## 4. Analysis and Theoretical Foundation

As presented in earlier chapters, the main contributions and final goal of this thesis concerns vehicle traking which and the possibility of it being deployed from a broad range of locations under minimal placement constraints. This chapter will clarify and reson about the details of the presented methods from chapter three and thier scope within the propoposed architecture architecture. We will logically deduct the steps performed for the implementation of the system.

The figure 4.1 describes the general conceptual architecture of vehicle tracking system in which we show the flow of data between the composing modules, dependencies and services provided by the components to other components.



## 

You will find out that the architecture that we propose for our solution closely resembles the trends also approached in the works of [1?] [2?] [3?] [4?]. Our change in the architecture was purely motivated by performance reasons, and consists of the tracker module simultaneously consuming the outputs from the Detector and Background Subtractor module. We will detail later in chapter 5 where we discuss the dependencies of deploying these component on various hardware and software platforms.

Next we give a brief description of the basic building blocks that compose the architecture and the tasks that they play in achieving the final goal.

* The video camera is the primary input for the system, responsible for interfacing the system with the real world by capturing video frames of the surveillance scene.
* The Training Set Acquisition module is responsible for building the Image Training Set that is later used in extracting vehicle specific patterns with which the Classifier is trained.
* The Image Training Set is just a container for a collection of positive training samples, images that depict single vehicles in their area of concern, and negative training samples, images that are suppose not to contain any vehicles.
* The Classifier has the capacity to discriminate between patterns of vehicles and non-vehicles on an entire image, by learning to distinguish them based on the derivated features as a result from processing the training set.
* The Detector is also fed with incoming frames from the acquisition device and has the responsibility of communicating with the classifier and search the video frames for the location and size of vehicles. We call *a detection* a bounding box on the input frame which centrally encapsulates a vehicle.
* The Background Subtractor module is responsible with performing motion segmentation, i.e. identifying the moving regions, in the frames whether they are pedestrians or vehicles. The Background Subtractor module will provide the detector a foreground mask in which the value of each pixel will indicate if the corresponding pixel from the frame contains the projection of a moving object, or any kind of entity.
* The Tracker consumes detections provided by the Detector and the foreground mask provided by the Background Subtractor and performs additional processing using mechanisms to construct the tracks of moving vehicles over time and also validates the consistency of being generated by real vehicles.

Now that we have brief description of what each module is responsible for performing we will dive into the algorithmically details of the architecture and decompose each module as they are scheduled to perform their work in the general process of the system.

## 4.1 Training set acquisition

In general, all machine learning algorithms take as input thousands of data vectors made up of many features, where the number of features might also reach thousands.

The first problem encountered is how to collect the images that make up the dataset. Do you search for online datasets or do you collect it from a security camera and what kind of information do you capture: static or movement. We must end up with thousands of images.

But our job is to recognize a certain type of object, vehicles showing in specific image regions. The second problem encountered is how to label training data that falls into positive (there is a vehicle in the scene) and negative (no vehicles) cases. We must end up with thousands of images, how do we label that?

You will soon realize that vehicles appear at different scales, sizes, orientations, design, their image may consist of just a few pixels, or may cover whole screen. Vehicles will often be occluded: a vehicle passing behind a tree, a man passing in front of the car, or vehicles occluding each other.

An algorithm that recognizes vehicles from ground level might fail when applied on top views of vehicles. We need to capture the slight variations in the data: different views of vehicles, different lightings, weather conditions, shadows, different car producer designs.

The training set acquisition module has the objective of addressing the problem of capturing and labeling the whole training dataset. The proposed method for acquiring the training set has the advantage of gathering images offering meaningful patterns for vehicles that are relative to the scene and locality of our acquisition device and also variations in the data, mentioned above.

For example, if we were to deploy this system in two different locations, one filming vehicles close and frontally, and another filming vehicles from far away and sideways, we would expect that this module to collect front facing pictures of cars, in the first case, and sideways pictures of cars, in the second. Moreover the vehicles in one database will be of similar scale and capture only specific variations of orientation that the scene permits: cars having normal orientation in the direction the road, cars engaged in overtaking.

