# KARADENİZ TEKNİK ÜNİVERSİTESİ OF TEKNOLOJİ FAKÜLTESİ

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**



**MOBESE**

**BİTİRME PROJESİ**

# Emre DOĞANGÜN

**2020 -2021 BAHAR DÖNEMİ**

**KARADENİZ TEKNİK ÜNİVERSİTESİ**

**OF TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Mobese**

**BİTİRME PROJESİ**

# Emre DOĞANGÜN

**Bu projenin teslim edilmesi ve sunulması tarafımca uygundur.**

**Danışman Dr.Öğr.Üyesi EYÜP GEDİKLİ** **………..**

# 2019-2020 GÜZ DÖNEMİ

**IEEE Etik Kuralları**

**IEEE Code of Ethics**



Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990’da onaylanmıştır.

**ÖNSÖZ**

Bu projede trafik ile ilgilenen yetkililerin işini kolaylaştırmak amaçlanmaktadır. Yüzeysel olarak anlatılması gerekirse öncelik trafikteki araçları algılamaktır. Trafikteki araçların plakalarını okumak, trafik yoğunluğunu tespit etmek ve araçların hızını ölçmektir. Bu proje trafikteki araçların ve yetkililerin daha güvenli bir şekilde trafikte yer almalarını ağlamaktadır. Trafikteki araçların hız kurallarına uymadığı durumlarda yetkilileri bilgilendiren ve hangi araçların hız sınırını geçtiğini tespit eden program tasarlanmaktadır. Trafik yoğunluğundan sürücülerin bilgi sahibi olması hedeflenmektedir.

Tez çalışmam boyunca bana yardımlarını esirgemeyen değerli danışman hocam sayın **Dr.Öğr.Üyesi EYÜP GEDİKLİ**’e ve diğer tüm bölüm hocalarıma teşekkür ve saygılarımı sunarım.

Tüm öğrenim hayatım boyunca bana maddi manevi destekleriyle beni yalnız

bırakmayan aileme de sonsuz teşekkür ederim.

Emre Doğangün

Trabzon, 2020

**İÇİNDEKİLER**

|  |  |
| --- | --- |
|  | Sayfa No |
| IEEE ETİK KURALLARI | I |
| ÖNSÖZ | II |
| İÇİNDEKİLER | III |
| ÖZET | IV |
| Şekiller Dizini | V |
| 1.Genel Bilgiler | 1 |
| 1.1. Proje Tanıtımı | 1 |
| * + 1. Görüntü İşleme Nedir | 1 |
| * + 1. Görüntü İşleme Amaçları | 1 |
| 1.1.3. Neden Görüntü İşleme | 2 |
| 1.2 Çalışmanın Amacı | 3 |
| 1.3 Araştırma Sorusu | 3 |
| 2. Projede Kullanılan Yöntemler Metotlar | 3 |
| 2.1. Planlama | 3 |
| 2.2. Analiz | 3 |
| 2.2.1. Video Okuma İşlemi | 6 |
| 2.2.2. Gri Formata Çevirme | 6 |
| 2.2.3. Arka Plan Ayrımı ve İkili Seviyeye Dönüşüm İşlemi | 7 |
| 2.2.4. Kullanım Senaryoları | 12 |
| 2.2.5 Sınıf Diyagramı | 13 |
| 2.2.6 Akış Şeması | 13 |
| 2.2.7 Yazılım Geliştirme Araçları | 14 |
| 2.2.7.1 OpenCV | 14 |
| 2.2.7.1.1 Core Modülü | 16 |
| 2.2.7.1.2 İmgproc Modülü | 16 |
| 2.2.7.1.3 Video Modülü | 17 |
| 2.2.7.1.4 Calib3d Modülü | 17 |
| 2.2.7.1.5 Features2d Modülü | 17 |
| 2.2.7.1.6 Objdetect Modülü | 17 |
| 2.2.7.1.7 Highgui Modülü | 17 |
| 2.2.7.1.8 Gpu Modülü | 18 |
| 2.2.7.2 Python | 18 |
| 2.3 Tasarım | 19 |
| 2.4 Üretim | 19 |
| 2.5 Test | 19 |
| 2.6 Bakım | 19 |
| 3. Proje iş zaman planı | 19 |
| 4. Sonuçlar | 21 |
| 5. Proje Kurulumu | 34 |
| 6.Proje Linki | 35 |
| Kaynakça | 22 |
| STANDARTLAR ve KISITLAR FORMU | 23 |
| Özgeçmiş | 25 |

**ÖZET**

Bu tez çalışmasın da radar ve mobese sistemlerini araştırarak başlandı. Projenin amacı trafikteki araçları algılamak öncelikli olup araç plakası okumak, araç hızı ölçmek, trafikteki araçları sayarak trafik yoğunluğunu bulmaktır. Yapılacak olan bu çalışmanın özeti: Programa gönderilen videodaki araçlara ait bilgiler çıkartmak ve araçların plakasını okuyup hızlarını tespit etmek. Trafikteki yoğunluğu ölçmek ve trafikteki olağanüstü durumları tespit etmektir.

Program örnek tek bir video için çalışma şekli aşağıdaki gibi olabilir: Burada uygulamanın yaptığı adımlar şöyledir: Uygulama çalışır, parametre olarak gelen videoyu alır, uygulamaya verilen video işlenir, video üzerindeki araçları tespit eder, tespit edilen araçların hızı ölçüler, hız sınırını geçenler tespit edilir, video üzerindeki araç sayısı tespit edilir, buna bağlı olarak trafik yoğunluğu tespit edilir, program sonlanır. Günümüzde trafikteki araçların izlenmesi, trafik yoğunluğunun belirlenmesi ve bunlarla ilgili analizlerin yapılması kazaların azaltılması açısından önem arzetmektedir. Yol güzergahında araç hız tesbiti ve yoldaki araç yoğunluğu çeşitli yöntemlerle yapılmaktadır. Bu amaçla kameraların kullanılması yapılan işlemleri kolaylaştırmaktadır. Bu çalışmada kameralardan alınan video görüntüleri üzerinde, seyir halindeki araçların hızlarının tespit edilmesi gerçekleştirilmiştir. Ayrıca araçların seyir ettiği yol üzerinde, araçlar sayılarak yoldaki araç yoğunluğu tesbit edilmiştir. Böylece yoldaki trafik durumunun gerçek zamanlı olarak izlenmesi ve trafik kazalarının azaltılmasında kameralı sistemlerin daha etkin kullanılması amaçlanmıştır. Kamera ile elde edilen hız ve araç yoğunluk değerleri analiz edilmiş, gerçek değerleri ile karşılaştırılmıştır.

## ŞEKİLLER DİZİNİ

### Sayfa No

Şekil 1. Arka plandan Ayrıştırma 5

Şekil 2. Araç Takip Algoritması Akış Şeması. 5

Şekil 3. OpenCV video okuma fonksiyonu. 6

Şekil 4. Gri formata çevrilmiş bir görüntü karesi. 6

Şekil 5. Elde edilen arka plan görüntüsü. 7

Şekil 6. Arka Plan Ayrımı için Çerçeve Fark Algoritması Akış Şeması. 7

Şekil 7. Plaka Algılama 9

Şekil 8. Araç sayım uygulama ekranı 11

Şekil 9. Use Case Diyagramı 12

Şekil 9. Sınıf Diyagramları 13

Şekil 10. Akış şeması 14

Şekil 11. OpenCV yapısı ve içeriği 15

Şekil 12. Python dilini öne çıkaran yönler 18

Şekil 13. Yapılan iş paket tablosu 20

Şekil 14. Yapılacak olan iş paket tablosu 20

## 1. GENEL BİLGİLER

### 1.1. Proje Tanımı

Bu projede, genelde mobese sistemleri ile suçun önlenmesi ve suç sonrasında kanıtların ve şüphelilerin tespiti, çalıntı araç bulunması, trafik kazası tespiti ve tüm şüpheli durumların kolayca tespiti mümkün olmaktadır. Mobese kelimesi ; Mobil Elektronik Sistem Entegrasyonu kelimelerinin baş harflerinin kullanımıyla oluşturulmuş bir kelimedir. Güvenlik kameralarıyla oluşturulmuş bir kent izleme, denetim ve kontrol teknolojisini anlatır. Yapılacak olan projede araç tanıma işlemleri OpenCV ile gerçekleşmektedir. Yapılacak olan projenin büyük kısmı python programlama dili ile kodlanmaktadır.

* + 1. **Görüntü İşleme Nedir?**

Görüntü işleme bir fotoğrafın dijital formata geçirmek ve bu formatta bazı işlemler yapmaktır. Asıl amaç işlenmiş görüntüyü almak veya işlenmiş görüntüden yararlı, istenilen bilgileri almaktır. Bu aslen bir sinyal dağıtımıdır. Bu sinyal dağıtımındaki girdi bir fotoğraf, videodaki bir an (frame), görüntü ya da çıktısı görüntü ya da fotoğrafa benzeyen çıktıların hepsi bu dağıtımda girdi olabilir. Genellikle Görüntü işleme sistemi girdilere iki boyutlu sinyal olarak davranır.

Görüntü işleme genel olarak 3 adımdan oluşur.

1.Girdi(resim, fotoğraf vb. ) optik okuyucular ya da kameralar ile alınır.

2.Alınan veriyi analiz etme sıkıştırma oynama yapılması ve insan gözü ile görülemeyecek düzenleri yakalamak.

3.Çıktı son bölümdür. Değişen görüntü ya da görüntü analizine dayalı bir rapordur.

### 1.1.2 Görüntü İşlemenin Amaçları

Günümüz teknolojilerinde oldukça sık kullanılan görüntü işleme yöntem ve teknolojileri vardır. Görüntü işleme yapmanın temel amaçları aşağıdaki gibidir.

