# T.C

# KONYA TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ

## BİLGİSAYAR MÜHENDİSLİĞİ UYGULAMASI – 2

## (BİTİRME PROJESİ-2) FİNAL RAPOR FORMU

|  |  |
| --- | --- |
| Öğrencinin Adı- Soyadı | Berkcan Barut |
| Numarası: | 181213098 |
| Danışmanı Adı Soyadı: | ERSİN KAYA |
| Sınav Tarihi: | 17.06.2022 |
| Projenin Konusu: Robotaksi – Otonom Sürüş Sistemi | |
| **DÖNEM İÇİ YAPILAN ÇALIŞMALARIN ÖZETİ**  Projenin ikinci kısmında, birinci dönemde gerçekleştirmiş olduğumuz object detection işleminde kullanmış olduğumuz yapay zeka modelinin versiyonunu, modelin daha başarılı sonuçlar elde etmesi amacıyla değiştirdik. İlk olarak Yolov5 versiyonu yerine başarı olasılığından ötürü Yolov4 versiyonunu tercih ettik. Yolov4 model eğitiminde Darknet framework’ünden yararlandık. Model eğitiminden önce Yolov4’e ait olan cfg dosyası üzerinde ilgili vermiş olduğumuz nesne tespitinde görev alan class’lara göre yeniden üzerinde düzenlemeler yaptık ve daha sonrasında datasetimizi eğitime soktuk. Yolov4 ile nesne tespiti eğitimi esnasında Google Colab’ta karşılaşmış olduğumuz problemler için birtakım pratik kodlar üreterek bunları eğitimimize dahil ettik. Bu sayede Google Colab’ta geliştirmiş olduğumuz eğitim kesintisiz bir şekilde gerçekleştirilmiş oldu. Eğitim süresince elde etmiş olduğumuz ağırlık dosyalarımızı belirlenmiş bir epoch değerine geldiğinde Google Drive hesabımıza kaydetme işlemlerini gerçekleştirdik ve kaydetmiş olduğumuz ağırlık dosyalarından en iyi çalışan ağırlık dosyasını test ortamında sınayarak en optimum sonuçlar veren ağırlık dosyasını Object Detection’da kullanmak üzere seçtik.  Geçtiğimiz dönem boyunca eğitmiş olduğumuz yapay zeka modelimizin performansında problemler meydana gelmişti. Levhaların bazı durumlarda yanlış tanımlanmasını önlemek için eğitimimizi güçlendirmek üzere araştırmalar yapıldı. Bu araştırmalara sonucunda, eğitimlerimiz önceden eğitilmiş (pretrained) ağırlık dosyalarımız üzerinden yapıldı. Bu şekilde modelin optimum performansının bulunması amaçlandı. Yapay zeka modelimizin simülasyon ortamına entegresi aşamasında OpenCV kütüphanesi ve modelin eğitiminin sonucunda elde etmiş olduğumuz config dosyası(.cfg) ile en iyi performansı veren ağırlık dosyası(best.pt) Darknet Framework’ünden yardım alarak yazmış olduğumuz entegre yazılımında kullanılmıştır.  Ağırlık dosyamızı Webots Ortamında örnek aldığımız bir videoda ve daha sonrasındaysa Webots ortamında simülasyon aktifken kameradan alınan frame’lere göre Object Detection başarısını sınadık ve yapay zeka modelimizin başarı yüzdesini ölçtük.    Yeni oluşturulan ağırlık dosyasını kullanarak simülasyon ortamanından aldığımız görüntü verilerindeki başarı yüzdeleri incelenip, doğruluğu kontrol edildi.  Bu görüntülerden bazı örnek görseller alındı ve rapora eklendi.    Eğitmiş olduğumuz yapay zeka modelinin webots’a entegresi aşamasında, simülasyon haritası üzerinde aracın tespit etmiş olduğu nesnelere göre aracın hangi davranışları sergileyeceğinin tespitinde karar ağacı yönteminden faydalanmaya karar verdik. Karar ağacı yöntemi kullanılırken kendi oluşturmuş olduğumuz veri setinden faydalandık. Veri setini oluşturma esnasında giriş vektörü olarak herhangi bir yön belirten levhalarımızdan faydalanırken, çıkış vektörü olarakta kendi belirlemiş olduğumuz çıkış etiketlerinden faydalandık. Sonuç olarak eğitmiş olduğumuz karar