Robust Multiple Car Tracking with Occlusion Reasoning

Dieter Koller, Joseph Weber, and Jitendra Malik

University of California at Berkeley,
EECS Dept., Berkeley, CA., 94720

Abstract

In this work we address the problem of occlusion in tracking multiple 3D objects in a known environment and propose a new approach for tracking vehicles in road traffic scenes using an explicit occlusion reasoning step. We employ a contour tracker based on intensity and motion boundaries. The motion of the contour of the vehicles in the image is assumed to be well describable by an affine motion model with a translation and a change in scale. A vehicle contour is represented by closed cubic splines the position and motion of which is estimated along the image sequence. In order to employ linear Kalman Filters we decompose the estimation process in two filters: one for estimating the affine motion parameters and one for estimating the shape of the contours of the vehicles. Occlusion detection is performed by intersecting the depth ordered regions associated to the objects. The intersection part is then excluded in the motion and shape estimation. This procedure also improves the shape estimation in case of adjacent objects since occlusion detection is performed on slightly enlarged regions. In this way we obtain robust motion estimates and trajectories for vehicles even in the case of occlusions, as we show in some experiments with real world traffic scenes.

Keywords: Multitarget Tracking, Occlusion Reasoning, Active Contours.

1 Introduction and Related Work

Research in machine vision based traffic surveillance systems is of increasing interest for communities in areas with a high traffic density on highways. The vision component of a traffic surveillance system, like the one proposed in this article, should provide descriptions of vehicles which are rich enough to derive higher level descriptions necessary for a decision component. Such a decision component draws decisions like "stalled car in lane 2 detected", "high traffic with stop and go", "accident in lane 3", which can be transmitted to a central computer station which decides for an action such as assigning a detour (i.e. [Huang *et al.* 93]).

The issue in vision-based traffic surveillance systems is to extract descriptions of moving vehicles from video data which enables further symbolic reasoning about the vehicles. The video data are usually provided from stationary cameras mounted at a location high above the road, e.g. at an overpass or on poles beside the road (see e.g. Figure 4) in order to cover a large field of view and to reduce occlusions of vehicles.

Now there is a tradeoff in a vision system between the accuracy of the data and the performance of the system. The ideal case would be, of course, the evaluation of images at video rate. While real-time tracking of single objects has been achieved using a contour tracker on a transputer network (i.e. [Curwen & Blake 92]) multitarget tracking in realtime has only been achieved for single points ([Rao 92]).

Performance can be increased without loosing the richness of the description by using a-priori knowledge about the scene, the objects and their motions. This enables the use of dedicated models for shape and motion, but reduces also the applicability. A full 3D-model-based and accurate, but computational expensive tracker has been suggested by [Koller *et al.* 93]. A simpler and faster but less accurate model-based tracker for single objects has been implemented by [Worrall *et al.* 91] and [Marslin *et al.* 91].

There is a need for a simple and fast but *robust* tracker. We decided to perform the tracking in the 2D image without any specific a-priori knowledge about the shape of the moving objects, except the assumption that the image projection of the objects can reasonably be described by a closed contour. One advantage of this approach over one based on complete models is that we gain robustness. A model-based approach requires models for all kinds of vehicles that might appear in the traffic domain. These models have to be parameterized since it is very unlikely that exact models of all vehicles are available (How does one expect to have models of every kind of car, truck and motorcycle?). We further heavily use a constraint on motion of the objects, which enables the use of simple and fast (linear) Kalman Filter techniques for the motion estimation.

Multitarget tracking requires not only an estimation algorithm that generates an estimate of the state of the object (target) to be tracked but also a data association component that decides which measurement to use for updating the state of which object. According to [Bar-Shalom 90] there are three approaches for data association:

1. heuristic — based on some simple rule or a-priori knowledge
2. probabilistic non-Bayesian — hypothesis testing or likelihood function
3. probabilistic Bayesian — conditional probabilities are computed for the various possible associations

Various work has been devoted to provide solutions to the problem of data association in multitraget tracking applications (i.e. [Chong *et al.* 90; Kurien 90; Rao *et al.* 93]). They all provide general probabilistic solutions. In our case we can exploit some a-priori knowledge given by the scene geometry in order to solve the data association problem. Our measurements can be corrupted by white Gaussian measurement noise (as in a single object tracking case) and by occlusions (measurements are missing or wrong due to overlayed data from another object). Our data association approach falls therefore into the heuristic category. Using this heuristic has the advantage that we can avoid the computational load, which emerges in using a probabilistic approach, where either several hypotheses for each object are traced simultaneously or complex posteriori probabilities have to be computed. But nevertheless, multitarget tracking of n objects requires not only n times the computation load compared to the single object case, but requires also a $n \log n$ or n^2 component for data association (i.e. checking for occlusions).

There is little work on multi-object tracking application using vision sensor data. In [Irani *et al.* 92] affine motion parameters are estimated and a temporal integration used to track a single object. They do not address the problem of data association in the case of potential occlusions. [Meyer & Bouthemy 93] explicitly handle the occlusion problem. In their work they address the problem of linking partial spatiotemporal trajectories in the case of total occlusion using motion coherence, but they do not address the problem of data association and shifts in the estimated positions due to partial occlusions. Partial occlusions usually occur prior to a total occlusion and a missing occlusion reasoning step may already affect the shape and motion estimation which causes the motion coherence to be ineffective. [Létang *et al.* 93] realized that in the case of a partial occlusion the center of gravity of a blob mask is not a reliable feature to track. They handle the (partial) occlusion problem by tuning the measurement noise according to the change in blob surface. In this way their tracker relies more on the motion coherence than on the measurement in a case of an occlusion. This method is rather heuristic and not applicable in the case of an affine motion with a change in blob surface.

Our work has been influenced by [Meyer & Bouthemy 92] and [Blake *et al.* 93]. [Meyer & Bouthemy 92] use convex polygons as descriptions of moving regions. [Blake *et al.* 93] successfully extended their real-time contour tracking system ([Curwen & Blake 92]) by exploiting affine motion models. They use so-called *Snakes* (active contour models) as descriptions of projected objects. Snakes have been introduced as deformable contours by [Kass *et al.* 88] and since then used in different variants in various areas of computer vision with emphasize to tracking rigid and non-rigid objects. In [Cipolla & Blake 90], real-time snakes are used to track the occluding contours of 3D objects as seen from a camera attached to a moving robot arm. The Kalman-Filter theory ([Gelb 74]) is incorporated in the snake technique to form so-called *Kalman-Snakes* ([Terzopoulos & Szeliski 92]).

The common approach for updating a snake contour associated to a rigid or non-rigid object along the image sequence is to formulate an elastic-model approach. In such an elastic-model approach forces are introduced based on image gradients normal to the contour in order to let the contour fall into local greyvalue minima found in the vicinity of the current contour. These approaches do not exploit any motion information contained in the images. Instead a motion model is usually used to formulate system dynamic equations and to predict the contour in a subsequent frame. Our approach for tracking contours deviates from this common snake technique in the sense that we do not use a force model, but we explicitly exploit motion information in the image if the object is moving.

Another important component in tracking systems is the *track* formation or *initialization*. Our tracker is initialized by a motion segmentation step described in Section 2. From our experience with different motion segmentation approaches using optical flow or displacement vector segmentation, the described approach based on background image differencing yields the best results and is also fast. In Section 3 we describe our affine motion model used in the tracking stage. We continue with the contour extraction based on a convex polygon enclosing some sample points from the motion segmentation. We use these polygons for a a subsequent spline approximation for the final shape representation, as described in Section 4. Section 5 introduces the time recursive motion and shape estimation. The occlusion reasoning step is developed in Section 6. In Section 7 we show results of our approach with real-world traffic scenes and finally close with a conclusion in Section 8.

A block scheme of the complete tracker including the occlusion reasoning step is shown in Figure 1.

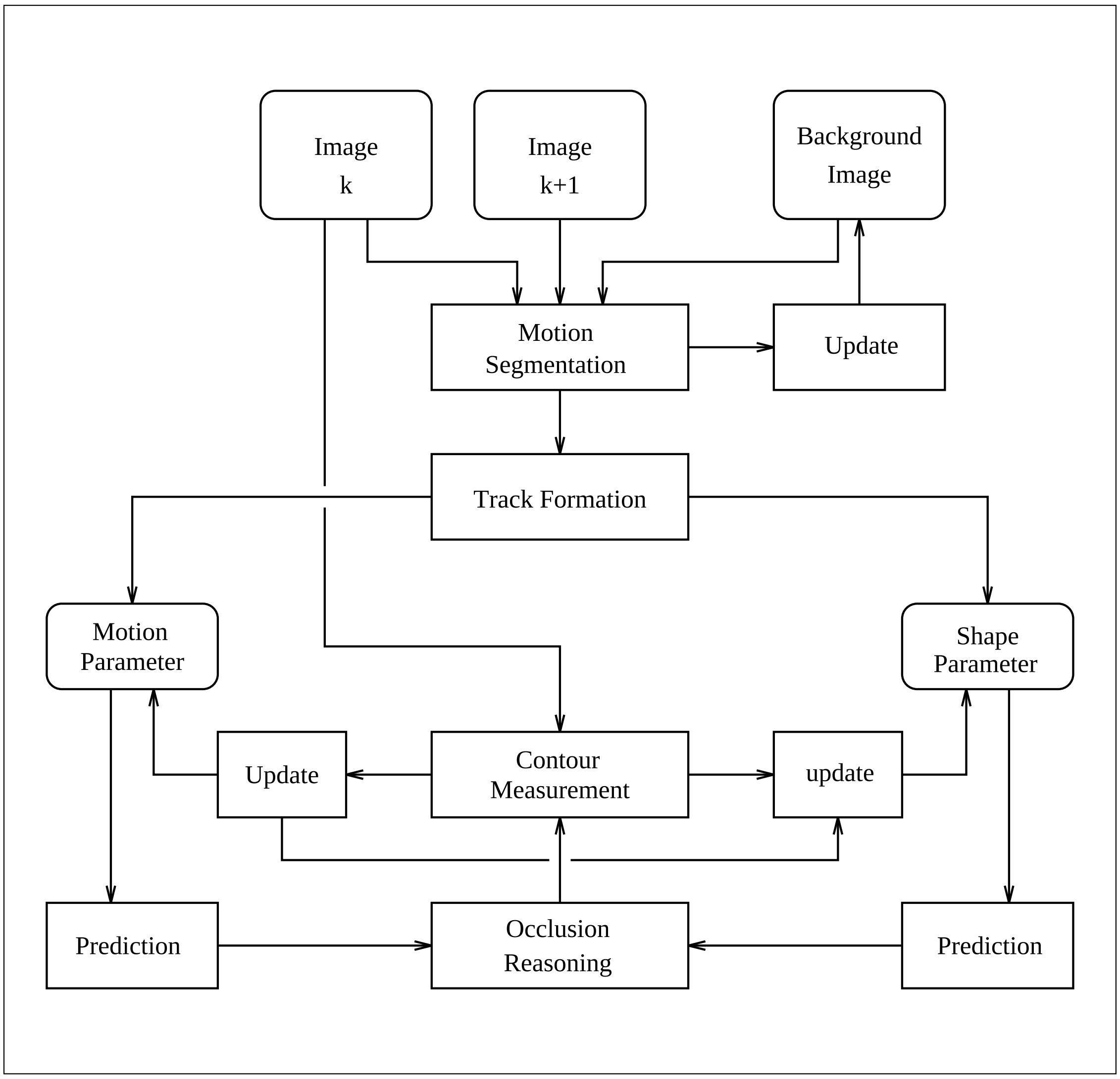


Figure 1: A block scheme of the complete tracker. Processes are shown in rectangular boxes while data are shown in rectangular boxes with rounded corners.

2 Motion Segmentation

The simplest technique for separating moving objects from a stationary background is through examining the difference between each new frame and an estimate of the stationary background. Segmentation of moving objects in an outdoor environment also requires that the background estimate evolve over time as lighting conditions change. Thus changes in stationary parts of the image must be differentiated from changes due to moving objects.

We use a modified version of the moving object segmentation method suggested by [Karmann & von Brandt 90] and implemented by [Kilger 92]. This method uses an adaptive background model. The background model is updated in a Kalman filter formalism, thus allowing for dynamics in the model as lighting conditions change. The background is updated each frame via the update equation

$$B_{t+1} = B_t + (\alpha_1(1 - M_t) + \alpha_2 M_t)D_t \tag{1}$$

Where B_t represents the background model at time t, D_t is the difference between the present frame and the background model, and M_t is the binary moving objects hypothesis mask. The gains α_1 and α_2 are based on an estimate of the rate of change of the background. In a complete Kalman filter implementation, these values would be estimated along with the background since they correspond to elements of the error covariance. For our simulations, we found that small constant values of $\alpha_1 = 0.1$ and $\alpha_2 = 0.01$ produced good results. A block diagram of the procedure is shown in Figure 2.