Bunlar:

1. Görüntüleme - Görünmeyen nesneleri gösterme.
2. Görüntü keskinleştirme ve restorasyon - Daha iyi bir görüntü oluşturma.
3. Görüntü geri kazanma - Bozulmuş görüntü eski haline getirme.
4. Düzenin bulunması - Görüntülere gizlenmiş düzenlerin bulunması.
5. Görüntü tanıma - Görüntülerdeki nesneleri tanıma.

“Günümüzde kameralı donanımların ucuzlaması ve yaygınlaşması ile birlikte insanoğlunun değerlendirmesi gereken devasa büyüklükte görüntü verileri her an yığılmaya devam ediyor. Her kavşakta rastlanan trafik kameralarından, binalardaki güvenlik kameralarından, profesyonel cihazlardan herkesin cebinde taşıdığı mobil telefonlara kadar pek çok kaynaktan gelen görüntüler işlenmeyi bekliyor. Cadde üzerine konumlandırılmış denetim kamerasını düşünün; yol durumunu sürekli kayıt altına alabilir. Ancak artık ihtiyaç duyduğumuz sadece devasa boyutlardaki bu görüntüyü saklamak değil. Gerçek ihtiyacımız trafik kurallarını ihlal eden araçların plakalarını da otonom olarak tespit edebilmek.

## 1.1.3 Görüntü İşlemenin Kullanıldığı Alanlar

Görüntü işleme yöntem ve teknikleri günümüzde bir çok alanda kullanılır. Görüntü işleme çok geniş bir konsepttir. Sağlık alanında insan hastalıklarını tespitine kadar kullanılmaktadır. Ayrıca,

* Robot teknolojileri
* Savunma sanayisi
* İnsansız hava araçları
* Montaj Hatları (Örneğin, otomobil üretim hattında çalışan otomasyon robotları)
* Snapchat, Instagram’da kullanılan gerçek zamanlı filtreler
* Yüz kilidi teknolojisi
* Google harita teknolojisi

ve daha bir çok alanda görüntü işleme teknolojisi mevcuttur.

#### 1.2. Çalışmanın Amacı

Bu projede, trafikte oluşabilecek kural dışı durumlarda araç sahiplerini ve yetkililere bilgi vermek içindir. Araç kullanıcılarına trafik ile ilgili bilgi vermek ve trafik yoğunluğunu düzenlemektir.

Günümüzde şirketlerde ki araç sahiplerini veya araç kullanıcılarını kolaylıkla bilgilendirmek. Yapılan bu uygulamayla python dilinde kodlanmış ve görüntü işleme esas alınarak openCV kütüphanesiyle videolarda ki araçları algılayıp, kuraldışı olayları bildirmek ve trafikteki güvenliği artırmaktır.

#### 2. PROJEDE KULLANILAN YÖNTEMLER VE METOTLAR

Proje fikrinin hayata geçirilmesi, yazılım yaşam döngüsü adımlarına göre geliştirilecektir. Planlama aşamasında araştırmalar yapılarak temel ihtiyaçlar belirlenmiştir ve fizibilite çalışmaları yapılmıştır. Bu çalışmaların sonucunda, Görüntü İşlenmesi ve egitilmesi tarafında OpenCv kütüphanesi kullanılacaktır. OpenCv nin bu alanda yapılmış hazır araç tanıma uygulamaları mevcuttur.

##### 2.1. PLANLAMA

Yazılım yaşam döngüsünün ilk aşamasıdır. Problemin tanımlanmasıyla başlayıp, literatür taraması yapılarak devam eder. Projenin temel ihtiyaçlarının belirlendiği, proje için fizibilite çalışmalarının yapıldığı aşamadır. Bu projede videoda bulunan araçlar taraması yapılarak gerekli işlemler sağlanacaktır. İlk aşama olarak programa video yüklenecektir, istenilen formata getirilip, video okunabilir hale getirilecektir. Yüklenen bu videoda araç algılaması yapılacaktır. Bu algılanan araçların plaka tespitleri yapılacaktır. Bu işlemler tamamlandıktan sonra sürücüler ve yetkililer için kolaylık sağlayan trafik yoğunluğu tespiti yapılacaktır, bu işlem araç tespit ve araç sayma işlemlerini kapsamaktadır.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | Bölümün Adı | **Yapılan İş** | **Süresi (Hafta)** |
| 1 | Analiz | Video Okuma işlemleri yapma (Kod kısmı) | 1 |
| 2 | Analiz | Veri seti hazırlama(OpenCV Haar Cascade) | 1 |
| 3 | Tasarım | Arayuz Yapımı | 1 |
| 4 | Tasarım | Tkinter kütüphanesi kullanımı | 1 |
| 5 | Tasarım | Butonlanlar ile fonksiyon bağlantıları kurma | 1 |
| 6 | Tasarım | Araç algılma işlemi yapımı | 1 |
| 7 | Tasarım | Tek yönlü Araç sayma | 1 |
| 8 | Tasarım | Çift yönlü Araç sayma | 1 |
| 9 | Tasarım | Araç plaka Okuma | 1 |
| 10 | Tasarım | Veri seti detaylandırma | 1 |
| 11 | Tasarım | Araç Hız tespit | 1 |
| 12 | Tasarım | Araç Hız tespit | 1 |
| 13 | Tasarım | Araç Hız tespit | 1 |
| 14 | Tasarım | Araç yoğunluğu bulma | 1 |
| 15 | Test | Test | 1 |
| 16 | Bakım | Bakım | 1 |

##### 2.2. ANALİZ

Analiz aşaması bir yazılım projesindeki en önemli aşamalardan biridir. Analiz aşaması projede bulunan tüm fonksiyonları detaylı bir biçimde belirlendiği aşamadır. Fonksiyonlar belirlendiği için gereksinimler de belirlenmiş olur. Genel olarak bütün projenin detayları bu aşamada belirlenmiş olur.

Yapılacak işlemlerden ve algoritmadan bahsedilmek gerekirse. Öncelikle videodaki arabaları bulunması gerekir. Araba bulmak için iki farklı yöntem uygulanır.

1. Haarcascade sınıflandırıcısı ile araç bulma
2. Arka plan çıkarma yöntemi ile araç bulma

**Haarcascade sınıflandırıcısı:**Pozitif ve negatif resimlerin bilgilerinden oluşan "xml" uzantılı bir belgedir. Pozitif resimler videoda aranan nesneleri, negatif resimler ise aranmayan nesneleri temsil eder.

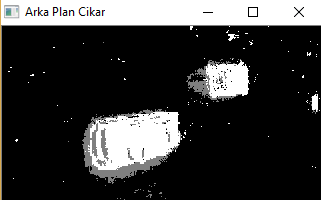
**Arka Plan Çıkarma:** Videoda değişen piksellerin yerini belirler ve videodan ayrıştırır.

Bu projede derleme hızı açısından daha hızlı olacağı ve cascade sınıflandırıcısı oluştururken yaşanacak vakit kaybı nedeniyle **arka plan çıkarma yöntemi** ile yapılacak.

Algoritma anlatılacak olunursa;

1. Video okunur.
2. Arka plan çıkartılır.
3. Kontürler bulunur.
4. Kontür alanı bulunur ve alanın koordinatları bulunur
5. Kontür alan parametreleri ile karşılaştırılır.(bu aşama diğer hareket eden nesneleri araç olarak saymaması için yapılır.)
6. Kontür koordinatları belirli bir koordinat değerinden geçerse sayaç bir arttırılır.

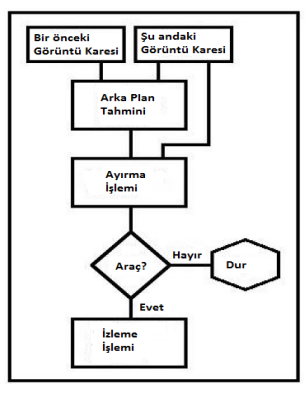
Arka plan çıkarma yöntemi uyguladıktan sonra kontürleri bulma nedenimizi ise şu şekilde açıklayabiliriz. Herhangi bir videoya arka plan çıkarma uyguladığınızda hareketli nesnenin beyaz geri kalan bölgelerin ise siyah olduğunu görürüz.

[](https://1.bp.blogspot.com/-temzC3r-8Jw/WIKbbKpQV0I/AAAAAAAAAhI/MuzoMi_KXFISJWGgwzKy2gW55ErZjxjcQCLcB/s1600/Ekran%2BG%25C3%25B6r%25C3%25BCnt%25C3%25BCs%25C3%25BC%2B%252838%2529.png)

Şekil 1. Arka plandan Ayrıştırma

Kontür işleminin tanımına baktığımız zaman aynı renk veya yoğunluğa sahip olan tüm kesintisiz noktaları (sınır boyunca) birleştiren bir eğri olarak basitçe açıklanabilir. yani videomuzda ki beyaz pikselleri hiyerarşik bir yapıya aktarır. Ve böylece araç alanını rahatça belirleyip belirli parametrelerle karşılaştırmamızı sağlar.

Bu temel ayrımın sebebi uygulamanın iki farklı alan arasında gerçekleşen bir iş akışına sahip olmasıdır. Araç takip algoritmasının GPU tarafından yürütülecek olan bölümün akış şeması Şekil 2’de gösterilmiştir.



Şekil 2. Araç Takip Algoritması Akış Şeması.

Kullanılan algoritmanın adımları ayrıntılı olarak aşağıda verilmiştir.

