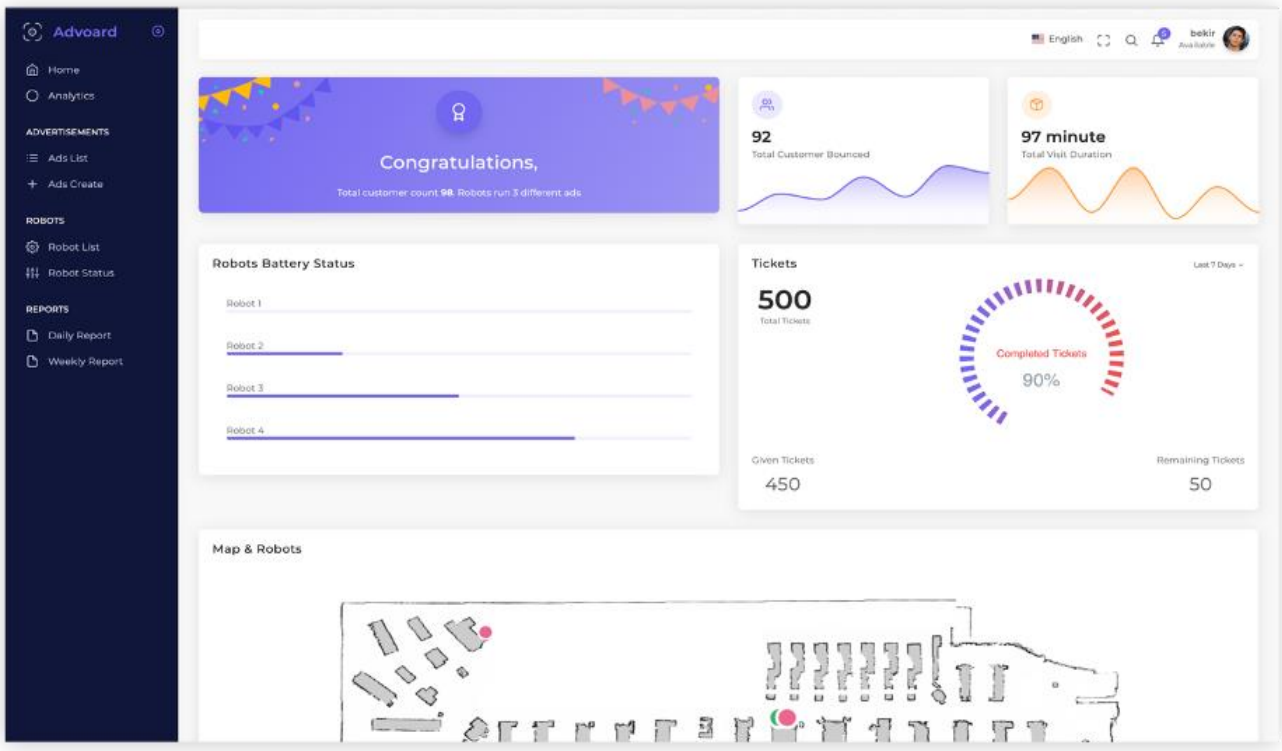
Pozyx UWB algılayıcıların (pozyx 2022) hem gerçek testlerde hem de benzetimlerde kullanılması hedeflendi, bu yüzden her birinden gelen uzaklık ölçümleri ROS ile paylaşılmalıydı. Robot konumunun belirlenmesi için en az 3 veya 4 algılayıcıdan gelen ölçümlerin kullanılması gerekir. Ek olarak bazı algılayıcılardan ölçüm alındığı sırada hatalar meydana gelebilir ve bunun bilinmesi gereklidir. Bu nedenle ROS'ta destination_id array, distance array and stamp array içeren yeni bir mesaj türü yarattık. UWB ölçüm verileri her bir algılayıcı tarafından "uwb_data_topic" altında bu mesaj türü ile paylaşılmaktadır. UWB sensörlerine dair mesaj türünün yaratılması ve de ROS ekosistemine entegresi daha önce literatürde yoktu ve geliştirdiğimiz kodları github sayfamızda genel erişim için paylaştık (Bostanci vd. 2020). Ek olarak benzetim yapmak için "gazebo/model_states" node'na abone olan ve robotun gerçek konumunu paylaşan yeni bir UWB benzetim node'u yarattık. Bu adımdan sonra gerçek konumla bağlantılı olan tüm uzaklık ölçümleri hesaplanıp üzerine Gaussian gürültüsü eklendikten sonra "uwb_data_topic" kullanılarak aynı mesaj tipiyle paylaşılmaktadır. Bu yöntem UWB algılayıcıları ve onlardan gelen verileri Gazebo ortamında benzetimini sağlamaktadır. Benzer şekilde Lidar sensöründen gelen dataları toplamak ve de paylaşmak için scan topic ve de odometre sensöründen gelen bilgileri toplamak için odom topic kullanmaktayız.

UWB algılayıcılarından verileri “uwb_data_topic” ile ve odometreyi “odom” ile iki farklı nodedan almaktayız. “odom” 30Hz ile çalışmaktadır. Fakat “”uwb_data_topic” 5Hz ile veri göndermektedir. Böylece odom bilgisi 0.033 saniyede bir robot pozisyonunu güncellemekte ve 0.2 saniyede bir “uwb_data_topic” ile birlikte onaylanmaktadır. Bu adımların ardından robot pozisyonu “localization_data_topic” altında ROS üzerinde paylaşılmaktadır. Geliştirdiğimiz farklı konumlandırma algoritmalarını localization node altında kullandık. Bu node hem benzetim hem de gerçek veriler ile çalışabilmektedir. Yine geliştirdiğimiz ilklendirme algoritmalarından elde ettiğimiz verileri initial_pose topic altında bütün sistemle paylaşmaktayız. Ve topiclerden paylaşılan bilgilere nodelar abone olup kullanabilmekte. Bu yapıyı eklediğimiz her yeni sensör ve de geliştirdiğimiz ya da kullandığımız her algoritma için kolayca tekrarlayabilmekteyiz.

Yaptığımız simülasyonlarda gerçek UWB ölçümlerine standart sapması gerçek uzaklığın %1.5’i olan Gaussian noise ekledik. Gerçek testlerde ise UWB ölçümlerinin 30 m x 30 m bir alan içinde yaklaşık 4 cm ile 8 cm arasında hatalı uzaklık ölçümleri verdiğini gördük. Arayüzler tasarlama iş paketindeki 10 cm’lik bir hata hedeflemiştik ve de hedefimize ulaştık.
Robotun konumunu, algoritmaların performansını gözlemleyebilmek için ROS’ta Gazebo gibi bir simülasyon ortamı ya da Rviz gibi bir görsel arayüz bulunmaktadır. Fakat ilerde robotun uzaktan denetlenebilmesi için bir web arayüzü tasarlamış bulunmaktayız.

Robotların internet aracılığı ile kontrol edilmesini verilerin ve uyarların kullanıcılara iletilmesi amacıyla bir web kontrol paneli hazırlanmıştır. Bu amaç doğrultusunda web sitesini oluşturmak için html, css ve javascript kullanılarak bir web arayüzü hazırlanmıştır. Aynı zamanda verilerin internet üzerinde güvenli bir şekilde saklanması ve gerçek zaman ile verilerin eksiksiz bir şekilde internet ile paylaşılması için ise bir none sql veri tabanı olan firebase tercih edilmiştir. Bunlara ek olarak php ile de kullanıcı verileri çapraz olarak farklı bir sunucuda şifreli olarak konumlandırılmaktadır. Robot tarafında verilerin internete gönderilmesi için python yazılım dili kullanarak bir paket hazırlanmıştır. Hazırlanan paketler ile robotun adı, kullanıldığı map, mapin adresi gibi robot bilgileri ile motor markası, lidar markası gibi robotu oluşturan parçaların bilgisi de sunucuya aktarılmaktadır. Robot her başlangıçta bu verileri günceller böylece herhangi bir konfigürasyon değişikliğinden sonra robotun yeniden başlatılması yeterli olmaktadır. Her robot veri güvenliği için kendi kullanıcısına sahiptir, robotun kullanıcı bilgisi ve şifresi robot içerisinde konumlandırılmıştır. Robot kullanımı için öncelikle robotun kullanıldığı alanın slam algoritması ile haritasının çıkarılması gerekmektedir. Ardından kullanılacak haritalar daha önceden sunucuya yüklenerek ve gerekli parametreler girilerek robotun web arayüzü ile kullanılmasına

hazırlanmaktadır. Robot'a web arayüzü ile hedeflenen konumlar ve zaman bilgisi girilerek robot belirtilen konumlara yönlendirilebilir. Robot listeleme bölümünde konfigürasyonu girilmiş robotlar listelenir. Seçilen robotların bilgileri böylece web arayüzü ile kullanıcıya sunulur. Aynı zamanda robotun şarj ve konum bilgisi gibi anlık verileri de Firebase'in gerçek zamanlı veri paylaşımı ile kullanıcıya veriler anlık olarak aktarılır. Harita üzerinde robot konumlarını görselleştirmek için ayrıca leaflet.js kütüphanesi kullanılmıştır. Bu kütüphane ile haritalar kullanıcıya az boyutlarda dosyalar indirilerek hızlı bir şekilde ulaştırılmaktadır. Ayrıca map üzerinde daha önceden tanımlanabilen istasyonlar ile robotun daimî olarak gitmesi gereken konumlar başka bir web sayfası ile kullanıcıya ulaştırılmaktadır. Kullanıcıların robot ve harita bilgileri üzerinde değişiklik yapabilmeleri için yetkilendirme sistemi uygulanmıştır. Bir robot üzerinde herhangi bir veriyi gözlemlemek için sadece sisteme üye olmak yeterli değildir. Ayrıca yönetici tarafından kullanıcılara bu yetkilerin sağlanmış olması bir şarttır. Böylece robot verileri ve gizlilik gibi konular sadece firebase üzerinden saklı kalmaktadır. Kullanıcı verileri çift veritabanı ile tamamen koruma altına alınmakta ve çapraz kontrol ile veri güvenlik 2 kat arttırılmaktadır. Veriler firebase yani sunucuya şifreli olarak gönderilmektedir ve böylece dataların sunucuya varıncaya kadar güvenliği böylece sağlanmaktadır. Geliştirdiğimiz web arayüzünün bir versiyonu Şekil 3.5'de görülebilir.



