Real-Time Pose Estimation Piggybacked on Object Detection

Roman Juránek, Adam Herout, Markéta Dubská, Pavel Zemčík
Graph@FIT, Brno University of Technology
Brno, Czech Republic
ijuranek,herout,idubska,zemcik@fit.vutbr.cz

Abstract

We present an object detector coupled with pose estimation directly in a single compact and simple model, where the detector shares extracted image features with the pose estimator. The output of the classification of each candidate window consists of both object score and likelihood map of poses. This extension introduces negligible overhead during detection so that the detector is still capable of real time operation. We evaluated the proposed approach on the problem of vehicle detection. We used existing datasets with viewpoint/pose annotation (WCVP, 3D objects, KITTI). Besides that, we collected a new traffic surveillance dataset COD20k which fills certain gaps of the existing datasets and we make it public. The experimental results show that the proposed approach is comparable with state-of-the-art approaches in terms of accuracy, but it is considerably faster – easily operating in real time (Matlab with C++ code). The source codes and the collected COD20k dataset are made public along with the paper.

1. Introduction

Reliable detection of vehicles is a crucial part of traffic monitoring systems. The cost of cameras, processing power, and communication bandwidth are decreasing and the traffic monitoring systems are not composed of dozens of cameras anymore. Instead, the number of cameras involved in traffic monitoring is rapidly increasing. Therefore, each camera receives less and less expert human input such as manual calibration and the systems are expected to work invariably for various views on the scene.

Interestingly enough, Benenson et al. [2] recently showed that in specific detection tasks (pedestrians, faces, traffic signs), less is more and good old methods are capable of producing best results and beat sophisticated systems with elaborated models. We were curious, if for real-time detection+pose estimation task we need separated detectors and regressors. The questions we asked and try to answer are *Does omni-directional tree-based detector internal clustering of objects depend on their viewpoint? Is this separation usable for pose estimation?* As we show in this paper, the answers to both questions are *Yes* and with special impurity function in the tree splits, the pose regression can be improved without significant detector degradation (Fig. 1).

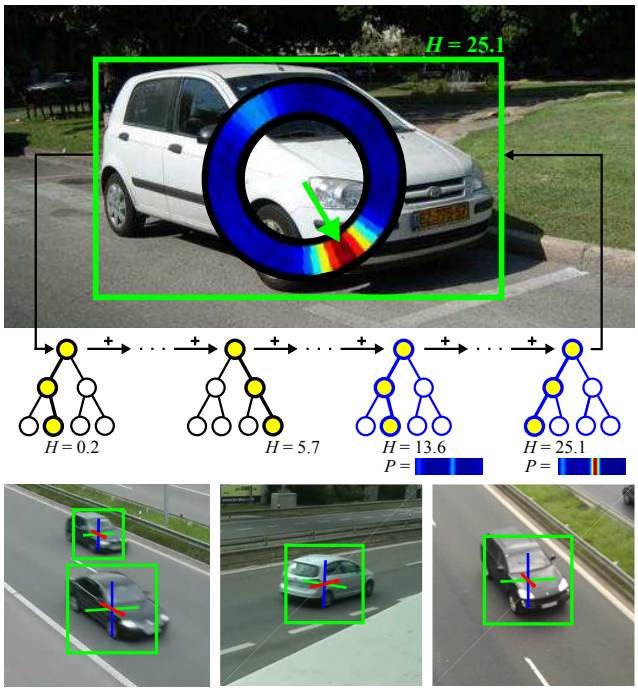


Figure 1: Our algorithm estimates pose along with object detection. Later stages of the soft cascade estimate the pose by accumulating regression maps, without extracting any information from the image.

Methods for multi-view detection and pose estimation can be divided into two main groups. The first group of methods requires a 3D model, whether as a CAD model [18, 29, 25, 34, 37, 20, 26], point cloud [13], or in the form of a wireframe [36]. Methods from the second group can solve

the task using just 2D information [23, 19, 15, 12, 27, 26]. Many methods – both those using 2D and 3D information – are part-based detectors. Algorithms with state-of-the-art results are using Deformable Part Models [19, 26], however, they are computationally very expensive (several seconds per image). Other part based detectors use Random Forests [37] or other techniques [18, 34]. Some methods treat pose estimation as a continuous problem [27, 26], other authors use several view-dependent detectors [20, 15, 29, 19, 28, 14] or 1-vs-all classification [12]. Some of these works (that we compare with) will be described in more detail later in the text.

In our work, we are following the recommendations of Benenson [2] by sticking with the existing and well-performing simple detector – Boosted soft cascade [31, 6] and modify its training and testing scheme to provide pose regression without compromising their speed performance. We are constructing a single omnidirectional detector – "one detector to see them all" – in contrast to the majority of existing approaches which tend to train multiple detectors/regressors for individual pose classes [20, 15, 29, 19, 28, 14] (this includes the recent work of He et al. [15] who infer continuous pose, but use multiple regressors for different view classes). The weak classifiers in the cascade are decision trees using channel features as in [6]. As these trees are trained, we propose to collect statistics of viewpoints (or other features to be estimated) in the leaves. After a successful detection, the trained information is used in order to estimate the object's pose by summing pre-computed data of a subset of the leaves. This approach allows that in terms of extracting information from the image, the pose estimation is completely costless. The speed of the detector (meant for real-time operation) is therefore not compromised and the technology of the whole detection/estimation system is very simple – and thus suitable for industrial and embedded use.

