A Method of Vehicle Classification Using Models and Neural Networks

WEI Wu Member, IEEE, ZHANG QiSen, and Wang Mingjun
Institute of Intelligent Transportation System, Department of Traffic Engineering,
Changsha Communications University, Changsha 410076, P. R. China.
Email:weiwu@ieee.org

Abstract

This paper presents a novel method of vehicle classification using parameterized model and neural networks. First, we propose the parameterized model, which can describe features of vehicle. In this model, vertices and their topological structure are regarded as the key features. Then we adopt classifier based on multi-layer perceptron networks (MLPN) to recognize vehicles. In this neural network classifier, learning algorithms based on the gradient descent method for the least exponential function error (LEFE) are adopted. Experimental results show that parameterized model can satisfactorily and effectively describe vehicles, and correct rate for vehicle recognition using neural networks classifier is more than 91%. This novel method can be used in real world systems such as the vehicle verifying system in toll collecting station. However, it is not difficult to adapt algorithms and improve model to fit for other traffic scene.

Keywords: Vehicle Classification, Parameterized Model, Multi-layer Perceptron Networks, Features Extracting of Vehicle in Model and Image, Matching.

1. Introduction

Recently, researchers pay more attention to the technologies of vehicle recognition. Vehicle recognition plays an important role in the fields of road traffic monitoring and management. For example, in the toll collecting system on highway, vehicles should be classified and verified, because the toll collecting depends on the type of vehicles.

The typical method for vehicle recognition is based on modeling technologies. 3D structured models often are used to describe the vehicle shapes and structures[1][2][3][4][5]. In these 3D models, point, lines, faces and topological structures are regarded as the key features to deal with the problems of translating, rotating, scaling and occluding. In the case of road traffic, vehicles are normally confined to be upright in contact with the road, and this ground-plane constraint (GPC) can reduce the pose recovery problem from 6 to 3 degree of freedom[6], which are translation (X, Y) on the ground-plane and rotation (θ) about the vertical axis. In these methods, the vehicles are recognized by searching the pose space and by matching the features of model and the features in image. Unfortunately, the computational cost for pose space searching is expressive.

Base on the 3D structured model, this paper first proposes a method based on parameterized model, which can represent the features of vehicles, and while gives an improved method to reduce the computational cost. Then we present a method based on the neural network to classify vehicles. Finally, the experimental results are presented which show that our method can effectively deal with vehicle recognition, and are satisfy to the requirement of real world system.

The rest of this paper is organized as follows. In Section 2, we present the parameterized model for vehicles. Section 3 investigates the method of neural networks for vehicle classification. Finally, experiments and conclusions are shown in Section 4.

2. The Parameterized Model

2.1 The parameters selection for vehicle models

The underlying geometrical model is based on bilaterally symmetrical, extruded 8-sided polygon. Under the ground-plane constraint (GPC), we can express the vehicle pose and position using only three parameters, noted (X, Y, θ). The model can be conveniently parameterized with respect to model-centered coordinate system (x,y,z), whose yz-plane is the symmetry plane, and whose xy-plane is constrained to lie on the ground plane in the scene coordinate system (XY). The 8 vertices (refer figure 1) in the positive x direction are free to move in (x,y,z), though will be strongly constrained. The vertices in the negative x direction are at corresponding points $(-x, y, z)$. An additional line, corresponding to bottom of the side windows, is also modeled using 2 further parameters, and two semi-decagonal wheel arches are added to the bottom of the vehicles, with equal radius

(*R*) and longitudinal position (*D*). The model therefore has 3+8*3+2=29 structural parameters (see [7] and [8]).

The parameterized model is built and adapted in the model, image projected by geometric model (called image P) and image captured by CCD cameras (called image I).

The parameterized model was fitted to individual images of vehicles in following way. First, the class of vehicle was identified by eye, and a standard instance of this class was onto the ground plane in the image. The pose parameters (X, Y, θ) were then set by eye to provide an approximate initial fit to the image capture by CCD Camera, it is need to compare the model, image P and image I by human. The pose of this fixed shape model was then refined using the passive method. This provided a first estimate of the pose and shape of the vehicles.

