Bizden 3 tane **kritik tasarım raporlarını** özellikle yazılımını inceleyerek anladıklarımızı yazmamızı istemişsiniz. Ben de olabildiğince KO-DRİVE, KOÜ-MEKATRONOM ve TALOS isimli 3 takımın raporunu incelemeye çalışıp KO-DRİVE arabasını özetleyerek anlatıp diğerleriyle olan farklarını daha çok not aldım hangi algoritma kullanılır neden diğerleri değil de o tercih edilmiş gibi.(Çok resmi gibi ya da çok saçma duruyor olabilir :(nasıl yazacağımı bilemedim pek)

OTONOM ARAÇ KTR RAPORU İNCELEME KO-DRİVE TAKIMI

İlk olarak otonom sürüşte görüntüyü alıp işlemeyle aracın gideceği rotayı belirleyip karşısında çıkacak tabelalar, engeller ya da çeşitli yol sürüşlerine göre aksiyonları alıp hareket edecek şekilde ayarlamışlar. Aracın bu hareketleri de PID ile kontrol edilmiştir. bunların hepsi başka sürüş algoritmaları ile sağlanmıştır.

1. Trafik İşareti Tespit Algoritması

Bu algoritmada amaç görüntüdeki nesnelerin sınıfları ve konum bilgileriyle birlikte bulma yaklaşımıdır. Üç aşamadan oluşur : nesnelerin bulunduğu potansiyel bölgelerin belirlenmesi, öznitelik çıkarımı ve sınıflandırma. Bunlarda YOLOP modeli kullanılmaktadır. Fakat bu modele alternatif olması açısından farklı derin öğrenme ağlarının geliştirmelerini de testleri devam etmektedir.(Makine öğrenmesini amaçlamışlar) Derin öğrenmeyse yapay sinir ağları ve benzeri öğrenme algoritmaları kapsayan çalışma alanıdır. YOLOV4 algoritmasını kullanmaya karar vermişler. (Biraz baktım YOLO nun baya bir çeşidi var v1 v2 v4 v5 gibi farklı modelleri varken neden v4 diye YOLOV5 daha fazla gpu istemesinden ve diğer sistemlerle uyumlu olmayabileceğinden dolayı kullanılmamış bir de python yazılımı gerekiyormuş galiba o yüzden tercih edilmemiş) Bu sistemlerin hepsinde de lidar ve kamera sensörü kullanılarak görüntü alınmış.YOLO ile incelemişlerdir.

2. Otonom Sürüş Başlangıç-Bitiş Noktası Tespit Algoritması

Başarılı bir rota belirlenebilmesi için belirli alana bir eşik değeri uygulanıp beyaz renkli şeritlere göre rota belirlenmiş en veya uzunluk verilerine göre de yüksek olanları başlangıç ve bitiş olarak belirlenmiş. Gazebo simülasyon ortamındaki başlangıç ve bitiş noktaları belirlenmiştir.

3. Şerit Takibi ve Sürülebilir Alan Algoritması

Lidar ve kamera sensörleri ile alınan veriler şerit sınırlarını belirlemişlerdir. Derin öğrenme temelli algoritma olan YOLOP algoritmasını kendi sistemlerine optimize etmişler. Gerçek hayatta da kullanılabilmesi için farklı ortam koşullarında da algoritmalarını denemişler.

4. Sürüş Kontrol Algoritması

Önceki şerit takip, nesne algılama ve rota belirleme algoritmalarından çıkan verileri düzenleyip kullanır. Kamera konumunu 0-0 yapıp referans oluşturarak şerit tespitiyle ilerleyecektir. Örnek olarak bir viraj döndükten sonra kamera oluşturulan rotaya göre kendini 0-0 noktasına alıp ondan sonra yoluna devam eder. Engelleri LIDAR sensörü ile belirlenmiştir. Daha genel haliyle bütün veriler LIDAR IMU ve GPS ile belirlenip PID kontrol mekanizmasına gönderilmiş. PID ise hatalara tepki verip oranını azaltan geri beslemeli kontrolü sağlar.3 ana bölümden oluşup sistemin olmazsa olmazıdır.KTR raporunda bu kadar

bilgi verilmiş başka uygulama ya da yazılım birimleri geçmiyor. Simülasyonlar ve hız gibi bilgilerde Gazebo ve rviz üzerinden halledilmiş

