Model-Based Vehicle Pose Estimation and Tracking in Videos Using Random Forests*

Michael Hödlmoser[1] Branislav Micusik[2] Marc Pollefeys[3] Ming-Yu Liu[4] Martin Kampel[1]

[1] CVL, Vienna University of Technology
[hoedl,kampel]@caa.tuwien.ac.at

[2] AIT Austrian Institute of Technology
branislav.micusik@ait.ac.at

[3] Computer Vision and Geometry Lab, ETH Zürich
marc.pollefeys@inf.ethz.ch

[4] Mitsubishi Electric Research Labs (MERL)
mliu@merl.com

Abstract

This paper presents a computational effective framework for tracking and pose estimation of vehicles in videos reaching comparable performance to state-of-the-art methods. We cast the problem of vehicle tracking as ranking possible poses for each frame and connecting subsequent poses by exploiting a feasible motion model over time. As a novelty, we use random forests trained on a set of existing 3D models for estimating the pose. We discretize the viewpoint space for training, where a synthetic camera is orbiting around the models. To compare projections of 3D models to real world 2D input frames, we introduce simple but discriminative principle gradient features to describe both images. A Markov Random Field ensures to pick the perfect pose over time and the vehicle to follow a feasible motion. As can be seen from our experiments performed on a variety of videos with vast variation of vehicle types, the proposed framework achieves similar results in less computational time compared to state-of-the-art methods.

1. Introduction

Visual traffic surveillance is an important task in computer vision which enables multiple novel applications ranging from estimating the scene topology and geometry of a traffic scene, controlling the traffic activities of 3D objects present in the scene, detecting abnormal behaviors and making control decisions in real time up to autonomous vehicle navigation. Using 3D information helps when performing vehicle detection, classification and pose estimation. First, the appearance of objects varies substantially with the viewing angle and local features may often be occluded in 2D. Second, using 3D information allows making some a-priori assumptions about the scene and relaxing the problem (*e.g.* a car is more likely to be located on a road than in the sky). Unfortunately, most of the traffic surveillance systems do only provide a single camera, which means that the 3D information of a recorded scene gets lost. Going back from a monocular video stream to the 3D reconstructed scene is therefore an ill-posed problem.

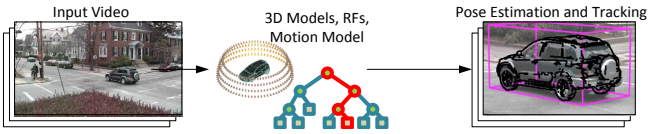


Figure 1. A computationally effective 3D model based vehicle detection and tracking framework. See text for details.

Object tracking in 3D space can be seen as a combination of determining the pose for each frame and these poses following a feasible motion model over time. Traditional pose estimation approaches try to exploit the diversity or so-called intra-class variations by extracting a discrete set of the diversity to perform object classification [5, 6]. Other approaches learn a sparse 3D model from training images taken from various viewpoints [17, 18]. Recent methods try to exploit 3D models having known dimensions [16, 23, 9] or generic models [12] for object classification, pose estimation and tracking.

In this paper we exploit recent developments and try to overcome the objects' intra-class variations as well as the ill-posed problem between a monocular image stream and its 3D reconstruction by using a calibrated, monocular video stream and 3D models having known dimensions for pose estimation and tracking in videos.

As a contribution, we introduce a random forest (RF) ensemble, which is trained on a set of existing 3D models and used to rank the vehicles' possible poses and locations in real world input frames (see Fig. 1). Different to [9], the pose estimation does not rely on the correct 3D model being in the training set. We have a generic classifier trained

*Supported by FFG projects 835916 (PAMON), 830042 (CAPRI) and CogVis Ltd. M.-Y. Liu contributed to the work prior to joining MERL.

on multiple models, whereas [9] needs to have the correct model available and evaluates all models in various poses to obtain the correct pose. The synthetic camera orbiting around the model gives us correctly labeled training images based on the viewpoints and provides the advantage that we do not suffer from wrongly classified training images due to manual pose estimation in real world images. We introduce simple but discriminative principle gradient features to describe the training images as well as the input images. These features give us the opportunity to describe synthetically rendered objects without using depth, texture or color information. Note that of course, the models can also be rendered with color and texture information but the synthesized appearances usually differ from the real world counterparts due to illumination and other ambient factors. To the best of our knowledge, RFs trained on existing 3D models have never been used to overcome the problem of pose estimation. Different to [3], we do not use regression based RFs for pose estimation which gives us the opportunity to keep possible poses for each frame and remove outlier poses at a later stage over time which makes the whole pipeline more robust. We therefore use a classification based approach, rank all possible poses and incorporate Markov Random Field (MRF) to ensure temporal consistency between poses of consecutive frames, as proposed by [9].

