A PCA-based Vehicle Classification Framework

Chengcui Zhang, Xin Chen, Wei-bang Chen
Computer and Information Sciences Department
University of Alabama at Birmingham, Birmingham, AL 35294, U.S.A.
{zhang, chenxin, chenwb}@cis.uab.edu

Abstract

Due to its great practical importance, Intelligent Transportation System has been an active research area in recent years. In this paper, we present a framework that incorporates various aspects of an intelligent transportation system with its ultimate goal being vehicle classification. Given a traffic video sequence, the proposed system first proceeds to segment individual vehicles. Then the extracted vehicle objects are normalized so that all vehicles are aligned along the same direction and measured at the same scale. Following the preprocessing step, two classification algorithms – Eigenvehicle and PCA-SVM, are proposed and implemented to classify vehicle objects into trucks, passenger cars, vans, and pick-ups. These two methods exploit the distinguishing power of Principal Component Analysis (PCA) at different granularities with different learning mechanisms. Experiments are conducted to compare these two methods and the results demonstrate the effectiveness of the proposed framework.

1. Introduction

Traffic surveillance systems rely on a suite of sensors for estimating traffic parameters. Vision-based video monitoring systems offer a number of advantages such as providing vehicle counts, vehicle classifications, illegal U-turns, etc. Vehicle classification, as an important signal processing task, has found widespread military and civilian applications such as intelligent transportation systems. Vehicle classification can be used to compute the percentages of vehicle classes which are often counted manually by human operators and often outdated due to the lack of an automated updating system. The use of an automated vehicle classification system can lead to cost-efficient decision about the thickness of road pavements. In addition, it can provide data about vehicle categories that use a particular street.

There are a large amount of literature on vehicle detection and tracking [7][9][10] which lead to the prototyping and initial development of the important components of an Intelligent Transportation System such as event detection [12] [13] for traffic surveillance system. However, there has been relatively little work done in the field of vehicle classification. This is because it is an inherently hard problem and is often subject to the result of vehicle segmentation. Therefore, most of the existing work is purely dimension based (such as height and length of a vehicle) or shape based.

Gupte et al. [3] proposed a system for vehicle detection and classification. The tracked vehicles are classified into two categories: cars and non-cars. The classification is based on vehicle dimensions and is implemented at a very coarse granularity – it can only differentiate cars from non-cars. The basic idea of [3] is to compute the length and height of a vehicle, according to which a vehicle is classified as a car or non-car. In order to achieve a finer-level classification of vehicles, we need to have a more sophisticated method that can detect the invariable characteristics for each vehicle category considered. Towards this goal, a method is developed by Lai et al. [6]. In their work, they use virtual loop assignment and direction-based estimation methods to identify vehicle types. Through their analysis, each vehicle type is represented by a 1-D signature chart. In the experiment, vehicles are classified into four categories: 7-seat van, fire engine, sedan and motor cycle. The problem of this method is that, as mentioned in the paper, only rough type identification based on vehicle length is possible. Therefore, this method cannot distinguish those vehicles whose lengths are in the same range, e.g. truck and bus. Another problem of this method is that only those vehicles traversing across virtual loops along the road direction can be detected. Therefore, there is a need for a more flexible mechanism that can unveil the real, invariant characteristic of a type of vehicle. In this paper, based on our previous work in automated vehicle segmentation and tracking [2], we proposed an automated framework for vehicle classification using PCA. Part of the proposed work is motivated by the use of Eigenface analysis in the face detection problem.

Eigenfaces [8] are a set of eigenvectors derived from the covariance matrix of the probability distribution of the high-dimensional vector space of possible faces of human beings. These eigenvectors are used in the computer vision society for human face recognition. The Eigenfaces are the invariant characteristics of a human's face. To generate these Eigenfaces, a mechanism called Principal Component Analysis (PCA) is applied.

The similarity between face recognition and vehicle classification lies in the fact that both analyze a 2-D image object and try to find out the invariant features of the image objects within the same class (e.g., to recognize the faces of the same person, to identify vehicles of the same class). Therefore, we adopt the concept of "eigenvehicle" in our vehicle classification framework. Given a set of training data i.e. 2-D vehicle objects of the same vehicle class extracted from traffic surveillance videos, we can generate a set of "eigenvehicles". Again, they are actually a set of eigenvectors. The eigenvector that has the largest eigenvalue is the one that best represents the blend average of this class of vehicles. Other "eigenvehicles" (eigenvectors) correspond to different aspects of the characteristics of this type of vehicles. In the testing phase, each test instance is compared to the "eigenvehicles" of each vehicle type. In this way, vehicles are classified based on their invariant characteristics.

In this work, we go one step further to explore the possible ways of incorporating data mining algorithms in classifying vehicles. More specifically, One-Class Support Vector Machine [11] is used in this method for its known robustness and its ability to deal with high-dimensional data which is an innate characteristic of multimedia data. This data mining based method is implemented as an alternative solution to vehicle classification problem and is compared with the method of straightforward comparison on "eigenvehicles". Since the data mining based classification method also relies on the eigenvectors generated by PCA analysis, we call it PCA-SVM in this paper. Both the PCA-SVM method and the Eigenvehicle method are based on PCA analysis. However, they exploit PCA features in different ways and at different granularities. While Eigenvehicle method exploits eigenvectors for a group of training vehicle objects, PCA-SVM uses PCA to extract the eigenvectors for each vehicle in the training data set. Further, in PCA-SVM, the representative eigenvectors of each vehicle of the same type are used as the training data. In the testing phase of PCA-SVM, each vehicle in the testing data set is tested against all the classifiers. A test vehicle is then classified according to the highest score it gets from each classifier. The comparison of the "Eigenvehicle" method and the "PCA-SVM" method is provided and the results are presented in Section 5.

