T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARACA EK SENSORLAR İLE MODÜLLER EKLENEREK VE GÖRÜNTÜ İŞELEME TEKNİKLERİYLE SÜRÜŞ KALİTESİ TESPİTİ

BİTİRME TEZİ

İbrahim TOSUNOĞLU

Anabilim Dalı: Yazılım Mühendisliği

Danışman: Dr. Öğr. Üyesi Özgür KARADUMAN

Tezin Enstitüye Verildiği Tarih: …/…/2019

HAZİRAN – 2019

ÖNSÖZ

Bu tez çalışmasında, trafik kazalarının nedenlerinin büyük çoğunluğunu oluşturan kalitesiz ve agresif sürüşün tespit edilmesi, bu tür sürücülerin belirlenmesiyle gerekli cezai işlemin uygulanabilmesi için zemin hazırlanmak istenmiştir. Bu sayede daha kaliteli bir trafik elde edilmesi hedeflenmiştir.

Öncelikle tez konusunu seçerken isteklerimi göz önünde bulundurup bana yardımcı olan danışmanım Dr. Öğr. Üyesi Özgür Karaduman'a, geliştirme süreci boyunca destek aldığım Yaz. Müh. Harun Tosunoğlu, Erhan Osmanoğlu ve Safa Müjdeci 'ye, ayrıca maddi ve manevi desteğini her an yanımda hissettiğim aileme ve arkadaşlarıma teşekkürü borç bilirim.

İbrahim TOSUNOĞLU

ELAZIĞ – 2019

İÇİNDEKİLER

ÖZET

Dünyada her yıl milyonlarca trafik kazası gerçekleşmektedir. Ve bu kazalar sonucu yaklaşık 1,25 milyon kişi hayatını kaybetmektedir. Dünya Sağlık Örgütü'nden (WHO) elde edilen tahminlere göre insan ölümüne yol açan başlıca sebepler arasında trafik kazaları 2004 yılında 9. sırada iken, 2030 yılında 5. sıraya kadar yükselecektir [1]. Türkiye İstatistik Kurumu'nun verilerine göre ise 2017 yılında toplam trafik kazası sayısı 1 202 716 iken bu kazalarda ölen kişi sayısı 7 424 olarak tespit edilmiştir. Yine aynı verilere bakarak bu kazalardaki nedenlerin başında %88,2'lik bir oranla sürücü hataları gelmektedir [2]. Bu verilere dayanarak daha güvenli bir trafik için sürücünün ve sürüş kabiliyetinin denetlenmesi gerekmektedir.

Gelişmekte olan akıllı şehirlerin önemli bir parçasını akıllı araçlar oluşturmaktadır. Bu akıllı araçların kullanılmakta veya yapım aşamasında olan birçok özelliği mevcuttur. Fakat sürüş kalitesi ve sürücü yetkinliğini denetleyen çok az çalışma bulunmaktadır. Şuan kullanılmakta olan sistemde, trafik cezaları, mobeseler, trafik kontrol noktaları yetersiz kalmaktadır. Her alana, her caddeye ve sokağa mobese koymak imkânsız veya fazla maliyetli olduğundan, bu denetimin araç içinde olması gerekmektedir.

Daha önce yapılmış olan çalışmalarda araç içi kamera sayesinde elde edilen verilere göre, mobil cihazlardan alınan veriler sayesinde veya aracın kendi içinde bulundurduğu CAN bus verileri üzerinden "data mining" yapılarak sürüş kalitesinin veya agresif sürüşün tespiti yapılmıştır.

Bu çalışmada ise akıllı araçlara entegre edilebilecek bu sistem sürücünün sürüş kalitesini ve kaza riski belirlenmiştir. Trafik ışıklarına uyulmaması, emniyet kemeri takılmaması, şehir içinde ve dışında hız sınırlarının aşılması, ani hızlanma ve frenlemeler, çarpmalar sürücüye eksi puan olarak döndürülecektir. Bunun için bir Raspberry Pi B 3+ kullanılmış olup yanı sıra görüntü alma işlemi için bir kamera, hız tespiti ve ani hız değişiklikleri (ani frenleme veya hızlanma) için GPS modülü, emniyet kemeri kontrolü için bir switch, kaza tespiti için de mesafe sensoru kullanılmıştır.

Projeden elde edilen veriler daha sonrasında isteğe bağlı olarak online veya offline olarak Emniyet Müdürlüğü ya da sigorta şirketleri gibi kurumlarla paylaşılabilir olup, bunun sayesinde gereken trafik cezası işlemleri veya sigorta ücreti hesaplanmasında kullanılabilecektir. Sonuç olarak bu sistem caydırıcı bir unsur olacak ve daha güvenli bir trafik elde edilebilecektir.

ABSTRACT

Millions of traffic accidents occur every year in the world. As a result of these accidents, approximately 1.25 million people die. According to the estimates from the World Health Organization (WHO), traffic accidents were the 9th most common cause of human death in 2004 and will rise to 5th in 2030 [1]. According to data from the Statistics Institute of Turkey's total number of traffic accidents in 2017 were identified 1,202,716 while the number of people who died in this accident as 7424. According to the same data, driver errors are seen at the beginning of these accidents with a rate of 88.2% [2]. Based on this data, the driver and his driving ability need to be checked for safer traffic.

Smart cars are an important part of developing smart cities. These smart tools has available with many features being used or under construction. However, there are very few studies that control driving quality and driver competence. In the current system, traffic fines, security cameras, traffic control points are insufficient. Since it is impossible or too costly to put cameras on every area and every street, this control must be in the vehicle.

In the literature, according to the data obtained from the in-car camera, the data obtained from the smartphone or the with CAN bus data contained in the vehicle, an data mining to determine for the driving quality or aggressive driving.

In this study, this system determine the driving quality and accident risk of the driver which can be integrated into smart vehicles. Failure to comply with traffic lights, not wearing a seat belt, exceeding speed limits in and outside the city, sudden acceleration and braking, impacts will be returned as negative points to the driver. For this purpose, a Raspberry Pi B 3 was used, as well as a camera for image acquisition, a GPS module for speed detection and sudden speed changes (sudden braking or acceleration), a switch for seat belt control, and a distance sensor for accident detection.

The data obtained from the project can then optionally be shared online or offline with agencies such as the Police Department or insurance companies, which can be used to calculate required traffic penalty transactions or insurance fees. As a result, this system will be a deterrent and a safer traffic will be achieved.

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

KISALTMALAR LİSTESİ

WHO : World Health Organization (Dünya Sağlık Örgütü)

TÜİK : Türkiye İstatistik Kurumu

MOBESE : Trafik Güvenlik Kamerası

GPS : Global Positioning System

GPIO : General-Purpose Input/Output

VCC : IC Power-Supply Pin

GND : Chassis Ground

RX : Receive X (Almak)

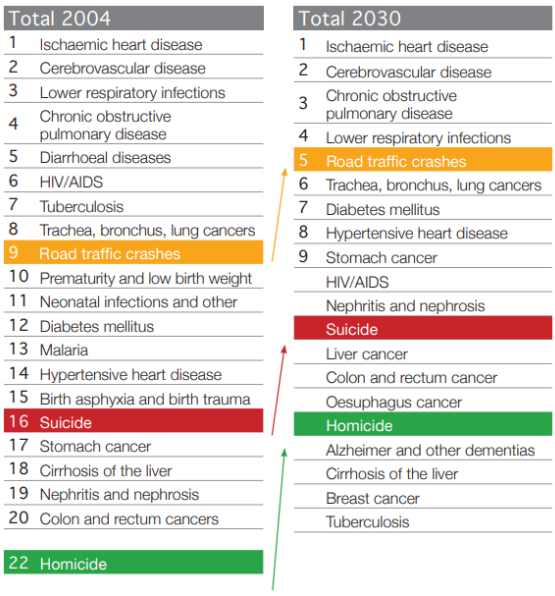
TX : Transmit X (İletmek)

USB : Universal Serial Bus (Evrensel Seri Veriyolu)

CAN : Controller Area Network

1. GİRİŞ

Günlük hayatın ve ulaşımın vazgeçilmez bir parçası, araçlar ve araçların oluşturduğu trafiktir. İnsanlar gerek toplu taşıma araçları gerekse özel araçlar sayesinde ulaşımlarını sağlamaktadırlar. Trafikteki yoğunluktan, dikkatsizlikten, agresif sürüşlerden, kurallara uyulmamasından kaynaklı birçok trafik kazası gerçekleşmektedir. Ve bu sayı git gide artmaktadır. Trafikte güvenli bir şekilde yol alınması için belirli kurallar vardır, fakat bu kurallara her zaman uyulmamaktadır. Bu nedenle ölümlü ve maddi kayıplı trafik kazaları yaşanmaktadır. İnsan ölümünün nedenlerinin istatistiklerini paylaşan Dünya Sağlık Örgütü (WHO) trafik kazalarının 2004'te 9. sırada iken, gösterdiği hızlı artış sayesinde 2030 yılına kadar ilk 5 sıraya kadar yükseleceği tahmin edilmektedir [1]. Şekil 1'de Dünya Sağlık Örgütü'nün paylaştığı tablo görülmektedir.



Şekil 1 Dünya Sağlık Örgütü İstatistikleri

Dünya Sağlık Örgütü'nün yanı sıra TÜİK (Türkiye İstatistik Kurumu) tarafından elde edilen verilere göre 2017 yılında Türkiye'de gerçekleşen trafik kazası sayısı 1.202.716 iken bu kazalarda toplam ölen insan sayısı 7.427'dir. Bir başka açıdan bakılacak olursa 2008 yılından bu yana trafik kazalarında ciddi bir artış gözlenmektedir ve kaza sayısıyla doğru

orantılı olarak ölüm sayısı da artmaktadır [2]. Tablo 1'de TÜİK'in yayınlamış olduğu istatistikler görülmektedir.

YILLAR	TOPLAM KAZA SAYISI	ÖLÜMLÜ, YARALANMALI KAZA SAYISI	MADDİ HASARLI KAZA SAYISI	ÖLÜ SAYISI			YARALI SAYISI
				TOPLAM	KAZA YERİNDE	KAZA SONRASI (1)	
2008	950.120	104.212	845.908	4.236	4.236	-	184.468
2009	1.053.345	111.121	942.224	4.324	4.324	-	201.380
2010	1.105.201	116.804	988.397	4.045	4.045	-	211.496
2011	1.228.928	131.845	1.097.083	3.835	3.835	-	238.074
2012	1.296.634	153.552	1.143.082	3.750	3.750	-	268.079
2013	1.207.354	161.306	1.046.048	3.685	3.685	-	274.829
2014	1.199.010	168.512	1.030.498	3.524	3.524	-	285.059
2015	1.313.359	183.011	1.130.348	7.530	3.831	3.699	304.421
2016	1.182.491	185.128	997.363	7.300	3.493	3.807	300.812
2017	1.202.716	182.669	1.020.047	7.427	3.534	3.893	300.383

Tablo 1 TÜİK Yıllara Göre Kaza İstatistikleri

Yine TÜİK'in yayınladığı bir başka tabloya göre bu trafik kazalarında hataların büyük çoğunluğunu %88,2 gibi bir oranla sürücüler oluşturmaktadır [2]. Bu sebeple denetimin asıl yapılması gereken yerin araç içi ve sürücünün kendisi olduğu açık bir şekilde görülmektedir. Tablo 2 ise bu verileri belirtmektedir.

KAZA FAKTÖRLERİ	YERLEŞİM YERİ		YERLEŞİM YERİ DIŞI		TOPLAM	
	Kusur Sayısı	%	Kusur Sayısı	%	Kusur Sayısı	%
Sürücü	142.540	88,2	53.846	96,2	196.386	90,3
Yaya	17.313	10,7	843	1,5	18.156	8,3
Yolcu	537	0,3	276	0,5	813	0,4
Taşıt	616	0,4	585	1,0	1.201	0,6
Yol	571	0,4	400	0,7	971	0,4
TOPLAM	**161.577**	**100**	**55.950**	**100**	**217.527**	**100,0**

Tablo 2 TÜİK Kaza Faktörleri Tablosu

Bu ölümlü kazaların sürücü kaynaklı başlıca nedenleri, TÜİK'in paylaştığı veriler baz alınarak incelendiğinde;

- Hız limitlerine uyulmaması
- Trafik ışıklarına ve levhalarına uyulmaması
- Emniyet kemerinin kullanılmaması
- Alkollü araç kullanımı
- Geçiş üstünlüğüne uymamak

Ve benzeri nedenlerden oluşmaktadır [2].

