JOURNAL OF SCIENTIFIC, TECHNOLOGY AND ENGINEERING RESEARCH
*Bilim, Teknoloji ve Mühendislik Araştırmaları Dergisi, (2021) - 2(2): 31-45*
*ISSN* : 2717-8404 *https://dergipark.org.tr/tr/pub/jster*

RESEARCH ARTICLE

# Industrial Robot Motion Planning Algorithms Performance Benchmarking

[1] Alim Kerem Erdoğmuş,[2*] Uğur Yayan

[1]İnovasyon Mühendislik Ltd. Şti., R&D, Eskişehir, Turkey,kerem.erdogmus@inovasyonmuhendislik.com Orcid. 0000-0001-5111-5965
[2]Eskisehir Osmangazi University, Software Engineering Department, Eskişehir, Turkey,ugur.yayan@ogu.edu.tr Orcid. 0000-0003-1394-5209

## HIGHLIGHTS

- *Effect and important of this article in literature*
- *Exchange between sources in related subjects of this article*
- *Contribution and strongest impact on the related sobject of this article*
- *Examined study and obtained results why is important*

**Keywords:**

- Robotics
- Verification&Validation
- Trajectory Planning
- Industrial Quality Control

**Article Info:**

Received : *07.08.2021*
Accepted : *17.09.2021*
Published : *21.12.2021*

**DOI:10.53525/jster.979689**

***Correspondence:**
Uğur Yayan
ugur.yayan@ogu.edu.tr
Tel: +90 222 239 37 50

## GRAPHICAL ABSTRACT

In robotics studies, motion and trajectory planning for industrial robot systems is one of the most actively studied topics. "Automated Robot Inspection Cell for Quality Control of Automotive Body-in-White (ROKOS)" has been defined as the Use-case Scenario within the scope of the VALU3S project, where studies are carried out for the verification and validation (V&V) of autonomous systems. In this study, ROKOS system, developed in the real environment, was transferred to the GAZEBO simulation environment for the V&V of the defined use case scenario, and SRVT (Simulation Based Robot Verification Testing Tool) was revealed and simulation-based tests of the evaluation scenarios created within the scope of the project were carried out on this system. Figure A shows the results of the Full Test with Reset.

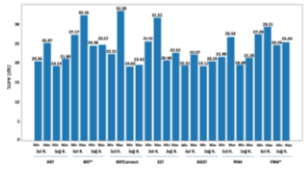


**Figure A.** Full test with reset results

**Aim of Article:** In this study, within the scope of "Safety Trajectory Optimization", which is one of the evaluation scenarios, it is aimed to verify and validate the software developed by creating trajectories in which the robot arms of the ROKOS system will move safely according to the determined evaluation criteria for the quality control of the bus chassis transferred to the simulation environment.

**Theory and Methodology :** The OMPL and EST trajectory planning algorithms of the ROS MoveIt tool were studied, and the pros and cons of these algorithms were determined. In order to verify these algorithms in the simulation environment, detailed tests were carried out on the usage scenario and the results were analyzed. Tests were conducted for 3 different scenarios (Speed Test, Full Test with Reset (FTR) and Full Test without Reset (FToR)) and validation activities were improved in terms of time and cost for the existing ROKOS system transferred to SRVT.

**Findings and Results:** Within the scope of these studies, more than 400 tests were carried out on many motion planning algorithms of the OMPL planner, and the most optimal planning algorithm was tried to be found. At the end of the study, it was determined that this planning algorithm should be BiEST or PRM and the tests were completed.

**Conclusion : T**he SRVT system was created ROS tools, Gazebo and ROKOS. The study was carried out on the examination and comparison of the planning performances of the motion planning algorithms used through the software. In order to reduce the task completion times of the ROKOS integrated into the SRVT ecosystem, it has been focused on determining the most suitable trajectory planning algorithm for this system. Before moving on to these applications, many basic motion planning algorithms in the OMPL library were examined, and the operability and effectiveness of the motion planning mentality of these algorithms in the SRVT integration of the ROKOS system were studied. In the light of research and tests, it has been seen that the task completion times of the ROKOS system can be considerably reduced when appropriate algorithms are detected.

JOURNAL OF SCIENTIFIC, TECHNOLOGY AND ENGINEERING RESEARCH
*Bilim, Teknoloji ve Mühendislik Araştırmaları Dergisi, (2021) - 2(2): 31-45*
*ISSN* : 2717-8404 *https://dergipark.org.tr/tr/pub/jster*

ARAŞTIRMA MAKALESİ

# Endüstriyel Robot Hareket Planlama Algoritmaları Performans Karşılaştırması

[1] Alim Kerem Erdoğmuş, [2*] Uğur Yayan

[1]İnovasyon Mühendislik Ltd. Şti., Ar-Ge Bölümü, Eskişehir, Türkiye.kerem.erdogmus@inovasyonmuhendislik.com,Orcid.0000-0001-5111-5965
[2]Eskişehir Osmangazi Üniversitesi, Yazılım Mühendisliği Bölümü, Eskişehir, Türkiye., ugur.yayan@ogu.edu.tr, Orcid.0000-0003-1394-5209

**Alıntı :**
Erdoğmuş, A.K., Yayan, U., (2021). Endüstriyel Robot Hareket Planlama Algoritmaları Performans Karşılaştırması, *Journal of Scientific Technology and Engineering Research.* (2021) - 2(2) : 31-45. DOI: 10.53525/jster.979689

## ÖNE ÇIKANLAR

- *Emniyetli Yörünge Optimizasyon kapsamında, simülasyon ortamına aktarılmış olan otobüs şasesinin kalite kontrolü*
- *ROKOS sistemine ait robot kollarının belirlenen değerlendirme kriterine göre emniyetli bir şekilde hareket edecekleri yörüngelerin oluşturulması*
- *ROS MoveIt aracına ait OMPL ve EST yörünge planlama algoritmaların artı ve eksi yönlerinin belirlenmesi*
- *İncelenen algoritmaların simülasyon ortamında doğrulanmaları için kullanım senaryosu üzerinde detaylı testler*

**Makale Bilgileri**

*Geliş Tarihi : 07.08.2021*
*Kabul Tarihi: 17.09.2021*
*Yayım Tarihi*: *21.12.2021*

**DOI:10.53525/jster.979689**

***Sorumlu Yazar:**

Uğur Yayan,
ugur.yayan@ogu.edu.tr
Tel: +90 222 239 37 50

**ÖZET**

*Robotik çalışmalarında, endüstriyel robot sistemleri için hareket ve yörünge planlama, aktif olarak çalışılan konuların başında gelmektedir. Otonom sistemlerin doğrulanması ve onaylanması (V&V) için çalışmalar yürütülen VALU3S projesi kapsamında Kullanım Senaryosu olarak "Araç Şase Kalite Kontrolü için Otonom Robot Denetleme Hücresi (ROKOS)" tanımlanmıştır. Bu çalışmada, tanımlanmış kullanım senaryosunun doğrulanması ve onaylanması amacıyla gerçek ortamda geliştirilmiş ROKOS sistemi, GAZEBO simülasyon ortamına aktarılarak SRVT (Simulasyon Tabanlı Robot Doğrulama Test Sistemi) ortaya çıkarılmış ve bu sistem üzerinde ise proje kapsamında oluşturulan değerlendirme senaryolarının simülasyon tabanlı testleri yapılmıştır. Bu çalışmada, değerlendirme senaryolarından "Emniyetli Yörünge Optimizasyon" kapsamında, simülasyon ortamına aktarılmış olan otobüs şasesinin kalite kontrolü için ROKOS sistemine ait robot kollarının belirlenen değerlendirme kriterine göre emniyetli bir şekilde hareket edecekleri yörüngelerin oluşturularak geliştirilen yazılımların doğrulama ve onaylama işlemlerinin yapılması hedeflenmiştir. Bu kapsamda, ROS MoveIt aracına ait OMPL ve EST yörünge planlama algoritmaları üzerinde çalışılmış, bu algoritmaların artı ve eksi yönleri belirlenmiştir. Bu algoritmaların simülasyon ortamında doğrulanmaları için kullanım senaryosu üzerinde detaylı testler yapılmış ve sonuçları analiz edilmiştir. Testler 3 farklı senaryo (Hızlı test, Resetli Tam Test ve Resetsiz Tam Test) için yapılmış ve SRVT'ye aktarılmış mevcut ROKOS sistemi için doğrulama faaliyetleri zaman ve maliyet açısından iyileştirilmiştir.*

**Anahtar Kelimeler:** *Robotik, Doğrulama&Onaylama, Yörünge Planlama Algoritmaları, Endüstriyel Kalite Kontrol*

## I. GİRİŞ

Son yıllarda insan gücü ve aklı ile ilerlemekte ve gelişmekte olan endüstriyel çalışmalar için kullanılmaya başlanmış basit robotlar yerini karmaşık otonom robotlara bırakmakta, endüstrinin yanı sıra sağlık, uzay ve askeriye gibi kritik alanda da bu tarz robotlar ve robotik uygulamaların desteklendiği ve yaygınlaştığı görülmektedir [19-22]. Bu çok hızlı gelişen yayılmanın bir nedeni, robotların mükemmel çok yönlülüğü ve esnekliğidir. Bu esneklik onları farklı görevleri yerine getirmek için çok uygun hale getirmektedir [16]. Endüstriyel robotlarla ilgili olarak, bunları bir manipülatör ve eklemlerle [17] birbirine bağlanan bir dizi sert koldan oluşan mekanik bir yapı olarak tanımlamak mümkündür.

Hareket kabiliyeti sağlayan bir yapı, el becerisi veren bir bilek ve robotun kullanıldığı görevi yerine getiren bir uç efektör belirlenmesi ile endüstriyel alanda çalışan kompleks yapıda bir robotun temel altyapısı tanımlanmış olur.

Konu endüstriyel robotik sistemlere geldiğinde, sahadaki görevlerin tamamlanmasında zaman çok önemli bir faktördür. Sistemlere zaman kazandıracak ve görevleri sorunsuz gerçekleştirmelerini sağlayacak unsurlar da robotik kol sistemleri için kritik önem taşıyan yörünge planlama algoritmalarıdır. Yörünge planlama, hesaplama açısından veri yoğunluğu oluşturabilen ve robotikte zaman alıcı olabilen bir konudur [18]. Bir robotun, bir ortam hakkında bilgi edinmek için sensörlerden gelen bilgilere erişmesi ve bunları işlemesi gerekir. Yörünge planlamanın zorluğu, en uygun yolu bulmayı ve hassas manipülasyonları yönetmeyi içerir. Bir yörünge planlama algoritması için tüm olası yörüngelerde bir arama işlenirken, bir kullanıcının robottan yanıt almadan önce prosedürün tamamlanmasını beklemesi gerekir. Bu tarz prosedürler, robotikte hareket planlamanın performansının, yapılan işin kalitesi ve zamanlaması konusundaki hassasiyetini arttırmaktadır. Bu durum da planlama algoritmalarının sisteme uygun bir şekilde seçilmesinin önemini vurgulamaktadır.