2. Related Work

The field of object tracking and pose estimation using existing 3D models has been studied extensively in the last few years. A polyhedral 3D vehicle model for classification, following a motion model over time, was first introduced by [10]. Detailed 3D models of cars and motorcycles are first exploited in [14] for classification in still images. The training is performed using a synthetic camera orbiting around the models. Classification is done by comparing all possible projections to the input image. An approach for matching vehicles in still images under large body transformations using detailed 3D models is described in [8]. They gain an initial pose from meta-data, match the 2D image projection to the 2D model projection by using Chamfer distance and the Iteratively Closest Point algorithm. Rendering is done on manually labeled semantic parts, where occlusions in the rendering are handled by filling gaps using an MRF. The approach of [14] was extended to videos in [23]. The area of the projected model is matched to a segmented foreground input image mask. The area overlap as well as the shape similarity is used as matching score. The temporal inference is established by introducing a Conditional Random Field. The authors in [24] utilize real images, HOG features and RFs for training the pose estimator. Different to our approach, they are not using 3D models. We proposed to use principle gradient edge features instead of HOG features because we found it empirically works better for the 2D to 3D model matching problem. Using a deformable model [11] for vehicle tracking was first introduced by [12]. The authors are changing parts of the model online between consecutive frames and align the rendered projection with the input image. A Kalman filter is used to predict a pose from a frame to a subsequent one.

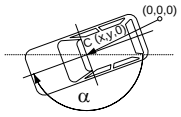


Figure 2. Vehicle, described by orientation α and its centroid on the ground plane $\mathbf{C} = (x, y, z = 0)$.

The principle of an RF is described in [1] and extended by [2]. The classifier is a very popular method in computer vision classification problems due to the fact that i) it can handle large training data sets and many classes, ii) it is robust against outliers and iii) the classification task is performed very quickly. Using classification-based RFs for pose estimation of humans was first introduced by [19]. Human people are detected in a variety of real world images and a training set is built up to generate an RF. Classes are generated by encountering both the pose and the action of a human. Using regression instead of classification is described in [7]. Objects are found by exploiting the generalized Hough transform. Detections of object parts individually vote for the localization of the complete object. By using depth images, regression is used for determining a human's head pose [3]. These features are also used in [21] for obtaining a human's pose. Each body part casts votes for a single class. The final pose is estimated by generating confidence-scored 3D proposals of how the body parts are connected. Conditional RFs are used for pose estimation in [22] which allows incorporating relationships between output variables by a global latent variable.

3. Vehicle Pose Estimation in Videos

In this paper we propose a framework for accurate 3D vehicle tracking and pose estimation by using existing 3D models and a video as input source. As others [12, 9], we assume, that given the ground plane, the vehicle's pose is parameterized by $\mathbf{p} = (x, y, \alpha)$ as shown in Fig. 2. The car's centroid on the ground plane is therefore denoted as $\mathbf{C} = (x, y, z = 0)$, its orientation is described by the angle α. Tracking a vehicle can be seen as finding the perfect pose in continuous space for each frame and connecting subsequent poses by exploiting a feasible motion model over time. Due to computational complexity, evaluating all possible poses is not feasible in practice, so we cast the problem as the determination of a set of poses in discrete space. Having a video, which provides N frames and given

a discrete input vehicle sequence $\mathcal{S} = \{\mathbf{s}_1 \dots \mathbf{s}_t \dots \mathbf{s}_N\}$ we want to find the pose sequence $\mathcal{R} = \{\mathbf{p}_1 \dots \mathbf{p}_t \dots \mathbf{p}_N\}$. This is established by finding the best matching model projection at each time instance t as well as determining the best transition between consecutive frames. This means the model must move in 3D space by following a feasible motion. Finding a solution to this problem is done by calculating the sequential inference which is established by using an MRF, a chain-structured undirected graphical model. The pose for a current frame is therefore inferred from past and future poses in a batch process. Given $\mathcal{S}$, the joint distribution for a model sequence $\mathcal{R}$ is denoted by

$$P(\mathcal{R}|\mathcal{S}) = \frac{1}{Z(\mathcal{S})} \prod_{t}^{N} Y(\mathbf{p}_t|\mathcal{S}) Y(\mathbf{p}_t, \mathbf{p}_{t-1}|\mathcal{S}), \quad (1)$$

where $Y(\mathbf{p}_t|\mathcal{S})$ is the matching score between a vehicle pose and the vehicle shown in the video at time instance t, $Y(\mathbf{p}_t, \mathbf{p}_{t-1}|\mathcal{S})$ describes the transition of the model between consecutive frames and $Z(\mathcal{S})$ assures a probability distribution. The following sections explain how to get both terms of Eq. 1 for our specific tracking problem.

