Vision meets Robotics: The KITTI Dataset

Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun

***Abstract*—We present a novel dataset captured from a VW station wagon for use in mobile robotics and autonomous driving research. In total, we recorded 6 hours of traffic scenarios at 10-100 Hz using a variety of sensor modalities such as high-resolution color and grayscale stereo cameras, a Velodyne 3D laser scanner and a high-precision GPS/IMU inertial navigation system. The scenarios are diverse, capturing real-world traffic situations and range from freeways over rural areas to inner-city scenes with many static and dynamic objects. Our data is calibrated, synchronized and timestamped, and we provide the rectified and raw image sequences. Our dataset also contains object labels in the form of 3D tracklets and we provide online benchmarks for stereo, optical flow, object detection and other tasks. This paper describes our recording platform, the data format and the utilities that we provide.**

***Index Terms*—dataset, autonomous driving, mobile robotics, field robotics, computer vision, cameras, laser, GPS, benchmarks, stereo, optical flow, SLAM, object detection, tracking, KITTI**

I. Introduction

The KITTI dataset has been recorded from a moving platform (Fig. 1) while driving in and around Karlsruhe, Germany (Fig. 2). It includes camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system. The main purpose of this dataset is to push forward the development of computer vision and robotic algorithms targeted to autonomous driving [1]–[7]. While our introductory paper [8] mainly focuses on the benchmarks, their creation and use for evaluating state-of-the-art computer vision methods, here we complement this information by providing technical details on the raw data itself. We give precise instructions on how to access the data and comment on sensor limitations and common pitfalls. The dataset can be downloaded from http://www.cvlibs.net/datasets/kitti. For a review on related work, we refer the reader to [8].

II. Sensor Setup

Our sensor setup is illustrated in Fig. 3:

- 2 × PointGray Flea2 grayscale cameras (FL2-14S3M-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter
- 2 × PointGray Flea2 color cameras (FL2-14S3C-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter
- 4 × Edmund Optics lenses, 4mm, opening angle $\sim 90°$, vertical opening angle of region of interest (ROI) $\sim 35°$
- 1 × Velodyne HDL-64E rotating 3D laser scanner, 10 Hz, 64 beams, 0.09 degree angular resolution, 2 cm distance accuracy, collecting $\sim$ 1.3 million points/second, field of view: 360° horizontal, 26.8° vertical, range: 120 m

A. Geiger, P. Lenz and C. Stiller are with the Department of Measurement and Control Systems, Karlsruhe Institute of Technology, Germany. Email: {geiger,lenz,stiller}@kit.edu

R. Urtasun is with the Toyota Technological Institute at Chicago, USA. Email: rurtasun@ttic.edu

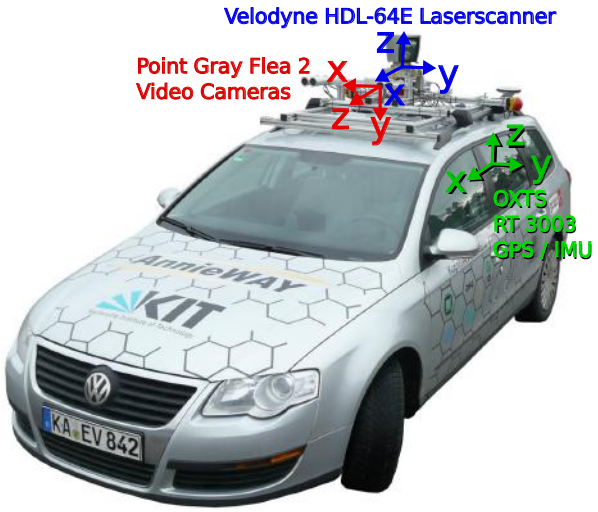

Fig. 1. **Recording Platform.** Our VW Passat station wagon is equipped with four video cameras (two color and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system.

- 1 × OXTS RT3003 inertial and GPS navigation system, 6 axis, 100 Hz, L1/L2 RTK, resolution: 0.02m / 0.1°

Note that the color cameras lack in terms of resolution due to the Bayer pattern interpolation process and are less sensitive to light. This is the reason why we use two stereo camera rigs, one for grayscale and one for color. The baseline of both stereo camera rigs is approximately 54 cm. The trunk of our vehicle houses a PC with two six-core Intel XEON X5650 processors and a shock-absorbed RAID 5 hard disk storage with a capacity of 4 Terabytes. Our computer runs Ubuntu Linux (64 bit) and a real-time database [9] to store the incoming data streams.

III. Dataset

The *raw data* described in this paper can be accessed from http://www.cvlibs.net/datasets/kitti and contains $\sim 25\%$ of our overall recordings. The reason for this is that primarily data with 3D tracklet annotations has been put online, though we will make more data available upon request. Furthermore, we have removed all sequences which are part of our benchmark test sets. The raw data set is divided into the categories 'Road', 'City', 'Residential', 'Campus' and 'Person'. Example frames are illustrated in Fig. 5. For each sequence, we provide the raw data, object annotations in form of 3D bounding box tracklets and a calibration file, as illustrated in Fig. 4. Our recordings have taken place on the 26th, 28th, 29th, 30th of September and on the 3rd of October 2011 during daytime. The total size of the provided data is 180 GB.

