ANKARA UNIVERSITY

FACULTY OF ENGINEERING

ELECTRIC-ELECTRONIC ENGINEERING DEPARTMENT

EEE 482 PROJECT REPORT

SMART TRAFFIC SYSTEM

FURKAN ÖZKAN 16290608

Süleyman Cevdet ÖZBEY 16290617

ASSISTANT PROFESSOR GÖKHAN SOYSAL

Contents

INTRODUCTION

1.1 INTRODUCTION

Today, the traffic has become mixed with the proliferation of vehicles. The most effective way to regulate traffic is traffic police. However, it is not possible to have traffic police everywhere, so the traffic lights automatically regulate. Traffic lights now turn red, yellow and green color cycles for certain periods. The duration of this cycle varies according to the number of traffic and pedestrians. However, this period does not change into the day, depending on the region.

Therefore, we decided to work to adjust the time of the traffic lights change according to the traffic density. It is necessary to know the number of vehicles, the number of pedestrians, and the intensity of the traffic lights to adjust the changeover times.

We decided to move forward in this project based on the definition of the number of vehicles, we will work on finding the number of vehicles with image processing.

Our analyzes on the images in the project, will be done in **MATLAB** environment.

MATLAB is a multi-paradigm numerical computing software and a fourth-generation programming language. Developed by MathWorks. Matlab allows matrix operations, function and data drawing, algorithm development, user interface, as well as working with programs written in other languages such as C, C ++ and Java. MATLAB performs many mathematical calculations effectively and quickly, such as linear algebra, statistics, optimization, numerical analysis, optimization, and fourier analysis. Also MATLAB has lots of tools for image processing techniques. So we are working in MATLAB environment.

1.2 SUMMARY OF LAST SEMESTER

Last term, we worked for the smart traffic system to increase our basic knowledge before we started working on cars or on the road. In this context, it first defined the MATLAB environment and performed image upload operations. We examined the matrix of the uploaded picture. We then examined noise cleaning and rng layers with filters such as medfilt() through this image. We had a black and white binary image that we wanted to work on by performing the substraction operations that we needed to do to get the layer of color we wanted. After this stage, we tried to get to know various shapes.

To identify shapes, we found the center point coordinates of each shape at the matrix. For this, we used the regionprops() function. We used "for" loops to use the data we obtained in images with multiple objects. We classified these objects in various situations with if else command using the location diameter, radius, circularity. We used functions such as rectangle() and viscircles() to mark the classified shape.

Thus, we identified the shape of the object by obtaining values such as circularity by finding and using pixel lengths in black and white cases by using the information on the matrix by first parsing the objects according to the colors and determining the number of them according to the situation.

Also we examined functions like regionprops(), finding centroid, major minor length, area finding, bounding box, extent, radius, diameter, circularity, edge detection and grain last semester.

With this work, we have taken the preliminary step towards our work on determining the density of cars and recognizing the number and regulating traffic.

2 GENERAL INFORMATIONS

2.1 PROJECT OBJECTIVE

Count the number of vehicles on the roads with the camera at traffic intersections and change the green light setting. In this way, the green light time will be very short on empty roads and the number of vehicles will be much more green light time. In this way, the traffic density and fuel waste and environmental pollution caused by continuous vehicles stop and go will be reduced. In the first phase of the project, we can provide some convenience in distinguishing between circle, circle, rectangle and triangles. For example, when the correct color filters are applied by distinguishing circle-like shapes, an X-ray result can be used to determine whether tumor-like objects are applied. Or, if it is known in the field of technology that capacitors are rectangular in the card of any device, it can be verified by checking the top images of the cards produced by image processing to check if the rectangular shape is below a certain number, and that it is a manufacturing error.

2.2 PROGRESS STEPS OF PROJECT

In order to make the smart traffic control system in the first step, we will first work on identifying and counting cars on a photo using the basic functions and methods we use in the first part of the project. We will explain some of the new functions we use here and the code we wrote on MATLAB. Then we'll be trying to apply the method on the image in the video. In the first period, we had a tracking of a moving ball in a video.

In the second step, we will use matlab toolboxes and various functions. We will examine all the functions here separately and explain them. First of all, we will examine how the location or background is determined with the foreground() function. We will then use what we call blob analysis to identify and mark cars in black and white (binary) images. Of course, these definitions will apply only to a frame in the video, so we will review each frame separately, determine and mark the number and location of the cars (put them in a box with bbox) and play the framas into videos again. This allows us to examine the number and location of cars in the video.

3 FINDING AND COUNTING VEHICLES IN THE FOTO

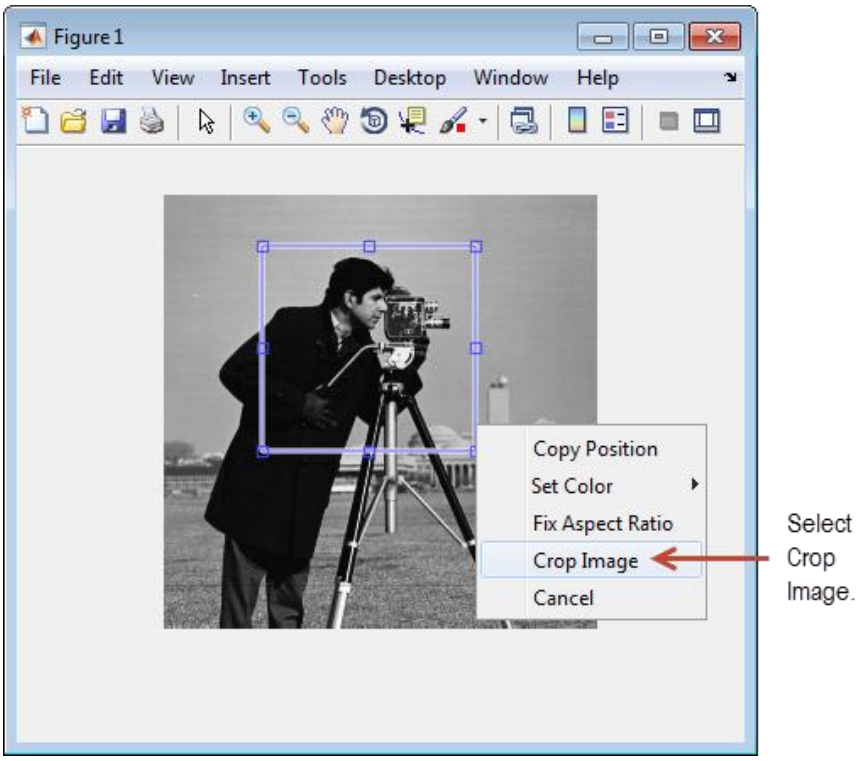
In this section, we'll try to find a car on the picture before we go on video. After our actions on the picture, we will transfer it to the video environment. 1 among the codes we write. We did not disclose some functions in detail because there are functions that we have written during the period. If you want to look at it in more detail, go to 1. you can look at our thesis we've written. We will show you the whole version of the first head code, and then we'll explain each line.

```
clc; clear;
car_number=0;
car=imread('C:\location\name.jpg');
car=imcrop(car,[300 175 520 1000]);
figure
imshow(car);
gray=rgb2gray(car);
   bw=im2bw(gray, 0.75);
   bw=medfilt2(bw, [3 3]);
   bw=bwareaopen(bw, 2000);
   bw = bwlabel(bw, 8);
   cc=bwconncomp(bw, 8);
   stats = regionprops(bw, 'BoundingBox', 'Centroid','Extent');
figure, imshow(bw);
figure
imshow(gray);
figure
imshow(car);
hold on
   for object=1:length(stats)
      A=stats(object).Extent;
      bb = stats(object).BoundingBox;
      bc = stats(object).Centroid;
      if (0.2<=A)&&(A<=1.8)
         car_number=car_number+1;
         rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
         a=text(bc(1)+15,bc(2), strcat('Car   ',num2str(car_number)));
         set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
      end
   end
   hold off
```

