
# Gebze Teknik Üniversitesi
# Computer Vision Dersi
# 4. Ödev
# Rapor

141044090
Emre Aydın

# Ödev Açıklaması

**Gebze Technical University**
**Department of Computer Engineering**
**BIL 665 / BIL 463**
**(Introduction to) Computer Vision**
**Fall 2016**
**HW4**
**Jan 5th 2017**
**No late submissions**

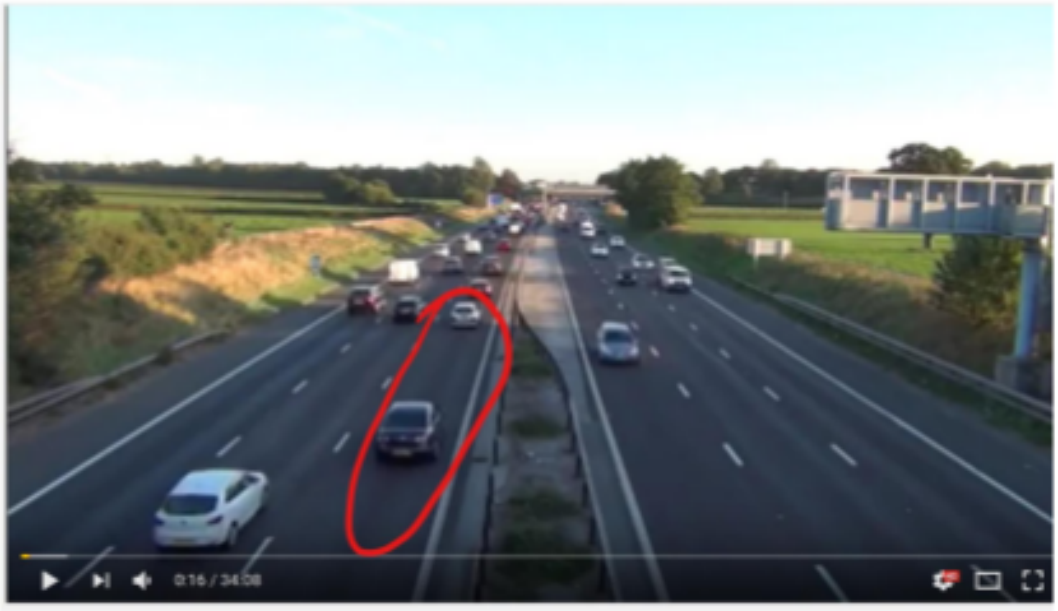
In this homework, you will recognize and count vehicles on a road using a surveillance camera. The videos that you will use for this HW can be seen at
https://www.youtube.com/watch?v=PNCJQkvALVc

Download the video in HD 1080p format.

You will count the number of vehicles, identify their types (trucks, cars, buses, minivans) and print this information at the top of the screen in real time for the marked lane shown below. Here are the steps that you may follow

- Estimate the background using OpenCV routines.
- Find the foregound.
- Handmark vehicles for at least 10 minutes of video for machine learning training.
- Use HOG + SVM classification for vehicle recognition and classification.

Write a 3 page report that discusses your algorithm and your performance results.



# RAPOR

Bu ödevde SVM(Support Vector Machine) ve HOG kullanılarak verilen videodaki araçların tespiti yapılması istenmiştir.

Bunun için öncelikli olarak problemi parçalara ayırdım.İlk iş olarak ham araç verisi elde etmeyi planladım. Bunun için videoda ilk 10 dakikadan geçen araçları kırparak ham veri elde edeceğim. Daha sonrasında aldığım tüm resimleri kendim ayrı ayrı kümeleyerek klasör haline getirip bir veri kümesi elde etmiş olacağım. En son olarak artık hazırladığım veri kümesinden bir HOG listesi çıkarıp videoda araçları tespit edeceğim.

Bu raporda bu iki kısımdan, yaşanılan problemlerden ve çözümlerinden ayrıntılı olarak bahsedilecektir.

# 1.Adım Araç Verisi Elde Etme:

Bunun için ilk önce şöyle bir yöntem geliştirdim ; Videodan bir sahne alıp bu sahneyi 0,0(satır sütün) konumundan başlayarak eşit aralıklara 125 e 125 lik karelere ayırdım ve elime geçen bütün subimageleri dosyaya yazdırdım.

Elimde en son olarak 50000 üzerinde görüntü vardı.Daha sonra bunları incelemeye başladım.Elde ettiğim görüntüler, tamamen rastgele olduğu için araçların tam olarak yakalanabildiği görüntü sayısı oldukça düşük sayıdaydı ve elimde çok fazla sayıda bakmam gereken görüntü vardı … Bundan dolayı yeni bir yol aramaya başladım.Geliştirmiş olduğum yöntem aşağıdadır.(Bu yöntem ekte createRawDataWithConstantSize.cpp dosyası olarak eklenmiştir.)