It is not uncommon that a traffic video sequence contains vehicles driving at various directions. It is especially so for those surveillance videos featuring a major road intersection. For normalization purpose, some data preprocessing work has to be done before the vehicle classification. The preprocessing phase includes vehicle segmentation and orientation adjustment and scaling. Thus the proposed framework is an integrated framework that combines vehicle segmentation, tracking, normalization, and classification. It can automatically track and categorize vehicles within a video sequence.

The Two main contributions of our work are: 1) As far as we know, this is the first proposed system that integrates all stages of traffic video analysis from vehicle extraction and tracking, to vehicle classification. 2) The use of eigenvectors in vehicle classification allows one to see the intrinsic nature of vehicle images and hence classify them at a fine level.

The detailed design and implementations are illustrated in the following order: Section 2 briefly introduces our previous work on vehicle segmentation [2]. Section 3 describes the details of the orientation adjustment and scaling for vehicle objects. Section 4 includes details of the two classification algorithms. Section 5 presents the experimental results. Section 6 concludes the paper.

2. Vehicle extraction

In this paper, an unsupervised segmentation method called the Simultaneous Partition and Class Parameter Estimation (SPCPE) algorithm, coupled with a background learning and subtraction method, is used to identify the vehicle objects in the traffic video sequence [2]. The technique of background learning and subtraction is used to enhance the basic SPCPE algorithm in order to better identify vehicle objects in traffic surveillance videos. For a detailed discussion of this method, please refer to our previous work proposed in [2].

2.1. Vehicle segmentation

The SPCPE (Simultaneous Partition and Class Parameter Estimation) algorithm is an unsupervised object segmentation method to partition video frames [2], [4], [5]. A given class description determines a partition, and vice versa. In the SPCPE algorithm, the partition and the class parameters are treated as random variables. The method for partitioning a video frame starts with an arbitrary partition and employs an iterative algorithm to estimate the partition and the class parameters jointly. Since the successive frames in a video do not differ much, the partitions of adjacent frames do not differ significantly. Each frame is partitioned by using the partition of the previous frame as an initial condition to speed up the convergence rate of the algorithm. A randomly generated initial partition is used for the first few frames during the initial background learning.

As shown in Figures 1(a)-(b), the algorithm starts with an arbitrary partition of the data in the first video frame and computes the corresponding class parameters. Fig. 1(c) shows the intermediate segmentation result for the video frame in Fig. 1(a) while Fig. 1(d) shows the final segmentation result. The following two key steps are executed iteratively till there is no significant change in the class labels for pixels in the image.

1) Class Parameter Estimation: The mathematical description of a class specifies the pixel values within one class as functions of the spatial coordinates of the pixels. The parameters of each class can be computed directly by using a least squares technique [4, 5].

2) Class Partition Estimation: It estimates the best partition as that which maximizes the *a posteriori* probability (MAP) of the partition variable given the image data [4, 5]

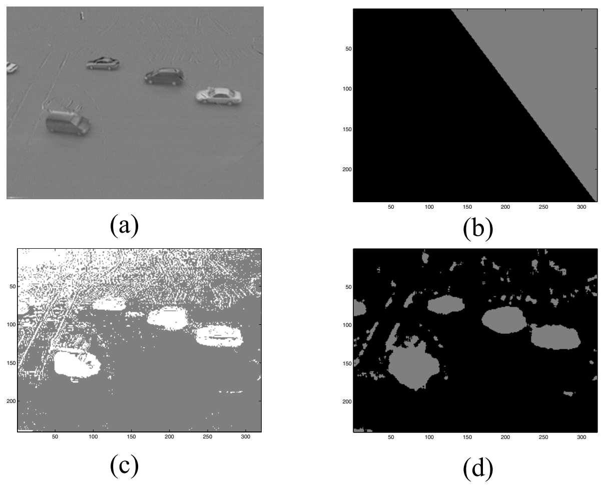

Figure 1. (a) The original traffic video frame; (b) Initial random partition; (c) Intermediate segmentation result; (d) Final segmentation result.

2.2. Background learning and subtraction

It is not uncommon that the non-semantic content (or background) in the traffic video frames is very complex. This is especially true for traffic intersection monitoring. For example, at a traffic intersection, there are non-semantic objects such as the road pavement, trees, and pavement markings/signage in addition to the semantic objects (vehicles), which introduces complications for the segmentation methods. Therefore, an effective way to obtain background information can enable better segmentation results. This leads to the idea of background learning and subtraction.

Background subtraction is a technique to remove non-moving components from a video sequence. The main assumption for its application is that the camera remains stationary. The basic principle is to create a reference frame of the stationary components in the image. Once created, the reference frame is subtracted from any subsequent images. The pixels resulting from new (moving) objects will generate a difference not equal to zero. The traditional way to eliminate background details is to manually select video sequences containing no moving objects and then average them together. As suggested in our previous work [2], instead of manually selecting the suitable frames to generate one reference background image at a time, an adaptive background learning method is used to achieve this goal. The idea is to first use the unsupervised segmentation method together with the segment bounding box information to distinguish the static objects from the mobile objects. Then, these static objects are grouped with the already identified background area to form a new estimation of the background. Detailed discussion can be found in [2].

In the situation of object occlusion, a more sophisticated object tracking algorithm, namely the *backtrack-chain-update split* algorithm, is also given in [2], which can handle the situation of two objects overlapping under certain assumptions (e.g., the two overlapped objects should have similar sizes) and recover their individual spatial locations. In this paper, we will consider only the non-overlapped vehicles and thus isolate the problem of vehicle classification from that of handling vehicle object occlusions.