Hali hazırda yürürlükte olan sistemde caydırıcı unsur olarak trafik cezaları, MOBESE kayıtları, trafik kontrol noktaları, hız radarları gibi önlemler alınmış olsa da artık yeterli gelmemektedir. Bu kontrol noktalarının artırılması, her caddeye ve sokağa MOBESE'lerin uygulanması mümkün olmadığı gibi hem maliyet olarak hem de iş yükü olarak sınırları zorlayacaktır. Bu nedenle bu denetimlerin trafik içinde değil, araç içinde yapılması gerekmektedir. Araçlara entegre edilebilecek bu sistem sayesinde sürücünün davranışları kayıt altına alınabilecek ve ekstra bir denetime gerek kalmadan güvenli bir trafik ortamı için zemin hazırlanacaktır.

Araç içinde yapılacak denetim sayesinde, sadece hız radarlarının, trafik kontrol noktalarının, MOBESE'lerin bulunduğu ortamlarda değil trafiğin her alanında caydırıcı cezai işlemlere maruz kalan sürücü daha dikkatli olmaya zorlanmış olacak ve hem kendisinin hem de trafikteki diğer insanların can sağlığı için kurallara uyulması sağlanacaktır.

1.1 Literatür Taraması

Sürüş kalitesini veya agresif sürücü davranışlarını izleyen ve tespit eden birçok çalışma yapılmıştır. Bunlardan biri 2012 yılında Ahmad Aljaafreh, Nabeel Alshabatat ve Munaf S. Najim Al-Din tarafından geliştirilmiş olup bulanık mantık (fuzzy logic) ile sürüş stilini sınıflandırmak üzere yapılmıştır. Bu sınıflar "normalin altında, normal, agresif ve çok agresif" olarak belirlenmiştir. Araçlara entegre edilen GPS ve 2 eksenli ivme ölçer kullanarak veriler toplanmıştır. Enlem ve boylamın Euclidian normlarının gücü bir bulanık çıkarsama algoritmasına girdi olarak verilmiştir. Bu algoritmanın çıktısı, verilen verileri daha önce belirlenen sınıflardan birine atamaktır. Bu sayede sürücünün sürüş tarzının sınıflandırılması gerçekleştirilmiştir [3].

Diğer bir yöntem ise yine 2012 yılında H. Eren, S. Makinist, E. Akın ve A. Yılmaz tarafından gerçekleştirilen "Akıllı Telefonla Sürüş Davranışını Tahmin Etme" adlı çalışmadır. Bu çalışmada ise ekstra bir donanım veya sensor kullanılmadan, akıllı telefonlar üzerinde bulunan ivmeölçer, jiroskop ve manyetometreyi kullanarak araçtan bağımsız bir sistem geliştirilmiştir. Bu sensorlar kullanılarak hız, hızlanma, yavaşlama ve sapma açısı verileri elde edilmiştir. Daha sonra bu veriler üzerinden istatistiksel çıkarımlar yapılarak

sürüş kalitesi hesaplanmaktadır. Bu sayede daha az maliyetle tespit yapılabileceği gösterilmektedir [4].

2011 yılında Derick A. Johnson ve Mohan M. Trivedi'nin gerçekleştirdiği çalışmaya bakılacak olursa sürüş tipleri "agresif" ve "agresif olmayan" olarak ikiye ayrılmıştır. Yine akıllı telefon tabanlı sensor füzyonu (ivmeölçer, jiroskop, manyetometre, GPS, video) kullanarak sürüş tarzlarını sınıflandırmaktadır [5].

O. Karaduman, H. Eren, H. Kurum ve M. Çelenk ise 2013 yılında agresif/sakin sürüş tiplerini denetlemek için araçlarda bulunan CAN (Kontrolör Alan Ağı) veri yolundan elde edilen verilerdeki önemsiz değişkenleri ortadan kaldırarak daha sade bir veri üzerinde çalışmış ve agresif sürüş tespiti yapmışlardır [6].

Farklı bir pencereden bakılacak olursa Seyed Mohammad Reza Noori ve Mohammad Mikaeili, sürüş kalitesini bir uyuşukluk testi yaparak tespit etmişlerdir. 2016 Ocak'ta yayınlanan bu çalışmada elektroensefalografi, elektrookülografi sinyalleri kullanılarak sürücünün uyuşukluk ve uyku hali tespit edilmiştir [7].

Yine bir bulanık çıkarsama algoritması kullanan T. Imkamon, P. Saensom, P. Tangamchit ve P. Pongpaibool, sürüş olaylarını kayıt eden ve güvenli olmayan sürüş davranışlarını tespit eden bir sistem önermişlerdir. 3 sensor, bir motor kontrol ünitesi okuyucusu (ECU), 3 eksenli bir ivmeölçer ve bir kamera kullanılarak toplanan verileri bulanık çıkarsama algoritmasına girdi olarak veren ekip, bu algoritmanın çıktısı olarak 1 ile 3 arasında değişen bir çıktı almaktadır. 1 en güvenli sürüş seviyesi iken 3 riskli bir sürüşü işaret etmektedir [8].

Trafik kazalarının ana nedenini alkollü araç kullanımı olarak gören Jiangpeng Dai, Jin Teng, Xiaole Bai, Zhaohui Shen ve Dong Xuan cep telefonlarının kullanılmasıyla bu durumu tespit eden bir sistem geliştirmişlerdir. Sarhoş sürüş sonucu tehlikeli araç manevralarının erken tespiti sonucu sürücünün uyarılmasını sağlayan bu sistemde araç ivmelerinin hesaplanması sonucu daha önce sistemde kayıtlı olan tehlikeli manevra kayıtları üzerinde benzerlik testi yaparak polise de haber verilmesi sağlanmaktadır [9].

Akıllı telefon tabanlı tehlikeli sürüş davranışını algılayan bir diğer yöntem ise Fu Li, Hai Zhang, Huan Che ve Xiaochen Qiu tarafından gerçekleştirilmiştir. Yine telefon üzerindeki sensorları kullanan projede yalpalama açısı algılama algoritması geliştirilmiştir. Ve bu algoritma sonucunda 4 farklı tip sürüşten biri çıktı olarak alınabilmektedir [10].

Zhongyang Chen, Jiadi Yu, Yanmin Zhu, Yingying Chen ve Minglu Li 2015 yılında yine akıllı telefon sensorlarını kullanarak "hızlı U dönüşü", "ani frenleme" gibi hareketlerin algılanmasını sağlamaktadır. Bu sayede 95.36% başarı elde edildiğini göstermektedirler [11].

"Kenar Tabanlı Şerit Değişimi ve Şüpheli Sürüş Davranışı Analizi" adlı çalışmada ise şeritlerin tespiti yapılmakta ve şeritlerin durumuna göre sürücünün davranışları kayıt altında tutulmaktadır. Sin-Yu Chen ve Jun-Wei Hsieh bu çalışma sayesinde tehlikeli sürüş tespiti yapmışlardır [12].

2007 yılında Azim Eskandarian ve Ali Mortazavi sürüş kalitesini etkileyen bir diğer unsur olan uyuşukluk testini, direksiyon hareketlerini içeren veriler sayesinde tespit etmişlerdir. Direksiyon hareketlerinin bir yapay sinir ağı algoritmasına verilmesiyle, direksiyon hareketleriyle uyuşukluk hali arasındaki korelasyon bulunmaktadır. Ve uyuşuk olan sürücü tespit edilmektedir [13].

Bir diğer çalışmada ise Wang Hailin ve ekibi sürüş kalitesinin parametresini yorgunluk olarak kabul etmiştir. Hızlanma, frenleme, vites değiştirme ve sürüş gibi sürüş davranışlar girdi olarak ele alınmış olup, sürücülerin yorgunluğa bağlı olarak bu sinyallerin değişimi tespit edilmiştir. Bunun üzerine yorgunluk tespiti gerçekleştirilmiştir [14].

2018 yılında Xinrong Wu, Junwei Zhou, Jinghe An ve Yanchao Yang, otobüslerdeki anormal sürüş davranışlarının tespiti için otobüse ait bir mobil telefondan ivmeölçer ve oryantasyon sensoruyla toplanan verilerle Bayesian sınıflandırıcısını eğitip ani frenleme, kısa şerit değiştirme, hızlı dönüş, hızlı U dönüşü ve uzun süreli park etme gibi belirli gizli anormal sürüş davranışlarının tespitini yapmaktadırlar [15].

Hashim Saeed, Tabish Saeed, Muhammed Tahir ve Momin Uppal 2018'de gerçekleştirdikleri çalışmada riskli sürüş davranışlarını araç içi Wi-Fi sinyallerini kullanarak tespit etmişlerdir. Riskli sürüş 4 ana kritere göre belirlenmiştir. Bunlar esneme, baş hareketleri, yandan hareket ve akıllı telefon kullanımı olarak belirlenmiştir. Bu tür veriler üzerinde çalışarak %88.64'lük bir başarı elde edilmiştir [16].

Sürücünün bakış açısının da önemli olduğunu savunan Mohd Nizam Husen ve ekibi ise 2017 yılında bir çalışma gerçekleştirmiştir. Sürcünün göz bakışlarından oluşan veriler ile birlikte hız, direksiyon açısı ve sinyal göstergeleri gibi çeşitli veriler üzerinde çalışılan

projede örüntü tanıma sistemleri kullanılarak sürüş kalitesi tespiti yapılması amaçlanmıştır [17].

Harici sensorlardan, donanımlardan veya akıllı telefonlardan kullanmayan Luyang Liu, Çağdaş Karataş ve ekibi, giyilebilir teknolojiler sayesinde sürüş tespiti yapmak için bir çalıştırma gerçekleştirmiştir. 2015 yılında gerçekleştirilen bu çalışmada, akıllı saatler veya spor bandlarından alınan veriler sayesinde, direksiyon dönüş açılarını belirlemek gibi etmenlere bakılarak sürüş güvenliği tespit edilmektedir. Çalışma sonucunda %98'lik bir başarı alındığı gösterilmektedir [18].

İnteraktif bir çalışma gerçekleştiren Alex Găvrută, Marius Marcu ve Răzvan Bogdan sürücünün hareketlerini analiz edip sürüş stilini iyileştirmek üzere sürücüye puan veren ve sürücüye sürüş ile ilgili tavsiyelerde bulunan bir program tasarlamışlardır. Verilen tavsiyelerle sürücünün daha dikkatli davranması amaçlanmaktadır [19].

Bunun dışında çok değişkenli normal bir model kullanarak yinelemeli olarak sürücünün, aracın ve akıllı telefon kombinasyonunun istatistiksel bir modelini oluşturan uyarlamalı bir sürüş manevrası algılama mekanizması geliştiren German Castignani ve ekibi araç, sürücü ve yol topolojisine uyarlayan bir eğitim mekanizması sayesinde sürücünün puanlanmasına dayalı bir çözüm sunmaktadır. Bu sayede manevralar ve riskli sürüş biçimleri tespit edilmektedir [20].

Yang Zheng ve John HL Hansen'in 2016'da gerçekleştirdiği çalışmada araç içinde serbest bir şekilde konumlandırılmış akıllı telefonlar üzerinden sürüş olaylarını yinelemeli ayraçları tespit ederek sınıflamak için denetimsiz kümeleme tekniklerini kullanarak riskli sürüşler tespit edilmektedir [21].

Ayrıca sürüş kalitesini denetleyen çalışmalarda görüntü işleme teknikleri şerit tanıma gibi alanlarda kullanılmış olsa da trafik ışıklarını veya levhalarını tanıyan bir modülü içermemektedirler. Bu kapsamdan ayrı inceleyecek olursak trafik ışıklarının tespiti için çalışmalar mevcuttur.

Buna örnek olarak Raoul de Charette ve Fawzi Nashashibi'nin 2009'da gerçekleştirdiği çalışma gösterilebilir. Algılama ve uyarlamalı trafik ışık şablonları yardımıyla gerçek zamanlı trafik ışıklarının tanınması sağlanmaktadır. 2.9 GHz tek çekirdekli bir bilgisayar yardımıyla 480 görüntü işlenebildiğini göstermişlerdir [22]. Fakat

araç içinde kullanılabilmesi için daha düşük frekans hızlarında çalışan işlemciler sayesinde bu işlemin gerçekleştirilmesi gerekmektedir.

Masako Omachi ve Shinichiro Omachi bu sistemin daha hızlı çalışmasını sağlamak için bir çalışma gerçekleştirmiştir. İlk olarak renk alanının RGB'den normalize RGB'ye çevrilen çalışmada bazı bölgeler trafik ışığı adayı olarak seçilmektedir. Daha sonra kesin yargıya varmak için Hough dönüşümüne dayanan bir sistem uygulanmaktadır. Ve bu sayede şablonlara bağlı kalmayarak daha hızlı bir tanıma sistemi gerçekleştirilmektedir [23].

