Robust Automatic Monocular Vehicle Speed Estimation for Traffic Surveillance

Jerome Revaud Martin Humenberger
NAVER LABS Europe
firstname.lastname@naverlabs.com

Abstract

Even though CCTV cameras are widely deployed for traffic surveillance and have therefore the potential of becoming cheap automated sensors for traffic speed analysis, their large-scale usage toward this goal has not been reported yet. A key difficulty lies in fact in the camera calibration phase. Existing state-of-the-art methods perform the calibration using image processing or keypoint detection techniques that require high-quality video streams, yet typical CCTV footage is low-resolution and noisy. As a result, these methods largely fail in real-world conditions. In contrast, we propose two novel calibration techniques whose only inputs come from an off-the-shelf object detector. Both methods consider multiple detections jointly, leveraging the fact that cars have similar and well-known 3D shapes with normalized dimensions. The first one is based on minimizing an energy function corresponding to a 3D reprojection error, the second one instead learns from synthetic training data to predict the scene geometry directly. Noticing the lack of speed estimation benchmarks faithfully reflecting the actual quality of surveillance cameras, we introduce a novel dataset collected from public CCTV streams. Experimental results conducted on three diverse benchmarks demonstrate excellent speed estimation accuracy that could enable the wide use of CCTV cameras for traffic analysis, even in challenging conditions where state-of-the-art methods completely fail. Additional information can be found on our project web page: https://rebrand.ly/nle-cctv

1. Introduction

Being able to accurately measure the traffic speed on road networks is important for many applications like live intelligent transportation system and itinerary planning, traffic analysis [24, 40], and anomaly detection [18, 10]. This is normally achieved through dedicated hardware sensors like roadside radar sensors on highways, inductive loop detectors at intersections, GPS data collected from probe fleet, etc. [32, 23]. Nevertheless, such equipment is expensive and can hardly be deployed quickly and at a large scale.

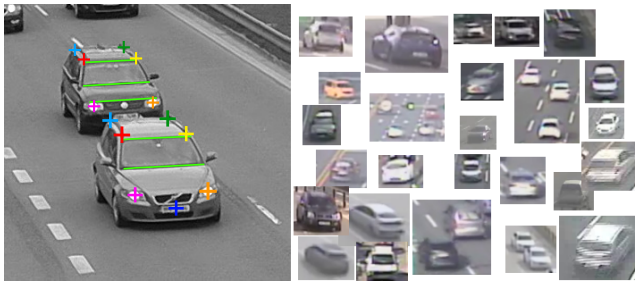

Figure 1. Cars in the BrnoCompSpeed [44] benchmark (left) and our CCTV dataset (right) displayed at their actual apparent size. Calibration methods [16, 43, 9, 5, 4] are based on image processing techniques (estimating angles of lines on moving cars, green lines) or keypoint localization (colored crosses) that are both highly impacted by image quality and resolution. Applying the same techniques on actual CCTV footage, where the resolution is low and quality often mediocre, turns out to be nearly impossible.

Traffic surveillance cameras, often (improperly) dubbed CCTVs, are already widely deployed for manual traffic monitoring. Since they are directly filming roads and vehicles, they provide a rich flow of video data that implicitly contains nearly all relevant traffic information. It has therefore been noted that CCTV cameras have the potential to be turned into traffic speed sensors at little cost [2, 32, 30, 18, 28, 23]. In this paper, we precisely focus on this particular problem, *i.e.* the estimation of vehicle speed from monocular videos captured via surveillance cameras. This is in fact a challenging problem. As a matter of fact, there is to the best of our knowledge no large-scale usage of CCTV cameras for automated traffic speed surveillance, despite the public availability of CCTV real-time video streams [23, 2, 30, 18].

One of the explanation lies in the fact that state-of-the-art approaches often assume that cameras are already calibrated [42, 35], which is rarely the case for CCTVs. Still, recent methods were developed to automatically calibrate cameras by looking for specific clues on moving vehicles [17, 43, 9, 6, 6, 5]. For instance, one solution is to look for straight edges perpendicular to the car motion and parallel to the ground (green lines in Fig. 1), as the

Figure 2. Example of frames from typical CCTV cameras. Resolution and quality are typically low and cars can appear quite small.

apparent angle between perpendicular edges is related to the camera focal [16, 43]. Another option is to localize certain keypoints (colored crosses in Fig. 1) which, combined with the knowledge of the corresponding 3D car model, establish 2D-3D correspondences that lead to the camera 3D pose [9, 6, 6, 5]. These techniques, however, are complex and highly sensitive to noise, hence high-quality footage with high-resolution is required in practice. The BrnoCompSpeed dataset [44], which serves as benchmark for all these methods, incidentally comprises only high-resolution videos (2M Pixels) captured with high-quality optics that are unlikely to be encountered in field conditions. CCTV footage typically consists of low-quality, low-resolution and blurry/noisy video clips with tiny-looking cars, as exemplified in Fig. 1 and 2.

In this paper, we take a radically different approach for estimating the speed of vehicles *solely based on vehicle detections* that are output by an off-the-shelf object detector. **As a first contribution**, we propose a novel calibration method based on minimizing a 3D reprojection error that leverages general assumptions about the 3D shape and dimension of cars. **As a second contribution**, we propose a trainable version of the first method that instead *learns* to predict the scene geometry directly. Both approaches can handle non-straight roads and recover the full camera calibration in order to calculate vehicle speeds. Finally, **as a third contribution**, we introduce a novel dataset collected from public CCTV video streams and annotated with GPS tracks. Our experiments demonstrate excellent performance on synthetic and real data, even in challenging conditions.

2. Related work

In this section, we present prior art on vehicle speed estimation, camera calibration, and 3D vehicle pose estimation. We restrict our review to the vision-based methods for the sake of brevity.

Vehicle speed (velocity) estimation. Measuring the speed of vehicles purely from visual input, i.e. without using dedicated physical sensors, has been considered actively since at least two decades [42, 19, 21] (see [34] for a comprehensive survey). The vast majority of approaches, including modern ones [17, 18, 22, 28, 33, 35, 46, 23, 6, 5], can be formulated as a 3-step pipeline consisting of (1) detecting and (2) tracking vehicles, followed by (3) converting displacements from pixels to meters. Earlier works achieved vehicle detection via handcrafted methods such as background subtraction [15, 19, 21]. Nowadays, object detectors based on deep networks (*e.g.* Faster-RCNN [41]), and possibly fine-tuned to CCTVs conditions [27], are preferred due to their robustness and superior performance [18, 43, 28, 46]. Temporally connecting these detections in order to form vehicle tracks can then be performed either heuristicly (*e.g.* Kalman filter [8], [17, 46]) or with learned models [28]. The last step involves to convert each track, *i.e.* the pixel coordinates of a given vehicle along time, to meters in a world coordinate system. To the best of our knowledge, this step is systematically performed under the planar road assumption, which assumes that a homography directly maps image pixels to metric coordinates [27, 28, 33, 35, 44, 46, 43, 23, 30, 17, 18].