**2.2.1. Video Okuma İşlemi**

Değişen video formatları, bir görüntü işleme uygulaması için en önemli sorunlardan biridir. Bu uygulama içerisinde video okuma fonksiyonu CPU üzerinde çalışmaktadır. OpenCV, HighGUI sınıfı altında sağladığı yapılar ile bu işlemi en basit şekilde gerçekleştirmektedir. Şekil 2’de dosyadan görüntü okuması yapan kod görülmektedir



Şekil 3. OpenCV video okuma fonksiyonu.

Bu fonksiyon ile videonun ve videodaki her bir görüntü karesinin özellikleri hakkında bilgi sahibi olunabilmektedir.

**2.2.2. Gri Formata Çevirme**

Video içerisindeki her bir görüntü karesi RGB (kırmızıyeşil-mavi) formatında elde edilmiştir. Bu görüntü karelerinin işlenebilmesi için ilk olarak gri formata çevrilmesi gerekmektedir. Bu çevirim işlemi için aşağıdaki denklem kullanılmıştır. Gri\_Format = Kırmızı\*0.3 + Yeşil\*0.6 + Mavi\*0.1 (1) Bütün piksel değerleri rgb\_to\_kernel isimli çekirdek ile aynı anda gri formata çevrilmiştir. Şekil 4’de gri formata çevrilmiş bir görüntü karesi örneği gösterilmiştir.



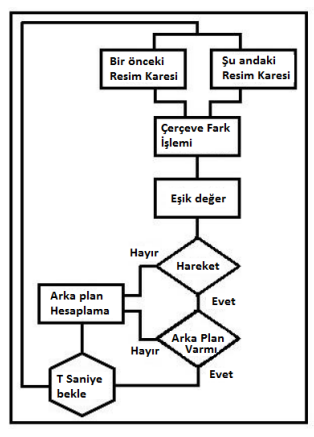
Şekil 4. Gri formata çevrilmiş bir görüntü karesi.

**2.2.3. Arka Plan Ayrımı ve İkili Seviyeye Dönüşüm İşlemi**

Arka plan ayrımı işlemi araç takip uygulamasının başarı oranını etkileyen önemli bir adımdır. Takip algoritmasının başarısı arka plan ayrımı işlemine bağlıdır. Arka plan ayrımı işlemi ne kadar başarılı gerçekleştirilirse ön plandaki araçların görüntüleri de o kadar net elde edilir. Literatürde arka plan ayrımı işlemi için çeşitli algoritmalar geliştirilmiştir. Bu çalışmada arka plan ayrımı işlemi ön plandaki araçları algılama işlemine dayanmaktadır. Bu işlem için çerçeve fark algoritması kullanılmıştır. Elde edilen arka plan görüntüsü Şekil 5’de, çerçeve fark algoritmasının akış şeması Şekil 6’de gösterilmiştir.



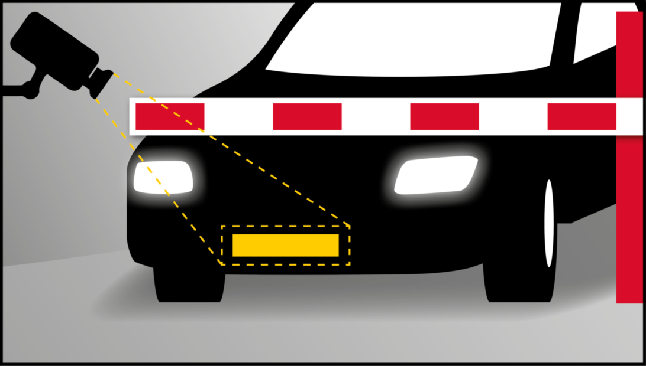
Şekil 5. Elde edilen arka plan görüntüsü.

****

Şekil 6. Arka Plan Ayrımı için Çerçeve Fark Algoritması Akış Şeması.

Bu algoritmada görüntü üzerinde kameradan ya da ışık değişiminden kaynaklanarak oluşan gürültü piksellerini azaltmak için başlangıçta ortalama filtre metodu kullanılmıştır. Her bir görüntü karesinin aynı pozisyonda bulunan piksel değerlerinin ortalama değeri hesaplanmıştır. Bu ortalama değer arka plan resminin o pozisyondaki piksel değerini temsil etmektedir. Daha sonra her bir görüntü karesinden oluşturulan arka plan resminin mutlak farkı alınmıştır. Fark değerinden oluşan yeni görüntü karesinde sıfırdan farklı değerler hareketin olduğu yerleri temsil etmektedir. Bazı durumlarda, hareket olmadığı halde kamera ya da ışık değişimi hareket var olgusu oluşturabilmektedir. Bu durumu önlemek için bir eşik değer belirlenmiştir. Görüntü kareleri ve arka plan resmi arasında mutlak fark alma işlemi yapılırken, fark değeri belirlenen eşik değerinden yüksek ise o piksel değeri ön plan olarak kabul edilmiş ve 1 değeri atanmıştır; eşik değerinden küçük ise arka plan olarak kabul edilmiş ve 0 değeri atanmıştır. Böylece her bir görüntü karesi ikili seviye dönüştürülmüştür. Sonraki iş plaka okumadır. Günümüzde araçların sayısının hızlıca artması ve bu araçların sadece insan gücüyle kontrol edilmesi gittikçe zorlaşmıştır. Araçların kolayca kontrol edilmesi için çeşitli plaka tanıma teknikleri geliştirilmiş ve plaka tanıma sistemleri aracın olduğu her yer de kullanılmaktadır.

Plaka tanıma sistemi üç aşamadan oluşmaktadır. Birinci aşama araç üzerinde bulunan plakanın  ayrıştırılma işlemidir. İkinci aşama ise koparılan plaka üzerinde karakterlerin ayrıştırılma işlemidir. Karakterleri ayrıştırmak için karakterlerin başlangıç ve bitiş noktaları kontrol edilir. Plaka üzerinde gerekli olan morfolojik işlemler uygulanarak bu işlem gerekleştirilir. Üçüncü aşamada ise yapay sinir ağları kullanılarak daha önceden öğretilmiş olan plakalarla karşılaştırmaktır.



Şekil 7. Plaka Algılama

Plaka tanıma adımları: Plaka tanıma işlemi yapmak için yedi  temel algoritma vardır. Bu algoritmalar ile plaka bulma işlemi yapılmaktadır. Bu algoritmaların hassasiyeti  plaka bulma işleminde çok fazla etkilemektedir.  Plaka tanıma işlemlerinde kullanılan algoritmalar şunlardır.

1. Plaka yerinin tespiti
2. Plakanın sonraki algoritmalara uygun bir biçimde yeniden konumlandırılması ve boyutlandırılması
3. Zıtlık ve parlaklık gibi görüntü özelliklerinin normalizasyonu
4. Karakter ayırma ile görüntü üzerinden karakterlerin çıkarılması
5. Optik karakter tanıma
6. Ülkeye özgü kelime dizimi ve kontrol işlemi
7. Daha güvenilir bir sonuç elde etmek için tanımlanan değerin birden fazla görüntüde ortalamasının alınması.

 Görüntü Üzerinden Plakanın Ayrıştırılması

Görüntü üzerindeki plaka kısmının doğru bir şekilde ayrılması oldukça önemlidir. Bu adımda yapılacak bir yanlış sistemin tamamını etkilemektedir. Plakanın görüntü üzerinden ayrılmasında  belirli koşullar vardır. Sistem tasarlanırken hava koşulları, parlaklık gibi kriterler önemlidir. Tüm şartlarda üst seviyede çalışabilecek sistem tasarlanması için burada yapılacak işlemler önemlidir.

Sistem günün her saatinde çalışması gerektiği için alınan görüntüler farklı kontrast değerlerine sahip olacaklardır. Işığın az olduğu veya ışığın çok olduğu zamanlar da olacaktır. Bu dengesizlikler alınan görüntüler üzerinde bozulmalara neden olmaktadır. Alınan görüntü üzerinde filtre işlemleri yapmadan direk olarak plaka bölgesi bulunmaya çalışıldığında doğru bir sonuç elde edilmeyecektir. Doğru bir sonuç elde etmek için gerekli olan işlemlerin yapılması gerekmektedir. Bu işlemler birbirini takip eden işlemlerden oluşmaktadır. İşlemler sırasıyla ilk olarak alınan görüntünün griye çevrilir.

