Combining shadow detection and simulation for estimation of vehicle size and position

Björn Johansson, Johan Wiklund, Per-Erik Forssén and Gösta Granlund

Original Publication:

Björn Johansson, Johan Wiklund, Per-Erik Forssén and Gösta Granlund, Combining shadow detection and simulation for estimation of vehicle size and position, 2009, PATTERN RECOGNITION LETTERS, (30), 8, 751-759.
http://dx.doi.org/10.1016/j.patrec.2009.03.005

Combining shadow detection and simulation for estimation of vehicle size and position

Björn Johansson * , Johan Wiklund, Per-Erik Forssén, Gösta Granlund
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, SE-583 32 Linköping, SWEDEN

Abstract

This paper presents a method that combines shadow detection and a 3D box model including shadow simulation, for estimation of size and position of vehicles. We define a similarity measure between a simulated image of a 3D box, including the box shadow, and a captured image that is classified into background/foreground/shadow. The similarity measure is used in an optimization procedure to find the optimal box state. It is shown in a number of experiments and examples how the combination shadow detection/simulation improves the estimation compared to just using detection or simulation, especially when the shadow detection or the simulation is inaccurate.

We also describe a tracking system that utilizes the estimated 3D boxes, including highlight detection, spatial window instead of a time based window for predicting heading, and refined box size estimates by weighting accumulated estimates depending on view. Finally we show example results.

Key words: Vehicle tracking, 3D box model, object size estimation, shadow detection, shadow simulation

1. Introduction

Intersection accidents are over-represented in the statistics of traffic accidents. This applies to everything from fatal accidents to mere property damage. This problem has received greater attention by the Swedish Road Administration (SRA) in connection with its work on *Vision Zero* [1] . As part of the IVSS (Intelligent Vehicle Safety Systems) project 'Intersection accidents: analysis and prevention', video data is collected at intersections. The video data and other data (e.g. data collected using a test vehicle, and data collected from a driving simulator) will be used to test research hypotheses on driver behavior and traffic environment factors, as well as a basis for proposing ways to improve road safety at intersections.

A vehicle tracking system intended for processing of a large amount of data will have to deal with many different types of weather conditions and traffic situations. This is a difficult task and is still not sufficiently solved, hence the vast amount of continuing publications in this field. Nagel and co-workers (Nagel, 2004) have for example written an interesting exposé over the progress of object tracking during 30 years, with frame differentiation, feature based optical flow, dense optical flow, polyhedral models, and inclusion of more a priori knowledge of e.g. lane locations.

One common approach in tracking systems is to first classify the image pixels into background and foreground regions, as in the Stauffer-Grimson statistical background subtraction method (Stauffer and Grimson, 1999). One problem in this approach is shadows, which are often included in the foreground and cause errors in segmentation and tracking. The conditions in Sweden, and other northern countries,

* Corresponding author. Fax: +46 13 13 85 26
Email address: bjorn@isy.liu.se (Björn Johansson).

[1] The policy on reduction of traffic accidents

may be especially difficult in that respect, since the sun elevation angle in the middle of Sweden is at best about 55°, thus casting long shadows even in summertime. There exist a number of methods for detection of shadows, (Prati et al., 2003) evaluates a few methods. One of the optimal methods use the color model introduced in (Horprasert et al., 1999), which in (Wood, 2007) is combined with the Stauffer-Grimson background subtraction method to get a statistical background/foreground/shadow classification. This is the method used in this paper, and we also describe a generalization for dealing with highlights as well (section 2). It is worth noticing from the results in (Prati et al., 2003) that the best methods have about 80% classification accuracy on the difficult sequences (and thus even less in particular cases), which is not extremely high.

In a recent approach, (Joshi et al., 2007) use a method similar to ours to detect shadows, but they also added edge cues to improve the shadow detection. Another recent similar method is (Carmona et al., 2008) that classifies pixels into even more categories such as reflections (similar to our highlights), ghosts, and fluctuations. They use truncated cone regions instead of our cylinder regions to model the different class regions in the color space. It is also possible to use more intensity invariant features in the Stauffer-Grimson method, as in e.g. (Ardö and Berthilsson, 2006). It is likely that many of these new methods can improve the classification if integrated into our framework.

Still, method parameters are often tuned to certain sequences and light conditions, and it is difficult to have perfect detection under general and varying conditions (the many publications in this field also indicate this challenge). Another problem is that light from a vehicle reflects onto the ground and changes the color on the ground depending on the color of the vehicle, this will affect the shadow classification if one vehicle reflects light into another vehicles shadow. Our assumption is that the foreground/shadow classification cannot be made perfect in general conditions using automatic parameter tuning.

An alternative to detecting and removing shadows is to simulate them. (Dahlkamp et al., 2004) showed how a shadow added to their polyhedral model could improve the accuracy when fitting the model to edge features. (Song and Nevatia, 2007) give examples of how simulating shadows on a 3D box (with fixed size) can improve detection. The detection is still useful when the simulation is insufficient, e.g. when the vehicle model is inaccurate, or when other indirect shadows appear. We will in this paper combine both shadow detection and shadow simulation, and show that the combination improves performance by reducing clutter while still allowing detection of small objects. The results are also improved whenever either the shadow classification or the simulation are incorrect, e.g. when the shadow is absent while being simulated, or when a dark vehicle is misclassified as shadow.