3. Orientation adjustment and scaling

After a vehicle object is extracted from the traffic video sequence by using the mechanism described in Section 2, we search for the following four control points for that vehicle segment: the left-most point L (L_x, L_y), the top point T (T_x, T_y), the right-most point R (R_x, R_y), and the bottom point (B_x, B_y). They are used to obtain two points P_1 (L_x, T_y) and P_2 (R_x, B_y) as the top left point and the bottom right point of a bounding box for that vehicle segment. An example of a car segment (bounded by dotted line) and its bounding box (dash line) is shown in Figure 2.

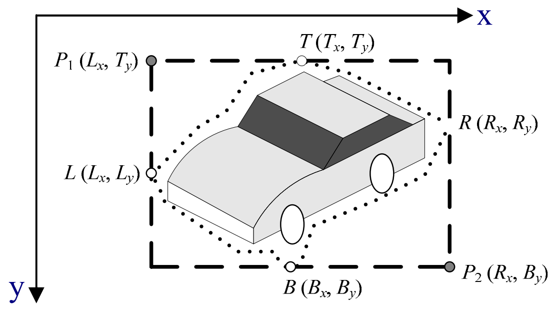

Figure 2. A car segment and its bounding box

However, these video frames taken by a stationary video camera are actually a series of artificial perspective projection images of the real world scenes. The artificial perspective projection image comes with the problem of perspective distortion which depicts the appearance of a closer subject larger than that in the remote scene.

Besides, the perspective projection also causes an orientation distortion problem which makes two parallel objects seem to be oriented in different direction. For example, if we place three identical blocks vertically parallel in front of the camera, we will see the perspective distortion image as shown in Figure 3. The A and B portions which are the front and back half of the middle block illustrate the size change due to the perspective distortion. The A and C portions which are the front-half of the left and middle blocks illustrate the orientation change due to the perspective distortion.

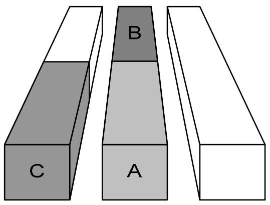

Figure 3. The perspective distortion problem

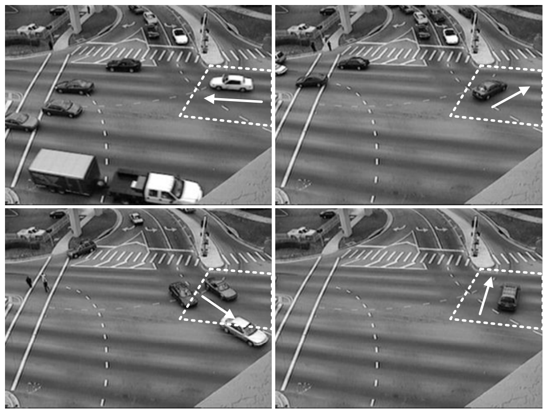

Figure 4. Driving directions in different phases

Since our goal in this paper is to classify vehicles based on their 'looks' in traffic videos, the image distortion problem has to be dealt with by adjusting the vehicle objects to the same direction and the same scale. A transformation model is constructed in this work to correct the image distortion by rotating the vehicle objects to a consistent orientation and scaling the vehicle objects to the same level.

There are two important issues in determining the rotation angle of the subject vehicle - the driving direction of that vehicle and the perspective distortion caused by the camera shooting angle. First of all, the driving direction of the subject vehicle plays a major role in determining the rotation angle. However, it is hard to determine the moving direction of the subject vehicle in a single video frame. To solve this problem, we assume that the driving direction was generally constrained by the lane orientation. Therefore, we can use the lane orientation to represent the driving direction of a car. However, since a road intersection may contain a group of lanes along different directions, it becomes even more complicated when representing driving directions by using lane orientations at the intersection area. Thus, we divide the entire video frames into several phases based on the traffic signal phases. In each phase, there is only one moving direction at any location in the intersection area. For example, in Figure 4, we demonstrate the different driving directions (white arrows) in the same area (dot lines) in different phases. As shown in Figure 4, after the continuous video frames are divided into phases based on traffic signals, there is only one possible driving direction at a given location in each phase, which enables us to use the lane orientation to represent the driving direction of a vehicle.

As also gleaned from Figure 4, the perspective distortion problem caused by the camera makes the width of the lanes narrower at left end and wider at right end. It also makes the orientation of the parallel lanes look 'unparallel' in video frames. In regard to this problem, assuming the slope of the lane in a 2-D video frame is m, the angle θ between the lane and the horizontal line is the arctangent of the slope of the lane.

$$\theta = \tan^{-1} m \tag{1}$$

As mentioned before, for a given vehicle object in a video frame, its main direction is determined by the orientation (slope) of the lane at the position where the vehicle is located. Once the slope m is identified for that vehicle, we can obtain the rotation angle θ and rotate the car to the horizontal position by the angle θ.

Hence, before we can rotate a vehicle object, it has to be figured out in which lane that vehicle is located and to find the slope of the lane at the position of the subject vehicle. To locate on which lane the subject vehicle drives, the boundaries for each lane need to be modeled and the coordinates of the vehicle have to be determined. In regard to this, two linear equations are used to represent the boundaries of a straight lane (solid lines in Figure 5), and one-dimension data interpolation in piecewise polynomial form is used to represent the boundaries of a curve lane (dotted lines in Figure 5). Figure 5 illustrates the boundaries of each lane. The x- and y- coordinates of a vehicle object are represented by $L_x + ½ (R_x – L_x)$ and $T_y + ⅔ (B_y – T_y)$, respectively. It is worth mentioning that we do not use the centroid to represent a vehicle object because it is possible that the coordinates of a centroid of a large vehicle such as a truck or a van may fall into the wrong lane in a 2-D video frame due to the perspective projection problem.