CLion is ready to update.

```
int createRawDataWithConstantSize(string videoName,string path,string extension,
                                  Size rawDataSize,int frameNumber){
    // Define video capture.
    VideoCapture videoCapture;
    videoCapture.open(videoName);
    if (!videoCapture.isOpened()){
        cerr << "Unable to open the video" << endl;
        return -1;
    }

    Mat frame;
    int imgCounter = 0;
    bool flag;
    for (int i = 0;
         i != frameNumber && videoCapture.read(frame) != false;
         ++i){
        cerr << i << endl;
        for (int j = 0; j + rawDataSize.height < frame.rows; j += rawDataSize.height) {
            for (int k = 0 ; k + rawDataSize.width < frame.cols; k += rawDataSize.width){
                Mat data = frame(Rect(k,j, rawDataSize.height, rawDataSize.width));
                flag = imwrite(path + to_string(imgCounter) + extension, data);
                if (!flag) {
                    cerr << "Create raw data failed" << endl;
                    return -1;
                }
                ++imgCounter;
            }}
    return 1;
```

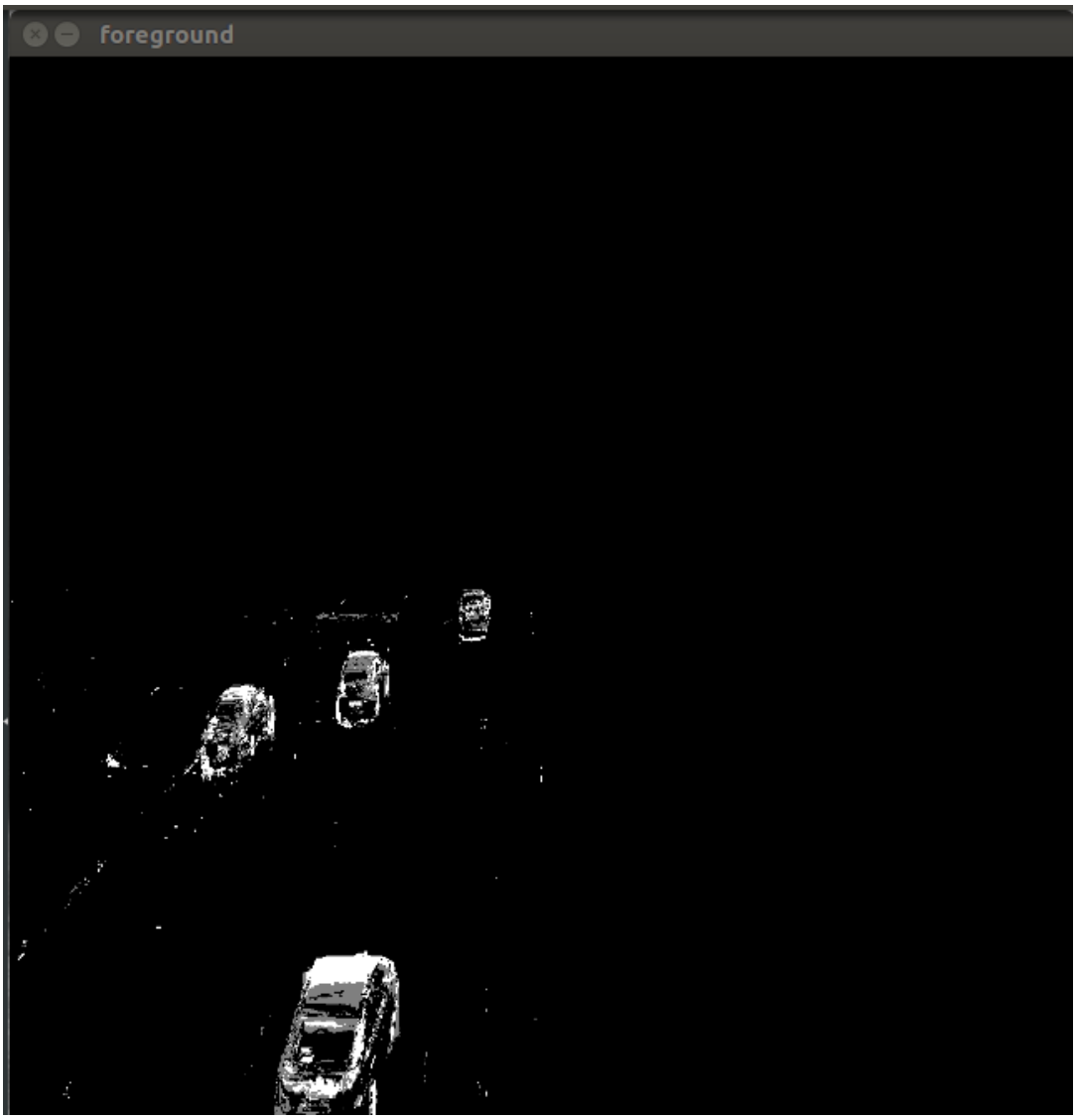
Bu yöntem çok fazla zamanımı aldığı için yeni bir yöntem aramaya başladım.

