INFINITE RECHARGE℠ at Home Rehberleri

FIRST® GAME CHANGERS℠ powered by *Star Wars: Force for Change*

2021 *FIRST*® Robotics Competition

FIRST ROBOTICS COMPETITION

FIRSTINSPIRES.ORG/ROBOTICS/FRC

İÇİNDEKİLER

GİRİŞ

Sahada yalnız olarak Yetenek Yarışması'nı tamamlamak *FIRST* Robotics Competition takımlarının daha önce yaşamadığı bir tecrübe. *FIRST*, Yetenek Yarışması'na nasıl hazırlanabilecekleri ve var olan robotları üzerinde yeteneklerini nasıl geliştirebilecekleri konularında takımlara yardım etmek için bu rehberleri hazırladı. Bu rehberler, tamamen opsiyonel olmakla beraber adım adım izlenecek bir süreç olarak tasarlanmamıştır. Bu rehberler 2021 Yetenek Yarışması görevleri için özel olarak hazırlanmıştır ancak takımlar gelecek sezonlar için benzer aktivitelerin nasıl geliştirilip kullanılabileceği konusunda düşünmek isteyebilir.

OTOYÖN GÖREVİ'NİN (AUTONAV CHALLENGE) ANALİZİ

OtoYön Görevi (AutoNav Challenge) tüm takımları karmaşık bir konu olan otonom sürüşü keşfetmeye davet eder. Otonom sürüşe bir giriş yapan bu rehber takımlara ilk otonom sürüş denemelerinde yardımcı olmayı hedefler.

Robot Pozisyonunun Takibi

Hangi yöntemin izleneceğini seçmeden ve bu yöntemle devam etmeden önce, iki yöntem için de büyük önem taşıyan temel bir konsept üzerinde durmamız gerekir. Bu konsept robotun pozisyonunun belirlenmesi ve takibidir.

Sahadaki sürüş esnasında robot pozisyonunu takip etmenin yollarından bazıları aşağıda listelenmiştir:

1. **Zaman** – Robot pozisyonunu ölçmenin en basit yolu hareketleri belirli zaman aralıklarında gerçekleştirmektir. Bu yol sensor gerektirmez ve her robot tasarımına uygulanabilir ancak bu yolla gerçekleşen asıl hareket ile ilgili hiçbir ölçüm yapılamaz. Ölçüm yapılamadığı için robotun gerçek hareketi, akü gerilimine, teker kaymasına ya da sürtünmeyi attıran veya azaltan bir etmene bağlı olarak değişebilir ve sonuç olarak robot beklenenden hızlı ya da yavaş hareket edebilir.
2. **Sensörler** – Takımlar, çeşitli sensörler kullanarak yapılan ölçümlerle, zamanlanan herhangi bir hareketin performansına kıyasla daha hassas bir hareket performansı elde edebilir. Bu sensörler, iki ana kategoride incelenebilir:
    - **İç Sensörler** – Bu sensörler robotun ya da parçalarının hareketlerini dışarıdan bir referans kullanmadan ölçer. Bu sensörler, zamanlanan hareketlere kıyasla çoğu hata unsurunu ortadan kaldırır ancak teker kayması ve sensor sapması gibi muhtemel hataların üstesinden gelemez. Takımların robot pozisyonunu takip etmek için kullanabilecekleri iki temel iç sensör türü şunlardır:
        - **Enkoder** – Şaftlardaki (bu örnekte, tekerlerin dönme miktarını) dönme miktarını ölçer. Enkoderler düz bir çizgi üzerindeki pozisyonu ve bir ölçüde de dönüşleri takip edebilir. (Çoğu robotun sürüş ekipmanı dönüş esnasında enkoderin güvenilirliğini düşüren teker kayması durumunu yaşar.)
        - **Jiroskop** – Açısal pozisyonları (daha doğrusu, daha sonradan integral işlemi uygulanarak pozisyon bilgisine çevrilen açısal hızları) takip eder. Çoğu jiroskopun içinde bütünleşik ivmeölçer de bulunmaktadır. (İçinde bir veya birden fazla jiroskop ve ivmeölçer bulunan sisteme Atalet Ölçüm Ünitesi {*İng. Inertial Measurement Unit [IMU]*} denir.) Teknik olarak ivmeölçerin ölçtüğü ivme verisinin çift integralinin alınması ile pozisyon verisi elde edilebilir fakat integral işlemini iki kez uygulamak veri üzerindeki gürültünün yükselmesine sebep olur. Çift integral işlemi sonrasındaki ölçüm, *FIRST* Robotics Competition özelinde kullanmak için çok gürültülüdür.

      Çoğu takım iç sensörleri sürüş için yeterli bulmaktadır ancak atış görevleri gibi hassasiyet gerektiren durumlar için dış sensörlere geçilmek istenebilir.
    - **Dış Sensörler –** Dış sensörler direkt olarak robotun çevresi hakkında bilgi toplar. Sahanın önceden bilinen özellikleri ile birleştirildiklerinde robotun saha üzerindeki konumunu belirlemek için kullanılabilirler. Laser uzaklıkölçer (LIDAR), IR uzaklıkölçer ve ultrasonik sensör gibi meseafe ölçen sensörler ve kameralar dış sensör olarak sınıflandırılabilir. Takımlar genelde saha üzerindeki konumlarını ve oryantasyonlarını hassas olarak belirlemeleri gerektiğinde (atış yapmak vb.) dış sensörlere başvurur ancak dış sensörlerin kullanım alanları bunlarla sınırlanmak zorunda değildir.