Griye çevrilen resim üzerinde gürültü temizleme işlemi yapılır. Görüntü üzerinde histogram eşitleme işleme işlemi yapılır. Morfolojik işlem uygulanır. Histogram eşitleme yapılan görüntüden, morfolojik işlem  yapılmış işlem piksel piksel çıkartılır. Görüntü eşikleme işlemi yapılır. Canny edge uygulanır. Genişletme işlemi yapılır. Kontur işlemi gerçekleştirilir. Son olarak plaka bölgesi maskeleme yapılarak görüntü üzerinden ayrıştırılır. Sonra araç Hız tespiti yapılır.Trafikteki araçların belirlenmesinde izlenen yol hareket eden nesnenin takip edilmesiyle başlar. Hareket edilen nesnenin takibinin sağlanması ardından kare değişimi üzerinden geliştirilen yazılımla hız verisinin alınması sağlanmıştır. 1/30 Fps değerine sahip kamera ile yapılan ölçümlerde yaklaşık 200 m’lik alan içersinde ölçüm işlemi yapılmıştır. Burada toplam yer değiştirme hesaplaması üzerinden hız verisi hesaplanmıştır. Ortalama hız bilgisi toplam yer değiştirmenin toplam zamana bölünmesi ile Eşitlik 2'de verildiği gibi hesaplanmıştır. Araç hız verisinin elde edilmesinde aşağıdaki yol izlenmektedir. Öncelikle araç hız ölçümü yapılacak olan yön belirlenmelidir. Yön ve hatta ölçüm yapılacak şeridin tanımlanması ölçüm sonuçlarının doğruluğunu arttıracaktır. Çok şeritli yollarda araçların hızlarının belirlenmesi için her şerite paralel olarak birer kamera uygulaması yapılmalıdır. Çoklu ölçüm ile aynı yada hızları yüksek olan araçların belirlenmesi daha kolay olacaktır. 1/30 değerinde bir fps’ye sahip kameradaki hareket algılama duyarlılığa göre karelerdeki değişimin hesaplanmasında 1.frame, 3.frame, 5.frame şeklinde bir lineer artış gösteren mantık üzerinden gidilmiştir. Fps değeri yüksek donanım aygıtlarıyla yapıldığında ise hareketin algılanması ve hız verisinin ölçülmesi çok daha hassas ve doğru olacağı düşünülmektedir. Sonraki işlem araç yoğunluğu tespitidir. Mevcut durumda trafik ekipleri araç sayımını araç hız tespiti sırasında matbu bir form aracılığı ile yazarak doldurmaktadırlar. Bu formlar ile elde edilen veriler ışığında yerel, bölgesel ve ulusal denetim raporları çıkarılmaktadır. Bu raporlar neticesinde araç yoğunluğu, hız limit aşımı, kaza riski, kaza oranı, günlere ve saat aralıklarına göre araç sayılarına bakılarak, trafik denetim sıklıkları ve konumları belirlenmektedir. Araç sayımı trafik ekipleri tarafından manuel ve Karayolları Genel Müdürlüğü tarafından ise, donanımların yol üzerine şerit şeklinde serilmesi suretiyle yapılmakta olduğu görülmüştür. Geliştirilen yazılım ile araç hızının tespit edilmesine paralel olarak, belirlenen ölçüm alanında araç sayım işlemi yapılmıştır. Hareketi algılanan ve kimliklenen aracın hızının ölçülmesi yerine araç sayımı işlemi yapılmıştır. Şekil 8’de araç sayım uygulama ekranı görülmektedir.

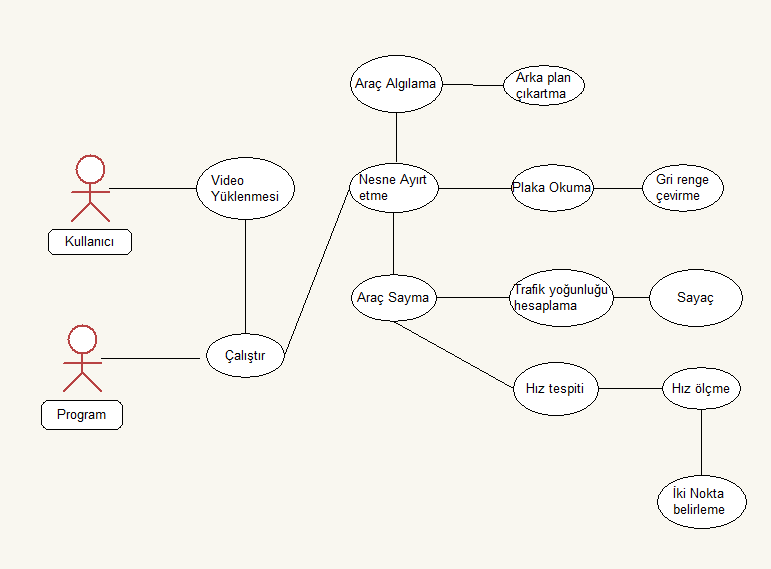


Şekil 8. Araç sayım uygulama ekranı

Projenin sistem gereksinimlerini belirlemek amacıyla kullanım senaryoları (use case) oluşturulur. Kullanım senaryoları genel olarak projede hangi aktörlerin bulunacağını, bu bulunan aktörlerin neler yapabileceğinin anlatır.

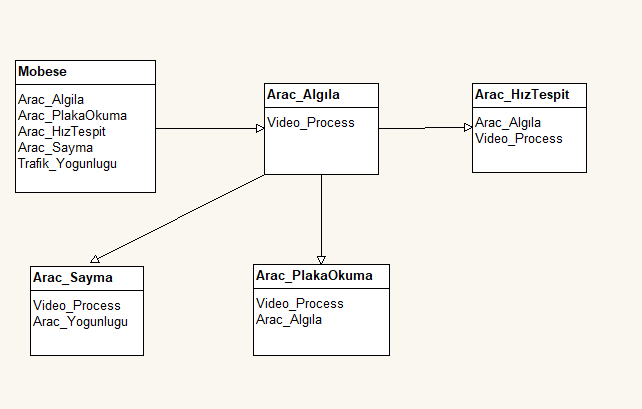
###### 2.2.4. Kullanım Senaryoları

İşletmelerin sistemi kullanabilmeleri için python 3.7 ve openCV kütüphanesi install edilmiş ve içinde kullanılacak olan kütüphanelerinde install edilmesi gerekmektedir. Kullanıcının yapması gereken işlem, işlenecek videoyu programda çalıştırmaktır. Bu sayede program çalışır hale gelir ve istediği bilgileri elde eder. Şekil 9. Use case diyagramında kullanıcının programı nasıl kullanacağı şemasal olarak belirtilmiştir.



Şekil 9. Use Case Diyagramı

2.2.5 Sınıf Diyagramları

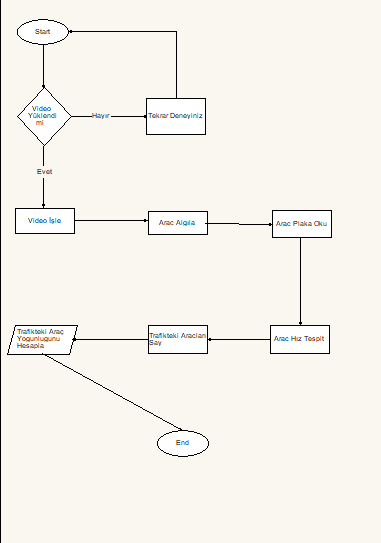


Şekil 10. Sınıf Diyagramları

Yukarıda gösterilen sınıf diyagramında öğrencilerin fotoğraflarının çekilmesi ve işlenmesine dair bir sınıf diyagramı çizilmiştir. Görüldüğü üzere nesne yönelimli programlamaya uygun ve SOLID tekniklerine uygun şekilde tasarlanmıştır.

2.2.6. Akış Şeması

Akış diyagramı, bir problemin çözümüne yönelik izlenecek yöntemlerin şekil ve sembollerle ifade edilmesidir. Akış şemasında her adım birbirinden farklı anlamlar taşıyan şekillerden oluşur ve adımlar arasındaki ilişki oklar ile tayin edilir. Kodlanacak programın akış şemasının oluşturulması, sürecin daha kolay çözümlenmesine yardımcı olur. Akış şemamız tasarım aşamasında kolaylık sağlamak amacıyla çizilmiştir(Şekil 11).



Şekil 11. Akış şeması

**2.2.7. Yazılım Geliştirme Araçları**

**2.2.7.1. OpenCV**

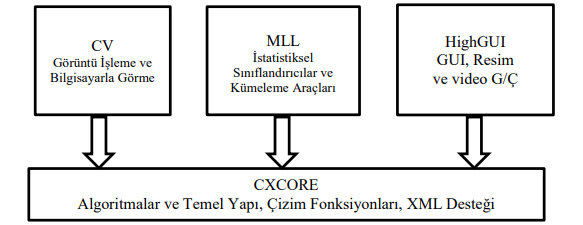
Intel firması tarafından ilk sürümü 1999 yılında çıkarılan OpenCV, bilgisayarlı görü ve gerçek zamanlı görüntü işleme uygulamalarının gerçekleştirildiği platformdur. OpenCV’nin İngilizce açılımı “Open Source Computer Vision Library” dir. “Açık Kaynak Kodlu Bilgisayarlı Görü” olarak Türkçe’de kullanılmaktadır.

OpenCV, Linux, Windows, PSP (PlayStation Portable), IOS, android ve birçok gömülü ve mobil sistem platformunda çalışabilen, C ve C++ dilleriyle yazılmış, görüntü işleme (image processing) uygulamaları ve gerçek zamanlı bilgisayarla görme (real time computer vision) için kullanılabilen, açık kaynak kodlu bir kütüphanedir [31]. Açık kaynak kodlu olması, ticari kullanımının ve eğitim amaçlı kullanılmasının ücretsiz olması diğer görüntü işleme araçlarından en büyük farklılığıdır.

OpenCV, birçok deneyimli yazılımcının çalıştığı Willow Garage şirketi tarafından desteklenmektedir. OpenCV görüntü işleme aracı birçok açık kaynak yazılımına ev sahipliği yapan SourceForge sitesinden indirilebilmektedir. İçerdiği fonksiyonların birçoğu platformdan bağımsız olarak çalışır. Mevcut olan C ara yüzüne ek olarak OpenCV 2.0 sürümünden itibaren, C++ ara yüzü de eklenmiştir.

Günümüzde, OpenCV içerisindeki görüntü işleme algoritmaları ve bilgisayarla görme kullanılarak;

* İnsan-Bilgisayar Etkileşimi
* Nesne Kimliklendirme, Nesne Bölümleme ve Nesne Tanıma
* Yüz Tanıma
* İşaret Dili Tanıma
* Hareket Yakalama, Hareket Algılama ve Hareket Takibi
* Çiftli ve Çoklu Kamera Kalibrasyonu ve Derinlik Hesaplama
* Hareketli Robot Teknolojileri uygulamaları geliştirilebilmektedir.