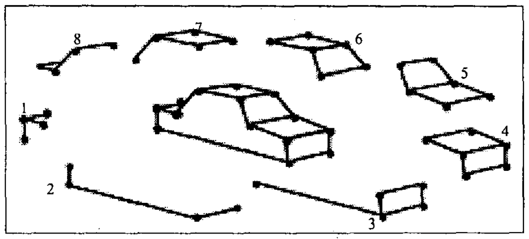

Figure 1: **The sub-models including 8 vertices**

The model was then decomposed into 8 separate sub-models, illustrate in figure 1. Each sub-model comprises two adjacent quadrilaterals, parallel to the axis of extrusion (x) of the model, and is therefore defined by three adjacent vertices pf the polygon. Each sub-model was deformed in turn, by allowing the middle vertex (and its mirror image) to very in (x,y,z), while keeping the four other vertices fixed. Thus the fold line between the two quadrilaterals moved in 3D, while the bilateral symmetry of the sub-model was maintained.

The processes of parameters adapted and updated can be simply described in figure 2, and Figure 3 gives examples of image P and image I.

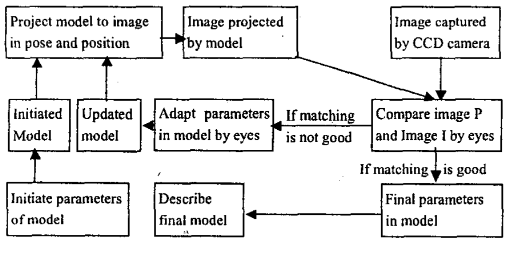
Project model to image in pose and position
Image projected by model
Image captured by CCD camera
Initiated Model
Updated model
Adapt parameters in model by eyes
If matching is not good
Compare image P and Image I by eyes
If matching is good
Initiate parameters of model
Describe final model
Final parameters in model

Figure 2: **The processes of parameters adapted**

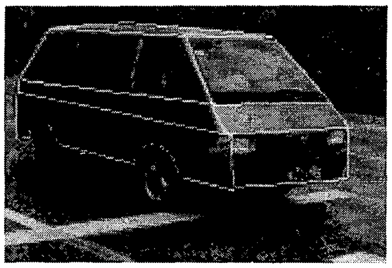

Figure 3a: **Examples of image I**

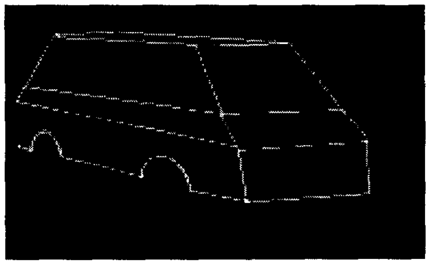

Figure 3b: **Examples of image P**

2.2 The mathematic description of parameterized model

We use a vector S_M to describe parameterized model as expression (1).

$$S_M = (X, Y, \theta, R_M, D_M, x_1^M, y_1^M, z_1^M, x_2^M, y_2^M, z_2^M, \cdots, x_8^M, y_8^M, z_8^M) \qquad (1)$$

Where, (X, Y, θ) represents pose of vehicle model, R_M, D_M is corresponding radius and longitudinal position of vehicle wheel, and (x_i^M, y_i^M, z_i^M) $i = 1,2,\cdots,8$ are the coordinate of vertices.

Likely, we describe the vehicle in projected image (called image P) with a vector S_P, which is the projection from model coordinate system to image coordinate system

$$S_P = (X, Y, \theta, R_P, D_P, x_1^P, y_1^P, z_1^P, x_2^P, y_2^P, z_2^P, \cdots, x_8^P, y_8^P, z_8^P) \qquad (2a)$$

Where, $z_i^P = 0$, $i = 1,2,\cdots,8$.

The vehicle in image capture by CCD camera (called image CC) can be defined by following vector S_I.

$$S_I = (X, Y, \theta, R_I, D_I, x_1^I, y_1^I, z_1^I, x_2^I, y_2^I, z_2^I, \cdots, x_8^I, y_8^I, z_8^I) \qquad (2b)$$

Where, $z_i^I = 0$, $i = 1,2,\cdots,8$.