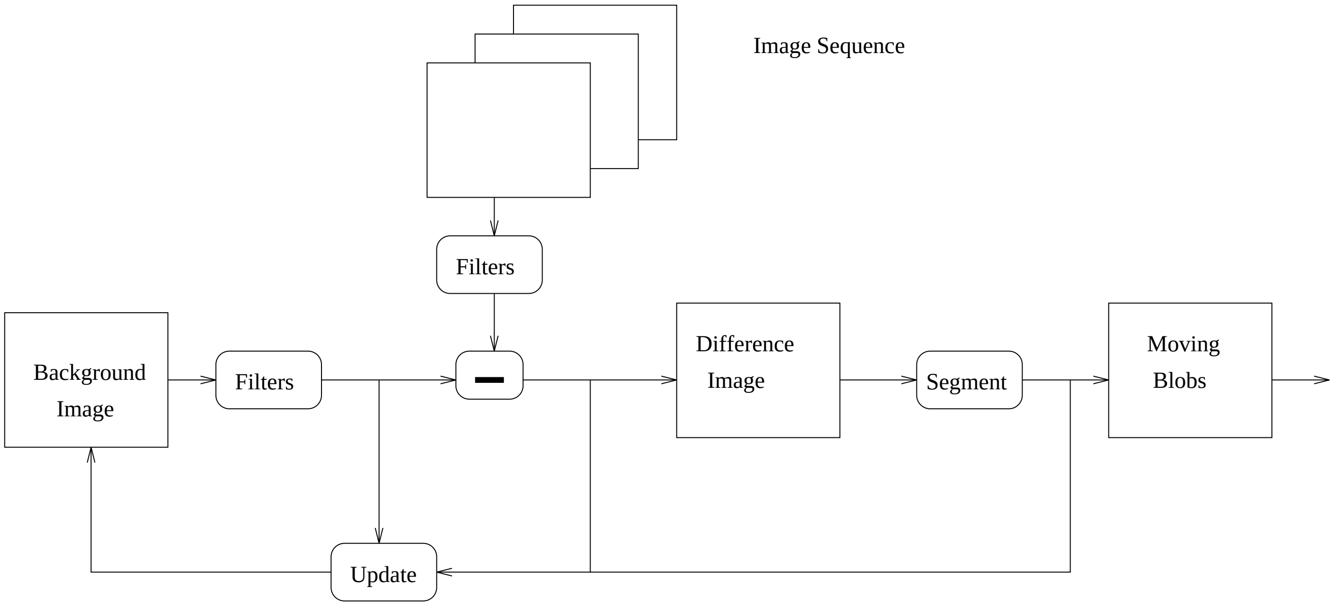


Figure 2: Block diagram of segmentation and background update procedure.

The hypothesis mask, M_t, attempts to identify moving objects in the current frame. In the original implementation this mask was obtained by thresholding the difference image. For the image position $\mathbf{x}$,

$$M_t(\mathbf{x}) = \begin{cases} 1 & |D_t(\mathbf{x})| > T_t \\ 0 & \text{else} \end{cases} \tag{2}$$

This equation assumes that the power in the difference image is greater than T_t at locations of moving objects. If we model the difference image as the summation of the image of moving objects plus noise, then we know that given the spectral characteristics of the signal and noise, a filter can be implemented to increase the power of signal over noise. Our implementation differs from the one used in [Karmann & von Brandt 90; Kilger 92] in that we employ linear filters to increase the accuracy of the decision process. Thus in the notation above, B_t and D_t represent a vector of filtered responses instead of single images.

For the known camera geometry, we can assume that images of moving vehicles will consist of regions of uniform color and edges almost entirely in horizontal or vertical orientations. (This assumption may become invalid if future vehicles are radically different in appearance then contemporary ones). Thus a matched linear filter will have power at DC and at spatial frequencies consistent with horizontal and vertical edges. We choose as filter kernels the Gaussian and the Gaussian derivative along the horizontal and along the vertical directions. The Gaussian components of all three filters reduce high spatial frequencies where we assume the power of the noise is significant compared to the signal. The derivatives respond to edges in the two orientations we expect in the signal. This was pointed out in [Canny 86]. In addition, the Gaussian functions are separable and allow for an efficient computational implementation. We then threshold the magnitudes of the filtered responses in order to make the moving object hypothesis for each channel. The logical OR of the separate thresholded channels forms the complete object hypothesis mask. Since each channel is tuned to frequencies where we expect the signal to be strong, we can use a higher threshold than if we used a single channel which was less selective.

To demonstrate the improved prediction via filtering, we added noise to a sequence where the locations of the vehicles had been decided by hand. The difference between the number of pixels labeled correctly and those labeled incorrectly is plotted as a function of added Gaussian noise in Figure 3. The three plots shown were obtained from (1) using the three Gaussian filters outlined above, (2) A single Gaussian function for noise reduction and (3) no filtering. The thresholds for the plots were set to 200 times the power of the filter for the three Gaussian filters, 100 times the power for the single Gaussian and a value of 50 for the case without filters. The first two thresholds were chosen to give the same number of correct responses on the vehicles for the two cases when no extra noise is present. The 200 times the power threshold was the same as used in the algorithm.

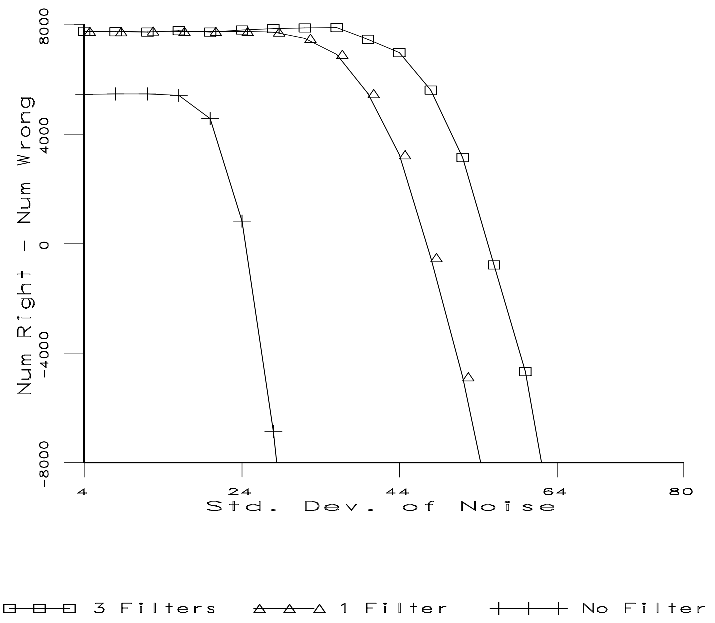


Figure 3: Plot of the number of pixels correctly labeled minus the number incorrectly labeled as a function of added noise for the case of (1) Three selective filters, (2) Single Gaussian filter and (3) No filtering.

An example of the difference mask obtained from our modified method compared to an intensity alone based method is shown in Figure 4. The binary difference mask is then translated into distinct

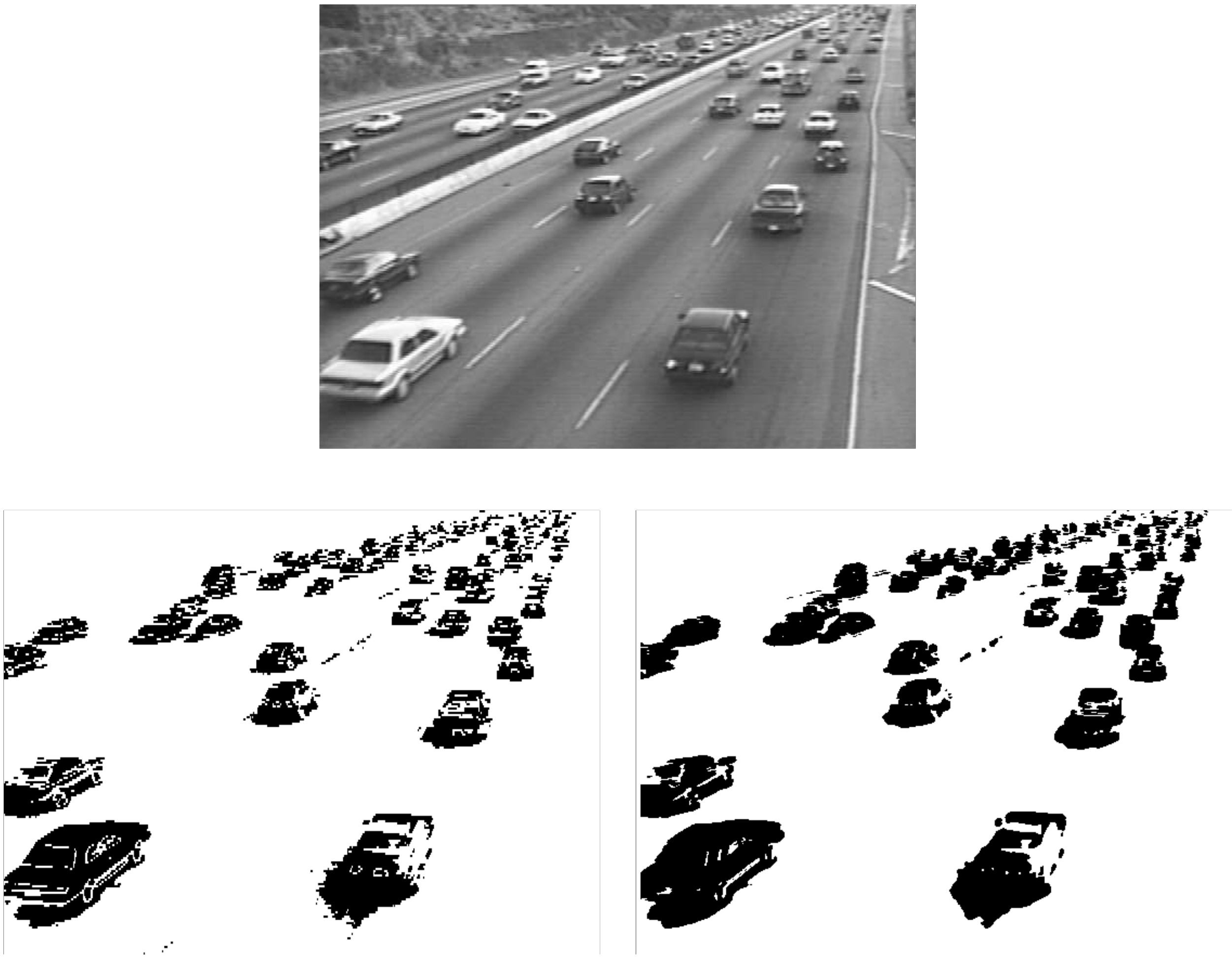


Figure 4: Original image (upper image) and motion hypothesis images produced by an intensity differencing method (left) and filtered differencing method (right). The filter method attempts to increase the signal to noise ratio in the difference image and thus increase the accuracy of the motion detection.

connected objects through merging of connected regions and elimination of small regions. These connected objects are used as the tracking initialization.

Our hypothesis of white noise is only a convenient approximation. Image sequences contain various forms of correlated noise. In the case of shot noise, the second stage where the binary hypothesis mask is reduced to connected components of significant size helps reduce the influence of isolated pixel fluctuations. Also implicit in the threshold levels chosen is that changes in overall brightness due to lighting changes or camera auto-aperture effects produce responses below the Gaussian filter threshold.

3 The Affine Motion Model

The motion segmentation step is assumed to provide image patches due to projections of single moving 3D objects. Now as already pointed out by other researchers, for a sufficiently small field of view and independently moving objects, the image velocity field and change in apparent image shape for smooth surfaces is well approximated by a linear (affine) transformation (i.e. [Koenderink 86]). Thus affine motion models are widely used in computer vision for motion segmentation and tracking (i.e. [Cipolla & Blake 92; Murray *et al.* 93; Zheng & Chellappa 93]).