Şekil 12. OpenCV yapısı ve içeriği

OpenCV kütüphane yapısında beş temel bileşen mevcuttur. Bu bileşenlerden dördü Şekil 10.’da gösterilmektedir. CV bileşeni ismini Computer Vision (Bilgisayarla Görme) kelimesinin baş harflerinden almıştır. CV bileşeninin içeriğinde bilgisayarla görme algoritmaları ve görüntü işleme fonksiyonları vardır. Machine Learning Library kelimelerinin baş harflerini alan MLL bileşeni ise isminden de anlaşılacağı üzere, makina öğrenmesi için ihtiyaç duyulan istatistiksel verilere ulaşmak ve mevcut olan verileri sınıflandırmak amacıyla kullanılan araç ve fonksiyonlardan oluşur. Diğer bir bileşen olan HighGUI bileşeni ise OpenCV kütüphanesinde tanımlanmış olan pek çok nesneyi (slider, form) oluşturabilmemizi sağlayan bir grafik arabirimidir. Ayrıca bu bileşen resim ve videoları yüklemek, hafızaya kaydetmek, hafızadan veri silmek için gerekli giriş/çıkış (I/O) komutlarını da içerir. CXCore bileşeni ise, OpenCV’ deki Ip1Image, cvPoint, cvMat, cvSize, cvHistogram gibi veri yapılarına sahip XML desteği ne sahip bir kütüphanedir. Son olarak CvAux bileşeni, şekil eşleştirme (shape matching), yüz tanıma (face-recognation), objenin ana hatlarını bulma (finding skeletons), vücut hareketlerini tanıma (gesture recognition), ağız hareketlerini izleme (mouth-tracking) ve kamera kalibrasyonu gibi daha pek çok deneysel algoritmaları bünyesinde barındıran bir kütüphanedir.

Bilgisayarlı görme ve makine öğrenmenin bir biriyle olan sıkı ilişkisinden dolayı OpenCV, genel amaçlı bir MLL (Machine Learning Library) kütüphanesini içerir. Bu alt kütüphanede istatistiksel örüntü tanıma ve kümelenme (clustering) odaklanılmıştır. MLL daha çok görme eylemi için tasarlanmasına rağmen herhangi bir makine öğrenmesi probleminde de kullanılabilir. OpenCV modüler bir yapıya sahiptir. Yani paylaşılan veya statik kütüphaneleri içeren parçalardan oluşur. Güncel OpenCV modülleri kısaca özetlenecektir.

**2.2.7.1.1. Core Modülü**

Diğer modüllerin kullandığı temel veri yapıları, çok boyutlu diziler ve temel fonksiyonların tanımlandığı kompakt bir modüldür. OpenCV resim işleme ve bilgisayarlı görme için kolay kullanımlı veri yapıları içerir ve bu veri yapıları bu modülde tanımlanmıştır. Kural olmamakla birlikte temel olarak her OpenCV uygulamasına bu modül eklenilir. Çünkü OpenCV’nin gücü birazda bilgisayarlı görme için sunduğu basit veri yapılarıyla da ilişkilidir.

**2.2.7.1.2. İmgproc Modülü**

Görüntü işleme modülüdür. Bu modülde doğrusal ve doğrusal olmayan görüntü filtreleri, geometrik görüntü dönüşümleri (yeniden boyutlandırma, affine ve perspektif eğilme), renk uzayı dönüşümü, histogramlar v.b. gibi işlemler için fonksiyonların bulunduğu modüldür.

**2.2.7.1.3. Video Modülü**

Hareket tahmini, arka plan çıkarma ve nesne izleme algoritmalarını içeren bir video analiz modülüdür. Görüntü işleme ve bilgisayar görmesi alanlarında iyi bilinen problemlerde kullanılan bir modüldür.

**2.2.7.1.4. Calib3d Modülü**

Temel çoklu görünüm geometri algoritmaları, tek ve stereo kamera kalibrasyonu, nesne poz tahmini, stereo benzeşme algoritmaları ve 3D yeniden kurma unsurları içerir. Bilgisayarlı görmede yeni yeni araştırmacıların ilgisini çeken 3D yöntemler sunar.

**2.2.7.1.5. Features2d Modülü**

Giriş verisi için ayırt edici özellik algılayıcıları, veri tanımlayıcıları ve tanımlayıcı eşleştiricileri içeren bir modüldür. Örneğin giriş verisindeki köşelerin algılanarak bunun sisteme giriş olarak verilmesi için kullanılabilir.

**2.2.7.1.6. Objdetect Modülü**

Nesneleri ve önceden tanımlanmış sınıfların örneklerini (yüzler, gözler, kupalar, insanlar, araba v.b. gibi) algılamak için kullanılan modüldür. Bu modül kullanılarak yaygın bir çok algoritma için OpenCV ile gerçeklenmiş uygulamalar mevcuttur.

**2.2.7.1.7. Highgui Modülü**

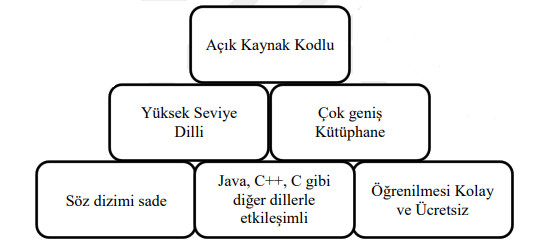
Kolay kullanım ara yüzüyle video çekimi, görüntü ve video kodekleri sağlamasında kullanılır. Ayrıca OpenCV platform bağımsız olduğu için kullanıcı ara yüzü tasarımı için de bu modül kullanılır. Bu şekilde programlama dili ve platform sorunlarına takılmadan ara yüz tasarlanabilir.

**2.2.7.1.8. Gpu Modülü**

Farklı OpenCV modüllerindeki GPU (Graphics Processing Unit) algoritmalarını hızlandıran bir modüldür. Temel olarak tüm modellerle içe içe veya onların efektifliğini arttırmayı hedefleyen bir modüldür. OpenCV kütüphanesinde bu modüller haricinde, Flann ve Google test sarmalayıcılar ve Python bağlamaları gibi bazı yardımcı modüller de bulunmaktadır.

2.2.7.2. Python

Guido Van Rossum tarafından 1991 yılında geliştirilen Python, çok güçlü ve yüksek seviyeli, dinamik nesne yönelimli programlama dilidir. Python platformdan bağımsız bir programlama dili olduğundan dolayı Windows, Linux, BSD, Mac OS X, Solaris, AIX, AS/400, AROS, MorphOS, BeOS, S60, iPhone, iPOD, Macintosh ve Android dahil tüm işletim sistemleri ve büyük donanım platformları üzerinde çalışabilmektedir. Ayrıca Python programlama dilinin temiz ve basit söz dizimi, onu Ruby ve Perl gibi alternatiflerin önüne geçirmiştir. Phyton programı Eric S. Raymond gibi birçok programcı ve Google tarafından tercih edilen bir dil haline gelmiştir. Python dilinin söz diziliminin basit olması sayesinde hem program yazmak, hem de başkası tarafından yazılmış bir programı okumak, başka dillere kıyasla daha kolaydır.



Şekil 13. Python dilini öne çıkaran yönler

Şekil 12.’te Python dilini öne çıkaran yönler gösterilmiştir. “Thinking in C++” ve “Thinking in Java” kitaplarının yazarı Bruce Eckel’e göre Python dili kadar hiçbir dil üretken değildir ve diğer programlama dillerinin programcıların işini kolaylaştırmaya yönelik bir amacı yoktur.

###### 2.3. TASARIM

Tasarım aşaması analiz aşamasından sonra gelir. Analiz aşamasında çıkarılmış olan dokümantasyonlar ile projenin genel tasarımı ortaya çıkarılır. Bu tasarım projeye sonradan dahil olabilecek yeni yazılımcılara referans olabilecek derecede açık ve kolay anlaşılabilir bir şekilde hazırlanması gerekmektedir.

# 2.3.1 Arayüz Tasarımı, Python

“Tk” birçok programlama dili tarafından GUI tasarlamak için kullanılan açık kaynaklı, multi-platform bir toolkittir. Tkinter da, Python için Tk’ye bir kapıdır, bir arayüzdür. Tkinter tabi nihayetinde bir, Python modülüdür. Fakat kaynak koduna bakıldığında görülür ki, aslında Tcl/Tk kütüphanelerini kullanan bir C uzantısıdır. Küçük masaüstü uygulamalarından, bilimsel modellemelere kadar birçok uygulama alanı var bu Tkinter’ın. Python ve GUI okumalarındaysanız, Tkinter nispeten işinizi halletmeniz için hızlı bir yol sunabiliyor. Buradaki amacım da zaten tüm bu Python’da arayüz tasarımı konseptinin üzerinde durmak olacak. Fakat Tkinter dışındaki modüllerden bahsetmiyeceğim. Tkinter yüklü standart python dağıtımlarınca yüklü gelir. Ve Java’da JavaFX falan kurcaladıysanız hiç, ona kıyasla birçok şey daha az kod parçasında dönebiliyor bu frameworkde. Bu makalede de, dilim döndüğünce örnekler ile bir şeyler aktarmaya çalıştırıldı ve tabi kronolojik bir düzende aşağıya doğru detaylandırıldı.

Başlamadan önce, yukarda dediğim gibi sisteminizde zaten python3 yüklüyse tkinter olması lazım. Python’u da kendi sitesinden indirebilirsiniz. command shell’de python3 yazıp aşağıdaki kod parçalarını yazın:

import tkinter  
tkinter.\_test()

“This is Tcl/Tk version 8.6” yazdığından emin olun; çünkü fark edecek.

from tkinter import \*

Python’da bu şekilde import etmeyi, “import all” ifadesi olduğundan memory’de sızıntılara ya da floodinglere yol açar..