Our method is in its nature similar to Cascaded Pose Regression (CPR) [7, 4]. We also accumulate evidence provided by weak regressors. The important difference is that we regress a likelihood map of poses, not the pose itself, and we do not extract any new information from the image, as we rely on the information extracted for the detector. CPR has been tuned to regress poses of objects after their detection; we, on the other hand, focus on estimation of the full 3D orientation of objects during their detection.

The main contributions presented in this paper are the following:
I) We are showing that an object detector inadvertently works with information relevant to the viewpoint from which the object is observed. Even an unmodified detector can be used for pose regression. Besides, we are showing that a slight modification of the training criteria (selection of split functions in the decision tree) can considerably improve the pose estimation results, while keeping the detection performance. These observations lead to more efficient vehicle pose estimation with object detection and may stimulate further investigation of the information processed by existing detectors.
II) We introduce a new dataset COD20k of cars observed from various viewpoints (Section 3). Our new dataset covers diverse viewpoints and contains 20,000 original car instances with annotated 2D bounding box, viewpoint vector, and relative position on the projection plane.
III) We propose algorithmic modifications to the ACF detector [6], which lead to considerably better detection performance in the domain of vehicle detection. Our algorithms are made public in the form of their Matlab source code.

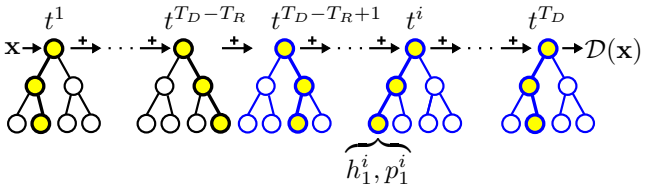


Figure 2: Our detector accumulates classification response H^t for a sample $\mathbf{x}$. Last T_R trees, marked by blue color, additionally accumulate pose estimation response P^t. If the sample is classified as positive, the object pose can be retrieved from $\mathcal{P}(\mathbf{x})$.

2. How Object Detectors Also Predict Pose

Our baseline detection model is a scanning window-based soft cascade of boosted trees similar to Aggregated Channel Features [6] (ACF). In principle, this solution is based on the AdaBoost Cascade by Viola and Jones [30]. The detector sweeps over positions and scales of the input image and classifies each window. Positive detections are grouped by a non maxima suppression algorithm in order to get final detections. We extend this model with boosted multi-label regression in order to determine object pose during detector evaluation, at virtually zero price.

Detector is a sequence of decision trees t^i of depth d. The total length of the detector is T_D (Figure 2). Interior nodes contain a binary split function routing samples to left or right, and each leaf l contains classification response h_l^i and pose regression map p_l^i. Regression maps are stored only in the last T_R trees and they contain all zeros for the trees at the beginning of the detector. The detector also stores a matrix $\mathbf{V} \in \mathbb{R}^{3\times K}$, $\mathbf{V} = [V_1, V_2, \ldots, V_K]$, where each V_i is a unit column vector representing a viewpoint and corresponding to one bin in the regression map. Viewpoints V_i are chosen arbitrarily and depend on the task. For example, if we want to predict one out of 64 view angles around

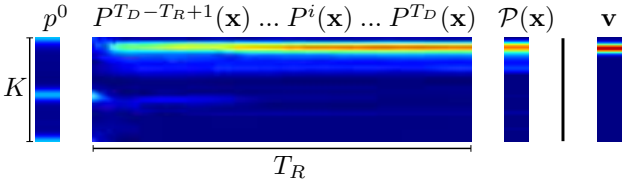


Figure 3: Pose estimation on a sample begins with p^0 (left) and proceeds with accumulation of T_R regression maps $p^i(\mathbf{x})$ (center), resulting in predicted pose vector $\mathcal{P}(\mathbf{x})$. True pose vector $\mathbf{v}$ is shown on the right side for comparison.

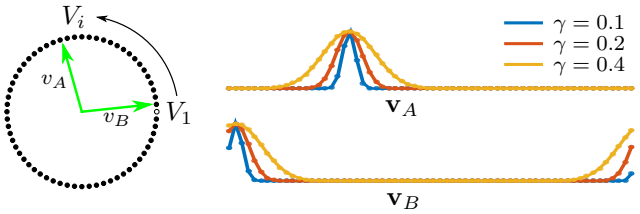


Figure 4: Examples of pose vectors $\mathbf{v}$. Value of i-th item of $\mathbf{v}$ corresponds to the similarity of the viewpoint vector v to V_i, Equation (2).

the object, $K = 64$ and viewpoint vectors V_i are uniformly distributed across the azimuth.

Given a sample $\mathbf{x}$, each tree in the detector responds with $t^i(\mathbf{x}) = (h^i(\mathbf{x}), p^i(\mathbf{x}))$, where $h^i(\mathbf{x}) = h^i_l$ and $p^i(\mathbf{x}) = p^i_l$ and l is the leaf where sample $\mathbf{x}$ is routed to. The whole detector responds with a tuple $\mathcal{D}(\mathbf{x}) = (\mathcal{H}(\mathbf{x}), \mathcal{P}(\mathbf{x})) = (H^{T_D}(\mathbf{x}), P^{T_D}(\mathbf{x}))$ by accumulating partial responses from all the trees (1).

$$\begin{aligned} H^t(\mathbf{x}) &= \textstyle\sum_{i=1}^{t} h^i(\mathbf{x}) \\ P^t(\mathbf{x}) &= p^0 + \textstyle\sum_{i=1}^{t} p^i(\mathbf{x}) \end{aligned} \tag{1}$$

Evaluation of h^i and p^i is done simultaneously as the values are stored in leafs of a common tree t^i. This means that the model solves sample classification and object pose estimation simultaneously.

