Supplementary Material for: Robust Automatic Monocular Vehicle Speed Estimation for Traffic Surveillance

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

A. Additional information about the CCTV dataset

In this section, we include additional qualitative and quantitative information about the CCTV dataset. Even though all clips were taken in daylight in relatively good conditions, they constitute realistic CCTV footage. Figure A.2 shows examples containing intersections (22 scenes out of 40), complex layouts, and road turns. Figure A.1 presents detailed statistics about the CCTV dataset such as the image size, video duration, vehicle tracks, and vehicle speeds. Finally, Figure A.9 shows snapshots of all 40 videos. Note that several clips originate from the same scene (*i.e.* CCTV camera).

B. Simplified camera model

One might argue that the simplified camera model used in our approach (Section 3.1 of the main paper) could negatively impact performance in real-world situations where cameras are far from being perfect. We therefore conduct an experimental study on the impact of these simplifying assumptions on the final speed error. Figure A.3 presents the accuracy of both proposed approaches ('3D-Reproj' and 'Learned+RANSAC') when the camera lens is in fact imperfect (*i.e.* off-centered with roll and radial distortion implemented as proposed in [3]), measured on the synthetic dataset under perfectly controlled conditions. As can be seen in Figure A.3, we observe limited impact on final speed accuracy despite the simplifying assumptions being used.

In addition, we have also experimented with more complex camera models (*e.g.* with explicit principal point and camera roll). We did not observe significant improvements compared to the simplified camera model (perhaps due to the larger search space). We believe that this reason caused other researchers to adopt this strategy as well [5, 3, 2].

C. Implementation details for the learned method

C.1. Homography fitting using RANSAC

In Section 3.2 of the main paper, we explain how we perform a RANSAC procedure to robustly fit a homography to all predictions. In more detail, we sample random pairs of predicted Jacobians and compute for each pair a tentative homography that we score against all predicted Jacobians. We now explain how to compute a homography $\hat{\mathcal{H}}_{i,j}$ given 2 predictions $(\hat{\mu}_i, \hat{J}_i)$ and $(\hat{\mu}_j, \hat{J}_j)$, $i \neq j$, generated by the deep network f_θ.

The homography $\mathcal{H} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that maps (metric) road coordinates to pixels coordinates is defined by its matrix $H \in \mathbb{R}^{3\times3}$. In the main paper, we make some reasonable assumptions and, as a result, the homography is only governed by 3 free parameters according to Eq. (3) from the main paper:

$$\begin{aligned} H &= \mathcal{T}\left(\frac{I_w}{2}, \frac{I_h}{2}\right) F \left[\mathcal{R}^x(\gamma) \middle| \begin{matrix} 0 \\ 0 \\ -z \end{matrix}\right] D \\ &= \begin{bmatrix} f & 0 & I_w/2 \\ 0 & f & I_h/2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & 0 \\ 0 & -s_\gamma & -z \end{bmatrix} \\ &\propto \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ 0 & H_{22} & H_{23} \\ 0 & H_{32} & 1 \end{bmatrix}. \end{aligned} \tag{C.1}$$

By design, we have $H_{21} = H_{31} = 0$. This essentially means that there is no camera roll and that the first vanishing point is always at infinity. Moreover, notice that a 2D translation in the xy road plane has no effect on road distances and thus $H \equiv H.\mathcal{T}(u, v)$ where $(u, v) \in \mathbb{R}^2$ is a translation vector and $\equiv$ denotes equivalence:

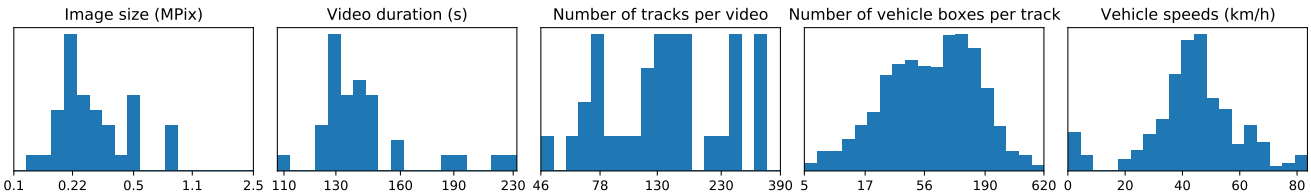

Figure A.1. Detailed statistics for the CCTV dataset. Each histogram represents the distribution of a particular metric in the CCTV dataset.