The lane on which the subject vehicle drives on is determined by using the coordinates of the vehicle and the boundary equations of lanes. For example, as illustrated in Figure 5, assume the coordinates of the vehicle are C_x and C_y. If $C_y > L_7(C_x)$ and $C_y < L_8(C_x)$. Then we know the subject vehicle was located on the lane between L_7 and L_8.

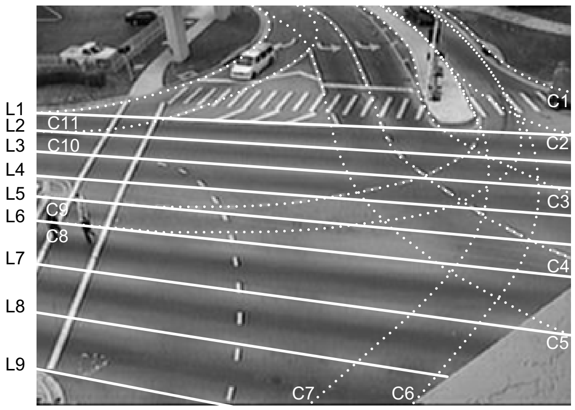

Figure 5. Boundaries of lanes

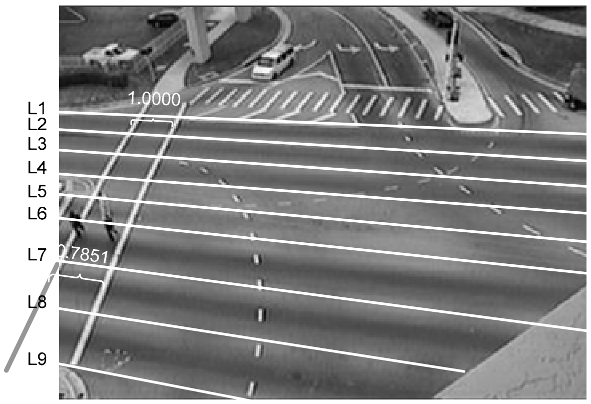

Figure 6. Scaling factors of L_1 and L_7

The lane equation is then derived from its associated two boundary equations. To generate a lane equation, we simply take the sum of the two boundary equations of a lane and then divided it by two. Therefore, each straight lane, such as the horizontal lanes in Figure 5, will be represented by a line equation corresponding to the central line of that lane. For each curved lane like the one bounded by C_6 and C_7 in Figure 5, it will be represented by a 'central' curve which has equal distance from the two boundary curves such as C_6 and C_7. The lane equations are then used to obtain the slope at the location of the subject vehicle represented by its coordinates. More precisely, the first derivative of the lane equation determines the slope m of that lane at any given point. The obtained slope m for the subject vehicle can be further used to compute the rotation angle θ by Equation (1).

In addition to rotation problem, we also need to deal with the scaling problem. First, a reference for computing the scaling factor s needs to be identified. In our case, we took the pedestrian crossing as the scaling reference area which is bounded by two straight lines as shown in the left part of the frame. The width of the pedestrian crossing is used as a reference for deriving scaling factors. We measure the width of the pedestrian crossing along each lane, and define the pedestrian crossing width along the farthest lane (the lane bounded by L_1 and L_2 in Figure 6) as the standard reference. The scaling factor of each lane is then computed by dividing the width of the standard reference by the width of pedestrian crossing along that lane. Thus, the scaling factor of the farthest lane is 1 which means there is no need to scale the vehicle object. In Figure 6, the scaling factor for the lane bounded by L_7 and L_8 is shown. It is worth mentioning that the scaling factor obtained this way is just a rough estimate rather than an accurate calculation of scaling factors. It is the simplicity of this method that makes it a feasible solution in our case.

Once we have both rotation angle $\boldsymbol{\theta}$ and the scaling factor $\boldsymbol{s}$, we can start to build our transformation model. To preserve the collinearity property (i.e., all points lying on a line initially should still lie on the same line after transformation) and the relative distances within the vehicle object, we use the affine transformation to rotate and scale the vehicle object to a new yet comparable state. The affine transformation is defined as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = s \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} \qquad (2)$$

where $\boldsymbol{\theta}$ is the rotation angle and $\boldsymbol{s}$ is the scaling factor. After applying the affine transformation to rotate and scale all the vehicle objects, we make all subject vehicles aligned in the same direction and scaled to the same level of resolutions.

4. Classification

In vehicle classification, we can roughly divide vehicles into two categories: "trucks" (which contain extremely long vehicles such as 18-wheelers and buses) and "cars" (passenger cars, vans, and pick-ups). However, in order to further classify these vehicles, we need to have a method that can categorize them at a finer granularity. In this work, we adopt a hierarchical classification method.

At the very top level, a dimension based method is used to categorize vehicles into truck and car. The dimension information, i.e. length and height, is extracted from the "normalized" vehicle segments by using the method described in Section 3. Since "trucks" and "cars" constitute very distinct types of vehicles in respect to their dimensions, the result of this part of the classification is near perfect. However, we cannot use the same method to further dividing "cars" category into passenger cars, vans and pick-ups. The dimension of a van or a pick-up is only slightly different from that of a passenger car. In some cases, a full size passenger car may be even bigger than a regular-size SUV. Therefore, we need a more sophisticated method that can identify the shape features as well as the invariant characteristics of a vehicle type.

The key component we use for this finer-level vehicle classification is the Principal Component Analysis (PCA). In this paper, PCA is exploited in two ways. The first method is called "Eigenvehicle" method inspired by the well-known "Eigenface" technique used in human face recognition. The second method is called "PCA-SVM" which uses PCA to capture the representative vectors of vehicles and classify them by using One-Class Support Vector Machines (SVM). In the following subsections, these two methods will be illustrated in detail.

