

# Learning based system for detection and tracking of vehicles

Hakan Ardo<sup>1</sup>

Center for Mathematical Sciences, Lund University, Sweden,  
ardo@maths.lth.se

**Abstract.** In this paper we study how learning can be used in several aspects of car detection and tracking. The overall goal is to develop a system that learns its surrounding and subsequently does a good job in detecting and tracking all cars (and later pedestrians and bicycles) in an intersection. Such data can then be analyzed in order to determine how safe an intersection is. The system is designed to, with minimal supervision, learn the location of the roads, the geometry needed for rectification, the size of the vehicles and the tracks used to pass the intersection. Several steps in the tracking process are described. The system is verified with experimental data, with promising results.

## 1 Introduction

There is an increased need for automatic analysis of car and pedestrian traffic. Methods for detecting and tracking cars in images and image sequences has been developed for quite some time now. Most of the existing methods, do however concentrate on car detection using appearance models [11], [4].

This project is initiated by the needs of governmental organizations such as 'Väg och trafik Institutet' but also department of public works in cities. One of the prime motives here is the possibilities to assess traffic safety. Often safety is neglected when constructing new roads and intersections. If safety is considered there are very few tools to actually measure and monitor traffic safety. One possibility is to study the number of accidents reported to the police from a certain intersection. Recent research [6] has, however, shown that it is possible to predict how many such accidents there is by manually observing certain events during a shorter time interval, e.g. 2-3 days. These traffic studies are however very time-consuming and requires that trained personnel study the traffic.

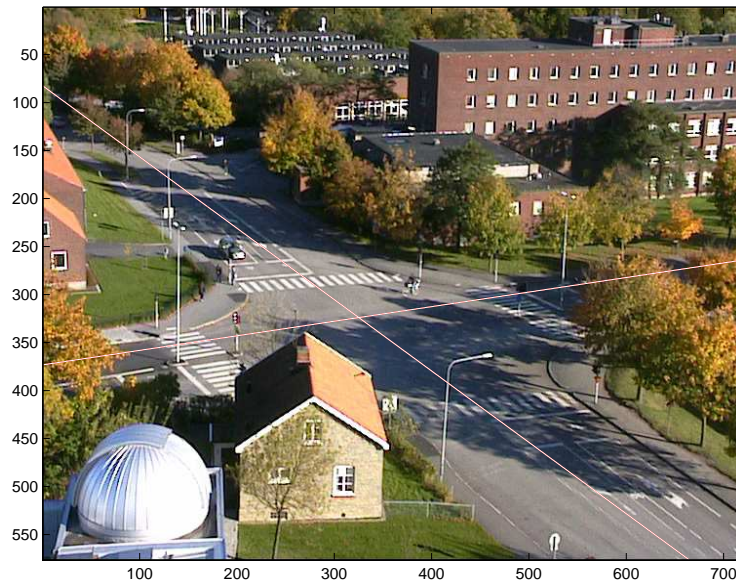
The goal of this study is to use learning to increase the robustness of a tracking system. Learning is used, to estimate models of scene background, to automatically rectify the image and to locate the lanes in the images. The problem of counting the number of cars driving through a intersection, such as the one shown in Figure 1 will be addressed as well as for each car deciding which entry and exit lane is used.

Several methods for estimating and updating the background images exist. One such method by Stauffer and Grimsson[10] is based on modeling the probability distribution of each background pixel as a mixture of Gaussians using the EM algorithm. The dominant component is considered to be the background and is used for estimating fore-

ground pixels. A more recent paper [5] extends this to include pan-tilt rotations of the camera.

Methods for detecting the common object trajectories in a general scene under surveillance have been developed by for example Makris and Ellis in [8] and, Johnson and Hogg in [7]. In both articles it's mentioned that this kind of models can be used to predict the future position of the tracked objects, which could be very useful in the prediction step of a tracking algorithm. But no tracking algorithm exploiting this learnt information are presented or tested. In this paper a tracker developed that uses learnt models for both prediction and detection of cars is implemented and it's results is compared to a tracking operating on the same input data but using no prior information. Also, it is shown that the models can be used to reduce a 2D tracking problem into several 1D tracking problems. The road model presented in this paper explicitly models the right-hand-side driving of cars in the road and are thus able to detect the roads in the scene even if there are a lot of pedestrians walking across the road in strange directions. This would probably not be the case of the more general purpose algorithms [8] and [7].