Our detector is a soft-cascade and therefore each detector stage is assigned a threshold θ^i which terminates detector evaluation if $H^i(\mathbf{x}) < \theta^i$. This threshold can be calculated during training or set up by using a validation set. Early stages reject a vast majority of background samples (non-objects) and for a sample to be recognized as the object of interest, it must proceed through the whole cascade. For this reason, the regression maps are stored only for later stages of the cascade. Figure 6c shows how the number of regression stages T_R (always placed at the end of the cascade) influences the pose estimation error. Otherwise, the algorithm follows standards of scanning window detectors [30, 6]: the scanning windows are scanning a pyramid representation of the image with 8 steps per scale octave. For each scale in the pyramid, it extracts channel features: pixel intensity, gradient magnitude and 6 bins of gradient orientations. Detailed analysis of feature channels pyramid can be found in [35]. Candidate locations that passed the whole cascade are processed by a non-maxima suppression algorithm, keeping only the location with maximal responses.

The only computations added above the standard detector evaluation is the accumulation of $\mathcal{P}(\mathbf{x})$ during the final stages of the detector. This is done sparsely due to soft-cascade nature of the detector and thus the effect on computation time is negligible.

Each item of the final pose response $\mathcal{P}(\mathbf{x})$ corresponds to a column in the matrix $\mathbf{V}$. The predicted view is V_j where j is the index of maximal item from $\mathcal{P}(\mathbf{x})$, or the values can be interpolated and sub-pixel search for continuous pose estimation applied. Figure 3 shows the evolution of $P^t(\mathbf{x})$ for a random sample.

Detector training starts with a set of labeled training samples $X = \{(\mathbf{x}, y, v)_j\}$, where $\mathbf{x} \in \mathbb{R}^{M \times N \times C}$ is a C-channel image feature matrix, $y \in \{-1, +1\}$ is class label, and $v \in \mathbb{R}^3$ is an object viewpoint represented as a unit vector $v = [x, y, z]'$ ($\mathbf{a}'$ stands for transposed vector, we reserve $\mathbf{a}^T$ for indexing). Viewpoints v_j of samples are transformed to pose vectors $\mathbf{v}_j \in \mathbb{R}^K$, where each i-th item represents the similarity between v and V_i. In our experiments, we use Gaussian function (2) as the similarity measure, where $\mathbf{V}$ is the matrix with viewpoints (see above), and γ is a parameter controlling the size of Gaussian smoothing, illustrated in Figure 4.

$$\mathbf{v}_j = \exp\left(-\frac{1}{2}\left(\frac{acos(v'_j \cdot \mathbf{V})}{\gamma}\right)^2\right) \tag{2}$$

We found out that the pose estimation error is not sensitive to setting the value γ when set to reasonable values, see Figure 6b. In our experiments, we use $\gamma = 0.2$. When view vector v is not defined (e.g. negative training samples), the similarity pose vector is set to $\mathbf{v} = \mathbf{0}$.

Detector $\mathcal{D}$ is then a sequence of T_D decision trees trained with the Real AdaBoost algorithm [30]. During the training, the dataset is bootstrapped from a large pool of negative samples, low scoring samples are dropped and replaced by hard examples. Once all the stages are found, each of them is assigned with an early termination threshold calculated so that the detector produces a low number of false alarms while keeping a (pre-configured) high detection rate.

Each tree is trained according to the Random Forest framework [3]. Training of a tree t starts with the root node which is assigned training samples X_n. Many random splits which divide X_n to subsets X^L_n and X^R_n are generated. The split which minimizes the error measure (3) is selected for the node. Child nodes are trained recursively

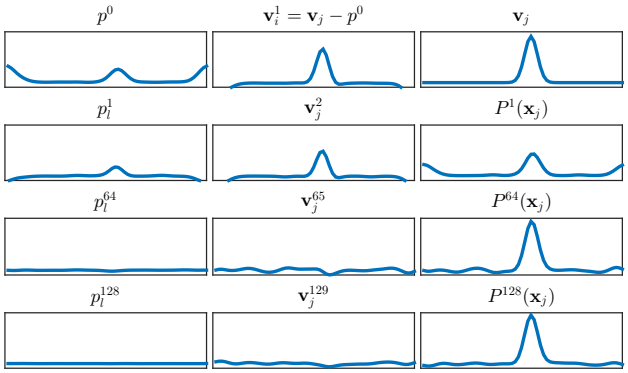


Figure 5: Example of pose estimation of a sample $\mathbf{x}_j$. Left column shows initialization value p^0 and contributions of leafs. Central column shows evolution of pseudo-residual $\mathbf{v}_j$ during training. And right column shows the true values of pose vector $\mathbf{v}_j$ (top), and evolution of response $P(\mathbf{x}_j)$ (bottom). See how the values of $\mathbf{v}_j^i$ are minimized as the predicted P gets closer to $\mathbf{v}_j$.

on the split training sets. The training stops when a maximal depth is reached or the number of training samples falls below a threshold. Then, the prediction model in leaf nodes is trained.

$$I(X_n) = \sum_{i \in \{L,R\}} \frac{X_n^i}{X_n} E(X_n^i) \tag{3}$$

For detector training, we use *classification* criterion $E_c(X) = 2\sqrt{W^+W^-}$, where W^+ and W^- are sums of normalized sample weights with $y = +1$, $y = -1$, respectively, in the set X. The weight of a sample is defined according to AdaBoost as $w_j = \exp(-y_j H^t(\mathbf{x}_j))$. Additionally, we define a *regression* criterion

$$E_r(X) = \sum_{\mathbf{v}_j \in X} (\mathbf{v}_j - \bar{\mathbf{v}})^2, \quad \bar{\mathbf{v}} = \frac{1}{|X|} \sum_{\mathbf{v}_j \in X} \mathbf{v}_j, \tag{4}$$

which is used for the last T_R stages of the detector. This minimizes the **pose residual error** of pose vectors $\mathbf{v}$ in split nodes, enforcing better regression and pose estimation. We demonstrate the effect of T_R in Figure 6c. We also explored the linear combination of E_r and E_c, evaluated in Figure 7. It shows that both E_r and E_c give good results in terms of regression. However, E_r converges more rapidly and it yields low errors with just 16 regressors.