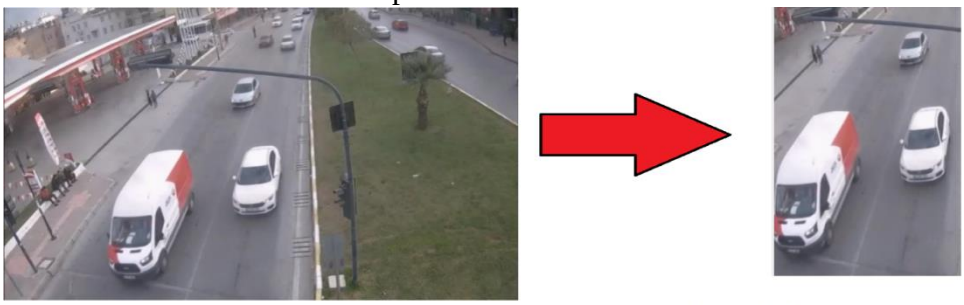
imcrop:

This function helps to crop pictures. We use it only to leave the space on the picture that we want to process. We're writing the first picture we want to crop into the function. There are 8 types, but there are 2 most commonly used types. These two types:

1- J=imcrop(I): With this command, we can trim the desired part of the picture. We plug the mouse's right key where we start, and then we expand it as far as we want to crop. It can only be clipped to the rectangle shape. The example of the picture is below.



2- J=imgrop(I,rectangle): We used this function in our study. After writing the picture we want to crop, we enter values [x, y, width, height]. We trim the picture with the values we specify. (x,y) values specify which point to start on the picture. (width, height) values determine how long and wide it will be after the starting point. For example, when we enter values [20 20 20 100 150], start at (20.20), "100" pixel width and "150" pixel length skirmish. The illustrated example is below.



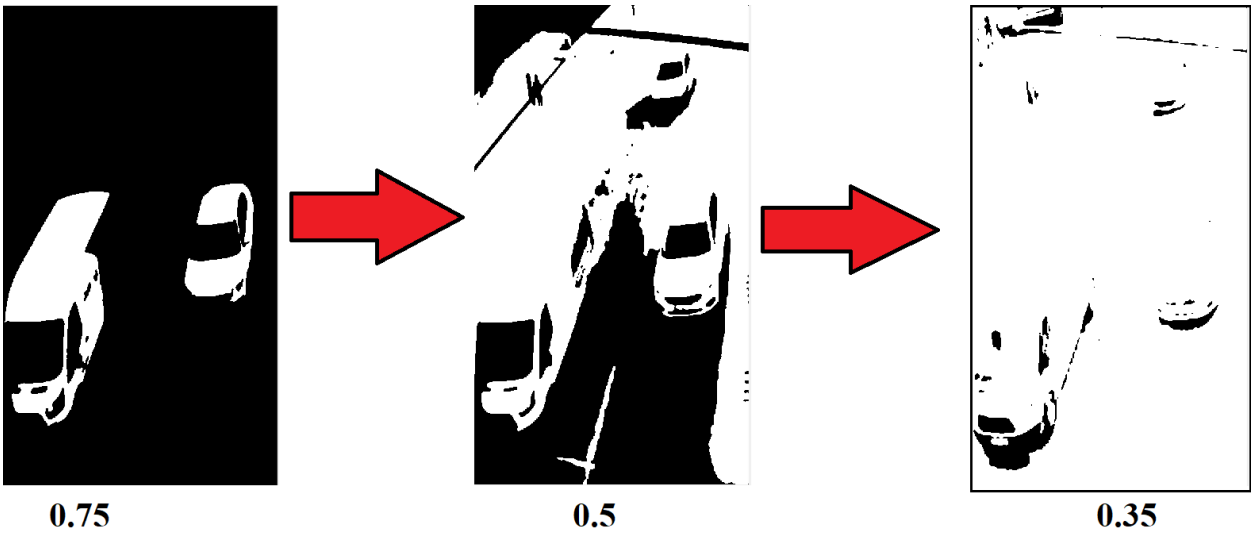
Orjinal resim

[300 175 520 1000] parametleri ile kırpıldığında

The reason we use type 2 is because after switching to video media, we're going to have to go to the 2nd floor. because the type is more useful.

im2bw(Picture, Number (between zero to one)):

Im2bw command 1. in detail in the thesis of the period. To put it simply, we were taking our picture of rgb space to gray-scale format first, then we were in black-white (bw) format. Black-white space is made up of only zeros and one, making it easier for us to process on them. Unlike the one we use in the 1st semester, we enter a number in the im2bw command in the range "0" and "1". We're determining how sensitive this range should be when we take our picture from gray-scale to black-white. To open up a little bit more, gray-scale space is a set of values ranging from "0-256". If we type without any numbering to the im2bw command, it automatically gets the value "0.5". This means if the number in the pixel you correspond to in gray-scale space is less than 123, accept "0", if the number in the pixel is equal to or greater than 123, accept "1.". When we change the rate "0.5", the value "123" that we accept changes at the same rate. Below is the same picture shown in different proportions.



0.75 **0.5** **0.35**

As shown in the pictures, the value is "0.75", while black density is higher and only our cars stay, which will be much easier to process on. As the value decreases to 0, the white ratio in the picture increases, which prevents us from taking action.

clc; clear;
It is used to delete the writings in the order window at Matlab and the previously recorded data. Our goal of doing this is to keep things in the command window if we want to examine something later. The reason we delete data that was previously held is to prevent errors, such as possible data mixing.

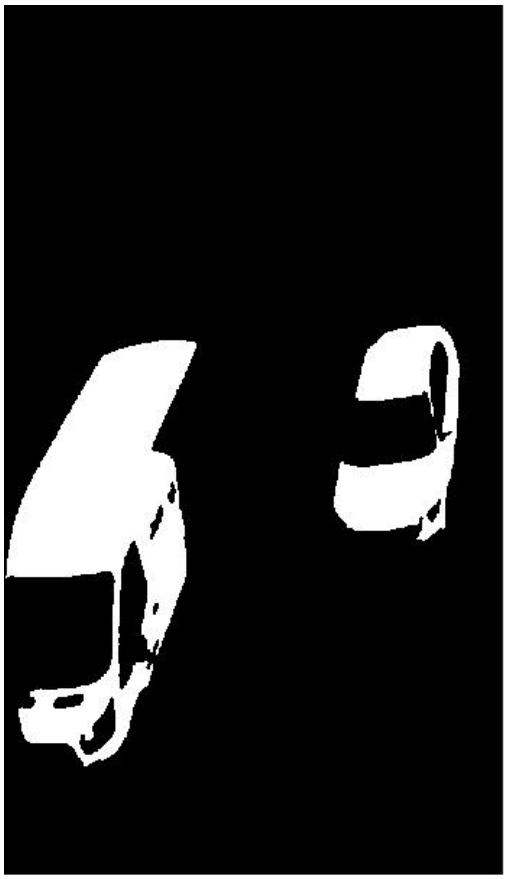
car_number=0;
It is an int variable. To use this variable later, we assign the number of cars we found on the picture.

car=imread('C:\location\name.jpg');
We assign the image variable to a variable named car. This allows us to process the picture.

car=imcrop(car,[300 175 520 1000]);
We trim the car variable with the parameters we specify. The sample version of the image is available in the description of the imcrop function above.

gray=rgb2gray(car);
We take the car variable from RGB to gray-scale space and assign our new image to the gray variable.

bw=im2bw(gray, 0.75);
We'll pass our picture from gray-scale to black-white with the rate "0.75" we've specified.



bw=medfilt2(bw, [3 3]);
It makes the image we're going to process better.

bw=bwareaopen(bw, 2000);
Clears the little pixels we don't want on the picture. As an example, there may be a person walking down the road, or there may be noises on the picture.

bw = bwlabel(bw, 8);
This command submerges pixels with "1" on the binary image under a single label. That way, we'll identify our objects on the picture.

cc=bwconncomp(bw, 8);
It assigns paramets to variables in it by taking certain pixel communities. In this way, we have a total of how many areas there are and how large these areas are.

stats = regionprops(bw, 'BoundingBox', 'Centroid','Extent');
With this function, we account for the center, endpoints, and area of the object on the picture.

imshow(car);
The imshow command will open the picture when the commands are finished.

hold on
The hold on command allows us to process the "car" variable. If we don't use this command, we can't take any action on the picture.

for object=1:length(stats)
We rotate the loop for each object, how many objects we have.

