Otonom Mobil Robotlar İçin Gazebo Test Otomasyonu Aracı Geliştirilmesi

Development of Gazebo Test Automation Tool for Autonomous Mobile Robots

Emre Can Contarlı[1,2], *Volkan Sezer*[1,2]

[1]Kontrol ve Otomasyon Mühendisliği Bölümü
İstanbul Teknik Üniversitesi, İstanbul
contarli@itu.edu.tr, sezerv@itu.edu.tr

[2]Akıllı ve Otonom Sistemler Laboratuvarı (Smart and Autonomous Systems Laboratory, SASLab)
İstanbul Teknik Üniversitesi, İstanbul
contarli@itu.edu.tr, sezerv@itu.edu.tr

Özetçe

Bu çalışmada, otonom mobil robotların özellikle lokal planlayıcılarının performanslarını ölçmeye yönelik bir simülasyon test otomasyonu aracı geliştirilmiştir. Araç, sırasıyla gerçek dünyayı birebir taklit edebilen bir fizik motoruna sahip olan 3 boyutlu Gazebo simülasyon ortamında ortam haritasında bulunmayan beklenmedik engellerin istenen büyüklük ölçülerinde ve rastgele konumlarda oluşturulmasını, sonrasında bunun istenen sayıda tekrarlanıp Monte Carlo simülasyonları için lazım olacak miktarda yeni ortamlar oluşturulmasını sağlamaktadır. Son olarak oluşturulan fazla sayıdaki bu ortamların her birinde aracın bir başlangıç noktasından bitiş noktasına gidene kadar engelleri aşıp aşamayacağını test etmek için her ortamda simülasyonu çalıştırıp, başarı metriklerini bir kayıt dosyasına işlemektedir. Aynı simülasyonlar farklı lokal planlayıcı da kullanılarak tekrarlandığında performans metriklerinin kıyaslamasını yapmak için gerekli veriler elde edilmiş olmaktadır. Test otomasyonu aracının kullanılması sonucunda geliştirmekte olduğumuz lokal planlayıcı algoritmasının eski versiyonuna göre artıları ve eksileri tespit edilmiş olup, geliştirmelerimize ışık tutulmuştur.

Abstract

In this study, we have developed a simulation test automation tool aimed at evaluating the performance of autonomous mobile robots, particularly their local planners. The tool facilitates the generation of environments with unexpected obstacles of random positions and dimensions, not present in the environment map, within a 3D Gazebo simulation environment equipped with a physics engine that faithfully emulates the real world. Subsequently, these environments are replicated as necessary to create a sufficient number of new scenarios for Monte Carlo simulations. To assess the robot's ability to navigate from a starting point to a destination while overcoming obstacles in numerous environments, each scenario is executed, and the corresponding success metrics are logged. Furthermore, by conducting the same simulations using different local planners, the obtained data enables a comparative analysis of performance metrics. Overall, our developed simulation test automation tool provides a comprehensive framework for assessing the efficiency and robustness of various local planners employed in autonomous mobile robots, offering valuable insights into their real-world applicability and performance optimization. The use of the test automation tool has enabled us to identify the pros and cons of our local planner algorithm under development in comparison to its previous version, hence it is shedding light on our improvements.

1. Giriş

Çeşitli mühendislik alanlarında uygulamaların performansını değerlendirmek için farklı senaryolara maruz bırakmak ve başarılı olup olmadıklarını gözlemlemek önemlidir. Ancak, bu senaryo testlerini gerçek fiziksel sistemlerde gerçekleştirmek genellikle yüksek maliyetli, zaman alıcı ve tehlikeli olabilmektedir. Bu nedenle, simülasyon ortamlarının kullanımı ve bu ortamlarda hızlı test otomasyonu oluşturmak, önemli bir gereklilik haline gelmiştir [1].