Training leaf nodes. Each leaf l of tree t^i contains a tuple with detector response and pose regression map (h_l^i, p_l^i). The value of h is trained according to standard AdaBoost formula $h = \frac{1}{2} \log \frac{W^+}{W^-}$ from normalized sample weights reaching the node.

Training of p is done by the Gradient Boosting Trees [10] approach (Figure 5). Let X_p be a set of all positive training samples (i.e. with $y = +1$). We define the initialization map (5) simply as the mean over all pose vectors in X_p.

$$p^0 = \frac{1}{|X_p|} \sum_{\mathbf{v}_j \in X_p} \mathbf{v}_j. \tag{5}$$

For each stage i, we keep pseudo-residual vectors $\mathbf{v}_j^i$. Initialization values are set to $\mathbf{v}_j^1 = \mathbf{v}_j - p^0$. The pose regression map in leaf l is trained from samples X_l reaching the leaf as

$$p_l^i = \alpha \frac{1}{|X_l|} \sum_{\mathbf{v}_j \in X_l} \mathbf{v}_j^i, \tag{6}$$

where α is a constant learning rate (we use $\alpha = 0.5$). Finally, pseudo-residuals are updated before the next stage with

$$\mathbf{v}_j^{i+1} = \mathbf{v}_j^i - p_l^i, \quad \mathbf{v}_j \in X_l. \tag{7}$$

This way, the later stages correct errors (captured by pseudo-residual values) caused by prior stages.

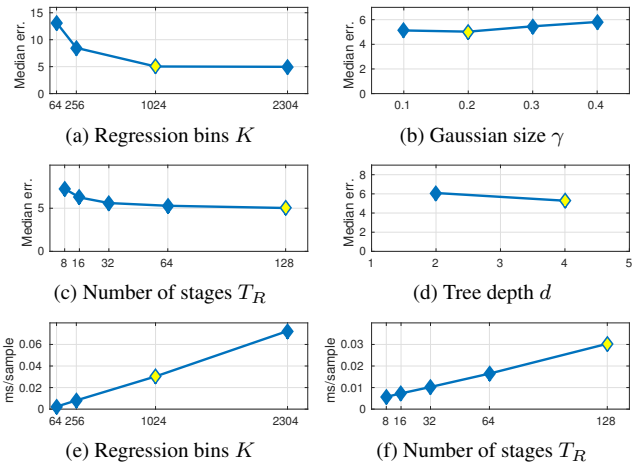


Figure 6: Setting of pose estimation parameters (default in yellow) on COD20K dataset. We measured median error of pose estimation (azimuth and elevation) for: (a) different number of pose bins K, (b) gaussian smoothing γ, (c) number of stages T_R, and (d) depth of trees d. (e) and (f) shows the linear complexity of pose estimation per sample. Results on other datasets show similar trends.

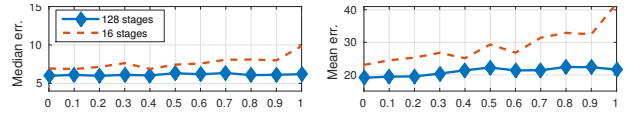


Figure 7: Median (left) and mean (right) errors of regression when using linear combination of E_r and E_c as a split selection criterion (λ is the mixing factor). $\lambda = 0$ corresponds to pure E_r and $\lambda = 1$ to E_c. For 128 regressors, the errors are approximately equivalent with the mean error slightly in favor of E_r.

3. COD20k: New Omnidirectional Car Dataset

There is a number of datasets that were previously used for training car detectors. The oldest ones are perhaps MIT Cars [24] and UIUC [1]. These datasets contain a few hundreds of low resolution cars captured from rear/front (MIT) and side (UIUC) views. A popular dataset PASCAL VOC [9] contains several thousand cars (and other object categories) captured from street level without 3D annotation, which is available in PASCAL 3D+ [33]. Other available car datasets are TME dataset [5], TU Graz [22], and EPFL cross [16]. Datasets that contain viewpoint annotations are WCVP [13], 3D objects [28], KITTI [11], and EPFL [23]. Most of these datasets contain only a small number of instances for training scanning window detectors, or lack reliable viewpoint annotations.

We propose a new dataset COD20k[1] containing total of 19,126 car instances in 9,530 training images, and 4,457 cars in 1,127 testing images. Cars are captured from diverse views (in highway surveillance scenario), see Figure 8. Each car instance is annotated with a 2D bounding box and a viewpoint vector.

3.1. Dataset Collection and its Properties

We used a set of videos provided by [8]. Each video was captured by a static camera and automatically calibrated to get camera position and rotation relative to the road and its focal length. From each video, we extracted images 40 video frames apart and generated annotations for moving vehicles (using background subtraction). Frames from the beginning of videos are part of the training data, frames from the end are part of testing data. All annotations were manually filtered and corrected to get tight bounding boxes for all cars. For each bounding box, we calculated the 3D viewpoint vector (the difference vector between camera position and car position) using known camera calibration. We also store the normalized car position in the image plane relative to the camera principal point.

