Three-Dimensional Vehicle Pose Estimation from Two-Dimensional Monocular Camera Images for Vehicle Classification

U.U.SHEIKH, S.A.R. ABU-BAKAR
Computer Vision, Video & Image Processing Lab, Dep. of Microelectronics and Computer Engineering,
Faculty of Electrical Engineering, Universiti Teknologi Malaysia,
81310 Skudai, Johor,
MALAYSIA

Abstract: - In this paper, a new method is proposed to estimate the pose of a moving vehicle in a typical traffic scene. Pose determination is crucial in the process of fitting or matching an existing 3D model in the database with the captured moving object. In this work, pose estimation is determined by estimating the 3D position of the moving vehicle in world space and then computing the motion vector of the vehicle. The pose estimation is first initialized by loosely calibrating the camera viewport and the perspective distortion. The 3D position is then determined by computing the intersection of a ray trace of the vehicle's centroid obtained from the video image originating from the camera eye to the vehicle's motion plane, i.e. the ground. Once a motion vector is obtained, the 3D model is aligned to match the vehicle's pose. The computation is performed on a 3D graphics card. Results on real-world traffic scenes as well as synthetic data are presented and several issues are outlined.

Key-Words: - Vehicle Pose Detection, 3D Pose Estimation, Model Matching

1 Introduction

Vehicle classification is one active area in intelligent transportation systems. It is important for traffic management and for obtaining traffic parameters. By classifying the vehicles into correct groups such as truck, car, motorcycle, traffic parameters such as flow and congestion can be obtained easily. There are several different methods used to classify vehicles, including techniques based on spatial information and 3D model fitting techniques.

One of the earliest researches on vehicle classification was done by A.D Houghton et al [1]. The method proposed used a simple vehicle outline template matching. Over the years, more advanced methods were proposed by researchers. Works on vehicle detection and recognition using image processing can be divided into several categories, based on the methods used. Researches based on stereo vision are such as by M. J. J. Burden et al. [2], T. Aizawa [3] and M. Kimachi [4].

Works which utilizes monocular vision and coupled with statistic analysis on 2D images include [5] which utilize the objects dimension to determine object type. Other parameters used are like Fourier descriptors [6], size and linearity [7], compactness and aspect ratio [8], vehicle edge information at selected points [9], and feature training using neural networks [10].

There are several works on model based or 3D based vehicle recognition. Among them are from [11] that use parameterized models applied using principal component analysis on video sequences. Wei Wu et al. [12] used models to train neural network for classification. In [13], a fixed camera position was used (*35 degrees looking angle and camera height of greater than 7m*) with combination of polyhedral models of 8 types which resulted in an accuracy of more than 90%. Other similar model based approaches are such as in [14, 15, 16] and [17].

The method of pose detection used in [11] is based on minimizing the measure of a reference model and the trained model using least squares approximation. The reference model is scaled and translated to fit the image. The traffic scene is limited to one direction only. In [12], the pose parameters $(x, y, \theta)$is first determined by the user manually to provide initial approximation. The pose is not calculated automatically, and the pose selected by the user is a fixed pose for the whole scene. This is only suitable for traffic conditions where turning does not occur, such as on highways. The works in [13] does not use any pose detection, although it is model based. Instead, the system proposed uses pre-computed models saved in the database. The moving object is then matched to the models in the database. For a typical junction, the system keeps over 2.2 million models. Although a large number of models are available, an organization method is proposed for fast model searching. On top of that, intrinsic camera parameters are required.

T.N. Tan et al. [14], proposed a generalized Hough transform with explicit probability-based voting models to identify vehicle pose. The method proposed is

proven to work well in traffic scenes. Perspective transformation matrix and vehicle scale must be known beforehand. In [15], the author used a recursive estimation of shape parameters to estimate the pose and motion parameters.

2 Motivation

Most real-time vehicle classification systems do not use 3D models for classification, due to the complexity in processing such as the determination of object pose. Using only spatial information, the level of classification is degraded due to limited information available. On the other hand, many techniques which use 3D models employ expensive and iterative matching algorithm to determine vehicle pose. This makes them inadequate for real-time applications, especially when there are multiple vehicles in a scene.