3.1. Discrete Vehicle Pose Definition

In order to rank poses for each frame, we first need to specify which poses are possible and how similar poses are clustered to the same class. We therefore place the 3D model at the origin of the coordinate system and orbit a synthetic camera around it. Vehicles may provide a vast variety of dimensions and shapes but all of them provide some common features if seen from the same viewpoint. We therefore propose to tag all projections of multiple models seen from the same viewpoint with a common class label. As proposed by [14], we orbit around the object in discrete space to decrease computational complexity by varying azimuth and elevation angle, as well as the distance between the synthetic camera and the coordinate center, as can be seen in Fig. 3. To avoid misclassifications because of having too many classes, we sample the 3D model by a azimuth stepsize of $5°$, an elevation stepsize of $10°$ and a distance stepsize of one meter. We set these parameters to be an empirically found trade-off between having an accurate enough pose estimation whilst avoiding too many classes. We do not use regression based RFs. Using regression has the advantage of penalizing poses based on their distance to a correct result but the problem is that this method does not provide any confidence how well a certain pose fits the input frame. When the regression based RF obtains a completely wrong pose, our framework will not recover any more which will yield to a wrong result. Using classification instead gives us the opportunity to count the number of trees voting for a specific class, take this as the pose confidence and remove outliers over time. Decreasing

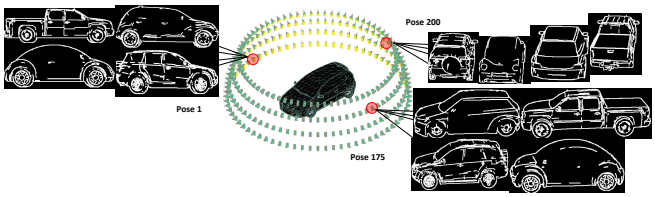


Figure 3. Discrete rendering of 3D models placed at the origin of the coordinate system. Multiple models in the dataset seen from the same viewpoint are clustered into the same class, where each class is defined by azimuth and elevation angle of the synthetic camera as well as a distance between the camera and the origin.

the stepsize means increasing the number of classes. Having too many classes is equal to using regression with the same penalty for all wrong poses which does also result in increasing the number of misclassifications and may not be handled by further steps anymore. Vehicles provide low-textured but specular surfaces. Following [12, 18, 9], edges are the most informative and stable features to use in this case. In our framework, we are therefore use synthetic 3D model projections. Our training data is a set of synthetically rendered 3D models using a synthetic camera with a given focal length. Edges must be shown when sharp edges between adjacent faces or edges from the projection's silhouette occur. We perform this rendering on the GPU due to computational complexity. Contour edges are found by projecting all faces onto the 2D image plane. Sharp edges are drawn when normals of adjacent faces of a 3D model are pointing in different directions. For obtaining realistic looking projections, this difference is empirically chosen to be $20°$.

3.2. Random Forests for Pose Estimation

For each frame, we rank all model projections based on how well they fit to the input frame. This can efficiently be done by exploiting RFs. An RF is a classifier which consists of multiple decision trees and can be used for solving multi-class labeling problems. It was first introduced by [1] and extended in [2]. By evaluating a feature using all the trees in the forest, each tree votes for a class. Each of those features of course only provides a weak assumptions about which class it belongs to but by evaluating multiple features and cleverly building up the decision forest, the whole classifier is proven to be very robust [21, 13].

Let an RF, built up by $b = 1 \dots B$ of trees, classify between $l = 1 \dots L$ classes. Each tree consists of split nodes (denoted by $\circ$ in Fig. 4), built up by a feature θ and a threshold as well as leaf nodes (denoted by $\square$ in Fig. 4). The threshold is used for following the tree to its left or right branch. Each of the trees then votes for a single class l. We then use the whole vote distribution of the forest for ranking the poses.

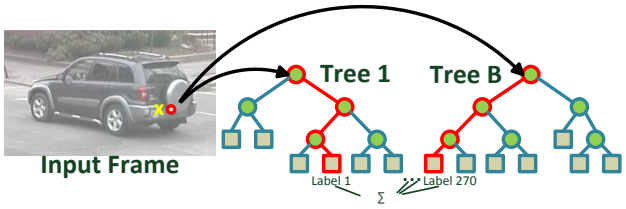


Figure 4. Pose estimation by classification using RFs and multiple features extracted at pixel $\mathbf{X} = (x, y)$. See text for details.

3.2.1 Random Forest Training

In our proposed framework, multiple principle gradient features describe a 3D model projection. Synthetically generated data is a synthesized copy of its real world counterpart without any noise, occlusions, or variations. The classification of real world images can therefore be improved by intentionally varying the training images on which the classifier is trained. This can be done by changing the threshold for sharp edges, by introducing noise, and by varying the 2D location of the projection. All these variations can be seen in Fig. 5, where the introduction of noise is performed by simply placing the model projection on a different background image.

