

**İstanbul Teknik Üniversitesi Robotik Arama Kurtarma Ekibi**

**Görev Raporu**

**Görev Konusu:** ROS’u Tanıma Araştırması

**Hazırlayan:** Hasan Salih Şahin

**Teslim Tarihi:** 30.11.2024

**ROS’u Tanıma Araştırması**

**ROS Nedir**

Açılımı Robot Operating System olan robotik yazılım platformudur. Robotik uygulamalar geliştirmek ve çalıştırmak için bir **araçlar, kütüphaneler ve konvansiyonlar seti olarak da tanımlanır**.Açık kaynaklıdır.Yani kaynak kodları herkesin erişimine açıktır.Bu sayede kullanıcılar kodları inceleyebilir,değiştirebilir ve yeniden dağıtabilir.2007 yılında Willov Garage firması tarafından Stanford Üniversitesi Yapay Zeka Laboratuvarında üretilmiştir.

**Robotikte Hangi Amaçla Kullanılır**

ROS robotik sistemlerin geliştirilmesini kolaylaştırmak amacıyla tasarlanmıştır.Robotik sistemleri modüler,verimli ve işbirliği halinde kodlayabilmek amacıyla kullanılır.Mevcut kütüphane ve araçlarıyla hızlılık,düğüm tabanlı sistemiyle modüler,farklı donanımlara sürücü desteği sağlamasıyla uyumluluk gibi özellikleri onu tercih edilebilir kılar.

**Robotikte ROS kullanmanın avantajları/dezavantajları nelerdir?**

**Avantajları**

-Açık kaynaklı ve ücretsiz olması geniş bir kullanıcı topluluğuna ulaşmasını sağlar.

-Modüler yapıda olması sayesinde büyük projeler küçük parçalara ayrılabilir ve kolayca yönetilebilir,yazılımın test edilmesi kolaylaşır ve hatalar daha kolay giderilir

-Farklı donanımlarla uyumlu bir biçimde çalışabilmesi sayesinde aynı yazılım farklı platformlarda kullanılabilir.

-Birçok hazır hütüphane bulundurması sayesinde hazır çözümler kullanarak zamandan tasarruf etmemize yarayabilir.

-Simülasyon araçlarıyla entegre çalışması ucuz prototip oluşturmayı ve sanal ortamda test etmeyi sağlar.

-Geniş topluluk desteği bulunur ve sorun gidermek topluluk desteğiyle kolaylaşır.

-Birçok programlama dilini desteklemesi geliştiricilere tercih ettikleri dili kullanma imkanı tanır.

**Dezavantajları**

**-**Gerçek zamanlı bir sistem olmaması( ile kritik zamanlarda gecikmeler yaşatabilir.

-Modüler ve geniş yapısı ile yeni başlayanlar için öğrenmesi karmaşık olabilir.

-Bazı donanımlarla uyumsuz olabilir.Bu durumda cihaza özel sürücü yazılması gerekir.

-Duruma göre güçlü donanım gerektirebilir.

-Genellikle Linux tabanlı sistemlerde çalışması Linux’u çok kullanmamış kullanıcılar için zorluk çıkarabilir.

**Bir robotik projesinde ROS ve açık kaynak ROS paketi kullanımı hangi seviyede olmalıdır?**

Projenin ne olduğuna ve hedeflerine göre değişir.Araştırma ve prototipleme projeleri için idealdir çünkü hızlı geliştirmeyi ve test etmeyi sağlar ve yüksek miktarda kullanım gerektirir.Ticari ve Endüstriyel projeler için temel altyapılar için kullanılabilir fakat kritik görevler için özelleştirilmiş çözümler üretilir.Düşük-orta düzey kullanım yeterli olabilir.Eğitim projelerinde ise oldukça fazla kullanılır.

**ROS kullanımı nasıl azaltılabilir?**

-Gerçek zamanlı olmaması nedeniyle kritik görevlerde gecikmeyi önlemek amacıyla doğrudan donanımla iletişim kuran özel yazılımlar kullanılabilir.Bu özel yazılımlarla ROS’tan bağımsız hale getirilebilir.

-Daha düşük seviyeli dillerde görüntü işleme motor kontrolü gibi işlemler yapılabilir

-ROS düğümleri arasında yüksek miktarda iletişim performans sorunlarına yol açar ise ROS dışında doğrudan iletişim protokollerine başvurulabilir(TCP/IP veya UDP).

