Robust car tracking using Kalman filtering and Bayesian templates

Frank Dellaert[a] and Chuck Thorpe[b]

[a]Department of Computer Science and [b]The Robotics Institute
Carnegie Mellon University, Pittsburgh, PA 15213 USA

ABSTRACT

We present a real-time model-based vision approach for detecting and tracking vehicles from a moving platform. It was developed in the context of the CMU Navlab project and is intended to provide the Navlabs with situational awareness in mixed traffic. Tracking is done by combining a simple image processing technique with a 3D extended Kalman filter and a measurement equation that projects from the 3D model to image space. No ground plane assumption is made. The resulting system runs at frame rate or higher, and produces excellent estimates of road curvature, distance to and relative speed of a tracked vehicle. We have complemented the tracker with a novel machine learning based algorithm for car detection, the CANSS algorithm, which serves to initialize tracking.

Keywords: Car Detection, Car Tracking, Deformable Templates, Kalman Filter, Model-Based Vision

1. INTRODUCTION

This paper presents an approach to detecting and tracking vehicles from a moving platform, with the goal being to provide situational awareness to autonomous vehicles. CMU has been developing autonomous driving since 1984, beginning with the Terregator, up to the current series of Navlabs (6-11). Recently, the concept of an automated highway system (AHS) was demonstrated to the general public at the NAHSC feasibility demo '97 in San Diego. In this project, CMU is one of the forerunners. An overview of recent AHS related research at CMU can be found in [1]. Situational awareness, or knowing where other vehicles are, is an important capability for autonomous vehicles. If the relative position and speed of other vehicles is known, a course of action can be planned that might involve accelerating or decelerating, executing lane changes, overtaking vehicles, and other tactical driving decisions. Even in semi-autonomous operation, e.g., smart cruise control, tracking vehicles would allow a computer to assist the driver with automatic speed adjustment, car following, or alerting the driver when dangerous situations are detected.

We are investigating *computer vision* approaches to providing situational awareness. In an AHS scenario where all vehicles are automated and equipped with GPS, vehicles can just tell each other where they are over a communications link. However, in a mixed traffic scenario, where autonomous vehicles coexist with other non-automated vehicles, that is not a reliable option. Currently, radar[2] and laser range finders[1] are among the sensors being considered at CMU and elsewhere. Perhaps the most exciting, but also the most challenging, technology under investigation is that of computer vision, where images from an on board camera are to be analyzed in real time to form a 3D representation of the world around the vehicle, including other vehicles in it. Dickmanns called this *Dynamic Vision*, and his group has done most outstanding work in this area.[3]

In this work, we present both a novel car detection algorithm and a robust 3D approach to real time vehicle tracking. Car detection has been approached in many different ways, including detection of symmetry,[4] color segmentation,[5] motion segmentation,[6,7] and various methods based on generalized Hough Transforms.[8,7,9] Our Candidate Selection and Search (CANSS) algorithm uses a Hough Transform on the image gradient to advance candidate image rows and columns that might contain edges of the car's bounding box, after which a search takes place for the bounding box most probably generated by a car. In addition to this novel two-step approach, we apply machine learning to use the information extracted from the image optimally (in the Bayesian sense). This is in contrast to most if not all of the work cited above, where more ad hoc techniques are used.

Our approach to tracking uses model based vision techniques to track a bounding box in the image. Tracking has been approached from a pure image point of view,[10–12,7] some going as far as to extract some 3D information

Other author information: (Address correspondence to F.D.) e-mail: {dellaert,cet}@cs.cmu.edu, http://www.cs.cmu.edu/~dellaert

from the tracking process.[13,6] However, the most powerful approaches to tracking moving objects use a 3D model based approach.[14,15] The work most relevant to us combines the 3D model based approach with the ability to handle moving cameras.[16–19] In our approach, we track cars by predicting how an imaginary 3D bounding box around the car will give rise to a 2D bounding box in the image. This idea has been used before by Schmid,[16] although we were unaware of this fact at the time we developed our own version of it. By tracking the 2D bounding box in the video stream, we can update our estimate of the 3D position of the tracked vehicle. Since only line segments need to be tracked, the technique is fast and the image processing relatively simple. It is based on an energy minimization scheme that can be given a probabilistic interpretation. Although the underlying image processing is simple, it is also easily distracted by spurious features. Thus, we use a Kalman filter both to predict more accurately where the bounding box will appear, and to extract useful information from the tracking process. No ground plane assumption is made. We also use the Kalman filter to estimate the changing pitch of the Navlab with respect to the road, since changes in the pitch of the camera are amplified into large displacements of the tracked cars in the image. The resulting tracking system has excellent performance, is robust, and runs at frame rate.

In the remainder of the paper, the complete tracking system is explained in a bottom-up fashion. We start out in Section 2 by motivating the underlying image processing technique by which we track segments of a 2D bounding box. This technique works only if a good initial guess for the position of the bounding box is available, so we spend some time in Section 3 explaining the CANSS car detection algorithm, which serves to initialize tracking. In Section 4 we discuss how we use the Kalman filter, describing the image measurement equation and deriving 2 formulations for the system dynamics. On line pitch estimation is also explained. Tracking performance is illustrated at length in Section 5, followed by a concluding section (Section 6) that discusses opportunities for future work.

2. IMAGE PROCESSING FOR TRACKING

Here we explain the basic idea behind the image processing technique we use to track cars in video sequences. The problem can be stated as follows: given a frame of video and the location of the car in the previous frame, find its location in the current frame. Although many features on a car could be used to track it, we track only the 2D bounding box around the car. If you look at the image of a car, you can see that it has strong edges on all sides where the image transitions from the car to the background. This contour of strong edges is not always regularly shaped, but it can be approximated by a rectangle. This approximation holds rather well for a large range of aspects. Also, because frames are taken in rapid succession, we can assume that this bounding box will not have moved much in the image since the previous frame. Thus, at each time step, we can take the previous position of the bounding box as a starting guess to search for a new bounding box in the current frame.