We first determine the gradient direction image $G = \arctan(\frac{G_x}{G_y})$, where each pixel represents an angle in radians and G_x and G_y are the derivatives of an input image in both horizontal and vertical direction. Since we are using a synthetic camera, all training images are aligned to each other. Given a pixel location $\mathbf{X} = (x, y)$, we determine two points $\mathbf{X}_1 = (x_1, y_1)$ and $\mathbf{X}_2 = (x_2, y_2)$, both having a random offset ϕ to $\mathbf{X}$ in both directions. Given a certain blocksize (20 pixels in our experiments), we calculate the mean gradient directions $\psi(\mathbf{X}_1)$ and $\psi(\mathbf{X}_2)$ of each block. The feature at location $\mathbf{X}$ is then computed by

$$\theta(\mathbf{X}, \phi) = \mod(\arctan(0.5(\sum_{i=1}^{2} \sin(\psi(\mathbf{X}_i))), 0.5(\sum_{i=1}^{2} \cos(\psi(\mathbf{X}_i)))), 2\pi). \quad (2)$$

To cover variations between classes, we roughly even distribute $M = 500$ features randomly over the area covered by all 3D model projections to obtain random features $\theta(\mathbf{X}, \phi_1) \dots \theta(\mathbf{X}, \phi_M)$. To build up an RF, each tree within the same RF chooses a location X at random but common over all nodes of the tree. For each of the nodes of the tree, $\mathbf{X}_1$ and $\mathbf{X}_2$ are chosen randomly. As stated in Sec. 3.1, we render the models using a stepsize of one meter for the distance between the model and the camera center. We decided to generate an RF for each distance in order to keep misclassifications and therefore wrong poses ranked high

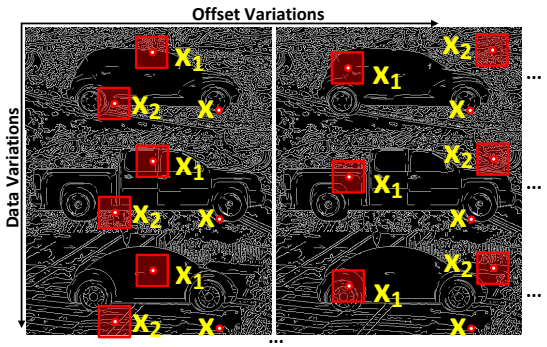


Figure 5. Feature Extraction for a variety of different projections but for the same class. A class is defined by the same viewpoint, where robustness to outliers is gained by placing the projection on different backgrounds, changing the threshold for sharp edges at the rendering process, shifting the model in 2D and using a variety of different 3D models.

on a minimum. This gives $L = 288$ classes for each RF within our framework.

3.2.2 Single Frame Pose Estimation

The RF can be used for localizing the vehicle in 2D. However, it is time-consuming because all decision trees at each pixel location have to be evaluated. We hence did not use the RF for localization but a state-of-the-art object detector [4]. It is applied at each frame and returns a bounding box for the 2D vehicle location. The detector by its nature does also provide a rough pose estimation but it does not provide any 3D information. Hence, the detector's pose output cannot be used in this case. As can be seen in [23] it is obviously not enough to perform background subtraction to obtain an accurate foreground mask due to highlights and shadows in the scene.

We determine the vehicle's pose and its location on the ground plane by exploiting an RF ensemble. We use $a = 1 \dots A$ RFs having $b = 1 \dots B$ trees, where each tree is trained with a different distance between camera and vehicle model. To increase efficiency, the number of RFs to be evaluated is minimized by determining the rough distance between the car and the camera center. For our experiments, we choose the threshold between the rough distance between camera and 3D model and the trained distance of the RFs to be $\pm$ 1.4 meters, so that only the closest three RFs are being evaluated for each frame. We then extract a feature vector $\mathbf{\Theta}(\mathbf{X}) = \theta(\mathbf{X}, \phi_1) \dots \theta(\mathbf{X}, \phi_M)$ with the same offsets $\phi_1 \dots \phi_M$ as in the training stage. The vehicle's 2D bounding box in the gradient image G of the input image is aligned with the training images and the features are calculated for the 2D bounding box of each video frame. Each tree from the RF ensemble holds a vote distribution $P_{a,b}(l|G, \mathbf{\Theta}(\mathbf{X}))$ for label l. The probability for label l is

then given by

$$P(l|G,\mathbf{\Theta}(\mathbf{X})) = \frac{1}{A}\sum_{a=1}^{A}\frac{1}{B}\sum_{b=1}^{B} P_{a,b}(l|G,\mathbf{\Theta}(\mathbf{X})). \quad (3)$$