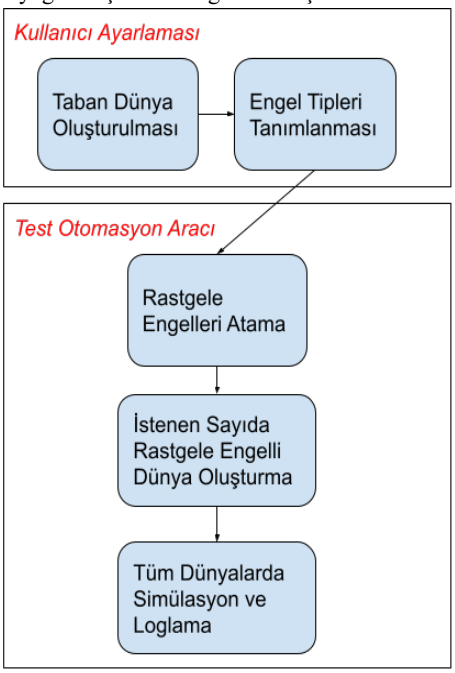
Robotik ve otonom araç teknolojileri üzerinde çalışan mühendisler, geliştirmelerini test etmek için gerçekçi, tekrarlanabilir ve kolay senaryo üretimi sağlayan bir simülasyon aracı kullanmanın çalışmalarını hızlandırdığını belirtmektedir [2]. Bu bağlamda, Gazebo simülasyon ortamı, gerçek dünyaya yakın bir fizik motoru kullanması, simülasyon ortamındaki çalışmaların gerçek sistemlere kolaylıkla entegre edilebilmesi, birçok sensörün kullanılabilmesi ve arayüz kolaylığı gibi nedenlerle ROS (Robot Operating System) yapısıyla birlikte otonom robot geliştirmelerinde sıklıkla tercih edilmektedir [3,4]. Gazebo ve ROS kullanılarak otonom mobil robotlar alanında SLAM (Simultaneous Localization and Mapping), navigasyon gibi uygulamaların simülasyonu yaygın olarak yapılmaktadır. Yeni algoritmaların başarısını ölçmek için Gazebo'da "dünya" olarak adlandırılan 3 boyutlu birçok yeni ortamın oluşturulması gerekebilmektedir. Bu amaçla, literatürde 2 boyutlu haritalardan 3 boyutlu Gazebo dünyaları oluşturmak için çeşitli çalışmalara rastlanmaktadır [5-7]. Bu

çalışmalarda, gri ölçekli görüntülerden yükseklik bilgisi eklenerek 3 boyutlu Gazebo dünyaları oluşturulduğu gözlenmektedir. Bu yaklaşımla elde edilen dünyalar, global planlayıcıların hızlı bir şekilde test edilmesini sağlasa da lokal planlayıcıların test edilmesi için, haritası bilinen ortamların 3 boyutlu Gazebo modelinin oluşturulmasından ziyade, bilinen haritalarda olmayan, beklenmedik engellerin modele rastgelelik tabanlı bir yöntemle eklenmesi ve birçok farklı dünya oluşturulması gerekmektedir.

Bu çalışmada, lokal planlayıcıların seri bir şekilde test edilmesini mümkün kılmak için Gazebo ortamında rastgele engeller atayarak istenen sayıda yeni dünya oluşturulması sağlanmıştır. Sonrasında bu dünyalarda simülasyonlar otomatik bir şekilde sırasıyla gerçekleştirilmiş ve istenen performans metrikleri kaydedilmiştir. Bölüm 2'de yöntemin adım adım çalışma prensibi açıklanmış, bölüm 3'te bu yöntemin kullanıldığı ve sağladığı kolaylıkların gösterildiği bir örnek çalışma anlatılmış, bölüm 4'te ise sonuçlar yorumlanmış ve gelecekte yapılabilecek geliştirmelerden bahsedilmiştir.

