TEKNOFEST İSTANBUL

HAVACILIK, UZAY VE TEKNOLOJİ FESTİVALİ

ROBOTAKSİ – BİNEK OTONOM ARAÇ YARIŞMASI

KRİTİK TASARIM RAPORU

ARAÇ VE TAKIM ADI: T-Electro V1 / T-Electro Team

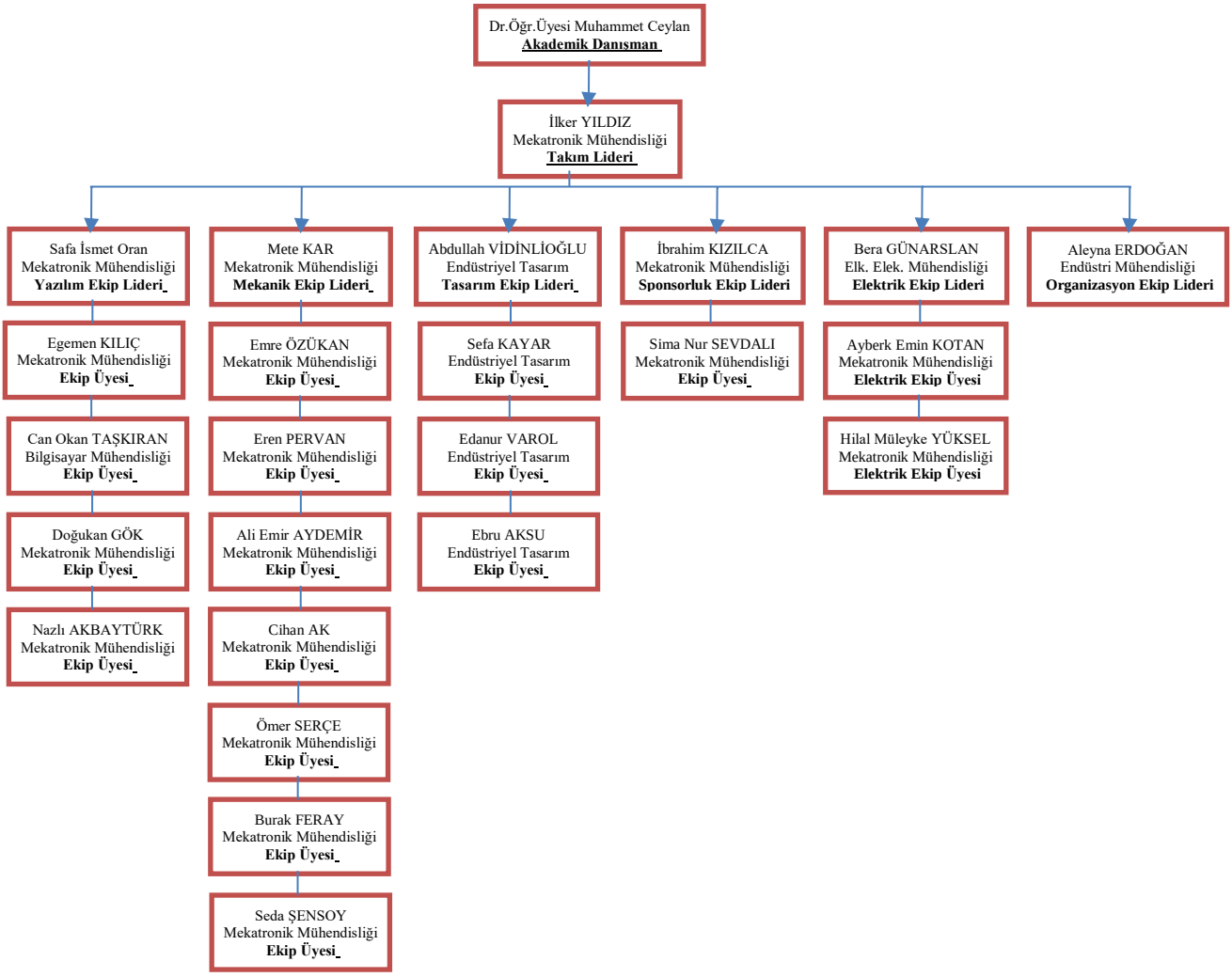
TAKIM KAPTANI: İlker YILDIZ

DANIŞMAN ADI: Dr. Öğr. Üyesi Muhammet Ceylan

İÇERİK

1. Takım Organizasyonu

T-Electro Team 1 akademik danışman, 1 takım lideri, 5 ekip lideri ve 19 ekip üyesinden oluşmaktır. Takımın bünyesinde Mekanik Üretim, Yazılım ve Kodlama, Tasarım, Organizasyon Planlama ve Elektrik ekibi olmak üzere 6 alt birim bulunmaktadır. Mekanik ve üretim ekibi şase ve rollcage, fren, süspansiyon ve kabuk üretimini gerçekleştirecektir.Tasarım Ekibi aracın dış kabuk ve iç tasarımını gerçekleştirecektir. Elektrik ekibi aracın motor, motor sürücü, batarya, şarj ünitesi ve araç kontrol ünitesinin tasarımını ve üretimini gerçekleştirecektir. Kodlama ve yazılım ekibi aracın otonom sürüş ve haberleşme sistemi üzerine çalışacaktır. Organizasyon ve Sponsorluk ekipleri aracın sponsorluk ve pazarlama kısmında yer alacaktır.



2. Ön Tasarım Raporu Değerlendirmesi

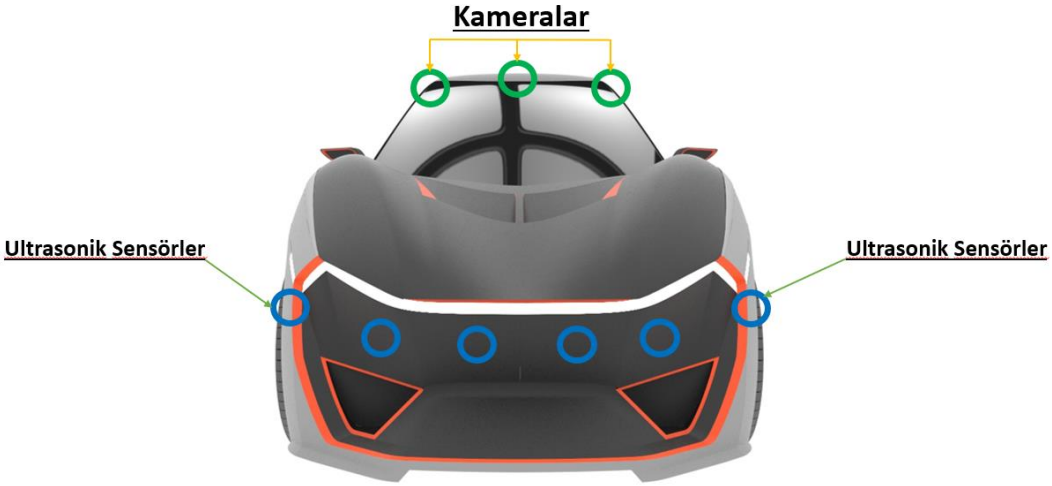