Şekil 3. 5. Web arayüzü

3.7. Şarj istasyonuna yanaşma algoritması

Testler iş paketinde çalışmalarımız devam ederken daha hassas bir güzergah planlama ve otonom sürüş algoritması geliştirilmiş ve gerçek ortam testleri yapılmıştır. Bu şarj istasyonunda şarj edilecek holonomik olmayan bir mobil platform için LiDAR ile özellik tabanlı bir kenetlenme stratejisi sunan bir algoritmadır. Şarj/kenetlenme istasyonu, LiDAR taramalarından tespit edilebilecek yalnızca V şeklinde bir yapıya sahiptir. Önerilen yerleştirme algoritması, V-şekil özelliğinin LiDAR ölçümleriyle bulunması, beşli polinom ile yol oluşturulması ve yolun oluşturulan noktalarını izlemek ve düzeltmek için Orantılı İntegral Türev (PID) uygulaması olmak üzere üç ana bölümden oluşur. Sentetik olarak oluşturulan V-şekli özelliği mevcut Lidar taramaları ile eşleştirilerek kenetlenme istasyonunun göreceli konumu tespit edildiğinde, beşli polinomun katsayıları hesaplanarak yörüngeler oluşturulur. Daha sonra, yol boyunca noktalardaki başlangıç durumları, durumlar ve sapma açıları girdi olarak alınır ve algoritma, PID kontrolü ile yolu takip ederken robotun sapma açısını düzeltir. Algoritma hem simülasyon ortamında hem de gerçek test ortamında test edildi. Platform, x ve y'de ± 2 cm, θ'de ± 1^o civarında olan iyi bir doğrulukla yanaşmaktadır. Önerilen strateji ile platformda veya şarj istasyonunda ek sensörler gerekli değildir. Üstelik robot, ışık olmadığında bile iyi bir doğrulukla hareket edebilir. Böylelikle otonom mobil robotik uygulamalarının önemli bir problemi için daha ucuz bir çözüm elde edilmektedir.

Sistem, iletişim için ROS kullanan bilgisayara bilgileri işlemek ve iletmek için bir ana bilgisayar ve bir mikro denetleyici olmak üzere iki ana sensörden oluşur. RPLidar ve odometri verileri sensör ölçümleri olarak kullanılmakta ve veriler ROS ile ana bilgisayara aktaran ESP32 ile toplanmaktadır. Duvarın yanında bulunan şarj istasyonunun (Şekil 3.6) üzerine V şeklinde bir unsur yerleştirilmiştir. Yerleştirme işlemini başlatmak için istasyona yakın bir yere gidebilmesi için robota önceden G-haritalama ile elde edilen harita verilerinden (Grisetti ve diğerleri, 2005) yaklaşık olarak konumu bilinmektedir. Yerleştirme işlemi tamamlandıktan sonra şarj istasyonunun hem pozitif hem de negatif terminallerinin Şekil 3.7'daki robotun pil terminalleriyle temas etmesi beklenir.

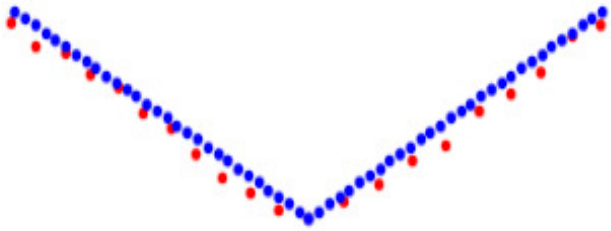
Şekil 3. 6. İstasyon

Şekil 3. 7. Robotun yanaşma tarafı

Yerleştirme algoritması, V-şekil özelliğinin LiDAR ölçümleri ile lokalizasyonu, quintic polinom ile yol üretimi ve yolun oluşturulan noktalarını izlemek ve yol boyunca açı düzeltme için Oransal-İntegral-Türev (PID) uygulaması olmak üzere üç ana bölümden oluşmaktadır.

LiDAR verileriyle yerelleştirme

Başarılı bir yanaşma elde etmek için öncelikle istasyonun tam konumu belirlenmeli, böylece ona giden bir yol oluşturulabilir. Bu, her biri 1°'ye karşılık gelen 360 örnekle 2 boyutlu nokta bulutları çıkaran LiDAR ölçümleriyle elde edilir. Ancak, özelliğin kesin şeklinin algoritma tarafından bilinmesi gerekir ki, bunu LiDAR ölçümleriyle eşleştirebilsin ve özelliği algılayabilsin. Bu nedenle, istasyonun üstüne yerleştirilen gerçek boyutlarına tam olarak uyan bilgisayarda 60 örnek ile sentetik olarak bir örnek özellik ölçümü oluşturulur. Eşleştirme algoritması, numuneler ve ölçümler arasındaki hatayı hesaplamak için öklid mesafelerinden yararlanır. Numune şekli de ölçülen her noktada 360° döndürülür ve her yineleme için eşleşen noktalar hesaplanır. Eşleşen özelliğin bir örneği ve ölçüm verileri Şekil 3.8'de görülebilir.



Şekil 3. 8. mavi: referans noktaları, kırmızı: LiDAR tarama datası

Yerleştirme sonunda daha doğru bir konum ve açı elde etmek için robot hedefi yörüngenin ortasına ayarlayabilir ve kendini düzeltmek için yerelleştirme segmentini daha yüksek bir hassasiyetle yeniden çalıştırabilir. Ancak, yerelleştirme işlemi, 360 LiDAR noktalarının her birinde referans özellik noktasını 360° döndürdüğünden, yaklaşık 35 saniye sürer ve bu da çok zaman açısından verimli değildir. Algoritmayı olduğu gibi çalıştırmak mümkün olsa da, ikinci adımda lokalizasyon işlemini yaklaşık 3 saniyeye kadar hızlandırmak için (gereksiz nokta ve döndürmeler ihmal edilerek) bazı değişiklikler yapılabilir. Bunu yapmak için robot orta noktaya hareket ettiğinde özelliğin nerede olacağını bilmek gerekir. Bu nedenle $\vec{r}_{o2o3}$ vektörünün ikinci karedeki (Şekil 3.9'daki orta durum) özelliğin koordinatlarını verdiği şu şekilde tanımlanmıştır (Özgören, 2020)

$$\{\vec{r}_{o2o3} = \vec{r}_{o2o1} + \vec{r}_{o1o3} = -\vec{r}_{o1o2} + \vec{r}_{o1o3}\} \qquad (15)$$

İki nokta arasındaki pruva açısı farkı şu şekilde tanımlanır:

$$\Delta\theta = \theta_2 - \theta_1 \qquad (16)$$

ve dönüşüm matrisi aşağıdaki gibi ile tanımlanır

$$\hat{C}^{(1,2)} = \begin{bmatrix} cos\Delta\theta & -sin\Delta\theta \\ sin\Delta\theta & cos\Delta\theta \end{bmatrix} \qquad (17)$$

Dönüşüm, onları ilk çerçeveye almak için uygulandığında, aşağıdaki denklemler elde edilir.

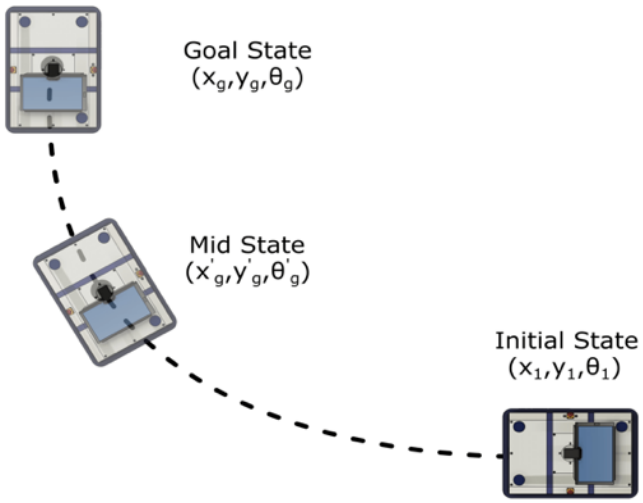
$$\vec{r}_{o2o3}^{(2)} = -\hat{C}^{(2,1)}\vec{r}_{o1o2}^{(1)} + \hat{C}^{(2,1)}\vec{r}_{o1o3}^{(1)} \quad (18)$$

$$\vec{r}_{o2o3}^{(2)} = \hat{C}^{(1,2)T}(\vec{r}_{o1o3}^{(1)} - \vec{r}_{o1o2}^{(1)}) \qquad (19)$$

$$\vec{r}_{o2o3}^{(2)} = \hat{C}^{(1,2)T}\begin{bmatrix} x_g - x_g' \\ y_g - y_g' \end{bmatrix} \qquad (20)$$

Bazı sadeleştirmelerden sonra, hedef noktasının koordinatları $\hat{C}^{(1,2)T} = \hat{C}^{(2,1)}$ bulunur.

Bu ihmal edilen özniteliklere, önerilen türevlerden robotun konumu ve yalpalaması alınarak karar verilir, böylece özelliğin konumu orta noktada tahmin edilir. Böylece robot Şekil 3.9'deki gibi orta durumdayken, tüm gereksiz LiDAR ölçümleri göz ardı edilebilir ve bunun yerine referans özellik dönüşleri bu sefer 360 dereceden çok daha az olacaktır. Bu nedenle, yineleme sayısı azaltılacaktır.



Şekil 3. 9. Robotun başlangıç durumu, orta durum ve hedef durumundaki konumu ve oryantasyonu

İki adımda bir hedefe ulaşmak için bir yörünge örneği

Özelliğin robota göre konumu bulunduğunda, artık bu amaca giden bir yol oluşturmak mümkündür. Bu, (Takashi ve diğerleri, 1989)'da açıklandığı gibi tek boyutlu robot hareketi için beşli polinom yaklaşımı ile gerçekleştirilecektir ve aşağıdaki gibi verilmiştir.

$$x(t)=a_0 +a_1t+a_2t^2 +a_3t^3 +a_4t^4 +a_5t^5 \qquad (21)$$

burada a_0, a_1, a_2, a_3, a_4 ve a_5 beşli polinom katsayılarıdır. Başlangıç ve durma noktaları bilindiğinden, başlangıç konumu, hız ve ivme sırasıyla x_s, v_s ve a_s olarak alınır. x_e, v_e ve a_e sırasıyla bitiş konumu, hız ve ivmedir. t = 0 anında $x(0)=a_0 =x_s$.