To search for a car bounding box in each frame, we rely on an energy minimization technique. We are looking for a rectangular contour with strong edges on all sides. An obvious way to proceed is to define an objective function F that measures the strength of the edges around a particular contour. Thus, F encodes our domain knowledge about what constitute likely bounding boxes. If (t, l) and (d, r) are the top-left and bottom-right coordinates of the bounding box, respectively, and $I_v(I_u)$ is the horizontal (vertical) gradient image, a possible choice for F is the integral around the bounding box of the average gradient perpendicular to the contour on all 4 sides:

$$F = \frac{1}{r-l}\int_l^r I_u(t,v)\,dv + \frac{1}{r-l}\int_l^r I_u(d,v)\,dv + \frac{1}{d-t}\int_d^t I_v(u,l)\,du + \frac{1}{d-t}\int_d^t I_v(u,r)\,du \tag{1}$$

We can relate this objective function to a Bayesian likelihood function by using the Gibbs/Boltzmann distribution.[10,11] That formula has its origin in statistical mechanics and reflects the intuition that lower energy states are more probable than higher energy states. Thus, if we define $\alpha E_d = -F$ as the energy term we are trying to minimize, and $\mathbf{x} = [t\,d\,l\,r]^T$ as the 4-dimensional vector encoding the position of the bounding box, then the likelihood that the bounding box is at position $\mathbf{x}$ given the information available in the image $\mathbf{z}$ can be expressed as (where Z_d is a normalization factor):

$$P(\mathbf{z}|\mathbf{x}) = \frac{1}{Z_d}\exp[-\alpha E_d(\mathbf{x},\mathbf{z})] \tag{2}$$

A Bayesian prior can be introduced in like fashion. By the derivation above, we have conveniently expressed our intuitions about 'being a good bounding box' into a probabilistic framework. The advantage to that is that we

can now elegantly add in our *prior* beliefs of what a good bounding box should look like. In our case, this is the assumption that 'the bounding box will not move much with respect to its previous position'. A simple way to do add this prior is to use a Gaussian probability density centered on the previous position $\mathbf{x}_0$, with the covariance matrix Σ of the Gaussian encoding how uncertain we are about our assumption:

$$P(\mathbf{x}) = N(\mathbf{x}_0, \Sigma) \tag{3}$$

By combining the likelihood and the prior into a posterior probability using Bayes rule, we arrive at a maximum a posteriori (MAP) estimate for the bounding box position $\mathbf{x}$ (where we have used formulas (2) and (3), and in the last step take the negative logarithm to obtain a sum of energy terms):

$$\begin{aligned}
\mathbf{x} &= \arg\max_{\mathbf{x}} \; P(\mathbf{x}|\mathbf{z}) && (4)\\
&= \arg\max_{\mathbf{x}} \; k\, P(\mathbf{z}|\mathbf{x}) P(\mathbf{x}) && (5)\\
&= \arg\max_{\mathbf{x}} \; \frac{1}{Z_d} \exp[-\alpha E_d(\mathbf{x}, \mathbf{z})] * N(\mathbf{x}_0, \Sigma) && (6)\\
&= \arg\min_{\mathbf{x}} \; \alpha E_d(\mathbf{x}, \mathbf{z}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}_0) && (7)
\end{aligned}$$

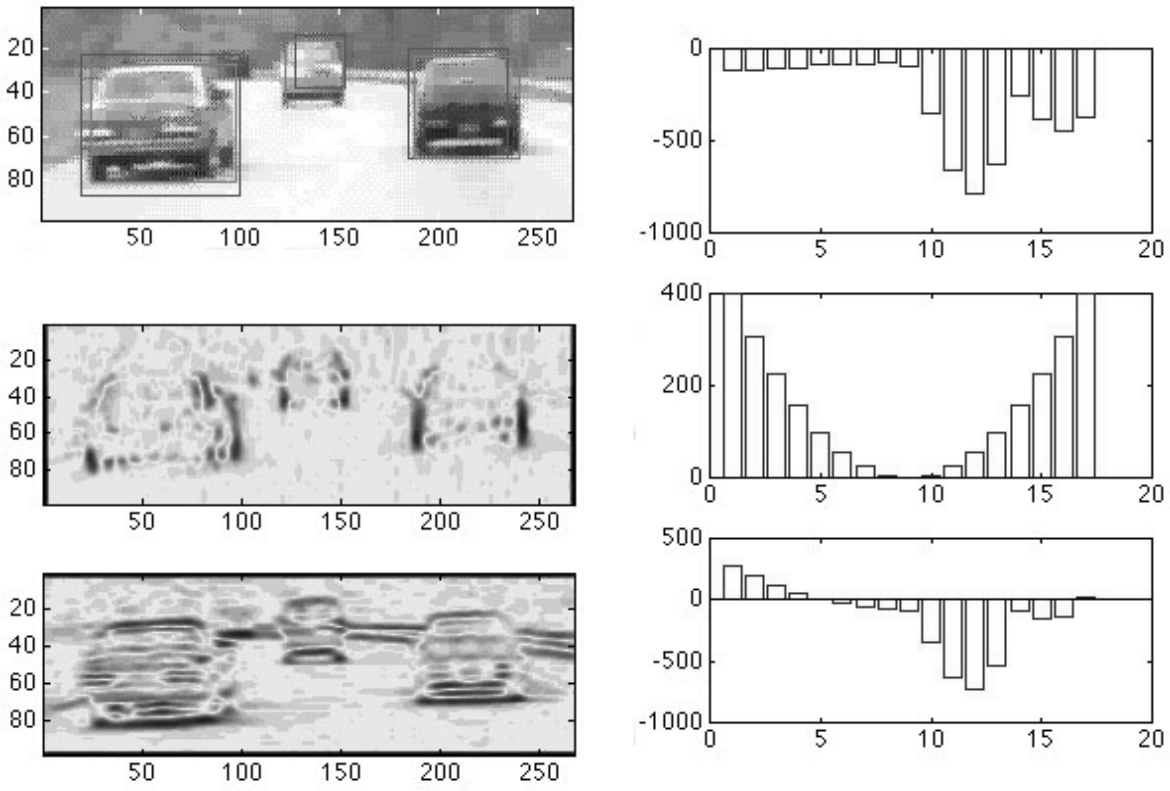

Figure 1. Left: the original image and the blurred horizontal and vertical gradient images. In the image, the dark bounding boxes are the initial estimates, and the lighter ones are the obtained MAP estimates. Right: Illustration of the energy minimization technique in one dimension. To position the left bounding box border of the rightmost car, we minimize the sum (bottom) of the likelihood energy (top) and the prior (middle). Since this is a vertical border, the likelihood energy corresponds to the average horizontal gradient along the border.