Ground plane calibration have been done by Stauffer and Grimson in [3] and by Renno, Orwell and Jones in [9]. In both cases the a ground plane were estimated from the height of tracked objects, which assumes that the objects used for the calibration is high and thin, e.g. people. In this paper the area is used instead which allows lower and wider objects to be used for the calibration such as cars.



**Fig. 1.** Video sequences used for the experiments, with the detected roads plotted.

## 2 Methods

The idea is to start out with a simple tracker based on the background/foreground-segmentation described by Stauffer-Grimsson [10]. From this binary segmentation image, connected segments are extracted and considered objects and then segments overlapping between adjacent frames are connected into tracks. If there are several overlapping segments only the largest segments are considered. The resulting tracks are not very good. Typically a relatively high proportion of them are broken into several tracks and some are erroneously merged. However, the results are good enough to allow the system to automatically locate the roads (Section 2.1), rectify the image (Section 2.2) and estimate the mean size of the objects.

In Section 2.2 a novel system for automatic rectification is presented. The idea here is that all cars are approximately the same size, which can be estimated. Then the probability of finding an object of this size at a certain position in the rectified images is estimated in Section 2.3. This improves the tracking as a lot of noise can be removed by assuming a minimum size of the tracked objects. In Section 2.4 the tracks are organized into classes depending on how they drive through the intersection. For each of those classes a bspline-curve is fitted to the tracks describing the mean motion. The final tracker then detects objects of the estimated size following one of those bsplines.

The calibration steps of finding the roads, the rectification and the splines are typically performed off-line. Once that is done the main processing time of the 1D tracker is done by the background/foreground segmentation followed by the viterbi-optimization described in Section 2.4. For both those algorithms there exists fast implementations that should allow real time performance.

### 2.1 Automatic road detection

The first step in the analyze is to determine where in the image the roads are located. The idea here is to model a lane as a straight line of a certain width containing objects traveling in one direction along the line. Then a road can be modelled as two parallel lanes with opposite traveling directions. The problem is to find a probability distribution function,  $P_{road}(o|\theta)$ , that, given a set of parameters describing the road,  $\theta$ , generates the probability that an object,  $o$ , is located on the road. Here an object is described by its position  $(x, y)$  and it's traveling direction  $(\delta_x, \delta_y)$ , a unit length vector. That is  $o = (x, y, \delta_x, \delta_y)$ .

*Lane modelling* The model originates from a straight line  $ax + by + c = 0$ , representing the center of the lane. Assume that the distance between a car in the lane and this line is Gaussian distributed with a mean value zero and a variance depending on the width of the lane. The distance from an object at  $(x, y)$  to the line is

$$t = \frac{ax + by + c}{\sqrt{a^2 + b^2}}, \quad (1)$$

which is inserted into the one-dimensional Gaussian probability distribution function. The result is divided by  $L$ , the length of the line, to make sure the the pdf still integrates

to one. The resulting distribution is

$$P(x, y|a, b, c, \sigma, L) = \frac{1}{L\sigma\sqrt{2\pi}} e^{-\frac{(ax+by+c)^2}{(a^2+b^2)2\sigma^2}}. \quad (2)$$

The parameters  $a, b, c$  can be rescaled without changing the line they represent. This degree of freedom is used to represent the variance,  $\sigma$ . A particularly simple form is achieved by setting  $\sigma = 1/\sqrt{2(a^2+b^2)}$ , then

$$P(x, y|a, b, c, L) = \frac{\sqrt{a^2+b^2}}{L\sqrt{\pi}} e^{-(ax+by+c)^2}. \quad (3)$$