3. The Vehicle Classification Using Neural Networks

3.1 The feature extracting for neural networks

We select the distance between every two vertices as features for vehicle model and vehicle in image P. Therefore, we can obtain 28 features. These features describe as follows.

$$d_{i,j}^M \quad , i,j = 1,2,3,4,5,6,7,8 ; i < j \tag{3a}$$

$$d_{i,j}^P \quad , i,j = 1,2,3,4,5,6,7,8 ; i < j \tag{3b}$$

There, i,j means the number of vertices for models or vehicle in image P.

These features are valuable to recognition of vehicle. For example, $d_{2,3}^M$ means the proximate value of vehicle bottom plane length, $d_{6,7}^M$ and $d_{7,8}^M$ express the proximate values corresponding length and width of vehicle roof plane.

Likely, we can obtain features for vehicle in image I, whose format is presented in expression (4).

$$d_{k,l}^I \quad , k,l = 1,2,3,4,5,6,7,8 ; k < l \tag{4}$$

There, k,l means the number of vertices sequence for vehicles in image I.

The number of all features are 30, 28 for distance between all vertices, and 3 for radius and longitudinal position.

The recognizing of vehicle is a matching process between vehicle in image P and corresponding vehicle in image I. It is needed to build one-to-one map between features of vehicle in image P and features of vehicle in image. In fact, the map is uncertainty because of different pose and position (X,Y,θ). For a given vertices in image P, it is not easy to find the corresponding vertices in image I, it need to try each vertices in image I in turn. Therefore, for N features, we should deal with N! times matching between image P and image I. Obviously, the computational cost is very expensive. In order to solve this problem, we adopt an effective method to avoid the multi-map relations between features of vehicle in image P and features in image I. The process is described as next parts.

Create Vehicle classes depending on the different (X,Y,θ). For a given vehicle class, if there are k different (X,Y,θ), we regard vehicle in each (X,Y,θ) as a new class vehicle, that is to say, vehicle class A are divided into new sub-classes $A(X_1,Y_1,\theta_1)$, $A(X_2,Y_2,\theta_2)$, ..., $A(X_k,Y_k,\theta_k)$. If k is larger enough, it is easier to build the map between two types images. In most cases, the map is one-to-on map, in other words, the relative relations between vertices of two types of images are fixed, for example, if vertices (2) in image P locate in left and bottom corner, then vertices (2) in image I also locate in left and bottom corner. We store all features and their map relations to a database.

The number of new classes are determined area by range of scene in which vehicle appear. Smaller the ranger is, smaller the k can be selected; accordingly, the number of new classes can be reduced. In real world system, we constrain the ranger to a special small area. For example, in toll collecting station, we can use vehicle-position sensor to confine vehicle to appear in a small windows in which image is captured.

Expected vehicle classes in neural network consist of all sub-classes. For example, if the number of classes is six (car, large carriage, small carriage, large truck, small truck and other vehicle), and k=20, then the number of expected classes is 6*20=120.

3.2 The topologic structure of neural networks

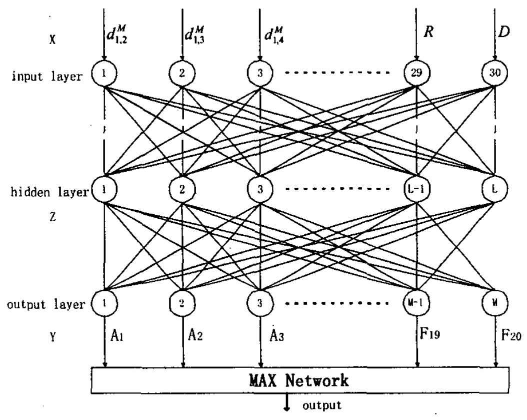

Figure 4: **The topologic structure of neural networks for vehicle classification**

The topologic structure of neural networks for vehicle classification is shown as figure 4. This is a mixture of three-layer multi-layer perceptron networks (MLPN) and a maximum networks (MN). In MLPN, input layer consists of 30 neurons, which represent 30 input features. The number of neurons in hidden layer is N which is determined by a large number of experiment and trial. Output layer reflects the results of classes, and the number of neuron equal the number of all classes (all sub-classes) of vehicle. The MAX network is simple, and just finds the max unit from the output vector by calculating the max value in the output vector, while its output is the classified results of vehicles.