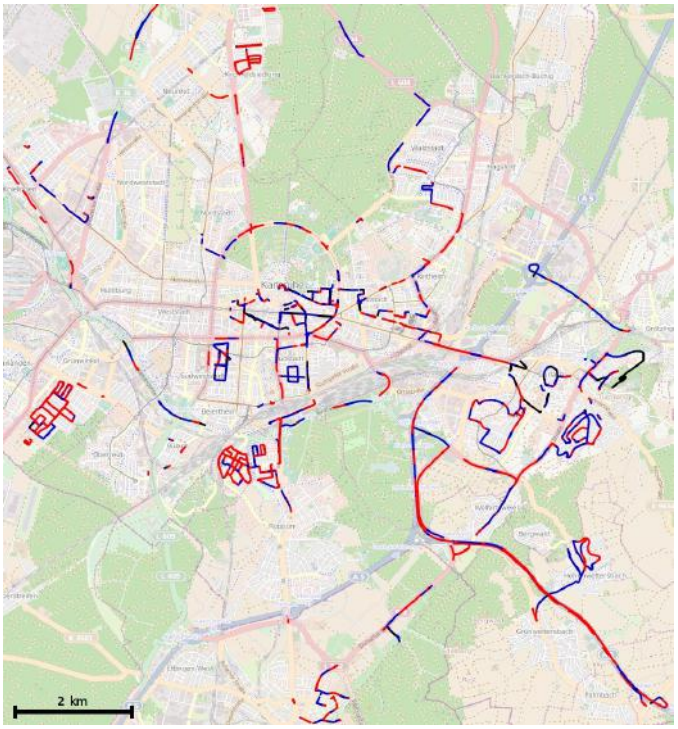

Fig. 2. **Recording Zone.** This figure shows the GPS traces of our recordings in the metropolitan area of Karlsruhe, Germany. Colors encode the GPS signal quality: Red tracks have been recorded with highest precision using RTK corrections, blue denotes the absence of correction signals. The black runs have been excluded from our data set as no GPS signal has been available.

A. Data Description

All sensor readings of a sequence are zipped into a single file named `date_drive.zip`, where `date` and `drive` are placeholders for the recording date and the sequence number. The directory structure is shown in Fig. 4. Besides the raw recordings ('raw data'), we also provide post-processed data ('synced data'), i.e., rectified and synchronized video streams, on the dataset website.

Timestamps are stored in `timestamps.txt` and per-frame sensor readings are provided in the corresponding data sub-folders. Each line in `timestamps.txt` is composed of the date and time in hours, minutes and seconds. As the Velodyne laser scanner has a 'rolling shutter', three timestamp files are provided for this sensor, one for the start position (`timestamps_start.txt`) of a spin, one for the end position (`timestamps_end.txt`) of a spin, and one for the time, where the laser scanner is facing forward and triggering the cameras (`timestamps.txt`). The data format in which each sensor stream is stored is as follows:

a) Images: Both, color and grayscale images are stored with loss-less compression using 8-bit PNG files. The engine hood and the sky region have been cropped. To simplify working with the data, we also provide rectified images. The size of the images after rectification depends on the calibration parameters and is ~ 0.5 Mpx on average. The original images before rectification are available as well.

b) OXTS (GPS/IMU): For each frame, we store 30 different GPS/IMU values in a text file: The geographic coordinates including altitude, global orientation, velocities, accelerations, angular rates, accuracies and satellite information. Accelerations and angular rates are both specified using two coordinate systems, one which is attached to the vehicle body (x, y, z) and one that is mapped to the tangent plane of the earth surface at that location (f, l, u). From time to time we encountered short ($\sim$ 1 second) communication outages with the OXTS device for which we interpolated all values linearly and set the last 3 entries to '-1' to indicate the missing information. More details are provided in `dataformat.txt`. Conversion utilities are provided in the development kit.

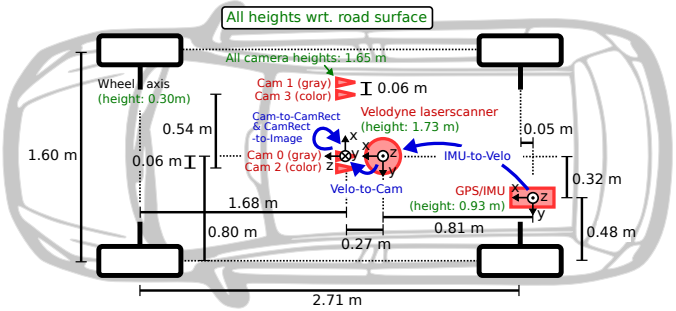

Fig. 3. **Sensor Setup.** This figure illustrates the dimensions and mounting positions of the sensors (red) with respect to the vehicle body. Heights above ground are marked in green and measured with respect to the road surface. Transformations between sensors are shown in blue.

