‘Gheorghe Asachi’ Technical University of Iasi

Faculty of Electronics, Telecommunications and Information Technology

Vehicle detection and movement estimation in video sequences

COORDINATOR

Dr. Eng. Nicolae Cleju

STUDENT

Cătălina Raluca Bularda

2018

TABLE OF CONTENTS

INTRODUCTION

Movement detection and estimation in videos has been analyzed for many years now, different techniques being already developed in this direction. Nowadays, image processing is considered to be a middle area between computer vision and signal processing which makes it more interesting to be studied.

This field has grown so fast in the last period, interesting applications being already implemented in various systems. Some nice examples are applications like pedestrian recognition, license plate recognition and so many others. Computer vision applications became more and more important, especially considering the future trends, where everything is wanted to be autonomous. It is clear that the possibility to make machines to take decisions by itself without human intervention is closer to reality now.

My determination to study this area, started from the curiosity of finding how all these work, how computers can develop such a precise vision and how they can react and take decisions. By making this application, for my final project, I have understood the basic concepts of the signal processing field.

Considering all of these, the main studied subject for this project was the detection and estimation of vehicle motion in a video sequence, by analyzing and extracting information from each frame of a video.

In this paper, it will be described an algorithm which is used for object detection, motion estimation and vehicle counting in a video sequence. I focused my attention to study different techniques for motion estimation which can be used to detect the vehicle direction on a particular street. Obviously, the algorithm can have multiple uses; it can definitely work in different applications, depending on the needs.

The project is structured in two chapters.

In the first chapter the theoretical aspects based on which the application has been developed will be described. There are presented information about video and image processing, how the images are acquired and the digitalization process. Different motion estimation techniques used to estimate the object direction will also be described in this chapter.

In the second chapter it is explained in detail the implemented algorithm and the conditions in which it will work. Some basic block schemes are drawn for a better understanding of the process and different figures are shown to illustrate the simulation results. Of course this algorithm will work in limited situations.

In the future, everyone must be capable to understand how these systems work, how computers perceive the environment and how they process and analyze the information. This will simplify our understanding in their behavior and will contribute to a more sustainable future.

CHAPTER 1: FUNDAMENTALS OF COMPUTER VISION AND IMAGE PROCESSING

Computer vision is quite an interesting field and it starts to gain more and more importance in the Artificial Intelligence area. There are a large group of algorithms that can be implemented to give to a computer the ability to understand and to perceive objects found in real life. The machines are now capable to process information at very high speeds and they are able to react in real life situations. The idea is to know how things work and how to achieve visualization as close as possible to the human perception, taking the right decision in each situation.

Nowadays, these systems work well. Object recognition systems have already been developed in some applications which are part of our daily routine for a few years now. Application like face recognition for pedestrians has been successfully implemented in some countries, to see when someone breaks the law and some measures were established contributing to a consistent decrease in the road accidents.

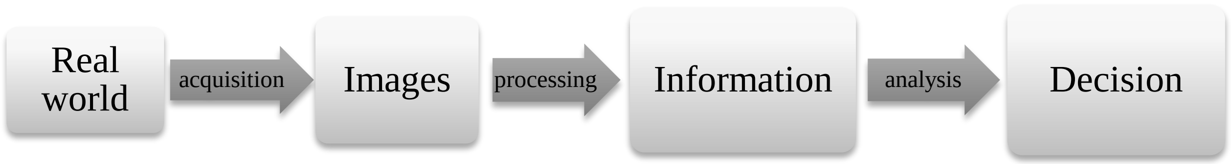


Figure 1.1: Processing a signal

This is a significant area in the scientific field which deals with development of systems which can perform different actions in accordance with the input processed data. Various algorithms are developed to process the images or a sequence of images in form of a video, to extract the useful data and to perform different tasks based on the interpreted information.

It includes a lot of image manipulation techniques and different algorithms that can help to achieve the desired results. The rate at the information bits are transmitted is much higher now than in the past, thus image processing has gained a high place in fields like medical area, automotive, military and many others. The developed tools are used for video acquisition, image enhancement, and segmentation and now, object recognition is easier to be analyzed. This leads to more accurate results, to high speed data transmissions and to a considerable increase in the real time system applications.

1.1 VIDEO AND IMAGE PROCESSING

A video is formed by a number of images moving with a certain rate. As the number of frames is increased, the movie formed will look more realistic. In the figure presented below, the process of how images are captured by the camera is presented.

Image acquisition consists of the image being captured and stored using a camera in order to process it for the use in further applications. This process can be divided in three steps:

- The light reflected by the object of interest measured as the amount of energy
- This energy must be focused by an optical system
- The conversion of light into electrical signals is done by a sensor

The hardware part of the camera contains more sensors which will sense the light and will convert it into electrical signals. Each camera has an analog to digital converter which will further convert the voltages in different values and will be stored in binary form. For example, for camera with an 8 bit ADC, the pixels intensities values will be between 0 and 255.

Camera principles work exactly like the human eye. There are different sensors in the camera, placed to measure the amount of light incoming. The light wave is then splatted in 3 wavelengths using filters. A color image is formed by 3 pixels, each of it describing the amount of red, green and blue.

The light is a physical signal and like any measurable quantity it can be described by a two-dimensional signal containing useful information. However, more complex circuits need to be used to process that information and to store it properly, without any data loss, which can be critical in real-time operations. As source of light, the sun is considered, and the desire object to be observed is a vehicle. What a camera does to capture the picture it is described in the following paragraphs.

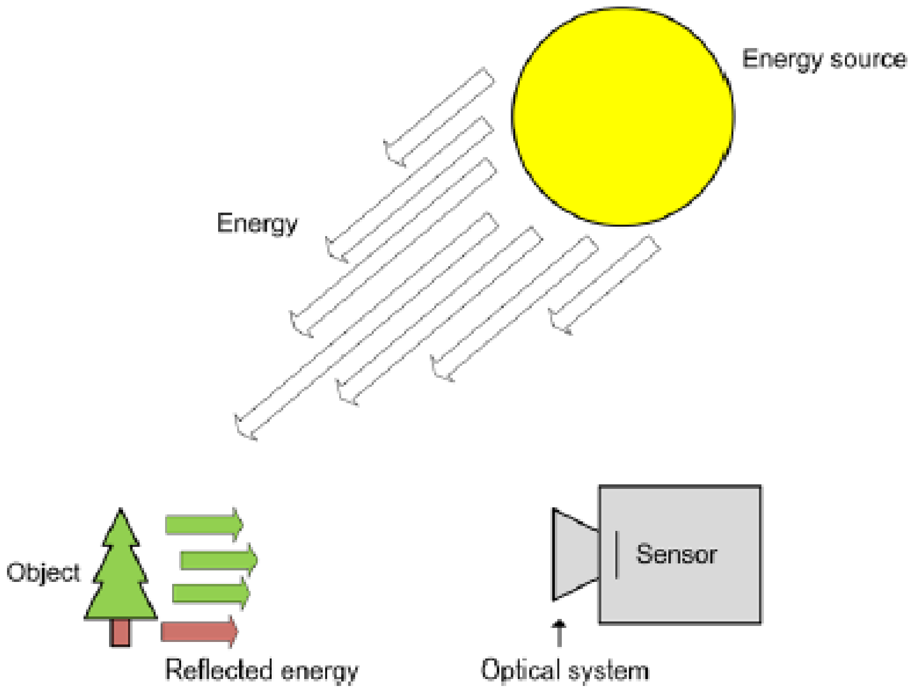


Figure 1.2: Image formation[1]

Photons are reflected from the object into the camera lens. Behind that lens, there is a sensor which sense the amount of energy coming and it converts it into electrical signal. The amount of the voltage generated is proportional with the quantity of light coming in. All this data is converted into a digital form, represented by a matrix with different values, these values representing the pixels intensities. These numbers are stored in the CCD array, in forma of a 2D matrix, representing the complete image being the result of sampling and quantization processes.

[1] Image from https://www.springer.com/cda/content/document/cda.../9781447173182-c2.pdf

Digital image processing

An image can be represented as a two variables function, a function which gives the brightness of the pixel at the specified coordinates, x and y. It is a continuous function, but after passing through the sampling and quantization processes, it will be stored as a matrix which contains integer values. The pixel, also called the representative element of an image, gives the amount of energy received by the camera sensor in a certain point at a moment in time.

How a pixel is formed plays an important role in image creation, and implicitly in a video making process. Let us look on how an image is composed. A video is set up by having more pictures moving one after another with a specific frequency, showing how many frames run in a second. The quality of the video depends a lot on the frequency used and on the image resolution.

At first, acquisition is done by translating the real word captured on videos or images in a digitized form. To do that, the physical signals represented as analog signals must be converted to digital ones using different electronic circuits.

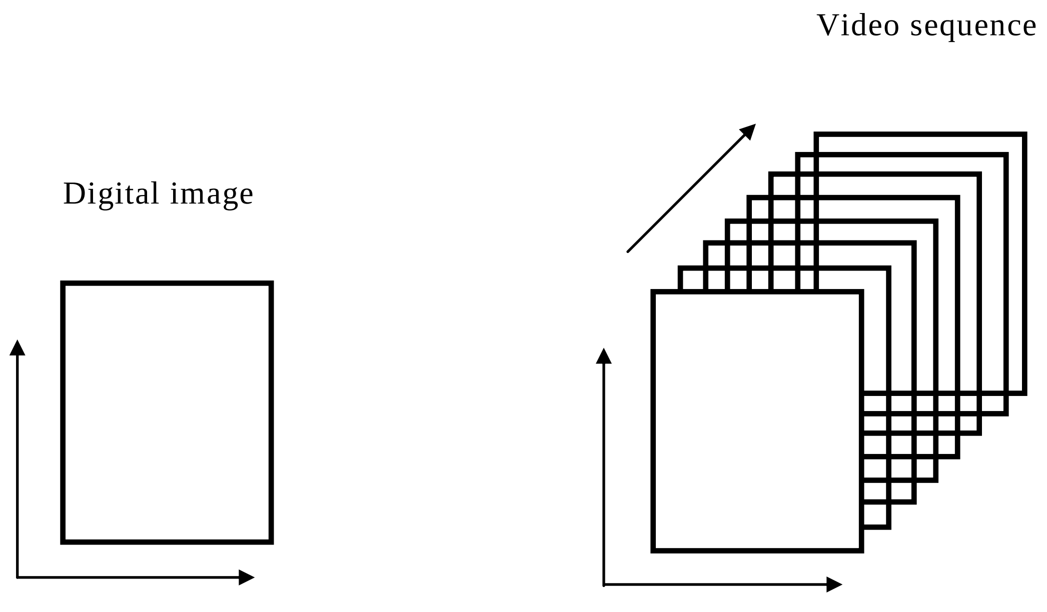


Figure 1.3: Video recording

The second step assumes all types of image processing methods. This includes the use of different well-known algorithms like segmentation, smoothing operations, edge detection which can be applied to the captured images to find and to highlight the regions of interest.

Analog signals are represented by any signal which contains information. These are continuous signals and they are variable in time. Some common examples of analog signals are voice and images. Due to the infinite memory these kind of signals require to be stored, they will be converted in discrete signal. In this form they are easier to be stored and to be analyzed.

I have presented some information about how the digitalization of the image is done, namely through sampling and quantization. Let us see a basic description of what sampling of images exactly means.

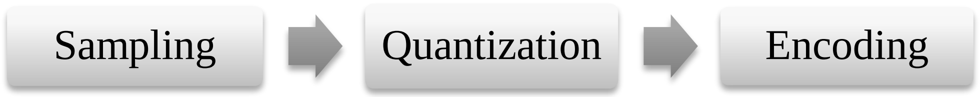


Figure 1.4: Image digitization

The sampling process is one of the most important in image processing.

Because the image comes as a continuous signal, we need to process it by meaning of a digitalization process ant to store the discrete samples of the signal. The concept of sampling still gives us some issues, the sampling rate playing a major role in the reconstruction of the original signal.

One problem that can appear if the sampling rate is incorrectly chosen can be the phenomenon of aliasing. What aliasing means, I will try to show and explain in the following. Basically, the reconstructed wave does not look like the original one, some details of the image are lost, and this is because the sampling frequency is too low.

The Sampling Theorem states that the recommended sampling frequency needs to be twice higher than the maximum signal frequency to obtain a signal close to the original one and to avoid aliasing.

Figure showing a signal in its original form and after sampling:

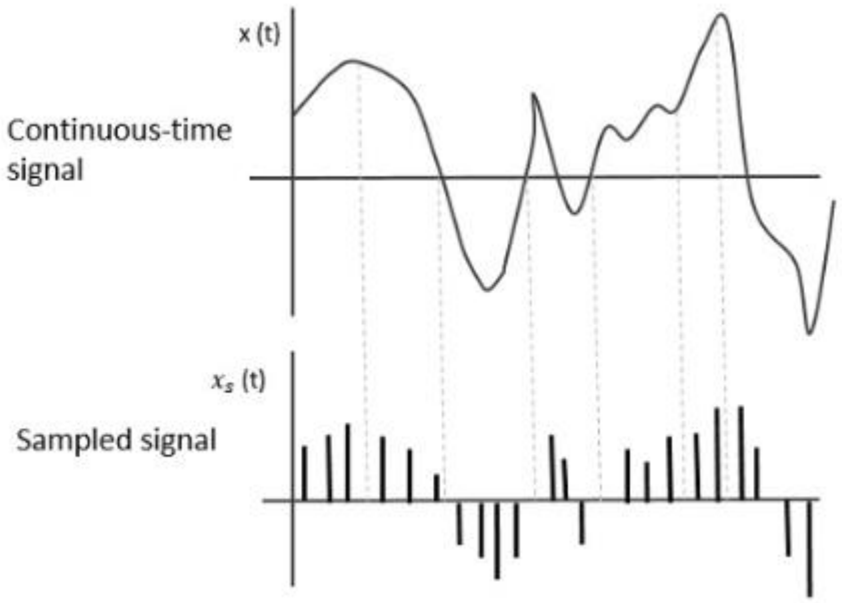


Figure 1.5: Original and sampled signal[2]

In addition to the sampling process, quantization is the following step to be done. A threshold is set to convert the lightness of the pixel at each sample in a discrete value. The quantization levels have a significant importance, because depending on this value, many details can be lost in the reconstructed image, especially if a low number of levels is chosen. So, threshold values must be properly set to obtain good results.