Camera calibration. Estimating the homography that relates the camera view with the road plane essentially boils down to *calibrating* the camera. This step is critical for the accuracy of vehicle speed measurements, as the speed estimation is highly sensitive to the calibration quality. The calibration consists of determining the intrinsic and extrinsic camera parameters describing, *e.g.*, the focal length and camera 3D pose (translation and rotation) [13]. We refer the reader to [44] for a more detailed review on camera calibration and now only include a brief description of existing methods. Calibration parameters are either manually entered by the user [42] or estimated automatically from CCTV footage. Manual methods typically require the user to annotate several points on the road with known coordinates [42, 44, 36, 35]. Automatic methods usually assume a straight planar road and rely on detecting vanishing points as an intersection of road markings (*i.e.* line paintings) [12, 11, 21] or from vehicle motion [13, 17, 16, 43]. Note that finding the vanishing points is not sufficient to fully calibrate the camera as it yields an homography up to an unknown scaling factor. This factor also needs to be estimated accurately since it affects all speeds linearly. Semi-automatic approaches therefore adopt some form of manual annotations where several known distances are typically carefully measured on the image by an operator [18, 28, 46]. FullACC and its improved version [16, 43] have been proposed to overcome these limitations and perform a fully-automatic calibration. After recovering the vanishing points using image-processing techniques, the scaling factor is estimated by fine-grained categorization of the vehicles (SUV, sedan, combi, ...), retrieving a corresponding 3D CAD

model and measuring the reprojection error w.r.t. the observed bounding box. While our first method also leverages box reprojection errors, it is able to recover *all* camera parameters (not just the scale), including the focal. Furthermore, our method does not require a specially trained network for fine-grained vehicle classification and works with any off-the-shelf object detector.

Vehicle 3D pose estimation. Another line of research to calibrate the camera aims to estimate the 3D pose (translation and rotation) of rigid objects with known dimensions. There exists a vast literature on 3D pose estimation and we now briefly review relevant works (for a recent review see *e.g.* [47]). Earlier works on 3D vehicle pose estimation are based on edge-guided non-rigid matching with 3D models [31] or cascades [25, 50]. With the advent of deep learning, deep network have successfully been applied to this task in various contexts (object pose estimation [50], car pose estimation for autonomous driving and for surveillance purposes [47, 45]). Existing works for estimating the 3D poses of vehicles either directly regress the global vehicle translation and rotation w.r.t. the camera [29, 38, 39, 45, 50] under constrained conditions (usually for autonomous driving), or predict the 2D positions of certain keypoints [48, 49, 3, 47, 4] and solve a PnP problem to jointly recover the homography and scale factor [9, 4, 5, 6]. Our second method regresses the homography Jacobian, which is a subset of the 3D vehicle pose (see Section 3.2), hence it is conceptually closer to this latter category of methods than to the calibration-based approaches. An important drawback of all these 3D pose estimation methods is that they require both a fine-grained categorization of vehicles and an accurate prediction of the 2D keypoint positions [9, 45, 4, 5, 6]. While this is feasible with high-resolution and high-quality videos, it becomes close to impossible with typical surveillance cameras, even for the human eye, as shown in Figure 1. In contrast, our method is able to predict the 3D pose of vehicles from weak clues and without categorizing vehicles nor detecting landmarks, but rather by accumulating evidences from multiple detections, making it broadly applicable without further requirements.

3. Automatic calibration method

We present in this section two different methods, yet based on similar insights, for automatically calibrating the camera given a set of vehicle detections. The first method consists in minimizing a 3D reprojection error given a rendering of the scene based on the calibration (Section 3.1). The second method learns to predict the outcome of this handcrafted and computationally expensive minimization using a transformer network (Section 3.2).

3.1. Reprojection-based method

We seek to minimize an objective function $\mathcal{E}(\mathcal{H}) \mapsto \mathbb{R}$ that measures the fit between a camera calibration $\mathcal{H}$ and a set of vehicle detections $\mathcal{D}$. The overall calibration procedure can thus be simply formulated as

$$\mathcal{H}^* = \arg\min_{\mathcal{H}} \mathcal{E}(\mathcal{H}, \mathcal{D}). \tag{1}$$

In this paper, we make the (reasonable) assumption that the road portion visible in the scene is planar. In this case, calibrating the camera amounts to recovering an homography $\mathcal{H} : \mathbb{R}^2 \to \mathbb{R}^2$ that maps a position $x \in \mathbb{R}^2$ in the metric road coordinate system to a pixel position $\mathcal{H}(x)$:

$$\mathcal{H}(x) = \left(\frac{H_1 \bar{x}}{H_3 \bar{x}}, \frac{H_2 \bar{x}}{H_3 \bar{x}} \right),$$

where $H \in \mathbb{R}^{3\times3}$ is the homography matrix (H_i denotes the i-th row) and $\bar{x} = (x_0, x_1, 1)$. We point out that H conveniently includes all extrinsic and intrinsic camera parameters, *i.e.* it can be uniquely decomposed into a rigid motion (rotation R and translation T forming $[R|T] \in SE(3)$) followed by a projection on the image plane using the intrinsic calibration matrix $K \in \mathbb{R}^{3\times3}$:

$$H = K\,[R\,|T]\,D, \tag{2}$$

where $D = \text{diag}([1, 1, 0, 1])$ projects onto the xy plane. While an homography normally has 8 degrees-of-freedom (DOF), we can reduce this number to only 3 free parameters in our particular case by assuming square pixels, no skew, a principal point at the image center as in [5], no camera roll and noticing that vehicle speeds, *i.e.* length measures, are invariant to translations and rotations in the 2D road plane. Note that these simplifying assumptions have little impact on the final system accuracy even when they are strongly violated, see Section B in the Supplementary. Specifically, an homography is generated by selecting a focal f, a camera tilt angle γ and a camera height z (in meters) in Eq. (2):

$$H = \mathcal{T}\left(\frac{I_w}{2}, \frac{I_h}{2}\right) F \left[\begin{array}{c|c} \mathcal{R}^x(\gamma) & \begin{matrix} 0 \\ 0 \\ -z \end{matrix} \end{array}\right] D, \tag{3}$$

where $\mathcal{T} : \mathbb{R}^2 \to \mathbb{R}^{3\times3}$ is a 2D translation, $F = \text{diag}([f, f, 1])$, $\mathcal{R}^x : \mathbb{R} \to \mathbb{R}^{3\times3}$ is a 3D rotation around the x-axis, and (I_w, I_h) is the image dimension. In the following, we denote the Jacobian of $\mathcal{H}$ at a position x as $J_{\mathcal{H}}(x) \in \mathbb{R}^{2\times2}$.