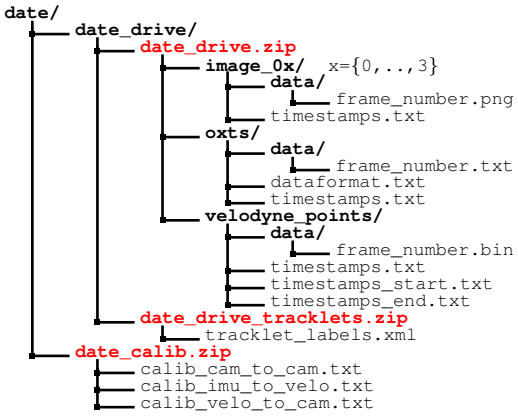

Fig. 4. **Structure of the provided Zip-Files** and their location within a global file structure that stores all KITTI sequences. Here, 'date' and 'drive' are placeholders, and 'image_0x' refers to the 4 video camera streams.

c) Velodyne: For efficiency, the Velodyne scans are stored as floating point binaries that are easy to parse using the C++ or MATLAB code provided. Each point is stored with its (x, y, z) coordinate and an additional reflectance value (r). While the number of points per scan is not constant, on average each file/frame has a size of $\sim$ 1.9 MB which corresponds to $\sim 120,000$ 3D points and reflectance values. Note that the Velodyne laser scanner rotates continuously around its vertical axis (counter-clockwise), which can be taken into account using the timestamp files.

B. Annotations

For each dynamic object within the reference camera's field of view, we provide annotations in the form of 3D bounding box tracklets, represented in Velodyne coordinates. We define the classes 'Car', 'Van', 'Truck', 'Pedestrian', 'Person (sitting)', 'Cyclist', 'Tram' and 'Misc' (e.g., Trailers, Segways). The tracklets are stored in `date_drive_tracklets.xml`.

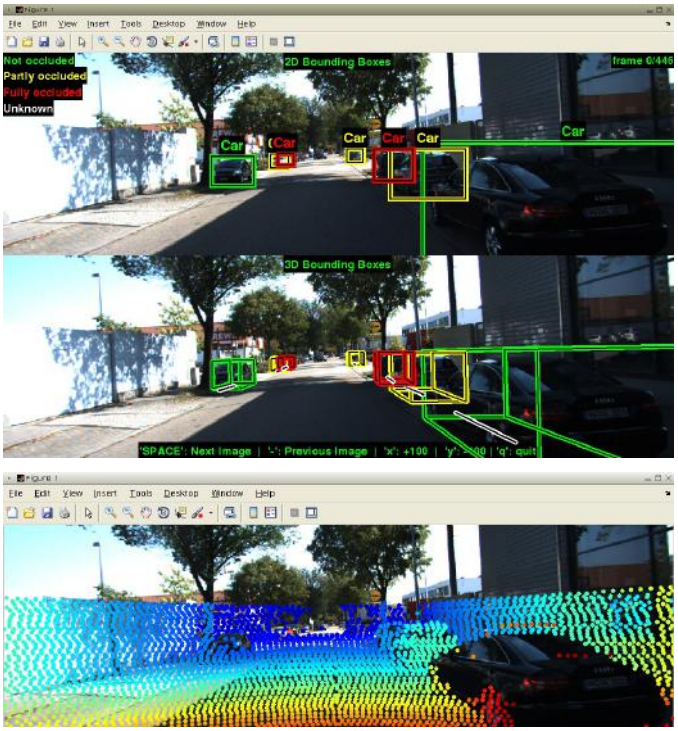

Fig. 6. **Development kit.** Working with tracklets (top), Velodyne point clouds (bottom) and their projections onto the image plane is demonstrated in the MATLAB development kit which is available from the KITTI website.

Each object is assigned a class and its 3D size (height, width, length). For each frame, we provide the object's translation and rotation in 3D, as illustrated in Fig. 7. Note that we only provide the yaw angle, while the other two angles are assumed to be close to zero. Furthermore, the level of occlusion and truncation is specified. The development kit contains C++/MATLAB code for reading and writing tracklets using the `boost::serialization`[1] library.

To give further insights into the properties of our dataset, we provide statistics for all sequences that contain annotated objects. The total number of objects and the object orientations for the two predominant classes 'Car' and 'Pedestrian' are shown in Fig. 8. For each object class, the number of object labels per image and the length of the captured sequences is shown in Fig. 9. The egomotion of our platform recorded by the GPS/IMU system as well as statistics about the sequence length and the number of objects are shown in Fig. 10 for the whole dataset and in Fig. 11 per street category.