It is therefore, the aims of this proposed technique is to simplify the pose detection problem and improve the overall vehicle classification system. Before determining the pose of the vehicle, there are several assumptions made, which are;

1. the ground plane is relatively flat, at least as it is viewed from the camera,
2. the motion of the vehicles are limited to the ground plane, and
3. the vehicles are moving forward is all cases.

3 Proposed Method

A central problem in computer vision and computer graphics is to fit an existing 3D model of an object to a scene given its image. This is because a real-life scene (*the world space*) is in 3D, $W = \{(x_w, y_w, z_w) \in \mathbb{R}^3\}$ in the sense that it has a depth of view. However, when a scene is projected to the camera plane, the depth information if flattened and thus removed, leaving only two-dimensional information in the image, $S = \{(x_s, y_s) \in \mathbb{Z}^2 : 0 \leq x_s < m-1, 0 \leq y_s < n-1\}$. In such case, the object pose cannot be determined from the image, because the depth information is lost, leaving us with only the (x, y) coordinates.

The transformation operation of a 3D homogenous point in world space $\boldsymbol{\lambda} = [x_w, y_w, z_w, 1]^T \in W$ to a 2D point $\boldsymbol{\gamma} = [x_s, y_s]^T \in S$ in an image or screen space is due to the transformations of camera viewport, and projective transformation. Projective geometry is an extension to Euclidean geometry and encompasses not only transformations like rotations and translations but in particular the perspective projection performed by the camera. The perspective projection is important because it projects and distorts the objects in the viewing frustum on to the image plane, in this case the screen space. Projective coordinates represent naturally the operation performed by a camera.

In determining the pose of the vehicle, the scene perspective projection must be known like in [14], and in addition to that, a ground plane must be defined. The perspective projection is closely related to the camera calibration parameters, however in the proposed technique, such information is not required. Only an approximation of perspective is needed and can be obtained by visually inspecting the scene. The overall pose detection flow is shown in figure 1.

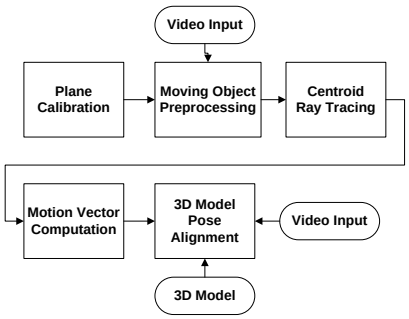

Figure 1. The process of pose detection

3.1 Camera Calibration

The first step is to 'fit' a plane (*ground*) to the scene on which the moving vehicles actually are on. The ground is required to reduce the computation of the ray-tracing process in the preceding section.

As mentioned, the ground in assumed to be flat. In building a model of the scene, we define out the ground as flat plane, $\boldsymbol{\Pi} = [\pi_1, \pi_2, \pi_3, \pi_4]^T \in \mathbb{R}^3$ which represents the equation of the plane $\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$ and $\sum_i \pi_i \neq 0, \{i = 1..4\}$. The plane $\boldsymbol{\Pi}_\infty = [0,0,0,1]^T$ is known as plane at infinity and the point $\boldsymbol{\lambda}_\infty$ is the vanishing point in the perspective projection.

The purpose of the perspective projection and ground plane calibration is to simulate the actual projection of the world space to the screen space. In reality, objects in the distance appear smaller compared to objects nearer to the camera. This is also true for the distance of motion. Although in this work, the plane is assumed to be flat, the method proposed is not limited to flat surfaces. An arbitrary terrain can also be used. The total transformation required to obtain a 2D point from a 3D scene is by applying the world, view and perspective transformation, where each $\mathcal{P}, \mathcal{V}, \mathcal{T}$ is a 4x4 transformation matrix. The world matrix basically determines the position of the plane in the 3D world, and the viewport matrix determines how the camera views the world. The perspective matrix distorts the image to simulate a perspective distortion.

$$(\mathcal{P} \circ \mathcal{V} \circ \mathcal{T})(\boldsymbol{\lambda}) = \boldsymbol{\gamma}$$

Perspective transformation matrix used in this work is similar to that available in Microsoft®'s DirectX® 9.0 and OpenGL®. The perspective transformation is given as [18];

$$M_{\mathcal{P}} = \begin{bmatrix} \cot(\frac{fov_w}{2}) & 0 & 0 & 0 \\ 0 & \cot(\frac{fov_h}{2}) & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -QZ_n & 0 \end{bmatrix}$$

$$Q = \frac{Z_f}{Z_f - Z_n}$$

whereby fov_w and fov_h is the field of view in the horizontal and vertical directions of the viewing camera. While Z_f and Z_n are the far and near clipping planes of the viewing frustum.