Denklemi (21) farklılaştırarak, aşağıdaki formülasyonlar elde edilir

$$x'(t) = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4 , \quad (22)$$

$$x'(0) = a_1 = v_s. \qquad (23)$$

Daha sonra (21) denkleminin tekrar t ile türevi alınırsa,

$$x''(t) = 2a_2 + 6a_3t + 12a_4t^2 , \quad (24)$$

$$x''(0) = 2a_2 = a_s \qquad (25)$$

a_0, a_1 ve a_2 (21), (23) ve (24) ve sınır koşulları kullanılarak hesaplanabilir. a_3, a_4 ve a_5, (22)'de hala bilinmiyor. Bir manevranın bitiş zamanının T olduğunu varsayalım, aşağıdaki denklemler (22), (23) ve (24)'den elde edilebilir.

$$x(T) = a_0 +a_1T +a_2T^2 +a_3T^3 +a_4T^4 +a_5T^5 = x_e$$

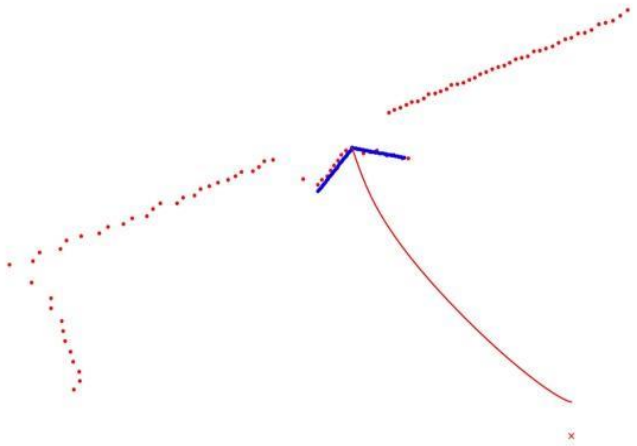
$$x'(T)=a_1 +2a_2T +3a_3T^2 +4a_4T^3 +5a_5T^4 =v_e$$

$$x''(T) = 2a_2 + 6a_3T + 12a_4T^2 + 20a_5T^3 = a_e$$

yukarıdaki denklemleri kullanarak aşağıdaki lineer sistemle a_3, a_4 ve a_5 hesaplanabilir

$$\begin{bmatrix} T^3 & T^4 & T^5 \\ 3T^2 & 4T^3 & 5T^4 \\ 6T & 12T^2 & 20T^3 \end{bmatrix} \begin{bmatrix} a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_e - x_s - v_sT - 0.5a_sT^2 \\ v_e - v_s - a_sT \\ a_e - a_s \end{bmatrix}$$

Tüm katsayılar bulunduğunda, Şekil 3.10'da gösterildiği gibi yol oluşturmak mümkündür. Son adım olarak, bu fonksiyon örneklenecek ve her bir iterasyon açısının düzeltilebilmesi için her nokta PID'ye hedef olarak verilecektir.



Şekil 3. 10. Mavi noktaların referans modelini temsil ettiği, kırmızı noktaların ve 'x'in ise robot üzerindeki sırasıyla LiDAR verilerini ve LiDAR konumunu gösterdiği hesaplanmış rota.

PID yolu izleme

Kontrol sistemlerinde çeşitli şekillerde kullanılan PID kontrolör, sistemi ayarlamak ve düzeltmek için geri bildirim veren bir kontrol döngüsü olarak çalışır. Bir önceki kısımdan izlenmesi gereken yol alındıktan sonra PID başlatılabilir. Başlangıç durumları ve yol boyunca noktalardaki sapma girdi olarak alınır ve algoritma, yolu takip ederken robotun sapma açısını düzeltir. Burada hata her adımda aşağıdaki gibi hesaplanır:

$$\varepsilon = arctan(\delta y, \delta x) - \theta \ ,$$

Burada ε, hedef açısı ile robot açısı arasındaki hatadır, δy ve δx, hedef ve mevcut durumlar arasındaki x ve y konumlarındaki farklardır ve θ mevcut açıdır. Robot, 0.08 m/s'lik lineer hızla sürülür ve açısal hız şu şekilde hesaplanır:

$$\omega = \kappa_P \, \varepsilon + \kappa_I \, \varepsilon + \kappa_D \, e.$$

Burada κ_P, κ_I ve κ_D, PID ayarından elde edilen PID sabitleridir, ε kümülatif hatadır ve *e* önceki hatadır.

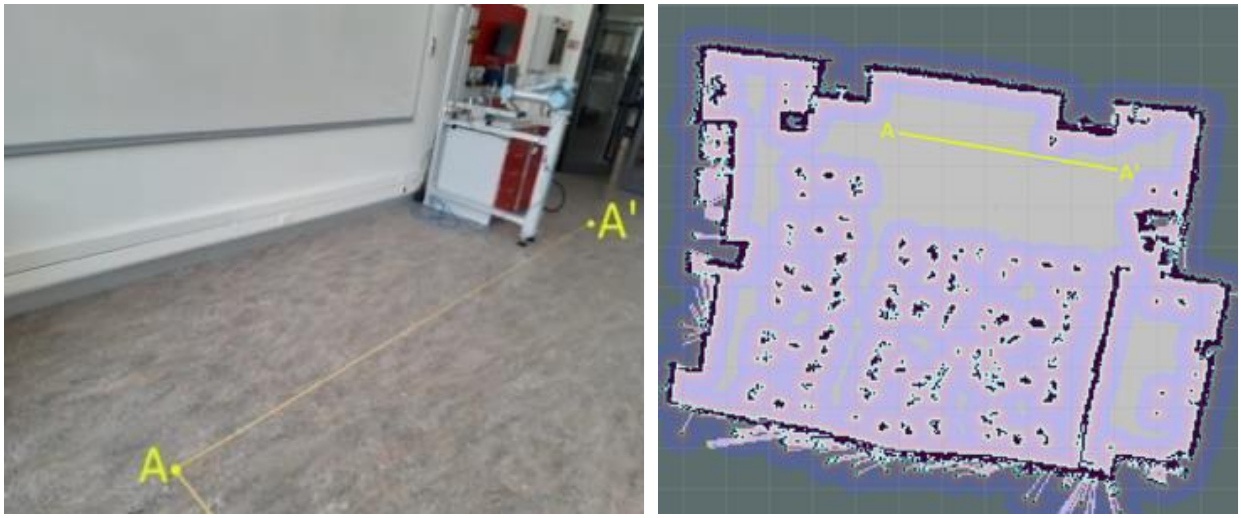
4. BULGULAR ve TARTIŞMA

Sensorlerden veriler toplanmakta, geliştirilen algoritmalar tarafından işlenmekte, ilgili arayüzler Ubuntu'ya kurulu ROS ve de python scriptleri ile çalışır durumdadır. Simulasyon ortamında testler gerçekleştirilmiştir. Simülasyon ve gerçek testlerde Şekil 3.4'de gösterilen akış izlenmektedir. Bitirdiğimiz iş paketleri ve de bazı testlerden elde ettiğimiz sonuçları yayınladığımız makale ve de bildirilerde ara çıktılar olarak sunduk.

Tüm sistemin akış şeması Şekil 3.4'de gösterilmektedir. Farklı görevler için kullanılabilmesi için tam simülasyon ve gerçek test ortamları yaratılmıştır. Robot başlar ve ardından simülasyon veya gerçek dünya uygulaması seçilir. Uygulama simülasyonda çalışıyorsa, Gazebo simülasyonu ve Pozyx simülasyonu başlatılır ve bu iki simülatör sentetik sensör verilerini ve harita verilerini sağlar. Uygulama gerçek dünyada çalışıyorsa gerçek sensör verileri elde edilir. Sentetik verileri veya gerçek veri başlatma paketini aldıktan sonra, otonom başlatmayı tam olarak filtrelemek için UWB uzaklık verilerini ve LiDAR tarama verilerini kullanır. Daha sonra amcl algoritması, robot yörüngesi sırasında robot konumunu hesaplamak için bu bilgileri, odometreyi ve Lidar tarama verilerini alır. Kalman Filtresi yerelleştirme düğümü, hareket modelini ayarlamak için odometre verilerini kullanır ve UWB aralık ölçümlerini ve amcl poz bilgilerini kullanarak poz bilgilerini günceller. Son olarak, GUI haritayı ve mevcut robot konumunu gösterir. Gerekirse, gezinti yığını (navigation stack) ya amcl poz bilgisi ile ya da daha iyi poz bilgisi sağladığı kanıtlanmış Kalman Filtresinden elde edilen poz bilgisi ile beslenebilir.

4.1. İlklendirme Testleri

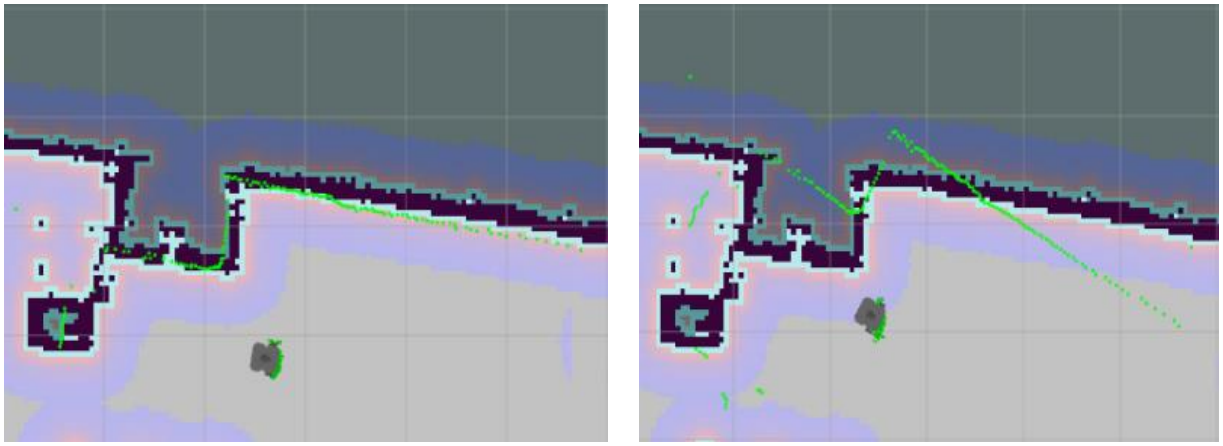
İlklendirme, ROS'un gezinti yığını ile uygun bir navigasyona sahip olmak için gereklidir. Robot nerede olduğunu ve hangi yöne doğru baktığını bilmiyorsa ulaştığı hedef oldukça hatalı olabilir. Robot, A noktasına yerleştirilir ve karşılaştırma için A 'hedef noktasına gönderilir (Şekil 4.1a). Şekil 4.1b, Gmapping algoritması ile daha önce LiDAR verilerinden elde edilen haritayı göstermektedir (Grisetti vd. 2005).



a) b)

Şekil 4. 1. a) Gerçek test ortamı. b) LiDAR ile elde edilen çevre haritası.