Sampling is used to obtain a discrete signal by taking samples at different points in time of a continuous signal, it is a very important technique used in digital signal processing. Followed by quantization process, in which are established some intervals, to get a specific value. Hence, the amplitude signal is divided into more levels, and a binary value is assigned to each voltage level,

[2] Image from https://www.tutorialspoint.com/digital_communication/digital_communication_sampling.htm

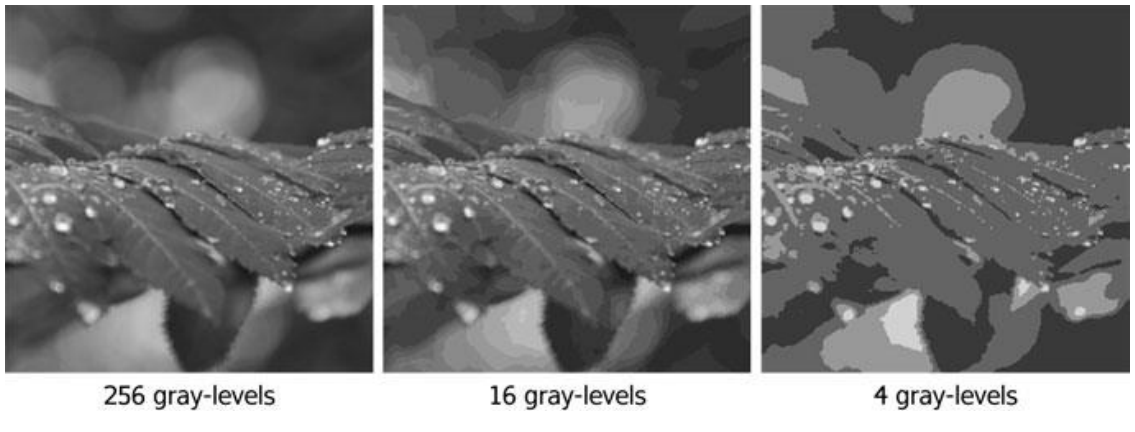
representing the encoding process. Getting a signal through these processes can lead to loss of data, giving a wrong conversion. To avoid this, multiple solutions can be developed. For example, in the quantization processes, as much levels of quantization are used as accurate the signal obtained is and the error is reduced.

Resolution of an image can be described in many ways. Two used terms are pixel resolution and spatial resolution. Usually the resolution of an image can be described in pixels per inch, showing how many pixels are in an inch within an image. Basically, the resolution tells us how many pixels an image contains. Obviously, the image quality will increase with the increase of number of pixels.

An image can be processed in different ways; multiple filters can be applied to it to increase its quality, to reduce noise and to highlight the points of interest. A high quality means a high number of pixels, which also means a high resolution. Manipulating a high-resolution image takes time, which can be critical in systems working with real time data, the processor taking too much time to make all the mathematical operations, resulting to an ineffective work.

The spatial resolution is described by the number of pixels describing the image, meaning that if we have a high resolution, more details can be seen, while decreasing the resolution, less pixels will be used to form the image and this will result in a low quality image without many details

Size of image is given by the image number of pixels multiplied with the number of bits used for each pixel representation. In the next figure[3] it is shown how the image loos like depending on the quantization. We can observe that the best quality was achieved when 8 bits per pixel were used.



[3] Image from https://www.springer.com/cda/content/document/cda.../9781447173182-c2.pdf

An image is like a matrix formed by 3D vectors in which each value represents the amount of Green, Blue and Red sensed by the sensor. In image processing it is difficult to manage the pictures in this form, that is why a helpful trick is to convert the images to black and white and then to manipulate them in this form. This reduces the operations needs to be done to obtain the desired results.

The grayscale conversion of a color image can be made using two methods. The simplest way to find the corresponding value of the pixel in gray level is to use the averaging method. This method implies that the value resulted after conversion is the average of the three colors, red, green and blue.

The quality of the grayscale image obtained after the conversion will not be very good, taking into consideration that the colors have different wavelength and their contribution is not equal in an image formation. For that reason, another method is preferred to be used, namely the weighted or luminosity method. In this method the grayscale image will be calculated depending on each color weight.

A good understanding of pixels is important, how they are represented as an element of an image. In a bit scale, the value of the pixel can be between 0 and 255, and it represents how many photons have stroked that point at a certain time. The image dimension is given by the number of pixels of the rows and columns.

Color models in image processing

An image can be represented as a two variables function, a function which gives the brightness of the pixel at the specified coordinates, x and y. It is a continuous function, but after passing through the sampling and quantization processes, it will be stored as a matrix which contains integer values. The pixel, also called the representative element of an image, gives the amount of energy received by the camera sensor in a certain point at a moment in time.

There are basically three quantities which describe the quality of light, respectively the radiance, the luminance and the brightness. Radiance is the total amount of energy which comes out of a light source. Luminance is the amount of energy perceived by the observer and brightness gives a feeling of light intensity.

The fundamentals of colors are based on some characteristics observed many years ago by the Isaac Newton. The next paragraph, taken from *Digital Image Processing* book about image processing will describe better the concept.

'In 1666, Sir Isaac Newton discovered that when a beam of sunlight passes through a glass prism, the emerging beam of light is not white but consists instead of a continuous spectrum of colors ranging from violet at one end to red at the other. As the next figure shows, the color spectrum may be divided into six broad regions: violet, blue, green, yellow, orange and red. When viewed in full color, no color in the spectrum ends abruptly, but rather each color blends smoothly into the next.'[4]

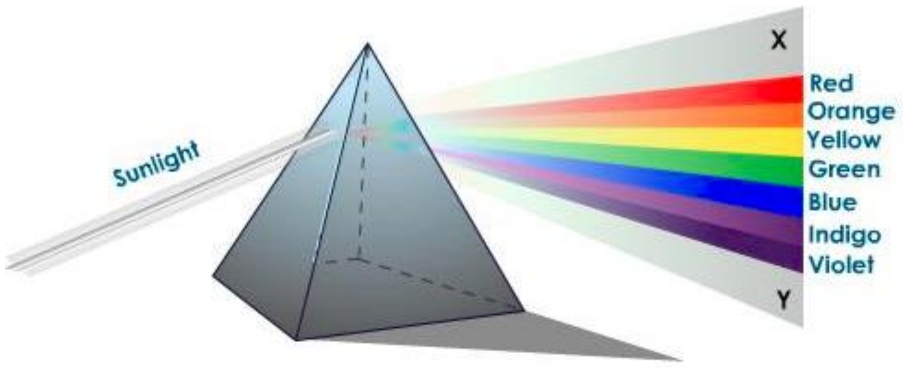


Figure 1.6: Glass prism[5]

Color image processing is an important subject, due to the fact that some object recognition is easier to be done if the images are described by RGB colors.

There are two ways to describe color processing; the closest to the human perception is so called pseudo color processing method. Pseudo color processing states that the gray levels with the range between 0 and 255 are further subdivided into more ranges and for each subdivision a particular color is assigned.

We consider the primary colors to be red, green and blue and we assume that mixing these colors in different portion, we can obtain the other color present in the spectrum.

[4] Rafael C. Gonzales, Richard E. Woods, *Digital Image Processing(2nd Edition),* New Jersey, page 283
[5] Image from http://catalog.sciencekitstore.com/prisms

When we look at the image, we don’t really think at how much of green component or how much of blue component that particular color has but rather we describe three color characteristics called brightness, hue and saturation.

Brightness is nothing but a chromatic motion of intensity, hue represents the dominant wavelength present in a mixture of colors and saturation indicates what the purity of that particular color is. Until now we have presented these three concepts of color perception. For an easier use in the application developments they are divided into two parts; the brightness and the chromaticity represented by the hue and saturation.

1.2 MOTION ESTIMATION TECHNIQUES

One important step in analyzing video sequences is the process of finding the motion vectors in consecutive frames, to determine in which direction an object moves. There are different methods that can be used among which the Block matching technique which is based on region matching and Optical flow method, based on gradient based method.

Block - Matching technique

This method is considered to be one of the most efficient methods for motion estimation due to its simplicity of implementation. It searches the displacement of the pixels by using block of pixels, and it does not calculate the motion vector for each pixel.

It is indeed interesting that although the approach is so intuitive, block matching is still an active research and development topic in the field.

By using this method to estimate the direction, the output will be a vector which will describe the horizontal and vertical displacement of the block we are looking for. The motion vector magnitude and phase will determine the direction and moving speed of the object between frames.

Having a video sequence, formed by consecutive frames, the point is to find out how a detected object has moved from one frame to another. Firstly, each frame is divided into blocks of pixels of a certain size. This method suggests to take a block of pixels from a frame and to find the closest matching in the previous frame. Then the motion vector will be computed as the difference of the two blocks spatial coordinates. Depending if the objects in the video moves faster or not, the dimensions of the block will vary. We must keep in mind that if a larger block will be chosen, the mathematical operations will take more time.

This algorithm has two methods by which the search of the blocks is made. One is the Exhaustive method ant the other is called Three step method. The difference between these two is that using the first one, the block will search pixel by pixel leading to an inefficient process, while the second one will change that block in a predefined window. Based on what I found in that region, it will continue with searching.

Optical flow method

The optical flow technique is used to estimate the direction of an object based on each pixel variations from one frame to another taking in consideration also the changes in the neighborhood pixels.

The idea is that we want to know where each pixel from the current frame goes in the next frame. The spatial coordinates of the pixel in frame two will be the coordinates of the same pixel in frame one plus its horizontal and vertical displacements. The motion vector is computed assuming that the pixel intensity will not change during its movement.

This method for motion estimation uses searching at pixel level. Actually, it assumes that during the object motion, the pixels intensities will not vary on its trajectory. The motion between two consecutive frames will be calculated by considering all the pixels variations, on horizontal and also vertical planes.

CHAPTER 2: APPLICATION DEVELOPED FOR VEHICLE MOTION ESTIMATION AND COUNTING

For this project I have chosen to develop an algorithm which will have as main scope the detection of the vehicles passing through a certain area, to observe the direction they move and to count them. This algorithm can be used in multiple applications regarding computer vision field.

The applications could be used in a number of different scenarios. One could be to detect the cars entering and leaving a parking lot and another can be the detection of the cars going in the opposite direction on a one-way street. For the first application, regarding the parking lot I will have an illustrative video in the presentation showing how this work.

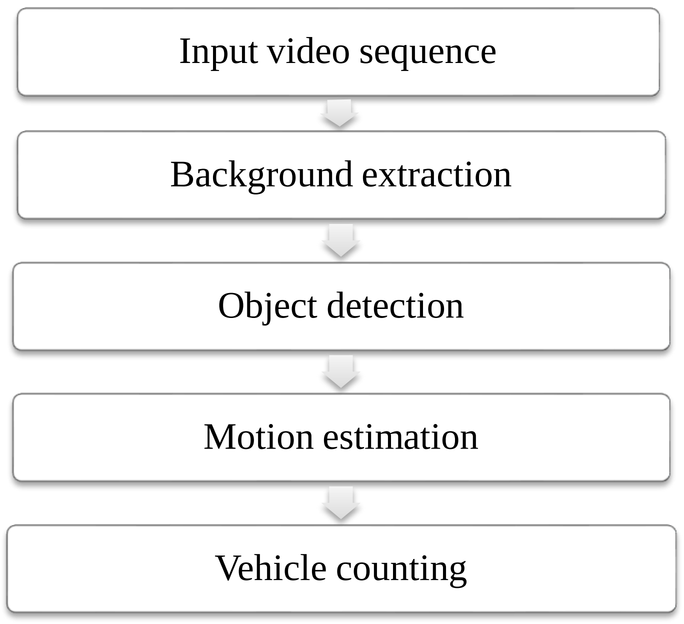


Figure 2.1: Application flow diagram

For the second application, if we think that drivers don't entirely respect the traffic rules, I have thought that such a system, if implemented correctly would contribute to a safe driving. Maybe, a further development of a system to be able to recognize also the license plate will be a major improvement of the whole idea. In such a way, those who disobey the traffic rules can be more easily detected.

2.1 APPLICATION DESCRIPTION

Three main important objectives need to be accomplished by the application:

- Detect the moving vehicles
- Estimate the movement direction
- Count the vehicles

These being said, in the following I will describe each step which contributes to an effective algorithm.

Firstly, the video need to be stored, and for further operations it is preferred the frames to be converted in grayscale. Since we are interested in detecting the moving object from the video, a high resolution or a good quality of the video is not necessarily what we need.

Instead, we are interested in changes of pixels brightness to see in which direction the car is moving. So more important than the quality of the video is its stabilization. It is important to have a stable background, in this way the moving objects are easier to be detected.

Video acquisition

First step is to extract the frames from the video. This is done in order to have the possibility to make different image processing operations on each frame. By applying techniques like background subtraction or noise reduction, we will acquire the desired results.

Each frame of the video will be further processed and analyzed.

A scaling factor is set to reduce the video dimensions. This was done to obtain a faster processing of the video such that all the computations are made now at a higher speed. So, I have chosen a scaling factor of 10, meaning that the height and the width of the video are now 10 times smaller.