The traveling direction of each object is defined as a unit length vector  $(\delta_x, \delta_y)$  indicating in which direction it is moving. For a given line there are two possible traveling directions, which are found by rotating the normal  $\pm\frac{\pi}{2}$ . By defining the traveling direction as the normal rotated  $\frac{\pi}{2}$  the sign of the normal will decide which the model represents. Assuming that the traveling direction of the cars also is Gaussian distributed with this mean direction, some variance  $\sigma_x, \sigma_y$  and that it is independent of  $(x, y)$  gives

$$P_{lane}(o|\theta) = \frac{\sqrt{a^2+b^2}}{2L\pi^{3/2}\sigma_x\sigma_y} e^{-(ax+by+c)^2} \cdot e^{-\frac{\left(\delta_x + \frac{b}{\sqrt{a^2+b^2}}\right)^2}{2\sigma_x^2} - \frac{\left(\delta_y - \frac{a}{\sqrt{a^2+b^2}}\right)^2}{2\sigma_y^2}}, \quad (4)$$

where

$$o = (x, y, \delta_x, \delta_y, L) \quad (5)$$

$$\theta = (a, b, c, \sigma_x, \sigma_y) \quad (6)$$

*Road modelling* A road consists of two lanes of opposite traveling directions. Let the line defined by  $ax + by + c = 0$  represent the center of the road, and use the sign function to indicate which side of the line a point is located. If right hand side driving is assumed the normal rotated  $\frac{\pi}{2}$  is the traveling direction on the positive side and  $-\frac{\pi}{2}$  on the negative side, which gives

$$P_{road}(o|\theta) = \frac{\sqrt{a^2+b^2}}{2L\pi^{3/2}\sigma_x\sigma_y} e^{-(ax+by+c)^2} \cdot e^{-\frac{\left(\delta_x - \frac{\text{sign}(ax+by+c)b}{\sqrt{a^2+b^2}}\right)^2}{2\sigma_x^2} - \frac{\left(\delta_y + \frac{\text{sign}(ax+by+c)a}{\sqrt{a^2+b^2}}\right)^2}{2\sigma_y^2}}. \quad (7)$$

*Parameter estimation* The task of finding the two roads of the image is now reduced to that of estimating the parameters  $\theta_1$  and  $\theta_2$  as defined by equation 6 and 5. For this the EM-algorithm [1] is used to maximize (7) for the data produced by the initial tracker. It requires an estimate of  $\theta$  from a set of measured data-points. For that an algorithm similar to RANSAC [2] is used, where a set of candidate parameters is generated by iterating the steps below. Then the candidate with the highest probability is chosen.

- (i) Chose two points  $p_1$  and  $p_2$  in the dataset at random.
- (ii) Let  $(a, b, c)$  be the line through  $p_1$  and  $p_2$ .
- (iii) Rotate all data-points to make the above line parallel to the x-axis.

- (iv) Estimate the standard deviation,  $\sigma$  along the x-axis.
- (v) Encode  $\sigma$  into  $a$ ,  $b$  and  $c$  by scaling them by  $\frac{1}{\sigma\sqrt{2(a^2+b^2)}}$ .
- (vi) Estimate  $\sigma_x$  and  $\sigma_y$

Setting  $L$  to the length of the line visible in the image will give short roads a higher probability. To prevent this  $L$  is chosen to be a constant equal to the diagonal of the image.

The EM-algorithm is initialized with an initial guess of two roads one parallel to the x-axis and one parallel to the y-axis. The algorithm is then executed for a few iterations to get a better starting point before an uniform background distribution is introduced. It absorbs much of noise outside the lanes.

## 2.2 Automatic rectification

Once the roads have been located all tracks not located within the roads can be removed the rest can be used to rectify the image. The assumption here is that most objects traveling on the roads are of the same size and that the roads can be approximated with a plane. This will be used to find a rectification of the image that will give all objects the same area in the image. The estimated road models from the previous sections are used to filter out tracks not originating from cars.

To rectify the image a 3x3 matrix,  $H$ , has to be found, that transforms the coordinates of the original image  $(x, y)$  into the coordinates of a rectified image  $(\hat{x}, \hat{y})$  according to

$$\lambda \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \quad (8)$$

The area of objects in the original image,  $A$ , will in the rectified image be scaled by the functional determinant into

$$\hat{A} = \left| \begin{array}{cc} \frac{\partial \hat{x}}{\partial x} & \frac{\partial \hat{x}}{\partial y} \\ \frac{\partial \hat{y}}{\partial x} & \frac{\partial \hat{y}}{\partial y} \end{array} \right| A = \frac{\det H}{(h_{31}x + h_{32}y + h_{33})^3} A. \quad (9)$$