-ROS’un içerisinde bulunan bir kütüphaneyi kullanmak yerine doğrudan açık kaynaklı kütüphane kullanılabilir.

-Çok sayıda düğüm iletişim trafiğine neden olacağından birden fazla işlev aynı düğümde birleştirilerek düğüm sayısı azaltılabilir.

-Açık kaynaklı büyük bir paketin sadece kullandığımız kısımlarını içeren bir mini paket oluşturulabilir.

-Eğer proje buna uygun ve küçük bir projeyse YARP,LCM gibi alternatif çerçeveler kullanılabilir.

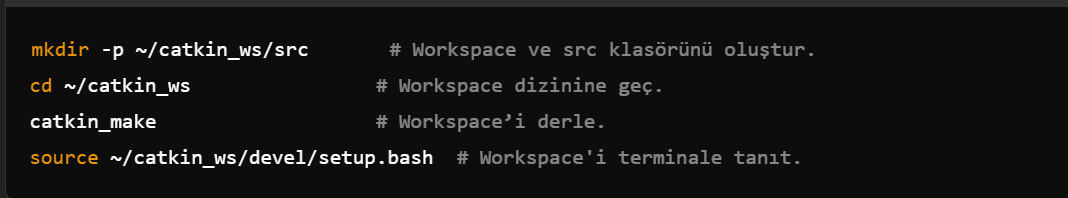
### ROS Elemanlarını Öğrenme Araştırması

**ROS workspace, nodes, topics, services, master-slave structure, parameter server, action lib**

**ROS workspace**

Adı üstünde ROS tabanlı projelerin yürütüldüğü çalışma alanı yani birden fazla paketi ve bu paketlere ait dosyaları içerir.4 temel klasörden oluşur.src,build,devel,install.Src kaynak kodlarının bulunduğu klasördür ve yeni paket oluşturmak için kullanılabilir.Build derleme sırasında geçici dosyaların oluşturulduğu klasör.Devel derleme sonrası çalıştırılabilir dosyalar ve kütüphanelerin yer aldığı ara klasör.Install tamamlanmış dosyaların dağıtımı için var olan klasör.

ROS workspace oluşturmak için aşağıdaki adımlar komut istemi(aslında bash yani linux için) üzerinde izlenir:



(chatgpt ile oluşturuldu)

Şekildeki ilk satırda “mkdir(make directory)” bir dizin oluşturur.”-p” birden fazla klasör oluşturmayı sağlar. “~/catkin\_ws/src kodu” soldan sağa sırasıyla önce ev dizininde(home directory) “catkin\_ws” adında bir ana dizin ve onun altında “src” alt dizini oluşturur.src klasörü ROS paketlerinin saklandığı yerdir.

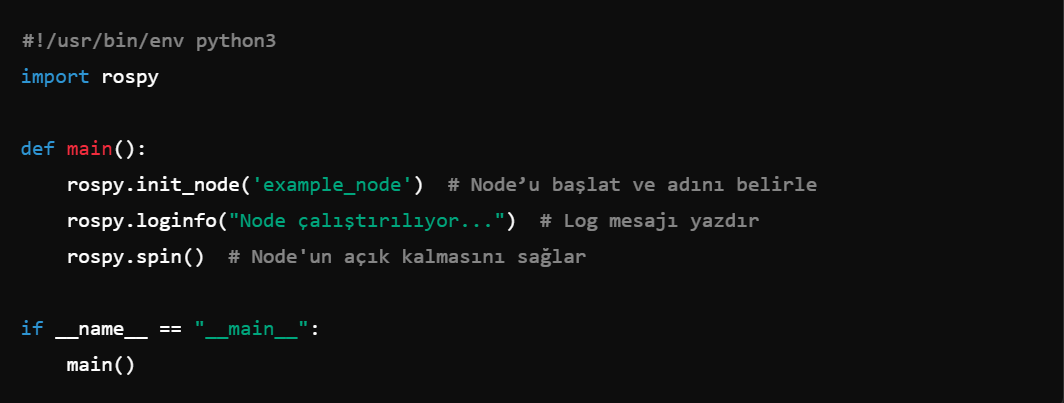
İkinci satırda “cd” change directory yani dizin değiştirme komutu,” ~/catkin\_ws” ise daha önce oluşturulan catkin\_ws dosyasına gidilmesini söyler.