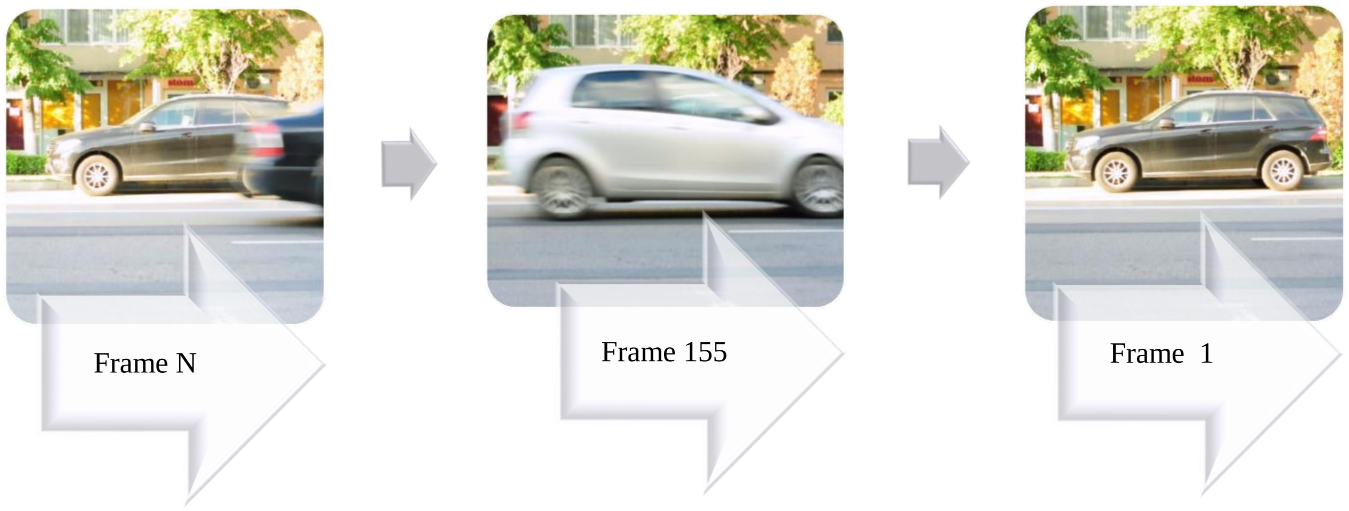


Figure 2.1.1: Video acquisition

Loading of the video is done as follows:

```
>> videofilename = 'veh.mp4'

>> scalingfactor = 10;

>> v = VideoReader(['data/' videofilename]);

>> height = v.Height / scalingfactor;

>> width = v.Width / scalingfactor;

>> NoF = v.NumberOfFrames;
```

In the next figure it can be seen how the resampled image looks like after the scaling process. The quality of the frames is affected, but we can observe in the next part of the application that this will not influence the simulation results.

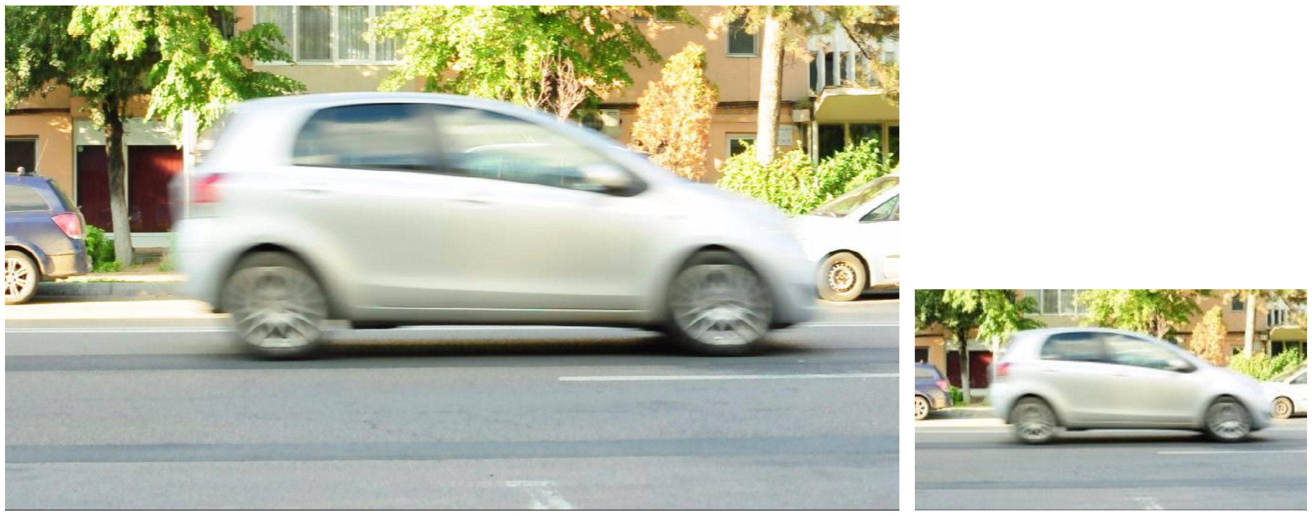


Figure 2.1.2: The original frame and the resized frame

Preprocessing

In this part the way the frames are converted into grayscale will be presented. The grayscale conversion is done for an easier manipulation of the images for further operations.

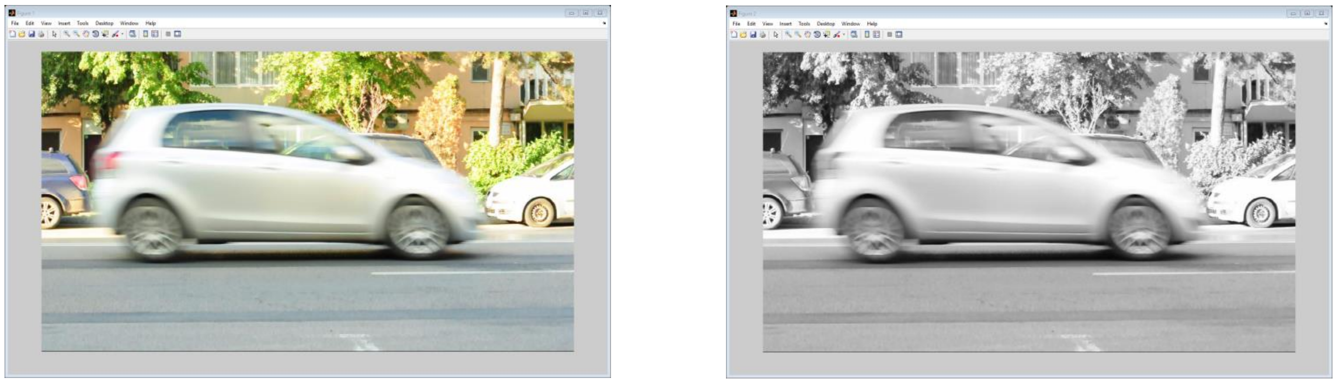


Figure 2.1.3: The original and the grayscale frame

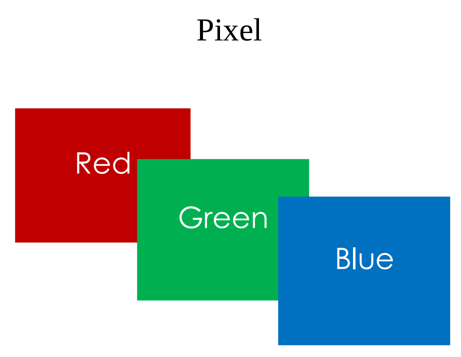
So far, the video frames are stored in a 2D integer array. For the moment we have obtained an array with pixel values of integer type between 0-255 ranges, each value representing the brightness of the pixel. Because many Matlab functions require arguments of type double, the next thing needs to be done is to change the pixels values in floating point format. This is the result of dividing each pixel value to the maximum value, 255. In this way the pixels will have values in the range 0 to 1.

The preprocessing script is as follows:

```
>> frame = v.read(i);

>> frame = imresize(frame, 1/scalingfactor);

 >> frame = rgb2gray(frame);

 >> frame = double(frame)/255;
```

Taking the image in integer format, wherever the second image has a pixel value larger than the first image, in the image resulted from the operation, we will get a pixel value equal to 0, because the image cannot be represented using negative numbers.

This conversion is done not only to ease the operations with images, but there is a more important reason for doing that.



Different operations applied to images are done easier if the image pixels are represented in floating point format. Some example can be the subtraction of an image from another image or multiplying an image with a scalar.

0	25	50	75
100	125	150	175
200	225	250	255

0	0.09	0.19	0.29
0.39	0.49	0.59	0.68
0.78	0.88	0.98	1

Figure 2.1.4: Frame pixels conversion in double format

Vehicle detection area

I set as a region of interest to be studied and by analyzing the change of pixels values in that area, we can deduce when a car passes. For each frame of the video, a vehicle is detected when passing through a small portion chosen of the total width of the entire frame. I have chosen a vehicle to be encountered when passing through the middle of the frame. This can be seen in the following image.

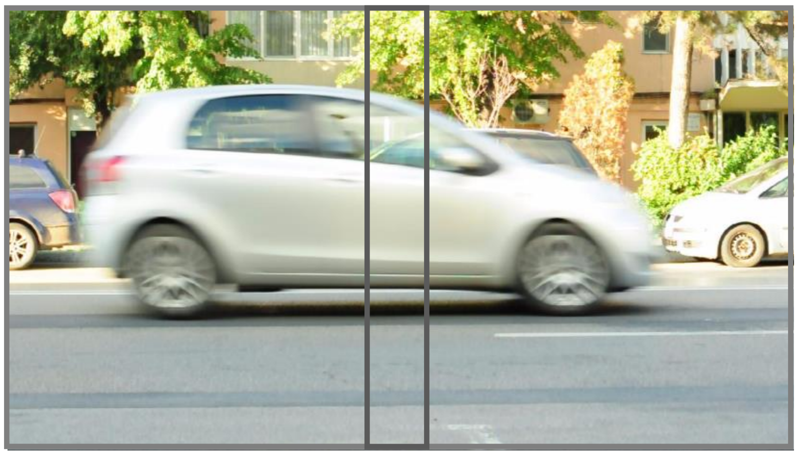
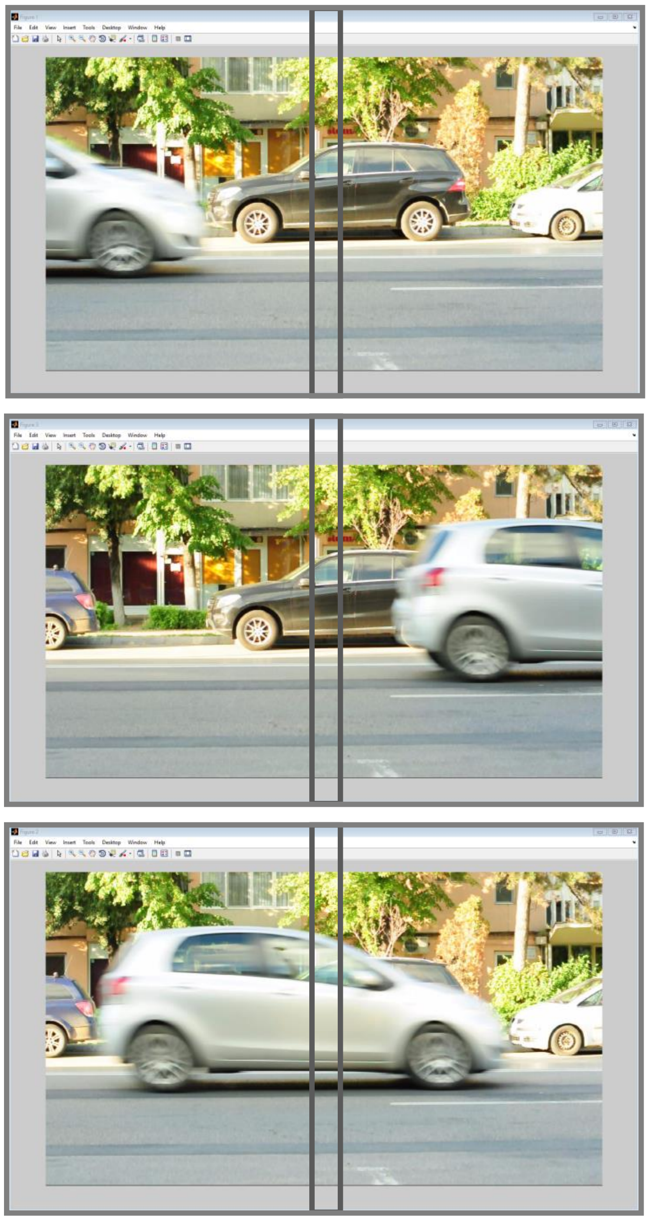


Figure 2.1.5: Region of interest

The vehicle detection algorithm works as follows

- For each frame the mean value of the pixels between the specified columns is calculated;
- When a car passes through that region, most of the pixels will be close to 1, representing white. Therefore, the mean gets a higher value when an object passes and a peak appears on the signal plot;
- For each frame the resulted value is stored in a vector.



The vehicle detection was established to be made when the object is approaching the middle of the frame. This was decided after more trials I have made by using different values. It was noticed that the output signal looks more accurate and clear in these conditions, meaning that the peak is more prominent and is less affected by the present noise.

Script implementation to select the region of interest:

```
>> colwidth = 0.1 * width;

>> referencecolumn = 0.5 * width;

>> colstart = round(referencecolumn - colwidth/2);

>> colend   = round(referencecolumn + colwidth/2);.
```

Background and foreground extraction

Due to the fact that in a video the background varies very slowly, a low pass filter will be used to remove the fast changing components present in each frame, and to keep the backgrounds. Actually, the foreground is the one that moves much faster, so by applying this filter, the foreground is separated from the original frame.

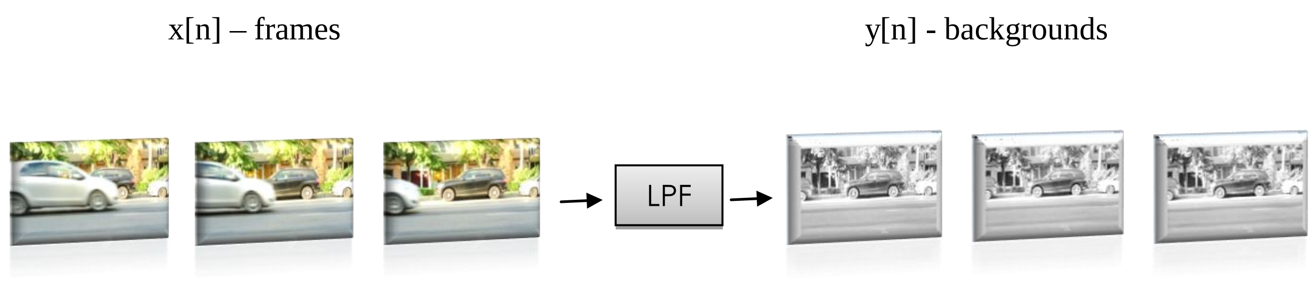


Figure 2.1.6: Background extraction

The foregrounds were extracted in order to analyze the variations of the signal which will show us how many cars were passing.