A=stats(object).Extent;
We're throwing objects into variable A.

bb = stats(object).BoundingBox;
We're throwing the ends of the objects into the bb variable.

bc = stats(object).Centroid;
We're throwing the centers of objects into bc variable.

The three commands above allow us to locate the object on the image and mark the object.

if (0.2<=A)&&(A<=1.8)
Compares the object to the field variable, and if the ratios are between "0.2" and "1.8", the processing begins.

car_number=car_number+1;
Once we're in the if cycle, we're increasing our counter.

rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
It inserts the object into the rectangle with the endpoints we have already determined. Other parameters determine the color of the edge, the shape of the drawing (dashed, flat, etc.), and the thickness of the lines.

a=text(bc(1)+15,bc(2), strcat('Car ',num2str(car_number)));
Allows us to overwrite the rectangles we receive on the picture.

set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
Determines the shape of the article we will write.

end
End of the if commend

end
End of the cycle.

hold off

We're shutting down the command we've opened because we won't take any further action on the picture.

As a result of the code we have written, we get a shape like the one above.

4 WORKING WITH INPUT VIDEO

Our foto detection technique is usefull for detection off miss component on circuit or detection of tumor in human body X-ray film but for our topic which is smart traffic system we need to detect moving objects on video so we applied our method in video.

4.1 INPUT VIDEO FORMAT

In this part we will look how can we get input video and we will see how can we track objects.

In the first step, we transfer the video we will examine as input to the matlab environment.

```
videoFileReader=vision.VideoFileReader('location\name.mp4');
```

So if input video is mp4 or mpeg-4 format we can use it as input with throwing into variable by vision.FileReader.

If input video is not mp4 some of matlab functions won't support our video. So there are some programs to convert also we can convert it from matlab. Code for transform video format is below took from mathworks. we won't explain it in detail because we concentrate on vehicle counting and tracking at this steps.

```
clc;
clear all;
close all;
% Browse Video File :
[ How_One_Driver_Can_Prevent_a_Traffic_Jam -
YouTube.mp4,C\Users\turhal\Desktop ] = uigetfile({'*.mov'});
if(video_file_path==0)
    return;
end
% Output path
output_image_path =
fullfile(video_file_path,[video_file_name(1:strfind(video_file_name,'.')-
1),'.mp4']);
% mkdir(output_image_path);
input_video_file = [video_file_path,video_file_name];
% Read Video
videoFReader = VideoReader(input_video_file);
% Write Video
videoFWrite = VideoWriter(output_image_path,'MPEG-4');
open(videoFWrite);
for count = 1:abs(videoFReader.Duration*videoFReader.FrameRate)
    disp(count);
    key_frame = read(videoFReader,count);
    writeVideo(videoFWrite,key_frame);
end
% Release video object
close(videoFReader);
close(videoFWrite);
```

4.2 VIDEO ANAYLYZE METHOD

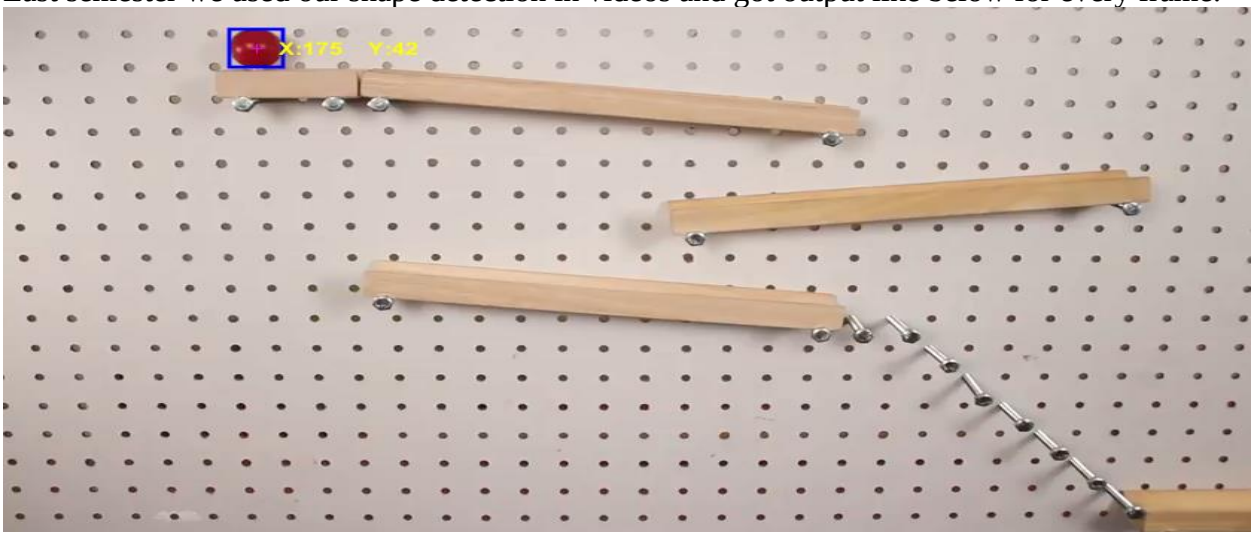
To examine moving objects in a video, we need to apply our separate operations in each frame. Therefore, we must first assign each frame to a variable and implement the operations. We combine the frame processed and get video again. To do this, we use the following methods.

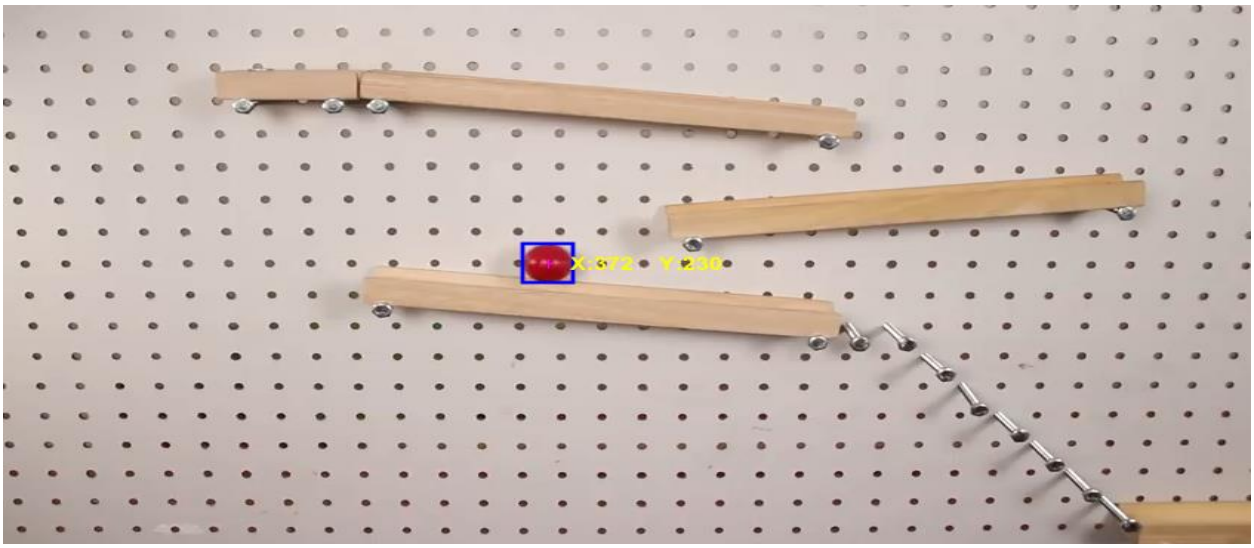
```
videoFrame      = step(videoFileReader);
```

After reviewing the frames individually, we create a blank video with the properties of the original video to create the video again.

```
videoInfo    = info(videoFileReader);
videoPlayer  = vision.VideoPlayer('Position',[300 300 videoInfo.VideoSize+30]);
```

Last semester we used our shape detection in videos and got output like below for every frame.