Robotik sistemlerde yörünge planlamaları için farklı yörünge planlama algoritmalarından faydalanılmakta ve bu sistemler için yeni birçok algoritma geliştirilmektedir. Gasparetto ve ekibinin [23] yaptığı çalışmalar, doğrudan robotikteki yol bulma ve yörünge belirleme algoritmalarından bazıları üzerinde yoğunlaşmıştır. Bu çalışmadaki planlama çeşitleri, robot kollarının bulundukları konumlar ile ortam engelleri arasındaki mesafelerin referans alındığı yol haritası (roadmap) teknikleri, hücre ayrışması (cell decomposition) algoritmaları ve yapay potansiyel (artificial potential) metotları olarak üç farklı çeşitte incelenmiştir. Bu metotların hepsi, ortamın geometrik açıdan sınıflandırılması ile geliştirilen farklı yaklaşımları ele alır. Robotik kollar için yörünge planlaması için geliştirilen yaklaşımlardan biri Streinu'ya aittir [24]. Yaptığı çalışmada robot kollarının hareket kinematiklerini ve terminolojilerini belirlemiş, bu bağlamda robot kolu için geliştirilecek algoritmalara yol gösteren bir çalışma gerçekleştirmiştir. Moll ve ekibinin çalışmalarında [36], Streinu'nun çalışmasındakinden farklı mantalitede bir değerlendirme (benchmark) yapısı geliştirmişlerdir. Bu yapı, kullanıcı tanımlı hareket planlama algoritma sorunlarını çözme odaklıdır. Cohen ve ekibinin çalışmalarında [37], hareket planları ile ilgili farklı bir sorun olan hareketli robotik kolların, robot hareketi ile kol hareketi hızlarının uyumsuzluğundan kaynaklanan planlama sıkıntısına değinilmiştir. Bu sorunu çözmek için yine farklı planlama algoritmalarını incelemişler ve bir değerlendirme sunucusu oluşturarak bir yaklaşım geliştirmişlerdir. Liu ve ekibinin çalışmaları [38], robotik yörünge planlama optimizasyonu testlerine bir alternatif olacak şekildedir. Tünel, kütüphane ve endüstriyel alanlar gibi farklı ortamlarda, farklı planlama algoritmaları ile yapılan testler, algoritmaların planlama yapma süreleri baz alınarak gerçekleştirilmiştir. Testlerde algoritmaların başarı yüzdeleri, yolu pürüzsüz geçiş süreleri gibi parametreler de baz alınmıştır.

Yörünge planlama için kullanılan algoritmalardan birisi olan STOMP (Stoastic Trajectory Optimization) [7], genel kısıtlamalarla başa çıkabilen bir hareket planlama yaklaşımıdır. Bu yaklaşım, bir dizi gürültülü yörüngeden istifade ederek stokastik yörünge optimizasyonunu kullanmaktadır. Her tekrar uygulamada, bir dizi yörünge üretilir. Oluşturulan yörüngeler, daha sonra aday çözümünü güncellemek için kullanılan maliyetleri belirlemek amacı ile simüle edilmektedir. Bu süreçte gradyan bilgisi gerekmediğinden, genel kısıtlamalar ve ek sorunsuz maliyetler optimize edilebilir. Diğer bir planlama algoritması olarak CHOMP (Covariant Hamiltonian Optimization) [6], hareket planlaması için iyi yörüngeler elde etmek amacıyla oluşturulmuş basit bir varyasyon stratejisidir. CHOMP, dışbükey olmayan maliyet fonksiyonlarında yüksek boyutlu uzaylarda optimizasyon problemlerini ele almaktadır. Son olarak planlama algoritmalarının en yaygın tarafında ise OMPL (Open Motion Planning Library) kütüphanesi ve bu kütüphane bünyesindeki algoritmalar bulunmaktadır. Bu algoritmalar ROS temelli robotik uygulamalarda kullanılabilirliği yüksek, yazılım geliştirmelerine açık algoritmalar olduğu için, günümüzde çok daha yaygın kullanılmaya başlamıştır [1]. Xinyu ve ekibi [25] OMPL'in algoritmalarından RRT* algoritmasının çift yönlü hareketteki etkisi üzerine çalışmıştır. Çalışmasında RRT* algoritmasının oldukça optimize olduğunu ancak işlem süresinin uzun ve yavaş olduğuna dikkat çekmiştir. LaValle ve ekibinin çalışmasında [8], RRT (Rapidly-Exploring Random Trees) algoritmasının 12 eksene kadar yörünge planlaması yapabilecek kabiliyette olduğu, engel tespitleri ve dizayn parametrelerinin, ideal bir hareket planlayıcı algoritmasında olması gerektiği kalitede olduğunu çeşitli örneklerle açıklamışlardır. Kuffner ve ekibi [9], daha sonra RRT-Connect algoritması ile RRT'deki yörünge planlama menzilini arttıracak yeni

düzenlemeler geliştirmişlerdir. Karaman ve ekibi [10] de RRT ve PRM (Probabilistic Roadmaps) algoritmalarının RRT* ve PRM* yükseltmeleri üzerinde çalışmalar yapmışlardır. Bu geliştirmeler de yine RRT-Connect örneğinde olduğu gibi algoritmaların planlama zamanlarını kısaltmak ve ortam menzillerini genişletmek üzerine tasarlanmıştır. EST (Expansive Space Trees) algoritması, Hsu ve ekibi [34] tarafından ortaya çıkarılmış bir hareket planlama algoritmasıdır. Bu algoritma temelinde, rastgele örneklenmiş kilometre taşlarından oluşan bir yol haritasıyla bağlantısı etkin bir şekilde yakalanabilen bir robot konfigürasyon alanı üzerine odaklanmıştır. Bu alan ailesini karakterize etmek için genişleme kavramı oluşturmak üzerine kurulmuş bir algoritma ortaya çıkarmışlardır.

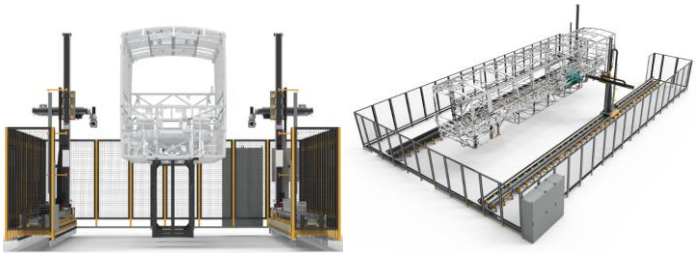
Bu çalışmada, tanımlanmış kullanım senaryosunun doğrulanması ve onaylanması amacıyla gerçek ortamda geliştirilmiş ROKOS sistemi GAZEBO simülasyon ortamına aktarılarak SRVT ortaya çıkarılmış ve bu sistem üzerinde ise proje kapsamında oluşturulan değerlendirme senaryolarının simülasyon tabanlı testleri yapılmıştır.

Bu çalışmada, değerlendirme senaryolarından "Emniyetli Yörünge Optimizasyon" kapsamında, simülasyon ortamına aktarılmış olan otobüs şasesinin kalite kontrolü için ROKOS sistemine ait robot kollarının belirlenen değerlendirme kriterine göre emniyetli bir şekilde hareket edecekleri yörüngelerin oluşturularak geliştirilen yazılımların doğrulama ve onaylama işlemlerinin yapılması hedeflenmiştir. Bu kapsamda, ROS MoveIT aracına ait OMPL ve EST yörünge planlama algoritmaları üzerinde çalışılmış, bu algoritmaların artı ve eksi yönleri belirlenmiştir. Bu algoritmaların simülasyon ortamında doğrulanmaları için kullanım senaryosu üzerinde detaylı testler yapılmış ve sonuçları analiz edilmiştir. Testler 3 farklı senaryo (Hızlı test, Resetli Tam Test ve Resetsiz Tam Test) için yapılmış ve SRVT'ye aktarılmış mevcut ROKOS sistemi için doğrulama faaliyetleri zaman ve maliyet açısından iyileştirilmiştir.

Makale akışı olarak, Bölüm 2'de ROKOS sistemi ve bu sistemin işlevi, Bölüm 3'te bu sistemin entegre edildiği SRVT'yi oluşturan yazılımlar ile ilgili detaylar açıklanmıştır. Bölüm 4'te yörünge planlaması için test edilecek OMPL planlama algoritmaları incelenmiş, Bölüm 5'te ise bu algortimaların performans kıyaslamaları, ROKOS sisteminin test süresi tabloları üzerinden incelenmiştir. Çalışmanın sonuçları ise Bölüm 6'da verilmiştir.

## II. ROKOS (ROBOT KONTROL SİSTEMİ)

ROKOS sisteminin kullanım senaryosu, yenilikçi görsel kontrol teknikleriyle ürünün kalite kontrol süresinin kısaltarak otobüs karoserisindeki parçaların varlık-yokluk kontrolünü daha hassas yapmaya odaklanmaktadır. Bu kullanım senaryosunun temeli, daha iyi kalite kontrolü elde etmek için hataya dayanıklılık açısından daha iyi performanslı bir üretim sistemi sağlamaktır. (Şekil 1). 2500-3000 gövde parçasının varlığının kontrolünün kartezyen robot ve kamera sensör sistemi ile tam otomatik olarak yapılması planlanmaktadır. OTOKAR tarafından geliştirilen ve uzaktaki sunucuda çalıştırılan dijital ikiz yazılımı, tüm parçaların varlığını kontrol etmek için güvenli robot yörünge noktalarını belirler. Sunucudaki veri tabanında depolanan her robot yörüngesi noktası için yazılım, sanal bir kamerayı CAD (Computer-aided Design) ortamında konumlandırır ve sanal iki boyutlu görüntüler oluşturur. Sunucudan gelen yörünge noktalarını girdi olarak işleyen ve sistem bilgisayarında çalıştırılan yazılım, robot eksen motorlarını PLC ve sürücüler aracılığıyla kontrol ederek Otokar'ın geliştirdiği kartezyen robotları gerçek üretim ortamında konumlandırır. Yörüngedeki her konum için kamera sensörleri, gerçek üretim ortamından 2B görüntüler yakalar. Bu konumlar için CAD verilerinden elde edilen sentetik 2B veriler ile kamera sisteminden elde edilen gerçek 2B görüntü verileri karşılaştırılarak parça varlık-yokluk kontrolü yapılır. Sistem durumu ve kalite kontrol raporları, kalite personeline arayüzlerle gösterilir ve depolanır.