The following equations describe the type of filter used for the background extraction process.

$$y[n] = y[n-1] + \alpha * (x[n] - y[n-1])$$

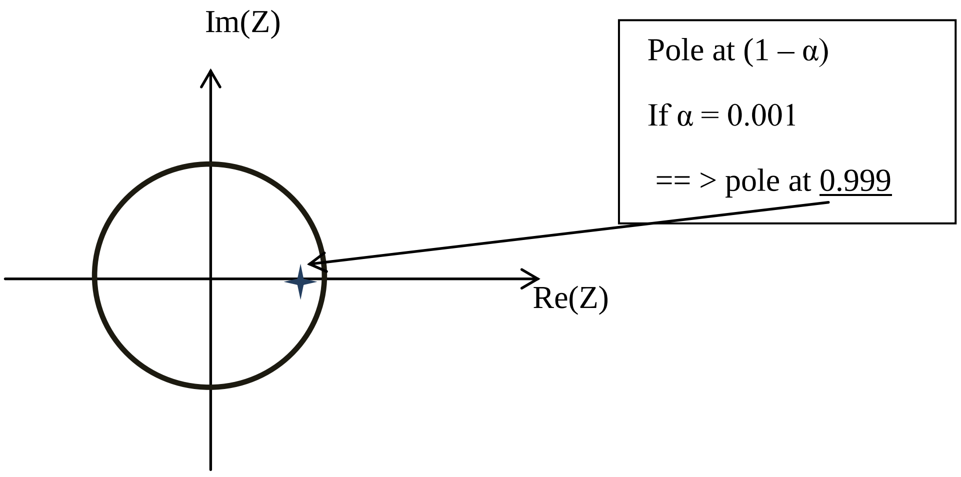
$$y[n] = y[n-1] * (1-\alpha) + \alpha * x[n]$$

$$Y(Z) = (1-\alpha) * z^{-1} * Y(Z) + \alpha * X(Z)$$

$$H(Z) = \frac{Y(Z)}{X(Z)}$$

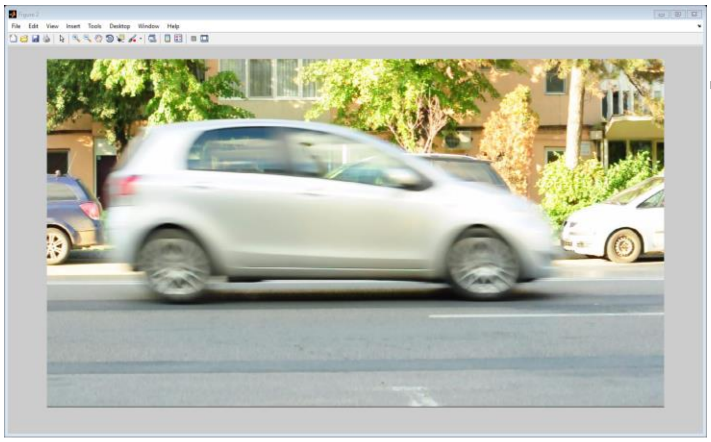
$$H(Z) = \frac{\alpha}{1-(1-\alpha) * z^{-1}} = \frac{\alpha * z}{z-(1-\alpha)}$$

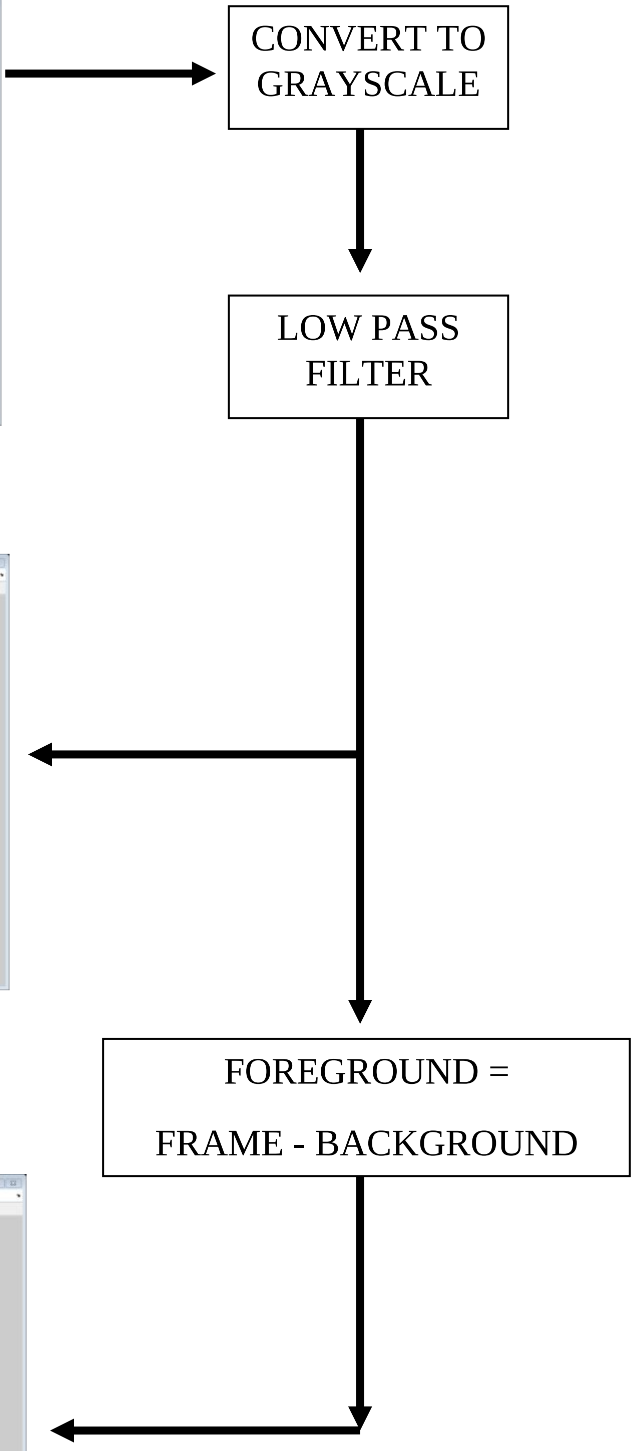
The pole-zero diagram describing the filter is shown in the next figure.



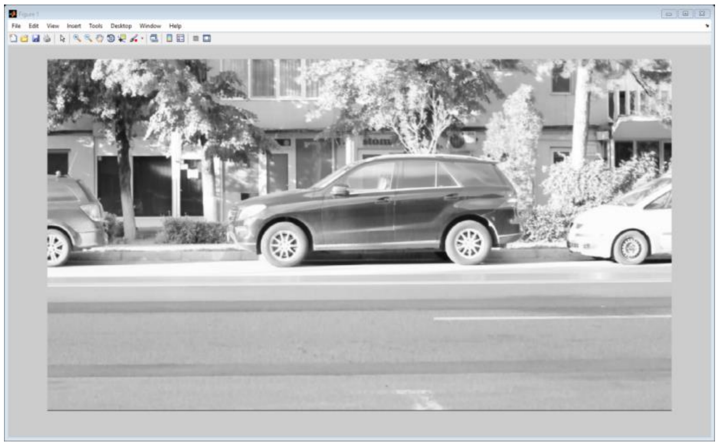
A detailed description of the process is presented below.

Original frame

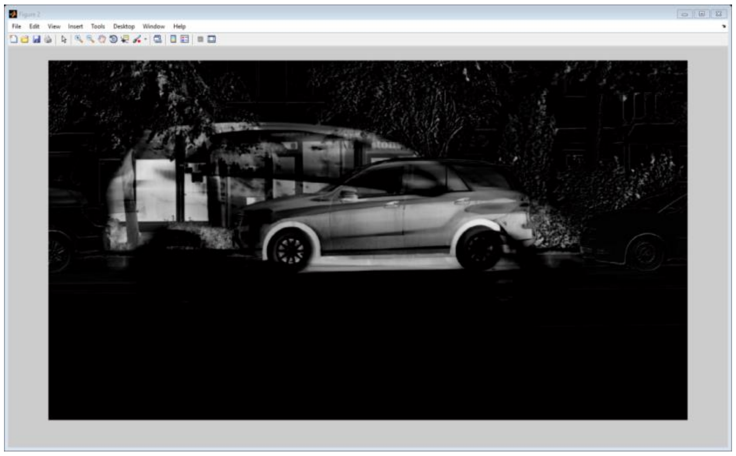




Background



Foreground



Vehicle tracking using Optical Flow

For this application, Optical flow method was used for motion estimation of the vehicles.

In the next image the optical flow lines are drawn on each frame of the video to show how the vehicle motion is estimated.

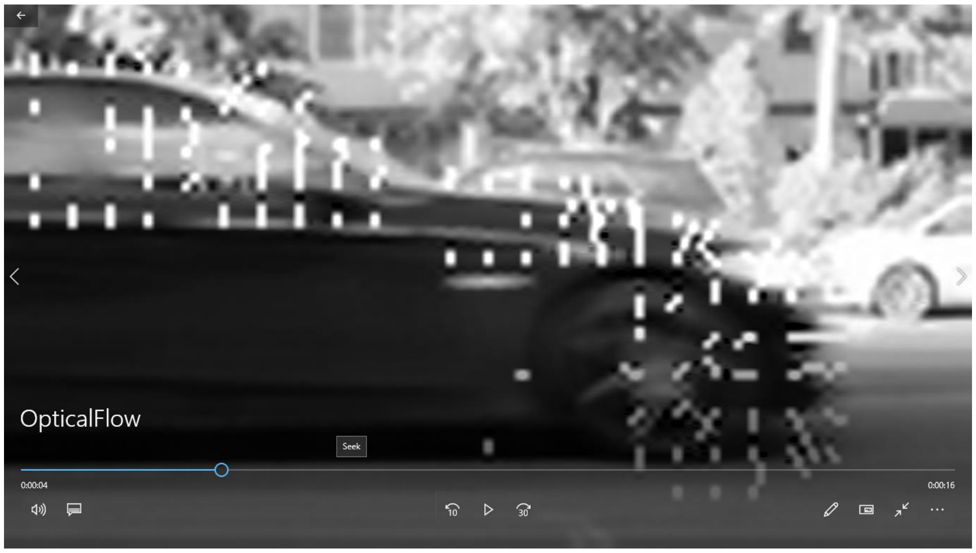


Figure 2.1.7: Optical flow

In the next figure it is shown the motion vector representing the displacement of each pixel between two consecutive frames. By studying this, we can determine the vehicle direction.

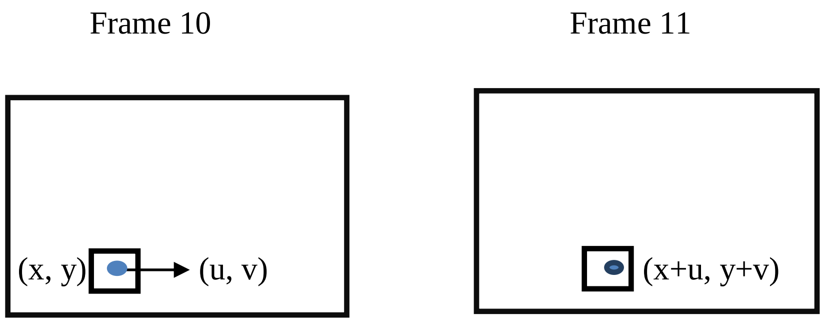


Figure 2.1.8: Displacement vector

The optical flow algorithm already implemented in Matlab was applied in order to estimate the vehicles direction.

```
>> hof = vision.OpticalFlow('ReferenceFrameDelay', 1);
>> hof.OutputValue = 'Horizontal and vertical components in complex form';
>> of = step(hof, frame);
>> ofsignal(i) = sum(sum(real(of(:, colstart : colend)))) / (height * (colend - colstart));
```

An object to store the optical flow matrix is created. The parameters were set such that the pixels displacement for the current frame was calculated having the previous frame as reference. In consequence, the optical flow object will estimate the direction of the motion between one frame to another. The velocity matrix, resulted after the computations performed on each video frame, will show the horizontal and vertical optical flow.

Optical flow algorithm used in the application is described below:

- in the first frame, find the coordinates of a pixel;
- search in the next frame the pixel having the same brightness value;
- the displacement vector is found by the difference between the spatial coordinates where the pixel was found in the current frame and its coordinates in the previous frame;
- the result is a matrix with complex values of form u + j*v;

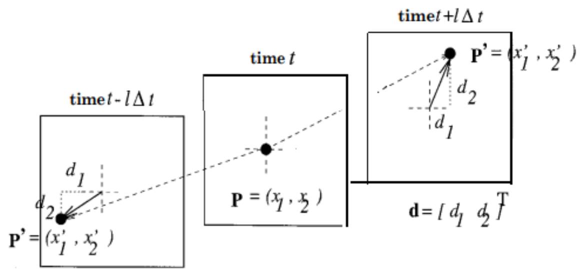
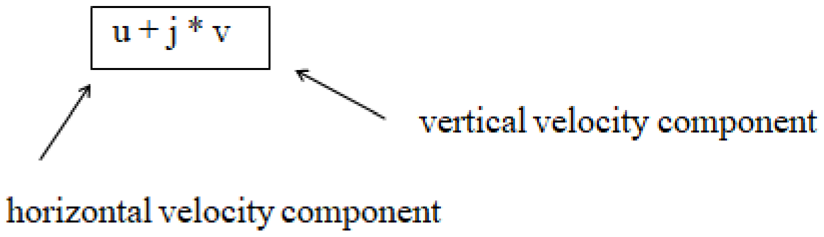

time $t - l\Delta t$
time t
time $t + l\Delta t$
d_1
d_2
$\mathbf{P'} = (x'_1, x'_2)$
$\mathbf{P} = (x_1, x_2)$
$\mathbf{P'} = (x'_1, x'_2)$
$\mathbf{d} = [d_1 \ d_2]^T$

Figure 2.1.9: Forward and backward estimation using Optical flow[6]

[6] A. Murat Tekalp, *Digital Video Processing*, Upper Saddle River, NJ 07458, p. 76

We are interested to find out how the pixels have changed from one frame to the next one in horizontal plane. An optical flow signal is plotted to detect the car direction and will be computed using the real values of the matrix. As in the case of vehicle detection, the region we are interested to detect all these pixels changes is set to be in the middle of the frame.