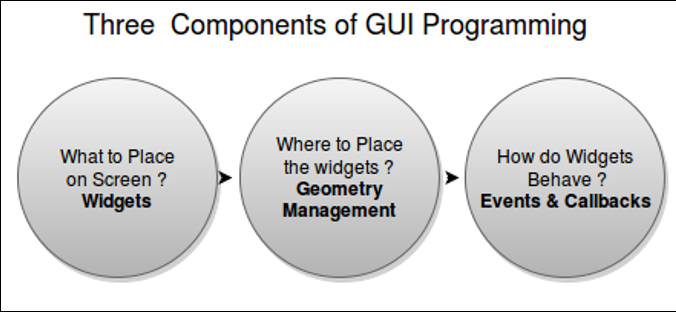
import tkinter

Bunu da yapınca “tkinter.bir\_method”

import tkinter as tk

Bu da birçok geliştirici tarafından uygulanan genel kullanım şeklidir zaten. Böyle de konumuzla hiç alakası olmayan kısa bir python inceliklerinden sonra bir şeylere bakabiliriz artık.

GUI tasarlamak isteyen biri olarak sizin kendinize sormanız gereken birkaç soru olmalı. “Ekranda hangi bileşenler gözükmeli?”. Ne bileyim işte, butonlar mı, entry fields mı, checkbox’lar mı? Bu komponentlere biz widgets diyoruz, Windows Gadgets’in kısaltması. “Bu bileşenler nereye gider”. Android Studio’da mesela Layout’lar falan vardır. Birçok grafiksel arayüz barındıran araçlarda, sözgelimi komponentlerin geometrik yerleşimi hep söz konusudur ve bir yaklaşım vardır. En önemli soru ise: “Bileşenler nasıl etkilenir ve davranır?”. Genelde kullanıcı olarak bu bileşenlerin davranışlarına çok alışkınızdır. Örneğin sağdaki kaydırma çubuğunu kaydırmanın neye sebep olacağını biliriz, ya da sağ üstteki çarpı işaretinin anlamını. İşte bu gibi fonksiyonlar bu görselliklere gömülü davranışlardır. Tabi bunlar en basiti. Sonuçta davranıştan söz ediyoruz ve verdiğim örnekler sadece farenin imlecini kullanarak tıklama yoluyla gerçekleşen şeylerdi. Ama bir davranış bundan daha fazlası olabiliyor, ne yaptığınızdan ziyade ne yapmadığınızda bir eylem oluşturabiliyor. Bu eylemlere, davranışlara “event” denir işte. “Event Handling” diye bir mevzuda bunların yönetimini kapsar. Aşağıdaki resim bahsettim 3 sorunun da, GUI’yı anlamak adına ne denli önemli olduğunu gösterecektir.



Şekil 14. GUI Programlama

Paletimiz genelde widget’dan ibaret olacak. Sonuçta bir program söz konusu olduğunda, GUI’ye sahip olmak zorunda değildir. Yapmayı hedeflediği daha komplex bir şeyler vardır ve istersiniz ki kullanıcı dostu da olsun, işte o zaman GUI de yapıverelime gelir olay. Oysa mesela, Kali Linux dağıtımında bulunan bir sürü tool size GUI sunmaz. Yapması gerekenleri terminalde yapar adamlar, göze hitap etmek değildir çünkü amaçları. Ama bilimsel alanlarda ya da bir ton yerde amaçlardan biri kullanıcı dostluğudur, anlaşılmaktır. Ve işte bahsettiğim palette de widgetlar var. Yani şuanda programın iddasını sunduğu o komplex şeyleri yaptığınızı varsayın ve şimdi de yemeğin sunumunu yapmaya çalışıyorsunuz o kadar. Tabağımız bu nokta “root window”. Yani palet benzetmesi için de “root window” tuvaldir. Anlatmamdan ziyade direk aşağıda kodları çalıştırın ne dediğimi anlayacaksınız:

from tkinter import \*  
root = Tk()  
root.mainloop()

“root” aslında benim isim seçimimdi. Genelde GUI konseptinde falan bu main pencere “root” diye alınır falan diye çok kasmadım ben de.”tkinter.Tk” methoduyla bir obje yaratmış olduk aslında. “root” objesinin bir methodunu da 3.satırda çağırdık, ve bu da bu root penceresini sürekli açık tutan şeydi. “mainloop” methodunu durduran şey mesela bu pencereyi çarpı işaretine basarak kapatmanız olur. Buraların çalışma prensibine methodların hiyerarşik düzenine, ilerledikçe daha iyi aşina olacağız hiç merak etmeyin. Bu kısım için hedeflediğim şey sadece ana penceremizi açmaktı.

Widgetlar ile doldurmaya başlayalım biraz. Aşağıdaki kod satırı bir yalancı kod(psuedue code gibi bişeydi lan sanırsam ama ilk yazdığım kelimeyi büyük ihtimal yanlış yazdım) ve bize genel yazım şeklini gösterecek. Nitekim hiçbirini bilmeden hepsini nasıl yazacağımızı öğrenmiş olacağız bu şekilde.

my\_widget = Widget-name (its container window, \*\* its configuration options

Normalde bu noktada hep screenshotları da paylaşmayı tercih ederdim ama, hem üşeniyorum hem de merak edin görün kendiniz bakın. Gerçekten de bu methodlar birçok programlama dilinde olduğu gibi çalışır. İlk argumanı Label() methodunun, root oldu. Yani container’ı, içereni, parenti. Sonrada bu labelin etiketini string olarak yazmış oldunuz. Buton’da da işler birebir aynı şekilde yürüdü. Peki pack() methodu neydi? Temel olarak bu widget’ları tanımlarken söylemediğimiz ama söylememiz gereken bir şey oldu. O da nerede olmaları gerektiğiydi. Tamam event falan da halletmedik, sabredin lan biraz. Ama bundan da önce root üzerinde “şuraya geçin” demedik bu adamlara. Aslında dedik işte pack() ile. Geometrik dizilimleri açıklarken buna da değineyeceğim hiç merak etmeyin, şimdilik geçiyorum. Object-Oriented programlama olayını duymuş iseniz, bu widget tanımlamalarında dönen olay budur işte.

button = Button(its\_parent)

Fark edersiniz ki, bu örneklerden bir pattern yakalayabiliyoruz. Tabi kendine has özellikleri de olan bu zırvalıkları da ancak şu haliyle genelleyebiliriz:

Widget-name(its\_parent, \*\*its\_configuration\_options)

Burada parent ve root kullanımındaki hiyerarşiyi aşağıdaki frame kısmında daha iyi anlıyor olacağız. Çünkü genelde çat çat root’a yığmaktansa “holder”lar ile hiyerarşik düzende tutmayı tercih edeceksiniz bu widgetları.

Frame widgetlar demişken hemen değineyim aslında. Frame widgetlar “holder widget” olarak geçiyor. Yani birkaç widget’e sahip çıkan, bünyesinde barındıran widgetlar. Syntax’ı da farklı bir şekilde değil. Baktığınız zaman zaten bu widgetların hallerine, arayacağınız, customize etmek istediğiniz özellikler genelde size sunulan şeyler oluyor. Ve argümanların konumunu bilmeyi bekliyorsunuz sadece.

frame = Frame(root)  
frame.pack()

Bu konuda aslında farklı frameler üzerine widgetları dizip, işlevsiz ama hatları olabilen bir program inşa edebilmemiz mümkün. Bunun üzerine fikir kazandıracağım fakat pratikte henüz karşılığını görmek için biraz daha sabretmenizi isticem.

Farz edin ki, “menu\_bar” adında bir frame oluşturduk. İsminden de anlaşılacağı üzere bu kısımda menü ile ilgili widgetları konumlandırmak istiyoruz. Widgetları “menu\_bar” içinde konumlandırırken komple bu bar’ı da nereye koyacağımı düşünmeme gerek yok. İstediğim yer olabilir zira. Örneğin işte “menu\_bar”da şu widgetlar olur:

Menubutton  
Menu

Root’a parent yapacağımız, ana hattımızı taşıyan bir frame olur ki ona da keyfe göre şunları dizebiliriz:

Label  
Entry  
Button  
Checkbutton  
Radiobutton  
OptionMenu  
Bitmap Class

Umarım bu widgetların programa döşenilmesindeki fikir anlaşılmışdır. Çünkü buradan doğan bir gereksinim olan “geometry manager” kısmına bakıcaz şimdi.

Yukarılarda frame’lerde ya da widgetlarda kullanıp açıklamasını ertelediğimiz “pack()” methodu bu geometri yönetiminde kullandığımız methodlardan biri işte. Arayüzünün geometrisini yönetmekte kullandığımız bu methodla birlikte 3 method bulunmakta.

**pack**: Şuana kadar kullandığımız method olan pack methodu, basit layout’larda kullanılıyor. Fakat komplex layout’larda kullanımı da zorlaşabiliyor.

**grid:** Çok sık kullanılan geometry manager’ı diyebileceğimiz bir methoddur. Grid Türkçe’de “kafes,ızgara” anlamlarına geliyor. Gridi aslında tablo gibi düşünün, sahip olduğunuz şeyi küçük karelere bölen ve büyük resim itibariyle tablo gibi görünen bir yönetim methodu.

**place:** Diğer ikisine kıyasla, en az popüleri denebilir place methoduna. Ama yine de nereye koymak isteyeceğinizden emin olduğunuz bir widget için nokta atışı yapmak istiyorsanız kullanacağınız bir method.

Bu denli hızlı bir göz aşinalığı edindikten sonra bu geometri yöneticilerine, hemen sırayla derin bir dalış yapalım. Fakat place methoduna değinmeyeceğim.