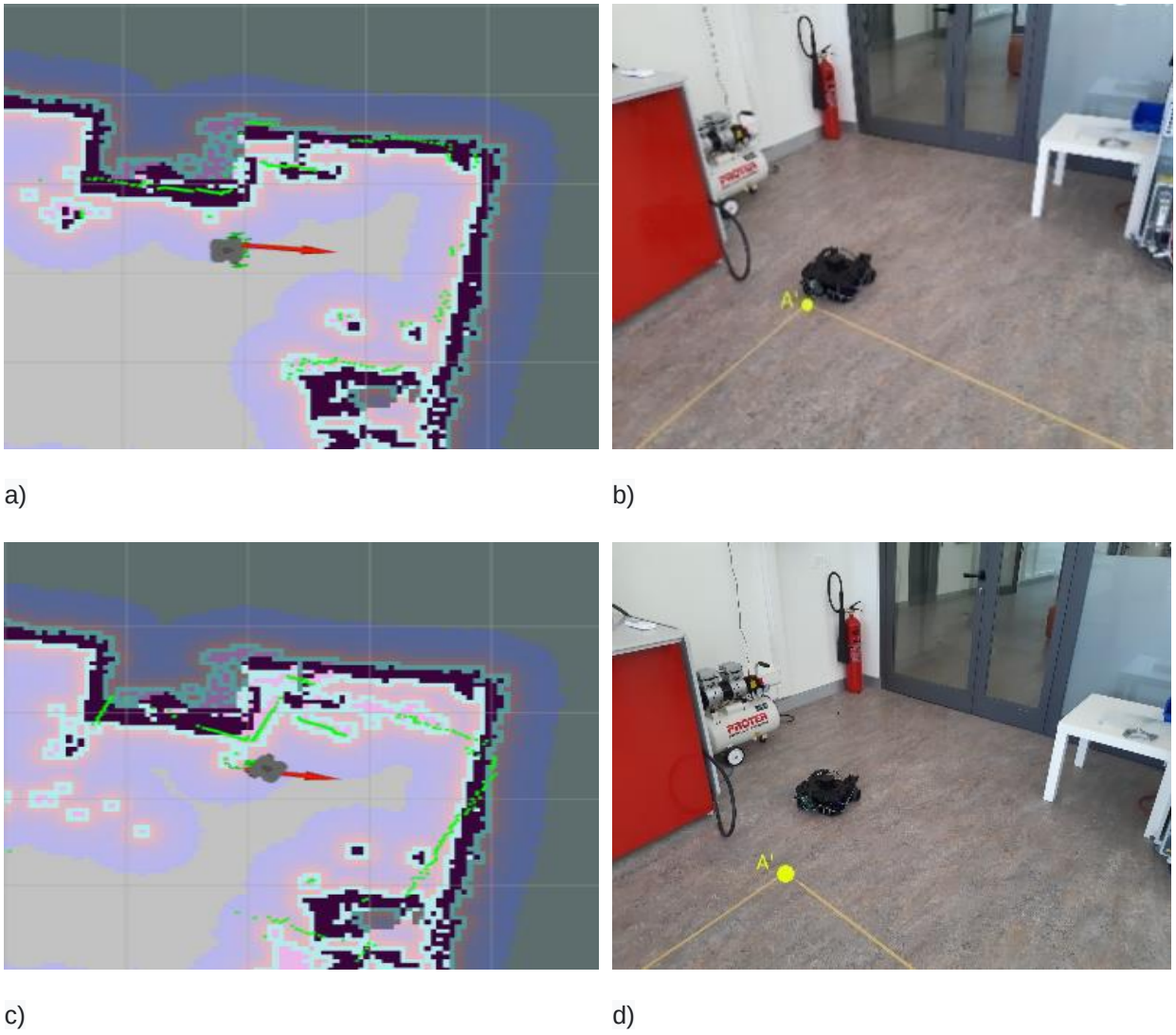
Önerilen algoritma, robot konumunu ve LiDAR'ı tahmin etmek için UWB uzaklık ölçümlerini kullanır ve baktığı yönü tespit etmek için verileri haritalandırır. Şekil 4.2a ve 4.2b, sırasıyla düzgün bir şekilde başlatılan konumu ve kötü bir şekilde başlatılan konumu gösterir. Buradaki yeşil noktalı çizgiler, önceden elde edilen haritanın üstündeki LiDAR tarama verilerini temsil eder. Görülebileceği gibi, Şekil 4.2a neredeyse mükemmel bir başlatmaya sahiptir, bu da robotun konumu ve gerçek dünyaya göre hizalanmasının harita bilgisi ile eşleştiği anlamına gelir. Aksine, Şekil 4.2b'deki sonuçlar, lazer verisinin, robotun kötü konum ve yön tahmini nedeniyle harita verileriyle kötü eşleştiğini göstermektedir.



a) b)

Şekil 4. 2. a) İyi başlatma. b) Kötü başlatma.

Robotun ilk pozu elde edildiğinde robot, ROS'un gezinme yığınındaki hareket temel algoritmasını kullanarak otonom olarak A'ya gider. Şekil 4.3a ve 4.3b, sırasıyla harita üzerinde ve gerçek ortamda iyi bir başlatma ile başlayan robotun son pozunu göstermektedir. Haritadaki LiDAR taramalarından da doğrulanabilen robot, hedefe mükemmel bir şekilde ulaştı. Kötü başlatmanın sonuçları Şekil 4.3c ve 4.3d'de gözlenebilir ve ilk pozda iyi bir tahmine sahip olmanın avantajı açıktır. Diğer bir deyişle, robot hedeften oldukça uzaktadır ve LiDAR verileri harita ile hiç eşleşmiyor. Yapılan testler sonucunda, robotun ilk yön tahmininde 10 derecenin üzerindeki hatanın, robotun tamamen yanlış bir son pozda konumlandırmasına neden olduğu görülmüştür. Bu nedenle, bu, ilk pozdaki hassas tahminin gerekliliğini vurgulamaktadır.



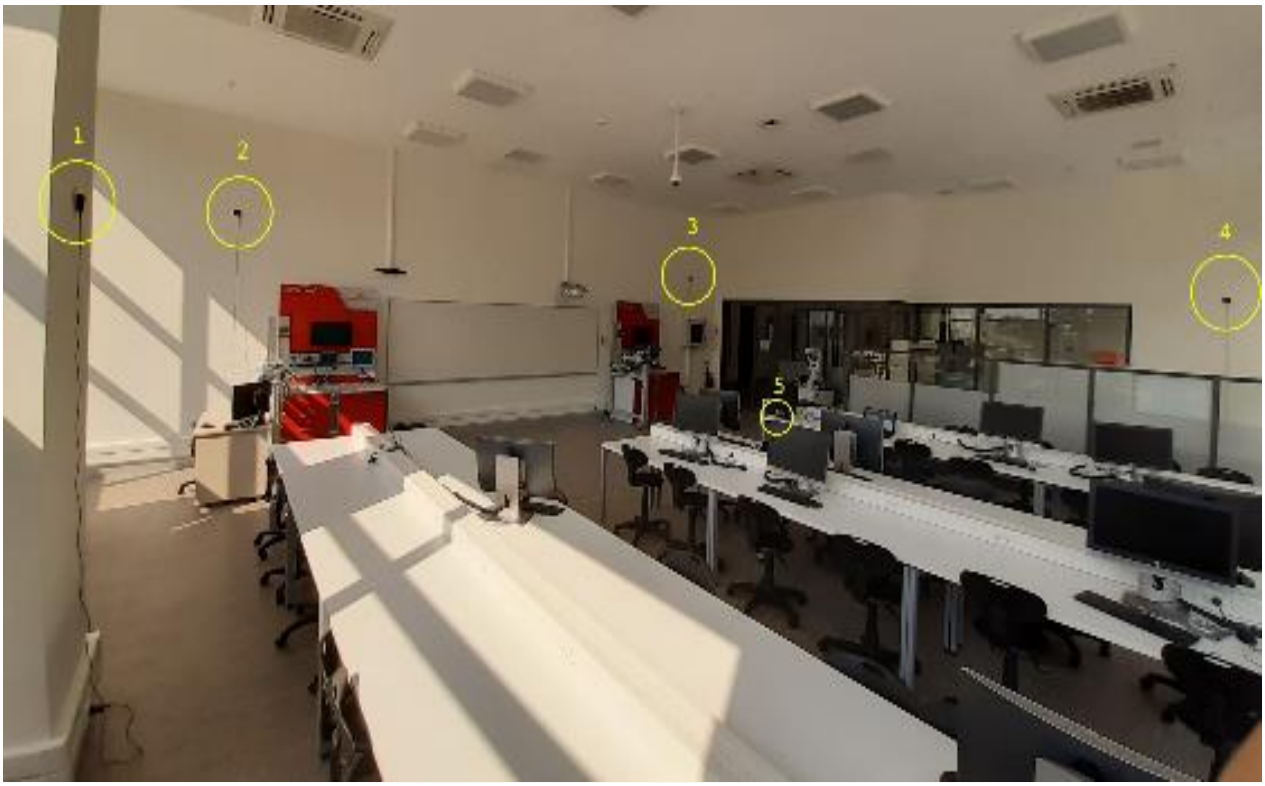
a) b)

c) d)

Şekil 4. 3. a) ve b) 'de iyi bir başlatma, c) ve b)' de kötü başlatma ile robotun son duruşu.

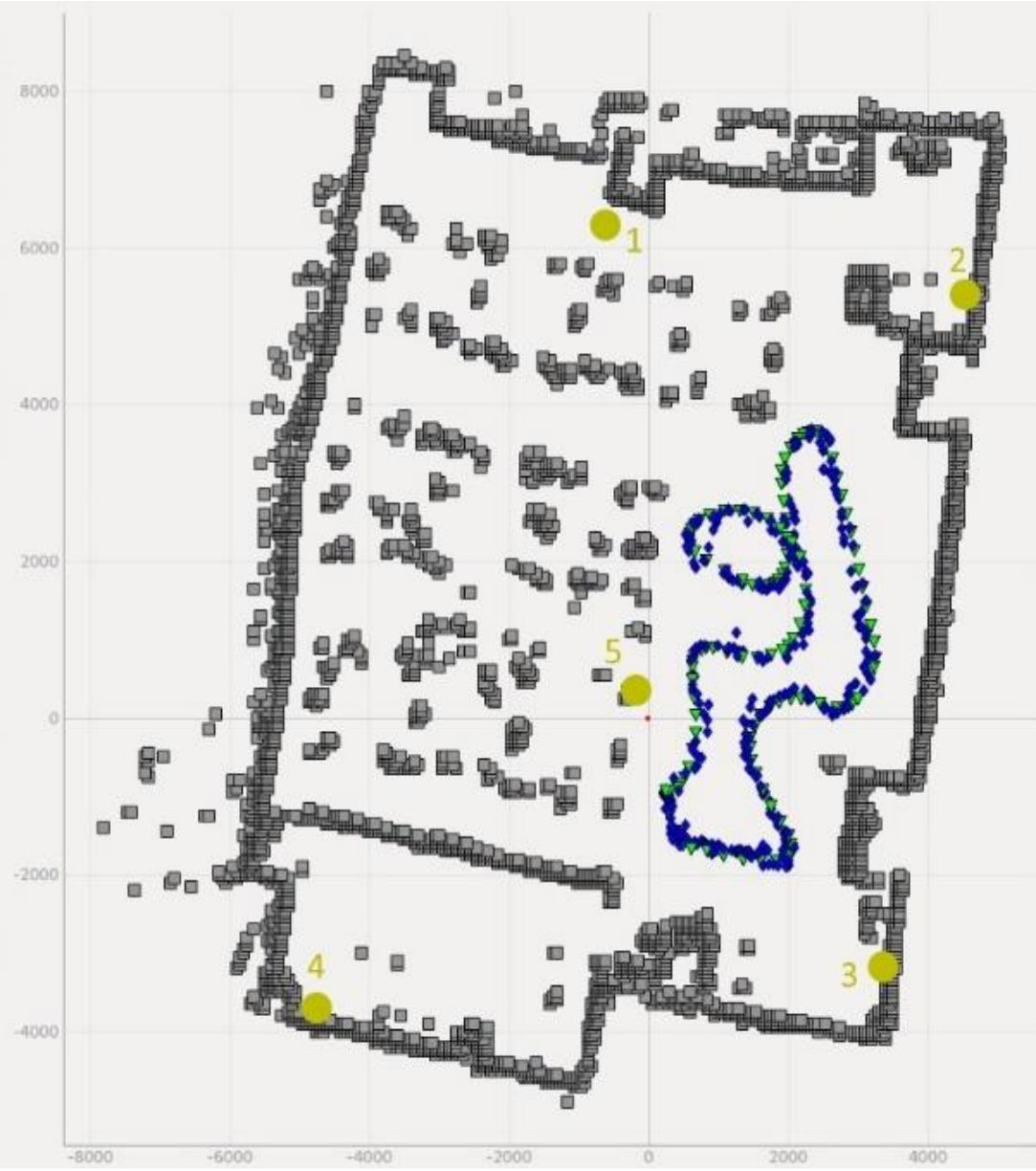
4.2. Konumlandırma Testleri

Hem başlatma hem de yerelleştirme testleri için, 10 m'ye 10 m'lik bir alanda sandalye ve masalarla dolu test ortamına beş UWB vericisi kurulur (Şekil 4.4). Robot üzerindeki UWB etiketi, uzaklık bilgisini almak için bu vericiler ile iletişim kurar. Eşzamanlı olarak, odometri ve amcl poz bilgilerini alır. Böylece, EKF algoritması daha iyi bir poz tahmini elde etmek için bu verileri birleştirir.



