

**İstanbul Teknik Üniversitesi Robotik Arama Kurtarma Ekibi**

**Görev Raporu**

**Görev Konusu:** MoveIt üzerine çalışmalar ve bazı ROS konseptleri

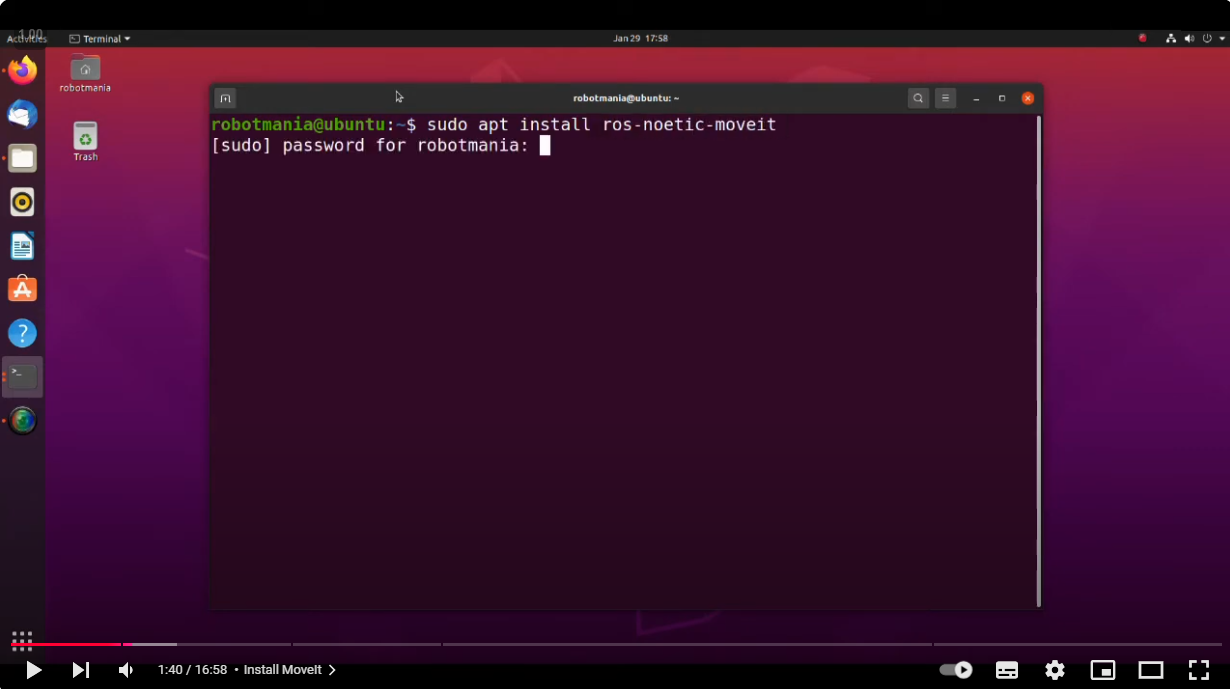
**Hazırlayan:** Hasan Salih Şahin

**Teslim Tarihi:** 30.11.2024

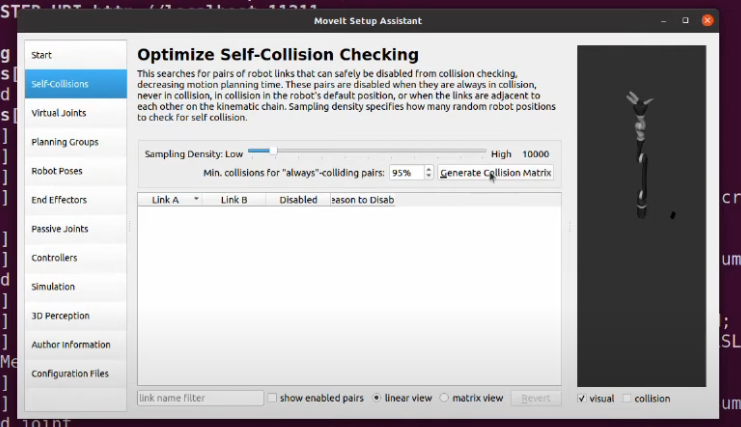
**MoveIt üzerinden robot kol kontrolü sağlanması**

İzlediğim bir videoyu aşama aşama burada anlatarak bu sunumu yapmak istiyorum çünkü gördüğüm github repoları,diğer birçok video ve kaynak anlaması çok zor geldi.

Öncelikle ubuntu üzerinden moveit kurulumu yapılıyor.