We assume that the apparent image motion $\boldsymbol{u}(\boldsymbol{x})$ at location $\boldsymbol{x}$ inside a detected image patch can be well approximated by the following affine motion:

$$\boldsymbol{u}(\boldsymbol{x}) = A\,(\boldsymbol{x} - \boldsymbol{x}_m) + \boldsymbol{u}_0, \tag{3}$$

with $\boldsymbol{x}_m$ the center of the patch, $\boldsymbol{u}_0$ the displacement of $\boldsymbol{x}_m$, and A a rotation and scaling matrix. Since we expect the rotation component to be small (the motion of the vehicles is constrained on a road plane and possible rotations are only along the normal of the plane), we end up with a single change in scale in the matrix A, which corresponds to a motion along the optical ($z-$) axis. Small rotational motion components may effect the shape of the contour due to a changed aspect of the 3D object and are well captured by an appropriate process noise. Equation 3 reduces then to:

$$\boldsymbol{u}(\boldsymbol{x}) = s\,(\boldsymbol{x} - \boldsymbol{x}_m) + \boldsymbol{u}_0, \tag{4}$$

with the scale parameter s. $s = 0$ stands for no change in scale, while $s < 0$ and $s > 0$ stands for a motion component along the optical axes away and towards the camera, respectively. The same consideration concerning the affine motion holds also, of course, for local image structures and image features.

We further tried to exploit the fact that the motion of road vehicles can be locally approximated by a pure translational velocity $\boldsymbol{V}$ on a ground plane. From this assumption it is possible to derive some motion invariants expressed in the affine motion parameters s and $\boldsymbol{u}_0$ in order to state the system dynamic equations for predicting the parameters in subsequent frames. The appropriate prediction of these affine motion parameters turns out to correspond to first derivatives of their estimates. These first derivatives are known to be noisy in the case of noisy measurements and thus uncertain estimates. As a result the predictions can be wrong and may destabilize the estimation process. We obtained the best results using the simple assumption of constant displacement $\boldsymbol{u}_0$ and constant scale parameter s. Small changes in these parameters are well captured by process noise. This is a known procedure of not trying to estimate model parameters which are expected to fall into the range of the noise. The best tracking results are obtained if these parameters are removed from the system dynamic equations (motion model) and their associated (physical) meaning included in the process noise.

As an additional *measurement* of the affine motion parameters s and $\boldsymbol{u}_0$ we tried to estimate them using a constrained affine optical flow estimation inside each image patch provided from the motion segmentation step. This would also provide an additional test for motion segmentation by rejecting those areas which are not consistent with the average affine motion parameters inside an image patch. It turns out that optical flow estimation using a differential method is too sensitive to vibrations of the camera and provides no reliable results for large image motion as in they appear in the lower part of the image.

4 Contour Extraction and Shape Estimation

For a contour description we exploit motion and greyvalue boundaries. A discrimination of greyvalue boundaries is performed by simply thresholding the spatial image gradient, already estimated in the motion segmentation step. Significant motion areas are obtained by thresholding the time derivative. An initial object description is then obtained by finding a convex polygon enclosing all the sample points of the locations that passed this first test for greyvalue boundaries and motion areas. Figure 4 shows an image section with a car (a), the detected image patch covering the image of the car (b), and the image locations with well defined spatial gradient and time derivative constituting the sample points. The convex polygon enclosing all these sample points of c) is shown in d).

For a time recursive shape estimation, a convex polygon is not suitable, since the number of vertices may and will change along the image sequence with new measurements. This problem may be solved be using a fixed number of vertices, i.e. from the first measurement, as done by [Meyer & Bouthemy 92], but this is a rather ad-hoc method and requires sophisticated distance measures of polygons ([Cox *et al.* 89]). A better solution is provided by *snakes*, spline approximation to contours [Kass *et al.* 88; Curwen & Blake 92; Bascle & Deriche 93]. Splines are defined by control points, for which a constant number can be selected. This makes the shape estimation quite easy by estimating the location and motion of a fixed number of control points.

We use a closed cubic spline approximation of the extracted convex polygon as a final shape description of the moving vehicles. We use 12 *control points* which approximate the polygonal contour by 12 segments (spans) of cubic order. It is not sufficient to use the vertices of the polygon as sample points for the spline approximation, since these vertices have unequal distances and would give rise to *oscillations* at contour locations with sparse sample points as opposed to locations with dense sample points. To obtain smooth contours, we therefore use equidistant sample points along each polygon. The locations of the control points are obtained by a least squares between these equidistant sample points along the contour of the polygon and the (uniform) spline segments ([Bartels *et al.* 87]). The details can be found in Section A.2. The spline approximation of the contour for the previous example is given in Figure 4 e).

5 Recursive Shape and Motion Estimation

As already mentioned in Section 1 we decompose the estimation process into two linear Kalman Filters: one for the motion estimation and one for the shape estimation. The measurements we use are the positions $\{X_i, Y_i\}_{i=1\dots n}$ of the n control vertices that approximate the convex polygon by cubic spline segments. For a concise description we define the following vectors:

$$\begin{aligned} \boldsymbol{\Xi} &= (X_1, Y_1, \dots, X_n, Y_n) \\ \boldsymbol{\xi} &= (u, v, s). \end{aligned} \tag{5}$$

$\boldsymbol{\Xi}$ stands for the state vector of the vertices and $\boldsymbol{\xi}$ for the state vector of the affine motion parameters. The covariance matrices associated with these state vectors are: Π, the covariance for $\boldsymbol{\Xi}$ and π, the covariance for $\boldsymbol{\xi}$. We furthermore indicate the time t_k with an index k to each variable.

The plant equation for the control vertices can be derived from the affine motion model (Eqn. 4) by integration to give:

$$\hat{\boldsymbol{X}}^-_{i,k+1} = \hat{\boldsymbol{X}}^+_{i,k} + s_k(\hat{\boldsymbol{X}}^+_{i,k} - \boldsymbol{C}_k) + \boldsymbol{u}_k, \tag{6}$$

with $i \in \{1, \dots, n\}$. $\boldsymbol{C}_k = (C_{x,k}, C_{y,k})$ denotes the center of all n control vertices. Here we adopt the usual dynamical systems notation (see, e.g., [Gelb 74]), where $\hat{\boldsymbol{\Xi}}^-_{k+1}$ denotes a prediction from time t_k to time t_{k+1} before an update and $\hat{\boldsymbol{\Xi}}^+_k$ is the update at time t_k. Using the definition (5) we can put Eqn. 6 into matrix form,

$$\hat{\boldsymbol{\Xi}}^-_{k+1} = \hat{\boldsymbol{\Xi}}^+_k + H_k \hat{\boldsymbol{\xi}}^+_k, \tag{7}$$

with,

$$H_k = \begin{pmatrix} \boldsymbol{I}_2 & (\hat{\boldsymbol{X}}^+_{1,k} - \boldsymbol{C}_k) \\ \vdots & \vdots \\ \boldsymbol{I}_2 & (\hat{\boldsymbol{X}}^+_{n,k} - \boldsymbol{C}_k) \end{pmatrix}. \tag{8}$$

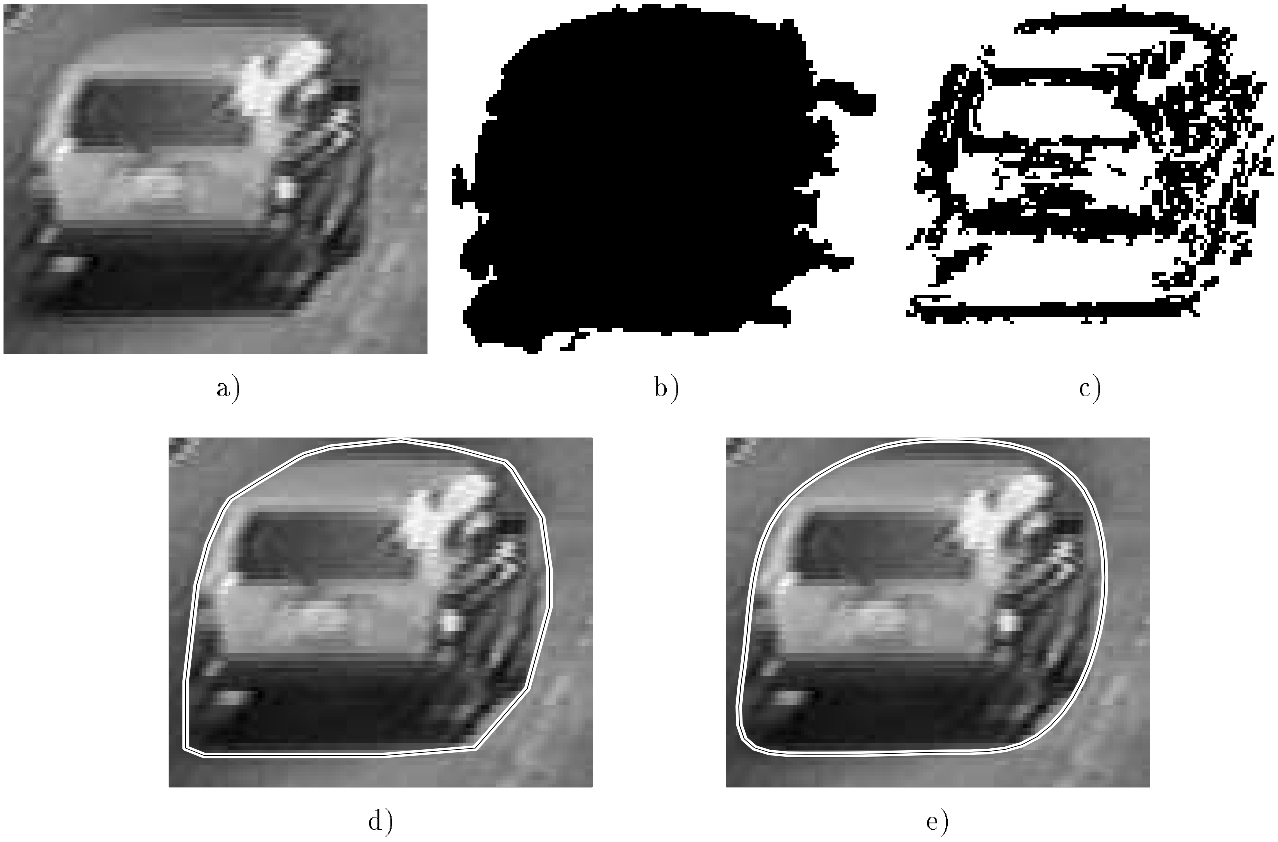


Figure 5: a) An image section with a moving car. b) the moving object mask provided by the motion segmentation step. c) the image location with well defined spatial gradient and temporal derivative, used as sample points to define d), the convex polygon enclosing these points of c). e) the final contour description by cubic spline segments approximating the polygon of d).

5.1 Motion Estimation

The parameters for the motion estimation are the affine motion parameters $\boldsymbol{\xi} = (\boldsymbol{u}, s)$. The plant equation for the motion parameters is simply:

$$\hat{\boldsymbol{\xi}}^{-}_{k+1} = \hat{\boldsymbol{\xi}}^{+}_{k} + \boldsymbol{q}_k, \tag{9}$$

with a white Gaussian process noise $\boldsymbol{q}_k$ with covariance Q_k: $\boldsymbol{q}_k \sim \mathcal{N}(\boldsymbol{0}, Q_k)$.

As a measurement vector we use the positions of the control vertices:

$$\boldsymbol{Z}_k = (X^z_1, Y^z_1, \ldots, X^z_n, Y^z_n), \tag{10}$$

the x and y The measurement equation reads as:

$$\boldsymbol{Z}_k = \boldsymbol{h}_k(\hat{\boldsymbol{\xi}}^{-}_k) + \boldsymbol{v}_k = H_k \hat{\boldsymbol{\xi}}^{-}_k + \boldsymbol{v}_k, \tag{11}$$

where each measurement is assumed to be corrupted by a white Gaussian noise $\boldsymbol{v}_k$ with known covariance matrix $R_{\boldsymbol{\Xi},k}$: $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, R_{\boldsymbol{\Xi},k})$.