Bu çalışmada ise buradaki yönteme benzer bir şekilde trafik ışığı tespit edilmektedir. Raspberry Pi'ın işlem kapasitesinin sınırlı olduğu ve hızlı çalışması göz önüne alındığında hafızanın verimli kullanılması ön plana çıkmaktadır. Bu nedenle önceden eğitilmiş şablonlar hız kaybettirecek olduğundan bu yöntemle tespit sağlanmıştır.

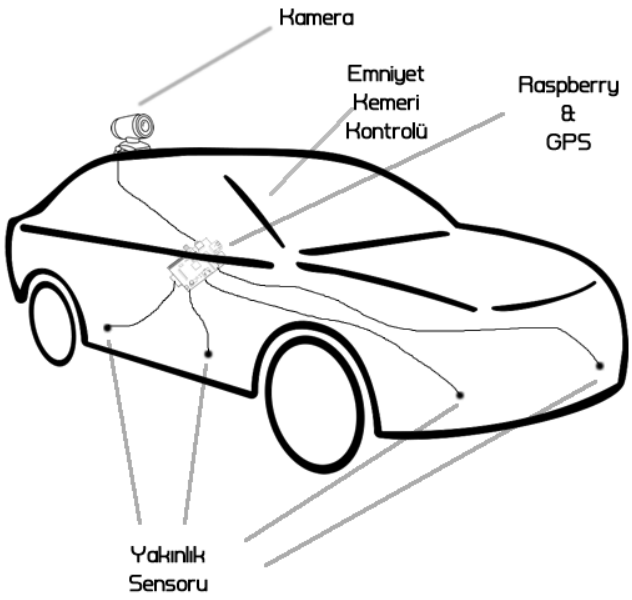
Yapılan çalışmanın amacı daha güvenli bir trafik için sürücülerin sürüş kalitelerini tespit etmek adına araçlara entegre edilebilecek bir sistem entegre edebilmek ve bu sistem sayesinde agresif ve kötü davranışlar içeren sürücülerin tespit edilmesidir. Her araçta bulunmasını hedeflediğimiz çalışmamızda trafiğin her alanında sürücülerin bilinçlenmesi ve kurallara uygun hareket etmesi sağlanması amaçlanmaktadır.

2. MATERYAL VE METOTLAR

Bu çalışmada trafikteki sürücülerin kabiliyetlerinin tespit edilmesi için bir Raspberry Pi ve ek donanımlar kullanılmıştır. Bu donanımlar;

- Neo-7M Çift Anten GPS Modülü
- USB kamera
- Emniyet Kemeri Yuvası ve Kontrol Switch
- HC-SR04 Ultrasonik Mesafe Sensorudur.

Ve bağlantının araç üzerinde uygulaması Şekil 2'de bir model olarak gösterilmektedir.



Şekil 2 Projenin Araç Üzerine Uygulanması

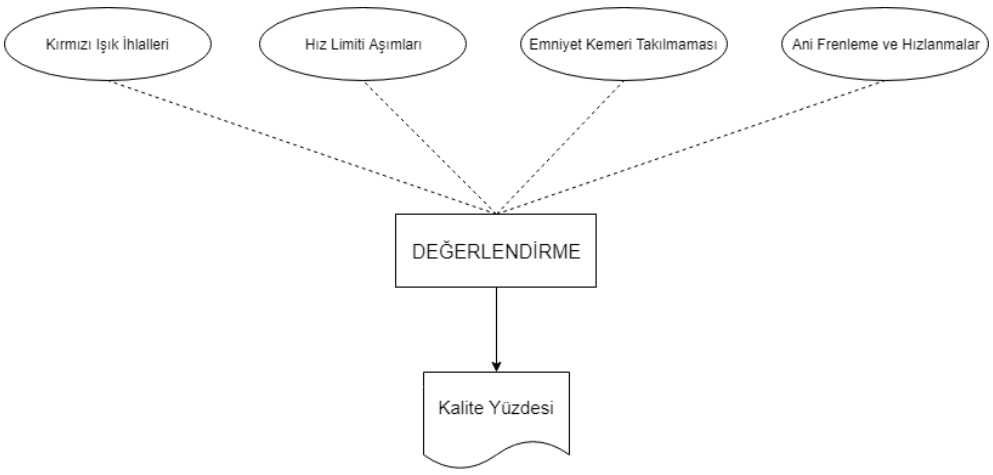
Bu donanımlar sistemin giriş verilerini sağlamaktadır. GPS modülü sayesinde anlık konum, hız ve ivme tespiti yapılmaktadır. USB kamera ile elde edilen görüntüler sayesinde trafik ışıkları tanınacaktır. Emniyet kemerinde bulunan switch sayesinde emniyet kemerinin

takılıp takılmadığının kontrolü yapılmaktadır. Ultrasonik mesafe sensorundan alınan veriler ise aracın bir yere çarpa tespitinde kullanılmaktadır.

Çalışma genel olarak 5 probleme çözüm aramaktadır. Bu nedenler 5 ana başlıkta incelenecektir. Bu başlıklar;

- Trafik içindeki hız limitlerine uyulmaması,
- Ani frenleme ve hızlanmalar,
- Kırmızı ışık ihlalleri,
- Aracın herhangi bir yüzeye çarpması,
- Emniyet kemersiz araç kullanımıdır.

Tüm bu denetimlerin ardından elde edilen, sürücüye ait hataların toplamını içeren veriler incelenmeli ve anlamlı bilgiler elde edilmelidir. Bu çalışmada bu bilgi bir yüzdelik dilim olarak elde alınmakta ve sistemden elde edilmiş verilerin sonucunda sürücünün yüzde kaç kaliteli bir sürüş gerçekleştirdiği elde edilmektedir.



Şekil 3 Toplanan Verilerin Değerlendirilmesi

Bu değerlendirme belirli ağırlıklara göre formüle edilmiştir. Bu formül aşağıdaki gibidir;

$$Z = A \times \beta_1 + B \times \beta_2 + C \times \beta_3 + D \times \beta_4 \qquad \textbf{(1)}$$

1 numaralı formülde gösterilen değerler şu şekildedir;

- Z : Kaliteyi ifade eden ve yüzdelik bir dilim veren değişken.
- A : Kırmızı ışık ihlalinin kaliteye etki ettiği ağırlık katsayısı. (0,3)
- β_1 : Sürücünün kırmızı ışıkta geçme oranı.
- B : Hız limiti aşımının kaliteye etki ettiği ağırlık katsayısı. (0,4)
- β_2 : Sürücünün hız limitini geçme oranı.
- C : Emniyet kemeri takılmamasının kaliteye etki ettiği ağırlık katsayısı. (0,1)
- β_3 : Sürücünün emniyet kemeri takmama oranı.
- D : Ani frenleme ve hızlanmaların kaliteye etki ettiği ağırlık katsayısı. (0,2)
- β_4 : Sürücünün ani frenleme ve hızlanma oranı.

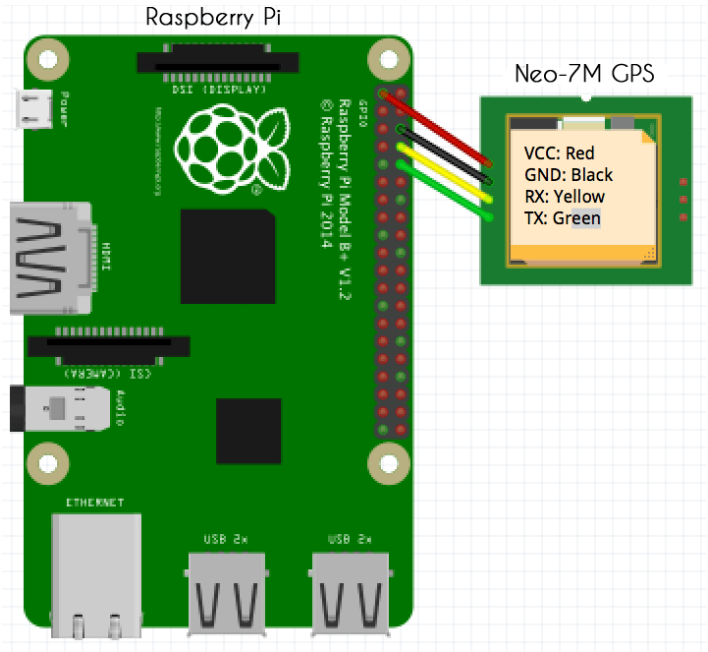
Yapılan araştırmalara göre kaza nedenlerinin sıralaması belirlenmiştir. Kaza ve ölüm nedenleri en çoktan aza doğru "Aşırı Sürat – Kırmızı Işık İhlali – Ani İvmeler – Emniyet Kemerinin Takılmaması" olarak sıralanmaktadır. Formülde bulunan katsayılar bu sıraya göre verilmiş olup önem derecesine göre kaliteye daha fazla etki etmesi sağlanmıştır.

2.1 Aşırı Hız Tespiti

Trafikte şehir içi ve şehir dışı hız limitleri vardır. Bu limitlere uyulmaması dâhilinde kaza kaçınılmaz hale gelmektedir. Bu nedenle bunun kontrol edilmesi gerekmektedir. Bu çalışmada hızın tespit edilmesi için Neo-7M çift antenli bir GPS modülü kullanılmaktadır. Hız ve konum verilerinin elde edilme aşamaları aşağıda belirtilmektedir.

2.1.1 GPS Modülü – Raspberry Pi Bağlantısı

GPS modülünden konum ve hız verilerinin alınması için öncelikle Raspberry Pi ile bağlantısının gerçekleştirilmesi gerekmektedir. Bu bağlantı Raspberry Pi üzerinde bulunan pinler yardımıyla gerçekleşmektedir. Bağlantını şekli Şekil-2'de gösterildiği gibidir.



Şekil 4 Raspberry Pi ve GPS Bağlantısı

Şekil 2'de görüldüğü üzere GPS modülünün TX çıkışı Raspberry'nin 10 numaralı pinine (GPIO 15), RX çıkışı 8 numaralı pinine (GPIO 14), GND topraklama pinine (Pin 6), VCC ise 5v çıkış almak için 1 numaralı pine bağlanmaktadır.

2.1.2 GPSD Kütüphaneleri

Bağlantı tamamlandıktan sonra GPS modülünden verilerin çekilebilmesi için bir kütüphane kullanılması gerekmektedir. Bu çalışmada GPSD kütüphaneleri kullanılmıştır. GPSD kütüphanesi sayesinde GPS modülünün bağlı olduğu pinler sisteme tanıtılır ve bu pinler üzerinden çekilen veriler anlamlı bir biçime sokularak kullanılmaktadır. Bu kütüphane sayesinde direkt olarak hız gibi veriler belirli fonksiyonlarla çekilip rahatlıkla kullanılabilmektedir.

2.1.3 Şehir İçi / Şehir Dışı Tespiti

Gerçekleştirilen projede sadece Elazığ şehri için tespit yapılmaktadır. Aynı şekilde diğer şehirler isteğe göre eklenebilecektir. İl sınırlarının koordinatları sistemde tutulmakta ve GPS modülünden çekilen konum bu il sınırlarının içinde kalıyorsa, şehir içi hız limitlerine göre, eğer dışında kalıyorsa şehir dışı hız limitlerine göre değerlendirme yapılmaktadır. Şekil 3'te bu sistemin anlaşılması için bir görsel gösterilmektedir.

Şekil 5 Şehir İçi / Şehir Dışı Tespiti

Bu sayede şehir içinde farklı parametreler kullanılabilecekken şehir dışı için farklı parametreler kullanılabilecektir.

2.1.4 Hız Verilerinin Alınması

Tüm bağlantılar ve konum alma işlemleri tamamlandıktan sonra aracın hızı belirlenmektedir. Bu işlem daha önce kurulan ve projeye dahil edilen GPSD kütüphaneleri sayesinde yapılmaktadır. Hız verisi alındıktan sonra sistemde belirtilen şehir içi ve şehir dışı hız limitleri ile karşılaştırılabilir hale getirmek için tip dönüşümleri yapılmaktadır.

2.1.5 Aşım Durumunda Veri Tabanına Kayıt

Bu çalışmada veri tabanı olarak SQLite kullanılmıştır. Her modül için ayrı bir tablo oluşturulmuş ve bu tablolara kayıtlar yapılmaktadır. GPS modülünden alınan hız verisi, şehir içindeyken şehir içi hız limitleriyle, şehir dışındaysa şehir dışı hız limitleriyle karşılaştırılmaktadır.

Hız sınırlarının ihlali durumunda anlık olarak konum, tarih, aşım miktarı ve ceza puanı veri tabanına daha sonradan incelebilir halde tutmak için kayıt edilmektedir.