2. Test Otomasyonu Yöntemi

Geliştirilen test otomasyonu yöntemi, kullanıcının kendi çalışmasına yönelik birtakım ayarlamalar yaptıktan sonra kullanılmaya hazır hale gelir. Bu ayarlamalar, kullanıcının belirlediği taban dünya ve engel tiplerinin tanımlanmasını içerir. Daha sonra, test otomasyonu aracının çalışması iki ana parçadan oluşur. İlk parça, Gazebo ortamında rastgele konumlu ve ebatlı engellerin atanması ve bu şekilde oluşturulan rastgele engellerin bulunduğu çok sayıda dünyanın oluşturulmasını içerir. İkinci parça ise performansı test edilecek navigasyon algoritmasının oluşturulan tüm dünyalarda çalıştırılmasını sağlama ve performans metriklerinin kayıt edilmesi aşamasıdır. Bu adımların sırasıyla kullanıcı tarafından yapılması gereken kısımları ve test otomasyon aracının yaptığı kısımları özetleyen çalışma diyagramı Şekil 1'de gösterilmiştir.



Şekil 1: Test otomasyonu çalışma diyagramı.

2.1. Çok Sayıda Simülasyon Dünyası Oluşturulması

Test otomasyon aracının ilk aşaması, rastgele konumlu ve ebatlı engellerin atanmış olduğu çok sayıdaki Gazebo dünyasının oluşturulmasıdır. Bu aşamada kullanıcı öncelikle her dünyada mevcut olacak sabit yapıları oluşturmalı, dünya ebatını belirlemeli ve araca girdi olarak vermeli. Sabit yapılar, duvarlar veya her dünyada bulunacak diğer sabit engelleri içerebilir. Ardından, engel olarak atanacak modellerin (örneğin küp, silindir, küre gibi) biçimleri tanımlanmalı ve .sdf dosyası olarak kaydedilmelidir. Eğer engel yüksekliği gibi parametrelerin sabit kalması isteniyorsa, model yapısı içinde bu parametreler de tanımlanmalıdır. Sabit dünya oluşturulduktan sonra, "Gazebo Spawn Model" servisi test otomasyon aracında ilk blok olarak çağrılır. Bu blok içinde, tanımlanan modellerden kaçar adetinin, hangi koordinat aralıklarına, hangi en, genişlik ve yükseklik aralıklarında rastgele atanacağı ile ilgili parametre ayarlamaları yapılabilir. Servis çağrıldıktan sonra, ilk rastgele engellerden oluşan dünya kullanıcıya sunulur ve bu Gazebo dünyasının .world uzantılı olarak kaydedilmesi kullanıcı tarafından yapılır.

Test otomasyon aracının ikinci bloğunda oluşturulmuş olan rastgele engellerden oluşan ilk dünyanın çoğaltılması yapılır. Burada .world uzantılı Gazebo dünyası XML formatında kayıtlı olduğundan test otomasyon aracı burada ilgili dosyayı okuyarak belirlenmiş olan küp, silindir, küre gibi rastgele atanacak engellerin pozisyon bilgilerini, en ve genişlik gibi bilgilerini yine istenen aralıklarda rastgele olacak şekilde değiştirip istenen sayıda yeni .world dosyaları üretmektedir. Bunun sonucunda Monte Carlo simülasyonları yapılacak miktarda rastgele ortam elde edilmiş olur.

2.2. Farklı Ortamlarda Otomatik Seri Simülasyon Yapılması ve Performans Metrikleri Kaydı

Çok sayıda ve istenen özelliklere sahip simülasyon ortamlarının elde edilmesinden sonra, manuel olarak ROS launch dosyalarını her adımda düzenlemek, dünya değiştirmek, otonom robotu dünyaya çağırmak ve navigasyon araçlarını çalıştırmak, geliştirici için zaman alıcı ve performans metriklerinin tam izlenmesini güçleştiren bir süreçtir. Bu nedenle, bu aşamayı da otomatize etmek gereklidir.

Test otomasyon aracının bu ikinci kısmında, kullanıcı manuel olarak çalıştırmak istediği launch dosyaları ve diğer kod bloklarının konumlarını araca girer. Ayrıca, otonom robotun başlangıç konumunu ve hedef gösterilecek noktayı belirler. Otomasyon aracı, simülasyonun başladığına dair bilgiyi, hedefin başarıyla robota iletildiğine dair bilgiyi ve bir sonraki simülasyon ortamına geçildiğine dair bilgiyi bir kayıt dosyasına işler. Aynı zamanda, kullanıcı kendi algoritmasının performans metriklerini (örneğin çalışma süresi, güvenlik metriği, alınan yol gibi) kendi algoritmasında hesaplıyorsa, bu metrikleri aynı kayıt dosyasına kaydetme imkanına sahiptir. Örnek olarak rota planlayıcı algoritmalarda kullanılan performans metrikleri aşağıdaki gibidir [8]:

- Güvenlik metriği:

$$dis(t) = \begin{cases} \frac{1}{d_{min_o}} - \frac{1}{d_0} & d_{min_o} \leq d_0 \\ 0 & d_{min_o} > d_0 \end{cases} \quad (1)$$

$$f_s(t) = \sqrt{\frac{\int (dis(t)^2 dt)}{T}} \quad (2)$$

- Toplam kat edilen mesafe:

$$D = \sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad (3)$$

- Konfor metriği:

$$|a_t(t)| = \sqrt{a_x^2 + a_y^2} \quad (4)$$

$$f_c(t) = \sqrt{\frac{\int |a_t(t)|^2 dt}{T}} \quad (5)$$

Denklem (1) ve Denklem (2)'de $dis(t)$, engele olan yakınlıktan hesaplanan mesafe değeridir, d_{min_o}, robot ile en yakın engel arasındaki mesafe, d_0 ise robot ile engel arasındaki herhangi bir çarpışma tehlikesi sunmayacak mesafe değeridir. $f_s(t)$ ise hesaplanan mesafenin RMS değeridir, güvenlik metriği olarak adlandırılır. Denklem (3)'te D, toplam kat edilen mesafeyi ifade etmektedir, burada x_i, i döngüsünde lokalizasyon algoritmasından gelen aracın x konumu bilgisini, y_i, i döngüsünde lokalizasyon algoritmasından gelen aracın y konumu bilgisini göstermektedir. Denklem (4) ve (5) için, $a_x$ve a_y sırasıyla aracın x ve y doğrultularında olan ivmelerini, $|a(t)|$ toplam ivmeyi göstermektedir. $f_c(t)$ ise toplam ivmenin RMS değeri olarak hesaplanan konfor metriğidir.

Test otomasyon aracı, tüm simülasyonları tamamladığında kıyaslanan algoritmaların performanslarını karşılaştırabilir ve hangi simülasyon dünyasında hangi algoritmanın daha başarılı olduğu bilgisini kullanıcıya sunar. Bu sayede geliştirici, farklı ortamlarda algoritmasının performansını otomatik olarak seri bir şekilde test edebilir ve karar verme sürecini kolaylaştırır.

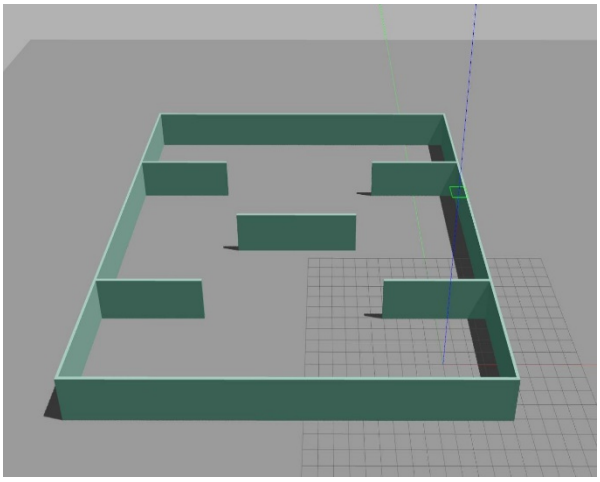
3. Örnek Çalışma

Bu yayındaki örnek çalışma, FGM (Follow the Gap Method) [9] olarak adlandırılan lokal planlayıcı ve engelden kaçınma algoritmasının geliştirilmekte olan yeni bir versiyonuyla karşılaştırılması amacıyla yapılmıştır.

FGM, otonom kara aracının karşısına çıkan engelleri aşmak için global planlayıcının işaret ettiği yönelim açısı ve çarpma tehlikesi olan engellerin arasındaki boşluğun hesaplanan orta noktası olan güvenli noktaya olan yönelim açısı arasında, engele olan yakınlığa göre bir ağırlıklandırma yaparak aracın çarpmadan ilerlemesini sağlayan basit, kullanımı kolay ve etkili bir algoritmadır.