Since we have a linear measurement function $\boldsymbol{h}_k(\boldsymbol{\xi}_k)$ with $\dfrac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{\xi}} = H_k$ we can use the simple (linear) Kalman Equations to obtain the state update $\hat{\boldsymbol{\xi}}^{+}_k$ with associated covariance matrix update

π_k^+ ([Gelb 74]):

$$\begin{aligned} \hat{\boldsymbol{\xi}}_k^+ &= \pi_k^+ \left(H_k^T R_{\boldsymbol{\Xi},k}^{-1} \boldsymbol{Z}_k + (\pi_k^-)^{-1} \hat{\boldsymbol{\xi}}_k^- \right), \\ \pi_k^{+^{-1}} &= H_k^T R_{\boldsymbol{\Xi},k}^{-1} H_k + (\pi_k^-)^{-1}. \end{aligned} \tag{12}$$

5.2 Shape Estimation

The second linear filter applies to the shape estimation, i.e. tracking the n control points of the splines describing the contour of the moving vehicles. The state vector comprising the control points is defined in Eqn 5. As a measurement vector we use the same $\boldsymbol{Z}_k$ (10) of Section 5.1. The measurement function is identical to the state vector itself, so we have again simple linear update equations:

$$\begin{aligned} \hat{\boldsymbol{\Xi}}_k^+ &= \Pi_k^+ \left(R_{\boldsymbol{\Xi},k}^{-1} \boldsymbol{Z}_k + (\Pi_k^-)^{-1} \hat{\boldsymbol{\Xi}}_k^- \right), \quad \text{with} \\ \Pi_k^{+^{-1}} &= R_{\boldsymbol{\Xi},k}^{-1} + \Pi_k^{-^{-1}}. \end{aligned} \tag{13}$$

5.3 Initialization

A very important issue is the initialization of a tracker. The stability of the tracker depends from the quality of the initialization data. In the case of a motion model with constant velocities, the tracker can be initialized after the second appearance of an object. For that purpose initial values for $\boldsymbol{u}_0$ of an object according to the affine motion model are derived by discrete time derivatives of the objects center locations $\boldsymbol{C}_k$ in the first two frames. As an initial value for the scale parameter s we set $s = 0$, since this parameter is usually very small;

$$\boldsymbol{u}_k = \boldsymbol{C}_k - \boldsymbol{C}_{k-1} \tag{14}$$

$$s_k = 0 \tag{15}$$

In order to identify segmented image patches from the motion segmentation step as projections of the same object during the initialization frames (the motion segmentation step provides only unique labels for an image patch in one frame and the label of the patch associated to the same object might change in the next frame), we identify the label number of the image patch associated to the same object in the next frame by finding the largest overlapping region of two patches in subsequent frames. This is actually a simple nearest neighbor approach in the context of data association for tracking ([Rao 92]).

5.4 Tracking Procedure

After the initialization step (usually after the second appearance of an object) we switch to the tracking mode, where we use the predicted contour as a mask for finding sample points providing a new contour as a measurement. The mask is the area defined by the predicted contour of the object expandend by factor of about 5% in order to allow the contour to increase after a not optimal initialisation. The motion segmentation step is thus only necessary during the initialization, which is usually performed for objects coming into the field of view and has therefore to be performed only in the lower part of the image. Initialization is performed full automatically without any user interference. The procedure is as follows:

1. Look for new blob labels that do not significantly overlap with the contour of an object already to be tracked.

2. Analyse all new found labels.

3. Move a object after the second appearance from the initialisation list into the tracking list.

4. Analyse all objects in the tracking list by handling different occlusion cases as described in Section 6.

6 Occlusion Reasoning

We are interested in extracting tracks of moving vehicles. These tracks are obtained by connecting the center of contours along the image sequence. Any contour distortion will therefore generate an artificial shift in the trajectory. In order to avoid those artificial shifts and to get reasonable tracks from the contours, we use an occlusion reasoning step.

An occlusion reasoning step is feasible if locations and extensions of other objects in the environment are known. In the case of tracking multiple objects we have to analyze the objects in an order according to their depth (distance to the camera), a common procedure in computer graphics to remove hidden surfaces. The depth ordered objects define the order in which objects are able to occlude each other. Our depth ordering procedure exploits the fact that the motion is assumed to be constraint on the ground plane. To prove the correct performance of this procedure we first assume that the z-axis of the camera is parallel to ground plane $y_c = -h$, where h is the height at which the camera is mounted above the road. The image coordinates for an object moving on the ground plane are then $(x', y')^T = (f\frac{x_c}{z_c}, f\frac{y_c}{z_c})^T = (f\frac{x_c}{z_c}, -f\frac{h}{z_c})^T$ which states that the larger the distance (the larger z_c) the smaller the y component in the image. The same consideration holds in a general case where the camera is rotated along an axis parallel to the ground plane by an inclination angle α towards the ground plane. In this case we have to distinguish between the camera coordinates $\boldsymbol{x}_c = (x_c, y_c, z_c)^T$ and the world coordinates $\boldsymbol{x}_w = (x_w, y_w, z_w)^T$ (see Figure 6). Let the ground plane be the x_w-z_w plane. For the y component of the image coordinate we get

$$y' = f\frac{y_c}{z_c} = f \tan \delta. \tag{16}$$

According to angles asigned in Figure 6 we find $\delta = \gamma - \alpha$ with $\gamma = \arctan \frac{h}{z_w}$ and finally

$$y' = f \tan(\arctan \frac{h}{z_w} - \alpha), \tag{17}$$

which is a monotonic function for $\| \arctan \frac{h}{z_w} - \alpha \| < \frac{\pi}{2}$ and thus proves the assumption that in the case of a planar motion a depth ordering of the object can be obtained by simply considering their y-coordinate in the image.

The occlusion reasoning step also improves robustness in cases which we call *near* occlusion. These are cases where objects come very close to each other or where objects move very close next to each other so that the contours of the object will interfer and the contour estimation process will be confused by the presence of other object. Knowledege about the other contour provides now the means to avoid imaga data from another object to be considered in the contour estimation of the object under investigation. For that purpose we remove the image areas covered by all other objects in the contour estimation mask[1].

The occlusion reasoning procedure is as follows:

1. Sort the objects in the tracking list by their y-coordinate of the center of the predicted contour.

[1] A mask is a binary image which states at which locations in the (sub)image the pixels are allowed to contribute to the contour estimation.

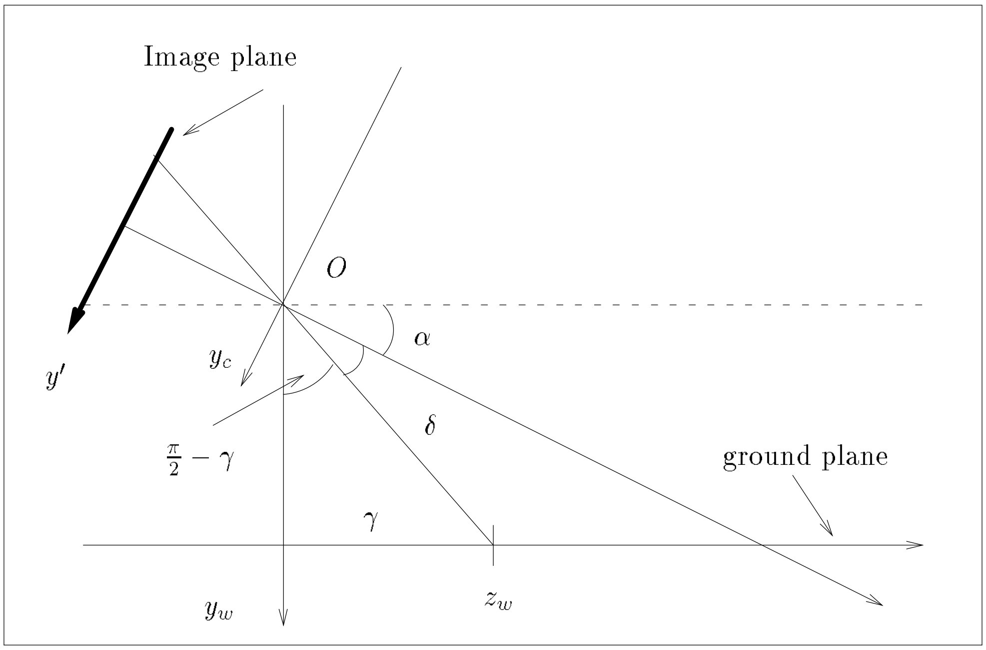


Figure 6: The camera geometry to clarify the proof that a depth ordering of objects on a plane can be determined by the y component of their image location.

2. Look for overlapping regions of the predicted contours and decide in the case of an overlapping region — according to the y-position — if the object *is occluded* or if the object *occludes* another object.

3. Analyse all objects in the tracking list by handling the different occlusion cases: In the case that the object under investigation occludes another object, we remove the image area associated of the other object in the contour estimation mask but keep the overlapping part; in the case of the object under investigation being occluded by another object, we remove the area associated of the other object and use the predicted contour in this region as the new measurement to update the motion and shape parameters.

This process is illustrated in Figure 7 with an example in Figure 8.

7 Results with Real World Traffic Scenes

In order to validate our approach we conducted several experiments. Due to lack of space we present here the result of tracking cars in two image sequences only. For both sequences we used the following parameters: $\sigma = 1.0$ for first derivative of Gaussian for convolution; as threshold for the spatial and temporal derivatives inside an image patch we used $\tau_g = 2.0$ and $\tau_t = 20.0$, respectively. For the affine motion estimation we set:

$$\pi_0 = \begin{pmatrix} 30.0 & 0.0 & 0.0 \\ 0.0 & 30.0 & 0.0 \\ 0.0 & 0.0 & 30.0 \end{pmatrix}, \qquad Q_k = \begin{pmatrix} 3 \cdot 10^{-3} & 0.0 & 0.0 \\ 0.0 & 3 \cdot 10^{-3} & 0.0 \\ 0.0 & 0.0 & 1 \cdot 10^{-8} \end{pmatrix} \tag{18}$$

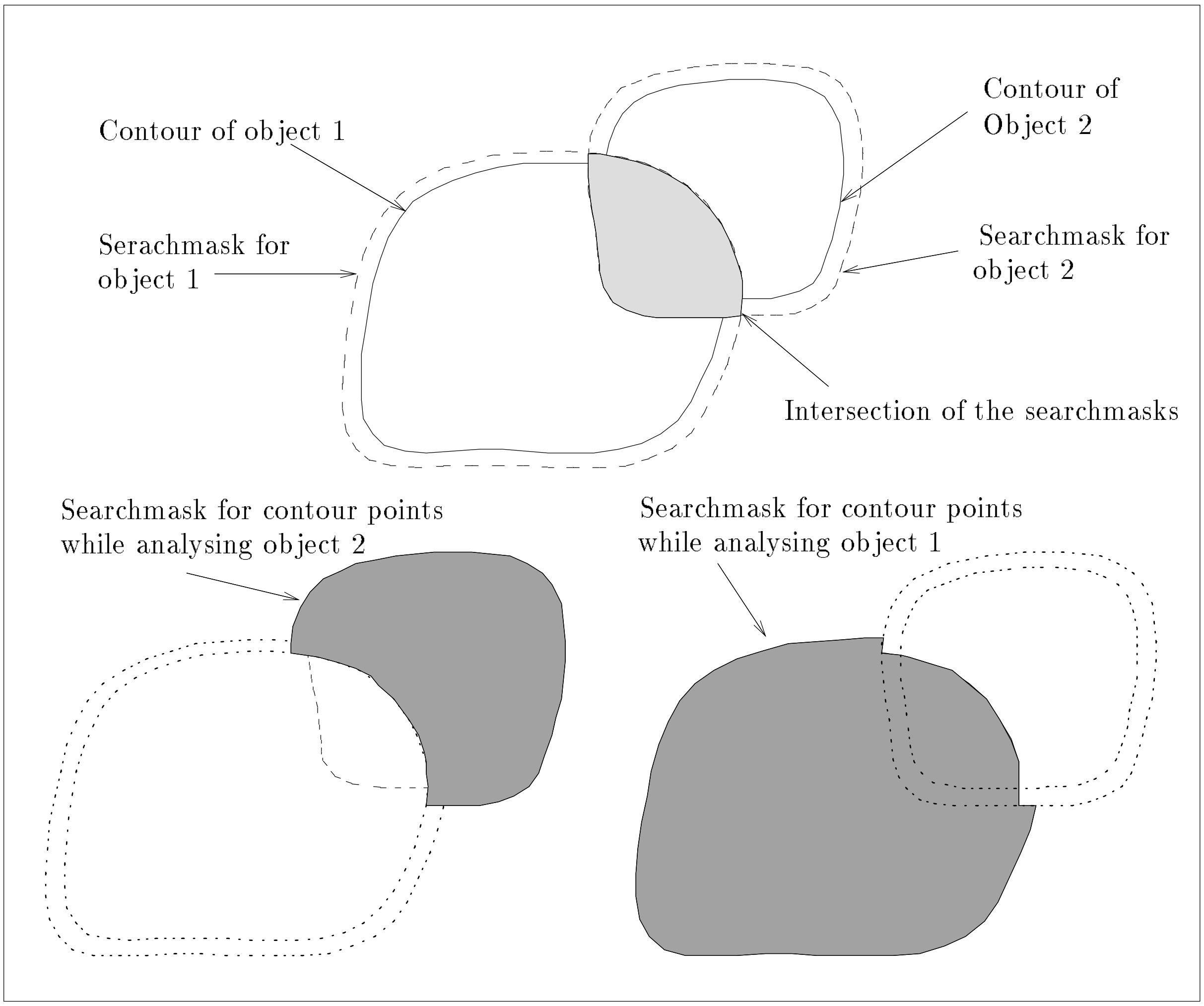


Figure 7: The upper figure shows the (filled) intersection polygon of convex polygons associated to two objects. These dashed contours represent the search area for finding contour points and is about 5% enlarged compared to the polygon descibing the contour of the object in order to ensure that also an expansion of the objects size is possible. The lower left image shows the final mask for finding the contour of object 2 which is assumed to be occluded from object 1, since object 1 has a lower position. The lower right image shows the final mask for finding the contour of object 1.

as an initial covariance for $\boldsymbol{\xi} = (u, v, s)$, and process noise covariance, respectively. In the shape estimation we assume independent measurement noise with $v = 0.4$ pixel for each vertex and a start covariance with

$$\Pi_0 = \begin{pmatrix} 10.0 & \cdots & 0.0 \\ \vdots & \vdots & \vdots \\ 0.0 & \cdots & 10.0 \end{pmatrix}, \qquad (19)$$

As a process noise for each vertex we set $w = 0.001$. The number of control vertices was always set to $n_v = 12$.