Üçüncü satırda “catkin\_make” ROS’un derleme aracı olan catkin kullanılarak workspace’i(src klasöründeki tüm ROS paketlerini)derler.Derleme sırasında workspace içerisinde build,devel ve install(opsiyonel) ek klasörleri oluşturulur.

Dördüncü satırda “source” belirtilen dosyayı çalıştırır. “~/catkin\_ws/devel/setup.bash” geliştirme ortamını yüklemek için gerekli betik dosyası.

Workspace kullanımının bazı avantajları vardır.Modüler ve taşınabilir olması bunlara örnektir.

**Nodes**

Türkçe “düğüm” olan node.Küçük program parçaları sayılabilir.Örneğin bir robot oluşturulacağı zaman farklı farklı parçalara birbirinden farklı görevler ve buna bağlı olarak birbirinden farklı kod blokları düşer.Bu kod bloklarını birbirinden bağımsız olarak yapabiliriz.Kabaca **modüler** bir geliştirme ortamına olanak sağlar.Düğümler arası iletişim **topics**,**services** veya **actions** aracılığıyla sağlanır.Düğümler genellikle python veya c++ gibi dillerde yazılır.Aynı anda birden fazla node çalıştırılabilir.Bir node çalıştırıldığı zaman **ROS Master’a** kendisini kaydeder.Bu kayıt işlemiyle diğer node’larla iletişim sağlanır.Node’ların adı benzersiz olmalıdır.ROS Master üzerinde nodeların birbirini tanıması için bu özellik gereklidir.

Şekilde python dilinde örnek bir node yazımı bulunmakta.”rospy” ROS’un pyython istemcisidir bunu başta içeri aktardık.”rospy.init\_node(‘example\_node’)” düğümü başlatır ve ROS Mastera kaydeder.’example\_node’ düğümün adıdır. “rospy.loginfo("Node çalıştırılıyor...")”düğümün çalıştığını belirtmek için kullanılır.İçerisindeki mesaj konsola yazılır.İf içeren kısmı tam anlayamadım.Modül olarak başka bir dosyadan çağrılırsa “main()” çalışmamasını sağlıyor imiş.Node’u çalıştırmak için ise komut istemine **rosrun package\_name example\_node.py** yazılır.Node yapısı sayesinde **yeni bir işlev eklemek kolaylaşır ve aynı anda birden fazla işlev yapılabilir.**

**Topics**

Düğümler arasında veri paylaşımı sağlayan **iletişim mekanizması**dır.**Yayınlama/abonelik(publish/subscribe)** modeline dayanır,tek yönlü iletişim(yayınlayandan aboneye) sağlar.Bir düğüm topic üzerinde veri yayınlar(publisher) ve diğer bir düğüm bu veriyi alır(subscriber).Her topic belirli bir mesaj türünde çalışır.**Asenkron iletişim** vardır yani yayınlanan veri anında iletilmesine rağmen abone düğüm uygun olunca veriyi alır.Bir topic üzerinde birden fazla abone ve yayımlayıcı bulunabilir.Örnek kullanım alanlarında kamera görüntüleri,navigasyon sırasında veri paylaşımı,bazı robot hareket komutlarının iletilmesi sayılabilir.Avantajlarından birkaçı birden fazla düğüm arasında iletişim kurmayı sağlaması,her düğüm kendi görevine odaklandığı için modüler olması,yeni düğüm eklenince topics’e kolayca abone olması sayılabilir.Dezavantajları büyük miktarda veri taşınırken gecikmeye sebep olabilmesi,zamanlama gerektiren durumlarda asenkron olduğundan senkronizasyonda sıkıntı çıkarabilmesidir.

**Services**

Düğümlerde çift yönlü iletişim kurmak için kullanılan bir mekanizmadır. **Talep-cevap** (request-response) mantığına dayanır.Bir düğüm bir servis oluşturur(service server) diğer bir düğüm serise talepte bulunur(service client).Servisler genellikle **anlık işlemlerde** kullanılır.Spesifik bir işlemi gerçekleştirmek ve hemen sonucu elde etmek için uygundur(örneğin sensörden anlık veri almak).request ve response mesaj türleri **.srv** dosyalarıyla tanımlanır.Bir servis çağrıldığında sonuç alıncaya kadar istemci düğümü bekler bu da **senkron** olduğu anlamına gelir.Kullanım alanlarına örnek olarak robot kollarında bir nesnenin hareket ettirilmesi için komut gönderme,robotun belirli bir görevi başlatması veya durdurması verilebilir.