2.2 Ani İvmelenmeler

Sürücünün agresifliğini belirten en önemli unsurlardan birisi de ani frenleme veya hızlanmalardır. Yapılan araştırmalar sonucu güvenli ivmeler hesaplanmıştır **[24]**. Şekil 4'te ise bu ivmelerin değerleri gösterilmiştir.

Deceleration g's

g's	mph / sec	Braking
0.30	6.6	Safe
0.35	7.7	Safe
0.47	10.3	Average Driver Max
0.62	13.6	Reasonably Skilled Driver Max
0.66	14.3	Skilled Driver Max
0.70	15.4	Vehicle Max
1	21.9	

Şekil 6 Güvenli İvme Değerleri

Yapılan bu araştırmaya göre ortalama kapasitedeki bir sürücü için ivme değeri 10,3 mph/sn'dir. Profesyonel bir sürücü için bu değer 14,3 mph/sn'ye kadar çıkarken araç için güvenli değer ise 21,9 mph/sn olarak belirlenmiştir.

Bu çalışmada güvenli bir trafik oluşturmak için ortalama sürücü değerleri ele alınmaktadır. 10,3 mph/sn değerini kullandığımız formata çevirecek olursak 4,6 m/s^2 olmaktadır.

GPS modülünden elde edilen veriler sayesinde belirli bir süre zarfındaki hız değişimi hesaplanarak geçen süreye bölündüğünde ivme elde edilmektedir. Bu ivme değeri standart olarak belirlenen 4,6 m/s^2 değeriyle karşılaştırıldıktan sonra ani ivme tespit edilmektedir.

Şayet ani bir frenleme veya hızlanma söz konusu ise veri tabanının ilgili tablosuna aşım miktarı, tarih ve ceza puanı olarak kayıt edilmektedir.

2.3 Trafik Işığı İhlalleri

Kaza nedenlerinin büyük çoğunluğunu oluşturan trafik ışığı ihlalleri, gerçekleştirilen çalışmada Raspberry Pi'ye doğrudan bağlanan bir USB kamera üzerinden elde edilen görüntülerde görüntü işleme teknikleri kullanılarak tespit edilmektedir. Bu aşamalar alt başlıklar halinde belirtilmektedir.

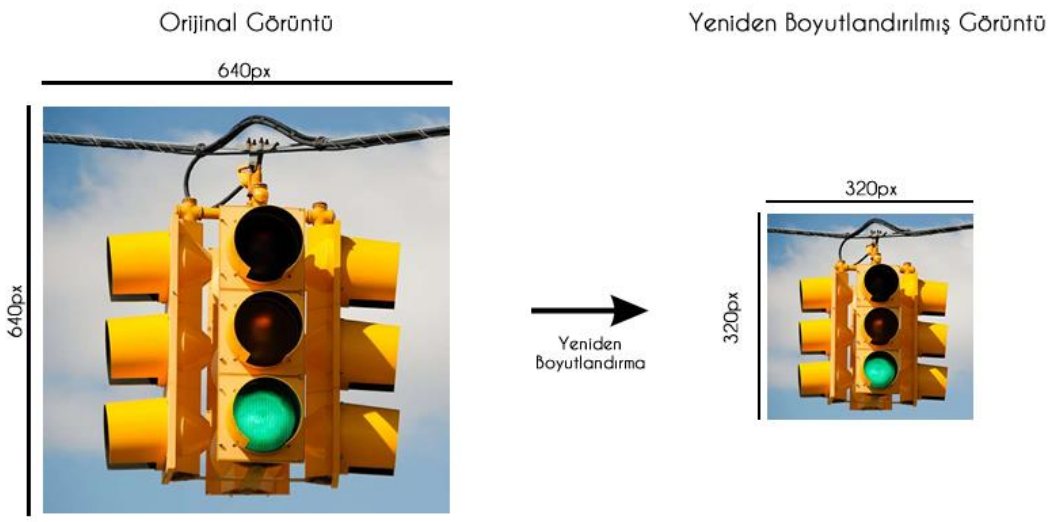
2.3.1 OpenCV Kütüphaneleri

OpenCV açık kaynak kodlu bir görüntü işleme kütüphanesidir. Görüntü üzerinde birçok hazır fonksiyonla işleme yapmaya ortam hazırlamaktadır. Trafik ışıklarının tespit edilmesi için görüntü işleme tekniklerinden yararlanılırken bu kütüphane kullanılmıştır. OpenCV kütüphanesi sayesinde USB kamera üzerinden anlık olarak görüntü alınabilmesine kırmızı, sarı ve yeşil trafik ışıkları için ayrı maskeler tanımlanabilmesine ve bu maskelenen görüntüler üzerinde Hough algoritmasının kullanılabilmesine olanak sağlayarak trafik ışığı tespitinin yapılabilmesine yardımcı olmuştur.

2.3.2 Orijinal Görüntünün Alınması ve Boyutlandırılması

Raspberry gibi küçük işlemci kapasiteli cihazlarda hafıza yönetimin ve işlem yükünün minimum seviyeye indirgenmesi hayati önem taşımaktadır. Bu nedenle gerçekleştirilen çalışmada, özellikle görüntü işleme alanında bu konu üzerinde değişiklikler yapılmak zorundadır.

OpenCV kütüphanesi sayesinde USB kamera üzerinden alınan anlık görüntüler, kolay işlenebilmesi ve işlemciyi yormamak adına küçültülmektedir. Bu küçültme oranı, kullanılan cihazın özelliklerine bağlı olarak değişebilse de projede kullanılan Raspberry Pi 3 için yarı yarıya olarak belirlenmiştir. Bu işlem sonucunda alınan görüntü yeniden boyutlandırılarak genişlik ve yükseklik yarıya indirilmektedir.



Şekil 7 Görüntüyü Yeniden Boyutlandırma

Şekil 5'te görüldüğü üzere orijinal görüntü işleme kapasitesini zorlayabilecek kadar büyük görüntüler elde edilebilecektir. Bu nedenle yeniden boyutlandırılmış ve yarı yarıya daha küçük bir görüntü elde edilmiştir.

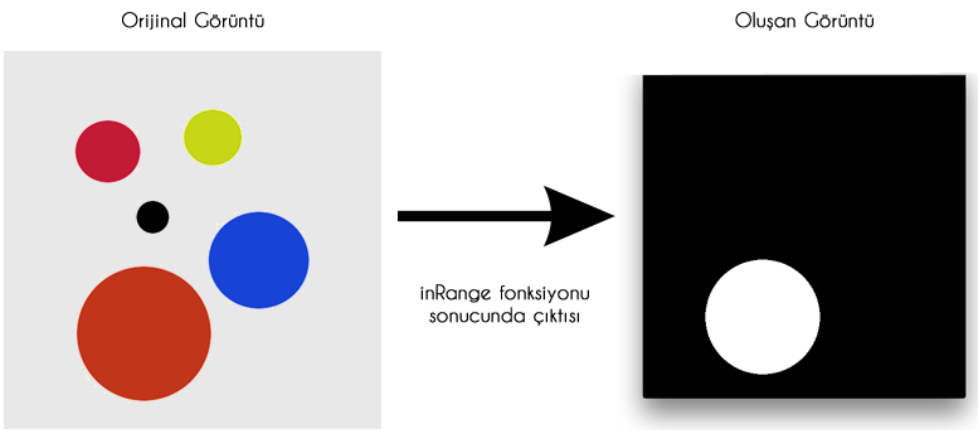
2.3.3 Maskeleme İşlemi ve Görüntüye Uygulanması

Her görüntü (image) bilgisayar ve benzeri ortamlarda birer matris olarak tutulmaktadır. Bu matrislerin anlamlı bir şekilde değerlendirilip işlenmesi ise görüntü işleme olarak adlandırılmaktadır. İmge matrislerinin her hücresi bir piksel değerine düşen renk değerlerinin verilerini içermektedir. Bu renk değerleri yapılacak çalışmaya göre işlenip değiştirilebilir.

Maskeleme işlemi de görüntü işleme tekniklerinden biridir. Bu işlemde genellikle istenilen görüntünün arka plandan ayrılması sağlanmaktadır. İstenilen renk aralıklarına ait pikseller alınırken, istenmeyen arka plan renkleri ayırt edilebilmektedir.

Trafik ışıklarını tanımak için en spesifik özellik renklerdir. Bu nedenle ilk önce renk değerlerine bakılmaktadır. Bu renk değerlerinin tanınması ve ayırt edilmesi için öncelikle tanınması istenen renkler, renk aralıkları şeklinde belirtilmeli ve sisteme tanınmalıdır. Renk aralıkları HSV renk uzayında belirlenmiş olup bu aralıkların uygulanabilmesi için USB kameradan alınan ve yeniden boyutlandırılan görüntü HSV formatına dönüştürülmüştür.

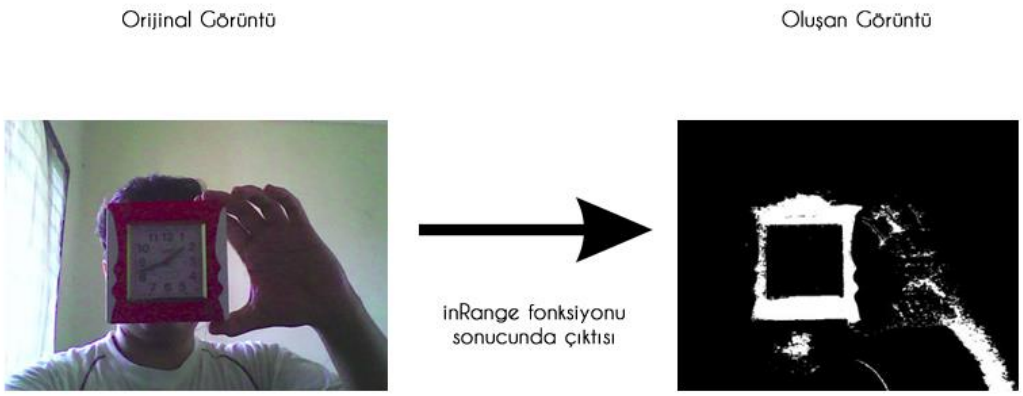
Her bir trafik ışığı için ayrı maskeler kullanılmış olup bu maskeler HSV formatına çevrilmiş imgeye OpenCV'nin "inRange" fonksiyonu ile uyarlanmıştır.



Şekil 8 Maskeleme İşlemi Çıktısı

Şekil 6’da görüldüğü üzere istenilen görüntü eğer kırmızı ışıksa uygun maskeler kullanılarak HSV formatında fonksiyona verilmekte ve çıktı olarak sadece belirlenen aralıktaki renklerin bulunduğu alan binary görüntü olarak geri döndürülür.

Teorik olarak bu işlem çalışsa da gerçek hayattan alınan görüntülerde ortamda bulunan diğer kırmızı nesneler de bu binary görüntünün içinde gürültü olarak bulunacaktır.



Şekil 9 Maskeleme Gürültüleri

Şekil 7’de görüldüğü üzere sadece kırmızı alan alınmak istendiğinde ortamda bulunan ve o renk aralığında pikseller içeren nesneler gürültü olarak imgenin içinde bulunacaktır.

Bu gürültülerden kurtulmanın birçok yolu vardır. Bu projede bu soruna çözüm olarak oluşan binary imgeden sadece daire içeren şekillerin alınması yöntemi kullanılmıştır.

2.3.4 Hough Dönüşümü ve Daire Tespiti

Dijital görüntü işleme tekniklerinde geometrik şekilleri tespit etmek için birçok algoritma vardır. Bu algoritmalar sayesinde bir imgenin içinde bulunan daire, dikdörtgen, doğru gibi geometrik şekillerin çıkarılması mümkündür. Bu çalışmada ise Hough dönüşümü, daire tespiti için kullanılmıştır. USB kamera sayesinde elde edilen imgeler önce yeniden boyutlandırıldıktan sonra maskelenmektedir. Bu maskeleme sayesinde istenen trafik ışığının rengine ait alanlar binary imgeye dönüştürülmektedir.

Maskeleme işleminden çıkan görüntü ise, trafik ışığını oluşturan daire şeklindeki ışıkları tespit etmek için Hough dönüşüm algoritmasına sokulur ve istenilen renkteki trafik ışıkları bir dizide tutulur.

