Jacobian Pseudo Inverse Yaklaşımı

I. Giriş

Bu raporda 2021 yılında kullanılan ters kinematik algoritmalarından Jacobian Pseudo Inverse algoritması genel hatlarıyla incelenecektir.

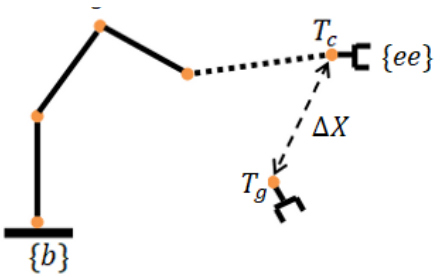
II. Algoritmanın incelenmesi

Jacobian yöntemine gelmeden önce ileri-ters kinematik arasındaki ayrımı hatırlayalım. İleri kinematik genel anlamda eklem bilgilerinden (açı vs.) end-effectorun konumunu bulmak, ters kinematik end-effectorun uzaydaki konumundan eklem bilgilerini hesaplamak şeklinde özetlenebilir.

Jacobian inverse ise ters kinematik problemlerinin çözümünde kullanılan yaklaşımlardan biridir. Nümerik bir yaklaşımdır, hedefe küçük adımlarla yaklaşılması ve belirli bir mesafeye ulaşınca yaklaşımı bitirme mantığıyla çalışır. Adım büyüklüğü ve hata toleransı algoritmayı kuran kişiye kalmıştır.

Bu yaklaşımda eklemlerin hızları ile end-effectorün konumu arasında bir ilişki kurulur. Zaten Jacobian matrisinin devreye girdiği nokta da burasıdır. Jacobian matrisi vektör değerli bir fonksiyonun bütün kısmi türevlerini içeren bir matristir. Türevin de anlık hız olduğunu biliyoruz :)

Anlatımı daha da karıştırmadan algoritmanın yüzeysel olarak nasıl işlediğini görelim:



1. End effecorün şimdiki konumu ile hedeflenen konum arasındaki fark hesaplanır (ΔX)

2. Adım büyüklüğünün belirlenmesi için mesafe küçük bir katsayı ile çarpılır (δX **=** f* ΔX)

3. Anlık eklem bilgileri (açılar) kullanılarak Jacobian matrisi hesaplanır

4. Jacobian matrisinin pseudo inverse hali hesaplanır (Bazı tekillik durumlarının önüne geçmek yerine sadece tersini almak yerine pseudo inverse önerilir)

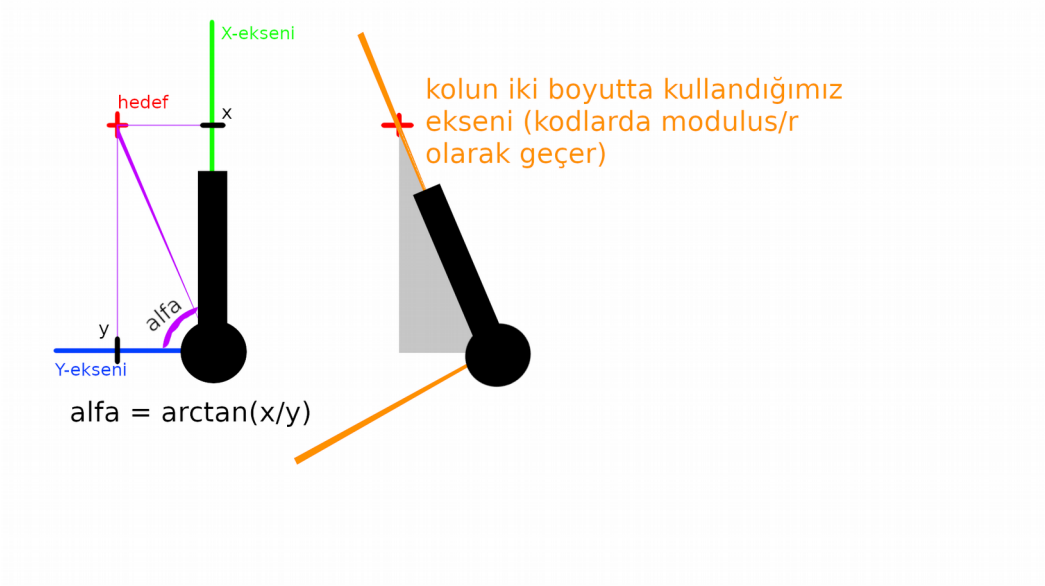
5. 4. adımda hesaplanan matris ile küçük adım çarpılır (δq = (J+) *(δX)) ve açı değişimleri vektörü elde edilir (J+: Pseudo inverse jacobian)