**Şekil. 1.** Kalite kontrol için robot denetleme sistemi

Şekil 2'de gösterildiği gibi, hedeflenen sistem topolojisi aşağıdaki gibidir: Sistem uzaktaki bir sunucuya gerçek zamanlı veri ileten ve giriş verileri alan iki güvenli bölgeden oluşmaktadır. Birinci güvenli bölgede sistem bilgisayarı, 2 adet Time of Flight (ToF) kamera ve 2 adet 2B kamera bulunmaktadır. Kameralar sistem bilgisayarına görüntü verilerini direk göndermektedir.

VALU3S projesi kapsamında belirlenmiş Değerlendirme

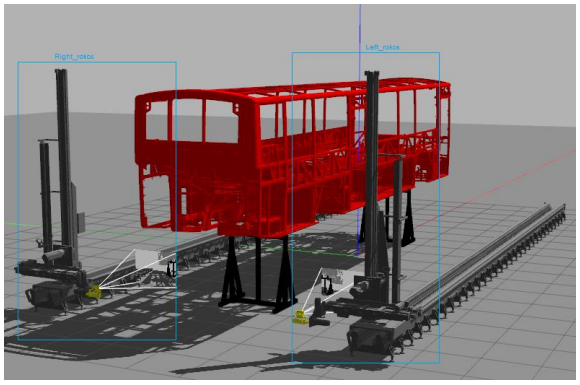
Senaryoları ve Kriterleri (Evaluation Scenarios and Criteria) kapsamında Otokar tarafından sağlanan bu sistemin, "Güvenli Yörünge Optimizasyonu (Safety Trajectory Optimization)" alanında çalışmalar yapılmıştır. Bu değerlendirme senaryosu, robotun ve robot aparatlarının güvenliğini, çalışma alanındaki statik nesneleri de otomatik olarak kapsayan robot yörünge noktaları oluşturmak, robotun güvenli bir hareket rotası çizimini planlamayı amaçlar. Yapılan bu çalışmalarda kullanılan iki ROKOS Kartezyen robot kolunun hareketli tüm eklemleri ayrı ayrı sisteme tanımlanmakta ve kontrol edilmektedir. Bu kontrol, ROS üzerinde geliştirilen yazılımlar vasıtası ile gerçekleştirilmekte, çeşitli kontrol parametreleri doğrultusunda gerçekçi robot hareketleri elde edilmektedir.

## III. YAZILIMLAR VE SİMULASYON ORTAMI

Bu bölümde, ROKOS sisteminin entegre edildiği SRVT ekosistemi ve bu ekosistemi oluşturan yazılımlar hakkında temel bilgilere yer verilmiştir.

Robotik İşletim Sistemi (ROS) [1],[2] olarak tabir edilen sistem, açık kaynak kodlu olarak paylaşılan, robotlar için donanım/yazılım kontrolü, servisler ve iletişim protokolleri gibi birçok robotik aracı sağlayan bir arakatmandır. Robotik sistemleri simulasyon ortamlarında modellemeyi, kontrol etmeyi ve yeni yazılım araçları entegre ederek geliştirmeyi mümkün kılmaktadır.

Gazebo [3], robot modellerini simulasyon ortamında, kinematik özelliklerini gerçeğiyle birebir modellemenin mümkün olduğu, gerçekçi bir ortam simulasyon yazılımıdır. ROS destekli çalışabilmekte, ROS üzerinde Gazebo için geliştirilmiş paketler ile kullanılabilmektedir. Gazebo'nun sağladığı simulasyon ortamında, gerçeğe çok yakın ortamlar oluşturmak ve robotları bu ortamlarda gerçekçi fizik kuralları altında kullanmak mümkündür. Bu çalışmada ROKOS sistemi ve bu sistemin kullanıldığı otobüs şaseleri, gerçeklerine birebir uygun ölçü ve atalet momenti değerleri ile Gazebo ortamına uyarlanmıştır (Şekil 2).



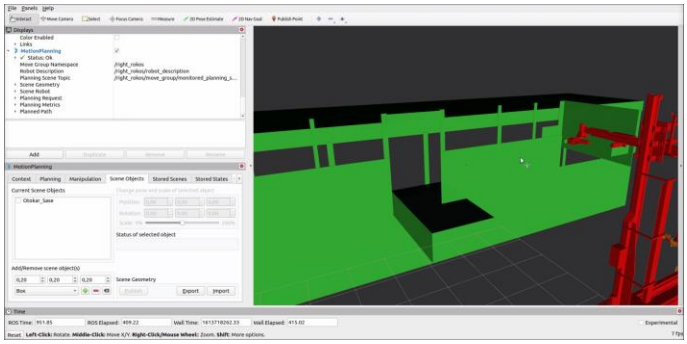

**Şekil. 2.** Gazebo simulasyon ortamına modellenmiş ROKOS sistemi ve otobüs şasesi

SMACH, hızlı bir şekilde karmaşık robot davranışları oluşturmak için görev düzeyinde bir mimaridir. SMACH özünde, hiyerarşik durum makineleri oluşturmak için ROS'tan bağımsız bir Python kitaplığıdır. SMACH, bakımı yapılabilen ve modüler kod ile hızlı bir şekilde sağlam robot davranışı oluşturmak için çok eski konseptlerden yararlanan yeni bir kütüphanedir. Bu kütüphane kullanılarak, robot kolların çeşitli görev durumlarının kolaylıkla kontrolü mümkün olmaktadır.

Moveit! [4],[15] ROS'un hareket planlama yazılımıdır (Şekil 3). Robot, robot kolu ve dronlar gibi robotik araçların hareket kontrollerini ve planlamalarını sağlayan bir ortam sağlar. Bu planlama sırasında robotların sensörlerini, ortam haritalarını ve hareket planlama algoritmalarını kullanır. Robotun URDF modellerini yeniden düzenleyerek, bu sistem içinde çalışabilir SRDF tanımlama dosyasına çevirir ve planlayıcı algoritmalarını işletir. Bu dosya, ilgili robotun hareketli ya da hareketsiz parçalarının tanımlanmasını ve bu parçaların gerekli parametrelerinin tanımlanmasında kullanılır.

Moveit!, ROS'ta hareket planlama ve mobil manipülasyon için birincil yazılım aracıdır ve PR2, Robonaut ve DARPA'nın Atlas robotu dahil olmak üzere birçok robotla başarılı bir şekilde entegre edilmiştir. Moveit! tamamen C++ ile yazılmıştır, ancak Python bağlantılarını da içerir. Moveit! varsayılan olarak çekirdek ROS derleme ve mesajlaşma sistemlerini kullanır. Bileşenleri kolayca değiştirebilmek amacıyla eklentileri kullanır: hareket planlama eklentileri (örn. OMPL), çarpışma algılama (örn. Hızlı Çarpışma Kitaplığı (FCL)), kinematik eklentileri (örn. OROCOS Kinematik ve Dynamics Library (KDL)). MoveIt!'in hedef uygulaması endüstriyel, ticari ve araştırma ortamlarında manipülasyonaa yönelik geliştirilmiştir [35].

JOURNAL OF SCIENTIFIC, TECHNOLOGY AND ENGINEERING RESEARCH
*Bilim, Teknoloji ve Mühendislik Araştırmaları Dergisi, (2021) - 2(2): 31-45*
*ISSN* : 2717-8404 *https://dergipark.org.tr/tr/pub/jster*



**Şekil. 3.** MoveIt! arayüzü

Moveit! yazılımı, içerisinde OMPL, CHOMP ve STOMP planlayıcıları ve bu planlayıcıların hareket planlama algoritmaları bulundurmaktadır. CHOMP (Covariant Hamiltonian Optimization for Motion Planning) kütüphanesi, robotik sistemlerdeki rotasyon optimizasyonunu sağlayabilmek için yeniden parametrelendirme yapılabilmesini sağlayan yöntemler oluşturan algoritmaları barındıracak bir kütüphane olarak tasarlanmıştır [6]. Bu kütüphane, bir başlangıç yörüngesinin kalitesini tekrarlı olarak iyileştirmek için işlevsel gradyan adı verilen teknikleri kullanır. STOMP ise, daha düşük maliyetle oluşturulmuş bir yörünge oluşturmak için birleştirilen ilk (muhtemelen mümkün olmayan) bir yörünge etrafındaki alanı keşfetmek için, gürültü eklenmiş yörüngeler oluşturmaya dayanan bir teknik kullanır. Her tekrarda engel ve pürüzsüzlük maliyetinin bir kombinasyonunu temel alan bir maliyet işlevi optimize edilerek çalışır [7]. Bu çalışmada OMPL algoritmaları üzerinde çalışılmıştır.

## IV.OMPL HAREKET PLANLAMA ALGORİTMALARI

OMPL (Open Motion Planning Library), birçok son teknoloji planlama algoritmasının uygulamalarını içeren, örneklemeye dayalı hareket planlaması için geliştirilmiş bir yazılım kütüphanesidir [5]. Bu kütüphane, kullanıcının çeşitli karmaşık hareket planlama problemlerini minimum girdi ile kolayca çözmesine olanak tanıyacak şekilde tasarlanmıştır. OMPL, yeni hareket planlama algoritmalarının eklenmesini kolaylaştırmakta ve diğer yazılım bileşenleriyle uygun bir şekilde arayüzlenebilmektedir.

OMPL, bünyesinde RRT (Rapidly-exploring Random Trees) [8], RRT-Connect (Rapidly-exploring Random Trees - Connect) [9], RRT* (Rapidly-exploring Random Trees - Star) [10], PRM (Probabilistic Road Map) [11], PRM* (Probabilistic Road Map - Star) [12], EST (Expansive Space Trees) [13], KPIECE (Kinodynamic Motion Planning by Interior-Exterior Cell Exploration) [14] ve BKPIECE (Bidirectional implementation of KPIECE) [14] gibi birçok algoritma bulundurmaktadır. Bu çalışmada verilen örneklerdeki algoritmalardan RRT, RRT*, RRTConnect, PRM, PRM*, EST (Expansive Space Trees) ve BiEST (Bidirectional Expansive Space Trees) algoritmaları kullanılmıştır. Şekil 4'te bu algoritmalardan bazılarının kompleksitesi ile alakalı bir tablo verilmiştir.