Figure 2 shows how a plane is fitted interactively by the user for a scene with the vanishing point on the top left. The only thing required is the field of view of the camera, which can be obtained from the focal length.

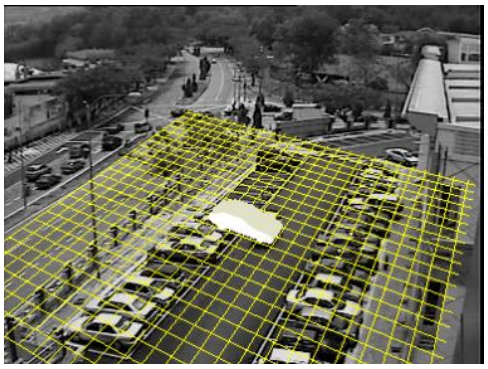

Figure 2. A plane mesh is calibrated according to perspective of the scene

3.2 Object Preprocessing

Once the plane has been setup, the next step is to process the incoming video. The incoming video will be in the screen space, $S \in \mathbb{Z}^2 : \{0 \leq x_s < m-1, 0 \leq y_s < n-1\}$, with a dimension of $m \times n$. We need to extract the moving vehicle from the video. There are several methods to extract moving objects, such as frame differencing, background subtraction and optical flow. In this proposed method, a simple background subtraction method is used as suggested by Heikkila and Olli [19].

The color image is first converted to grayscale. The current frame is then subtracted with a background model as defined below [19] and then binarized by a predefined threshold value, τ ;

$$S_d = |\boldsymbol{I}_t - \boldsymbol{B}_t| > \tau : \{0 \leq \tau \leq 255\}$$

The background model is then updated;

$$\boldsymbol{B}_{t+1} = \alpha \boldsymbol{I}_t + (1-\alpha)\boldsymbol{B}_t$$

$$: \{\alpha \in \mathbb{R}: 0 \leq \alpha \leq 1, t \in \mathbb{Z}^+\}$$

Where $\boldsymbol{B}_t$ is a background model at time t. Because this is a pose estimation stage, the quality of the moving blob is not crucial. Several processes of morphological dilation and erosion are performed after the thresholding process to obtain a clear and filled object. Median filter is applied to clear up any noise in the image.

To trace the ray projection of the vehicle, a ray origin point must be determined. From the binarized 2D difference image S_d, the ray origin is determined by taking the centroid of the vehicle's blob. In the case of several vehicles, each vehicle's centroid is calculated. The centroid (x_{cs}, y_{cs}) is actually the moments of the blob, and can be calculated as [20];

$$u_{pq} = \sum_{x_s=0}^{m-1} \sum_{y_s=0}^{n-1} {x_s}^p {y_s}^q$$

$$x_{cs} = \frac{u_{10}}{u_{00}}, y_{cs} = \frac{u_{01}}{u_{00}}$$

3.2 Position Estimation using Ray-Tracing

Once the vehicle's centroid is obtained from the video, the 3D position of the vehicle in the 3D virtual world must be determined. After all, the 3D pose estimation is to be calculated. Assume that the 2D video image is actually a projection of the 3D virtual world, hence

$$(\mathcal{P} \circ \mathcal{V} \circ \mathcal{T})(W) = S_d$$

thus

$$\forall \boldsymbol{\lambda}\{(x_w, y_w, z_w) \in W\} \ \exists \ \boldsymbol{\gamma}\{(x_s, y_s) \in S_d\}$$

However, this will yield too many points to be resolved. To minimize the number of points, the approach will only limit the 3D points ($\boldsymbol{\lambda}$) which lies only on the plane $\boldsymbol{\Pi}$. Thus, the number of points are reduced to,

$$\forall \boldsymbol{\lambda}\{(x_w, y_w, z_w) \in \boldsymbol{\Pi}\} \ \exists \ \boldsymbol{\gamma}\{(x_s, y_s) \in S_d\}$$

