Fast Vehicle Detection with Probabilistic Feature Grouping and its Application to Vehicle Tracking

ZuWhan Kim and Jitendra Malik
Computer Science Division
University of Berkeley, CA, USA
{zuwhan,malik}@cs.berkeley.edu

Abstract

Generating vehicle trajectories from video data is an important application of ITS (Intelligent Transportation Systems). We introduce a new tracking approach which uses model-based 3-D vehicle detection and description algorithm. Our vehicle detection and description algorithm is based on a probabilistic line feature grouping, and it is faster (by up to an order of magnitude) and more flexible than previous image-based algorithms. We present the system implementation and the vehicle detection and tracking results.

1. Introduction

Generating vehicle trajectories from video data is an important application of ITS (Intelligent Transportation Systems). Vehicle trajectories are used in analyzing traffic flow parameters for ATMIS (Advanced Transportation Management & Information Systems). In particular, we are interested in the application to evaluating existing driver behavior models or finding a new one. Since the 1950s, researchers from many different fields have proposed well over a hundred traffic models, [13], [11], [1]. However, it has been difficult to evaluate those models due to the lack of real data (vehicle trajectories).

To model drivers behaviors well it is important to have accurate information about inter-vehicle spacing and vehicle trajectory. Previous video-based approaches, [9], [3], were suitable enough for counting the number of vehicles or finding general traffic flows. However, they were either restricted to being used under favorable lighting and traffic conditions [9] or did not give accurate location and dimension of vehicles [3]. We present an approach which works well under various lighting conditions and provides high quality trajectories. These are critical ingredients in developing good models of traffic flow.

Other ways of generating traffic flow parameters are to use loop detectors or instrumented vehicles. Loop detectors show very good detection performance, but they only give limited information because they are point detectors (no trajectories can be obtained). Instrumented vehicles generate long trajectories with detailed maneuvering parameters. However, given a travel, we only get a single trajectory of a single driver (and some additional information on the leading or following vehicles). Therefore, only trajectories from a small number of drivers (who are aware of the experiments) are available, which may bring in a bias to the result. On the contrary, a huge number of trajectories (much less biased, but shorter) can be obtained from video data. Also, the cost of the data collection is significantly lower than those of other methods. We expect that our work will provide good complementary data of other methods.

(a) (b) (c)

Figure 1. Example images from three cameras taken at the same time.

Example images from our video data are shown in Figure 1. The images were obtained from 3 cameras installed on the roof of a 30-story building alongside a freeway. There are small overlapping regions between the cameras covering nearby regions. The nearest image, Figure 1c, is close to a nadir (top) view while the farthest one, Figure 1a, is very oblique.

It is difficult to generate reliable vehicle trajectories from such video data because:

- The tracking targets (vehicles) vary in size, shape, and color.
- The video data includes various times and weather

Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003) 2-Volume Set
0-7695-1950-4/03 $17.00 © 2003 IEEE

conditions. Thus, the illumination conditions, such as the direction of the shadows, vary.

- Vehicles can be occluded by other vehicles or structures.
- Traffic conditions vary, and many of the tracking algorithms degrade with heavy traffic congestion, where vehicle moves slowly, and the distances between vehicles are small. In this case, motion-based vehicle detection and background extraction are difficult and it is also hard to separate nearby vehicles.

In this paper, we present a new vehicle tracking approach which is based on a model-based vehicle detection algorithm. Our vehicle detection algorithm is based on line features, and it is faster than previous image-based algorithms. It is also flexible (from scales and viewing angles) because our model is based on the probabilistic density functions (PDF's) of the 3-D distances between lines. In addition, it gives 3-D description of the vehicles which can possibly be used for the vehicle classification in the future.

In Section 2, we present related work and overview our approach. The vehicle detection and description algorithm is shown in Section 3, and other implementation issues including the tracking algorithm is presented in Section 4. In Section 5, we show detection and tracking results. Finally, we present the conclusion and future work in Section 6.

2. Related Work and Our Approach

2.1. Driver Behavior Models

Driver behavior models explain driver's maneuvers, such as accelerating, decelerating, and changing lanes, with respect to the environmental variables, such as the current speed, the distance to the leading vehicle, and the orientation and destination. For example, car-following models explain following vehicle's (follower) speed change with respect to follower's speed, leading vehicle's (leader) speed, and the distance between two vehicles. Following equation is an example car following model:

$$a(t) = v^m(t)\frac{\Delta\dot{x}(t-T)}{\Delta x^l(t-T)}$$

where $v(t)$ and $a(t)$ are follower's speed and acceleration at time t, and $\Delta x(t-T)$ is the distance between two vehicles at time $t-T$ (T is the *reaction time*). Many different parameters for m, l, and T have been suggested from various experiments. Our goal is to generate useful trajectories to assess or generate these parameters.