In summary, to find the MAP estimate for a given bounding box, we minimize the sum of two energy functions: E_d, which tells us how probable the image $\mathbf{z}$ is given $\mathbf{x}$, and E_p, which encodes our prior beliefs about $\mathbf{x}$. We

have illustrated this process for one dimension in Figure 1. In practice we have found that breaking up this 4D minimization into 4 one-dimensional minimization problems is efficient and does not lead to degraded performance. This because the effect of changing the integration limits is negligible compared to the effect of changing the position of the bounding box edges.

When a Kalman filter is used to integrate the measurement from the image with a prior belief state, we can use the same technique as described above, albeit slightly modified. In deriving (7), we have used Bayes law to combine the measurement with the prior. However, when using a Kalman filter the prior is really the current belief state of the filter, and Bayes law is applied in the measurement update equation of the filter (see below). Applying Bayes law twice would be an error. Thus, when measuring for the Kalman filter, we only minimize the likelihood energy term E_d, without adding in the prior energy term E_p.

3. INITIALIZATION

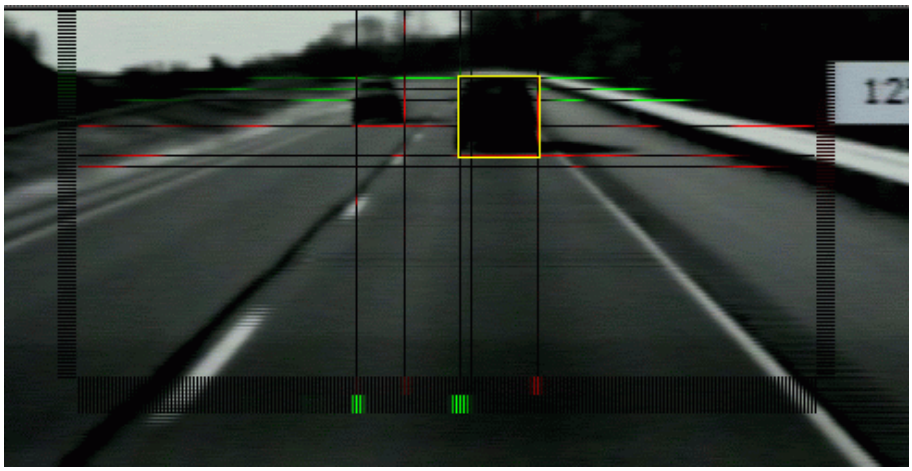

Figure 2. Successful car detection. See text for an explanation.

In the preceding section we have finessed an important point: how will tracking ever get started ? The technique described above will only be successful if we have an initial bounding box estimate that is very close to the true position. The image contains many distracting features that will confuse the tracking algorithm otherwise. Clearly a different technique is needed to initialize tracking.

We have come up with the CANSS (Candidate Selection and Search) algorithm to find probable cars in still images without needing a prior estimate. CANSS' output, when combined with a confidence measure, can be used to initialize tracking. The algorithm looks for the same 2D bounding box used in tracking, and tries to minimize the same likelihood energy E_d that we discussed above, subject to a prior probability distribution over different bounding boxes. Although the technique is explained in more detail elsewhere,[20] we will explain it briefly below.

The CANSS algorithm searches for the bounding box that maximizes a posterior probability, but proceeds in a two step process to eliminate most of the possible bounding boxes from consideration. There are millions of configurations in the image that can be assumed by a bounding box. Clearly it is infeasible to perform an exhaustive search over all of them and meet the real time constraints. Thus, the algorithm works in two steps: (1) select candidate rows and columns that probably contain a top, bottom, left or right border of a car-enveloping bounding box, and (2) search over all possible bounding boxes that can be formed using these candidates for the best one.

The *candidate selection step* selects candidates by ranking rows (columns) according to the probability that they contain a given bounding box border, and then outputting the m most probable ones for each border. Thus, in total 4 lists are produced, each containing the m different row numbers (column numbers) of the most promising candidates. For ease of exposition we shall from now on talk only about the left border. Whether a specific column in the image contains a left border can be posed as a classification problem. Properties or features of the column are the input to the classification problem, and the output is a yes or no answer. We have investigated a number of

features of the image columns, and the two most salient were (not surprisingly) average horizontal gradient value and (obvious but often overlooked) the column number itself (i.e., its v coordinate in a (u, v) image coordinate system).

We use *kernel regression* to calculate for each column the posterior probability that it contains a left edge. By picking 100 images at random and labeling them with the correct position of the cars in them, we obtained 10,000 training samples for the rows and 20,000 training samples for the columns. These can be used to train a classifier. For each of the images we have 100 rows and 200 columns in the window of interest we defined. The window was chosen such that most labeled cars in the training images fall into it. Thus, for left borders, each training sample is a tuple (position, average gradient over the column, 0/1). Kernel regression[21] simply forms a weighted average over the training samples, where the weighting function is a Gaussian centered on the query point, and the quantity to be averaged is the 0/1 output of each tuple. It can be shown that this procedure is equivalent to performing Parzen window density estimation[22] followed by applying Bayes law to arrive at a posterior probability.

The *combination search step* then takes the candidate lists, and searches over all valid bounding boxes that can be formed by combining the m candidates for each border. In the worst case, 4^m bounding boxes formed need to be searched. Typically, m has a value of 3 or 5, so that number is quite acceptable. To arrive at the MAP bounding box, we essentially use the same technique as for tracking. We minimize the sum of the likelihood energy E_d, and a prior energy term E_p. However, in this case the prior is defined by a Gaussian probability density over all possible positions and sizes of bounding boxes likely to be produced by the image of a car. This density is obtained by maximum likelihood estimation from the 100 training examples of which we spoke earlier. Successful car detection is illustrated in Figure 2. The horizontal and vertical lines represent the rows and columns, respectively, retained by the candidate selection step. The light box surrounds the bounding box advanced by the algorithm as the one most probably generated by a car.