Figure 4… is the expansion of the Training Set Acquisition module from the Conceptual architecture figure 4…., depicting the internal processing modules and stages performed on the input frame in order to obtain the cropped out vehicles (positive samples) and also negative samples which are saved in the Image Training Set, as the last operation. As you can see, the architecture follows a simple pipeline model, each module taking as input the processed output of the previous module. Let us analyze what are the objectives of each module and what operations and algorithms engaged in fulfilling their purpose.



*Video Camera*

The video camera acts is the acquisition device which captures frames from the observed scene. A discussion is here necessary based on one constraint that we have imposed on our system, namely we have imposed that the mounting point of the camera should be elevated from the ground providing top-views of the passing vehicles.

Because at this stage of processing we are interested in acquiring quality training samples, i.e. pictures in which vehicles are centered under small variations of orientation and scale this constraint helps us with processing because:

* We minimize the occlusion regions between vehicles at projection of the scene onto the image plane
* It increases our efficiency in extracting individual vehicles, acquiring thousands of samples



Figure 4… sample frame

***Frame cropping***

***[7]Thamersoy Thesis***

**Figure 4.1 also demonstrates the perspective effect of projecting the scene onto the image plane, vehicles closer to the camera will appear larger than the ones farther away, another effect is the and also introduces significant occlusions. In [7] the region closer to the camera is called *entrance region*, and in this region occlusions between vehicles are minimized.**

**Also in [7] they automate the process of finding this region by running Canny Edge Detection algorithm to and then a Hough line transform to identify the orientation of the highway.**

**Our approach is different in that we demand the user to select to select with the mouse a region of interest in which he expects to find minimal occlusion between vehicles and in which vehicles appear to have a median scale within their passing through the video. Once the region of interest is selected further processing will be carried on only this region within the frames.**

**The constraints for selecting this region are as follows:**

* **It should minimize the occlusion between vehicles**
* **The vehicles should appear to have a median scale**
* **The height of the region of interest should be slightly bigger than the height of vehicles. This information will be later used to filter the detected blobs.**



**Figure 4…, Selecting the ROI**

***Background Subtractor***

**The Background Subtractor module has the responsibility of learning the background model of the region of interest and output the foreground mask, which is a binary image segmenting moving parts or regions of region of interest from the static regions, the background.**

**In order to learn the background model several frames in the beginning of the video are reserved for the construction of the background model, after which the foreground mask is obtained by differentiating the background model with each incoming frame, additionally the model has to be continuously updated in order for it integrate slow varying changes, like the variation of light.**

**The quality of the foreground mask will affect not only the blob detection later in processing, but also the accuracy of the classifier and because that at this stage of training we are not concerned with real-time processing we choose a high performing algorithm for this task, the MOG (Mixture of Gaussian).**

**In the approach of the Mixture of Gaussian, each pixel history is modeled by a mixture or linear combination of *K* Gaussian distributions, where a pixel history is the sequence of pixel values over the last *t* frames. The probability of observing a pixel value as a background value is given by evaluating the expression:**

**Notations:**

**When updating the pixel history with a new pixel value its *K* Gaussian mixtures are evaluated. If it does not have a high enough value against any Gaussian then the distribution with the lowest cost will be replaced with a new one and also distribution means and covariance matrices are updated.**

**The background model for each pixel is chosen to be composed of *B* most reliable and least variance distributions. When evaluating a new pixel value *X* which gives a high probability within these B distributions then the pixel value is considered to be a background pixel.**

**In this manner the MOG is continually adapting to small changes of lighting and weather. We also observed that the foreground masks resulting from this procedure are of high quality and are robust to noise and very small camera movements which we have experienced. The next figure shows the constructed background model, and extracted foreground mask for two random frames.**







**Figure 4… The first and second columns represent separately the region of interest, background model, and foreground mask from two random frames.**