Note that we need trajectories in pairs. Therefore, in this application, detection rate is less important than, for example, the accuracy of the trajectories, because we only use pairs of trajectories of two nearby detected vehicles. However, higher detection rate is still required because it provides more pairs of trajectories.

Another observation is that the distance between vehicles, to be precise the distance between the rear end of the leader and the front end of the follower, is an important parameter. Therefore, good localization on the vehicle positions and correct estimation of the vehicle dimensions (lengths) are also required.

2.2. Previous Work

There are two well-known vehicle tracking approaches. The first one, [9], [5], is the *background subtraction algorithm*. In this approach, the background is dynamically estimated from incoming images, and the difference between the current and the background images is thresholded to form "blobs" corresponding to vehicles. This algorithm gives reliable vehicle detection given a favorable illumination condition and a camera angle.

However, the performance of the background subtraction algorithms significantly degrades in the presence of heavy shadows. It is difficult to separate a shadow from the vehicle because the shadow moves along with the vehicle. Also, often, an occlusion or a shadow cast on nearby vehicles makes the separation between vehicles difficult. In addition, the background estimation performance is degraded when the traffic is heavy because the movements of the vehicles are small and a significant part of the background is not observable.

The other approach, [3], uses corner features. In this approach, individually extracted and tracked corner features are grouped based on the proximity of their positions and the similarity of the motion. This approach gives good detection even with less favorable illumination conditions. However, it still has several limitations:

- The location and the dimension of a detected vehicle may not be accurate because they are estimated from the corner features which do not cover the whole vehicle (moreover, some of them may belong to the shadow).
- The position error caused by missing features (tracking failures) may introduce a significant error in the velocity estimation.
- The feature grouping is based on only the locations and the motions of corner features. Thus, there are times that features of nearby vehicles (of the same speed) are grouped together, or the features of a large vehicle (for example, trailer trucks) are not grouped together.

2.3. Our Approach

We introduce a new approach based on a 3D vehicle detection and description algorithm. We first detect vehicles at the entrance area (where the viewing angle is favorable), and track the detected vehicles based on their intensity profiles. Although our application does not require realtime processing, fast computation is still important because we need to process a huge volume of video data. Most of the previous vehicle detection algorithms (whether they are based on template matching or not) work on the intensity image pixels directly, and requires significant processing time. In the next section, we present a fast algorithm which uses the line features. For robust detection, we apply probabilistic reasoning. We also present a dynamic programming algorithm for fast reasoning.

3. 3D Vehicle Detection and Description with Probabilistic Feature Grouping

In this section, we introduce a model-based car detection and description algorithm. Vehicle detection, description, and/or recognition have been an active research area [8], [16], [12], [14], [2], [7], [17]. Early model-based approaches, [8], [16], have been focused on generic pose estimation with pre-defined shape. Our model is more flexible because it is based on probability distributions.

Other approaches have been focused on high resolution ground views, [14], [2], or uses the background subtraction algorithm, [7]. Most of these approaches require a large amount of computation, except the background subtraction algorithms which suffer from shadow and traffic congestion.

In [12], Rajagopalan *et al.* presented vehicle detection algorithm based on higher order image statistics. However, it requires too much computation while the detection rate (73% with 14% false alarm) was not satisfactory (although the experiment was performed on relatively complex scenes). In addition, they have focused on the detection algorithm and the localization performance may not be as good. In [17], Zhao and Nevatia presented an algorithm of detecting cars from aerial images by examining their rectangular shape, front and rear windshields, and shadow. It showed good detection rate (about 90% with 5% false alarm) with less computation (about 30 secs plus preprocessing for 1000×870 image on a PII 400 MHZ), but it still requires image-based comparison and the amount of the computation is still large for our application. In addition, the detection is based on the rectangular boundaries, which is only applied for aerial images.

In this section, we present a faster and more flexible algorithm, which gives 3D structures (descriptions) of the detected cars at the same time. Our algorithm uses line features. Example line features (horizontal and vertical) are shown in Figure 2. Given the orientation of the cars (which is known) the line feature extraction algorithm is as follows: 1) apply a 2-D oriented edge detectors (horizontal and vertical, separately), 2) apply the *non-maxima suppression algorithm* [4] on the given orientation, and 3) perform the *connected-component analysis* for line grouping.