4. INTEGRATION: THE EXTENDED KALMAN FILTER

To extract useful information from the tracking sequence and at the same time improve tracking performance, we use a Kalman filter. The quantities we are really interested in have to do with the relative 3D position and velocity of the tracked vehicle with respect to the moving camera platform. Thus, we will formulate a 3D model of the dynamics of a tracked car, whose state variables we want to estimate over time. To update this model, we predict where a car will appear in the image given its hypothesized 3D position. We then use the difference between this prediction and what we actually measure to update the state variables of the model. All this is conveniently done by means of an extended Kalman(-Bucy) filter.[23] Having a more realistic model of how a tracked car behaves also allows us to better predict how a its appearance will change over time. In each frame we will then have a more accurate initial estimate to initialize the search for a bounding box, which in turn will lead to better tracking performance.

To make the Kalman filter work, we need to supply a description of the *system dynamics*, and a *measurement equation*. One iteration of the Kalman filter then proceeds as illustrated in Figure 3. (1) The system dynamics are used to predict the state over time. In our case, the new relative position of the tracked car should be calculated, taking into account its and our current velocity and yaw rate. (2) The measurement equation is used to predict what measurement should be seen, given that the state estimate is correct. As explained below, this will entail projecting a 3D bounding box around the car into the image. The predicted measurement is then the 2D bounding box in image space. (3) The difference between the predicted and actual measurement, the *innovation*, is then used to update the state estimate. This *measurement update* automatically takes into account how uncertain we are of the current state estimate –quantified in the covariance matrix– and how much we can trust the measurements, as described by measurement noise terms.

We will discuss two formulations for the system dynamics: (1) a 3D formulation with a ground plane assumption, and (2) a formulation in road-based coordinates that assumes we know the lane the tracked car is traveling in. We have experimented thoroughly with both formulations, although the results we report on used the road-based filter. Yet, it is useful to introduce the road-based filter in the context of the 3D formulation.

4.1. Image measurement equation

The image measurement equation simply consists of projecting the 3D shape of the tracked vehicle into the image, after which a 2D bounding box is determined. To make this computationally cheap, and also because we do not know the true shape of a vehicle a priori, we approximate the true, unknown shape of a car by a 3D bounding box,

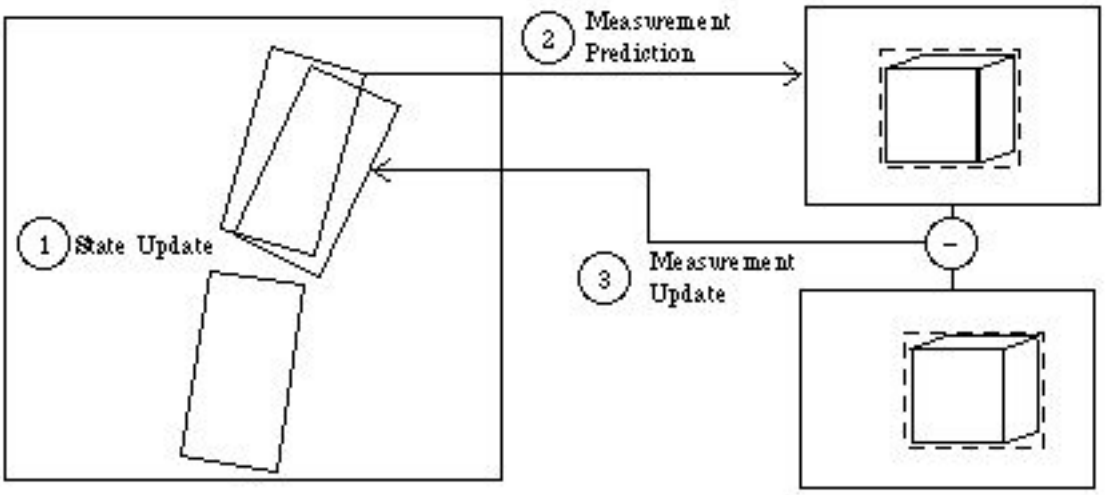

Figure 3. Schematic diagram of one iteration of the Kalman filter.

defined by the three parameters $[W\,L\,H]^T$, width, length and height. Given this simplifying assumption, it is then a simple matter to use the appropriate camera transformation to project the eight vertices of this 3D box into the image, and calculate the 2D bounding box. The 3D position and attitude of the car can be obtained either directly from the 3D filter, or indirectly from the road-based filter (see below). The output of the measurement equation is in the form $[t\,d\,l\,r]^T$, respectively the u coordinate of the top and bottom borders of the bounding box, and the v coordinate of the left and right borders.

The equation is not available in algebraic form, but is evaluated at each time step by pushing the eight vertices through a 3D graphics pipeline at runtime. The whole operation is efficiently implemented by making use of an OpenGL compatible 3D graphics card, which is becoming an affordable commodity at the time of writing. An additional benefit to using OpenGL is that we can easily visualize what is going on in the tracker by simply turning on the 3D rendering of the scene. The Jacobian of this process with respect to the state vector is evaluated numerically as well, using central differencing. Although we used to work with an analytic expression for this Jacobian, deriving it is not a pleasant experience, and inhibits experimentation with different models and filters.

4.2. The 3D filter

The 3D filter describes the dynamics of a tracked vehicle C in a coordinate frame attached to the tracking vehicle B. To make the filter tractable and robust, we will make certain simplifying assumptions. (1) The forward velocities of the cars will remain constant. This assumption will be violated when a vehicle accelerates, or decelerates, in response to changing traffic patterns. However, we can put bounds on the expected accelerations. (2) The angular velocities or yaw rate of each car will remain constant. This assumption will be violated when the cars enter a curve or when one of the cars executes a maneuver, say a lane change. (3) We make the ground plane assumption, i.e., the z coordinate of the tracked vehicle C is zero at all times. Using these assumptions, we can write down the equations of motion of the tracked vehicle C in the tracker coordinate frame B:

$$\dot{\mathbf{x}} = \begin{bmatrix} {}^b\dot{x}_c \\ {}^by_c \\ {}^b\psi_c \\ v_c \\ \omega_c \\ v_b \\ \omega_b \\ W \\ L \\ H \end{bmatrix} = \begin{bmatrix} -v_c \sin {}^b\psi_c + {}^by_c\omega_b \\ v_c \cos {}^b\psi_c - v_b - {}^bx_c\omega_b \\ \omega_c - \omega_b \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ w_{v_c} \\ w_{\omega_c} \\ w_{v_b} \\ w_{\omega_b} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where the components of the 10-dimensional state vector $\mathbf{x} = [{}^b x_c \; {}^b y_c \; {}^b \psi_c \; v_c \; \omega_c \; v_b \; \omega_b \; W \; L \; H]^T$ are defined as:

${}^b x_c, {}^b y_c$: the (x, y) position of the tracked vehicle C in the coordinate frame B attached to the ego-vehicle

${}^b \psi_c$: yaw angle, defined as the aspect angle of the tracked car with respect to frame B

v_c, v_b : the forward velocity, measured in the world coordinate frame, of the tracked vehicle and the ego-vehicle

ω_c, ω_b : the yaw rate, measured in the world coordinate frame, of the tracked vehicle and the ego-vehicle

W, L, H : the dimensions of the 3D bounding box around tracked vehicle

A few details about the filter are worth mentioning. The mixed global-local formulation, where we express positional variables in a relative coordinate frame but rate variables in the world coordinate frame, is interesting, since our constant velocity assumptions are only valid with respect to the world frame. However, expressing positional variables in the world frame is neither desirable nor needed, and we do not have to cope with global uncertainty in position. The additive Gaussian white noise terms $w_{v_c}, w_{\omega_c}, w_{v_b}$, and w_{ω_b} are needed to absorb the error we make by using the constant velocity assumptions. The strengths of these noise terms should reflect the knowledge we have about the cases in which the assumption is violated. Finally, the last three equations reflect the fact that the 3D dimensions of the car will not change (although we might not know them exactly).

There is a trade-off to using this rather general 3D formulation. On the positive side, it is flexible. For example, it can perfectly handle the behaviors of cars that drive on a road perpendicular to the road we are on, or vehicles traveling in the opposite direction. If we set the rate variables v_c and ω_c to zero, we can also model stationary objects. On the other hand, this very flexibility is also a disadvantage. Nowhere have we used the domain knowledge that we are in fact trailing cars that are on the same road. Intuitively, this means that we are not using the available information in an optimal way. In addition, given only image measurements, the filter can not adequately disambiguate between different traffic situations. As a simple example, if the car is slightly off to the left and does not seem to move in the image, that is compatible with two very different situations: either it concerns a vehicle in the same lane in a left curve, or a vehicle one lane over on a straight road. Thus, in order to disambiguate this filter we need to incorporate the information of other sensors, e.g., the road curvature estimate output by the RALPH[24] vision system.

4.3. The road-based filter

We formulated the road-based filter to get additional leverage from the domain knowledge at our disposal. Thus, we make the assumptions that (a) the tracked vehicle is on the same road, and (b) we know which lane it is in. The first assumption is valid in our application, where we are interested in tracking cars on the highway. The second point is more contentious, but makes sense after careful consideration. If we are rigorous about car behavior, we realize that lane changes are discrete events that should not be handled by the same filter used for smooth, in-lane tracking. We could inject more noise in the 3D filter to cope with these sudden lane changes, so called 'maneuvering noise', but that would also mean that we compromise the tracking performance in the most common case, i.e., when the tracked vehicle simply cruises along in its lane. Instead, we propose to recognize lane changes for the discrete events that they are, and handle them in a different way. We intend to address this problem more fully in future work, and concentrate here on in-lane tracking.

In the road-based filter, we define the position of the tracked vehicle in terms of the curvature of the road κ, the distance along the road to the vehicle, i.e., its longitude lng, and the lateral position of the vehicle in its lane, i.e., its latitude lat. In this filter, *we have dropped the ground plane assumption*, meaning that a car can have a non-zero z coordinate. The assumptions we will make here are that (a) the tracked vehicle is in the same lateral position within the lane as the ego-vehicle, i.e. lat is zero at all times, (b) the relative velocity of the tracked vehicle with respect to the ego-vehicle is constant, i.e. another constant velocity assumption, and (c) implicit in the road-based model is that both vehicles are along different points of a constant curvature road, and thus have well defined yaw rates as dictated by the curvature. This will only be approximate due to two effects: the swerving of the cars within their lane, and the fact that the curvature is not constant when entering or exiting a curve.

The road-based filter has a 8-dimensional state vector $\mathbf{x} = [lat\, lng\, z\, \kappa\, \Delta v\, W\, L\, H]^T$, where all components have been discussed before. The above assumptions lead to the following *linear* system equation:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{lat} \\ lng \\ z \\ \kappa \\ \Delta v \\ W \\ L \\ H \end{bmatrix} = \begin{bmatrix} 0 \\ \Delta v \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w_z \\ w_\kappa \\ w_{\Delta v} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

4.4. Modeling and estimating pitch bounce

We can augment either filter with additional state variables to simultaneously estimate the changing pitch of the ego-vehicle. As we have explained, the measurement equation is a projection from 3D to the 2D image, and is very sensitive to changes in the pitch of the camera. In addition, since the camera platform is a car moving at high speed, the pitch bounce introduced by irregularities on the road surface is quite significant. This bounce will cause large apparent motion of all objects imaged, including the tracked vehicle. One could deal with this by simply increasing the noise on the measurements, but this is clearly suboptimal. First of all, it will lead to degraded tracking performance, as we extract less information from each frame that way. And second, the effect in the image is very dependent on the relative position of the tracked car.

It is simple to augment the Kalman filter with an additional state variable modeling pitch. Intuitively, it is clear that one can disambiguate the pitch bounce from real motion because the top and the bottom of the car in the image will move simultaneously. It is also clear that the pitch bounce will not be a white noise, but rather time correlated noise, due to the relatively long time constants in the suspension. An additional issue comes into play when driving on long slopes. Here, the ground plane assumption can still be valid, but the suspension of the ego-vehicle makes that the pitch of the vehicle with respect to the road is now non-zero. Not accounting for this effect will introduce a bias in the measurement equations that the Kalman filter has no way of handling properly. Introducing a pitch baseline removes this difficulty. Thus, we have modeled the pitch baseline α_0 as a constant, and the pitch bounce α_n as a time-correlated, first-order Markov process, leading to the following state equations:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\alpha_0} \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 0 \\ -\alpha_n/T \end{bmatrix} + \begin{bmatrix} w_{\alpha_0} \\ w_{\alpha_n} \end{bmatrix}$$

5. RESULTS

Preliminary results suggest excellent performance for the resulting tracking system, as will be illustrated here in both a qualitative and quantitative manner. In Figure 4 we present the output of the tracker on a long sequence of video, recorded on the Navlab 6 while it was driving autonomously on I-79N near Pittsburgh, PA. The images were taken early in the morning with the sun still low on the horizon, throwing long shadows of trees across the road. These shadows created strong distracting edges on the road surface. In addition, the terrain at that location was quite hilly, often violating the ground plane assumption for extended periods of time, in particular on hill crests. Some typical images are shown in Figure 5 to 7. Despite the challenging nature of the sequence, the system kept track at all times for the duration of the sequence. Since the processing was done at frame rate, and the sequence lasted 120 seconds, this represents a continuous track over more than 3600 individual frames, while the distance covered was close to 4 km.