Ön tasarım raporunda belirttiğimiz araç özelliklerine sadık kalarak aracımızın tasarımını tamamlamış bulunmaktayız. Yarışma kurallarının analizini tekrardan gözden geçirerek aracımızdan yarışma kuralına aykırı olabilecek herhangi bir sorun çıkmaması için bütün kontrollerini tamamlamış bulunmaktayız. Kurallara uygun yolcu alma ve indirme yapabilmek için tabela tanımlamasını sistem üzerinden Faster R-CNN COCO modeli ile yapmaktayız. Aracın doğru rotayı takip etmek ve şerit içerisinde düzgün hareket etmesi için Nvidia'nın öğrenme algoritmasını kullanmaktayız. Bu model ve algoritma ile ön tasarım raporunda belirttiğimiz 1500 puan sınırına ulaşmaktayız. Aracın şerit takibi yapması ve tabelalara göre hareket etmesi sonucunda aracımız pisti sorunsuz bir şekilde bitirecek ve park kısmına giriş yapacaktır. Aracın özelliklerinde herhangi bir değişiklik yapılmamıştır. Aracın dış kabuk tasarımın rüzgar testlerinde ön taraftan çok fazla rüzgar yediğini fark ederek aracın ön tarafında gerekli düzeltmeler yapılmıştır. Ön tasarım raporunda belirttiğimiz sensörler yapılan testlerde başarılı sonuçlar vermiştir bundan dolayı alternatif başka sensörler kullanılmasına gerek duyulmamıştır. Aracın kontrol ünitesi ve otonom sürüş algoritmaları ön tasarım raporunda belirttiğimiz şekilde kullanılmış olup bir değişiklik yapılmamıştır. Ön tasarımda planladığımız bütçe aracımızın eksik bileşenlerinin tamamlanması için yeterli olmuştur ve bütçemizde bir değişiklik olmamıştır.