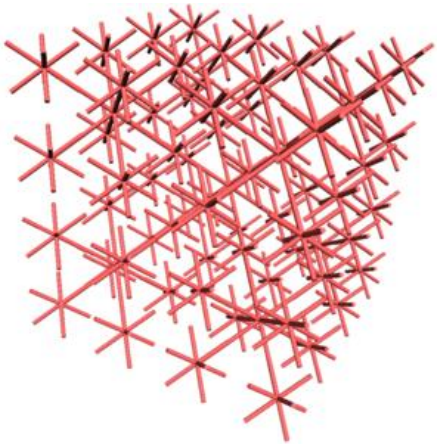
| | Algorithm | Probabilistic Completeness | Asymptotic Optimality | Monotone Convergence | Time Complexity | | Space Complexity |
|---|---|---|---|---|---|---|---|
| | | | | | Processing | Query | |
| Existing Algorithms | PRM | Yes | No | Yes | $O(n \log n)$ | $O(n \log n)$ | $O(n)$ |
| | sPRM | Yes | Yes | Yes | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ |
| | $k$-sPRM | Conditional | No | No | $O(n \log n)$ | $O(n \log n)$ | $O(n)$ |
| | RRT | Yes | No | Yes | $O(n \log n)$ | $O(n)$ | $O(n)$ |
| Proposed Algorithms | PRM* | Yes | Yes | No | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| | $k$-PRM* | | | | | | |
| | RRG | Yes | Yes | Yes | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ |
| | $k$-RRG | | | | | | |
| | RRT* | Yes | Yes | Yes | $O(n \log n)$ | $O(n)$ | $O(n)$ |
| | $k$-RRT* | | | | | | |

**Şekil. 4.** OMPL algoritmaları

Aşağıda, bu çalışmada teste tabi tutulan algoritmaların temel özellikleri ve diğer algoritmalardan benzerlik ve farklılıkları incelenmiştir.

### *A. RRT Algoritmaları (RRT, RRT-Connect, RRT*)*

RRT algoritması LaValle ve Kuffner'in çalışmalarında [29,30] ortaya çıkarılmış bir yörünge planlama algoritmasıdır. Boşluk dolduran bir ağaç yapısını (space-filling tree) [31] rastgele oluşturarak, konveks olmayan ve yüksek boyutlu olan alanlar içerisinden, verimli bir yörünge aramak için tasarlanmış bir algoritmadır (Şekil 6). Bu ağaç yapısı, arama alanından rastgele alınan örneklerden aşamalı olarak oluşturulur ve doğası gereği, uygun yörüngeyi bulabilmek için algoritmanın incelemediği geniş alanlara doğru büyüyerek tarama yapmaya meyillidir. Engeller ve diferansiyel kısıtlamalarla (holonomik olmayan ve kinodinamik) sorunları kolayca çözmekte ve otonom robotik hareket planlamasında yaygın olarak kullanılmaktadır.



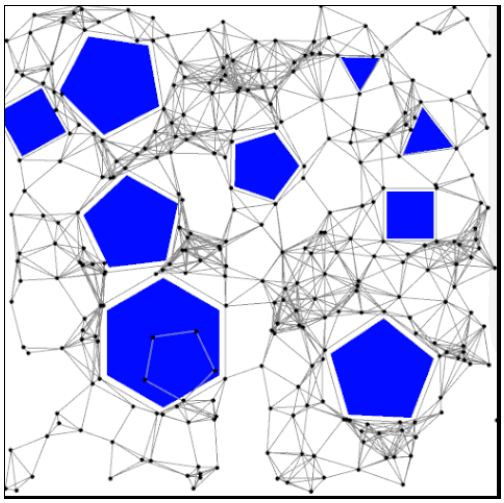
**Şekil. 5.** Küp şeklinde örnek bir "space-filling tree" yapısı

Probabilistik yol haritası (probabilistic roadmap) methodundan esinlenilerek oluşturulmuş bir sistem olan RRT'nin, en belirgin avantajı, holonomik olmayan ve kinodinamik planlamaya doğrudan uygulanabilmeleridir. Bu avantaj, RRT'lerin konfigürasyon çiftleri (veya durumlar) arasında herhangi bir bağlantı yapılmasını gerektirmemesinden, olasılıklı yol haritalarının ise tipik olarak on binlerce bağlantı gerektirmesinden kaynaklanmaktadır. Özetle RRT'ler, holonomik yörünge planlaması için temel bir olasılıklı yol haritasından daha verimli bir algoritma özelliği taşımaktadır [29,30].

Daha sonraki yıllarda RRT algoritmasının bazı eksikliklerini (planlama hızı ya da uygulama hızı sorunları gibi) gidermek için, RRT algoritmasından türetilmiş yeni algoritmaları ortaya çıkarılmıştır. Bunlardan kısaca bahsetmek gerekirse:

**RRT-Connect:** Yine Kuffner ve LaValle tarafından oluşturulmuş, RRT'nin hareket planlama menzilini arttırmak için geliştirilmiş RRT algoritmasıdır [9].

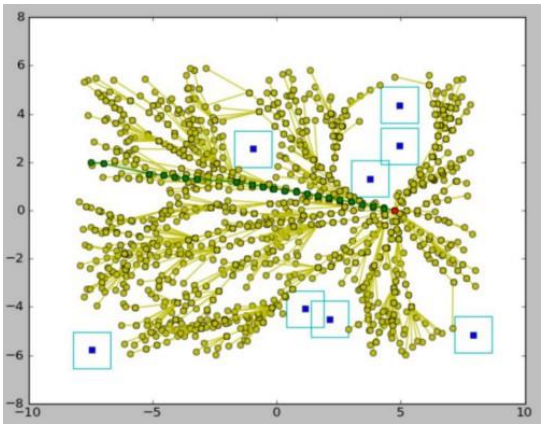
**RRT*:** RRT*, çalışılan düğüm sayısının sonsuza yaklaştığı durumlarda, hedefe ulaşmak için mümkün olan en kısa yolu verecek şekilde revize edilmiş bir RRT algoritmasıdır. Karaman ve ekibi tarafından oluşturulmuş olan bu algoritma [10, 32], gerçekçi olarak sonsuza yakın düğüm ihtimali mümkün olmasa da algoritmanın en kısa yolu geliştirmek için çalıştığını öne sürmektedir. RRT*'ın temel prensibi RRT ile aynıdır, ancak algoritmaya yapılan iki önemli ekleme önemli ölçüde farklı sonuçlar elde edilmesini sağlar. Bu farklılıklar, RRT*'ın her bir tepe noktasının ana tepe noktasına göre kat ettiği mesafeyi kaydetme ve bunun üzerinde çalışma yapacak şekilde tanımlanması ve boşluk dolduran ağacın yeniden bağlanması ve yapılandırılmasının gerçekleştirilmesidir [10, 29, 30, 32] (Şekil 6).



**Şekil. 6.** RRT* yörünge planlaması

### *B. PRM Algoritmaları (PRM, PRM*)*

Kavraki ve ekibi tarafından ortaya çıkarılan PRM yörünge planlama algoritması [11] çarpışmalardan kaçınırken robotun bir başlangıç konfigürasyonu ile bir hedef konfigürasyonu arasında bir yol belirleme sorununu çözen robotikte bir hareket planlama algoritmasıdır. PRM algoritmasının arkasındaki temel fikir, robotun hareket edeceği sahadan rastgele örnek noktalar alarak bu noktaların boş alanda kalıp kalmadıklarını sınamak ve bu noktaları yakındaki diğer noktalara bağlamak için yerel bir planlayıcıdan istifade etmeye dayanır (Şekil 7).




**Şekil. 7.** Bir dizi poligonal engel etrafında uygun yolları araştıran bir PRM algoritması örneği.

PRM algoritması çalışırken, başlangıç ve hedef noktaları eklenir ve bu noktalar arasında bir yol belirlemek için ortaya çıkan grafiğin üzerinde bir de grafik arama algoritması uygulanır. Bu çift planlama ile optimal bir planlama algoritması oluşturulmaya çalışılmıştır.

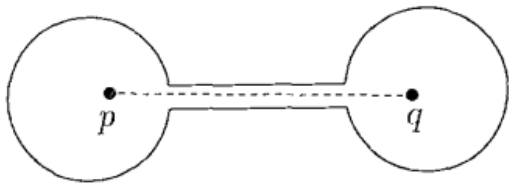
PRM algoritmasının planlama işlemi iki aşamadan oluşur: Oluşturma aşaması ve Sorgulama aşaması. Oluşturma

aşamasında, çevrede yapılması mümkün olan hareketlere ya da yönelimlere yakın bir yol haritası (grafik) oluşturulur. Bu grafik önce rastgele bir konfigürasyon temelinde meydana gelir. Daha sonra genelde ya en yakın komşulara ya da önceden belirlenmiş bir mesafeden daha az olan tüm komşulara bağlanılır. Yol haritası yeterince yoğun olana kadar grafiğe konfigürasyonlar ve bağlantılar eklenir. Sorgulama aşamasında ise, başlangıç ve hedef konfigürasyonları grafiğe bağlanır ve yol Dijkstra'nın en kısa yol sorgusu [33] ile elde edilir.

PRM'nin avantajı genel ve neredeyse her tür holonomik robota uygulanabilir bir algoritma yapısına sahip olmasıdır. Ayrıca, kolaylıkla özelleştirilebilir. Bu özelleştirme, yerel planlayıcı gibi yöntemin genel yapılarının, dikkate alınan ortam özelliklerine daha iyi uyan daha spesifik yapılarla değiştirilmesi ile mümkündür [11]. Özelleştirilmiş uygulama, çok sayıda serbestlik derecesi içeren robot için çok zor yörünge planlama sorgularını, birkaç dakikalık bir öğrenme süresinden sonra bir saniyeden kısa bir sürede çözebilmektedir.