4.3 FINDING AND COUNTING VEHICLES IN VIDEO

In this section, we will transfer the actions we do through the image to the video environment. We've already told you what a lot of functions do. In this section, we will explain the codes for image processing in the video environment. The whole version of the codes is as follows.

```
clc; clear;
obj=VideoReader('C:\Users\cevozby\Desktop\ders\arabakısa2.mp4');
nFrames=obj.NumberOfFrames;

for frame=1:nFrames
   kare=0;
   thisFrame=read(obj,frame);
   araba=imcrop(thisFrame,[250 100 400 600]);
   gray=rgb2gray(araba);
   bw=im2bw(gray, 0.75);
   bw=medfilt2(bw, [3 3]);
   bw=bwareaopen(bw, 2000);
   bw = bwlabel(bw, 8);
   cc=bwconncomp(bw, 8);
   stats = regionprops(bw, 'BoundingBox', 'Centroid','Extent');
   imshow(bw);
   hold on
   for object=1:length(stats)
      A=stats(object).Extent;
      bb = stats(object).BoundingBox;
      bc = stats(object).Centroid;
      if (0.2<=A)&&(A<=1.8)
         kare=kare+1;
         rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
         a=text(bc(1)+15,bc(2), strcat('Car   ',num2str(kare)));
         set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
      end
   end
   hold off
end
```

```
clc; clear;
obj=VideoReader('video ');
```
The structure of the VideoReader command is similar to the structure of the imread command. We use this command to transfer the video to the drill media.

```
nFrames=obj.NumberOfFrames;
```
Videos are actually made up of many pictures. These pictures appear fluent when they come together. Here we're throwing the nFrames variable the number of pictures in the video.

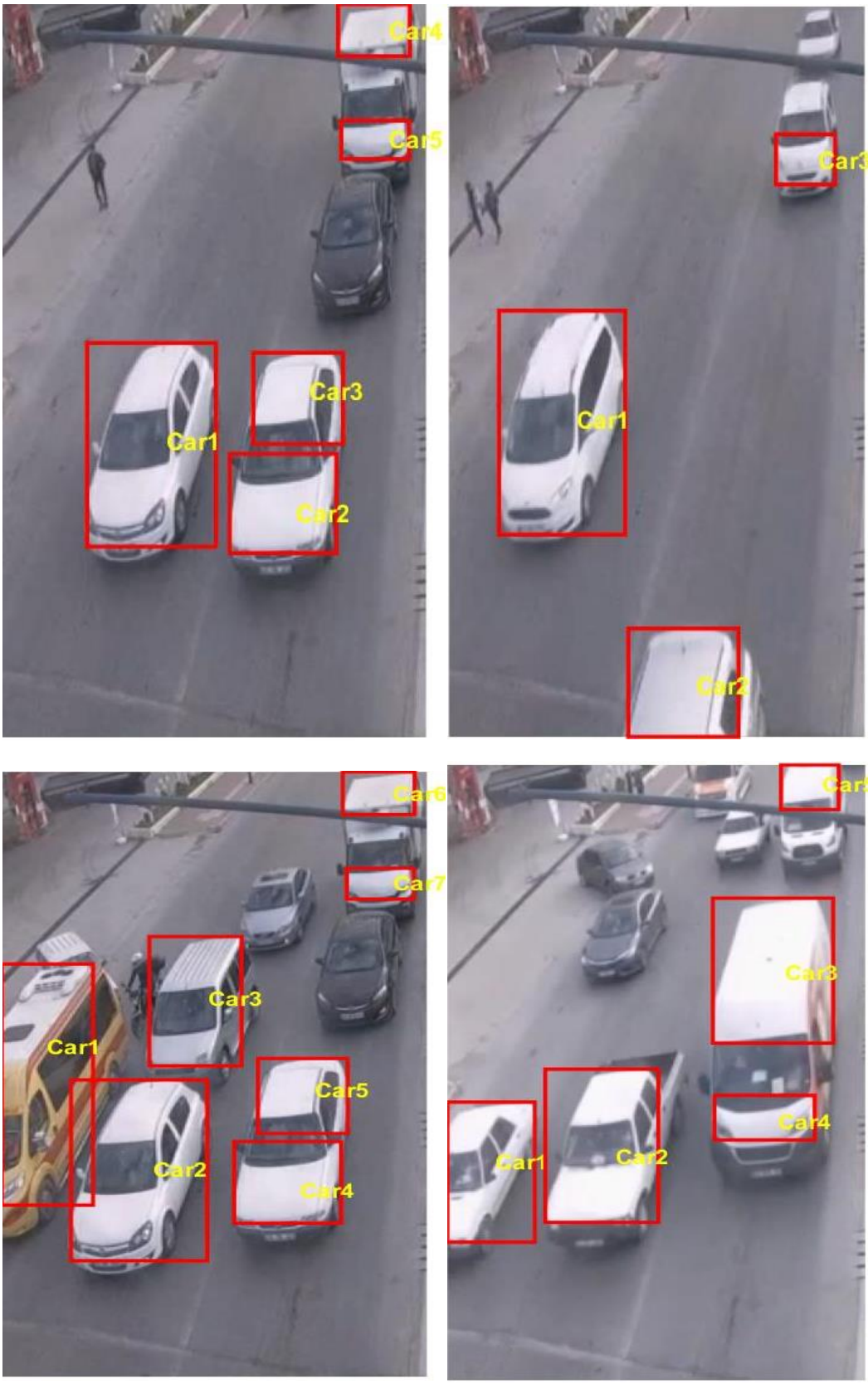
```
for frame=1:nFrames
```
This cycle will rotate throughout the entire number of pictures in the video.

```
   kare=0;
   thisFrame=read(obj,frame);
```

Whatever frame is the image in the video, the matlab takes the picture in that frame and processes it through that image.

```
   araba=imcrop(thisFrame,[250 100 400 600]);
   gray=rgb2gray(araba);
   bw=im2bw(gray, 0.75);
   bw=medfilt2(bw, [3 3]);
   bw=bwareaopen(bw, 2000);
   bw = bwlabel(bw, 8);
   cc=bwconncomp(bw, 8);
   stats = regionprops(bw, 'BoundingBox', 'Centroid','Extent');
   imshow(bw);
   hold on
   for object=1:length(stats)
      A=stats(object).Extent;
      bb = stats(object).BoundingBox;
      bc = stats(object).Centroid;
      if (0.2<=A)&&(A<=1.8)
         kare=kare+1;
         rectangle('Position',bb,'EdgeColor','r','LineWidth',2);
         a=text(bc(1)+15,bc(2), strcat('Car   ',num2str(kare)));
         set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
      end
   end
   hold off
end
```

The rest is the same as the car discovery part through the picture. We'll transfer the selected picture in the video to black-white space. We clear the picture noises in black-white space and identify the centers, endpoints, areas of the objects. Then we show the location of the objects for each picture. The following illustrations are available in the sample.


Car4
Car5
Car3
Car1
Car2
Car3
Car1
Car2
Car6
Car7
Car3
Car1
Car5
Car2
Car4
Car5
Car3
Car4
Car1
Car2

5 FINDING AND COUNTING VEHICLES IN THE VIDEO WITH MATLAB FUNCTIONS

In this section, we will use the MATLAB toolboxes to determine vehicle density. For this, we will use the foreground detector first. We will use foreground detector and blob analaysis on video to get background and binary image to separate vehicles. We will explain the functions used in this step in detail in the 6. section.