To solve the first part of Eq. 1, the matching score at time instance t between a model projection $\mathbf{p}_t$ and an input vehicle $\mathbf{s}_t$ is given by

$$Y(\mathbf{p}_t|\mathbf{s}_t) = P(l|G,\mathbf{\Theta}(\mathbf{X})). \quad (4)$$

3.3. Temporal Inference for Accurate Vehicle Tracking in Videos

After having a matching score, we determine the transition term $Y(\mathbf{p}_t,\mathbf{p}_{t-1}|\mathcal{S})$. Ideally, the vehicle should move with constant speed, which is established by using slow speed and applying a L_2 norm minimization. We ensure a feasible motion model by applying the Ackermann steering principle, [20]. The principle is defined by $\varphi = \frac{\gamma}{2}$, where φ and γ are the angles of the vehicle's inner and outer turning radius, respectively. Forcing the vehicle to move with constant and feasible motion is therefore described by [9]

$$Y(\mathbf{p}_t,\mathbf{p}_{t-1}|\mathcal{S}) = exp(-(\|\mathbf{p}_{t-1}-\mathbf{p}_t\|^2+\lambda_2(\varphi-\frac{\gamma}{2}))), \quad (5)$$

where λ_2 assures equal weighting between the impact of constant and feasible motion, $\mathbf{p}$ is the car's pose. For solving Eq. (1) and finding the best fitting sequence $\hat{\mathcal{R}}$ for a given $\mathcal{S}$, we need to determine the maximum posterior (MAP) configuration through the graph and compute both Eq. (4) for each of the model projections and Eq. (5) for each edge, where an edge should be from each projection of frame at time $t-1$ to each one at time t. The final MAP of the MRF is then given by $\hat{\mathcal{R}} = \text{argmax}_{\mathcal{R}}\, P(\mathcal{R}|\mathcal{S})$.

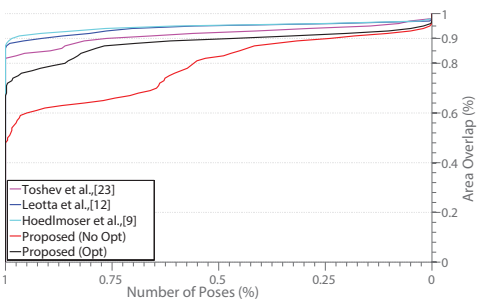
4. Experiments

We test our algorithm on five sequences from [12] showing a variety of different vehicles from different viewpoints. They provide a resolution of 1280x720 pixels and a calibrated setup. As the ground truth distances for our sequences range from 16-32 and 125-140 meters, we trained 32 RFs (stepsize 1 meter), where each of them consists of 200 trees, with an azimuth angle ranging from 0° - 360°, stepsize 5°, an elevation angle ranging from 0° to 30°, stepsize 10°. We use 9 different existing 3D models[1] to train the pose estimator (see Fig. 6) where the dimensions are taken from the manufacturers' specifications.

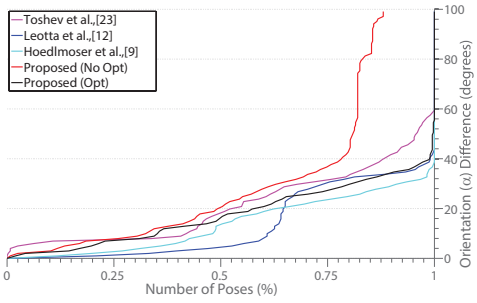
Quantitative experiments: We show the output for each step of our pipeline (RF only (No Opt) and applying MRF (Opt)) and compare them to Toshev's [23], Leotta's [12] and Hoedlmoser's [9] results. For generating ground truth data, we manually segment the 2D area of the vehicle as foreground for each frame. Since the approaches of [23] and [9] are not publicly available, we re-implemented them. For Toshev's method we used the manual foreground segmentation which prevents wrong classifications due to a bad segmentation. To assure fairness in our comparison, we re-implemented both methods such that we get comparable performance on similar videos. The dataset of [23] is generated with the same parameters as ours. Leotta's approach [12] is publicly available. First, we compare the overlap between the ground truth region and the projected 3D model. As this metric is used for the evaluation of the detected pose, it is only valid when the correct type of the vehicle is projected onto the image plane. When the vehicle's pose is known and a correct 3D model is available, its type can be determined by rendering all vehicle types with the determined pose. The best matching vehicle type is obtained by using FDCM, which is an edge-based matching method and known to be very robust against intra-class variations [15, 9]. We use the same models for determining the type as for training the pose estimator. In our experiments we obtain a correct vehicle type for all sequences, [23] misclassified the vehicle shown in sequence 1. Fig. 7(a) shows the overlap rate for our implementation (No Opt, Opt) as well as for Toshev's, Leotta's and Hoedlmoser's approach. As can be seen, we obtain similar results compared to their methods. Toshev's approach is using all scales of a discretely rendered model. This can lead to misclassifications, where the wrong, discrete scale of a wrong model may fit better than the next best discrete scale of the correct model. As can be seen in Fig. 7(a), this does not directly influence the overlap between ground truth and detected model. Therefore we compare the offset of the vehicle's centroid and its orientation α on the ground plane to the ground truth data (Tab. 8(b)). We clearly outperform Toshev's implementation and get similar results compared to Leotta's implementation both in terms of mean location and orientation error. Toshev's method is worse since it does not incorporate a ground plane estimation but does only classify each single frame based on the area overlap between training data and the input frame. To obtain a fair comparison, we compensate the missing ground plane of [23] by forcing poses to be oriented roughly in the ground truth direction of movement ($\pm 30°$). Fig. 7(b) shows the percentage of poses being below a certain difference between the calculated orientation α and the ground truth data. Our optimized pose estimator obtains comparable results to all other methods.