6. Açı değişimi anlık açıların üzerine eklenir ve yeni açı değerlerine erişilir. (qnew = q + δq)

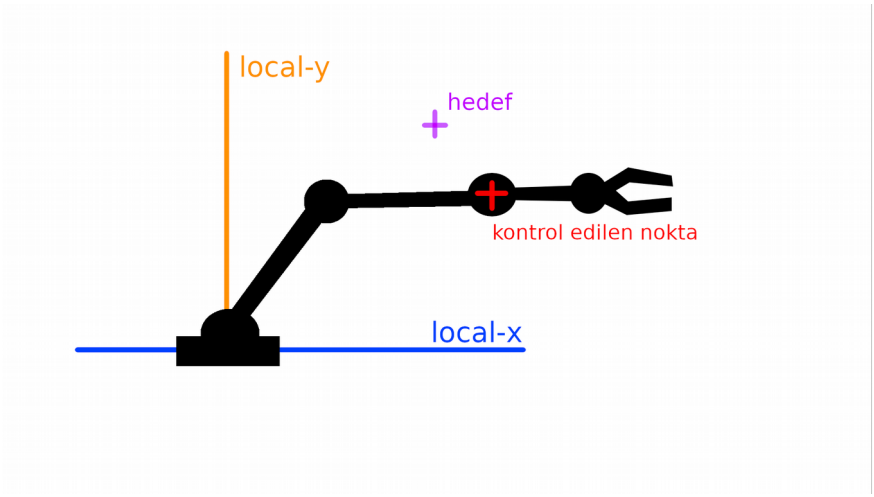
7. Yeni açı bilgileri ile motorlar/simülasyon beslenir

8. 1-7 adımları ΔX = 0 ya da ΔX <= 'belirlenen tolerans bandı' şartı sağlanana kadar tekrar edilir

Biz önce 20 sonrasında 21 robot kol üzerinde çalışacakken hem geliştirme sürecimizi hızlandırmak adına hem de başlangıçta lüzum olmadığını varsaydığımızdan ilk üç ekseni konumlandırma, diğer üç ekseni yönelimi kontrol etme işlerinde kullanmayı amaçladık. Bu yaklaşımla ters kinematikle ilk üç ekseni kontrol edecektik. Bunun sonucunda aslında robot kolun karmaşık bir transformasyon matrisini çıkarmaya gerek kalmadan işlem yapabilecektik. Zaten ilk ekseni direkt konum bilgisiyle kontrol etmek mümkün olduğundan problem -en azından benim algoritmamda- iki eksenli ve iki boyutta bir robot kola dönüşmüş oldu. Son üç ekseni ise ileri kinematikle sürerek operasyonları gerçekleştirebildik.



Yukarıdaki şekilde gösterildiği gibi robot kola tepeden baktığımızda hedef noktaya yönelimi sağlamak aslında çok kolay. Hedef noktanın koordinatları bilindiği için direkt arctanjant ile ilk eksenin açısı hesaplanabiliyor. Robot kol hedefe yöneldiğinde ise artık iki boyutta çalışmaya başlıyoruz. (NOT: Üstte yeşil ve mavi renkle gösterilen eksenleri global olarak tanımlayabiliriz. Robot kolun yönelimiyle değişmiyorlar, konumları sabit)



İki boyutta çalışma fikrinin gözünüzde daha iyi canlanabilmesi için yukarıdaki görseli ekliyorum. İlk eksen dönüşünü tamamladıktan sonra ters kinematik hesaplamalarındaki hedef atama da kontrol edilen nokta da aynı şekilde gösterildiği gibi iki boyut oluşturacak şekilde robot kolun kendi eksenleri üzerinden gerçekleştiriliyor. Jacobian yöntemi bu katmanda çalışıyor.