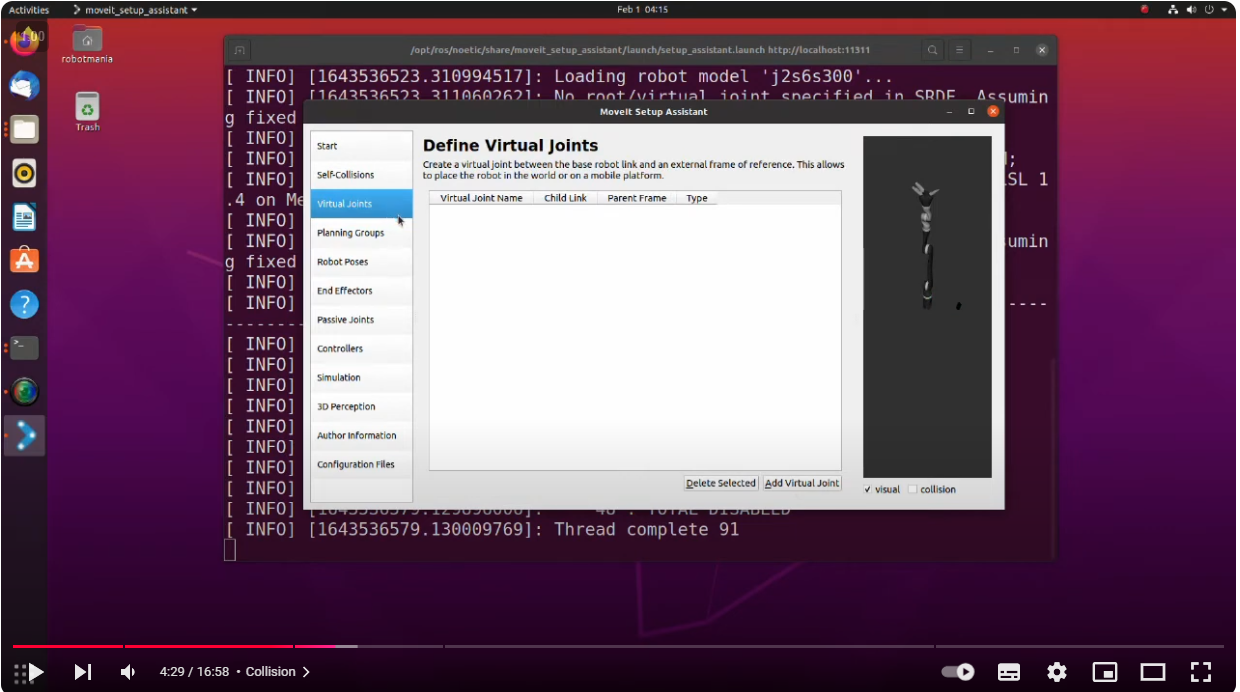
****

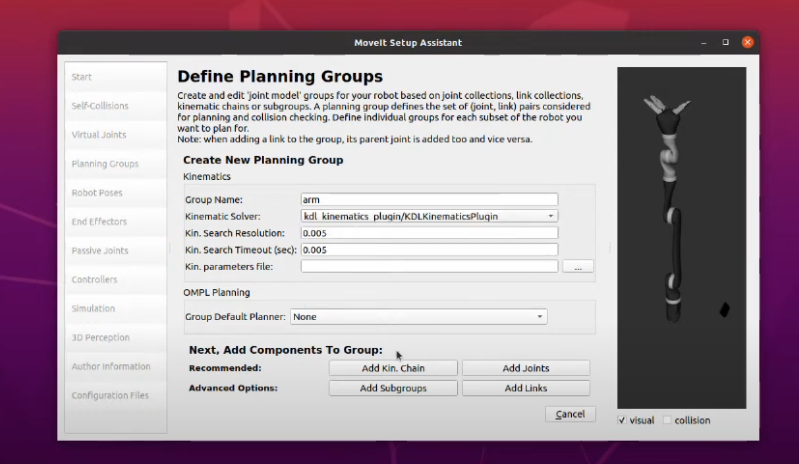
Bu kısmı anlatmayı gereksiz buluyorum ve atlıyorum.Daha sonra moveit setup asistanı karşımıza çıkıyor ve “Kendi Kendine Çarpışma Kontrolünü Optimize Etme” gibi orada birkaç ayar yapıyoruz.Bu ayarla robot bağlantıları arasında mantıksal olarak asla çarpışamayacak olan ya da sürekli çarpışma içinde olan çiftleri belirleriz ve bu çiftleri çarpışma kontrolünden çıkarırız.

****

Daha sonra videoda bir altta gözüken Virtual joints yani sanal eklem ayarına giriyoruz