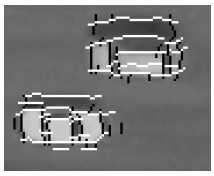

Figure 2. The horizontal and vertical line features used in the algorithm.

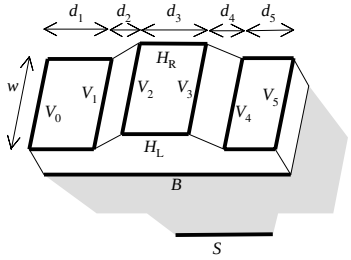

Figure 3. The car model. H_L and H_R are the left and right lines, $V_0, \ldots, V_1$ are the vertical lines, B is the base line, and S is the shadow line. The probability density functions of the configuration parameters (eg. w and d_i in the world coordinates) are estimated from examples and used for the detection and description.

Figure 3 shows our car model. Once line features are extracted, we fit the car model to them. We assume that the front line (V_0), the rear line (V_5), the left line (H_L), and the right line (H_R) of a car are always detected. Our algorithm is applied for each vertical line feature, v_0, which we assume to be V_0. We then gather near-by horizontal line features assuming maximum car width and length. For each pair of horizontal line features, l, and r (which we assume to be H_L and H_R), we gather vertical line features, $v_1, \ldots, v_n$ (ordered from front to rear), and apply a dynamic programming algorithm to find the configuration among them.

In the following equations, we will use $w_{i,j}$ as a shorthand for $V_i = v_j$, $w_{0 \neq 0}$ for $V_0 \neq v_0$, and $P(l, r)$ for $P(H_L = l, H_R = r)$. Our goals are:

Detection: for each v_0, estimate $P(w_{0,0}|\mathbf{E})$ where $\mathbf{E}$ is all the evidence we gather and use.

Description: find l, r, m_0, ..., m_5 which maximize $P(l, r, w_{1,m_1}, \ldots, w_{5,m_5}|w_{0,0}, \mathbf{E})$.

We estimate $P(w_{0,0}|\mathbf{E})$ by summing up all the possible configurations of H_L, H_R, and V_i because our model requires evidence derived from an unknown assignment of these variables, such as the width of the vehicle and the distances between the vertical lines.

$$P(w_{0,0}|\mathbf{E}) = \sum P(w_{0,0}, l, r, w_{1,m_1}, \ldots, w_{5,m_5}|\mathbf{E})$$

where

$$P(w_{0,0}, l, r, w_{1,m_1}, \ldots, w_{5,m_5}|\mathbf{E}) = \frac{P(w_{1,m_1}, \ldots, w_{5,m_5}|w_{0,0}, l, r, \mathbf{E})P(\mathbf{E}|l, r, w_{0,0})P(l, r|w_{0,0})P(w_{0,0})}{P(\mathbf{E})}$$

We assume that $P(w_{0,0})$ is uniform (over all the possible v_0) as well as $P(l, r|w_{0,0})$ (which is $1/|H_L \times H_R|$). In the following subsections, we describe how we estimate $P(\mathbf{E}|l, r, w_{0,0})$ and $P(w_{1,m_1}, \ldots, w_{5,m_5}|w_{0,0}, l, r, \mathbf{E})$.

Unfortunately, $P(\mathbf{E})$ is difficult to estimate. A typical approach to handle this problem is to use $P(w_{0,0}|\mathbf{E})/P(w_{0\neq 0}|\mathbf{E})$ instead of $P(w_{0,0}|\mathbf{E})$. In this case,

$$\frac{P(w_{0,0}|\mathbf{E})}{P(w_{0\neq 0}|\mathbf{E})} = \frac{P(\mathbf{E}|w_{0,0})P(w_{0,0})}{P(\mathbf{E}|w_{0\neq 0})P(w_{0\neq 0})}.$$

However, in most of the feature grouping problems, $P(\mathbf{E}|w_{0\neq 0})$ is also difficult to estimate. Fergus *et al.* uses this approach in [6], but it is not clear how they handle this problem (or they use unrealistic assumptions). Therefore, with limited choices, we use an assumption that $P(\mathbf{E})$ is uniform over v_0. Then, $P(w_{0,0}|\mathbf{E}) \propto P(w_{0,0}|\mathbf{E})P(\mathbf{E}) = P(w_{0,0}, \mathbf{E})$.

3.1 $P(\mathbf{E}|l, r, w_{0,0})$

We use the distance (in the world coordinates) between l and r and the existence of the shadow and the base lines (S and B of Figure 3) to estimate $P(\mathbf{E}|l, r, w_{0,0})$. In addition, we use a gradient sign of l or r according to the shadow location. For example, when the shadow is cast on the left side of the vehicle, the left side of l will be darker than its right side.