ağacı modelinin ağırlık dosyasını(.pkl) ileride kullanmak üzere elde ettik. Elde ettiğimiz bu karar ağacını simülasyondaki aracın kamera verisinden gelen görüntüye göre yönlendirme çıktısı kullanıldı. Bu üç adet yön bilgisine göre (sola dönüş, sağa dönüş, ileri) araçtaki tek şerit takip sistemi yeniden kodlandı. Bu sayede aracın levhaların belirlediği doğru yönde hareket edebilmesi sağlandı. Bu algoritmaların tamamı teknofest robotaksi yarışmasının geçmiş senedeki haritası üzerinde gerçekleştirildi.    Yukarıdaki görselde görünen harita üzerinde yapılan tüm algoritma ve object detection işlemleri test edilerek simülasyon üzerinde otonom sürüş sistemi tamanlandı.    Araca verilecek yön bilgisi için tasarlanan karar ağacı modeline ait verilerden elde edilen örnek kısmı yandaki görselde gösterilmiştir  Çift şerit takip algoritması, aracın yönlendirilmesinde oluşturduğu zorluklardan dolayı yeni ve daha efektif çözüm için çalışmalar yapıldı. Çalışma sonucunda Karar ağacı kullanılarak alınan kararlara göre eski çift şerit takibi tek şerit takibi olarak güncellendi bu sayede otonom sürüşün daha tutarlı olması sağlandı. Araca verilecek yöne göre hangi şerit için takip edileceği bir karar algoritma sistemi yardımıyla gerçekleştirildi.    Aracımız trafik levhalarına göre dönme yönüne karar verdikten sonra, çift şeritli yolda bulunduğu şeridi algılayarak yeniden karar sistemini çalıştırır. Karar sistemi, araca gelen yön ile bulunduğu şerit arasındaki bağlantıya göre bir çıkış üretir. Buradaki mantık, araca gelen yön bilgisi ile aracın takip ettiği şerit aynı (sağ,sol) tarafta ise araç bulunduğu durumda kalır. Hali hazırda takip ettiği şeride göre çalışmaya devam eder. Fakat yönler ters ise, araç kendini tabeladan gelen yöne göre şerit değiştirme algoritmasını çalıştırarak, uygun şeride geçme işlemini başarıyla gerçekleştirir. Bu şekilde araç kendi her zaman uygun şeritte tutmayı amaçlar.  Araç şerit değiştirme algoritmasını kullanarak kendini uygun şeride aktarır. Haritanın ilgili bölümlerinde trafik levhalarına uygun şekilde keskin dönüşler bulunmaktadır. Araç bu dönüşleri takip ettiği şeridin kameradan kaybolması sonucu algılamaktadır. Bu durum yaşandığı anda araç üzerinde bulunan lidar verisi kontrol edilmeye başlar. Lidar verisinden herhangi bir değer dönmesi durumunda, aracın etrafında tabela olduğu anlaşılmakta ve ilgili dönüş algoritmasının çalışması gerçekleşmektedir. Bazı olumsuz durumlarda da şeridin kamera görüntünde gözükmemesi gerçekleşmektedir. Bu durumların hepsinin dönüş olarak algılanmaması için lidar verisinden dönen değer kontrol edilmektedir.  Araç görevlerinden biri olan durak alanında durup yolcu alması için, yolcu alma/indirme algoritmaları yazılmıştır. Araç kamerasından gelen görüntüde durak tabelası tanındıktan sonra, şerid takip sistemi aynı şekilde devam ederken, lidar verisi sürekli olarak okunur. Lidar verisinden gelen değerin belirlenen uzaklığın altına düşmesi sonucu araç, yolcu alma/indirme durumuna geçer. Bu şekilde görevini tamamlamış olur.  Kameradan gelen görüntü veri üzerinde trafil levhaları tanınmaktadır. Bu nesne tanıma içinde trafik ışıklarıda mevcuttur. Araç kırmız ışıkta durma ve bu stop durumundan çıkmak için yeşil ışığa ayarlanmıştır.  