Figure 6. Models used for training the pose estimator.

[1] The models can be downloaded from http://www.caa.tuwien.ac.at/cvl/people/hoedl/



(a)

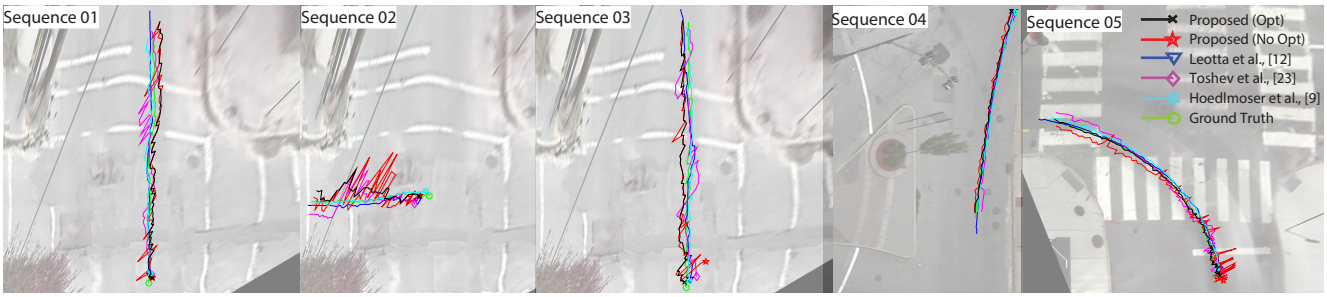


(b)

Proposed (No Opt)	Proposed (Opt)	Toshev et al.,[23]	Leotta et al.,[12]	Hoedlmoser et al.,[9]
0.73	0.87	21.13	4.16	127.87

(c)

Figure 7. (a) Area overlap between ground truth and results from all methods over all sequences for correctly classified vehicles. (b) Difference of orientation α over all sequences between all methods and ground truth data. (c) Mean processing time for one frame in seconds .



(a)

Location Distance on Ground Plane to Ground Truth (cm)

	Proposed(Opt)	Proposed(No Opt)	Toshev et al.,[23]	Leotta et al.,[12]	Hoedlmoser et al.,[9]
μ	30.15	49.12	42.94	29.85	25.38
σ	26.85	38.52	33.21	16.36	11.98
max	202.67	332.24	288.45	120.17	66.29

Orientation Distance on Ground Plane to Ground Truth (degrees)

	Proposed(Opt)	Proposed(No Opt)	Toshev et al.,[23]	Leotta et al.,[12]	Hoedlmoser et al.,[9]
μ	17.60	28.72	20.96	13.20	13.06
σ	11.90	38.82	14.58	14.49	11.05
max	55.04	120.17	58.95	43.30	37.68

(b)

Figure 8. (a) From left to right: Top-view 3D Tracks of the car's centroid on a warped top-view image for all sequences. Comparison between our approach (No Opt, Opt), as well as [23], [12], [9] and ground truth. (b) Offsets of the vehicle's center (left) and orientation α (right) on the ground plane compared to ground truth. Best viewed in color.

More than 50% of all poses yield an error smaller than $15°$. Fig. 8(a) shows the trajectory of the vehicle's centroid over a whole sequence for each method for all sequences from left to right. The vehicle's starting position is denoted by $\times$, $\star$, $\bigtriangledown$, $\diamondsuit$, $\hexagon$, $\circ$ for Opt, No Opt, Leotta, Toshev, Hoedlmoser and ground truth, respectively. As can be seen, there are more jumps in space between consecutive frames using Toshev's method than using ours since we assume the 3D model to be located on the ground plane. The main advantage of our approach is that we reach comparable results to state-of-the-art methods in much less computational time. The mean processing time for one frame can be seen in Tab. 7(c). As can be seen, we provide a much faster runtime than the approach presented in [9] since they need to evaluate all models in order to obtain a matching score between 3D model and input video. We do only compare pose estimation times for each method, where detecting and tracking the vehicle are excluded. The test is performed on an Intel i5, 2.4 GHz and 8GB RAM, our method, Toshev's and Hoedlmoser's are implemented in Matlab, Leotta's in C++.