To determine which $\boldsymbol{\gamma}$ point corresponds to its counterpart in the 3D space, $\boldsymbol{\lambda}$, a ray-trace is used. Ray-tracing is normally used for surface rendering and bump mapping. In this case, a 2D point $\boldsymbol{\gamma}(x_{cs}, y_{cs})$ is used as a start point for ray-tracing. Instead of calculating the light properties, the ray-tracing is used to check the collision of a ray from the 2D point to the reference plane $\boldsymbol{\Pi}$ *(or terrain)*. The point of collision is the 3D point $\boldsymbol{\lambda}$. In

other words, assume a ray is projected from a 2D point $\boldsymbol{\gamma}(x_{cs}, y_{cs})$ which is multiplied with the inverse of the projection matrix known to us to obtain the ray in view space. Assume this point as the ray origin and ray direction in 3D as $\boldsymbol{r}_o = [x_o, y_o, 1.0]^T$ and $\boldsymbol{r}_d = [x_d, y_d, z_d]^T$. We can then compute the ray vector as $\boldsymbol{r} = \boldsymbol{r}_d - \boldsymbol{r}_o$. Since a ray is just a straight line (*but in 3D space*), the intersection can be obtained by solving the ray vector with the plane equation. Given an equation of a straight line (*ray*) in $\mathbb{R}^3$ as;

$$\boldsymbol{R}(t) = \boldsymbol{r}_o + \boldsymbol{r}_d t$$

Substitute this in the plane equation,

$$\pi_1(x_o + x_d t) + \pi_2 (y_o + y_d t) + \pi_3 (z_o + z_d t) + \pi_4 = 0$$

$$t = \frac{-(\pi_1 x_o + \pi_2 y_o + \pi_3 z_o + D)}{\pi_1 x_d + \pi_2 y_d + \pi_3 z_d}$$

$$t = \frac{-(\boldsymbol{P}_n . \boldsymbol{R}_o + D)}{\boldsymbol{P}_n . \boldsymbol{R}_d}$$

$$t = \frac{\boldsymbol{V}_o}{\boldsymbol{V}_d};\ \boldsymbol{\lambda}\ exists\ iff\ \boldsymbol{V}_d > 0$$

The intersection point for the vehicle is obtained by substituting t in the following equation;

$$\boldsymbol{\lambda} = [x_o + x_d t, y_o + y_d t, z_o + z_d t]^T$$

This developed method has the advantage of being extremely flexible. Transformation is performed online during runtime, and in our implementation, the calculation is offloaded to the 3D graphics accelerator.

In the case of a terrain, T is used instead of a simple plane, the ray-terrain intersection can be computed by checking ray intersection with all the vertexes, υ of the terrain. In that case, we compute the ray-trace which produces the shortest ray distance, d .

$$\arg\min_d \{raytrace(\boldsymbol{R}(t), \forall\, \upsilon \in \mathrm{T})\};\ d > 0\ \&\ t > 0$$

3.3 Pose Estimation

Once the 3D position in obtained, the vehicle's pose can be determined. To compute the pose, two set of 3D positions are required, the 3D position at the current time $\boldsymbol{\lambda}_t$ and the 3D position at the previous frame $\boldsymbol{\lambda}_{t-1}$. The vehicle pose, $\boldsymbol{\phi}$ is just the difference between the two vectors.

$$\boldsymbol{\phi} = \boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}$$

For this to work properly, a tracking mechanism is required to keep a history and associate 3D positions with the correct object blob, in the case of more than one moving object. The computation of this $\boldsymbol{\phi}$ pose vector is shown in figure 3.