C. Development Kit

The *raw data development kit* provided on the KITTI website[2] contains MATLAB demonstration code with C++ wrappers and a `readme.txt` file which gives further details. Here, we will briefly discuss the most important features. Before running the scripts, the mex wrapper `readTrackletsMex.cpp` for reading tracklets into MATLAB structures and cell arrays needs to be built using the script `make.m`. It wraps the file `tracklets.h` from the

[1]http://www.boost.org

[2]http://www.cvlibs.net/datasets/kitti/raw_data.php

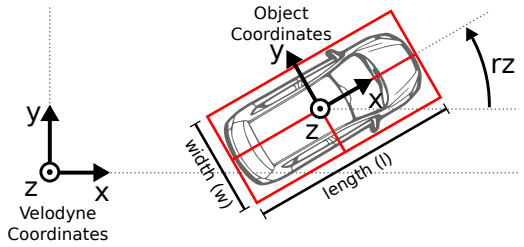

Fig. 7. **Object Coordinates.** This figure illustrates the coordinate system of the annotated 3D bounding boxes with respect to the coordinate system of the 3D Velodyne laser scanner. In z-direction, the object coordinate system is located at the bottom of the object (contact point with the supporting surface).

`cpp` folder which holds the tracklet object for serialization. This file can also be directly interfaced with when working in a C++ environment.

The script `run_demoTracklets.m` demonstrates how 3D bounding box tracklets can be read from the XML files and projected onto the image plane of the cameras. The projection of 3D Velodyne point clouds into the image plane is demonstrated in `run_demoVelodyne.m`. See Fig. 6 for an illustration.

The script `run_demoVehiclePath.m` shows how to read and display the 3D vehicle trajectory using the GPS/IMU data. It makes use of `convertOxtsToPose()`, which takes as input GPS/IMU measurements and outputs the 6D pose of the vehicle in Euclidean space. For this conversion we make use of the Mercator projection [10]

$$x = s \times r \times \frac{\pi\, lon}{180} \quad (1)$$

$$y = s \times r \times \log\left(\tan\left(\frac{\pi(90+lat)}{360}\right)\right) \quad (2)$$

with earth radius $r \approx 6378137$ meters, scale $s = \cos\left(\frac{lat_0 \times \pi}{180}\right)$, and (lat, lon) the geographic coordinates. lat_0 denotes the latitude of the first frame's coordinates and uniquely determines the Mercator scale.

The function `loadCalibrationCamToCam()` can be used to read the intrinsic and extrinsic calibration parameters of the four video sensors. The other 3D rigid body transformations can be parsed with `loadCalibrationRigid()`.

D. Benchmarks

In addition to the raw data, our KITTI website hosts evaluation benchmarks for several computer vision and robotic tasks such as stereo, optical flow, visual odometry, SLAM, 3D object detection and 3D object tracking. For details about the benchmarks and evaluation metrics we refer the reader to [8].

IV. Sensor Calibration

We took care that all sensors are carefully synchronized and calibrated. To avoid drift over time, we calibrated the sensors every day after our recordings. Note that even though the sensor setup hasn't been altered in between, numerical differences are possible. The coordinate systems are defined as illustrated in Fig. 1 and Fig. 3, i.e.:

- Camera: x = right, y = down, z = forward
- Velodyne: x = forward, y = left, z = up