3.1.1 Energy function

Our goal is to evaluate the fit of a given calibration $\mathcal{H}$ w.r.t. a set of detected vehicle tracks $\mathcal{D}$. Specifically, we define $\mathcal{D} = \{\mathcal{T}_j\}$ as a set of $|\mathcal{D}|$ vehicle tracks produced by a vehicle detector and tracker, where each track

$\mathcal{T}_j = \{(b_{j,n}, t_{j,n})\}$ is composed of a sequence of 2D car bounding boxes $b_{j,n} \in \mathbb{R}^4$ and their corresponding timestamps $t_{j,n} \in \mathbb{R}$. The key insight is that the size of the bounding boxes should roughly match what the calibration $\mathcal{H}$ would predict, knowing the actual 3D car's positions and dimensions in the 3D world. In other words, the energy $\mathcal{E}(\mathcal{H}, \mathcal{D})$ consists of a reprojection error between the true bounding boxes and the ones fitted using $\mathcal{H}$, *i.e.* $\mathcal{E}(\mathcal{H}, \mathcal{D}) = \sum_{j=1}^{|\mathcal{D}|} \mathcal{E}_j(\mathcal{H}, \mathcal{T}_j)$ with

$$\mathcal{E}_j(\mathcal{H}, \mathcal{T}_j) = \sum_{n=1}^{|\mathcal{T}_j|} d(b_{j,n}, p_{\mathcal{H}}(b_{j,n})), \tag{4}$$

where $d : \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}^+$ is a box error function and $p_{\mathcal{H}}(b_{j,n})$ is the predicted bounding box computed as follows. Assume a car visible on the road and its corresponding (observed) bounding box $b \in \mathbb{R}^4$. Further assume knowledge of the car's geometry, which we denote as an array of 3D points $A \in \mathbb{R}^{3\times N}$ bottom-centered at 0 and aligned with the three main axis (*i.e.* length, width and height respectively aligned with x-, y- and z-axis). Further denote the rotation $R_b \in \mathbb{R}^{3\times 3}$ and translation $T_b \in \mathbb{R}^3$ that transport the car on the road in the 3D world and orientate it properly, then the bounding box $p_{\mathcal{H}}(b)$ can be computed as

$$p_{\mathcal{H}}(b) = \left(\min_i A'_{1,i}, \min_i A'_{2,i}, \max_i A'_{1,i}, \max_i A'_{2,i}\right) \tag{5}$$

with $A' = \mathcal{P}(R_b A + T_b)$ and $\mathcal{P} : \mathbb{R}^3 \to \mathbb{R}^2$ is the 3D counterpart of $\mathcal{H}$, constructed as in Eq. (2) but without D. The car geometry A can be assumed fixed as most cars have similar shapes, hence the goal of the fitting procedure is to calculate R_b and T_b knowing the observed bounding box b.

For the rotation R_b, we first point out that the car motion in the image and in the real-world road are related by the homography's Jacobian. More specifically, let μ_b denote the center of the bounding box. The apparent motion of the car in the image can be expressed as $m_b = \frac{\partial \mu_b}{\partial t} \in \mathbb{R}^2$. The Jacobian of $\mathcal{H}^{-1}$, denoted as $J_{\mathcal{H}^{-1}} = J_{\mathcal{H}}^{-1}$ yields by definition the motion $q_b \in \mathbb{R}^2$ in the real-world road plane:

$$q_b = J_{\mathcal{H}}^{-1}(\mu_b)\, m_b. \tag{6}$$

Hence the car's principal axis must then be aligned with q_b. Since the car is horizontally sitting on the xy plane, its vertical axis is $z = [0, 0, 1]^\top$. All in all, we have $R_b = [\bar{q}_b, z \times \bar{q}_b, z] \in \mathbb{R}^{3\times 3}$ where $\bar{q}_b = q_b / \|q_b\|$ and $\times$ denotes the cross-product.

For the translation T_b, we proceed with an iterative fixed-point algorithm. As initial estimate, we project μ_b on the road manifold $T_b^0 = \mathcal{H}^{-1}(\mu_b)$. Due to perspective effects and the fact that the car's center μ_b is slighlty above the road plane, the predicted box $p_{\mathcal{H}}^0(b)$ is not well centered on b. We then iteratively correct T_b^k, $k = 0 \dots K$, as follows. Based on T_b^k, we first compute $p_{\mathcal{H}}^k(b)$ from Eq. (5) and its apparent center $\mu\left(p_{\mathcal{H}}^k(b)\right)$. The translation T_b^k is then updated according to the difference between the expected and predicted centers:

$$T_b^{k+1} = T_b^k + T_b^0 - \mathcal{H}^{-1}\left(\mu\left(p_{\mathcal{H}}^k(b)\right)\right). \tag{7}$$

This procedure converges very quickly in practice, as we find that $K = 3$ iterations suffice to almost perfectly align the observed and predicted bounding boxes.

3.1.2 Box error function

We measure the error $d(b, b')$ between 2 boxes, defined by their boundaries $b, b' = (\text{left}, \text{top}, \text{right}, \text{bottom})$, using their weighted intersection-over-union (IoU):

$$d_{\text{IoU}}(b, b') = \alpha_b \left(1 - \frac{\text{intersection}(b, b')}{\text{union}(b, b')}\right). \tag{8}$$

We find that larger boxes convey more information due to perspective effects being more pronounced for objects closer to the camera. Because perspective effects are ultimately the only way to precisely estimate the focal, we upweight errors on those closer (hence larger) boxes with $\alpha_b = \sqrt{\text{area}(b)}$.

Masked IoU. We also consider a more precise version of this error that receives masks instead of rectangular boxes. A binary mask indicating the presence or absence of the car for each image pixel can be obtained either as a direct output of the vehicle detector, for instance using Mask-RCNN [20], or it can be computed using background subtraction techniques [16]. The formula stays the same as in Eq. (8) except that $b, b' \in \{0, 1\}^{I_w \times I_h}$ are binary masks. We denote the masked IoU similarity as $d_{\text{msk-IoU}}$.