4.1. Eigenvehicle

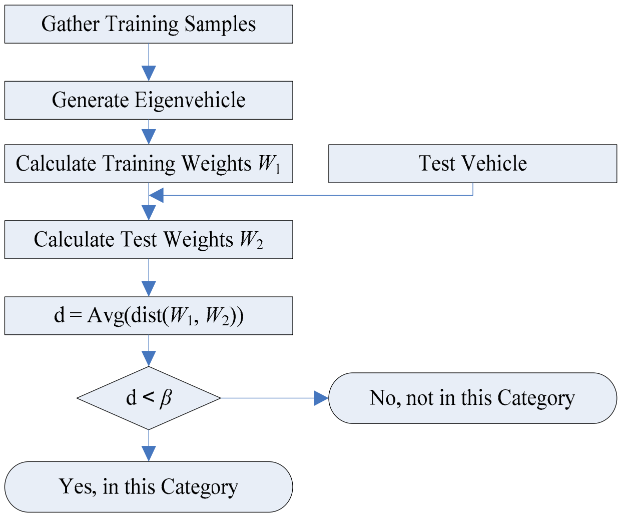

Figure 7. Overview of the Eigenvehicle method

The term "Eigenvehicle" is adopted to name this method because eigenvectors play an important role in this algorithm. PCA is used to generate eigenvectors. Figure 7 is the flow chart of this algorithm. After the pre-processing - vehicle segmentation and normalization (transform and scale), the algorithm goes through the following steps:

(1) Gather Training Vehicle Samples: The bounding boxes of transformed and scaled vehicle segments are first extracted. Since vehicles are located at different parts of the frame, their backgrounds may be different which may cause trouble in vehicle classification. To alleviate this problem, intensities of background pixels are set to 0 (black) and thus they will not be considered in the following steps.

Another factor we need to take into consideration is that the sizes of bounding boxes may be different due to the size differences among vehicles (even with the same vehicle type). This factor could adversely affect the result of our next step – Eigenvehicle Generation. Therefore, we specify a uniform bounding box whose size is the biggest among all training samples. For those whose bounding boxes are smaller, we pad the surrounding areas with zeros. In this way, we can gather a set of training vehicle samples for each type of vehicle. The figure below shows a set of 9 "passenger car" samples.

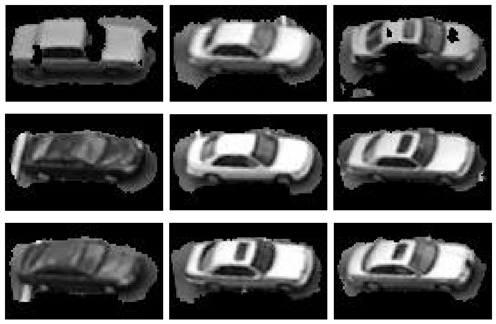

Figure 8. Training samples: Passenger Cars

(2) Eigenvehicle Generation: From the first step, we obtain a set of regularized training vehicle samples. In our experiment, all samples are oriented and scaled in accordance with that of the farthest lane. This choice of downscaling is to keep low frequency data so as to better represent the original data. Since each sample is actually a 2-D image $A_i \in \Re^{m \times n}$, it can be represented as an m by n vector with m being image height and n being image width. We then read A_i in column major order, one pixel at a time, and reshape it as $A_i' \in \Re^{1 \times mn}$. With k being the number of samples in the training set, we can have a matrix of k columns $A' = [\, A_1' \; A_2' \ldots A_k' \,]$. The length of each column is $m \times n$. Then we can compute the mean vector ω as below:

$$\omega = \frac{1}{k}\sum_{i=1}^{k} A_i' \qquad (3)$$

Since ω is a $1 \times mn$ vector, we can restore it into an m by n matrix and output it as an image. The mean "passenger car" of the above mention sample set is shown in Figure 9.

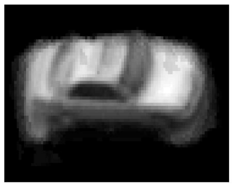

Figure 9. The mean image of passenger Car samples

A' is the set of training samples. Let $\sigma_i = A_i' - \omega$ and $\sigma = [\, \sigma_1 \; \sigma_2 \ldots \sigma_k \,]$. The covariance matrix of A' is:

$$C = \frac{1}{k}\sum_{i=1}^{k}\sigma_i\sigma_i^T = \sigma\sigma^T \quad (4)$$

The principal components are the eigenvectors of *C*. Those eigenvectors that have the biggest associated eigenvalues contribute most to categorizing the training samples. From the above equation, we know that $C \in \Re^{mn\times mn}$. For a typical training set in our experiments, the sizes of *m* and *n* are 92 and 165. This makes the size of *C* 15180×15180, which is not feasible in computing principal components. Turk and Pentland [1] proposed a scheme to solve this problem. Instead of directly using $\sigma\sigma^T$, we find the eigenvectors and eigenvalues of $\sigma^T\sigma$ first.