Aracımız son görevi olarak uygun park alanına, park işlemini gerçekleştirmektedir. Bu arada, levha tanıma sistemi sürekli olarak çalışmaktadır. Araç uygun park alanlarından birine hedeflenerek, park alanına aracı yönlendirmeye çalışır. Araç kamerası ile tespit edilen park levhasının görüntüdeki konumun araç kamerasının orta noktasına olan piksel bazındaki uzaklığı hata oranı olarak kabul edilmiştir. Bu hata oranı PID kontrol sistemine gönderilerek, araç yönlendirilmesi sağlanır. Tüm işlemler sonucunda araç, kendini uygun park alanına konumlayarak durdurur.  **KAYNAK ARAŞTIRMASI**  Proje süresince geliştirecek olduğumuz yapay zeka modelimizi eğitmede kullanacak olduğumuz datasetin verilerini (.jpg) toplarken hangi kaynaklardan yardım alabileceğimiz hakkında araştırmalar yaptık. Araştırmalarımız sonucu, Google Haritalar, Google Resimler, Open Images Dataset v6, Kaggle ve Roboflow platformları araştırıldı.  Modelin, dataset de bulunan verilerin (.jpg) üzerinde ilgili objenin tespitini başarıyla yapabilmesi için gerekli olan koordinat bilgilerini içeren ve bulmuş olduğu objeleri sınıflandırıp birbirlerinden ayırmasında gerekli olan etiket bilgisinin oluşturulması aşamasında Make Sense ve LabelImg platformları araştırıldı.  Hazırlanılan dataset de verileri çeşitlendirip çoğaltarak dış ortama aktarabilmek amacıyla Roboflow platformu incelendi ve araştırıldı.  Yapay zeka modelinin eğitilmesi ve test edilmesi için kullanıcılara yüksek performans ve donanım desteği sunan Google Colaboratory ortamı detaylıca incelendi ve araştırıldı.  Yapay zeka modelimizin eğitilme sürecinde çeşitli eğitim modelleri arasından Yolo modelinin kullanılmasında karar kılındı. Kullanmış olduğumuz yolo modelinin versiyonunun tercihinde ise test ortamında bize en yüksek başarı yüzdesini sunan ve modelin simülasyon ortamına entegresinde bize en fazla performans ve kullanım avantajı sağlayan versiyon 4’ü(Yolov4) tercih ettik.  Eğitmiş olduğumuz yapay zeka modelinin simülasyon ortamına entegresi işleminde, OpenCV kütüphanesi ve yapay zeka modelimizle entegre için yazmış olduğumuz yazılım arasında ki bağlantıyı kurmamızda Darknet Framework’ünden yararlanıldı.  Eğitilmiş modelimizin simülasyon ortamında Object Detection işleminin kararının sonucunda sergileyecek olduğu davranışın belirlenmesi amacıyla, kod performansının ve okunabilirliğinin arttırılarak kodumuzun daha ‘clean code’ olabilmesi için ve buna ek olarak if-then yapısının öğrenilmesinde oldukça başarılı olan ‘Karar Ağaçları Yöntemi’nden faydalanıldı.  Haritadaki dönüş kısımlarındaki keskinlik ve şerit sisteminin kamera görüntüsündeki veriden çıkması sonucu oluşan olumsuz durumların önüne geçmek için dönüş algoritmalarına bakıldı. Bu dönüş algoritmalarında kullanılmak üzere zamanlama ve karar sistemleri araştırıldı.  Harita üzerinde bulunan çift şerit yol üzerinde, gerekli durumlarda aracın ilgili şeride geçmesini sağlayan şerit değiştirme algoritmaları araştırıldı. Bunun için gerekli sistemler ve temel mantıkları incelendi.  Aracın haritadaki tüm görevlerini tamamlaması için, otonom park sistemleri hakkında araştırmalar yapıldı. Araç park levhasını tanımlayarak uygun şekilde hareketini gerçekleştirmesi için, park sistemlerine bakıldı. Yönlendirme ve karar verme sistemleri araştırıldı. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJEDE KULLANILAN MATERYAL VE METOTLAR**  Datasetlerin verilerini oluştururken Google Haritalar, Google Resimler, Open Images Dataset v6, Kaggle ve Roboflow’da bulunan hazır datasetlerden faydalanıldı.  