As a first experiment we evaluated an image sequence of 84 frames recorded from an overpass of a divided 4 lane freeway. The left columns of Figure 9 and Figure 10 show a few images of

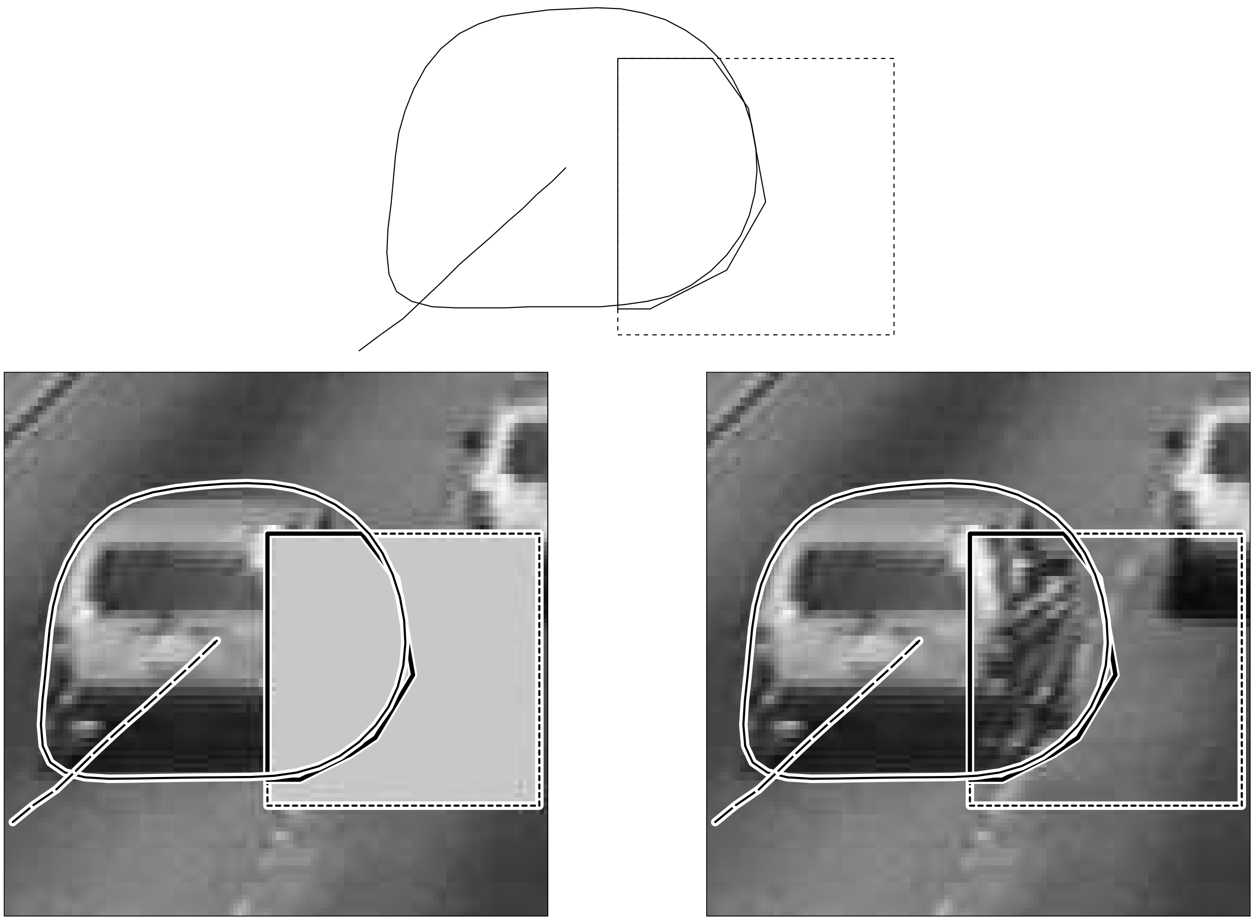
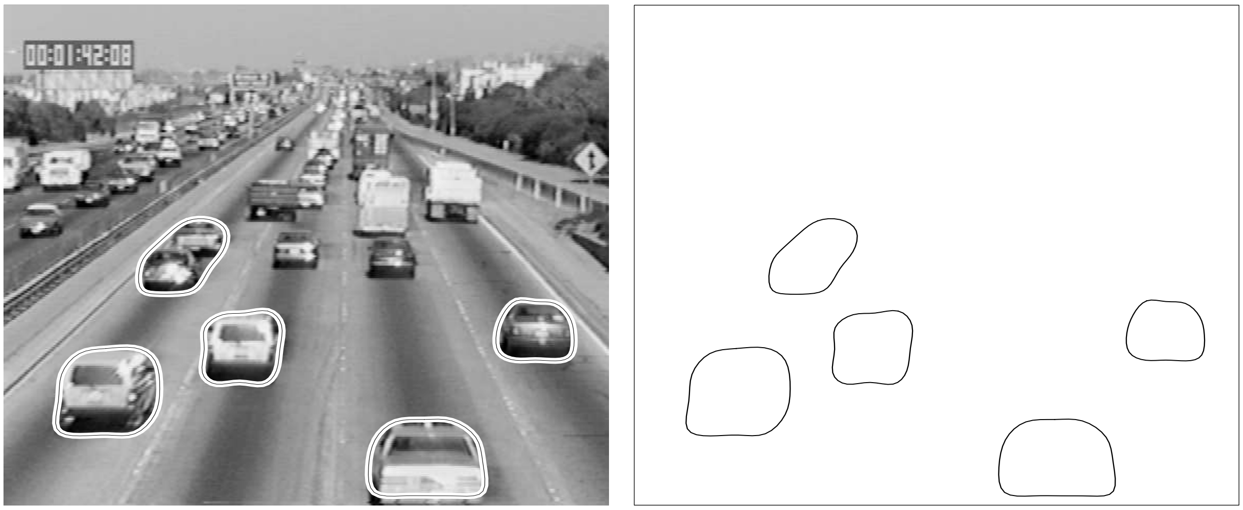


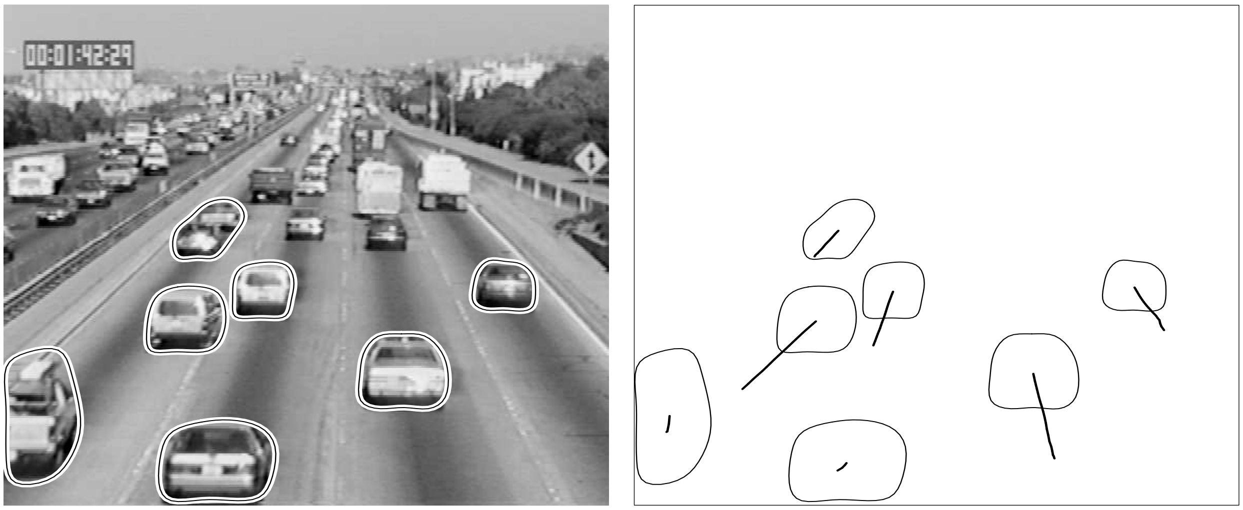
Figure 8: In order to elucidate the occlusion reasoning in detail we generated an artificial occlusion in an image sequence. The occlusion is represented by a grey square. The upper image shows the (artificial) occluding object by dashed lines. The occluding contour is shown in thick lines, while the contour estimate of the object to be tracked is shown by thin lines. The vectors represent the trajectory of the object. The lower left image shows the same lines overlayed in the image including the occlusion while the lower right image shows the lines overlayed to the original image without occlusion in order to compare the result.

the sequence with the overlayed contour estimates of the cars. The right columns of Figure 9 and Figure 10 show the contour estimates with the tracks. In this image sequence we have some near occlusion and a partial occlusion. The contour estimate in the upper left part of the image captures two cars. This is due to the fact that the images of these cars are already overlapping from the start of the image sequence. Initialization should only be done in the lower part of the image where there is a low probability of an occlusion.

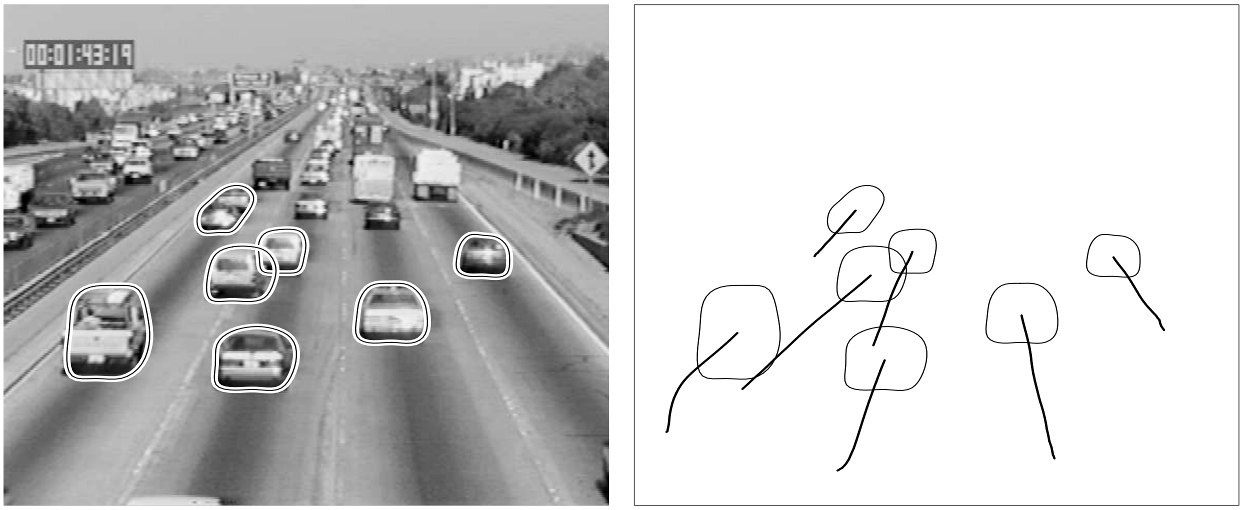
The second image sequence captures a similar scene of about 96 frames with a divided 4 lane freeway, going slightly uphill. This sequence is also recorded from an overpass but with a slightly different view angle. The left columns of Figure 11 and Figure 12 show again some intermediate images of the sequence with the overlayed contour estimates. The right columns of Figure 11 an Figure 12 show the contour estimates with the tracks. Due to a slightly different view angle and less traffic we have only near occlusions.