III. Kodların incelenmesi

Öncelikle belirtmeliyim ki bu raporda kodlar satır satır incelenmeyecektir çünkü zaten neredeyse her fonksiyon/metotta gerekli yorumlar kodlara eklenmiştir. Burada genel olarak hangi dosyanın ne yaptığı, hangi sırayla çalıştırılması gerektiği üzerinde durulacaktır.

Neler çalıştırılmalı:

1. roslaunch arm_21_gazebo arm_gazebo.launch

Robot kolun URDF'ini içeren simülasyonu başlatır. URDF'teki son sıkıntılar nedeniyle simülasyonu çalıştırmadan Gazebo içerisinde physics→gravity kısmından yerçekimini sıfırlıyoruz. Aksi halde robot kol düzlemde kayıyor.

2. roslaunch arm_control distinct_controllers.launch

Bu dosya gerekli kontrolcüleri yüklüyor. Kontrolcüler yüklendikten sonra Gazebodan 'play' tuşuna basıyoruz. Artık simülasyon ortamımız hazır.

3. sudo chmod 777 /dev/input/js0

Bu komutla birlikte joystickimize gerekli izinleri vermiş oluyoruz.

4. rosrun joy joy_node

ROS'un kendi joystick node'u. Burada joystick verileri okunuyor

5. rosrun jacobian inverse.py

Ters kinematik / ileri kinematik / alt yürür arası geçiş yapabildiğimiz node'u çalıştırır. Simülasyon üzerinden sürmek için bu kadarı yeterli.

6. rosrun serial_arm SerialMessage.py

Bu node algoritma ile elektronik kartlar arasındaki iletişimi sağlıyor.

Dosyalar:

.../jacobian/src/RoboticArm.py

Bu dosya genel hatlarıyla robot kolun niteliklerini içeren bir sınıf tanımlıyor. Eklem uzunlukları, çalışma uzayı sınırlandırmaları, eksenlerin açı sınırlandırmaları vs. bu sınıfta belirleniyor.

.../jacobian/src/RoverJacobianSolver.py

Jacobian yöntemini uygulamak için gerekli nitelik, dönüşüm metotları ve algoritmanın kendisini içeren bir dosya. Yine sınıf tanımlanıyor, daha sonra bu sınıf bir üst programda örneklenip çalıştırılıyor.

.../jacobian/src/inverse.py

RoboticArm ve RoverJacobianSolver dosyalarından sınıf örneklerini alıp işleyen, aslında tüm işi döndüren dosya. Joystick, gazebo ve serial node'u ile bu program üzerinden haberleşiliyor. Veriler gerektiği gibi işlenip ters/ileri kinematik ve alt yürür sürüşleri buradan gerçekleştiriliyor.

.../serial_arm/src/SerialMessage.py

Robot kol ve alt yürürü sürerken gerekli serial mesajın oluşturulması ve iletişimin kurulması bu dosya sayesinde mümkün oluyor.

IV. Kaynaklar

Aşağıya algoritmayı yazarken yararlandığım ve faydalı olabileceğine inandığım birkaç link bırakıyorum.

https://www.rosroboticslearning.com/inverse-kinematics

https://www.rosroboticslearning.com/jacobian

https://nrsyed.com/2017/12/10/inverse-kinematics-using-the-jacobian-inverse-part-1/

https://nrsyed.com/2017/12/10/inverse-kinematics-using-the-jacobian-inverse-part-2/

https://nrsyed.com/2017/12/17/animating-the-jacobian-inverse-method-with-an-interactive-matplotlib-plot/

NOT: Algoritmayla ilgili sorunuz olursa kenaremirtaha@gmail.com üzerinden bana ulaşabilirsiniz

Emir