5.1 OUR CODE BASED ON MATLAB FUNCTIONS

Our full code at below we will explain each line after this.

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...
    'NumTrainingFrames', 50);

videoReader = vision.VideoFileReader('visiontraffic.avi');
for i = 1:150
    frame = step(videoReader);
    foreground = step(foregroundDetector, frame);
end
figure; imshow(frame); title('Video Frame');
figure; imshow(foreground); title('Foreground');
se = strel('square', 3);
filteredForeground = imopen(foreground, se);
figure; imshow(filteredForeground); title('Clean Foreground');
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 150);
bbox = step(blobAnalysis, filteredForeground);
result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');
numCars = size(bbox, 1);
result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
    'FontSize', 14);
figure; imshow(result); title('Detected Cars');
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
videoPlayer.Position(3:4) = [650,400];
se = strel('square', 3);

while ~isDone(videoReader)

    frame = step(videoReader);

    foreground = step(foregroundDetector, frame);

    filteredForeground = imopen(foreground, se);

    bbox = step(blobAnalysis, filteredForeground);

    result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');

    numCars = size(bbox, 1);
    result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
        'FontSize', 14);
```

```
    step(videoPlayer, result);
end

release(videoReader);
```

5.2 SEPARATE AND ANALYZE EACH FRAME OF THE VIDEO REASSEMBLING

In this part we explain all lines of code.

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...
    'NumTrainingFrames', 50);
```

In the first step, we try to get background using foreground detector. Simply we train the algorithm for 50 frames to decide where is background. We will explain Foreground Detector at next section in detail.

```
videoReader = vision.VideoFileReader('visiontraffic.avi');
```

We read the video to get it in MATLAB environment.

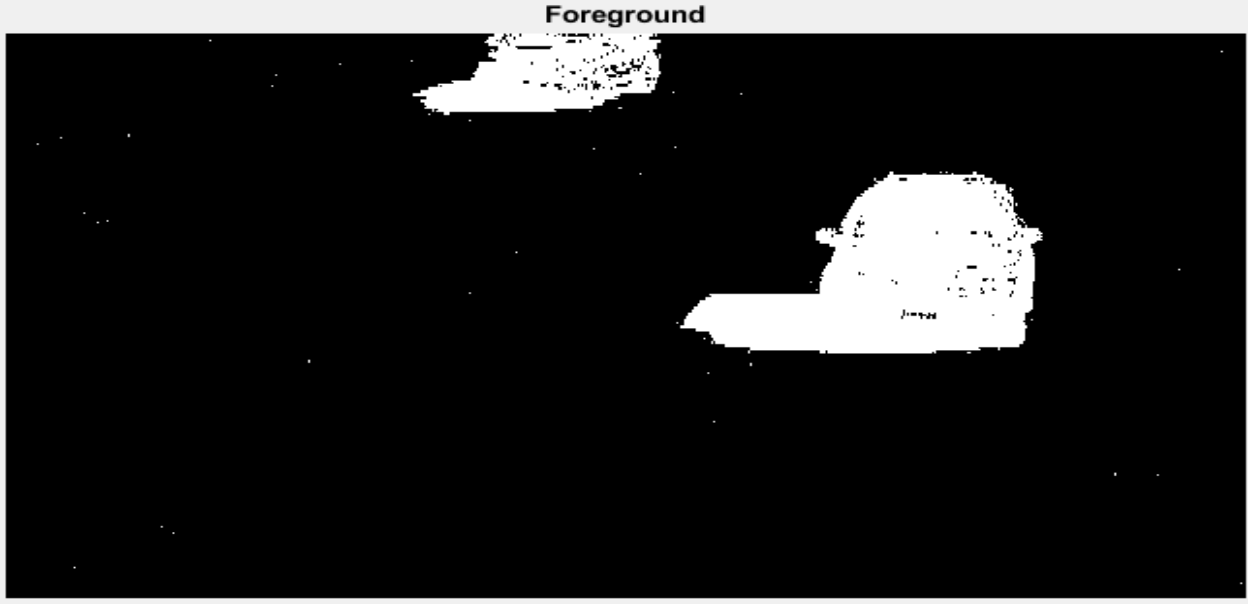
```
for i = 1:150
    frame = step(videoReader); % read the next video frame
    foreground = step(foregroundDetector, frame);
end
```

With this loop we are getting frames from video and training our foreground.

```
figure; imshow(frame); title('Video Frame');
```

We are showing the first frame of video. Output is at below.

```
figure; imshow(foreground); title('Foreground');
```
We see the first step and at loop we applied foreground detector to it so we get background and cars with binary image. We show it at figure. Output is at below.

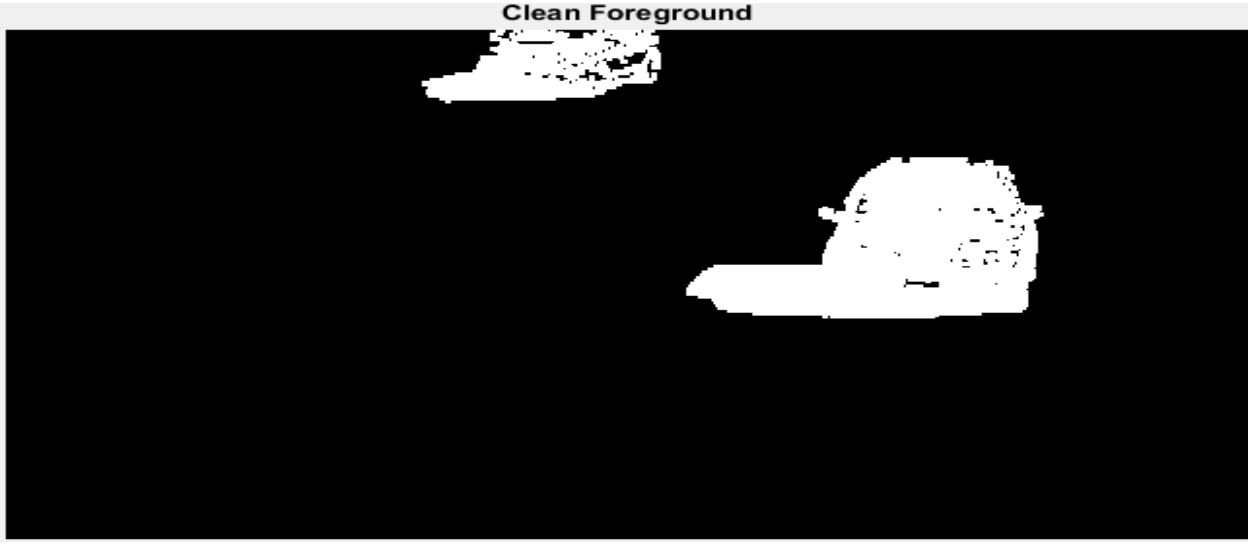


As we can see there are lots of noise. Which means white pixels here which are member of background so we apply filter to get cleaner output.

```
se = strel('square', 3);
```
This filter is using to clear noise from output. At this step we adjust value of filter. Value can be change according to the conditions of the environment where the traffic system will be used.

```
filteredForeground = imopen(foreground, se);
```
We apply the filter to our foreground and we get clean output.

```
figure; imshow(filteredForeground); title('Clean Foreground');
```
Showing clean output. Output is at below for first frame.



```
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', 150);
```
After using filter to clear noise, we are analyzing the frames to get white areas centroid by using "blob analysis" we will examine in detail it at 6. section.