The state estimates obtained by the tracker correspond closely to ground truth, which was obtained by recording from the on-board sensors of the Navlab. A small Delco radar and a gyroscope provided measurements of distance to target and yaw rate, respectively. From the yaw rate and the measured velocity, we could calculate the instantaneous curve the Navlab was driving, which we smoothed out to get an estimate of road curvature. In addition, RALPH's estimate of road curvature was also recorded, giving us additional data for comparison. In Figure 4, the output of the tracker is compared with these measurements. Panels (c) to (e) show that both target distance and curvature

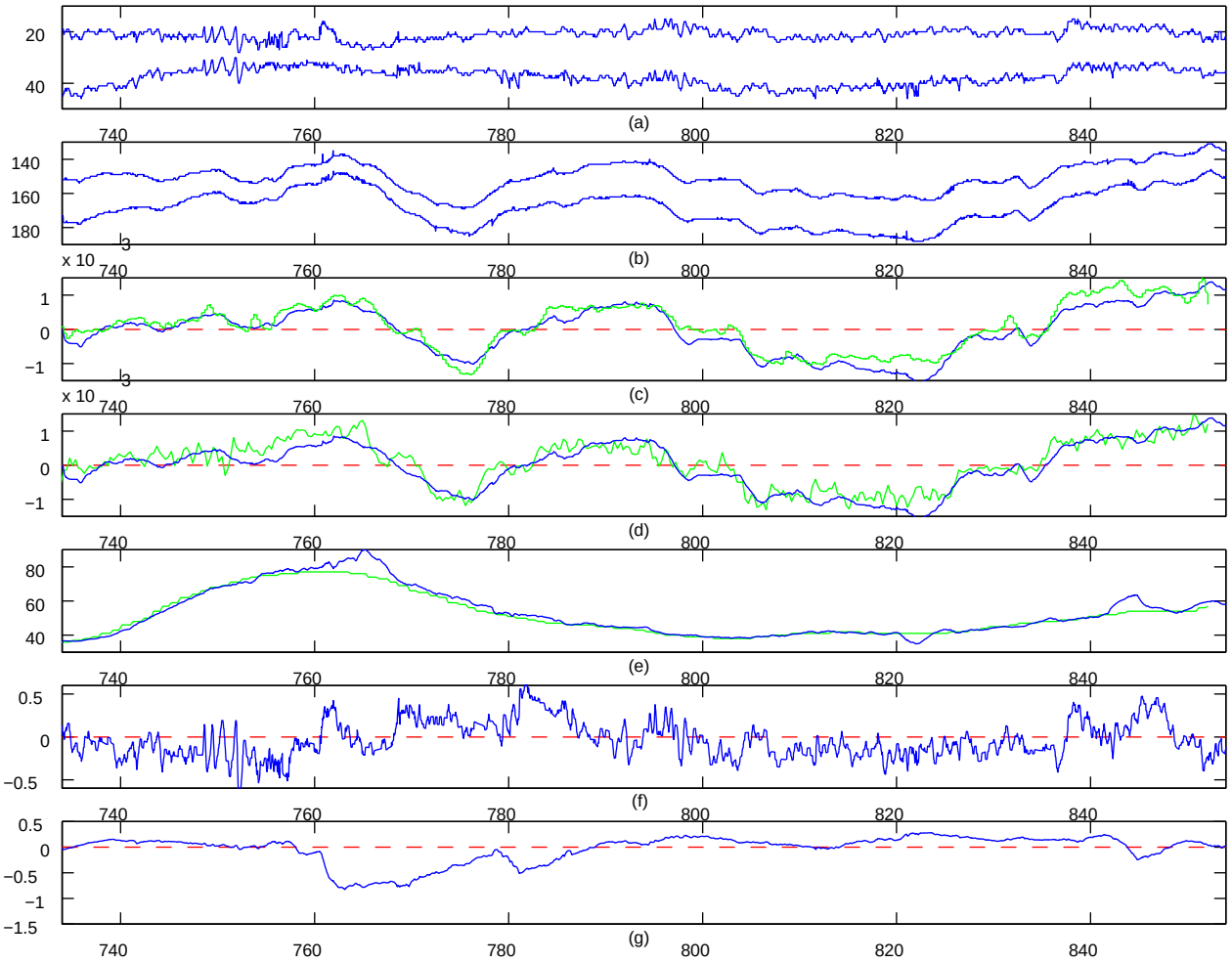

Figure 4. Tracker state estimate. From top to bottom: (a) measured image coordinates (in pixels) of top and bottom and (b) left and right edges of the tracked car; (c) curvature κ compared with the gyroscope's and (d) Ralph's estimate; (e) distance to target compared with radar measurement; (f) colored pitch bounce α_n; (g) z coordinate

estimates correspond very well with the recorded radar and curvature measurements. At about $t = 762$ the estimated distance diverges for a few seconds, where the long range (80m) and some challenging scenes (see Figure 6 at left) temporarily throw off the tracking of the top edge. Other difficult moments arise in shadow rich segments later in the sequence (at $t = 822$ and $t = 845$).

An interesting result is that the tracker provides a smooth and largely correct estimate of road curvature. This is not so surprising: even human drivers often use the location of cars up ahead as clues to where the road is, particularly at night. In our system, *nothing* is measured *except* for the bounding box around the car in the image. Under the assumption that the tracked vehicle is in the same lane, this provides enough information to obtain a clean curvature estimate. We have illustrated this by drawing the estimated lane boundaries on the images (Figure 6), where the lane width was obtained from RALPH, and it was assumed the Navlab drove in the middle of the lane. The filter's curvature estimate may diverge from the truth if the target vehicle has a non-zero lateral offset, as this violates the assumptions made above in the formulation of the filter. This failure mode shows up in the right panel of Figure 5. However, the error incurred will be small and will be automatically driven to zero if the Navlab actually steers this overestimated (or underestimated) curve. The problem can be solved by measuring the target's lateral offset in some other way, as we intend to do in future work. This difficulty aside, Figure 4 (d) shows clearly that the estimate of road curvature obtained in this way is much smoother than the curvature estimate output by RALPH.