Counting the vehicles passing

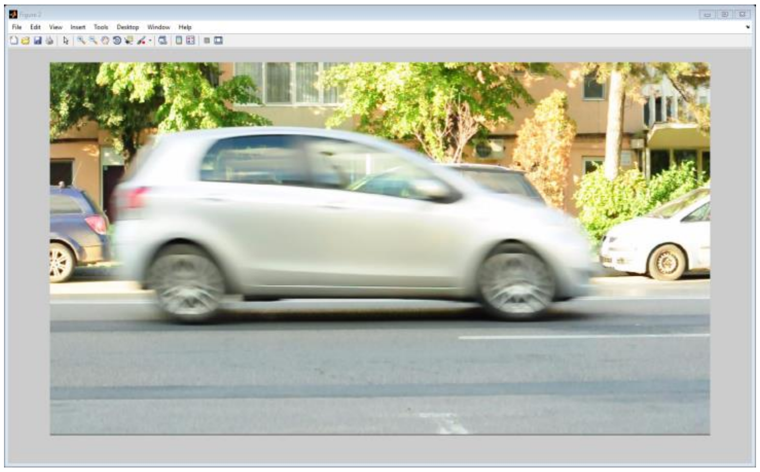
We want to keep track of the cars moving on the street, so the number of vehicles passing no matter in which direction will be summed up and displayed on the up – left corner of the video. The following algorithm is implemented.

Analyzing the changes of pixels in foregrounds, we want to see when a car passes through the selected region. When a vehicle passes, the next step is to see in which direction it moves. If it moves from left to right, the mean value will have a positive value, hence, if the car comes from right to left, the result will have negative values.

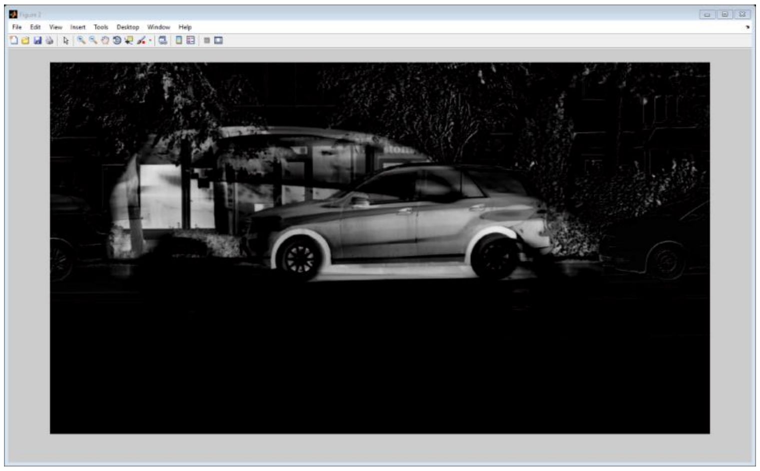
The counting algorithm is presented below:

- Analyze the signal representing the foreground for each frame;
- A threshold is set; when it is exceeded, we know that a vehicle is passing;
- Now, we must look at the optical flow signal, to observe in which direction the vehicle goes;
- If the vehicle goes from left to right, the number of cars increases;
- If the vehicle goes from right to left, the number of cars decreases.

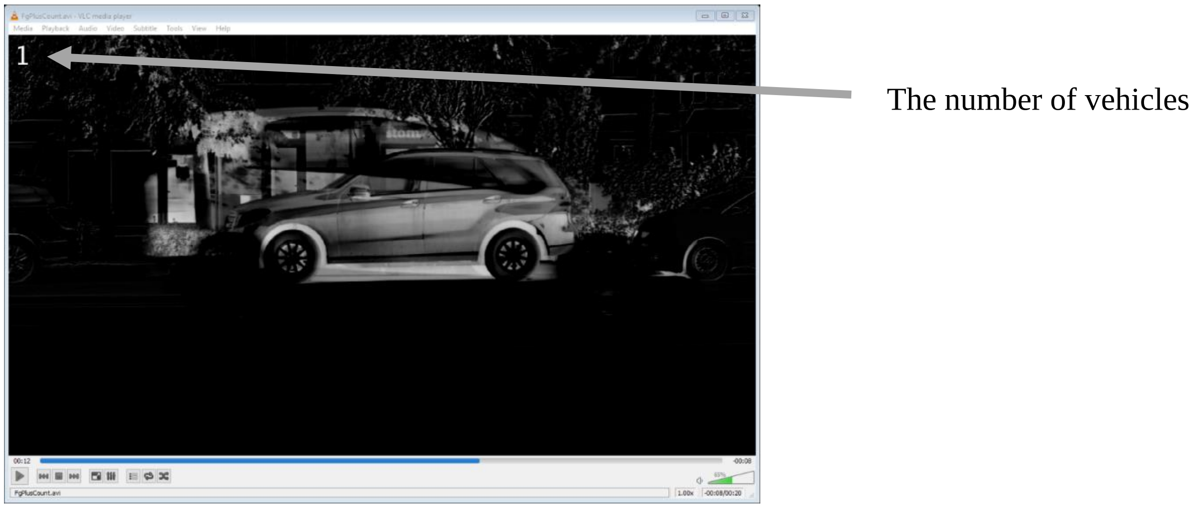
Original frame



Foreground



The number of vehicles



Filtering the signals

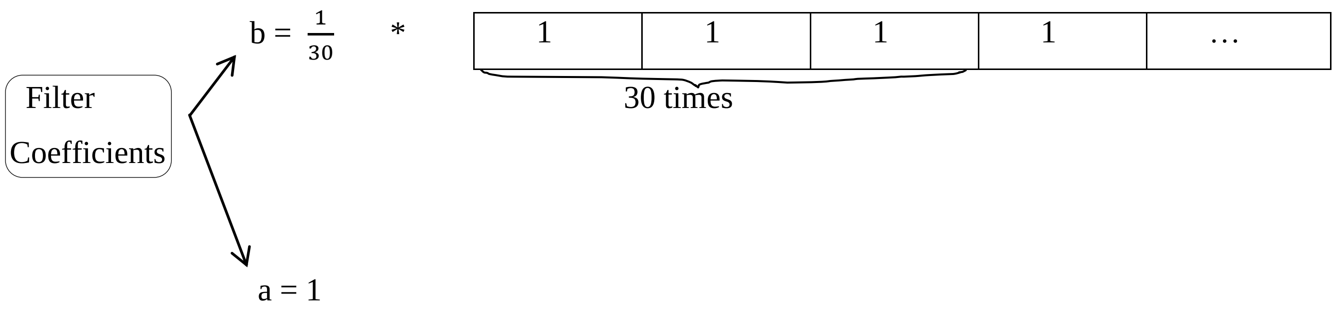
This process is done to obtain a smoother and to reduce the noise. A low pass FIR filter is used with a mask of 30 elements for computations. Meaning that for approximately one second, the mean value of the signal is calculated.

The script used is the following:

```
>> b = 1/30 * ones(1,30);

>> a = 1;

>> ofsignal_filtered = filter(b,a,ofsignal);

>> foreground_filtered = filter(b,a,foreground_signal);
```

The line *ofsignal_filtered = filter(b,a,ofsignal)* filters the data in vector 'ofsignal' with the filter described by numerator coefficient vector *b* and denominator coefficient vector *a*.

The *filter()* function will filter the data of the optical flow signal, by applying a filter mask of 30 elements. Each output element after filtering will be an average value of its previous 30 elements.



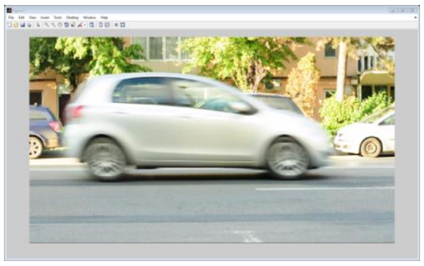
2.2 SIMULATION RESULTS

Video sequence

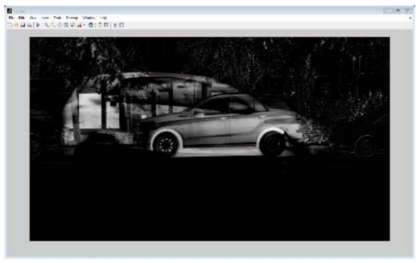
In the following images the original video frame, the background and the foreground calculated for each frame are shown. The implemented algorithm for background and foreground extraction was already presented in details in the previous subchapter, so I will write a short description here.

Briefly, for each frame of the video, a low pass filter is applied in order to keep the frame small variations, represented by the background. The foreground is calculated as difference between the image and the background.

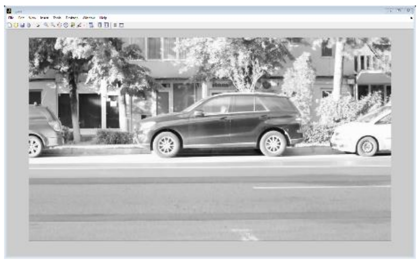
Original frame



Foreground



Background



As shown in the application description presented in the previous section, the vehicle detection was established to be made when the object approaches the middle of the frame.

For the computations a small part of the frame was considered. The mean value of the pixels in the selected region is calculated and as a result we can see how the signal describing the variation of the foregrounds changes over time.

When a car passes through that region, most of the pixels will be close to 1, representing white. Therefore, the mean gets a higher value when an object passes and a peak appears on the signal plot;

A low pass filter will be applied to filter the signal representing the foregrounds variations during the whole video. We must be careful to the delays that appear when using a smoothing filter, because for some applications these delays can represent a problem.

In the next figure the foreground variations can be seen. The mean value of the pixels in the selected area is shown for each video frame.

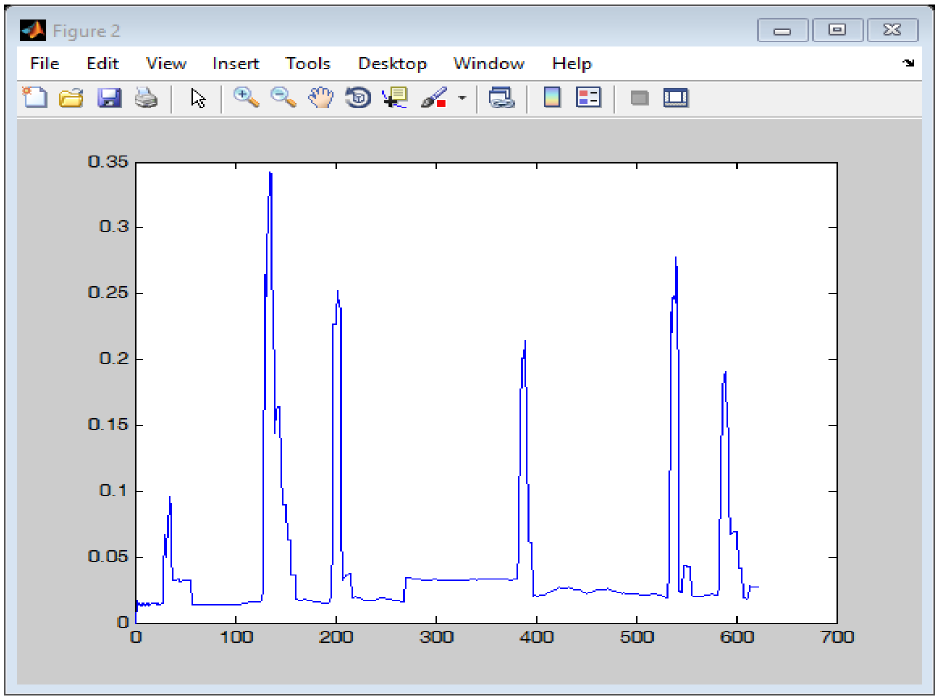


Figure 2.2.2: Foreground variations

In the following figure is presented how the foreground signal varies when a car passes. We can see that six vehicles were passing during the video recording.

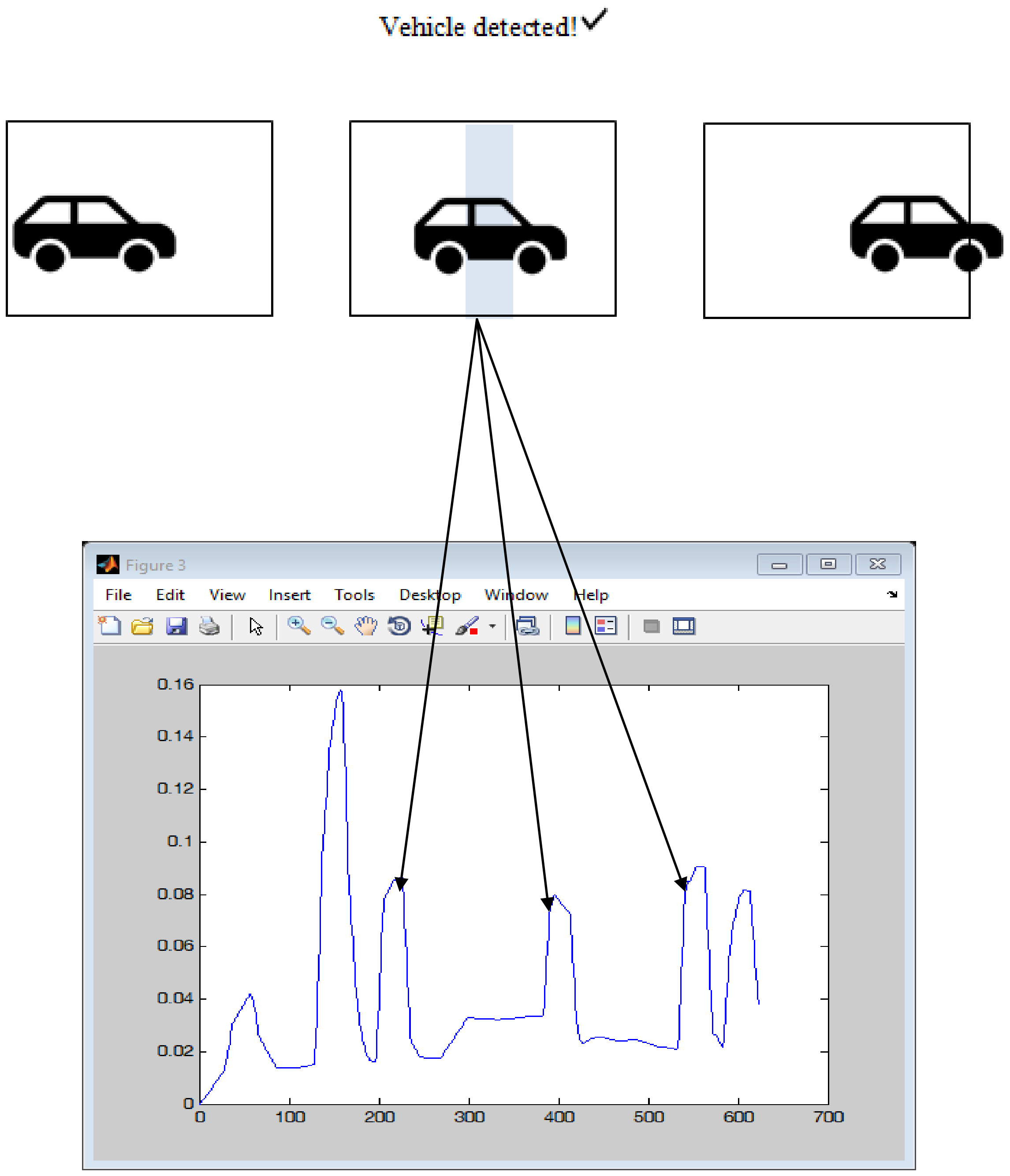


Figure 2.2.3: Foreground variation signal after filtering

The optical flow signal shows the vehicle direction and it is computed as follows:

- for each frame the optical flow is computed having as reference the previous frame;
- the values are of form $u + j * v$;
- to see the vehicle direction, we need to see the pixel displacements between frames on horizontal plane;
- an average value for the pixels situated in the selected region is made;
- if the mean value is: $\rightarrow$ negative $\rightarrow$ car moves from right to left

  $\searrow$ positive $\rightarrow$ car moves from left to right

In the next figures the signal showing the object direction is plotted. In the left image is represented the original signal and in the right one the same signal after filtering.