3.3 Input and output vector in neural networks

In MLPN, input vector consists of the features in image, which are shown in expression (5), and output vector is presented in expression (6) whose one unit outstand expected class. In expression (6), A, B, C, D, E, F represent corresponding car, large carriage, small carriage, large truck, small truck and other vehicle. If more classes need to be classified, the other classes such as $G, H, \cdots, P$ will be added.

$$\begin{aligned} X &= [x_1, x_2, \cdots, x_{29}, x_{30}] \\ &= [d^l_{1,2}, d^l_{1,3}, \cdots, d^l_{1,8}, \cdots, d^l_{2,3}, d^l_{2,4}, \cdots, d^l_{7,8}, R, D] \end{aligned} \tag{5}$$

$$\begin{aligned} Y &= [y_1, y_2, \cdots, y_{119}, y_{120}] = [A_1, \cdots, A_{20}, B_1, \cdots \\ &\quad , B_{20}, C_1, \cdots, C_{20}, D_1, \cdots, D_{20}, F_1, \cdots, F_{20}] \end{aligned} \tag{6}$$

3.4 The algorithm of neural network for classification

For MLPN networks, we present an improved BP algorithm. This algorithm use learning rule based on the gradient descent method of the least exponential function error (LEFE) rather than the least square error (LSE). Exponential function error is shown in equation (7), where E is a function of square error. Experimental results show that if we can choose fit parameters a and b, converge of LEFE algorithm is better than LSE algorithms.

$$J = a^{bE} \quad a > 0, a \neq 1, b > 0 \tag{7}$$

We use LEFE to make converging speed for neural networks faster, and in most cases, avoid exponential function error ending up in a local minimum. We present the explanation in detail as follows.

For the algorithm of (LSE), suppose: w is the weighted vector in neural network for algorithm of (LSE). The differential equation of error function to vector w, which is used to update weighted vector is defined as

$$\dot{w} = -\mu \frac{\partial E}{\partial w_i} \tag{8}$$

It is not difficult to obtain expression (9)

$$\begin{aligned} \dot{E} &= \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_P}\right] [\dot{w}_1, \dot{w}_2, \cdots, \dot{w}_P]^T \\ &= -\mu \sum_{i=1}^{P} \left(\frac{\partial E}{\partial w_i}\right)^2 \end{aligned} \tag{9}$$

Where, P is the number of the units in weighted vector.

Likely, for the algorithm of LEFE, the differential equation of error function to vector w can be defined as expression (10).

$$\dot{w}_i^J = -\mu \frac{\partial J}{\partial w_i} = -\mu a^{bE} b \ln a \frac{\partial E}{\partial w_i} = \gamma \dot{w}_i \tag{10}$$

Where, $\gamma = a^{bE} b \ln a$ is a "jumping" coefficient.

It is easy to derivate the expressions (11) and (12).

$$\dot{E}^J = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_P}\right] [\dot{w}_1^e, \dot{w}_2^e, \cdots, \dot{w}_P^e]^T = \gamma \dot{E} \tag{11}$$

$$\dot{E}^J - \dot{E} = (\gamma - 1)\dot{E} = -\mu(\gamma - 1) \sum_{i=1}^{P} \left(\frac{\partial E}{\partial w_i}\right)^2 \tag{12}$$

Obviously, if "jumping" coefficient $\gamma \geq 1$ 时, $\dot{E}^J - \dot{E} \leq 0$, then converging speed for neural networks is faster. In the first stage of learning for neural network, E is large and corresponding γ is large, so fasten the error E decent, however, it is easy to jump out the local minimum. During the last stage of learning for neural networks, E is small and corresponding γ is small, it can ensure the stability of converging process, and avoid the surging.

Frequently, for MLPN networks traditional algorithm is back-propagated (BP) error algorithm, and the learning rule based on the stochastic gradient decent method is adopted. In each iteration, gradient values are influenced by noises of the examples. To reduce influence of noise, it is necessary to adopt batch-processing method in which gradient values of several examples are averaged to obtain estimated gradient values. But, when the number of examples is very large, this method increases cost of calculation in each iteration, and discards the difference among individual examples and reduces the sensibility of learning.