We use 3D boxes as vehicle appearance model, as we have done before to aid the segmentation (Johansson et al., 2006), but here also to simulate shadows. As said in (Dahlkamp et al., 2004, 2007), switching from 2D tracking in the image plane to 3D tracking in the scene domain often results in a substantially reduced failure rate, because more prior knowledge about the objects can be utilized. For example, foreground regions are not always homogeneous, and we need some criteria to merge regions that are likely to belong to the same object. It can be difficult to choose an image based scale threshold for merging regions, especially if the objects vary in size e.g. due to varying distance. The use of 3D boxes is a way to choose this scale threshold based on geometric information.

Many other models of different complexities have been proposed, ranging from a few 3D line features (Kim and Malik, 2003), rectangular 2D boxes (Magee, 2004), and more explicit models like polyhedral models (Nagel, 2004; Dahlkamp et al., 2004, 2007; Lou et al., 2005). In our system the 3D box appears to be a sufficiently complex model, and it is simple enough to cover most types of vehicles. It is difficult to have polyhedral models for all sorts of vehicles, especially in industrial areas where many uncommon types of vehicles occur (tractors, trucks, vehicles with different types of trailers, etc.). Furthermore, full adaptation of the polyhedral model to image data can be unstable (Böckert, 2002), or non-robust (Nagel, 2004).

The paper is organized as follows: Section 2 describes the classification of pixels into background/foreground/shadow/highlight. Section 3 discusses a few issues regarding the shadow simulation. Next, section 4 describes our similarity measure between a classified image and a simulated image of a 3D box. The optimization of a 3D box to a classified image is then described in section 5. This section also includes some experiments comparing our method to the simpler methods of just simulating shadows or detecting and removing shadows.

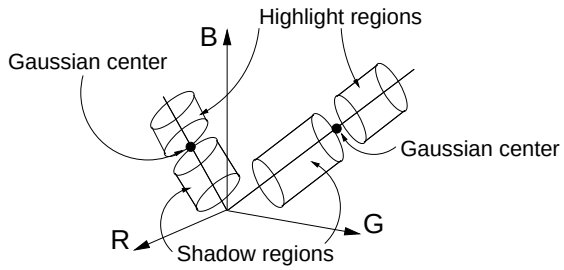

Fig. 1. Illustration of the shadow/highlight classification in the RGB color space.

Finally, section 6 describes a vehicle tracking system that makes use of the 3D box measurements.

2. Statistical background subtraction and foreground classification

We use the statistical method in (Wood, 2007) for classifying each pixel into background or foreground. The method is a modification of the well known Stauffer-Grimson background subtraction method (Stauffer and Grimson, 1999), with a somewhat different update rule and a lower regularization limit for the Gaussian standard deviations. Basically, a Gaussian mixture model is used in each pixel to estimate the color distribution over time and foreground pixels are detected if the color is unlikely to belong to the distribution.

The foreground pixels are further classified into foreground/shadow/highlight. In (Wood, 2007) the Stauffer-Grimson subtraction is combined with the shadow detection method in (Horprasert et al., 1999) to get a foreground/shadow classification. Basically, a pixel is classified as a shadow pixel if the color lies in a cylinder region between the black (the origin) and any of the center colors of the background Gaussians. Here we also apply the same idea for highlight as well, so that a pixel is classified as a highlight pixel if the color lies in a cylinder located on the opposite side of any of the Gaussian center colors. Figure 1 illustrates the class regions. We use a shadow cylinder that lies between 0.3 and 0.95 of the Gaussian center and with radius 15 (using RGB space with range [0,255]). The highlight cylinder lies between 1.05 and 1.5 above the Gaussian center and with radius 30.

We define highlight as pixels that do not belong to the foreground objects, and that are brighter than the average background color (as in figure 1). Highlight can occur from many different phenomena, e.g. reflection on a bright vehicle onto the ground, bloom and streak effects, and rapid light changes caused by moving clouds, see figure 2.

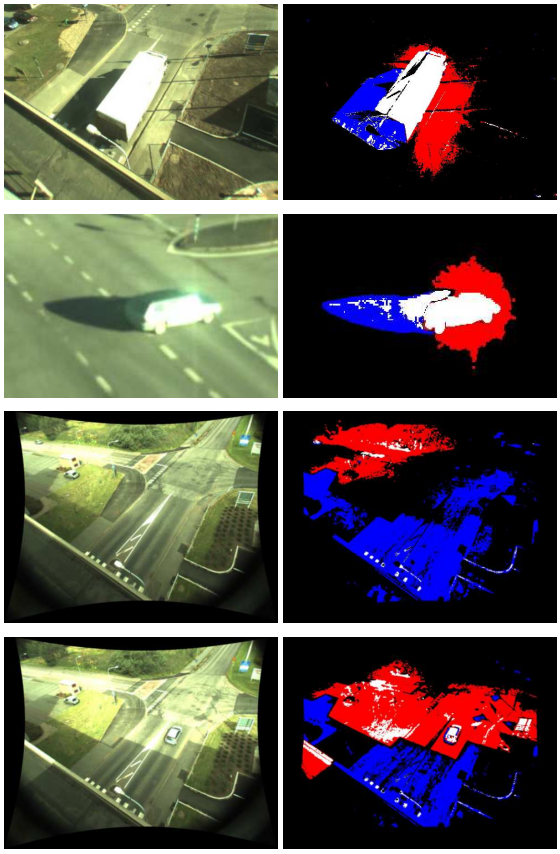

Fig. 2. Examples of highlight and shadow detection. Top: Reflection of a bright vehicle onto the ground. Second: Bloom and streak effects. Third and bottom: Sun reappearing from behind moving clouds. The colors, or gray-values depending on the type of visualization, denote the following: black=background, white=foreground, blue/darkgray=shadows, red/lightgray=highlight.