Suppose u_i is an eigenvector of $\sigma^T\sigma$ and λ_i is the associated eigenvalue. We have:

$$\sigma^T\sigma\, u_i = \lambda_i u_i \Rightarrow \sigma\sigma^T\sigma\, u_i = \lambda_i\, \sigma\, u_i$$

We can see from the above deduction that $\sigma\, u_i$ is an eigenvector of $\sigma\sigma^T$. This trick greatly reduced the computation complexity since the dimension of $\sigma^T\sigma$ is only *k* by *k*. Therefore, we can get he top *k* principal components of $\sigma\sigma^T$ by the following equation.

$$v_i = \sum_{j=1}^{k} u_{ij}\sigma_j \quad (5)$$

The first principal component is the eigenvector that has the biggest eigenvalue. Its length is $m\times n$. As we did before with the mean image, we can reconstruct this vector into a 2-D image. This is what we call "eigenvehicle". Similarly, we can reconstruct an "eigenvehicle" for each eigenvector. It is not necessary to use all eigenvectors to categorize a vehicle type. Instead, we only need to choose the most significant eigenvectors i.e. those who have significant eigenvalues. The following figure shows the top 6 "eigenvehicles" reconstructed from the principal components – eigenvectors of "passenger car" training set. The first "eigenvehicle" is a blend average of all samples in the training set. The rest signifies different aspects of the characteristics of the training set. In computing principal components, the matrix is normalized. Therefore, the entries of the generated eigenvectors need to be rescaled into the normal range of image pixel values in order to reconstruct the image. This is why the backgrounds of the following "eigenvehicle" images shown in Figure 10 are not necessarily in pure black.

(3) Vehicle Classification: The process of classifying a new (unknown) vehicle A_{new} to one of the known vehicle classes proceeds in three steps.

First, reshape A_{new} into A'_{new} and we can obtain $\sigma_{new} = A'_{new} - \omega$.

Second, transform the new vehicle into its "eigenvehicle" components. The resulting weights w_i form the weight vector *W*.

$$w_i = v_i^T\,\sigma_{new} \quad (6)$$

$$W = (w_1, w_2, \ldots, w_k) \quad (7)$$

Third, Euclidean distance between the weight vectors of two images provides a measure of similarity between them. Therefore, the weight vector of the new vehicle is compared with the weight vectors of vehicles in the training set. If the mean distance exceeds some threshold β, we claim that the new vehicle does not belong to this class.

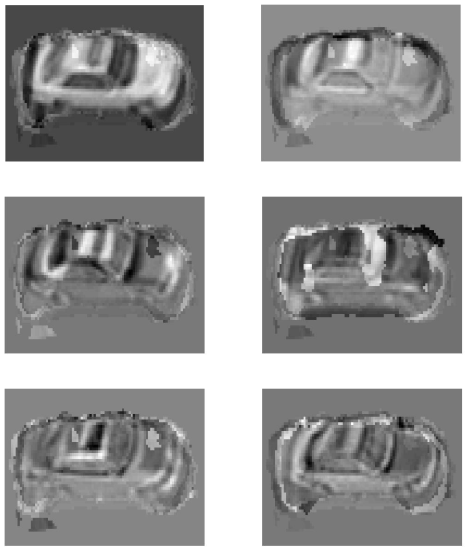

Figure 10. The top six Eigenvehicles of the vehicle type "Passenger Car".

4.2. PCA-SVM

The second method explored in this paper is called "PCA-SVM" which uses PCA to capture the representative vectors of vehicles and classify them by using One-Class Support Vector Machines (SVM). It exploits the distinguishing power of PCA analysis in a different way than the Eigenvehicle method does. The purpose of proposing an alternative here is two-fold: 1) to examine the possible ways of integrating data mining techniques into this kind of application; 2) to exploit the effect of PCA analysis at a finer level towards analyzing the features of individual vehicles rather than analyzing a set of training vehicles all at a time as described in the Eigenvehicle method.

4.2.1. One-class SVM. One-Class classification is a kind of unsupervised learning mechanism. It tries to assess whether a test point is likely to belong to the distribution underlying the training data. In our case, the training set is composed of a set of vehicles of the same class. One-Class SVM has so far been studied in the context of SVMs. The objective is to create a binary-valued function that is positive in those regions of input space where the data predominantly lies and negative elsewhere.

The idea is to model the dense region as a "ball". Ideally, vehicles belonging to that class are inside the "ball" and all the others are outside. If the origin of the "ball" is $\vec{\alpha}$ and the radius is r, a point $\vec{x}_i$ is inside the "ball" iff $\left\|\vec{x}_i - \vec{\alpha}\right\| \le r$. In our case the point is a feature vector of a vehicle.

This "ball" is actually a hyper-sphere. The goal is to keep this hyper-sphere as "pure" as possible and include most of the vehicles that belong to this class. Since this involves a non-linear distribution in the original space, the strategy of Schölkopf's One-Class SVM [11] is first to do a mapping θ to transform the data into a feature space F corresponding to the kernel K:

$$\theta(x_i) \cdot \theta(x_j) \equiv K(x_i, x_j) \tag{8}$$

where x_i and x_j are two data points. In this study, we choose to use Radial Basis Function (RBF) Machine below.

$$K(u,v) = \exp\left(\left\|x_i - x_j\right\| / 2\sigma\right) \tag{9}$$

Mathematically, One-Class SVM solves the following quadratic problem:

$$\min_{w,\xi,\rho} \frac{1}{2}\|w\| - \rho + \frac{1}{\alpha n}\sum_{i=1}^{n} \xi_i \tag{10}$$

subject to

$$(w \cdot \theta(x_i)) \ge \rho - \xi_i, \quad \xi_i \ge 0 \text{ and } i = 1,\ldots,n \tag{11}$$

where ξ_i is the slack variable, and $\alpha \in (0,1)$ is a parameter that corresponds to the ratio of "outliers" in the training dataset. n is the total number of data points in the training set. If w and ρ are a solution to this problem, then the decision function is $f(x) = sign(w \cdot \theta(x) - \rho)$ and it will be 1 for most examples x_i contained in the training set.

4.2.2. Feature extraction. Given the above mentioned One-class SVM model, the key step is to extract features out of each vehicle segment. The features of a vehicle are expected to well represent the characteristics of the vehicle so that it can be distinguished from other vehicles. PCA analysis is chosen for this purpose.