Due to the surveillance character of our data, the sizes of annotated cars are quite small (Figure 9a), and the viewpoints are concentrated around rear/front views (Figures 9b, 9c). Mean width and height are 63 resp. 46 pixels, and median width and height are 52 resp. 38 pixels. In total, there are 2,868 car instances whose width is greater than 100 pixels.

4. Experimental Results

We experimented on the COD20k, KITTI, WCVP, and 3D Objects datasets – see Figure 13 for examples of detections. Unless specified otherwise, the settings for experiments were the following. The feature channels included

[1]Data and code can be downloaded from www.fit.vutbr.cz/research/groups/graph/PoseEstimation

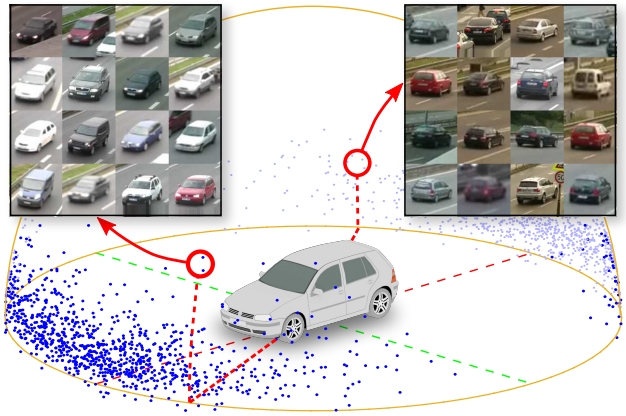
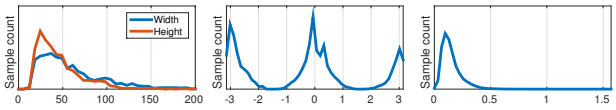


Figure 8: Distribution of viewpoints in the COD20k dataset, and examples of cars captured from two distinct views. The distribution of the vehicles viewpoints is not uniform, because of surveillance characteristics of the videos.



(a) Size (px) (b) Azimuth (rad) (c) Elevation (rad)

Figure 9: Distribution of annotated sample sizes and viewpoint directions in the dataset.

image intensity, gradient magnitude, 6 gradient orientation channels ($C = 8$). The cascades are $T_D = 1024$ stages long, the depth of trees is $d = 2$, except for the 'regression stages', where $d = 4$. The experiments report results for two detectors: **Our A** is only trained with the E_c criterion, meaning that it is a more-or-less standard ACF detector, with the later stages equipped with the regression maps, but otherwise trained normally for detection. **Our B** is an improved detector with E_r criterion used for the last T_R stages ($T_R = 128$ in all experiments).

4.1. Results on the COD20k Dataset

For all experiments on this dataset we used $K = 1024$ with vector V distributed uniformly across azimuth (32 bins) and elevation (32 bins). Detected objects were matched to the ground truth using Pascal criteria. Pose estimation error was evaluated for all true positive detections as the angular difference of the estimated viewpoint vector from the ground truth vector.

We were curious about the answers to the following questions: *i)* Can we add regression maps to an already trained detector? *ii)* Does the regression criterion E_r really improve pose estimation performance, and how does it affect detection? *iii)* Is the 'regression tail' trained with the E_r criterion a mere pose estimator, or does it improve

detection as well?

Figure 10 gives the answers to these questions. The *baseline* detector (magenta) yields reasonable regression performance, although it is a standard ACF detector without any modifications, only equipped with regression maps in the leaves. The answer to question *i)* is therefore yes, the simple piggybacking is possible and useful. The measurements also answer positively question *ii)* by showing that the E_r regression criterion improves pose estimation, while sticking to E_c maximizes the detection performance, resulting in slightly worse pose estimation. Finally, the answer to question *iii)* is positive for the proposed piggybacking: the 'regression tail' still improves the detection rate considerably (yellow line clearly outperforms blue dashed line in detection).

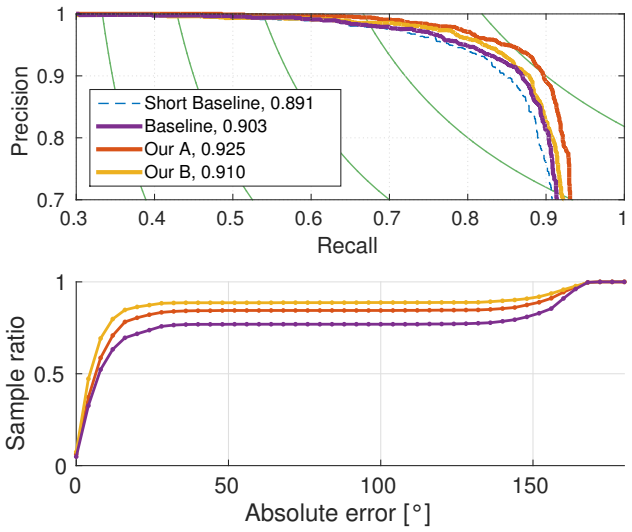


Figure 10: Results on COD20k dataset. **top:** PRC of the detectors with average precision, **bottom:** cumulative histogram of angular error. **blue dashed:** $T_D = 896$, $d = 2$ detector without the 'regression tail' – *short detector*. **magenta:** $T_D = 1024$, $d = 2$ – standard full length detector, with $T_R = 128$ last stages with regression maps – *baseline*. **red:** same as magenta; $d = 4$ for the last T_R stages – *Our A* detector. **yellow:** same as red; E_r criterion for the last T_R stages – *Our B* detector.