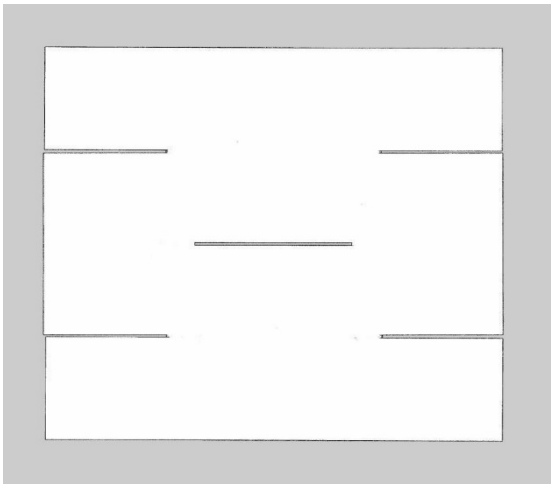
Karşılaştırma amacıyla, FGM ve yeni geliştirilmekte olan FGM algoritmaları iki farklı lokal planlayıcı olarak seçilmiştir. Bu iki algoritmanın performanslarının karşılaştırılması için, engelleri rastgele dağıtılmış 500 farklı simülasyon ortamında her iki algoritmanın da otonom aracı başlangıç noktasından haritada olmayan engelleri aşıp hedef noktaya ulaştırma performansı incelenmiştir. Bu şekilde farklı senaryolara maruz bırakılan iki lokal planlayıcı, Monte Carlo simülasyonları yöntemiyle kıyaslanarak performansları ortaya çıkarılmıştır.

Örnek çalışma için ilk olarak, otonom tekerlekli sandalye simülasyonunda aracın boyutları göz önünde bulundurularak ve lokal planlayıcı algoritmalarının performansını test etmeye uygun bir alan sağlanması amacıyla 30m x 30m ölçülerinde sabit duvarlara sahip bir taban dünya oluşturulmuştur. Oluşturulan taban dünyanın görüntüsü Şekil 2'de verilmiştir.



Şekil 2: Kullanılan taban dünya.

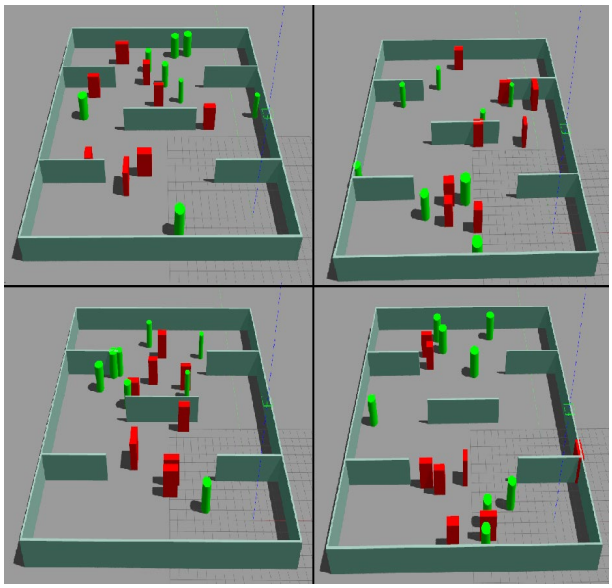
Bu taban dünyada, otonom araç modeli ile ROS'ta mevcut olan SLAM yöntemlerinden Gmapping [10] kullanılarak harita çıkarma işlemi yapılmıştır. Bu harita kullanılarak navigasyon araçları çalıştırıldığında global planlayıcı engellerin olmadığı ortamda hedefe giden rotayı çizecek ancak aracın bu rotada karşısına çıkan engelleri lokal planlayıcının başarısına göre aşıp aşamayacağı görülmüş olacaktır. Elde edilen harita Şekil 3'te gösterilmiştir.



Şekil 3: Taban dünyanın SLAM ile elde edilmiş haritası.