***Blob Detector***

**Having the foreground mask the Blob Detector’s job is to quickly find the blobs, the connected components on the foreground masks. It does this by applying a contour tracking algorithm on the foreground mask following the border white object pixels.**

**Additionally this component also performs the following tasks:**

* **It discards blobs which have the width, height and area less than predefined thresholds. In our application the thresholds for width and height are 5 and for the area the threshold is 25. This filtering step will most likely small blobs which are not of interest resulting from noise of small object movements, example: leaf movements in trees, birds flying.**
* **It also discards blobs that are on or very close to the region of interest’s border. The reason for applying this filtering step is that later compute a bounding box around the blob and we want this bounding box to be contained in the region of interest. Another reason for this operation is that we do not want for a vehicle to be saved too many times in the Training Set and thus we ignore it when entering and exiting the region of interest.**

***Blob Filtering***

**The Blob Filtering module performs additional filtering functions and lastly saves positive samples containing vehicles and negative samples which are preferred not to contain vehicles in the Training Set. This module’s goal is to make an educated and simple decision about the incoming blobs from the Bob Detector module to decide which of them are masks generated by moving vehicles.**

**The decision to classify the blobs as vehicles is done based on a feature of shape, namely the *circularity* of the *convex hull*. In the first iterations of development this module was missing and we were collecting elongated objects mainly because we were experiencing slight camera movements. Figure 4… shows that street markings were saved as the background subtraction was generating blobs under camera movement.**

E:\Academic\Licence\lic_doc\pics\chapter 4\bad_blob3.png E:\Academic\Licence\lic_doc\pics\chapter 4\bad_blob1.png 

Figure 4… Bad images saved in the absence

of the Blob Filtering Module.

**It was a necessity to use a discriminative feature that would correctly classify cars at this stage, unless we were to manually go over all the generated samples again and hand-pick them.**

**The idea of using the eccentricity of the convex hull is also mentioned in [7]. They argued that a vehicle is roughly a convex object, and the silhouette is very close to its convex hull. When two vehicles tend to occlude themselves the blob containing them generally does not have a convex shape. With this assumption in mind we can further filter passing blobs and discard joint vehicles appearing as one blob.**

**At this stage of processing we are able the robustly filter and discard the blobs which have a high probability of not belonging to the vehicle class. The last function that this module performs is to save the detected blob as a positive sample in the Training Set. This step is performed by centering a cropping rectangle over the center of gravity of the detected blob. The cropping rectangle’s size is computed relative to the size of the region of interest, namely the height of the cropping rectangle in our case is the largest multiple of 8 which is also smaller than the height of the region of interest, and we compute the width based on an empirically selected aspect ratio which in our case is 1. For our tested video a good cropping size for the saved patches was 64x64.**

Figure 4… Detected blobs

Once we have saved the cropped images as positive samples in the Training set, the next question is from where do we obtain the images serving as negative samples. In [8] they propose the idea of saving the negative samples relative to the location of the blob detections. Specifically, they select rectangular patches centered around the corners and edge midpoints.

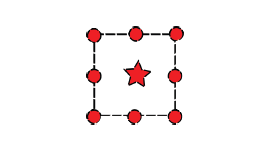


Figure 4…, Source: [8], Relative position of positive and negative training samples.

The bullets are the centers of the negative patches centered placed around the corners and edge midpoints of the detection.

**Another strategy of choosing the negative samples would be to select them randomly over the frame but there is the risk that one of them would end up containing a vehicle in the center which will have a negative impact on the training of every type of classifier.**

**The authors of [8] point out that this approach offers a great advantage for the training of the classifier: it is not just trained with plain and random clutter, but with hard examples generally containing margins and portions of vehicles. Thus the classifier is forced to learn to rule out zones in which it might think that it has detected parts of vehicles, and even filter the regions where vehicles are very close to each other, which is beneficial for accurate localization of vehicles. We also point out that as opposed collecting random negative training samples from different sources or internet databases this method collects samples specific to the scene we are observing thus it further reduce the risks of having false detections from the classifier. Another idea that we propose it to use the background, model obtained in previous stages, by segmenting it in random selected patches which we will also use as negative training samples. The classifier will learn scene specific features that it will easily distinguish as not containing vehicles, and reduce the number of false detections.**