4.2. Results on the WCVP Dataset

Wiezman Cars dataset [13] contains 1,530 images of 22 distinct cars, each taken from multiple (mostly eye-level) viewpoints around the car. The data are separated to 3 fixed folds, each with approximately $^2/_3$ of the cars for training and the rest for testing.

We trained the two detectors *Our A* and *Our B* with $K = 32$, and with viewpoints V distributed uniformly across the azimuth. We follow the evaluation from previous works [13, 36, 20, 27] and report the median error in azimuth estimation (Table 1) and the histogram of errors (Figure 11).

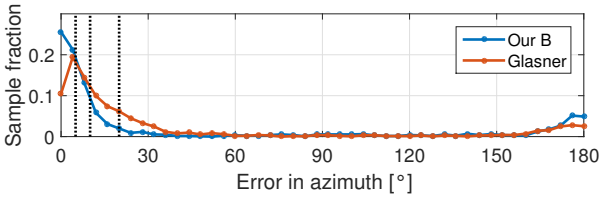


Figure 11: Histogram of the angular errors in azimuth estimation on WCVP dataset. Our detector exhibits more precise estimations compared to [13]. Dotted lines mark 5°, 10°, and 20° margins.

Method	[13]	[36]	[20]	[27]	**Our A**	**Our B**
Med. Error [°]	12.3	23.2	7.6/8.4	7	9.1	8.6
AP [%]	-	-	-	79	87	87

Table 1: Median error in azimuth estimation and average precision (AP) of detection on WCVP dataset. Methods [13, 36, 20] used 3D models.

Our method is comparable to state of the art methods [20, 27]. The average speed of detection and pose estimation was 83 ms on this dataset (single core, 3 GHz). Movshovitz et al. [20] use a large set of 40 view-tuned correlation filters trained from 3D CAD models. We beat the other two methods. Glasner [13] create 3D point cloud from a set of 2D images and uses it for training a 6D voting model (for position and viewpoint). Yoruk [36] created a wireframe model from 2D blueprints and fit the model to the input image so that the edges of the model match the image. Redondo-Cabrera et al. [27] use probabilistic voting similar to Hough Forests in the position-pose space to detect cars.

4.3. Results on the 3D Objects Dataset

In 3D Objects [28], there are 8 object categories, each with 10 individual object instances under 8 viewing angles, 3 heights and 3 scales. The azimuth estimation is actually an 8-class classification problem (in our notation $K = 8$). We report results for the 'car' category which contains 480 images.

We follow the evaluation strategy of [18, 29] and use 4-fold cross validation. In each fold, we randomly selected 7 instances for training and the rest for testing. We report average precision (AP) of detection and Mean precision of pose estimation (MPPE), Table 2. This table also compares the processing speed of the discussed algorithms – row RT stands for real-time. Most of the works do not report processing speed, their assessment is therefore estimated from similar works and the algorithm's principle. The question

2D	[26]	[25]	[19]	[12]	[27]	[15]	**Our A**	**Our B**
AP	100	–	96	88	89	98	95	95
MPPE	98	86	89	98	90	88	82	88
RT	✗	✗	✗	✗	✗?	✗?	✓	✓
3D	[18]	[29]	[13]	[34]	[36]	[37]	[26]	[26]
AP	77	81	99	98	93	97	99	100
MPPE	70	81	85	93	73	97	98	98
RT	✗?	✗?	✗	✗	✗	✗	✗	✗

Table 2: Results for 'car' category in 3D Objects dataset. Values of AP and MPPE are given in %. RT stands for ability to process images in real time (within 1s). Our method is comparable to other results and is fastest of them all.

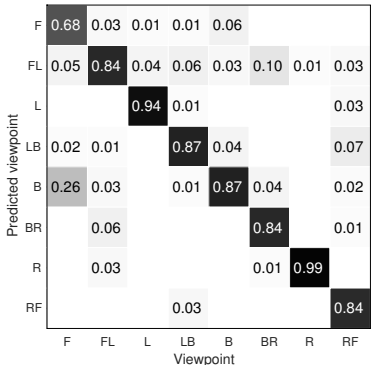


Figure 12: Confusion matrix of view angle estimation on "car" category from 3D Objects dataset for detector *Our B*.

marks in this row indicate our uncertainty about the statement. We also show the confusion matrix for viewpoint classification, Figure 12.

The most accurate state-of-the-art work of Pepik et al. [26] extends the Deformable Part Model with 3D information to capture the structure of detected objects. This kind of model usually takes several seconds per image on contemporary hardware.

To our knowledge, we propose the first reportedly real-time solution – on *3D Objects* we achieve 48 ms average time per image (single thread, 3 GHz, i5) to the problem of coupled detection and pose estimation for vehicles. At the same time, our detection rates and pose estimation accuracy beats several of the reference works. Our solution also does not rely on any outside knowledge (e.g. 3D model, 2D blueprints) and is only trained on the given dataset. It should be noted that the amount of samples in the dataset (480 in total) is not perfectly sufficient for training a method such as ours. This is one of the reasons for collecting the new COD20k dataset which provides sufficient training data.

4.4. Results on the KITTI Dataset

The KITTI dataset [11] contains 7,481 training images and 7,518 test images taken by an on-board camera from a vehicle in urban environment. Images are high resolution 1242×375 pixels. Annotations include passenger cars, vans, trucks, cyclists, and people, and it additionally specifies information about object occlusion, truncation, viewpoint and 3D bounding box of objects. We trained only the *Our B* version of the detector: $K = 64$, E_r criterion, vectors V distributed uniformly across azimuth.