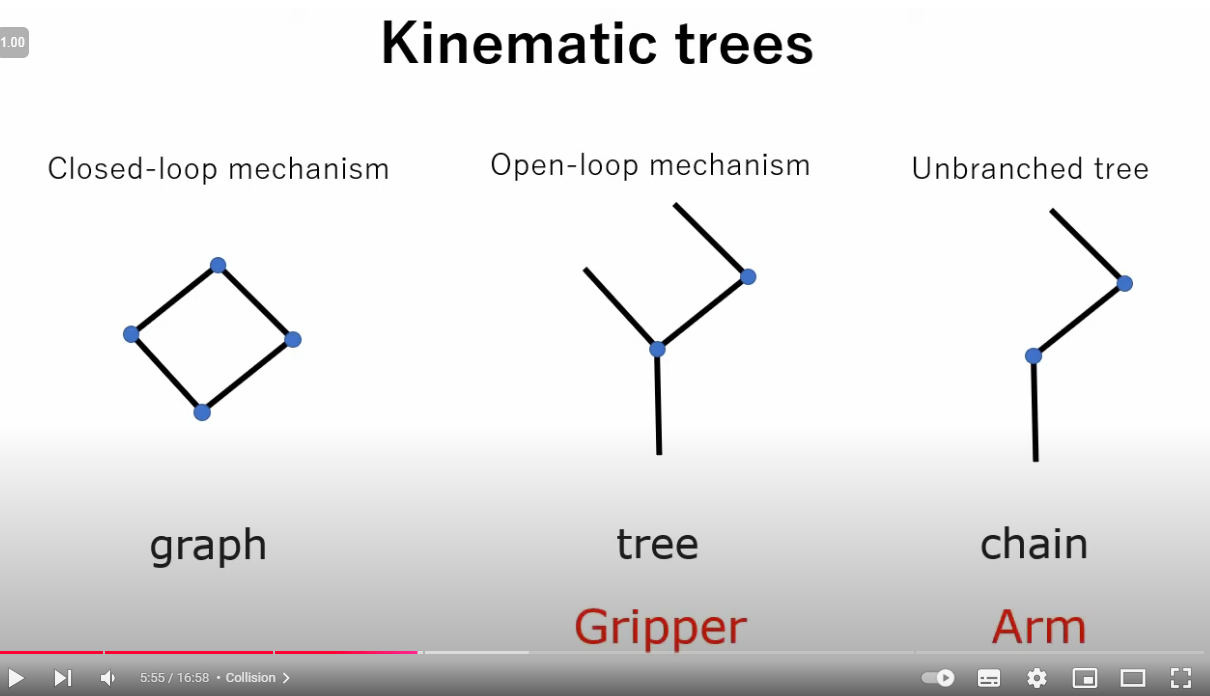
Burada eklediğimiz sanal eklem,mesela mobil robot platformu ile robot arasında bağlantı kurmamızı sağlıyor.



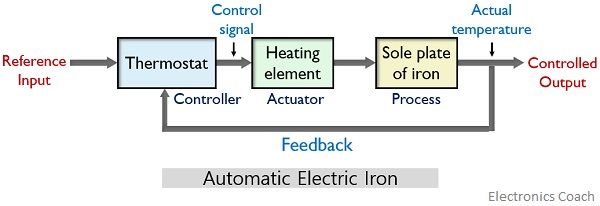
Daha sonra robot üzerinde kinematik hesaplamaları için gruplar oluşturduğumuz kısım geliyor.

”Bir planlama grubu, robotun belirli bir kısmını (örneğin, bir robot kolu veya uç efektör) kontrol etmek için kullanılan eklem (joint) veya bağlantı (link) koleksiyonlarını ifade eder.”

Daha sonra kinematik zincir ekle kısmına tıkladığımızda bize kinematik zincirin ne olduğu anlatılıyor.

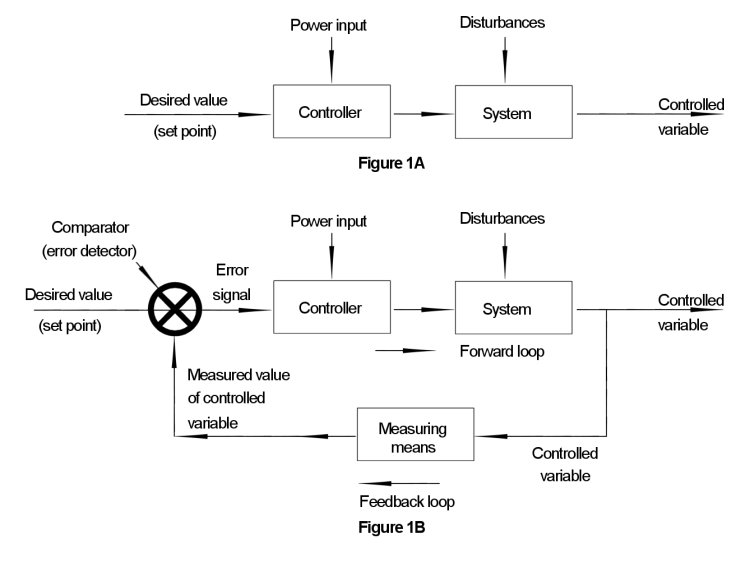
****

Closed loops mechanism anlamak için biraz daha araştırdım. Kapalı bir döngü oluşturacak şekilde bağlantılar ve eklemlerin birleştirilmesi diyebiliriz.