Ödev pdfinde geçen background subtraction metodunu incelemeye başladım. Bu metodda kullanılan MOG2 ve KNN yöntemleri ile karşı karşıya kaldım. KNN MOG2 ye göre daha kaliteli çalışmasına rağmen yavaş çalıştığı için MOG2 ye yöneldim.Öncelikli olarak http://docs.opencv.org/3.1.0/d1/dc5/tutorial_background_subtraction.html

Linkini vermiş olduğum sitedeki örneği kodu koduma göre çevirip denedim. Örnek koduma çevirirken çok uzaktaki araçları ve ters şeritteki araçlar ile uğraşmamak için o kısımları siyaha boyadım.

```
for (int pi = 0; pi < original.rows; ++pi)
    for (int pj = 0; pj < original.cols; ++pj)
        if (pj > original.cols / 2 || pi < original.rows / 2) {
            original.at<Vec3b>(pi, pj)[0] = 0;
            original.at<Vec3b>(pi, pj)[1] = 0;
            original.at<Vec3b>(pi, pj)[2] = 0;
```

Bu metod sayesinde foreground maskesini elde edebiliyordum.Elde ettiğim foreground maskesi aşağıdaki gibiydi.




Daha sonra elde ettiğim maskedeki siyah noktaları gerçek görüntüde de siyah ile işaretledim ve elde ettiğim görüntüdeki arabaları program yardımı ile kırpmaya başladım. Fakat bunu daha nasıl akıcı hale getirebilirim diye

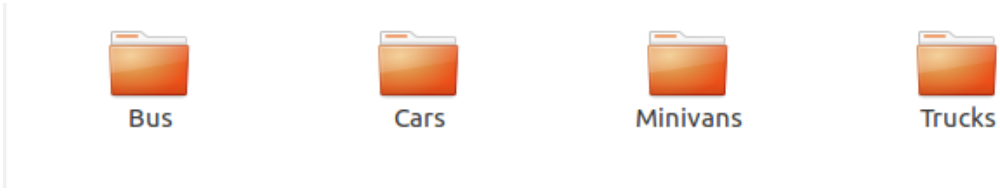
düşünürken `findContours` yöntemini buldum bu yöntemde verdiğim görüntüyü griye çevirip arabaların yerini çerçeve içine alabiliyordum. Bu da işimi oldukça kolaylıştırmıştı. Fakat aldığım görüntüde gürültü çok fazla olduğu için istediğim şekilde findContours fonksiyonunu istediğim gibi kullanamıyordum. Bunun için gürültüyü nasıl azaltabileceğimi araştırmaya başladım. Ve OpenCV dökümantasyonunda bu sayfa ile karşılaştım : http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html

Bu iki fonksiyon sayesinde resimdeki gürültüyü çok büyük oranda azaltmayı başardım.Ve sonunda findContours istediğim gibi çalıştırarak arabaları otomatik bulup bulduğum tüm arabaları dosyaya yazdırıyordum.

Elimdeki verinin büyüklüğü 4 te 1 oranın düşmekle kalmayıp elimde ilk haline göre çok çok daha sağlıklı veriler bulunmaktaydı. İlerleyen zamanlarda bunu daha da ilerletip sadece istediğim arabayı dosyaya yazdıracak şekilde kodumu düzenledim ve son olarak aşağıdaki halini almış bulunmaktadır.


Görüntüyü frame frame ve ya otomatik şekilde ilerleterek istediğim görüntüyü istediğim zaman durdurak elde edebiliyordum. Artık ham verim tamamlanmıştı.(Ham veriyi elde eden bu kod ekte : createRawDataBGSubtraction adı ile belirtilmiştir.)

Daha sonra görüntüleri 4 farklı araç tipi için oluşturmuş olduğum klasörlerin içindeki pozitif kısmına yerleştirildim. Her araç tipi pozitif ve negatif olmak üzere 2 alt klasörden oluşmaktadır. Ayırdığım görüntüler ile tüm araçların pozitif kısmını doldurdum daha sonra her aracın kendi pozitif verilerini kendisi haricinde geri kalan araçların negatif kısımlarına yerleştirdim.



## 2. Elde Edilen Verinin İşlenmesi:

Bu kısımda artık elde ettiğim görüntüleri işlemem gerekiyordu.Bunun için öncelikli olarak hazırladığım veri kümesindeki tüm resimleri aynı boyutlara getirecek basit bir kod yazdım.(Bu kod resizeAllImages.cpp adı ile ekte yer almaktadır.).