Qualitative experiments: Fig. 9 shows from left to right the best matching five poses for random frames. The rightmost image shows the best matching pose and its corresponding rank taken from the MRF. As can be seen, the best pose must not be ranked first to get a smooth result but all highly ranked poses are similar to the correct one. Fig. 10 shows qualitative example results using our approach. One sequence is represented by four columns where each column corresponds to our method, [23], [12] and [9]. The upper left image of each sequence presents its first frame and our optimized projected track. The following frames of each row provide the refined pose in combination with the vehicle type estimated by FDCM projected onto the image plane. For viewing purposes, we crop out the region around

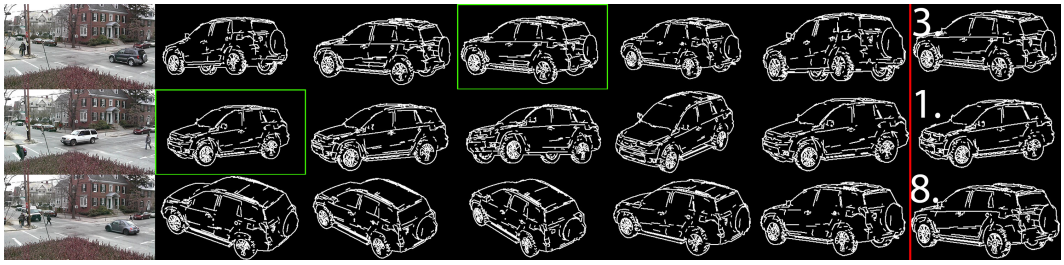


Figure 9. Best five poses for three random frames ranked from left to right. The rightmost image shows the pose and its rank chosen from the MRF. Note that we use a random model type for visualization.

the vehicle. As can be seen, we obtain comparable results for all evaluated methods. The last four columns on the lower right show a wrong model projected onto the image plane.

5. Conclusion

We presented a framework for estimating vehicles' poses in videos by exploiting existing 3D models having known dimensions. We cast the problem of vehicle tracking as ranking possible poses for each frame and connecting subsequent poses by exploiting a feasible motion model over time. We use a classification-based approach to rank all possible poses for each frame. The random forest classifier is trained on a synthetic set of existing 3D models, rendered from discrete viewpoints. Different to existing approaches, we train a generic pose estimator on a variety of 3D models which does therefore not rely on having a corresponding 3D models in the training set to the vehicle shown in the input frame. A novel edge based feature was introduced to match 2D projections to 2D input frames. These features give us the opportunity to describe synthetically rendered objects and compare them to their real world counterparts. A Markov Random Field ensures a feasible vehicle motion between consecutive frames. As can be seen from our experiments, we obtained similar results compared to state-of-the-art methods but dramatically outperformed them in terms of processing time.

References

[1] Y. Amit and D. Y. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997. 2, 3

[2] L. Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001. 2, 3

[3] G. Fanelli, J. Gall, and L. V. Gool. Real time head pose estimation with random regression forests. In *CVPR*, pages 617–624, June 2011. 2

[4] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32:1627–1645, 2010. 4

[5] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *ICCV*, 2005. 1

[6] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005. 1

[7] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *PAMI*, 33(11):2188–2202, 2011. 2

[8] Y. Guo, C. Rao, S. Samarasekera, J. Kim, R. Kumar, and H. Sawhney. Matching vehicles under large pose transformations using approximate 3d models and piecewise mrf model. In *CVPR*, 2008. 2

[9] M. Hoedlmoser, B. Micusik, M.-Y. Liu, M. Pollefeys, and M. Kampel. Classification and pose estimation of vehicles in videos by 3d modeling within discrete-continuous optimization. In *3DIMPVT*, pages 1–8, 2012. 1, 2, 3, 5, 6

[10] D. Koller. Moving object recognition and classification based on recursive shape parameter estimation. In *ICAI*, 1993. 2

[11] M. Leotta. *Generic, Deformable Models for 3-D Vehicle Surveillance*. PhD thesis, Brown University, Providence, RI, 2010. 2

[12] M. Leotta and J. Mundy. Vehicle surveillance with a generic, adaptive, 3d vehicle model. *PAMI*, 33(7):1457–1469, 2011. 1, 2, 3, 5, 6

[13] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, pages 775–781, 2005. 3

[14] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *CVPR*, 2008. 2, 3

[15] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, A. Agrawal, and H. Okuda. Pose estimation in heavy clutter using a multi-flash camera. In *ICRA*, 2010. 5

[16] J. Lou, T. Tan, W. Hu, H. Yang, and S. J. Maybank. 3-d model-based vehicle tracking. *Image Processing*, 14:1561–1569, 2005. 1

[17] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009. 1

[18] N. Payet and S. Todorovic. From contours to 3d object detection and pose estimation. In *ICCV*, 2011. 1, 3

[19] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. Torr. Randomized trees for human pose detection. In *CVPR*, 2008. 2