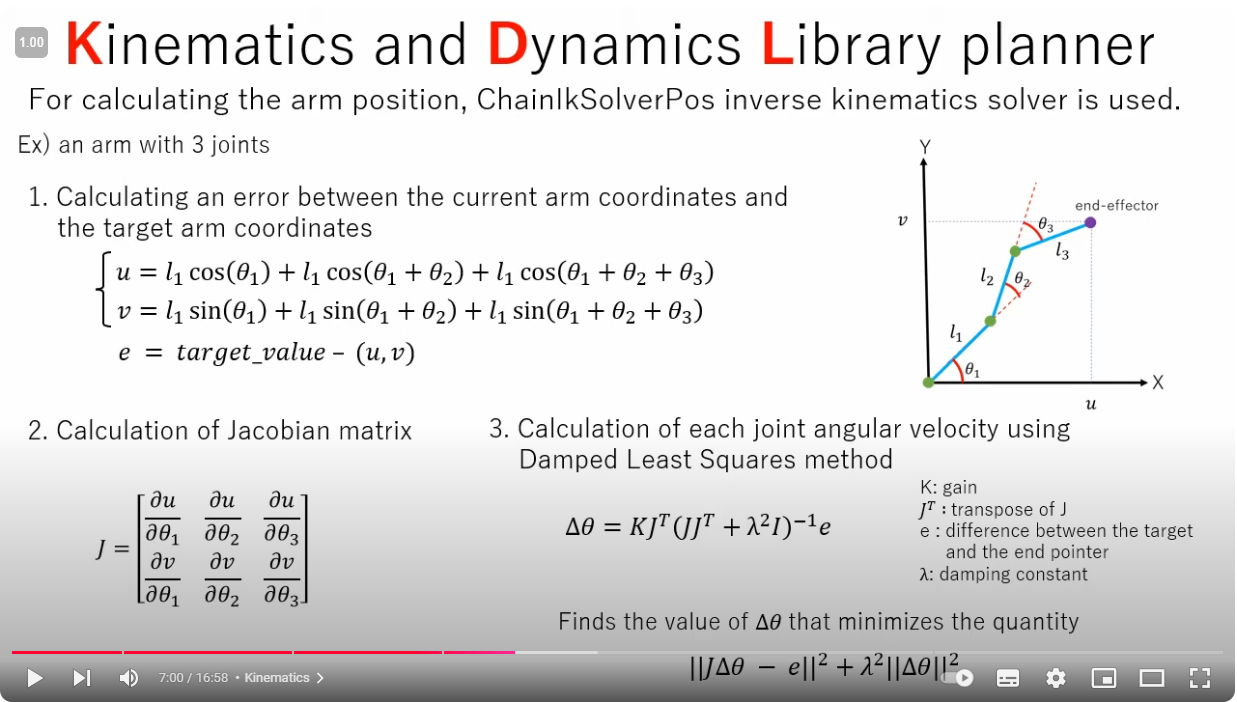
Görselde de gösterildiği gibi sonucun devamlı olayın tekrarlanmasını sağlayan bir feedback olması gibi **kapalı** döngü içeren mekanizmalara verilen ad.

Bir diğer sistem ise open-loop mechanism.Burada bağlantılar ve eklemler bir ağacın dallarına benzer şekilde düzenlenir.Olayın devamlı tekrarlanmasını sağlayan bir “feedback” döngüsü yoktur.Aşağıdaki şekilde aradaki farkı anlatan 2 farklı durum gösterilmiş.



Unbranched tree’de ise dallanma bulunmuyor.Örnek olarak da geleneksel robot kolları gösterilmiş.

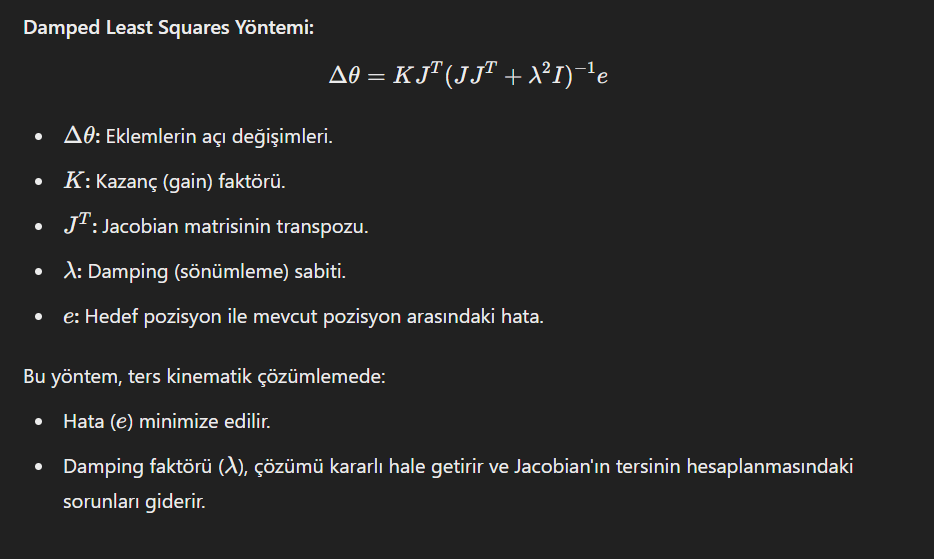
Daha sonra “Kinematics and Dynamics Library planner” olarak geçen , robotların hareketlerini hesaplamak ve simüle etmek için kullanılan bir yazılım kütüphanesinin tanıtımına geçiliyor.

****

Burada error olarak geçen e, hedef pozisyon ile mevcut pozisyon arasındaki fark ve burda amacımız bu hatayı minimize ederek uç “efektör”ü hedefe ulaştırmak.

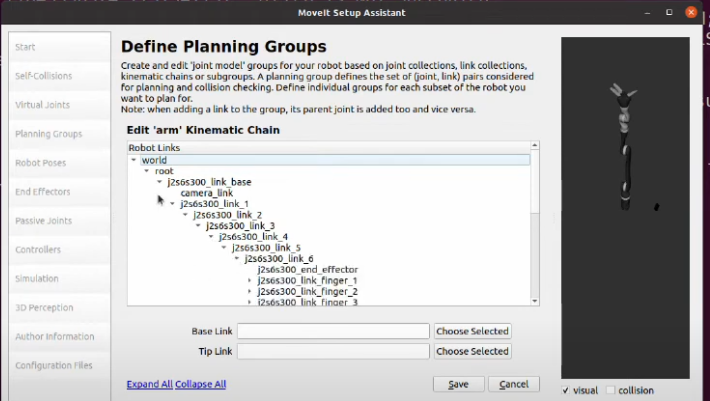
“Jacobian matrisi, uç efektörün pozisyonundaki küçük değişimlerin eklem açılarına olan etkisini temsil eder.”