The noise presence affects the signals, so they need to be filtered, in order to obtain more accurate results. After the filter was applied, we can observe that the signals are smoothen, especially the signal showing the vehicle direction. It looks much better after the smoothen process.

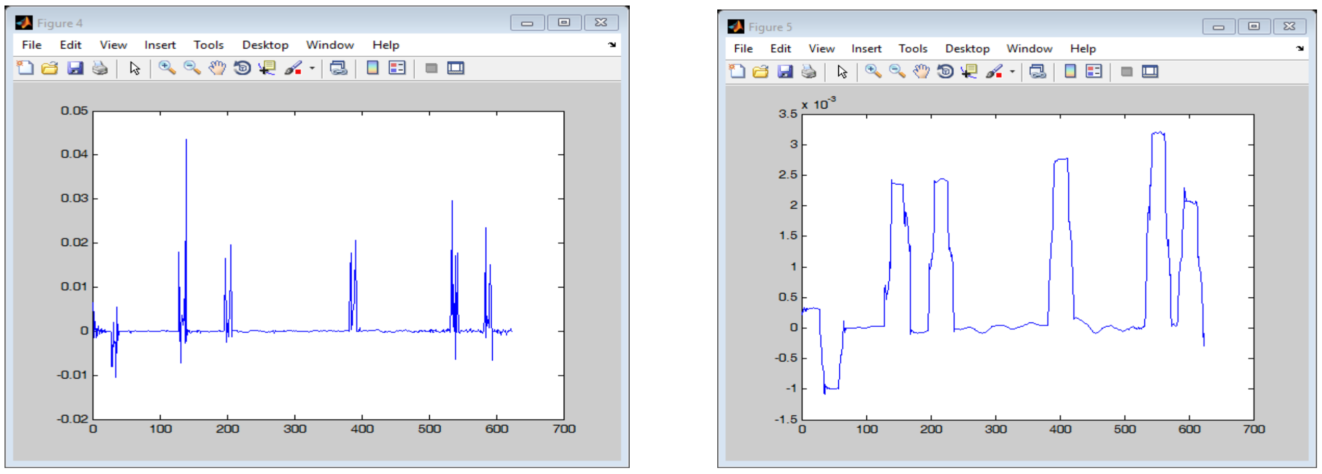
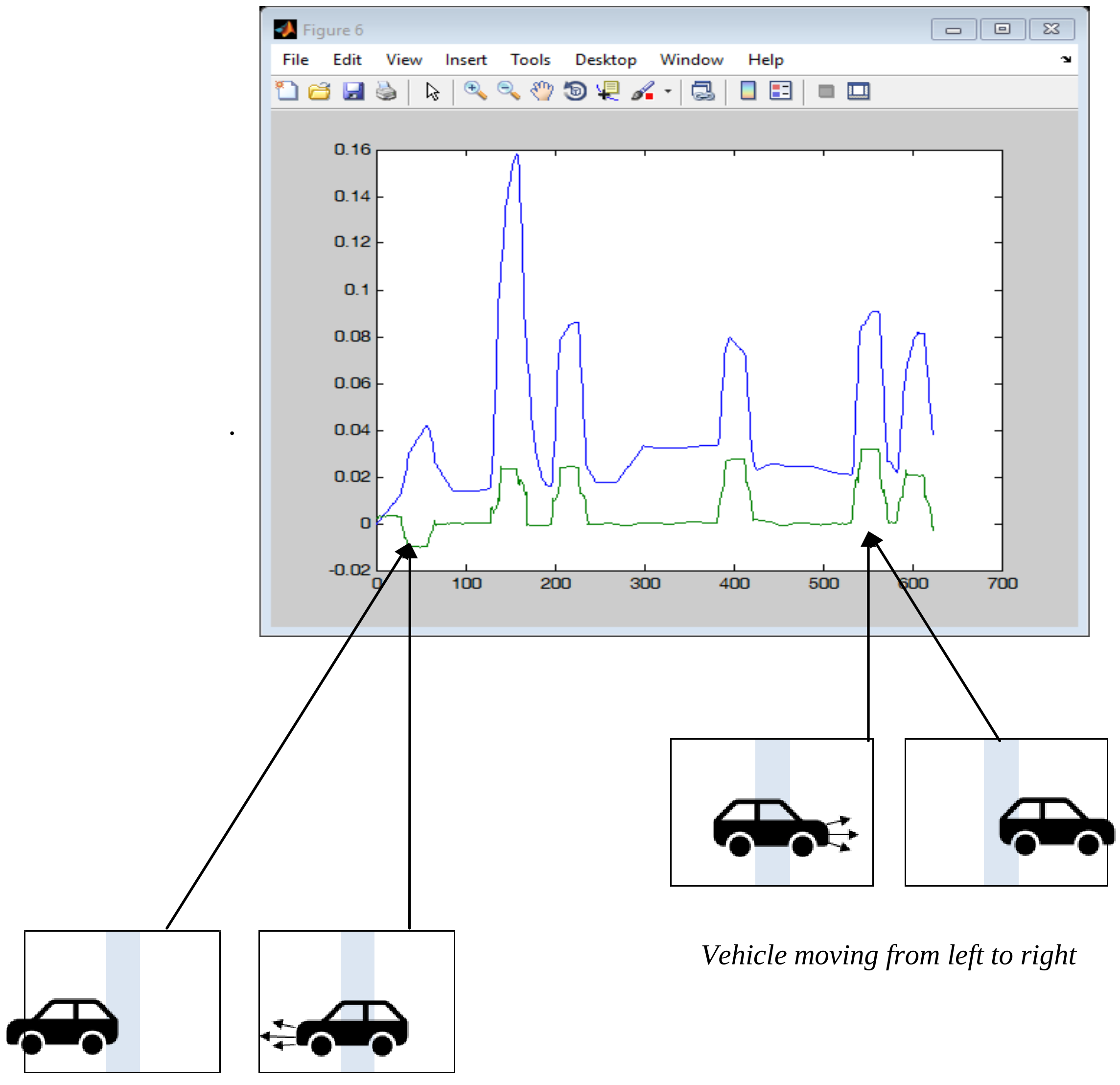


Figure 2.2.4: Optical flow signal

In the following, the signals representing the foreground variations and the optical flow of the video are shown. It can be seen from the following figures how the direction of a vehicle is plotted, depending on its moving direction.

🟦 Foreground variations

🟩 Optical flow signal



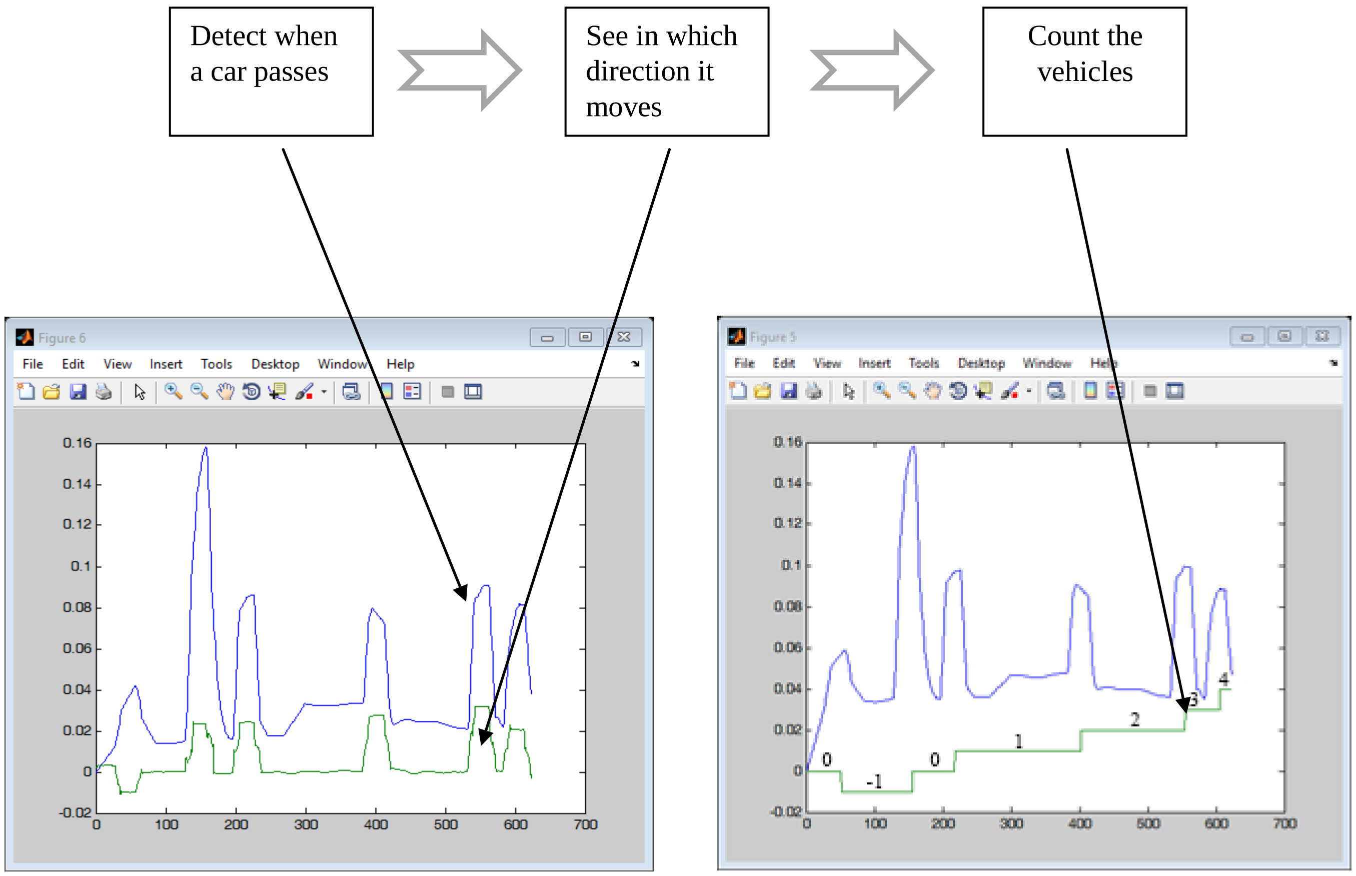
Vehicle moving from left to right

Vehicle moving from right to left

We are interested to see in which direction each car moves, so looking at the green signal, we can notice that the first vehicle moves from right to left and the other five go in the opposite direction. As I already explained in the previous chapter, if the vehicle moves form right to left, the mean value of the pixels has a negative value, exactly as the graph shows for the first car.

The idea is to keep the moments when the signal starts to increase, meaning that a vehicle passes, and also the moment when it starts to drecrease. By doing this, we can find out approximately at which index we have the maximum value of the peak. At that particular moment, we will take a look at the signal describing the vehicle direction. By analyzing it, we will see that the number of cars moving form left to right will be added to the total number of cars, being seen as vehicles which enter in an area. While the cars moving from right to left, will be interpreted as the cars leaving an area, and will be subtracted from the total number of cars.