3.1.3 Dealing with car categories

We assumed above that the car geometry was known in advance. In reality, this is not quite the case as there exists several categories of cars (*e.g.* sedan, SUV, etc.) which have different shapes and dimensions, see Fig. 3. We therefore collect a catalog $\mathcal{A}$ of several 3D car models and modify Eq. (4) accordingly in order to identify the optimal prototype for each vehicle track:

$$\mathcal{E}(\mathcal{H}, \mathcal{T}_j) = \min_{A \in \mathcal{A}} \sum_{n=1}^{|\mathcal{T}_j|} d(b_{j,n}, p_{\mathcal{H},A}(b_{j,n})). \tag{9}$$

Computing speeds. As a direct by-product of this energy minimization, we compute vehicle speeds straightforwardly based on the recovered 3D box translations $\{T_b\}$ from Eq. (7), i.e. $v_b = \left\| \frac{\partial T_b}{\partial t} \right\|$.

Figure 3. The catalog $\mathcal{A}$ of 3D car models used in Eq. (9).

3.2. Learning-based method

The reprojection method presented above is robust and works well in practice but is computationally demanding. Indeed its complexity linearly increases with (i) the number of detections, (ii) the number of 3D models in the catalog $\mathcal{A}$, and (iii) the number of 3D points per model. In practice, the minimization takes about 20 minutes for a short video clip of 30 seconds (see Section 4.2). In this section, we propose a fast method that learns to directly predict the calibration. It consists of a deep network f_θ that takes as input a set of car detections $\mathcal{D} = \{b, \dots\}$ and outputs a corresponding set $\hat{\mathcal{J}} = \left\{(\hat{\mu}_b, \hat{J}_b), \dots\right\}$, where $\hat{\mu}_b \simeq \mathcal{P}(T_b)$ predicts the 2D position of the actual 3D car bottom-center T_b, and $\hat{J}_b \simeq J_{\mathcal{H}}(\hat{\mu}_b)$ predicts the Jacobian of $\mathcal{H}$ at this position.

Direct speed estimation. The network output $\hat{\mathcal{J}}$ contains enough information to fully recover the car speed v_b. Indeed the speed is defined as the norm of its 3D motion $v_b = \|q_b\| = \left\|\frac{\partial T_b}{\partial t}\right\|$ and, according to Eq. (6):

$$\hat{q}_b \simeq \left(\hat{J}_b\right)^{-1} \frac{\partial \hat{\mu}_b}{\partial t}. \tag{10}$$

Homography prediction. In practice, the regressions output by the network are noisy, hence it is desirable to filter them and improve their overall consistency. Specifically, we aim at recovering an homography $\hat{\mathcal{H}}$ that achieves a good consensus among all output Jacobians in $\hat{\mathcal{J}}$. We achieve this goal using a RANSAC procedure. At each iteration, we randomly sample 2 predictions $(\hat{\mu}_i, \hat{J}_i)$ and $(\hat{\mu}_j, \hat{J}_j)$, $i \neq j$, and recover a homography $\hat{\mathcal{H}}_{i,j}$ consistent with both Jacobians (see Supplementary). We then compute the consensus score of $\hat{\mathcal{H}}_{i,j}$ with respect to all Jacobians in $\hat{\mathcal{J}}$. Finally we select the homography that maximizes the consensus:

$$\hat{\mathcal{H}} = \arg\max_{i,j} \sum_{n=1}^{|\hat{\mathcal{J}}|} \text{score}\left(\hat{\mathcal{H}}_{i,j}, \hat{\mu}_n, \hat{J}_n\right). \tag{11}$$

The score is defined as the similarity between $\hat{J}_n$ and the corresponding Jacobian according to $\hat{\mathcal{H}}_{i,j}$:

$$\text{score}(\hat{\mathcal{H}}, \hat{\mu}, \hat{J}) = \text{sim}\left(J_{\hat{\mathcal{H}},1}(\hat{\mu}), \hat{J}_1\right) \text{sim}\left(J_{\hat{\mathcal{H}},2}(\hat{\mu}), \hat{J}_2\right).$$

The similarity is computed for both column vectors of $\hat{J} = [\hat{J}_1, \hat{J}_2]$ using a criterion that penalizes the orientation difference (1^{st} factor) and the norm difference (2^{nd} factor):

$$\text{sim}(V, V') = \frac{V^\top V'}{\|V\| \, \|V'\|} \cdot \frac{\min\left(\|V\|, \|V'\|\right)}{\max\left(\|V\|, \|V'\|\right)}.$$

Figure 4. Visualisation of the representation r_b extracted for car mask b. *Left*: Synthetic scene. *Right*: ellipses (1^{st} and 2^{nd} order moments) and normalized motion (arrows) on top of binary car masks. More examples are in the supplementary.

3.2.1 Network architecture and training

We use a transformer architecture for the network as it naturally handles inputs in the form of a set. Specifically, we stack several encoder blocks as in, *e.g.*, BERT [14]. Since the network takes as input a set of vectors, we need to compute a representation $r_b \in \mathbb{R}^B$ for each detection b. For the sake of simplicity as well as for bridging the domain gap between synthetic and real data (see below), we choose a simple and straightforward representation. Given a car detection in the form of a pixel mask $b \in \{0,1\}^{I_w \times I_h}$, with non-zero pixels $P_b = \{(i,j) | b_{i,j} = 1\}$, we concatenate the first- and second-order moments of P_b (*i.e.* an ellipse) together with its apparent motion m_b (see Eq. (6)), all being normalized for generalization purposes. We obtain an 8-dimensional representation that formally writes as

$$r_b = \left[\text{mean}(\frac{P_b}{I_w}), \text{cov}(\frac{P_b}{I_w}), \frac{m_b}{\|m_b\|}\right] \in \mathbb{R}^8.$$

While it can be argued that more sophisticated representations based *e.g.* on auto-encoders [37] of car images could be used, this representation offers the advantage of being compatible with fully synthetic training, almost completely eliminating the domain gap. In addition, it is simple and yields good performance in practice, see Section 4.

Training. To train the network, we generate synthetic training data on the fly by randomly generating scenes with different camera, road, and car poses (see Eq. (3)). For each scene, we render each car's mask and then extract their representations $\{r_b, \dots\}$. Examples of rendered masks and corresponding representations are shown in Figure 4.