Frame #0

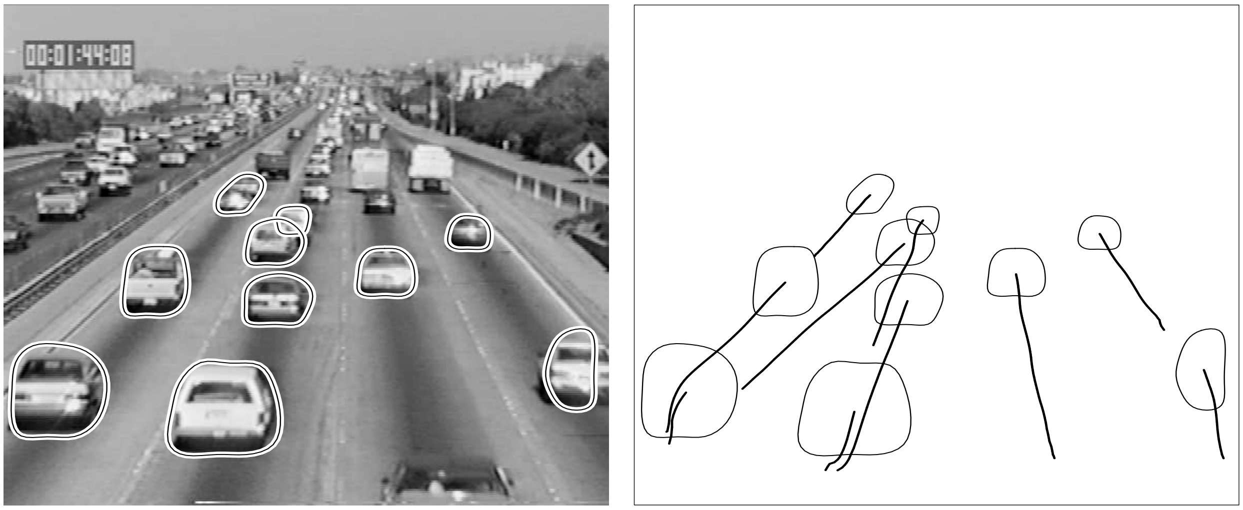


Frame #20

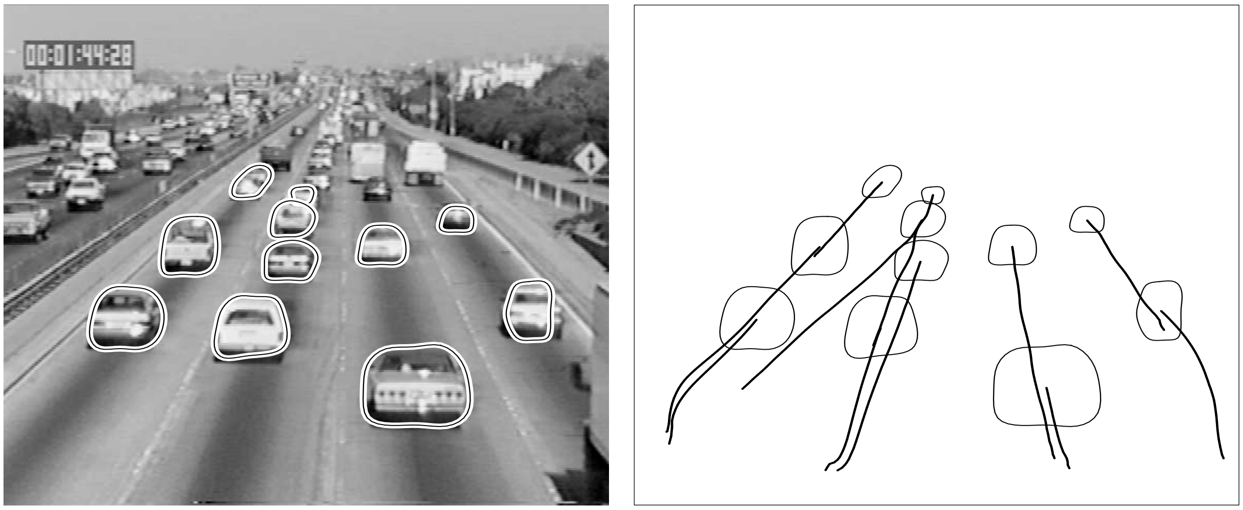


Frame #40

Figure 9: The left column shows frame #0, #20, and #40 of the image sequence with the overlayed overlayed contour estimates of the cars. The right column shows the contour estimates with their tracks, starting from frame #0.

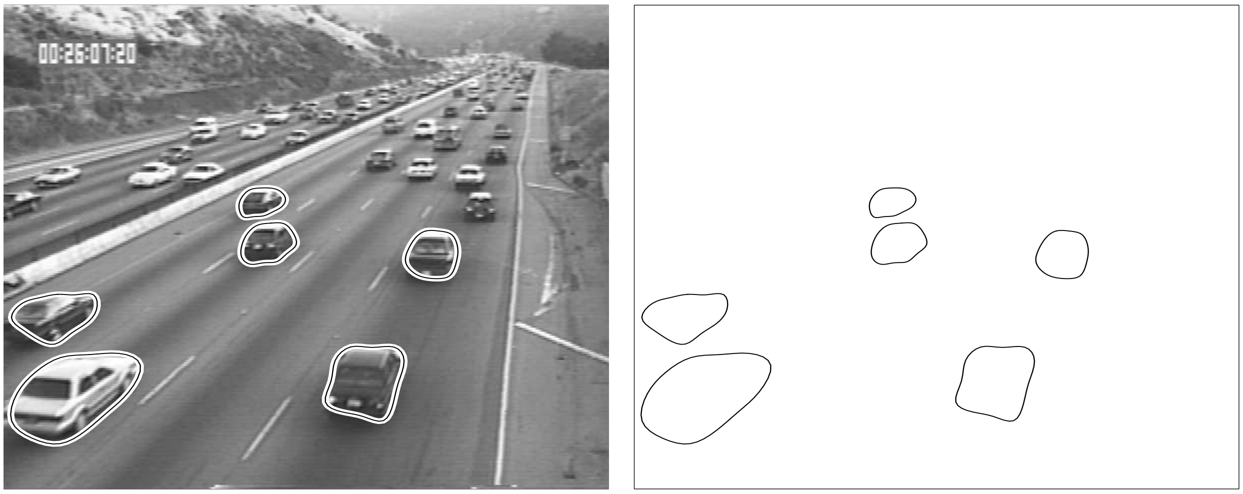


Frame #60

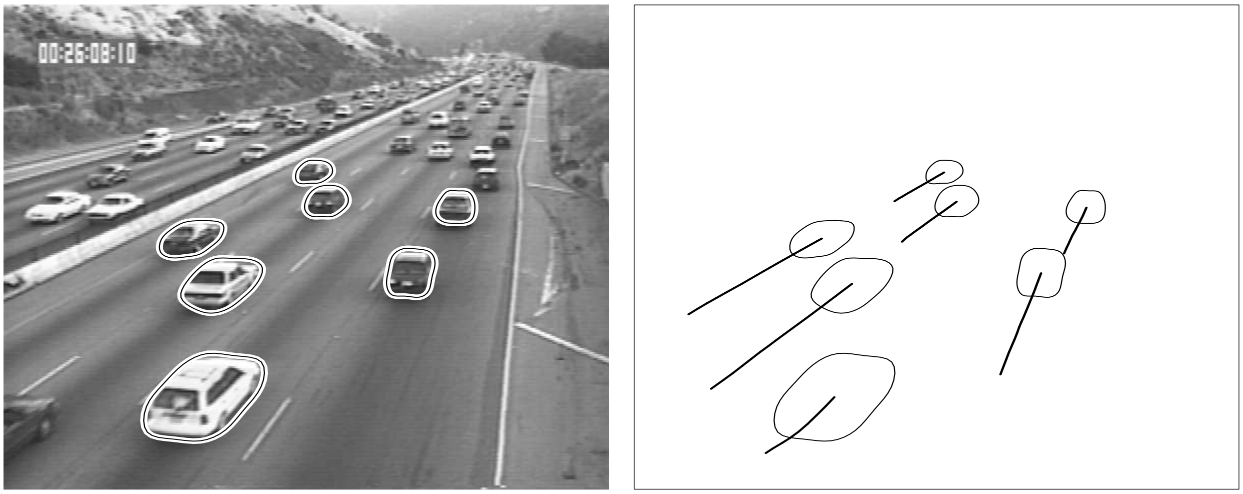


Frame #80

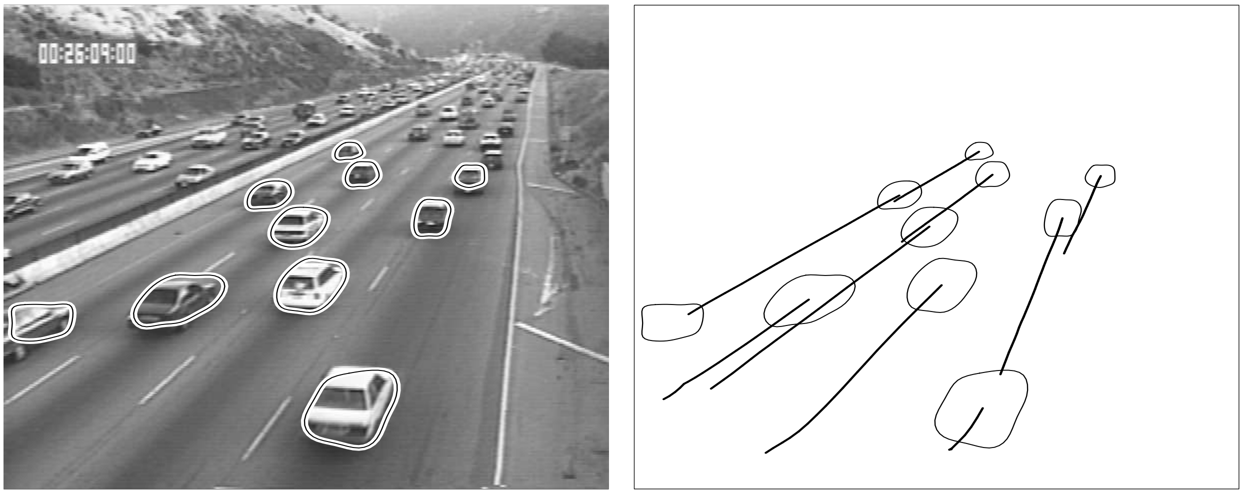
Figure 10: The same like Figure 9 for frame #60 and #80.