	AP [%]		AOS [%]		TIME [s] (Cores)
method	easy	moder.	easy	moder.	
[6]	55.9	54.7	-	-	$<$ 0.2 s (1)
[32]	87.5	75.8	86.9	74.6	40 s (8)
[21]	84.1	75.7	80.9	64.9	0.7 s (6)
[26]	74.9	64.7	72.3	61.8	8 s (1)
[17]	84.4	71.9	43.8	38.2	3 s (4)
Our B	61.5	52.9	58.6	50.7	$<$ 1.0 s (1)

Table 3: AP and AOS (average orientation similarity) for KITTI dataset. Our method is comparable to ACF detector (reported by other researches), and our pose estimation beats the work of Li et al. [17].

Evaluation on KITTI requires precise detection of object bounding box (overlap with ground truth > 0.7). We use only one rigid model with fixed aspect ratio to detect cars with arbitrary orientation, and for this reason we correct the aspect ratio of our detections according to the estimated azimuth with a correction function trained on the training data.

Table 3 reports the results generated by the automated testing procedure available online, and compares them to alternatives (see KITTI web page for more results). It is not a surprise that our method is comparable to ACF which was reported by other researchers. At the same time, we beat the work of Li et al. [17] in pose estimation performance. We report detection time for *moderate* setting which require the detection of objects as small as 25 pixels in height, and therefore we need to scan a large number of candidate positions. The average processing time for the *easy* setting was 200 ms (single thread, i5, 3GHz).

5. Conclusions

We propose genuinely interleaved vehicle detection and pose estimation. Our solution is simple – slightly modifying the ACF detector – and very fast: the pose estimation overhead is negligible. It is meant and suitable for traffic surveillance systems operating in real time. Our algorithm seems to be the first real-time solution of vehicle viewpoint/pose estimation with results comparable to the state-of-the-art works relying on complex external knowledge (3D CAD models, specially rendered datasets, etc.), and are far from being usable in surveillance (reported times up to 30s per frame).

We are showing that a part of the detection cascade can be used for both purposes: detection and pose estimation, at the same time. Our experiments verify that the regression