For each scene, a set of 32 detections is fed to the network. We zero-pad the representation to 64 dimension and use 256 hidden layers in the encoder blocks. The 64-dimensional network outputs are linearly projected to 8-dimensional vectors $z = [\hat{\mu} - \hat{J}_1, \hat{\mu} - \hat{J}_2, \hat{\mu} + \hat{J}_1, \hat{\mu} + \hat{J}_2] \in \mathbb{R}^{4\times 2}$ from which we can recover both $\hat{\mu} = \frac{1}{4}\sum_n z_n$ and

$\hat{J} = \frac{1}{2}[z_3 - z_1, z_4 - z_2]$. We minimize the ℓ_2 loss between the output and the ground-truth targets μ^* and J^*:

$$L(\hat{\mu}, \hat{J}, \mu^*, J^*) = \|\hat{\mu} - \mu^*\|^2 + \lambda \left\|\hat{J} - J^*\right\|^2, \quad (12)$$

where λ is a hyper-parameter. We also tried to directly optimize for the speed error, but due to the instability of the matrix inversion involved in Eq. (10) we obtained consistently inferior results.

4. Experiments

We now evaluate the two proposed speed estimation approaches on synthetic and real data. In particular, we introduce in Section 4.4 a new dataset that most closely represents actual CCTV footage.

Metrics. In order to get unbiased performance with respect to the target task, we report results in terms of speed error (in km/h if absolute or % if relative). Unless told otherwise, we compute the median error for each video clip and report the average and median errors at the dataset level.

4.1. Implementation details

3D car models. We obtain from the Unity 3D engine catalog[1] a set of 10 realistically shaped vehicles shown in Figure 3. They comprise a variety of car categories and shapes (sedan, SUV, etc.). Each vehicle track in the synthetic datasets that we generate is randomly assigned to one of these 10 models.

Energy minimization. The energy function defined in Eq. (9) is not convex and may comprise many local minima, but we seek to find the global one (see Eq. (1)). We thus adopt a robust Gaussian optimization algorithm [7] to approach this goal in minimal time. The algorithm iteratively samples the 3 homography parameters from Eq. (3) randomly in the following ranges: focal $f \in [I_w/5, 5I_w]$, tilt $\gamma \in [0, \pi/2]$, and height $z \in [2, 50]$. The value of the error $\mathcal{E}(\mathcal{H}, \mathcal{D})$ is used to update a tree-based probabilistic model for drawing the next random samples [7]. We return the best model found after $\Gamma = 5000$ trials.

Network training. We use a stack of 8 encoder layers to construct the network f_θ. We train the network for 10,000 epochs, where each epoch comprises 128 scenes and each scene contains 32 cars. We feed the network with batches of 16 scenes and perform gradient descent using Adam [26] with an exponentially decaying learning rate starting at 3.10^{-3} and ending at 10^{-4}, without weight decay. We fix the loss hyper-parameter to $\lambda = 4$ to favor an accurate Jacobian $\hat{J}$. At each epoch, we measure the average speed error using Eq. (10) on a held-out validation set, randomly generated as well. We select the model with the minimum validation error and use it for the experiments. Training curves are shown in Figure 5.

[1] https://assetstore.unity.com

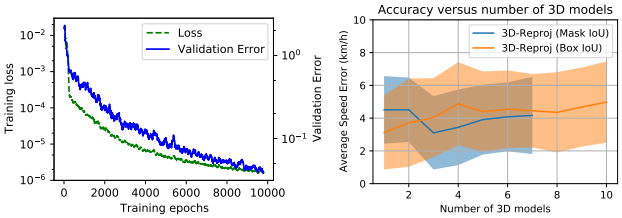

Figure 5. *Left*: Training curves for the learned approach. *Right*: Accuracy as a function of the number of 3D models in the library on the synthetic benchmark. Variance for each curve is indicated with a shaded area.

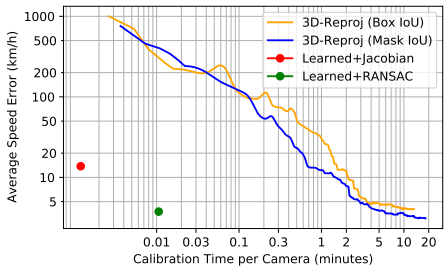

Figure 6. Speed error as a function of the time spent to compute the calibration for different methods.

4.2. Synthetic dataset

We first evaluate our proposed approaches in perfectly controlled conditions with a synthetic benchmark and adjustable noise levels. Using the same procedure than for the training dataset, we generate 128 short video clips of 128 frames each with a resolution of 1024x768 pixels. For each clip, vehicles are placed randomly on the road (no collisions are allowed) and are given a random speed in the range $[30, 100]$ km/h. Sample scenes are presented in Figure 4.

Number of 3D car categories. We first focus on the reprojection error minimization method from Section 3.1, denoted as '3D-reproj' in the following, in noiseless conditions. We plot in Figure 5 the speed error for different numbers of 3D models in the catalog $\mathcal{A}$ (Section 3.1.3) and the box error functions (Section 3.1.2). Surprisingly, we observe no significant difference in accuracy regardless of the box error function or the number of models used. In fact, the error even slightly increases when this number augments, which may be explained by the fact that Eq. (9) becomes more complex and thus harder to minimize. Since computing time is directly proportional to the number of used 3D models, we limit the catalog $\mathcal{A}$ to 3 models for the reprojection-based method in the remainder of this paper.

Accuracy versus time. We plot the calibration accuracy as a function of the computation time for each method in Figure 6. For 3D-reproj, we vary the number of trials $\Gamma \in \{1, \ldots, 5000\}$. For the learning-based method (Section 3.2), we measure the average time taken for the

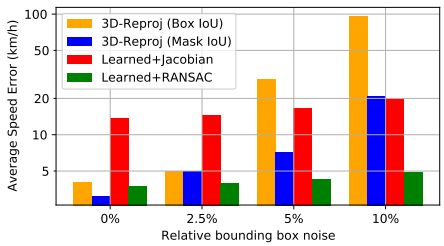

Figure 7. Impact of noise on the different methods.

Figure 8. Sample frames from the BrnoCompSpeed dataset [44]. Note the lack of diversity, high resolution (2 MPix) and the overall lack of challenges (*e.g.* roads are straight, well illuminated, etc.)

direct speed estimation using the Jacobians (10) or using RANSAC (11). While all methods except Learned-Jacobian obtain good results (*i.e.* below 4 km/h in average absolute error), it is clear that 3D-reproj is many orders of magnitude slower than learning-based methods. The slowest masked-IoU version yet achieves the best accuracy overall. In contrast, learned methods are almost instantaneous. For the case of Learned-Jacobian, we observe poor accuracy caused by noisy regression output $\hat{\mathcal{J}}$, further reinforced by the simplistic speed estimation scheme from Eq. (10) that tends to accumulate errors. Filtering $\hat{\mathcal{J}}$ with RANSAC instead turns out very competitive, almost reaching the same accuracy than 3D-reproj combined with masked IoU.