The algorithm for counting the vehicles passing through a certain region is presented in the next figures



Short summary of the implemented algorithm:

- read the video sequence;
- extract each frame in RGB form;
- convert each frame to grayscale;
- consider a region of interest, chosen in the middle of the frame;
- for each frame of the video apply a low pass filter to extract the background;
- the foregrounds are determined as difference between the frame and the background;
- see the foreground signal obtained as the variation of pixels in that region, for each particular frame;
- when a car passes, the pixels values will be close to 1, white, so the mean value of the pixels will have a higher value;
- compute the optical flow to estimate the vehicle direction;
- see vehicle direction by analyzing the optical flow signal;
- if the car moves from left to right, a positive peak will be shown; otherwise, a negative peak will appear;
- when a car passes through the middle of the frame, look at its direction and decide the following:
    - if the car moves from right – left, decrease the number of total cars;
    - if car moves from left – right, increase the number of total cars;

2.3 RESULTS OBTAINED IN DIFFERENT CONDITIONS

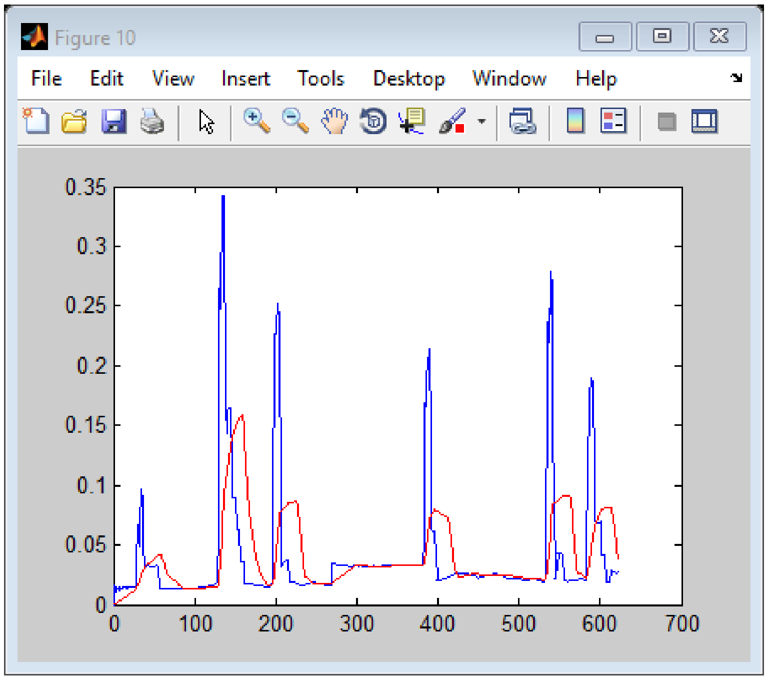
In this chapter I will describe how this algorithm works in different conditions, by changing certain parameters.

Changing the filter mask size

In the next images will see how the foreground signal is smoothen, by applying different low pass filters. First image shows the results after a filter mask of 30 elements is applied, and the next ones are computed with different values. The blue signal is the original one and the red signal is the filtered one. It can be observed a small delay in the signal at the filter output.

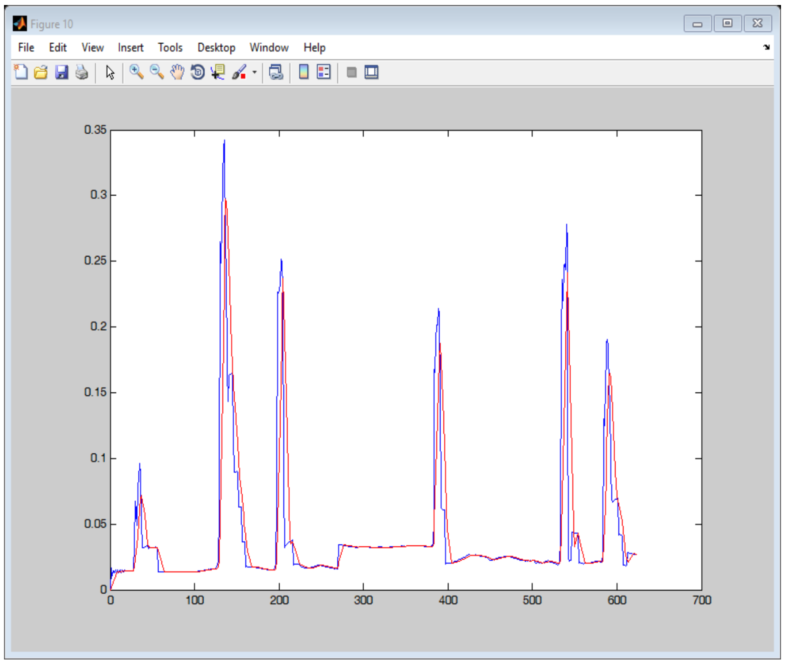
The vector used as a mask is formed by 30 values of 1/30.

Using a mask of size 30



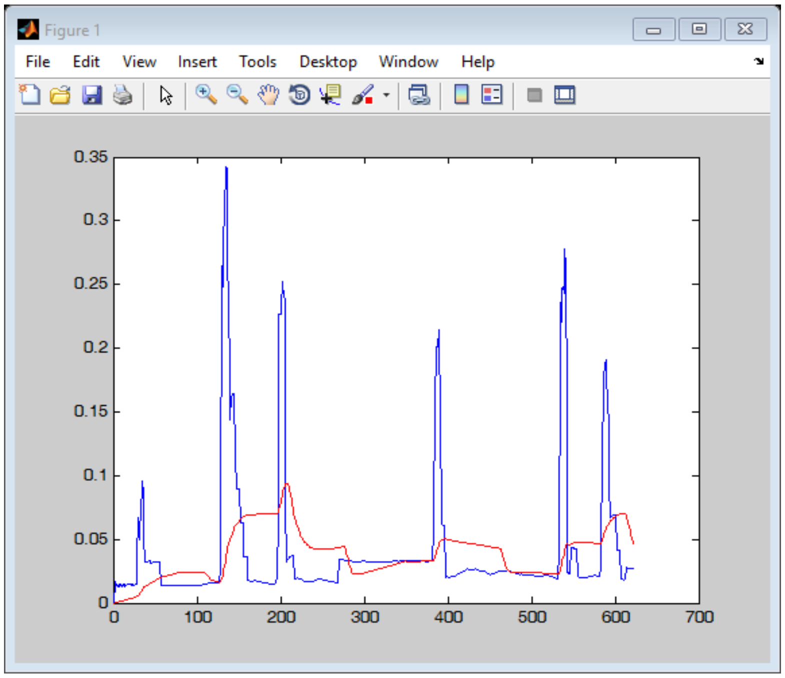
🟦	The original signal
🟥	The smoothed signal

Using a mask of size 8



🟦	The original signal
🟥	The smoothed signal

Using a mask of size 80



I have made a comparison of the results, depending on the size of the filter. We can observe that the most suitable choice for this application is to take a 30 elements mask. The output signal after passing through the smoothing process follows original one quite well.

But, this is not the case for the other two filters, for which I have made the computations. As it can be seen in the second figure, a mask of size 8 is used. This did not give a good result at the output because the noise was hardly reduced.

Using a high order low pass filter, the signal was filtered too much, losing some important details, such that the detection of the vehicle became difficult to be done correctly.

The low pass filtering is done by imposing a mask of order 30 and then neighboring calculations are done. The idea of a low pass filter, as we know, is to allow the low frequency components to be present in the output and to stop the high frequency components. A noise is always considered to have high frequencies.

This type of filtering is the same as convolution. The kernel being used is a matrix with each elements of value 1/30. We use this to remove the noise in the video and to obtain better results.

Using 100% of the frame width for vehicle detection

When the entire image is considered, the foreground signal will be calculated as the sum of differences for all lines and columns.

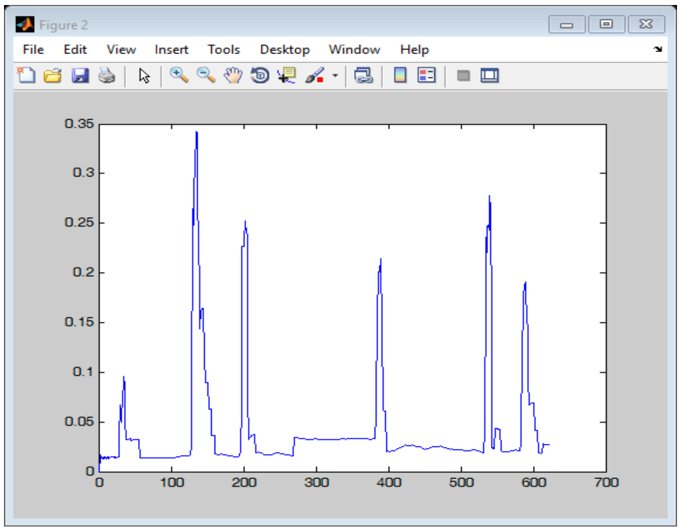
Script implementation to select the region of interest:

```
>> referencecolumn = 0.5*width;

>> colstart =1;

>> colend   = width – 1;

>> foreground_variations(i) = sum(sum(abs(foreground(:, colstart : colend)))) / (height * (colend - colstart));
```

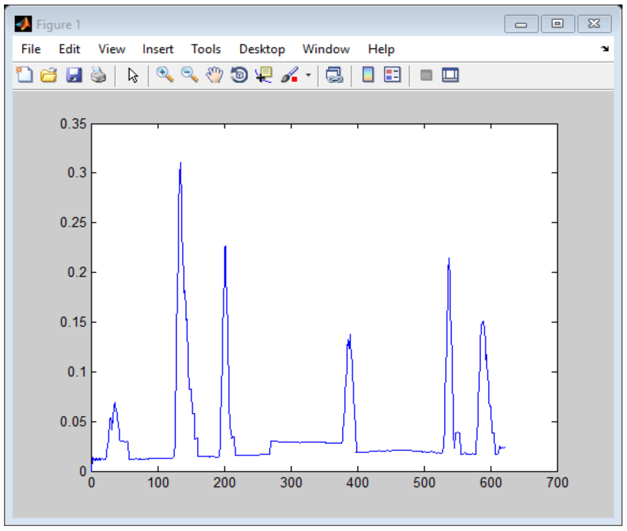
In the next images, it is shown how the output signal looks like when is considered 10% of the frame width and how it looks when the total frame is considered. We can point out the differences, to see which method fits better for our application.

We can see that the signal looks very clear when taking into consideration the whole frame too, and I can conclude that both situations would work well for this application. The only difference being the mean value of the pixels.

10% of the frame width



100% of the frame width



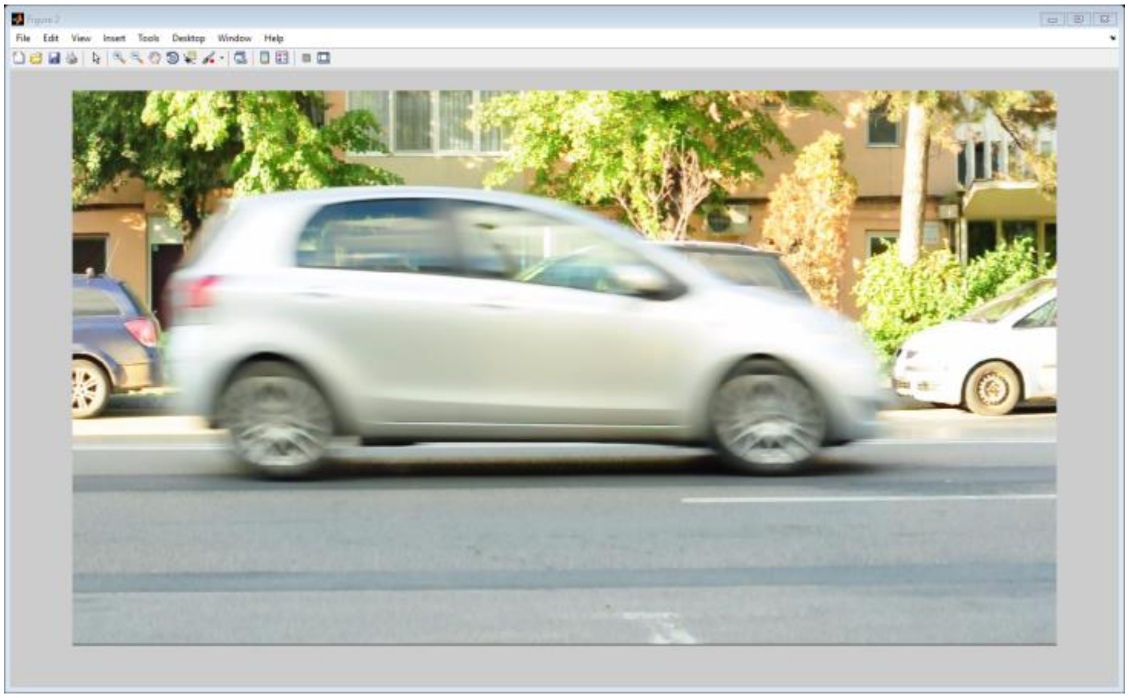
Changing the low pass filter parameter – alpha

By changing the value of alpha, I have made the following conclusions:

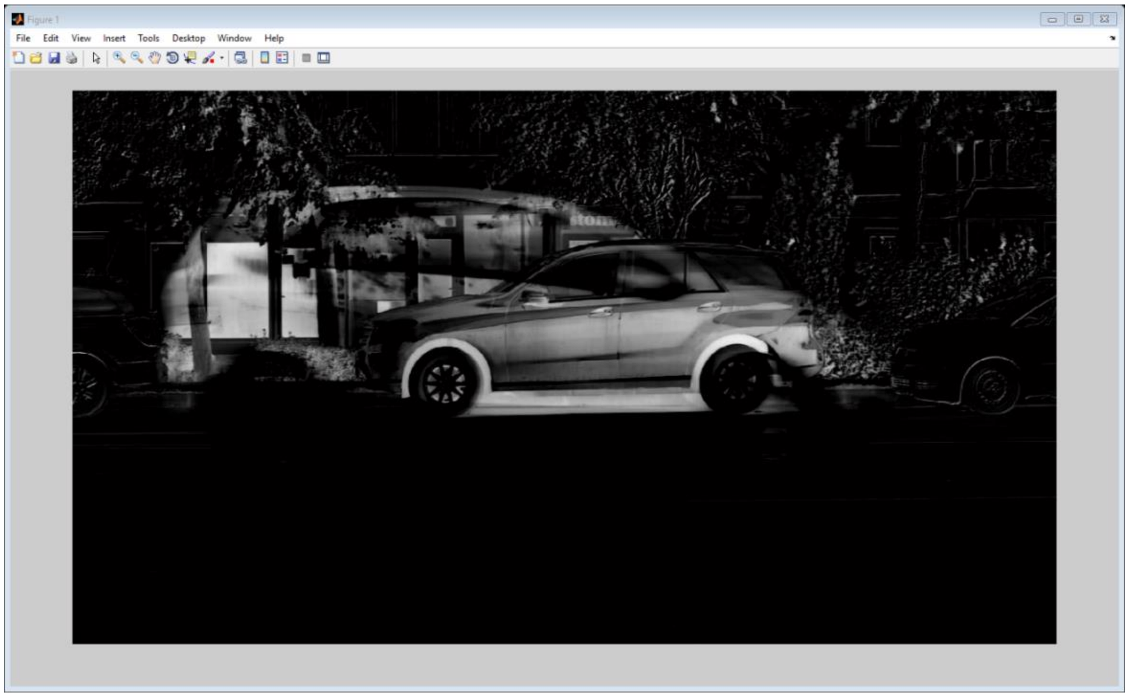
- While the alpha parameter is increased, we can observe that the background extraction process will suffer modifications, therefore the foreground will be also affected;
- If alpha parameter is increased too much → the car will belong to the background → the foreground will be a black image because the frame and the background will have approximately the same pixels values.