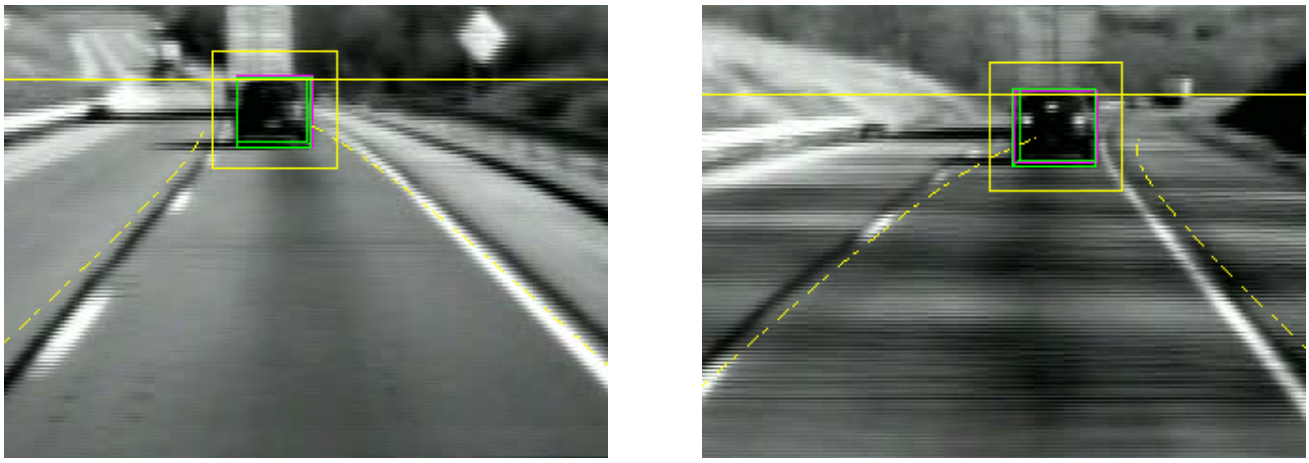

Figure 5. Challenges for the tracker. Left: frame at $t = 791$ showing typical terrain; Right: frame at $t = 819$ shows strong shadows and a curvature error (see text).

Figure 6. Two frames at $t = 762$ and $t = 775$, both with large curvature and with the tracked vehicle clearly below the ground plane.

The comparison is not entirely fair, for one is a filtered estimate while the other is a raw measurement of curvature. Thus, ideally RALPH's measurement should be integrated by our Kalman filter to obtain a superior estimate than both techniques can provide separately.

Figure 6 also shows how the filter adequately deals with violations of the ground plane assumption. In these two images one can clearly see that both tracker and tracked car are scaling the crest of a hill. Since both vehicles are driving on a vertical curve, they are no longer in the same plane, and the z coordinate of the target in the tracker's coordinate frame should be negative. From looking at the filter's z estimate in Figure 4 (g), it can be seen that this situation is correctly recognized and handled by the filter. An effect of the non-zero coordinate is that the horizon line on Figure 6 no longer coincides with the top of the tracked car, as would be the case with a ground plane assumption and our particular camera placement.

Another nice feature of our filter formulation, without which we have found tracking to be not nearly as good, is the image stabilization provided by the on-line estimation of the correlated pitch noise. Figure 7 shows two frames taken closely after one another, when the Navlab drove over a bump on the road at around $t = 750$. The filter output in Figure 4 shows a dramatic oscillation resulting in image displacements of more than 10 pixels, as can be seen from panel (a). The pitch bounce estimated by the filter is shown in panel (f), which clearly follows the disturbances that show up in the image. The quality of the resulting image stabilization can be appreciated by looking closely at

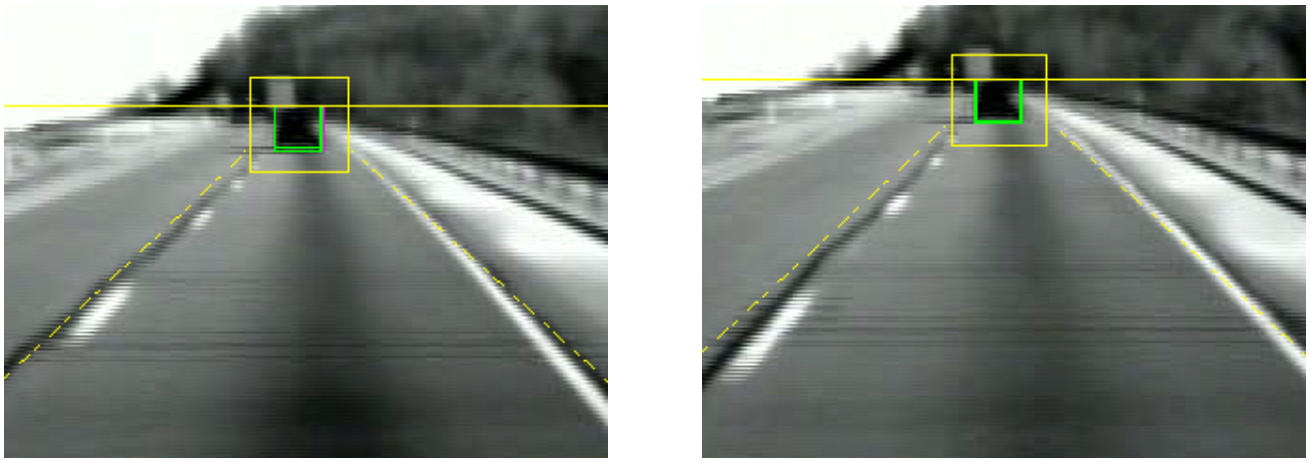

Figure 7. Image stabilization. Two frames (one close after the other around $t = 750$) illustrating image stabilization by augmenting the state vector with pitch, modeled as colored (time correlated) noise.

Figure 7, and noting that the horizon line superimposed on the image nicely tracks the image features at a distance.