In principle, the sun angle can be estimated from the approximate position of the vehicle (on the earth) and the time and the date. When such information is not available, we can estimate the angle of the Sun from the direction and the length of shadow cast. We have implemented the second approach which is good enough to analyze a video sequence of 10 minutes or shorter.

3.2 $\sum P(w_{1,m_1}, \ldots, w_{5,m_5}|w_{0,0}, l, r, \mathbf{E})$

For simplicity, we define $\Phi \equiv \{w_{0,0}, l, r\} \cup \mathbf{E}$. It is much harder to estimate $\sum P(w_{1,m_1}, \ldots, w_{5,m_5}|\Phi)$ because of its complexity. Although there are only 5 lines that we need to configure, we still have $P_{5,n}$ different assignments (multiplied by $|H_L| \times |H_R| \times |V_0|$, altogether) when n is the number of the vertical line features. In addition, we need to deal with at least six dimensional joint probabilities which is huge enough to result serious overfitting. In this section, we present a dynamic programming algorithm to estimate this value efficiently.

The evidence features we use are 1) the distances between lines, 2) the gradient changes at the lines, 3) the sampled intensity levels between lines, and 4) the length (coverage) of the lines, The distances are estimated in the world coordinate. For this, we assume that the height of the car is fixed (1.4 meter for the passenger cars) and the heights of the front hood and the trunk of the car are 1 meters each. Then we back-project the (center) positions of the line features to 3-D coordinate.

The gradient changes on the edge of the windshields are useful cues to detect cars [17]. As in [17], we assume that, for most of the bright cars, the windshields are usually darker than the car frame, and for most of the dark cars, the windshields are brighter than the car. In other words, with a high probability p, the gradient directions of V_1 and V_2 (or V_3 and V_4) are opposite to each other while those of V_1 and V_3 (or V_2 and V_4) are the same.

To reduce the computational complexity of the problem, we apply a dynamic programming algorithm based on following Markov-style assumptions:

$$\begin{aligned}
&P(w_{i,j}|w_{i-1,k}, v_{i-2}, \ldots, v_1, \Phi) = P(w_{i,j}|w_{i-1,k}, \Phi),\\
&P(w_{i,j}|w_{i-1,\phi}, w_{i-2,k}, v_{i-3}, \ldots, v_1, \Phi)\\
&\quad = P(w_{i,j}|w_{i-1,\phi}, w_{i-2,k}, \Phi), \text{ and}\\
&P(w_{i,j}|\Phi) = \sum P(w_{i,j}, v_{i-1}, \ldots, v_1|\Phi),
\end{aligned} \tag{1}$$

where $w_{i,\phi}$ is the probability of V_i being missing. The first and the third assumptions are typical Markov assumptions, and the second one extends them to a case of missing features.

Then,

$$\begin{aligned}
&\sum P(w_{i,j}, w_{i-1,m_{i-1}}, \ldots, w_{1,m_1}|\Phi)\\
&= \sum_{k<j} P(w_{i,j}|w_{i-1,k}, \Phi)P(w_{i-1,k}|\Phi)\\
&+ \sum_{k<j} P(w_{i,j}|w_{i-1,\phi}, w_{i-2,k}, \Phi)P(w_{i-1,\phi}|w_{i-2,k}, \Phi)P(w_{i-2,k}|\Phi)\\
&+ \ldots,
\end{aligned} \tag{2}$$

where the parameters can be obtained as follows:

$$P(w_{i,j}|w_{i-1,k},\Phi) = P(w_{i,j}|w_{i-1,k},l,r,\mathbf{E}) = \frac{P(\mathbf{E}|w_{i,j},w_{i-1,k},l,r)(1-P(w_{i,\phi}|w_{i-1,k},l,r,\mathbf{E}))}{\sum_{j'>k} P(\mathbf{E}|w_{i,j'},w_{i-1,k},l,r)}$$
$$P(w_{i,j}|w_{i-1,\phi},w_{i-2,k},l,r,\mathbf{E}) = \frac{P(\mathbf{E}|w_{i,j},w_{i-1,\phi},w_{i-2,k},l,r)(1-P(w_{i,\phi}|w_{i-1,\phi},w_{i-2,k},\mathbf{E},l,r))}{\sum_{j'>k} P(\mathbf{E}|w_{i,j'},w_{i-1,\phi},w_{i-2,k},l,r)}$$
$$\ldots \tag{3}$$

Our implementation allows at most two consecutive missing features (*i.e.* $P(w_{i,j}|w_{i-1,\phi},w_{i-2,\phi},w_{i-3,\phi}) = 0$).