Bir Yöntem Seçin

Otonom robot sürüşünün iki yaygın yöntemi şunlardır:

1. **Tek Hareket Yöntemi** – Ana hareket daha küçük parçalara (genelde düz gidilen yörüngelere ve dönüş yapılan noktalara, bazen de tek başına yaylara) ayrılır. Her parçayı tek başına tamamlayacak bir kod geliştirilir. (İdeal olarak bu kodun başka bir parçada tekrar kullanılabilmesi için mesafe ve açı değerleri parametre olarak alınır.) Daha sonra bu kod parçaları bütünü tamamlayacak şekilde birleştirilir.
2. **Yörünge Planlama Yöntemi** – Tek veya birden fazla yaydan oluşan parçalar halinde kesintisiz bir yörünge hesaplamak için yörünge planlama yöntemini kullanın.

Tek Hareket Yöntemi

Tek Hareket Yöntemi genellikle belirlenen bir mesafede düz gitmeyi (ileri ve geri) ve belirlenen açılarda dönmeyi (sağa ve sola) sağlayan kodların geliştirilmesi ile başlar. Daha sonra istediğiniz herhangi bir yörüngeyi, bu üç kod bloğunu kullanarak daha küçük parçalara ayırabilirsiniz. Bu kod bloklarını oluşturuken çoğunlukla iki kontrol yaklaşımı kullanılmaktadır:

1. **Aç-Kapat Kontrol –** Aç-kapat kontol yaklaşımında açık ("tam hız" kullanılmasını gerektirmez) ve kapalı olmak üzere iki durum vardır. Basitçe motorları belirli bir değerde döndürmeye başlayın ve periyodik olarak hareketi durdurma koşuluna ulaşılıp ulaşılmadığını kontrol edin. Durdurma koşuluna ulaşıldığında, motorları durdurun. Bu yöntem pozisyon veya açı kullanıldığında çoğunlukla hedefin üssten aşımı (*İng. overshoot*) ile sonuçlanır.

TEKNO ÖNERİ: LabVIEW Teleop VI ile Timed ve Command şablonlarının hâlihazırda XXPeriodic fonksiyonları etrafında bir döngü içerdiğini hatırlayın. Kullanıcılar bunun gibi tamamlanması uzun zaman alan döngüleri bu fonksiyonların içine koymamalıdır. Bunun yerine, XXPeriodic metodunu döngünün bir iterasyonu olarak düşünün.

---

2. **PID Kontrol** –PID kontrol çıktıyı hataya (muhtemelen hatanın toplamına ve hatanın değişim hızına da) bağlı olarak dinamik şekilde belirler. Temelde, PID kontrolünü kırmızı ışığa yaklaşmakta olan bir araba olarak düşünebilirsiniz. Araba kademeli olarak durur, hedefe yaklaştıkça daha yavaş hareket eder.

Yörünge Planlama Yöntemi

Bu yöntem, genellikle robotun herhangi bir yörüngeyi izleyebilmesi için robottaki kontrol döngüsünün veya döngülerinin (çoğunlukla hız kontrolü) akort edilmesi ile başlar. Daha sonra, izlenilmesi istenen her yörünge, robotun birbiri ardına takip ederek tüm yörüngeyi izleyeceği "ara noktalar"a bölünür.

TEKNO ÖNERİ: Bu yöntem hakkında yardımcı olabilecek WPILib araçları hakkında bilgi WPI dokümanlarının Yörünge Oluşturma ve Yörünge Takibi kısmında bulunabilir. Adım adım bir örnek ise Yörünge Örneği'nde bulunabilir. LabVIEW kullanıcıları benzer bir işlev için şu kütüphaneyi kullanabilir.

Hepsini Birleştirin

Kullanacağınız yönteme karar verdikten ve izlemek istediğiniz yörüngeyi inceledikten sonra, aşağıdaki basamaklar birleştirilmelidir. Robot kodunuz için kullanılan programlama diline ve yazılımlara bağlı olarak izleyeceğiniz basamaklar değişiklik gösterebilir.

- **LabVIEW veya Timed Robot Auto Init –** Otonom bir program oluşturmanın en kolay yolu, programınızı üzerine kurgulayacağınız her kod bloğunu, içlerinde bir döngü olan birer metot ya da VI olarak yazıp gerektiğinde bu metotları sırayla çağırmaktır. Bu, LabVIEW'da Sequence Structure kullanılarak yapılabilir.

C++ ve Java'da ise oluşturulan kod bloklarının sırayla AutoInit() metodunun içinden çağırılması gerekir. Bu yol izlenebilecek en basit yoldur ancak önemli dezavantajlara (en azından C++ ve Java'da) neden olabilir. C++ ve Java'da kodunuzu bu şekilde oluşturmak robotun sürüşü esnasında başka fonksiyonların koda eklenmesini zorlaştırır. Robotunuz sürekli kod bloğunun içindeki döngüyü tamamlamaya çalışacağından, paralelde başka şeyler yapamaz. LabVIEW kodları dilin yapısı gereği paralel olarak çalışabildiğinden LabVIEW'da durum biraz daha farklıdır.