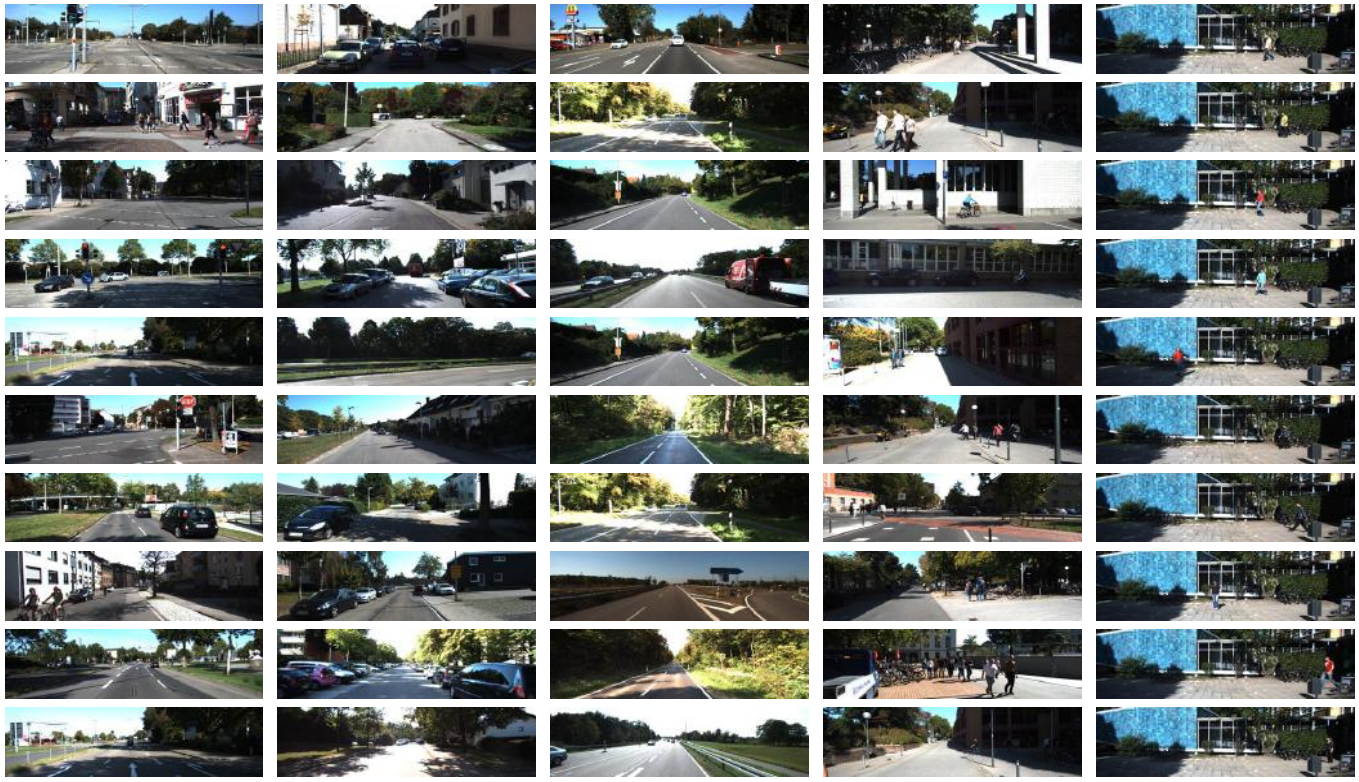

Fig. 5. **Examples from the KITTI dataset.** This figure demonstrates the diversity in our dataset. The left color camera image is shown.

- GPS/IMU: x = forward, y = left, z = up

Notation: In the following, we write scalars in lower-case letters (a), vectors in bold lower-case ($\mathbf{a}$) and matrices using bold-face capitals ($\mathbf{A}$). 3D rigid-body transformations which take points from coordinate system a to coordinate system b will be denoted by $\mathbf{T}_a^b$, with $\mathbf{T}$ for 'transformation'.

A. Synchronization

In order to synchronize the sensors, we use the timestamps of the Velodyne 3D laser scanner as a reference and consider each spin as a *frame*. We mounted a reed contact at the bottom of the continuously rotating scanner, triggering the cameras when facing forward. This minimizes the differences in the range and image observations caused by dynamic objects. Unfortunately, the GPS/IMU system cannot be synchronized that way. Instead, as it provides updates at 100 Hz, we collect the information with the closest timestamp to the laser scanner timestamp for a particular frame, resulting in a worst-case time difference of 5 ms between a GPS/IMU and a camera/Velodyne data package. Note that all timestamps are provided such that positioning information at any time can be easily obtained via interpolation. All timestamps have been recorded on our host computer using the system clock.

B. Camera Calibration

For calibrating the cameras intrinsically and extrinsically, we use the approach proposed in [11]. Note that all camera centers are aligned, i.e., they lie on the same x/y-plane. This is important as it allows us to rectify all images jointly.

The calibration parameters for each day are stored in row-major order in `calib_cam_to_cam.txt` using the following notation:

- $\mathbf{s}^{(i)} \in \mathbb{N}^2$ ..... original image size (1392×512)
- $\mathbf{K}^{(i)} \in \mathbb{R}^{3\times3}$ ..... calibration matrices (unrectified)
- $\mathbf{d}^{(i)} \in \mathbb{R}^5$ ..... distortion coefficients (unrectified)
- $\mathbf{R}^{(i)} \in \mathbb{R}^{3\times3}$ ..... rotation from camera 0 to camera i
- $\mathbf{t}^{(i)} \in \mathbb{R}^{1\times3}$ ..... translation from camera 0 to camera i
- $\mathbf{s}_{rect}^{(i)} \in \mathbb{N}^2$ ..... image size after rectification
- $\mathbf{R}_{rect}^{(i)} \in \mathbb{R}^{3\times3}$ ..... rectifying rotation matrix
- $\mathbf{P}_{rect}^{(i)} \in \mathbb{R}^{3\times4}$ ..... projection matrix after rectification

Here, $i \in \{0, 1, 2, 3\}$ is the camera index, where 0 represents the left grayscale, 1 the right grayscale, 2 the left color and 3 the right color camera. Note that the variable definitions are compliant with the OpenCV library, which we used for warping the images. When working with the synchronized and rectified datasets only the variables with *rect*-subscript are relevant. Note that due to the pincushion distortion effect the images have been cropped such that the size of the rectified images is smaller than the original size of 1392×512 pixels.