Şekil 4. 4. UWB vericileri ve gerçek test ortamları.

Sonuçlar, boyutların mm cinsinden verildiği Şekil 4.5'de gösterilmektedir. Sarı noktalar UWB çapalarını temsil ederken mavi noktalar ve yeşil üçgenler sırasıyla EKF ve AMCL tarafından tahmin edilen robot yörüngesini temsil eder. EKF, AMCL hesaplama karmaşıklığı nedeniyle konum bulmakta zorlandığı durumlarda bile konum bilgilerini sağlamaya devam eder. Ek olarak, çoklu sensörlü EKF'nin doğruluğu çoğu zaman amcl'den daha iyidir. Bu, sensör füzyonunun kullanımını motive eder, bir sensör ölçümü türü engellenmiş veya hatalı olsa bile diğer tip sistemi desteklemeye devam eder.



Şekil 4. 5. Robotun tahmini yörüngesi, gerçek testlerde sırasıyla EKF ve AMCL tarafından mavi ve yeşil olarak gösterilmiştir.

Robotik uygulamalarındaki lokalizasyon probleminin çözümü için UWB, amcl çıktısı ve odometri bilgilerini birleştiren EKF tabanlı lokalizasyon algoritması önerildi. Karesel aralık bilgisini ve tarama verilerini birleştiren en küçük kareler ve tarama eşleştirme tabanlı ilklendirme algoritmaları geliştirildi. ROS ekosistemi için UWB sensörlerini kullanmak için yeni modüller geliştirildi. Otonom robotik uygulamalar için eksiksiz bir sistem akışı oluşturulur. Simülasyon sonuçları ve Turtlebot3

ile gerçekleştirilen gerçek ortamdaki testler, UWB sensörlerinin hem yerelleştirme hem de navigasyon sistemlerinin doğruluğu, sağlamlığı ve pratikliği açısından başlatılması için faydalarını göstermektedir.

4.3. IMU-Encoder-Lidar Verileri birleştirme testleri

Robotların genel projelerinin çalıştırılması için ROS kullanılmaktadır. Robot konumlandırmasını daha iyi noktaya getirmek için farklı sensörler ile konumlandırma geliştirilmektedir. Enkoder (odometre) verisi, laser range finder (lidar) ve imu (inertial mesurement units) sensörleri ile bu konumlandırma sistemi desteklenmesi gerekmektedir. Konumlandırma sistemlerinde farklı sensörlerden gelen verileri fitrelemek aynı zamanda da sensörlerin kendi içlerindeki gürültüleri engellemek amacı ile bir nonlineer filtreleme yöntemi olan Extended Kalman tercih edilmiştir. Bu sistem içerisinde enkoder verisi üzerinde zamana bağlı olarak oluşan rastgele gürültüyü tek bir sensör ile uzun vadeli filtrelemenin bir yolu bulunmamaktadır. Bundan dolayı imu sensörü sayesinde elde edilen lineer ivmelenme, açısal ivmelenme ve gyro verilerin odometre verisi ile doğru bir şekilde birleştirilerek hataların azaltılması hedeflenmiştir. Bu işlemler için ROS ile hazırlanmış robot_localization paketi kullanılması uygun bulunmuştur. Sensörlerin seçimi ve kullanılması konusunda “pololu imu altimu 10v5” ve “mpu 6050” sensörleri test edilmiştir. IMU ile elde edilen ham verilerin yaw pitch roll verilerine dönüştürülmesi işlemi hem bir SCB (single computer board) hem de bir MCU (microcontroller unit) ile test edilmiştir. Yapılan testler esnasında “Altimu 10v5” modeli için geliştirilmiş olunan kütüphanelerin yeterli olmadığı gözlemlenmiştir. “MPU 6050” modeli ile gerçekleştirilen testlerin de veri transfer işlemleri için I2C protokolü kullanılmıştır. Ancak otomatik kalibrasyon işleminin yetersizliği ve işlemcide oluşabilecek yavaşlamalardan kaynaklanabilecek hatalar göz önünde bulundurularak bu işlem için bir MCU kullanılması daha uygun görülmüştür. Sonrasında ESP32 MCU için geliştirilmiş “i2cdevlib” kütüphanesi kullanılarak ROS sistemine ait olan rostcp protokolü kullanılmıştır. ROSTCP ile saniyede ortalama 80Hz bandında veri transferi yapılabilmektedir. Ancak IMU sensörlerinin doğası gereği yüksek frekansta veri transferine sahip olmaları gerektiği için (ortalama 250Hz) I2C veri transferi protokolü kullanılarak haberleşme işlemi tekrar sağlanmıştır. Böylece küçük hataların artması engellenmiş ve veri kayıplarının önüne geçilmiştir. C++ ile geliştirdiğimiz bir paket aracılığı ile I2C protokolünden elde edilen veriler bir publisher aracılığı ile ROS sistemi içerisine dahil edilmiştir. Yapılan testlerde ortalama 200Hz bandında veri akışı tespit edilmiştir ve veriler sensor.msgs/Imu.mgs tipinde kullanılmıştır. Elde edilen verilerin doğru bir biçimde odometre verisi ile birleştirilmesi için isterler doğrultusunda düzenlenmesi gerekmektedir.

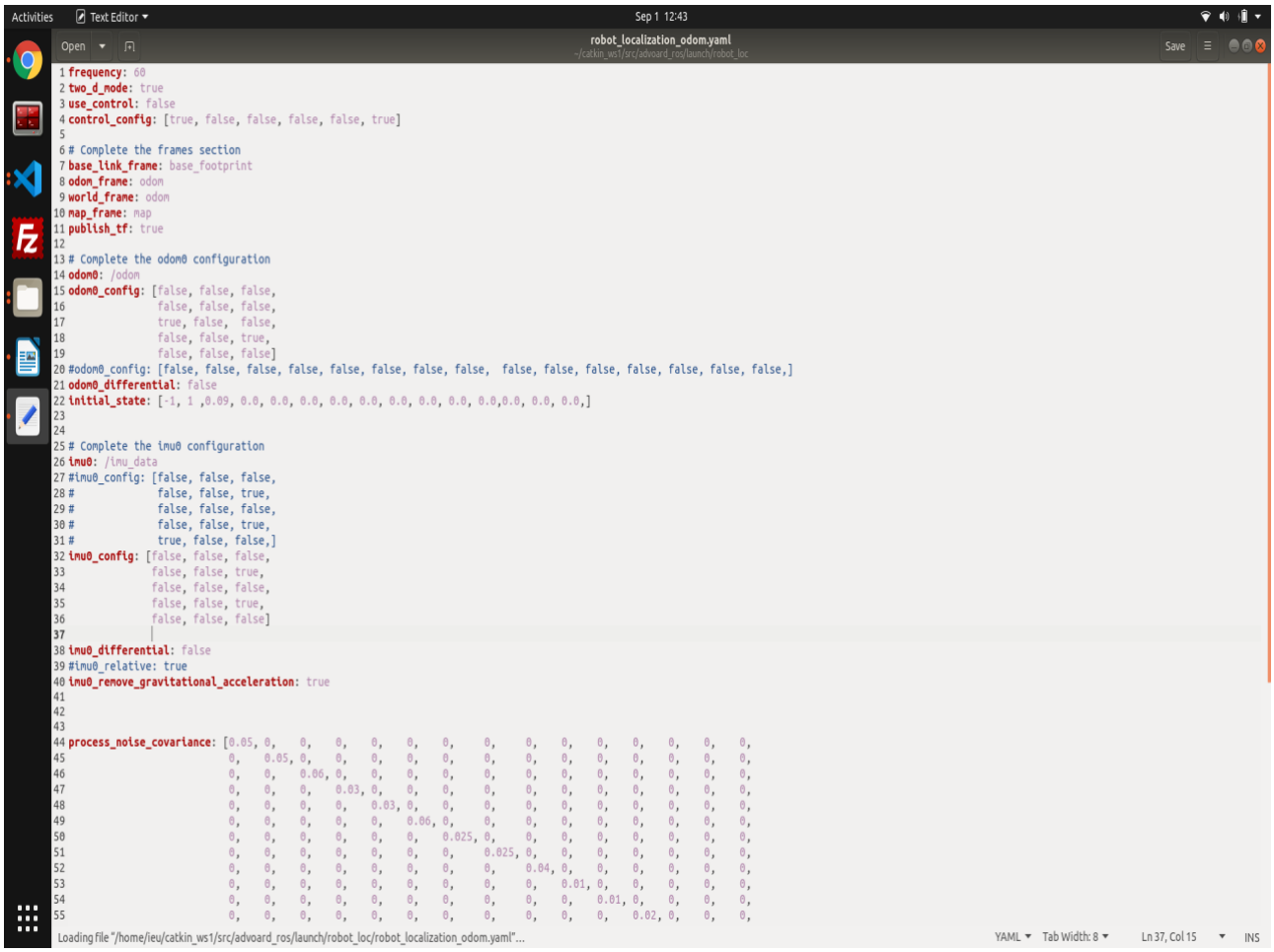
Yaptığımız testler sonucunda türetilmiş verilerin filtre içerisindeki etkisinin çok yüksek olduğu gözlemlenmiştir. Örneğin gyro verisi odometre ve imu sensörlerinden aynı zamanda filtre içerisinde yer aldığı durumda robot eksenlerinin ani değişikliklere neden olduğu gözlemlenmiştir. Kovaryans matrixindeki değerlerde yapılan değişiklikler sonucunda yapılan testlerde aynı sorunun devam ettiği gözlemlenmiştir. Devam eden aşamalarda yapılan testler sonucunda türetilmiş verilerden sadece bir değerin filtre içerisinde yer alması ancak açısal hız ve lineer ivme verilerinin robot üzerindeki etkisinin gösterilmesi hedeflenmiştir. Bu doğrultuda 2 farklı konfigurasyonda testler hazırlanmıştır.