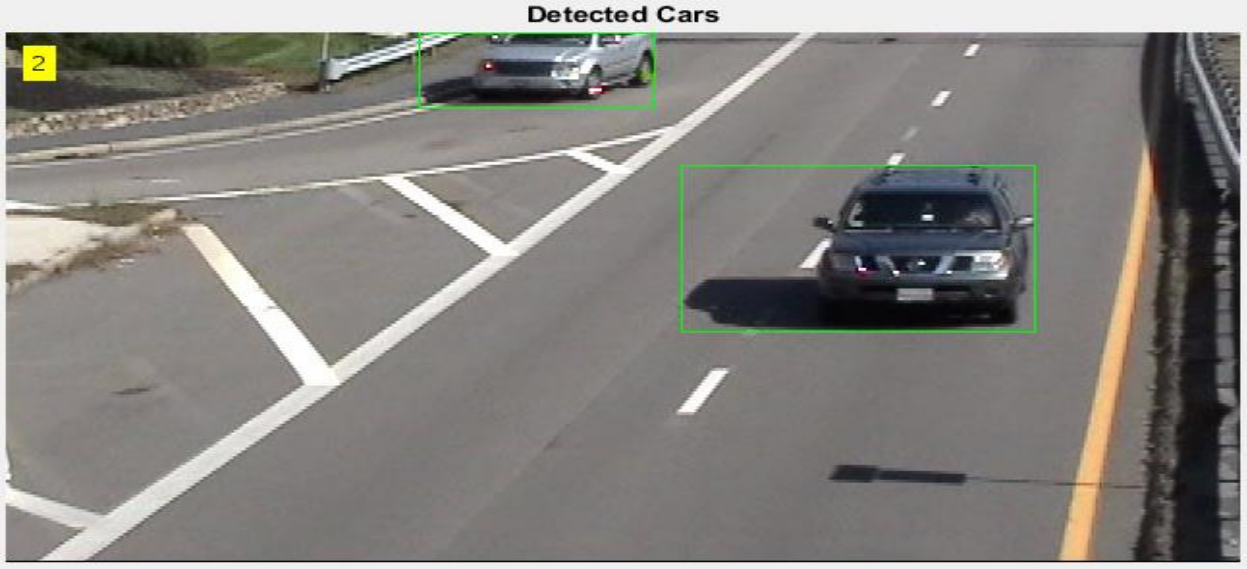
```
bbox = step(blobAnalysis, filteredForeground);
```
After blob analyze we get features from clean black white frame and save it in bbox variable to count and mark in next steps.

```
result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');
```
As we mention last step we are using bbox variable to mark white areas at frame which determines cars. We are getting white areas in box.

```
numCars = size(bbox, 1);
```
We are adjusting variable matrix width for up to bbox matrix size.

```
result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
    'FontSize', 14);
```
Writing number of vehicles in raleted frame. Lovation of text is starting from [10 10] pixels and font size is 14.

```
figure; imshow(result); title('Detected Cars');
```
After adjusting frame as clean we applied blob analysis to get vehicle locations and numbers than we mark vehicles in box and counted. After this we write number of cars as text on frame so for first step we show output. It is at below.

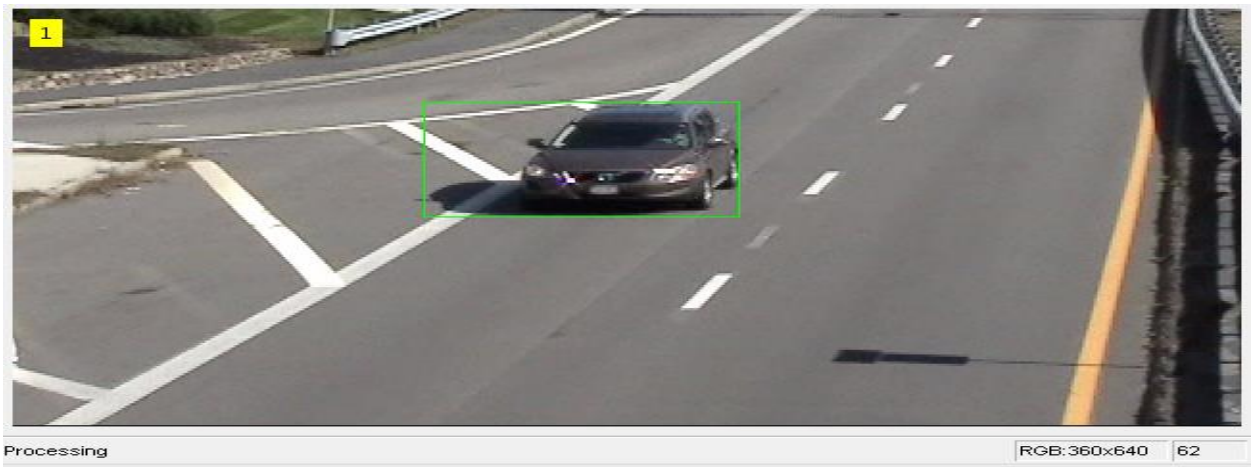


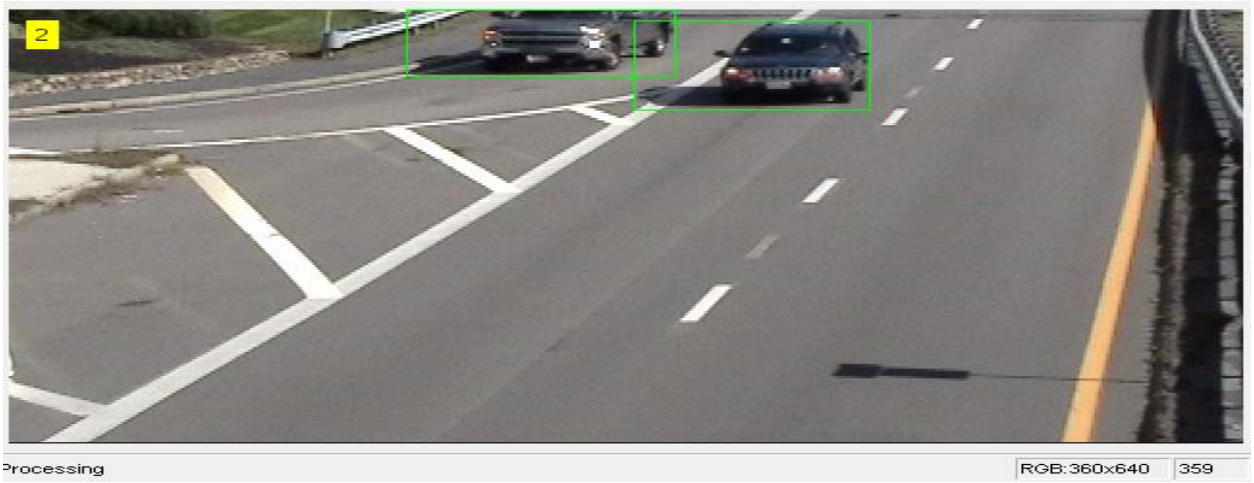
```
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
```
After all we have to put together every frame of video to get outout as video and play it.