### *C. EST Algoritmaları*

EST kütüphanesi ile geniş hareket alanlarının analizi yapan, yeni bir dinamik planlama algoritmaları tasarlanmıştır (Şekil 8). Bu algoritmalar, konfigürasyon alanının sadece mevcut sorgu ile ilgili kısmını örneklemeye çalışır ve tüm konfigürasyon alanı için bir yol haritasını önceden hesaplamanın maliyetini ortadan kaldırır. Bu nedenle, belirli bir ortam için tek bir sorgunun gönderildiği problemler için çok uygundur [34].




**Şekil. 8.** Dar bir geçidin varlığı nedeniyle rastgele örnekleme yoluyla bağlanabilirliği yakalanması zor olan bir konfigürasyon alanı

## V. HAREKET PLANLAYICILARDA PERFORMANS KARŞILAŞTIRMASI

Bu kısımda, SRVT sisteminin bünyesinde bulunan hareket planlayıcılarının içerisinde yer alan OMPL ve EST planlayıcı algoritmalarının performanslarının karşılaştırılması ve teknik bazı özellikler açıklanmıştır.

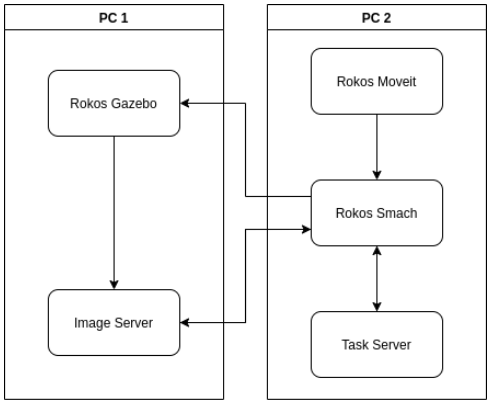
### *A. Test Donanımlarının Özellikleri*

ROKOS sisteminin simule edildiği SRVT sistemi, yüksek performanslı işlemci gücü gerektiren bir görev planlama, uygulama ve simule etme sistemi olduğundan, çalışmalar sırasında bu sistem iki ayrı PC'den kullanıldı. Tablo 1'de bu iki PC'nin donanım özellikleri verilmiştir.

**Tablo I.** Test Donanımlarının Özellikleri

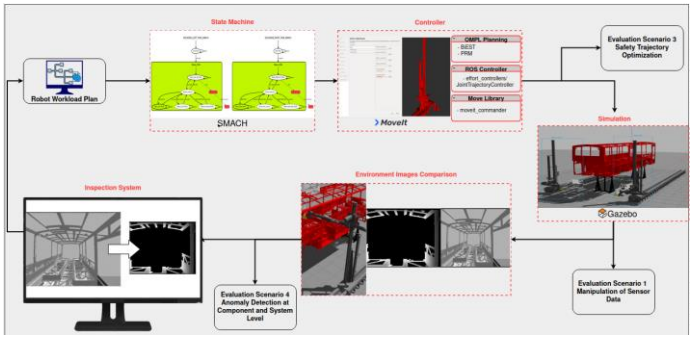
| Özellikler | PC1 | PC2 |
|---|---|---|
| Platform | Linux x86_64 | Linux x86_64 |
| Platform Dist. | Ubuntu 20.04.02 LTS | Ubuntu 20.04.02 LTS |
| Kernel Release | 5.8.0-50-generix | 5.8.0-50-generix |
| ROS Dist. | ROS Noetic | ROS Noetic |
| CPU Model | Intel® Core(TM) i7-7700HQ CPU | Intel® Core(TM) i7-8750H CPU |
| CPU Core | 8 | 12 |
| CPU Speed | 2.80 GHz | 2.20 GHz |
| RAM | 16 GB | 8 GB |
| Graphic Card | NVIDIA GeForce GTX 1050 | NVIDIA GeForce GTX 1050 |

Yukarıda donanım özellikleri verilen PC1, SRVT sisteminin simulasyon ortamının çalıştırıldığı donanımken, PC2 bu simulasyon ortamındaki görev atamalarını ve planlamalarını gerçekleştiren donanımdır. Bu iki donanımda çalışan ROS düğümlerinin oluşturduğu sistem mimari diyagramı Şekil 9'da gösterilmiştir. Sistemin genel çalışma diyagramı da Şekil 10'da gösterildiği gibidir.




**Şekil. 9.** SRVT ROKOS entegrasyonun ROS sistem mimarisi

SRVT sistemi, Bölüm 2'de anlatıldığı gibi iki kartezyen robotun aynı anda çalıştığı, belirli görevleri icra ettiği bir işlemdir. Bu işlem süreci, iki ayrı robota, belirlenen algoritmaların ya da komutların uygulandığı iki ayrı kod sisteminin uyumlu çalışması ile mümkün olmaktadır. Şekil 10'da verilen SRVT çalışma diyagramında, bu iki ayrı robot kolu için işletilen SMACH yapısı görülebilir.




**Şekil. 10.** SRVT çalışma diyagramı

### *B. Görev Listesinin Oluşturulması*

ROKOS sistemi, kendisine gönderilen görev listelerinde belirtilen koordinatlara, verilen eksen hareket sıralarına uyacak şekilde hareketler ile intikal etme üzerine tasarlanmıştır. Bu görev listeleri, SRVT sisteminin ROS paketlerinde ROKOS kollarına ait konfigürasyon dosyaları içerisinde yer alan parametre klasörlerinde saklanmaktadır. Tablo 2'de bu parametre dosyalarından birinin içeriği görülebilir.

**Tablo II.** Rokos sol kol görev parametresi içerik kesiti

```
---

Left_Rokos:

  - Task:
      Task_ID: 35694  %Görev Numarası

      Vehicle_CODE: '13M38_1' %Şase Kodu

      Mode: 1 %Kamera Kayıt Modu

      Tag: 'ON' %Şaseye Göre Görev Konumu

      Position:  %Robot Kol Koordinatları
        X: 6.079
        Y: 2.840
        Z: 1.603
        C_AX: 10.000
        C_AZ: 170.000

      Queue:  %Hareketin Uygulanma Sırası
        X: 1
        Y: 3
        Z: 2
        C_AX: 4
        C_AZ: 5
...
```

Tablo 2'de bir kesiti yer alan .yaml uzantılı görev listesi dosyası, ROKOS kollarının ROS paketleri çalıştığında sistem tarafından istenir, bu verilere uygun olarak robot kolları sırasıyla görevleri gerçekleştirir. Kesitte yorum satırı olarak vurgulanmış detaylara göre sınıflandırılmış görevler, SRVT sistemi tarafından uygulanmakta, sistem çıktılarının kayıtları bu detaylara göre tutulmakta, robot kol hareketleri ilgili koordinatlar ve bu koordinatlara intikal için gerekli yörünge hareket sıralamalarına uygun hareketler yaparak görevleri tamamlamaktadır.

**Tablo III.** Gerçek ortamdaki görev koordinatlar listesi

| **Y** | **35269** | **35270** |
|---|---|---|
| **Araç_Kodu** | 13M38_1 | 13M38_1 |
| **Robot_NO** | 1 | 1 |
| **Kamera_X** | 1750 | 1150 |
| **Sira_X** | 3 | 3 |
| **Kamera_Y** | 1082 | 1682 |
| **Sira_Y** | 2 | 2 |
| **Kamera_Z** | 50 | 50 |
| **Sira_Z** | 1 | 1 |
| **Kamera_AX** | 10 | 10 |
| **Sira_AX** | 4 | 4 |
| **Kamera_AZ** | 0 | 0 |
| **Sira_AZ** | 5 | 5 |
| **Durum** | 1 | 1 |
| **Created_at** | 2020-10-18 2:09:57.000 | 2020-10-18 02:09:57.000 |
| **Etiket** | ARKA | ARKA |

Tablo 3'te SRVT ortamına göre uyarlanmış ROKOS görev bilgilerinin bulunduğu dosya içeriğinden bir kesit görülmektedir. Yeni bir görev oluşturulurken ya da görevlerde değişiklik yapıldığında bu değişikliklerin ".yaml" uzantılı görev dosyalarına çevrilmesi için bir ROS düğümü geliştirilmiştir. Bu düğüm, hazırlanan .csv uzantılı görev tablolarını otomatik olarak .yaml uzantılı konfigürasyon dosyalarına çevirmektedir. Bu şekilde ROKOS sistemi için görev dosyaları oluşturulmuş olur.

ROKOS sistemi için tasarlanmış görev listelerinde, robotun yörünge planı olmaksızın engellere çarpmadan hareketi şeklinde bir çalışma mantalitesi benimsendiğinden, belirli konumlar arasına yerleştirilmiş reset noktaları bulunmaktadır. Bu noktalar sayesinde robot, intikal etmesi gereken iki nokta arasında ilerlerken önce belirlenen reset konumlarına ulaştığından engellere çarpmadan hareketler gerçekleştirebilmektedir. Toplamda iki ROKOS kolu için 330 görev noktası bulunmaktadır ve bu noktaların 58'i reset noktalarıdır. ROKOS sisteminde ise çarpışmasız dinamik yörünge planlama olduğu için bu reset noktalarının kullanılmasına gerek kalmamaktadır. Testler gerçekleştirilirken hem eski görev listesi hem de reset noktaları çıkarılmış yeni görev listesi kullanılarak süreler elde edilmiştir.

### *C. Test Yöntemlerinin Belirlenmesi*

Hızlı test, görev listesinden rastgele seçilen 15'er görev ile oluşturulan kısa görev listesini tamamlama sürelerinin hesaplanmasını içerir. Belirlenen noktalar, tam görev listesinin içerisinden, ilerleme sırasına göre rastgele

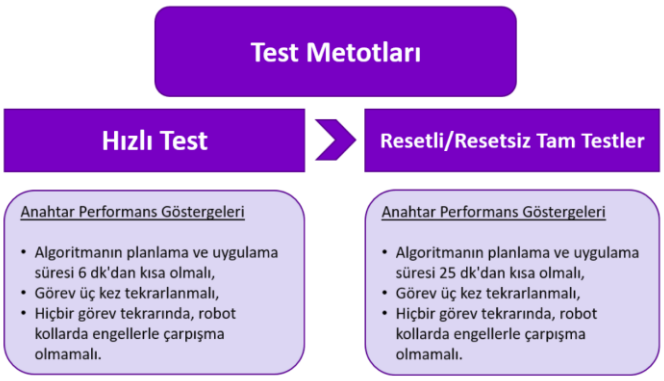
seçilmiştir. Bu da robotun tüm görevleri yapmasa da hareket güzergahı boyunca hareket etmesini sağlaması, testin robot kollarının menzillerini de hesaba katabilmesi için belirlenmiştir. Resetli Tam Test, gerçek ortamdaki ROKOS sistemi için oluşturulmuş, görev sırasında çarpışmaları engellemek için bazı reset noktaları belirlenmiş 330 görev koordinatından oluşan listeye uygulanan testtir. Resetsiz Tam Test ise bu reset noktaları çıkarılarak, görev listesinin sayısının 272 noktaya indirilmesiyle oluşturulmuş yeni listeye uygulanan testtir. Bu testler sonucunda elde edilecek süreler her iki kol aynı anda çalışırken, ayrı ayrı hesaplanır ve verilen görev listesini üçer defa tamamlanması ile elde edilen veriler üzerinden gerçekleştirilmiştir. Testlerin başarı kriterleri Şekil 10'daki diyagramda verilmiştir. Ayrıca Tablo 4'te bu görev listelerinin dosya yapısı gösterilmiştir.