Frame #0

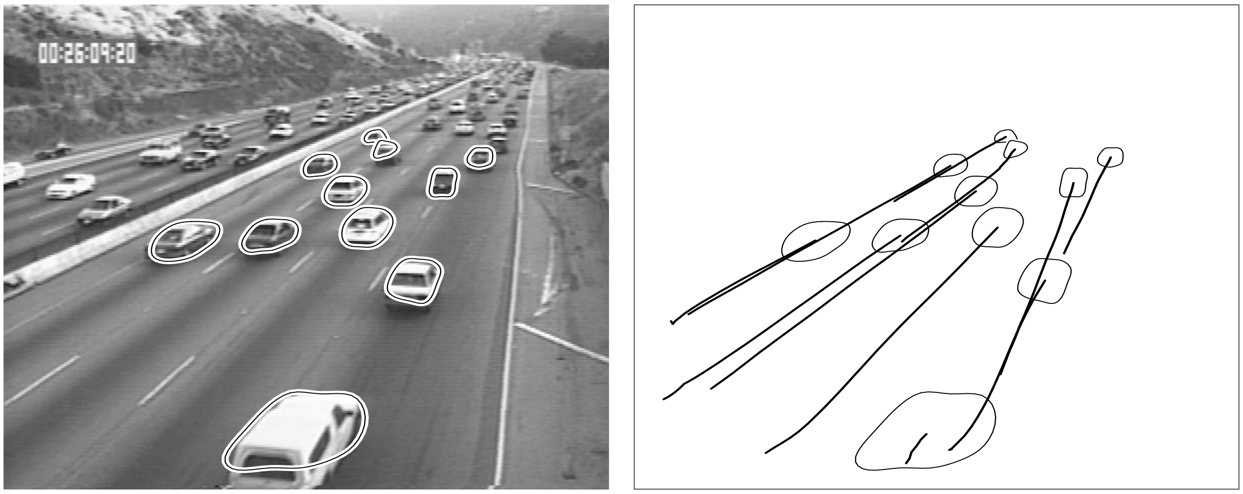


Frame #20

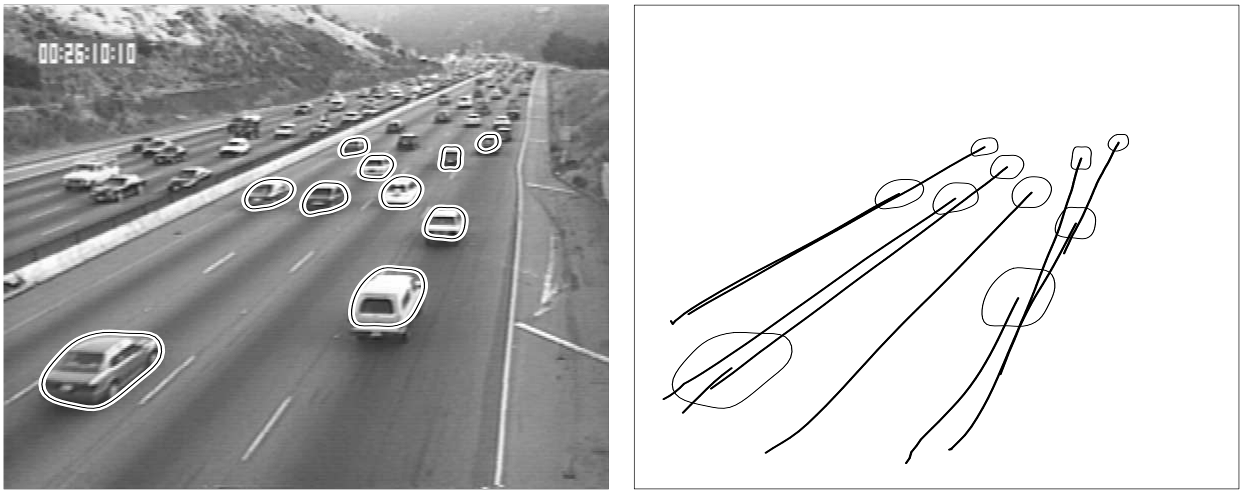


Frame #40

Figure 11: The left column shows frame #0, #20, and #40 of the image sequence with the overlayed overlayed contour estimates of the cars. The right column shows the contour estimates with their tracks, starting from frame #0.



Frame #60



Frame #80

Figure 12: The same like Figure 11 for frame #60 and #80.

8 Conclusion

We designed a system for robust detection and tracking of multiple vehicles in road traffic scenes. The system provides tracks and shape description of vehicles which are suitable for further evaluation with symbolic reasoning in a traffic surveillance system. As typical for a surveillance system, the recording camera is assumed to be stationary, mounted, i.e., on a bridge or a pole beside the road, in order to cover a large field of view and to reduce occlusions of vehicles. Running on a Sun SparcStation 10, the performance reaches about 5 seconds per frame for simultaneously tracking of about 10 vehicles (this is including reading images from disc and producing graphical output). A (near) real-time implementation on a special hardware using C-40 Digital Signal Processors is under investigation.

Reliable trajectories of vehicles in a highly cluttered environment like in highway scenes with heavy traffic are obtained by considering an occlusion reasoning step. The new contribution of this work consists of an occlusion reasoning step which solves the problem of data association for a multitarget tracking application exploiting a-priori knowledge about constraint motion: the assumption of moving objects on a float road allows to compute a depth order of the image regions associated to the moving objects. According to the depth order we finally know which object can occlude other objects. Since the occlusion reasoning step is performed on slightly enlarged areas representing the image of a moving vehicle, this procedure also improves the shape estimation while tracking adjacent vehicles in the image (where the contour estimation might be corrupted by the presence of the other object).

In order to get robust results and reduced noise sensitivity, we exploit a certain amount of a-priori knowledge about the objects and the motion: the objects are assumed to be well describable by convex contours and the motion is expected to be predominantly translational on a plane. We describe the contour by a closed cubic spline (well known as *snakes*) and use an affine motion model for the innovation of the contour. The tracker is based on two simple Kalman Filters, estimating the affine motion parameters and the control points of the closed spline contour of the vehicles. As measurements we take the control points of a spline contour approximating a convex polygon, obtained by greyvalue and motion boundaries.

The initialization of the tracker is performed by a kind of image differencing between a continuously updated background image and a newly acquired image. Update of the background image is based on the motion estimate. Trajectories of the moving vehicles are derived from the motion of the center of the control points of the closed spline contour. In the case of a partial occlusion, the center of the control points does not provide a reliable feature for the trajectory of an object, since the contour will be corrupted by wrong contour measurements. We solve this problem by an explicit occlusion reasoning step. Occlusion detection is performed by detecting overlapping regions enclosed by a closed spline contour. Overlapping areas are then specifically analyzed in the motion and shape estimation dependent on the type of occlusion: the object can be occluded by another object, or the object can occlude another object. Since the recording cameras are assumed to be mounted high above the road, we do not expect any total occlusion. But this approach should also be able to track vehicles during total occlusion without any additional recovering step with a linkage of the trajectories after reappearance of the vehicle. Appropriate experiments are under investigation.

9 Acknowledgments

We gratefully acknowledge the help of C. McCarley and his group at at Cal Poly, San Luis Obispo in providing us with video tapes of traffic scenes covering all kind of traffic conditions. We also like to thank B. Rao for discussions about data association and multitarget tracking. This work was supported by California Department of Transportation through the PATH project grant no. MOU-83.

A Appendix

A.1 Convex Hull of Points in a Pixmap

As opposed to finding the convex hull of general points in two dimensions we can exploit the fact that points in a pixmap are already sorted by x and y coordinates. We can therefore devise an algorithm which is much simpler than one in the general case (cf. Figure 13). We start from the first scanline and look for the leftmost and rightmost point in each scanline. Let $\boldsymbol{p}_{i-2} = (x_{i-2}, y_{i-2})$ and $\boldsymbol{p}_{i-1} = (x_{i-1}, y_{i-1})$ be the latest two points associated to the convex hull and let the selected leftmost/rightmost point of the current scanline be $\boldsymbol{p}_i = (x_i, y_i)$. Then remove $\boldsymbol{p}_{i-1}$ as a vertex of the convex hull, if

$$D_i \left\{ \begin{array}{ll} \leq 0 & \text{for } \boldsymbol{p}_i \text{ on the left side} \\ \geq 0 & \text{for } \boldsymbol{p}_i \text{ on the right side} \end{array} \right. \tag{20}$$

with

$$D_i = \left| \begin{array}{ccc} x_{i-2} & y_{i-2} & 1 \\ x_{i-1} & y_{i-1} & 1 \\ x_i & y_i & 1 \end{array} \right|, \tag{21}$$

which actually determines the orientation of the three points.

A.2 B-Spline Approximation of Convex Polygons

The contour of a moving object is provided by the convex hull enclosing all candidate contour points inside the object's mask. This convex hull is represented by the vertices of the associated convex polygon. Since the number of these vertices can change along the image sequence while the object may move and change its scale and shape, the vertices do not provide an appropriate set of features to track. For this reason we represent the tracking contour by a closed cubic spline approximation to the convex polygons. Splines can be represented by fixed number of so-called *control points* which define the shape of the spline contour.

A uniform B-Spline of degree k (which is order $k-1$) represented by M control points can be written as ([Bartels *et al.* 87]):

$$\boldsymbol{Q}(u) = \begin{pmatrix} X(u) \\ Y(u) \end{pmatrix} = \sum_{j=0}^{M-1} \boldsymbol{V}_j B_j^k(u) = \sum_{j=0}^{M-1} \begin{pmatrix} X_j \\ Y_j \end{pmatrix} B_j^k(u), \qquad 0 \leq u \leq M-1. \tag{22}$$

The $B_j^k(u)$ are shifted copies of the so-called *Blending*-functions $\{b_0^k(t), \cdots, b_{k-1}^k(t)\}$ with $0 \leq t \leq 1$. They make sure that the support of each span is only along k subsequent control points:

$$B_j^k(u) = \left\{ \begin{array}{ll} b_{\mathrm{int}(u)-j}^k(t) & \text{for} \quad j \leq \mathrm{int}(u) \leq j+k \\ 0 & \text{else} \end{array} \right. \tag{23}$$

with $t = u - \mathrm{int}(u)$, $0 \leq t \leq 1$. $\mathrm{int}(u)$ is the integer value of parameter u. For cubic splines ($k=4$) we find (see e.g. [Bartels *et al.* 87]):

$$b_0^4(t) \;=\; \frac{1}{6}t^3$$

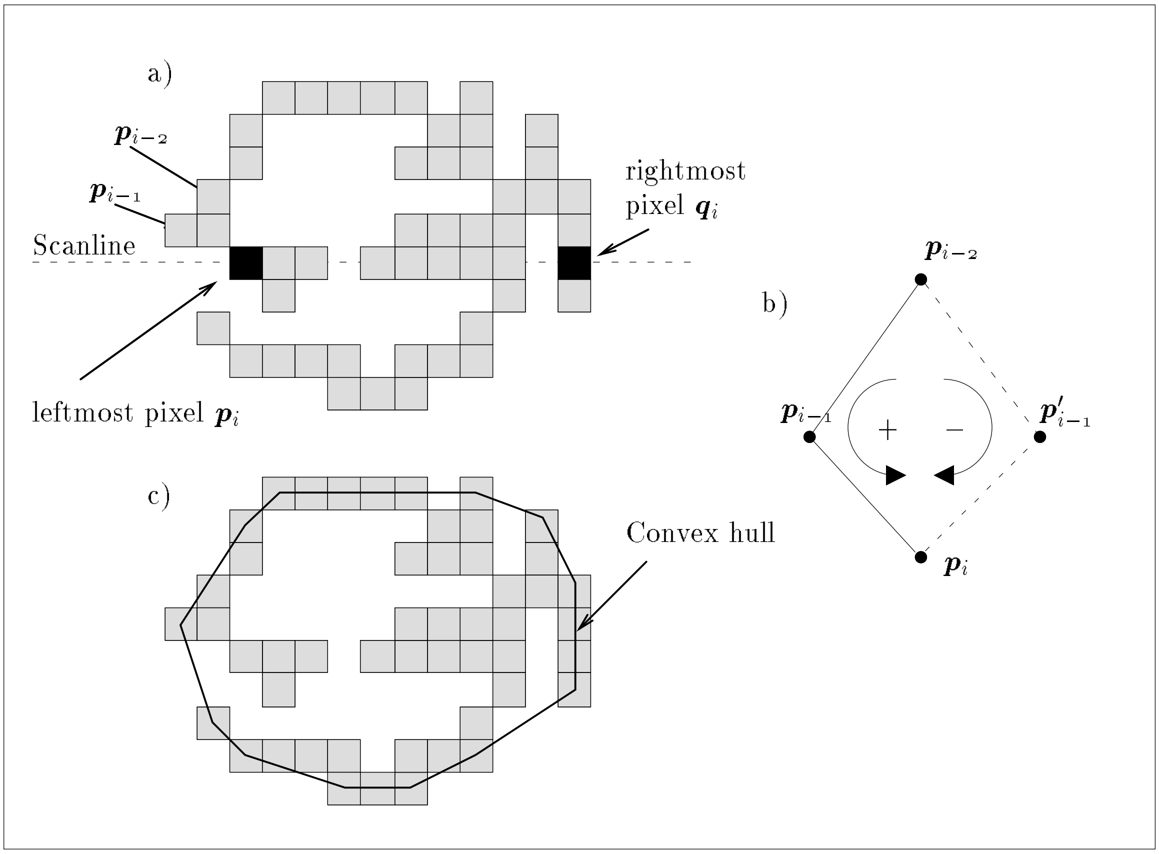


Figure 13: The upper left figure (a) shows the leftmost pixel $\mathbf{p}_i$ and the rightmost pixel $\mathbf{q}_i$ of a scanline with the last two pixels associated to the left side of the convex polygon: $\mathbf{p}_{i-2}$ and $\mathbf{p}_{i-1}$. The right figure (b) shows the two cases for $\mathbf{p}_i$ giving D_i a positive value (clockwise order of the points — full line) or a negative value (counterclockwise order of the points in the triangle — dashed line). The lower figure (c) shows the final convex polygon.

$$\begin{aligned} b_1^4(t) &= \frac{1}{6}(1 + 3t + 3t^2 - 3t^3) \\ b_2^4(t) &= \frac{1}{6}(4 - 6t^2 - 3t^3) \\ b_3^4(t) &= \frac{1}{6}(1 - 3t + 3t^2 - t^3) \end{aligned} \tag{24}$$

The *curve* parameter u generates the spline contour as it runs from 0 to $M - 1$. For closed contours we find:

$$\boldsymbol{Q}(0) = \boldsymbol{Q}(M-1). \tag{25}$$

For a B-Spline of degree k a *span* of the curve is defined for each k subsequent control points $\{\boldsymbol{V}_j, \boldsymbol{V}_{j+1}, \cdots, \boldsymbol{V}_{j+k-1}\}$. The convention is that each span is obtained by varying the curve parameter u from one integer value to the next integer value, e.g. the first span ranges from $0 \leq u \leq 1$. For an open spline curve we obtain $M + 1 - k$ spans whereas we obtain M spans for a closed spline, since the control points are used cyclic to generate the curve ($\boldsymbol{V}_{P-1+j} = \boldsymbol{V}_j$).