3. Araç Fiziksel Özellikleri

T-Electro takımı otonom kurallarına uygun şekilde bir araç tasarlamaktadır. Frenleme için uygun lineer motor seçilip montajı için testlerini yapmaktadır. Direksiyon sistemi için aracın dönüşlerde maksimum açısı hesaplanmıştır. Hesaplanan ölçüler dahilinde kremayer dişli direksiyon kutusu üretilmiştir.Otonom parkur için direksiyon kutusu bir adet nema 34 motor ile desteklenmiştir. Kontrol ünitesine gönderilen değerlerin motora iletilmesiyle aracın hızlanması sağlanmıştır. Otonom sürüş için 3 adet kamera aracın ön ve yan taraflarına konumlandırıldı.Aracın etrafındaki engellerin tanınmasına yardımcı olmak için atmış derecelik açılar ile dört tanesi önde , iki tanesi yan taraflara konumlandırılmış ultrasonik sensörler yerleştirildi.Üretilecek aracın genel ölçüleri 3100mm uzunluk, 1150mm genişlik ve 1100mm yükseklik olarak planlanmış ve modellenmiştir. Aracın motoru 72V 1,5 kW HUB motor olup jantın içinde yer almaktadır. Aracın ana şase ve kabuk kısmında karbonfiber kompozit malzeme kullanılmaktadır. Aynı zamanda araçta kullanılan birimlerden motor, motor sürücü, batarya yönetim sistemi, yerleşik şarj birimi ve araç kontrol ünitesini ekip olarak tasarlayıp üretecektir. Batarya ünitesinde Li-ion piller kullanılmakta olup kapasitesi 3kW'dır.

4. Sensörler

Aracımızın ön tarafında üç adet kamera ve altı adet ultrasonik mesafe sensörleri bulunmaktadır. Ultrasonik sensörlerin görme açısı 30 derece olduğu için 6 adet konumlandırarak toplamda 180 derecelik görüş açısı sağlanmıştır.Bunlardan dört tanesi aracın ön tarafına eşit aralıklar ile yerleştirilip, kalan iki tanesi ise aracın her iki kenarına altmış derecelik açılarla konulmuştur. Ortada bulunan kamera yol çizgisini takip ederken kenarlarda ki 2 kamera ise trafik levhalarını ve trafik ışıklarını algılamaktadır.



5. Araç Kontrol Ünitesi

Araç kontrol ünitesinde entegre haberleşme sistemi ile oluşturulmuş CANBUS ağı kullanılarak; sürücü ile araç arasındaki fiziki arayüzden alınan verilerde dahil olmak üzere, tüm verinin ilgili sistemlere dağıtılması planlanmaktadır. Ağ üzerinde bulunacak kontrol sisteminde, sahip olduğu dahili donanımlar sebebiyle mikrodenetleyici olarak STM32Fxx serisi kullanılması öngörülmektedir.

Motor kontrol algoritması için motor üzerindeki hall effect sensörleri ile geri bildirim sağlanacaktır. STM32Fxx içerisinde gömülü kontrol yazılımı FOC(Field Oriented Control) algoritması kullanılarak oluşturulması hedeflenmektedir. Kontrol algoritmasının tune edilmesinde motorun karakteristik parametreleri kullanılacaktır. FOC tork ve üç fazlı AC elektrik motorlarının hızını kontrol etmek için değişken frekanslı sürücülerde kullanılan bir yöntemdir. Bu yöntem, motor akımlarının ölçülmesini ve art arda makinenin rotoru ile dönen bir koordinat sistemine dönüşümlerini içerir, donanım çevre birimlerinin skaler kontrollere kıyasla daha yoğun hesaplama yapabilmelerini gerektirir, ancak daha iyi dinamik tepkiler, daha doğru makine tork regülasyonu ve genellikle daha sessiz çalışma sağlar.

Moturun verimli çalışma aralığına uygunluğunu simüle edilebilmek amacıyla, kontrol algoritmasının sanal ortamda koşturularak analizi gerçekleştirilirken, fiziki testlere de aynı süreçte yer verilmesi planlanmaktadır. Elde edilen veriler doğrultusunda, araç kontrol ünitesinin sürücü ve algoritması düzenlenecektir. Simyalizasyon, batarya yönetimi, motor verileri, sürücü arayüzü, hareket verileri: Analog ve dijital olarak alınıp işlenerek, kablolu ve kablosuz olarak aktarılacaktır.