Ardından, bu taban dünyada belirlenen koordinat aralıklarına rastgele engel atama işlemi için engel tipleri tanımlanmıştır. Seçilen engel tipleri küp ve silindir şeklinde düşünülmüş, modelleri kütle, malzeme gibi bilgilerini barındıracak birer .sdf dosyası olarak kaydedilmiştir. ROS servisleri kullanılarak 8 küp ve 8 silindir yine belirlenen ebat aralıklarında rastgele ebata sahip olacak şekilde çağrılmıştır. Bu işlemin ardından elde edilen, rastgele silindir ve küp engellerin

atanmış olduğu simülasyon ortamlarından bazıları Şekil 4'te görülmektedir.



Şekil 4: Rastgele dağıtılmış engellerden oluşan dünyalar.

Engellerle beraber olan simülasyon dünyası .world dosyası olarak kaydedilmiştir. Bu .world dosyası XML formatında, simülasyon ortamında bulunan tüm duvarların, engellerin atalet, kütle, konum, görünüm gibi bilgilerini bulundurmaktadır. 500 simülasyon ortamı oluşturmak için tekrar tekrar ROS servisi çağırıp ortamı .world formatında kaydetmek zaman alıcı ve verimsiz olacağından, XML formatındaki .world dosyasında küp ve silindir olarak atanmış olan engellerin koordinat bilgileri belirlenen aralıkta rastgele şekilde değiştirilip kopyalanıp tekrar kaydedilmesiyle 500 ayrı .world dosyası oluşturulmuştur.

İki lokal planlayıcı kıyaslanmasında ise seçtiğimiz performans metrikleri aşağıdaki gibidir:

- Aracın çarpışma olmadan hedefe ulaşması
- Hedefe ulaşıncaya kadar aracın toplam kat ettiği yol mesafesi
- Lokal planlayıcının rota boyunca sergilediği ortalama çalışma hızı

Burada kat edilen toplam yol mesafesi için bölüm 2'de Denklem (3) ile gösterilmiş olan eşitlik kullanılmıştır. Bu metriklerin de test otomasyonu aracı kullanılarak kaydedilebilmesi için bir kayıt dosyası oluşturulmuştur. Bu kayıt dosyasına her simülasyon için 500 ortamdan hangisinin başlatıldığı, hedef noktanın başarıyla navigasyon aracına iletilip iletilmediği, aracın çarpışma yaşayıp yaşamadığı, hedefe ulaştığında toplam kat ettiği mesafe, ortalama çalışma hızı yazdırılmıştır. Bu şekilde iki lokal planlayıcının kaydedilen kayıt dosyaları performans metriklerini inceleme ve kıyaslama yapma imkanı ortaya koymaktadır. Elde edilen kayıt dosyası örneği Şekil 5'te standart FGM Şekil 6'da geliştirilmekte olan FGM için gösterilmiştir.

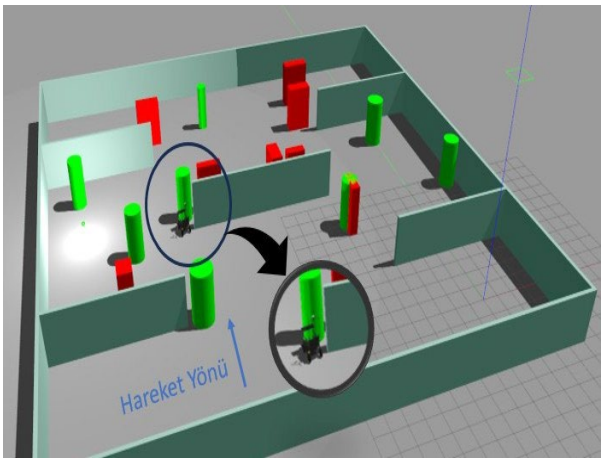
```
Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 137
Goal Reached! Total distance traveled is: 35.0726 || Avg execution time per cycle is: 0.000582011

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 138
Collision occured! || Avg execution time per cycle is: 0.000753994

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 139
Collision occured! || Avg execution time per cycle is: 0.000743682

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 140
Goal Reached! Total distance traveled is: 34.7451 || Avg execution time per cycle is: 0.000688841

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 141
Collision occured! || Avg execution time per cycle is: 0.000753788

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 142
Collision occured! || Avg execution time per cycle is: 0.000704724

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 143
Goal Reached! Total distance traveled is: 34.5757 || Avg execution time per cycle is: 0.000656842
```

Şekil 5: Standart FGM kayıt dosyası.