****

Services ve Topics farkları kabaca böyledir.

Services avantajları arasında anlık işlemlere uygun olması,modüler olması ve kolay kullanımı vardır.Dezavantajlarında ise sürekli veri akışı için uygun olmaması,servisten cevap alıncaya kadar beklemesi gerektiğinden senkronizasyon gereksinimi ön plana çıkar.

**Master-Slave Structure**

İki bileşen arasında iletişimi anlatan yapıdır.Master bileşeni diğer bileşenlere görevler verir ve görevlerin uygulanmasını takip eder.Kontrol kendisindedir.Slave bileşeni ise görevleri yerine getirip master’a iletir.Yukarıda bahsettiğim ROS Master ve düğümler arasındaki ilişki buna örnektir.Düğümler arası iletişimi ROS Master kontrol eder.Bu yapı merkezi koordinasyon sağlar.Anladığım kadarıyla bu yapının en büyük dezavantajı ROS Master’da bi sıkıntı çıkarsa tüm sistem çökebiliyor veya tüm sistemde bi sorun oluşabiliyor**.”Single point of failure”** diye geçiyor.

**Parameter Server**

Düğümler arasında paylaşılan parametrelerin merkezi bir depoda saklanmasını ve üzerinde işlem yapabilmemizi sağlar.ROS Master tarafından yönetilir.Anladığım kadarıyla parameter server belirli veri tiplerini destekliyor onlar da temel olanlar.Float,integer,string,boolean,listeler ve sözlükler.Örneğin robotun maksimum hızı bir başlangıç ayarıdır ve parametre server’ı kullanılır.ROS Master ile çalıştığından ROS Master arıza verirse kendisi de arıza verir bu da kendisinin dezavantajıdır.Byükü veriler için uygun değildir.

**Action Lib**

ROS’da **uzun süreli görevler** için kullanılan bir iletişim yöntemidir.Feedback ve status updates sağlayarak işlem hakkında devamlı bilgi edinmeyi sağlar ve gereken zamanda işlemi iptal edebilir.Bir robotun belirli bir noktaya gitmesi kullanılabileceği örnek bir senaryodur.Görev sırasında istemciye devamlı bilgi verir.Durum bilgisine göre başarılı başarısız işlemciyi yönlendirir.Robotun hareketi esnasında mevcut konumun düzenli olarak istemciyle paylaşılması bir örnektir.Robot kolunun belirli bir pozisyona ulaşması için de kullanılabilir.

**ROS .launch files, .yaml files, CMakelist.txt**

**ROS .launch files**

.launch dosyaları,birden fazla node u çalıştırmak için kullanılan XML tabanlı dosyadır.Anladığım kadarıyla birden fazla düğümü aynı anda kullanacağımız zaman bize kolaylık sağlıyor.Bu dosyalarla düğümlerin başlama sırasını belirleyebilir,dosya içerisine parametre tanımlayabilir,bir düğümün çalışması için koşul tanımlayabiliriz.

**.yaml files**

ROS gibi sistemlerde, parametre dosyalarını düzenlemek ve yönetmek için yaygın olarak kullanılır.İnsan tarafından okunabilir olması dikkat çeker.Hiyerarşik verileri(katmanlı yapı gibi de düşünülebilir)tanımlamak için kullanılır.Birçok programlama dilinde desteklenir.Veriler anahtar-değer sistemindeki gibi tanımlanıyor.Verilerin saklanmasında kullanıldığı gibi parametrelerin ayarlarıyla oynanacağı zaman da kullanılır.

**CMakelist.txt**

Basitçe ROS projelerinde c++ veya python kodlarını derlemek ve bağımlılıklarını yönetmek için bu dosya kullanılır.Çapraz platform derleme sistemi de deniyor.Derlenecek kaynak dosyalarını önce listeler,sonra derleme sırasında gerekli kütüphaneleri “bağlar”(linking),derleme sonrası yürütülebilir dosyalar oluşturur.Çapraz platform desteği vardır(farklı işletim sistemleri).

**ROS MoveIt Package**

ROS üzerinde bazı görevleri yerine getiren açık kaynaklı yazılım paketidir.Robot kollarının hareket planlamasında sıkça kullanılır.Robotun çevresel algılamasını adapte ederek esnek kontrol sistemi sağlar.Paketin bazı temel özellikleri vardır.Hareket planlama,kinematik çözümleri,çarpışma algılama ve önleme,3d algılama ve haritalama bunlardan bazılarıdır. Üretim hatlarında robot kollarının bazı görevleri üzerinde kullanılır.Hareket planlamasını basitleştirir.