Figure A.2. Sample video clips with overlaid vehicle tracks featuring intersections and turning roads from the CCTV dataset.

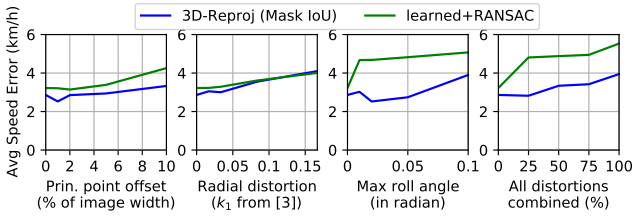

Figure A.3. Speed accuracy of the proposed approaches on the synthetic dataset under distortions caused by imperfect camera lens.

$$
\begin{aligned}
H &\equiv H.\mathcal{T}(u,v) \\
&= \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ 0 & H_{22} & H_{23} \\ 0 & H_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} H_{11} & H_{12} & H'_{13} \\ 0 & H_{22} & H'_{23} \\ 0 & H_{32} & 1 \end{bmatrix} \qquad \text{(C.2)}
\end{aligned}
$$

We can thus arbitrarily choose u and v, in particular, we choose them so that $H'_{13} = H'_{23} = 0$. As a result, we are left with an homography matrix H with only 4 non-zero parameters. Direct calculation shows that the Jacobian of $\mathcal{H}(x,y)$ develops as

$$
\begin{aligned}
J_{\mathcal{H}}(x,y) &= \begin{bmatrix} J_{11} & J_{12} \\ 0 & J_{22} \end{bmatrix} \\
&= \begin{bmatrix} \frac{H_{11}}{D} & \frac{1}{D^2}(H_{12} - xH_{11}H_{32}) \\ 0 & \frac{H_{22}}{D^2} \end{bmatrix} \qquad \text{(C.3)}
\end{aligned}
$$

where the denominator $D = 1 + yH_{32}$. Given 2 Jacobians of $\mathcal{H}$ denoted $J = J_{\mathcal{H}}(x,y)$ and $J' = J_{\mathcal{H}}(x',y')$ at locations $\mu = (x,y)$ and $\mu' = (x',y')$, we can jointly solve for H_{11} and H_{32} by posing

$$
\begin{cases} J_{11} = \frac{H_{11}}{1+yH_{32}} \\ J'_{11} = \frac{H_{11}}{1+y'H_{32}} \end{cases} \qquad \text{(C.4)}
$$

and provided that $y \neq y'$ we obtain

$$
\begin{cases} H_{32} = \frac{J'_{11} - J_{11}}{yJ_{11} - y'J'_{11}} \\ H_{11} = J_{11}(1 + yH_{32}) \end{cases} \qquad \text{(C.5)}
$$

It follows from Eq. (C.3) that $H_{22} = J_{22}(1 + yH_{11})^2$ and $H_{12} = J_{12}(1 + yH_{11})^2 + xH_{11}H_{32}$.

To summarize, we have shown that given two Jacobians of an homography, we can recover an equivalent homography. Since predicted Jacobians ideally approximate the true Jacobian, *i.e.* we have $\hat{J}_i \simeq J_{\hat{\mathcal{H}}_{i,j}}(\hat{\mu}_i)$ and $\hat{J}_j \simeq J_{\hat{\mathcal{H}}_{i,j}}(\hat{\mu}_j)$,

we apply the exact same algorithm to recover $\hat{\mathcal{H}}_{i,j}$ based on $\hat{J}_i$, $\hat{J}_j$, $\hat{\mu}_i = (\hat{x}_i, \hat{y}_i)$ and $\hat{\mu}_j = (\hat{x}_j, \hat{y}_j)$. In practice, the reconstructed homography $\hat{\mathcal{H}}_{i,j}$ does not exactly correspond to the predicted Jacobians due to estimation noise. Nevertheless the robust RANSAC procedure can reasonably deal with noise and imperfect estimations.

C.2. Test procedure for the learned method

The RANSAC procedure described in Section 3.2 of the main paper feeds upon a set of Jacobians estimated by the transformer network f_θ. During training, as explained in Section 3.2.1, we randomly sample 32 vehicle detections among all detections, extract a representation for each of them, and input these to the network to get their Jacobians (see Figure A.4).