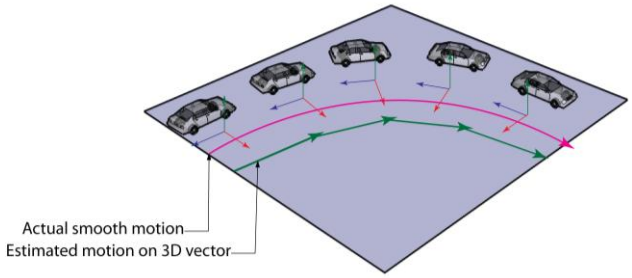

Figure 3. Pose detection

4 Experimental Results

In order to demonstrate its usability, the proposed pose estimation algorithm has been applied on different types of video sequences and different types of scenes. In figure 4, a simple synthetic scene whereby a toy car is making a U-turn. First the ground plane is calibrated. Once that has been done, the moving toy car is extracted and the centroid is defined. From there on, the 3D position is calculated using the ray-trace method proposed above, and the two consecutive 3D positions are saved. A vector is calculated to determine the pose. Finally, a 3D model from the database is aligned to have the same pose with the moving toy-car. From this experiment, it is found that the accuracy of pose is close to $\pm 5^o$. The inaccuracy is caused by the finite resolution of the video image (384×288) pixels.

Figure 5 shows an actual scene showing a curved movement of a vehicle in a typical street scene. The vehicle takes right through the T-junction. The algorithm is capable of matching a 3D model pose with the moving vehicle.

5 Conclusion

A novel method to determine vehicle pose from 2D video image has been presented. The method proposed utilizes 3D information to compute the 2D pose. The overall algorithm was implemented using C++ and executes at more than 30 frames per second on a 1.66GHz Intel Core Duo processor. Besides that, part of the algorithm is off-loaded to be computed on a 3D graphics card (*such as an ATI Radeon or NVidia GeForce*) series which supports DirectX 9.0 capabilities.

In the experiments performed, the algorithm is capable of achieving pose estimation with an accuract of $\pm 5^o$. This in return, makes it suitable for further image processing such as for vehicle classification. The

proposed method improves vehicle detection, which is a very critical task in video surveillance and vision-based traffic monitoring systems. With a correct pose determined, correct features can be extracted for vehicle classification. Our complete system includes vehicle classification and velocity estimation.

The simplicity of the algorithm is proven with no exact values of the camera setup is required. Only focal length and ground plane setup is required. Although the algorithm is robust enough, it is still limited to several key issues. For a scene where the moving object is moving too slowly, the computation of pose vector will become incorrect due to very small distance travelled in the 2D image. This will cause the pose vector $\boldsymbol{\phi}$ to have incorrect direction, causing wrong pose being estimated. The mechanism described is only limited to forward moving vehicles.

6 Acknowledgment

This project has been made possible through the research contract grant managed by Universiti Teknologi Malaysia (UTM) under the vote 68305. The authors would like to express their gratitude for this support to UTM.

References:

[1] A.D. Houghton, G.S. Hobson, N.L. Seed, R.C. Tozer, Automatic Vehicle Recognition, *Road Traffic Monitoring, 1989., Second Int. Conf.*, 7-9 Feb 1989, London, UK, pp. 71-78

[2] M. J. J. Burden, M. G. H. Bell, Vehicle Classification Using Stereo Vision, *Image Processing and Its Applications, 1997., Sixth International Conference on,* 14-17 July 1997, Dublin, pp. 881 - 885 vol.2

[3] T. Aizawa, A. Tanaka, H. Higashikage, Y. Asokawa, M. Kimachi, S. Ogata, Road Surface Estimation Robust against Vehicles' Existence for Stereo-Based Vehicle Detection, *The IEEE 5th Int. Conf. on Intelligent Transportation Systems*, 3-6 September 2002, Singapore, pp. 43-48

[4] M. Kimachi, Y. Wu, S. Ogata, A Vehicle Recognition Method Robust against Vehicle's Overlapping Based on Stereo Vision, *IEEE 1999*, pp. 865-869

[5] Andrew H.S. Lai, George S.K. Fung, Nelson H.C., Vehicle Classification from Visual-Based Dimension Estimation, *2001 IEEE Intelligent Transportation Systems Conf. Proc.,* Oakland (CA), USA, August 25-29, 2001, pp. 201-206

[6] D. Toth, Til Aach, Detection and Recognition of Moving Objects using Statistical Motion Detection and Fourier Descriptors, *ICIAP 2003*, Mantova, Italy, September 17-19, 2003, pp. 1 -6

[7] S. Yu, J. Hsieh, Y. Chen, W. F. Hu, An Automatic Traffic Surveillance System for Vehicle Tracking and Classification, *SCIA 2003*, Springer-Verlag Berlin Heidelberg 2003, pp. 379-386