Impact of noise. We add Gaussian noise to the bounding box coordinates (*i.e.* masks are translated). We experiment with different strengths of noise relatively to the box sizes and present results in Figure 7. While all methods are affected by noise, we observe that 3D-reproj is much more sensitive, especially if the pixel mask is not used. Therefore, we use the mask-based error in all remaining experiments. Surprisingly, the learning-based method is nearly unaffected and appears very robust, even though it is trained on noiseless data.

4.3. Results on the BrnoCompSpeed dataset

The BrnoCompSpeed dataset [44] consists of 18 videos (6 locations, 3 viewpoints per location) and comprises a total of 20,865 vehicles annotated with ground-truth speed. It covers viewpoints typical for traffic surveillance (see Figure 8) and various traffic conditions (low traffic in Session 3, high traffic in Sessions 5 and 6).

Detection and tracking. We use an off-the-shelf object detector and tracker to compute the detections set $\mathcal{D}$. Specifically, we use the default pre-trained Mask-RCNN [20] model from PyTorch [1]. To track detections, similarly to Kumar *et al.* [28] we use SORT [8], an online Kalman-filter-based algorithm that is simple and efficient. Note that tracking also enables to filter out most false detections. Namely, we eliminate still tracks (no motion) and spurious tracks that contain not enough detections. We also remove non-car tracks (*e.g.* trucks) based on the label provided by Mask-RCNN as well as tracks that are in masked regions according to the provided video mask. This off-the-shelf pipeline achieves state-of-the-art recall on the BrnoCompSpeed dataset [44], see Table 1.

	Recall	FPPM
FullACC [16]	0.885	9.77
FullACC++ [43]	0.863	1.91
Ours	0.948	4.61

Table 1. Evaluation of our detection and tracking pipeline on the BrnoCompSpeed dataset [44] in terms of recall and false positive per minutes (FPPM).

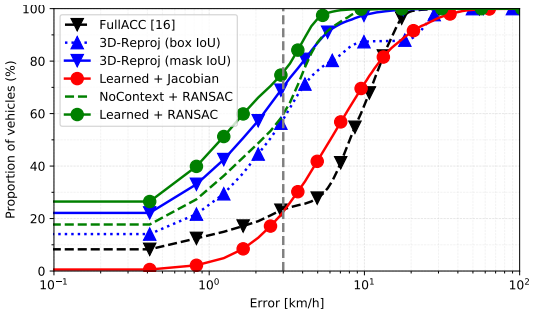

Figure 9. Cumulative histogram of absolute errors for the BrnoCompSpeed dataset [44]. The vertical dashed line indicates the 3 km/h error threshold.

	Abs error (km/h)		Rel error (%)		Time (s)
	avg	median	avg	median	avg
3D-reproj (box IoU)	5.70	2.85	7.04	3.61	1.49K
3D-reproj (mask IoU)	2.84	2.03	3.46	2.58	2.84K
Learned+Jacobian	9.21	6.47	11.51	8.12	15.9
NoContext+RANSAC	3.01	2.59	3.69	3.31	**2.5**
Learned+RANSAC	**2.15**	**1.60**	**2.65**	**2.07**	2.9
FullACC [16]	8.59	8.45	10.89	11.41	200
FullACC++ [43]	**1.10**	**0.97**	**1.39**	**1.22**	>200

Table 2. Results for the BrnoCompSpeed dataset. Note that FullACC++ [43] results are not strictly comparable as they were obtained on a subset of 9/18 videos, the other 9 videos being used to train their method.

Results. To calibrate cameras on the BrnoCompSpeed dataset [44], we run our methods on a subset of 100 vehicles tracks (or less) detected in the first 6 minutes of each video. We compute results on the split A using the official evaluation code and report them in Figure 9 and Table 2. Overall, the proposed methods perform excellently, being it handcrafted (3D-reproj) or learned (Learned-RANSAC). They also largely outperform the fully automatic method FullACC [16] in terms of absolute and relative errors. Note that [43] reports even better results for their improved version FullACC++, but they are not strictly comparable as

	Abs error (km/h)		Rel error (%)		Time (s)
	avg	med	avg	med	avg
3D-reproj (mask IoU)	9.51	5.45	31.9	13.8	2500
Learned+Jacobian	20.1	18.6	53.2	45.8	**0.11**
Learned+RANSAC	**12.8**	**5.82**	**29.4**	**16.0**	0.45
FullACC++* [43]	32.6	22.4	56.0	52.9	27

Table 3. Results on the CCTV dataset. FullACC++* is our re-implementation of [43].

they are obtained on a subset only, *i.e.*, on half of the videos. The other half was used to train and fine-tune their method. In comparison, we find remarkable that our method yields a median absolute error below 2 km/h while being trained solely from synthetic data using off-the-shelf 3D car models. This shows that car shapes are indeed well normalized, at least on the long run and in a median sense. Finally, we also report average calibration times per video on a single CPU core in Table 2 (not counting the detection and tracking steps). Our learned method is orders of magnitude faster than other methods.

Ablative study. For the learned method, we also report results with direct speed estimation using Jacobians. Confirming earlier findings, this naive way of estimating speeds performs relatively poorly and emphasizes the necessity of filtering predictions. We also experiment with a network trained to process each detection individually (*i.e.* we set the transformer attention mask to an identity matrix). This variant, denoted as 'NoContext+RANSAC' in Figure 9 and Table 2, prevents the network to use the context provided by the other detections to infer the global geometry of the scene. Yet, results for this variant are only marginally worse than using full context. This suggests that there exists strong priors on the representation of individual cars (*i.e.* their position, shape, and motion) and the scene geometry, which the network appears to learn effectively.

4.4. The CCTV dataset

The BrnoCompSpeed dataset was constructed to evaluate traffic monitoring algorithms using cameras specifically designed and installed for this purpose. As a result, it does not reflect the typical quality of CCTV cameras (compare Figure 8 and 2 for instance) which intent to provide a rough overview of the traffic situation for human operators. Namely, it features high-resolution videos (1920x1080 = 2M pixels) shot with a high-quality camera and thus does not represent the diversity of capturing conditions, including motion blur, compression artifacts, and lens imperfections.

We thus introduce a novel dataset, denoted as the CCTV dataset, to better reflect the actual content and conditions of real-life CCTV cameras. It comprises 40 short video clips sampled from publicly streaming CCTV cameras located in South Korea. Since the clips originate from actual installed cameras, the exact camera calibration is unknown. Instead, in order to obtain ground-truth vehicle speeds, we manually annotated for each clip a sequence of bounding boxes corresponding to the passage of a car equipped with a GPS tracker storing its speed and location.