Assuming that the evidence features are independent to each other given two vertical lines,

$$P(\mathbf{E}|w_{i,j},w_{i-1,k},l,r) = D(d_{j,k}|i,i-1)G(g_{j,k}|i,i-1)I(i_{j,k}|i,i-1)C(c_j|i),$$

where $D(d_{j,k}|i,i-1)$ is a PDF of distance between V_i and V_{i-1} being $d_{j,k}$ (the distance between v_j and v_k), $G(g_{j,k}|i,i-1)$ is a probability of the gradient difference between V_i and V_{i-1} being that of v_j and v_k ($g_{j,k}$), $I(i_{j,k}|i,i-1)$ is a probability of the intensity samples between V_i and V_{i-1} being similar to that of vehicle frame intensity, and $C(c_j|i)$ is a PDF of the length (coverage) of the extracted line feature. The vehicle frame intensity is sampled near the front line. We assume that $D(d_{j,k}|i,i-1)$ is Gaussian, and $G(g_{j,k}|i,i-1)$ is binary (whether the signs of the gradients are the same or the opposite). For example, the sign of the gradient of V_1 is opposite to that of the V_2 with the probability $G(\text{opposite}|2,1)$. The parameters of $D(d|i,i-1)$ and $G(g|i,i-1)$ can be obtained by observing learning examples.

Similarly, $P(\mathbf{E}|w_{i,j},w_{i-1,\phi},w_{i-2,k},l,r) = D(d_{j,k}|i,i-2)G(g_{j,k}|i,i-2)I(i_{j,k}|i,i-2)C(c_j|i)$. In fact, $D(d|i,i-2)$ can be obtained from $D(d|i,i-1)$ and $D(d|i-1,i-2)$ when we assume that they are all Gaussian:

$$E[D(d|i,i-2)] = E[D(d|i,i-1)] + E[D(d|i-1,i-2)]$$
$$V[D(d|i,i-2)] = \sqrt{V[D(d|i,i-1)]^2 + V[D(d|i-1,i-2)]^2}.$$

We assume that $P(w_{i,\phi}|w_{i-1,k},\Phi) = P(w_{i,\phi})$, where $P(w_{i,\phi})$ can also be obtained from learning examples.

In summary, our dynamic programming algorithm follows the steps below:

1. Given v_0, l, and r, gather n vertical line candidates (on the right side of v_0) for $V_1,\ldots,V_5$.
2. Make a $6 \times n$ table of $P(w_{i,j}|\Phi)$.
3. $P(w_{0,0}|\Phi) = 1$ and $P(w_{0,i}|\Phi) = 0$ for all $i \neq 0$.
4. Fill in the table using Eq. 2 and Eq. 3.
5. Sum up the last row of the table: $\sum P(w_{5,m_5}|\Phi) = \sum P(w_{5,m_5},\ldots,w_{1,m_1}|\Phi)$ (Eq. 1).

Our algorithm is different from Hidden Markov Model (HMM) because it allows to use *relative* features such as the distances, the gradient differences, and the intensity similarities. For example, we use the distances between the *candidates* of i and $i-1$, not the near-by features (i.e. v_j and v_{j-1}). It is hard to model such properties with HMM when random insertion exists.

3.3. Description Algorithm

Finding $P(w_{5,m_5}|\Phi)$ is sufficient for the detection purpose but the description can also be given. The description of the car can be obtained by applying our *backtracking algorithm*: given that $V_i = v_j$, find v_k which maximizes $P(w_{i-1,k}|w_{i,j},\Phi)$. We find

$$P(w_{i-1,k}|w_{i,j},\Phi) = \frac{P(w_{i,j}|w_{i-1,k},\Phi)P(w_{i-1,k}|\Phi)}{P(w_{i,j}|\Phi)}.$$

Note that $P(w_{i,j}|w_{i-1,k},\Phi)$ is calculated when we calculate $P(w_{i,j}|\Phi)$ (see the previous section). Therefore, when we fill in the table of $P(w_{i,j}|\Phi)$, we make another table, $\arg\max_k P(w_{i-1,k}|w_{i,j},\Phi)$, for the backtracking.