Henüz 2. Sınıf olduğum için jacobian matrisinin daha önce görmüş olsam da tam bilemiyorum ama kısmi türev alınıyor.Jaobianı açıları bulurken kullanıyoruz o da sağ alttaki formül.Chatgpt’nin sağ alttaki formüle açıklaması ise şöyle:

****

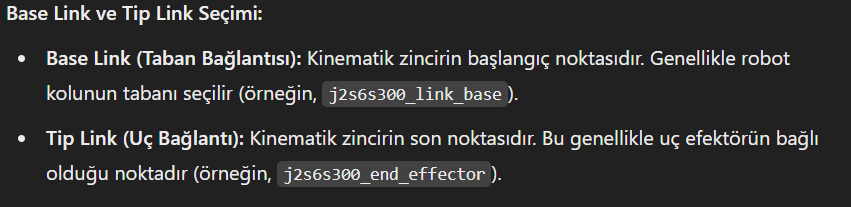
Daha fazla açıklama yapmayıp bir sonraki aşamaya geçmek istiyorum.

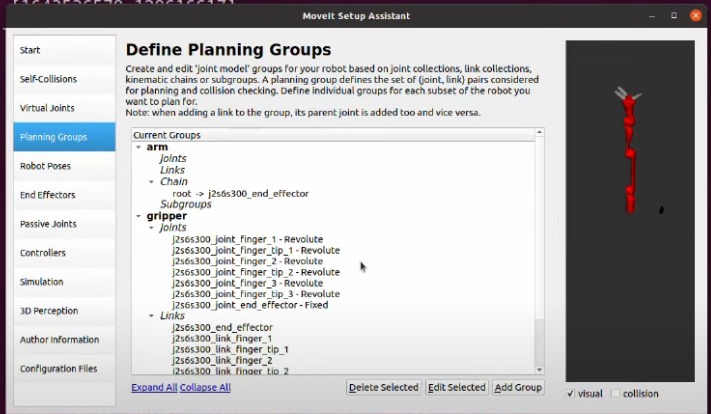
“Kinematik zincir” düzenleme ekranına geliyoruz.



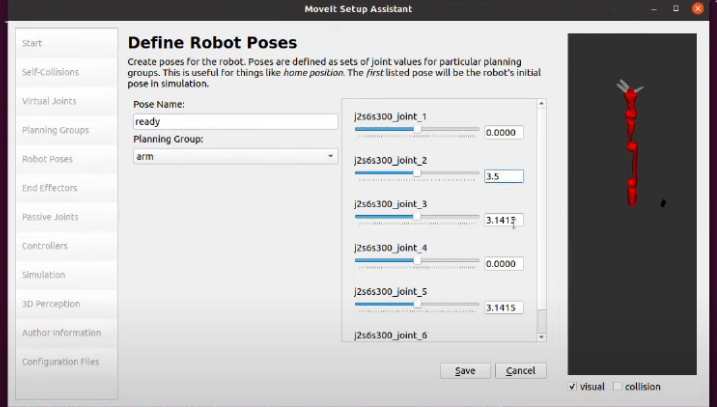
Burada robotun bağlantıları ve eklemleri belirleniyor.Örneğin, bir robot kolunun pençesinin pozisyonu ve yönü, bu kinematik zincir boyunca belirlenir.

Örneğin burada “root” dünya koordinat sistemini ifade eder.Sürekli alt başlık olarak ilerleyen duruma hiyerarji deniyor.Chatgpt’den özellikle aldığım cevap:



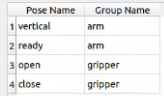
Açıklayıcı olduğu için direk yazayım dedim.Sağ taraftaki görsel ise kinematik zincirin robot üzerindeki fiziksel karşılığını gösteriyor.

Burada arm ve gripper olarak artık 2 tane grup tanımlıyor.

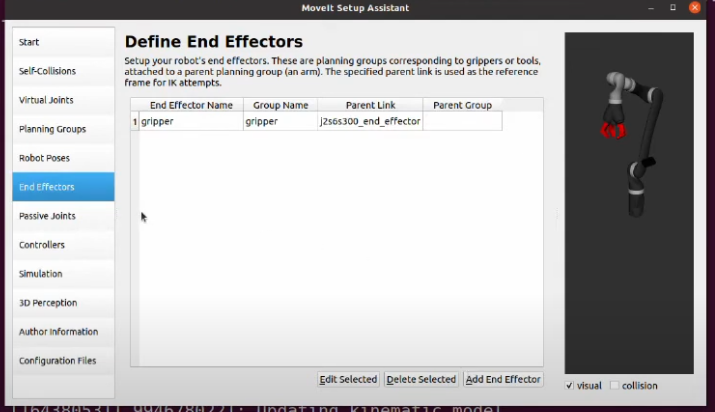
****Bir de buradan sonra “robot poses” denen yere geçiyor.

Burada, robotun farklı pozisyonlarına karşılık gelen eklem değerleri (joint positions) ayarlanır ve bu pozisyonlar isimlendirilerek kaydedilir.