In contrast to the BrnoCompSpeed dataset, it encompasses all challenges that are normally encountered in practice: image resolution is variable and often low, quality ranges between poor and mediocre, roads are not necessarily straight, camera lens is imperfect, etc. Detailed statistics can be found in the Supplementary (Figure A.1) and examples of frames in Figure 2. Compared to the BrnoCompSpeed dataset, the image resolution is up to 20x smaller, making cars appear sometimes as small as 20 pixels at their maximum size, and the overall amount of noise is much higher.

Results. We use the same detection and tracking pipeline than earlier for all methods. Results are presented in Table 3. As expected, results are noticeably worse than those obtained on the clean BrnoCompSpeed dataset. Still, the 3D-reproj and Learned-RANSAC methods both yield a median absolute error of about 5 km/h, which is reasonably low given the dataset challenges mentioned above.

We also compare with a re-implementation of FullACC++ [43] based on code snippets shared by the authors. We try several parameter ranges for the initialization of the focal and the camera roll and report the best results in the last row of Table 3. FullACC++, a method based on subpixel-accurate image processing, performs dramatically poorly (over 20 km/h absolute speed error). This is logically explained by the poor resolution and quality of CCTV cameras that prevent any accurate estimates of the vanishing point positions, as exemplified in Figure 1.

Finally, our Learned+RANSAC method is 2 orders of magnitude faster than FullACC++ [43] and also much simpler, both conceptually and practically as it does not require additional components to categorize vehicle sub-types, detect keypoints nor requires specific 3D CAD models.

5. Conclusion

We have presented two novel approaches for automatically estimating vehicle speeds in traffic surveillance footage. In contrast to existing methods, both approaches require minimal input (that is, a small set of tracked bounding boxes) to output a fully-calibrated homography from which velocities can be computed directly given pixel motion. By doing so, we reduce most problems caused by the poor resolution and other types of noise inevitably present in traffic surveillance footage. Extensive experiments on different datasets, including a realistic CCTV benchmark, validate our calibration-from-boxes concept while emphasizing the shortcomings of state-of-the-art methods.

Given the wide availability of public CCTV streams, we believe that our method can be applied broadly as a low-cost traffic speed sensor in order to improve traffic analysis.

References

[1] https://pytorch.org/vision/stable/_modules/torchvision/models/detection/mask_rcnn.html. 7

[2] Heba A. Kurdi. Review of Closed Circuit Television (CCTV) Techniques for Vehicles Traffic Management. *International Journal of Computer Science and Information Technology*, 6:199–206, Apr. 2014. 1

[3] Junaid Ahmed Ansari, Sarthak Sharma, Anshuman Majumdar, J. Krishna Murthy, and K. Madhava Krishna. The Earth Ain't Flat: Monocular Reconstruction of Vehicles on Steep and Graded Roads from a Moving Camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Madrid, Spain*, pages 8404–8410. IEEE, 2018. 3

[4] V. Bartl and A. Herout. OptInOpt: Dual Optimization for Automatic Camera Calibration by Multi-Target Observations. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8, Sept. 2019. ISSN: 2643-6213. 1, 3

[5] Vojtěch Bartl, Roman Juranek, Jakub Špaňhel, and Adam Herout. PlaneCalib: Automatic Camera Calibration by Multiple Observations of Rigid Objects on Plane. In *2020 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Melbourne, Australia, Nov. 2020. IEEE. 1, 2, 3

[6] Vojtěch Bartl, Jakub Špaňhel, Petr Dobeš, Roman Juránek, and Adam Herout. Automatic camera calibration by landmarks on rigid objects. *Machine Vision and Applications*, 32(1):2, Oct. 2020. 1, 2, 3

[7] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *NeurIPS*, page 9, 2011. 6

[8] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468. IEEE, 2016. 2, 7

[9] Romil Bhardwaj, Gopi Krishna Tummala, Ganesan Ramalingam, Ramachandran Ramjee, and Prasun Sinha. Autocalib: Automatic traffic camera calibration at scale. *ACM Transactions on Sensor Networks (TOSN)*, 14(3-4):1–27, 2018. 1, 2, 3

[10] Huikun Bi, Zhong Fang, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. Joint Prediction for Kinematic Trajectories in Vehicle-Pedestrian-Mixed Scenes. In *ICCV*, page 10, 2019. 1

[11] Frederick W. Cathey and Daniel J. Dailey. Mathematical theory of image straightening with applications to camera calibration. In *IEEE Intelligent Transportation Systems Conference, ITSC 2006, Toronto, Ontario, Canada, 17-20 September 2006*, pages 1364–1369. IEEE, 2006. 2

[12] A.B. Chan and N. Vasconcelos. Classification and retrieval of traffic video using auto-regressive stochastic processes. In *IEEE Proceedings. Intelligent Vehicles Symposium*, pages 771–776, 2005. 2

[13] Roberto Cipolla, Tom Drummond, and Duncan Robertson. Camera Calibration from Vanishing Points in Image of Architectural Scenes. In *BMVC*, volume 2, 1999. 2

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, Volume 1*, pages 4171–4186, 2019. 5

[15] Markéta Dubská and Adam Herout. Real Projective Plane Mapping for Detection of Orthogonal Vanishing Points. In *Procedings of the British Machine Vision Conference 2013*, pages 90.1–90.11, Bristol, 2013. British Machine Vision Association. 2

[16] Markéta Dubská, Adam Herout, Roman Juranek, and Jakub Sochor. Fully Automatic Roadside Camera Calibration for Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1162–1171, June 2015. 1, 2, 4, 7

[17] Markéta Dubská, Adam Herout, and Jakub Sochor. Automatic Camera Calibration for Traffic Understanding. In *Proceedings of the British Machine Vision Conference 2014*, pages 42.1–42.12, Nottingham, 2014. British Machine Vision Association. 1, 2

[18] Panagiotis Giannakeris, Vagia Kaltsa, Konstantinos Avgerinakis, Alexia Briassouli, Stefanos Vrochidis, and Ioannis Kompatsiaris. Speed Estimation and Abnormality Detection from Surveillance Cameras. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 93–936, Salt Lake City, UT, USA, June 2018. IEEE. 1, 2

[19] Lazaros Grammatikopoulos, George Karras, and Elli Petsa. Automatic estimation of vehicle speed from uncalibrated video sequences. In *International Symposium on Modern Technologies, Education and Professional Practice in the Geodesy and Related Fields*, 2005. 2

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv:1703.06870 [cs]*, Jan. 2018. arXiv: 1703.06870. 4, 7