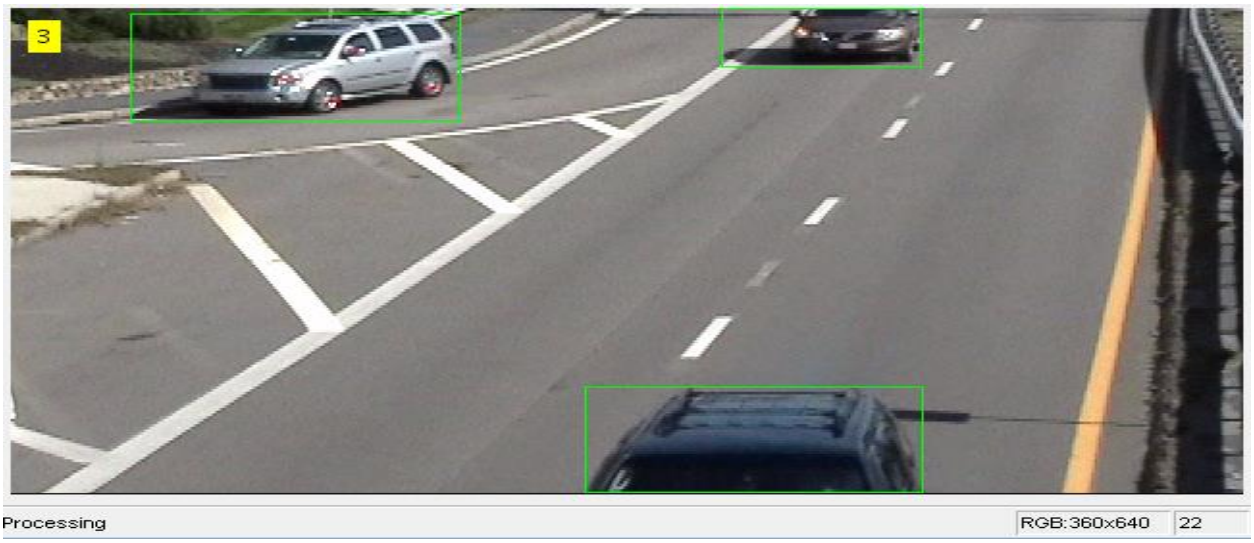
```
videoPlayer.Position(3:4) = [650,400];
```
Adjusting video window.

```
while ~isDone(videoReader)

    frame = step(videoReader);

    foreground = step(foregroundDetector, frame);

    filteredForeground = imopen(foreground, se);

    bbox = step(blobAnalysis, filteredForeground);

    result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');

    numCars = size(bbox, 1);
    result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
        'FontSize', 14);

    step(videoPlayer, result);
end

release(videoReader);
```

With this loop we analyze every frame until the end of video. Than we detect cars and mark them for every frame and count car numbers. After all we get output as video so we can see our results than after video is completed we close the video. We mentioned about video creating and releasing first part of the project. For different times at video there are some moments with different car numbers at next page.


1
Processing
RGB:360x640
62


2
Processing
RGB:360x640
359


3
Processing
RGB:360x640
22

6 FOREGROUND DETECTOR

The ForegroundDetector compares a color or grayscale video frame to a background model to determine whether individual pixels are part of the background or the foreground. It then computes a foreground mask. By using background subtraction, you can detect foreground objects in an image taken from a stationary camera.

6.1 GENERAL INFORMATIONS

Foreground detection aims to detect changes to the image. This method ensures better identification of foreground movements and objects by removing backgrounds.

In many applications, it is sufficient to determine changes in the foreground. You don't need every pixel data. For example, in our smart traffic system, we will put the camera in a fixed position, and the input image we will receive will be in the background, so we just need to get the information of the moving cars in the foreground.

All detection techniques are based on modelling the background of the image, i.e. set the background and detect which changes occur. Defining the background can be very difficult when it contains shapes, shadows, and moving objects. In defining the background it is assumed that the stationary objects could vary in color and intensity over time.

Scenarios where these techniques apply tend to be very diverse. There can be highly variable sequences, such as images with very different lighting, interiors, exteriors, quality, and noise. In addition to processing in real time, systems need to be able to adapt to these changes.

6.2 BACKGROUND SUBTRUCTION

Statik kameraların kullanılmasında arka plan sabit olacağı için bu çıkarma işlemi ile hareketli nesneleri yakalayabiliriz. Buradaki mantık arka planı bir model olarak düşünüp bu modelle karşılaştırılma yapılarak nesnelerin algılanması üzerinedir. Elbette yağmur gibi etkenler arka plan modelinin bozulmasına veya uygulanmasının zorlaşmasına neden olabilir.

6.3 TEMPORAL AVERAGE FILTER

This system estimates the background model from the median of all pixels of a number of previous images. The system uses a buffer with the pixel values of the last frames to update the median for each image.

To model the background, the system examines all images in a given time period called training time. At this time we only display images and will find the median, pixel by pixel, of all the plots in the background this time.

After the training period for each new frame, each pixel value is compared with the input value of funds previously calculated. If the input pixel is within a threshold, the pixel is considered to match the background model and its value is included in the pixbuf. Otherwise, if the value is outside this threshold pixel is classified as foreground, and not included in the buffer.

6.4 USING FRAME DIFFERENCING

A motion detection algorithm begins with the segmentation part where foreground or moving objects are segmented from the background. The simplest way to implement this is to take an image as background and take the frames obtained at the time t, denoted by I(t) to compare with the background image denoted by B. Here using simple arithmetic calculations, we can segment out the objects simply by using image subtraction technique of computer vision meaning for each pixels in I(t), take the pixel value denoted by P[I(t)] and subtract it with the corresponding pixels at the same position on the background image denoted as P[B].

6.5 RUNNING GAUSSIAN AVERAGE

We know pdf(probabilistic density function) = $ae^{-\frac{(x-b)^2}{2c^2}}$. The pdf of every pixel is characterized by mean μ and varianceσ^2 with t. With using this model we can estimate background we won't examine it with mathematically.

6.8 BACKGROUND MIXTURE MODELS

Mixture of Gaussians method approaches by modelling each pixel as a mixture of Gaussians and uses an on-line approximation to update the model. In this technique, it is assumed that every pixel's intensity values in the video can be modeled using a Gaussian mixture model.[6] A simple heuristic determines which intensities are most probably of the background. Then the pixels which do not match to these are called the foreground pixels. Foreground pixels are grouped using 2D connected component analysis.

7 GAUSSIAN MIXTURE MODEL

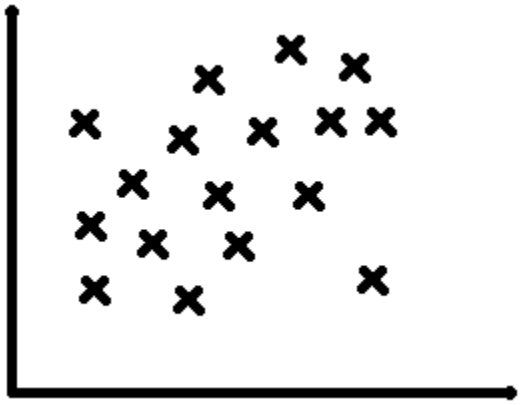
It consists of two segmentations. GMM is a clustering technique with small changes.

Unsupervised: If the data points in the examined image are divided into 4 different areas, they will find them.

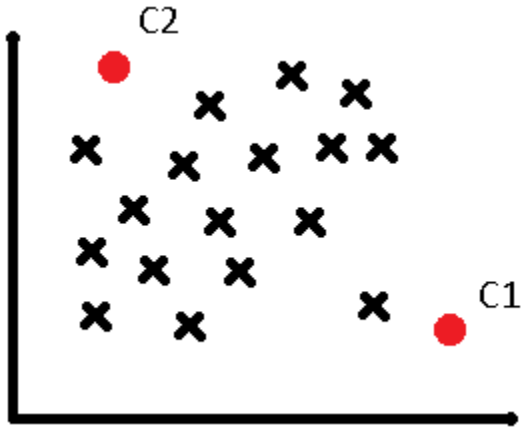
Supervised: The location contains data with the correct values. This data is taught to the algorithm using it and used to segmation in the future.