- **LabVIEW veya Timed Robot State Machine –** Kodunuzun sürüş esnasında diğer işlevleri de yerine getirebilmesi konusunda daha esnek olmasını istiyorsanız, "durum makinesi"ni (*İng. state machine*) deneyebilirsiniz. Durum makineleri hakkında internette birçok kaynak bulabilirsiniz. *FIRST* Robotics Competition özelinde otonom bir program yaratmak için kullanılacak durumun makineleri genelde aşağıdakilerden oluşur:
    - **Bir durum değişkeni –** Bu değişken içinde bulunulan durumu takip eder.
    - **Koşullu/Dallanan kod –** AutoPeriodic() metodunun içinde bir "switch" (LabVIEW için "case") veya bir dizi "if" ifadesiyle durum değişkenine bağlı olarak her durumda ne yapılacağını belirleyen yapıdır.
        - Her dal içinde bulunan kodun genellikle bir eylem (ör. motorların hızının belirlenmesi) gerçekleştirmesi ve sonrasında yeni bir duruma geçilip geçilmeyeceğini kontrol edilmesi gerekir. Basit otonom durum makinelerinde, akış ileri doğrudur, istenen mesafe ya da açı yakalandıktan sonra bir sonraki duruma geçilir. Daha karmaşık durum makinelerinde akış durumlar arasında atlayabilir ve akışın lineer olması gerekli değildir.
        - Her dalı bir kod bloğu olarak kullanın. Motorların hızını ayarların, bir sonraki duruma geçmeden hedefe ulaşıp ulaşmadığınızı kontrol edin ve kendi döngünüzü yazmak yerine AutoPeriodic() metodunun sağladığı ana döngüyü kullanın.
- **Komut tabanlı yazılım çatısı (*İng. Command-based framework)* –** Komut tabanlı yazılım çatısında yazılım çatısı sizin için bir durum makinesi oluşturur. Her kod bloğu bir komuttur, Init() ve/veya Execute() metotları motorlara komut gönderirken, IsFinished() metodu hedefe ulaşılıp ulaşılmadığını kontrol eder. Daha gelişmiş kontrol sistemleri elde etmek için PIDCommand ya da RamseteCommand'ı kullanabilirsiniz, bunlar bazı işlemlerini sizin için halledebilir. Bütün bileşenleri bir araya getirmek için, CommandGroups'u kullanın. Basit bir Otonom Programını Komut Tababanlı Yapmak dokümanında "eski komut tabanlı" kütüphane kullanılmaktadır ancak anlatılan prensipler "yeni" ya da "eski" komut tabanlı yazılım çatılarına uygulanabilir.

GÖRME İLE HEDEFLEME

FIRST Robotics Competition görme sistemleri ile hedeflerin bulunması hakkında çok fazla doküman bulunabilir fakat bu dokümanların içinde görme sisteminden gelen verinin nasıl kullanılacağına dair anlaşılır açıklamalar bulmak çok da kolay değildir. Görme sisteminizden almaya çalışmanız gereken veri ve bu veriyi robot kontrolünde nasıl kullanacağınız konuları hakkında genel bir özet bu rehberde sunulmuştur.

Adım 1: Amaçlarınızı Belirleyin ve Yönteminizi Planlayın

Problemin üzerinden geçin ve yazılıma ne yaptırmak istediğinize karar verin. Bu süreçte aşağıdaki soruları cevaplamak isteyebilirsiniz:

- Hedef sürekli olarak görme sisteminin görüş alanı içinde mi olacak? (Hedefe sürücünün kontrolünde mi yoksa yörünge tabanlı bir otonom sürüşle mi yaklaşacaksınız?) Ya da kodun herhangi bir hedefin tespit edilememesi durumunda ne yapacağını bilmesi mi gerekecek? Kodun hedefi bulması gerekirse bu sürüş sistemi ile mi yoksa bir taret ya da kameraya bağlı bir servo ile mi gerçekleştirilecek?
- İstenilen işlevi gerçekleştirebilmesi için robotun hedefe olan uzaklığını bilmesi gerekli midir? Gerekli ise, bu veri görme sisteminden mi yoksa başka bir yerden mi gelmektedir? Atış hızının/açısının hesaplanması veya robotun hedefe doğru ilerlemesi gibi durumlarda robotun hedefe olan uzaklığını bilmek isteyebilirsiniz. Başka durumlarda ise uzaklık bilgisine olan ihtiyaç farklı şekillerde (ör. uzaklığın sürücü tarafından ayarlanması ya da uzaklığın başka bir sistem tarafından ölçülmesi) çözümlenebilir.
- Belirlenen amaca ulaşmak için robotun özel bir oryantasyonda (ör. robotun keskin bir açıda atış gerçekleştirememesi, robotun bir oyun parçasını yerleştirebilmek için hedefe dik olarak pozisyonlanmayı gerektirmesi) mı olması gerekir? Öyleyse, bu limitlerin detaylarını tespit etmeye çalışın. Robotun oryantasyonu bu limitler dışında olduğunda, kod durumu nasıl idare etmelidir?