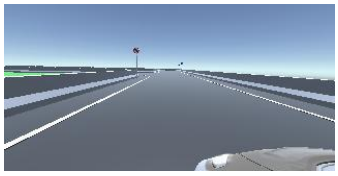
Denetleyici Alan Ağı (CAN veri yolu), mikro denetleyicilerin ve aygıtların ana bilgisayar olmayan uygulamalarda birbirleriyle iletişim kurmasını sağlamak için tasarlanmış güçlü bir araç veri yolu standardıdır. Araç içi aygıtlar arası iletişimler CAN Bus ile sağlanıp veriler mikro işlemciye gönderilecek, tasarım aşamasında olan bu sistem daha sonrasında radyo frekanslı haberleşme de bilgi aktarımını kolaylaştırmak amacıyla kullanılacaktır. Kablosuz haberleşme için Xbee modülleri kullanılacaktır. XBee modülleri, elektromanyetik spektrumdaki dalgaların modülasyonu yoluyla hava yoluyla iletişim kurar, bilgi iletir ve alır. Radyo frekansı (RF) aygıtları olarak işlev görürler. Veriler aynı frekansta olan İki Xbee modülü arasında çift yönlü olarak aktarılacaktır.

Can bus ile birbirlerine bağlanan araç içi aygıtlarından alınan veriler mikro işlemcide toplanıp xbee modülleri aracılığı ile radyo frekanslı olarak dışa iletilecektir. Tasarım aşamasında olan bu sistemde araç optimizasyonuna göre değişiklik gösteren parametlereler sebebi ile Xbee modül modelleri değişiklik gösterebilir.

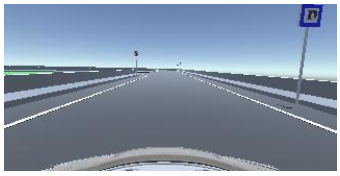
Xbee modül konfigürasyonları ve modül ağ izlenimleri için XCTU platformu kullanılacaktır. XCTU, geliştiricilerin Digi RF modülleriyle basit kullanımlı bir grafik arayüz üzerinden etkileşime girmelerini sağlamak için tasarlanmış bir çoklu platform uygulamasıdır. Aynı ağın 2 radyo modülü arasında menzil testinin yapılması veya RF cihazının yönetimi ve yapılandırması gibi alanlarda kullanılacaktır.

6. Otonom Sürüş Algoritmaları

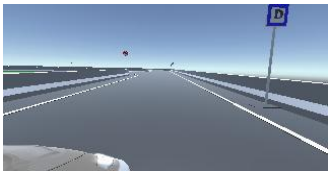
Otonom sürüş algoritmalarında öncelikle tasarladığımız simülasyonun eğitim modundan test sürüşlerini gerçekleştimekteyiz. Sağlıklı bir öğrenim için üç defa pistin normal güzergahında üç defa da güzergahın tersine sürüş yaparak kayıt alıyoruz. Aldığımız kayıt bizlere csv uzantılı dosya olarak geri dönüyor. Bu dosyanın içerisinde dönüş açısı, hız, fren, gaz ve kameradan gelen görüntüler tutulmaktadır.



Sol Kamera Görüntüsü



Orta Kamera Görüntüsü



Sağ Kamera Görüntüsü

Center camera	Left camera	Right camera	Steering Angle	Throttle	Brake	Speed
C:\...\IMG\center_2018_01_18_11_17_53_634.jpg	C:\...\IMG\left_2018_01_18_11_17_53_634.jpg	C:\...\IMG\right_2018_01_18_11_17_53_634.jpg	0	0	0	0.491714
C:\...\IMG\center_2018_01_18_11_17_53_750.jpg	C:\...\IMG\left_2018_01_18_11_17_53_750.jpg	C:\...\IMG\right_2018_01_18_11_17_53_750.jpg	0	0	0	0.48678
C:\...\IMG\center_2018_01_18_11_17_53_870.jpg	C:\...\IMG\left_2018_01_18_11_17_53_870.jpg	C:\...\IMG\right_2018_01_18_11_17_53_870.jpg	0	0	0	0.479955
C:\...\IMG\center_2018_01_18_11_17_53_985.jpg	C:\...\IMG\left_2018_01_18_11_17_53_985.jpg	C:\...\IMG\right_2018_01_18_11_17_53_985.jpg	0	0	0	0.475139

Datalar toplanırken CSV Çıktı Tablosu

Ardından evrişimli sinir ağları (Convolutional Neural Networks) kullanarak oluşturduğumuz yazılımın içine bu csv dosyasını çeşitli filtreleme işlemlerinden geçirerek ve Nvidia'nın öğrenme algoritmasını kullanarak kendi modelimizi eğitiyoruz.