Etiket bilgisi bulunmayan veriler için MakeSense platformunda verilerimize koordinatlar üzerinde işaretleme yaptık ve daha sonra modeldeki sınıfları birbirlerinden ayırabilmek için sınıf etiketleri atanıldı.  Hazırlamış olduğumuz datasetlerimizdeki verileri çoğaltmak ve dış ortama aktarabilmek amacıyla Roboflow platformundan yararlanıldı.  Modelin eğitimi aşamasında Roboflow’un sunmuş olduğu API bağlantısı ile datasetimizi Google Colab ortamına aktarıldı ve eğitim aşamasında Roboflowda hazırlamış olduğumuz datasetler kullanıldı.  Modelin eğitilmesi için Yolo versiyonlarından Yolov4 kullanılmıştır. Modelin eğitimi ve simülasyon ortamına entegresi aşamalarında kullanılmak üzere Darknet Framework’ünden yararlanıldı.  Simülasyon ortamında Object Detection sonucunda yapılacak olan davranışların belirlenmesinde ‘Karar Ağaçları Yöntemi’nden yararlanıldı.  Yeni haritadaki çift şeritli yolda, aracın ilgili şeride geçmesi için şerit değiştirme algoritması gerçekleştirildi. Bunun için timer ve statik yön değeri kullanıldı. Bu işlemdeki parametreler belirlenirken, haritadaki yolun genişliği baz alındı.  Haritada bulunan keskin dönüş kısımlarına uygun olarak, statik dönüş algoritması gerçekleştirilidi Bu sistem de yine timer ve statik yön değeri kullanıldı. Aracın dönüş süresi için belirlenen süre tamamlandıktan sonra, dönülen yön değerinin tersindeki şerit takip sistemi aktif hale getirildi.  Araç görevlerinden olan, yolcu alma/indirme kısmında levha tanıma sisteminden gelen durak tabelası uyarısına göre lidar sistemi kullanıldı. İlgili şerit takip sistemi devam ettirilirken, aracın durması için lidar sisteminden gelen veri değeri kullanıldı. Bu uzaklık değeri durak alanına göre belirlendi.  Trafik ışıklarına uygun hareketin gerçekleştirilmesi için, yine levha tanıma ve lidar sistemi ortak olarak çalıştırıldı. Kırmızı ışık, levha tanıma sistemi tarafından tespit edildikten sonra lidar verisine göre uygun konumda aracın durdurulması sağlandı. Bu kırmızı ışıktaki stop durumunun bozulması için yeşil ışığın tespit edilmesi hedeflendi. Bu sayedi trafik ışıklarına uygun araç hareketin yapılması sağlandı.  Aracın son görevi olan, park sistemi için otonom park sistemleri ve algoritmaları araştırıldı. Kameradan gelen görüntü verisine uygun olarak park alanı belirlendi ve o alana parabolik hareket yardımıyla araç yerleştirilmesi sağlandı. Son duruma geldiğinde lidar sisteminden gelen veriye göre belirlenen uzaklıkta aracın durdurulması sağlandı.   |  | | --- | | **DÖNEM SONU HEDEFLERİNİN DEĞERLENDİRİLMESİ**  Proje geliştirme sürecinde öğrenmiş olduğumuz makine öğrenmesi, yapay sinir ağları ve yapay zeka uygulamaları eğitimleri tamamlanmıştır. Multi object detection (tekli-çoklu nesne tespiti ) işlemlerinin başarıyla yapılması için seçmiş olduğumuz bazı trafik işaret ve levhalarının görüntülerinin tanımlanması ve etiketlenmesi işlemlerinin yapılarak datasetlerin doğru bir şekilde oluşturulması gerçekleştirilmiştir. Datasetlerinin istenilen ölçütlere uygun hazırlanmasıyla bir sonraki işlem olan görüntü üzerinde nesne tespiti ve sınıflandırılmasını yapacak yapay zeka modelinin oluşturulması ve eğitilmesi aşamasına geçilmiştir. Google Colab’ın bizlere sunmuş olduğu yüksek donanım ve performans içeren platformunda projeler halinde geliştirilip test edilerek öğrenimleri başarıyla gerçekleştirilmiştir. Eğitmiş olduğumuz modelin simülasyon ortamına entegresi işlemi başarıyla gerçekleştirilmiştir. Bu işlemler ile birlikte dönem sonu için planan hedefler tamamlanmıştır.  