Where we regard the missing features, the algorithm is slightly modified: given that $V_i = l_j$, find l_k and r which maximize $P(w_{i-1,\phi},w_{i-2,\phi},\ldots,w_{i-r,k}|w_{i,j},\Phi)$. Therefore, we calculate $P(w_{i-1,\phi},w_{i-2,\phi},\ldots,w_{i-r,k}|w_{i,j},\Phi)$ for all the possible values of r ($r \in \{1,2,3\}$, in our implementation). For example, when $r = 2$,

$$P(w_{i-1,\phi},w_{i-2,k}|w_{i,j},\Phi) = \frac{P(w_{i,j}|w_{i-1,\phi},w_{i-2,k},\Phi)P(w_{i-1,\phi},w_{i-2,k}|\Phi)}{P(w_{i,j}|\Phi)} = \frac{P(w_{i,j}|w_{i-1,\phi},w_{i-2,k},\Phi)P(w_{i-1,\phi}|w_{i-2,k},\Phi)P(w_{i-2,k}|\Phi)}{\text{constant}}.$$

3.4. Learning Data Collection

Parameters, such as $D(d|i,i-1)$, $G(g|i,i-1)$, $I(i|i,i-1)$, and $P(w_{i,\phi})$, can be learned from examples. We implemented a user-interface to collect such learning examples. A learning example is made by clicking 8 points for $V_0,\ldots,V_5$, H_0, and H_1 (one point for each). We can infer a car structure from given points. Once we have a car model, we match models to the image line features, and estimate the parameters from those line features. Note that we only need a relatively small number of learning examples because all of the above parameters are independent to each other.

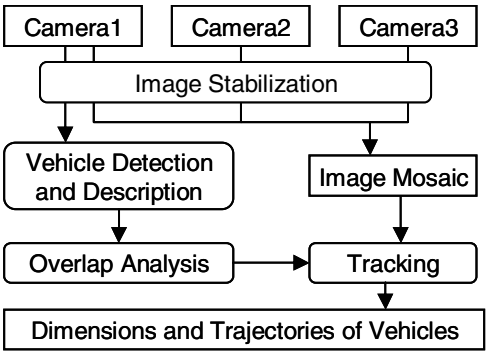

Figure 4. Flow diagram of the system

4. Tracking and System Implementation

The flow diagram of our system is shown in Figure 4. The cameras are installed on the roof of a tall building and, because of the wind, we sometimes get unstable video streams. Therefore, we first apply a stabilization algorithm to all images. For each image, we manually assign several "static" (background) areas for the stabilization. Then, for each frame, we find corners from these areas, find their matches in the previous frame, estimate camera transformation (affine), and generate new images using the transformation matrix.

Then, we apply the vehicle detection and description algorithm on a small entrance area of the first image. Since the algorithm is applied to each and every line feature, we may have many overlapping hypotheses for a single car. We choose the best hypothesis by comparing $P(w_{0,0}, \mathbf{E})$. The overlap analysis also includes the vehicle hypotheses detected in the past frames. It results redundant detection in all the frames where the vehicle is visible, which increases the detection rate.

The tracking is performed on the mosaic of all three images. We manually calibrated all three cameras. We rectify images using the calibration parameters and attach them to generate the mosaic image. Note that the brightness and contrast levels of all three cameras are different from each other (see Figure 1). Therefore, we adjust the brightness and contrast levels by examining those of overlapping areas: we estimate mean and standard deviation of the intensity pixels, and (linear) transform all the intensity levels of the images so that the brightness and contrast levels of all three images be the same. We only allow small changes (w.r.t. frames) on the parameters for modifying the brightness and contrast levels because a radical change on such parameters degrades the tracking performance. Figure 5 shows the resulting mosaic image.

The tracking is performed based on the *zero-mean cross-correlation matching* [15]. To reduce computation, we perform the search on a two-level image pyramid. For this, we make two mosaic images of different resolution, where the resolution is automatically determined with respect to the resolution of the original images. The search is performed with 9×9 RGB image patches (15×15 for the fine-level image) on 11×11 search windows (5×5 for the fine-level image).

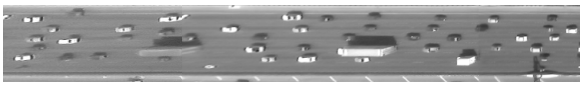

Figure 5. The mosaic image of Figure 1.

Figure 6. An example detection result. Vehicles detected with higher confidence are shown in red.

The track may be lost for several consecutive frames due to occlusions or other accidental alignments. For example, in Figure 1c, a part of the first (lower, in the picture) two lanes are occluded by the shadow of a traffic sign structure. To deal with such a case, when a search is failed in one frame, the system continues to search in the following frames based on the previously estimated vehicle speed. A trajectory is discarded when it loses the track for more than a certain number of frames (3 in our implementation). We do not try to refine the tracking result, such as by applying the Kalman filter, because it is better not introduce any bias on the resulting trajectories than produce smooth ones.