**Gazebo, Rviz**

**Gazebo**

Robot simülasyonları için kullanılır.Bir tür yazılımdır ve robotları gerçek dünya dışında test etmemizi sağlar.Simülasyon ortamı için fizik motorları kullanır.3 boyutlu bir ortamda test imkanı sağlar.Aynı anda birden fazla robot test edilebilir.Dinamik nesneler simülasyona eklenebilir bu da daha gerçekçi test imkanı sağlar.Açık kaynaklı olması ve gerçekçi olması avantajı iken yüksek düzey simülasyonda performans sorunlarına neden olabilmesi dezavantajıdır.

**Rviz**

Açılımı ROS visualization ve adı üstünde robotların ve sensör verilerin görselleştirilmesi için kullanılır.Robot modellemede,haritalamada,hareket planlamada görselleştirmeden dolayısıyla Rvizden faydalanılır.Beni en çok heyecanlandıran özelliği ise dinamik verileri görselleştirebilmesi yani bunu anlık yapabilmesi oldu.Hataları tespit etmede de kolaylık sağlıyor.

**ROS tools->rqt\_plot, rqt\_graph, rosbag kavramları**

**rqt\_plot**

Topic’te yayınlanan verilerin zamanla değişimini grafik üzerinde gösterir.Gerçek zamanlı veri görselleştirme imkanı tanır.Birden fazla topic’i görselleştirerek veri kıyaslaması yapmamızı sağlayabilir.

**rqt\_graph**

Düğümler ve topicler arasındaki bağlantıyı görselleştirir.Bir diğer deyişle düğümler hangi topic’lere veri yayınlıyor ve hangi topic’lere abone, bunları görselleştirir.Sistemin genel yapısını anlama ve hata ayıklamada kullanılabilir.

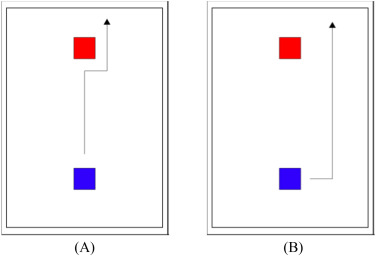
**Rosbag**

ROS verilerinin kaydedilmesi için kullanılır.Test ve hata ayıklamada kullanılır.Kayıt süreci filtrelenebilir yani sadece istediğimiz topic’ler kaydedilebilir.Depolanan dosyanın adına kabaca bag denir.

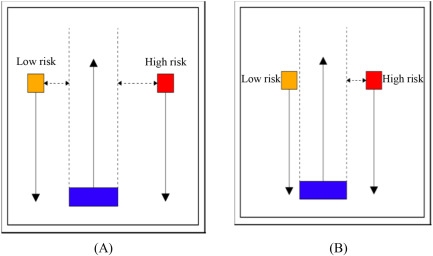
## Çarpışma Önleyici Algoritmalar Araştırması

**Robotikte sıkça kullanılan Çarpışma Önleyici Algoritmalar nelerdir ve nasıl çalışırlar?**

Bu algoritmalar robotların çevrelerindeki nesnelere çarpmalarını engellemek için kullanılır.Robotun nesneye algılaması gerkeceğinden sensör,matematiksel modelleme ve kontrol yöntemlerini içerirler.Anladığım kadarıyla robotlar Q-Learning algoritması ile risk hesabı yaparak çalışıyor.

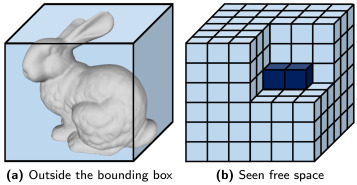


Şekildeki örnekte 2 adet kaçış yöntemi varken birisi düşük risk diğeri yüksek risk barındırıyor.Bu Q-Learning algoritmasında yüksek öncelik obje ve düşük öncelik obje bulunuyor.Duruma göre hangi yöntemin kullanılacağına karar veriliyor.B seçeneği daha insanımsı davranış iken A seçeneği daha acemi bir davranış olarak tanımlanmış.