Each vehicle segment can be regarded as a point cloud. Each point corresponds to a pixel of the image. In order to reduce the influence of background, we eliminate the background pixels from the point cloud. Each point has three values: x-coordinate, y-coordinate, and its intensity. With each point represented by a 3-dimensional vector, the point cloud is subject to Principal Component Analysis. Figure 11 below gives an example of point cloud.

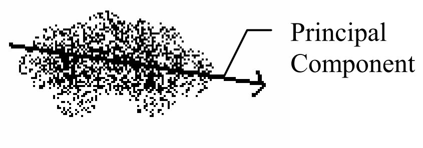

Figure 11. The point cloud with principal component

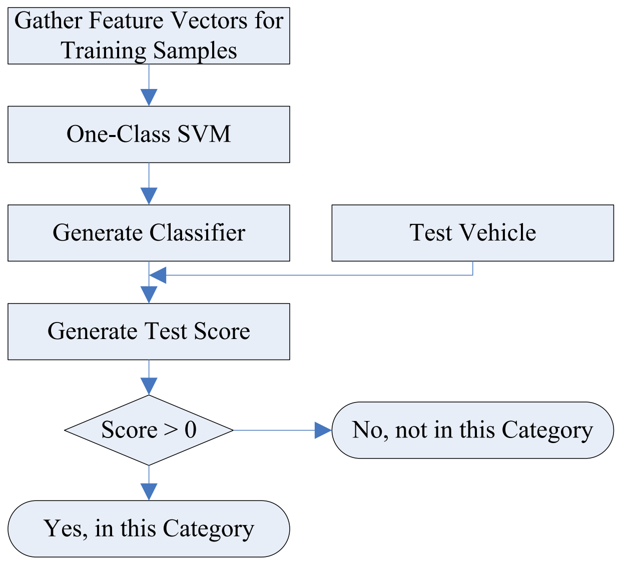

Figure 12. Overview of the PCA-SVM algorithm

For each vehicle segment, only the most significant eigenvector of its point cloud is chosen to represent the vehicle. This eigenvector is the principal component that best describes the distribution of the point cloud. We feed the principal components as feature vectors into One-class SVM for training and testing.

In our experiment, there are three training sets, one for each type of vehicles: passenger car, pick-up, and van. Each type of vehicles is represented by a set of feature vectors and trained by a One-class SVM. Then we use the trained One-class SVM classifier to test on new vehicle to see which class it belongs to. Figure 12 summarizes the algorithm details in Sections 4.2.1 and 4.2.2. As shown in this figure, given a test vehicle object, each classifier will output one score indicating the possibility of that vehicle belonging to that class. The classifier that returns the highest score determines the class label for that vehicle. It has to be noted that in PCA-SVM method, PCA analysis is performed on each vehicle in order to generate a set of eigenvectors as its features, while in Eigenvehicle method PCA analysis is performed on the entire set of training vehicles. The comparison of these two methods and the related discussions are presented in Section 5.

5. Experiments

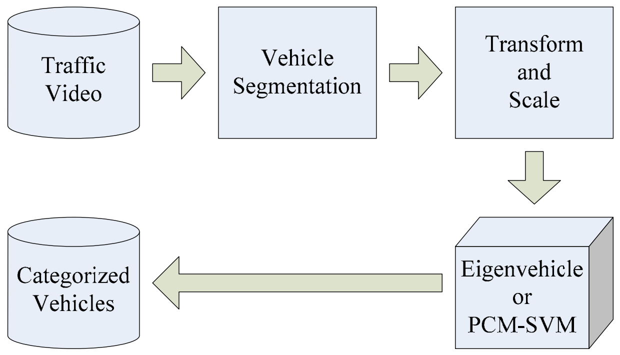

Figure 13. Overall system architecture

By plugging in either Eigenvehicle or PCA-SVM algorithms after preprocessing, we have an integrated system that can automatically extract and classify vehicles in traffic videos. The system flowchart is shown in Figure 13. A real-life traffic video sequence consisting of 2943 frames is used to analyze and compare the performance of the two proposed algorithms on vehicle classification. The video sequence features one major road intersection where vehicles drive at four directions controlled by traffic lights. The frame rate of the video sequence used in our experiments is 5frames/sec.

In the pre-processing phase, all distinct vehicle segments are extracted and form a sample pool. By "distinct", we mean that repetitive appearances of the same vehicle across continuous video frames are counted as only one instance in the training set to avoid bias. In other words, we only include one representative segment/instance for each distinct vehicle in the training set.

In our experiment, three sets of training samples are formed for three categories of vehicles. They are "passenger cars (PC)", "pickup trucks (PK)" and "vans and SUVs (VAN)". In each training set, there are 30 distinct vehicles.

Three sets of test samples are formed with each containing 100 vehicles randomly chosen from the sample pool. Table 1 and Table 2 show the precision and recall values by evaluating the Eigenvehicle method and the PCA-SVM method on all the three test sets, respectively. "Test 1" to "Test 3" correspond to the above mentioned three test sets, respectively. Table 1 is for Eigenvehicle algorithm and Table 2 is for PCA-SVM algorithm.

From the two tables we can see that overall "PCA-SVM" performs slightly better than that of "Eigenvehicle" algorithm. Both of them use eigenvectors but in different ways. The precision of both on "pickup" and "van" categories are relatively low. This is due to the lack of sufficient samples in these two categories: "pickup" and "van". In the sample pool, the ratio of "passenger car" to "pick-up" or "van" is approximately 4:1. That is, the number of "passenger cars" that appear in the video sequence is much more than that of the other types of vehicles. Another reason for the low precision is the imperfection of segmentation results. According to our observations, these three types of vehicles do not differ much with respect to dimensions (i.e. length, height etc.). In addition, the slight differences among these three types of vehicles are further obscured by the imperfect segmentation results. This makes the lack-of-sample problem even worse. For example, the misclassification of one badly-segmented "pick-up" vehicle would lower the classification accuracy significantly.