**Tablo IV.** ROKOS görev listeleri dosya yapısı

```
---
Rokos_Task:
  # None = Full Task

  Right_Rokos: None
    #- 35272
    #- 35277
    #- 35279
    #- 35291
    #- 35310
    #- 35320
    #- 35330
    #- 35340
    #- 35349
    #- 35360
    #- 35374
    #- 35380
    #- 35390
    #- 35414
    #- 35432

  Left_Rokos: None
    #- 35702
    #- 35705
    #- 35700
    #- 35710
    #- 35730
    #- 35749
    #- 35760
    #- 35770
    #- 35780
    #- 35790
    #- 35818
    #- 35825
    #- 35830
    #- 35840
    #- 35875
```




**Şekil. 11.** Test yöntemleri ve anahtar performans göstergeleri diyagramı

Şekil 11'de verilen test yöntemlerindeki başarı kriterleri, "Güvenli Yörünge Optimizasyonu" değerlendirme senaryosu kapsamında yer alan kriterlere göre oluşturulmuştur.

Testler sol ve sağ ROKOS kolları için ayrı ayrı süreler alınmak suretiyle uygulanmaktadır. Bunun sebebi her iki robot kolu için planlama algoritmalarının ayrı ayrı çalıştırılıyor olmasıdır. Ayrıca iki kol için de anlık görev koordinatları ve yörüngeler farklılık göstermektedir. Bu farklılıklarla birlikte aslında tek bir robotik sistemde iki farklı robot kolunun ayrı ayrı planlama ve planı uygulama sürecinden geçirilmesi durumunu oluşturur. Bu nedenle testler esnasında her kol için algoritma performansları ayrı ayrı değerlendirilmekte ve bu sonuçların ortalamaları alınarak sonuçlar analiz edilmektedir.

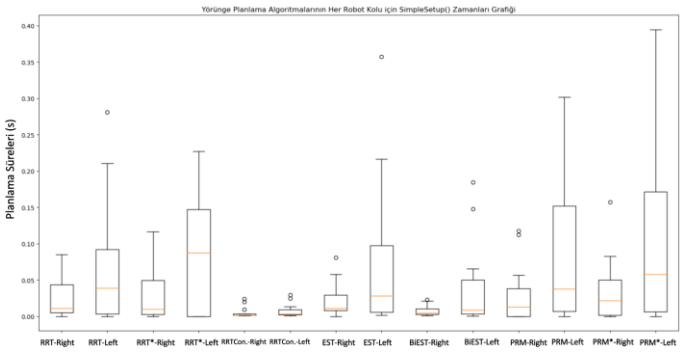
### *D. Algoritma Testleri*

Yörünge planlama algoritmaları için belirlenen test yöntemlerinin uygulamasından önce, bu algoritmaların planlama ve planı uygulama çözümlerini bulma sürelerinin de testleri gerçekleştirilmiştir (Planlama Kurulum ve Çözüm Bulma değerleri). Algoritmaların bu zaman değerlerinin ortalama ve standart sapma değerleri, Tablo 5'te gösterilmiştir.

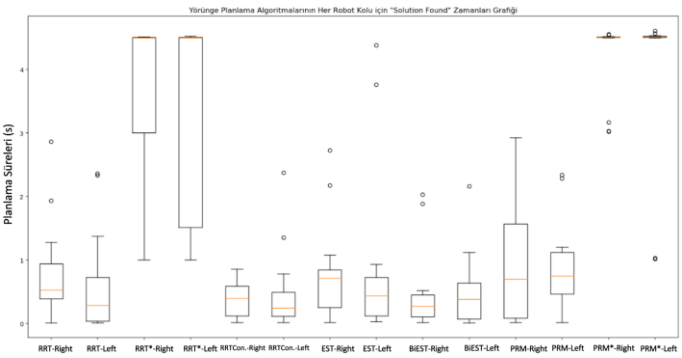
**Tablo V.** OMPL Algoritmaları Uygulanan Rokos Kollarının PlanlamaKurulum (SimpleSetup) ve ÇözümBulma (SolutionFound) Zamanları Tablosu

| Planlayıcılar&Rokos Kolları | Simple Setup Ortalama (sn) | Simple Setup Standart Sapma (sn) | Solution Found Ortalama (sn) | Solution Found Standart Sapma (sn) |
|---|---|---|---|---|
| **RRT-Sol** | 0.027837 | 0.030814 | 0.829886 | 0.788524 |
| **RRT-Sağ** | 0.070042 | 0.083717 | 0.676665 | 0.844881 |
| **RRT*-Sol** | 0.027837 | 0.030814 | 0.829886 | 0.788524 |
| **RRT*-Sağ** | 0.095220 | 0.09871 | 3.371867 | 1.564152 |
| **RRTCon.-Sol** | 0.005815 | 0.007541 | 0.399085 | 0.282422 |
| **RRTCon-Sağ** | 0.008482 | 0.008846 | 0.504321 | 0.640946 |
| **EST-Sol** | 0.025089 | 0.024726 | 0.817884 | 0.761156 |
| **EST-Sağ** | 0.080437 | 0.101916 | 0.905613 | 1.332297 |
| **BiEST-Sol** | 0.008851 | 0.007632 | 0.527579 | 0.648691 |
| **BiEST-Sağ** | 0.041339 | 0.056627 | 0.531412 | 0.579818 |
| **PRM-Sol** | 0.031796 | 0.040521 | 1.074248 | 1.030772 |
| **PRM-Sağ** | 0.100877 | 0.104994 | 0.921887 | 0.756369 |
| **PRM*-Sol** | 0.037562 | 0.042537 | 4.229041 | 0.599214 |
| **PRM*-Sağ** | 0.115702 | 0.121598 | 4.059866 | 1.233106 |

Yukarıda verilen tabloda yer alan SimpleSetup zamanları, algoritmaların planlamayı gerçekleştirdiği süreleri vermektedir. SolutionFound süreleri ise bu planları, robot kollarına uygulattıkları süredir. Bu süreler, robot kollarının bir görev boyunca yaptıkları tüm hareket planlarından elde edilen zamanların analizi ile hesaplanmıştır. Bu sürelerle elde edilen süre grafikleri de Şekil 12 ve Şekil 13'te verildiği gibidir.




**Şekil. 12.** Yörünge planlama algoritmalarının her robot kolu için SimpleSetup zamanları grafiği



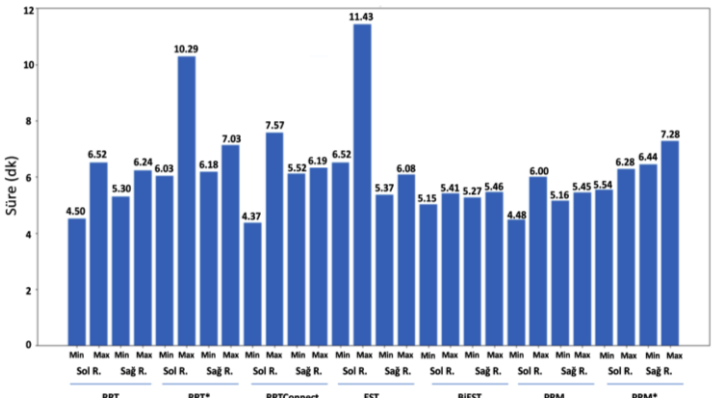

**Şekil. 13.** Yörünge planlama algoritmalarının her robot kolu için SolutionFound zamanları grafiği

### *E. Test Sonuçları*

Hızlı test sonucunda elde edilen veriler, Tablo 6 ve Şekil 14'te gösterilmiştir.

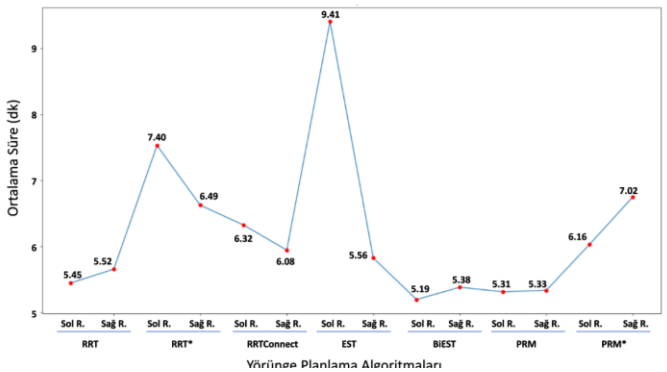
**Tablo VI.** OMPL Algoritmaları Uygulanan Rokos Kollarının Hızlı Test Tamamlama Süreleri ve Ortalaması

| Planlayıcılar&Rokos Kolları | Tur 1 | Tur 2 | Tur 3 | Ortalama Süre |
|---|---|---|---|---|
| **RRT-Sol** | 06:52.97 | 04:50.54 | 05:31.53 | 05:45.01 |
| **RRT-Sağ** | 06:24.42 | 05:42.52 | 05:30.48 | 05:52.47 |
| **RRT*-Sol** | 10:29.36 | 06:03.42 | 06:27.28 | 07:40.05 |
| **RRT*-Sağ** | 07:13.12 | 06:56.35 | 06:18.12 | 06:49.20 |
| **RRTCon.-Sol** | 07:03.23 | 04:37.18 | 07:57.60 | 06:32.67 |
| **RRTCon-Sağ** | 06:19.93 | 06:12.19 | 05:52.78 | 06:08.30 |
| **EST-Sol** | 11:43.22 | 06:52.42 | 10:28.21 | 09:41.28 |
| **EST-Sağ** | 05:37.37 | 06:02.84 | 06:08.21 | 05:56.14 |
| **BiEST-Sol** | 05:15.27 | 05:41.59 | 05:02.73 | 05:19.86 |
| **BiEST-Sağ** | 05:41.93 | 05:46.64 | 05:27.91 | 05:38.83 |
| **PRM-Sol** | 06:00.56 | 04:48.77 | 05:45.50 | 05:31.61 |
| **PRM-Sağ** | 05:16.33 | 05:45.56 | 05:39.12 | 05:33.67 |
| **PRM*-Sol** | 05:54.98 | 06:27.28 | 06:28.12 | 06:16.79 |
| **PRM*-Sağ** | 07:28.44 | 06:53.26 | 06:44.68 | 07:02.13 |




**Şekil. 14.** Hızlı Test Sonuçları (Robot kolları bazında algoritmaların en iyi ve en kötü süreleri)

Şekil 14'teki grafikte, hızlı test sonucunda her algoritmanın her robot kolundaki en iyi ve en kötü görev tamamlama süreleri verilmiştir.