During test, however, we use a slightly different procedure. Instead of randomly selecting a subset of detections, we feed vehicle tracks one by one to the network. Specifically, we only use 10 detections per track (regularly sampled), as it is faster to process without noticeable loss of performance. Figure A.5 illustrates this process. Overall, we thus accumulate 10 estimated Jacobians per vehicle track[1], and finally run the RANSAC procedure from this accumulated set of Jacobians. We use 1024 iterations of random pair sampling in the RANSAC procedure.

The reasons for this different test procedure w.r.t. training are two-fold: (1) this is more stable, since less randomness is involved (all detections are considered, not just a random subset); and (2) on real dataset, it often occurs that vans or small trucks are labelled as "cars" by Mask-RCNN and thus not filtered out. Unfortunately, these detections are significantly larger than any of the 3D car models used for training. As a result, we observe significantly impacted Jacobians when some of these outlier detections are included in the input to the network. On the contrary, treating tracks independently allows to isolate these bad cases. RANSAC is then able to successfully dismiss these outlier Jacobians, since they correspond to a small subset of all Jacobians that is not consistent with the rest.

D. Implementation details for the reprojection-based method

In Section 4.2 of the main paper, we experiment with different numbers of 3D models for the reprojection-based method. As explained in the main paper, the reason for this experiment is to reduce the computational complexity of the approach. In fact, the computational cost is linearly dependent on two factors: (i) the number of points per 3D car models (typically thousands); (ii) the number of 3D models. In practice, it is intractable to experiment with the original collection of 3D models.

[1]Sometimes less, if the track contains fewer than 10 detections.

We therefore perform the following two operations: (i) we simplify the original 3D meshes and (ii) we cluster 3D models into a smaller number of models. Note that these operations could be accomplished in several manners. In our case, we choose to use the vtk toolbox [4] to compute the convex hull of each car and to decimate it until 20 points remain. The resulting simplified 3D meshes are shown in Figure A.6. Next, we perform k-means clustering on the set of 3D models using the concatenated masks of each model on the X, Y, and Z axes as descriptor. We vary the number of target clusters between $K = 1$ and 10. For each value of K, we select the K models that are closest to the mean code-books output by the k-means algorithm. The selection procedure tends to select distinct but well-representative car categories, for instance for $K = 2$ a sedan and SUV type of cars are selected.

E. Additional results

E.1. The BrnoCompSpeed dataset

We report additional results for the BrnoCompSpeed dataset [6]. Figure A.8 shows the cumulative histogram of relative speed errors, which was omitted from the main paper for the sake of space. Table 1 reports additional comparisons with OptInOpt [1], PlaneCalib [2] and FullACC++ [5] in term of root mean-squared errors (RMSE) on distances between pairs of points on the road plane with corresponding manual annotations (see [2] for details).

Overall our proposed methods perform slightly inferior to those other methods, but still largely better than FullACC [3]. In term of computation time, our learned-method is at least 2 orders of magnitude faster than any of the other methods. Again, we find remarkable that our method yields an RMSE of just 5% while being trained solely from synthetic data using off-the-shelf 3D car models. This is in contrast with OptInOpt [1], PlaneCalib [2], and FullACC++ [5] where in each case, multiple deep networks (for *e.g.* vehicle detection, vehicle categorization, landmark localization) and exact 3D CAD models of the cars (exploiting the fact that most cars in the BrnoCompSpeed dataset, Czech Republic, are made by Skoda) with millimeter precise landmark localization (see Fig. 5 in the PlaneCalib paper [2]) were used to calibrate the system.

E.2. Qualitative results

We show some results of the calibration output by FullACC++ [5] and our proposed method Learned-RANSAC in Fig. A.7 on the CCTV dataset. The final homography is represented as a square grid overlaid onto the road. For the sake of visualization, we manually rotate the homography in the road coordinate space in order to roughly align the first vanishing point with the road direction. Indeed our method always outputs a homography with the first vanish-