Figure 2 also shows the result of the foreground/shadow/highlight detection. Much of the highlight can be detected and removed by the algorithm, but there is still some misclassifications, for example the white region close to the center of the image in the bottom example.

Figure 3 shows two examples of the shadow detection. Note that there are some misclassifications, especially in the core shadow near the vehicle, i.e. the umbra region.

A fundamental problem is also the time window for learning of the statistical background model. It is difficult to tune the time window to suite all varieties of traffic density and light conditions. For example, (Joshi et al., 2007) mention that the learning rate of the background update is tuned depending on the

Fig. 3. Examples of misclassification in the umbra region. The Highway I sequence (second row) is used by courtesy of the Computer Vision and Robotics Research Laboratory of UCSD.

type of sequence and the expected speed of motion, which is not desirable when designing an automatic system. Heavy traffic, or if the light is varying too much due to moving clouds, will cause slower convergence of the background model since the model is not updated in the foreground/shadow/highlight regions. As we have seen, we can recover from some of the errors, by re-classifying parts of the foreground into shadows or highlight, but there will still be misclassifications, and it is desirable that the background model adapts to new conditions. Furthermore, vehicles that are temporarily stationary for a longer period of time will eventually disappear and become part of the background model. They may then leave a ghost vehicle behind when resuming their course.

3. Calibration for 3D box simulation

Let $\mathcal{B} = \{L, W, H, x, y, \varphi\}$ denote the state of a 3D box on a ground plane in 3D, where $(L, W, H) = (Length, Width, Height)$, (x, y) is the position on the ground plane, and φ is the orientation.

To simulate an image of a 3D box we need knowledge of the ground plane and the camera calibration parameters, i.e. lens distortion parameters and the projection matrix that defines the mapping of 3D world points to 2D image points. Lens-distortion calibration is done using the technique in (Claus and Fitzgibbon, 2005), and the projection matrix is found with the normalized DLT algorithm (Hartley and Zisserman, 2000), used on GPS coordinates of world points, which are subsequently indicated in the image. The ground plane transformation is needed, as it allows us to later do filtering of vehicle motion in physically meaningful quantities. If GPS data is difficult to obtain, the ground plane could in principle be found using automatic ground-plane rectification (Ardö, 2004), followed by indication of the ground plane north. However, a manual calibration has the additional advantage that the measurements obtained are in metric units, and are thus directly useable in driver behaviour studies.

Our calibrated setup allows us to accurately predict shadows. This is implemented using the software package (SOLPOS, 2000), which computes the solar position from date, time, and location on Earth.

To summarize, calibration allows trajectory filtering in world coordinates, prediction of shadows, and produces output trajectories in units useful for driver behaviour analysis. Thus we recommend it whenever possible.

4. Similarity function

Let $I(\mathbf{x})$ be the classified image and $S(\mathbf{x})$ be the simulated image, and represent the classes as follows:

$$I(\mathbf{x}), S(\mathbf{x}) = \begin{cases} 0 \text{ if background} \\ 128 \text{ if shadow} \\ 255 \text{ if foreground.} \end{cases} \quad (1)$$

Furthermore, let $\mathcal{V}$ denote the valid pixels, i.e. pixels that are not occluded by other objects (buildings, vehicles, image rectification borders, etc.). Define the similarity between the classified image and the simulated image as

$$s(I, S) = \sum_{\mathbf{x} \in \mathcal{V}} p(I(\mathbf{x}))S(\mathbf{x}) \,, \quad (2)$$

where

$$p(I) = \begin{cases} p_b < 0 \text{ if I=0 (background)} \\ p_s > 0 \text{ if I=128 (shadow)} \\ p_f > 0 \text{ if I=255 (foreground).} \end{cases} \quad (3)$$

For example $p_b = -0.2, p_s = 0.1, p_f = 1$. This function is quite ad-hoc, but it has some nice properties. A simulated box is rewarded more than its simulated shadow in a detected shadow region. For example, black cars are often misclassified as shadows,

in which case the box should be covering the region instead of the simulated shadow of the box.

Initially, this similarity measure was intended to improve the box fitting in sunny situations when the shadow was misclassified as foreground. But the improvement compared to just using one class (foreground) and simulating shadows was minor. This is because knowledge of the solar position (and box orientation) seems sufficient to reduce the ambiguities. Instead, the improvement was most evident on days with diffuse light when the light causes shadows to appear around the vehicles, and also when a vehicle does not cast a shadow (for example when the vehicle is already in the shadow of a building). Figure 4 shows some examples of optimal boxes (from the optimization in the next section) using this similarity function.

We also tried to use normalized correlation as a similarity measure, but the results were worse. Correlation often causes the optimization to terminate at some local optimum if the classified region is sparse. Our measure favors large boxes, which helps to glue sparse segments together, unless there is too much background in-between, or if the other segments are invalid (e.g. covered by another box).

5. 3D box optimization

We employ a fairly simple optimization method. Basically, the algorithm is initiated with a predicted box state (from e.g. a Kalman filter) and tries different changes of size and position iteratively to find the box state that gives the highest similarity measure. The changes in size are not arbitrary, but chosen from a list of common sizes, see table 1 for examples. The iteration loops over box sizes and position transformations, with gradually decreasing step size, see algorithms 1 and 2 for details.