Hazırlanan 1. konfigürasyonda robot yaw bilgsinin odometre üzerinden alınacağı ayrıca imu ve odometreden gelen açısal hız verisinin dahil edileceği ve lineer ivme verisinin yalnızca odometreden dahil edilerek test hazırlanmıştır. Tekerleklerin hareketli olmadığı durumda imu sensörün hareket ettirilerek robotun konumunun değiştiği gözlemlenmiştir. Ancak robotun uzun süreli hareketi sonucunda yaw açısındaki hatanın zamanla arttığı gözlemlenmiştir.
Hazırlanan 2. pakette robotun yaw açısının verisinin imu sensöründen gelen veriler ile sağlanması, açısal hız verisinin odometre ve imu ile sağlanması ve robotun lineer ivme verisinin sadece odometre'den gelen veriler ile hesaplanması sağlanmıştır. Test sonucunda robotun yaw açısının 15 dakikalık bir süreçte +-10 derece hassasiyet ile hesaplanması sağlanmıştır.

```
<rosparam param="twist0_config">[false, false, false,
                                 false, false, false,
                                 true,  false, false,
                                 false, false, false,
                                 false, false, false]</rosparam>
```

Note: The order of the boolean values are $(X, Y, Z, roll, pitch, yaw, \dot{X}, \dot{Y}, \dot{Z}, \dot{roll}, \dot{pitch}, \dot{yaw}, \ddot{X}, \ddot{Y}, \ddot{Z})$.

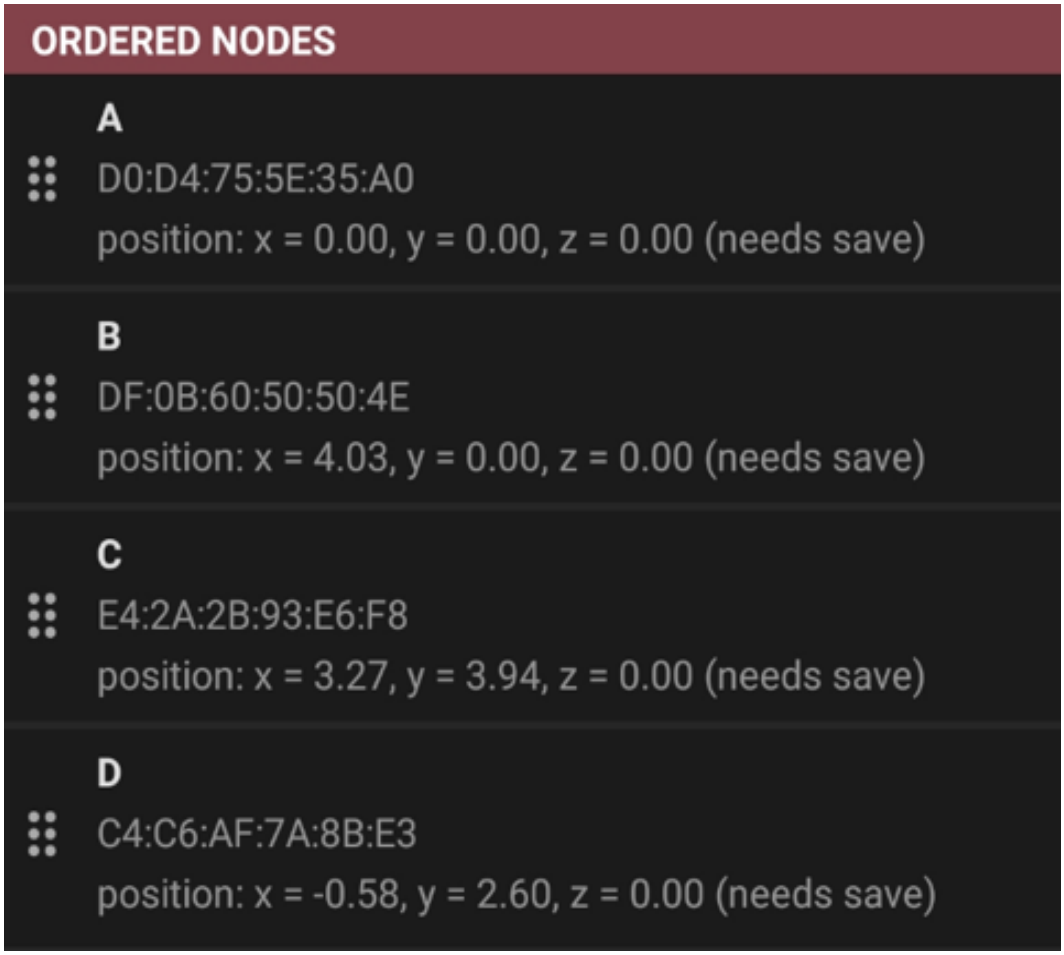
Bu hesaplamalara ek olarak robotun konumunun belirlenmesi için local konumlandırma hesaplamaları bir üst katman olan global konum katmanına taşınmaktadır. Bu katmanda ise AMCL (Adaptive Monte Carlo Localization) algoritmasından gelen veriler ile buradaki sonuçtan elde edilen veriler yine EKF aracılığı ile filtrelenmekte ve geri beslemesi yapılmaktadır. Şekil 4.6'da parametre seçimleri gösterilmiştir.



Şekil 4. 6. IMU-encoder entegrasyonunda parametre seçim

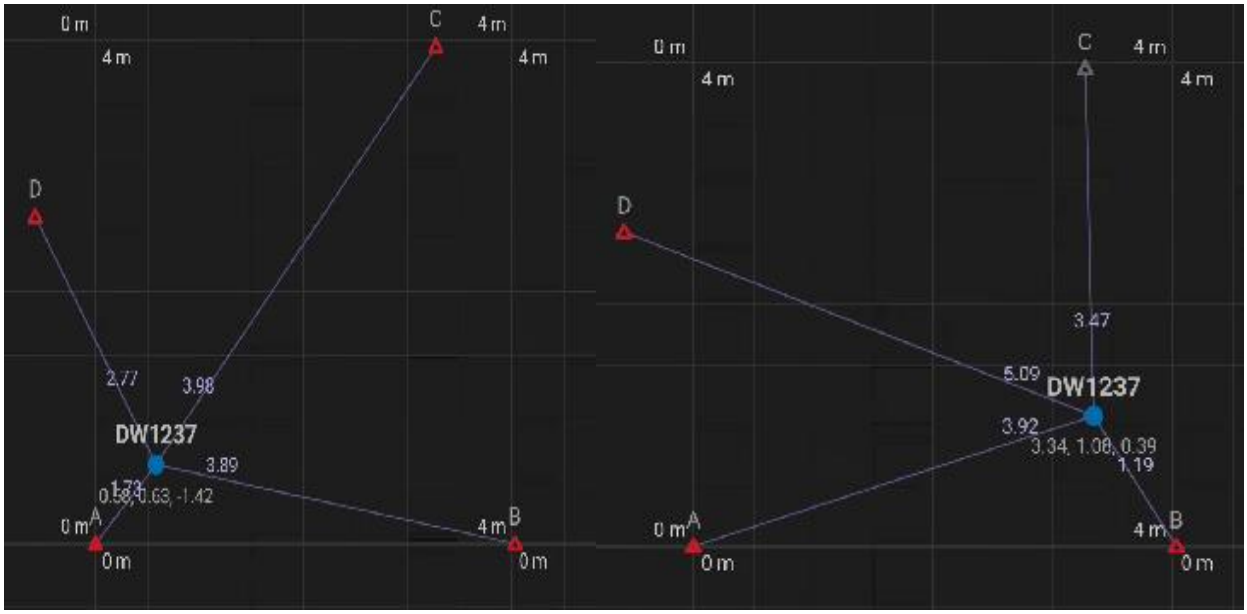
4.4. UWB ile ilgili Bazı Gerçek Ortam Testleri

Decawave MDEK1001 Development Kitinin Decawave web sitesinden Android Apk File ile telefon uygulaması kurulur. Sistemin çalışması için uygulama üzerinden bazı sensörleri Anchor (dayanak noktası), bazı sensörleri Tag (etiketli) olarak işaretlenir. Başlangıçta önerilen olarak verilen 4 Anchor ve 1 Tag işaretlenir. 4 Anchor'ı odanın köşelerine sabitlenip,1 Anchorı başlangıç noktası (0,0,0) olarak belirledikten sonra otomatik konumlama yapması için uygulama üzerinden komut verilir. Telefon kalan Anchorların konumlarını otomatik belirleyip x,y,z koordinatlarını çıkarır. Daha hassas ölçümler için tüm Anchorların konumları el ile ayarlanabilir.



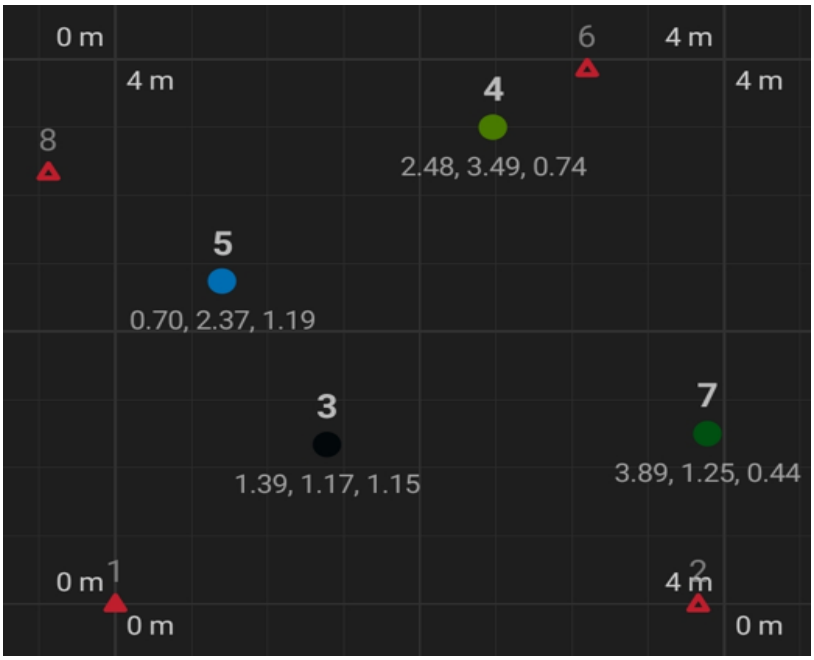
Şekil 4. 7. Otomatik konumlama ile Anchorların x,y,z konumlarını belirleme

Anchorların konumlarını belirledikten sonra 1 Tag'i hareket ettirip anlık konum değişimini uygulama ekranı üzerinden Şekil 4.8'teki gibi görüntülenebilir.