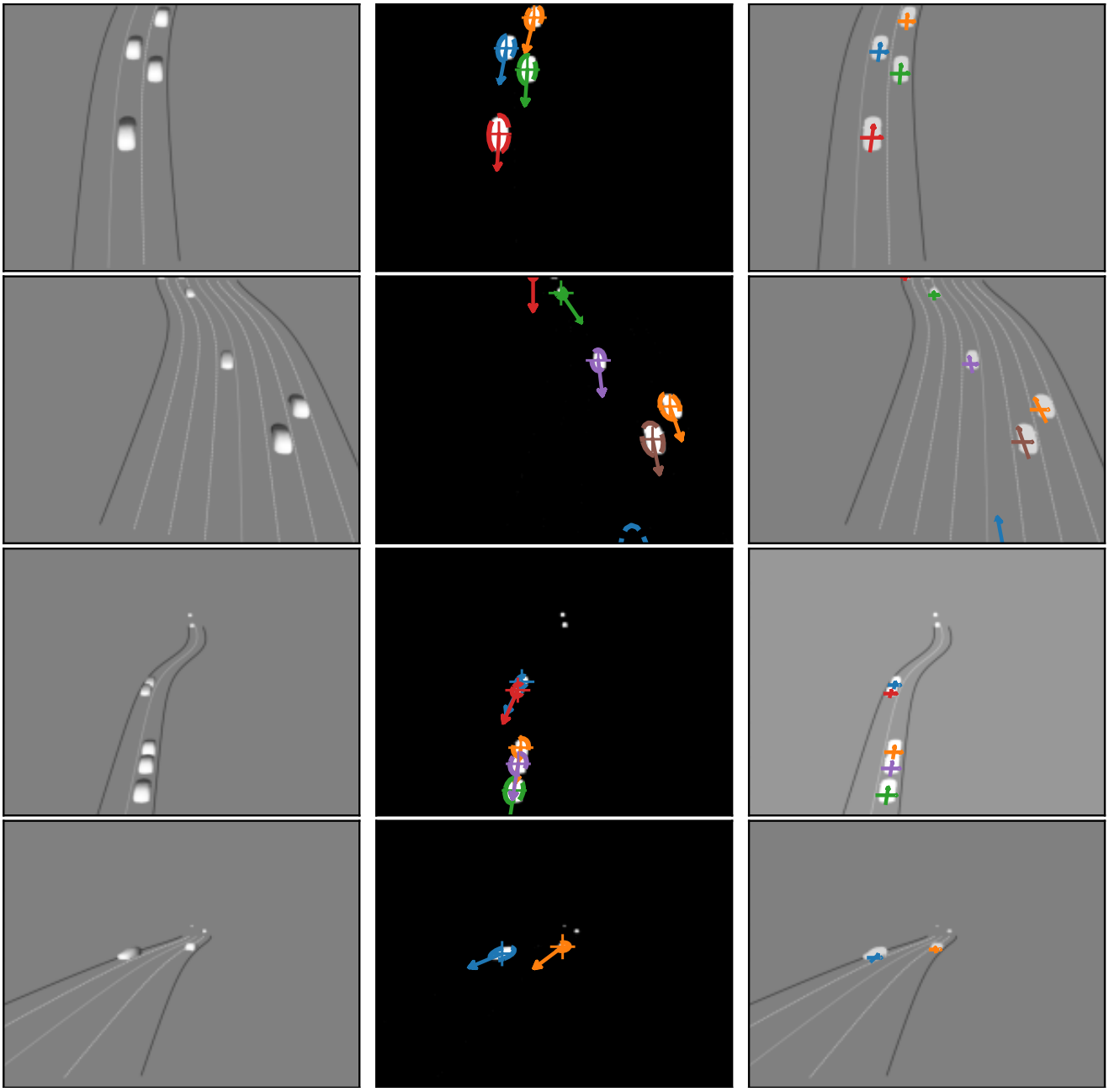

Figure A.4. Illustration of the Jacobian prediction on synthetic scenes. *Left*: synthetic scenes. *Middle*: binary car masks overlaid with their embedding representations (*i.e*. ellipses and normalized motion (arrows)) input to the network. *Right*: Corresponding predicted Jacobians $\hat{J} = [\hat{J}_1, \hat{J}_2]$ displayed as arrows ($\hat{J}_1$ is purely horizontal due to Eq. (C.3), $\hat{J}_2$ is enlarged 2x for the sake of visualization).

ing point at infinity, which we find quite impractical to visualize. This rotation has no effect on speed estimation. Note that FullACC++ [5] already performs this alignment by design of the method.