Sağ tarafta eklemler için verilmiş açılar radyan cinsinden giriliyor.”Planning group”ta hareket edecek parça seçilip,”pose name” ise o açılardaki duruşun adıdır.

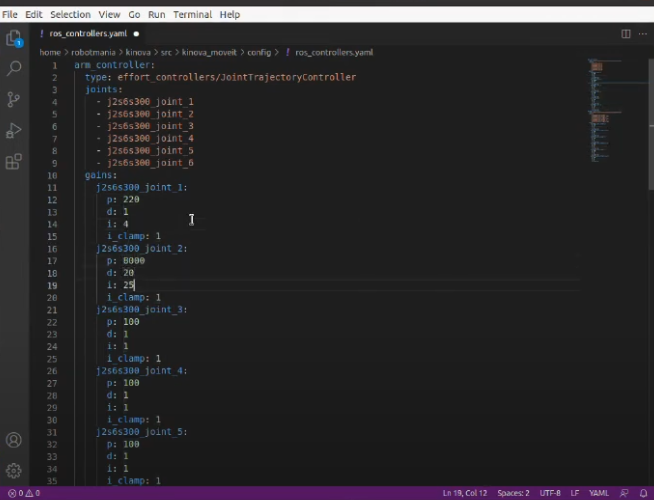
****

Burada oluşturduğu 4 pozdan olayın varacağı yeri az çok tahmin etmeye başlıyorum.Pozlar arası geçiş yapılacak.

Daha sonra “end effector” olarak gripper’ı ayarlayabilmek için bir ayar yapıyor:

Daha sonra gazebo simülasyonuna geçeceğini fakat ondan önce birkaç modifikasyon yapması gerektiğinden bahsediyor.”ros\_controller.yaml” dosyası açılıyor.

Bu dosyadan sırasıyla tüm eklemlerdeki p,i,d değerlerini arttırıyor.



Anladığım kadarıyla hareketi hızlıca ve doğru gözlemleyebilmek için bu ayarları yapıyoruz.PID kontrolü olarak geçen mekanizma,hedef konuma ulaşılmasını sağlayan bir algoritma.Sırayla “Proportional - Orantılı Kontrol”,” Integral - İntegral Kontrol”,” Derivative - Türevsel Kontrol”.

Joint\_limits.yaml dosyasını açarak da eklem hızı ve ivmesini arttırıyor ki rahat görelim.

Gazebo.launch dosyasından da bizim projeyi seçecek referans seçiliyor.

Simülasyonu çalıştırmak için de birtakım işlem yapıyoruz.Önce python3-pip indirilir, opencv-contrib paketi açılır.Drivedan bir link vermiş,onu indirip sıkıştırılmış dosyayı açıyoruz.src ve ui dosyalarını kinovaya yapıştırıyor.catkin\_make komutunu yazıyor.Gazebo dosyasına bir dosyayı yapıştırıyor.Daha sonra kinovayı açıp gazebo çalıştırıyor. “aruco\_pose\_estimation.py” komutu ile python betiği kullanılarak markerın uzaydaki konumu hesaplanıyor.

**Robotik sistemi oluşturan konseptler**

**Localization**

Robotun bulunduğu ortamda kendi konumunu ve yönelimini belirlemesi sürecidir.Başka bir deyişle haritada nerede olduklarını anlamalarını ve harita olmayan ortamda çevreyi analiz edebilmelerini de içerir.Özellikle otonom robotlar için önemli olduğu belirtilmiş.Haritalamada,çevre haritalandırıldığı için robotun konumu ve yönelimi önemlidir o yüzden kullanılır.Sensörlerdeki hatlaarı düzeltmede kullanılır.Navigasyon sisteminde de doğal olarak kullanılır.

Localization ile ilgili bazı terimlere değinmek istiyorum.

**Global Localization:**Adından da tahmin edileceği üzere robotun en başta kendi konumunu bulunduğu dünyada bilmemesi durumunda önce kendi konumunu saptaması olayıdır.

**Position Tracking:**Robotun hareket ettiği yönde pozisyonunu devamlı güncellemesi olayı.

**Kidnapped Robot Problem:**Robotun ani bir şekilde farklı bir yere taşındığında konumunu yeniden bulma durumu.

Localization için de birkaç yöntem bulunuyor.Belli başlı mantığa dayananları mevcut.

Örneğin **Sensör Tabanlı Yerelleştirme:**

Bu yöntemde sensörlerden faydalanılarak konum bulunmaya çalışılıyor.Ultrasonik sensörler ses dalgalarıyla mesafe ölçümü yapabiliyor,Lidar denilen bir yöntem mevcut onda da lazerlerden faydalanılıyor(kulağa oldukça sofistike geldi).Veyahut kamera ile klasik görüntü işleyerek de bu durum yapılabiliyor.

Bir diğer yöntem ise **Matematiksel ve Algoritmik Yerelleştirme:**

Buradaki yöntemler hep karmaşık geldi bana.Graph-based localization,monte carlo localization bunlardan bazıları.

Bir de SLAM denilen bir muhabbet var.Robot aynı anda hem kendi yerini belirliyor hem de çevrenin haritasını çıkarıyor.Gerçekten buraları incelerken ufkum açılıyor diyebilirim.Çok yaratıcı yöntemler.