For each position of each tracked object there is one data-point consisting of a position  $x_k, y_k$  and a corresponding area  $A_k$ . All areas in the rectified image,  $\hat{A}$ , are assumed to be 1 (equal), and  $\det H$  can be ignored as it is only a constant, rescaling all areas equally. This turns (9) into the set of equations

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \dots & \dots & \dots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \begin{pmatrix} A_1^{1/3} \\ A_2^{1/3} \\ \dots \\ A_n^{1/3} \end{pmatrix} \quad (10)$$

with three unknowns,  $h_{31}$ ,  $h_{32}$  and  $h_{33}$ .

Solving this set of equations gives the last row in  $H$ . If the camera is calibrated the rectification is a pure rotation and the last two rows can be chosen such that  $H$  becomes a rotation matrix. This still leaves one degree of freedom which corresponds to an in-plane rotation of the rectified image and can thus be chosen arbitrarily.

### 2.3 Car indicator

From the rectification algorithm it is also straight forward to extract the area of the cars. Using this area estimate it is then possible to construct a car indicator. The probability of finding a car at position  $L_c = (x_c, y_c)$  in frame  $f$ ,  $P_c(L_c, f)$ , can be calculated from the binary background/foreground segmentation,  $S_f(x, y)$ . This image is one for every pixel considered foreground and zero for every pixel considered background. Assuming that the pixel has the correct classification with a probability of  $p_{fg}$  and that all cars are square with a radius of  $r$ ,  $P_c$  can be found as

$$P_c(L_c, f) = \prod_{\substack{|x - x_c| < r \\ |y - y_c| < r \\ S_f(x, y) = 1}} p_{fg} \prod_{\substack{|x - x_c| < r \\ |y - y_c| < r \\ S_f(x, y) = 0}} (1 - p_{fg}) \quad (11)$$

### 2.4 1D Tracking

The final step is to learn the typical tracks a car can use to legally cross the intersection and then use these tracks to extract full car tracks from the  $P_c(L_c, f)$  images. This is done by fitting a bspline-curve,

$$(x, y) = B_k(t), \quad 0 \leq t \leq 1, \quad k = 1..N. \quad (12)$$

to each of the  $N$  typical tracks. In a medium sized intersection  $N = 16$  as there is 4 roads entering the intersection and 4 leaving it (12 if u-turns are not considered).

For each  $B_k$   $P_c$  is restricted to  $M$  points along the curve. This generates  $k$  new images,  $I_k$ , with one column per frame  $f$ ,

$$I_k(f, t) = P_c(B_k(\frac{t}{M}), f), \quad (13)$$

where  $k = 1..N$  and  $t = 0..M$ , cf. Figure 5.

Every car driving, legally, through the intersection will drive along one of the bspline-curves and always in the same direction. The parameterization,  $t$ , can thus be chosen so that a car always enters at  $t = 0$  and exits at  $t = 1$ . Considering each of the  $I_k$  images separately and assuming that cars are driving continuously, along one of the bspline curves, a car consists of a sequence of points

$$C = \{(f_i, t_i)\}_{i=1}^L, \quad (14)$$

where  $f_i \leq f_{i+1} \leq f_i + 1$  and  $t_i \leq t_{i+1} \leq t_i + 1$ .

If the motion is also assumed to be continuous this means that one of

$$\begin{cases} f_i = f_{i-1} + 1 \\ t_i = t_{i-1} \end{cases}, \begin{cases} f_i = f_{i-1} \\ t_i = t_{i-1} + 1 \end{cases} \text{ or } \begin{cases} f_i = f_{i-1} + 1 \\ t_i = t_{i-1} + 1 \end{cases} \quad (15)$$

will always be true.