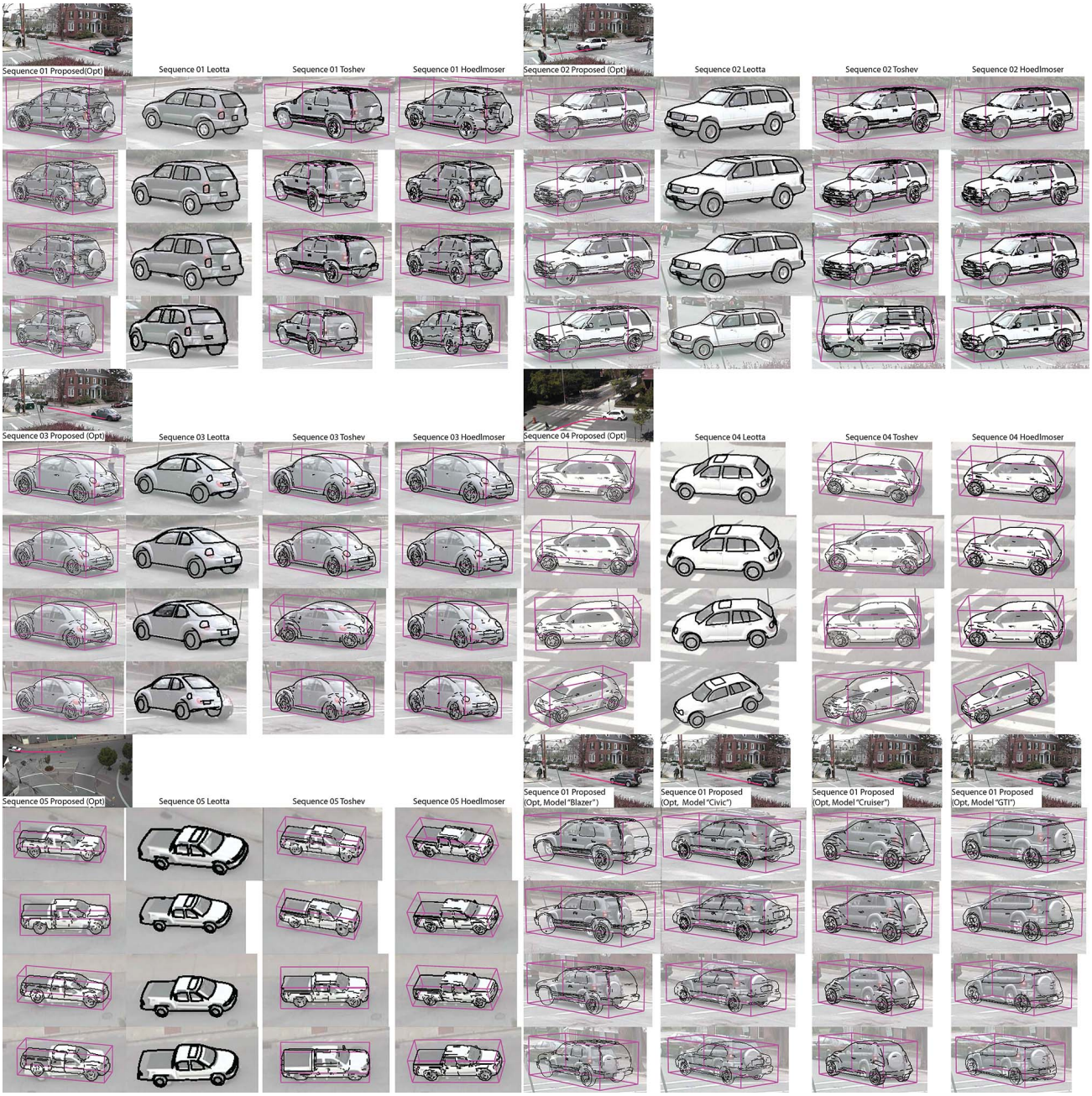


Figure 10. Classification and pose estimation results. One sequence is represented by four columns where each column shows results using different methods. The upper leftmost image of each sequence shows its first frame and our optimized projected track. The last four columns on the lower right show a variety of wrong model projected onto the image plane. See text for details.

[20] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *ICRA*, 2009. 5

[21] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, 2011. 2, 3

[22] M. Sun, P. Kohli, and J. Shotton. Conditional regression forests for human pose estimation. In *CVPR*, 2012. 2

[23] A. Toshev, A. Makadia, and K. Daniilidis. Shape-based object recognition in videos using 3d synthetic object models. In *CVPR*, 2009. 1, 2, 4, 5, 6

[24] M. Villamizar, H. Grabner, F. Moreno-Noguer, J. Andrade-Cetto, L. V. Gool, and A. Sanfeliu. Efficient 3d object detection using multiple pose-specific classifiers. In *BMVC*, pages 20.1–20.10, 2011. 2