Figure 4 shows some example results. For comparison, the same figure also shows the result when simulating, but not detecting shadows, as well as when detecting and removing shadow pixels (and without simulating shadows). As we show, the other methods are in some cases somewhat better, but the new proposed method does better on average.

A more quantitative comparison is given in figure 5. Here we have accumulated trajectories at two occasions (during one hour each) one in the morning, 9am, and one at noon (and thus with different solar angles). We use the tracking system described in section 6 to obtain the trajectories, but only as initialisation for the box optimisation. (Hence the figure shows non filtered measurements.) The trajectories are painted using Gaussian kernel density estimation (Bishop, 1995), which is sampled with a spatial distance of 0.25m, and with $\sigma = 0.15$m. As shown in the figure, removal of shadows reduces detections of vehicles that occupy only a small part of the image. Simulating shadows helps to detect small vehicles, but introduces clutter e.g. from shadows of waving flags, and trees. Our approach of shadow detection and simulation allows detection of small objects, without increasing the clutter.

Vehicle type	(Length, Width, Height)
Pedestrian	(1,1,2)
Motorcycle	(3.00,1.00,1.50)
Small car, VW Golf	(4.20,1.73,1.48)
V70	(4.73,1.86,1.56)
Van, VW Multivan	(4.89,1.90,1.94)
Van, slightly bigger	(6.00,2.00,3.00)
V70 with trailer	(7.73,1.86,1.56)
Truck, Volvo FL/FE	(9.00,2.50,4.00)
Extra ad-hoc size	(7.50,2.25,3.50)
Extra ad-hoc size	(10.00,2.50,3.50)
Extra ad-hoc size	(12.00,2.50,4.00)

Table 1
Examples of vehicle sizes.

The two experiments in figures 4 and 5 show some of the advantages of the combined detection and simulation. Another advantage appears to be an improved localization of the trajectories. However, a proper evaluation of this would require accurate groundtruth trajectories, as obtained from IMU+GPS sensors. Unfortunately we do not have access to such equipment at present, and this has to be left for future work.

We give some observations from experiments here. Initially, we tried to use free sizes of boxes in algorithm 1, i.e. using the transformations $\{\mathcal{T}_n\}$ = { Increase/decrease front position, Increase/decrease rear position, Increase/decrease height, Increase/decrease x, Increase/decrease y }. However this seemed too unstable if the foreground/shadow-classification fails too much, and using a fixed set of sizes to reduce the degrees of freedom gave a more stable performance.

We have also tried to optimize with the shadow simulation as a degree of freedom (testing both turned on and off), but this also gave a more unsta-