Adım 2: Hedefi Bulun

Görme sistemlerinde kullanılan üç yaygın yöntem aşağıda listelenmiştir:

1. **"Klasik bilgisayarla görme" –** Bu yaklaşım NI Vision veya OpenCV gibi makine öğrenmesi içermeyen yöntemleri kapsar. Bu yöntemler hakkında detaylı bilgi WPILib dokümantasyonunun Görüntü İşleme bölümünde bulunabilir.
2. **Makine Öğrenmesi** – Bu yöntem, hedeflerin etiketlenmiş örnek görüntülerini kullanarak yazılıma bir hedefin nasıl göründüğünü öğretir. Bu yöntem hakkında detaylı bilgiye WPILib dokümantasyonunun Makine Öğrenmesi bölümü incelenerek ulaşılabilir.
3. **Hazır çözümler** – Bu yöntem hedefleri hazır olarak alınabilen bir çözüm kullanarak bulur. Bu çözümler genelde kullanıcının bazı ayarlar yapmasına izin verir ancak temelde yaşattıkları tecrübe baştan bir yazılım tasarlamaktan çok farklıdır. Bu çözümlere örnek olarak Chameleon Vision, Limelight, Opensight, ve PhotonVision/Gloworm verilebilir. Çözümlerden bazıları özel donanım gerektirmekte olup bazıları da kendilerine özel donanımın takımlar tarafından montaj edilmesi için tasarlanmıştır. WPI'ın geliştirdiği GRIP, bu yöntemle klasik bilgisayarla görme arasında melez bir çözümdür. GRIP, hazır çözümlerin çoğunda olduğu gibi bir arayüz sunar ancak sonradan kullanıcı tarafından üzerinde oynama yapılabilen bir OpenCV kodu da oluşturur.

Adım 3: Görme Sistemi Çıktısı

Adım 1'deki soruların cevapları ve görme sisteminin kendine özgü özellikleri görme sisteminin oluşturacağı çıktıyı ve çıktıya nasıl ulaşıldığını belirleyecektir. NetworkTables, çıktıya ulaşmanın en çok kullanılan yoludur. Görme sisteminin en tipik çıktıları arasında aşağıdakiler bulunur:

- **Yatay sapma ölçümü** – Bu ölçüm bazı durumlarda bir açı değeri olsa da sıklıkla görüntüde tespit edilen hedefin piksel cinsinden sapmasının ölçümüdür. Bu ölçüm direkt olarak kullanılabileceği gibi robotun hareketini belirleyecek bir açının hesaplanmasında da kullanılabilir.
- **Düşey sapma ölçümü** – Bu ölçüm de bazı durumlarda bir açı değeri olsa da sıklıkla görüntüde tespit edilen hedefin piksel cinsinden sapmasının ölçümüdür. Çoğunlukla şurada anlatılan yaklaşıma benzer bir şekilde hedefe olan uzaklığın hesaplanmasında kullanılır. (Verilen örnek en bilgisini kullansa da örnektekine benzer prensipler ve hesaplamalar kullanılabilir.)
- **Hedefin eninin ve/veya boyunun ölçümü –** Bu ölçümler neredeyse her zaman piksel ölçümleridir. Kullanılmaları durumunda, genelde şurada anlatılan yöntemde olduğu gibi hedefe olan uzaklığın hesaplanmasında kullanılırlar. (Verilen örnekte hedef eni kullanılmıştır ancak hesaplar çok kolay bir şekilde yükseklik kullanımı için değiştirilebilir.)
- **Robot pozu** – Bazı görme sistemleri, SolvePNP poz tahmin teknikleri kullanarak robotun hedefe göre olan pozu hakkında bir tahmin sunar. Poz hem pozisyon hem de açı ölçümlerinden oluşmaktadır.

Adım 4: Çıktıyı Robotunuza Komut Vermek için Kullanın

Görme sisteminden hedef hakkında bilgi alındıktan sonra bu bilgi robota komut vermek için kullanılabilir. Aşağıda sunulanlara ek olarak, Limilight kullanmasanız bile Limelight dokümantasyonunda bulunulan birçok “Vaka Çalışması” bölümünü faydalı bulabilirsiniz.

Görme Bilgisini Açısal Komutlara Çevirin

Adım 3’te anlatıldığı gibi, görme sisteminiz hedefin görüntüdeki yatay pozisyonu hakkında bir çıktı üretmelidir. Robotunuzu hedefe yöneltmek için, bu çıktıyı kullanarak açısal komutlar oluşturun. Bu aşamada genellikle bir kontrol döngüsü kullanılır. Bunu yapmanın iki yolu aşağıdaki gibidir (Üçüncü bir alternatif, “Yörünge planlama” bölümünde paylaşılmıştır.):

1. **Döngüyü görme ile kapatın** – Bir yönlendirme komutu oluşturmak için sürekli olarak direkt görme sisteminin ürettiği ölçüm değerlerini kullanın. Bu yöntem kameranın saniyedeki kare hızı görüntü kararlılığı ile sınırlıdır. Dönüş çok hızlı gerçekleşirse yeni bir görüntü alamadan hedef üstten aşılabilir ya da kamera sarsıldığı için kötü bir görüntü alınabilir. Kameranın kapasiteleri ile uyumlu dönüş hızları sağlandığında bu yöntem uygulanabilir en kolay yöntemdir.
2. **Döngüyü bir jiroskop ile kapatın** – Görme sisteminden gelen bilgiyi dönme açısını hesaplamak için kullanın sonrasında da jiroskop ile geri besleme yaparak bu açıya doğru dönüşü gerçekleştirin. Bu yöntem, jiroskopların hızlı tepki sürelerinden faydalanarak, robotu görme sisteminin belirlediği açıya, görme sisteminin hedefin ortalanıp ortalanmadığını kontrol etmek için yeni bir görüntü alması gereken andan önce döndürmüş olur (belirli bir hata payıyla). Bu yöntem anlatılan ilk yöntemden biraz daha karmaşık olup hızlı görme sistemleri için gerekli olmayabilir.

TEKNO ÖNERİ: Bu bölüm robotun sürüş mekanizmasının kontrolüne odaklanmış olsa da robotun bir bölümünü, taret gibi, oynatmak için de aynı prensipler geçerlidir. İkinci yöntemin kullanılması için jiroskopun hareket edecek olan robot bileşenine takılması gerekir.