The different curves,  $B_k$ , overlap. Thus a car belonging to class  $k$  has to drive along the entire curve  $B_k$ . That is  $t_1 = 0$  and  $t_L = M$ . For such a sequence, the probability of it being a car is

$$p(C) = \prod_{i=1}^L I_k(f_i, t_i). \quad (16)$$

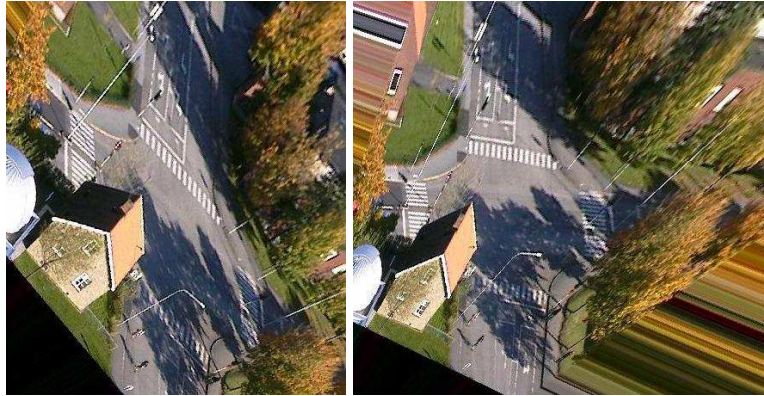
Finally, assume that all cars driving through the intersection generate local maximas to (16) above some small threshold,  $P_{nc}$ . Then for each potential entry point (local maxima to  $(f, 0)$ ) and exit point (local maxima to  $(f, M)$ ) a local maxima to (16) can be found by viterbi-optimization. All those sequences form a finite set of candidate cars  $S_{cc} = \{C_j\}$ ,  $j = 1..N_{cc}$ , and the set of cars driving through the intersection,  $S_c$ , can be found as the non-overlapping subset of  $S_{cc}$  maximizing  $\sum_{C_j \in S_c} p(C_j)$ , where  $S_c$  non-overlapping means that

$$\begin{cases} C_i, C_j \in S_c \\ (f, t) \in C_i \end{cases} \Rightarrow (f, t) \notin C_j. \quad (17)$$

As  $I_k$  is a sampled, discrete image two close local maximas might be merged to one. So accepting overlapping points in  $S_c$  for a few  $t$ -values, makes the system more robust.

### 3 Experimental results

The initial tracker simply combining overlapping connected segments were tested on a 5 min sequence (see Figure 1) containing 62 vehicles (1 tractor, 1 bus and 60 cars) and 761 tracks were found. This includes vehicles, bicycles, pedestrians and pure noise. Many tracks are slitted into several. No objects are entirely missed though.



**Fig. 2.** Output from the automatic rectification. To the left the algorithm worked on all tracks available. To the right some good tracks were manually chosen.



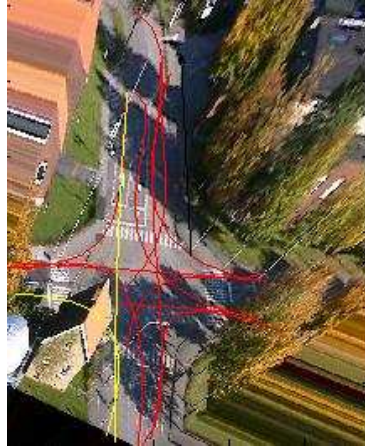
**Fig. 3.** The two roads in the image located approximately. Each road is represented by two lanes of opposite driving direction. The solid line shows the center of the road separating the two lanes and the two dashed lines shows the border of the road.

The road detection algorithm were then successfully applied, as shown in Figure 1, followed by the rectification algorithm resulting in the left image in Figure 2. The road is clearly tilted to the right. This is due to the simple linear optimization method used and the fact that there is quiet a lot of outliers and noise. By manually selecting a set of good tracks the right image is achieved, which is the wanted result. This means that by using non-linear optimization or better measurements the algorithm will probably perform just fine.

The road detection algorithm have been tested on tracks originated from both the original image and from the rectified, with similar results shown in Figure 1. The horizontal road is detected above the road centrum line in the image, which is correct as the height of the cars will place the centrum point of the tracked segments slightly above there position on the road. The result of the later is shown i Figure 3.

The result of detecting the typical tracks used to traverse the crossing is shown in Figure 4. 10 of all 12 tracks were found. The two missed are marked yellow in the Figure. The problem with the last two is that the initial tracker produced no tracks belonging to these classes, which is because there is very few cars using them.