5. Experimental Results

The detection algorithm works in realtime (faster than 10 frame/sec for the 200×200 entrance area). However, the whole system does not because of the time required for the image retrieval (from hard disk, 3 MB of uncompressed images per frame) and tracking.

Figure 6 shows an example detection result in a single frame. We see that the detection and description quality is very good. The quality of the image is poor (smoothed interlaced video) and one car was not detected because important lines were not detected including the front line. Currently, we are in a process of converting the data collection procedure into digital, and we expect the detection rate be significantly increased.

To evaluate our grouping performance, we applied Another detection result is shown in Figure 7. It is much more difficult than the highway images. Shadows of the trees generates distracting lines and vehicles are parked very close from each other which makes the segmentation among

Figure 7. Another detection result with a much more complicated scene

.

vehicles very difficult (see line segments). Nevertheless, our algorithm shows reasonable grouping performance. Out of 21 cars (we do not count the van on the lower-left corner), 14 were correctly detected with 1 false alarm (or significant location errors). Four of the detected vehicles had small location errors which caused misdetection of several vehicles (which were detected but removed in the overlap analysis).

An example tracking result is shown in Figure 8. We find that most of the vehicles are correctly localized and tracked. We observe slight location errors for the vehicles on the left side. This is not the detection error but due to the image-based tracking algorithm which does not handle perspective changes. Our future work includes model-based tracking (Section 6). Our detection algorithm does not handle large trucks but a different algorithm will be applied for the truck detection. In fact, it is an easier problem because it is just to find a long rectangular structure. For example, techniques from building detection and description ([10]) can easily be applied.

A comparison with a manual count is shown in Table 1. The detection rate was 85% (116 out of 137) and the false alarm rate was less than 1% (only one). The localization performance was very satisfactory, and only two vehicles were detected with a significant position error (an error bigger than 1/3 of the size of the vehicle). The false alarm was generated from a carpool lane mark combined by the shadow of a dark vehicle (which was counted as misdetection). However, no valid trajectories were generated from it.

Tracking failures occurred on several vehicles because of 1) occlusion by a large truck (or their shadow casts) for more than 10 frames, and 2) specular highlights. For example, the specular highlight of vehicle on the top of Figure 6 (in the lower right corner) gradually disappeared after several frames.

Table 1. A detection result on 137 vehicles. Large trucks were not included.

total # of vehicles	137
# of correctly detected (passenger cars)	97
# of correctly detected (other vehicles)	19
detected with wrong position	2
# of missed detection	19
# of false alarms	1

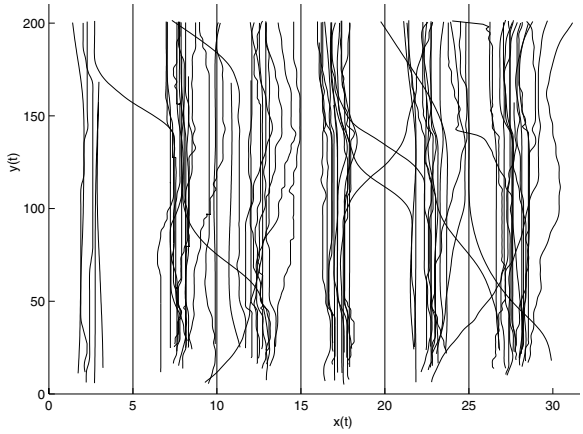

Figure 9. Resulting trajectories.

The resulting trajectories are shown in Figure 9 and Figure 10. We find that the quality of trajectories is superior (less noisy) than that of other vision-based data. We observe a good quality of shockwaves (delayed deceleration) on the second and the third lane, which will serve as useful data to generate the parameters of car-following models.

6 Conclusion and Future Work

We presented a vehicle detection and tracking system based on model-based vehicle detection. It provides high quality vehicle trajectories with accurate localization, which will bring a significant improvement in traffic flow analysis. We also introduced a new 3-D vehicle detection and description algorithm based on line features. Our feature-based algorithm has significant advantages over image-based algorithms (such as [17]). It is fast enough that it can be applied to many other applications which requires fast (or even realtime) processing. It is also flexible. It is much more free from the scale problem, and allows detection from more oblique views. The performance does not depend on the small change of the view points.