Görme Bilgisini Sürüş Komutlarına Çevirin

Eğer amaçlanan robotu hedeften belirli uzaklıktaki bir noktaya ulaştırmaksa, robotunuzun sürüş komutlarını oluşturmak için uzaklık ölçümleri kullanılabilir. Kullanılan “açı” ve “jiroskop” ifadelerinin sırasıyla “uzaklık” ve “enkoder” olarak değiştirilmesiyle, yukarıdaki yöntemler bu amaç için de kullanılabilir.

Açı ve Sürüş Komutlarını Birleştirin

Bir hedefe doğru hareket ederken, sürüşün doğru açıyı koruyarak gerçekleşmesini isteyebilirsiniz. Bunu yapmak için, hesapladığınız açısal ve sürüş komutlarını birbirine ekleyin.

TEKNO ÖNERİ: İki komut döngüsünün üretebileceği maksimum değerler birbirlerine eklendiğinde sonucun sistemin maksimum çıktı değerini aşma (ör. PWM kontrolcüleri veya % VBus modunun kullanıldığı durumlarda 1.0'ı aşma) ihtimali varsa, bu ihtimalin gerçekleşip gerçekleşmediğini kontrol edin ve iki tarafın çıktısını da büyük olan komutun sistem maksimumunu aşmayacağı şekilde bölün. Aksi takdirde, istenilen dönme komutu sağlanamaz. (ör. İki komutun toplamı, sol taraf için 1.0 sağ taraf için 2.0 çıktı değerini üretiyor. Bu durumda sistem sadece 1.0 çıktı değeri sağlayabileceği için, iki taraf da aynı komutu almış olur ve bir tarafın diğerine oranla iki kat hızlı dönmesi gerekirken, iki taraf da aynı hızda döner.)

Görme Bilgisini Atış Komutlarına Çevirin

Oyun parçaları bir hedefe atılıyor veya fırlatılıyorsa, mesafe ölçümleri robot sürüşü yerine oyun parçasının atılışı ile ilgili komutlara çevrilebilir. Robotun tasarımına göre değişebilen bu komutlar çoğunlukla hız, açı ya da hız ve açının beraber olduğu komutlardır. Fizik içeren yaklaşımlar bu ölçümler için iyi bir başlangıç noktası belirleyebilir ancak bu problemin 100%'nün fizik denklemlerine oturtularak çözülmesi zor olabilir. Bunun yerine toplanan deneysel veri bir denkleme oturtularak ya da bir taramalı tabloya (İng. look-up table) (ara değer hesabı ile ya da olmadan) girilerek uzaklık ölçümleri (veya görme sisteminin başka çıktıları) atış komutlarına çevrilmiş olur.

Alternatif Yöntem – Yörünge Planlama

İstenen ve şimdiki pozisyon arasında kesintisiz bir robot yörüngesi yaratmanın alternatif bir yolu yörünge planlama araçlarını kullanmaktır. Bu yöntem robot pozunu hedefe göre saptayabilme yetisini gerektirir. Kod, robotun bulunduğu konumu başlangıç, ulaşılmak istenen pozisyonu (genelde asıl hedeften biraz sapan bir konum) ve açıyı da bitiş noktası olarak kullanarak istenen poza doğru kesintisiz bir yörünge planlar. Robot, planlanan bu yörüngeyi olduğu gibi izleyebilir ya da yörüngeyi izlemeye başlayıp yörünge üzerindeyken görme sisteminden güncel pozisyonunu okuyarak yeni yörüngeler hesaplayabilir.

SÜRÜCÜ SEÇİMİ

Robotun sürülmesi ve idare edilmesi, FIRST Robotics Competition oyunlarında gerekli olan önemli bir yetenektir. Spor, sanat, müzik ve eğitim gibi birçok alanda, yetenek, çalışmanın ve becerinin uygulanmasının bir ürünüdür. Robot yapma sürecinin yoğunluğu nedeniyle çoğunlukla gözden kaçsa da sürücü seçimi rekabetçi çoğu takımı diğerlerinden ayıran önemli bir noktadır. Bu rehber sürücülerin seçiminde yardımcı olacak bazı hususları içermektedir.

Sürücüler, iyi iletişim kurabilmeli, robot operasyonu konusunda yeteneklerini gösterebilmeli ve yarışma sezonu boyunca çalışarak tecrübe kazanmaya kendilerini adayabilmelidir. Sürücülerin seçimi konusunda bazı teknikler ve dikkat edilmesi gerekenler aşağıdaki gibidir:

- **Beceri mi Tecrübe mi? –** Sürücüleri değerlendirirken, bir adayın ham becerisine ve zaman içinde edindiği tecrübesine bakın. Beceri, bir adayın robot kontrolüne olan doğal yatkınlığı, bir adayı diğerlerinden ayıran önemli bir unsurdur. Yüksek beceriye ancak düşük tecrübeye sahip bir aday, düşük beceri yüksek tecrübeye sahip bir adayınkine eşit veya daha iyi bir performans sergileyebilir. Spor, sanat, müzik ve eğitimde de olduğu gibi yüksek beceriye sahip bir kişi sadece becerisine bel bağlayamaz, bu kişi çalışmak ve gelişimi hakkında düşünmek zorundadır. Sürücü seçim sürecinde, becerikli ve yeteneklerini geliştirmek için gereken zamanı ve çabayı harcamaya hazır adaylar aranmalıdır. Robot ile antrenman yapmak için ne kadar zamanınız olacağını göz önünde bulundurun. Antrenman için çok kısa bir zamanınız varsa, becerisi yüksek bir aday iyi bir seçim olabilir. Antrenman için daha fazla zamanınızın olması hâlinde, daha az becerikli ancak iletişim yetenekleri ve soğukkanlılık ile öne çıkan bir aday zamanla daha iyi bir seçenek olabilir. Sürücü becerilerini test etmek için en güncel robotunuzun tamamlanmasını beklemeniz gerekli değildir. Önceki senelerden bir robot, test için yapılmış bir sürüş robotu ya da güncel robotun sürüş için yeterli olan kısmı sürücülerinizin becerilerini değerlendirmek için kullanılabilir.
- **İletişim –** Yeni fikirleri, stratejileri ve gelişim fırsatlarını etkili ve saygılı bir şekilde paylaşabilme yetisi tüm sürücüler için kritiktir. Etkili sürücüler aynı zamanda robotun sorunlarını ve iyileştirilebilecek yönlerini diğer takım üyelerine aktarabilir. Sürücüler, sürüş takımdakilerin "kimya"ları uyuştuğunda iletişimin daha kolay sağlandığını ve fikir alışverişinin daha kolay yapıldığını fark etmektedir. Bu uyum, genelde iyi ilişkilerin göstergesidir ve takım çalışması aktiviteleri gibi ortak tecrübelerle daha da geliştirilebilir. Sürüş takımı üyeleri, rahatsızlık duydukları şeyleri ve başarıları birbirleri ile güven içinde paylaşabilmeli ve birlikte öğrenmek ve gelişmek için beraber çalışmalıdır. Maç sonrası bir kısa toplantı yapmak, geliştirilebilecek yönleri tartışmak ve başarıları kutlamak için muhteşem bir yoldur.
- **Soğukkanlılık –** Bir *FIRST* Robotics Competition turnuvası içerisinde yoğun rekabet ve zaman baskısı bulunduran karmaşık bir ortamdır. Etkili sürücüler baskı altında sakin kalmayı başarır ve durumları sakince değerlendirdikten sonra en iyi aksiyonu alırlar. Sürücüleri değerlendirirken, kötü giden bir duruma kolayca ayak uydurabilen ve hızlıca alternatif planlar üretebilen kişilere odaklanmaya çalışın. Dikkatli bir planlama süreci beklenmedik senaryoların gerçekleşmesi ihtimalini ve bu senaryonun gerçekleşmesi durumundaki paniği azaltacaktır ancak turnuvalar boyunca beklenmedik şeylerin yaşanması muhtemeldir ve sürücüler bu beklenmedik durumlara hızlıca adapte olmalıdır.
- **Kurallar –** Kuralları okumak ve anlamak sadece sürüş takımının görevi olmasa da sürüş takımının her üyesinin kurallar hakkında bilgi sahibi olmalıdır. Sürüş takımının her bir üyesi özellikle kendi görevlerine dair kurallara hâkim olmalıdır. Başarılı sürüş takımları, Oyun Kılavuzu'nun bir kopyasını gittiklerini her etkinlikte yanlarında bulundurarak ve Team Update güncellemelerini ile Q&A'yi sürekli takip ederek kurallarla ilgili herhangi bir duruma karşı sürekli hazırlıklı olur. Bazı maçların sonucu tek bir ceza puanı ile değişmektedir, bu nedenle kuralları iyi anlamak çok önemlidir!

SÜRÜŞ PERFORMANSINI ARTTIRMA

Neredeyse her FIRST Robotics Competition oyununun standart öğesi oyun içinde oyun parçalarının alması ve bu oyun parçaları ile puan toplanması döngüsüdür. Bu döngü, puan aldıktan hemen sonra oyun parçası alma ve tekrar oyun parçası ile tekrar puan alma, bir "seri" olarak da adlandırılır. Çoğu oyunda, bir takımın maç içindeki zamanının çoğu bu seriyi gerçekleştirmek ile geçtiğinden, bir seri için harcanan zamanın optimum değere ulaştırılması elzemdir. Bu rehber nasıl ve neyin üzerinde antrenman yapılabileceği ve seri zamanının nasıl analiz edilip daha iyiye taşınabileceği hakkında bazı fikirler sunar.

Bu rehbere ek olarak, benzer konular hakkında yazılmış topluluk kaynaklarına da göz atmak isteyebilirsiniz. Takım 610'un Seri Optimizasyonu ve Takım 2168'in Sürüş Takımı Kılavuzu adlı kaynaklar, bu rehber için ilham veren iki ana kaynaktır.

Antrenman Nasıl Yapılır?