Our task is now to fit a B-Spline curve with M control points $\{\boldsymbol{V}_j\}_{j=0,\cdots,M-1}$ to P *sample points* $\{\boldsymbol{P}\}_{i=0,\cdots,P-1}$. The sample points are not the vertices of our polygon but equidistant points along each segment of the polygon. In this way we do not approximate a spline to the vertices but to the polygon defined by the vertices. Next have to parameterize the spline curve in an appropriate way in order to associate a certain location on the curve to each sample point. We follow the approach in [Bartels *et al.* 87] for fitting an *open* spline to sample points and denote the parameter associated to sample point $\boldsymbol{P}_i$ as $\hat{u}_i$ and use the following parameterization, which is better than using simple uniform splines (equidistant $\hat{u}_i$)[2]:

$$\begin{aligned} \hat{u}_0 &= 0 \\ \hat{u}_{i+1} &= \hat{u}_i + M\frac{\|\boldsymbol{P}_{i+1} - \boldsymbol{P}_i\|}{S}, \qquad \text{with} \qquad S = \sum_{i=1}^{P-1} \|\boldsymbol{P}_i - \boldsymbol{P}_{i-1}\|. \end{aligned} \tag{26}$$

As an objective function to minimize we define the following residual error as the sum of distances between the sample points and the spline curve (the coordinates of the control points are $\boldsymbol{V}_j = (X_j, Y_j)^T$ and the coordinates of the sample points are $\boldsymbol{P}_i = (x_i, y_i)^T$):

$$\begin{aligned} R &= \sum_{i=0}^{P-1} \|\boldsymbol{Q}(\hat{u}_i) - \boldsymbol{P}_i\|^2 \\ &= \sum_{i=0}^{P-1} \left\| \sum_{j=0}^{2} \boldsymbol{V}_j \cdot \left(B_j^k(\hat{u}_i) + B_{j+M}^k(\hat{u}_i)\right) + \sum_{j=3}^{M-1} \boldsymbol{V}_j B_j^k(\hat{u}_i) - \boldsymbol{P}_i \right\|^2 \\ &= \sum_{i=0}^{P-1} \left\| \sum_{j=0}^{M-1} \boldsymbol{V}_j \hat{B}_j^k(\hat{u}_i) - \boldsymbol{P}_i \right\|^2 \end{aligned} \tag{27}$$

with

$$\hat{B}_j^k(\hat{u}_i) = \begin{cases} B_j^k(\hat{u}_i) + B_{j+M}^k(\hat{u}_i) & \text{for} \quad 0 \le j \le 2 \\ B_j^k(\hat{u}_i) & \text{else} \quad 2 < j \le M-1 \end{cases} \tag{28}$$

which reflects the fact the support of a closed spline curve is cyclic in the control points as opposed to splines with open ends.

Differentiating (27) with respect to the control points yields the following linear system of equations:

$$\left.\begin{aligned} \sum_{j=0}^{M-1} \left\{ \sum_{i=0}^{P-1} \hat{B}_j^k(\hat{u}_i)\hat{B}_l^k(\hat{u}_i) \right\} X_j &- \textstyle\sum_{i=0}^{P-1} x_i \hat{B}_l^k(\hat{u}_i) = 0, \\ \sum_{j=0}^{M-1} \left\{ \sum_{i=0}^{P-1} \hat{B}_j^k(\hat{u}_i)\hat{B}_l^k(\hat{u}_i) \right\} Y_j &- \textstyle\sum_{i=0}^{P-1} y_i \hat{B}_l^k(\hat{u}_i) = 0, \end{aligned}\right\} \text{ for } 0 \le l \le M-1 \tag{29}$$

We can put Equation 29 in concise matrix equations by defining the following vectors and matrices:

$$\left.\begin{aligned} \boldsymbol{X} &= \{X_j\}, \\ \boldsymbol{Y} &= \{Y_j\}, \\ \boldsymbol{C} &= \{\xi_l\} \quad \text{with} \quad \xi_l = \sum_{i=0}^{P-1} x_i \hat{B}_l^k(\hat{u}_i), \\ \boldsymbol{D} &= \{\eta_l\} \quad \text{with} \quad \eta_l = \sum_{i=0}^{P-1} y_i \hat{B}_l^k(\hat{u}_i), \\ S_{jl} &= \sum_{i=0}^{P-1} \hat{B}_j^k(\hat{u}_i)\hat{B}_l^k(\hat{u}_i), \end{aligned}\right\} \quad 0 \le j,l \le M-1 \tag{30}$$

[2] A *correct* distance measure would be to use the normal distance of P_i to the spline curve. This results, however, in a non-linear minimization method which is hard to solve.

and we obtain the following matrix equations:

$$\left.\begin{array}{rcl} A \cdot \boldsymbol{X} - \boldsymbol{C} & = & 0, \\ A \cdot \boldsymbol{Y} - \boldsymbol{D} & = & 0, \end{array}\right\} \tag{31}$$

which we solve using a LU decomposition ([Press *et al.* 86]).

References

[Bar-Shalom 90] Y. Bar-Shalom (ed.), *Multitarget-Multisensor Tracking*, Artech House, Inc, Norwood, MA, 1990.

[Bartels *et al.* 87] R. Bartels, J. Beatty, B. Barsky, *An Introduction to Splines for use in Computer Vision*, Morgan Kaufmann, 1987.

[Bascle & Deriche 93] B. Bascle, R. Deriche, Stereo Matching, Reconstruction and Refinement of 3D Curves using deformable Contours, in *Proc. Int. Conf. on Computer Vision*, Berlin, Germany, May. 11-14, 1993, pp. 421–430.

[Blake *et al.* 93] A. Blake, R. Curwen, A. Zisserman, Affine-invariant contour tracking with automatic control of spatiotemporal scale, in *Proc. Int. Conf. on Computer Vision*, Berlin, Germany, May. 11-14, 1993, pp. 66–75.

[Canny 86] J. Canny, A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-8** (1986) 679–698.

[Chong *et al.* 90] C.Y. Chong, S. Mori, K.C. Chang, *Multitarget-Multisensor Tracking*, Artech House, Inc, Norwood, MA, 1990, chapter 8: Distributed Multitarget Multisensor Tracking, pp. 247–295.

[Cipolla & Blake 90] R. Cipolla, A. Blake, The dynamic analysis of apparent contours, in *Proc. Int. Conf. on Computer Vision*, Osaka, Japan, Dec. 4-7, 1990, pp. 616–623.

[Cipolla & Blake 92] R. Cipolla, A. Blake, Surface Orientation and Time to Contact from Image Divergence and Deformation, in *Proc. Second European Conference on Computer Vision*, S. Margherita, Ligure, Italy, May 18-23, 1992, G. Sandini (ed.), Lecture Notes in Computer Science **588**, Springer-Verlag, Berlin, Heidelberg, New York, 1992, pp. 187–202.

[Cox *et al.* 89] P. Cox, H. Maitre, M. Minoux, C. Ribeiro, Optimal Matching of Convex Polygons, *Pattern Recognition Letters* **9** (1989) 327–334.

[Curwen & Blake 92] R. Curwen, A. Blake, *Active Vision*, MIT Press, Cambridge, MA, 1992, chapter Dynamic Contours: Real-time Active Snakes, pp. 39–57.

[Gelb 74] A. Gelb (ed.), *Applied Optimal Estimation*, The MIT Press, Cambridge, MA and London, UK, 1974.

[Huang *et al.* 93] T. Huang, G. Ogasawara, S. Russell, Symbolic Traffic Scene Analysis Using Dynamic Belief Networks, in *AAAI Workshop on AI in IVHS*, Washington D.C., 1993.

[Irani *et al.* 92] M. Irani, B. Rousso, S. Peleg, Detecting and Tracking Multiple Moving Objects Using Temporal Integration, in *Proc. Second European Conference on Computer Vision*, S. Margherita, Ligure, Italy, May 18-23, 1992, G. Sandini (ed.), Lecture Notes in Computer Science **588**, Springer-Verlag, Berlin, Heidelberg, New York, 1992, pp. 282–287.

[Karmann & von Brandt 90] Klaus-Peter Karmann, Achim von Brandt, Moving Object Recognition Using an Adaptive Background Memory, in V Cappellini (ed.), *Time-Varying Image Processing and Moving Object Recognition, 2*, Elsevier, Amsterdam, The Netherlands, 1990.

[Kass *et al.* 88] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active Contour Models, *International Journal of Computer Vision* **1** (1988) 321–331.

[Kilger 92] Michael Kilger, A Shadow Handler in a Video-based Real-time Traffic Monitoring System, in *IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, 1992, pp. 1060–1066.

[Koenderink 86] J.J. Koenderink, Optic flow, *Visual Research* **26** (1986) 161–180.

[Koller *et al.* 93] D. Koller, K. Daniilidis, H.-H. Nagel, Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes, *International Journal of Computer Vision* **10**:3 (1993) 257–281.

[Kurien 90] T. Kurien, *Multitarget-Multisensor Tracking*, Artech House, Inc, Norwood, MA, 1990, chapter 3: Issues in the Design of Practical Multitarget Tracking Algorithms, pp. 43–83.

[Létang *et al.* 93] J.M. Létang, V. Rebuffel, P. Bouthemy, Motion detection robust to pertubations: a statistical regularization and temporal integration framework, in *Proc. Int. Conf. on Computer Vision*, Berlin, Germany, May. 11-14, 1993, pp. 21–30.

[Marslin *et al.* 91] R.F. Marslin, G.D. Sullivan, K.D. Baker, Kalman Filters in Constrained Model-Based Tracking, in *Proc. British Machine Vision Conference*, Glasgow, UK, Sept. 24-26, 1991, pp. 371–374.

[Meyer & Bouthemy 92] F. Meyer, P. Bouthemy, Region-based tracking in an image sequence, in *Proc. Second European Conference on Computer Vision*, S. Margherita, Ligure, Italy, May 18-23, 1992, G. Sandini (ed.), Lecture Notes in Computer Science **588**, Springer-Verlag, Berlin, Heidelberg, New York, 1992, pp. 476–484.

[Meyer & Bouthemy 93] F. Meyer, P. Bouthemy, Exploiting the Temporal Coherence of Motion for Linking Partial Spatiotemporal Trajectories, in *IEEE Conf. Computer Vision and Pattern Recognition*, New York City, NY, June 15-17, 1993, pp. 746–747.

[Murray *et al.* 93] D.W. Murray, P.F. McLauchlan, I.D. Reid, P.M. Sharkey, Reactions to Peripheral Image Motion using a Head/Eye Platform, in *Proc. Int. Conf. on Computer Vision*, Berlin, Germany, May. 11-14, 1993, pp. 403–411.

[Press *et al.* 86] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986.

[Rao 92] B.S.Y. Rao, *Active Vision*, MIT Press, Cambridge, MA, 1992, chapter Data Association Methods for Tracking Systems, pp. 91–105.

[Rao *et al.* 93] B.S.Y. Rao, H.F. Durrant-Whyte, J.A. Sheen, A Fully Decentralized Multi-Sensor System for Tracking and Surveillance, *Intern. Journal of Robotics Research* **12**:1 (1993) 20–38.

[Terzopoulos & Szeliski 92] D. Terzopoulos, R. Szeliski, *Active Vision*, MIT Press, Cambridge, MA, 1992, chapter Tracking with Kalman Snakes, pp. 3–20.

[Worrall *et al.* 91] A.D. Worrall, R.F. Marslin, G.D. Sullivan, K.D. Baker, Model-Based Tracking, in *Proc. British Machine Vision Conference*, Glasgow, UK, Sept. 24-26, 1991, pp. 310–318.

[Zheng & Chellappa 93] Q. Zheng, R. Chellappa, Automatic Feature Point Extraction and Tracking in Image Sequences for Unknown Camera Motion, in *Proc. Int. Conf. on Computer Vision*, Berlin, Germany, May. 11-14, 1993, pp. 335–339.