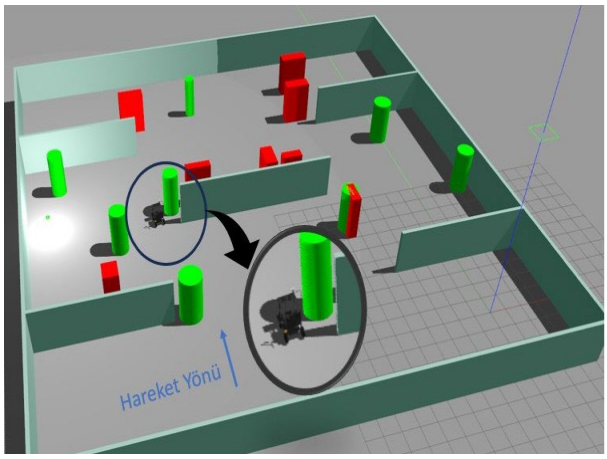
```
Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 137
Goal Reached! Total distance traveled is: 35.1337 || Avg execution time per cycle is: 0.0343953

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 138
Collision occured! || Avg execution time per cycle is: 0.0255659

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 139
Goal Reached! Total distance traveled is: 34.3717 || Avg execution time per cycle is: 0.0161139

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 140
Goal Reached! Total distance traveled is: 34.8314 || Avg execution time per cycle is: 0.0231711

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 141
Goal Reached! Total distance traveled is: 34.3172 || Avg execution time per cycle is: 0.0147511

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 142
Collision occured! || Avg execution time per cycle is: 0.0181831

Simulation Started! || Goal successfully published at (-22.0, 22.0) in world 143
Goal Reached! Total distance traveled is: 34.8367 || Avg execution time per cycle is: 0.0232428
```

Şekil 6: Geliştirilmekte olan FGM kayıt dosyası.

Kayıt dosyasından da görülebileceği üzere kıyaslanan iki lokal planlayıcıda, geliştirilmekte olan FGM yönteminin aracı hedefe ulaştırırken başarılı olduğu, ancak standart FGM yönteminin ise çarpışma yaşadığı örnek simülasyon kayıtları bulunmaktadır. Bunun yanında kat edilen mesafede önemli bir değişim olmadığı, geliştirilmekte olan FGM yönteminin çalışma hızının lokal planlayıcı için kabul edilebilir düzeyde daha yavaş olduğu görülmektedir. Bu kayıtlardan yola çıkarak, bir lokal planlayıcının diğerine göre avantajları ve dezavantajları kolaylıkla tespit edilmiş olur. Bu şekilde geliştirilmekte olan FGM'nin standart FGM'ye üstün olduğu tespit edilen simülasyon ortamlarından biri olan Şekil 5 ve Şekil 6'dan da görülen "world 139" manuel olarak çalıştırıldığında, Şekil 7'de standart FGM için ve Şekil 8'de geliştirilmekte olan FGM için gösterilen performans farklılıkları Gazebo üzerinde izlenebilir. Aynı şekilde her iki algoritmanın da başarısız olduğu "world 142" de tekrar manuel çalıştırılarak iki algoritmanın benzer davranışlarda bulunduğu senaryolar da incelenebilir. Bu veriler ışığında algoritmaların hangi senaryoda nasıl farklı işledikleri daha detaylı incelenebilir. Test otomasyon aracı kullanılarak performans farklılıkları ve benzerlikleri kolaylıkla tespit edilmiş olur, yeni geliştirmeler için öngörüler elde edilir.



Şekil 7: Standart FGM engele takılma durumu.



Şekil 8: Geliştirilmekte olan FGM engeli aşma durumu.