In many cases, FullACC++ [5] correctly estimates the first vanishing point but fails for the second one, resulting in an erroneous focal and subsequent depth estimates. This is clearly visible in most scenes of Fig. A.7, where the grid width is about the same for both methods but the depth strongly differs. Sometimes, FullACC++ completely collapses, like in the second and third row of Fig. A.7. In the first case, failure is due to the vehicle instances being too small (hence the estimation of the 2nd vanishing point completely fails). In the second case, it is due to the road being not straight, which causes a failure in the estimation of both vanishing points. In contrast, the proposed method

Figure A.5. Per-track detection sampling and Jacobian prediction for the learned method. *Left*: observed track (10 detections regularly sampled) and extracted representation r_b (ellipse and motion vector) overlaid for each detection. *Right*: Corresponding predicted Jacobians $\hat{J} = [\hat{J}_1, \hat{J}_2]$ displayed as arrows ($\hat{J}_1$ is purely horizontal due to Eq. (C.3), $\hat{J}_2$ is enlarged 2x for the sake of visualization).

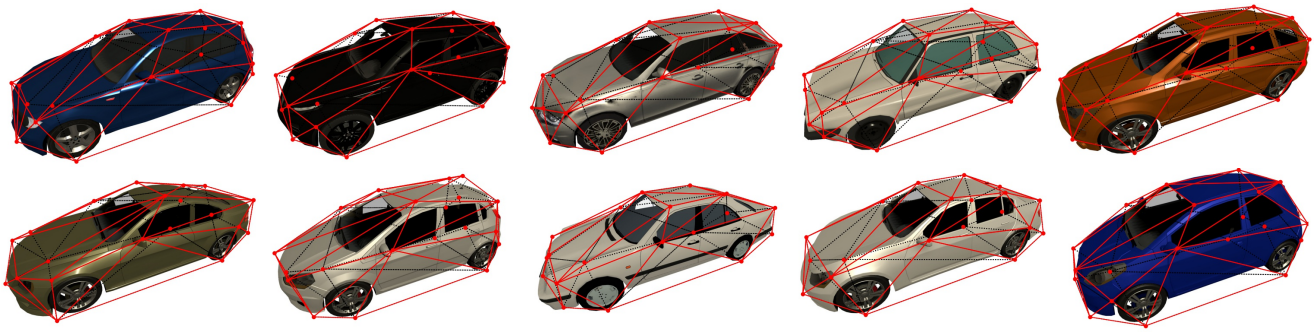

Figure A.6. Simplified 3D meshes (in red) for each of the 10 Car models.