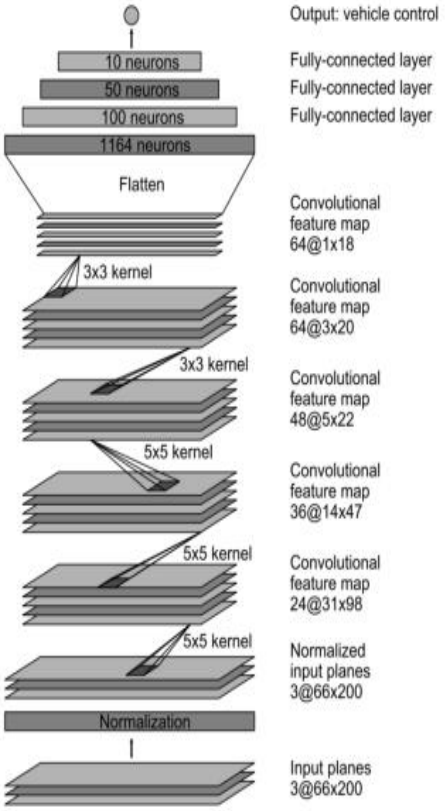
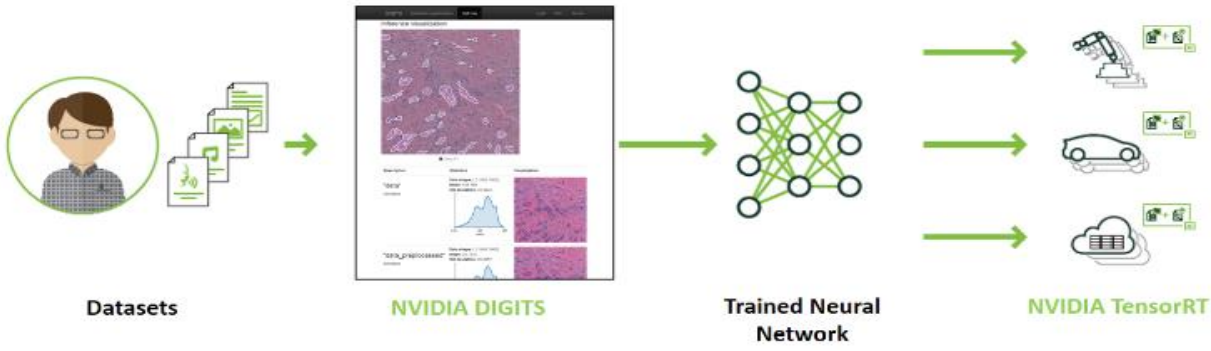


Figure 4: CNN architecture. The network has about 27 million connections and 250 thousand parameters.

```python
def batch_generator(image_paths, steering_ang, batch_size, istraining):

  while True:
    batch_img = []
    batch_steering = []

    for i in range(batch_size):
      random_index = random.randint(0, len(image_paths) - 1)

      if istraining:
        im, steering = random_augment(image_paths[random_index], steering_ang[random_index])

      else:
        im = mpimg.imread(image_paths[random_index])
        steering = steering_ang[random_index]

      im = img_preprocess(im)
      batch_img.append(im)
      batch_steering.append(steering)
    yield (np.asarray(batch_img), np.asarray(batch_steering))
x_train_gen, y_train_gen = next(batch_generator(X_train, y_train, 1, 1))
x_valid_gen, y_valid_gen = next(batch_generator(X_valid, y_valid, 1, 0))

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(x_train_gen[0])
axs[0].set_title('Training Image')

axs[1].imshow(x_valid_gen[0])
axs[1].set_title('Validation Image')
def nvidia_model():
  model = Sequential()
  model.add(Convolution2D(24, 5, 5, subsample=(2, 2), input_shape=(66, 200, 3), activation='elu'))
  model.add(Convolution2D(36, 5, 5, subsample=(2, 2), activation='elu'))
  model.add(Convolution2D(48, 5, 5, subsample=(2, 2), activation='elu'))
  model.add(Convolution2D(64, 3, 3, activation='elu'))

  model.add(Convolution2D(64, 3, 3, activation='elu'))
#   model.add(Dropout(0.5))


  model.add(Flatten())

  model.add(Dense(100, activation = 'elu'))
#   model.add(Dropout(0.5))

  model.add(Dense(50, activation = 'elu'))
#   model.add(Dropout(0.5))

  model.add(Dense(10, activation = 'elu'))
#   model.add(Dropout(0.5))

  model.add(Dense(1))

  optimizer = Adam(lr=1e-3)
  model.compile(loss='mse', optimizer=optimizer)
  return model
model = nvidia_model()
```



Trafik işaretleri için Tensorflow kütüphanesi ile Faster R-CNN COCO modelini kullanarak yüzlerce farklı bir açılardan çekilmiş trafik işaretlerini isimlendirip modelemizin içerisine atarak eğitiyoruz.Daha sonrasında python da canlı olarak görüntü yakalayabilmek için fonksiyon oluşturuyoruz.