Günümüzde kullanılan araçlara göre otonom araç teknolijisi hızla ilerlerken bu konuda yeni otonom araçlar piyasaya sürülüyor. İnsan gücü ile değil yapay zeka gücü kullanılmaya başlandığı için böyle bir projede olası ticaret araçlarından taksilerin günümüz hayatında kullanılabileceğini gösterdik.  Otonom araçların 5 seviyesinden en üst seviye olan tam otomasyon olarak simülasyon ortamında deneyerek gerçek hayata en yakın şekilde oluşturulmuştur.  Araç lidar ve kamera aracılığı ile önce kameradan alınan verilerle şerit takibi yapıyor . Şerit takibi yaparken lidar verisinden gelen bilgileri kullanarak sağında, solunda, karşısında olan nesnelere olan uzaklığı bildirebiliyor bu sayede levhaları lidar ile uzaklık tespiti yapılıp yapılan tespite göre aksiyon alınıyor. Görülen levhalara belli bir mesafe içerisinde levha tanıma yapılıyor bu görülen levhalara göre aracın dönüş açısı verilerek dönüş aksiyonu alınıyor. Durak tabelası tanımlandığında durağa girip yolcu alıyor ve bir sonraki durakta durak tabelası tanımlandığında yolcuyu indiriyor. Son aşamada lidar ve levha tanımadan gelen veriler yardımı ile park yapılıyor.  P | | **KAYNAKLAR** | | * What is YOLO algorithm? | Deep Learning Tutorial 31 (Tensorflow, Keras & Python) : https://www.youtube.com/watch?v=ag3DLKsl2vk * YOLO Object Detection (TensorFlow tutorial) : https://www.youtube.com/watch?v=4eIBisqx9\_g * Bilgisayarlı Görü: YOLOv4 ile Nesne Tanıma :   https://www.udemy.com/course/bilgisayarl-goru-yolov4-ile-nesne-tanma/learn/lecture/22439242#questions   * Deep Learning ve Python: A'dan Z'ye Derin Öğrenme :   <https://www.udemy.com/course/deep-learning-ve-python-adan-zye-derin-ogrenme-5/learn/lecture/11742030#overview>   * YOLOv4 in the CLOUD: Build and Train Custom Object Detector (FREE GPU):   <https://www.youtube.com/watch?v=mmj3nxGT2YQ&t=2001s>   * Train a custom YOLOv4 detector online ( Free GPU ) :   <https://www.youtube.com/watch?v=SCAgktactKE&t=193s>   * How to Train YOLOv4 on a Custom Dataset in Darknet:   <https://www.youtube.com/watch?v=N-GS8cmDPog>   * Object Detection Using YOLO v4 on Custom Dataset | Practical Implementation:   <https://www.youtube.com/watch?v=yGMZOD44GrI>   * Counting Objects Using YOLOv4 Object Detection | Custom YOLOv4 Functions with TensorFlow:   <https://www.youtube.com/watch?v=jDwC5m7c7BU&t=1007s>   * YoloV4 Object Detection using OpenCV-Python only tutorial, Pre-trained model:   <https://www.youtube.com/watch?v=1aL6tewfxFY>   * YOLOv4 Object Detection with TensorFlow, TensorFlow Lite and TensorRT Models (images, video, webcam):   <https://www.youtube.com/watch?v=iPwepy-SVCQ>   * YoloV4 ile Object Detection(Nesne Tanıma) Modeli Nasıl Eğitilir ?:   <https://www.youtube.com/watch?v=fGwjtIM2B00>   * YOLOv4 Custom Object Detection Tutorial: Part 1 (Preparing Darknet YOLOv4 Custom Dataset):   <https://www.youtube.com/watch?v=sKDysNtnhJ4>   * YOLOv4 in the CLOUD: Build Object Tracking Using DeepSORT in Google Colab (FREE GPU): <https://www.youtube.com/watch?v=_zrNUzDS8Zc> * Single Lane Detection : <https://www.hackster.io/kemfic/curved-lane-detection-34f771> * Highly Curved Lane Detection : <https://www.mdpi.com/2076-3417/10/7/2372/htm> | |  | |