Hough dönüşümünün genel adımları aşağıdaki gibidir;

- Alınan imge üzerindeki kenarlar çıkarılır,
- Bir eşikleme yöntemi kullanılır ve görüntü siyah beyaz hale getirilir,
- Her kenar pikseli için noktanın üzerinde olabileceği olası geometrik şekillerin polar koordinattaki değerleri kullanılan bir akümülatör matrisi üzerinde birer artırılarak her kenar pikselin olası şekilleri oylaması sağlanmış olur.
- Akümülatör değeri en yüksek olan şekiller en çok oy alan şekiller olduklarından görüntü üzerinde bulunma veya belirgin olma olasılıkları en yüksek olmaktadır.
- Bulunan şekiller isteğe bağlı olarak görüntü üzerine yazdırılabilir [25].

$$r^2 = (x - a)^2 + (y - b)^2$$
$$x = a + r \times \sin(\theta)$$
$$y = b + r \times \cos(\theta)$$

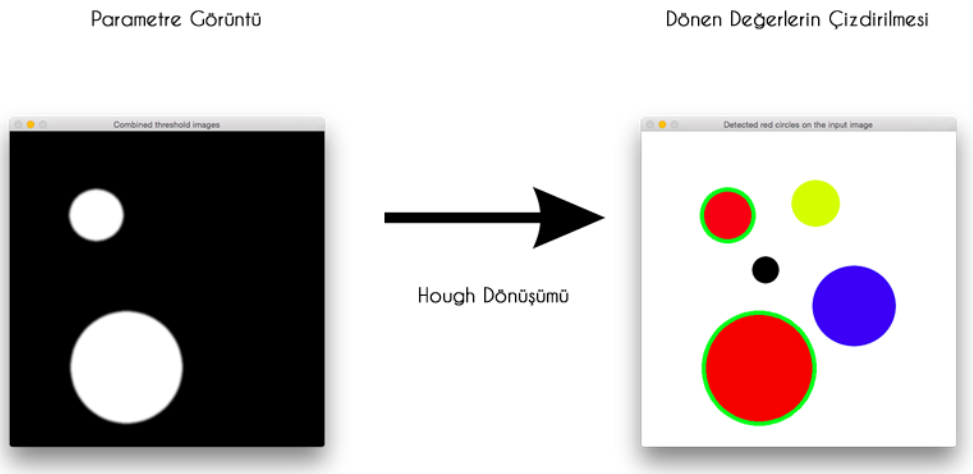
Şekil 10 Merkezi Çember Denklemi

Hough algoritmasında aranan şekil daire ise kullanılan akümülatör matrisi, dairenin merkezi olan ve (a,b) koordinatına sahip bir nokta ve yarıçapı (r) olmak üzere 3 boyuttan oluşmaktadır. Bu yarıçap algoritmaya girdi olarak verilir ve bu yarıçap aralığındaki daireler

aranır. Daireler aranırken merkezi çember denkleminden faydalanılarak bulunmaktadır (Şekil 8).

Hough dönüşümü sayesinde daire tespiti, OpenCV kütüphanesinde "HoughCircles" fonksiyonuyla sağlanmaktadır. Bu fonksiyon yanıt olarak, içinde aranan dairelerin merkezlerinin koordinatı ve yarıçap uzunluklarının bulunduğu bir dizi döndürmektedir.

Hough daire dönüşümü ile tespit edilen dairelerin renklerine göre o anda yanan ışık tespit edilmiş ve ona göre algoritmanın işleyişi devam ettirilmiştir.



Şekil 11 Hough Dönüşümü

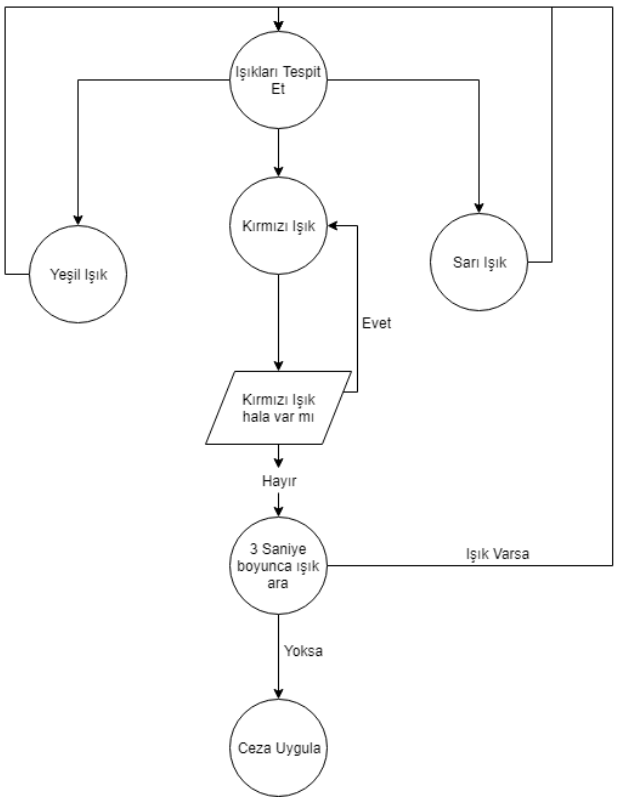
"HoughCircles" fonksiyonundan dönen değerler tek tek orijinal görüntünün üzerinde çizilmesi Şekil 9'da gösterilmektedir. Bu çizim ise yine OpenCV kütüphanesinin "circle" fonksiyonu ile gerçekleştirilmektedir.

Şekil 12 Trafik Işığı Tespiti Çıktısı

Şekil 10'da ise aynı algoritmanın trafik ışığı görüntüsüne uygulandıktan sonraki çıktısı verilmektedir. USB kamera üzerinden alınan veriler işlenerek ışık tespiti Şekil 10'daki gibi yapılmıştır.

2.3.5 İhlal Tespiti

Görüntüden tespit edilen ışıklardan sonra işleyecek olan algoritmada, aracın kırmızı ışıkta geçip geçmediği tespit edilmelidir. Bu çalışmada bu tespit Şekil 11'deki akış diyagramıyla açıklanabilir.



Şekil 13 Işık İhlali Tespiti Akış Diyagramı

Program çalışmaya başladığı andan itibaren sonsuz bir döngü içine girer ve anlık görüntü almaya başlar. Bu görüntüler içinde sarı veya yeşil ışık tespit edildiği zaman herhangi bir denetime maruz kalmadan yeniden görüntü taranır. Kırmızı ışık tespit edilene kadar bu şekilde devam edilir.

Şayet bir kırmızı ışığa rastlanırsa aracın durması gerekmekte ve o kırmızı ışığın hep görüntüde kalması gerekmektedir. Bu nedenle kırmızı ışık tespit edildiğinde bir iç döngüye girer ve kırmızı ışığın sönüp sönmediği kontrol edilir. Kırmızı ışık kaybolmadığı sürece bu döngü devam eder.

Kırmızı ışık tespit sınırlarından çıkarsa, kırmızı ışık ihlali mi yoksa ışık değişimi mi olduğu belirlenmelidir. Görüntülerin alınması durumunda anlık görüntü kaybolmaları veya anlık tanıma kaybolmaları yaşanabilmektedir. Bunun önüne geçmek için 3 saniyelik bir zamanlayıcıdan yardım alınmıştır. Kırmızı ışığın kaybolduğu 3 saniye içinde kırmızı ışık aranır. Kırmızı veya herhangi bir ışığın bulunamadığı durumda kırmızı ışıktan geçilmiş olarak ele alınır.

Eğer bu 3 saniyelik zaman diliminde yeşil ışık veya sarı ışığa denk gelinirse iç döngüden çıkılır ve ana döngüde tespit aşaması devam edilir. Yeşil veya sarı değil de kırmızı ışık tespit edilirse iç döngüde kalınır ve ışık tespit edilmeye devam edilir.

2.3.6 İhlalin Kayıt Altına Alınması

İhlalin tespiti gerçekleştikten sonra veri tabanının uygun tablosuna ceza kaydı düşürülür. Bu kayıtta ceza puanı ve tarih yer almaktadır. Bu ceza sistemdeki en ağır ceza olarak belirlenmiştir.

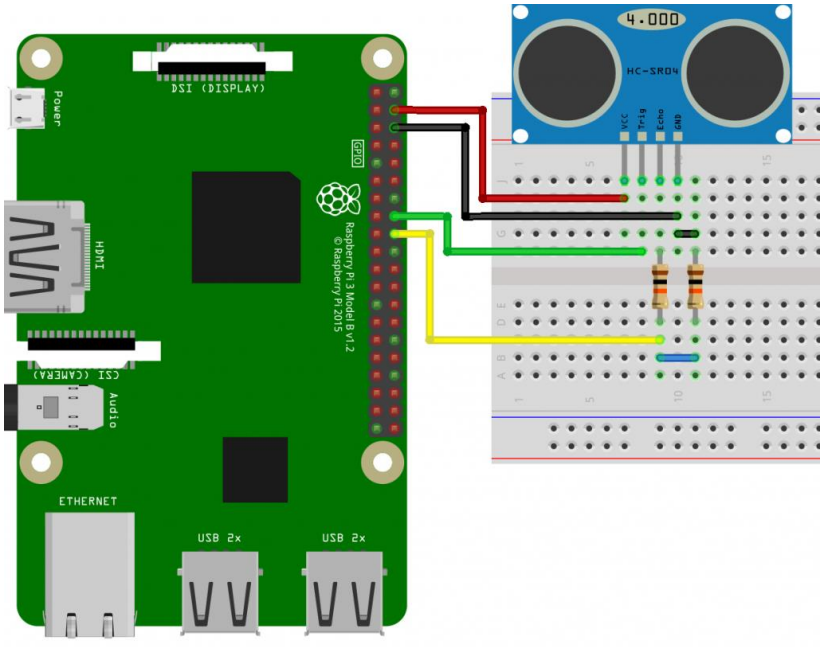
Sistemde bulunan ceza puanları göreceli olarak belirlenmiş olup sistem içinde değiştirilebilmektedir. Kullanılacak şehirde veya ülkedeki kurallara göre yeniden ayarlanabilmektedir.

2.4 Çarpışma Kontrolü

Denetlenmekte olan bir diğer başlık olan çarpışma kontrolü, sürücünün park etme esnasında veya akıcı trafik anında herhangi bir yüzeyle yapılan temasın tespiti için yapılan denetlemedir. Bunun kontrolünün yapılması için Raspberry Pi'ye entegre edilen bir mesafe sensoru kullanılmıştır. Mesafe sensoru olarak "HC-SR04 Ultrasonik Mesafe Sensoru" kullanılmıştır. Tespit aşamaları alt başlıklar halinde gösterilmektedir.

2.4.1 Mesafe Sensoru – Raspberry Pi Bağlantısı

Mesafe sensorundan veri elde edilebilmesi için Raspberry Pi ile bağlantısının doğru bir şekilde yapılması gerekmektedir. Bu çalışmada kullanılan bağlantı şekli Şekil 12'de gösterilmiştir.



Şekil 14 Mesafe Sensoru - Raspberry Pi Bağlantısı

Sensorun VCC pini Raspberry Pi'deki 3v çıkışına, GND pini topraklama pinine, TRIG pini 23 numaralı, ECHO pini ise 24 numaralı GPIO pinine bağlanmaktadır.

2.4.2 Mesafe Sensorundan Veri Elde Edilmesi

Bağlantının başarılı bir şekilde yapılmasının ardından mesafe sensorundan veri elde edilmesi gerekmektedir. Bu sensor gönderilen dalga ile alınma zamanı arasında geçen süreye göre mesafe tespiti yapmaktadır. İki zaman arasında geçen süre "17150" ile çarpıldığında aradaki mesafe santimetre cinsinde elde edilmiş olmaktadır.

Her sensorun algılayabildiği mesafe aralığı farklıdır. Bu çalışmada kullanılan sensor 2cm ile 4m arası uzaklıkları ölçebilecek hassasiyete sahiptir. Bu nedenle 2cm altındaki uzaklıkların hepsi çarpma olarak ele alınmıştır.

2cm'den düşük uzaklıklar algılandığında bir miktar hata payı olsa da veri tabanına ceza puanı ile birlikte kayıt edilmektedir.

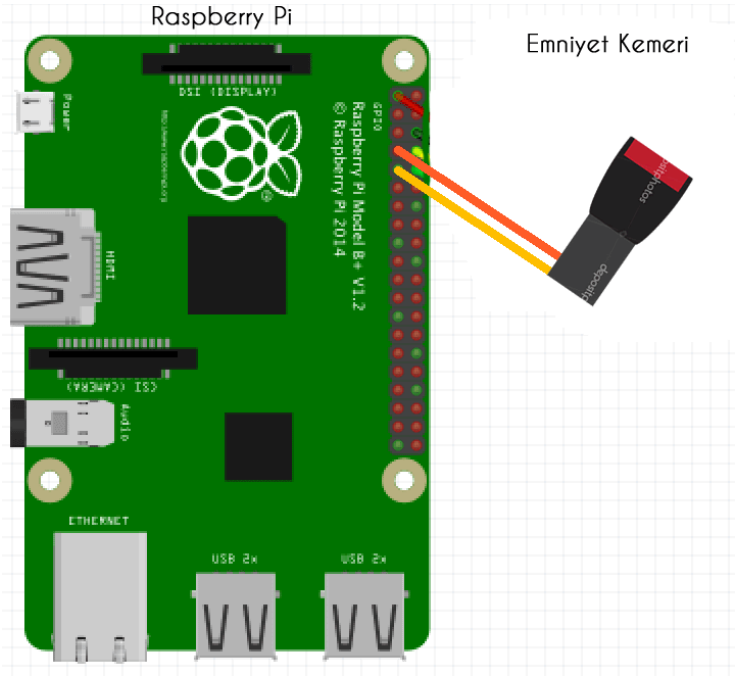
Daha hassas sensorların kullanılmasıyla daha doğru sonuçlar elde edilecektir.