# Pack

Bu methodu anlamamız için Tkinter’ın yazarı, Fredrik Lundh’ın kurduğu bir analoji üzerinden gidelim. Root’u, ortasında küçük açıklık bulunan elastik bir sayfa olarak düşünün. Pack methodu da, bu elastik sayfa üzerinde mevzubahis widget’ı alabilecek büyüklükte bir delik açıyor. Widget için, default olarak üst kenardan itibaren konumlandırma yapılmaya başlanıyor. Ardından tüm bu işleri, çağrılan tüm widgetlar için yapıyor. Peşisıra dizilen widget sayısına göre Bounding Box, yani sınırlandırılmış bölge hesaplanıyor. Analojide kurulan, sayfanın olayı da, tamamen bu belirlenime bağımlı bir değişken olduğundan dolayı “esnek,elastik” olmuştur. Parent’ı pack edilen tüm çocuk widgetları içine alabilecek kadar esnetiyor yani. Widget, doğası gereği, parent üzerinde kazandırılan alanın tamanını kullanamıyor. Çünkü widgetların, onlara ayırdığımız yeri ölümüne kullandığını varsayarsak bu estetik olmayan bir görüntü yaratır. Kendisinin üzerinde bulunduğu konumdan, 4 farklı konuma açıklıklar bırakarak yerleştirir widgetları. Bu da aşağıda literatürdeki haliyle aldığım, esnetme sonucu kazanılan 3 farklı boşluk türüdür:

• The unclaimed space  
• The claimed but unused space  
• The claimed and used space

Alınan ve kullanılan, alınan ve kullanılmayan yukarda bahsettiğim olaydı zaten. E hiç alınmayan da, kaçınılmaz durum. Aşağıya pack methodunun içerdiği seçenek ayarlarını koydum, bir göz atın:

• side : LEFT , TOP , RIGHT , and BOTTOM (widgetların hizalandırılması)  
• fill : X , Y , BOTH , and NONE (widgetların büyümesi olayına karar verir)  
• expand : Boolean değerler, şunlar gibi: tkinter.YES / tkinter.NO , 1 / 0 , True /  
False  
• anchor : NW , N , NE , E , SE , S , SW , W , and CENTER (kardinal yönlere karşılık geliyor)  
• Internal padding ( ipadx and ipady ) -filling- içine doldurma ayarı ya da Padding( padx and pady) dıştan doldurma ayarı. Hepsi default olarak sıfır değerine eşit.

Label’ı direkt oluşturup düz pack methodunu çağırdık, ve buraya kadar bildiğimiz bir şey. Button’daki argümanlara ne verildiğine bakarak sürekli resme bakmanızı istiyorum. Zira kısmen anlaşılabileceğini düşünüyorum. Fakat yine de dilim döndüğünce anlatmaya çalışacağım, rahat olun.

Önce teker teker detaylandırmadan önce bir şeyi fark etmenizi isticem. Üstteki butonların tamamını frame’e eklediğimizi ve alttakileri de direkt root’a eklediğimizi görün. Ve üstteki buttonlarda “fill=X” argümanının, yatay düzlemdeki genişlemenin ancak label’ın bittiği yere kadar olmasına sebep olurken, alttaki butonlarda “fill=X”in root’un bitimine kadar olmasından, umuyorumki layout’daki label ve root hiyerarşisi anlaşılmıştır. Ha diyeceksiniz ki, peki frame ne ayak. Ne yani onun sınırlarını ne belirliyor. Aslında root için de aynı soruyu sormalısınız, çünkü ikisi de aynı terane. Pack reis, size eklediğiniz widgetlar neticesinde bu frame’ler için istediğin kadarını veriyor. Widgetları yığarken frame’in sınırları umrunuzda değil açıkçası. İkisine de ekledikçe “esnediğini” göreceksiniz zaten.

Frame’e widget yığdığımız 5 satırlık sonunda frame’in sınırları zaten belirlenmiş oluyor. A butonunu sola yatırıp dikey düzlemde uzat demek aslında az sonra belirlenecek sınırlara göre bunu yapmasını söylemek demek oluyor. Az sonra belirlenecek bu sınırların üzerinden şimdi konuşmak bize yine kolaylık sağlıyor, çünkü olmasını istediğimiz yer her halükarda sola yapışık ve o alanda dikeyi işgal etmesi şeklinde olacak. Ve “frame.pack()” satırına baktığınızda aslında bu kısmı da pekala detaylandırabileceğimizi fark edebilirsiniz. Fakat düz çağırdığımızdan dolayı sadece minimum koşullarda istediğini alıp işini bitiriyor. Bu durum şu anlama geliyor: Eğer root penceresini sağdan soldan genişletir, ya da direkt büyüt penceresine basarsanız, frame her zaman olduğu yerde kalacaktır. Kodu çalıştırıp hemen deneyin ve görün. Bunu yaptığınızda hem frame olayını kavrayacak hem de root düzleminde fill ve expand’ın farkını daha net göreceksiniz. Fakat burada dikkat çekmek istediğim daha nokta daha ziyade frame’in statik bir konumlandırılmasın olması, tabi pack methodunu seçeneklendirmenize bağlı.

expand ve fill’in farkına bakalım. Direkt olarak pencereyi büyütmenizi istiyorum sizden. ne fill ne de expand seçeneği verilmiş buton, direkt olarak frame’in “top” kısmından başlayarak peşisıra konumlandırılan butonlarının altına konumlandırıldı. Ve pencereyi büyüttümüzde bu değişmedi, bu durumda aynı frame’de gözlemlediğimiz statik mevzu burada da rol oynadı. Fakat hemen altındaki buton için gözlemlediğimiz üzere, ne zaman ki “expand=1” dedi, bu sefer root penceresine göreli hareket etmeye başladı widget. Yani dinamizm kazanarak, ilk andaki konumunu, root değiştikten sonra bu değişime ayak uydurarak korumuş oldu. Fill argümanı ise verilen koordinat düzleminde, “root ne olursa olsun sen genişlemene bak” der gibi çalıştı. Bunları anlamanın en kolay yolu pencerenizle oynamak olacaktır. Bu arada küçük bir not, örneğin “side=TOP, anchor=SE” demek yerine “side=’top’, anchor=’se’” demek aynı şeye karşılık gelecektir. Pack geometri yöneticisine genelde widgetlerın alta alta ya da yan yana konumlandırılması söz konusu olduğunda.

Grid methoda kıyasla biraz daha karışık olduğu aşikar pack’in. Fakat şu iki durum kullanımı kaçınılmaz:

**1.** Frame’in tamamını doldurması gereken bir widget söz konusu olduğunda.  
**2.**Alta alta ya da yan yana sıralanması gereken widgetler söz konusu olduğunda.(üstteki bahsedilen)

Farklı frame’leri birbiri içerisinde nest’leyerek komplex layoutlar yaratabilirsiniz fakat bu gibi durumlar için grid daha çok kullanışlıdır.

# Grid

Anlaşılması en kolay ve Tkinker’daki en yararlı geometri yöneticisi grid diyebiliriz. Grid’de anafikir, satır ve sütunlardan oluşan bir tablo şekline bölünen frame’i, buna göre organize etmektir. Her bir hane, bir tane widget’ı tutmak için görevlendirilebilir. Bu bağlamda, bir hane hayali satır ve sütunların kesişimidir. Ve bir hane sadece bir widget tutabilir. Ancak widgetlar birkaç haneyi işgal edebilir. Her bir hanenin içerisinde “sticky” opsiyonu kullanılarak widget konumlandırılabilir. “sticky” opsiyonu widget’ın nasıl genişletildiğini söyler aslında. “Sticky”de bu opsiyonlar “N,S,W,E,NW,NE,SW,SE”(north,south,vs) şeklinde yönler ile belirtilir. Örnek üzerinden bakalım:

Resme baktığınızda, kodda row ve column değişkenlerine verilen argümanlara göre hayali bir tabloda yerleşkeler mantıklı gelecektir. Sticky argümanındaki opsiyonlardan, örneğin ilkinde, “W” yazarak widget’ın hizasını sola dayalı yapmıştır. Aynı pack methodunda olduğu gibi burada da tüm widget’lara göre bir tablo uzunluğu ve genişliği belirlenir. Ve dolayısıyla bunlara göre, hanelerin boyutları da belirlenmiş olur. Tabi kontrolü sağlamak istiyorsanız da, hanelere boyutlarını siz atayabilirsiniz. Örneğin bu durumda “sticky=NSEW” yaparak widget’ı genişletilebilir ve grid’i tamamen doldurabilir hale getirebilirsiniz. Tabi grid kullanırken sıkça ihtiyaç duyacağımız “row, column, padx, pady, rowspan, columnspan” gibi argümanlar olacak ve örneklerle anlaşılır diye umuyorum.

Bir ton kod var ama hepsi birbirinin aynısı neredeyse, boş kalabalık yani. Bu denli komplex bir düzeni pack ile yapmak daha pahalıya patlardı.

Fakat buraya kadar olan kısımlar dizayn olarak çoğu zaman bir de foreground’a sahip olabilen kısımları içeriyordu. Yani JavaFX’de SceneBuilder varken, bu tarz layout falan filan uğraşayım demezsiniz sanıyorum. Python’da, “PAGE — Python Automatic GUI Generator” programı ile mümkün olan aynı olayı da bu kısımlara tercih edebilirsiniz. Bizi ilgilendiren bunlardan ziyade, programa hayat katmak. Yani event-handle. Umuyorum ki, Java ya da Python üzerinden programın matematiğinin -background- konseptini net şekilde ele alan bir makaleyi daha sonralarda eklerim.