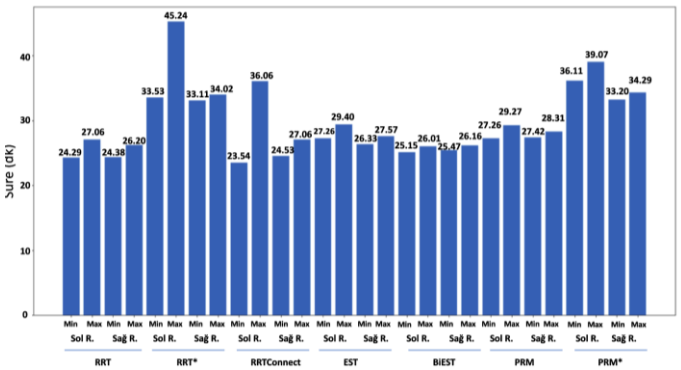



**Şekil. 15.** Hızlı Test Sonuçları (Robot kolları bazında algoritmaların ortalama süreleri)

Şekil 15'deki grafikte ise hızlı test sonucundaki her algoritmanın her robot kolundaki ortalama görev tamamlama süreleri verilmiştir. Bu tablodan ve grafiklerden de anlaşılacağı üzere RRT, PRM ve BiEST algoritmaları, hızlı test sonucunda iyi süreler çıkarmıştır.

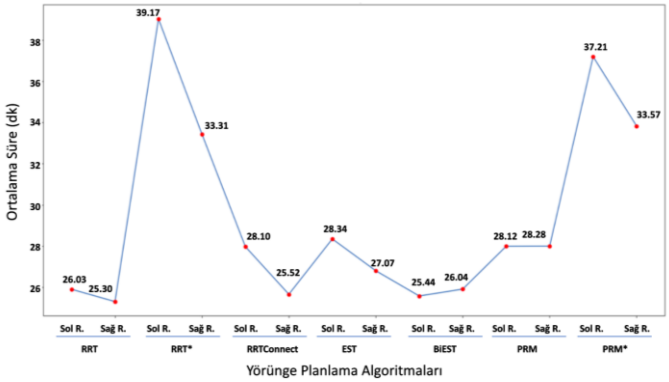
**Tablo VII.** OMPL Algoritmaları Uygulanan Rokos Kollarının Tam Test Tamamlama Süreleri ve Ortalaması

| Planlayıcılar&Rokos Kolları | Tur 1 | Tur 2 | Tur 3 | Ortalama Süre |
|---|---|---|---|---|
| **RRT-Sol** | 26:34.13 | 27:06.65 | 24:29.95 | 26:03.58 |
| **RRT-Sağ** | 26:20.79 | 25:31.29 | 24:38.43 | 25:30.17 |
| **RRT*-Sol** | 45:24.15 | 38:35.71 | 33:53.49 | 39:17.78 |
| **RRT*-Sağ** | 34:02.28 | 33:11.62 | 33:19.71 | 33:31.20 |
| **RRTCon.-Sol** | 23:54.40 | 36:06.81 | 24:30.71 | 28:10.64 |
| **RRTCon-Sağ** | 24:53.51 | 27:06.96 | 25:37.17 | 25:52.55 |
| **EST-Sol** | 27:26.21 | 29:40.67 | 28:37.24 | 28:34.71 |
| **EST-Sağ** | 27:57.94 | 26:49.73 | 26:33.45 | 27:07.04 |
| **BiEST-Sol** | 26:01.29 | 25:15.89 | 25:55.85 | 25:44.34 |
| **BiEST-Sağ** | 26:08.30 | 25:47.53 | 26:16.96 | 26:04.26 |
| **PRM-Sol** | 29:27.44 | 27:26.18 | 27:43.53 | 28:12.38 |
| **PRM-Sağ** | 28:25.82 | 28:31.74 | 27.42.99 | 28:28.78 |
| **PRM*-Sol** | 36:11.85 | 36:43.32 | 39:07.91 | 37:21.03 |
| **PRM*-Sağ** | 34:00.81 | 34:29.87 | 33:20.94 | 33:57.21 |




**Şekil. 16.** Tam Test Sonuçları (Robot kolları bazında algoritmaların en iyi ve en kötü süreleri)

Şekil 16'daki grafikte, tam test sonucunda her algoritmanın her robot kolundaki en iyi ve en kötü görev tamamlama süreleri verilmiştir.




**Şekil. 17.** Tam Test Sonuçları (Robot kolları bazında algoritmaların ortalama süreleri)

Şekil 17'deki grafikte ise tam test sonucundaki her algoritmanın her robot kolundaki ortalama görev tamamlama süreleri verilmiştir. Verilen tablo ve grafiklerde yer alan verilere göre BiEST ve RRT algoritmaları, anahtar performans göstergelerine göre tam testlerde de yeterli bir performans göstermiştir. Hızlı testlerde yeterli performans gösteren PRM algoritması tam testlerde aynı performansı gösterememiştir. Ortalamada yeterli bir performans gösterse de RRT algoritmasının sol ROKOS kolu için en kötü görev turu tamamlama süresine sahip olması nedeniyle çok da stabil bir çalışma göstermediği görülmektedir. Bu anlamda BiEST algoritması oldukça stabil görev tamamlama süreleri ortaya koymuştur.

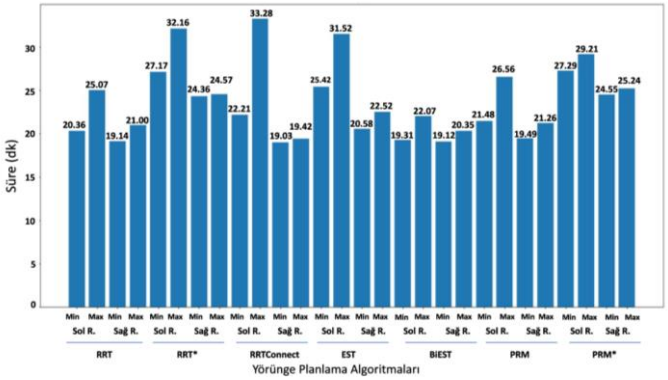
Gerçekleştirilen tam testin görev listesi Bölüm V.B'de bahsi geçen reset noktalarına sahip olan eski görev listesi üzerinde gerçekleştirilmiştir. Testin son bölümünde, reset noktaları çıkarılarak 330'dan 272 noktaya indirilmiş yeni

görev listeleri üzerinde (sağ ve sol ROKOS görev listeleri) tam testin tekrar edilmesi gerçekleştirilmiştir. Tablo 8'de güncel görev listesi üzerinde yapılan tam testin sonuçları ve önceki listeye nazaran elde edilen süre kazancı yüzdeleri görülebilir.

**Tablo VIII.** OMPL Algoritmaları Uygulanan Rokos Kollarının Güncel Görev Listesinde Tam Test Tamamlama Süreleri, Ortalaması ve Eski Görev Listesi Sürelerinden Kazanç Oranları

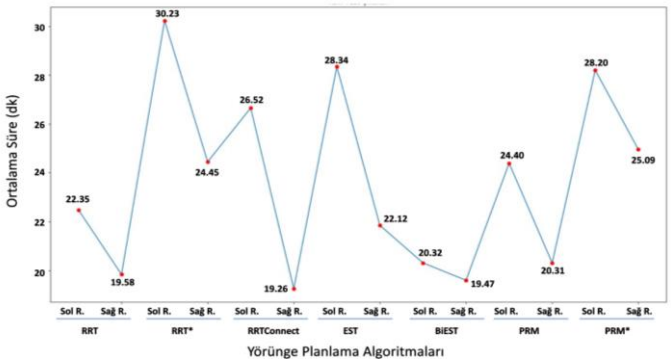
| Planlayıcılar &Rokos Kolları | Tur 1 | Tur 2 | Tur 3 | Ortalama Süre | Süre Kazancı |
|---|---|---|---|---|---|
| **RRT-Sol** | 20:36.31 | 22:03.76 | 25:07.26 | 22:35.78 | 13.29% |
| **RRT-Sağ** | 19:14.61 | 21:00.89 | 19:38.84 | 19:58.11 | 21.7% |
| **RRT*-Sol** | 31:35.66 | 27:17.49 | 32:16.33 | 30:23.16 | 22.67% |
| **RRT*-Sağ** | 24:42.85 | 24:57.86 | 24:36.64 | 24:45.78 | 26.12% |
| **RRTCon.-Sol** | 24:47.58 | 22:21.49 | 33:28.19 | 26:52.42 | 4.63% |
| **RRTCon-Sağ** | 19:42.34 | 19:33.87 | 19:03.21 | 19:26.47 | 24.87% |
| **EST-Sol** | 28:08.15 | 25.42.31 | 31.52.57 | 28:34.34 | 0.02% |
| **EST-Sağ** | 20:58.26 | 22:52.11 | 22:45.94 | 22:12.10 | 18.13% |
| **BiEST-Sol** | 22:07.90 | 19:31.22 | 19:57.43 | 20:32.18 | 20.21% |
| **BiEST-Sağ** | 20:35.92 | 19:12.75 | 19:34.17 | 19:47.61 | 24.08% |
| **PRM-Sol** | 21:48.94 | 26:56.35 | 25:14.94 | 24:40.08 | 12.54% |
| **PRM-Sağ** | 19:49.56 | 21:26.52 | 20:19.86 | 20:31.98 | 27.9% |
| **PRM*-Sol** | 29:21.22 | 27:29.25 | 28:10.61 | 28:20.36 | 24.13% |
| **PRM*-Sağ** | 25:08.47 | 25:24.70 | 24:55.36 | 25:09.51 | 25.9% |

Şekil 18'deki grafikte, tam test sonucunda her algoritmanın her robot kolundaki en iyi ve en kötü görev tamamlama süreleri verilmiştir.