**We show in figure 4… and figure 4… the positive and respectively the negative samples that we end up saving in the Training Set. Experiments have demonstrated that 100% of the captured positive samples are indeed containing a single centered vehicle. We also have to mention that we recorded having centered vehicles in negative samples also, mainly due to the fact that separate vehicle blobs were captured moving close and in the process of saving one of them the other was saved as a negative sample. Nevertheless we did not hand-remove these samples and left the Training Set exactly as it was constructed in the described process of acquisition.**



**Figure 4… Extract of the positive samples training set**



**Figure 4… Extract of negative samples training set**

## 4.2 Feature Extraction and Classifier

**4.2.1. Feature Extraction**

**One of the main challenges in developing a machine learning application concerns the features you use in training the classifier. This decision was taken only after we carried out experiments with different feature extraction methods and studying the results and improvements proposed in other papers.**

It is unavoidable that moving vehicles in the scene will suffer different scale and pose transformations depending perspective projection, their motion direction, elevation of the camera: their image may be as small as just a few pixels, or in close proximity they may be covering a large portion of the screen. The interactions and events that take place even in an outside-door environment are of such high complexity that they cannot be properly understood by the computers in order to convey the semantic meaning.

Often vehicles will often be occluded: a vehicle passing behind a tree, a man passing in front of the car, or vehicles occluding each other. In the domain of tracking occlusion is a very complex problem which has been approached in many paper subjects. The reason for this is that is very hard to identify if an object is occluded by the background and other objects or it is occluding himself with new untracked parts.

As every classification problem the goal is to maximize inter-class variance and minimize intra-class variance. We want to be able to correctly recognize all type of cars, regardless of size, position, scale specific for our scene and also discard any kind of objects falling in other classes like pedestrians.

Our goal is to capture the variations in the data: different views of vehicles, different lightings, weather conditions, shadows, and so on.

Detecting cars is a considerably more difficult problem than textured objects which have simple, semi-rigid structure, where the localization of components does not vary much between samples. Cars have a semi-rigid structure as well, but that structure will vary more between samples, because their shapes and configurations have been designed with product differentiation in mind. Besides the intra-class variations due to color, shape, and ornamentation, which similarly plague face and pedestrian detection systems, there are other issues that complicate car detection. Compared to the other object classes, vehicles also lack texture and have a highly reflective surface. Experimentally we observe, from the background subtraction technique, that the majority of vehicles tend to resemble the color of the pavement in some regions.

In the yearly days of our project we have analyzed the possibility of using the Bag of Words method for extracting a data vector that describes the image sample. In short this method extracts SIFT keypoints/features from the collection of training samples and clusters them to obtain cluster centroids which denote the most relevant features from the whole set. Next an SVM is trained using as feature vectors the frequencies of the most representative keypoints in each of the training samples. The disadvantage of this method is that it lacks locality, the position of the extracted keypoints is discarded, and also we were not obtaining the results that we needed.

At this stage of the process we were inspired by the works of [6] with the formulization of HOG (Histogram of Oriented Gradient) in the context of human detection and [8] with the application in the context of tracking vehicles on a surveillanced highway.

The idea of Histogram of Oriented Gradient descriptors is to represent an image patch by the distribution of gradients directions. A good approach in describing a whole image is to divide in a grid manner where each segmented region is called a cell and computing for the level of each cell a histogram of gradient orientations.

that local object appearance and shape within an image can be described by the

The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell. The combination of these histograms then represents the descriptor. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination or shadowing.

The HOG descriptor maintains a few key advantages over other descriptor methods. Since the HOG descriptor operates on localized cells, the method upholds invariance to geometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. (Wikipedia, Histogram of Oriented Gradients)

Detection

Background Subtraction Top Level

Tracking

* State diagram
* How to merge predictions
* Features
  + NCC
  + LBP