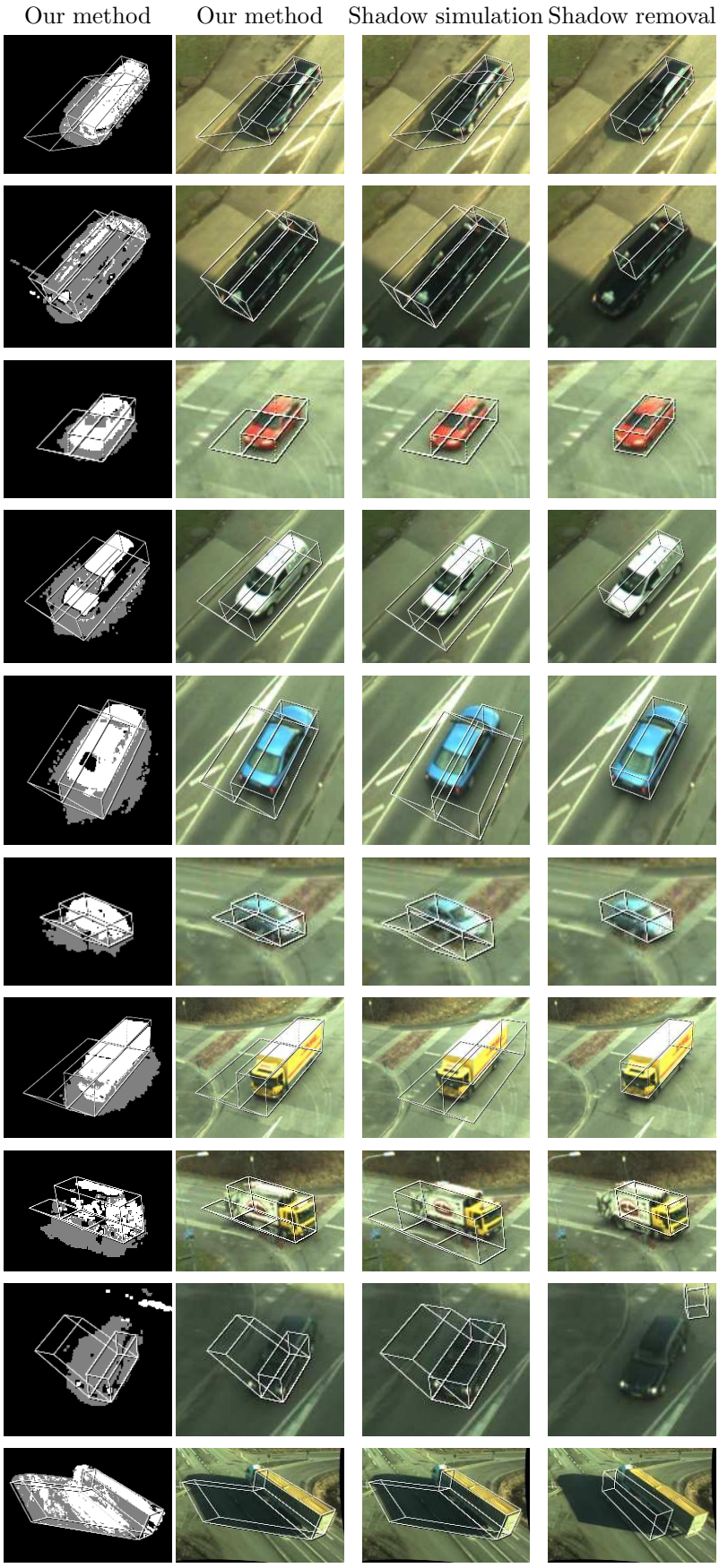

Fig. 4. Examples of results from algorithm 2, using fixed parameters in (3). First and second column: Optimal box using our method, overlaid on the classified and original image respectively. Third column: Optimal box when not detecting shadows (i.e. setting all shadow pixels in the left column to foreground). Fourth column: Optimal box when removing all detected shadow pixels and without simulated shadows. The orientation is manually selected here, but could for example be predicted from previous positions or in some cases from known lane orientation. The initial position is roughly estimated from the image blob.

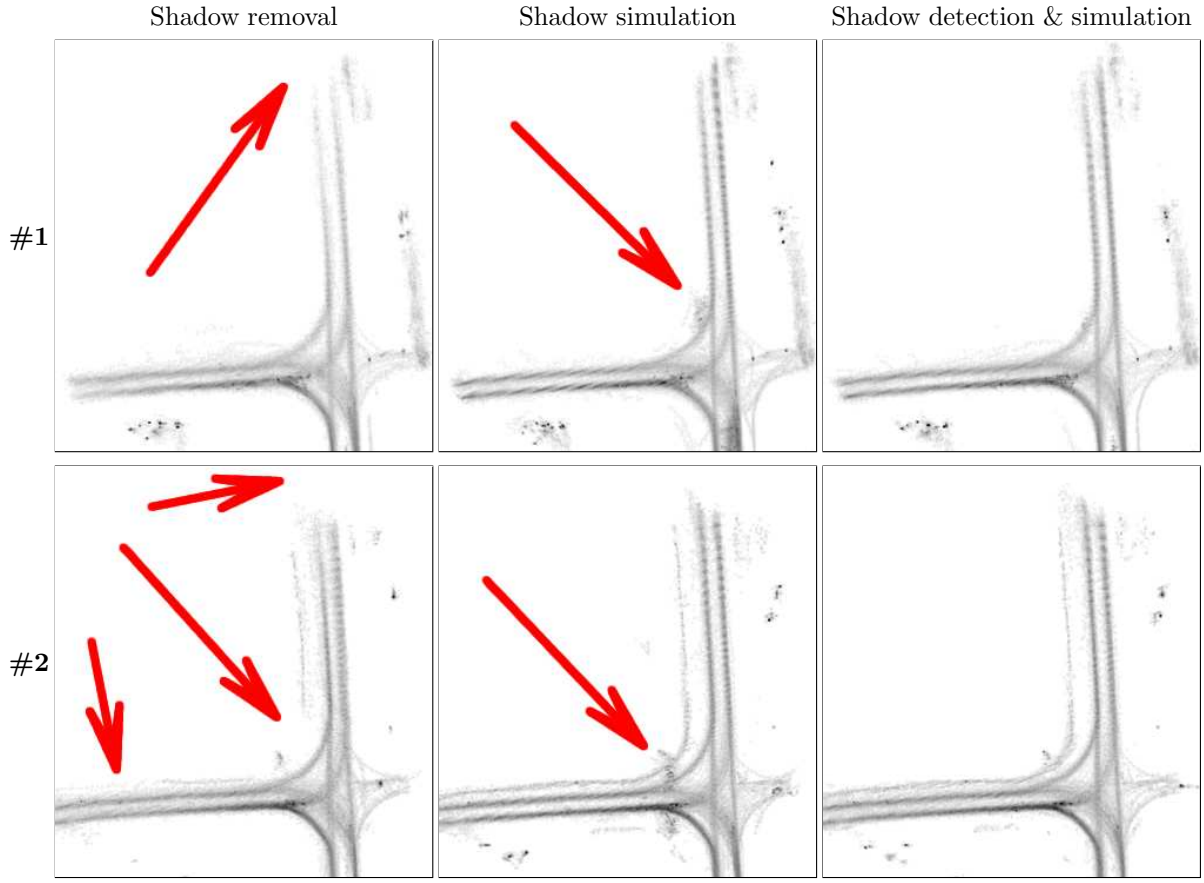

Fig. 5. Accumulated trajectories for two different runs in the intersection shown in figure 3, top left. Removal of shadows reduces detections of vehicles that occupy only a small part of the image (see regions at arrows in left column). Simulating shadows only helps to detect small vehicles, but introduces clutter (see regions at arrows in centre column) e.g. from shadows of waving flags, and trees. Shadow detection and simulation allows detection of small objects, without increasing the clutter.

ble result.