Sürücülerin yeteneklerini geliştirmek için tekniklerin ve süreçlerin belirlenip uygulanması robot performansının en optimum seviyeye taşınması için önem arz eder. Sürüş yeteneklerinin gelişmesi için en önemli yollardan biri antrenman yapmaktır. Antrenman için robotun tamamlanmasına gerek yoktur. Önceki senelerden bir robot, test için yapılmış bir sürüş robotu ya da güncel robotun sürüş için yeterli olan kısmı asıl robot tamamlanıncaya kadar antrenman için kullanılabilir. Antrenman yapmak tek başına yeterli olsa da antrenmanı nasıl yaptığınız da önemlidir. Sürüş antrenmanını daha etkili yapmak için kullanılabilecek tekniklerden bazıları aşağıda paylaşılmıştır:

- **Temelden Başlayın ve Üzerine Koyarak İlerleyin** – Yeni bir şey öğrenirken, temelden başlamak en iyi yöntemdir. Başarılı sürücüler robotları sürmeye ve görevleri tamamlamaya genelde yavaşça başlarlar. Robot kontrolüne, görevlere ve gerçekleştirilen eylemlerin sıralarına aşina oldukça her görevi daha hızlı bir şekilde tamamlamayı deneyebilirsiniz. Sürücünün daha fazla yetkinlik ya da kendine güven kazanmak için belirli bir hızda ya da seviyede antrenman yapmaya devam etmesi tamamen olağan bir durumdur. Sürücü gitgide kas hafızası oluşmaya başlar ve yapılan eylemlerin çoğu alışkanlık hâline gelir. Bu sürücülerin yarışma maçlarındaymış gibi antrenman yapmaya başlamalarını ve kabiliyetlerini geliştirmelerini sağlar.
- **Destek Kullanmaktan Çekinmeyin –** Destek verici bileşenlerin kullanılması yeni bir şey öğrenme yöntemlerin arasındadır. Destek verici bileşenlere örnek olarak aşağıdakiler verilebilir:
    - Bir dönme sekansının nasıl tamamlanacağını öğrenmek için yeri oklarla işaretlemek.
    - Yerlere veya saha bileşenlerine yapıştırılan bantları konumlama ve hassasiyet için kullanmak.
    - Saha bileşenlerini farklı renklerde yaparak öğrenme esnasında bileşenler arasındaki farkı vurgulamak.
- Sürücüler zamanla destek veren bileşenlerin haricinde de ipuçları yakalamaya başlar. Bunun örnekleri, robotun belirli bir uzaklıktan sürücüye nasıl göründüğü ya da tamponların bir saha elemanı ile hizalanması olabilir. Bu noktada, destek veren bileşenler aşamalı olarak kaldırılabilir.
- **Ustalıkla Sürün** – Takımların bunu söylemek için değişik ifadeleri vardır, "Yavaş Sorunsuzdur, Sorunsuz Hızlıdır." veya "Zaman Paha Biçilmezdir." bu ifadelerden bazılarıdır. Bu deyişlerin hepsinin ortak bir maksadı vardır: sürüşe sade güç ve sade hız olarak yaklaşmak yerine ustalık ile yaklaşın. Bir oyun parçasını sürekli saha bileşenlerine çarpmak anlık olarak tatmin edici olabilir ancak bu her zaman verimli sonuçlar doğurmaz ve gereksiz robot hasarına, boşa geçen zamana ve cezalara neden olabilir. Rekabetçi sürücüler, iyi oynanan beş saniyenin bütün maçın gidişatını değiştirebileceğinin farkındadır.
- **Bir Amaç ile Antrenman Yapın** – Sürücüler genellikle robotu bozmamaları gerektiğini ve böyle bir durumda sorumluluğun kendilerinde olduğunu düşünürler. Bu düşünce yapısı sürücüleri robotlarını tam potansiyellerinde kullanmamaya iter ve bu nedenle robotun tasarım ve inşa sürecindeki zayıf noktaların erkenden fark edilmesi şansı ortadan kalkar. Tam aksine "çalmış gibi sür" tekniği robottaki olası problemlerin etkinlikten önce fark edilmesine yardımcı olur. Takımınızın teknik birikimine robotunuzu her

zaman en iyi noktaya getirebileceği konusunda güvenin ve robotunuzu ustalıkla sürerken limitlerini zorlamaktan kaçınmayın.

Neyin Antrenmanı Yapılmalı?

Elinizde olanlarla gerçek sahaya en benzer antrenman ortamını oluşturmaya çalışın. Oyun Sahası'nda bulunan Yerleşim ve İşaretleme diyagramı ile Takım Bileşenleri inşa rehberi bu konuda size yardımcı olabilir.

- **Basitten Başlayın –** Yeni bir robot ile antrenmana başladığınızda, basitten başlayın. Robotunuzun en temel fonksiyonlarına odaklanın. Maç esnasında kullanmayı hedeflediğiniz fonksiyonları deneyin ve nasıl çalıştıklarını gözlemleyin. Sürücüler robota ve robot kontrollerine karşı aşinalık kazanmalıdır. Tuhaf hissettiren ya da zorluk çıkaran durumları gözlemlemeye çalışın, bunlar ileride üzerinde çalışılabilecek şeyler olabilir.

- **Genel Alıştırmalar** – Müzisyenler yeni bir enstrümanı çalmayı öğrenmenin etkili bir yolunun bu yeni enstrümanda ölçülerin ve belirli bir sıradaki notaların nasıl çalınacağını öğrenmekten geçtiğini bilir. Ölçüler, müzisyenlerin hızlarının, el becerilerinin ve kas hafızalarının artmasını sağlar. Ölçüler enstrümanlar için neyse, genel alıştırmalar da robotlar için odur. Genel alıştırmalar, slalom, engelli veya değişik düzenlerdeki sürüşleri içerebileceği gibi oyun parçalarının yerden alınmalarını ve bırakılmalarını, puan almak için oyun parçalarının antrenman yapılarına atılmalarını da içerebilir. Genel alıştırmalar sürücü seçimde de kullanılabilir. Bu alıştırmalar ışığında sürücülerin becerileri keşfedilebilir ve sonrasında da antrenmanlarla sürücülerin yetenekleri geliştirilebilir.