2.5 Emniyet Kemeri Kontrolü

Trafik kazalarında ölümlerin en başta gelen nedenlerinden biri emniyet kemeri kullanmadan araç kullanmaktır. Bu çalışmada bu etkenin de kontrol edilmesi sağlanmaktadır.

2.5.1 Emniyet Kemeri – Raspberry Bağlantısı

Emniyet kemerinin takılıp takılmadığını kontrol edebilmek için öncelikle kemer yuvası ile Raspberry Pi'nin bağlantısı yapılması gerekmektedir. Bu bağlantı Şekil 13'te gösterilmektedir.



Şekil 15 Emniyet Kemeri - Raspberry Pi Bağlantısı

Şekil 13'te de görüldüğü gibi bir giriş 7 numaralı pine bağlanırken diğeri GND pinine bağlanmıştır. Emniyet kemerinin içindeki switche bağlı olan pinler sayesinde emniyet kemeri kontrolü yapılabilecektir.

2.5.2 Emniyet Kemeri İhlali Kontrolü ve Kaydı

Emniyet kemerlerinin yuvaları içinde emniyet kemerinin takılı olup olmadığının kontrolünün yapılabilmesi için bir switch konumlandırılmıştır. Bu switch emniyet kemerinin takılması durumuna göre tıpkı bir anahtar gibi açık veya kapalı duruma geçmektedir. Switchin iki bacağını Raspberry pinlerine bağlayarak bu kontrolü Raspberry üzerinde yapmak mümkündür.

Bu çalışmada her 10 saniyede bir kontrol yapılmakta ve ihlal durumunda veri tabanının uygun tablosuna ceza puanı ile birlikte kayıt edilmektedir. Aynı şekilde bir kolon da tarihe ayrılmış ve ihlallerin zamanları da tutulmaktadır.

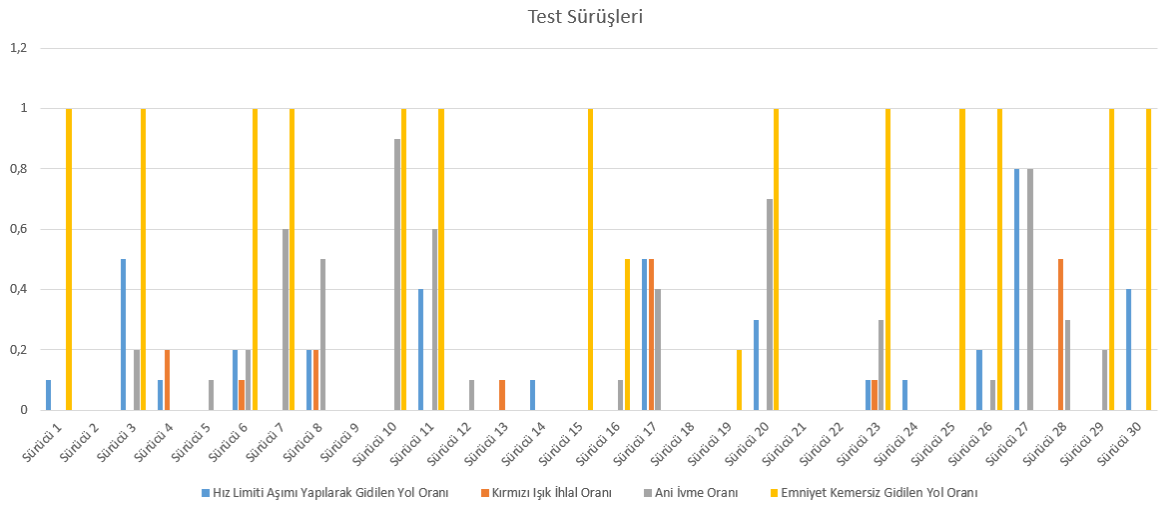
3. DENEYSEL SONUÇLAR

Bu çalışmada 30 farklı sürücü tipiyle sürüşler gerçekleştirilmiştir. Test sürüşleri Şekil 16'da gösterdiği alan üzerinde yapılmıştır. Test sürüşü alanı yaklaşık 3,5km olup yol üzerinde 10 adet trafik ışığı, düz yol, yokuş yukarı veya çeşitli etmenlerin bulunduğu bir güzergâh seçilmiştir.

Şekil 16 Test Sürüşü Alanı

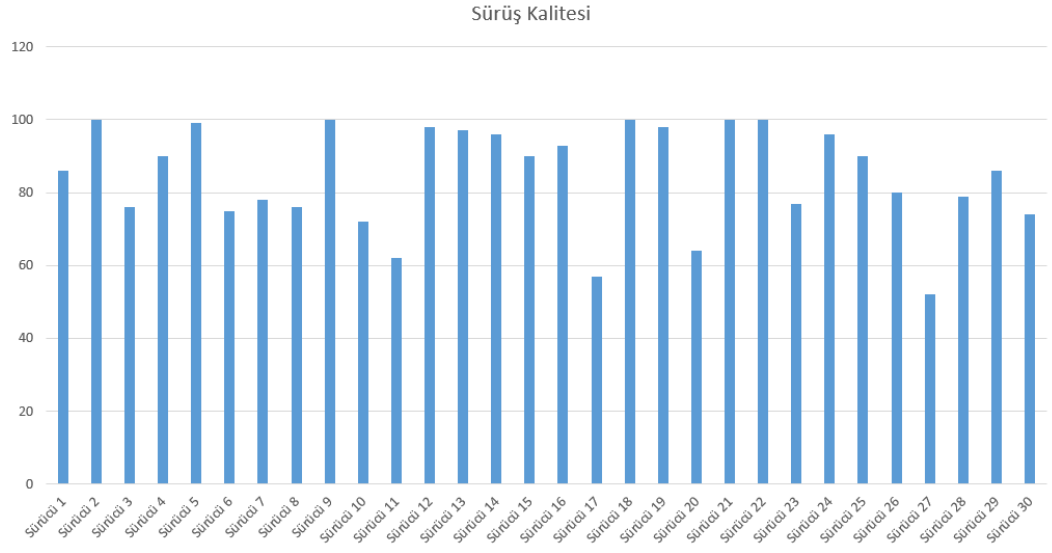
Test sürüşü boyunca yaklaşık 20'şer defa dur-kalk yapılmıştır. Ani frenleme ve hızlanmaların oranı buraya göre yapılmaktadır. 10 farklı trafik ışığından geçilmiş ve oranlama bu 10 ışığa göre kurulmuştur. 3,5km yol gidilmiş olup bu yol boyunca hız limitlerinin aşılarak gidildiği yolun tamamına oranı alınmaktadır. Ve aynı şekilde emniyet kemersiz gidilen yol tamamına oranlanmaktadır.

Bu oranların tamamı daha sonra kalitenin hesaplanması için tutulmuştur. Kalitenin hesaplanması için bu oranların ağırlık katsayılarıyla çarpılıp bir toplam değer elde edilmelidir. Şekil 17'de bu oranlar her bir sürücü için ayrı ayrı gösterilmektedir. Bu grafikte verilen veriler ekler bölümüne de eklenmiştir.



Şekil 17 Test Sürüşlerinin Hata Oranları

Daha önceden belirlenen maliyet fonksiyonuna (cost function) deneysel olarak toplanan oranlar koyularak her sürücü için sürüş kalitesi yüzdesi çıkarılmıştır. Bu maliyet fonksiyonunda her bir hatanın bir ağırlık katsayısı bulunmaktadır. Ağırlık katsayısıyla çarpılan oranların toplanmasıyla yüzdelik bir dilim elde edilmektedir. Bu yüzdelik dilim sürücünün kalitesini belirtmekte ve sistemin çıktısı olmaktadır. Her sürücü için hesaplanan kalite yüzdeleri Şekil 18'de belirtilmektedir.



Şekil 18 Sürücüler - Sürüş Kalitesi İstatistiği

Test edilen sürücü davranışlarına göre sürücülerin birçoğu %80 barajının üzerine geçerken bir kısmı ise altında kalmıştır.

Bu veriler kısıtlı bir yol haritası ve dar bir zamanda elde edildiğinden doğruluk payı kısmen düşüktür. Daha anlamlı çıktılar elde etmek için daha uzun süreli test sürüşleri yapılmalı ve daha fazla veri toplanmalıdır.

Bu çalışmada verilerin elde edilmesi ve sürüş kalitelerinin çıkarımı başarılı bir şekilde tamamlanmış ve sürücülerin sınıflandırılması sağlanmıştır.

4. SONUÇLAR VE TARTIŞMA

Son yıllarda teknoloji çağının da getirdiği imkânlarla motorlu araçların kullanımı küçümsenemeyecek kadar artmıştır. Bu artışın getirdiği sorunların başında ise yoğun bir trafik gelmektedir. Yoğun trafiğin içindeki araçlar yılda yüz binlerce kez trafik kazasına maruz kalmaktadır. Yaşanan trafik kazalarının ana nedeni de sürücü kaynaklı hatalar, kural ihlalleri ve dikkatsizliktir. Bu nedenle sürücülerin kontrol altında tutulması ve hatalarının denetlenmesi gerekmektedir.

Trafik kazalarının seneden seneye katlanarak artması bu konuda yapılan çalışmaların veya önlemlerin yeterli düzeyde olmadığını açıkça göstermektedir. Bu çalışmalar ya başarılı bir şekilde hayata geçirilememiş ya da yetersiz kalmıştır. Gelişen teknolojide bunun denetlenmesi daha kolay hale getirilebilir.

Güvenlik kameraları gibi belirli alanlarda bulunan ve herkes için genel bir önlem almak yerine problemin özelleştirilmesi gerekmektedir. Bilgisayar ve akıllı sistemler hayatımızın her alanına girmişken akıllı araçların içinde bu denetimin yapılmasında hiçbir engel bulunmaktadır. Bu çalışma, denetimin herkese açık ve belirli noktalarda yapılması yerine çözüm ortamının özele indirgenmesiyle araç içinde çözülmesini savunmaktadır.

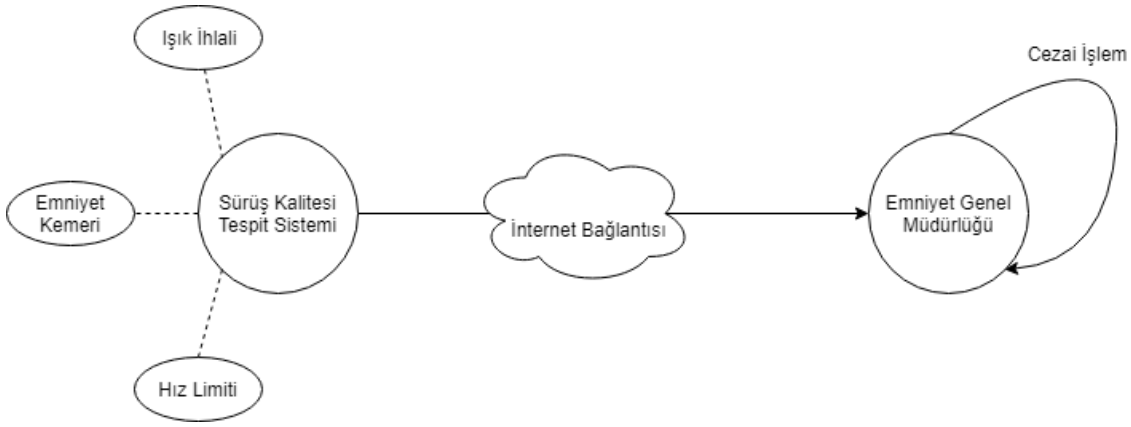
Aracın içinde gömülen Raspberry Pi'ye bağlı olarak GPS, kamera ve sensorlar sayesinde bu denetim kolaylıkla yapılabilecektir. Bu denetim sayesinde aşağıdaki maddelerin kontrolü yapılacaktır;

- Hız Tespiti
- Ani Frenleme ve Hızlanma Tespiti
- Kırmızı Işık İhlali
- Emniyet Kemeri
- Çarpışma Kontrolü

Bu sayede sürücünün yetkinliği belirlenmiş olacaktır.