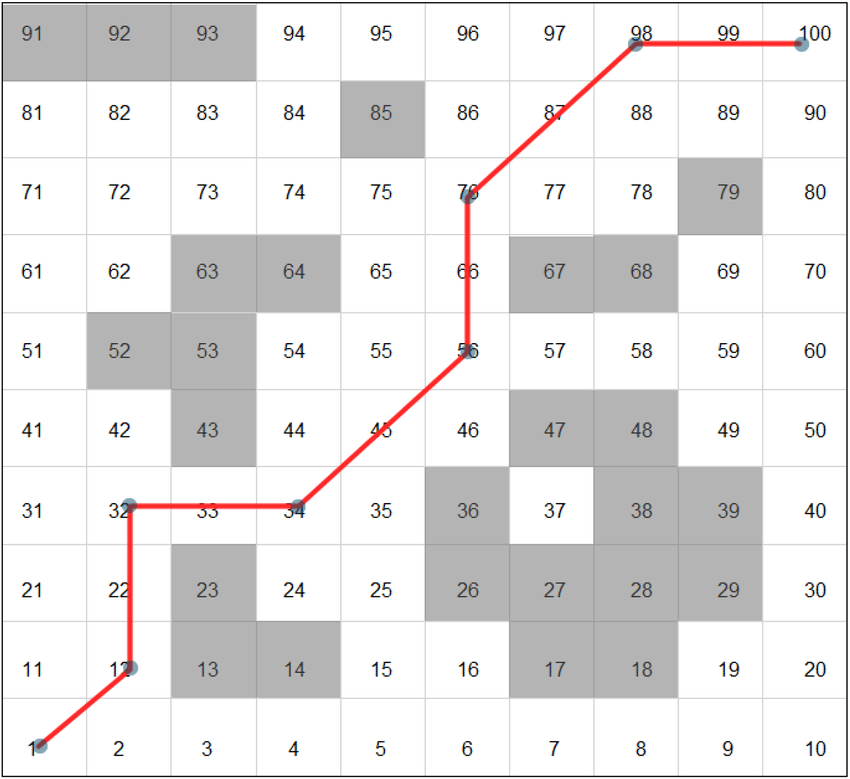
Localization sürecinde çıkabilecek zorluklardan da bahsetmek istiyorum.Sensörlerden gelen verilerin hatalı olmamasına dikkat etmek lazım.Dinamik bir ortamda çalışılıyorsa her işte olduğu gibi dinamik ortam bu işi de zorlaştırıyor.

**Mapping**

Aslında localization’ı anlatırken kullandığım bazı tanımlara çok benziyor.Robotun çevrenin haritasını çıkarması diyebiliriz.Engellerden kaçınmada ve otonom robotlarda elzem bir durum olduğu özellikle belirtilmiş.Mappig önemlidir çünkü robot hareket edeceği ortamı daha iyi anlar.Hareket ederken doğru rotayı seçer.Statik haritalama ve dinamik haritalama diye 2 çeşidi bulunuyor.Statikte adı üstünde çevre sabit,dinamikte ise çevre sürekli değişiyor.SLAM kavramını burada da kullanmışlar.Haritalamada da belli başlı yöntemler var bazılarından bahsetmek istiyorum.

Grid-Based Mapping:

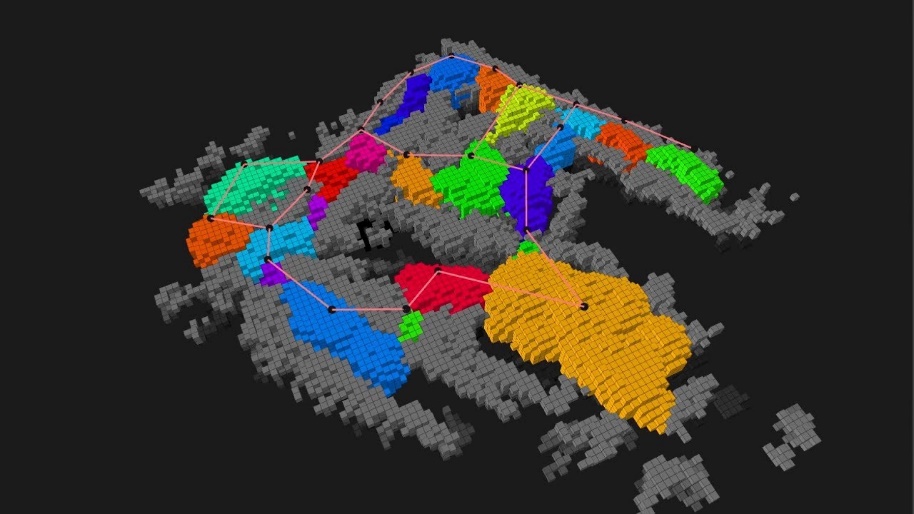
Harita ızgaralara bölünüyor



Şekildeki örnekteki gibi oluyor.Her hücrede robotun bu hücreye girip girmediği kontrol edilir.Yüksek bellek gerektirmesi bir dezavantaj oluşturuyor.

Topological Mapping:

Deep rock galactic diye bir oyun oynamıştım orada haritanın çalışma prensibine çok benziyor.Detaylı geometrik bilgi vermiyor.Çevre bağlantılı düğümler arası geçişlerle modelleniyor.