Moreover, we also tried to add change in orientation to the list of transformations, but it is too unstable (at least for some object views) to include a free change in orientation in the optimization when using only a single camera view. One solution might be to add a punishing term for the orientation in the similarity measure, i.e.

$$s' = s\cos(\varphi_0 - \varphi)^{p_\varphi sign(s)} \tag{4}$$

where φ_0 is the initial estimated orientation. However, the utility of this rule is still unclear.

We have also made experiments with merging of estimates from up to four cameras with partly overlapping views of an intersection. Initially we tried to optimize the 3D box to all views simultaneously, by adding the similarity measures from each camera. However, this requires a very accurate time synchronization between the cameras. Otherwise the optimized box usually becomes larger than the actual vehicle since the box tries to cover all camera blobs (since foreground is rewarded more than background is punished). If the cameras are not fully in sync, it thus appears to be better to optimize to each camera separately and then merge the resulting 3D boxes by e.g. averaging the state parameters.

The optimization algorithm is very crude and may sometimes end up in a local optimum if the initial position is too far from the global optimum. It can then be useful to add a few more random transformations to the list in algorithm 1, or to use more elaborate optimization schemes. However, the improvement has not been significant when initializing the optimization using a prediction from previous frames (e.g. using a Kalman filter), and the estimation of size can also be improved by averaging esti-

Algorithm 1 Optimization of box position (fixed size and orientation). Basically, the algorithm tries different positions, with decreasing step size, around the currently best value/state until the similarity measure is no longer increasing.

Let $\{\mathcal{T}_n\}$ = { Increase x, Decrease x, Increase y, Decrease y }, with initially 1 meter step size (then divided by two a few times during the iterations).

```
# Given size, orientation, and initial position
(L, W, H, x_0, y_0, φ) = ...
# Initial optimum
s_opt = −∞
B_opt = {L, W, H, x_0, y_0, φ}
# Loop over transformation scales
for m ∈ [0, ..., M] do
  # Loop over list of transformations as long as optimum is increasing
  iter = 0
  n = 0
  while iter < |{T_k}| do
    iter = iter + 1
    n = mod(n + 1, |{T_k}|) # Get next transformation
    B_new = transform(B_opt, T_n/2^m) # Transform box
    S_new = simulate_box_image(B_new) # Simulate a box image
    s_new = s(I, S_new) # Compute similarity
    # Store new optimum if new box is better
    if s_new > s_opt then
      s_opt = s_new
      B_opt = B_new
      iter = 0
    end if
  end while
end for
```

Algorithm 2 Optimization of box position and size (fixed orientation). Basically, the algorithm tries different sizes in algorithm 1.

```
# Given orientation, and initial position
(x_0, y_0, φ) = ...
# Loop over list of box sizes
s_opt = −∞
for all (L_k, W_k, H_K) in {(L_k, W_k, H_k)} do
  # Optimize position using algorithm 1
  (s_k, B_k) = algorithm_1(L_k, W_k, H_k, x_0, y_0, φ)
  # Store new optimum if new box is better
  if s_k > s_opt then
    s_opt = s_k
    B_opt = B_k
  end if
end for
```

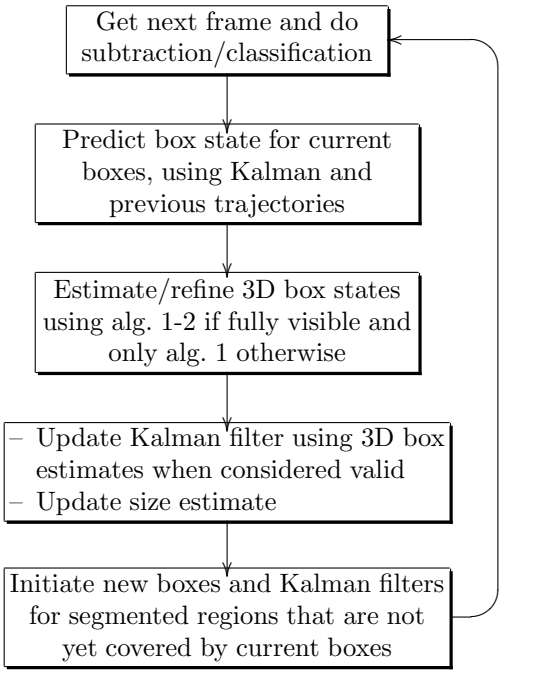

Fig. 6. Sketch overview of the vehicle tracking system.

mates in time (as described below).

6. Experiments in a tracking system

We have also used the 3D box optimization in a vehicle tracking system. It is beyond the scope of this paper to describe a fully functional tracking system for all traffic situations and weather conditions, but we will discuss some details and observations here. In essence, the 3D box position estimates are used as measurements to an extended Kalman filter (EKF) in the 3D ground plane domain, see figure 6 for an overview. The details can be worked out in many different ways, we propose some solutions here.

The vehicle size is estimated by weighting together the estimates from each frame in time. Each estimate of the (Length,Width,Height) triplet is weighted depending on the view from which the vehicle was seen. For example, if the vehicle is viewed from above, the height estimate gets a low weight and the length and width get higher weights.

When a vehicle becomes partly occluded (as can be detected by analyzing the Kalman predictions of all other current boxes, and manually labeled building regions etc.), it has seemed more stable to lock

the size to the current time average estimate during optimization. Hence, we use algorithm 2 if the box is considered to be fully visible, and algorithm 1 otherwise with other objects marked as invalid pixels in $\mathcal{V}$. Moreover, in cases where shadows are largely misclassified as foreground, it has appeared to be more robust to also include the regions that fall within simulated shadows as invalid pixels. This means that the tracker will treat vehicles that fall within (simulated) shadows from other vehicles as occluded.