[21] Xiao-Chen He and N. H. C. Yung. A Novel Algorithm for Estimating Vehicle Speed from Two Consecutive Images. In *8th IEEE Workshop on Applications of Computer Vision (WACV 2007), 20-21 February 2007, Austin, Texas, USA*, page 12. IEEE Computer Society, 2007. 2

[22] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint Monocular 3D Vehicle Detection and Tracking. In *ICCV*, page 10, 2019. 2

[23] Tingting Huang. Traffic speed estimation from surveillance video data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 161–165, 2018. 1, 2

[24] Sabbani Imad, Perez-uribe Andres, Bouattane Omar, and El Moudni Abdellah. Deep convolutional neural network architecture for urban traffic flow estimation. *IJCSNS*, 2018. 1

[25] Roman Juranek, Adam Herout, Markéta Dubská, and Pavel Zemcik. Real-Time Pose Estimation Piggybacked on Object Detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2381–2389, Santiago, Dec. 2015. IEEE. 3

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun,

editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 6

[27] Viktor Kocur. Perspective transformation for accurate detection of 3D bounding boxes of vehicles in traffic surveillance. In *Proceedings of the 24th Computer Vision Winter Workshop*, 2019. 2

[28] Amit Kumar, Pirazh Khorramshahi, Wei-An Lin, Prithviraj Dhar, Jun-Cheng Chen, and Rama Chellappa. A Semi-Automatic 2D Solution for Vehicle Speed Estimation from Monocular Videos. In *CVPRW*, pages 137–1377, Salt Lake City, UT, USA, June 2018. IEEE. 1, 2, 7

[29] Abhijit Kundu, Yin Li, and James M. Rehg. 3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3559–3568. IEEE Computer Society, 2018. 3

[30] A Kurniawan, A Ramadlan, and EM Yuniarno. Speed monitoring for multiple vehicle using closed circuit television (cctv) camera. In *2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM)*, pages 88–93. IEEE, 2018. 1, 2

[31] Matthew J. Leotta and Joseph L. Mundy. Vehicle surveillance with a generic, adaptive, 3d vehicle model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1457–1469, 2011. 3

[32] Adrien Lessard, Francois Belisle, Guillaume-Alexandre Bilodeau, and Nicolas Saunier. The Counting App, or How to Count Vehicles in 500 Hours of Video. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1592–1600, Las Vegas, NV, USA, June 2016. IEEE. 1

[33] Jing Li, Shuo Chen, Fangbing Zhang, Erkang Li, Tao Yang, and Zhaoyang Lu. An Adaptive Framework for Multi-Vehicle Ground Speed Estimation in Airborne Videos. *Remote Sensing*, 11(10):1241, Jan. 2019. 2

[34] David Fernández Llorca, Antonio Hernández Martínez, and Iván García Daza. Vision-based Vehicle Speed Estimation for ITS: A Survey. *arXiv:2101.06159 [cs]*, Jan. 2021. arXiv: 2101.06159. 2

[35] Diogo Luvizon, Bogdan Tomoyuki Nassu, and Rodrigo Minetto. A Video-Based System for Vehicle Speed Measurement in Urban Roadways. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–12, Sept. 2016. 1, 2

[36] C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimation of vehicle velocity and traffic intensity using rectified images. In *2008 15th IEEE International Conference on Image Processing*, pages 777–780, Oct. 2008. ISSN: 2381-8549. 2

[37] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 5

[38] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: Monocular Lifting of 2D Detection to 6D Pose and Metric Shape. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2069–2078. Computer Vision Foundation / IEEE, 2019. 3

[39] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5632–5640. IEEE Computer Society, 2017. 3

[40] Julian Nubert, Nicholas Giai Truong, Abel Lim, Herbert Ilhan Tanujaya, Leah Lim, and Mai Anh Vu. Traffic density estimation using a convolutional neural network. *arXiv preprint arXiv:1809.01564*, 2018. 1

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2

[42] Todd Nelson Schoepflin. *Algorithms for estimating mean vehicle speed using uncalibrated traffic management cameras*. University of Washington, 2003. 1, 2

[43] Jakub Sochor, Roman Juránek, and Adam Herout. Traffic Surveillance Camera Calibration by 3D Model Bounding Box Alignment for Accurate Vehicle Speed Measurement. *Computer Vision and Image Understanding*, 161:87–98, Aug. 2017. arXiv: 1702.06451. 1, 2, 7, 8

[44] Jakub Sochor, Roman Juránek, Jakub Špaňhel, Lukáš Maršík, Adam Široký, Adam Herout, and Pavel Zemčík. Comprehensive Data Set for Automatic Single Camera Visual Speed Measurement. *IEEE Transactions on Intelligent Transportation Systems*, 20(5):1633–1643, May 2019. 1, 2, 7

[45] Jakub Sochor, Jakub Špaňhel, and Adam Herout. BoxCars: Improving Fine-Grained Recognition of Vehicles using 3-D Bounding Boxes in Traffic Surveillance. *IEEE Trans. Intell. Transport. Syst.*, 20(1):97–108, Jan. 2019. arXiv: 1703.00686. 3

[46] Jakub Sochor, Jakub Spanhel, Roman Juranek, Petr Dobes, and Adam Herout. Graph@FIT Submission to the NVIDIA AI City Challenge 2018. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 77–777, Salt Lake City, UT, USA, June 2018. IEEE. 2

[47] Zheng Tang, Milind Naphade, Stan Birchfield, Jonathan Tremblay, William Hodge, Ratnesh Kumar, Shuo Wang, and Xiaodong Yang. PAMTRI: Pose-Aware Multi-Task Learning for Vehicle Re-Identification Using Highly Randomized Synthetic Data. In *ICCV*, page 10, 2019. 3

[48] Zheng Tang, Gaoang Wang, Tao Liu, Young-Gun Lee, Adwin Jahn, Xu Liu, Xiaodong He, and Jenq-Neng Hwang. Multiple-Kernel Based Vehicle Tracking Using 3D Deformable Model and Camera Self-Calibration. *CoRR*, abs/1708.06831, 2017. _eprint: 1708.06831. 3

[49] Zhongdao Wang, Luming Tang, Xihui Liu, Zhuliang Yao, Shuai Yi, Jing Shao, Junjie Yan, Shengjin Wang, Hongsheng Li, and Xiaogang Wang. Orientation Invariant Feature Embedding and Spatial Temporal Regularization for Vehicle Re-identification. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 379–387. IEEE Computer Society, 2017. 3

[50] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3109–3118. IEEE Computer Society, 2015. 3