Lidar kullanılarak çevredeki mesafeler lazerlerle de ölçülebiliyor.Buna da lidar tabanlı haritalama deniyor.

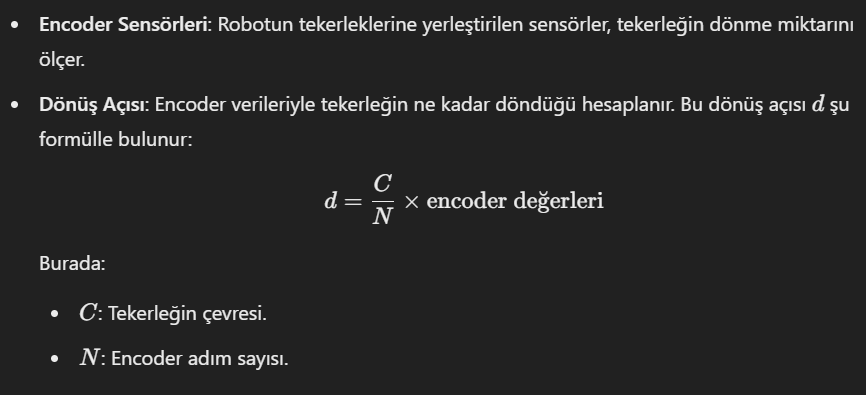
Mapping yapılırken kullanılan algoritmalar diye bir başlık da var.Gmapping,Cartographer,RTAB-Map bunlardan bazıları.Günümüz otonom araçlar mapping konseptini birinci dereceden aldığı belirtilmiş.Takımımızı ilgilendiren arama kurtarma durumunda da oldukça büyük öneme sahip.

**Object Detection**

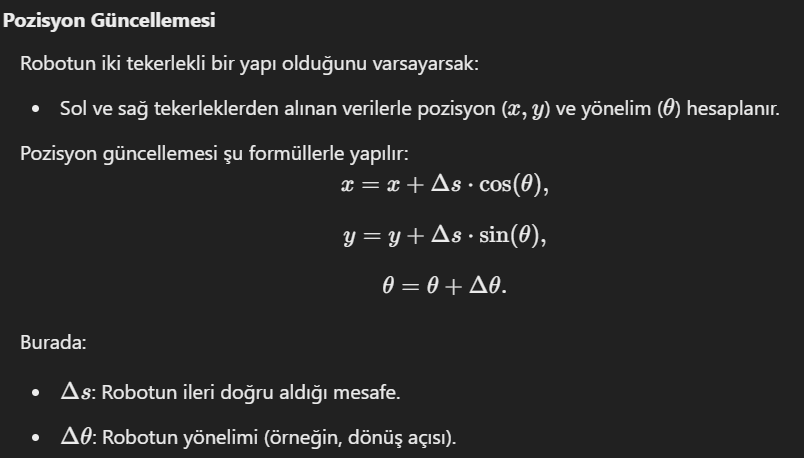
Robotun çevresinde bulunan nesneleri algılayıp sınıflandırması ve pozisyon belirlemesi süreci diyebiliriz.ROS’un halihazırda bunun için sunduğu bir sürü paket ve algoritma bulunuyormuş.Robotikte bu konseptin öneminden bahsetmek istiyorum.Robot çevresini algılarken,harita oluştururken,bir görevi yerine getirirken neredeyse her görevde etraftaki objeleri tespit etmesi gerekiyor.Üretim hatlarında nesnelerin ayrıştırılması gibi yerlerde kullanılıyor.Nesne tespiti süreci 3 parçadan oluşuyor.Algılama,sınıflandırma,konumlandırma.

**Odometry**

Bir robotun x,y koordinatlarında pozisyonunu ve yönelimini belirlemek için kullanılan bir yöntemdir. Genellikle robotun tekerlek hareketlerinden veya diğer hareket sensörlerinden gelen verilerle gerçekleştirilir. Bir robotun çevresinde nereye hareket ettiğini anlamak ve navigasyon yapmak için temel bir bileşendir.Genellikle tekerlek tabanlı hesaplama yapılır.Bunu ise chatgpt ile araştırdım.



Daha sonra pozisyon güncellemesi yapılıyor.Onun için ise süreci şöyle anlatmış.

****

Odometri içerisinde bir sürü formül bulunduğunu fark ettim.Gerçekten hepsini çalışmam va açıklamamın konu dışına çıkacağını düşünüyorum.

SLAM süreçlerinde de odometrinin sık sık kullanıldığını gördüm.Odometri verisi,robotun hareket tahmini için kullanılıyor.ROS’ta belli başlı odometri paketleri bulunuyor. robot\_localization, diff\_drive\_controller, gazebo\_ros bunlardan bazıları.Ayrıca hesaplamada hata olursa kümülatif hata oluşabiliyormuş yani hatalar birikip büyük hatalar oluşturuyor.

**Kaynakça**

1-Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. MIT Press

2-Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. MIT Press.

3-Durrant-Whyte, H., & Bailey, T. (2006). "Simultaneous Localization and Mapping: Part I." *IEEE Robotics & Automation Magazine*.

4-Quigley, M., Gerkey, B., & Smart, W. D. (2015). *Programming Robots with ROS*. O'Reilly Media.