K-mean:

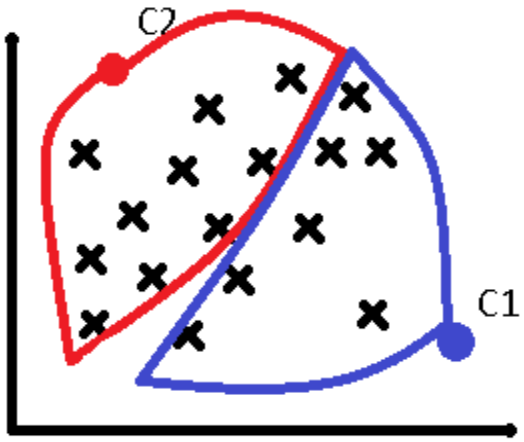
Let's think of a picture with pixel values at some point. Get the chart as in the following figure:



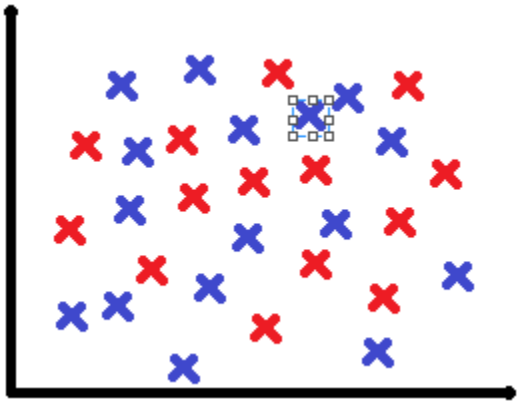
If we want to divide the data points here into 2 sets, for example, if we reference the C1 and C2 center points:



Here, clustering is performed according to the distances of each data points to C1 and C2. Each point is clustered to the nearest appetizer. This will be segmentation with C1 and C2 values.

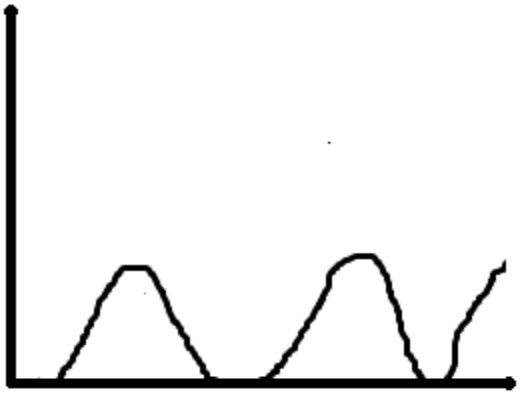


The average values of blue and red fields are calculated here. Data close to the average value in the blue field is accepted in the blue area, and data close to the average value in the red field is accepted in the red area. K-mean can be used to separate such data points, but using k-mean in pictures with different colors and pixel values variable, such as the tool, does not work correctly because data points in the red and blue area are intertwined.



Segmentation of such mixed data points is the easiest with GMM. We'll continue through histogram to better understand GMM.

If we have a discrete picture of data points, our histogram chart is similar to the way it is.



In such cases, it can be easily segmented using k-mean.

But when there's a histogram like the one below, we can't use k-mean because we have a graph that doesn't fall to 0, so the data nodes aren't discrete.

When we look at the chart, we can see 4 different areas, although there are no discrete data, there are different regions in the region. In such cases, we need to use the Gaussian Mixture Model. GMM takes different gaussians from histogram. The whole histogram consists of a combination of several gaussias. For example, let's say we have gaussians with four red lines like this:

Here, our main graphic with Black is a combination of 4 gaussia slated for red. Each gaussia has its air width height and weight.

GMM is a probability model that assumes that the data consists of separate combinations of gaussians.

Each data is likely to be in different regions. For example, a data point can be at a 98% chance c1 and 2% at C2. Each data point is weighted with average and weight update. This is called the expectation maximization method. The expectation is to calculate the probability at each data point and update the variant and weight of the maximization average.

So if we're working on a picture with non-discreconsumical data points. We'll be segmented using GMM to remove the background. We will explain this process in more detail in the future.

8 EXPECTATION-MAXIMIZATION(EM)

This algorithm used in statistics is a repeated method for finding the maximum probability of parameters or maximum information and estimates in statistical models where the model depends on hidden variables that we cannot observe. EM iteration varies between performing an expectation step that creates a function for the log-probability expectation evaluated using the current estimate for parameters and a maximization step that calculates the expected log-maximized parameters. We use these parameter estimates to determine the hidden variable distribution. To put it simply, it's an example: We'll have a trick yonocity, but we don't know the odds. Once we've thrown this money 10 times, we'll get some data. With this data, we determine the possibilities of coin and coin. Then let's do this 10 more throws 10 more times. Let's compare the data we get in every 10 shooting models with the previous data. At the end of the 10th round, we will have the possibility of a near-truth possibility. The more we do this experiment, the closer we get to the truth. The formula is;

$$\binom{10}{k}\theta_i^k(1-\theta_i)^{10-k}$$

$$i \in \{A, B\}$$

9 SOME IMPORTANT FUNCTIONS

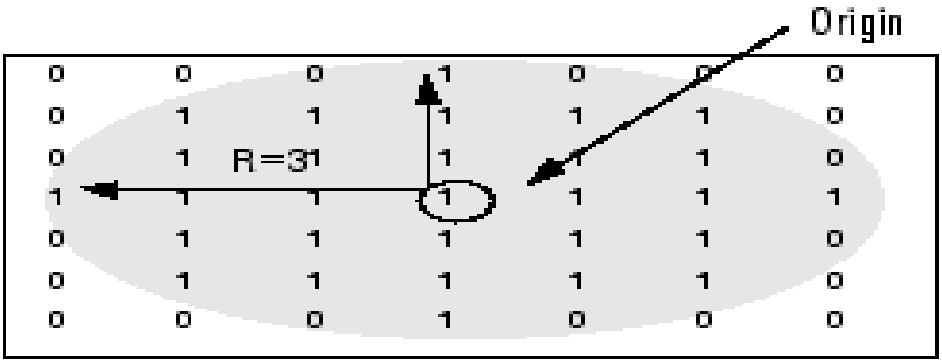
At this section we will explain used functions in matlab to detection and counting of cars.

9.1 STREL()

strel();

A strel object represents a flat morphological *structuring element*, which is an essential part of morphological dilation and erosion operations.

A flat structuring element is a binary valued neighborhood, either 2-D or multidimensional, in which the true pixels are included in the morphological computation, and the false pixels are not. The center pixel of the structuring element, called the *origin*, identifies the pixel in the image being processed. Use the strel function (described below) to create a flat structuring element. You can use flat structuring elements with both binary and grayscale images. The following figure illustrates a flat structuring element.



9.2 VISION.BLOBANALYSIS()

To compute statistics for connected regions in a binary image. Setting the desired threshold value to standard and converting a gray image to values 0 and 1 are called "binarization process" (black 0 and white is 1). Using this process, it is possible to focus only on the control object and perform various analyses.

The method of analyzing an image that has gone through the ikilization process is called "blob understanding". A drop means a piece. "Blob analysis" is the most basic method for analyzing the shape properties of an object, such as image processing, the presence, number, field, location, length, and direction of the parts.

Example of using:

Create the blob analysis object.

hBlob = vision.BlobAnalysis('AreaOutputPort',false,'BoundingBoxOutputPort',false);

Create the blob.

```
img = logical([0 0 0 0 0 0; ...
               0 1 1 1 1 0; ...
               0 1 1 1 1 0; ...
               0 1 1 1 1 0; ...
               0 0 0 0 0 0]);
```

Find the coordinates for the centroid.

centroid = hBlob(img);

Example 2: When we use blob analysis we get binary image as output from gray image.

10 CONCLUSION

At this project we aim to create smart traffic system. Firstly, we worked on MATLAB environment image reading, color detection, object detection at images. Secondly, we worked on object detection in video inputs because our traffic system will get video input from camera. We firstly worked on different shapes and at first semester we get basics of image processing algorithms.

This semester we worked on vehicle recognition techniques in video input. We aimed getting input from camera and adjusting traffic lights up to vehicle numbers but we finally finished vehicle recognition in video by using GMM and foreground detector. These operators have different variable values which are changing due to tilt of vehicle way, bright and weather. So, these codes work on specific highways. That’s all we come because of extraordinary situations(covid-19).

11 REFERANCES

https://www.mathworks.com/

Digital Image Processing 3rd ed. - R. Gonzalez, R. Woods

https://en.wikipedia.org/wiki/Foreground_detection

https://www.mathworks.com/help/vision/ref/vision.foregrounddetector-system-object.html

https://www.mathworks.com/help/vision/examples/detecting-cars-using-gaussian-mixture-models.html?prodcode=VP&language=en