In addition, improving the description performance can enable the vehicle classification. For example, we can discriminate passenger cars from SUV's and pick-up trucks with the description results (line configuration). Thorough

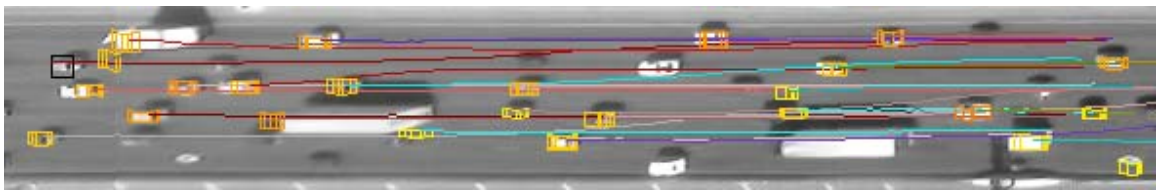

Figure 8. An example tracking result.

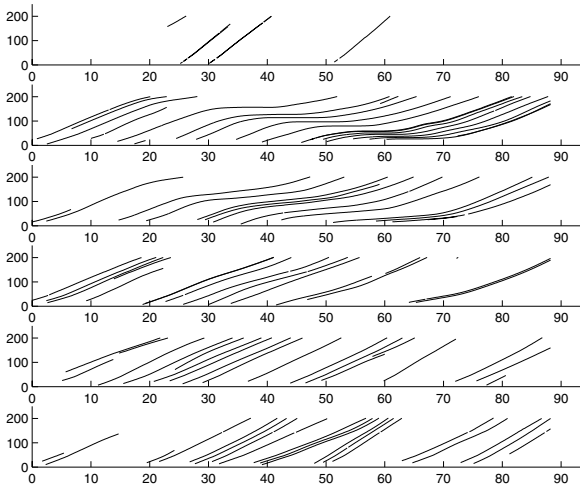

Figure 10. Resulting trajectories plotted lane by lane.

analysis with such level of classification is unprecedented in transportation studies. Another possibility is to apply a similar technique for tracking. The current tracking performance is not satisfactory due to occlusions, specular highlights and the change of perspectives. We believe that introducing invariant features such as lines and 3-D structure of the vehicle will improve the tracking performance significantly.

Acknowledgment

The aerial photo used in this paper is provided by University of Southern California by the courtesy of Tao Zhao and Ram Nevatia. We also thank Erik Miller for helping us with writing. This work was supported by CalTrans/PATH (TO4154).

References

[1] Traffic flow theory: A state-of-the-art report. http://www.tfhrc.gov/its/tft/tft.htm.

[2] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miché, S. Mousset, and G. Toulminet. A cooperative approach to vision-based vehicle detection. In *Proc. IEEE Intl. Conf. on Intelligent Transportation Systems*, pages 209–214, 2001.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real time computer vision system for measuring traffic parameters. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[4] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[5] R. Ervin, C. MacAdam, J. Walker, S. Bogard, M. Hagan, A. Vayda, and E. Anderson. System for assessment of the vehicle motion environment (savme): volume i, ii, 2000.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.

[7] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos. Detection and classification of vehicles. *IEEE Trans. Intelligent Transportation Systems*, 3(1):37–47, 2002.

[8] D. Koller, K. Daniilidis, and H.-H. Nagel. Model–based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[9] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. European Conf. on Computer Vision*, pages A:189–196, 1994.

[10] C. Lin and R. Nevatia. Building detection and description from a single intensity image. *Computer Vision and Image Understanding*, 72(2):101–121, 1998.

[11] L. Pipes. An operational analysis of traffic dynamics. *Journal of Applied Physics*, 24:274–281, 1953.

[12] A. Rajagopalan, P. Burlina, and R. Chellappa. Higher order statistical learning for vehicle detection in images. In *Proc. IEEE Intl. Conf. Computer Vision*, volume 2, pages 1204–1209, 1999.

[13] A. Reuschel. Fahrzeugbewegungen in der kolonne bei gleichfoermich beschleunigtem verzoegertem leitfahrzeug. *Oesterr. Ing.-Archiv*, 4, 1950.

[14] H. Schneiderman and T. Kanade. A statistical model for 3d object detection applied to faces and cars. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2000.

[15] P. Smith, D. Sinclair, R. Cipolla, and K. Wood. Effective corner matching. In *Proc. 9th British Machine Vision Conf.*, 1998.

[16] A. Worrall, G. Sullivan, and K. Baker. Pose refinement of active models using forces in 3d. In *Proc. 3rd European Conf. Computer Vision*, pages 341–352, 1994.

[17] T. Zhao and R. Nevatia. Car detection in low resolution aerial image. In *Proc. IEEE Intl. Conf. Computer Vision*, 2001.