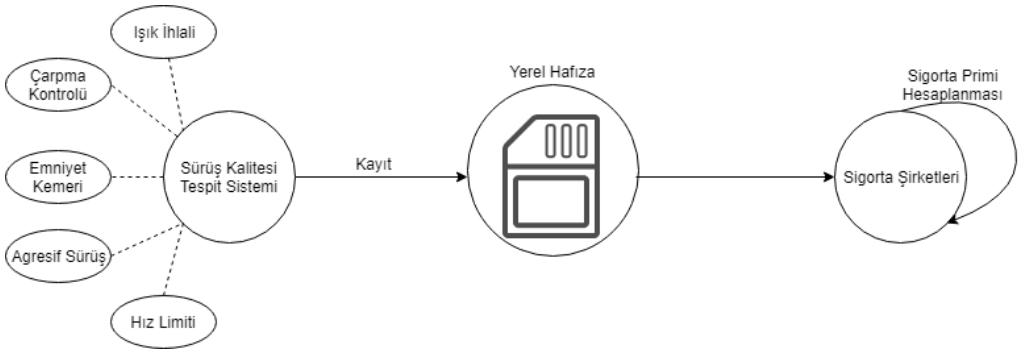
4.1 Çalışmanın Gerçek Hayata Uyarlanması

Sürücü için belirli parametrelere bakılarak Toplanan verilerin nasıl kullanılabileceği Şekil 15 ve 16'da gösterilmiştir.



Şekil 19 Sistem İle Emniyet Müdürlüğü Bağlantısı

Şekil 15'te bu sistemde toplanan verilerin, anlık olarak bir internet bağlantısıyla Emniyet Genel Müdürlüğü ile bağlantıya geçilip gönderilmesi önerisinin modellenmesi gösterilmektedir. Bu sistem sayesinde ceza karşılığı olması gereken tespitlerin sisteme düşürülmesi sonrasında cezai işlem uygulanabilecektir.



Şekil 20 Sistem ile Sigorta Şirketleri Bağlantısı

Bir diğer öneri ise Şekil 16'da görüldüğü gibi sistemde toplanan verilerin, aracın sigortası yapılırken sigorta şirketine teslim edilmesidir. Bu verileri teslim alan sigorta şirketleri sürücünün daha önce yaptığı kazalardan ziyade sürüş kalitesine göre bir ücret hesaplaması yapabilmelidir.

Tüm bu caydırıcı unsurlar, sürücünün daha bilinçli hareket etmesini sağlayacaktır. Sadece şehir içinde kurallara uymak yerine her alanda kurallara uyulmasını, sadece radarların bulunduğu yerde yavaşlaması yerine trafiğin bütününde limitlerin aşılmaması sağlanacaktır.

4.2 Belirlenen Plandan Sapmalar

Proje planlama aşamasındayken uygulanacak metotlar belirlenmiştir. Fakat belirlenen plana tamı tamına uymak her zaman mümkün olmamaktadır. Bu çalışmada ise planlandığı gibi gitmeyen gelişmeler yaşanmıştır.

Trafik ışıklarının tespiti yapılırken saf görüntü işleme tekniklerinden ziyade eğitilmiş verilerle çalışılması hedeflenmekteydi. Bunun için OpenCV kütüphanesinin bir parçası olan "haarcascade" şablonlarının kullanılması planlanmaktaydı.

Bunun için yüzlerce trafik ışığı imgesinden oluşan büyük çaplı veriler toplandı. Bu veriler üzerinde derin öğrenme algoritmaları kullanılarak sistem eğitildi. Çıkarılan haarcascade ile denemeler yapıldı. Fakat bu denemeler sırasında sistemin verimli çalışmadığı belirlenmiştir.

Bu verimsizliğin en önemli nedeni olarak kullanılan Raspberry Pi'nin işlem kapasitesinin düşük olduğu kanaatine varılmıştır. Bu nedenle saf görüntü işleme teknikleri kullanılarak sistemin kendine has algoritmasıyla tespit edilmeye çalışılmıştır.

4.3 Yenilikler

Literatürdeki çalışmalara bakılacak olursa genel olarak, mobil cihazlar, direksiyon hareketleri veya CAN bus verileri kullanılarak bu tespit yapılmıştır. Bu çalışmada ise bir

Raspberry Pi kullanılmıştır. Raspberry Pi ile de bu tespitin başarılı bir şekilde yapılabileceği kanıtlanmıştır.

Yapılan çalışmalar içinde görüntü işleme teknikleri kullanılmış olsa da bu teknikler genelde şerit tanıma gibi modüller için kullanılmıştır. Trafik ışıklarını tanıyan bir sistem bulunamamıştır. Bu çalışmada ise trafik ışıkları tanınmaktadır.

5. ÖNERİLER

Agresif sürüş veya sürüş kalitesi tespiti yapılması durumunda dikkat edilmesi gereken bazı hususlar tespit edilmiştir. Bunlar aşağıda maddeler halinde belirtilmiştir.

- Şayet görüntü işlenecekse öncelikle geniş açılı bir kamera seçilmelidir. Bu çalışmada kullanılan kameranın açısı çok dar olduğundan konumlandırma açısından zorluk çekilmektedir.
- Alınan görüntü üzerinde optimizasyon işlemleri dikkatli bir şekilde yapılmalıdır. Örneğin küçültme işleminde aşırıya kaçılırsa tanıma işlemleri verimli çalışmayacaktır. Ya da yeterine küçültülmemesi işleme esnasında gecikmelere yol açacaktır.
- Bir GPS modülü kullanılacaksa bu modüle giden akım önemlidir. Raspberry Pi'ye verilen akım düşük olduğunda Raspberry'nin çalışmasında herhangi bir problem olmazken GPS modülünden veri çekilememe sorunu ortaya çıkmaktadır. Bu sorunun kaynağını araştırmak çok fazla zaman kaybı yaşatmıştır.
- Özellikle görüntü işleme kullanılacaksa işlenecek cihazın işlem kapasitesi iyi belirlenmelidir. Görüntü işleme işlemciye ağır bir yük bindirdiği için bu işlem kapasitesi yüksek cihazlar seçilmelidir.
- Sensorlarla iletişimi sağlamadan önce dirençlerin doğru bir şekilde bağlanması önemlidir. Yanan sensorların geri dönüşümü sağlanamamaktadır.

KAYNAKLAR

[1] «Injury Deaths Rise in Rank,» WHO, 2008. https://www.who.int/violence_injury_prevention/key_facts/VIP_key_fact_3.pdf.

[2] «Karayolu Trafik Kaza İstatistikleri,» TÜİK, Temmuz 2017. http://www.tuik.gov.tr/PreHaberBultenleri.do?id=27668.

[3] A. Aljaafreh, N. Alshabatat ve M. N. Al-Din, «Driving style recognition using fuzzy logic,» *IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012),* Istanbul, Turkey, 2012.

[4] H. Eren, S. Makinist, E. Akin ve A. Yilmaz, «Estimating driving behavior by a smartphone,» *IEEE Intelligent Vehicles Symposium,* Alcala de Henares, Spain, 2012.

[5] D. A. Johnson ve M. M. Trivedi, «Driving style recognition using a smartphone as a sensor platform,» *14th International IEEE Conference on Intelligent Transportation Systems (ITSC),* Washington, DC, USA, 2011.

[6] O. Karaduman, H. Eren, H. Kurum ve M. Celenk, «An effective variable selection algorithm for Aggressive/Calm Driving detection via CAN bus,» *International Conference on Connected Vehicles and Expo (ICCVE),* Las Vegas, NV, USA, 2013.

[7] S. M. R. Noori ve M. Mikaeili , «Driving Drowsiness Detection Using Fusion of Electroencephalography, Electrooculography, and Driving Quality Signals,» *Journal of Medical Signals and Sesors,* 2016.

[8] T. Imkamon, P. Saensom, P. Tangamchit ve P. Pongpaibool, «Detection of hazardous driving behavior using fuzzy logic,» *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology,* Krabi, Thailand, 2008.

[9] J. Dai, J. Teng, X. Bai, Z. Shen ve D. Xuan, «Mobile phone based drunk driving detection,» *4th International Conference on Pervasive Computing Technologies for Healthcare,* Munich, Germany, 2010.

[10] F. Li, H. Zhang, H. Che ve X. Qui, «Dangerous driving behavior detection using smartphone sensors,» *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC),* Rio de Janeiro, Brazil, 2016.

[11] C. Zhongyang, Y. Jiadi, Z. Yanmin, C. Yingying ve L. Minglu, «D3: Abnormal driving behaviors detection and identification using smartphone sensors,» *12th*

Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Seattle, WA, USA, 2015.

[12] C. Sin-Yu ve H. Jun-Wei, «Edge-based Lane Change Detection and its Application to Suspicious Driving Behavior Analysis,» *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, Kaohsiung, Taiwan, 2007.

[13] A. Eskandarian ve A. Mortazavi, «Evaluation of a Smart Algorithm for Commercial Vehicle Driver Drowsiness Detection,» *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, 2007.

[14] H. Wang , H. Liu ve Z. Song , «Fatigue Driving Detection System Design Based on Driving Behavior,» *International Conference on Optoelectronics and Image Processing*, Haikou, China, 2011.

[15] W. Xinrong , . Z. Junwei, A. Jinghe ve Y. Yanchao , «Abnormal driving behavior detection for bus based on the Bayesian classifier,» *Tenth International Conference on Advanced Computational Intelligence (ICACI)*, Xiamen, China, 2018.

[16] . S. Hashim, T. Saeed, M. Tahir ve M. Uppal, «Risky Driving Behavior Detection Using In-Vehicle WiFi Signals,» *IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, IL, USA, USA, 2018.

[17] M. N. Husen , S. Lee ve M. Q. Khan, «Syntactic pattern recognition of car driving behavior detection,» *IMCOM '17 Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, Beppu, Japan, 2017.

[18] L. Liu, C. Karatas, H. Li, S. Tan, M. Gruteser, J. Yang, Y. Chen ve R. P. Martin , «Toward Detection of Unsafe Driving with Wearables,» *WearSys '15 Proceedings of the 2015 workshop on Wearable Systems and Applications*, Florence, Italy, 2015.

[19] A. Găvrută, M. Marcu ve R. Bogdan, «Software Solution for Monitoring and Analyzing Driver Behavior,» *IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, Romania, 2018.

[20] G. Castignani, T. Derrmann, R. Frank ve T. Engel, «Driver Behavior Profiling Using Smartphones: A Low-Cost Platform for Driver Monitoring,» *IEEE Intelligent Transportation Systems Magazine*, 2015.

[21] Y. Zheng ve J. H. Hansen, «Unsupervised driving performance assessment using free-positioned smartphones in vehicles,» *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016.

[22] R. d. Charette ve F. Nashashibi, «Real time visual traffic lights recognition based on Spot Light Detection and adaptive traffic lights templates,» *IEEE Intelligent Vehicles Symposium*, Xi'an, China, 2009.

[23] M. Omachi ve S. Omachi, «Traffic light detection with color and edge information,» *2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, China, 2009.

[24] «Acceleration Parameters,» Copradar, https://copradar.com/chapts/references/acceleration.html.

[25] «Hough dönüşümü ile dairesel şekil tespiti,» http://www.isikdogan.com/turkce-blog/hough-donusumu-ile-dairesel-sekil-tespiti.html.