Şekil 4. 8. Tag'in ilk konumu ve hareket sonra konum değişimi

Daha farklı sayıda Anchor ve Tagler ile deneyler yapıldı. Toplam 12 tane sensörü istediğimiz Şekilde Tag veya Anchor olarak işaretleyip Şekil 4.9'da ki gibi kullanabiliriz.

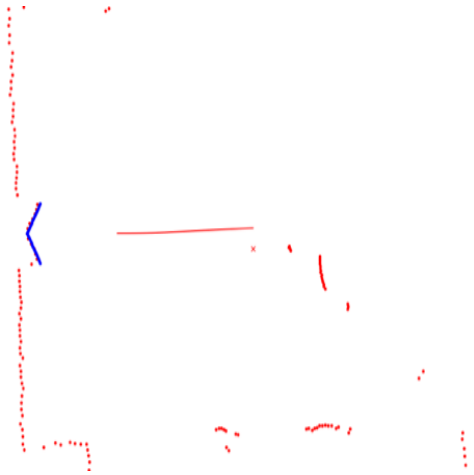


Şekil 4. 9. 4 Anchor ve 4 Tag ile uygulama

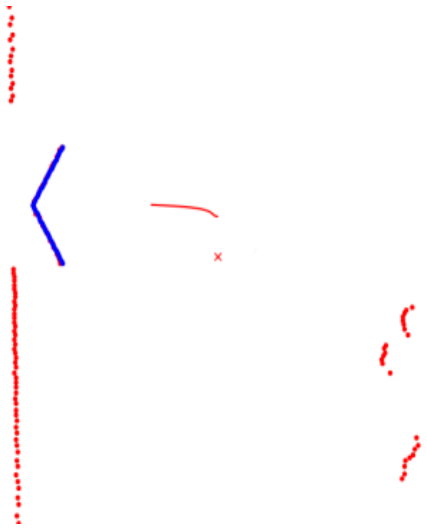
Dış ortamda çok uzak mesafelerde uygulamalar yapıldı. Bunun sonucunda sensörlerin 40 metreye kadar çalıştığı görüldü.

4.5. Şarj İstasyonuna Yanaşma Testleri

Bölüm 3.7'de geliştirilen algoritmalar ve sistem bu bölümde test edilmiştir. Orta hali olmayan tek kademeli yapı ile sistem istasyona bir yol oluşturabildi ve 8 santimetre hata ve ±3 derece farkla yanaştı. Oldukça doğru olmasına rağmen, konektörlerin arkasındaki yaylar kısa ve güçlüdür ve bu bir bağlantı sorununa neden olabilir. Bu nedenle, yerleştirme işlemi iki ana adıma bölünmüştür. Her adım, V-şekilli özelliğin tespitini, quintic yol oluşturmayı ve yol izlemeyi içerir. Direk istasyona gitmek yerine artık Şekil 4.10'da gösterildiği gibi istasyonun 1 metre önüne taşınır ve ilk adım Şekil 4.11'de gösterildiği gibi o yerde tekrarlanır. Yaklaşık 2 santimetre hata ve ±1 derece farkla doğru yerleştirme olduğu gözlemlendi.



Şekil 4. 10. Hedef noktası, referans noktasının 1 metre önüne ayarlanmıştır.



Şekil 4. 11. Model tekrar daha yüksek bir hassasiyet ve doğrulukla yürütülür, hedef noktası referans noktasının 0,5 metre önüne ayarlanır, çünkü kenetlenmiş durumda robotun merkezi ile referans noktası arasındaki mesafe 50 santimetredir.

5. SONUÇ

Son yıllarda insan kaynağının bulunmasının güç ve maliyetli olmasından ve de teknolojinin ilerlemesi ile beraber pek çok uygulama ve sektörde (otomotiv, lojistik, servis vb.) otonom sistemlere geçilmek istenmektedir. Bu uygulamalarda kullanılacak otonom cihazların ise görevlerini yerine getirebilmek için çevresini algılaması, yerini bilmesi, uzaktan görev ataması yapıldığında rotasını planlayabilmesi çözülmesi gereken önemli problemlerdir. Bu problemlere çözüm olarak bu projede iç alanlarda kullanılan otonom yer araçları için yer bulma, haritalandırma ve güzergâh planlama yapabilen cihaz tasarımı yapılmış ve geliştirilmiştir. Geliştirilen cihaz seçilen donanım ve geliştirilen yazılımlar sayesinde oldukça hassas (yaklaşık 5 cm konum hassasiyeti) ve fiyat açısından muadillerine göre uygundur. Ek olarak bu cihazın kullanılması ile hedef uygulamalarda verim artırımı sağlanacaktır. Bu proje ile ülkemize katma değeri yüksek ve ithalatı azaltabilecek bir cihaz kazandırılmıştır.

KAYNAKLAR

Arulampalam, M. S., Maskell S., Gordon, N. ve Clapp, T., 2002. “A tutorial on Particle Filters for online nonlinear/Non-Gaussian Bayesian tracking”, IEEE Transactions on Signal Processing, 50(2), 174-188.

Atia M. M., Liu S., Nematallah H., Karamat T. B. and Noureldin A., 2015. "Integrated Indoor Navigation System for Ground Vehicles with Automatic 3-D Alignment and Position Initialization", IEEE Transactions on Vehicular Technology, 64(4), pp. 1279-1292.

Bar-Shalom, Y., Li, X. R., ve Kirubarajan, T. 2001. Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software (1. Basım), New York, NY, USA: Wiley.

Bai, J., Lian, S., Liu, Z., Wang, K., ve Liu, D., 2018. “Virtual blind road following based wearable navigation device for blind people”, IEEE Transactions on Consumer Electronics, 64(1), 136-143.

A. Beck, P. Stoica, and J. Li., 2008 "Exact and approximate solutions of source localization problems", IEEE Transactions on Signal Processing, 56(5):1770-1778.

Besl P., McKay N.. "A method for registration of 3-D Shapes, 1992." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256.

Bregar K., ve Mohorcic, M., 2018. “Improving indoor localization using convolutional neural networks on computationally restricted devices”, IEEE Access, 6, 17429-17441.

Blanco, J., Gonzalez J., ve Madrigal J., 2008. “A pure probabilistic approach to range only SLAM”, ICRA, 1436-1441.

Canedo-Rodriguez, A., Alvarez-Santos, V., Regueiro, C.V., Iglesias, R., Barro, S., Presedo, J., 2016. “Particle filter robot localisation through robust fusion of laser, WiFi, compass, and a network of external cameras”, Information Fusion, 27, 170-188.

Deibler, T. ve Thielecke J., 2010. “UWB SLAM with Rao Blackwellized Monte Carlo data association”, IPIN.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999, May). Monte carlo localization for mobile robots. In Proceedings of the 1999 IEEE International conference on robotics and automation (ICRA), 2, 1322-1328.

Dobrev, Y., Gulden, P., ve Vossiek, M., 2018. “An indoor positioning system based on wireless range and angle measurements assisted by multi-modal sensor fusion for service robot applications”, IEEE Access, 6, 69036-69052.

Dudek, D. ve Jenkin, M., 2010. Computational Principles of Mobile Robotics, USA: Cambridge University Press.

Firebase. Dökümanlar.
https://firebase.google.com/docs
Son erişim tarihi: 20.06.2019

Fritsche, P., ve Wagner, B., 2017. “Modeling structure and aerosol concentration with fused radar and lidar data in environments with changing visibity”, IROS, 2685-2690.

Fox D., Burgard W. and Thrun S., 1997. "The dynamic window approach to collision avoidance," in *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23-33.

Gamallo, C., Mucientes, M., Regueiro, C.V., 2015. ”Omnidirectional visual slam under severe occlusions”, Robotics and Autonomous Systems, 65, 76-87.

Grisetti, G., Stachniss, C. ve Burgard, W., 2005. “Improving grid-based SLAM with Rao Blackwellized particle filters by adaptive proposals and selective resampling”, ICRA, 2432-2437.

Grisetti, G., Kummerle, R., Stachniss, C., ve Burgard, W., 2010. “A tutorial on graph based SLAM”, IEEE Intelligent Transportation Systems Magazine, 2(4), 31-43.
Golub, G. and Van Loan, C., 1996. Matrix Computations. Johns Hopkins University Press.

Guruj, A. K., Agarwal, H., ve Parsediya, D. K., 2016. “Time-efficient A* algorithm for robot path planning”, Procedia Technology, 23, 144-149.

Hahnel, D., Burgard, W., Fox D., Fishkin, K. ve Philipose, M., 2004. “Mapping and localization with RFID technology”, ICRA, 1015-1020.

Hasegawa, Y., ve Fujimoto, Y., 2016. “Experimental verification of path planning with SLAM”, IEEE Journal of Industry Applications, 5(3), 253-260.

He, S. ve Chan, S. H. G., 2016. “Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparions”, IEEE Communications Surveys and Tutorials, 18(1), 466-490.

Huang L., Pan S., Wang S., Zeng P., and Ye F., 2018. "A fast initialization method of Visual-Inertial Odometry based on monocular camera", 2018 Ubiquitous Positioning, Indoor Navigation and Location-Based Services (UPINLBS).

Hwang, S. ve Song, J., 2011. “Monocular vision-based SLAM in indoor environment using corner, lamp, and door features from upward-looking camera”, IEEE Transactions on Industrial Electronics, 58(10), 4804-4812.

Ionicframework. Dökümanlar.
https://ionicframework.com/docs
Son erişim tarihi: 20.06.2019

Khaleghi, B., Khamis, A., Karray, F.O., Razavi, S.N., 2013. “Multisensor data fusion: a review of the state-of-the-art”, Information Fusion, 14(1), 28-44.

Kohlbrecher, S., Stryk, V. O., Meyer, J., ve Klingauf U., 2011. “A flexible and scalable SLAM system with full 3D motion estimation”, IEEE International Symposium on Safety, Security and Rescue Robotics, 155-160.

Kummerle, R., Grisetti, G. ve Burgard, W., 2012. “Simultaneous parameter calibration, localization, and mapping”, Advanced Robotics, 26(17), 2021-2041.

Le, A., Prabakaran, V., Sivanantham, V., ve Mohan, R., 2018. “Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor”, Sensors, 18(8), 2585.

Leung, C., Huang, S., ve Dissanayake, G., 2006. “Active SLAM using model predictive control and attractor based exploration”, IEEE/RSJ International Conference on Intelligent Robots and Systems, 5026-5031.

Li, X., Wu, W., Guo, J., ve Song, L., 2013. “The method and development trend of laser ranging”, 5th International Conference on Intelligent Human-Machine Systems and Cybernetics.

Liu, Y., Sun, F., Tao, T., Yuan, J., ve Li, C., 2007. “A solution to active simultaneous localization and mapping problem based on optimal control”, IEEE International Conference on Mechatronics and Automation, 314-319.

Liu, H., Darabi, H., Banerjee, P., ve Liu, J., 2007. “Survey of wireless indoor positioning techniques and systems”, IEEE Transaction on Systems, Man. And Cybernetics-Part C: Appl. Rev., 37(6), 1067-1080.

