

# Use of learning for detection and tracking of vehicles

Håkan Ardo

Centre 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 analysed in order to determine how safe an intersection is. 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, [5].

This project is initiated by the needs of governmental organizations 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 policereported accidents from a certain intersection. Recent research 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 personell 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 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[4] is based on modeling the probability distribution of each background pixel as a mixture of Gaussians using the EM algorithm. A more recent paper [3] extends this to include pan-tilt rotations of the camera.

## 2 Methods

The idea is to start out with a simple tracker based on the background/foreground-segmentation described by Stauffer-Grimsson [4]. From this binary segmentation image, connected segments are extracted and considered as 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. 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 organised 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.

## 2.1 Automatic road detection

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 its 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 centre 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 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 inde-

pendent 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  $o = (x, y, \delta_x, \delta_y)$  and  $\theta = (a, b, c, L, \sigma_x, \sigma_y)$ .

**Road modelling** A road consists of two lanes of opposite traveling directions. Let the line defined by  $ax + by + c = 0$  represent the centre 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

$$P_{road}(o|\theta) = \frac{\sqrt{a^2 + b^2}}{2L\pi^{3/2}\sigma_x\sigma_y} e^{-(ax+by+c)^2} \cdot e^{-\frac{\left(\delta_x - \frac{\text{sign}(ax+by+c)b}{\sqrt{a^2+b^2}}\right)^2}{2\sigma_x^2} - \frac{\left(\delta_y + \frac{\text{sign}(ax+by+c)a}{\sqrt{a^2+b^2}}\right)^2}{2\sigma_y^2}}. \quad (5)$$

**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 above. For this the EM-algorithm [1] is used to maximize (5) 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  $p_2$  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  $\sigma$  along the x-axis.
- (v) Encode  $\sigma$  into  $a, b$  and  $c$  by scaling them with  $\frac{1}{\sigma\sqrt{2(a^2+b^2)}}$ .
- (vii) Estimate  $\sigma_x$  and  $\sigma_y$

## 2.2 Automatic rectification

To rectify the image a  $3 \times 3$  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

$$\lambda \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \quad (6)$$

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

$$\hat{A} = \left| \begin{array}{cc} \frac{\partial \hat{x}}{\partial x} & \frac{\partial \hat{x}}{\partial y} \\ \frac{\partial \hat{y}}{\partial x} & \frac{\partial \hat{y}}{\partial y} \end{array} \right| A = \frac{\det H}{(h_{31}x + h_{32}y + h_{33})^3} A. \quad (7)$$

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 (7) into the set of equations

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \dots & \dots & \dots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \begin{pmatrix} A_1^{1/3} \\ A_2^{1/3} \\ \dots \\ A_n^{1/3} \end{pmatrix} \quad (8)$$

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 arbitrary.

### 2.3 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,  $F_g(x, y)$ . Methods for that are being worked on, but not used in this paper. Instead  $P_c(L_c, f)$  is approximated with the convolution of  $F_g(x, y)$  and a gaussian kernel of the same size as the estimated car area.

### 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,

$$(x, y) = B_k(t), \quad 0 \leq t \leq 1, \quad k = 1..N. \quad (9)$$

to each of the  $N$  typical tracks.

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  $f$ ,

$$I_k(f, t) = P_c(B_k(\frac{t}{M}), f), \quad (10)$$

where  $k = 1..N$  and  $t = 0..M$ , cf. Figure 4.

Every car driving, legally, through the intersection will drive along one of the bspline-curves and always in the same direction. The parametrisation,  $t$ , 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, a car consists of a sequence of points

$$C = \{(f_i, t_i)\}_{i=1}^L, \quad (11)$$

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

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

$$p(C) = \prod_{i=1}^L I_k(f_i, t_i). \quad (12)$$

Finally, assume that all cars driving through the intersection generate local maximas to (12) 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 (12) 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

$$\begin{cases} C_i, C_j \in S_c \\ (f, t) \in C_i \end{cases} \Rightarrow (f, t) \notin C_j. \quad (13)$$

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



Figure 1: Video sequences used for the experiments, with the detected roads plotted.

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 splitted into several. No objects are entirely missed though.

The road detection algorithm were then successfully applied, as shown in Figure 1, followed by the rectification algorithm resulting in Figure 2. The linear optimization method used are sensitive to outliers, but if they are manually removed good results are achieved.



Figure 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.

The road detection algorithm have been tested on tracks originated from both the original image and from the rectified, which similar results shown in Figure 1. The horizontal road is detected above the road centrumline in the image, which is correct as the height of the cars will place the centumpoint of the tracked segments slightly above there position on the road. The result of detecting the typical tracks used to traverse the crossing is shown in Figure 3. All but the two yellow tracks were found automatically. The last two had to be added manually as the tracker produced no tracks belonging to these classes.

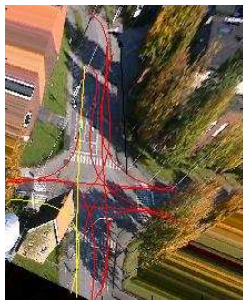


Figure 3: All typical tracks used to pass the intersection.

Finally the  $I_k$  images were produced, see Figure 4, and the 1D tracker were executed. In total

66 tracks were found including 3 bikes, with significantly lower  $p(C)$ , and 1 ghost car originating from a combination of cars. Also the buss and the tracktor were detected as 2 cars each and two cars were missed, one occluded by the building. All other cars were detected correctly.

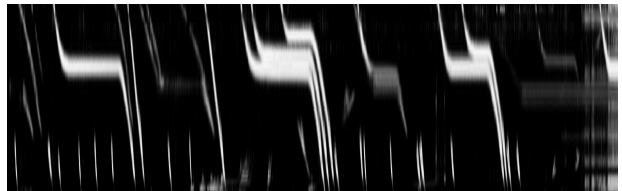


Figure 4: The  $I_k(f, t)$  image showing the probability of finding a car at position  $t$  (y-axis) of bspline  $k$  in frame  $f$  (x-axis) for one  $k$ .

## 4 Conclusions

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

An optimization method less sensitive for outliers are needed for the rectification algorithm, and rectifying uncalibrated cameras should be possible by assuming some aspect ratio of the cars. Also locating tracks in all  $I_k$  images simultaneously using some marcovmodell describing the car motion instead of the splines would be interesting.

## 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] E. Hayman and J.-O. Eklundh. Background subtraction for a mobile observer. In *Proc. 9th Int. Conf. on Computer Vision, Nice, France*, 2003.
- [4] C. Stauffer. Adaptive background mixture models for real-time tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 246–252, 1999.
- [5] T. Zhao and R. Nevatia. Car detection in low resolution aerial image. pages 710–717, 2001.