EK – 1
Test Sürüşünden Elde Edilen Veriler

Sürücüler	Hız Limiti Aşımı Yapılarak Gidilen Yol Oranı	Kırmızı Işık İhlal Oranı	Ani İvme Oranı	Emniyet Kemersiz Gidilen Yol Oranı
Sürücü 1	0,1	0	0	1
Sürücü 2	0	0	0	0
Sürücü 3	0,5	0	0,2	1
Sürücü 4	0,1	0,2	0	0
Sürücü 5	0	0	0,1	0
Sürücü 6	0,2	0,1	0,2	1
Sürücü 7	0	0	0,6	1
Sürücü 8	0,2	0,2	0,5	0
Sürücü 9	0	0	0	0
Sürücü 10	0	0	0,9	1
Sürücü 11	0,4	0	0,6	1
Sürücü 12	0	0	0,1	0
Sürücü 13	0	0,1	0	0
Sürücü 14	0,1	0	0	0
Sürücü 15	0	0	0	1
Sürücü 16	0	0	0,1	0,5
Sürücü 17	0,5	0,5	0,4	0
Sürücü 18	0	0	0	0
Sürücü 19	0	0	0	0,2
Sürücü 20	0,3	0	0,7	1
Sürücü 21	0	0	0	0
Sürücü 22	0	0	0	0
Sürücü 23	0,1	0,1	0,3	1
Sürücü 24	0,1	0	0	0
Sürücü 25	0	0	0	1
Sürücü 26	0,2	0	0,1	1
Sürücü 27	0,8	0	0,8	0
Sürücü 28	0	0,5	0,3	0
Sürücü 29	0	0	0,2	1
Sürücü 30	0,4	0	0	1

EK – 2
Hız Limiti ve İvmelenme Tespiti Kodları

```
import os
from gps import *
from time import *
import time
import threading
import math
import sqlite3 as lite
import datetime




global max_latitude
global max_longitude
global min_latitude
global min_longitude




max_latitude = 38.69805556
max_longitude = 39.27083333
min_latitude = 38.63000000
min_longitude = 39.12611111




os.system('sudo kilall gpsd')
os.system('stty -F /dev/ttyS0 9600')
os.system('sudo gpsd /dev/ttyS0 -F /var/run/gpsd.sock')
gpsd = None #seting the global variable

os.system('clear') #clear the terminal (optional)

class GpsPoller(threading.Thread):


  def __init__(self):
    threading.Thread.__init__(self)
    global gpsd #bring it in scope
    gpsd = gps(mode=WATCH_ENABLE) #starting the stream of info
```

```
    self.current_value = None
    self.running = True #setting the thread running to true

  def run(self):
    global gpsd
    while gpsp.running:
      gpsd.next() #this will continue to loop and grab EACH set of
gpsd info to clear the buffer




if __name__ == '__main__':
  gpsp = GpsPoller() # create the thread
  con = None
  try:
    con = lite.connect('logs.db')



  except lite.Error, e:


    print "Error {}:".format(e.args[0])
    sys.exit(1)


  try:
    gpsp.start() # start it up
    last_speed=gpsd.fix.speed
    while True:
      #It may take a second or two to get good data
      #print gpsd.fix.latitude,', ',gpsd.fix.longitude,'  Time:
',gpsd.utc

      os.system('clear')


      speed = gpsd.fix.speed*3.6
      print speed
      if (gpsd.fix.latitude > min_latitude and gpsd.fix.latitude <
max_latitude) and (gpsd.fix.longitude > min_longitude and
gpsd.fix.longitude < max_longitude):
```

```
          if (speed > 0) and (speed<80) :
              now = datetime.datetime.now()
              con.execute('INSERT INTO gps(konum, asim, tarih,
ceza) values("'+str(gpsd.fix.latitude)+',
'+str(gpsd.fix.longitude)+'", "'+str(speed)+'", "'+str(now)+'",
10);');
              con.commit()
          elif (speed > 80) and (speed<90) :
              now = datetime.datetime.now()
              con.execute('INSERT INTO gps(konum, asim, tarih,
ceza) values("'+str(gpsd.fix.latitude)+',
'+str(gpsd.fix.longitude)+'", "'+str(speed)+'", "'+str(now)+'",
20);');
              con.commit()
          elif speed > 90:
              now = datetime.datetime.now()
              con.execute('INSERT INTO gps(konum, asim, tarih,
ceza) values("'+str(gpsd.fix.latitude)+',
'+str(gpsd.fix.longitude)+'", "'+str(speed)+'", "'+str(now)+'",
40);');
              con.commit()
      else:
          if speed > 120:
              now = datetime.datetime.now()
              con.execute('INSERT INTO gps(konum, asim, tarih,
ceza) values("'+str(gpsd.fix.latitude)+',
'+str(gpsd.fix.longitude)+'", "'+str(speed)+'", "'+str(now)+'",
15);');
              con.commit()

      ivme = (gpsd.fix.speed - last_speed)/3
      print ivme
      if ivme > 4.604:
         now = datetime.datetime.now()
         con.execute('INSERT INTO gps_ivme(konum, asim, tarih,
ceza) values("'+str(gpsd.fix.latitude)+',
'+str(gpsd.fix.longitude)+'", "'+str(ivme)+'", "'+str(now)+'",
50);');
         con.commit()
      elif ivme < -4.604:
          now = datetime.datetime.now()
          con.execute('INSERT INTO gps_ivme(konum, asim, tarih,
ceza) values("'+str(gpsd.fix.latitude)+',
'+str(gpsd.fix.longitude)+'", "'+str(ivme)+'", "'+str(now)+'",
50);');
```

```
            con.commit()

        last_speed=gpsd.fix.speed
        time.sleep(3)




except (KeyboardInterrupt, SystemExit): #when you press ctrl+c
  print "\nKilling Thread..."
  con.close()
  gpsp.running = False
  gpsp.join() # wait for the thread to finish what it's doing
print "Done.\nExiting."
```

EK – 3
Trafik Işığı Tespiti Kodları

```
importcv2
            import numpy as np
            import time
            import math
            import sqlite3 as lite
            import datetime




            kamera=cv2.VideoCapture(0)
            con = None


            con = lite.connect('logs.db')
            lower_red1 = np.array([0,100,100])
            upper_red1 = np.array([10,255,255])
            lower_red2 = np.array([160,100,100])
            upper_red2 = np.array([180,255,255])
            lower_green = np.array([40,50,50])
            upper_green = np.array([90,255,255])
            # lower_yellow = np.array([15,100,100])
            # upper_yellow = np.array([35,255,255])
            lower_yellow = np.array([15,150,150])
            upper_yellow = np.array([35,255,255])




            while(1):



                font = cv2.FONT_HERSHEY_SIMPLEX
                _, img_buyuk = kamera.read()
                img=cv2.resize(img_buyuk,None,fx=0.5,fy=0.5)
                cimg = img
                hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)


                # color range
```

```
    mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
    mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
    maskg = cv2.inRange(hsv, lower_green, upper_green)
    masky = cv2.inRange(hsv, lower_yellow, upper_yellow)
    maskr = cv2.add(mask1, mask2)


    size = img.shape
    # print size


    # hough circle detect
    r_circles = cv2.HoughCircles(maskr, cv2.HOUGH_GRADIENT, 1, 80,
                                 param1=50, param2=10, minRadius=0,
maxRadius=30)


    g_circles = cv2.HoughCircles(maskg, cv2.HOUGH_GRADIENT, 1, 60,
                                  param1=50, param2=10, minRadius=0,
maxRadius=30)


    y_circles = cv2.HoughCircles(masky, cv2.HOUGH_GRADIENT, 1, 30,
                                  param1=50, param2=5, minRadius=0,
maxRadius=30)


    # traffic light detect
    r = 5
    bound = 4.0 / 10
    if r_circles is not None:
        r_circles = np.uint16(np.around(r_circles))


        for i in r_circles[0, :]:
            if i[0] > size[1] or i[1] > size[0]or i[1] > size[0]*bound:
                continue


            h, s = 0.0, 0.0
            for m in range(-r, r):
                for n in range(-r, r):
```

```
                    if (i[1]+m) >= size[0] or (i[0]+n) >= size[1]:
                        continue
                    h += maskr[i[1]+m, i[0]+n]
                    s += 1
            if h / s > 50:
                cv2.circle(cimg, (i[0], i[1]), i[2]+10, (0, 255, 0), 2)
                cv2.circle(maskr, (i[0], i[1]), i[2]+30, (255, 255, 255),
2)
                cv2.putText(cimg,'RED',(i[0], i[1]), font,
1,(255,0,0),2,cv2.LINE_AA)


        while(g_circles is None):
                _, img_buyuk = kamera.read()
                img=cv2.resize(img_buyuk,None,fx=0.5,fy=0.5)
                cimg = img
                hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
                maskg = cv2.inRange(hsv, lower_green, upper_green)
                mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
                mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
                maskr = cv2.add(mask1, mask2)
                r_circles = cv2.HoughCircles(maskr, cv2.HOUGH_GRADIENT,
1, 80,
                                param1=50, param2=10, minRadius=0,
maxRadius=30)
                if(r_circles is None):
                    now = time.time()
                    future = now + 3
                    while(time.time()<future):
                        _, img_buyuk = kamera.read()
                        img=cv2.resize(img_buyuk,None,fx=0.5,fy=0.5)
                        cimg = img
                        hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
                        mask1 = cv2.inRange(hsv, lower_red1, upper_red1)
                        mask2 = cv2.inRange(hsv, lower_red2, upper_red2)
                        maskr = cv2.add(mask1, mask2)
                        r_circles = cv2.HoughCircles(maskr,
cv2.HOUGH_GRADIENT, 1, 80,
                                            param1=50, param2=10, minRadius=0,
maxRadius=30)
                        maskg = cv2.inRange(hsv, lower_green,
upper_green)
                        g_circles = cv2.HoughCircles(maskg,
cv2.HOUGH_GRADIENT, 1, 60,
```

```
                                param1=50, param2=10, minRadius=0,
maxRadius=30)
                        if(r_circles is not None or g_circles is not
None):
                            break

                    if(r_circles is None and g_circles is None):
                        #now = datetime.datetime.now()
                        #con.execute('INSERT INTO traffic_light(ceza,
tarih) values("50","'+str(now)+'");');
                        #con.commit()
                        print("ceza")
                        r_circles=None
                        g_circles=None
                        break




    if g_circles is not None:
        g_circles = np.uint16(np.around(g_circles))


        for i in g_circles[0, :]:
            if i[0] > size[1] or i[1] > size[0] or i[1] > size[0]*bound:
                continue


            h, s = 0.0, 0.0
            for m in range(-r, r):
                for n in range(-r, r):


                    if (i[1]+m) >= size[0] or (i[0]+n) >= size[1]:
                        continue
                    h += maskg[i[1]+m, i[0]+n]
                    s += 1
            if h / s > 100:
                cv2.circle(cimg, (i[0], i[1]), i[2]+10, (0, 255, 0), 2)
                cv2.circle(maskg, (i[0], i[1]), i[2]+30, (255, 255, 255),
2)
                cv2.putText(cimg,'GREEN',(i[0], i[1]), font,
1,(255,0,0),2,cv2.LINE_AA)
```

```
    if y_circles is not None:
        y_circles = np.uint16(np.around(y_circles))


        for i in y_circles[0, :]:
            if i[0] > size[1] or i[1] > size[0] or i[1] > size[0]*bound:
                continue


            h, s = 0.0, 0.0
            for m in range(-r, r):
                for n in range(-r, r):


                    if (i[1]+m) >= size[0] or (i[0]+n) >= size[1]:
                        continue
                    h += masky[i[1]+m, i[0]+n]
                    s += 1
            if h / s > 50:
                cv2.circle(cimg, (i[0], i[1]), i[2]+10, (0, 255, 0), 2)
                cv2.circle(masky, (i[0], i[1]), i[2]+30, (255, 255, 255),
2)
                cv2.putText(cimg,'YELLOW',(i[0], i[1]), font,
1,(255,0,0),2,cv2.LINE_AA)


    cv2.imshow('detected results', cimg)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
kamera.release()
cv2.destroyAllWindows()
```

EK – 4
Emniyet Kemeri Kontrol Kodları

```
importRPi.GPIO
as GPIO
                import time
                import math
                import sqlite3 as lite
                import datetime


                GPIO.setmode(GPIO.BCM)


                GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP)


                con = None
                try:
                    con = lite.connect('logs.db')


                except lite.Error, e:
                    print "Error {}:".format(e.args[0])
                    sys.exit(1)
                while True:
                    input_state = GPIO.input(4)
                    if input_state == True:
                        now = datetime.datetime.now()
                        con.execute('INSERT INTO emniyet(ceza, tarih)
                values("5","'+str(now)+'");');
                        con.commit()
                        time.sleep(10)
                    else:
                        print "takili"
```

EK – 5
Çarpışma Kontrolü Kodları

```
İmportRPi.GPIO
as GPIO
                import time
                import sqlite3 as lite
                import datetime
                GPIO.setmode(GPIO.BCM)


                TRIG=23
                ECHO=24
                con = None


                con = lite.connect('logs.db')









                GPIO.setup(TRIG, GPIO.OUT)
                GPIO.setup(ECHO, GPIO.IN)


                GPIO.output(TRIG, False)
                time.sleep(2)
                while(True):
                    GPIO.output(TRIG,True)
                    time.sleep(0.00001)
                    GPIO.output(TRIG, False)


                    while GPIO.input(ECHO)==0:
                        pulse_start=time.time()


                    while GPIO.input(ECHO)==1:
                        pulse_end=time.time()
```

```
    pulse_duration=pulse_end-pulse_start


    distance=pulse_duration*17150


    distance=round(distance,2)


    if(distance>400):
        now = datetime.datetime.now()
        con.execute('INSERT INTO carpisma(ceza, tarih)
values("300","'+str(now)+'");');
        con.commit()

    print ("uzaklık: ",distance,"cm")
    time.sleep(3)


GPIO.cleanup()
```