Artık veri kümemi tamamlamış düzenlemiş olarak araç tespit etme kısmına başladım. Buradan sonraki kısımda yapmış olduğum implementasyon adımlarını anlatacağım.

1) Oluşturmuş olduğum 4 araç tipi için toplam da 8 klasörün her biri için ayrı ayrı olarak aşağıdaki işlemleri gerçekleştirdim:

1.a ) Belirtilen klasördeki tüm resimleri bir listenin içine doldurdum.
1.b ) Listedeki her bir eleman önce griye çevrilip daha sonrasında hog compute fonksiyonuna verilerek oluşan outputlar başka bir listede toplanmıştır.
1.c ) Oluşan çıktı listesi her bir araç için pozitif negatif mi olduğuna bakılırak 4 tane descriptor list oluşturulmuştur.

2) Oluşturulan 4 ayrı HOG Descripterı ayrı ayrı svm train metoduna verilerek train ettirilmiş ve 4 ayrı dosyaya sonuçlar yazılmıştır.

```
svm->train(trainData, ROW_SAMPLE, Mat(carHog));
svm->save(savedFile);
```

3 ) Training işleminden sonra ise bu ayrı 4 descriptor için her görüntüden sonra detectMultiScale metodunu çağırarak çalışmasını sağladım.

# Yaşanılan Problemler ve Çözümleri:

1. Sorun : FPS Problemi.
Açıklama : Programı ilk açtığımda saniye de 1 ya da 2 saniyede bir frame ilerliyordu. Bunun sebebi detect metodunun her frame üzerinde lineer olarak ilerlemesi ve benim her frame de 4 detecti de bekliyor olmamdan kaynaklı olarak oldukça yavaş çalışıyordu.

Çözüm 1:

- Thread : Problemin sebebi 4 ayrı detectin hepsini tek tek bekliyor olmamdı. Buna çözüm olarak bu 4 işlemi de aynı anda yapacak 4 thread oluşturuyorum ve bu 4 threadin işlerini bitirmelerini bekliyorum.

Döngü içinde oluşturulan threadler :

```
for (int i = 0; i < VEHICLE_TYPES; ++i) {
    int *arg = (int *) malloc(sizeof(*arg));
    *arg = i;
    pthread_create(&mainThreads[i], NULL, detect, (void *) arg);
}

for (int i = 0; i < VEHICLE_TYPES; ++i)
    pthread_join(mainThreads[i], NULL);
```

Detect metodu:

```
vector<Rect> locations;
int ind = *((int *) type);
int order;
locations.clear();
hogDescriptor[ind].detectMultiScale(copyFrame, locations,0,Size(),Size(),1.10)
```

Çözüm 2:

- Resmin boyutunu küçürmek.Eğer resmin boyutunu küçültürsem daha az karşılaştırma olacak ve daha az işlem yapılacak bu yüzden de hız artacaktır.

```
videoCapture >> frame;
resize(frame, frame, Size(640, 640));
copyFrame = Mat(frame, Rect(0, frame.rows / 2, frame.cols / 2, frame.rows / 2));
drawImage = frame.clone();
```

Yukarıda belirtilen resimde resize fonksiyonu ile videodan alınan frame 640,640 formatına resize edilmiştir. Daha sonrasında detect fonksiyonunda kullanılmak üzere copyFrame oluşturdum. Bu frame e ise resmin sol alt köşesini yani arabaların gitmekte olduğu bizim ilgilendiğimiz kısımı kırptım. Böylece ters yönde giden trafiği ve çok uzaklarda giden araçlar ile uğraşmaktan kurtulmuş oldum.

2. Sorun : Veri Kümesi problemi
Açıklama : Programı ilk başlattığımda gökyüzünü bulutları vs araba olarak işaretlediğini fark ettim bunun nedeni negatif kısımlara hiç gökyüzü ve ya hiç araba olmayan resimleri eklememiş olmamdı.

Çözüm : Veri kümemin negatif kısımlarını gökyüzü resimleri ile doldurdum.

3.Sorun : Tır ve Otobüs problemi.
Açıklama : Videoda tır ve otobüs sayısı oldukça azdır bundan dolayı tespit etmek oldukça zor.

Çözüm : Aynı tırın farklı açılardan görüntülerini ekledim. Fakat tam bir çözüm olmadı.

Yapılan ve İncelediğim Benzer Bazı Çalışmalar:

→ https://github.com/bikz05/object-detector

→ https://github.com/CheriPai/car-detector

→ https://github.com/Kulbear/MNIST-SVM

→ https://github.com/AbdealiJK/object-detector

Çalıştırılması :

```
$ cmake CmakeLists.txt
$ make
$ ./141044090
```