The projection of a 3D point $\mathbf{x} = (x, y, z, 1)^T$ in *rectified (rotated) camera coordinates* to a point $\mathbf{y} = (u, v, 1)^T$ in the i'th camera image is given as

$$\mathbf{y} = \mathbf{P}_{rect}^{(i)} \, \mathbf{x} \tag{3}$$

with

$$\mathbf{P}_{rect}^{(i)} = \begin{pmatrix} f_u^{(i)} & 0 & c_u^{(i)} & -f_u^{(i)} b_x^{(i)} \\ 0 & f_v^{(i)} & c_v^{(i)} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4}$$

the i'th projection matrix. Here, $b_x^{(i)}$ denotes the baseline (in meters) with respect to reference camera 0. Note that in order to project a 3D point $\mathbf{x}$ in *reference camera coordinates* to a point $\mathbf{y}$ on the i'th image plane, the rectifying rotation matrix of the reference camera $\mathbf{R}_{rect}^{(0)}$ must be considered as well:

$$\mathbf{y} = \mathbf{P}_{rect}^{(i)} \; \mathbf{R}_{rect}^{(0)} \; \mathbf{x} \tag{5}$$

Here, $\mathbf{R}_{rect}^{(0)}$ has been expanded into a 4×4 matrix by appending a fourth zero-row and column, and setting $\mathbf{R}_{rect}^{(0)}(4,4) = 1$.

C. Velodyne and IMU Calibration

We have registered the Velodyne laser scanner with respect to the reference camera coordinate system (camera 0) by initializing the rigid body transformation using [11]. Next, we optimized an error criterion based on the Euclidean distance of 50 manually selected correspondences and a robust measure on the disparity error with respect to the 3 top performing stereo methods in the KITTI stereo benchmark [8]. The optimization was carried out using Metropolis-Hastings sampling.

The rigid body transformation from Velodyne coordinates to camera coordinates is given in `calib_velo_to_cam.txt`:

- $\mathbf{R}_{velo}^{cam} \in \mathbb{R}^{3\times3}$ .... rotation matrix: velodyne $\rightarrow$ camera
- $\mathbf{t}_{velo}^{cam} \in \mathbb{R}^{1\times3}$ ... translation vector: velodyne $\rightarrow$ camera

Using

$$\mathbf{T}_{velo}^{cam} = \begin{pmatrix} \mathbf{R}_{velo}^{cam} & \mathbf{t}_{velo}^{cam} \\ 0 & 1 \end{pmatrix} \tag{6}$$

a 3D point $\mathbf{x}$ in Velodyne coordinates gets projected to a point $\mathbf{y}$ in the i'th camera image as

$$\mathbf{y} = \mathbf{P}_{rect}^{(i)} \; \mathbf{R}_{rect}^{(0)} \; \mathbf{T}_{velo}^{cam} \; \mathbf{x} \tag{7}$$

For registering the IMU/GPS with respect to the Velodyne laser scanner, we first recorded a sequence with an '∞'-loop and registered the (untwisted) point clouds using the Point-to-Plane ICP algorithm. Given two trajectories this problem corresponds to the well-known hand-eye calibration problem which can be solved using standard tools [12]. The rotation matrix $\mathbf{R}_{imu}^{velo}$ and the translation vector $\mathbf{t}_{imu}^{velo}$ are stored in `calib_imu_to_velo.txt`. A 3D point $\mathbf{x}$ in IMU/GPS coordinates gets projected to a point $\mathbf{y}$ in the i'th image as

$$\mathbf{y} = \mathbf{P}_{rect}^{(i)} \; \mathbf{R}_{rect}^{(0)} \; \mathbf{T}_{velo}^{cam} \; \mathbf{T}_{imu}^{velo} \; \mathbf{x} \tag{8}$$

V. Summary and Future Work

In this paper, we have presented a calibrated, synchronized and rectified autonomous driving dataset capturing a wide range of interesting scenarios. We believe that this dataset will be highly useful in many areas of robotics and computer vision. In the future we plan on expanding the set of available sequences by adding additional 3D object labels for currently unlabeled sequences and recording new sequences, for example in difficult lighting situations such as at night, in tunnels, or in the presence of fog or rain. Furthermore, we plan on extending our benchmark suite by novel challenges. In particular, we will provide pixel-accurate semantic labels for many of the sequences.