In the next figures, some images are shown to see how the foreground extraction is affected if the value of the low pass filter parameter '*alpha*' is changed.

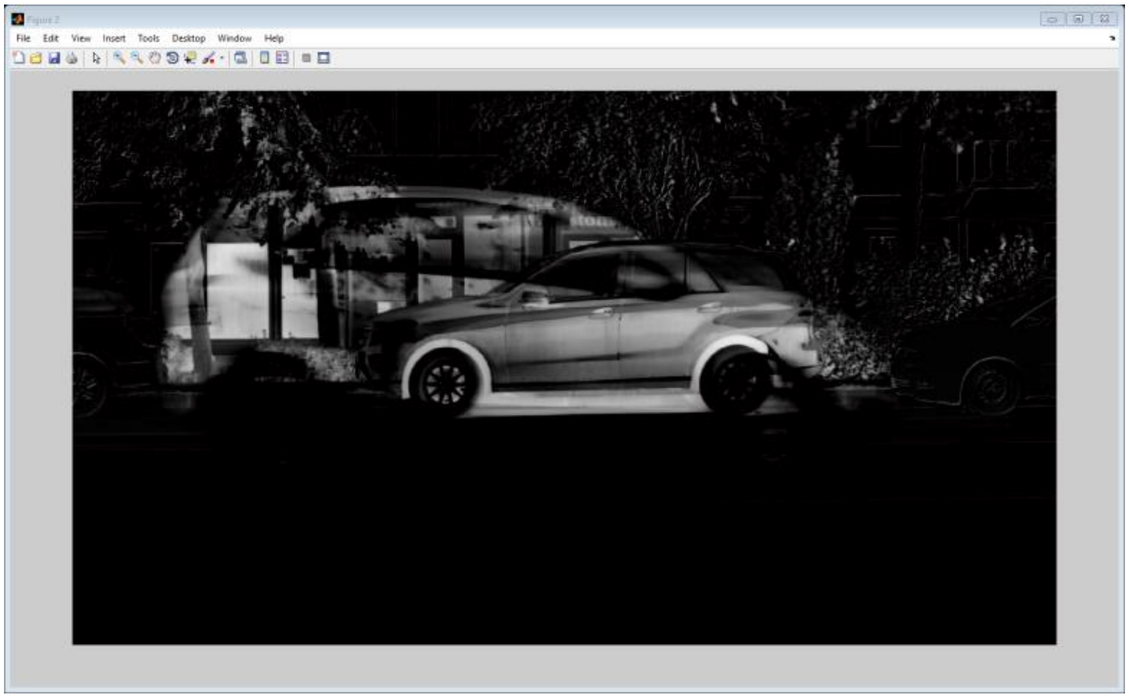
original frame



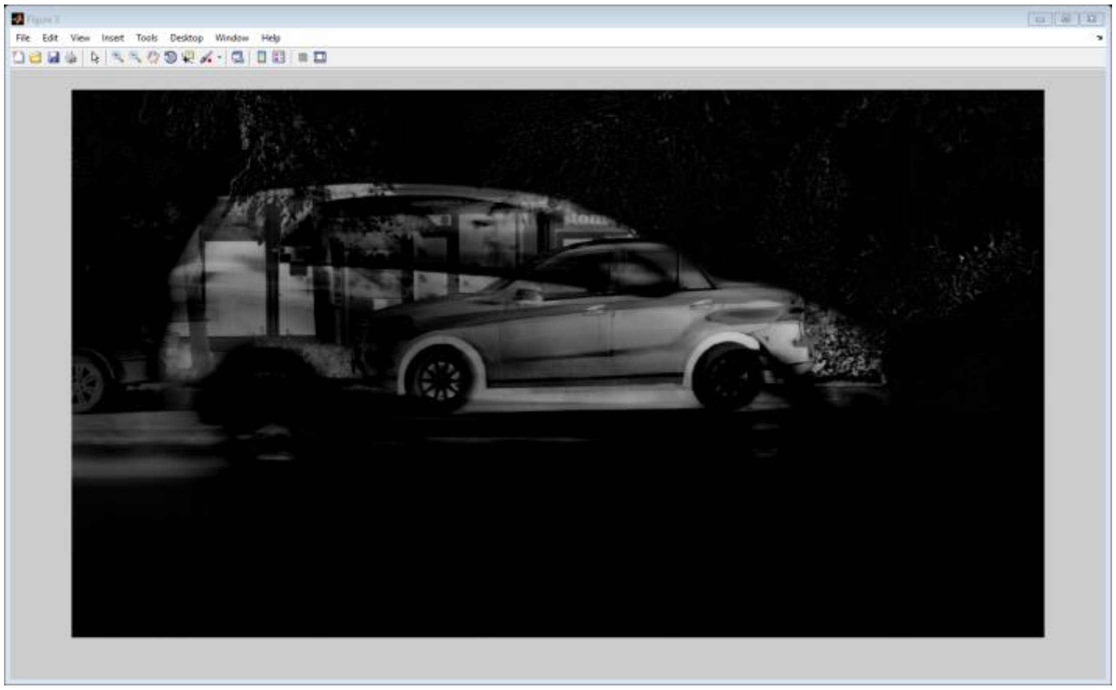
alpha = 0



alpha = 0.001



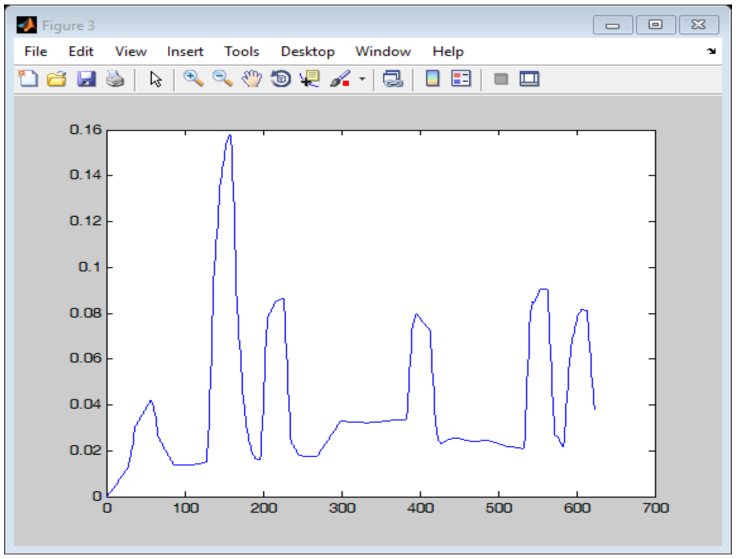
alpha = 1



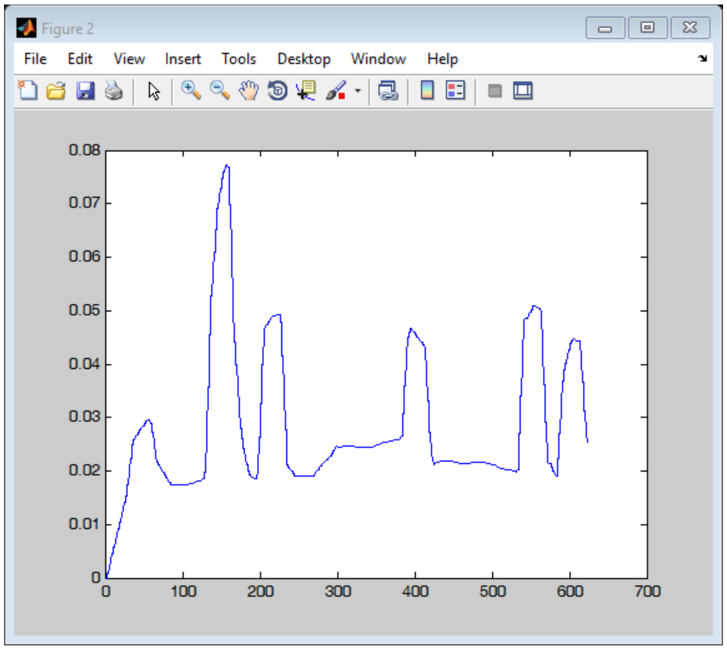
Changing the video frame rate

We can observe that by modifying the frame rate, no significant change was produced in the output.

30 fps



15 fps



Using another method for motion estimation

An alternative for Optical flow to estimate motion is the Block matching method.

Block matching technique works by dividing the current frame in blocks of pixels. An algorithm is used to search the best match for a specific block of pixels in the previous frame. The motion vector will be represented as being the displacement between the coordinates of the searched block and the coordinates of the found block.

Basic assumptions :

- The light is constant
- Objects are rigid
- The object comes or goes out from the scene

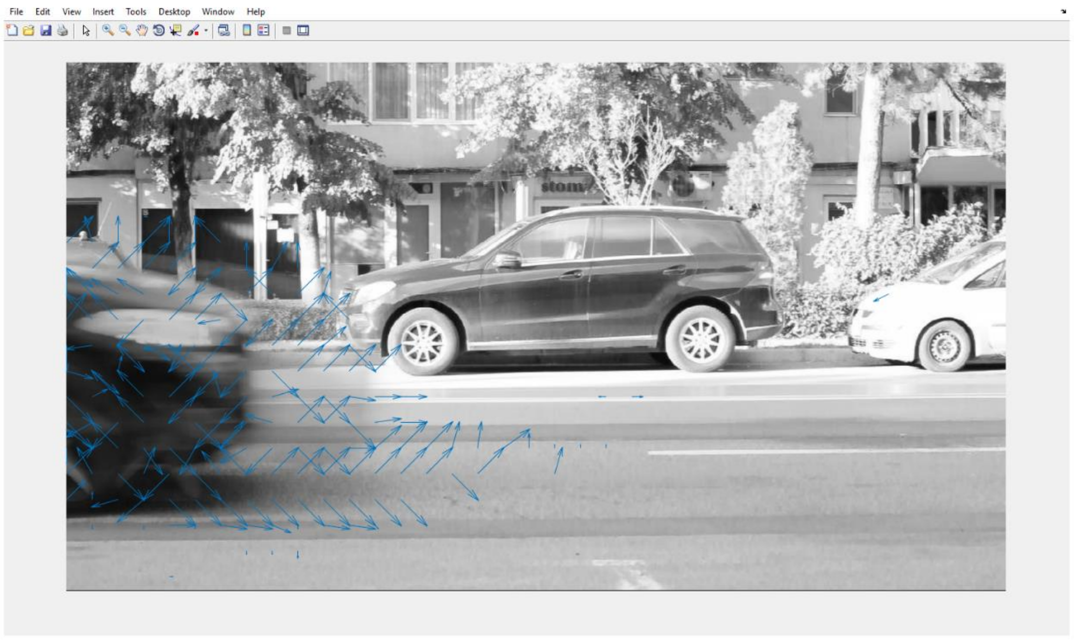


Figure 2.3.2: Motion estimation between frames 127 and 128

CONCLUSIONS

Image processing and computer vision subjects are in a continuous growing in the last years. Every day applications are developed more and more to maintain a sustainable evolution in this field.

Nowadays, the computers are able to understand and to take decisions without human intervention, based on the information collected from the real world. Various applications are implemented in the medical field, automotive field, military and many others.

Initially, my idea was to develop an algorithm to detect the vehicle direction in a video on a one way street. The point was to detect if a car is going in the opposite direction. In this manner, we can see who obeys the traffic rules and to apply different countermeasures.

While researching for this project, I realized that it would be interesting if I could also count the vehicles passing through a certain region according to their direction. So, I have improved the algorithm to accomplish this. One application idea would be to count the number of cars that enter or leave a parking lot to know the free parking spaces.

In the chapter where the experimental results were presented, it could be observed how the algorithm works in the given conditions. When we test the algorithm it can be noticed that even for bad quality videos the simulation results are very good. In addition, in the second part of the chapter, we have shown how the program behaves if some parameters are changed.

It can be seen that we have achieved very good results in the conditions in which the video was recorded during the day. On the other hand, definitely if there was not enough light outside, the detection of cars would not be possible.

Remark the fact that in the analyzed video we do not have the case when two cars are passing at the same time, so each car is easily and correctly detected. Otherwise, then again we would have a false detection of a vehicle and thus a wrong counting.

Some possible improvements can be:

- to develop a color-based processing algorithm for the video;
- to find an automatic method for choosing the threshold used for car detection;
- to detect when overlapping occurs to count the vehicles properly;
- to use the algorithm in real-time systems.

As a conclusion we can say that the algorithm will work pretty well in some situations, but it also has limitations if it is used in more complex applications.

BIBLIOGRAPHY

[1] Rafael C. Gonzales, Richard E. Woods, *Digital Image Processing (2nd Edition)*, New Jersey

[2] Alan Bovik, *Handbook of Image and Video Processing*, 2000

[3] A. Murat Tekalp, *Digital Video Processing*, Upper Saddle River

[4] https://www.tutorialspoint.com/dip/applications_and_usage.htm

[5] Laurentiu-Mihail Ivanovici, *Procesarea imaginilor*, 2006

[6] Rafael C. Gonzales, Richard E. Woods, Steven L. Eddins, *Digital Image Processing Using MATLAB*, Upper Saddle River

[7] International Journal of Scientific & Engineering Research Volume 4, March-2013, *Vehicle Detection and Tracking from Video frame*

[8] International Journal of Computer Applications Volume 83, *December 2013, Real Time Traffic Density Count using Image Processing*

[9], Thomas B. Moeslund, *Introduction to Video and Image Processing*, 2012

[10] Forsyth and Jean Ponce, *Computer Vision: A Modern Approach (2nd Edition)*, Prentice Hall, 2003

[11] John G. Proakis, Dimitris G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications (4th Edition)*, 2006

[12] Béatrice Pesquet-Popescu, Marco Cagnazzo, Frédéric Dufaux, *Motion Estimation Techniques*

[13] Chris Solomon, Toby Breckon, *Fundamentals of Digital Image Processing*, 2011