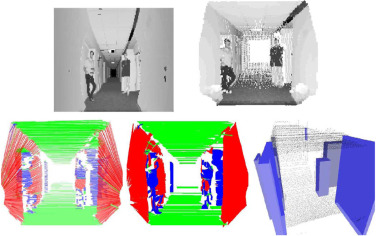


Bu görselde yapay zeka navigasyon sisteminin davranış örneği tanıtılmış.Danger zone denilen alana girmediği zaman obje düz ilerlerken,cisimler alana yakınlaşınca yüksek riskli(aynı zamanda yüksek değerde obje de denebilir) cisme çarpmamak için diğerine çarpma ihtimaline rağmen ona daha yakın gidiyor.

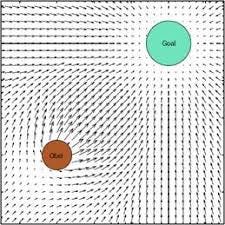
Çarpışma önleyici yöntemlere birkaç örnek vereceğim.

Örneğin aynı websitesi üzerinde “Outside the bounding box” denilen bir yöntemden de bahsetmiş.

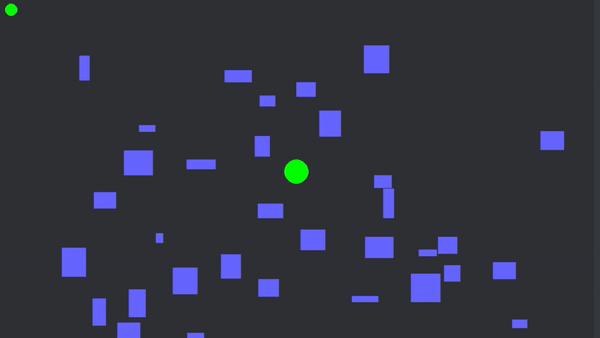
A şıkkındaki yöntemde nesneler etrafında “bounding box” denilen kutular tanımlanır robot bu kutuların dışından hareket eder.Basit olması avantajlı iken büyük nesnelerde sorun çıkarabildiğinden bahsedilmiş.B şıkkındaki “seen free space” yöntemi ise anladığım kadarıyla bir sensör bölgeyi devamlı tarayarak “boş” veya “dolu” olarak alanları ikiye ayırıyor ve sadece “boş” olan yerlerden geçiyor.Büyük veri yoğunluğu içerebilmesi ve zaman alması dezavantajı olarak görülüyor.



Bu görselde ise robotun engelleri algılamak için sensör kullandığı örnek bir durum görsellendirilmiş.

Bir başka yöntem ise “potential field method” denilen bir yöntem.

Temelinde Laplace eşitliğini kullanıyor bu denklem.Algoritma alan üzerinde yapay bir potansiyel alan grafiği oluşturur.Varış noktası çekim kuvvetine sahip iken etraftaki nesneler itiş kuvvetine sahiptir.Toplam kuvvetin yönü ve büyüklüğüne göre robotun yönü belirlenir.Robotun hedefe ulaşamadığı yerel minimum problemlerine yol açabiliyormuş.



Yukardaki diğer yöntem ise “Rapidly Exploring Random Tree” yöntemi olarak geçiyor.Rastgele noktalardan başlayarak robotun hareket edebileceği bir ağaç oluşturuyor.Görseli görene dek anlamak çok zor olduğu için özellikle buraya eklemek istedim.Engellerin yoğun olduğu ortamlarda işe yarayabilir iken rastgelelik nedeniyle uzun yollar üretebiliyor.Daha birsürü yöntem var bunlar bana anlaması en kolay gelenlerdendi.Robotun bulunduğu ortama göre kullanılması tavsiye edilen yöntem değişiyor.Örneğin statik ortamlarda potansiyel alan yöntemi.Fakat dinamik ortamlar ve onlar için önerilen yöntemler açıkçası daha karmaşık ve zor geldi.

**Çarpışma önleme bir robot için neden önemlidir?**

Bunun birkaç sebebi var.Öncelikle robotun ve çevrenin hasar görmesini istemiyoruz.Çevreye zarar vermek güvenlik riski ve ekstra maaliyet demek.Robot hasar gördüğünde ise bileşenleri hasar görebilir ve bu durumda tamir maaliyeti ortaya çıkabilir.Örneğin üretim hatlarında insan ve robotlar birlikte çalışır.Oralarda insan güvenliği de önemli bir sebep oluşturur.Ayrıca bir diğer sebep ise görev boyunca çarpışmalardan kaçınılması görev süresince zaman tasarrufu sağlamamıza neden olması.Algoritmalarda ne kadar ileri seviye olursak robotlar da o kadar karmaşık ortamlarda çalışabilir hale gelir.Son olarak da çarpışma sayısının azalması uzun vadede bakım masraflarının da azalması demek.