4. Sonuçlar

Bu çalışmada, Gazebo simülasyon ortamı ve ROS kullanılarak rastgele engellerden oluşan çoklu simülasyon ortamlarının elde edilmesine yönelik bir yöntem paylaşılmıştır. Bu yöntemin özellikle navigasyon araçlarının lokal planlayıcılarının test edilmesi ve geliştirilmesinde faydalı olabileceği açıklanmıştır. Elde edilen simülasyon ortamlarıyla seri şekilde Monte Carlo simülasyonları yapılma yöntemi aktarılmış ve örnek bir çalışma olarak da kendi geliştirmekte olduğumuz lokal planlayıcının önceki versiyonuyla kıyaslaması örneği verilmiştir.

Geliştirilen test otomasyonu aracı, kullanılacak robota göre, hesaplanmak istenen performans metriklerine göre, engel tipleri ve boyutlarına göre kullanıcıya yönelik şekillendirilerek uygulanmaya müsaittir. Kullanıcı, bu aracı kullanarak farklı senaryoları hızlı bir şekilde test edebilir ve algoritmalarının performansını kıyaslayabilir.

İlerideki çalışmalarda, otomasyon aracının daha da geliştirilerek istenen algoritma parametrelerinin otomatik olarak değiştirilmesi ve simülasyonların sürdürülmesi sağlanabilir. Böylece otomasyon aracı, optimum parametreleri bulma konusunda da katkı sağlayabilir. Aynı zamanda, farklı engel tipleri ve dünya özellikleri kullanarak daha geniş bir test yelpazesine ulaşmak için çeşitli senaryoların eklenmesi de olasıdır.

Sonuç olarak, bu çalışma, otonom mobil robotların geliştirilen algoritmalarının performansını değerlendirmek ve yeni algoritmaların testini hızlı ve güvenli bir şekilde yapmak için etkili bir test otomasyonu aracının geliştirilmesini sağlamıştır. Bu tür araçlar, otonom sistemlerin geliştirilmesi sürecinde önemli bir rol oynayabilir ve yeni algoritmaların daha hızlı bir şekilde piyasaya sürülmesine katkıda bulunabilir.

Teşekkür

Bu çalışma, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından 121E537 proje numarasıyla desteklenmiştir.

Kaynakça

[1] Sotiropoulos, T., Waeselynck, H., Guiochet, J., & Ingrand, F. (2017, July). Can robot navigation bugs be found in simulation? an exploratory study. In *2017 IEEE International conference on software quality, reliability and security (QRS)* (pp. 150-159). IEEE.

[2] Afzal, A., Katz, D. S., Le Goues, C., & Timperley, C. S. (2021, April). Simulation for robotics test automation: Developer perspectives. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)* (pp. 263-274). IEEE.

[3] Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)* (Vol. 3, pp. 2149-2154). IEEE.

[4] Noori, F. M., Portugal, D., Rocha, R. P., & Couceiro, M. S. (2017, October). On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo?. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)* (pp. 19-24). IEEE.

[5] Lavrenov, R., Zakiev, A., & Magid, E. (2017, May). Automatic mapping and filtering tool: From a sensor-based occupancy grid to a 3D Gazebo octomap. In *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)* (pp. 190-195). IEEE.

[6] Lavrenov, R., & Zakiev, A. (2017, June). Tool for 3D Gazebo map construction from arbitrary images and laser scans. In *2017 10th International Conference on Developments in eSystems Engineering (DeSE)* (pp. 256-261). IEEE.

[7] Abbyasov, B., Lavrenov, R., Zakiev, A., Yakovlev, K., Svinin, M., & Magid, E. (2020, May). Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 7226-7232). IEEE.

[8] Sezer, V. (2022). An optimized path tracking approach considering obstacle avoidance and comfort. *Journal of Intelligent & Robotic Systems*, *105*(1), 21.

[9] Sezer, V., & Gokasan, M. (2012). A novel obstacle avoidance algorithm:"Follow the Gap Method". *Robotics and Autonomous Systems*, *60*(9), 1123-1134.

[10] Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, *23*(1), 34-46.