We use a simple bicycle model in the extended Kalman filter that contains position, orientation, and speed, (x, y, φ, v). We basically only get position measurements, and more complex models may therefore cause instability. More complex models can however be applied as a post-processing step to further refine the trajectories.

The box orientation is predicted from the trajectory of previously collected estimates, by computing the tangent at the end of the trajectory in a certain spatial window (initially using the lane orientation). It has seemed more stable to use a spatially based window rather than a time based window like the orientation in the Kalman model. For vehicles that are temporarily stationary over extended periods the latter estimate may become unstable. In a recent paper (Atev et al., 2008) (which is an improved version of (Atev et al., 2005)), propose another model including curvature of the motion path which reduces the stop-and-go problem. But even if we use an exact vehicle model, noisy position measurements can still in theory make the car appear to move forward and backward and eventually turn around. It may also be possible to use optical flow as a hint for the orientation, as in e.g. (Song and Nevatia, 2007), but this only works for moving vehicles.

The 3D box estimates are sometimes corrupted and should then be treated as outliers. Various outlier tests can be applied such as thresholds for maximally allowed occlusion (both in 2D and 3D), lower limits for pixel area, threshold for maximal foreground sparsity inside a box, maximally allowed distance from the prediction, and so on. These tests may depend on the situation. For example, we have collected data from an intersection where flag poles on the side of the road cast moving shadows onto the road, which sometimes initiate new boxes. A 3D test for overlapping boxes would not allow these boxes to be 'run over' by true boxes.

Deciding when to initiate new boxes is also a difficult problem, especially in heavy traffic. In our system, new boxes are currently only allowed to be initiated if the segment does not overlap much with existing boxes. This means that new vehicles must be almost fully visible to be initiated, which also means that occluded vehicles that are lost (decided to be terminated) may disturb neighboring measurements.

7. Conclusions

We have in this paper presented an algorithm for estimation of size and position of vehicles that combines both shadow detection and shadow simulation. It is shown in some examples and experiments that this combination can improve the performance compared to only using shadow detection or shadow simulation. The improvement is most evident in cases where the shadow detection or the shadow simulation is inaccurate. There are some cases though where one of the comparison methods is somewhat better than our method, and a topic for future research may be to improve the similarity measure even further.

The paper also describes an example how to use the 3D box optimization in a tracking system. However, better performance is still needed in situations with large occlusions and heavy traffic. This is a well known and currently unsolved tracking problem.

Acknowledgment

This work has been supported by the Swedish Road Administration (SRA) within the IVSS project *Intersection accidents: analysis and prevention.*

References

Håkan Ardö, Use of learning for detection and tracking of vehicles, Proceedings of the Swedish Symposium on Image Analysis, 2004

Håkan Ardö and Rikard Berthilsson, Adaptive Background Estimation using Intensity Independent Features, Proceedings of the 2006 British Machine and Vision Conference BMVC'06, III:1069, September 2006

Stefan Atev, Nikolaos Papanikolopoulos, Multi-view 3D Vehicle Tracking with a Constrained Filter, IEEE International Conference on Robotics and Automation, 2277–2282, May 2008, Pasadena, CA, USA,