Maurović, I., Seder, M., Lenac, K., ve Petrović, I., 2017. “Path planning for active SLAM based on the D* algorithm with negative edge weights”, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48(8), 1321-1331.

Mota, F., Rocha, M., Rodrigues, J., Albuquerque, V., ve Alexandria, A., 2018. “Localization and navigation for autonomous mobile robots using petri nets in indoor environments”, IEEE Access, 6, 31665-31675.

Mur-Artal, R., Montiel, J. M. M., ve Tardos, J. D., 2015. “ORB-SLAM: A versatile and accurate monocular SLAM system”, IEEE Transactions on Robotics, 31(5), 1147-1163.

Murtra, A.C., 2011. “Map–based localization for urban service mobile robotics”, Ph.D. Thesis, Universitat Politecnica de Catalunya.

Olson, E. B., 2009. “Real-time correlative scan matching”, IEEE International Conference on Robotics and Automation.

Oguz-Ekim P., 2020a. “Localization with Particle Filter Based on Odometer and UWB”, HORA’20, June, Ankara/Turkey.

Oguz-Ekim P., 2020b “TDOA based localization and its application to the initialization of LiDAR based autonomous robots,” Robotics and Autonomous Systems, vol. 131.

Oguz-Ekim P., Bostancı B., Tekkok S., Soyunmez E., 2020c. “The EKF based Localization and Initialization Algorithms with UWB and Odometry for Indoor Applications and ROS Ecosystem”, SIU’20, October, Gaziantep/Turkey.

Oguz-Ekim P. 2020d, “Localization and Initialization Algorithms based on UWB, LiDAR and Odometry for Robotic Applications with ROS Ecosystem,” European Journal of Science and Technology, vol. 20, pp. 343-350.

Oguz-Ekim P., Bostancı B., Tekkok S., Soyunmez E., 2021. “A novel docking algorithm based on the LIDAR and the V-shape”, European Journal of Science and Technology, vol 26, pp. 35-40.

Oguz-Ekim, P., Bostanci, B., Tekkok, S., Soyunmez, E., https://github.com/ieuagv>

Özgören M.K., 2020. Kinematics of General Spatial Mechanical Systems. Wiley, 2020.

Patle, B. K., 2016. “Intelligent navigational strategies for multiple wheeled mobile robots using artificial hybrid methodologies”, Doctoral Dissertation.

Patle, B. K., Pandey, A., Parhi, D. R. K., ve Jagadeesh, A., 2019. “A review: on path planning strategies for navigation of mobile robot”, Defence Technology.

Peng, J., Huang, Y., ve Luo, G., 2015. “Robot path planning based on improved A* algorithm”, Cybernetics and Information Technologies, 15(2), 171-180.

Pozyx. Ürün sayfası.
https://www.pozyx.io/shop/product/creator-kit-lite-67
Son erişim tarihi: 19.06.2019

RPLidar. Bilgi formu.
https://www.robotshop.com/media/files/pdf2/ld108_slamtec_rplidar_datasheet_a1m8_v1.1_en_2_.pdf
Son erişim tarihi: 19.06.2019

Robot Operating System. http://www.ros.org

Rowekamper, J., Sprunk, C., Tipaldi, G. D., Stachniss, C., Pfaff, P. ve Burgard, W., 2012. “On the position accuracy of mobile robot localization based on Particle Filters combined with scan matching”, IEEE RSJ International Conference on Intelligent Robots and Systems, 3158-3164.

Sahinoglu, Z., Gezici, S., ve Güvenc, İ., 2008. “Ultra-Wideband Positioning Systems: Theoretical Limits, Ranging, Algorithms, and Protocols”, Cambridge, U.K.: Cambridge University Press.

Siciliano B. ve Khatib O., 2016. Chapter 47, Springer Handbook of Robotics, Springer International Publishing.

Schweinzer, H. ve Kaniak, G. 2010. "Ultrasonic device localization and its potential for wireless sensor network security", Control Engineering Practice, 18(8), 852–862.

Siegwart, R. ve Nourbakhsh I.R., 2004. Autonomous Mobile Robots, Cambridge, MA: MIT Press.

Song, Y., Guan, M., Tay, W.P., Law, C.L., ve Wen, C., 2018. “UWB/Lidar Fusion for Cooperative Range Only SLAM”, arXiv:1811.02854v1 [cs.RO] 7 Nov 2018.

Sprunk, C., Tipaldi G., Cherubini, A., ve Burgard, W., 2013. “Lidar based teach-and-repeat of mobile robot trajectories”, IROS.

Takahashi A., Hongo T., Ninomiya Y., and Sugimoto G., 1989. Local Path Planning And Motion Control For Agv In Positioning. IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '89), The Autonomous Mobile Robots and Its Applications, Tsukuba, Japan, 1989, pp. 392- 397.

Tatlı N. E., Fidan D., Kalaycı B., Çeber C. and Oğuz Ekim P., 2021. "The Application of Leader Following Method and Cubic Polynomial Path Planning Algorithm with Formation Control on Multi-Robot Systems", European Journal Of Science And Technology , No. 31, pp. 38-46.

Tardos, J. D., Neira, P.M. ve Leonard J. J., 2002. "Robust mapping and localization in indoor environments using sonar data", International Journal of Robotics Research, 21(4), 311-330.

Tipaldi, G. D., Dellius, D. M. ve Burgard, W., 2013. "Lifelong localization in changing environment", International Journal of Robotics Research, 1662-1678.

Thrun S., Burgard, W., ve Dieter F., 2005. Probabilistic Robotics, Cambridge, MA: MIT Press.

Wachaja, A., Agarwal, P., Zink, M., Adame, M. R., Möller, K. ve Burgard, W., 2017. "Navigating blind people with walking impairments using a smart walker", Autonomous Robots, 41(3), 555 – 573.

Wang, C., Zhang, H., Nguyen, T. ve Xie, L., 2017. "Ultra wideband aided fast localization and mapping", IROS, 1602-1609.

Wolf, J., Burgard, W., ve Burkhardt, H., 2005. "Robust vision based localization by combining an image retrieval system with Monte Carlo localization", IEEE Transactions on Robotics, 21(2), 208-216.

Weiteng, Z., Baoming, H., Dewei, L., ve Bin, Z., 2013. "Improved reversely a star path search algorithm based on the comparison in valuation of shared neighbour nodes", ICICIP, 161-164.

Yılmaz Z. Ve Bayındır L. 2019. "Simulation of lidar-based robot detection task using ros and gazebo", European Journal Of Science And Technology, pp:375-388.

Zhang, G., Lee, J. H., Lim, J. Ve Suh, H., 2015. "Building a 3-D line-Based Map Using Stereo SLAM", IEEE Transactions on Robotics, 31(6), 1364-1377.

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Dr. Öğr. Üyesi PINAR OĞUZ EKİM
Proje No:	119E376
Proje Başlığı:	Kapalı Alanlarda Kullanılan Otonom Yer Araçları Için Yer Bulma, Haritalandırma Ve Güzergâh Planlama Yapabilen Cihaz Tasarımı Ve Geliştirilmesi
Proje Türü:	3501 - Kariyer
Proje Süresi:	24
Araştırmacılar:	
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	İZMİR EKONOMİ Ü.
Projenin Başlangıç ve Bitiş Tarihleri:	15/03/2020 - 15/06/2022
Onaylanan Bütçe:	287236.68
Harcanan Bütçe:	236531.57
Öz:	Otonom sistemler son yıllarda insan için tehlikeli uygulamalarda veya insanla iş birliği içinde oldukları alanlarda oldukça önem kazanmıştır. Bu uygulamaların pek çoğunda otonom sistemin bulunduğu yeri tespit etmesi, çevresinin haritasını çıkartması ve de bu bilgilere dayanarak görevine uygun olan güzergâhı planlaması önemli alt problemler olarak karşımıza çıkmaktadır. Farklı sensörlerden gelen bilgilerin güvenilir ve verimli bir şekilde birleştirilmesinin, konum bulma ve daha birçok robotik sorununun çözümüne yardımcı olmaktadır. Çünkü bir sensör belirli çevresel koşullar altında ölçüm alamadığında diğeri hata yayılımını azaltmak için kullanılabilir. Buna ek olarak, bir sensörün hata özellikleri, farklı özelliklere sahip başka bir sensörün kullanılması ile düzeltilebilir. Sensör füzyonunun önemi yaygın olarak bilinmesine rağmen, yeni geliştirilen sensör teknolojileri ve uyumlu algoritmalar henüz kapsamlı bir şekilde çalışılmamıştır. Bu nedenle projemizde özellikle Odometre, UWB ve de Lidar sensörlerinden gelen bilgiler harmanlanmıştır. Algoritmalar Python ve C dilleri ile geliştirilmiş ve ROS aracılığı ile gerçek zamanlı olarak donanımlardan gelen bilgileri işleyebilmişler. 5 cm-10 cm konumlandırma hassasiyetinde oldukça gürbüz bir sistem ortaya çıkarılmıştır.
Abstract:	In recent years, autonomous systems have gained importance in applications that are dangerous for humans or in areas where they are in cooperation with humans. In many of these applications, determining the location of the autonomous system, mapping its environment, and planning the appropriate route based on this information appear as important sub-problems. Combining information from different sensors reliably and efficiently helps to solve many robotics problems such as positioning and more. Because when one sensor fails to measure under certain environmental conditions, the other can be used to reduce error propagation. In addition, the error characteristics of a sensor can be corrected by using another sensor with different characteristics. Although the importance of sensor fusion is widely known, newly developed sensor technologies and compatible algorithms have not yet been extensively studied. For this reason, information from Odometer, UWB and Lidar sensors has been blended in this project. The algorithms which were developed with Python and C languages, were able to process the information coming from the hardware in real time via ROS. Hence a very robust system with a position accuracy of 5 cm-10 cm has been achieved.
Anahtar Kelimeler:	Yer bulma, haritalandırma, güzergâh planlama, algılayıcı harmanlama
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Evet

Projeden Yapılan Yayınlar:	1- The Application of Leader Following Method and Cubic Polynomial Path Planning Algorithm with Formation Control on Multi-Robot Systems (Makale - null), 2- Localization and Initialization Algorithms based on UWB, LiDAR and Odometry for Robotic Applications with ROS Ecosystem (Makale - null), 3- A Novel Docking Algorithm Based On The LiDAR And The V-shape Features (Makale - null), 4- TDOA based localization and its application to the initialization of LiDAR based autonomous robots (Makale - null), 5- Localization with Particle Filter Based on Odometer and UWB (Bildiri - Sözlü Sunum), 6- The EKF based Localization and Initialization Algorithms with UWB and Odometry for Indoor Applications and ROS Ecosystem (Bildiri - Sözlü Sunum), 7- Izmir'de üretilecek robotlar daha akıllı olacak (Yayılım - Medyada Haber), 8- geleceğe yatırım otonom taşıma robotu (Yayılım - Medyada Haber), 9- Izmir'in akıllı robotuna ödül (Yayılım - Medyada Haber), 10- Otonom Robotlar Zirvesi (Yayılım - Düzenlenen Kolokyum),