**Şekil. 18.** Güncel Görev Listesine Uygulanan Tam Test Sonuçları (Robot kolları bazında algoritmaların en iyi ve en kötü süreleri)

Şekil 19'daki grafikte ise tam test sonucundaki her algoritmanın her robot kolundaki ortalama görev tamamlama süreleri verilmiştir.




**Şekil. 19.** Güncel Görev Listesine Uygulanan Tam Test Sonuçları (Robot kolları bazında algoritmaların ortalama süreleri)

Verilen tablo ve grafiklerde yer alan verilere göre güncel görev listesinde de BiEST ve RRT algoritmaları, anahtar performans göstergelerine bakıldığında başarılı bir performans göstermiştir. Görev listelerinden planlayıcı için gereksiz olabilecek noktalar çıkarıldığında görev tamamlama yüzdelerinin %27.9'a kadar arttığı gözlemlenmiştir. Ayrıca görev tamamlama sürelerinin 20 dakikanın altına indirilebildiği de görülmüştür. Buradan yola çıkılarak ROKOS sistemine uygulanan dinamik planlama sisteminin, az konum noktasında daha etkili çalıştığı sonucuna ulaşılmıştır.

Bu testlerle birlikte, ROKOS kolları çok tekrarlı bir şekilde görev listesinin ya bir kısmıyla ya da tamamıyla sınanmıştır. Hızlı test ile diğer yandan alternatif görev listelerinde robotların vereceği tepkiler de gözlenmiştir.

Bu çalışmalar kapsamında, OMPL planlayıcısına ait birçok hareket planlama algoritması üzerinde 400'ü aşkın, saatlerce süren testler gerçekleştirilmiş, en optimal planlama algoritması bulunmaya çalışılmıştır. Çalışma sonunda bu planlama algoritmasının BiEST olması gerektiği tespit edilerek testler tamamlanmıştır.

## VI. SONUÇ VE DEĞERLENDİRME

Bu çalışma ROS araçlarından ve Gazebo simulasyon ortamının entegrasyonu ile ortaya çıkarılan SRVT sistemine, ROKOS robot kol sisteminin entegrasyonu ile SRVT içerisindeki bir diğer unsur olan Moveit! yazılımı aracılığıyla kullanılan hareket planlama algoritmalarının planlama performanslarının incelenmesi ve kıyaslanması üzerine gerçekleştirilmiştir. SRVT ekosistemi içerisine entegre edilmiş ROKOS robot kol sisteminin, görev tamamlama sürelerinin düşürülmesi için, bu sisteme en uygun yörünge planlama algoritmasının belirlenmesine odaklanılmıştır. Bu uygulamalara geçilmeden önce OMPL kütüphanesi içindeki birçok temel hareket planlama algoritması incelenmiş, bu algoritmaların hareket

planlama mantalitesinin, ROKOS sisteminin SRVT entegrasyonunda çalıştırılabilirliği ve verimliliği üzerinde çalışma yapılmıştır. Araştırmalar ve yapılan testler ışığında, ROKOS sisteminin görev tamamlama sürelerinin, uygun algoritmalar tespit edildiğinde oldukça düşürülebildiği görülmüştür.

Gelecekte ROKOS'un görev tamamlama sürelerinin 25 dakikanın altında stabil hale getirilmesi için algoritma test çeşitliliğinin arttırılması, farklı görev listelerinde testler yapılması gibi optimizasyon çalışmalarına odaklanılması planlanmaktadır.

### ***BİLGİLENDİRME***[ACKNOWLEDGMENT]

The research leading to this paper has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey. The views expressed in this document are the sole responsibility of the authors and do not necessarily reflect the views or position of the European Commission.

Bu çalışma İnovasyon Mühendislik tarafından yürütülen 120N803 nolu "Otomatik Sistemlerin Emniyet ve Güvenliğinin Doğrulanması ve Geçerlenmesi" 1071 projesi kapsamında Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından desteklenmiştir.

### ***ÇIKAR ÇATIŞMASI*** *[CONFLICTS OF INTEREST]*

Yazarlar arasında ve ilgili kurumları arasında herhangi çıkar çatışması olmadığını bildirmişlerdir.

### ***ETİK KURALLARA UYGUNLUK*** *[RESEARCH AND PUBLICATION ETHICS]*

Yazarlar bu makalenin etik kurul onayı veya herhangi bir özel izin gerektirmediğini beyan ederler.

## KAYNAKLAR

1 Ragel, R., Maza, I., Caballero, F., & Ollero, A. (2015, November). Comparison of motion planning techniques for a multi-rotor UAS equipped with a multi-joint manipulator arm. In 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS) (pp. 133-141). IEEE.

2 Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

3 Gazebo website. [Online]. Available: http://gazebosim.org/, (2021)

4 Chitta, S., Sucan, I., & Cousins, S. (2012). Moveit![ros topics]. IEEE Robotics & Automation Magazine, 19(1), 18-19.

5 Sucan, I. A., Moll, M., & Kavraki, L. E. (2012). The open motion planning library. IEEE Robotics & Automation Magazine, 19(4), 72-82.

6 Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., ... & Srinivasa, S. S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. The International Journal of Robotics Research, 32(9-10), 1164-1193.

7 Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011, May). STOMP: Stochastic trajectory optimization for motion planning. In 2011 IEEE international conference on robotics and automation (pp. 4569-4574). IEEE.

8 LaValle, S. M., Kuffner, J. J., & Donald, B. R. (2001). Rapidly-exploring random trees: Progress and prospects. Algorithmic and computational robotics: new directions, 5, 293-308.

9 Kuffner, J. J., & LaValle, S. M. (2000, April). RRT-connect: An efficient approach to single-query path planning. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065) (Vol. 2, pp. 995-1001). IEEE.

10 Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. The international journal of robotics research, 30(7), 846-894.

11 Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation, 12(4), 566-580.

12 Marble, J. D., & Bekris, K. E. (2013). Asymptotically near-optimal planning with probabilistic roadmap spanners. IEEE Transactions on Robotics, 29(2), 432-444.

13 Plaku, E., Bekris, K. E., Chen, B. Y., Ladd, A. M., & Kavraki, L. E. (2005). Sampling-based roadmap of trees for parallel motion planning. IEEE Transactions on Robotics, 21(4), 597-608.

14 Sucan, I. A., & Kavraki, L. E. (2011). A sampling-based tree planner for systems with complex dynamics. IEEE Transactions on Robotics, 28(1), 116-131.

15 Görner, M., Haschke, R., Ritter, H., & Zhang, J. (2019, May). Moveit! task constructor for task-level motion planning. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 190-196). IEEE.

16 Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2012). Trajectory planning in robotics. Mathematics in Computer Science, 6(3), 269-279.

17 Sciavicco, L., Siciliano, B., Villani, L., Oriolo, G.: Robotics. Modelling, Planning and Control. Springer, London (2009)

18 Krishnaswamy, K., Sleeman, J., & Oates, T. (2011, May). Real-time path planning for a robotic arm. In Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments (pp. 1-4).

19 Roy, R., Mahadevappa, M., & Kumar, C. S. (2016). Trajectory path planning of EEG controlled robotic arm using GA. Procedia Computer Science, 84, 147-151.

20 Hao, W. G., Leck, Y. Y., & Hun, L. C. (2011, May). 6-DOF PC-Based Robotic Arm (PC-ROBOARM) with efficient

trajectory planning and speed control. In 2011 4th International Conference on Mechatronics (ICOM) (pp. 1-7). IEEE.
21 Xie, B., Zhao, J., & Liu, Y. (2011, June). Human-like motion planning for robotic arm system. In 2011 15th International Conference on Advanced Robotics (ICAR) (pp. 88-93). IEEE.
22 Iqbal, J., Islam, R. U., & Khan, H. (2012). Modeling and analysis of a 6 DOF robotic arm manipulator. Canadian Journal on Electrical and Electronics Engineering, 3(6), 300-306.
23 Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2015). Path planning and trajectory planning algorithms: A general overview. Motion and operation planning of robotic systems, 3-27.
24 Streinu, I. (2000, November). A combinatorial approach to planar non-colliding robot arm motion planning. In Proceedings 41st Annual Symposium on Foundations of Computer Science (pp. 443-453). IEEE.
25 Xinyu, W., Xiaojuan, L., Yong, G., Jiadong, S., & Rui, W. (2019). Bidirectional potential guided RRT* for motion planning. IEEE Access, 7, 95046-95057.
26 Savsani, P., Jhala, R. L., & Savsani, V. J. (2013, April). Optimized trajectory planning of a robotic arm using teaching learning based optimization (TLBO) and artificial bee colony (ABC) optimization techniques. In 2013 IEEE International Systems Conference (SysCon) (pp. 381-386). IEEE.
27 Cousins, S. (2010). Ros on the pr2 [ros topics]. IEEE Robotics & Automation Magazine, 17(3), 23-25.
28 Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., & Burgard, W. (2010, May). OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation (Vol. 2).
29 LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.
30 LaValle, S. M., & Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. The international journal of robotics research, 20(5), 378-400.
31 Sagan, H. and J. Holbrook: "Space-filling curves", Springer-Verlag, New York, 1994.
32 Karaman, S., & Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. Robotics Science and Systems VI, 104(2).
33 Sniedovich, M. (2006). Dijkstra's algorithm revisited: the dynamic programming connexion. Control and cybernetics, 35(3), 599-620.
34 Hsu, D., Latombe, J. C., & Motwani, R. (1997, April). Path planning in expansive configuration spaces. In Proceedings of International Conference on Robotics and Automation (Vol. 3, pp. 2719-2726). IEEE.
35 Coleman, D., Sucan, I., Chitta, S., & Correll, N. (2014). Reducing the barrier to entry of complex robotic software: a moveit! case study. arXiv preprint arXiv:1404.3785.
36 Moll, M., Sucan, I. A., & Kavraki, L. E. (2014). An extensible benchmarking infrastructure for motion planning algorithms. arXiv preprint arXiv:1412.6673.
37 Cohen, B., Şucan, I. A., & Chitta, S. (2012, October). A generic infrastructure for benchmarking motion planners. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 589-595). IEEE.
38 Liu, S., & Liu, P. (2021). Robot Motion Planning Benchmarking and Optimization using Motion Planning Pipeline.
39