6. CONCLUSIONS AND FUTURE WORK

We have presented a real-time model-based vision approach for detecting and tracking vehicles from a moving platform such as the Navlab. Tracking relies on a principled image measurement technique based on Bayesian templates, and the formulation of an extended Kalman filter in road coordinates, augmented with an image stabilization part. In contrast to other approaches, we make no ground plane assumption. The resulting system runs at frame rate or higher, and produces excellent estimates of road curvature, distance to and relative speed of the target vehicle. Finally, we have complemented the tracker with the novel machine learning based CANSS car detection algorithm, which serves to initialize tracking.

There are many potential improvements to the image processing that could be implemented. In particular, we would like to use machine learning to aid tracking in a principled way, rather than fiddle with the parameters or use ad-hoc techniques. This can be readily done by means of the likelihood energy term E_d, whose relationship to the image can be learned from training data. Other improvements that might involve learning include knowing about different types of cars, about how left and right edges differ, etc. The latter would be especially helpful, since sometimes the tracker is distracted by edges of other vehicles that appear from behind the target vehicle. Occlusion reasoning, automatic error recovery and robust initialization are other opportunities for future work.

An elegant property but also a potential weakness of the current system is that it relies solely on the tracking of one bounding box to derive everything else. Luckily, the Kalman filter easily allows for integration with other measurement devices such as RALPH, and other on board sensors. Nevertheless, we would like to investigate just how far one can get using only vision. Clearly, humans rely on many cues in the image to obtain a stable visual image. In future work, we plan to track other image features to obtain better ego-motion estimates that tracking one car alone can provide. A first step in that direction could simply be multiple vehicle tracking.

Once extra information is available, we can be more sophisticated about the filter formulation itself. For example, we could model all of lateral position, z coordinate and the pitch baseline as zero-mean time correlated noises with different time constants. As it is now, not enough information is available to distinguish between changes in baseline and z, for example. Finally, going up one level higher, we should try to recognize lane changes and other tactical maneuvers as discrete events, and model them as such. Only when we think of the system in mixed discrete-continuous terms will the world of highways start making sense.

ACKNOWLEDGEMENTS

This work was supported supported by USDOT under Cooperative Agreement Number DTFH61-94-X-00001 as part of the National Automated Highway System Consortium.

REFERENCES

1. C. Thorpe, "Mixed traffic and automated highways," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)*, vol. 2, (Grenoble, France), September 1997.
2. D. Langer, "An integrated mmw radar system for outdoor navigation," PhD Dissertation CMU-RI-TR-97-03, Carnegie Mellon University, January 1997.
3. E. Dickmanns, "Vehicles capable of dynamic vision," in *IJCAI-97, (Nagoya, Japan)*, 1997.
4. T. Zielke, M. Brauckmann, and W. von Seelen, "Intensity and edge–based symmetry detection applied to car–following," in *Proceedings of Computer Vision (ECCV '92)*, G. Sandini, ed., vol. 588 of *LNCS*, pp. 865–873, Springer, (Berlin, Germany), mai 1992.
5. S. Hirata, Y. Shirai, and M. Asada, "Scene interpretation using 3-d information extracted from monocular color images," in *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 1603, (Raleigh, NC), July 1992.
6. D. Koller, J. W. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in *European Conference on Computer Vision*, pp. 189–196, LNCS 800, pringer-Verlag, May 1994.
7. M. Betke, E. Haritaoglu, and L. S. Davis, "Multiple vehicle detection and tracking in hard real time," Technical Report CS-TR-3667, University of Maryland, College Park, July 1996.
8. N. Tan, G. Sullivan, and K. Baker, "Fast vehicle localisation and recognition without line extraction and matching," in *Proceedings of the British Machine Vision Conference 1994 (BMVC '94)*, vol. 1, pp. 85–94, 1994.
9. V. Graefe and W. Efenberger, "A novel approach for the detection of vehicles on freeways by real-time vision," in *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, (Tokyo, Japan), September 1996.
10. D. Terzopoulos and R. Szeliski, "Tracking with kalman snakes," in *Active Vision*, A.Blake and A.Yuille, eds., pp. 3–20, MIT Press, Cambridge, MA, 1992.
11. A. L. Yuille, P. W. Hallinan, and D. S. Cohen, "Feature extraction from faces using deformable templates," *International Journal of Computer Vision* **8**(2), pp. 99–111, 1992.
12. R. Curwen and A. Blake, "Dynamic contours: Real-time active splines," in *Active Vision*, A. Blake and A. Yuille, eds., MIT press, Cambridge, MA, 1992.
13. N. Ferrier, S. Rowe, and A. Blake, "Real-time traffic monitoring," in *2nd IEEE Workshop on Applications of Computer Vision, Sarasota, Florida*, 1994.
14. J. Wu, R. Rink, T. Caelli, and V. Gourishankar, "Recovery of the 3-d location and motion of a rigid object through camera image (an extended kalman filter approach)," *International Journal of Computer Vision* **2**, pp. 373–394, 1989.
15. C. Harris, "Tracking with rigid models," in *Active Vision*, A. Blake and A. Yuille, eds., MIT press, Cambridge, MA, 1992.
16. M. Schmid, "An approach to model-based 3-d recognition of vehicles in real time by machine vision," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '94)*, vol. 3, (Munich, Germany), September 1994.
17. F. Thomanek and D. Dickmanns, "Obstacle detection, tracking and state estimation for autonomous road vehicle guidance," in *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 1399, (Raleigh, NC), July 1992.
18. F. Thomanek, "Visuelle erkennung und zustandsschtzung von mehreren straenfahrzeugen zur autonomen fahrzeugfhrung," *Fortshritt-Berichte VDI* **12**(272), 1996.
19. J. Weber, D. Koller, Q.-T. Luong, and J. Malik, "An integrated stereo-based approach to automatic vehicle guidance," in *Proceedings, International Conference on Computer Vision*, (Boston), June 1995.
20. F. Dellaert, "CANSS: A candidate selection and search algorithm to initialize car tracking," Tech. Rep. CMU-RI-TR-97-34, Robotics Institute, Carnegie Mellon University, 1997.
21. C. G. Atkeson, S. A. Schaal, and A. W. Moore, "Locally weighted learning," *AI Review* **11**, pp. 11–73, 1997.
22. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.
23. P. Maybeck, *Stochastic Models, Estimation ond Control*, vol. 2, Academic Press, New York, 1982.
24. D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert* **11**, April 1996.