Şimdi gelelim Talos adlı araca bu araçta aynı nesne algılama sensörlerini kullanmıs kullsndıkları sensörler : LİDAR kamera, GPS ve IMU. Tek tek amaçları ise :
-Lidarıdan uzaklık ve RGB değerlerini ROS(Robot Operating System) üzerinden alarak pyzed ile işlenmiştir.
-Kameradan RGB derinlik alınmaktadır ve genel tabela verileride burdan alınmaktadır.
-GPS sensörüyle ise CAN hattı üzerinden alınıp enlem boylam verileri alınarak rota oluşturmadan kolaylık sağlamaktadır.
-IMU verileri de kamera üzerinden pyzed ile aktarılmıştır daha çok ivme açısal hız ve manyetik alan gibi verileri kullanıp hesaplar.
KOÜ-MEKATRONOM arabasında ise aynı sistemleri kullanmasına rağmen bunları pythonun pyzed değil openCV kütüphanesi yardımıyla gerçekleştirmiştir. Araştırdığımda pyZedin sadece zed kamera türüyle kullanıldığı ve derinlik hesaplamasıyla öne çıktığı belirtiliyordu.Kısaca eğer derinliğe yönelmek istiyorsak pyZED kullanmamız gerekiyor diye anladım. OpenCV ise daha genel ve nesne tespiti için kullanılıyordu ve genel olarak renk hassasiyeti için kullanıldığından KOÜ-MEKATRONOM kameradan alınan görüntüleri renk uzayı türü olan HSV ya çevirmeyi daha iyi yapabilmesi de openCV kullanma sebebi olabilir

Başka bir farklılıkta hepsi VOLO algılama algoritmasını kullanırken KOÜ-MEKATRONOM diğerleri gibi YOLOV4 yerine YOLOV5 kullanmıştır. Burda da araca göre istenileni kullanmışlar aslınd TALOS arabası en optimize ve çok fazla gpu kullanımı istemediğinden VOLOV4 ü kullanırken KOÜ-MEKATRONOM optimizasyondan çok sisteme uyumu ve en az hata payına odaklandığından VOLOV5 i kullanmıştır. Ayrıca TALOS günün her saati ışık farklı vursa az çok olsa dahi tabelaları net bir şekilde algılayabilmesi için görüntü toplayıp data augmentation yaparak veri setini geliştirdi.

TALOS arabasında hep derin öğrenme yapay zeka temeli kullanılmaya çalışılmış ve bu şerit takip algoritmasınada yansımış simülasyon ortamından görüntü alıp LabelIMG ve makesense.ai uygulamaları üzerinde segmentasyonu yapılmış ve yapay zekanın sonradan karşısına çıktığında kendisinin bu sorunun üstesinden gelebilmek amaçlanmış. Şerit çizgileri arasındaki sürülebilir alan tespit edilerek güvenli sürülebilir bölge diğer alanlardan ayrılmıştır. Bu alan maskelenerek şerit takip algoritmasına verilir. Sapma değerleriyle ayarlanarak da şerit takip algoritması desteklenir. (KOÜ-MEKATRONOM şerit takip etme hakkında çok bilgi bulunmamakta çok fazla bir şey yazamadım)

KTR da geçmeyen bazı özel algoritmalarını da inceleyip buraya ekliyorum

1. Pure Pursuit (Saf Takip Algoritması)

Aracın mevcut konumuna göre önceden belirlenmiş yolun üzerindeki bir hedef noktayı seçip, o noktaya ulaşmak için gerekli direksiyon açısını hesaplar.

Avantajları:
- Basit ve anlaşılır bir yapısı vardır.
-Az işlem gücü gerektirir, küçük sistemler için uygundur.

Dezavantajları:
- Keskin virajlarda sapmalar ve overshoot (fazla dönme) görülebilir.
-Yüksek hızlarda stabilite azalabilir.

2. Stanley Algorithm

Araç, hem **yoldan sapmayı (lateral error)** hem de **yön hatasını (heading error)** aynı anda düzelterek yolu doğru şekilde takip etmeyi hedefler.

Avantajları:
- Hem pozisyon hem yön hatasını dikkate alır, bu da daha kararlı bir sürüş sağlar.
- Hız değişimlerine karşı nispeten dayanıklıdır.

Dezavantajları:
- Düşük hızlarda küçük titreşimler (oscillation) yapabilir.
- Yüksek hızda lateral hata çok büyürse kontrol zorlaşabilir.

3. PID Controller (Proportional–Integral–Derivative)

Görevi:
Belirlenen hedefle mevcut durum arasındaki farkı (hata) minimize etmek için geri besleme kontrolü uygular.

Avantajları:
- Uygulaması kolaydır, çoğu kontrol sistemine entegre edilebilir.
-Hata geri beslemesi sayesinde sistem kendini düzeltir.

Dezavantajları:
- Karmaşık araç dinamiklerinde yetersiz kalabilir.
-Değişken hız koşullarında performans düşebilir.

4. MPC (Model Predictive Control – Model Öngörülü Kontrol)

Görevi:
Aracın dinamik modelini kullanarak gelecekteki davranışını tahmin eder, belirli bir zaman dilimi boyunca en uygun kontrol sinyallerini hesaplar.

Avantajları:
- Araç dinamiklerini ve fiziksel kısıtları (örneğin maksimum direksiyon açısı) hesaba katar.
- Virajlarda ve yüksek hızda çok kararlı sonuçlar verir.

Dezavantajları:.
-Gerçek zamanlı sistemlerde gecikme riski olabilir
-Model doğru tanımlanmazsa performans düşer.

Genel anlamıyla böyle özet sunum gibi bir ödev hazırladım. Doğru bir şekilde yapmışımdır inşallah.