**Afet sonrası ortamlarda çarpışma önleme neden önemlidir?**

Bu başlık için aslında birçok sebep sayılabilir.Örneğin yapısal çöküntüler ve enkazların olduğu bir ortamda oluşan düzensiz yapı robotların kolayca çarpışmasına neden olabilir.Robot enkaz içerisinde güvenle hareket etmelidir.Eğer ortada bir hayat kurtmarma operasyonu var ise işin ucunda insan hayatı vardır ve bu oldukça önemli bir nedendir aslında.Diyelim ki bir arama kurtarma robotu ile çalışıyoruz.Robotun enkazda çöküntüye neden olmasını istemeyiz.Robot eğer üzerinde hassas bir cisim taşıyorsa cisme zarar gelmesini istemeyiz.Afet bölgelerinde sadece robotlar değil,kurtarma ekipleri ve gönüllüler de olabilir.Bu durumda robot bu kişilerle beraber çalışabilmelidir.Diğer robotlarla da eğer var ise koordinasyon halinde çalışması gerekebilir.İnsan hayatı söz konusu olduğunda bir saniyelik bir zaman tasarrufu bile işimize yarar.

**Çarpışma önleme hangi ortamda daha zordur?**

* + **Sadece bir robot ve etrafındaki statik ortam.**
  + **Hareket eden objelerin ya da başka robotların olduğu bir ortam.**

Bu soruya henüz araştırmaya başlamadan bile ikinci seçenek olduğunu geçmiş sorulardan söyleyebilirim.Zaten dinamik ortamlar için oluşturulmuş algoritmalar oldukça anlaması zor da gelmişti :).Örneğin dinamik ortamlarda öngörülemez engeller çıkma olasılığı daha fazladır ve robotun daha hızlı karar alması gerekir.Yani robot gerçek zamanlı karar vermek zorunda kalabilir.Üstüne üstlük bir de diğer robotlarla böyle bir ortamda iletişim kurması zorluk seviyesini katbekat arttıracaktır diye tahmin ediyorum.Hareket eden objelerin algılanması daha zor iken ayrıca robot manevra yapacağı için objenin gelecekteki pozisyonunu da tahmin etmesi gerekir.

**Kaynakça**

1. <http://wiki.ros.org>
2. Quigley, M., Gerkey, B., & Smart, W. D. (2015). *Programming Robots with ROS: A Practical Introduction to the Robot Operating System.* O'Reilly Media.
3. <https://www.openrobotics.org>
4. Buttazzo, G. C. (2011). *Hard Real-Time Computing Systems*. Springer.
5. <http://wiki.ros.org/Parameter%20Server>
6. <http://wiki.ros.org/ROS/Tutorials>
7. Mastering ROS for Robotics Programming, Lentin Joseph, Packt Publishing.
8. <http://wiki.ros.org/rospy>
9. <http://wiki.ros.org/actionlib>
10. <http://wiki.ros.org/roslaunch>
11. <https://yaml.org/>
12. <http://wiki.ros.org/catkin/CMakeLists.txt>
13. <https://moveit.ros.org/>
14. <http://gazebosim.org/>
15. <http://wiki.ros.org/rviz>
16. <http://wiki.ros.org/rqt_plot>
17. <http://wiki.ros.org/rqt_graph>
18. <http://wiki.ros.org/rosbag>
19. Khatib, O. (1985). *Real-time obstacle avoidance for manipulators and mobile robots*. The International Journal of Robotics Research, 5(1), 90-98.
20. LaValle, S. M. (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Technical Report, Iowa State University.
21. <https://medium.com/@rymshasiddiqui/path-planning-using-potential-field-algorithm-a30ad12bdb08> potansiyel alan görselleri için
22. <https://www.sciencedirect.com/topics/engineering/collision-avoidance> çarpışma engellemeyi anlatan birkaç makalenin sırayla verilmiş hali.
23. Murphy, R. R. (2004). *Trial by fire: Activities of the rescue robots at the World Trade Center from 11–21 September 2001*. IEEE Robotics & Automation Magazine, 11(3), 50-61. DOI: 10.1109/MRA.2004.1371615