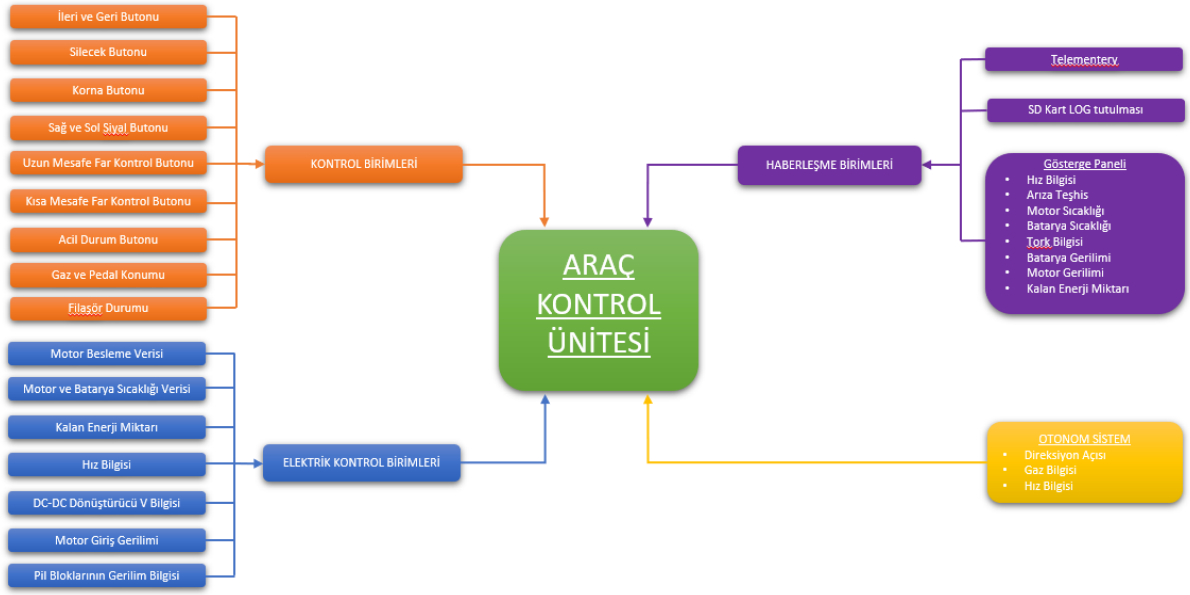
```python
def screen_record():
    last_time = time.time()
    while(True):
        # 800x600 windowed mode
        printscreen =  np.array(ImageGrab.grab(bbox=(400,160,680,310)))
        print('loop took {} seconds'.format(time.time()-last_time))
        last_time = time.time()
        cv2.imshow('window',cv2.cvtColor(printscreen, cv2.COLOR_BGR2RGB))
        if cv2.waitKey(25) & 0xFF == ord('q'):
            cv2.destroyAllWindows()
            break

# Import utilites
from utils import label_map_util
from utils import visualization_utils as vis_util

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'inference_graph'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')

# Number of classes the object detector can identify
NUM_CLASSES = 6

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
```

Oluşturduğumuz görüntü yakalama fonksiyonu ile sürekli olarak trafik levhalarını ve trafik lambasını algılamak için önce görüntüleri belirli bir filtreleme işlerinden geçirip daha sonrasında yakaladığı nesneyi kare içerisine alıp bize göstermesini sağlıyoruz.

```
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

# Initialize webcam feed
#video = cv2.VideoCapture(0)
#ret = video.set(3,1280)
#ret = video.set(4,720)
last_time=time.time()
while(True):

    # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
    # i.e. a single-column array, where each item in the column has the pixel RGB value
    printscreen = np.array(ImageGrab.grab(bbox=(800,250,1200,500)))
    printscreen=cv2.cvtColor(printscreen,cv2.COLOR_BGR2RGB)
    print('loop took {} seconds'.format(time.time() - last_time))
    last_time = time.time()

    frame = printscreen
    frame_expanded = np.expand_dims(frame, axis=0)

    # Perform the actual detection by running the model with the image as input
    (boxes, scores, classes, num) = sess.run(
        [detection_boxes, detection_scores, detection_classes, num_detections],
        feed_dict={image_tensor: frame_expanded})

    # Draw the results of the detection (aka 'visulaize the results')
    vis_util.visualize_boxes_and_labels_on_image_array(
        frame,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8,
        min_score_thresh=0.85)
```