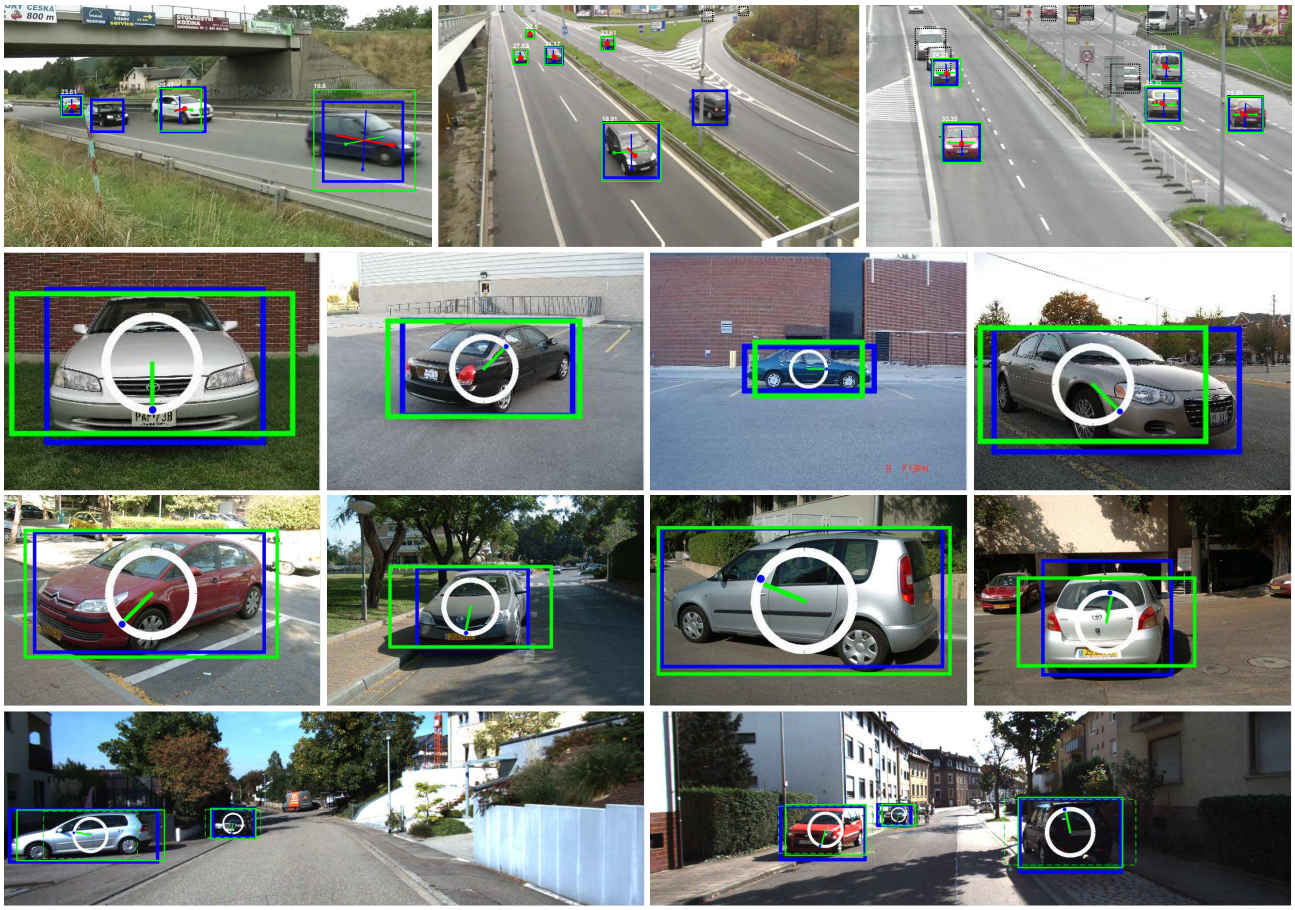


Figure 13: Examples of successful detections and pose estimations. In all images, ground truth is marked by blue rectangles, and our detections by green ones. **Row 1:** COD20k. Pose estimation in both, azimuth and elevation. We show a visualization of the predicted orthogonal basis. **Row 2:** 3D Objects. **Row 2:** WCVP dataset. **Row 4:** KITTI dataset, dashed line – original detection with fixed aspect ratio, solid line – adjusted detection.

Figure 14: Examples of wrong detections or pose estimations. These cases include: missed detections due to wrongly adjusted bounding box caused by bad viewpoint estimation; missed detections of objects outside image boundary; and bad viewpoint caused by opposite view similarity.

part is not "run after detection" but contributes to detection performance considerably and a designer of the surveillance system can arbitrarily control the trade-off between these two factors and also the factor of speed (length of the cascade, depth of the decision trees involved). In order to support the research in omnidirectional vehicle detection, we collected a dataset of 20k samples with annotated viewpoint direction. This dataset, along with the source codes of the detector and its training are made public for further development and evaluation.

Acknowledgements This research was supported by the CEZMSMT project IT4I – CZ 1.05/1.1.00/02.0070, and by the project Visual Computing Competence Center – V3C, TE01020415.

References

[1] S. Agarwal, A. Awan, , and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE PAMI*, 26(11):1475–1490, Nov. 2004.

[2] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV*, 2014.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] X. P. Burgos-Artizzu, P. Perona, and P. Dollár. Robust face landmark estimation under occlusion. In *ICCV*, pages 1513–1520. IEEE, 2013.

[5] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas. A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera. In *ITSC*, pages 975–982, Sep. 2012.

[6] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE PAMI*, 2014.

[7] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *IEEE CVPR*, pages 1078–1085. IEEE, 2010.

[8] M. Dubská, J. Sochor, and A. Herout. Automatic camera calibration for traffic understanding. In *BMVC*, 2014.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010.

[10] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE CVPR*, 2012.

[12] A. Ghodrati, M. Pedersoli, and T. Tuytelaars. Is 2D information enough for viewpoint estimation. In *BMVC*, 2014.

[13] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and continuous pose estimation. *Image and Vision Comp.*, 2012.

[14] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *ECCV*. Springer, 2010.

[15] K. He, L. Sigal, and S. Sclaro. Parameterizing object detectors in the continuous pose space. In *ECCV*, 2014.

[16] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3D scene analysis from a moving vehicle. In *CVPR*, pages 1–8, June 2007.

[17] B. Li, T. Wu, and S.-C. Zhu. Integrating context and occlusion for car detection by hierarchical and-or model. In *ECCV*, 2014.

[18] J. Liebelt and C. Schmid. Multi-view object class detection with a 3D geometric model. In *CVPR*, June 2010.

[19] R. J. Lopez-Sastre, T. Tuytelaars, and S. Savarese. Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV Workshops*, pages 1052–1059. IEEE, 2011.

[20] Y. Movshovitz-Attias, Y. Sheikh, V. Naresh Boddeti, and Z. Wei. 3D pose-by-detection of vehicles via discriminatively reduced ensembles of correlation filters. In *BMVC*, 2014.

[21] E. Ohn-Bar and M. M. Trivedi. Fast and robust object detection using visual subcategories. In *CVPR Workshops*, pages 179–184. IEEE, 2014.

[22] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Generic object recognition with boosting. Technical Report TR-EMT-2004-01, EMT, TU Graz, Austria, 2004. Submitted to the IEEE Tran. PAMI.

[23] M. Özuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR*, pages 778–785, June 2009.

[24] C. Papageorgiou and T. Poggio. A trainable object detection system: Car detection in static images. Technical Report 1673, October 1999. (CBCL Memo 180).

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

[26] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Multi-view and 3D deformable part models. *IEEE PAMI*, 2015.

[27] C. Redondo-Cabrera, R. Lopez-Sastre, and T. Tuytelaars. All together now: Simultaneous detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting. In *BMVC*. BMVA Press, 2014.

[28] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *IEEE ICCV*, 2007.

[29] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3D CAD data. In *BMVC*, 2010.

[30] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 2004.

[31] J. Šochman and J. Matas. WaldBoost – learning for time constrained sequential detection. In *CVPR*, 2005.

[32] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3D voxel patterns for object category recognition. In *CVPR*, 2015.

[33] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *WACV*, 2014.

[34] Y. Xiang and S. Savarese. Estimating the aspect layout of object categories. In *CVPR*, pages 3410–3417. IEEE, 2012.

[35] B. Yang, J. Yan, Z. Lei, and S. Li. Aggregate channel features for multi-view face detection. In *IEEE IJCB*, 2014.

[36] E. Yoruk and R. Vidal. Efficient object localization and pose estimation with 3D wireframe models. In *ICCV Workshops*, pages 538–545. IEEE, 2013.

[37] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3D representations for object recognition and modeling. *IEEE PAMI*, 35(11):2608–2623, 2013.