[8] L. Bo, Z. Heqin, Using Object Classification to Improve Urban Traffic Monitoring System, *IEEE Int. Conf.e on Neural Networks and Signal Proc.*, Nanjing China, Dec. 14-17, 2003, pp. 1155 -1159

[9] X. Ma, W. Eric L. Grimson, Edge-based Rich Representation for Vehicle Classification, *Proc. of the 10th IEEE Int. Conf. on Computer Vision (ICCV'05),* 17-21 Oct. 2005, pp. 1185 - 1192

[10] N.D. Matthews, P.E. An, D. Charnley, C.J. Harris, Vehicle Detection and Recognition in Greyscale Imagery, *Control Engineering Practice* Vol. 4. No. 4, Elsevier Science Ltd., 1996, pp. 473-479

[11] X. Limin, Vehicle Shape Recovery and Recognition Using Generic Models, *Proc.s of the 4th World Congress on Intelligent Control and Automation*, IEEE 2002, Shanghai, China, June 10-14 2002, pp. 1055 – 1059

[12] W. Wu, Z. QiSen, W. Mingjun, A Method of Vehicle Classification Using Models and Neural Networks, *VTC'01*, IEEE 2001, pp. 3022 – 3026

[13]S. Messelodi,C. M. Modena, M. Zanin, A Computer Vision System for the Detection and Classification of Vehicles at Urban Road Intersections, *Pattern Analysis & App.*, 2005, pp.17-31

[14] T.N. Tan, G.D. Sullivan, K.D. Baker, Model Based Localisation and Recognition of Road Vehicles, *Int. Journal of Computer Vision* 27, 1998, pp. 5-25

[15] D. Koller, Moving Object Recognition and Classification based on Recursive Shape Parameter Estimation, *In Proc. 12th Israel Conference on Artificial Intelligence, Computer Vision,* Ramat Gan, Israel, December 27-28, 1993, pp. 359-368

[16] M. Schmid, An Approach to Model Based 3D Recognition of Vehicles in Real Time by Machine Vision, *Proc. of Int. Conference on Intelligent Robots and Systems, IROS'94*, pp. 2064-2071

[17] G.D. Sullivan, K.D. Baker, A.D. Worrall, C.I. Attwood, P.R. Ramagnino, Model-Based Vehicle Detection and Classification using Orthographic Approximations, *Proc. of 7th British Machine Vision Conference*, Vol. 2, Sep 1996, pp. 695-704

[18] James M., V. Verth, L. M. Bishop, *Essential Mathematics for Games and Interactive Applications*, Morgan Kauffman Publisher, 2004.

[19] J. Heikkila, O. Silven, A Real-time System for Monitoring of Cyclists and Pedestrians, *2nd IEEE Workshop on Visual Surveillance*, Fort Collins, Colorado, June 1999, pp. 82-90

[20] M. K. Hu, Visual Pattern Recognition by Moment Invariants, *IRE Trans. Information Theory*, Vol. 24, No. 8, pp. 179-187.

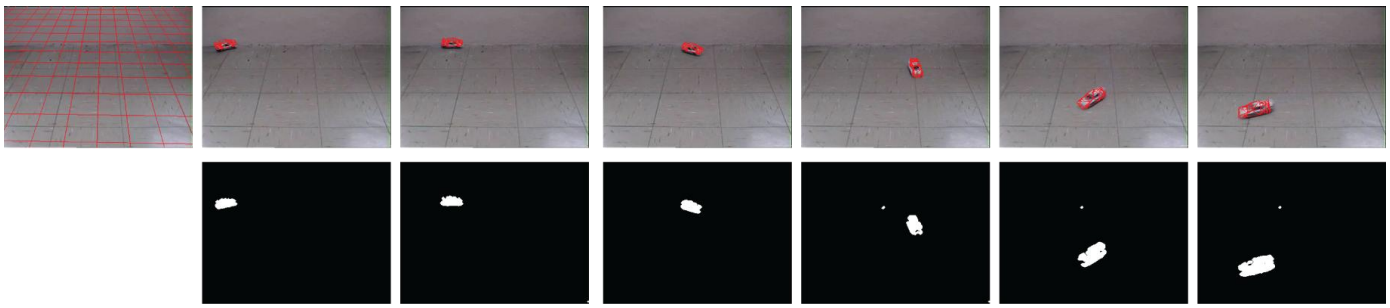

Figure 4. Pose detection for a toy car, (top-left) calibrated ground plane, (top-row) actual 2D video superimposed with the pose detected 3D model, (bottom-row) background subtraction blob