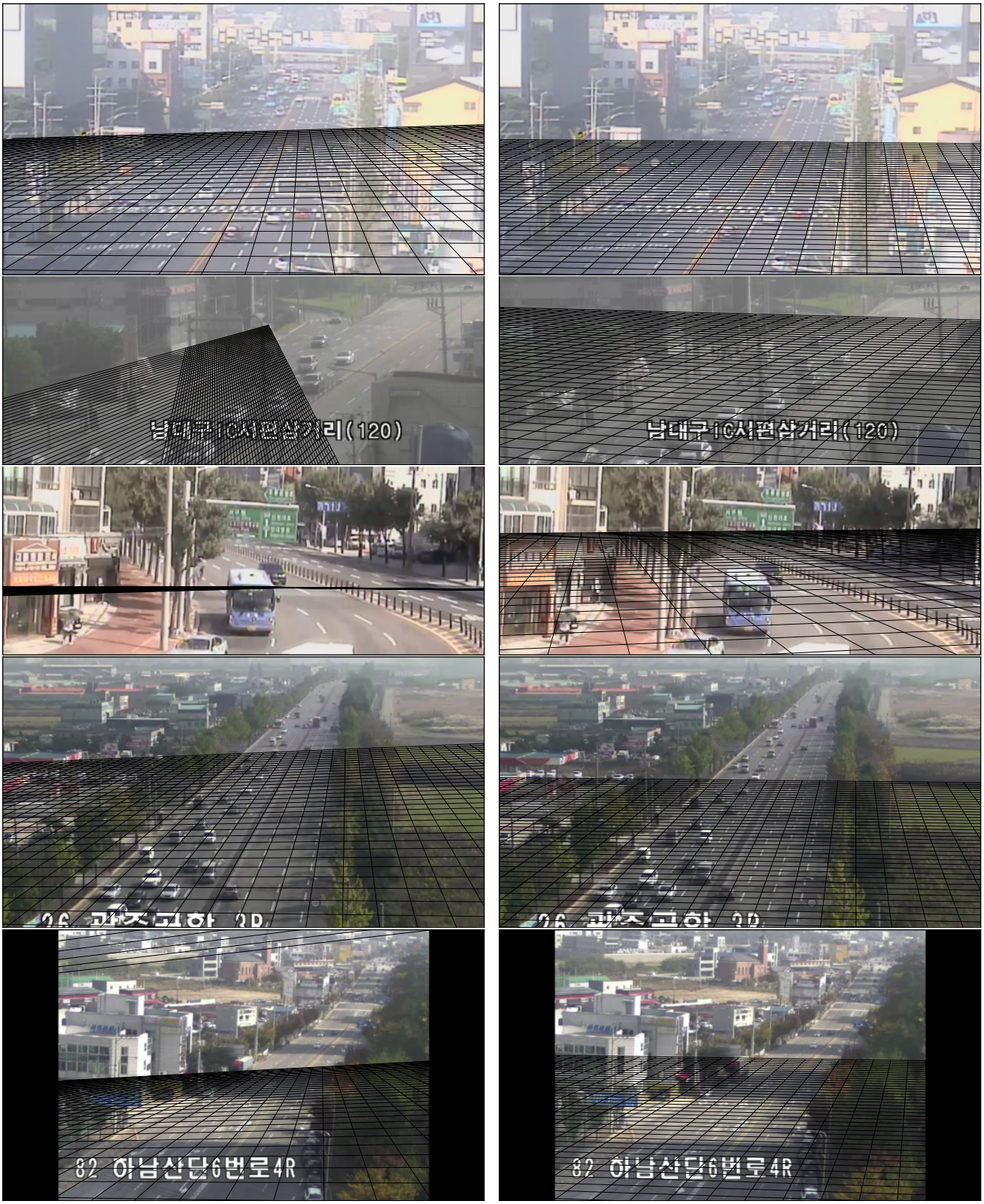

Figure A.7. Visual comparison of the homographies estimated using FullACC++† [5] (left column) and our Learned-RANSAC (right column). The grid size correspond to squares of 3.5 meters. Notice how depth is poorly estimated using FullACC++†, when it simply does not completely collapse (2nd and 3rd row) due to low resolution or non-straightness of the road.

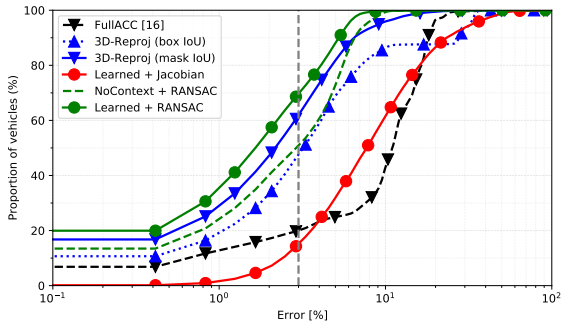

Figure A.8. Cumulative histogram of relative errors for the BrnoCompSpeed dataset [6]. The vertical dashed line indicates the 3% error threshold.

	Abs error (km/h)		Rel error (%)		RMSE	Time (s)
	avg	median	avg	median	(%)	avg
3D-reproj (box IoU)	5.70	2.85	7.04	3.61	8.27	1.49 K
3D-reproj (mask IoU)	2.84	2.03	3.46	2.58	6.87	2.84 K
Learned+Jacobian	9.21	6.47	11.51	8.12	-	15.9
NoContext+RANSAC	3.01	2.59	3.69	3.31	5.81	**2.5**
Learned+RANSAC	**2.15**	**1.60**	**2.65**	**2.07**	5.05	2.9
FullACC [3]	8.59	8.45	10.89	11.41	13.9	1.5 K
FullACC++ [5]*	**1.10**	**0.97**	**1.39**	**1.22**	-	-
FullACC++† [5]	2.39	1.72	3.03	2.17	3.46	2.1 K
OptInOpt [1]	-	-	-	-	3.01	325 K
PlaneCalib [2]	-	-	-	-	3.65	266

Table 1. Results for the BrnoCompSpeed dataset (split A). Note that FullACC++ [5] 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. FullACC++† is our re-implementation using code snippets shared by the authors.

naturally handles non-straight roads and small instances by design.

References

[1] 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. 3, 7

[2] 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, 3, 7

[3] 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, 3, 7

[4] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit*. 2006. 3

[5] 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, 3, 4, 6, 7

[6] 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. 3, 7

Figure A.9. Snapshots of all clips from the CCTV dataset.