7. Sistem Entegrasyonu

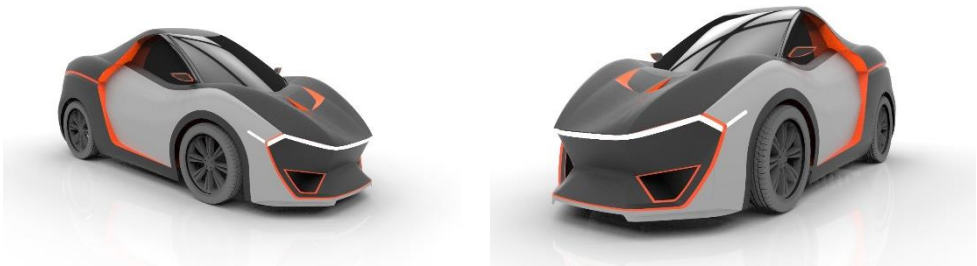
Araç içi aygıtlar arası iletişimler CAN Bus ile sağlanıp veriler mikro işlemciye gönderilecektir.
Direksiyon Kontrol Sistemi: Direksiyonu otomatik olarak çevirmek için bir step motor (NEMA34)bağlanmıştır. Direksiyon miline ve step motora özel dişli set hazırlanıp araca monte edilmiştir.
Fren Kontrol Sistemi: Aracın frenine otomatik olarak basmak için bir 12V lineer elektrik motoru bağlanmıştır.
Gaz Kontrol Sistemi: Aracın gaz kontrol sistemi kontrol ünitesinden direk müdahale edilecektir.



Otonom tarafında yapılan yazılımla beraber aracın gaz,fren,direksiyon kontrolleri için CNN(Convolutional Neural Network) tarafından gelen çıktılarla aracın kontrolleri sağlanacaktır.

8. Özgün Bileşenler

Aracın dış kabuki tasarımı ve iç tasarımı kendi ekibimizde bulunan Endüstriyel tasarımı öğrencileri tarafından dizayn edilmiştir. İç tasarımın da yine aynı ekip tarafından gerçekleştirilecek olup henüz tasarım aşamasındadır. Kabuk ve kalıp üretimi kendi tarafımızdan gerçekleştirilecek olup geri kalan motor sürücü, motor, batarya paketi gibi bölümlerin üretimi ve genel bilgileri Araç kontrol ünitesi bölümünde açıklanmıştır.



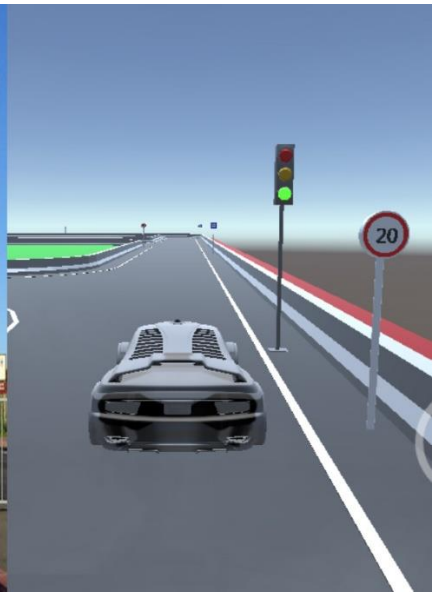
Araç dış görselleri

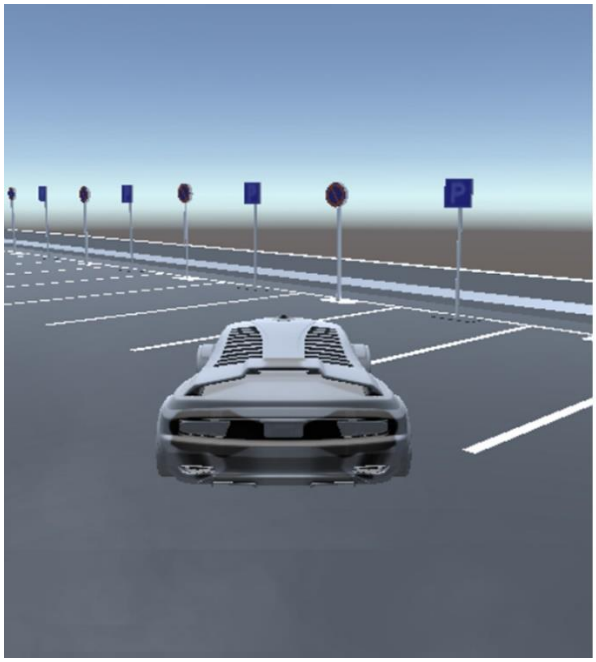
9. Güvenlik Önlemleri

Aracımızın üzerinde bulunan ultrasonic sensörler ile aracın yoldan çıkması durumunda bunu algılamaması gibi bir durumun oluşması sonucunda aracın otonom modundan çıkarak tam fren pozisyonunda aracı durduracak ve elektriği kapatarak bekleme pozisyonuna geçecektir.