- **Seri Denemeleri –** Oyun parçası alımı, sürüş ve puan kazanma döngüsünün antrenmanını yapın. Bu antrenman bir noktadan sonra çok tekrara düştüğünüz hissini verebilir ancak sürece ne kadar hâkim olursanız, stres altında bu süreci tekrarlamak o kadar kolaylaşır. Serileri gerçekleştirirken süre tutmayı veya antrenmanın video kaydını almayı deneyin, böylece farklı tekniklerin sonucu nasıl etkilediğini gözlemlemek ve anlamak için nesnel bir veri elde etmiş olursunuz. Farklı yörüngeler, maç esnasında karşılaşabileceğiniz durumlar ve eylemler için de antrenman yaptığınızdan emin olun.

- **Hassasiyet Ekleyin** – Sürücü hassasiyetini test etmeyi ve arttırmayı hedefleyen genel alıştırmalar deneyin. Sürekli 100% hassasiyet sağlayıncaya kadar optimum veya dar bir yörünge sürüşünü oluşturmak için nesneler kullanın ya da bir oyun parçasının yerden alınışını ve bırakılışını tekrar edin.

- **Değişkenler Ekleyin –** Sürücüler maç esnasında olabilecek çoğu şey için hazırlıklı olmalıdır. Maç esnasında oluşabilecek "şöyle olursa" senaryolarını düşünün ve bu senaryoların en muhtemel olanları için planlama ve antrenman yapmaya başlayın:

    - Optimum bir yörüngenin ya da atış noktasının engellenmiş olması durumunda ne olacak?
    - Robotunuzun bazı fonksiyonlarının maç esnasında bozulması durumunda ne yapılacak?
    - Defansa karşı ne yapacaksınız?
    - Kullandığınız alanı partnerlerinizle nasıl paylaşabilirsiniz?
    - Planladığınız yörüngeleri ters yönde kullanabilir misiniz?

Serileri Analiz Edin ve Geliştirin

Tek başına antrenman yapmak serilerin iyileştirilmesine yardımcı olur ancak seri antrenmanları sırasında nelerin çalıştığının ve çalışmadığının analizini yapmak gelecekte yapılabilecek geliştirmelerin belirlenmesini sağlar. Bunu yapmanın bazı yolları aşağıdaki gibidir:

- **Robot kontrolü hakkında düşünün -** Robotunuzun nasıl kontrol edildiği hakkında düşünün. WPILib düz, açılı ve tank sürüş modlarını hazır olarak içermektedir. Bunlar arasında olan farkları WPILib dokümanyasyonundan öğrenebilirsiniz. Farklı sürüş modları topluluk içerisinde paylaşılmaktadır ve bu modlar sizin sürücünüz için daha uygun ya da daha az uygun olabilir. Değerlendirilmesi gereken başka bir yön ise sürüşün robot tabanlı mı saha tabanlı mı olacağıdır. Saha tabanlı sürüşte, sürücü oyun kumandasını oynattığında robotun oryantasyonunu düşünmek zorunda değildir. Bu tarz sürüşte oyun

kumandasının ileri itilmesi robotu oryantasyonundan bağımsız olarak sürücüden uzağa götürür. Saha tabanlı sürüş, genelde mecanum ya da swerve drive sürüş sistemleri kullanılarak oluşturulmuş çok yönlü (*İng. omni-directional*) hareket kabiliyetine sahip robotlarda kullanılır ancak diferansiyel sürüş ile de kullanmak mümkündür. Sürücülerin yapılacak herhangi bir değişime alışmaları zaman alacağından sürüş kontrolleri olabildiğince erken belirlenmelidir. Kontrollerin kolayca uygulanabilir olduğundan emin olun. İlk denemelerde birkaç yanlış butona basmakta problem yoktur ancak bu tarz hatalar kontrol yerleşimi konusunda sürücünün rahat olmadığının göstergesi olabileceğinden bu durumun yakından takip edilmesi önerilir.

- **Kontrol ekipmanınızı değerlendirin –** Üzerinde düşünülmesi gereken bir başka konu da sürücülerin kullandığı kontrol ekipmanıdır. Farklı takımlar farklı joystick, oyun kumandası ve direksiyon/pedal kombinasyonları ile başarılı olmuşlardır. Sürücüler için kullanması kolay ve doğru bir hareket alanına sahip olan opsiyonu bulun.
- **Serileri analiz edin –** Gerçekleştirildikleri gibi serileri analiz edin. Süreyi üç kategoriye bölün: oyun parçası alımı, sahada sürüş ve oyun parçasının bırakılması ya da puan alınması. Bu üç kategori içinden hangisi sizin seriniz içindeki en büyük orana sahip? Her kategoriyi nasıl daha iyi hâle getirebilirsiniz? Hızınızı arttırabilecek tasarım ya da yazılım değişiklikleri yapılabilir mi? Boş harcanan zaman ya da boş yere yapılan bir hareket var mı? Hassasiyet seri verimliliğinizi nasıl etkiliyor?
- **Otomasyon ekleyin –** Seri süresini azaltmanın bir yolu da otomasyon eklemektir. Örnek olarak şunlar verilebilir:
    - Görme ile Hedefleme bölümünde bahsedildiği gibi, bazı hedeflere ve konumlara göre hizalanma konusunda bilgisayarla görmeden yardım alın.
    - Robot mekanizmalarını otomatik olarak pozisyonlayın.
    - Robot içindeki oyun parçalarının hareketini otomatik hâle getirin.