Finally the  $I_k$  images were produced, see Figure 5, and the 1D tracker were executed. In total 66 (of totally 62) tracks were found including 3 bikes, with significantly lower  $p(C)$ , and 1 ghost car originating from a combination of cars, which could be removed by removing overlapping cars from the final set of cars from all classes. Also the buss and the tractor were detected as 2 cars each, as they are significantly larger



**Fig. 4.** All tracks used to pass the intersection clustered into 12 clusters based on their entry and exit lane. Each cluster is represented with a spline (shown in the figure) fitted to all the tracks in the cluster. The red splines were found by the proposed algorithm and the two yellow ones had to be added by hand.

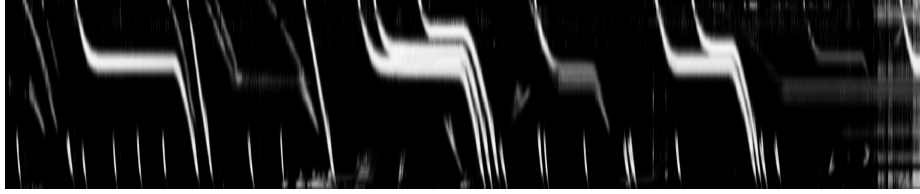
than a car. Also, two cars were missed, one occluded by the building. All other cars were detected correctly.

## 4 Conclusions

In this paper we have introduced methods for automatic rectification, road, lane and car detection based on learning. The results on real data are promising but additional work is needed.

Experiments have indicated that the rectification algorithm seems robust to errors in the estimation of the focal-length. This means that rectifying an uncalibrated camera should also be possible by for example assuming that the aspect ratio of a car is 1x2 and let the traveling direction indicate the longer side. Also, an optimization method less sensitive for outliers are needed, or cleaner input data which could be achieved by using a better initial tracker, or by implementing some feedback loop passing the data produced by the final tracker back to the rectification to allow it to tune the rectification parameters. It should also be quite possible to remove the assumption that all cars have the same size, and instead assume that any single object traveling through the intersection does not change it's size. Then all the tracks could be clustered not only on their entry and exit lanes, but also on their area and different 1D-trackers could be used for vehicles of different sizes.

One single car will produce data in several of the  $I_k$  images and thus optimizing over all of them at once instead of one by one should yield better results. In that case a more advanced model of how cars are allowed to move among the sample points is required, such as for example a Markov model. In that case a uniformly distributed set



**Fig. 5.** The probability that there is a car located at some position (y-axis) along one of the splines shown in Figure 4 over time (x-axis). The straight lines are cars passing through the intersection without stopping. The lines with a long horizontal part in the middle are cars that stop and wait for the red light. This image is denoted  $I_k(f, t)$  where  $t$  is the position along spline  $k$  in frame  $f$ .

of sample points might be enough and there will be no need to estimate the bspline curves as the Markov model will represent the relations between the sample points.

## References

1. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38, 1977.
2. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications-of-the-ACM*, 24(6):381–95, 1981.
3. W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 22. IEEE Computer Society, 1998.
4. Michael Haag and Hans-Hellmut Nagel. Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):295–319, 1999.
5. E. Hayman and J-O. Eklundh. Background subtraction for a mobile observer. In *Proc. 9th Int. Conf. on Computer Vision, Nice, France, 2003*.
6. C. Hydén. *The development of a method for traffic safety evaluation: The Swedish traffic conflicts technique*. PhD thesis, Institutionen för trafikteknik, LTH, Lund, 1987.
7. Neil Johnson and David Hogg. Learning the distribution of object trajectories for event recognition. In *BMVC '95: Proceedings of the 6th British conference on Machine vision (Vol. 2)*, pages 583–592. BMVA Press, 1995.
8. Dimitrios Makris and Tim Ellis. Path detection in video surveillance. *Image Vision Comput.*, 20(12):895–903, 2002.
9. J. R. Renno, James Orwell, and Graeme A. Jones. Learning surveillance tracking models for the self-calibrated ground plane. In *BMVC*, 2002.
10. Chris Stauffer. Adaptive background mixture models for real-time tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 246–252, 1999.
11. Tao Zhao and Ram Nevatia. Car detection in low resolution aerial image. In *Proc. ICCV 2001*, pages 710–717, 2001.