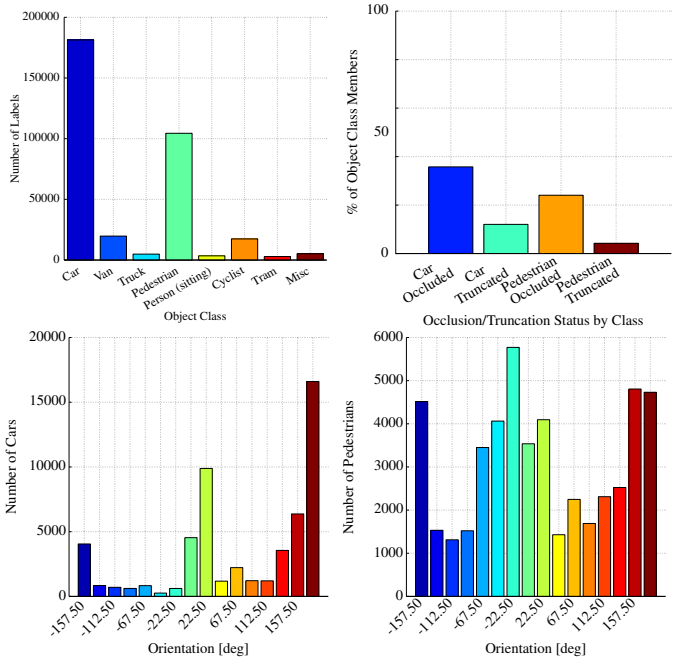

Fig. 8. **Object Occurrence and Orientation Statistics of our Dataset.** This figure shows the different types of objects occurring in our sequences (top) and the orientation histograms (bottom) for the two most predominant categories 'Car' and 'Pedestrian'.

References

[1] G. Singh and J. Kosecka, "Acquiring semantics induced topology in urban environments," in *ICRA*, 2012.

[2] R. Paul and P. Newman, "FAB-MAP 3D: Topological mapping with spatial and visual appearance," in *ICRA*, 2010.

[3] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele, "Monocular visual scene understanding: Understanding multi-object traffic scenes," *PAMI*, 2012.

[4] D. Pfeiffer and U. Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in *IV*, 2010.

[5] A. Geiger, M. Lauer, and R. Urtasun, "A generative model for 3d urban scene understanding from movable platforms," in *CVPR*, 2011.

[6] A. Geiger, C. Wojek, and R. Urtasun, "Joint 3d estimation of objects and scene layout," in *NIPS*, 2011.

[7] M. A. Brubaker, A. Geiger, and R. Urtasun, "Lost! leveraging the crowd for probabilistic visual self-localization." in *CVPR*, 2013.

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

[9] M. Goebl and G. Faerber, "A real-time-capable hard- and software architecture for joint image and knowledge processing in cognitive automobiles," in *IV*, 2007.

[10] P. Osborne, "The mercator projections," 2008. [Online]. Available: http://mercator.myzen.co.uk/mercator.pdf

[11] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "A toolbox for automatic calibration of range and camera sensors using a single shot," in *ICRA*, 2012.

[12] R. Horaud and F. Dornaika, "Hand-eye calibration," *IJRR*, vol. 14, no. 3, pp. 195–210, 1995.

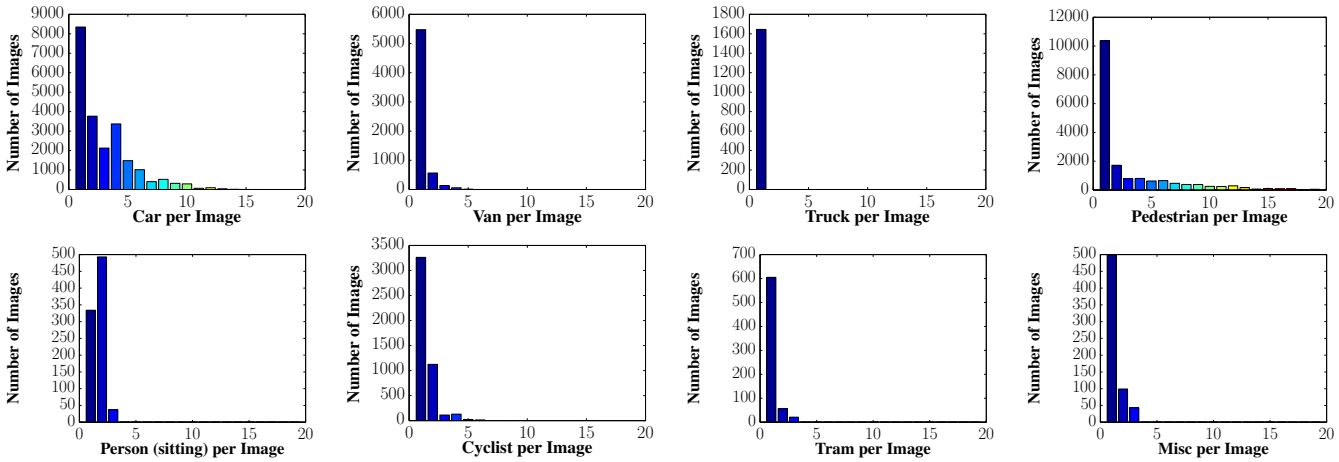

Fig. 9. **Number of Object Labels per Class and Image.** This figure shows how often an object occurs in an image. Since our labeling efforts focused on cars and pedestrians, these are the most predominant classes here.