Table 1 Test results for "Eigenvehicle" algorithm

		Test 1	**Test 2**	**Test 3**
PC	***Recall***	68.3%	73.3%	76.1%
	Precision	64.1%	64.7%	64.3%
PK	***Recall***	50%	90%	85%
	Precision	46.7%	65.4%	55.4%
VAN	***Recall***	75%	50%	50%
	Precision	54.1%	47.8%	41.3%

Table 2 Test results for "PCA-SVM" algorithm

		Test 1	**Test 2**	**Test 3**
PC	***Recall***	75%	75%	70%
	Precision	86.5%	72.6%	63.6%
PK	***Recall***	65%	65%	70%
	Precision	44.1%	46%	53.7%
VAN	***Recall***	85%	70%	75%
	Precision	55.8%	51.2%	55%

To our best knowledge, the system proposed in this paper is the first one that incorporates the overall process of video segmentation, vehicle tracking, data cleaning and vehicle classification in a single framework. Especially, the classification mechanisms proposed in this paper are designed to categorize vehicles into different types by examining the invariant features innate in each vehicle type. Therefore, it allows us to classify vehicles at a much finer granularity. As this is our first work in designing and implementing such a system, there are still some issues such as the segmentation and classification accuracy. In our future work, more data will be collected and tested using the proposed framework. And we also need to reduce the effect of segmentation and the errors introduced by orientation adjustment and scaling. A possible alternative is to use the entire bounding box area instead of using the vehicle segment only. Since the background (road condition) does not vary much within a short period of time, the background noise may not affect the result as much as an inaccurate segmentation result does.

6. Conclusion

In this paper, a PCA-based, integrated vehicle classification framework is proposed which incorporates several key components important to an Intelligent Transportation Surveillance System. First, traffic video sequences are processed to extract vehicle segments, which provides a means for vehicle tracking and classification. Secondly, vehicle segments are properly normalized so that all vehicles are aligned along the same direction and scaled to the same level of resolution. In the classification stage, we proposed and implemented two algorithms "Eigenvehicle" and "PCA-SVM" for classification purposes. Both of them are PCA based. PCA is chosen for its known ability to find innate characteristics of a group of data with similarities. Experimental results demonstrate the capability of the proposed system in classifying vehicles at a finer level. In our future work, more sample data need to be collected and tested using the proposed system. In addition, mechanisms that can alleviate the segmentation problem need to be developed.

7. Acknowledgements

The work of Chengcui Zhang was supported in part by SBE-0245090, the UAB ADVANCE program of the Office for the Advancement of Women in Science and Engineering, and the UAB Faculty Development Award.

8. References

[1] M. Turk and A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuro-science,* Vol. 3, pp. 72-86, March 1991.

[2] S.-C. Chen, M.-L. Shyu, S. Peeta, and C. Zhang, "Learning-Based Spatio-Temporal Vehicle Tracking and Indexing for Transportation Multimedia Database Systems", *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 154-167, September 2003.

[3] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, "Detection and Classification of Vehicles", *IEEE Transactions on Intelligent Transportation Systems,* vol. 3, no. 1, pp. 37-47, March 2002.

[4] S.-C. Chen, S. Sista, M.-L. Shyu, and R. L. Kashyap, "An Indexing and Search Structure for Multimedia Database Systems," *Proc. of IS&T/SPIE Conference on Storage and Retrieval for Media Databases 2000*, San Jose, CA, USA, pp. 262-270, Jan. 2000.

[5] S. Sista and R. L. Kashyap, "Unsupervised Video Segmentation and Object Tracking", *Computers in Industry,* vol. 42, no. 2-3, pp. 127-146, June 2000.

[6] A. H. S. Lai and N. H. C. Yung, "Vehicle-Type Identification through Automated Virtual Loop Assignment and Block-Based Direction-Biased Motion Estimation", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 2, pp. 86-97, June 2000.

[7] K. P. Karmann and A. von Brandt, "Moving Object Recognition Using an Adaptive Background Memory", *Proc. of Time-Varying Image Processing and Moving Object Recognition*, Vol. 2. V. Capellini, Ed., pp. 289-296, 1990.

[8] M. Turk and A. Pentland, "Face recognition using eigenfaces", *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–591. 1991.

[9] Z. Sun, G. Bebis, and R. Miller, "Evolutionary Gabor Filter Optimization with Application on Vehicle Detection", *Proc. of the Third IEEE International Conference on Data Mining (ICDM'03)*, pp. 307-314, 2003.

[10] Z. W. Kim and J. Malik, "Fast Vehicle Detection with Probabilistic Feature Grouping and its Application to Vehicle Tracking", *Proc. of the Ninth IEEE International Conference on Computer Vision (ICCV'03)*, pp. 524-531, 2003.

[11] B. Schölkopf, J. C. Platt et al, "Estimating the Support of a High-dimensional Distribution", *Microsoft Research Corporation Technical Report MSR-TR-99-87*, 1999.

[12] A. Yoneyama, C. H. Yeh, and C.-C. J. Kuo, "Robust Traffic Event Extraction via Content Understanding for Highway Surveillance System", *Proc. of International Conference on Multimedia Expo. (ICME)*, pp. 1679-1682, 2004.

[13] Y.-K. Jung, K.-W. Lee, and Y.-S. Ho, "Content-Based Event Retrieval Using Semantic Scene Interpretation for Automated Traffic Surveillance", *IEEE Transactions on Intelligent Transportation Systems,* Vol. 2, No. 3, pp.151-163, September 2001.