###### 2.4. ÜRETİM(GERÇEKLEŞTİRME)

Projede planlama, analiz ve tasarım aşamasında sonra yapılacak işlemlerin detaylı bir şekilde geliştirildiği aşamadır. Bu noktadan sonra önceki adımlara geri dönüş olmamalıdır. Üretim aşamasına geçmeden önce gereksinimlerin iyi belirlenmesi, analizlerin düzgün yapılmış olması gerekmektedir.

###### 2.5. TEST

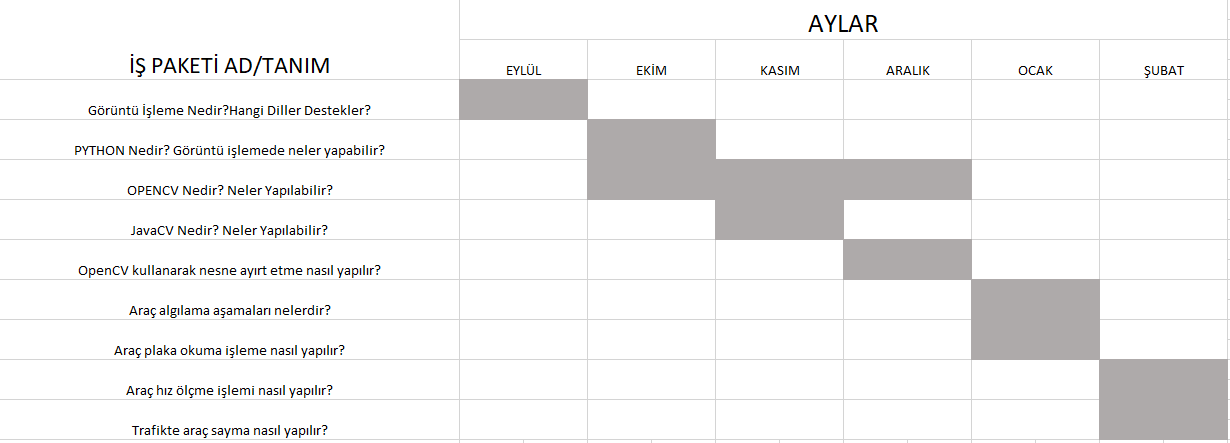
Test aşaması geliştirilmesi tamamlanmış projenin müşteriye sunulmadan önce test ekibi tarafından beta testlerinin uygulandığı kısımdır. Bulunan hatalar giderildikten sonra proje öğrenci ve öğretim görevlilerine sunulmalıdır. Testler veri setlerindeki videolarla test edilerek proje başarı oranını test edilir.

###### 2.6. BAKIM

Proje yayınlandıktan sonra istenmeyen hataları önlemek, stabil halini korumak ve yeni işlevlerin eklendiği kısımdır. Gelen istekler ve geri bildirimler ile bakım aşaması sağlanmaktadır. Proje kullanıldığı sürece bakım aşaması devam etmektedir.

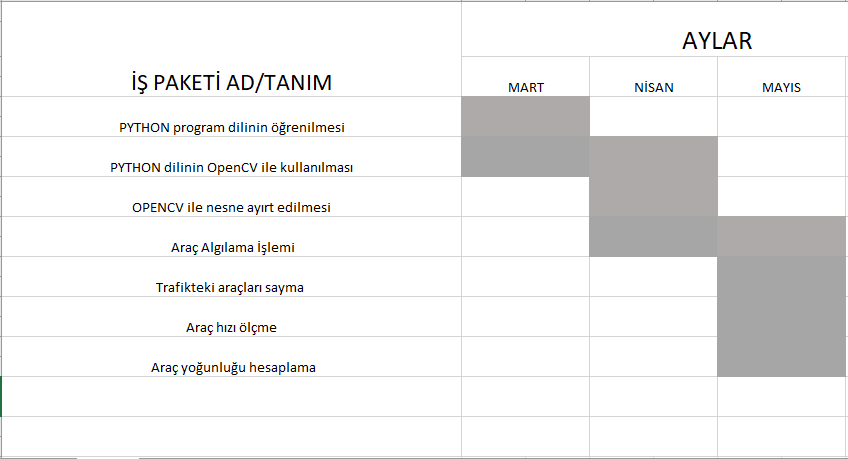
## PROJE İŞ-ZAMAN PLANI

Taraması haziran aylarında başlayıp proje teslim edilene dek sürecektir. Haziran Eylül ayları arasında gereksinim analiz yapılmış bulunmaktadır. Ekim ayından itibaren tasarım çalışmasına başlanmıştır. Bu çalışmalar aralık ayının son haftasına kadar devam edecektir. Bu proje 2020 yılının mayıs ayına kadar devam edecektir.



Şekil 14. Yapılan iş paket tablosu

Bundan sonraki aşamada projenin gerçekleştirim ve test çalışmalarının beraber yapılması, proje tamamlanıncaya kadar devam edecektir. Tüm bu aşamalardan sonra sistemin son testleri yapılacak ve hedeflenen ölçütlere ulaşılması sağlanacaktır. Yapılacak olan işler Şekil 15. Yapılacak olan iş paket tablosunda belirtilmiştir.



Şekil 15. Yapılacak olan iş paket tablosu

## 4. SONUÇLAR

Hedefler doğrultusunda, projem şirketler de hali hazırda kullanılan el ile tespit yerine geçecektir. Günümüzde gelişen teknoloji ile birlikte, şirketlerde karşılaştığımız video işleme sorunlarını ortadan kaldıracak sistemi tasarlamak amaçlanmıştır.

Projenin hem trafikte çalışanların hem de uygulamalarda ki videolar için değerli zaman kazandıracağını, kaliteli ve yaygın olarak kullanılacağını beklenmektedir.

## PROJE KURULUMU

## Python sürümü 3.6.8

* OpenCV sürümü 4.0
* PyTesseract v 0.3.7
* Tkinter
* Numpy
* Dlib
* Time
* Threading
* Math

Yukarıda yer alan tüm kütüphaneler indirilecek.

1. **PROJE LİNK**

https://github.com/Emredogangun/Mobese

## 5. KAYNAKLAR

1-) <https://maker.robotistan.com/>

2-) [https://cahitisleyen.wordpress.com](https://cahitisleyen.wordpress.com/)

3-) [https://muhammeddincer.com](https://muhammeddincer.com/)

4-)<https://peakup.org/>/

5-[)](https://docs.python.org/3/library/asyncio-eventloop.html)[https://medium.com](https://medium.com/)

6-[)](https://cloudbunny.net/docker-nedir-avantajlari-nelerdir-447.html)<https://cokcabilgi.com/>

7-[)](https://websockets.readthedocs.io/en/stable/intro.html)[https://www.mobilhanem.com](https://www.mobilhanem.com/)

8-[)](http://www.buyukveri.co/python-sinif-class/)[http://mesutpiskin.com](http://mesutpiskin.com/)

9-[)https://maker.robotistan.com/python-ve-opencv-ile-temel-geometrik-sekillerinalgilanmasi/](https://maker.robotistan.com/python-ve-opencv-ile-temel-geometrik-sekillerin-algilanmasi/)

10-[)https://pypi.org/project/opencv-python/](https://pypi.org/project/opencv-python/)

11-)Eldem A, Eldem H ve Palalı A, Görüntü İşleme Teknikleriyle Yüz Algılama Sistemi Geliştirme BEÜ Fen Bilimleri Dergisi BEU Journal of Science 6(2), 2017, 44-48.

12-)<https://www.elektrikport.com/>

13-)[https://teknolojiprojeleri.com](https://teknolojiprojeleri.com/)

14-) <https://www.pythontr.com/>

15-) Güler, E., & Geçer, B. (2013). People Counting. <http://www.ebubekirguler.com/goruntu-isleme-yontemleri-ile-insansayma/>.

16-) El-Azim, S. A., Ismail I., & El-Lati, H. A. (2002). An efficient object tracking technique using block-matching algorithm. Proc. Of the Nineteenth National, Radio Science Conf., 427-433.

17-) Wren, C., Azarhayejani, A. Darrell, T., & Pentland, A.P. (1997). Pfinder: real-time tracking of the human body. IEEE Trans. on Pattern Analysis. And Machine Intelligence, 19, 7, 780-785.

### STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Buna benzer projeler elbette olduğunu düşünüyorum ama şirketlerin kullanması bu kadar kolay ve işlevsel olduğu için yeni bir projedir.

1. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Proje henüz yapım aşamasında olduğu için henüz tamamlanmamıştır.

1. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Yapay zeka, Görüntü işleme ve Algoritma derslerindeki hemen hemen bütün konulara dair bilgi ve beceriler kullanıldı veya kullanılacak.

1. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

SOLID prensiblerini uygulamaktayım.

1. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

1. Ekonomi

Projemiz bir yazılım projesi olduğu için ekonomik kısıtı yoktur.

1. Çevre sorunları:

Projemiz de görüntü işleme tabanlı olduğu için görüntünün alınması sırasındaki sorunların tümü çevre sorunları olarak geçmektedir.

1. Sürdürülebilirlik:

Projemiz SOLID prensiplerine göre gerçekleştirildiği için ihtiyaca göre eklemeler yapılarak kullanılabilir.

1. Üretilebilirlik:

Projemizde bir üretilebilirlik kısıtı bulunmamaktadır.

1. Etik:

Projemizde bir etik kısıtı bulunmamaktadır.

1. Sağlık:

Projemizde bir sağlık kısıtı bulunmamaktadır.

1. Güvenlik:

Projemizin gerçekleştirimi ve kullanımı boyunca herhangi bir güvenlik sorunu bulunmamaktadır.

1. Sosyal ve politik sorunlar:

Projemizde sosyal ve politik sorun oluşturabilecek herhangi bir öğe bulunmamaktadır.