To solve preview problems, this paper presents a method that includes several steps. First, training examples are divided into several groups. Then, all groups learn in turn. After previous group complete learning, final weights of previous group act as the initial weights of the next group. When all groups complete learning, then the whole learning is completed if the final error fits to requirement. Otherwise, the cycle of learning is repeated until the final error fits to requirement. Experimental results show that these problems are effectively solved

3.5 The training and testing samples of neural network for classification

Training samples should be selected when parameterized models are built. It is needed to deal with training samples under human monitoring. First, we specify one-to-one relations between features of models and features in image by eyes observing. Then, features of training samples input to database.

In order to ensure the correct rate of classification, when training samples are selected, the following requirement should be considered.

1) Ensure training samples enough;
2) Consider of tiny differences between vehicles for the same class. For example cars from different manufactory maybe have tiny difference in shape;
3) When the number of classes is increase, select training sample according to requirement of classification, it is needed to add new training samples.

Test sample come from image, and are used to evaluate performance of neural networks classifier. We should collect enough test samples to check the algorithm and evaluate performance of classifier.

4. Experiments and Conclusions

Our experimental data come from real vehicle and the image captured by CCD camera installed in toll collecting station. We select 500 frames, which consider a larger number of poses, positions and different vehicle classes. All features of training samples are stored in a database. 300 test samples captured randomly by CCD camera are utilized to check performance of classifier after finishing train samples learning.

Table 1: **The experimental results**

Classes	TS	CC	EC	RC
Car	50	44	3	3
Small carriage	50	45	2	3
Larger carriage	50	47	1	2
Small truck	50	45	3	2
Large truck	50	48	1	1
Other vehicle	50	45	2	3

Table 1 shows our experimental results. In table 1, TS, CC, EC, and RC are, respectively, Test Samples, Correct Classification, Error Classification, and Refused classification. Observing the table 1, it is not difficult to conclude that the correct rate of classification more than 91%, the error rate is about 4% and about 5% vehicle are refused to classify because of effects of vehicle occultation and shadow. Training of neural network can be processed out-line. After finishing training, recognition of neural networks is carried through on-line. The recognition algorithm runs on PC computer (CPU is PIII-667, RAM 128M). Running results show that the average recognizing time for one vehicle is less than 0.4 s. Obviously, the computation cost is very small, and it is satisfied with requirement of real world system.

It is mentioned that our results are obtained under following conditions: the image quantity is good and algorithm of image processing is efficient, in addition the range is small in which vehicles appear.

We are doing further works to improve our classification system. The future work include: 1) Enlarger the range which vehicles appear so that this system can be widely used in different scene; 2) collect more and better training samples and test samples; 3) Increase parameters in models and more accurately describe models; 4) add more some good features; 5) improve algorithms for classification.

References

[1] T. F. Cootes, D. H. Cooper, C. J. Taylor and J. A. Graham, Training Models of Shape from Sets of Examples, Proc. 3rd British Machine Vision Conference, 9-18, 1992.

[2] A. D. Worrall, G D. Sullivan and K. D. Baker. Advances in Model-Based Traffic Vision, Proc. 4th British Machine Vision Conference, 559-568, 1993.

[3] A. D. Worrall, K. D. Baker and G D. Sullivan. Model-based perspective inversion, Image and Vision computing Journal, 7(1):17-23, 1989.

[4] T. N. Tan, G D. Sullivan and K. D. Baker. Fast Vehicle Localization and Recognition Without Line Extraction and Matching, Proc. 5th British Machine Vision Conference, 85-94, 1994.

[5] T. F. Cootes, D. H. Cooper, C. J. Taylor and J. A. Graham, A Trainable Method of Parametric Shape Description, Proc. 2nd British Machine Vision Conference, 54-61, 1991.

[6] G D. Sullivan. Model-based Vision for Traffic Scenes using the Ground-plane Constraint, Real-time Computer Vision, Eds: C Brown and D Terzopoulos, CUP, 1995.

[7] J. M. Ferryman, A. D. Worrall, G.D. Sullivan and K.D. Baker. A Generic Deformable Model for Vehicle Recognition, Proc.6th British Machine Vision Conference, 1995, 1: 127-136.

[8] A. Lanities, C. J. Taylor and T. F. An Automatic Face Identification System Using Flexible Appearance Model, Proc. 5th British Machine Vision Conference, 65-74, 1994.