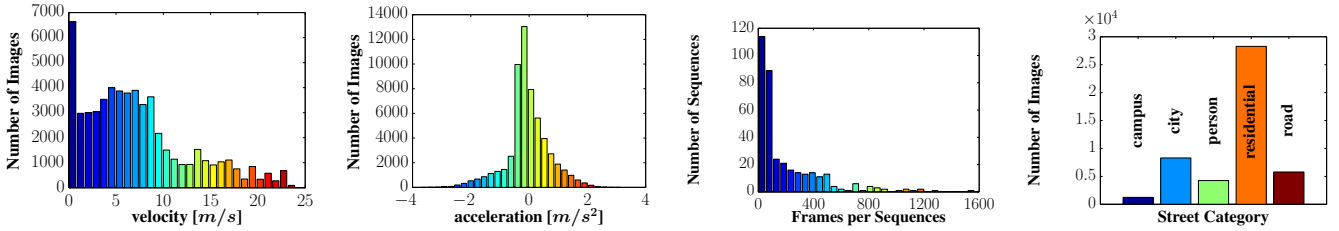

Fig. 10. **Egomotion, Sequence Count and Length.** This figure show (from-left-to-right) the egomotion (velocity and acceleration) of our recording platform for the whole dataset. Note that we excluded sequences with a purely static observer from these statistics. The length of the available sequences is shown as a histogram counting the number of frames per sequence. The rightmost figure shows the number of frames/images per scene category.

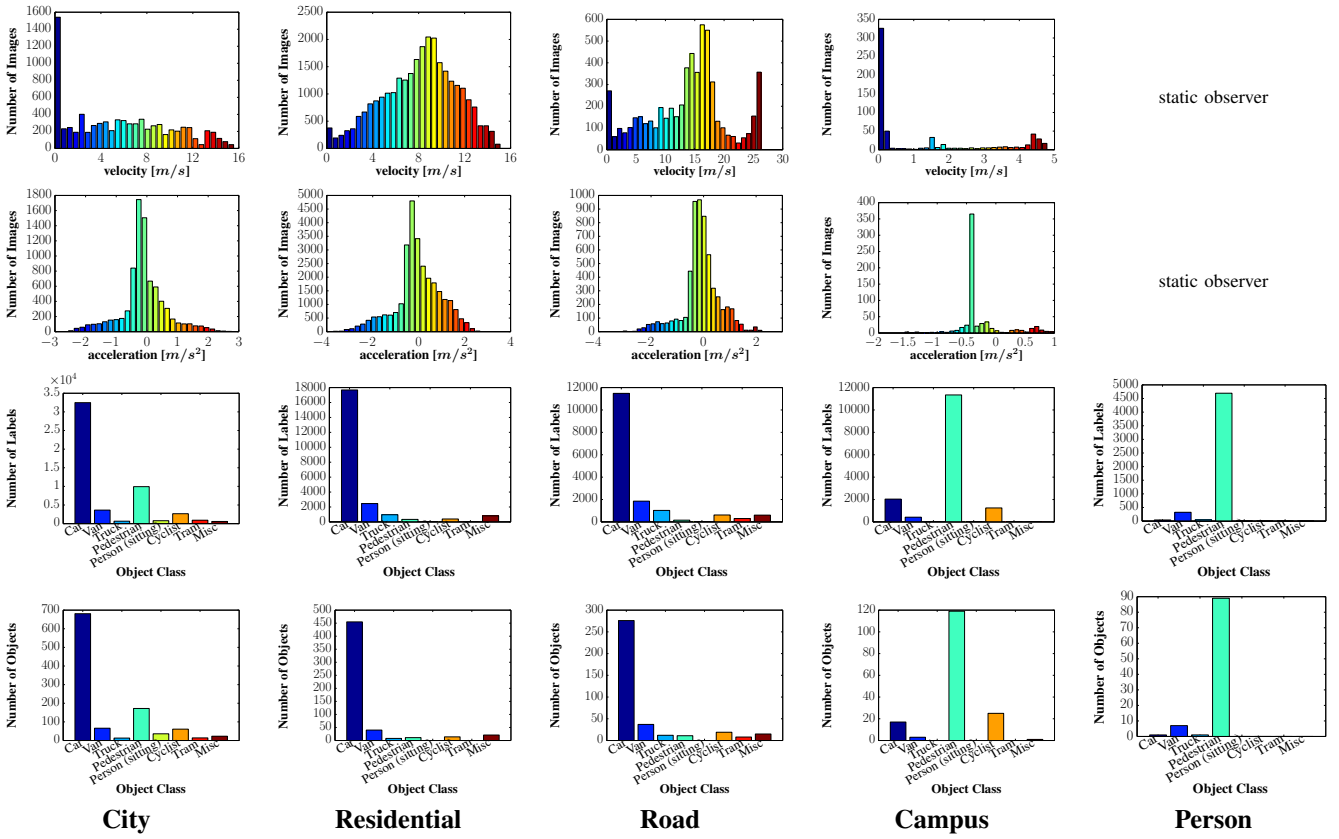

Fig. 11. **Velocities, Accelerations and Number of Objects per Object Class.** For each scene category we show the acceleration and velocity of the mobile platform as well as the number of labels and objects per class. Note that sequences with a purely static observer have been excluded from the velocity and acceleration histograms as they don't represent natural driving behavior. The category 'Person' has been recorded from a static observer.