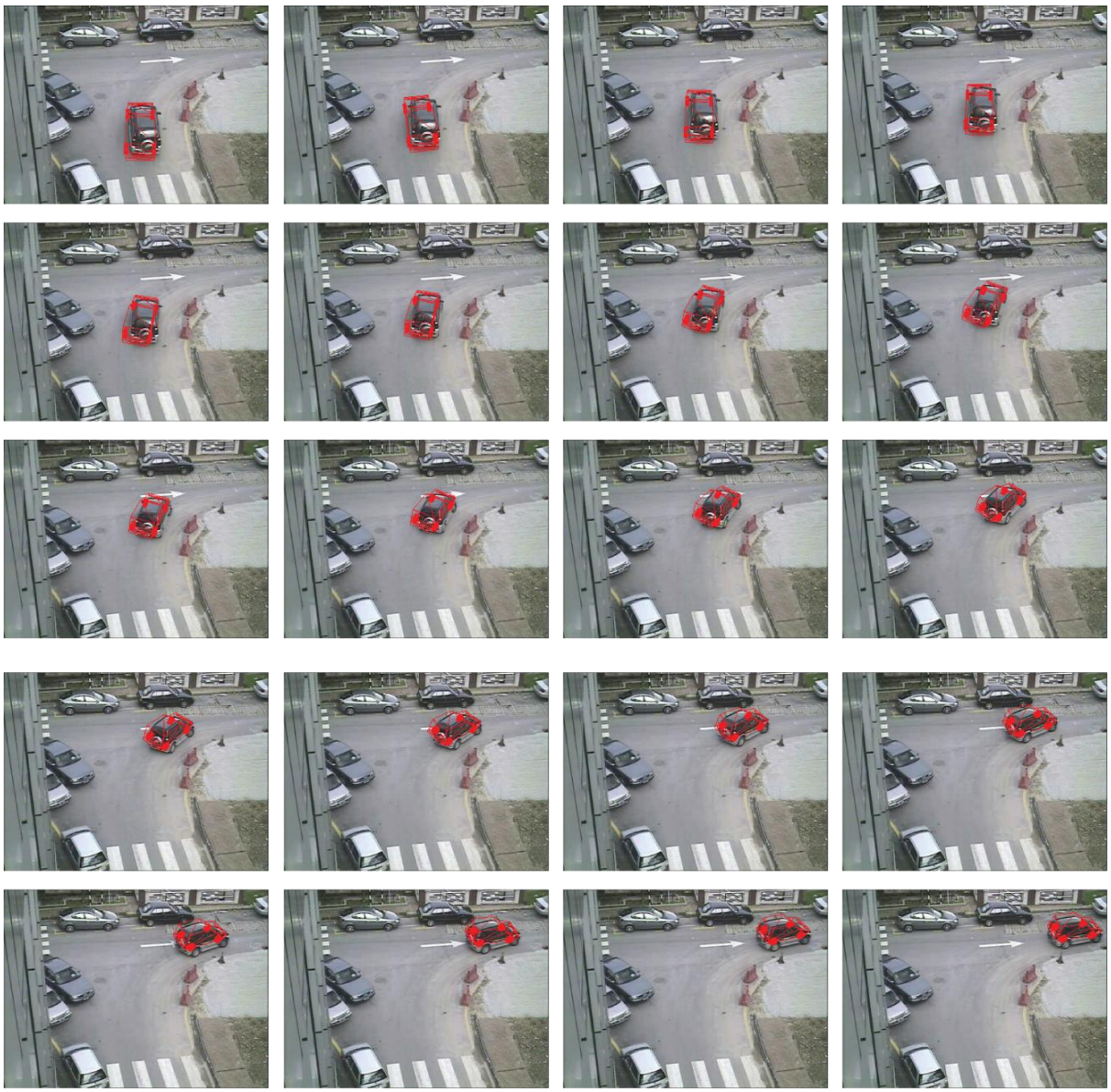

Figure 5. Pose detection in a real-world scene