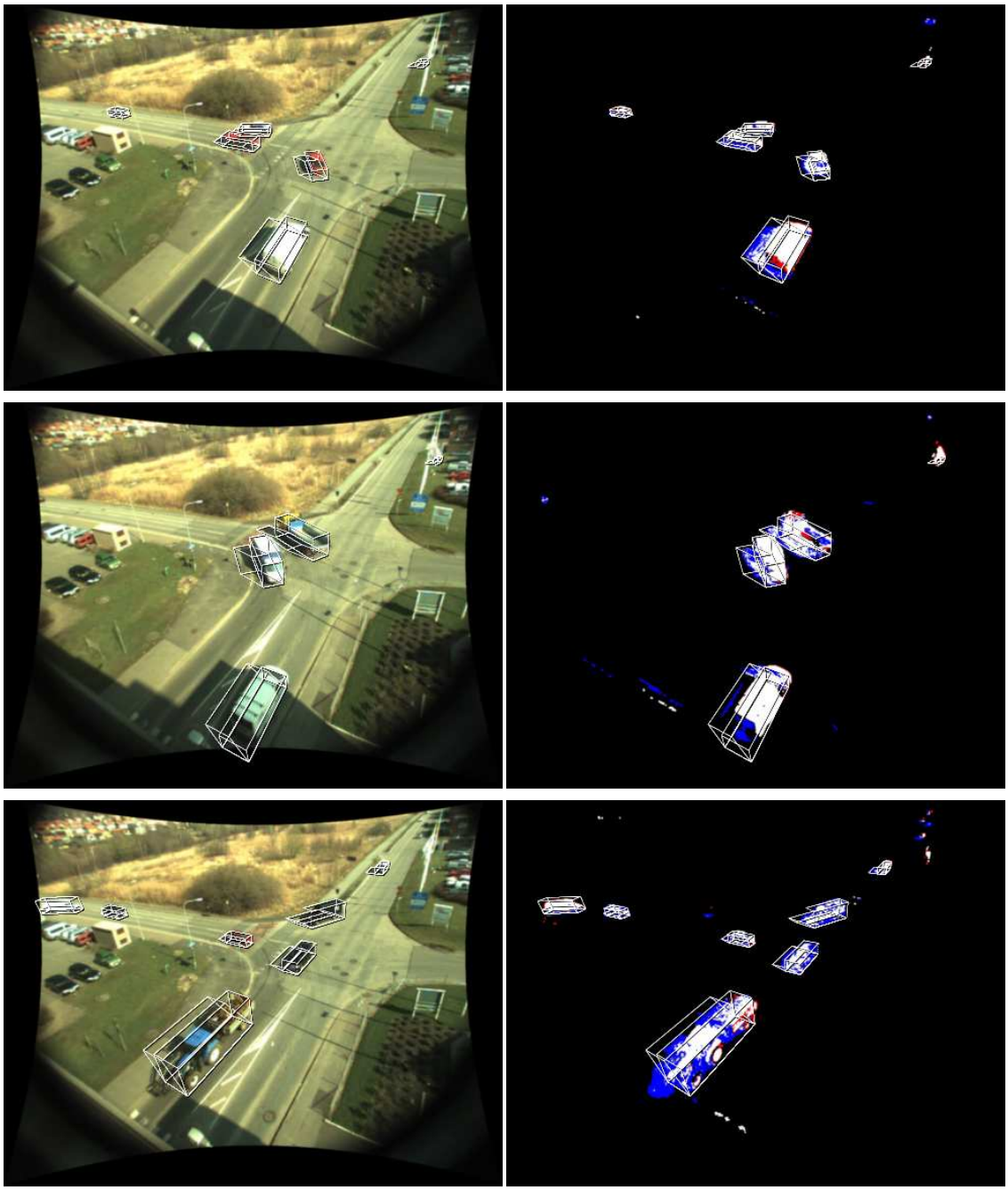

Fig. 7. Examples of tracking results.

Stefan Atev, Hemanth Arumugam, Osama Masoud, Ravi Janardan, Nikolaos Papanikolopoulos, A Vision-Based Approach to Collision Prediction at Traffic Intersections, IEEE Transactions on Intelligent Transportation Systems, 6(4):416–423, December 2005

Christopher M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995, ISBN 0-19-853864-2

Andreas Böckert, Vehicle Detection and Classification in Video Sequences, Linköping University, SE-581 83 Linköping, Sweden, 2002, Masters thesis LiTH-ISY-EX-3270

Enrique J. Carmona, Javier Martínez-Cantos, José Mira, A new video segmentation method of moving objects based on blob-level knowledge, Pattern Recognition Letters, 29:272-285, 2008

David Claus, Andrew W. Fitzgibbon, A Rational Function Lens Distortion Model for General Cameras, CVPR'05, 2005

Hendrik Dahlkamp, Hans-Hellmut Nagel, Artur Ottlik, Paul Reuter, International Journal of Computer Vision, A Framework for Model-Based Tracking Experiments in Image Sequences, 73(2):139–157, 2007

Hendrik Dahlkamp, Arthur E.C. Pece, Artur Ottlik, Hans-Hellmut Nagel, Differential Analysis of Two Model-Based Vehicle Tracking Approaches, DAGM'04, 71–78, 2004

R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000

Horprasert, T., Harwood, D., Davis, L.S., A Statistical Approach for Real-Time Robust Background Subtraction and Shadow Detection, Proceedings of IEEE ICCV'99 FRAME-RATE Workshop, 1999

Björn Johansson, Johan Wiklund, Gösta Granlund, Goals and status within the IVSS project, Seminar on "Cognitive vision in traffic analysis", May 2006, Lund, Sweden URL: `http://www.lth.se/index.php?id=10904`

Ajay J. Joshi, Stefan Atev, Osama Masoud, Nikolaos Papanikolopoulos, Moving Shadow Detection with Low- and Mid-Level Reasoning, IEEE International Conference on Robotics and Automation, April 2007

Zu Whan Kim, Jitendra Malik, Fast Vehicle Detection with Probabilistic Feature Grouping and its Application to Vehicle Tracking, Proceedings of ICCV'03, 524–531, 2003

Jianguang Lou, Tieniu Tan, Weiming Hu, Hao Yang, Steven J. Maybank, 3-D Model-Based Vehicle Tracking, IEEE Transactions on image processing, 14(10):1561–1569, October 2005

Derek R. Magee, Tracking multiple vehicles using foreground, background and motion models, Image and Vision Computing, 22:143–155, 2004

Hans-Helmut Nagel, Steps Towards a Cognitive Vision System, AI Magazine, 25(2):31–50, 2004

Andrea Prati, Ivana Mikić, Mohan M. Triveldi, Rita Cucchiara, Detecting Moving Shadows: Algorithms and Evaluation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(7):918–923, 2003

NREL, SOLPOS 2.0. Distributed by the National Renewable Energy Laboratory. Center for Renewable Energy Resources. Renewable Resource Data Center, February, 2000, URL: `http://rredc.nrel.gov/solar/codes_algs/solpos`

Xuefeng Song, Ram Nevatia, Detection and Tracking of Moving Vehicles in Crowded Scenes, IEEE Workshop on Motion and Video Computing, WMVC'07, 4–4, 2007

C. Stauffer, W. Grimson, Adaptive Background Mixture Models for Real-Time Tracking, IEEE Conference on Computer Vision and Pattern Recognition, 246–252, 1999

John Wood, Statistical Background Models with Shadow Detection for Video Based Tracking, Linköping University, SE-581 83 Linköping, Sweden, March, 2007, LiTH-ISY-EX–07/3921–SE