10. Test

Aracın genel ölçüleri ve dönüş açıları Unity programında birebir modellendi. Yol , trafik ışıkları ve levhalar gerçek boyutlarında tasarlandı. Aracın üzerine yerleştirilecek kameralar simülasyon ortamında arabanın üzerine konumlandırıldı. Bütün tasarımlar birleştirilip simülasyon ortamında test edildi. Test sonucu olumlu çıkıncaya kadar olumsuz sonuçlar düzeltildi. Araca uygulanan test senaryosu birebir gerçek ortam ile aynı şekilde yapılmıştır. Aracın tasarım ve dönme açılarının simülasyon ile tam uyumlu şekilde testleri yapılmıştır.





11. Referanslar

- daimler.com. (2018). Daimler - Home. [online] Available at: https://www.daimler.com/innovation/autonomous-driving/special/definition.html [Accessed 14 Feb. 2018].
- bmw.com. (2018). BMW - Home. [online] Available at: https://www.bmw.com/en/automotivelife/autonomous-driving.html [Accessed 14 Feb. 2018].
- DataLab TUM. (2018). Data Innovation Lab - Autonomous lane following in a simulated environment project. [online] Available at: http://www.di-lab.tum.de/index.php?id=45&L=1 [Accessed 14 Feb. 2018].
- Wikipedia (2018). Wikipedia - Home. [online] Available at: https://en.wikipedia.org/wiki/Main_Page [Accessed 14 Feb. 2018].
- GitHub.com (2018). GitHub - Home. [online] Available at: https://github.com/ [Accessed 14 Feb. 2018].
- Introduction to Udacity Self-Driving Car Simulator (2018). Towards Data Science. [online] Available at: https://towardsdatascience.com/introduction-to-udacity-self-driving-car-simulator-4d78198d301d [Accessed 14 Feb. 2018].
- Python 3.5 Documentation (2018). Python documentation and libraries. [online] Available at: https://docs.python.org/3/library/ [Accessed 14 Feb. 2018].
- Udacity self-driving-car-simulator project at GitHub (2018). Udacity/self-driving-car-sim. [online] Available at: https://github.com/udacity/self-driving-car-sim [Accessed 14 Feb. 2018].
- Git-scm.com (2018). Git - Home. [online] Available at: https://git-scm.com/ [Accessed 14 Feb. 2018].
- Unity game engine (2018). Unity3D - Home. [online] Available at: https://unity3d.com/ [Accessed 14 Feb. 2018].
- [11] TensorFlow.com (2018). TensorFlow - Home. [online] Available at: https://www.tensorflow.org/ [Accessed 14 Feb. 2018].
- OpenDRIVE 1.4 Specifications Document (2018). openDRIVE® - Home. [online] Available at: http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.4H.pdf [Ac-cessed 14 Feb. 2018].
- OpenRoadEd.net (2018). OpenRoadEd - Home. [online] Available at: https://sourceforge.net/projects/openroaded/ [Accessed 14 Feb. 2018].
- keras.io (2018). Keras - Home. [online] Available at: https://keras.io/ [Accessed 14 Feb. 2018].
- CS231n: Convolutional Neural Networks for Visual Recognition (Spring 2017). Courses at Stanford University. [online] Available at: http://cs231n.stanford.edu/index.html [Accessed 14 Feb. 2018].

- Deeplearning4J.org (2018). Deep Learning Tutorials. [online] Available at: https://deeplearning4j.org/logistic-regression [Accessed 14 Feb. 2018]. Autonomous Lane Following in a Simulated Environment –confidential– Seite 32 von 32 © ITK Engineering GmbH, Dok-ID: xxx, v x.y.z, gültig ab 15.02.2017
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- Alex Graves, Greg Wayne, and Ivo Danihelka. Neural tuning machines. arXiv preprint arXiv:1410.5401, 2014.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097– 1105, 2012.
- Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. IEEE Software, 17(4), 26-32.